Publications Office of the European Union
2 rue Mercier, 2985 Luxembourg, LUXEMBOURG • Tel. +352 2929- • Fax +352 2929-

Boosting your data analysis skills with data visualisation and ggplot2

Direction Access to and Reuse of Public Information

Unit EU Open Data and CORDIS

Sector EU Open Data

The context

This training course is organized in the scope of OP project within the ISA2 programme

ISA2 supports the development of **digital solutions** enabling public administrations, businesses and citizens in Europe to benefit from **interoperable cross-border** and **cross-sector public services**.

How OP is involved in ISA2?

OP is aiming at developing open data related activities in the areas of:

- Data visualisation
- Linked open data
- Persistent identification

Upcoming training & workshop sessions

Topic	Type of session	Lux + webex	Bxl
Telling your story through data visualisation	Training	25/06	28/06
Making great online data visualisations without coding	workshop	26/06	-
Going beyond bars and lines: practising non-standard data visualisation	Training	24/09	Sep-Oct
Making data visualisations like a pro: D3.js	Workshop	25/09	-
Applying data visualisation best practices in real use cases	workshop	24/10	-

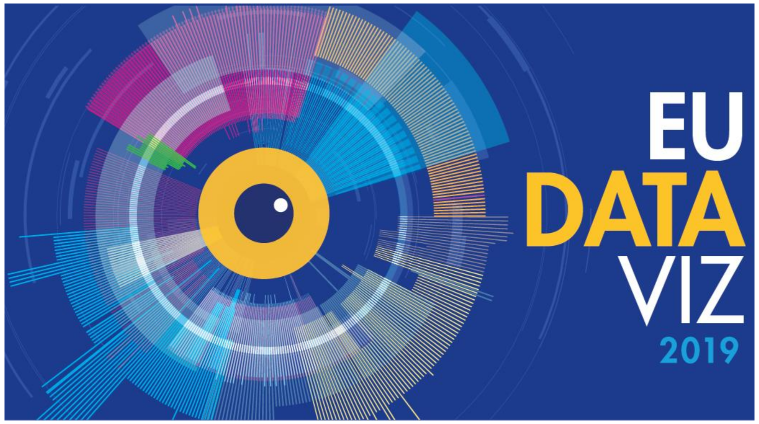
and webinars (topic like for the trainings) … stay tuned!

Materials will be published on https://data.europa.eu/euodp/en/knowledge-center

Data visualization events in 2019

EU Datathon 2019

- Date: 13 June 2019
- Venue: Residence Palace - Brussels
- Website: https://publications.europa.eu/eudatathon
- e-mail: op-datathon@publications.europa.eu



EU DataViz 2019 – Data Visualisation for the Public Sector

- Date: 12 November 2019
- Venue: European Convention Center - Luxembourg
- Website: https://publications.europa.eu/eudataviz
- e-mail: op-eu-dataviz@publications.europa.eu

Where to find the information about our training workshop and webinar sessions?

Visit our data visualisation community to find all the information about previous and upcoming sessions at:

https://webgate.ec.europa.eu/fpfis/wikis/display/EUODDVC/Data+Visualisation+-+Training+Package+2019

Agenda

09:00	Introduction
	Visualisation in data analysis
	ggplot, R and RStudio
10:30	Coffee break
	Data diagnostics
	Visualising distributions
12:00 – 13:00	Lunch
	Visualising covariation
	Visualising multiple dimensions
14:30	Coffeebreak
	Visualising time series
	Saving and sharing visualisations
	Other tools
16:30	Q&A

1. INTRODUCTION

Participants

Institution/DG and role?

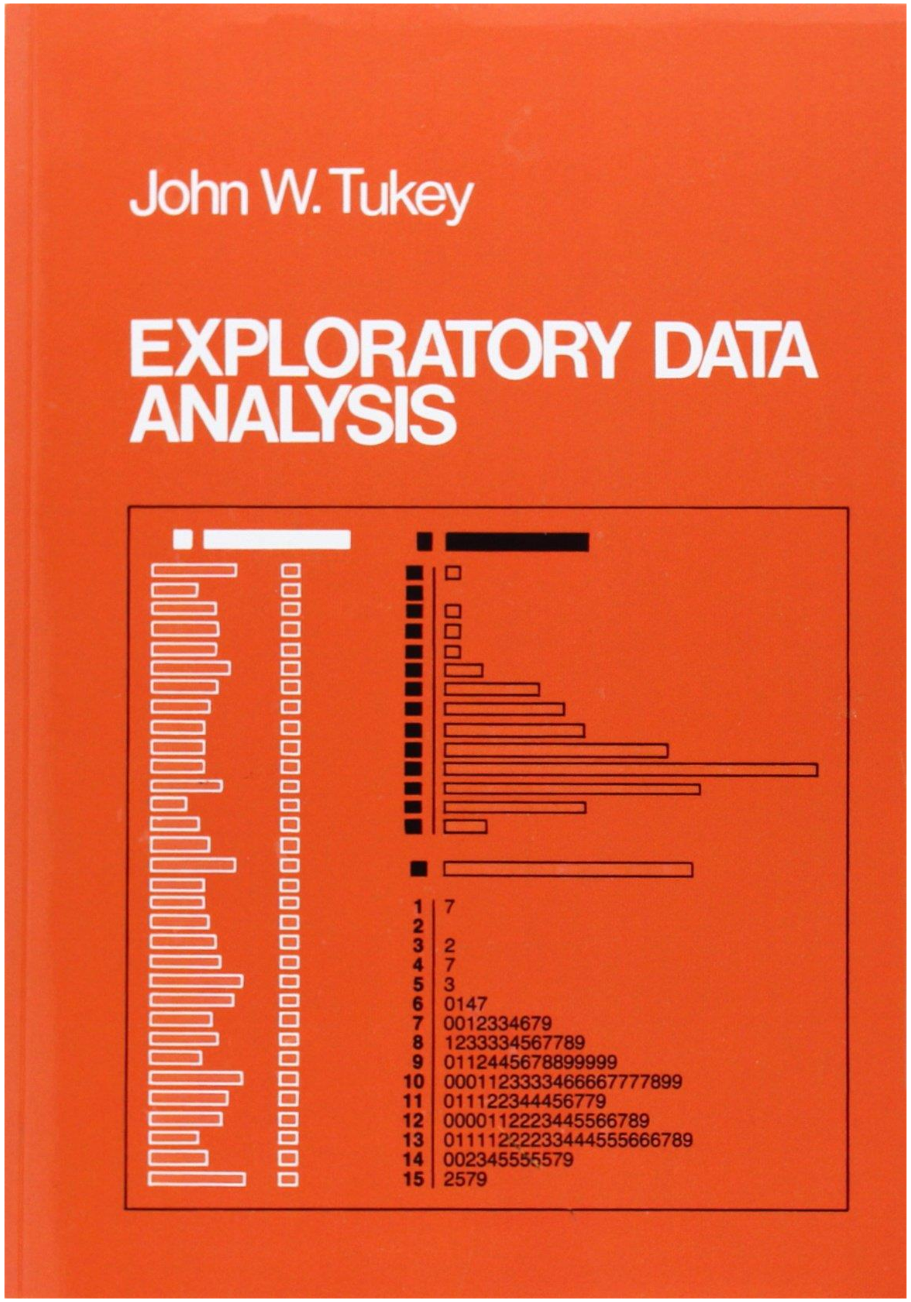
Experience with coding, R and ggplot2?

Expectations for today?

2.

VISUALISATION IN DATA ANALYSIS?

EDA
What is it?



John W. Tukey

EXPLORATORY DATA ANALYSIS

```
 1 | 7
 2 |
 3 | 2
 4 | 7
 5 | 3
 6 | 0147
 7 | 0012334679
 8 | 1233334567789
 9 | 0112445678899999
10 | 000112333346666777899
11 | 011122344456779
12 | 0000112223445566789
13 | 01111222233444555666789
14 | 002345555579
15 | 2579
```

EDA
What is it?

John Tukey

"The simple graph has brought more information to the data analyst's mind than any other device"

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

EDA
What is it?

Wikipedia

"EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task"

EDA
What is it?

From R for Data Science, by Hadley Wickham

Hadley Wickham is the creator of ggplot2

Your goal during EDA is to develop an understanding of your data.

EDA is an iterative cycle. You

- generate questions about your data
- search for answers by visualising, transforming, and modelling your data
- use what you learn to refine your questions and/or generate new questions

EDA
Tool requirements

Fast	Slice & dice data Iterate over visualisations Good defaults
Flexible	Many visualisation types Combine visualisations
Integrates with data manipulation	The Tidyverse

EDA

R Tidyverse

tidyverse.org

Declaritavely create graphics

Grammar for data manipulation

Get data in tidy format

3.

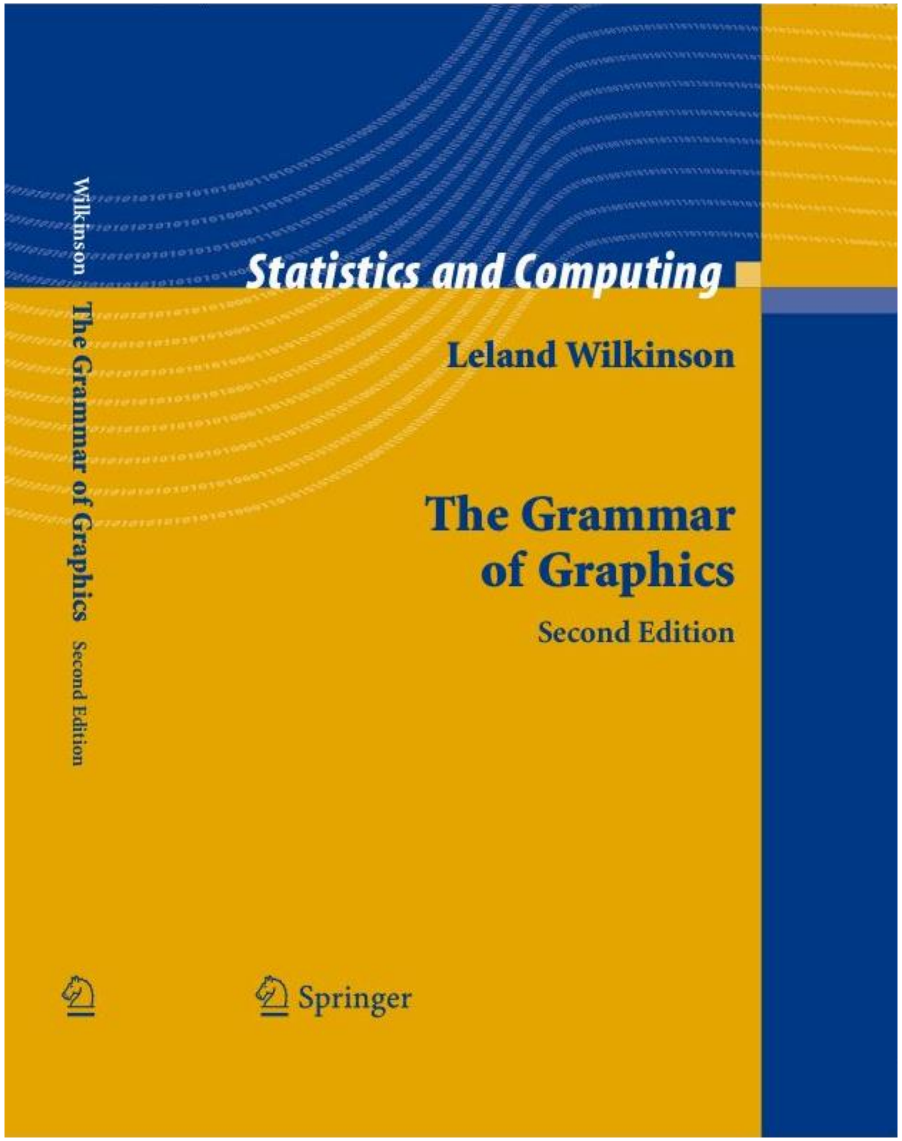
GGPLOT2

ggplot2
Grammar of graphics

R package for making visualisations

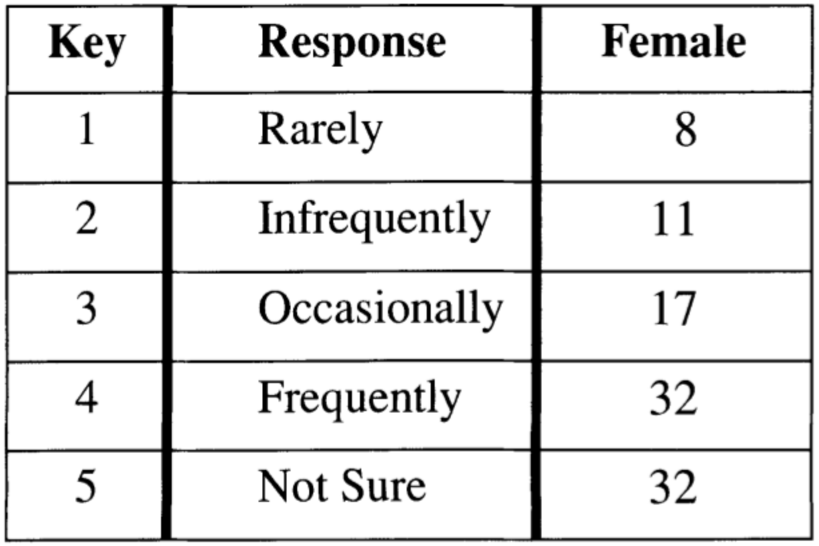
Based on the "Grammar of Graphics" by Leland Wilkinson

