Analysing data using R

A gentle guide into the R world

...Where we are

About me

Arnau Sangrà Rocamora

- 🎂 Born on May 10, 1992 (Barcelona)
- 🏡 3 at home (sister & brother)
- 💼 Data Engineer @ Qustodio
- 🎓 Msc. CyberSecurity Management - UPC
- 🎓 Informatics Engineering - FIB UPC
- 🖥️ 🍿 🎥 🏍️ 🏔️ 🏝️ 🛬 🗺️

Working Agreements

Promote safe and brave learning environment

- Celebrate our failures and our wins
- No judgment
- Respect for each other
- Interrupt when needed (and use Slack whenever needed)
- Keep a positive & constructive attitude
- Ask before our info on social networks (i.e., check for privacy needs)
- Have fun
- Self-organizing silence, by raising our hands
- Fred joins to help us improve our students' experience

The R swiss army knife

Milestones

1975

John Chambers creates ***S***

1992

Start its conception

2000

First **stable release**

1995

Public release

July 2018

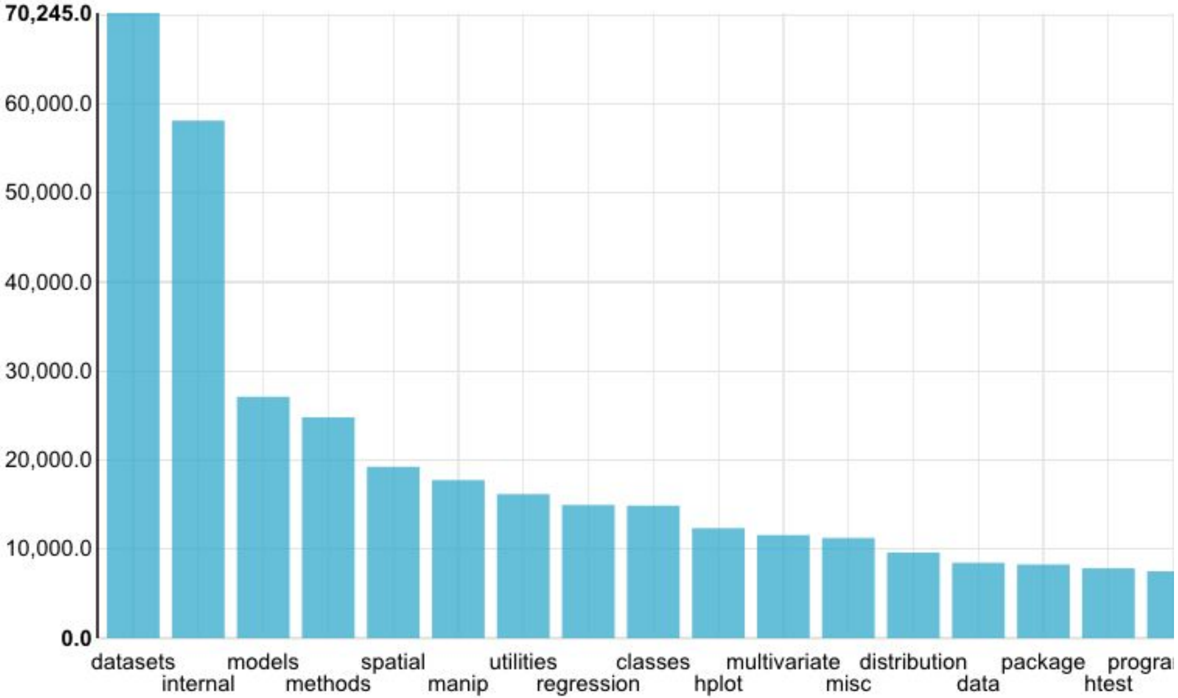
Feather Spray released, version **3.5.1**

Being around for more than 20 years, R has a very active community.

There are more than 16,395 published packages available in CRAN.

Source: https://www.rdocumentation.org/trends

Top keywords for packages

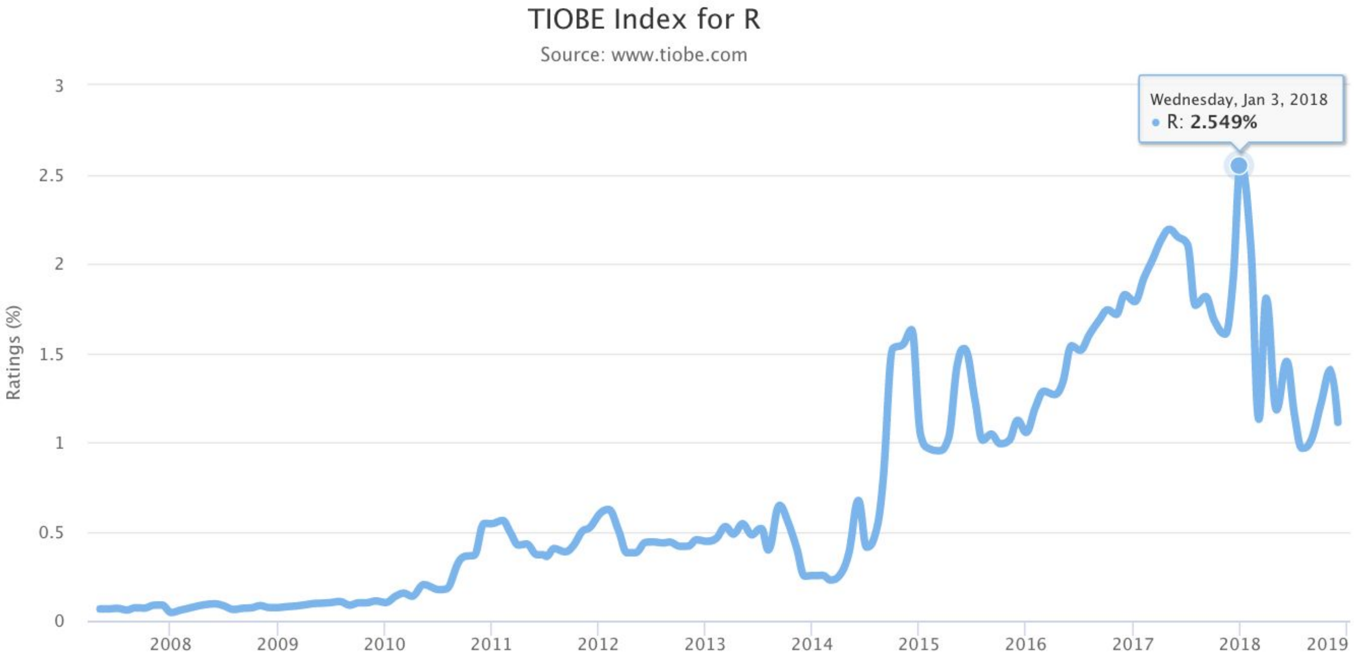


Source: https://www.rdocumentation.org/trends

8th most popular PL

on 2017 according to TIOBE.

Long term history



Homework

As you might have guess, most famous alternative is Python.

Nevertheless, there are far more programming languages designed for data analysis.

What about R's most important competitors?

- ❖ Do some research on the internet and create a MarkDown report including details, quirks and comparative table for one of the Data Analysis PL alternatives to **R**.

The essentials of R

Language basics

"The *best* thing about **R** is that it was written by statisticians, the *worst* thing about **R** is that it was written by statisticians."

