Basics of R

Workshop: Analysis of Longitudinal Data

12th Nov 2024

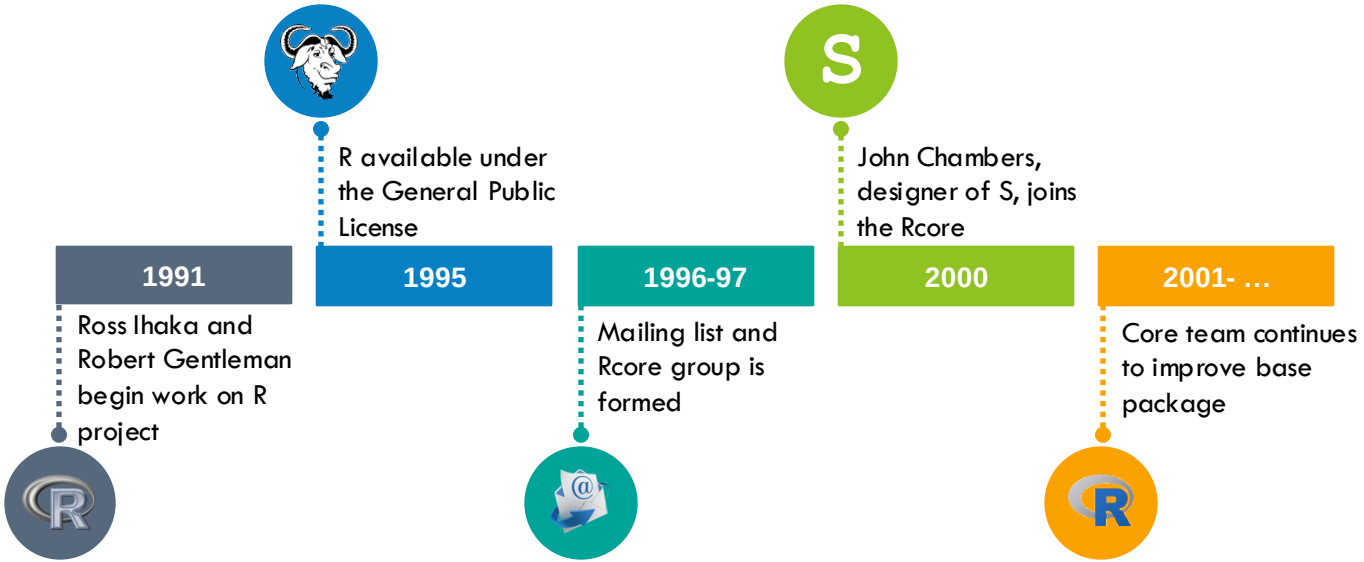
Jaroslaw Harezlak
Armando Teixeira-Pinto

THE UNIVERSITY OF SYDNEY

Outline

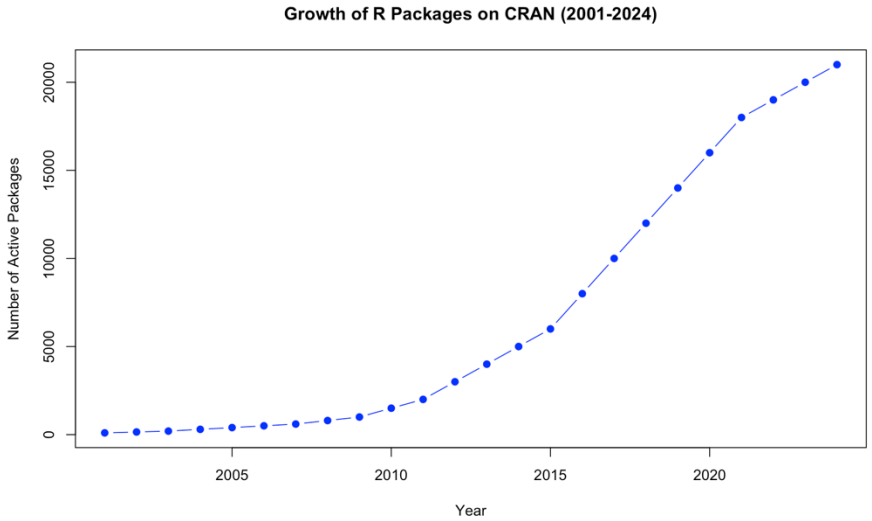
- R and RStudio
- Objects and functions
- Managing data
- Packages/Libraries

Some history

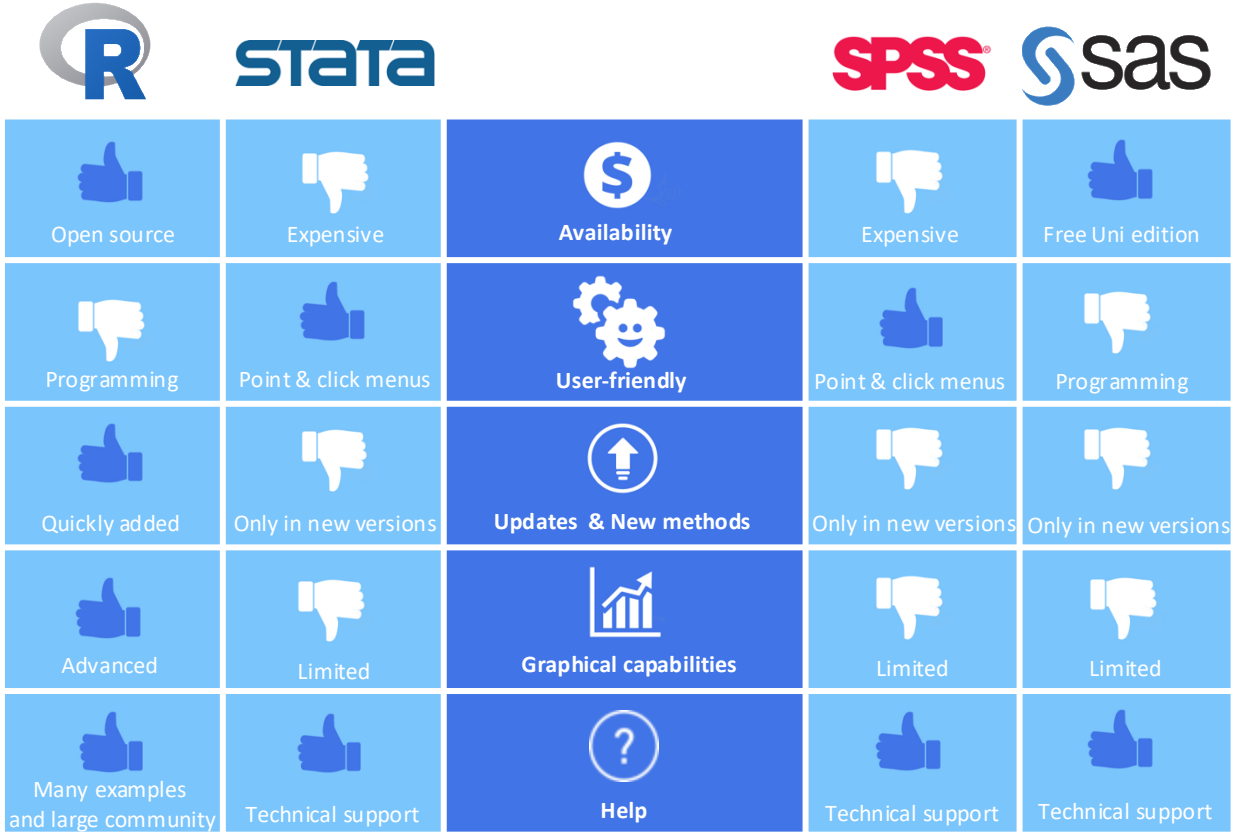


Current state

- Version 4.4.2
- Almost 22,000 packages available in CRAN



Statistical software



Should you learn R?

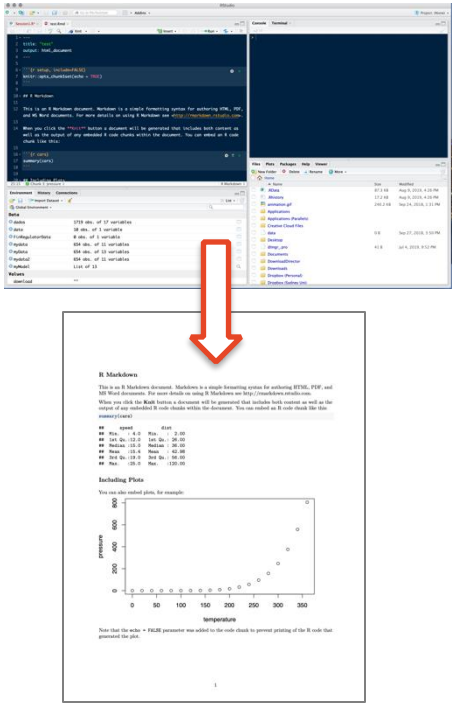
- Free
- Nice graphics
- Use advance methods
- Programming (e.g., implement new methods or run simulations)
- Reproducible research
- Look smart!

- But a steeper learning curve!

