Introduction to use of R in Data Analysis: Data cleaning, Data exploration and Data visualization

Umar Ahmad

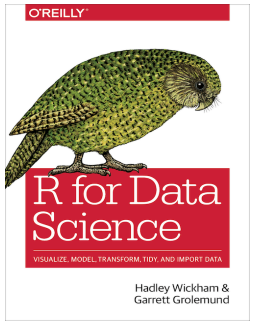
September 30, 2020

GENOM MALAYSIA
Malaysia Genome Institute

@babasaraki1

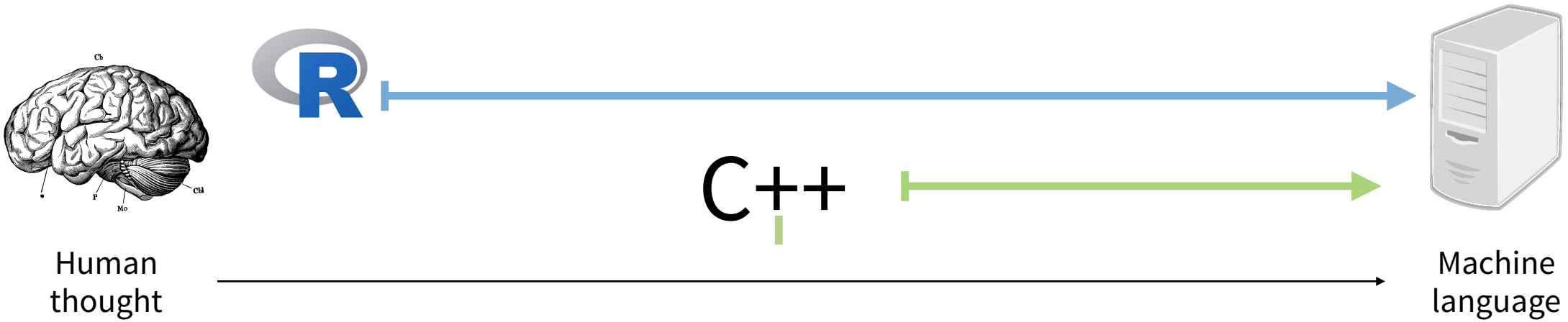
Data Cleaning

Umar Ahmad



This webinar is will be based on
R for Data Science http://r4ds.had.co.nz/

R - A computer language for scientists

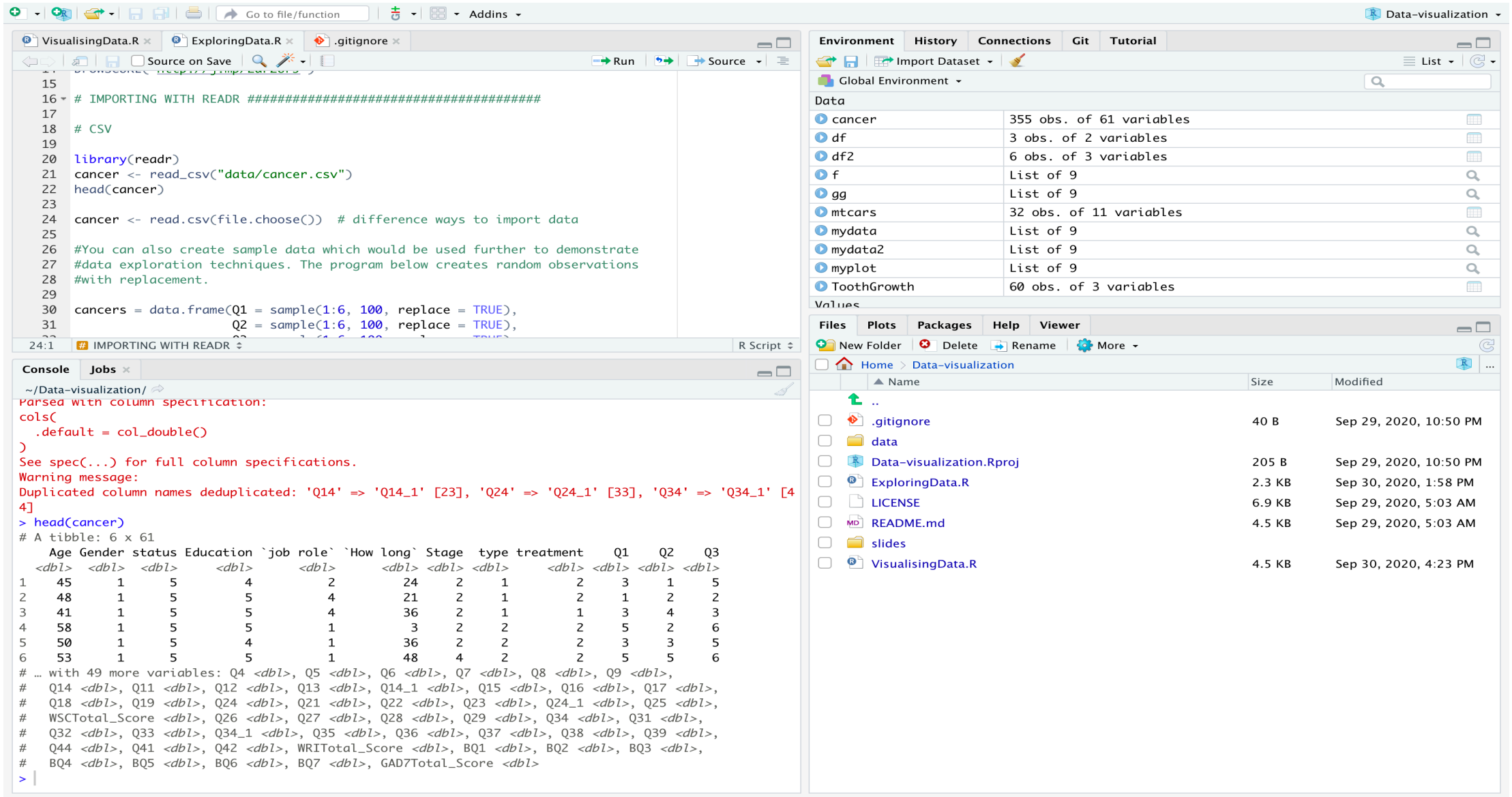


RStudio

R: Engine	RStudio: Dashboard
	

Courtesy Modern Dive

RStudio


Go to file/function
Addins
Data-visualization
VisualisingData.R
ExploringData.R
.gitignore
Source on Save
Run
Source
15
16 # IMPORTING WITH READR ##
17
18 # CSV
19
20 library(readr)
21 cancer <- read_csv("data/cancer.csv")
22 head(cancer)
23
24 cancer <- read.csv(file.choose()) # difference ways to import data
25
26 #You can also create sample data which would be used further to demonstrate
27 #data exploration techniques. The program below creates random observations
28 #with replacement.
29
30 cancers = data.frame(Q1 = sample(1:6, 100, replace = TRUE),
31 Q2 = sample(1:6, 100, replace = TRUE),
24:1
IMPORTING WITH READR
R Script
Console
Jobs
~/Data-visualization/
Parsed with column specification:
cols(
.default = col_double()
)
See spec(...) for full column specifications.
Warning message:
Duplicated column names deduplicated: 'Q14' => 'Q14_1' [23], 'Q24' => 'Q24_1' [33], 'Q34' => 'Q34_1' [44]
> head(cancer)
A tibble: 6 x 61
Age Gender status Education `job role` `How long` Stage type treatment Q1 Q2 Q3
<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 45 1 5 4 2 24 2 1 2 3 1 5
2 48 1 5 5 4 21 2 1 2 1 2 2
3 41 1 5 5 4 36 2 1 1 3 4 3
4 58 1 5 5 1 3 2 2 2 5 2 6
5 50 1 5 4 1 36 2 2 2 3 3 5
6 53 1 5 5 1 48 4 2 2 5 5 6
… with 49 more variables: Q4 <dbl>, Q5 <dbl>, Q6 <dbl>, Q7 <dbl>, Q8 <dbl>, Q9 <dbl>,
Q14 <dbl>, Q11 <dbl>, Q12 <dbl>, Q13 <dbl>, Q14_1 <dbl>, Q15 <dbl>, Q16 <dbl>, Q17 <dbl>,
Q18 <dbl>, Q19 <dbl>, Q24 <dbl>, Q21 <dbl>, Q22 <dbl>, Q23 <dbl>, Q24_1 <dbl>, Q25 <dbl>,
WSCTotal_Score <dbl>, Q26 <dbl>, Q27 <dbl>, Q28 <dbl>, Q29 <dbl>, Q34 <dbl>, Q31 <dbl>,
Q32 <dbl>, Q33 <dbl>, Q34_1 <dbl>, Q35 <dbl>, Q36 <dbl>, Q37 <dbl>, Q38 <dbl>, Q39 <dbl>,
Q44 <dbl>, Q41 <dbl>, Q42 <dbl>, WRITotal_Score <dbl>, BQ1 <dbl>, BQ2 <dbl>, BQ3 <dbl>,
BQ4 <dbl>, BQ5 <dbl>, BQ6 <dbl>, BQ7 <dbl>, GAD7Total_Score <dbl>
>
Environment
History
Connections
Git
Tutorial
Import Dataset
List
Global Environment
Data
cancer 355 obs. of 61 variables
df 3 obs. of 2 variables
df2 6 obs. of 3 variables
f List of 9
gg List of 9
mtcars 32 obs. of 11 variables
mydata List of 9
mydata2 List of 9
myplot List of 9
ToothGrowth 60 obs. of 3 variables
Values
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Data-visualization
Name
Size
Modified
..
.gitignore 40 B Sep 29, 2020, 10:50 PM
data
Data-visualization.Rproj 205 B Sep 29, 2020, 10:50 PM
ExploringData.R 2.3 KB Sep 30, 2020, 1:58 PM
LICENSE 6.9 KB Sep 29, 2020, 5:03 AM
README.md 4.5 KB Sep 29, 2020, 5:03 AM
slides
VisualisingData.R 4.5 KB Sep 30, 2020, 4:23 PM

```
> head(cancer)
# A tibble: 6 x 61
    Age Gender status Education `job role` `How long` Stage  type treatment    Q1    Q2    Q3
  <dbl>  <dbl>  <dbl>     <dbl>      <dbl>      <dbl> <dbl> <dbl>     <dbl> <dbl> <dbl> <dbl>