Anonymous

Data Types Recap

Atomic

- Numeric
- Character
- Boolean
- Factor

Compound

- Array / Vector
- List
- Matrices
- DataFrame

```
v1 <- c(5, 11, 13, 17, 23)                # Vector/array de “numerics”

m1 <- matrix(1:12, ncol = 3, nrow = 4)    # Matriz 4x3

d1 <- data.frame(Col1 = c(1,3,5,7,9), Col2 = c("a","b","c","d","e"), Col3 = c(T, F, F, T, T), stringsAsFactors = FALSE)

d1[d1$Col1 == 9 | d1$Col3 == FALSE,]      # Filter dataframe by rows and columns
```

Operators & Vectorization

Arithmetic

❖ +, -, *, /, %, ^, sqrt, ...

Logical

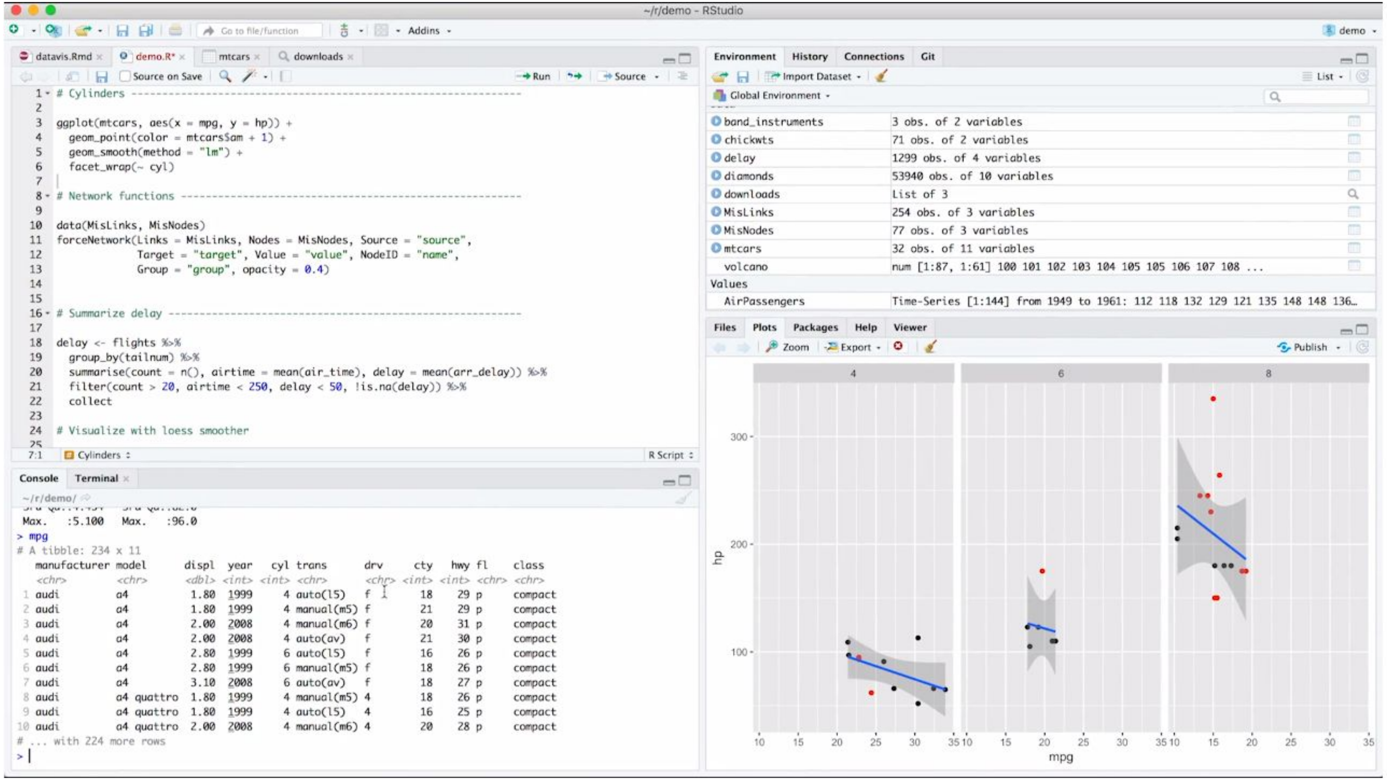
❖ !, &, &&, |, ||, xor, all, any, %in%, isTRUE(), ...

Functional

❖ sapply(), lapply(), mapply(), vectorize()

```r
v <- c(5, 11, 29, 37, 51)

sapply(v, function(elem) {
    if (elem %% 10 > 3) elem^3
    else 0
})

m <- matrix(1:12, ncol = 3, nrow = 4)
m * 4
apply(m, 1, sum)
```

RStudio IDE


~/r/demo - RStudio
Go to file/function
Addins
demo
datavis.Rmd
demo.R*
mtcars
downloads
Source on Save
Run
Source
Cylinders
ggplot(mtcars, aes(x = mpg, y = hp)) +
geom_point(color = mtcars$am + 1) +
geom_smooth(method = "lm") +
facet_wrap(~ cyl)
# Network functions
data(MisLinks, MisNodes)
forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
Target = "target", Value = "value", NodeID = "name",
Group = "group", opacity = 0.4)
# Summarize delay
delay <- flights %>%
group_by(tailnum) %>%
summarise(count = n(), airtime = mean(air_time), delay = mean(arr_delay)) %>%
filter(count > 20, airtime < 250, delay < 50, !is.na(delay)) %>%
collect
Visualize with loess smoother
7:1
Cylinders
R Script
Console
Terminal
~/r/demo/
Max. :5.100 Max. :96.0
> mpg
A tibble: 234 x 11
manufacturer model displ year cyl trans drv cty hwy fl class
1 audi a4 1.80 1999 4 auto(l5) f 18 29 p compact
2 audi a4 1.80 1999 4 manual(m5) f 21 29 p compact
3 audi a4 2.00 2008 4 manual(m6) f 20 31 p compact
4 audi a4 2.00 2008 4 auto(av) f 21 30 p compact
5 audi a4 2.80 1999 6 auto(l5) f 16 26 p compact
6 audi a4 2.80 1999 6 manual(m5) f 18 26 p compact
7 audi a4 3.10 2008 6 auto(av) f 18 27 p compact
8 audi a4 quattro 1.80 1999 4 manual(m5) 4 18 26 p compact
9 audi a4 quattro 1.80 1999 4 auto(l5) 4 16 25 p compact
10 audi a4 quattro 2.00 2008 4 manual(m6) 4 20 28 p compact
... with 224 more rows
Environment
History
Connections
Git
Import Dataset
List
Global Environment
band_instruments 3 obs. of 2 variables
chickwts 71 obs. of 2 variables
delay 1299 obs. of 4 variables
diamonds 53940 obs. of 10 variables
downloads List of 3
MisLinks 254 obs. of 3 variables
MisNodes 77 obs. of 3 variables
mtcars 32 obs. of 11 variables
volcano num [1:87, 1:61] 100 101 102 103 104 105 105 106 107 108 ...
Values
AirPassengers Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136…
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
4
6
8
hp
mpg

The cause of all

We work for a successful tech start-up that offers guided visits to wineries.

Business sells touristic experience packs around many localities and relies mainly on the web and online marketing to acquire customers.

Company’s web is brand new, simple and maintainable.

But more and more departments are gaining interest to know metrics, KPIs to become data driven.

We work for a successful tech start-up that relies mainly on the web and online marketing to acquire customers.

Company’s web is brand new, simple and maintainable.

