R: A Hitchhikers Guide to Reproducible Research

- We built this software on base R code

Brendan Palmer,
Clinical Research Facility - Cork &
School of Public Health
@B_A_Palmer

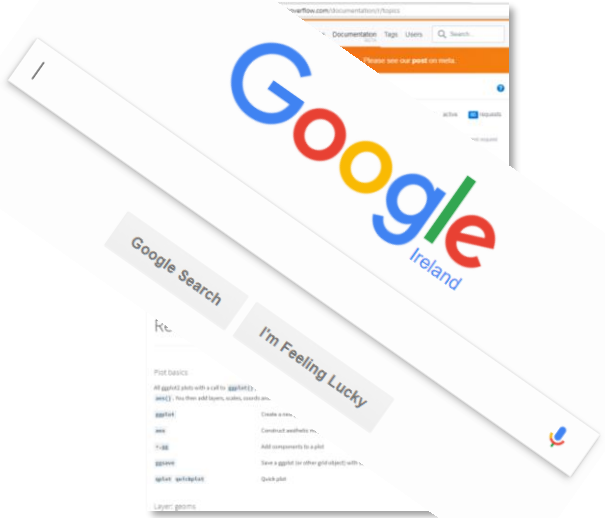
CRF-C
HRB Clinical Research Facility Cork

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh
School of Public Health

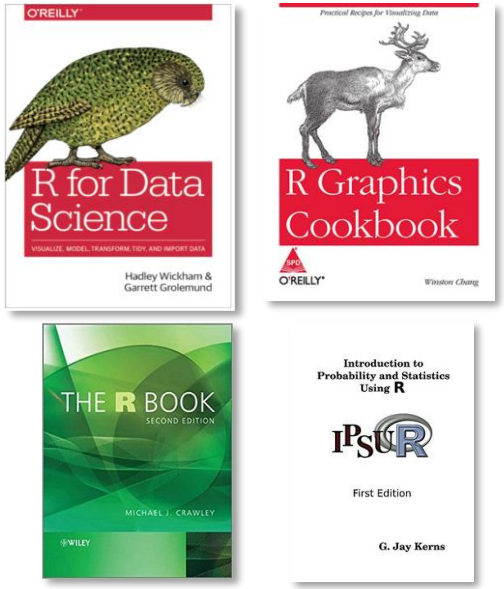
R is for Resources

Vignettes

Webpages



eBooks



Cheatsheets

Twitter

Mara Averick
@dataandme

tidyverse @rstudio, hoop head, gnashgab, blatherskite, lesser ½ of @batpigandme

Massachusetts

One R Tip a Day
@RLangTip

One tip per day M-F on the R programming language #rstats. Brought to you by the R community team at Microsoft.

Hadley Wickham
@hadleywickham

R, data, visualisation.

Houston, TX

hadley.nz

David Robinson
@drob

Data Scientist at @StackOverflow, #rstats fan/evangelist

New York, NY

varianceexplained.org

Jenny Bryan
@JennyBryan

Software engineer @rstudio, humane #rstats, adjunct prof @UBC where I created @STAT545, part of @ropensci

Vancouver, BC

jennybryan.org

Darren L Dahly
@statsepi Follows you

Principal Statistician, Epidemiologist, Sr Lecturer | @HRBIreland Clinical Research Facility @CRF_CORK | Cork #Rstats Users Group meetup.com/Cork-Ireland-R...

Cork, Ireland

darrendahly.github.io

Data Scientists IRL
@DataSci_Ireland Follows you

Promoting the Data Science professions in Ireland.

Ireland

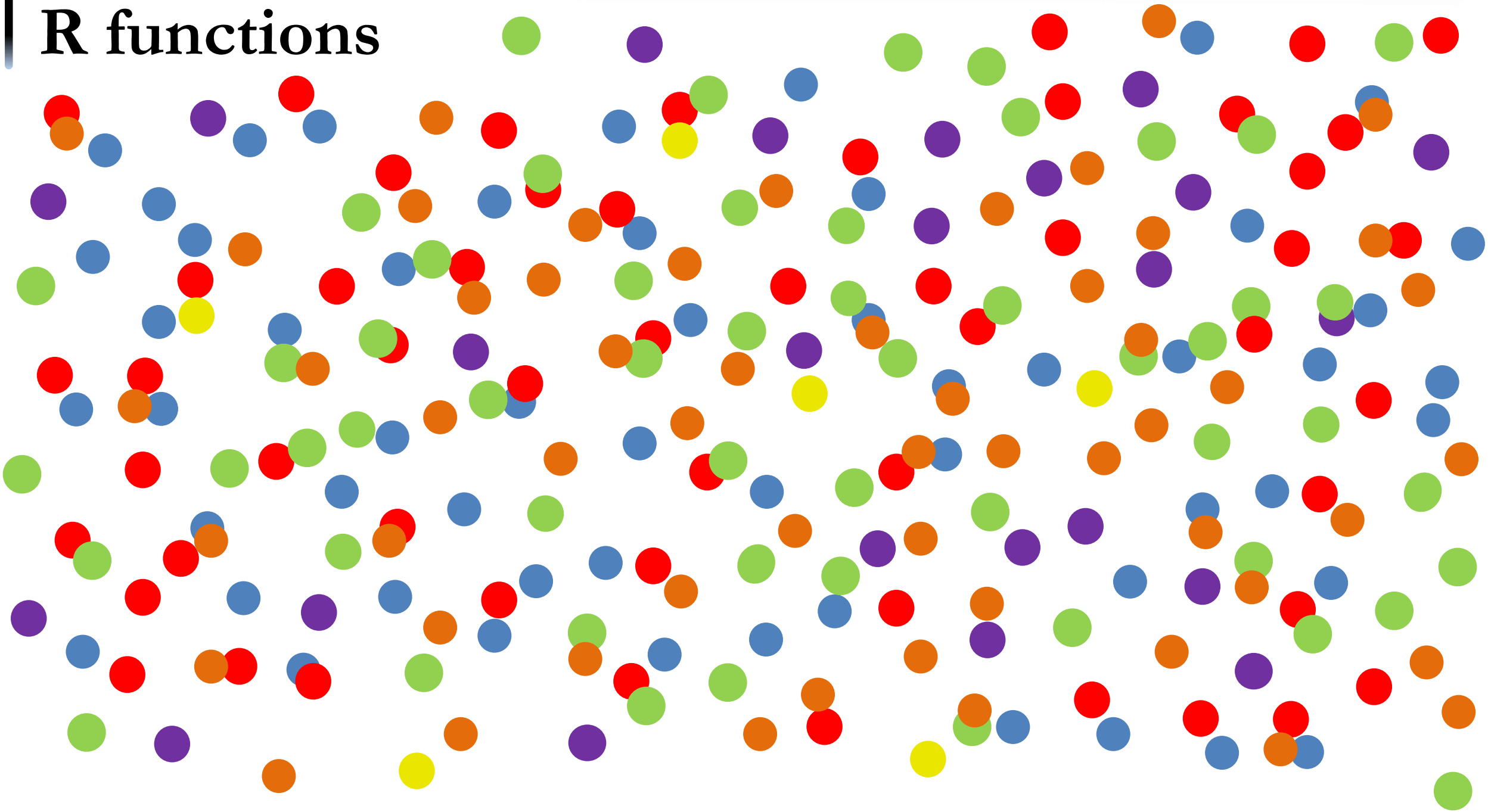
facebook.com/DataScientists...

