

# Before the workshop

❏ Download Data Files: [https://github.com/julipetal/Intro\\_to\\_R](https://github.com/julipetal/Intro_to_R)

# Introduction to R



Part I

# Part I

Please interrupt with questions

Basic Introduction

Good Coding Practice

Data Handling

Basic Data Visualization

---

**Basic**

**Introduction**

# What is R?

Free software environment for statistical computing and graphics

Runs on most UNIX platforms, Windows and MacOS

Website: <http://www.r-project.org/>

CRAN - Comprehensive R Archive Network

Popular in academia, and growing software in industry

Frequently releases updated versions with funny names (like 'Bug in Your Hair'), check every 6 month or so to ensure you are using an up-to-date version

Tons of different libraries (of which some do the same thing)

# R code

Any R-script (fileName.r) can be opened in an editor of your choice

- For windows: Notepad++ is good because it highlights different attributes of the code, unlike Notepad

Copy and past R-code from scripts into the R window, also called R console

Directly write your code into the R window

Using UP and DOWN arrows will reveal lines of code that have been typed before within the current R history

To execute a line of code press enter

# Help!!!

```
> help.start()
```

```
> help(<functionName>)
```

Replace `<functionName>` with which

```
> ?<functionName>
```

Again try which and then for

```
> help.search("<functionName>")
```

```
> example(<functionName>)
```

```
> demo()
```

```
> demo(graphics)
```

Easy examples and explanations for basic R functions

<http://www.statmethods.net/>

RStudio as an editor

<https://www.rstudio.com/products/rstudio/download/#download>

# Build in Functions

- Standard math operations and functions

- `+, -, *, /`
- `exp()`, `log()`, `sin()`, ...

- Standard statistics

- `mean()`, `sd()`, `density()`

- Assigning numbers to variables

- `> X = 65`
- `> x = 43`
- `> X + x`
- You will also see `<-` instead of an equal sign, it does the same thing
- `> y <- 2`

**NOTE: R is case sensitive**

- R Reference Card: <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>



# Closing an R Session/Console

To close a session

```
> q()
```

R will prompt you if you want to save your session

If you kept a script and saved important data to file, there is no need to save the session.

Otherwise you might want to say yes. If you do, an .Rhistory file will be created. This will overwrite any previously created file if not renamed. The .Rhistory file includes all lines of code without output and also will include nonfunctional code.

**Good Coding Tip: Instead of saving the R session, keep a clean script file with comments.**

**Good**

**Coding Practice**

# Good Practice

Save work and code by using scripts

Keep these scripts clean

- Avoid saving code that did not work or produced unwanted results
- Avoid saving R output within these script files (.r)
- Comment your code using #
  - Write a bullet point of what the line of code does for future reference

Find a way to structure and organize your scripts and data

Save often ... save again

Open your R window in a work directory away from backups and original files - you do not want to accidentally overwrite them. R does not ask if “you are sure”.

# Script files - your notebook of code

Here is an example of a script file  
viewed in Notepad++

Comments start with #

R recognizes # and will not try to  
execute these lines. Also notepad++  
highlights comment lines in green

Different colors make the code visually  
easier to read

```
10 # Checking for zero data entries
11 length(as.vector(which(as.vector(expM)==0)))
12 # [1] 419362
13 |
14 ### MM file created, checked if all measured were included
15 ### split MM into 6 files for computational reasons
16
17 library(petal)
18 ### first sub-matrix done step-wise to double-check output
19 m1 = readinTable("smallMM_1.txt")
20 vec = as.numeric(m1[,3])
21 indexRM = as.vector(which(abs(as.numeric(m1[,3]))<0.5))
22 newCurrentMM = m1[-indexRM,]
23 rm(indexRM, m1)
24
25
26 ### integrating remaining 5 files automatically
27 for(i in 2:6){
28   currentMM = readinTable(paste("smallMM_",i,".txt",sep=""))
29   tmpVec = as.numeric(currentMM[,3])
30   vec = c(vec, tmpVec)
31   indexRM = as.vector(which(abs(as.numeric(currentMM[,3]))<0.5))
32   tmpnewCurrentMM = currentMM[-indexRM,]
33   newCurrentMM = rbind(newCurrentMM, tmpnewCurrentMM)
34   rm(currentMM, tmpVec, indexRM, tmpnewCurrentMM)
35 }
36
37 ### checking distribution of association measure (Spearman)
38 measures = vec
39 min(measures, na.rm = TRUE) - 0.5, digits = 0)
40 maxExp = round(max(measures, na.rm = TRUE) + 0.5, digits = 0)
41 png(filename = "Histogram_MM.png", width = 7, height = 4.25, units = "in", pointsize = 12, bg = "white", res = 600)
42 h = hist(measures, breaks = seq(-1, 1, by = 0.05))
43 dev.off()
44
45 min(measures)
46 # [1] -0.872
47 max(measures)
48 # [1] 1
49 dim(newCurrentMM)
50 # [1] 33132192      3
51
52 ### saving output
53 write.table(newCurrentMM, file = "MM_reduced.txt", sep = "\t", quote=FALSE, row.names = FALSE, col.names = TRUE)
54 save(newCurrentMM, file="newCurrentMM.RData")
```