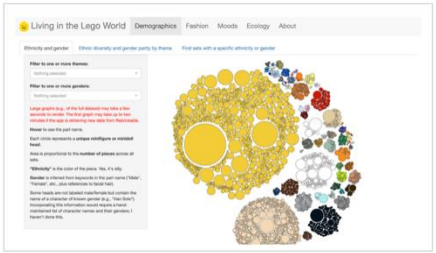
R is a programming language for statistical computing and graphics

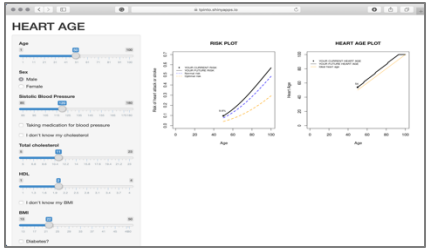
R “extensions”

- Rmarkdown / quarto

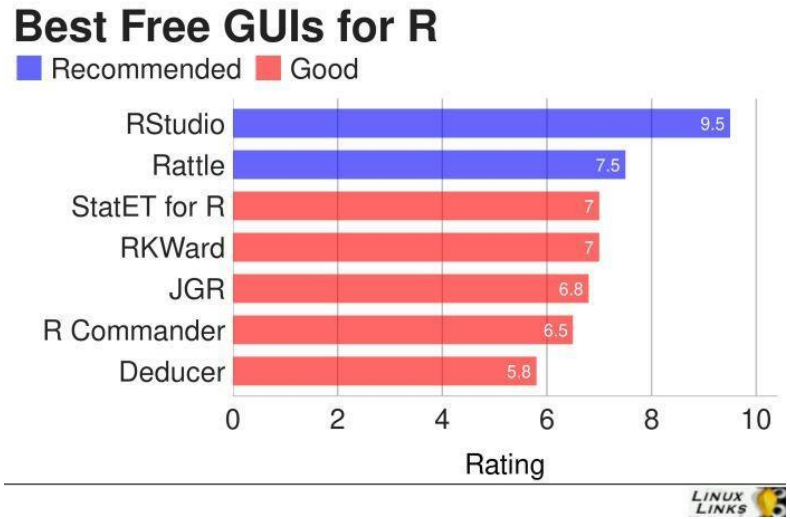


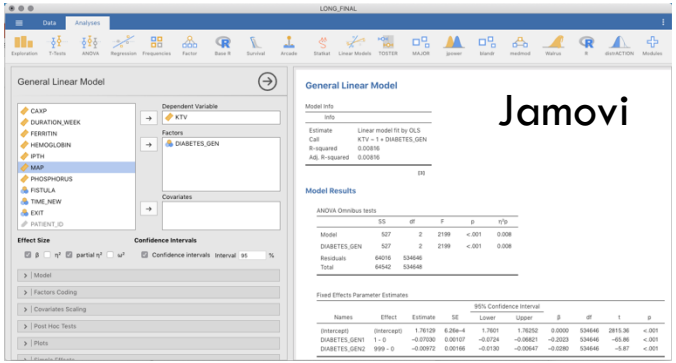
- R shiny (https://tpinto.shinyapps.io/shinyapp/)



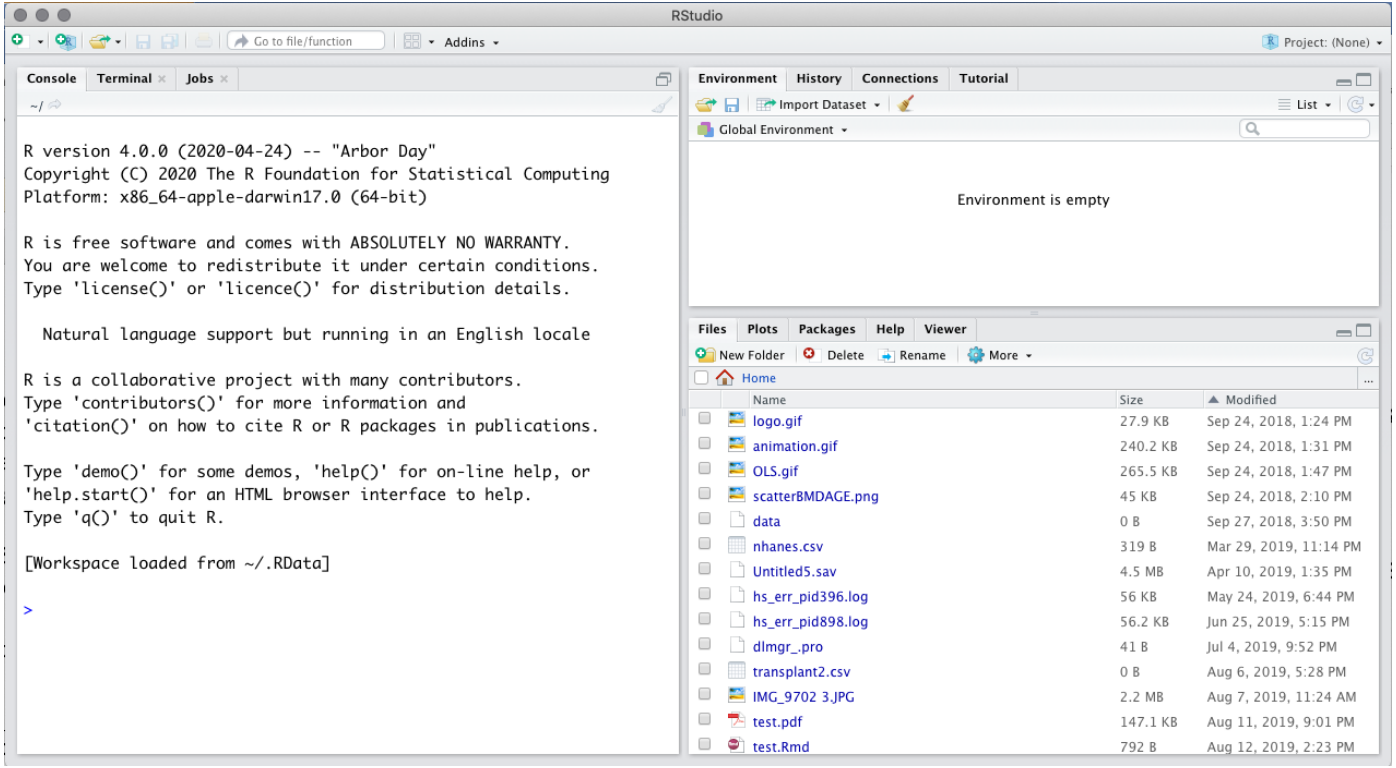


RStudio

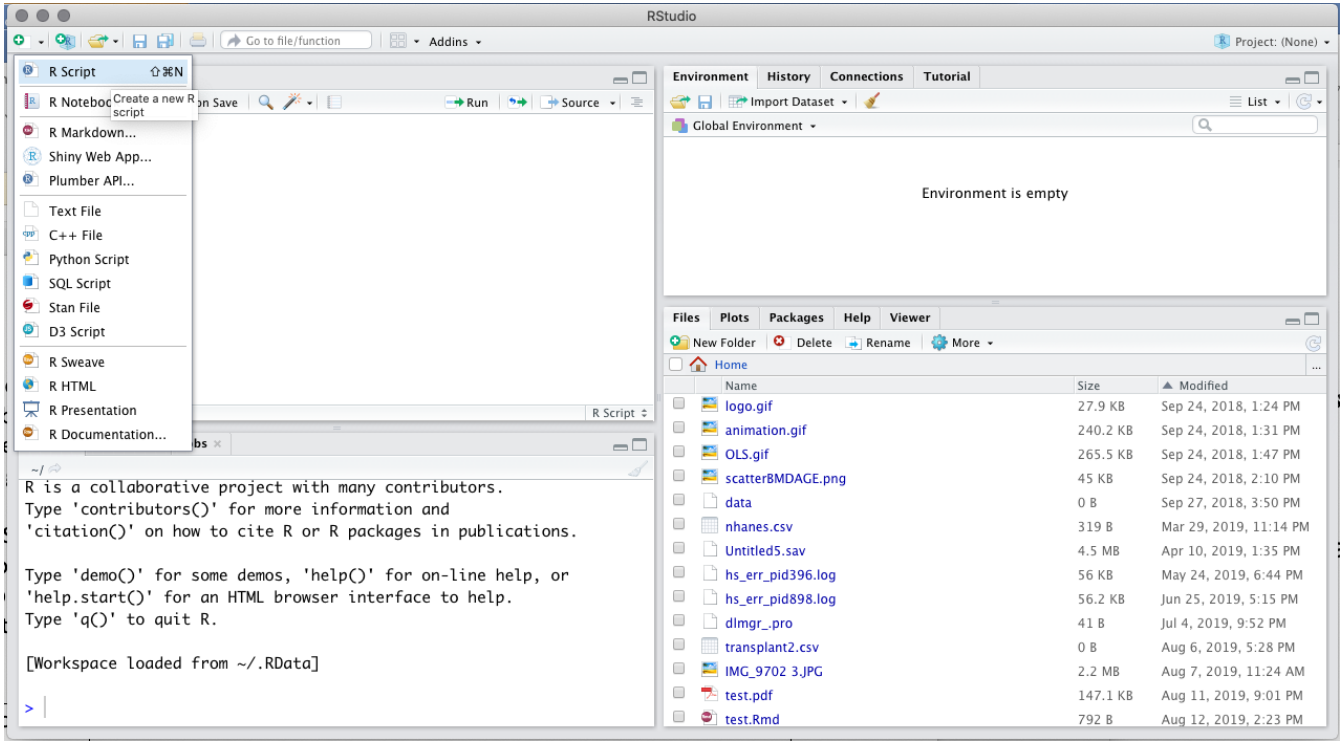

Best Free GUIs for R
Recommended
Good
RStudio
Rattle
StatET for R
RKWard
JGR
R Commander
Deducer
9.5
7.5
7
7
6.8
6.5
5.8
0
2
4
6
8
10
Rating
LINUX LINKS


Jamovi
General Linear Model
Model Results

RStudio


RStudio
Go to file/function
Addins
Project: (None)
Console
Terminal
Jobs
~/
R version 4.0.0 (2020-04-24) -- "Arbor Day"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
>
Environment
History
Connections
Tutorial
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
logo.gif 27.9 KB Sep 24, 2018, 1:24 PM
animation.gif 240.2 KB Sep 24, 2018, 1:31 PM
OLS.gif 265.5 KB Sep 24, 2018, 1:47 PM
scatterBMDAGE.png 45 KB Sep 24, 2018, 2:10 PM
data 0 B Sep 27, 2018, 3:50 PM
nhanes.csv 319 B Mar 29, 2019, 11:14 PM
Untitled5.sav 4.5 MB Apr 10, 2019, 1:35 PM
hs_err_pid396.log 56 KB May 24, 2019, 6:44 PM
hs_err_pid898.log 56.2 KB Jun 25, 2019, 5:15 PM
dlmgr_.pro 41 B Jul 4, 2019, 9:52 PM
transplant2.csv 0 B Aug 6, 2019, 5:28 PM
IMG_9702 3.JPG 2.2 MB Aug 7, 2019, 11:24 AM
test.pdf 147.1 KB Aug 11, 2019, 9:01 PM
test.Rmd 792 B Aug 12, 2019, 2:23 PM

