A Gentle Introduction to Statistical Computing Using R

Amiya Ranjan Bhowmick

February 19, 2025

Table of contents

Preface

Introduction

The Department of Mathematics, Institute of Chemical Technology, Mumbai has offered the Multidisciplinary Minor (MDM) Programme in `Machine Learning and Artificial Intelligence` under the National Education Policy (NEP 2020). This material has been developed during the lectures of the course Statistical Computing (MAT 1501). This course is designed to give some fundamental statistical ideas to the students so that they can grasp deeper concepts in Machine Learning and Deep Learning courses in the upcoming semesters. All the codes in this study material were written live in the classroom to demonstrate various statistical concepts. Yes, you read that correctly—the codes were developed in real-time during the sessions. Later, some refinements were made by incorporating mathematical expressions and adding general explanations of the concepts used. Based on my experience, I find Quarto to be an exceptionally user-friendly dynamic document generation platform that supports multiple programming languages.

Additionally, I would like to emphasize that I have rarely used external packages for demonstration. Instead, I have primarily relied on basic loops and matrices to write programs. Over time, I have observed that students often focus on memorizing package names and perceive R/Python as magical software, treating them as black boxes. In this document, you will see that we performed an entire regression analysis using matrix notation before later implementing it with built-in functions.

This document is not intended to make you an expert in machine learning; rather, it aims to help you grasp fundamental concepts in Data Science. My goal is for students to understand the underlying ideas and make informed decisions when selecting appropriate algorithms in future courses, rather than blindly following R/Python instructions. Typically, these concepts are taught through PowerPoint presentations, visually appealing slides, or engaging talks. I initially considered taking a similar approach, but the students' eagerness to explore and understand the concepts in depth inspired me to demonstrate live programming in R. Ultimately, it was the students who guided this teaching approach, and I found that it worked quite well. Codes are written live, therefore, they may not be efficient and I am sure that there will be better and smarter way to write code. But, the primarily goal is to understand the statistical ideas, not to learn programming using R.

Rationale

This course is a foundation course covering major concepts from Probability and statistical estimation theory for the Undergraduate Engineering students. Introduced concepts will be useful in understanding the concepts related to Data Science, Machine Learning, and Deep Learning having wider applications in various engineering disciplines.

Prerequisites

Basic linear algebra, differential calculus, basic probability theory, knowledge of conditional probability and Bayes theorem.

Acknowledgement

A big thank you to the undergraduate engineering students who have chosen the Multidisciplinary Minor (MDM) Degree in Machine Learning and Artificial Intelligence (ML and AI), offered by the Department of Mathematics under NEP 2020. A special shoutout to Prathamesh, Agastya, Venkatesh, Arya Rane, Ajinkya, Arya Shimpi, Arya Kale, Reenesh, and many others. Interestingly, there are three students named Arya, and in class, I often say that a question will be answered by one Arya (selected with probability 1/3)In addition, special thanks to the M.Sc. in Engineering Mathematics students for their support in the classroom in expanding this material with more advanced examples tailored for the M.Sc. level. A special mention to Hiloni, Sangeeta, Vaishnavi and others for their invaluable support.

And certainly thanks to my Ph.D. student Dipali for introducing me to Quarto and resolving all issues immediately which I have been facing while compiling this document. Thanks to Riddhi for many thoughtful discussion on education practices in general, which gave me a kind of confidence to try something new. Encouragement of the students Ruqaiya, Sanyukta, Urbi, Shweta, Supriya and many others have finally convinced me to put these lectures online . Thanks to all the students who have been patient listeners in my lectures and also giving great suggestions.

Introduction to R Programming

A glimpse to the R Programming

To learn more about software or programming, it is essential to grasp the basic data storage facilities within the environment. R is particularly user-friendly when it comes to handling basic data structures, requiring no prior programming experience. This article introduces vectors, matrices, and lists in R, providing a solid foundation for using R effectively. Understanding these data structures is crucial for performing data analysis and various programming tasks with ease.

Installation

- Go the link: https://posit.co/download/rstudio-desktop/

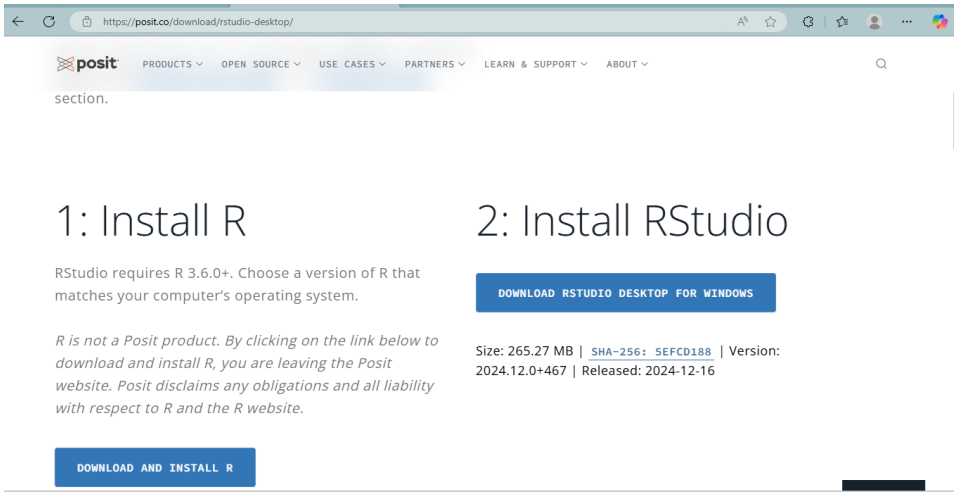


Figure 1: Download R from the webpage and install. Then download RStudio and install in your PC Separate instructions for Windows/Linux/Mac is provided.

- Open RStudio and Create a New R Script

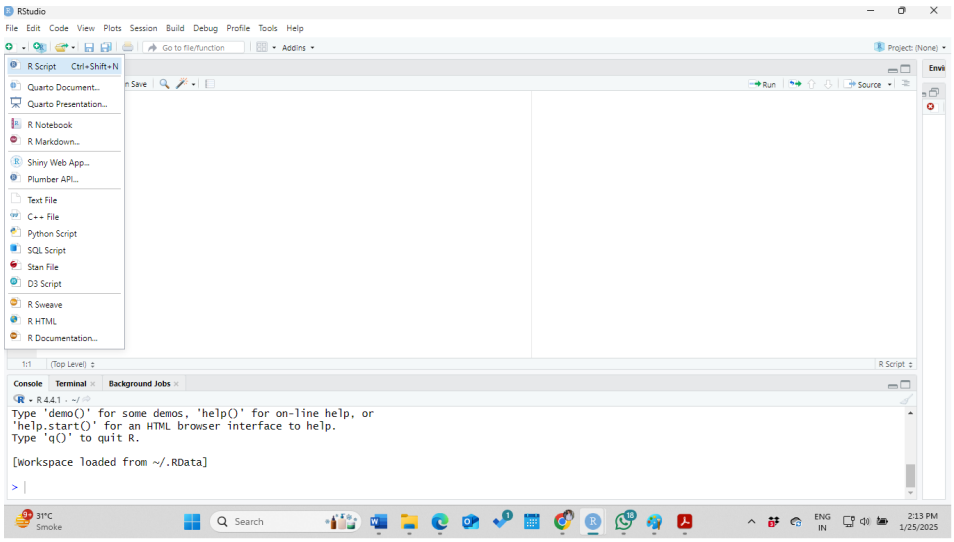


Figure 2: Typical look of the RStudio environment. It may be observed that RStudio provides a whole suits of programming options including dynamic documentations and website design. See the other options in the dropdown menu.

Typically, the look of RStudio contains four windows which are flexible that can be stretched or adjusted as the requirement of the user. However, the options in each pane can be customized by following the sequence: **Tools -> Global Options -> Pane Layout**

Concatenation Operator `c`

Execute the following codes and check the output in the console. The object `x` basically stores the numbers $\{1, 2, 3, 4, 5\}$. There are multiple ways to store these numbers in `x`.

```
x = c(1,2,3,4,5)
print(x)
x = c(1:5)
print(x)
x = 1:5
print(x)
x = 5:1
print(x)
x = c(1:4, 5)
print(x)
x = 11:20
print(x)
```

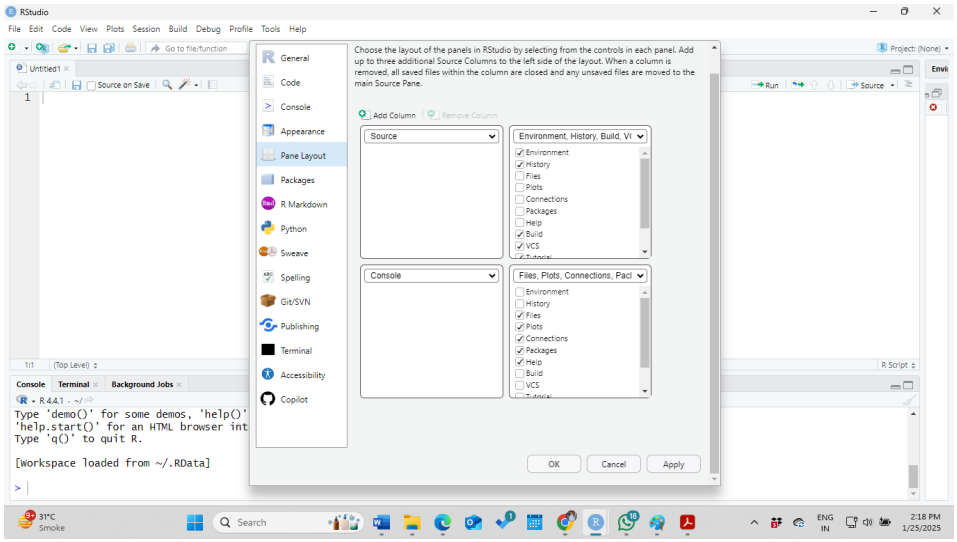


Figure 3: Tools -> Global Options -> Pane Layout. Tick marks the options that you want to view. Customizing the environment is important for individual preferences.

Some basic inbuilt functions in R

The following functions will be helpful in basic operations on vectors or arrays in R. Below the above piece of codes, you can run the following codes as a continuation of the previous steps.

R as a calculator

In the following code snippet, we shall see that R can be used as basic calculators including basic mathematical functions

```
x = 3
print(x)

y = 5
print(y)

z = x + y
print(z)

u = x - y
print(u)
```

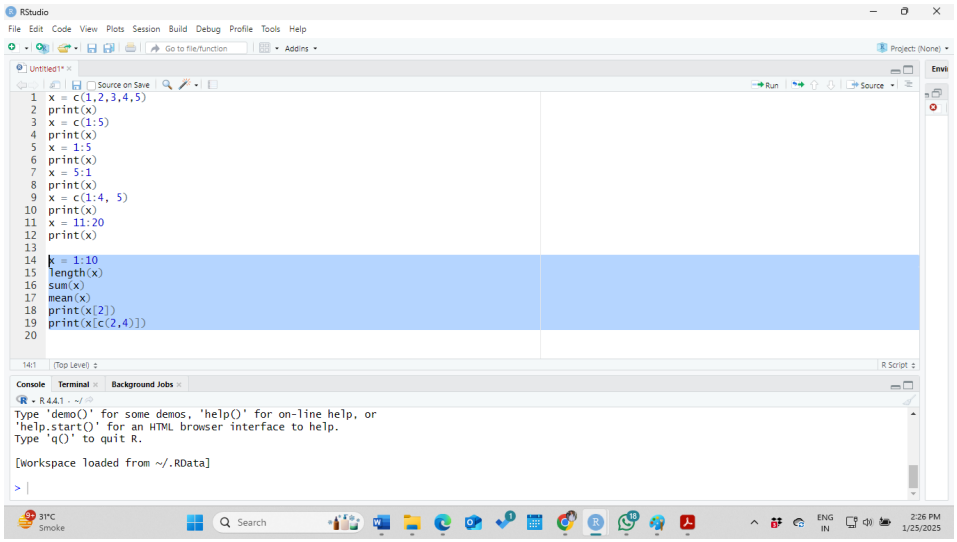


Figure 4: Select the code snippet that you would like to execute, then click Run or press CTRL + Enter. The output of the codes will be printed on the Console

```
v = x/y
print(v)

w = x*y
print(w)
```

In the following some examples are provided involving common functions:

```
x = 2
log(x)
log(x, base = 2)
log10(x)
exp(x)
sin(x)
cos(x)
tan(x)
```

Array of numbers in R

```r
x = 1:10
length(x)
sum(x)
mean(x)
print(x[2])
print(x[c(2,4)])
rev(x)                  # reverse the numbers
cumsum(x)               # cumulative sum
prod(x)                 # product of the numbers
cumprod(x)              # cumulative product of numbers

x = c(1,2,3,4,5)
print(x)
class(x)                # what type it is
length(x)               # length of the array
sum(x)                  # sum of the numbers
x^2                     # square of all numbers
sqrt(x)                 # square root of all the numbers
mean(x)                 # average of these number
cumsum(x)               # cumulative sum
```

In the following, we see some mathematical operations of arrays.

```r
x = 1:5
print(x)
y = 5:1
print(y)
z = x + y               # element wise addition
print(z)
u = x - y               # element wise subtraction
print(u)
v = x*y                 # element wise multiplication
print(v)
w = x/y                 # element wise division
print(w)
```

Although R provides direct addition of two sets of numbers if they are of same length or length of one array is a multiple of another array. We can explicitly write the codes how to perform element wise addition as given below:

```r
x = 1:5                              # the first array
y = 5:1                              # the second array
length(x)
length(y)
length(x) == length(y)               # checking equality of length
z = numeric(length = length(x))      # initialization of array
print(z)
for (i in 1:length(x)) {             # for loop starts here
  z[i] = x[i] + y[i]
}
print(z)
```

There are multiple ways to create arrays in R with specific requirements based on the problem in hand. In the following, we show some examples.

```r
x = 1:100
print(x)
x = seq(1, 100, by = 1)              # understand on your own
print(x)
y = seq(1, 100, by = 3)              # difference is same
print(y)
z = seq(0, 1, by = 0.1)              # creating a mesh for interval
print(z)
length(z)
w = seq(0, 1, length.out= 10)        # understand the difference with the previous
print(w)
length(w)

p = rep(1, 5)
print(p)
q = rep(1:5, each = 5)
print(q)
```

Missing values are very important to deal with in real life applications. In R, the missing values are indicated as NA. The function `is.na()` is used to check whether some entry in the data vector is missing or not. We can see some examples in the following:

```r
x = c(1, 2, 3, NA, 5)
length(x)
is.na(x)
!is.na(x)
```

```r
mean(x)                                  # output NA
median(x)                                  # output NA
sum(x)                                   # output NA
prod(x)                                  # output NA
```

It is important to note that if there is some missing information in the data, basic functions like `mean()`, `sum()` etc will not work. It will also return `NA`. Within the function include the option `na.rm = TRUE`, which indicates that there are missing values in the data. Therefore, the functions will compute the summary ignoring the missing values.

```r
mean(x, na.rm = TRUE)
median(x, na.rm = TRUE)
sum(x, na.rm = TRUE)
prod(x, na.rm = TRUE)
var(x, na.rm = TRUE)              # variance of the data
sd(x, na.rm = TRUE)             # standard deviation of the data
```

Matrices in R

The `matrix()` function is used to create a matrix object with specific number of rows and columns.

```r
m = 4                          # number of rows
n = 3                          # number of columns
A = matrix(data = NA, nrow = m, ncol = n)      # blank matrix
print(A)                                       # Contains NA
```

```
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
[3,]   NA   NA   NA
[4,]   NA   NA   NA
```

```r
A[1,1] = 1                                     # Filling each position
A[1,2] = 2
A[1,3] = 3
A[2,1] = 4
A[2,2] = 5
A[2,3] = 6
```

```r
A[3,1] = 7
A[3,2] = 8
A[3,3] = 9
A[4,1] = 10
A[4,2] = 11
A[4,3] = 12
print(A)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Run the following codes and understand the functioning of each of the functions.

```r
nrow(A)                          # number of rows
ncol(A)                          # number of columns
rowSums(A)                       # sum of each row
colSums(A)                       # sum of each column
rowMeans(A)                      # average of each row
colMeans(A)                      # average of each column
rownames(A) = c("R1", "R2", "R3", "R4")
print(A)
colnames(A) = c("C1", "C2", "C3")
print(A)
```

Suppose that we want to add the row sums and column sums in the same matrix and create a new one.

```r
B = cbind(A, rowSums(A))                    # column bind
print(B)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    6
[2,]    4    5    6   15
[3,]    7    8    9   24
[4,]   10   11   12   33
```

```r
colnames(B)
```

```
NULL
```

```r
colnames(B)[4] = "RowSum"                    # add last column name
colnames(B)                                  # check column names again
```

```
[1] NA       NA       NA       "RowSum"
```

```r
C = rbind(B, colSums(B))                     # adding column sums
print(C)
```

```
     <NA> <NA> <NA> RowSum
[1,]    1    2    3      6
[2,]    4    5    6     15
[3,]    7    8    9     24
[4,]   10   11   12     33
[5,]   22   26   30     78
```

```r
rownames(C)
```

```
NULL
```

```r
rownames(C)[5] = "ColSum"                    # add last row name
print(C)
```

```
       <NA> <NA> <NA> RowSum
<NA>      1    2    3      6
<NA>      4    5    6     15
<NA>      7    8    9     24
<NA>     10   11   12     33
ColSum   22   26   30     78
```

Addition of two matrices

```r
P = matrix(data = 1:9, nrow = 3, ncol = 3)
print(P)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```r
dim(P)                                          # dimension of a matrix
```

```
[1] 3 3
```

```r
Q = matrix(data = 11:19, nrow = 3, ncol = 3)
print(Q)
```

```
     [,1] [,2] [,3]
[1,]   11   14   17
[2,]   12   15   18
[3,]   13   16   19
```

```r
dim(Q)
```

```
[1] 3 3
```

```r
dim(P) == dim(Q)                                # checking dimension
```

```
[1] TRUE TRUE
```

```r
R = matrix(data = NA, nrow = nrow(P),
           ncol = ncol(P))
print(R)                                        # blank matrix
```

```
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
[3,]   NA   NA   NA
```

```r
for (i in 1:nrow(P)) {
  for (j in 1:ncol(P)) {
    R[i,j] = P[i,j] + Q[i,j]                    # element wise addition
  }
}
cat("The addition of the given matrices P and Q are\n", R)
```

```
The addition of the given matrices P and Q are
 12 14 16 18 20 22 24 26 28
```

! Classroom Assignment: Matrix multiplication

- Suppose that A is $m \times n$ matrix and B is an $n \times k$ matrices of real numbers. Write a programme to compute the product of these two matrices AB (matrix multiplication).
- Using R write down the following to matrices

$$E = \begin{bmatrix} 5.1 & 3.2 & 6.0 & 7.1 \\ 4.8 & 2.9 & 5.5 & 6.8 \\ 5.4 & 3.5 & 6.2 & 7.3 \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \\ 0.1 \end{bmatrix}$$

- Multiply these two matrix EW and report its dimension.
- From each column of E subtract the column means and call it $center_E$.
- Consider the following piece of codes:

```r
z = sample(1:1000, replace = TRUE)
print(z)
```

- Check the `help(sample)` to gain knowledge about the `sample` function.
- Pick out the values in `z` which are greater than 600.
- What are the index positions in `z` of the values which are greater than 600?
- Create a vector

$$\left(|z_1 - \bar{z}|^{1/2}, |z_2 - \bar{z}|^{1/2}, ..., |z_n - \bar{z}|^{1/2}\right),$$

, where $\bar{z}$ denotes the mean of the vector $z = (z_1, z_2, ..., z_n)$

- How many numbers in `z` are divisible by 2? (Note that the modulo operator is denoted %%.)

- Sort the numbers in the vector `z` in the order of increasing values.
- Compute the following series
  - $\sum_{i=1}^{5}\sum_{j=1}^{1} 5\frac{i^3}{5+j}$
  - $\sum_{i=1}^{5}\sum_{j=1}^{1} 5\frac{i^3}{5+ij}$
  - $\sum_{i=1}^{5}\sum_{j=1}^{i} \frac{i^3}{5+ij}$

In ecological problems, we do always deal with numeric values, there are observations which are stored in strings as well. For example, we often collect data from different locations, and these identified by letters A, B, C, D. From these locations, the numbers of observations are 32, 43, 20 and 12, respectively. Therefore, to store this data, we need to have two vectors, one for locations and another for the counts. In addition, the location A and C are categorized as Urban and location B and D are categorized as Others. Therefore, we may require logical vector as well to store this information.

```
loc = c("A", "B", "C", "D")      # note inverted comma for characters
count = c(32, 43, 20, 12)          # numeric vector
urban = c(TRUE, FALSE, TRUE, FALSE)
print(loc)
print(count)
class(loc)                              # character vector
class(count)                            # numeric vector
class(urban)                            # logical vector
```

The function `class()` is a useful function to understand the type of the data and large scale data analysis problems often compatibility of different data types need to be checked. These three vectors are of same length equal to 5. However, they are now different objects, and we usually want to have an Excel view of these objects in a single dataset with multiple columns. In R, `data.frame()` function is used to create datasets.

```
data = data.frame(loc, count, urban)        # create dataset
print(data)
dim(data)                                           # dimension of data
nrow(data)                                        # number of rows
names(data)                                     # column names
ncol(data)                                        # number of columns
```

Suppose that we have information from another two locations E and F. E and F are categorized as urban and non-urban areas, respectively. From F the count is 29, however, from E, the count is not available. The new information can be added to the existing data in the following ways:

```r
loc = c("E", "F")                              # new location
count = c(NA, 29)                              # counts
urban = c(TRUE, FALSE)                       # whether urban
new = data.frame(loc, count, urban)      # new data information
data = rbind(data, new)                        # adding to existing data
print(data)                                    # print in the console
```

You may would like to have an Excel kind of view.

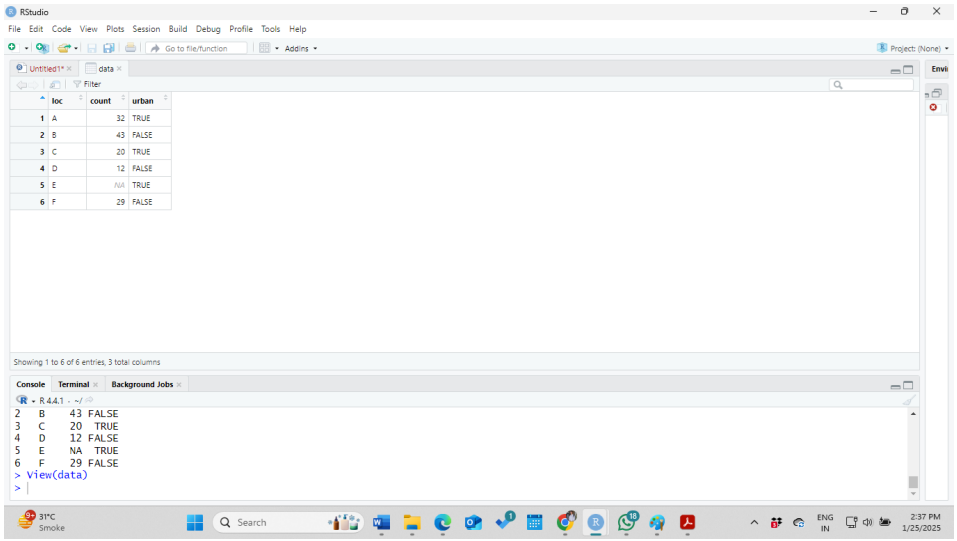


Figure 5: The View(data) command will generate an Excel kind of view in the R environment. Not the presence of missing value.

There are important functions which directly helps to understand the structure of the data.

```r
summary(data)
complete.cases(data)          # rows without missing information
!complete.cases(data)         # rows with missing information
data[,3]                      # third column of the data
data[1,2]
data[3,]                      # third row of the data
```

If you wish, you can also change the names of the rows and columns. The functions `rownames()` and `colnames()` are used to execute this task. Execute the following pieces of codes to understand their functionalities.

```r
colnames(data) = c("Location", "Count", "Urban")
print(data)
rownames(data) = c("R1", "R2", "R3", "R4", "R5", "R6")
print(data)
```

A little understanding of the matrices is helpful in statistical modelling of the real data sets irrespective of the academic background. In the following codes, we see some operations on matrices and also learn how to create a matrix in R. Matrices are two dimensional arrangements of numbers and the dimension of a matrix A is 2×3, which means that the number of rows is 2 and the number of columns is 3. There is a total of six number are stored in the matrix. Let us create this matrix using the following codes:

```r
A = matrix(data = 1:6, nrow = 2, ncol = 3)
print(A)
A[1,]                                    # First row of A
A[,3]                                    # Third column of A
A[1,3]                                 # number in the (1,3) position
dim(A)                               # dimension of A
nrow(A)                                  # number of rows
ncol(A)                                  # number of columns
rowSums(A)                               # sum of numbers in rows
colSums(A)                               # sum of numbers in columns
rowMeans(A)                            # average of numbers in rows
colMeans(A)                            # average of numbers in columns
sum(A)                                   # addition of all numbers in A
```

When the rows and columns are exchanged, we say the transpose operation on the matrix. Therefore, transpose of the matrix A, will contain the same numbers, but number of rows will be 3 and the number of columns will be 2.

```r
t(A)
dim(t(A))
```

If two matrices A and B are of the same dimension, then they can be added or subtracted from each other. This operation is elementwise which means the (1,2) position number in A will be added (subtracted) with the (1,2) position number in B. If C = A+B, then C[1,2] = A[1,2]+B[1,2]. Similar will be followed for all other entries. The following example will demonstrate this fact:

```r
A = matrix(data = 1:6, nrow = 2, ncol = 3)
B = matrix(data = 6:1, nrow = 2, ncol = 3)
C = A + B                                          # matrix addition
print(C)
D = A - B                                          # matrix subtraction
print(D)
E = A*B                                          # element wise multiplication
print(E)
F = A/B
print(F)                                 # element wise division
```

If we have two matrices A and B as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \text{ and } B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

Then,

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}.$$

$$A \times B = \begin{bmatrix} a_{11} \times b_{11} & a_{12} \times b_{12} \\ a_{21} \times b_{21} & a_{22} \times b_{22} \end{bmatrix}.$$

Matrix multiplication, however, is different. Two matrices A and B of dimensions $m \times n$ and $p \times q$ can be multiplied if the number of columns in A is equal to the number of rows in B, that is, $n = p$. The resulting matrix will be of order $m \times q$. In R, the symbol **%*%** is used to perform matrix multiplication.

For example:

$$A\%*\%B = \begin{bmatrix} a_{11} \times b_{11} + a_{12} \times b_{21} & a_{11} \times b_{12} + a_{12} \times b_{22} \\ a_{21} \times b_{11} + a_{22} \times b_{21} & a_{21} \times b_{12} + a_{22} \times b_{22} \end{bmatrix}.$$

```r
A = matrix(data = 1:6, nrow = 3, ncol = 2)
print(A)
B = matrix(data = 1:6, nrow = 2, ncol = 3)
print(B)
ncol(A) == nrow(B)                       # condition of multiplication
C = A%*%B                                      # matrix multiplication
print(C)
dim(C)                                     # 3 by 3 matrix
```

A matrix is called a square matrix if the number of rows is equal to the number of columns of the matrix. The entries in the positions (1,1),(2,2), etc. are called diagonal entries. A matrix will diagonal entries 1 and off-diagonal entries as zero is called the identity matrix. In the following, A is a square matrix of order 3. Using the `diag()` function, we can obtain the diagonal entries.

```
A = matrix(data = 1:9, nrow = 3, ncol= 3)
diag(A)                                  # diagonal elements
```

If A and B are two square matrices of order n, we say that B is the inverse of A (denoted by A^{-1}) if $A\% * \%B = I = B\% * \%A$. The inverse matrix B can be obtained by the `solve()` function in R.

```
A = matrix(data = c(1,3,1,2,1,2,4,3,1), nrow = 3, ncol = 3)
print(A)
solve(A)                                   # inverse of A
eigen(A)                                   # eigenvalues of A
solve(A)%*%A                             # identity matrix
det(A)                                     # determinant of A
rowMeans(A)                              # average of each row
colMeans(A)                              # average of each column
colSums(A)                               # sum of each column
rowSums(A)                                 # sum of each row
```

In above, we have seen data.frame and matrix objects in R. We observed that the matrix can not hold data of two different types. In another words, if we have two columns, one represents the abundance of some species (numeric) and the other represents the location of the site (in character). We cannot store this two information in a single matrix. Even if we store them, both the columns will be converted to character types. A small demonstration is shown below for understanding. The variable Density is a numeric variable, however, when it is stored in a matrix with another column Sites, the Density column is converted to character. In the `print(M)`, observe that all the numbers are within inverted comma.

```
Density = c(100, 138, 80, 20, 41)      # numeric type
Sites = c("A", "B", "C", "D", "E")  # character type
M = cbind(Density, Sites)                  # Column binding
class(M)                                        # Matrix
class(M[,"Sites"])                         # character type
class(M[,"Density"])                 # character type
class(Density)                               # numeric type
print(M)                                        # all within inverted comma
```

Apart from the data.frame and matrix, the knowledge of the list data structure also comes very handy in data analysis applications of R programming. You can think of list as a sequence of big containers. In each container, you can store different types of objects. For example, if you have a list of length 3, then in the first container, you can keep a matrix and in the second container, you can keep a data.frame and in the third container, you can keep a vector also.

```
List = list(length = 3)              # Empty list of length 3
D = data.frame(A = rep(c(1,2), 4), B = rnorm(n=8))
M = matrix(data = rnorm(n = 9), nrow = 3, ncol = 3)
x = 1:10
List[[1]] = D                          # First container
List[[2]] = M                          # Second container
List[[3]] = x                          # Third container
print(List)                           # print the list
```

Writing functions

It is important to have a basic idea of writing functions in R. Functions are often helpful to automate some data analysis process and help in reducing time for doing repetitive tasks. For example, I am interested in computing the average of the first and the last number of a set of values. The following code will do this task.

```
x = 1:10                                 # data values
(x[1] + x[10])/2                        # average of the first and last
```

Suppose the above process we want to do it for 5 sets of values and each of different lengths. Every time, we need to write this. However, if we write a function, the process can be done in much nicer way.

```
a = rep(1, 5)                            # repeat 1 five times
b = c(5:10)                             # numbers 5 to 10
c = rev(10:1)                            # 1,2,3,…,10
d = 1:20                                 # 1,2,3,…,20
f = function(x){                         # function in R
  (x[1] + x[length(x)])/2
}
f(a)                                        # Check the return value
f(b)
f(c)
f(d)
```

We can also do customization of the function. Suppose instead of numeric value someone enters a character in the entry, then the summation will not be computed and result in error. Inside the function body, we can add more comments.

```
e = c("A", "B", "C")
f(e)                                          # error and read the error
```

You can modify the function as follows. As you can see that you can add multiple conditions and modify the function as per your requirements.

```
f = function(x){                              # modified function
  if(!is.numeric(x) == TRUE)
    return("Can not compute: the input is not numeric.")
  else
    return((x[1]+x[length(x)])/2)
}
f(e)                                          # customized output
```

You can also write down mathematical functions. The common algebraic and trigonometric functions are available in R. Some common functions are `sin()`, `cos()`, `log()`, etc. In the following, we show an example. The curve() function is very important to understand when dealing with mathematical functions.

```
par(mfrow = c(1,2))
f = function(x){
  x^2*sin(10*x)                       # body of the function
}
curve(f(x), -1,1, col = "red", lwd = 3, cex.lab = 1.3,
      main = expression(x^3*sin(10*x)))

g = function(x){
  x^3
}
h = function(x){
  sin(10*x)
}
curve(g(x), col = "red", lwd = 3, -1, 1, cex.lab = 1.3,
      ylim = c(-1.2, 1.6), lty = 2)
curve(h(x), add = TRUE, col = "blue", lwd = 3, lty = 3)
legend("topleft", legend = c(expression(x^3), expression(sin(10*x))), bty = "n",
       lwd = c(3,3), col = c("red", "blue"), lty = c(2,3),
       cex = 0.8)
```

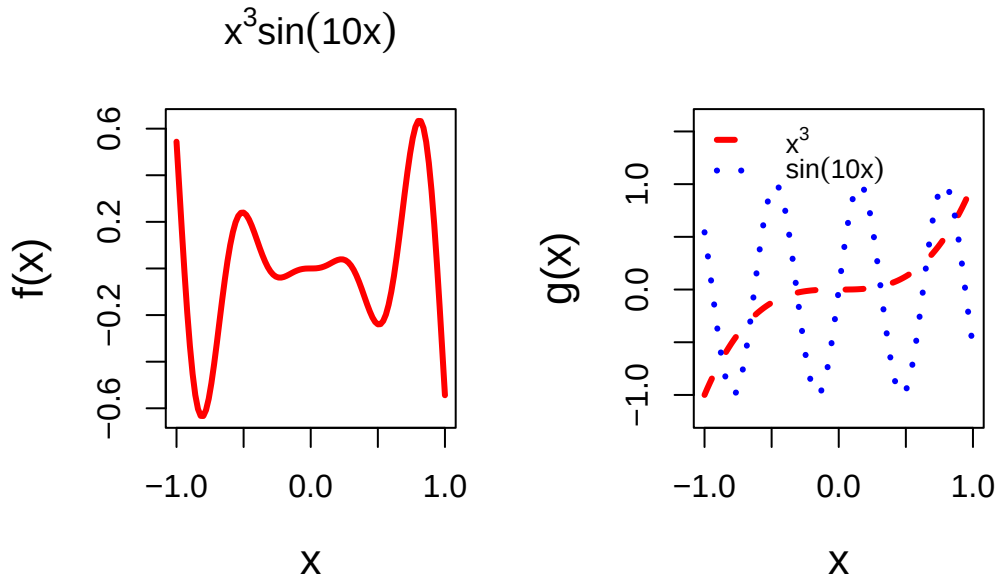


Figure 6: The curve function can be efficiently used to plot mathematical functions in an effective way. You can also perform mathematical annotations on the plot using the expression() function.

The curve function contains several important arguments, `main`, `cex.main`, `add`, `lty`, `lwd`, `col`, etc. You are encouraged to play with these options and see the changes in the plots. This hands-on experience will increase the comfort level in handling R programming tasks. You can add legends as well using the `legend()` function. The option `bty = "n"` specifies that we do not want to have a bounding box about the legend. The `expression()` function is useful for mathematical annotations in the plot window.

Some important functions in R

In the following, we list some common R functions which operates on the list of values in R. We give a small example, however, the same can be applied to large data sets as well.

```
x = c(rep(1,3), rev(1:5))
print(x)
which.max(x)
which.min(x)
which(x == 1)
which(x != 1)
sum(x == 1)
x[x == 1]
```

```r
x[x!=1]
x[x > 1]
```

Run the above codes and interpret them on your own.

Symbolic computation

R also provides some options for performing symbolic computation. The operator `D` is used for this computation. In the following, we perform the symbolic derivative of the function $\psi(x) = \frac{2x}{3+x}$ to compute $\psi'(x)$ using R. Another example is also given with $\psi(x) = \frac{2x^2}{1+x^2}$. We need to store the function within the `expresion` command.

```r
par(mfrow = c(1,2))
expr = expression(2*x/(3+x))
cat("The derivative of the function is given by \n")
```

```
The derivative of the function is given by
```

```r
D(expr, 'x')
```

```
2/(3 + x) - 2 * x/(3 + x)^2
```

```r
diff_psi = function(x){
  (2/(3 + x) - 2 * x/(3 + x)^2)*(x>0)
}

expr = expression(  ((2*x^2)/(1+x^2)))
cat("The derivative of the function is given by \n" )
```

```
The derivative of the function is given by
```

```r
D(expr, 'x')
```

```
2 * (2 * x)/(1 + x^2) - (2 * x^2) * (2 * x)/(1 + x^2)^2
```

```r
diff_phi = function(x){
  (2 * (2 * x)/(1 + x^2) - (2 * x^2) * (2 * x)/(1 + x^2)^2)*(x>0)
}
curve(diff_psi(x), col = "red", lwd = 3,
      0.001, 10,
      ylab = expression(nabla(psi(x))))
curve(diff_phi(x), col = "blue", lwd = 3,
      0.001, 10, ylab = expression(nabla(phi(x))))
```

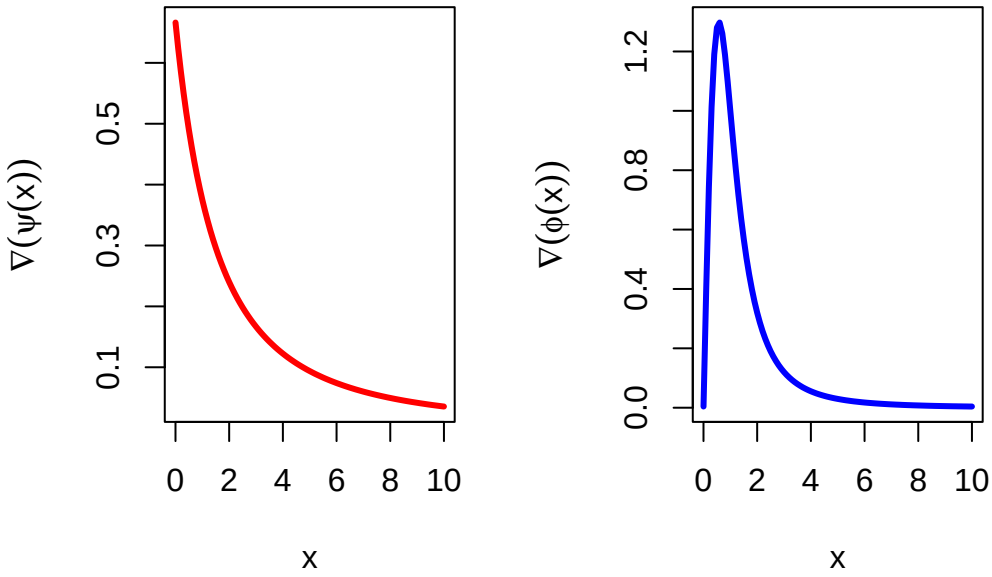


Figure 7: The derivative of the functions $\psi(x)$ and $\phi(x)$ for $x \in (0, \infty)$.

Introduction to Probability Distributions

Probability mass function

Suppose that we have a fair coin and we toss it two times, then the sample space is given by

$$\mathcal{S} = \{HH, TH, HT, TT\}$$

Suppose that $X : \mathcal{S} \to \mathbb{R}$, defined as

$$X(s) = \text{number of H in } s,$$

therefore $X \in \{0, 1, 2\}$. Since it is a fair coin, the following probabilities can be easily computed.

$$\mathrm{P}(X = 0) = \frac{1}{4}, \quad \mathrm{P}(X = 1) = \frac{1}{2}, \quad \mathrm{P}(X = 2) = \frac{1}{4}.$$

We can visualize these values as a mass on the real line having masses at the values $\{0, 1, 2\}$. Let us visualize it using R.

In the following, we first visualize the probability mass function for $n = 2$.

```
n = 2                                # number of throws
x = 0:n                              # possible values of the random variable X
print(x)
```

```
[1] 0 1 2
```

```
p = 0.5                              # probability of head
p_x = choose(n, x)*p^x*(1-p)^(n-x)
print(p_x)
```

```
[1] 0.25 0.50 0.25
```

```
plot(x, p_x, pch = 19, ylab ="P(X=x)", type="h", lwd = 2,
     col = "red", lty =2, main = "Binomial (n,p)")
points(x, p_x, pch = 19, col = "blue", cex = 1.5)
```

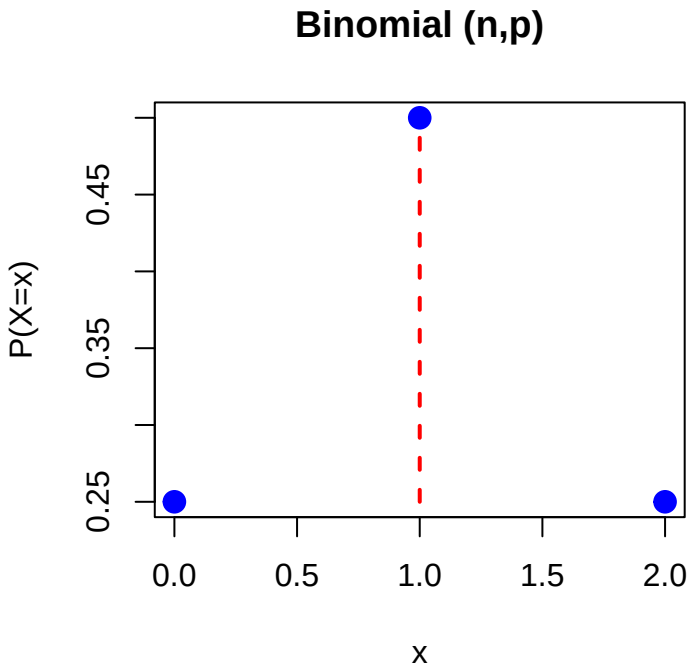


Figure 1: The probability mass function of the binomial probability distribution when the coin is fair. The random variable X represents the number of heads when the coin is tossed twice.

For $n = 10$, we now visualize various shapes of PMF of X for different choice of $p \in (0, 1)$. We can now generalize this situation to consider a coin with probability of head as $p \in (0, 1)$. Therefore, we are considering tossing with a biased coin as well. It is easy to compute that (based on the classroom exercise) is that

$$\mathrm{P}(X = x) = \binom{2}{x} p^x (1-p)^{2-x}, x \in \{0, 1, 2\},$$

and zero otherwise. In the following, we check various shapes of the distribution of the mass on the real line.

Listing 0.1 R Code: The reader is encouraged to see how the loop is written for different choices of p.

```
p_vals = c(0.05, seq(0.1, 0.9, by = 0.1), 0.95)     # different values of prob of head
print(p_vals)
n = 10
par(mfrow = c(2,3))
for (p in p_vals) {
  x = 0:n
  p_x = choose(n, x)*p^x*(1-p)^(n-x)
  print(p_x)
  plot(x, p_x, pch = 19, ylab ="P(X=x)", type="h", lwd = 2,
       col = "red", lty =2, main = paste("p = ", p))
  points(x, p_x, pch = 19, col = "blue", cex = 1.5)

}
```

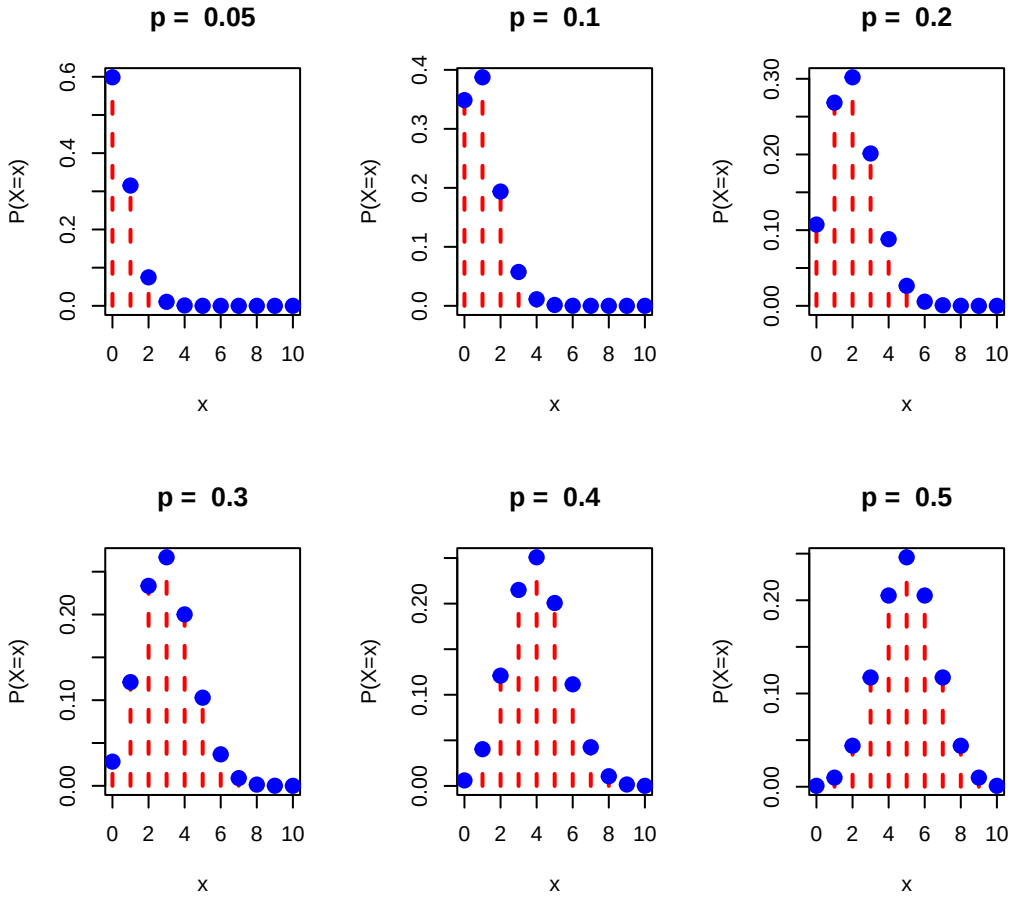


Figure 2: The shapes of the binomial PMF for different choice of the success probability p. As p is close to 1, more number of heads is expected to occur.

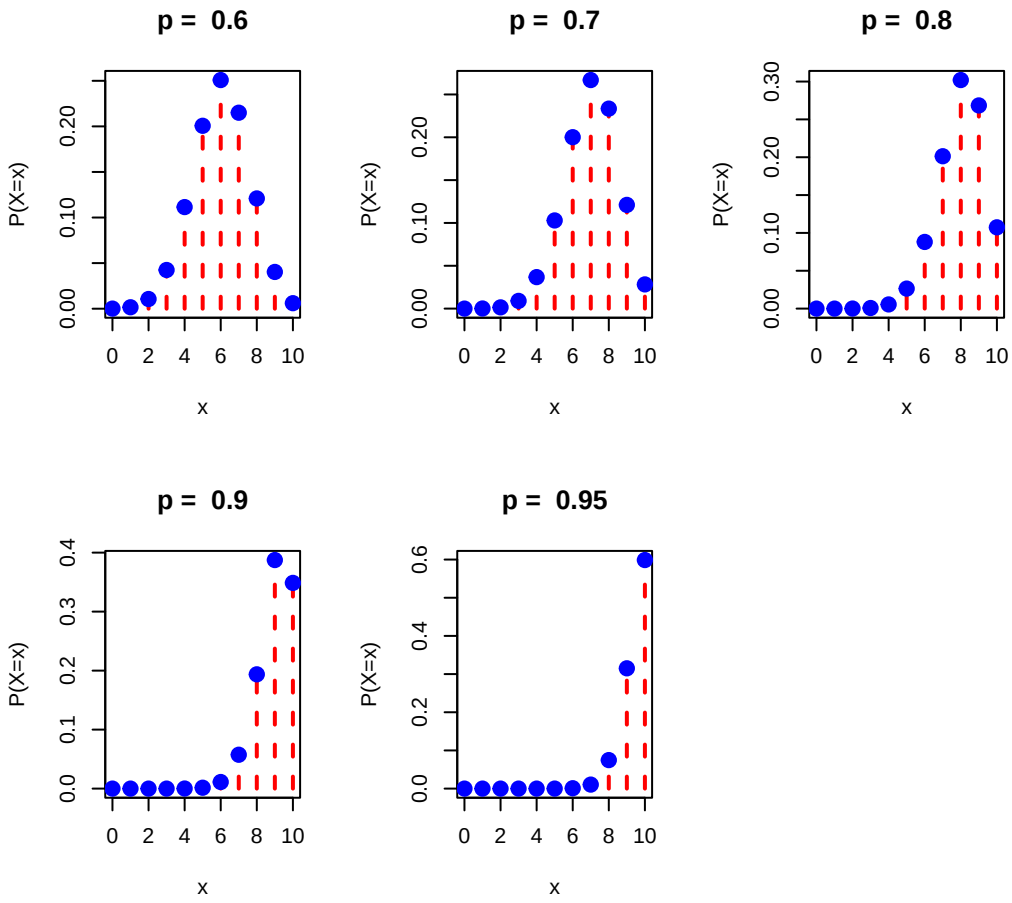


Figure 3: The shapes of the binomial PMF for different choice of the success probability p. As p is close to 1, more number of heads is expected to occur.

Listing 0.2 R Code: The reader is encouraged to see how the loop is written for different choices of n.

```
par(mfrow = c(2,3))
p = 0.7
n_vals = c(2,4,6,8,10, 12, 15,20, 25, 50, 75, 100)
print(n_vals)
for (n in n_vals) {
  x = 0:n
  p_x = choose(n, x)*p^x*(1-p)^(n-x)
  print(p_x)
  plot(x, p_x, pch = 19, ylab ="P(X=x)", type="h", lwd = 2,
       col = "red", lty =2, main = paste("n = ", n))
  points(x, p_x, pch = 19, col = "blue", cex = 1.5)
}
```

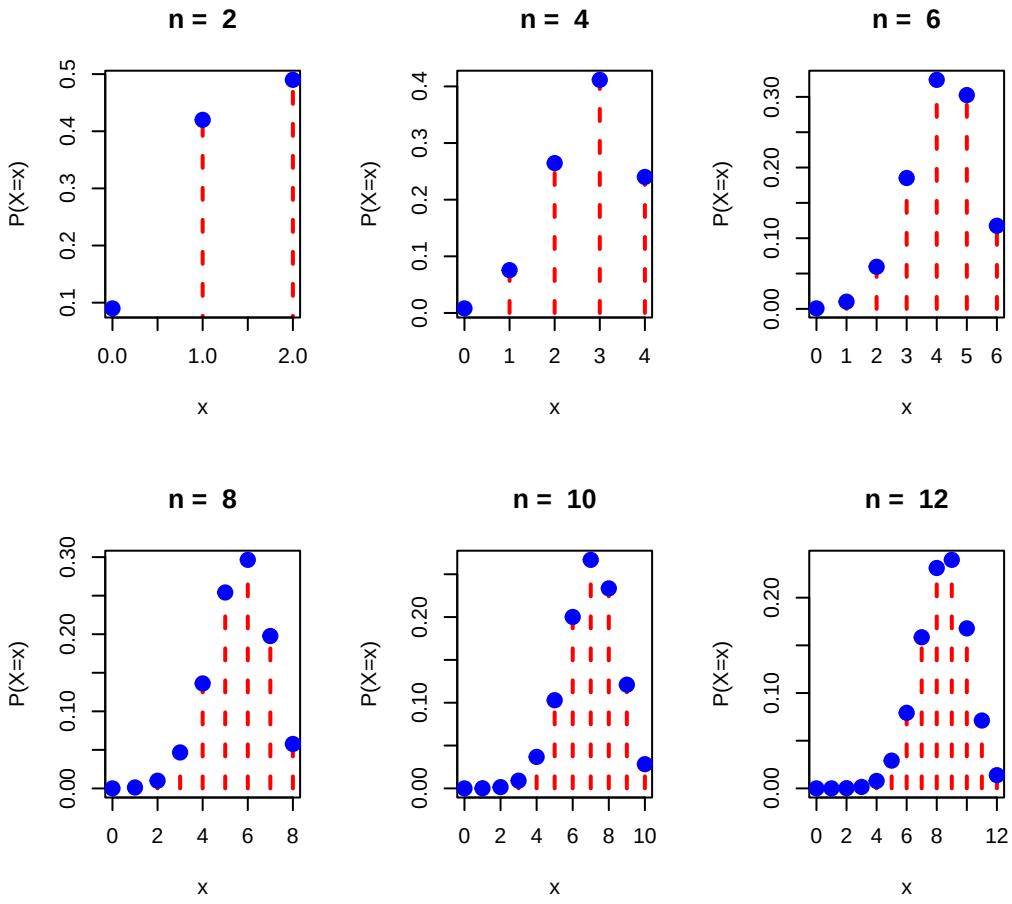


Figure 4: The shapes of the binomial PMF for different choice of n and the success probability p is fixed.

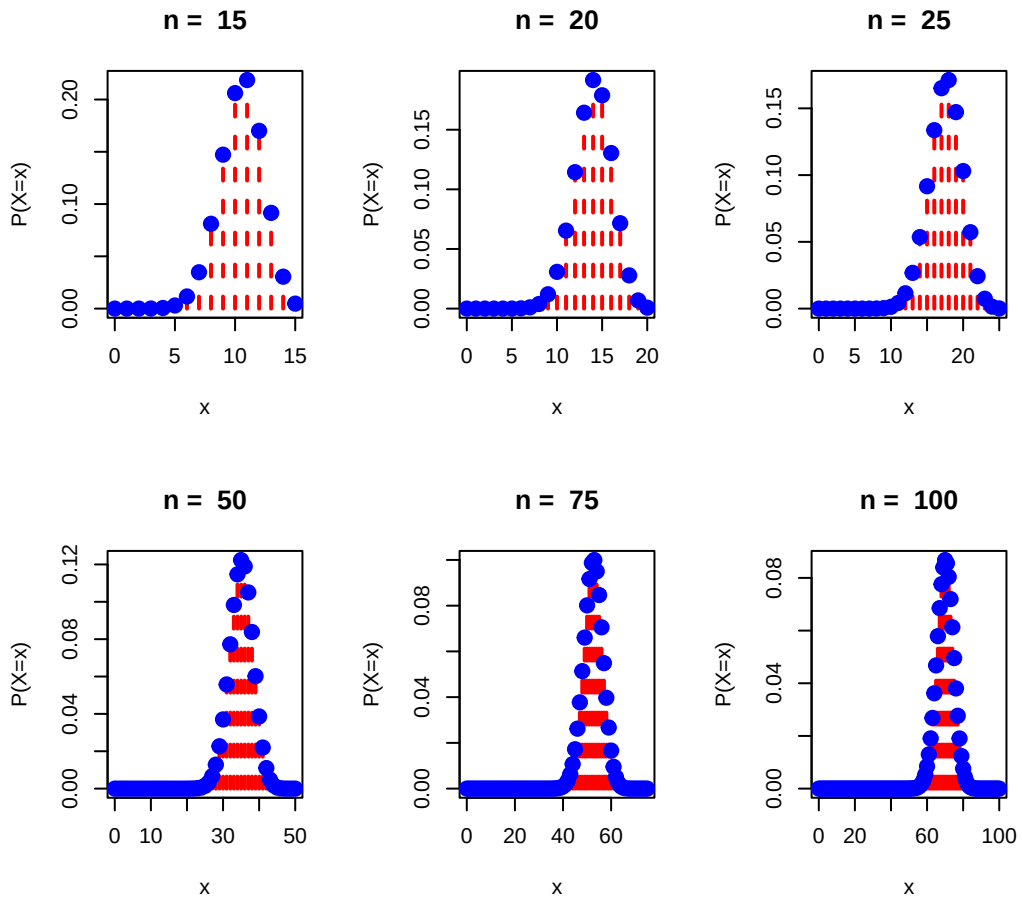


Figure 5: The shapes of the binomial PMF for different choice of n and the success probability p is fixed.

We can compute the center of mass of these masses using the formula $\sum m_i x_i / \sum m_i$. You can check that the center of mass is given by $2p$. It can be easily verified that if the coin is tossed thrice, then the center of mass is given by $3p$. In the plot, we have added the centre of mass using a vertical magenta colored dotted line.

Listing 0.3 R Code: The students are encouraged to realize that how simple plots can be more effective for a better understanding of the concepts. We usually compute the expectation in a mathematical way and hardly realize its use. However, the idea of center of mass or the average value give a better understanding of the concepts.

```
n = 2                                                  # number of throws
p_vals = c(0.05, seq(0.1, 0.9, by = 0.1), 0.09)
par(mfrow = c(2,3))
x = 0:n  # values taken by X
for(p in p_vals){
  p_x = choose(n, x)*p^x*(1-p)^(n-x)                   # probability values
  plot(x, p_x, pch = 19, type = "h", ylab = "P(X=x)",
       cex = 1.3, col = "darkgrey", ylim = c(0,1),
       lwd = 3, main = paste("p = ", p), lty = 2)
  points(x, p_x, pch = 19, col = "red",
         cex = 1.5)
  CM = n*p                                             # expected value
  abline(v = CM, col = "magenta",
         lwd= 2, lty = 2)
  abline(h = 0, col = "darkgrey", lwd = 2)
}
```

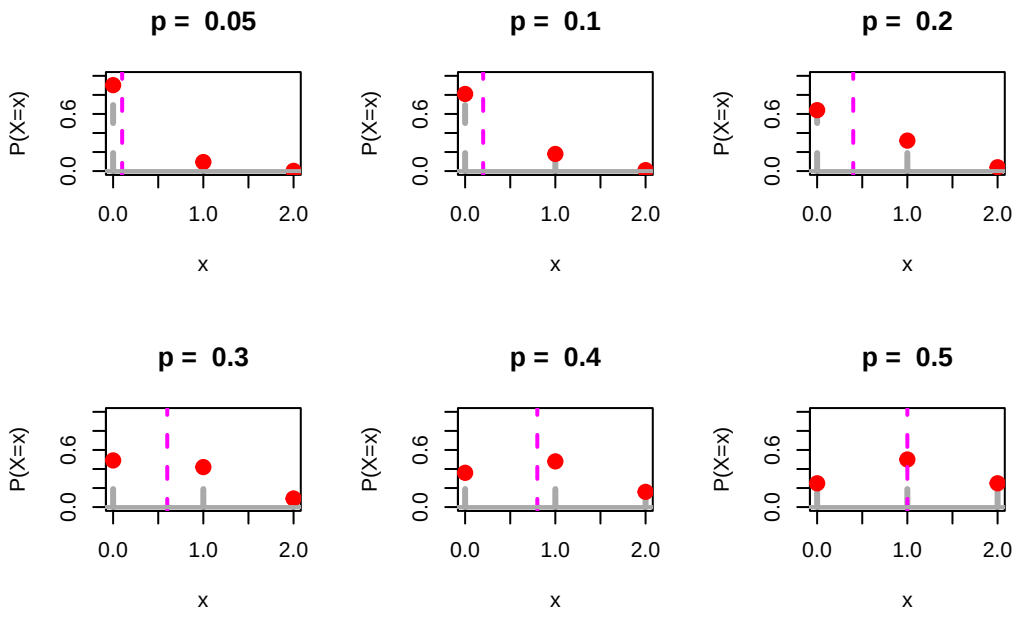


Figure 6: It should be noted that as the probability of head changes, the center of mass (the expected value) also changes. For example, as p increases, the mass shifts towards the right. When $p = \frac{1}{2}$, the center of mass is exactly at the point 1. The vertical dotted magenta color line indicates the center of mass. For students, it may be considered that the point through which the eartch is attracting the mass.

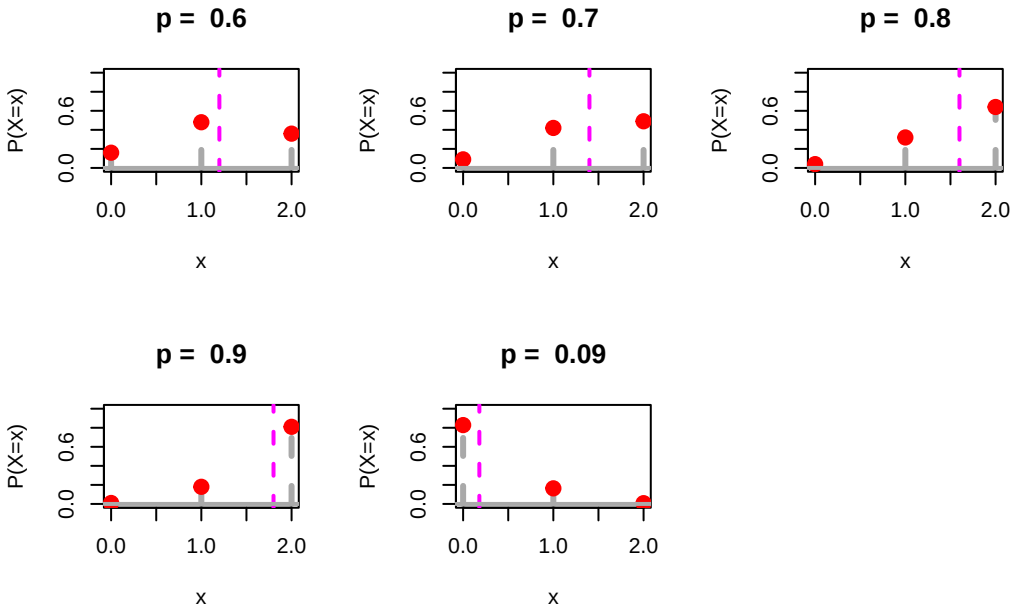


Figure 7: It should be noted that as the probability of head changes, the center of mass (the expected value) also changes. For example, as p increases, the mass shifts towards the right. When $p = \frac{1}{2}$, the center of mass is exactly at the point 1. The vertical dotted magenta color line indicates the center of mass. For students, it may be considered that the point through which the eartch is attracting the mass.

! Classroom exercises

- Find the value of a so that the following function is a PMF.

$$g(x) = \begin{cases} a\frac{\lambda^x}{x!}, & x \in \{0, 1, 2, \dots, \} \\ 0, & \text{otherwise} \end{cases}.$$

  Plot the PMF for $\lambda \in \{1, 2, 3, 4, 5, 6\}$. Use the option `par(mfrow = c(2,3))`

- Find the value of a so that the following function is a PMF:

$$g(x) = \begin{cases} \frac{a}{x^2}, & x \in \{0, 1, 2, \dots, 100\} \\ 0, & \text{otherwise} \end{cases}.$$

- Find the value of a so that the following function is a PMF:

$$g(x) = \begin{cases} a(1-p)^{x-1}, & x \in \{1, 2, \dots\} \\ 0, & \text{otherwise} \end{cases}.$$

  Plot the PMF using R for different choices of p.

- Find the value of a so that the following function is a PMF:

$$g(x) = \begin{cases} \frac{a}{n+1}, & x \in \{0, 1, 2, \dots, n\}, \text{ where } n \in \mathbb{N} \\ 0, & \text{otherwise} \end{cases}.$$

Probability density function

In the above discussion, we have considered random variable that takes only a finite or countable values on the real line. We start our discussion with a family of functions $f_m(x)$, $m \in \{1, 2, 3, ...\}$, which is defined as

$$f_m(x) = \begin{cases} mx^{m-1}, & 0 < x < 1, \\ 0, & \text{Otherwise.} \end{cases} \tag{0.1}$$

In the following, we first draw a shape of these functions using the following code

Listing 0.4 R Code: The students are encouraged to see how the support of the probability density function is given in the body of the function using the star symbol

```
f_m = function(x){
  m*x^(m-1)*(x>0)*(x<1)
}

par(mfrow = c(2,3))
m_vals = 1:6                                    # different values of m
for(m in m_vals){
  curve(f_m(x), col = "red", lwd = 2, main = paste("m = ", m),
        xlim = c(-0.2, 1.1), ylim =c (0,6),
        ylab = expression(f[m](x)))
  points(m/(m+1), 0, pch = 19, col = "blue", cex = 1.3)
}
```

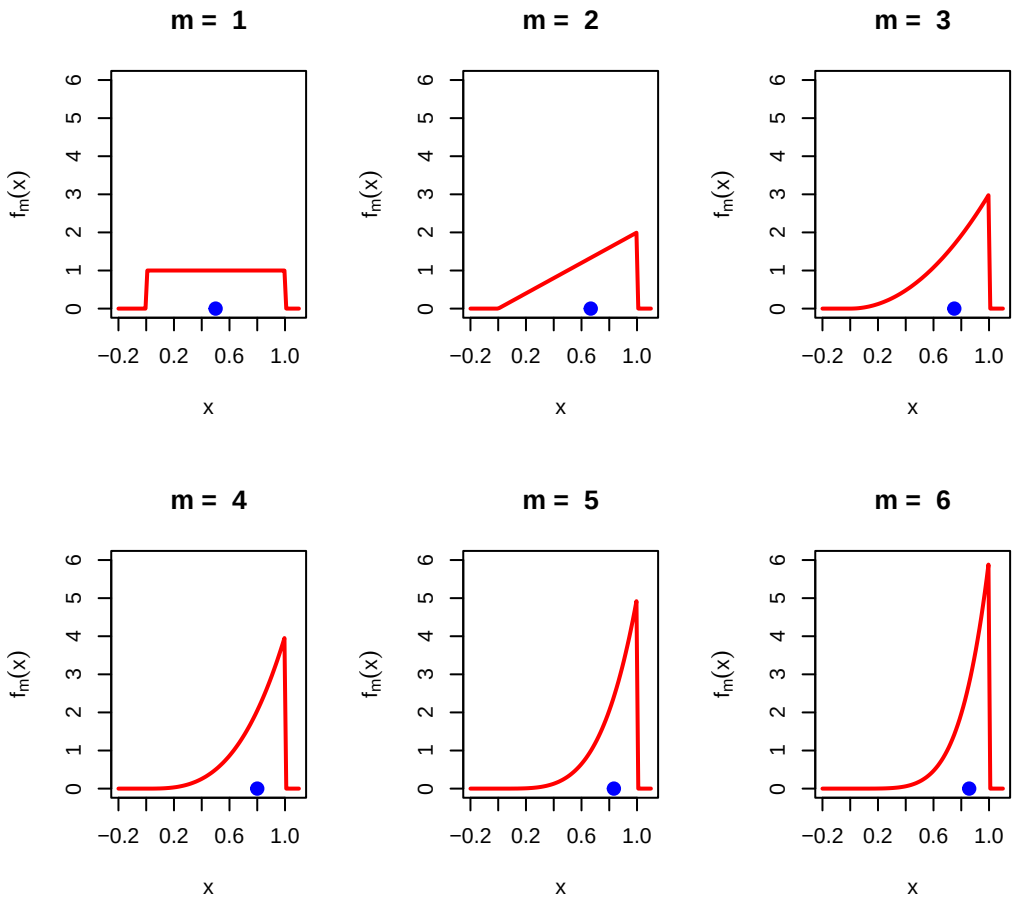


Figure 8: The shape of the function $f_m(x)$ for different choices of m.

Probability density function

A function $f(x)$ is called a probability density function (PDF) if

- $f(x) \geq 0$ for all $x \in (-\infty, \infty)$.
- $\int_{-\infty}^{\infty} f(x)dx = 1$.

Using the `integrate()` function, we can check whether the functions $f_m(x), m \in \{1, 2, 3, \dots, \}$ are PDF or not. Before doing it by using R, we can easily check that by using the direct

integration as follows:

$$\begin{aligned}\int_{-\infty}^{\infty} f_m(x)dx &= \int_{-\infty}^{0} f_m(x)dx + \int_{0}^{1} f_m(x)dx + \int_{1}^{\infty} f_m(x)dx \\ &= \int_{-\infty}^{0} 0 \cdot dx + \int_{0}^{1} mx^{m-1}dx + \int_{1}^{\infty} 0\cdot \\ &= m\left[\frac{x^m}{m}\right]_{x=0}^{x=1} = 1.\end{aligned}$$

Using R, we can check whether the integral is 1.

```
m = 3                                  # change the value of m
integrate(f_m, lower = 0, upper = 1)
```

```
1 with absolute error < 1.1e-14
```

Some more examples

Suppose that we consider the set of functions $g_m(x)$ for $m \in \{1, 2, \dots, \}$

$$g_m(x) = \begin{cases} x^{m-1}e^{-x}, & 0 < x < \infty \\ 0, & \text{otherwise.} \end{cases}$$

In the classroom, many students may not be knowing about the `gamma()` function. In such a case, the instructor is encouraged to start with the computation of integration by hand

$$\int_0^{\infty} g_m(x)dx, \quad \text{for} \quad m \in \{1, 2, 3\}$$

and come up with the generalization

$$\int_0^{\infty} g_m(x)dx = (m-1)\int_0^{\infty} g_{m-1}(x)dx = \cdots = (m-1)!$$

Let us now plot these functions

Listing 0.5 R Code: It is always a better option to plot the functions whichever is discussed during the lecture. This gives a better understanding for the students and the visual displays play a longstanding impact on the students. The students are encouraged to integrate these functions and see whether their theoretical computation matches with the output of the `integrate()` function.

```
g_m = function(x){
  x^(m-1)*exp(-x)*(x>0)
}
par(mfrow = c(2,3))
m_vals = 1:12
for(m in m_vals){
  curve(g_m(x), col = "red", lwd = 2, main = paste("m = ", m),
        xlim = c(-0.2, 15), ylab = expression(g[m](x)))
}
```

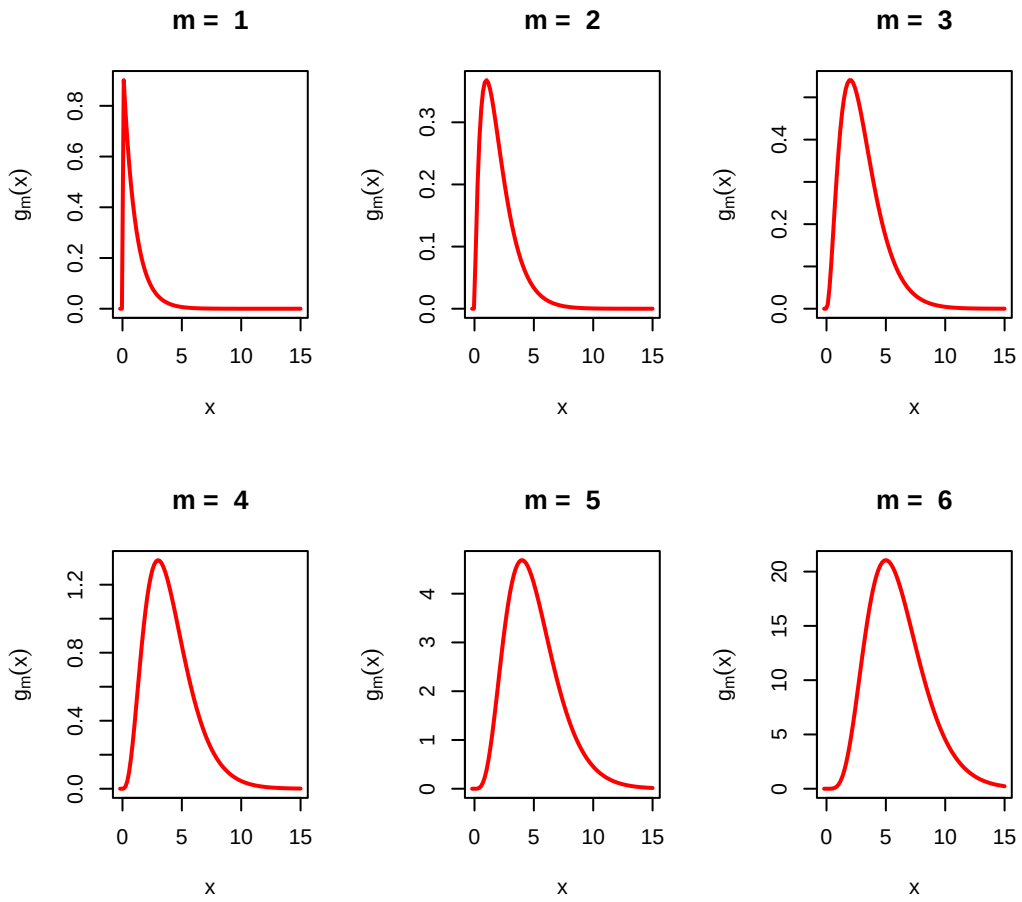


Figure 9: The shape of the function $g_m(x)$ for different choices of m. It may be noted that close to zero x^m term dominates, and for large values of x, the exponential term dominates and the PDF is exponentially decaying to zero as $x \to \infty$.

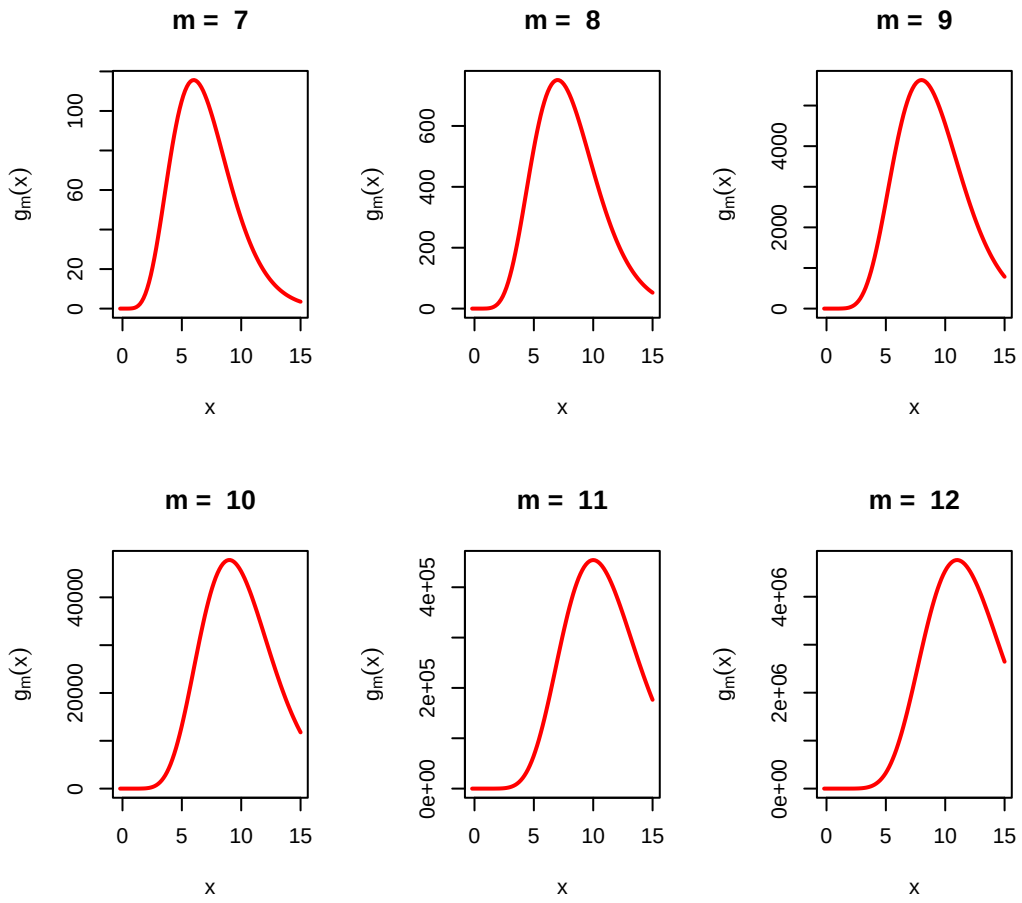


Figure 10: The shape of the function $g_m(x)$ for different choices of m. It may be noted that close to zero x^m term dominates, and for large values of x, the exponential term dominates and the PDF is exponentially decaying to zero as $x \to \infty$.

The following numerical integration output suggests that $g_m(x)$ is PDF only for $m = 1$ and $m = 2$. However, for $m \geq 3$, $\int_0^\infty g_m(x)dx \neq 1$. However, these functions can be converted to the PDF by appropriately scaling them (by dividing with the total area under the curve) to probability density functions. Consider

$$f_m(x) = \begin{cases} \frac{1}{(m-1)!} g_m(x), & 0 < x < \infty \\ 0, & \text{Otherwise} \end{cases} = \begin{cases} \frac{e^{-x}x^{m-1}}{(m-1)!}, 0 < x < \infty \\ 0, \quad \text{otherwise} \end{cases} .$$

```
for(m in m_vals){
  print(integrate(g_m, lower = 0, upper = Inf))
}
```

```
1 with absolute error < 5.7e-05
```

```
1 with absolute error < 6.4e-06
2 with absolute error < 7.1e-05
6 with absolute error < 2.6e-06
24 with absolute error < 2.2e-05
120 with absolute error < 7.3e-05
720 with absolute error < 0.0047
5040 with absolute error < 0.35
40320 with absolute error < 0.0013
362880 with absolute error < 0.025
3628800 with absolute error < 0.34
39916800 with absolute error < 3.1
```

Listing 0.6 R Code: It is important to note that we did not define the function again, rather we have passed the function $g_m(x)$ to the body of the function $f_m(x)$. Also note the use of the `factorial()` function.

```r
par(mfrow = c(2,3))
m_vals = 1:6                              # different values of m
f_m = function(x){
  g_m(x)/factorial(m-1)
}

for(m in m_vals){
  curve(f_m(x), col = "red",
        lwd = 2, main = paste("m = ", m),
        xlim = c(-0.2, 15),
        ylab = expression(f[m](x)))
  points(m, 0, pch = 19, col = "blue", cex = 1.3)
}
```

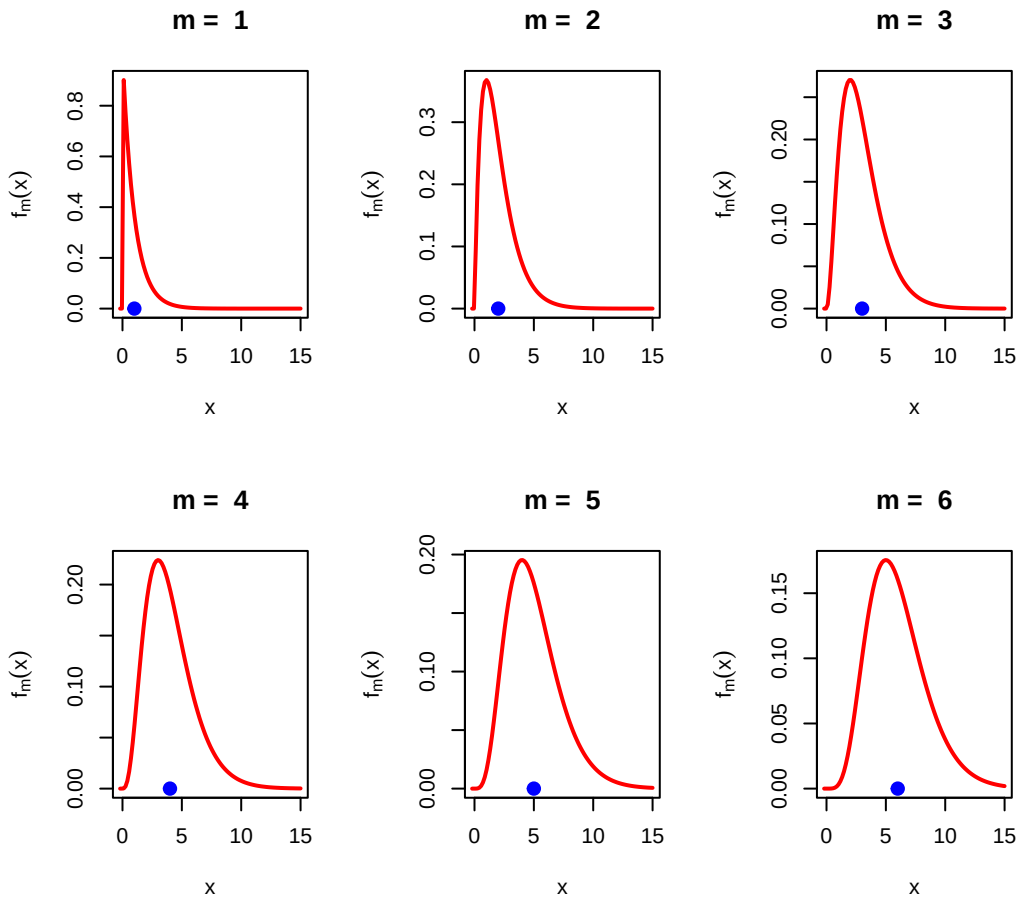


Figure 11: We first integrated the function $g_m(x)$ and the obtain $f_m(x)$ by dividing the total are under the function $g_m(x)$. It should be noted that the integration is performed in the interval $(0, \infty)$ and the support of the distribution with the PDF $f_m(x)$ is $(0, \infty)$.

Using the following R Code, we can verify that for different choices of m, the area under the function $f_m(x)$ is 1 in the interval $(0, \infty)$.

```r
m_vals = 1:6
for(m in m_vals){
  print(integrate(f_m, lower = 0, upper = Inf))
}
```

```
1 with absolute error < 5.7e-05
1 with absolute error < 6.4e-06
1 with absolute error < 3.5e-05
```

```
1 with absolute error < 4.4e-07
1 with absolute error < 9.3e-07
1 with absolute error < 6.1e-07
```

In the following, let us check whether the following function is a PDF. The function

$$g(x) = x^2 e^{-3x}, 0 < x < \infty,$$

and zero otherwise.

```
g = function(x){
  x^2*exp(-3*x)*(0<x)
}
curve(g(x), col = "red", -1,5, lwd = 2)
integrate(g, lower = 0, upper = Inf)
```

```
0.07407407 with absolute error < 4.9e-07
```

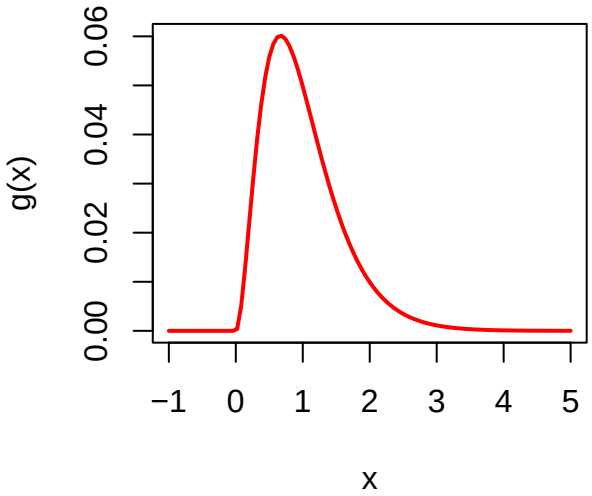


Figure 12: The graph of the function $g(x)$ and we need to check whether this function is a PDF.

The above function is not a PDF. Can we convert this function to a PDF? Yes. If

$$\int_{-\infty}^{\infty} g(x)dx = M$$

hold, then we can obtain a PDF $f(x) = \frac{1}{M}g(x)$ is a PDF.

```r
val = integrate(g, lower = 0, upper = Inf)$value
print(val)
```

```
[1] 0.07407407
```

```r
f = function(x){
  g(x)/val
}
curve(f(x), -1, 5, col = "blue", lwd = 2)
integrate(f, lower = 0, upper = Inf)
```

```
1 with absolute error < 6.6e-06
```

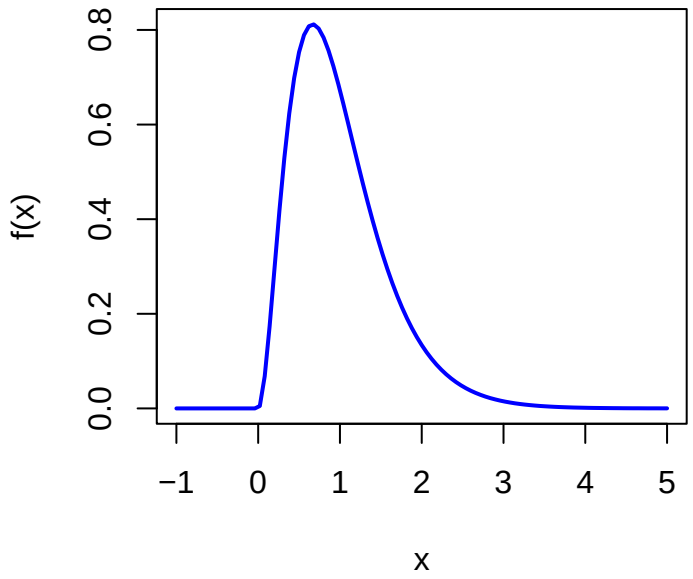


Figure 13: The shape of the PDF $f(x)$ which is obtained from the function $g(x)$, after dividing by the constant M, which is the area under the function $g(x)$. Using the integrate() function, it is verified that $f(x)$ is indeed a PDF.

! Exercises - I

Check whether the following functions are PDF or not. - $f(x) = \begin{cases} 0, & x < 0 \\ \frac{1}{(1+x)^2}, & 0 \leq x < \infty \end{cases}$.

-

Let us expand the scope of this problem. Consider the following choice of $g(x)$:

$$g(x) = e^{-\lambda x} x^{\alpha - 1}, 0 < x < \infty$$

and zero otherwise.

```
alpha = 3
lambda = 3

M = gamma(alpha)/lambda^alpha

g = function(x){
  exp(-lambda*x)*x^(alpha - 1)*(0<x)
}

f = function(x){                      # define the PDF
  g(x)/M
}

par(mfrow = c(2,3))
alpha_vals = c(1,3,5)      # filling up first row
lambda = 3
for (alpha in alpha_vals) {
  curve(g(x), -1, 8, col = "red", lwd = 2)
  legend("topright", legend = bquote(alpha == .(alpha)),
         bty = "n", cex = 1.3, lwd = 2, col = "red")
}

alpha = 3
lambda_vals = c(1,5,7)
for (lambda in lambda_vals) {
  curve(g(x), -1, 8, col = "red", lwd = 2)
  legend("topright", legend = bquote(lambda == .(lambda)),
         bty = "n", cex = 1.3, lwd = 2, col = "red")
}
```

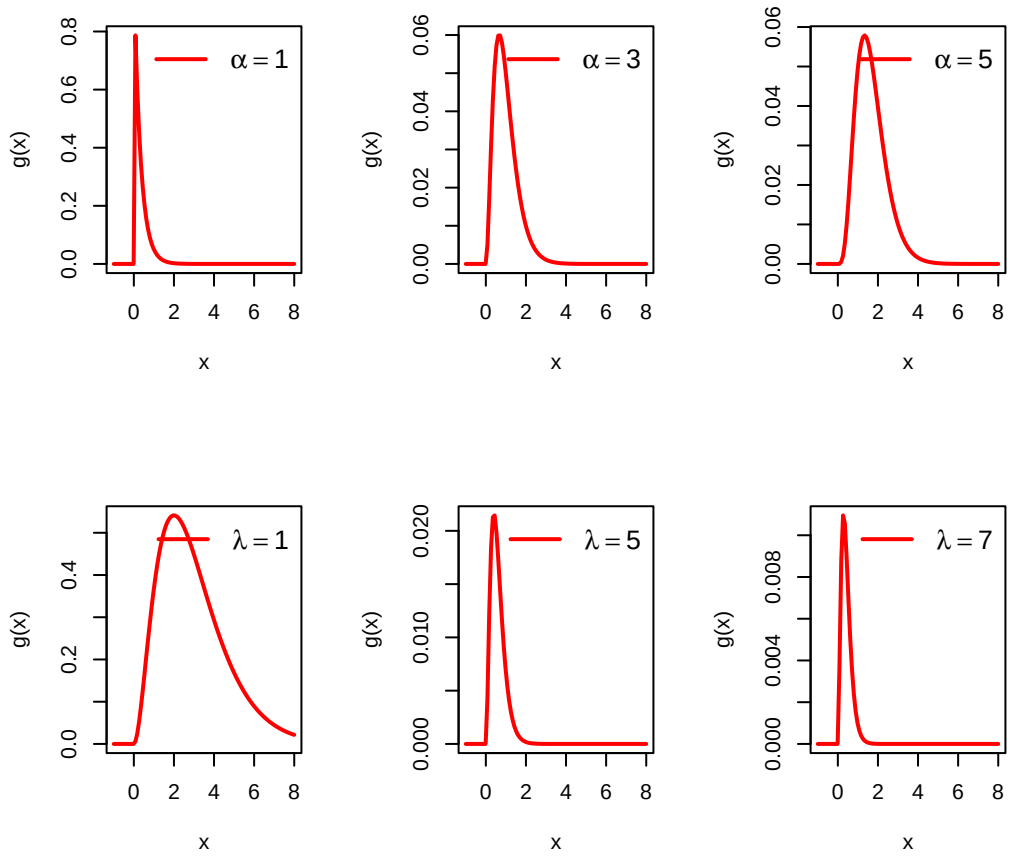


Figure 14: The shape of the fucntion $g(x)$ for different choices of α and λ.

In the following, code, we convert the function $g(x)$ to a PDF $f(x)$. Students can identify that the integral can be converted to gamma integral and

$$\int_0^\infty g(x)dx = \frac{\Gamma(\alpha)}{\lambda^\alpha}.$$

Therefore, the PDF obtained from $g(x)$ is given by

$$f(x) = \begin{cases} \frac{e^{-\lambda x}x^{\alpha-1}\lambda^\alpha}{\Gamma(\alpha)}, 0 < x < \infty, \\ 0, \text{otherwise} \end{cases}$$

```r
par(mfrow = c(2,3))
alpha_vals = c(1,3,5)      # filling up first row
lambda = 3
for (alpha in alpha_vals) {
  curve(f(x), -1, 8, col = "red", lwd = 2)
```

```
  legend("topright", legend = bquote(alpha == .(alpha)),
         bty = "n", cex = 1.3, lwd = 2, col = "red")
}

alpha = 3
lambda_vals = c(1,5,7)
for (lambda in lambda_vals) {
  curve(f(x), -1, 8, col = "red", lwd = 2)
  legend("topright", legend = bquote(lambda == .(lambda)),
         bty = "n", cex = 1.3, lwd = 2, col = "red")
}
```

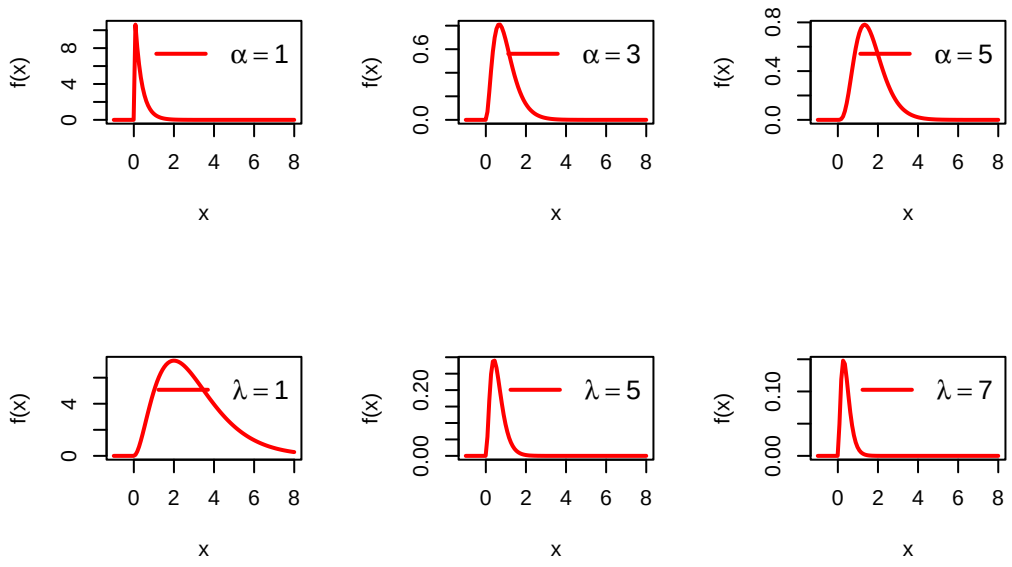
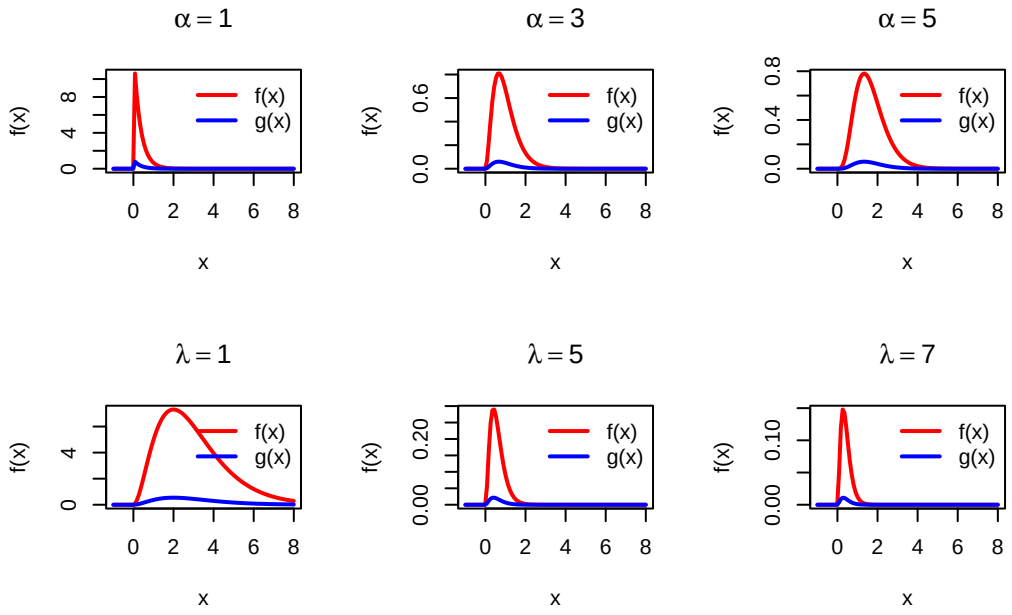


Figure 15: The shape of the probability density function $f(x)$ which is also known as the gamma distribution with parameters α and λ.

Let us plot f and g together

```
par(mfrow = c(2,3))
alpha_vals = c(1,3,5)      # filling up first row
lambda = 3
for (alpha in alpha_vals) {
  curve(f(x), -1, 8, col = "red", lwd = 2, main = bquote(alpha == .(alpha)))
  curve(g(x), add = TRUE, col = "blue", lwd = 2)
  legend("topright", legend = c("f(x)", "g(x)"),
         col = c("red", "blue"), lwd = c(2,2), bty = "n")
```

```r
}

alpha = 3
lambda_vals = c(1,5,7)
for (lambda in lambda_vals) {
  curve(f(x), -1, 8, col = "red", lwd = 2,
        main = bquote(lambda == .(lambda)))
  curve(g(x), add = TRUE, col = "blue", lwd = 2)
  legend("topright", legend = c("f(x)", "g(x)"),
       col = c("red", "blue"), lwd = c(2,2), bty = "n")
}
```




```r
mu = 1
sigma = 1
f = function(x){
  (1/(sigma*sqrt(2*pi)))*exp(-(x-mu)^2/(2*sigma^2))
}
integrate(f, lower = -Inf, upper = +Inf)
```

```
1 with absolute error < 1.6e-05
```

```r
f1 = function(x){
  x*f(x)
```

```
}
integrate(f1, lower = -Inf, upper = +Inf)
```

```
1 with absolute error < 4.4e-07
```

```
f2 = function(x){
  x^2*f(x)
}
integrate(f2, - Inf, +Inf)
```

```
2 with absolute error < 7.9e-07
```

```
M2 = integrate(f2, - Inf, +Inf)$value
M1 = integrate(f1, lower = -Inf, upper = +Inf)$value
M2 - M1^2
```

```
[1] 1
```

Exercises

1. Consider the following function

Transformation of random variables

Introduction

In the Chapter `Introduction to Probability Distribution`, we have learnt about the basic ideas about the probability mass function and the probability density function. Suppose that we have a random variable X with the probability density function $f_X(x)$ and we are interested in obtaining the PDF of $Y = g(X)$, which is a transformation of the random variable X. To start this concept, we start with an illustrative example.

Suppose that $X \sim \mathcal{N}(0, 1)$ and we aim to find the probability distribution of $Y = X^2$. We can think of the Y as the squared distance of the random variable X from 0. We define the support of a random variable X as set of all points on the real line for which the PDF $f_X(x)$ is positive and denoted by the symbol $\mathcal{X}$. Therefore,

$$\mathcal{X} = \{x \in \mathbb{R} \colon f_X(x) > 0\}.$$

Y is a transformation of X and we define the sample space of Y as

$$\mathcal{Y} = \{y \in \mathbb{R} \colon f_Y(y) > 0\}.$$

In this context we have $g(x) = x^2$, $\mathcal{X} = (-\infty, \infty)$ and $\mathcal{Y} = (0, \infty)$. Before going into the mathematical computations, let us do some simulation and try to see whether the distributions show some commonly known patterns. From an algorithmic point of view, we do the following steps:

- Fix m
- Simulate $X_1, X_2, \dots, X_m \sim \mathcal{N}(0, 1)$
- Compute $Y_1 = X_1^2, \dots, Y_m = X_m^2$.
- Draw the histogram using the values $Y_1, Y_2, \dots, Y_m$.

```r
par(mfrow = c(1,2))
x = rnorm(n = 1000, mean = 0, sd = 1)
hist(x, probability = TRUE, breaks = 30)
y = x^2
hist(y, probability = TRUE, breaks = 30)
```

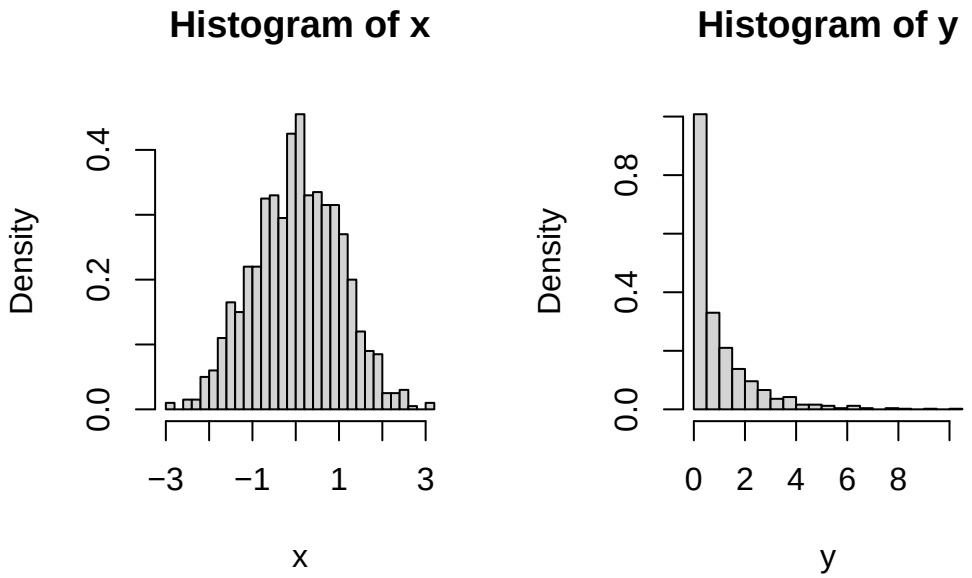


Figure 1: The histogram of the simulated realizations from the standard normal distribution is shown in the left panel. The simulated values of X are squared to obtain the realizations of Y. The distribution of realizations from the distribution of Y is shwon in the right panel.

It can be observed that the simulated realizations from the distribution of Y is highly positively skewed. Let us try to obtain the PDF of Y by explicit computation. We take the following strategy:

- Compute the CDF of Y, $F_Y(y), y \in \mathcal{Y}$.
- Take the derivative of $F_Y(y)$ to obtain the PDF $f_Y(y)$

$$\begin{aligned} F_Y(y) &= \mathrm{P}(Y \leq y) = \mathrm{P}(X^2 \leq y) && (0.1)\\ &= \mathrm{P}\left(-\sqrt{y} \leq X \leq \sqrt{y}\right) && (0.2)\\ &= 2\mathrm{P}(0 \leq X \leq \sqrt{y}) \ \text{ (even function)} && (0.3)\\ &= 2 \times \int_0^{\sqrt{y}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx && (0.4) \end{aligned}$$

It is worthwhile to remember the Leibnitz rule for differentiation under the integral sign:

$$\frac{d}{dy} \int_{a(y)}^{b(y)} \psi(x,y)dx = \psi\left(b(y),y\right) b'(y) - \psi\left(a(y),y\right) a'(y) + \int_{a(y)}^{b(y)} \frac{\partial}{\partial y}\psi(x,y)dx,$$

provided the required mathematical requirements on the function $\psi(x,y)$ is satisfied.

The PDF of Y is given by

$$f_Y(y) = \frac{e^{-\frac{y}{2}} y^{\frac{1}{2}-1}}{\Gamma\left(\frac{1}{2}\right) 2^{\frac{1}{2}}}, 0 < y < \infty,$$

and zero otherwise. The important observation is that the square of the standard normal distribution belongs to the $\mathcal{G}(\alpha, \beta)$ family of distributions. Let us expand this problem in a two dimensional scenarios.

During the lecture, we post the following question: Suppose that we consider the two dimensional plane and call (x_1, x_2) plane. We have $X_1 \sim \mathcal{N}(0,1)$ and $X_2 \sim \mathcal{N}(0,1)$ and we are interested to compute the probability that $\mathrm{P}\left(X_1^2 + X_2^2 \leq 1\right)$. Let us understand this probability statement in a simple language: Suppose we randomly choose a point on the (x_1, x_2) plane, where each coordinate is chosen randomly and independently from the $\mathcal{N}(0,1)$ distribution, what is the probability that the selected point will fall within the circle of unit radius. There are two strategies to compute this probability.

First idea is more intuitive rather than mathematically rigorous. We consider the following steps to be performed using R

- Fix m
- Randomly select $X_1 \sim \mathcal{N}(0,1)$
- Randomly select $X_2 \sim \mathcal{N}(0,1)$
- Plot the point on the (x_1, x_2) plane and compute the squared distance $Y = X_1^2 + X_2^2$
- If $Y <= 1$, then set counter $= 1$, otherwise set counter $= 0$.
- Repeat steps - II to IV m times and compute $\frac{\text{counter}}{m}$, which will be approximately equal to the desired probability.

```r
par(mfrow = c(1,2))
m = 1000
x1 = rnorm(n = m, mean = 0, sd = 1)
x2 = rnorm(n = m, mean = 0, sd = 1)
plot(x1, x2, pch = 19, col = "grey",
     xlab = expression(x[1]), ylab = expression(x[2]))
abline(h = 0, col = "red", lwd = 2)
abline(v = 0, col = "red", lwd = 2)
y = x1^2 + x2^2
curve(sqrt(1-x^2), -1 ,1, col = "blue", lwd = 2, lty = 2,
      add = TRUE)
curve(-sqrt(1-x^2), -1 ,1, col = "blue", lwd = 2, lty = 2,
      add = TRUE)
cat("Approximate probability based on m = 1000 simulated data point is ", mean(y<=1))
```

```
Approximate probability based on m = 1000 simulated data point is  0.383
```

```r
hist(y, probability = TRUE, breaks = 30)
```

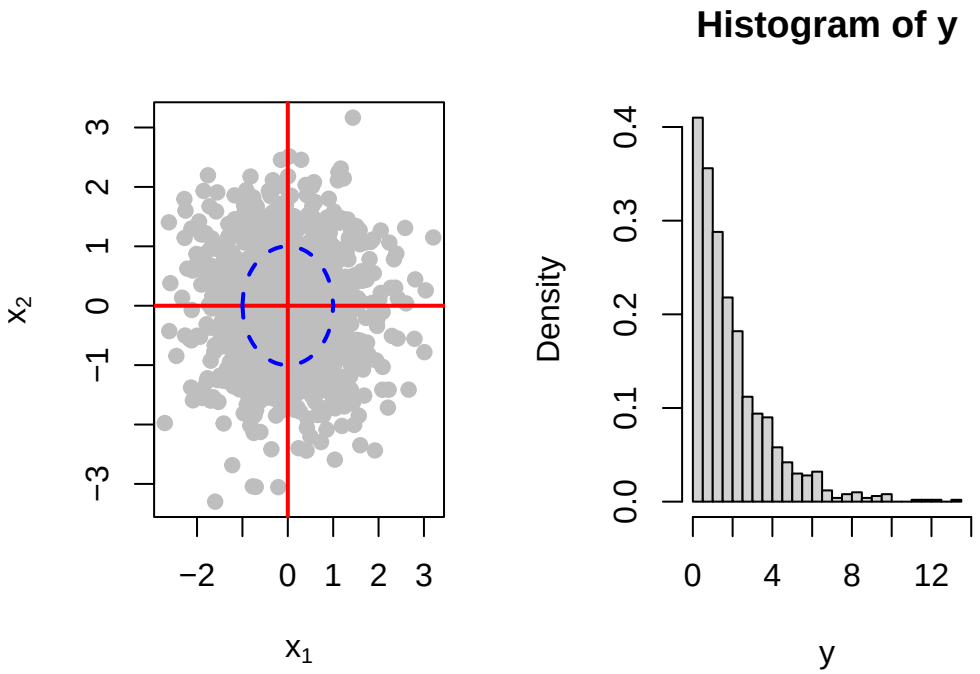


Figure 2: The histogram of the simulated realizations of Y based on 1000 simuations of pairs of independent standard normal random variables. The proportion of the number of points falling in inside the unit circle gives an approximation of the probability $\mathrm{P}(Y \leq 1)$.

The above simulation scheme suggested a highly positively skewed distribution for the random variable Y. Let us try to compute the exact PDF of Y. We have already learnt how to compute $f_Y(y)$ by computing the CDF from the definition. In this case, the following integration needs to be carried out:

$$F_Y(y) = \mathrm{P}\left(X_1^2 + X_2^2 \leq y\right) = \int\int_{\{(x_1,x_2):\, x_1^2+x_2^2 \leq y\}} \frac{1}{2\pi} e^{-\frac{1}{2}(x_1^2+x_2^2)} dx_1 dx_2.$$

You are encouraged to perform this integration, however, we can try some alternative approach as well. For example, we now understood that both X_1^2 and X_1^2 follows $\sim \mathcal{G}\left(\alpha = \frac{1}{2}, \beta = 2\right)$ and Y is nothing but sum of two independent $\mathcal{G}(\cdot, \cdot)$ distributions. We recollect the moment generating function of the $\mathcal{G}(\alpha, \beta)$, where α and β are the shape and scale parameters, respectively. The MGF is given by

$$M(t) = (1 - \beta t)^{-\alpha}, t < \frac{1}{\beta}.$$

In addition, the expected value and variance of this distribution is $\alpha\beta$ and $\alpha\beta^2$, respectively.

Suppose that $W_1 \sim \mathcal{G}(\alpha_1, \beta)$ and $W_2 \sim \mathcal{G}(\alpha_2, \beta)$ and W_1, W_2 are independent random variables and let $W = W_1 + W_2$. Then the MGF of W is can be computed as

$$\begin{aligned} M_W(t) &= \mathrm{E}\left(e^{tW}\right) = \mathrm{E}\left(e^{tW_1+tW_2}\right) && (0.5)\\ &= \mathrm{E}\left(e^{tW_1}\right)\mathrm{E}\left(e^{tW_2}\right) = M_{W_1}(t)M_{W_2}(t) \quad \text{(independence)} && (0.6)\\ &= (1-t\beta)^{-(\alpha_1+\alpha_2)}, \quad t < \frac{1}{\beta}. && (0.7) \end{aligned}$$

Therefore, $W \sim \mathcal{G}(\alpha_1 + \alpha_2, \beta)$; in particular the addition of these two independent random variables also belonging to the $\mathcal{G}$ family of distributions. Using this result, we can see that

$$Y = X_1^2 + X_2^2 \sim \mathcal{G}\left(\alpha = 1, \beta = 2\right).$$

Let us check whether the theory is matching with the simulated histograms of the Y values in the previous figure.

```
par(mfrow = c(1,2))
m = 1000
x1 = rnorm(n = m, mean = 0, sd = 1)
x2 = rnorm(n = m, mean = 0, sd = 1)
plot(x1, x2, pch = 19, col = "grey",
     xlab = expression(x[1]), ylab = expression(x[2]))
abline(h = 0, col = "red", lwd = 2)
abline(v = 0, col = "red", lwd = 2)

y = x1^2 + x2^2
hist(y, probability = TRUE, breaks = 30)
curve(dgamma(x, shape = 1, scale = 2), add = TRUE,
      col = "red", lwd = 2)
```

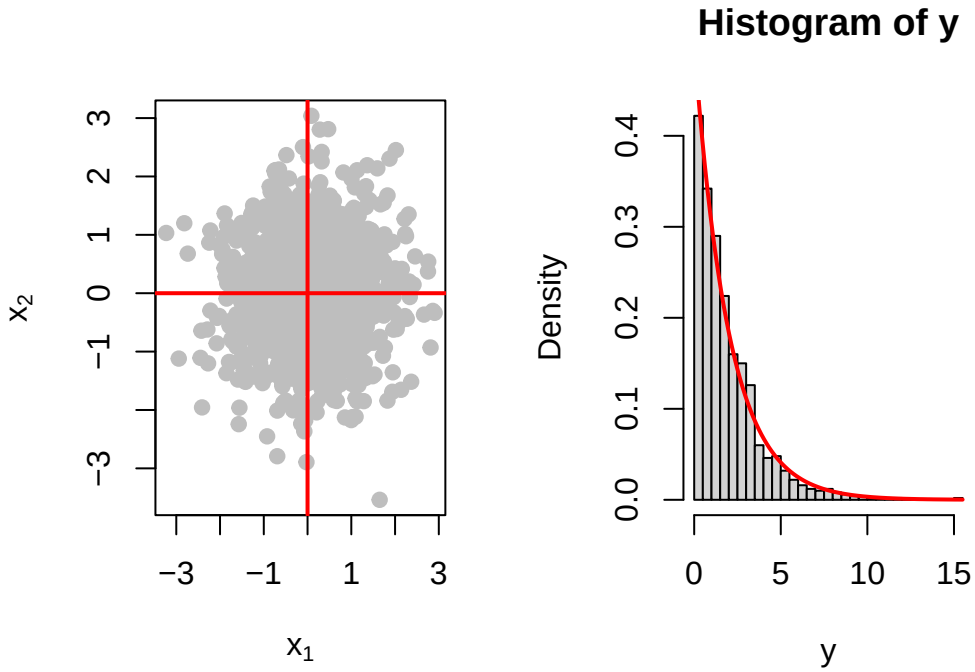


Figure 3: The histogram of the simulated realizations of Y based on 1000 simuations of pairs of independent standard normal random variables. In addition, we observe that the histogram is well approximated by the $\mathcal{G}(\alpha = 1, \beta = 2)$ PDF which basically the exponential PDF with mean 2.

I believe, now are in a position to generalize this idea. We just extend the idea using an important result which states that if $W_i \sim \mathcal{G}(\alpha_i, \beta)$ for $1 \leq i \leq n$ and they are independent, then

$$\sum_{i=1}^{n} W_i \sim \mathcal{G}\left(\alpha_1 + \cdots + \alpha_n, \beta\right).$$

Let us write them in a sequential manner:

- $X_1 \sim \mathcal{N}(0,1)$, then $Y_1 \sim \mathcal{G}\left(\alpha = \frac{1}{2}, \beta = 2\right)$
- $X_1, X_2 \sim \mathcal{N}(0,1)$, then $Y_2 = X_1^2 + X_2^2 \sim \mathcal{G}\left(\alpha = 1, \beta = 2\right)$
- $\vdots$
- $X_1, X_2, \ldots, X_n \sim \mathcal{N}(0,1)$, then $Y_n = \sum_{i=1}^{n} X_i^2 \sim \mathcal{G}\left(\alpha = \frac{n}{2}, \beta = 2\right)$

Therefore, for $n \in \mathbb{N}$, the PDF of Y_n is given by

$$f_{Y_n}(y) = \frac{e^{-\frac{y}{2}} y^{\frac{n}{2}-1}}{2^{\frac{n}{2}} \Gamma\left(\frac{n}{2}\right)}, 0 < y < \infty,$$

and zero otherwise. This probability density function is celebrated as the chi-squared distribution with n degrees of freedom and usually denoted as $Y_n \sim \chi^2_n$. From the properties of the $\mathcal{G}(\cdot,\cdot)$ distribution, we can easily conclude that

$$\mathrm{E}(Y_n) = n, \quad Var(Y_n) = \frac{n}{2} \times 2^2 = 2n.$$

Let us visualize the distribution of the Y_n for different choices of n values based on $m = 1000$ replications.

```
par(mfrow = c(2,3))
n_vals = c(2,5, 10, 30, 40, 100)
for(n in n_vals){
  m = 1000
  sim_data = matrix(data = NA, nrow = m, ncol = n)
  for(j in 1:n){
    sim_data[,j] = rnorm(n = m, mean = 0, sd = 1)
  }
  # head(sim_data)
  y = numeric(length = m)
  for (i in 1:m) {
    y[i] = sum(sim_data[i,]^2)
  }
  hist(y, probability = TRUE, main = paste("n = ", n),
       breaks = 30, col = "lightgrey")
  curve(dgamma(x, shape = n/2, scale = 2),
        add = TRUE, col = "red", lwd = 2)

}
```

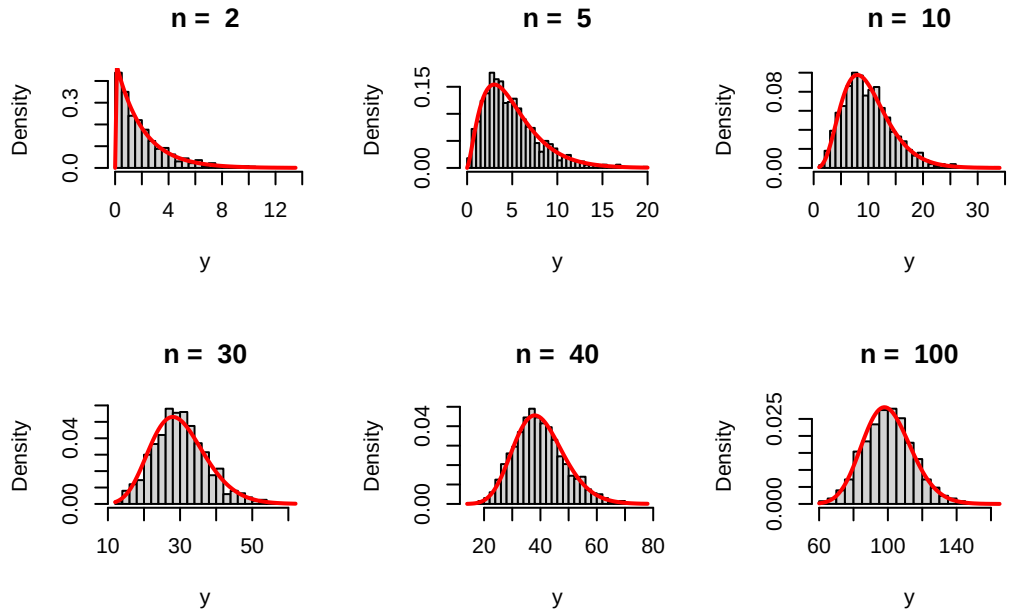


Figure 4: The histogram of the simulated realizations of Y based on 1000 simuations of n copies of independent standard normal random variables and their sum of squared values. The Chi-squared distribution with n degrees of freeom is overlaid on the histograms.

While these histograms are well approximated by the χ^2_n distribution, one student noticed that as n increases, the histograms and the PDFs are behaving similar to bell curve, that is the normal distribution. Therefore, a natural question arises, is the χ^2_n PDF looks like a bell curve for large n? In addition, what will be mean and variance of this bell curve if these distributions are well approximated by bell curve. For experiment purpose, let us draw the χ^2_n PDF and the normal PDF with mean n and variance $2n$, that is $\mathcal{N}(n, 2n)$ for different choices of n. The functions dgamma(), dchisq(), dnorm() suggest the uniform use of the letter d in plotting the density function in R.

```
par(mfrow = c(2,3))
n_vals = c(2,8,10,30,50,100)
for(n in n_vals){
  curve(dchisq(x, df = n), col = "red", lwd = 2,
        xlim = c(n-4*sqrt(2*n),n+4*sqrt(2*n)), main = paste("n = ", n),
        ylab = expression(f(x)))
  curve(dnorm(x, mean = n, sd = sqrt(2*n)),
        add = TRUE, col = "blue", lwd = 2, lty = 2)
  legend("topright", legend = c(expression(chi[n]^2), "N(n,2n)"),
         col = c("red", "blue"), lwd = c(2,2), lty = c(1,2), bty = "n")
}
```

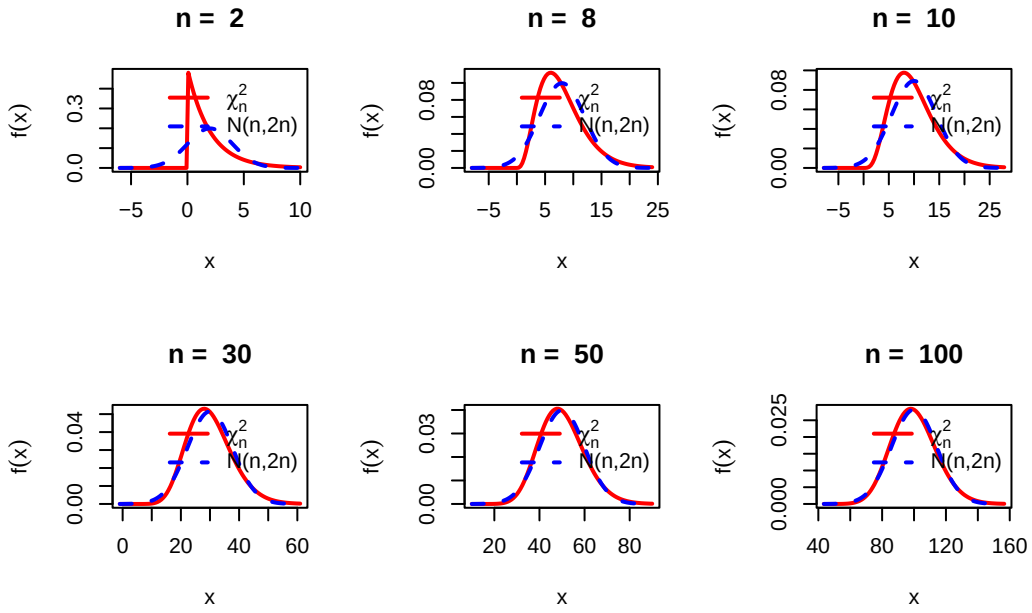


Figure 5: The figures clearly suggest that as the degrees of freedom increases, the χ^2_n distributions are well approximated by the normal distribution. The normal distributions are overlaid with dotted blue color line.

For a theoretical proof of the above observations, we can prove in the class that for large n, the Moment Generating Function of $\frac{X-n}{\sqrt{2n}}$ is approximately equal to $e^{\frac{t^2}{2}}$ which is the MGF of the standard normal distribution. Since $\frac{X-n}{\sqrt{2n}} \sim \mathcal{N}(0,1)$ for large n, therefore, $X \sim \mathcal{N}(n, 2n)$ for large n.

It is important to plot the curves of the PDFs for different choices of the parameters.

Most surprising transformation (Probability Integral Transform)

The probability integral transform is a fundamental theorem in statistical simulation. Suppose that we are interested in simulating random numbers from the distribution of X, whose CDF is given by $F_X(x)$, say. If we consider the transformation $U = F_X(X)$, basically, the transformation function $g(\cdot)$ is the CDF itself $F_X(\cdot)$, then $U \sim \text{Uniform}(0,1)$. This is quiet surprising as the result is true for any random variable. Therefore, if we simulate $U \sim \text{Uniform}(0,1)$, then we can obtain a realization from the distribution of X, by considering the following inverse transformation:

$$X = F_X^{-1}(U).$$

We must take caution in defining the inverse mapping F_X^{-1} of F_X. For example, if X is a discrete random variable, then $F_X(x)$ is a step function, therefore, the inverse can not be

defined in usual sense. We will not discuss it further here and close this session with an illustrative example using the exponential(1) distribution. The CDF is given by $F_X(x) = 1 - e^{-x}, 0 \leq x < \infty$ and zero for $x < 0$. Therefore, for $F_X^{-1}(U) = -\log(1-U), 0 < U < 1$. The following code is used to demonstrate the simulation of exponential(1) random variables starting with the Uniform$(0, 1)$ random numbers.