# Directory

Directory is just another word for folder

Make sure you know in which directory your R window is

Work in an directory away from backups and original data files

```
> getwd()
```

```
> setwd("C:/ <TAB> ")
```

```
> getwd()
```

# You are learning a new language

## Syntax

- What is the difference between  $f(x)$  versus  $f[x]$ ?

## Vocabulary

- What functionality is part of R and what do I have to code myself?
- What are these functions called?

## Thinking in that language rather than translating

- “Did you go to school?” Versus “Did you to school go?” (German)

Do not try to memorize everything - write scripts and know where to find them, plus google and other people's code are your best friends.

# Variables

- Any letter, sequence of letters, or letters followed by a number can be used as variable names
  - `X`, `xx`, `myData`, `myData1`, `myData2`, `my_data`
- A variable name cannot start with a number
- Short variable names are great for typing, long variable names often make it easier to understand the code at a later time
- Do not overwrite R's predefined functions, in some cases an error message will appear
- `<Tab>` autocompletes variable names, function names, or filenames - very handy
- R is case sensitive
- Avoid overriding variables

# Data Handling

Indexing, Simple Manipulation, Uploading, Investigation,  
Exporting data



# Vectors & Operations & Functions

A vector is an ordered sequence of numbers

Multiple ways to create a vector

```
> x = c(5, 64, 7, 3)
```

```
> y = 4:7
```

```
> z = seq(4,7)
```

```
> t = seq(1, 11, by=2)
```

```
> u = seq(1, 11, by=2)
```

```
> tmp = rep(10, 6)
```

```
> x = c(5, 64, 7, 3)
```

```
> y = c(34, 65, 0, 7)
```

```
> new1 = cbind(x,y)
```

```
> new2 = rbind(x,y)
```

```
> x+y
```

```
> x^2
```

```
> x[3]
```

```
> x[10]= 4
```

```
> length(x)
```

```
> sum(y)
```

```
> sum(y)/length(y)
```

```
> mean(y)
```

# Indexing and Simple Manipulations

R has some integrated dataset available

```
> data()
> women
> dim(women)
> rownames(women)
> colnames(women)
> women[1,1]
> women[1,]
> women[,2]
> women[1:4,1:2]
```

```
> women$height
```

```
> meanh = mean(women[,1])
```

```
> meanw = mean(women[,2])
```

```
> hmin = min(women[,1])
```

```
> hmax = max(women[,1])
```

```
> table(women)
```

```
> hist(women[,1])
```

```
> hist(women[,1], breaks = seq(57, 73, by=1))
```

# **Investigating Data & Correlation & Standard Stats**

Refer to worksheet

**LET'S TAKE  
A BREAK**



# Data Upload

Function	
<code>read.csv()</code>	Convenient function for .csv files  Returns a list-object
<code>read.table()</code>	Any delimited text files can be read-in, can be clunky at times,  Returns a list-object
<code>scan()</code>	Reads file line by line  Returns a vector-object

```
read.csv("data.csv", header = TRUE)
```

```
read.table("data.txt", header=TRUE,  
           sep="\t")  
read.table("data.txt", header=FALSE,  
           sep="\t")
```

```
scan("data.txt", what="", sep="\t",  
     nline=1, skip=1)  
scan("data.txt", what="", sep="\t",  
     n=15, skip=1)
```

# Data Upload Script

```
fileName = ""

# make sure the file is in table format
# line 1 should have the same length as line 2

l1 = scan(fileName, sep = "\t", what = "", nlines = 1)

tmpM= matrix(scan(fileName, what = "", sep = "\t", skip = 1), ncol = length(l1), byrow = TRUE)
myData= matrix(as.numeric(tmpM[, -1]), ncol = length(l1) -1, byrow = FALSE)
colnames(myData) = l1[-1]
rownames(myData) = tmpM[, 1]

# remove variables (objects) you no longer need
rm(fileName, tmpM, l1)

# check if the file was read in correctly
dim(myData)
head(myData)

# myData is a numeric matrix object
```

