INTRODUCTION TO DATA ANALYSIS IN R

Quantitative Workshop III
a SHANTU project

Erik de Jong

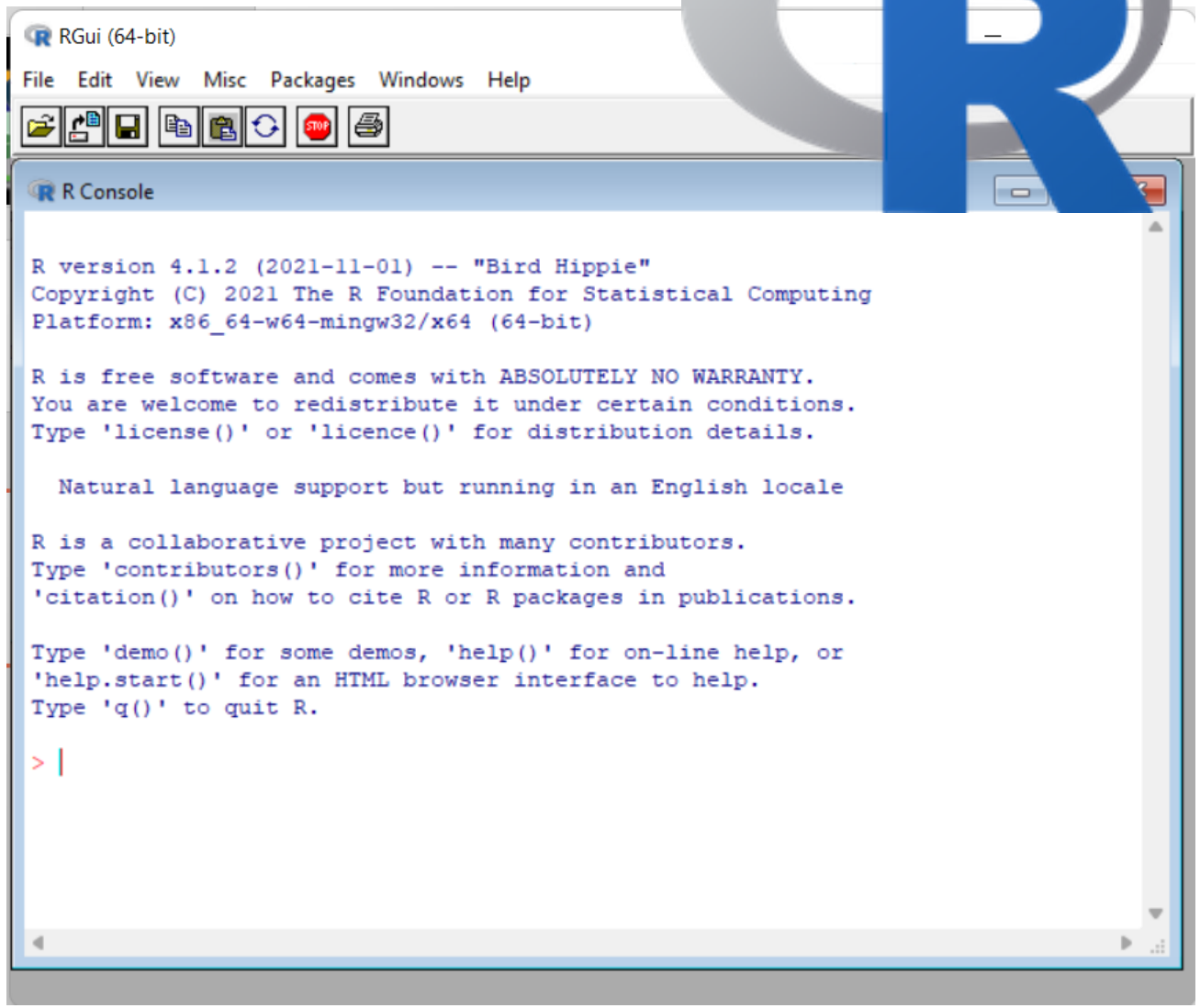
TODAY'S AGENDA

- R and R studio
- Packages
- Functions
- Tips
- Hands-on practice

THE R PROJECT

- R for statistical computing and graphics
- Command line interface

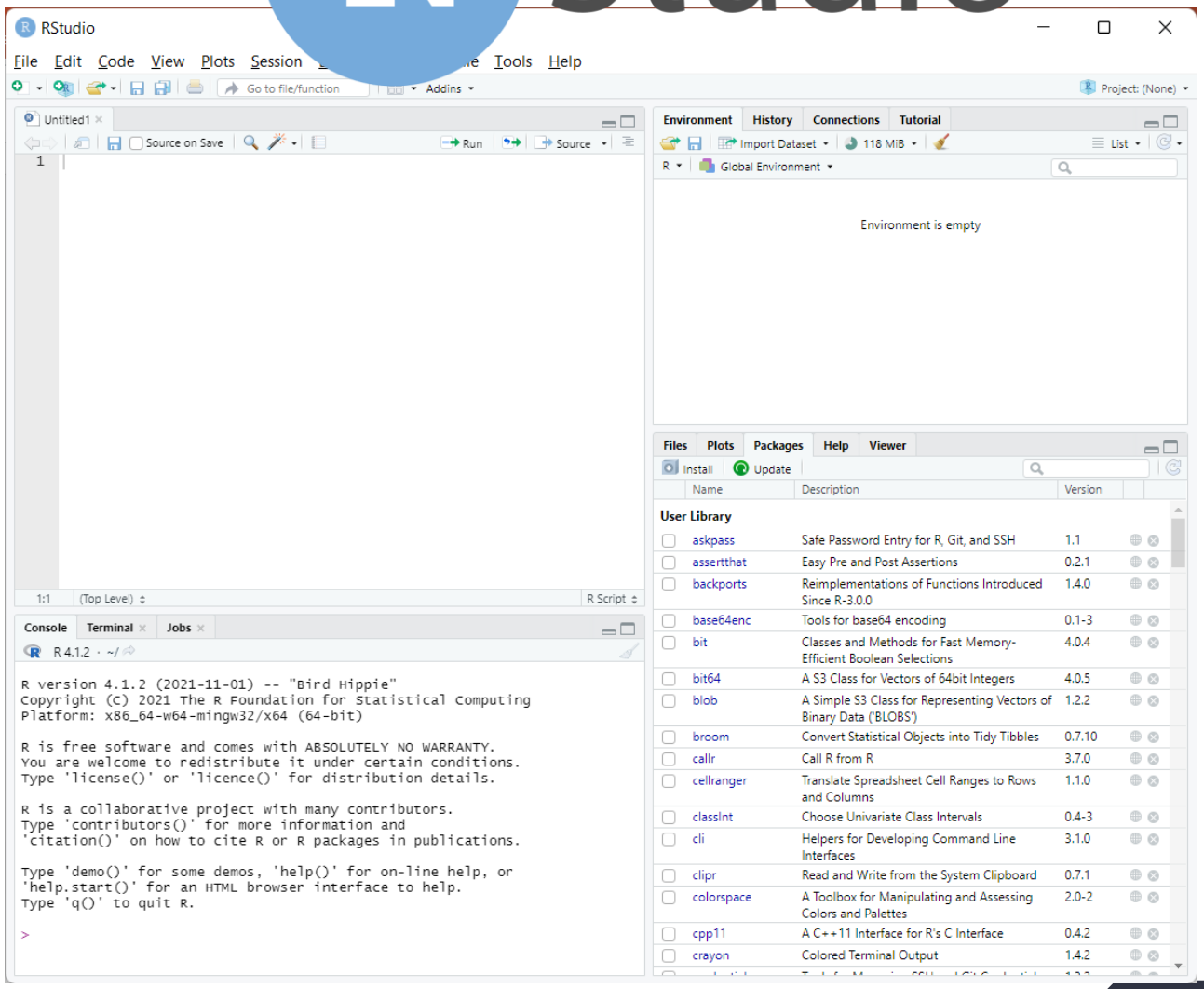
- Download R at https://cran.r-project.org/ and choose system and “base R”