Kara Woo
@kara_woo

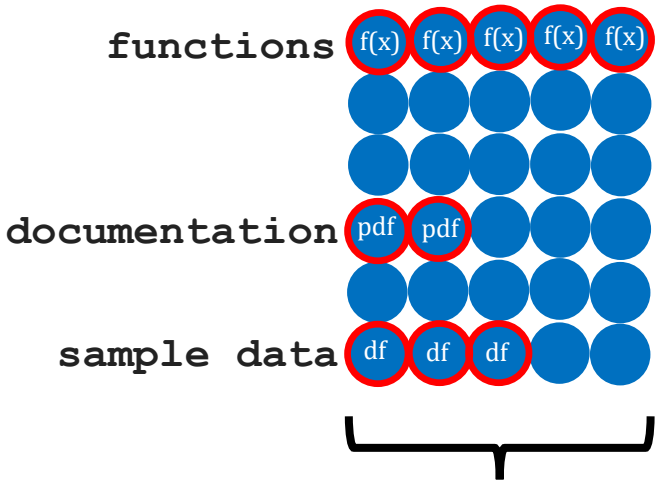
Research scientist at @sagebio. Data curation, visualization, #rstats, reproducibility, open science, ballet

karawoo.com

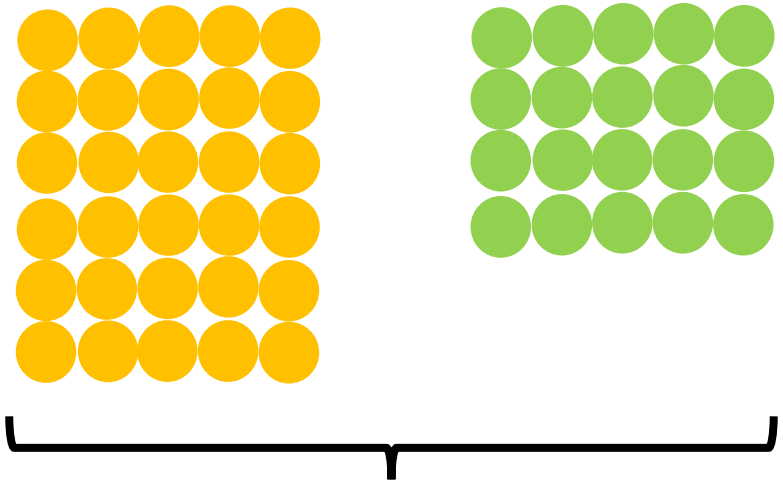
R functions



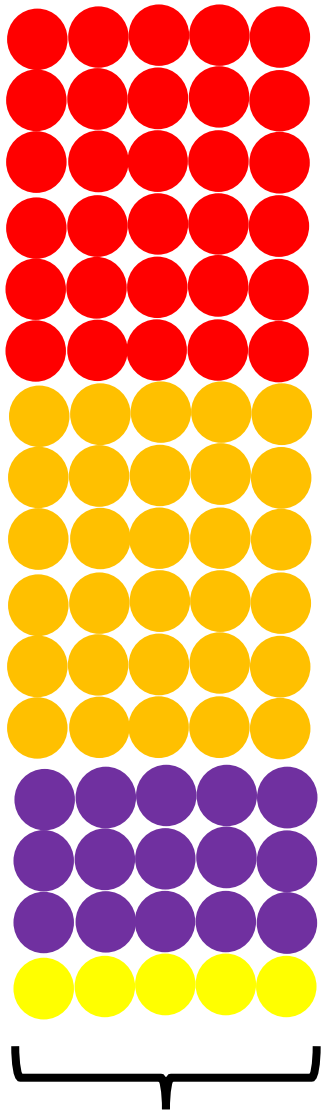
R packages



R comes pre-loaded with ~30 other packages (e.g. base, stats, graphics etc.)



Other packages:
Install once
Update regularly
Load each session



tidyverse

So many packages, so little time

https://cran.r-project.org/web/views/

CRAN Task Views

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. They give a brief overview of the included packages and can be automatically installed using the ctv package. The views are intended to have a sharp focus so that it is sufficiently clear which packages should be included (or excluded) - and they are *not* meant to endorse the "best" packages for a given task.

- To automatically install the views, the ctv package needs to be installed, e.g., via
  `install.packages("ctv")`
  and then the views can be installed via `install.views` or `update.views` (where the latter only installs those packages are not installed and up-to-date), e.g.,
  `ctv::install.views("Econometrics")`
  `ctv::update.views("Econometrics")`
- The task views are maintained by volunteers. You can help them by suggesting packages that should be included in their task views. The contact e-mail addresses are listed on the individual task view pages.
- For general concerns regarding task views contact the ctv package maintainer.

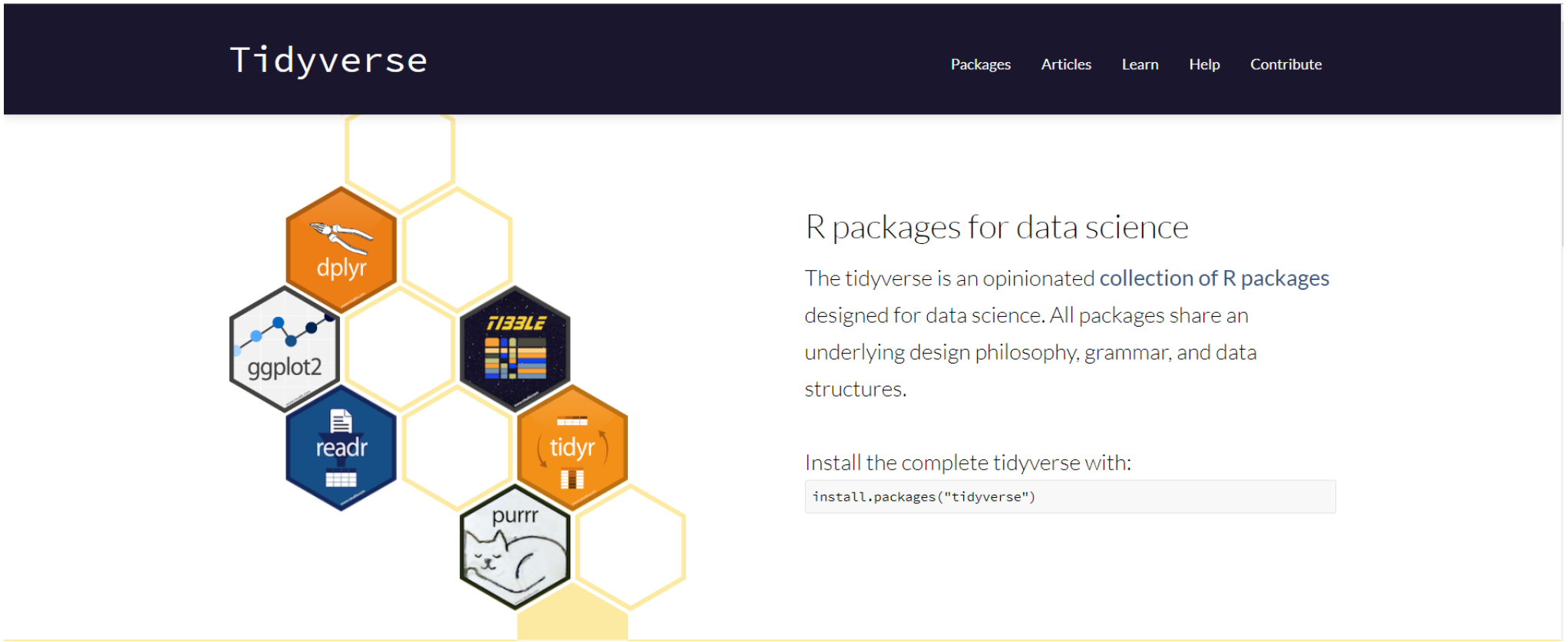
Topics

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data

To understand R, remember the following

- Everything that exists is an object
- Everything that happens is a function

What is the tidyverse?

