STA 141
Homework 1

by

Jiewei Chen

999 494 235

# Problem 1

```
# Markov chain: Write a function to generate a sequence of random variables {Xi}
#             taking values 0 and 1, following the Markov Chain probability model

# function name - Markov
# Two arguments - n is the length of the sequence generated
#                         - P00 is the probability of the case X(i+1) = 0 when Xi = 0
#                         - P10 is the probability of the case X(i+1) = 0 when Xi = 1

# Output is a vector containing the Markov Chain

Markov <- function (n, P00, P10) {

            if (P00 < 0 || P00 > 1 || P10 < 0 || P10 > 1) {
                        stop("Probability cannot be negative or larger than 1!")
            }

            if (n <= 0 || !identical(round(n), n) ) {
                        stop("n must be a positive integer!")
            }

            P01 = 1 - P00
            P11 = 1 - P10
            M = rep(0, n)
            M[1] = sample(c(0,1), size = 1, prob = c(0.5, 0.5))

            for (i in 2:n) {

                        if (M[i-1] == 1) {
                                    M[i] = sample(c(0,1), replace = TRUE, size = 1, prob =
c(P10, P11))
                        }
                        else if (M[i-1] == 0) {
                                    M[i] = sample(c(0,1), replace = TRUE, size = 1, prob = c(P00,
P01))
                        }
```

```
        }

        return(M)
}
```

# Problem 2

```
# Write a function that takes a sequence (vector) of 0 or 1 as input
#       returns the starting locations of runs of 0’s and runs of 1’s,
#       where the length of a run is set to an integer K ≥ 1

# function name - Loc_0and1
# Two arguments - X is the input vector, which contains 0's or 1's
#                 - K is the length of a run
# Output is a list of length two
#       First element is a vector states all the starting locations of 0's
#       Second element is a vector states all the starting locations of 1's
#       If no run is found, return 0 as the location.

Loc_0and1 <- function(X, K) {

        if (sum(X %in% c(0,1)) != length(X)) {
                stop("X can only contain either 0 or 1!")
        }
        if (K > length(X)) {
                stop("K must be smaller or equal to the length of input vector!")
        }

        # Initializing a list that can store the result
        location = list("Location of Runs of 0's" = 0, "Location of Runs of 1's" = 0)

        j = 0
        for (i in 1: (length(X)-K+1) ) {
                if (X[i] == 0) {
                        check = ( X[i:(i+K-1)] == 0)
                        if (sum(check) == K) {
                                j = j + 1
```

```
                                    location[[1]][j] = i
                                    }
                  }
            }

            l = 0
            for (i in 1: (length(X)-K+1) ) {
                  if (X[i] == 1) {
                        check = ( X[i:(i+K-1)] == 1)
                        if (sum(check) == K) {
                              l = l + 1
                              location[[2]][l] = i
                        }
                  }
            }

            return(location)
}
```

# Problem 3

```
# Write a function that takes a sequence (vector) of 0 or 1 as input
#           and returns the starting location of all subsequences
#           that start and end with prespecified motifs (vectors of 0's and 1's).

# function name - Loc
# Three arguments - X is the input vector, which contains 0's or 1's
#                       - S is the starting motif
#                       - E is the ending motif
# Output is a data frame containing the starting/ending locations and
#           the length of all subsequences that start and end with
#           prespecified motifs.
#           First column is the starting points.
#           Second column is the starting points.
#           Third column is the length of subsequence.
```

```
#		If no run is found, return 0 as the location.

Loc <- function(X, S, E) {
	# if starting or ending motif is longer than the input sequence
	#		return an error message
	if ( min(length(S), length(E)) > length(X) ) {
		stop("The starting or ending motif should be smaller or as equal to 
the input vector")
	}
	# Initializing the output
	location = data.frame("Start Loc" = 0, "End Loc" = 0, "Length" = 0)
	# Initializing a vector that contains all the starting locations of Starting Motif
	Loc_S = numeric(0)
	j = 0
	for (i in 1: (length(X)-length(S)+1) ) {
		if ( identical(X[i:(i+length(S)-1)],S) ) {
			j = j + 1
			Loc_S[j] = i
		}
	}

	# Initializing a vector that contains all the starting locations of Ending Motif
	Loc_E = numeric(0)
	l = 0
	for (i in 1: (length(X)-length(E)+1) ) {
		if ( identical(X[i:(i+length(E)-1)],E) ) {
			l= l + 1
			Loc_E[l] = i
		}
	}

	# Now we have all the starting locations where we can find the Starting or 
Ending Motif
	ind = 1 # index of location
	for (i in 1:length(Loc_S) ) {
		for (j in 1: length(Loc_E)) {
			if ( Loc_E[j] >= Loc_S[i] && ((Loc_E[j]+length(E)-1) >= 
(Loc_S[i]+length(S)-1)) ) {
				location[ind,1] = Loc_S[i]
```

```
                                        location[ind,2] = Loc_E[j]
                                        location[ind,3] = Loc_E[j] + length(E) - Loc_S[i]
                                        ind = ind + 1
                        }

                        }
          }
          return(location)
}
```