R STUDIO

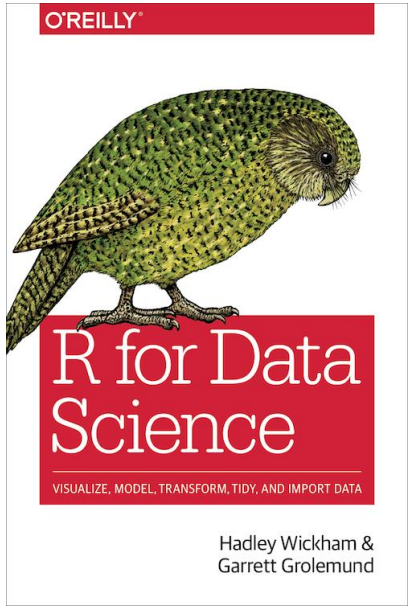
- Depends on the base version of R
- “all in one” environment
- Runs R in the R studio environment, besides:
  - Code editor & data viewer
  - Environment overview and log
  - Plots, packages, files and help documentation
- Download R studio desktop at:

  https://www.rstudio.com/products/rstudio/download/

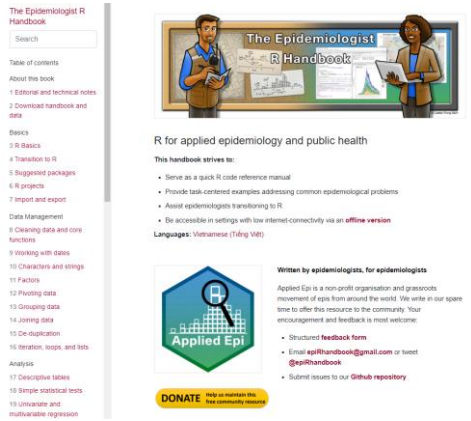


WHY R AND R STUDIO?