"A a tool that enables us to concisely describe the components of a graphic"



ggplot2
Grammar of graphics

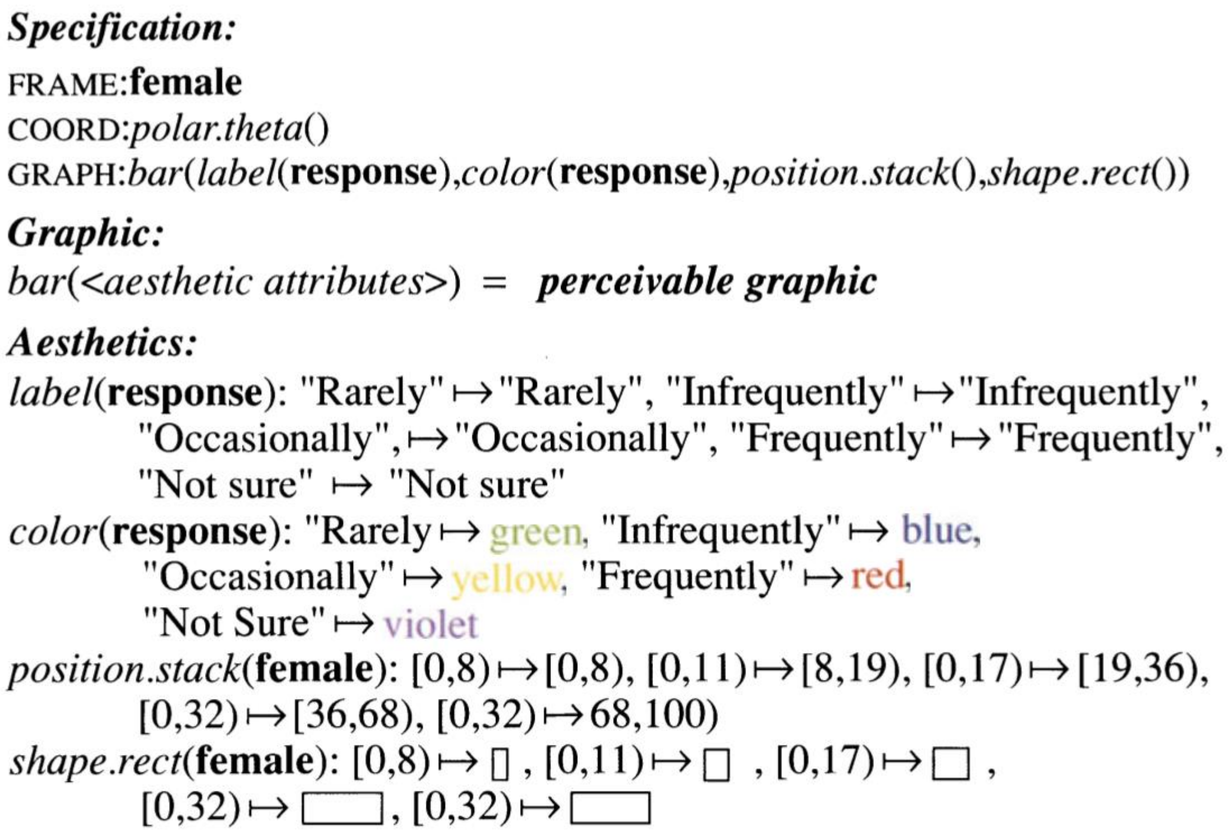
An example of how a visualisation is constructed from a data set



Key	Response	Female
1	Rarely	8
2	Infrequently	11
3	Occasionally	17
4	Frequently	32
5	Not Sure	32

ggplot2
Grammar of graphics

Grammar of graphics specification



Specification:

FRAME:**female**
COORD:*polar.theta*()
GRAPH:*bar*(*label*(**response**),*color*(**response**),*position.stack*(),*shape.rect*())

Graphic:

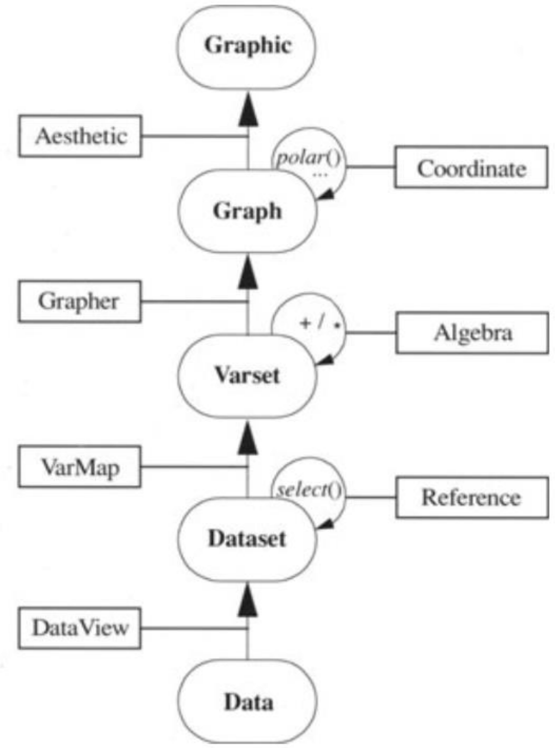
bar(<*aesthetic attributes*>) = ***perceivable graphic***

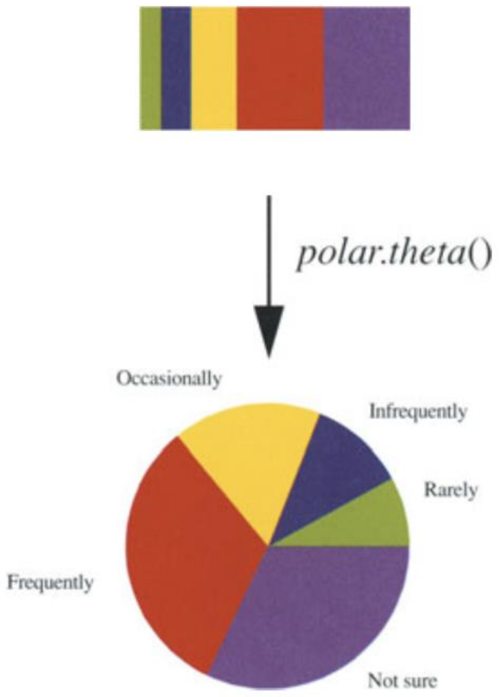
Aesthetics:

label(**response**): "Rarely" ↦ "Rarely", "Infrequently" ↦ "Infrequently",
"Occasionally", ↦ "Occasionally", "Frequently" ↦ "Frequently",
"Not sure" ↦ "Not sure"
color(**response**): "Rarely ↦ green, "Infrequently" ↦ blue,
"Occasionally" ↦ yellow, "Frequently" ↦ red,
"Not Sure" ↦ violet
position.stack(**female**): [0,8) ↦ [0,8), [0,11) ↦ [8,19), [0,17) ↦ [19,36),
[0,32) ↦ [36,68), [0,32) ↦ 68,100)
shape.rect(**female**): [0,8) ↦ ▯ , [0,11) ↦ ▯ , [0,17) ↦ □ ,
[0,32) ↦ ▭, [0,32) ↦ ▭

ggplot2
Grammar of graphics

Grammar of graphics:
result





ggplot2

Ggplot2 is an implementation of the grammar of graphics in R

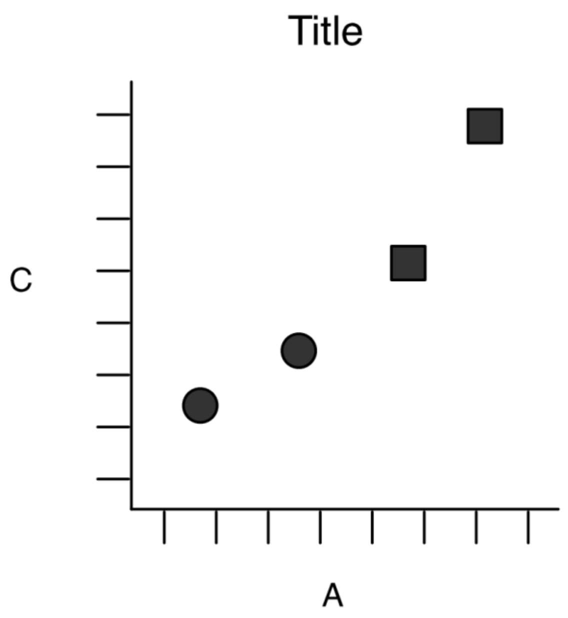
How do we go from data to a visualisation in ggplot2?

Data

A	*B*	*C*	*D*
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Used aesthetics

x	*y*	Shape
2	4	a
1	1	a
4	15	b
9	80	b



Plot

x	*y*	Shape
25	11	circle
0	0	circle
75	53	square
200	300	square

Data in aesthetic space

ggplot2

Map **data** to

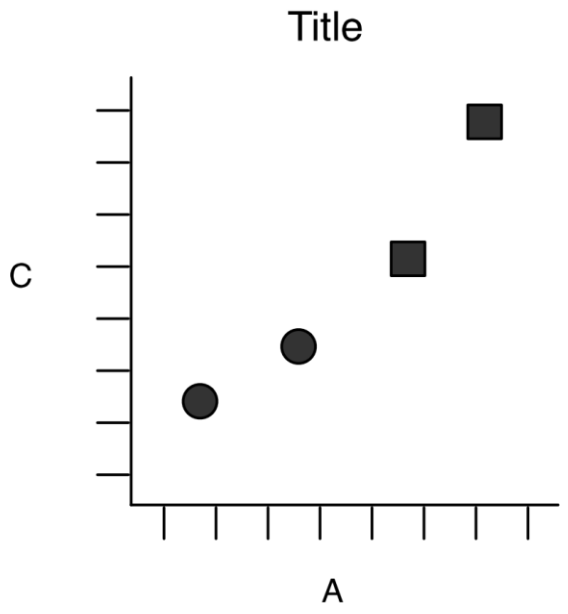
aesthetics of

geometries

Data

A	*B*	*C*	*D*
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

```
ggplot(data=Data,
aes(x=A, y=B, shape=D)) +
geom_point()
```



Plot

Demo

3-ggplot-intro.R

```
library(ggplot2)

exampledata <- read.csv("ggplot-example-
data.csv")

View(exampledata)

ggplot(data = exampledata, aes(x = A, y = B,
shape = D)) + geom_point()
```

ggplot2

Components of a ggplot2 plot

Data

Scales (map data to aesthetics)

Geometric objects

Coordinate system (a pie chart is just a bar chart in polar coords)

Statistics

Facets

4.

R & RSTUDIO

R

Free, open source programming language

Initially for statistics and graphics

Today: maps, publishing documents, interactive dashboards, blogging, etc.

Big community: learning resources, packages, events

RStudio

Free, open source client for R

rstudio.com

rstudio.cloud

Rstudio online

Runs in the browser,
no installation needed

Share R projects

rstudio.cloud

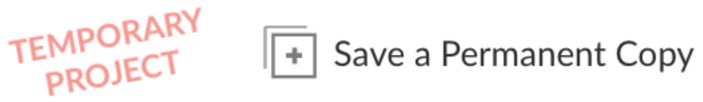
rstudio.cloud
Set up

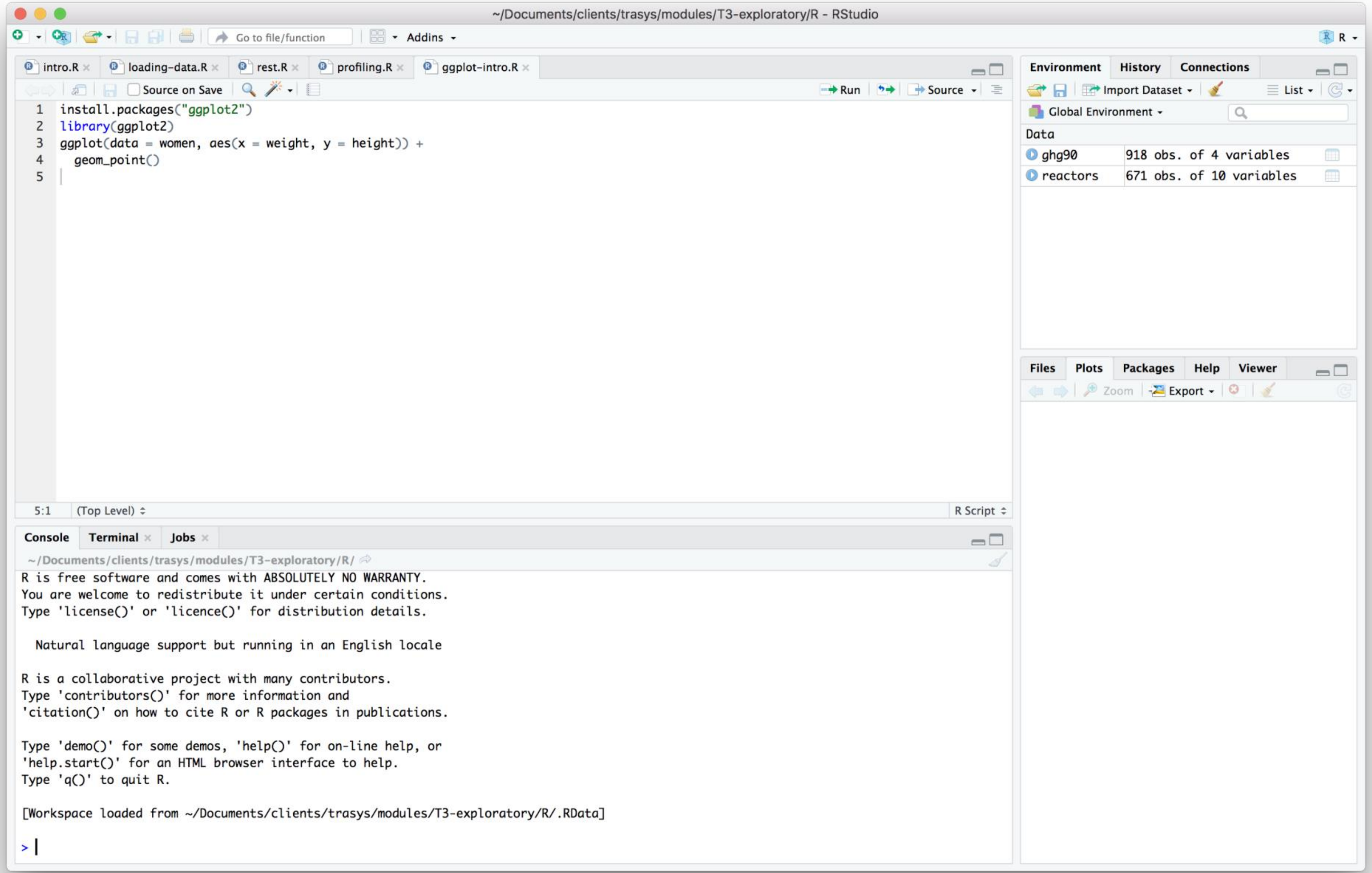
Go to rstudio.cloud

Click "Sign Up" and create an account

Go to rstudio.cloud/project/351652

Click "Save a permanent copy"

