Getting Started with R

Chapter 1 : What is R?

–> A free, open source intergrated development environment of IDE for R (the statistical programming Language)

Create a vector

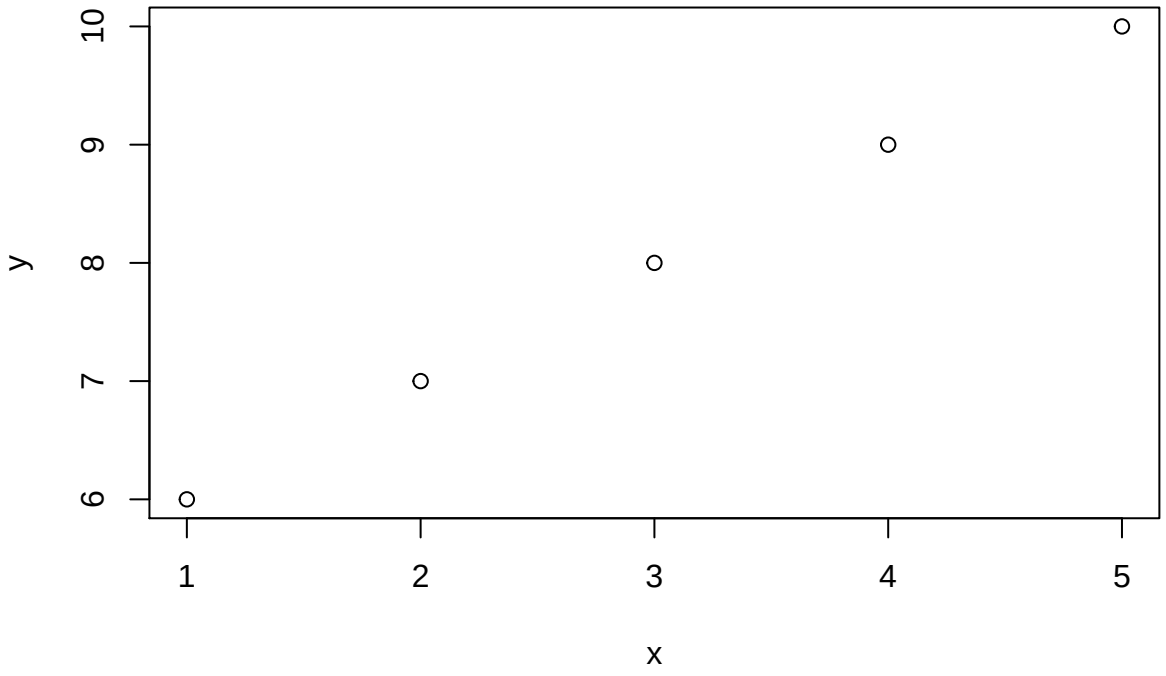
```
x <- 1:5
x
```

```
## [1] 1 2 3 4 5
```

```
y <- 6:10
y
```

```
## [1]  6  7  8  9 10
```

```
plot(x,y)
```




```
ls()
```

```
## [1] "x" "y"
```

##Installing R

Chapter 2: Arithmetic and Coding

```
x <- 11
print(x)
```

```
## [1] 11
```

NOTE: R is case sensitive

```
y <- 7
y
```

```
## [1] 7
```

ls() command to check what is stored in workspace

```
ls()
```

```
## [1] "x" "y"
```

rm() remove objects from workspace

```
rm(y)
```

```
rm(x)
```

```
ls()
```

```
## character(0)
```

```
xx <- "vikas"
xx
```

```
## [1] "vikas"
```

```
yy <- "1" # this is not a number
yy
```

```
## [1] "1"
```

Arithmatic Operations

```
10+14
```

```
## [1] 24
```

```
7*9
```

```
## [1] 63
```

```
a <- 10
```

```
b <- 12
```

```
 a+b
```

```
## [1] 22
```

```r
c <- a+b
c
```

```
## [1] 22
```

```r
a/b
```

```
## [1] 0.8333333
```

```r
a^2
```

```
## [1] 100
```

```r
sqrt(a)
```

```
## [1] 3.162278
```

```r
log(a)
```

```
## [1] 2.302585
```

```r
exp(b)
```

```
## [1] 162754.8
```

```r
log2(b)
```

```
## [1] 3.584963
```

```r
abs(-14)
```

```
## [1] 14
```

Chapter 3: Creating Vectors, Matrics, and Performing simple operation on them

Create a vector using "c" or concatenate command

```
x1 <- c(1,3,5,7,9)
x1
```

```
## [1] 1 3 5 7 9
```

```
gender <- c("male","female")
gender
```

```
## [1] "male"   "female"
```

```
2:7
```

```
## [1] 2 3 4 5 6 7
```

```
seq(from=1,to=7,by=1)
```

```
## [1] 1 2 3 4 5 6 7
```

```
seq(from=1,to=7,by=1/3)
```

```
##  [1] 1.000000 1.333333 1.666667 2.000000 2.333333 2.666667 3.000000
##  [8] 3.333333 3.666667 4.000000 4.333333 4.666667 5.000000 5.333333
## [15] 5.666667 6.000000 6.333333 6.666667 7.000000
```

```
seq(from=1,to=7,by=0.25)
```

```
##  [1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25
## [15] 4.50 4.75 5.00 5.25 5.50 5.75 6.00 6.25 6.50 6.75 7.00
```

```
rep(1,times=10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
rep("vikas",times=5)
```

```
## [1] "vikas" "vikas" "vikas" "vikas" "vikas"
```

```
rep(1:3,times=5)
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