But more and more departments are gaining interest to know metrics, KPIs to become data driven.

As the main (and only one) junior data analyst in the company you are required to deliver requested information.

Load data 101

Surviving data formats

"It is a capital mistake to theorize before one has data."

Arthur Conan Doyle

1. Goals

Be able to load data to the working environment.

➔ **Basic**
Loads data without prior check.

➔ **Advanced**
Studies data and uses common parameters.

➔ **Expert**
Careful examination. Knows advanced options to ease further steps

Common data formats

- RAW
- CSV
- XML
- JSON
- PARQUET

Load data functions

RAW or Tabular origins (CSV, TSV, etc)

Depending on the source, different importing functions to consider.

Start with functions from *base*

```
df <- read.csv(file = "/path/to/file.csv", header = T, skip = 2, check.names = T, stringsAsFactors = T)
```

Alternatively advanced functions from other packages (readr, data.table)

```
df <- data.table::fread(file = "/path/to/file", header = F, stringsAsFactors = F, showProgress = TRUE)
df <- readr::read_log("/path/to/file")
```

Load data functions II

Semistructured: JSON

Plenty of packages help importing sources into a list, objects or dataframes

```
# Load the package required to read JSON files.
library("rjson")


result <- rjson::fromJSON(file = "input.json")    # Give the input file name to the function.
json_data_frame <- as.data.frame(result)          # Convert JSON file to a data frame.
```

Load data functions III

Hierarchical Formats: XML

Parse data and apply XPath expressions to extract selected fields

```r
library("XML")
result <- XML::xmlParse(file = "input.xml")              # use internal nodes i.e. compatible with xpath
xmlfile <- XML::xmlTreeParse(file = "/path/to/file.xml")  # use xml functions to extract content

topxml <- XML::xmlRoot(xmlfile)                           # get root element
topxml <- XML::xmlSApply(topxml, function(x) XML::xmlSApply(x, XML::xmlValue))
df <- XML::xmlToDataFrame(result)
```

Exploration Functions

Basic exploration

Glimpse at data to inspect source and spot loading errors

```
names(df)
dim(df)
class(df$Col1)
unique(df$Col23)
length(unique(df$Name))
summary(df)
```

Remember S3?

Operations team has recently enabled logging on the company’s website.

Collected data is still very basic and far from actionable. All data is stored in the storage service from chosen cloud provider: AWS.

Additionally, data is not stored in rotating log files nor collected in centralized destination but in per-request file.

Operations team has recently enabled logging on the company's website.

Collected data is still very basic and far from actionable. All data is stored in the storage service from chosen cloud provider: AWS.

Additionally, data is neither stored in rotating log files nor collected in centralized destination but in per-request file.

You are responsible for the data acquisition and loading into your working environment.

Inspect existing log files and proceed to its retrieval. Load all the available files into a single data frame to be analysed.

You are responsible for the data acquisition and loading into your working environment.

Inspect existing log files and proceed to its retrieval. Load all the available files into a single data frame to be analysed.

Log in to S3 console

s3://logs.bdatainstitute.com

Load files in RStudio

Tip

Don't let **R** stole your heart & mind.

Other tools can be far superior for some tasks (Unix tools?)

cat, sed, awk, tr, cut?

Share Results & Conclusions

Loading data is a *fundamental* step in the whole data analysis cycle.

A thorough load ensures that all records are taken into account saving countless time.

I don't understand at all

I need to go over this again

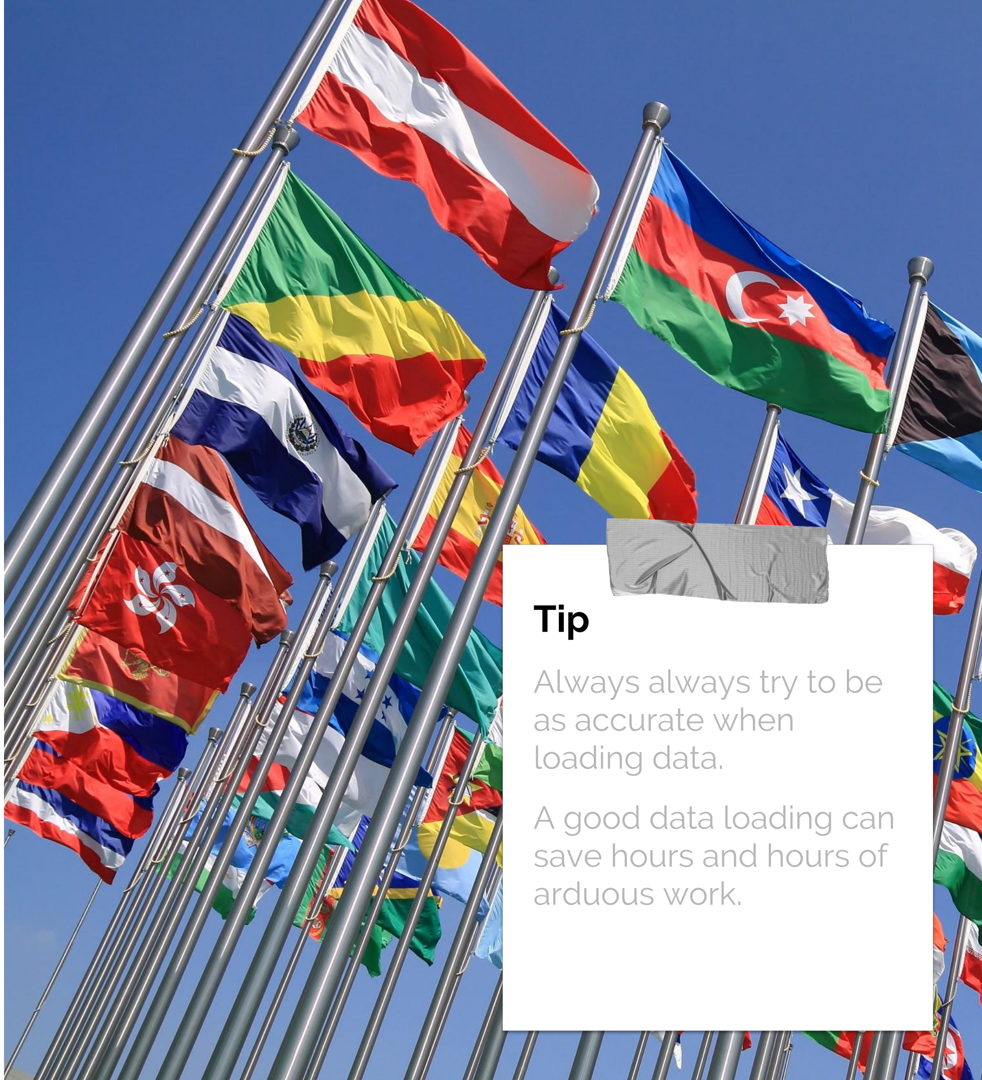
I think I got it, but am not completely comfortable

I got it

I can explain it to someone else

I can **successfully load** the data I want to analyse into the environment.

From here on I can **do some basic exploration** on imported data.



Tip

Always always try to be as accurate when loading data.

A good data loading can save hours and hours of arduous work.

Happiness Door time!

I don’t think this is for me.

Can’t we do it all again?

I don’t see how to apply that…

I’m still not convinced this is going to be useful.

I’m satisfied with my learnings so far.

Break!

Tidy up!

Clean your data

"Not everything that can be counted counts, and not everything that counts can be counted."