# Problem 4

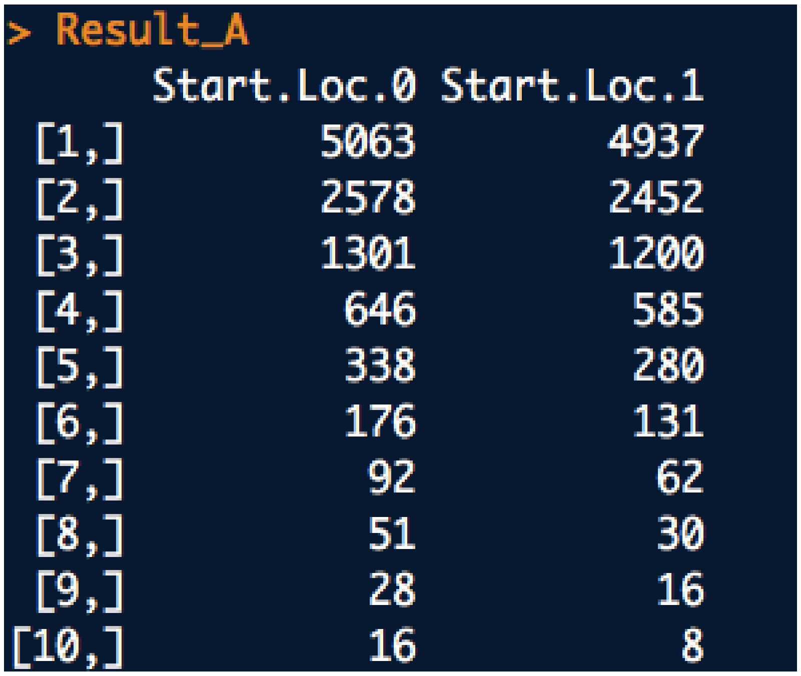
(Codes are marked)

```
# Markov Chain a
#          π00 = 0.5     π01 = 0.5
#          π10 = 0.5     π11 = 0.5

Chain_A = Markov(10000, 0.5, 0.5)

# Markov Chain b
#          π00 = 0.8     π01 = 0.2
#          π10 = 0.1     π11 = 0.9

Chain_B = Markov(10000, 0.8, 0.1)

# ========================
```

4a

```
# For Chain a, the starting location of runs of 0's and 1's of lengths up to 10
A = sapply( 1:10, function(i) {Loc_0and1(Chain_A, i)} )
# Answer is a list which contains the result of the starting location of runs
#          of 0's and 1's of lengths up to 10.
# To better present the result, I created a matrix that contains the length of
#          the vectors of the starting location.The i(th) row represents the number of
#          starting locations can be found in the runs of 0's or 1's with length i.
```

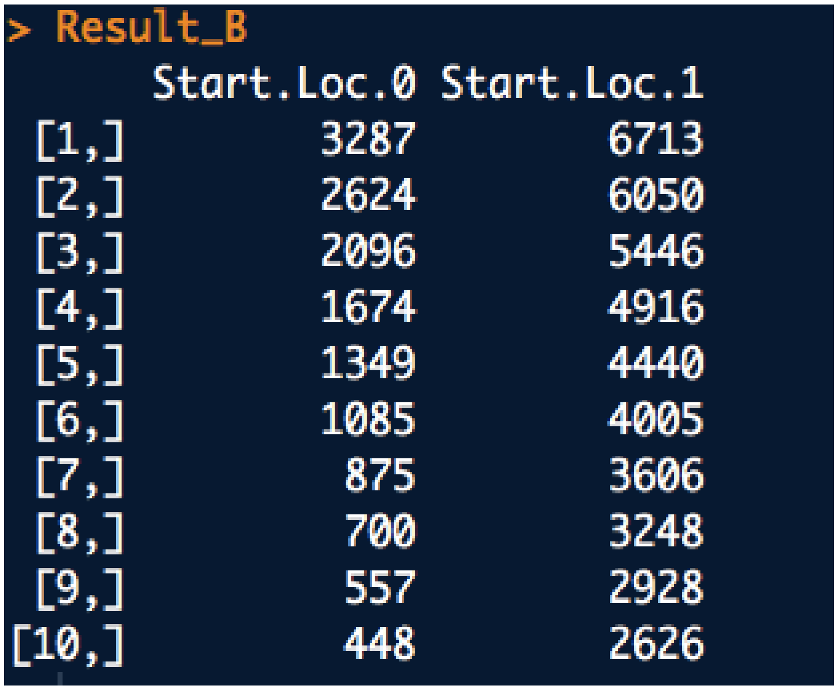
```
Start.Loc.0 = rep(0, 10)
Start.Loc.1 = rep(0, 10)
for (i in 1:10) {
        Start.Loc.0[i] = length( A[,i][[1]] )
        Start.Loc.1[i] = length( A[,i][[2]] )
        Result_A = cbind(Start.Loc.0, Start.Loc.1)
}
Result_A
# Answer
#       Start.Loc.0 Start.Loc.1
#   [1,]       5063        4937
#   [2,]       2578        2452
#   [3,]       1301        1200
#   [4,]        646         585
#   [5,]        338         280
#   [6,]        176         131
#   [7,]         92          62
#   [8,]         51          30
#   [9,]         28          16
#   [10,]        16           8
```



```
> Result_A
      Start.Loc.0 Start.Loc.1
 [1,]        5063        4937
 [2,]        2578        2452
 [3,]        1301        1200
 [4,]         646         585
 [5,]         338         280
 [6,]         176         131
 [7,]          92          62
 [8,]          51          30
 [9,]          28          16
[10,]          16           8
```