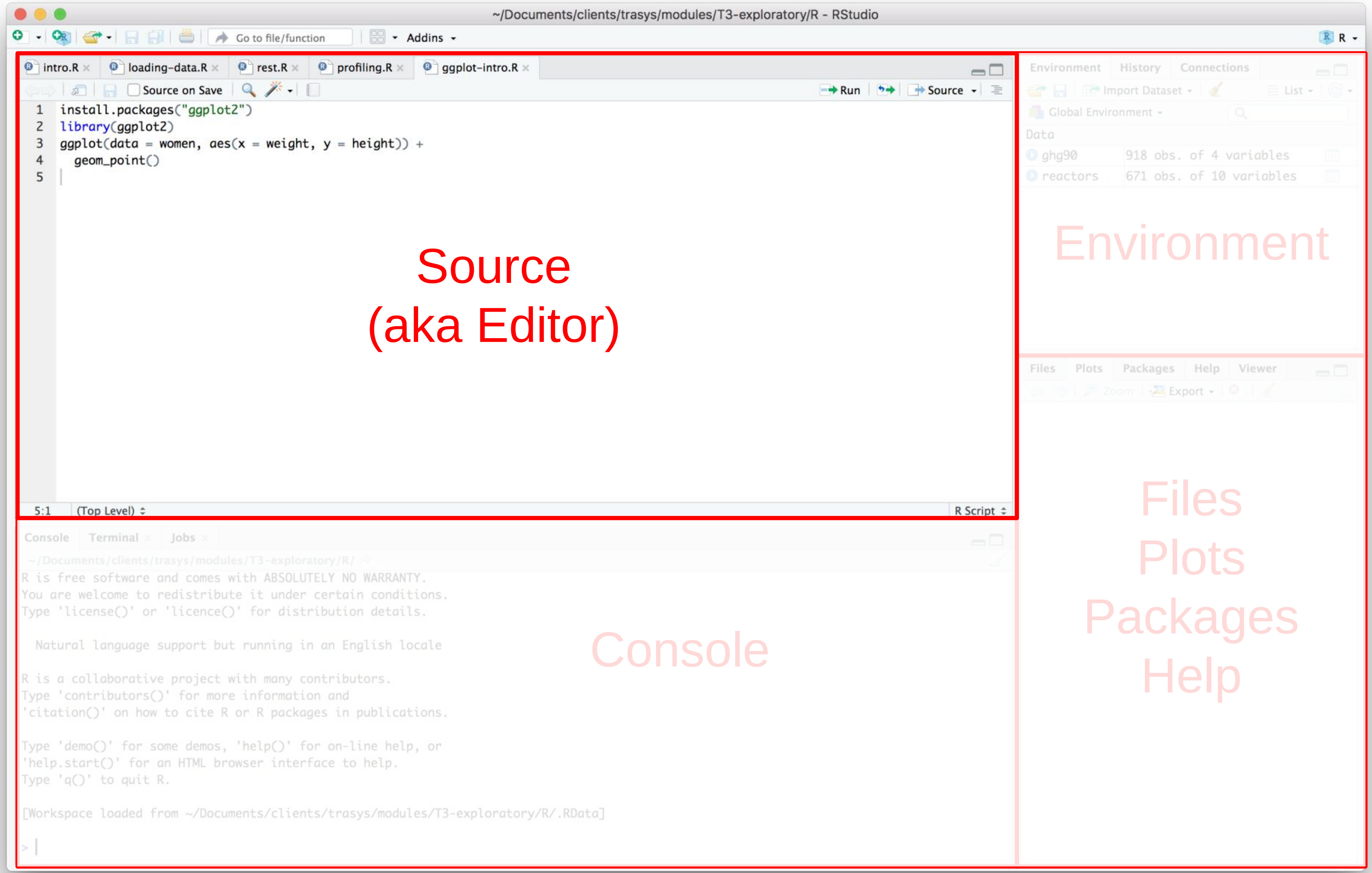




```r
install.packages("ggplot2")
library(ggplot2)
ggplot(data = women, aes(x = weight, y = height)) +
  geom_point()
```

Data	
ghg90	918 obs. of 4 variables
reactors	671 obs. of 10 variables

```
~/Documents/clients/trasys/modules/T3-exploratory/R/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/Documents/clients/trasys/modules/T3-exploratory/R/.RData]

> 
```

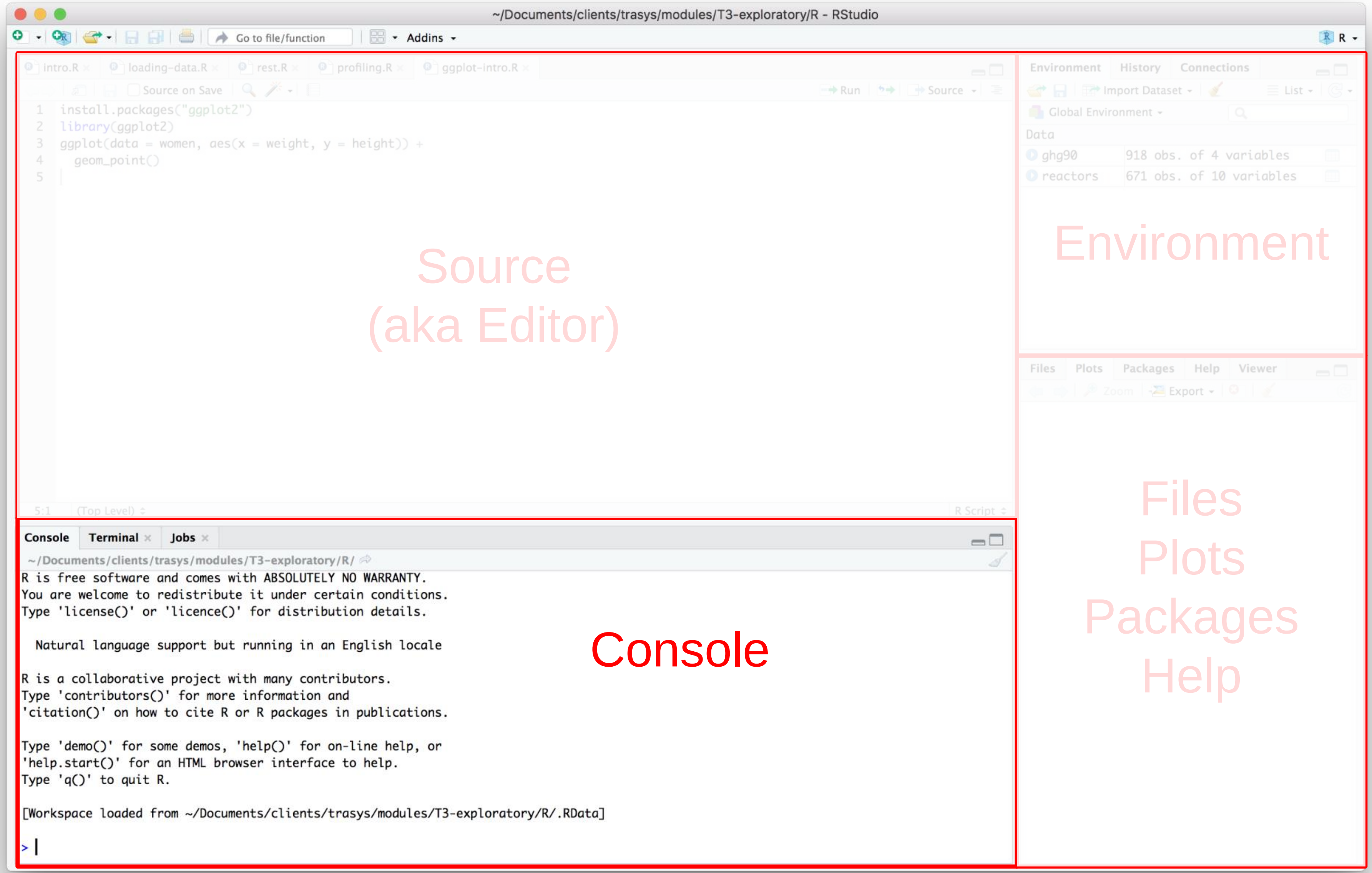


```
install.packages("ggplot2")
library(ggplot2)
ggplot(data = women, aes(x = weight, y = height)) +
  geom_point()
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/Documents/clients/trasys/modules/T3-exploratory/R/.RData]

>
```

RStudio interface

Source	Write scripts
	Preview data
	Uses tabs

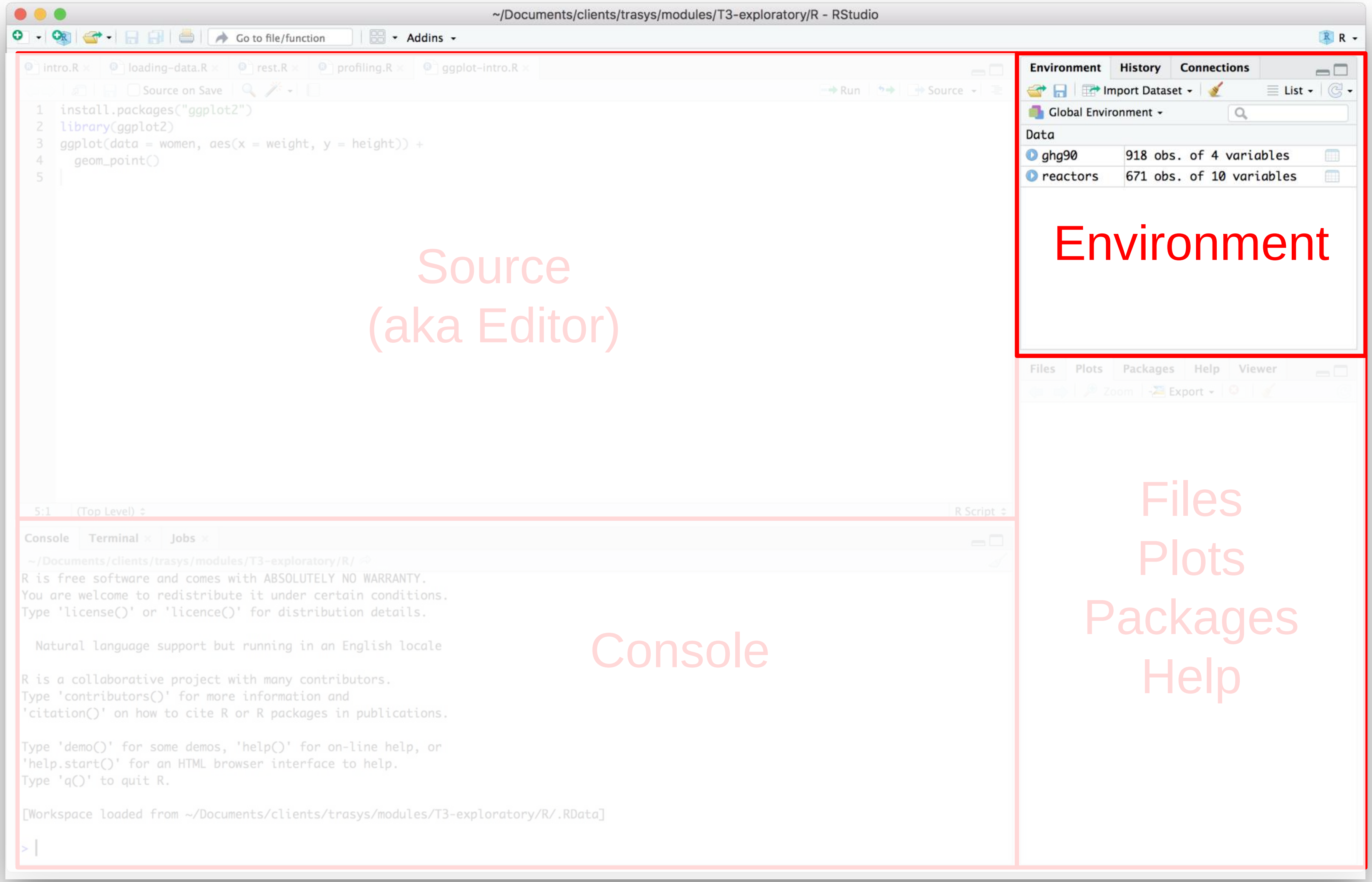


```r
install.packages("ggplot2")
library(ggplot2)
ggplot(data = women, aes(x = weight, y = height)) +
  geom_point()
```

```
~/Documents/clients/trasys/modules/T3-exploratory/R/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/Documents/clients/trasys/modules/T3-exploratory/R/.RData]

> 
```

