Lea*R*ning R

Cyril Statzer

All information is my personal opinion

Getting started

1. Download R
   - for either Mac or Windows: https://cran.rstudio.com/
2. Install Rstudio
   - scroll down to installers and download either the Windows or Mac version: https://www.rstudio.com/products/rstudio/download/#download
3. Open Rstudio
   a) type install.packages('tidyverse') in the bottom left panel (the console)
   b) let R install all required packages for you (can take 10 min)
   c) after the command finished type library(tidyverse).
   d) If you encounter problems / errors along the way let me know and I can help you. You're all set!

Overview

1. **Why R?**
2. **Interacting with R**
   - Rstudio
   - R basics
   - Data workflow

   $\longrightarrow$ What you need to know to get started

3. **Visualizing data in R**
   - The ggplot approach
   - Data exploration (can we beat The Economist?)

   $\longrightarrow$ Learning to speak the *grammar of graphics*

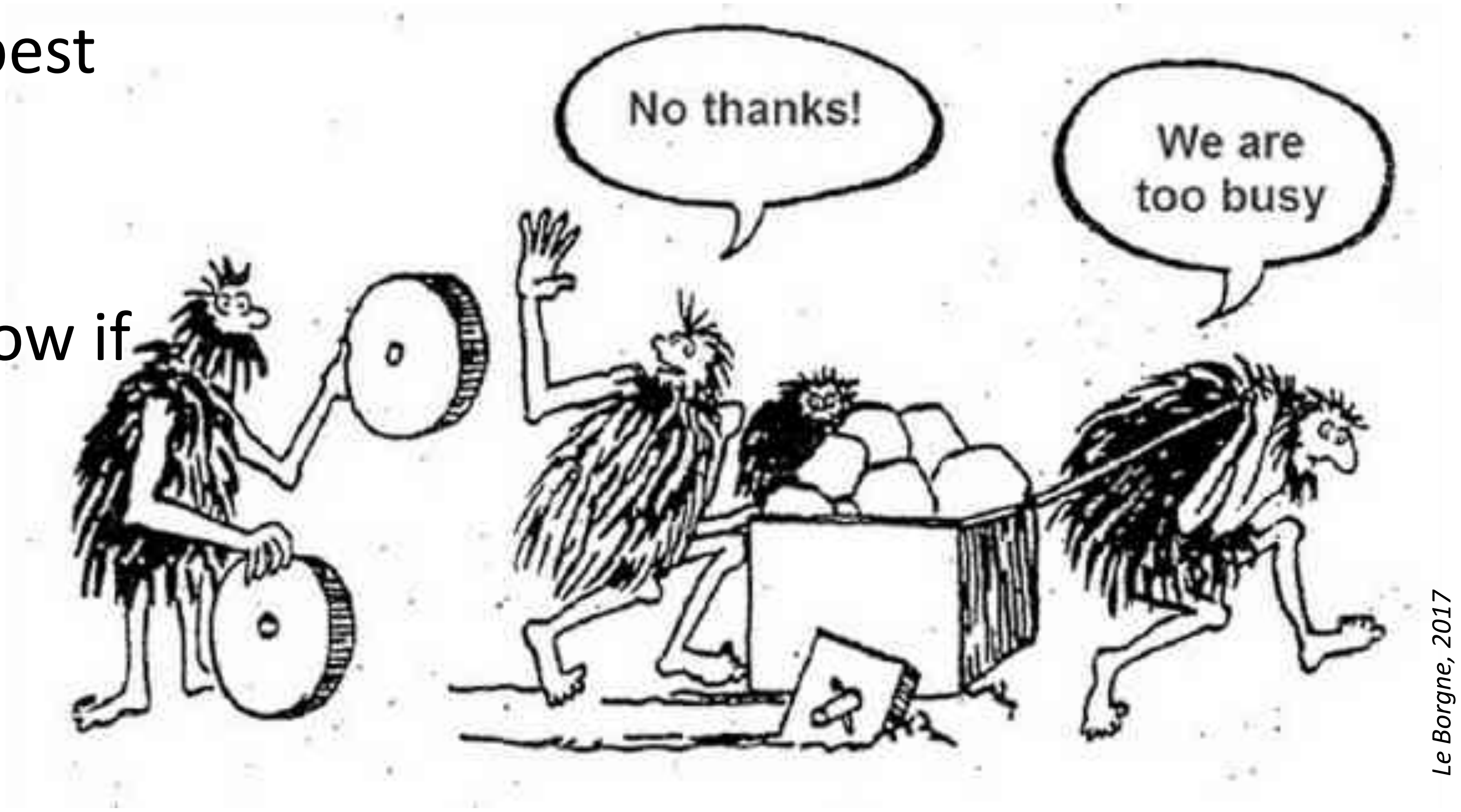
4. **How to use R in the lab (from easy to advanced)** $\longrightarrow$ Potential use cases
5. **Outlook**

Why R?

(my personal opinion)

Why I think learning R is a good idea?

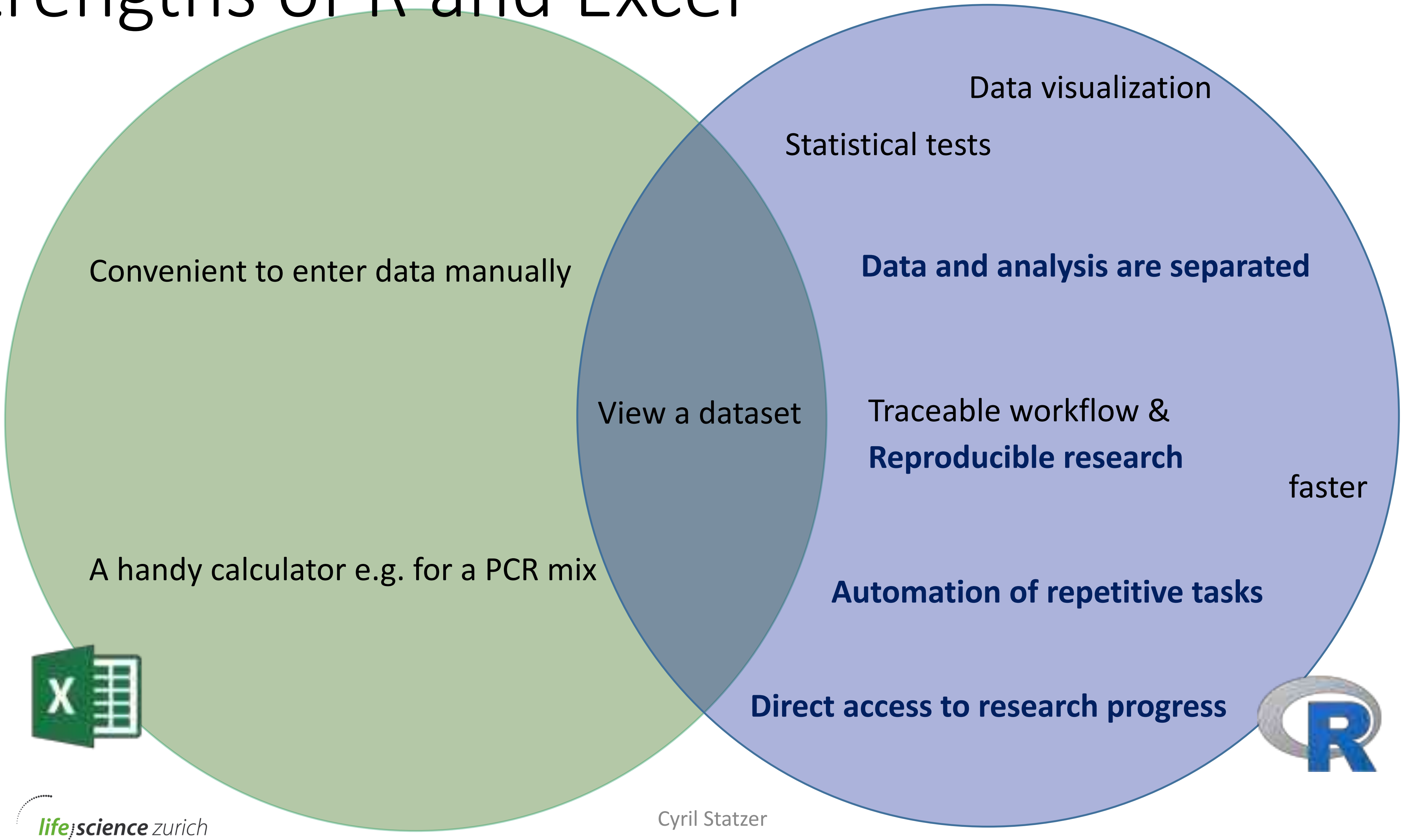
- Are you using the best available tools?
- How would you know if you were not?



Le Borgne, 2017

most of us

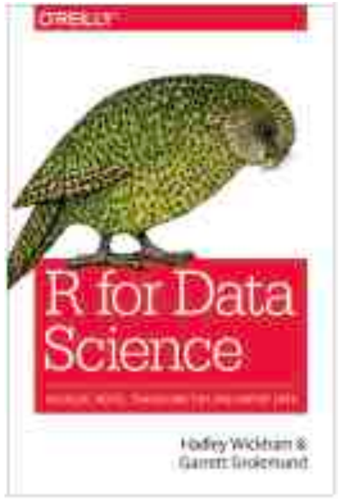
Strengths of R and Excel


Convenient to enter data manually
A handy calculator e.g. for a PCR mix
View a dataset
Data visualization
Statistical tests
Data and analysis are separated
Traceable workflow &
Reproducible research
faster
Automation of repetitive tasks
Direct access to research progress

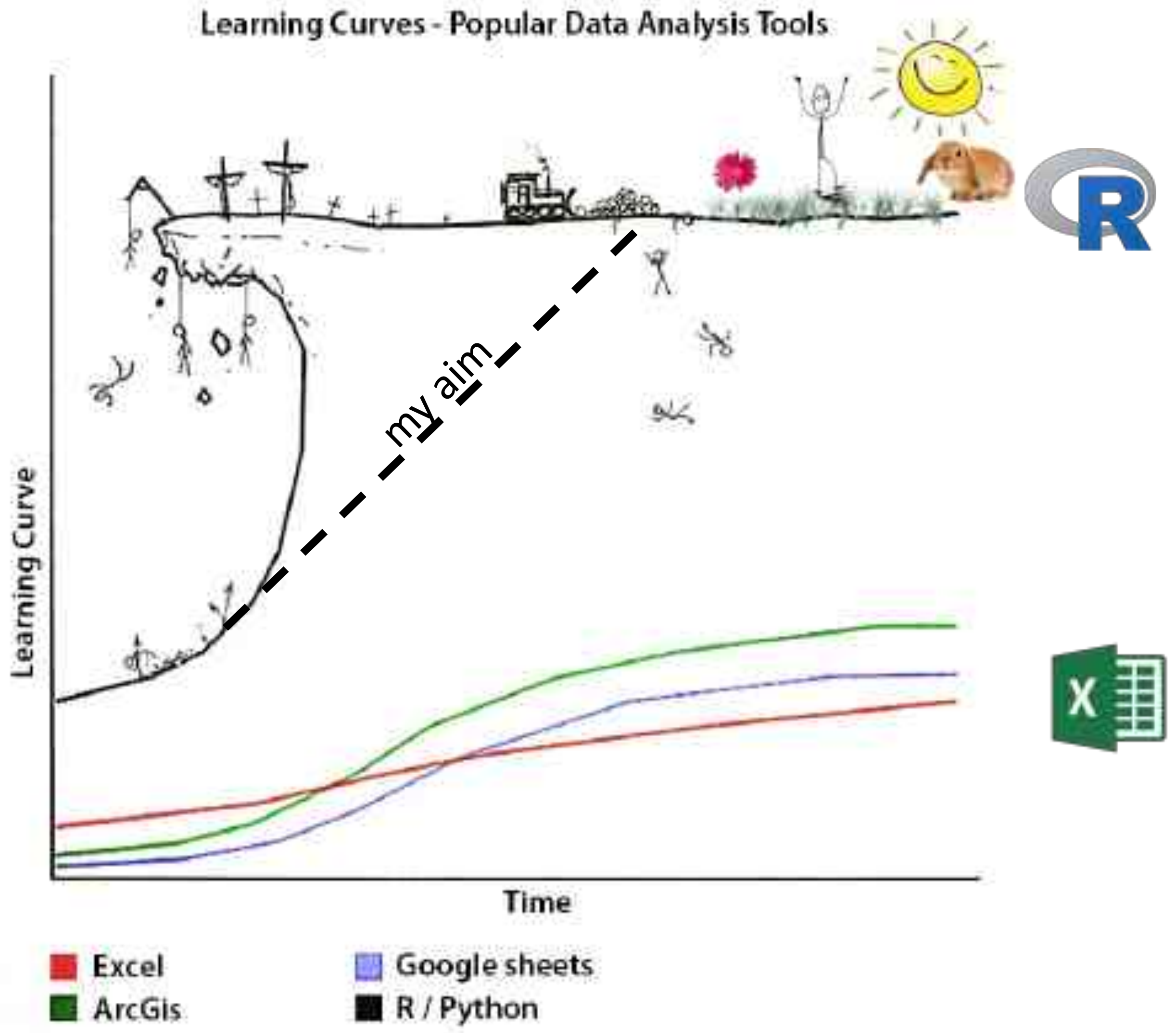
Getting started is hard

- Initial work required but it pays off greatly

- You can get help everywhere



The best book is for free: https://r4ds.had.co.nz



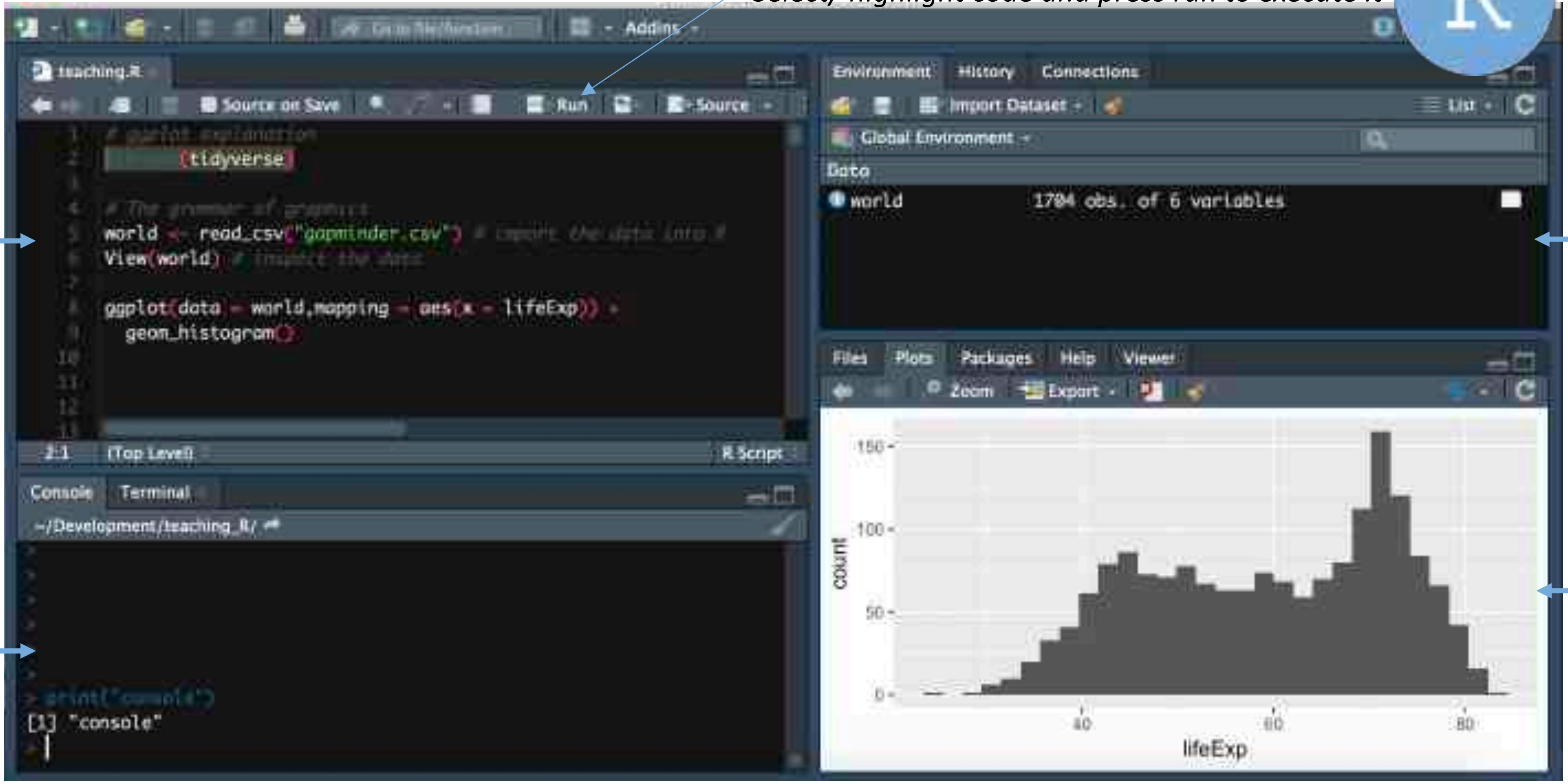
Disclaimer

- This presentation will become overwhelming towards the end
- We’ll go from learning a new language to writing a short story in one lesson
- Try to hang on and understand the overall concepts

How can we use R & RStudio?