RStudio


RStudio
Go to file/function
Addins
Project: (None)
R Script ⇧⌘N
R Notebook
Create a new R script
R Markdown...
Shiny Web App...
Plumber API...
Text File
C++ File
Python Script
SQL Script
Stan File
D3 Script
R Sweave
R HTML
R Presentation
R Documentation...
Run
Source
R Script
Environment
History
Connections
Tutorial
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
logo.gif 27.9 KB Sep 24, 2018, 1:24 PM
animation.gif 240.2 KB Sep 24, 2018, 1:31 PM
OLS.gif 265.5 KB Sep 24, 2018, 1:47 PM
scatterBMDAGE.png 45 KB Sep 24, 2018, 2:10 PM
data 0 B Sep 27, 2018, 3:50 PM
nhanes.csv 319 B Mar 29, 2019, 11:14 PM
Untitled5.sav 4.5 MB Apr 10, 2019, 1:35 PM
hs_err_pid396.log 56 KB May 24, 2019, 6:44 PM
hs_err_pid898.log 56.2 KB Jun 25, 2019, 5:15 PM
dlmgr_.pro 41 B Jul 4, 2019, 9:52 PM
transplant2.csv 0 B Aug 6, 2019, 5:28 PM
IMG_9702 3.JPG 2.2 MB Aug 7, 2019, 11:24 AM
test.pdf 147.1 KB Aug 11, 2019, 9:01 PM
test.Rmd 792 B Aug 12, 2019, 2:23 PM
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
>

Objects

- R is an **object-oriented** programming language
- Examples of "objects":
  - **number, string, logic value**

    7 *"USyd"* *TRUE*

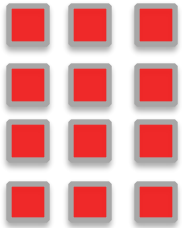
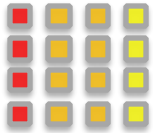
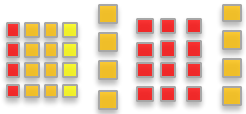
    

    Data values
  - **vector (variable), matrix, data frame, list**

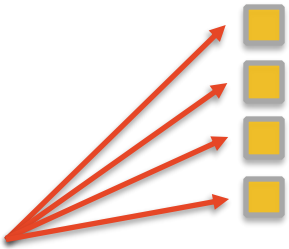
    Data structures
  - **results of a model, plot, table, …**

Objects

Vector	Matrix	Data frame	list
			

Objects

Vector	Matrix	Data frame	list



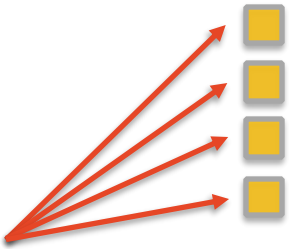
Same type of data values

Age

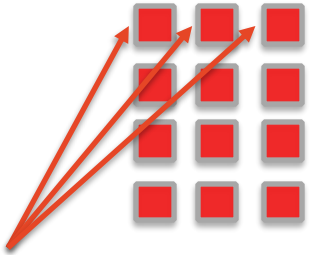
32
54
87
54
37
...

Objects

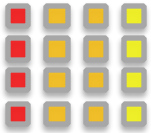
Vector	Matrix	Data frame	list

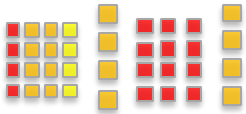


Same type of data values



Same type of vectors





Age

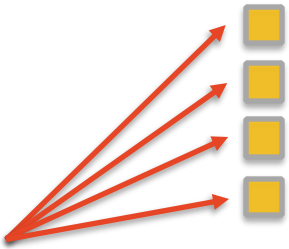
32
54
87
54
37
...

Correlation matrix

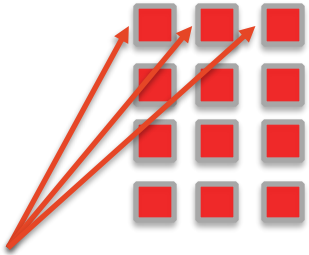
$$\begin{pmatrix} 1 & 0.32 & 0.43 \\ 0.32 & 1 & 0.27 \\ 0.43 & 0.27 & 1 \end{pmatrix}$$

Objects

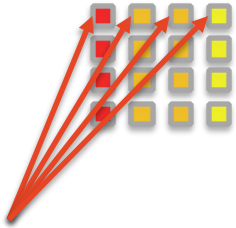
Vector	Matrix	Data frame	list



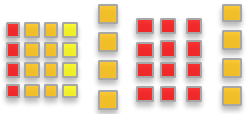
Same type of data values



Same type of vectors



Possibly different type of vectors



Age

32
54
87
54
37
...

Correlation matrix

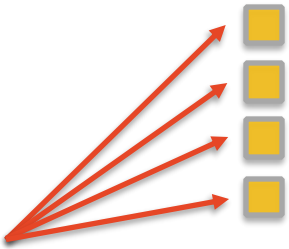
$$\begin{pmatrix} 1 & 0.32 & 0.43 \\ 0.32 & 1 & 0.27 \\ 0.43 & 0.27 & 1 \end{pmatrix}$$

Dataset

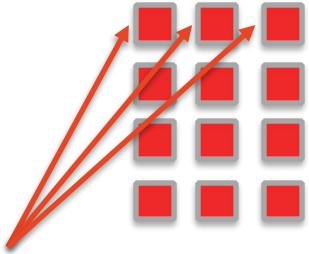
id	sex	sc1m	egfr_1m	cig	lung	coronary	c
1	M	134.74573	52.120399	N	N	N	
2	F	98.27387	49.141222	N	N	N	
3	F	85.16724	68.361577	N	Y	Y	
4	M	NA	NA	N	N	Y	
5	M	476.46724	9.899353	F	N	Y	
6	F	NA	NA	F	N	N	
7	F	162.64212	31.572802	N	N	N	
8	F	377.51957	13.208504	F	N	N	
9	M	100.00430	80.077657	C	N	N	

Objects

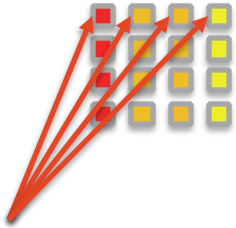
Vector	Matrix	Data frame	list



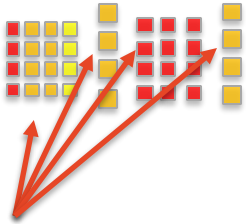
Same type of data values



Same type of vectors



Possibly different type of vectors



Possibly different type of objects

Age

32
54
87
54
37
...

Correlation matrix

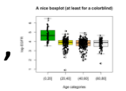
1	0.32	0.43
0.32	1	0.27
0.43	0.27	1

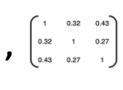
Dataset

id	sex	sc1m	egfr_1m	cig	lung	coronary	c
1	M	134.74573	52.120399	N	N	N	
2	F	98.27387	49.141222	N	N	N	
3	F	85.16724	68.361577	N	Y	Y	
4	M	NA	NA	N	N	Y	
5	M	476.46724	9.899353	F	N	Y	
6	F	NA	NA	F	N	N	
7	F	162.64212	31.572802	N	N	N	
8	F	377.51957	13.208504	F	N	N	
9	M	100.00430	80.077657	C	N	N	

List

32
54
87
54
37
...





1	0.32	0.43
0.32	1	0.27
0.43	0.27	1

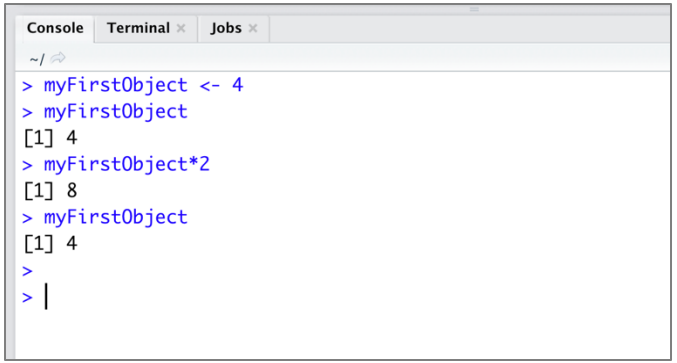
Objects

- You can name give a name to the ”objects”
- For example,

```
myFirstObject <- 4
```

- This creates an object (numeric) with value 4 and name `myFirstObject` (the ”<-” (“less then”, “minus”) makes suggests an arrow)

```
#Try in R:

myFirstObject <- 4
myFirstObject
myFirstObject*2
myFirstObject
```

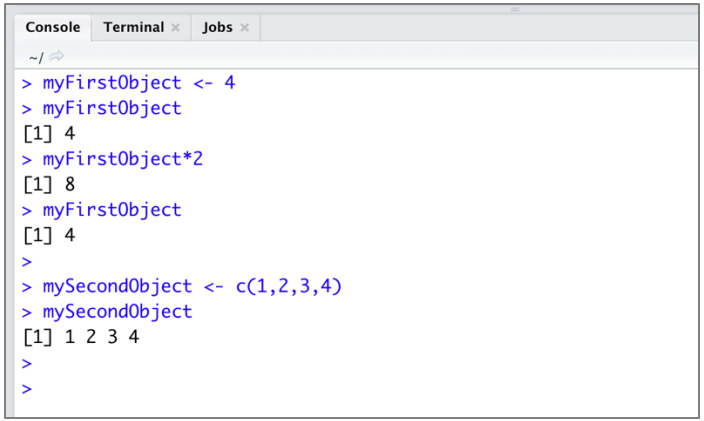


Functions

- Now, a vector

```
mySecondObject <- c(1,2,3,4)
```

```
#Try in R:

mySecondObject <- c(1,2,3,4)
mySecondObject
```



Functions

- Now, a vector

```
mySecondObject <- c(1,2,3,4)
```

- And we can now apply functions to the objects

function(object)

