GRAU EN ENGINYERIA DE DADES

104365 Visualització de Dades

Seminari 1. *R / ggplot*. Introducció

Index

1.1. Quick summary: **Basic operations**

Basic mathematical operations:

- **+** (add), **-** (substract), **/** (divide) and ***** (multiply)
- **:** to print all numbers between the start value and end values (Example: `15:25`)
- `sqrt()`, `log10()`

Basic logical operations (return TRUE/FALSE):

- **==** to ask whether two values are identical (`3==3`)
- **>** is the first value greater than the second? (`6>7`)
- **<** is the first value smaller than the second? (`5<7`)
- **<=** is the first value less than or equal to the second

1.2. Quick summary: **Slide notes & character information**

Character information: If you type a word in quotes or double quotes, R will repeat it back to you.

! if you do the same without quotes, it will not work - you will get an error.

Example:

```
> "Pepe"
[1] "Pepe"
> Pepe
Error: objeto 'Pepe' no encontrado
>
```

Slide notes: The lines that start with # are comments in our R code - R will not interpret them.

Example:

```
> "Pepe" # Comment
[1] "Pepe"
>
```

1.3. Quick summary: **Variables, vectors and assignment**

Variable: any characteristic or measurement that varies among individuals.

- **Numerical vectors:** `c(1,2,3,4)`
- **Character vector:** `c(“nau”, “vaixell”, “coet”)`
- **Assignment:** `a<-c(1,2,3,4) or b=c(“nau”, “vaixell”, “coet”)`
- **Check what type of vectors we have :** `class(a)`

Example:

```
> a<-c(1,2,3,4)
> class(a)
[1] "numeric"
> a
[1] 1 2 3 4
> b=c("nau","vaixell","coet")
> class(b)
[1] "character"
> b
[1] "nau"     "vaixell" "coet"
>
```

1.3. Quick summary: **Factors in R- categories**

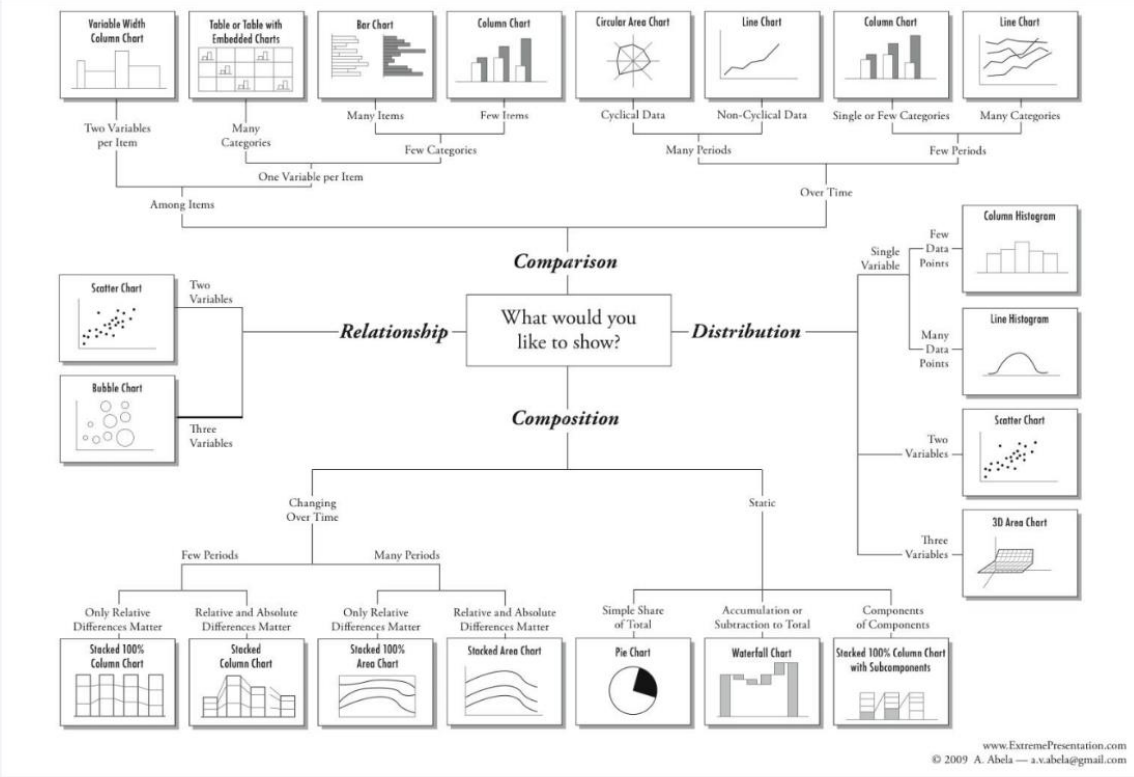
!! **Factors**, a character vector with **different groups or categories** (hence it is categorical), **which in R are called levels**: `as.factor()` /levels

To work with factors, we'll use the **forcats package**, which is part of the core **tidyverse**.

https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/factor

1.4. Quick summary: **What would you like to show?**

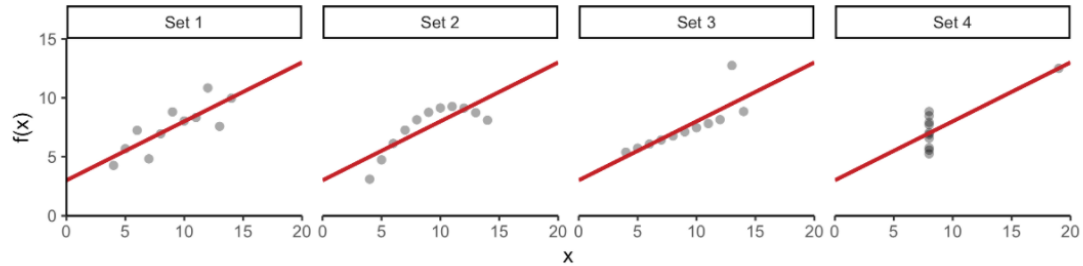
Para elegir la visualización más adecuada
es necesario conocer muchos tipos de visualizaciones



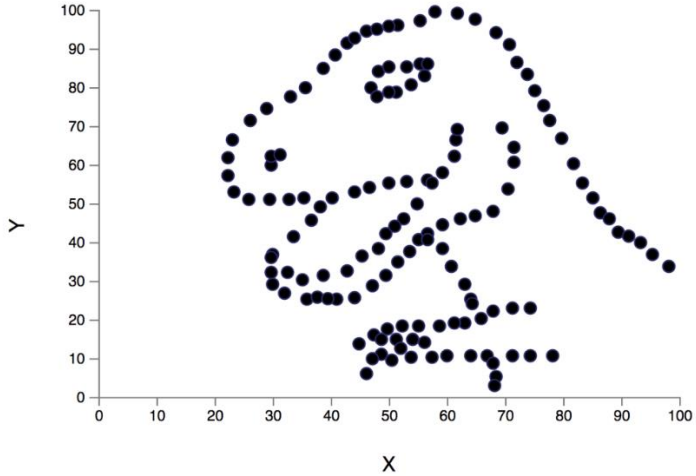
You saw it with Guillermo

1.4. Quick summary: **What would you like to show?**

Anscombe's plots



Datasaurus!!