Rstudio (1/3)

Select/ highlight code and press run to execute it



```r
# ggplot explanation
library(tidyverse)

# The grammar of graphics
world <- read_csv("gapminder.csv") # import the data into R
View(world) # inspect the data

ggplot(data = world,mapping = aes(x = lifeExp)) +
  geom_histogram()
```

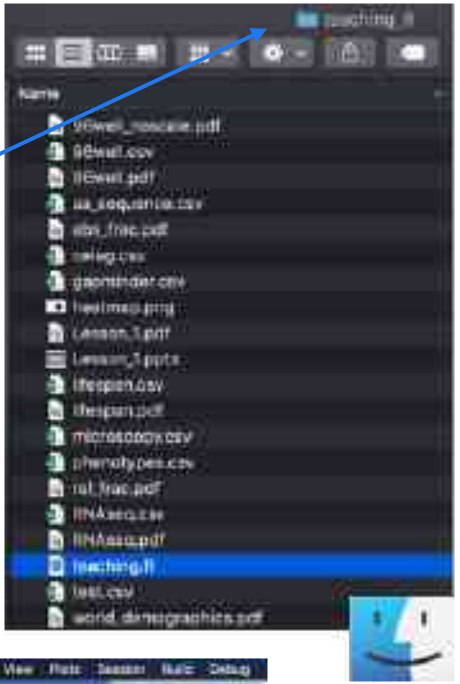
```
world          1704 obs. of 6 variables
```

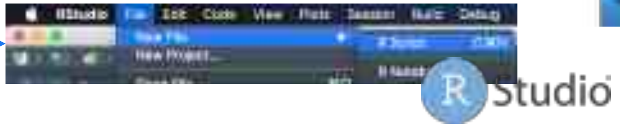
```
~/Development/teaching_R/
> print("console")
[1] "console"
>
```

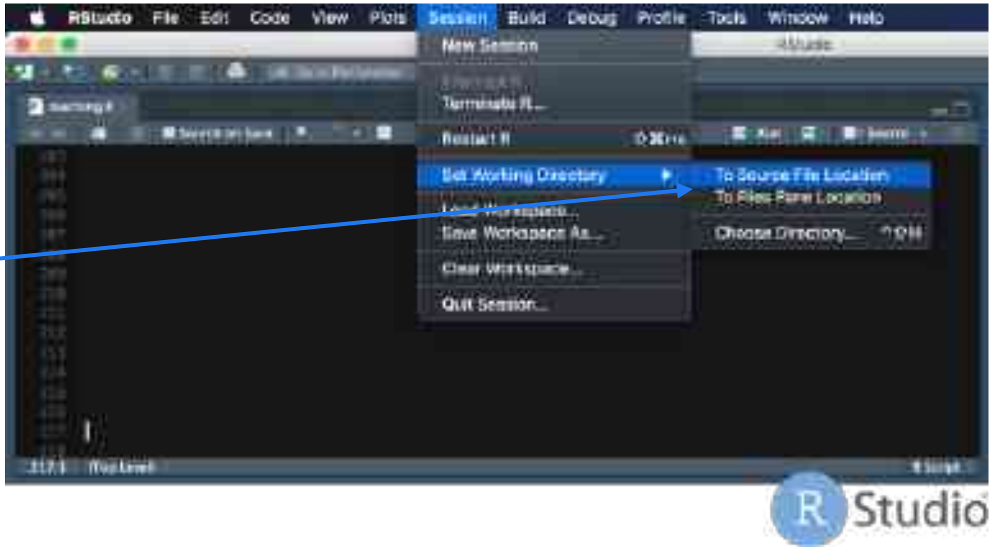
1.) new script, 2.) save script to folder, 3.) Set working directory to script location

Rstudio (2/3)

1. Make a **folder for your R project**
2. Put your **data into this folder**
3. Open Rstudio, open and then **save the R script to this folder**
4. When you start Rstudio **set the Working directory to the location of your script**

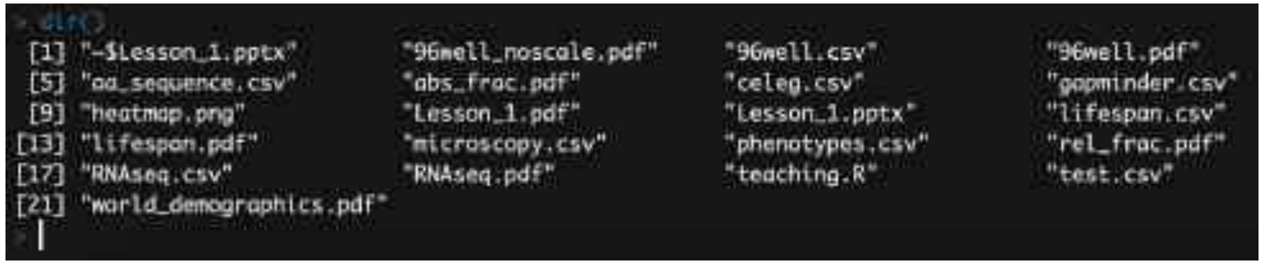






Rstudio (3/3)

1. Now R can “see” the files you want to work with:



```
> dir()
 [1] "~$Lesson_1.pptx"            "96well_noscale.pdf"   "96well.csv"       "96well.pdf"
 [5] "aa_sequence.csv"            "abs_frac.pdf"         "celeg.csv"        "gapminder.csv"
 [9] "heatmap.png"                "Lesson_1.pdf"         "Lesson_1.pptx"    "lifespan.csv"
[13] "lifespan.pdf"               "microscopy.csv"       "phenotypes.csv"   "rel_frac.pdf"
[17] "RNAseq.csv"                 "RNAseq.pdf"           "teaching.R"       "test.csv"
[21] "world_demographics.pdf"
> 
```

The **`dir()`** functions shows all the files in your folder.

2. You can directly import them by name since R knows now where to look for them

`read_csv(”gapminder.csv”)`

R basics (1/2)

Assignment is giving names to things. It is like labelling a box and then put anything you want into it. We call this "an object" in R.

- Assignment

Assignment	`Kelvin <- 273.15`	"`273.15` is saved in `Kelvin`"
	`tempsC <- c(5,20,30)`	"`tempsC` has 3 numerical entries: `5, 20` and `30`"
	`feeling <- c("cold","medium","hot")`	"`feeling` has 3 character entries: `"cold", "medium"` and `"hot)`

- Data types

The data type determines what you can do with it: 10 / 2 works but "hello" / "world" not.

Numeric:	`Kelvin` and `tempsC` are numeric	Test with: `class(Kelvin)`
Character	`feeling` is character	Test with: `class(feeling)`

R basics (2/2)

- Functions

```
mean(tempsC)                                   18.33333

max(tempsC)                                    30

length(feeling)                                3
```

- Multiple functions

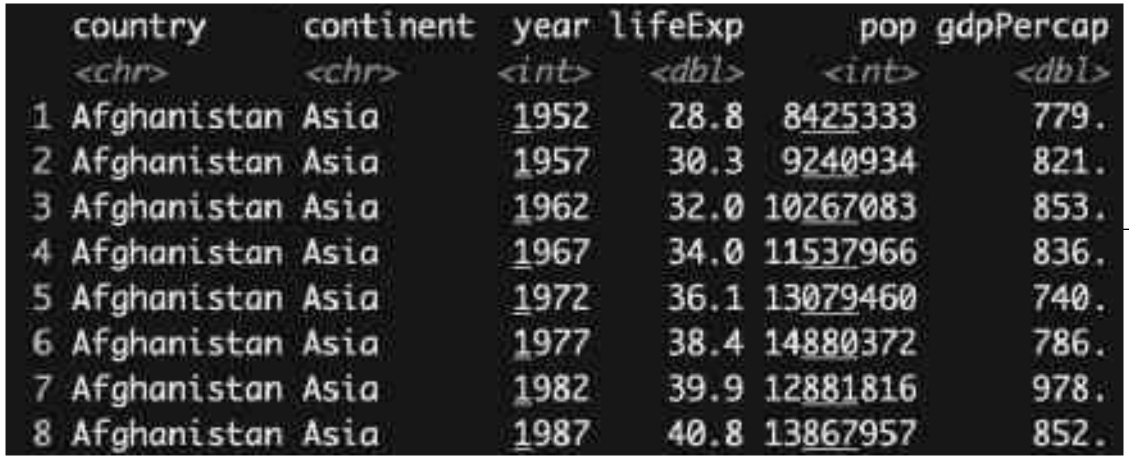
```
temp <- mean(tempsC)
round(temp)                                    18

tempsC %>% mean() %>% round()                  18
```

Data workflows (1/3)

```
world <- read_csv("gapminder.csv")
world
```

Import the dataset into the object world

Display the contents of the object



```
  country     continent  year lifeExp      pop gdpPercap
  <chr>       <chr>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
6 Afghanistan Asia       1977    38.4 14880372      786.
7 Afghanistan Asia       1982    39.9 12881816      978.
8 Afghanistan Asia       1987    40.8 13867957      852.
```

This is well-structured (tidy) data!

- Each **variable** is saved in its own **column**.
- Each **observation** is saved in its own **row**.

```
View(world)
```

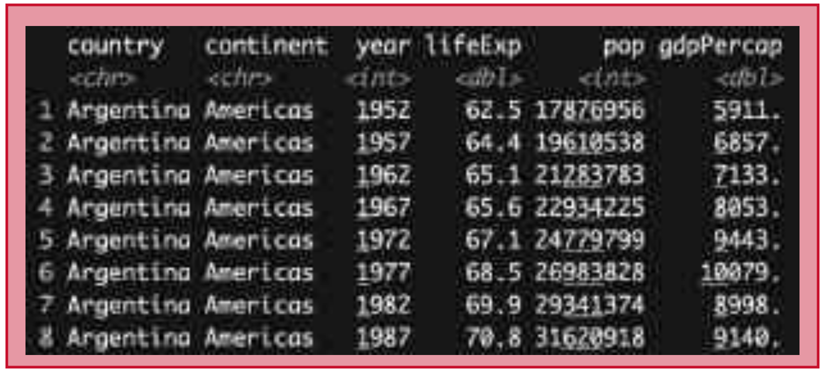
Browse the table like in Excel

%>%	is called a pipe, passing the date to the next function
==	is equal ("=" in mathematics)

Data workflows (2/3)

*Take the **world** dataset **and** filter it for the Americas*

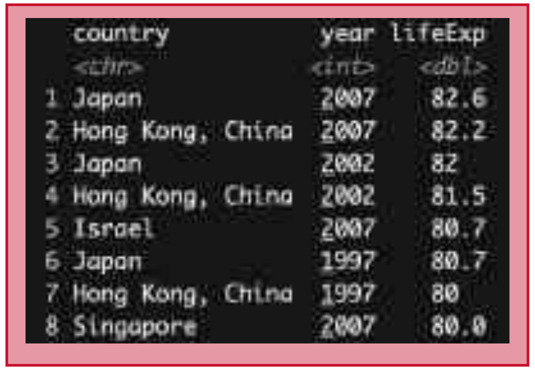
```
americas <- world %>%
  filter(continent == "Americas")
```



```
  country   continent  year lifeExp      pop gdpPercap
  <chr>     <chr>     <int>   <dbl>    <int>     <dbl>
1 Argentina Americas   1952    62.5 17876956     5911.
2 Argentina Americas   1957    64.4 19610538     6857.
3 Argentina Americas   1962    65.1 21283783     7133.
4 Argentina Americas   1967    65.6 22934225     8053.
5 Argentina Americas   1972    67.1 24779799     9443.
6 Argentina Americas   1977    68.5 26983828    10079.
7 Argentina Americas   1982    69.9 29341374     8998.
8 Argentina Americas   1987    70.8 31620918     9140.
```

*Take the **world** dataset **and** filter it for Asia **and** select the specified columns **and** arrange them by descending life expectancy.*

```
Asia_lifexp <- world %>%
  filter(continent == "Asia") %>%
  select(country,year,lifeExp) %>%
  arrange(desc(lifeExp))
```

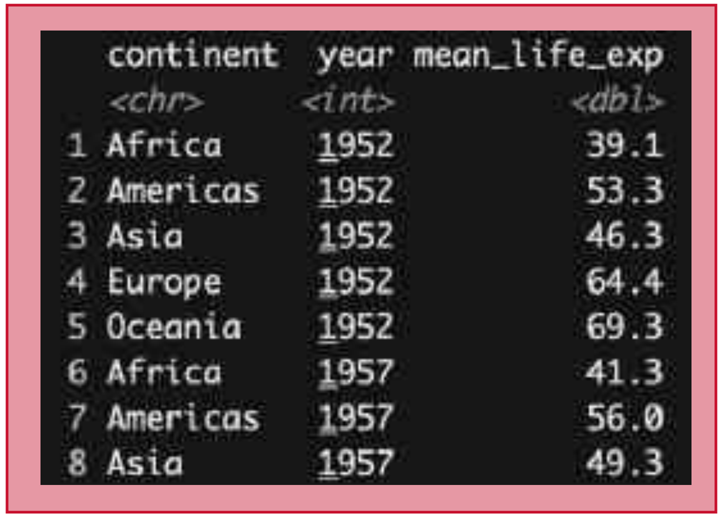


```
  country            year lifeExp
  <chr>             <int>   <dbl>
1 Japan              2007    82.6
2 Hong Kong, China   2007    82.2
3 Japan              2002    82
4 Hong Kong, China   2002    81.5
5 Israel             2007    80.7
6 Japan              1997    80.7
7 Hong Kong, China   1997    80
8 Singapore          2007    80.0
```

Data workflows (3/3)

*Take the **world** dataset **and** group it by continent and year and for every group **then** summarize the life expectancy by taking the mean **and then** arrange the output by year and continent*

```
lifeexp_by_continent <- world %>%
    group_by(continent, year) %>%
    summarize(mean_life_exp = mean(lifeExp)) %>%
    arrange(year,continent)
```

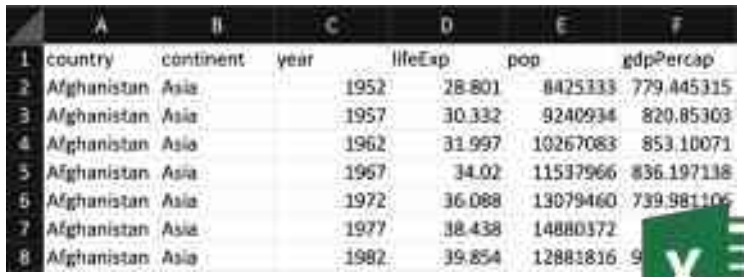


```
  continent  year mean_life_exp
  <chr>     <int>         <dbl>
1 Africa     1952          39.1
2 Americas   1952          53.3
3 Asia       1952          46.3
4 Europe     1952          64.4
5 Oceania    1952          69.3
6 Africa     1957          41.3
7 Americas   1957          56.0
8 Asia       1957          49.3
```

If you "speak" R you can directly "talk" to your data and tell it what you want to do.

How can we visualize data in R?

Data visualization with ggplot

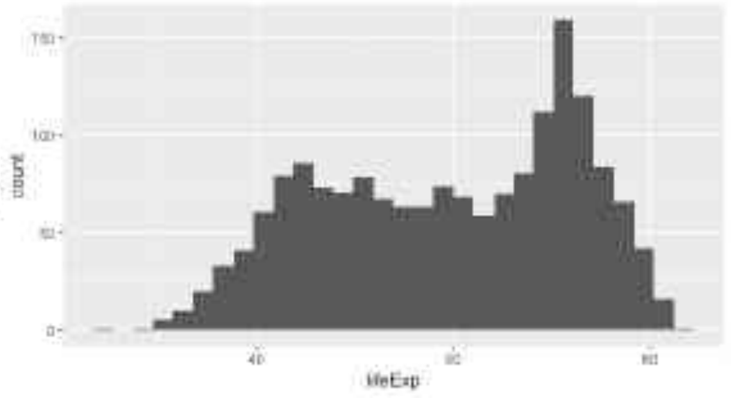


	A	B	C	D	E	F
1	country	continent	year	lifeExp	pop	gdpPercap
2	Afghanistan	Asia	1952	28.801	8425333	779.445315
3	Afghanistan	Asia	1957	30.332	9240934	820.85303
4	Afghanistan	Asia	1962	31.997	10267083	853.10071
5	Afghanistan	Asia	1967	34.02	11537966	836.197138
6	Afghanistan	Asia	1972	36.088	13079460	739.981106
7	Afghanistan	Asia	1977	38.438	14880372	[illegible]
8	Afghanistan	Asia	1982	39.854	12881816	9

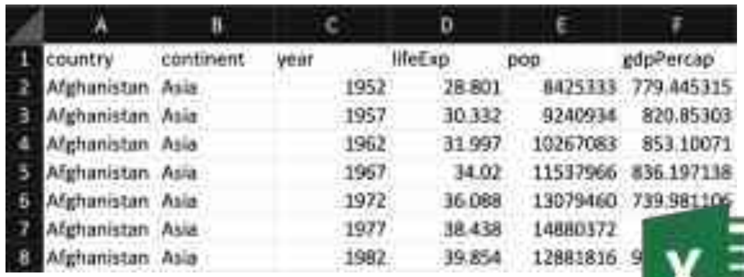
https://www.gapminder.org/data/documentation/

```
world <- read_csv("gapminder.csv")
```

```
ggplot(data = world, mapping = aes(x = lifeExp)) +
  geom_histogram()
```



Data visualization with ggplot



	A	B	C	D	E	F
1	country	continent	year	lifeExp	pop	gdpPercap
2	Afghanistan	Asia	1952	28.801	8425333	779.445315
3	Afghanistan	Asia	1957	30.332	9240934	820.85303
4	Afghanistan	Asia	1962	31.997	10267083	853.10071
5	Afghanistan	Asia	1967	34.02	11537966	836.197138
6	Afghanistan	Asia	1972	36.088	13079460	739.981106
7	Afghanistan	Asia	1977	38.438	14880372	[illegible]
8	Afghanistan	Asia	1982	39.854	12881816	[illegible]