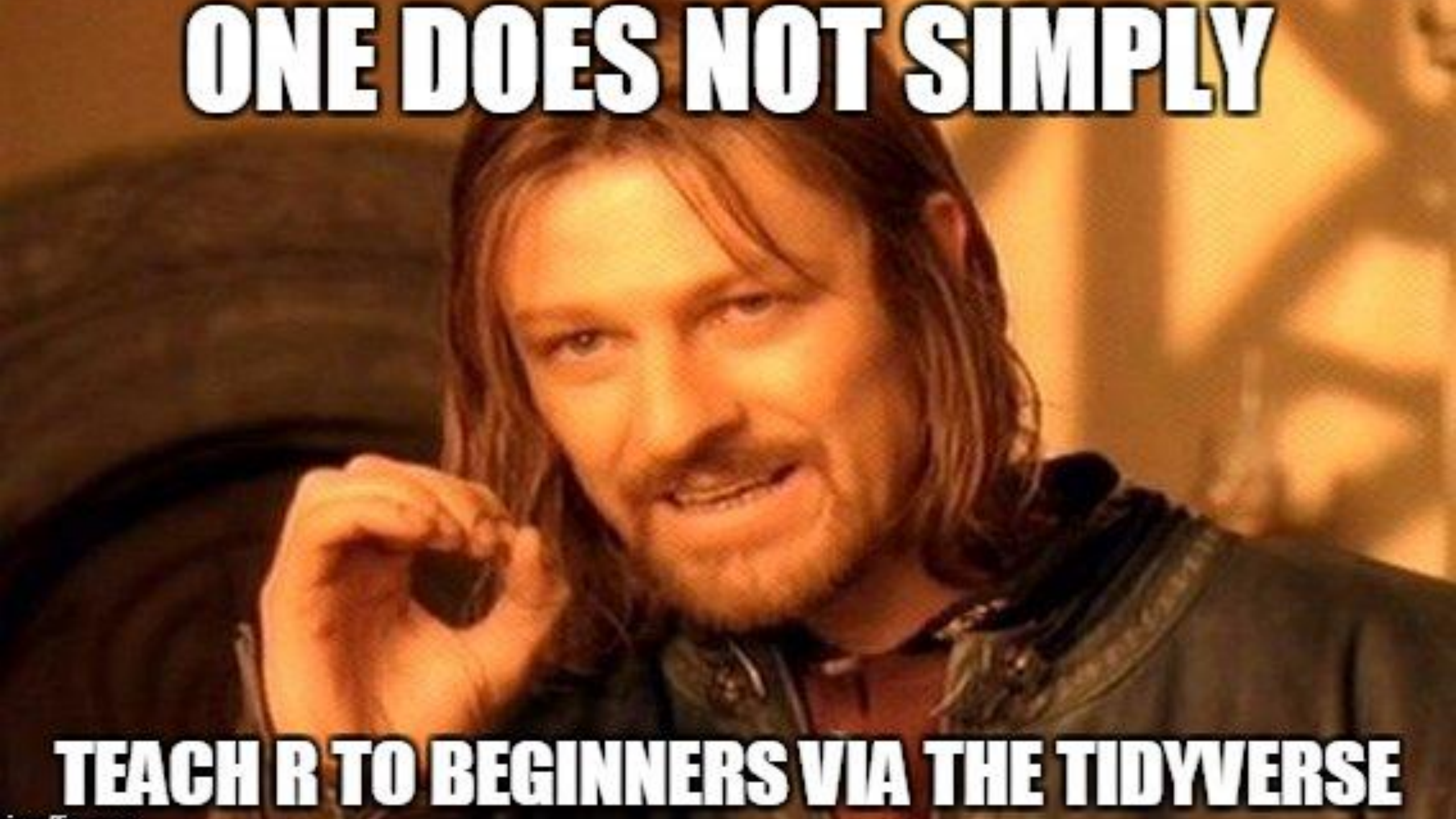


- Joined up collection of packages for data analysis
    - Consistent functions
    - Uses (tidy) data
    - Supports end-to-end workflows

What is the tidyverse?

```
> install.packages(c(“broom”, “cli2”, “crayon”,
“dbplyr”, “dplyr”, “forcats”, “ggplot2”, “haven”,
“hms”, “httr”, “jsonlite”, “lubridate”,
“magrittr”, “modelr”, “pillar”, “purrr”, “readr”,
“readxl”, “reprex”, “rlang”, “rstudioapi”,
“rvest”, “stringr”, “tibble”, “tidyr”, “xml2”)
```

```
> install.packages(“tidyverse”)
```


ONE DOES NOT SIMPLY
TEACH R TO BEGINNERS VIA THE TIDYVERSE

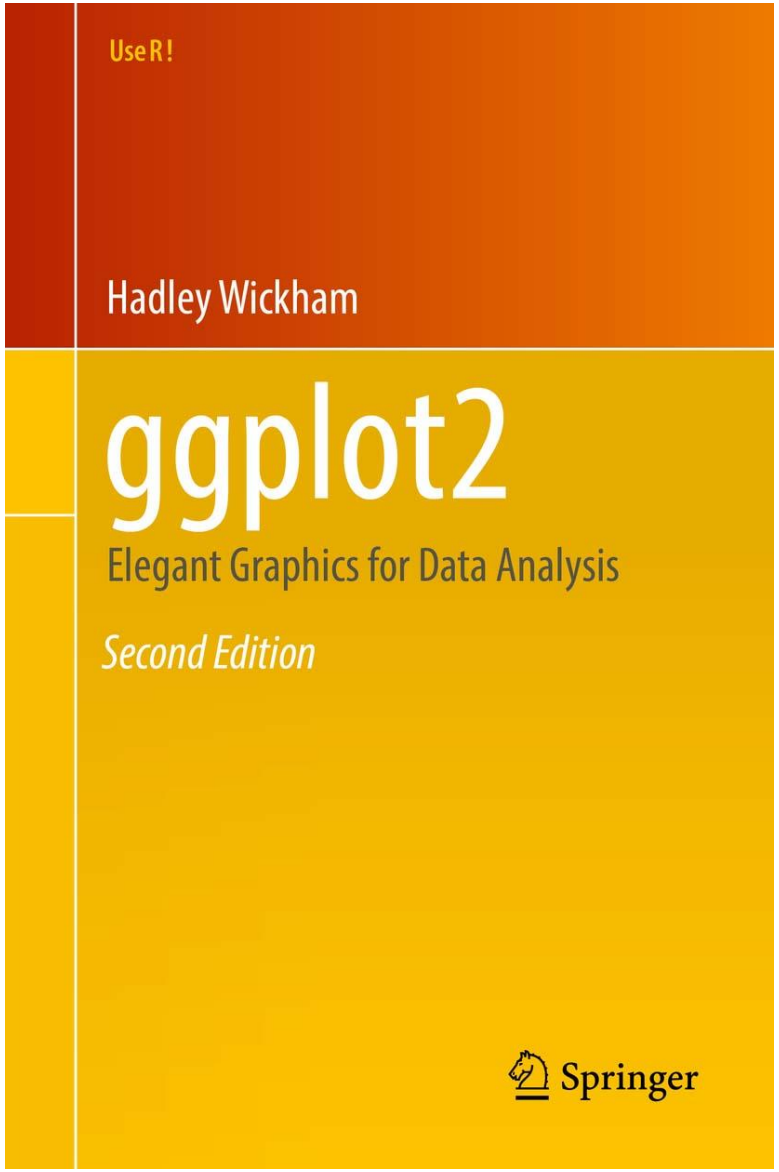
The tidyverse Oct 2017

```
> library(tidyverse)
Loading tidyverse: ggplot2
Loading tidyverse: tibble
Loading tidyverse: tidyr
Loading tidyverse: readr
Loading tidyverse: purrr
Loading tidyverse: dplyr
```