```
rep(seq(from=2,to=5,by=0.25),times=5)
```

```
##  [1] 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00 2.00
## [15] 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00 2.00 2.25
## [29] 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00 2.00 2.25 2.50
## [43] 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00 2.00 2.25 2.50 2.75
## [57] 3.00 3.25 3.50 3.75 4.00 4.25 4.50 4.75 5.00
```

```
rep(c("m","f"),times=5)
```

```
##  [1] "m" "f" "m" "f" "m" "f" "m" "f" "m" "f"
```

```
x <- 1:5
x
```

```
## [1] 1 2 3 4 5
```

```
y <- c(1,3,5,7,9)
y
```

```
## [1] 1 3 5 7 9
```

```
x + 10
```

```
## [1] 11 12 13 14 15
```

```
x - 10
```

```
## [1] -9 -8 -7 -6 -5
```

```
x * 10
```

```
## [1] 10 20 30 40 50
```

```
x/2
```

```
## [1] 0.5 1.0 1.5 2.0 2.5
```

If the vectors are of the same length, we may add/substract/mult/div corresponding elements.

```
x
```

```
## [1] 1 2 3 4 5
```

```
y
```

```
## [1] 1 3 5 7 9
```

Extracting elements

Extract only 3rd elements from the vectors

```
y[3]
```

```
## [1] 5
```

Extract all elements expcept 3rd element.

```
y[-3]
```

```
## [1] 1 3 7 9
```

```
y[c(1,5)]
```

```
## [1] 1 9
```

```
y[-c(1,5)]
```

```
## [1] 3 5 7
```

```
y[y<6]
```

```
## [1] 1 3 5
```

Creating Matrics

```
matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3,byrow = TRUE) # rowwise fashion
```

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

```
matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3,byrow = FALSE) # columnwise fashion
```

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

```
mat <- matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3,byrow = TRUE) # rowwise fashion
```

Extracting from matrix

```
mat
```

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

```
mat[1,2]
```

```
## [1] 2
```

```
mat[c(1,3),2]
```

```
## [1] 2 8
```

```
mat[2,] ## before comma row comes in
```

```
## [1] 4 5 6
```

```
mat[,1] # after comma columns comes in
```

```
## [1] 1 4 7
```

Chapter 4: Import Data

```
#data1 <- read.csv(file.choose(),header=T,sep = "\t")
#head(data1)
```

```
data2 <- read.csv("LungCapData.txt",header = T,sep="\t")
head(data2)
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 1   6.475   6   62.1    no   male        no
## 2  10.125  18   74.7   yes female        no
## 3   9.550  16   69.7    no female       yes
## 4  11.125  14   71.0    no   male        no
## 5   4.800   5   56.9    no   male        no
## 6   6.225  11   58.7    no female        no
```

```
data3 <- read.table("LungCapData.txt",header = T,sep = "\t")
head(data3)
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 1   6.475   6   62.1    no   male        no
## 2  10.125  18   74.7   yes female        no
## 3   9.550  16   69.7    no female       yes
## 4  11.125  14   71.0    no   male        no
## 5   4.800   5   56.9    no   male        no
## 6   6.225  11   58.7    no female        no
```

Chapter 5: Import Data from Excel file

install.packages("readxl")

```
library(readxl)
meat <- read_excel("meat.xls", sheet = "Sheet2")
head(meat)
```

```
## # A tibble: 6 x 5
##     DEF    IN    PB    PL    PP
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  121.   355  25.7  20.0  45.7
## 2  124.   380  26.3  18.7  53
## 3  132.   426  30.2  22.5  57.9
## 4  119.   353  33.1  27.1  56.6
## 5  120.   354  31.6  20.7  55.2
## 6  115.   361  30.2  18.9  55.7
```

```
meat2 <- read_excel("meat.xls", sheet = "Sheet1")
head(meat2)
```

```
## # A tibble: 6 x 5
##      QB    IN    PB    PL    PP
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  121.   355  25.7  20.0  45.7
## 2  124.   380  26.3  18.7  53
## 3  132.   426  30.2  22.5  57.9
## 4  119.   353  33.1  27.1  56.6
## 5  120.   354  31.6  20.7  55.2
## 6  115.   361  30.2  18.9  55.7
```

Chapter 6 : Export data from R

```
write.table(mat,file = "mat.csv",sep = ",")
```

```
write.csv(meat,file = "meat.csv",row.names = FALSE)
```

Chapter 7 : Importing , Checking and Working with Data

Reading or Importing data

```
LungcapData <- read.table("LungCapData.txt",header = T,sep = "\t")
```

```
#LungCapData2 <- read.table(file.choose(),header = T,sep = "\t")
```

```
dim(LungcapData)
```

```
## [1] 725   6
```

```
head(LungcapData)
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 1   6.475   6   62.1    no   male        no
## 2  10.125  18   74.7   yes female        no
## 3   9.550  16   69.7    no female       yes
## 4  11.125  14   71.0    no   male        no
## 5   4.800   5   56.9    no   male        no
## 6   6.225  11   58.7    no female        no
```

