

INTRODUCTIE IN R, QUARTO EN RSTUDIO

Gerko Vink 

Methodology & Statistics @ Utrecht University

Laurence Frank 

Methodology & Statistics @ Utrecht University

2 Jun 2025



TOPICS OF THIS LECTURE

1. Introduction to R and RStudio

- Working with packages
- Getting help in R

2. Reproducible data analysis with Quarto

3. Organise your work with R Projects

4. R data objects

- vectors
- matrices
- data frames
- lists
- factors



INTRODUCTION TO R AND RSTUDIO



WHAT IS R

- R is a language and environment for statistical computing and for graphics
- Based on the object-oriented language S (1975)
- 100% free software
- Managed by the R Foundation for Statistical Computing, Vienna, Austria.
- Community-driven:
 - More than 10.000 packages developed by community
 - New packages are constantly being developed
 - New features are constantly being added to existing packages



FUN FACT ABOUT R:

Every version of R that is released is named after a topic in a Peanuts comic. The R version 4.3.3 (2024-02-29) is called "Angel Food Cake".

R version

```
R version 4.3.3 (2024-02-29) -- "Angel Food Cake"  
Copyright (C) 2024 The R Foundation for Statistical Computing  
Platform: aarch64-apple-darwin20 (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.
```

Charlie Brown cartoon



WHAT IS RSTUDIO?

- RStudio is an Integrated Development Environment (IDE)
- RStudio has all functionality in one place and makes working with R much easier.
- Use RStudio to:
 - Edit scripts, Run scripts
 - Manage your code with highlighting
 - Navigate files, organize projects
 - Utilize version control (e.g. Github)
 - View static and interactive graphics
 - Create different file types (RMarkdown, Shiny apps)
 - Work with different languages (Python, JavaScript, C++, etc.)



THE 4 PANES IN RSTUDIO

The image shows the RStudio interface with four main panes:

- Source pane:** edit and run scripts, view data sets
- Environment pane:** Overview of objects (data sets, parameters) you have created
- Console pane:** Where the commands are run, display of errors, warnings
- File management, Plots, Packages and Help**

Source pane:

```

13 ~ ``{r}
14 y <- 8
15 x <- 10
16 x * y
17 ```

[1] 80

18
19
20 ## R Markdown
21
22 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
23
24 When you click the **Knit** button a document will be generated that includes both content as
well as the output of any embedded R code chunks within the document. You can embed an R code
chunk like this:
25
26 ``{r cars}
35:15 Chunk 4: pressure
  
```

Environment pane:

Values	x	10
y	8	

Console pane:

```

R 4.3.3 · ~/Sync/DocsUU/Onderwijs/RSummerschool/RSummerschool2024/TM_RSummer2024/
  
```

```

y <- 8
x <- 10
x * y
[1] 80
summary(cars)
   speed      dist
Min.   : 4.0   Min.   : 2.00
1st Qu.:12.0   1st Qu.: 26.00
Median :15.0   Median : 36.00
Mean   :15.4   Mean   : 42.98
3rd Qu.:19.0   3rd Qu.: 56.00
Max.   :25.0   Max.   :120.00
  
```

File management, Plots, Packages and Help

A scatter plot showing the relationship between temperature (x-axis, ranging from 0 to 350) and pressure (y-axis, ranging from 0 to 800). The data points show a positive correlation, with pressure increasing as temperature increases.

WORKING WITH R PACKAGES



R PACKAGES: BASE INSTALLATION

- When you start RStudio and R only the base packages are activated: the basic installation with basic functionality.
- There are almost 20.000 packages that have been developed by R users all over the world. See the [Comprehensive R Archive Network \(CRAN\)](#)
- Not efficient to have all these packages installed every time you use R. Install only the packages you want to use.



