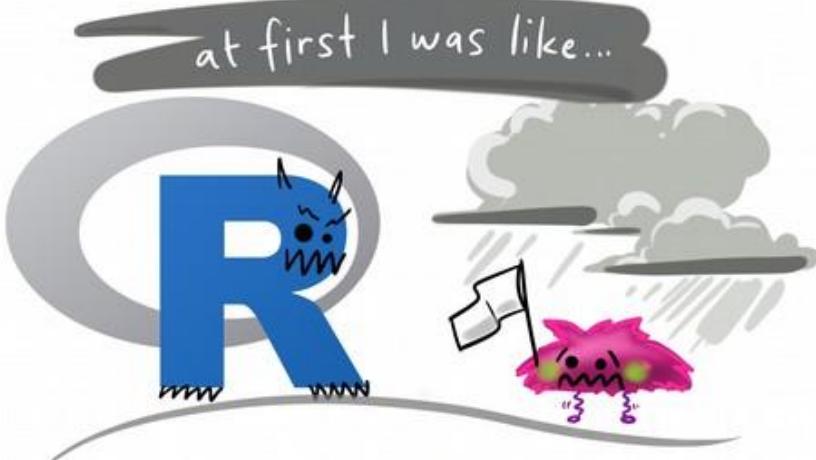


Our goal

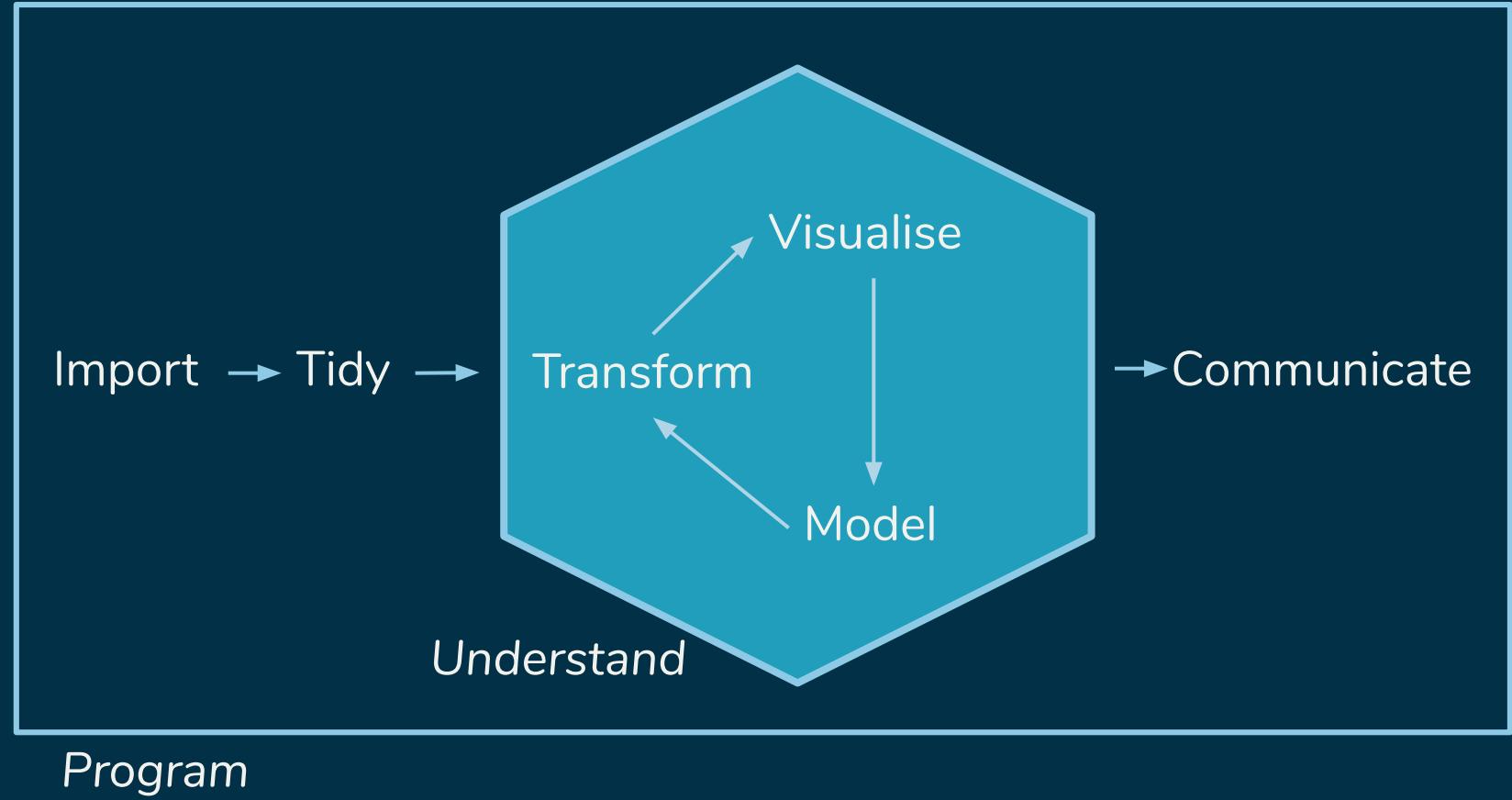


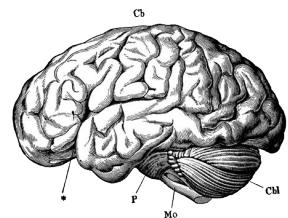
...but now it's like...



Artwork by @allison_horst

Data science is the art
of turning raw data into
understanding

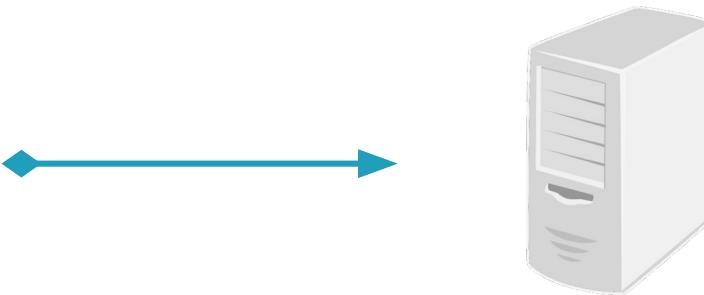
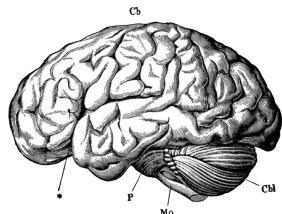




Human thought



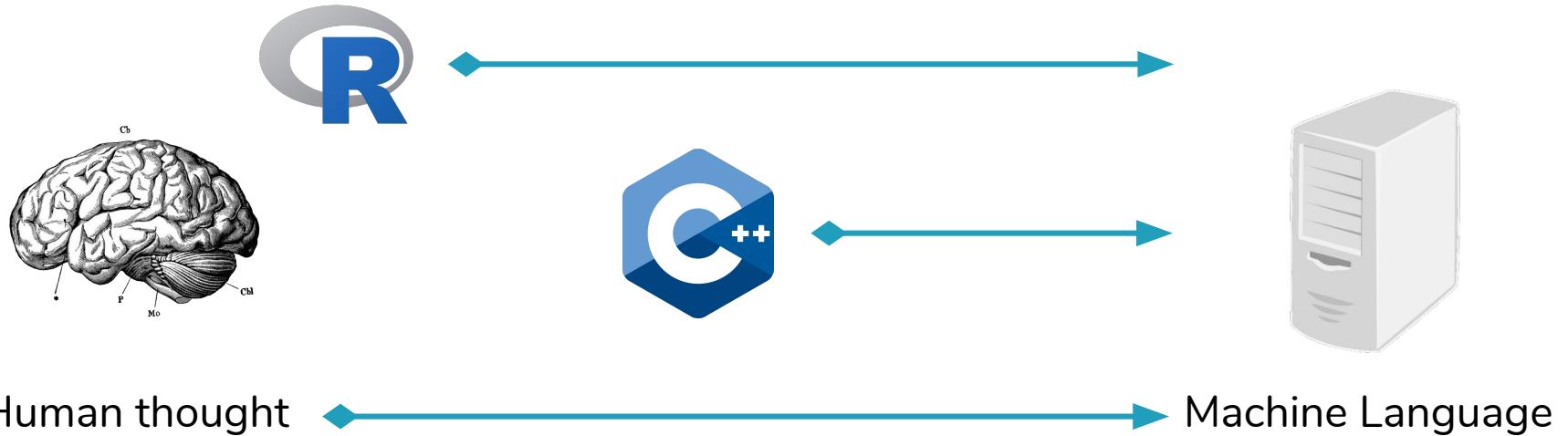
Machine Language



Human thought

Machine Language





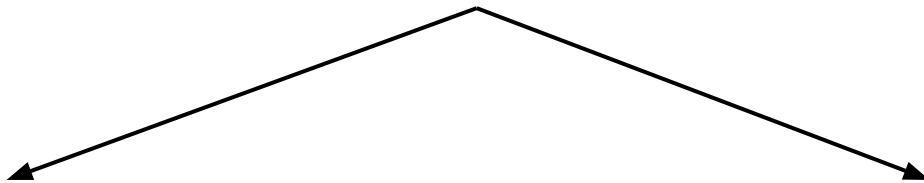
How do we process
the data?

```
function(arg1, arg2, arg3, ...)
```

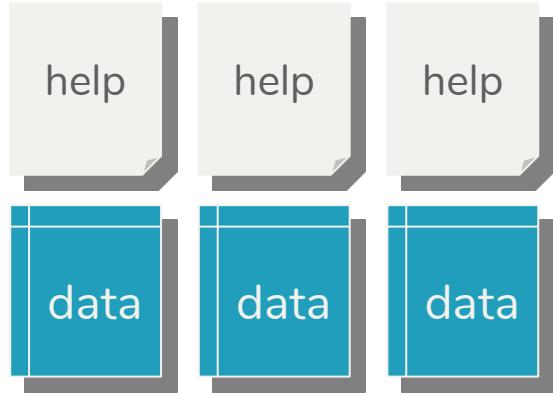
```
change_the_env(...)
```

```
calculate_value(...)
```

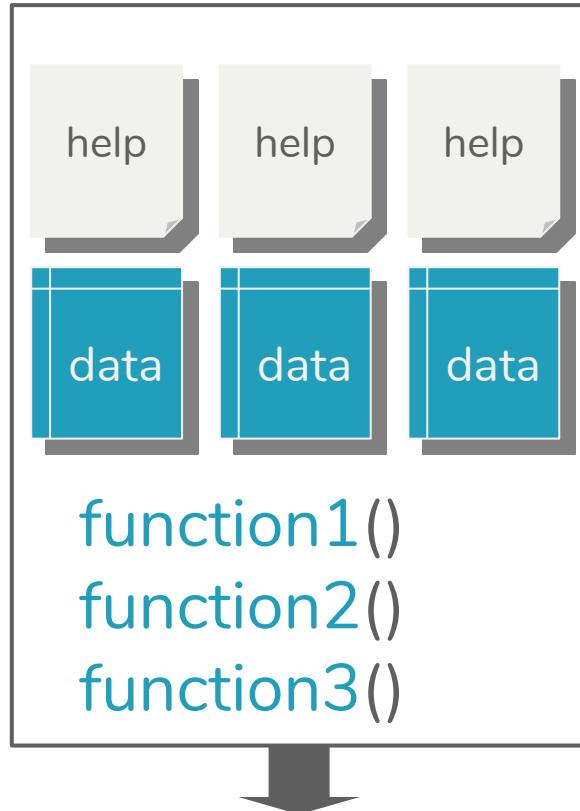
assign. <- , =, ->



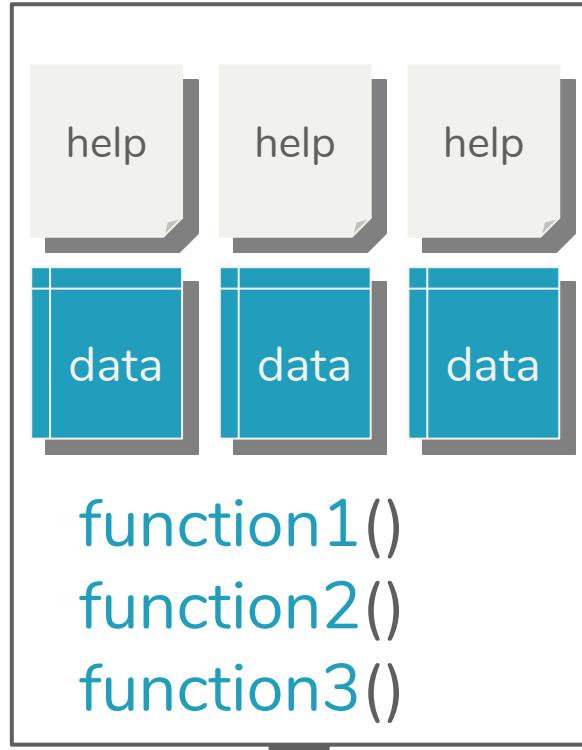
- arguments are contexts of a function
- arguments are matched by name, or
- arguments are matched by position, **be careful!**