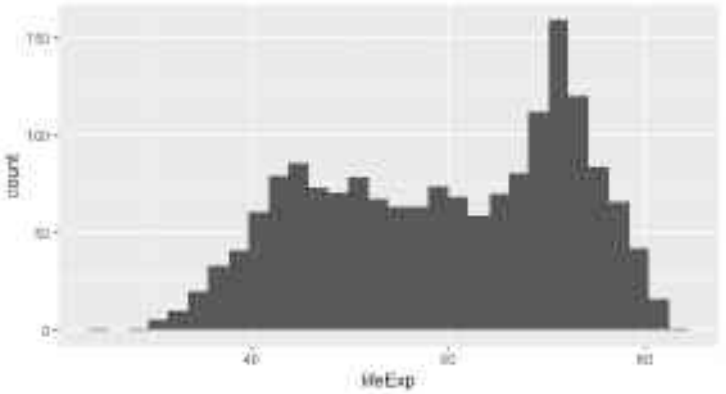
https://www.gapminder.org/data/documentation/

```
world <- read_csv("gapminder.csv”)
```

What *Where*

```
ggplot(data = world, mapping = aes(x = lifeExp)) +
  geom_histogram()
```

How



*“I want to plot the **world dataset** with the **life expectancy mapped to the x-axis** using **a histogram**”*

What

Where

```
plot <- ggplot(data = world, mapping = aes(x = continent, y = lifeExp))
```

How

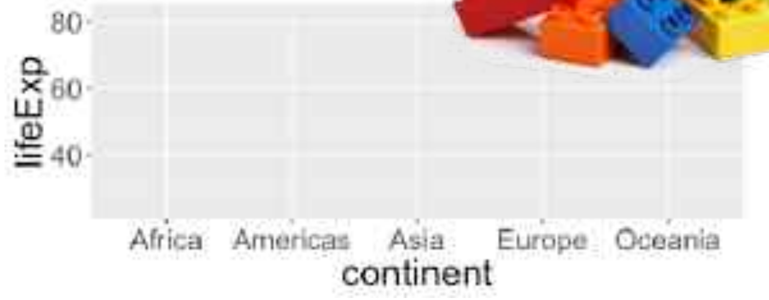
```
plot
```

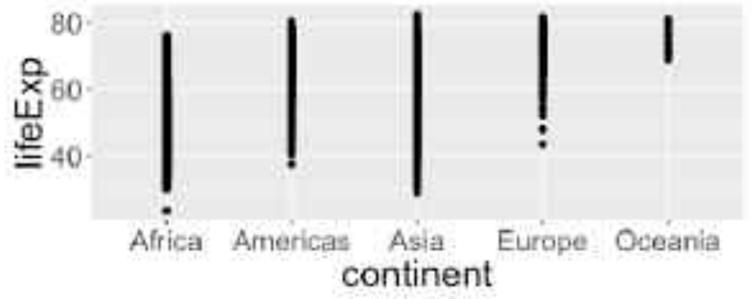
How

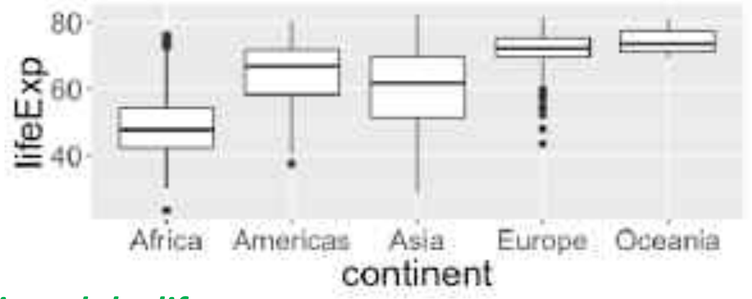
```
plot + geom_point()
```

How

```
plot + geom_boxplot()
```







*"I want to plot the **world dataset** with the **continents mapped to the x-axis and the life expectancy to the y-axis using points, boxplots or more**"*

What

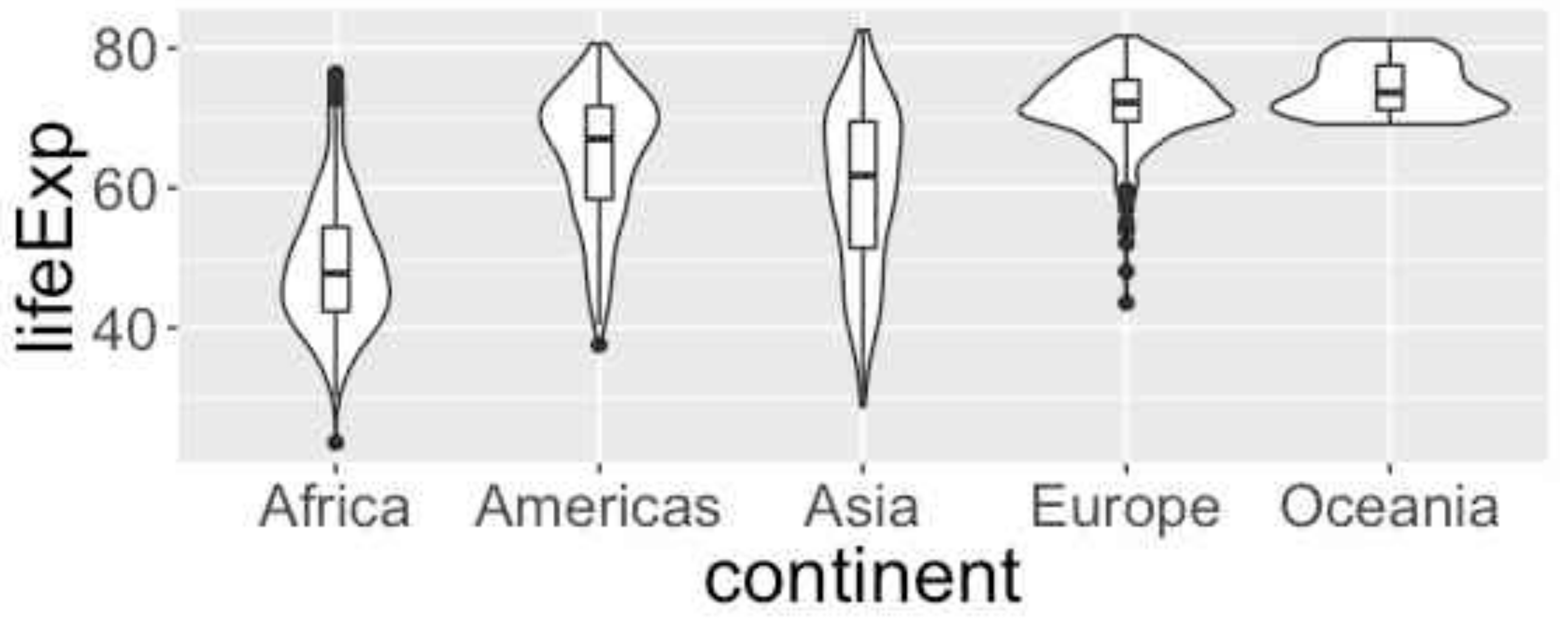
Where

```
plot <- ggplot(data = world, mapping = aes(x = continent, y = lifeExp))
```

How

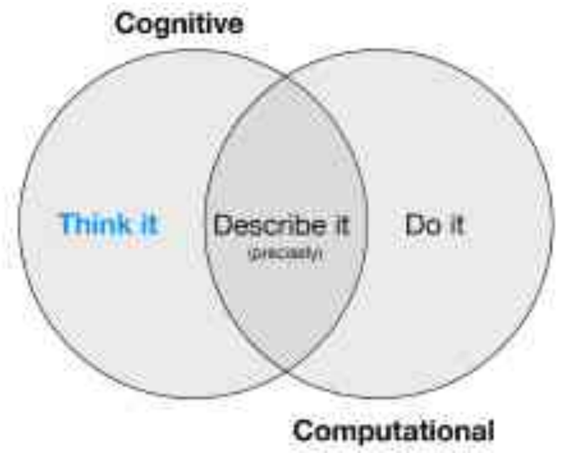
```
plot + + geom_violin() + geom_boxplot(width = 0.1)
```

All geoms are combinable!



*"I want to plot the **world dataset** with the continents mapped to the x-axis and the life expectancy to the y-axis using points, boxplots or more"*

ggplot syntax



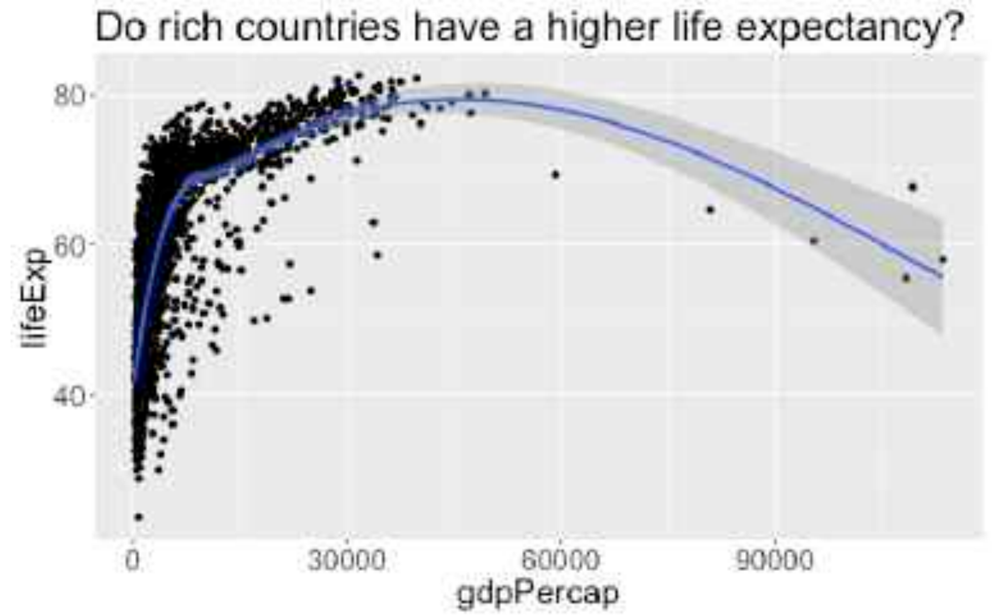
```
ggplot(what dataset, map parameters to plot dimensions ) +
  what to draw
```

- Let's start asking questions!
  - How did the world change? How the continents and individual countries?
  - How are GDP and lifespan linked? Is this the same across continents?

Data exploration

Do rich countries have a higher life expectancy?

```
ggplot(data = world, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Do rich countries have a higher life expectancy?")
```



Up to 50000 gdpPercap yes, above life expectancy decreases again

Data exploration

Reviewer: is this continent-specific?

Do rich countries have a higher life expectancy?

```
ggplot(data = world, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Do rich countries have a higher life expectancy?")
```

How would you change the code?

?

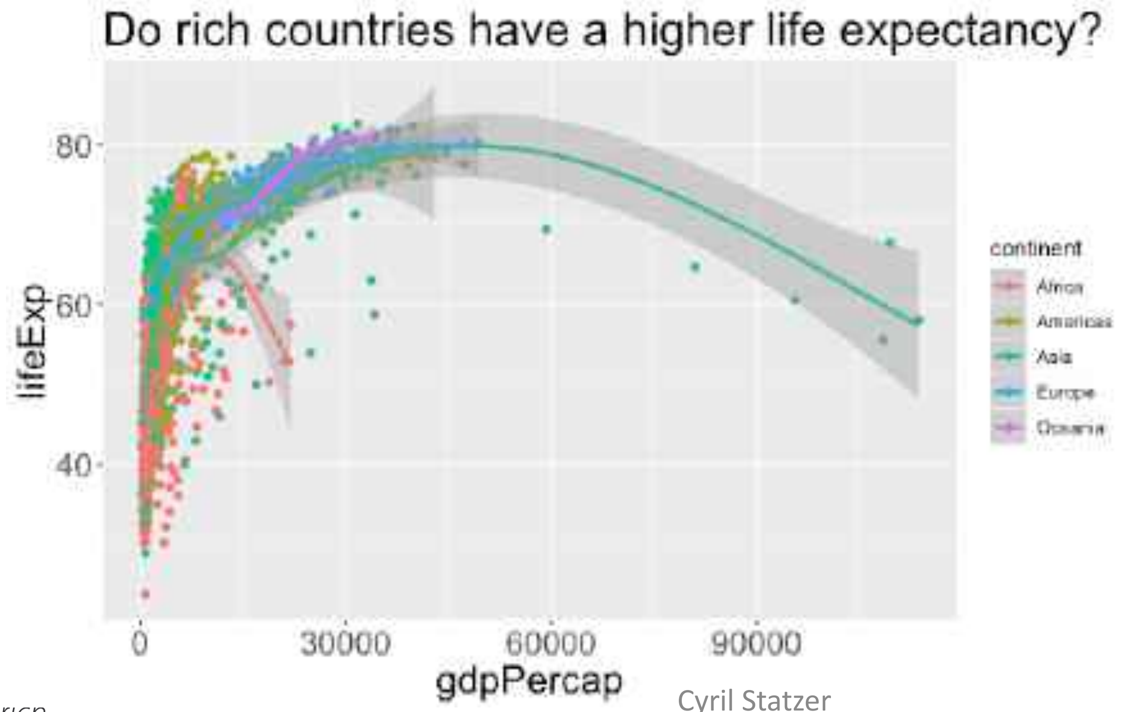
?

Reviewer: is this continent-specific?

Data exploration

Do rich countries have a higher life expectancy?

```
ggplot(data = world, mapping = aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Do rich countries have a higher life expectancy?")
```



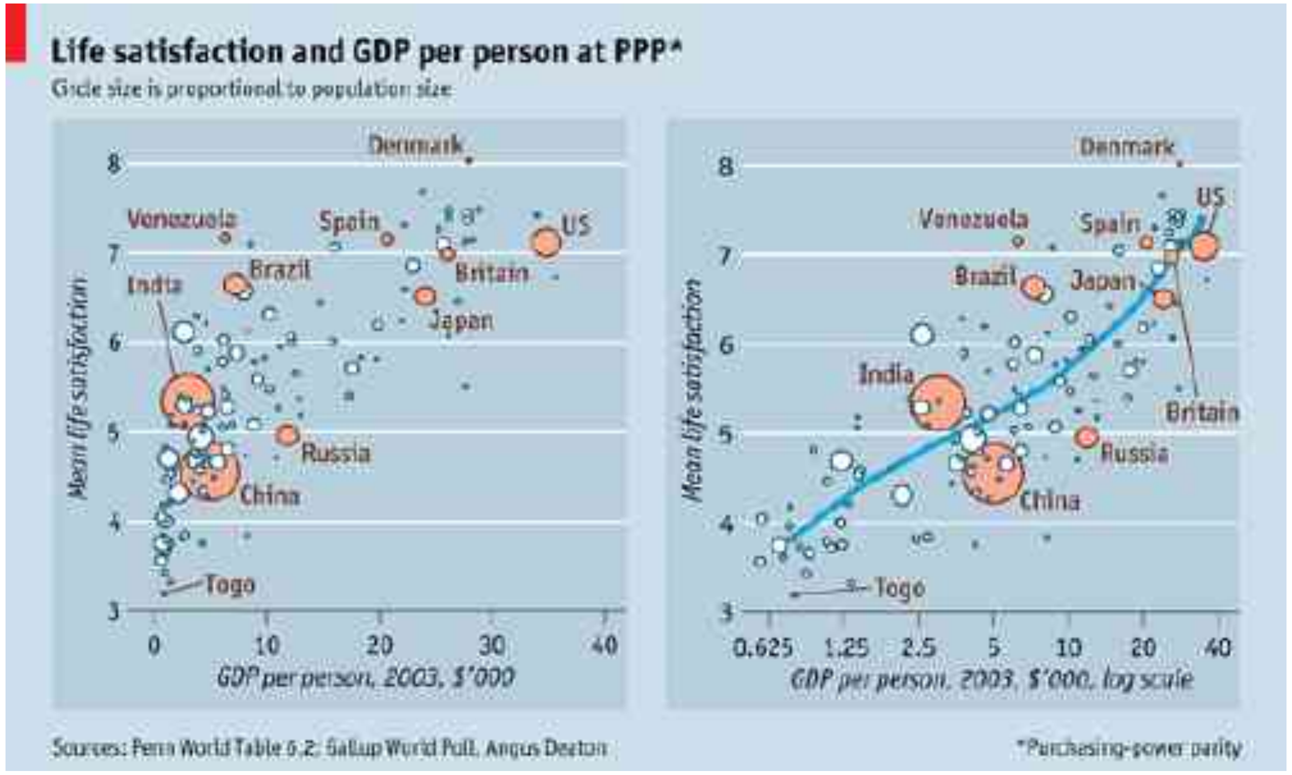
life expectancy increases monotonously with gdpPercap in Oceania, Europe and the Americas

Up to 10000 gdpPercap yes, above life expectancy decreases again in Africa

Up to 50000 gdpPercap yes, above life expectancy decreases again in Asia

Outlook

Can we do better than The Economist?