```
CDF_Exp = function(x){
  (1 - exp(-x))*(x>0)
}
# curve(CDF_Exp(x), -1, 6, col = "red", lwd = 2)

par(mfrow = c(2,3))
n = 1000
x = rexp(n = n, rate = 1)
hist(x, probability = TRUE)
plot(ecdf(x), ylab = expression(F[n](x)))
curve(CDF_Exp(x), -1, 6, col = "red", lwd = 2,
      add = TRUE)

U = CDF_Exp(x)
head(U)
```

```
[1] 0.13142579 0.19010268 0.07735331 0.13824538 0.83712382 0.48945245
```

```
hist(U, probability = TRUE)
curve(dunif(x), add = TRUE, col = "red", lwd = 2)

inv_CDF_Exp = function(u){
  -log(1-u)
}
U = runif(n = n)
hist(U, probability = TRUE)
x = inv_CDF_Exp(U)
plot(ecdf(x), col = "grey", ylab = expression(F[n](x)))
curve(CDF_Exp(x), -1, 6, col = "red", lwd = 2, add = TRUE)

hist(x, probability = TRUE)
```

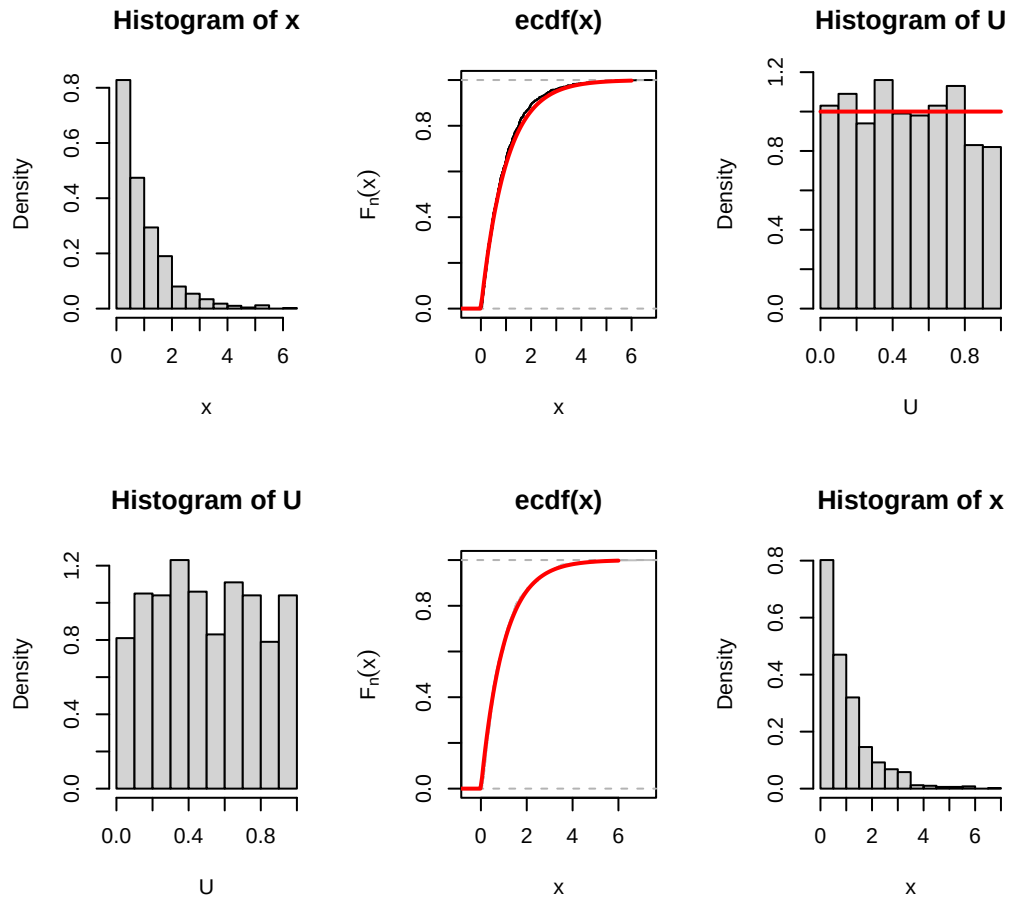


Figure 6: Top panel: We simulated observations from the Exponential(1)$ distribution using the rexp() function. Then simulated value are transformed using $F_X(\cdot)$ function, and histogram of the resulting values are drawn. It is evident that the transformed values are indeed Uniform$(0, 1)$ distributed. In the botton panel, we imulated from the Uniform$(0, 1)$ distribution and transformed them using $F_X^{-1}(\cdot)$ and the histogram of the resulting values gives the confirmation of the Exponential(1) distribution.

Maximum and Minimum of Two random variables

Suppose U_1 and U_2 are two independent uniformly distributed random variables from the interval $(0, t)$ for $t > 0$. We are interested to understand the sampling distribution of the $\max(U_1, U_2)$ and $\min(U_1, U_2)$. In the following code, we simulate the sampling distribution of these two functions. You are encouraged to simulate the sampling distributions for the maximum and minimum of n independent and identically distribution Uniform$(0, t)$ random variables. These are also called the maximum and minimum order statistics and typically denoted as $U_{(n)}$ and $U_{(1)}$, respectively.

```r
t = 3
U = runif(n=2, min = 0, max =t)
n = length(U)
U1 = min(U)
U2 = max(U)
print(U1)
```

```
[1] 1.964568
```

```r
print(U2)
```

```
[1] 2.95591
```

```r
M = 10000
U1 = numeric(length = M)
U2 = numeric(length = M)

for (i in 1:M) {
  U = runif(n=2, min = 0, max =t)
  U1[i] = min(U)
  U2[i] = max(U)
}
f_U2 = function(x){
  n*(x/t)^(n-1)*(1/t)*(0<x)*(x<t)
}

f_U1 = function(x){
  n*(1-x/t)^(n-1)*(1/t)*(0<x)*(x<t)
}
par(mfrow=c(1,3))
hist(U1, probability = TRUE, xlim = c(0,t),
     breaks = 30, xlab = expression(U[1]),
     main = expression(f[U[(1)]](u[1])))
curve(f_U1, add = TRUE, col = "red", lwd = 2)
hist(U2, probability = TRUE, xlim = c(0,t),
     breaks = 30, xlab = expression(U[2]),
     main = expression(f[U[(2)]](u[2])))
curve(f_U2, add = TRUE, col = "red", lwd = 2)


library(gplots)
```

```
Attaching package: 'gplots'


The following object is masked from 'package:stats':

    lowess
```

```r
hist2d(U1, U2, col = c("green", heat.colors(12)),
       xlab = expression(U[(1)]), ylab = expression(U[(2)]))
```

```
----------------------------
2-D Histogram Object
----------------------------

Call: hist2d(x = U1, y = U2, col = c("green", heat.colors(12)), xlab = expression(U[(1)]),
    ylab = expression(U[(2)]))

Number of data points:  10000
Number of grid bins:  200 x 200
X range: ( 0.0002043291 , 2.991646 )
Y range: ( 0.03813706 , 2.999945 )
```

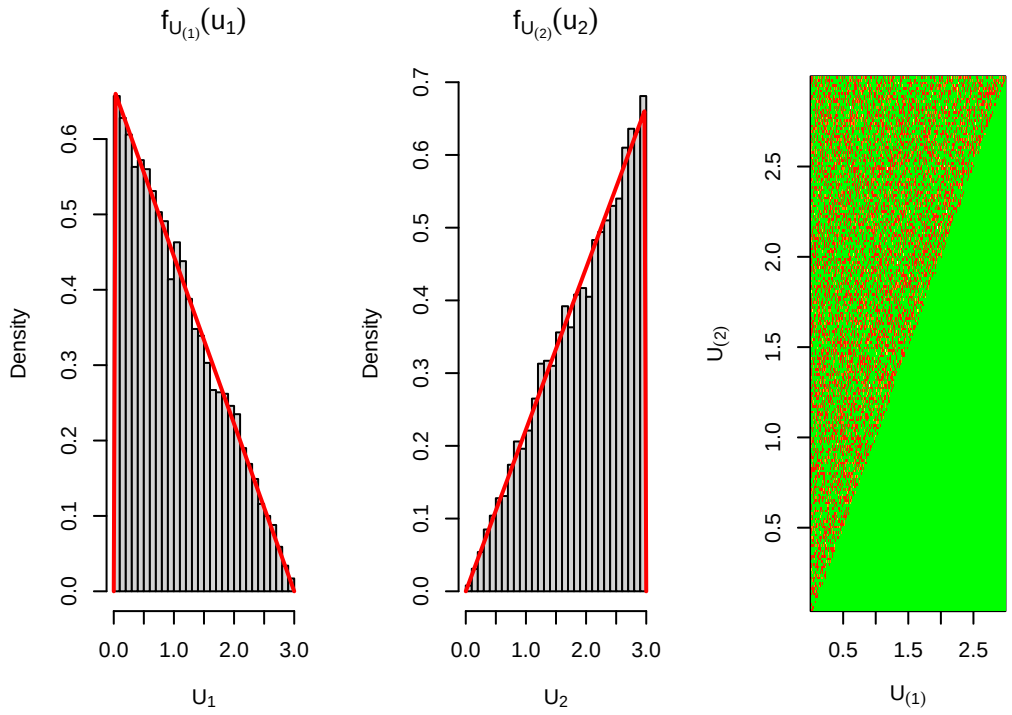


Figure 7: The sampling distributions of the maximum and minimum of two independent uniformly distributed random variables from the interval $(0, t)$. The joint distribution of these two functions of Uniform$(0, t)$ ranom variables are shown at the right most panel.

You are encouraged to execute the following code to understand that the how different is the two dimensional histogram of $\left(U_{(1)}, U_{(2)}\right)$ than the histogram of random numbers simulated from the joint distribution of (U_1, U_2)

```
# IID Uniform(0,t)
U1 = runif(n = M, min = 0, max = t)
U2 = runif(n = M, min = 0, max = t)
hist2d(U1, U2, col = c("green", heat.colors(12)),
       xlab = "U1", ylab = "U2")
```

Sampling distributions and Convergence ideas

Understanding sampling

In a typical statistics classroom, the teacher often begins by discussing either a coin-tossing experiment or by collecting data randomly from a normally distributed population, with the goal of estimating the probability of success or the population mean, respectively. In both cases, no actual statistical sampling occurs; instead, students are asked to imagine a hypothetical experiment carried out by the teacher, calculating the sample proportion or sample mean based on imagined sample data. Consequently, students often miss the actual connection between fixed sample values (obtained once data is collected) and theoretical probability density functions (which are purely mathematical constructs).

This gap in understanding could be greatly reduced if real-time random experiments were conducted in the classroom, similar to experiments in Physics, Chemistry, Biology, or Engineering courses. By utilizing software that can simulate various probability distributions, we can bridge this gap and enhance comprehension. This chapter is developed to facilitate an understanding of randomness in various computations based on sample data using computer simulations. Concepts that are often only intuitively grasped can, through simulated experiments, be transformed into concrete understanding. This document contains several R codes developed during the lectures on Statistical Computing for students enrolled in the Multidisciplinary Minor Degree Programme in Machine Learning and Artificial Intelligence, offered by the Department of Mathematics, Institute of Chemical Technology, Mumbai, under the umbrella of the National Education Policy NEP 2020. I extend my heartfelt thanks to all the students for their excellent support during the lectures and for inspiring me to simulate theoretical concepts for practical illustration.

Writing the first Computer simulation

If we set up a simulation experiment, the first task is to assume the population distribution. In this case, we start our discussion with the normal distribution and we use the symbol $f(\cdot)$ to represent the population PDF or PMF throughout this chapter. Suppose that we consider problem of estimating the mean of the population distribution $\mathcal{N}(\mu, \sigma^2)$.

```r
mu = 3                          # population mean
sigma = 1                       # population standard deviation
f = function(x){
  dnorm(x, mean = mu, sd = sigma)
}
curve(f(x), col = "red", lwd = 2, -3, 9)
```

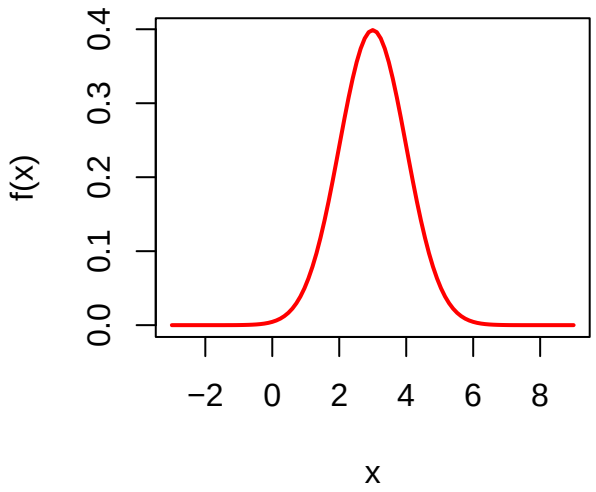


Figure 1: The population probability density function. For simulation study, we consider the population distribution characterized by the normal distribution with mean $\mu = 3$ and variance $\sigma^2 = 1$

Step - I

- Fix the sample size n
- Draw a random sample of size n from the population PDF
- Draw the sample multiple times for different choices of n and also draw the histogram for each sample data.
- Overlay the population density function on the histograms to show that the histograms are actually acting as a good approximation of the theoretical PDF.

In practice while teaching, the drawing of the histogram of the sampled data has been a very effective way to bridge the connection between the theoretical PDF and the sample.

```r
n = 50                                          # sample size
x = rnorm(n = n, mean = mu, sd = sigma)
print(x)
```

```
 [1] 3.045711 2.175084 2.005294 3.448046 3.132728 3.223940 3.835518 2.588221
 [9] 2.925667 4.478519 3.024026 2.434604 3.158217 3.853893 3.242717 3.353689
[17] 1.489799 2.569764 3.780934 1.087869 5.555160 3.269059 4.556536 1.389750
[25] 3.398520 2.077598 3.728611 2.131878 3.392468 1.269570 3.916995 5.209538
[33] 1.458062 3.250667 2.734397 2.234266 4.074366 3.279568 3.523204 2.210733
[41] 2.659411 2.941551 3.067508 3.746852 5.031223 1.753011 2.320144 1.974013
[49] 2.355845 2.549186
```

```r
hist(x, probability = TRUE)
curve(f(x), add= TRUE, col = "red", lwd = 2)
```

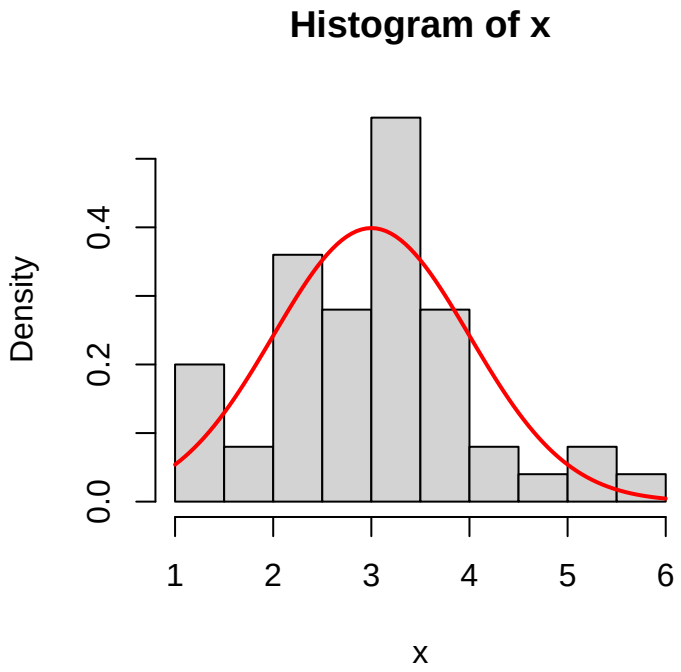


Figure 2: The histogram of the simulated data is presented and the population PDF $\mathcal{N}(3,1)$ is overlaid on this. Students are encouraged to execute the following codes for different sample sizes and check out various shapes of the histograms. In addition, you can experiment with the `breaks =` option in the `hist` function.

Step - II

- Compute Sample mean $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and also sample variance $S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X_n}\right)^2$

```r
sample_mean = mean(x)
sample_var = var(x)
cat("The sample mean and variance are\n")
```

```
The sample mean and variance are
```

```r
print(sample_mean)
```

```
[1] 2.998279
```

```r
print(sample_var)
```

```
[1] 1.01009
```

Step - III

Invariably, the sample mean will keep on changing as we execute the step - I and step - II multiple times as different run will give different set of random samples. To understand the sampling distribution of the sample mean, we repeat the above process $m = 1000$ times and approximate the actual probability density of the sample mean by using histograms.

```r
m = 1000
sample_mean = numeric(length = m)
for(i in 1:m){
  x = rnorm(n = n, mean = mu, sd = sigma)
  sample_mean[i] = mean(x)
}
hist(sample_mean, probability = TRUE, main = paste("n = ",n),
     xlab = expression(bar(X[n])))
points(mu, 0, pch = 19, col = "red", lwd = 2, cex = 1.3)
points(mean(sample_mean), 0, col = "blue", lwd = 2, cex = 1.5)


cat("The average of the 1000 many sample mean values is \n")
```

```
The average of the 1000 many sample mean values is
```

```r
print(mean(sample_mean))
```

```
[1] 2.998659
```

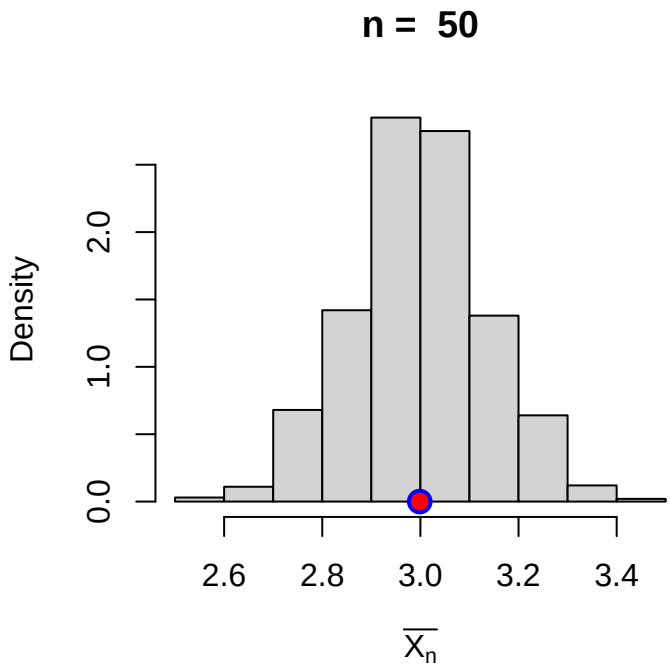


Figure 3: The sampling distribution of $\overline{X_n}$ approximated by histogram based on $m = 1000$ replications. The population distribution is governed by $\mathcal{N}(3, 1)$. Students are encouraged to modify the variables m and n and see the resulting histograms. Check that all the histograms look bell shaped.

Sampling distribution of $\overline{X_n}$

In the above simulation experiment, what are your observations if you perform the following:

- Fix m, how the shapes of histogram change as n increases. (Keep an on the x-axis of the histogram)
- Fix n. How the shapes of the histogram changes if m is small and m is large.
- Does the bell shaped pattern changes if you change μ and σ?

Step - IV

Let us do this experiment for different sample sizes and see how the distribution behaves. Our theory suggested that $\mathrm{E}(\overline{X_n}) = \mu$, that is, the sample mean is an unbiased estimator of population mean and the expected value does not depend on n.

```
par(mfrow = c(1,3))
m = 1000
n_vals = c(3,10,25)
for(n in n_vals){
  sample_mean = numeric(length = m)
  for(i in 1:m){
    x = rnorm(n = n, mean = mu, sd = sigma)
    sample_mean[i] = mean(x)
  }
  hist(sample_mean, probability = TRUE, main = paste("n = ", n),
       xlim = c(1,5))
  points(mu, 0, pch = 19, col = "red", lwd = 2, cex = 1.3)
  points(mean(sample_mean), 0, col = "blue", lwd = 2,
         cex = 1.5)
}
```

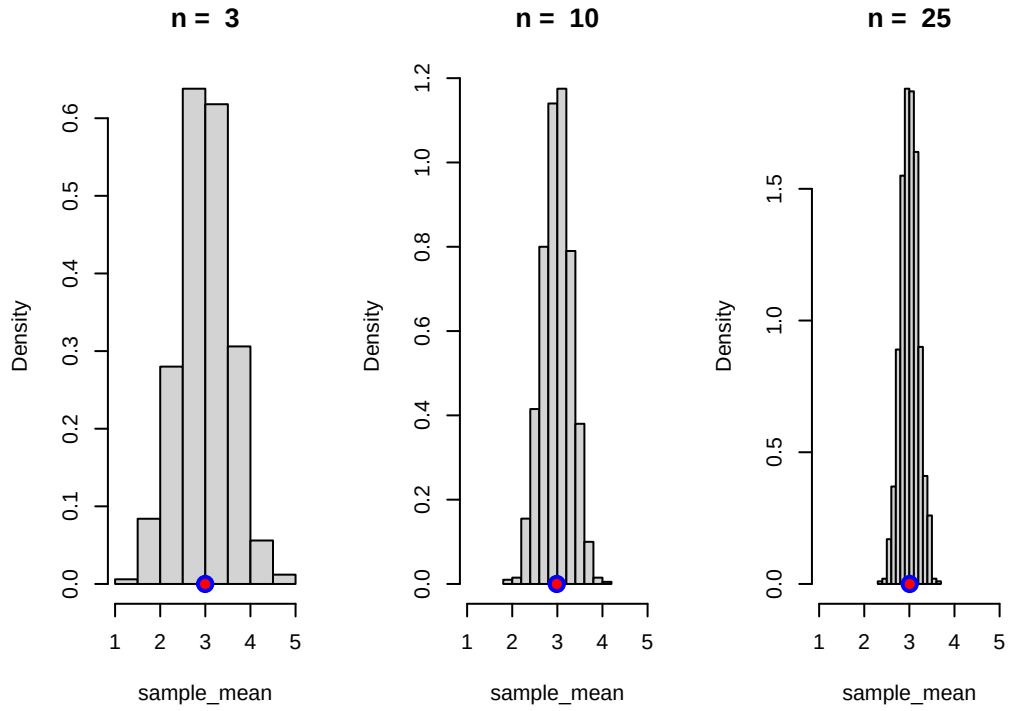


Figure 4: The sampling distribution of the sample mean for differnt choices of n. The population distribution is considered as the normal distribution with mean 3 and variance 1. It is important to note the average of the simulated sample mean values (blue colour dot) coincides with the true mean of the population (marked as red coloured dot).

i Expectation is equivalent to averaging using simulation

In the simulation (Step - IV), the true population mean is indicated using the red dot and the average of the sample means based on 1000 replication is shown using blue circle. The averaging of the sample mean values can be thought of as an expected value of $\overline{X_n}$, that is, $\mathrm{E}\left(\overline{X_n}\right)$. This simulation gives an idea that $\mathrm{E}\left(\overline{X_n}\right)$ is equal to the true population mean μ (here it is 1). Although not a mathematical proof, but such simulation outcomes gives an indication of the unbiasedness of the sample mean.

Step - V

In theory, we have computed by using Moment Generating Functions that

$$\overline{X_n} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right).$$

Let us check whether the theoretical claim can be verified by using simulation.

```r
par(mfrow = c(2,3))
n_vals = c(3, 10, 25, 50, 100, 250)
m = 1000
for(n in n_vals){
  sample_mean = numeric(length = m)
  for(i in 1:m){
    x = rnorm(n = n, mean = mu, sd = sigma)
    sample_mean[i] = mean(x)
  }
  hist(sample_mean, probability = TRUE, main = paste("n = ", n))
  points(mu, 0, pch = 19, col = "red", lwd = 2, cex = 1.3)
  points(mean(sample_mean), 0, col = "blue", lwd = 2,
         cex = 1.5)
  curve(dnorm(x, mean = mu, sd = sqrt(sigma^2/n)),
        add = TRUE, col = "red", lwd = 2)

}
```

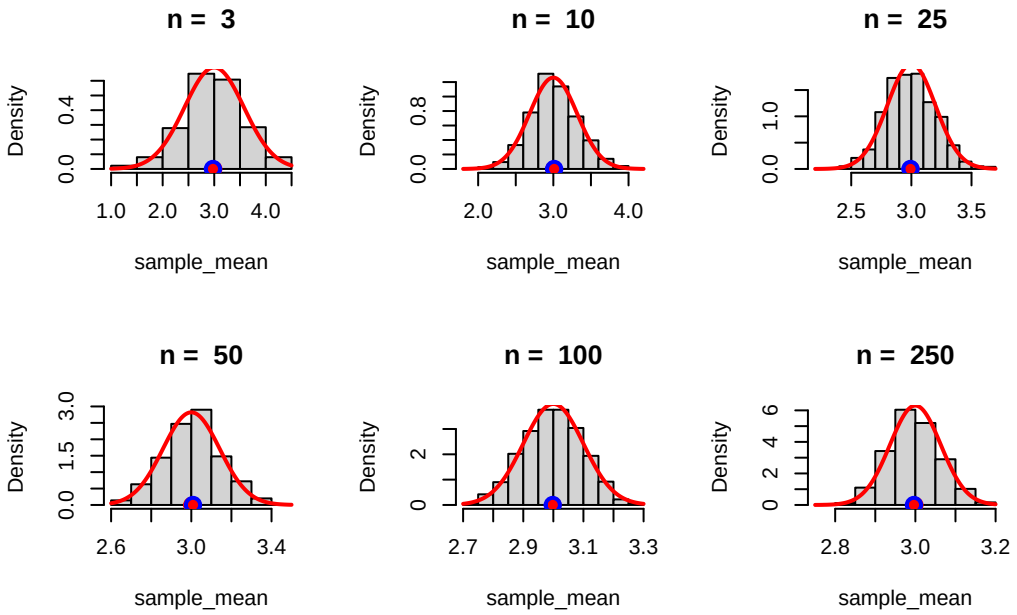


Figure 5: The theoretical PDF of the $\overline{X_n}$ is overlaid on the histograms. The histograms are obtained based on 1000 replications for different sample sizes. For simulation purpose, the population parmaeters have been fixed at $\mu = 3$ and $\sigma^2 = 1$.

A natural question asked by the students was that how we were actually writing $\mathrm{E}(X_1) = \mu$

and $Var(X_1) = \sigma^2$. In fact, the question is much deeper which states that what do we really mean by writing $X_1, X_2, \dots, X_n$ are independent and identically distributed random variables each having the same population distribution.

$$X_1, X_2, \dots, X_n \sim \mathcal{N}(\mu, \sigma^2)$$

From a simulation perspective, if we want to check whether X_1 follows the same population distribution, we need to repeat the sampling $m = 1000$ times (say) and from each sample of size n, we record the first entry, which is a realization of X_1.

In the following, we do this process for $n = 3$. Let us write down the steps in algorithmic way:

- Fix n (sample size)
- Fix M (number of replications)
- For each $m \in \{1, 2 \dots, M\}$
  - Simulate $X_1^{(m)}, X_2^{(m)}, X_3^{(m)} \sim \mathcal{N}(\mu, \sigma^2)$
- Draw histograms of $\left\{X_j^{(m)}\right\}_{m=1}^{M}$ for $j \in \{1, 2, 3\}$.
- Overlay $\mathcal{N}(\mu, \sigma^2)$ on each histogram.
- All three histograms matches closely approximate the population distribution.

```
n = 3
M = 1000
mu = 3; sigma = 1
x1_vals = numeric(length = M)
x2_vals = numeric(length = M)
x3_vals = numeric(length = M)
for(i in 1:M){
  x = rnorm(n = n, mean = mu, sd = sigma)
  x1_vals[i] = x[1]
  x2_vals[i] = x[2]
  x3_vals[i] = x[3]
}
par(mfrow = c(1,3))
hist(x1_vals, probability = TRUE, xlab = expression(X[1]),
     main = paste("n = ", n))
curve(dnorm(x, mean = mu, sd = sigma), add = TRUE,
      col = "red", lwd = 2)
hist(x2_vals, probability = TRUE, xlab = expression(X[2]),
     main = paste("n = ", n))
curve(dnorm(x, mean = mu, sd = sigma), add = TRUE,
```

```
      col = "red", lwd = 2)
hist(x3_vals, probability = TRUE, xlab = expression(X[3]),
     main = paste("n = ", n))
curve(dnorm(x, mean = mu, sd = sigma), add = TRUE,
      col = "red", lwd = 2)
```

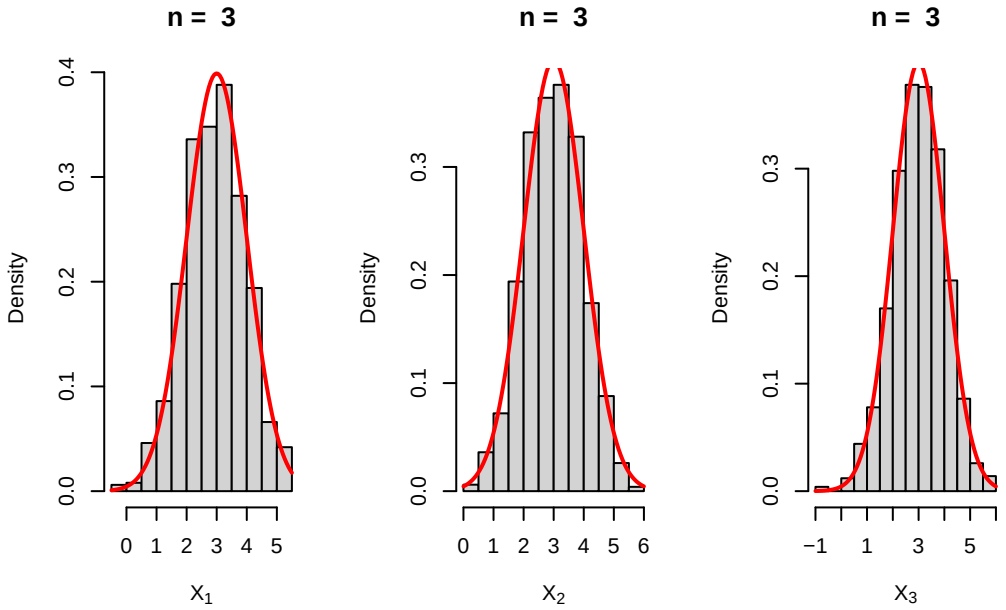


Figure 6: The histogram obtained from the realizations of X_1 is well approximated by the population PDF. This is same for the X_2 and X_3. In addition, you can draw a pairwise scatterplot of these values to check that correlations among them are very close to zero.

Convergence in Probability

Suppose that $\{X_n\}$ be a sequence of random variables defined on some probability space $(\mathcal{S}, \mathcal{B}, \mathrm{P})$ and X be another random variable defined on the same probability space. We say that the sequence $\{X_n\}$ converges to X in probability, denoted as $X_n \xrightarrow{P} X$, if for any $\epsilon > 0$ (however small), the following condition holds

$$\lim_{n\to\infty} \mathrm{P}\left(|X_n - X| \geq \epsilon\right) = 0.$$

Basically for every X_n, we construct a sequence of real numbers (a_n) obtained by

$$a_n = \mathrm{P}\left(|X_n - X| \geq \epsilon\right) = \int\int_{\{(x_n,x):\, |x_n - x| \geq \epsilon\}} f_{X_n,X}\left(x_n, x\right) dx_n dx.$$

If $\lim_{n\to\infty} a_n = 0$, then the sequence of random variables $\{X_n\}$ converges to X in probability. Let us consider a simple example.

Suppose $\{X_n\}$ be a sequence of independent random variables having Uniform$(0,\theta), \theta \in (0,\infty)$ distribution. Consider the following sequence of random variables

$$Y_n = \max(X_1, X_2, \ldots, X_n)$$

which is also known as the maximum order statistics. We show that $Y_n \to \theta$ in probability. First of all, we compute the sampling distribution of Y_n. We first find the CDF of Y_n and by differentiating the CDF, we can obtain the PDF $f_{Y_n}(y)$.

$$\begin{aligned} F_{Y_n}(y) &= \mathrm{P}\left(Y_n \le y\right) = \mathrm{P}\left(\max(X_1, \ldots, X_n) \le y\right) && (0.1)\\ &= \mathrm{P}\left(X_1 \le y, X_2 \le y, \ldots, X_n \le y\right) && (0.2)\\ &= \mathrm{P}\left(X_1 \le y\right) \cdots \left(X_n \le y\right) && (0.3)\\ &= \left(\mathrm{P}\left(X_1 \le y\right)\right)^n = \left(\tfrac{y}{\theta}\right)^n, 0 < y < \theta. && (0.4) \end{aligned}$$

By differentiating the CDF, we obtain the PDF as

$$f_{Y_n}(y) = n\left(\frac{y}{\theta}\right)^{n-1}\frac{1}{\theta}, 0 < y < \theta,$$

and zero, otherwise. Before going to the mathematical proof, let us consider the following simulation exercises. In the following we obtain the sampling distribution of Y_n based on computer simulation using the following algorithm:

- Fix n.
- Fix θ.
- Fix m, the number of replications
- Simulate $X_1, X_2, \ldots X_n \sim$ Uniform$(0,\theta)$
- Compute $Y_n = \max(X_1, X_2, \ldots, X_n)$
- Repeat the previous two steps m times to obtain $Y_n^{(1)}, \ldots, Y_n^{(m)}$.
- Draw the histogram of the values $\left\{Y_n^{(j)}\right\}_{j=1}^{m}$.
- Overlay the exact PDF $f_{Y_n}(y)$ on the histogram.

In the above step, at each step $j \in \{1, 2, \ldots, m\}$ we fix `set.seed(j)`, so that the figures are reproducible. In the following codes, we do the same experiment for minimum order statistics as well which is given by

$$Y_1 = \min(X_1, X_2, \ldots, X_n).$$

```r
theta = 1
n = 10
x = runif(n = n, min = 0, max = theta)
# print(x)
print(max(x))
```

```
[1] 0.9239462
```

```r
m = 1000
y_1 = numeric(length = m)
y_n = numeric(length = m)
for(i in 1:m){
  set.seed(i)
  x = runif(n = n, min = 0, max = theta)
  y_1[i] = min(x)
  y_n[i] = max(x)
}
par(mfrow = c(1,2))
hist(y_1, probability = TRUE, main = paste("n = ", n),
     xlab = expression(Y[1]), breaks = 30)
curve(n*(1-x/theta)^(n-1)*(1/theta), add= TRUE, col = "red",
      lwd = 2)
hist(y_n, probability = TRUE, main = paste("n = ", n),
     xlab = expression(Y[n]), breaks = 30)
curve(n*(x/theta)^(n-1)*(1/theta),
      add = TRUE, col = "red", lwd = 2)
```

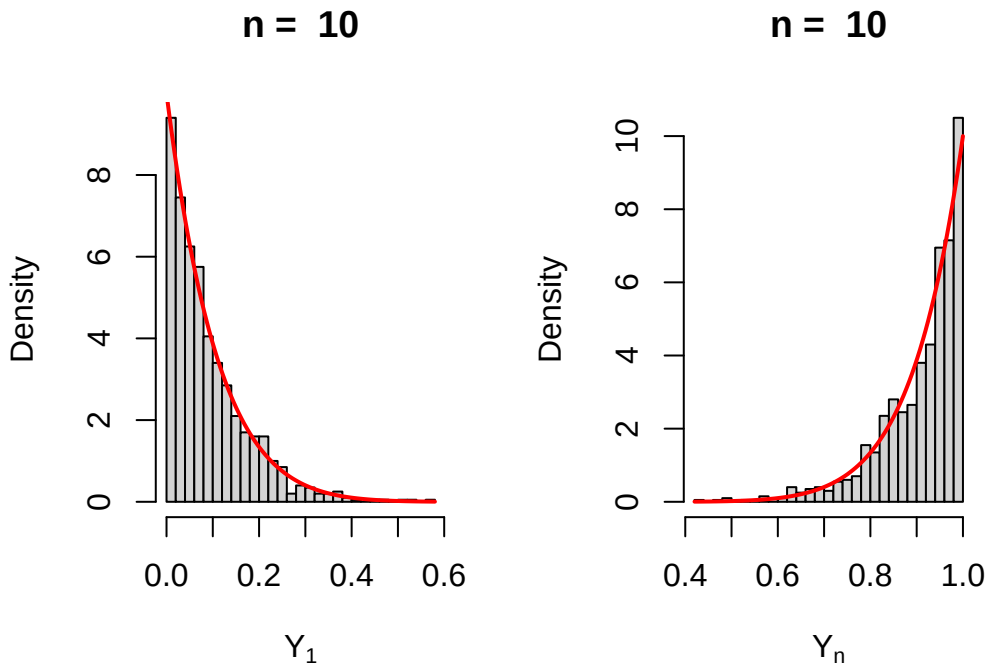


Figure 7: We simulate the sampling distribution of the maximum and minimum order statistics. Y_n and Y_1. The sampling distribution of Y_n is more concentrated towards θ, and Y_1 is more concentrated towards 0. The exact distribution is overlaid on the histograms obtained from simulation studies and the simulation agrees with the theoretical claim.

The exact sampling distribution of the minimum order statistics Y_1 can be derived in a similar fashion. First, we compute the CDF of Y_1.

$$\begin{aligned} F_{Y_1}(y) &= \mathrm{P}(Y_1 \leq y) = 1 - \mathrm{P}(Y_1 > y) && (0.5)\\ &= 1 - \mathrm{P}\left(\min(X_1, X_2, \ldots, X_n) > y\right) && (0.6)\\ &= 1 - \mathrm{P}\left(X_1 > y, \ldots, X_n > y\right) && (0.7)\\ &= 1 - \prod_{i=1}^{n} \mathrm{P}\left(X_i > y\right) = 1 - \left(1 - \mathrm{P}(X_1 \leq y)\right)^n = 1 - \left(1 - \frac{y}{\theta}\right)^n, 0 < y < \theta. && (0.8) \end{aligned}$$

The PDF of Y_1 is given by

$$f_{Y_1}(y) = \begin{cases} n\left(1 - \frac{y}{\theta}\right)^{n-1} \frac{1}{\theta}, & 0 < y < \theta \\ 0, & \text{Otherwise.} \end{cases}$$

An alternative visualization

In the above explanation, we have fixed the sample size n and simulated the observations from the sampling distribution of Y_1 and Y_n. The process is repeated 1000 times and check whether the resulting histograms are well approximated by the theoretically derived PDFs. In the following, we vary $n \in \{1, 2, 3 ...\}$ and for each n, we simulate a random sample of size n and compute Y_1 and Y_n. Therefore, in this scheme, we do not have any replications. We plot the sequences $\{Y_1^{(1)}, Y_1^{(2)}, ...\}$ and $\{Y_n^{(1)}, Y_n^{(2)}, ...\}$ values against $n \in \{1, 2, 3 ...\}$. This will give an idea, as sample size increases, whether these random quantities converge to some particular values.

```
n_vals = 1:100
y_1 = numeric(length = length(n_vals))
y_n = numeric(length = length(n_vals))
for(n in n_vals){
  x = runif(n = n, min = 0, max = theta)
  y_1[n] = min(x)
  y_n[n] = max(x)
}
plot(n_vals, y_n, type = "p", col = "red", lwd = 2,
     xlab = "sample size (n)", ylim = c(0,1), ylab = " ")
lines(n_vals, y_1, col = "blue", type = "p", lwd = 2)
legend(50, 0.5, legend = c(expression(Y[1]), expression(Y[n])),
       col = c("blue", "red"), bty = "n", lwd =c(2,2))
```

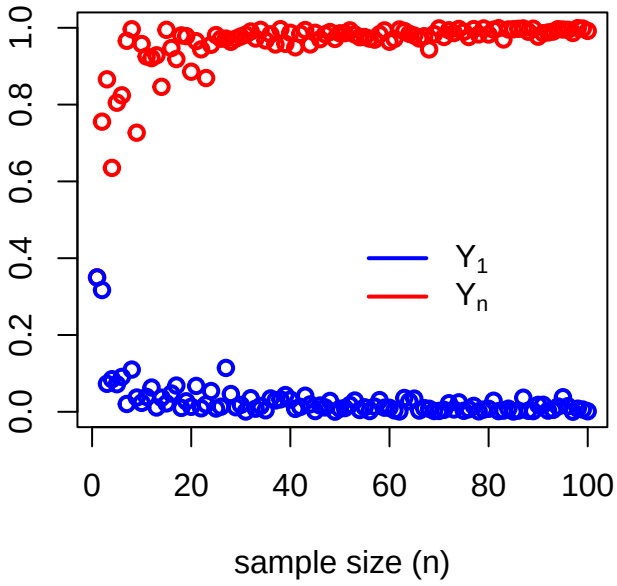


Figure 8: Random sample of size n is simulated from the uniform distribution from Uniform$(0, \theta)$. $\theta = 1$ is assumed for simulation. Simulated realization of the minimum and maximum order statistics are shown in the graph. It is observed $Y_n = \max(X_1, \ldots, X_n) \to 1$ and $Y_1 = \min(X_1, \ldots, X_n) \to 0$ as $n \to \infty$.

Central Limit Theorem

The Central Limit Theorem is one of the most fundamental concepts in Statistics which has profound applications across all domains of Data Science and Machine Learning. It says that the sample mean is approximately normally distributed for large sample size n. However, we shall see it more critically and what are the assumptions needed to understand it in a better way. We shall see some mathematical proof and a lots of simulation to understand this idea better. To understand CLT, first we understand the concept of the convergence in distribution. We elaborate it using a couple of motivating examples along with visualization using R.

Convergence in Distribution

Suppose that $X_1, X_2, \ldots, X_n$ be a sequence of random variables follows Uniform$(0, \theta)$ distribution, where $\theta \in (0, \infty) = \Theta$ (say). Suppose that Y_n be the maximum order statistics whose CDF is given by

$$F_n(x) = \mathrm{P}(Y_n \leq x) = \begin{cases} 0, & -\infty < x < 0, \\ \left(\frac{x}{\theta}\right)^n, & 0 \leq x < \theta, \\ 1, & \theta \leq x < \infty. \end{cases}$$

We are interested to know as $n \to \infty$, how the function $F_n(x)$ behaves at each value of x. If is easy to observe that

- If $x < 0$, $\lim n \to \infty F_n(x) = 0$.
- If $0 \leq x < \theta$, then $\lim_{n\to\infty} F_n(x) = 0$.
- If $\theta \leq x < \infty$, then $\lim_{n\to\infty} F_n(x) = 1$.

Therefore, the limiting CDF is given by the following function which represents the CDF of a random variable X which takes the value θ with probability 1.

$$F_X(x) = \begin{cases} 1, \theta \leq x < \infty \\ 0, -\infty < x < \theta. \end{cases}$$

In the following, we see visualize the convergence graphically. It is important to note that each $F_n(x)$ is a continuous function for $n \in \{1, 2, 3 ...\}$, however the limiting CDF is not continuous at $x = 0$. We say that the maximum order statistics $Y_n = \max(X_1, X_2, \dots, X_n)$ converges in distribution to the random variable X.

```
theta = 2
n = 1
F_n = function(x){
  (x/theta)^n*(0<=x)*(x<theta) + (theta<=x)
}
curve(F_n(x), col = 2, lty = 2, lwd = 2, -0.5, 2.5,
      ylab = expression(F[n](x)))
n_vals = c(3,5,10,25)
for(n in n_vals){
  curve(F_n(x), col = n, lty = 2, lwd = 2, add = TRUE)
}
points(0, 0, pch = 19, col = "red", cex = 1.3)
points(theta, 1, pch = 19, col = "red", cex = 1.3)

segments(-05,0, theta,0, col = "blue", lwd = 3)
segments(theta,1, 2.5,1, col = "blue", lwd = 3)
legend("topleft", legend = c("n = 1", "n = 3", "n = 5", "n = 10", "n = 25"),
       col= c(2,3,5,10,25), bty = "n", lwd = 2, lty = rep(2,5))
```

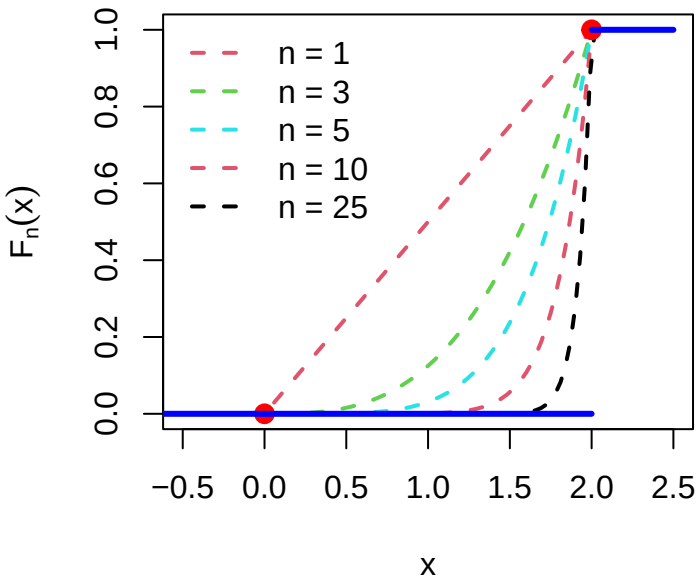


Figure 9: As the sample size increases, the function $F_n(x)$ converges to the function $F_X(x)$ at all points x except 0. However, note that the point $x = 0$, is a point of discontinuity.

In the light of the above example, let us formalize the concept of the convergence in distribution.

! Convergence in Distribution

A sequence of random variables X_n defined on some probability space $(\mathcal{S}, \mathcal{B}, \mathrm{P})$ with CDF $F_n(x)$ is said to converge to distribution to another random variable X having CDF $F_X(x)$, if the following holds

$$\lim_{n\to\infty} F_n(x) = F_X(x)$$

at all points x where the function $F_X(x)$ is continuous.

An important point to be noted here that for checking the convergence in distribution, we need to check on the convergence at all the points where the function $F_X(x)$ is continuous.

Let us now see another example of the convergence in distribution. Suppose that

$$X_n \sim \sqrt{n} \times \text{Uniform}\left(0, \frac{1}{n}\right).$$

Let $U_n \sim \text{Uniform}\left(0, \frac{1}{n}\right)$ and then $X_n = \sqrt{n} \times U_n$.

- U_n shrinks towards 0 as $n \to \infty$.
- It is being amplified again by multiplying $\sqrt{n}$.

Therefore, X_n is basically a product of two components, one is exploding as $n \to \infty$ and another component is shrinking as $n \to \infty$.

- Let us investigate the U_n component as follows through some visualization. The CDF and PDF of U_n is given by

$$f_{U_n}(x) = \begin{cases} n, 0 < x < \frac{1}{n} \\ 0, \text{otherwise} \end{cases}$$

$$F_{U_n}(x) = \begin{cases} 0, -\infty < x < 0 \\ nx, 0 \leq x < \frac{1}{n} \\ 1, \frac{1}{n} \leq x < \infty \end{cases}$$

```r
par(mfrow = c(1,2))
n = 1
f_n = function(x){
  n*(0<x)*(x<1/n)
}
curve(f_n(x), 0, 1, lwd = 2, col = n, ylim = c(0,11),
      lty = 2, ylab = expression(f[n](x)))
n_vals = c(3,5,8,10)
for(n in n_vals){
  curve(f_n(x), add = TRUE, lwd = 2, col = n, lty = 2)
}
legend("topright", legend = c("n = 1", "n = 3", "n = 5", "n = 8", "n = 10"),
       col= c(1,n_vals), bty = "n", lwd = 2, lty = rep(2,5))

F_n = function(x){
  n*x*(0<x)*(x<1/n) + (1/n<=x)
}
n = 1
curve(F_n(x), lwd = 2, col = n, -0.1, 1.1,
      lty = 2, ylab = expression(F[n](x)))
n_vals = c(3,5,8,10)
for(n in n_vals){
  curve(F_n(x), add = TRUE, lwd = 2, col = n, lty = 2)
}
legend("bottomright", legend = c("n = 1", "n = 3", "n = 5", "n = 8", "n = 10"),
       col= c(1,n_vals), bty = "n", lwd = 2, lty = rep(2,5))
segments(-0.1,0, 0,0, col = "blue", lwd = 3)
segments(0,1, 1.5,1, col = "blue", lwd = 3)
points(0,1, pch = 19, col = "blue", cex = 1.2)
```

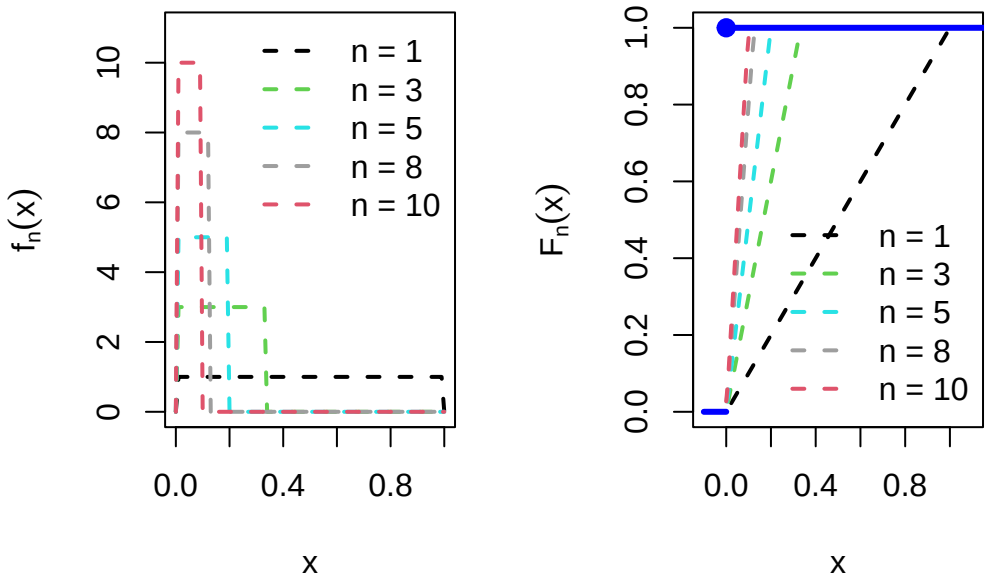


Figure 10: As $n \to \infty$, $f_n(x)$ becomes highly concentrated at 0 and $F_n(x)$ approaches to the CDF of a random variable X which takes 0 with probability 1.

Therefore, U_n converges in distribution to 0 and we are now ready to check that if we multiply $\sqrt{n}$ with U_n, will it still go to zero? In the following let us consider the simulation of U_n for different values of n and also compute X_n and see how these simulated numbers behave as $n \to \infty$.

```
n = 1000
U_n = numeric(length = n)
for(i in 1:n){
  U_n[i] = runif(n = 1, min =0, max = 1/i)
}
plot(1:n, U_n, xlab = "n", type = "l", col = "blue",
     lwd = 2, ylab = " ")
X_n = numeric(length = n)
for(i in 1:n){
  X_n[i] = sqrt(i)*U_n[i]
}
lines(1:n, X_n, col = "red", lwd = 2)
legend("topright", legend = c(expression(U[n]), expression(X[n])),
       col = c("blue", "red"), lwd = c(2,2), bty = "n")
```

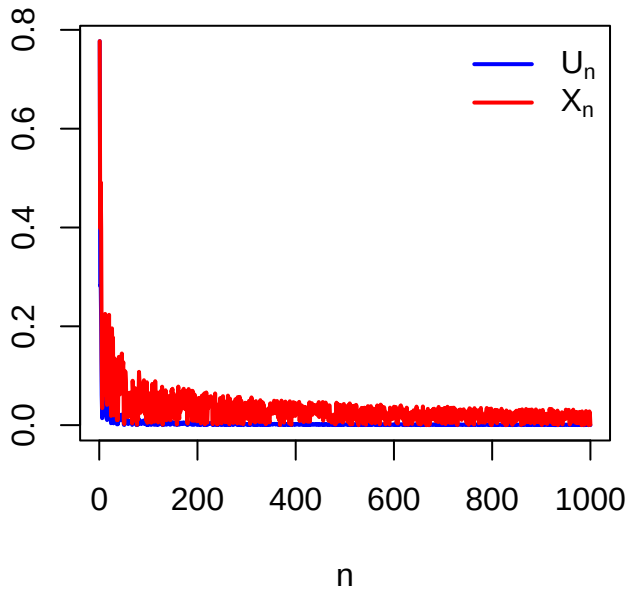


Figure 11: Simulations suggest that both U_n and X_n converges to 0, but converges of X_n to 0 is slower. Convergence of U_n and X_n are shown in blue and red colour only.

We can expand the scope of this problem by expanding the definition of X_n as follows:

$$X_n = n^{\delta} U_n, \delta > 0.$$

In the following, we choose different values of δ and see how to simulations behave. The following simulations suggest that

- $0 < \delta < 1$, $X_n \to 0$ in probability and distribution as well.
- $\delta = 1$, $X_n \to \text{Uniform}(0, 1)$ as $n \to \infty$.
- $\delta > 1$, X_n does not converges as $n \to \infty$.

```
delta_vals = c(0.1, 0.3, 0.5, 0.8, 1, 1.2)
par(mfrow = c(2,3))
for(delta in delta_vals){
  n = 1000
  U_n = numeric(length = n)
  for(i in 1:n){
    U_n[i] = runif(n = 1, min =0, max = 1/i)
  }
  X_n = numeric(length = n)
  for(i in 1:n){
    X_n[i] = i^delta*U_n[i]
  }
```

```r
  plot(1:n, X_n, xlab = "n", ylab = "",
     type = "l", col = "red", main = bquote(delta == .(delta)),
     ylim = c(0,1))
  lines(1:n, U_n, col = "blue", pch = 19, lwd = 2)
  legend("topright", legend = c(expression(U[n]), expression(X[n])),
         col = c("red", "blue"), lwd = c(2,2),
         bty = "n")
}
```

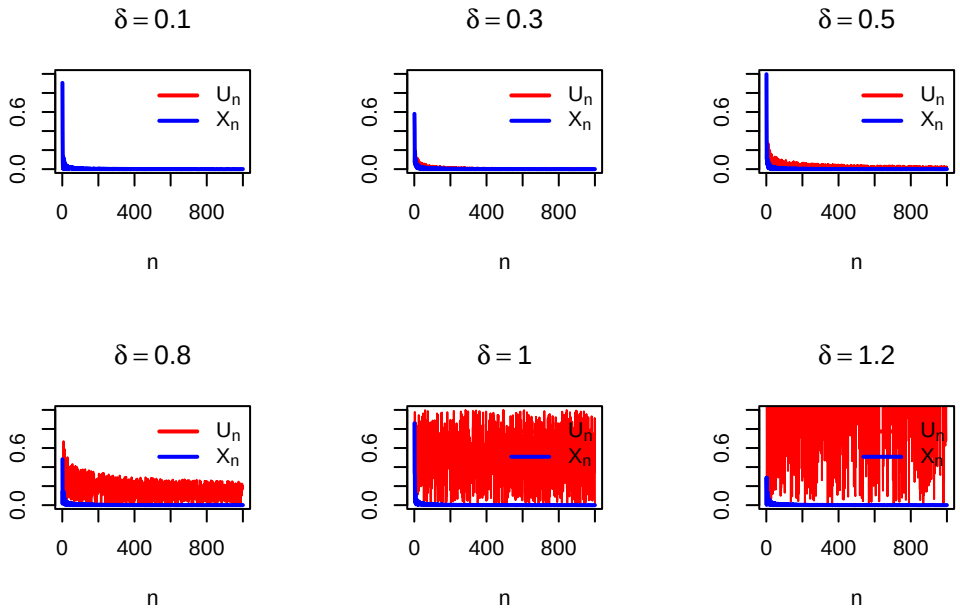


Figure 12: The convergence of X_n is shown for different choices of δ values. As δ increases (but less than 1), the rate of convergence to 0 is slower. At $\delta = 1$, it converges to Uniform$(0, 1)$ and for $\delta > 1$, the sequence diverges.

We have learnt two different convergence concepts: Convergence in probability and the convergence in distribution. In notation they are denoted as

$$X_n \xrightarrow{P} X, \qquad X_n \xrightarrow{d} X.$$

The following results hold:

- If $X_n \xrightarrow{P} X$, then $X_n \xrightarrow{d} X$. The converse of the statement is not true in general. However, if $X_n \xrightarrow{P} X$ and X is a degenerate random variable (that is a constant), then $X_n \xrightarrow{P} X$ as well.

- If $X_n \xrightarrow{P} X$ and $Y_n \xrightarrow{P} Y$, then $X_n + Y_n \xrightarrow{P} X + Y$.
- If $X_n \xrightarrow{P} X$ and a is a real number, then $aX_n \xrightarrow{P} aX$.
- If (a_n) be a sequence of real numbers with $a_n \to a$ as $n \to \infty$ and $X_n \xrightarrow{P} X$, then $a_n X_n \xrightarrow{P} aX$.

Insight into the Central Limit Theorem

Suppose that we have a random sample of size n from the population distribution whose PDF of PMF is given by $f(x)$. In addition, assume that the population has finite variance $\sigma^2 < \infty$. Let $\mu = \mathrm{E}(X)$ be the population mean. We denoted the sample mean using the symbol $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Due to the WLLN, we have observed that $\overline{X_n} \to \mu$ in probability as $n \to \infty$. It is to be noted that for every $n \geq 1$, $\overline{X_n}$ is a random variable and depending on the problem, its exact probability distribution may be computed. For example, if $X_1, X_2, \ldots, X_n$ be a random sample of size n from the normal distribution with mean μ and variance σ^2, then

$$\overline{X_n} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right), \text{ for all } n \geq 1.$$

One can show that the MGF of $\overline{X_n}$ is given by

$$M_{\overline{X_n}}(t) = e^{\mu t + \frac{1}{2}\frac{\sigma^2}{n}t}, -\infty < t < \infty.$$

which establishes the result by the uniqueness of the MGF.

Now can we generalize this result for any population distribution with finite variance? In particular, can for large sample size n, the sampling distribution of $\overline{X_n}$ be approximated by some normal distribution? If so, what will be the mean and variance of the approximating normal distribution?

Before going to provide an exact answer of the above questions, let us perform some computer simulations and see how the sampling distribution of the sample mean behaves for large sample size n and for different population distributions.

Experiment with exponential

In the following code, we repeatedly simulate a sample of size n from the exponential distribution with rate λ, which is given by

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x}, & 0 < x < \infty \\ 0, & \text{otherwise} \end{cases}.$$

The distribution is positively skewed and does not have any apparent connection with the normal distribution. We implement the following algorithm:

- Fix sample size n
- Fix the population parameter λ
- Fix the number of replication m
- Simulate $X_1, X_2, \ldots, X_n \overset{\text{iid}}{\sim} \text{Exponential}(\lambda)$
- Compute the Sample mean $\overline{X_n}$
- Repeat the previous two steps m times to obtain m realizations $\left\{\overline{X_n^{(j)}}\right\}_{j=1}^{m}$ from the distribution of $\overline{X_n}$.
- Plot the histogram of $\left\{\overline{X_n^{(j)}}\right\}_{j=1}^{m}$ values
- Overlay $\mathcal{N}\left(\frac{1}{\lambda}, \frac{1}{\lambda^2 n}\right)$ to the histogram.
- Repeat the above exercises for different (in increasing order) of sample size n.

```r
lambda = 2
n_vals = c(3,5,10,25,50,100)
rep = 1000
par(mfrow = c(2,3))
for(n in n_vals){
  sample_mean = numeric(length = rep)
  for(i in 1:rep){
    x = rexp(n = n, rate = lambda)
    sample_mean[i] = mean(x)
  }
  hist(sample_mean, probability = TRUE, breaks = 30,
       xlab = expression(bar(x)), main = paste("n = ", n))
  curve(dnorm(x, mean = 1/lambda, sd = sqrt(1/(lambda^2*n))),
        add = TRUE, col = "red", lwd =2)
  curve(dgamma(x, shape = n, rate = lambda*n),
        col = "blue", lwd = 3, lty = 2, add = TRUE)
}
```

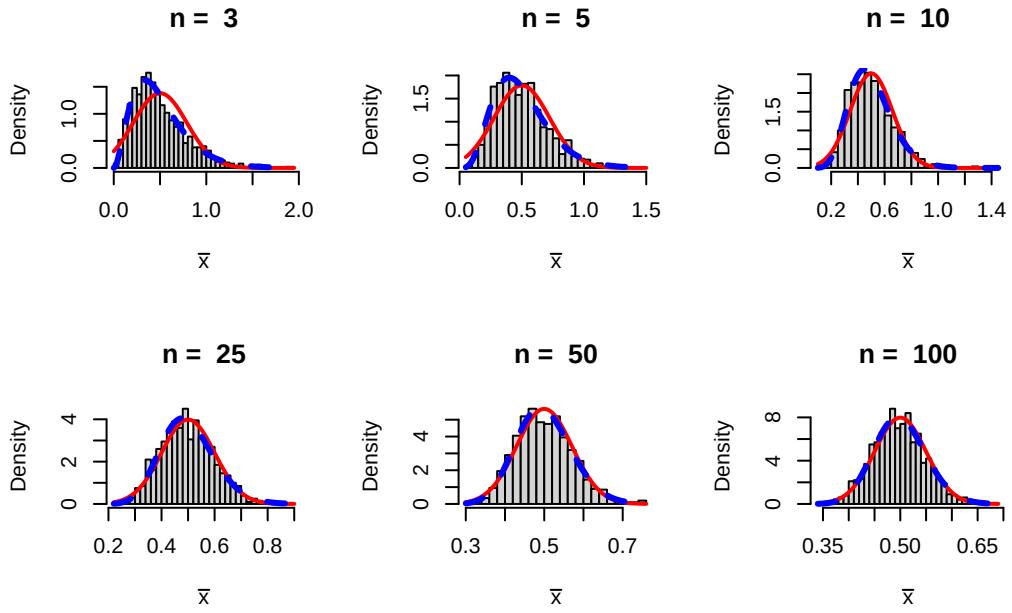


Figure 13: As the sample size increases, the sampling distribution of the sample mean can be well approximated by the normal distribution with mean $\frac{1}{\lambda}$ and variance $\frac{1}{\lambda^2 n}$. The histograms are obtained based on 1000 replications from the expoenential distribution with rate $\lambda = 2$.

In this simulation study we are lucky that the exact sampling distribution of the sample mean $\overline{X_n}$ can be derived using the MGF. The sampling distribution is given by

$$f_{\overline{X_n}}(x) = \begin{cases} \frac{(n\lambda)^n e^{-n\lambda x} x^{n-1}}{\Gamma(n)}, 0 < x < \infty \\ 0, \qquad\qquad \text{otherwise} \end{cases} .$$

Experiment with Poisson

We repeat the above simulation exercise for the Poisson distribution with rate λ. In the resulting histograms we overlay the normal distribution with mean λ and variance $\frac{\lambda}{n}$.

```
lambda = 2
par(mfrow = c(2,3))
n_vals = c(3, 5, 10, 25, 50, 100)
rep = 1000
for(n in n_vals){
  sample_mean = numeric(length = rep)
  for(i in 1:rep){
```

```
    x = rpois(n = n, lambda = lambda)
    sample_mean[i] = mean(x)
  }
  hist(sample_mean, probability = TRUE, breaks =30,
       xlab = expression(bar(x)), main = paste("n = ", n))
  curve(dnorm(x, mean = lambda, sd = sqrt(lambda/n)),
        add = TRUE, col = "red", lwd =2)
}
```

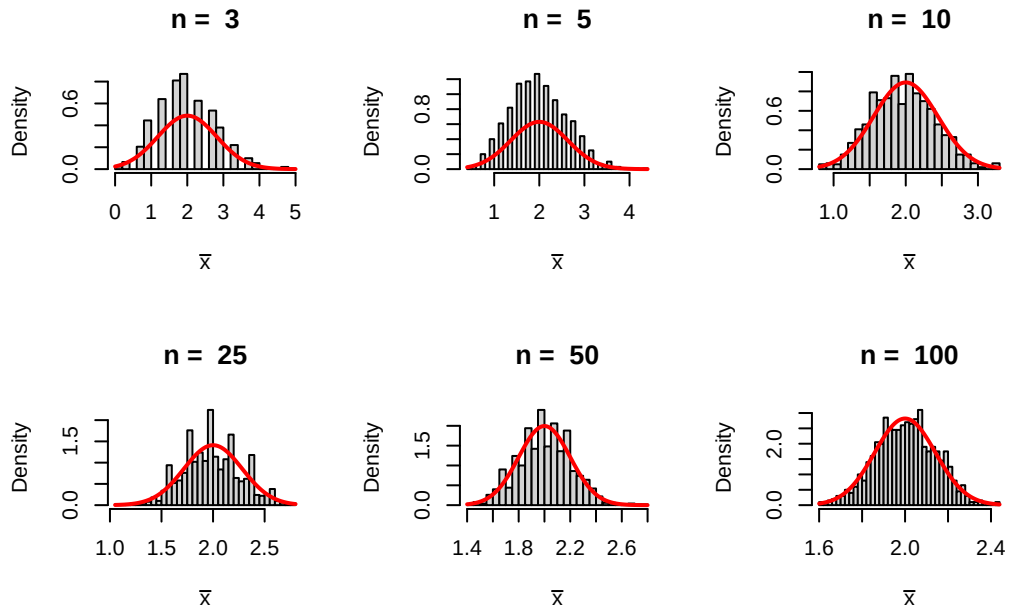


Figure 14: As the sample size increases, the sampling distribution of the sample mean can be well approximated by the normal distribution with mean λ and variance $\frac{\lambda}{n}$. The histograms are obtained based on 1000 replications from the Poisson distribution with rate $\lambda = 2$.

A natural question arises that whether for all probability distributions, the Central Limit Theorem holds. The answer is negative. Let us have a simulation experiment using the standard Cauchy distribution which is given by

$$f(x) = \frac{1}{\pi(1+x^2)}, -\infty < x < \infty.$$

Experiment with the Cauchy distribution

```r
par(mfrow = c(1,3))
curve(dcauchy(x), -5, 5, col = "red", lwd = 2)
curve(dnorm(x), add = TRUE, col = "blue", lwd = 2)

n_vals = 1:1000
sample_mean = numeric(length = length(n_vals))
for(n in n_vals){
  sample_mean[n] = mean(rcauchy(n = n))
}
plot(n_vals, sample_mean, col = "red", lwd = 2,
     type = "l", xlab = "sample size (n)")

n = 500
M = 1000
sample_mean = numeric(length = M)
for(i in 1:M){
  sample_mean[i] = mean(rcauchy(n = n))
}
hist(sample_mean, probability = TRUE,
     main = paste("n = ", n))
```

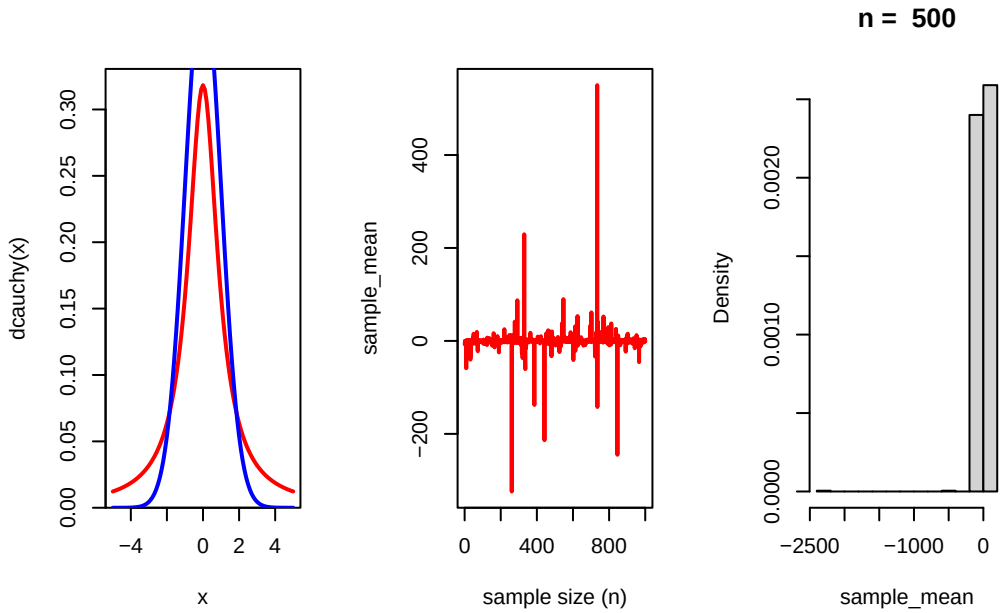


Figure 15: As sample size increases, the sample mean does not converge to μ, the location parameter. Also, for large enough n, the sampling distribution of the sample mean does not converge to a normal distribution. Therefore, for the Cauchy distribution, the WLLN and CLT both do not hold.

! WLLN and CLT: Points to Remember

Based on our discussion, we conclude the following:

- If the population has finite mean, then WLLN holds
- If the population has finite variance, then CLT holds.
- If the population has finite mean, but the second order moment does not exits, then WLLN will hold but CLT will not hold.
- If the population has finite variance, then both WLLN and CLT hold true.

A further generalization of the CLT led us investigate the sampling distribution of the function of sample mean, $g\left(\overline{X_n}\right)$ for some function $g(\cdot)$. For reasonable choice of the function, can we have some large sample approximation of the sampling distribution of $g\left(\overline{X_n}\right)$?

A Case study on Probabilistic Approximation

The Weak Law of Large Numbers is often used in the computation of numerical integration. The idea is to express the integral as the expected value of a function of a random variable.

This expected value is then approximated by averaging the appropriate transformation of the sample values. Let us consider a simple example of integrating the function $g(x) = e^{-x^2}$ in the interval $(0, 1)$. Let us denote the integral as

$$I = \int_0^1 e^{-x^2}.$$

Certainly the integral is very simple to compute numerically and we can get its value also using the following code:

```
g = function(x){
  exp(-x^2)
}
integrate(g, 0, 1)
```

```
0.7468241 with absolute error < 8.3e-15
```

Suppose that we take a strategy to write down this integral as an expected value of a function as follows:

$$I = \int_0^1 e^{-x^2} dx = \int_{-\infty}^{\infty} e^{-x^2} (f_m(x))^{-1} f_m(x) dx = \mathrm{E}_{X \sim f_m} \left(e^{-X^2} (f_m(X))^{-1} \right)$$

where $f_m(x)$ is a probability density function for each $m \in \{1, 2, ...\}$ defined by

$$f_m(x) = \begin{cases} mx^{m-1}, 0 < x < 1, \\ 0, \quad \text{otherwise.} \end{cases}$$

It is easy to follow that $f_1(x)$ is basically the uniform distribution over $(0, 1)$. Let us first consider the case $m = 1$. Therefore, we simulate random numbers $X_1, \dots, X_n \sim \text{Uniform}(0, 1)$ and compute the average of $e^{-X_1^2}, \dots, e^{-X_n^2}$, $\frac{1}{n} \sum_{i=1}^n e^{-X_i^2}$ which is an estimate of I, call it $\widehat{I_n}$. By WLLN $\widehat{I_n} \to I$, in probability.

```
n = 1000
x = runif(n)
I_n = mean(exp(-x^2))
print(I_n)                # Monte Carlo approximate of I
```

```
[1] 0.7557172
```

It is clear that $\widehat{I_n}$ is quite close to I and as n increases it becomes more close to I. Students are encouraged to write a code for a visual demonstration of this fact.

Now suppose that instead of uniform distribution, that is for $m = 1$, we would like to try for different values of m as well. We call

$$\psi_m(x) = \frac{1}{m} e^{-x^2} x^{1-m}, 0 < x < 1$$

and zero otherwise, and $m \in \{1, 2, ...\}$. We simulate $X_1, \dots, X_n \sim f_m(x)$ and compute

$$\widehat{I_n^{(m)}} = \frac{1}{n} \sum_{i=1}^{n} \psi_m(X_i).$$

In the following, we utilize the probability integral transform to simulate the observations from $f_m(x)$. The CDF of $f_m(x)$ is given by

$$F_m(x) = \begin{cases} 0, & x \leq 0, \\ x^m, & 0 < x \leq 1 \\ 1, & 1 < x < \infty \end{cases}$$

If $U \sim \text{Uniform}(0, 1)$, then $X = U^{1/m} \sim f_m(x)$. Students are encouraged to run the following codes and check on their own.

```
f_m = function(x){
  m*x^(m-1)*(0<x)*(x<1)
}
m = 2
n = 100
u = runif(n = n)
x = u^(1/m)
hist(x, probability = TRUE, main = paste("m = ", m))
curve(f_m(x), add = TRUE, col = "red", lwd = 2)
```

In the following we see how the sequence $I_n^{(m)}$ converges to I for different choices of m as $n \to \infty$.

```
par(mfrow = c(2,3))
f_m = function(x){                # the PDF
  m*x^(m-1)*(0<x)*(x<1)
}
psi_m = function(x){              # psi function g(x)/f_m(x)
  (1/m)*exp(-x^2)*x^(1-m)
}
```

```r
n_vals = 1:1000                 # n values
m_vals = c(1,2,3,4,5,6)         # different values of m
for(m in m_vals){
  I_n = numeric(length = length(n_vals))
  for(n in n_vals){
    u = runif(n)                # simulate from uniform(0,1)
    x = u^(1/m)                 # probability integral transform
    I_n[n] = mean(psi_m(x))
  }
  plot(n_vals, I_n, col = "darkgrey", lwd = 2, type = "l",
       xlab = "n", ylab = expression(I[n]^((m))),
       main = paste("m = ", m))
}
```

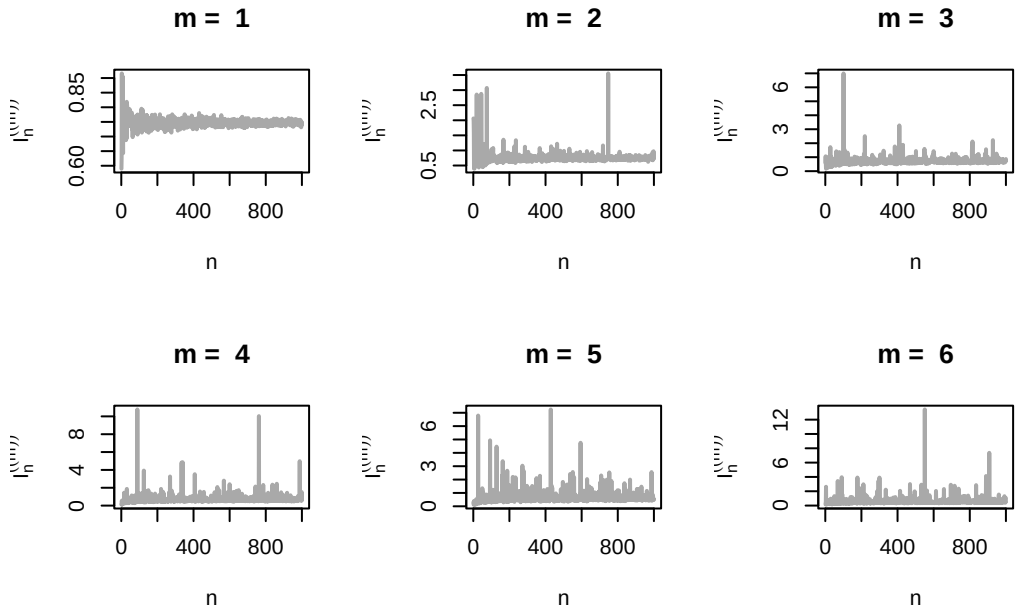


Figure 16: We check the convergence of the sequence $I_n^{(m)}$ for different choices of m as $n \to \infty$. It is evident that as m increases the rate of convergence becomes slower.

For different choices of m, we can estimate the standard error associated with the approximation. The estimated standard error associated with Monte Carlo approximation $\widehat{I_n}$ is given by $s/\sqrt{n}$, where $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\psi_m(X_i) - \widehat{I_n}\right)^2}$. Let us plot the standard error with respect to different choices of m.

```r
n = 10000
m_vals = 1:9
mat_psi_m = matrix(data = NA, nrow = n, ncol = length(m_vals))
I_n_SE = numeric(length = length(m_vals))
for(m in m_vals){
  set.seed(m+100)
  u = runif(n = n)
  x = u^(1/m)
  I_n = mean(psi_m(x))
  I_n_SE[m] = sd(psi_m(x))/sqrt(n)
  mat_psi_m[ ,m] = psi_m(x)
}
plot(m_vals, I_n_SE, type = "b", pch = 19,
     col = "red", cex = 1.3, xlab = "m", lwd = 2,
     main = expression(SE(I[n]^{(m)})), ylab = "Standard Error")
```

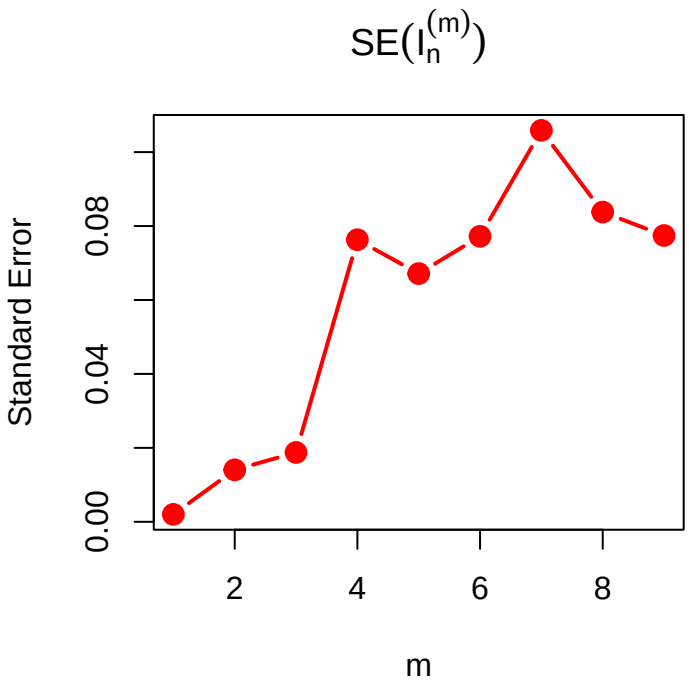


Figure 17: The estimated variance of the Monte Carlo approximation for different choices of m based on $n = 10000$ values are shown. It is clear that as m becomes large, the standard error associated with the approximation increases. In the code we have used the `set.seed()` function to generate reproducible figures.

```r
par(mfrow = c(2,3))
for(m in m_vals){
  hist(log(mat_psi_m[,m]), probability = TRUE,
       xlab = expression(I[n]^(m)), main = paste("m = ", m))
}
```

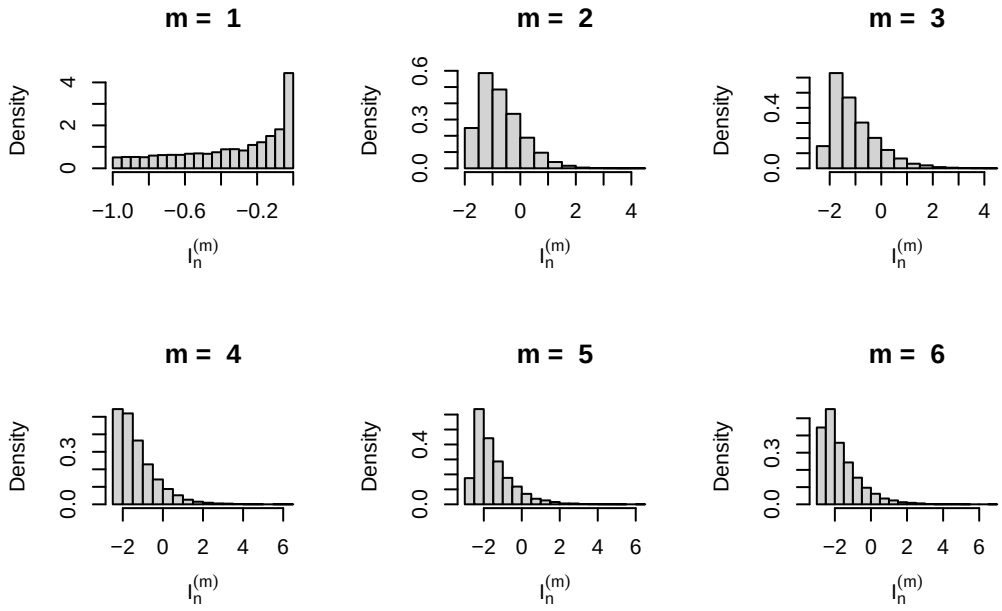


Figure 18: The sampling distribution of $I_n^{(m)}$ for different choices of m. The distributions are highly positively skewed for $m > 1$. In these figures, the histograms are drawn from the log transformed values of $\psi_m(X_i)$'s for $i \in \{1, 2, \ldots, n\}$

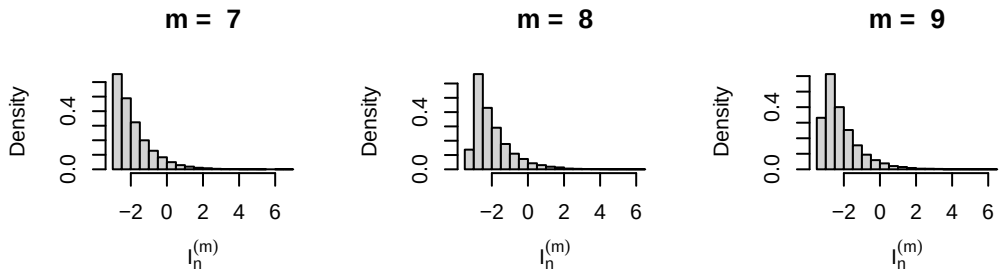


Figure 19: The sampling distribution of $I_n^{(m)}$ for different choices of m. The distributions are highly positively skewed for $m > 1$. In these figures, the histograms are drawn from the log transformed values of $\psi_m(X_i)$'s for $i \in \{1, 2, \ldots, n\}$

The problem can be generalized further using the fact that m not necessarily be a positive

integer. In fact for any $m > 0$, $f_m(x)$ is a probability density function. Let us relax this condition for m and consider an interval of choices for $m \in (0, 2)$ (say). Let us execute the above exercise for real values of m. The shape of the distribution of $\psi_m(X)$ is shown in Figure 23 (left panel). In the right panel of Figure 23, the estimated standard error of $I_n^{(m)}$ has been shown as a function of m. It is evident that there is a moderate value of m at which the approximation is the best.

```
par(mfrow = c(2,3))
m_vals = seq(0.1, 1.8, by = 0.1)
n_vals = 1:1000
for(i in 1:length(m_vals)){
  m = m_vals[i]                                  # value of m
  I_n = numeric(length = length(n_vals))         # value of psi_m(x_i)
  for(n in n_vals){
    u = runif(n = n)
    x = u^(1/m)
    I_n[n] = mean(psi_m(x))
  }
  plot(n_vals, I_n, main = paste("m = ", m),
       col = "darkgrey", lwd = 2, xlab = "n",
       ylab = expression(I[n]), type = "l")

}
```

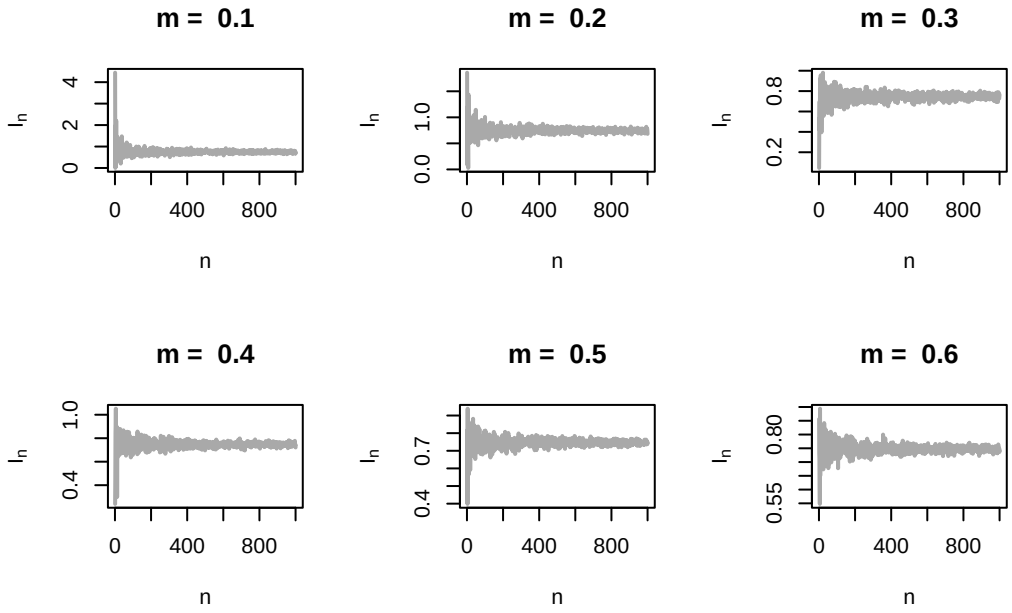


Figure 20: The convergence of the sequence $I_n^{(m)}$ for different choices of m and it should be noted that m is not an integer. The fluctuations in the values clearly indicate the different rate of convergence as $n \to \infty$.

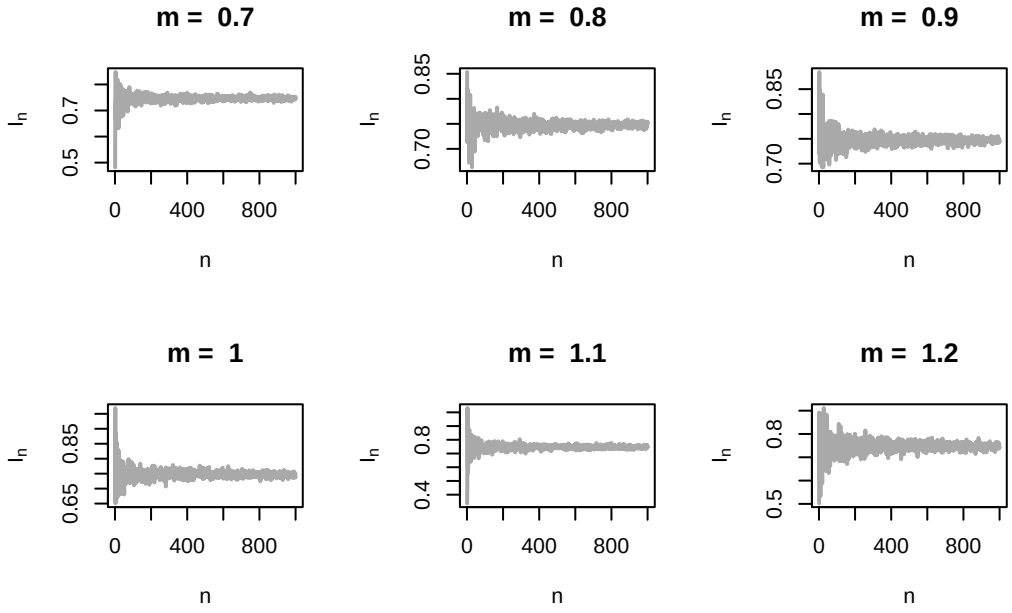


Figure 21: The convergence of the sequence $I_n^{(m)}$ for different choices of m and it should be noted that m is not an integer. The fluctuations in the values clearly indicate the different rate of convergence as $n \to \infty$.

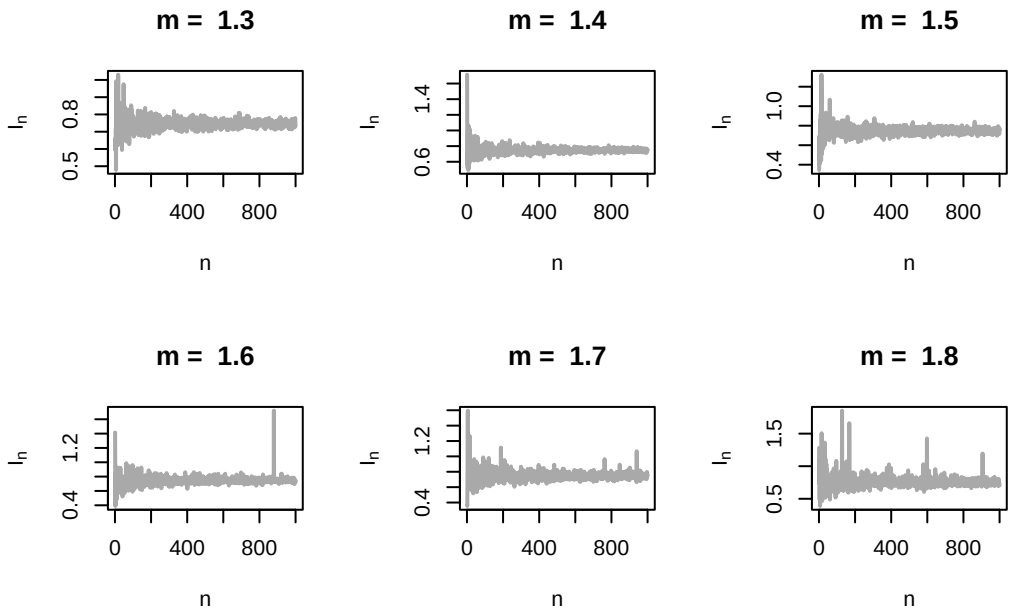


Figure 22: The convergence of the sequence $I_n^{(m)}$ for different choices of m and it should be noted that m is not an integer. The fluctuations in the values clearly indicate the different rate of convergence as $n \to \infty$.

In the following code, let us estimate the standard error associated with the approximation.

```
par(mfrow = c(1,2))
m_vals = seq(0.1, 1.5, by = 0.08)
n = 1000
mat_psi_m = matrix(data = NA, nrow = n, ncol = length(m_vals))
for(j in 1:length(m_vals)){
  m = m_vals[j]
  u = runif(n = n)
  x = u^(1/m)
  mat_psi_m[, j] = psi_m(x)
}
boxplot(mat_psi_m, names = m_vals)
plot(m_vals, apply(mat_psi_m, 2, sd)/sqrt(n), type = "b",
     col = "red", lwd = 2, pch = 19, xlab = "m",
     ylab = "Standard Error", main = expression(SE(I[n]^(m))))
```

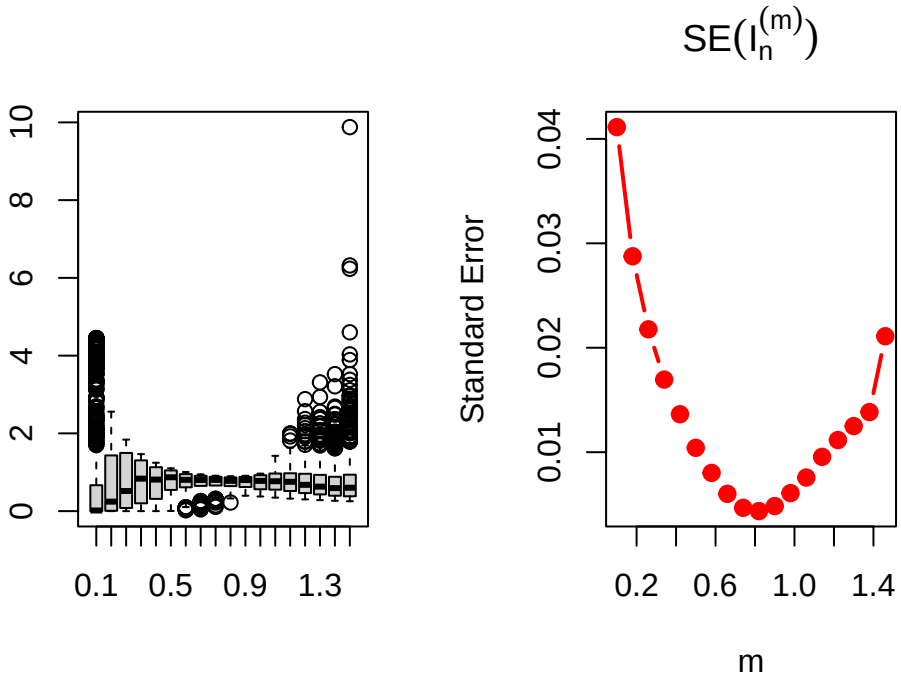


Figure 23: The standard error associated with the approximation I_n is shown in the right panel. It is clear that there is some intermediate value around 1 which gives the best approximation. Fluctuations around the true value is high for both lower (close to zero) and large value ($>> 1$) of m. There is a an intermediate state, where the flucuation is the least.

The Delta method

Suppose that we have a random sample of size n from the Poisson distribution with rate parameter λ. We are interested in estimating the probability $\mathrm{P}(X \geq 1)$. In other words, suppose that the number of telephone calls received by a customer in a day is assumed to follow the Poisson distribution with mean λ and we are interested to approximate the probability that the customer will receive at least one call. We know that the MLE of λ is $\overline{X_n}$, the sample mean. In addition, we have observed some of the important properties of $\overline{X_n}$.

- $\overline{X_n}$ converges to the population mean λ as $n \to \infty$ in probability. That is for any $\epsilon > 0$, $\lim_{n\to\infty} \mathrm{P}\left(\left|\overline{X_n} - \lambda\right| > \epsilon\right) = 0$. In other words, we also say that $\overline{X_n}$ is a consistent estimator of λ, population mean. This is also followed from the Weak Law of Large Numbers (WLLN) which states that the sample mean converges to the population mean in probability.

- Using the Central Limit Theorem, we have also learnt that

$$\overline{X_n} \sim \mathcal{N}\left(\lambda, \frac{\lambda}{n}\right), \text{ for large } n,$$

  which states that the sampling distribution of the sample mean can be well approximated by the normal distribution for large sample sizes, when the samples are drawn from a population distribution with finite variance.

In the current problem, the problem is to estimate a function of the parameter λ which is

$$\psi(\lambda) = \mathrm{P}(X \geq 1) = 1 - (1+\lambda)e^{-\lambda}.$$

A natural choice is to approximate the parameter $\psi(\lambda)$ by using

$$\psi\left(\overline{X_n}\right) = 1 - \left(1 + \overline{X_n}\right) e^{-\overline{X_n}}.$$

Now our task is to study the behavior of $\psi\left(\overline{X_n}\right)$, like unbiasedness, consistency and check the asymptotic distribution for large n. Even if we are not able to obtain the exact sampling distribution, but it will be important to check whether we can have some large sample approximations by some known distributions.

By using Taylor's approximation (first order) about λ and neglecting higher order terms, we can see that

$$\psi\left(\overline{X_n}\right) \approx \psi(\lambda) + \left(\overline{X_n} - \lambda\right)\psi'(\lambda).$$

Taking expectations on both sides, we see that

$$\mathrm{E}\left[\psi\left(\overline{X_n}\right)\right] \approx \psi(\lambda).$$

This means that $\psi\left(\overline{X_n}\right)$ is an approximately unbiased estimator of $\psi(\lambda)$ at least for large n.

From the above computation, after some rearranging

$$\mathrm{E}\left(\psi\left(\overline{X_n}\right) - \psi(\lambda)\right)^2 \approx \left(\psi'(\lambda)\right)^2 \mathrm{E}\left(\overline{X_n} - \lambda\right)^2 = \left(\psi'(\lambda)\right)^2 Var\left(\overline{X_n}\right) = \frac{\lambda^3 e^{-2\lambda}}{n}.$$

Therefore, approximate variance of $\psi\left(\overline{X_n}\right)$ is $\frac{\lambda^3 e^{-2\lambda}}{n}$. In the following simulation, we visualize the sampling distribution of $\psi\left(\overline{X_n}\right)$.

```r
par(mfrow = c(2,3))
n_vals = c(4, 10, 20, 50, 100, 500)
rep = 1000
lambda = 2
psi = 1-(1+lambda)*exp(-lambda)
for(n in n_vals){
  psi_vals = numeric(length = rep)
```

```r
  for(i in 1:rep){
    x = rpois(n = n, lambda = lambda)
    psi_vals[i] = 1-(1+mean(x))*exp(-mean(x))
  }
  hist(psi_vals, probability = TRUE,
       xlab = expression(psi), main = paste("n = ", n), breaks = 30)
  points(psi, 0, pch = 19, col = "red", cex = 1.3)
  curve(dnorm(x, mean = psi, sd = sqrt(lambda^3*exp(-2*lambda)/n)), add = TRUE,
        col = "red", lwd =2)
}
```

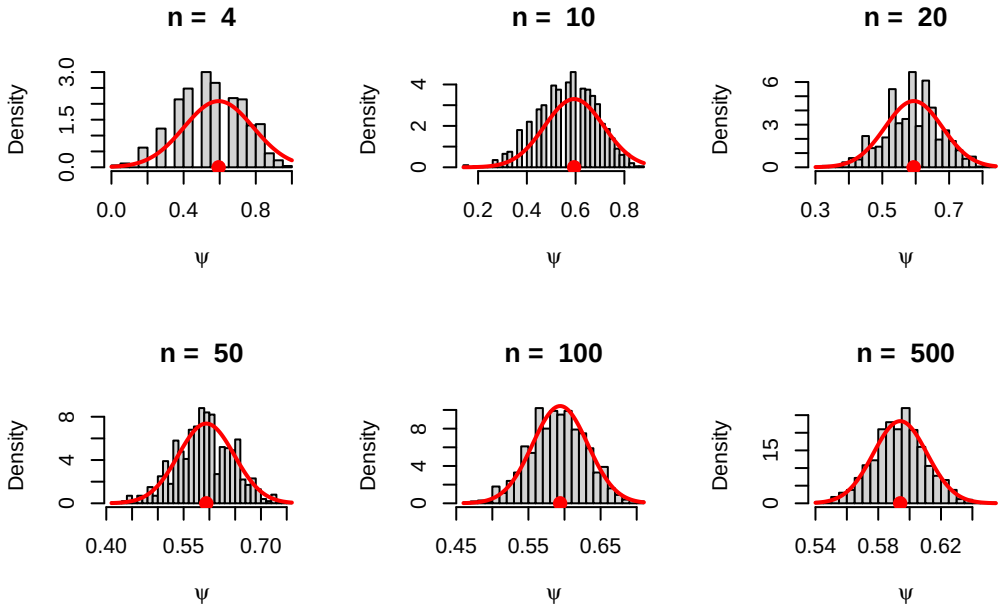


Figure 24: The sampling distribution of the function of the sample mean $\psi\left(\overline{X_n}\right)$ is visualized for different sample sizes by histograms based of 1000 replications.

Exercises

- For random variables $X_1, X_2, \ldots, X_n$, show that

$$\mathrm{E}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \mathrm{E}(X_i)$$

and

$$Var\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} Var(X_i) + 2\sum\sum_{i<j} Cov(X_i, X_j).$$

- Let $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_m$ be two sets of random variables and let $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$ be two sets of constants; then prove that

$$Cov\left(\sum_{i=1}^{n} a_i X_i, \sum_{j=1}^{m} b_j X_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_i b_j Cov\left(X_i, X_j\right)$$

- Suppose that $X_1, X_2, \ldots, X_n$ be a random sample of size n from a population with PDF $f(x)$ and CDF $F(x)$. Define $Y_1 = \min(X_1, \ldots, X_n)$ and $Y_n = \max(X_1, X_2, \ldots, X_n)$ be the minimum and the maximum order statistics. Obtain the sampling distribution of Y_1 and Y_n.

- Suppose that $X_1, X_2, \ldots, X_n \sim$ Exponential(β) with mean $\beta \in (0, \infty)$. Obtain the exact sampling distribution of Y_1 and Y_n. Fix β and the sample size n of your own choice. Simulate a random sample of size n from the population and obtain the approximate sampling distribution of Y_1 and Y_n using histograms based on $m = 1000$ replications. Check whether your theoretical derivation matches with the simulation.

- Suppose that $X_1, X_2, \ldots, X_n$ be random sample of size n from the exponential distribution with mean β. Using MGF, obtain the exact sampling distribution of the sample mean $\overline{X_n}$ and verify your theoretical result using computer simulation.

- Suppose that $X_1, X_2, \ldots, X_n$ be a random sample of size n from the population distribution whose PDF is given by $f(x|\theta) = \frac{1}{\theta}, 0 < x < \theta$ and zero otherwise. Suppose that we are interested to estimate the parameter θ using the sample mean $\overline{X_n}$. Check whether $\overline{X_n}$ is an unbiased estimator of θ. If not can you compute the bias? Also compute the variance of $\overline{X_n}$. Compute the average squared distance of the estimator from the true value, that is $\mathrm{E}\left(\overline{X_n} - \theta\right)^2$.

- Suppose that in the above problem, instead of estimating θ using $\overline{X_n}$, we plan to estimate using the largest order statistics, that is, $Y_n = \max(X_1, X_2, \ldots, X_n)$. Is it an unbiased estimator of θ? Compute the variance of Y_n and also compute $\mathrm{E}(Y_n - \theta)^2$.

- Suppose that $X_1, X_2, \ldots, X_n$ be a random sample of size n from the geometric distribution with parameter p, $(0 < p < 1)$. The population random variable X represents the number of throws required to get the first success. Starting with uniform random numbers from the interval $(0, 1)$, simulate $n = 30$ realizations from the distribution of X. Suppose that we are interested to estimate $1/p$ using $\overline{X_n}$, the sample mean. Based on $M = 10^6$ replications obtain the sampling distribution of $\overline{X_n}$ and draw the histogram. Do this exercise for $n \in \{3, 5, 10, 25, 50, 100\}$. Do the histograms shrink as the sample size increases? On each histograms check whether $\frac{1}{M}\sum_{i=1}^{M} \overline{X_n}^{(i)}$ is approximately equal to the $\frac{1}{p}$. What is your conclusion about the unbiasedness of the estimator?

Loss function and the Risk function

Introduction

At the beginning of my Statistical Computing lectures, I often share a thought with my students: In statistics, our ultimate goal is to uncover the truth. However, this truth is beyond the reach of humanity—not only now but even in the distant future. When we collect data from natural processes, we aim to understand the workings of nature through mathematical models. Yet, all our efforts based on the collected data merely approximate the true functioning of nature, and achieving 100% accuracy is inherently impossible.

This naturally leads to an important question: If the absolute truth is unattainable, how can we validate statistical approximations and assess their accuracy? This question lies at the very core of understanding the principles and philosophy of statistical science.

However, we need to devise strategies to evaluate how accurate statistical approximations are in estimating the unknown parameters. Suppose that we have a random sample of size n from the population density function $f(x;\theta)$ for $\theta \in \Theta$. We can approximate θ based on some function $\psi(X_1, \ldots, X_n)$, based on a sample of size n. We need to measure the closeness of the estimator ψ_n to θ.

Simulation experiment with Bernoulli distribution

In this section, we approximate the risk function of the sample proportion as an estimator of the true proportion when sampling from the Bernoulli(p) populations. We perform this task in three steps.

- `Step - I`
  - Fix the parameter p
  - Fix the sample size n
  - Simulation $X_1, X_2, \ldots, X_n \sim$ binomial$(1, p)$.
  - Compute the loss $(\widehat{p_n} - p)^2$

Repeat the following codes multiple times to realize that the loss function $l(\widehat{p_n}, p)$ is a random variable.

```r
p = 0.1                                          # true probability
n = 10                                           # sample size
x = rbinom(n = n, size=1, prob = p)              # simulation
head(x)
```

```
[1] 0 0 1 0 1 0
```

```r
pn_hat = mean(x)
head(pn_hat)
```

```
[1] 0.2
```

```r
(pn_hat - p)^2                                   # computing squared loss
```

```
[1] 0.01
```

- Step - II

To compute the average loss, we need to repeat the above process M times (say) to get an estimate of the risk at a specific value of p.

```r
M = 1000                                         # number of replication
pn_loss = numeric(length = M)
for(m in 1:M){
  x = rbinom(n = n, size=1, prob = p)
  pn_hat = mean(x)
  pn_loss[m] = (pn_hat - p)^2
}
head(pn_loss)
```

```
[1] 0.00 0.01 0.04 0.01 0.01 0.00
```

```r
pn_risk = mean(pn_loss)                          # compute risk
head(pn_risk)
```

```
[1] 0.00806
```

`Step - I` and `Step - II` have been carried out for a fixed value of p. If we want to obtain the performance of the estimator irrespective of the true value, we must evaluate its performance for each $p \in (0,1)$, which will give us the risk profile of the estimator. This also called a risk function, which is a function of the parameter p and contain no randomness.

$$R(\widehat{p_n}, p) = \mathrm{E}\left(l(\widehat{p_n}, p)\right).$$

We can plot this function as a function of p.

- `Step - III`
  - Discretize the parameter space $(0,1)$ into distinct points $p_1 < p_2 < ... < p_L$.
  - For each $p_j, 1 \leq j \leq L$, perform step - I and step - II.

```
prop_vals = seq(0.01, 0.99, by = 0.01)
pn_risk = numeric(length = length(prop_vals))
M = 1000                                        # number of replications
for(i in 1:length(prop_vals)){
  p = prop_vals[i]
  pn_loss = numeric(length = M)
  for(m in 1:M){
    x = rbinom(n = n, size=1, prob = p)
    pn_hat = mean(x)
    pn_loss[m] = (pn_hat - p)^2
  }
  pn_risk[i] = mean(pn_loss)

}
head(pn_risk)
```

```
[1] 0.001232 0.001758 0.002740 0.003590 0.004500 0.005570
```

```
plot(prop_vals, pn_risk, col = "red",
     pch = 19, xlab = "p", ylab = expression(R(hat(p[n]),p)),
     main = paste("n = ", n))
curve(x*(1-x)/(n), add = TRUE, col = "blue", lty = 2,
      lwd = 3)
```

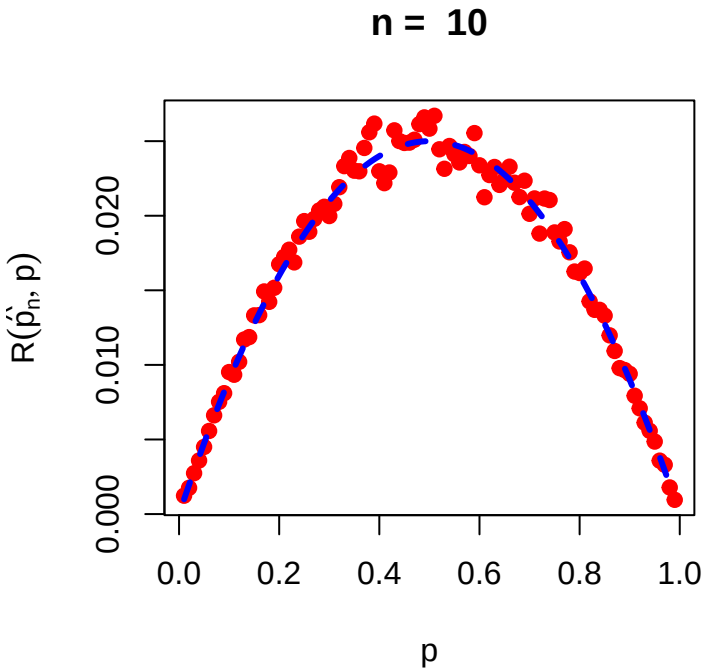


Figure 1: The simulated risk function of the MLE of the probability of head p under the squared error loss. The risk, that is, expected loss has been approximated based on the 1000 replications.

Basically, the risk function says that on an average what will be the mistake in squared error scale, if we replace the true value of p with the sample proportion.

- `Step - IV`

A natural question arises, what will happen if we change the sample size n. Intuition suggests that as n increases, the distance between the truth and estimate would be small, that $\mathcal{R}(\widehat{p_n}, p)$ would be a decreasing function of n at each $p \in (0, 1)$. In the following code, we see the behavior of the risk function for different choices of p.

```
n_vals = c(5, 10, 20, 30, 50, 100)
M = 1000                                        # number of replications
par(mfrow = c(2,3))
for(n in n_vals){
  prop_vals = seq(0.01, 0.99, by = 0.01)
  pn_risk = numeric(length = length(prop_vals))

  for(i in 1:length(prop_vals)){
    p = prop_vals[i]
```

```r
    pn_loss = numeric(length = M)
    for(m in 1:M){
      x = rbinom(n = n, size=1, prob = p)
      pn_hat = mean(x)
      pn_loss[m] = (pn_hat - p)^2
    }
    pn_risk[i] = mean(pn_loss)

  }
  plot(prop_vals, pn_risk, col = "red",
       pch = 19, xlab = "p", ylab = expression(R(hat(p[n]),p)),
       main = paste("n = ", n), ylim = c(0,0.055))
}
```

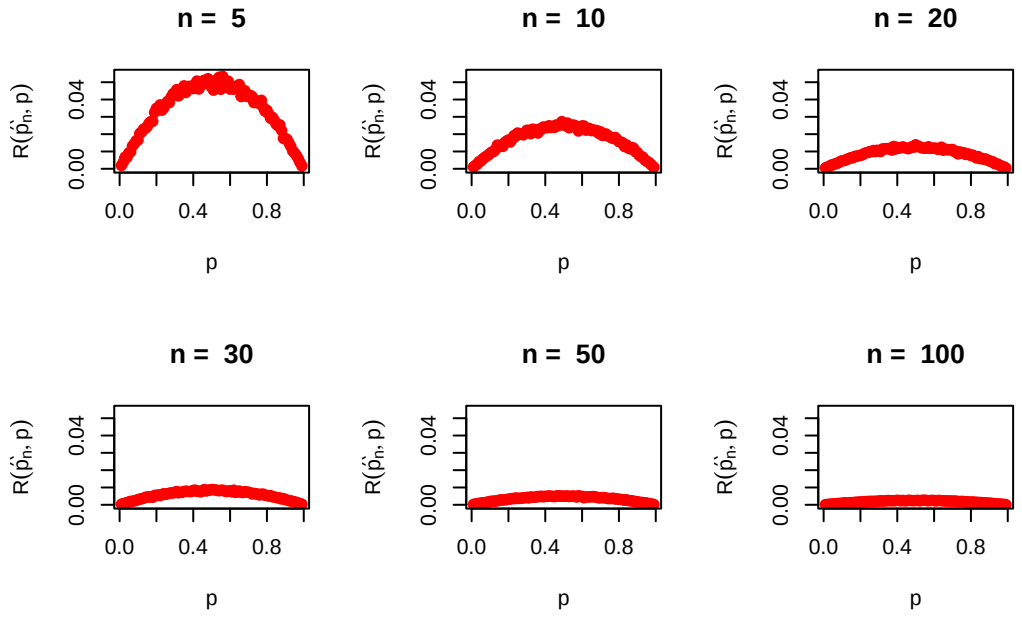


Figure 2: The risk function is obtained by using computer simulation based on 1000 replications for different sample sizes n. It is clear that as $n \to \infty$, te risk goes to zero.

It would be interesting to see whether $\mathcal{R}\left(\widehat{p_n}, p\right) \to 0$ as $n \to \infty$ for all $p \in (0,1)$. Theoretical computation can help us in establishing/disproving this claim. The simulated shapes actually supports the theoretical computations as well. The theoretical computation

$$\mathrm{E}(\widehat{p_n}) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}(X_i) = \frac{1}{n}np = p.$$

Therefore,

$$\mathrm{E}\left(\overline{X_n} - p\right)^2 = Var_p\left(\overline{X_n}\right) = \frac{1}{n^2} n \times Var(X_1) = \frac{p(1-p)}{n}.$$

In the above simulated plots of the risk profiles for different choices of n, you can add the curve $p(1-p)/n$ and see that the curve is in well agreement is with the simulation.

Simulation experiment with the normal distribution

Suppose that we have a random sample of size n from the $\mathcal{N}(\mu, 1)$ population distribution. We aim to estimate the parameter μ using two different estimators $T_1 = \overline{X_n}$ and $T_2 = \text{Median}(X_1, \ldots, X_n)$.

We are aiming to compare the performance of these two estimators at different values of the unknown parameter μ. Therefore, we compute $\mathrm{E}\left(T_1 - \mu\right)^2 = R(T_1, \mu)$ and $\mathrm{E}\left(T_2 - \mu\right)^2 = R(T_2, \mu)$.

Consider an interval of μ values of your own choice and compute the risk functions using computer simulations and plot them in a single plot window. What is conclusion about the choice of the estimator?

Sample Mean or Sample Median

```
n = 10
sigma = 1          # fixed
a = -1
b = 1
M = 1000           # number of replications

mu = a
x = rnorm(n = n, mean = mu, sd = sigma)
mean(x)
```

```
[1] -1.020313
```

```
median(x)
```

```
[1] -0.9375056
```

```r
sample_mean = numeric(length = M)
sample_median = numeric(length = M)

for(m in 1:M){
  x = rnorm(n = n, mean = mu, sd = sigma)
  sample_mean[m] = mean(x)
  sample_median[m] = median(x)
}

mean((sample_mean - mu)^2)
```

```
[1] 0.09855808
```

```r
mean((sample_median - mu)^2)
```

```
[1] 0.130316
```

```r
par(mfrow = c(1,2))
hist(sample_mean, probability = TRUE, main = paste("n = ", n),
     breaks = 30, xlab = expression(bar(X[n])))
hist(sample_median, probability = TRUE, main = paste("n = ", n),
     breaks = 30, xlab = expression(Med(X[n])))
```

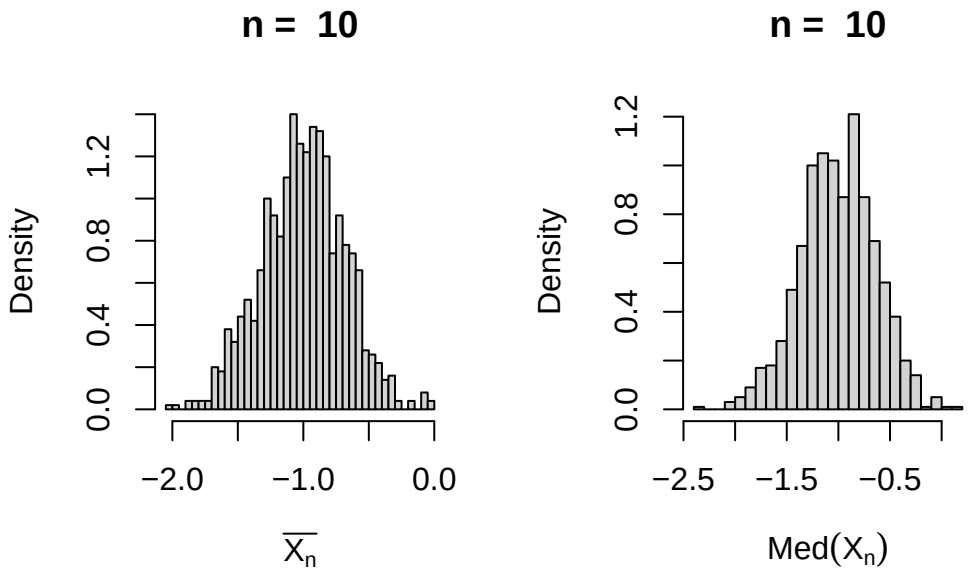


Figure 3: The sampling distribution of the sample mean and the sample median have been simulated for sample size $n = 10$ based on 1000 replications. We consider the normal distribution with mean -1 and variance 1 for demonstration.

As the value of μ changes, we need to check the performance of the sample median and the sample mean.

