



Workshops on useR

Basics

Day 1

S. Morteza Najibi
Persian Gulf University

Feb 25, 2017



Workshops on user, Basics

Session 1-2

Introduction to R and Rstudio

S. Morteza Najibi
Persian Gulf University

Popular Statistical software

- SPSS: **S**tatistical **P**ackage for **S**ocial **S**ciences
- STATA: **S**tatistical **D**ata
- SAS: **S**tatistical **A**nalysis **S**ystem
- S and S-PLUS
- R

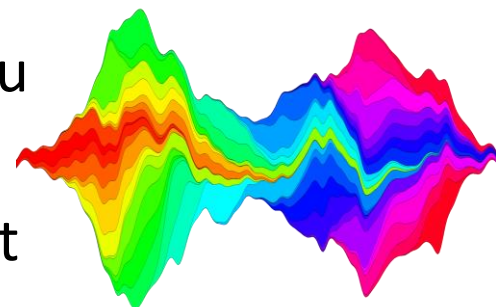
Are they ordered in terms of quality? importance? popularity?

Of course, Not!

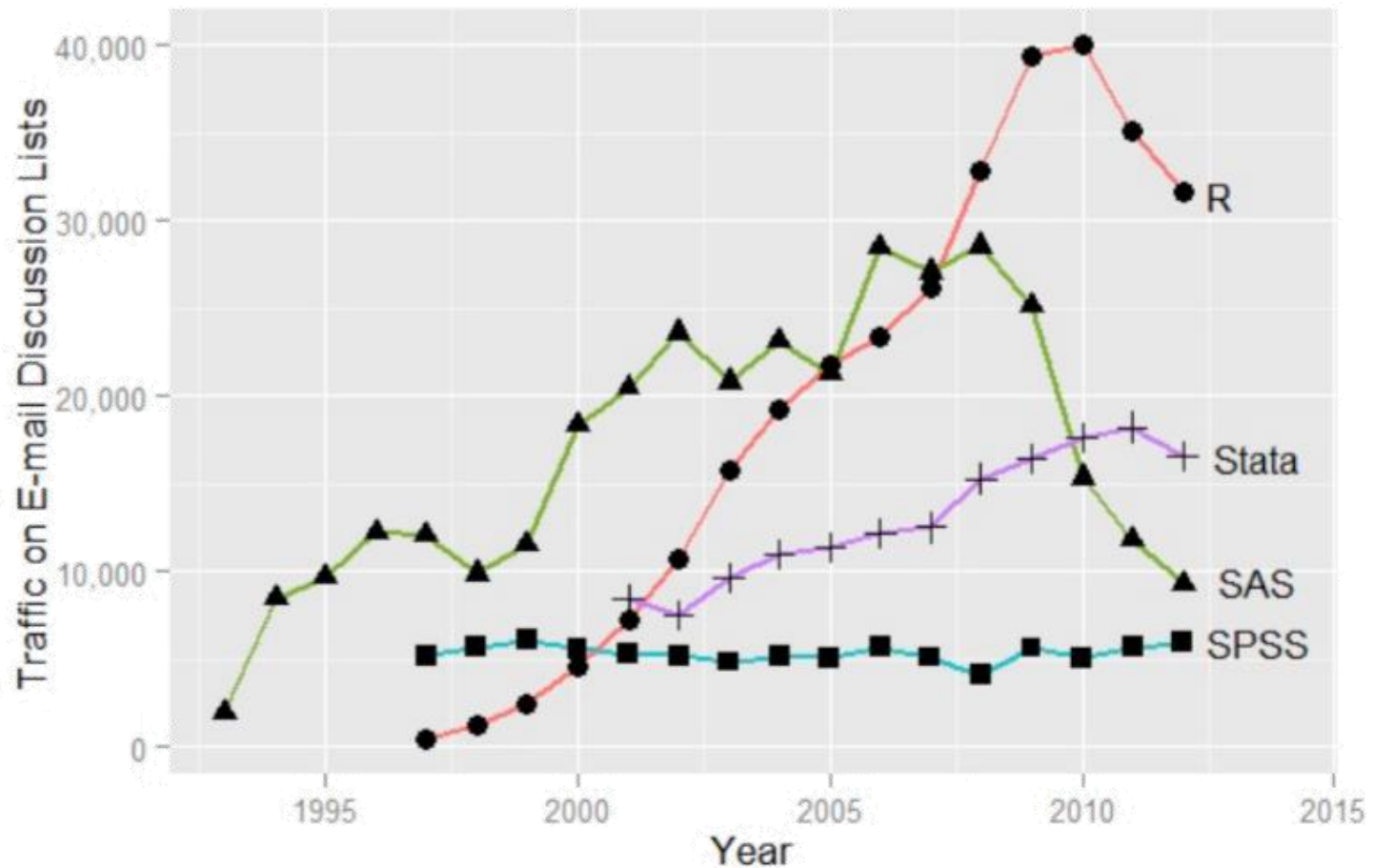
Why R?

What is R and Why Use R?