SEE WHICH PACKAGES ARE ACTIVE

- Use `sessionInfo()` to see which packages are active.
- This is how the basic installation looks like:

```
Matrix products: default
BLAS:    /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBLAS.dylib
LAPACK:  /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

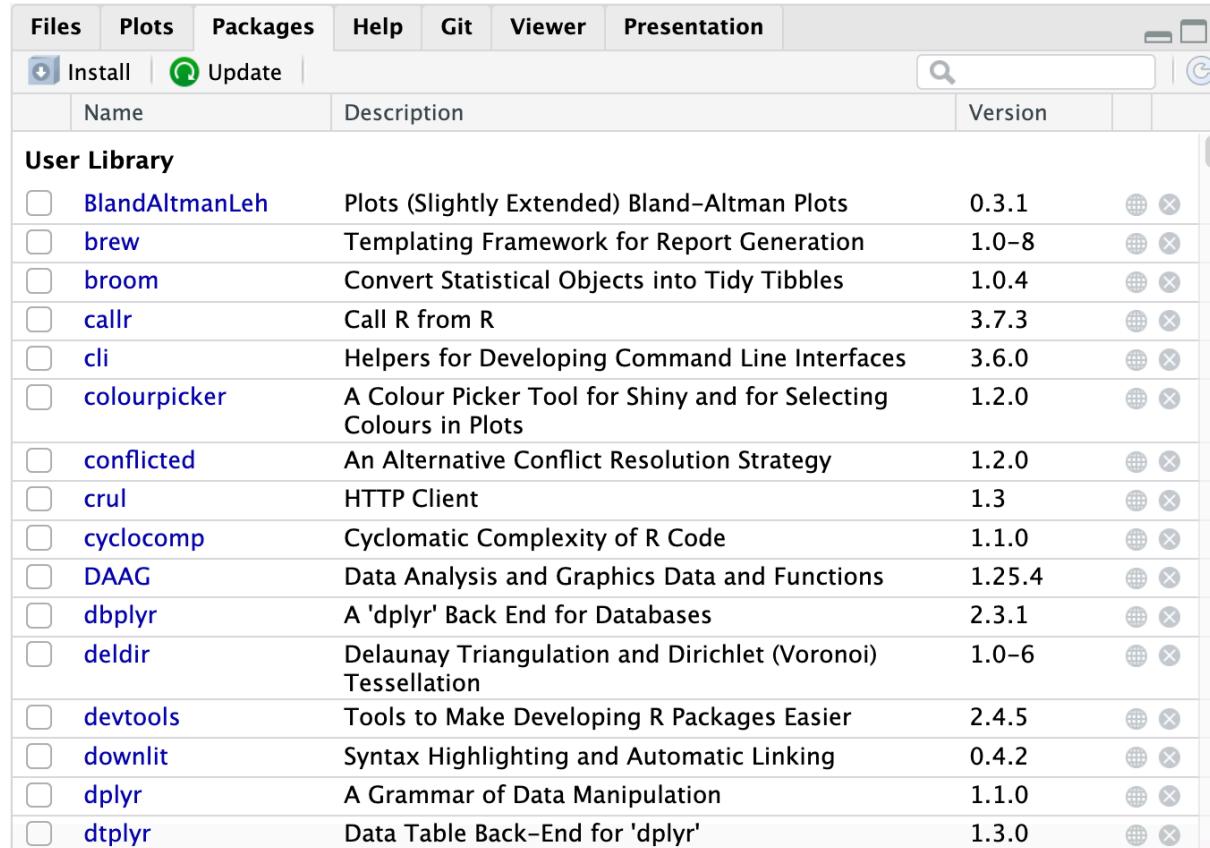
attached base packages:
[1] stats      graphics   grDevices utils      datasets   methods    base

loaded via a namespace (and not attached):
[1] compiler_4.2.2  fastmap_1.1.0   cli_3.6.0      htmltools_0.5.4
[5] tools_4.2.2     rstudioapi_0.14  yaml_2.3.5    rmarkdown_2.14
[9] highr_0.9       knitr_1.39     xfun_0.37     digest_0.6.29
[13] rlang_1.0.6     evaluate_0.15
```



OVERVIEW OF INSTALLED PACKAGES

An overview of the packages you have installed, see the tab “Packages” in the output pane:



Files	Plots	Packages	Help	Git	Viewer	Presentation		
		Install		Update		Search		
		Name	Description			Version		
User Library								
<input type="checkbox"/>	BlandAltmanLeh	Plots (Slightly Extended)	Bland–Altman Plots	0.3.1				
<input type="checkbox"/>	brew	Templating Framework for Report Generation		1.0-8				
<input type="checkbox"/>	broom	Convert Statistical Objects into Tidy Tibbles		1.0.4				
<input type="checkbox"/>	callr	Call R from R		3.7.3				
<input type="checkbox"/>	cli	Helpers for Developing Command Line Interfaces		3.6.0				
<input type="checkbox"/>	colourpicker	A Colour Picker Tool for Shiny and for Selecting Colours in Plots		1.2.0				
<input type="checkbox"/>	conflicted	An Alternative Conflict Resolution Strategy		1.2.0				
<input type="checkbox"/>	curl	HTTP Client		1.3				
<input type="checkbox"/>	cyclocomp	Cyclomatic Complexity of R Code		1.1.0				
<input type="checkbox"/>	DAAG	Data Analysis and Graphics Data and Functions		1.25.4				
<input type="checkbox"/>	dbplyr	A 'dplyr' Back End for Databases		2.3.1				
<input type="checkbox"/>	deldir	Delaunay Triangulation and Dirichlet (Voronoi) Tessellation		1.0-6				
<input type="checkbox"/>	devtools	Tools to Make Developing R Packages Easier		2.4.5				
<input type="checkbox"/>	downlit	Syntax Highlighting and Automatic Linking		0.4.2				
<input type="checkbox"/>	dplyr	A Grammar of Data Manipulation		1.1.0				
<input type="checkbox"/>	dtplyr	Data Table Back-End for 'dplyr'		1.3.0				

HOW TO WORK WITH PACKAGES

Packages are to R what apps are on your mobile phone.

- When you want to use a package for the first time, you have to **install** the package.
- Each time you want to use the package, you have to **load** (activate) it.

OPENING AND CLOSING PACKAGES

To **load** a package use the following code (similar to opening an app on your phone):

```
1 library(ggplot2)
```

To close a package use (similar to closing an app on your phone):

```
1 detach(ggplot2)
```

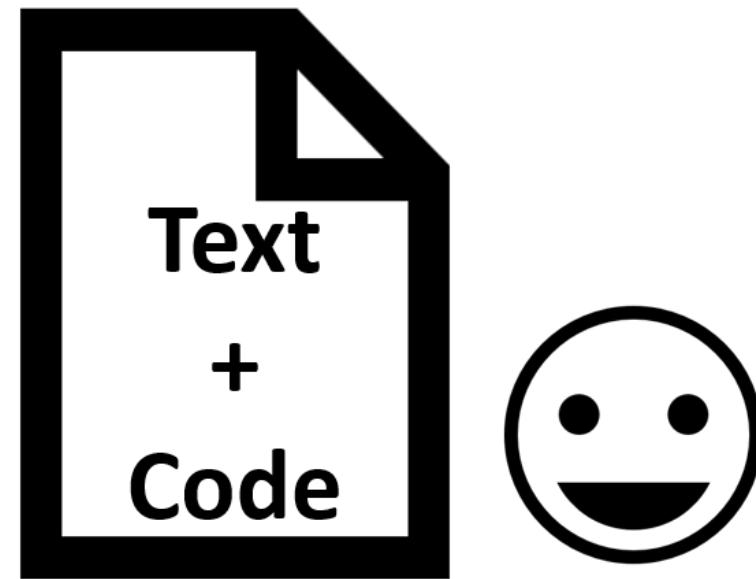
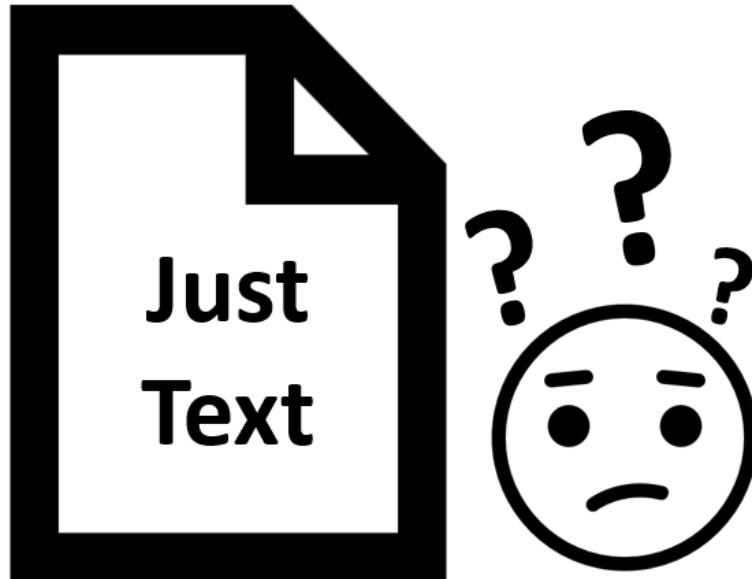