```
mu_vals = seq(a, b, by = 0.01)
risk_mean = numeric(length = length(mu_vals))
risk_median = numeric(length = length(mu_vals))
for (i in 1:length(mu_vals)) {

  sample_mean = numeric(length = M)
  sample_median = numeric(length = M)

  for(m in 1:M){
    x = rnorm(n = n, mean = mu, sd = sigma)
    sample_mean[m] = mean(x)
    sample_median[m] = median(x)
  }

  risk_mean[i] = mean((sample_mean - mu)^2)
  risk_median[i] = mean((sample_median - mu)^2)

}
plot(mu_vals, risk_median, type = "p", col = "red", lwd = 2,
```

```
     ylim = c(0, .16), ylab = expression(R[T](mu)),
     xlab = expression(mu))
points(mu_vals, risk_mean, type = "p", col = "blue", lwd = 2)
legend("bottomleft", legend = c(expression(T[1]), expression(T[2])),
       col = c("blue", "red"), bty = "n", lwd = c(2,2))
```

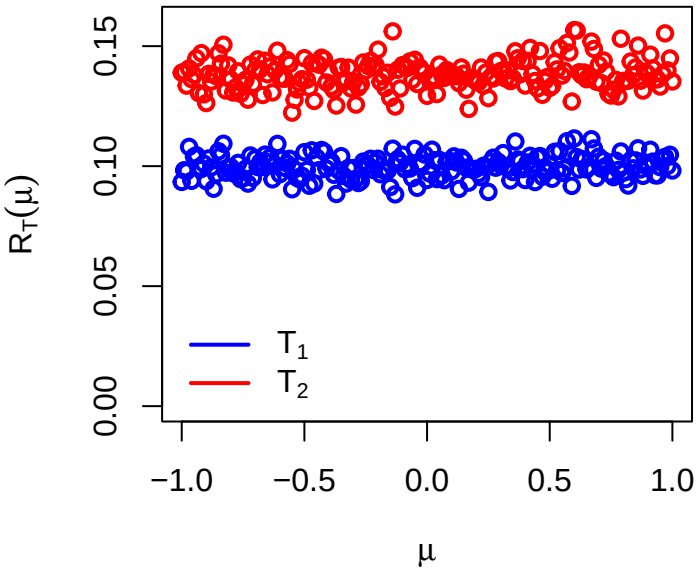


Figure 4: The risk values will be constant when the number of replications is large, that is, $M \to \infty$. From the simulation it is clear that sample mean has uniformly lesser risk than the sample median when used as an etimator for estimating the mean of normally distributed population. Students are encouraged to experiment with this simulation for different values of μ and sample size n.

A desirable property of the sample mean and sample median would be to become close to the true value of μ as $n \to \infty$. Using the following visualization, we can check whether these estimators are consistent estimators of the population mean. The following simulation suggests that indeed both of them are consistent, however, fluctuations about the true value is more for the sample median as compared to the sample mean. The following algorithm is implemented to check the consistency.

- Fix $\mu = \mu_0$
- Fix σ^2
- Consider $n \in \{1, 2, \dots, n_{\max}\}$ (sample size)
- For each n,
  - simulate $X_1, X_2, \dots, X_n \sim \mathcal{N}(\mu_0, \sigma^2)$

  - Compute Sample Mean $\overline{X_n}$
  - Compute Sample Median $\text{Med}(X_n)$
- Plot $\left(n, \overline{X_n}\right)$ and $(n, \text{Med}(X_n))$ for $n \in \{1, 2, \ldots, n_{\max}\}$.

```r
n_vals = 1:1000
mu = 0
sigma = 1
sample_median = numeric(length = length(n_vals))
sample_mean = numeric(length = length(n_vals))
for(n in n_vals){
  x = rnorm(n = n, mean = mu, sd = sigma)
  sample_median[n] = median(x)
  sample_mean[n] = mean(x)
}
par(mfrow = c(1,2))
plot(n_vals, sample_median, type = "l", col = "grey", lwd =2,
     xlab = "sample size (n)", main = expression(Med(X[n])),
     ylab = "")
abline(h = mu, col = "blue", lwd = 3, lty = 2)

plot(n_vals, sample_mean, type = "l", col = "grey", lwd =2,
     xlab = "sample size (n)", main = expression(bar(X[n])),
     ylab = "")
abline(h = mu, col = "blue", lwd = 3, lty = 2)
```

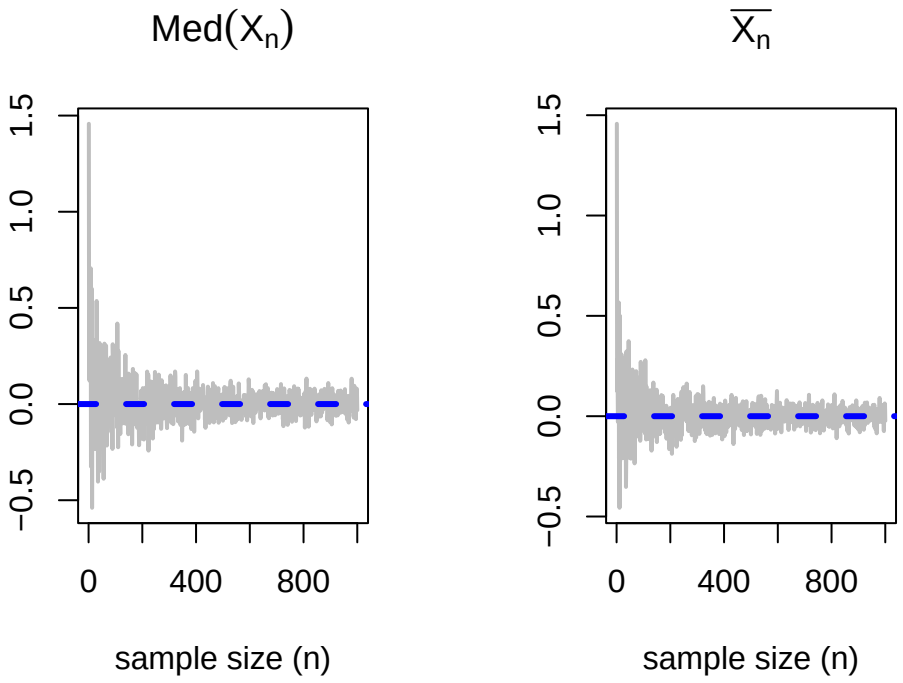


Figure 5: As the sample size increases, the sample mean and sample median converge to the true value μ as $n \to \infty$.

Asymptotic Comparison of Sample Mean and Sample Median

If $(X_1, X_2, \ldots, X_n)$ be a random sample of size n from the $\mathcal{N}(\mu, \sigma^2)$. Then it can be shown that

$$\sqrt{n}\left(\overline{X_n} - \mu\right) \xrightarrow{D} \mathcal{N}\left(0, \sigma^2\right)$$

and

$$\sqrt{n}\left(\text{Med}(X_n) - \mu\right) \xrightarrow{D} \mathcal{N}\left(0, \sigma^2 \frac{\pi}{2}\right).$$

This result also supports the simulation experiment which showed that both sample mean and sample median converged to the true value of μ, however, sample median has a larger variance as compared to the sample mean.

Sample Mean or Sample Variance (Poisson distribution)

To estimate the parameters of the population, the method of moments is a way to obtain estimator(s). In this method, population moments are made equal to the sample moments and the population parameters are expressed as the function of samples.

If the population distribution follows the Poisson distribution with mean λ. Then to obtain the MoM estimator of λ, we equate the population mean and the sample sample mean $\overline{X_n}$, which gives the following equation

$$\overline{X_n} = \mathrm{E}(X) = \lambda.$$

Therefore, the MoM moment estimator of λ is $\overline{X_n}$. However, one can also observe that the population variance is also λ therefore, equating the sample variance S_n^2 with the population variance (λ), we find that the sample variance is also an MoM estimator of λ. Therefore, the first conclusion is that the Method of Moment estimator is not unique.

A natural question arises, which estimator to prefer in estimating the unknown parameter λ! We have the following observations,

$$\mathrm{E}\left(\overline{X_n}\right) = \lambda = \mathrm{E}\left(S_n^2\right).$$

Therefore, both are unbiased estimators of λ. We now aproximate the risk functions corresponding to $\overline{X_n}$ and S_n^2, which are denoted by $\mathcal{R}\left(\overline{X_n}, \lambda\right)$ and $\mathcal{R}\left(S_n^2, \lambda\right)$, respectively, for $\lambda \in (0, \infty)$.

In the following

```r
n = 5                                          # sample size
M = 1000                                       # number of replications
lambda = 1
sample_mean = numeric(length = M)
sample_var = numeric(length = M)
for(i in 1:M){
  x = rpois(n = n, lambda = lambda)     # simulate from Poisson(1)
  sample_mean[i] = mean(x)
  sample_var[i] = var(x)
}
par(mfrow = c(1,2))
hist(sample_mean, probability = TRUE,
     xlab = expression(bar(X[n])), main = paste("n = ",n))
points(lambda, 0, pch = 19, col = "red", cex = 1.5)
hist(sample_var, probability = TRUE,
     main = paste("n = ", n), xlab = expression(S[n]^2))
points(lambda, 0, pch = 19, col = "red", cex= 1.5)

# Computing the risk values
risk_sample_mean = mean((sample_mean - lambda)^2)
cat("The estimated risk the sample mean under squared error loss function is \n",risk_sample_
```

```
The estimated risk the sample mean under squared error loss function is
 0.22404
```

```r
risk_sample_var = mean((sample_var - lambda)^2)
cat("The estimated risk of the sample variance under squared error loss function is \n",risk_
```

```
The estimated risk of the sample variance under squared error loss function is
 0.64821
```

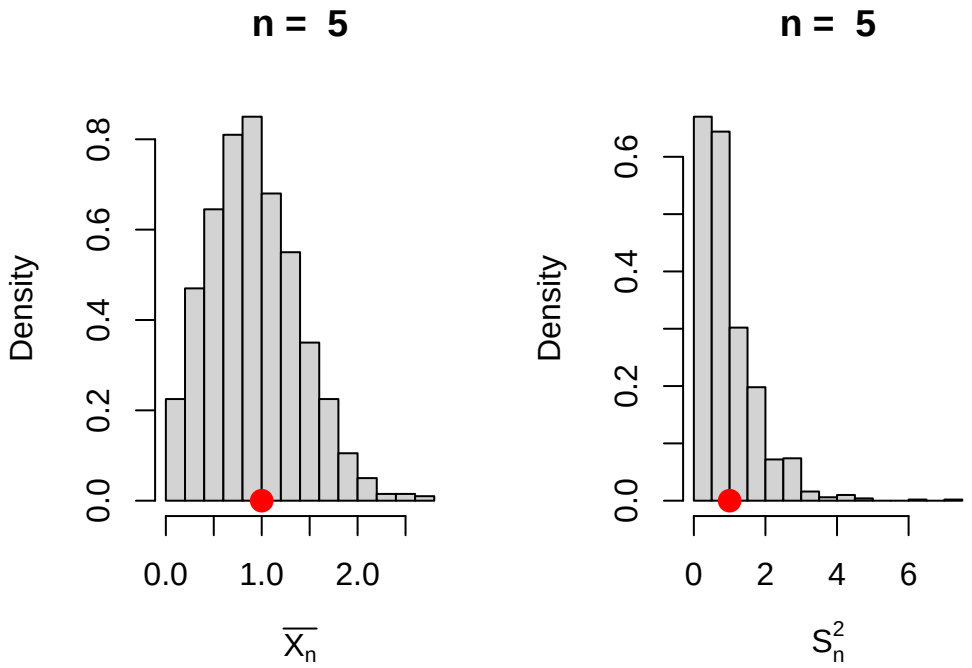


Figure 6: The simulated histograms are obtained based on 1000 replications. The population parameter λ is fixed at 1 and the sample size $n = 5$ is fixed for simulation. The histograms clearly suggests that the spread is more for the sample variance as compared to the sample mean about the true value of λ, which is denoted by the red dot.

Since, we do not know what is the true value of λ, in the following code, we performed the same task at different choices of λ. For simulation, purpose, we discretize the $(0.1, 3)$ interval for possible choices of λ.

```r
n = 5
M = 1000
par(mfrow = c(1,1))
```

```r
lambda_vals = seq(0.1, 2, by = 0.03)
risk_sample_mean = numeric(length = length(lambda_vals))
risk_sample_var = numeric(length = length(lambda_vals))
for(j in 1:length(lambda_vals)){
  lambda = lambda_vals[j]
  sample_mean = numeric(length = M)
  sample_var = numeric(length = M)
  for(i in 1:M){
    x = rpois(n = n, lambda = lambda)
    sample_mean[i] = mean(x)
    sample_var[i] = var(x)
  }
  risk_sample_mean[j] = mean((sample_mean - lambda)^2)
  risk_sample_var[j] = mean((sample_var - lambda)^2)
}
plot(lambda_vals, risk_sample_var, type = "p",
     col = "red", lwd = 2, ylab = expression(R[T](lambda)),
     xlab = expression(lambda), cex = 1.3)

lines(lambda_vals,lambda_vals/n + 2*lambda_vals^2/(n-1), col = "black",
      lwd  = 3, lty = 2)
points(lambda_vals, risk_sample_mean, type = "p", col = "magenta", lwd = 2, cex = 1.3)
legend("topleft", legend = c(expression(bar(X[n])), expression(S[n]^2)),
       col = c("magenta", "red"), bty = "n", lwd = c(2,2))
lines(lambda_vals, lambda_vals/n, col = "black",
      lwd  = 3, lty = 2)
```

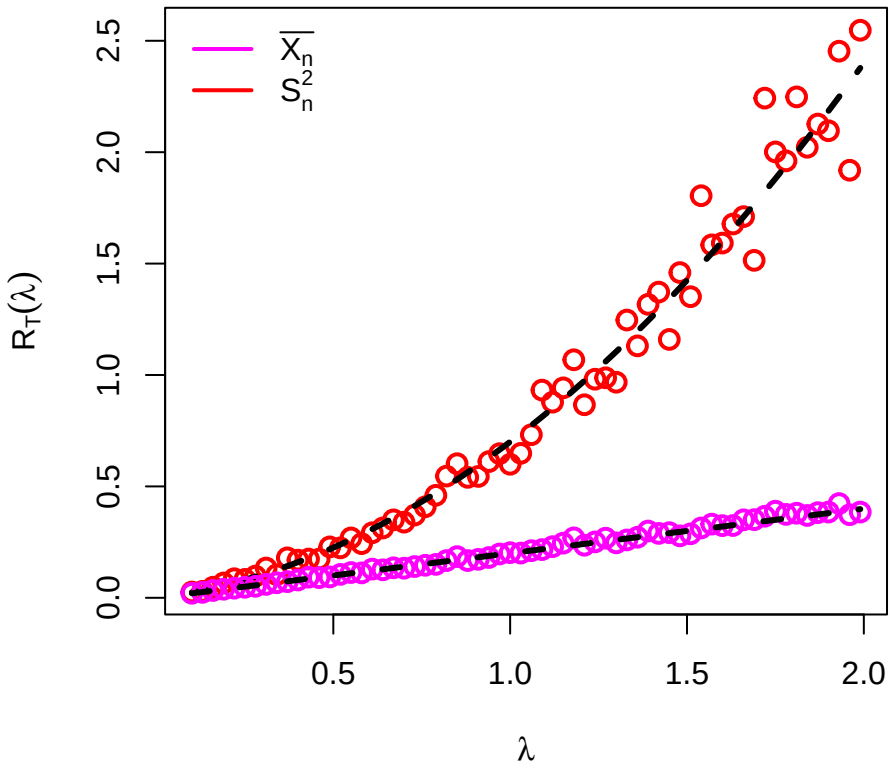


Figure 7: Approximation of the risk function for the sample mean and sample variance in estimating the parameter λ for the Poisson distribution. It is obvious from the plot that the sample sample mean has a lower risk as compared the sample variance. Therefore, $\overline{X_n}$ is a better estimator as compared S_n^2, although, both are unbiased estimators of λ. The magenta lines indicate the exact risk function obtained by theoretical computation.

In the classroom, a student named Vaishnavi asked which method would likely converge faster to the true value. The idea is related to the consistency, which means that as $n \to \infty$, the following claims are true or not.

$$\overline{X_n} \xrightarrow{P} \lambda, \qquad S_n^2 \xrightarrow{P} \lambda$$

To answer this question, let us assume that the true value is $\lambda = \lambda_0 = 2$. We simulate the a sample of size n from the Poisson(λ_0) and compute $\overline{X_n}$ and S_n^2 and check how these random quantities behave as $n \to \infty$. The following code will do this task.

```
par(mfrow = c(1,1))
lambda_0 = 2
n_vals = 1:1000
sample_mean = numeric(length = length(n_vals))
```

```r
sample_var = numeric(length = length(n_vals))
for(n in n_vals){
  x = rpois(n = n, lambda = lambda_0)
  sample_mean[n] = mean(x)
  sample_var[n] = var(x)
}
plot(n_vals, sample_var, col = "grey", lwd = 2,
     xlab = "sample size (n)", type = "l",
     ylab = "Estimator")
lines(n_vals, sample_mean, col = "red", lwd = 2)
legend("topright", legend = c(expression(bar(X[n])),
                              expression(S[n]^2)),
       col = c("grey", "red"), lwd = c(2,2), bty = "n")
```

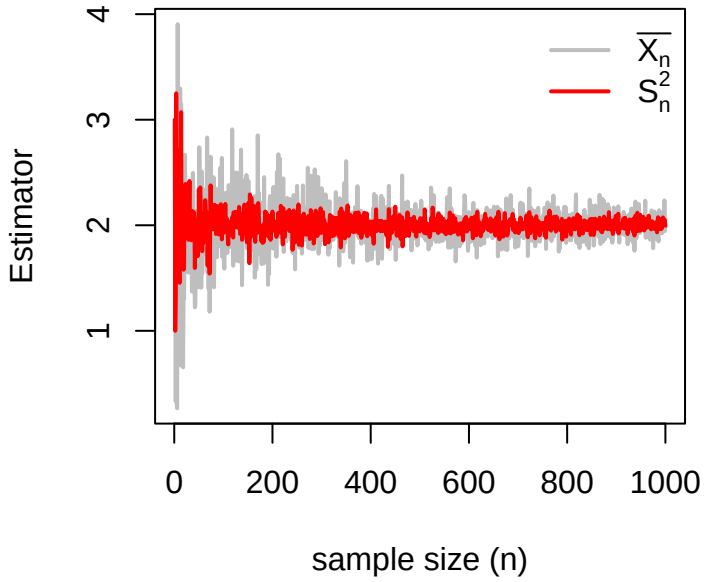


Figure 8: It can be observed that as the sample size increases, the convergence of the Sample variance is much faster as compared to the sample mean to the true value of $\lambda = 2$.

The following is the python implementation of the above convergence concept. Thanks to Sangeeta for the Python implementation.

```python
import numpy as np
import matplotlib.pyplot as plt

lambd_0 = 2
```

```
n_vals = np.linspace(1,1000,num=1000)
sample_mean = []
sample_var = []
for n in n_vals:
  x = np.random.poisson(lambd_0,int(n))
  sample_mean.append(np.mean(x))
  sample_var.append(np.var(x))
plt.plot(n_vals,sample_var,c="r",label="$S^2_{n}$")
plt.plot(n_vals,sample_mean,c="g",label="$\overline {X_{n}}$")
plt.legend()
plt.xlabel("sample size (n)")
plt.ylabel("Estimator")
plt.figure(figsize=(1,1))
plt.show()
```

Figure 9: It can be observed that as the sample size increases, the convergence of the Sample variance is much faster as compared to the sample mean to the true value of $\lambda = 2$.

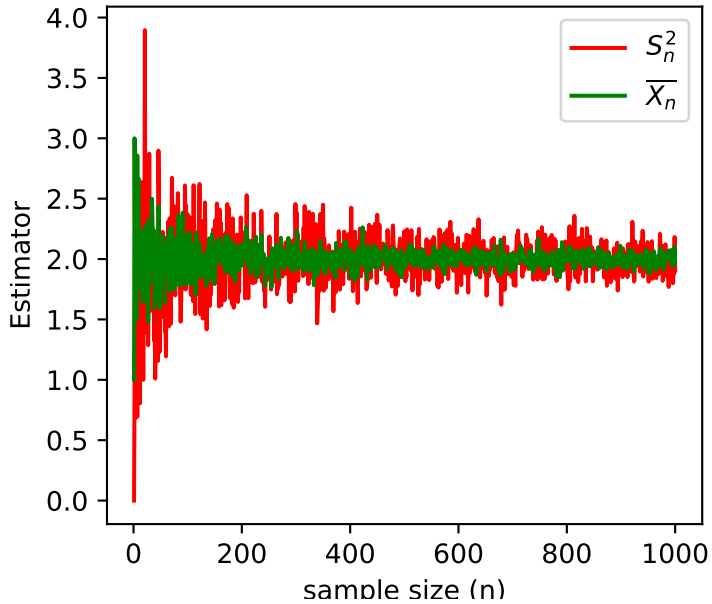


Figure 10: It can be observed that as the sample size increases, the convergence of the Sample variance is much faster as compared to the sample mean to the true value of $\lambda = 2$.

Exact Computation of the variance of S_n^2

We recall that if we have a random sample of size n from the a population with mean μ and variance σ^2. Then

$$Var\left(\overline{X_n}\right) = \frac{\sigma^2}{n}.$$

In addition, $S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X_n}\right)^2$ is an unbiased estimator of σ^2. To compute the variance $Var(S_n^2)$, we observe that the sample variance can also be expressed as

$$S_n^2 = \frac{1}{2n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(X_i - X_j\right)^2.$$

The following general results hold for all population distributions with finite fourth order moment.

- $\mathrm{E}\left(S_n^2\right) = \sigma^2$.
- $Var\left(S_n^2\right) = \frac{1}{n}\left(\mu_4 - \frac{n-3}{n-1}\mu_2^2\right)$, where $\mu_1 = \mathrm{E}(X_i)$ and $\mu_j = \mathrm{E}\left(X_i - \mu_1\right)^j, j = 2, 3, 4$.
- In addition, the covariance between $\overline{X_n}$ and S_n^2, $Cov\left(\overline{X_n}, S_n^2\right)$ can also be expressed in terms of $\mu_1, \mu_2, \mu_3, \mu_4$.

For our problem, the population follows the Poisson distribution with parameter λ. Therefore, $\mathrm{E}(S_n^2) = \lambda$. The fourth order central moment for the Poisson distribution:

$$\mu_4 = \mathrm{E}(X-\lambda)^4 = \sum_{i=0}^{4} \binom{4}{i} \mathrm{E}(X^{4-i})\lambda^i = 3\lambda^2 + \lambda.$$

The raw moments $\mathrm{E}(X^i), i = 1, 2, 3, 4$, can be computed using the Moment Generating function by taking derivatives and evaluating at $t = 0$. The MGF of Poisson distribution is given by

$$M_X(t) = e^{\lambda(e^t-1)}, -\infty < t < \infty.$$

$$\begin{aligned} M_X'(0) &= \lambda && (0.1)\\ M_X''(0) &= \lambda^2 + \lambda && (0.2)\\ M_X'''(0) &= \lambda^3 + 3\lambda^2 + \lambda && (0.3)\\ M_X''''(0) &= \lambda^4 + 6\lambda^3 + 7\lambda^2 + \lambda && (0.4) \end{aligned}$$

This gives

$$\mu_4 = 3\lambda^2 + \lambda.$$

Therefore, the variance of S_n^2 is given by

$$\begin{aligned} Var(S_n^2) &= \frac{1}{n}\left(3\lambda^2 + \lambda - \frac{n-3}{n-1}\lambda^2\right) && (0.5)\\ &= \frac{\lambda}{n} + \frac{2\lambda^2}{n-1}. && (0.6) \end{aligned}$$

In the following R code, we add the analytically computed risk function with the risk function estimated by using simulation. The exact risk function is given by

$$\mathcal{R}(S_n^2, \lambda) = \frac{\lambda}{n} + \frac{2\lambda^2}{n-1}, 0 < \lambda < \infty.$$

Making this problem more interesting

We found that both $\overline{X_n}$ and S_n^2 are unbiased estimators of λ. However, after comparing the risk functions, we found that

$$\mathcal{R}\left(\overline{X_n}, \lambda\right) \leq \mathcal{R}\left(S_n^2, \lambda\right)$$

for all $\lambda \in (0, \infty)$. We can extend this to a class of estimators defined for every constant a as follows:

$$W_a\left(\overline{X_n}, S_n^2\right) = a\overline{X_n} + (1-a)S_n^2.$$

For every a, the estimator W_a is an unbiased estimator of λ. Although $\overline{X_n}$ is better that S_n^2, is it better than W_a for all constants a? In other words, is the following statements true?

$$\mathcal{R}\left(\overline{X_n}, \lambda\right) = \mathcal{R}\left(W_1, \lambda\right) \leq \mathcal{R}\left(W_a, \lambda\right), \text{ for all } \quad 0 < \lambda < \infty.$$

Before getting into some statistical theories that may guide us to determine whether $\overline{X_n}$ is indeed the best choice, you can plot the risk function of $\mathcal{R}\left(W_a, \lambda\right), 0 < \lambda < \infty$ for different choices of a using computer simulation. Note that that for $a = 1$ (corresponds to $\overline{X_n}$) and $a = 0$ (corresponds to S_n^2) risk functions are already obtained in the previous section. In addition, check that whether for certain values of a, the risk function falls below $\mathcal{R}\left(\overline{X_n}, \lambda\right)$ for some λ values. In the following, let us try to compute the risk of W_a.

$$\begin{aligned}
\mathcal{R}(W_a, \lambda) &= \mathrm{E}_\lambda (W_a - \lambda)^2 && (0.7)\\
&= \mathrm{E}\left[a\left(\overline{X_n} - \lambda\right) + (1-a)\left(S_n^2 - \lambda\right)\right]^2 && (0.8)\\
&= a^2 \mathcal{R}\left(\overline{X_n}, \lambda\right) + (1-a)^2 \mathcal{R}\left(S_n, \lambda\right) + 2a(1-a) \times \mathrm{Cov}\left(\overline{X_n}, S_n^2\right) && (0.9)\\
&= a^2 \frac{\lambda}{n} + (1-a)^2 \left(\frac{\lambda}{n} + \frac{2\lambda^2}{n-1}\right) + 2a(1-a)\frac{\lambda}{n} && (0.10)
\end{aligned}$$

To obtain the best choice of a, we need to minimize the risk function as a function of a. The equation $\frac{d}{da}\mathcal{R}(W_a, \lambda) = 0$, gives $a^* = 1$ and $\frac{d^2}{da^2}\mathcal{R}(W_a, \lambda)|_{a=1} = \frac{2\lambda^2}{n-1} > 0$. Therefore the minimum risk is obtained what $a = 1$, which corresponds to the sample mean $\overline{X_n}$ for all $\lambda \in (0, \infty)$.

Therefore, we are able to establish that if we consider the class of estimators W_a (unbiased) indexed by constant $a \in \mathbb{R}$, then $\overline{X_n} = W_1$ is the best estimator in this class when compared with respect to the risk function under the squared error loss.

However, it does not guarantee that the sample mean is the best estimator amongst all estimators of λ existing in this globe. Therefore, we need to develop theories which will be helpful to determine whether the sample mean is indeed THE BEST CHOICE to estimate the unknown parameter λ.

A natural extension is to extend the class to any unbiased estimator of λ. That is can we claim that for any unbiased estimator T_n of λ,

$$\mathcal{R}\left(\overline{X_n}, \lambda\right) \leq \mathcal{R}\left(T_n, \lambda\right)$$

for all $\lambda \in (0, \infty)$.

Search for the holy grail (the best estimator)

See Next Chapter (Unbiased Estimation)

Conceptual Exercises

Suppose that a random sample $(X_1, X_2, \ldots, X_n)$ of size n from a population which is characterized by the following probability density function:

$$f(x) = \begin{cases} \frac{2x}{\theta^2}, & 0 < x < \theta \\ 0, & \text{otherwise} \end{cases},$$

where $\theta \in \Theta = (0, \infty)$ be the parameter space. We are interested in estimating the parameter θ based on the given random sample. Let $Y_n = \max(X_1, X_2, \ldots, X_n)$ be the maximum order statistics and you decided to estimate θ using Y_n. Answer the following:

1. Plot the above PDF for different choices of θ in a single plot window. Obtain the CDF of the given PDF and plot the CDF different choices of θ in a single plot window. Make a side by side plot using `par(mfrow = c(1,2))`.

2. Obtain the exact sampling distribution of Y_n and write down the both probability density function and the cumulative distribution function of Y_n.

3. Under the squared error loss function, compute the risk function $\mathcal{R}(Y_n, \theta), 0 < \theta < \infty$. Also, plot the risk function.

4. Show that as $n \to \infty$, $Y_n \to \theta$ in probability. For a fixed $\epsilon > 0$, compute the limit $\lim_{n\to\infty} \mathrm{P}(|Y_n - \theta| > \epsilon)$ and show the limit is equal to zero as $n \to \infty$. Another possibility is to apply some inequality to bound the $\mathrm{P}(|Y_n - \theta| > \epsilon)$ with constant a_n which converges to 0 as $n \to \infty$ (Hint: Markov Inequality). Write a computer simulation to visualize that the largest order statistics Y_n indeed converges to θ as $n \to \infty$.

5. Compute the expected value of Y_n and compute the bias of the estimator Y_n, which is given by
$$\mathrm{Bias}_\theta(Y_n) = \mathrm{E}(Y_n) - \theta, \theta \in (0, \infty).$$
Also check whether the bias $\mathrm{Bias}_\theta(Y_n)$ tends to zero as $n \to \infty$. That means is the estimator is asymptotically unbiased? Compute the value of the constant c_n so that
$$\mathrm{E}_\theta(c_n Y_n) = \theta \text{ for all } \theta \in (0, \infty).$$

6. Compute the variance of Y_n, which can be computed as
$$Var_\theta(Y_n) = \mathrm{E}_\theta(Y_n^2) - (\mathrm{E}_\theta(Y_n))^2.$$

7. Show the following identity holds for all $\theta \in (0, \infty)$:
$$\mathcal{R}(Y_n, \theta) = \mathrm{E}_\theta(Y_n - \theta)^2 = Var_\theta(Y_n) + (\mathrm{Bias}_\theta(Y_n))^2.$$

8. Using computer simulation, approximate the risk function Y_n, $\mathcal{R}(Y_n, \theta), \theta \in (0, \infty)$. The challenge, the reader may find is to simulate from the population distribution as some inbuilt function in R may not be available. To simulate a random number from the given distribution (a) Compute the CDF $F_X(x)$, (b) Simulate $U \sim \text{Uniform}(0, 1)$. (c) Compute $X = F_X^{-1}(U)$, which is the inverse image of U under F_X. Then $X \sim f(x)$. Overlay the analytically computed risk function on the plot of the risk function obtained by computer simulation.

Another illustrative example

Suppose that we have a sample of size n $(X_1, X_2, \ldots, X_n)$ from a population which is characterized by the following probability density function and our goal is to estimate the parameter $\theta \in (0, \infty)$.

$$f(x) = \begin{cases} \frac{\theta}{(1+x)^{1+\theta}}, 0 < x < \infty \\ 0, \quad \text{otherwise} \end{cases}$$

At the first step, we compute the cumulative distribution function and plot both PDF and CDF for different choices of θ values.

$$F_X(x) = \begin{cases} 1 - (1+x)^{-\theta}, 0 \leq x < \infty \\ 0, \quad \text{otherwise} \end{cases}$$

```
theta = 3
f = function(x){
  (theta/(1+x)^(1+theta))*(x>0)
}

F = function(x){
  (1 - (1+x)^(-theta))*(x>0)
}
par(mfrow = c(1,2))
curve(f(x), -1, 4, col= "red", lwd = 2)
curve(F(x), -1, 4, col = "red", lwd = 2)
```

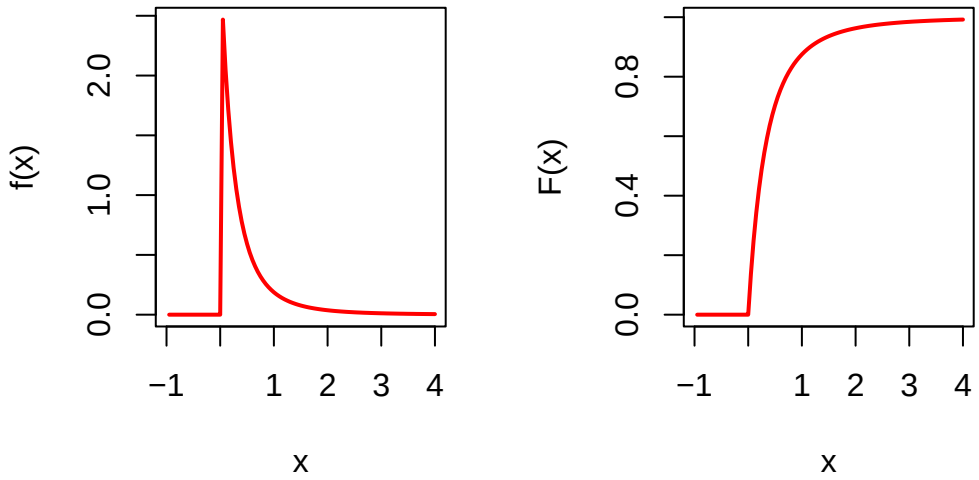


Figure 11: The PDF and CDF of the population random variable X is shown. The value of $\theta = 3$ is considered. The reader is encouraged to modify the R codes to draw different shapes for different choices of θ values.

Suppose one starts with the maximum and minimum order statistics to estimate the parameter θ, which are defined as $Y_n = \max(X_1, \ldots, X_n)$ and $Y_1 = \min(X_1, X_2, \ldots, X_n)$. First of all, we compute the sampling distribution of Y_1 and Y_n and plot the PDFs.

$$f_{Y_1}(y) = \begin{cases} n\theta(1+y)^{-(n\theta+1)}, 0 < y < \infty, \\ 0, \text{otherwise.} \end{cases}$$

$$f_{Y_n}(y) = \begin{cases} n\left(1-(1+y)^{-\theta}\right)^{n-1} \frac{\theta}{(1+y)^{1+\theta}}, 0 < y < \infty, \\ 0, \text{otherwise.} \end{cases}$$

```
par(mfrow = c(1,2))
n = 30
f_Y1 = function(x){
  (n*theta*(1+x)^(-n*theta-1))*(x>0)
}
curve(f_Y1(x), -1, 4, col = "red", lwd = 2,
      main = paste("n = ",n),
      ylab = expression(f[Y[1]](y)), xlab = "y")

f_Yn = function(x){
  n*((1-(1+x)^(-theta))^(n-1))*theta/(1+x)^(1+theta)*(x>0)
}
curve(f_Yn(x), -1, 10, col = "red", lwd = 2,
```

```
14        main = paste("n = ",n),
15        ylab = expression(f[Y[n]](y)), xlab = "y")
```

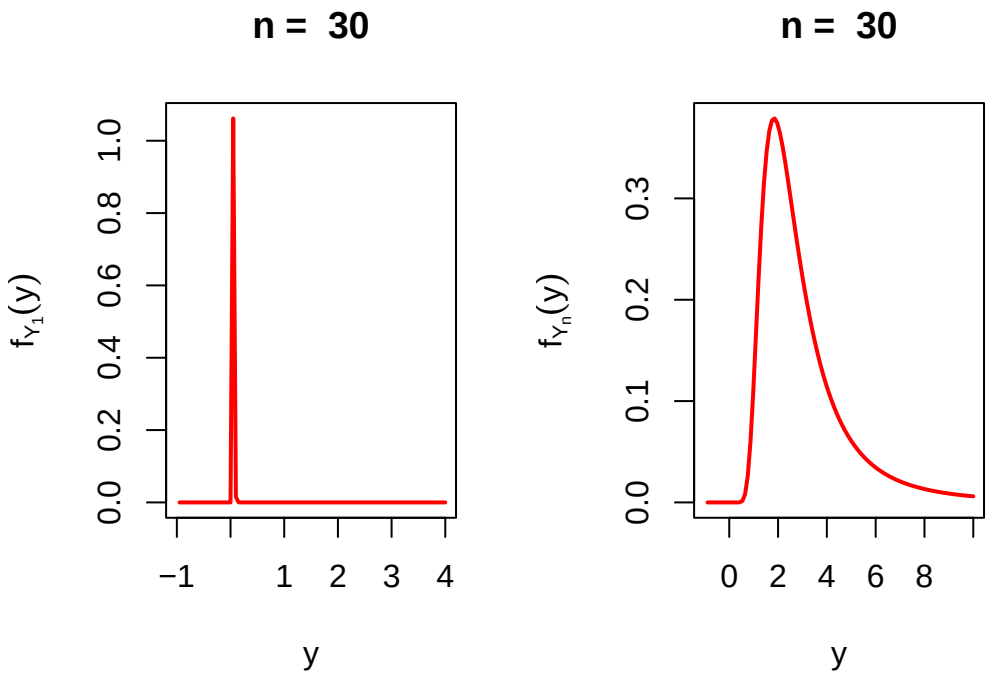


Figure 12: The sampling distribution of the Y_1 and Y_n. The value of θ is set to 3. It is clear that Y_1 is highly concentrated at 0 and the concentration increases as n becomes large. Therefore, Y_1 does not appear to be a good choice for estimating θ. The sample size is fixed at $n = 30$.

One may probably think of using the maximum order statistics $Y_n = \max(X_1, X_2, \ldots, X_n)$ to estimate the parameter θ. Let us investigate the properties of the estimator Y_n. At the first step we can compute the mean and higher order moments of Y_n.

The mean of Y_n can be computed by computing the expectation of $Y_n + 1$ as follows:

$$\begin{aligned} \mathrm{E}_\theta(Y_n + 1) &= \int_0^\infty (1+y)n\left[1-(1+y)^{-\theta}\right]^{n-1} \cdot \frac{\theta}{(1+y)^{1+\theta}} dy \qquad (0.11) \\ &= n\mathcal{B}\left(1-\frac{1}{\theta}, n\right), \theta \in (1,\infty) \qquad (0.12) \end{aligned}$$

The expectation exists only when $\theta \in (1, \infty)$. More importantly, $\mathrm{E}(Y_n) \neq \theta$, therefore, Y_n is not an unbiased estimator of θ. A similar computation will give us $\mathrm{E}_\theta\left(Y_n + 1\right)^2 = n\mathcal{B}\left(1 - \frac{2}{\theta}, n\right), \theta \in (2, \infty)$. The above result can be generalized as

$$\mathrm{E}_\theta(Y_n+1)^m = \int_0^\infty (1+y)^m \cdot n\left[1-(1+y)^{-\theta}\right]^{n-1} \cdot \frac{\theta}{(1+y)^{1+\theta}} dy \tag{0.13}$$

$$= n\mathcal{B}\left(1-\frac{m}{\theta}, n\right), \theta \in (m, \infty). \tag{0.14}$$

Then mth order raw moments exists when $\theta \in (m, \infty)$. Use the idea of change of variable as $z = (1+y)^{-\theta}$ to compute the above integrals. Therefore, we can evaluate the risk of Y_n under the squared error loss as

$$\mathcal{R}(Y_n, \theta) = \mathrm{E}_\theta\left(Y_n - \theta\right)^2 = \mathrm{E}_\theta\left[(Y_n+1) - (\theta+1)\right]^2 \tag{0.15}$$

$$= n\mathcal{B}\left(1-\frac{2}{\theta}, n\right) - 2n(\theta+1)\mathcal{B}\left(1-\frac{1}{\theta}, n\right) + (1+\theta)^2 \tag{0.16}$$

We need to keep in mind that the risk function exists only when $\theta \in (2, \infty)$. Let us plot the risk function for different choices of θ. The following figure suggests that the performance of this estimator is reasonable at a small subset of the parameter space.

```r
risk_fun_Yn = function(theta){
  n*beta(1-2/theta,n)-2*n*(theta+1)*beta(1-1/theta,n) +(1+theta)^2
}
curve(risk_fun_Yn(x), 2.3, 10, col = "red", lwd = 2,
      ylab = expression(R(Y[n],theta)), xlab = expression(theta))
```

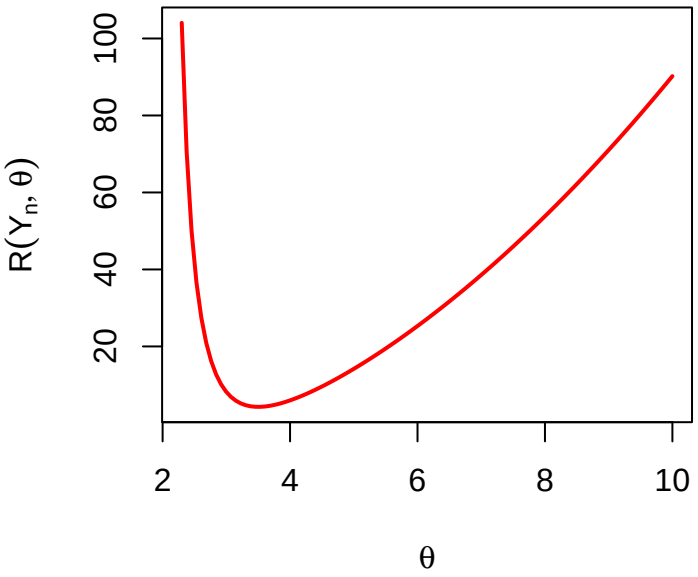


Figure 13: The risk function of Y_n as a function of θ. The sample size is fixed at $n = 10$.

Let us check how the shapes changes with respect to sample size n. The most important observation from Figure 14 is that as $n \to \infty$, $\mathcal{R}(Y_n, \theta) \not\to 0$, therefore Y_n is not a mean squared consistent estimator, which is certainly not a desirable property. The example of Y_n is just for an illustration purpose, how for a given statistic what properties to look at. In the following, we compare two estimators of θ, one is the method of moment and the other one is the MLE.

```
n_vals = c(10,20,50,100,500)
n = 5
par(mfrow = c(1,1))
curve(risk_fun_Yn(x), 2.3, 10, col = "red", lwd = 2,
      ylab = expression(R(Y[n],theta)), xlab = expression(theta))
for(i in 1:length(n_vals)){
  n = n_vals[i]
  curve(risk_fun_Yn(x), col = i, lwd = 2,
        lty = i, add = TRUE)
}
```

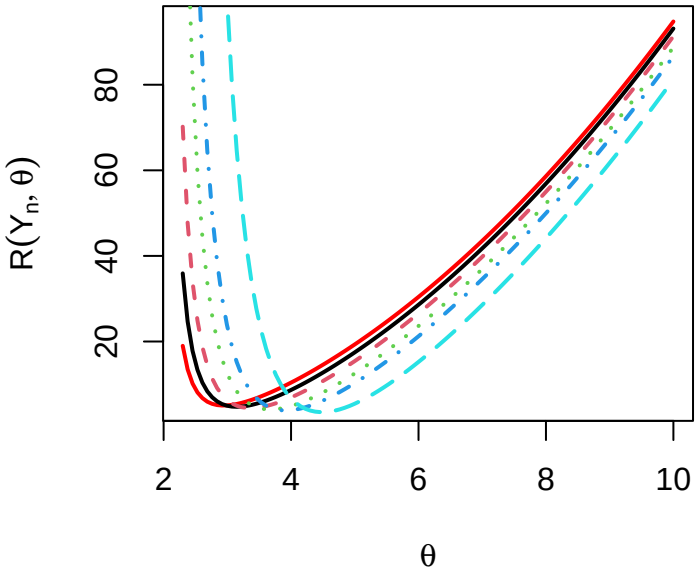


Figure 14: The risk function of the maximum order statistics for different choices of the sample size n under the squared error loss.

Method of moments Estimator

Using the method of moments, we can obtain an estimate of θ. The population mean $\mathrm{E}(X) = \frac{1}{\theta - 1}$, which exists only when $\theta \in (1, \infty)$. Therefore, equating the sample mean $\overline{X_n}$ with

population mean, we obtain the method of moment estimator of θ as

$$\widehat{\theta_{\text{MOM}}} = \frac{1}{\overline{X_n} + 1} = V_n(\text{say})$$

Method of Maximum Likelihood

Using the method of maximum likelihood, we can also obtain the estimator of θ as follows: The likelihood function is given by

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} \frac{\theta}{(1+x_i)^{1+\theta}}, 0 < \theta < \infty$$

and the log-likelihood function is given by

$$l(\theta) = n \log \theta - n \sum_{i=1}^{n} \log(1 + x_i).$$

Equating $l'(\theta) = 0$ implies, $\theta^* = \frac{n}{\sum_{i=1}^{n} \log(1+x_i)}$ and $l''(\theta) = -\frac{n}{\theta^2} < 0$ for all $\theta \in (0, \infty)$. Therefore, $l(\theta)$ attains its maximum at θ^*. Therefore, the Maximum Likelihood Estimator of θ is given by

$$\widehat{\theta_{\text{MLE}}} = \frac{n}{\sum_{i=1}^{n} \log(1 + X_i)} = W_n \quad (\text{say}).$$

As per our notation, V_n and W_n are the method of moments and maximum likelihood estimators of θ, respectively. Now we have three estimators of θ and we are interested in comparing their risk functions which are listed as $\mathcal{R}(Y_n, \theta)$, $\mathcal{R}(V_n, \theta)$ and $\mathcal{R}(W_n, \theta)$. It is important to note that the risk functions may exists only on a subset of the parameter space $\Theta = (0, \infty)$.

We first try to compute the sampling distribution of W_n. We compute it step by step.

- `Step - I` Consider $Y = \log(1 + X)$, then the CDF of Y is given by

$$\begin{aligned} F_Y(y) = \mathrm{P}(Y \leq y) &= \mathrm{P}(\log(1 + X) \leq y) && (0.17) \\ &= \mathrm{P}\left(X \leq e^y - 1\right) && (0.18) \\ &= 1 - e^{-\theta y}, 0 < y < \infty. && (0.19) \end{aligned}$$

Therefore, $Y = \log(1 + X) \sim$ Exponential(rate $= \theta$) which is $\mathcal{G}\left(\alpha = 1, \beta = \frac{1}{\theta}\right)$ distribution.

- `Step - II` Let $Y_i = \log\left(1 + X_i\right), 1 \leq i \leq n$, then by using the Moment Generating Function technique, we can easily show that $Z_n = \sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \log(1 + X_i) \sim \mathcal{G}(\alpha = n, \beta = \frac{1}{\theta})$, whose PDF is given by

$$f_{Z_n}(z) = \frac{\theta^n z^{n-1} e^{-\theta z}}{\Gamma(n)}, 0 < z < \infty.$$

  and zero otherwise.

- `Step - III` The MLE $W_n = \frac{n}{Z_n}$. Using the transformation formula, we obtain $f_{W_n}(w)$ as

$$f_{W_n}(w) = f_{Z_n}\left(\frac{n}{w}\right)\left|\frac{d}{dw}\left(\frac{n}{w}\right)\right|, 0 < w < \infty,$$

which gives the PDF of W_n as

$$f_{W_n}(w) = \frac{n(n\theta)^n e^{-\frac{n\theta}{w}}}{w^{n+2}\Gamma(n)}, 0 < w < \infty,$$

and zero otherwise. In the following R code, we visualize the sampling distribution of the MLE of θ (PDF) for different choices of θ and sample size n.

```
par(mfrow = c(1,2))
f_Wn = function(w){
  n*(n*theta)^n*exp(-n*theta/w)/(w^(n+2)*gamma(n))*(w>0)
}
theta = 3
n = 20
curve(f_Wn(x), -0.5, 7, col = "red", lwd = 2, xlab = expression(w),
      ylab = expression(f[W[n]](w)), main = bquote(theta == .(theta)))
n = 10
curve(f_Wn(x), add= TRUE, col = "blue", lwd = 2)
n = 5
curve(f_Wn(x), add= TRUE, col = "magenta", lwd = 2)
points(theta, 0, pch = 19, col = "green", cex = 1.3)
legend("topright", legend = c("n = 5", "n = 10", "n = 20"),
       col = c("magenta", "blue", "red"), lwd = c(2,2,2), bty = "n")

n = 10
theta = 3
curve(f_Wn(x), -0.5, 8, col = "red", lwd = 2, xlab = expression(w),
      ylab = expression(f[W[n]](w)), main = bquote(n == .(n)))
theta = 5
curve(f_Wn(x), add= TRUE, col = "magenta", lwd = 2)
legend("topright", legend = c(bquote(theta == 3), bquote(theta == 5)),
       col = c("red", "blue"), lwd = c(2,2), bty = "n")
```

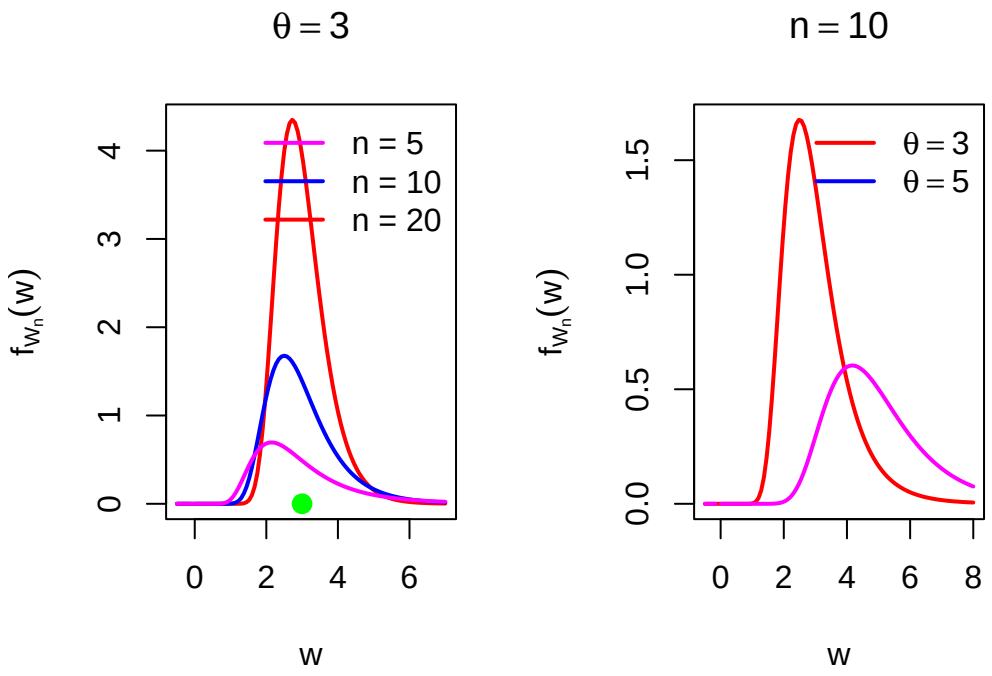


Figure 15: The sampling distribution of the MLE W_n of θ for different choices of θ and different sample size. Left panel: As the sample size increases, the sampling distribution of W_n is highly concentrated about the true value of $\theta = 3$, which is a desirable property of the MLE (consistency).

- `Step - IV` To compute the risk $\mathcal{R}(W_n, \theta)$, we compute

$$\mathrm{E}(W_n) = \frac{n\theta}{n-1}, n > 1,$$

and

$$\mathrm{E}\left(W_n^2\right) = \frac{n^2\theta^2}{(n-1)(n-2)}, n > 2.$$

Therefore, the risk under the squared error loss is given by

$$\begin{aligned} \mathrm{E}_\theta\left(W_n - \theta\right)^2 &= \mathrm{E}_\theta(W_n^2) - 2\theta\mathrm{E}_\theta(W_n) + \theta^2 \quad (0.20) \\ &= \frac{\theta^2(n+2)}{(n-1)(n-2)}, n > 2. \quad (0.21) \end{aligned}$$

The following observation is important.

- $\mathrm{Bias}_\theta(W_n) = \mathrm{E}_\theta(W_n) - \theta = \frac{n\theta}{n-1} - \theta = \frac{\theta}{n-1} \to 0$ as $n \to \infty$. Therefore, W_n is asymptotically unbiased estimator.

- $\mathcal{R}(W_n, \theta) = \mathrm{E}_\theta(W_n - \theta)^2 \to 0$ as $n \to \infty$. Therefore, $W_n \to \theta$ in probability as $n \to \infty$ (a simple application of Markov Inequality).

Let us now verify whether the theoretical computation of the risk function for the W_n is supported by the simulation study as well. We follow the same scheme as previous.

- Fix θ and fix n, sample size.
- Simulate $X_1, X_2, \ldots, X_n \sim f(x|\theta)$
- Compute W_n
- Compute the loss $(W_n - \theta)^2$
- Repeat the above three steps M times to compute the average loss.
- Repeat above five steps for different choices of θ from the parameter space.
- Plot the average loss values (approximate risk) at each value of θ.

In the above algorithm, primary challenge is to simulate from the given PDF as some inbuilt function may not be available to directly simulate using R or Python. Let us use the probability integral transform to simulate $X_1, \ldots, X_n \sim f(x|\theta)$. Simulate $U \sim \text{Uniform}(0, 1)$ and compute $X = F_X^{-1}(U)$, where F_X^{-1} is the inverse of the CDF. In this case, the equation to generate $X \sim f(x|\theta)$ becomes

$$X = (1 - U)^{-\frac{1}{\theta}} - 1, 0 < U < 1.$$

In the following first verify whether the simulation study.

```r
n = 100
theta = 3                                       # true parameter value
u = runif(n = n, min = 0, max = 1)              # simulate U(0,1)
x = (1-u)^(-1/theta) - 1                        # probability integral transform
hist(x, probability = TRUE, main = bquote(theta == .(theta)),
     breaks = 30)                               # histogram
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the original pdf
```

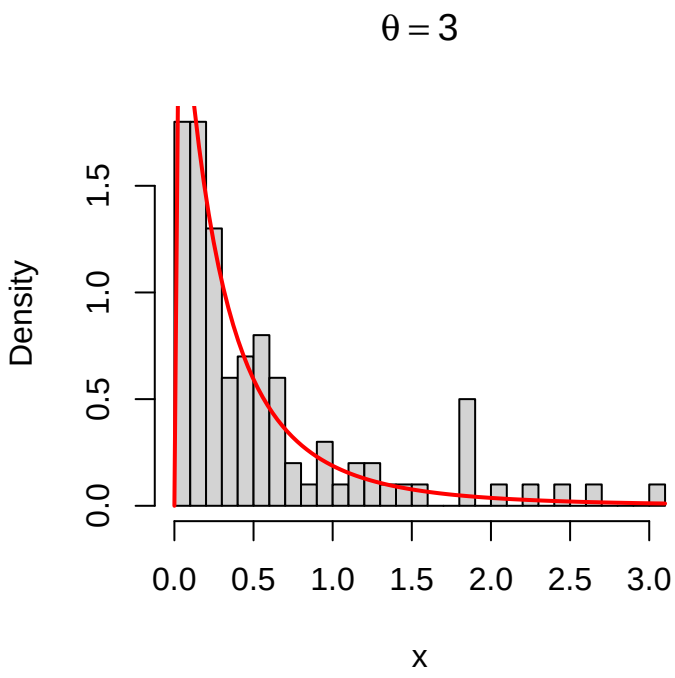


Figure 16: simulated realization from the given PDF $f(x;\theta = 3)$ using probability integral transform.

Let us now execute simulation of the risk function using computer simulation and verify with the theoretical result.

```
M = 1000            # number of replications
n = 10
theta_vals = seq(0.1, 4, by = 0.1)
risk_vals = numeric(length = length(theta_vals))
for(i in 1:length(theta_vals)){
  theta = theta_vals[i]
  loss_vals = numeric(length = M)
  for(j in 1:M){
    u = runif(n = n, min = 0, max = 1)
    x = (1-u)^(-1/theta) - 1
    W_n = n/sum(log(1+x))
    loss_vals[j] = (W_n - theta)^2
  }
  risk_vals[i] = mean(loss_vals)
}
plot(theta_vals, risk_vals, col = "red", pch = 19,
     xlab = expression(theta), ylab = expression(R(W[n],theta)))
```

```r
curve(x^2*(n+2)/((n-1)*(n-2)), lwd = 3, col = "blue",
      add = TRUE, lty = 2)
```

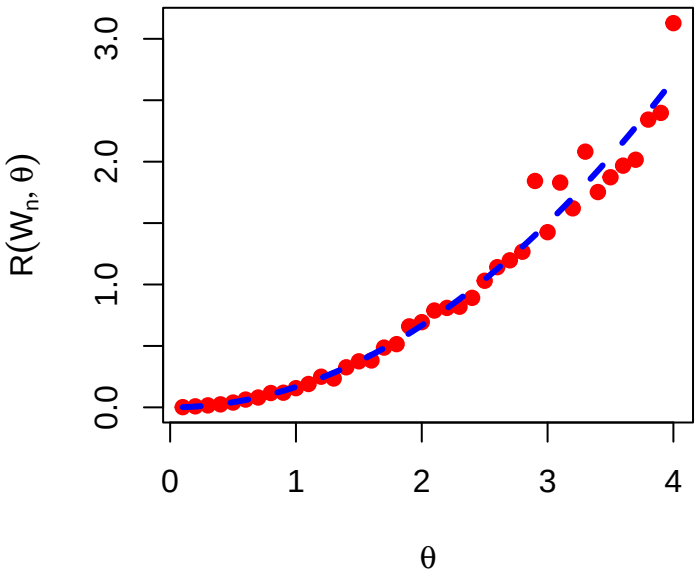


Figure 17: The risk function is obtained by computer simulation for the MLE W_n of θ. True risk function is added for reference (blue dotted line).

The MLE W_n has desirable properties. We are yet to check it for the Method of Moment estimator V_n.

Approximation of $\mathcal{R}(V_n, \theta)$

The exact sampling distribution of the method of moment estimator $V_n = \overline{X_n}^{-1} + 1$ is difficult to obtain. Therefore, an analytically tractable expression of the risk function is almost impossible to obtain. However, we can possibly obtain an approximation of the risk function for large sample size n for a subset of the parameter space $\Theta = (0, \infty)$. In the following, we first obtain the risk function by computer simulation.

```r
M = 500             # number of replications
n = 10
theta_vals = seq(0.1, 10, by = 0.1)
risk_vals = numeric(length = length(theta_vals))
for(i in 1:length(theta_vals)){
  theta = theta_vals[i]
  loss_vals = numeric(length = M)
```

```
  for(j in 1:M){
    u = runif(n = n, min = 0, max = 1)
    x = (1-u)^(-1/theta) - 1
    V_n = 1/mean(x)+1
    loss_vals[j] = (V_n - theta)^2
  }
  risk_vals[i] = mean(loss_vals)
}
plot(theta_vals, risk_vals, col = "red", pch = 19,
     xlab = expression(theta), ylab = expression(R(V[n],theta)))
```

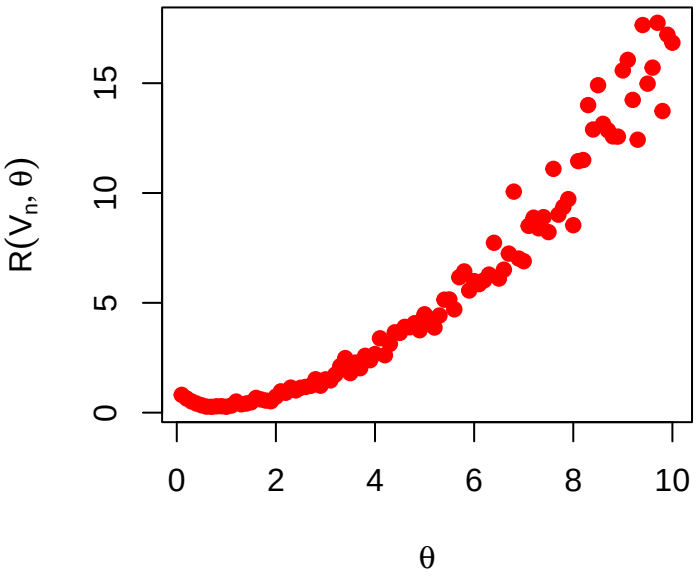


Figure 18: The risk function is obtained by computer simulation for the MLE W_n of θ.

$\mathrm{E}(X) = \mu = \frac{1}{\theta-1}$ exists for $\theta > 1$ and $Var(X) = \sigma^2 = \frac{2}{(\theta-1)(\theta-2)}$ which exists for $\theta \in (2, \infty)$. Therefore, when $\theta \in (2, \infty)$, we can apply the CLT, which ensures the large sample approximation of the sampling distribution of $\overline{X_n}$ as

$$\overline{X_n} \sim \mathcal{N}\left(\frac{1}{\theta-1}, \frac{2}{n(\theta-1)(\theta-2)}\right).$$

Now we can approximate the sampling distribution of $g(\overline{X_n}) = \overline{X_n}^{-1} + 1$ by using first order Taylor's approximation and $g(\mu) = \frac{1}{\mu} + 1 = \theta$.

$$g\left(\overline{X_n}\right) \approx g(\mu) + \left(\overline{X_n} - \mu\right) g'(\mu)$$

which implies

$$V_n = \frac{1}{\overline{X_n}} + 1 \approx \frac{1}{\mu} + 1 + (\overline{X_n} - \mu)\left(-\frac{1}{\mu^2}\right) = \theta + \left(\overline{X_n} - \frac{1}{\theta - 1}\right)(-(\theta-1)^2)$$

Therefore, by the first order Taylor Approximation,

$$\mathrm{E}_\theta(V_n) \approx \theta$$

and the variance is obtained as

$$\begin{aligned} Var_\theta(V_n) &\approx \mathrm{E}_\theta\left(V_n - \theta\right)^2 && (0.22)\\ &= \mathrm{E}_\theta\left(\overline{X_n} - \frac{1}{\theta-1}\right)^2 (\theta-1)^4 && (0.23)\\ &= Var_\theta\left(X_n\right)(\theta-1)^2 && (0.24)\\ &\approx \frac{1}{n(\theta-1)(\theta-2)}(\theta-1)^4 = \frac{(\theta-1)^3}{n(\theta-2)}. && (0.25) \end{aligned}$$

Therefore, by the application of Delta method

$$V_n = g\left(\overline{X_n}\right) \sim \mathcal{N}\left(\theta, \frac{(\theta-1)^3}{n(\theta-2)}\right), \text{for large } n, \theta \in (2, \infty)$$

In the following code, the above approximation is verified by computer simulation.

```
M = 1000
theta = 4                          # true value of theta
n_vals = c(5, 10, 30, 50, 100,500)
par(mfrow = c(2,3))
for(n in n_vals){
  xbar = numeric(length = M)
  g_xbar = numeric(length = M)
  for(i in 1:M){
    u = runif(n = n, min = 0, max = 1)
    x = (1-u)^(-1/theta) - 1           # probability integral transform
    xbar[i] = mean(x)
    g_xbar[i] = 1/mean(x) + 1
  }
  hist(xbar, probability = TRUE, main = paste("n = ", n),
       xlab = expression(bar(x[n])), breaks = 30)
  curve(dnorm(x, mean = 1/(theta-1), sd = sqrt(1/(n*(theta-1)*(theta-2)))),
        col = "red", lwd = 2, add = TRUE)
  hist(g_xbar, probability = TRUE, main = paste("n = ", n),
       xlab = expression(V[n]), breaks = 30)
```

```
    curve(dnorm(x, mean = theta, sd = sqrt((theta-1)^3/(n*(theta-2)))),
          col = "red", lwd = 2, add = TRUE)
}
```

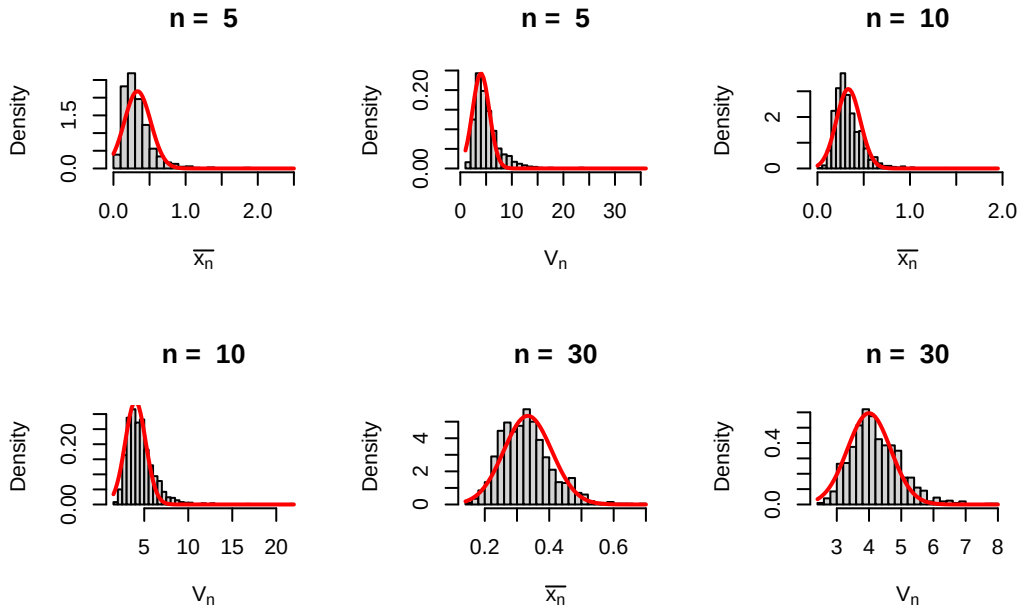


Figure 19: As the sample size increases, the sample mean $\overline{X_n}$ is approximately normally distributed with mean $\mu = \frac{1}{\theta-1}$ and variance $\frac{\sigma^2}{n} = \frac{1}{n(\theta-1)(\theta-2)}$. Using the first order Taylor's approximation the sampling distribution of V_n is obtained by computer simulation. The sampling distribution is well approximated by the normal distribution for large n. The approximate mean and variance of V_n is provided in the main text.

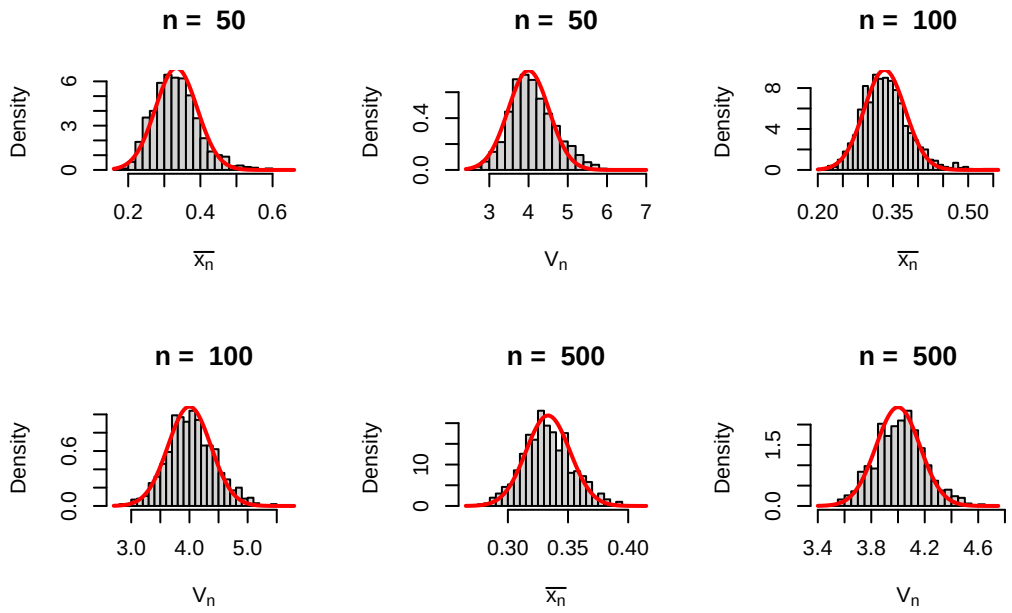


Figure 20: As the sample size increases, the sample mean $\overline{X_n}$ is approximately normally distributed with mean $\mu = \frac{1}{\theta-1}$ and variance $\frac{\sigma^2}{n} = \frac{1}{n(\theta-1)(\theta-2)}$. Using the first order Taylor's approximation the sampling distribution of V_n is obtained by computer simulation. The sampling distribution is well approximated by the normal distribution for large n. The approximate mean and variance of V_n is provided in the main text.

```
n_vals = c(5, 10, 25, 50, 100, 500)
theta_vals = seq(3, 20, by = 0.5)
par(mfrow = c(2,3))
M = 500             # number of replications
for(n in n_vals){
  risk_vals = numeric(length = length(theta_vals))
  for(i in 1:length(theta_vals)){
    theta = theta_vals[i]
    loss_vals = numeric(length = M)
    for(j in 1:M){
      u = runif(n = n, min = 0, max = 1)
      x = (1-u)^(-1/theta) - 1
      V_n = 1/mean(x)+1
      loss_vals[j] = (V_n - theta)^2
    }
    risk_vals[i] = mean(loss_vals)
  }
  plot(theta_vals, risk_vals, col = "red", pch = 19,
```

```
       xlab = expression(theta), ylab = expression(R(V[n],theta)),
       main = paste("n = ", n))
  curve((x-1)^3/(n*(x-2)), add = TRUE, col = "blue", lty =2,
        lwd = 2)
}
```

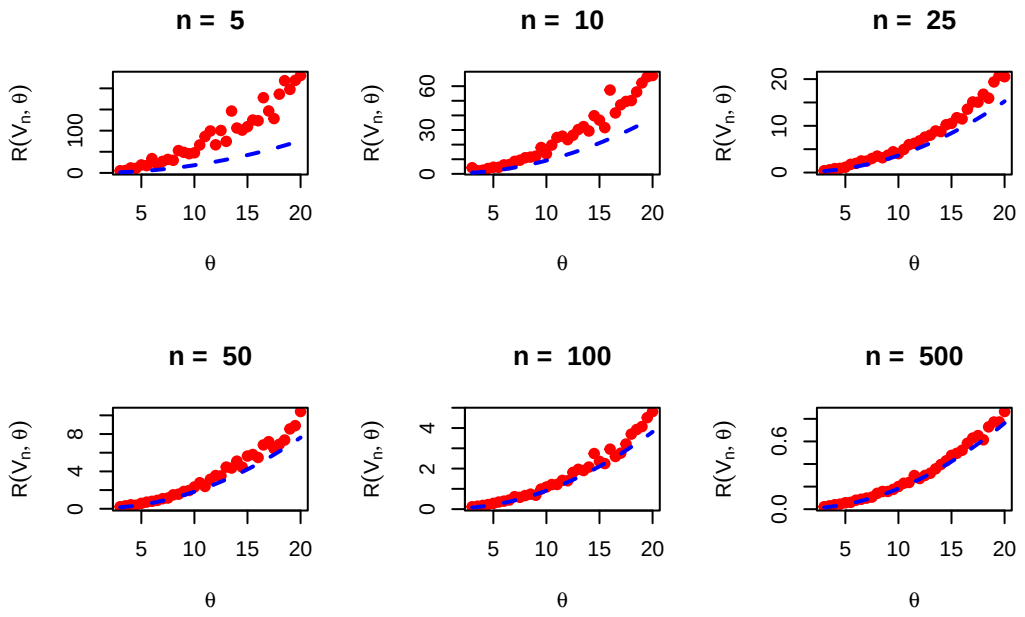


Figure 21: The approximation of the risk function of V_n by the first order Taylor's approximation. As $n \to \infty$, the approximations are accurate.

Comparing estimators V_n and W_n

```
n = 50
theta_vals = seq(0.1,5, by = 0.05)
M = 10000
risk_Vn = numeric(length = length(theta_vals))    # method of moments
risk_Wn = numeric(length = length(theta_vals))    # method of maximum likelihood
for (i in 1:length(theta_vals)) {
  theta = theta_vals[i]
  loss_Vn = numeric(length = M)
  loss_Wn = numeric(length = M)
  for(j in 1:M){
    u = runif(n = n, min = 0, max = 1)
```

```r
    x = (1-u)^(-1/theta) - 1
    V_n = 1/mean(x) + 1
    W_n = n/sum(log(1+x))
    loss_Vn[j] = (V_n - theta)^2
    loss_Wn[j] = (W_n - theta)^2
  }
  risk_Vn[i] = mean(loss_Vn)
  risk_Wn[i] = mean(loss_Wn)
}
plot(theta_vals, risk_Wn, col = "red", lwd = 2, type = "l",
     xlab = expression(theta), ylab = expression(R(., theta)),
     main = paste("n = ", n))
lines(theta_vals, risk_Vn, col = "blue", lwd = 2)
legend("bottomright", legend = c(expression(W[n]), expression(V[n])), col = c("red", "blue")
       lwd = c(2,2), bty = "n")
```

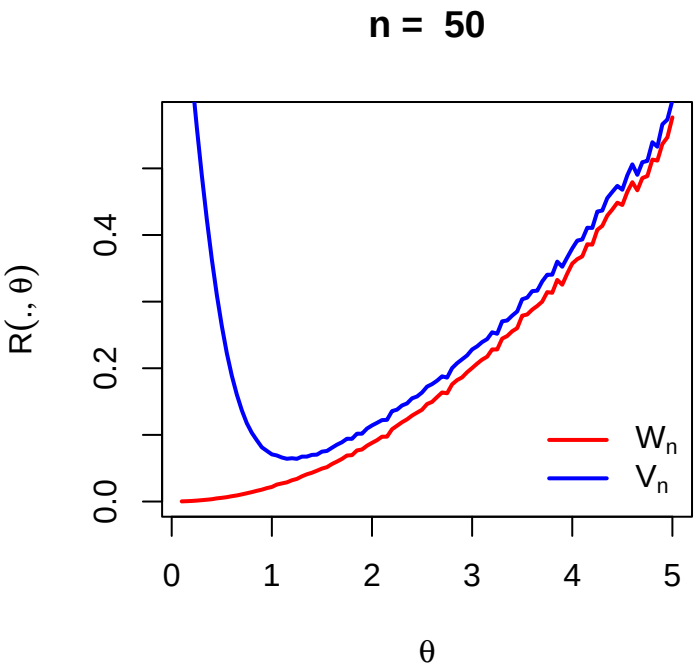


Figure 22: Comparison of the risk function of the method of moments and method of maximum likelihood. It can be observed that the MLE has uniformly smaller risk than the MoM estimator. Therefore, W_n is a preferred estimator than V_n. Similar exercise can be carried out for Y_n as well and we can find that MLE has outperformed both the estimator. A natural question arises, is MLE the best among all estimators of θ. This will be answered in the next chapter.

Unbiased Estimation and Search for the Best Estimator

Search for the Best Unbiased Estimator

Till now we have learnt the following: We can compute the risk function of an estimator under different loss functions. If the computation is not possible to execute analytically, we know how to obtain the risk function by computer simulation. We have seen several examples, but we should keep in mind that always the simulation from the population distribution may not be an easy task. This idea will be explained later.

We can compute the maximum likelihood estimator of the parameters both analytically and numerically.

Still we are not in a position to decide whether we have obtained the best estimator which has the uniformly minimum risk against all estimators of the parameter. This is definitely a difficult task and we can not even imagine the universe of all possible estimators to get hold of one best estimator. In addition, even if we get hold of the best estimator, we are not really sure whether this is the unique one or not. Let us try to address these questions in this document.

Concentrating on Unbiased Estimators

To obtain the best estimator, we reduce our universe of all possible estimators to the set of all unbiased estimators. Suppose that we have a random sample of size n from the population PDF (PMF) $f(x;\theta), \theta \in \Theta \subseteq \mathbb{R}$. An obvious consequence of having the class of unbiased estimators is that if $\psi(X_1, ..., X_n)$ be an unbiased estimator of θ, then $\mathrm{E}_\theta(\psi) = \theta$ for all $\theta \in \Theta$, which implies that under the squared error loss

$$\mathrm{E}_\theta(\psi_n - \theta)^2 = \mathrm{E}_\theta(\psi - \mathrm{E}_\theta(\psi_n))^2 = Var_\theta(\psi_n).$$

We keep the subscript n to emphasize the important role of sample size in studying the properties of the estimator. Therefore, for two unbiased estimators ψ'_n and ψ''_n, ψ'_n will be a better estimator of θ if

$$Var_\theta(\psi'_n) \leq Var_\theta(\psi''_n), \text{ for all } \theta.$$

Uniformly Minimum Variance Unbiased Estimator

Let $X_1, X_2, \ldots, X_n$ be a random sample of size n from the population PDF (PMF) $f(x|\theta)$. Suppose we are interested in estimating $g(\theta)$, a function of θ. An estimator $\psi_n^*(X_1, \ldots, X_n)$ of $g(\theta)$ is defined to be a uniformly minimum variance unbiased estimator of $g(\theta)$ if $\mathrm{E}_\theta(\psi_n^*) = g(\theta)$ for all $\theta \in \Theta$, that is, ψ_n^* is unbiased and $Var_\theta(\psi_n^*) \leq Var_\theta(\psi_n)$ for any other estimator $\psi_n(X_1, \ldots, X_n)$ of $g(\theta)$ which satisfies $\mathrm{E}_\theta(\psi_n) = g(\theta)$ for all $\theta \in \Theta$.

Search for the minimum bound

In general we may not be always interested in estimating the parameter θ, rather we may be interested in estimating some function of the parameter $g(\theta)$, say. For example, suppose a random sample of size n is available from the exponential distribution with rate parameter λ and we are interested in estimator $\mathrm{P}(X > 1) = e^{-\lambda} = g(\lambda)$. The next discussion is generalized for estimating some function of the parameter $g(\theta)$.

In search of the best estimator for $g(\theta)$, some function of the parameter θ, we first try to find the least possible risk attained by any unbiased estimator of $g(\theta)$ under squared error loss function. In other words, we want to compute a lower bound (sharp) for the variance of any unbiased estimator ψ of $g(\theta)$.

Cramer-Rao Inequality

Suppose that $X_1, X_2, \ldots, X_n$ be a random sample (not necessarily IID) of size n from the population PDF (PMF) $f(x|\theta)$ and let $\psi(\mathbf{X})$ be any estimator satisfying

$$\frac{d}{d\theta}\mathrm{E}_\theta(\psi(\mathbf{X})) = \int_\chi \frac{\partial}{\partial\theta}[\psi(\mathbf{X})f(\mathbf{x}|\theta)]d\mathbf{x}$$

and

$$Var_\theta(\psi(\mathbf{X})) < \infty,$$

then

$$Var_\theta(\psi_n(\mathbf{X})) \geq \frac{\left(\frac{d}{d\theta}\mathrm{E}_\theta(\psi(\mathbf{X}))\right)^2}{\mathrm{E}_\theta\left(\left(\frac{\partial}{\partial\theta}\log f(\mathbf{X}|\theta)\right)^2\right)}.$$

IID Case

If the assumptions of the Cramer-Rao inequality are satisfied and, additionally, if $X_1, X_2, \ldots, X_n$ are IID with PDF $f(x|\theta)$, then

$$Var_\theta \psi(\mathbf{X}) \geq \frac{\left(\frac{d}{d\theta}\mathrm{E}_\theta(\psi(\mathbf{X}))\right)^2}{n\mathrm{E}_\theta\left(\left(\frac{\partial}{\partial\theta}\log f(X|\theta)\right)^2\right)}$$

Consider estimating the mean parameter β from the exponential PDF given by $f(x) = \frac{1}{\beta}e^{-\frac{x}{\beta}}, 0 < x < \infty$ and $\beta \in (0, \infty)$. In this case $g(\beta) = \beta$. Consider the MLE $\overline{X_n}$ which is an unbiased estimator of β and $Var_\beta\left(\overline{X_n}\right) = \frac{\beta^2}{n}$. Let us compute the CR bound for any unbiased estimator of β. The denominator in the expression for C-R bound is computed below:

$$\begin{aligned} n\mathrm{E}_\beta\left(\left(\frac{\partial}{\partial\beta}\log f(X|\beta)\right)^2\right) &= n\mathrm{E}_\beta\left(\frac{\partial}{\partial\beta}\left(-\log\beta - \frac{X}{\beta}\right)\right)^2 && (0.1)\\ &= n\mathrm{E}_\beta\left(-\frac{1}{\beta} + \frac{X}{\beta^2}\right)^2 = \frac{n}{\beta^4}\mathrm{E}_\beta\left(X-\beta\right)^2 && (0.2)\\ &= \frac{n\beta^2}{\beta^4} = \frac{n}{\beta^2}. && (0.3) \end{aligned}$$

Therefore, the C-R bound is $\frac{\beta^2}{n}$ and it is equal to $Var_\beta\left(\overline{X_n}\right)$. Therefore, $\overline{X_n}$ must be the UMVUE.

When C-R bound does not hold

Suppose that we have a random sample of size n from the following PDF

$$f(x|\theta) = \begin{cases} \frac{2x}{\theta^2}, 0 < x < \theta \\ 0, \quad \text{otherwise.} \end{cases}$$

- Show that the MLE of θ is $Y_n = \max(X_1, \ldots, X_n)$.
- The MLE is not unbiased as $\mathrm{E}_\theta(Y_n) = \frac{2n\theta}{2n+1} \neq \theta$ for all $\theta \in (0, \infty)$. However,

$$\mathrm{Bias}_\theta(Y_n) = \mathrm{E}_\theta(Y_n) - \theta = -\frac{\theta}{2n+1} \to 0.$$

  as $n \to \infty$. Therefore, MLE is asymptotically unbiased.
- However, we can obtain an unbiased estimator which is a function of Y_n as $\psi_n = \frac{2n+1}{2n}Y_n$, then

$$\mathrm{E}_\theta\left(\psi_n\right) = \frac{2n+1}{2n} \times \frac{2n}{2n+1}\theta = \theta.$$

- The variance of ψ_n is obtained as follows:
  - $\mathrm{E}_\theta\left(Y_n^2\right) = \frac{2n}{2n+2}\theta^2$.
  - $Var_\theta\left(Y_n\right) = \mathrm{E}_\theta\left(Y_n^2\right) - \left(\mathrm{E}_\theta(Y_n)\right)^2 = \frac{n\theta^2}{(n+1)(2n+1)^2}$.
  - $Var_\theta\left(\psi_n\right) = \left(\frac{2n+1}{2n}\right)^2 Var_\theta\left(Y_n\right) = \frac{\theta^2}{4n(n+1)}$.
- Compute the C-R bound to verify that it is equal to $\frac{\theta^2}{4n}$.

In the above problem, it is surprising to see that

$$Var_\theta(\psi_n) = \frac{\theta^2}{4n(n+1)} < \frac{\theta^2}{4n} = \text{C-R bound.}$$

- Check that the required condition which allows the interchange of the derivative with respect to parameter θ and the expectation does not satisfy for the given PDF.
- The result can be generalized further to observe that the support of the distribution depends on the unknown parameter θ, which is $(0, \theta)$. For such distributions, the criteria involving C-R lower bound can not be utilized to identify the best unbiased estimator. In fact, when the support depends on the parameter, C-R bound theorem does not work.
- Most of the book contains the Uniform$(0, \theta)$ example to demonstrate the above exception, but, here we have a different example and you are encouraged to execute the above task. In line with this concept, one can actually test this using computer simulation as well. We can simulate the risk function of ψ_n^* using simulation and check that it completely lies below the C-R bound. One can use the probability integral transform to simulate from the given PDF.

```
par(mfrow = c(1,3))
theta_vals = c(2,4)
n = 100
for(theta in theta_vals){
  u = runif(n = n, min = 0, max = 1)
  x = theta*sqrt(u)
  hist(x, probability = TRUE, main = bquote(theta == .(theta)), ylim = c(0, 2/theta+0.1), ce
  curve(2*x/(theta^2)*(0<x)*(x<theta), add = TRUE,
        col = "red", lwd = 2)
}
# simulation of risk function for n = 4
n = 4
theta_vals = seq(1,4, by = 0.1)
risk_vals = numeric(length = length(theta_vals))
M = 1000              # number of replications
for(i in 1:length(theta_vals)){
  theta = theta_vals[i]
```

```r
  psi_n = numeric(length = M)
  loss_vals = numeric(length = M)
  for(j in 1:M){
    u = runif(n = n, min = 0, max = 1)
    x = theta*sqrt(u)
    psi_n[j] = ((2*n+1)/(2*n))*max(x)
    loss_vals[j] = (psi_n[j] - theta)^2
  }
  risk_vals[i] = mean(loss_vals)
}
plot(theta_vals, risk_vals, col = "red", pch = 19,
     cex = 1.3, ylim = c(0, max(theta_vals)^2/(4*n)),
     xlab = expression(theta), ylab = expression(R(psi[n],theta)), main = paste("n = ", n))
curve(x^2/(4*n*(n+1)), col = "blue", lwd = 2, add = TRUE)
curve(x^2/(4*n), add = TRUE, col = "magenta",
      lwd = 2, lty = 2)
legend("topleft", legend = c("C-R bound"),
       lty = 2, col = "magenta", lwd = 2, bty = "n")
```

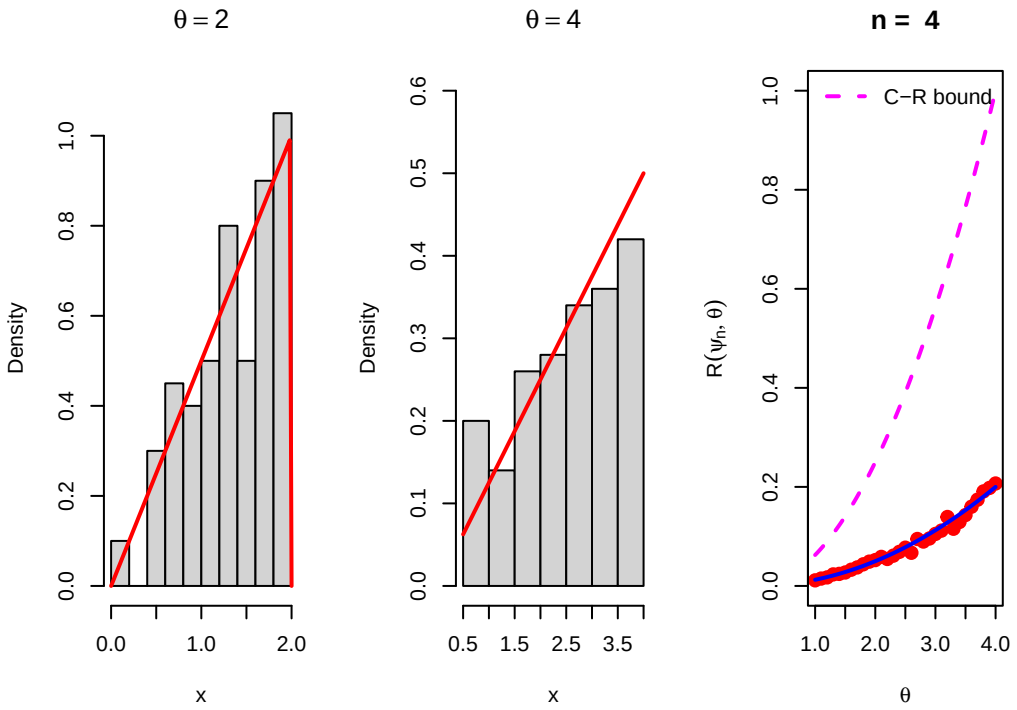


Figure 1: The left two panels demonstrate the simulation of random samples from the given probability density function using probability integral transform. A sample size of $n = 100$ has been simulated for two values of θ. The overlaid exact PDF confirms the effective use of the PIT for simulation. The right most panel shows the risk function of ψ_n which is obtained based on $M = 1000$ replicaions (red dots). The true risk function (blue color) is added for reference. The C-R bound is shown using the dotted mangeta colour. It is clear from the picture that the for the given PDF, characterization of the best estimator based on the C-R bound can not be established.

Therefore, we need to understand additional theories to characterize the UMVUE. We shall now introduce two important concepts, `Sufficient Statistics` and `Complete Statistics` which will help us address the gaps we encountered in characterizing the UMVUE.

Fisher Information Number

The following quantity which appears in the denominator of the lower bound for the variance of any unbiased estimator is known as the Fisher Information

$$\mathrm{E}_\theta\left(\left(\frac{\partial}{\partial\theta}\log f(\mathbf{X}|\theta)\right)^2\right).$$

This quantity tells us as the information number increases, we have more information about the parameter θ. An alternative computation for the Fisher Information can be eased by the following theorem.

! Computing Fisher Information

If $f(x|\theta)$ satisfies

$$\frac{d}{d\theta}\mathrm{E}_\theta\left(\frac{\partial}{\partial\theta}\log f\left(X|\theta\right)\right)=\int\frac{\partial}{\partial\theta}\left[\left(\frac{\partial}{\partial\theta}\log f(x|\theta)\right)f(x|\theta)\right]dx$$

, then

$$\mathrm{E}_\theta\left(\left(\frac{\partial}{\partial\theta}\log f(X|\theta)\right)^2\right)=-\mathrm{E}_\theta\left(\frac{\partial^2}{\partial\theta^2}\log f\left(X|\theta\right)\right)$$

! When the CR bound will be attained

Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed random variables following $f(x|\theta)$, and $f(x|\theta)$ satisfies the conditions of the Cramer-Rao theorem. Let $\mathcal{L}(\theta|\mathbf{x}) = \prod_{i=1}^{n} f(x_i|\theta)$ denote the likelihood function. If $W(\mathbf{X}) = W(X_1, \ldots, X_n)$ be any unbiased estimator of $g(\theta)$, then $W(\mathbf{X})$ attains the Cramer-Rao Lower Bound if and only if

$$a(\theta)[W(\mathbf{x}) - g(\theta)] = \frac{\partial}{\partial\theta}\log\mathcal{L}(\theta|\mathbf{x}),$$

for some function $a(\theta)$.

Sufficient Statistics

Let $X_1, X_2, \ldots, X_n$ be a random sample of size n from the PDF (PMF) $f(x|\theta)$, where θ may be a vector of parameters. A statistic $T(X_1, \ldots, X_n)$ is said to be a sufficient statistic if the conditional distribution of $(X_1, \ldots, X_n)$ given $T = t$ does not depend on θ for any value t of T.

! Sufficiency principle

If $T(\mathbf{X}) = T(X_1, \ldots, X_n)$ is a sufficient statistic for θ, then any inference about θ should depend on the sample $\mathbf{X}$ only through the value $T(\mathbf{X})$. That is, if $\mathbf{x}$ and $\mathbf{y}$ are two sample points such that $T(\mathbf{x}) = T(\mathbf{y})$, then the inference about θ should be the same whether $\mathbf{X} = \mathbf{x}$ of $\mathbf{Y} = \mathbf{y}$ is observed.

Exponential Family and minimal sufficient statistics

Exponential family of densities

A one-parameter family (the parameter $\theta \in \Theta \subseteq \mathbb{R}$) of PDF (PMF) $f(x|\theta)$ that can be expressed as

$$f(x|\theta) = a(\theta)b(x)e^{c(\theta)d(x)}$$

for all $x \in (-\infty, \infty)$ and for all $\theta \in \Theta$, and for a suitable choice of functions $a(\cdot)$, $b(\cdot)$, $c(\cdot)$ and $d(\cdot)$ is defined to belong to the exponential family or exponential class.

If we have a random sample of size n, from an exponential family, then the joint distribution of the data can be expressed as follow:

$$\prod_{i=1}^{n} f(x_i|\theta) = (a(\theta))^n \left[\prod_{i=1}^{n} b(x_i)\right] \exp\left[c(\theta)\sum_{i=1}^{n} d(x_i)\right].$$

Worked examples

- $\mathcal{G}(\alpha, \beta)$ family:

  $$f(x|\alpha,\beta) = \frac{x^{\alpha-1}e^{-\frac{x}{\beta}}}{\Gamma(\alpha)\beta^{\alpha}} = a(\alpha,\beta)b(x)\exp\left[(\alpha-1)\log(x) + \left(-\frac{1}{\beta}\right)x\right],$$

  where $a(\alpha,\beta) = \Gamma(\alpha)\beta^{\alpha}$, $b(x) = 1$, $c_1(\alpha,\beta) = \alpha - 1$, $c_2(\alpha,\beta) - \frac{1}{\beta}$, $d_1(x) = \log(x)$ and $d_2(x) = x$. Therefore, $\left(\sum_{i=1}^{n} X_i, \sum_{i=1}^{n}\log(X_i)\right)$ is jointly minimal sufficient for (α, β).

- Show that the following PMFs belong to the exponential family and identify the corresponding minimal sufficient statistics

  - binomial(n, θ), n known.
  - Poisson(λ)
  - Geometric(p)
  - Discrete Uniform$\{1, 2, \ldots, \theta\}$, $\theta \in \{1, 2, 3, \ldots\}$.

- Identify which of the following PDFs belong to the exponential family and identify the corresponding minimal sufficient statistics

  - $\mathcal{N}(\mu, \sigma^2)$
  - $\mathcal{B}(a, b)$
  - $\mathcal{G}(a, b)$
  - Cauchy(μ, σ)
  - Uniform$(0, \theta)$

- According to the factorization criteria, $\sum_{i=1}^{n} d\left(X_i\right)$ is a sufficient statistic for θ.

- However, the above statement can be made much stronger in a sense that the above sufficient statistic $\sum_{i=1}^{n} d(X_i)$ is in fact minimal sufficient.

k-parameter exponential family

A family of PDF (PMF) $f(x|\theta_1, \dots, \theta_k)$ that can be expressed as

$$f(x|\theta_1, \dots, \theta_k) = a(\theta_1, \dots, \theta_k)b(x) \exp\left[\sum_{j=1}^{n} c_j(\theta_1, \dots, \theta_k)d_j(x)\right]$$

for suitable choice of functions $a(\cdot, \dots, \cdot)$, $b(\cdot)$, $c_j(\cdot, \dots, \cdot)$ and $d_j(\cdot)$, $j = 1, 2, \dots, k$ is defined to belong to the exponential family.

! Examples of distribution not belonging to the exponential family

- The family of Uniform$(0, \theta)$ does not belong to the exponential family.
- The statement cane is true in general. That is, any family of densities for which the range of the values were the density is non-negative depends of the parameter θ does not belong the the exponential class.

Sufficiency and Unbiasedness

! Rao-Blackwell

Let W be any unbiased estimator of $g(\theta)$, and T be a sufficient statistic for θ. Define $\phi(T) = \mathrm{E}(W|T)$. Then,

- $\mathrm{E}_\theta \phi(T) = g(\theta)$.
- $Var_\theta \phi(T) \leq Var_\theta W$ for all θ. The above two conditions imply that $\phi(T)$ is uniformly better unbiased estimator of $g(\theta)$.

The Rao-Blackwell theorem states a very important fact that conditioning any unbiased estimator on a sufficient statistic will result in a uniform improvement, so we need to consider only statistics that are functions of a sufficient statistic in our search for the best estimators.

! Uniqueness of the best estimator

If W is a best unbiased estimator of $g(\theta)$, then W is unique.

Complete Statistics

! Complete family

Let $f(t|\theta)$ be a family of PDFs or PMFs for a statistic $T(\mathbf{X})$. The family of probability distributions is called complete if $\mathrm{E}_\theta g(T) = 0$ for all θ implies $\mathrm{P}(g(X) = 0) = 1$ for all θ. Equivalently, $T(\mathbf{X})$ is a complete statistic.

Examples of Complete family

- Let $(X_1, X_2, \dots, X_n)$ be iid Uniform$(0, \theta)$ observations, $0 < \theta < \infty$. Then $T(\mathbf{X}) = \max(X_1, \dots, X_n) = X_{(n)}$ is a complete statistic.
- Let $X_1, \dots, X_n$ be a random sample from a population with PDF

  $$f(x|\theta) = \theta x^{\theta-1}, 0 < x < 1, 0 < \theta < \infty.$$

  Find a complete sufficient statistics for θ.
- Let $X_1, X_2, \dots, X_n$ be IID with geometric distribution

  $$\mathrm{P}_\theta(X = x) = \theta(1-\theta)^{x-1}. x = 1, 2, \dots, \ 0 < \theta < 1.$$

  Show that $\sum_{i=1}^n X_i$ is sufficient for θ, and find the family of distributions of $\sum X_i$. Check whether the family is complete.

Complete statistics in the exponential family

Let $X_1, X_2, \dots, X_n$ be iid observations from an exponential family with PDF or PMF of the form

$$f(x|\theta) = a(\theta)b(x)\exp\left(\sum_{j=1}^k c_j(\theta)d_j(x)\right),$$

where $\theta = (\theta_1, \dots, \theta_k)$. Then the statistic

$$T(\mathbf{X}) = \left(\sum_{i=1}^n d_1(X_i), \sum_{i=1}^n d_2(X_i), \dots, \sum_{i=1}^n d_k(X_i)\right)$$

is complete if $\{(c_1(\theta), c_2(\theta), \dots, c_k(\theta)) \colon \theta \in \Theta\}$ contains an open set in $\mathbb{R}^k$.

- If $X_1, \dots, X_n \sim \mathcal{N}(\theta, \theta^2)$. Here the parameter space (θ, θ^2) does not contain a two-dimensional open set, as it consists of only the points on the parabola.

The following theorem delineates the connection between complete sufficiency of an unbiased estimator with the best unbiased estimator.

Best unbiased estimator

> **!** Important
>
> Let T be a complete sufficient statistic for the parameter θ, and let $\phi(T)$ be any estimator based only on T. Then $\phi(T)$ is the unique best unbiased estimator of its expected value.

Let us see an illustrative example. Suppose $X_1, X_2, \ldots, X_n$ be IID $f(x|\theta)$, where

$$f(x|\theta) = \theta x^{\theta-1} I_{(0,1)}(x), 0 < \theta < \infty.$$

- First we observe that this PDF belongs to the exponential family as

$$f(x|\theta) = \theta \times 1 \times \exp\left[(\theta - 1)\log x\right] = a(\theta)b(x)\exp[c(\theta)d(x)]$$

- Therefore, $\sum_{i=1}^{n} \log X_i$ is complete sufficient (in fact minimal) statistic for θ.
- Compute the MLE of θ: the likelihood function $\mathcal{L}(\theta) = \theta^n \left(\prod_{i=1}^{n} x_i^{\theta-1}\right)$ and

$$l(\theta) = \log \mathcal{L}(\theta) = n \log \theta + (\theta - 1) \sum_{i=1}^{n} \log X_i.$$

  Solving $l'(\theta) = 0$, we obtain the MLE of θ as

$$\widehat{\theta_n} = T_n = -\frac{n}{\sum_{i=1}^{n} \log X_i}.$$

- First let us check whether $\widehat{\theta_n}$ is unbiased. We must note that the MLE is in fact a function T_n, which is a complete sufficient statistic $\sum_{i=1}^{n} \log X_i$. We obtain the sampling distribution of $\widehat{\theta_n}$.
  - (Step - I) Show that $Y_i = -\log X_i \sim$ Exponential(rate $= \theta$).
  - (Step - II) Show that $Y = \sum_{i=1}^{n} Y_i = -\sum_{i=1}^{n} \log X_i \sim \mathcal{G}\left(\alpha = n, \beta = \frac{1}{\theta}\right)$.
  - (Step - IV) Show that the PDF of $\widehat{\theta_n}$ is given by $f_{T_n}(z) = \frac{(n\theta)^n e^{-\frac{n\theta}{z}}}{\Gamma(n) z^{n+1}}, 0 < z < \infty$.
  - (Step - V) Show that $\mathrm{E}_\theta(T_n) = \frac{n\theta}{n-1}, n > 1$. Therefore, T_n is not an unbiased estimator of θ.
  - (Step - VI) We can get an unbiased estimator which is a function of T_n, which is a function of a complete sufficient statistic as $T_n^* = \frac{n-1}{n} T_n = -\frac{n-1}{\sum_{i=1}^{n} \log X_i}$. Therefore, T_n^* is the best unbiased estimator.
  - (Step - VII) Compute the variance of T_n^*
  - (Step - VIII) Compute the C-R bound for any unbiased estimator of θ and check whether $Var_\theta(T_n^*) = I_n(\theta)^{-1}$ for all $\theta \in (0, \infty)$.

Connection with the MLE

Method of Maximum Likelihood

A motivating example

We have a random sample $(X_1, X_2, \dots, X_n)$ of size n from the Geometric(p) distribution, whose probability mass function is given by

$$\mathrm{P}(X = x) = (1-p)^{x-1}p,\ x \in \{1, 2, ...\}.$$

The random variable X represents the number of throws required to obtain the first success if we continue tossing a coin with probability of head p until the first head is observed.

Suppose that a company produces a large number of identical coins. We are interested in estimating the probability of head. The following experimental procedure is planned to be followed to estimate the true probability of head. Out of a large number of coins n coins have been chosen and they are numbered as $\{1, 2, \dots, n\}$. For each $i \in \{1, 2, \dots, n\}$, ith coin has been kept on tossing till the first head appears. X_i denotes the number of throws required to observe the first head for the ith coin.

Suppose that $n = 5$ and the above experiment gave the following observations:

```
[1] 4 6 1 1 2
```

Let us compute the likelihood of observing the above sample as a function of p as follows:

$$\mathrm{P}\left(X_1 = 4, X_2 = 6, X_3 = 1, X_4 = 1, X_5 = 2\right) = (1-p)^9 p^5.$$

The sample space of $(X_1, \dots, X_5)$ is given by the following set:

$$\{1, 2, 3, ...\}^5 = \{(x_1, x \dots, x_5) \colon x_i \in \{1, 2, 3, ...\}, 1 \leq i \leq 5\}.$$

Each of the points in this sample space has a positive probability of being included in the random sample of size 5. However, since the given sample is observed, is not unreasonable to consider that it has significantly larger likelihood. In other words, we ask the question, for what value of $p \in (0, 1)$, the probability of observing the given sample is as large as possible? To answer this, we can express the likelihood of given sample as a function of p, which is here given by

$$\mathcal{L}(p) = (1-p)^9 \times p^5,\ p \in (0, 1).$$

We want to identify that for what value of p, the likelihood of the observed sample is maximum, that is p^*, so that $\mathcal{L}(p^*) \geq \mathcal{L}(p)$ for all $p \in (0, 1)$. It boils down to a maximization problem. Instead of maximizing $\mathcal{L}(\theta)$, we can maximize $\log \mathcal{L}(\theta)$, which are referred as the likelihood and the log-likelihood function, respectively.

$$l(p) = \log \mathcal{L}(p) = 9 \log(1-p) + 5 \log p.$$

$$\frac{d}{dp} (\log \mathcal{L}(p)) = l'(p) = -\frac{9}{1-p} + \frac{5}{p}.$$

Equating $l'(p) = 0$, we get $p^* = \frac{5}{14}$ and it is easy to verify that $l''(p^*) < 0$.

```
par(mfrow = c(1,2))
p_star = 5/14
curve((1-x)^9*x^5, 0.01, 0.99, col = "red", lwd = 2,
      xlab = "p", ylab = "L(p)")
points(p_star, 0, pch = 19, col = "blue", cex = 1.4)
abline(v = p_star, col = "grey", lty = 2, lwd = 3)
curve(9*log(1-x) + 5*log(x), col = "red", lwd  = 2,
      xlab = "p", ylab = "l(p)")
points(p_star, 0, pch = 19, col = "blue", cex = 1.4)
abline(v = p_star, col = "grey", lty = 2, lwd = 3)
```

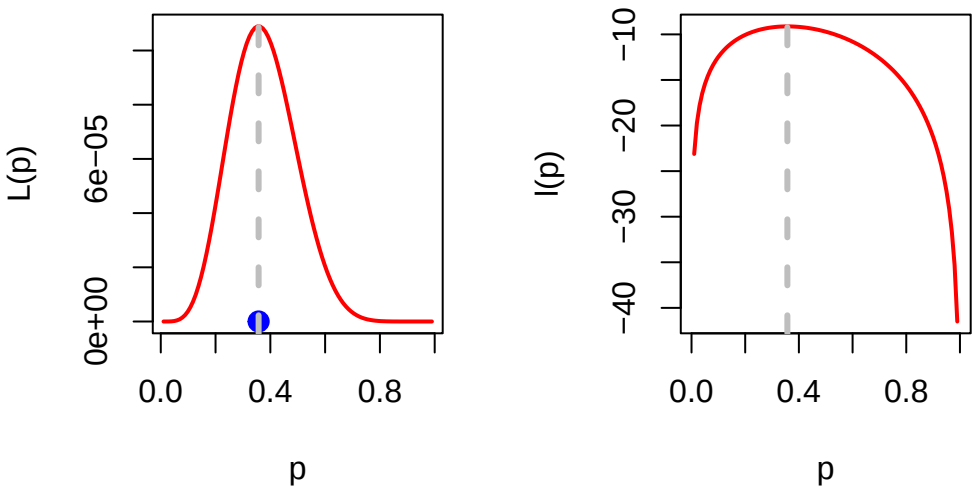


Figure 1: The maximum likliehood estimate of p is $\frac{5}{14}$. The left panel represents the likelihood function $\mathcal{L}(p)$ and the right panel depicts the log-likelihood function $\log \mathcal{L}(p)$. The vertical corresponds to the $p^* = \frac{5}{14}$, at which the function is maximum.

Based on the above data set, we obtained the estimate of the probability as $\frac{5}{14}$. It is intuitively clear that this estimate is subject to uncertainty. By this statement, I essentially mean that if

the this experiment would have been carried out by ten different individuals, then we would have obtained ten different samples of size 5. Certainly, the value of p^* is not expected to be the same for all the samples. Therefore, we are interested in computing the risk associated with this estimate. To formalize it further, let us try to obtain some explicit expression for p^*.

Suppose that $x_1, x_2, \dots, x_n$ are realizations of the random sample $(X_1, X_2, \dots, X_n)$. Then the likelihood function is given by

$$\mathcal{L}(p) = \prod_{i=1}^{n} \mathrm{P}(X_i = x_i) = (1-p)^{\sum x_i - n} p^n, 0 < p < 1,$$

and the log-likelihood function is given by

$$l(p) = \left(\sum x_i - n\right) \log(1-p) + n \log p.$$

$l'(p) = 0$ gives $p^* = \frac{n}{\sum x_i} = (\overline{x_n})^{-1}$. Therefore the maximum likelihood estimator of p is given by

$$\widehat{p_n} = \frac{1}{\overline{X_n}}.$$

Our goal is to estimate the risk function

$$\mathcal{R}(\widehat{p_n}, p) = \mathrm{E}\left(\frac{1}{\overline{X_n}} - p\right)^2, \quad p \in (0,1).$$

! The likelihood and the log-likelihood

If $X_1, X_2, \dots, X_n$ be a random sample of size n from the population with PDF (PMF) $f(x|\theta)$, then the likelihood function $\mathcal{L}_n \colon \Theta \to [0, \infty)$ defined as

$$\mathcal{L}_n(\theta) = \prod_{i=1}^{n} f\left(X_i|\theta\right), \theta \in \Theta.$$

It is important to note that the likelihood function is the joint PDF (PMF), but considered as a function of θ, and certainly the statement $\int_\Theta \mathcal{L}_n(\theta) d\theta = 1$ not necessarily be true. The log-likelihood function is defined by

$$l_n(\theta) = \log \mathcal{L}_n(\theta), \ \theta \in \Theta.$$

Is the estimator consistent!

A desirable property of an estimator is that as the sample size increases, the sampling distribution of the estimator should be more concentrated about the true parameter value. We

should not forget that an estimator is a random quantity which is subject to sampling variation. In this case, before going to some mathematical computation, we first check whether this is indeed happening or not for $\widehat{p_n}$, the MLE of p. We employed the following scheme to understand the behavior of $\widehat{p_n}$ as $n \to \infty$:

- Fix $p_0 \in (0, 1)$
- Fix sample size, $n \in \{1, 2, 3, \dots, 1000\}$.
- For each $n \in \{1, 2, 3, \dots, 1000\}$
  - Simulate $X_1, \dots, X_n \sim \text{Geometric}(p_0)$.
  - Compute $\overline{X_n}$
  - Compute $\widehat{p_n} = \overline{X_n}^{-1}$.
- Plot the pairs $(n, \widehat{p_n}), n \in \{1, 2, \dots, 1000\}$.
- Do the above experiment for different choices of $p_0 \in (0, 1)$.

```r
p = 0.4
n_vals = 1:1000
pn_hat = numeric(length = length(n_vals))
for(n in n_vals){
  x = rgeom(n = n, prob = p)+1
  pn_hat[n] = 1/mean(x)
}
plot(n_vals, pn_hat, type = "l", col = "grey", lwd  = 2,
     xlab = "sample size (n)", ylab = expression(widehat(p[n])))
abline(h = 0.4, col = "blue", lwd = 3, lty = 2)
```

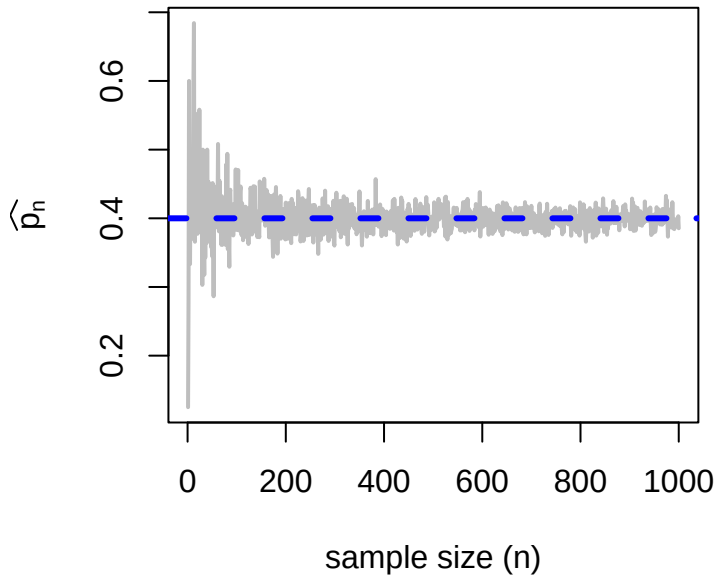


Figure 2: As the sample size increases, the MLE $\overline{p_n}$ converges to the true value of p. The reader is encouraged to perform the simulation experiment for various choices of $p \in (0, 1)$. The horizontal blue dotted line indicates the true probability p.

The statement can also be established theoretically. From the Weak Law of Large Numbers, we know that $\overline{X_n} \to \mathrm{E}(X) = \frac{1}{p}$ in probability as $n \to \infty$. If we choose $g(x) = \frac{1}{x}$, which is a continuous function, then

$$g\left(\overline{X_n}\right) = \frac{1}{\overline{X_n}} = \widehat{p_n} \xrightarrow{P} p.$$

! Continuous function and convergence in probability

If $X_n \xrightarrow{P} X$, then $g(X_n) \xrightarrow{P} g(X)$, where $g(\cdot)$ is a continuous function.

Simulating the risk function of the MLE of p

The problem is to compute the risk function for different choices of $p \in (0, 1)$,

$$\mathcal{R}\left(\overline{X_n}^{-1}, p\right), 0 < p < 1.$$

The following steps have been performed using R Programming to approximate the risk function of $\overline{p_n}$ under the squared error loss function, which is also referred to as the Mean Squared Error (MSE) of $\widehat{p_n}$.

- Discretize $(0, 1)$ as $(p_1, \dots, p_L)$.
- Fix sample size n
- Fix M, the number of replications
- For each $p \in \{p_1, \dots, p_L\}$
  - For each $m \in \{1, 2, \dots, M\}$
    - Simulate $X_1, X_2, \dots, X_n \sim \text{Geometric}(p)$.
    - Compute loss $l_m = \left(\frac{1}{\overline{X_n}} - p\right)^2$
  - Compute the risk $\mathcal{R}\left(\frac{1}{\overline{X_n}}, p\right) = \frac{1}{M}\sum_{m=1}^{M} l_m$.
- Plot the pairs of values $\left\{p, \mathcal{R}\left(\frac{1}{\overline{X_n}}, p\right)\right\}, p \in \{p_1, \dots, p_L\}$.
- Repeat the above exercise for different choices of n

```r
par(mfrow = c(2,3))
n_vals = c(5, 10, 30, 50, 100, 500)                  # different sample sizes
M = 5000                                             # number of replications
prob_vals = seq(0.01, 0.9, by = 0.01)
for(n in n_vals){
  risk_pn_hat = numeric(length = length(prob_vals))
  for(i in 1:length(prob_vals)){
    p = prob_vals[i]
    loss_pn_hat = numeric(length = M)
    for(j in 1:M){
      x = rgeom(n = n, prob = p) + 1
      loss_pn_hat[j] = (1/mean(x)-p)^2               # compute loss
    }
    risk_pn_hat[i] = mean(loss_pn_hat)               # compute risk (average loss)
  }
  plot(prob_vals, risk_pn_hat, col = "red", xlab = "p",
       ylab = expression(R(widehat(p[n]),p)),
       main = paste("n = ", n))
  lines(prob_vals, prob_vals^2*(1-prob_vals)/n,
        col = "blue", lwd = 3, lty = 2)              # add true risk function
}
```

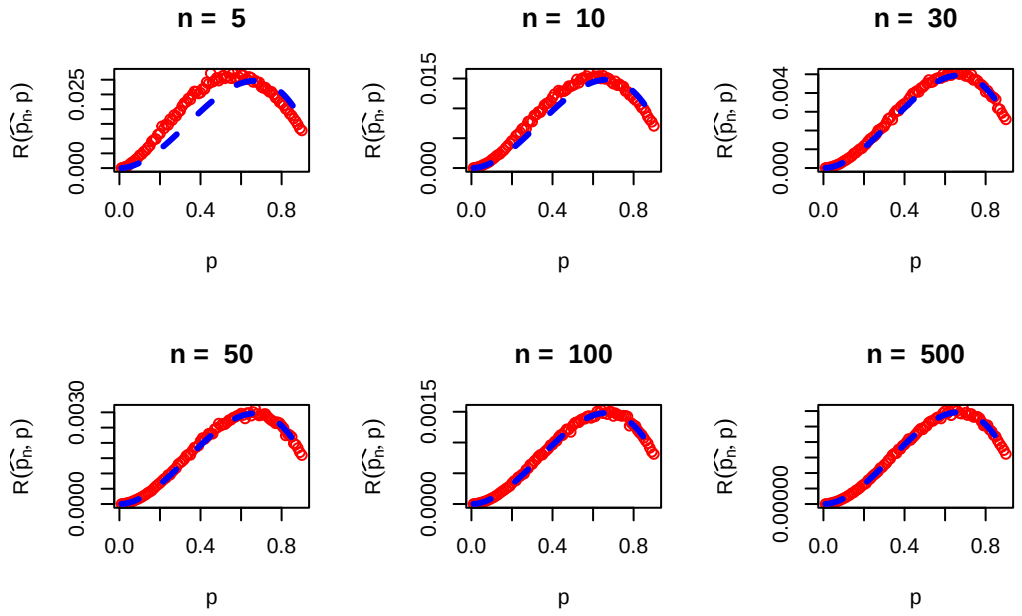


Figure 3: The approximation of the risk function of the MLE of p based on random sample of size n from the Geometric(p) distribution. The evaluation is carried out based on $M = 1000$ replications under the squared error loss function.

For this problem it is almost impossible to compute the risk function analytically. Let us try to obtain an approximation of the risk function which performs well for large n. Consider $g(x) = \frac{1}{x}$. Expanding the Taylor's Polynomial of $g(\overline{X_n})$ about $\frac{1}{p}$ gives

$$\begin{aligned} g\left(\overline{X_n}\right) &\approx g\left(\frac{1}{p}\right) + \left(\overline{X_n} - \frac{1}{p}\right) g'\left(\frac{1}{p}\right) \\ &\approx p + \left(\overline{X_n} - \frac{1}{p}\right)(-p^2). \end{aligned} \tag{0.1}$$

From the Central Limit Theorem we have

$$\overline{X_n} \overset{a}{\sim} \mathcal{N}\left(\frac{1}{p}, \frac{1-p}{p^2 n}\right),$$

therefore, for large n, $Var\left(\overline{X_n}\right) \approx \frac{1-p}{p^2 n}$. Taking expectation on both sides of the Taylor's polynomial we see that

$$\mathrm{E}\left[g\left(\overline{X_n}\right)\right] \approx p$$

and the approximate risk is given by

$$\mathcal{R}\left(\frac{1}{\overline{X_n}}, p\right) = \mathrm{E}\left(\frac{1}{\overline{X_n}} - p\right)^2 \approx \mathrm{E}\left(\overline{X_n} - \frac{1}{p}\right)^2 (-p^2)^2 \approx Var\left(\overline{X_n}\right) p^4 = \frac{(1-p)p^2}{n}.$$

Therefore

$$\mathcal{R}\left(\frac{1}{\overline{X_n}}, p\right) \approx \frac{(1-p)p^2}{n}, \quad \text{for large } n.$$

In the Fig, the simulated risk function is overlaid with the approximate risk function obtained the Taylor's approximation (first order). It can be noted that for large n, the approximations are remarkable close to the true risk function.

It is interesting to observe that the risk is different at different values of the parameter p. Therefore, if the estimate of the parameter is p^*, then the estimated risk will be $\frac{(1-p^*)p^{*2}}{n}$, which is approximately equal to $\frac{0.082}{n}$ for $p^* = \frac{5}{14}$ and $n = 5$. However, one may possibly want to get an upper bound of the risk which is independent of the choices of p, which can be obtained by maximizing the risk function with respect to p.

$$\frac{d}{dp}\left[\frac{(1-p)p^2}{n}\right] = \frac{2p-3p^2}{n} = 0$$

gives $p^* = \frac{2}{3}$. Therefore,

$$\max_{p\in(0,1)} \mathcal{R}\left(\overline{X_n}^{-1}, p\right) = \frac{\left(1-\frac{2}{3}\right)\left(\frac{2}{3}\right)^2}{n} = \frac{4}{27n}.$$

Therefore, for large n

$$\mathcal{R}\left(\overline{X_n}^{-1}, p\right) \leq \frac{4}{27n}, \text{ for all } p \in (0,1).$$

One might ask the question that what would be the minimum required sample size so that the maximum risk would be less than some small number $\epsilon = 0.001$ (say). From the inequality, we can obtain $\frac{4}{27n} \leq 0.001$ implies $n \geq \frac{4}{27\times 0.001} = 148.1481$. Therefore, at least a sample size of $n \geq 149$ would be required to ensure the accuracy of the estimate less than 0.001. A general formula can be written as $n \geq \left[\frac{4}{27\epsilon}\right] + 1$ for a given accuracy level $\epsilon > 0$, where $[x]$ represents the greatest integer $\leq x$.

A not so common discrete distribution

The discrete uniform distribution on the set $\{1, 2, \ldots, \theta\}$, where $\theta \in \Theta = \{1, 2, 3, \ldots\}$ is given by

$$\mathrm{P}(X = x) = \begin{cases} \frac{1}{\theta}, x \in \{1, 2, \ldots, \theta\} \\ 0, \quad \text{otherwise.} \end{cases}$$

Suppose, we have a random sample of size n, $(X_1, X_2, \ldots, X_n)$ are available and let $Y_n = \max(X_1, \ldots, X_n)$. We are interested in estimating the parameter θ which can take any value from the countable set $\{1, 2, \ldots\}$. The likelihood function is given by

$$\mathcal{L}(\theta) = \begin{cases} \left(\frac{1}{\theta}\right)^n, & \theta \in \{y_n, y_n + 1, y_n + 2, \ldots\} \\ 0, & \text{otherwise} \end{cases}$$

In addition, it is easy to understand that the likelihood function attains it's maximum at y_n.