1    45      1      5         4          2         24     2     1         2     3     1     5
2    48      1      5         5          4         21     2     1         2     1     2     2
3    41      1      5         5          4         36     2     1         1     3     4     3
4    58      1      5         5          1          3     2     2         2     5     2     6
5    50      1      5         4          1         36     2     2         2     3     3     5
6    53      1      5         5          1         48     4     2         2     5     5     6
# … with 49 more variables: Q4 <dbl>, Q5 <dbl>, Q6 <dbl>, Q7 <dbl>, Q8 <dbl>, Q9 <dbl>,
#   Q14 <dbl>, Q11 <dbl>, Q12 <dbl>, Q13 <dbl>, Q14_1 <dbl>, Q15 <dbl>, Q16 <dbl>, Q17 <dbl>,
#   Q18 <dbl>, Q19 <dbl>, Q24 <dbl>, Q21 <dbl>, Q22 <dbl>, Q23 <dbl>, Q24_1 <dbl>, Q25 <dbl>,
#   WSCTotal_Score <dbl>, Q26 <dbl>, Q27 <dbl>, Q28 <dbl>, Q29 <dbl>, Q34 <dbl>, Q31 <dbl>,
#   Q32 <dbl>, Q33 <dbl>, Q34_1 <dbl>, Q35 <dbl>, Q36 <dbl>, Q37 <dbl>, Q38 <dbl>, Q39 <dbl>,
#   Q44 <dbl>, Q41 <dbl>, Q42 <dbl>, WRITotal_Score <dbl>, BQ1 <dbl>, BQ2 <dbl>, BQ3 <dbl>,
#   BQ4 <dbl>, BQ5 <dbl>, BQ6 <dbl>, BQ7 <dbl>, GAD7Total_Score <dbl>
```

Why do we clean your data?

- Accurate - Is the data true
- Uniform - Are all units within a column and across datasets measured the same way
- Consistent - Is data consistent across variables
- Valid - Does the data conform to constraints
- Complete - Missingness will happen but there are ways to mitigate this

Other good data cleaning rules

- Only one piece of information per column
- No unclear values in cells
- Descriptive variable names with no spaces
- No characters (ex: $) in columns unless it is a string variable
- No duplicate entries

Typical data cleaning steps

1. Read in file/s and explore data
2. Drop columns (Ex: De-identify data - remove name)
3. Rename columns (Ex: Descriptive names, remove spaces)
4. Filter data (Ex: remove those with missing IDs)
5. Remove duplicates
6. Transform/create cols (Ex: string->numeric, or remove $ or %)
7. Recode variables (Ex: NA->0, reverse code likert scale)
8. Transform data (Ex: from wide to long or long to wide)
9. Merge data and/or append data
10. Add variable labels
11. Make codebook
12. Export data

Associated packages and functions

Step	Package::Function
Read file	readxl::read_excel; readr::read_csv
Read files from folder	list.files, lapply (base)
Explore data	dplyr::glimpse; names, str, summary, table (base)
Explore cont.	skimr::skim; janitor::tabyl
Select cols	dplyr:: select; starts_with, contains, ends_with
Rename cols	purr::set_names; setNames (base R); dplyr::rename dplyr::filter

Packages

Packages add functionality that is not present in base R. They're where much of the power of R is found.

R: A new phone	R Packages: Apps you can download
	

Courtesy Modern Dive

Common data cleaning (cont.)

Step	Package::Function
Remove duplicate rows	dplyr::distinct
Create/Transform cols	dplyr::mutate, stringr::str_remove, str_extract; tidyr::extract
Split column	tidyr::separate; stringr::str_split
Concatenate 2 cols	paste0 (base); tidyverse::glue
Change col class	lubridate::mdy; as.numeric, as.string, as.factor (base)
Recode cols	dplyr::recode, na_if; tidyr::replace_na; ifelse (base)
Add value labels	labelled::labelled

Tidyverse (https://www.tidyverse.org)