- For example, we can compute the mean

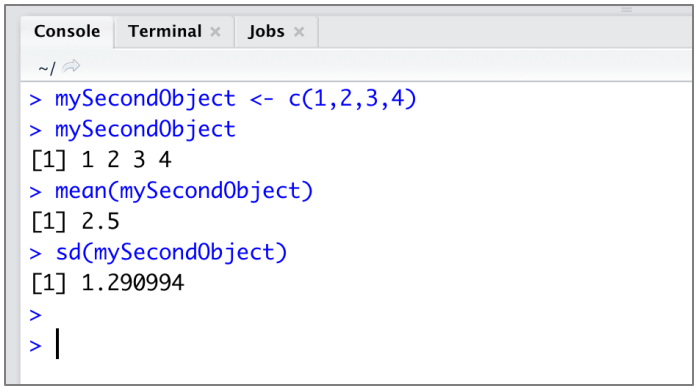
function object

```
mean(mySecondObject)
```

- And the standard deviation

```
sd(mySecondObject)
```

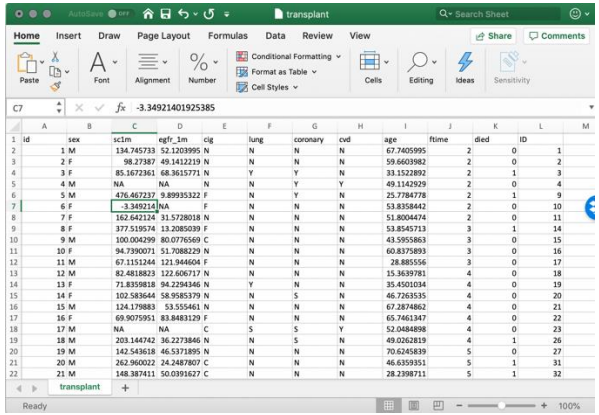
```
#Try in R:

mySecondObject <- c(1,2,3,4)
mySecondObject
mean(mySecondObject)
sd(mySecondObject)
```



Data management

```
#Try in R:

#Usually, the file would be stored locally
#myData <- read.csv("c:\\yourpath\transplant.csv")

myData <-
read.csv("https://www.dropbox.com/scl/fi/9qidag3d5odyvlqcrixln/transplant.csv?rlkey=livvep6peuz2xpvmr2qcqt84b&st=wjoqpefr&dl=1")
myData
```

– Let's now load a dataset `transplant.csv`




```
Console   Terminal   Jobs
~/
> myData <- read.csv("https://www.dropbox.com/s/fsle1fd9qr53iuo/transplant.csv?dl=1")
> myData
   id sex      sc1m    egfr_1m cig lung coronary  cvd       age ftime died
1   1   M 134.74573  52.120399   N    N        N    N 67.740600     2    0
2   2   F  98.27387  49.141222   N    N        N    N 59.660398     2    0
3   3   F  85.16724  68.361577   N    Y        Y    N 33.152289     2    1
4   4   M        NA         NA   N    N        Y    Y 49.114293     2    0
5   5   M 476.46724   9.899353   F    N        Y    N 25.778478     2    1
6   6   F        NA         NA   F    N        N    N 53.835844     2    0
7   7   F 162.64212  31.572802   N    N        N    N 51.800447     2    0
8   8   F 377.51957  13.208504   F    N        N    N 53.854571     3    1
9   9   M 100.00430  80.077657   C    N        N    N 43.595586     3    0
10 10   F  94.73901  51.708823   N    N        N    N 60.837589     3    0
11 11   M  67.11512 121.944604   F    N        N    N 28.885556     3    0
12 12   M  82.48188 122.606717   N    N        N    N 15.363978     4    0
```

Data management

- We can also apply ***functions*** to this new ”object” `myData`
- For example,

`summary(myData)`

```
#Try in R:
#let's first check that myData is a data frame

class(myData)
summary(myData)
```

Console | Terminal | Jobs

```
> class(myData)
[1] "data.frame"
> summary(myData)
       id             sex                 sc1m            egfr_1m
 Min.   :  1.0   Length:654         Min.   :  16.59   Min.   :  2.476
 1st Qu.:164.2   Class :character   1st Qu.: 101.41   1st Qu.: 36.967
 Median :327.5   Mode  :character   Median : 131.96   Median : 50.664
 Mean   :327.5                      Mean   : 153.40   Mean   : 57.793
 3rd Qu.:490.8                      3rd Qu.: 174.59   3rd Qu.: 63.431
 Max.   :654.0                      Max.   :1221.81   Max.   :649.355
                                    NA's   :6         NA's   :6
     cig                lung              coronary           cvd
 Length:654         Length:654         Length:654         Length:654
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

Data management

- We can also apply ***functions*** to this new "object" `myData`
- For example,

`summary(myData)`

Notice that if we apply the function `summary` to a different type of object, e.g. a model, it will produce a different output
R (usually) understands the context of the object and function

```
#Try in R:
#let's first check that myData is a data frame

class(myData)
summary(myData)

# "summary" for a model will produce a different output

myModel <- lm(egfr_1m ~ age, data = myData)
summary(myModel)
```

```
Console  Terminal  Jobs
~/
> myModel <- lm(egfr_1m ~ age, data = myData)
> summary(myModel)

Call:
lm(formula = egfr_1m ~ age, data = myData)

Residuals:
   Min     1Q Median     3Q    Max
-71.34 -22.57  -4.28  13.80 542.07

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 115.4282     5.0182   23.00   <2e-16 ***
age          -1.3263     0.1083  -12.25   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 44.34 on 646 degrees of freedom
  (6 observations deleted due to missingness)
Multiple R-squared:  0.1884,    Adjusted R-squared:  0.1872
F-statistic:   150 on 1 and 646 DF,  p-value: < 2.2e-16
```

Data management

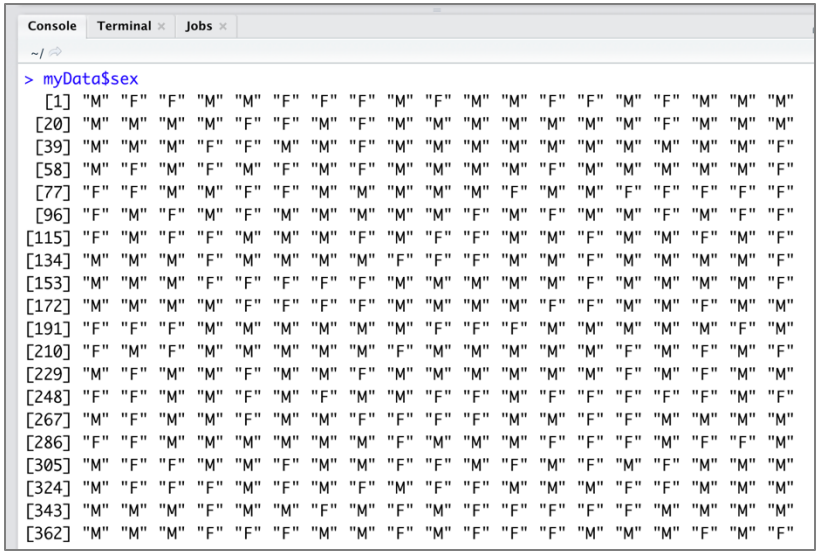
- Each variable in `myData` is also an object

  id, sex, sc1m, egfr_1m,…

- The variable `sex`, e.g., is identified as

  `myData$sex`

```
#Try in R:

myData$sex
```



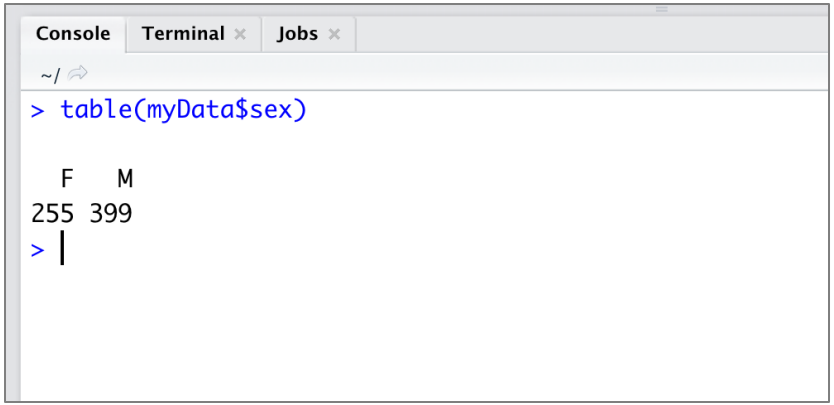
Data management

- **Exercise:** Identify the variable age and get the mean age

```
#Try in R:
mean(myData$age)

myData$sex
table(myData$sex)
```

- Let's use the function `table` to get frequencies of each category of sex

```
table(myData$sex)
function( object )
```



- **Exercise:** How many patients died (variable `died`)?

Data management

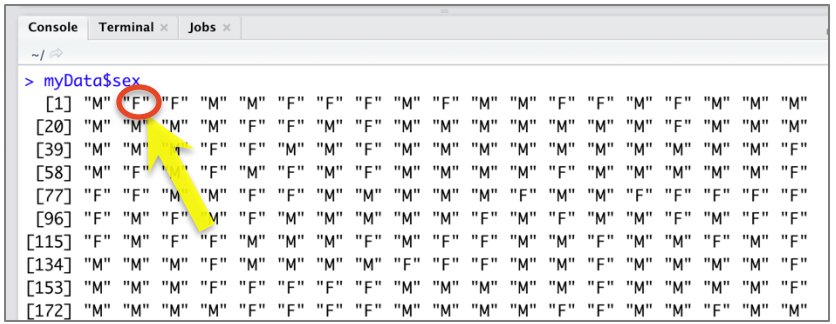
- Specific observation (values) in the variable can be identified using [ ]
- If you want the first four observations of the variable `sex`

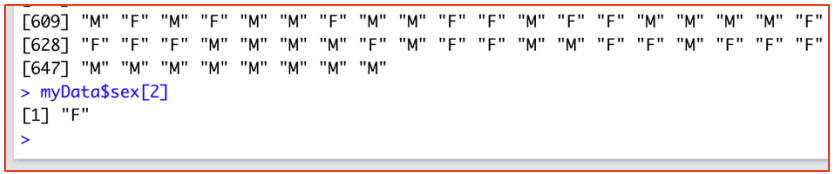
  `myData$sex[2]`

- If you want the first four observations of the variable `sex`

  `myData$sex[c(1,2,3,4)]`