```
tail(LungcapData)
```

```
##     LungCap Age Height Smoke Gender Caesarean
## 720   7.325   9   66.3    no   male        no
## 721   5.725   9   56.0    no female        no
## 722   9.050  18   72.0   yes   male       yes
## 723   3.850  11   60.5   yes female        no
## 724   9.825  15   64.9    no female        no
## 725   7.100  10   67.7    no   male        no
```

```
LungcapData[c(5,6,7,8,9),]
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 5   4.800   5   56.9    no   male        no
## 6   6.225  11   58.7    no female        no
## 7   4.950   8   63.3    no   male       yes
## 8   7.325  11   70.4    no   male        no
## 9   8.875  15   70.5    no   male        no
```

```
LungcapData[5:9,]
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 5   4.800   5   56.9    no   male        no
## 6   6.225  11   58.7    no female        no
## 7   4.950   8   63.3    no   male       yes
## 8   7.325  11   70.4    no   male        no
## 9   8.875  15   70.5    no   male        no
```

```
head(LungcapData[-c(4:722)])
```

```
##   LungCap Age Height
## 1   6.475   6   62.1
## 2  10.125  18   74.7
## 3   9.550  16   69.7
## 4  11.125  14   71.0
```

```
## 5   4.800   5   56.9
## 6   6.225  11   58.7
```

Chapter 8 : Working with Variables and Data

```
names(LungcapData)
```

```
## [1] "LungCap"   "Age"       "Height"    "Smoke"     "Gender"    "Caesarean"
```

Option 1 : use doller sign to attach variable

```
mean(LungcapData$Age)
```

```
## [1] 12.3269
```

option 2 : attach data into R memory using attach command

```
attach(LungcapData)
```

```
mean(Age)
```

```
## [1] 12.3269
```

Class command to ask R what type of variable is

```
class(LungCap)
```

```
## [1] "numeric"
```

```
  class(Age)
```

```
## [1] "integer"
```

```
class(Height)
```

```
## [1] "numeric"
```

```
class(Smoke)
```

```
## [1] "factor"
```

```
class(Caesarean)
```

```
## [1] "factor"
```

```
levels(Gender)
```

```
## [1] "female" "male"
```

```
summary(LungcapData)
```

```
##     LungCap            Age            Height      Smoke        Gender
##  Min.   : 0.507   Min.   : 3.00   Min.   :45.30   no :648   female:358
##  1st Qu.: 6.150   1st Qu.: 9.00   1st Qu.:59.90   yes: 77   male  :367
##  Median : 8.000   Median :13.00   Median :65.40
##  Mean   : 7.863   Mean   :12.33   Mean   :64.84
##  3rd Qu.: 9.800   3rd Qu.:15.00   3rd Qu.:70.30
##  Max.   :14.675   Max.   :19.00   Max.   :81.80
```

```
##  Caesarean
##  no :561
##  yes:164
##
##
##
##
```

Chapter 9 : Subsetting (Sort/Select) Data

```
dim(LungcapData)
```

```
## [1] 725   6
```

```
length(Age) # Count
```

```
## [1] 725
```

```
Age[11:14]
```

```
## [1] 19 17 12 10
```

```
LungcapData[11:14,]
```

```
##    LungCap Age Height Smoke Gender Caesarean
## 11  11.500  19   76.4    no   male       yes
## 12  10.925  17   71.7    no   male        no
## 13   6.525  12   57.5    no   male        no
## 14   6.000  10   61.1    no female        no
```

```
mean(Age[Gender == "female"])
```

```
## [1] 12.44972
```

```
mean(Age[Gender == "male"])
```

```
## [1] 12.20708
```

```
levels(Gender)
```

```
## [1] "female" "male"
```

```
Femdata <- LungcapData[Gender == "female",]
```

```
maledata <- LungcapData[Gender == "male",]
```

```
dim(Femdata)
```

```
## [1] 358   6
```

```
dim(maledata)
```

```
## [1] 367   6
```

```
summary(Gender)
```

```
## female   male
##    358    367
```

```
Femdata[1:4,]
```

```
##    LungCap Age Height Smoke Gender Caesarean
## 2   10.125  18   74.7   yes female        no
## 3    9.550  16   69.7    no female       yes
## 6    6.225  11   58.7    no female        no
## 14   6.000  10   61.1    no female        no
```

```
MaleOver15 <- LungcapData[Gender == "male" & Age >15,]
```

```r
dim(MaleOver15)
```

```
## [1] 89  6
```

```r
MaleOver15[1:4,]
```

```
##    LungCap Age Height Smoke Gender Caesarean
## 11  11.500  19   76.4    no   male       yes
## 12  10.925  17   71.7    no   male        no
## 23  10.025  16   72.4    no   male        no
## 40  11.325  17   77.7    no   male        no
```

Chapter 10 : Logic Statements (TRUE/FALSE), cbind and rbind Functions