Tidyverse

Packages Blog Learn Help Contribute

R packages for data science

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

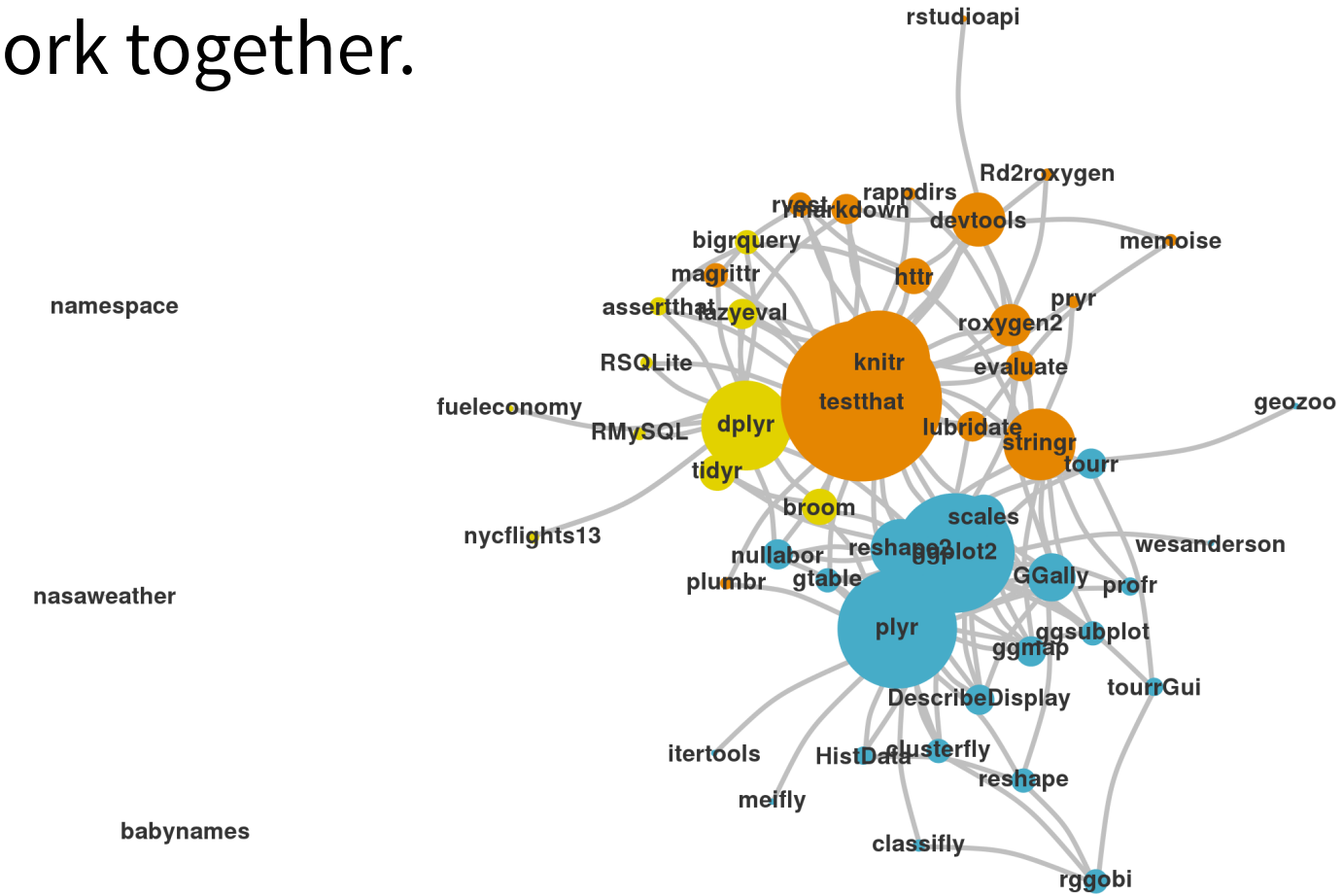
Install the complete tidyverse with:

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

Learn the tidyverse

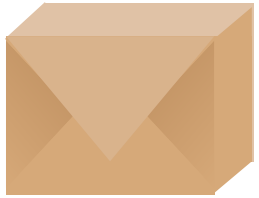
The Tidyverse

A collection of modern R packages that share common philosophies, embed best practices, and are designed to work together.



Display by Adolfo Álvarez

tidyverse

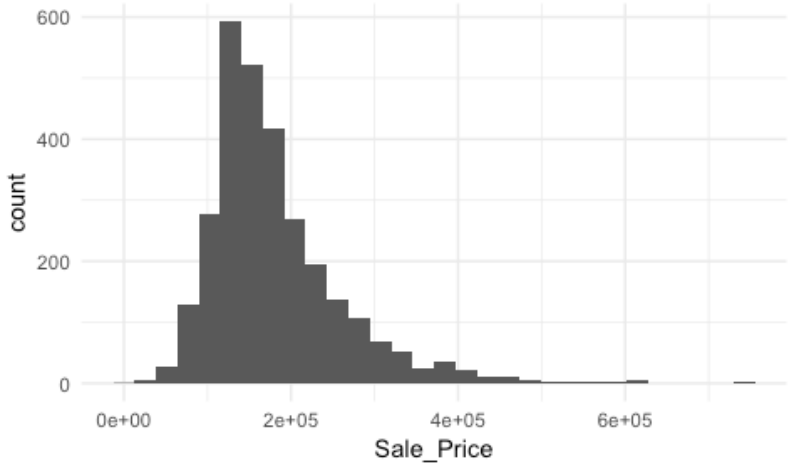


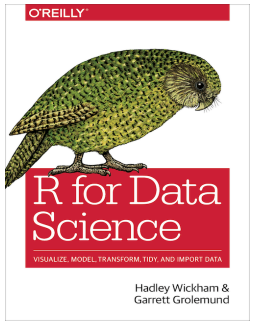
An R package that serves as a short cut for installing and loading the components of the tidyverse.

```
library("tidyverse")
```