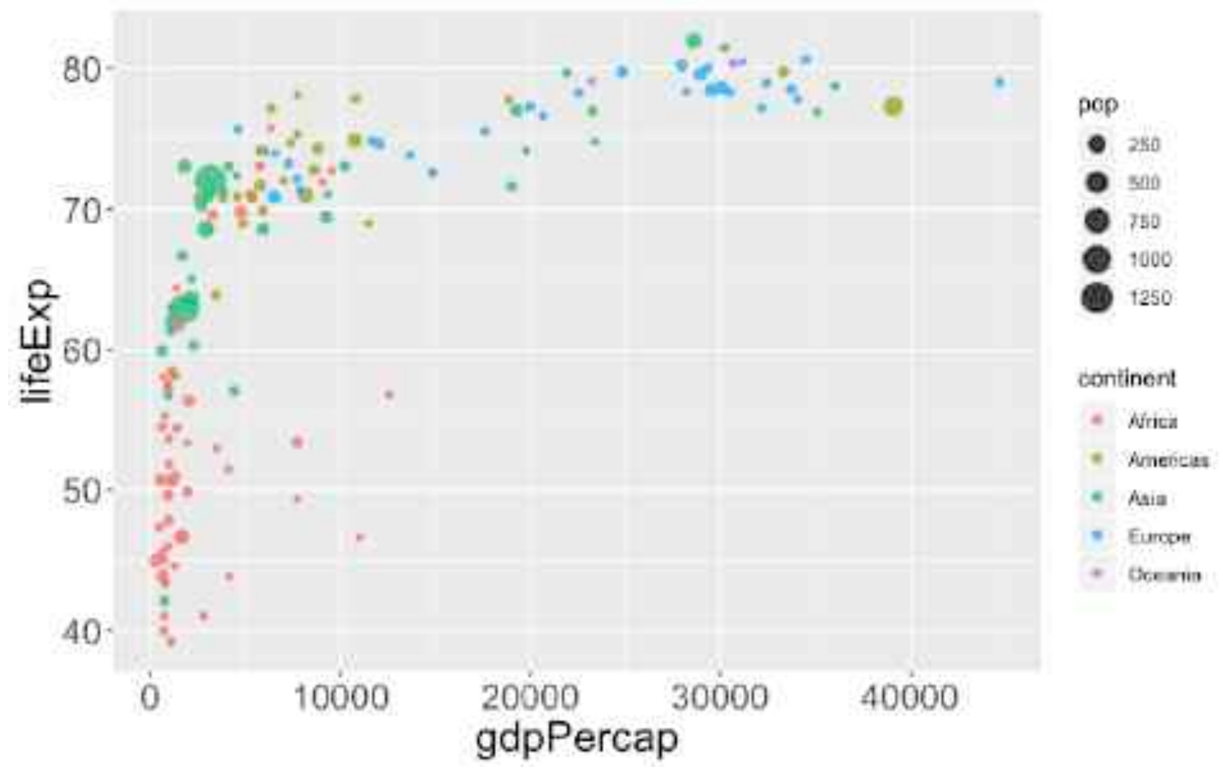
The Economist

They looked at:
GDP vs. life satisfaction in 2003

We will compare
GDP vs. life expectancy in 2002

Outlook
Life expectancy vs. GDP in 2002

```
world_2002 <- world %>%
  filter(year == 2002) %>%
  mutate(pop=pop/1000000)



plot <- ggplot(data = world_2002,
               aes(x= gdpPercap,
                   y= lifeExp,
                   size = pop,
                   color = continent)) +
  geom_point(alpha=0.8)

plot
```

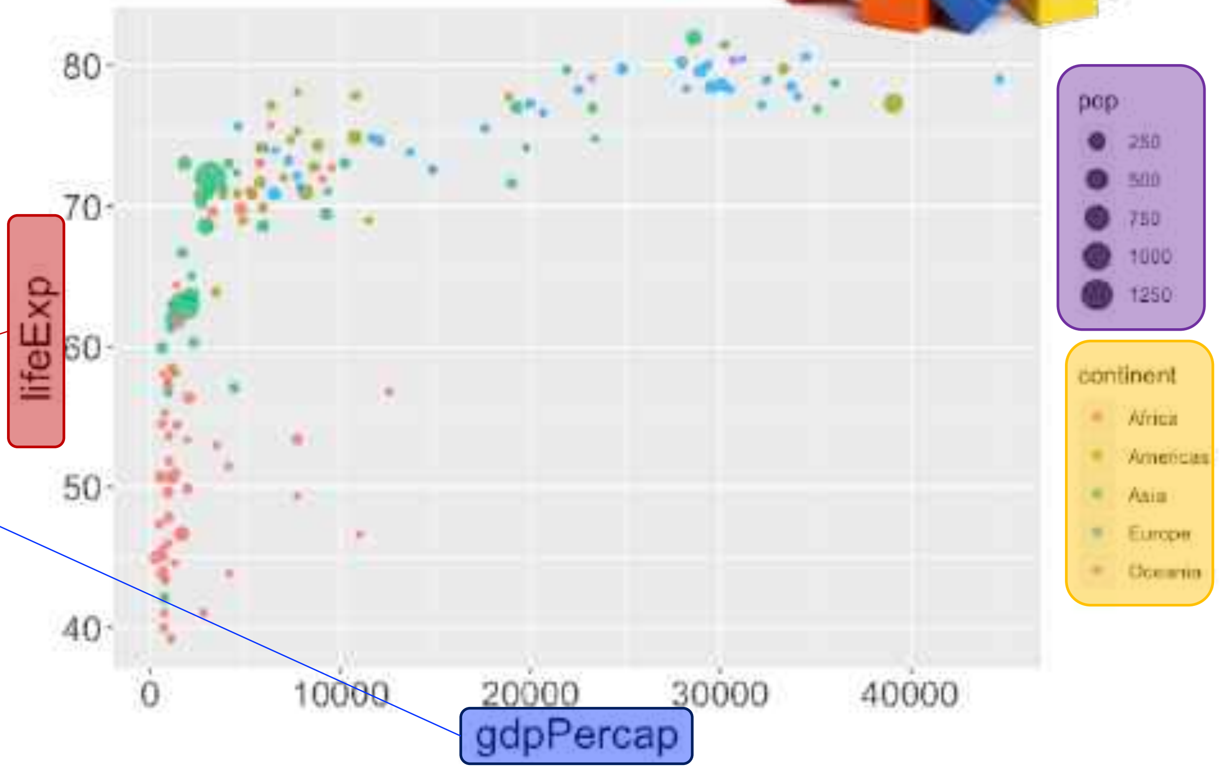


…not great

Outlook
Life expectancy vs. GDP in 2002

```
world_2002 <- world %>%
  filter(year == 2002) %>%
  mutate(pop=pop/1000000)
```

```
plot <- ggplot(data = world_2002,
               aes(x= gdpPercap,
                   y= lifeExp,
                   size = pop,
                   color = continent)) +
  geom_point(alpha=0.8)

plot
```

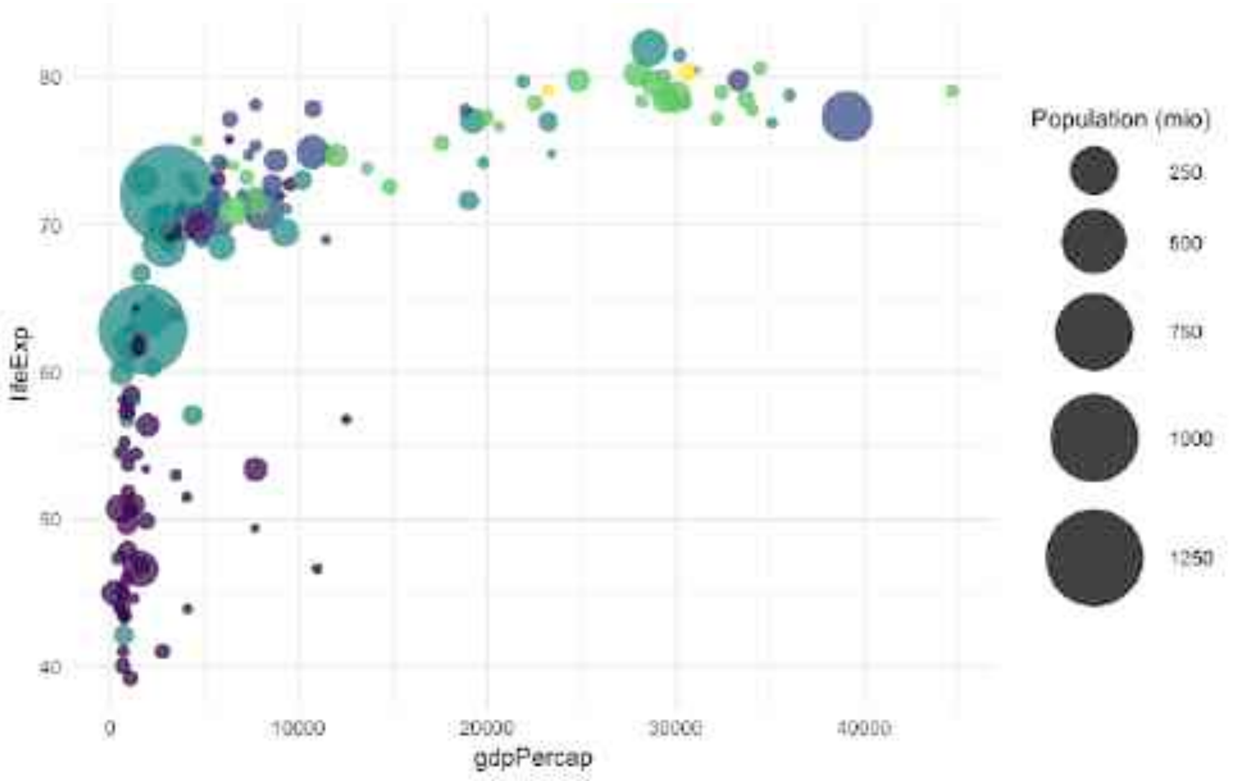


…not great

Outlook

Life expectancy vs. GDP in 2002

```
plot +
  scale_size(range = c(1, 20),
             name="Population (mio)") +
  scale_color_viridis(discrete=TRUE,
                      guide=FALSE) +
  theme_minimal()
```

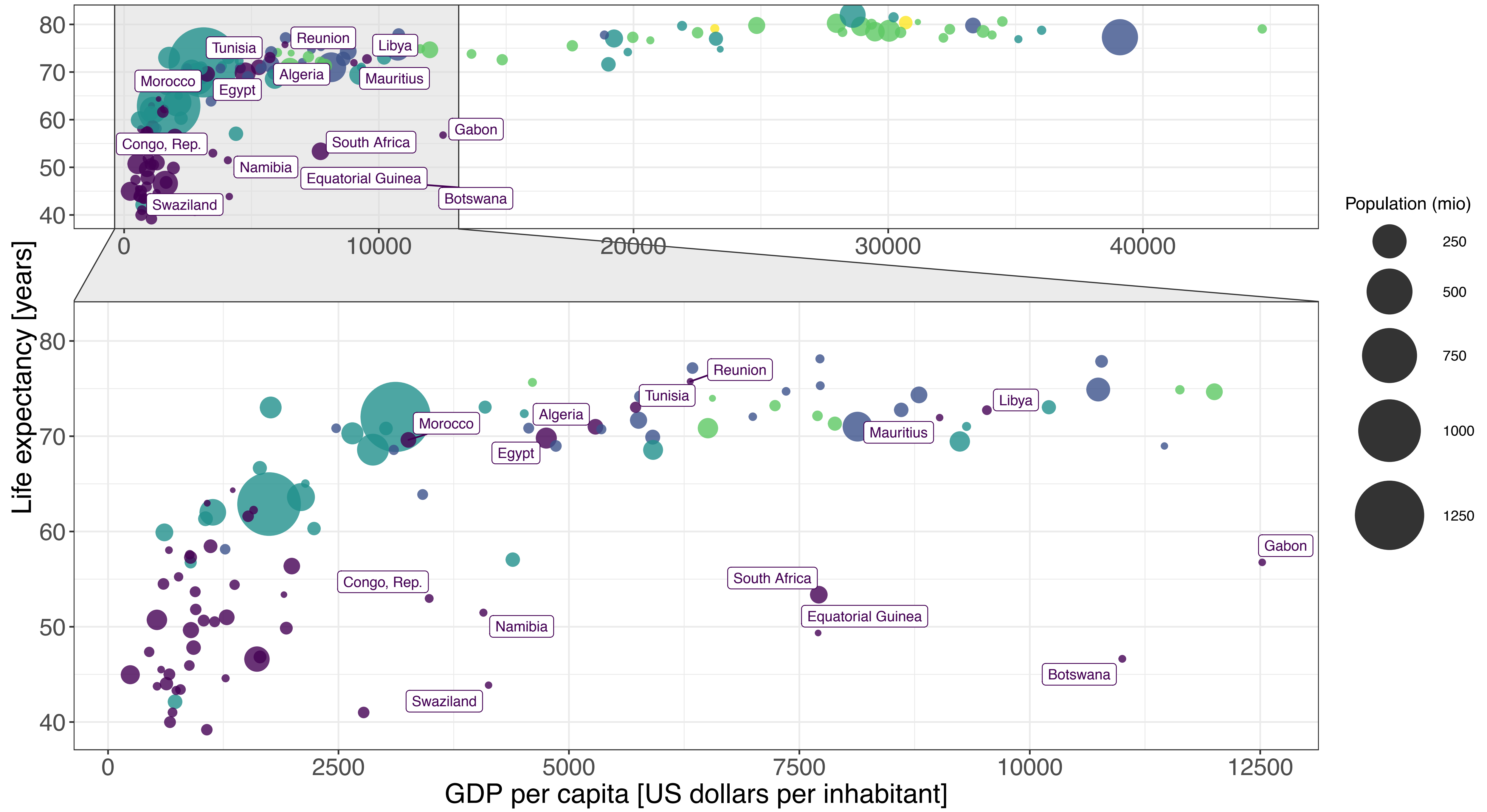


"styling the plot"

Next level: what is the life expectancy in the wealthiest African countries?

…better

World demographics in 2002



Outlook

Life expectancy vs. GDP in 2002

```
library(viridis)
library(ggforce)
library(ggrepel)

world_2002 <- world %>%
  filter(year == 2002) %>%
  mutate(pop=pop/1000000)

africa_leading <- world_2002 %>%
  filter(continent == "Africa",
         gdpPercap > 3000)

plot <- ggplot(data = world_2002, aes(x=gdpPercap, y=lifeExp, size = pop, color = continent, label = country)) +
  geom_point(alpha=0.8) +
  scale_size(range = c(1, 20), name="Population (mio)") +
  scale_color_viridis(discrete=TRUE, guide=FALSE) +
  facet_zoom(x = continent == "Africa") +
  geom_label_repel(data = africa_leading,size = 3) +
  theme_bw() +
  labs(title = "World demographics in 2002",x = "GDP per capita [US dollars per inhabitant]", y = "Life expectancy
  [years]")

plot
```

Year 2002

Labelling Africa

Logic of the plot

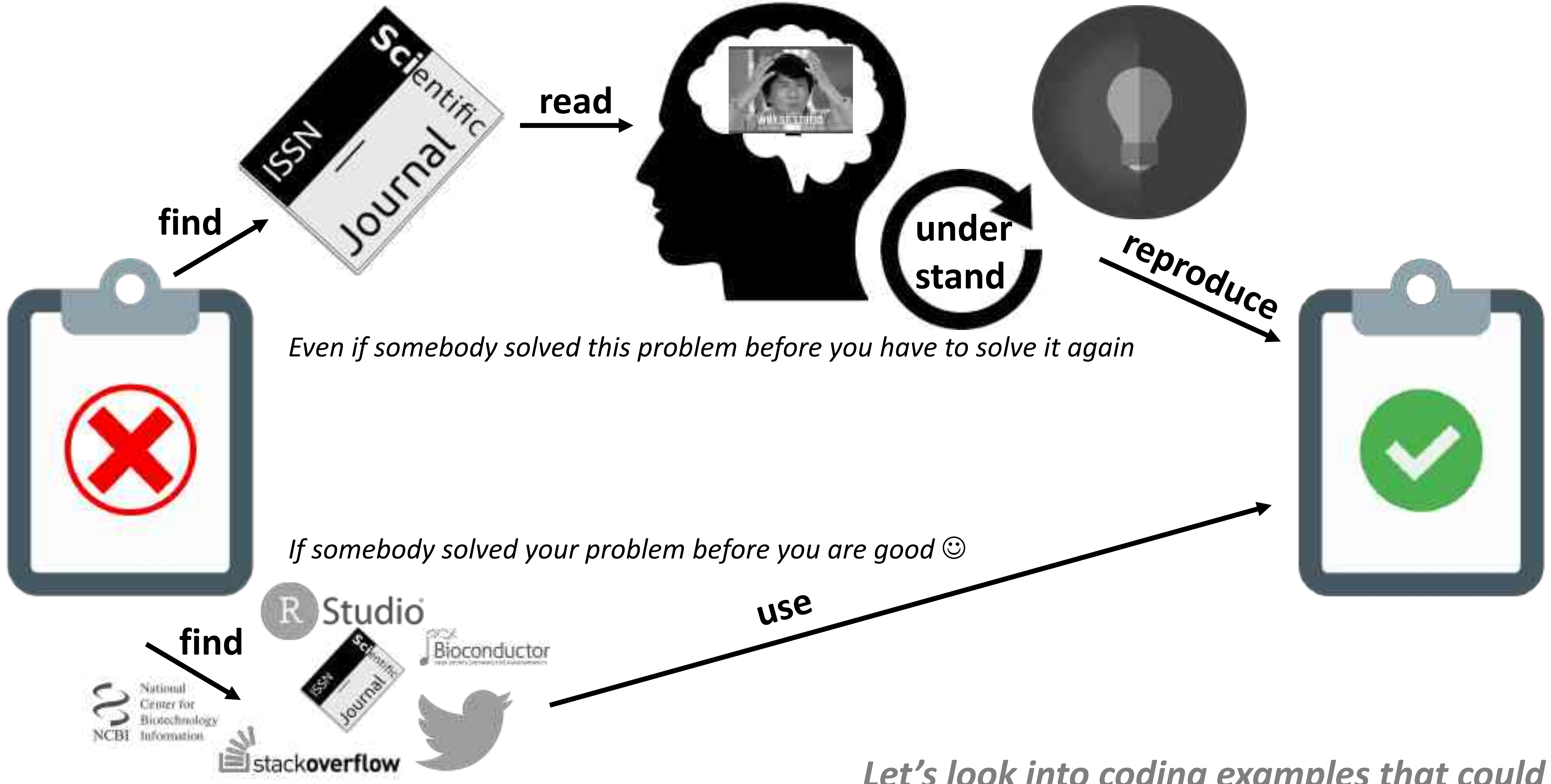
Styling

Labels

8 lines of code

... much better

R in the lab

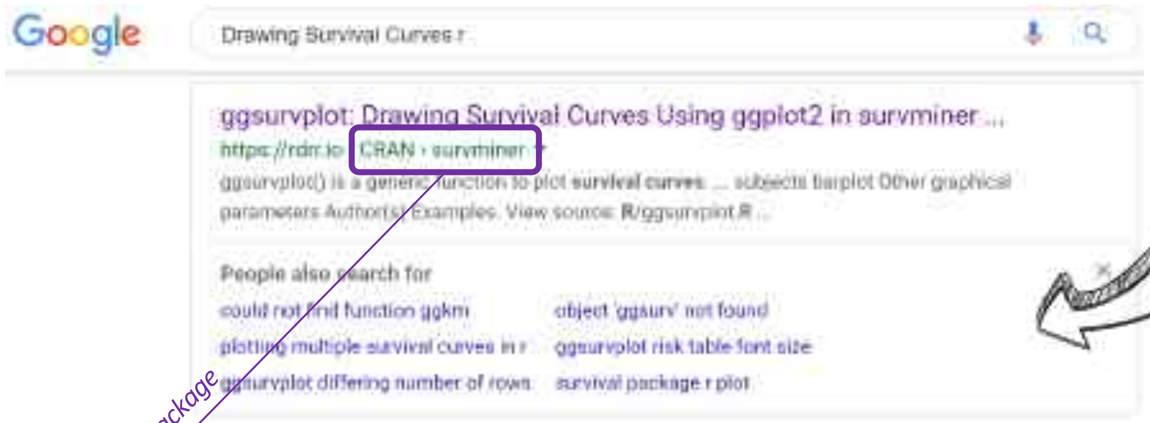


Even if somebody solved this problem before you have to solve it again

If somebody solved your problem before you are good ☺

Let's look into coding examples that could be useful for everyday lab work:

Example 1: plotting lifespans


Drawing Survival Curves r

ggsurvplot: Drawing Survival Curves Using ggplot2 in survminer ...
https://rdrr.io › CRAN › survminer ›
ggsurvplot() is a generic function to plot survival curves. ... subjects barplot Other graphical parameters Author(s) Examples. View source: R/ggsurvplot.R ...