Albert Einstein, Physicist

2. Goals

Be able to clean and transform untidy data into a cleanly arrangement dataset:

- ➔ **One row per observation**
  Each entry relates to an entire collection of measured attributes

- ➔ **One column per field**
  Each field is contained within its own column, cell.

Data Frames Manipulation

As native types, dataframes can be filtered without requiring extra libraries

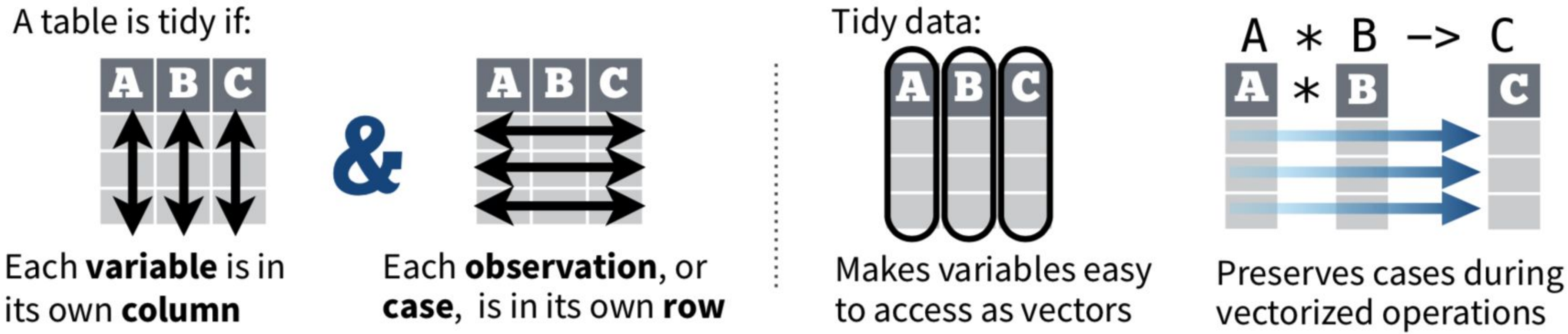
```
subset.columns <- c("Name", "Status", "Description")
df.subset3 <- df[names(df) %in% subset.columns]      # filter by columns


df$Comments[df$Name == "user@domain"]               # filter by row
df$Comments <- NULL                                 # drop column
df$Environment <- "production"                      # new column
df.phase.na <- df[is.na(df$Phase), ]                # NA values


df.by.name <-  df[order(df$Name, na.last = TRUE, decreasing = FALSE),]
```

Tidyverse

A collection of essential packages that infinitely ease the data manipulation

- ❖ dplyr: data analysis
- ❖ tidyr: data tidying
- ❖ readr: load data
- ❖ purr: enhance functional
- ❖ ggplot2: graphics generation

Tidy Data


A table is tidy if:
A B C
&
A B C
Each **variable** is in its own **column**
Each **observation**, or **case**, is in its own **row**
Tidy data:
A B C
Makes variables easy to access as vectors
A * B -> C
A * B C
Preserves cases during vectorized operations

tidyr

Very often, data load results in columns containing many fields collated

```
df <- data.frame( fact_type = c('user_visit', 'user_regitration', "user_visit"),
                  uid = c( '1b5a794a0e68ea69ef', '9fb9b32f61b2b3ce01', 'b5bf20c31c3f392aab' ),
                  date = c("2012/12/12", "2015/12/25", "2016/03/14"),  stringsAsFactors = F)

df2 <- tidyr::separate(df, date, c("y", "m", "d"), sep = "/")
```

	fact_type	uid	date
1	user_visit	1b5a794a0e68ea69ef	2015/12/12
2	user_regitration	9fb9b32f61b2b3ce01	2015/12/25
3	user_visit	b5bf20c31c3f392aab	2016/03/14

	fact_type	uid	y	m	d
1	user_visit	1b5a794a0e68ea69ef	2015	12	12
2	user_regitration	9fb9b32f61b2b3ce01	2015	12	25
3	user_visit	b5bf20c31c3f392aab	2016	03	14

tidyr

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE**)**

gather() moves column names into a **key** column, gathering the column values into a single **value** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

gather(table4a, `1999`, `2000`, key = "year", value = "cases")

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL**)**

spread() moves the unique values of a **key** column into the column names, spreading the values of a **value** column across the new columns.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

key value

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

spread(table2, type, count)

When source is raw or semistructured..

At the company, there is no CDO. Thus, data is still immature.

Having overcome the initial acquisition phase, next pitfall comes with the initial exploration:

Data is far from the having the desired structure.

- Malformed columns
- Wrong data types
- Fields spread embedded or spread across multiple columns

At the company, there is no CDO. Thus, data is still immature.

Having overcome the initial acquisition phase, next pitfall comes with the initial exploration:

Data is far from the having the desired structure.

- Malformed columns
- Wrong data types
- Fields spread embedded or spread across multiple columns

Tidy data so that it is arranged in a actionable format and you can start your assignment.

Share Results & Conclusions

Without clean and tidy data it is unlikely to obtain any good results in further analysis.

Having data properly arranged eases the application of functions and algorithms.

I don't understand at all

I need to go over this again

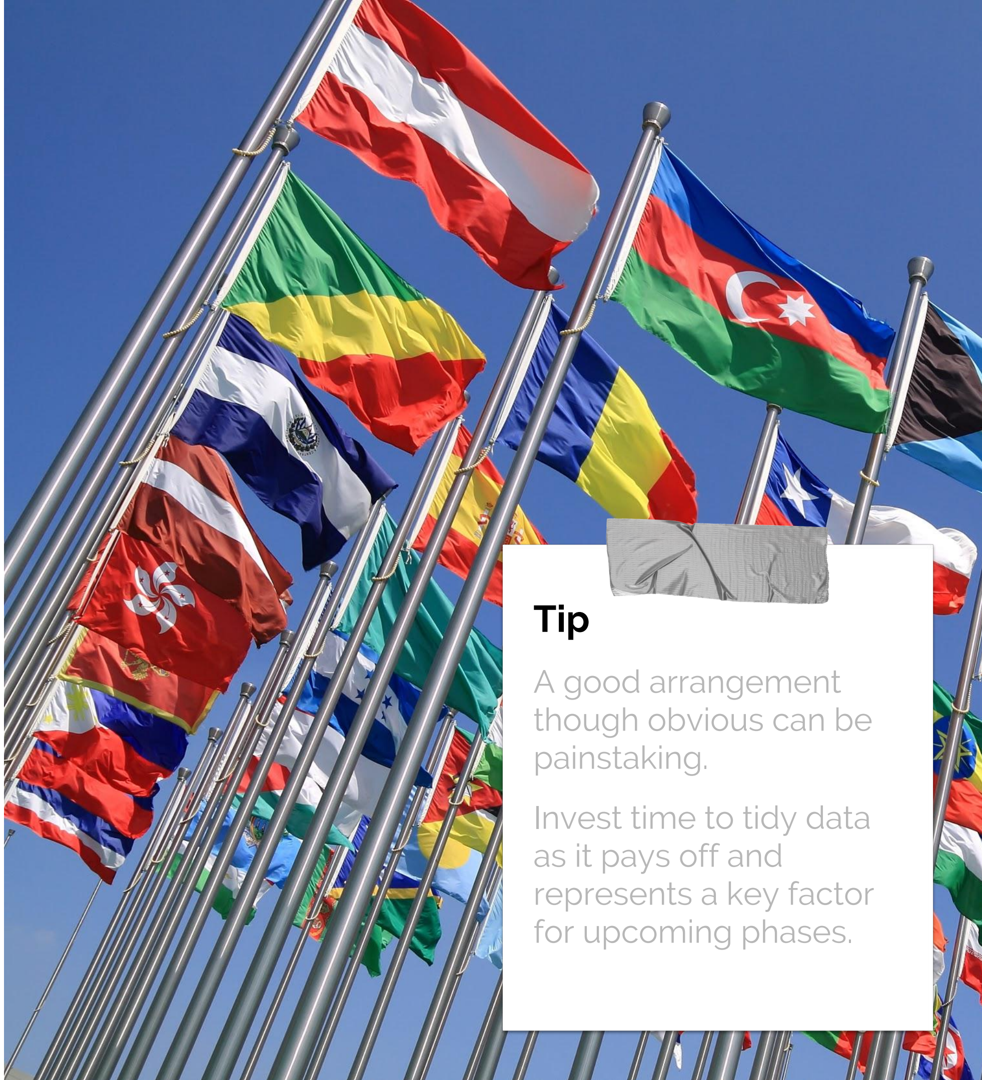
I think I got it, but am not completely comfortable