The tidyverse May 2019

```
> library(tidyverse)
-- Attaching packages --------------------------------------- tidyverse 1.2.1 --
v ggplot2 3.1.1       v purrr   0.3.2
v tibble  2.1.1       v dplyr   0.8.0.1
v tidyr   0.8.3       v stringr 1.4.0
v readr   1.3.1       v forcats 0.4.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

You could write a book on that!!



R user interface versus RStudio

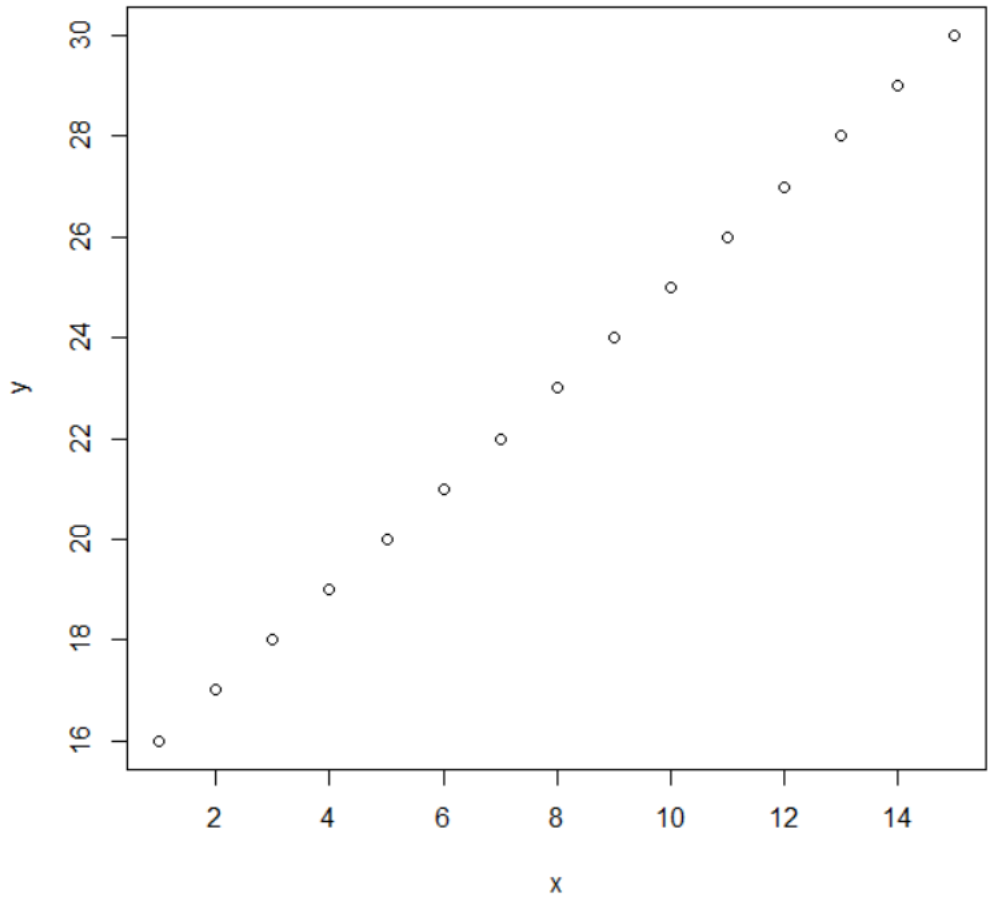
RGui (64-bit)

File Edit View Misc Packages Windows Help

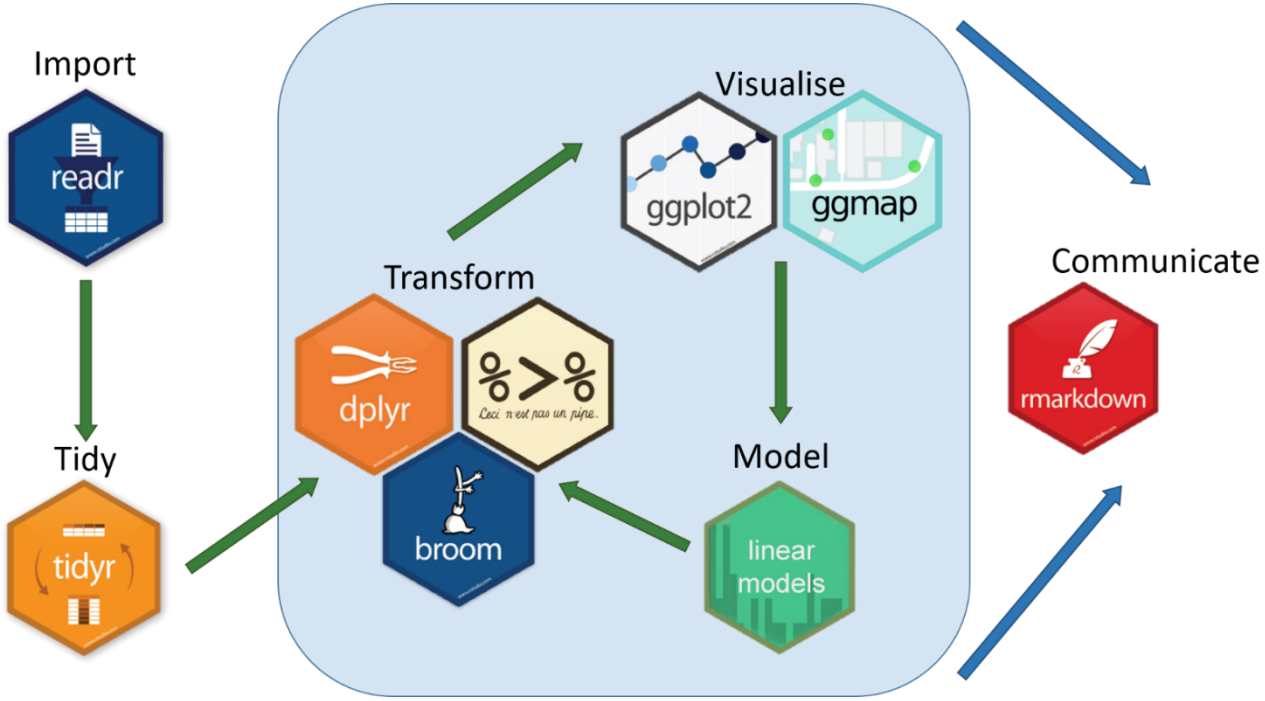
R Console

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 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.

  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.

[Previously saved workspace restored]

> print("Hello world!!")
[1] "Hello world!!"
> object <- "Hello world!!"
> object
[1] "Hello world!!"
> x <- 1:15
> y <- 16:30
> z <- mean(x + y)
> z
[1] 31
> plot(x,y)
>
```