For data analysis and visualization?

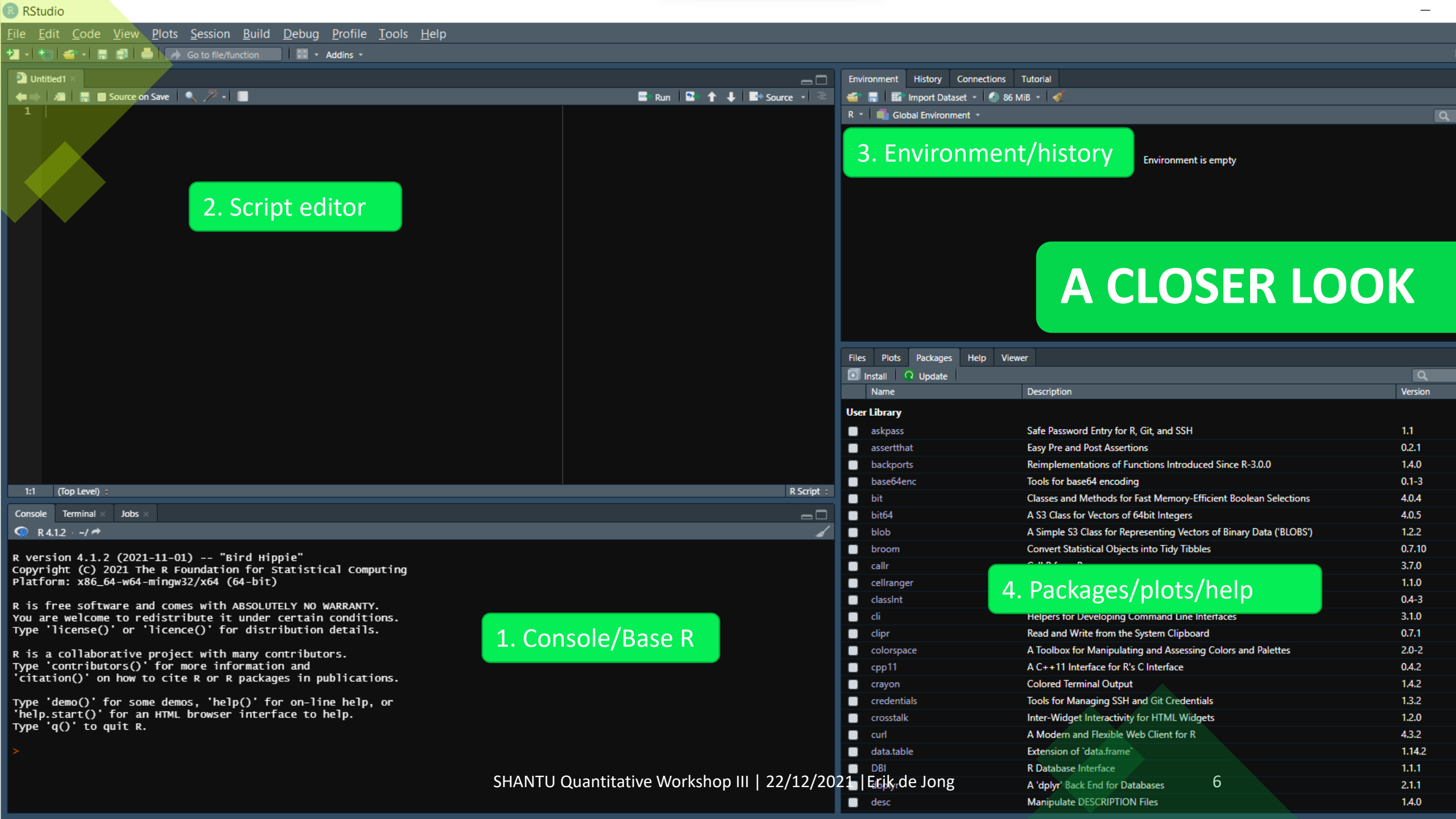


https://r4ds.had.co.nz/



https://www.epirhandbook.com/en/index.html

- Open source → Free for personal use
- Available for many platforms/distributions (and also older)
- Handles large data
- Every step is documented → analysis can be applied to other data
- Packages designed for specific tasks
- Interactive graphs and plots
- Plotting of special information (GIS)
- Lots of resources
  - Tutorials
  - Free eBooks
  - Stack overflow with specific questions
  - Package documentation
  - Package vignettes