```
Age[1:5]
```

```
## [1]  6 18 16 14  5
```

```
temp <- Age>15
```

```
temp[1:5]
```

```
## [1] FALSE  TRUE  TRUE FALSE FALSE
```

```
temp2 <- as.numeric(Age>15)
temp2[1:5]
```

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

```
LungcapData[1:5,]
```

```
##   LungCap Age Height Smoke Gender Caesarean
## 1   6.475   6   62.1    no   male        no
## 2  10.125  18   74.7   yes female        no
## 3   9.550  16   69.7    no female       yes
## 4  11.125  14   71.0    no   male        no
## 5   4.800   5   56.9    no   male        no
```

```
FemSmoke <- Gender=="female" & Smoke=="yes"
```

```
FemSmoke[1:5]
```

```
## [1] FALSE  TRUE FALSE FALSE FALSE
```

We can attach vectors & Matrices in a column or row wise fashion using cbind command as well as rbind command

```
MoreData <- cbind(LungcapData,FemSmoke)
```

```
MoreData[1:5,]
```

```
##   LungCap Age Height Smoke Gender Caesarean FemSmoke
## 1   6.475   6   62.1    no   male        no    FALSE
## 2  10.125  18   74.7   yes female        no     TRUE
## 3   9.550  16   69.7    no female       yes    FALSE
## 4  11.125  14   71.0    no   male        no    FALSE
## 5   4.800   5   56.9    no   male        no    FALSE
```

removing all objects from workspace

```
#rm(list=ls())
```

Chapter 11 : Setting working directory

```
#setwd(Path)
```

Chapter 12 : Apply Function

Apply functions are a set of loop functions in R

The main difference is that apply functions are more efficient than a 'for loop'

apply functions require less lines of code (less chance for coding error) and are often faster than a simple for loop

```
stockData <- read.csv("StockExample.csv",header = T,sep = ",")
stockData
```

```
##        X Stock1 Stock2 Stock3 Stock4
## 1   Day1 185.74   1.47   1605  95.05
## 2   Day2 184.26   1.56   1580  97.49
## 3   Day3 162.21   1.39   1490  88.57
## 4   Day4 159.04   1.43   1520  85.55
## 5   Day5 164.87   1.42   1550  92.04
## 6   Day6 162.72   1.36   1525  91.70
## 7   Day7 157.89     NA   1495  89.88
## 8   Day8 159.49   1.43   1485  93.17
## 9   Day9 150.22   1.57   1470  90.12
## 10 Day10 151.02   1.54   1510  92.14
```

Calculate the mean price of each stock

```
apply(stockData[,-1],MARGIN = 2,FUN=mean,na.rm=TRUE)
```

```
##      Stock1      Stock2      Stock3      Stock4
##  163.746000    1.463333 1523.000000   91.571000
```

OR

```
colMeans(stockData[,-1],na.rm = TRUE)
```

```
##      Stock1      Stock2      Stock3      Stock4
##  163.746000    1.463333 1523.000000   91.571000
```

Find the MAXIMUM Stock price , for each stock