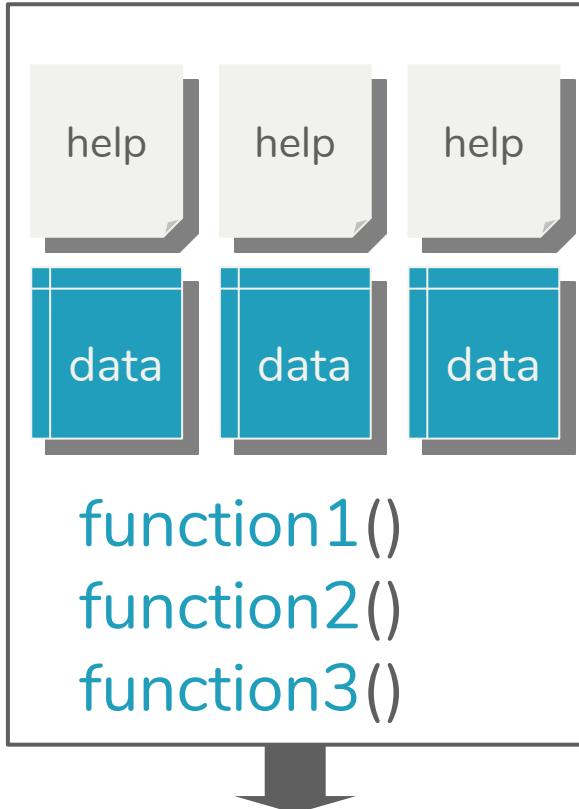
`function1()`
`function2()`
`function3()`



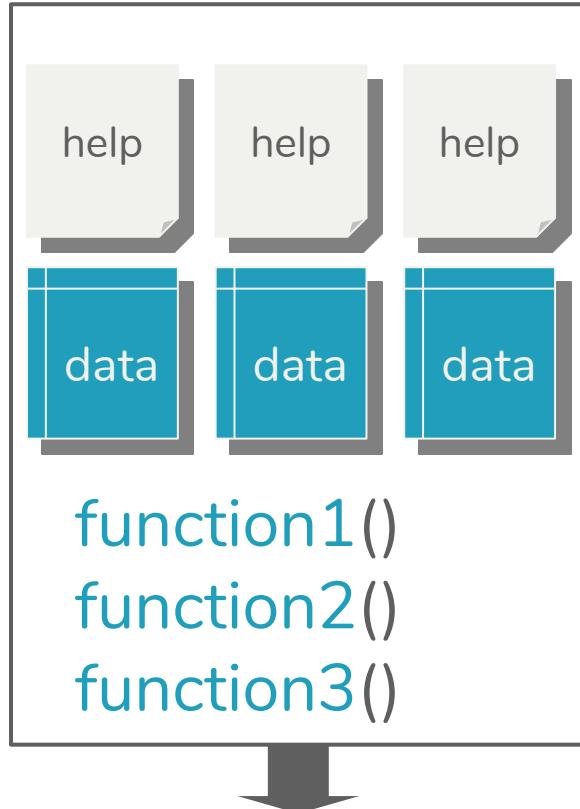
stats, graphics, grDevices, utils,
datasets, methods, base



stats, graphics, grDevices, utils,
datasets, methods, base



R packages



CRAN Task Views

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

The screenshot shows a web browser window with the title "The Comprehensive R Arc" and the URL "https://cran.r-project.org". The page content is the "CRAN Task Views" section of the CRAN website. It features a large blue "R" logo on the left, followed by a sidebar with navigation links like "CRAN Mirrors", "What's new?", "Task Views", "Search", "About R", "R Homepage", and "The R Journal". Below the sidebar is a "Software" section with links for "R Sources", "R Binaries", "Packages", and "Other". Further down is a "Documentation" section with links for "Manuals", "FAQs", and "Contributed". The main content area is titled "CRAN Task Views" and contains a paragraph about the purpose of task views. It includes a bulleted list of instructions for installing and maintaining task views, and a "Topics" section listing various R packages categorized by topic.

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
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance
FunctionalData	Functional Data Analysis
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

GitHub

[https://github.com/
search?q=r](https://github.com/search?q=r)

Search · r · GitHub

GitHub, Inc. [US] | https://github.com/search?q=r

Why GitHub? Enterprise Explore Marketplace Pricing

Sign in Sign up

Repositories 5M+

- Code ?
- Commits 54M+
- Issues 2M
- Marketplace 9
- Topics 1
- Wikis 232K
- Users 33K

R

R is a free programming language and software environment for statistical computing and graphics.

See topic

★ Star

Showing 5,189,217 available repository results Sort: Best match

dmpe/R

Exercises (incl. analyses) with R language (math+statistics)

r data-analysis exercise statistics course

MIT license Updated on Oct 28, 2018

shifteight/R

R exercises and examples

HTML ★ 50

Updated on Dec 8, 2018

Installing package, only once

```
install.packages("pkg") # from CRAN/local  
remotes::install_github("user/pkg") # from GitHub  
remotes::install_bioc("repo") # from Bioconductor
```

Loading package, once in every session

```
library(pkg)  
pacman::p_load(pkg) # load or install if not  
available
```

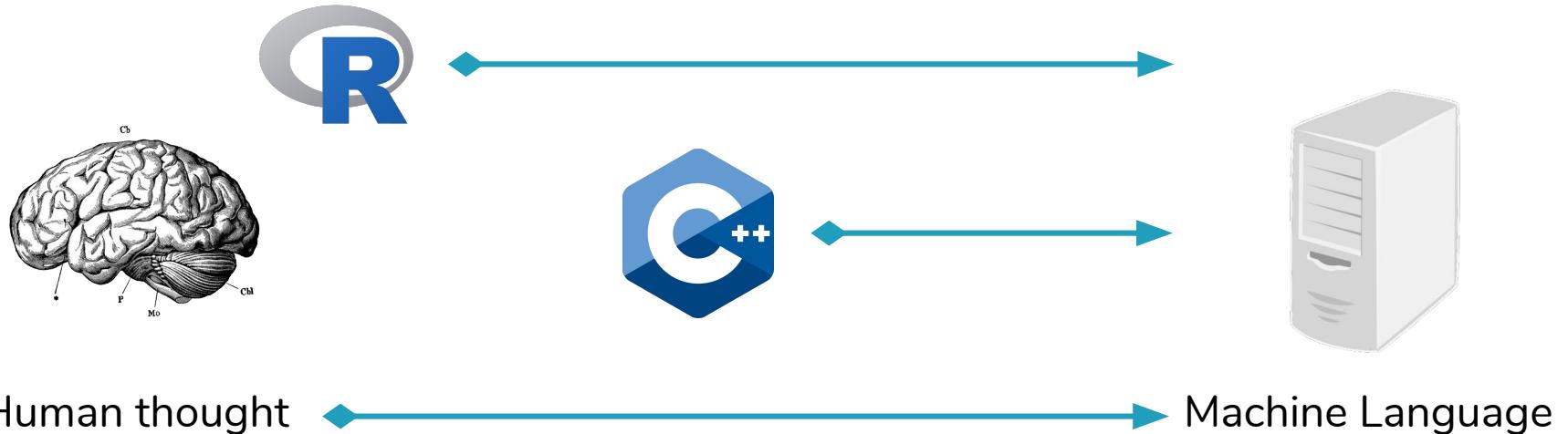
A lot of R packages
to use! :D

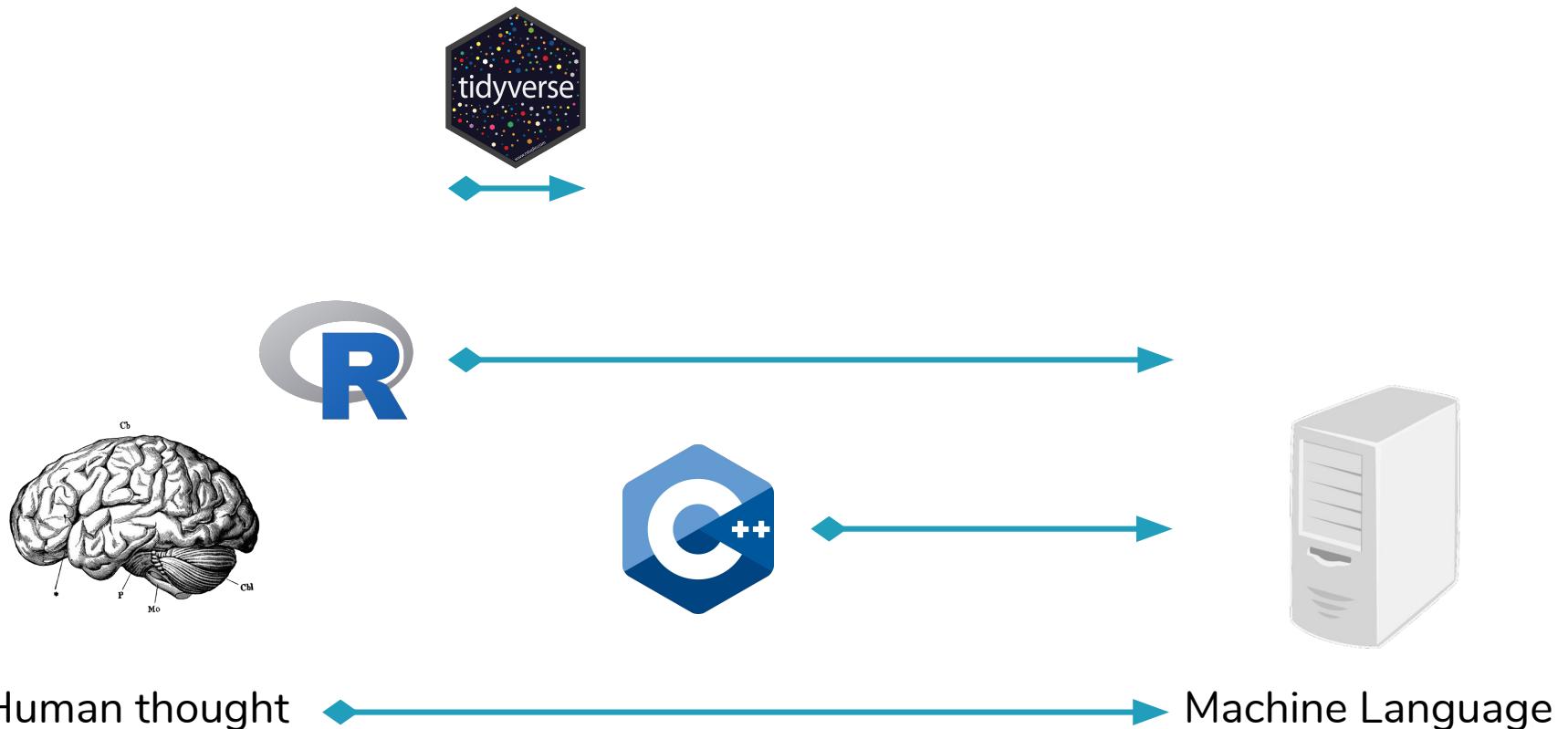
A lot of R packages
to use! :(





The tidyverse is an opinionated **collection of R packages** designed for **data science**. All packages **share** an underlying **design philosophy, grammar, and data structures**.





tidymodels

tidyverts

tidygraph

tidycode

tidync

tidycensus

tidytext

tidygenomics

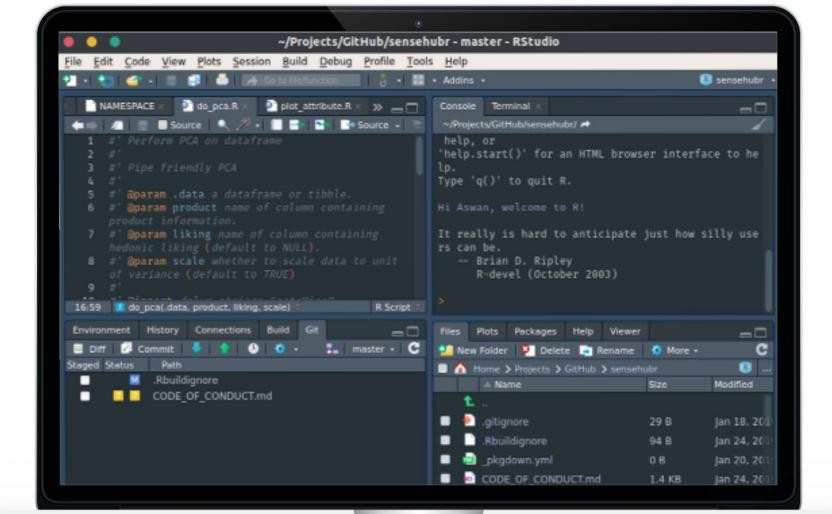


Where to
code?