# For Chain b, the starting location of runs of 0’s and 1’s of lengths up to 10

```
B = sapply( 1:10, function(i) {Loc_0and1(Chain_B, i)} )
# Answer is a list which contains the result of the starting location of runs
#           of 0’s and 1’s of lengths up to 10.
# To better present the result, I created a matrix that contains the length of
#           the vectors of the starting location. The i(th) row represents the number of
#           starting locations can be found in the runs of 0's or 1's with length i.
Start.Loc.0 = rep(0, 10)
Start.Loc.1 = rep(0, 10)
for (i in 1:10) {
            Start.Loc.0[i] = length( B[,i][[1]] )
            Start.Loc.1[i] = length( B[,i][[2]] )
            Result_B = cbind(Start.Loc.0, Start.Loc.1)
}
Result_B
# Answer
#          Start.Loc.0    Start.Loc.1
#   [1,]          3287           6713
#   [2,]          2624           6050
#   [3,]          2096           5446
#   [4,]          1674           4916
#   [5,]          1349           4440
#   [6,]          1085           4005
#   [7,]           875           3606
#   [8,]           700           3248
#   [9,]           557           2928
#   [10,]          448           2626
```



```
> Result_B
      Start.Loc.0 Start.Loc.1
 [1,]        3287        6713
 [2,]        2624        6050
 [3,]        2096        5446
 [4,]        1674        4916
 [5,]        1349        4440
 [6,]        1085        4005
 [7,]         875        3606
 [8,]         700        3248
 [9,]         557        2928
[10,]         448        2626
```

# Findings
# Yes. The patterns between runs of 0’s and runs of 1’s in Chain A and B are quite
#	different.
#	First, for both of them, as the length of the run getting longer, the possible
#	starting locations we can find in Markov chain becomes less. Because the
chance
#	of finding longer consecutive 0 or 1 becomes less.
#	Secondly, by comparing the two matrix I got. It can be observed that for
#	Chain A, the number of runs of 0's and 1's are much smaller than those of
#	Chain B. For example, for Chain A, the number of finding continueing 0's and
1's
#	are 16, and 8, respectively. For Chain B,the number of finding ten continueing
#	0's and 1's are 448, and 2626, respectively.

=======================

4b

# For Chain A
Loc_n (Chain_A, c(0,0,0,0), c(1,1,1,1), 200)
#	for more detail about Function Loc_n, please refer to the code below.
length(Loc_n (Chain_A, c(0,0,0,0), c(1,1,1,1), 200))

```
# The answer is a vector with length 43.
head((Loc_n (Chain_A, c(0,0,0,0), c(1,1,1,1), 200)))
# [1]   421   958 1262 1489 1818 1918
tail(Loc_n (Chain_A, c(0,0,0,0), c(1,1,1,1), 200))
# [1] 8693 8807 9450 9462 9692 9790

# For Chain B
Loc_n (Chain_B, c(0,0,0,0), c(1,1,1,1), 200)
length(Loc_n (Chain_B, c(0,0,0,0), c(1,1,1,1), 200))
# The answer is a vector with length 814.
head((Loc_n (Chain_B, c(0,0,0,0), c(1,1,1,1), 200)))
# [1] 10 11 12 13 14 15
tail(Loc_n (Chain_B, c(0,0,0,0), c(1,1,1,1), 200))
# [1] 9657 9658 9659 9660 9661 9685

# -------------------------
## Code for function Loc_n
# This function that takes a sequence (vector) of 0 or 1 as input and
#         returns the starting location of all subsequences with specified length
#         that start and end with prespecified motifs (vectors of 0's and 1's).

# function name - Loc_n
# Two arguements - X is the input vector, which contains 0's or 1's
#                  - S is the starting motif
#                  - E is the ending motif
#                  - n is length of the subsequence required
# Output is a numeric vector containing the starting location of
#         all subsequences that start and end with prespecified motifs
#         If no run is found, return 0 as the location.

# Loc_n <- function(X, S, E, n) {
#          # if starting or ending motif is longer than the input sequence
#          #         return an error message
#          if ( min(length(S), length(E)) > length(X) ) {
#                  stop("The starting or ending motif should be smaller or as equal to the input
vector")
#          }
#
#          location = numeric(0)
#	j = 0
#	# subsetting the vector with length n
#	for (i in 1: (length(X)-n+1) ) {
#		# subvector with length n
#		Y = X[i:(i+n-1)]
#		# subsetting the start motif
#		W = Y[1:length(S)]
#		# subsetting the last motif
#		Z = Y[(n-length(E)+1):n]
#		check1 = ( W == S )
#		check2 = ( Z == E )
#		if (sum(check1) == length(S) && sum(check2) == length(E)) {
#			j = j + 1
#			location[j] = i
#		}
#	}
#	return(sort(location))
# }
# -------------------------
# ========================
```

4c

```
# It can be observed that for Chain A, the number of runs of 0's and 1's are
#	much smaller than those of Chain B. For Chain A, the number of finding
#	ten continueing 0's and 1's are 16, and 8, respectively. For Chain B,
#	the number of finding ten continueing 0's and 1's are 448, and 2626,
#	respectively. I think this difference is due to the difference in the
#	probabilities of these two Markov Chains. In Chain A, the probability
#	of getting 1 or 0 is the same, no matter what the previous number is.
#	So it behaves like random sequence, of which the probability of getting
#	1's and 0's is the same at a given position. However, for Chain B, the
#	probabilities of getting 1's and 0's are dependent on the previous number
#	in the sequence. According to the transition probability matrices of B,
#	there is much higher chance to get 1 right after 1, or get 0 right after 0.
#	This is why we could find much more consecutively 0's or 1's from Chain B.
#
```

Problem 5

i) AirPassengers

The graph I can generate from these data set is below.