```
apply(stockData[,-1],MARGIN = 2,FUN=max,na.rm=TRUE)
```

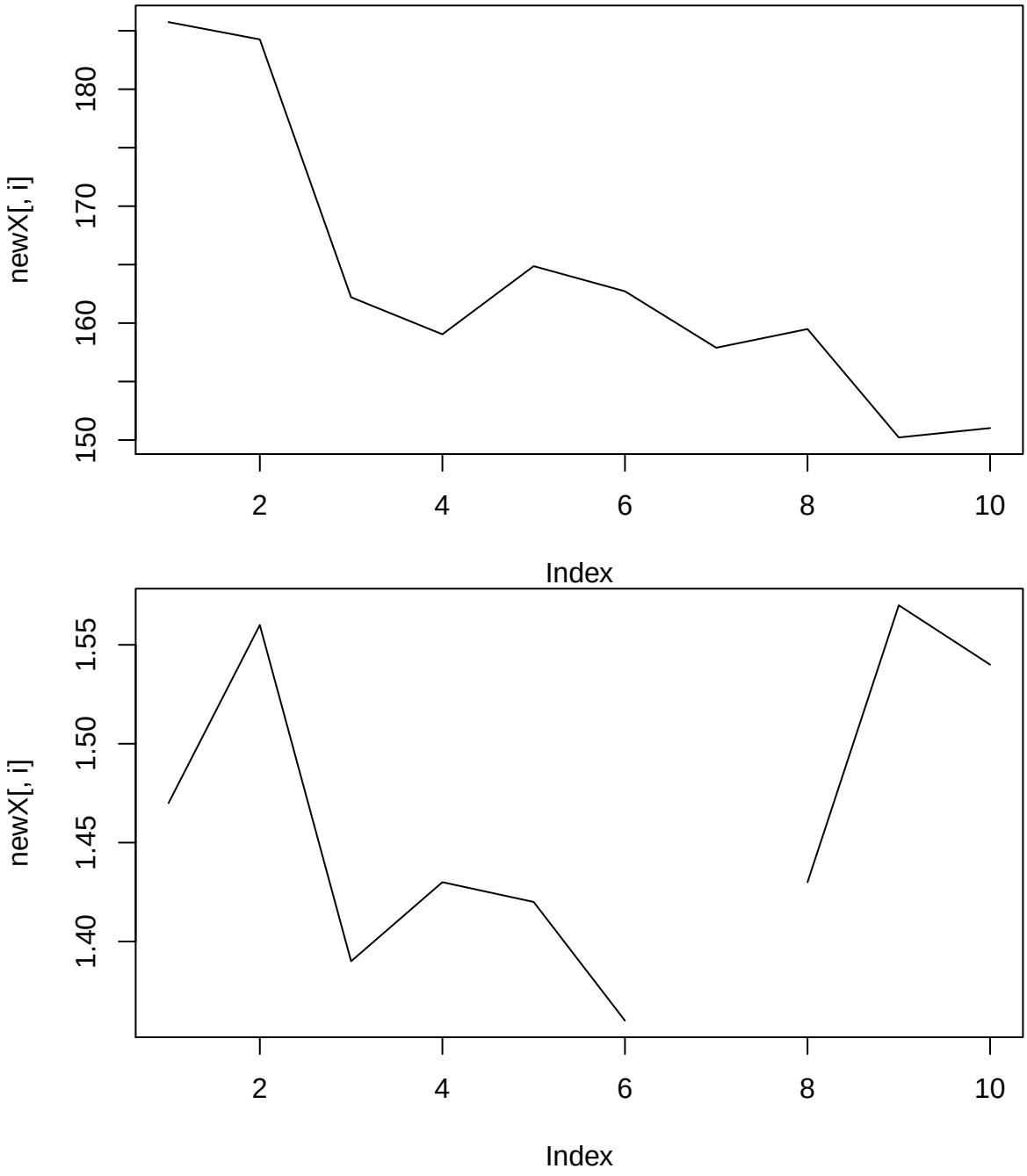
```
##  Stock1  Stock2  Stock3  Stock4
##  185.74    1.57 1605.00   97.49
```

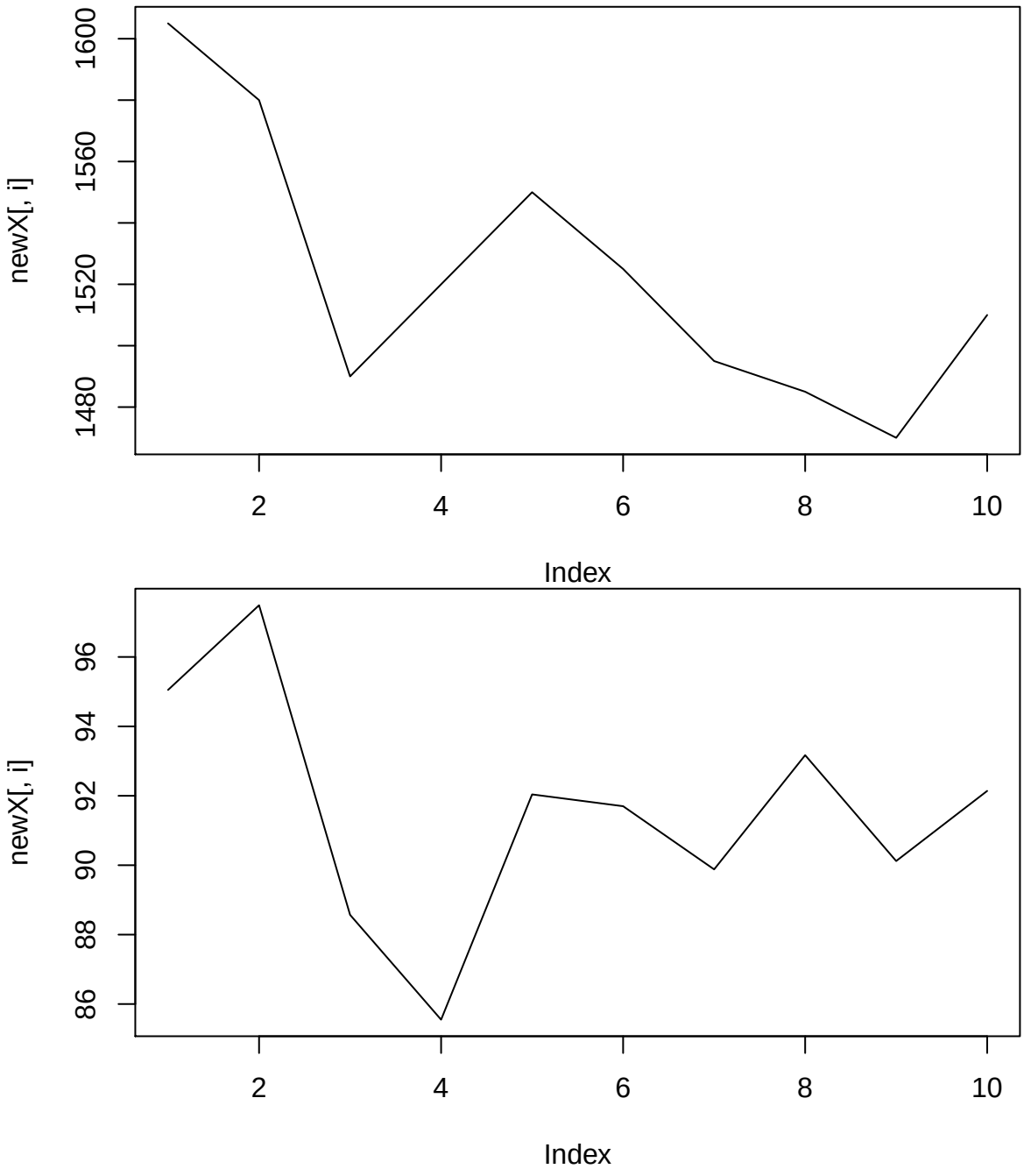
```
apply(stockData[,-1],MARGIN = 2,FUN=quantile,probs=c(0.2,.80),na.rm=TRUE)
```