REPRODUCIBLE DATA ANALYSIS WITH Quarto



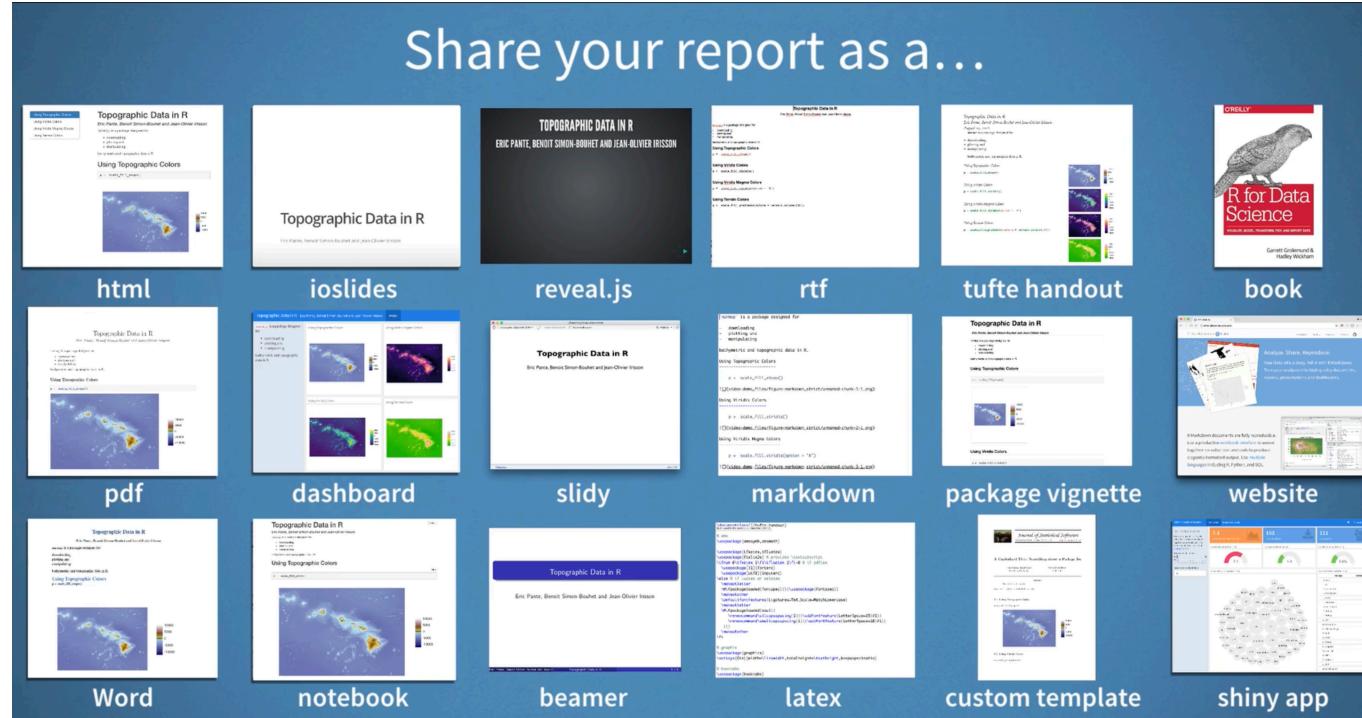
WHY WORK WITH Quarto?