I got it

I can explain it to someone else

I can cleanly arrange data so that further analysis is straightforward.

I reckon data tidying as a fundamental step towards quality results.



Tip

A good arrangement though obvious can be painstaking.

Invest time to tidy data as it pays off and represents a key factor for upcoming phases.

Homework

Typical tidying tasks include:

- Column type adjustment/conversion
- Reshape of rows/columns
- Extraction of entangled fields

Ok, that was the minimum enough cleaning to start with data processing.

❖ Discuss which other changes could we need to apply to loaded source so that it become the perfect dataset.

Break!

Getting to the results

Crunching information

"Errors using inadequate data are much less than those using no data at all."

Charles Babbage

3. Goals

Application of rather basic functions is often enough to answer business demands.

➔ **Deep dive into data**
Deliver data within the context of a story you've already told

➔ **Aggregate**
Make big numbers digestible by putting them in the context of something familiar

dplyr

Transform, filter, aggregate or data

```
purchases <- data.frame(date = c("2015/12/12", "2015/12/25", "2016/03/14"),
                        uid = c( '794a0e68ea69ef', '9b32f61b2b3ce01', 'f20c31c3f392aab'),
                        amount = c(25.99, 54.99, 77.99),
                        discount = c(0, 5, 10), stringsAsFactors = F)
purchases <- dplyr::mutate(purchases, total = (amount * (100 - discount)) / 100)
```

	date	uid	amount	discount
1	2015/12/12	1b5a794a0e68ea69ef	25.99	0
2	2015/12/25	9fb9b32f61b2b3ce01	54.99	5
3	2016/03/14	b5bf20c31c3f392aab	77.99	10

	date	uid	amount	discount	total
1	2015/12/12	1b5a794a0e68ea69ef	25.99	0	25.9900
2	2015/12/25	9fb9b32f61b2b3ce01	54.99	5	52.2405
3	2016/03/14	b5bf20c31c3f392aab	77.99	10	70.1910

dplyr

Perform almost any type of data transformation

Data Transformation with dplyr : : CHEAT SHEET

dplyr

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes

x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

group_by(.data, ..., add = FALSE)
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …)
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)
ends_with(match)
matches(match)
num_range(prefix, range)
one_of(…)
starts_with(match)
:, e.g. mpg:cyl
-, e.g, -Species

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns.
rename(iris, Length = Sepal.Length)

RStudio

Everyone needs analytics

Backend and Marketing are the first departments that approached you with its inquiries.

Head of Backend is concerned about scalability issues regarding the website. She personally asked about users and their interaction.

On the other hand, whilst having less priority Marketing team is willing to improve acquisition with new locally focused campaigns.

Backend and Marketing are the first departments that approached you with its inquiries.

Head of Backend is concerned about scalability issues regarding the website. She personally asked about users and their interaction.

On the other hand, whilst having less priority Marketing team is willing to improve acquisition with new locally focused campaigns.

Provide some insights regarding the users of the website. How many are they? How many pages do they visit on average? From where do they come?

Share Results & Conclusions

Too many times, business questions do not really require of very advanced transformation and analysis techniques.

I don't understand at all

I need to go over this again

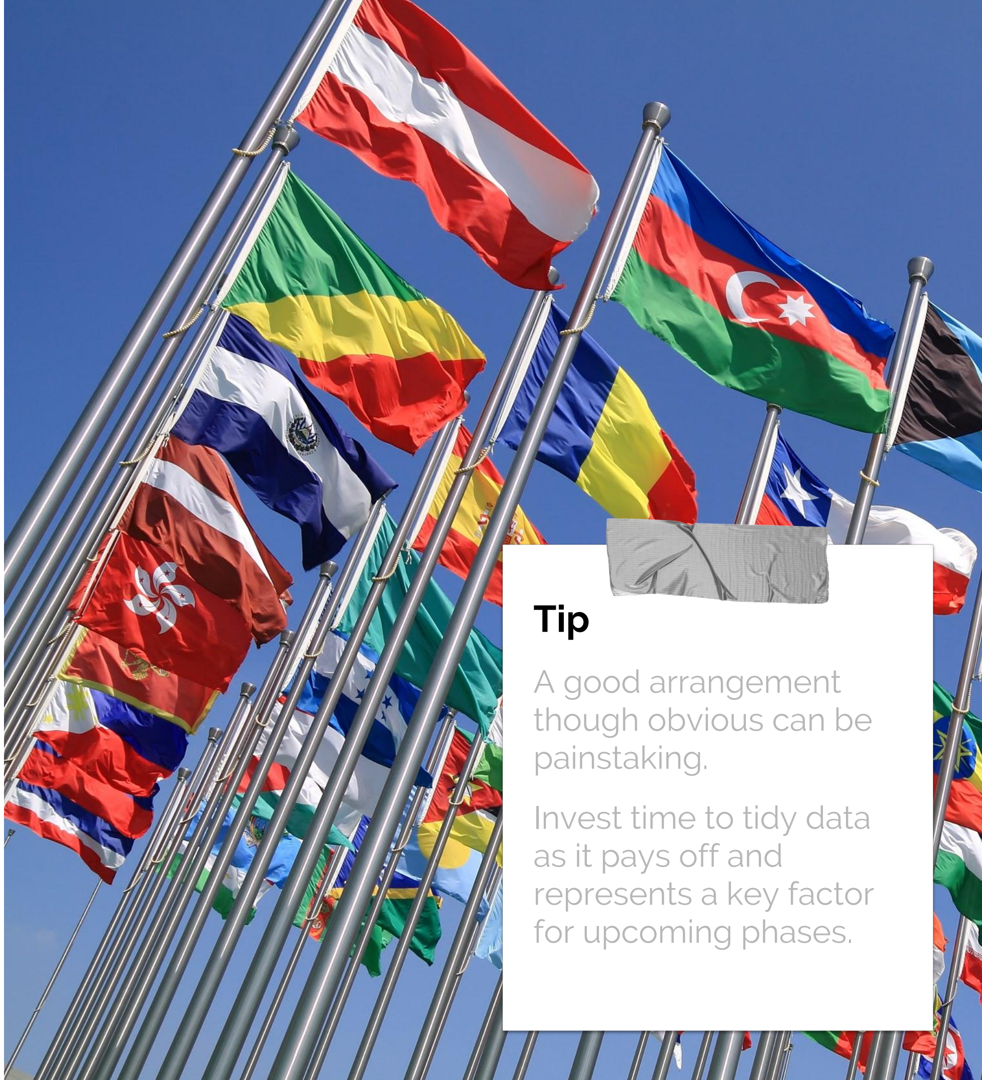
I think I got it, but am not completely comfortable

I got it

I can explain it to someone else

I can use R as an analysis tool to extract insights concealed in data.