```
##      Stock1 Stock2 Stock3 Stock4
## 20% 156.516  1.408   1489 89.618
## 80% 168.748  1.548   1556 93.546
```

Create a plot of each column, using a "line

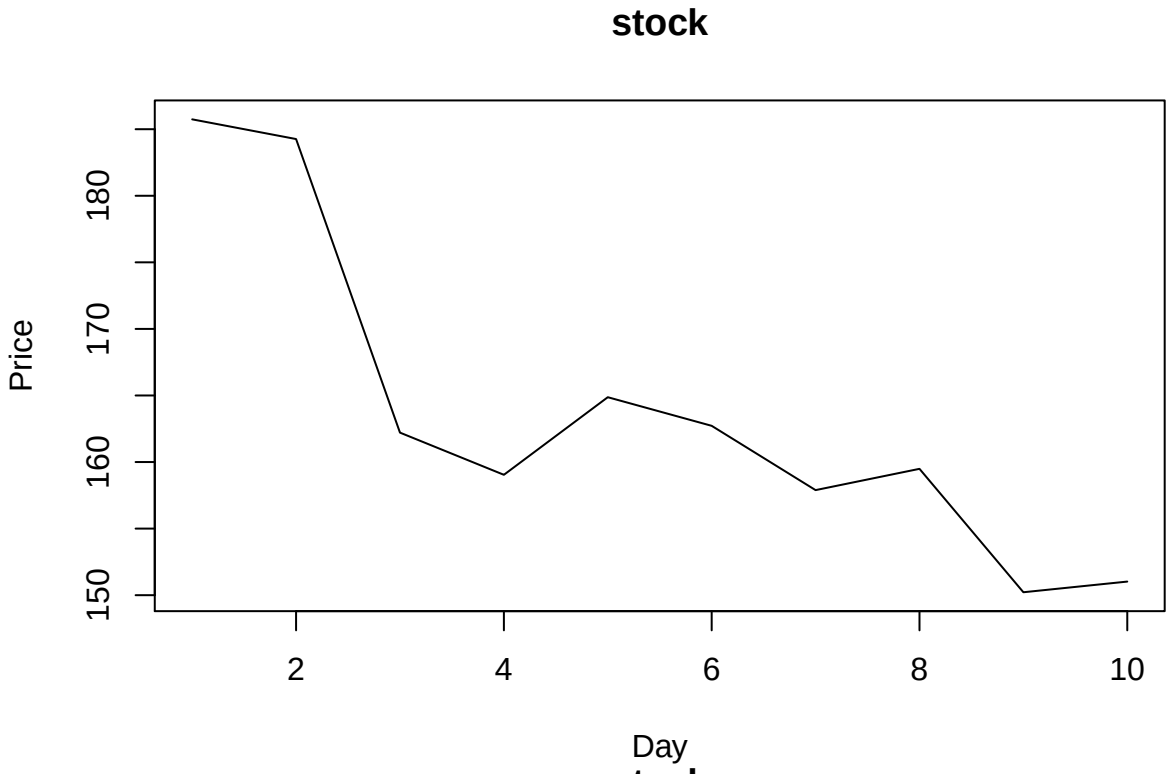