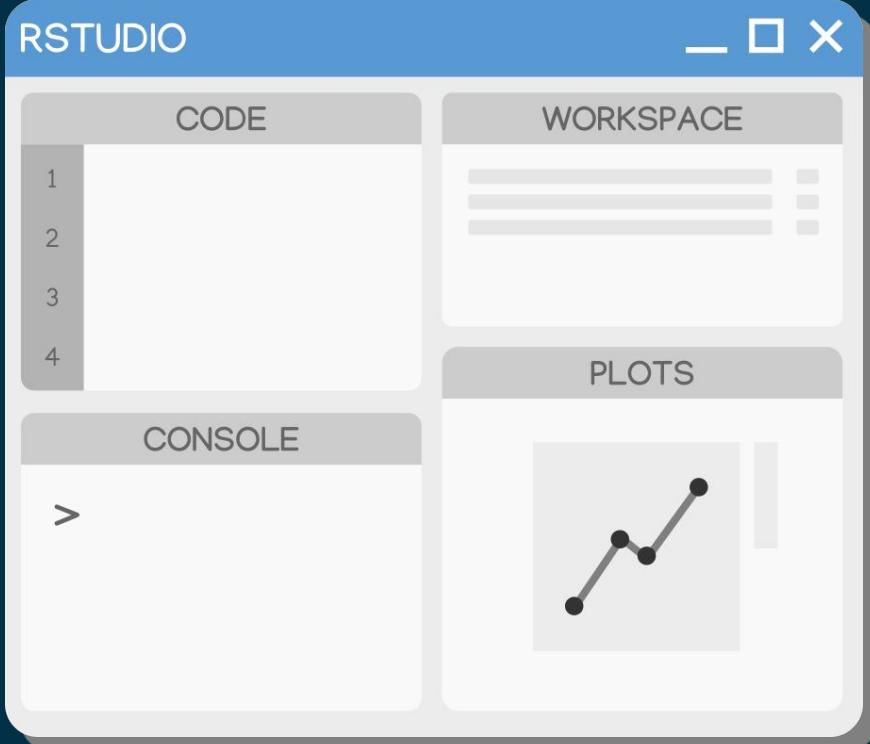


Main features:

- Console
- Syntax-highlighting editor
- Tools for plotting, history, debugging and workspace management



rstudio.com/products/rstudio/download/



- **Tab**, autocomplete & path navigation
- **Alt + -**, for assignment operator <->
- **Ctrl + Shift + M**, for pipe operator %>%
- **Ctrl + Enter**, run current line code/example on help page
- **Ctrl + Up**, search for code history on console or editor pane
- **Alt + Up/Down**, move code to above or below
- **Alt + Shift + Up/Down**, copy code to above or below
- **Ctrl + D**, delete current line
- **Ctrl + Shift + F10**, restart R session
- **Ctrl + Alt + B**, run code up to current line

Cheatsheet: <https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf>

Rmarkdown

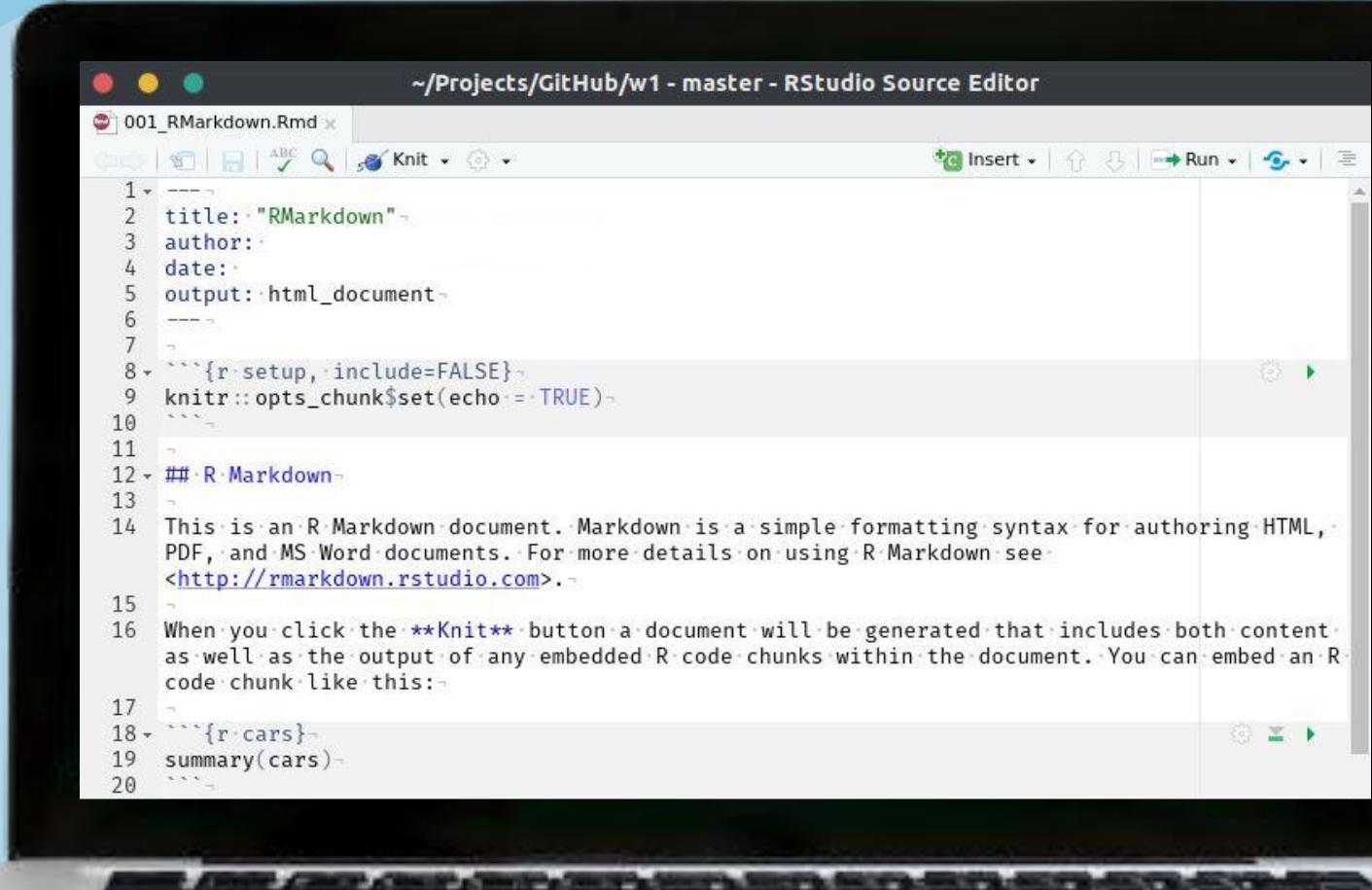
TEXT. CODE. OUTPUT.
(GET IT TOGETHER, PEOPLE.)



Artwork by @allison_horst

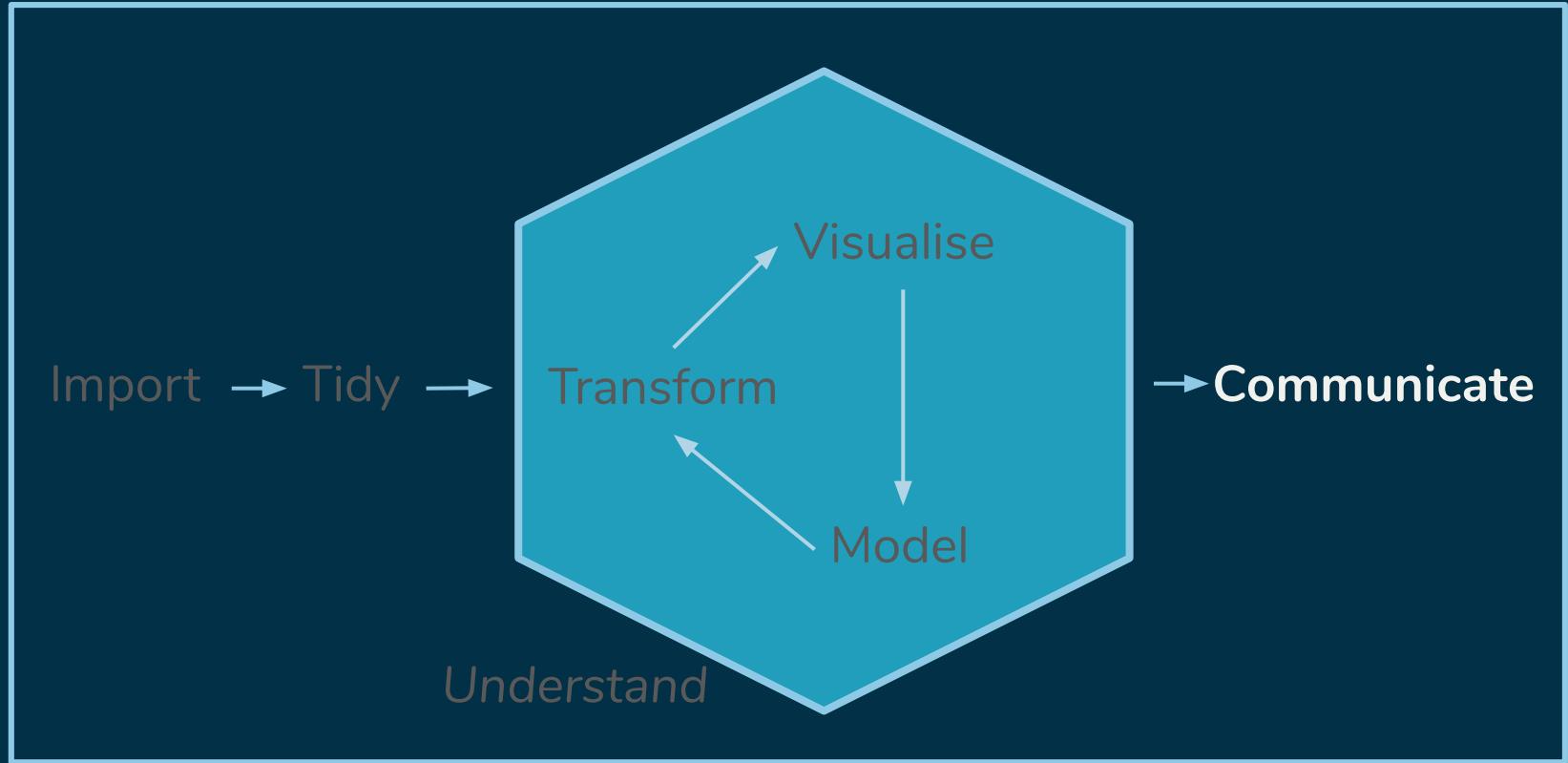
R Markdown

A document format for authoring data science project



The screenshot shows the RStudio Source Editor interface. The title bar reads " ~/Projects/GitHub/w1 - master - RStudio Source Editor". The main window displays an R Markdown file named "001_RMarkdown.Rmd". The code is as follows:

```
1 ---  
2 title: "RMarkdown"  
3 author:  
4 date:  
5 output: html_document  
---  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10  
11  
12 ## R Markdown  
13  
14 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>.  
15  
16 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:  
17  
18 ```{r cars}  
19 summary(cars)  
20```
```



Program

- Use script (R Markdown or R Script), try to avoid console
- Use Projects, not setwd(...) in script

- Learn the handy shortcuts
- Do not save and load .Rdata
- Use version control system: git!