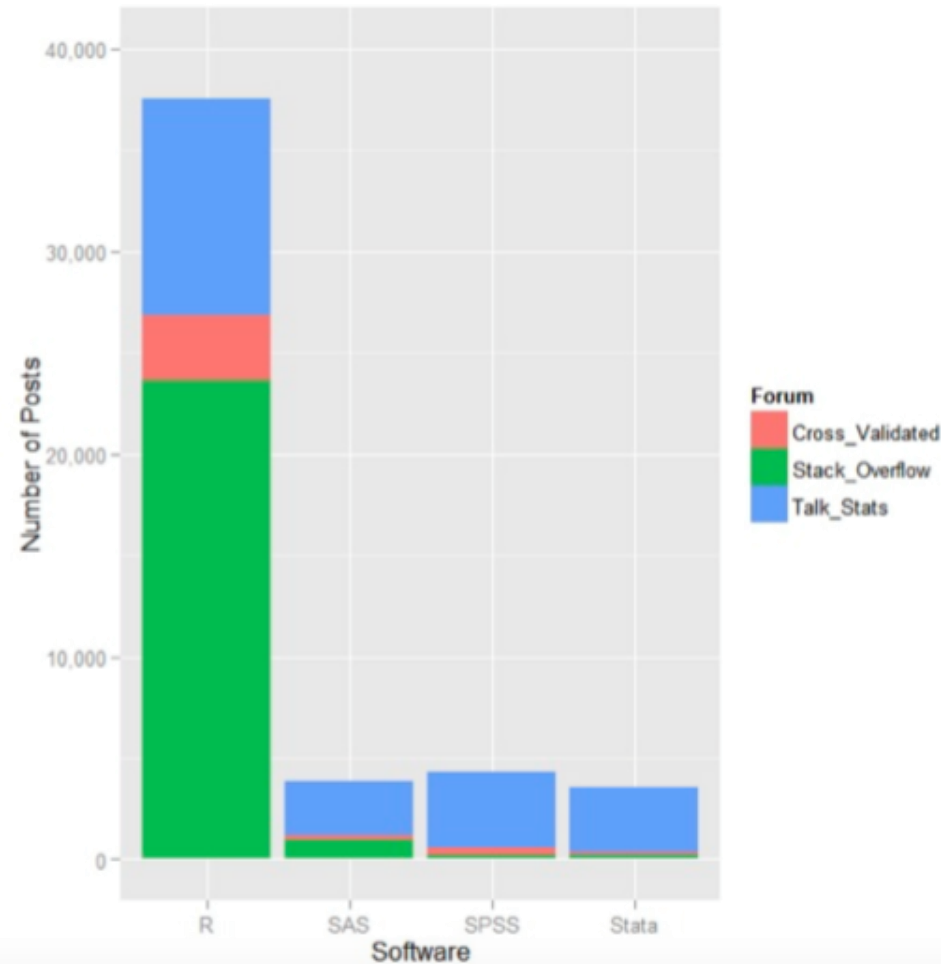
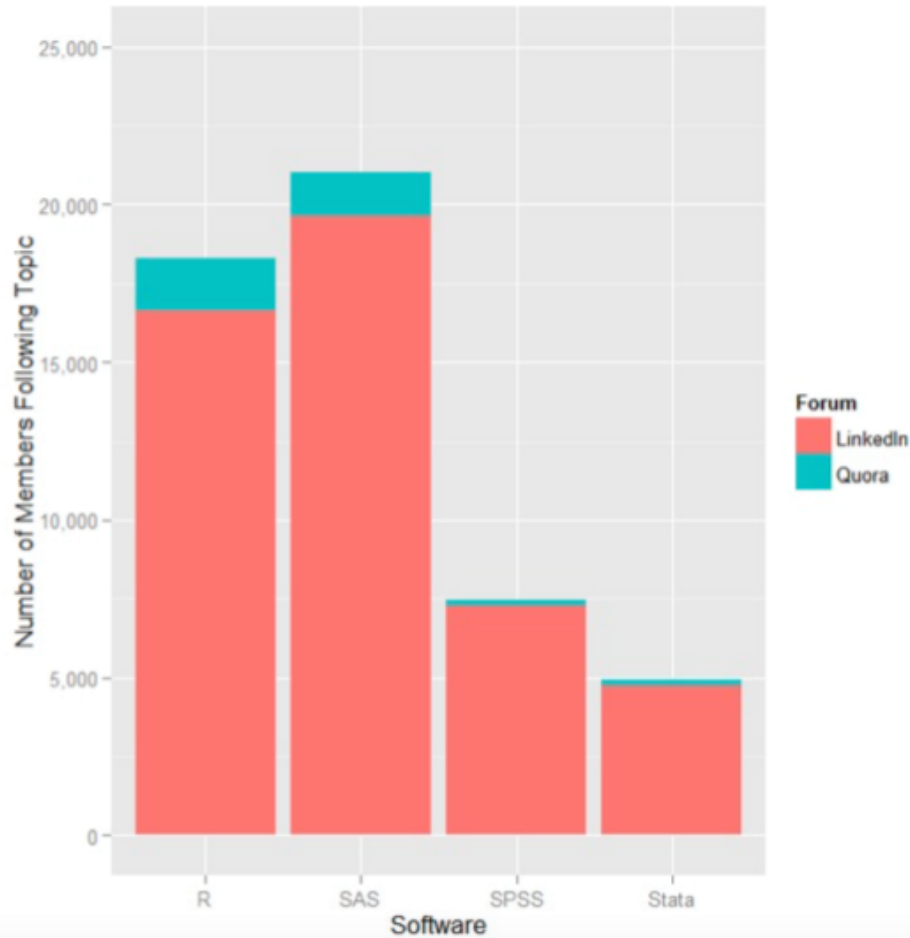
- R is a statistical programming language
 - Freedom to do your analysis, your way (it's a programming language, not a fixed suite of tools)
- It is freely available under the GNU license.
- R has great supports and large community around the world!
- Tools for reproducible research (Rstudio lets you publish your analyses via R markdown)
- The source code for the R software environment was written primarily in C, Fortran, and R.
- New statistical tools get published as R packages first