RStudio interface

Console	Execute R code
	Output of code
	Errors and warnings



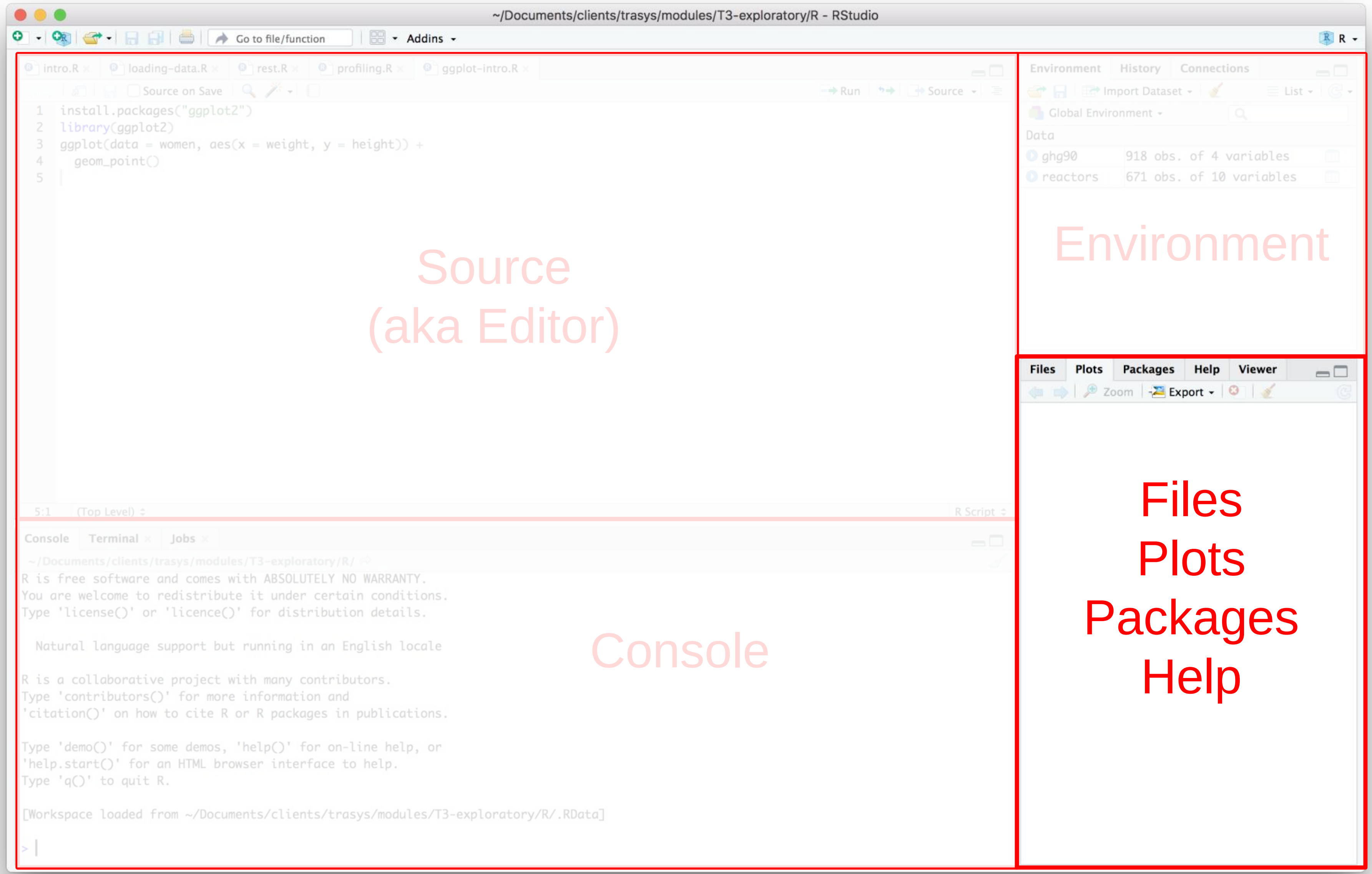
```
install.packages("ggplot2")
library(ggplot2)
ggplot(data = women, aes(x = weight, y = height)) +
  geom_point()
```

ghg90	918 obs. of 4 variables
reactors	671 obs. of 10 variables

```
~/Documents/clients/trasys/modules/T3-exploratory/R/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/Documents/clients/trasys/modules/T3-exploratory/R/.RData]

>
```

RStudio interface

Environment	Loaded objects
	History of commands
	"Import dataset" wizard



```r
install.packages("ggplot2")
library(ggplot2)
ggplot(data = women, aes(x = weight, y = height)) +
  geom_point()
```

Data	
ghg90	918 obs. of 4 variables
reactors	671 obs. of 10 variables

```
~/Documents/clients/trasys/modules/T3-exploratory/R/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/Documents/clients/trasys/modules/T3-exploratory/R/.RData]

> 
```

RStudio interface

Files	File explorer
Plots	ggplot output
Packages	Installed packages
Help	Documentation of functions and packages

R basics

Load data	"Import dataset" wizard
	`read.csv("mydatafile.csv")`
	`eurostat::get_eurostat("ilc_hch10")`
Assign	`x <- 5`
	`mydataframe <- read.csv(...)`
Inspecting data	`View(...)`
	`head(...)`
	Inspect object
Get help	`?read.csv`
	`?head`

R & Rstudio basics

Packages	`install.packages("ggplot2")`
	`library(ggplot2)`

Run code	"Run" button in Source pane Using the console
Saving script	File => Save as .R files

Exercise
4-load-data.R

`View(...)`	open data in the data viewer
`read.csv(...)`	load csv files
`library(...)`	load (installed) package
`readxl::read_excel(...)`	load Excel files and sheets
`?...`	open the documentation
`eurostat package`	find and load eurostat data

5.

DATA PROFILING

Data profiling

A first glimpse of the data

How big is the data?

What are the data types?

Missing data? Errors?

Starting to understand the data

Data profiling

Missing data	NA values
Inspect	Check the object under Environment
Run the data frame	Data frame will be outputted in the console

Exercise

5-1-data-profiling.R

`ncol(...)`	number of columns
`nrow(...)`	number of lines
`typeof(...)`	data type of a column
`str(...)`	compact display of R object
`summary(...)`	summary statistics
`visdat::vis_dat(...)`	visual profile of dataframe

Exercise

5-2-check-assumptions.R

`geom_col()`	bar and column charts
`dplyr::filter()`	from the dplyr package, filter data on 1 or more conditions

6.

VISUALISING DISTRIBUTIONS

Visualising distributions

1 numerical column

1 optional categorical column

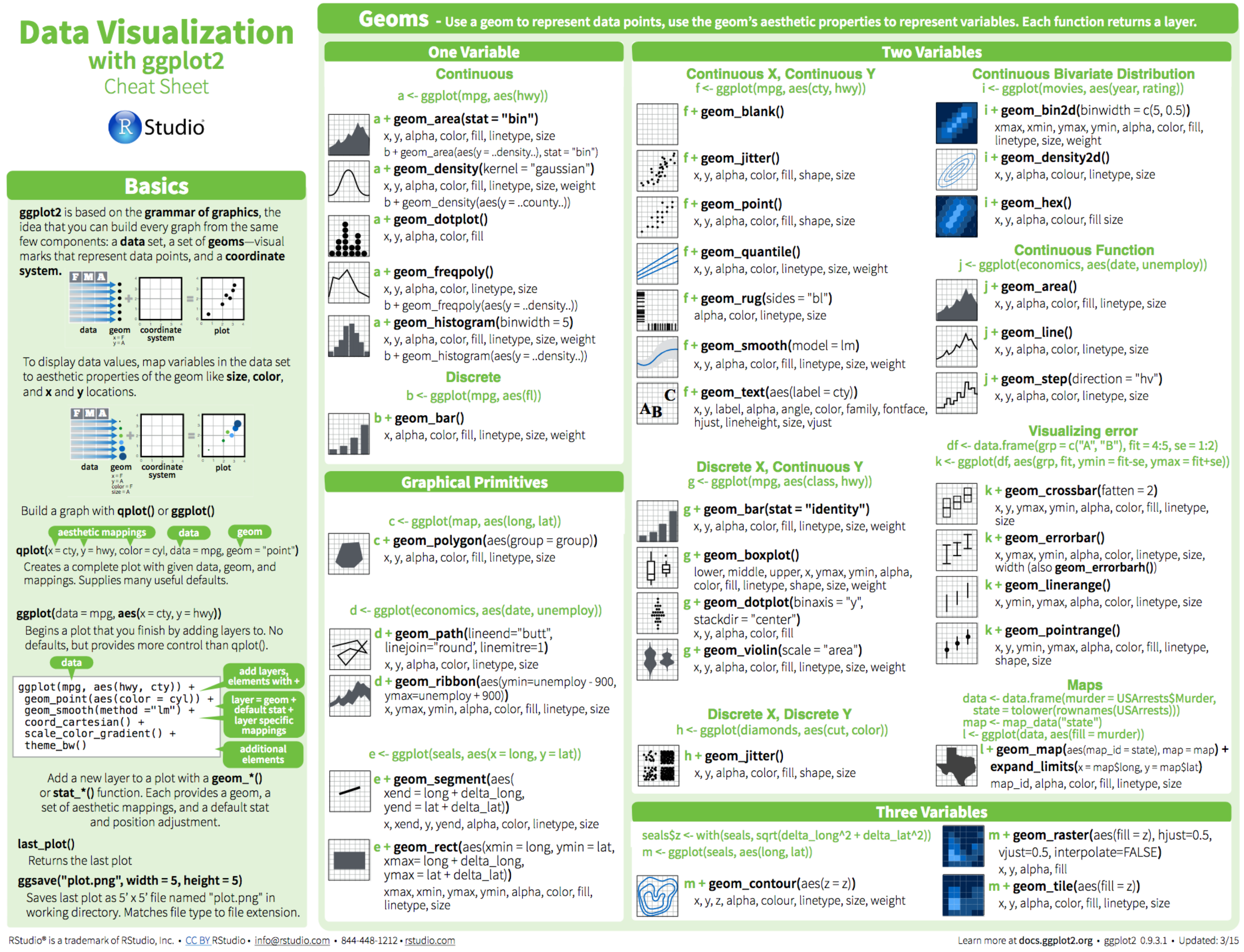
Range of the distribution

Shape of the distribution (symmetrical, bimodal, long tails, ...)

Central values

Outliers

ggplot2
Cheat sheet



Data Visualization
with ggplot2
Cheat Sheet

RStudio

Basics

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

F M A
data
geom
x = F
y = A
coordinate system
plot

To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.

Build a graph with **qplot()** or **ggplot()**

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

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**
Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

add layers, elements with +
layer = geom + default stat + layer specific mappings
additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

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.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

a + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path(**lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, linetype, size

d + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)**
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment(**aes(
xend = long + delta_long,
yend = lat + delta_lat)**)**
x, xend, y, yend, alpha, color, linetype, size

e + **geom_rect(**aes(xmin = long, ymin = lat,
xmax= long + delta_long,
ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

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

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug(**sides = "bl"**)**
alpha, color, linetype, size

f + **geom_smooth(**model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text(**aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

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

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

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

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d(**binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step(**direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder,
  state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + **geom_map(**aes(map_id = state), map = map**) +**
expand_limits(x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

m + **geom_contour(**aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

m + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

m + **geom_tile(**aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size

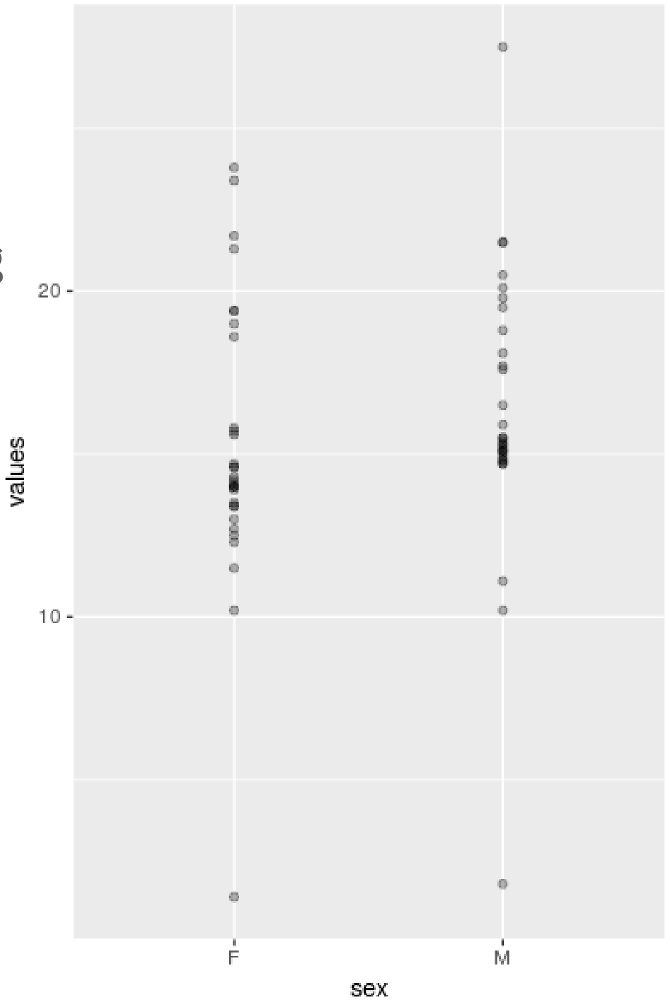
Learn more at **docs.ggplot2.org** • ggplot2 0.9.3.1 • Updated: 3/15