```r
theta = 6
n = 5
x = sample(1:theta, size = n, replace = TRUE)
y_n = max(x)
Lik = function(theta){
  if(theta<y_n)
    return(0)
  if(is.integer(theta) && theta>=y_n)
    return(theta^(-n))
}
theta_vals = 1:10
Lik_vals = numeric(length = length(theta_vals))
for(i in 1:length(theta_vals)){
  Lik_vals[i] = Lik(theta = theta_vals[i])
}
plot(theta_vals, Lik_vals, type = "h", pch = 19, col = "grey",
     cex = 1.4, ylab = expression(L(theta)), xlab = expression(theta))
points(theta_vals, Lik_vals, pch = 19, col = "blue", cex = 1.3)
```

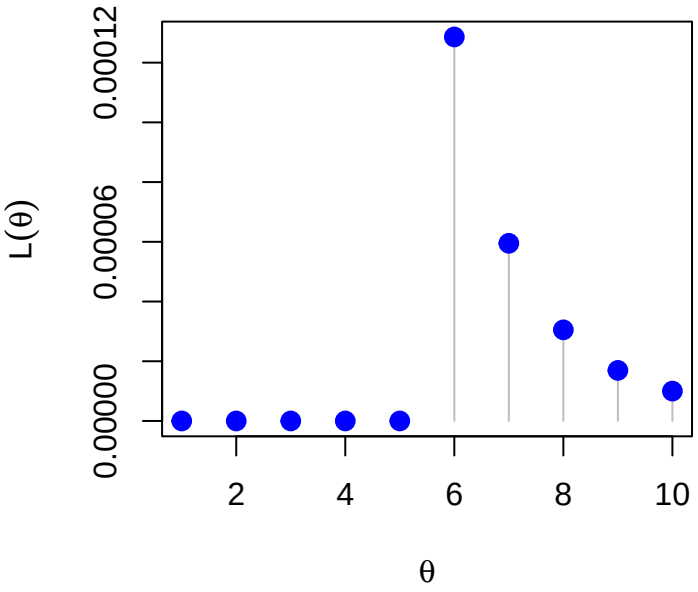


Figure 4: The likelihood function for the parameter θ based on a random sample of size $n = 5$ from the discrete uniform distribution on the set $\{1, 2, 3 \ldots, \theta = 6\}$. The `sample` function in R has been used to simulate the observations from the discrete uniform distribution.

An experiment with the continuous distribution

Suppose that we have a random sample $(X_1, \ldots, X_n)$ of size n from the exponential distribution with rate parameter λ. Let $X \sim \text{Exponential}(\lambda)$ be the population distribution and the parameter space is $\Theta = (0, \infty)$. We are interested in estimating the probability $\mathrm{P}(X \leq 1)$.

First we make the observation that the desired quantity $\psi = \mathrm{P}(X \leq 1)$ is a function of λ, which is given by

$$\psi = \int_0^1 \lambda e^{-\lambda x} dx = 1 - e^{-\lambda}.$$

The original parameter $\lambda \in (0, \infty)$, which implies that $\psi \in (0, 1)$. As $\lambda \to \infty$, $\psi \to 1$ and as $\lambda \to 0$, $\psi \to 0$. Our aim is to estimate ψ based on the observations which is a function of λ. Let us understand this problem step by step. First we need to estimate the parameter λ and then we can use it to approximate ψ.

Computation of MLE of the parameter

Suppose that $(x_1, x_2, \ldots, x_n)$ be the collected sample of size n: - The likelihood function is given by

$$\mathcal{L}(\lambda) = f(x_1, x_2, \ldots, x_n|\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i}.$$

The function is explicitly written as

$$\mathcal{L}(\lambda) = \begin{cases} \lambda^n e^{-\lambda \sum_{i=1}^n x_i}, & 0 < \lambda < \infty \\ 0, & \text{otherwise.} \end{cases}.$$

The log-likelihood function is $l(\lambda) = \log \mathcal{L}(\lambda)$ is given by

$$l(\lambda) = n \log \lambda - \lambda \sum_{i=1}^{n} x_i.$$

Equating $l'(\lambda) = 0$, we obtain $\lambda^* = \frac{n}{\sum_{i=1}^n x_i} = \frac{1}{\overline{x_n}}$, inverse of the sample mean. It is easy to see that $l''(\lambda) < 0$ for all $\lambda \in (0, \infty)$, therefore, $l''(\lambda^*) < 0$. Therefore, the Maximum Likelihood Estimator (MLE) of the parameter λ, based on a sample of size n, is given by

$$\widehat{\lambda_n} = \left(\overline{X_n}\right)^{-1}.$$

Here I would like to make an extremely important point which many people miss. It is important to note that in the above discussion λ^* is a `fixed quantity` which is computed based on a single realization of the sample $(X_1, \ldots, X_n)$, whereas $\widehat{\lambda_n}$ is a `random quantity` which can be characterized by a probability distribution.

Computation of MLE of the function of parameter

To obtain the estimator of ψ, based on the above sample, a natural choice of the estimator of ψ is given by

$$\widehat{\psi_n} = 1 - e^{-\widehat{\lambda_n}},$$

where $\widehat{\lambda_n}$ is the MLE of λ. The following observations we have:

- The MLE $\hat{\lambda}$ is a function of the sample mean $\overline{X_n}$.
- For the sample mean $\overline{X_n}$, we have large sample normal approximation due to the Central Limit Theorem. The underlying population has finite variance, therefore, CLT holds. It would be interesting to see whether the MLE $\widehat{\lambda_n} = \frac{1}{\overline{X_n}}$, a function of $\overline{X_n}$ follows some approximate distribution at least for large n. Also, can this result be extended for $\widehat{\psi_n}$, a nonlinear function of the MLE?
- By the WLLN, we know that $\overline{X_n} \xrightarrow{P} \mathrm{E}(X)$. Is it true that $\widehat{\lambda_n} \xrightarrow{P} \lambda$ for all $\lambda \in (0, \infty)$. Also, can this result be extended for $\widehat{\psi_n}$, a nonlinear function of the MLE?

Convergence of Estimators $\widehat{\lambda_n}$ and $\widehat{\psi_n}$ for large n

Using the Weak Law of Large Numbers, we know that

$$\overline{X_n} \xrightarrow{P} \mathrm{E}(X) = \frac{1}{\lambda}.$$

Our claim is that

$$\widehat{\lambda_n} \xrightarrow{P} \lambda, \qquad \widehat{\psi_n} \xrightarrow{P} \psi.$$

Before going into the theoretical justifications, let us see by computer simulations, how these estimators behaves as the sample size n increases. We implement the following algorithm:

- Fix $\lambda \in (0, \infty)$
- Fix $n \in \{1, 2, \ldots, 1000\}$
- For each n
  - Simulate $X_1, X_2, \ldots, X_n \sim$ Exponential(λ)
  - Compute $\overline{X_n}$
  - Compute $\widehat{\lambda_n} = \overline{X_n}^{-1}$
  - Compute $\widehat{\psi_n} = 1 - e^{-\widehat{\lambda_n}}$.
- Plot the pairs $\left(n, \overline{X_n}\right), n \in \{1, 2, \ldots, 1000\}$
- Plot the pairs $\left(n, \widehat{\lambda_n}\right), n \in \{1, 2, \ldots, 1000\}$.
- Plot the pairs $\left(n, \widehat{\psi_n}\right), n \in \{1, 2, \ldots, 1000\}$

```r
par(mfrow = c(1,3))
lambda = 2                                          # rate parameter
psi = 1-exp(-lambda)                                # true probability

n_vals = 1:1000                                     # increasing sample size
sample_means = numeric(length = length(n_vals))
lambda_hat = numeric(length = length(n_vals))
psi_hat = numeric(length = length(n_vals))

for(n in n_vals){
  x = rexp(n = n, rate = lambda)
  sample_means[n] = mean(x)
  lambda_hat[n] = 1/sample_means[n]                 # estimate of the rate parameter
  psi_hat[n] = 1 - exp(-lambda_hat[n])              # estimate of the probability (psi)
}
plot(n_vals, sample_means, col = "red", ylab = "",
     type = "l", xlab = "sample size (n)",
     main = expression(bar(X[n])), cex.lab = 1.5,
     cex.main = 1.5)                                # Sampling distribution of MLE of lambda
abline(h = 1/lambda, lty = 2, col = "blue", lwd = 3)

plot(n_vals, lambda_hat, col = "red", ylab = "",    # Consistency of MLE of lambda
     type = "l", xlab = "sample size (n)",
     main = expression(widehat(lambda)), cex.lab = 1.5,
     cex.main = 1.5)
abline(h = lambda, lty = 2, col = "blue", lwd = 3)

plot(n_vals, psi_hat, col = "red", ylab = "",       # Consistency of MLE of psi
     type = "l", xlab = "sample size (n)",
     main = expression(widehat(psi[n]) == 1-e^{-widehat(lambda[n])}) , cex.lab = 1.5,
     cex.main = 1.5)
abline(h = psi, lty = 2, col = "blue", lwd = 3)
```

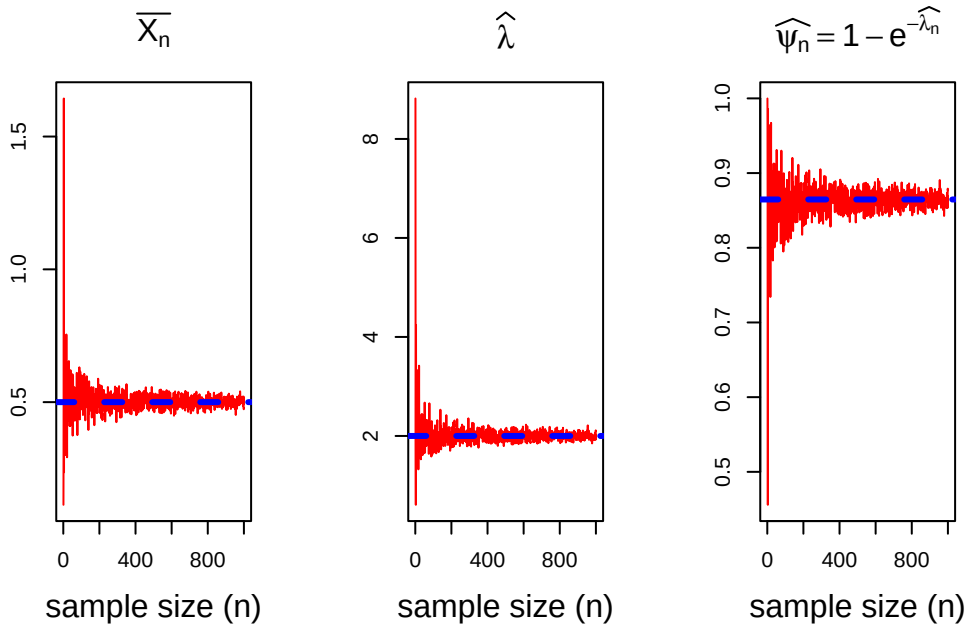


Figure 5: As the sample size n increase, sample mean converges to the population mean (WLLN) (left panel), the MLE converges to the true value of the parameter (middle panel), the function of the MLE converges to the function of the parameter (right most panel).

The figures clearly suggests that all the estimators converge to the true value as $n \to \infty$. This is due to the fact that both $\widehat{\lambda_n}$ and $\widehat{\psi_n}$ are continuous transformation of $\overline{X_n}$. Let us now look into the large sample approximation of the sampling distributions of $\widehat{\lambda_n}$ and $\widehat{\psi_n}$.

Large sample approximations

We recall that as the sample size $n \to \infty$, $\overline{X_n}$ can be well approximated by the normal distribution. More specifically, in our context, for large n,

$$\overline{X_n} \overset{a}{\sim} \mathcal{N}\left(\frac{1}{\lambda}, \frac{1}{\lambda^2 n}\right)$$

as for the given population $\mathrm{E}(X) = \frac{1}{\lambda}$ and $Var(X) = \frac{1}{\lambda^2}$. Before going into some theoretical justifications, let us visualize the sampling distribution of $\widehat{\lambda_n}$ and $\widehat{\psi_n}$ for different choices (ascending order) of n. The simulation scheme is already demonstrated for the Geometric(p) distribution in the previous section.

```r
par(mfrow = c(1,3))
M = 1000                                    # number of replications
n = 500                                     # sample size
lambda = 2                                  # true rate parameter
sample_means = numeric(length = M)          # store sample means
lambda_hat = numeric(length = M)            # store MLEs of lambda
psi_hat = numeric(length = M)               # store MLEs of psi
for (i in 1:M) {
  x = rexp(n = n, rate = lambda)            # simulate from population
  sample_means[i] = mean(x)                 # compute sample means
  lambda_hat[i] = 1/mean(x)                 # compute MLE of lambda
  psi_hat[i] = 1 - exp(-lambda_hat[i])      # compute MLE of psi
}
hist(sample_means, probability = TRUE,
     main = paste("n = ", n), xlab = expression(bar(X[n])), cex.lab = 1.3)
curve(dnorm(x, mean = 1/lambda, sd = 1/sqrt(lambda^2*n)),
      add = TRUE, col = "red", lwd = 2)

hist(lambda_hat, probability = TRUE,
     main = paste("n = ", n), xlab = expression(widehat(lambda[n])), cex.lab = 1.3)
curve(dnorm(x, mean = lambda, sd = sqrt(lambda^2/n)),
      add = TRUE, col = "red", lwd = 2)

hist(psi_hat, probability = TRUE,
     main = paste("n = ", n), xlab = expression(g(widehat(lambda[n]))), cex.lab = 1.3)
points(psi, 0, pch = 19, col = "red",
       cex = 1.4)
curve(dnorm(x, mean = psi, sd = sqrt(lambda^2*exp(-2*lambda)/n)),
      add = TRUE, col = "red", lwd = 2)
```

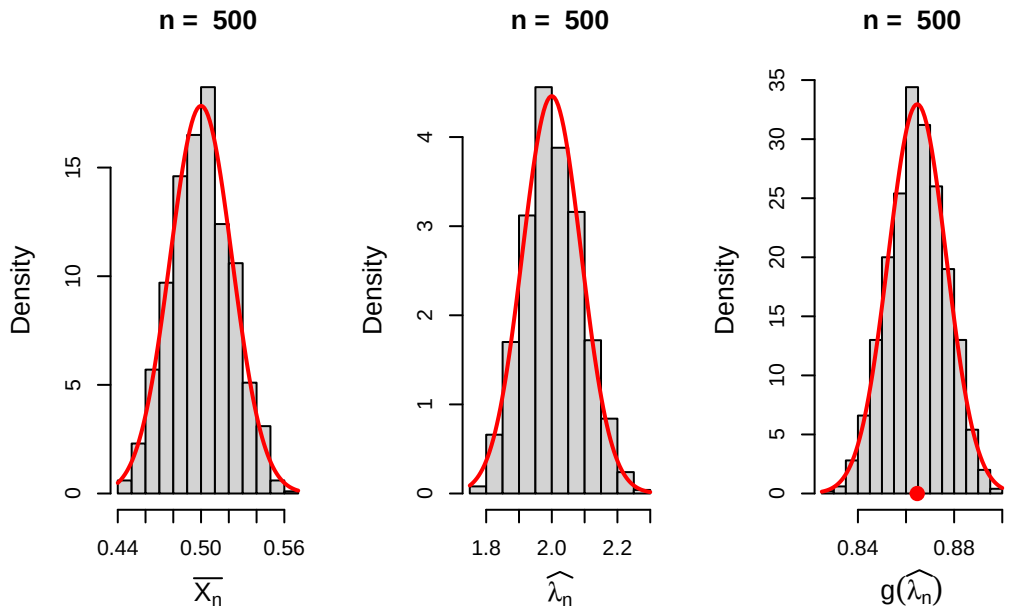


Figure 6: The sampling distribution for $\overline{X_n}$, $\widehat{\lambda_n}$, $\widehat{\psi_n}$ are obtained based on $M = 1000$ replications. The value of n is kept are $n = 500$ and it can be observed the sampling distributions take bell shaped nature. The parameter $\lambda = 2$ is consdiered for the simulation purpose. Therefore, the true value of $\psi = 1 - e^{-2} = 0.8646647$, which is marked as a red dot in the right most panel.

The figure clearly suggested as the sample size n increases, the sampling distributions of the MLE and the function of MLE are well approximated by the normal distribution. Therefore, if we can compute the mean and variance of the estimators at least for large n, we can get the normal approximation of these histograms.

Normal approximation of $\widehat{\lambda_n}$

Consider $\widehat{\lambda_n} = g\left(\overline{X_n}\right) = \frac{1}{\overline{X_n}}$. Taking Taylor expansion of first order about the expected value of $\overline{X_n} = \frac{1}{\lambda}\ \lambda$, we get

$$g\left(\overline{X_n}\right) \approx g\left(\frac{1}{\lambda}\right) + \left(\overline{X_n} - \frac{1}{\lambda}\right) g'\left(\frac{1}{\lambda}\right).$$

Since $\mathrm{E}\left(\overline{X_n}\right) = \frac{1}{\lambda}$, we get

$$\mathrm{E}\left(\widehat{\lambda_n}\right) = \mathrm{E}\left(\frac{1}{\overline{X_n}}\right) \approx g\left(\frac{1}{\lambda}\right) = \lambda.$$

Therefore, the MLE $\widehat{\lambda_n}$ is asymptotically unbiased estimator of λ. To compute the variance, we observe that

$$Var\left(\widehat{\lambda_n}\right) \approx \mathrm{E}\left(\widehat{\lambda_n} - \lambda\right)^2 \approx \mathrm{E}\left(\overline{X_n} - \frac{1}{\lambda}\right)^2 \left(g'\left(\frac{1}{\lambda}\right)\right)^2 = \frac{\lambda^4}{\lambda^2 n} = \frac{\lambda^2}{n}.$$

Therefore, we can overlay a normal distribution on the simulated histograms approximating the sampling distribution of $\widehat{\lambda_n}$ and check whether

$$\widehat{\lambda_n} \overset{a}{\sim} \mathcal{N}\left(\lambda, \frac{\lambda^2}{n}\right).$$

In the next phase, we can approximate the mean and variance of $\widehat{\psi_n} = g\left(\widehat{\lambda_n}\right)$, (say), where $g(x) = 1 - e^{-x}$. Considering the first order Taylor's approximation of $g\left(\widehat{\lambda_n}\right)$ about the expected value of $\widehat{\lambda_n}$, which is approximately equal to λ for large n, we obtain:

$$g\left(\widehat{\lambda_n}\right) \approx g(\lambda) + \left(\widehat{\lambda_n} - \lambda\right) g'(\lambda)$$

As $n \to \infty$, $\mathrm{E}\left(\widehat{\psi_n}\right) = \mathrm{E}\left[g\left(\widehat{\lambda_n}\right)\right] \approx g(\lambda) = 1 - e^{-\lambda} = \psi$, therefore, $\widehat{\psi_n}$ is an asymptotically unbiased estimator of ψ. The approximate variance of $\widehat{\psi_n}$, for large n is obtained as

$$Var\left(\widehat{\psi_n}\right) \approx \mathrm{E}\left(\widehat{\psi_n} - \psi\right)^2 = \mathrm{E}\left[g\left(\widehat{\lambda_n}\right) - g(\lambda)\right]^2 \approx \mathrm{E}\left(\widehat{\lambda_n} - \lambda\right)^2 \left(g'(\lambda)\right)^2.$$

This gives $Var\left(\widehat{\psi_n}\right) \approx \frac{\lambda^2 e^{-2\lambda}}{n}$. Therefore, we can claim that

$$\widehat{\psi_n} = 1 - e^{-\widehat{\lambda_n}} \overset{a}{\sim} \mathcal{N}\left(1 - e^{-\lambda}, \frac{\lambda^2 e^{-2\lambda}}{n}\right).$$

Visualization for different choices of n

In the above approximations, we have approximated the sampling distribution of $\overline{X_n}$, $\widehat{\lambda_n}$ and $\widehat{\psi_n} = g(\widehat{\lambda_n}) = 1 - e^{-\widehat{\lambda_n}}$ for a fixed value of n using the normal distribution. In the following codes, we visualize the sampling distribution for different choices of n. The histograms are overlaid with the normal distribution with the mean and variance computed by the Delta method.

```
n_vals = c(5, 10, 50, 100, 500, 1000)
par(mfrow = c(2,3))
M = 1000
lambda = 2
for(n in n_vals){
```

```r
  sample_means = numeric(length = M)
  lambda_hat = numeric(length = M)
  psi_hat = numeric(length = M)
  for (i in 1:M) {
    x = rexp(n = n, rate = lambda)
    sample_means[i] = mean(x)
    lambda_hat[i] = 1/mean(x)
    psi_hat[i] = 1 - exp(-lambda_hat[i])
  }
  hist(sample_means, probability = TRUE,
       main = paste("n = ", n), xlab = expression(bar(X[n])), cex.lab = 1.3)
  curve(dnorm(x, mean = 1/lambda, sd = 1/sqrt(lambda^2*n)),
        add = TRUE, col = "red", lwd = 2)

  hist(lambda_hat, probability = TRUE,
       main = paste("n = ", n), xlab = expression(widehat(lambda[n])), cex.lab = 1.3)
  curve(dnorm(x, mean = lambda, sd = sqrt(lambda^2/n)),
        add = TRUE, col = "red", lwd = 2)

  hist(psi_hat, probability = TRUE,
       main = paste("n = ", n), xlab = expression(g(widehat(lambda[n]))), cex.lab = 1.3)
  points(psi, 0, pch = 19, col = "red",
         cex = 1.4)
  curve(dnorm(x, mean = psi, sd = sqrt(lambda^2*exp(-2*lambda)/n)),
        add = TRUE, col = "red", lwd = 2)
}
```

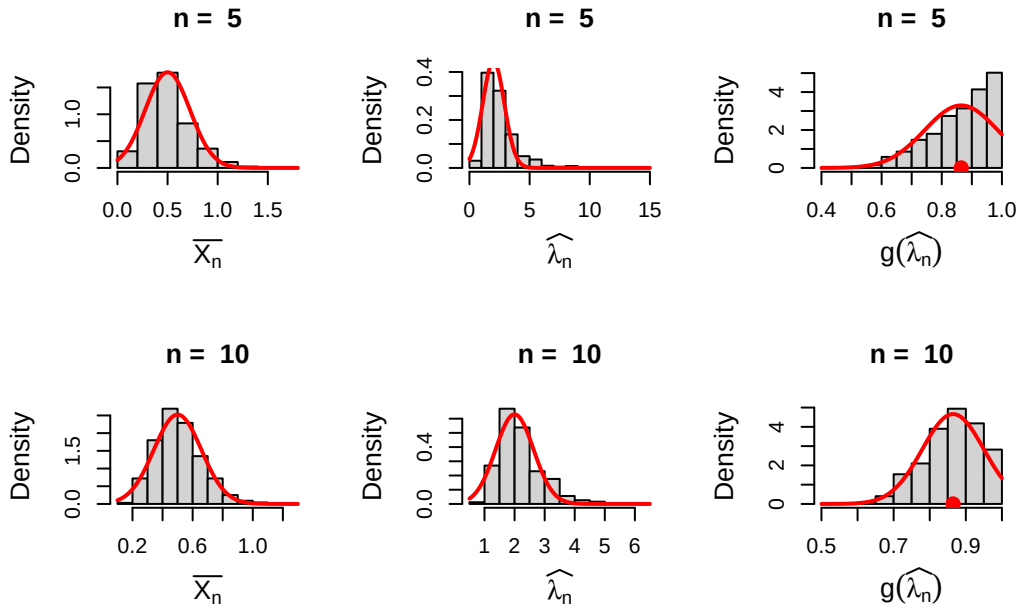


Figure 7: As the sample size n increases, the sampling distributions of the functions of the data points are well approximated by the normal distributions.

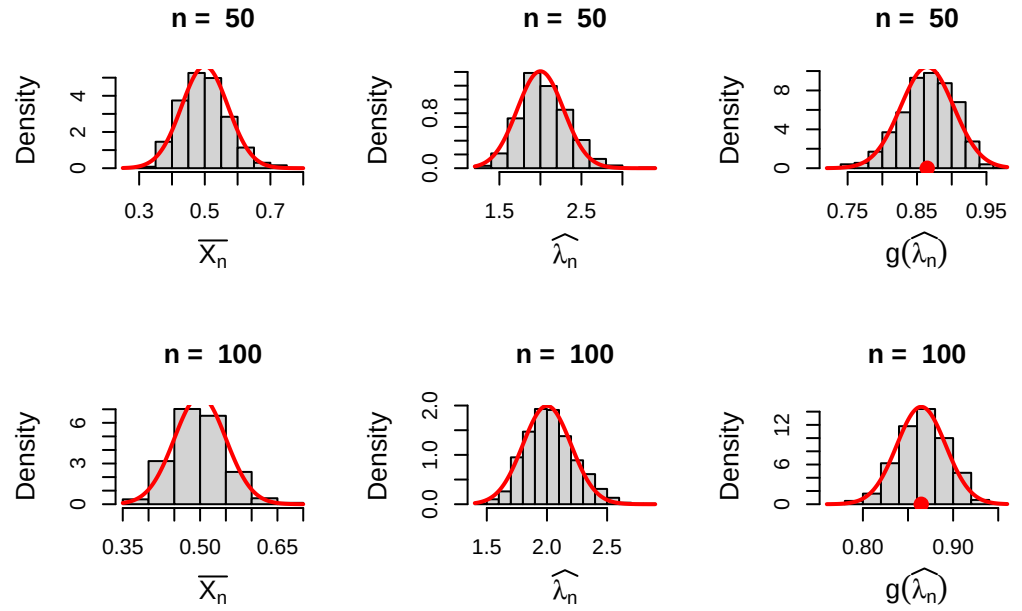


Figure 8: As the sample size n increases, the sampling distributions of the functions of the data points are well approximated by the normal distributions.

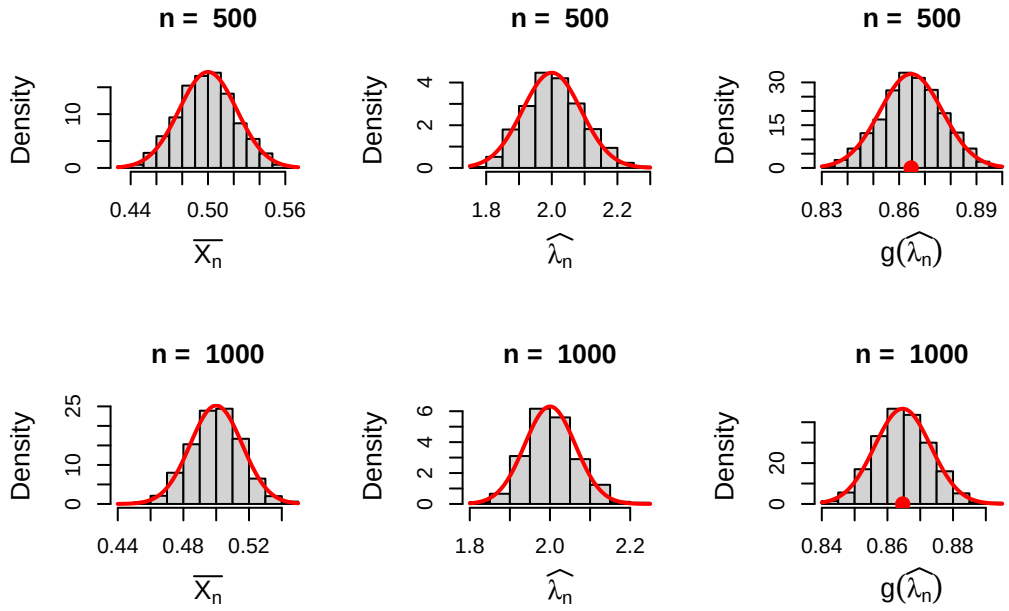


Figure 9: As the sample size n increases, the sampling distributions of the functions of the data points are well approximated by the normal distributions.

i Invariance property of MLE

Suppose $X_1, \ldots, X_n$ be a random sample from the population PDF (PMF) $f(x|\theta)$. If $\widehat{\theta_n}$ be the MLE of θ, then for any function $g(\theta)$, the MLE of $g(\theta)$ is $g\left(\widehat{\theta_n}\right)$.

Approximation of the Risk function

In the language of risk function, we can write that for large n,

$$\mathcal{R}\left(\widehat{\lambda_n}, \lambda\right) = \mathrm{E}\left(\widehat{\lambda_n} - \lambda\right)^2 \approx \frac{\lambda^2}{n}.$$

However, for the estimator $\widehat{\psi_n}$, we have computed the risk as a function of λ only. By expressing λ in terms of ψ (it is a monotone transformation from $\lambda \to \psi$) as $\lambda = -\log(1-\psi)$, we obtain the approximate risk function as

$$\mathcal{R}\left(\widehat{\psi_n}, \psi\right) \approx \frac{(-(1-\psi)\log(1-\psi))^2}{n}, \text{ for large } n, \psi \in (0,1).$$

The verification of the above expression can be performed by computer simulation. Instead of discretizing the range of λ, discretize the possible parameter space corresponds to $\psi \in (0,1)$.

```r
par(mfrow = c(2,3))
n_vals = c(5, 10, 30, 50, 100, 500)
M = 5000
psi_vals = seq(0.01, 0.9, by = 0.05)
for(n in n_vals){
  risk_psi_hat = numeric(length = length(psi_vals))
  for(i in 1:length(psi_vals)){
    psi = psi_vals[i]                       # true probability
    lambda = -log(1-psi)                    # rate for exponential
    loss_psi_hat = numeric(length = M)
    for(j in 1:M){
      x = rexp(n = n, rate = lambda)
      psi_hat = 1-exp(-1/mean(x))
      loss_psi_hat[j] = (psi_hat-psi)^2
    }
    risk_psi_hat[i] = mean(loss_psi_hat)
  }
  plot(psi_vals, risk_psi_hat, col = "red", xlab = expression(psi),
       ylab = expression(R(widehat(psi[n]),psi)),
       main = paste("n = ", n), pch = 19, cex = 1.3)
  lines(psi_vals, (-(1-psi_vals)*log(1-psi_vals))^2/n,
        col = "blue", lwd = 3, lty = 2)
}
```

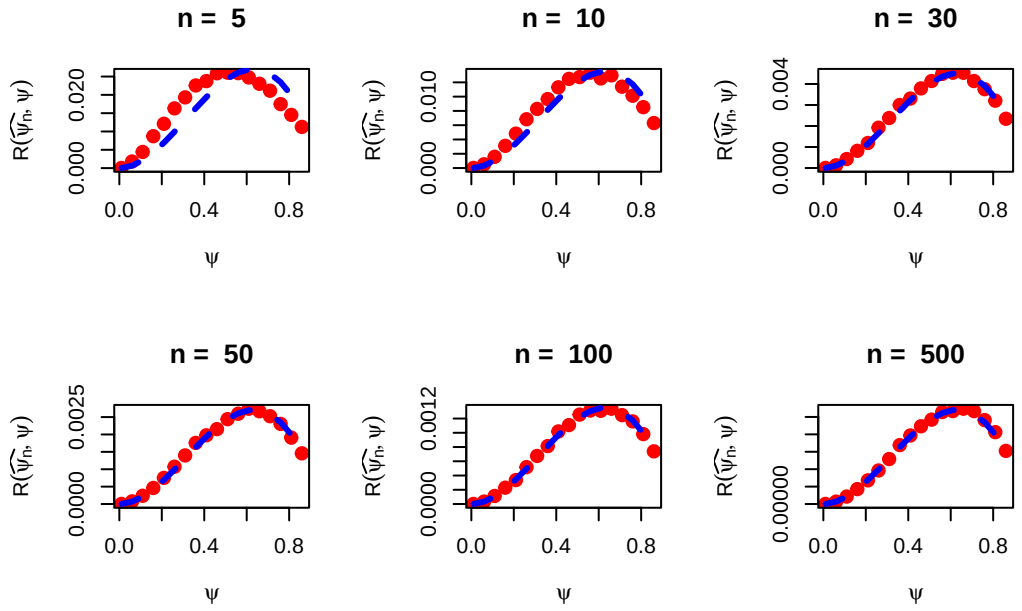


Figure 10: As the sample size n increases, the risk function $\mathcal{R}\left(\widehat{\psi_n}, \psi\right)$ can be well approximated by the approximate risk obtained by the first order Taylor's polynomial approximation. The approximated risk function based on $M = 1000$ replications is shown in red color, which is the approximation of the true risk function based on simulation. The approximation of the true risk function by the first order Taylor's polynomial is shown in blue dotted line. It is evident that as the sample size increases, the approximation is very close.

! Delta Method

Let W_n be a sequence of random variables that satisfies

$$\sqrt{n}\left(W_n - \theta\right) \overset{a}{\sim} \mathcal{N}(0, \sigma^2)$$

in distribution. For a given function $g(\cdot)$ and a specific value of $\theta \in \Theta$, suppose that $g'(\theta)$ exists and $g'(\theta) \neq 0$. Then

$$\sqrt{n}\left[g\left(W_n\right) - g(\theta)\right] \overset{a}{\sim} \mathcal{N}\left(0, \sigma^2\left[g'(\theta)\right]^2\right),$$

in distribution.
The basic idea is that as the sample size increases, the function of asymptotically normally distributed random variables can also be approximated by the normal distribution mean and variance can be easily computed by using the first order Taylor's polynomial approximation.

! Comparing Risk Functions

Suppose in the same problem of estimating $\psi = \mathrm{P}(X \geq 1)$, another alternative estimator

$$\widehat{\xi_n} = \frac{1}{n}\sum_{i=1}^{n} I(X_i \geq 1)$$

is suggested, where the quantity $I(X \geq 1)$ is defined as

$$I(X \geq 1) = \begin{cases} 1, X \geq 1 \\ 0, X < 1 \end{cases} .$$

Therefore, $\widehat{\xi_n}$ basically the proportion of observations which are greater than or equal to 1. It is easier to check that $\mathrm{E}\left(\widehat{\xi_n}\right) = \psi$ and $Var\left(\widehat{\xi_n}\right) = \frac{\psi(1-\psi)}{n}$. Using computer simulation, simulate the risk function $\mathcal{R}\left(\widehat{\xi_n}, \psi\right)$ and compare it with $\mathcal{R}\left(\widehat{\psi_n}, \psi\right)$ at different values of $\psi \in (0, 1)$ for different sample sizes n. Write your conclusion in simple English language.

Multi-parameter optimization

In the previous sections, we have learnt that how one can obtain the sampling distribution of the MLE and also tested whether the MLE converges in probability to the true parameter value by using computer simulations. Many real life problems are modeled by probability distributions which is parameterized by more than one parameters, for example, $\mathcal{N}(\mu, \sigma^2)$, $\mathcal{B}(a, b)$, $\mathcal{G}(a, b)$, etc. In this section, we consider the computation of the MLE for multiparameter probability distributions.

Suppose that we have a random sample $(X_1, X_2, \ldots, X_n)$ of size n from the normal distribution with parameters μ and σ^2. Our goal is to obtain the Maximum Likelihood Estimators of μ and σ^2 based on the sample observations and study properties of these estimators. The parameter space is

$$\Theta = \left\{(\mu, \sigma^2) : -\infty < \mu < \infty, 0 < \sigma < \infty\right\}$$

```r
n = 30
mu = 3
sigma2 = 1.5
x = rnorm(n = n, mean = mu, sd = sqrt(sigma2))
hist(x, probability = TRUE)
```

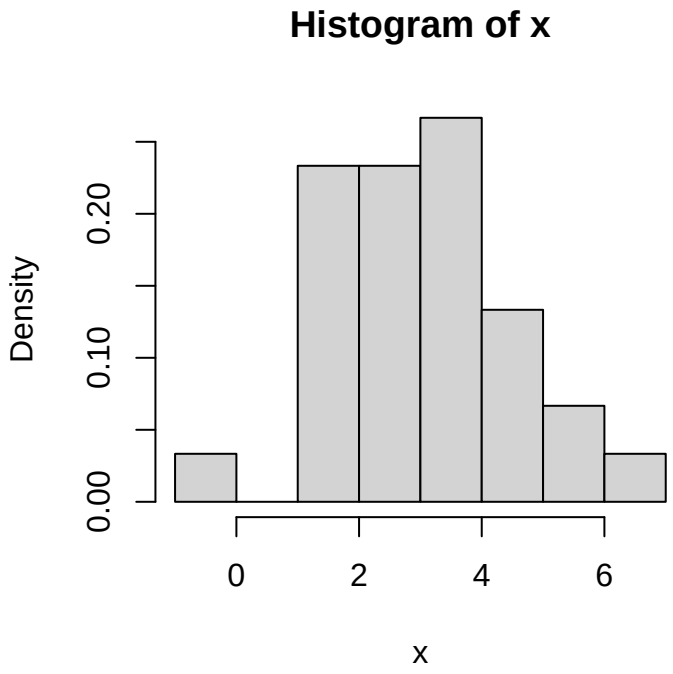


Figure 11: A random sample of size $n = 30$ has been simulated from the normal distribution with mean $\mu = 3$ and variance $\sigma^2 = 1.5$ for demonstration. The corresponding histogram is used for graphical display of the data.

Likelihood function

If $(x_1, x_2, \ldots, x_n)$ be a fixed observed sample of size n, then the likelihood function is given by

$$\mathcal{L}(\mu, \sigma^2) = \left(2\pi\sigma^2\right)^{-\frac{n}{2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2}, -\infty < \mu < \infty, 0 < \sigma < \infty.$$

The log-likelihood function is given by

$$l\left(\mu, \sigma^2\right) = \log \mathcal{L}(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i - \mu\right)^2.$$

- Compute $\frac{\partial l}{\partial \mu}$ and $\frac{\partial l}{\partial \sigma^2}$ and set them equal to zero. - Solve the above system simultaneously and obtain the critical points μ^* and σ^{2^*}. - Show the the the following matrix

$$H = \begin{bmatrix} \frac{\partial^2 l}{\partial \mu^2} & \frac{\partial^2 l}{\partial \mu \partial \sigma^2} \\ \frac{\partial^2 l}{\partial \mu \partial \sigma^2} & \frac{\partial^2 l}{\partial (\sigma^2)^2} \end{bmatrix}$$

is negative definite when the partial derivatives are evaluated at (μ^*, σ^{2^*}). This can be checked by ensuring the following two conditions:

$$H_{11} < 0, \quad \text{and} \quad \det(H) = H_{11}H_{22} - \left(H_{12}\right)^2 > 0.$$

In the following, we plot the likelihood function and the log-likelihood function as a function of the parameters μ and σ^2.

```r
Likelihood = function(mu, sigma2){
  (1/(sigma2*2*pi)^(n/2))*exp(-sum(x-mu)^2/(2*sigma2))
}

LogLikelihood = function(mu, sigma2){
  log(Likelihood(mu = mu, sigma2 = sigma2) + 0.5)
}

mu_vals = seq(2, 4, by = 0.1)
sigma2_vals = seq(1, 1.5, by = 0.1)

Lik_vals = matrix(data = NA, nrow = length(mu_vals),
                  ncol = length(sigma2_vals))
LogLik_vals = matrix(data = NA, nrow = length(mu_vals),
                  ncol = length(sigma2_vals))
for(i in 1:length(mu_vals)){
  for(j in 1:length(sigma2_vals)){
    Lik_vals[i,j] = Likelihood(mu = mu_vals[i],
                               sigma2 = sigma2_vals[j])
    LogLik_vals[i,j] = LogLikelihood(mu = mu_vals[i],
                               sigma2 = sigma2_vals[j])
  }
}
par(mfrow = c(1,2))
persp(mu_vals, sigma2_vals, Lik_vals, theta = 60, col = "yellow", )
persp(mu_vals, sigma2_vals, LogLik_vals, theta = 60, col = "yellow")
```

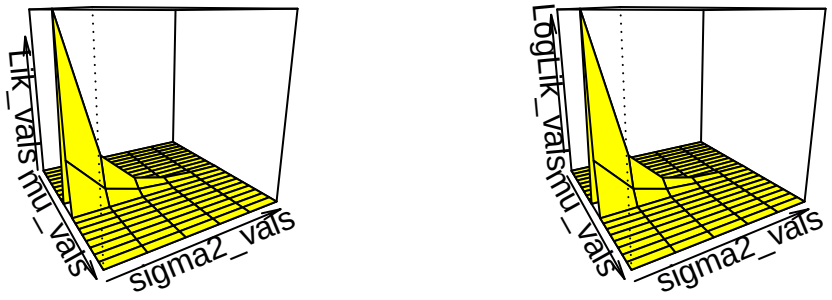


Figure 12: The shape of the likelihood and the log-likelihood function (surface) which is a function of μ and σ^2. The likelihood surface clearly indicates that there is unique maximum.

From the computation, we observed that the MLE of the μ and σ^2 are given by

$$\widehat{\mu_n} = \overline{X_n}, \quad \widehat{\sigma_n^2} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X_n}\right)^2$$

Given an estimators, the following questions immediately appear, such as (a) are these estimators unbiased? (b) are these estimators consistent, (c) are these estimators are the best choices under some given loss function? We shall not address all these questions here only, but, certainly show how we basic properties of these estimators can be checked and when feasible, we shall also compute the exact sampling distribution of these estimators.

At the first step, let us check whether as the sample size increases, the MLEs converge to the true parameter values. Mathematically, whether the following statements true:

$$\overline{X_n} \xrightarrow{P} \mu, \quad \text{and} \quad \widehat{\sigma_n^2} \xrightarrow{P} \sigma^2.$$

The following simulation will help us to understand whether the above statements are true.

- Fix sample size $n \in \{2, 3, \ldots, 1000\}$.
- Fix $\mu = \mu_0$ and $\sigma^2 = \sigma_0^2$.
- For each n
  - Simulate $X_1, X_2, \ldots, X_n \sim \mathcal{N}(\mu_0, \sigma_0^2)$.
  - Compute $\overline{X_n}$ and $\widehat{\sigma_n^2}$
- Plot the pairs $\left(n, \overline{X_n}\right)$ and $\left(n, \widehat{\sigma_n^2}\right)$
- Add a horizontal straight line to the plots at μ_0 and σ_0^2, respectively.

```
n_vals = 2:1000
mu_hat = numeric(length = length(n_vals))
sigma2_hat = numeric(length = length(n_vals))
for(i in 1:length(n_vals)){
  n = n_vals[i]
  x = rnorm(n = n, mean = mu, sd = sqrt(sigma2))
  mu_hat[i] = mean(x)
  sigma2_hat[i] = (1/n)*sum((x-mean(x))^2)
}
par(mfrow = c(1,2))
plot(n_vals, mu_hat, col = "grey", lwd = 2,
     xlab = "sample size (n)", ylab = expression(widehat(mu[n])))
abline(h = mu, col = "blue", lwd = 3, lty = 2)
plot(n_vals, sigma2_hat, col = "grey", lwd = 2,
     xlab = "sample size (n)", ylab = expression(widehat(sigma[n]^2)))
abline(h = sigma2, col = "blue", lwd = 3, lty = 2)
```

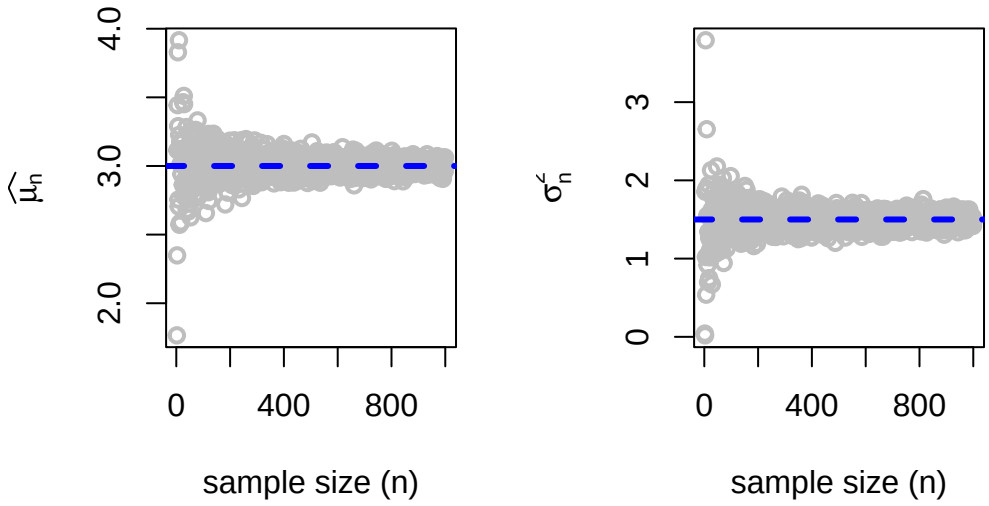


Figure 13: The simulations cleraly indicates that as the sample size increases, MLEs are getting highly concentrated about the true values, which is a signature of the convergence in probability. The simulation has been carried out by fixing the population distribution as $\mathcal{N}(\mu = 3, \sigma^2 = 1.5)$.

To quantify the uncertainty associated with the MLE, we can either obtain the exact sampling distributions of $\overline{X_n}$ and $\widehat{\sigma^2_n}$, or we can approximate the standard error of these estimators at different true values by computer simulations by fixing n. Let us fix the value of n and simulate $M = 1000$ realizations from the sampling distributions of the MLEs and visualize their distribution through histograms.

```
n = 10                                          # sample size
M = 1000                                        # number of replications
mu_hat = numeric(length = M)
sigma2_hat = numeric(length = M)
for(i in 1:M){
  x = rnorm(n = n, mean = mu, sd = sqrt(sigma2))
  mu_hat[i] = mean(x)
  sigma2_hat[i] = (1/n)*sum((x-mean(x))^2)
}
par(mfrow = c(1,2))
hist(mu_hat, probability = TRUE, main = paste("n = ", n),
     xlab = expression(widehat(mu[n])), breaks = 30)
curve(dnorm(x, mean = mu, sd = sqrt(sigma2/n)),
      add = TRUE, col = "red", lwd = 2)
hist(sigma2_hat, probability = TRUE, main = paste("n = ", n),
     xlab = expression(widehat(sigma[n]^2)), breaks = 30)
dist_sigma2_hat = function(x){
```

```r
  num = exp(-n*x/(2*sigma2))*(n*x/sigma2)^((n-1)/2-1)*n*(x>0)
  den = gamma((n-1)/2)*2^((n-1)/2)*sigma2
  return(num/den)
}
curve(dist_sigma2_hat(x), add = TRUE, col = "red", lwd = 2)
```

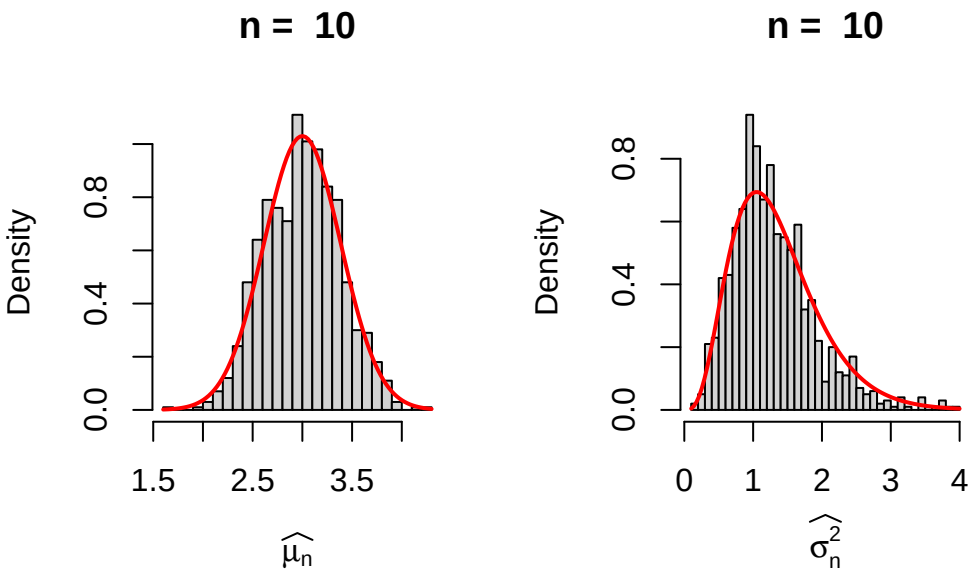


Figure 14: The sampling distribution of the MLEs of μ and σ^2 have been visualized through histograms based on $M = 1000$ replications. The reader is encouraged to update the code and visualize the histograms for different choices of n. The exact sampling distributions are overlaid on the hisgraoms.

In the classroom, Sangeeta has pointed out that sample mean would in fact follow the normal distribution due to the Central Limit Theorem (n large). The statement is indeed true. However, since the sampling has been carried out from the normal distribution, the sample mean has an exact normal distribution with mean μ and variance $\frac{\sigma^2}{n}$ for every n (not only for large n), which can be proved using the MGF technique, shown below:

$$M_{\overline{X}_n}(t) = \cdots \text{ fill the gap } \cdots = e^{\mu t + \frac{1}{2}\left(\frac{\sigma^2}{n}\right)t}, -\infty < t < \infty.$$

While for the sample mean, the computation is apparently straightforward application of the MGF. For the the MLE of σ^2, a slightly lengthier calculations is required. In the following subsection, we list main results related to the sampling from the normally distributed population.

! Sampling from the normal distribution

Let $X_1, X_2, \ldots, X_n$ be a random sample of size n from the $\mathcal{N}(\mu, \sigma^2)$ population distribution. Then

- $\overline{X_n}$ and S_n^2 are independent random variables.
- $\overline{X_n}$ has a $\mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$ distribution.
- $\frac{(n-1)S_n^2}{\sigma^2}$ has a χ^2_{n-1} distribution with $n-1$ degrees of freedom.

Reader is encouraged to see the proofs in (Casella and Berger 2002) and (Rohatgi and Saleh 2000). I encourage to see both the proofs as they used two different approaches. Casella and Berger (2002) used the transformation formula to explicitly derive the joint distribution of $\overline{X_n}$ and S_n^2, whereas Rohatgi and Saleh (2000) elegantly used the joint moment generating function to prove the above results.

Considering $U_n = \frac{(n-1)S_n^2}{\sigma^2} \sim \chi^2_{n-1}$, the sampling distribution of the MLE $V_n = \widehat{\sigma_n^2} = \frac{\sigma^2}{n} \times U_n$ can be easily obtained. The CDF of V_n is obtained as

$$F_{V_n}(v) = \mathrm{P}(V_n \leq v) = \mathrm{P}\left(U_n \leq \frac{nv}{\sigma^2}\right).$$

Therefore,

$$f_{V_n}(v) = \frac{d}{dv} F_{V_n}(v) = f_{U_n}\left(\frac{nv}{\sigma^2}\right) \cdot \frac{n}{\sigma^2}, 0 < v < \infty.$$

For every n, the sampling distribution of the MLE of σ^2, $\widehat{\sigma_n^2}$ is given by:

$$f_{\widehat{\sigma_n^2}}(v) = \frac{e^{-\frac{nv}{2\sigma^2}} \left(\frac{nv}{\sigma^2}\right)^{\frac{n-1}{2}-1}}{\Gamma\left(\frac{n-1}{2}\right) 2^{\frac{n-1}{2}}} \cdot \frac{n}{\sigma^2}, 0 < v < \infty.$$

```r
mu = 3
sigma2 = 1.5
n_vals = c(3,5,10,50,100, 200)
M = 1000
par(mfrow = c(2,3))
for(n in n_vals){
  mu_hat = numeric(M)
  sigma2_hat = numeric(M)
  for(i in 1:M){
    x = rnorm(n = n, mean = mu, sd = sqrt(sigma2))
    sigma2_hat[i] = (1/n)*sum((x-mean(x))^2)
  }
  hist(sigma2_hat, probability = TRUE, main = paste("n = ", n), xlab = expression(widehat(sig
```

```r
  curve(dist_sigma2_hat(x), add = TRUE, col = "red", lwd = 2)
  points(sigma2, 0, pch = 19, col = "red", cex = 1.5)
  points(mean(sigma2_hat), 0, col = "blue", lwd = 3,
         cex = 2)
}
```

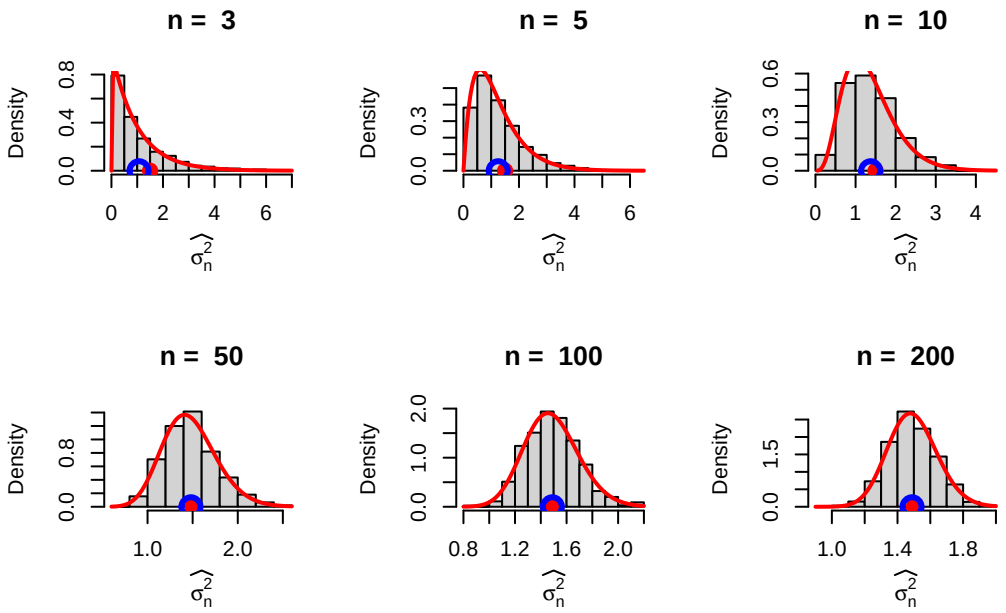


Figure 15: The sampling distribution of the MLE of σ^2 is obtained through simulation for different sample sizes. The exact sampling distribution is shown using the red curve. The red dot indicates the true value of σ^2, wheres the blue circle indicates $\frac{1}{M}\sum_{m=1}^{M}\widehat{\sigma_n^2}^{(m)}$, which is the approximate expectation of $\widehat{\sigma_n^2}$ based on M replications. As the sample size increases, the circle engulfs the red dot which is a signature of asymptotic unbiasedness. In other words, it ensures that Bias $\left(\widehat{\sigma_n^2}, \sigma^2\right) \to 0$ as $n \to \infty$.

By performing the above exercises for different choices of n, we observed that the this sampling distribution is valid for every n. However, as n increases, the shapes of the sampling distributions eventually tends to the normal distribution. We need to give some reasoning for this which is intuitively appealing.

Let us explore the connections between $U_n = \frac{n-1}{n}S_n^2$ and $V_n = \widehat{\sigma_n}^2 = \frac{\sigma^2}{n} \times U_n$ and their approximation by the normal distribution for large n. Suppose that we are given the fact that for every n, $U_n \sim \chi^2_{n-1}$ distribution. The PDF of V_n has been obtained by using the transformation formula. The shapes of the PDF of V_n in fact shows a normally distributed

behavior for large n values. Since, the chi squared distribution belongs to the $\mathcal{G}(\cdot,\cdot)$ family, V_n is a constant multiple of a $\mathcal{G}\left(\alpha = \frac{n-1}{2}, \beta = 2\right) \equiv \chi^2_{n-1}$.

Therefore, if we can show that for large n, χ^2_n distribution can be well approximated by the normal distribution, we are done. In the following we show that for large n, χ^2_n PDF is well approximated by the normal distribution with mean n and variance $2n$. We investigate the MGF of $\frac{X-n}{\sqrt{2n}}$ and see that it is approximately $e^{\frac{t^2}{3}}$ for large n.

$$\log\left[M_{\frac{X-n}{\sqrt{2n}}}(t)\right] = -\sqrt{\frac{n}{2}}t + \frac{n}{2}\log\left(1 - \frac{t}{\sqrt{\frac{n}{2}}}\right) = \frac{t^2}{2} + \mathcal{O}\left(\frac{1}{n^{3/2}}\right).$$

Therefore,

$$\frac{X-n}{\sqrt{2n}} \overset{a}{\sim} \mathcal{N}(0,1) \implies X \overset{a}{\sim} \mathcal{N}(n, 2n).$$

Therefore $U_n \sim \mathcal{N}(n-1, 2(n-1))$ for large n. Hence,

$$V_n = \widehat{\sigma^2_n} = \frac{\sigma^2}{n} \times U_n \overset{n \text{ large}}{\sim} \mathcal{N}\left(\frac{(n-1)\sigma^2}{n}, \frac{2(n-1)\sigma^4}{n^2}\right).$$

In the following, R Codes, the exact sampling distribution of $\widehat{\sigma^2_n}$ and the normal approximation is shown graphically and the approximations are remarkable accurate for large n values.

```r
n_vals = c(3,5,10,30,50,100)
par(mfrow = c(2,3))
for(n in n_vals){
  lower_lim = sigma2*(n-1)/n- 5*sqrt(sigma2^2*2*(n-1)/n^2)
  upper_lim = sigma2*(n-1)/n+ 5*sqrt(sigma2^2*2*(n-1)/n^2)

  curve(dist_sigma2_hat(x), col = "red", lwd = 2, lower_lim, upper_lim, ylab = "f(x)")

  curve(dnorm(x,mean = sigma2*(n-1)/n, sd = sqrt(sigma2^2*2*(n-1)/n^2)),
        add = TRUE, lty = 2, col = "blue", lwd = 2)
}
```

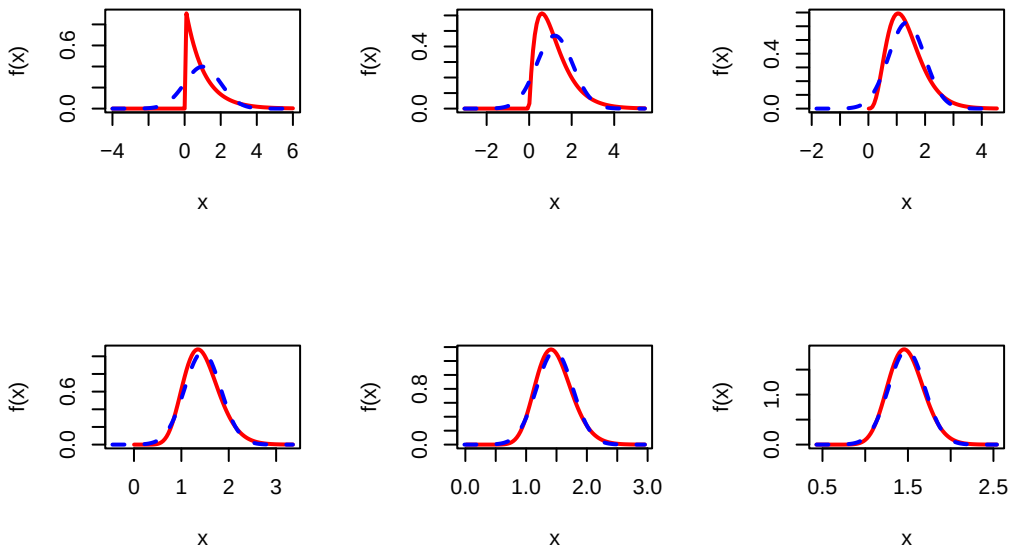


Figure 16: The exact sampling distribution of the MLE of $\widehat{\sigma_n^2}$ is shown for different choices of n (red curve). The normal approximation with mean $\frac{(n-1)\sigma^2}{n}$ variance $\frac{2(n-1)\sigma^4}{n^2}$ is shown using the blue dotted line. It is evident that as the sample size increases, the exact sampling distribution can be well approximated by the normal distribution.

Connection between the Hessian and Fisher Information

In the examples discussed above, the MLEs have always appeared to be approximately normal at least for large n and centered about the true value of the parameter. However, we did not explore yet the computation of the variance of the MLEs. In fact, some analytical computation for the variance can be carried out and established as well by computer simulation. We consider some definition first.

! Score Function and Fisher Information

If $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ be a random sample of size n from the PDF (PMF) $f(x|\theta)$, $\theta \in \Theta$.
The Score function is defined as

$$S(\mathbf{X};\theta) = \frac{\partial \log f(\mathbf{X}|\theta)}{\partial \theta}.$$

It can be easily shown that

$$\mathrm{E}_\theta\left[S(\mathbf{X};\theta)\right] = 0.$$

The Fisher Information is defined as

$$I_n(\theta) = Var_\theta\left(\sum_{i=1}^{n} S\left(X_i;\theta\right)\right) = \sum_{i=1}^{n} Var_\theta\left(S\left(X_i;\theta\right)\right) = \sum_{i=1}^{n} \mathrm{E}_\theta\left(S\left(X_i;\theta\right)\right)^2.$$

The Fisher Information can also be expressed as

$$I_n(\theta) = -\mathrm{E}_\theta\left(\frac{\partial^2 \log f\left(\mathbf{X}|\theta\right)}{\partial\theta^2}\right) = -\int_{\mathbb{R}^n}\left(\frac{\partial^2 \log f\left(\mathbf{x}|\theta\right)}{\partial\theta^2}\right) f(\mathbf{x}|\,)d\mathbf{x}.$$

A natural question arises, how to interpret the expectation of the score function. Here is an interpretation in terms of simulation.

- Fix parameter $\theta = \theta_0$
- Fix the sample size n
- Fix the number of replications M
- For each $m \in \{1, 2, \ldots, M\}$
  - Simulate $\mathbf{X} = (X_1, \ldots, X_n) \sim f(x|\theta_0)$.
  - Compute $S(\mathbf{X};\theta_0) = S_m$ (say)
- $\frac{1}{M}\sum_{m=1}^{M} S_m \overset{M\to\infty}{\approx} \mathrm{E}\left(S\left(\mathbf{X};\theta_0\right)\right) = 0$

Visualization of the Score function

In the following, we visualize the score function for the Poisson distribution. We simulate multiple sets of random sample of size $n = 10$ from the Poisson(λ_0) distribution and plot the score function as a function of λ.

```
par(mfrow = c(1,1))
lambda_0 = 3                        # true value
n = 10                              # sample size
x = rpois(n = n, lambda = lambda_0)
score_poisson = function(lambda){
  -n + sum(x)/lambda
}
lambda_vals = seq(1, 5, by = 0.01)
score_vals = numeric(length = length(lambda_vals))
for(i in 1:length(lambda_vals)){
  score_vals[i] = score_poisson(lambda = lambda_vals[i])
}
plot(lambda_vals, score_vals, type = "l",
     col = "grey", lwd = 2, xlab = expression(lambda),
```

```
     ylab = expression(S(bold(X),lambda)))
points(lambda_0, 0, pch = 19, col = "red", cex = 1.4)
abline(h = 0, lwd = 2, col = "blue", lty = 2)
points(mean(x), 0, col = "blue", cex = 2, lwd = 2,
       pch = 19)
```

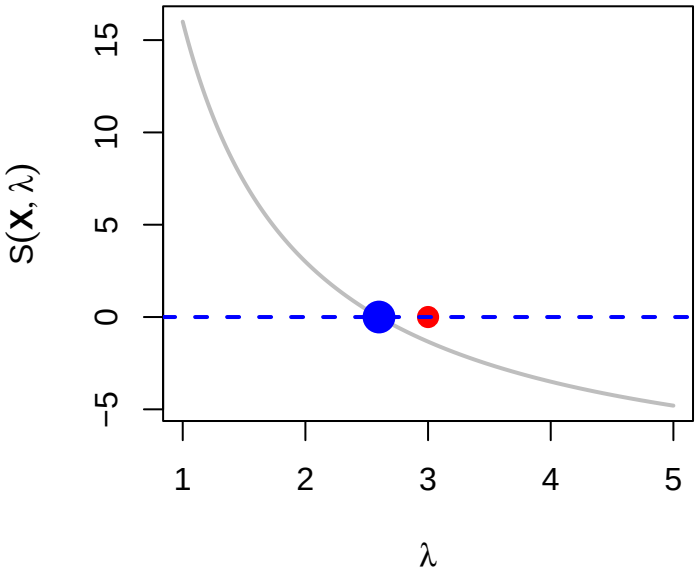


Figure 17: The shape of the score function is shown for a simulated sample of size $n = 10$ from the Poisson($\lambda_0 = 3$) distribution. The blue dot represents the MLE of λ and the red dot is the true value.

Let us repeat the above process and plot the score functions in a single plot.

```
n = 10
M = 50
lambda_0 = 3
for (i in 1:M) {
  data = rpois(n = n, lambda = lambda_0)
  if(i == 1)
    curve(-n+sum(data)/x, 1, 5, col = "grey", lwd = 2,
          xlab = expression(lambda), ylab = expression(S(bold(X),lambda)))
  else
    curve(-n+sum(data)/x, add= TRUE, col = "grey",
          lwd = 2)
  points(mean(data), 0, col = "blue", cex = 1, lwd = 2,
       pch = 19)
```

```
}
points(lambda_0, 0, pch = 19, col = "red", cex = 1.4)
abline(h = 0, lwd = 2, col = "blue", lty = 2)
abline(v = lambda_0, col = "magenta", lty = 2, lwd = 3)
```

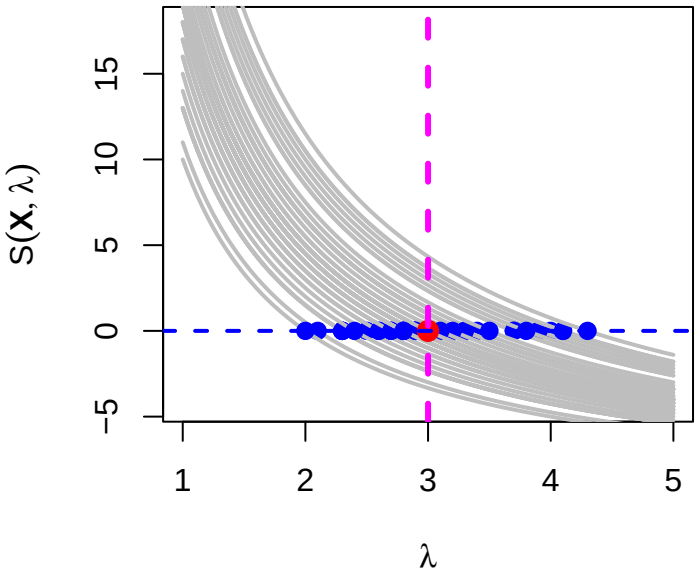


Figure 18: Shapes of the score function for different sets of samples each of size $n = 10$. The blue dot represents the MLE of of λ which is $\overline{X_n}$. The red dot is the true value. The vertical magenta line at λ_0 intersects the score function at $S(\mathbf{X}, \lambda_0)$ and the average of these values would be close to zero. $\frac{1}{M}\sum_{m=1}^{M} S(\mathbf{X}^{(m)}, \lambda_0) \approx \mathrm{E}S(\mathbf{X}, \lambda_0) = 0$.

Let us do the same experiment for the Cauchy distribution whose score function is given by

$$S(\mathbf{X}; \mu) = 2\sum_{i=1}^{n} \frac{(x_i - \mu)}{[1 + (x_i - \mu)^2]}$$

```
par(mfrow = c(1,1))
mu_0 = 3            # true value
n = 10
x = rcauchy(n = n, location = mu_0)
score_cauchy = function(mu){
  2*sum((x-mu)/(1+(x-mu)^2))
}
mu_vals = seq(1, 5, by = 0.01)
score_vals = numeric(length = length(mu_vals))
for(i in 1:length(mu_vals)){
```

```r
  score_vals[i] = score_cauchy(mu = mu_vals[i])
}
plot(mu_vals, score_vals, type = "l",
     col = "grey", lwd = 2, xlab = expression(mu),
     ylab = expression(S(bold(X),mu)))
points(mu_0, 0, pch = 19, col = "red", cex = 1.4)
abline(h = 0, lwd = 2, col = "blue", lty = 2)
```

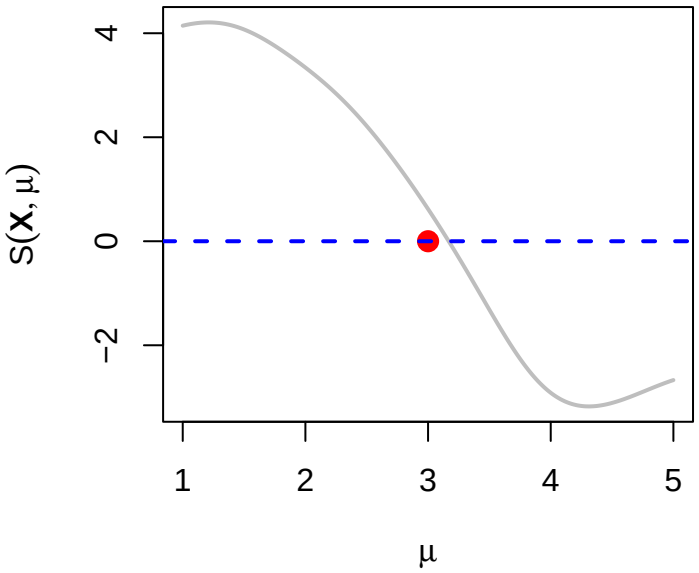


Figure 19: The shapes of the score function for different samples of size $n = 10$. The true value of $\mu = \mu_0 = 3$ has been used for simulation purpose (marked as red dot). Reader is encouraged to execute the following codes multiple times and check the shapes for different choices of μ_0 and sample size n.

Asymptotic distribution of the MLE

Under certain regularity conditions for the studied population distribution $f(x|\theta)$, if $\widehat{\theta_n}$ be the MLE of θ and $I_n(\theta)$ be the Fisher Information. Then $\text{SE}\left(\widehat{\theta_n}\right) = \sqrt{Var\left(\widehat{\theta_n}\right)} \approx \sqrt{\frac{1}{I_n(\theta)}}$. In addition

$$\frac{\widehat{\theta_n} - \theta}{\text{SE}\left(\widehat{\theta_n}\right)} \to \mathcal{N}(0,1), \quad \text{in distribution.}$$

In addition, $\widehat{\text{SE}}\left(\widehat{\theta_n}\right) = \sqrt{\widehat{Var}\left(\widehat{\theta_n}\right)} \approx \sqrt{I_n\left(\widehat{\theta_n}\right)^{-1}}$ and

$$\frac{\widehat{\theta_n} - \theta}{\widehat{\text{SE}}\left(\widehat{\theta_n}\right)} \to \mathcal{N}(0,1), \quad \text{in distribution.}$$

In the following simulation experiment, we verify the above results for the Poisson distribution. The MLE for the parameter λ is $\widehat{\lambda_n} = \overline{X_n}$. $\text{SE}\left(\widehat{\lambda_n}\right) = \sqrt{\frac{\lambda}{n}}$ and $\widehat{\text{SE}}\left(\widehat{\lambda_n}\right) = \sqrt{\frac{\widehat{\lambda_n}}{n}} = \sqrt{\frac{\overline{X_n}}{n}}$.

```r
par(mfrow = c(2,3))
lambda_0 = 3
M = 1000
n_vals = c(3, 5, 10, 35, 50, 100)
for(n in n_vals){
  U = numeric(length = M)
  V = numeric(length = M)
  for(i in 1:M){
    x = rpois(n = n, lambda = lambda_0)
    U[i] = (mean(x)-lambda_0)/(sqrt(lambda_0/n))
    V[i] = (mean(x)-lambda_0)/(sqrt(mean(x)/n))
  }
  hist(U, probability = TRUE, main = paste("n = ", n), breaks = 30,
       xlab = expression(U[n]))
  curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
  hist(V, probability = TRUE, main = paste("n = ", n), breaks = 30,
       xlab = expression(V[n]))
  curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
}
```

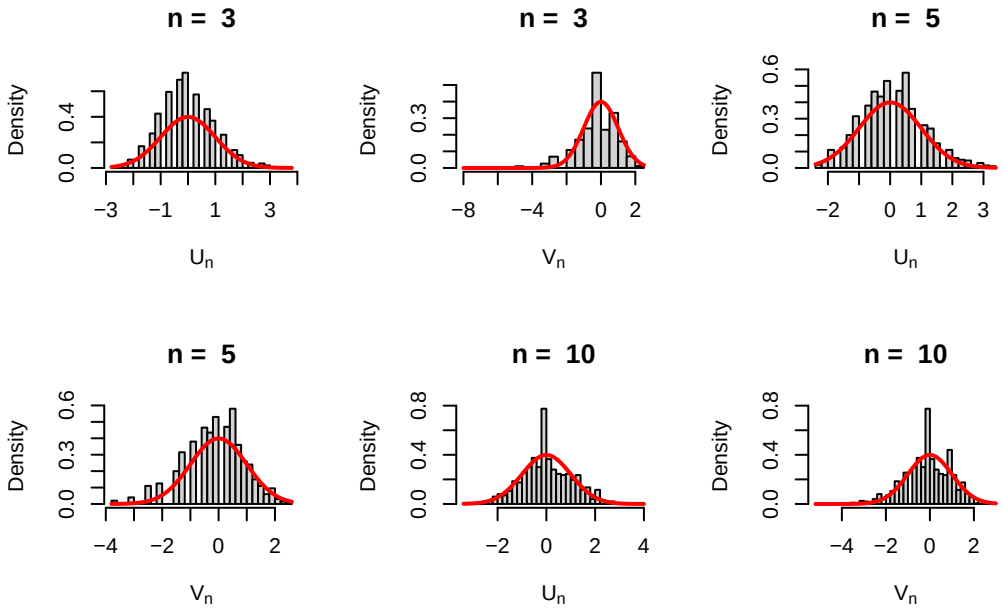


Figure 20: As the sample size increases the standardized estimators of the MLE of λ is approximately $\mathcal{N}(0,1)$ distributed.

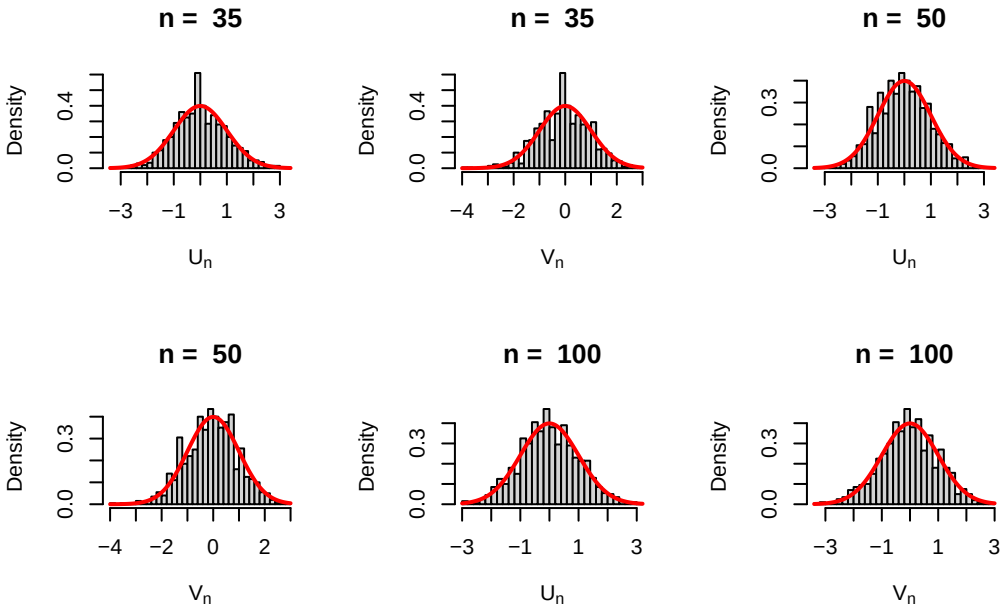


Figure 21: As the sample size increases the standardized estimators of the MLE of λ is approximately $\mathcal{N}(0,1)$ distributed.

Multiparameter setting and Score function

Suppose that $X_1, X_2, \dots, X_n$ be a random sample of size n from the population PDF (PMF) $f(x|\theta)$ and $\theta \in \Theta \subseteq \mathbb{R}^p$ and $p > 1$. Let $\theta = (\theta_1, \dots, \theta_p)$ and the corresponding MLE is expressed as

$$\widehat{\theta_n} = \left(\widehat{\theta_1}, \widehat{\theta_2}, \dots, \widehat{\theta_p}\right).$$

MLE is computed by solving the system of p equations given by

$$\frac{\partial l_n(\theta)}{\partial \theta_j} = 0, j \in \{1, 2, \dots, p\}$$

We define

$$H_{jj} = \frac{\partial^2 l_n(\theta)}{\partial \theta_j^2}, \quad \text{and} \quad H_{jk} = \frac{\partial^2 l_n(\theta)}{\partial \theta_j \partial \theta_k}$$

for $i, j \in \{1, 2, \dots, p\}$. Similar to the one variance case, here we will have the Fisher Information Matrix defined as

$$I_n(\theta) = \begin{bmatrix} \mathrm{E}_\theta(H_{11}) & \mathrm{E}_\theta(H_{12}) & \cdots & \mathrm{E}_\theta(H_{1p}) \\ \mathrm{E}_\theta(H_{21}) & \mathrm{E}_\theta(H_{22}) & \cdots & \mathrm{E}_\theta(H_{2p}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{E}_\theta(H_{p1}) & \mathrm{E}_\theta(H_{p2}) & \cdots & \mathrm{E}_\theta(H_{pp}) \end{bmatrix}$$

When the partial derivatives are evaluated at the MLE $\widehat{\theta_n}$, then we obtain the `Observed` Fisher Information Matrix.

! Fisher Information Matrix and Multivariate Normality of MLE

Let $J_n = I_n^{-1}(\theta)$, the inverse of the expected Fisher Information Matrix. Under appropriate regularity conditions

$$\sqrt{n}\left(\widehat{\theta_n} - \theta_{p\times 1}\right) \overset{n\to\infty}{\sim} \mathrm{MVN}_p\left(0, J_n(\theta)\right).$$

In particular

$$\frac{\sqrt{n}\left(\widehat{\theta_j} - \theta_j\right)}{\widehat{\mathrm{SE}}(\widehat{\theta_j})} \overset{n\to\infty}{\sim} \mathcal{N}(0,1)$$

where $\widehat{\mathrm{SE}}(\widehat{\theta_j}) = J_n(j,j)$, the jth diagonal entry of J_n, evaluated at $\hat{\theta}$. Also

$$\mathrm{Cov}\left(\widehat{\theta_j}, \widehat{\theta_k}\right) \approx J_n(j,k).$$

The famous regularity conditions

Throughout the discussions in the statistical computing lectures, we have observed only nice things about the MLE of model parameter(s) θ, two prominent properties are: (a) MLEs are consistent estimators, that means as the sample size goes to infinity, the sampling distribution of the MLEs will be highly concentrated at the true parameter value θ, or in other words, the limiting distribution of the MLE is degenerate at the true value. In fact, this property is true for any probability density (or mass) functions $f(x|\theta), \theta \in \Theta$ which belongs to the family satisfying the following four conditions:

- The random sample $X_1, X_2, ..., X_n$ are independent and identically distributed (IID) following $f(x|\theta)$.
- The parameter is identifiable, that is, if $\theta_1 \neq \theta_2$, then $f(x|\theta_1) \neq f(x|\theta_2)$.
- For every θ, the density functions $f(x|\theta)$, have common support, and $f(x|\theta)$ is differentiable in θ.
- The parameter space Ω contains an open set ω of which the true parameter value θ_0 is an interior point.

In many examples in this document, we also observed that MLEs appeared to be asymptotically normal and asymptotically efficient as well. In addition to the above four conditions, the following two conditions are needed to ensure the above two properties.

- For every $x \in \chi$, support of the PDF (PMF) $f(x|\theta)$ is three times differentiable with respect to θ, the third derivative is continuous in θ, and $\int f(x|\theta)dx$ can be differentiated three times under the integral sign.
- For any $\theta_0 \in \Omega$, there exists a positive number c and a function $M(x)$ (both of which depend on θ_0) such that

$$\left|\frac{\partial^3}{\partial\theta^3}\log f(x|\theta)\right| \leq M(x) \quad \text{for all } x \in \chi, \quad \theta_0 - c < \theta < \theta_0 + c,$$

  with $\mathrm{E}_\theta[M(X)] < \infty$.

These conditions are typically known as the regularity conditions.

Exception: MLE is not asymptotically normal Uniform$(0, \theta)$

Some more examples

```r
library(datasets)
plot(datasets::JohnsonJohnson)
hist(JohnsonJohnson, xlab = "Quarterly earnings (dollars)",
     main = "", probability = TRUE)

n = length(JohnsonJohnson)
Lik = function(lambda){
  (lambda^n) * exp(-lambda * sum(JohnsonJohnson))

}
lambda_vals = seq(0.01, 0.4, by = 0.001)
Lik_vals = numeric(length = length(lambda_vals))
for (i in 1:length(lambda_vals)) {
  Lik_vals[i] = Lik(lambda_vals[i])
}

lambda_hat = n/sum(JohnsonJohnson)      # MLE

par(mfrow = c(1,2))
plot(lambda_vals, Lik_vals, pch = 19, col = "red",
     xlab = expression(lambda), ylab = expression(L(lambda)),
     type = "p", lwd = 2)
points(lambda_hat, 0, pch = 19, col = "blue", cex = 1.2)

plot(lambda_vals, log(Lik_vals), pch = 19, col = "red",
     xlab = expression(lambda), ylab = expression(l(lambda)),
     type = "p", lwd = 2)
abline(v = lambda_hat, col = "blue", lwd = 2, lty = 2)

par(mfrow = c(1,1))
hist(JohnsonJohnson, probability = TRUE,
     xlab = "Quarterly earnings (dollars)",
     main = "",)
curve(lambda_hat*exp(-lambda_hat*x), add = TRUE,
      col = "red", lwd = 2)

f = function(x){
  lambda_hat*exp(- lambda_hat*x)*(x>0)
}

# P(X>18)
exp(-18*lambda_hat)
```

```r
integrate(f, lower = 18, upper = Inf)
```

Exploratory Data Analysis

Introduction

This section is primarily devoted to the exploration of various datasets using R. Students explore different datasets available in the `datasets` package in R, giving them exposure to real-world data. In the previous few lectures, we primarily studied various probability distributions and their properties, which are theoretical in nature. In the classroom, students explore different datasets, discuss them among themselves, and read the help files to gain detailed information about each dataset. Some examples of datasets explored by the students are provided here. Some testing problems have also been discussed.

The `ChickWeight` dataset

The `datasets` is a package in R, part of the `base` R packages. Let us get into real data set. We consider the `ChickWeight` data set available in R. The body weights of the chicks were measured at birth and every second day thereafter until day 20. They were also measured on day 21. There were four groups on chicks on different protein diets.

```
# datasets::ChickWeight
head(ChickWeight)
```

```
  weight Time Chick Diet
1     42    0     1    1
2     51    2     1    1
3     59    4     1    1
4     64    6     1    1
5     76    8     1    1
6     93   10     1    1
```

```
nrow(ChickWeight)
```

```
[1] 578
```

```
# help("ChickWeight")
class(ChickWeight)
```

```
[1] "nfnGroupedData" "nfGroupedData"  "groupedData"    "data.frame"
```

```
names(ChickWeight)
```

```
[1] "weight" "Time"   "Chick"  "Diet"
```

```
summary(ChickWeight)
```

```
     weight           Time           Chick     Diet
 Min.   : 35.0   Min.   : 0.00   13     : 12   1:220
 1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   2:120
 Median :103.0   Median :10.00   20     : 12   3:120
 Mean   :121.8   Mean   :10.72   10     : 12   4:118
 3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
 Max.   :373.0   Max.   :21.00   19     : 12
                                 (Other):506
```

The function `class` is important to understand different data types. In data analysis, this function is often useful to check for the correct input type to specific functions in R.

```
class(ChickWeight$weight)
```

```
[1] "numeric"
```

```
class(ChickWeight$Time)
```

```
[1] "numeric"
```

```
class(ChickWeight$Chick)
```

```
[1] "ordered" "factor"
```

```r
class(ChickWeight$Diet)
```

```
[1] "factor"
```

Checking for missing values

```r
head(is.na.data.frame(ChickWeight))   # check for missing values
```

```
  weight  Time Chick  Diet
1  FALSE FALSE FALSE FALSE
2  FALSE FALSE FALSE FALSE
3  FALSE FALSE FALSE FALSE
4  FALSE FALSE FALSE FALSE
5  FALSE FALSE FALSE FALSE
6  FALSE FALSE FALSE FALSE
```

```r
sum(is.na.data.frame(ChickWeight))
```

```
[1] 0
```

`is.na` function

```r
x = c(1, 2, 3, 4, NA, 5, 4:1000)
print(x)
is.na(x)
sum(is.na(x))
which(is.na(x) == TRUE)
```

Reading columns in a data.frame

```r
ChickWeight[ ,"Chick"]
```

```
  [1] 1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  2  3
 [26] 3  3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4  4  4  4  5  5
 [51] 5  5  5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6  6  6  6  6  7  7  7
 [76] 7  7  7  7  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9
[101] 9  9  9  9  9  9  9  10 10 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11
[126] 11 11 11 11 11 11 12 12 12 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13
[151] 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 14 14 15 15 15 15 15 15 15 15
[176] 16 16 16 16 16 16 16 17 17 17 17 17 17 17 17 17 17 17 17 18 18 19 19 19 19
[201] 19 19 19 19 19 19 19 19 20 20 20 20 20 20 20 20 20 20 20 20 21 21 21 21 21
[226] 21 21 21 21 21 21 21 22 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23
[251] 23 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25
[276] 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27 27
[301] 27 27 27 27 28 28 28 28 28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29
[326] 29 29 29 30 30 30 30 30 30 30 30 30 30 30 30 31 31 31 31 31 31 31 31 31 31
[351] 31 31 32 32 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33 33 33 33 33 33
[376] 33 34 34 34 34 34 34 34 34 34 34 34 34 35 35 35 35 35 35 35 35 35 35 35 35
[401] 36 36 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 37 37 38
[426] 38 38 38 38 38 38 38 38 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 40 40
[451] 40 40 40 40 40 40 40 40 40 40 41 41 41 41 41 41 41 41 41 41 41 41 42 42 42
[476] 42 42 42 42 42 42 42 42 42 43 43 43 43 43 43 43 43 43 43 43 43 44 44 44 44
[501] 44 44 44 44 44 44 45 45 45 45 45 45 45 45 45 45 45 45 46 46 46 46 46 46 46
[526] 46 46 46 46 46 47 47 47 47 47 47 47 47 47 47 47 47 48 48 48 48 48 48 48 48
[551] 48 48 48 48 49 49 49 49 49 49 49 49 49 49 49 49 50 50 50 50 50 50 50 50 50
[576] 50 50 50
50 Levels: 18 < 16 < 15 < 13 < 9 < 20 < 10 < 8 < 17 < 19 < 4 < 6 < 11 < ... < 48
```

```
1 ChickWeight[ ,"Time"]
```

```
  [1]  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0
 [26]  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2
 [51]  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4
 [76]  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20  0  2  4  6  8
[101] 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10
[126] 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12
[151] 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14
[176]  0  2  4  6  8 10 12  0  2  4  6  8 10 12 14 16 18 20 21  0  2  0  2  4  6
[201]  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8
[226] 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10
[251] 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12
[276] 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14
[301] 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16
[326] 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18
```

```
 [351] 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20
 [376] 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21
 [401]  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0
 [426]  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2
 [451]  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4
 [476]  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6
 [501]  8 10 12 14 16 18  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12
 [526] 14 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14
 [551] 16 18 20 21  0  2  4  6  8 10 12 14 16 18 20 21  0  2  4  6  8 10 12 14 16
 [576] 18 20 21
```

```
1 ChickWeight[ ,"weight"]
```

```
  [1]  42  51  59  64  76  93 106 125 149 171 199 205  40  49  58  72  84 103
 [19] 122 138 162 187 209 215  43  39  55  67  84  99 115 138 163 187 198 202
 [37]  42  49  56  67  74  87 102 108 136 154 160 157  41  42  48  60  79 106
 [55] 141 164 197 199 220 223  41  49  59  74  97 124 141 148 155 160 160 157
 [73]  41  49  57  71  89 112 146 174 218 250 288 305  42  50  61  71  84  93
 [91] 110 116 126 134 125  42  51  59  68  85  96  90  92  93 100 100  98  41
[109]  44  52  63  74  81  89  96 101 112 120 124  43  51  63  84 112 139 168
[127] 177 182 184 181 175  41  49  56  62  72  88 119 135 162 185 195 205  41
[145]  48  53  60  65  67  71  70  71  81  91  96  41  49  62  79 101 128 164
[163] 192 227 248 259 266  41  49  56  64  68  68  67  68  41  45  49  51  57
[181]  51  54  42  51  61  72  83  89  98 103 113 123 133 142  39  35  43  48
[199]  55  62  65  71  82  88 106 120 144 157  41  47  54  58  65  73  77  89
[217]  98 107 115 117  40  50  62  86 125 163 217 240 275 307 318 331  41  55
[235]  64  77  90  95 108 111 131 148 164 167  43  52  61  73  90 103 127 135
[253] 145 163 170 175  42  52  58  74  66  68  70  71  72  72  76  74  40  49
[271]  62  78 102 124 146 164 197 231 259 265  42  48  57  74  93 114 136 147
[289] 169 205 236 251  39  46  58  73  87 100 115 123 144 163 185 192  39  46
[307]  58  73  92 114 145 156 184 207 212 233  39  48  59  74  87 106 134 150
[325] 187 230 279 309  42  48  59  72  85  98 115 122 143 151 157 150  42  53
[343]  62  73  85 102 123 138 170 204 235 256  41  49  65  82 107 129 159 179
[361] 221 263 291 305  39  50  63  77  96 111 137 144 151 146 156 147  41  49
[379]  63  85 107 134 164 186 235 294 327 341  41  53  64  87 123 158 201 238
[397] 287 332 361 373  39  48  61  76  98 116 145 166 198 227 225 220  41  48
[415]  56  68  80  83 103 112 135 157 169 178  41  49  61  74  98 109 128 154
[433] 192 232 280 290  42  50  61  78  89 109 130 146 170 214 250 272  41  55
[451]  66  79 101 120 154 182 215 262 295 321  42  51  66  85 103 124 155 153
[469] 175 184 199 204  42  49  63  84 103 126 160 174 204 234 269 281  42  55
[487]  69  96 131 157 184 188 197 198 199 200  42  51  65  86 103 118 127 138
[505] 145 146  41  50  61  78  98 117 135 141 147 174 197 196  40  52  62  82
```

```
[523] 101 120 144 156 173 210 231 238  41  53  66  79 100 123 148 157 168 185
[541] 210 205  39  50  62  80 104 125 154 170 222 261 303 322  40  53  64  85
[559] 108 128 152 166 184 203 233 237  41  54  67  84 105 122 155 175 205 234
[577] 264 264
```

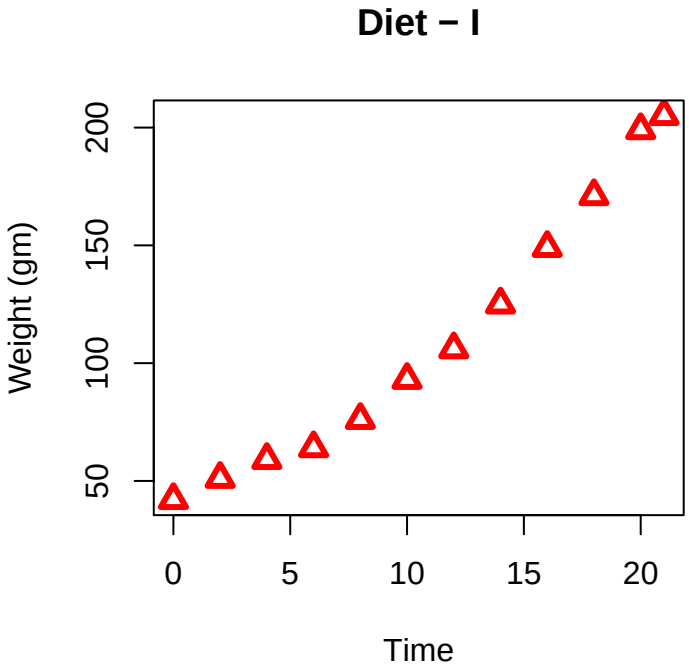
```
ChickWeight[ ,"Diet"]
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
[223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[334] 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[371] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[408] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[445] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[482] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[519] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
[556] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
Levels: 1 2 3 4
```

Subsetting of data

- We will collect only the weight profile of the chick (coded as 1) and given the diet type 1. The following code will help us to obtain the data and plot the profile.

```
w = ChickWeight[1:12 ,"weight"]
t = ChickWeight[1:12 ,"Time"]
plot(t,w, col = "red", pch = 2,
     xlab = "Time", ylab = "Weight (gm)",
     main = "Diet - I", cex = 1.3, lwd = 3)
```



We can also separate the data for different types of diet. The function `subset` is useful in doing so.

```
diet_1 = subset(ChickWeight, Diet == 1, select = c(weight, Time, Chick))
diet_2 = subset(ChickWeight, Diet == 2, select = c(weight, Time, Chick))
diet_3 = subset(ChickWeight, Diet == 3, select = c(weight, Time, Chick))
diet_4 = subset(ChickWeight, Diet == 4, select = c(weight, Time, Chick))
dim(diet_1)
```

```
[1] 220   3
```

```
dim(diet_2)
```

```
[1] 120   3
```

```
dim(diet_3)
```

```
[1] 120   3
```

```
dim(diet_4)
```

```
[1] 118   3
```

Understanding the summary of the data is an important part of any exploratory data analysis.

```
summary(ChickWeight)
```

```
     weight           Time           Chick     Diet
 Min.   : 35.0   Min.   : 0.00   13     : 12   1:220
 1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   2:120
 Median :103.0   Median :10.00   20     : 12   3:120
 Mean   :121.8   Mean   :10.72   10     : 12   4:118
 3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
 Max.   :373.0   Max.   :21.00   19     : 12
                                 (Other):506
```

```
summary(diet_1)
```

```
     weight           Time           Chick
 Min.   : 35.00   Min.   : 0.00   13     : 12
 1st Qu.: 57.75   1st Qu.: 4.00   9      : 12
 Median : 88.00   Median :10.00   20     : 12
 Mean   :102.65   Mean   :10.48   10     : 12
 3rd Qu.:136.50   3rd Qu.:16.00   17     : 12
 Max.   :305.00   Max.   :21.00   19     : 12
                                  (Other):148
```

```
summary(diet_2)
```

```
     weight           Time           Chick
 Min.   : 39.0   Min.   : 0.00   24     :12
 1st Qu.: 65.5   1st Qu.: 5.50   30     :12
 Median :104.5   Median :11.00   22     :12
 Mean   :122.6   Mean   :10.92   23     :12
 3rd Qu.:163.0   3rd Qu.:16.50   27     :12
 Max.   :331.0   Max.   :21.00   28     :12
                                 (Other):48
```

```
summary(diet_3)
```

```
     weight           Time           Chick    
 Min.   : 39.0   Min.   : 0.00   33     :12  
 1st Qu.: 67.5   1st Qu.: 5.50   37     :12  
 Median :125.5   Median :11.00   36     :12  
 Mean   :142.9   Mean   :10.92   31     :12  
 3rd Qu.:198.8   3rd Qu.:16.50   39     :12  
 Max.   :373.0   Max.   :21.00   38     :12  
                                 (Other):48  
```

```
summary(diet_4)
```

```
     weight           Time           Chick    
 Min.   : 39.00   Min.   : 0.00   45     :12  
 1st Qu.: 71.25   1st Qu.: 4.50   43     :12  
 Median :129.50   Median :10.00   41     :12  
 Mean   :135.26   Mean   :10.75   47     :12  
 3rd Qu.:184.75   3rd Qu.:16.00   49     :12  
 Max.   :322.00   Max.   :21.00   46     :12  
                                  (Other):46  
```

We can also have some first few or last few observations from the data set.

```
head(diet_1, n =13)
```

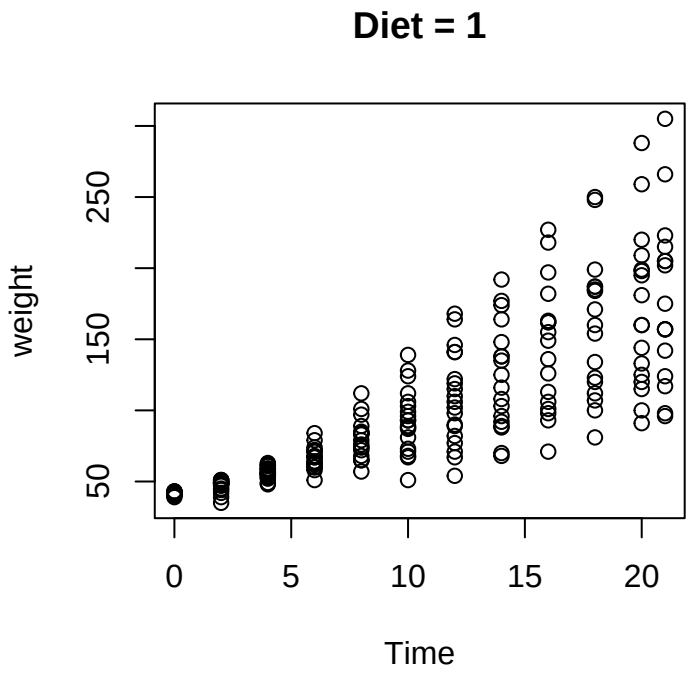
```
   weight Time Chick
1      42    0     1
2      51    2     1
3      59    4     1
4      64    6     1
5      76    8     1
6      93   10     1
7     106   12     1
8     125   14     1
9     149   16     1
10    171   18     1
11    199   20     1
12    205   21     1
13     40    0     2
```

```r
tail(diet_1, n = 10)
```

```
    weight Time Chick
211     54    4    20
212     58    6    20
213     65    8    20
214     73   10    20
215     77   12    20
216     89   14    20
217     98   16    20
218    107   18    20
219    115   20    20
220    117   21    20
```

basic plotting function

```r
plot(weight ~ Time, data = diet_1, main = "Diet = 1")
```

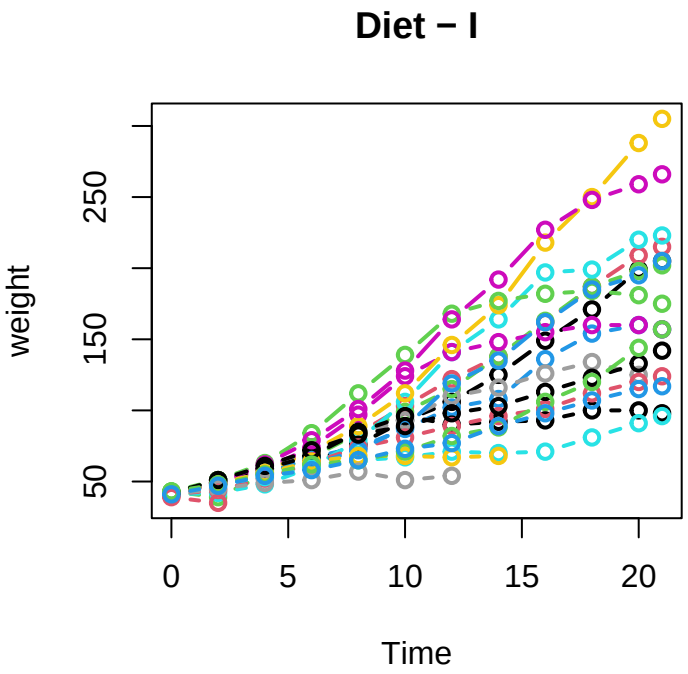


```r
head(diet_1$Chick)
```

```
[1] 1 1 1 1 1 1
50 Levels: 18 < 16 < 15 < 13 < 9 < 20 < 10 < 8 < 17 < 19 < 4 < 6 < 11 < ... < 48
```

In the above plot, growth profiles of different chicks can not be identified separately. We can draw the growth profiles of each chick separately. We can have multiple growth profiles also in the same plot. We can create a single growth profile first (for chick = 1) and others can be added using the `lines` or `points` commands.

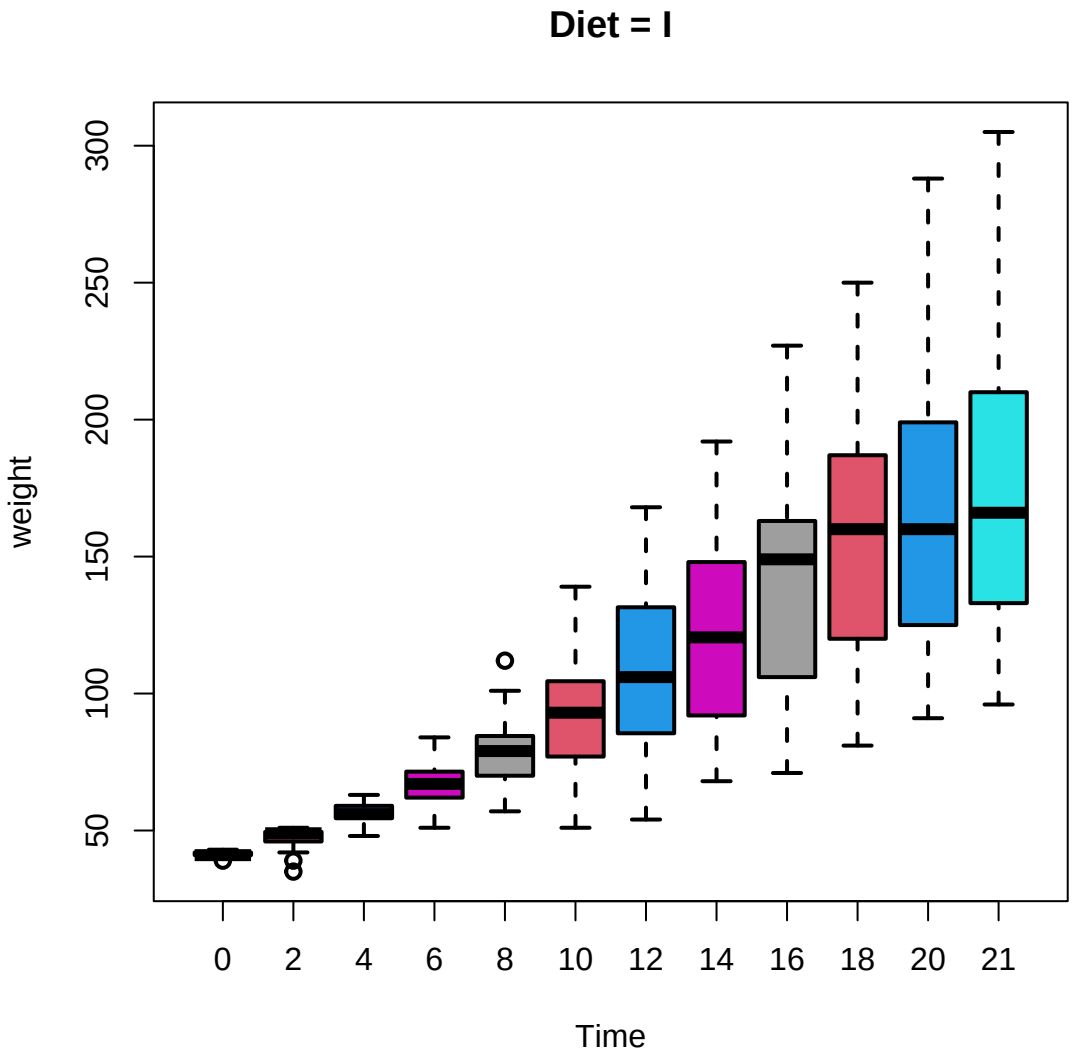
```r
par(mfrow = c(1,1))
plot(weight ~ Time, data = subset(diet_1, Chick == 1),
     col = 1, lwd = 2, type = "b",
     ylim = c(min(diet_1$weight), max(diet_1$weight)),
     main = "Diet - I")
for(i in 2:20){
  lines(weight ~ Time, data = subset(diet_1, Chick == i), col = i,
        lwd = 2, type = "b")
}
```



It is evident from the picture that the variance associated with measurements of weights increases as the time increases. Using the box plot, we can get a more clearer understanding

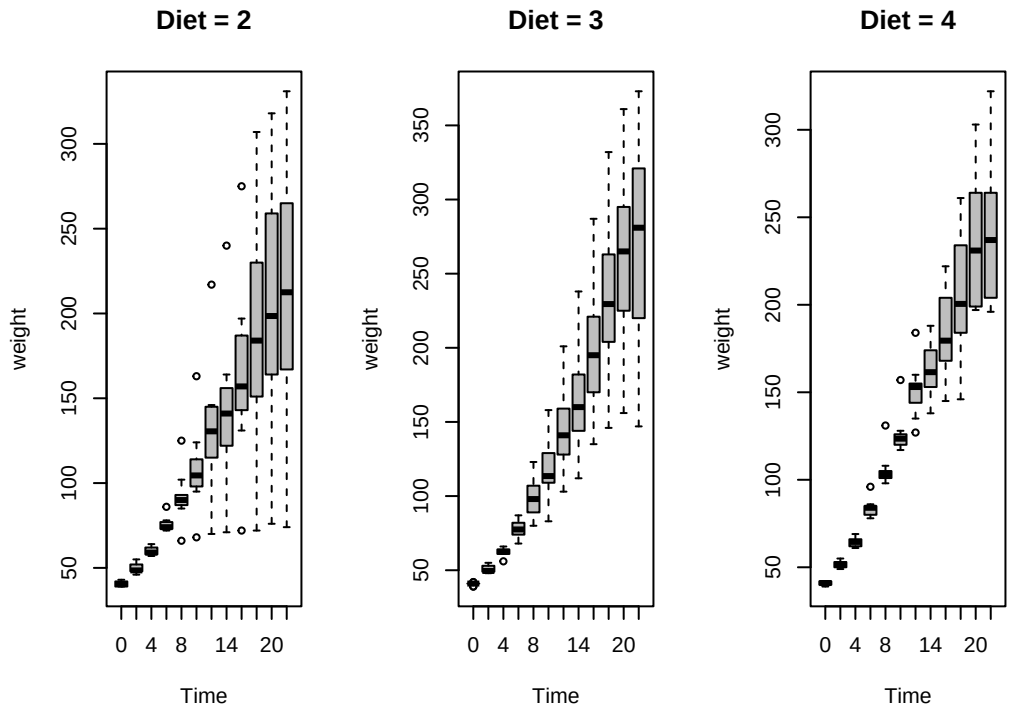
about the exact shape of the spread. Using `par(mfrow = c(2,2))` we can obtain four plots in a single plot window.

```
par(mfrow = c(1,1))
boxplot(weight ~ Time, data = diet_1,
        main = "Diet = I", col = diet_1$Time,
        lwd = 2)
```



We can execute the same task for other diet type as well.