The rise of R

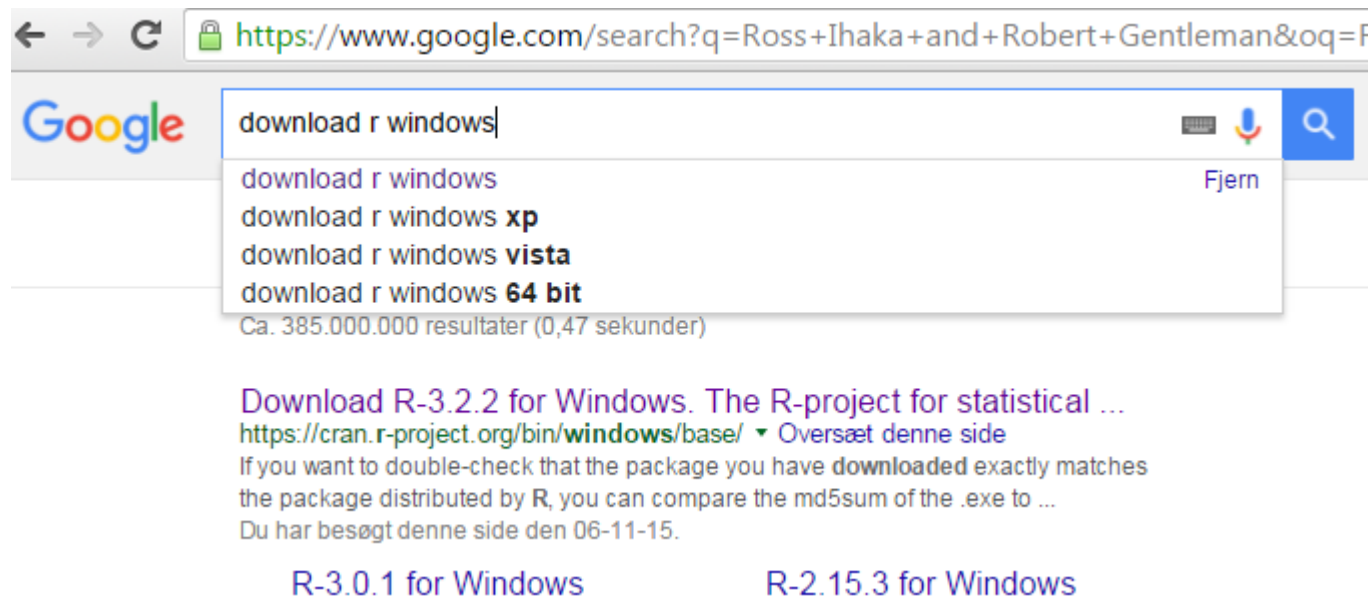


The rise of R



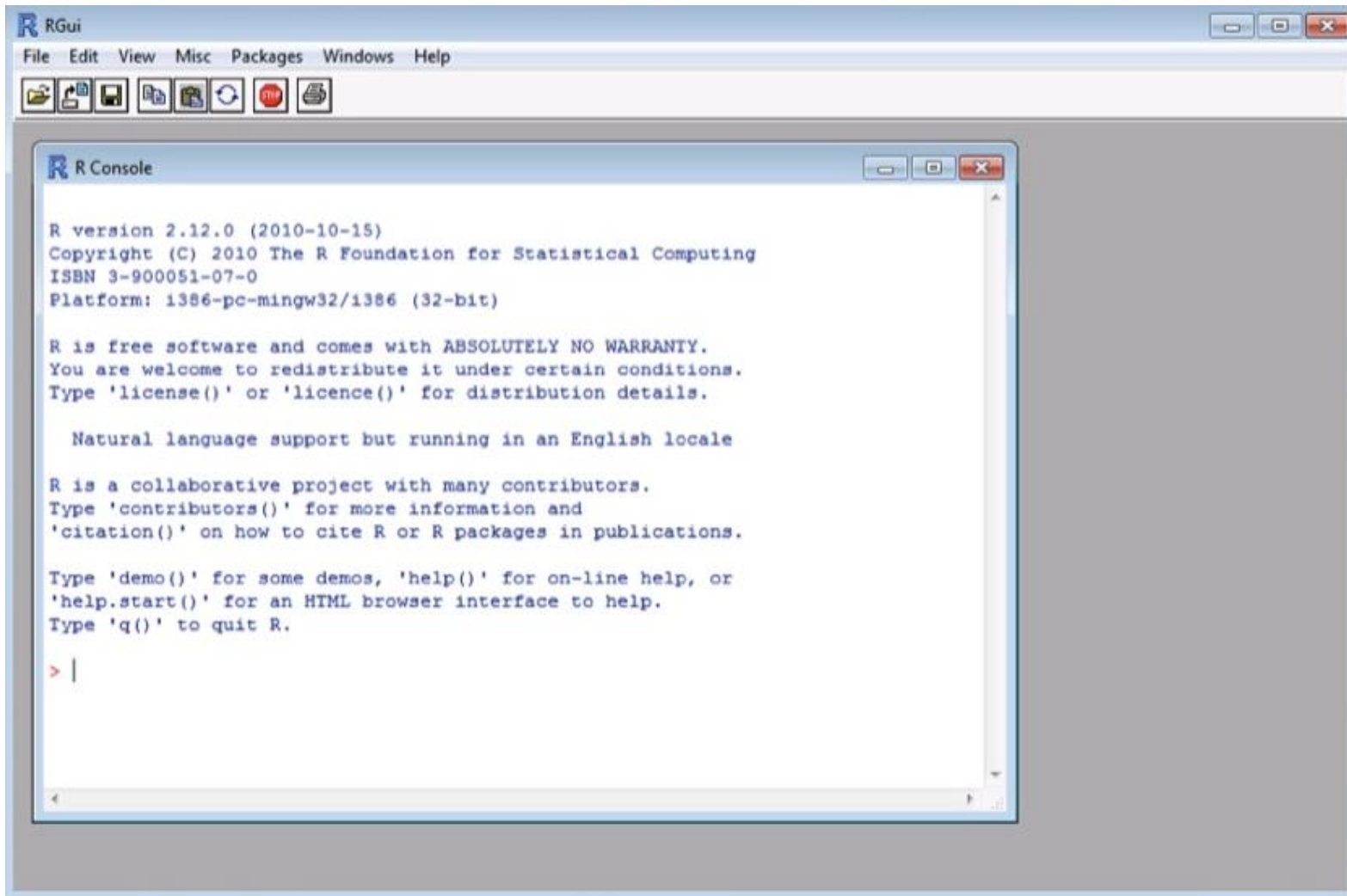
How to Access R?

- Find it on Google! It is free and downloadable!



Note: A R and Rstudio Installation Tutorial file is available in Workshop dashboard in SDAT courseware System.

R Screenshot (windows)

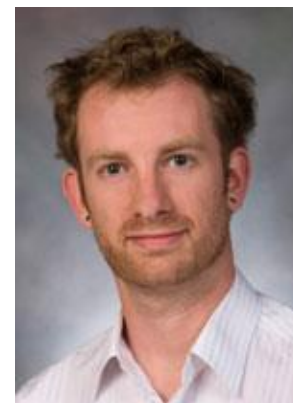


What is Rstudio Editor?

- Unlike SPSS, R is not an “application”.
- It is a programming language

Therefore:

- Use Rcommander if you want to force R to be SPSS
- But if
 - To make it enjoyable and comfortable working with R,
 - To facilitate many jobs in R,
 - To make it easy making R packages,
 - And for having much more fun and relaxation,

