4. Data Structures

4.1 Variables and assignment

Variables

- A variable is an identifier (name) that points to a memory location in RAM that stores a value that can change when the program is run.

Assignment

- Set up a vector, x

```
> x <- c(1,2,3,4,5)
> assign("x",c(1,2,3,4,5))
> c(1,2,3,4,5) -> x
```

- Putting a value into a variable is known as assignment.
- R uses to work with temporary variables in the functions, with only a return().
- If you assign a variable in the Global environment or when you are using functions more than scripts, you can use
- super-assignment operator: <<-
- assign function: assign("b“ ,value, envir = globalenv())

4.2 Data Types

Basic Data types in R

1. **Numeric**
2. **Integer**
3. **Complex**
4. **Logical**
5. **Character**

```
> setwd("D:/R_data")
> a <- c(1,2,3,4,5);class(a)
[1] "numeric"
> i <- as.integer(a);class(i)
[1] "integer"
> c <- c(1+0i,1+2i,1+3i,3+4i,5+5i);class(c)
[1] "complex"
> a1 <- seq(11,15)
> l <- a < a1;class(l)
[1] "logical"
> s <- c("a","b","c","d","e")
> class(s)
[1] "character"
>
```

4.2 Data Types - continued

Data object type	Description
Vector	a sequence of numbers or characters, or higher-dimensional arrays like matrices
list	a collection of objects that may themselves be complicated
factor	a sequence assigning a category to each index
data.frame	a table-like structure
Environment (hash-table)	A collection of key-value pairs

4.2 Data Types - continued

Data objects

1. Vector
2. List
3. Factor
4. Data frame
5. Matrix
6. Time Series

- A name (a.k.a symbol) is a way to refer to R objects by name.

```
> ans <- as.name("cat")
> typeof(ans)
[1] "symbol"
> ans
cat
```

4.2 Data Types - continued

```
> # Data object types
> #----------------------------------------------
> #
> # 1. Vector
> #
> v<-seq(16,20)
> names<-c("Alex","Alwin","Arun","Asin","Austin")
> #
> #
> # 2. List
> #
> lis<-list(id = v,name = names);class(lis)
[1] "list"
> print(lis) # 2
$id
[1] 16 17 18 19 20

$name
[1] "Alex"   "Alwin"  "Arun"   "Asin"   "Austin"
```

4.2 Data Types - continued

```
> # 3. Factor
> #
> fac<-factor(c(15:11,10:14,16,17));class(fac)
[1] "factor"
> print(fac);levels(fac) # 3
 [1] 15 14 13 12 11 10 11 12 13 14 16 17
Levels: 10 11 12 13 14 15 16 17
[1] "10" "11" "12" "13" "14" "15" "16" "17"
> #
> # 4. Data Frame
> #
> df1<-data.frame(id = v,name = names);class(df1)
[1] "data.frame"
> print(df1);levels(df1) # 4
  id   name
1 16   Alex
2 17  Alwin
3 18   Arun
4 19   Asin
5 20 Austin
NULL
```

4.2 Data Types - continued

```
> #
> # 5. Matrix
> #
> mat1<-as.matrix(df1);class(mat1)
[1] "matrix"
> print(mat1);levels(mat1) # 5
     id   name
[1,] "16" "Alex"
[2,] "17" "Alwin"
[3,] "18" "Arun"
[4,] "19" "Asin"
[5,] "20" "Austin"
NULL
> #
> # 6. Time Series
> #
> ts1<-ts(sample(seq(1,30),30), frequency = 12, start = c(2012,9));class(ts1)
[1] "ts"
> print(ts1) # 6
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2012                                  18  19   6  13
2013  24  15  23  30  16  12  20  26   4  10  11   2
2014   1  22  21  25   5  14  29  27  28   7   8   9
2015  17   3
```

4.3 Indexing, sub-setting

- R contains several constructs which allows access to individual elements or subsets through indexing operations.
- In case of the basic vector types one can access the ith element using x[i], but there is also indexing of lists, matrices, and multi-dimensional arrays.
- There are several forms of indexing in addition to indexing with a single integer.
- Indexing can be used both to extract part of an object and to replace parts of an object (or to add parts).