Reading: happygitwithr.com



Download: git-scm.com

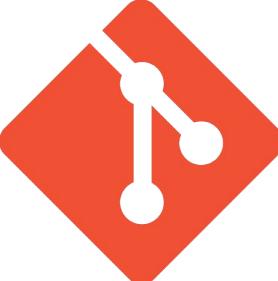


With great codes,
comes great bugs!
- (not) Uncle Ben

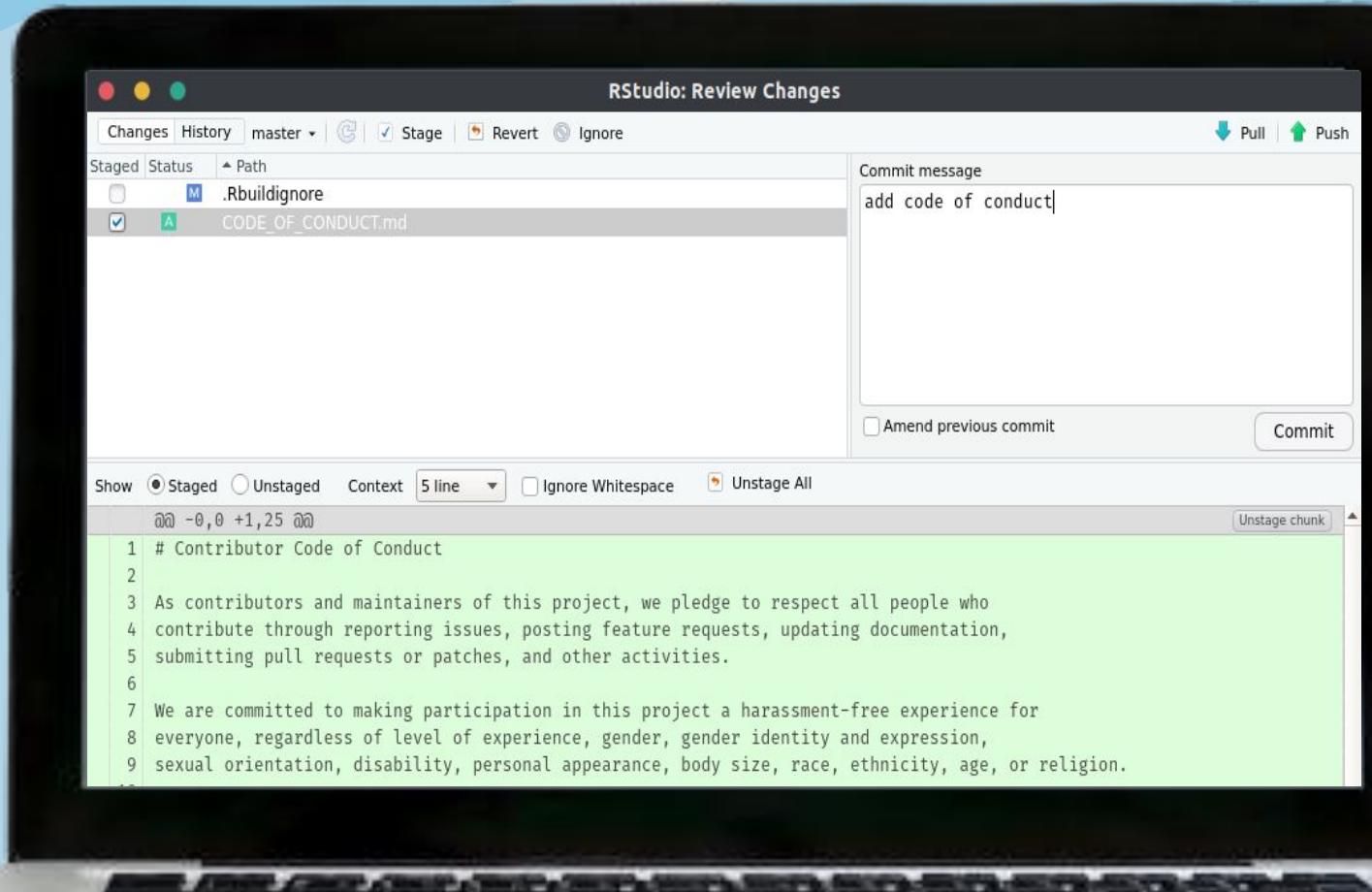
Store and share! Why sharing your work? Motivation [here](#).

- git clone <https://github.com/user/repo>
- Do some works!
- git add file.R or git add .
- git commit -m “what you have done”
- git push origin master
- Repeat: work, git add, git commit, git push



- 
- The Git logo consists of the word "git" in a large, dark brown, sans-serif font. To the left of the "g", there is a red diamond shape containing a white icon of two nodes connected by a line, representing a commit graph.
- git init
 - git remote add origin <https://github.com/user/repo>
 - Do some works!
 - git add file.R or git add .
 - git commit -m “what you have done”
 - git push -u origin master #use -u only once
 - Repeat: work, git add, git commit, git push

It is available in RStudio!



Let's get them!

- . Download R at
cran.r-project.org
- . Download RStudio at
rstudio.com
- . Download git
git-scm.com

How to set?

- Create account at github.com
- In Rstudio, Tools – Global Options – Git/SVN – Browse git executable:
 - Windows:
 - **C:/Program Files/Git/bin/git.exe**
 - UNIX/UNIX-like:
 - **/usr/bin/git**

How to set?

- Open git bash/shell/terminal
- Run:

```
git config --global user.mail  
"email@domain.com"
```

```
git config --global user.name  
"Your Name"
```

github.com/DS-ifupnyk/praktikum2020.git

Let's get started!

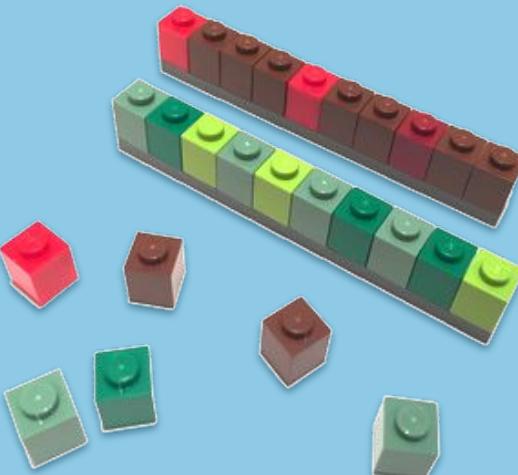
- Hit ‘Fork’ button
- Click ‘Clone or Download’, copy the URL
- In RStudio, File – New Project – Version Control – Git. Paste URL

How to use git?

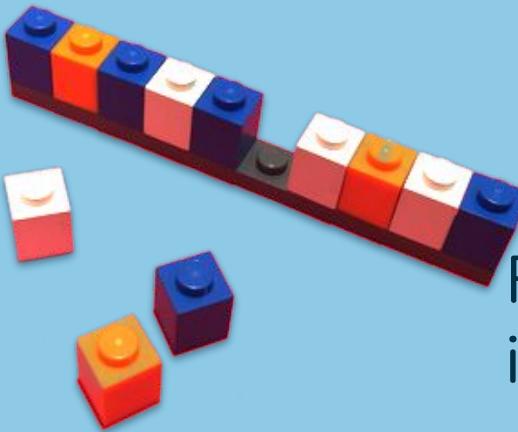
- In Environment Pane, hit ‘Git’ tab
- Click commit, a window pane will appear
- Select all files (Ctrl + A), click ‘Stage’
- Fill commit message, then click ‘Commit’
- Hit ‘Push’ Button, done!
- You may check your GitHub now!

Data structure

Dimension	Homogenous	Heterogenous
1d	Atomic vectors	List
2d	Matrix	Dataframe
nd	Array	



Integer, Double, Character

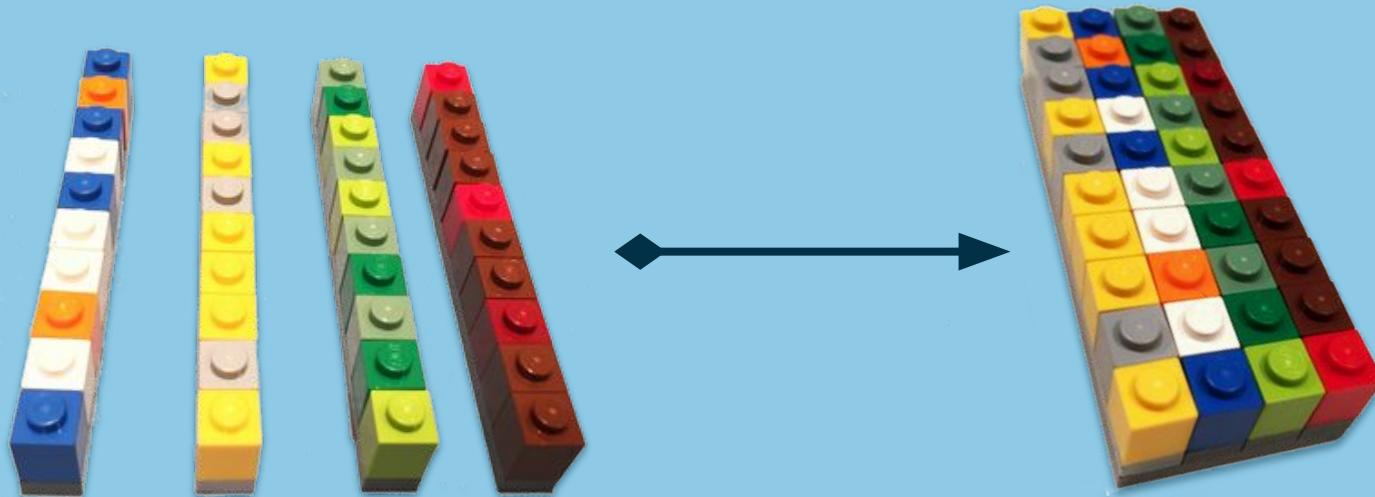


Factor (basically
integer with class)



Logical

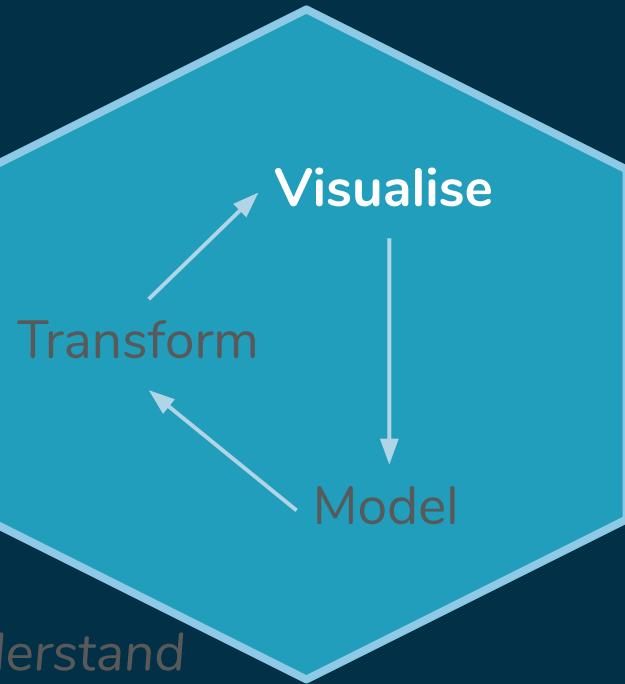




Vectors of same length

Dataframe

Import → Tidy



→ Communicate

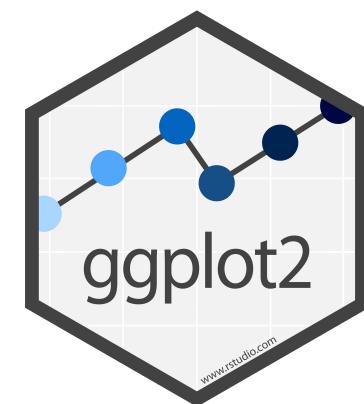
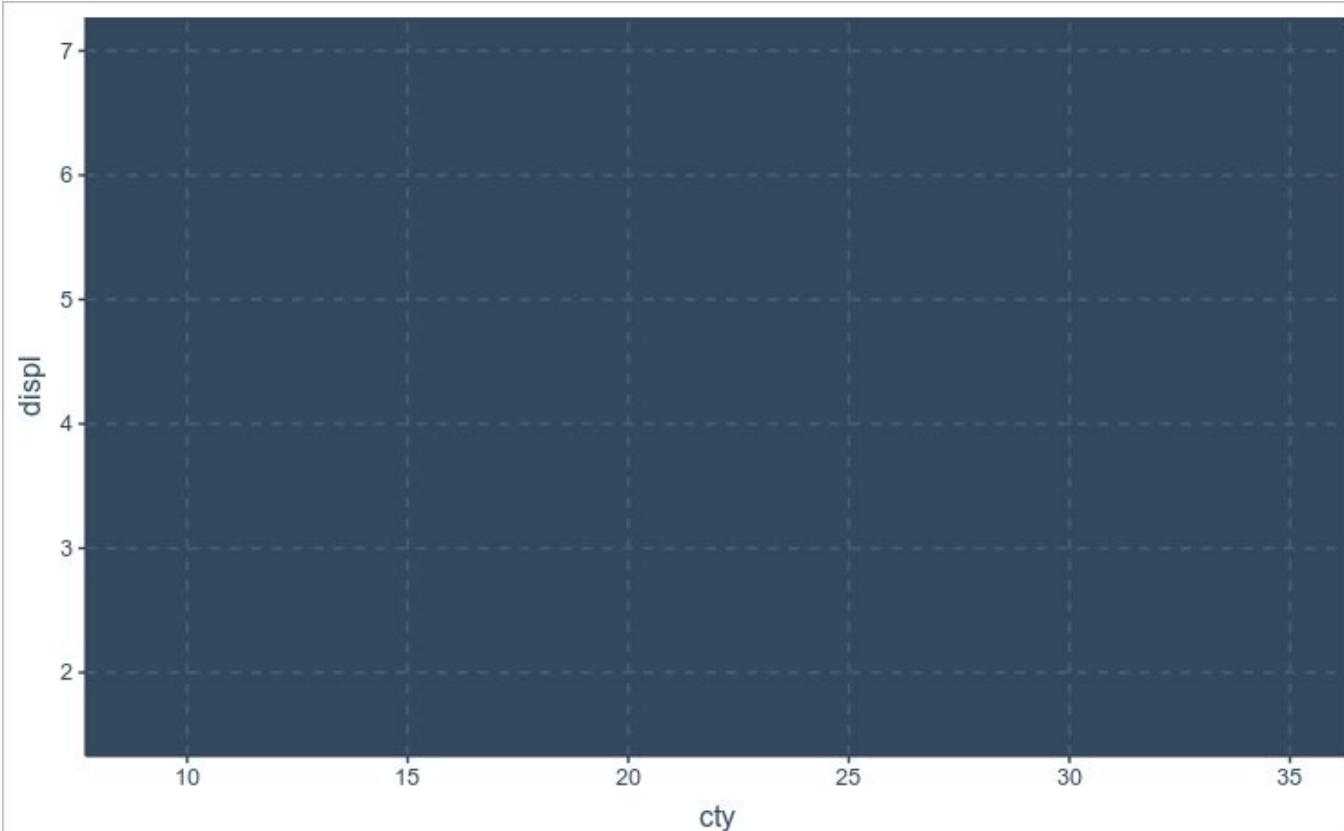
Program