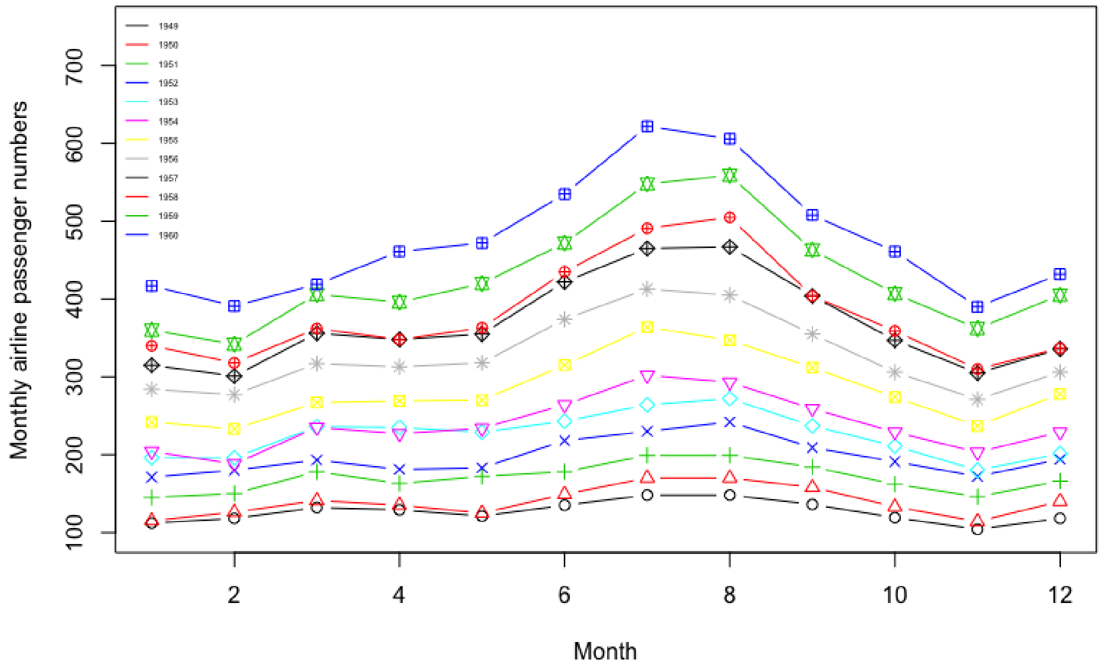


Figure 1-1. The monthly airline passenger numbers from 1949 to 1960.

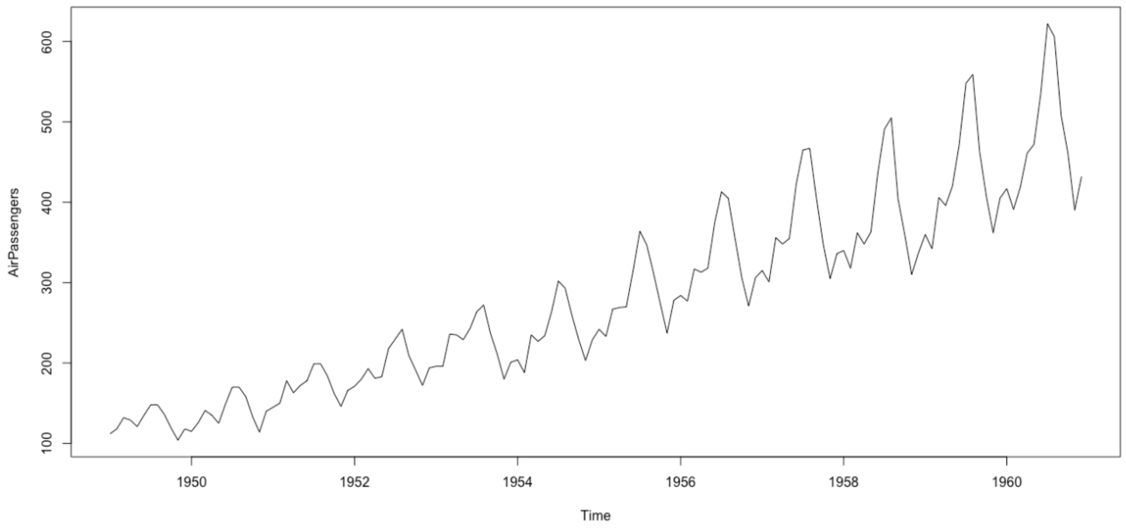


Figure 1-2. The monthly airline passenger numbers from 1949 to 1960.

<u>Brief Description</u>

This “AirPassengers” dataset is a time-series object, containing the monthly airline passenger numbers from 1949 to 1960. From Fig. 1-1, it is observed that, for each year, in general, the passenger number is decreasing in February, followed by a continuous increase in the next five months. The passenger number reaches its peak at July, and then decrease again. The two peak value can be found in Jan and July. The demand for airline each year is increasing, according to Fig. 1-2.

```
# ---------------R Code for generating the plot or getting parameters for the dataset-----------------#
# Figure 1-1
j = 0
Y = matrix( rep(0,12*12), 12, 12)
for (i in 1948:1960) {
            Y[j,] = AirPassengers[ time(AirPassengers) >= i & time(AirPassengers) < (i+1)]
            j = j + 1
}
plot(Y[1,], xlab = "Month", ylab = "Monthly airline passenger numbers",
       type = "b", pch = 1,col= 1, ylim = c(100,750))
for (i in 2:12) {
            points(Y[i,], type = "b", pch = i, col = i)
}
legend("topleft", legend = c(1949:1960), lty=rep(1,12), col=c(1:12), bty = "n", cex = 0.4)

# Figure 1-2
plot(AirPassengers)

# Parameters
class(AirPassengers) # "ts"
typeof(AirPassengers) # "double"
length(AirPassengers) # 144
# ---------------R Code for generating the plot or getting parameters for the dataset-----------------#
```

ii) EuStockMarkets

The graph I can generate from these data set is below.