4.3 Indexing, sub-setting - continued

- There are three subsetting operators [ ], [[ ]] and $.

```
> m<-matrix(c(1,2,3,4,5,6,7,8),
+ nrow = 2, ncol = 4, byrow = TRUE)
> i=3
> j=5
> print(m)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
> print(m[,i])    # third column
[1] 3 7
> print(m[2,2])   # value at second row &
[1] 6             # second column
>
```

4.3 Indexing, sub-setting - continued

- [[is similar to [, except it can only return a single value and it allows you to pull pieces out of a list.
- $ is a useful shorthand for [[combined with character subsetting.
- You need [[when working with lists.
- This is because when [is applied to a list it always returns a list; it never gives you the contents of the list. to get the contents, you need [[.

Reference: http://stackoverflow.com/questions/22431261/understanding-list-indexing-and-bracket-conventions-in-r

4.3 Indexing, sub-setting - continued

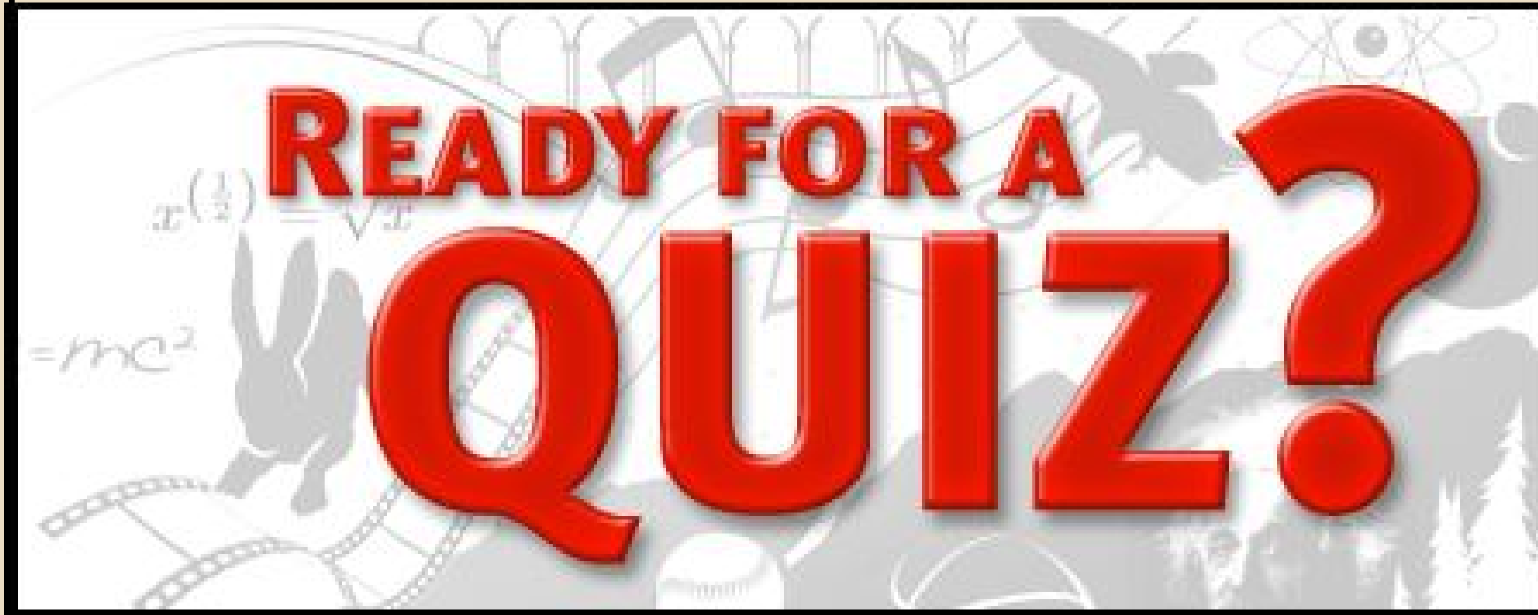
```
> f<-list(1,2,"three")
> list_ext<-f[1];print(list_ext);class(list_ext)
[[1]]
[1] 1

[1] "list"
> val_ext<-f[[1]];print(val_ext);class(val_ext)
[1] 1
[1] "numeric"
```

```
> rec<-list(length = 5, width = 10)
> rec$length;class(rec$length)
[1] 5
[1] "numeric"
> rec[1];class(rec[1])
$length
[1] 5

[1] "list"
```