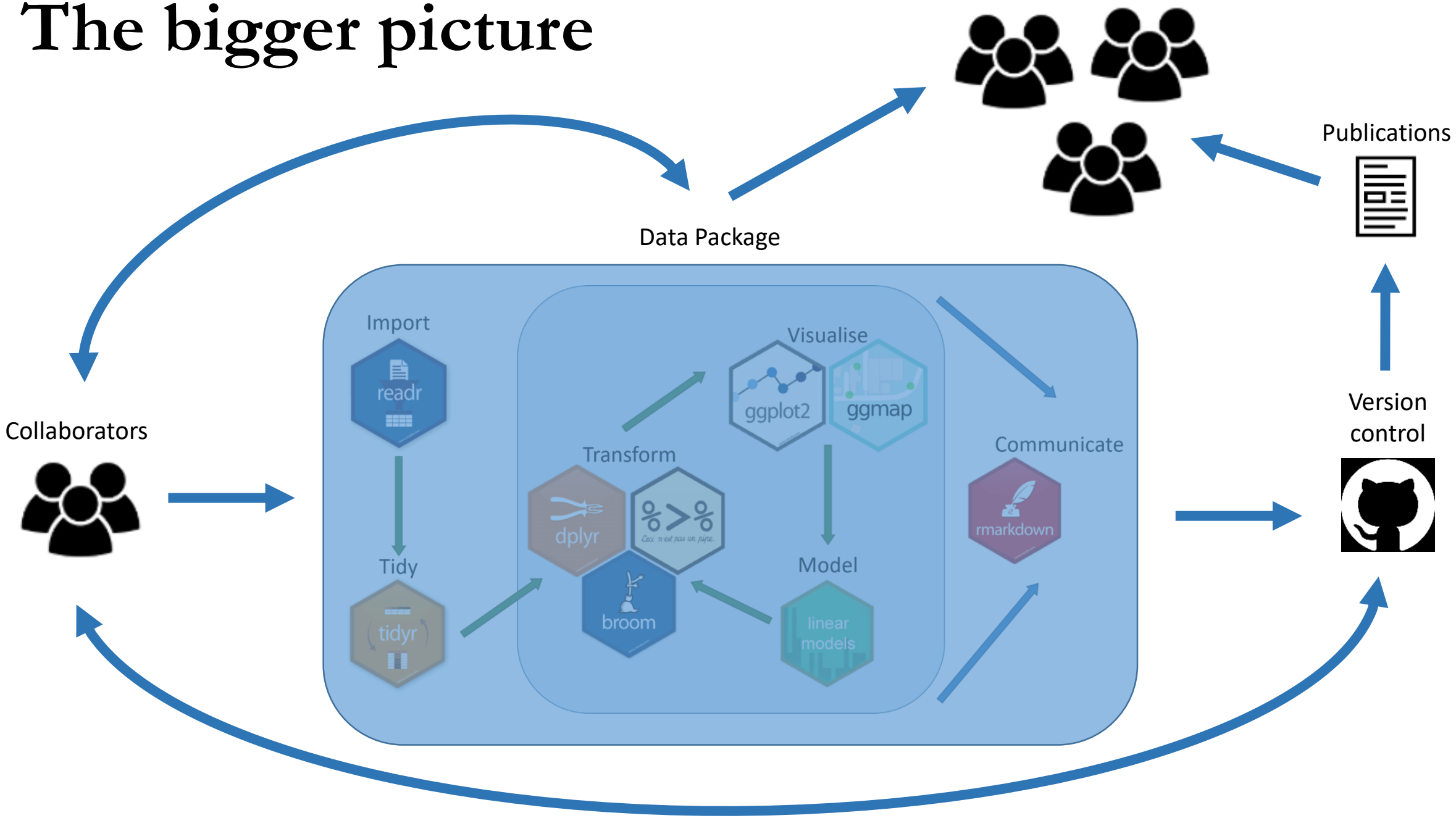
R Graphics: Device 2 (ACTIVE)



Putting the pieces together via the tidyverse



The bigger picture



File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Pre-workshop

test_script.R

Source on Save | Run | Source

```
31
32   gather(sample, expression, G0.05:U0.3) %>%
33
34   separate(sample, c("nutrient", "rate"), sep = 1, convert = TRUE) %>%
35
36   mutate(nutrient = plyr::revalue(nutrient, nutrient_names)) %>%
37
38   filter(!is.na(expression), systematic_name != "")
39
40 # Plot the clean data
41
42 cleaned_genes_tbl %>%
43
44   filter(BP == "leucine biosynthesis") %>%
45
46   ggplot(mapping = aes(x = rate, y = expression, color = nutrient)) +
47   geom_point() +
48   geom_smooth(method = "lm", se = FALSE) +
49   facet_wrap(~ name)
```

Code editor

49:21 | Does this work on your system? | R Script

Console | Terminal

/R_Users_Workshop/PG_module/course_notes/R-A_Hitchhikers_Guide_to_Reproducible_Research/Pre-workshop/

```
  GID = col_character(),
  YORF = col_character(),
  NAME = col_character()
)
See spec(...) for full column specifications.
> cleaned_genes_tbl %>%
+
+   filter(BP == "leucine biosynthesis") %>%
+
+   ggplot(mapping = aes(x = rate, y = expression, color = nutrient)) +
+   geom_point() +
+   geom_smooth(method = "lm", se = FALSE) +
+   facet_wrap(~ name)
>
```

R console

Environment | History | Connections | Git

Import Dataset | Grid

Global Environment

Name	Type	Length	Size	Value
cleaned_g...	tbl_df	7	11.3 ...	198430 obs. of 7...
nutrient_...	charac...	6	984 B	Named chr [1:6] "G...
url	charac...	1	168 B	"http://varianceex...