Data Exploration

Umar Ahmad





This webinar is will be based on
R for Data Science http://r4ds.had.co.nz/

Examine the Data

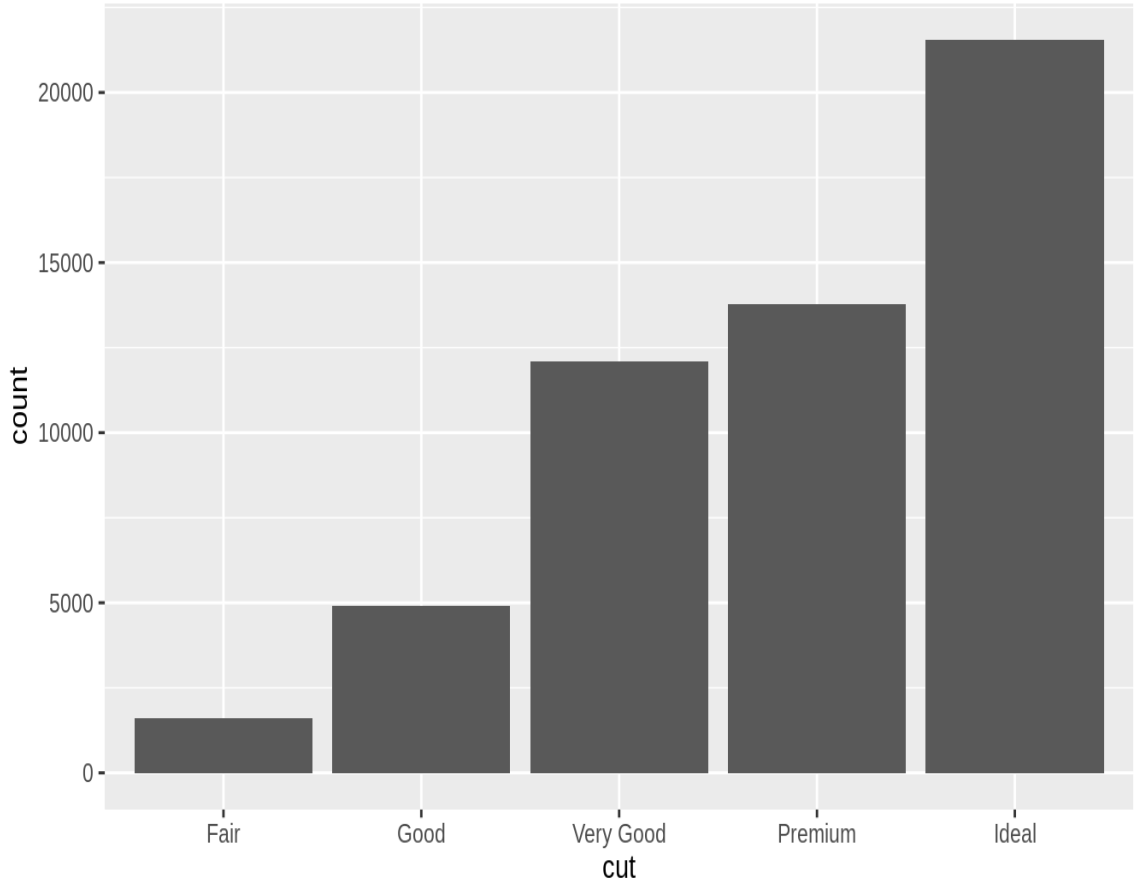
If you type the name of your data frame (i.e. cancer), R will output the following:

```
# A tibble: 6 x 61
    Age Gender status Education `job role` `How long` Stage  type treatment    Q1    Q2    Q3
  <dbl>  <dbl>  <dbl>     <dbl>      <dbl>      <dbl> <dbl> <dbl>     <dbl> <dbl> <dbl> <dbl>
1    45      1      5         4          2         24     2     1         2     3     1     5
2    48      1      5         5          4         21     2     1         2     1     2     2
3    41      1      5         5          4         36     2     1         1     3     4     3
4    58      1      5         5          1          3     2     2         2     5     2     6
5    50      1      5         4          1         36     2     2         2     3     3     5
6    53      1      5         5          1         48     4     2         2     5     5     6
# … with 49 more variables: Q4 <dbl>, Q5 <dbl>, Q6 <dbl>, Q7 <dbl>, Q8 <dbl>, Q9 <dbl>,
#   Q14 <dbl>, Q11 <dbl>, Q12 <dbl>, Q13 <dbl>, Q14_1 <dbl>, Q15 <dbl>, Q16 <dbl>, Q17 <dbl>,
#   Q18 <dbl>, Q19 <dbl>, Q24 <dbl>, Q21 <dbl>, Q22 <dbl>, Q23 <dbl>, Q24_1 <dbl>, Q25 <dbl>,
#   WSCTotal_Score <dbl>, Q26 <dbl>, Q27 <dbl>, Q28 <dbl>, Q29 <dbl>, Q34 <dbl>, Q31 <dbl>,
#   Q32 <dbl>, Q33 <dbl>, Q34_1 <dbl>, Q35 <dbl>, Q36 <dbl>, Q37 <dbl>, Q38 <dbl>, Q39 <dbl>,
#   Q44 <dbl>, Q41 <dbl>, Q42 <dbl>, WRITotal_Score <dbl>, BQ1 <dbl>, BQ2 <dbl>, BQ3 <dbl>,
#   BQ4 <dbl>, BQ5 <dbl>, BQ6 <dbl>, BQ7 <dbl>, GAD7Total_Score <dbl>
```

```
+]
> head(cancer)
# A tibble: 6 x 61
    Age Gender status Education `job role` `How long` Stage  type treatment    Q1    Q2    Q3
  <dbl>  <dbl>  <dbl>     <dbl>      <dbl>      <dbl> <dbl> <dbl>     <dbl> <dbl> <dbl> <dbl>
1    45      1      5         4          2         24     2     1         2     3     1     5
2    48      1      5         5          4         21     2     1         2     1     2     2
3    41      1      5         5          4         36     2     1         1     3     4     3
4    58      1      5         5          1          3     2     2         2     5     2     6
5    50      1      5         4          1         36     2     2         2     3     3     5
6    53      1      5         5          1         48     4     2         2     5     5     6
# … with 49 more variables: Q4 <dbl>, Q5 <dbl>, Q6 <dbl>, Q7 <dbl>, Q8 <dbl>, Q9 <dbl>,
#   Q14 <dbl>, Q11 <dbl>, Q12 <dbl>, Q13 <dbl>, Q14_1 <dbl>, Q15 <dbl>, Q16 <dbl>, Q17 <dbl>,
#   Q18 <dbl>, Q19 <dbl>, Q24 <dbl>, Q21 <dbl>, Q22 <dbl>, Q23 <dbl>, Q24_1 <dbl>, Q25 <dbl>,
#   WSCTotal_Score <dbl>, Q26 <dbl>, Q27 <dbl>, Q28 <dbl>, Q29 <dbl>, Q34 <dbl>, Q31 <dbl>,
#   Q32 <dbl>, Q33 <dbl>, Q34_1 <dbl>, Q35 <dbl>, Q36 <dbl>, Q37 <dbl>, Q38 <dbl>, Q39 <dbl>,
#   Q44 <dbl>, Q41 <dbl>, Q42 <dbl>, WRITotal_Score <dbl>, BQ1 <dbl>, BQ2 <dbl>, BQ3 <dbl>,
#   BQ4 <dbl>, BQ5 <dbl>, BQ6 <dbl>, BQ7 <dbl>, GAD7Total_Score <dbl>
> summary(cancer[4])
   Education
 Min.   :0.0
 1st Qu.:4.0
 Median :5.0
 Mean   :4.4
 3rd Qu.:5.0
 Max.   :6.0
>
```

Visualising distribution…

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```



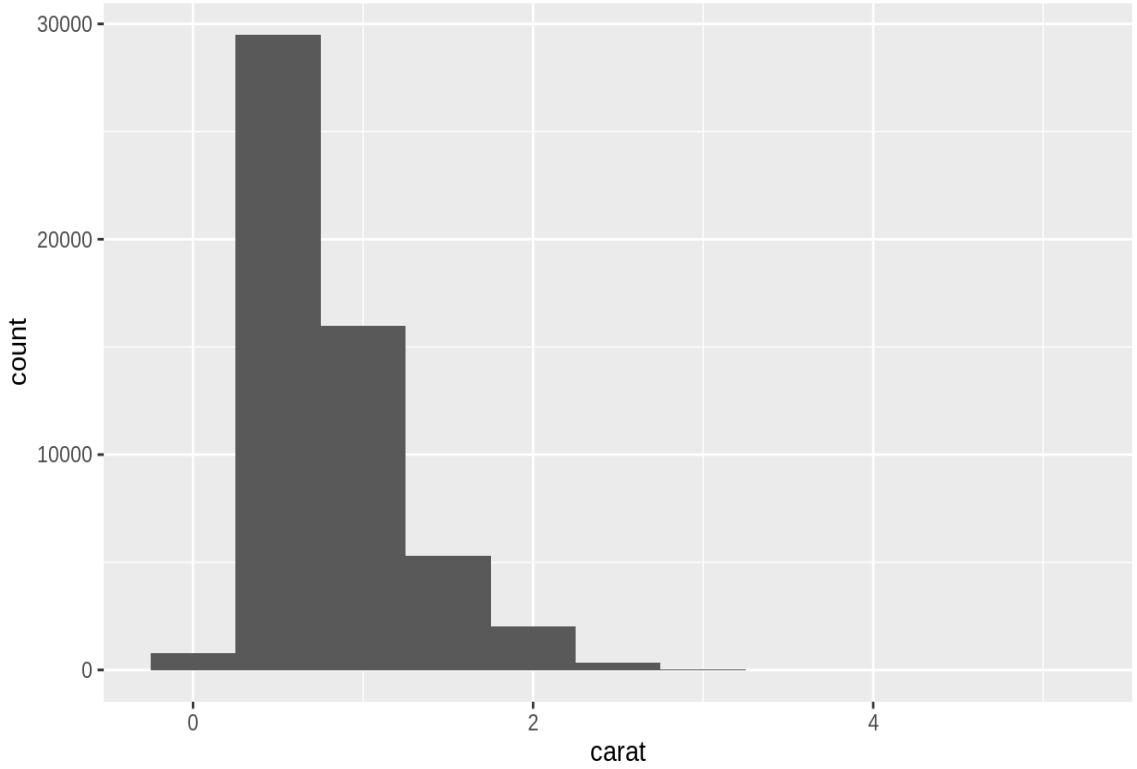
- The height of the bars displays how many observations with each x value

- You can compute these values manually with dplyr::count():

```
diamonds %>%
  count(cut)
#> # A tibble: 5 x 2
#>   cut         n
#>   <ord>     <int>
#> 1 Fair       1610
#> 2 Good       4906
#> 3 Very Good 12082
#> 4 Premium   13791
#> 5 Ideal     21551
```