The need to *combine code and text* and to document all the steps to make **reproducible** (scientific) reports of data analyses.



WHY WORK WITH Quarto?

It is **efficient**. Generate and update reports in all kinds of formats:



Source: What is R Markdown? Video RStudio

DEMO RSTUDIO AND Quarto



WRITING TEXT IN Quarto



See the [R Markdown Cheat Sheet](#) for a complete list of options.

Syntax	Becomes
Plain text	Plain text
End a line with two spaces to start a new paragraph.	End a line with two spaces to start a new paragraph.
italics and _italics_	<i>italics</i> and <i>italics</i>
bold and __bold__	bold and bold
superscript^2^	superscript ²
~~strikethrough~~	strikethrough
[link](www.rstudio.com)	link
# Header 1	<h1>Header 1</h1>
## Header 2	<h2>Header 2</h2>
### Header 3	<h3>Header 3</h3>
#### Header 4	<h4>Header 4</h4>
##### Header 5	<h5>Header 5</h5>
###### Header 6	<h6>Header 6</h6>



WRITING TEXT IN Quarto



Syntax

```
* unordered list
* item 2
  + sub-item 1
  + sub-item 2
```

```
1. ordered list
2. item 2
  + sub-item 1
  + sub-item 2
```

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

Becomes

- unordered list
- item 2
 - sub-item 1
 - sub-item 2

1. ordered list
2. item 2
 - sub-item 1
 - sub-item 2

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4



WRITING CODE IN Quarto



Code chunks start with `{r }` (for R code). You can give code chunks names (here cars).

```
```{r cars}
summary(cars)
```
```



This is how the result looks like in the rendered html document. Display of both R code and results:

```
summary(cars)
```

```
##      speed          dist
## Min.   : 4.0   Min.   : 2.00
## 1st Qu.:12.0   1st Qu.: 26.00
## Median :15.0   Median : 36.00
## Mean   :15.4   Mean   : 42.98
## 3rd Qu.:19.0   3rd Qu.: 56.00
## Max.   :25.0   Max.   :120.00
```

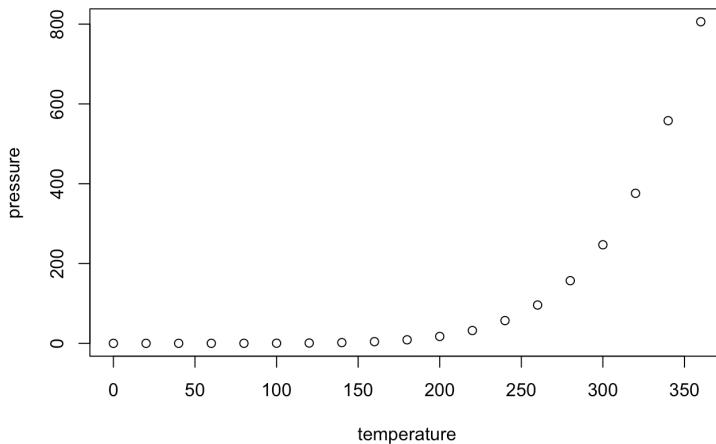


CODE CHUNK OPTIONS



You can choose to hide the R code with `echo=FALSE` in the chunk header:

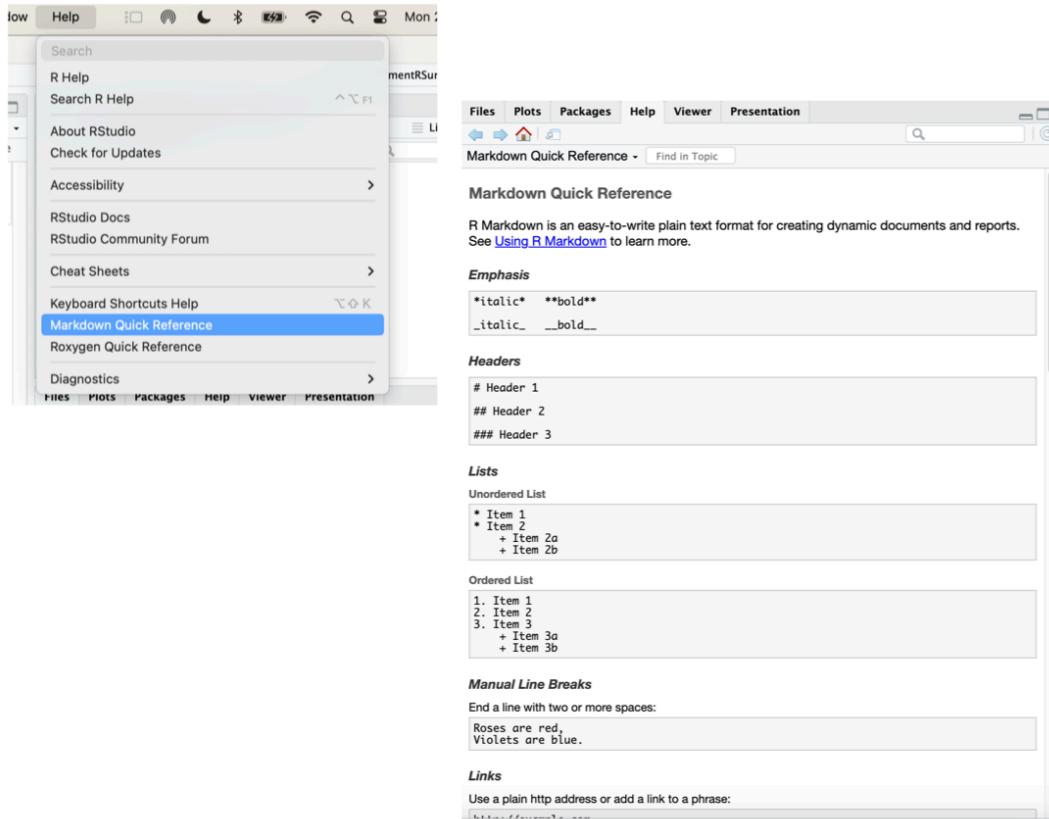
```
```{r pressure, echo=FALSE}
plot(pressure)
````
```



See the [Quarto reference page](#) for a complete list of chunk options.

GETTING HELP WITH Quarto

Quarto is the evolution of R Markdown. In [RStudio](#) you can find the Markdown Reference:



R STUDIO PROJECTS



USE RSTUDIO PROJECTS

Every time you start a new (data analysis) project, make it a habit to create a new **RStudio Project**.

Because you want your project to work:

- not only now, but also in a few years;
- when the folder and file paths have changed;
- when collaborators want to run your code on their computer.

RStudio Projects create a convention that guarantees that the project can be moved around on your computer or onto other computers and will still “just work”. It creates relative paths (no more broken paths!).



EXAMPLE: DATA ANALYSIS RSTUDIO PROJECT



All data, scripts, and output should be stored within the project directory.

Every time you want to work on this project: open the project by clicking the `.Rproj` file.

| Example-project-structure | | | |
|---------------------------------|-----------|---------------|--|
| Name | Kind | Date Modified | |
| ▼ data | Folder | Today, 21:20 | |
| ► processed | Folder | Today, 21:20 | |
| ► raw | Folder | Today, 21:20 | |
| Example-project-structure.Rproj | R Project | Today, 20:11 | |
| ▼ output | Folder | Today, 21:28 | |
| ► figures | Folder | Today, 21:28 | |
| ► tables | Folder | Today, 21:28 | |
| ► scripts | Folder | Today, 21:20 | |
| ► temp | Folder | Today, 21:20 | |

R DATA OBJECTS



USING R AS A CALCULATOR

The simplest thing you could do with R is do arithmetic:

```
1 100 + 10  
2 ## [1] 110
```

```
1 9 / 3  
2 ## [1] 3
```



USING R AS A CALCULATOR

Here are the common signs to use in arithmetic:

| arithmetic | sign |
|----------------|---------------------|
| Addition | + |
| Subtraction | - |
| Multiplication | * |
| Division | / |
| Exponents. | \wedge or ** |



ASSIGNMENT OPERATOR

If you have used R before, you know the `<-` assignment operator.