Environment

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish

Files/Plots/Help

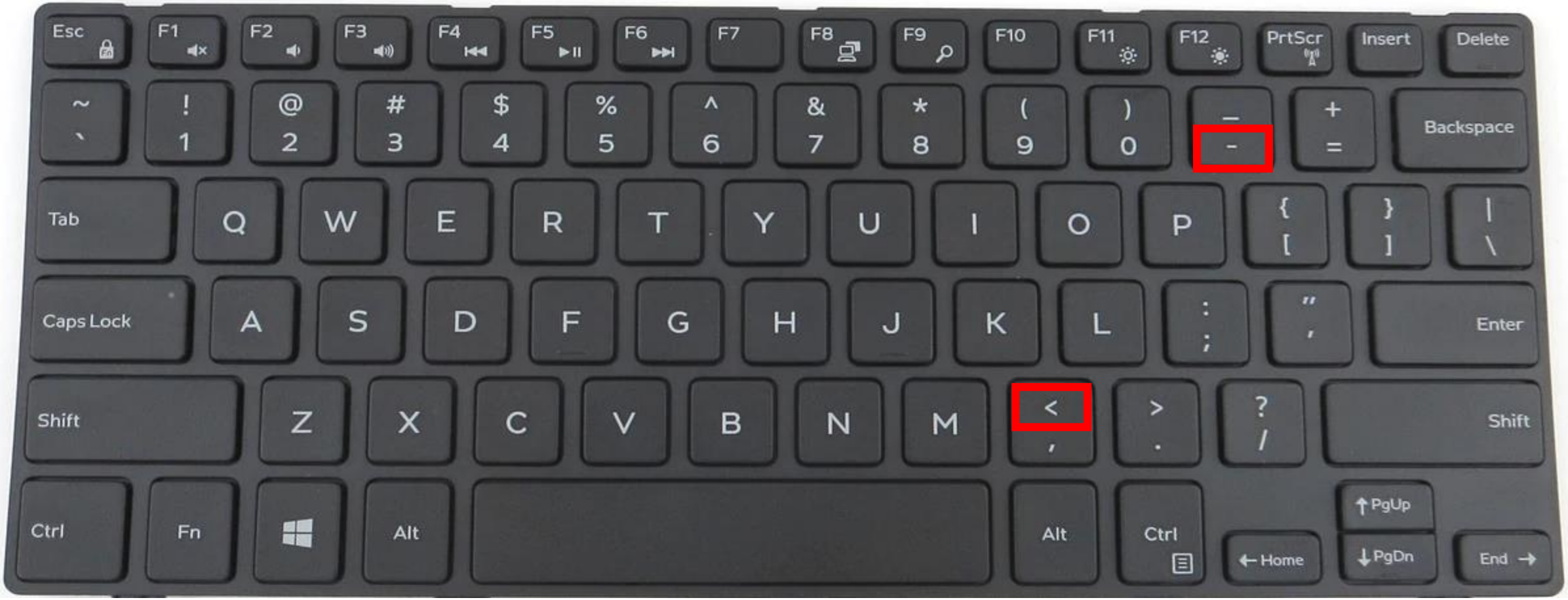
Basics of R code

Symbol	What it does	Example 1	Example 2
<-	Assign operator Creates new objects	> x <- 5 > x [1] 5	> y <- "This" > y [1] "This"
c()	Helps create objects with more than one element	> v <- c(5,6,7,8) > v [1] 5 6 7 8	> w <- c("This", "is", "easy! ") > w [1] "This" "is" "easy!"
#	Computer ignores what is written. Used for adding notes to code	> # print(“hello”) >	> print(“hello”) [1] "hello"
%>%	Literally translates as “then do this”	> data %>% do_something_to(data)	> data %>% do_something_to(data) %>% do_something_else_to(data)
%in%	returns a logical vector indicating if there is a match	> "x" %in% c("x", "y", "z") [1] TRUE	> c("x", "y", "z") %in% "x" [1] TRUE FALSE FALSE
?	Access help	> ?mean()	> ?geom_point()

FYI: R is case sensitive!! Name.of.data ≠ name.of.data

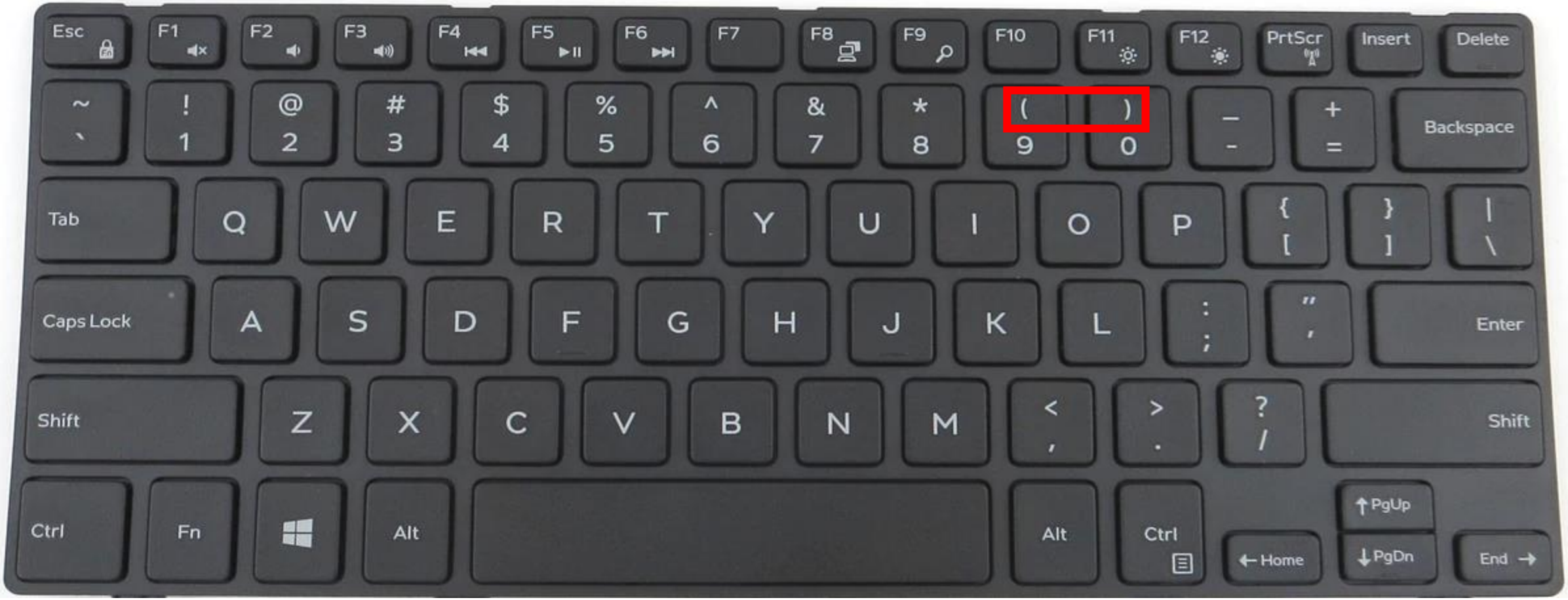
Say hello to the lesser known keyboard keys

Assignment operator: <-



Say hello to the lesser known keyboard keys

Functions take arguments inside round brackets: function()



Say hello to the lesser known keyboard keys

Indexing occurs inside square brackets: **[]**

Say hello to the lesser known keyboard keys

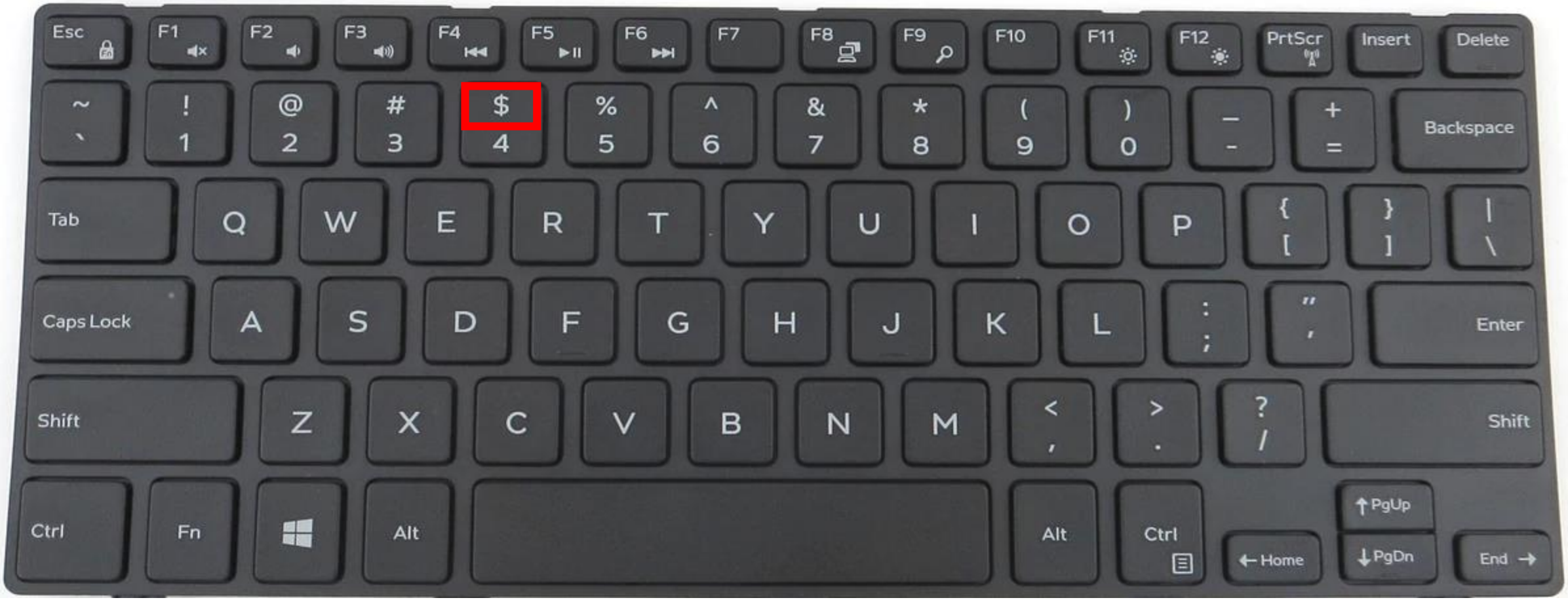
Functions are defined inside curly brackets: { }