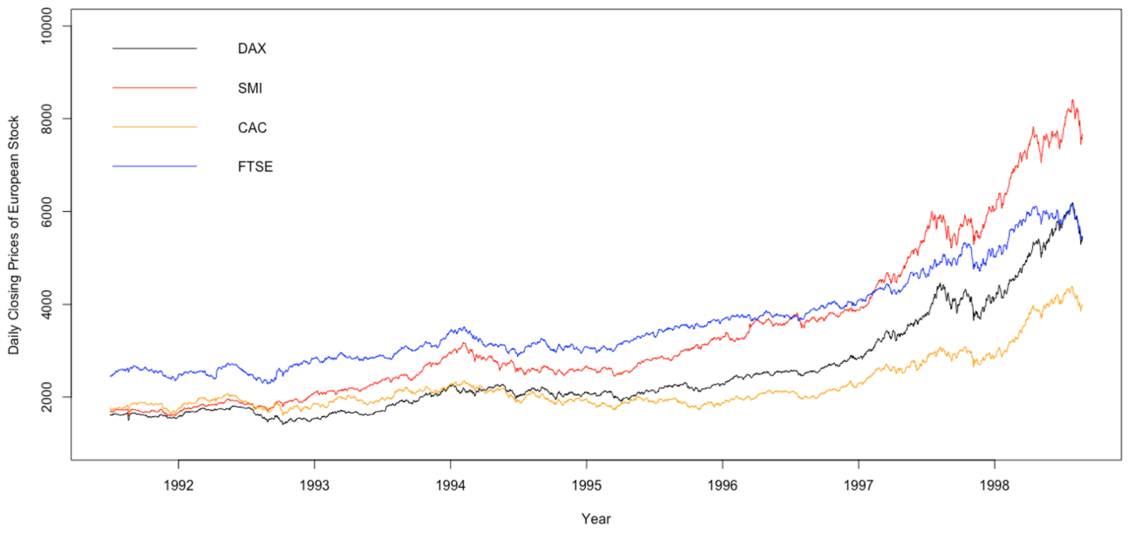


Figure 2-1. The daily closing prices of major European stock Indices from 1991–1998

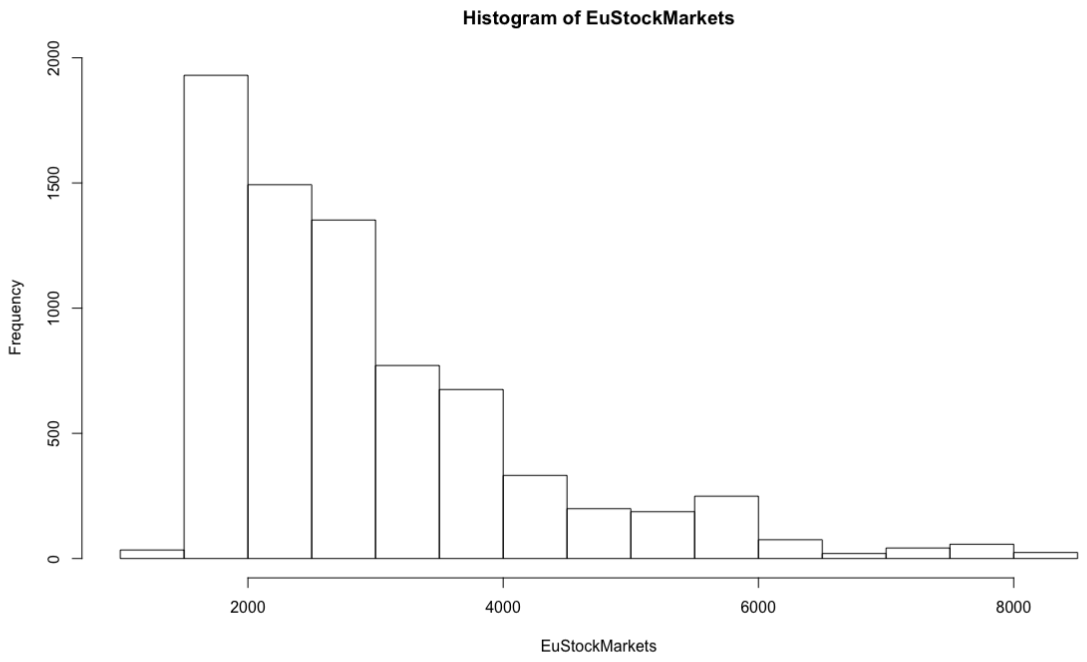


Figure 2-2. The histogram of daily closing prices of major European stock Indices from 1991–1998.