People also search for

could not find function ggkm	object 'ggsurv' not found
plotting multiple survival curves in r	ggsurvplot risk table font size
ggsurvplot differing number of rows	survival package r plot

package

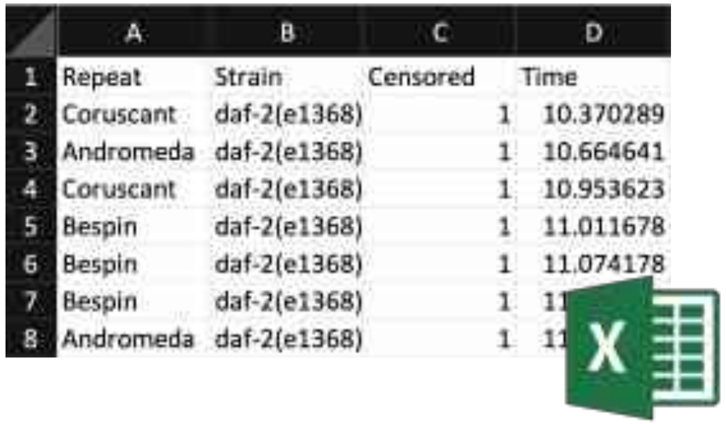
literatue

import

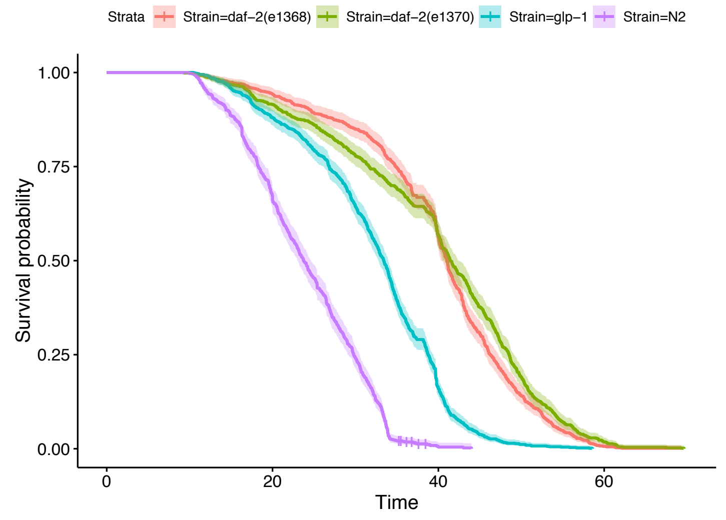
plot

```
library(tidyverse)
library(survminer)
library(survival)

data <- read_csv(file = "lifespan.csv")

fit<- survfit(Surv(Time, Censored) ~ Strain, data = data)
ggsurvplot(fit = fit, data = data)
```

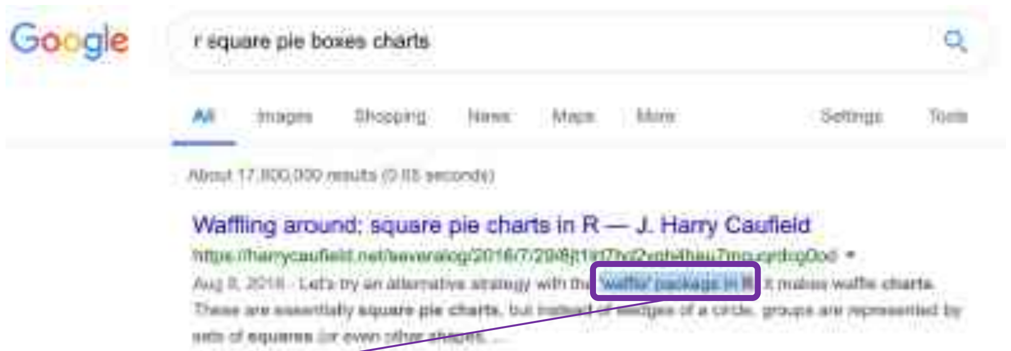
lifespan.csv



	A	B	C	D
1	Repeat	Strain	Censored	Time
2	Coruscant	daf-2(e1368)	1	10.370289
3	Andromeda	daf-2(e1368)	1	10.664641
4	Coruscant	daf-2(e1368)	1	10.953623
5	Bespin	daf-2(e1368)	1	11.011678
6	Bespin	daf-2(e1368)	1	11.074178
7	Bespin	daf-2(e1368)	1	11
8	Andromeda	daf-2(e1368)	1	11



Example 2: plotting microscopy scoring


r square pie boxes charts

All Images Shopping News Maps More Settings Tools

About 17,800,000 results (0.18 seconds)

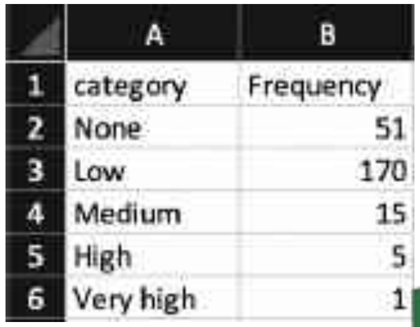
Waffling around: square pie charts in R — J. Harry Caufield

https://harrycaufield.net/[illegible] ▾

Aug 8, 2016 - Let's try an alternative strategy with the 'waffle' package in R. It makes waffle charts. These are essentially square pie charts, but instead of wedges of a circle, groups are represented by sets of squares (or even other shapes ...

package

microscopy.csv



	A	B
1	category	Frequency
2	None	51
3	Low	170
4	Medium	15
5	High	5
6	Very high	1

literatue

```
library(tidyverse)
library(waffle)

colors <- c("grey50","bisque1", "darkorange2", "red","darkred")
```

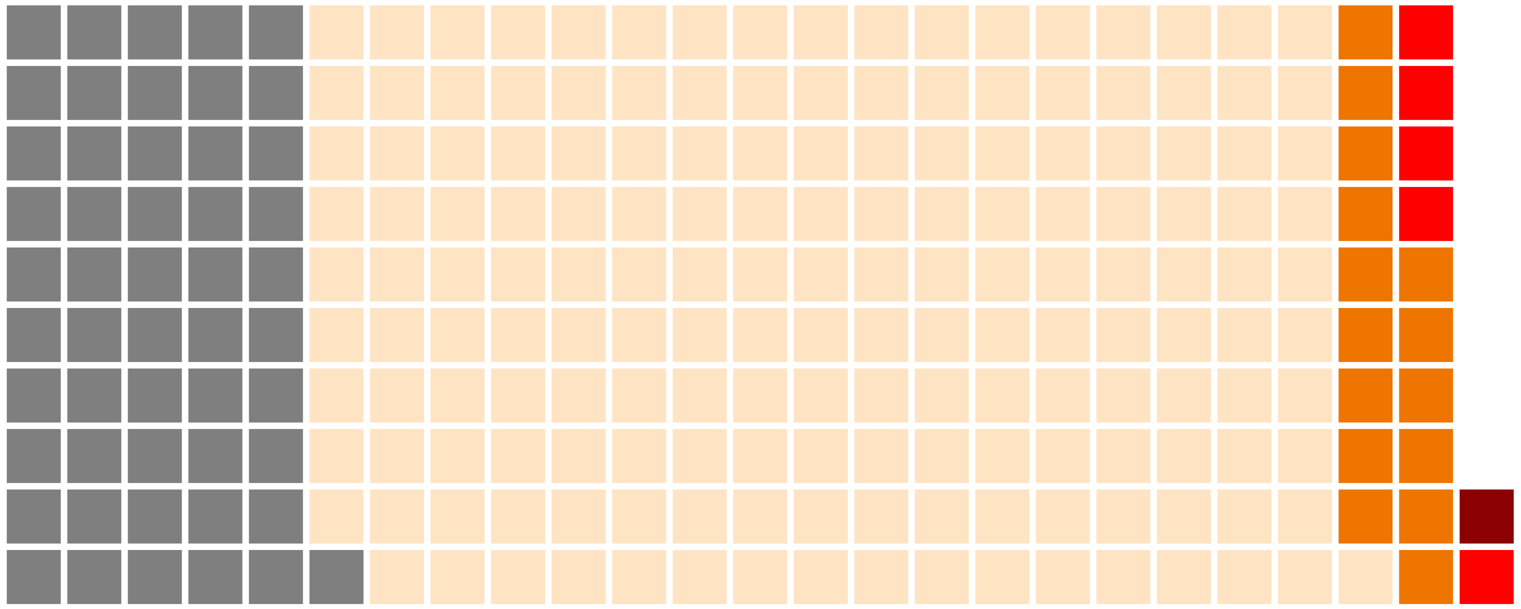
import

```
data <- read_csv(file = "microscopy.csv")
absolute <- data$Frequency
names(absolute) <- data$category
```

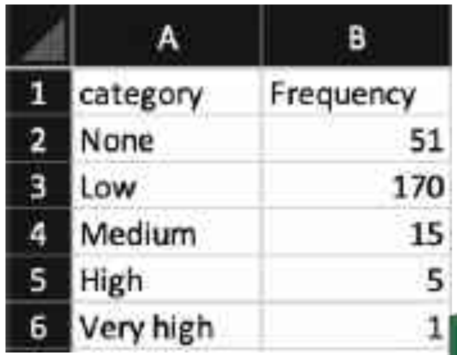
plot

```
waffle(absolute, rows = 10, size = 1, colors = colors, legend_pos = "bottom",title = "Absolute observations")
relative <- round(absolute/sum(absolute) * 25)
waffle(relative, rows = 1, size = 1, colors = colors, legend_pos = "bottom",title = "Relative fractions")
```

Absolute observations

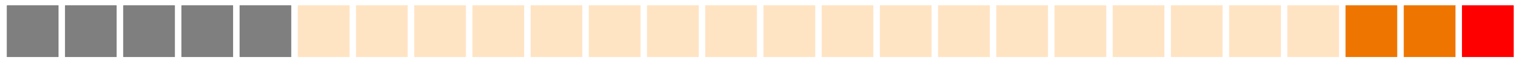


microscopy.csv



	A	B
1	category	Frequency
2	None	51
3	Low	170
4	Medium	15
5	High	5
6	Very high	1

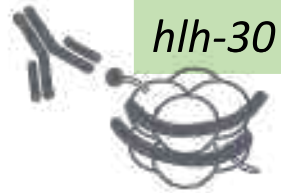
Relative fractions



⬛ None 🟨 Low 🟧 Medium 🟥 High 🟫 Very high

Example 3: Merging CHIP & RNA-seq data

gene	CHIP_count
daf-2	0.96
emb-9	1.80
hlh-30	1.60



gene	RNA_FC	RNA_pval
cdc-40	1.01	0.20
pmp-30	0.89	0.07
Y11D7A.11	1.13	0.60
hlh-30	1.43	0.04
gst-4	2.04	0.0001
daf-2	0.87	0.002

gene	CHIP_count	RNA_FC	RNA_pval
daf-2	0.96	0.87	0.002
emb-9	1.8	NA	NA
hlh-30	1.6	1.43	0.04
cdc-40	NA	1.01	0.2
pmp-30	NA	0.89	0.07
Y11D7A.11	NA	1.13	0.6
gst-4	NA	2.04	1.00E-04

15000 – 60000 rows in each dataset

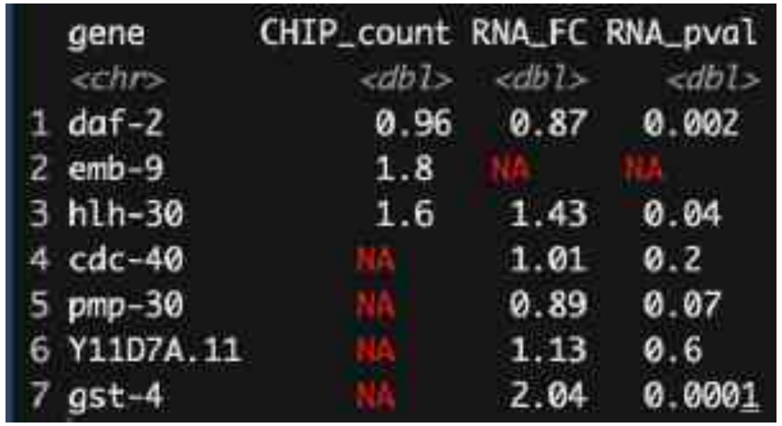
Example 3: Merging CHIP & RNA-seq data

gene	CHIP_count
daf-2	0.96
emb-9	1.80
hlh-30	1.60

gene	RNA_FC	RNA_pval
cdc-40	1.01	0.20
pmp-30	0.89	0.07
Y11D7A.11	1.13	0.60
hlh-30	1.43	0.04
gst-4	2.04	0.0001
daf-2	0.87	0.002

gene	CHIP_count	RNA_FC	RNA_pval
daf-2	0.96	0.87	0.002
emb-9	1.8	NA	NA
hlh-30	1.6	1.43	0.04
cdc-40	NA	1.01	0.2
pmp-30	NA	0.89	0.07
Y11D7A.11	NA	1.13	0.6
gst-4	NA	2.04	1.00E-04

```
chip <- read_csv("chip.csv")

rnaseq <- read_csv("rna_seq.csv")

full_join(chip,rnaseq, by = "gene")
```



```
  gene      CHIP_count RNA_FC RNA_pval
  <chr>          <dbl>  <dbl>    <dbl>
1 daf-2           0.96   0.87   0.002
2 emb-9           1.8   NA     NA
3 hlh-30          1.6    1.43   0.04
4 cdc-40         NA      1.01   0.2
5 pmp-30         NA      0.89   0.07
6 Y11D7A.11      NA      1.13   0.6
7 gst-4          NA      2.04   0.0001
```

Common code patterns

Kirschner, 2018

- Packages («literature you use»)

```
library(tidyverse)
```

general packages

```
library(survminer)
library(waffle)
library(survival)
```

specialized packages

- Import your data

`read_csv()`

`waffle()` **`full_join()`**

- Perform computation / plotting

`survfit()` **`ggsurvplot()`**

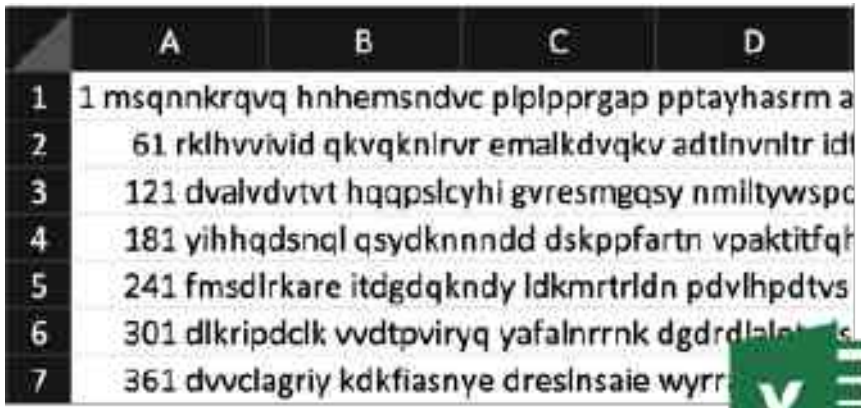
Most of your problems have already been solved by somebody! There is no need to drag squares around in powerpoint to make a waffle chart or to manually merge datasets in excel, let R do it for you

Advanced functions

• `as.numeric()`	`interpret as a number`
• `unlist()`	`remove indexing`
• `paste()`	`combine characters`
• `str_replace_all()`	`similar to Cmd Find Replace`
• `is.na()`	`Is data missing? YES or NO`
• `!`	`Invert logic`
• `separate`	`Split characters on pattern`