R Programming Course



R Programming Course

1. Which line(s) in the following code snippet produces error?

Untitled - R Editor

```
setwd("D:/R")
#
# cat is useful for producing output in user-defined functions.
#     It a) converts its arguments to character vectors,
#        b) concatenates them to a single character vector,
#        c) appends the given sep = string(s) to each element and
#        d) then outputs them
#
x   <-  list(id = c(1,2,3,4,5),label= c("A","B","C","D","E"))
d   <-  c(x[1],x[2])
cat("\n d =",d,"class =",class(d),sep=" ","\n")     # cat line 1
#
e   <-  c()
e   <-  c(x[[1]],x[[2]])
cat("\n e =",e,"class =",class(e),sep = " ","\n")  # cat line 2
```

a) Both cat line 1 and cat line 2

b) Neither cat line 1 nor cat line 2

c) Only cat line 1

d) Only cat line 2

Note: cat can handle only atomic vectors or names.

2. **The four main modes, which describe the basic type of elements of the object. They are:**
   i. *Numeric*
   ii. *Character*
   iii. *_________*
   iv. *Logical*

   The missing type is:

   a) *Binary*
   b) *Complex*
   c) *All of the above*
   d) *None of the above*

3. **If I execute the expression i <- 50 in R language, what is the value of class(x)?**
   - *a) Numeric*
   - *b) Integer*
   - *c) Real*
   - *d) complex*

4. **What is the class of the object defined by the expression a <- c(120,"i",FALSE)?**
   - *a) Numeric*
   - *b) Character*
   - *c) Integer*
   - *d) Logical*

1. c) cat line 1
2. b) Complex
3. a) Numeric
4. b) Character

R Programming Course

Lab Exercise 1:

- ***All the 100 students are assigned Student ID ranging from 1 to 100. Select ten students from B.Com Final year at random.***

- ***The importance of the theory of sampling lies in the fact that for a large population, it is neither practical nor possible to collect data for each and every number of the population.***

Lab Exercise 1 - continued:

- ***The function sample generates random integers without replacement.***
- ***The first argument specifies a vector containing the specified range of valid numbers.***
- ***The second argument indicates the number of random integers to be returned.***
- ***The function sort, sorts the output.***

```
> # Program to generate ten random integers
> # from the range of 1 to 100
> id <- sort(sample(1:100, 10, replace = FALSE))
> cat("\n selected ids ",id,"\n",sep=" ")

 selected ids  9 10 19 24 41 49 50 55 65 77
>
```

Lab Exercise 2:

- ***Find the mean, median and mode of the set of following observations:***

  27,36,28,18,35,26,20,35,40,26

- ***An average is considered as a typical representative of the whole data.***
- ***The various averages in common use are:***

  1. Mean

  2. Median

  3. Mode

Lab Exercise 2 – continued:

- ***Arithmetic mean is the most popular measure of central tendency.***
- ***Median is the middle value for a data that has been arranged in order of magnitude. The median is less affected by outliers and skewed data.***
- ***Mode is the value that occurs most often in the data set.***

Lab Exercise 2 – continued:

- ***The functions mean() and median() calculates the mean and median of the given data set respectively.***

- ***However, we need to write a function to calculate mode. Note that the mode is a reserved word in R and it denotes the basic data type.***

R Programming Course

Lab Exercise 2 - continued:

```
> # Function to find mode of a given data set
> Mode<- function(x) {
+ ux<- unique(x)
+ uy<- tabulate(match(x,ux))
+ xm<- cbind(ux,uy)[uy == max(uy),]
+ if (class(xm) %in% c("numeric","character"))
+ return(xm[1])
+ else
+ return(xm[,1])
+ }
> #--------------------------------------------------------------
> # To find mean, median and mode of data set
> #--------------------------------------------------------------
> x <- c(27,36,28,18,35,26,20,35,40,26)
> median(x) # to find the median of x
[1] 27.5
> mean(x) # to find the mean of x
[1] 29.1
> Mode(x) # to find the mode of x
[1] 35 26
```

Lab Exercise 3:

- Find the first quartile, median and third quartile; 5th percentile and 95th percentile of the twenty random integers ranging from 1 to 1000.
- The first quartile is the value that cuts off the first 25% of the data when it is sorted in ascending order.
- The second quartile or median is the value that cuts off the first 50% of the data when it is sorted in ascending order.