You saw it with Guillermo

2.1. R tools: plot - **Scatter plot**

You used "plot" in the Statistical Analysis subject

LET's do a simple example together, to refresh it:

Variation between values on two different vectors:

1. Assign to the numerical variable x, all the values between 1 and 10
2. Assign to the numerical variable y, all the values between 25 and 34
3. Use the **plot** function to plot them easily: `plot(x,y)`

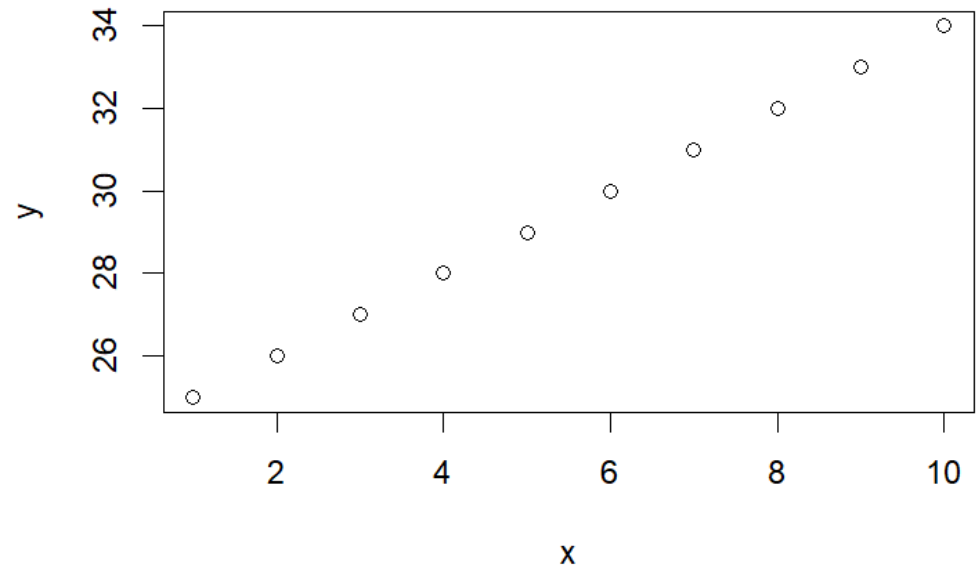
2.1. R tools: plot - **Scatter plot**

Variation between values on two different vectors:

```
> x <- 1:10 #assign to x, all the values between 1:10
> y <- 25:34 #assign to y, all the values between 25:34
> plot(x,y)
>
```

Option 2:

```
> x = 1:10
> y = 25:34
> plot (x,y)
>
```



2.1. R tools: plot - **Scatter plot**

Basic edition of `plot()` using arguments:

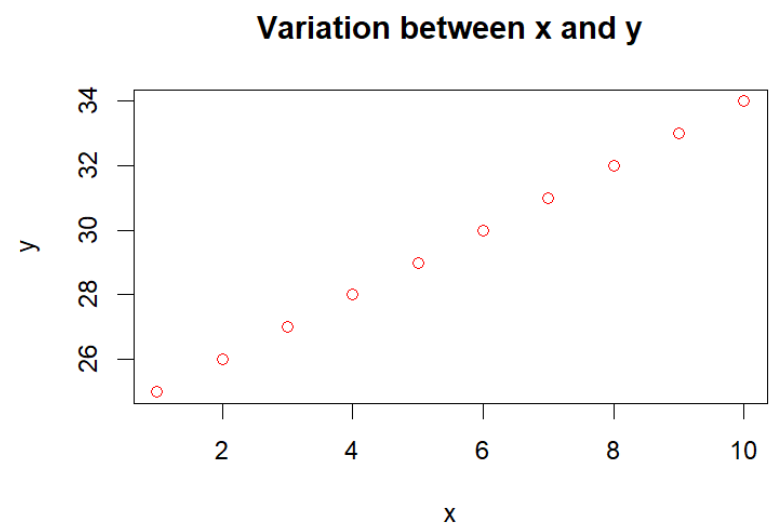
1. Add a tittle to the graph by using the **`main`** argument

```
> plot(x,y, main="Variation between x and y")
```

2. Make the points in our plot a different colour ("`red`") by using the **`col`** argument

```
> plot(x,y, main="Variation between x and y", col="red")
```

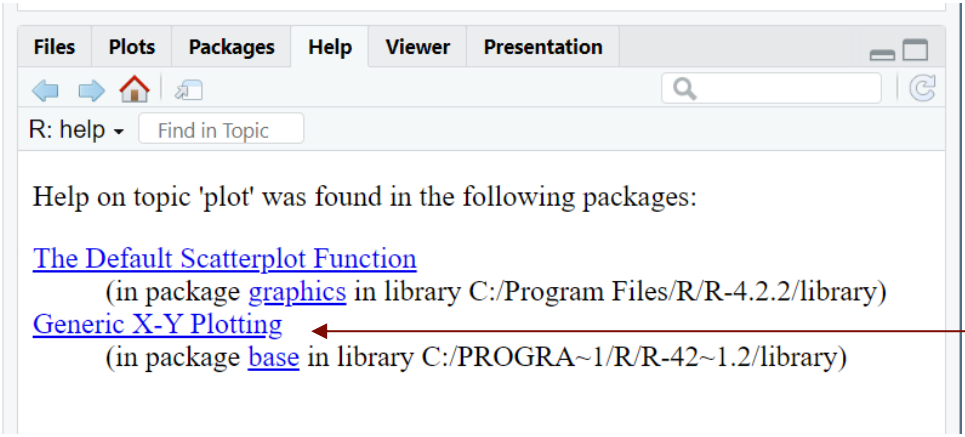
3. Use R Help to do it: **`help(plot)`** and/or **`?plot`**



2.1. R tools: plot - **Scatter plot**

Basic edition of `plot()` using arguments:

1. Add a tittle to the graph by using the **`main`** argument
2. Make the points in our plot a different colour (`"red"`) by using the **`col`** argument
3. Use R Help to do it: **`help (plot)`** and/or **`?plot`**



See possible arguments and examples

2.1. R tools: plot - **Scatter plot**

Basic edition of `plot()` using arguments:

1. Add a tittle to the graph by using the **`main`** argument
2. Make the points in our plot a different colour (red) by using the **`col`** argument
3. Use R Help to do it: **`help (plot)` and/or `?plot`**
4. Similarly, use a line using the **`type`** argument

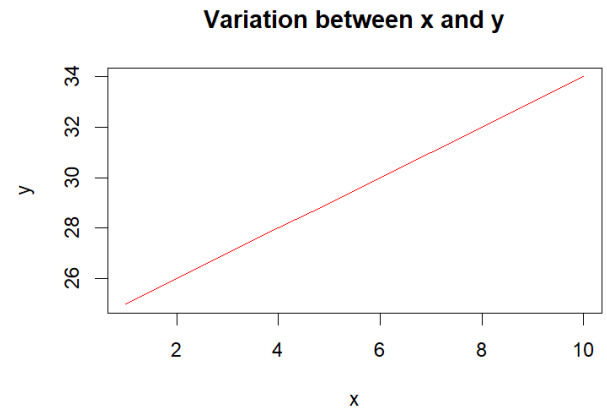
```
> plot(x,y, main="Variation between x and y", col="red", type="l")
```

! To see examples of graphics, you can also use **`demo(graphics)`**

2.1. R tools: plot - **Scatter plot – to fit a line**

We have:

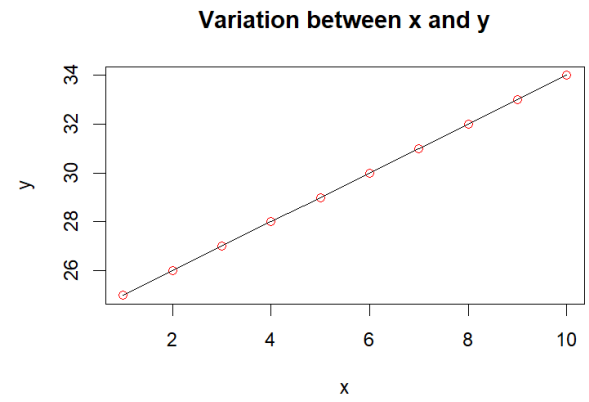
```
> plot(x,y, main="Variation between x and y")
> plot(x,y, main="Variation between x and y", col="red")
> plot(x,y, main="Variation between x and y", col="red", type="l")
>
```



Now use only the arguments **`main`** and **`col`**.

Afterwards type **`lines(x,y)`**. You will get:

```
> plot(x,y, main="variation between a and y",col="red")
> lines(x,y)
>
```



2.2. R tools for **Visualizing distribution**

$N(\mu, \sigma^2)$

μ mean

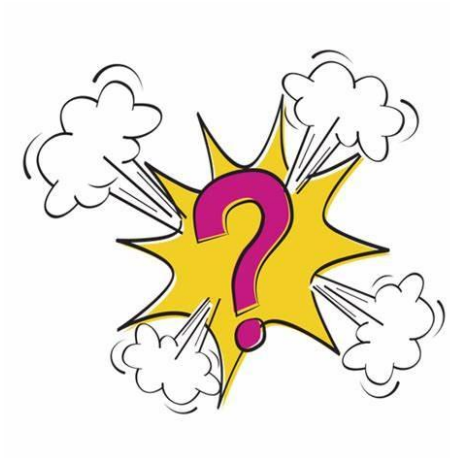
σ **s**tandard **d**eviation

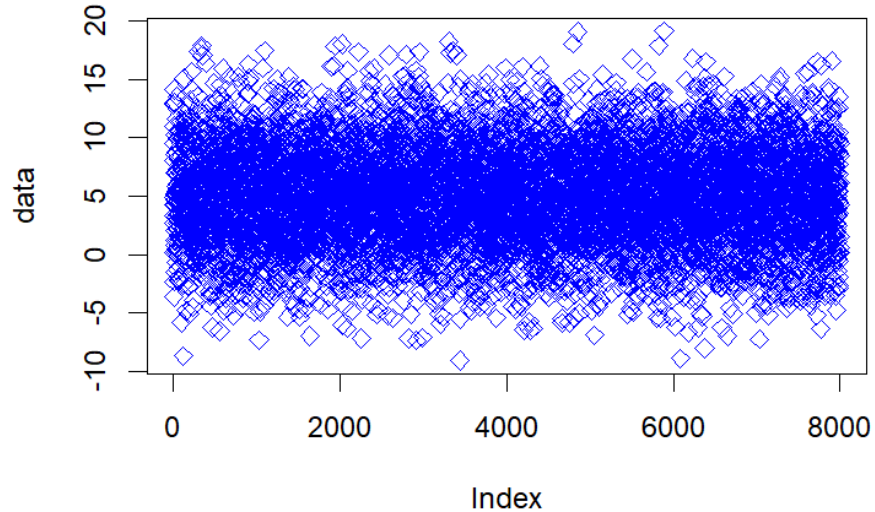
```
> # rnorm(n) create a sample of n numbers which are normally distributed
> rnorm (8000)
```

```
> # rnorm(8000, mean=5, sd=4.1) create a sample of 8000 with a known mean and sd
```

– **Try to plot it**

```
> data <- rnorm(8000, mean=5, sd=4.1)
> plot(data, pch=5, col="blue") #pch changes the o symbol
>
```