<u>Brief Description</u>

This “EuStockMarkets” dataset is a matrix with four columns, each of which represents a stock index from 1991 to 1998 by a time-series object. From Fig. 2-1, it is observed that, from 1991 to 1997, the stock index is slowly increasing with some up-and-downs around 1994. Starting from 1997, there is a steep increase in all stocks, among which SMI increases the most. High frequency of stock index falls into the range of 1,500 to 3,000.

```
# ---------------R Code for generating the plot or getting parameters for the dataset-----------------#
# Figure 2-1
DAX = EuStockMarkets[,1]
SMI = EuStockMarkets[,2]
CAC = EuStockMarkets[,3]
FTSE = EuStockMarkets[,4]
plot(DAX, xlab = "Year", ylab = "Daily Closing Prices of European Stock",
      type="l",col="black", ylim = c(1000,10000))
points(SMI, type = "l", col = "Red")
points(CAC, type = "l", col = "Orange")
points(FTSE, type = "l", col = "Blue")
legend("topleft", legend = c("DAX", "SMI", "CAC", "FTSE"), lty=c(1,1,1,1),
        col=c("black", "red", "orange", "blue"), bty='n')
# Figure 2-2
hist(EuStockMarkets)
# Parameters
class(EuStockMarkets) # "mts"     "ts"      "matrix"
head(EuStockMarkets)
# ---------------R Code for generating the plot or getting parameters for the dataset-----------------#
```

iii) PlantGrowths

The graph I can generate from these data set is below.

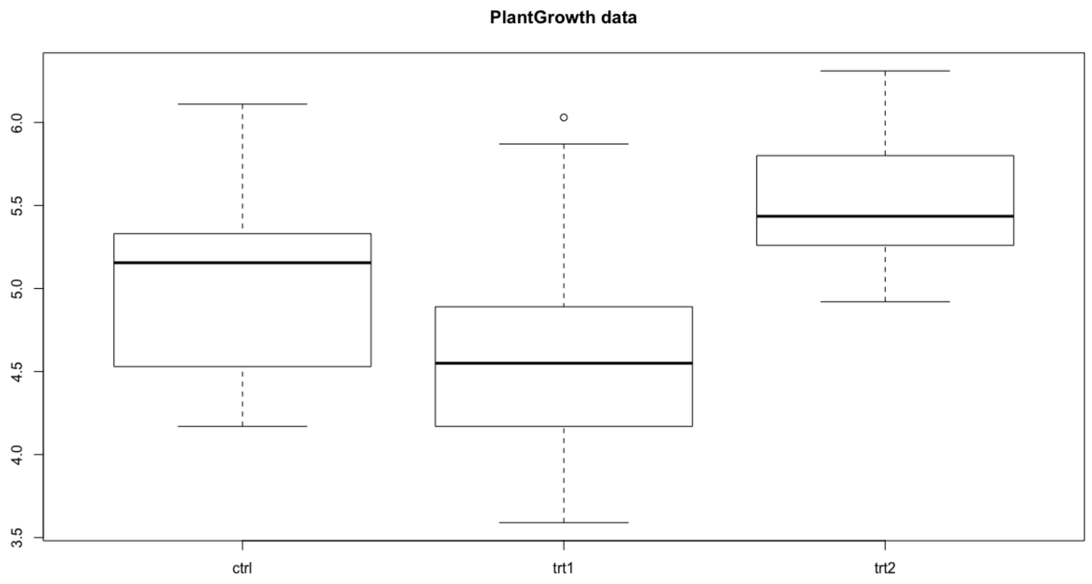


Figure 3. The box plot for comparing yields (as measured by dried weight of plants) obtained under a control and two different treatment conditions

Brief Description

This "PlantGrowth" dataset is a data frame with two columns, one represents the yields, the other is factors stating if it is "control" or "treated" group. From Fig. 3, it can be observed by comparing the median of each group that, **trt1** seems to result in less yields, whereas **trt2** tends to lead to high yields. Although, the median of **trt2** is only slightly higher than that of ctrl, the majority of **trt2** has higher yields than **ctrl**.

```
# ---------------R Code for generating the plot or getting parameters for the dataset-----------------#
# Figure 3
boxplot(weight ~ group, data = PlantGrowth, main = "PlantGrowth data")

# Parameters
class(PlantGrowth) # "data.frame"
# ---------------R Code for generating the plot or getting parameters for the dataset-----------------#
```

iv) trees

The graph I can generate from these data set is below.