I feel comfortable with data manipulation and underlying language quirks.



Tip

A good arrangement though obvious can be painstaking.

Invest time to tidy data as it pays off and represents a key factor for upcoming phases.

Homework

Hint: there are plenty of services that provide data sources or APIs to locate users.

Some of these, though free, use a freemium strategy using rates or quotas.

Backend department requests more insights.

❖ What time do we have more requests?
❖ Which is the most downloaded resource?

After an intense meeting with Marketing, they tell you its plan to boost their locally focused campaign.

❖ Can we apply clustering to know more about our visitors?

It's all about communication

Transmit the results

"The greatest value of a picture is when it forces us to notice what we never expected to see."

John Tukey

4. Goals

People need to understand how you came up to the results. Reproduce your findings:

- ➔ **Display conclusions**
  Deliver data within the context of a story you've already told
- ➔ **Reproducible Research**
  Analysis should be well documented, repeatable.
- ➔ **Effortlessly**
  Create automated reports at minimum cost.

The data analysis cycle

Always include the final report all the phases of the cycle.

1

Definition

Goals must be clear so that quest for the answers can be accomplished without doubt

2

Acquisition

Define necessary data to resolve the goal. Sometimes provided, other times it has to be fetched or extracted.

3

Cleansing

Seldom data is arranged ideally for analysis. Convert, transform or subset as necessary.

4

Processing

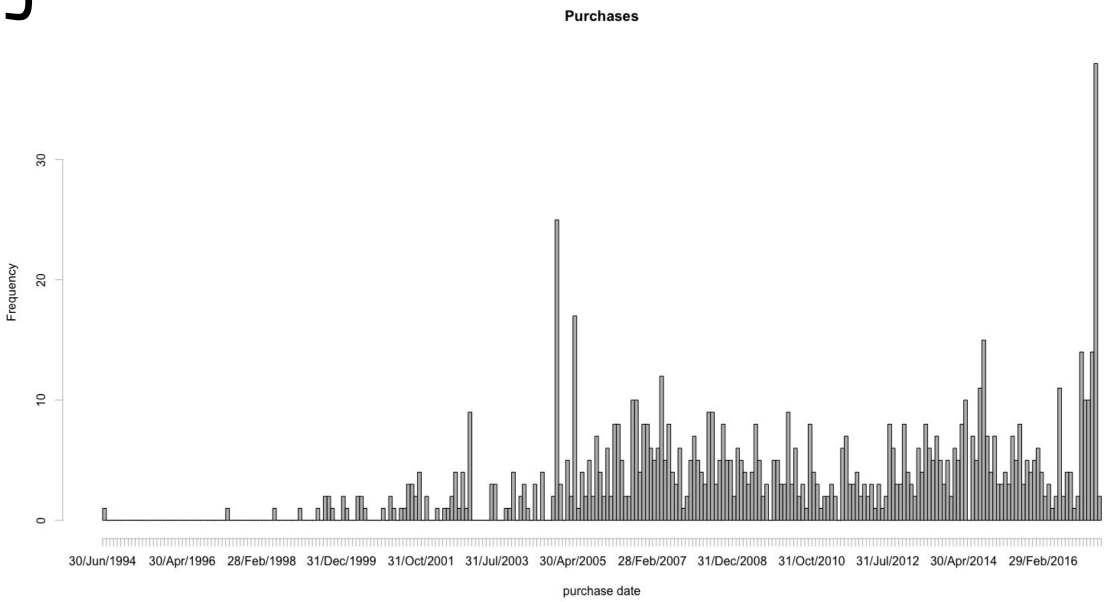
Analysis. Crunch data in order to obtain results that answer the demands of business.

5

Documentation

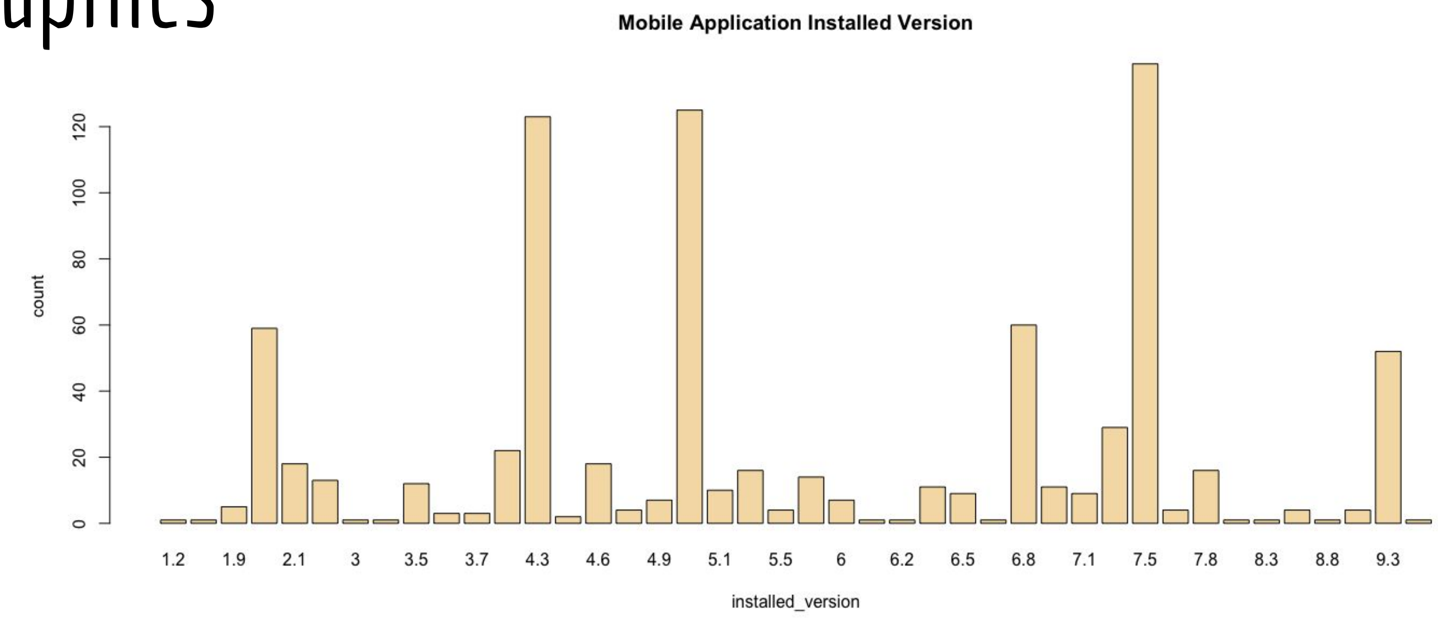
Results are not useful unless are properly documented thus, reproducible

Graphics




```r
hist(x = purchases$date[!is.na(purchases$date)], col = "gray",
  breaks = "month", format = "%d/%b/%Y", freq = T,
  main = "Purchases", xlab = "purchase date")
```