ggplot2: VISUAL DATA EXPLORATION

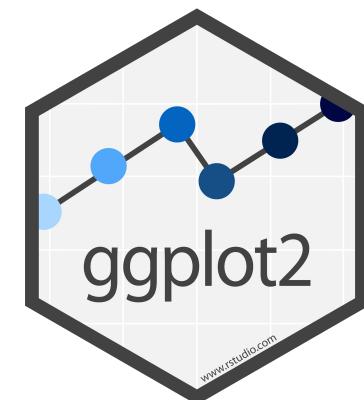
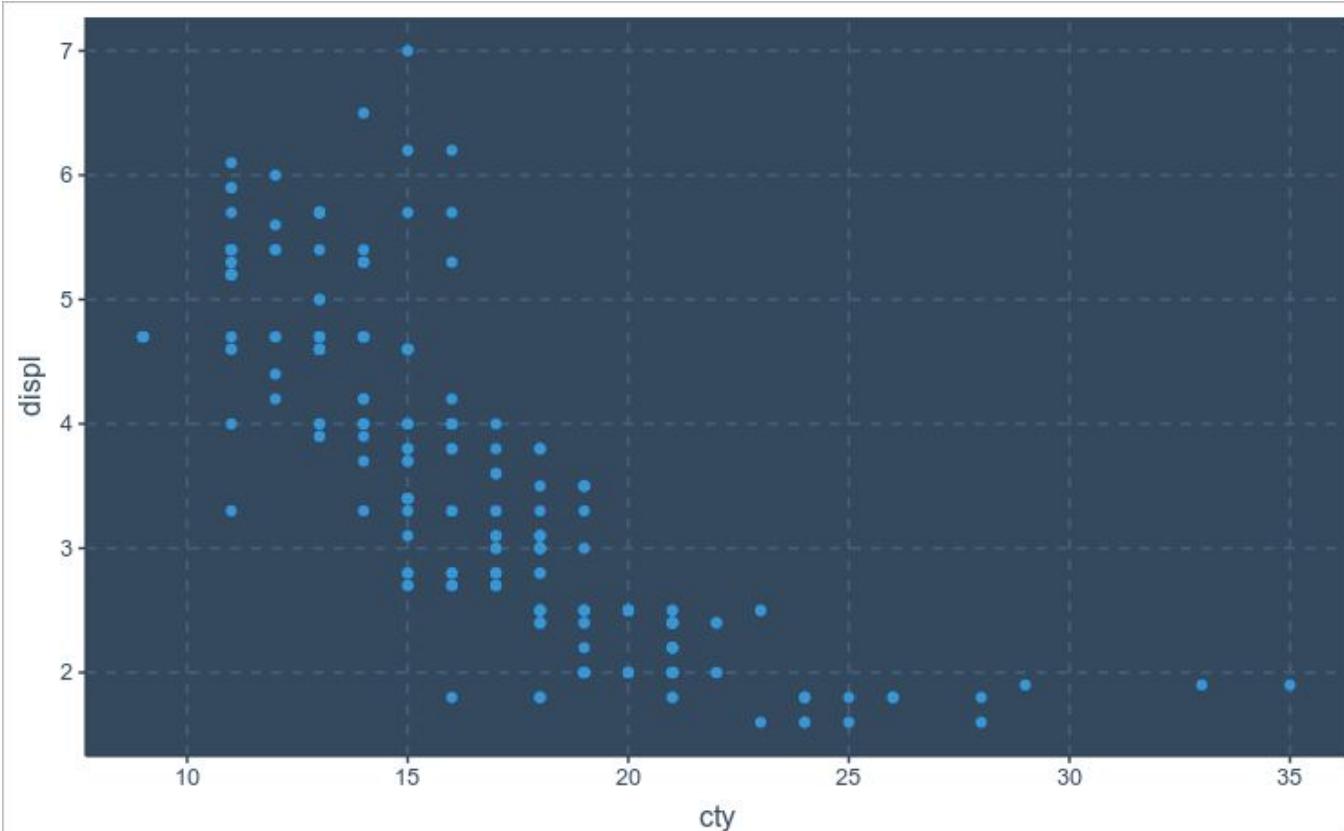


Horst '18

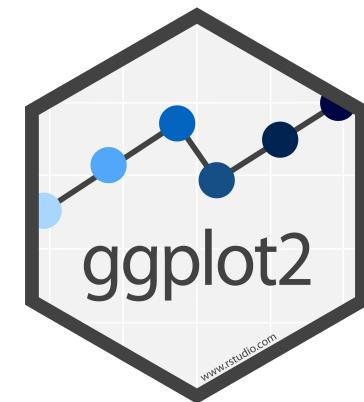
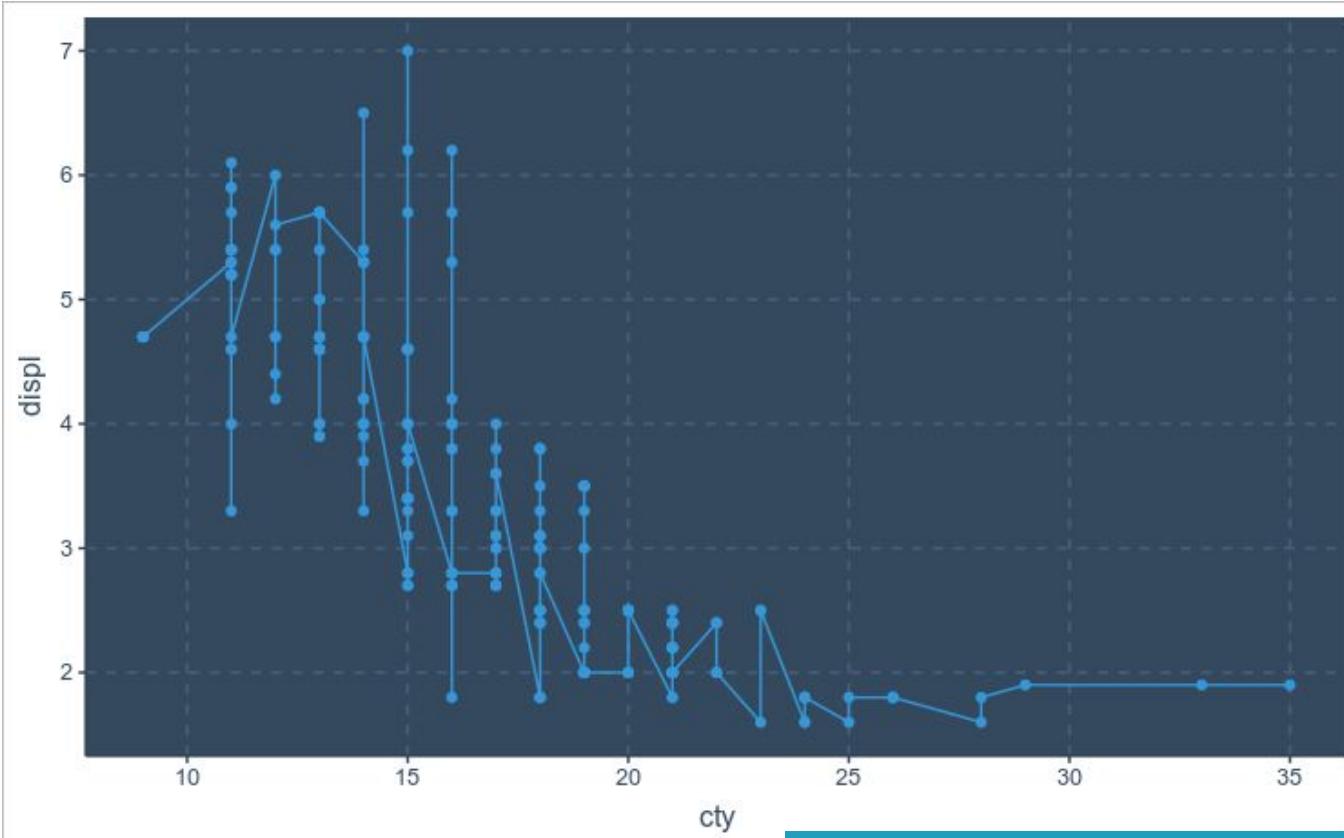
Artwork by @allison_horst



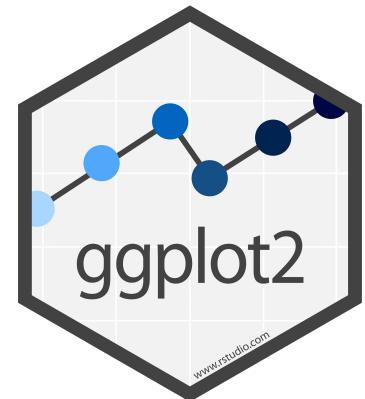
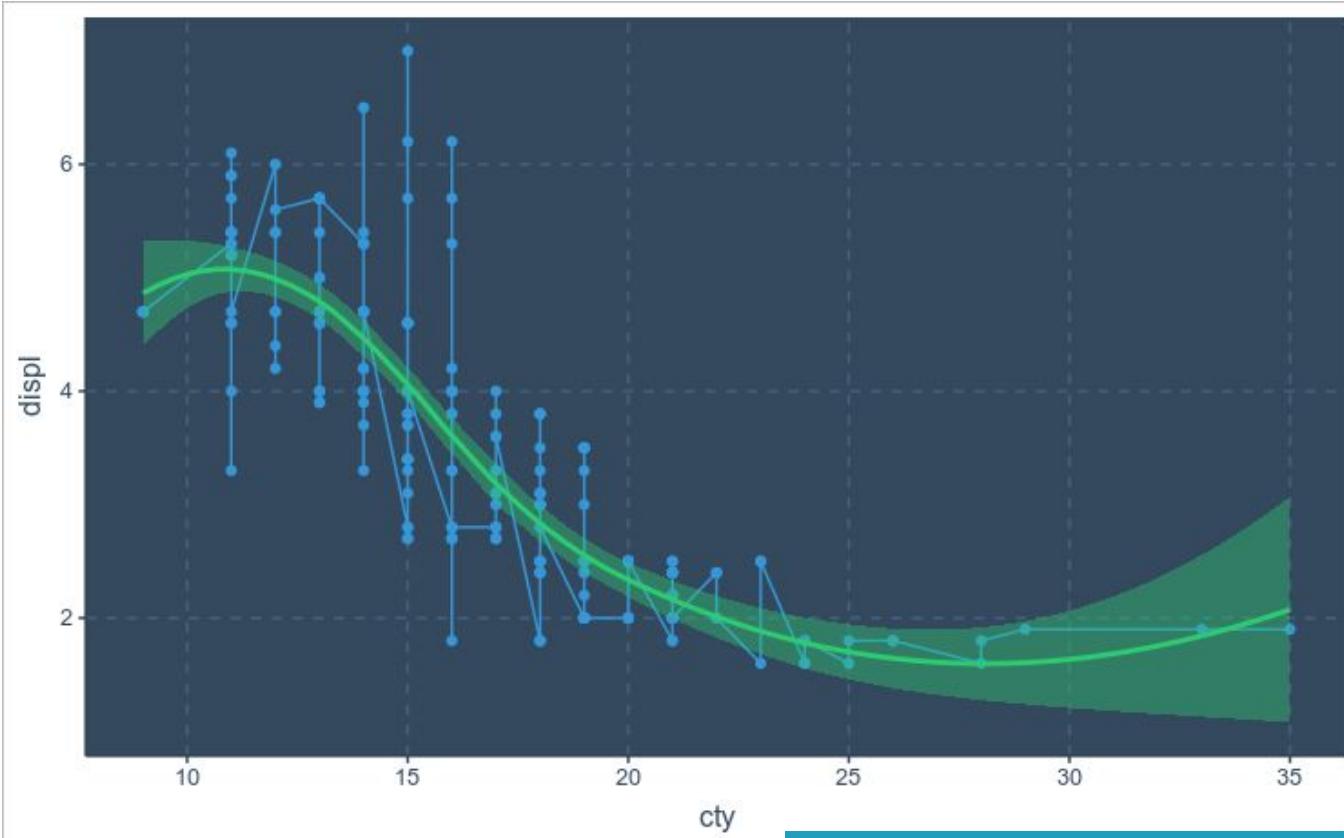
ggplot(mpg, aes(x = cty, y = displ))



```
ggplot(mpg, aes(x = cty, y = displ)) +  
  geom_point()
```



```
ggplot(mpg, aes(x = cty, y = displ)) +  
  geom_point() +  
  geom_line()
```



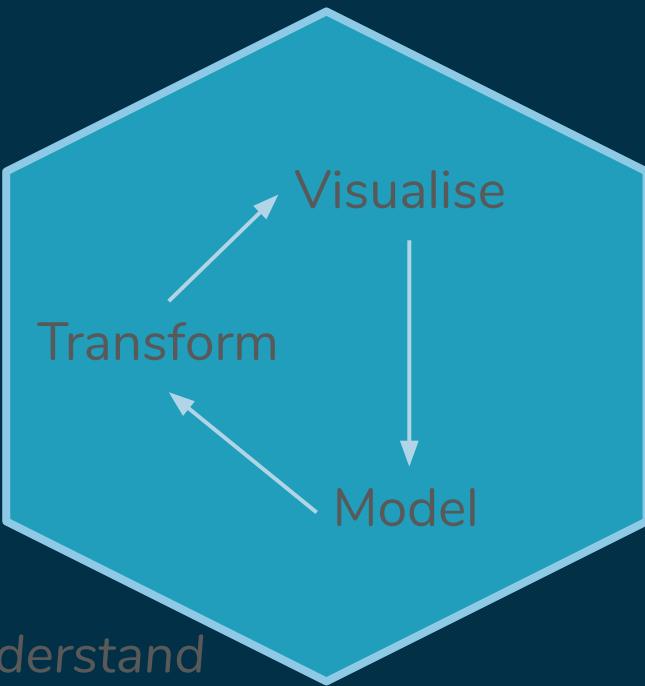
```
ggplot(mpg, aes(x = cty, y = displ)) +  
  geom_point() +  
  geom_line() +  
  geom_smooth()
```

Let's do practice!