Here `x` is assigned the value 8

```
1 x <- 8
```

If you run this code:

- a new value will be saved in your work space (uses computer memory!)
- In the environment pane, the tab “Environment”, you will see `x` under “Values” followed by 8

The screenshot shows the RStudio interface with the "Environment" tab selected in the top navigation bar. Below the tabs, there are several icons: a folder (File), a document (New File), a gear (Import Dataset), a disk (Save), a circular progress bar (178 MiB), and a paintbrush (Edit). To the right of these are buttons for "List" and "C". Under the "Global Environment" section, there is a search bar with a magnifying glass icon. The main pane is titled "Values" and contains a table with one row. The row has two columns: the first column is labeled "x" and the second column contains the value "8".



PRINTING VALUES

```
1 x <- 8
```

Assigning does not print the value 8.

If you want to print to value 8 you can do:

```
1 x  
2  
3 # or:  
4  
5 print(x)
```



R IS AN OBJECT-ORIENTED PROGRAMMING LANGUAGE

When you assign values with the assignment operator `<-` you create an R **object**.

Objects can contain data, functions or even other objects.

The most commonly used objects are:

- vector
- matrix
- data frame
- list
- formulas and models



VECTORS



VECTOR

A vector is a concatenation of values (data). The simplest object in R is a vector with one element:

```
1 x <- 8
```



VECTOR GENERATING FUNCTIONS



The function `c(...)` collects elements in a vector

```
1 v <- c(1, 2, 3, 4, 5)
```

- `seq(from, to)` or `:` generate a sequence of integers

```
1 seq(from = 1, to = 5)
```

```
[1] 1 2 3 4 5
```

```
1 1:5
```

```
[1] 1 2 3 4 5
```

- `rep(..., times)` repeats ... a number of `times`

```
1 rep(1:5, times = 2)
2 ## [1] 1 2 3 4 5 1 2 3 4 5
```



CLASSES



Vectors (and other R objects) can contain be of data types (classes)

Numeric

```
1 v <- c(1, 2, 3, 4, 5)
2 class(v)
3 ## [1] "numeric"
```

Character

```
1 char <- c("cat", "dog")
2 typeof(char)
3 ## [1] "character"
```

A vector can only be of *one* class. If you combine different classes, R will convert them to the most general class.

CLASSES



Logical data can take only one of two values: TRUE or FALSE.

```
1 v <- c(1,2,3,4,5)
2
3 # Identify elements > 3 in numeric vector v:
4 logical <- v > 3
5
6 print(logical)
7 ## [1] FALSE FALSE FALSE  TRUE  TRUE
```



VECTOR CLASSES

- all elements of a vector (are forced to) have the same class

```
1 class(num <- c(1, 2))
2 ## [1] "numeric"
3
4 class(char <- c("cat", "dog"))
5 ## [1] "character"
6
7 c(num, char)
8 ## [1] "1"    "2"    "cat"  "dog"
9
10 class(c(num, char))
11 ## [1] "character"
```



MATRICES



MATRIX GENERATING FUNCTIONS



- `matrix(data, nrow, ncol)` generates a matrix
 - all elements (are forced to) have the same class

```

1 M <- matrix(data = 1:6, nrow = 2, ncol = 3)
2 M
3 ##      [,1] [,2] [,3]
4 ## [1,]    1    3    5
5 ## [2,]    2    4    6
6 class(M)
7 ## [1] "matrix" "array"

```

`cbind(...)` collects vectors in a matrix as columns:

```

1 cbind(a = 1:2, b = c("cat", "dog"))
2 ##      a     b
3 ## [1,] "1"  "cat"
4 ## [2,] "2"  "dog"

```

`rbind(...)` collects vectors as rows:

```

1 rbind(a = 1:2, b = c("cat", "dog"))
2 ##      [,1] [,2]
3 ## a "1"  "2"
4 ## b "cat" "dog"