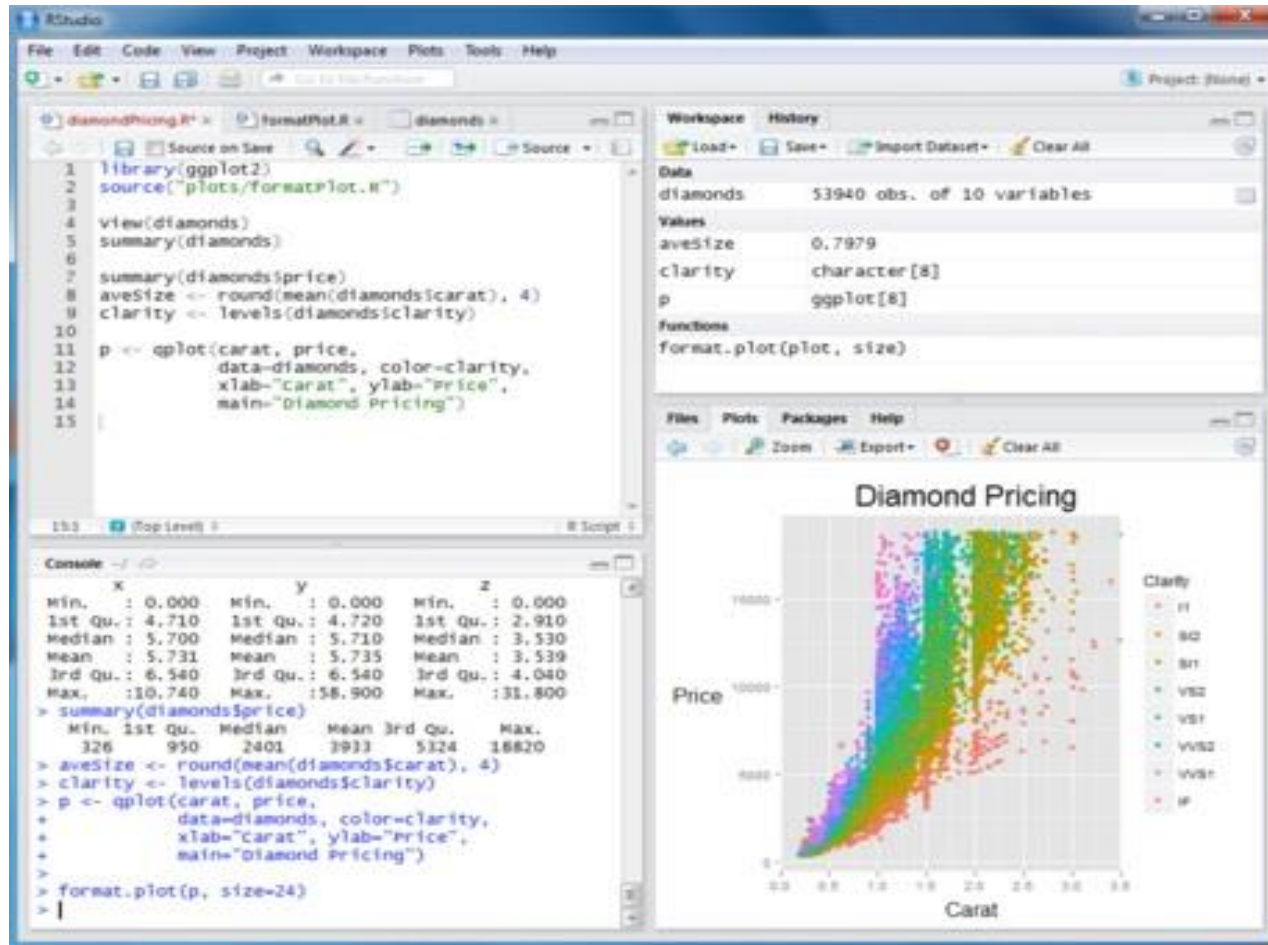


[Hadley Wickham](#)

One should DEFINITELY use RStudio! 😊



Rstudio Screenshot



Rstudio is a front end to R.

It's intended to provide the same user experience regardless of what kind of computer you're using

Rstudio webpage:
www.rstudio.org

Note: A Keyboard Shortcuts for Rstudio isfile is available in Workshop dashboard in SDAT courseware System.

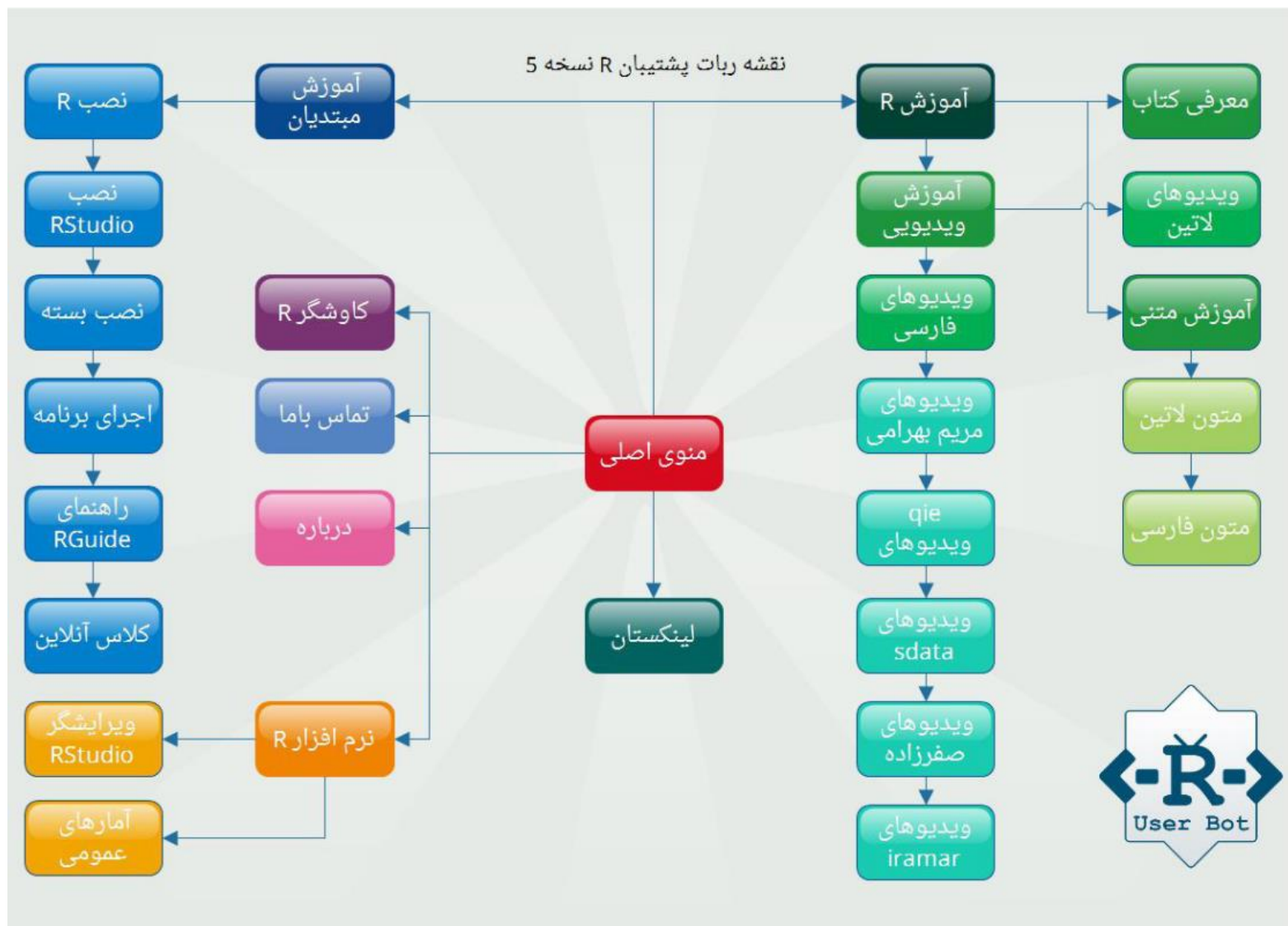


R User Bot by SDAT

R User Bot in Telegram Platform

- Created by SDAT Developer Team
- Team Leader: S. Jamal Mirkamali (Should you know him!!)
- For more information see: telegram.me/R_user_bot