Solution: bit.ly/modul8RMD

- Open ‘modul8.Rmd’
- Do not forget to push your works into GitHub!

Import → **Tidy**



→ Communicate

Program

Tidy datasets are all alike,
but every messy dataset is
messy in its own way!

- Hadley Wickham

A Tidy dataset

	Name	Gender	Age
1	Jane	Male	44
2	Lisbon	Female	38
3	John	Male	NA

A variable has its own column

	Var. 1	Var. 2	Var. 3
Obs. 1	A	B	C
Obs. 2	D	E	F
Obs. 3	G	H	I

There
Each
Each
Each

An observation has its own row

	Var. 1	Var. 2	Var. 3
Obs. 1	A	B	C
Obs. 2	D	E	F
Obs. 3	G	H	I

A value has its own cell

	Var. 1	Var. 2	Var. 3
Obs. 1	A	B	C
Obs. 2	D	E	F
Obs. 3	G	H	I

There
Each
Each
Each

dplyr : go wrangling



horst '18

Artwork by @allison_horst

dplyr basic functions:

- `filter()` selects rows based on their values
- `mutate()` creates new variables
- `select()` picks columns by name
- `summarise()` calculates summary statistics
- `arrange()` sorts the rows

Operators:

- `!` (not)
- `|` (or)
- `&` (and)
- `==, !=`
- `<, <=, >, >=`
- `%in%`
- `is.na()`

```
function(arg1, arg2, arg3, ...)
```

```
arg1 %>%  
  function(arg2, arg3, ...)
```

```
function(arg1, arg2, arg3, ...)
```

```
arg2 %>%  
  function(arg1, arg2=., arg3,  
          ...)
```

magrittr



How can I
chain?

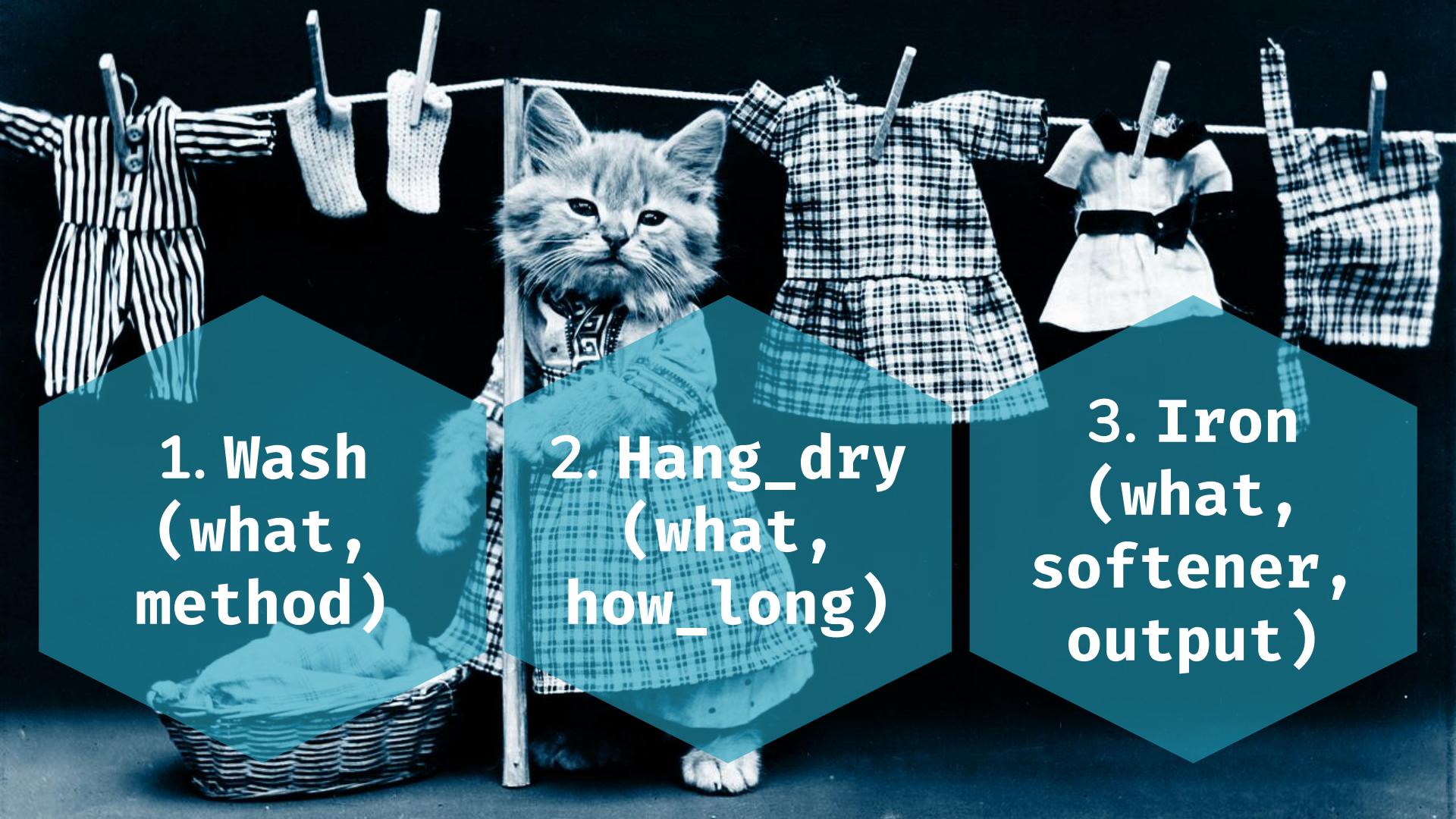




1. Wash

2. Hang_dry

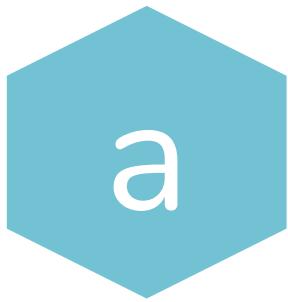
3. Iron



**1. Wash
(what,
method)**

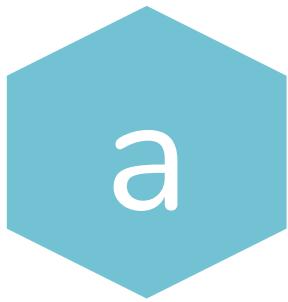
**2. Hang_dry
(what,
how_long)**

**3. Iron
(what,
softener,
output)**



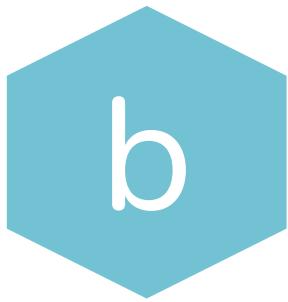
a

```
> washed_clothes <- wash(what = dirty_clothes,  
                           method = "handwash")  
> dry_clothes <- hang_dry(what =  
                           washed_clothes,  
                           how_long = 8)  
> iron(what = dry_clothes, softener = TRUE,  
        output = "neat.clothes")
```



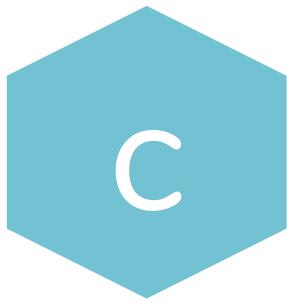
a

```
> washed_clothes <- wash(what = dirty_clothes,  
                           method = "handwash")  
> dry_clothes <- hang_dry(what =  
                           washed_clothes,  
                           how_long = 8)  
> iron(what = dry_clothes, softener = TRUE,  
        output = "neat.clothes")
```



b

```
> iron(  
  hang_dry(  
    wash(what = dirty_clothes, method =  
          "handwash"),  
    how_long = 8),  
  softener = TRUE,  
  output = "neat.clothes")
```



C

```
> wash(what = dirty_clothes,  
       method = "handwash") %>%  
  hang_dry(how_long = 8) %>%  
  iron(softener = TRUE,  
        output = "neat.clothes")
```

```
x <- something  
for (i in seq_along(x) {  
  function(x[[i]])  
}
```

```
x <- something  
*apply(x, function)
```

```
x <- something  
map_*(x, function)
```