```r
par(mfrow = c(1,3))
boxplot(weight ~ Time, data = diet_2,
        main = "Diet = 2", col = "grey",
        lwd = 1)
boxplot(weight ~ Time, data = diet_3,
        main = "Diet = 3", col = "grey",
        lwd = 1)
boxplot(weight ~ Time, data = diet_4,
        main = "Diet = 4", col = "grey",
        lwd = 1)
```



There are multiple ways to visualize the distribution of the data. `violin` plot is also commonly used to visualize the data distribution. This can be done by loading the package `vioplot`.

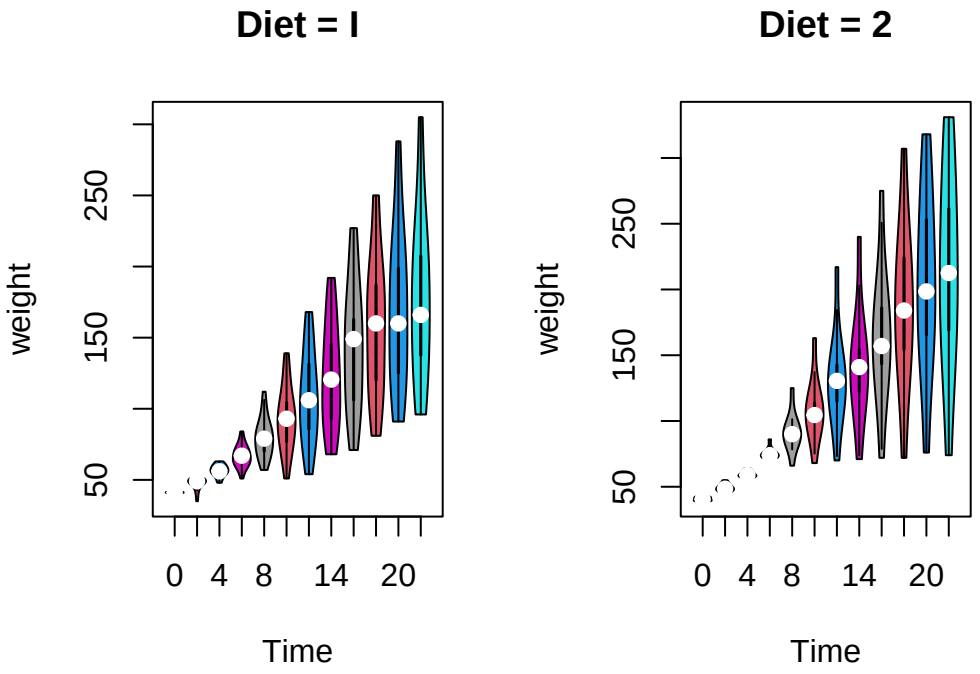
```r
library(vioplot)
```

```
Loading required package: sm
```

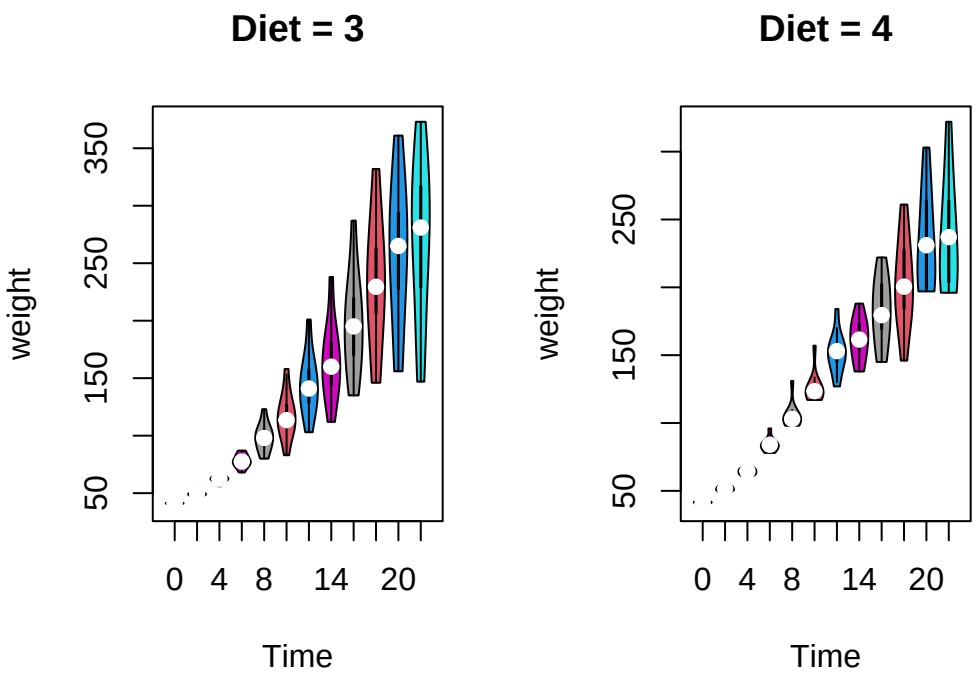
```
Package 'sm', version 2.2-6.0: type help(sm) for summary information
```

```
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

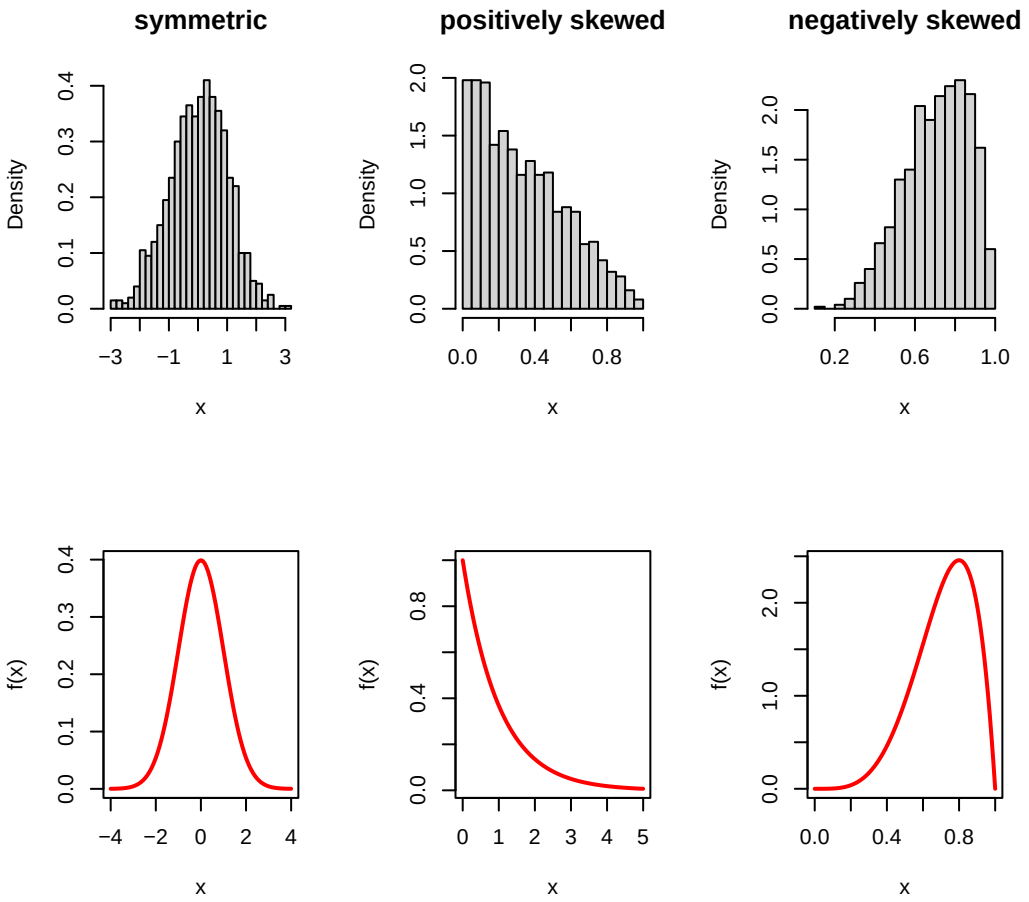
```r
par(mfrow = c(1,2))
vioplot(weight ~ Time, data = diet_1,
        main = "Diet = I", col = diet_1$Time,
        lwd = 1)
vioplot(weight ~ Time, data = diet_2,
        main = "Diet = 2", col = diet_2$Time,
        lwd = 1)
```




```r
par(mfrow = c(1,2))
vioplot(weight ~ Time, data = diet_3,
        main = "Diet = 3", col = diet_3$Time,
        lwd = 1)
vioplot(weight ~ Time, data = diet_4,
        main = "Diet = 4", col = diet_4$Time,
        lwd = 1)
```


Diet = 3
Diet = 4
weight
50
150
250
350
0
4
8
14
20
Time

Shapes of data distributions



Comparing the final growth

```
diet_1_T21 = subset(diet_1, Time==21, select = c(weight))
head(diet_1_T21)
```

```
   weight
12    205
24    215
36    202
48    157
60    223
72    157
```

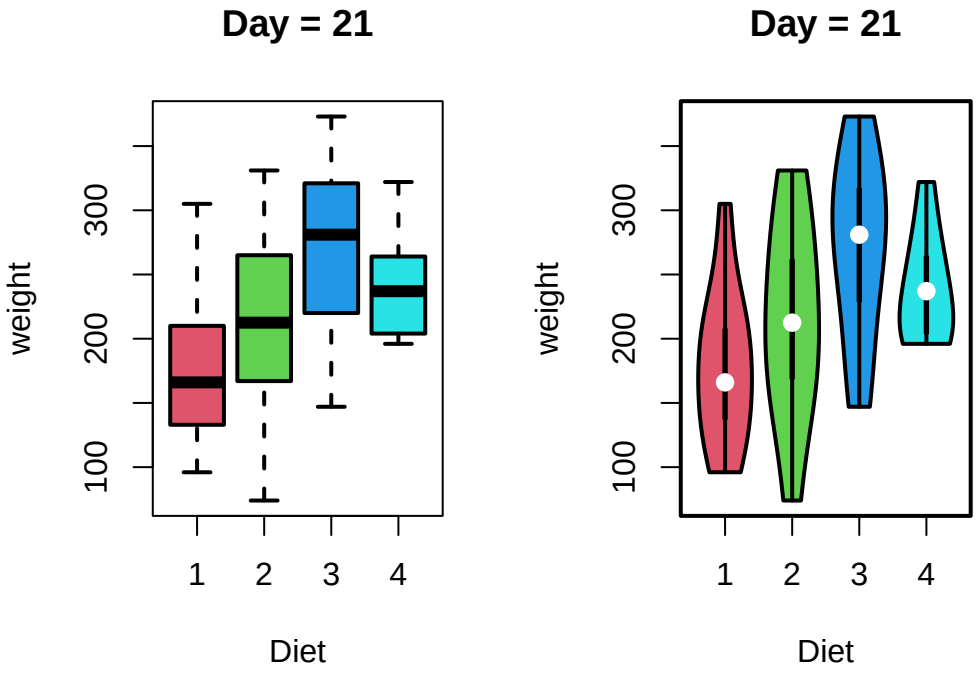
```r
diet_2_T21 = subset(diet_2, Time==21, select = c(weight))
diet_3_T21 = subset(diet_3, Time==21, select = c(weight))
diet_4_T21 = subset(diet_4, Time==21, select = c(weight))
```

Let us create a new data set with only the final weight of the chicks. The output of the subset function returns a data.frame, not a vector.

```r
weight = c(diet_1_T21$weight, diet_2_T21$weight,
           diet_3_T21$weight, diet_4_T21$weight)
Diet = c(rep(1, nrow(diet_1_T21)), rep(2, nrow(diet_2_T21)), rep(3, nrow(diet_3_T21)), rep(4
Diet = as.factor(Diet)         # store as factor
ChickWeight_T21 = data.frame(weight, Diet)
```

Let us make some informative plot of only the distribution of weights on 21st day for different dietary treatments.

```r
par(mfrow = c(1,2))
boxplot(weight ~ Diet, data = ChickWeight_T21,
        lwd = 2, main = "Day = 21", col = 2:5)
vioplot(weight ~ Diet, data = ChickWeight_T21,
        lwd = 2, main = "Day = 21",
        col = 2:5)
```



If $\mu_{21}^{(j)}, 1 \leq j \leq 4$ be the mean weight of the chick when considered under dietary type $j \in \{1, 2, 3, 4\}$, respectively. We are interested to test the following null hypothesis given by

$$H_0 : \mu_{21}^{(1)} = \mu_{21}^{(2)} = \mu_{21}^{(3)} = \mu_{21}^{(4)}$$

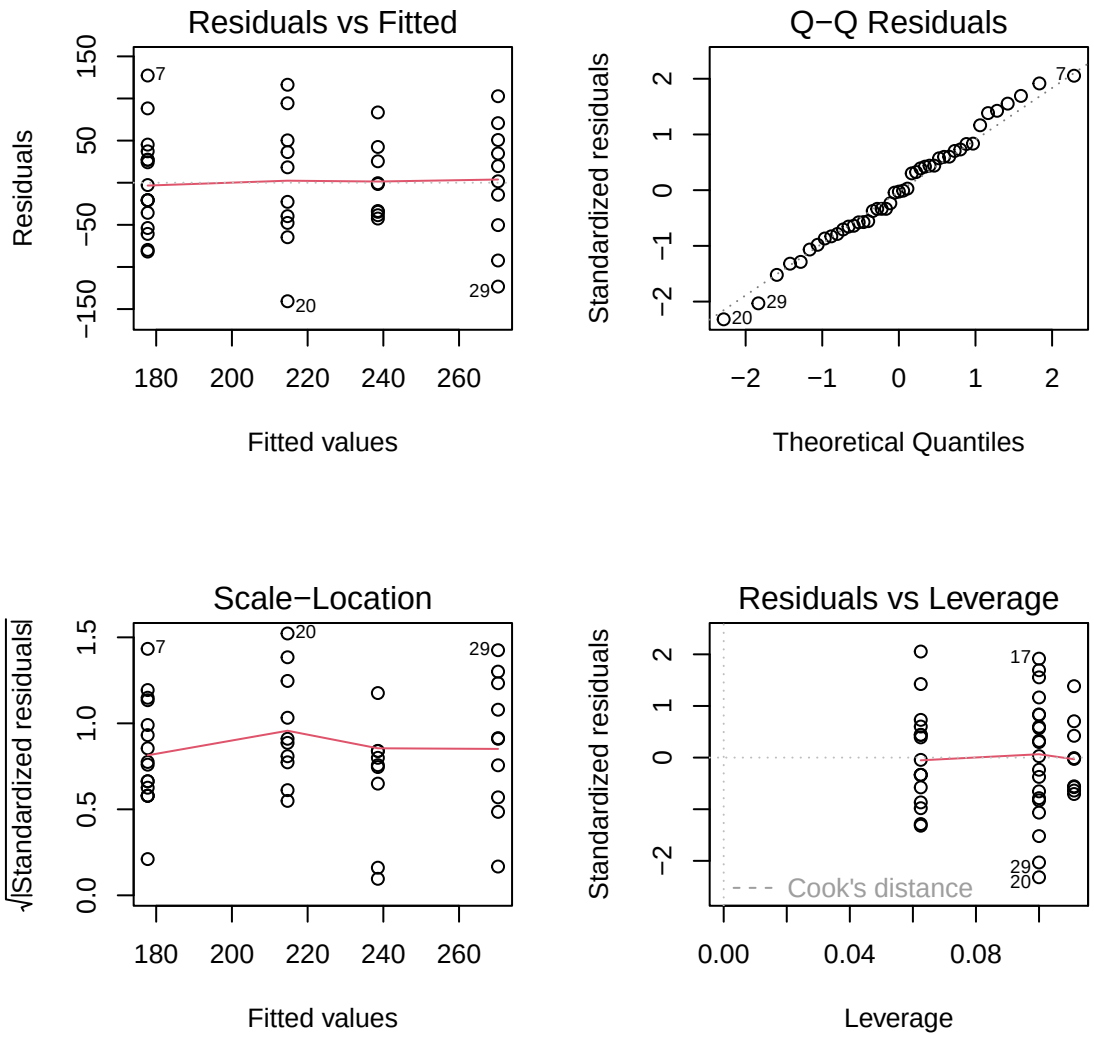
against the alternative H_1 which specifies that if at least two means are not equal. This falls under the statistical exercise known as the analysis of variance.

```
out = aov(weight ~ Diet, data = ChickWeight_T21)
summary(out)
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
Diet         3  57164   19055   4.655 0.00686 **
Residuals   41 167839    4094
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

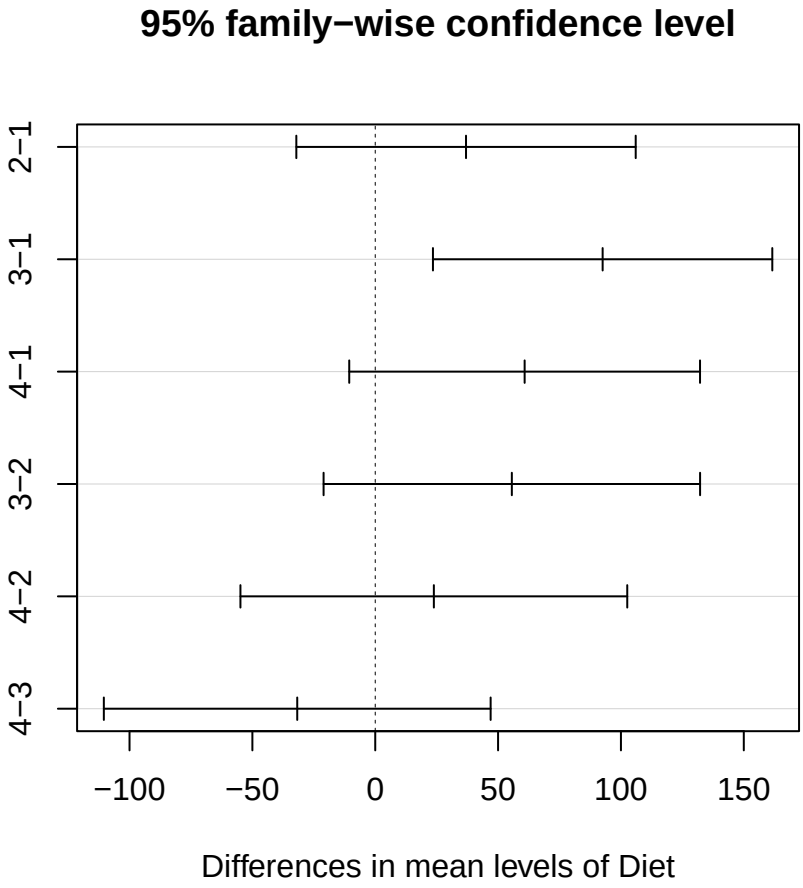
- Small p-value 0.00686 indicates that the null hypothesis is rejected at 5% level of significance.

```
par(mfrow = c(2,2))
plot(out)
```



- Natural question arises which two are different?

```
out_HSD = TukeyHSD(out, data = ChickWeight_T21)
plot(out_HSD)
```



Role of assumptions

Checking for normality assumptions of the weight of chicks under each diet. We can use the function `shapiro.test()` to check whether the data distribution is normal. The null hypothesis is

$$H_0: \quad \text{the data are normally distributed} \tag{0.1}$$
$$H_1: \quad \text{the data are not normally distributed} \tag{0.2}$$

```
1 shapiro.test(ChickWeight_T21$weight[Diet == 1])
```

```
    Shapiro-Wilk normality test

data:  ChickWeight_T21$weight[Diet == 1]
W = 0.95602, p-value = 0.5905
```

```r
shapiro.test(ChickWeight_T21$weight[Diet == 2])
```

```
    Shapiro-Wilk normality test

data:  ChickWeight_T21$weight[Diet == 2]
W = 0.97725, p-value = 0.9488
```

```r
shapiro.test(ChickWeight_T21$weight[Diet == 3])
```

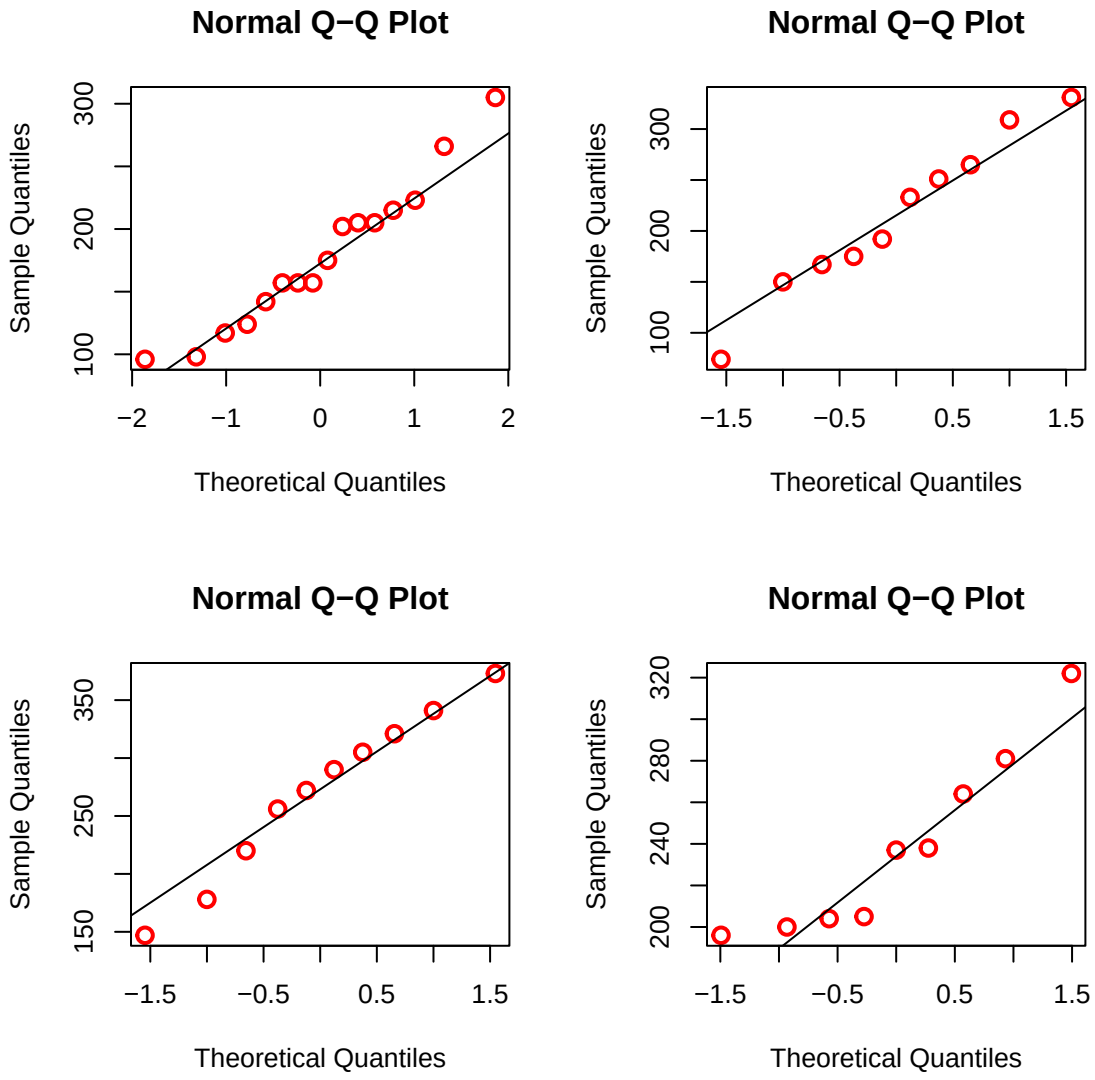
```
    Shapiro-Wilk normality test

data:  ChickWeight_T21$weight[Diet == 3]
W = 0.97045, p-value = 0.895
```

```r
shapiro.test(ChickWeight_T21$weight[Diet == 4])
```

```
    Shapiro-Wilk normality test

data:  ChickWeight_T21$weight[Diet == 4]
W = 0.88694, p-value = 0.1855
```

A large p value indicates that the data are normally distributed. Here the null hypothesis is that the data distribution is normal. Therefore, a large p-value indicates the acceptance of the null hypothesis. There are other visual representations to check the normality assumption.

```r
par(mfrow = c(2,2))
qqnorm(ChickWeight_T21$weight[Diet == 1], col = "red", cex = 1.3, lwd = 2)
qqline(ChickWeight_T21$weight[Diet == 1])
qqnorm(ChickWeight_T21$weight[Diet == 2], col = "red", cex = 1.3, lwd = 2)
qqline(ChickWeight_T21$weight[Diet == 2])
qqnorm(ChickWeight_T21$weight[Diet == 3], col = "red", cex = 1.3, lwd = 2)
qqline(ChickWeight_T21$weight[Diet == 3])
qqnorm(ChickWeight_T21$weight[Diet == 4], col = "red", cex = 1.3, lwd = 2)
qqline(ChickWeight_T21$weight[Diet == 4])
```

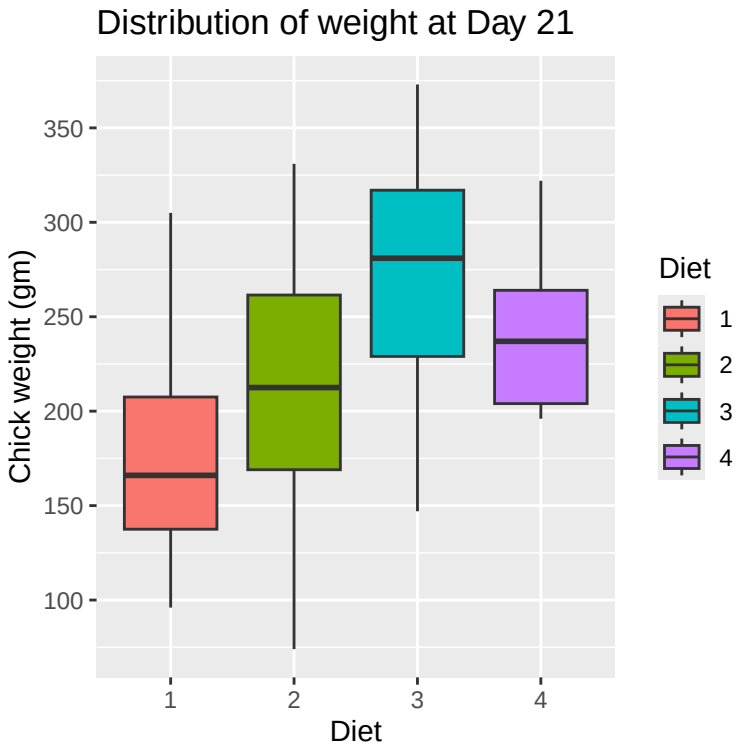


Some more options of boxplot using `ggplot2`

The package `ggplot2` offers several options for beautiful graphics using R. In the following, we demonstrate how to draw the side by side boxplot of continuous variable with respect to a factor variable in the data.

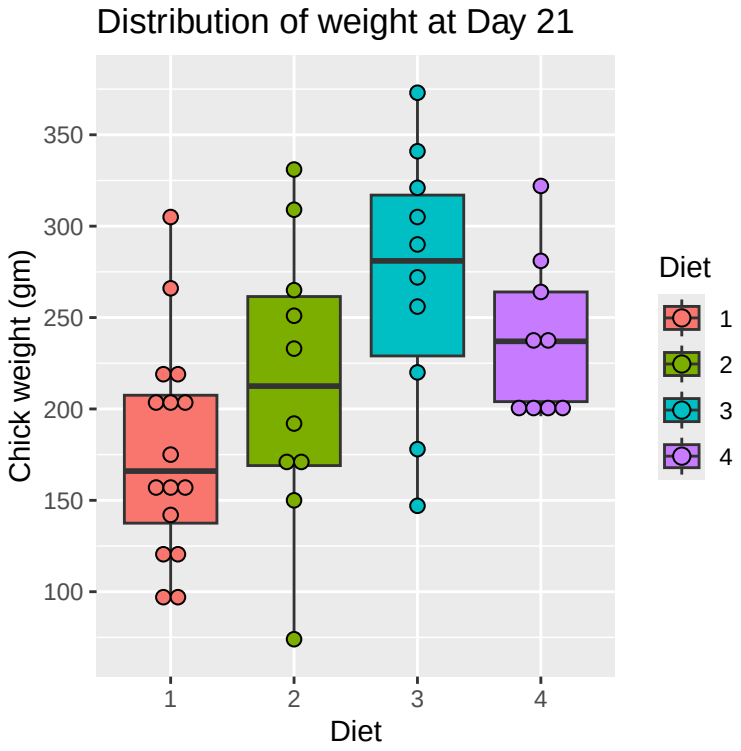
```
library(ggplot2)
ggplot2::ggplot(data = ChickWeight_T21,
                aes(Diet, weight, fill = Diet)) +
  geom_boxplot() +
  scale_y_continuous("Chick weight (gm)",
```

```
                      breaks = seq(50, 400, by = 50)) +
  labs(title = "Distribution of weight at Day 21", x = "Diet")
```



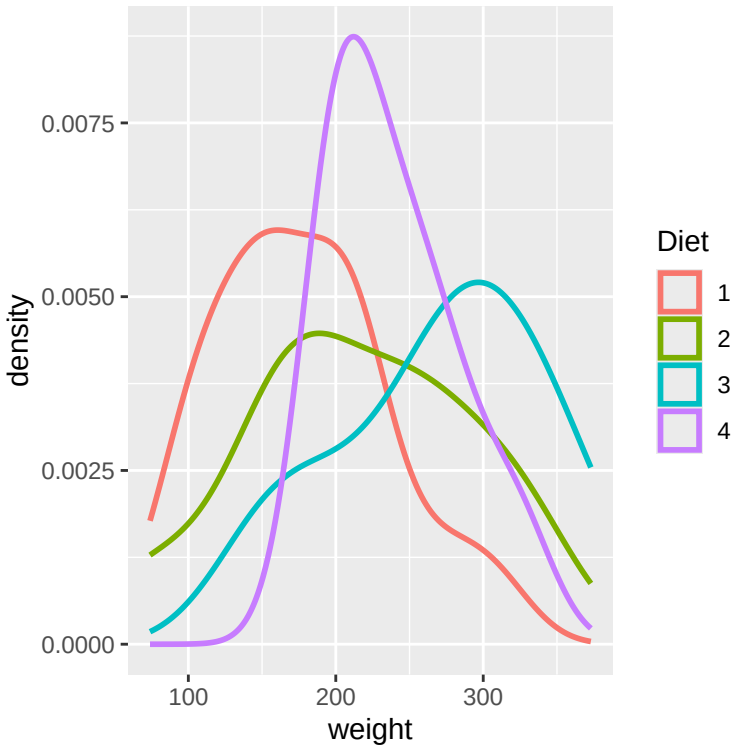

```
library(ggplot2)
ggplot2::ggplot(data = ChickWeight_T21,
                aes(Diet, weight, fill = Diet)) +
  geom_boxplot() +
  scale_y_continuous("Chick weight (gm)",
                     breaks = seq(50, 400, by = 50)) +
  labs(title = "Distribution of weight at Day 21", x = "Diet") +
  geom_dotplot(binaxis='y', stackdir='center', dotsize=0.8)
```

```
Bin width defaults to 1/30 of the range of the data. Pick better value with
`binwidth`.
```



To visualize the shape of the distribution with respect to different categories, we can also use the density plot.

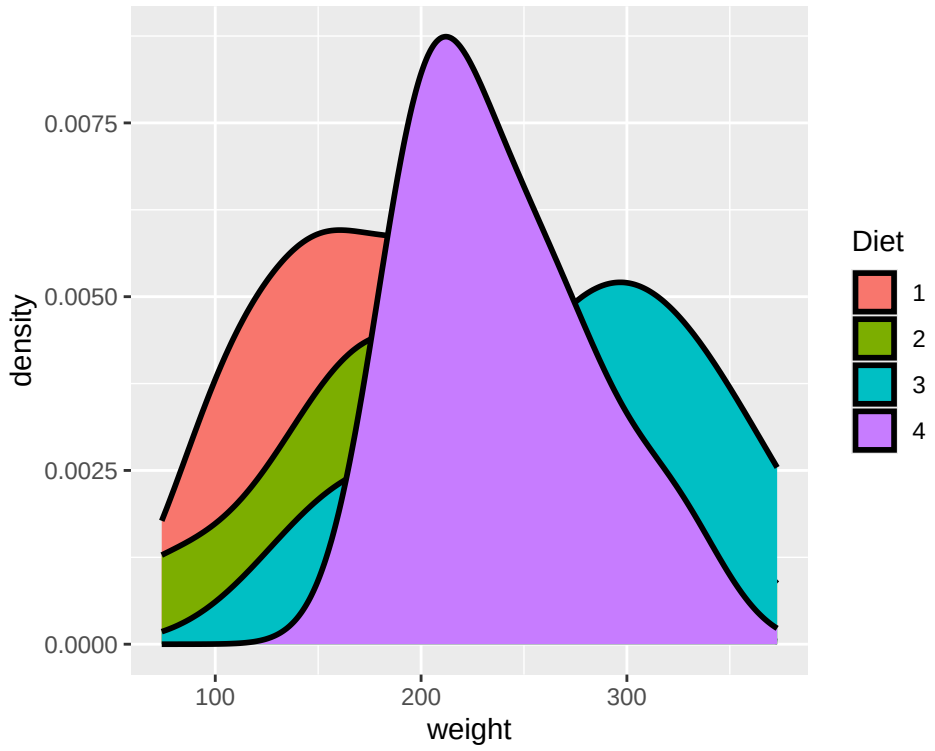
```
library(ggplot2)
ggplot(ChickWeight_T21, aes(x = weight, color = Diet)) +
  geom_density(linewidth = 1)
```



We can have vertical lines at the mean value of each diet category. First we need to compute the means of each group.

```
Diet_means = c(mean(ChickWeight_T21$weight[Diet==1]),
               mean(ChickWeight_T21$weight[Diet==2]),
               mean(ChickWeight_T21$weight[Diet==3]),
               mean(ChickWeight_T21$weight[Diet==4]))
```

```
library(ggplot2)
ggplot(ChickWeight_T21, aes(x = weight, fill = Diet)) +
  geom_density(linewidth = 1)
```



Testing for Equality of variances

```
var.test(ChickWeight_T21$weight[Diet == 1],
         ChickWeight_T21$weight[Diet == 2])
```

```
    F test to compare two variances

data:  ChickWeight_T21$weight[Diet == 1] and ChickWeight_T21$weight[Diet == 2]
F = 0.56439, num df = 15, denom df = 9, p-value = 0.3146
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1497316 1.7624337
sample estimates:
ratio of variances
         0.5643921
```

The Levene Test (Levene 1960) can be used to check for the equality of variances for multiple groups.

```r
library(car)
```

```
Loading required package: carData
```

```r
car::leveneTest(weight ~ Diet, data = ChickWeight_T21, center = mean)
```

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  3  1.2412 0.3072
      41
```

The Joyner–Boore Attenuation Data

`attenu` data set, known as `The Joyner-Boore Attenuation Data`, is available in the `datasets` package in R. This data gives peak accelerations measured at various observation stations for 23 earthquakes in California. The data have been used by various workers to estimate the attenuating affect of distance on ground acceleration. Type `help("attenu")` in the console for more information and also interested readers can check out the article by (Boore and Joyner 1982).

```r
library(datasets)
# datasets::attenu
head(attenu)
```

```
  event mag station dist accel
1     1 7.0     117   12 0.359
2     2 7.4    1083  148 0.014
3     2 7.4    1095   42 0.196
4     2 7.4     283   85 0.135
5     2 7.4     135  107 0.062
6     2 7.4     475  109 0.054
```

```r
names(attenu)                 # variable names
```

```
[1] "event"   "mag"     "station" "dist"    "accel"
```

```r
dim(attenu)                    # dimension of the data
```

```
[1] 182   5
```

```r
summary(attenu)                # basic summary of the data
```

```
     event            mag           station         dist
 Min.   : 1.00   Min.   :5.000   117     :  5   Min.   :  0.50
 1st Qu.: 9.00   1st Qu.:5.300   1028    :  4   1st Qu.: 11.32
 Median :18.00   Median :6.100   113     :  4   Median : 23.40
 Mean   :14.74   Mean   :6.084   112     :  3   Mean   : 45.60
 3rd Qu.:20.00   3rd Qu.:6.600   135     :  3   3rd Qu.: 47.55
 Max.   :23.00   Max.   :7.700   (Other):147   Max.   :370.00
                                 NA's    : 16
     accel
 Min.   :0.00300
 1st Qu.:0.04425
 Median :0.11300
 Mean   :0.15422
 3rd Qu.:0.21925
 Max.   :0.81000
```

- There are 16 observations in the `station` column in the data which is denoted by `NA`.
- Other variables are continuous in nature and their basic numerical summaries are provided and also they do not have any missing observations.

```r
# suppose we remove all stations whose ID is missing
complete.cases(attenu)    # which rows are complete
```

```
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
 [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
 [97]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
[109]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
```

```
[121]  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
[157]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[169]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[181]  TRUE  TRUE
```

```
attenu[81, ]
```

```
   event mag station dist accel
81    17 7.6    <NA> 32.9 0.064
```

```
# indices of rows with at least missing observations
miss_row = which(complete.cases(attenu) == FALSE)
print(miss_row)
```

```
 [1]  79  81  94  96  99 107 108 114 116 118 123 126 128 155 156 160
```

```
# Creating a new data after removing the missing observations
attenu_new = attenu[ -miss_row, ]
attenu_new$station
```

```
  [1] 117  1083 1095 283  135  475  113  1008 1028 2001 117  1117 1438 1083 1013
 [16] 1014 1015 1016 1095 1011 1028 270  280  116  266  117  113  112  130  475
 [31] 269  135  1093 1093 111  116  290  112  113  128  126  127  141  266  110
 [46] 1027 111  125  135  475  262  269  1052 411  290  130  272  1096 1102 112
 [61] 113  1028 2714 2708 2715 3501 655  272  1032 1377 1028 1250 1051 1293 1291
 [76] 1292 283  885  2734 2728 1413 1445 1408 1411 1410 1409 1377 1492 1251 1422
 [91] 1376 286  5028 942  5054 958  952  5165 117  955  5055 5060 412  5053 5058
[106] 5057 5051 5115 931  5056 5059 5061 5062 5052 724  5066 5050 2316 5055 942
[121] 5028 5165 952  958  955  117  412  5053 5054 5058 5057 5115 5056 5060 1030
[136] 1418 1383 1308 1298 1299 1219 1030 1418 1383 1299 1308 1219 1456 5045 5044
[151] 5160 5043 5047 c168 5068 c118 5042 5067 5049 c204 5070 c266 c203 5069 5073
[166] 5072
117 Levels: 1008 1011 1013 1014 1015 1016 1027 1028 1030 1032 1051 1052 ... c266
```

```
dim(attenu_new)
```

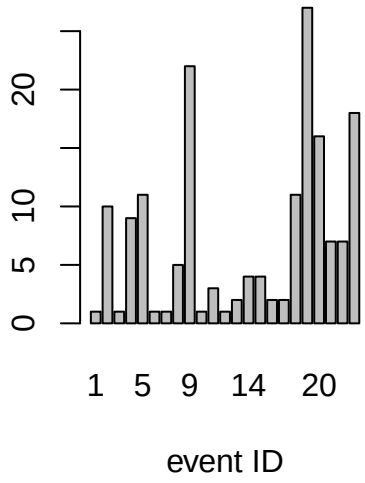
```
[1] 166   5
```

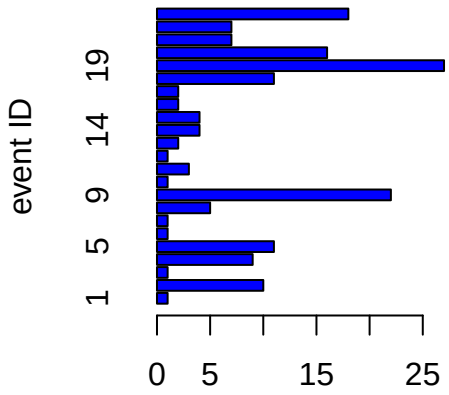
```r
summary(attenu_new)
```

```
     event             mag           station         dist
 Min.   : 1.00   Min.   :5.000   117    :  5   Min.   :  0.60
 1st Qu.: 9.00   1st Qu.:5.300   1028   :  4   1st Qu.: 12.03
 Median :18.00   Median :6.100   113    :  4   Median : 24.40
 Mean   :14.31   Mean   :6.064   112    :  3   Mean   : 48.39
 3rd Qu.:20.00   3rd Qu.:6.600   135    :  3   3rd Qu.: 49.80
 Max.   :23.00   Max.   :7.700   475    :  3   Max.   :370.00
                                 (Other):144
     accel
 Min.   :0.0030
 1st Qu.:0.0400
 Median :0.1100
 Mean   :0.1462
 3rd Qu.:0.2000
 Max.   :0.8100
```

Explore individual variables

```r
par(mfrow  = c(1,2))
w = table(attenu_new$event)
barplot(w, xlab = "event ID", width = 1,
        col = "grey")
barplot(w, ylab = "event ID", width = 1,
        col = "blue", horiz = TRUE)
```

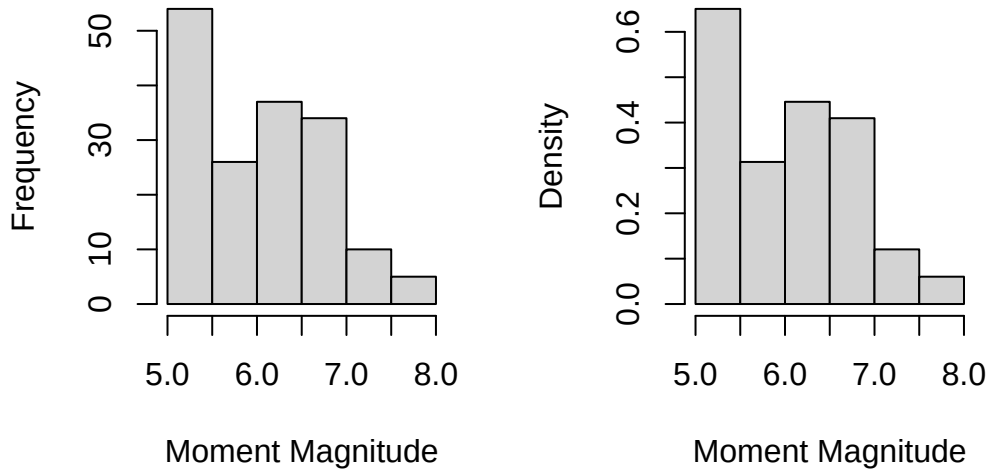




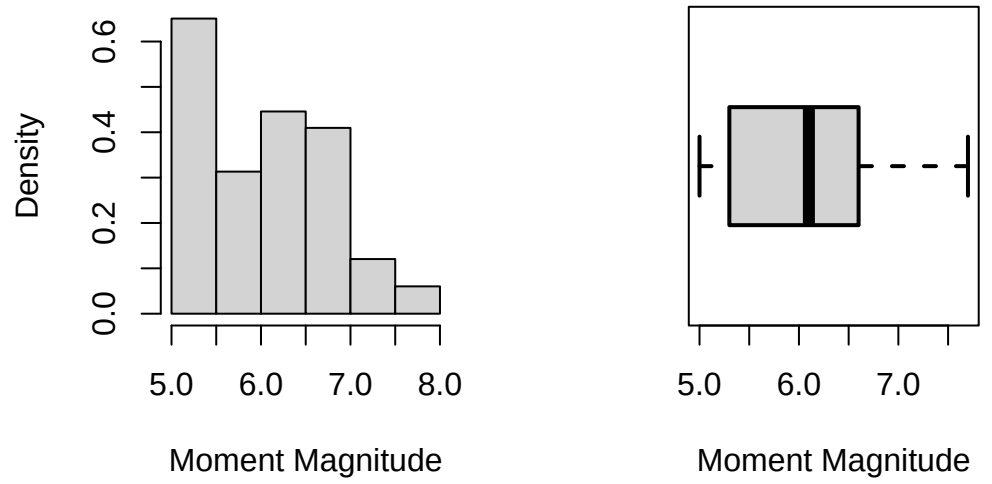

```
par(mfrow = c(1,2))
attenu_new$mag
```

```
  [1] 7.0 7.4 7.4 7.4 7.4 7.4 7.4 7.4 7.4 7.4 7.4 5.3 6.1 6.1 6.1 6.1 6.1 6.1
 [19] 6.1 6.1 6.1 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 5.6 5.7 5.3 5.3
 [37] 5.3 5.3 5.3 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6 6.6
 [55] 6.6 6.6 6.6 6.6 6.6 6.6 6.6 5.3 7.7 7.7 7.7 6.2 5.6 5.6 5.2 5.2 5.2 5.2
 [73] 6.0 6.0 6.0 6.0 5.1 5.1 7.6 7.6 5.8 5.8 5.8 5.8 5.8 5.8 5.8 5.8 5.8 5.8
 [91] 5.8 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5
[109] 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
[127] 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.8 5.8 5.8 5.8 5.8 5.8 5.8 5.5 5.5 5.5
[145] 5.5 5.5 5.5 5.5 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3
[163] 5.3 5.3 5.3 5.3
```

```
hist(attenu_new$mag, xlab = "Moment Magnitude", main = " ")

hist(attenu_new$mag, xlab = "Moment Magnitude", main = " ",
     probability = TRUE)
```

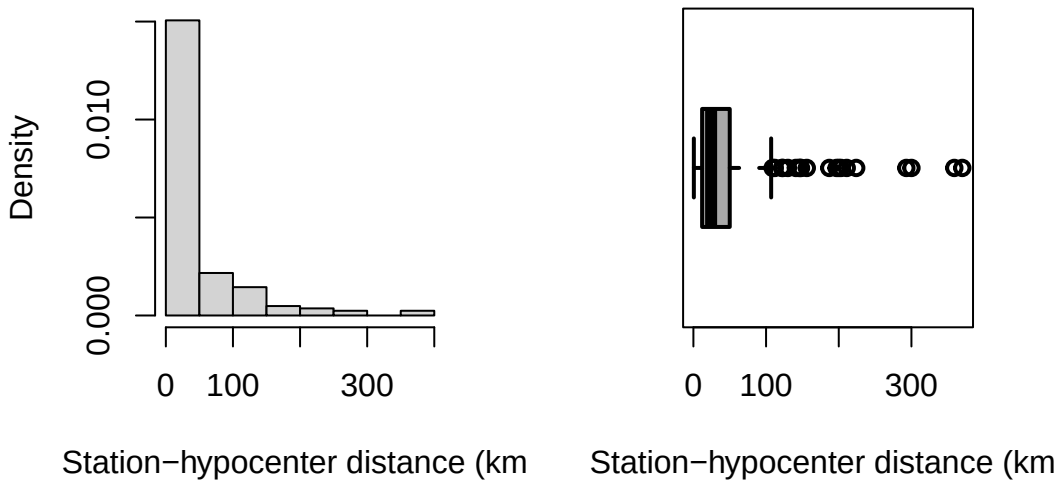



```
par(mfrow = c(1,2))
hist(attenu_new$mag, xlab = "Moment Magnitude", main = " ",
     probability = TRUE)
boxplot(attenu_new$mag, horizontal = TRUE, lwd = 2,
        xlab = "Moment Magnitude")
```

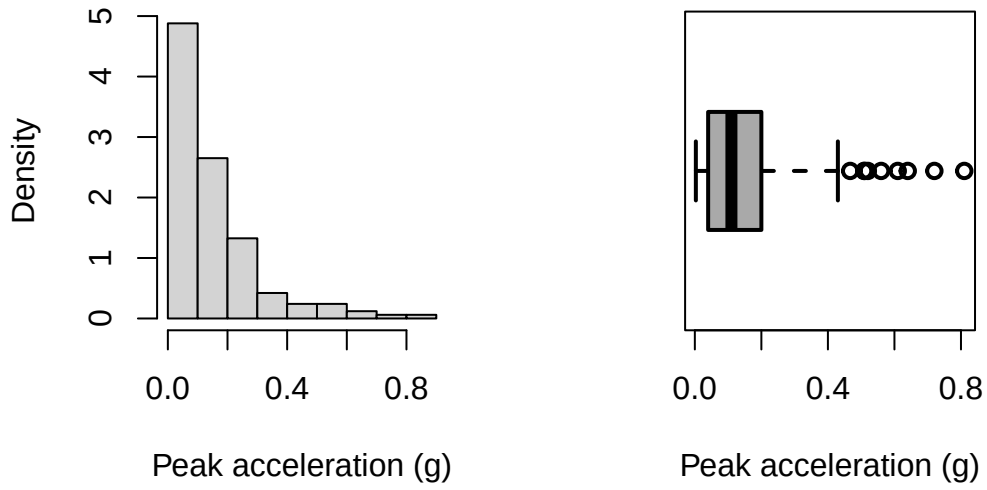



```
par(mfrow = c(1,2))
hist(attenu_new$dist, probability = TRUE,
     xlab = "Station-hypocenter distance (km)", main = "")
```

```r
boxplot(attenu_new$dist, xlab = "Station-hypocenter distance (km)",
        horizontal = TRUE, lwd = 2, col = "darkgrey")
```




```r
par(mfrow = c(1,2))
hist(attenu_new$accel, probability = TRUE,
     xlab = "Peak acceleration (g)", main = "")
boxplot(attenu_new$accel, xlab = "Peak acceleration (g)",
        horizontal = TRUE, lwd = 2, col = "darkgrey")
```



Important functions:

- `complete.cases`, `dim`, `which`

- `summary`
- `boxplot`, `hist`, `barplot`

The cars dataset

The data give the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s (Ezekiel 1930).

```
# datasets::cars
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

```
tail(cars)
```

```
   speed dist
45    23   54
46    24   70
47    24   92
48    24   93
49    24  120
50    25   85
```

```
class(cars)
```

```
[1] "data.frame"
```

```
dim(cars)
```

```
[1] 50  2
```

```r
names(cars)
```

```
[1] "speed" "dist"
```

```r
complete.cases(cars)
```

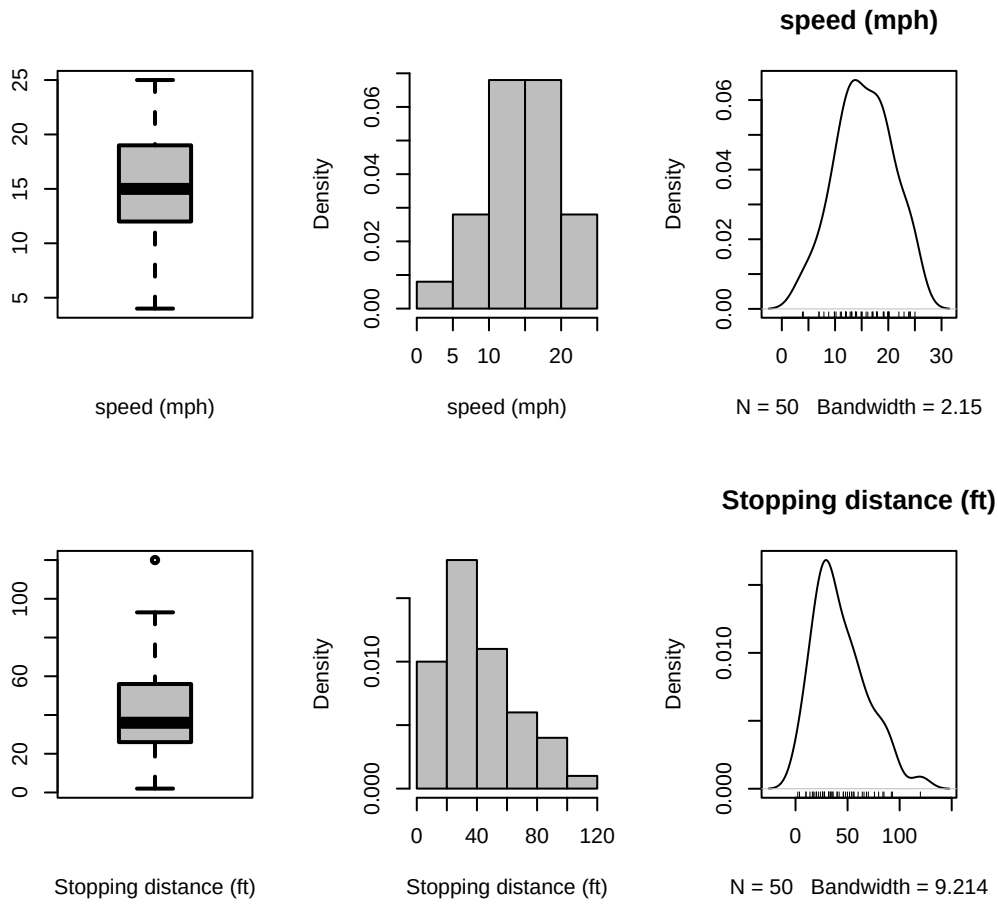
```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[46] TRUE TRUE TRUE TRUE TRUE
```

The function `complete.cases()` returns `FALSE` if there is any row with at least one variable information is missing. Also try using `!complete.cases()`.

Basic data exploration

Looking at individual variables is always suggested as a part of the EDA.

```r
par(mfrow= c(2,3))
boxplot(cars$speed, col = "grey", lwd = 2,
        xlab = "speed (mph)")
hist(cars$speed, probability = TRUE, col = "grey",
        xlab = "speed (mph)", main = "")
plot(density(cars$speed), main = "speed (mph)")
rug(jitter(cars$speed))
boxplot(cars$dist, col = "grey", lwd = 2,
        xlab = "Stopping distance (ft)")
hist(cars$dist, probability = TRUE, col = "grey",
        xlab = "Stopping distance (ft)", main = "")
plot(density(cars$dist), main = "Stopping distance (ft)")
rug(jitter(cars$dist))
```



We can use the `shapiro.test()` function to test whether the data is normally distributed.

```
1 shapiro.test(cars$speed)
```

```
    Shapiro-Wilk normality test

data:  cars$speed
W = 0.97765, p-value = 0.4576
```

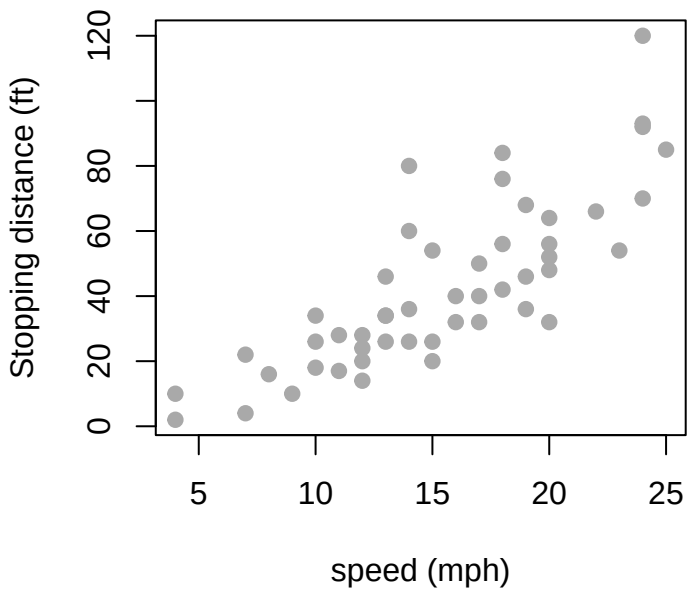
```
1 shapiro.test(cars$dist)
```

```
    Shapiro-Wilk normality test

data:  cars$dist
W = 0.95144, p-value = 0.0391
```

From the output, at 5% level of significance, we reject the null hypothesis for the variable Stopping distance (ft) that it is normally distributed using Shapiro Wilks test Royston (1995). However, it is important to note that the p-value is 0.0391, therefore, we can not reject it at 1% level of significance.

A natural question arises how Stopping distance (ft) varies as a function of speed (mph). First let us explore the scatter plot between these two variables.

```
par(mfrow = c(1,1))
plot(cars$speed, cars$dist, type = "p", pch = 19,
     col = "darkgrey", ylab = "Stopping distance (ft)",
     xlab = "speed (mph)")
```



To explore some kind of linear relationship between Stopping distance and Speed, we first check whether there is significant correlation between them. Therefore, we test the following hypothesis for the correlation coefficient $\rho = \text{Corr}(\text{dist, speed})$

$$H_0 \colon \rho = 0 \quad \text{against} \quad H_1 \colon \rho \neq 0.$$

The test is performed using the function `cor.test()`.

```
cor(cars$speed, cars$dist)
```

```
[1] 0.8068949
```

```r
cor.test(cars$speed, cars$dist)
```

```
    Pearson's product-moment correlation

data:  cars$speed and cars$dist
t = 9.464, df = 48, p-value = 1.49e-12
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6816422 0.8862036
sample estimates:
      cor
0.8068949
```

```r
fit = lm(dist ~ speed, data = cars)
summary(fit)
```

```
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

```r
plot(cars$speed, cars$dist, type = "p", pch = 19,
     col = "darkgrey", ylab = "Stopping distance (ft)",
     xlab = "speed (mph)")
abline(fit, col = "red", lwd = 3, lty = 2)
```

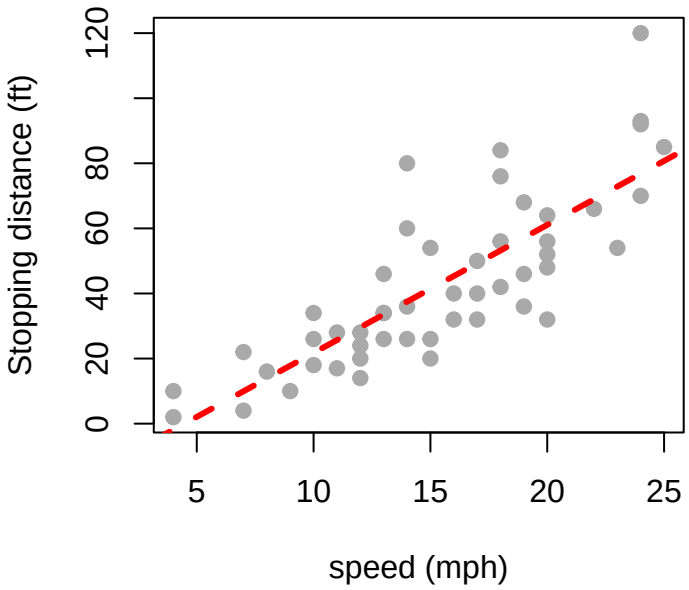


Figure 1: Fitting of the simple linear regression model.

- beta-coefficient of `speed` is statistically significant, which means the stopping distance depends on speed.
- The Multiple R-squared = 0.6511, which means that approximately 65% of the variability in `dist` can be explained by the the variable `speed` through a linear function.

```
coefficients(fit)
```

```
(Intercept)       speed
 -17.579095    3.932409
```

In this problem, we have assumed a linear relationship between the dist and speed. However, we do not know whether the linearity assumption is a legitimate choice to approximate the true relationship (which we do not know). We do some diagnostic tests to check whether the linearity assumption is tenable.

```
par(mfrow = c(2,2))
plot(fit)
```

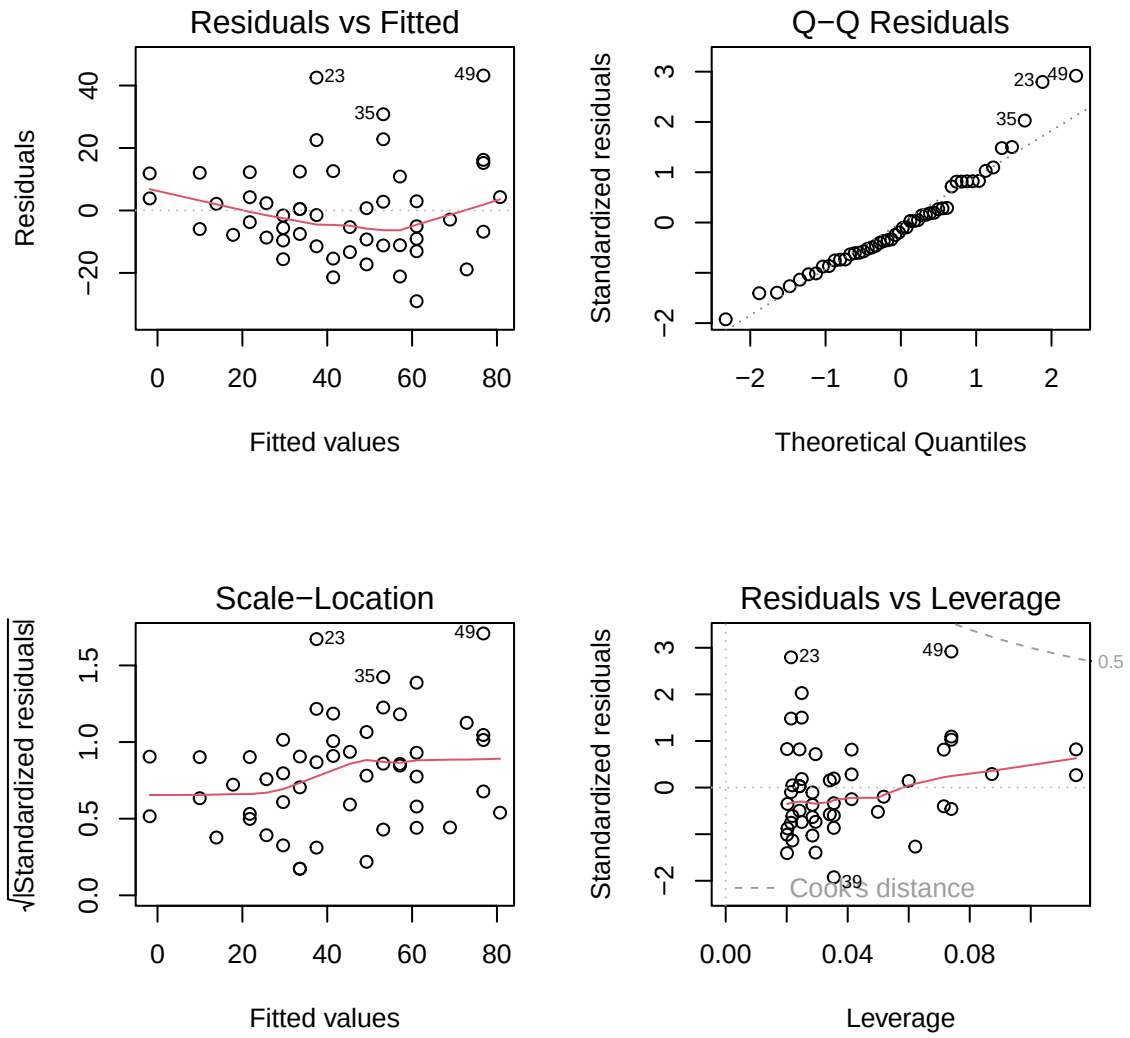


Figure 2: Regression diagnostic plot for the simple linear regression. The top panel indicates the existence of outlying observations, particularly the row numbers 23, 35 and 49 have been identified as the outlying observations.

```r
par(mfrow = c(1,3))
error = residuals(fit)           # residuals
hist(error, probability = TRUE, xlab = expression(widehat(epsilon)),
     main = "")
boxplot(error, xlab = expression(widehat(epsilon)),
        lwd = 2, col = "darkgrey")
library(car)
qqPlot(error, ylab = expression(widehat(epsilon)))
```

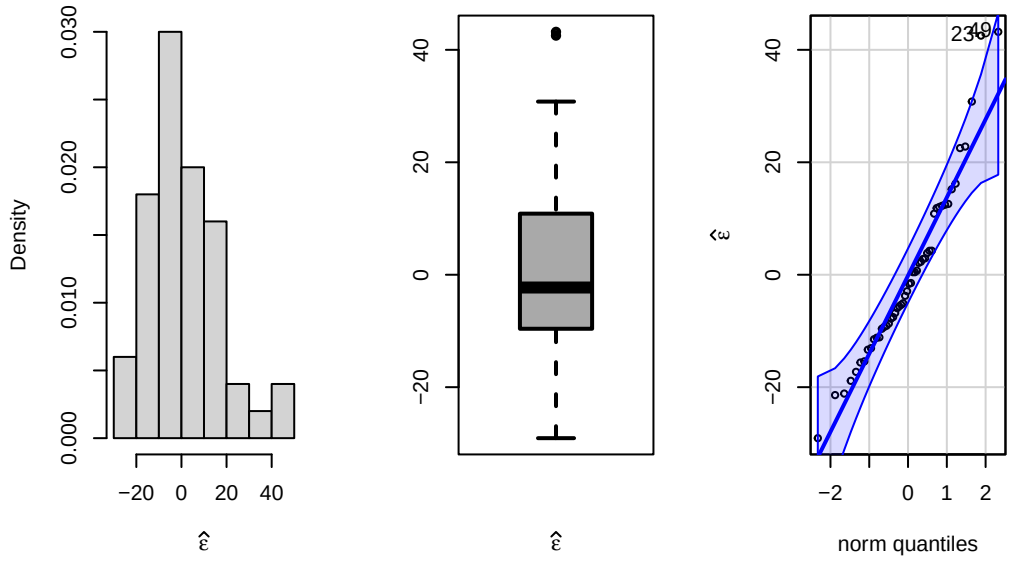


Figure 3: The `car` package has been used to draw the quantile plot the plots empirical quantiles of a variable against theoretical quantiles of a comparison distribution (here it is normal)(Fox 2016).

```
[1] 49 23
```

- There may be existence of nonlinearity.
- Regression diagnostics gave some outlying observations.

You can possibly remove those outliers and perform a linear regression, or you can go for a quadratic regression.

Quadratic fitting

We are fitting the following quadratic polynomial equation

$$\text{dist} = \beta_0 + \beta_1 \times \text{speed} + \beta_2 \times \text{speed}^2.$$

Using the following R Codes, we can obtain the least squares estimate of the coefficients $\beta_0, \beta_1, \beta_2$. The symbol `I`$(\cdot)$ is used as wrapper for polynomial terms to create the formula for the `lm()` function.

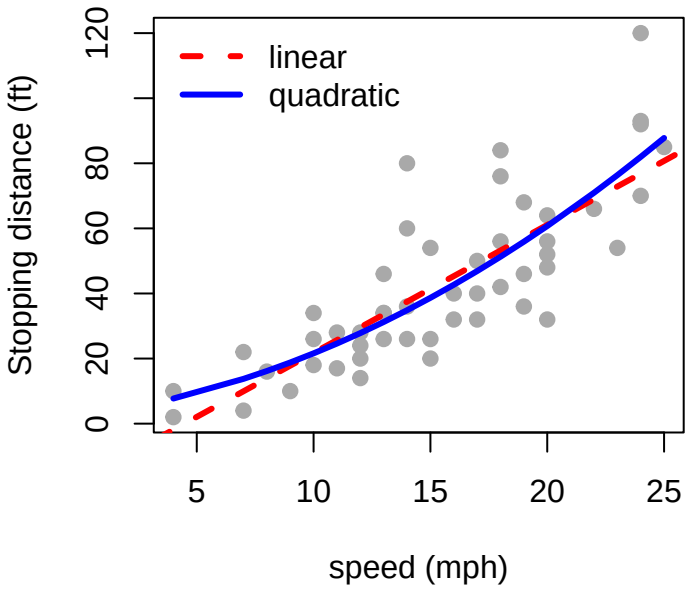
```
1 par(mfrow = c(1,1))
2 fit2 = lm(dist ~ speed + I(speed^2), data = cars)
3 summary(fit2)
```

```
Call:
lm(formula = dist ~ speed + I(speed^2), data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-28.720  -9.184  -3.188   4.628  45.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.47014   14.81716   0.167    0.868
speed        0.91329    2.03422   0.449    0.656
I(speed^2)   0.09996    0.06597   1.515    0.136

Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-squared:  0.6673,    Adjusted R-squared:  0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

```r
coef(fit2)
```

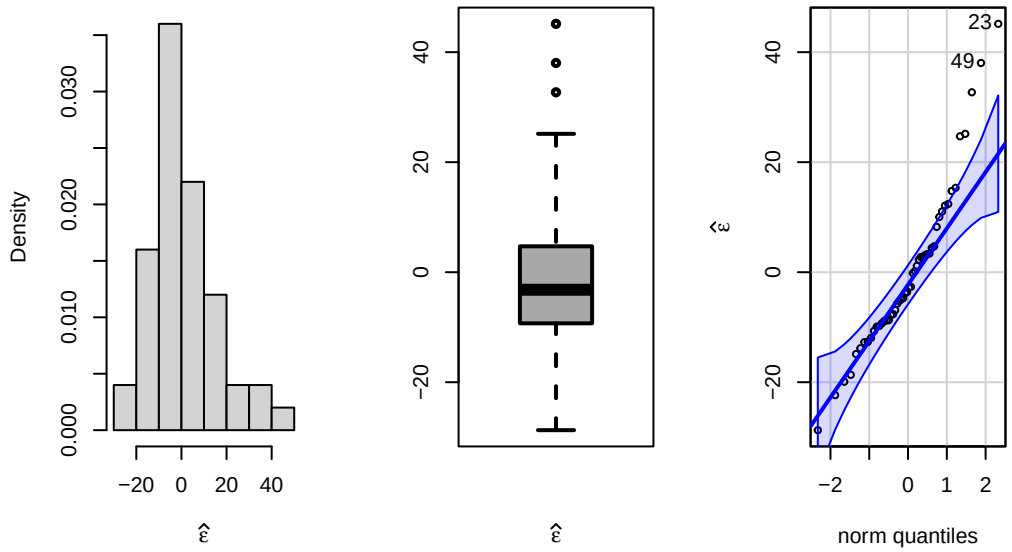
```
(Intercept)       speed  I(speed^2)
  2.4701378   0.9132876   0.0999593
```

```r
dist_hat = coef(fit2)[1] + coef(fit2)[2]*cars$speed + coef(fit2)[3]*cars$speed^2

plot(cars$speed, cars$dist, type = "p", pch = 19,
     col = "darkgrey", ylab = "Stopping distance (ft)",
     xlab = "speed (mph)")
abline(fit, col = "red", lwd = 3, lty = 2)
lines(cars$speed, dist_hat, type = "l", col= "blue",
      lwd = 3)
legend("topleft", legend = c("linear", "quadratic"),
       col = c("red", "blue"), lwd = c(3,3),
       bty = "n", lty = c(2,1))
```



Regression diagnostic for quadratic regression

```
par(mfrow = c(1,3))
error = residuals(fit2)
hist(error, probability = TRUE, xlab = expression(widehat(epsilon)),
     main = "")
boxplot(error, xlab = expression(widehat(epsilon)),
        lwd = 2, col = "darkgrey")
library(car)
qqPlot(error, ylab = expression(widehat(epsilon)))
```




```
[1] 23 49
```

We can obtain the diagnostic plot using the `plot()` function.

```
1 par(mfrow = c(2,2))
2 plot(fit2)
```

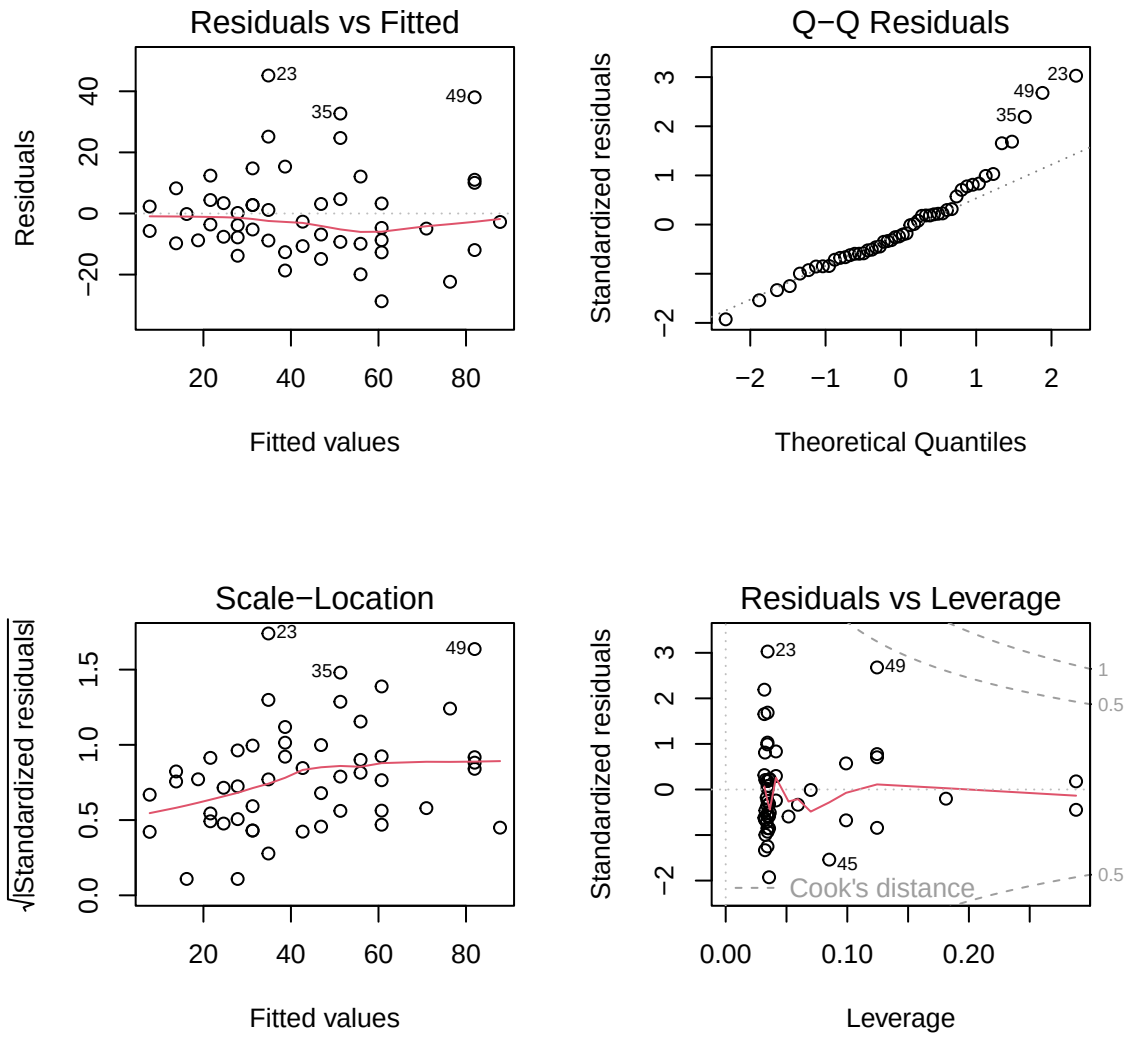


Figure 4: Regression diagnostc for the quadratic function fitting. In fact, the Q-Q Residuals plot gave more number of outliers and to some extent we get a feeling that quadratic regression might not be a good option as the there is deviance from the normality of residuals. In addition, the residuals versus fitted values plot provides a slight indication of heteroschedasticity.

There seem to be slight improvement using a quadratic equation than a linear equation. However, it can be statistically check whether this increment is statistically significant or not. The function `anova()` is used to compare the performance of fitting exercises using two different functions.

```r
anova(fit, fit2)
```

```
Analysis of Variance Table

Model 1: dist ~ speed
Model 2: dist ~ speed + I(speed^2)
  Res.Df   RSS Df Sum of Sq     F Pr(>F)
1     48 11354
2     47 10825  1    528.81 2.296 0.1364
```

As per the `qqPlot()` function from the `car` package, two data points (row number 23 and 49) have been identified as outliers leading to deviation from the normality of the residuals. Let us remove them.

```r
fit_new = lm(dist ~ speed, data = cars[-c(23, 35, 49),])
summary(fit_new)
```

```
Call:
lm(formula = dist ~ speed, data = cars[-c(23, 35, 49), ])

Residuals:
    Min      1Q  Median      3Q     Max
-25.032  -7.686  -1.032   6.576  26.185

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -15.1371     5.3053  -2.853  0.00652 **
speed         3.6085     0.3302  10.928    3e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.84 on 45 degrees of freedom
Multiple R-squared:  0.7263,    Adjusted R-squared:  0.7202
F-statistic: 119.4 on 1 and 45 DF,  p-value: 3.003e-14
```

```r
par(mfrow = c(1,1))
plot(cars$speed, cars$dist, type = "p", pch = 19,
     col = "darkgrey", ylab = "Stopping distance (ft)",
     xlab = "speed (mph)")
abline(fit, col = "red", lwd = 3, lty = 2)
```

```
abline(fit_new, col = "magenta", lwd = 3, lty = 2)
points(cars[c(23,35,49),], col = "red", cex = 2,
       lwd = 3, pch = 4)
```

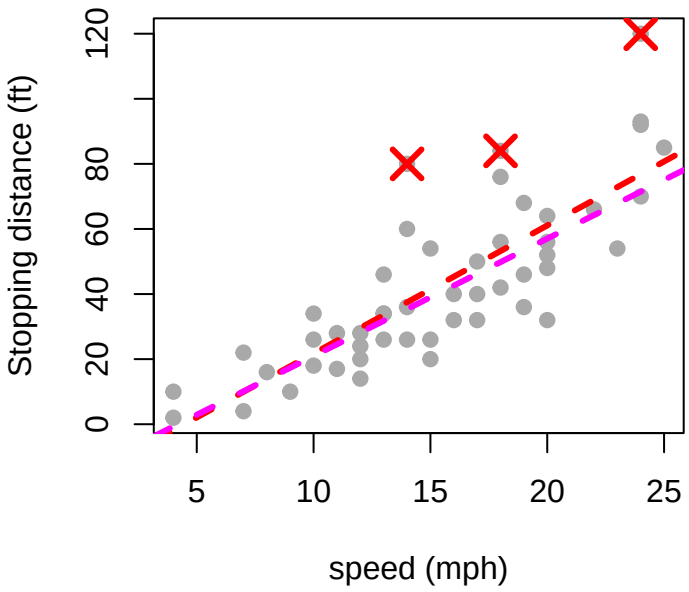


Figure 5: The dotted magenta line indicates the fitted linear regression line after removing the three outlying observations as identified by the regression diagnostics.

```
par(mfrow = c(2,2))
plot(fit_new)
```

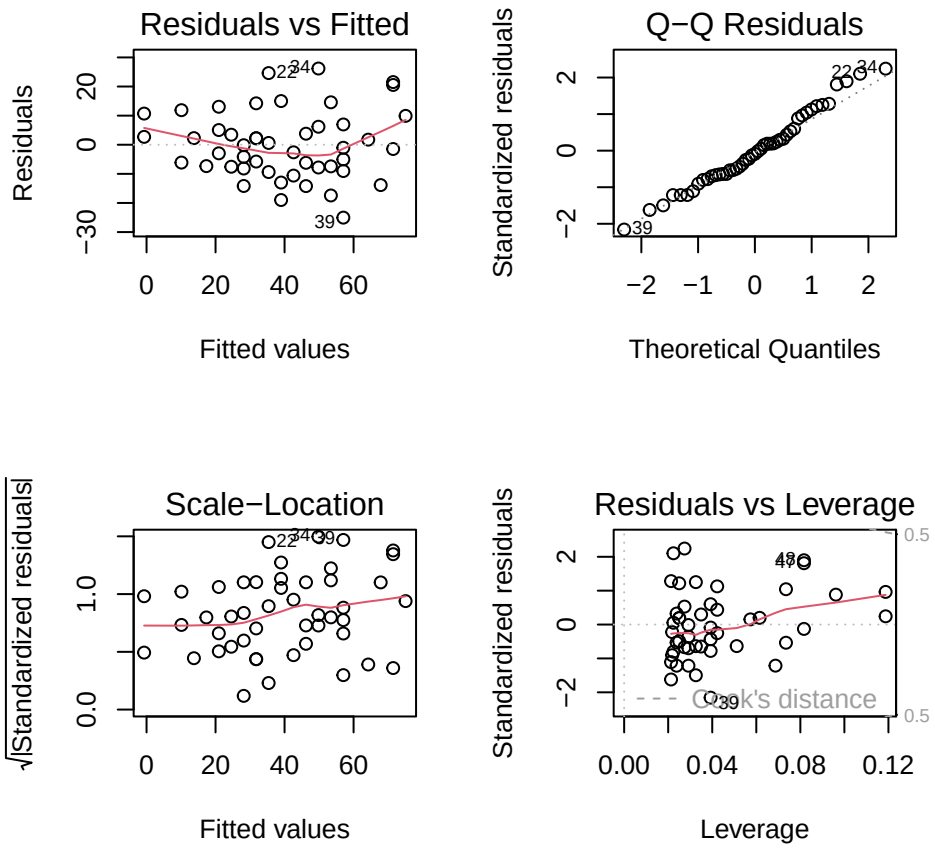


Figure 6: Regression diagnostic for the linear regression model after removing three outlying observations.

Although after removing the designated outliers from the regression diagnostic plots, we obtained a significant improvement in the R^2 values (approx 75%), however, the diagnostic plot of the new regression fit, gives stronger evidence towards the existence of nonlinear relationship.

```
fit3 = lm(dist ~ speed + I(speed^2) + I(speed^3) , data = cars)
summary(fit3)
```

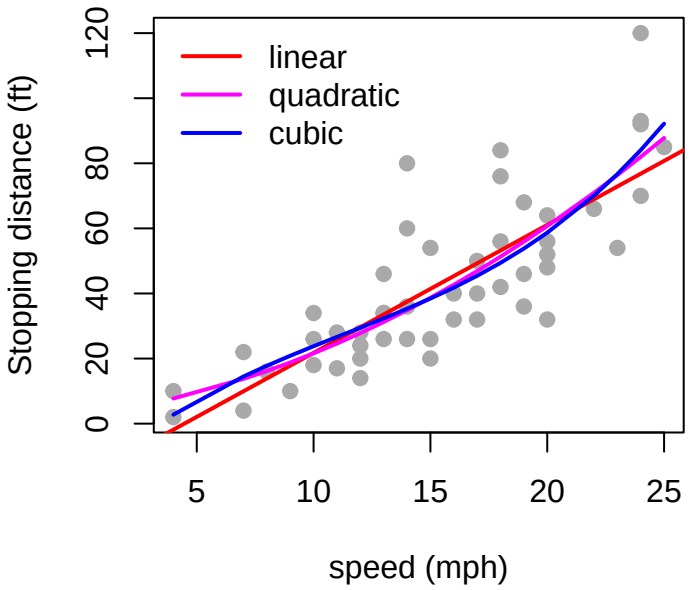
```
Call:
lm(formula = dist ~ speed + I(speed^2) + I(speed^3), data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-26.670  -9.601  -2.231   7.075  44.691
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -19.50505   28.40530  -0.687    0.496
speed         6.80111    6.80113   1.000    0.323
I(speed^2)   -0.34966    0.49988  -0.699    0.488
I(speed^3)    0.01025    0.01130   0.907    0.369

Residual standard error: 15.2 on 46 degrees of freedom
Multiple R-squared:  0.6732,    Adjusted R-squared:  0.6519
F-statistic: 31.58 on 3 and 46 DF,  p-value: 3.074e-11
```

```r
dist_hat = predict(fit3)
print(dist_hat)
```

```
       1        2        3        4        5        6        7        8
 2.760981  2.760981 14.485912 14.485912 17.774747 20.856365 23.792277 23.792277
       9       10       11       12       13       14       15       16
23.792277 26.643997 26.643997 29.473036 29.473036 29.473036 29.473036 32.340907
      17       18       19       20       21       22       23       24
32.340907 32.340907 32.340907 35.309122 35.309122 35.309122 35.309122 38.439194
      25       26       27       28       29       30       31       32
38.439194 38.439194 41.792634 41.792634 45.430956 45.430956 45.430956 49.415670
      33       34       35       36       37       38       39       40
49.415670 49.415670 49.415670 53.808290 53.808290 53.808290 58.670328 58.670328
      41       42       43       44       45       46       47       48
58.670328 58.670328 58.670328 70.048706 76.688071 84.042903 84.042903 84.042903
      49       50
84.042903 92.174715
```

```r
par(mfrow = c(1,1))
plot(cars$speed, cars$dist, type = "p", pch = 19,
     col = "darkgrey", ylab = "Stopping distance (ft)",
     xlab = "speed (mph)")
abline(fit, col = "red", lwd = 2)
lines(cars$speed, predict(fit2), col = "magenta", lwd = 2)
lines(cars$speed, predict(fit3), col = "blue", lwd = 2)
legend("topleft", legend = c("linear", "quadratic", "cubic"),
       col = c("red","magenta", "blue"), lwd = c(2,2,2),
       bty = "n")
```



Let us check how the R^2 values changes as the degree of the polynomial increases.

```
fit4 = lm(dist ~ speed + I(speed^2) + I(speed^3) + I(speed^4) ,
          data = cars)
summary(fit4)
```

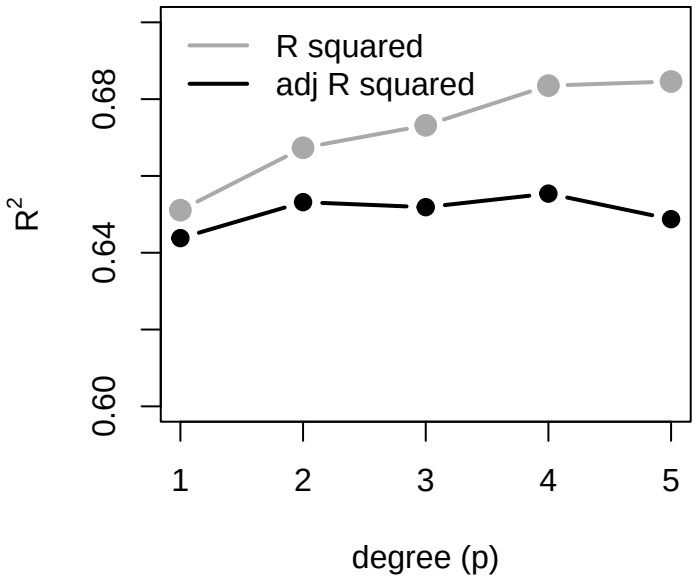
```
Call:
lm(formula = dist ~ speed + I(speed^2) + I(speed^3) + I(speed^4),
    data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-23.701  -8.766  -2.861   7.158  42.186

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  45.845412  60.849115   0.753    0.455
speed       -18.962244  22.296088  -0.850    0.400
I(speed^2)    2.892190   2.719103   1.064    0.293
I(speed^3)   -0.151951   0.134225  -1.132    0.264
I(speed^4)    0.002799   0.002308   1.213    0.232

Residual standard error: 15.13 on 45 degrees of freedom
Multiple R-squared:  0.6835,    Adjusted R-squared:  0.6554
F-statistic:  24.3 on 4 and 45 DF,  p-value: 9.375e-11
```

```r
fit5 = lm(dist ~ speed + I(speed^2) + I(speed^3) + I(speed^4) + I(speed^5) ,
          data = cars)
summary(fit5)
```

```
Call:
lm(formula = dist ~ speed + I(speed^2) + I(speed^3) + I(speed^4) +
    I(speed^5), data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-23.370  -8.165  -2.395   7.294  42.342

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.650e+00  1.401e+02  -0.019    0.985
speed        5.484e+00  6.736e+01   0.081    0.935
I(speed^2)  -1.426e+00  1.155e+01  -0.124    0.902
I(speed^3)   1.940e-01  9.087e-01   0.214    0.832
I(speed^4)  -1.004e-02  3.342e-02  -0.300    0.765
I(speed^5)   1.790e-04  4.648e-04   0.385    0.702

Residual standard error: 15.27 on 44 degrees of freedom
Multiple R-squared:  0.6846,    Adjusted R-squared:  0.6487
F-statistic:  19.1 on 5 and 44 DF,  p-value: 4.65e-10
```

```r
R2_fit = summary(fit)$r.squared
R2_fit2 = summary(fit2)$r.squared
R2_fit3 = summary(fit3)$r.squared
R2_fit4 = summary(fit4)$r.squared
R2_fit5 = summary(fit5)$r.squared

adjR2_fit = summary(fit)$adj.r.squared
adjR2_fit2 = summary(fit2)$adj.r.squared
adjR2_fit3 = summary(fit3)$adj.r.squared
adjR2_fit4 = summary(fit4)$adj.r.squared
adjR2_fit5 = summary(fit5)$adj.r.squared

R2_vals = c(R2_fit, R2_fit2, R2_fit3, R2_fit4, R2_fit5)
adjR2_vals = c(adjR2_fit, adjR2_fit2, adjR2_fit3,
               adjR2_fit4, adjR2_fit5)
plot(1:5, R2_vals, type = "b", pch = 19, cex = 1.3,
```

```
      lwd = 2, col = "darkgrey", ylab = expression(R^2),
     xlab = "degree (p)", ylim = c(0.6, 0.7))
lines(1:5, adjR2_vals, col = "black", lwd = 2, type = "b", pch = 19)
legend("topleft", c("R squared", "adj R squared"),
       col = c("darkgrey", "black"), lwd = c(2,2), bty = "n")
```

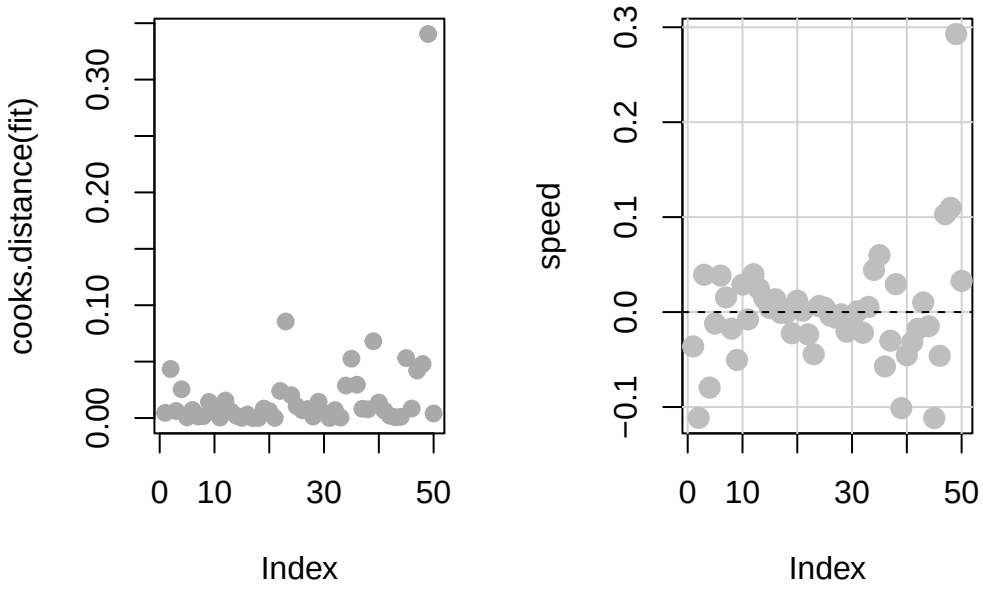



```
anova(fit, fit2, fit3, fit4, fit5)
```

```
Analysis of Variance Table

Model 1: dist ~ speed
Model 2: dist ~ speed + I(speed^2)
Model 3: dist ~ speed + I(speed^2) + I(speed^3)
Model 4: dist ~ speed + I(speed^2) + I(speed^3) + I(speed^4)
Model 5: dist ~ speed + I(speed^2) + I(speed^3) + I(speed^4) + I(speed^5)
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1     48 11354
2     47 10825  1    528.81 2.2671 0.1393
3     46 10634  1    190.35 0.8161 0.3712
4     45 10298  1    336.55 1.4428 0.2361
5     44 10263  1     34.59 0.1483 0.7020
```

Leverage and Influence Plot

```
par(mfrow = c(1,2))
plot(cooks.distance(fit), col = "darkgrey", pch = 19, cex = 1.1)
dfbetaPlots(fit, pch = 19, col = "grey", cex = 1.4)
```



A cross verification from the numerical method lectures

```
X = cbind(rep(1,nrow(cars)), cars$speed)
print(X)
```

```
      [,1] [,2]
 [1,]    1    4
 [2,]    1    4
 [3,]    1    7
 [4,]    1    7
 [5,]    1    8
```

```
 [6,]    1    9
 [7,]    1   10
 [8,]    1   10
 [9,]    1   10
[10,]    1   11
[11,]    1   11
[12,]    1   12
[13,]    1   12
[14,]    1   12
[15,]    1   12
[16,]    1   13
[17,]    1   13
[18,]    1   13
[19,]    1   13
[20,]    1   14
[21,]    1   14
[22,]    1   14
[23,]    1   14
[24,]    1   15
[25,]    1   15
[26,]    1   15
[27,]    1   16
[28,]    1   16
[29,]    1   17
[30,]    1   17
[31,]    1   17
[32,]    1   18
[33,]    1   18
[34,]    1   18
[35,]    1   18
[36,]    1   19
[37,]    1   19
[38,]    1   19
[39,]    1   20
[40,]    1   20
[41,]    1   20
[42,]    1   20
[43,]    1   20
[44,]    1   22
[45,]    1   23
[46,]    1   24
[47,]    1   24
[48,]    1   24
```

```
[49,]    1   24
[50,]    1   25
```

```r
Y = cars$dist
beta_star = (solve(t(X)%*%X))%*%t(X)%*%Y
print(beta_star)
```

```
           [,1]
[1,] -17.579095
[2,]   3.932409
```

Questions

- Using the `airquality` dataset, answer the following descriptive statistics questions:
  - Using `help(airquality)` command obtain basic description of the data and also check the source of the data set.
  - Calculate the mean, median, standard deviation, and range of the `Ozone` variable.
  - How many missing values are there in the `Ozone` variable?
  - Draw boxplot and histograms of each variable present in the `airquality` data. Using boxplot in R identify the outliers.
  - Calculate the average `Temp` for each month (May through September).
  - Which month recorded the highest average `Temp`?
  - What is the correlation between Ozone and `Solar.R`? Interpret the result.
  - Create a scatter plot of `Wind` vs. `Temp`. Does the plot suggest a relationship?
  - Plot a histogram of the `Wind` variable. Comment on the shape of the distribution.
  - What is the interquartile range (IQR) of `Wind`?
  - Identify the percentage of rows with missing values in the dataset.
  - Provide one strategy to handle missing values in the `Ozone` variable and explain its implications.
- The `rivers` dataset contains the lengths (in miles) of 141 major rivers in North America.
  - What is the mean, median, and standard deviation of the river lengths?
  - What is the range (minimum and maximum) of the river lengths?
  - Create a histogram of the river lengths. What is the shape of the distribution (e.g., symmetric, skewed)?
  - How many rivers are longer than 1,000 miles?
  - Create a boxplot of the river lengths. Are there any outliers?
  - What is the interquartile range (IQR) of the river lengths?
  - What percentage of rivers fall within one standard deviation of the mean?

  - Apply a log transformation to the river lengths. How does the histogram of the transformed data compare to the original?
  - Why might a log transformation be useful for this dataset?
  - What is the mean length of the 10 shortest rivers?
  - What is the mean length of the 10 longest rivers?
  - How does the variability (standard deviation) compare between the shortest and longest rivers?
  - Create a density plot of the river lengths. How does it compare to the histogram?
  - Overlay a rug plot on the histogram. Does it provide additional insights about the data distribution?
  - Compare the distribution of river lengths to a normal distribution using a Q-Q plot. What do you observe?
  - Test the normality of the river length data using the Shapiro-Wilk test. What is the result?
  - Identify any regions or patterns in the data that suggest natural groupings or clusters of river lengths.

- The `EuStockMarkets` dataset contains daily closing prices of major European stock indices from 1991 to 1998.
  - What are the mean, median, standard deviation, and range of closing prices for each stock index (`DAX`, `SMI`, `CAC`, and `FTSE`)?
  - Which stock index has the highest average closing price over the entire dataset?
  - What is the minimum and maximum closing price for each index?
  - Create histograms for the closing prices of each stock index.
  - Which indices show skewed distributions?
  - Plot density plots for the indices. Do they suggest unimodal or multimodal distributions?
  - Use boxplots to compare the distributions of the four stock indices. Are there any outliers?
  - (Temporal trends) Plot the time series of the closing prices for each index. What trends do you observe?
  - Identify the periods of significant increases or decreases for each stock index.
  - Which index shows the highest volatility (largest fluctuations) over time?
  - (Correlation analysis) Compute the pairwise correlations between the indices. Which pair of indices has the strongest correlation?
  - Create a scatterplot matrix of the indices. Do the relationships appear linear?
  - Conduct a hypothesis test to determine if the correlation between DAX and CAC is significantly different from zero.
  - (Volatility) Calculate the daily returns $((P_t - P_{t-1})/P_{t-1})$ for each index. Which index has the highest average daily return?
  - Plot the time series of daily returns. Which index shows the most frequent large spikes?
  - Identify the days with the largest positive and negative returns for each index.

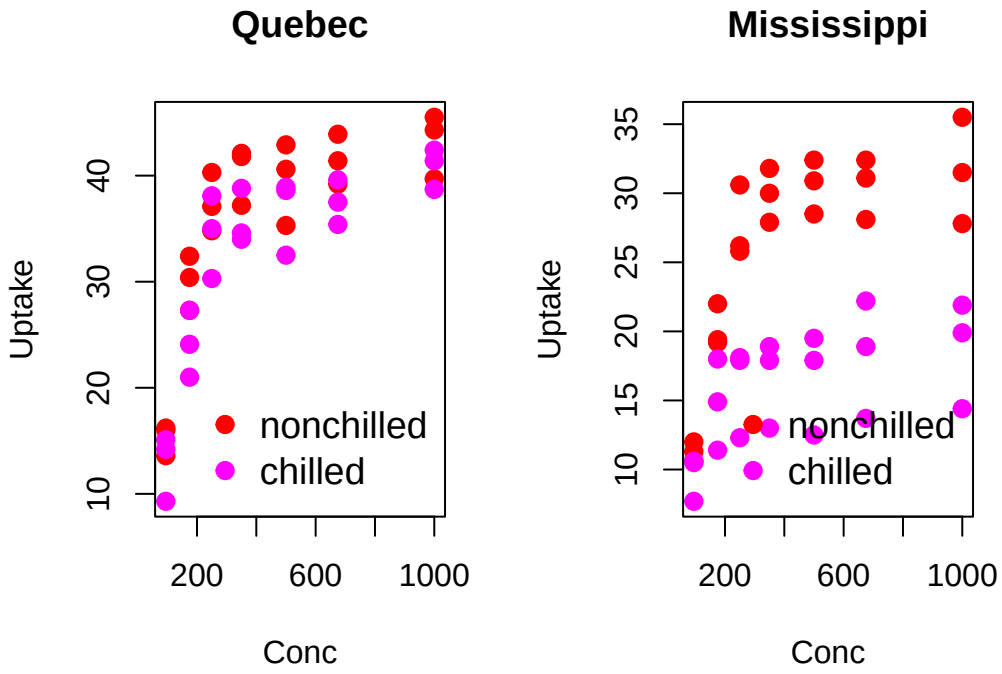
  - Use Q-Q plots to assess whether the closing prices of each index follow a normal distribution. Which indices deviate most from normality?
  - Suggest real-world economic events during the dataset period (1991–1998) that might have influenced the trends in these indices.

- The `CO2` dataset contains data on carbon dioxide uptake in grass plants under varying environmental conditions. The `CO2` uptake of six plants from Quebec and six plants from Mississippi was measured at several levels of ambient `CO2` concentration. Half the plants of each type were chilled overnight before the experiment was conducted.

  - What is the mean, median, standard deviation, and range of the variable `uptake`?
  - What is the mean and standard deviation of `conc` across all plants?
  - Which plant has the highest carbon dioxide uptake (`uptake`)?
  - (Comparison among groups) What is the average uptake for each Type of plant (`Quebec` and `Mississippi`)?
  - Compare the mean and median uptake for plants under different Treatment conditions (`nonchilled` vs `chilled`).
  - Calculate the mean uptake for each combination of `Type` and `Treatment`.
  - Which group has the highest mean uptake?
  - (Understanding individual distributions) Create histograms of uptake for the two Type groups (`Quebec` and `Mississippi`).
  - How do the distributions differ?
  - Plot a boxplot of uptake by Treatment. Are there any outliers?
  - Create density plots of uptake for the two Type groups. What do you observe about their shapes?
  - Create a scatter plot of `uptake` vs. `conc`. What kind of relationship do you observe?
  - Fit a linear model of `uptake` as a function of `conc`. What does the slope of the line indicate?
  - Does the relationship between `uptake` and `conc` differ across `Type` or `Treatment`? Visualize and explain.
  - (Advanced Visualizations) Create a faceted scatterplot of `uptake` vs. `conc` by Type and Treatment. What patterns emerge?
  - Create a heatmap of the average `uptake` for each combination of `Type`, `Treatment`, and `conc` ranges. What does this visualization reveal?
  - Plot boxplots of `uptake` grouped by individual Plant. Which plants have the highest median and variability?
  - (Transformation) Apply a log transformation to `uptake`. How does the distribution change?
  - Group `conc` into intervals (e.g., 0–200, 201–400, etc.) and calculate the average `uptake` for each interval. What trend do you observe?

```
DQ = CO2[1:42, ]
DM = CO2[43:84, ]
```

```r
par(mfrow = c(1,2))
plot(DQ$conc, DQ$uptake, type = "p",
     col = ifelse(DQ$Treatment =="nonchilled", "red", "magenta"),
     pch = 19, cex = 1.2, main = "Quebec",
     xlab = "Conc", ylab = "Uptake")
legend("bottomright", legend = c("nonchilled", "chilled"),
       col = c("red", "magenta"), pch = c(19, 19),
       cex = c(1.2, 1.2), bty = "n")
plot(DM$conc, DM$uptake, type = "p",
     col = ifelse(DQ$Treatment =="nonchilled", "red", "magenta"),
     pch = 19, cex = 1.2, main = "Mississippi",
     xlab = "Conc", ylab = "Uptake")
legend("bottomright", legend = c("nonchilled", "chilled"),
       col = c("red", "magenta"), pch = c(19, 19),
       cex = c(1.2, 1.2), bty = "n")
```



Regression Models and Interdisciplinary Applications

Least squares method

At the outset of this chapter, I wish to share a note of pedagogical intent, particularly for my fellow teachers. Linear regression is often introduced in classrooms through the use of built-in functions and statistical packages in R or other software. However, the demonstrations in this book deliberately avoid reliance on such packages. Instead, we focus on applying the formulas derived by students themselves during classroom instruction.

The core idea is to bridge the gap between students' immediate mathematical understanding and real-world datasets. As a result, many of the formulas in this chapter are expressed directly using the design matrix, reflecting the theoretical framework students have worked with in class.

This approach was developed and implemented for the MDM students of the Machine Learning and Artificial Intelligence minor at ICT, Mumbai (batch 2023–2027). In practice, I observed that when students used R in the lab, they tended to memorize functions such as `lm()` and `summary()` without grasping how these relate to the theory learned in class. R became, for them, a disconnected tool—separate from the mathematical foundation.

Therefore, throughout this chapter, we rely on formulas derived and understood by the students themselves. They have not only learned these derivations but also implemented them in the classroom, reinforcing the deep connection between theory and computation.

Suppose that we are given freezing point of different ethylene glycol - water mixtures. We are interested in determining the relationship between the freezing point of ethylene glycol and the weight percent of ethylene glycol in a water solution shown in the following data:

```r
glycol = c(0.00, 5.09, 11.3, 15.47, 20.94, 30.97, 31.22,
           36.62, 42.76, 48, 49.34, 51.36, 56.36, 59.05)
freezing_point = c(273.15, 267.46, 258.5, 251.72, 241.58,
                   225.28, 225.49, 228.03, 229.89, 230.5,
                   230.54, 230.37, 232.12, 234.62)
data = data.frame(glycol, freezing_point)
```

Suppose that we want to address questions of the form:

- When we have 33.3 wt% of ethylene glycol in a solution, what is the freezing point?
- What is the uncertainty associated with the prediction of the freezing point?

First we check out basic scatterplot which usually guide us to choose appropriate equation for the modelling the relationship between the response and the predictor variable(s). The scatterplot indicates that a linear relationship does not seem to an appropriate choice and a higher order polynomial may need to be considered. However, as a starting point, we shall discuss the least squares method for linear function and extrapolate those ideas for polynomial equations.

```
plot(glycol, freezing_point, type = "p", pch = 19,
     col = "darkgrey", xlab = "Mole percent ethylene glycol",
     ylab = "Freezing point (K)", cex = 1.4)
```

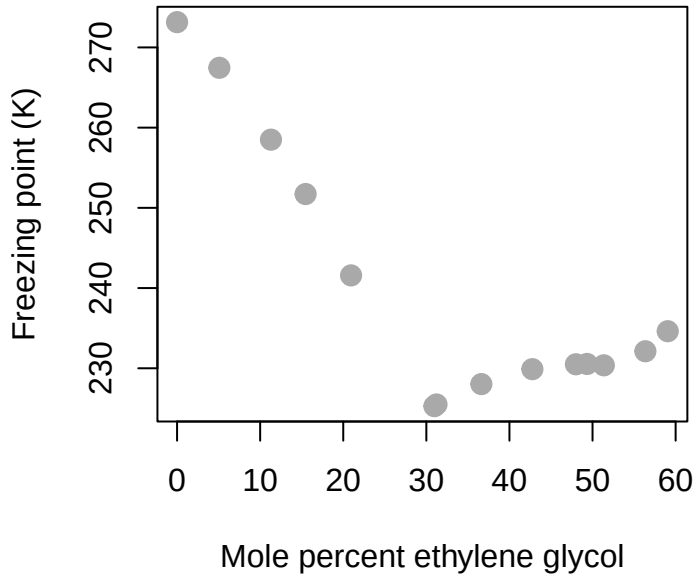


Figure 1: Scatterplot of the predictor and the response variable.

Using the `summary` function, we can obtain a basic descriptive summary of the variables in the given data set.

```
summary(data)                    # basic data summary
```

```
     glycol      freezing_point
```

```
 Min.   : 0.00   Min.   :225.3
 1st Qu.:16.84   1st Qu.:230.0
 Median :33.92   Median :231.3
 Mean   :32.75   Mean   :239.9
 3rd Qu.:49.01   3rd Qu.:249.2
 Max.   :59.05   Max.   :273.1
```

Minimization of the error sum of squares

In the above plot, it is evident that the relationship does not seem to be linear. However, at a starting point, we shall try to fit a linear function of the form

$$y = b_0 + b_1 x + e,$$

where the parameters b_0 and b_1 are to be estimated from the given data. The term e denotes the error which typically denotes the different between the observed response and the predicted values obtained by the fitted equation.

Suppose that we denote the response (freezing point) using the symbol y and Mole percent ethylene glycol is represented using the symbol x. There are $n = 14$ data points. The expression for the error sum of squares is given by

$$E(b_0, b_1) = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2$$

We want to choose values of b_0 and b_1 so that $E(b_0, b_1)$ is minimized. Let us try to have some visual display of the error sum of squares as a function of b_0 and b_1.

```
ESS = function(b0,b1){
  return(sum((freezing_point - b0 - b1*glycol)^2))
}
```

Let us plot the surface with respect to b_0 and b_1.

```
b0_vals = seq(-100, 300, by = 2)
b1_vals = seq(-100, 300, by = 2)
ESS_vals = matrix(data = NA, nrow = length(b0_vals), ncol = length(b1_vals))
for(i in 1:length(b0_vals)){
  for(j in 1:length(b1_vals)){
    ESS_vals[i,j] = ESS(b0_vals[i], b1_vals[j])
  }
}
```

```r

library(plot3D)
persp3D(b0_vals, b1_vals, ESS_vals, phi = 20, theta = 120,
      xlab = "b0", ylab = "b1", zlab = "ESS")
```

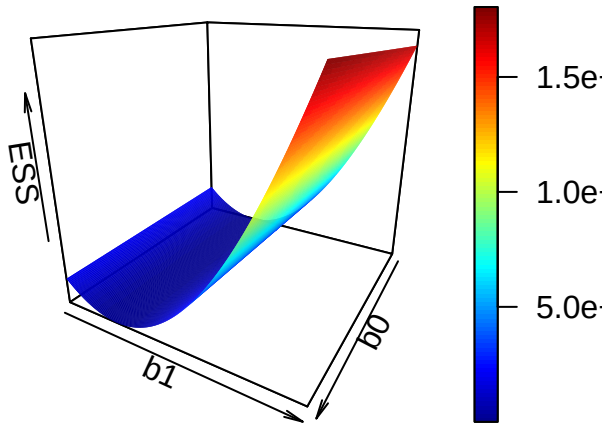


Figure 2: The shape of the residual sum of squares as a function of the model parameters. The right panel shows the surface with colour gradients.

Matrix Notation

We adopt some matrix notation for computation

- The response variable $Y = (y_1, y_2, \dots, y_n)'$. Therefore, $Y_{n\times 1}$ is a column vector and n is the number of rows in the data.
- The design matrix X. In this case, we have the predictor variable $x = (x_1, x_2, \dots, x_n)'$, a column vector. If the linear equation contains an intercept term b_0, we consider the design matrix as

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

- Parameter vector $b = (b_0, b_1)'$ which is a 2-dimensional column vector.

The linear model now can be represented as

$$Y = Xb$$

To find the best choice of b, we need to minimize the error sum of squares with respect to b and in matrix notation the error sum of squares is given by

$$ESS(b) = (Y - Xb)'(Y - Xb)$$

We compute $\frac{\partial}{\partial b}\text{ESS}(b)$ and equal to zero which gives the system of linear equations as (in matrix form)

$$X'Y = (X'X)b$$

If $\det(X'X) \neq 0$, then the least squares estimate of b is obtained as

$$\hat{b} = (X'X)^{-1}X'Y$$

```r
Y = freezing_point
X = cbind(rep(1, length(Y)), glycol)
print(X)
```

```
        glycol
 [1,] 1   0.00
 [2,] 1   5.09
 [3,] 1  11.30
 [4,] 1  15.47
 [5,] 1  20.94
 [6,] 1  30.97
 [7,] 1  31.22
 [8,] 1  36.62
 [9,] 1  42.76
[10,] 1  48.00
[11,] 1  49.34
[12,] 1  51.36
[13,] 1  56.36
[14,] 1  59.05
```

```r
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
print(b_hat)
```

```
              [,1]
      262.2041053
glycol  -0.6796534
```

```r
plot(glycol, freezing_point, type = "p", pch = 19,
     col = "darkgrey", xlab = "Mole percent ethylene glycol",
     ylab = "Freezing point (K)", cex = 1.4)
curve(b_hat[1]+b_hat[2]*x, add = TRUE, col = "red", lwd = 2)
Y_predicted = X%*%b_hat
points(glycol, Y_predicted, pch = 19,
       col = "blue", cex = 1.4)
legend("topright", bty = "n", legend = c("observed", "predicted"),
       col = c("darkgrey", "blue"), pch = c(19,19),
       cex = c(1.4, 1.4), lwd = 1, lty = c(NA, NA))

# prediction at glycol = 33.3
glycol_star = 33.3
Y_star = b_hat[1]+b_hat[2]*glycol_star
points(glycol_star, Y_star, pch = 18, col = "magenta", cex = 1.5)
```

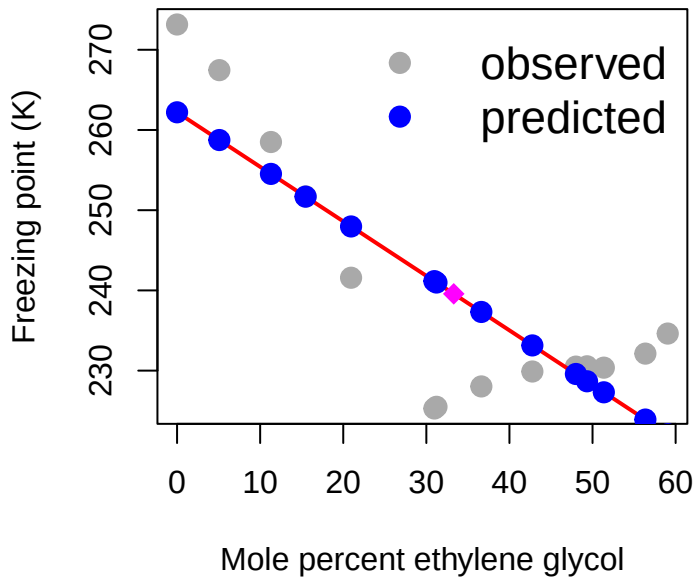


Figure 3: The least square regression plot of a linear function

```r
error = Y - Y_predicted                # computation of error
ess_hat = sum(error^2)
print(ess_hat)
```

```
[1] 1080.218
```

```r
par(mfrow = c(1,2))
plot(Y_predicted, error, pch = 19, col = "red",
     xlab = expression(widehat(Y)), ylab = expression(widehat(e)))
abline(h = 0, col = "grey", lwd = 2, lty = 2)
hist(error, probability = TRUE, xlab = expression(widehat(e)))
```

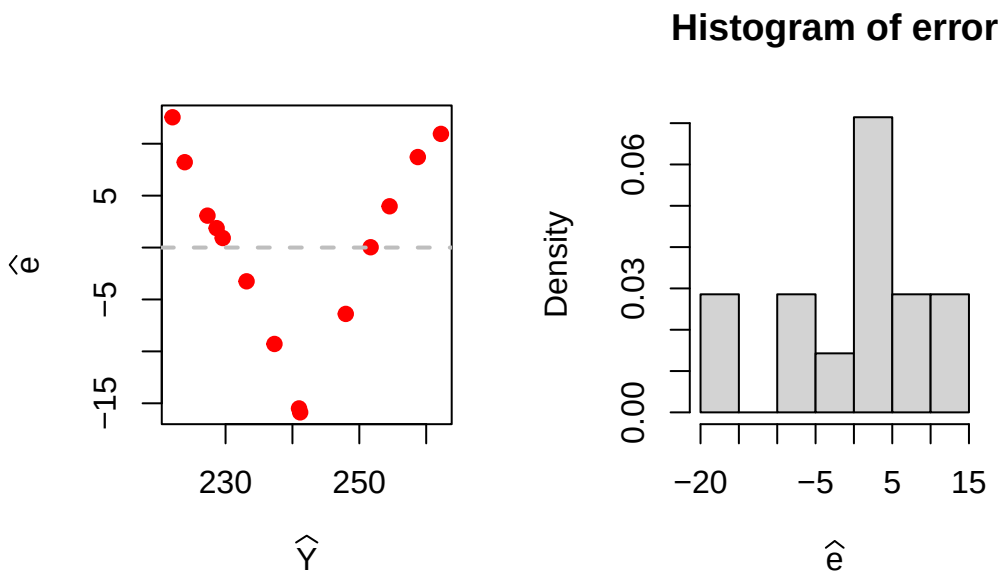


Figure 4: Distribution of error for the fitted linear equation. The desiarable property is the the histogram should be well approximated by the normal distribution.

A patterned relationship between the residuals and the fitted values gives an indication of the possible existence of nonlinear relationship between the predictor and the response.

Estimating the parameters using R

```r
fit = lm(glycol ~ freezing_point, data = data)
summary(fit)
```

```
Call:
lm(formula = glycol ~ freezing_point, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
```

```
-16.4434  -7.4094  -0.0959   6.8410  20.9756

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    272.6694    47.6565   5.722 9.58e-05 ***
freezing_point  -0.9999     0.1982  -5.045 0.000287 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.51 on 12 degrees of freedom
Multiple R-squared:  0.6796,    Adjusted R-squared:  0.6529
F-statistic: 25.45 on 1 and 12 DF,  p-value: 0.000287
```

Least Squares for quadratic function

We consider the fitting of function of the form

$$y = b_0 + b_1 x + b_2 x^2 + e.$$

We need to minimize the error sum of squares

$$\text{ESS}(b_0, b_1, b_2) = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i - b_2 x_i^2)^2.$$

The design matrix X is given by

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix}$$

and the parameter vector $b = (b_0, b_1, b_2)'$.

```r
Y = freezing_point
X = cbind(rep(1, length(Y)), glycol, glycol2 = glycol^2)
print(X)
```

```
      glycol   glycol2
 [1,] 1   0.00    0.0000
 [2,] 1   5.09   25.9081
 [3,] 1  11.30  127.6900
 [4,] 1  15.47  239.3209
 [5,] 1  20.94  438.4836
```

```
 [6,] 1  30.97  959.1409
 [7,] 1  31.22  974.6884
 [8,] 1  36.62 1341.0244
 [9,] 1  42.76 1828.4176
[10,] 1  48.00 2304.0000
[11,] 1  49.34 2434.4356
[12,] 1  51.36 2637.8496
[13,] 1  56.36 3176.4496
[14,] 1  59.05 3486.9025
```

```r
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
cat("Estimated coefficients b0, b1 and b2 are")
```

```
Estimated coefficients b0, b1 and b2 are
```

```r
print(b_hat)
```

```
                [,1]
        277.47637633
glycol   -2.36315724
glycol2   0.02793794
```

```r
plot(glycol, freezing_point, type = "p", pch = 19,
     col = "darkgrey", xlab = "Mole percent ethylene glycol",
     ylab = "Freezing point (K)", cex = 1.4)

curve(b_hat[1]+b_hat[2]*x + b_hat[3]*x^2,
      add = TRUE, col = "red", lwd = 2)           # fitted equation
Y_predicted = X%*%b_hat
points(glycol, Y_predicted, pch = 19,
       col = "blue", cex = 1.4)
legend("topright", bty = "n", legend = c("observed", "predicted"),
       col = c("darkgrey", "blue"), pch = c(19,19),
       cex = c(1.4, 1.4), lwd = 1, lty = c(NA, NA))
```

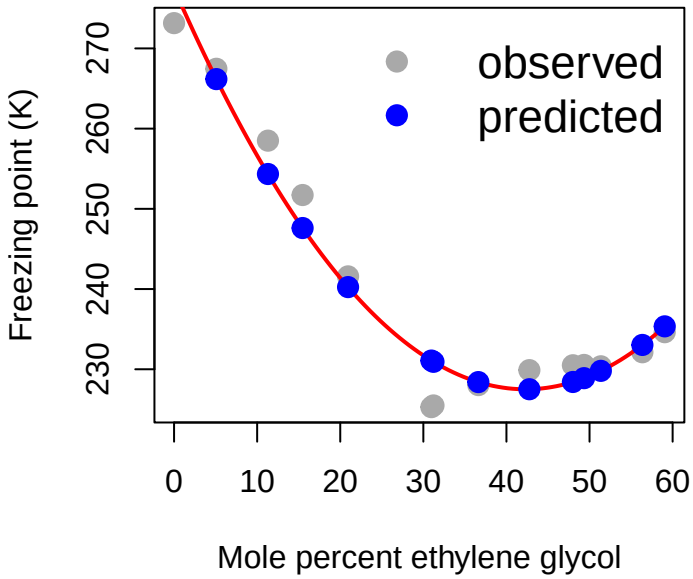


Figure 5: The least squares fitting using a quadratic function

```
error = Y - Y_predicted                # computation of error
ess_hat = sum(error^2)                 # estimated ESS
print(ess_hat)
```

```
[1] 134.2652
```

```
par(mfrow = c(1,2))
plot(Y_predicted, error, pch = 19, col = "red",
     xlab = expression(widehat(Y)), ylab = expression(widehat(e)))
abline(h = 0, col = "grey", lwd = 2, lty = 2)
hist(error, probability = TRUE, xlab = expression(widehat(e)))
```

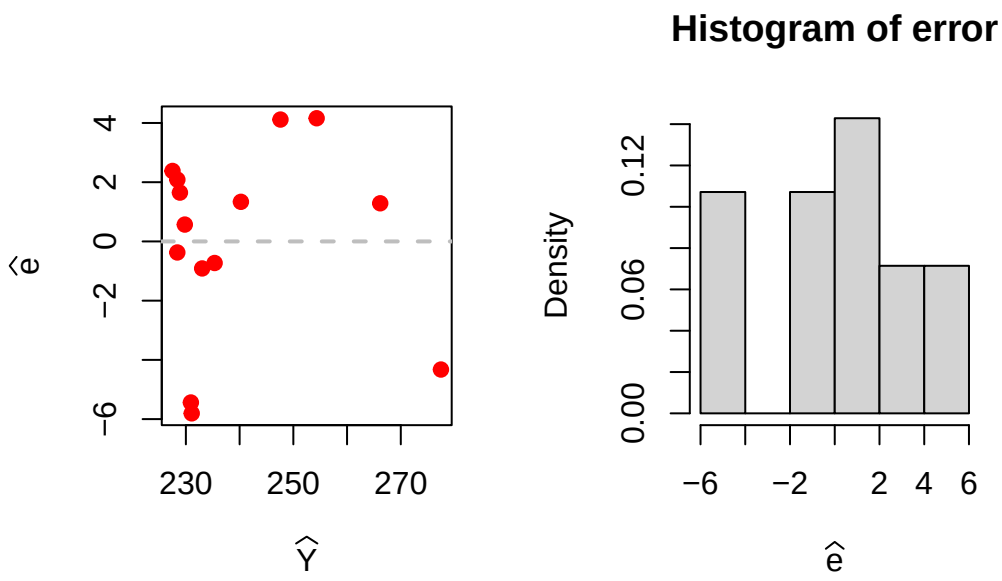


Figure 6: Distribution of error for the fitted quadratic equation

Least square method for cubic regression

We consider the fitting of function of the form

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + e.$$

```
Y = freezing_point
X = cbind(rep(1, length(Y)), glycol, glycol2 = glycol^2, glycol3 = glycol^3)
print(X)
```

```
      glycol   glycol2     glycol3
 [1,] 1   0.00    0.0000      0.0000
 [2,] 1   5.09   25.9081    131.8722
 [3,] 1  11.30  127.6900   1442.8970
 [4,] 1  15.47  239.3209   3702.2943
 [5,] 1  20.94  438.4836   9181.8466
 [6,] 1  30.97  959.1409  29704.5937
 [7,] 1  31.22  974.6884  30429.7718
 [8,] 1  36.62 1341.0244  49108.3135
 [9,] 1  42.76 1828.4176  78183.1366
[10,] 1  48.00 2304.0000 110592.0000
[11,] 1  49.34 2434.4356 120115.0525
```

```
[12,] 1  51.36 2637.8496 135479.9555
[13,] 1  56.36 3176.4496 179024.6995
[14,] 1  59.05 3486.9025 205901.5926
```

```r
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
cat("Estimated coefficients for b0, b1, b2 and b3 are \n")
```

```
Estimated coefficients for b0, b1, b2 and b3 are
```

```r
print(b_hat)
```

```
                [,1]
         2.760038e+02
glycol  -1.985367e+00
glycol2  1.161896e-02
glycol3  1.819175e-04
```

```r
plot(glycol, freezing_point, type = "p", pch = 19,
     col = "darkgrey", xlab = "Mole percent ethylene glycol",
     ylab = "Freezing point (K)", cex = 1.4)

curve(b_hat[1]+b_hat[2]*x + b_hat[3]*x^2 + b_hat[4]*x^3,
      add = TRUE, col = "red", lwd = 2)
Y_predicted = X%*%b_hat
points(glycol, Y_predicted, pch = 19,
       col = "blue", cex = 1.4)
legend("topright", bty = "n", legend = c("observed", "predicted"),
       col = c("darkgrey", "blue"), pch = c(19,19),
       cex = c(1.4, 1.4), lwd = 1, lty = c(NA, NA))
```

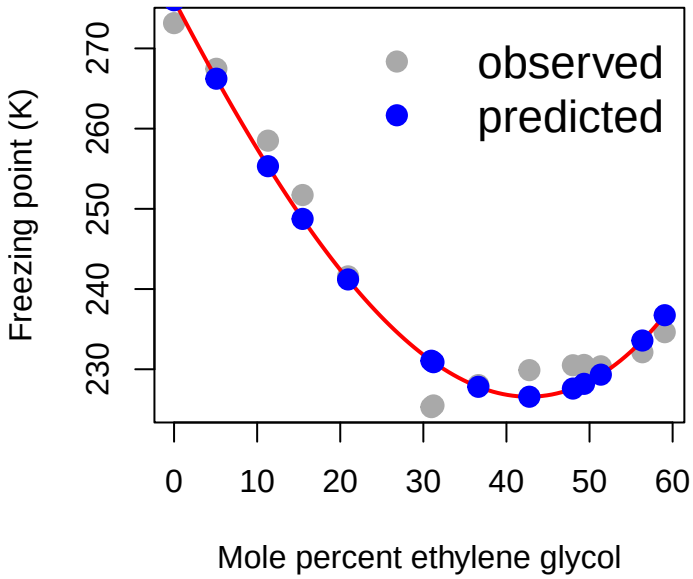


Figure 7: The least squares fitting using a cubic function

```
error = Y - Y_predicted                # computation of error
ess_hat = sum(error^2)
print(ess_hat)
```

```
[1] 124.0845
```

```
par(mfrow = c(1,2))
plot(Y_predicted, error, pch = 19, col = "red",
     xlab = expression(widehat(Y)), ylab = expression(widehat(e)))
abline(h = 0, col = "grey", lwd = 2, lty = 2)
hist(error, probability = TRUE, xlab = expression(widehat(e)))
```

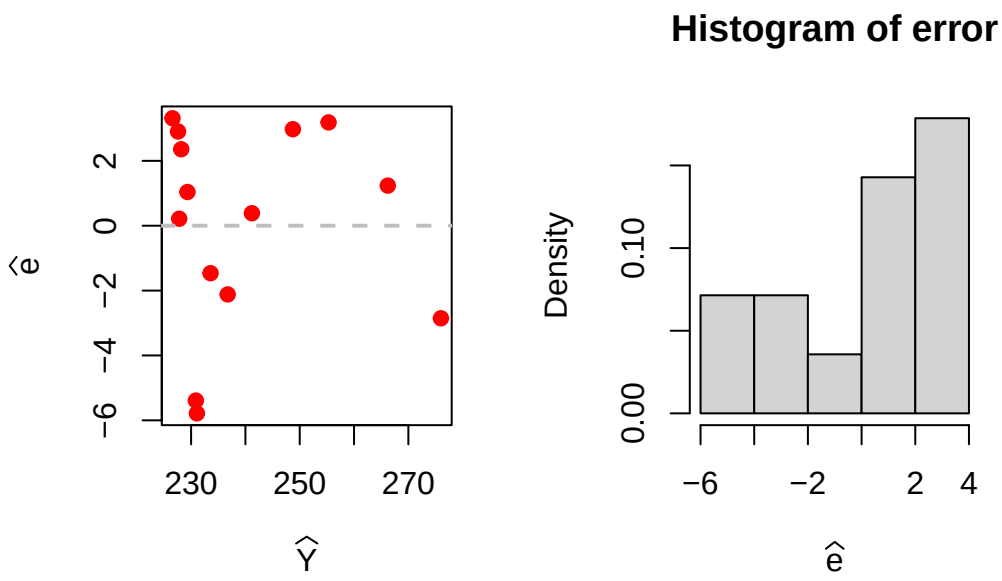


Figure 8: Distribution of error for the fitted cubic equation equation

Case Study (Modelling CO2 Uptake in Grass Plants)

In the above discussions, we have not discussed the statistical properties of the estimators of the parameters for the simple linear regression model. To discuss the statistical properties, we first need to make some assumptions about the distribution of the data, or more precisely the response which is expected to change as a function of the input variable.

We consider the CO2 data set available in the `datasets` package in R and use a smaller part of the data as demonstration. The CO2 data frame has 84 rows and 5 columns of data from an experiment on the cold tolerance of the grass species *Echinochloa crus-galli*. For demonstration we consider only the uptake of one plant at different concentration level.