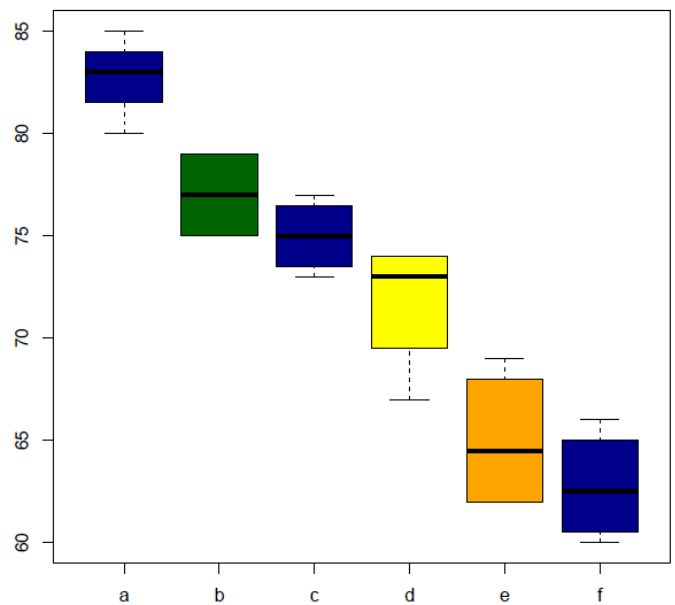




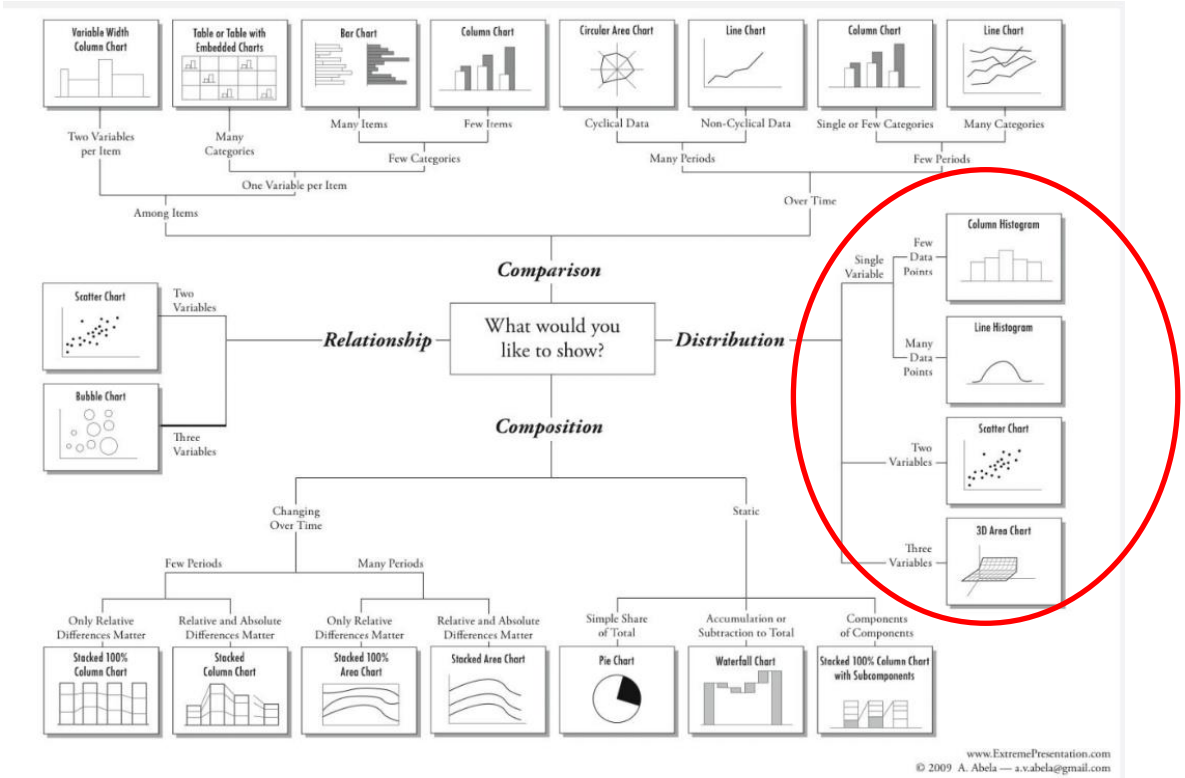
2.2. R tools for **Visualizing distribution**

You saw boxplots in statistical analysis (a tool to show distributions) & how to show distributions with Guillermo:

```
boxplot(diam~radon,col=c("darkblue","darkgreen","darkblue","yellow","orange"))
```



Boxplot example from the statistical analysis subject



Visualizing distribution from Guillermo

2.2. R tools for **Visualizing distribution**

$N(\mu, \sigma^2)$

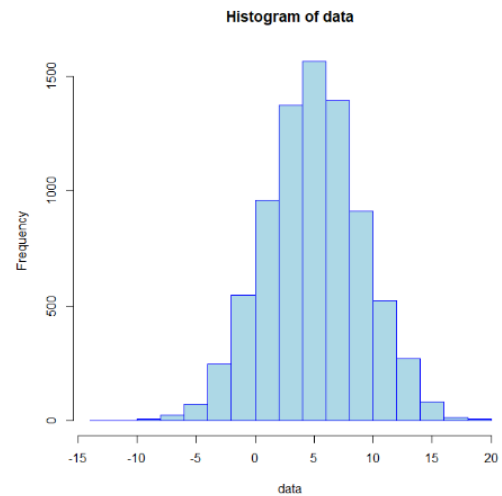
μ mean

σ **s**tandard **d**eviation

```
> # rnorm(n) create a sample of n numbers which are normally distributed
> rnorm (8000)
> # rnorm(8000, mean=5, sd=4.1) create a sample of 8000 with a known mean and sd
```

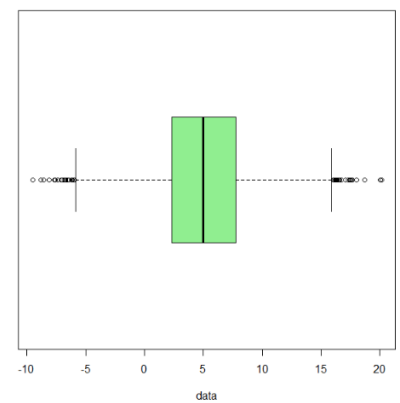
– **Histogram**

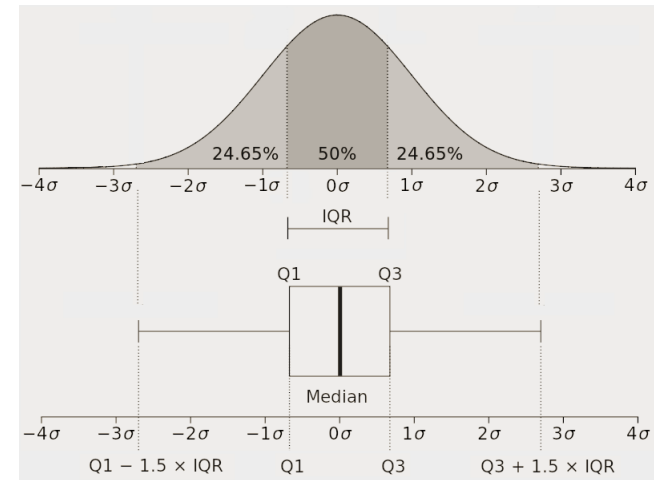
`hist (...)`



– **Boxplots**

`boxplot (...)`





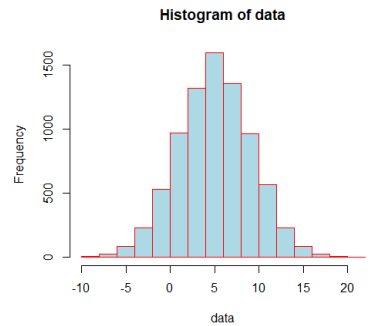
2.2. R tools for **Visualizing distribution**