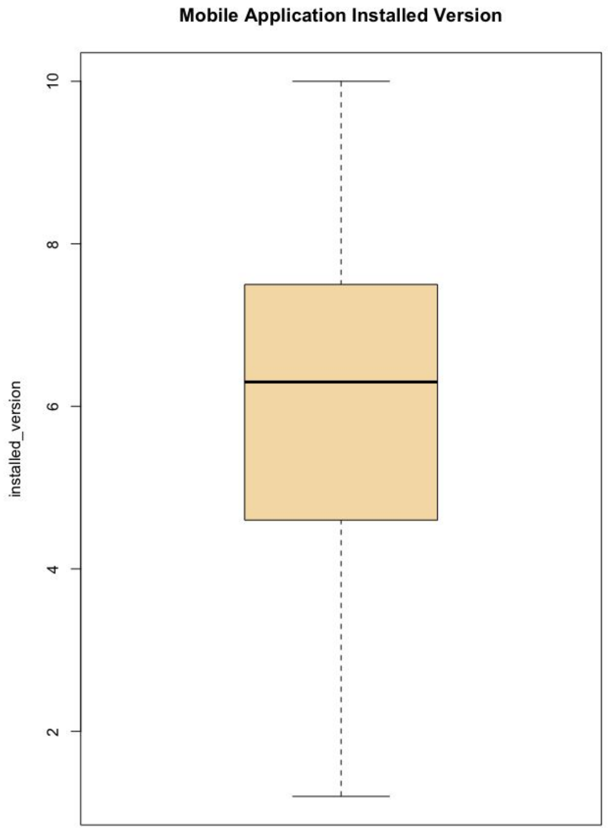
Graphics




```
barplot(height = table(purchases$price), main = "Mobile Application Installed Version",
        xlab = "installed_version", ylab = "count",  col = "wheat")
```

Graphics

```
boxplot(purchases$price,
    main = "Mobile Application Installed Version",
    ylab = "installed_version",
    col = "wheat")
```



ggplot2 (qplot)

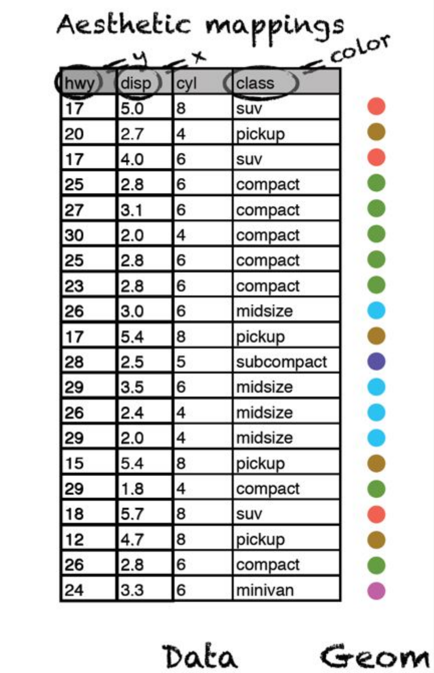
geom_histogram
Histogram

geom_bar
Bars, rectangles with bases on y-axis

geom_point
Points, as for a scatterplot

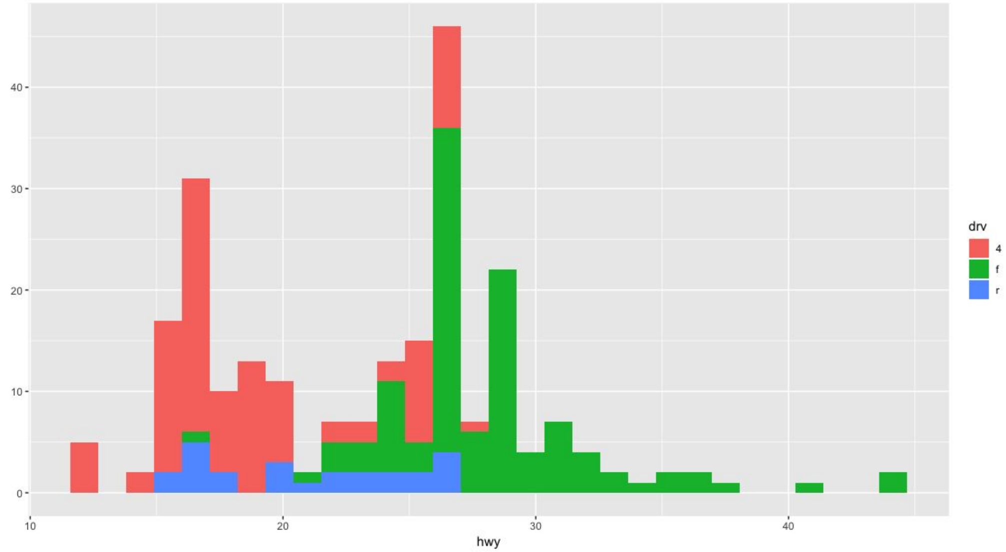
geom_text
Textual annotations

+



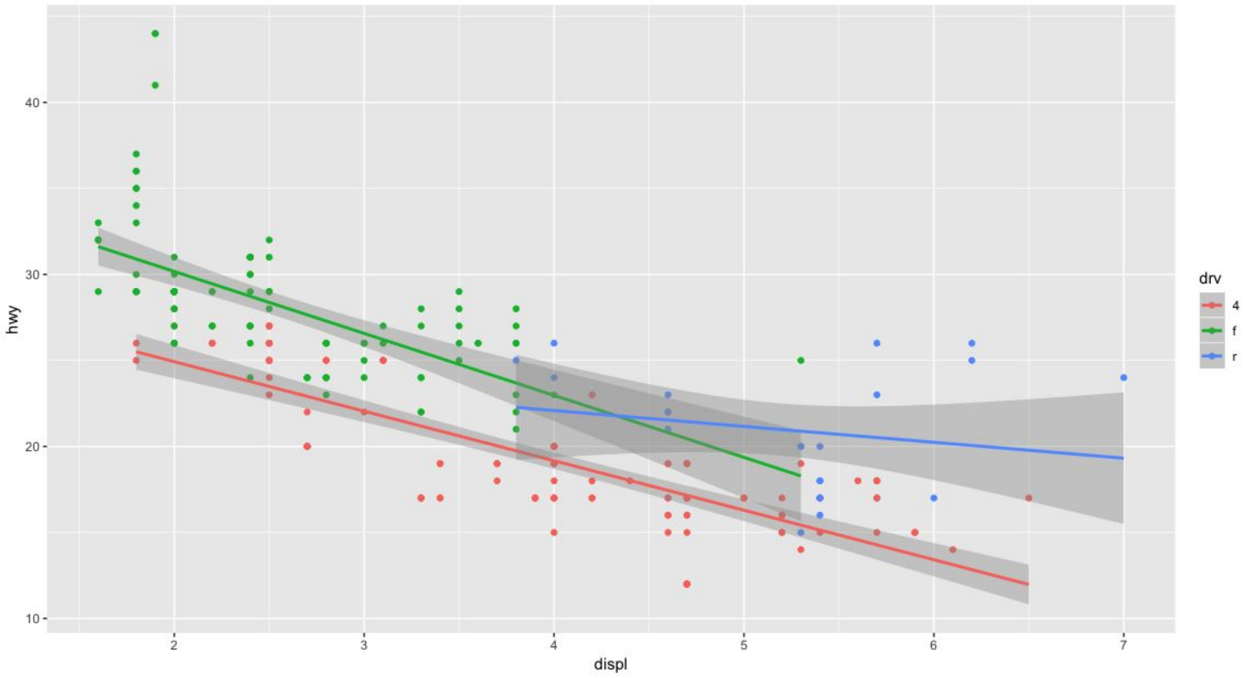
hwy	disp	cyl	class
17	5.0	8	suv
20	2.7	4	pickup
17	4.0	6	suv
25	2.8	6	compact
27	3.1	6	compact
30	2.0	4	compact
25	2.8	6	compact
23	2.8	6	compact
26	3.0	6	midsize
17	5.4	8	pickup
28	2.5	5	subcompact
29	3.5	6	midsize
26	2.4	4	midsize
29	2.0	4	midsize
15	5.4	8	pickup
29	1.8	4	compact
18	5.7	8	suv
12	4.7	8	pickup
26	2.8	6	compact
24	3.3	6	minivan