```
1 head(datasets::CO2)
```

```
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
```

```r
# help("CO2")

plot(CO2$conc, CO2$uptake, col = "grey",
     pch = 19, cex =1.2)
cor(CO2$conc, CO2$uptake)
```

```
[1] 0.4851774
```

```r
data = CO2[1:21,c("conc", "uptake")]
head(data)
```

```
  conc uptake
1   95   16.0
2  175   30.4
3  250   34.8
4  350   37.2
5  500   35.3
6  675   39.2
```

```r
dim(data)
```

```
[1] 21  2
```

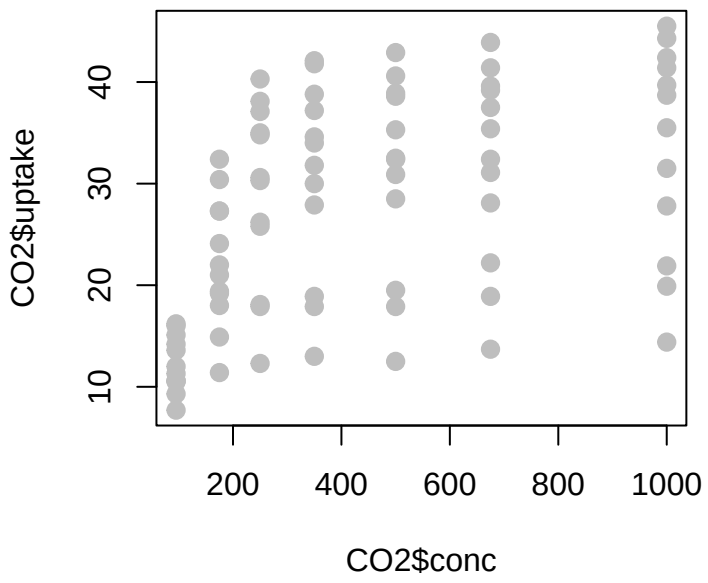


Figure 9

We consider the following statistical model which is given by

$$\text{uptake}_i|\text{conc}_i \sim \mathcal{N}\left(b_0 + b_1 \times \text{conc}_i, \sigma^2\right)$$

and `uptake` by the plant are different concentration levels are statistically independent. In other words, the distribution of `uptake` at different concentration are independently normally distributed with constant variance and its expectation varies with the concentration following a linear function.

- Write down the likelihood function of for estimating the parameters b_0, b_1 and σ^2.
- Analytically derive that the likelihood function is maximized at

$$b_1^* = \frac{S_{xy}}{S_{xx}}, \quad b_0^* = \bar{y} - b_1^*\bar{x}, \quad \sigma^{2*} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - b_0^* - b_1^* x_i\right)^2,$$

  where the notations have their usual meaning.
- Using the `optim()` function in R, maximize the function and report the estimates.
- Also report the standard error of the estimators.

The following R codes may be used to compute the estimates.

```
S_xx = sum((data$conc - mean(data$conc))^2)
S_yy = sum((data$uptake - mean(data$uptake))^2)
S_xy = sum((data$conc - mean(data$conc))*(data$uptake - mean(data$uptake)))
b1_star = S_xy/S_xx
b0_star = mean(data$uptake) - b1_star*mean(data$conc)
sigma2_star = sum((data$uptake - b0_star - b1_star*data$conc)^2)/nrow(data)
```

Using the following R codes, we can draw the fitted line.

```
plot(data$conc, data$uptake, pch = 19,
     col = "red", cex = 1.2)
conc_vals = seq(90, 1010, by = 5)            # creating fitted lines
uptake_vals = b0_star + b1_star*conc_vals    # predicted uptake values
lines(conc_vals, uptake_vals, lty = 2, col = "blue",
      lwd = 3)                               # add fitted line
```

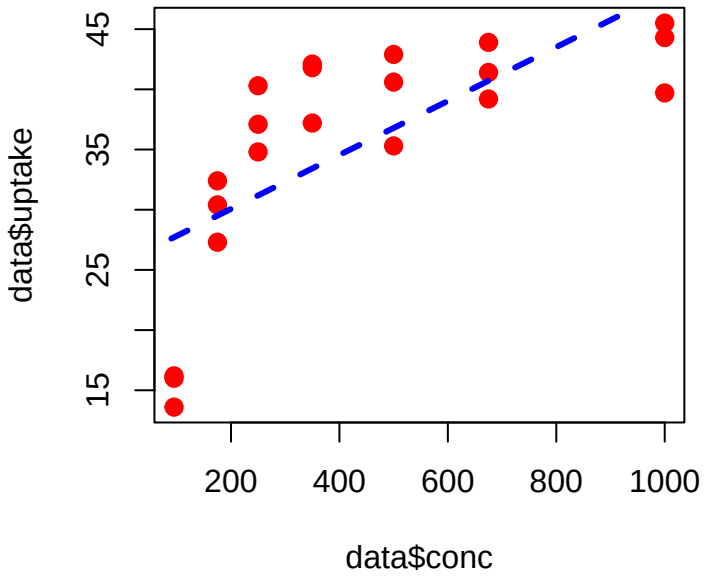


Figure 10: The CO2 uptake is modelled as a linear function of the concentration.

In the following, we check whether the model assumptions are met or not.

```
par(mfrow = c(1,2))
eps_star = data$uptake - b0_star - b1_star*data$conc  # error
hist(eps_star, probability = TRUE,
     xlab =expression(widehat(epsilon)), main = " ")
curve(dnorm(x, 0, sd = sqrt(sigma2_star)), col = "red",
      lwd = 2, add = TRUE)
# test for normality
shapiro.test(eps_star)
```

```
    Shapiro-Wilk normality test

data:  eps_star
W = 0.92759, p-value = 0.1232
```

```
plot(data$conc, eps_star, pch = 19, col = "red",
     lwd = 2, cex = 1.3, ylab = expression(widehat(epsilon)))
abline(h = 0, lwd = 3, col = "darkgrey", lty = 2)
```

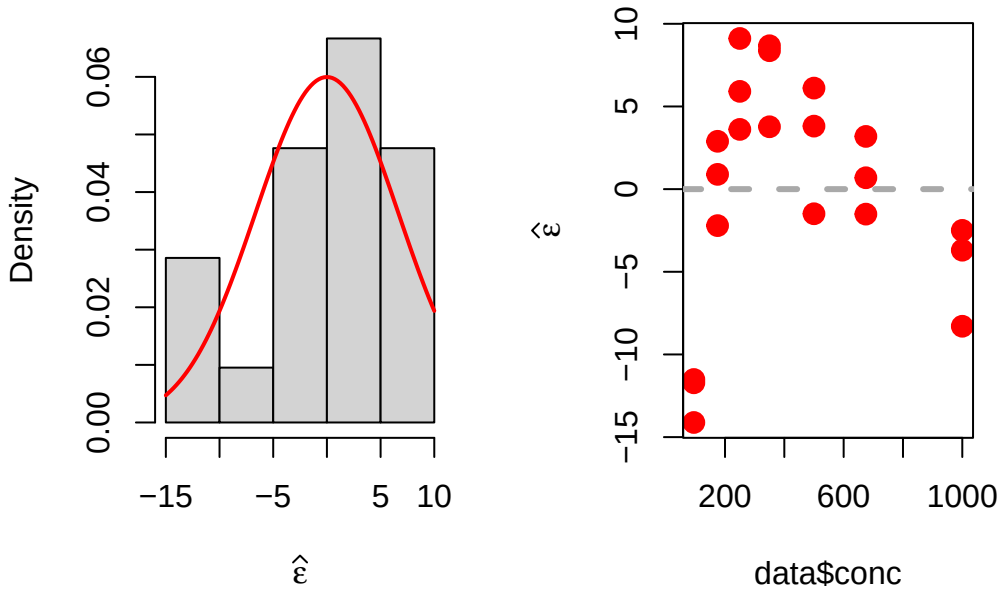


Figure 11: The left panel indicates the distribution of the residuals. A normal density is overlaid with mean 0 and variance σ^2*. The right panel depicts the estimated error at different concentration level. The figure indicates that ath lower concentration level model predictions have positive bias, means over estimation of the actual uptake level. The same is observed at the maximum concentration. However, at the moderate concentration (between 300 - 500), the residuals are positive the predicted uptake levels seem to underestimate the actual uptake.

More case studies

We consider the `cars` data set.

```
1 head(datasets::cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

```r
# help("cars")
names(cars)
```

```
[1] "speed" "dist"
```

```r
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")
n = nrow(cars)          # number of rows
Y = cars$dist           # response
# rep(1,n)
```

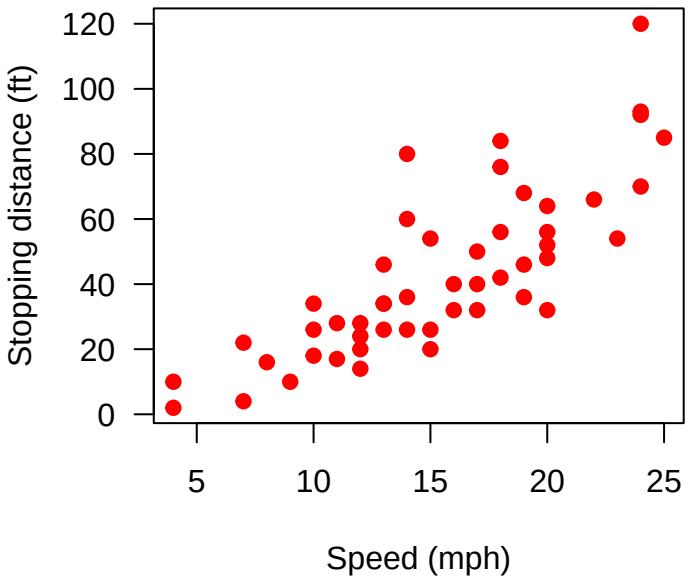


Figure 12

Linear and quadratic regression

In the following R codes, we build the linear and quadratic regression model using matrix notations.

```r
X = matrix(data = c(rep(1,n), cars$speed), nrow = n, ncol = 2)
# print(X)
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")

```

```r
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
print(b_hat)
```

```
           [,1]
[1,] -17.579095
[2,]   3.932409
```

```r
curve(b_hat[1] + b_hat[2]*x, add = TRUE,
      col = "blue", lwd = 2)

# quadratic regression

X = matrix(data = c(rep(1,n), cars$speed, cars$speed^2),
           nrow = n, ncol = 3)
# print(X)

b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
print(b_hat)
```

```
          [,1]
[1,] 2.4701378
[2,] 0.9132876
[3,] 0.0999593
```

```r
curve(b_hat[1] + b_hat[2]*x + b_hat[3]*x^2, add = TRUE,
      col = "magenta", lwd = 2)

legend("topleft", legend = c("linear", "quadratic"),
       lwd = c(2,2), col = c("blue", "magenta"), bty = "n")
```

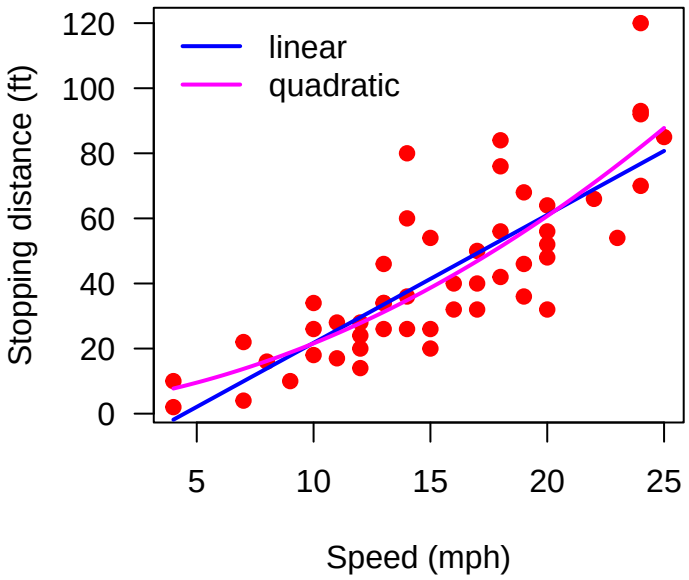


Figure 13: Fitting of linear and quadratic regression equation. For a visual display it seems that the quadratic equation is a better fit as compared to the linear regression. However, we shall keep in mind that quadratic equation contains more parameter, hence it is more complex. After performing apporpriate diagnostics, we should be able to choose which model should be considered.

Making predictions

```
# Making predictions (Linear)
par(mfrow = c(1,2))
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")
X = matrix(data = c(rep(1,n), cars$speed), nrow = n, ncol = 2)
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
curve(b_hat[1] + b_hat[2]*x, add = TRUE,
      col = "blue", lwd = 2)
H = X%*%(solve(t(X)%*%X))%*%t(X)      # Projection Matrix
dist_hat = H%*%Y
points(cars$speed, dist_hat, col = "green", pch = 19,
       cex = 1.2)
legend("topleft", legend = c("observed", "predicted"),
       col = c("red", "green"), cex = c(1,1), bty = "n", pch = c(19,19))
e_hat = cars$dist - dist_hat
e_hat = (diag(1, nrow = n) - H)%*%Y
```

```r
# print(e_hat)

hist(e_hat, probability = TRUE, xlab = "residuals",
     main = "Linear")
shapiro.test(e_hat)
```

```
    Shapiro-Wilk normality test

data:  e_hat
W = 0.94509, p-value = 0.02152
```

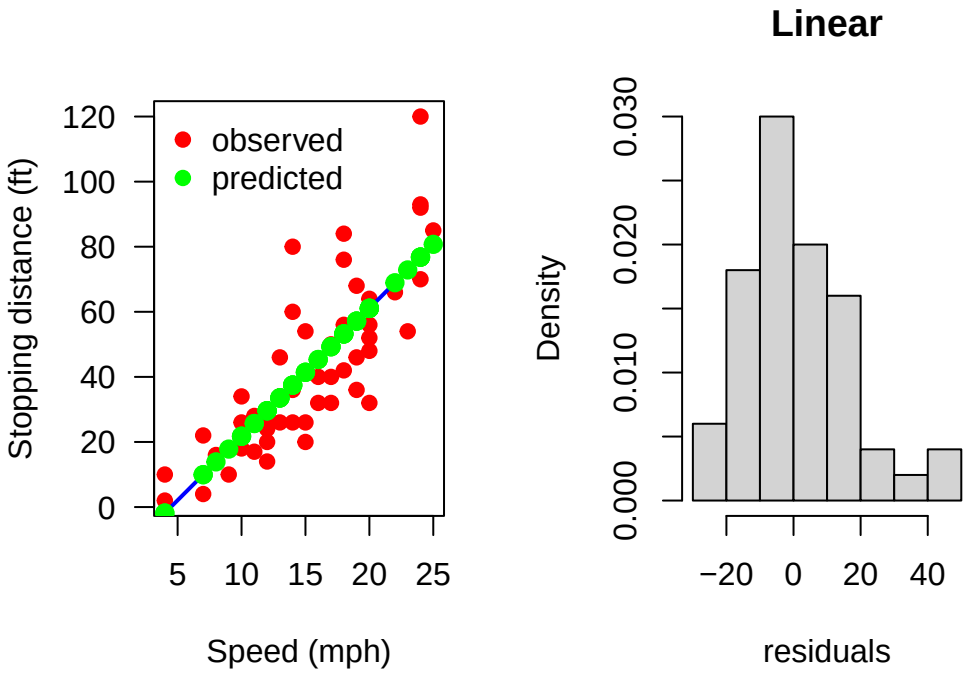


Figure 14: The least squares fit of the linear regression model and the predicted values of the response are shown in green color. The right panel indicates the distribution of the residuals.

In the following, one can check the properties of the projection matrix H which are $H' = H$ and $H^2 = H$.

```r
print(H)
t(H)
```

```r
H[34,2] == t(H)[34,2]
sum(abs(H - t(H)) < 10^(-3))
sum(abs(H%*%H - H)>10^(-15))
```

Making predictions (Quadratic)

In the following code, we make the predictions using the quadratic regression model

```r
par(mfrow = c(1,2))
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")
X = matrix(data = c(rep(1,n), cars$speed, cars$speed^2),
           nrow = n, ncol = 3)
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
curve(b_hat[1] + b_hat[2]*x + b_hat[3]*x^2, add = TRUE,
      col = "magenta", lwd = 2)
H = X%*%(solve(t(X)%*%X))%*%t(X)     # projection matrix
# print(H)
# t(H)

H[34,2] == t(H)[34,2]
```

```
[1] TRUE
```

```r
sum(abs(H - t(H)) < 10^(-3))
```

```
[1] 2500
```

```r
sum(abs(H%*%H - H)>10^(-15))
```

```
[1] 57
```

```r
dist_hat = H%*%Y
points(cars$speed, dist_hat, col = "green", pch = 19,
       cex = 1.2)
e_hat = cars$dist - dist_hat
e_hat = (diag(1, nrow = n) - H)%*%Y
# print(e_hat)

```

```r
hist(e_hat, probability = TRUE, xlab = "residuals",
     main = "Quadratic")
shapiro.test(e_hat)
```

```
    Shapiro-Wilk normality test

data:  e_hat
W = 0.93419, p-value = 0.007988
```

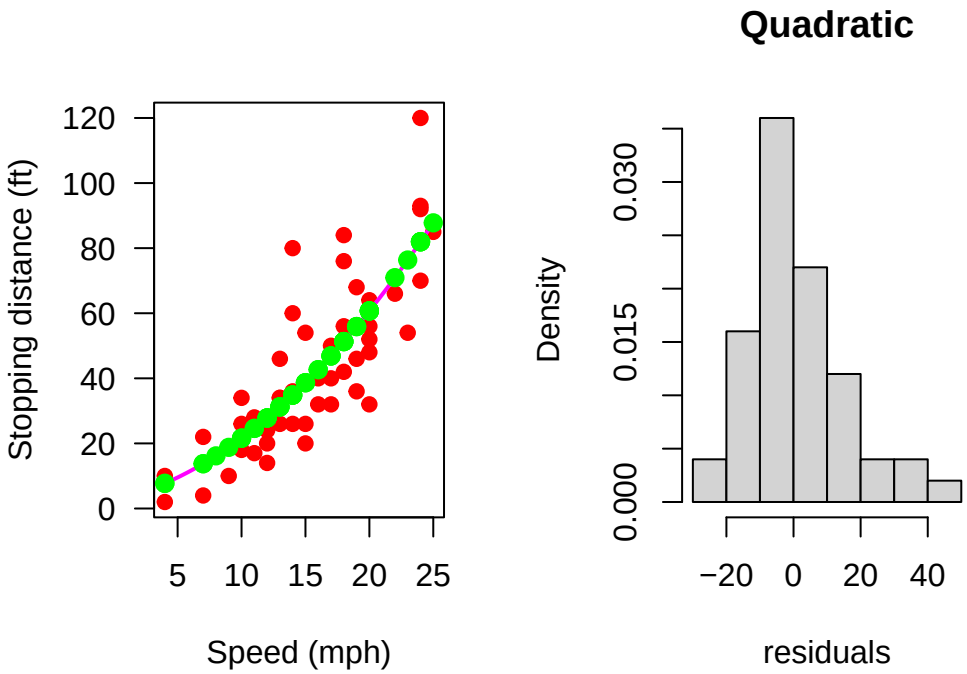


Figure 15: The least squares fitting by the quadratic regression model is shown magenta color and predictions are shown in green color. Corresponding histogram of the residuals of the fitted model is shown in the right panel.

Making predictions (Cubic)

In the following code we use the cubic regression equation to make the predictions.

```r
par(mfrow = c(1,2))
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")
X = matrix(data = c(rep(1,n), cars$speed, cars$speed^2, cars$speed^3),
```

```
           nrow = n, ncol = 4)
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
curve(b_hat[1] + b_hat[2]*x + b_hat[3]*x^2 + b_hat[4]*x^3,
      add = TRUE, col = "magenta", lwd = 2)
H = X%*%(solve(t(X)%*%X))%*%t(X)

dist_hat = H%*%Y
points(cars$speed, dist_hat, col = "green", pch = 19,
       cex = 1.2)
e_hat = cars$dist - dist_hat
e_hat = (diag(1, nrow = n) - H)%*%Y
# print(e_hat)

hist(e_hat, probability = TRUE, xlab = "residuals",
     main = "Cubic")
shapiro.test(e_hat)
```

```
    Shapiro-Wilk normality test

data:  e_hat
W = 0.92868, p-value = 0.004928
```

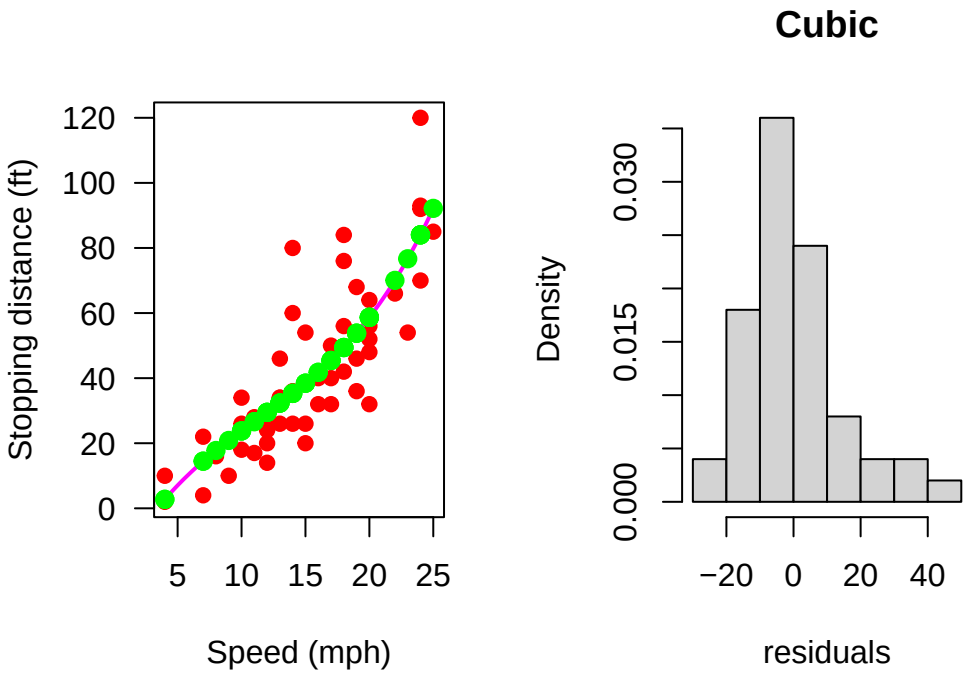
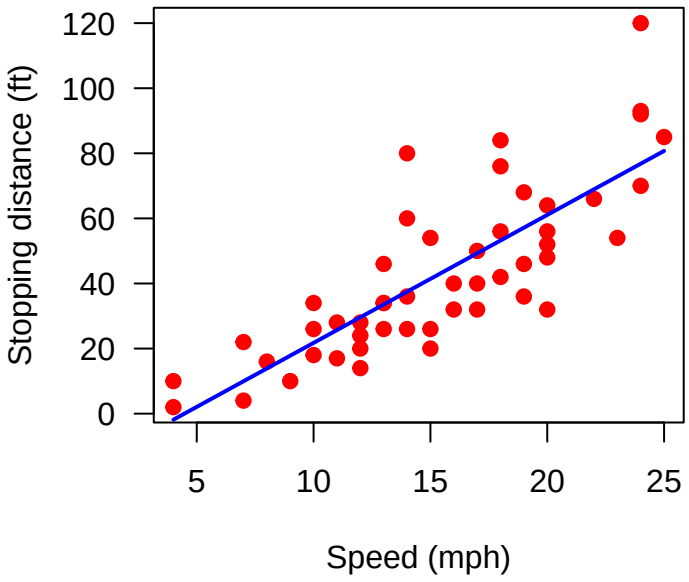


Figure 16: The least squares fitting using the cubic regression model is shown in magenta, and the corresponding predictions are displayed in green. The histogram of the residuals from the fitted model is presented in the right panel.

Sensitivity of the estimates

Coming back to the Simple Linear Regression and its dependence on individual data points. We remove the first observation and obtain the estimates of b_0 and b_1 and store it. Then remove the second row from the data set and compute the parameter estimates and store. The same has been done for all the rows and we obtain a total of 50 estimates of $b = (b_0, b_1)'$ and call them as b_{0-j} and b_{1-j} for $j \in \{1, 2, \ldots, 50\}$.

```
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")
X = matrix(data = c(rep(1,n), cars$speed), nrow = n, ncol = 2)
b_hat = (solve(t(X)%*%X))%*%t(X)%*%Y
curve(b_hat[1] + b_hat[2]*x, add = TRUE,
      col = "blue", lwd = 2)
```




```r
H = X%*%(solve(t(X)%*%X))%*%t(X)

b_hat_j = matrix(data = NA, nrow = n, ncol = 2)
for (j in 1:n) {
  Y_j = Y[-j]
  X_j = X[-j, ]
  H_j = X_j%*%(solve(t(X_j)%*%X_j))%*%t(X_j)
  b_hat_j[j, ] = (solve(t(X_j)%*%X_j))%*%t(X_j)%*%Y_j
}
```

In the following, we plot the deviations of the estimates of b_0 and b_1 from the original estimates obtained using the whole data set.

```r
par(mfrow = c(1,2))

plot(1:n, b_hat_j[,1], col = "red", pch = 19,
     xlab = "data index", ylab = expression(hat(b[0-j])))
abline(h = b_hat[1], lwd = 3, col = "magenta", lty = 2)
points(49, b_hat_j[49,1], cex = 2, col = "blue", lwd = 2)

plot(1:n, b_hat_j[,2], col = "red", pch = 19,
     xlab = "data index", ylab = expression(hat(b[1-j])))
abline(h = b_hat[2], lwd = 3, col = "magenta", lty = 2)

points(49, b_hat_j[49,2], cex = 2, col = "blue", lwd = 2)
```

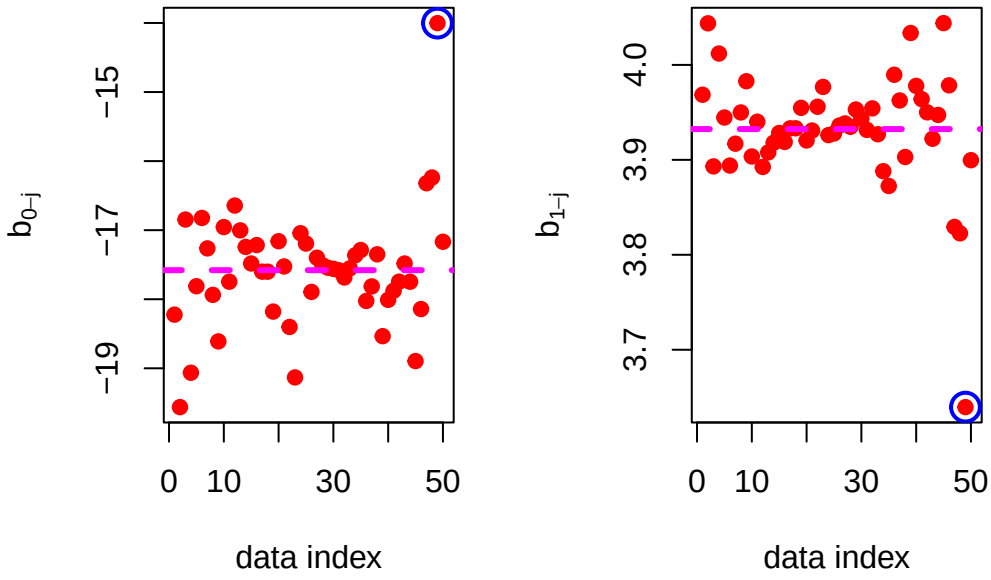


Figure 17: The estimated coefficients are plotted against each row (after removing it from the original data). The horizontal dotted magenta line represents the estimates when the complete data set is used.

From the above figure it is evident that 49th observation has a very high impact on the fitted regression line. Let us remove it and refit the equations and perform the residual analysis.

Removing 49th observation (the outlier visually)

In the following, we remove the 49th observation manually and check whether

```
par(mfrow = c(1,2))
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, pch = 19, col = "red")
points(cars$speed[49], cars$dist[49], pch = 4, lwd = 3)
X = matrix(data = c(rep(1,n), cars$speed), nrow = n, ncol = 2)
X_49 = X[-49, ]
Y_49 = Y[-49]
b_hat_49 = (solve(t(X_49)%*%X_49))%*%t(X_49)%*%Y_49
curve(b_hat_49[1] + b_hat_49[2]*x, add = TRUE,
      col = "blue", lwd = 2)
H_49 = X_49%*%(solve(t(X_49)%*%X_49))%*%t(X_49)
dist_hat_49 = H_49%*%Y_49
points(cars$speed[-49], dist_hat_49, col = "green", pch = 19,
       cex = 1.2)
```

```r
e_hat_49 = cars$dist[-49] - dist_hat_49
e_hat_49 = (diag(1, nrow = n-1) - H_49)%*%Y_49
# print(e_hat_49)

hist(e_hat_49, probability = TRUE, xlab = "residuals",
     main = "Linear")
shapiro.test(e_hat_49)
```

```
    Shapiro-Wilk normality test

data:  e_hat_49
W = 0.95814, p-value = 0.07941
```

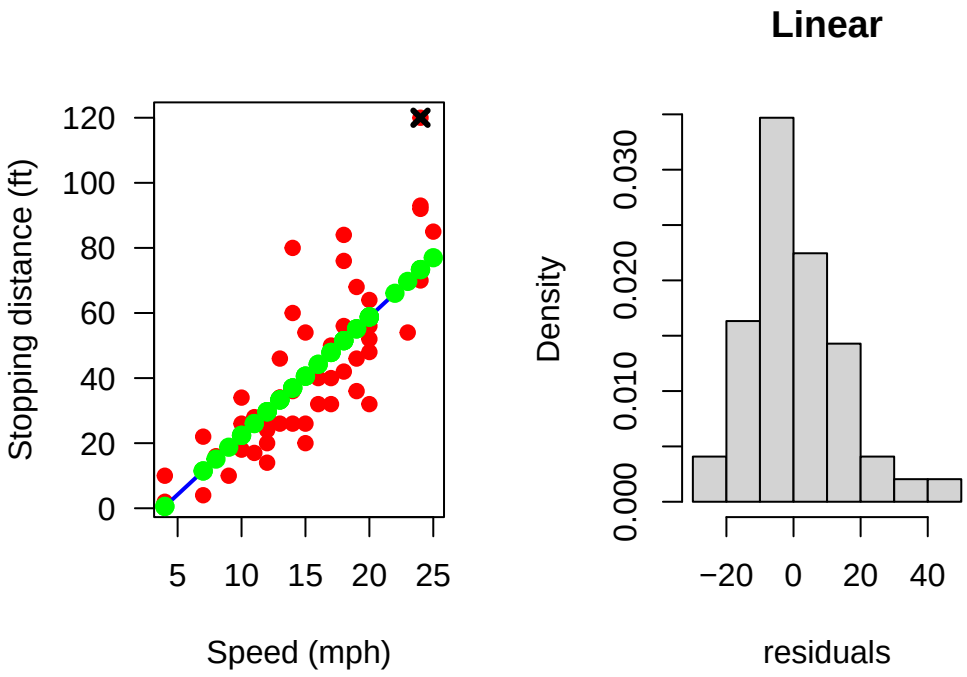


Figure 18: The model predictions by a linear regression model fitted after removing the 49th row from the data set is shown in green color. The histogram of the residuals is show in the right panel.

Classroom Quiz: Outlier identification

- Remove the 49th observation from the data set and fit both quadratic and cubic regression equation and check whether the residuals are normally distributed.

- Try to search for some inbuilt functions in R which may be used to identify the outliers.

Homework Assignment

Homework: Explore individual variables with appropriate plots Boxplot, histogram, vioplots and also check outliers from the boxplots if any. The following codes will be useful for the next step. Also check whether the variables are normally distributed (use `shapiro.test`). Make bar plots for the categorical variables. Also make a pie chart wherever appropriate.

```
library(ISLR)
ISLR::Auto
head(Auto)
names(Auto)
View(Auto)
help("Auto")
dim(Auto)

boxplot(Auto$displacement, main = "Engine displacement (cu. inches)")
help("boxplot")

hist(Auto$displacement, probability = TRUE,
     main = "Engine displacement (cu. inches)")

library(vioplot)
vioplot(Auto$displacement, main = "Engine displacement (cu. inches)")

# Whether the variable is normally distributed
shapiro.test(Auto$displacement)

mean(Auto$displacement)
sd(Auto$displacement)
var(Auto$displacement)
median(Auto$displacement)
library(moments)
skewness(Auto$displacement)

mean((Auto$displacement - mean(Auto$displacement))^3)/(sd(Auto$displacement))^3
# Check in the internet why the difference happend!
class(Auto$displacement)
```

```r

# Origin (Actually a categorical variable but coded as numeric)
class(Auto$origin)
unique(Auto$origin)
table(Auto$origin)
barplot(table(Auto$origin), main = "Origin", col = "red")


boxplot(Auto$displacement ~ Auto$origin)
# Make this plot more beautiful

pairs(Auto, col = "darkgrey")
help(pairs)   # explor pairs() function

summary(Auto)

# Regression model
plot(mpg ~ displacement, data = Auto, pch = 19,
     col = "darkgrey")
plot(Auto$displacement, Auto$mpg)

# Fit a linear regression and quadratic regression equation and perform
# analysis of the residuals and also find if there is some outliers

# Fit a multiple linear regression with mpg as response and
# displacements and weight as the predictors
```

Understanding correlation

In the following we consider four different cases of simulations of observations from two random variables X and Y. In each of the case, X and Y are related or not related in some way. In each case, you can understand on your own how they have been simulated. The following demonstration will help us to understand the formula of the correlation better which is given by

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}.$$

Whether the correlation is positive or negative, it completely depends of the value of the denominator which is a sum of the product of two quantities. Let us investigate it closely.

- For each $i \in \{1, 2, \ldots, n\}$, $x_i > \overline{x}$ or $x_i < \overline{x}$ and therefore $x_i - \overline{x}$ will be positive or negative, respectively.
- For each $i \in \{1, 2, \ldots, n\}$, $y_i > \overline{y}$ or $y_i < \overline{y}$ and therefore $y_i - \overline{y}$ will be positive or negative, respectively.
- Therefore, the sum of the numerator can be expressed into four parts.

$$\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \sum_{\{i:x_i<\bar{x},y_i<\bar{y}\}} + \sum_{\{i:x_i>\bar{x},y_i<\bar{y}\}} + \sum_{\{i:x_i<\bar{x},y_i>\bar{y}\}} + \sum_{\{i:x_i>\bar{x},y_i>\bar{y}\}}$$

Basically, if we shift the origin to the point $(\bar{x}, \bar{y})$, then the points will be distributed in four regions as demonstrated in Figure 19. We consider four cases of data generation process, viz. case - I: X and Y are independent; case - II: X and Y are negatively correlated; case - III: X and Y are positively correlated; case - IV: X and Y are negatively correlated. The corresponding R codes are provided in Listing 0.1.

Listing 0.1 Four different simulation study to generate the (x_i, y_i) pairs of values using R.

```
# Case - I
x = rnorm(n = 1000, mean = 0, sd = 1)
y = rnorm(n = 1000, mean = 0, sd = 1)

# Case - II
x = rnorm(n = 1000, mean = 0, sd = 1)
y = -x + rnorm(n = 1000, mean = 0, sd = 1)

# Case - III
x = rnorm(n = 1000, mean = 0, sd = 1)
y = x + rnorm(n = 1000, mean = 0, sd = 1)

# Case - IV
x = rnorm(n = 1000, mean = 0, sd = 1)
y = x^2 + rnorm(n = 1000, mean = 0, sd = 1)
```

X and Y are independent

```
par(mfrow = c(1,1))
x = rnorm(n = 1000, mean = 0, sd = 1)
y = rnorm(n = 1000, mean = 0, sd = 1)
```

```r
plot(x,y, pch = 19, col = "darkgrey", xlim = c(-5,5),
     ylim = c(-5,5))
abline(v = mean(x), col = "magenta", lwd = 3)
abline(h = mean(y), col = "magenta", lwd = 3)
text(3.5, 3.5, "I", cex = 1.5)
text(-3.5, 3.5, "II", cex = 1.5)
text(-3.5, -3.5, "III", cex = 1.5)
text(3.5, -3.5, "IV", cex = 1.5)
points(mean(x), mean(y), pch = 19, col = "blue",
       cex = 1.3)
text(4.5, 4.5, "(+,+)", cex = 1.2, col = "blue")
text(-4.5, 4.5, "(-,+)", cex = 1.2, col = "blue")
text(4.5, -4.5, "(+,-)", cex = 1.2, col = "blue")
text(-4.5, -4.5, "(-,-)", cex = 1.2, col = "blue")
cor(x,y)
```

```
[1] -0.008789047
```

```r
title(bquote(rho == .(cor(x,y))))
```

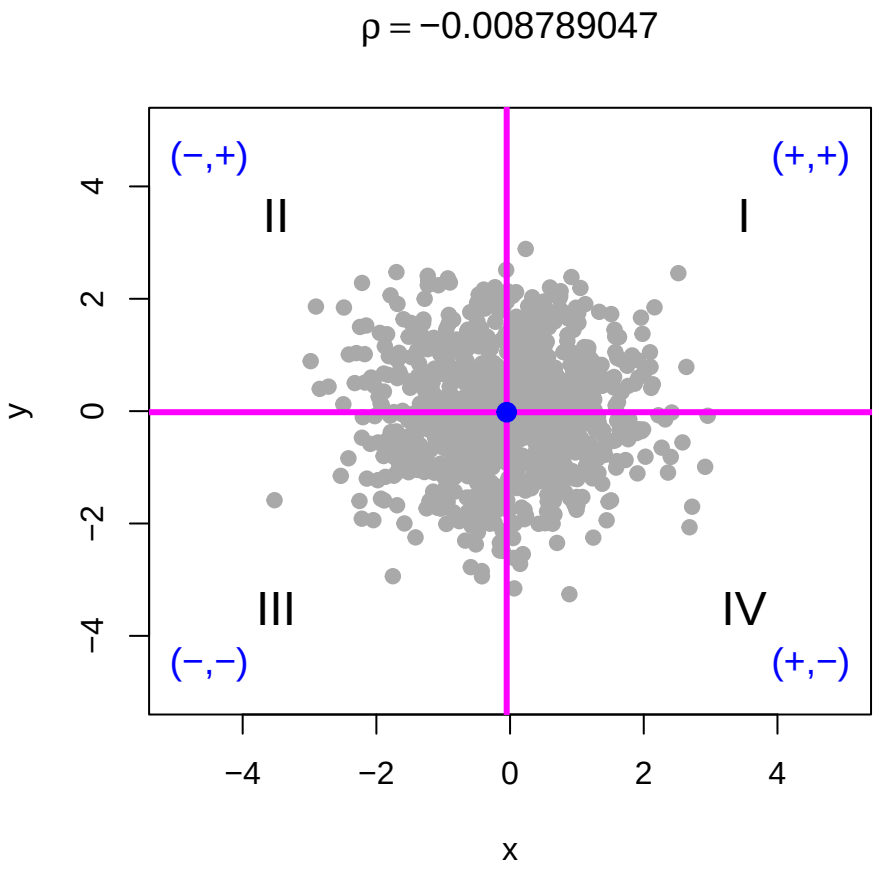


Figure 19: The plane is divided into four regions (quadrants). Depending on the location of the points, the sign of the product $(x_i - \bar{x})(y_i - \bar{y})$ varies. From the figure, it is evident that the data points are distributed almost equally across all four quadrants. As a result, the number of positive and negative products is approximately balanced, and neither dominates. Consequently, the numerator in the correlation formula, representing the sum of these products,is close to zero.

X and Y are negatvely correlated

```
y = -x + rnorm(n = length(x))
plot(x,y, pch = 19, col = "darkgrey", xlim = c(-5,5),
     ylim = c(-5,5))
abline(v = mean(x), col = "magenta", lwd = 3)
abline(h = mean(y), col = "magenta", lwd = 3)
text(3.5, 3.5, "I", cex = 1.5)
text(-3.5, 3.5, "II", cex = 1.5)
```

```
text(-3.5, -3.5, "III", cex = 1.5)
text(3.5, -3.5, "IV", cex = 1.5)
points(mean(x), mean(y), pch = 19, col = "blue",
       cex = 1.3)
text(4.5, 4.5, "(+,+)", cex = 1.2, col = "blue")
text(-4.5, 4.5, "(-,+)", cex = 1.2, col = "blue")
text(4.5, -4.5, "(+,-)", cex = 1.2, col = "blue")
text(-4.5, -4.5, "(-,-)", cex = 1.2, col = "blue")
cor(x,y)
```

```
[1] -0.7172201
```

```
title(bquote(rho == .(cor(x,y))))
```

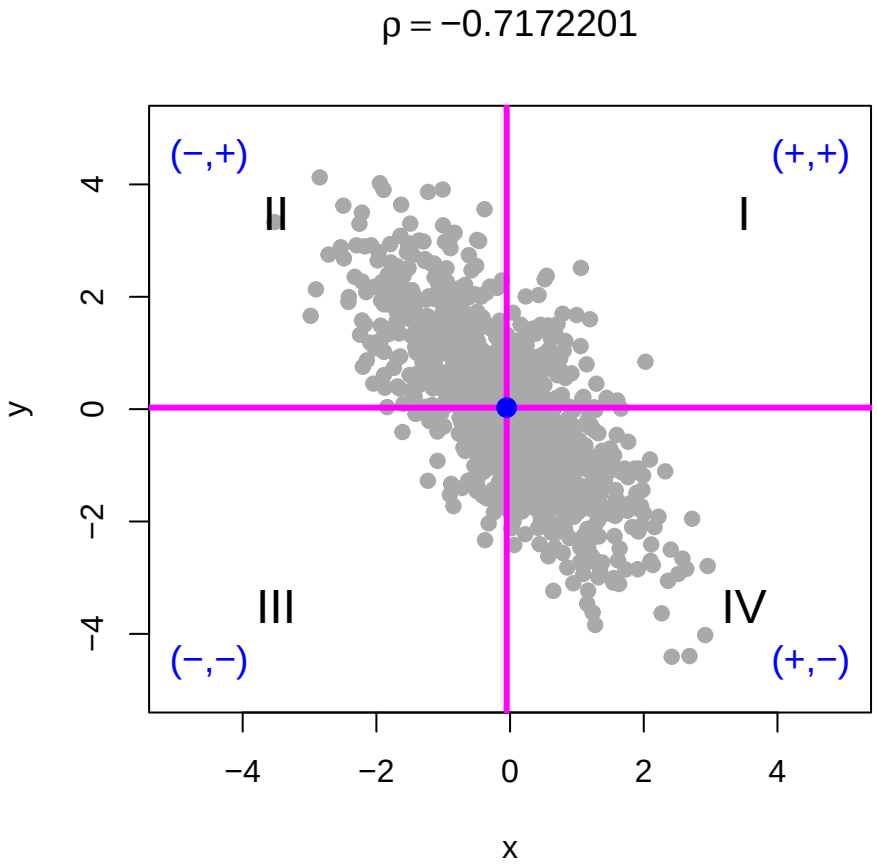


Figure 20: The plane is divided into four regions (quadrants). Depending on the location of the points, the sign of the product $(x_i - \bar{x})(y_i - \bar{y})$ varies. From the figure, it is evident that more data points are distributed in the second and the fourth quadrants. As a result, the number of negative products dominates the positive products. Consequently, the numerator in the correlation formula, representing the sum of these products, is negative.

X and Y are positively correlated

```
y = x + rnorm(n = length(x))
plot(x,y, pch = 19, col = "darkgrey", xlim = c(-5,5),
     ylim = c(-5,5))
abline(v = mean(x), col = "magenta", lwd = 3)
abline(h = mean(y), col = "magenta", lwd = 3)
text(3.5, 3.5, "I", cex = 1.5)
text(-3.5, 3.5, "II", cex = 1.5)
```

```
text(-3.5, -3.5, "III", cex = 1.5)
text(3.5, -3.5, "IV", cex = 1.5)
points(mean(x), mean(y), pch = 19, col = "blue",
       cex = 1.3)
text(4.5, 4.5, "(+,+)", cex = 1.2, col = "blue")
text(-4.5, 4.5, "(-,+)", cex = 1.2, col = "blue")
text(4.5, -4.5, "(+,-)", cex = 1.2, col = "blue")
text(-4.5, -4.5, "(-,-)", cex = 1.2, col = "blue")
cor(x,y)
```

```
[1] 0.7054136
```

```
title(bquote(rho == .(cor(x,y))))
```

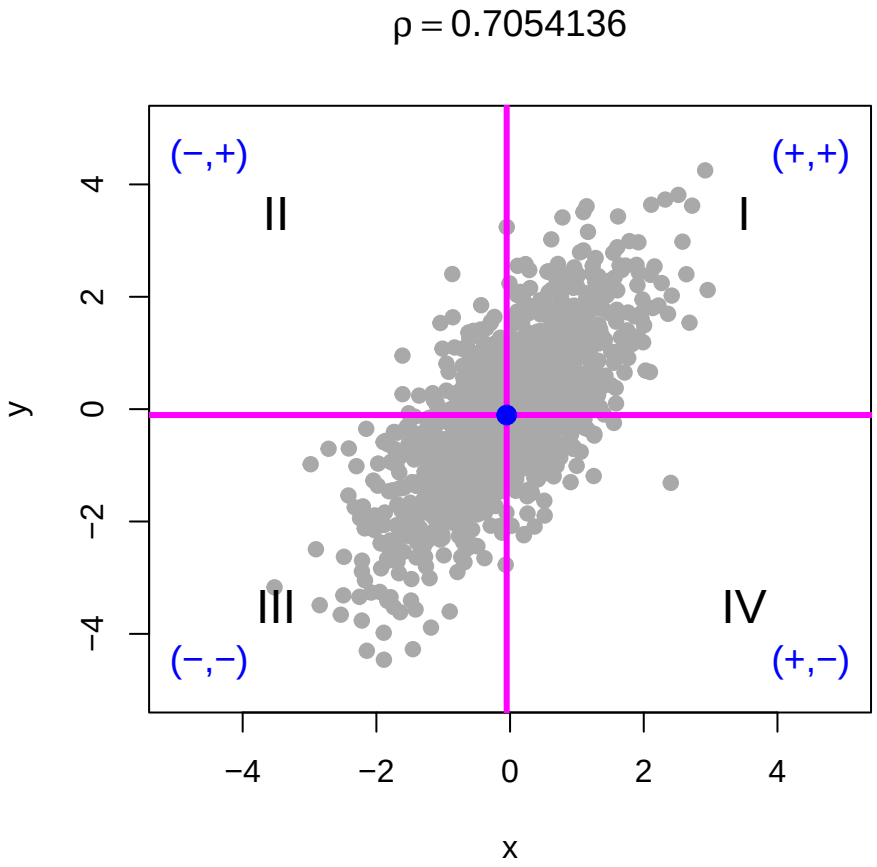


Figure 21: The plane is divided into four regions (quadrants). Depending on the location of the points, the sign of the product $(x_i - \bar{x})(y_i - \bar{y})$ varies. From the figure, it is evident that more data points are distributed in the first and the third quadrants. As a result, the number of positive products dominates the negative products. Consequently, the numerator in the correlation formula, representing the sum of these products, is positive.

X and Y are nonlinearly related

It should be noted that correlation is a measure of the strength of a linear relationship between two variables, which is also evident from the above discussion. Correlation cannot be used to assess independence, as there can be situations where the correlation is zero due to the absence of a linear relationship, even though the variables are nonlinearly dependent. From the graphical exploration using the division of the plane into four regions, we observe that the positive and negative contributions to the product $(x_i - \bar{x})(y_i - \bar{y})$ may cancel each other out in the presence of nonlinear dependence, resulting in a correlation close to zero. The corresponding R codes are given in Listing 0.3 and visual demonstration is given in Figure 22.

Listing 0.2 R Code: The simulation of the pairs of values are generated using the nonlinear relationship.

```
y = x^2 + rnorm(n = length(x))
plot(x,y, pch = 19, col = "grey",
     xlim = c(min(x), max(x)), ylim = c(min(y), max(y)))
abline(v = mean(x), col = "magenta", lwd = 2)
abline(h = mean(y), col = "magenta", lwd = 2)
points(mean(x), mean(y), col = "blue", cex = 1.4,
       pch = 19)
cor(x,y)
```

```
[1] -0.1125325
```

Listing 0.3 R Code: The simulation of the pairs of values are generated using the nonlinear relationship.

```
title(bquote(rho == .(cor(x,y))))
```

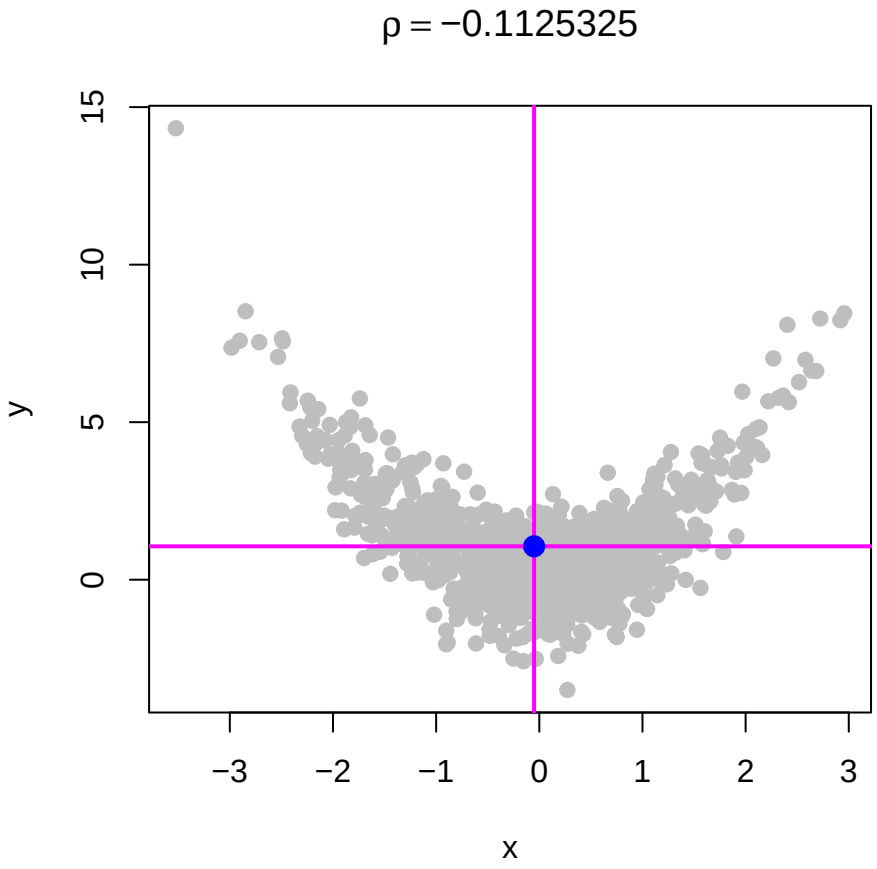


Figure 22: The nonlinear relationship between the random variables. The correlation is close to zero, however, they are non inpdependent as there is a nonlinear relationship between that $Y = X^2$.

Correlation and Linear Dependence

Therefore, we conclude that correlation measures the linear relationship between two variables. If two variables are independent, their correlation is zero. However, a zero correlation does not necessarily imply independence, as a nonlinear relationship between the variables may still exist.

Regularization and Real Data Analysis

We start our discussion of today's lecture with the usual multiple linear regression using the matrix set up which is given by

$$Y_{n\times 1} = X_{n\times(p+1)}\beta_{(p+1)\times 1} + \epsilon_{n\times 1},$$

and the symbols have the standard interpretations.

In addition, we assume that $\epsilon_{n\times 1} \sim \mathcal{N}_n\left(0_{n\times 1}, \sigma^2 I_{n\times n}\right)$. We have also checked in the class that $\mathrm{E}(\widehat{\beta}) = \beta$ that means the estimator $\widehat{\beta}$ an unbiased estimator of β. To recall that

$$\widehat{\beta} = \mathrm{argmax}_{\beta}(Y - X\beta)'(Y - X\beta).$$

The covariance matrix of $\widehat{\beta}$ is given by

$$Cov(\beta) = \sigma^2(X'X)^{-1}.$$

The diagonal entries of the above matrix gives us the variance associated with the estimator for individual component β_j of β for $j \in \{0, 1, 2, \ldots, p\}$. It is important to note that if the matrix $(X'X)$ is nearly singular, or ill conditioned, then the determinant $|X'X| \approx 0$ and the variance $\widehat{Var(\beta_j)}$ for $j \in \{0, 1, \ldots, p\}$ will be large and the estimates will unreliable. By unreliable, I mean the standard error of the estimators will be extremely large or may be infinity as well.

A natural question arises, in what scenarios such a possibility arise. From the theory of linear algebra, we know that if one column can be written as a constant multiple of another column, the determinant of the matrix is zero. It is not only between two columns, if a column can be written as a linear combination of two or more other columns (with at least one non-zero coefficient), then also the determinant is zero. In other words, the column space of the matrix does not have the dimension equal to the number of columns of X. In other words, the columns are linearly dependent.

Given the above discussion, if we want to avoid such situation where $|X'X| \approx 0$, we need to identify which columns of X are responsible for this. While discussing the concept of `correlation`, we have emphasized the fact that the correlation is a measure of `linear` relationship between random variables. Therefore, the degree of linear dependence between two columns of X (also called `feature` by machine learning experts) can be measured by the correlation between two columns of X. However, when one column can be written as a linear

combination of other columns, that correlation may not be able to capture. Such dependence can be checked by fitting linear regression of the following form

$$X_j = \beta_0^{(-j)} + \beta_1^{(-j)} X_1 + \cdots + \beta_{j-1}^{(-j)} X_{j-1} + \beta_{j+1}^{(-j)} X_j + \beta_p X_p + \epsilon$$

for $j \in \{1, 2, \ldots, p\}$. A large `r.squared` value of the above regression model R^2_{-j} indicates that X_j can be approximately represented as a linear combination of other columns.

Data scientists computes the quantity called, variance inflation factor (VIF) by

$$\text{VIF}_j = \frac{1}{1 - R^2_{-j}}, j = 1, 2, \ldots, p.$$

If the variance inflation factor is more than 5 or 10 (depending on the problem and also domain knowledge), an analyst may choose to drop those variables.

Let us consider the `Auto` dataset from the `ISLR2` package in R.

```
library(ISLR2)
pairs(Auto, col = 'red')
```

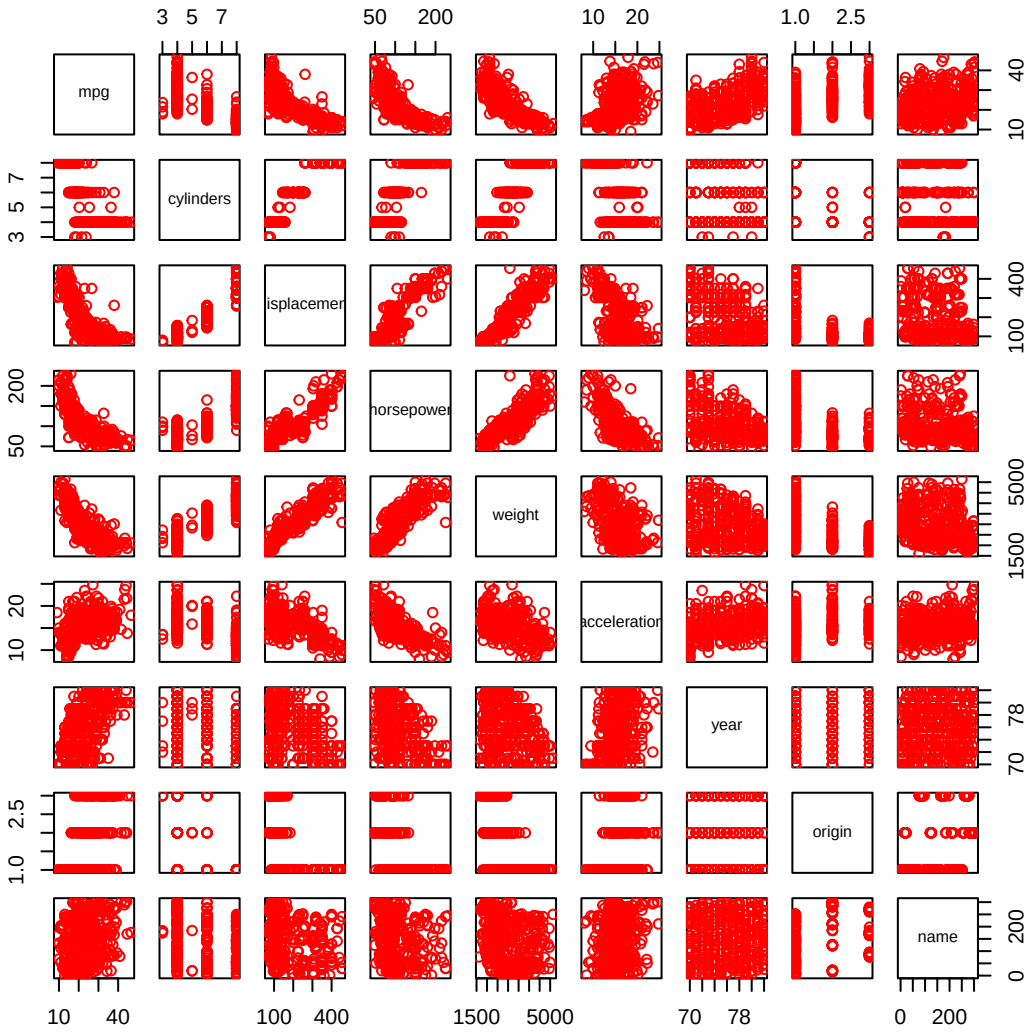


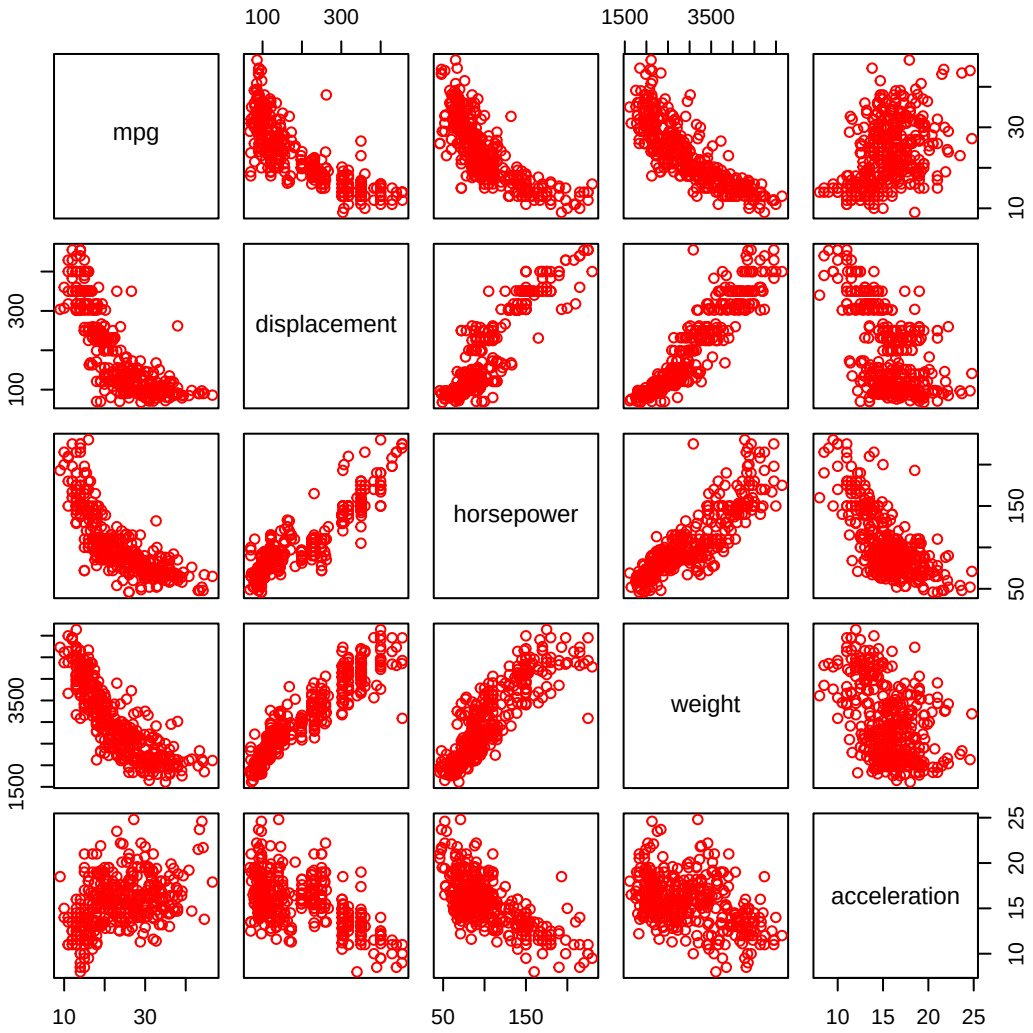
Figure 1: Pairs plot for the Auto data set with five selected columns.

```
data = Auto[,c("mpg", "displacement",
               "horsepower", "weight",
               "acceleration")]
head(data)
```

```
  mpg displacement horsepower weight acceleration
1  18          307        130   3504         12.0
2  15          350        165   3693         11.5
3  18          318        150   3436         11.0
4  16          304        150   3433         12.0
5  17          302        140   3449         10.5
```

```
6   15           429         198    4341          10.0
```

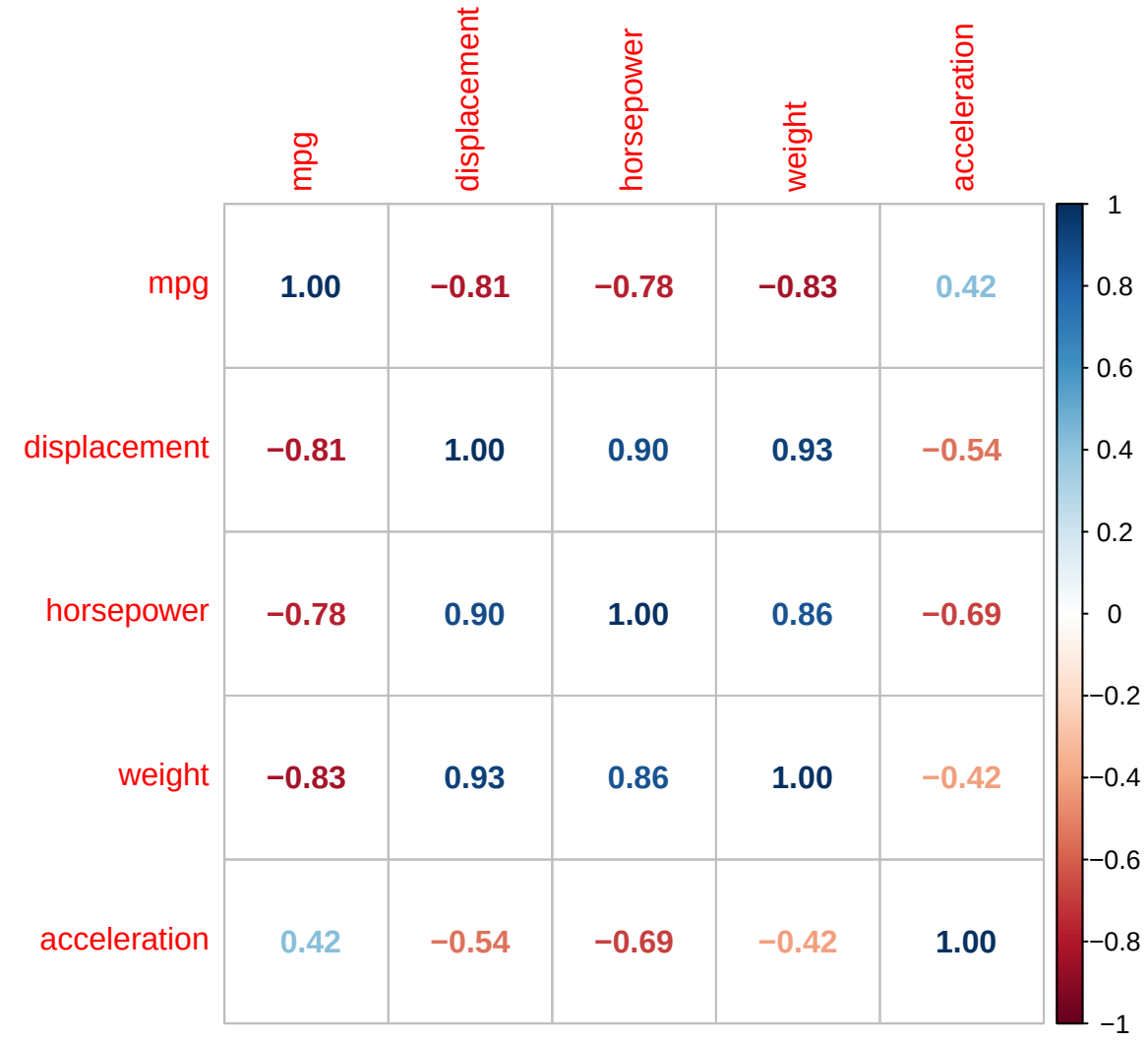
```
pairs(data, col = "red")
```




```
cor(data)
```

```
                mpg displacement horsepower     weight acceleration
mpg           1.0000000   -0.8051269 -0.7784268 -0.8322442    0.4233285
displacement -0.8051269    1.0000000  0.8972570  0.9329944   -0.5438005
horsepower   -0.7784268    0.8972570  1.0000000  0.8645377   -0.6891955
weight       -0.8322442    0.9329944  0.8645377  1.0000000   -0.4168392
acceleration  0.4233285   -0.5438005 -0.6891955 -0.4168392    1.0000000
```

```r
library(corrplot)
corrplot(corr = cor(data), method = "number")
```




```r
fit = lm(mpg ~ ., data = data)
summary(fit)
```

```
Call:
lm(formula = mpg ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-11.378  -2.793  -0.333   2.193  16.256

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  45.2511397  2.4560447  18.424  < 2e-16 ***
displacement -0.0060009  0.0067093  -0.894  0.37166
horsepower   -0.0436077  0.0165735  -2.631  0.00885 **
weight       -0.0052805  0.0008109  -6.512  2.3e-10 ***
acceleration -0.0231480  0.1256012  -0.184  0.85388
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.247 on 387 degrees of freedom
Multiple R-squared:  0.707, Adjusted R-squared:  0.704
F-statistic: 233.4 on 4 and 387 DF,  p-value: < 2.2e-16
```

```r
R2 = summary(fit)$r.squared
```

```r
fit_1 = lm(displacement ~ horsepower + weight + acceleration,
           data = data)
R2_displacement = summary(fit_1)$r.squared

fit_2 = lm(horsepower ~ displacement + weight + acceleration,
           data = data)
R2_horsepower = summary(fit_2)$r.squared

fit_3 = lm(weight ~ displacement + horsepower + acceleration,
           data = data)
R2_weight = summary(fit_3)$r.squared

fit_4 = lm(acceleration ~ displacement + horsepower + weight,
           data = data)
R2_acceleration = summary(fit_4)$r.squared
```

```r
print(R2)
```

```
[1] 0.7069812
```

```r
print(R2_displacement)
```

```
[1] 0.9064277
```

```r
print(R2_horsepower)
```

```
[1] 0.8866601
```

```r
print(R2_weight)
```

```
[1] 0.9027659
```

```r
print(R2_acceleration)
```

```
[1] 0.6158656
```

```r
vif_displacement = 1/(1-R2_displacement)
vif_horsepower = 1/(1-R2_horsepower)
vif_weight = 1/(1- R2_weight)
vif_acceleration = 1/(1-R2_acceleration)
```

```r
print(c(vif_displacement,
        vif_horsepower,
        vif_weight,
        vif_acceleration))
```

```
[1] 10.686922  8.823022 10.284456  2.603255
```

Verification with existing packages in R

```r
library(car)
```

```
Loading required package: carData
```

```r
vif(fit)
```

```
displacement   horsepower       weight acceleration
   10.686922     8.823022    10.284456     2.603255
```

Optimal choice of lambda

```r
lambda = 0.01
X = cbind(rep(1,nrow(data)), as.matrix(data[,2:5]))
head(X)
```

```
    displacement horsepower weight acceleration
1 1          307        130   3504         12.0
2 1          350        165   3693         11.5
3 1          318        150   3436         11.0
4 1          304        150   3433         12.0
5 1          302        140   3449         10.5
6 1          429        198   4341         10.0
```

```r
Y = data$mpg

Y_hat = numeric(length = nrow(data))
for(i in 1:nrow(data)){
  new_X = X[-i, ]
  beta_lambda = solve(t(new_X)%*%new_X + lambda*diag(ncol(X)))%*%t(new_X)%*%Y[-i]
  Y_hat[i] = X[i, ]%*%beta_lambda
}

error = Y - Y_hat
pred_error = mean(error^2)
```

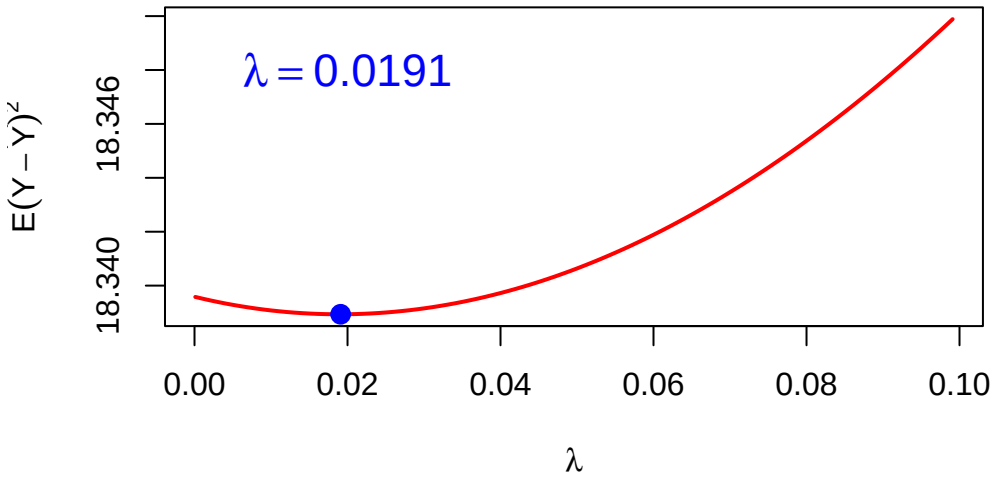
Repeat above for differnt lambda values

```r
lambda_vals = seq(0.0001, 0.1, by = 0.001)
pred_error = numeric(length = length(lambda_vals))
for(j in 1:length(lambda_vals)){
  lambda = lambda_vals[j]
  Y_hat = numeric(length = nrow(data))
  for(i in 1:nrow(data)){
    new_X = X[-i, ]
    beta_lambda = solve(t(new_X)%*%new_X + lambda*diag(ncol(X)))%*%t(new_X)%*%Y[-i]
    Y_hat[i] = X[i, ]%*%beta_lambda
  }

  error = Y - Y_hat
  pred_error[j] = mean(error^2)

}
```

```r
plot(lambda_vals, pred_error, col = "red", pch = 19,
     cex = 1.2, xlab = expression(lambda),
     ylab = expression(E(Y-widehat(Y))^2), type = "l",
     lwd = 2)
which.min(pred_error)
```

```
[1] 20
```

```r
lambda_star = lambda_vals[which.min(pred_error)]
points(lambda_star,pred_error[which.min(pred_error)],
       pch = 19, col = "blue",
       cex = 1.3)
text(0.02, 18.348, bquote(lambda == .(lambda_star)),
       cex = 1.4, col = "blue", pch = 19, bty = "n")
```



In the above analysis, it is clear that the computation associated with the approximation of the prediction error is heavy. The number of rows in the data set is 392, therefore, 392 many times, model training has been carried out. The advantage of the above method is basically is that every row of the data set has been used for both model training as well as model testing purpose. The number of model fitting exercises may be reduced by dividing the data into k many segments of approximately equal size.

```r
nrow(data)
```

```
[1] 392
```

```r
Y = data$mpg
X = cbind(rep(1,nrow(data)), as.matrix(data[,-1]))
Y
```

```
  [1] 18.0 15.0 18.0 16.0 17.0 15.0 14.0 14.0 14.0 15.0 15.0 14.0 15.0 14.0 24.0
 [16] 22.0 18.0 21.0 27.0 26.0 25.0 24.0 25.0 26.0 21.0 10.0 10.0 11.0  9.0 27.0
 [31] 28.0 25.0 19.0 16.0 17.0 19.0 18.0 14.0 14.0 14.0 14.0 12.0 13.0 13.0 18.0
 [46] 22.0 19.0 18.0 23.0 28.0 30.0 30.0 31.0 35.0 27.0 26.0 24.0 25.0 23.0 20.0
 [61] 21.0 13.0 14.0 15.0 14.0 17.0 11.0 13.0 12.0 13.0 19.0 15.0 13.0 13.0 14.0
 [76] 18.0 22.0 21.0 26.0 22.0 28.0 23.0 28.0 27.0 13.0 14.0 13.0 14.0 15.0 12.0
 [91] 13.0 13.0 14.0 13.0 12.0 13.0 18.0 16.0 18.0 18.0 23.0 26.0 11.0 12.0 13.0
[106] 12.0 18.0 20.0 21.0 22.0 18.0 19.0 21.0 26.0 15.0 16.0 29.0 24.0 20.0 19.0
[121] 15.0 24.0 20.0 11.0 20.0 19.0 15.0 31.0 26.0 32.0 25.0 16.0 16.0 18.0 16.0
[136] 13.0 14.0 14.0 14.0 29.0 26.0 26.0 31.0 32.0 28.0 24.0 26.0 24.0 26.0 31.0
[151] 19.0 18.0 15.0 15.0 16.0 15.0 16.0 14.0 17.0 16.0 15.0 18.0 21.0 20.0 13.0
[166] 29.0 23.0 20.0 23.0 24.0 25.0 24.0 18.0 29.0 19.0 23.0 23.0 22.0 25.0 33.0
[181] 28.0 25.0 25.0 26.0 27.0 17.5 16.0 15.5 14.5 22.0 22.0 24.0 22.5 29.0 24.5
[196] 29.0 33.0 20.0 18.0 18.5 17.5 29.5 32.0 28.0 26.5 20.0 13.0 19.0 19.0 16.5
[211] 16.5 13.0 13.0 13.0 31.5 30.0 36.0 25.5 33.5 17.5 17.0 15.5 15.0 17.5 20.5
[226] 19.0 18.5 16.0 15.5 15.5 16.0 29.0 24.5 26.0 25.5 30.5 33.5 30.0 30.5 22.0
[241] 21.5 21.5 43.1 36.1 32.8 39.4 36.1 19.9 19.4 20.2 19.2 20.5 20.2 25.1 20.5
[256] 19.4 20.6 20.8 18.6 18.1 19.2 17.7 18.1 17.5 30.0 27.5 27.2 30.9 21.1 23.2
[271] 23.8 23.9 20.3 17.0 21.6 16.2 31.5 29.5 21.5 19.8 22.3 20.2 20.6 17.0 17.6
[286] 16.5 18.2 16.9 15.5 19.2 18.5 31.9 34.1 35.7 27.4 25.4 23.0 27.2 23.9 34.2
[301] 34.5 31.8 37.3 28.4 28.8 26.8 33.5 41.5 38.1 32.1 37.2 28.0 26.4 24.3 19.1
[316] 34.3 29.8 31.3 37.0 32.2 46.6 27.9 40.8 44.3 43.4 36.4 30.0 44.6 33.8 29.8
[331] 32.7 23.7 35.0 32.4 27.2 26.6 25.8 23.5 30.0 39.1 39.0 35.1 32.3 37.0 37.7
[346] 34.1 34.7 34.4 29.9 33.0 33.7 32.4 32.9 31.6 28.1 30.7 25.4 24.2 22.4 26.6
[361] 20.2 17.6 28.0 27.0 34.0 31.0 29.0 27.0 24.0 36.0 37.0 31.0 38.0 36.0 36.0
[376] 36.0 34.0 38.0 32.0 38.0 25.0 38.0 26.0 22.0 32.0 36.0 27.0 27.0 44.0 32.0
[391] 28.0 31.0
```

```r
head(X)
```

```
    displacement horsepower weight acceleration
1 1          307        130   3504         12.0
2 1          350        165   3693         11.5
3 1          318        150   3436         11.0
4 1          304        150   3433         12.0
5 1          302        140   3449         10.5
6 1          429        198   4341         10.0
```

```r
k = 10          # number of folds
floor(nrow(data)/k)
```

```
[1] 39
```

```r
fold = sample(rep(1:k, nrow(data)),
              size = nrow(data), replace = FALSE)
fold
```

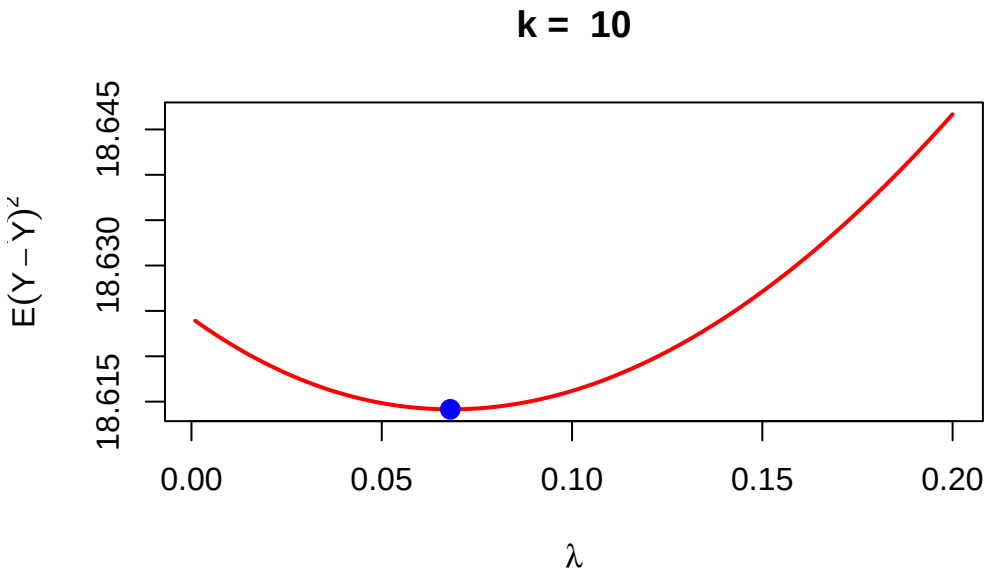
```
  [1]  7  6 10 10  5  5  5  7 10  6  4  7  2  9  5  2  5  3 10  5  8  3  8  5  2
 [26]  3 10  5  9  6  8  9  2  1  6  6 10  9  1  6  5  6  7  2  4  5  7  5  8  1
 [51]  3  6  3  8  8  3  1  5  5  7  2  4  6  3  9  2  5  1  4  7  7 10  6  5  4
 [76]  8  1  1  9  6  1 10  1  8  3  9  3  9  1  7 10  4  1 10  8  1  6 10  8  8
[101]  9  4 10  9  7  9  1  6  4  7  3  5  6 10  3  8  4  3  8  7  9  6  9  6  9
[126] 10  6 10  9  4 10 10  2  1  5  4  6  7  1  4 10 10  9  5  4  7  7  4 10  7
[151]  6  3  1  6  2  5  2  1  4  6  9  9  9  5  4 10  5  7  8  6 10  5  3  1  7
[176]  8  1  3  6  8  9  6  4 10  2 10  3  8  1  9  1  3  8  4  9  2  3  4  3  3
[201]  6 10  9  2  5  1  5  3  1  8 10  6  6  7  1  4  3  9  6  7  4  2 10  6  4
[226]  4  9 10  2  9 10  4  8 10  4 10  6  2  6  1  7  9  9 10  9  6  8  5  4  4
[251]  4  8  1  7  6  9 10  2  9  6  4  8  4  5  8  4 10 10  7  1  3  9  9  2 10
[276]  2  9  7 10  7  6  8  9  2  5  1  2  6  5  7  3  5  8  8  7  9  1  8  9  3
[301]  1  7  6  5  8  2  6  7  1  7 10  7  4  9  1  7  8  5  3  1  1  4  1  8 10
[326]  4  6 10  6  5  5  6  6  7  8  5 10  4  3 10  5  5  5  4  1  5  6  3  9  5
[351]  4  6  9  7  5  2  1  1  3  2  6  2  4  7  7  5 10  4  5  4  3  2  3  9  6
[376] 10  8  6  6  5  8  3  6  8  7  7  1  6  2  7  4  3
```

```r
table(fold)
```

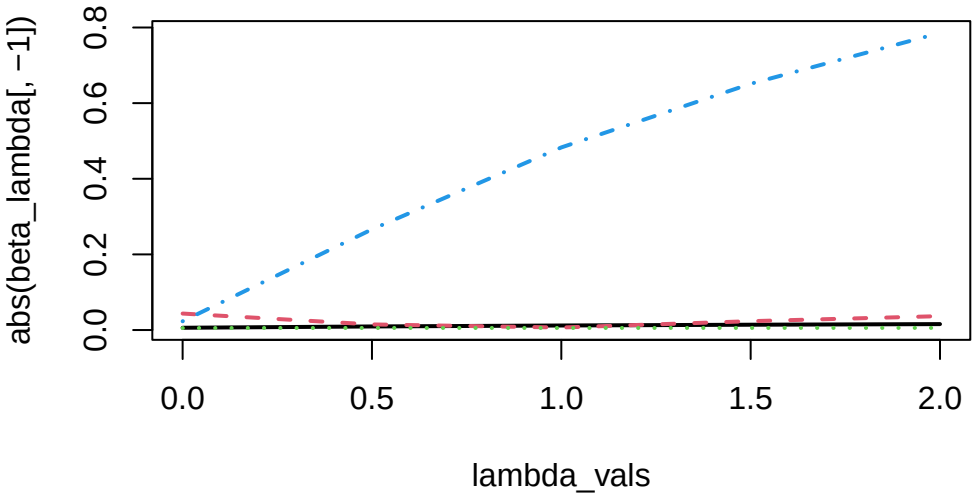
```
fold
 1  2  3  4  5  6  7  8  9 10
39 27 33 41 43 50 39 35 41 44
```

```r
# implementation of k-fold cross validation
lambda = 0.01
error = numeric(length = k)
for(i in 1:k){
  new_Y = Y[(fold != i)]
  new_X = X[(fold != i),]
  beta_lambda = solve(t(new_X)%*%new_X + lambda*diag(ncol(new_X)))%*%t(new_X)%*%new_Y
  Y_hat = X[fold==i,]%*%beta_lambda
```

```
  error[i] = mean((Y[fold == i] - Y_hat)^2)
}
k_fold_cv = mean(error)

# Do it for multiple lambda values
lambda_vals = seq(0.001,0.2, by = 0.001)
k_fold_cv = numeric(length = length(lambda_vals))
for(j in 1:length(lambda_vals)){
  lambda = lambda_vals[j]
  error = numeric(length = k)
  for(i in 1:k){
    new_Y = Y[(fold != i)]
    new_X = X[(fold != i),]
    beta_lambda = solve(t(new_X)%*%new_X + lambda*diag(ncol(new_X)))%*%t(new_X)%*%new_Y
    Y_hat = X[fold==i,]%*%beta_lambda
    error[i] = mean((Y[fold == i] - Y_hat)^2)
  }
  k_fold_cv[j] = mean(error)

}

plot(lambda_vals, k_fold_cv, col = "red",
     type = "l", lwd = 2,
     main = paste("k = ", k),
     xlab = expression(lambda),
     ylab = expression(E(Y-widehat(Y))^2))
points(lambda_vals[which.min(k_fold_cv)],
       k_fold_cv[which.min(k_fold_cv)],
       pch = 19, col = "blue", cex = 1.3)
```




```
# how the model coefficients shrinks as a function of $\lambda$

data = Auto[,c("mpg", "displacement",
               "horsepower", "weight",
               "acceleration")]
lambda_vals = seq(0, 2, by = 0.5)
Y = data$mpg
X = cbind(rep(1,nrow(data)), as.matrix(data[,-1]))
beta_lambda = matrix(data = NA, ncol = ncol(X),
                     nrow = length(lambda_vals))
for(i in 1:length(lambda_vals)){
  lambda = lambda_vals[i]
  beta_lambda[i,] = solve(t(X)%*%X + lambda*diag(ncol(X)))%*%t(X)%*%Y
}

matplot(lambda_vals, abs(beta_lambda[,-1]),
        type = "l", lwd = 2)
```


abs(beta_lambda[, −1])
0.0
0.2
0.4
0.6
0.8
0.0
0.5
1.0
1.5
2.0
lambda_vals

Nonlinear Regression Models

Introduction

In this tutorial, we shall understand how to fit nonlinear regression models using R for some give data set. We shall consider only a single input variable `age` and out variable `size`.

We shall first simulate the data set artificially with some fixed parameter choices and then apply nonlinear least squares on the data set. This will help us to understand the accuracy of the nonlinear least squares method using R.

Simulation of growth data

For demonstration we consider the following nonlinear function

$$\text{size} = a \times \text{age}^b + \text{e}$$

where a and b are fixed parameter and the error component e has normal distribution with mean 0 and variance σ^2. We use the function `set.seed()` to ensure that the simulation studies are reproducible. For simulation study, we fixed the parameter values as $a = 1$, $b = 0.4$ and $\sigma = 0.2$.

```
set.seed(123)
age = seq(1, 10, by = 0.1)           # age variable
print(age)
```

```
 [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4
[16]  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9
[31]  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
[46]  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9
[61]  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4
[76]  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9
[91] 10.0
```

```r
length(age)
```

```
[1] 91
```

```r
a = 1                                   # true value of a
b = 0.4                                 # true value of b

size = a*age^b
print(head(size))
```

```
[1] 1.000000 1.038860 1.075654 1.110650 1.144066 1.176079
```

```r
plot(age, size, type = "l", lwd = 3, col = "red",
     cex.lab = 1.5)
```

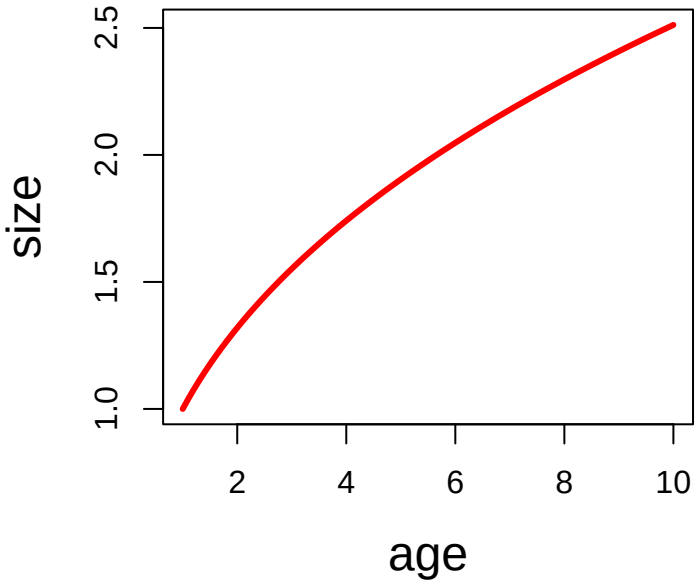


Figure 1: The mean growth profile. The parameters are fixed at $a = 1$ and $b = 0.4$. The plot should be considered as a conditional expectation of the `response variable` (size) at different values of age variable. E(size|age) as a function of age. If we fixe age = 4 (say), then expected size is $a \times 4^b$.

```r
set.seed(123)
size = a*age^b + rnorm(n = length(age), mean = 0, sd = 0.2)
# print(size)
```

```r
plot(age, size, col = "darkgrey",
       pch = 19)
lines(age, a*age^b, col = "red", lwd = 3)
data = data.frame(age, size)
head(data)
```

```
  age      size
1 1.0 0.8879049
2 1.1 0.9928246
3 1.2 1.3873954
4 1.3 1.1247520
5 1.4 1.1699239
6 1.5 1.5190920
```

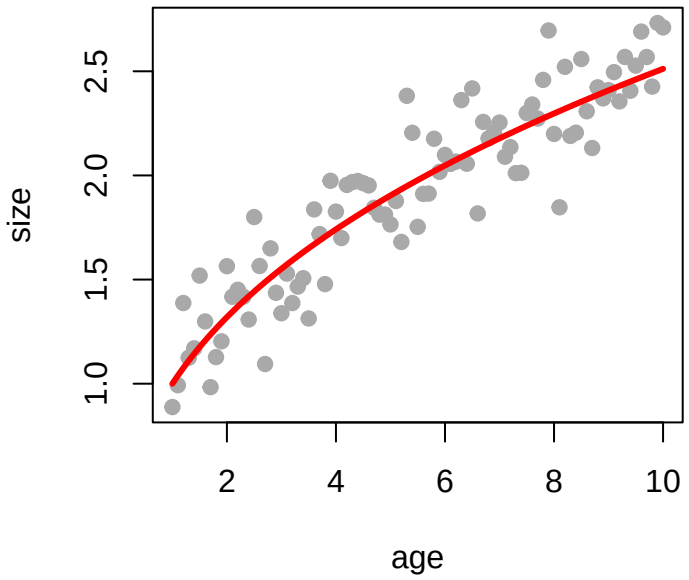


Figure 2: The simulated data set. The function set.seed() is set at 123 to ensure that the simulation study is reproducible. The parameters $a = 1$, $b = 0.4$ and $\sigma = 0.2$. The sample size is $n = 91$. The true conditional mean value of the response given age is added in red colour.

To estimate the parameters based using the simulated data, we need to minimize the error sum of squares which is given by

$$\text{ESS} = \sum_{i=1}^{n} \left(\text{size}_i - a \times \text{age}_i^b\right)^2$$

with respect to a and b. In the following we try to visualize the ESS as a function of a and b. n is the sample size or the number of rows in the data set.

```
ESS = function(a, b){                          # Error sum of squares
  sum((size - a*age^b)^2)
}

a_vals = seq(0, 2, by = 0.1)
b_vals = seq(0, 2, by = 0.1)
ESS_vals = matrix(data = NA,
                  nrow = length(a_vals),
                  ncol = length(b_vals))
for (i in 1:length(a_vals)) {
  for (j in 1:length(b_vals)) {
    ESS_vals[i,j] = ESS(a_vals[i], b_vals[j])
  }
}
cat("The matrix containing the error sum of squares values:\n")
```

```
The matrix containing the error sum of squares values:
```

```
head(ESS_vals)
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]      [,7]      [,8]
[1,] 356.1787 356.1787 356.1787 356.1787 356.1787 356.1787 356.17867 356.17867
[2,] 322.0559 315.9142 308.5870 299.8559 289.4726 277.1620 262.63008 245.58264
[3,] 289.7532 278.1504 264.4760 248.4342 229.7392 208.1552 183.56731 156.09720
[4,] 259.2705 242.8873 223.8456 201.9137 176.9783 149.1585 118.99036  87.72236
[5,] 230.6077 210.1248 186.6958 160.2942 131.1900 100.1718  68.89922  40.45811
[6,] 203.7650 179.8629 153.0267 123.5758  92.3742  61.1951  33.29390  14.30446
          [,9]     [,10]     [,11]     [,12]     [,13]      [,14]      [,15]
[1,] 356.17867 356.17867 356.17867 356.17867 356.17867  356.17867  356.17867
[2,] 225.75846 202.98900 177.29884 149.07140 119.31759   90.10601   65.24562
[3,] 126.29386  95.43884  66.03201  42.55778  32.68725   49.15634  112.69131
[4,]  57.78486  33.52818  22.37818  36.63783  96.28764  233.32965  498.51573
[5,]  20.23147  17.25702  46.33735 131.31154 310.11877  642.62595 1222.71888
[6,]  13.63369  46.62537 137.90952 326.57891 674.18063 1277.04523 2285.30076
          [,16]      [,17]      [,18]      [,19]      [,20]      [,21]
[1,]  356.17867  356.17867   356.1787   356.1787   356.1787   356.1787
[2,]   51.36231   59.58839   108.2047   226.7613   462.4931   890.2926
[3,]  256.55545  533.60446  1027.2224  1868.2602  3261.2594  5525.0426
```

```
[4,]  971.75809 1778.22689  3113.2319  5280.6753  8752.4778 14260.4286
[5,] 2196.97023 3793.45566  6366.2331 10464.0066 16936.1481 27096.4505
[6,] 3932.19187 6579.29079 10786.2261 17418.2542 27812.2704 44033.1084
```

```
dim(ESS_vals)
```

```
[1] 21 21
```

```
persp(a_vals, b_vals, ESS_vals,
      col = "yellow", xlab = "a",
      ylab = "b", zlab = "ESS",
      theta = 30)
```

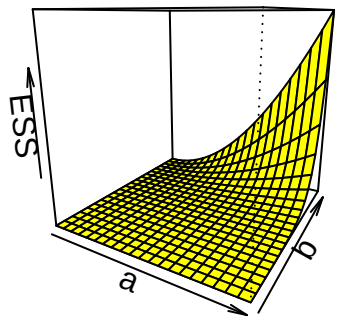


Figure 3: The variation of the error sum of squares as a function of a and b. We need to choose the values of the parameters at which the surface is minimum.

nonlinear least squares using R

The function `nls` is used to perform to obtain the nonlinear least squares estimate of a and b. To execute the nls function, we need to provide an initial starting point for the parameters.

```
fit = nls(size ~ a*age^b, data = data,
          start = list(a = 0.5, b = 1))
coefficients(fit)
```

```
        a         b
0.9994330 0.4040503
```

```
a_hat = coefficients(fit)[1]
b_hat = coefficients(fit)[2]

print(a_hat)
```

```
       a
0.999433
```

```
print(b_hat)
```

```
        b
0.4040503
```

Let us now plot the fitted curve to the data

```
class(fit)
```

```
[1] "nls"
```

```
size_hat = fitted.values(fit)  # store predicted values
error_hat = residuals(fit)

plot(age, size, pch = 19, col = "darkgrey")
lines(age, size_hat,
      col = "blue", lwd = 3, lty = 10)
```

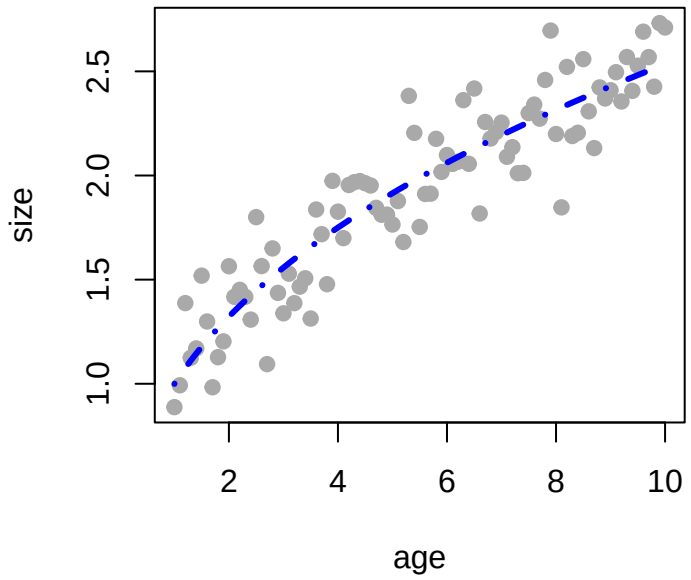


Figure 4: The fitted allometric growth equation using the nls routine in R.

Some diagnostics

```
par(mfrow = c(1,2))
hist(error_hat, probability = TRUE,
     xlab = expression(widehat(e)),
     cex.lab = 1.5, main = " ")
plot(age, error_hat, pch = 19,
     col = "red",
     ylab = expression(widehat(e)))
abline( h = 0, col = "blue",
        lwd = 3, lty = 2)
```

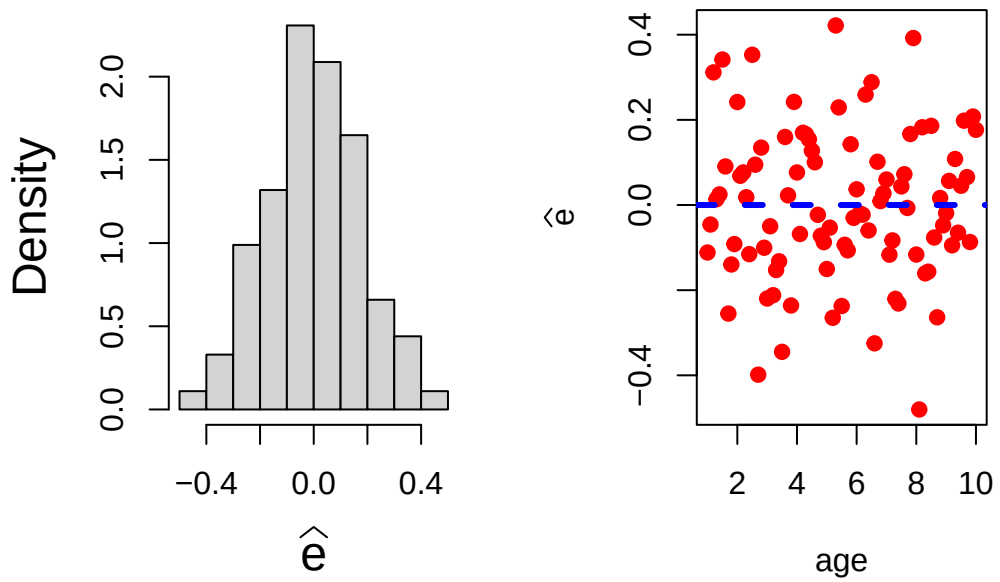


Figure 5: The behaviour of the residuals play the most critical role in nonlinear regression models.

Understanding the summary of nls

The `summary` function allows to understand the uncertainty associated with the estimated parameters.

```
1 summary(fit)
```

```
Formula: size ~ a * age^b

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a  0.99943    0.03719   26.88   <2e-16 ***
b  0.40405    0.02011   20.09   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1795 on 89 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 7.866e-06
```

```r
AIC(fit)
```

```
[1] -50.38962
```

```r
BIC(fit)
```

```
[1] -42.85704
```

Understanding the uncertainty of nls estimates

If we want to understand the uncertainty associated with the parameter estimates, we need to repeat the following process M times (say) and obtain M many estimates of a and b. The histograms of these estimated values will give us an idea how the estimates will vary if we repeatedly sample data from the same model population. Let us repeat the above task here $M = 1000$ times and obtain the estimates $\hat{a}$ and $\hat{b}$ in each repetition.

```r
M = 1000                              # number of repetitions
age = seq(1, 10, by = 0.1)              # age variable
a = 1                                   # true value of a
b = 0.4                                 # true value of b

a_hat = numeric(length = M)
b_hat = numeric(length = M)

for (i in 1:M) {
  set.seed(i)
  size = a*age^b + rnorm(n = length(age), mean = 0, sd = 0.2)
  data = data.frame(age, size)
  fit = nls(size ~ a*age^b, data = data,
            start = list(a = 0.5, b = 1))
  a_hat[i] = coefficients(fit)[1]
  b_hat[i] = coefficients(fit)[2]

}
par(mfrow = c(1,2))
hist(a_hat, probability = TRUE, xlab = expression(widehat(a)),
     main = " ", breaks = 30)
points(a, 0, pch = 19, col = "red", cex = 1.5)
hist(b_hat, probability = TRUE, xlab = expression(widehat(b)),
```

```
     main = " ", breaks = 30)
points(b, 0, pch = 19, col = "red", cex = 1.5)
```

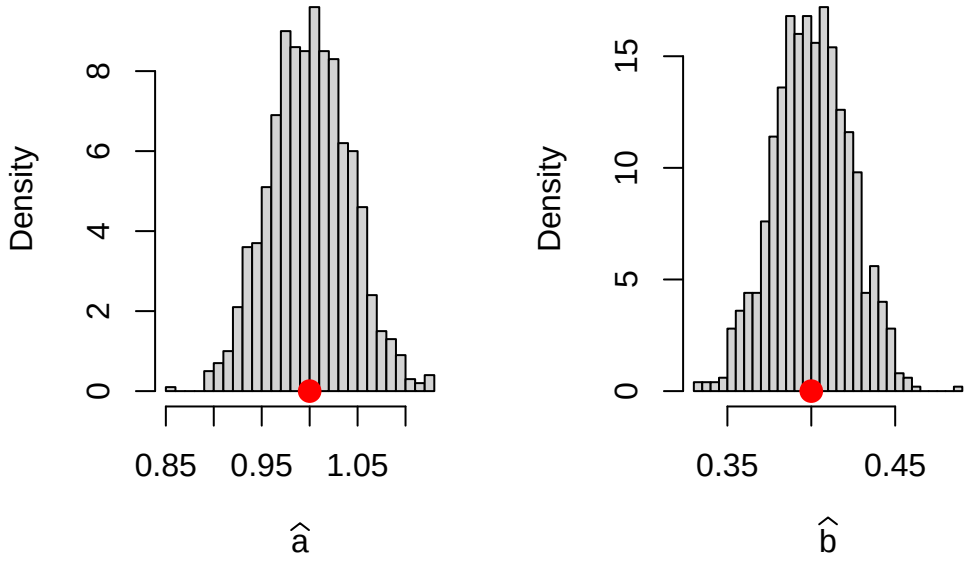


Figure 6: The approximate sampling distribution of the nonlinear least squares estimates of a and b. The sampling distribution is quiet well approxiamted by the normal distribution. In addition, note that the estimated values are centered about the true values of a and b using which the articial data sets have been simulated.

Homoschedasticity versus heteroschedasticity

The statistical model to perform the nonlinear regression model is of the following form:

$$\text{size}_i = a \times \text{age}_i^b + \text{e}_i, \ i \in \{1, 2, \dots, n\}.$$

In the previous simulations, we have consider that $Var(\text{e}_i) = \sigma^2$ for all $i \in \{1, 2, \dots, n\}$, that means, at different age, the variability in the size variable remains same about the true mean function. To simulate a scenario, we consider that $Var(\text{e}) = \sigma^2 \times \text{age}^2$. It means that as the age increases, the variability in size also increases.

```
par(mfrow = c(1,2))
a = 1
b = 0.4
age = seq(1, 20, by = 0.1)
```

```r
size = a*age^b + rnorm(n = length(age), mean = 0,
                       sd = 0.2*sqrt(age))
data = data.frame(age, size)
fit = nls(size ~ a*age^b, data = data,
          start = list(a = 0.5, b = 1))
size_hat = fitted.values(fit)  # store predicted values
error_hat = residuals(fit)

plot(age, size, pch = 19, col = "darkgrey")
lines(age, size_hat, col = "blue", lwd = 3, lty = 10)
plot(age, error_hat, pch = 19, col = "red",
     ylab = expression(widehat(e)))
abline( h = 0, col = "blue", lwd = 3, lty = 2)
```

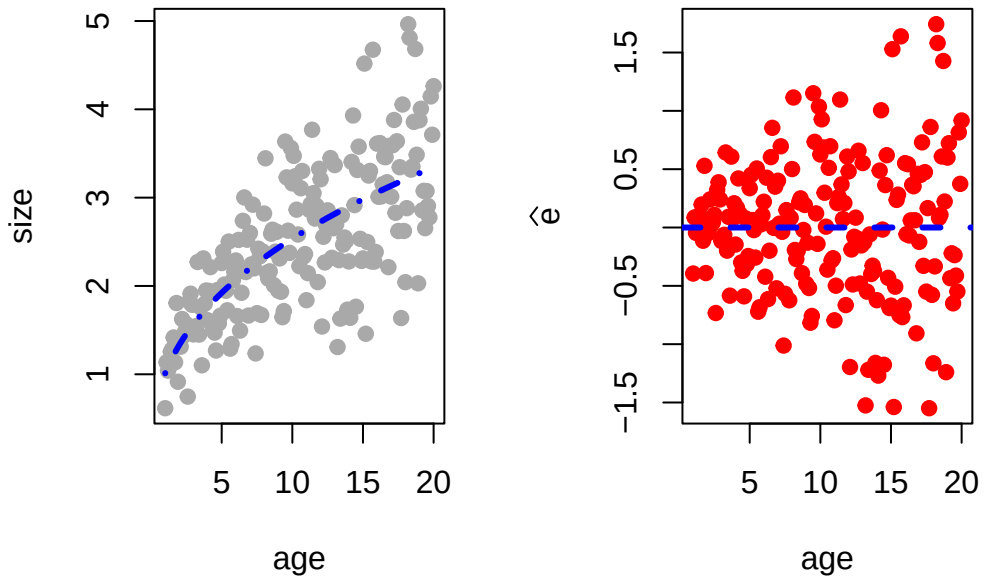


Figure 7: The simulated data gives clear indication of the presence of heteroschedasticity. The plot of residuals against the age gives whether the errors do not have a constant variance.

Confidence Interval and Prediction Interval

```r
a = 1
b = 0.4
```

```r
age = seq(1, 20, by = 0.1)
size = a*age^b + rnorm(n = length(age), mean = 0,
                       sd = 0.2)
data = data.frame(age, size)
fit = nls(size ~ a*age^b, data = data,
          start = list(a = 0.5, b = 1))
```

The nonlinear least squares estimates of a and b are obtained as

```r
coefficients(fit)
```

```
       a        b
0.981298 0.407464
```

The estimated variance covariance matrix of $\hat{a}$ and $\hat{b}$ is given by

```r
print(vcov(fit))
```

```
              a             b
a  0.0006579434 -0.0002611200
b -0.0002611200  0.0001081991
```

To obtain the confidence interval at a unseen value of `age`, say `age`*, we need to compute the variance of

$$\widehat{\texttt{size}}^* = \hat{a} \times \texttt{(age*)}^{\hat{b}}$$

. We observe that $\widehat{\texttt{size}}^*$ is a nonlinear function of $\hat{a}$ and $\hat{b}$ and we call it as $\psi\left(\hat{a}, \hat{b}\right)$ and we need to compute the variance of this quantity.

To approximate the variance of $\psi\left(\hat{a}, \hat{b}\right)$, we consider the first order Taylor's approximation about the true values of a and b as given below (neglecting the higher order terms):

$$\psi\left(\hat{a}, \hat{b}\right) = \psi(a, b) + (\hat{a} - a)\frac{\partial \psi}{\partial a} + \left(\hat{b} - b\right)\frac{\partial \psi}{\partial b}$$

Taking expectation on both sides, we obtain

$$\mathrm{E}\left(\psi\left(\hat{a}, \hat{b}\right)\right) \approx \psi(a, b)$$

and approximate variance of $\psi\left(\hat{a},\hat{b}\right)$ can obtained as

$$\begin{aligned}
\mathrm{E}\left(\psi\left(\hat{a},\hat{b}\right)-\psi(a,b)\right)^2 &\approx \mathrm{E}\left[(\hat{a}-a)\frac{\partial\psi}{\partial a}+\left(\hat{b}-b\right)\frac{\partial\psi}{\partial b}\right]^2 && (0.1)\\
&= Var(\hat{a})(\psi_a)^2+Var(\hat{b})(\psi_b)^2+2\times Cov(\hat{a},\hat{b})\psi_a\psi_b && (0.2)\\
&= \begin{bmatrix}\psi_a & \psi_b\end{bmatrix}\begin{bmatrix}Var(\hat{a}) & Cov(\hat{a},\hat{b})\\ Cov(\hat{a},\hat{b}) & Var(\hat{b})\end{bmatrix}\begin{bmatrix}\psi_a\\ \psi_b\end{bmatrix} && (0.3)
\end{aligned}$$

In the following code, we compute the variance of $Var\left(\psi\left(\hat{a},\hat{b}\right)\right)$. The estimate of the variance $\widehat{Var}\left(\psi\left(\hat{a},\hat{b}\right)\right)$ is obtained by evaluating $\frac{\partial\psi}{\partial a}$ and $\frac{\partial\psi}{\partial b}$ at $\hat{a}$ and $\hat{b}$.

```
a_hat = coefficients(fit)[1]
b_hat = coefficients(fit)[1]
psi_a = age^b_hat
psi_b = a_hat*age^b_hat*log(age)
size_var = numeric(length = length(age))
for(i in 1:length(age)){
  size_var[i] = (psi_a[i])^2*vcov(fit)[1,1] + (psi_b[i])^2*vcov(fit)[2,2] + 2*psi_a[i]*psi_b
}
```

We now construct an approximate 95% confidence interval which is given by

$$\left(\widehat{\texttt{size}}-1.96\times\sqrt{\widehat{Var}(\widehat{\texttt{size}})},\widehat{\texttt{size}}+1.96\times\sqrt{\widehat{Var}(\widehat{\texttt{size}})}\right).$$

In the construction of the confidence interval the irreducible error did not contribute anything. To compute the prediction interval, we need to consider the variation in the prediction due to sampling variation in the estimate of the parameters and also the random error component. Therefore, the variance of the $\widehat{\texttt{size}}$ is given by

$$Var\left(\widehat{\texttt{size}}\right)=Var\left(\psi\left(\hat{a},\hat{b}\right)\right)+Var(\hat{\epsilon})=Var\left(\psi\left(\hat{a},\hat{b}\right)\right)+\sigma^2.$$

Therefore the approximately 95% prediction interval is given by

$$\left(\widehat{\texttt{size}}\mp 1.96\times\sqrt{\widehat{Var}\left(\psi\left(\hat{a},\hat{b}\right)\right)+\widehat{\sigma^2}}\right)$$

```
par(mfrow = c(1,2))
plot(age, size, pch = 19, col = "darkgrey", cex.lab = 1.5,
     main = "Confidence Interval")
lines(age, fitted.values(fit)+1.96*sqrt(size_var), col = "blue", lwd = 2,
      lty = 2)
```

```r
lines(age, fitted.values(fit)-1.96*sqrt(size_var), col = "blue", lwd = 2,
      lty = 2)

plot(age, size, pch = 19, col = "darkgrey", cex.lab = 1.5,
     main = "Prediction Interval")
for(i in 1:length(age)){
  size_var[i] = (psi_a[i])^2*vcov(fit)[1,1] + (psi_b[i])^2*vcov(fit)[2,2] + 2*psi_a[i]*psi_b
}
lines(age, fitted.values(fit)+1.96*sqrt(size_var), col = "red", lwd = 2,
      lty = 2)
lines(age, fitted.values(fit)-1.96*sqrt(size_var), col = "red", lwd = 2,
      lty = 2)
```

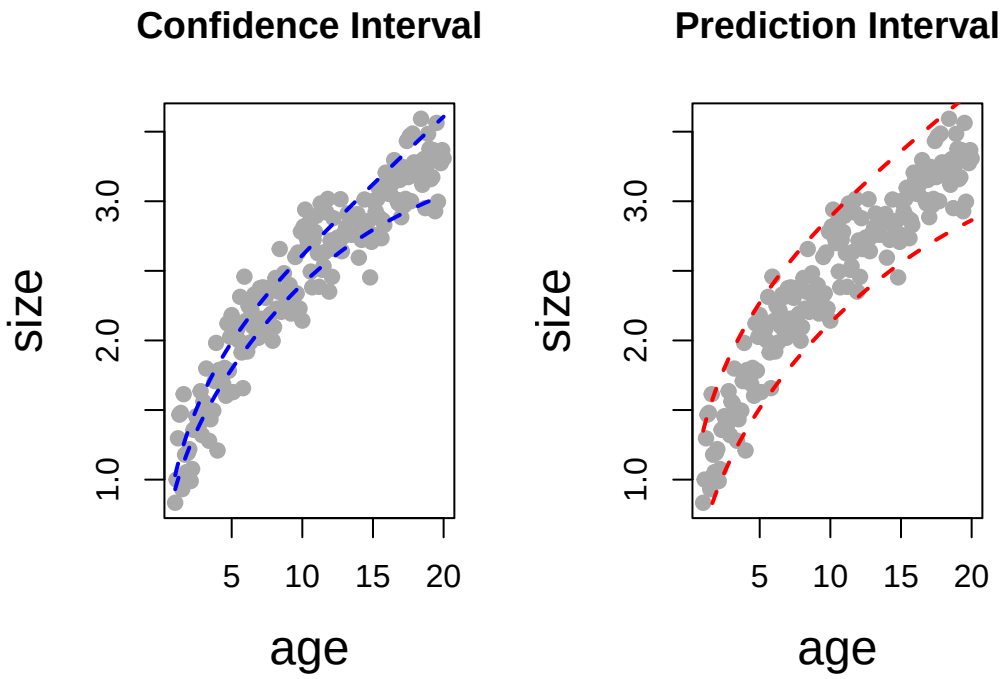


Figure 8: The demonstration of the confidence interval and prediction interval in the context of nonlinear regression model. It can be observed that the prediction interval is wider than the confidence interval. The prediction interval takes the variance of the error component also in account, where the confidence interval considers only the uncertainty associated with the estimated parameters.

Taylor's approximation for one variable

In computing the confidence and prediction interval, we have chosen the normal distribution. It is important to check how accurate these approximations are. To understand this concept, we consider a simple example. Suppose that $X_1, X_2, \ldots, X_n$ be a random sample from the Exponential distribution with rate parameter λ. We are interested to estimate the parameter λ. We know the following from the Central Limit Theorem that for large n

$$\overline{X_n} \overset{a}{\sim} \mathcal{N}\left(\frac{1}{\lambda}, \frac{1}{\lambda^2 n}\right).$$

By the method of moment, we see that the Method of Moment Estimator for λ is $\frac{1}{\overline{X_n}} = \overline{X_n}^{-1} = \psi\left(\overline{X_n}\right)$ (say).

By the first order Taylor's approximation of $\psi\left(\overline{X_n}\right)$ about $\frac{1}{\lambda}$, we obtain

$$\psi\left(\overline{X_n}\right) \approx \psi\left(\frac{1}{\lambda}\right) + \left(\overline{X_n} - \frac{1}{\lambda}\right)\psi'\left(\frac{1}{\lambda}\right)$$

Now taking expectation on both sides

$$\mathrm{E}\left[\psi\left(\overline{X_n}\right)\right] \approx \psi\left(\frac{1}{\lambda}\right) = \lambda.$$

For computing the variance, we see that

$$\mathrm{E}\left[\psi\left(\overline{X_n}\right) - \lambda\right]^2 \approx \mathrm{E}\left[\left(\overline{X_n} - \frac{1}{\lambda}\right)\psi'\left(\frac{1}{\lambda}\right)\right]^2 = Var_\lambda\left(\overline{X_n}\right)\lambda^4 = \frac{\lambda^2}{n},$$

provided $\psi'\left(\frac{1}{\lambda}\right) \neq 0$. Therefore, the approximately $Var\left(\frac{1}{\overline{X_n}}\right) \approx \frac{\lambda^2}{n}$. Let us verify the same using computer simulation. In addition, we overlay a normal density function with mean λ and variance $\frac{\lambda^2}{n}$ on the histograms for different values of n. The histograms are simulated using 1000 replications.

```
par(mfrow = c(2,3))
lambda = 3
n_vals = c(3,5, 10, 25, 50,100)
rep = 1000
for(n in n_vals){
  psi_xbar = numeric(length = rep)
  for(i in 1:rep){
    x = rexp(n = n, rate = lambda)
    psi_xbar[i] = 1/mean(x)
  }
  psi_xbar
```

```r
  var(psi_xbar)
  lambda^2/n
  hist(psi_xbar, probability = TRUE, main = paste("n = ", n),
       xlab = expression(psi(bar(X[n]))))
  curve(dnorm(x, mean = lambda, sd = sqrt(lambda^2/n)),
        add = TRUE, col = "red", lwd = 2)

}
```

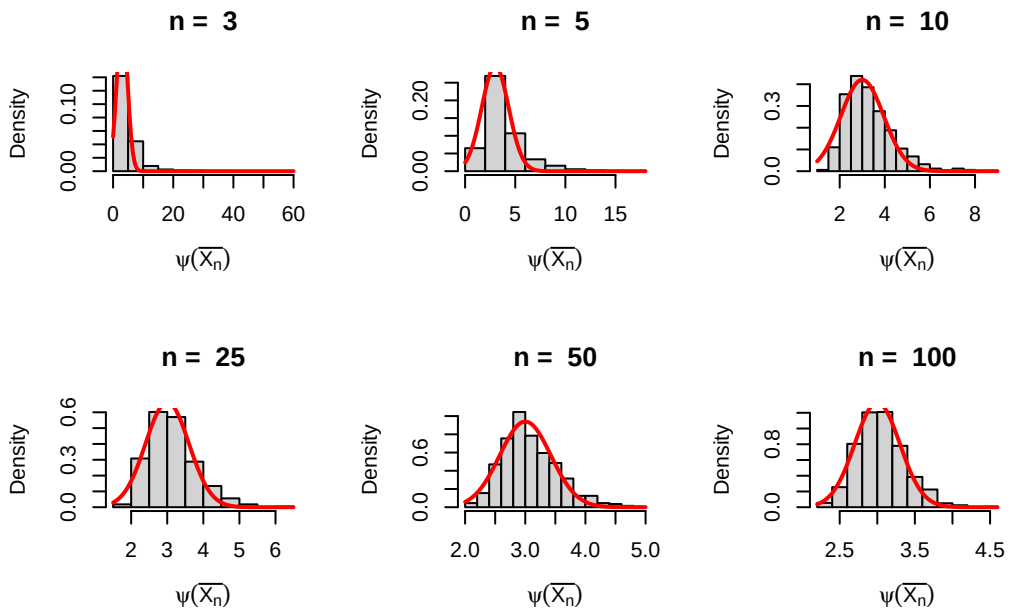


Figure 9: As the sample size increases, the sampling distribution of the function of the sample mean is well approximated by a normal distribution. This approximation is known as the Delta method. The histograms are obtained by repeatedly sampling 1000 times from the exponential distribution with rate parameter $\lambda = 3$.