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Untitled1
Source on Save
Run
Source
1
2. Script editor
1:1
(Top Level)
R Script
Console
Terminal
Jobs
R 4.1.2 · ~/
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.
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.
>
1. Console/Base R
Environment
History
Connections
Tutorial
Import Dataset
86 MiB
R
Global Environment
3. Environment/history
Environment is empty
A CLOSER LOOK
Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
askpass Safe Password Entry for R, Git, and SSH 1.1
assertthat Easy Pre and Post Assertions 0.2.1
backports Reimplementations of Functions Introduced Since R-3.0.0 1.4.0
base64enc Tools for base64 encoding 0.1-3
bit Classes and Methods for Fast Memory-Efficient Boolean Selections 4.0.4
bit64 A S3 Class for Vectors of 64bit Integers 4.0.5
blob A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS') 1.2.2
broom Convert Statistical Objects into Tidy Tibbles 0.7.10
callr 3.7.0
cellranger 1.1.0
classInt 0.4-3
cli Helpers for Developing Command Line Interfaces 3.1.0
clipr Read and Write from the System Clipboard 0.7.1
colorspace A Toolbox for Manipulating and Assessing Colors and Palettes 2.0-2
cpp11 A C++11 Interface for R's C Interface 0.4.2
crayon Colored Terminal Output 1.4.2
credentials Tools for Managing SSH and Git Credentials 1.3.2
crosstalk Inter-Widget Interactivity for HTML Widgets 1.2.0
curl A Modern and Flexible Web Client for R 4.3.2
data.table Extension of `data.frame` 1.14.2
DBI R Database Interface 1.1.1
dbplyr A 'dplyr' Back End for Databases 2.1.1
desc Manipulate DESCRIPTION Files 1.4.0
4. Packages/plots/help

GET STARTED IN R

What do you need to know beforehand

SHANTU Quantitative Workshop III | 22/12/2021
|Erik de Jong

PACKAGES IN R

- Many people develop (sometimes very specific) packages for R
- Packages are sets of functions that R can carry out specific tasks
- Examples include:
  - Readxl, to Import excel data into R
  - Tidyverse, a combination of 8 essential packages
- Every time you start R you need to load packages, but install is only once

purrr

purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. Once you master the basic concepts, purrr allows you to replace many for loops with code that is easier to write and more expressive. Go to docs...

tibble

tibble is a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not. Tibbles are data.frames that are lazy and surly: they do less and complain more forcing you to confront problems earlier, typically leading to cleaner, more expressive code. Go to docs...

stringr

stringr provides a cohesive set of functions designed to make working with strings as easy as possible. It is built on top of stringi, which uses the ICU C library to provide fast, correct implementations of common string manipulations. Go to docs...

forcats

forcats provides a suite of useful tools that solve common problems with factors. R uses factors to handle categorical variables, variables that have a fixed and known set of possible values. Go to docs...

INSTALL PACKAGES IN R

Install packages ----------

install.packages("readxl") #installs the package read xl

install.packages("tidyverse") #installs the tidyverse packages

install is only needed once, loading of packages (library) is needed every time you start R

Library functions ----------

library() #gives you a list of installed

library(readxl) #loads the package read xl for usage

library(tidayverse) #loads the tidyverse packages for usage

search() #show the packages loaded, available for direct use

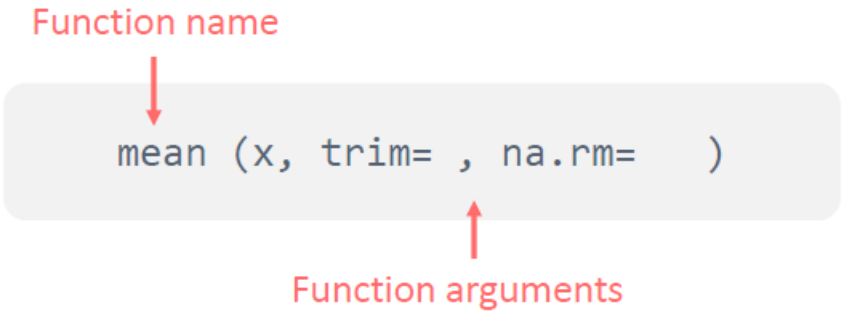
```
# Install packages ----------
install.packages("readxl") #installs the package read xl
install.packages("tidyverse") #installs the tidyverse packages


# Library functions ----------
library() #gives you a list of installed
library(readxl) #loads the package read xl for usage
library(tidayverse) #loads the tidyverse packages for usage
search() #show the packages loaded, available for direct use
```