# Exporting Data

```
> write.table(myData, "myDataM.csv", quote=FALSE, row.names=TRUE,  
              col.names=FALSE, sep=",")  
> write.table(myData, "myDataM.txt", quote=FALSE, row.names=FALSE,  
              col.names=TRUE, sep="\t", append=FALSE)
```

Objects can also be saved as .RData files

```
> save(new_trees, file="trees.RData")  
> q()  
> load("trees.RData")  
> objects()
```

Can save more than one object

Good coding tip: give the objects meaningful names, not `x` and `y` - two days from now you won't know what `x` and `y` are :)

```
> save(x, y, file="dataObjects.RData")
```

# Data Visualization

Histograms and Scatter Plots



# **Histogram** - `hist()` **&** **Scatterplot** - `plot()` **&** **Graphical Parameters**

Refer to worksheet

# Plotting two graphics into one window

```
vec = as.vector(myData)
```

```
MIN = round(min(vec)-0.5, digits=0)
```

```
MAX = round(max(vec)+0.5, digits=0)
```

```
par(mfrow = c(2, 1))
```

```
hist(myData[,1], breaks=seq(MIN, MAX, by = (MAX-MIN)/numBins), col="green",  
xlab="Control", main="Values for Control Group")
```

```
hist(myData[,2], breaks=seq(MIN, MAX, by = (MAX-MIN)/numBins), col="red",  
xlab="Treatment", main="Values for Treatment Group")
```

# Histogram - hist()

```
# number of histogram's subintervals
numBins = 100

# MIN and MAX will always be integers
MIN = round(min(myData[,1])-0.5, digits=0)
MAX = round(max(myData[,1])+0.5, digits=0)

h = hist(myData[,1], breaks=seq(MIN, MAX, by = (MAX-MIN)/numBins),
col="green", xlab="Control", main="Values for Control Group")

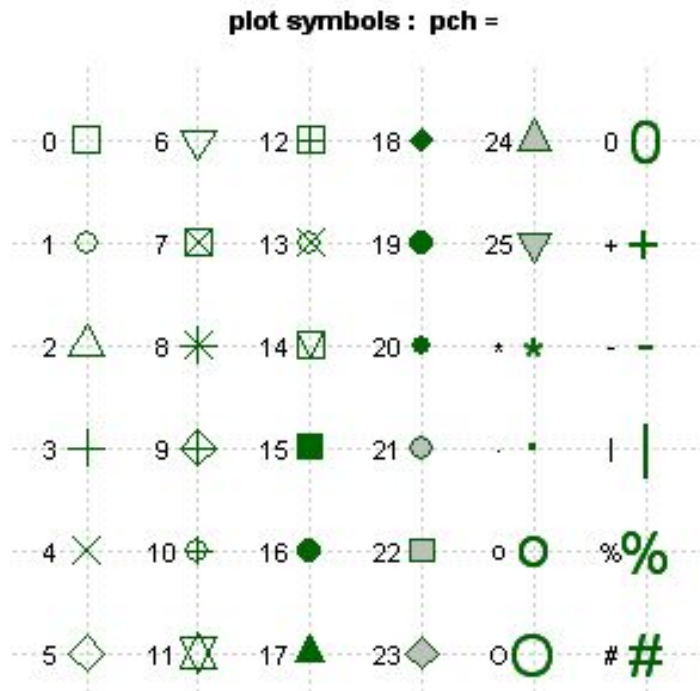
# see statistics of h
h

# barplot() versus hist()
barplot(myData[,1])
```

# Histogram - hist() script

```
TITLE =  
xAXIS =  
COLOR =  
vec =  
numBins =                # number of histogram's subintervals  
fileName =  
  
# MIN and MAX will always be integers  
MIN = round(min(vec)-0.5, digits=0)  
MAX = round(max(vec)+0.5, digits=0)  
  
png(filename = paste(fileName, ".png", sep=""), width=7, height=8.5,  
units="in", pointsize=12, bg="white", res=600)  
h = hist(vec, breaks=seq(MIN, MAX, by=(MAX-MIN)/numBins), col=COLOR,  
xlab=xAXIS, main=TITLE)  
dev.off()
```

# Plotting Symbols



# Line Types

lty

line type

lwd

line thickness, default =1

## Line Types: lty=