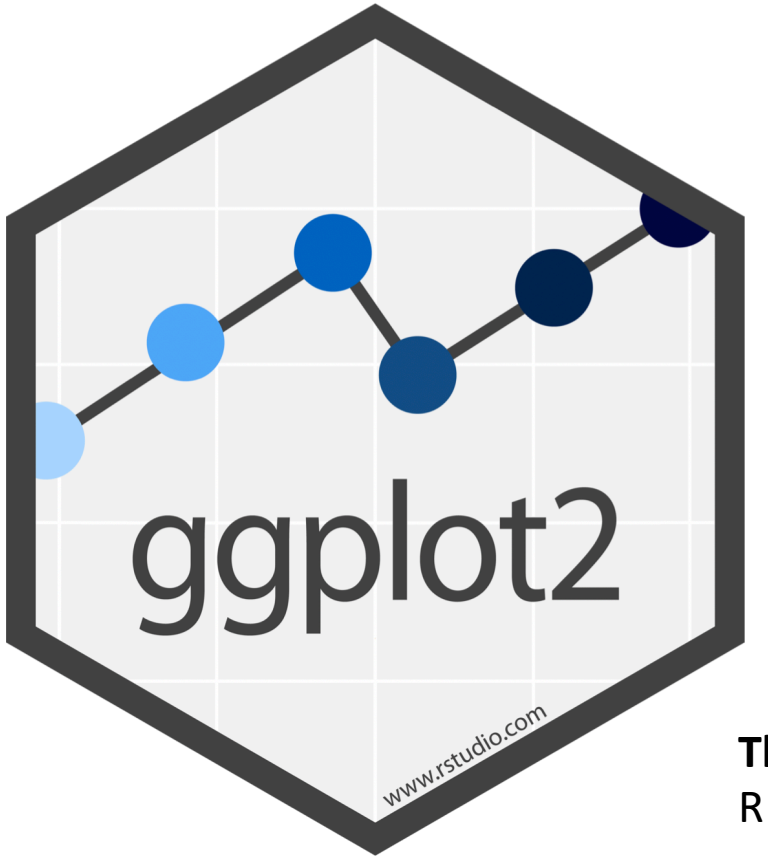
Visualising distribution…

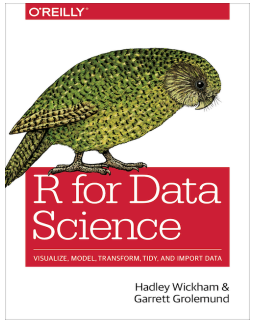
```
ggplot(data = diamonds) +
    geom_histogram(mapping = aes(x = carat),
    binwidth = 0.5)
```




```
diamonds %>%
  count(cut_width(carat, 0.5))
#> # A tibble: 11 x 2
#>   `cut_width(carat, 0.5)`     n
#>   <fct>                   <int>
#> 1 [-0.25,0.25]              785
#> 2 (0.25,0.75]             29498
#> 3 (0.75,1.25]             15977
#> 4 (1.25,1.75]              5313
#> 5 (1.75,2.25]              2002
#> 6 (2.25,2.75]               322
#> # … with 5 more rows
```

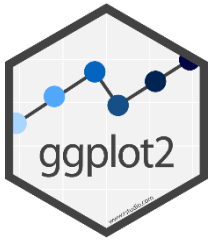
Data visualization with





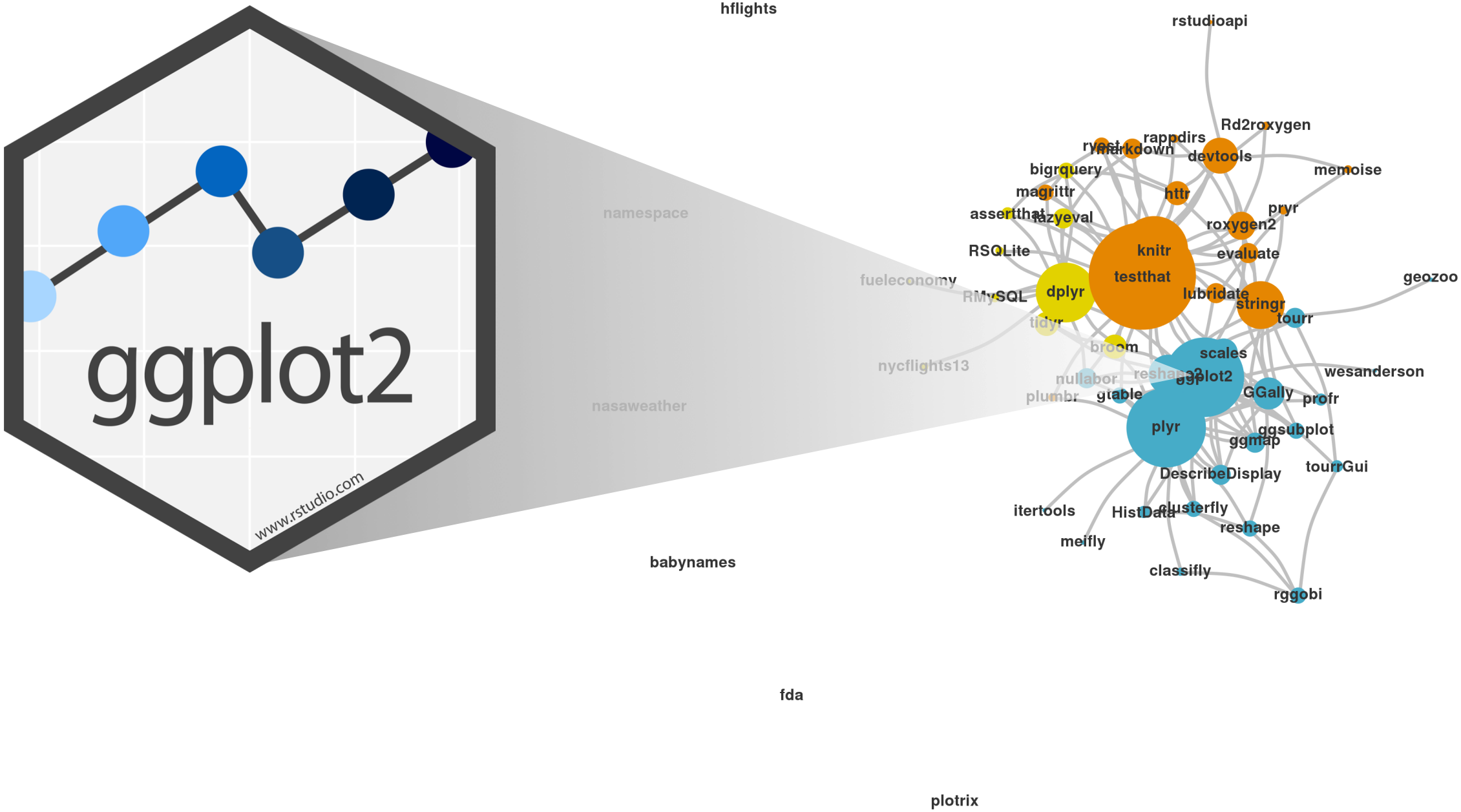
This webinar is will be based on
R for Data Science http://r4ds.had.co.nz/

Plotting figures and graphs with ggplot



- ggplot is the plotting library for tidyverse
  - Powerful
  - Flexible

- Follows the same conventions as the rest of tidyverse
  - Data stored in tibbles
  - Data is arranged in 'tidy' format
  - Tibble is the first argument to each function