FUNCTIONS IN R

And what they do

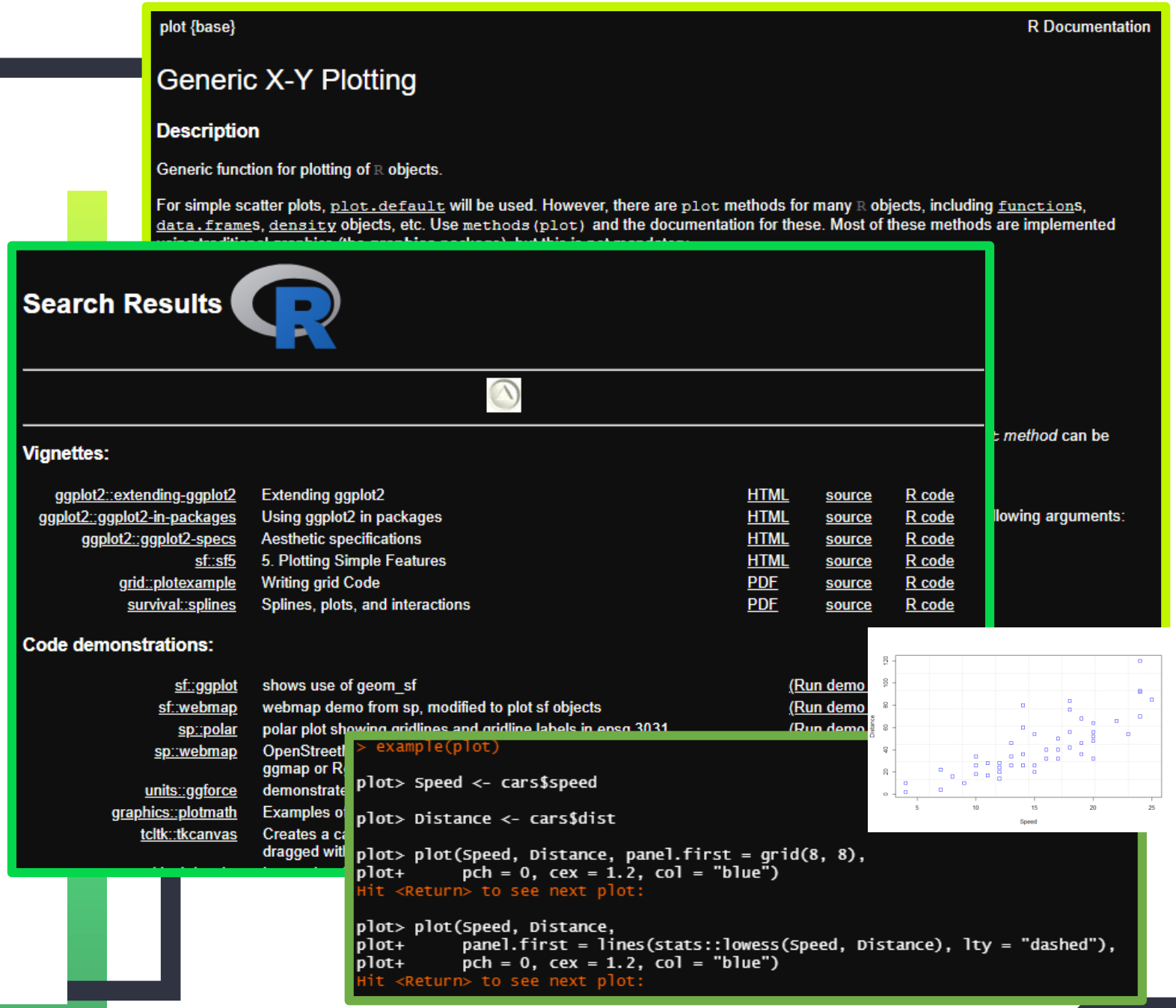
- Functions are designed to do a specific task, for example:
  - mean() gives the average of a string of numbers
  - plot() will make a plot (if certain parameters are given)
- Arguments can be passed on to the function between the ()
- Some arguments have default values



HELP FUNCTIONS

Build into R

- Many packages and functions, impossible to remember
- R has build in help functions:
  - ? Followed by function name
  - ?? Followed by function name
  - Example() with the function name in brackets



DATATYPES

Commonly used in R

- Numeric (for numbers)
- Integer (natural number)
- Logic (true of false)
- Character (text)

```
> class(4L)
[1] "integer"
> class(a)
[1] "numeric"
> class(s)
Error: object 's' not found
> class("sss")
[1] "character"
> 4
[1] 4
> class(4.4)
[1] "numeric"
>
```