Lab Exercise 3 - continued:

- **The third quartile is the value that cuts off the first 75% of the data when it is sorted in ascending order.**

- **The nth percentile of a data set is the value that cuts off the first n-percent of the data values when it is sorted in ascending order.**

R Programming Course

Lab Exercise 3:

```
> # to find 1st, 2nd, 3rd quartile
> # to find 5th and 95th percentile
> x <- sample(1000,20)
> m<-matrix(x,nrow=2,ncol=10,byrow= TRUE)
> cat("\n 20 random integers are \n")

 20 random integers are
> for (i in  1:2) {
+ cat("\n Row i",i,"\n ",sep = " ")
+ for (j in 1:10) {
+ cat(m[i,j],":",sep = " ")
+ }
+ cat("\n -------------------------------------------------------\n")
+ }

 Row i 1
 8 :205 :203 :491 :637 :699 :493 :997 :562 :508 :
 -------------------------------------------------------

 Row i 2
 808 :151 :97 :777 :437 :984 :721 :2 :796 :60 :
 -------------------------------------------------------
> q<-quantile(x,c(0.05,0.25,0.50,0.75,0.95))
> print(q)
    5%    25%    50%    75%    95%
  7.70 190.00 500.50 735.00 984.65
> 
```

Lab Exercise 4:

- There are two types of electric bulbs. Samples of size 30 are drawn each from the two types.

  Mean life in hours of type I electric bulbs:

  510,500,495,480,520,485,510,500,495,480,
  520,485, 510,500,495,480,520,485,500,505,
  501,502,498,490,512,498,505,504,491,489

- Mean life in hours of type II electric bulbs:

  610,580,595,580,620,595,610,600,598,580,
  620,595,610,600,608,607,608,605,600,605,
  601,602,598,599,612,598,605,604,591,596

- Which type of electric bulb has more variation and the range of the life in hours for both types?

Lab Exercise 4 continued:

- By dispersion, it is meant spreading of the observations from an average.
- They measure the variability in the observed values in a data set.
- Range is defined to be the difference between the largest and the smallest of the observations.

Lab Exercise 4 - continued:

- The standard deviation measures the variability between observations in the sample or population from the mean of that sample or population.

- Coefficient of variation is the relative measure of dispersion based on standard deviation; it is defined by (SD/Mean) * 100.

- It is used to compare dispersion in two sets of data especially when the units are different.

R Programming Course

Lab Exercise 4:

D:\Training\R_in_one_day\R_data\l4_act4.R - R Editor

```
A <- c(510,500,495,480,520,485,510,500,495,480,520,485,510,500,
495,480,520,485,500,505,501,502,498,490,512,498,505,504,491,489)
#
B <- c(610,580,595,580,620,595,610,600,598,580,620,595,610,600,
608,607,608,605,600,605,601,602,598,599,612,598,605,604,591,596)
#
mean_life_A  <- mean(A);sd_A <- sd(A)
mean_life_B  <- mean(B);sd_B <- sd(B)
#
CV_A         <- (sd_A / mean_life_A) * 100
CV_B         <- (sd_B / mean_life_B) * 100
#
range_A      <- range(A);range_B <- range(B)
#
cat("\n Range of Bulb life in hours for type","A:",range_A,
"Coefficient of variation",CV_A,"\n",sep=" ")
#
cat("\n Range of Bulb life in hours for type","B:",range_B,
"Coefficient of variation",CV_B,"\n",sep=" ")
#
ifelse(CV_A>CV_B,print("Variation is less in B"),print("Variation is less in A"))
```

```
> source("l4_act4.R")

 Range of Bulb life in hours for type A: 480 520 Coefficient of variation 2.334919

 Range of Bulb life in hours for type B: 580 620 Coefficient of variation 1.654466
[1] "Variation is less in B"
```

Lab Exercise 5:

- Monthly series of income from sales in lakhs of Indian Rupees for a large retail store in Chennai for the years 2010 and 2011 are given below:

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2010	100	120	130	140	150	160	170	180	190	200	210	220
2011	230	240	250	260	270	290	300	310	320	330	340	360

- Predict the income from sales for the year 2012 and plot in a graph the trend line from 2012 to 2016.