Bootstrapping regression model

In the following code, we investigate how we can compute the standard of the estimate using the nonparametric bootstrap procedure.

```r
a = 1
b = 0.4
age = seq(1, 20, by = 0.1)
set.seed(123)
```

```r
size = a*age^b + rnorm(n = length(age), mean = 0,
                       sd = 0.2)
plot(age, size, pch = 19, col = "darkgrey", cex.lab = 1.5)
data = data.frame(age, size)


B = 1000           # number of bootstrap data set
a_hat = numeric(length = B)
b_hat = numeric(length = B)
for (i in 1:B) {
  ind = sample(1:nrow(data), replace = TRUE)
  boot_data = data[ind, ]
  boot_fit = nls(size ~ a*age^b,
                 data = boot_data,
                 start = list(a = 1, b = 1))
  a_hat[i] = coefficients(boot_fit)[1]
  b_hat[i] = coefficients(boot_fit)[2]
}

mean(a_hat)            # bootstrap mean
```

```
[1] 1.008809
```

```r
sd(a_hat)              # bootstrap standard error or a_hat
```

```
[1] 0.02566318
```

```r
mean(b_hat)
```

```
[1] 0.3965658
```

```r
sd(b_hat)
```

```
[1] 0.01025991
```

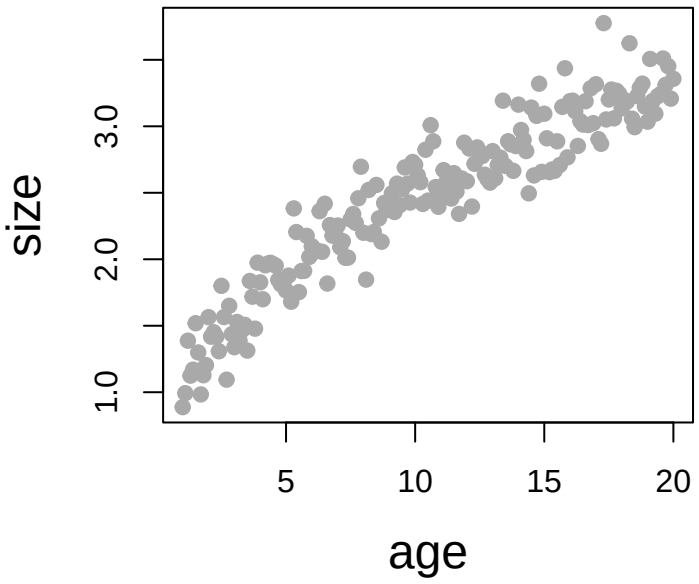


Figure 10: We fixed the set.seed(123) and simulate the data sets from the population nonlinear regression model. The parameters are fixed as $a = 1$ and $b = 0.4$ and the errors are assumed to be normally distributed. This data set will be resampled to obtain the bootrstrap estimate of the parmaeters and also compute the bootstrap standard error of the estimates.

We now visualize the sampling distribution of $\hat{a}$ and $\hat{b}$ based on $B = 1000$ bootstrap samples.

```
par(mfrow = c(1,2))
hist(a_hat, probability = TRUE,
     main = "B = 1000",
     xlab = expression(widehat(a)))
cat("95% bootstrap CI for a based on normal distribution is given as \n")
```

```
95% bootstrap CI for a based on normal distribution is given as
```

```
c(mean(a_hat) - 1.96*sd(a_hat), mean(a_hat) + 1.96*sd(a_hat))
```

```
[1] 0.958509 1.059109
```

```
cat("nonparametric 95% CI for a is \n")
```

```
nonparametric 95% CI for a is
```

```r
quantile(a_hat, c(2.5, 97.5)/100)
```

```
    2.5%    97.5%
0.958737 1.061958
```

```r
hist(b_hat, probability = TRUE,
     main = "B = 1000",
     xlab = expression(widehat(b)))
cat("95% bootstrap CI for b based on normal distribution is given as \n")
```

```
95% bootstrap CI for b based on normal distribution is given as
```

```r
c(mean(a_hat) - 1.96*sd(a_hat), mean(a_hat) + 1.96*sd(a_hat))
```

```
[1] 0.958509 1.059109
```

```r
cat("nonparametric 95% CI for b is \n")
```

```
nonparametric 95% CI for b is
```

```r
quantile(b_hat, c(2.5, 97.5)/100)
```

```
     2.5%     97.5%
0.3761927 0.4169939
```

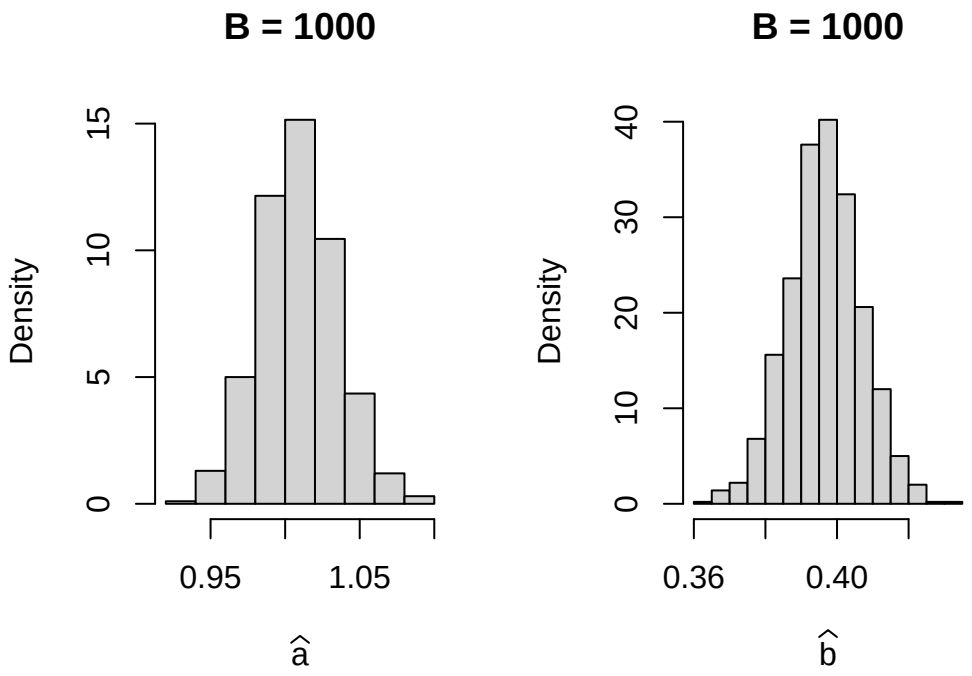


Figure 11: The bootstrap sampling distribution of $\hat{a}$ and $\hat{b}$ based on 1000 bootstrap samples. It may be noted that the bootstarp estimates are centered about the nls estimates of the parmaeters a and b based on the complete data sets, not centered around the true values of a and b using which the data has been simulated.

Case study using synthetic data generation

Suppose that we have the population regression function $f(x|\mathbf{b})$ parameterized by $\mathbf{b} = (b_0, b_1)'$ and the statistical model for the data is given by

$$y_i = f(x|\mathbf{b}) + \epsilon_i = \frac{b_0 x}{b_1 + x} + \epsilon_i, i \in \{1, 2, \ldots, n\}.$$

We assume that the errors ϵ_is are normally distributed with mean 0 and variance σ^2 and they are also independent. Statisticians formally call it as IID (Independent and Identically Distributed). In following code, we first simulate the synthetic data by fixing the population parameters b_0, b_1, σ^2.

```
par(mfrow = c(1,3))
# Plot the mean function
set.seed(1234)
b0 = 1
b1 = 0.5
```

```r
f = function(x){
  b0*x/(b1+x)
}
curve(f(x), col = "red", 0, 2.5, lwd = 2)

sigma2 = 0.01          # variance
x = seq(0, 2, by = 0.1)
print(x)
```

```
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8
[20] 1.9 2.0
```

```r
n = length(x)         # length of the data
y = b0*x/(b1+x) + rnorm(n = n, mean = 0, sd = sqrt(sigma2))
print(y)
```

```
 [1] -0.1207066  0.1944096  0.3941584  0.1404302  0.4873569  0.5506056
 [7]  0.4879805  0.5286701  0.5589394  0.5538534  0.6189474  0.5876614
[13]  0.6282570  0.7286681  0.8327915  0.7389715  0.7108038  0.6816077
[19]  0.6988915  1.0332502  0.8134088
```

```r
plot(x, y, col = "grey", pch = 19, cex = 1.2)

data = matrix(data = NA, nrow = 5, ncol = length(x))
for(i in 1:nrow(data)){
  data[i,] = b0*x/(b1+x) + rnorm(n = n, mean = 0, sd = sqrt(sigma2))
}
matplot(x, t(data), col = 1:10, pch = 19, ylab = "y")
```

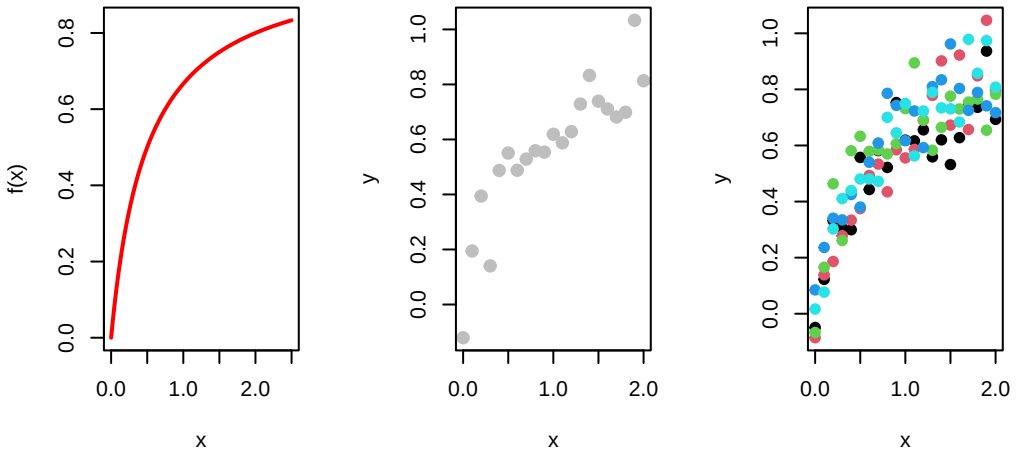


Figure 12: The left most panel is the mean population regression function and the right panel contains the simulated data using the seed value 1234. The seed value is used ensure the reproducibility of the plots. For simulation, the parameter choices are set as $b_0 = 1, b_1 = 0.5, \sigma^2 = 0.01$. In the right most panel some more simulation has been carried to demonstrate the randomness across different simulated data sets.

We consider the minimization of the error sum of squares as the first approach to estimate the parameters. We minimize the following function with respect to b_0 and b_1.

$$\text{ESS}(\mathbf{b}) = \sum_{i=1}^{n} \left(y_i - \frac{b_0 x_i}{b_1 + x_i} \right)^2 .$$

We first plot the surface of the ESS($\mathbf{b}$) with different choices of b_0 and b_1. For the user, show the plots using the `persp()` function and also by using `plot3D` package.

```
fun_ESS = function(b){
  b0 = b[1]
  b1 = b[2]
  sum((y - b0*x/(b1+x))^2)
}
b0_vals = seq(0.01,2, by = 0.05)
b1_vals = seq(0.01,2, by = 0.05)

ESS_vals = matrix(data = NA, nrow = length(b0_vals),
                  ncol = length(b1_vals))
print(head(ESS_vals))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
[2,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
[3,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
[4,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
[5,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
[6,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
     [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
[1,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[2,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[3,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[4,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[5,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[6,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
     [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
[1,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[2,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[3,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[4,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[5,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
[6,]    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
     [,39] [,40]
[1,]    NA    NA
[2,]    NA    NA
[3,]    NA    NA
[4,]    NA    NA
[5,]    NA    NA
[6,]    NA    NA
```

```r
for(i in 1:length(b0_vals)){
  for(j in 1:length(b1_vals)){
    ESS_vals[i,j] = fun_ESS( c(b0_vals[i], b1_vals[j]) )
  }
}
print(head(ESS_vals))
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
[1,] 7.751635 7.764175 7.774684 7.783834 7.791970 7.799305 7.805982 7.812108
[2,] 6.636982 6.704601 6.762262 6.812939 6.858289 6.899368 6.936911 6.971460
[3,] 5.618948 5.728946 5.824768 5.909939 5.986727 6.056670 6.120874 6.180175
[4,] 4.697534 4.837211 4.962204 5.074834 5.177284 5.271210 5.357872 5.438250
[5,] 3.872741 4.029396 4.174570 4.307624 4.429960 4.542989 4.647903 4.745687
[6,] 3.144567 3.305502 3.461865 3.608309 3.744755 3.872006 3.990969 4.102485
```

```
         [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
[1,] 7.817761 7.823003 7.827886 7.832450 7.836729 7.840753 7.844546 7.848129
[2,] 7.003432 7.033155 7.060896 7.086875 7.111276 7.134257 7.155951 7.176473
[3,] 6.235221 6.286534 6.334541 6.379595 6.421996 6.461998 6.499822 6.535658
[4,] 5.513127 5.583142 5.648822 5.710611 5.768888 5.823977 5.876160 5.925683
[5,] 4.837152 4.922978 5.003738 5.079923 5.151953 5.220193 5.284965 5.346548
[6,] 4.207294 4.306042 4.399291 4.487531 4.571190 4.650647 4.726236 4.798253
         [,17]    [,18]    [,19]    [,20]    [,21]    [,22]    [,23]    [,24]
[1,] 7.851521 7.854738 7.857794 7.860702 7.863474 7.866119 7.868646 7.871063
[2,] 7.195926 7.214397 7.231966 7.248701 7.264665 7.279913 7.294496 7.308457
[3,] 6.569673 6.602014 6.632812 6.662183 6.690230 6.717047 6.742717 6.767318
[4,] 5.972762 6.017588 6.060333 6.101147 6.140168 6.177519 6.213311 6.247646
[5,] 5.405192 5.461119 5.514527 5.565594 5.614480 5.661331 5.706277 5.749440
[6,] 4.866964 4.932607 4.995396 5.055524 5.113166 5.168482 5.221616 5.272701
         [,25]    [,26]    [,27]    [,28]    [,29]    [,30]    [,31]    [,32]
[1,] 7.873379 7.875598 7.877729 7.879775 7.881743 7.883636 7.885460 7.887218
[2,] 7.321840 7.334680 7.347012 7.358866 7.370271 7.381254 7.391838 7.402045
[3,] 6.790919 6.813581 6.835364 6.856318 6.876494 6.895935 6.914683 6.932775
[4,] 6.280615 6.312302 6.342785 6.372133 6.400412 6.427681 6.453997 6.479409
[5,] 5.790928 5.830843 5.869275 5.906309 5.942024 5.976491 6.009778 6.041946
[6,] 5.321860 5.369203 5.414834 5.458847 5.501331 5.542366 5.582028 5.620386
         [,33]    [,34]    [,35]    [,36]    [,37]    [,38]    [,39]    [,40]
[1,] 7.888913 7.890549 7.892130 7.893658 7.895135 7.896565 7.897949 7.899291
[2,] 7.411896 7.421409 7.430603 7.439494 7.448097 7.456426 7.464495 7.472315
[3,] 6.950246 6.967129 6.983455 6.999250 7.014542 7.029354 7.043710 7.057631
[4,] 6.503965 6.527710 6.550683 6.572924 6.594468 6.615348 6.635595 6.655239
[5,] 6.073052 6.103150 6.132290 6.160518 6.187877 6.214408 6.240150 6.265137
[6,] 5.657507 5.693451 5.728274 5.762030 5.794768 5.826535 5.857374 5.887327
```

```r
par(mfrow = c(1,2))
persp(b0_vals, b1_vals, ESS_vals,
      theta = 30, xlab = "b0", ylab = "b1",
      zlab = "ESS", col = "grey")

# beautiful surface plot
library(plot3D)
persp3D(b0_vals, b1_vals, ESS_vals)
```

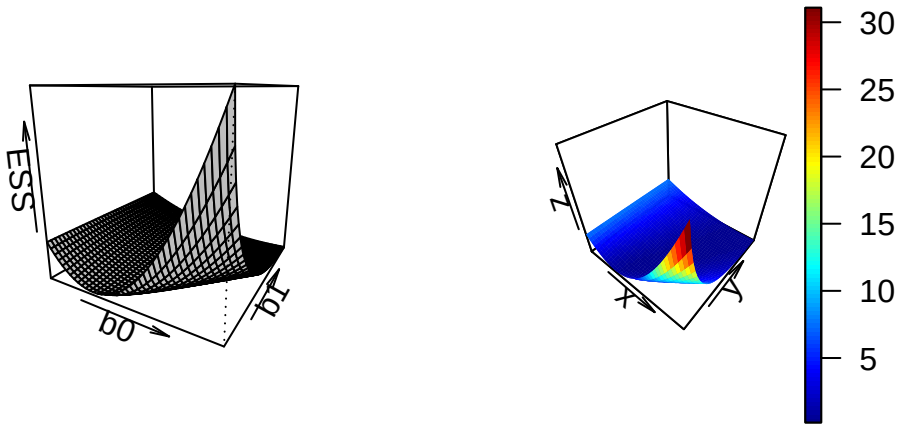


Figure 13: The left panel shows the surface plto using the persp function and the right panel shows the surface with color gradient.

In the following code, we use the function `optim()` to minimize the error sum squares function with respect to b_0 and b_1.

```
out = optim(c(1.5, 1), fn = fun_ESS)  # minimize the function
summary(out)
```

```
            Length Class  Mode
par         2      -none- numeric
value       1      -none- numeric
counts      2      -none- numeric
convergence 1      -none- numeric
message     0      -none- NULL
```

```
b_hat = out$par                       # extract the estimates
print(b_hat)
```

```
[1] 1.0820661 0.6865781
```

In the following, we obtain the estimates of the parameter by using the method of maximum likelihood. The likelihood function is given by

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} f(y_i|x_i, b_0, b_1, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}\left(y_i - \frac{b_0 x_i}{b_1 + x_i}\right)^2}$$

which simplifies to

$$\mathcal{L}(b_0, b_1, \sigma^2) = \left(\frac{1}{\sigma^2 2\pi}\right)^{n/2} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \frac{b_0 x_i}{b_1 + x_i}\right)^2}.$$

We write down the log-likelihood function as

$$l\left(b_0, b_1, \sigma^2\right) = -\frac{n}{2}\log(\sigma^2) - \frac{n}{2}\log(2\pi) - \sum_{i=1}^{n}\left(y_i - \frac{b_0 x_i}{b_1 + x_i}\right)^2.$$

Instead of maximizing the log-likelihood function, we can minimize the negative log-likelihood function $-l(b_0, b_1, \sigma^2)$.

```
# Maximum Likelihood Estimation
n = length(x)
fun_logLik = function(b){
  b0 = b[1]
  b1 = b[2]
  sigma2 = b[3]

  -(n/2)*log(sigma2)-(n/2)*log(2*pi) - (1/(2*sigma2))*sum((y-b0*x/(b1+x))^2)
}

fun_neglogLik = function(b){
  b0 = b[1]
  b1 = b[2]
  sigma2 = b[3]

  (n/2)*log(sigma2) + (n/2)*log(2*pi) + (1/(2*sigma2))*sum((y-b0*x/(b1+x))^2)
}
out = optim(par = c(1,1,0.1), fn = fun_neglogLik)
out$par
```

```
[1] 1.081961018 0.686410056 0.009225206
```

The estimate that we have obtained by employing the method of maximum likelihood is subject to uncertainty due to the random nature of the data set. Therefore, we need to report the uncertainty or the standard error of the estimate. We omit the following result without proof which states that for large sample size the variance covariance matrix of $\widehat{b_0}$, $\widehat{b_1}$ and $\widehat{\sigma^2}$ is well approximated by the inverse of the Hessian matrix evaluated at the MLE with a negative sign.

$$-H = -\begin{bmatrix} \frac{\partial^2 l}{\partial b_0^2} & \frac{\partial^2 l}{\partial b_0 \partial b_1} & \frac{\partial^2 l}{\partial b_0 \partial \sigma^2} \\ \frac{\partial^2 l}{\partial b_0 \partial b_1} & \frac{\partial^2 l}{\partial b_1^2} & \frac{\partial^2 l}{\partial b_1 \partial \sigma^2} \\ \frac{\partial^2 l}{\partial \sigma^2 \partial b_0} & \frac{\partial^2 l}{\partial \sigma^2 \partial b_1} & \frac{\partial^2 l}{\partial (\sigma^2)^2} \end{bmatrix}_{|\left(\widehat{b_0}, \widehat{b_1}, \widehat{\sigma^2}\right)}$$

In the `optim` function, we can use the argument `hessian = TRUE` to obtain the estimated Hessian matrix at the MLE. In the

```
out = optim(par = c(1,1,0.1), fn = fun_neglogLik,
            hessian = TRUE)
out$par
```

```
[1] 1.081961018 0.686410056 0.009225206
```

```
out$hessian
```

```
             [,1]         [,2]          [,3]
[1,]  721.5627924 -414.1891424 -1.328736e-01
[2,] -414.1891424  256.2930941  1.680828e-01
[3,]   -0.1328736    0.1680828  1.325758e+05
```

```
H = out$hessian
cat("The Hessian matrix evaluate at the MLE is given by \n")
```

```
The Hessian matrix evaluate at the MLE is given by
```

```
print(H)
```

```
             [,1]         [,2]          [,3]
[1,]  721.5627924 -414.1891424 -1.328736e-01
[2,] -414.1891424  256.2930941  1.680828e-01
[3,]   -0.1328736    0.1680828  1.325758e+05
```

```
cat("The inverse of the Hessian matrix with negative sign is given by \n")
```

```
The inverse of the Hessian matrix with negative sign is given by
```

```r
solve(H)
```

```
              [,1]          [,2]          [,3]
[1,]  1.915649e-02  3.095835e-02 -2.005022e-08
[2,]  3.095835e-02  5.393283e-02 -3.734945e-08
[3,] -2.005022e-08 -3.734945e-08  7.542853e-06
```

Therefore, the square root of the diagonal entries of the matrix $(-H)^{-1}$ will give the standard error of the MLE. A natural question arises how good these approximations are? To see this, we can visualize the sampling distribution of

$$W_0 = \frac{\widehat{b_0} - b_0}{\widehat{\mathrm{SE}}(\widehat{b_0})}$$

and

$$W_1 = \frac{\widehat{b_1} - b_1}{\widehat{\mathrm{SE}}(\widehat{b_1})}$$

by computer simulation based on 1000 replications.

```r
rep = 1000
w_0 = w_1 = numeric(length = rep)
for(i in 1:rep){
  x = seq(0, 2, by = 0.05)
  n = length(x)          # length of the data
  y = b0*x/(b1+x) + rnorm(n = n, mean = 0, sd = sqrt(sigma2))
  out = optim(par = c(1,1,0.1), fn = fun_neglogLik,
              hessian = TRUE)
  H = out$hessian
  w_0[i] = (out$par[1] - b0)/sqrt(solve(out$hessian)[1,1])
  w_1[i] = (out$par[2] - b1)/sqrt(solve(out$hessian)[2,2])
}

par(mfrow = c(1,2))
hist(w_0, probability = TRUE, xlab = expression(W[0]),
     main = paste("n = ", n), breaks = 30)
curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
hist(w_1, probability = TRUE, xlab = expression(W[1]),
     main = paste("n = ", n), breaks = 30)
curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
```

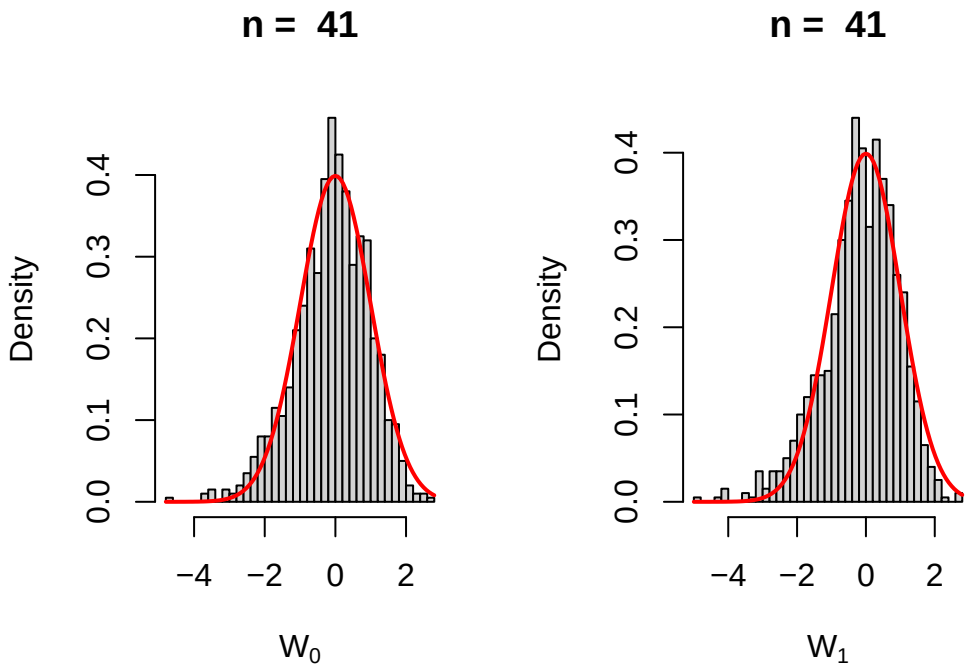


Figure 14: The sampling distribution of the estimator centered at the true value and scaled by the estimated standard error. An approximation with the standard normal distribution is shown for reference.

The simulations suggest that for large sample size n,

$$W_0 = \frac{\widehat{b_0} - b_0}{\widehat{\text{SE}}(\widehat{b_0})} \sim \mathcal{N}(0, 1),$$

there for

$$\widehat{b_0} \sim \mathcal{N}\left(b_0, \widehat{\text{SE}}\left(\widehat{b_0}\right)^2\right),$$

for large sample size n. Therefore, large n, $(1-\alpha)\%$ confidence interval for b_0 can be obtained as

$$\left(\widehat{b_0} - z_{\alpha/2}\widehat{\text{SE}}(\widehat{b_0}), \widehat{b_0} + z_{\alpha/2}\widehat{\text{SE}}(\widehat{b_0})\right),$$

where $\text{P}(Z > z_{\alpha/2}) = \frac{\alpha}{2}$ and $Z \sim \mathcal{N}(0, 1)$.

Classroom Quiz: Simulation

Perform the simulation study with the regression function

$$f(x) = \frac{b_0 x^2}{b_1^2 + x^2},$$

where $b_0, b_1 > 0$ and study the sampling distributions of the MLEs.

Case Study: Local maximization

In the classroom, I have randomly asked Shailaja to suggest a nonlinear function which will be used as the systematic component of the nonlinear regression model to simulate the data. She suggested the following function

$$f(x) = \sin(ax) + bx^2, x \in [0, 2],$$

Therefore, the statistical modelling framework is given by

$$y_i = f(x_i) + \epsilon_i, \quad i \in \{1, 2, \dots, n\}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. We simulate the data from the above regression model and estimate the model parameters a, b, σ^2 using the method of maximum likelihood.

In the first step, we simulate a sample of size n by fixing the parameters $a = 3, b = 1$ and $\sigma^2 = 0.05$. We use $\theta = (a, b, \sigma^2)'$ to denote the parameter vector.

```
# The mean regression function
a = 3
b = 1

f = function(x){
  sin(a*x) + b*x^2                    # population nonlinear regression function
}

n = 30                                # sample size
sigma2 = 0.1                          # population standard deviation

x = seq(0, 2, length.out = n)         # sequence of x values
y = numeric(length = n)               #
for(i in 1:n){
  y[i] = rnorm(n = 1, mean = f(x[i]), sd = sqrt(sigma2))
}
print(y)
```

```
 [1] 0.10143029 0.02054028 0.17456038 0.79865840 0.76382977 1.13188772
 [7] 1.23641594 0.90970110 1.57312055 1.63766518 1.19998373 0.93721434
[13] 1.56414402 1.05069019 1.20524024 1.62027205 0.56408903 1.27803134
[19] 1.44679588 0.82460828 1.66001008 1.34018311 0.52078554 1.95107356
[25] 2.08452032 1.94083023 2.48565244 2.30070775 3.06211908 3.48298509
```

```r
plot(x,y, col = "darkgrey", pch = 19, cex = 1.2)
curve(f(x), add = TRUE, col = "red", lwd = 2)
```

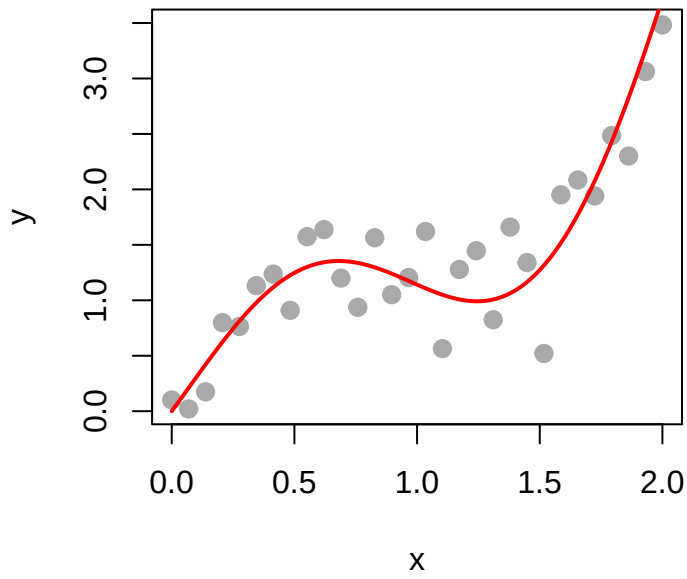


Figure 15: The simulated data set and the population regression mean function is overlaid for reference.

In the following R codes, we minimize the negative of the log-likelihood function. The reader is encouraged to write down the expression of the likelihood function explicitly and plot the surface of the likelihood function for different choices of a and b.

```r
fun_neglogLik = function(theta){
  a = theta[1]
  b = theta[2]
  sigma2 = theta[3]

  (n/2)*log(sigma2) + (n/2)*log(2*pi) + (1/(2*sigma2))*sum((y - sin(a*x) - b*x^2)^2)
}

out = optim(par = c(2.8,0.9,0.04), fn = fun_neglogLik,
            hessian = TRUE)
a_hat = out$par[1]
b_hat = out$par[2]
sigma2_hat = out$par[3]
```

The maximum likelihood estimates of the parameters are given below. In addition, we add both

the population nonlinear regression function and the estimated (sample) nonlinear regression function for reference. The estimated regression function is given by

$$\hat{f}(x) = \sin(\hat{a}x) + \hat{b}x^2.$$

```
print(a_hat)
```

```
[1] 2.878296
```

```
print(b_hat)
```

```
[1] 0.9969667
```

```
print(sigma2_hat)
```

```
[1] 0.09904459
```

```
plot(x,y, col = "darkgrey", pch = 19, cex = 1.2)
curve(f(x), add = TRUE, col = "red", lwd = 2)
curve(sin(a_hat*x) + b_hat*x^2, add = TRUE,
      col = "blue", lwd = 2)
legend("topleft", legend = c("f(x)", expression(widehat(f)(x))),
       col = c("red", "blue"), lwd = c(2,2), bty = "n")
```

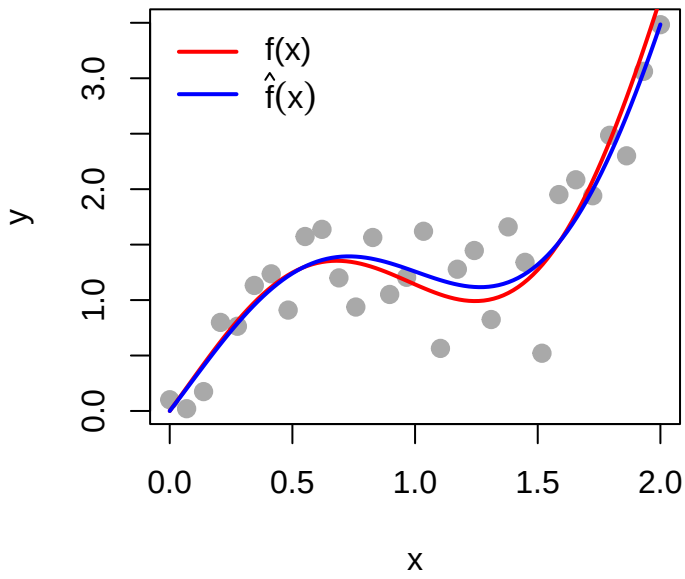


Figure 16: The estimated regression function and the population regression function is shown using different colour and they are in good agreement.

Certainly, the above estimates are subject to uncertainty. Basically, if we simulate another set of data set and compute the MLE, certainly, these estimates are going to differ. However, we want to get an idea, how much they can differ! To obtain the Hessian matrix (H) will be useful. The estimate of the variance of the MLE of $\hat{a}$ is given by H^{-1}_{11}, evaluated at $\hat{\theta}$.

```
H = out$hessian
print(H)
```

```
             [,1]          [,2]          [,3]
[1,] 166.11774761   53.77109274   -0.04929243
[2,]  53.77109274 1019.58001971   -0.09748603
[3,]  -0.04929243   -0.09748603 1529.70159162
```

```
solve(H)
```

```
              [,1]          [,2]         [,3]
[1,]  6.124376e-03 -3.229902e-04 1.767654e-07
[2,] -3.229902e-04  9.978300e-04 5.318261e-08
[3,]  1.767654e-07  5.318261e-08 6.537223e-04
```

To understand the uncertainty associated with these estimates, first we simulate the sampling distribution of $\hat{\theta}$ by repeating the simulation experiment M times. We simulate the sampling distribution of $\hat{a}$, $\hat{b}$ and $\widehat{\sigma^2}$.

We also compute the following three quantities:

$$W_a = \frac{\hat{a} - a}{\widehat{\text{SE}}(\hat{a})}, \qquad W_b = \frac{\hat{b} - b}{\widehat{\text{SE}}(\hat{b})}, \qquad W_{\sigma^2} = \frac{(n-2)\widehat{\sigma^2}}{\sigma^2}.$$

```
M = 1000
a_hat = numeric(length = M)
b_hat = numeric(length = M)
sigma2_hat = numeric(length = M)

se_a_hat = numeric(length = M)
se_b_hat = numeric(length = M)
se_sigma2_hat = numeric(length = M)

for (j in 1:M) {
  x = seq(0, 2, length.out = n)
  y = numeric(length = n)
```

```r
  for(i in 1:n){
    y[i] = rnorm(n = 1, mean = f(x[i]), sd = sqrt(sigma2))
  }

  out = optim(par = c(2.8,0.9,0.04), fn = fun_neglogLik,
              hessian = TRUE)
  a_hat[j] = out$par[1]
  b_hat[j] = out$par[2]
  sigma2_hat[j] = out$par[3]

  H = out$hessian
  se_a_hat[j] = sqrt(solve(H)[1,1])
  se_b_hat[j] = sqrt(solve(H)[2,2])
  se_sigma2_hat[j] = sqrt(solve(H)[3,3])

}

par(mfrow = c(2,3))
hist(a_hat, probability = TRUE, xlab = expression(widehat(a)),
     main = paste("n = ", n))
points(a, 0, pch = 19, cex = 1.5, col = "red")
hist(b_hat, probability = TRUE, xlab = expression(widehat(b)),
     main = paste("n = ", n))
points(b, 0, pch = 19, cex = 1.5, col = "red")
hist(sigma2_hat, probability = TRUE, xlab = expression(widehat(sigma^2)),
     main = paste("n = ", n))
points(sigma2, 0, pch = 19, cex = 1.5, col = "red")

hist((a_hat-a)/se_a_hat, probability = TRUE, xlab = expression(widehat(a)),
     main = paste("n = ", n))
curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
hist((b_hat-b)/se_b_hat, probability = TRUE, xlab = expression(widehat(b)),
     main = paste("n = ", n))
curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
hist((n-2)*sigma2_hat/sigma2 , probability = TRUE,
     xlab = expression(over((n-2)*widehat(sigma^2),sigma^2)),
     main = paste("n = ", n), cex.lab = 0.6)
curve(dchisq(x, df = n-2), add = TRUE, col = "red", lwd = 2)
```

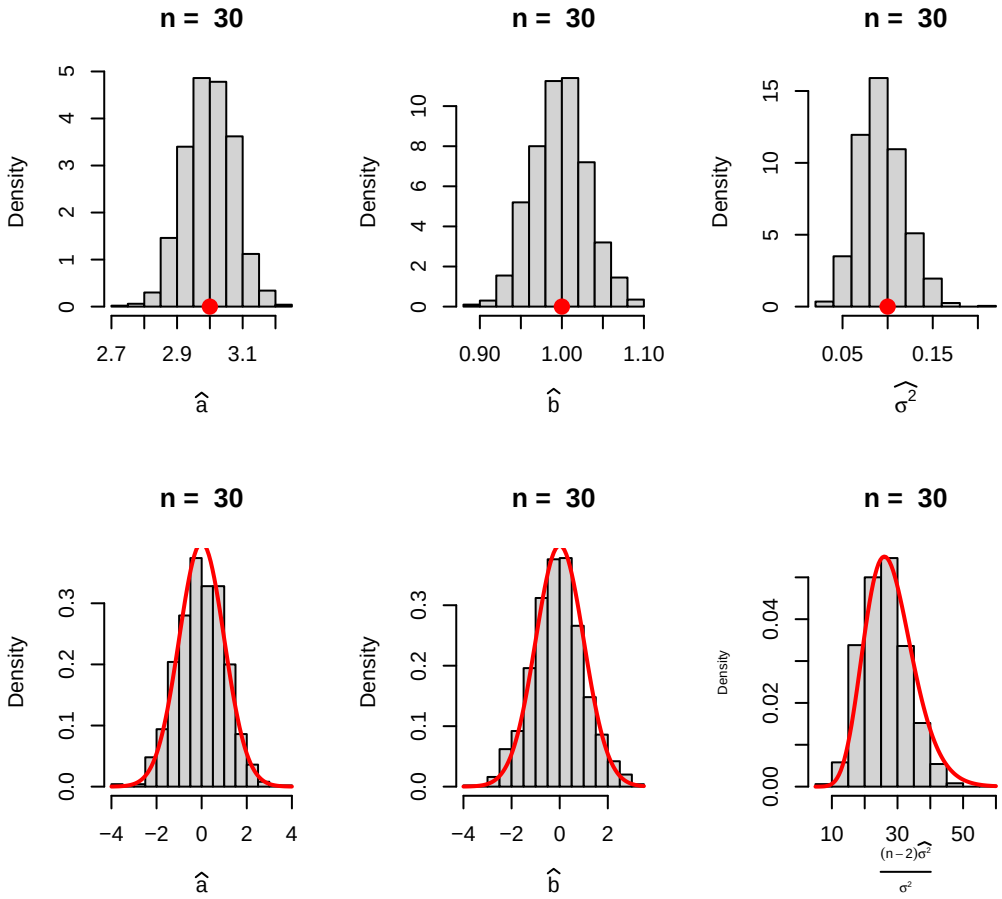


Figure 17: The top panel represents the simulated sampling distribution of the MLEs based on $M = 1000$ replications. The over red dot indicates the true value of the parameters using which the training data sets have been simulated. It is interesting to see that, MLEs are centered about the true valye, ensuring unbiasedness of the MLE. In the lower panel, the simulated distributions of W_a, W_b and W_{σ^2} are drawn. The standard normal distribution is overlaid on the first two histograms (bottom panel) and the last figure is overlaid with a χ^2 distribution with $(n-2)$ degrees of freedom.

```
theta_hat = cbind(a_hat, b_hat, sigma2_hat)
head(theta_hat)
```

```
        a_hat     b_hat sigma2_hat
[1,] 3.083447 0.9633365 0.08005781
[2,] 2.961624 0.9513831 0.14486206
[3,] 3.089259 1.0196278 0.10457618
[4,] 2.946431 0.9882319 0.11813399
```

```
[5,] 2.905273 0.9494119 0.09364506
[6,] 2.905965 0.9480337 0.09510966
```

```r
pairs(theta_hat, col = "darkgrey", pch = 19, cex = 1.2,
      labels = c(expression(widehat(a)),
                 expression(widehat(b)),
                 expression(widehat(sigma^2))))
```

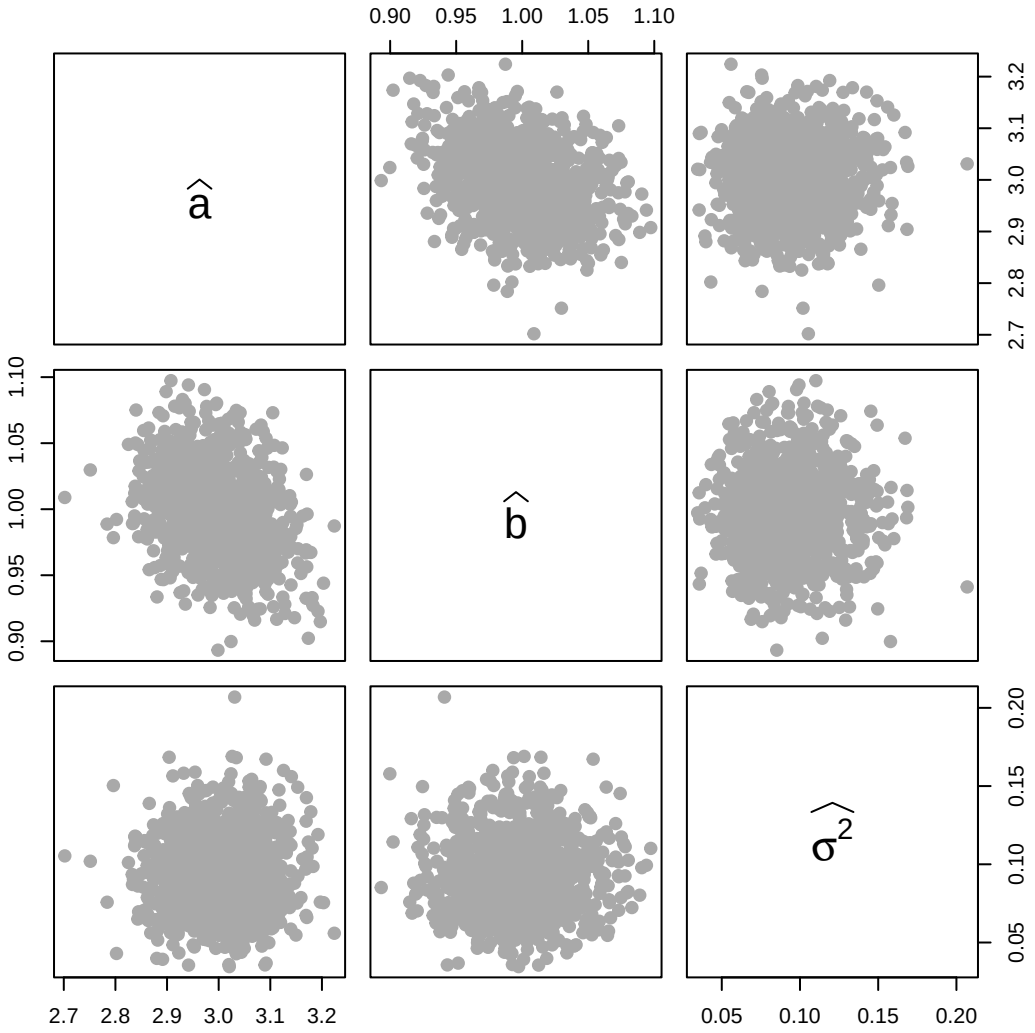


Figure 18: The pairs plot indicates a small negative correlations between $\hat{a}$ and $\hat{b}$, whereas, these estimators are independent with $\widehat{\sigma^2}$

Sensitivity to the Initial Conditions

In the following, we visualize the error sum of squares as a function a and b. The surface clearly demonstrates the existence of multiple local minimum and depending upon the initial conditions different minimum will be achieved.

```
fun_ESS = function(a,b){
  sum(y - sin(a*x) - b*x^2)^2
}
a_vals = seq(0,8, by = 0.01)
b_vals = seq(0,2, by = 0.01)
ESS_vals = matrix(data = NA, nrow = length(a_vals),
                  ncol = length(b_vals))
for(i in 1:length(a_vals)){
  for(j in 1:length(b_vals)){
    ESS_vals[i,j] = fun_ESS(a_vals[i], b_vals[j])
  }
}
library(plot3D)
persp3D(a_vals, b_vals, ESS_vals, xlab = "a", ylab = "b",
        zlab = "ESS", theta = 60, phi = 10)
```

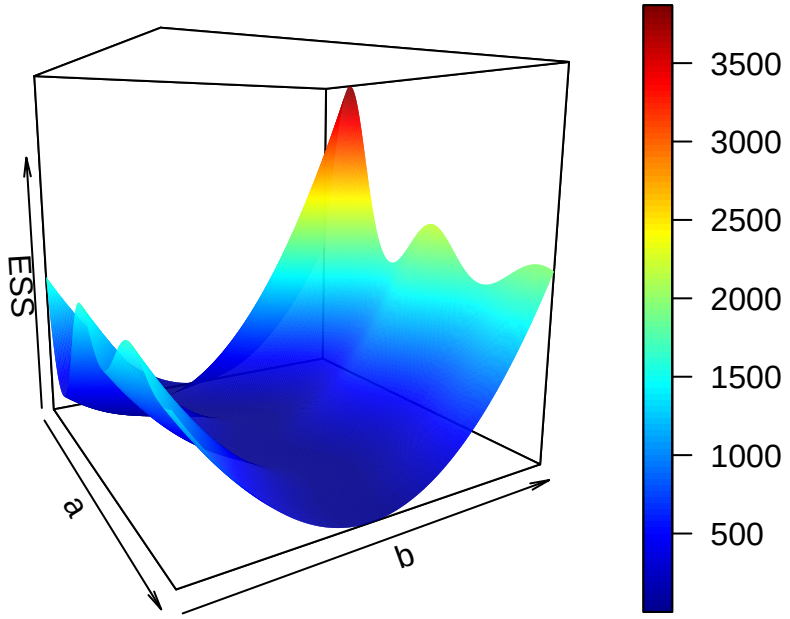


Figure 19: The surface of the negative log likelihood function for a specific choice of σ^2. Here we plot the error sum of squares, however, one must note that under the normality assumption, the surface is proportional to the negative log-likelihood function.

Large Sample Approximations

More ideas on consistent estimator

We say that a sequence of estimators W_n is a consistent estimator for $\theta \in \Theta$ if for every $\epsilon > 0$, and for every $\theta \in \Theta$, the following holds

$$\lim_{n\to\infty} \mathrm{P}_\theta \left(|W_n - \theta| < \epsilon\right) = 1.$$

The statement can also be equivalently written as $\lim_{n\to\infty} \mathrm{P}_\theta \left(|W_n - \theta| \geq \epsilon\right) = 0$ and by the Chebychev's inequality, we have

$$\mathrm{P}_\theta \left(|W_n - \theta| \geq \epsilon\right) \leq \frac{\mathrm{E}_\theta (W_n - \theta)^2}{\epsilon^2}.$$

Therefore, a sufficient condition for an estimator W_n to be consistent estimator for θ is to test whether $\mathrm{E}_\theta (W_n - \theta)^2 \to 0$ as $n \to \infty$ for every $\theta \in \Theta$.

In addition, we have the following well known decomposition, given by

$$\mathrm{E}_\theta (W_n - \theta)^2 = \mathrm{Var}_\theta (W_n) + \left(\mathrm{Bias}_\theta W_n\right)^2.$$

Therefore, we have the following theorem:

> **!** Characterization of consistent estimator [@casella2002statistical]
>
> If W_n is a sequence of estimators of a parameter θ satisfying the following conditions:
>
> - $\lim_{n\to\infty} Var_\theta (W_n) = 0$
> - $\lim_{n\to\infty} \mathrm{Bias}_\theta (W_n) = 0$
>
> for every $\theta \in \Theta$, then W_n is a sequence of consistent estimators of θ.

It is important to note that the sequence of estimators must have finite variance which is not a necessary condition to be consistent. One can construct a different sequence of consistent estimators as well by virtue of the following theorem:

! Many consistent estimators

If W_n is a consistent sequence of estimators of a parameter θ. Let (a_n) and (b_n) are sequence of real numbers satisfying $\lim_{n\to\infty} a_n = 1$ and $\lim_{n\to\infty} b_n = 0$. Then the sequence $U_n = a_n W_n + b_n$ is a consistent sequence of estimators of θ.

The proof is simple and reader is encouraged to show this. Compute $Var_\theta U_n$ and $\text{Bias}_\theta U_n$ and show that these quantities converges to 0 as $n \to \infty$.

Examples of consistent estimator using R

Using the software `R`, we can demonstrate that

$$\text{P}\left(\left|\overline{X_n} - \beta\right| < \epsilon\right) \to 0$$

as $n \to \infty$ for every choice of $\epsilon > 0$ and for all $\beta \in (0, \infty)$. In the following code, we compute the above probability which is basically the integration

$$\int_{n(\beta-\epsilon)}^{n(\beta+\epsilon)} \frac{e^{-\frac{y}{\beta}} y^{n-1}}{\Gamma(n)\beta^n} I_{(0,\infty)}(y) dy.$$

We use the function `pgamma()` to compute the probabilities $\text{P}(Y \leq n(\beta \mp \epsilon))$ where $Y \sim \mathcal{G}(\alpha = n, \beta)$.

```
n_vals = 1:5000
beta = 3
eps = 0.1
n = 10
prob_vals = numeric(length = length(n_vals))
for(n in n_vals){
  prob_vals[n] = pgamma(n*(beta + eps), shape = n, rate = 1/beta) - pgamma(n*(beta - eps), sh
}
plot(n_vals, prob_vals, xlab = "sample size (n)",
     main = expression(P(abs(bar(X[n])-beta) < epsilon)),
     type = "l", col = "grey", lwd = 2, ylab = "Probability")
legend("bottomright", legend = bquote(epsilon == .(eps)),
       lwd = 2, cex =1.4, bty = "n")
```

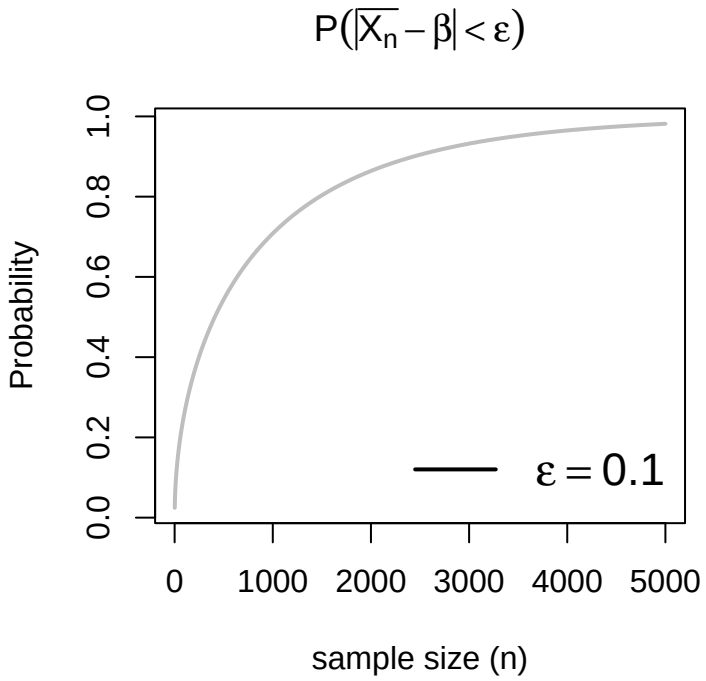


Figure 1: As the sample size increases, the probability converges to 1. Students are encouraged to experiment with different choices of β and ϵ.

The Python code for computing the probability is attached below. We use `gamma.cdf` function to calculate probabilities which works similar to `pgamma` in R.

Large Sample Approximation of Variance of Estimators

! Limiting Variance

For an estimator T_n, if

$$\lim_{n\to\infty} k_n Var(T_n) = \tau^2 < \infty,$$

where $\{k_n\}$ is a sequence of constants, then τ^2 is called the `limiting variance` or `limit of variances`.

```r
n_vals = c(3, 5, 10, 25, 50, 100)
mu = 2                      # true mean value
rep = 1000                  # number of replications
sigma = 0.5                 # population sd (given)
par(mfrow = c(2,3))
```

```r
sample_means = numeric(length = rep)
for(n in n_vals){
  t_n = numeric(length = rep)
  for(i in 1:rep){
    x = rnorm(n = n, mean = mu, sd = sigma)
    sample_means[i] = mean(x)
    t_n[i] = 1/mean(x)
  }

  hist(t_n, probability = TRUE, col = "lightgrey",
       xlab = expression(t[n]), main = paste("n = ", n),
       xlim = c(0,1))
  points(1/mu, 0, pch = 19, col = "red", cex = 1.4)
}
```

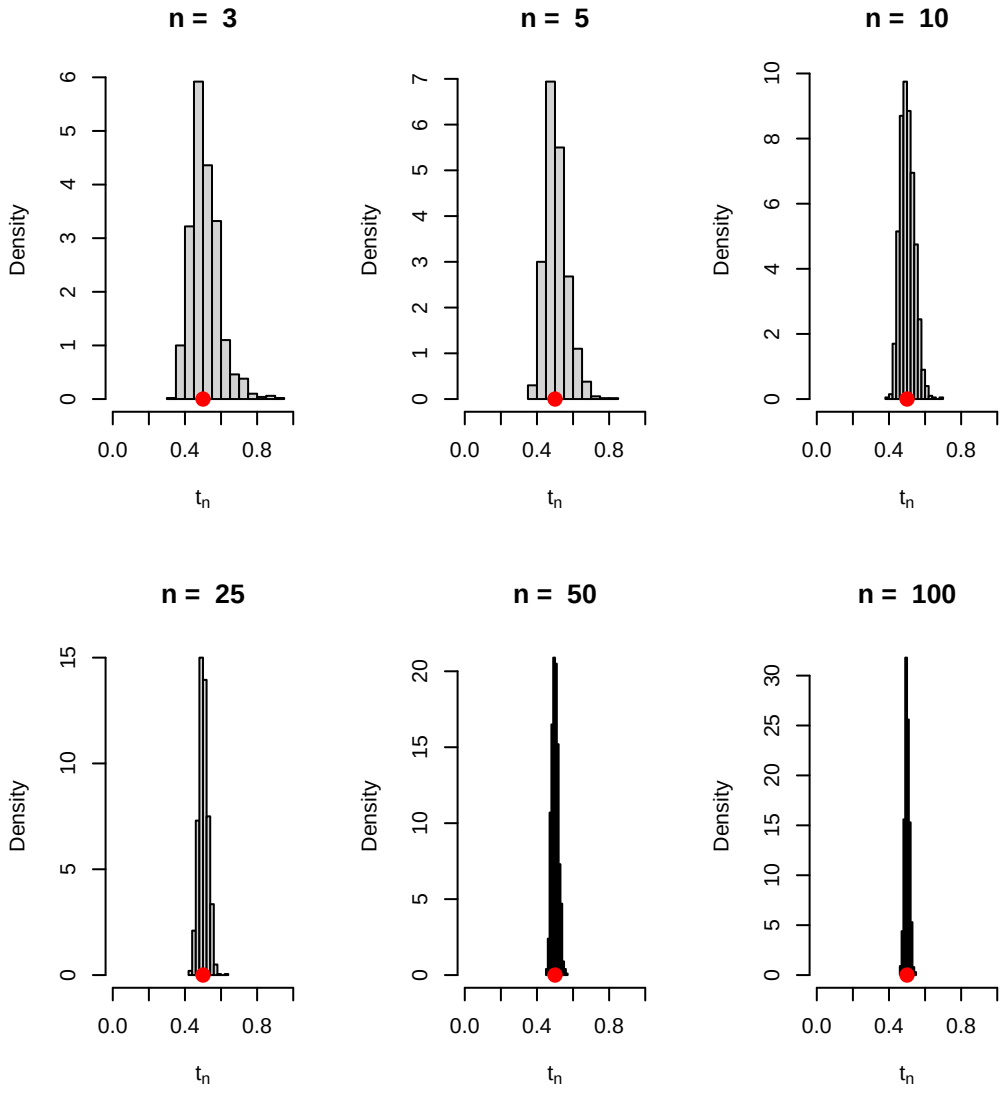


Figure 2: The sampling distribution of $1/\overline{X_n}$ is visualized using the histograms based on 1000 simulations for different sample size n. As the sample size increases, the sampling distribution gets highly concentrated about the value $1/\mu$.

The python code for visualizing the sampling distribution of $\overline{X_n}^{-1}$ is provided beow.

```
par(mfrow = c(1,2))
n_vals = 1:1000
t_n = numeric(length = length(n_vals))
sample_means = numeric(length = length(n_vals))
for(n in n_vals){
```

```
  x = rnorm(n = n, mean = mu, sd = sigma)
  sample_means[n] = mean(x)
  t_n[n] = 1/mean(x)
}
plot(n_vals, t_n, col = "grey", lwd = 2,
     xlab = "sample size (n)", main = expression(T[n]==1/bar(X[n])), ylab = "")
abline(h = 1/mu, col = "blue", lwd = 2, lty = 2)
plot(n_vals, sample_means, col = "grey", lwd = 2,
     xlab = "sample size (n)", main = expression(bar(X[n])), ylab = "")
abline(h = mu, col = "blue", lwd = 2, lty = 2)
```

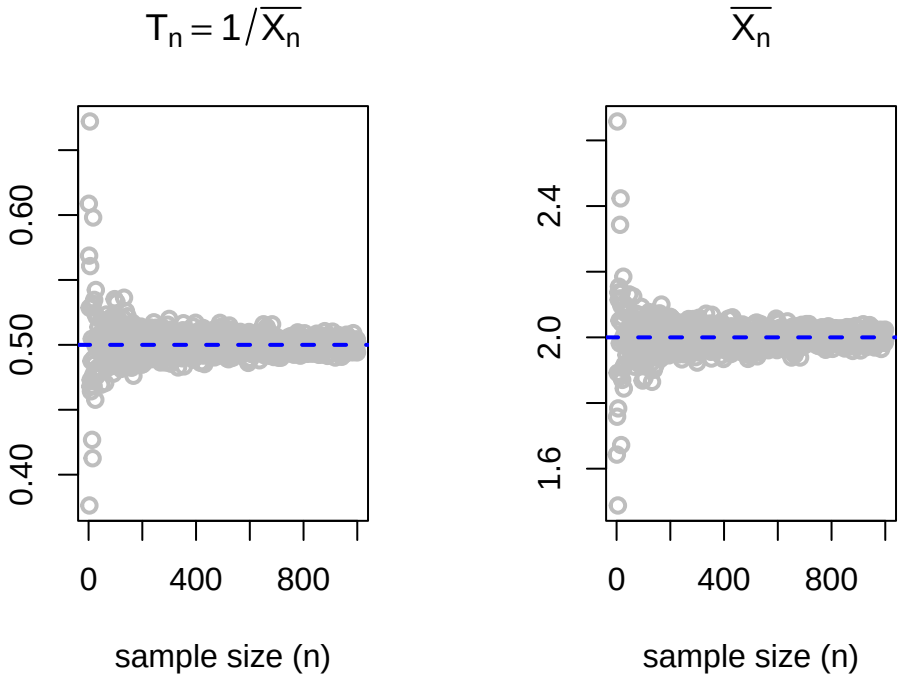


Figure 3: For large n, as $\overline{X_n}$ values becomes close to μ, then $1/\overline{X_n}$ values get closer to $1/\mu$ for $\mu \neq 0$. In fact it shows that as $n \to \infty$, $1/\overline{X_n} \to 1/\mu$.

Python implementation of the convergence of the $\overline{X_n}^{-1}$ to $\frac{1}{\mu}$ is given below.

Instead of the sample mean, one may also aim to estimate $1/\mu$ using the inverse of the sample median. For large n, the approximations may be compared if we can compute the limiting variances of the inverse of the sample median. Before, going into any mathematical computations, first let us check how the estimator based on the sample median behaves for large n values.

```r
par(mfrow = c(1,2))
n_vals = 1:1000
t_n = numeric(length = length(n_vals))
sample_medians = numeric(length = length(n_vals))
for(n in n_vals){
  x = rnorm(n = n, mean = mu, sd = sigma)
  sample_medians[n] = median(x)
  t_n[n] = 1/median(x)
}
plot(n_vals, t_n, col = "grey", lwd = 2,
     xlab = "sample size (n)", main = expression(T[n]==1/Med(X[n])))
abline(h = 1/mu, col = "blue", lwd = 2, lty = 2)
plot(n_vals, sample_medians, col = "grey", lwd = 2,
     xlab = "sample size (n)", main = expression(Med(X[n])))
abline(h = mu, col = "blue", lwd = 2, lty = 2)
```

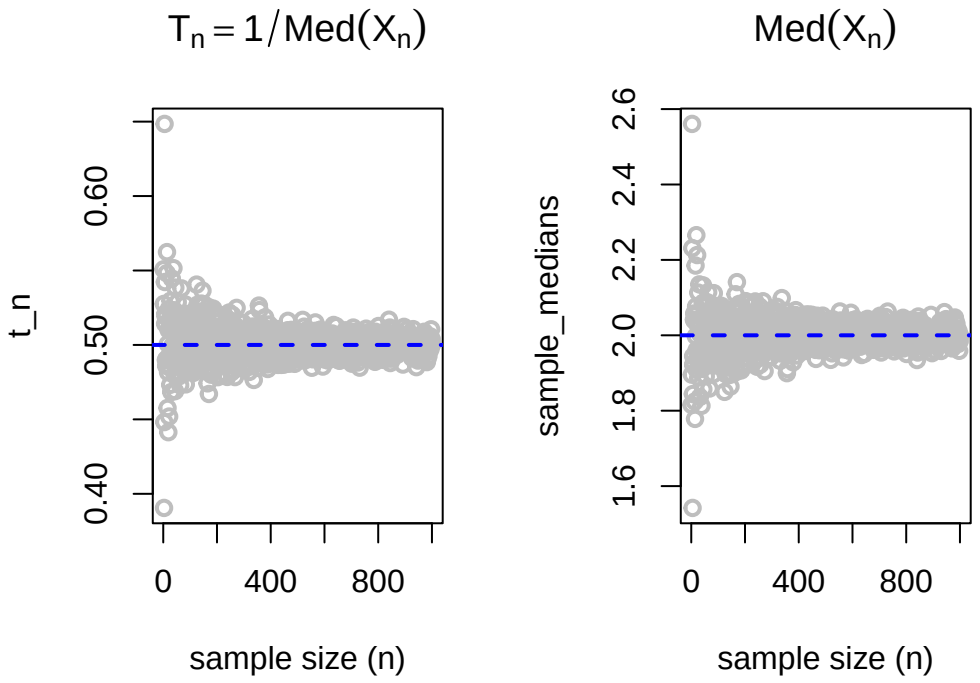
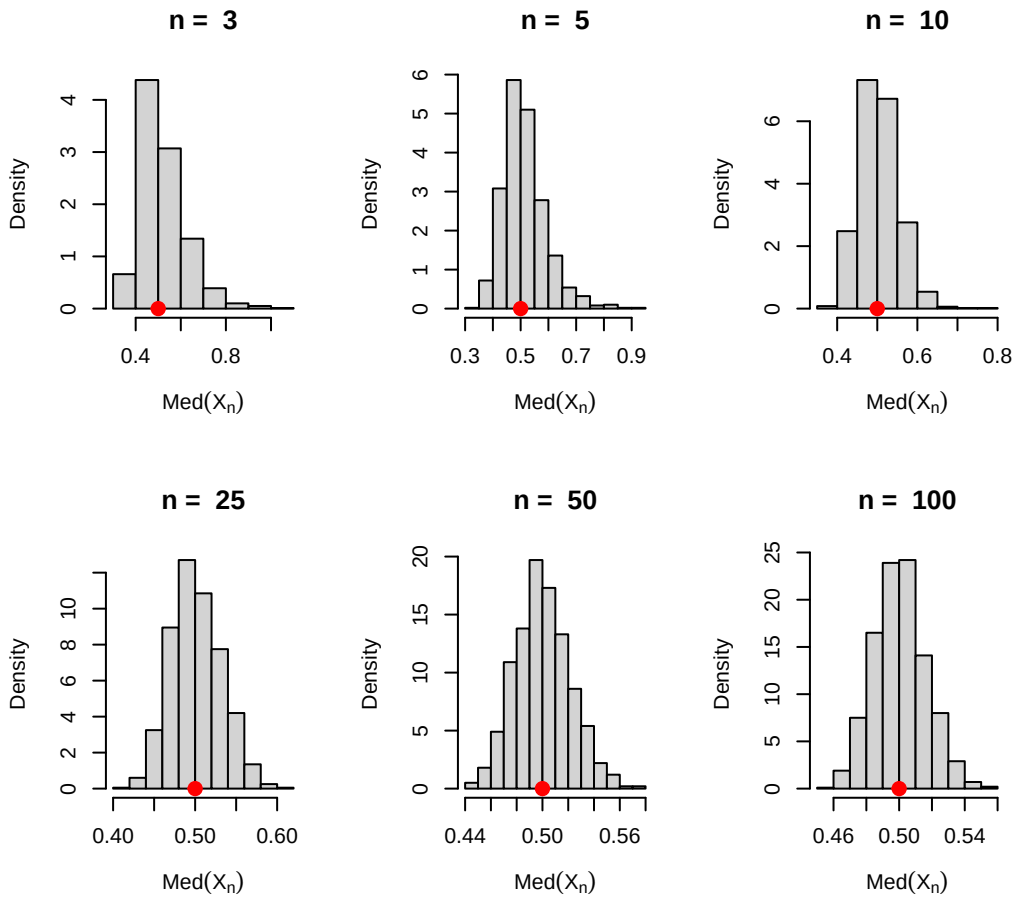


Figure 4: The inverse of the sample median also appears to be a consistent estimator for $1/\mu$.

The python implementation of the above code for the convergence of $\text{Med}(X_n)^{-1}$ to $\frac{1}{\mu}$ is demonstrated below.

Now let us check, how the sampling distribution of the estimator of $1/\mu$ based on the median, that is, $1/\text{Med}(X_n)$ behaves for large n values.

```r
n_vals = c(3, 5, 10, 25, 50, 100)
mu = 2
rep = 1000
sigma = 0.5
par(mfrow = c(2,3))
sample_medians = numeric(length = rep)
for(n in n_vals){
  t_n = numeric(length = rep)
  for(i in 1:rep){
    x = rnorm(n = n, mean = mu, sd = sigma)
    sample_medians[i] = median(x)
    t_n[i] = 1/median(x)
  }

  hist(t_n, probability = TRUE, col = "lightgrey",
       xlab = expression(Med(X[n])), main = paste("n = ", n))
  points(1/mu, 0, pch = 19, col = "red", cex = 1.4)
}
```



The python code for the sampling distribution of $\text{Med}(X_n)^{-1}$ for different choices of n is shown below:

For the above simulation experiments, it appears that both inverse of the sample mean and the sample median appears to be a nice choice and both are approximately normally distribution for large n. Let us now compare the inverse of the sample mean and sample median with respect to their asymptotic variances. We basically obtain the sampling distribution of the following two random variables for large n values.

$$\sqrt{n}\left(\frac{1}{\overline{X_n}} - \frac{1}{\mu}\right)$$

and

$$\sqrt{n}\left(\frac{1}{\text{Med}(X_n)} - \frac{1}{\mu}\right)$$

```r
n_vals = c(3, 5, 10, 25, 50, 100)
mu = 2
rep = 1000
sigma = 0.5
par(mfrow = c(2,3))
sample_means = numeric(length = rep)
sample_medians = numeric(length = rep)

for(n in n_vals){
  t_n = numeric(length = rep)    # inverse of sample mean
  w_n = numeric(length = rep)    # inverse of sample median
  for(i in 1:rep){
    x = rnorm(n = n, mean = mu, sd = sigma)

    t_n[i] = sqrt(n)*(1/mean(x) - 1/mu)
    w_n[i] = sqrt(n)*(1/median(x) - 1/mu)
  }

  plot(density(t_n), col = "red", lwd = 2, main = paste("n = ", n))
  lines(density(w_n), col = "blue", lwd = 2)
  legend("topright", legend = c(expression(bar(X[n])), expression(Med(X[n]))),
         col = c("red", "blue"), lwd = c(2,2), bty = "n")
  points(0, 0, pch = 19, col = "red", cex = 1.4)
}
```

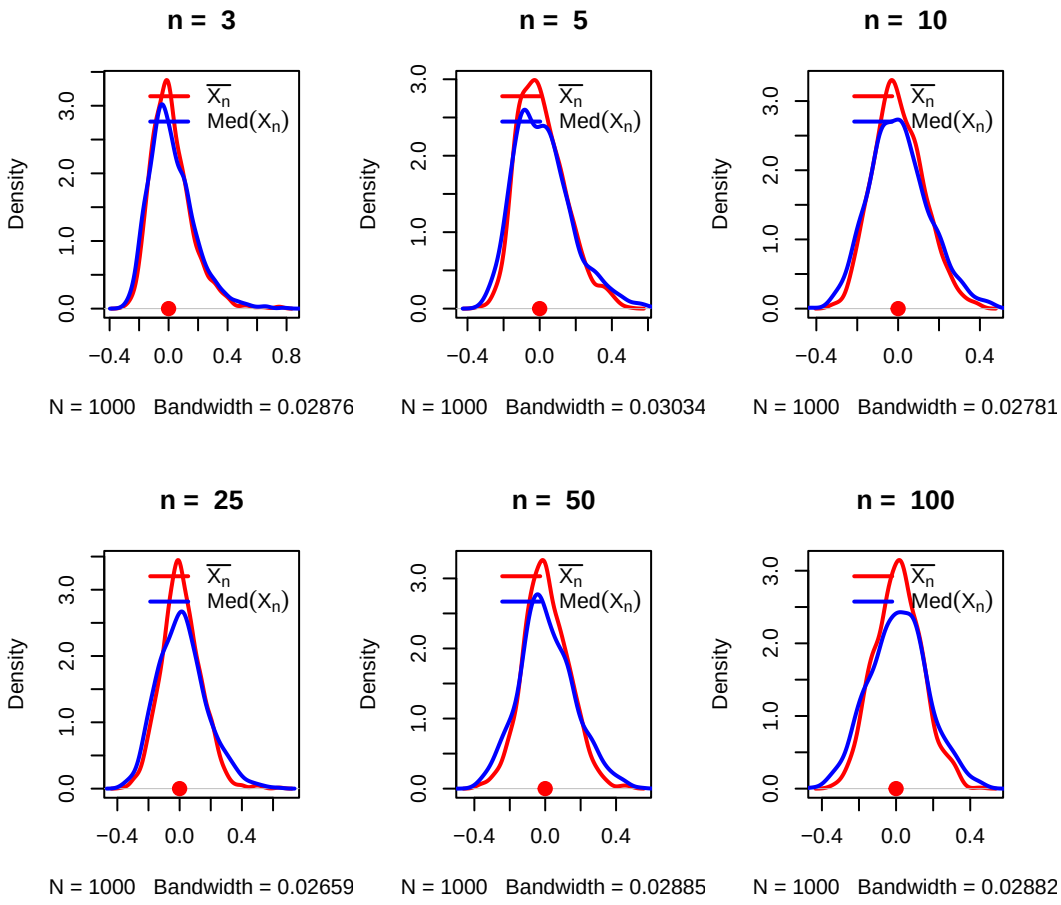


Figure 5: The simulation clearly demonstrates the comparison of the limiting variances of two estimators of $1/\mu$.

! Variance of the limit distribution of T_n

For an estimator T_n, suppose that

$$k_n\left(T_n - \tau(\theta)\right) \to \mathcal{N}(0, \sigma^2)$$

in distribution. The parameter σ^2 is called the asymptotic variance or variance of the limiting distribution of T_n.

The python code for the comparison of the approximate sampling distribution for the large sample size n is shown below:

In the above problem $T_n = \overline{X_n}^{-1}$ and

$$\sqrt{n}\left(T_n - \frac{1}{\mu}\right) \to \mathcal{N}\left(0, \frac{\sigma^2}{\mu^4}\right).$$

It is interesting to note that although the theoretical variance of $T_n = \overline{X_n}^{-1}$, $Var(T_n) = \infty$, but it has finite asymptotic variance $\frac{\sigma^2}{\mu^4}$ for $\mu \neq 0$, which is in fact more useful. The computation follows by a simple application of the Delta method, which gives $Var(T_n) \approx \frac{\sigma^2}{n\mu^4} < \infty$.

! Assymptotically efficient

A sequence of estimators of W_n is asymptotically efficient for a parameter $\tau(\theta)$ if

$$\sqrt{n}\left(W_n - \tau(\theta)\right) \to \mathcal{N}(0, v(\theta)),$$

where

$$v(\theta) = \frac{(\tau'(\theta))^2}{\mathrm{E}_\theta\left(\left(\frac{\partial}{\partial\theta}\log f(X|\theta)\right)^2\right)},$$

that is the asymptotic variance of W_n achieves the Cramer-Rao lower bound.

A natural question arises how to obtain an asymptotically efficient estimator, and we are lucky that the MLE is itself gives us algorithmic way of obtaining asymptotically efficient estimator. In the following section we discuss this in the light of an example.

MLE is asymptotically efficient

Suppose that $X_1, \ldots, X_n$ be a random sample of size n from the Poisson distribution with parameter λ. The Fisher Information is given by $I(\lambda) = \lambda^{-1}$. The MLE of the parameter λ is given by $\widehat{\lambda} = \overline{X_n}$. It can be easily shown by CLT that

$$\sqrt{n}\left(\overline{X_n} - \lambda\right) \to \mathcal{N}(0, \lambda),$$

in distribution, therefore, the asymptotic variance of $\overline{X_n}$ is λ, in fact, it is exact variance as well (why?). Let us perform some simulation experiment to see whether the claim is indeed true or not.

```
lambda = 3
rep = 1000
n = 10
w_n = numeric(length = rep)
for(i in 1:rep){
```

```
  x = rpois(n = n, lambda = lambda)
  w_n[i] = sqrt(n)*(mean(x) - lambda)
}
hist(w_n, probability = TRUE, col = "grey",
     main = paste("n = ", n), xlab = expression(W[n]))
curve(dnorm(x, mean = 0, sd = sqrt(lambda)), add = TRUE,
      col = "red", lwd= 2)
```

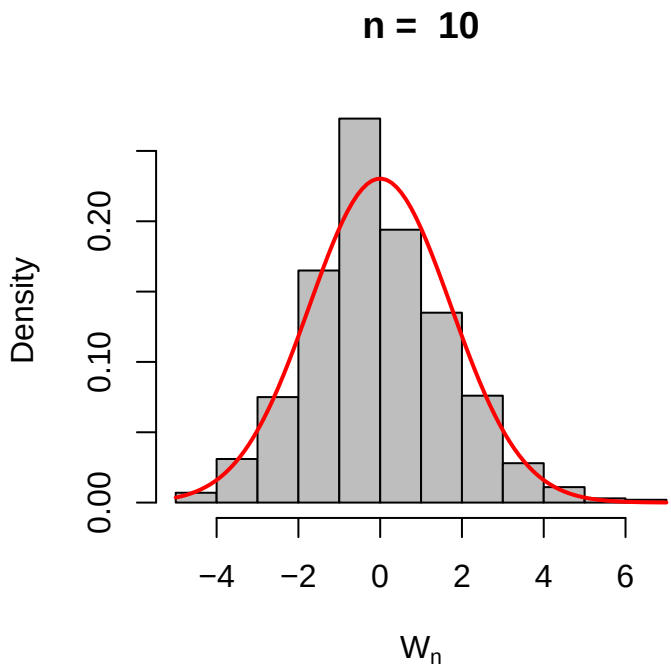


Figure 6: The experiment can be carried out for different choices of n. The overlaying of the normal distriubtion with the asymptotic variance agrees with the theoretical claim.

The above idea can be extended for estimating any continuous function of λ as well, say $h(\lambda)$. We start with a concrete example. Suppose, we are interested in estimating

$$h(\lambda) = \mathrm{P}(X = 2) = \frac{e^{-\lambda}\lambda^2}{2}.$$

Therefore, the estimator is given by

$$h(\hat{\lambda}) = e^{-\overline{X_n}} \left(\overline{X_n}^2\right)/2,$$

which is a highly nonlinear function of $\overline{X_n}$. The theory suggests that

$$\sqrt{n}\left(h(\hat{\lambda}) - h(\lambda)\right) \to \mathcal{N}(0, v(\lambda)),$$

in distribution where

$$v(\lambda) = \frac{(v'(\lambda))^2}{I(\lambda)} = \frac{\lambda^3 e^{-2\lambda}(2-\lambda)^2}{4}.$$

```
h = function(lambda){
  lambda^2*exp(-lambda)/2
}
lambda = 3
rep = 1000
n = 3
v_n = numeric(length = rep)
for(i in 1:rep){
  x = rpois(n = n, lambda = lambda)
  v_n[i] = sqrt(n)*(h(mean(x)) - h(lambda))
}
hist(v_n, probability = TRUE, col = "grey",
     main = paste("n = ", n), xlab = expression(v[n]))
curve(dnorm(x, mean = 0, sd = sqrt(lambda^3*exp(-2*lambda)*(2-lambda)^2/4)), add = TRUE,
      col = "red", lwd= 2)
```

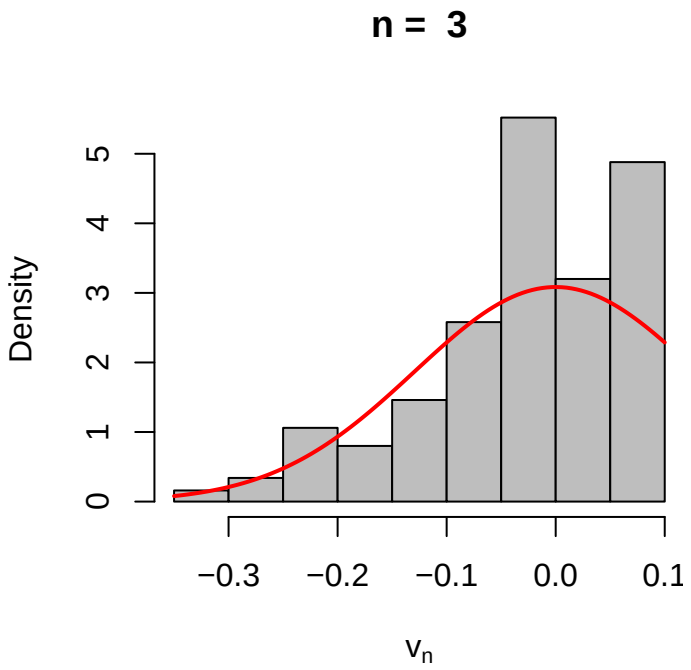


Figure 7: The sample size is small, therefore, the histogram is not a good approximation of the normal distribution. The reader is encouraged to do the simulation with different values of n.

```r
n_vals = c(5,10,25,50, 100, 500)
par(mfrow = c(2,3))
for(n in n_vals){
  v_n = numeric(length = rep)
  for(i in 1:rep){
    x = rpois(n = n, lambda = lambda)
    v_n[i] = sqrt(n)*(h(mean(x)) - h(lambda))
  }
  hist(v_n, probability = TRUE, col = "grey",
       main = paste("n = ", n), xlab = expression(v[n]))
  curve(dnorm(x, mean = 0, sd = sqrt(lambda^3*exp(-2*lambda)*(2-lambda)^2/4)),
        add = TRUE, col = "red", lwd= 2)
  print(var(v_n))

}
```

```
[1] 0.01143363
```

```
[1] 0.01377767
```

```
[1] 0.01556666
```

```
[1] 0.01512963
```

```
[1] 0.01624849
```

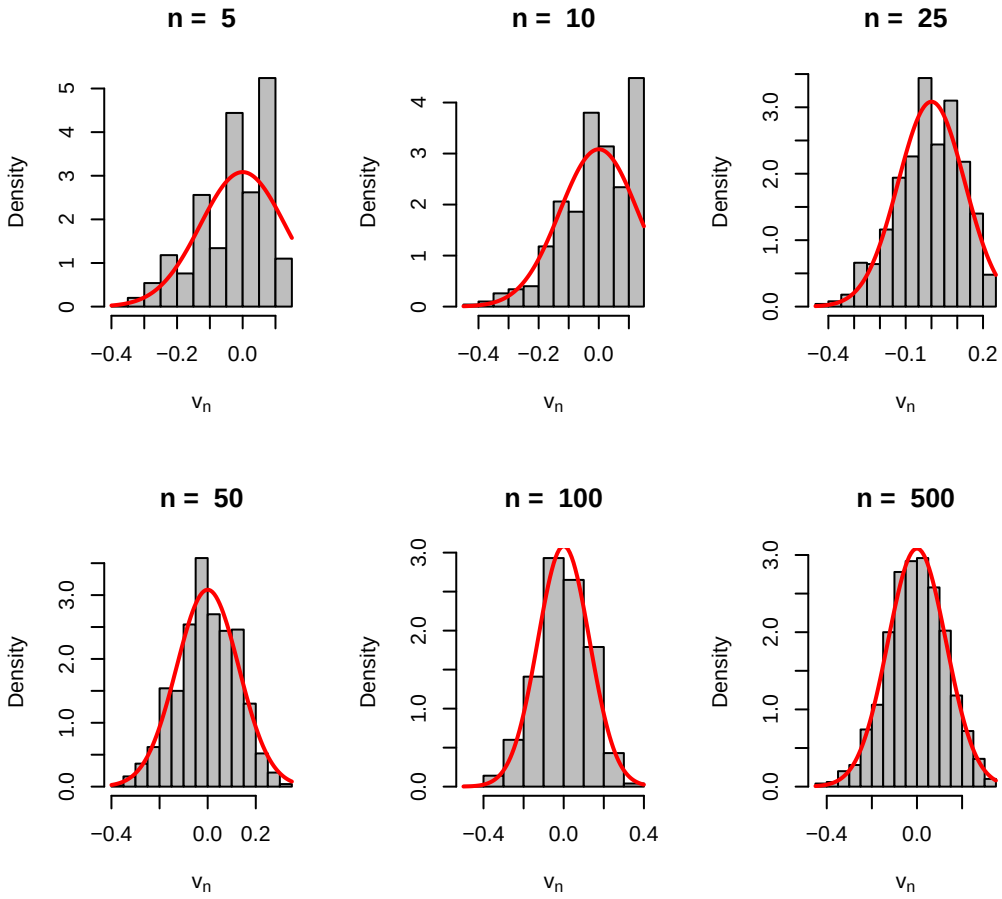


Figure 8: As the sample size increases, the approximation to the normal distribution is cleraly visible with the variance equal to the asymptotic variance.

```
[1] 0.01640191
```

In the following, we numerically (through simulation) verify that how accurate the approximation of the variance by plugging in the $\hat{\lambda}$ in place of λ.

$$
\begin{aligned}
Var\left(h\left(\hat{\lambda}\right)|\lambda\right) &\approx \frac{(h'(\lambda))^2}{I_n(\lambda)} && (0.1)\\
&= \frac{(h'(\lambda))^2}{\mathrm{E}_\lambda\left(-\frac{\partial^2}{\partial\lambda^2}\log\mathcal{L}(\theta|\mathbf{X})\right)} && (0.2)\\
&\approx \frac{\left[h'(\hat{\lambda})\right]^2}{-\frac{\partial^2}{\partial\lambda^2}\log\mathcal{L}(\theta|\mathbf{X})|_{\lambda=\widehat{\lambda}}}. && (0.3)
\end{aligned}
$$

In the above computation, two approximations have been carried out. In the first approximation the computation of the asymptotic variance has been carried out by the first order Taylor's approximation whereas in the second approximation, the expectation has been approximated by plugging in the MLE at the Fisher Information.

```
par(mfrow = c(1,1))
n_vals = 1:1000
asym_var = lambda^3*exp(-2*lambda)*(2-lambda)^2/4
var_v_n = numeric(length = length(n_vals))
for(n in n_vals){
  v_n = numeric(length = rep)
  for(i in 1:rep){
    x = rpois(n = n, lambda = lambda)
    v_n[i] = sqrt(n)*(h(mean(x)) - h(lambda))
  }
 var_v_n[n] = var(v_n)

}
plot(n_vals, var_v_n, type = "p", col = "grey",
     lwd= 2, xlab = "sample size (n)", ylab = "",
     main = expression(Var(h(hat(lambda[n])))))
abline(h = asym_var, lty = 2, col = "blue",
       lwd = 2)
```

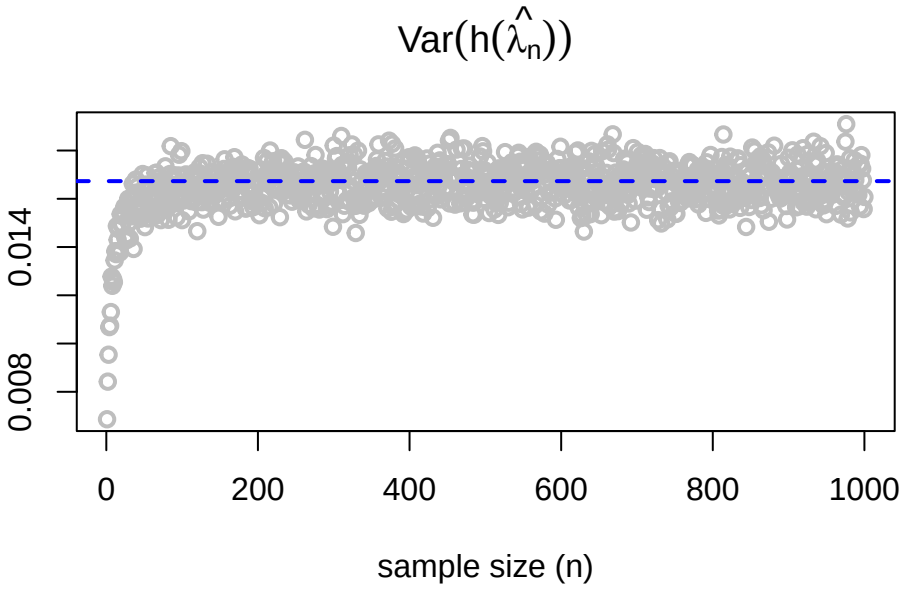


Figure 9: As the sample size increases, the approximated variance is close to the asymptotic variance. The asymptotic variance is shown using the dotted blue color line.

Based on the discussion about the optimal properties of the MLE, we have the following theorem:

! Asymptotic efficiency of MLE

Let $X_1, \ldots, X_n, \ldots$ be iid $f(x|\theta)$, let $\hat{\theta}$ denote the MLE of θ, and let $\tau(\theta)$ be a continuous function of θ. Under the regularity conditions on $f(x|\theta)$, and, hence on $\mathcal{L}(\theta|\mathbf{x})$, the likelihood function,

$$\sqrt{n}\left(\tau(\hat{\theta}) - \tau(\theta)\right) \to \mathcal{N}(0, v(\theta)),$$

where $v(\theta)$ is the Cramer-Rao Lowe Bound. That is, $\tau(\hat{\theta})$ is a consistent and asymptotically efficient estimator of $\tau(\theta)$.

Statistical Model for Contaminated data

Suppose that we have a random sample of size n from the normal distribution with mean μ and variance σ^2. However, there is a contamination with some values from the other distribution as well.

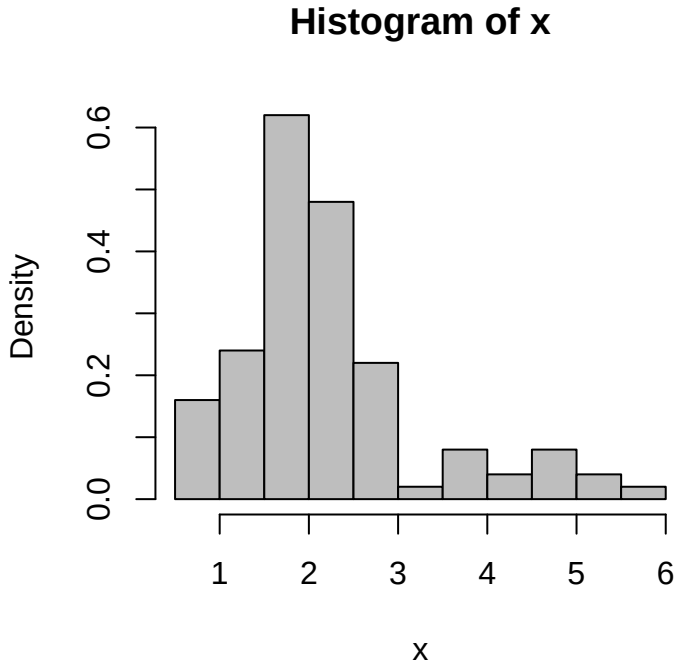
Consider the statistical model for the data with contamination as

$$X \sim \begin{cases} \mathcal{N}(\mu, \sigma^2), & \text{with probability } 1-\delta \\ f(x), & \text{with probability } \delta. \end{cases}$$

In the following we simulate a random sample of size n from the distribution with $100\delta\%$ contamination.

```
mu = 2
sigma2 = 0.5

theta = 5
tau2 = 0.5

delta = 0.1
n = 100
x = numeric(length = n)
for(i in 1:n){
  if(rbinom(n = 1, size = 1, prob = 1-delta)==1)
    x[i] = rnorm(n = 1, mean = mu, sd = sqrt(sigma2))
  else
    x[i] = rnorm(n = 1, mean = theta, sd = sqrt(tau2))
```

```
}
hist(x, probability = TRUE, col = "grey")
```



Show that the mean and variance of $\overline{X_n}$ is given by

$$Var\left(\overline{X_n}\right) = (1-\delta)\frac{\sigma^2}{n} + \delta\frac{\tau^2}{n} + \frac{\delta(1-\delta)(\theta-\mu)^2}{n}.$$

If $\theta \approx \mu$ and $\sigma \approx \tau$, then $Var(\overline{X_n}) \approx \frac{\sigma^2}{n}$, that means it achieves nearly optimal efficiency. However, the choice of $f(x)$ plays a critical role. For example, if $f(x)$ is Cauchy distribution, then the variance becomes infinite. You are encouraged to do some simulation considering the Cauchy distribution and plot the sampling distribution of $\overline{X_n}$ for different choices of δ.

! Breakdown value

Let $X_{(1)} < X_{(2)} < \cdots < X_{(n)}$ be an ordered sample of size n, and let T_n be a statistic based on this sample. T_n has breakdown value $b, 0 \leq b \leq 1$, if for every $\epsilon > 0$,

$$\lim_{X_{(\{(1-b)n\})} \to \infty} T_n < \infty \qquad \text{and} \qquad \lim_{X_{(\{(1-(b+\epsilon))n\})} \to \infty} T_n = \infty$$

- The sample mean $\overline{X_n}$ has break down value 0.
- The sample median M_n has breakdown value $\frac{1}{2}$.

! Trimmed mean

(Casella and Berger 2002) An estimator that splits the difference between the mean and median in terms of sensitivity is the $\alpha - trimmed$ mean, $0 < \alpha < \frac{1}{2}$, defined as follows. $\overline{X_n}^{\alpha}$, the $\alpha-trimmed$ mean, is computed by deleting the αn smallest observations and the αn largest observations, and taking the arithmetic mean of the remaining observations. Show that if $T_n = \overline{X_n}^{\alpha}$, the α-trimmed mean of the sample, $0 < \alpha < \frac{1}{2}$, show that $0 < b < \frac{1}{2}$.

Asymptotic normality of the M_n

Suppose that $X_1, \ldots, X_n$ be a random sample of size n from the population density function $f(x)$ with CDF $F(x)$. Assume that the CDF is differentiable and median is μ, that is $F(\mu) = \frac{1}{2}$.

- **Step - I:** Verify that, if n is odd, then

$$\mathrm{P}\left(\sqrt{n}(M_n - \mu) \leq a\right) = \mathrm{P}\left(\frac{\sum Y_i - np_n}{\sqrt{np_n(1-p_n)}} \geq \frac{(n+1)/2 - np_n}{\sqrt{np_n(1-p_n)}}\right)$$

- **Step - II:** Show that as $n \to \infty$, $p_n \to p = F(\mu) = \frac{1}{2}$ and

$$\frac{(n+1)/2 - np_n}{\sqrt{np_n(1-p_n)}} \to -2aF'(\mu) = -2af(\mu).$$

- **Step - III:** It is clear from the statement

$$\mathrm{P}\left(\sqrt{n}(M_n - \mu) \leq a\right) \to \mathrm{P}(Z \geq -2af(\mu))$$

that $\sqrt{n}(M_n - \mu)$ is asymptotically normal with mean 0 and variance $\frac{1}{(2f(\mu))^2}$.

First let us understand the above result in terms of computer simulation and visualization. In the following we first perform the experiment with the sampling from the normally distributed population.

```
mu = 2
sigma2 = 1
f = function(x){
  dnorm(x, mean = mu, sd = sqrt(sigma2))
}

```

```r
par(mfrow = c(2,3))
n_vals = c(3,5,10,25,50, 100)
rep = 1000
for (n in n_vals) {
  M_n = numeric(length = rep)
  W_n = numeric(length = rep)
  for(i in 1:rep){
    x = rnorm(n = n, mean = mu, sd = sqrt(sigma2))
    M_n[i] = median(x)
    W_n[i] = sqrt(n)*(M_n[i] - mu)
  }
  hist(W_n, probability = TRUE, main = paste("n = ",n),
       xlab =expression(W[n]), breaks = 30)
  curve(dnorm(x, mean = 0, sd = sqrt(1/(4*f(mu)^2))),
        add = TRUE, col = "red", lwd = 2)
}
```

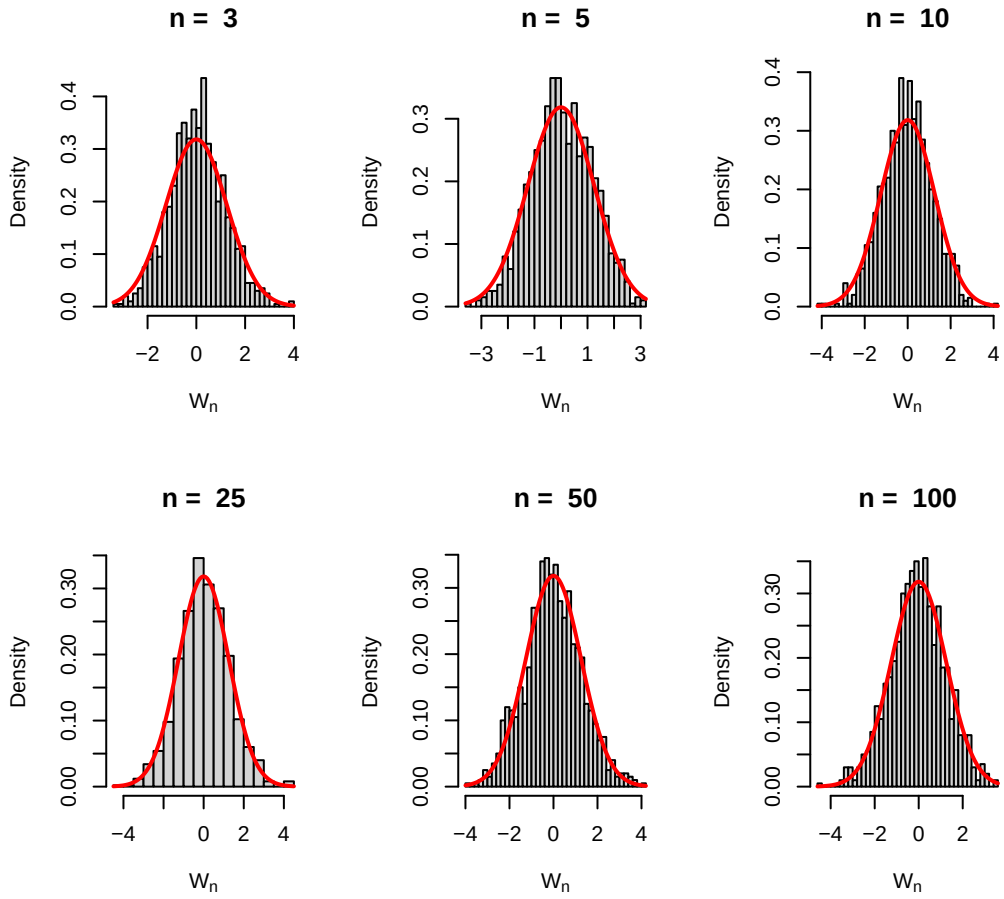


Figure 10: The sampling distribution of M_n is approximately normally distributed with asymptotic variance $1/(2f(\mu))^2$. For simulation $\mu = 2$ and $\sigma^2 = 1$ have been considered.

In the following, we perform the experiment with the exponential distribution with rate parameter λ. The median of the exponential distribution is given by $\mu = \frac{\ln 2}{\lambda}$. We simulate the distribution of $\sqrt{n}\left(M_n - \frac{\ln 2}{\lambda}\right)$ for different values of n and as $n \to \infty$, the normal approximation with the desired asymptotic variance is evident from the figures.

```
lambda = 2
mu = log(2)/lambda
f = function(x){
  dexp(x, rate = lambda)
}

par(mfrow = c(2,3))
n_vals = c(3,5,10,25,50, 100)
rep = 1000
```

```r
for (n in n_vals) {
  M_n = numeric(length = rep)
  W_n = numeric(length = rep)
  for(i in 1:rep){
    x = rexp(n = n, rate = lambda)
    M_n[i] = median(x)
    W_n[i] = sqrt(n)*(M_n[i] - mu)
  }
  hist(W_n, probability = TRUE, main = paste("n = ",n),
       xlab =expression(W[n]), breaks = 30)
  curve(dnorm(x, mean = 0, sd = sqrt(1/(4*f(mu)^2))),
        add = TRUE, col = "red", lwd = 2)
}
```

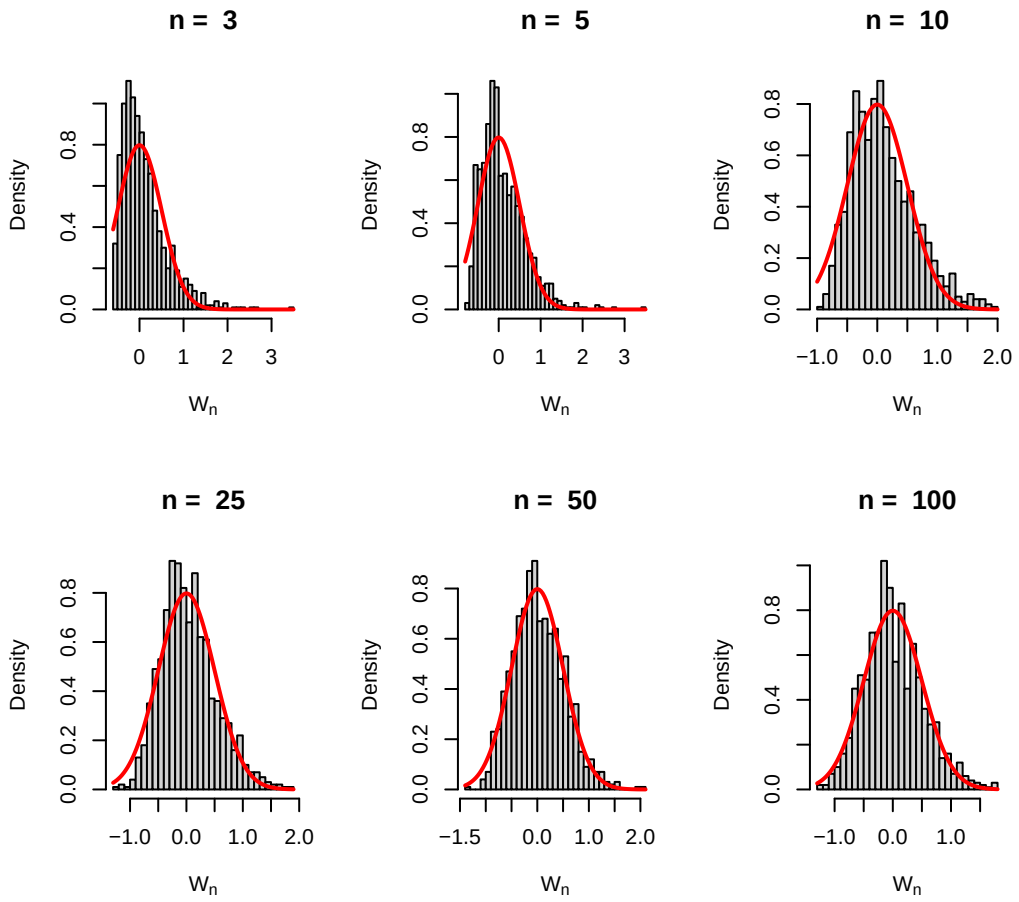


Figure 11: The simulation has been carried from the exponential dsitribution with rate parameter $\lambda = 2$, therefore, the true median is $\mu = 0.3465736$.

Based on the above discussion, the students are encouraged to explore the ARE of the median to the mean, that is $\text{ARE}\left(M_n, \overline{X_n}\right)$.

- Show that $\text{ARE}\left(M_n, \overline{X_n}\right)$ is unaffected by scale changes. This, it does not matter whether the underlying pdf if $f(x)$ or $\frac{1}{\sigma} f\left(\frac{x}{\sigma}\right)$.
- Compute the $\text{ARE}\left(M_n, \overline{X_n}\right)$ when the underlying distribution is Students' t with ν degrees of freedom, for $\nu \in \{3, 5, 10, 25, 50, \infty\}$. What is your conclusion about the ARE and the tails of the distribution?
- Compute the $\text{ARE}\left(M_n, \overline{X_n}\right)$ when the underlying pdf is the Tukey model

$$X \sim \begin{cases} \mathcal{N}(0,1) & \text{with probability } 1-\delta \\ \mathcal{N}(0,\sigma^2) & \text{with probability } \delta \end{cases}.$$

  Compute the ARE for a range of values of δ and σ. What can you conclude about the relative performance of the mean and the median?

Exercises

- Suppose that we are interested in estimating the location parameter μ for the normal, logistic and double exponential distribution based on a sample of size n. Obtain $\text{ARE}\left(M_n, \overline{X_n}\right)$ for each of the following distribution and discuss their comparative performance.
- Suppose that $X_1, X_2, \ldots, X_n$ be a random sample of size n from a population distribution characterized by the following probability density function

$$f(x) = \begin{cases} \frac{\theta}{x^2}, & \theta < x < \infty \\ 0, & \text{otherwise,} \end{cases}$$

  where $\theta \in \Theta = (0, \infty)$ and we are interested in estimating the parameter θ. Obtain the maximum likelihood estimator of θ, call it $\widehat{\theta_n}$. Check whether $\widehat{\theta_n}$ is an unbiased estimator of θ. If not compute $\text{Bias}_\theta\left(\widehat{\theta_n}\right)$ and check whether the estimator is asymptotically unbiased. Compute the mean squared error of the estimator $\text{MSE}_\theta\left(\widehat{\theta_n}\right)$. Comment whether the estimator is asymptotically consistent or not.
- Suppose that $X_1, \ldots, X_n$ be a random sample of size n from a population following $\mathcal{N}(\theta, \theta)$ distribution, where $\theta > 0$.
  - Show that the MLE of θ, $\widehat{\theta_n}$ is a root of the quadratic equation $\theta^2 + \theta - W = 0$, where $W = \frac{1}{n}\sum_{i=1}^{n} X_i^2$, and determine which root is equals the MLE.
  - Find approximate variance of $\widehat{\theta_n}$ using Delta method.

- Suppose that $Y_1, Y_2, \ldots, Y_n$ be a random sample of size n which satisfies the following equation

$$Y_i = \beta X_i + \epsilon_i, \ i = 1, 2, \ldots, n,$$

where $X_1, \ldots, X_n$ are independent $\mathcal{N}(\mu, \tau^2)$ random variables and $\epsilon_1, \ldots, \epsilon_n$ are IID $\mathcal{N}(0, \sigma^2)$, and X's and ϵ's are independent. In terms of μ, τ^2 and σ^2, find the approximate means and variances for the following quantities:

- $\sum X_i Y_i / \sum X_i^2$
- $\sum Y_i / \sum X_i$
- $\frac{1}{n} \sum (Y_i / X_i)$

Write a simulation program in R to check the sampling distributions of the above quantities and experiment with different choices of μ, τ^2 and σ^2. Comment on your findings.

- Consider the following hierarchical model

$$\begin{aligned} Y_n | W_n = w_n &\sim \mathcal{N}\left(0, w_n + (1 - w_n)\sigma_n^2\right) && (0.4) \\ W_n &\sim \text{Bernoulli}(p_n). && (0.5) \end{aligned}$$

Show that for Y_n, the limiting variance and the asymptotic variance differ.

  - Show that $\mathrm{E}(Y_n) = 0$ and $Var(Y_n) = p_n + (1 - p_n)\sigma_n^2$.
  - Show that $\mathrm{P}(Y_n) \to \mathrm{P}(Z < a)$, where $Z \sim \mathcal{N}(0, 1)$ for any $a \in \mathbb{R}$, and hence $Y_n \to \mathcal{N}(0, 1)$ in distribution.

- Suppose that $X_1, \ldots, X_n$ be iid observations from the normal distribution with mean $\mathrm{E}X = \mu$ and $Var(X) = \sigma^2$. Show that for $T_n = \frac{\sqrt{n}}{\overline{X}_n}$:

  - $Var(T_n) = \infty$.
  - If $\mu \neq 0$ and we delete the interval $(-\delta, \delta)$ from the sample space, then $Var(T_n) < \infty$.
  - If $\mu \neq 0$, then the probability content of the interval $(-\delta, \delta)$ approaches to 0 as $n \to \infty$.

- Suppose that $X_1, \ldots, X_n$ be a random sample of size n from the Poisson(λ) distribution and we are interested in estimating the 0 probability. For example, the number of customers that come into a bank in a given time period is sometimes modeled as a Poisson random variable, and the 0 probability is the probability that no one enter the bank in one time period. If $X \sim \text{Poisson}(\lambda)$, then $\mathrm{P}(X = 0) = e^{-\lambda}$.

  - Consider the estimator of $\tau = e^{-\lambda}$ as $\hat{\tau} = \frac{1}{n} \sum_{i=1}^{n} Y_i$, where $Y_i = I(X_i = 0)$. Show that $\mathrm{E}(\hat{\tau}) = e^{-\lambda}$ and $Var(\hat{\tau}) = \frac{e^{-\lambda}(1 - e^{-\lambda})}{n}$.
  - Consider another estimator of τ as $\tilde{\tau} = e^{-\hat{\lambda}}$, where $\hat{\lambda}$ is the MLE of λ. Obtain the asymptotic variance of $\tilde{\tau}$.

- Demonstrate using computer simulations that the following claims holds true (a) $\sqrt{n}(\hat{\tau} - e^{-\lambda}) \to \mathcal{N}(0, e^{-\lambda}(1 - e^{-\lambda}))$ and $\sqrt{n}\left(\tilde{\tau} - e^{-\lambda}\right) \to \mathcal{N}\left(0, \lambda e^{-2\lambda}\right)$. Also report $\text{ARE}\left(\hat{\tau}, \tilde{\tau}\right)$ and plot it as a function of λ and comment about your preferred estimator.

- (Casella and Berger 2002) Suppose that $X_1, X_2, \ldots, X_n$ are iid Poisson(λ). Find the best unbiased estimator of
  - $e^{-\lambda}$, the probability $X = 0$
  - $\lambda e^{-\lambda}$, the probability $X = 1$.
  - For the best unbiased estimators in part (a) and (b), calculate the asymptotic relative efficiency with respect to the MLE. Which estimators do you prefer and why?
  - A preliminary test of a possible carcinogenic compound can be performed by measuring the mutation rate of microorganisms exposed to the compound. An experimenter places the compound in 15 petri dishes and record the following number of mutant colonies:

    $$10, 7, 8, 13, 8, 9, 5, 7, 6, 8, 3, 6, 6, 3, 5.$$

    Estimate $e^{-\lambda}$, the probability that no mutant colonies emerge, and $\lambda e^{-\lambda}$, the probability that one mutant colony will emerge. Calculate both the best unbiased estimator of and the MLE.

- (Casella and Berger 2002) The performance of the sample mean in estimating the population may be compromised if there is a correlation in the sampling. This can seriously affect the properties of the sample mean. Suppose we introduce correlation in the sample $X_1, \ldots, X_n \sim \mathcal{N}(\mu, \sigma^2)$, but X_is are no longer independent.
  - For a equicorrelated case, that is, $\text{Corr}(X_i, X_j) = \rho, i \neq j$, show that

    $$Var(\overline{X_n}) = \frac{\sigma^2}{n} + \frac{n-1}{n}\rho\sigma^2,$$

    so that $Var(\overline{X_n}) \not\to 0$ as $n \to \infty$.
  - If the X_i's are observed throught time (or distance), it is sometimes assumed that the correlation decreases with time (or distance), with one specific model being $\text{Corr}(X_i, X_j) = \rho^{|i-j|}$. Show that the variance is given by

    $$Var(\overline{X_n}) = \frac{\sigma^2}{n} + \frac{2\sigma^2}{n^2}\frac{\rho}{1-\rho}\left(n - \frac{1-\rho^n}{1-\rho}\right),$$

    so that $Var(\overline{X_n}) \to 0$ as $n \to \infty$.

- The breakdown performance of the mean and the median continues with their scale estimate counterparts. For example $X_1, \ldots, X_n$:
  - Show that the breakdown value of the sample variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X_n}\right)^2$ is 0.

– A robust alternative is the mean absolute deviation, or MAD, the median of $|X_1 - M|$, $|X_2 - M|$, ...,$|X_n - M|$, where M is the sample median. Show that this estimator has a breakdown value 50%.

Interdisciplinary Teaching of Statistical Data Science

In many data science workshops, the instructor usually provides readymade code that participants simply run, often without fully understanding how it works. Also, when a workshop has too many speakers or unrelated sessions, participants can find it hard to follow the overall flow.

This chapter describes a different kind of approach, designed especially for ecologists, more generally to the interdisciplinary audience. The focus here is on learning through live discussions, practical examples, and writing code together. All the codes shown in this chapter were written live during the workshop, not prepared in advance and the participants wrote the same code alongside the instructor in real time. The goal was to help participants think critically and connect statistics to their own research. The case studies in this chapter reflect that interactive approach, showing how hands on learning can make statistical ideas clearer and more useful for ecologists. The case studies in this chapter reflect that interactive approach, showing how hands-on learning can make statistical ideas clearer and more useful, not just for ecologists, but for a broader interdisciplinary audience working with real world data.

Statistical Distributions

Probability distributions play a fundamental role in real-life data analysis problems. However, their importance is often underestimated in many applications, leading students to blindly accept the outcomes provided by software. Understanding probability distributions equips us with the ability to grasp the inherent uncertainty in natural processes. Data collected from the field are subject to various types of randomness, and probability theory provides a framework for these random phenomena. In this manual, we shall understand various probability distributions using R without getting into the mathematical intricacies of these distributions.

To understand the distribution theory using R, we need to learn about the four letters in R: `r`, `d`, `p`, `q`. We start our discussion with the coin tossing experiments. Suppose we want to simulate a coin tossing experiment using R instead of performing that experiment physically. Suppose the probability of success is $p = 0.3$. If we toss the coin 100 times say, we should expect an approximately 30 many heads (intuition!). In the following, we do this experiment once and observe that how many heads have been obtained. Certainly, the assumption that

such experiments can be identically performed an infinitely many numbers of times remain intact. As an applied researcher, we should just keep these statements as universally true.

```
p = 0.3                                  # probability of head
n = 100                                  # number of throws
x = rbinom(n = n, size = 1, prob = p)    # coin tossing experiment
print(x)                                 # 1 = head, 0 = tail
sum(x)                                   # number of heads
```

You are required to check whether the sum is actually 30 or not. If it appeared to be 30, do this experiment once again, you are likely to get another number, however, that is expected to be close to 30. As an ecologist, each run of the above code may be considered as follows: I visited a site consecutively 100 days and, on each day, we record 1 if we observe a phenomenon of our interest, otherwise record 0. Although, it is highly superficial to consider that conditions of all the 100 are identical, but somewhere, we need to start the discussion also to get into a more complicated statistical thinking.

In the above code the quantity x contains a possible realization of coin tossing experiment (called Bernoulli trial). Out of 100 positions, each position can be filled with 2 ways (either 0 or 1). Therefore, there are a total 2^{100} possible realizations of the coin tossing experiment. Similarly, in observations on the phenomenon of your interest, out of 2^{100} possibilities, we have obtained a particular sequence of 0's and 1's, which we call as the field observations. The Bernoulli(p) distribution essentially acts as a probability model for the natural process from which we have collected the observations. Our interest may be to get an accurate estimate of the unknown probability p (with certain confidence!).

Statisticians often say to ecologists that if you provide more data then he/she could give you better estimates. This is not a magic. This simple coin tossing experiment itself may reveal this justification adequately. Suppose that we toss the coin n times and record the number of successes, then obtain the proportion of success as an estimate of p. We then increase the number n and see whether, the proportion of success becomes close to the true probability 0.3 using which the simulation experiment has been carried out. We need to have cautionary note here, in the field experiment we never (ever) know what the underlying truth, that is what is the exact value of probability of observing the phenomena. However, in a computer simulation, we know that the experiments have performed by setting $p = 0.3$. Therefore, we can verify whether the proportion of success gets closer to the true value as the sample size increases. The following example will demonstrate this fact:

```
p = 0.3                                          # true probability
n_vals = 1:1000                                  # sample size
p_hat = numeric(length = length(n_vals))         # sample proportion
for(n in n_vals){
  x = rbinom(n = n, size = 1, prob = p)          # simulate coin tossing
```

```
  p_hat[n] = sum(x)/n                              # proportion of success
}
plot(n_vals, p_hat, type = "l", xlab = "sample size (n)",
     ylab = expression(widehat(italic(p))), col = "grey",
     cex.lab = 1.3, lwd = 2)
abline(h = p, col = "blue", lwd = 2, lty = 2)
```

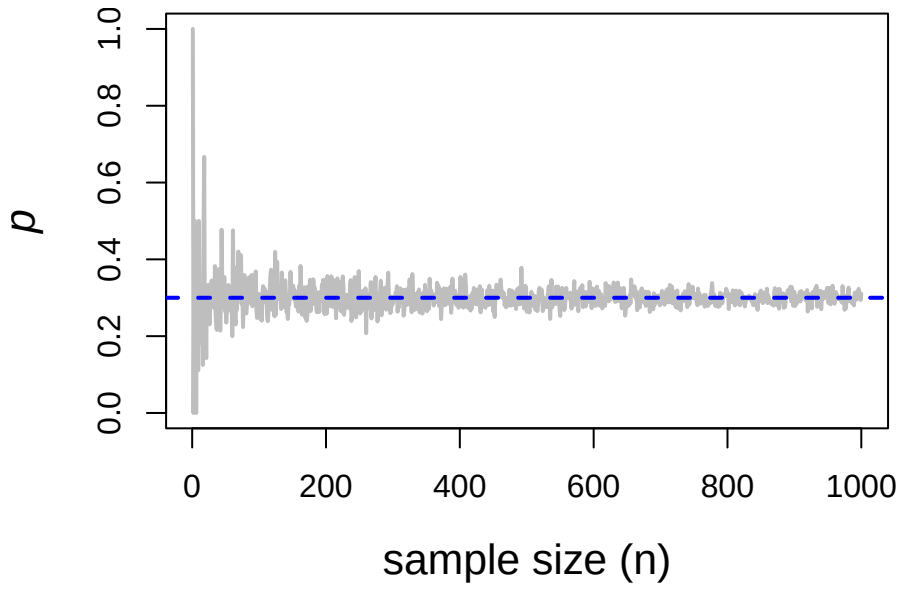


Figure 1: You will see the similar picture in your computer for the coin tossing experiment. It is visible that as the sampling size increases, the sample proportion of success are closer to the true probability 0.3.