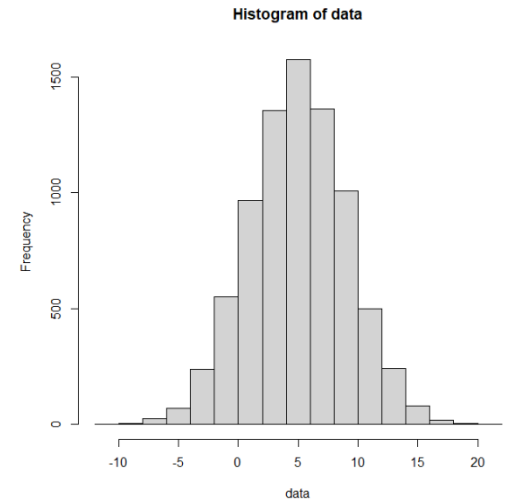
– Histogram

```
> data <- rnorm(8000, mean=5, sd=4.1)
> hist (data)
>
```

Use histograms when you want to explore the distribution of a single continuous variable, especially when you're interested in understanding its shape and characteristics.

```
> hist(data,col="lightblue", border="red")
> #play with different arguments - remember ?hist
>
```



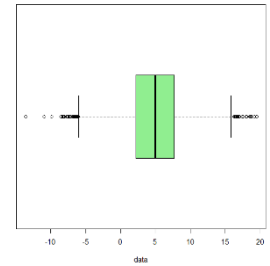


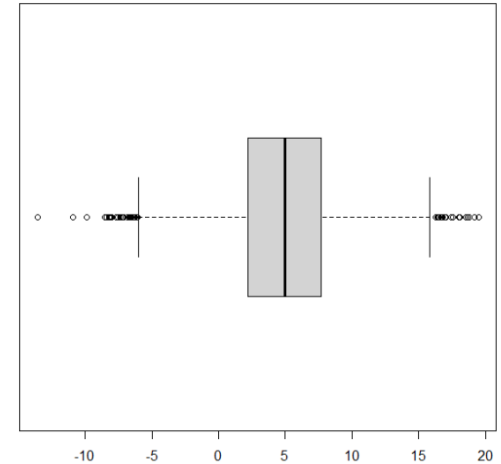
– Boxplots

```
> data <- rnorm(8000, mean=5, sd=4.1)
> boxplot(data)
> boxplot(data, horizontal=TRUE)
>
```

However, we will see later that we use boxplots when you want to compare the distributions of one or more continuous variables between different groups or categories. They are also useful for identifying outliers and understanding the variability within each group.

```
> boxplot(data, horizontal=TRUE, xlab='data', col="lightgreen
> # play with different arguments - remember ?boxplot
>
```

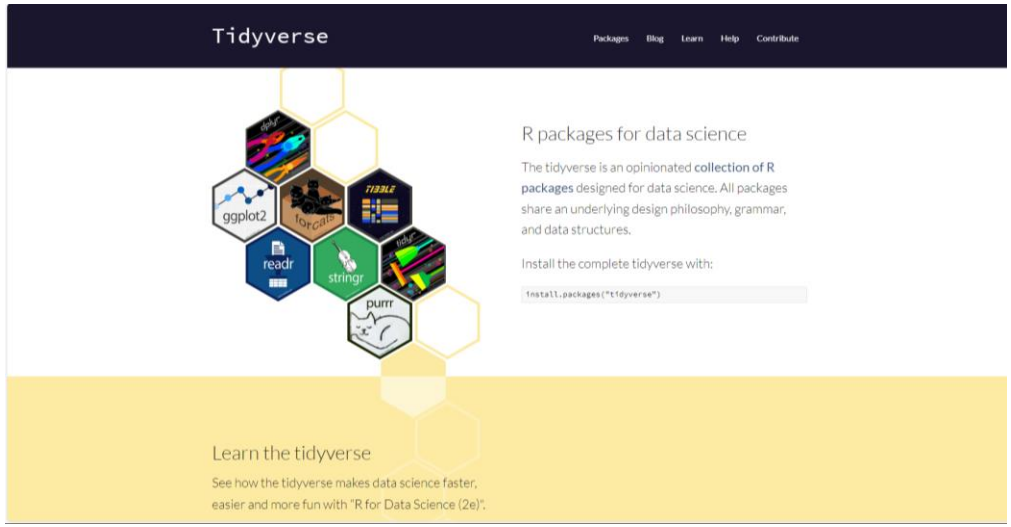




3. R tool: ggplot2

NEW R tool: ggplot2 can create simple and complicated data visualization:

- ggplot2 is part of the tidyverse data science toolkit.
- Tidyverse is a coherent system of packages for data manipulation, exploration and visualization that share a common design philosophy. These were mostly developed by Hadley Wickham himself, but they are now being expanded by different contributors.



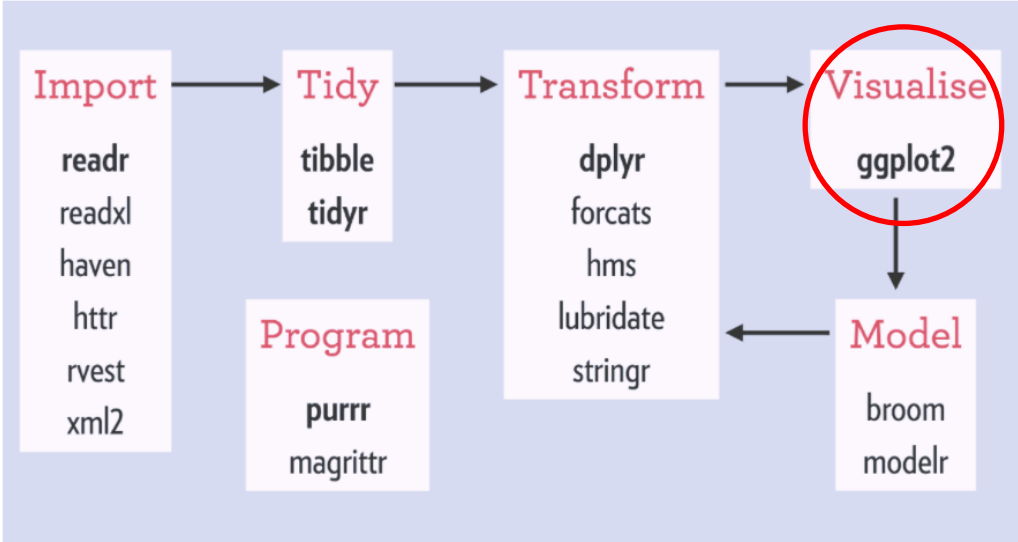
https://www.tidyverse.org/

3. R tool: ggplot2

NEW R tool: ggplot2 can create simple and complicated data visualization:

- ggplot2 is part of the tidyverse data science toolkit

The tidyverse includes:



- `readr` for importing data
- `dplyr` for data manipulation
- **`ggplot2`** for data visualization
- `stringr` for string manipulation
- `tidyr` for putting data into a tidy format

https://rviews.rstudio.com/2017/06/08/what-is-the-tidyverse/

4. ggplot2 prerequisites

ggplot2 can create simple and complicated data visualization:

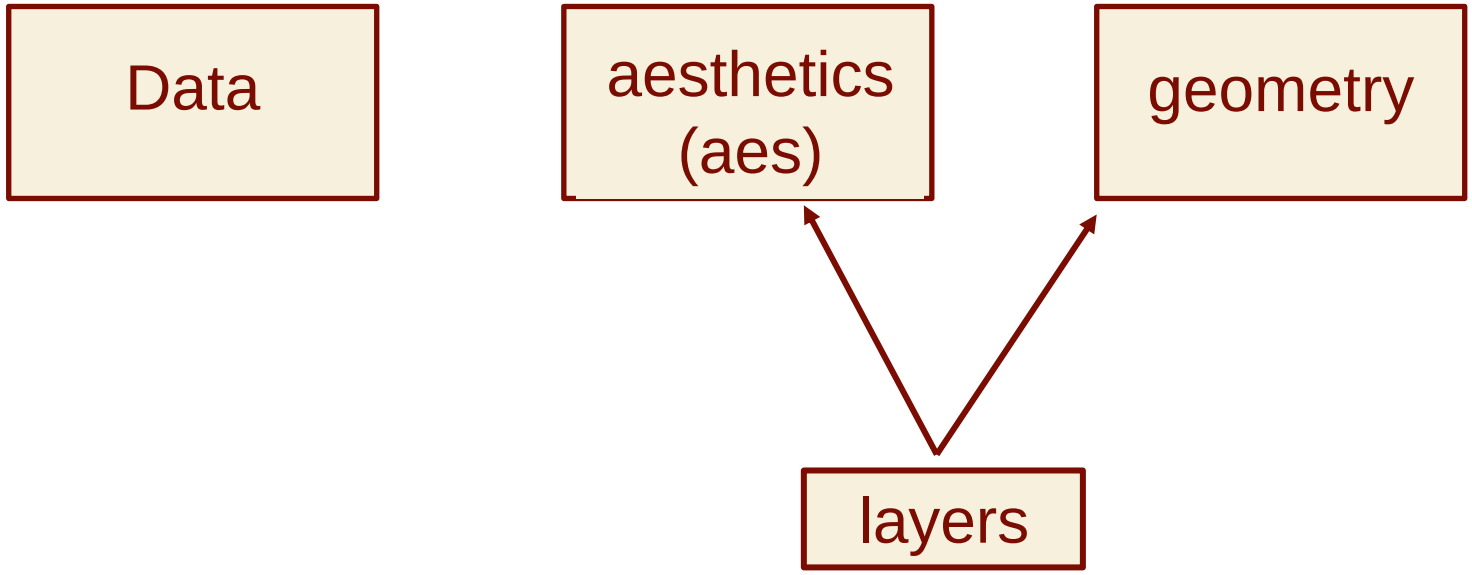
- ggplot2 is part of the tidyverse data science toolkit from Hadley Wicklham
- **Prerequisites:**
  - You will need *to install the tidyverse package and run the library*. You only need to install the package once
  - **! BUT you need to reload it every time you start a new session**

```
> install.packages("tidyverse")
> library(tidyverse)
```

```
> library(tidyverse)
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.0     ✓ readr     2.1.4
✓ forcats   1.0.0     ✓ stringr   1.5.0
✓ ggplot2   3.4.1     ✓ tibble    3.1.8
✓ lubridate 1.9.2     ✓ tidyr     1.3.0
✓ purrr     1.0.1
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✗ dplyr::filter() masks stats::filter()
✗ dplyr::lag()    masks stats::lag()
```

5.1. How ggplot2 works

In ggplot2, the graphics are created through “successive steps”. It has three fundamental parts:



What is the main difference with the R’s graphics tools used before?

It allows us to **interactively create graphics**: Starting with the data that we want to show and adding the different layers that will complement and design our graphic. We can combine independent “tools” in very different ways.

5.1. How ggplot2 works

Four critical pieces you need to know:

1. **The `ggplot()` function**: It is simply the function we use to initiate ggplot2 plot.
2. **The data parameter:** It tells ggplot2 the name of the dataframe that you can visualize.

! When you use ggplot, you need to use variables that are contained within a dataframe. The data parameter tells ggplot where to find those variables.

3. **The `aes()` function**: It tells `ggplot()` the "variable mappings".

Note: In the previous scatter plot, we connected one numeric variable x to another numeric variable y. We "mapped" these variables to different axes within the visualization. **The `aes()` function** allows us to specify those mappings; it **enables us to specify which variables in a dataframe should connect to which parts of the visualization.**

5.1. How ggplot2 works

Four critical pieces you need to know:

1. **The `ggplot()` function**: It is simply the function we use to initiate ggplot2 plot.
2. **The data parameter:** It tells ggplot2 the name of the dataframe that you can visualize.
3. **The `aes()` function**: It tells `ggplot ()` the "variable mappings".
4. **Geometric objects (AKA, "geoms"):** A geometric object is the thing that we draw. In ggplot2, we need to explicitly state the type of `geom` that we want to use (bars, lines, points, etc).

When drawing a scatter plot, we'll do this by using `geom_point()`.

5.2. ggplot2 Basics

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

data **geom** x = F · y = A **coordinate system** **plot**

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data **geom** x = F · y = A color = F size = A **coordinate system** **plot**

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
 stat = <STAT>, position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings data geom