```
#Try in R:

myData$sex[2]
myData$sex[c(1,2,3,4)]
```

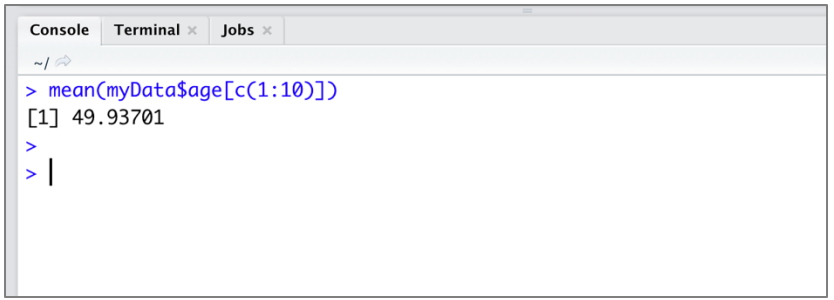




Data management

- **Exercise**: Get the mean age of the 10 first individuals

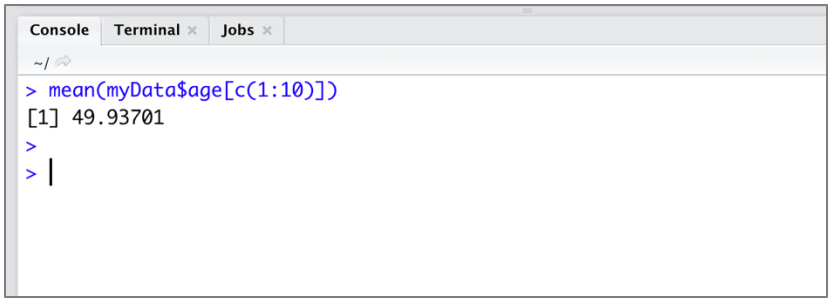
```
#Try in R:

#c(1:10) will give all 1,2, …, 10
mean(myData$age[c(1:10)])
```



Data management

- **Exercise**: Get the mean age of the 10 first individuals
- Now, let's try something useful! Let's compute the mean age of men
- First, let's see how do we get the ages of every men

```
#Try in R:

#c(1:10) will give all 1,2, …, 10
mean(myData$age[c(1:10)])
```



Data management

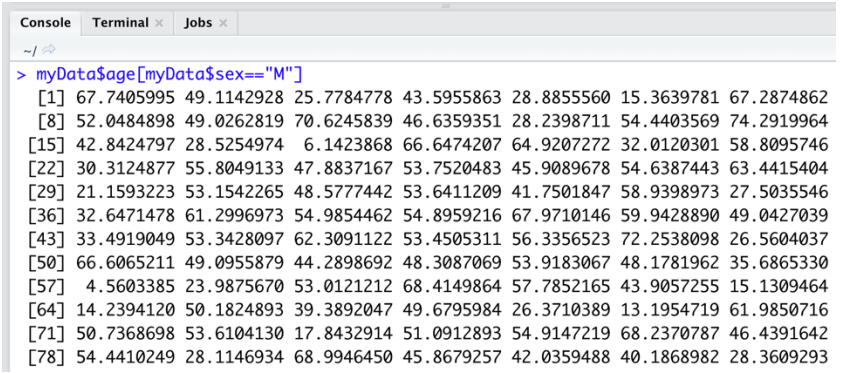
- **Exercise**: Get the mean age of the 10 first individuals
- Now, let's try something useful! Let's compute the mean age <u>of men</u>
- First, let's see how do we get the ages of every men

equal

`myData$age[myData$sex =="M"]`

Values of age in the positions where the variable sex is M

```
#Try in R:

#c(1:10) will give all 1,2, …, 10
mean(myData$age[c(1:10)])

myData$age[myData$sex=="M"]
```



Data management

- **Exercise**: Get the mean age of the 10 first individuals
- Now, let's try something useful! Let's compute the mean age of men
- First, let's see how do we get the ages of every men

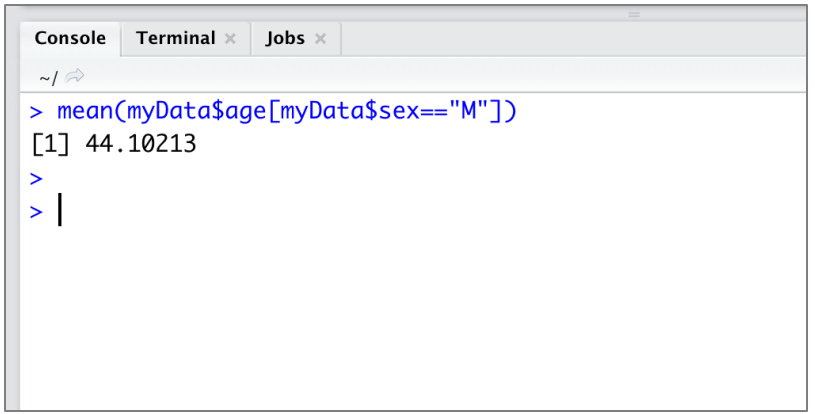
```
myData$age[myData$sex =="M"]
```

Values of age in the positions where the variable sex is M

- So, the only thing we need is to apply the mean function to the above

```
mean(myData$age[myData$sex=="M"])
```

```
#Try in R:

#c(1:10) will give all 1,2, …, 10
mean(myData$age[c(1:10)])

myData$age[myData$sex=="M"]
mean(myData$age[myData$sex=="M"])
```



Data management

- **Exercise**: get the number of men and women, older than 50 years old

Data management

– **Exercise**: get the number of men and women, older than 50 years old

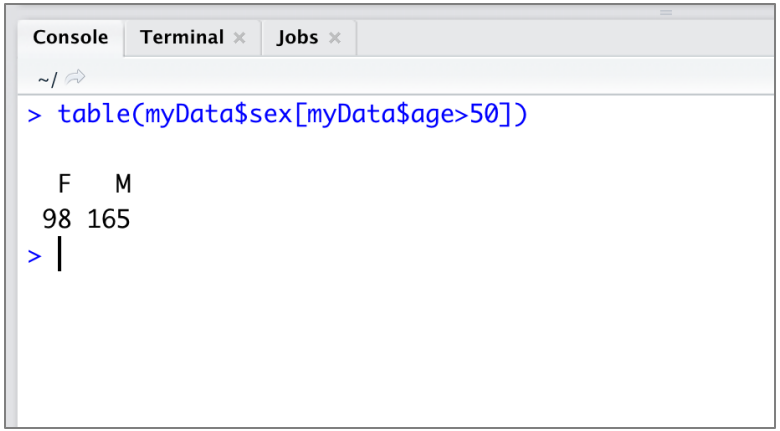
```
table(myData$sex[myData$age>50])
```

```
#Try in R:

table(myData$sex[myData$age>50])
```

The function `with()` simplifies the writing of the expression above:

```
with(myData, table(sex[age>50]))
```

For data management there is another way of writing commands (different grammar) that is very popular. If you want to know more about this, https://dplyr.tidyverse.org



Data management

– **Exercise**: get the number of men and women, older than 50 years old

```
table(myData$sex[myData$age>50])
```

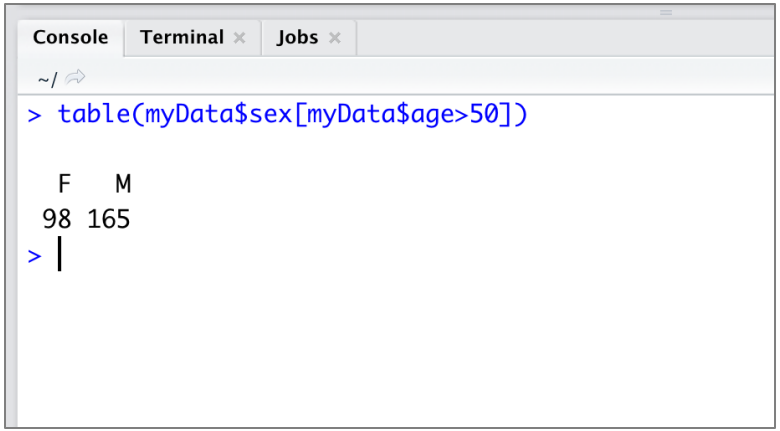
```
#Try in R:

table(myData$sex[myData$age>50])
```

The function `with()` simplifies the writing of the expression above:

```
with(myData, table(sex[age>50]))
```

For data ma
writing comm
very popula
this, https://

```
myData %>%
  filter(age>50) %>%
  group_by(sex) %>%
  summarize(Freq=n())
```



Data management

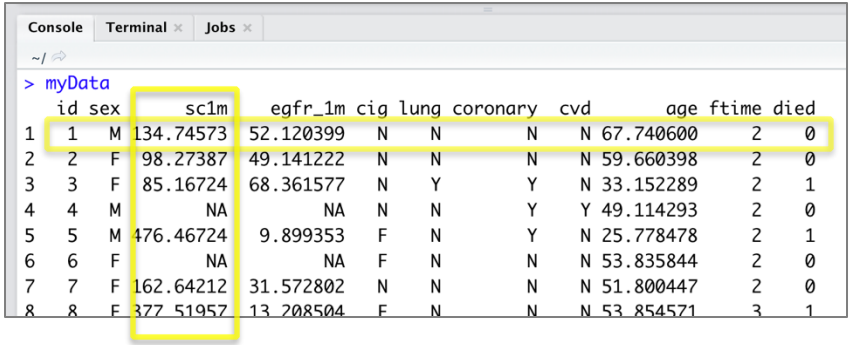
– Data frames use the same indexing idea but now with two dimensions, row number and column number

```
myData[1,3]
```

Row 1

Column 3 (3rd variable)