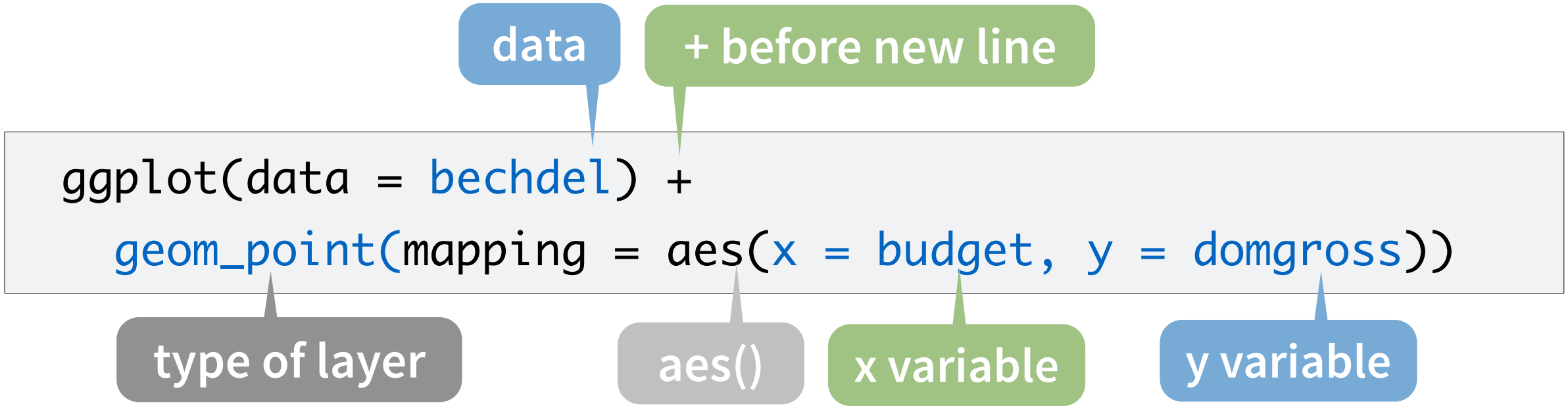


Display by Adolfo Álvarez

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Courtesy: Hadley Wickham



Courtesy: Hadley Wickham

Geometries and Aesthetics

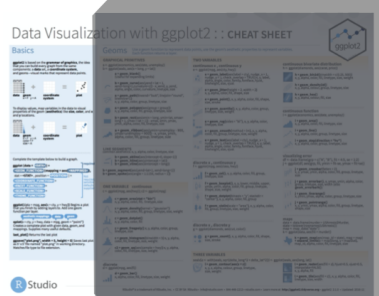
- Geometries are types of plot

`geom_point()`	Point geometry, (x/y plots, stripcharts etc)
`geom_line()`	Line graphs
`geom_boxplot()`	Box plots
`geom_col()`	Barplots
`geom_histogram()`	Histogram plots

- Aesthetics are graphical parameters which can be adjusted in a given geometry

geom_ functions

Each requires a mapping argument.



Geoms

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

ggplot2

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre= 1)
x, y, alpha, color, group, linetype, size

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

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

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

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

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

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

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

discrete x , discrete y

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

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

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

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

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

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

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

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

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

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

k + 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))l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