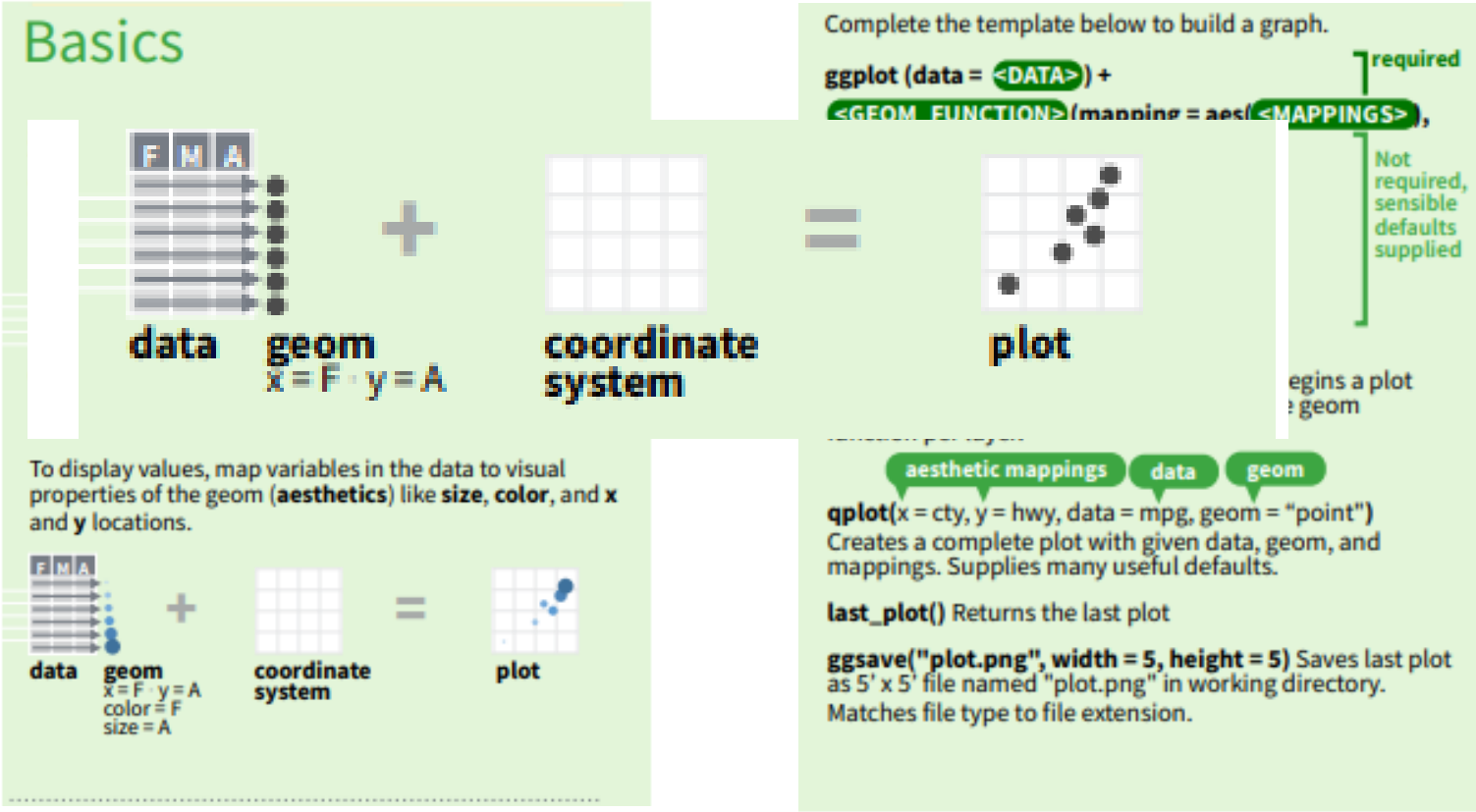
qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

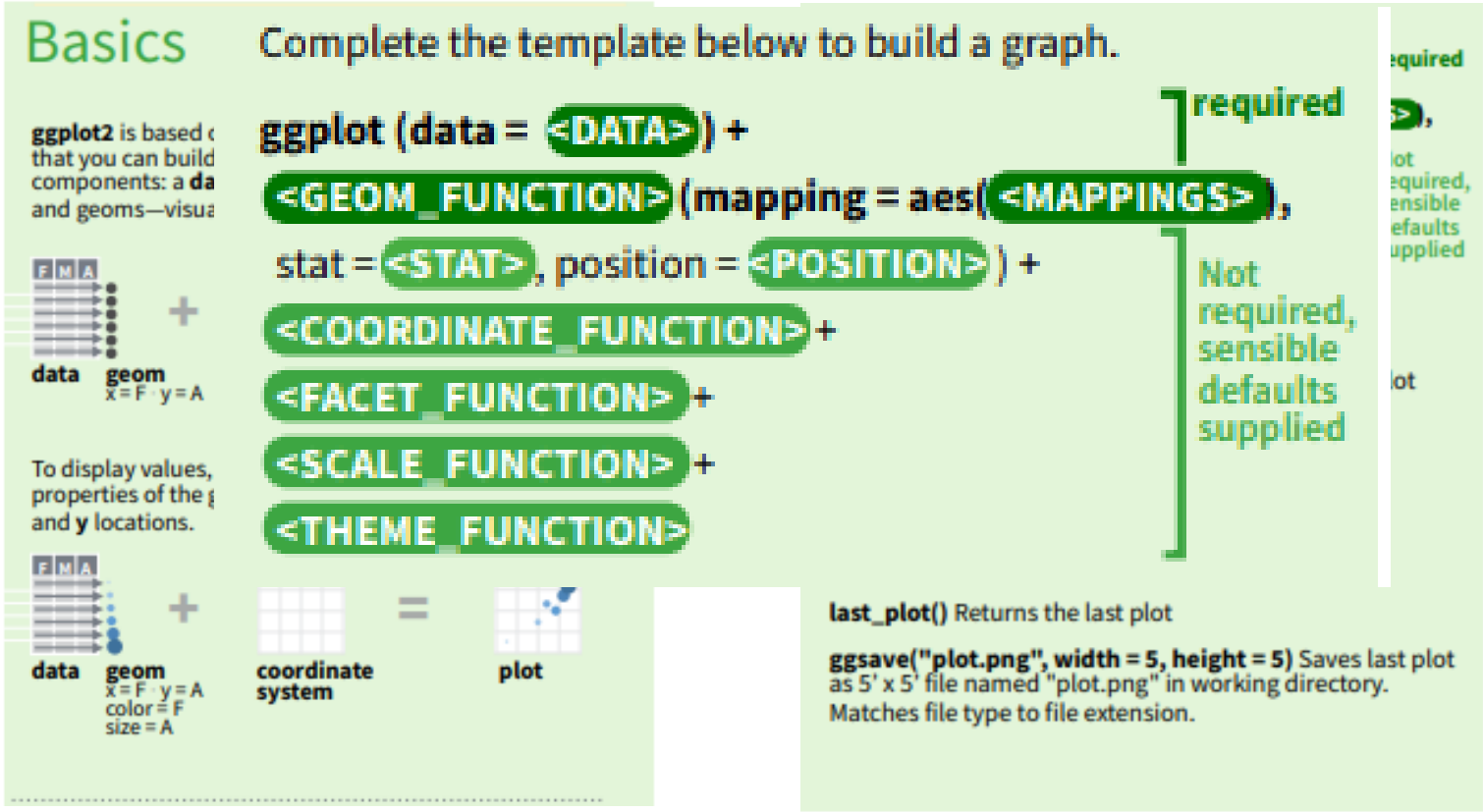
https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

5.2. ggplot2 Basics



1st: Data
2nd: A coordinate system (cartesian by default)
3rd: Geometry

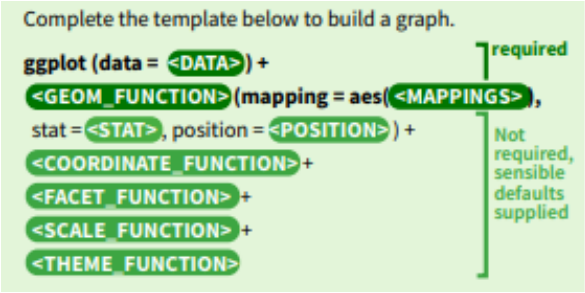
5.2. ggplot2 Basics



Afterwards, we can complement our graphic through the mapping.