purrr



Import → Tidy →

Understand

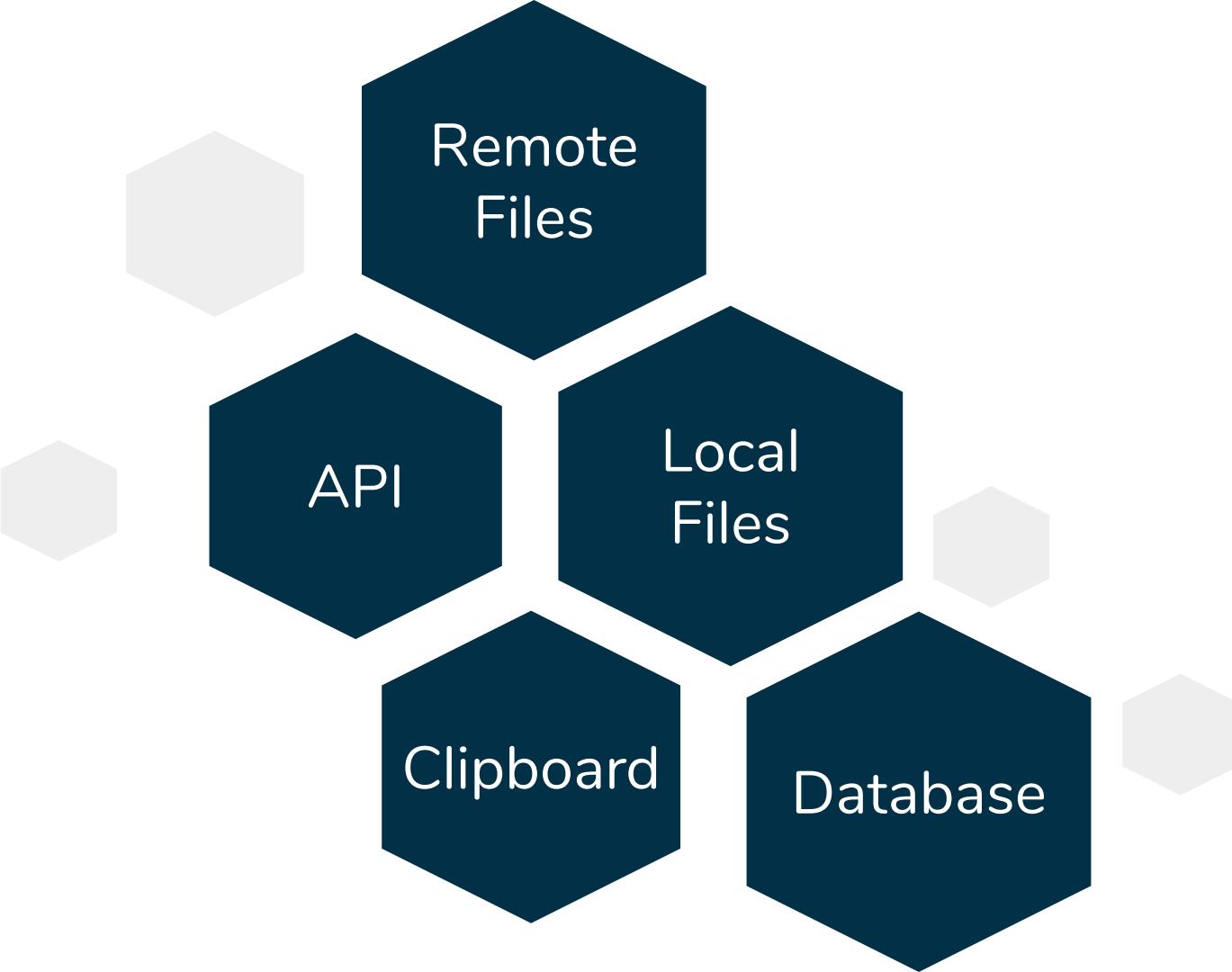
Transform

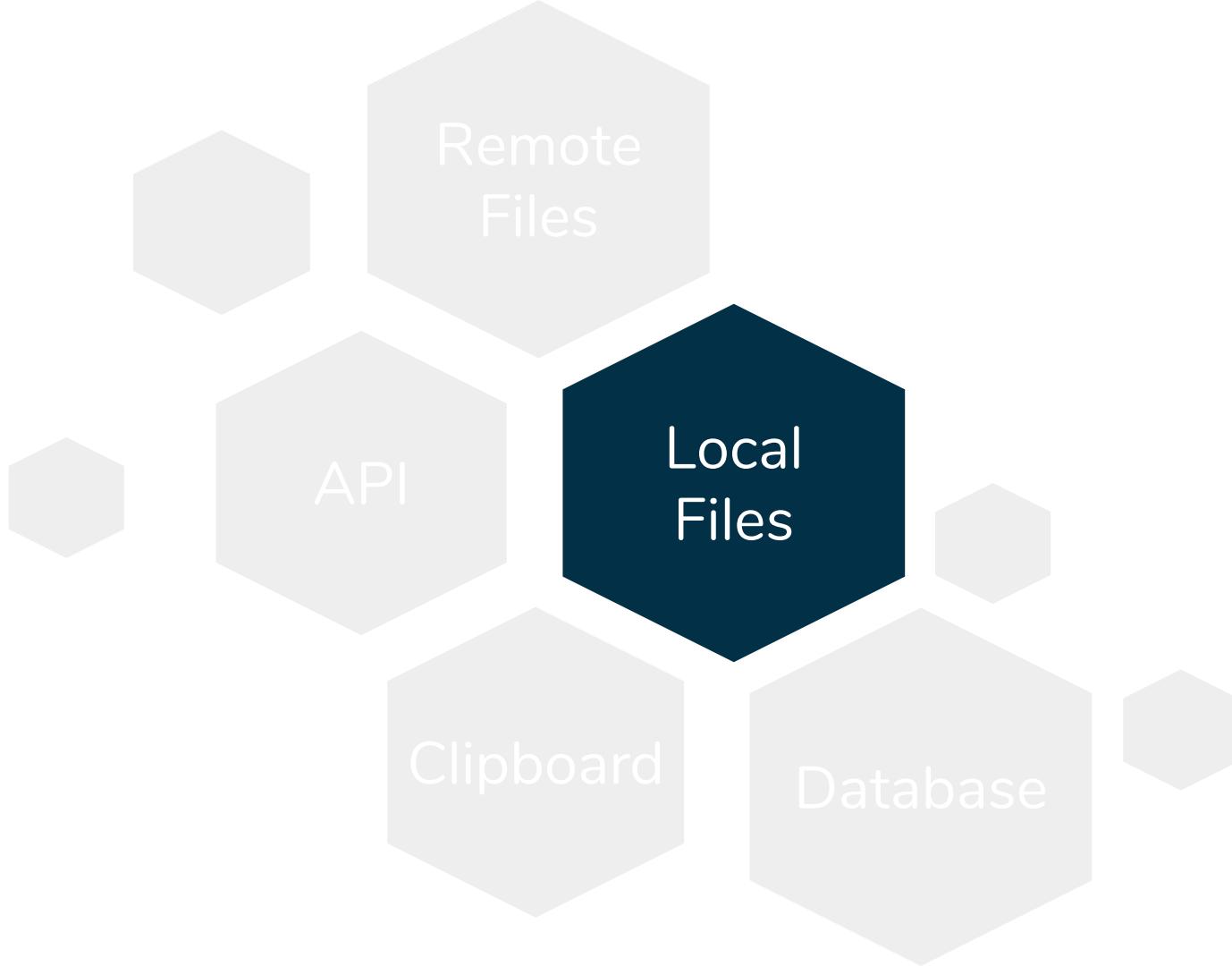
Visualise

Model

→ Communicate

Program





- `vroom()`: comma separated (CSV) files
- `vroom()`: tab separated files
- `vroom()`: general delimited files
- `vroom()`: fixed width files
- `vroom()`: tabular files where columns are separated by white-space.

vroom



Let's do practice!

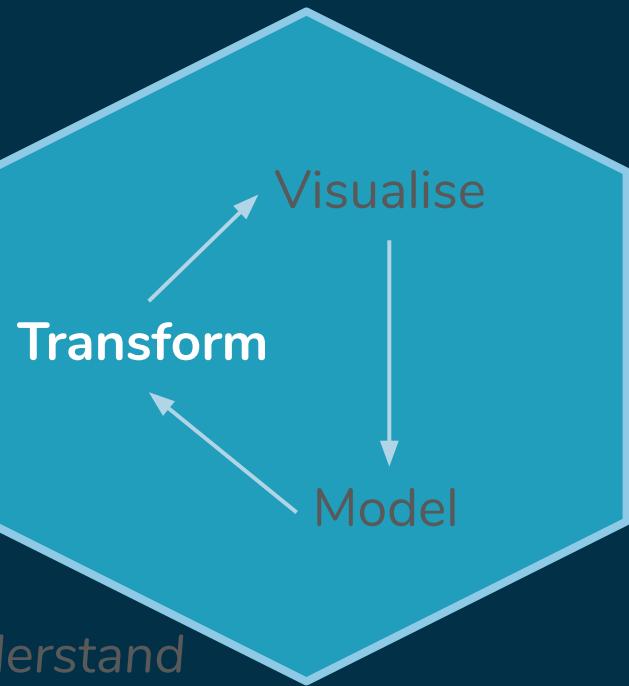
- Open ‘modul7.Rmd’
- Do not forget to push your works into GitHub!

Let's do practice!

Solution: Modul Praktikum Bab 7

- Open ‘modul7.Rmd’
- Do not forget to push your works into GitHub!