Lab Exercise 5 - continued:

- A time series is a set of observations, measured typically at successive points in time spaced at uniform time intervals and arranged in chronological order.
- Time series are used in weather forecasting, earthquake prediction, and largely in any domain of applied science and engineering which involves temporal measurements.
- Examples of time series include:
  a. the hourly series of temperature recorded by the Meteorological observatory
  b. the daily series closing price of shares in the National Stock Exchange

Lab Exercise 5 - continued:

- Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data.
- Time series forecasting is the use of a model to predict future values based on previously observed values.
- The Holt- Winters forecasting procedure is a widely used projection method which can cope with trend and seasonal variation.

Lab Exercise 5 - continued:

D:\Training\R_in_one_day\R_data\l4_act5.R - R Editor

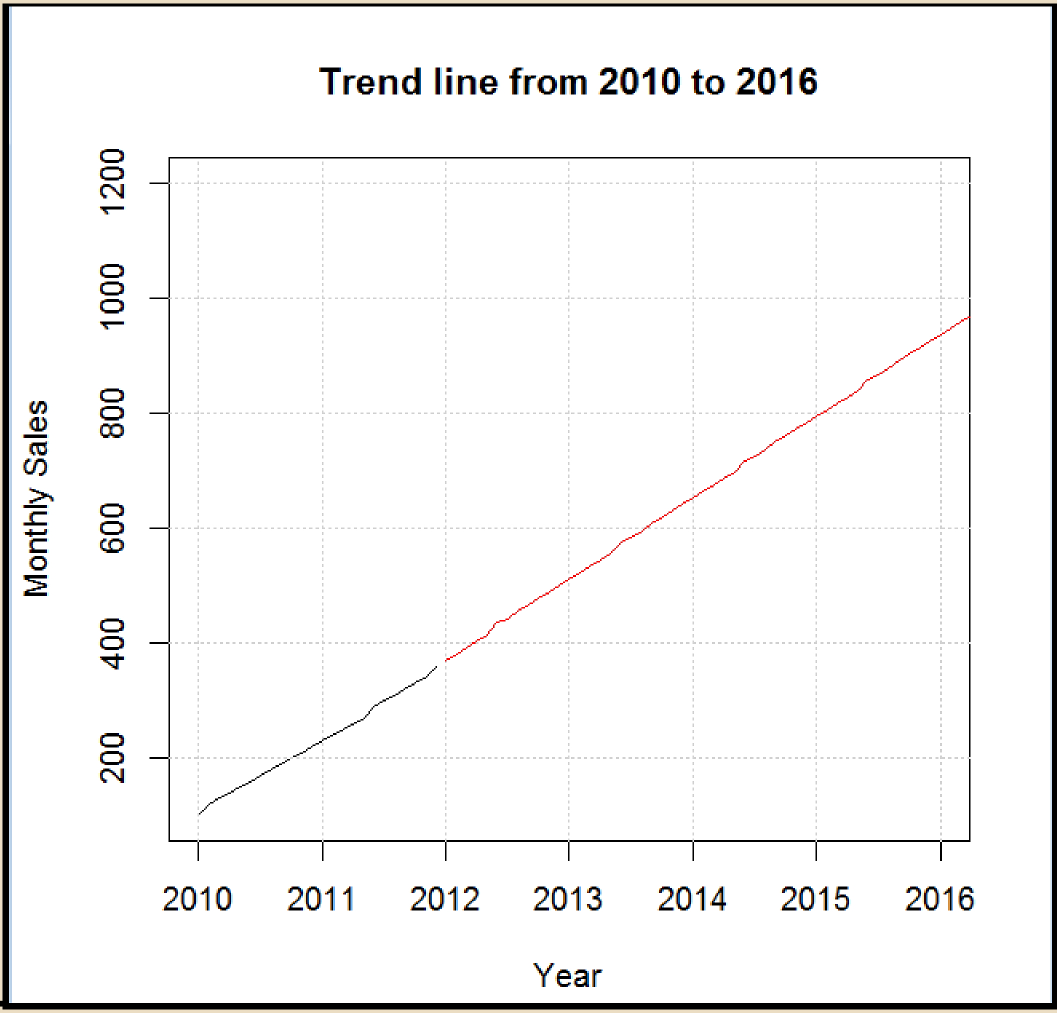
```
require(graphics)
sales <- c(100,120,130,140,150,160,
170,180,190,200,210,220,230,240,250,
260,270,290,300,310,320,330,340,360)
options(digits=5)
za <- ts(sales,start=c(2010,1),frequency=12)
za.hw <- HoltWinters(za)
cat("\n Sales for 2012 - Jan to Dec:",predict(za.hw,n.ahead=12),"\n",sep=", ")
plot(za,xlim=c(2010,2016),ylim=c(100,1200),ylab="Monthly Sales",
xlab="Year",main = "Trend line from 2010 to 2016")
grid()
lines(predict(za.hw,n.ahead=60),col="red")
```

```
> source("l4_act5.R")

 Sales for 2012 - Jan to Dec:  370.89, 381.73, 392.67, 403.6, 414.54, 435.07,
                               443.09, 454.45, 466.22, 477.99, 489.76, 501.14
>
```