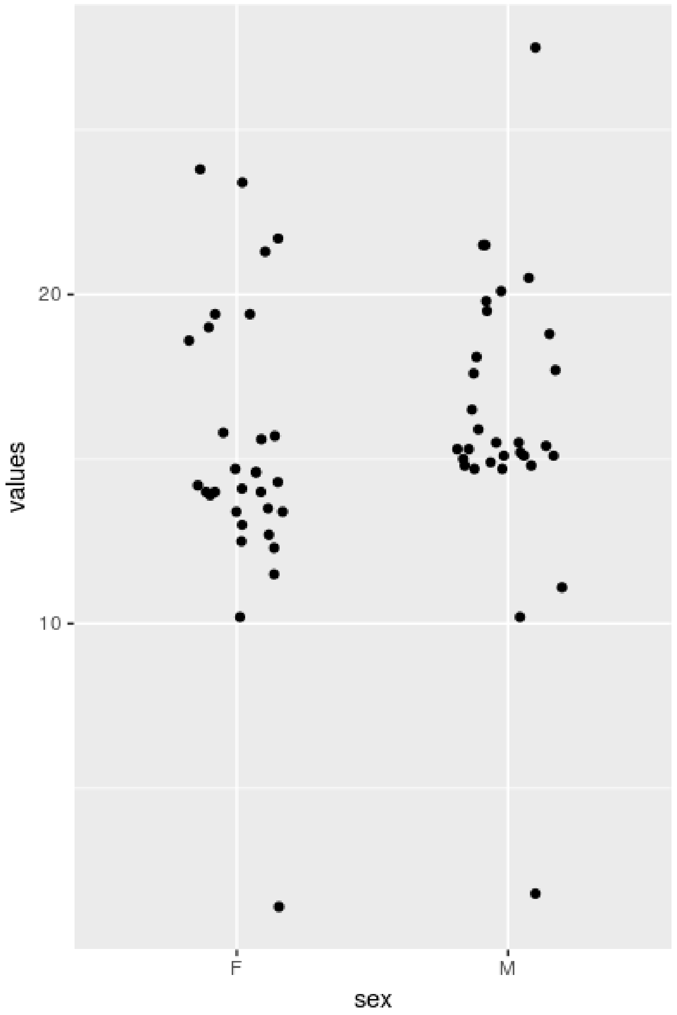
Distributions
Points

The easiest way to visualise a distribution: plotting each value

`geom_point()`

If you have overlap: use alpha or `geom_jitter()`

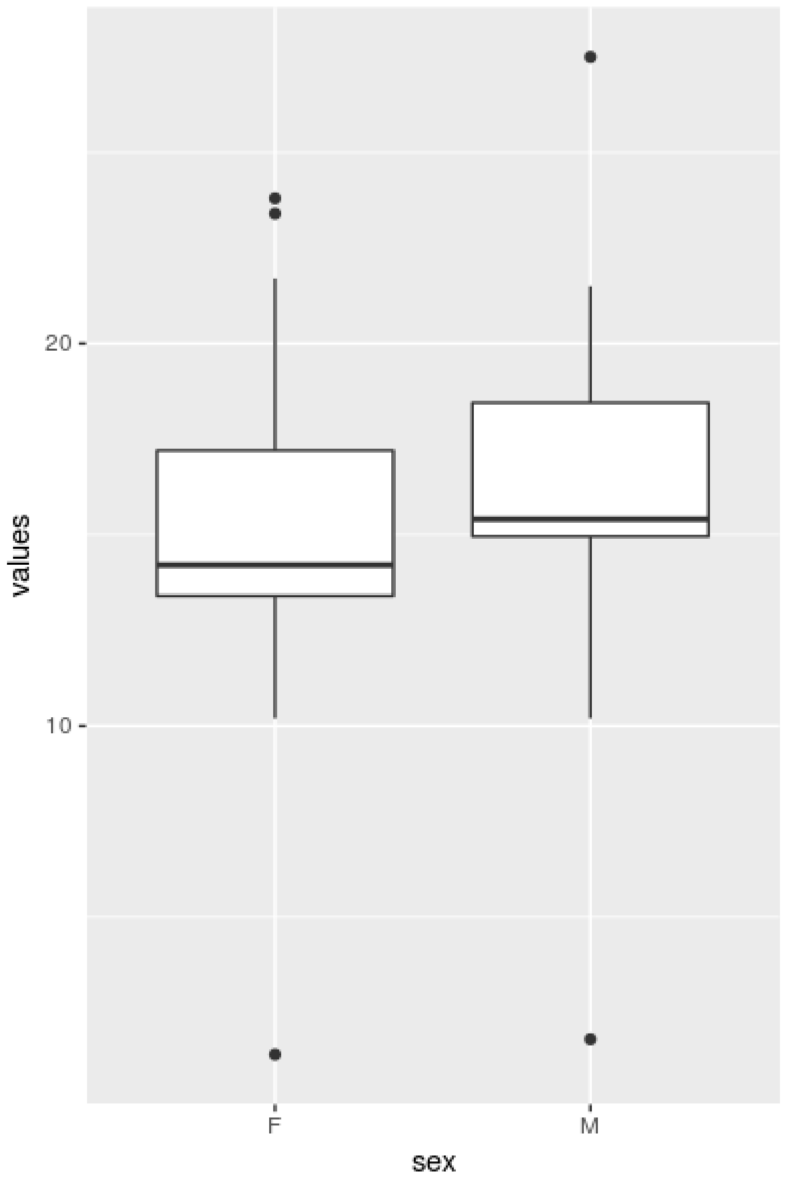




Distributions
Boxplots

geom_boxplot()

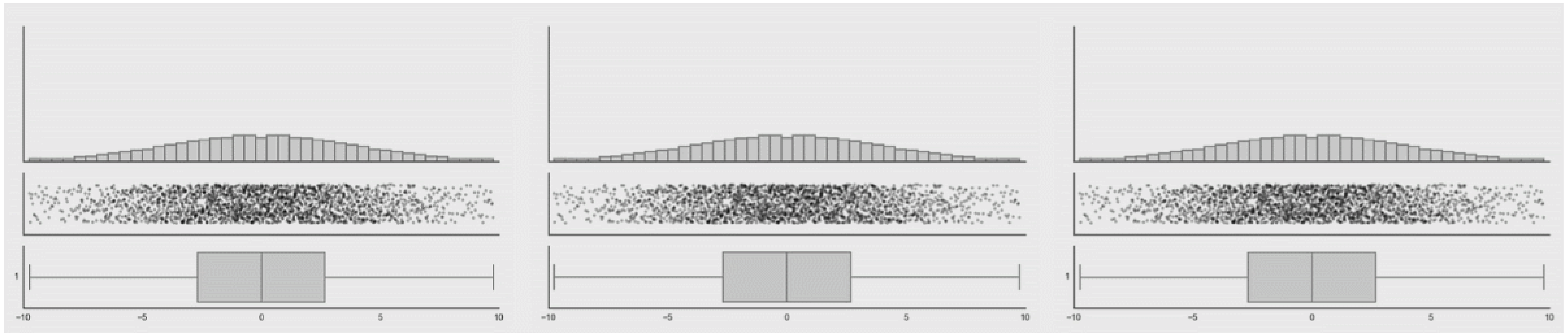
ggplot calculates the medians, quartiles and outliers for us



Exercise
6-1-distributions-points-boxplots.R

`geom_point()`	plot points (circles, symbols)
`geom_boxplot()`	plot boxplots
`geom_jitter()`	add jittering
`geom_text()`	plot text
`geom_1() + geom_2()`	combine geoms

Distributions
Boxplots alternatives

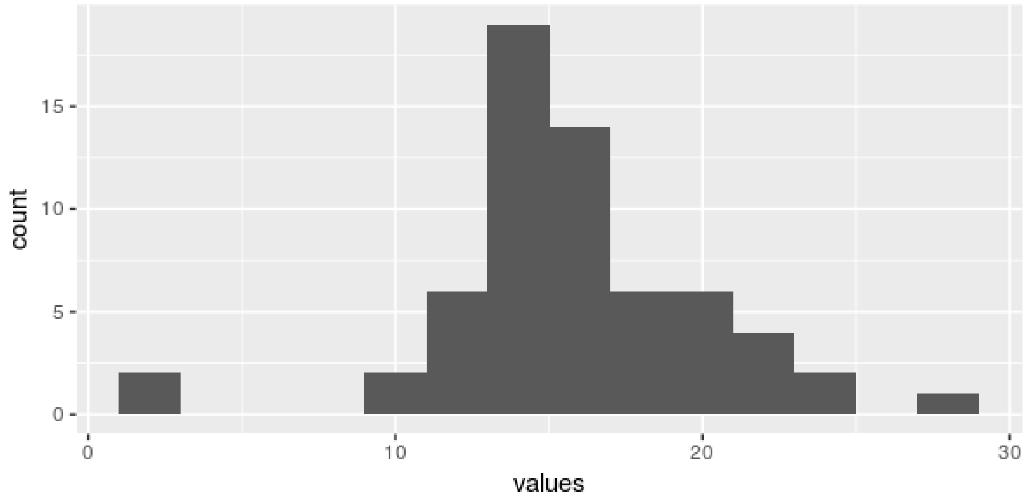


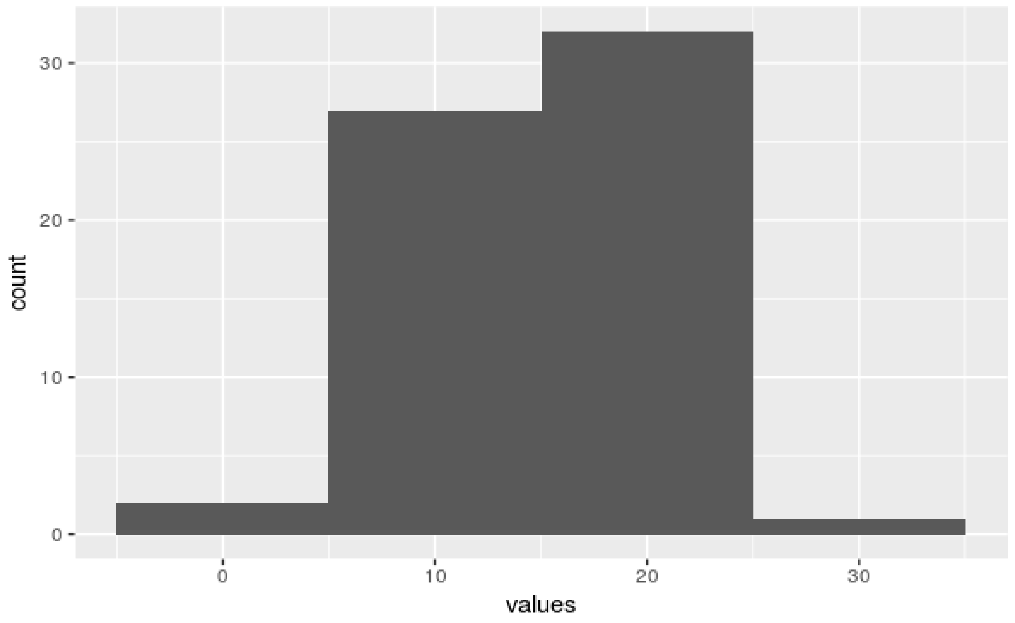
Distributions
Histograms

geom_histogram()

ggplot does the counting for us

binwidth/bins parameters is very important: try different bin sizes



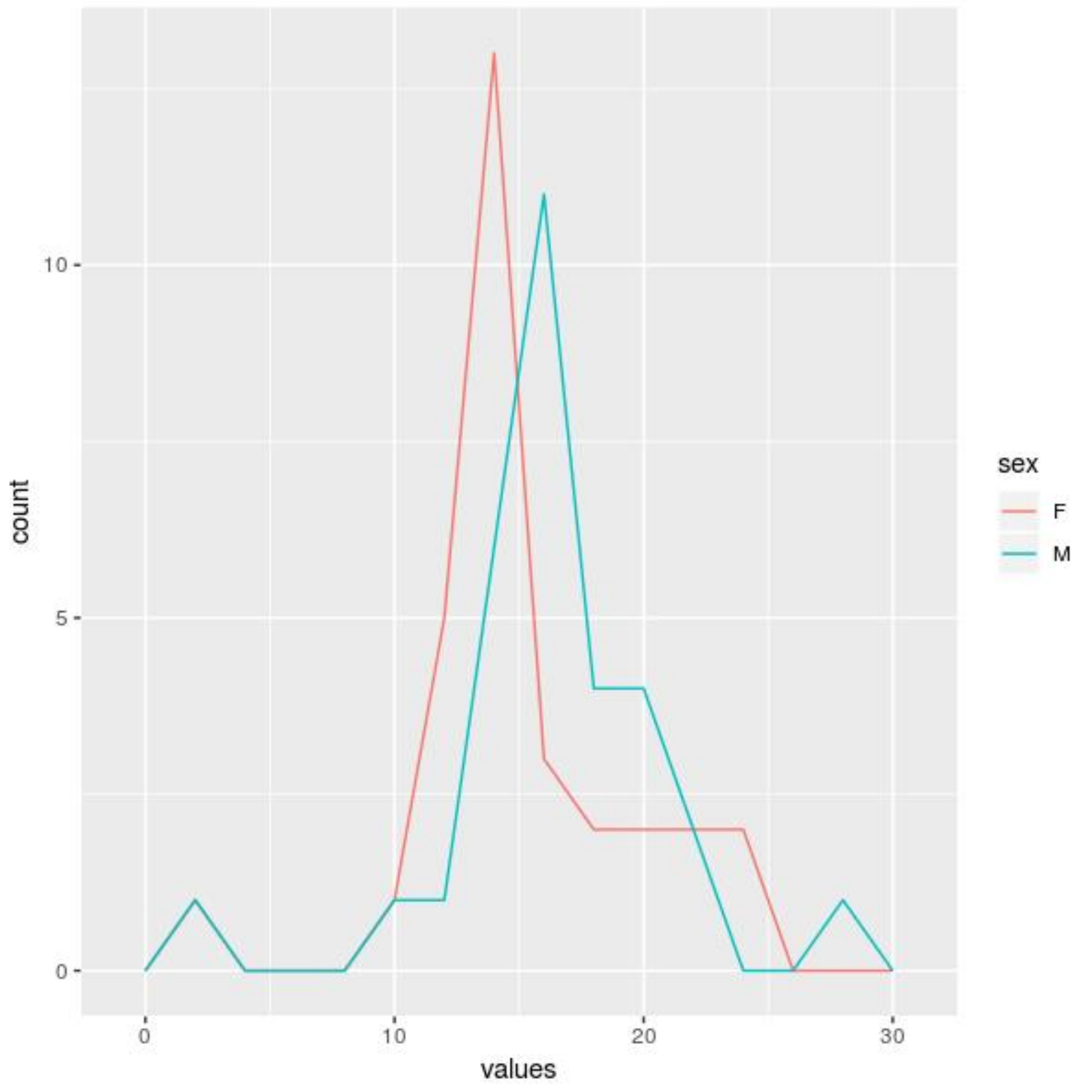


Distributions
Frequency polygons

geom_freqpoly()