```
#Try in R:

myData
myData[1,3]
```

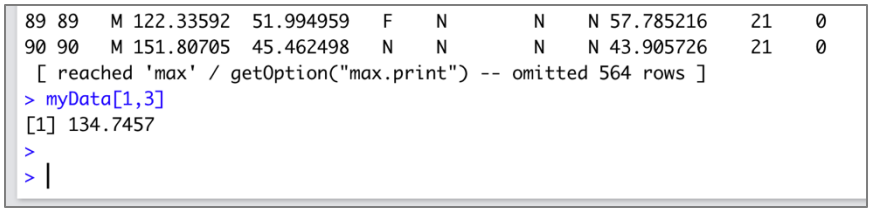


Console | Terminal | Jobs

```
> myData
   id sex     sc1m   egfr_1m cig lung coronary cvd      age ftime died
1   1   M 134.74573 52.120399   N    N        N   N 67.740600     2    0
2   2   F  98.27387 49.141222   N    N        N   N 59.660398     2    0
3   3   F  85.16724 68.361577   N    Y        Y   N 33.152289     2    1
4   4   M       NA        NA   N    N        Y   Y 49.114293     2    0
5   5   M 476.46724  9.899353   F    N        Y   N 25.778478     2    1
6   6   F       NA        NA   F    N        N   N 53.835844     2    0
7   7   F 162.64212 31.572802   N    N        N   N 51.800447     2    0
8   8   F 377.51957 13.208504   F    N        N   N 53.854571     3    1
```

Instead of the column's number, you can use the name of the variable instead

```
myData[1,"sc1m"]
```



```
89 89   M 122.33592  51.994959   F    N        N   N 57.785216    21    0
90 90   M 151.80705  45.462498   N    N        N   N 43.905726    21    0
 [ reached 'max' / getOption("max.print") -- omitted 564 rows ]
> myData[1,3]
[1] 134.7457
>
>
```

Data management

- If we want to subset the data, for example, only women, we can use

```
myData[myData$sex=="F", ]
```

All the rows where sex is F

No number indicates all the variables

```
#Try in R:

myData[myData$sex == "F", ]
myData.Females <- myData[myData$sex == "F", ]
```

Console | Terminal | Jobs

```
> myData[myData$sex == "F", ]
```

	id	sex	sc1m	egfr_1m	cig	lung	coronary	cvd	age	ftime	died
2	2	F	98.27387	49.141222	N	N	N	N	59.660398	2	0
3	3	F	85.16724	68.361577	N	Y	Y	N	33.152289	2	1
6	6	F	NA	NA	F	N	N	N	53.835844	2	0
7	7	F	162.64212	31.572802	N	N	N	N	51.800447	2	0
8	8	F	377.51957	13.208504	F	N	N	N	53.854571	3	1
10	10	F	94.73901	51.708823	N	N	N	N	60.837589	3	0
13	13	F	71.83598	94.229435	N	Y	N	N	35.450103	4	0
14	14	F	102.58364	58.958538	N	N	S	N	46.726353	4	0
16	16	F	69.90760	83.848313	F	N	N	N	65.746135	4	0
24	24	F	NA	NA	N	N	S	N	61.837629	7	0
25	25	F	71.44271	72.763561	F	N	N	N	58.572263	7	0
27	27	F	83.39249	65.666471	N	N	N	N	55.459120	7	0

Data management

– If we want to subset the data, for example, only women, we can use

```
myData[myData$sex=="F", ]
```

All the rows where sex is F

No number indicates all the variables

```
#Try in R:

myData[myData$sex == "F", ]
myData.Females <- myData[myData$sex == "F", ]
```

Console | Terminal | Jobs

```
> myData[myData$sex == "F", ]
```

	id	sex	sc1m	egfr_1m	cig	lung	coronary	cvd	age	ftime	died
2	2	F	98.27387	49.141222	N	N	N	N	59.660398	2	0
3	3	F	85.16724	68.361577	N	Y	Y	N	33.152289	2	1
6	6	F	NA	NA	F	N	N	N	53.835844	2	0
7	7	F	162.64212	31.572802	N	N	N	N	51.800447	2	0
8	8	F	377.51957	13.208504	F	N	N	N	53.854571	3	1
10	10	F	94.73901	51.708823	N	N	N	N	60.837589	3	0
13	13	F	71.83598	94.229435	N	Y	N	N	35.450103	4	0
14	14	F	102.58364	58.958538	N	N	S	N	46.726353	4	0
16	16	F	69.90760	83.848313	F	N	N	N	65.746135	4	0
24	24	F	NA	NA	N	N	S	N	61.837629	7	0
25	25	F	71.44271	72.763561	F	N	N	N	58.572263	7	0
27	27	F	83.39249	65.666471	N	N	N	N	55.459120	7	0

Data management

- If we want to subset the data, for example, only women, we can use

```
myData[myData$sex=="F", ]
```

All the rows where sex is F

No number indicates all the variables

```
#Try in R:

myData[myData$sex == "F", ]
myData.Females <- myData[myData$sex == "F", ]
```

Console | Terminal | Jobs

```
> myData[myData$sex == "F", ]
```

	id	sex	sc1m	egfr_1m	cig	lung	coronary	cvd	age	ftime	died
2	2	F	98.27387	49.141222	N	N	N	N	59.660398	2	0
3	3	F	85.16724	68.361577	N	Y	Y	N	33.152289	2	1
6	6	F	NA	NA	F	N	N	N	53.835844	2	0
7	7	F	162.64212	31.572802	N	N	N	N	51.800447	2	0
8	8	F	377.51957	13.208504	F	N	N	N	53.854571	3	1
10	10	F	94.73901	51.708823	N	N	N	N	60.837589	3	0
13	13	F	71.83598	94.229435	N	Y	N	N	35.450103	4	0
14	14	F	102.58364	58.958538	N	N	S	N	46.726353	4	0
16	16	F	69.90760	83.848313	F	N	N	N	65.746135	4	0
24	24	F	NA	NA	N	N	S	N	61.837629	7	0
25	25	F	71.44271	72.763561	F	N	N	N	58.572263	7	0
27	27	F	83.39249	65.666471	N	N	N	N	55.459120	7	0

- **Exercise:** Store the dataset with only females, in a new object

Data management

- How to compute new variables from existing ones?
- Serum creatinine seems to be measure in µmol/l.
- To convert it to mg/dl we need to multiply the original values by 0.0113

```
myData$sc.mgdl <- myData$sc1m * 0.0113
```

- This will create a new variable `sc.mgdl` in the dataset

```
#Try in R:

myData$sc.mgdl <- myData$sc1m * 0.0113
myData
```

Console | Terminal | Jobs

```
> myData$sc.mgdl <- myData$sc1m * 0.0113
> myData
   id sex      sc1m    egfr_1m cig lung coronary  cvd       age ftime died   sc.mgdl
1   1   M 134.74573  52.120399   N    N        N    N 67.740600     2    0 1.5226268
2   2   F  98.27387  49.141222   N    N        N    N 59.660398     2    0 1.1104947
3   3   F  85.16724  68.361577   N    Y        Y    N 33.152289     2    1 0.9623898
4   4   M        NA         NA   N    N        Y    Y 49.114293     2    0        NA
5   5   M 476.46724   9.899353   F    N        Y    N 25.778478     2    1 5.3840798
6   6   F        NA         NA   F    N        N    N 53.835844     2    0        NA
7   7   F 162.64212  31.572802   N    N        N    N 51.800447     2    0 1.8378560
8   8   F 377.51957  13.208504   F    N        N    N 53.854571     3    1 4.2659712
9   9   M 100.00430  80.077657   C    N        N    N 43.595586     3    0 1.1300486
10 10   F  94.73901  51.708823   N    N        N    N 60.837589     3    0 1.0705508
```

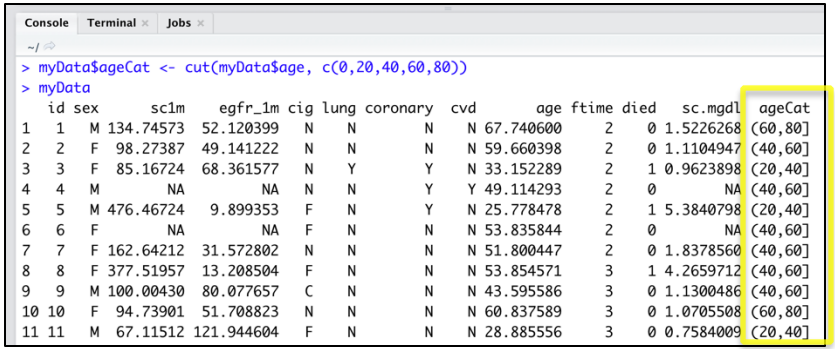
Data management

- Another common operation is to categorise continuous variables
- Let's create the following categories for age: <=20, (20-40], (40-60], >60
- We will use the function `cut()`

```
myData$ageCat <- cut(myData$age, c(0,20,40,60,80))
```

- **Exercise**: Get the frequencies for the age categories

```
#Try in R:

myData$ageCat <- cut(myData$age, c(0,20,40,60,80))
myData
```



Console | Terminal | Jobs