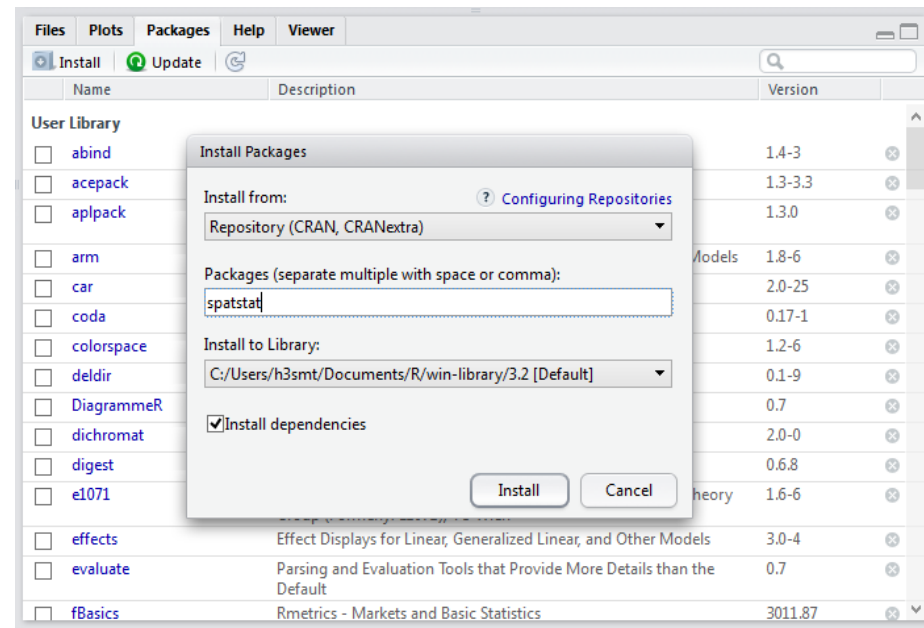
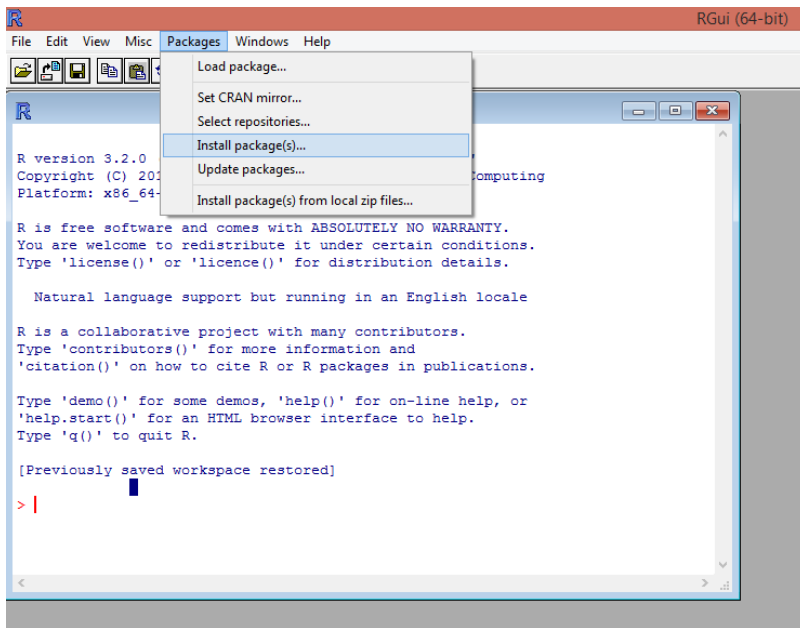
R Packages

Packages

- What is package?
 - A collection of R functions and data sets that someone has contributed to the R “ecosystem”
 - Packages extend the functionality of R: most of the value to R comes from the 5000+ packages out there
- Where do they come from?
 - Most packages are distributed “centrally” via CRAN (comprehensive R archive network)

Install R packages!

1. In the console, write: `install.packages ("package name")`
2. Use **Packages** → **Install** in RStudio!
3. Use **Packages** menu in R!



Call an R package; get help

- Calling an R package (R library)

> library(package name)

What does a library have?

Get help

- When we know the name of functions:

? Or **help("function name")**

- ?? Or **search.help("what we need")**

Some Exercises

- Find some useful R packages for Cognitive Sciences studies by googling!
- Try to install Rcmdr package.
- Find out what the package is used for
- Call the library to its (we will use this package).
- Find a function that can calculate the correlation of two variables by help of R

“Base” R comes with about 30 packages

```
> .packages(TRUE)
```

```
[1] "base"          "boot"          "class"         "cluster"
[5] "codetools"     "compiler"      "datasets"      "foreign"
[9] "graphics"      "grDevices"     "grid"          "KernSmooth"
[13] "lattice"       "MASS"          "Matrix"        "methods"
[17] "mgcv"          "nlme"          "nnet"          "rpart"
[21] "spatial"       "splines"       "stats"         "stats4"
[25] "survival"      "tcltk"         "tools"         "utils"
[29] "manipulate"
```


R as a calculator

Simple calculations

+	addition
-	subtraction
*	multiplication
/	division
^	taking powers

These are referred to as “**operators**”
(each operator is used to carry out
a particular kind of operation)

>	(6 - 4) / 2
[1]	1
>	6 - (4/2)
[1]	4

When performing multiple calculations
you can use parentheses to make sure R
performs the calculations in the desired order

(Note: Without parentheses, the order is: ^ first,
then * and / second (left to right), and then + and -
last (left to right). No-one remembers this at first.)

Logical Statements

`==` equality
`!=` inequality
`>` greater than
`>=` greater than or equal to
`<` less than
`<=` less than or equal to

`&` AND
`|` OR
`!` NOT

```
> 10 < 100  
[1] TRUE
```

```
> 2 + 2 == 5  
[1] FALSE
```

```
> (10 < 100) | (2 + 2 == 5)  
[1] TRUE
```

```
> (10 < 100) & (2 + 2 == 5)  
[1] FALSE
```

Funtions

Example: logarithms

```
> log( 32 )  
[1] 3.465736
```

- The function is called “log”.
- The 32 is the “**argument**” to the function.

Some functions that you might see on a scientific calculator

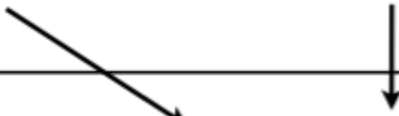
sqrt()	- Square root
round()	- Round a number
log()	- Logarithm
exp()	- Exponentiation
abs()	- Absolute value

Functions with multiple arguments

- Many functions can “take” more than one argument;
- Separate the arguments with commas.

Argument #1: The
number that needs to be
rounded

Argument #2: How many
digits to round it to?




```
> round( 3.1415, 2 )  
[1] 3.14
```

Arguments have names

- Most of the time, the arguments have “**names**”,
- You can use the names when typing commands

The number that needs to
be rounded is called **x**

The number of digits to
round to is called **digits**



```
> round( x = 3.1415, digits = 2 )  
[1] 3.14
```

(When you specify the names of the arguments, R doesn't care what order you type them in...but if you drop the names, then you must make sure you type them in the correct order)

Some arguments have defaults

- A lot of arguments have “**default values**”.
- If you don’t tell R what value to use, it uses the default

```
> round( x = 3.1415 )  
[1] 3
```

The default number of digits to round to is `digits = 0`, so that’s what R uses here

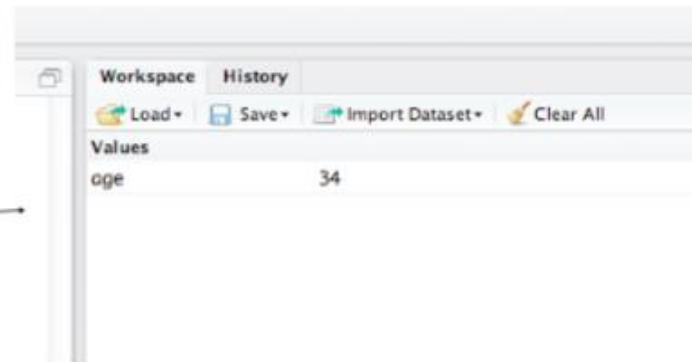
R Language

Let's talk in R language!