Same as histogram, but with lines instead of bars

Better suited to compare distributions between different categories

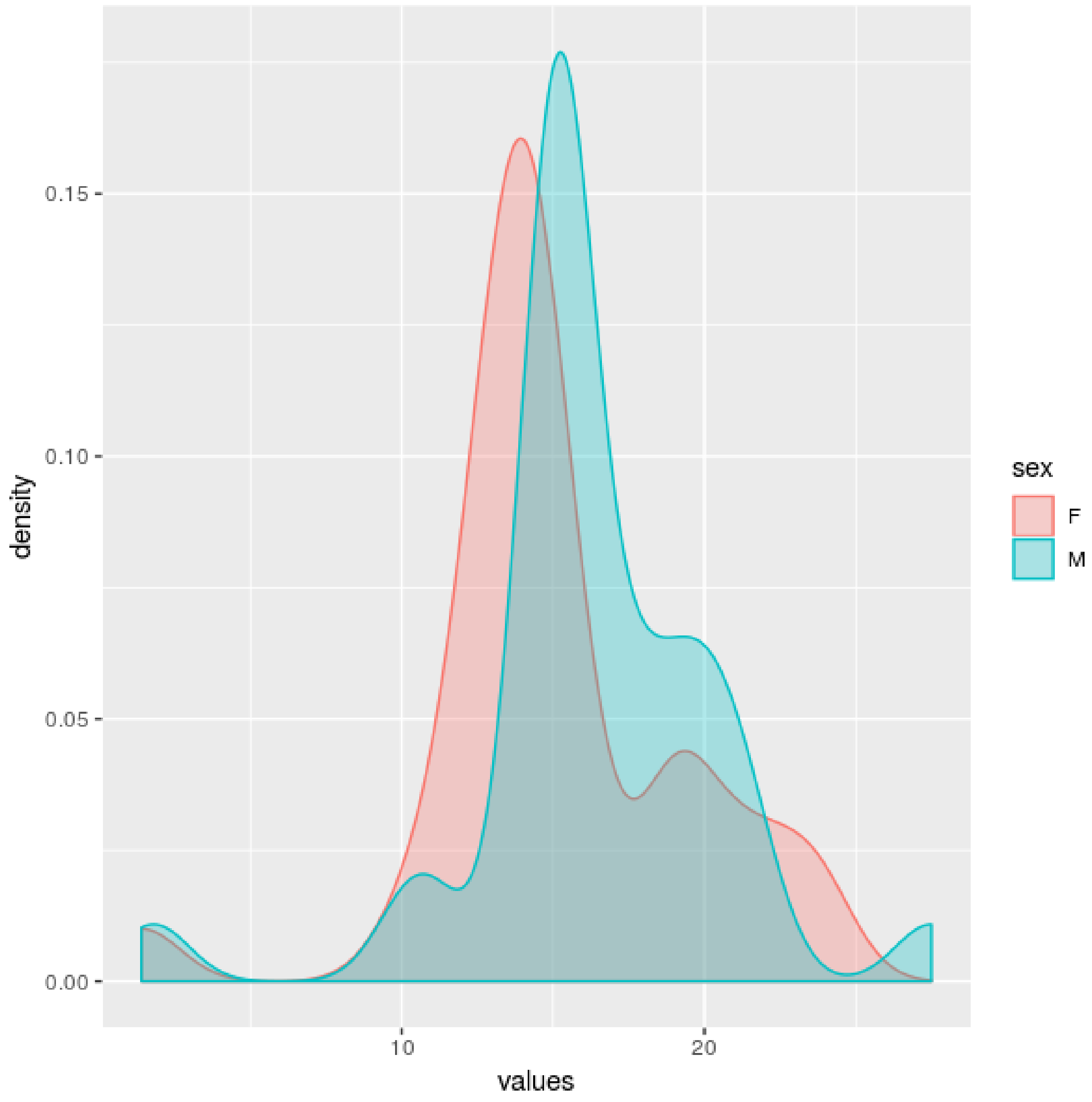


Distributions
Density plot

geom_density()

Smooth version of geom_freqpoly()

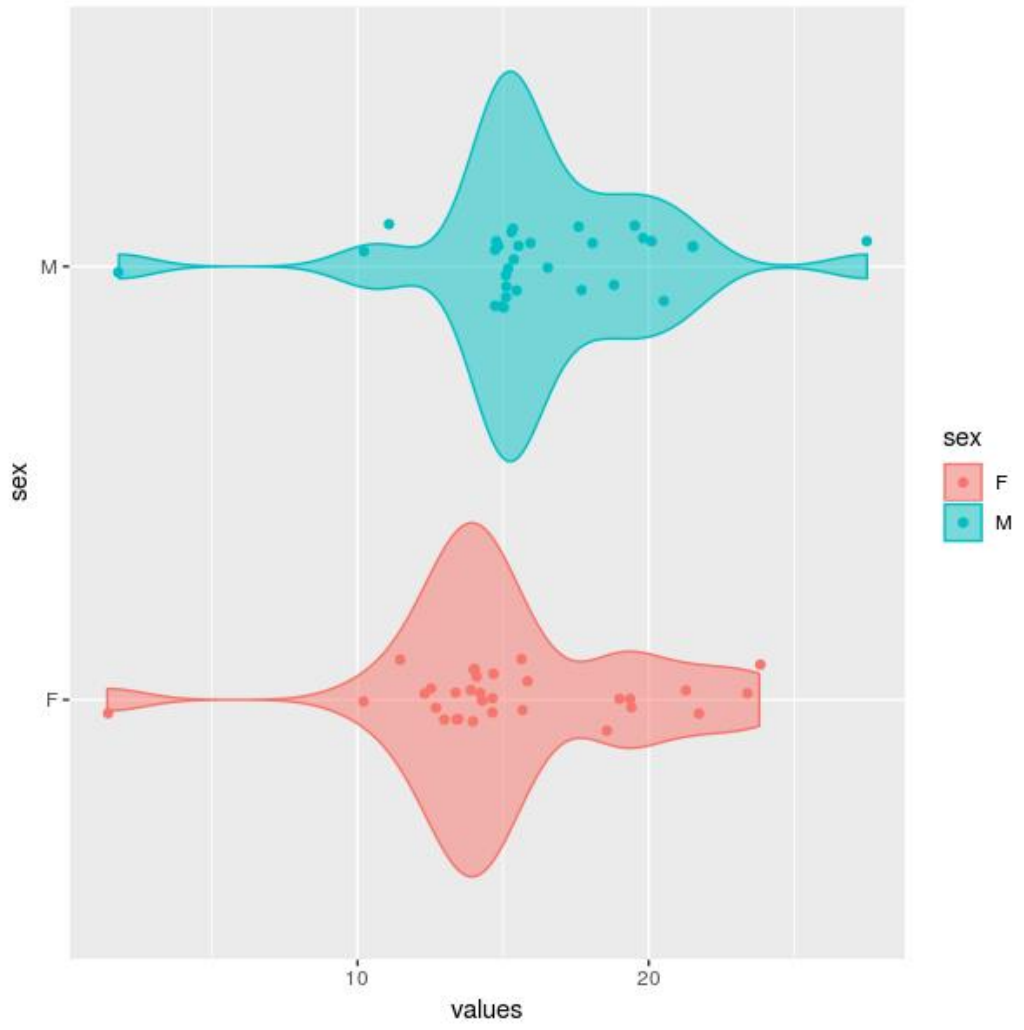
Use when underlying data is smooth



Distributions
Violin plot

geom_violin()

Compact, mirrored version of geom_density()



Exercise

6-2-distributions-othergeoms.R

`geom_histogram()`	plot histograms
`geom_freqpoly()`	plot frequency polygons
`geom_density()`	plot density estimates
`geom_violin()`	plot violin plots

7.

VISUALISING COVARIATION

Visualising covariation

Two numerical variables

Identify correlations

Optionally one categorical variable

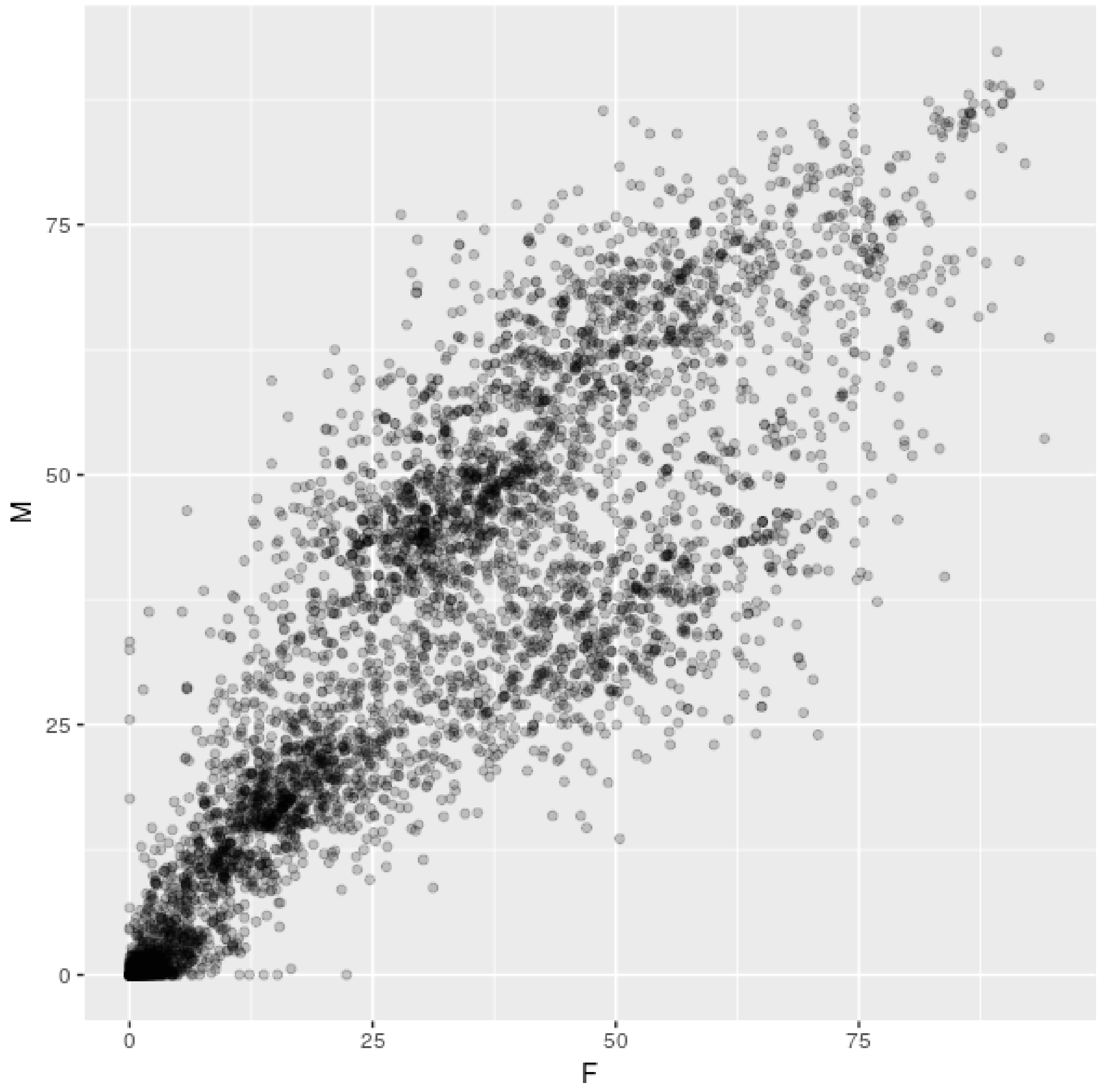
Detect outliers

Covariation
Scatterplots

geom_point()

geom_text()
geom_label()