OPERATORS

R Arithmetic Operators

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus
%/%	Integer Division

R Relational Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

OPERATORS

R Logical Operators

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

R Assignment Operators

Operator	Description
<-, <<-, =	Leftwards assignment
->, ->>	Rightwards assignment

DATA STRUCTURES

V e c t o r

- Store simple data
- One dimensional
- Can be anything from numeric, integer, logical, or character
- c() is a function to create a factor
- x <- c(45,5,66,5.5,NA) assigns a numeric values to the vector and names the vector a

```
# numeric vector c(1,2,3,4.2,-5)
[1]  1.0  2.0  3.0  4.2 -5.0
# vector of consecutive numbers
c(1:5)
[1] 1 2 3 4 5
# character vector
c("apple","grape","lemon","orange")
[1] "apple"  "grape"  "lemon"
"orange"
# logical
c(TRUE,FALSE,FALSE,TRUE,TRUE)
[1]  TRUE FALSE FALSETRUE TRUE
```

DATA STRUCTURES

Data frames

- Combination of multiple vectors
- Two dimensional
- Fundamental data structure to store datasets
- Variables are as columns
- Observations are as rows
- Each column contains the same data type
- Each row can contain different data types

```
# Create a dataframe
patient_id<-c(1:5)
gender<- c("M","F","F","F","M")
dob<- c("1990-01-04","1987-10-10","1970-11-12"
,"2001-05-05","1977-05-11")
malaria<- c(T,F,F,T,F)
df<- data.frame(patient_id,gender,dob,malaria)
df
```

```
  patient_id gender        dob malaria
1          1      M 1990-01-04    TRUE
2          2      F 1987-10-10   FALSE
3          3      F 1970-11-12   FALSE
4          4      F 2001-05-05    TRUE
5          5      M 1977-05-11   FALSE
```

DATA STRUCTURES

Data frame value selection

dataframe[row, column]

dataframe[1,]

dataframe[,variable_name]

dataframe$variable_name

```
df
  patient_id gender        dob malaria
1          1      M 1990-01-04    TRUE
2          2      F 1987-10-10   FALSE
3          3      F 1970-11-12   FALSE
4          4      F 2001-05-05    TRUE
5          5      M 1977-05-11   FALSE

df[,4]
 [1]  TRUE FALSE FALSE  TRUE FALSE
df[,"malaria"]
 [1]  TRUE FALSE FALSE  TRUE FALSE
df$malaria
 [1]  TRUE FALSE FALSE  TRUE FALSE
```