```

DATA FRAMES



DATA FRAME GENERATING FUNCTIONS



- `data.frame(...)` collects vectors as variables in a data frame
 - variables can have different classes

```
1 df <- data.frame(x = 1:2, y = c("cat", "dog"), z = c(T, F))
2 df
3 ##   x     y     z
4 ## 1 1 cat  TRUE
5 ## 2 2 dog FALSE
6 class(df)
7 ## [1] "data.frame"
8
9 sapply(df, class)
10 ##            x             y             z
11 ##      "integer"    "character"    "logical"
```



LISTS



LIST GENERATING FUNCTION



- `list(...)` creates a list
 - can contain objects of any dimension and class
 - used for collecting output from R function (e.g. linear regression)

```
1 L <- list(v = c(1, 2), matrix = M, df = df, list(1:10))
2 class(L)
3 ## [1] "list"
4 sapply(L, class)
5 ## $v
6 ## [1] "numeric"
7 ##
8 ## $matrix
9 ## [1] "matrix" "array"
10 ##
11 ## $df
12 ## [1] "data.frame"
13 ##
14 ## [[4]]
15 ## [1] "list"
```

FACTORS



FACTORS



- `factor(...)` makes / changes vector into factor
 - factors have levels
 - used for categorical variables in analyses (e.g. linear model)

```
1 animals <- rep(c("cat", "dog"), 4)
2 summary(animals)
3 ##      Length   Class    Mode
4 ##          8 character character
5
6 factor(animals)
7 ## [1] cat dog cat dog cat dog cat dog
8 ## Levels: cat dog
9 summary(factor(animals))
10 ## cat dog
11 ## 4    4
```



ASSIGN NAMES WITH `names()`



Use `names()` to assign names to elements in R objects.

For example to the elements of a list:

```
1  names(L) <-c("Vector", "Matrix", "Dataframe", "List")
2  L

$Vector
[1] 1 2

$Matrix
 [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

>Dataframe
  x     y      z
1 1 cat  TRUE
2 2 dog FALSE

>List
>List[[1]]
[1] 1 2 3 4 5 6 7 8 9 10
```

USE OF DATA OBJECTS

When to use the R data objects?

| Object | Use | Why |
|------------------|---------------------------|--|
| data frame | statistical analysis | can store variables of any class |
| model formula | statistical models, plots | concise and readable, flexible,
consistent across functions, packages |
| lists | storage of output | can store any object of any class |
| vectors/matrices | programming | can do fast calculations |



NAMING CONVENTIONS, STYLE GUIDE



FILE AND OBJECT NAMING

- File names should be **meaningful**.
- Avoid spaces in file names and use one of the **naming conventions**:
 1. **snake_case**: words are separated by underscores (_), and all letters are typically in lowercase. Examples: data_analysis.RData, my_data.csv.
 2. **camelCase**: each word within a compound word is capitalized, except for the first word, and no spaces or underscores are used to separate the words. Examples: calculateMean, summaryStatistics.
 3. **PascalCase**: the first letter of each word in a compound word is capitalized, and no spaces or underscores are used to separate the words. Examples: DataAnalysis, DescriptiveStatistics.





SPACING AND INDENTATION

- When indenting your code, use 2 spaces. RStudio does this for you!
- Never use tabs or a mix of tabs and spaces.
- Place spaces around all operators (=, +, -, <-). Use `x <- 5` not `x<-5`

Exception: spaces around = are optional when passing parameters in a function call.

```
1 lm(age ~ bmi, data=boys)
```

or

```
1 lm(age ~ bmi, data = boys)
```



COMMAS AND PUNCTUATION



- Do not put spaces before commas, but always put a space after commas.
 - `c(1, 2, 3)`
- For function arguments, follow the same rule.
 - `sum(a = 1, b = 2)`

Bad examples:

```
1 # No spaces around debug
2 if ( debug )
3
4 # Needs a space after the comma
5 x[1,]
```



COMMENTS



- Use # for single-line comments and place them above the code they reference.
- Keep comments concise and relevant.

```
1 # Read the msleep.csv data and save the data as msleep
2 msleep <- readr::read_csv("msleep.csv")
```