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

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

ggplot2

Aesthetics

aesthetic property

Variable to map it to

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, color = clean_test))
```

```
ggplot(bechdel) +
    geom_point(mapping = aes(x = budget, y = domgross, size = clean_test))
```

Courtesy: Hadley Wickham

Code structure of a ggplot graph

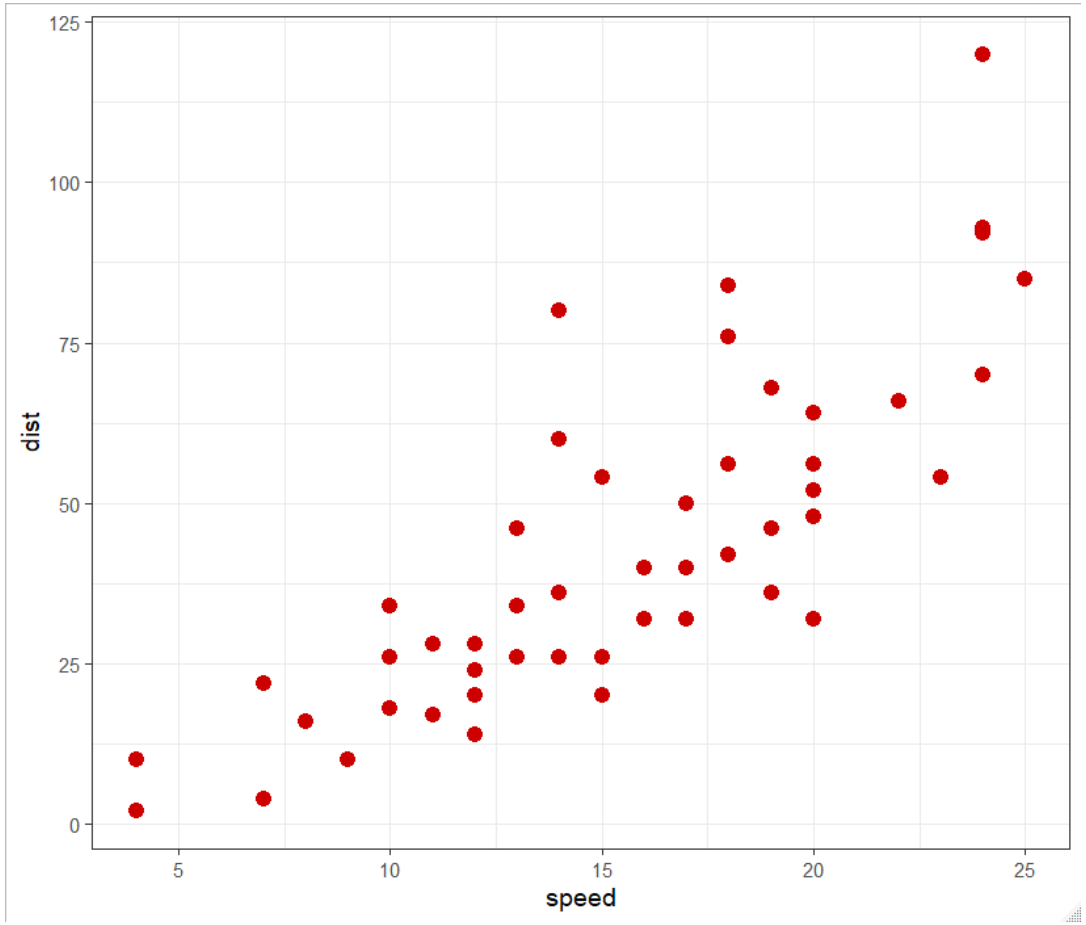
- Start with a call to `ggplot()`
  - Pass the tibble of data
  - Say which columns you want to use
  - Generates a value which you can store or print

- Say which graphical representation you want to use
  - Points, lines, barplots etc
  - "Add" results to the value from ggplot

- Customise labels, colours annotations etc.

- Print the value – draws the plot

Aesthetics for `geom_point()`

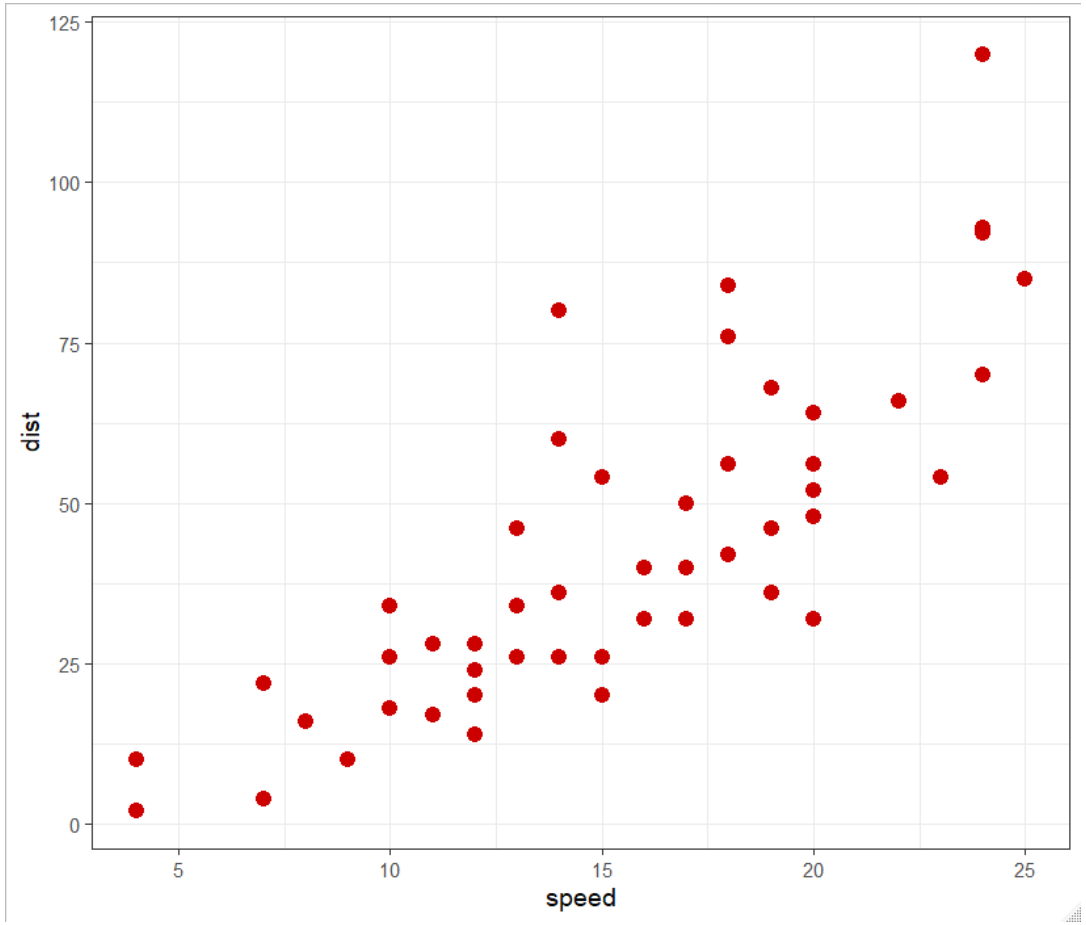


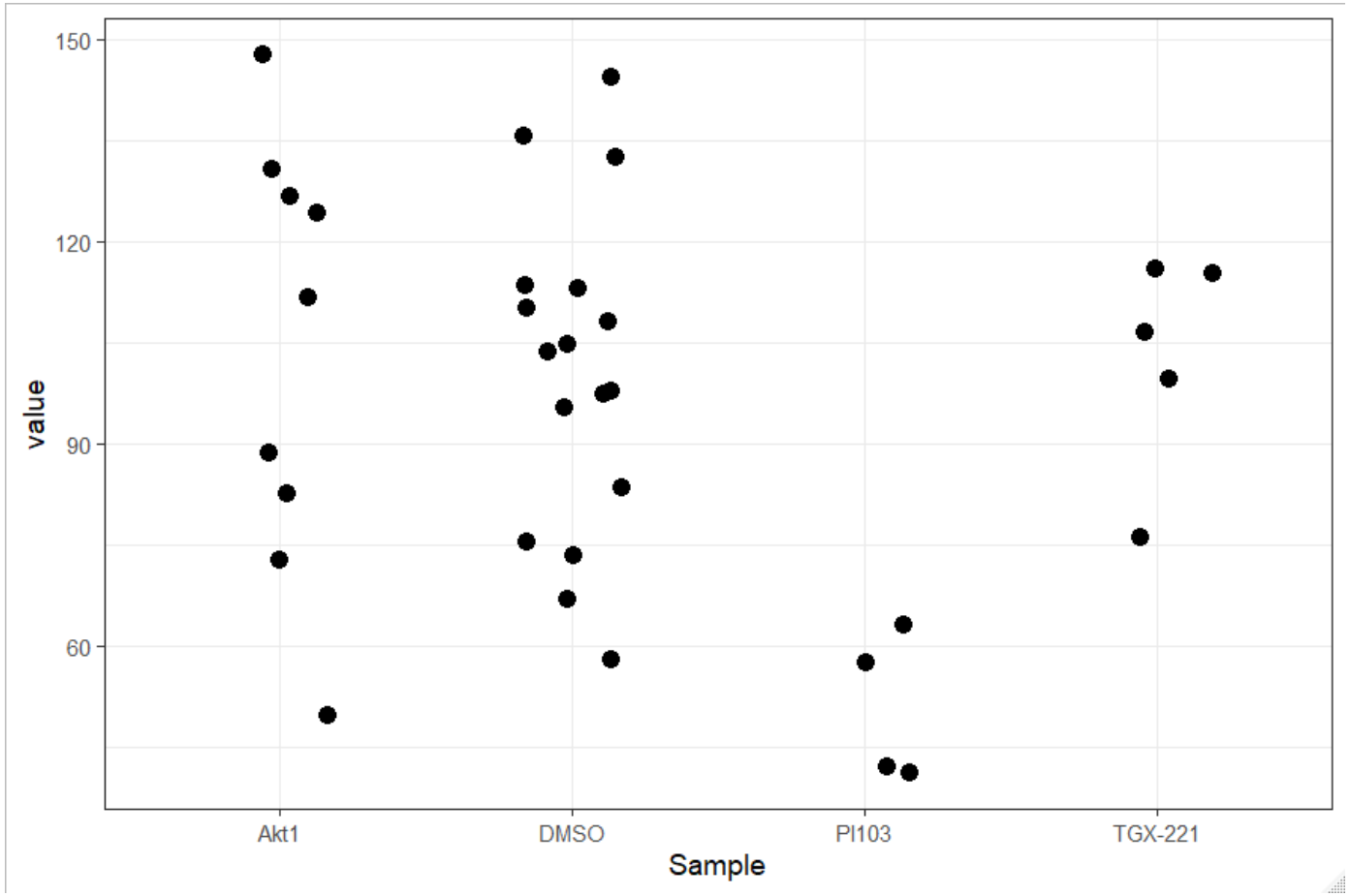
Aesthetics

`geom_point()` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `shape`
- `size`
- `stroke`

Mappings can be quantitative or categorical

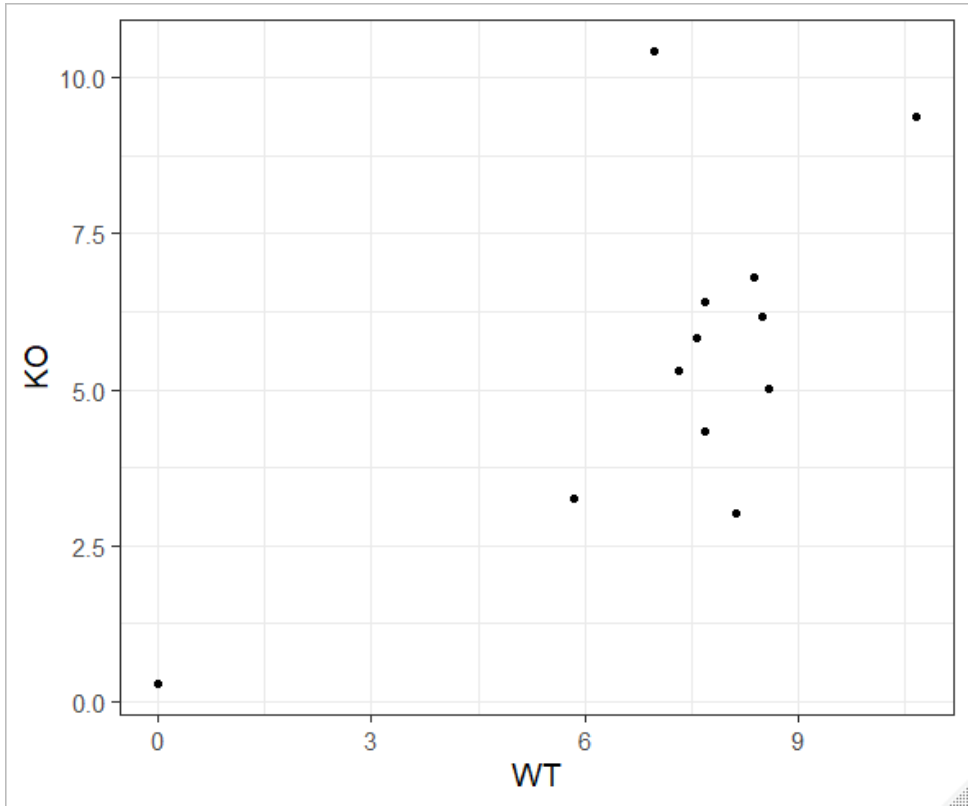




Our first plot…