```
apply(stockData[,-1],MARGIN = 2,FUN=plot,type="l")
```

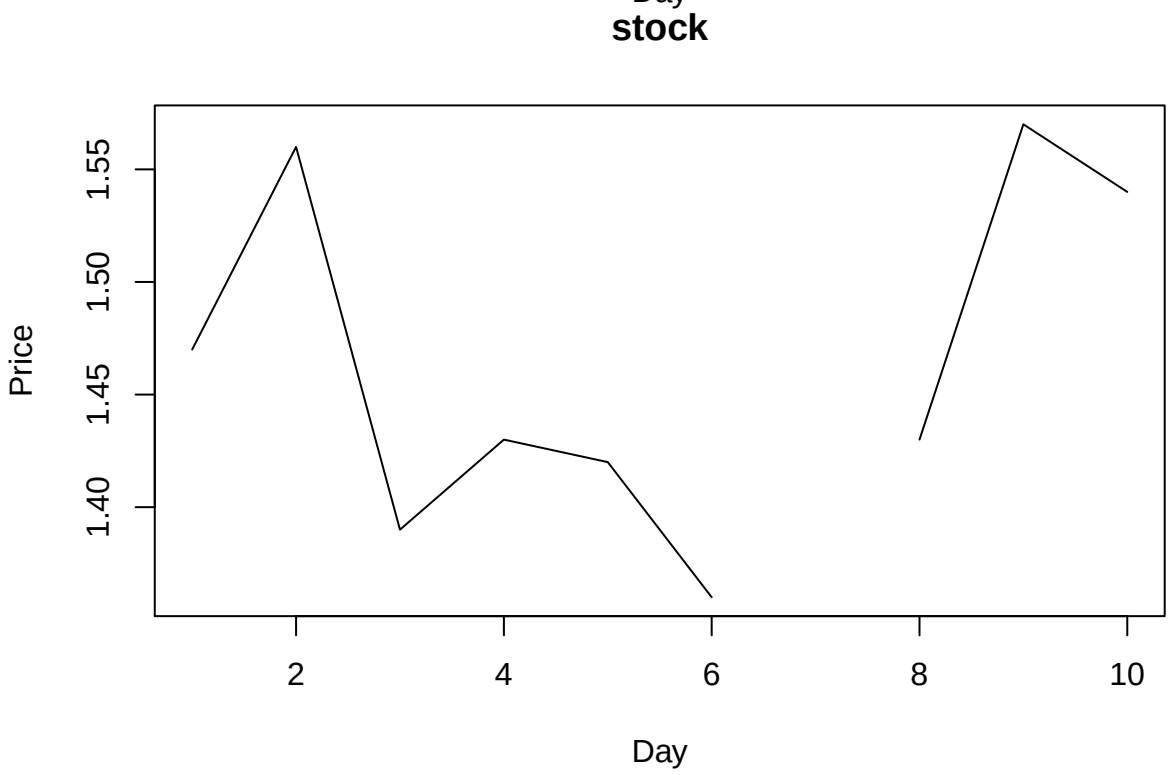


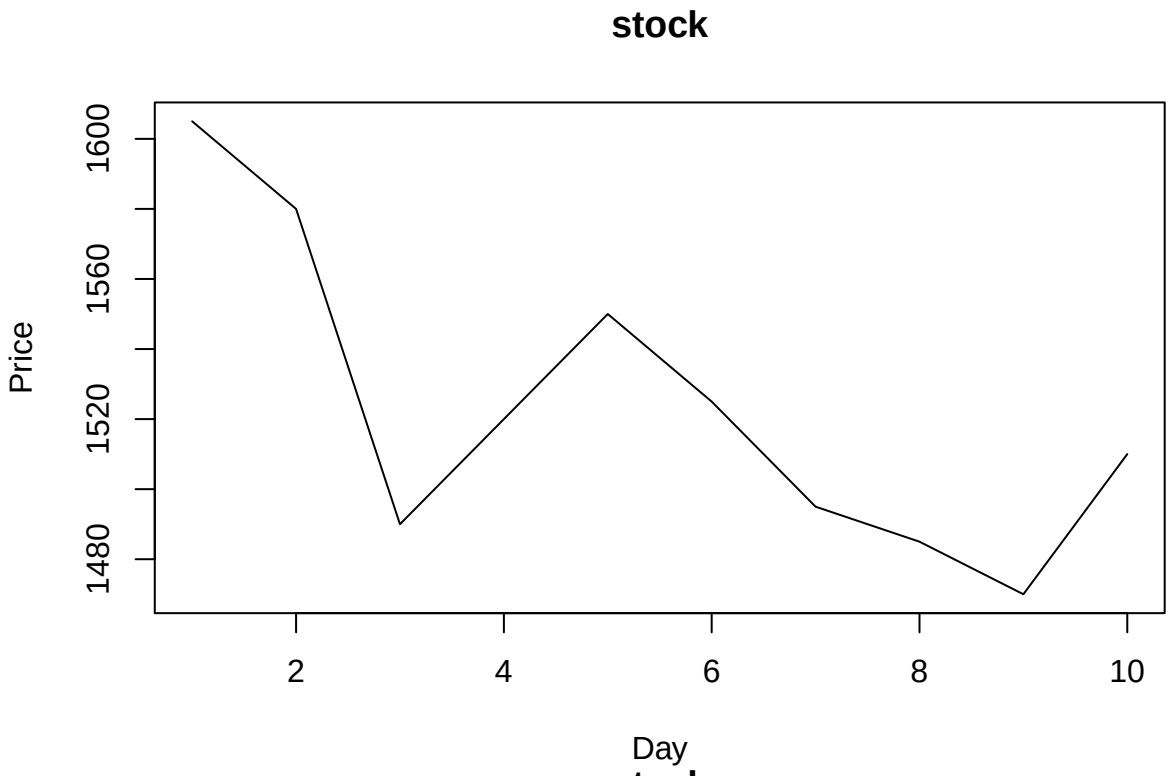


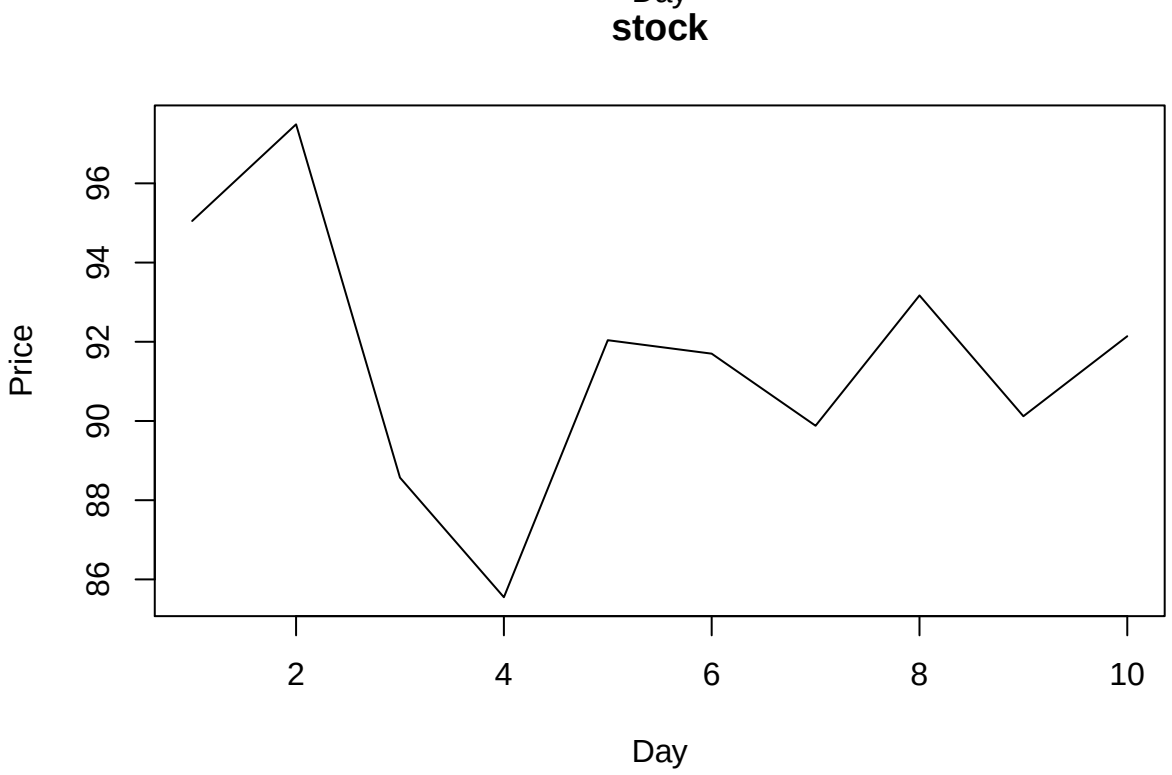