```
> myData$ageCat <- cut(myData$age, c(0,20,40,60,80))
> myData
   id sex      sc1m    egfr_1m cig lung coronary  cvd       age ftime died   sc.mgdl  ageCat
1   1   M 134.74573  52.120399   N    N        N    N 67.740600     2    0 1.5226268 (60,80]
2   2   F  98.27387  49.141222   N    N        N    N 59.660398     2    0 1.1104947 (40,60]
3   3   F  85.16724  68.361577   N    Y        Y    N 33.152289     2    1 0.9623898 (20,40]
4   4   M        NA         NA   N    N        Y    Y 49.114293     2    0        NA (40,60]
5   5   M 476.46724   9.899353   F    N        Y    N 25.778478     2    1 5.3840798 (20,40]
6   6   F        NA         NA   F    N        N    N 53.835844     2    0        NA (40,60]
7   7   F 162.64212  31.572802   N    N        N    N 51.800447     2    0 1.8378560 (40,60]
8   8   F 377.51957  13.208504   F    N        N    N 53.854571     3    1 4.2659712 (40,60]
9   9   M 100.00430  80.077657   C    N        N    N 43.595586     3    0 1.1300486 (40,60]
10 10   F  94.73901  51.708823   N    N        N    N 60.837589     3    0 1.0705508 (60,80]
11 11   M  67.11512 121.944604   F    N        N    N 28.885556     3    0 0.7584009 (20,40]
```

Data management

- **Exercise**: Get the frequencies for the age categories

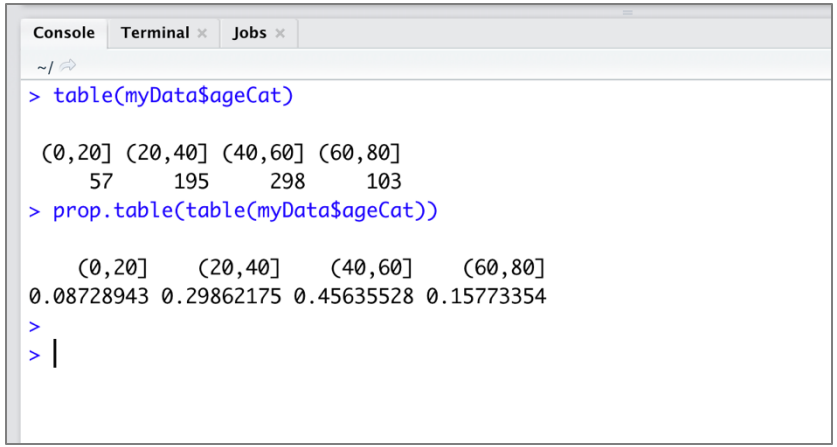
```
table(myData$ageCat)
```

- There is no direct function for the percentages of each category. We need to do:

```
prop.table(table(myData$ageCat))
```

- A bit clunky…

```
#Try in R:

table(myData$ageCat)
prop.table(table(myData$ageCat))
```

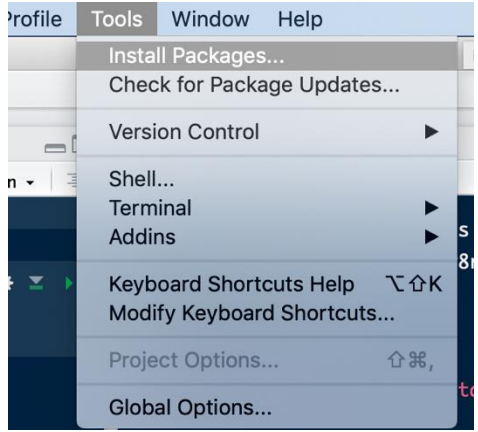


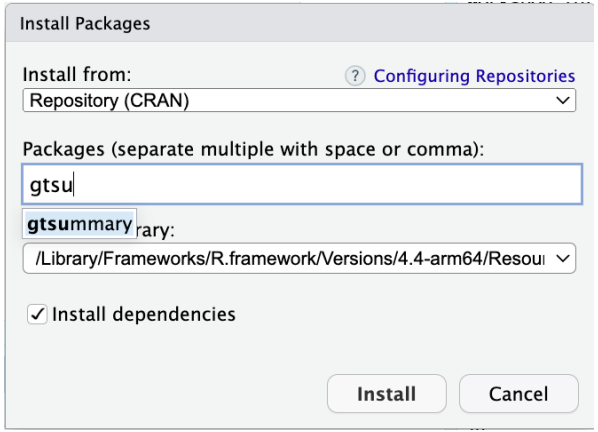
Libraries

- We are not restricted to the functions in the base version of R
- Having all community contributing with new functions is one of the main advantages of R
- Let's use the library `gtsummary`
- First we need to <u>download it</u>
  (can also use `install.packages("gtsummary")`)
- And then **load** it in R:

```
library(gtsummary)
```

- You'll need to load it every time you open R



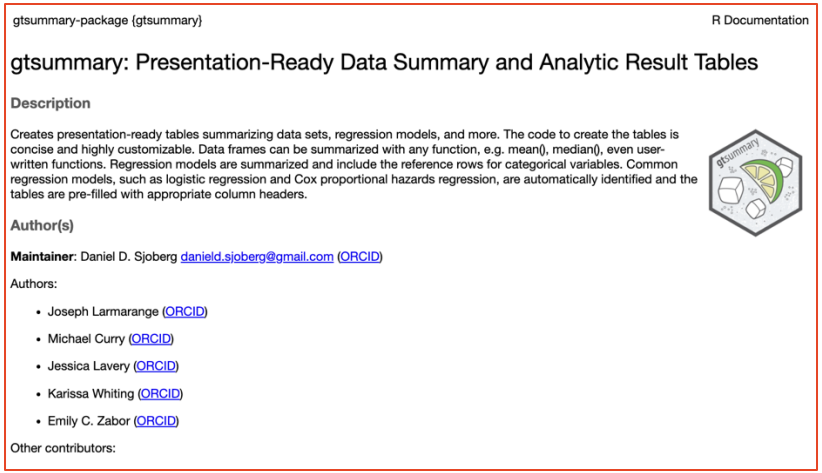


Libraries

- There are thousands of libraries available
- Each library usually have multiple functions
- The documentation is (most of the times) helpful
- So, if you type `?gtsummary`, you will have a brief description of the library and the main functions
- You can also use `?` for functions, e.g.,

```
?tbl_summary
```

```
#Try in R:
#After downloading "gtsummary"

library(gtsummary)
? gtsummary

# tbl_summary is one of the functions in the package
?tbl_summary
```



gtsummary-package {gtsummary} R Documentation

gtsummary: Presentation-Ready Data Summary and Analytic Result Tables

Description

Creates presentation-ready tables summarizing data sets, regression models, and more. The code to create the tables is concise and highly customizable. Data frames can be summarized with any function, e.g. mean(), median(), even user-written functions. Regression models are summarized and include the reference rows for categorical variables. Common regression models, such as logistic regression and Cox proportional hazards regression, are automatically identified and the tables are pre-filled with appropriate column headers.

Author(s)

Maintainer: Daniel D. Sjoberg danield.sjoberg@gmail.com (ORCID)

Authors:

- Joseph Larmarange (ORCID)
- Michael Curry (ORCID)
- Jessica Lavery (ORCID)
- Karissa Whiting (ORCID)
- Emily C. Zabor (ORCID)

Other contributors:

Libraries

```
#Try in R:

tbl_summary(data=myData)
```

- Let's used this function with the entire dataset

```
tbl_summary(data=myData)
```

When the function accepts many arguments (options), it is better to tell R what is the argument that you are passing to the function.

```
tbl_summary( data, by = NULL, label = NULL, statistic =
list(all_continuous() ~ "{median} ({p25}, {p75})", all_categorical() ~
"{n} ({p}%)"), digits = NULL, type = NULL, value = NULL, missing =
c("ifany", "no", "always"), missing_text = "Unknown", missing_stat =
"{N_miss}", sort = all_categorical(FALSE) ~ "alphanumeric", percent =
c("column", "row", "cell"), include = everything() )
```

Characteristic	N = 654[1]
id	328 (164, 491)
sex	
F	255 (39%)
M	399 (61%)
sc1m	132 (101, 175)
Unknown	6
egfr_1m	51 (37, 63)
Unknown	6
cig	
C	68 (10%)
F	190 (29%)

Libraries

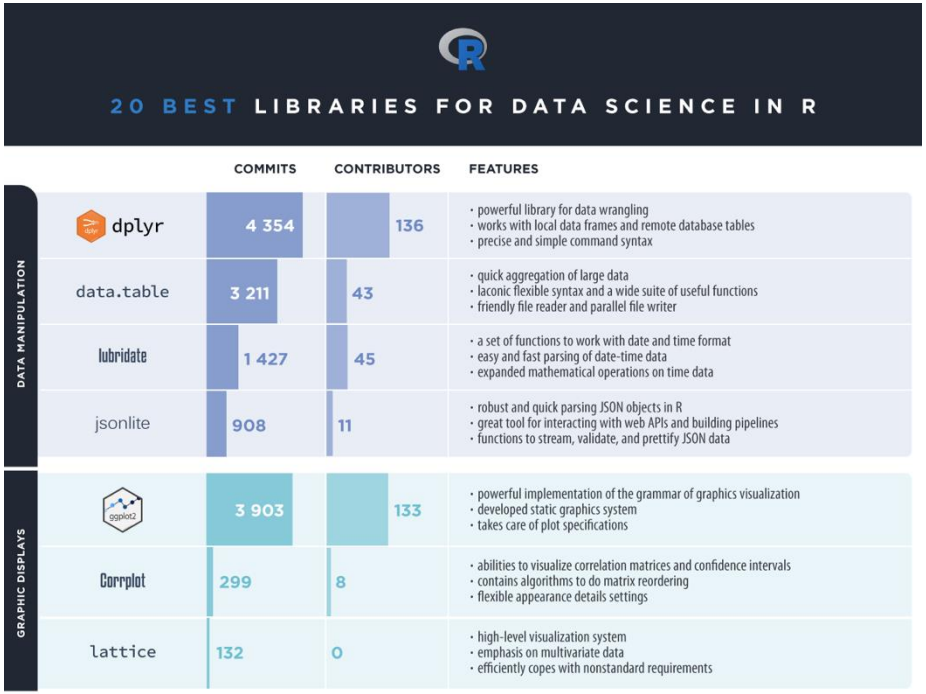
- And by survival status

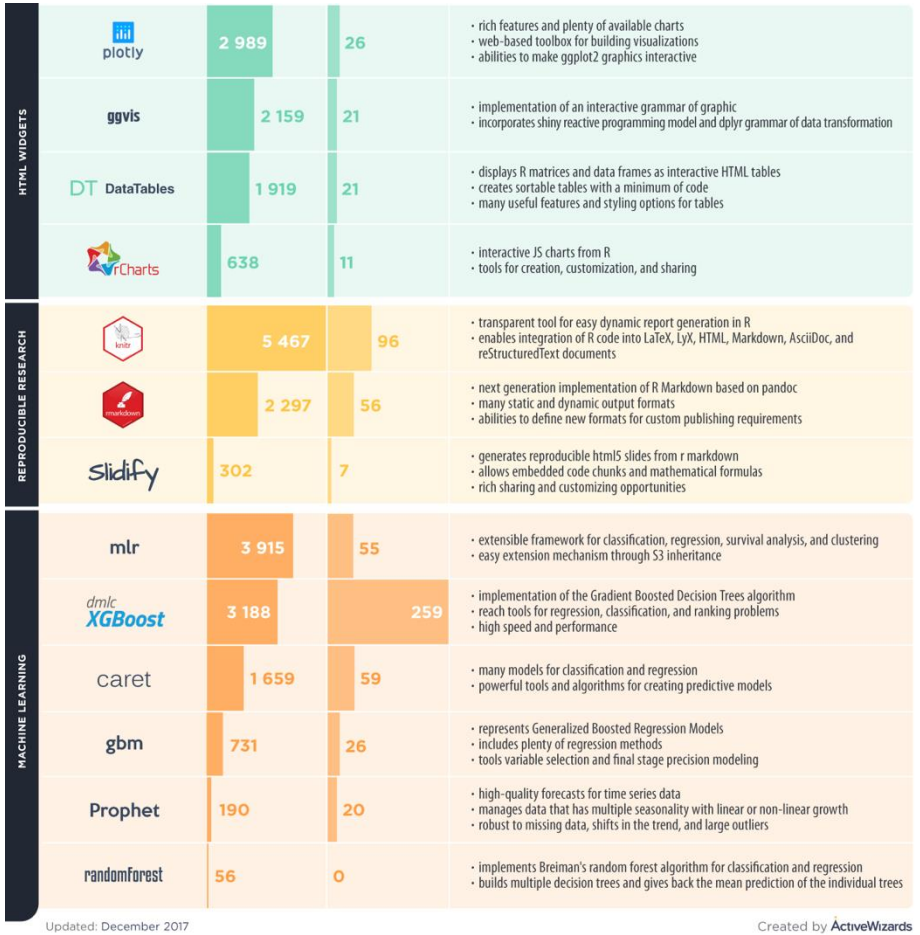
```
tbl_summary(data=myData, by='died')
```