Say hello to the lesser known keyboard keys

You can comment out your code using the hash key: **#**

Say hello to the lesser known keyboard keys

Dollar sign allows you extract elements by name: **$**



Say hello to the lesser known keyboard keys

Logical TRUE/FALSE operators equals, not equals, and, or:

==, !=, &, |

Say hello to the lesser known keyboard keys

Tilde operator for use in formulas: ~

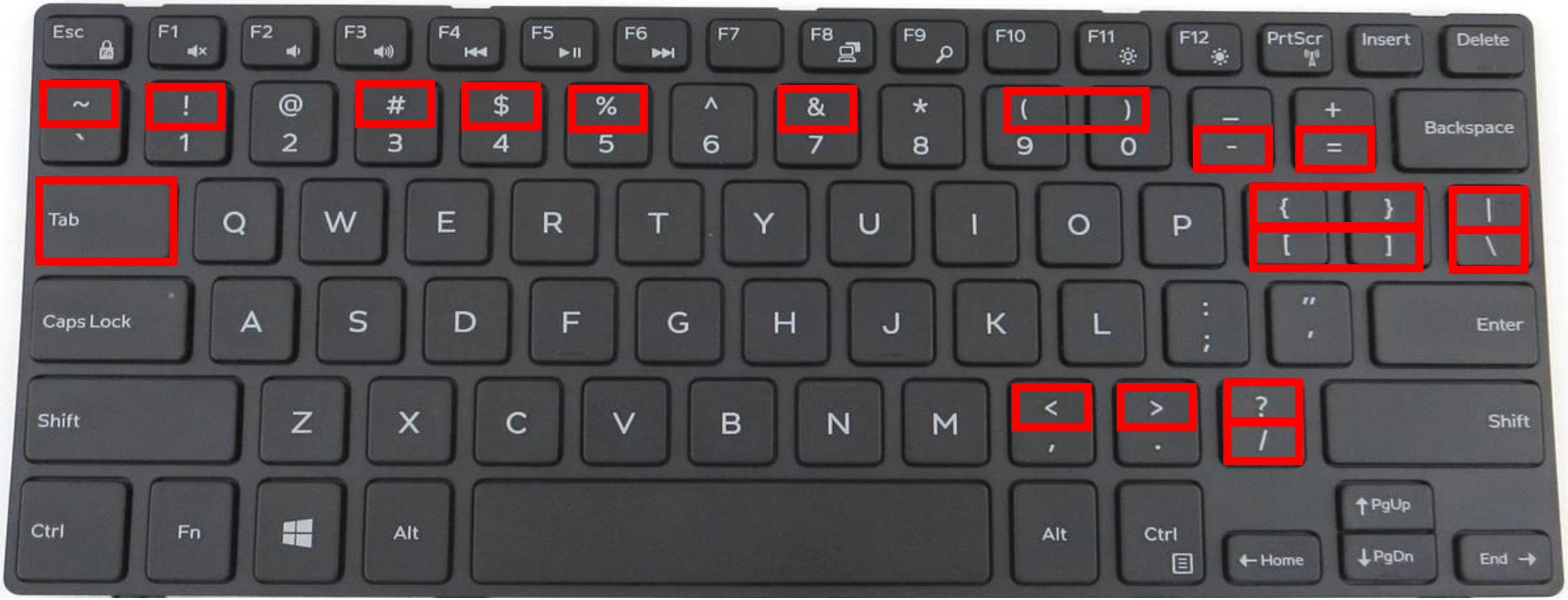
Say hello to the lesser known keyboard keys

Tab key for autocomplete

Say hello to the lesser known keyboard keys

Backslash and forward slash

Say hello to the lesser known keyboard keys



Creating objects

For most of us, R is simply the creation of and manipulation of objects:

```
new_object <- c(1, 2, 3)
```

- the objects are then fed into functions to create amazing new objects

```
amazing_new_object <- function(new_object)
```

Broadly speaking the following is true in R:

- information

```
> data_frame  <- function(information)
> plot        <- function(data_frame)
> model       <- function(data_frame)
```

Naming objects

There are a few simple rules to follow initially:

- Object names must start with a letter and can only contain letters, numbers, '_' and '.'

- Certain characters should not be used, e.g:
    - c is the concatenate function 'c()'
    - T is used as shorthand for TRUE
    - F is used as shorthand for FALSE

- In this course I'll always use x, y and z as object names when demonstrating quick examples

Types of data structure

The main data types are;

```
# double (for double precision floating point numbers)
typeof(1.23)

# character
typeof("string")

# logical
typeof(FALSE)

# missing values are represented by NA
example <- c(1, 2, NA, 4)
```

Other examples include integers and complex numbers

Types of data structure

```
- Vectors come in two forms

A: Atomic vectors contain exactly one type of data

all_numbers          <- c(1, 2, 0.5, -0.5, 3.4)
all_characters       <- c("One", "too", "3")
all_logical          <- c(TRUE, FALSE) # NOTE: Type it out

B: Lists allow combinations of different types of data

this_is_a_list       <- list(1, TRUE, "Three", "4")
typeof(this_is_a_list)
[1] "list"

this_is_also_a_list <- list(all_numbers, all_characters)
```

Types of data structure

```
# Matrices/Arrays:

- You can have a matrix of two or more dimensions
  a_matrix <- matrix(1:9, 3, 3)

- Vectors and matrices can only contain one type of data

- VERY VERY VERY NB: If you try to create a vector with more
  than one data type, then it will undergo coercion to the
  least common denominator

- The coercion rule goes:
     logical -> integer -> numeric -> complex -> character

- You can perform coercions yourself on vectors
```

Walkthrough example script

- 01_baseR_introduction.R
  - Basic Code entry

Types of data structures

```
# Dataframes:
- These are a special type of list
- Observations are in rows
- Variables are in columns
- Labels or other metadata may also be present

> a_data_frame <- data.frame(number = 1:10,
                             char = sample(letters, 10),
                             this_really_a_col_name  = rep(c(TRUE, FALSE), 5))

- In the tidyverse dataframes are called 'tibbles'

- Some older functions don't work with tibbles

- We'll go through this in more detail later
```

Indexing

- Indexing can occur in one or two dimensions
- One dimension:

```
new_object <- c(1, 2, 3)
new_object[1]
[1] 1
```

- Two dimensions

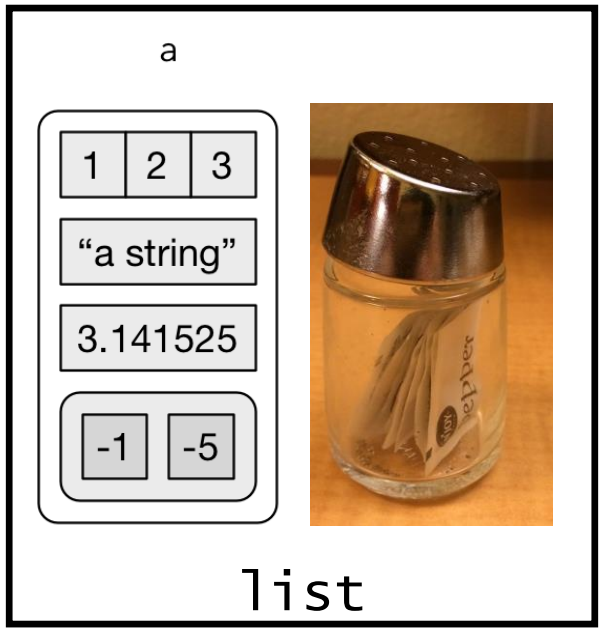
```
a_data_frame[1, 1] # i.e [Row number 1, Column number 1]
a_data_frame$number[1] # i.e. Column called number, row 1
```