Advanced 1: Parsing strings

aa_sequence.csv

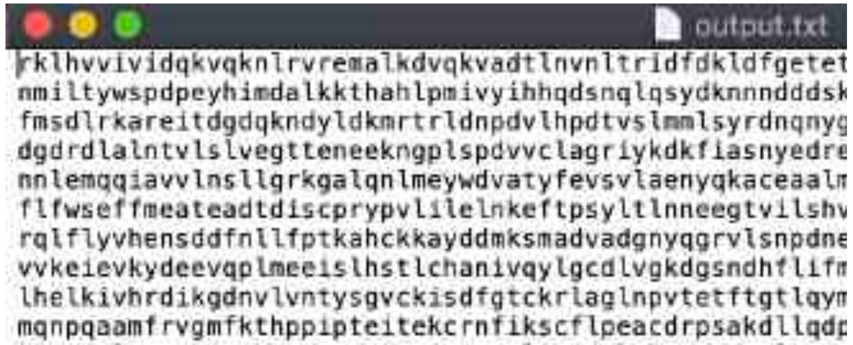


	A	B	C	D
1	1 msqnnkrqvq hnhemsndvc plplpprgap pptayhasrm a			
2	61 rklhvvivid qkvqknlrvr emalkdvqkv adtlnvnltr id			
3	121 dvalvdvtvt hqqpslcyhi gvresmgqsy nmiltywspc			
4	181 yihhqdsnql qsydknnndd dskppfartn vpaktitfq			
5	241 fmsdlrkare itdgdqkndy ldkmrtrldn pdvlhpdtvs			
6	301 dlkripdclk vvdtpviryq yafalnrrnk dgdrd			
7	361 dvvclagriy kdkfiasnye dreslnsaie wyrr			

```r
library(tidyverse)

input <- read_csv("aa_sequence.csv")

str <- input %>% unlist() %>% paste(., collapse=" ")
str_proc <- str_replace_all(string = str,pattern = "[[:digit:],[:space:]]",replacement = "")

write_lines(x = str_proc,path = "./output.txt")
```

More advanced

output.txt



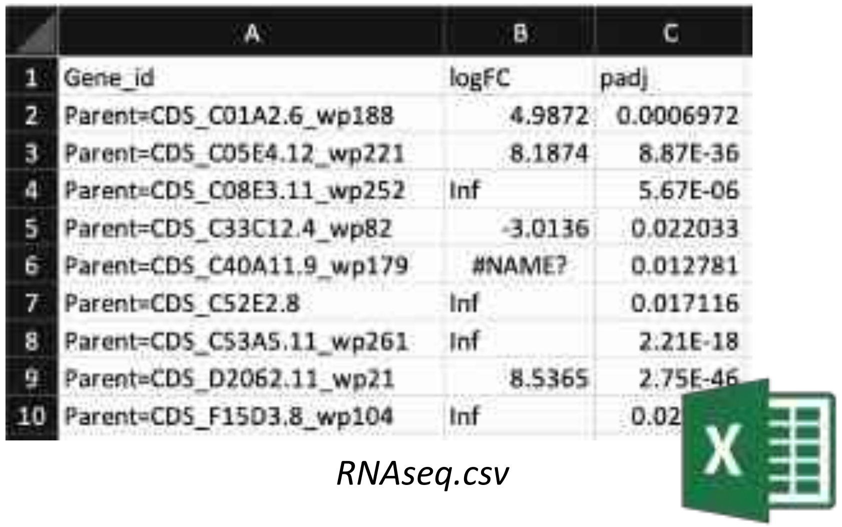
```
rklhvvividqkvqknlrvremalkdvqkvadtlnvnltridfdkldfgetet
nmiltywspdpeyhimdalkkthahlpmivyihhqdsnqlqsydknnndddsk
fmsdlrkareitdgdqkndyldkmrtrldnpdvlhpdtvslmmlsyrdnqnyg
dgdrdlalntvlslvegtteneekngplspdvvclagriykdkfiasnyedre
nnlemqqiavvlnsllgrkgalqnlmeywdvatyfevsvlaenyqkaceaalm
flfwseffmeateadtdiscprypvlilelnkeftpsyltlnneegtvilshv
rqlflyvhensddfnllfptkahckkayddmksmadvadgnyqgrvlsnpdne
vvkeievkydeevqplmeeislhstlchanivqylgcdlvgkdgsndhflifm
lhelkivhrdikgdnvlvntysgvckisdfgtckrlaglnpvtetftgtlqym
mqnpqaamfrvgmfkthppipteitekcrnfikscflpeacdrpsakdllqdp
```

Advanced 2: data cleaning (RNAseq)

literatue

import

```
library(tidyverse)

data <- read_csv(file = "RNAseq.csv")
```



	A	B	C
1	Gene_id	logFC	padj
2	Parent=CDS_C01A2.6_wp188	4.9872	0.0006972
3	Parent=CDS_C05E4.12_wp221	8.1874	8.87E-36
4	Parent=CDS_C08E3.11_wp252	Inf	5.67E-06
5	Parent=CDS_C33C12.4_wp82	-3.0136	0.022033
6	Parent=CDS_C40A11.9_wp179	#NAME?	0.012781
7	Parent=CDS_C52E2.8	Inf	0.017116
8	Parent=CDS_C53A5.11_wp261	Inf	2.21E-18
9	Parent=CDS_D2062.11_wp21	8.5365	2.75E-46
10	Parent=CDS_F15D3.8_wp104	Inf	0.02

RNAseq.csv

cleaning

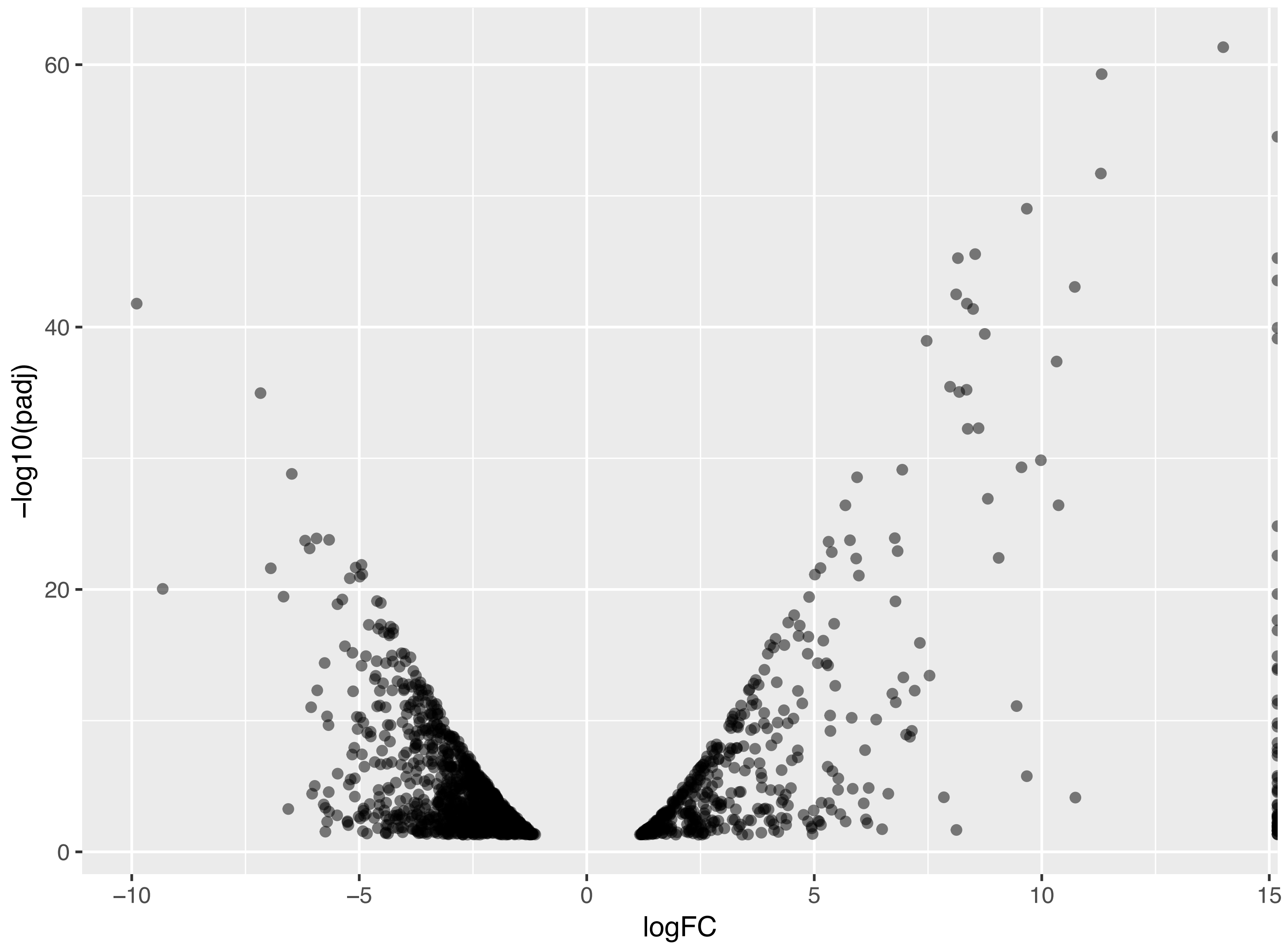
```
data <- data %>% mutate(logFC = as.numeric(logFC)) %>% filter(!is.na(logFC))
readable <- data %>% separate(col = Gene_id,into = c("Junk","ID"),sep = "_")
readable <- readable %>% filter(!is.na(ID))
```

plot

```
ggplot(data = data, aes(x = logFC,y = -log10(padj))) +
  geom_point(alpha = 0.5)
```

Parent=Transcript_**R02E12.2b.1**	**R02E12.2b.1**
Parent=Pseudogene_**Y105C5B.30** →	**Y105C5B.30**
Parent=CDS_**R06A4.3**	**R06A4.3**

Volcano plot



96-well assay

literatue

```
library(tidyverse)
library(ggplot2)
library(magrittr)
library(scatterpie)
```

Import & clean

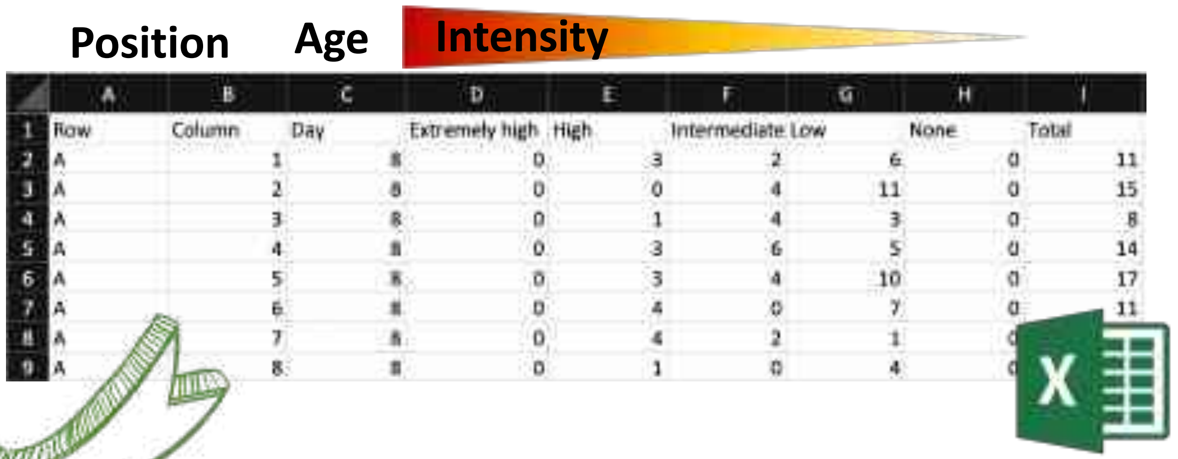
```
lets<- c("H","G","F","E","D","C","B","A")
df <- read_csv("96well.csv")
df <- df %>% mutate(region = paste0(Row,Column), Row = match(Row, lets))
df[,4:9] <- df[,4:9] / 80
```

plot

```
p <- ggplot() +
  geom_point(data=expand.grid(seq(1, 12), seq(1,8)), aes(x=Var1, y=Var2),
                             color="grey80", fill="white", shape=21, size=6) +
  geom_scatterpie(aes(x= Column, y= Row, r = Total, group = region), data=df,
                         cols=c("Extremely high","High", "Intermediate", "Low", "None" ))
```

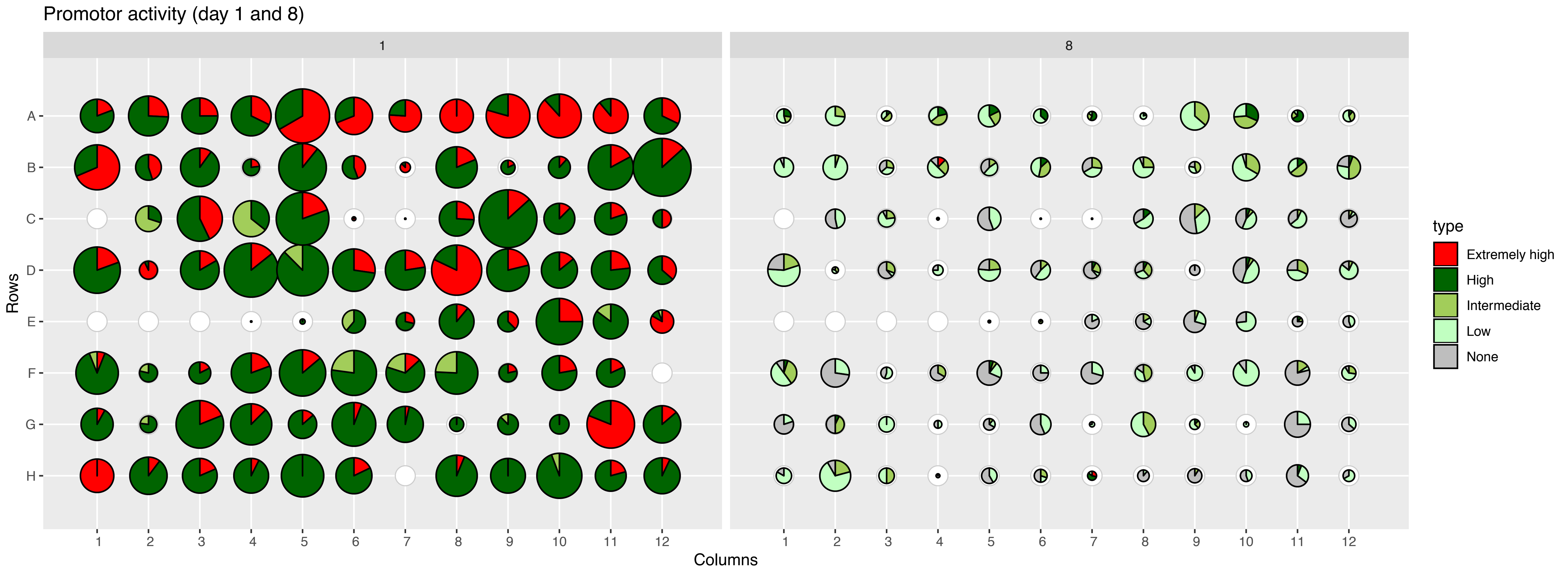
Tweak plot

```
p + scale_x_discrete(name ="Columns", limits=c(1:12)) +
  scale_y_discrete(name ="Rows", limits=lets) +
  scale_fill_manual(values = c("red","darkgreen","darkolivegreen3","darkseagreen1","grey")) +
  coord_fixed(ratio=12/12) + facet_grid(Day ~ .) +
  ggtitle("Promotor activity (day 1 and 8")
```


Position
Age
Intensity

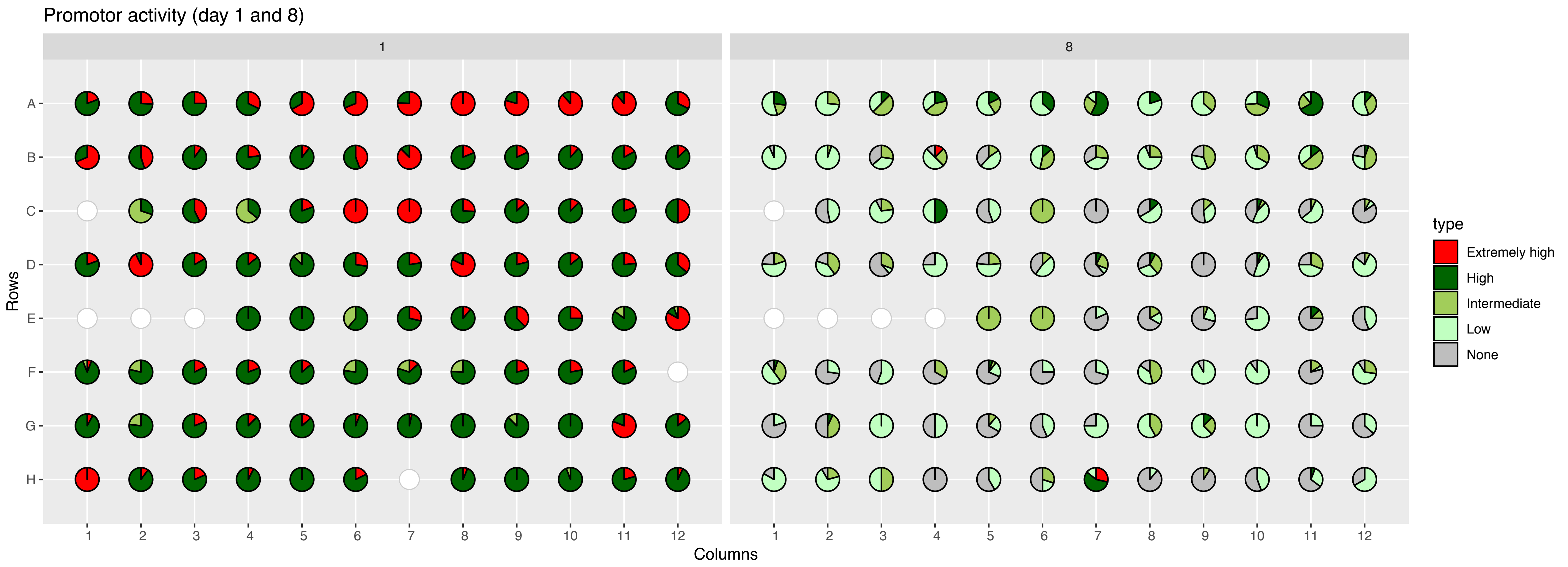
	A	B	C	D	E	F	G	H	I
1	Row	Column	Day	Extremely high	High	Intermediate	Low	None	Total
2	A	1	8	0	3	2	6	0	11
3	A	2	8	0	0	4	11	0	15
4	A	3	8	0	1	4	3	0	8
5	A	4	8	0	3	6	5	0	14
6	A	5	8	0	3	4	10	0	17
7	A	6	8	0	4	0	7	0	11
8	A	7	8	0	4	2	1	0	[illegible]
9	A	8	8	0	1	0	4	0	[illegible]

aes(x= Column, y= Row, **r = Total**, group = region)