Use **geom_jitter()** for discrete or categorical variables



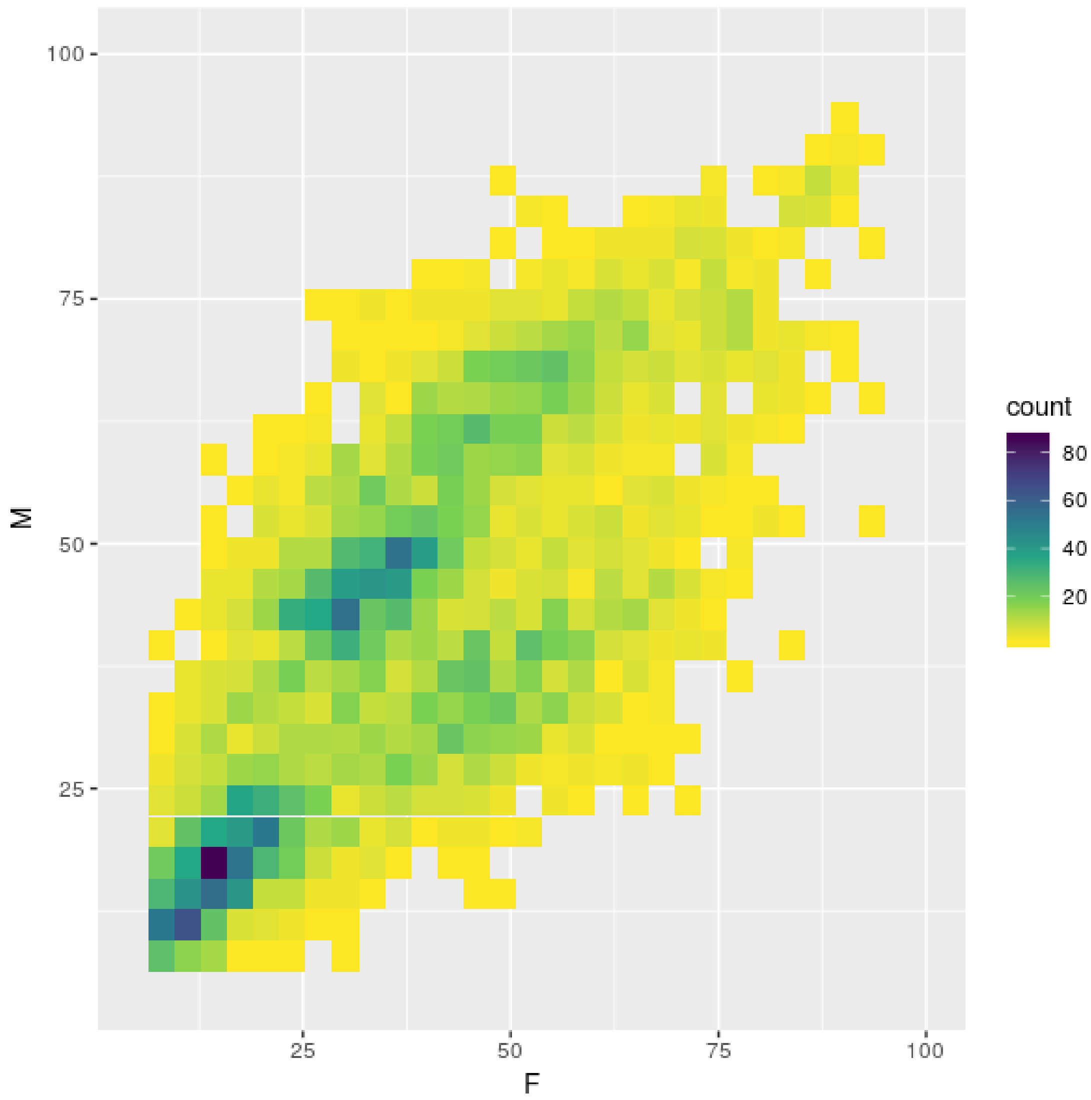
Covariation
Scatterplots

Overcome overplotting by binning with

geom_bin2d()

geom_hex()

Or use transparency: alpha



Exercise
7-1-scatterplots.R

`geom_point()`	plot points
`geom_bin2d()`	bin points in rectangles
`geom_hex()`	bin points in hexagons
`scale_x_continuous()`	customise continuous scale
`scale_fill_continuous()`	customise color scale

Exercise: Why visualise?
7-2-anscombe.R

Dataset "anscombe"

`View(anscombe)`

Check the means with
`summary(anscombe)`

Calculate some stats:

```
sd(anscombe$x1)
cor(anscombe$x1,
anscombe$y1)
```

Now make scatterplots:

x1 ~ y1

x2 ~ y2

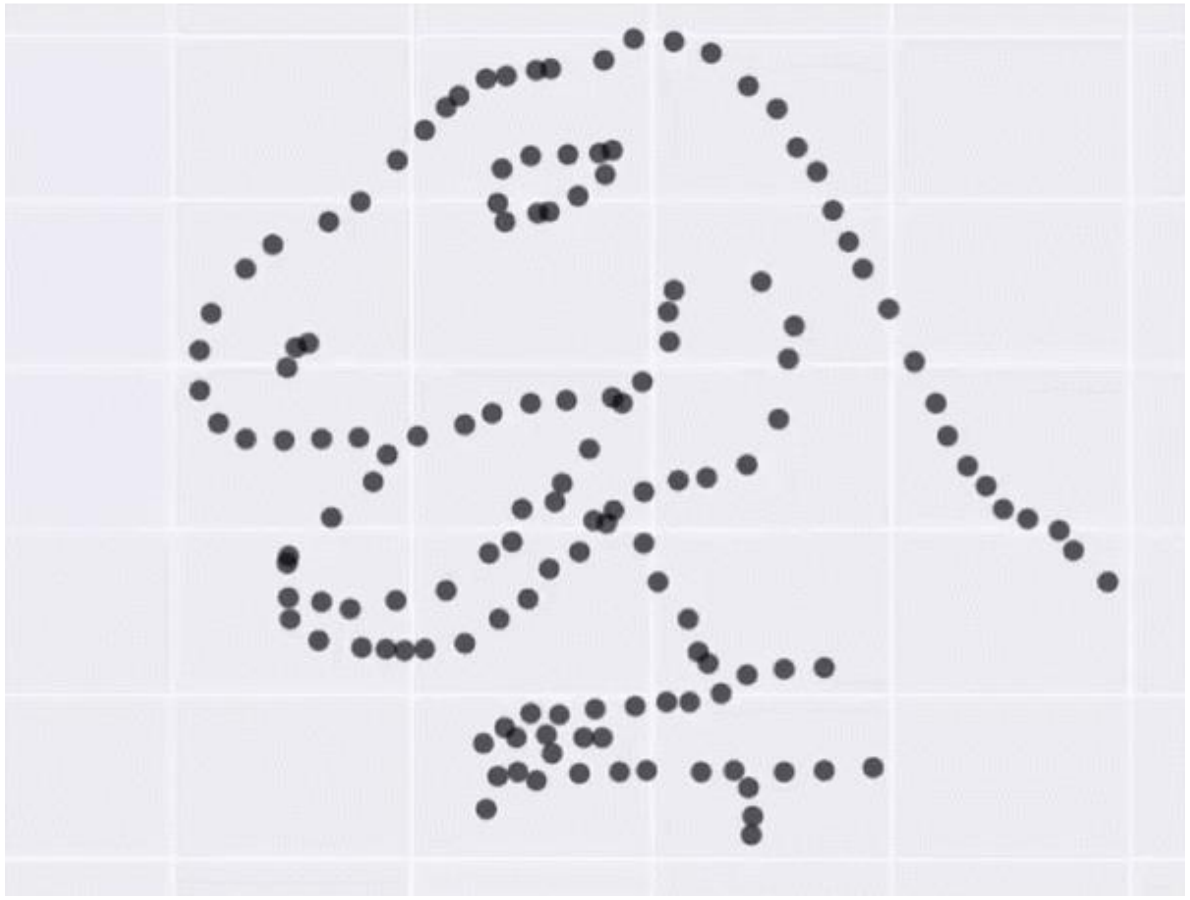
x3 ~ y3

x4 ~ y4

What do you see?

EDA
Why visualise?

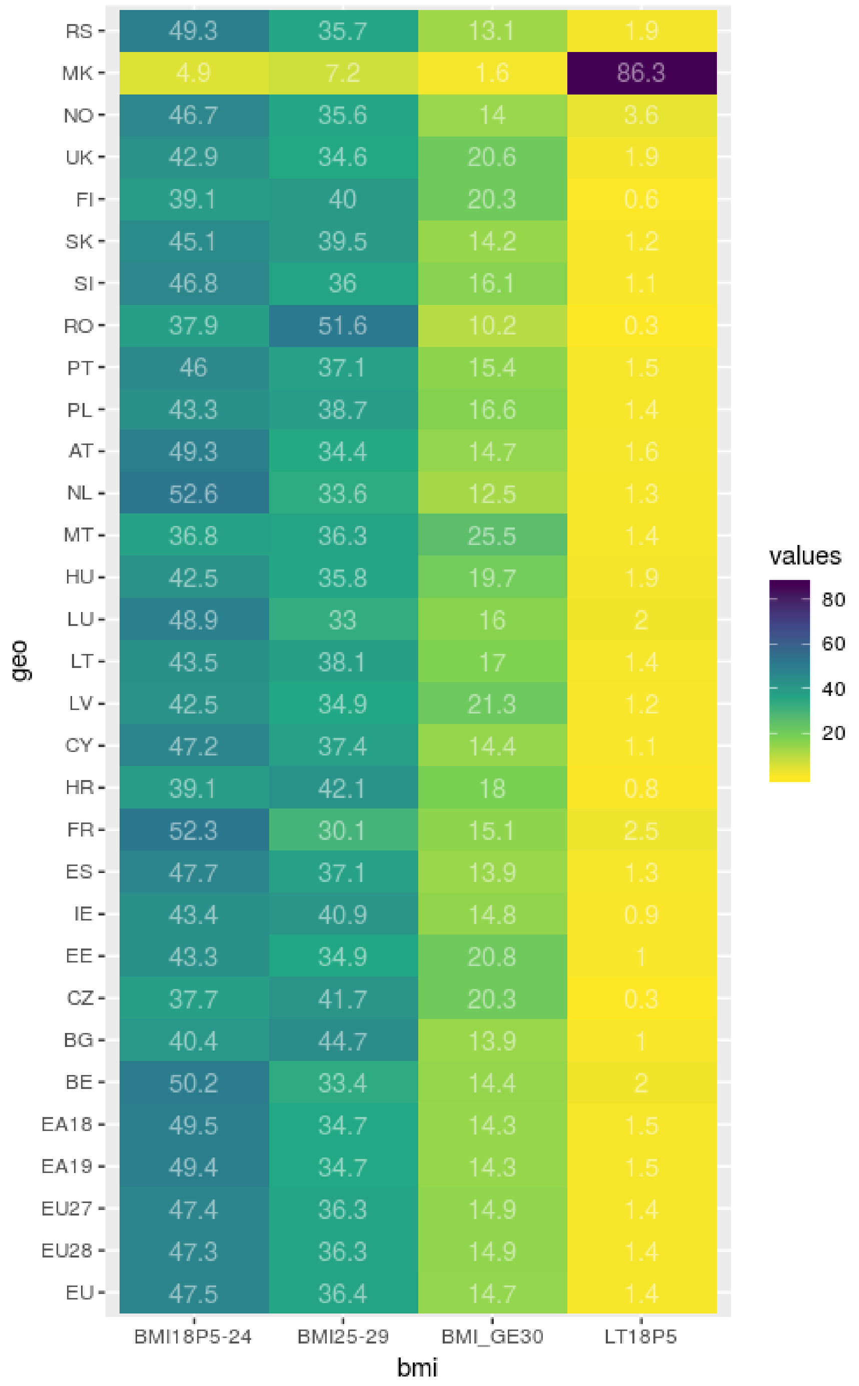
Datasaurus



Covariation
Heatmaps

2 categorical or discrete variables + 1 numerical variable

`geom_tile()`



Exercise
7-3-heatmaps.R

`geom_tile()`	plot rectangles

8.

MULTIDEMENSIONAL DATA

Multidimensional
Add aesthetics

geoms have many aesthetics. For example, geom_point() understands the following:

x*

y*

color

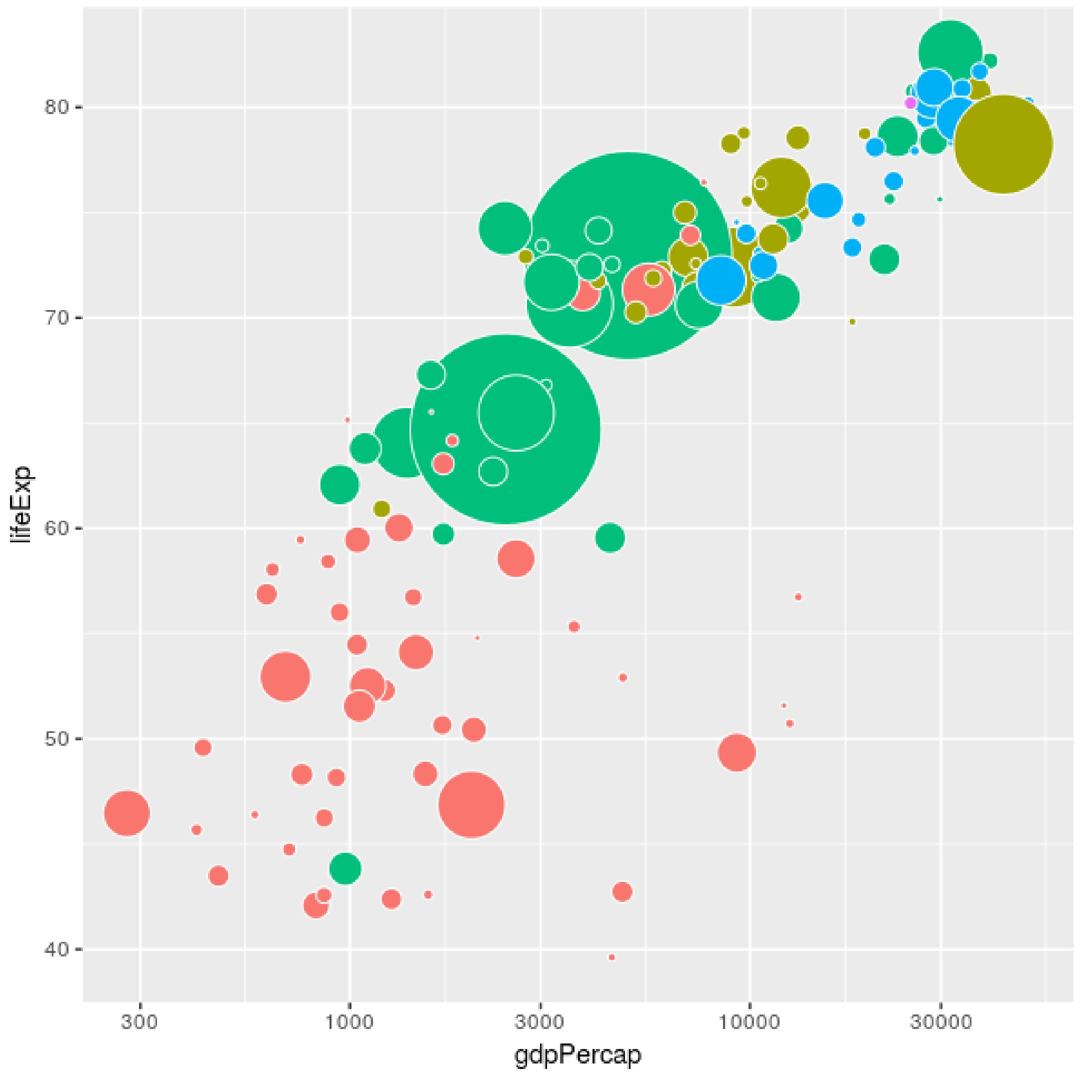
fill

group

shape

size

stroke



Exercise: Why visualize?