Import → Tidy



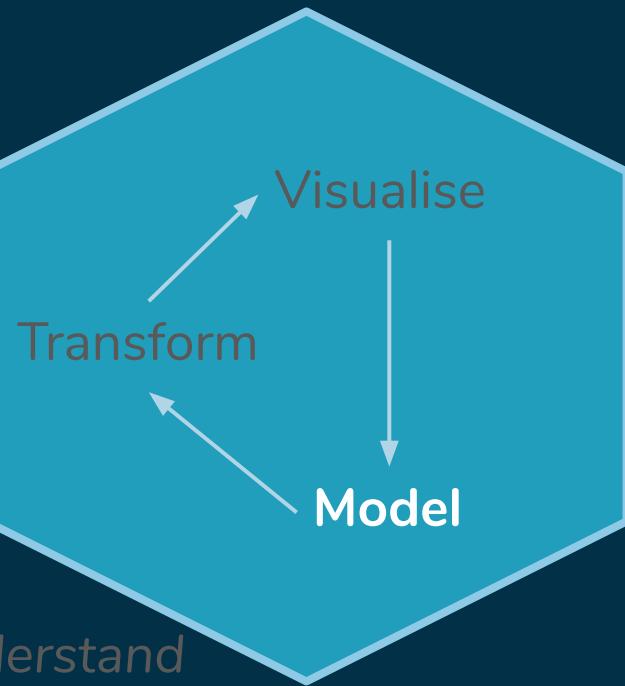
→ Communicate

Program



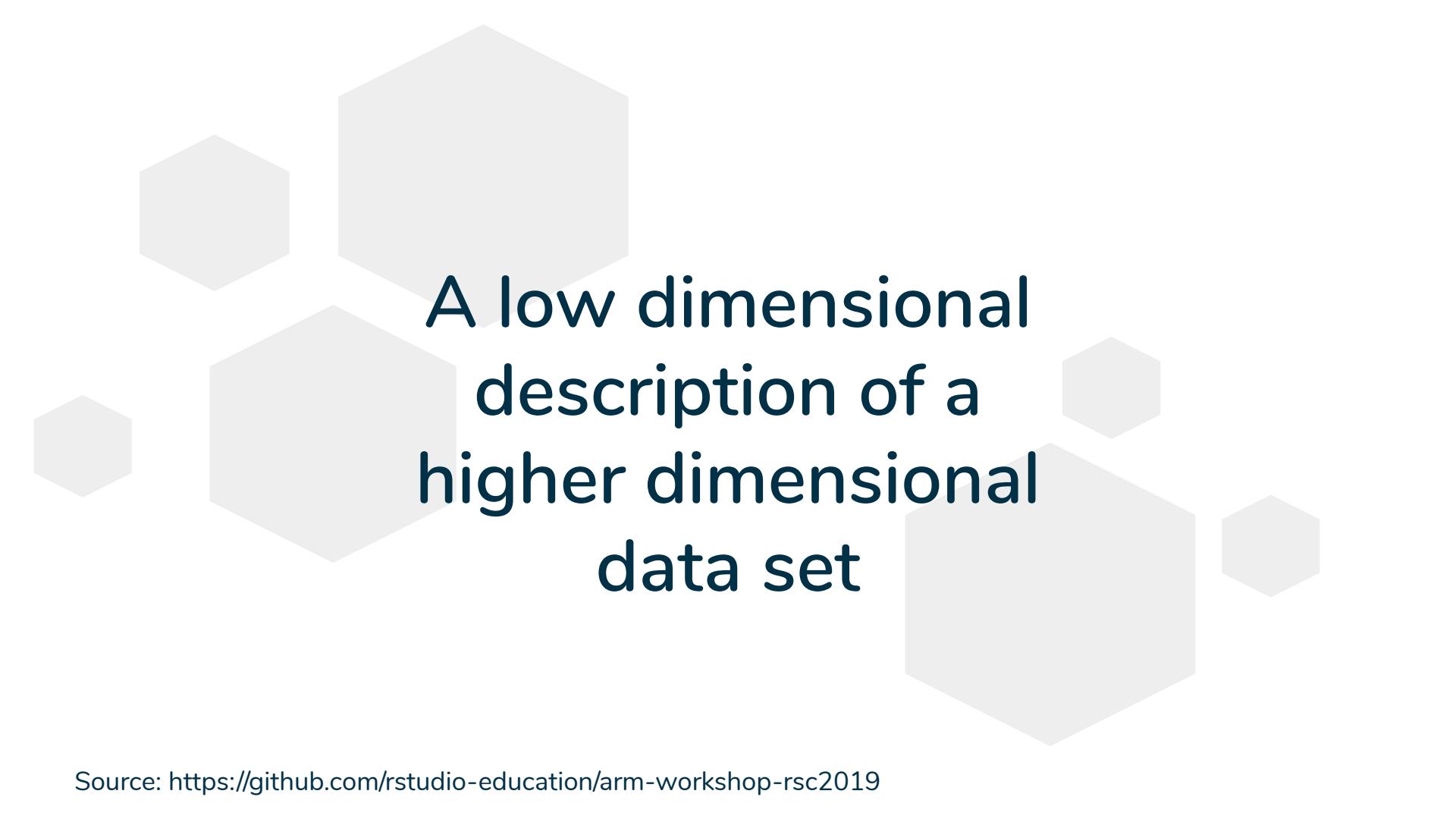
Artwork by @allison_horst

Import → Tidy

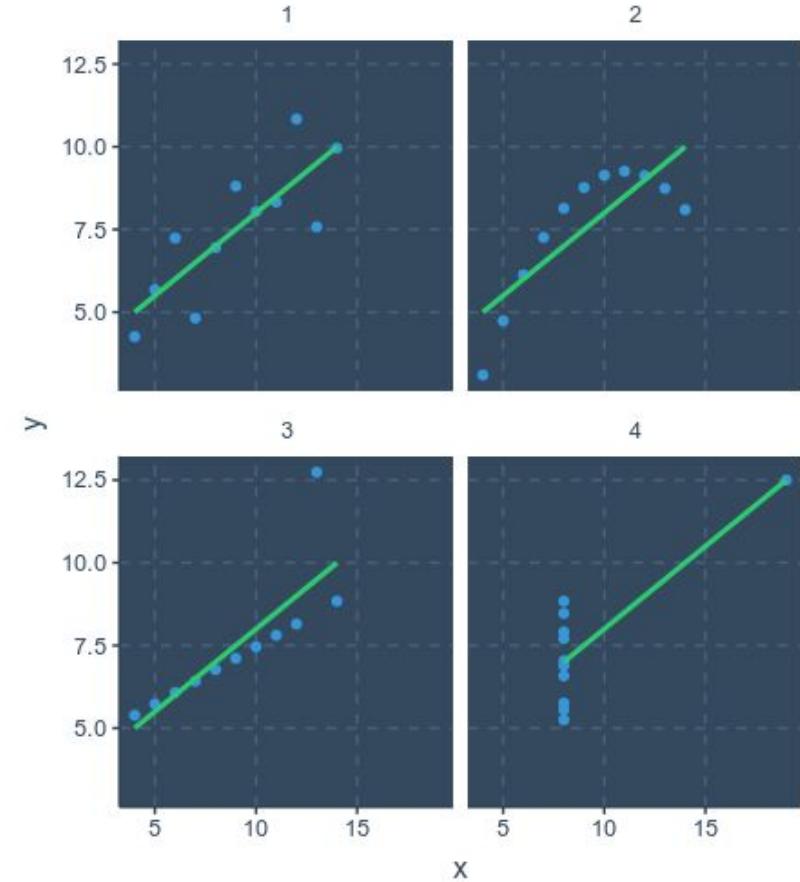


→ Communicate

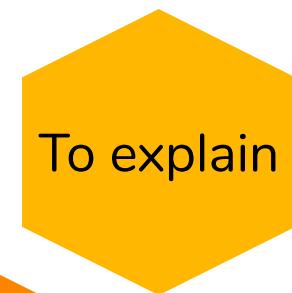
Program



A low dimensional
description of a
higher dimensional
data set



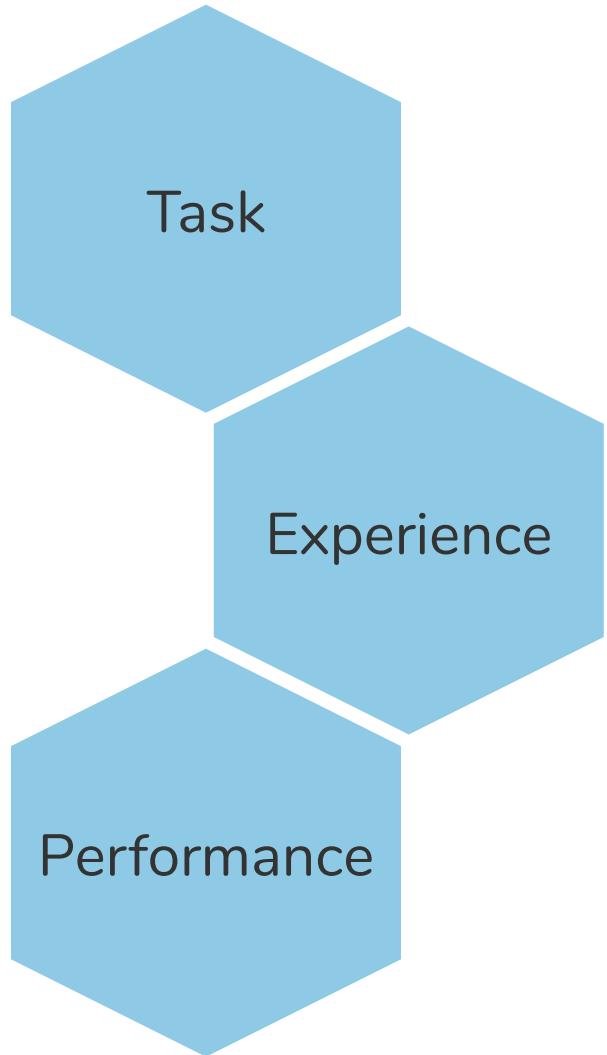
Outcome ~ Predictor/Explanatory



- All models are wrong, but some are useful – George Box

Machine Learning

$$y = f(x) + e$$



Supervised learning

Unsupervised learning

Springer Texts in Statistics

Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

An Introduction
to Statistical
Learning

with Applications in R

 Springer

Pre-process

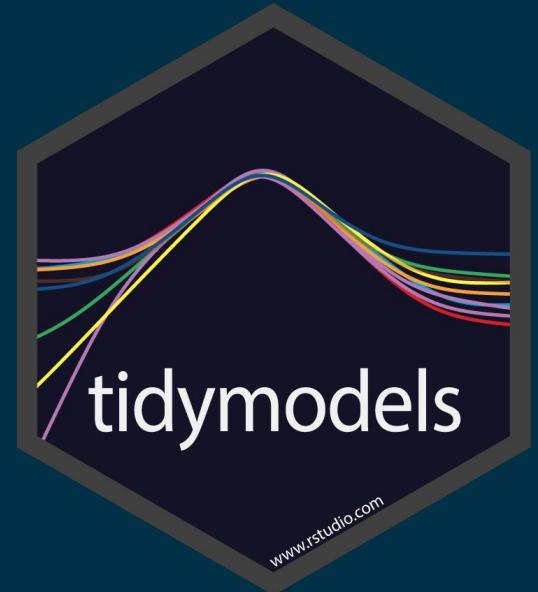


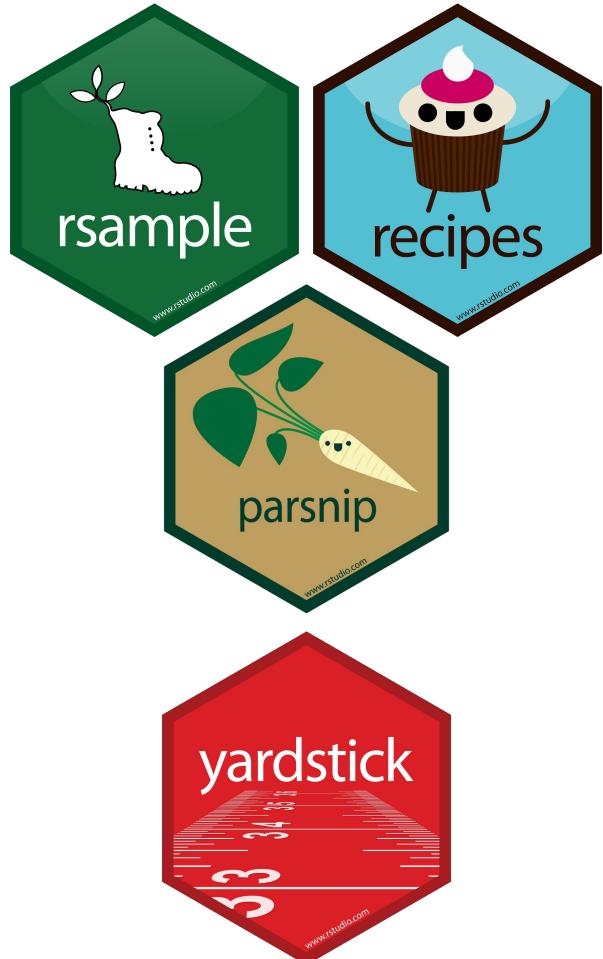
Train



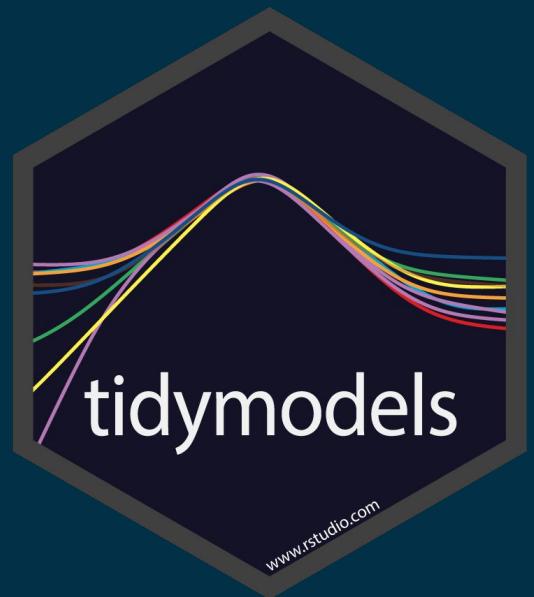
Validate

tidymodels





tidymodels



Let's do practice!

- Open ‘Supervised Learning - Linear Regression.Rmd’
- Do not forget to push your works into GitHub!

Let's do practice!

Solution: bit.ly/supervised-Solution

- Open ‘Supervised Learning - Linear Regression.Rmd’
- Do not forget to push your works into GitHub!

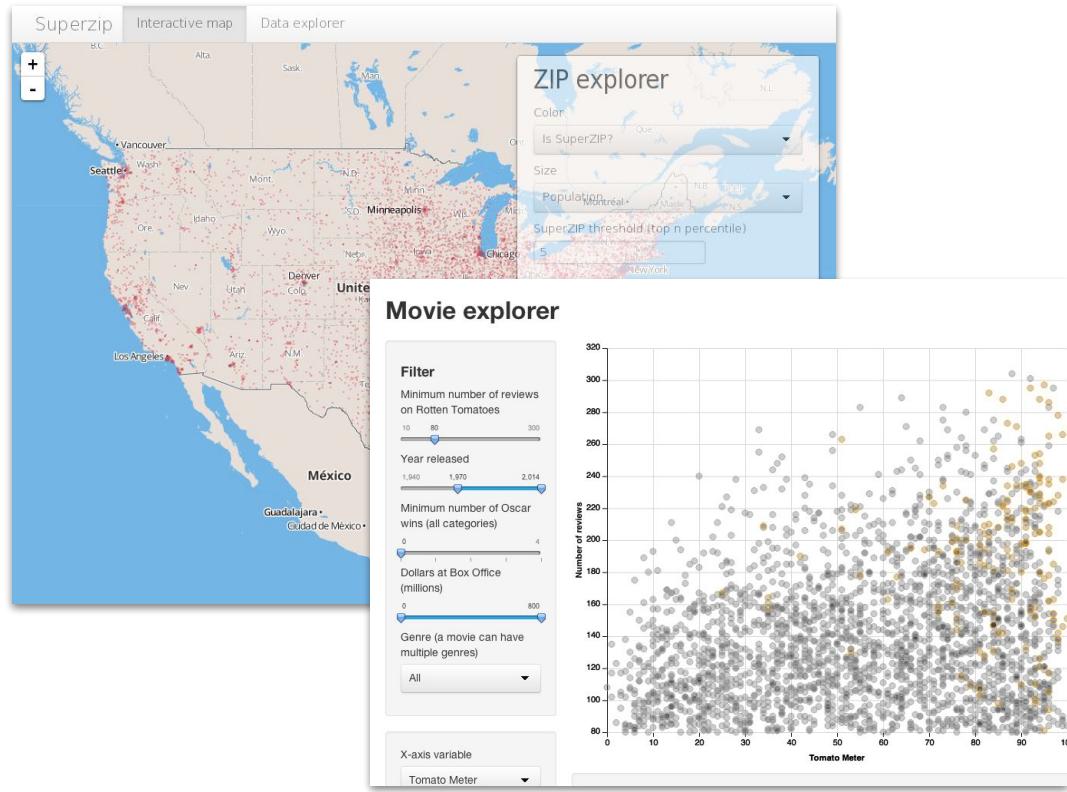
Let's do practice!

- Open ‘Unsupervised Learning.Rmd’
- Do not forget to push your works into GitHub!

Let's do practice!

- Open ‘Unsupervised Learning.Rmd’
- Do not forget to push your works into GitHub!

Solution: bit.ly/unsupervisedlearning-Solution



Images from shiny.rstudio.com



```
library(shiny)
mydata <- load("mydata.rda")

ui <- fluidPage(
  # define user interface here
)

server <- function(input, output) {
  # put logic and computation here
}

shinyApp(ui = ui, server = server)
```

Important terms!

Input

Output

Reactive
values

```
library(shiny)
mydata <- load("mydata.rda")

ui <- fluidPage(
  # define user interface here
)

server <- function(input, output) {
  # put logic and computation here
}

shinyApp(ui = ui, server = server)
```

```
library(shiny)  
mydata <- load("mydata.rda")
```

- Inputs are defined and laid out
- Outputs are laid out

```
ui <- fluidPage(  
  # define user interface here  
)
```

- Inputs are taken
- Outputs are calculated/made

```
server <- function(input, output) {  
  # put logic and computation here  
}
```

```
shinyApp(ui = ui, server = server)
```

1. Take input value from `input$xx`
2. Perform processing/calculation in reactive context (expression/observer)
3. Build objects to display using `render*()`
4. Save objects to display to `output$xx`

Let's do practice!

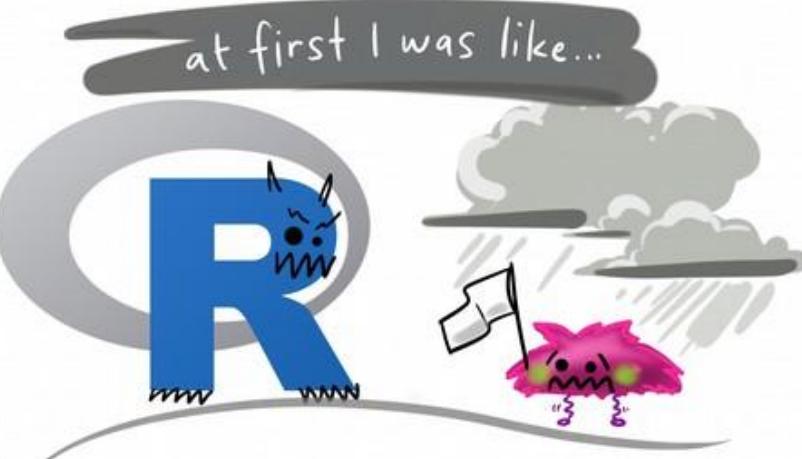
- Open ‘shiny.Rmd’
- Do not forget to push your works into GitHub!

Let's do practice!

- Open ‘shiny.Rmd’
- Do not forget to push your works into GitHub!

Solution: bit.ly/shiny-Solution

Congrats!



...but now it's like...



Artwork by @allison_horst

Contact me

Name : Vynska Amalia Permadi

Email : vynspermadi@upnyk.ac.id