- Assigning values to an object notations with `<-` or `=`

```
> age <- 34
```

No output appears in the console, but the variable shows up in the Rstudio "workspace" panel



- R is sensitive to small and capital letters!
- Role of `;` is just separator in a long line of short sentences!

Working with variables

```
> # this is a comment...
> # (R ignores everything after the #)
>
> age <- 34 # store the value 34 as a
variable, called "age"
> print(age) # ask R to print it out...
[1] 34
> age
[1] 34
> age+10
[1] 44
```

There are different “classes” of variable

“Numeric” variables store numbers

```
> age <- 34
```

“Character” variables store text.

```
> name <- "dan"
```

The quote marks are used to tell R that this is text, and aren't part of the data itself.

```
> access <- TRUE
```

“Logical” variables store truth values (i.e., TRUE or FALSE).

Data classes

- **vector**: an ordered series of numbers, characters, ...
- **matrix**: You know that already! (you don't? Ask me please!)
- **data.frame**: much richer and more flexible than a matrix!
- **list**: a basket of everything you want!
- **array**: we don't need them here!

Vectors

- Vectors are variables that store multiple pieces of information
 - Create vectors using `c()`
 - Extract specific elements using `[]`

Assigning variables

numeric vectors:

```
> age <- c( 34, 2 )  
> age  
[1] 34 2
```

Vectors can get names

```
> active.time <- c( time1 = 0.52,  
time2 = 0.82, time3 = 0.99 )  
> active.time  
time1 time2 time3  
0.52 0.82 0.99
```

Empty data

NULL is a special “value” in R that means “this variable does not exist” or “it has no value”.

```
> noexist <- NULL
```

NA means “the variable exists (and in principle has a value), but the value is missing/unknown”

```
> missed <- NA
```

How about **NaN**?

Some more dummy codes!

Try it in the break

```
> V1 <- 1:10
> V2 <- c(4, 2, 6, 1, 5)
> V3 <- c("taghi", "naghi")
> V4 <- "NA" ; V5 <- NA
> is.na(V4)
[1] TRUE
> is.na(V5)
[1] TRUE
> class(V1); class(V2); class(V3); class(V4); class(V5)
> V1[c(1,4)]
[1] 4 6
```

Vectors and Matrices

- cbind: bind columns into a matrix
- rbind: bind rows into a matrix
- Create a new matrix with a vector:

```
> M <- matrix( c(11, 7, 3, 4, 10, 5, 6, 8, 2)
               , nrow = 3, ncol= 3)
> M[,1]  # first column of M
[1] 11  7  3
> M[1,]  # first row of M
[1] 11  4  6
```

Exercises!

```
> x <- 1:10  
> y <- seq(1, 20, by = 2)  
> z1 <- rbind(x, y)  
> z2 <- cbind(x, y)
```

- Find the class of z1 and z2.
- Find their dimensions.
- Try to find number of rows and number of columns in z1 separately (use R help!)
- Check if x and z1 are vectors or matrix.
- Extract second element of x. Replace it with 11.
- Extract (2, 3)-th element of z1. Replace it with 120.
- Extract those vectors of x which are less than 5.
- Sort x from its larger value to its smaller value.
- Order elements of x.
- Sort first row of z1.

Exercises!

- I have made a matrix using the following command:

```
> M <- matrix(c(11, 7, 3, 4, 10, 5, 6, 8, 2), nrow = 3)
```

- Give a name to its rows!
- Give a name to its columns!
- Call its second row without using its name!
- Call its second row using its name!
- Using R help, try to find more feature of this matrix.
- Check if `M[, 1]` is a matrix or not.
- Check if `x` in the previous example is a matrix or not.

Data frames

Data frames are the way R stores a typical data set

- It is very similar to a data set in SPSS
- It is a collection of variables “bundled” together
- Organised into a “case by variable” matrix
- Each row is a “case” and Each column is a named “variable”

```
> x2 <- c(4, 3, 2, 1)
> y2 <- c("I", "you", "we", "they")
> z3 <- as.data.frame(x, y)
> z4 <- as.matrix(x, y)
> class(z3)
> class(z4)
```


A data frame we'll see several times...

```
toydata <-
```

```
read.csv("https://smnajibi.github.io/statcomp/useR/toydata.csv")
```

```
toydata
```

	id	age	gender	treatment	hormone	happy	sad
1	1	25	male	control	6.7	2.00	6.12
2	2	24	male	drug1	38.5	3.36	3.53
3	3	25	male	drug2	25.0	3.40	4.82
4	4	28	male	control	98.4	5.69	0.34
5	5	23	male	drug1	42.4	4.56	4.48
6	6	28	male	drug2	20.3	2.89	4.57
7	7	25	female	control	18.5	3.18	4.82
8	8	29	female	drug1	65.2	4.78	2.24
9	9	21	female	drug2	56.4	4.51	2.64
10	10	26	female	control	55.7	3.90	2.71
11	11	19	female	drug1	41.9	2.83	2.94
12	12	30	female	drug2	54.1	3.45	1.87

It is a “bundle” of 7 vectors...

> toydata

	id	age	gender	treatment	hormone	happy	sad
1	1	25	male	control	6.7	2.00	6.12
2	2	24	male	drug1	38.5	3.36	3.53
3	3	25	male	drug2	25.0	3.40	4.82
4	4	28	male	control	98.4	5.69	0.34
5	5	23	male	drug1	42.4	4.56	4.48
6	6	28	male	drug2	20.3	2.89	4.57
7	7	25	female	control	18.5	3.18	4.82
8	8	29	female	drug1	65.2	4.78	2.24
9	9	21	female	drug2	56.4	4.51	2.64
10	10	26	female	control	55.7	3.90	2.71
11	11	19	female	drug1	41.9	2.83	2.94
12	12	30	female	drug2	54.1	3.45	1.87

	id	age	gender	treatment	hormone	happy	sad
1	1	25	male	control	6.7	2.00	6.12
2	2	24	male	drug1	38.5	3.36	3.53
3	3	25	male	drug2	25.0	3.40	4.82
4	4	28	male	control	98.4	5.69	0.34
5	5	23	male	drug1	42.4	4.56	4.48
6	6	28	male	drug2	20.3	2.89	4.57
7	7	25	female	control	18.5	3.18	4.82
8	8	29	female	drug1	65.2	4.78	2.24
9	9	21	female	drug2	56.4	4.51	2.64
10	10	26	female	control	55.7	3.90	2.71
11	11	19	female	drug1	41.9	2.83	2.94
12	12	30	female	drug2	54.1	3.45	1.87

The command `expt$age` tells R to look for a vector called `age` that is stored in the data frame called `expt`:

```
> toydata$age
```

```
[1] 25 24 25 28 23 28 25 29 21 26 19 30
```

Note that still `[]` rules of matrix also work

Lists: Very Briefly

```
> list( age = 34, name = c("Taghi", "Naghi"))
```

Notice that I'm mixing data of different types and different lengths? "age" is numeric and has two elements, "name" is character with one element.