5.2. ggplot2 Basics

- Numerical variables:
  - Continuous
  - Discrete



aes	Discreta	Contínua
Color (color)	Arco iris de colors	Gradient de colors
Forma (shape)	Diferent formes	NO APLICA
Talla (size)	Escala discreta de talles	Mapeig lineal entre l'àrea i el valor
Transparència (alpha)	NO APLICA	Mapeig lineal a la transparència

5.2. ggplot2 Basics

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2) x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

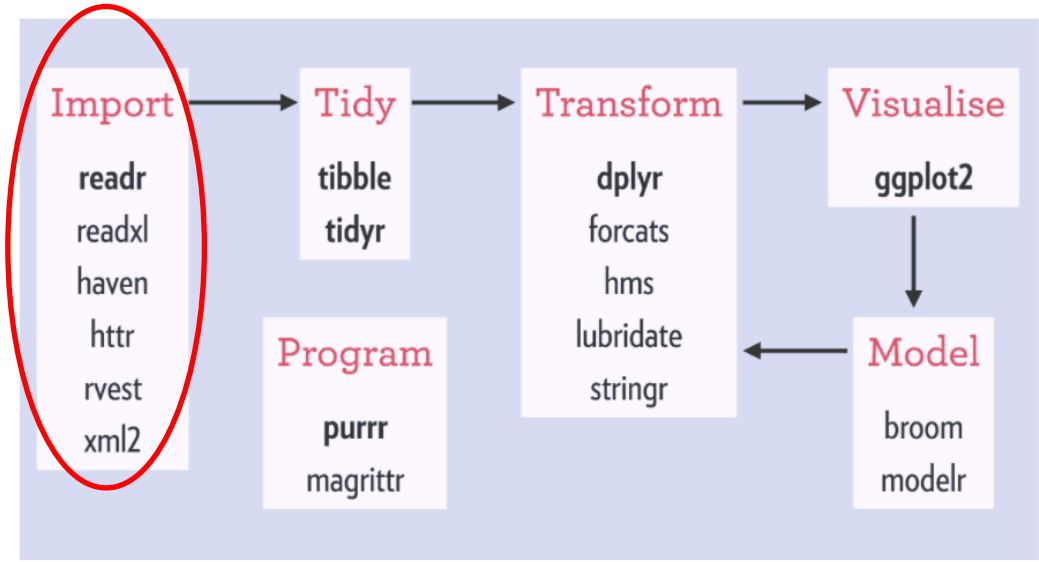
f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

6. Read data with tidyverse

Read data with tidyverse :



- **`readr:`** to read rectangular data (like csv, tsv, and fwf)
- **`readxl:`** to read excel files (.xls and .xlsx)
- **`haven:`** to import SPSS, Stata, and SAS files
- **`httr:`** to import web APIs
- **`rvest:`** to scrape information from web pages (html)
- **`xml2:`** to import xml

6. Read data with tidyverse

Tabular and non-tabular data:

Read Tabular Data - These functions share the common arguments:

```
read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"),
  quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000,
  n_max), progress = interactive())
```

Comma Delimited Files
read_csv("file.csv")
To make file.csv run:
write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

Semi-colon Delimited Files
read_csv2("file2.csv")
write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

Files with Any Delimiter
read_delim("file.txt", delim = "|")
write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")

Fixed Width Files
read_fwf("file.fwf", col_positions = c(1, 3, 5))
write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv") Also **read_table().**
write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

USEFUL ARGUMENTS

Example file
write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

No header
read_csv(f, **col_names = FALSE**)

Provide header
read_csv(f, **col_names = c("x", "y", "z")**)

Skip lines
read_csv(f, **skip = 1**)

Read in a subset
read_csv(f, **n_max = 1**)

Missing Values
read_csv(f, **na = c("1", ".")**)

Read Non-Tabular Data

Read a file into a single string
read_file(file, locale = default_locale())

Read each line into its own string
read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive())

Read Apache style log files
read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive())

Read a file into a raw vector
read_file_raw(file)

Read each line into a raw vector
read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive())

Reference:

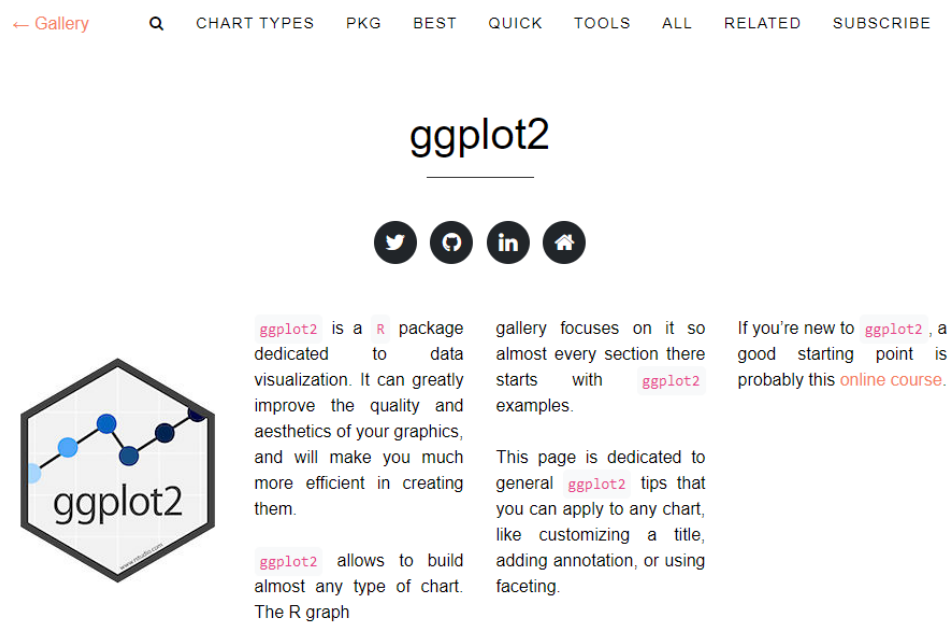
data-import (rawgit.com)

Examples to read .csv:
https://readr.tidyverse.org/articles/readr.html

Links

https://www.r-graph-gallery.com/index.html

https://r-graph-gallery.com/ggplot2-package.html



Let’s do some exercises