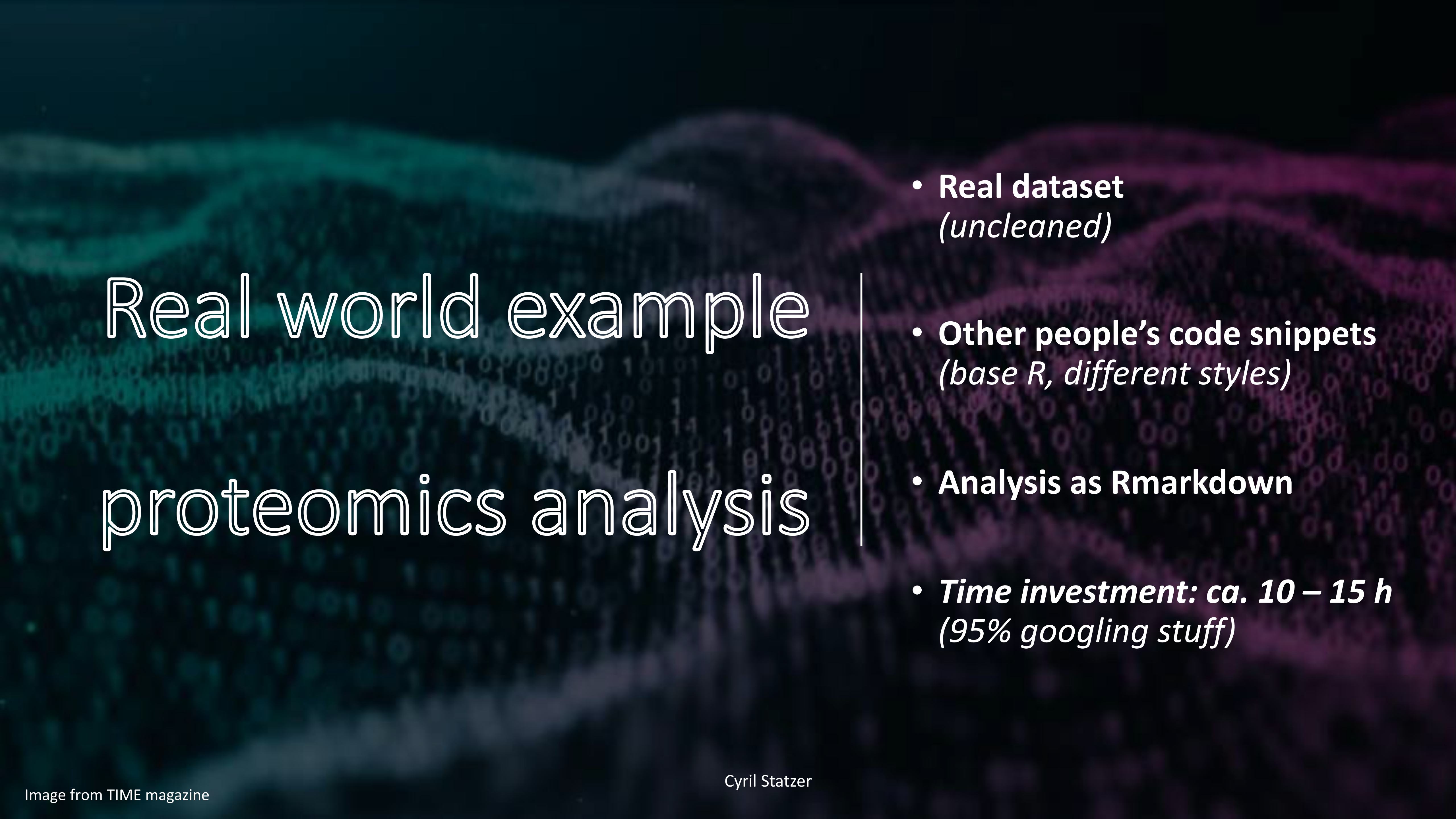
Cyril Statzer

aes(x= Column, y= Row, group = region)



Summary: R for data science

- R is a fantastic tool and a huge time saver
- While intimidating at first all scripts have common patterns that are very repetitive (think of Lego blocks)
- R makes your analysis reusable & reproducible
- You can get help everywhere!



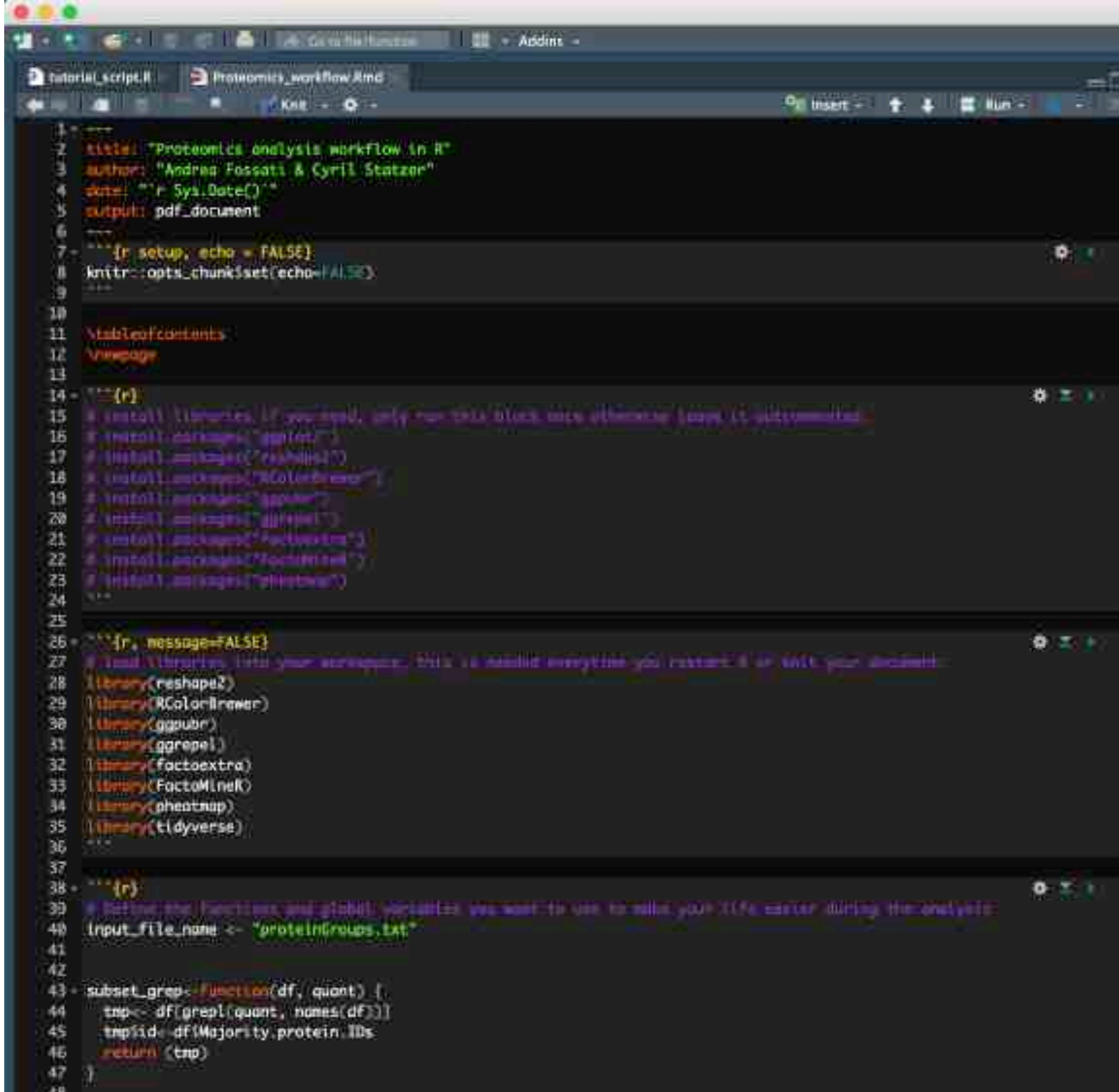
Real world example proteomics analysis

- **Real dataset**
  (uncleaned)
- **Other people's code snippets**
  (base R, different styles)
- **Analysis as Rmarkdown**
- ***Time investment: ca. 10 – 15 h***
  (95% googling stuff)

Image from TIME magazine

RMarkdown

- **Code is in the chunks** (light grey background)
- **Text in between chunks** (black background).
- **Hierarchy matters** like in a regular R script



~~~
---
title: "Proteomics analysis workflow in R"
author: "Andrea Fossati & Cyril Statzer"
date: "`r Sys.Date()`"
output: pdf_document
---
```{r setup, echo = FALSE}
knitr::opts_chunk$set(echo=FALSE)
~~~

\tableofcontents
\newpage

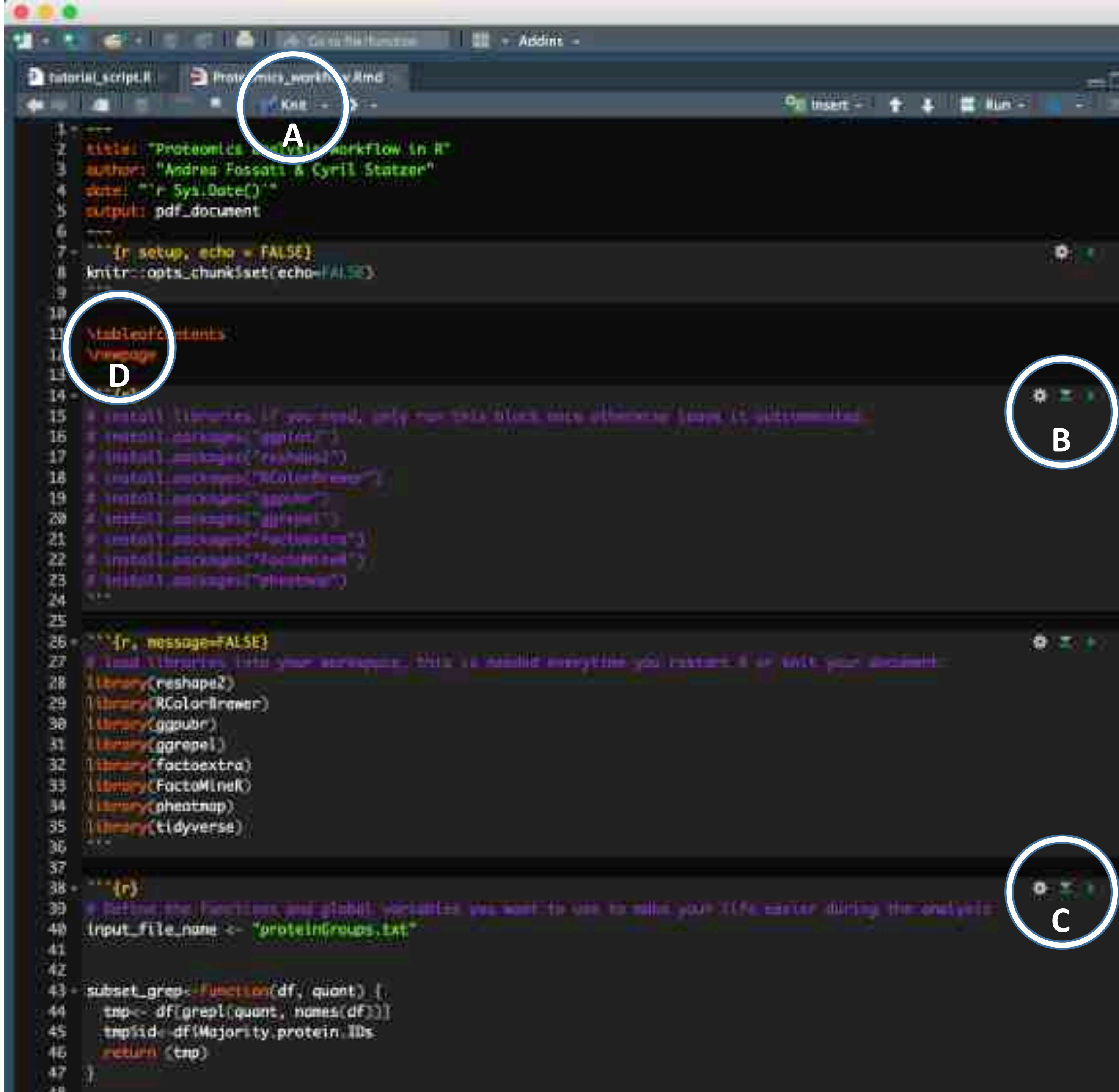
```{r}
# install libraries if you need, only run this block once otherwise leave it uncommented
# install.packages("ggplot2")
# install.packages("reshape2")
# install.packages("RColorBrewer")
# install.packages("ggpubr")
# install.packages("ggrepel")
# install.packages("factoextra")
# install.packages("FactoMineR")
# install.packages("pheatmap")
```

```{r, message=FALSE}
# load libraries into your workspace. this is needed everytime you restart R or knit your document
library(reshape2)
library(RColorBrewer)
library(ggpubr)
library(ggrepel)
library(factoextra)
library(FactoMineR)
library(pheatmap)
library(tidyverse)
```

```{r}
# Define the functions and global variables you want to use to make your life easier during the analysis
input_file_name <- "proteinGroups.txt"


subset_grep<-function(df, quant) {
  tmp<- df[grepl(quant, names(df))]
  tmp$id<-df$Majority.protein.IDs
  return (tmp)
}
```

RMarkdown

- **A:** Produces a html or pdf
- **B:** Evaluate one block by pressing play.
- **C:** evaluate all above by pressing the middle button.
- **D:** You can include LaTeX functions (for pdf)



~~~
---
title: "Proteomics analysis workflow in R"
author: "Andrea Fossati & Cyril Statzer"
date: "`r Sys.Date()`"
output: pdf_document
---
```{r setup, echo = FALSE}
knitr::opts_chunk$set(echo=FALSE)
~~~

\tableofcontents
\newpage

```{r}
# install libraries, if you need, only run this block once otherwise leave it outcommented
# install.packages("ggplot2")
# install.packages("reshape2")
# install.packages("RColorBrewer")
# install.packages("ggpubr")
# install.packages("ggrepel")
# install.packages("factoextra")
# install.packages("FactoMineR")
# install.packages("pheatmap")
```

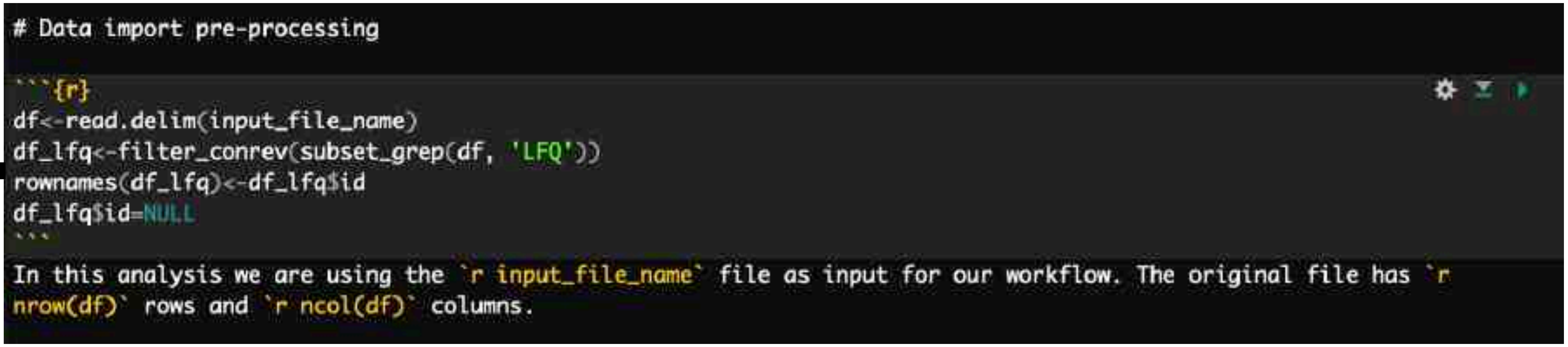
```{r, message=FALSE}
# load libraries into your workspace, this is needed everytime you restart R or knit your document
library(reshape2)
library(RColorBrewer)
library(ggpubr)
library(ggrepel)
library(factoextra)
library(FactoMineR)
library(pheatmap)
library(tidyverse)
```

```{r}
# Define the functions and global variables you want to use to make your life easier during the analysis
input_file_name <- "proteinGroups.txt"


subset_grep<-function(df, quant) {
  tmp<- df[grepl(quant, names(df))]
  tmp$id<-df$Majority.protein.IDs
  return (tmp)
}
```