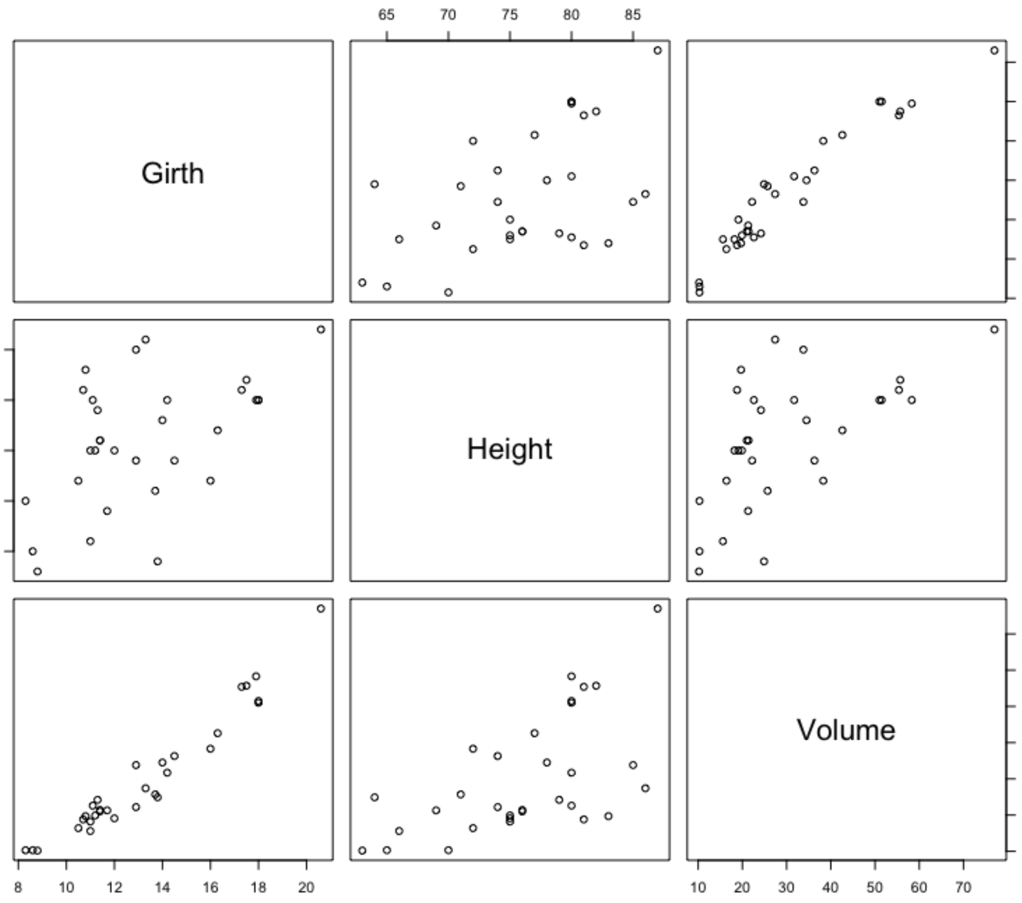


Figure 4-1. The plot of correlations among Girth, Height, and Volume of timber in 31 felled black cherry trees

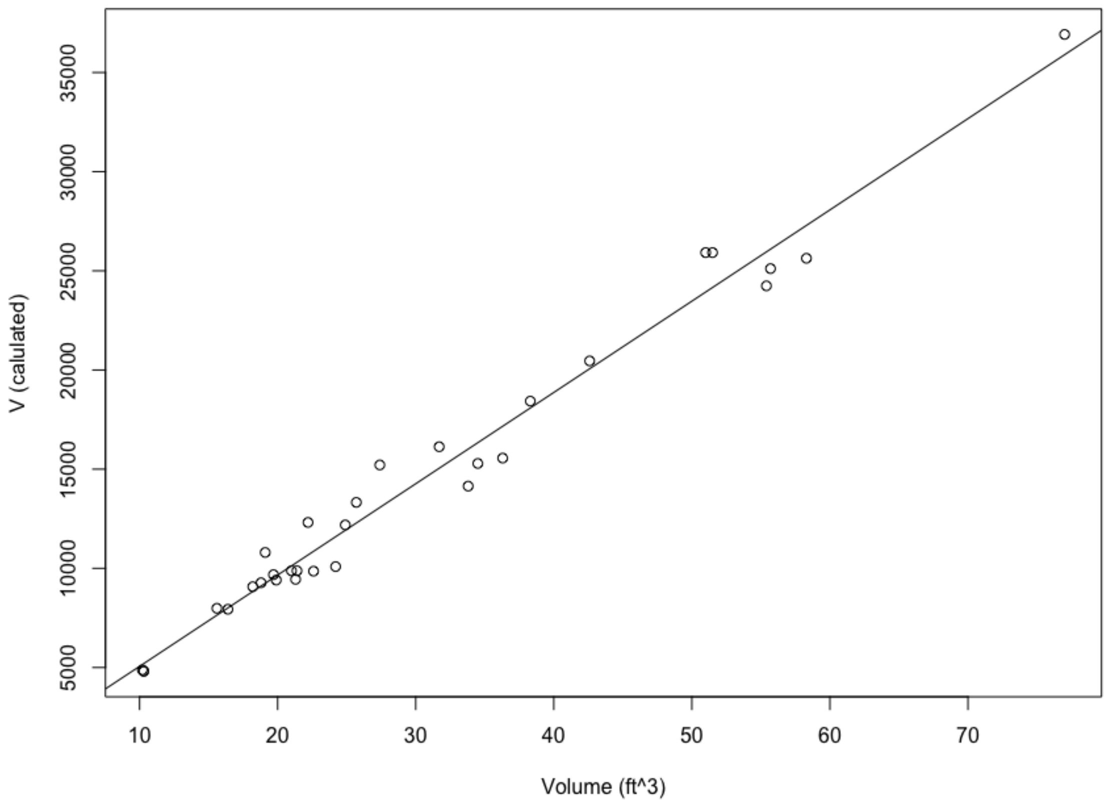


Figure 4-2. The plot of correlations between (Girth^2*Height) and Volume of timber in 31 felled black cherry trees

<u>Brief Description</u>
This “trees” dataset is a data frame with three columns, represents the girth, height, and volume of 31 trees. It can be observed from Fig. 4-1 that, there is positive correlation between Volume vs. Height, and Volume vs. Girth. When plotting (Girth^2*Height) vs. Volume, it is found that there is a linear relation between them, indicating that the estimation of Girth^2*Height as volume is reasonable.

```
# --------------R Code for generating the plot or getting parameters for the dataset------------------#
# Figure 4-1
plot(trees)
# Figure 4-2
par(mfrow = c(1,1))
plot(Girth^2*Height ~ Volume, data = trees, xlab = "Volume (ft^3)", ylab = "V (calulated)")
```

```
Model = lm(Girth^2*Height ~ Volume, data = trees)
abline(Model)
# Parameters
class(trees)
dim(trees)
# ---------------R Code for generating the plot or getting parameters for the dataset-------------------#
```

v) airquality

The graph I can generate from these data set is below.