```
#Try in R:

tbl_summary(data=myData)

#now, stratified by survival status
tbl_summary(data=myData, by='died')
```

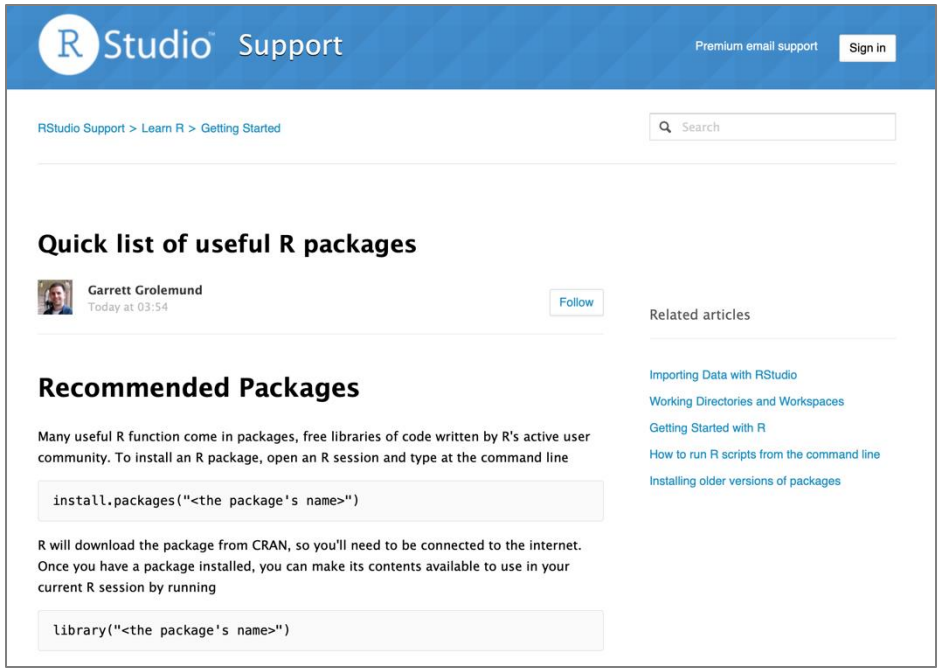
Characteristic	0 N = 516[1]	1 N = 138[1]
id	329 (173, 507)	323 (147, 461)
sex		
F	193 (37%)	62 (45%)
M	323 (63%)	76 (55%)
sc1m	129 (101, 167)	149 (103, 205)
Unknown	6	0
egfr_1m	52 (39, 64)	44 (29, 60)
Unknown	6	0
cig		
C	45 (8.7%)	23 (17%)
F	153 (30%)	37 (27%)
N	318 (62%)	78 (57%)

Libraries


20 BEST LIBRARIES FOR DATA SCIENCE IN R
COMMITS
CONTRIBUTORS
FEATURES
DATA MANIPULATION
dplyr
4 354
136
• powerful library for data wrangling
• works with local data frames and remote database tables
• precise and simple command syntax
data.table
3 211
43
• quick aggregation of large data
• laconic flexible syntax and a wide suite of useful functions
• friendly file reader and parallel file writer
lubridate
1 427
45
• a set of functions to work with date and time format
• easy and fast parsing of date-time data
• expanded mathematical operations on time data
jsonlite
908
11
• robust and quick parsing JSON objects in R
• great tool for interacting with web APIs and building pipelines
• functions to stream, validate, and prettify JSON data
GRAPHIC DISPLAYS
ggplot2
3 903
133
• powerful implementation of the grammar of graphics visualization
• developed static graphics system
• takes care of plot specifications
Corrplot
299
8
• abilities to visualize correlation matrices and confidence intervals
• contains algorithms to do matrix reordering
• flexible appearance details settings
lattice
132
0
• high-level visualization system
• emphasis on multivariate data
• efficiently copes with nonstandard requirements


HTML WIDGETS
plotly
2 989
26
• rich features and plenty of available charts
• web-based toolbox for building visualizations
• abilities to make ggplot2 graphics interactive
ggvis
2 159
21
• implementation of an interactive grammar of graphic
• incorporates shiny reactive programming model and dplyr grammar of data transformation
DT DataTables
1 919
21
• displays R matrices and data frames as interactive HTML tables
• creates sortable tables with a minimum of code
• many useful features and styling options for tables
rCharts
638
11
• interactive JS charts from R
• tools for creation, customization, and sharing
REPRODUCIBLE RESEARCH
knitr
5 467
96
• transparent tool for easy dynamic report generation in R
• enables integration of R code into LaTeX, LyX, HTML, Markdown, AsciiDoc, and reStructuredText documents
rmarkdown
2 297
56
• next generation implementation of R Markdown based on pandoc
• many static and dynamic output formats
• abilities to define new formats for custom publishing requirements
Slidify
302
7
• generates reproducible html5 slides from r markdown
• allows embedded code chunks and mathematical formulas
• rich sharing and customizing opportunities
MACHINE LEARNING
mlr
3 915
55
• extensible framework for classification, regression, survival analysis, and clustering
• easy extension mechanism through S3 inheritance
dmlc XGBoost
3 188
259
• implementation of the Gradient Boosted Decision Trees algorithm
• reach tools for regression, classification, and ranking problems
• high speed and performance
caret
1 659
59
• many models for classification and regression
• powerful tools and algorithms for creating predictive models
gbm
731
26
• represents Generalized Boosted Regression Models
• includes plenty of regression methods
• tools variable selection and final stage precision modeling
Prophet
190
20
• high-quality forecasts for time series data
• manages data that has multiple seasonality with linear or non-linear growth
• robust to missing data, shifts in the trend, and large outliers
randomForest
56
0
• implements Breiman's random forest algorithm for classification and regression
• builds multiple decision trees and gives back the mean prediction of the individual trees
Updated: December 2017
Created by ActiveWizards

Libraries

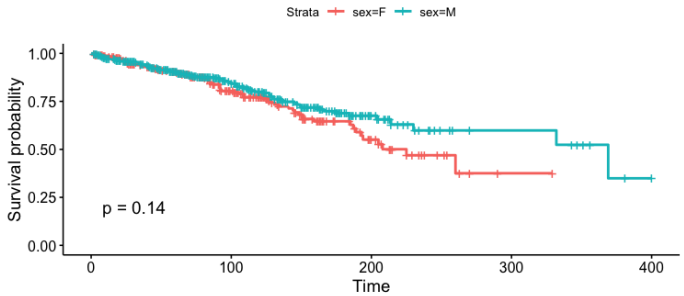


https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages

Example of a statistical analysis

- Let's look a differences on survival time after transplant between sexes
- We could start by looking at the Kaplan-Meier curve
- Perform a log-rank test
- And then move to a multivariable Cox regression

```
#####Survival analysis#####
library(survival)
library(survminer)

#Kaplan meier and log-rank test comparing sexes
KMsex <- survfit(Surv(ftime,died) ~ sex,
                 data = myData)
ggsurvplot(KMsex, pval=TRUE)

#Cox model
myCoxModel <- coxph(Surv(ftime,died) ~ sex + age + egfr_1m,
                    data = myData)
summary(myCoxModel)
ggforest(myCoxModel, data = myData)

#checking the PH assumption
test.ph <- cox.zph(myCoxModel)
test.ph
plot(test.ph)
```



Links for learning (more) about R

Edx.org (online courses)

- Statistics and R (Harvard School of Public Health) (https://www.edx.org/course/statistics-and-r-3)

datacamp.com (courses and tutorials for data science)

- Survival Analysis in R For Beginners (https://www.datacamp.com/community/tutorials/survival-analysis-R)
- Logistic Regression in R Tutorial (https://www.datacamp.com/community/tutorials/logistic-regression-R)

Books

- An Introduction to Biostatistics Using R - Waveland Press (http://waveland.com/Glover-Mitchell/r-guide.pdf)

News

- https://www.r-bloggers.com
- https://awesome-r.com
- https://www.statmethods.net (Quick-R)