STATISTICAL INFERENCE

Second Lab

Lab Agenda

- Accessing Data Frames with logical And\OR
- Control Statements
- Functions
- Workspace
- Data Import and Export with R

Data Frame – cont--

```
> x<-data.frame("var1"=sample(1:5),"var2" = sample(6:10),"var3"= sample(11:15))
> x
  var1 var2 var3
1    4    9   14
2    1    7   13
3    2    6   11
4    5    8   15
5    3   10   12
> x$var2[c(1,3)]=NA
> x
  var1 var2 var3
1    4   NA   14
2    1    7   13
3    2   NA   11
4    5    8   15
5    3   10   12
```

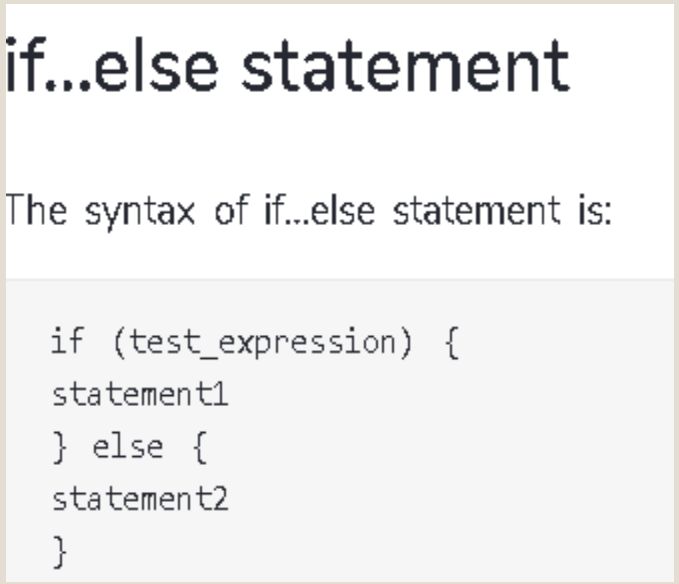
Logical And , OR

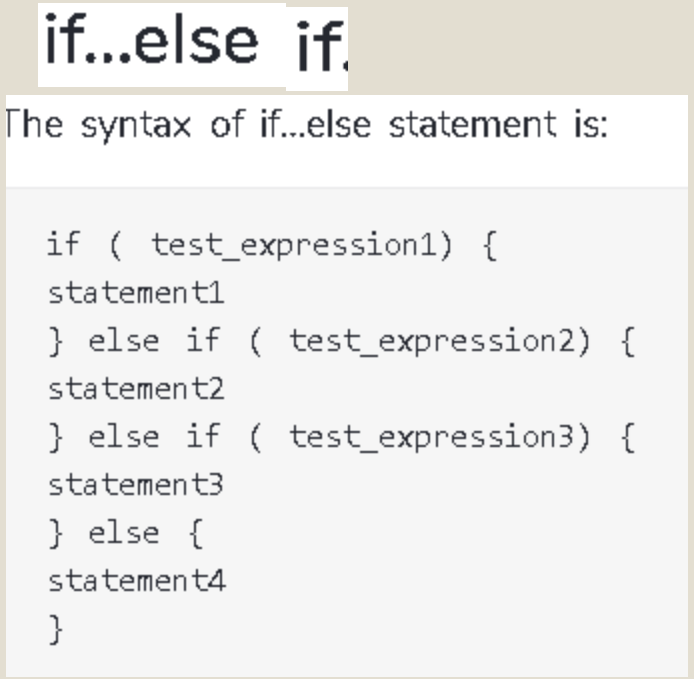
```
> x[x$var1<=3 & x$var3>11,]
  var1 var2 var3
2    1    7   13
5    3   10   12
> x[x$var1<=3 | x$var3>11,]
  var1 var2 var3
1    4   NA   14
2    1    7   13
3    2   NA   11
4    5    8   15
5    3   10   12
```