- Exercise:

Give me another example of a list.

Check the length of that list.

Extract an element of that list.

Check the class of that element of the list.

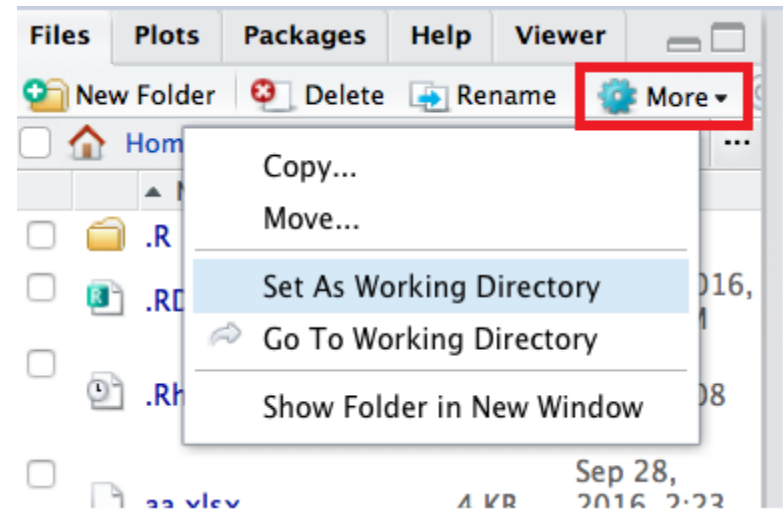
Import and Export data

Importing and Exporting Data

- **Importing data into R:**
 - Rstudio
 - Rcmdr package: txt, SPSS, Excel, ...
 - Foreign package: txt, SPSS, Excel, ...
 - gdata package: Excel files
 - No extra package is required! Try the following functions:
 - scan, read.table, read.csv and ... Can you name more?
- **Exporting R data sets:**
 - write, write.table, write.csv, ...

Working Directories

- Be careful when working with a wd.
- Be careful when saving objects with the same names in different wd's.
- `getwd()` and `setwd()`
- `load()`
- `ls()`
- `save()`
- `save.image()`



Import Data:

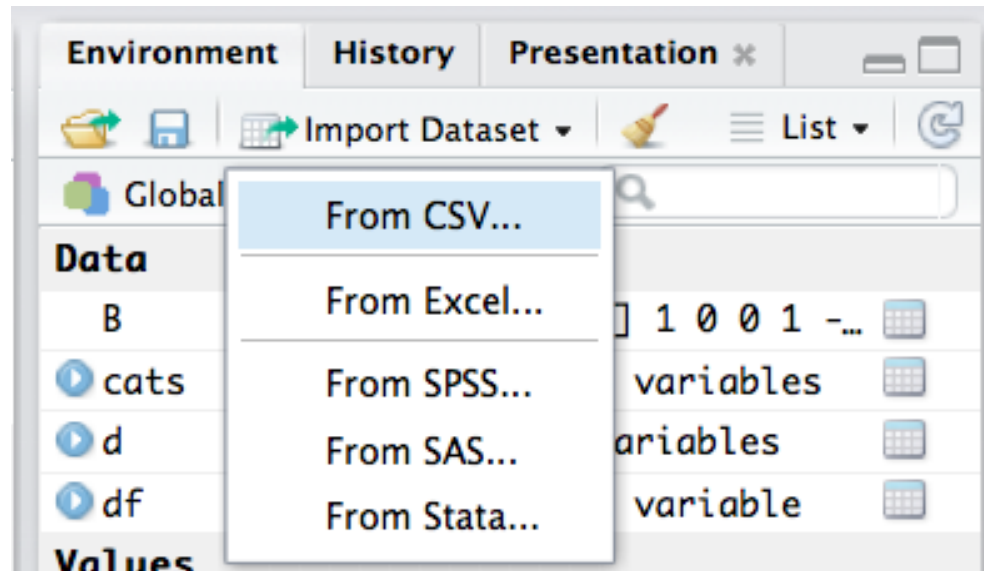
CSV is a standard “universal” format



```
1 "id","age","gender","treatment","hormone","happy","sad"
2 1,25,"male","control",6.7,2,6.12
3 2,24,"male","drug1",38.5,3.36,3.53
4 3,25,"male","drug2",25,3.4,4.82
5 4,28,"male","control",98.4,5.69,0.34
6 5,23,"male","drug1",42.4,4.56,4.48
7 6,28,"male","drug2",20.3,2.89,4.57
8 7,25,"female","control",18.5,3.18,4.82
9 8,29,"female","drug1",65.2,4.78,2.24
10 9,21,"female","drug2",56.4,4.51,2.64
11 10,26,"female","control",55.7,3.9,2.71
12 11,19,"female","drug1",41.9,2.83,2.94
13 12,30,"female","drug2",54.1,3.45,1.87
14
```

In R a CSV file
is imported as a
data frame

Importing CSV data using Rstudio



How input data into R

Import Text Data

File/Url: Browse...

Data Preview:

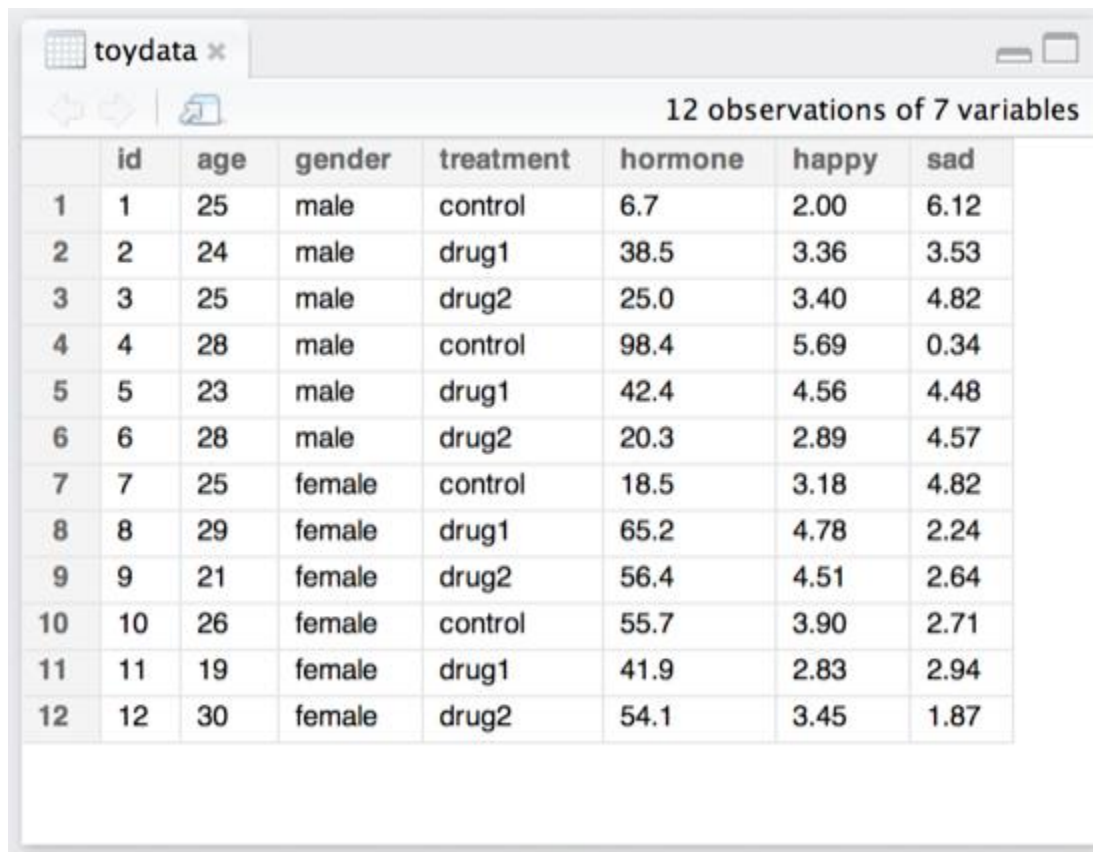
Import Options:

Name: ☒ First Row as Names Delimiter: Escape:
Skip: ☒ Trim Spaces Quotes: Comment:
☒ Open Data Viewer Encoding: NA:

Code Preview:

```
library(readr)
dataset <- read_csv(NULL)
View(dataset)
```

Import Cancel



toydata x

12 observations of 7 variables

	id	age	gender	treatment	hormone	happy	sad
1	1	25	male	control	6.7	2.00	6.12
2	2	24	male	drug1	38.5	3.36	3.53
3	3	25	male	drug2	25.0	3.40	4.82
4	4	28	male	control	98.4	5.69	0.34
5	5	23	male	drug1	42.4	4.56	4.48
6	6	28	male	drug2	20.3	2.89	4.57
7	7	25	female	control	18.5	3.18	4.82
8	8	29	female	drug1	65.2	4.78	2.24
9	9	21	female	drug2	56.4	4.51	2.64
10	10	26	female	control	55.7	3.90	2.71
11	11	19	female	drug1	41.9	2.83	2.94
12	12	30	female	drug2	54.1	3.45	1.87

Rstudio opens a tab showing you the contents of the data frame you just imported

- First set your working directory and put the file there

```
> toydata <- read.csv("toydata.csv")  
> View(toydata)  
> ls()
```

The primary file format used by R is .Rdata

- It is a saved workspace
- It contains whatever data sets, variables, functions etc that the workspace included when the file was created
- `save()` and `save.image` can be used to save part or whole of workspace

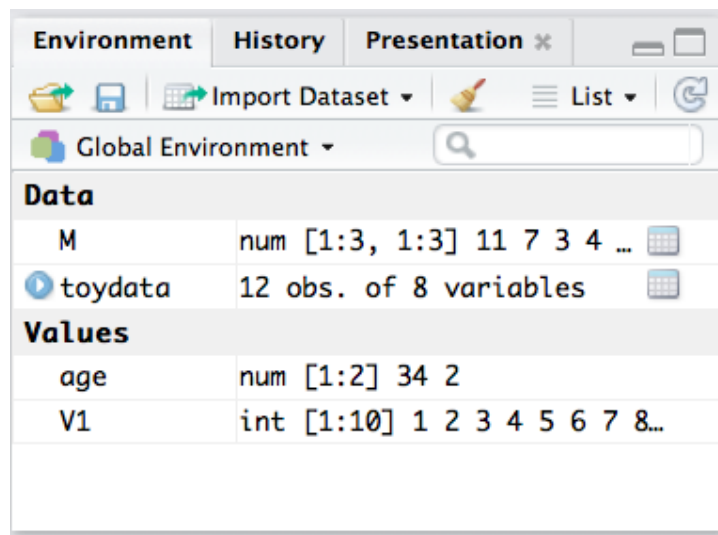
```
save(age,V1,M,toydate, file =  
      "first.save.Rdata")  
rm(list=ls) # delete all workspace variab
```

How to load

- Hard way: use the load() function manually

```
> load("first.save.Rdata")
```

- Easy way #1: double click on the .Rdata file in Finder/Explorer, and (as long as Rstudio is the default application for Rdata files) it will load automatically



Some more routine Functions

Some elementary math functions

- sum, min, max,
- mean, median, var, sd,
- factorial, det, diag, exp, log, abs, ...
- Aren't they familiar to MATLAB users?
- Why didn't we call any package when running them? In which packages they are?

Flow Control

R loops

- For loops

```
for (i in 1:10) {  
  print(log(i))  
}
```

- While loops

```
while (max(x) - 1 > 1e-06) {  
  x <- sqrt(x)  
}
```

- Compare them with MATLAB loops!

Conditional expressions!

- `if(expression) {
 do these
}`
- `If (expressions) {
 do S1
}
else if (expressions) {
 do S2
}
else {
 do S1
}`
- **Exercise:**
- **What is a different between `&` and `&&`? (Hint: check them using if expressions!)**

Writing R functions

- Structure:

```
MyFunction <- function(x,  
y) {  
  z <- x + y  
  return(z)  
}
```

Exercise

- Try to find out what “example” function does. Play with its arguments. Run it!

```
data <- matrix(rnorm(20), ncol = 2)
example <- function(data, a, b) {
  plot(data[,1], data[,2])
  abline(a = a, b = b)
}
```

Exercise

- Write a function that:
 1. gets n as the number of points you need to simulate
 2. Generate n uniform randomly scattered numbers between 0, 1
 3. Generate another n uniform randomly scattered numbers between 0, 1
 4. If these two variables are equal, generate another two n -element variables. O.W., make an scatter plot of these two random vectors vs each other with help of “plot” function.

How to access codes in R packages

I learned R from R!

A. Have you ever written the name of an R function and Enter it? No? Try it now!

B. Try to disclose R package source codes:

1. Go to CRAN
2. Choose a packages
3. Download its source file
4. Try to find its R functions and external codes!

R Graphic

R Graphs...

- get help of the following functions:
 - plot ()
 - barplot()
 - boxplot()
 - hist()
 - density()
- Investigate par() for Setting Graphical Parameters
- google about **ggplot2** package

😊We will back to it next two sessions 😊



Thank You