```
## NULL
```

```
apply(stockData[,-1],MARGIN = 2,FUN=plot,type="l",main="stock",ylab="Price",xlab="Day")
```


stock
Price
150
160
170
180
2
4
6
8
10
Day


stock
Price
1.40
1.45
1.50
1.55
2
4
6
8
10
Day






```
## NULL
```

```
apply(stockData[,-1],MARGIN = 2,FUN=sum,na.rm=TRUE)
```

```
##    Stock1    Stock2    Stock3    Stock4
##   1637.46     13.17 15230.00    915.71
```

OR

```
rowSums(stockData[,-1],na.rm = TRUE)
```

```
##  [1] 1887.26 1863.31 1742.17 1766.02 1808.33 1780.78 1742.77 1739.09
##  [9] 1711.91 1754.70
```

Chapter 13 : Using the apply family of Functions in R

apply() function

lapply() function

sapply() function

tapply() function

mapply() function

```
data <- matrix(c(1:10, 21:30), nrow = 5, ncol = 4)
```

```
data
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   21   26
## [2,]    2    7   22   27
## [3,]    3    8   23   28
## [4,]    4    9   24   29
## [5,]    5   10   25   30
```

apply : apply can be used to apply a function to a matrix.

```
apply(data, 1, mean)
```

```
## [1] 13.5 14.5 15.5 16.5 17.5
```

The second parameter is the dimension. 1 signifies rows and 2 signifies columns. If you want both, you can use c(1, 2).

lapply : is similar to apply, but it takes a list as an input, and returns a list as the output.

```
data <- list(x = 1:5, y = 6:10, z = 11:15)
data
```

```
## $x
## [1] 1 2 3 4 5
##
## $y
## [1]  6  7  8  9 10
##
## $z
## [1] 11 12 13 14 15
```

```
lapply(data,FUN = median)
```

```
## $x
## [1] 3
```

```
##
## $y
## [1] 8
##
## $z
## [1] 13
```

sapply : is the same as lapply, but returns a vector instead of a list.

```
sapply(data, FUN = median)
```

```
##  x  y  z
##  3  8 13
```

tapply : splits the array based on specified data, usually factor levels and then applies the function to it.

```
library(datasets)
tapply(mtcars$wt, mtcars$cyl, mean)
```

```
##        4        6        8
## 2.285727 3.117143 3.999214
```

The tapply function first groups the cars together based on the number of cylinders they have, and then calculates the mean weight for each group.

mapply is a multivariate version of sapply. It will apply the specified function to the first element of each argument first, followed by the second element, and so on.

```
x <- 1:5
b <- 6:10
mapply(sum, x, b)
```

```
## [1]  7  9 11 13 15
```

It adds 1 with 6, 2 with 7, and so on.