Control Statements

- if ... else ...
- for loops
- While loop

If … else …





- Operators **&&** or || or **!=** may be used
- Conditional execution of code

If … else …

```
> y<-5

>if (y>0)
 {print (”y postive”)}

[1] ”y is postive”
```

```
> x<- 3

>if (x != 4)
 { x <- x+4}
else {x}

>x
[1] 7
```

```
> z <- 0
>if (z < 0 )
 {print (”z negative”)}
else if (z > 0)
{print (”z positive”)}
 else {print (”z is zero”)}

[1] ”z is zero”
```

for

- **Loops** are used in programming to repeat a specific block of code.
- A **for loop** is used to iterate over a vector in **R** programming.

```
for (variable in vector) {
  do something
}
```

for

```
> x<- 0
>for (i in 1:20)
  { x <- x + 1}
>x


[1] 20
```

```
> x<- 1
> y<- c(1,2,3,4)

>for (i in y)
  { x <- x + i;}
>x

[1] 11
```

while

- **while** (*expr_1*) *expr_2* While *expr_1* is true, repeatedly evaluate *expr_2*
- break and next statements can be used within the loop

while

```
>while (x!=60)
{ x <- x+1}
> X


[1] 60
```

```
> i<- 1
> while (1<6)
{ print(i)
i<- i+1
}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Function Definition

```
function.name <- function(arg1, arg2, ...)
{ computations on the arguments some other code }
```

- Arguments can be assigned default values:
  arg_name = expression
- Return value is the last evaluated expression or can be set explicitly with return()

Function Examples

```
>square <- function(x = 10) {x * x}

> square()
[1] 100
> square(2)

[1] 4
```

```
square <- function(x) {x * x}

> square()

Error in square() : argument "x" is missing, with no default
```

Function Examples

```
>intsum <- function(from=1, to=10) { sum <- 0 ; for (i in from:to) sum <- sum + i; sum;}

> intsum(3) # use default values
[1] 55

> intsum(3) # Evaluates sum from 3 to 10
[1] 52

> intsum(to=3) … # Evaluates sum from 1 to 3 …
[1] 6
```

Return in Functions

```
>intsum <- function (from=1, to=10)
{ sum <- 0 ;
for (i in from:to)
sum <- sum + i;
return(sum); }

> intsum()
[1] 55

> s <- intsum()
> s
[1] 55
```

Hands On

- Create a Function that takes two arguments (x, y) and returns the “X” to the power of “Y”
- ex: Power(2,3) returns 8 default return value is 1.

Some Notes on Functions

- You can print the arguments for a function using args() command

  > **args(intsum)**

  function (from = 1, to = 10)

- You can print the contents of a function by typing only **function name, without the ()**
- You can edit a function using
  > **n_intsum <- edit(intsum)**

Debugging Functions

- Toggle debugging for a function with **debug()/undebug()** command
- With debugging enabled, R steps through function line by line

```
> debug(intsum)
> intsum(10)
```

- Use print() to inspect variables along the way
- Press <enter> to proceed to next line

Workspace

- The workspace is **your current R working environment**
- During an R session, all objects are stored in a temporary, working memory
- **list objects**
  - **ls()**
- **remove objects**
  - **rm(name of the object you want to delete)**
- objects that you want to access later must be saved in a “workspace”
  - from the menu bar: File->save workspace
  - from the command line: save(x,file=“MyData.Rdata”)

Prepare Working Directory

- The working directory is **the default location where R will look for files you want to load and where it will put any files you save**.

>getwd()

>setwd("path of the data set")

Data Import and Export with R

- **Read data from and write data to**
  - CSV files
  - EXCEL files
  - ODBC databases

Read and Write Data from csv file

```
>var1 <- 1:5
>var2 <- (1:5)/10
>var3 <- c("R", "and", "Data Mining", "Examples", "Case Studies")
>df1 <- data.frame(var1, var2, var3)
>names(df1) <- c("VarInt", "VarReal", "VarChar")

# save to a csv file
>write.csv(df1, "dummmyData.csv", row.names = FALSE)

# read from a csv file
>df2 <- read.csv("dummmyData.csv")
>print(df2)
```

Note : Check the working directory and ensure that you have write access to that directory.
You can check this with `getwd()` ;

Practice: Read and Write Data from Excel file

```
>install.packages("xlsx")
>library(xlsx)

>write.xlsx(df2, "dummmyData.xlsx", sheetName = "sheet1", row.names = F)
>df3 <- read.xlsx("dummmyData.xlsx", sheetName = "sheet1")

>df3
```

Read from Database

- Package ODBC: provides connection to ODBC databases.
- Function odbcConnect(): sets up a connection to database
- sqlQuery(): sends an SQL query to the database
- odbcClose() closes the connection.

Practice: Read from Database

```
>install.packages("odbc")
>library(“odbc”)

>db <- odbcConnect(dsn = "servername", uid = "userid",pwd=“”)

>sql <- "SELECT * FROM lib.table WHERE ..."
>myData <- sqlQuery(db, sql)
>odbcClose(db)
```