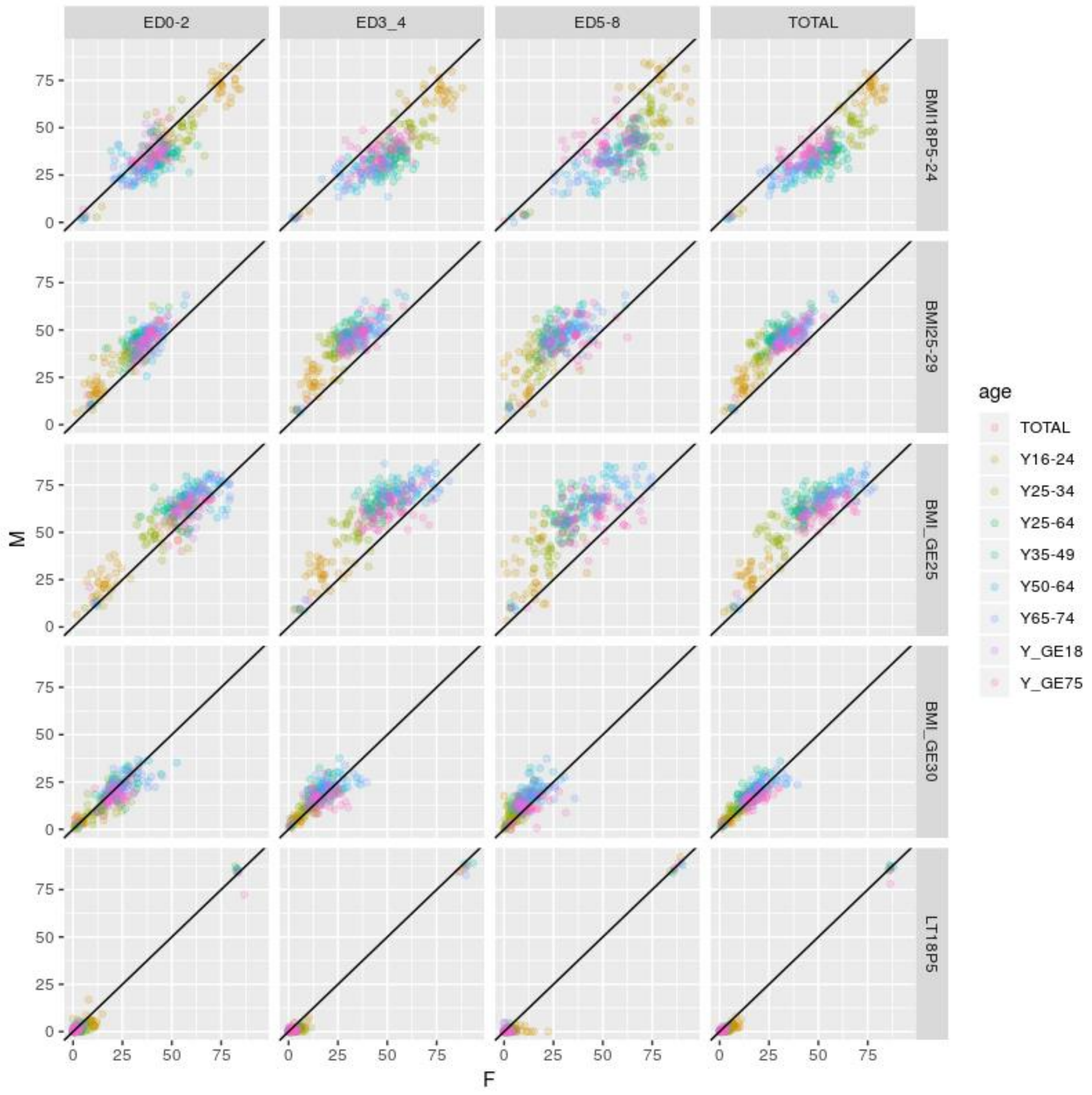
8-1-gapminder.R

`+ aes()`	Add aesthetic to a plot
`scale_x_log10()`	Logarithmic scale
`scale_size_area()`	Size scale

Multidimensional **Facetting**

Make small multiple charts by facetting plots with **facet_wrap()** and **facet_grid()**

Free vs fixed scales



Exercise
8-2-facetting.R

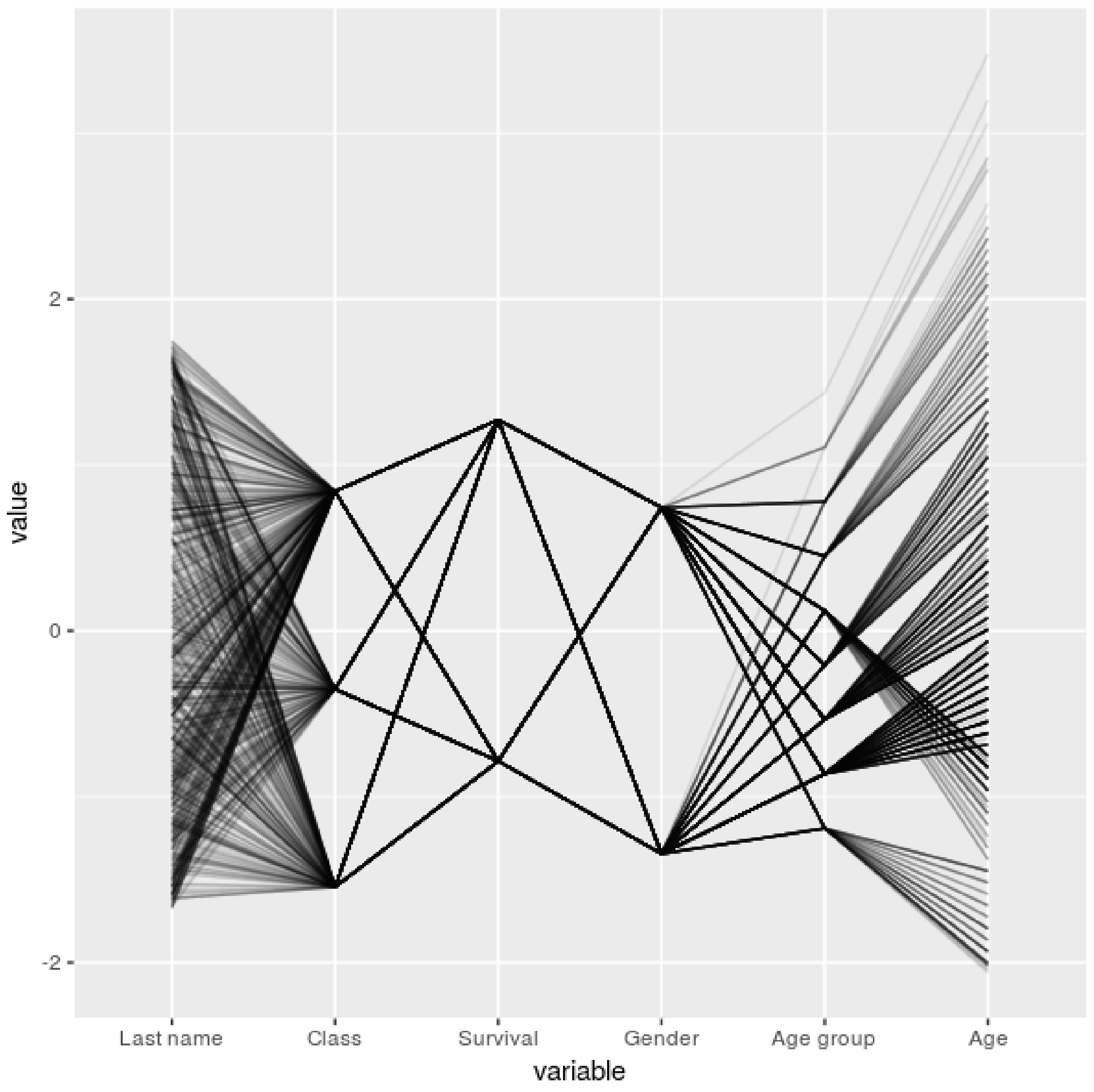
`facet_wrap()`	plot split into small multiples for 1 variable
`facet_grid()`	plot split into small multiples for 2 variables

Multidimensional
Parallel coordinates

Plot many dimensions and see correlation between them

GGally::ggparcoord()

Add boxplots



Exercise: Why visualize?

8-3-parallel-coordinates.R

`GGally::parcoord()`	plot parallel coordinate plot

Multidimensional
Multiple views & interactivity

Shiny package

Filters & interaction

Multiple visualisations

Example

Coordinated views

Example

9. VISUALISING TIME SERIES

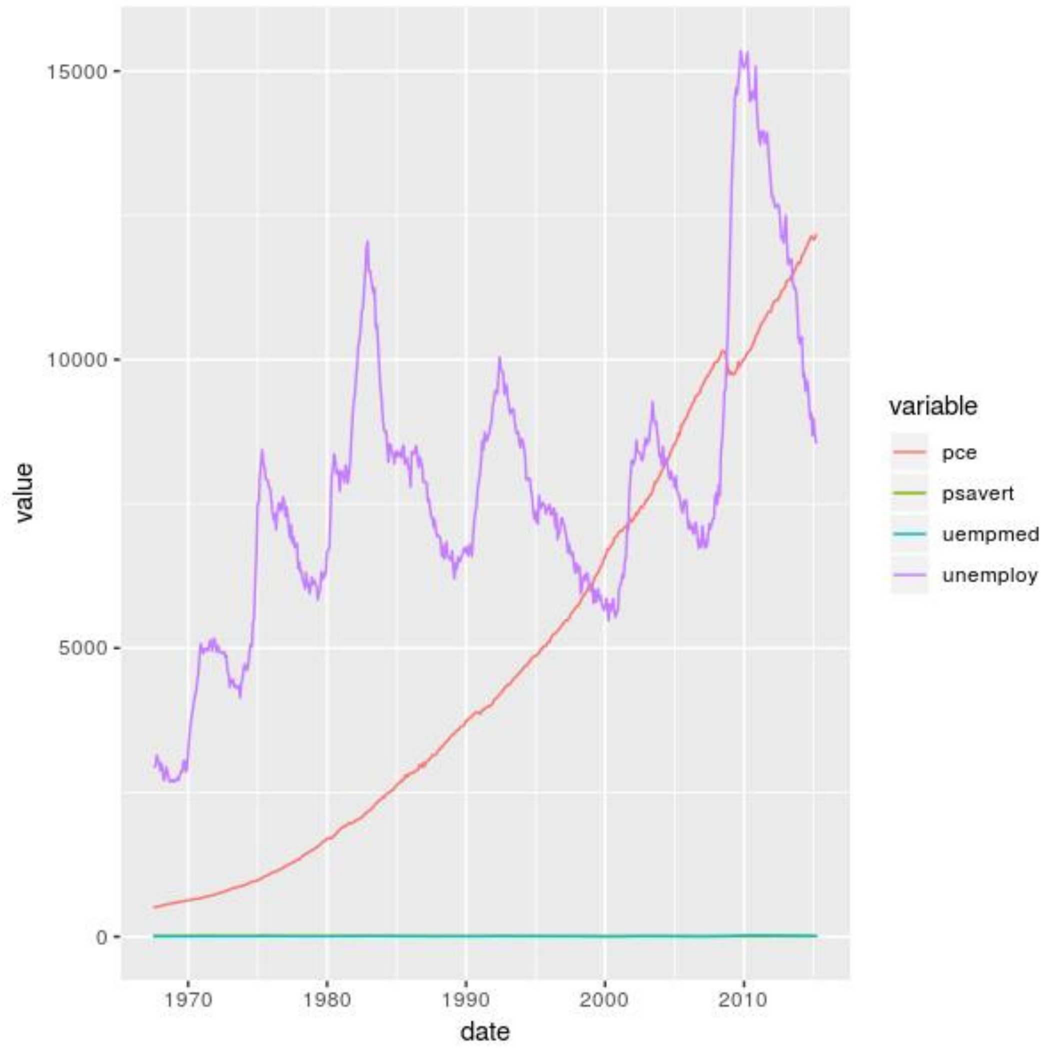
Time series
geom_line()

geom_line() with group or color aesthetic

geom_area()

scale_x_time()

scale_x_date()



Exercise
9-time-series.R

`geom_line()`	plot line charts
`geom_area()`	plot area charts
`scale_x_date()`	

10. SAVING AND SHARING VISUALISATIONS

Saving
ggsave()

Plots panel => Export	save a plot to your project/hard drive
ggsave()	file formats dpi units

Sharing
Rmarkdown

Mix text, R-code and R-output

Output to html, Word, Powerpoint, etc.

File => New file => R Markdown...

Example

(preparation for Why Budapest, Lithuania and Warsaw split themselves in two)

11.

OVERVIEW OF OTHER TOOLS

Visual EDA
Other tools

Python:
- pandas, seaborn, matplotlib
- altair

Without coding:
- Qlik
- Tableau

Q&A

Resources

Grammar of Graphics

A layered grammar of graphics

R & EDA

R for Data Science

ggplot2

Tidyverse: ggplot2

Data visualization: A practival introduction

Fundamentals of Data Visualisation

ggplot2 Cheat Sheet

Upcoming training & workshop sessions

Topic	Type of session	Lux + webex	Bxl
Telling your story through data visualisation	Training	25/06	28/06
Making great online data visualisations without coding	workshop	26/06	-
Going beyond bars and lines: practising non-standard data visualisation	Training	24/09	Sep-Oct
Making data visualisations like a pro: D3.js	Workshop	25/09	-
Applying data visualisation best practices in real use cases	workshop	24/10	-

and webinars (topic like for the trainings) … stay tuned!

Materials will be published on https://data.europa.eu/euodp/en/knowledge-center