In the above code snippet, we observed the use for loop, `numeric()` function, mathematical annotation using `expression()`, use of colour in plots, `abline()` function for horizontal line plot, etc. We have used to letter r to simulate random numbers from the Bernoulli(p) experiment. Similarly, we can simulate random numbers from the desired probability distributions. Now we discuss the favorite bell-shaped curve: the normal distribution which appears everywhere! The normal distribution is characterized by the two quantities mean μ and variance σ^2. The quantity μ can be any real number, where as the variance must be bigger than zero. First, we simulate n=100 random numbers from the normal distribution with parameters $\mu = 3$ and $\sigma^2 = 1$. Note that the square-root of the variance is the standard deviation (denoted as sd).

```
par(mfrow = c(1,3))                                # side by side plot
mu = 3                                             # mean
sigma = 1                                          # sd
```

```r
n = 100                                          # sample size
x = rnorm(n = n, mean = mu, sd =sigma)           # simulate numbers
hist(x, probability = TRUE, main = "", cex.lab = 1.4)
curve(dnorm(x, mean = mu, sd = sigma), lwd = 2,
      col = "red", add = TRUE)                   # add normal PDF
legend("topright", legend = "N(3,1)", lwd = 2,
       col = "red", bty = "n")                   # add legend

mu = 3                                           # mean
sigma = 2                                        # standard deviation
n = 100                                          # sample size
x = rnorm(n = n, mean = mu, sd = sigma)          # simulate observations
hist(x, probability = TRUE, main = "", cex.lab = 1.4)
curve(dnorm(x, mean = mu, sd = sigma), lwd = 2,
      col = "red", add = TRUE)
legend("topright", legend = "N(3,2)", lwd = 2,
       col = "red", bty = "n")

mu = 0
sigma = 1
n = 100
x = rnorm(n = n, mean = mu, sd = sigma)
hist(x, probability = TRUE, main = "", cex.lab = 1.4)
curve(dnorm(x, mean = mu, sd = sigma), lwd = 2,
      col = "red", add = TRUE)
legend("topright", legend = "N(0,1)", lwd = 2,
       col = "red", bty = "n")                   # add legend
```

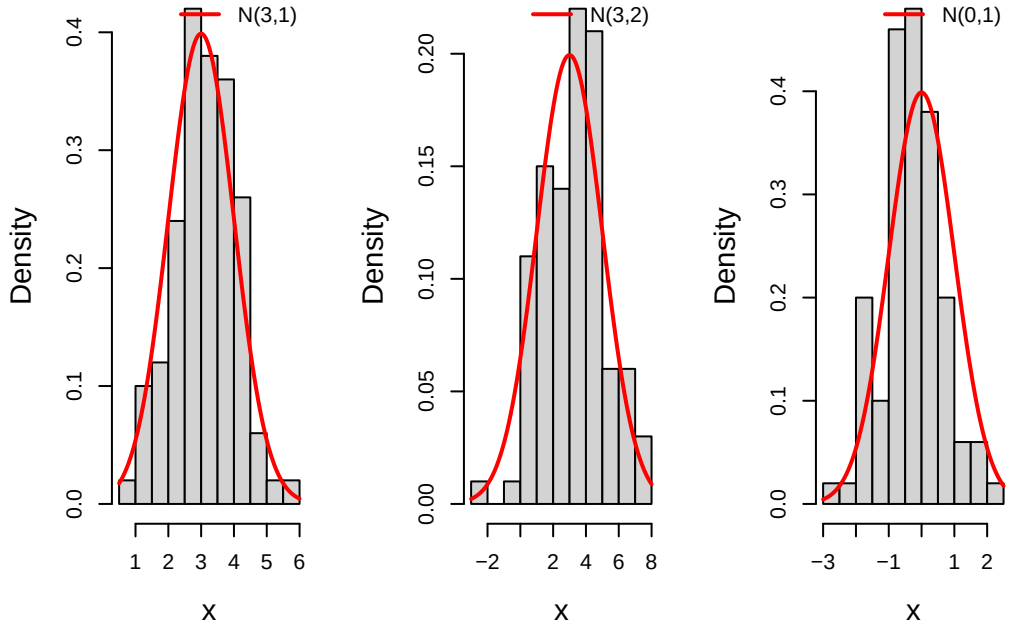


Figure 2: Simulation of normally distributed observations with fixed mean and variance specified by the user. The actual normal probability density function is overlaid on the same plot. The histograms are in close agreement with the theoretical probability density function.

You are encouraged to execute the following codes, to understand the role of letter `r` in simulation of random numbers in R.

```
n = 100                                          # sample size
x = rexp(n = n, rate = 1)                        # simulation of exponential rv
hist(x, probability = TRUE)                      # histogram
curve(dexp(x, rate = 1), add = TRUE)             # adding the exact PDF
help("rexp")                                     # get help in R

n = 100
x  = runif(n = n, min = 0, max = 2)              # simulation of uniform(0,2)
hist(x, probability = TRUE)
curve(dunif(x, min =0, max = 2), add = TRUE)
help("dunif")

n = 100
x = rgamma(n = n, shape = 3, rate = 2)           # simulation of Gamma(3,2)
hist(x, probability = TRUE)
curve(dgamma(x, shape = 3, rate = 2), add = TRUE)
```

```r
help("rgamma")
```

Poisson distribution

The Poisson distribution commonly appears in the ecological problems that deals with the count data. For example, ecologists studying the ant colonies often assume that the number of eggs laid by an individual ant follows a Poisson distribution with rate parameter $\lambda > 0$ (which is the parameter of interest). For the Poisson distribution, the mean and variance of the distribution are same (λ). Applications of Poisson distribution also appears in studying the distribution of rare plants in large forests (Krebs, Charles J. 1999. "Ecological Methodology." 2nd ed. Benjamin-Cummings). The following code can be used to simulate random numbers from the Poisson distribution with fixed rate parameter λ.

```r
par(mfrow = c(1,1))
n = 100                                                  # sample size
lambda = 4                                               # rate parameter
x = rpois(n = n, lambda = lambda)                        # simulate Poisson RV
print(x)
```

Using the function `dpois()`, we can draw the Poisson probability mass function. The following code will do this task.

```r
par(mfrow = c(1,2))                                      # side by side plot
x = 0:15
lambda = 2                                               # rate parameter
prob = dpois(x, lambda = lambda)                         # Poisson probability
plot(x, prob, type = "h", col = "grey", lwd = 2,
     cex.lab = 1.3, ylab = "P(X=x)",
     main = expression(paste(lambda, " = 2")))
points(x, prob, pch = 19, cex = 1.5, col = "blue")       # add points to existing plot

lambda = 5                                               # rate parameter
prob = dpois(x, lambda = lambda)                         # Poisson probability
plot(x, prob, type = "h", col = "grey", lwd = 2,
     cex.lab = 1.3, ylab = "P(X=x)",
     main = expression(paste(lambda, " = 5")))
points(x, prob, pch = 19, cex = 1.5, col = "blue")
```

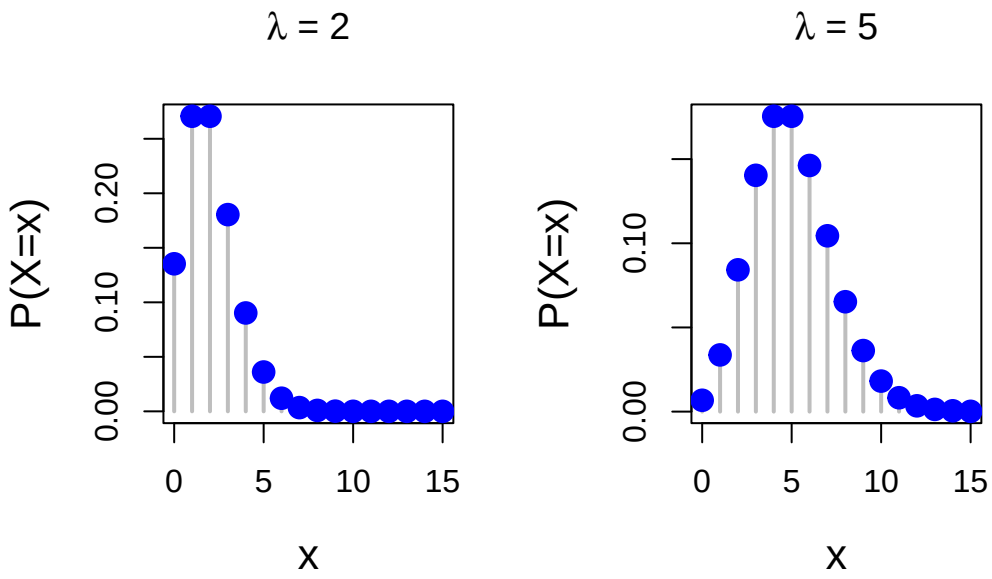


Figure 3: The Poisson distribution for different choices of λ. The function `dpois()` have been used to compute the exact probability values for the Poisson distribution at each value from the set $\{0,1,2,\ldots,15\}$.

Modelling and Simulation

Mathematical models are commonly used to model the growth process in natural sciences and these models often include a stochastic component to incorporate the uncertainty inherent in the natural processes. Regression models play important role in ecological data analysis. To demonstrate the next statistical model fitting exercises, we consider the following data set which is available in R. The following

```
data("trees")
head(trees)                          # first six observations
summary(trees)                       # summary of the data set
dim(trees)                           # data dimension
complete.cases(trees)                # checking for missing rows
tail(trees)                          # last six observations
class(trees)
names(trees)                         # names of the columns
head(trees, n = 10)                  # first ten observations
tail(trees)                          # last six observations
str(trees)                           # structure of the data frame
```

This data set provides measurements of the diameter, height and volume of timber in 31 felled black cherry trees. Note that the diameter (in inches) is erroneously labelled Girth in the data. The data set is available in datasets package in R. Before starting any analysis, it is very important to get an understanding of the data set with different visualization techniques.

```
par(mfrow=c(1,2))
data("trees")
summary(trees)                                  # data summary
```

```
     Girth           Height       Volume
 Min.   : 8.30   Min.   :63   Min.   :10.20
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40
 Median :12.90   Median :76   Median :24.20
 Mean   :13.25   Mean   :76   Mean   :30.17
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30
 Max.   :20.60   Max.   :87   Max.   :77.00
```

```
boxplot(trees, col = 2:4, main = "Box plot")
library(vioplot)                                # violin plot
vioplot(trees, col = 2:4, main = "Violin plot")
```

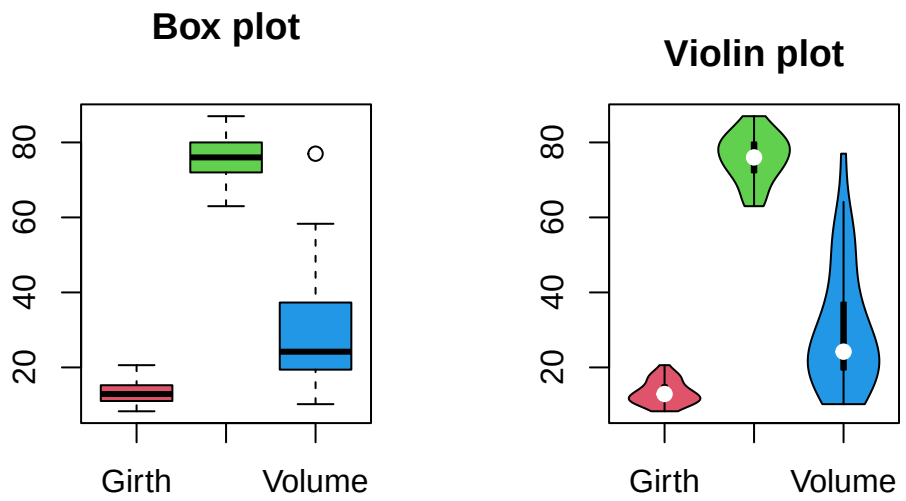


Figure 4: The left plot is the boxplot, and the right plot is the violin plot of the variables present in the trees data set. The volume variable is a positively skewed variable. The height of the plant is slightly negatively skewed.

Often histograms are also utilized for visualization of the data distribution of the observations. The following code can be used to draw the histograms. It is to be noted that the $ symbol is used to access the columns from the `data.frame` in R.

```r
par(mfrow = c(1,3))
hist(trees$Girth, probability = TRUE, main = "",
     xlab = "Girth", cex.lab = 1.3, col = 2)                    # histogram of Girth
hist(trees$Height, probability = TRUE, main = "",
     xlab = "Height", cex.lab = 1.3, col = 3)                   # histogram of height
hist(trees$Volume, probability = TRUE, main = "",
     xlab = "Volume", cex.lab = 1.3, col = 4)                   # histogram of volume
```

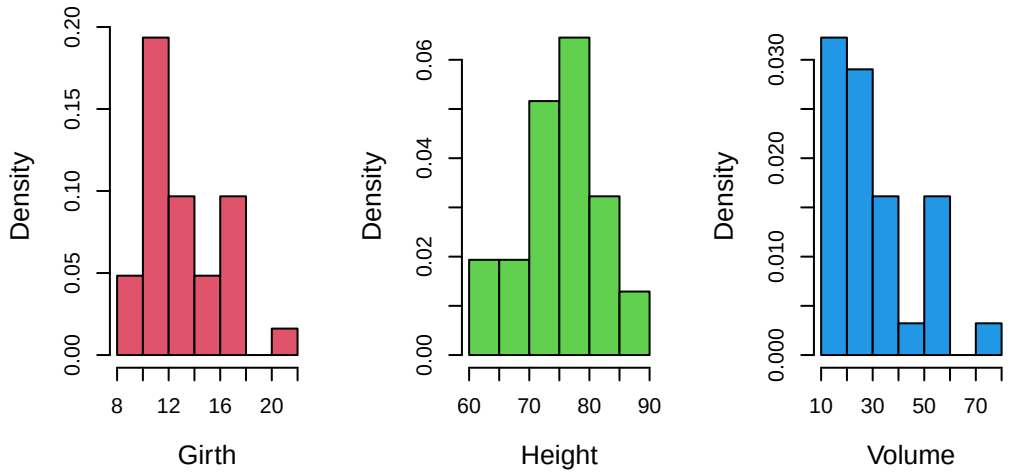


Figure 5: The histogram is based on the observations obtain from each tree and there is total 31 observations.

Exploring the relationship between different columns in the data set is often considered as the first step of modelling exercises. As a research statement, we may consider that whether the Girth and Height of a tree are a good predictor of the Volume. The `pairs()` function is a magical function in R that gives a great visual representation to explore the pairwise relationship of variables in the data.

```r
pairs(trees, col = "red", pch = 19)
```

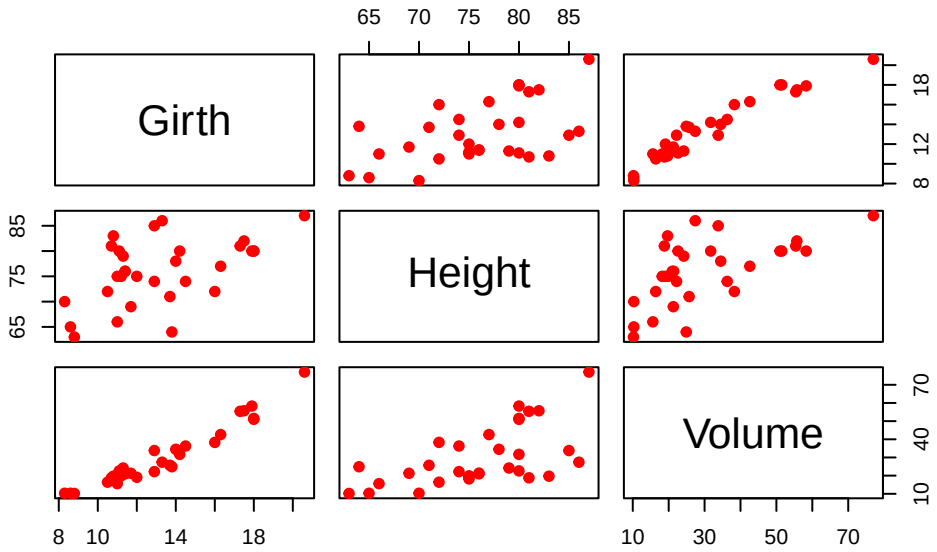


Figure 6: The data indicates that there is a linear relationship between the Girth and Volume. Between Height and Volume also it is there, however, it is not as strong as the Girth variable.

There are interesting packages in R that allows to explore the correlations between variables with great visualizations. The package corrplot allows great flexibility in presentation of correlation between variables. The following codes will help you to understand different varieties of representation of the correlation matrix.

```
par(mfrow = c(2,3))
library(corrplot)
corrplot(cor(trees))
cor(trees)
```

```
           Girth    Height    Volume
Girth  1.0000000 0.5192801 0.9671194
Height 0.5192801 1.0000000 0.5982497
Volume 0.9671194 0.5982497 1.0000000
```

```
#help("corrplot")
corrplot(cor(trees), method = "number")
corrplot(cor(trees), method = "number", type = "upper")
corrplot(cor(trees), method = "ellipse", type = "upper")
corrplot(cor(trees), method = "pie")
```

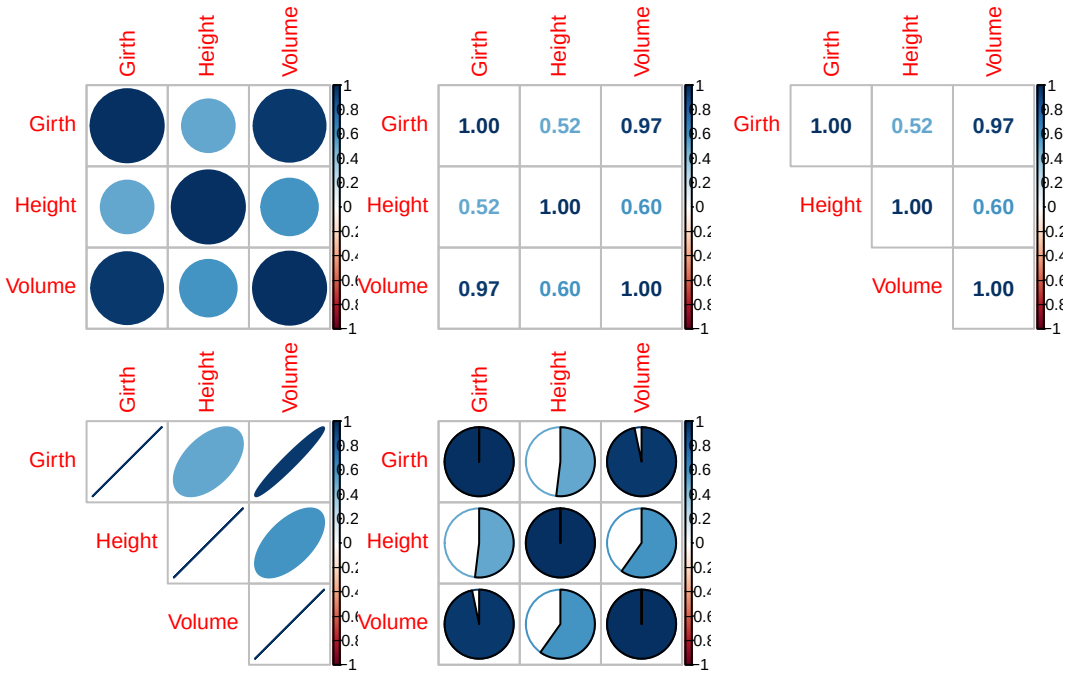


Figure 7: The corrplot package offers various graphical visualization to represent the correlations among the variables in the data set. The colour gradients are useful to represent the strong (positive or negative) correlation.

apply family of functions

The apply family of functions is an important way of computing various statistical characteristics of the data. Suppose that we are interested in computing the median of each column of the dataset. Individually, we can take each column of the data and compute it. However, the following code will help it to do it quickly.

```
apply(X = trees, MARGIN = 2, FUN = mean)          # column means
apply(X = trees, MARGIN = 2, FUN = median)        # column medians
apply(X = trees, MARGIN = 2, FUN = sd)            # column sd
```

In the apply function, `MARGIN = 2` corresponds to the columns, whereas `MARGIN = 1` corresponding to the rows.

In the first step of the analysis, suppose we aim to predict the Volume of the timber using the tree diameter. Therefore, we consider following statistical model

$$\text{Volume} = \beta_0 + \beta_1 \times \text{Girth} + \epsilon. \tag{0.1}$$

In the above expression, Volume is the response variable and Girth is the predictor variable. β_0 and β_1 are the intercept and the slope parameters of the population regression line, respectively. The ϵ represents the random component that cannot be explained through the linear relationship between the response and predictor. We also assume that $\epsilon \sim \mathcal{N}(0, \sigma^2)$. In the above equation, Girth is fixed and ϵ is random, which makes the Volume as a random quantity and follows normal distribution with mean $\beta_0 + \beta_1 \times$ Girth and variance σ^2.

It is very important to note that assumption of such relationships at the population level mostly driven by the visualization and existing knowledge pool about the underlying process. In this case, from the scatterplot, we have hypothesized that a linear relationship between the Volume and the Girth may be a reasonable assumption to consider. In principle, the true nature of the relationship is always unknown. With our scientific instruments, we can probably identify a most likely relationship from a set of reasonable hypotheses. However, in this case, our visualization is quite clear, and we go ahead with the assumption of linear relationships.

Our goal is to estimate the parameters β_0, β_1 and σ^2. There are 31 observations, and we write down the data model as

$$\text{Volume}_i = \beta_0 + \beta_1 \times \text{Girth}_i + \epsilon_i, i \in \{1, 2, \dots, n\}.$$

We seek to obtain the estimate of β_0 and β_1 that minimizes the error sum of squares:

$$\sum_{i=1}^{31} (\text{Volume}_i - \beta_0 - \beta_1 \times \text{Girth}_i)^2.$$

Using R, we can estimate the parameters of the interest. The following code will do this task for us. In this case, we have only one predictor and one response, therefore, it is called as a simple linear regression. The function `lm()` understand the formula `Volume ~ Girth` as an input and identifies which one is response and which one is predictors.

```
fit = lm(formula = Volume ~ Girth, data = trees)
coefficients(fit)                                    # estimated coefficients
```

```
(Intercept)       Girth
 -36.943459    5.065856
```

```
summary(fit)                                          # summary of the fitting
```

```
Call:
lm(formula = Volume ~ Girth, data = trees)
```

```
Residuals:
   Min     1Q Median     3Q    Max
-8.065 -3.107  0.152  3.495  9.587

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
Girth         5.0659     0.2474   20.48  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.252 on 29 degrees of freedom
Multiple R-squared:  0.9353,    Adjusted R-squared:  0.9331
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

The most important part in the output to understand the interpretation of the p-value which is represented as Pr(>|t|). The fitting exercise also perform the testing of two hypotheses

$$\mathrm{H}_0 : \beta_0 = 0 \text{ versus } \mathrm{H}_1 : \beta_0 \neq 0$$

$$H_0 : \beta_1 = 0 \text{ versus } H_1 : \beta_1 \neq 0$$

If reject the null hypothesis $\mathrm{H}_0 : \beta_1 = 0$ (at certain level of significance, say 5%), that means the tree diameter (Girth) has a good contribution in predicting the volume of timber.***indicates that the tree diameter is highly significant in determining the volume of timber. The same interpretation goes to the intercept as well. A multiple R-squared value of 0.9353 indicates that the approximately 94% of the variation in the Volume can be explained by the tree diameter through a linear function. In this case, this is quite satisfactory.

```
1 par(mfrow = c(1,2))
2 plot(Volume ~ Girth, data = trees, col = "red", pch = 19)
3 abline(fit, col = "blue", lwd = 2)
4
5 resid = residuals(fit)                                    # residuals of fit
6 hist(resid, probability = TRUE, xlab = "residuals",
7      main = "")
8 shapiro.test(resid)                                        # test for normality
```

```
    Shapiro-Wilk normality test

data:  resid
W = 0.97889, p-value = 0.7811
```

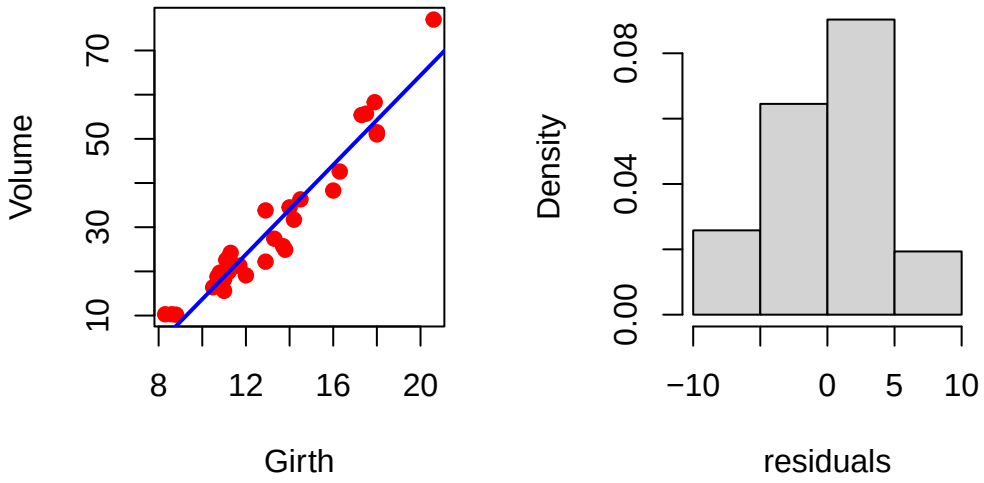


Figure 8: The fitting of the linear regression and the distribution of errors are shown using histograms. The red dots indicate the actual observations, and the blue line is the fitted line.

It is important to check whether the error is normally distributed. We have used the `shapiro.test()` to check for the normality. A large p-value indicates the acceptance of the null hypothesis that is the errors are normally distributed. After the fitting exercises, the regression diagnostics must be performed to check the validity of the assumptions that we have made at the starting of the discussion. Using the following piece of codes, we can get a visualization of the regression diagnostic tools.

```
par(mfrow = c(2,2))
plot(fit, pch = 19, col = "blue", lwd = 2)
```

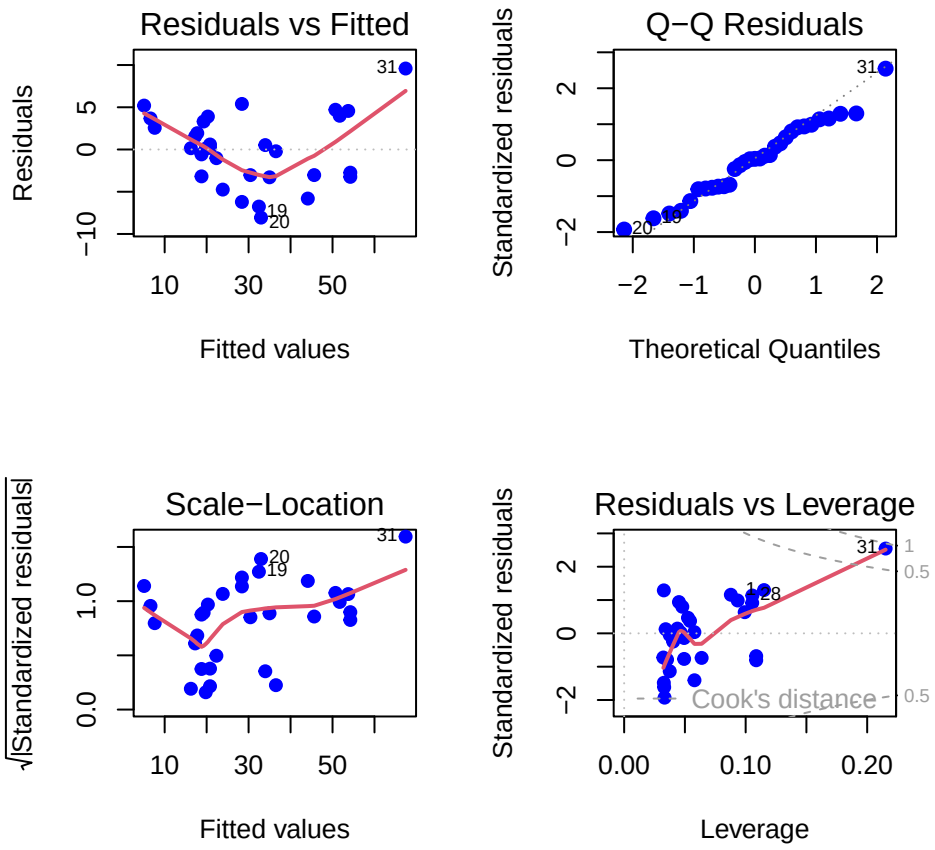


Figure 9: The plot in the top left corner gives indication of the existence of potential nonlinearity relationship between the response and the predictor. We expect the red line to be horizontally linear to have a better fitting. Such curvy relationship indicates that may be a polynomial expression for the predictor may be a better choice.

Although we have obtained the coefficient of determination (R^2) is approximately 94%, the residual versus fitted plot gives a slight indication of nonlinearity. Suppose we plan to make this model quadratic in the following way:

$$\text{Volume} = \beta_0 + \beta_1 \times \text{Girth} + \beta_2 \times \text{Girth}^2 + \epsilon. \tag{0.2}$$

Note that in this case, the model complexity increases (with additional parameter β_2 to be estimated). The other assumptions remain the same. The same function `lm()` in R can be utilized to estimate the parameters. We utilize the wrapper `I(.)` to include the polynomial expression in R.

```r
par(mfrow = c(1,2))
fit2 = lm(formula = Volume ~ Girth + I(Girth^2),
          data = trees)                              # quadratic fit
summary(fit2)                                          # fit summary
```

```
Call:
lm(formula = Volume ~ Girth + I(Girth^2), data = trees)

Residuals:
    Min      1Q  Median      3Q     Max
-5.4889 -2.4293 -0.3718  2.0764  7.6447

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.78627   11.22282   0.961 0.344728
Girth       -2.09214    1.64734  -1.270 0.214534
I(Girth^2)   0.25454    0.05817   4.376 0.000152 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.335 on 28 degrees of freedom
Multiple R-squared:  0.9616,    Adjusted R-squared:  0.9588
F-statistic: 350.5 on 2 and 28 DF,  p-value: < 2.2e-16
```

By considering quadratic regression model, the coefficient of determination has increased to 96% from 94%. However, the p-values suggest that the intercept and first order component do not have significant contribution in explaining the variation in Volume of timber through a quadratic regression model. It essentially says that the model is as good as having the expression as Volume$= \beta_2 \times \text{Girth} + \epsilon$.

```r
par(mfrow = c(1,2) )
plot(Volume ~ Girth, data = trees,
     col = "red", pch = 19)                            # original data
lines(trees$Girth, fitted.values(fit2),
      col = "blue", lwd = 2)                           # fitted curve
resid2 = residuals(fit2)                                 # residuals
hist(resid2, probability = TRUE, xlab = "residuals",
     main = "")                                            # residual distribution
shapiro.test(resid2)                                       # test for normality
```

```
    Shapiro-Wilk normality test
```

```
data:  resid2
W = 0.97393, p-value = 0.6327
```

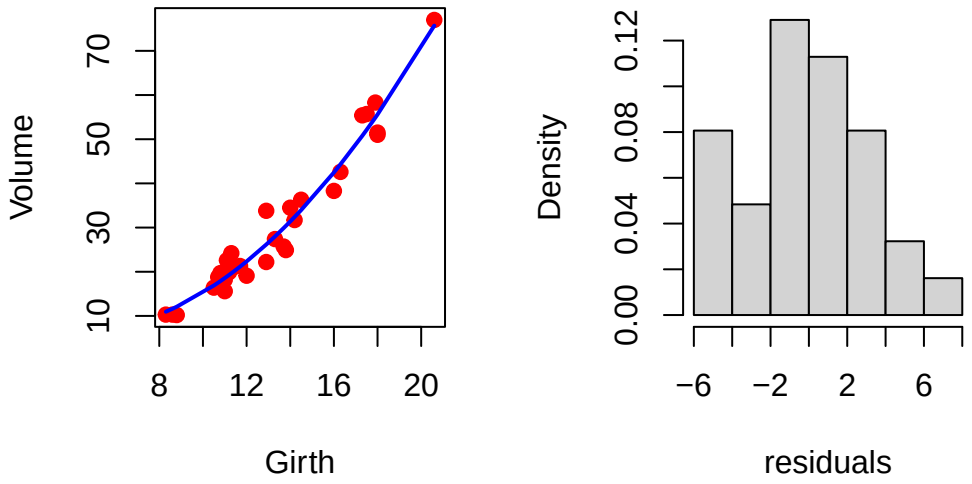


Figure 10: Fitting of the quadratic regression model. The fitted equation is Volume= $10.78627 - 2 \times \text{Girth} + 0.25454 \times \text{Girth}^2$. However, the parameters b_0 and b_1 are not statistically significant. The histogram of the error remains normally distributed.

```
1 par(mfrow = c(2,2))
2 plot(fit2, col = "blue", pch = 19, lwd = 2)
```

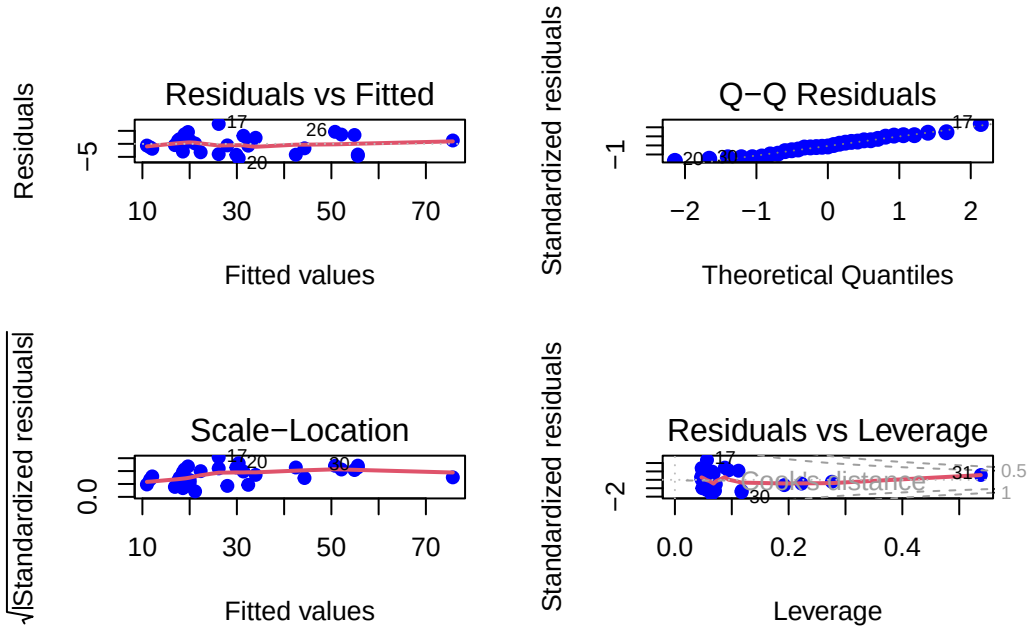


Figure 11: In the top left plot, we can see that the red horizontal line is approximately linear.

Now basically we have two statistical models. A natural question arises, whether making the mode complex, that is, considering quadratic from the linear regression, we have gained significantly or not. R provides a simple way of comparing two fitting exercises.

```
1 anova(fit, fit2)
```

```
Analysis of Variance Table

Model 1: Volume ~ Girth
Model 2: Volume ~ Girth + I(Girth^2)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     29 524.30
2     28 311.38  1    212.92 19.146 0.0001524 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis of the anova() function is that both the model performs equally well. However, from the obtained p-value of the output, the null hypothesis is rejected (very small p-value, ***).

Prediction and Confidence intervals

We go back to our original regression object `fit` (the simple linear regression) for discussing the difference between prediction and the confidence interval. Suppose we want to predict the volume of the timber for a new tree whose measurement of the diameter of the tree is provided and we plan to use the equation

$$\widehat{\text{Volume}} = \hat{\beta}_0 + \hat{\beta}_1 \times \text{Girth}.$$

It is an important task to report the uncertainty associated with the prediction. The variance associated with the prediction can be obtained by using the following formula:

$$Var(\widehat{\text{Volume}}) = Var(\hat{\beta}_0) + Var(\hat{\beta}_1) \times \text{Girth}^2 + 2 \times \text{Girth} \times Cov\left(\hat{\beta}_0, \hat{\beta}_1\right).$$

In the above, we have just applied the commonly known formula

$$Var(aX + bY) = a^2 Var(X) + b^2 Var(Y) + 2ab \times Cov(X, Y).$$

In the following we obtain the estimate of the Volume of timber as 59.30781 cubic ft for the tree having diameter of 19 inches with standard error of the estimate as 1.614811 cubic ft. Run the following code and understand the prediction interval is wider than the confidence interval.

```
new = data.frame(Girth = c(19))
predict(fit, newdata = new, se.fit = TRUE)
```

```
$fit
       1
59.30781

$se.fit
[1] 1.614811

$df
[1] 29

$residual.scale
[1] 4.251988
```

```r
predict(fit, newdata = new, interval = "prediction")
```

```
       fit     lwr      upr
1 59.30781 50.0055 68.61013
```

```r
predict(fit, newdata = new, interval = "confidence")
```

```
       fit      lwr      upr
1 59.30781 56.00515 62.61047
```

We first plot the prediction and the confidence intervals in the same plot and observe that the prediction intervals are wider than the confidence intervals. The reason is that the prediction intervals account for the randomness in the error component as well as the estimates in the systematic component. The confidence interval only considers the variation in $\hat{\beta}_0$ and $\hat{\beta}_1$, not the variance of $\hat{\epsilon}$. Therefore, for prediction interval the variance of the prediction is computed as:

$$\begin{aligned} Var(\widehat{\text{Volume}}) &= Var(\hat{\beta}_0 + \hat{\beta}_1 \times \text{Girth} + \hat{\epsilon}) && (0.3) \\ &= Var(\hat{\beta}_0) + Var(\hat{\beta}_1) \times \text{Girth}^2 + 2 \times \text{Girth} \times Cov\left(\hat{\beta}_0, \hat{\beta}_1\right) + Var(\hat{\epsilon}). && (0.4) \end{aligned}$$

The presence of an extra component $Var(\hat{\epsilon})$ is evident from the calculations.

```r
par(mfrow = c(1,1))
plot(Volume ~ Girth, data = trees, cex.lab = 1.2,
     col = "red", pch = 19, ylim = c(0, 80))
new = data.frame(Girth = seq(8, 22, by = 0.5))
pred_interval = predict(fit, newdata = new,
                        interval = "prediction")    # prediction
conf_interval = predict(fit, newdata = new,
                        interval = "confidence")    # confidence
lines(new$Girth, pred_interval[,2], col = "magenta",
      lwd = 2, lty = 2)
lines(new$Girth, pred_interval[,3], col = "magenta",
      lwd = 2, lty = 2)

lines(new$Girth, conf_interval[,2], col = "blue",
      lwd = 2, lty = 2)
lines(new$Girth, conf_interval[,3], col = "blue",
```

```
      lwd = 2, lty = 2)
legend("topleft", legend = c("Prediction Interval",
                             "Confidence Interval"),
       col = c("magenta", "blue"), lwd = c(2,2),
       lty = c(2,2), bty = "n")
```

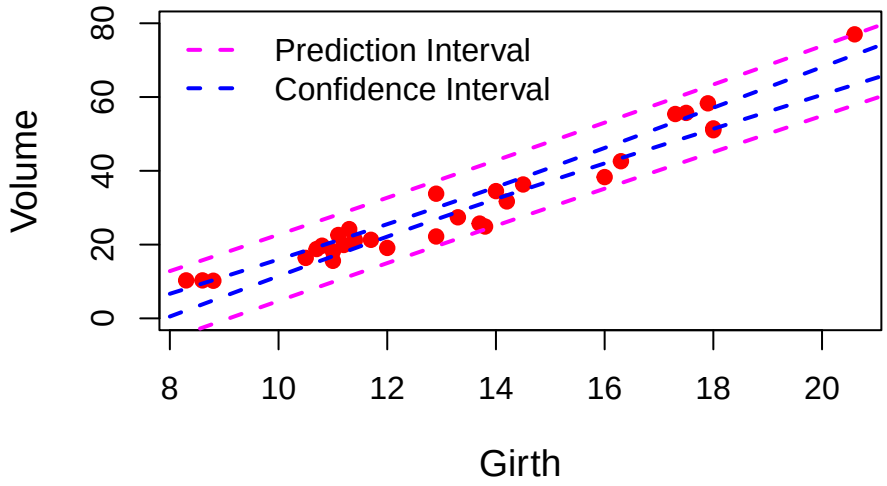


Figure 12: The prediction and the confidence intervals are plotted. The prediction intervals are wider than the confidence intervals (see text).

Incorporating other covariates

Till now we have discussed the regression model with a single predictor. However, there is another predictor Height (in ft) is also there. Suppose that we are now interested in including this covariate in modelling the Volume of timber using the following multiple linear regression model:

$$\text{Volume} = \beta_0 + \beta_1 \times \text{Girth} + \beta_2 \times \text{Height} + \epsilon$$

{$eq-trees_MLR}

Using the `lm()` function in R, we can perform the fitting exercises and subsequently perform the regression diagnostics. The formula `Volume ~ Girth + Height` is understood as a multiple linear regression model by the `lm()` function.

```
fit3 = lm(formula = Volume ~ Girth + Height, data = trees)
summary(fit3)                          # summary of MLR
```

```
Call:
lm(formula = Volume ~ Girth + Height, data = trees)

Residuals:
    Min      1Q  Median      3Q     Max
-6.4065 -2.6493 -0.2876  2.2003  8.4847

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
Girth         4.7082     0.2643  17.816  < 2e-16 ***
Height        0.3393     0.1302   2.607   0.0145 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared:  0.948, Adjusted R-squared:  0.9442
F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

Note that addition of another predictor gives the multiple R-squared value approximately 95%, that is, we have obtained approximately 1%. Using the `anova()` function, we can check whether this increment is significant or not.

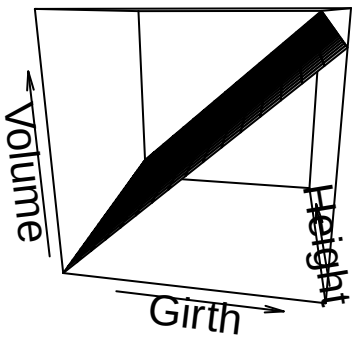
```
1 anova(fit, fit3)
```

```
Analysis of Variance Table

Model 1: Volume ~ Girth
Model 2: Volume ~ Girth + Height
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     29 524.30
2     28 421.92  1    102.38 6.7943 0.01449 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table suggests only a marginal improvement is obtained by adding an extra covariate in the model. Such decisions are very important in the analysis of field data and must be reported with proper explanation. Often it is cited as the Art of Data Analysis, rather than the Science of Data Analysis.

In this case, the fitted regression function will be a plane instead of a line. In the following, we write a small piece of codes, that will plot of the regression plane. The `persp()` function in R is useful to draw three dimensional plots.

```
Girth_vals = seq(min(trees$Girth)-2,
                 max(trees$Girth)+2, by = 1)
Height_vals = seq(min(trees$Height)-2,
                  max(trees$Height)+2, by = 1)
fit3_vals = matrix(data = NA, nrow = length(Girth_vals),
                   ncol = length(Height_vals))
for(i in 1:length(Girth_vals)){
  for(j in 1:length(Height_vals)){
    fit3_vals[i,j] = coef(fit3)[1] + coef(fit3)[2]*Girth_vals[i] + coef(fit3)[3]*Height_vals
  }
}                                                  # Computation of predicted values
par(mfrow = c(1,1))
persp(Girth_vals, Height_vals, fit3_vals,
      col = "grey", xlab = "Girth", ylab = "Height",
      zlab = "Volume", theta = 10, cex.lab = 1.3)
```



However, we can use plot3D package to generate nice plots. The colour gradients of the regression plane shade important light about the contribution of the predictors in the variation of the response variable.

```
library(plot3D)
persp3D(Girth_vals, Height_vals, fit3_vals,
        xlab = "Girth", ylab = "Height",
        zlab = "Volume", theta = 40, cex.lab = 1.3,
        alpha = 0.2)
points3D(trees$Girth, trees$Height, trees$Volume,
         col = "black", pch = 19, add = TRUE)
```

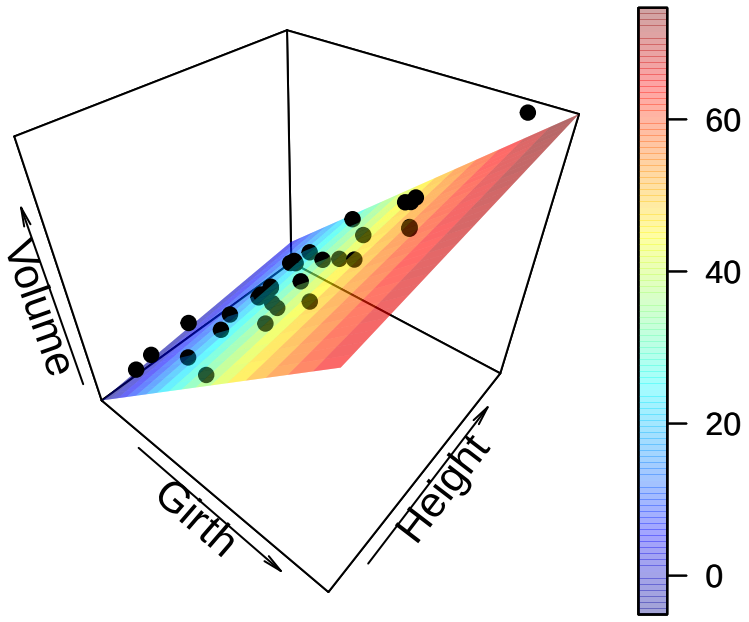


Figure 13: Fitted regression plane with two covariates Girth and Height. The response is the Volume of timber. The regression plane is given by the following equation: Volume=-57.9876589 + 4.7081605 × Girth + 0.3392512 × Height. The colour gradients shade important light about the effect of the predictors. Along the Girth axis, the changes in the colour intensity are large and as Girth increases, the changes in the colour intensity are significant. However, along the Height axis, there is no changes in the colour intensity. This also indicates that as Height changes, predicted volume of the timber does not change.

Using the package `scatterplot3d`, we can have interesting three dimensional visualization. The participants are encouraged to see the help file of the function `scatterplot3d()`.

```
library(scatterplot3d)
scatterplot3d(trees$Girth, trees$Height, trees$Volume,
              xlab = "Girth (inches)",
              ylab = "Height (ft)", zlab = "Volume (cubic ft)",
              pch = 19, cex.lab = 1.2)
```

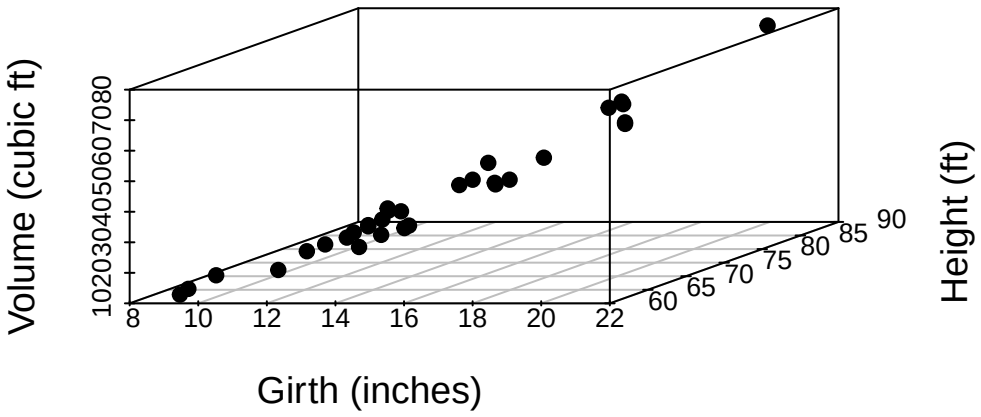


Figure 14: R also offers nice 3D visualization of the data sets using the scatterplot3D function.

Impact of outliers in a regression model

There are some interesting functions in R which helps us to identify the influential points and leverage points in the regression model.

```
par(mfrow = c(1,2))
cooks.distance(fit)                                  # computing Cook's distance
```

```
          1           2           3           4           5           6
1.098362e-01 4.924417e-02 2.223564e-02 4.160457e-05 3.971050e-03 6.044249e-03
          7           8           9          10          11          12
1.528743e-02 5.099320e-04 1.602747e-02 1.583849e-05 2.080153e-02 4.922094e-05
         13          14          15          16          17          18
4.656599e-04 1.278988e-03 2.524848e-02 3.718828e-02 2.809168e-02 8.762270e-03
         19          20          21          22          23          24
4.450995e-02 6.408063e-02 2.754799e-04 1.137428e-02 5.014444e-05 6.088904e-02
         25          26          27          28          29          30
1.847520e-02 6.459356e-02 5.008387e-02 7.597544e-02 2.844384e-02 3.976321e-02
         31
8.880581e-01
```

```
plot(cooks.distance(fit), pch = 19, col = "red",
     cex.lab = 1.3, main = "Cook's Distance",
     cex.main = 1.3)                                 # plot the distance per obs

plot(Volume ~ Girth, data = trees, col = "red",
     pch = 19, cex.lab = 1.3, main = "Fitted Lines",
```

```
     cex.main = 1.3)
abline(fit, col = "blue", lwd = 2)                    # adding the fitted line
fit4 = lm(formula = Volume ~ Girth, data = trees[-31,])
abline(fit4, col = "magenta", lwd = 2)                # fit without outlier
legend("topleft", legend = c("with 31", "without"),
       col = c("blue", "magenta"), lwd = c(2,2), bty = "n")
points(trees$Girth[31], trees$Volume[31], col = "magenta",
       cex = 2, lwd = 3)                              # mark the outlier
```

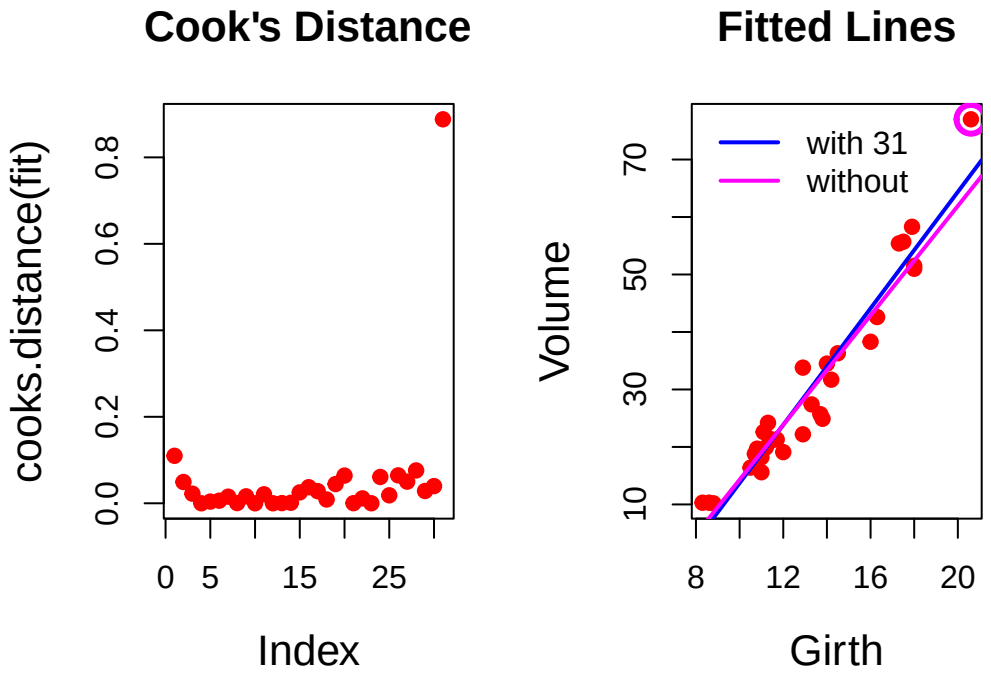


Figure 15: For each row, we obtained the Cook's distance and observe that the 31st observation has the highest Cook's distance. In the right panel we fitted the regression model with and without the 31st observation. The fitted lines are shown in blue and magenta colour respectively. It is evident that the presence deletion of the 31st observation has changed the slope estimate significantly.

Universality of the normal distribution

We recollect the example of estimating the true probability of the success (p) from the coin tossing experiment. If we toss the coin n times and compute the proportion of success ($\widehat{p_n}$). We keep the suffix n just to remind us that the distribution of the proportion of success changes as n changes.

a) Suppose we fix n (the sample size) and simulate n coin tossing experiment using R with fixed success probability $p = 0.3$.
b) We repeat this process $M = 1000$ times. In each repetition, we store the value of the proportion of success.
c) From the step (b) we have $M = 1000$ many proportions of success (possibly different values).
d) Draw a histogram of the values obtained in the previous steps.
e) Repeat (a) – (d) for different choices of n.

In the following, we implement the above steps. Let us remind ourselves that we are dealing with a coin tossing experiment and there is no discussion of normal distribution yet. One can envision the above simulation experiment as involving one thousand volunteers each tossing identical coins n times and reporting the proportion of successes. Intuitively, these proportions will vary due to inherent randomness. Our goal is to determine if this randomness can be described by a known probability distribution.

```
par(mfrow = c(2,3))
p = 0.3                                          # true P(success)
n_vals = c(5, 10, 20, 50, 100, 500)              # sample size
M = 1000                                         # number of replications
for(n in n_vals){                                # loop of sample size
  p_hat = numeric(length = M)
  for(i in 1:M){                                 # loop on replications
    x = rbinom(n = n, size = 1, prob = p)
    p_hat[i] = sum(x)/n                          # sample proportion
  }
  hist(p_hat, probability = TRUE, main = paste("n = ", n),
       xlab = expression(widehat(p[n])), cex.lab = 1.3,
       cex.main = 1.3)
}
```

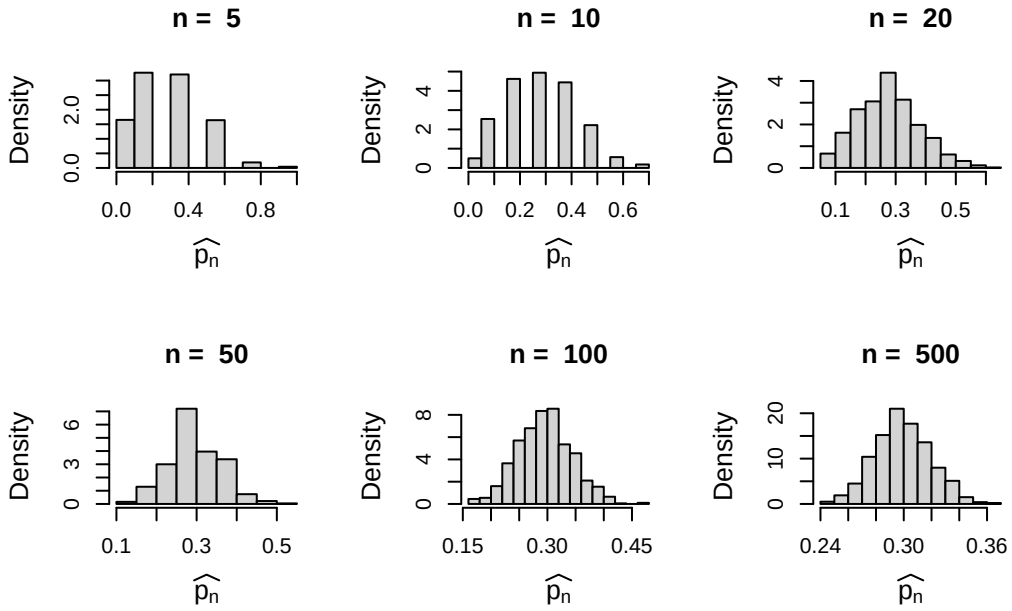


Figure 16: The variation in the proportion of success is visualized using histograms. The number of replications is 1000. It is interesting to see that as the sample size increases, the histograms tend to behave like bell-shaped.

In the above simulation experiments, we observed that the histograms are centred about the true value $p = 0.3$. Let us compute the variance. Although, we do not belong to the statistics background, but simple basic calculations will help us to become better quantitative ecologists. For example, here, we want to compute $Var(\hat{p_n})$.

First recall that if a random variable X takes only two values 1 or 0 with probability p and $1-p$ respectively. Then the expectation of the random variable X, denoted by $E(X)$ is computed as follows:

$$\mathrm{E}(X) = 0 \times \mathrm{P}(X = 0) + 1 \times \mathrm{P}(X = 1) = p.$$

Therefore, empirically, we can say that from a single through of a coin, we can expect p many successes. This statement makes much more sense, when we say that if we toss the coin 10 times, approximately, the number of successes will be 10p. In other words, the proportion of successes in 10 repetitions of coin tossing experiment will be close to p. The variance of X is computed as follows:

$$Var(X) = E(X^2) - (E(X))^2 = 0^2 \times P(X = 0) + 1^2 \times P(X = 1) - p^2 = p - p^2 = p(1 - p).$$

We can think of the outcome of each coin tossing experiment as a random variable taking values either 1 or 0. Let $X_1, \cdots, X_n$ be the outcome of the n coin tossing experiment and $P(X = 1) = P(\text{Head}) = p$. Therefore, the number of successes is $X_1 + X_2 + \cdots + X_n$ and outcome X_1 does not impact X_2 and similar for all the pairs.

$$Var(\widehat{p_n}) = Var\left(\frac{X_1 + X_2 + \cdots + X_n}{n}\right) = \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) = \frac{1}{n^2}\sum_{i=1}^{n} p(1-p) = \frac{p(1-p)}{n}.$$

Therefore, the standard error of $\widehat{p_n}$ is

$$\text{SE}(\widehat{p_n}) = \sqrt{\frac{p(1-p)}{n}}.$$

In the previous code snippet, we overlay each of the histogram with the normal density function with mean p and standard deviation $\sqrt{\frac{p(1-p)}{n}}$.

```r
par(mfrow = c(2,3))
p = 0.3
n_vals = c(5, 10, 20, 50, 100, 500)            # sample size
M = 1000                                       # number of replications
for(n in n_vals){
  p_hat = numeric(length = M)
  for(i in 1:M){
    x = rbinom(n = n, size = 1, prob = p)
    p_hat[i] = sum(x)/n
  }
  hist(p_hat, probability = TRUE, main = paste("n = ", n),
       xlab = expression(widehat(p[n])), cex.lab = 1.3,
       cex.main = 1.3)
  curve(dnorm(x, mean = p, sd = sqrt(p*(1-p)/n)),
        add = TRUE, col = "red", lwd = 2)       # add normal PDF
}
```

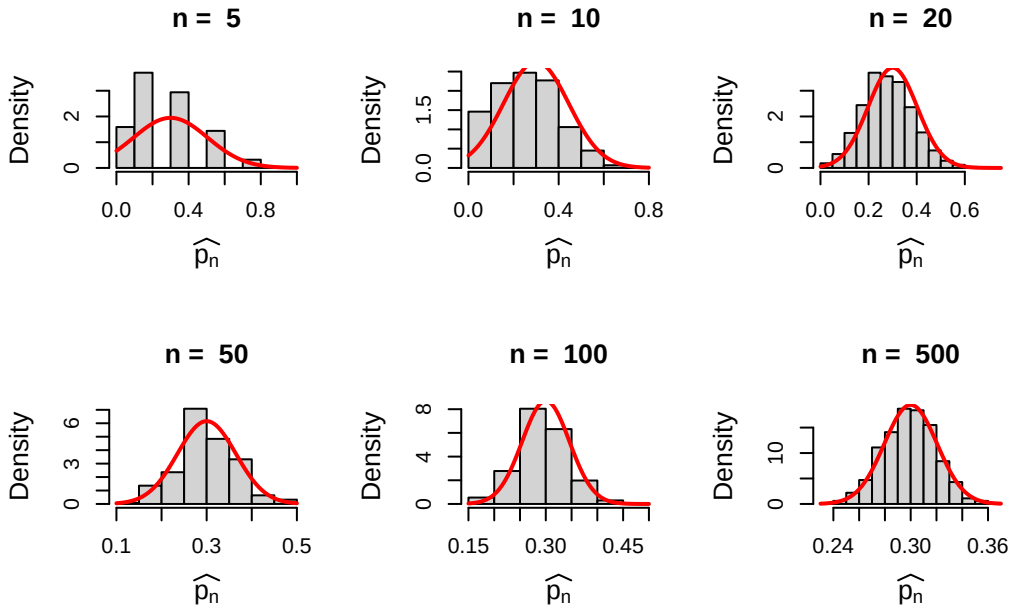


Figure 17: It is interesting to see that as the sample size increases, the variation in the proportion of success is well captured by the normal distribution with mean p and variance $\sqrt{\frac{p(1-p)}{n}}$.

The reader is encouraged to perform the above experiment for other distributions as well, like Poisson(λ), Geometric(p), Exponential(β), Gamma(α, β), Beta(a, b), etc. The letter `r` will be used for simulation purposes for all the distributions.

To summarize, basically, $\hat{p_n}$ is an average of the outcomes X_i's and the randomness in the average values can be well approximated by the normal distribution. This is due to the well-known Central Limit Theorem which states that the sample mean is well approximated by the normal distribution for large sample size n.

For the reader, we repeat this experiment using the exponential distribution with parameter $\beta = 1$. The exponential distribution with mean β is given by

$$f(x) = \begin{cases} \frac{1}{\beta} e^{-\frac{x}{\beta}}, & 0 < x < \infty \\ 0, & \text{Otherwise.} \end{cases}$$

The curve function can be conveniently used to draw the exponential PDF for different values of β. In the following, we plot three PDF side by side for $\beta \in \{1, 2, 0.5\}$.

```
par(mfrow = c(1,3))
curve(dexp(x, rate = 1), col = "red", -1, 6,
```

```
      lwd = 2, main = expression(paste(beta, " = ", 1)),
      cex.main = 1.4, cex.lab = 1.3)
curve(dexp(x, rate = 2), col = "red", -1, 4,
      lwd = 2, main = expression(paste(beta, " = ", 2)),
      cex.main = 1.4, cex.lab = 1.3)
curve(dexp(x, rate = 0.5), col = "red", -1, 10,
      lwd = 2, main = expression(paste(beta, " = ", 0.5)),
      cex.main = 1.4, cex.lab = 1.3)
```

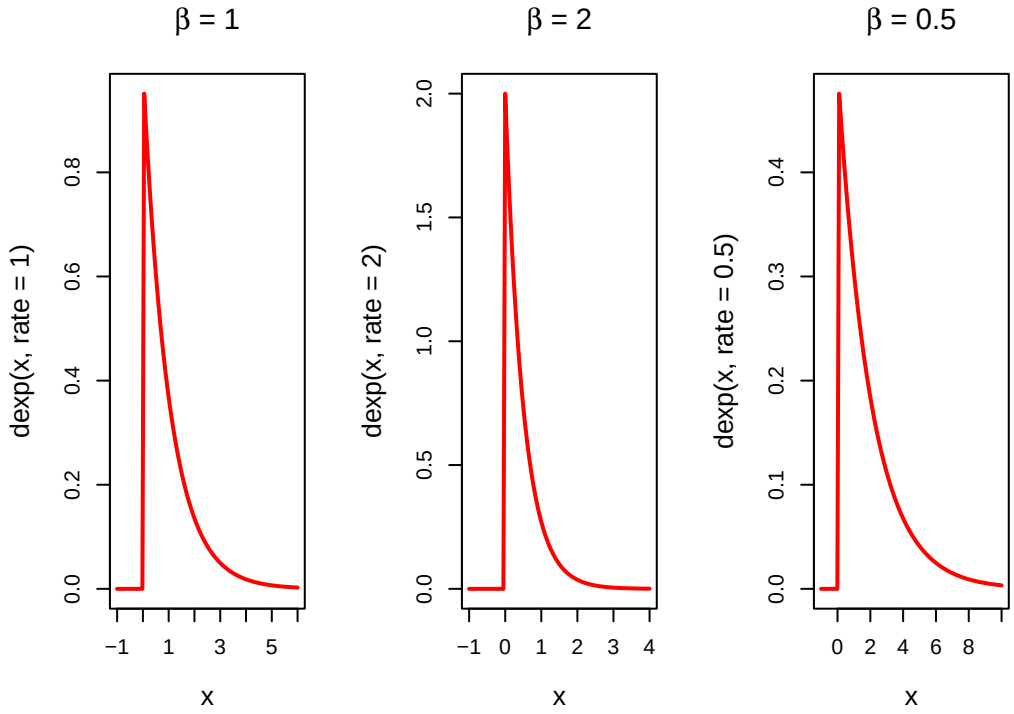


Figure 18: The exponential distribution for different choices of the parameter. It is evident that the random variable takes only positive values, and the distribution is positive skewed.

Now suppose that we draw a random sample of size n from the exponential distribution with $\beta = 1$. The mean of the distribution is β. The sample mean $\frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}_n$ can be well approximated by the normal distribution. In the following, we see that the histograms are bell shaped for large n values.

```
par(mfrow = c(2,3))
beta = 1
n_vals = c(5, 10, 20, 50, 100, 500) # sample size
```

```r
M = 1000                                # number of replications
for(n in n_vals){
  sample_means = numeric(length = M)
  for(i in 1:M){
    x = rexp(n = n, rate = 1/beta)
    sample_means[i] = sum(x)/n
  }
  hist(sample_means, probability = TRUE, main = paste("n = ", n),
       xlab = expression(bar(X[n])), cex.lab = 1.3,
       cex.main = 1.3)
  curve(dnorm(x, mean = beta, sd = beta/sqrt(n)),
        add = TRUE, col = "red", lwd = 2)
}
```

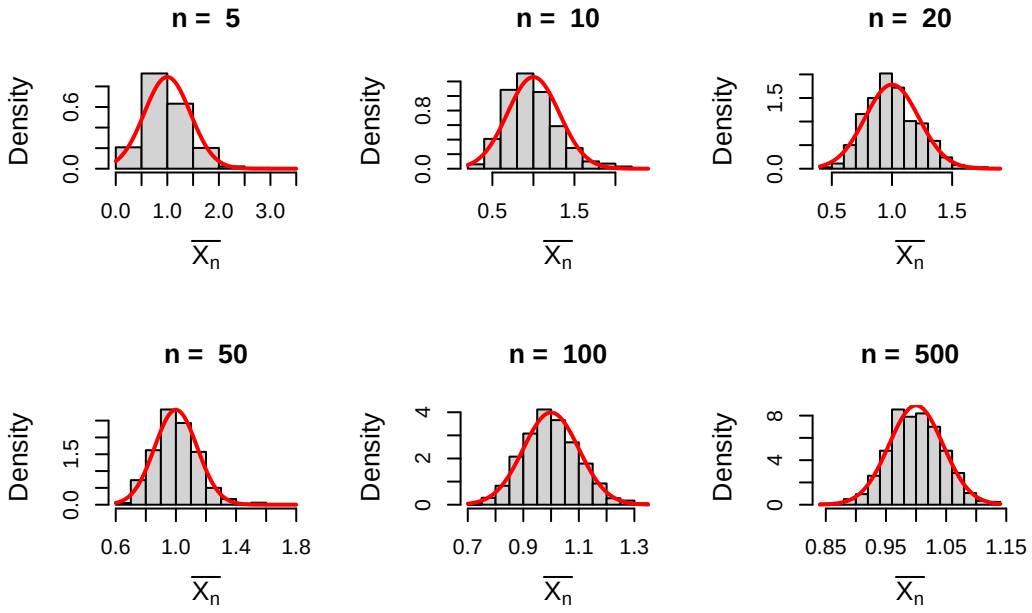


Figure 19: As the sample size increases, the sample mean is well approximated by the normal distribution. Note the mean of the normal distribution is at $\beta = 1$. The reader is encouraged to identify the variance of the normal distribution using which the histograms have been approximated.

Nonlinear Regression Models

In the investigations of natural processes, many times we come across nonlinear relationships between the response and the predictors. We have already discussed some aspects of linear

regression using R particularly the use of the function `lm()` in R. This function is particularly useful for the linear models or nonlinear models which can be converted to a linear model. However, there are many models that cannot be written as a linear function of the parameters. In such a case, nonlinear regression modelling is required. In R the `nls()` function takes care of the estimation of parameters for nonlinear function.

To investigate the nonlinear regression concepts in the context of ecological models, we first describe a phenomenon known as the Allee effect. The Allee effect, named after ecologist W. C. Allee, corresponds to density-mediated drop in the population fitness when they are small in numbers (Allee, 1931; Dennis, 1989; Fowler and Baker, 1991; Stephens and Sutherland, 1999). The harmful effects of inbreeding depression, mate limitation, predator satiation etc. reduce fitness as the population size decreases. For such dynamics, the maximum fitness is achieved by the species at an intermediate population size, unlike logistic or theta-logistic growth models. Such observations usually correspond to the mechanisms giving rise to an Allee effect. In recent decades, due to an increasing number of threatened and endangered species, the Allee effect has received much attention from conservation biologists. The related theoretical consequences and the empirical evidence have made the Allee effect an important component in both theoretical and applied ecology. In general, there are two types of Allee effects are considered in the natural populations across a variety of taxonomic groups, viz. component and demographic Allee effect. The component Allee effect modifies some component of individual fitness with the changes in population sizes or density. If the per capita growth rate is low at small density but remains positive is called the weak demographic Allee effect. The strong Allee effect is characterized by a threshold density below which the per capita growth rate is negative that leads to extinction deterministically. The critical density is called the Allee threshold and has significant importance in conservation biology (Hackney and McGraw, 2001), population management (Myers et al., 1995; Liermann and Hilborn, 1997) and invasion control (Johnson et al., 2006). There are a large number of studies available in the literature on the mathematical modelling of the demographic Allee effect (see Table 3.1 Courchamp et al. (2008)).

A bit of mathematical quantities to measure the growth of populations would be helpful to understand the context. If $X(t)$ represents the population size at time t, then, the derivative of the population size with respect to t, that is $\frac{dX(t)}{dt}$ is called the absolute growth rate and $\frac{1}{X(t)}\frac{dX(t)}{dt}$ is called the per capita growth rate (PGR) at time t. Note the that the PGR has dimension time^{-1}. If rate of growth is bigger than zero, then the population grows and if it is negative population decline.

Consider the following equation

$$\frac{dX}{dt} = rX\left(\frac{X}{A} - 1\right)\left(1 - \frac{X}{K}\right),\ X(0) = X_0.$$

The quantities A is the Allee threshold and K is the carrying capacity of the environment and r is the intrinsic growth rate. The parameters A and K satisfy the inequality $0 < A < K$.

It is easy to see that if $X_0 < A$, then $\frac{dX}{dt} < 0$, that means the population declines and for $0 < X_0 < K$, $\frac{dX}{dt} > 0$, therefore, the population grows.

In the following we learn how to solve such differential equations using R. Then, we can visualize various dynamics of the population with various initial population sizes X_0.

```
library(deSolve)                                # load the library
A = 30                                          # Allee threshold
K = 80;                                         # Carrying capacity
AlleeFun = function(t, state, param){           # Allee growth equation
  with(as.list(c(state,param)),{
    dX = r*X*(X/A - 1)*(1 - X/K);
    return(list(c(dX)));
  })
}
parameters = c(r = 0.5, K = 80,A = 30)          # parameters of the model
times = seq(0, 10, by = 0.01)                   # time frame
init.pop = 35                                   # initial population size
state = c(X = init.pop)
out = ode(y = state, times = times, func = AlleeFun,
          parms = parameters)                   # solving the ODE
plot(out, main = "Solution of Allee growth equation",
     ylim=c(10, 90), xlab = "time",
     ylab = "population size", col = 2, lwd = 2,
     cex.lab = 1.3)
abline(h = A, col = 6, lwd=2, lty=2)            # horizontal line at A
abline(h = K, col = 8, lwd=2, lty=3)            # horizontal line at K
text(0, 83, "K", lwd = 2)
text(0, 33, "A", lwd = 2)
```

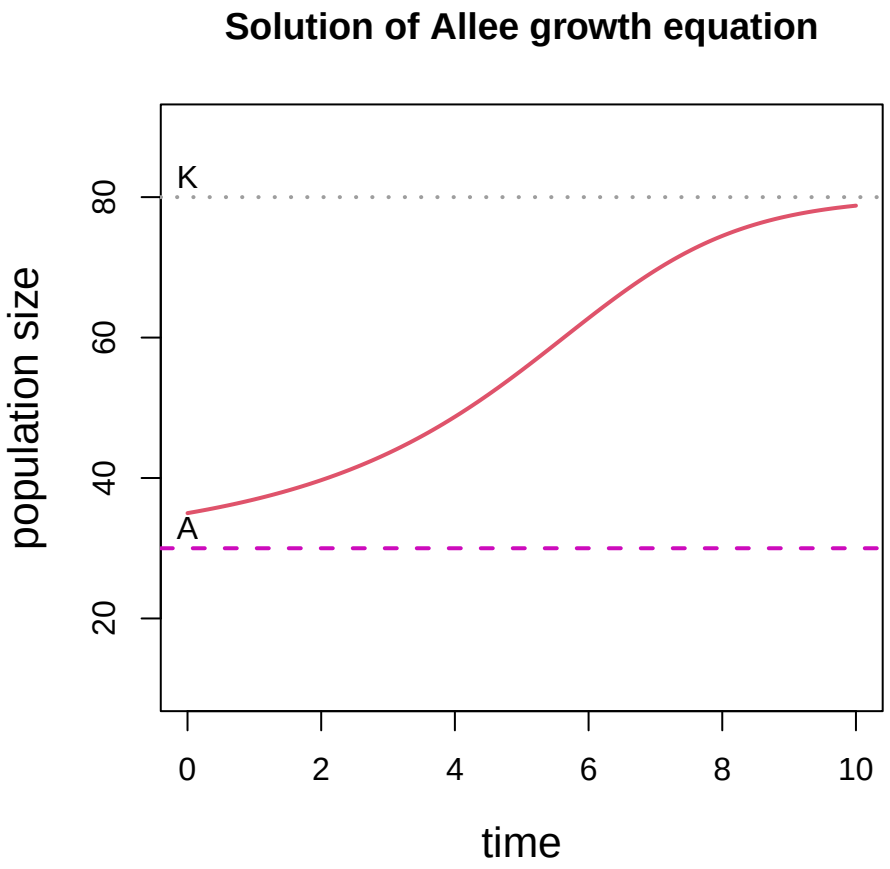


Figure 20: The solution of the Allee growth equation. If the initial condition is above A, then the solution converges to the carrying capacity.

We can consider multiple initial population size and see how the solution of the differential equation behaves. The following code will show that if the initial population size is below threshold A, the population goes to extinction and if the initial population size is above A, the population eventually settles down at K.

```
init.pop = c(10, 20, 40, 50, 60, 100)         # Different initial size
for(i in 1:length(init.pop)){
  state = c(X = init.pop[i])
  out = ode(y = state, times = times, func = AlleeFun,
            parms = parameters)
  if(i==1)
    plot(out, main = "Solution of Allee growth equation",
         ylim=c(0, 100), xlab = "time",
         ylab = "population size", col = i, lwd = 2,
         cex.lab = 1.3)
```

```
  else
    lines(out, col = i, lwd = 2)
}
abline(h = A, col = 6, lwd=2, lty=2)
abline(h = K, col = 8, lwd=2, lty=3)
text(0, 83, "K", lwd = 2)
text(0, 33, "A", lwd = 2)
```

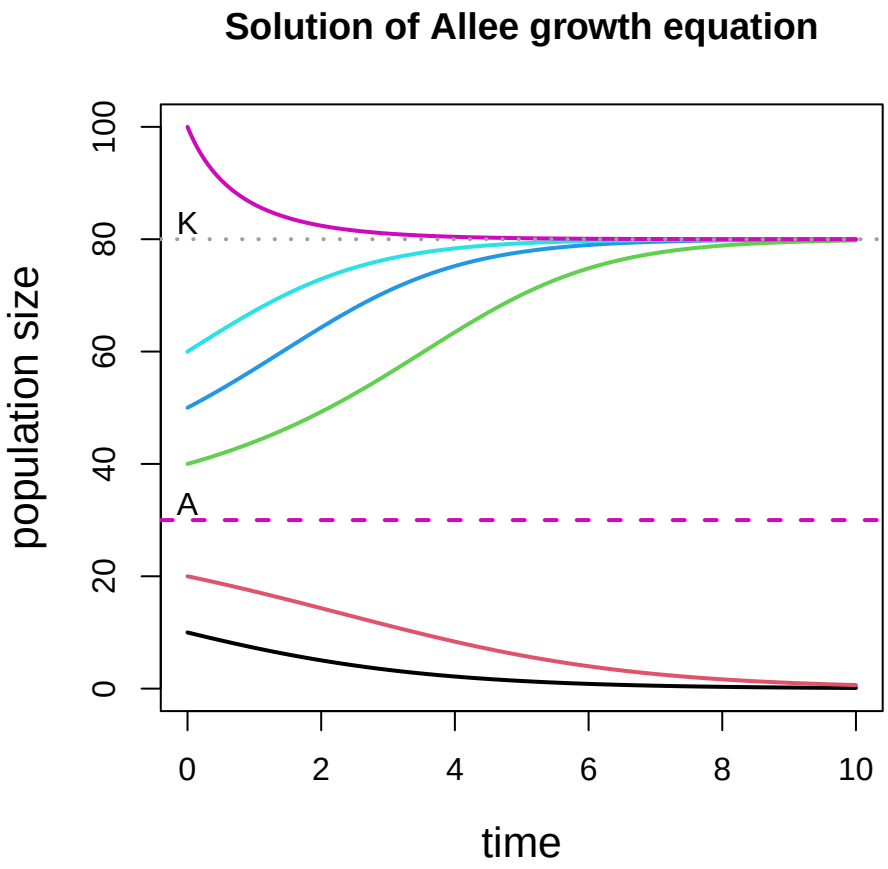


Figure 21: Solution of the differential equation representing the Allee growth profiles of the natural populations. If the initial population size is below the Allee threshold, the population size eventually extinct, whereas if the population growth starts with a population size bigger than A, the population settles down at K eventually.

Fitting of the Allee growth equations

Before going into fitting exercises, we shall learn how to simulate the population dynamics from the model populations. Suppose that we want to simulate the population dynamics of a pop-

ulation governed by the Allee growth equation and the population is subject to demographic stochasticity alone.

First, we fix the parameters for the simulations. We simulation the initial population size $X_0 \sim N(25, 3^2)$ and the simulate the per capita growth from $r_t \sim \mathcal{N}(\mu_t, \sigma^2 dt)$, where $\mu_t = r\left(\frac{X_t}{A} - 1\right)\left(1 - \frac{X_t}{K}\right)$. Then we simulate $X_1 \sim X_0 e^{r_t dt}$. The process is continued till the end time point. In the following code, we first fix the parameters:

```
r = 0.05                                    # true parameter value r
A = 20                                      # Allee threshold
K = 100                                     # Carrying capacity
time = seq(0, 10, by = 0.05)                # time points
dt = time[2] - time[1]                      # time difference
sig_e = 0.1                                 # environmental standard deviation
par(mfrow = c(1,2))
```

In the following code, we perform a single simulation of the population dynamics.

```
popSim = numeric(length = length(time))
x0 = rnorm(n = 1, mean = 30, sd = 3)
popSim[1] = x0
for (i in 1:(length(time)-1)) {
  mu = r*(popSim[i]/A - 1)*(1- popSim[i]/K)
  pgr = rnorm(n = 1, mean = mu, sd = sig_e*sqrt(dt))
  popSim[i+1] = popSim[i]*exp(pgr)
}
plot(time, popSim, col = "red", lwd = 2, type = "l",
     xlab = "time", ylab = "Population size", cex.lab = 1.3)
```

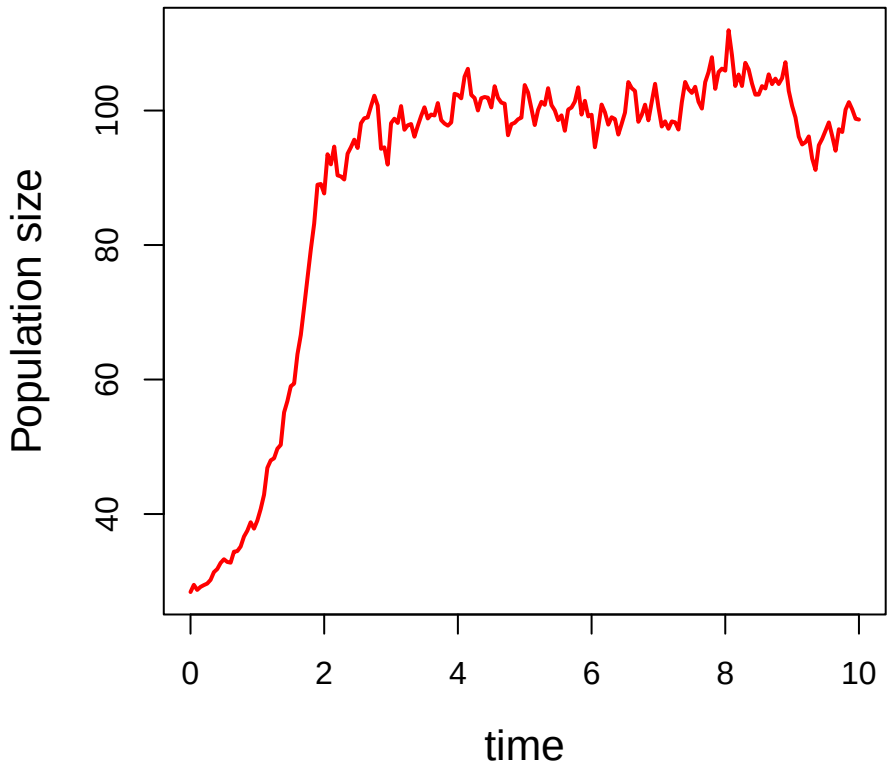


Figure 22: A single simulated trajectory of the population size subject to Allee effect. As the envrionmental variance σ_e^2 increases, the simulations will be more wild in nature. The reader is encouraged to do this experimentation using different values of σ_e^2 and also experiment with different values of K and A and x_0.

In the following we perform 50 simulations and obtain 50 possible population trajectories.

```
nsim = 50
popSim = matrix(data = NA, nrow = nsim, ncol = length(time))
popSim = matrix(data = NA, nrow = nsim, ncol = length(time))
for(i in 1:nsim){
  x0 = rnorm(n = 1, mean = 25, sd = 5)
  popSim[i,1] = x0
  for (j in 1:(length(time)-1)) {
    mu = r*(popSim[i,j]/A - 1)*(1- popSim[i,j]/K)
    pgr = rnorm(n = 1, mean = mu, sd = sig_e*sqrt(dt))
    popSim[i, j+1] = popSim[i,j]*exp(pgr)
  }
}
matplot(time, t(popSim), type = "l", lty = 1,
```

```
14          ylab = "Population size", cex.lab = 1.3)
```

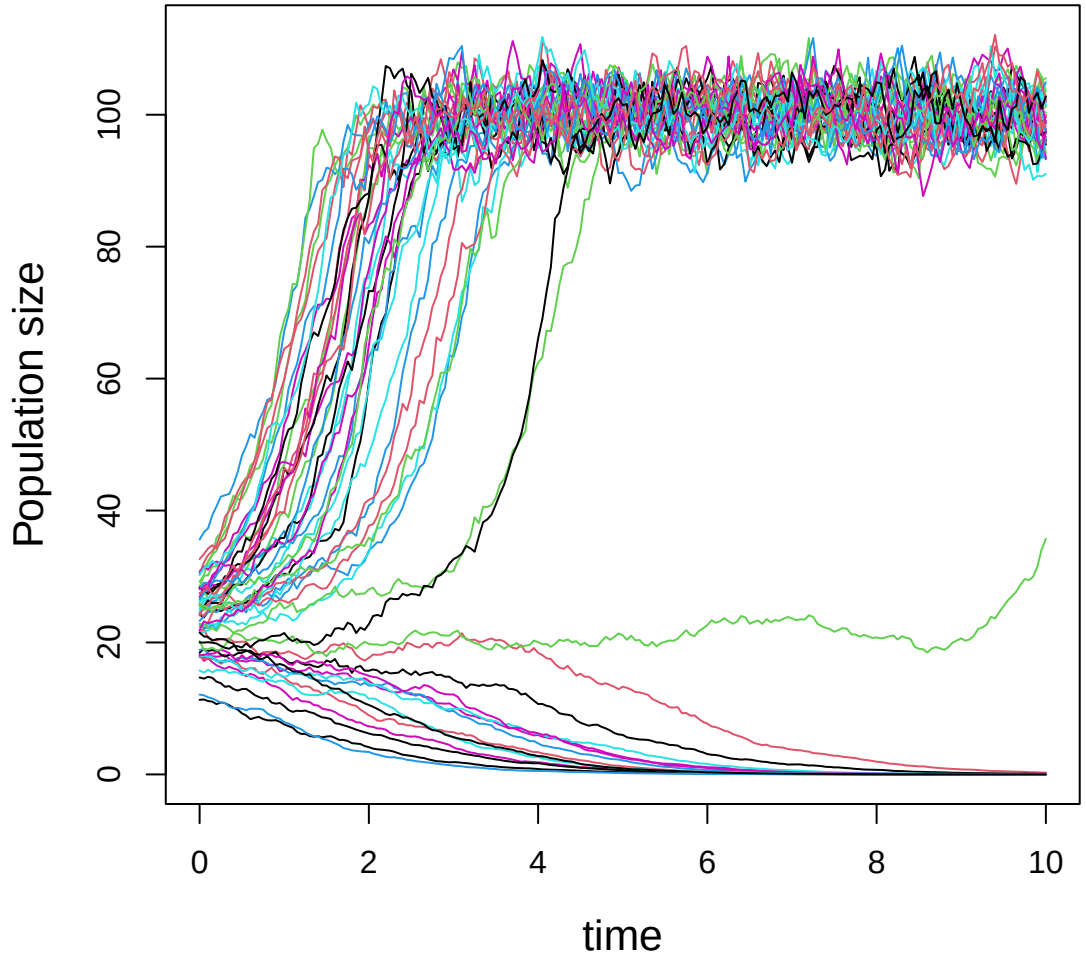


Figure 23: We consider several initial population sizes and simulate 50 trajectories of the population dynamics. It is to be noted that if the initial population size is below the Allee threshold, the population is likely to go extinct. Therefore, some trajectories lead to extinction of the population.

Now we concentrate on the fitting of the Allee growth equation to the real data set. Suppose that the following data has been collected from the field of some species populations.

29, 30, 31, 32, 34, 37, 40, 42, 43, 45, 52, 59, 61, 54, 58, 63, 65, 71, 74, 75, 77, 82, 83, 89, 91, 97, 95, 105, 97, 102, 104, 99, 101, 95, 100, 97, 98, 100, 102, 98, 97.

First, we store these data values in an object x using the concatenation operator c and then plot both size profile and time per capita growth profile of the population. We estimate the

per capita growth

```r
par(mfrow = c(1,2))
x = c(29, 30, 31, 32, 34, 37, 40, 42, 43, 45, 52, 59, 61, 54, 58, 63,
      65, 71, 74, 75, 77, 82, 83, 89, 91, 97, 95, 105, 97, 102, 104,
      99, 101, 95, 100, 97, 98, 100, 102, 98, 97)
plot(x, type = "b", pch = 19, col = "red",
     xlab = "Years", ylab = "Population size", cex.lab = 1.2)
n = length(x)                                          # length of the data
R = log(x[2:n]/x[1:(n-1)])                             # Per capita growth rate
plot(x[1:(n-1)], R, xlim = c(0, 120), col = "grey",
     ylab = "PGR", xlab = "Population size", pch = 19,
     cex.lab = 1.2)
abline(h = 0, col = "red", lwd = 2)                    # adding horizontal line

nls_fit = nls(R ~ r*(x[1:(n-1)]/A-1)*(1-x[1:(n-1)]/K),
              start = list(r = 0.04, A= 20,K=90))
summary(nls_fit)                                       # summary of the nls
```

```
Formula: R ~ r * (x[1:(n - 1)]/A - 1) * (1 - x[1:(n - 1)]/K)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
r  0.01018    0.06629   0.154    0.879
A  3.83204   22.88810   0.167    0.868
K 98.61951    4.36712  22.582   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04698 on 37 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 4.522e-08
```

```r
r_hat = coefficients(nls_fit)[1]                       # estimate of r
A_hat = coefficients(nls_fit)[2]                       # estimate of A
K_hat = coefficients(nls_fit)[3]                       # estimate of K

curve(r_hat*(x/A_hat-1)*(1-x/K_hat), add = TRUE,
      col = "blue", lwd = 2)                           # adding fitted curve
```

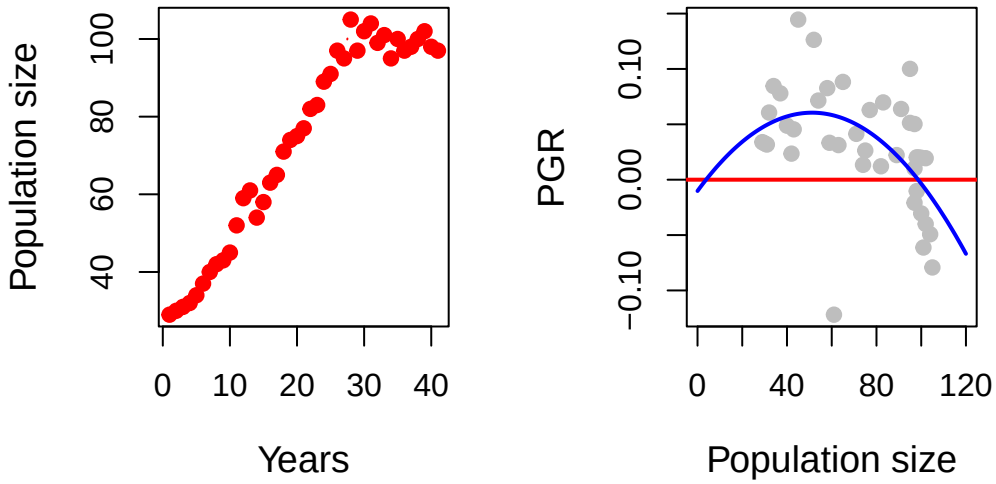


Figure 24: The left panel depicts the population size over time. The right panel is the per capita growth profile with respect to population size.

Similar to the linear regression, nonlinear regression models can also be compared. Suppose that the same per capita growth profile is also analysed using the logistic growth equation. The logistic growth equation is given by

$$\frac{dX}{dt} = r_m X\left(1 - \frac{X}{K}\right), \; X(0) = X_0, \tag{0.5}$$

where, r_m is the intrinsic growth rate (also referred to as the maximum per capita growth rate) and K is the carrying capacity of the environment. Add the following code snippet below the previous code snippet.

Here, the per capita growth profile of the logistic growth equation is a linear function of the population size $r_m\left(1 - \frac{X}{K}\right)$ which can be written in the form $a + bX$, where $a = r_m$ and $b = -\frac{r_m}{K}$. Therefore, we can apply linear regression model to perform this fitting exercise, however, we apply the `nls()` function only to get the estimates of r_m and K as given below. Therefore, `nls()` function can also be used to perform linear regression.

```
plot(x[1:(n-1)], R, xlim = c(0, 120), col = "grey",
     ylab = "PGR", xlab = "Population size", pch = 19,
     cex.lab = 1.2)
abline(h = 0, col = "red", lwd = 2)                    # adding horizontal line
curve(r_hat*(x/A_hat-1)*(1-x/K_hat), add = TRUE,
      col = "blue", lwd = 2)                           # adding fitted curve

nls_fit_log = nls(R ~ r_m*(1-x[1:(n-1)]/K),
                  start = list(r_m = 0.04, K=90))      # fitting of logistic
```

```r
rm_hat = coefficients(nls_fit_log)[1]                    # estimate of rm
K_hat = coefficients(nls_fit_log)[2]                     # estimate of K
curve(rm_hat*(1-x/K_hat), add = TRUE,
      col = "magenta", lwd = 2)                          # add fitted curve
legend("bottomleft", legend = c("Allee growth", "Logistic"),
       col = c("blue", "magenta"), lwd = c(2,2),
       bty = "n", cex = c(0.8, 0.8))                     # add legend to plot
```

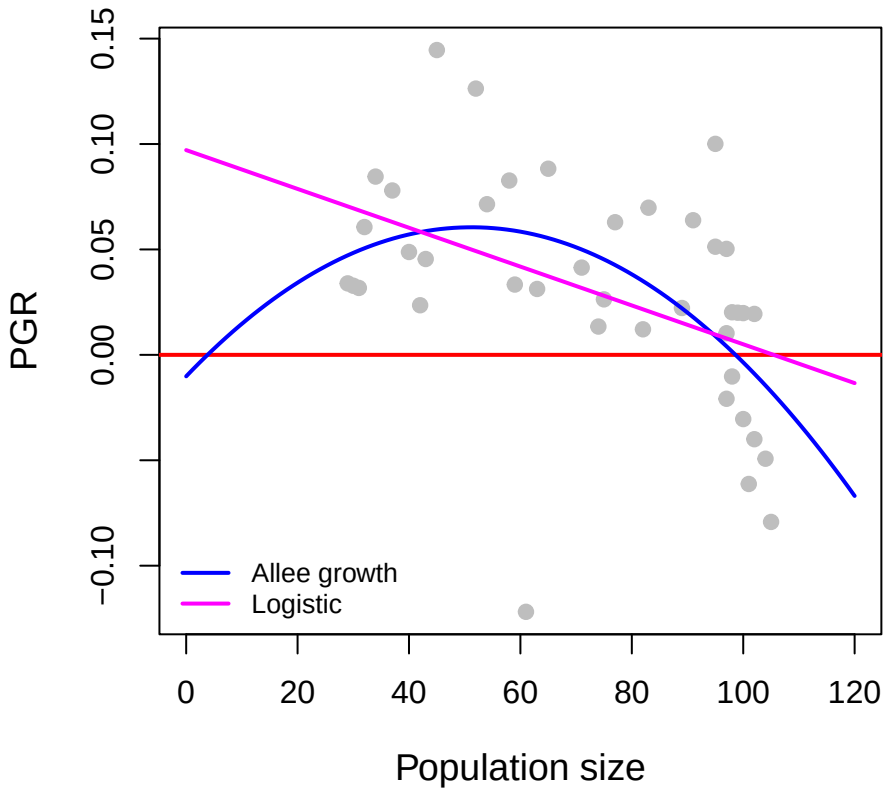


Figure 25: The fitted logistic growth and the Allee growth equations are shown in this plot. Both these equations are fitted using the nonliear regression model.

In the following, we compare the curve fitting exercises. One way to look at the actual versus predicted values and a high correlation between them gives indication of a better fitting. In the following, we plot the actual and fitted per capita growth rates obtained by Allee growth equation and the logistic growth equation. We also compute the corresponding correlations.

```r
par(mfrow = c(1,2))
plot(R, fitted.values(nls_fit), pch = 19, col = "grey",
     xlab = "Observed PGR", ylab = "Fitted PGR", cex.lab = 1.2,
```

```
     main = "Allee growth equation")
cor(R, fitted.values(nls_fit))                    # correlation value
```

```
[1] 0.5125905
```

```
abline(lm(fitted.values(nls_fit) ~R), col = "red", lwd = 2)
plot(R, fitted.values(nls_fit_log), pch = 19, col = "grey",
     xlab = "Observed PGR", ylab = "Fitted PGR", cex.lab = 1.2,
     main = "Logistic growth equation")        # R versus fitted values
abline(lm(fitted.values(nls_fit_log) ~R), col = "red", lwd = 2)
cor(R, fitted.values(nls_fit_log))                # correlation value
```

```
[1] 0.4515992
```

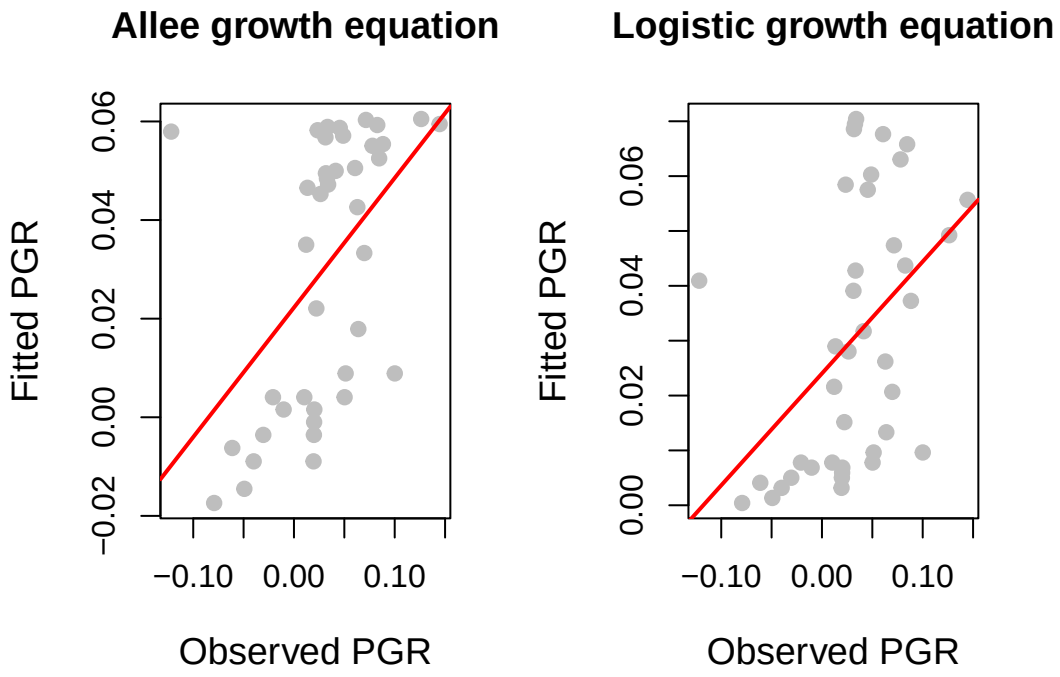


Figure 26: We expect the data points to be close to the red line to give a strong correlation between the actual versus the fitted values of the per capita growth rate.

For the Allee and logistic growth model fitting, the correlation between the observed per capita growth rate and predicted per capita growth rates are 0.51 and 0.45, respectively, which shows approximately 50% agreement. Using the AIC function, we also compute the Akaike Information Criterion to compare these two fitting exercises.

```r
AIC(nls_fit)
```

```
[1] -126.2541
```

```r
AIC(nls_fit_log)
```

```
[1] -125.1844
```

The difference between the AIC values is less than 5, therefore, these two models perform almost in a similar way.

Till now, in our discussion, we have considered the all the data points. It is important to note that one data point always lie away from the mass of the per capita growth rate values. In the scatterplot of actual versus fitted values, in the top left corner of the window, a data point is observed. As a practitioner, presence of such points needs to be considered and take appropriate action. This will be an exercise for the participants to redo the above exercises by fitting exercises without this stipulated point which is suspected to be an outlier.

Bootstrapping regression model

Bootstrapping is a powerful resampling technique which helps in evaluating the goodness of the fitting exercises. In the bootstrapping process, we create a new data set from the original data set. We have the data set which contains the first column as the population size and the second column as the per capita growth rates at those population sizes. If there are n rows in the data, we randomly select n rows from the data following the principle of simple random sampling with replacement (SRSWR) and create a new data set with those selected rows. It is understood that in the new data set some rows may be repeated (due to SRSWR). The new data set is called the bootstrap data set. We create $B = 1000$ bootstrap data sets from the original data set. We perform the fitting exercise on each of the bootstrap data sets and record the estimates of the parameters. The histogram of the B many estimates constitute the bootstrap sampling distribution of the estimators of the parameters. We carry out this exercise for the logistic growth model using the above data set.

```r
B = 1000                                    # number of bootstrap samples
rm_hat = numeric(B)                         # bootstrap estimate of rm
K_hat = numeric(B)                          # bootstrap estimate of K

D = data.frame(R = R, x = x[1:(n-1)])       # original data
for(i in 1:B){
```

```
  ind = sample(1:nrow(D), replace = TRUE)       # SRSWR
  nls_fit_boot = nls(R ~ r_m*(1-x/K), data = D[ind,],
                     start = list(r_m = 0.1, K = 80))
  rm_hat[i] = coefficients(nls_fit_boot)[1]     # bootstrap estimate
  K_hat[i] = coefficients(nls_fit_boot)[2]      # bootstrap estimate
}
par(mfrow = c(1,2))
hist(rm_hat, probability = TRUE, main = "",
     xlab = expression(widehat(r[m])), breaks = 30)
hist(K_hat, probability = TRUE, main = "",
     xlab = expression(widehat(K)), breaks = 30)
```

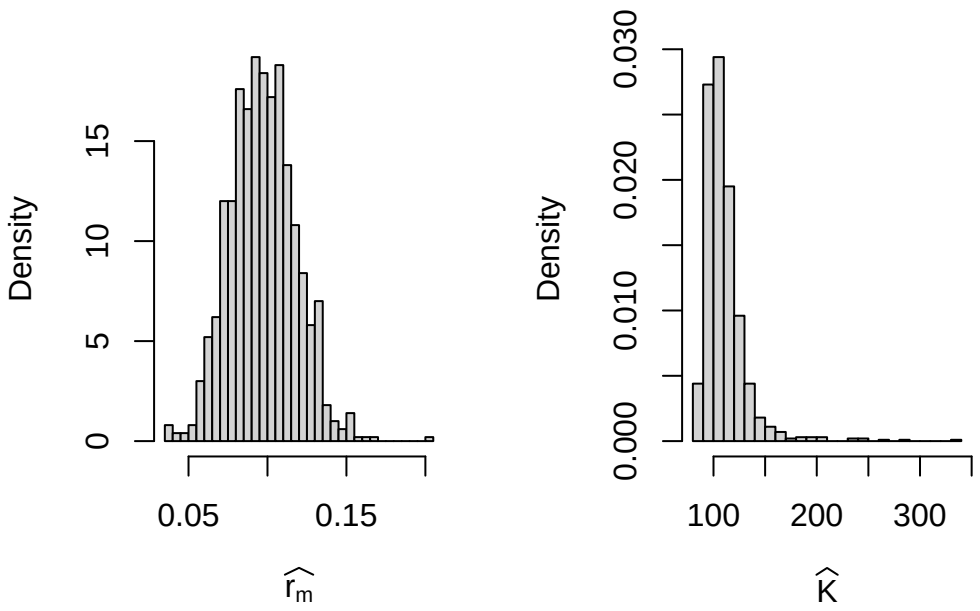


Figure 27: The bootstrap sampling distribution of the nonlinear least squares estimators of the parameters for the logistic growth equation. The number of bootstrap data set is 1000. Indication of skewed distribution for $\widehat{K}$ depicts a significant impact of some outlying observations.

The reader is encouraged to update the above code and compute the AIC values for each of the bootstrap fitting. If the bootstrap distribution of the AIC values is wide, which indicates the sensitivity of the fitting on the individual data points.

Some more concepts in Model selection

Let us get back to our data analysis task once again. We tried to predict the volume of the timber using the Girth variable as the predictor. In fact, we ended up with two regression models one is linear, and the other one is quadratic. We can predict the volume for some diameter of a tree for whom the volume is not available. A natural question arises, how would we validate the predictions, that is, how close the predictions to the true volume of the timer? One may argue that future is always unknown, therefore, there is no chance to check the goodness of the predictions. Or in other words, we need to rely completely on the model that is built on the whole data set (here there are 31 rows).

Training and test set

We may think of having a better strategy. Our whole data set consists of 31 rows, and we can keep about 20% of the rows aside (about 6 rows) and build the models on rest 80% of the rows. In the data science literature, the bigger set (80%) on which the model fitting exercises are carried out, is called the training data. The smaller set is referred to as the test data. The fitted model is used to predict the volume of timber using the girth values in the test data. It is important to note that the predicted values on the test data can now be assessed how close they are to the actual response values. The selection of the training data (rows) is done randomly, that is, randomly 25 rows are selected out of the 31 rows and create a training data consisting of the selected 25 rows and we create another data set with remaining 6 rows. In this new setting we minimize the following sum of squares and obtain the estimates of the parameters b_0, b_1 and σ^2:

$$\sum_{i=1}^{25} \left(\text{Volume}_i^* - b_0 - b_1 \times \text{Girth}_i^*\right)^2.$$

And we predict the volume on the test data using $\text{Volume}_i = b_0 + b_1 \times \text{Girth}_i$, where i represents the rows in the data which belongs to the test set. The test set contains six observations. Therefore, we have six predictions of the volume. The accuracy of the predictions is assessed by the mean squared error as follows:

$$\frac{1}{6}\sum_{i=1}^{6} \left(\widehat{\text{Volume}}_i - \text{Volume}_i\right)^2.$$

The above quantity gives an idea how good the predictions are and also referred to as test mean squared error (mse). In fact, this can be used to compare the prediction accuracy for two different models. The same exercise can be carried out for the quadratic regression model as well. If the prediction accuracy for the quadratic regression is more than the linear regression model, one may prefer to utilize the quadratic equation for the prediction purpose. Performing

the train-test procedure is essentially to provide a greater confidence level that the fitted model expected to perform well on the future observation.

Let us execute the above task and check the predictive accuracy of the linear and quadratic regression model to predict the volume of the timber using the diameter. The following R code will do this task. We will use the sample function to select 25 rows randomly out of 31 rows in the data set.

```
data(trees)
train = sample(1:nrow(trees), size = floor(nrow(trees)*0.8))
train_trees = trees[train, ]                                  # training data
print(train)                                                  # training rows
test_trees = trees[-train, ]                                  # test data
train_fit = lm(Volume ~ Girth, data = train_trees)
train_fit2 = lm(Volume ~ Girth + I(Girth^2),
                data = train_trees)
predict_fit = predict(train_fit, newdata = test_trees)
predict_fit2 = predict(train_fit2, newdata = test_trees)

mean((predict_fit - test_trees$Volume)^2)                     # test mse linear
mean((predict_fit2 - test_trees$Volume)^2)                    # test mse quadratic
```

The reader is encouraged to run above code snippet multiple times and observe the test mean squared error values change in different runs. This is due to the random selection of the rows. Using the `print(train)` you can check which rows are being selected in the training data.

To come to an objective conclusion, we should not deal with such type of randomness, and we seek to get a measure that takes care of this randomness. The test mean squared error is a random quantity. We may seek to compute the average test mean squared error. The idea is that we perform the train test process a large number of times (M, say) and every time compute the test mean squared error and compute the average error. The quadratic model will be preferred if the average test mean squared error is less than the average test mean squared error for the simple linear regression model. Let us do this task below and compute the average test mean squared error for both the models.

```
M = 500                                       # number of repetitions
test_mse_fit = numeric(length = M)            # store test mse linear
test_mse_fit2 = numeric(length = M)           # store test mse quadratic
for(i in 1:M){
  train = sample(1:nrow(trees), size = floor(nrow(trees)*0.8))
  train_trees = trees[train, ]                # training data
  test_trees = trees[-train, ]                # test data
  train_fit = lm(Volume ~ Girth, data = train_trees)
```

```r
  train_fit2 = lm(Volume ~ Girth + I(Girth^2),
                  data = train_trees)
  predict_fit = predict(train_fit, newdata = test_trees)
  predict_fit2 = predict(train_fit2, newdata = test_trees)
  test_mse_fit[i] = mean((predict_fit - test_trees$Volume)^2)
  test_mse_fit2[i] = mean((predict_fit2 - test_trees$Volume)^2)
}
par(mfrow = c(1,2))                                    # two plots side by side
hist(test_mse_fit, probability = TRUE,
     main = "Linear Regresion", breaks = 30,
     xlab = "test mse", cex.lab = 1.3)            # distribution of test mse
points(mean(test_mse_fit), 0, pch = 19,
       col = "red", cex = 1.3)                       # average test mse
hist(test_mse_fit2, probability = TRUE,
     main = "Quadratic Regression", breaks = 30,
     xlab = "test mse", cex.lab = 1.3)
points(mean(test_mse_fit2), 0, pch = 19,
       col = "red", cex = 1.3)
```

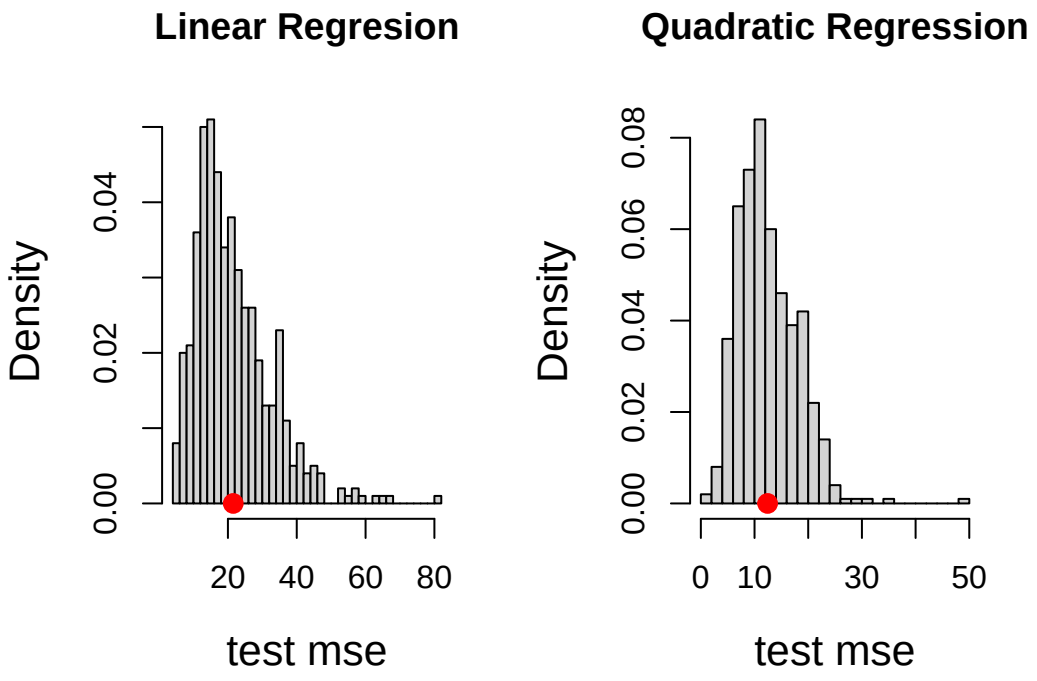


Figure 28: The variation in the test mean squared error in the train-test validation exercises is shown using histograms. The average test mean squared error is shown using red dot. The performance of the quadratic equation is better in predicting the future observations.

One may argue that by this process it is not guaranteed that all the individual data points has been used as a training point or as test point. Therefore, still some dependence on the individual data points remained. One way to deal with it to ensure that every row in the data set is a part of both training and test data. Suppose consider the first row as the test data and rest 30 rows as training data. Based on the fitted model we obtain the prediction for the first row and compute $\left(\widehat{\text{Volume}}_1 - \text{Volume}_1\right)^2$. The same process is carried out by removing the second row and compute $\left(\widehat{\text{Volume}}_2 - \text{Volume}_2\right)^2$ and continue this process till the 31^{st} row. Basically, each data point is used as both training and test data point and this process is known as Leave One Out Cross Validation (LOOCV). The LOOCV error is computed as:

$$\frac{1}{31}\sum_{i=1}^{31}\left(\widehat{\text{Volume}}_i - \text{Volume}_i\right)^2.$$

The model with the lesser LOOCV error may be preferred as the prediction equation for the given data analysis problem. In the following, let us compute the LOOCV error for both simple linear regression and quadratic regression model.

```
test_mse_fit = numeric(length = nrow(trees))
test_mse_fit2 = numeric(length = nrow(trees))
for(i in 1:nrow(trees)){
  train_trees = trees[-i, ]                          # drop ith the row
  test_trees = trees[i, ]                            # ith row in test
  train_fit = lm(Volume ~ Girth, data = train_trees)
  train_fit2 = lm(Volume ~ Girth + I(Girth^2),
                  data = train_trees)
  predict_fit = predict(train_fit, newdata = test_trees)
  predict_fit2 = predict(train_fit2, newdata = test_trees)
  test_mse_fit[i] = (predict_fit - test_trees$Volume)^2
  test_mse_fit2[i] = (predict_fit2 - test_trees$Volume)^2
}

cat("The LOOCV error of the linear regression model\n")
```

```
The LOOCV error of the linear regression model
```

```
mean(test_mse_fit)                                   # LOOCV linear regression
```

```
[1] 20.5653
```

```r
cat("The LOOCV error of the quadratic regression model\n")
```

```
The LOOCV error of the quadratic regression model
```

```r
mean(test_mse_fit2)                                    # LOOCV quadratic regression
```

```
[1] 11.93375
```

The output of the above code gave the LOOCV estimate of the prediction error (test mean squared error) 20.5652 and 11.93375 for the linear and quadratic regression, respectively. Therefore, by using the leave one out cross validation, one may choose to go ahead with the quadratic regression model to predict the volume of the timber as a function of tree diameter. The reader is encouraged to implement the above piece of code for the model

$$\text{Volume} = b_0 + b_1 \times \text{Girth} + b_2 \times \text{Height} + \epsilon.$$

An important I would like to stress here that, whenever we run a regression model (here using the `lm()` function), many matrix multiplications take place at the background and there is computational task involved it. Sometimes, depending on the implementation of the algorithm, it may take longer time to run the codes. We observed that for implementation of the LOOCV, we had to execute the `lm()` function code 31 times, or in general, it is the number of rows in the data. Imagine that if there were 10^5 rows in the data with several predictors, running the code several times, would be extremely difficult task. Therefore, LOOCV may not always computationally handy.

Therefore, we need to devise some mechanism, by which we need to implement a smaller number of runs like train-test process and at the same time we can ensure that all the data points are ensured to be the part of both set of training and test observations. The method, called, K-fold validation ease this task. Suppose that we have data set with 100 rows, and we want to implement a five-fold cross validation. In this process, we divide the data into five approximately equal segments. Before writing the codes, we keep a note of the following points:

- LOOCV is a special case of the k-fold cross validation, where k is chosen to be equal to the number of observations.
- LOOCV is time consuming due the number of experiments performed, if the dataset is large enough.
- With k-fold cross-validation, the computation time is reduced due to the number of experiments performed is less as well as the all the observations are eventually used to train and test the model.

Similar to the LOOCV, one can write a simple `for loop` to execute the task. However, we will use a simple but useful package in R. We will use the `caret` package to complete this task. The function `train()` in the `caret` package offers several options for model training. In the `trainConrol` option, we specify the number of folds.

```
#install.packages("caret", dependencies = TRUE)
library(caret)                                    # load the library
```

```
Loading required package: ggplot2
```

```
Loading required package: lattice
```

```
out = train(Volume ~ Girth, data = trees,
            method = "lm",
            trControl = trainControl(method = "cv", number = 5))
print(out$results[2])                          # RMSE
```

```
      RMSE
1 4.495616
```

```
out2 = train(Volume ~ Girth + I(Girth^2), data = trees,
            method = "lm",
            trControl = trainControl(method = "cv", number = 5))
print(out2$results[2])                           # RMSE
```

```
      RMSE
1 3.512168
```

From the output it is found that the 5-fold cross validation root mean squared errors are 4.45317 and 3.264002 for the linear and quadratic regression, respectively. Therefore, by using the K-fold cross validation, we consider the quadratic regression model as a better predictive model for the volume of timber. We would like to emphasize that the choice of the folds is randomly done. Therefore, if we run the above code once again, the values might be changed. Therefore, we may repeat this process a certain number of times and report the average K-fold cross validation error and corresponding standard error value. This can be carried out by fixing the `repeats` option in the `trainControl` argument. In the following, we just provide the updated code:

```r
out = train(Volume ~ Girth, data = trees,
            method = "lm",
            trControl = trainControl(method = "repeatedcv",
                                     number = 5, repeats = 100))
print(out$results[2])                  # linear regression
```

```
      RMSE
1 4.403799
```

```r
cat("The estimated RMSE is ", as.numeric(out$results[2]),
    " with standard error ", as.numeric(out$results[5]))
```

```
The estimated RMSE is  4.403799  with standard error  1.093214
```

```r
out2 = train(Volume ~ Girth + I(Girth^2), data = trees,
             method = "lm",
             trControl = trainControl(method = "repeatedcv",
                                      number = 5, repeats = 100))
print(out$results[2])                   # quadratic regression
```

```
      RMSE
1 4.403799
```

```r
cat("The estimated RMSE is ", as.numeric(out2$results[2]),
    " with standard error ", as.numeric(out2$results[5]))
```

```
The estimated RMSE is  3.416458  with standard error  0.7015495
```

References

Boore, D. M., and W. B. Joyner. 1982. "The Empirical Prediction of Ground Motion." *Bulletin of the Seismological Society of America* 72: S269–68.

Casella, George, and Roger L. Berger. 2002. *Statistical Inference*. 2nd ed. Pacific Grove, CA: Duxbury.

Ezekiel, Mordecai. 1930. *Methods of Correlation Analysis*. Wiley.

Fox, John. 2016. *Applied Regression Analysis and Generalized Linear Models*. Third. Sage.

Levene, Howard. 1960. "Robust Tests for Equality of Variances." In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, edited by Ingram Olkin, 278–92. Stanford University Press.

Rohatgi, Vijay K., and A. K. Md. Ehsanes Saleh. 2000. *An Introduction to Probability Theory and Statistics*. 2nd ed. New York: John Wiley & Sons.

Royston, Patrick. 1982. "An Extension of Shapiro and Wilk's W Test for Normality to Large Samples." *Applied Statistics* 31: 115–24. https://doi.org/10.2307/2347973.

———. 1995. "Remark AS R94: A Remark on Algorithm AS 181: The W Test for Normality." *Applied Statistics* 44: 547–51. https://doi.org/10.2307/2986146.