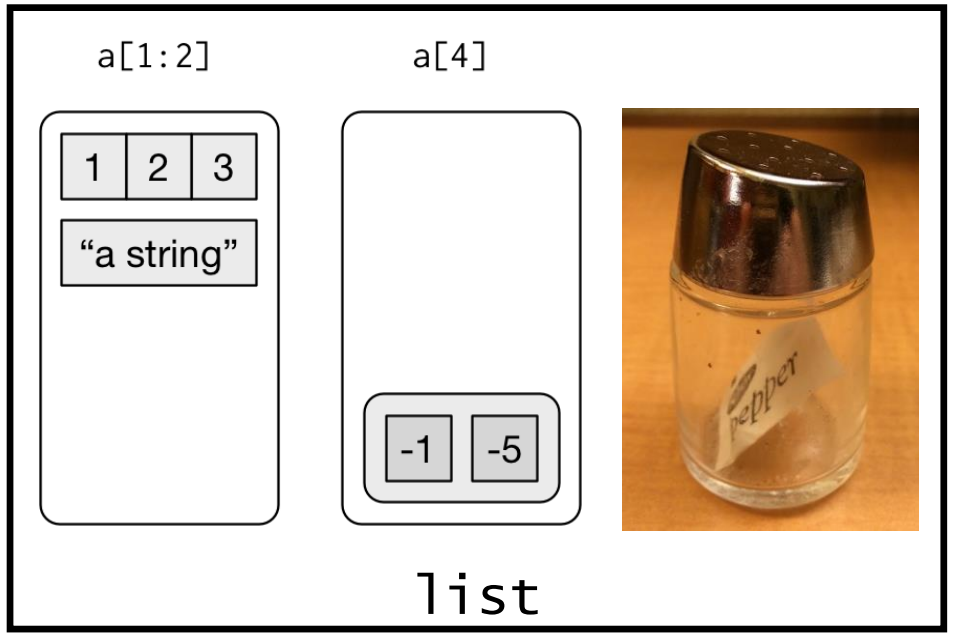
- In the tidyverse we don't use '[' much as `dplyr::filter()` and `dplyr::select()` allow you to solve the same problems

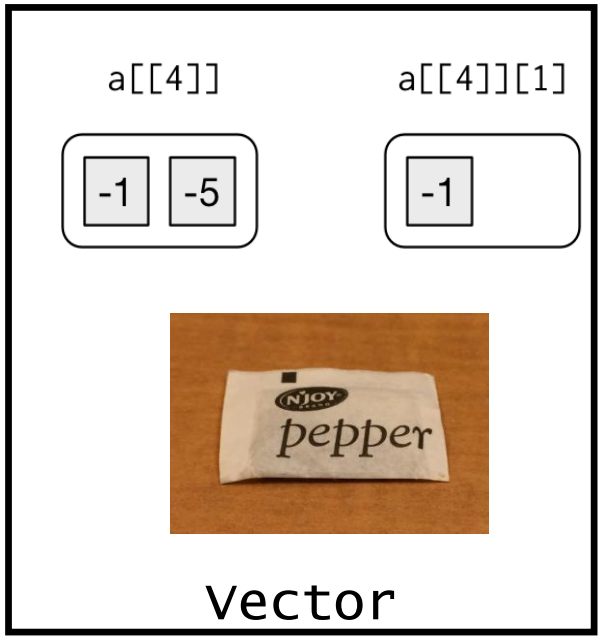
- However, given so much of the R has been written using these, it's worth recognising and understanding them

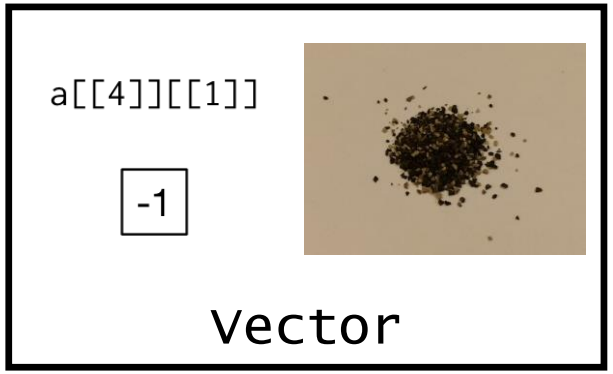
Indexing

```
# Recall
- this_is_also_a_list <- list(all_numbers, all_characters, all_logical)
```










```
# Important
-   [ extracts a sublist, results will be a list
-   [[ extracts a single component
```

Image: R for Data Science, Wickham and Grolemund

Walkthrough example script

- 01_baseR_introduction.R
  - Indexing dataframes and lists

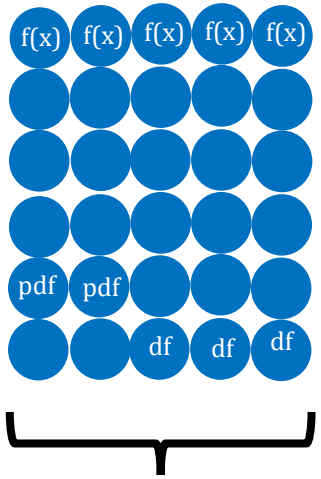
Types of data structures

```
# Factors:
- In R, factors are used to work with categorical variables

- Historically they were easier to work with than characters,
  hence many baseR functions automatically convert characters
  to factors

- This does not happen in the tidyverse

- One of the most important uses of factors is in statistical
  modeling;
      - since categorical variables are entered into
        statistical models differently than continuous
        variables, storing data as factors ensures that the
        modeling functions will treat such data correctly
```

Walkthrough example script

- 01_baseR_introduction.R
  - Factors example

Package contents



Base R: Comes pre-loaded

Functions

```
> base::
max.col                {base}
mean                   {base}
mean.Date              {base}
mean.default           {base}
mean.difftime          {base}
mean.POSIXct           {base}
mean.POSIXlt           {base}
mem.limits             {base}

mean(x, ...)
Generic function for the (trimmed) arithmetic mean.
Press F1 for additional help
```

Data sets

```
> data()
faithful
freeny
infert
iris
iris3
islands
lh
longley

iris
This famous (Fisher's or Anderson's) iris data set gives the measurements in
centimeters of the variables sepal length and width and petal length and
width, respectively, for 50 flowers from each of 3 species of iris. The species
are Iris setosa, versicolor, and virginica.
Press F1 for additional help
```

Conflicts

```
> filter
filter      {dplyr}
filter_     {dplyr}
filter_all  {dplyr}
filter_at   {dplyr}
filter_if   {dplyr}
Filter      {base}
Filters

filter(x, filter, method = c("convolution", "recursive"),
    sides = 2L, circular = FALSE, init = NULL)
Applies linear filtering to a univariate time series or to each series separately
of a multivariate time series.
Press F1 for additional help
```

Walkthrough example script

- `02_navigating_R_packages.R`

Worksheet script

- 03_practice_worksheet.R