ggplot2 (qplot)




```r
library(ggplot2)

qplot(x = hwy, data = mpg, fill = drv)
```

ggplot2




```r
library(ggplot2)

qplot(x = displ, y = hwy, data = mpg, color = drv) + geom_smooth(method="lm")
```

Graphical Representation of Data

Choosing the right graphic to display results

https://datavizcatalogue.com/index.html

Examples in R

https://www.data-to-viz.com/

RMarkDown

Include R expressions within MarkDown. Create reports that include dynamic content such as graphics, computed or up-to-date values

```
# Markdown text


Some _text_ that does not *really say much*

```{r, echo=T, cache=TRUE}

db_aggr <- dplyr::count(db, platform, sort = T)

ggplot(head(purchases), aes(x=platform, y=n, fill=platform)) + geom_bar(stat = "identity")

```

Continue with the markdown text
```

What’s that insight, again?

Your findings made through the weekly directive meeting and revealed fundamental metrics for proper web scalability concerns.

Backend is satisfied with your findings and wants to recurrently bring up visitors evolution to discover any trend.

Your findings made through the Monday weekly board meeting and revealed fundamental metrics for proper web scalability concerns.

Backend is satisfied with your findings and wants to recurrently bring up visitors evolution to discover any trend.

Next week, Head of Backend come to you in a rush asking again for results from past week.

Provide an up-to-date report containing requested metrics.

Share Results & Conclusions

Choosing the correct graphics is key to avoid misleading the reader.

Results and conclusions that are not reproducible are not useful since there is no possible validation.

I don't understand at all

I need to go over this again

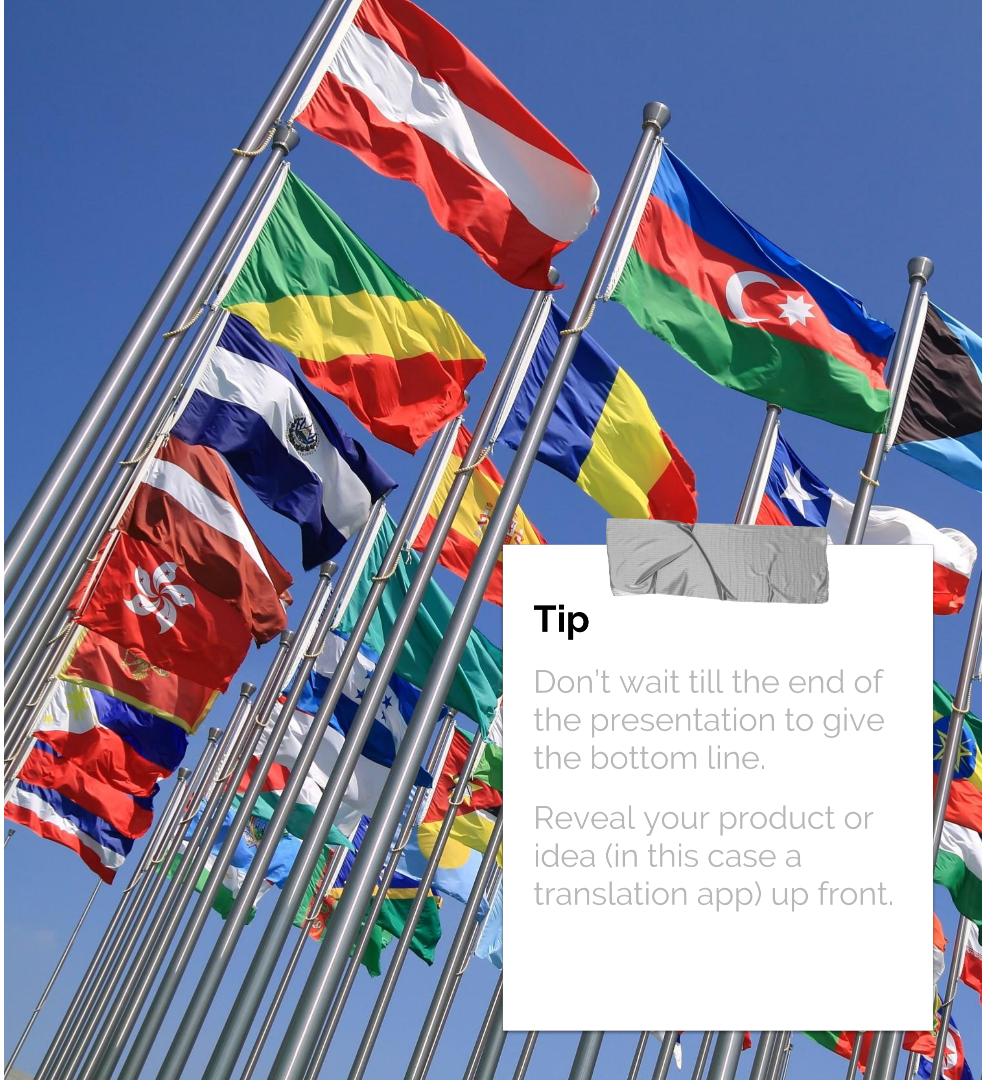
I think I got it, but am not completely comfortable

I got it

I can explain it to someone else

I can generate fitted graphics that reflect the analysis results.

I am able to create reproducible reports automate repetitive tasks.



Tip

Don't wait till the end of the presentation to give the bottom line.

Reveal your product or idea (in this case a translation app) up front.

Homework

Rather than graphic functions from base, give a try to *ggplot2*.

Though scary, graphics from ggplot2 are far superior to standard graphics

You see the pattern at work.
When things work out well, people come back for more.

- Create a RMarkdown document that includes graphics for homework assignment regarding Backend / Marketing.

To improve, we could use feedback...

- *"We're committed to give the best possible student experience."* - Bdata team
- Your feedback is very welcome and optional
- Survey link on Slack

- <u>Time</u>: 3 min

Deepen Learning

- With your super pen and in one post-it, write (2'):
  - Your 2 main learnings/takeaways
  - Your 2 main challenges for the week

- Find a partner that is not your table mate
- Share and help/support each other (3’)
- Find a new partner, repeat (3’)
  - Look at what is different, after the first share

To celebrate our learnings, we close the space

- Everyone, create a circle facing each other’s back
- Raise your right hand to retain the learning in your left brain (shake it a bit)
- Raise your left hand to retain the learning in your right brain (shake it a bit)
- Raise your left leg, because we can
- And, on the count of 3, clap to today’s close the learning
- Celebrate, with an applause to each other

Thank you all

Enjoy your week,
Share your learnings!