DATA STRUCTURES

O t h e r

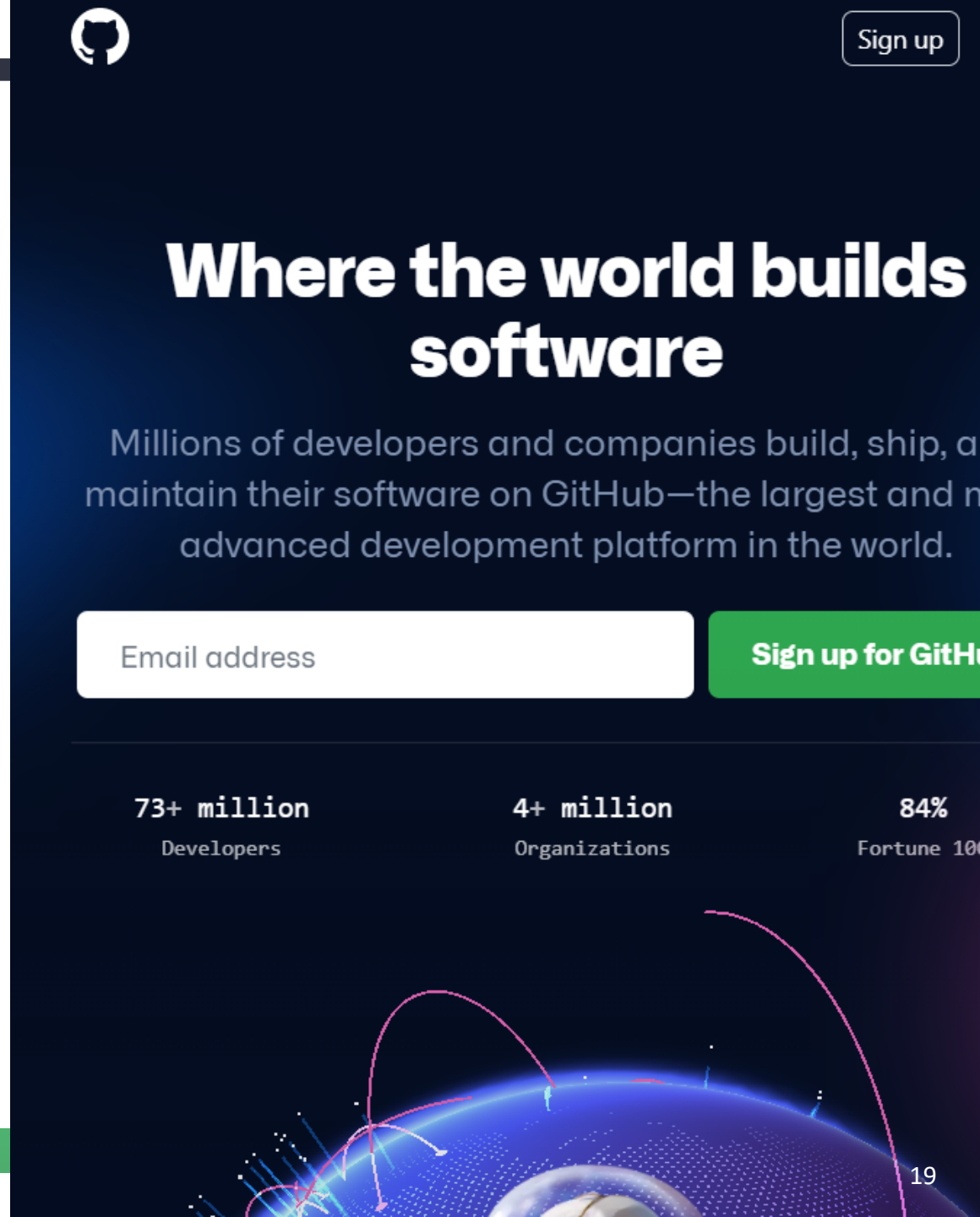
- Matrix
- List
- Factor

- Some packages work with their own datastructures

SOME TIPS

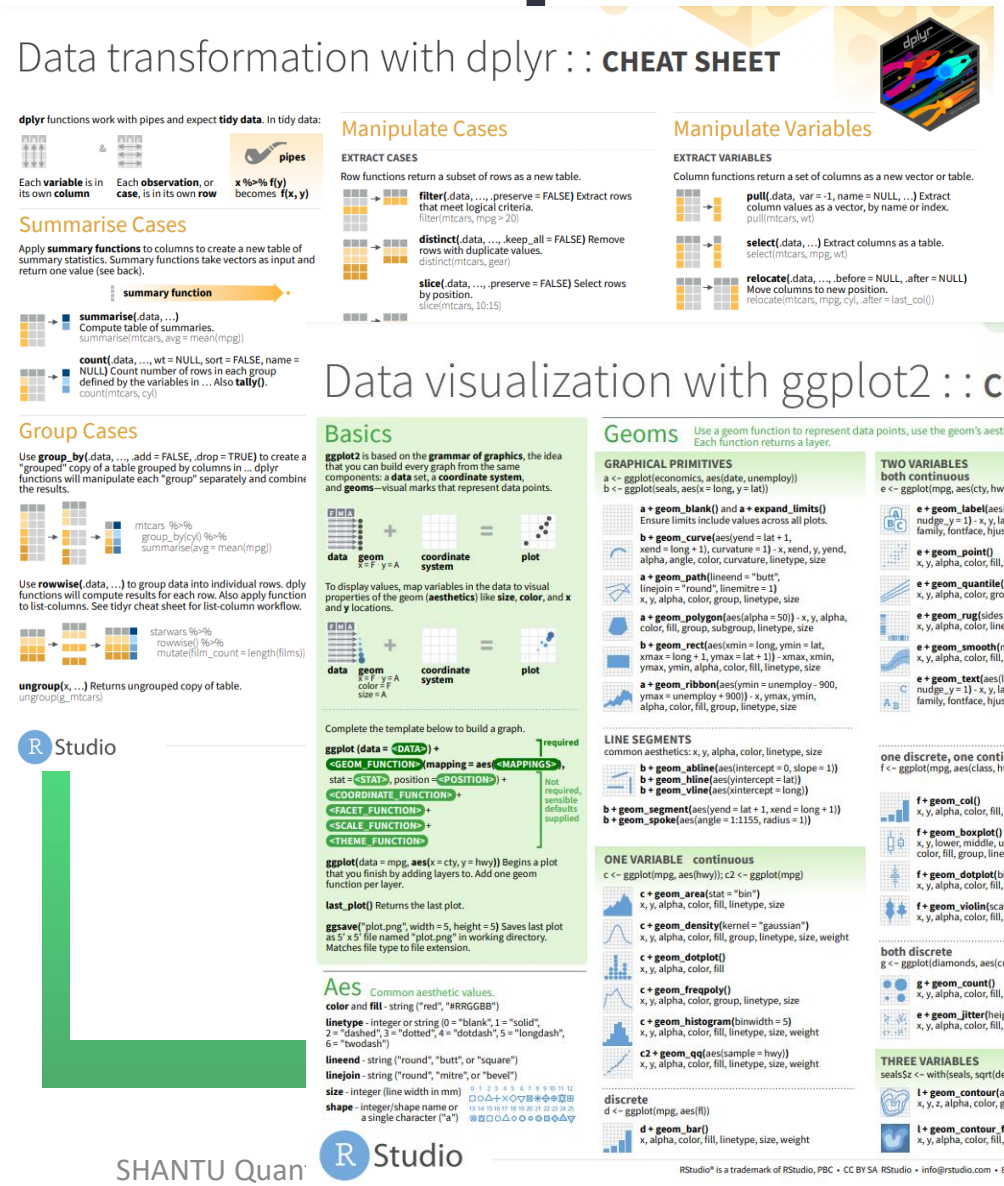
From my personal experience

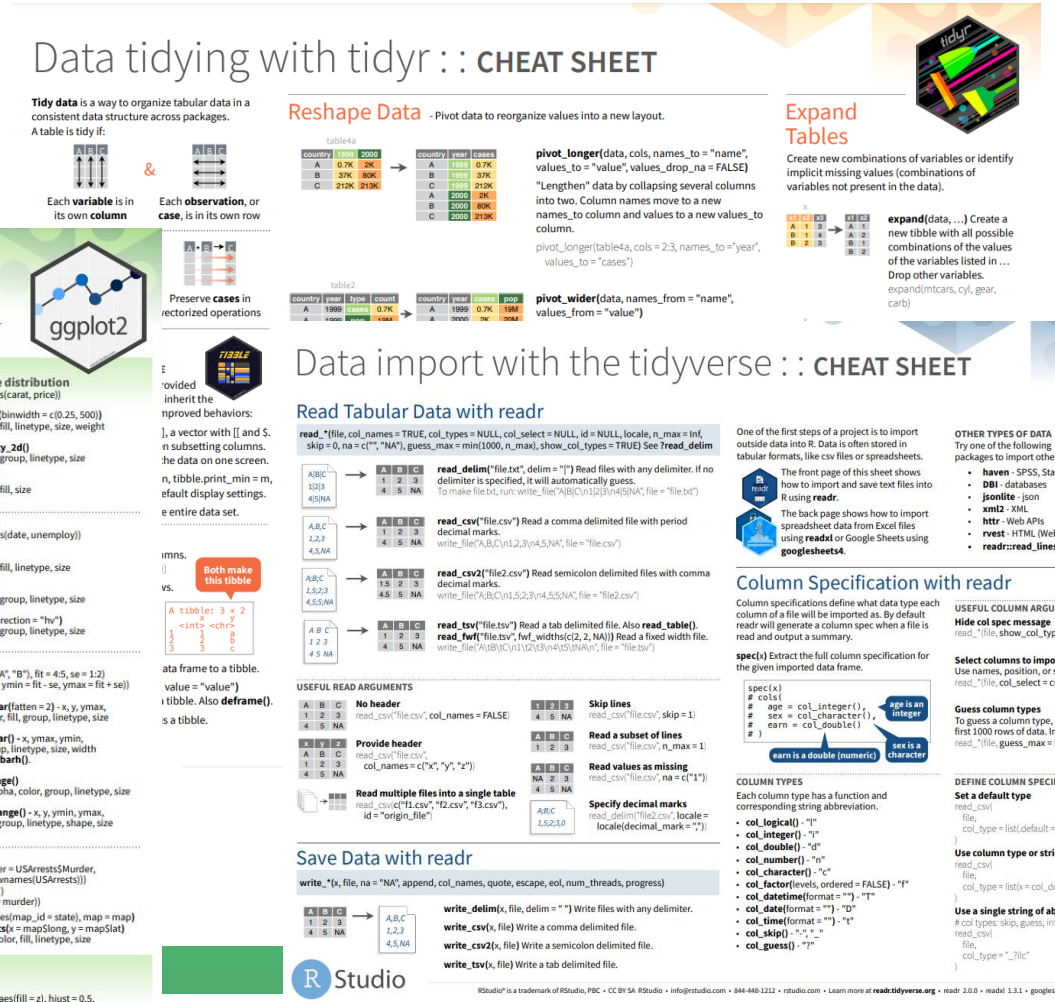
- Always use R scrips and save them
- The larger your data gets, data cleaning in R is fast
- If your scripts get more complex → make a connection to git(hub) for version control
- Make your coding readable and searchable:
  - For yourself and others
  - Easier to search later
  - Use # to comment on a line of coding (or use ctrl+shift+c)
  - Use section in your r code (ctrl+shift+r)



OTHER RESOURCES

https://www.rstudio.com/resources/cheatsheets/





HANDS-ON PRACTICE

In R