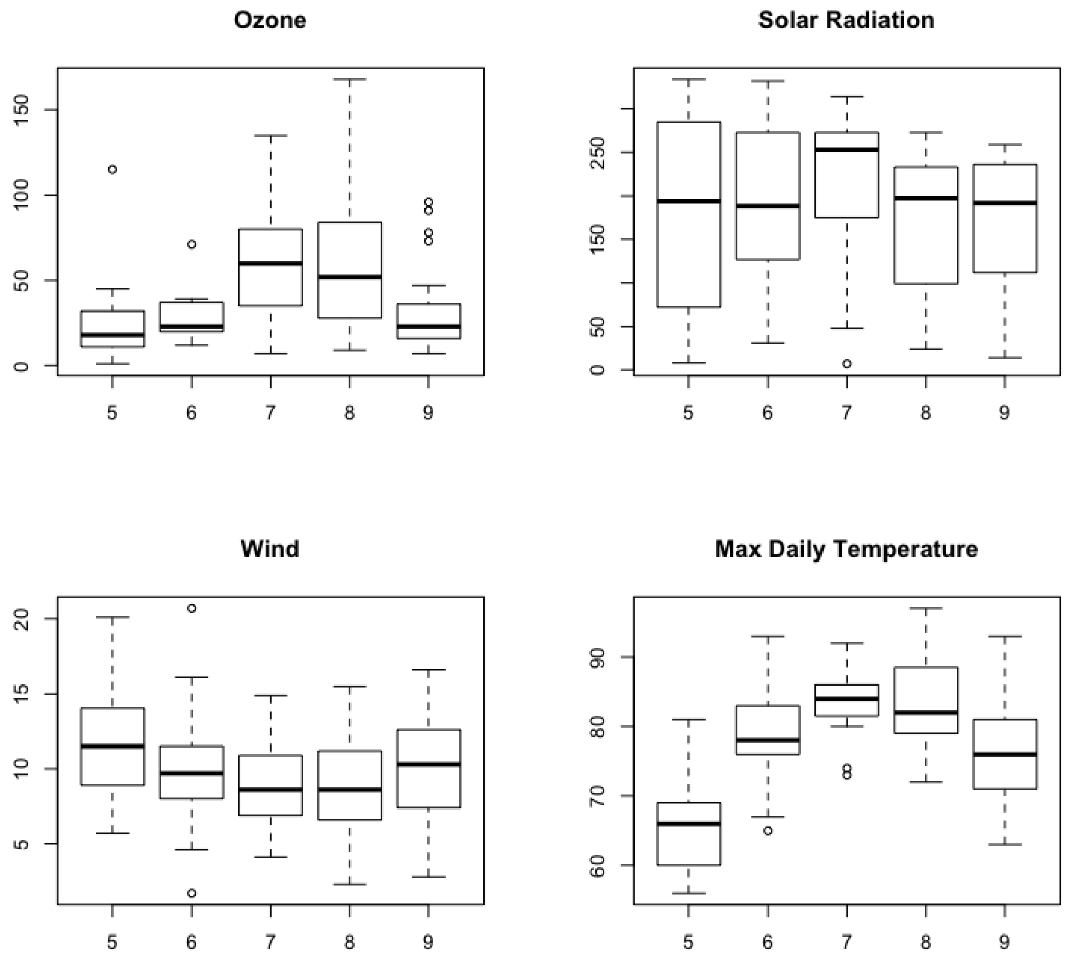


Figure 4-1. The box plot of daily air quality parameters (Ozone, Solar Radiation, Wind and Temperature) from May to September.

Brief Description

This "airquality" dataset is a data frame, which contains daily air quality measurements in New York from May to September 1973. The median of Ozone was slightly higher in July and August. Especially in August, the Ozone varied a lot, with the highest value 168 and the lowest value 1. The median of Solar Radiation and Wind didn't change much as Ozone did. The Max Temperature varied with the month, and reaches highest in August. The temperature of July is almost over 80 every day.

```
# ---------------R Code for generating the plot or getting parameters for the dataset-------------------#
# Figure 5
par(mfrow = c(2, 2))
boxplot(Ozone ~ Month, data = airquality, main = "Ozone")
boxplot(Solar.R ~ Month, data = airquality, main = "Solar Radiation")
boxplot(Wind ~ Month, data = airquality, main = "Wind")
boxplot(Temp ~ Month, data = airquality, main = "Max Daily Temperature")
# Parameters
class(airquality) # "data.frame"
dim(airquality) # 153      6
head(airquality)
#      Ozone Solar.R Wind Temp Month Day
# 1       41         190    7.4      67          5      1
# 2       36         118    8.0      72          5      2
# 3       12         149 12.6      74          5      3
# 4       18         313 11.5      62          5      4
# 5       NA            NA 14.3      56          5      5
# 6       28            NA 14.9      66          5      6
# ---------------R Code for generating the plot or getting parameters for the dataset-------------------#
```