R Programming Course

Lab Exercise 5 - continued:



Lab Exercise 5 - continued:

- **Sales vector is converted to a time series object, za.**
- **HoltWinters procedure is performed on dataset, za and stored in za.hw.**
- **Using the predict function, sales for the next 12 months are predicted.**
- **The time series observed values and predicted values are plotted in a graph.**
- **Grid lines are drawn in the graph.**
- ***Lines are drawn through the predicted points for 60 months from 2010 using the color red.***

Lab Exercise 6

- Six tomato plants, of the same variety, were selected at random and treated, weekly, with x grams of fertilizer dissolved in water. The yield of plant is recorded in kilograms.

Plant	A	B	C	D	E	F
x	1.1	2.3	2.4	3.5	2.0	4.0
y	4.5	6.4	6.7	7.4	5.7	7.5

- Calculate the equation of the least squares regression line of y on x.
- Estimate the yield treated weekly with 3.1 grams of fertilizer.

Lab Exercise 6 - continued:

- A simple linear regression model that describes the relationship between two variables x and y can be expressed as y= f(x); where y is called the response variable and x the predictor variable.

- The strength of the relationship is revealed by the correlation coefficient.

- Linear regression consists of finding the best-fitting straight line through the points (observations).

Lab Exercise 6 - continued:

- A line of best fit is a straight line that is the best possible approximation of the given set of data. It is used to study the nature of relation between two variables.

- Least squares is a technique we use for data fitting in linear regression.

- The error of prediction for the response (y) is the value of response (y) minus the predicted value using the regression equation.

R Programming Course

Lab Exercise 6 - continued:

```
> source("14_act6.R")

 y_est = 3.749378 + 1.026388 * x

  The predicted value for y when x = 3.1 grams is  6.93118


  We are 95% confident that the predicted value for the response variable
  y_Est lies in the range 5.738167 and 8.124193


 Strength of the relationship between x and y is  90.54473 %
>
```

```
# ------------------------------------------------------------------------------
# Simple linear regression example
# ------------------------------------------------------------------------------
x           <-    c(1.1, 2.3, 2.4, 3.5, 2.0, 4.0)
y           <-    c(4.5,6.4,6.7,7.4,5.7,7.5)
lmfit       <-    lm(y ~ x)
cat("\n y_est =",lmfit$coefficients[1],"+",
lmfit$coefficients[2],"* x","\n",sep=" ")
new_val     <-    data.frame(x=3.1)
#
predict_y   <-    predict(lmfit,new_val,interval="prediction",level=0.95)
#
cat("\n  The predicted value for y when x = 3.1 grams is ",predict_y[1,"fit"],"\n\n",sep=" ")
cat("\n  We are 95% confident that the predicted value for the response variable")
cat("\n  y_Est lies in the range",predict_y[1,"lwr"],"and",predict_y[1,"upr"],"\n",sep=" ")
#
cat("\n\n Strength of the relationship between x and y is ",
summary(lmfit)$r.squared*100,"%","\n",sep=" ")
# ------------------------------------------------------------------------------
```

Lab Exercise 6:

Read the file "U04_R Data Types_v1.pdf"