ETH zürich life science zurich

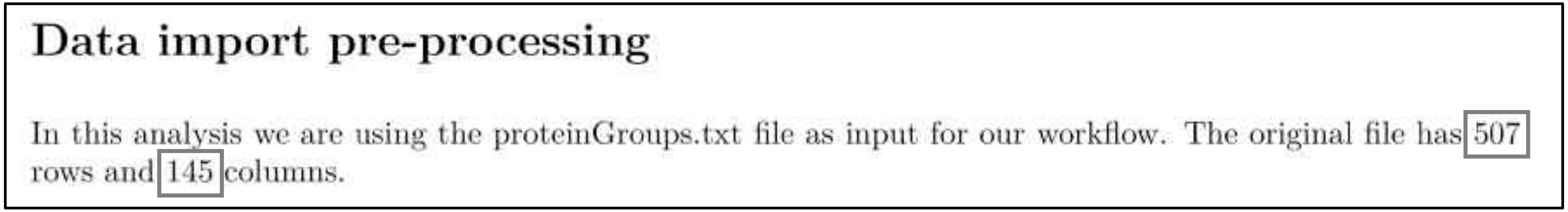
Markdown structure

- Break up the code into meaningful chunks.
- I would recommend one chunk for one logical step or one plot
- Add explanatory text so you (also in 6 months😉) and the reader of the document knows what you did.
- Use Hashtags to generate section headers (and other markup commands)
- You can also evaluate R code in the text.



~~~
# Data import pre-processing

```{r}
df<-read.delim(input_file_name)
df_lfq<-filter_conrev(subset_grep(df, 'LFQ'))
rownames(df_lfq)<-df_lfq$id
df_lfq$id=NULL
```
In this analysis we are using the `r input_file_name` file as input for our workflow. The original file has `r
nrow(df)` rows and `r ncol(df)` columns.
~~~

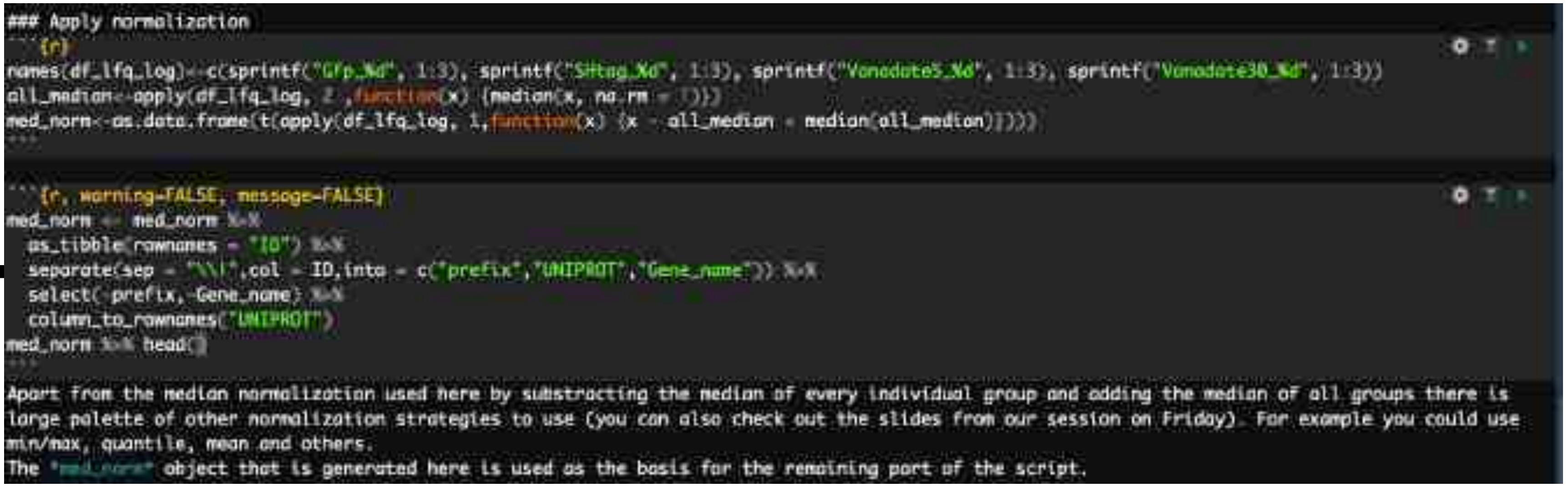
Knitted PDF



Data import pre-processing

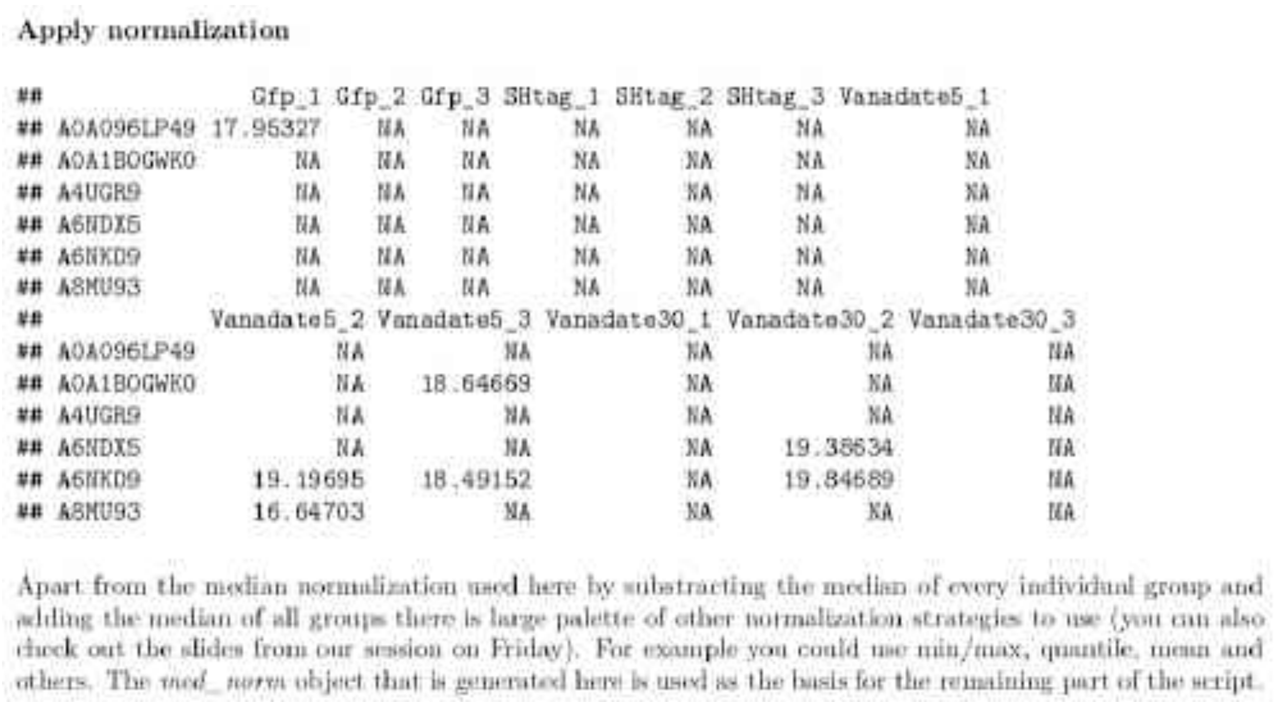
In this analysis we are using the proteinGroups.txt file as input for our workflow. The original file has 507 rows and 145 columns.

- This chunk imports the file and performs pre-processing
- The reader is informed about the dimensions of the file in the text



~~~
### Apply normalization
```{r}
names(df_lfq_log)<-c(sprintf("Gfp_%d", 1:3), sprintf("SHtag_%d", 1:3), sprintf("Vanadate5_%d", 1:3), sprintf("Vanadate30_%d", 1:3))
all_median<-apply(df_lfq_log, 2 ,function(x) {median(x, na.rm = T)})
med_norm<-as.data.frame(t(apply(df_lfq_log, 1,function(x) {x - all_median + median(all_median)})))
~~~

```{r, warning=FALSE, message=FALSE}
med_norm <- med_norm %>%
  as_tibble(rownames = "ID") %>%
  separate(sep = "\\|",col = ID,into = c("prefix","UNIPROT","Gene_name")) %>%
  select(-prefix,-Gene_name) %>%
  column_to_rownames("UNIPROT")
med_norm %>% head()
```

Apart from the median normalization used here by substracting the median of every individual group and adding the median of all groups there is
large palette of other normalization strategies to use (you can also check out the slides from our session on Friday). For example you could use
min/max, quantile, mean and others.
The "med_norm" object that is generated here is used as the basis for the remaining part of the script.
```

Knitted PDF



Apply normalization

```
##                Gfp_1 Gfp_2 Gfp_3 SHtag_1 SHtag_2 SHtag_3 Vanadate5_1
## A0A096LP49 17.95327    NA    NA      NA      NA      NA          NA
## A0A1B0GWK0       NA    NA    NA      NA      NA      NA          NA
## A4UGR9           NA    NA    NA      NA      NA      NA          NA
## A6NDX5           NA    NA    NA      NA      NA      NA          NA
## A6NKD9           NA    NA    NA      NA      NA      NA          NA
## A8MU93           NA    NA    NA      NA      NA      NA          NA
##            Vanadate5_2 Vanadate5_3 Vanadate30_1 Vanadate30_2 Vanadate30_3
## A0A096LP49          NA          NA           NA           NA           NA
## A0A1B0GWK0          NA    18.64669           NA           NA           NA
## A4UGR9              NA          NA           NA           NA           NA
## A6NDX5              NA          NA           NA     19.38634           NA
## A6NKD9        19.19695    18.49152           NA     19.84689           NA
## A8MU93        16.64703          NA           NA           NA           NA
```

Apart from the median normalization used here by substracting the median of every individual group and adding the median of all groups there is large palette of other normalization strategies to use (you can also check out the slides from our session on Friday). For example you could use min/max, quantile, mean and others. The *med_norm* object that is generated here is used as the basis for the remaining part of the script.
```

# Concept of missing values (NA)

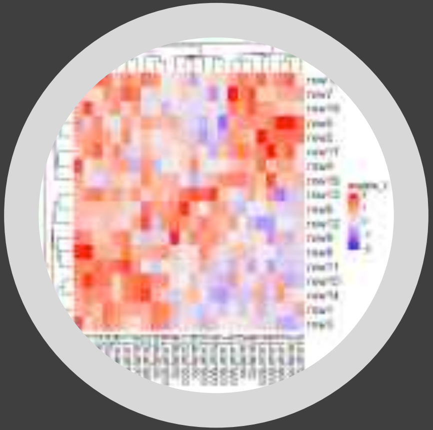
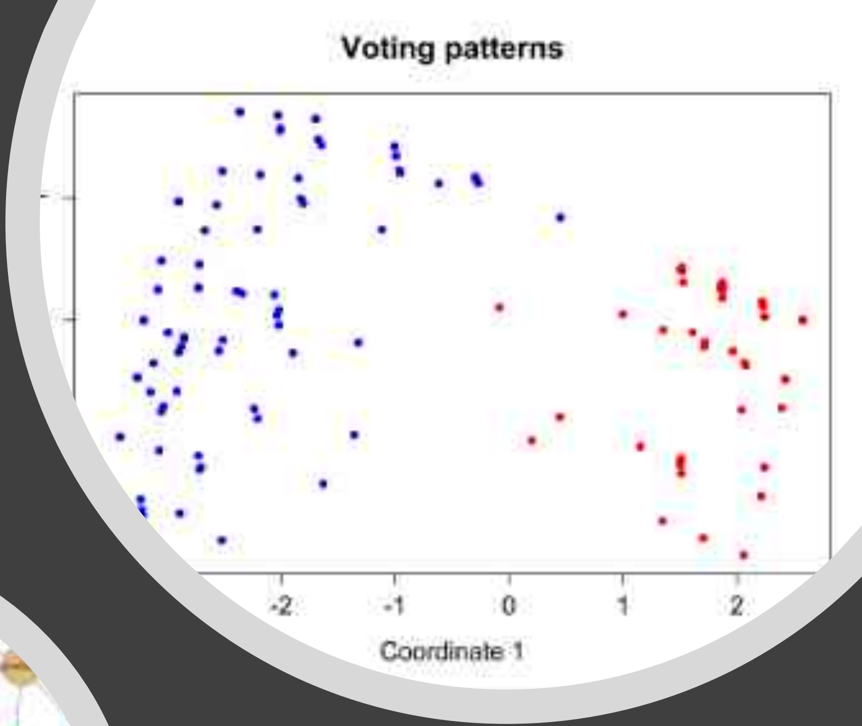
- R represents missing values by the symbol **NA** (**N**ot **A**vailable)

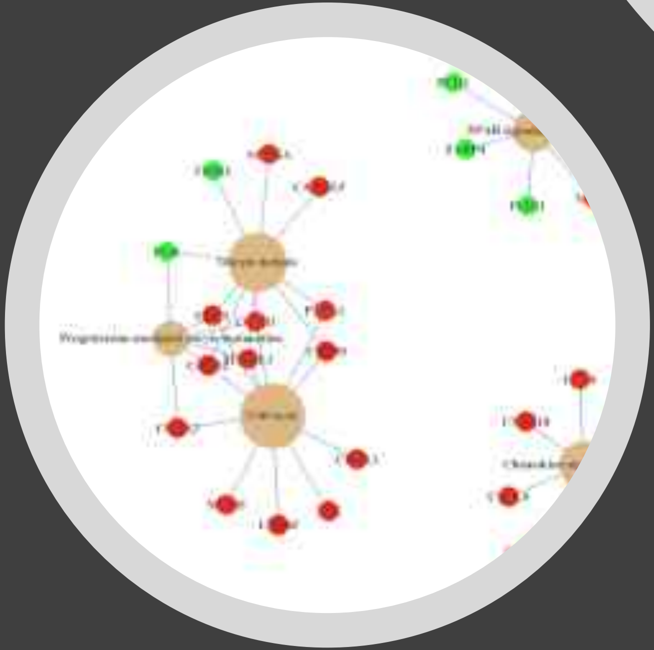
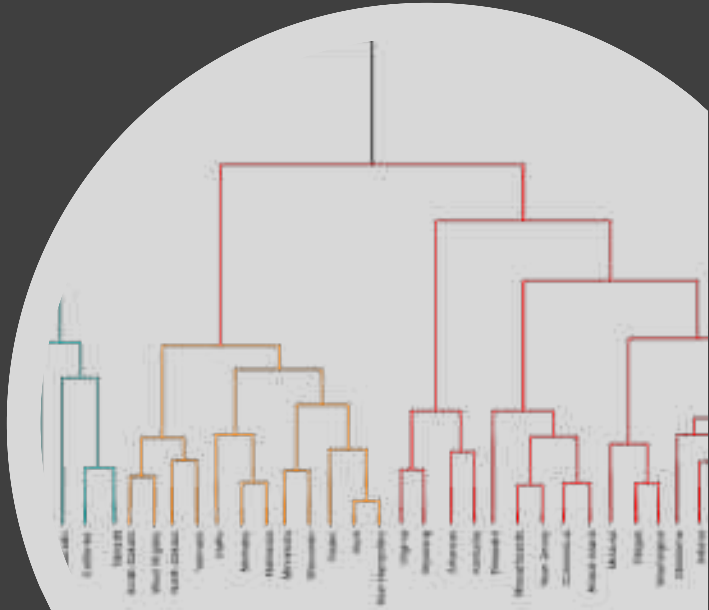
```
y <- c(1,NA,3)

y %>% is.na()
FALSE, TRUE, FALSE

y %>% mean()
NA

y %>% mean(na.rm = TRUE)
2
```

Visualizing data

- Heatmaps
- KEGG pathways
- MDS, Hierarchical clustering
- PCA

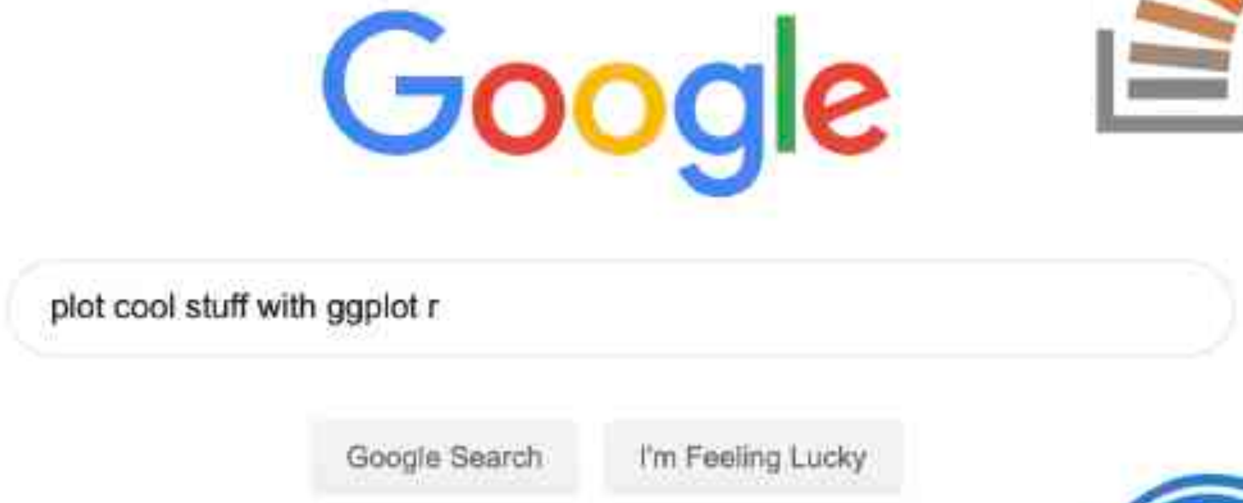
- … see last section of the markdown «Example of potential addition»

Outlook

Online resources **for specific questions**


plot cool stuff with ggplot r

stackoverflow

https://stackoverflow.com

STHDA

Statistical tools for high-throughput data analysis

http://www.sthda.com/english/

Online resources **for learning**

DataCamp

<30$ per month for all courses!

https://www.datacamp.com

Data Visualization Cheat Sheet

The ggplot2 package lets you make beautiful and customizable plots of your data. It implements the grammar of graphics, an easy to use system for building plots. See docs.ggplot2.org for detailed examples. Updated 11/16

DOWNLOAD

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

The best book is for free:
https://r4ds.had.co.nz