```
ggplot(expression, aes(x=WT, y=KO)) + geom_point()
```

```
> expression
# A tibble: 12 x 4
   Gene        WT     KO pValue
   <chr>    <dbl>  <dbl>  <dbl>
 1 Mia1      5.83  3.24   0.1
 2 Snrpa     8.59  5.02   0.001
 3 Itpkc     8.49  6.16   0.04
 4 Adck4     7.69  6.41   0.2
 5 Numbl     8.37  6.81   0.1
 6 Ltbp4     6.96 10.4    0.001
 7 Shkbp1    7.57  5.83   0.1
 8 Spnb4    10.7   9.38   0.2
 9 Blvrb     7.32  5.29   0.05
10 Pgam1     0     0.285  0.5
11 Sertad3   8.13  3.02   0.0001
12 Sertad1   7.69  4.34   0.01
```



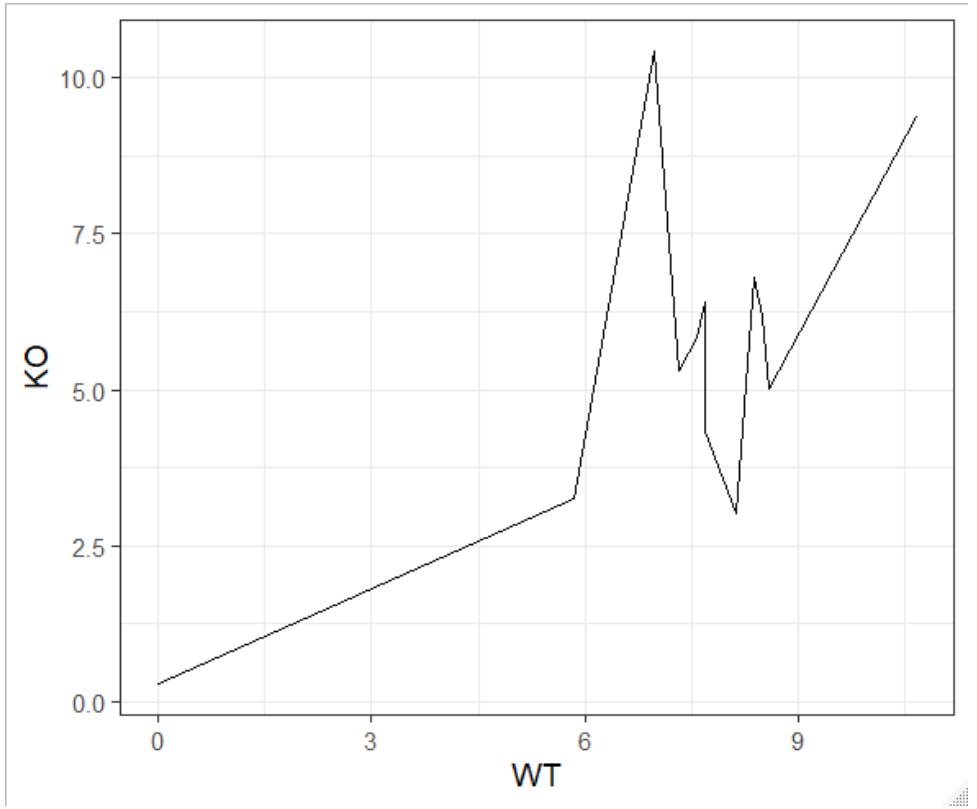
- Identify the tibble with the data you want to plot
- Decide on the geometry (plot type) you want to use
- Decide which columns will modify which aesthetic

- Call `ggplot(aes(…..))`
- Add a `geom_xxx` function call

Our second plot…

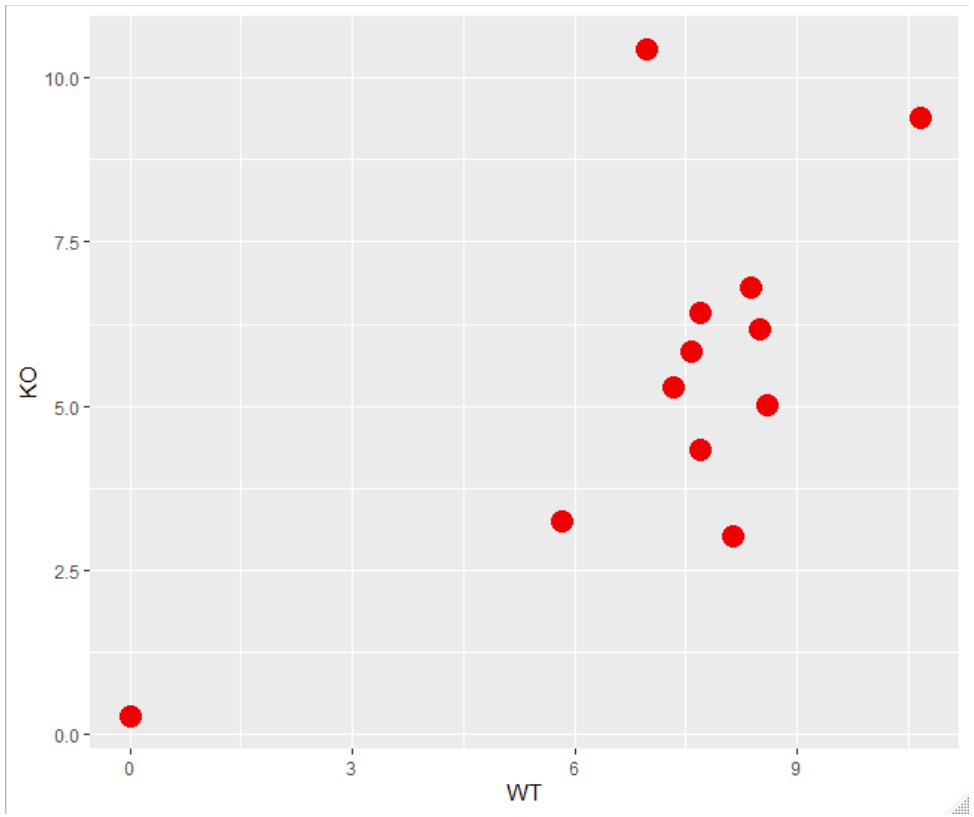
```
ggplot(expression, aes(x=WT, y=KO)) + geom_line()
```

```
> expression
# A tibble: 12 x 4
   Gene        WT     KO pValue
   <chr>    <dbl>  <dbl>  <dbl>
 1 Mia1      5.83  3.24   0.1
 2 Snrpa     8.59  5.02   0.001
 3 Itpkc     8.49  6.16   0.04
 4 Adck4     7.69  6.41   0.2
 5 Numbl     8.37  6.81   0.1
 6 Ltbp4     6.96 10.4    0.001
 7 Shkbp1    7.57  5.83   0.1
 8 Spnb4    10.7   9.38   0.2
 9 Blvrb     7.32  5.29   0.05
10 Pgam1     0     0.285  0.5
11 Sertad3   8.13  3.02   0.0001
12 Sertad1   7.69  4.34   0.01
```



Our third plot…

```
expression %>%
  ggplot (aes(x=WT, y=KO)) +
  geom_point(colour="red2", size=5)
```



Other data plot types (geometries)

- Barplots
  - geom_bar
  - geom_col
- Stripcharts
  - geom_jitter
- Distribution Summaries
  - geom_histogram
  - geom_density
  - geom_violin
  - geom_boxplot

Drawing a barplot (`geom_col()` or `geom_bar()`)

- Two different functions – depends on the nature of the data
- If your data has values which represents the height of the bars use `geom_col`
- If your data has individual values and you want the plot to either count them or calculate a summary (usually the mean) then use `geom_bar`

Drawing a bar height barplot (`geom_col()`)

Aesthetics

`geom_bar()` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- linetype
- size

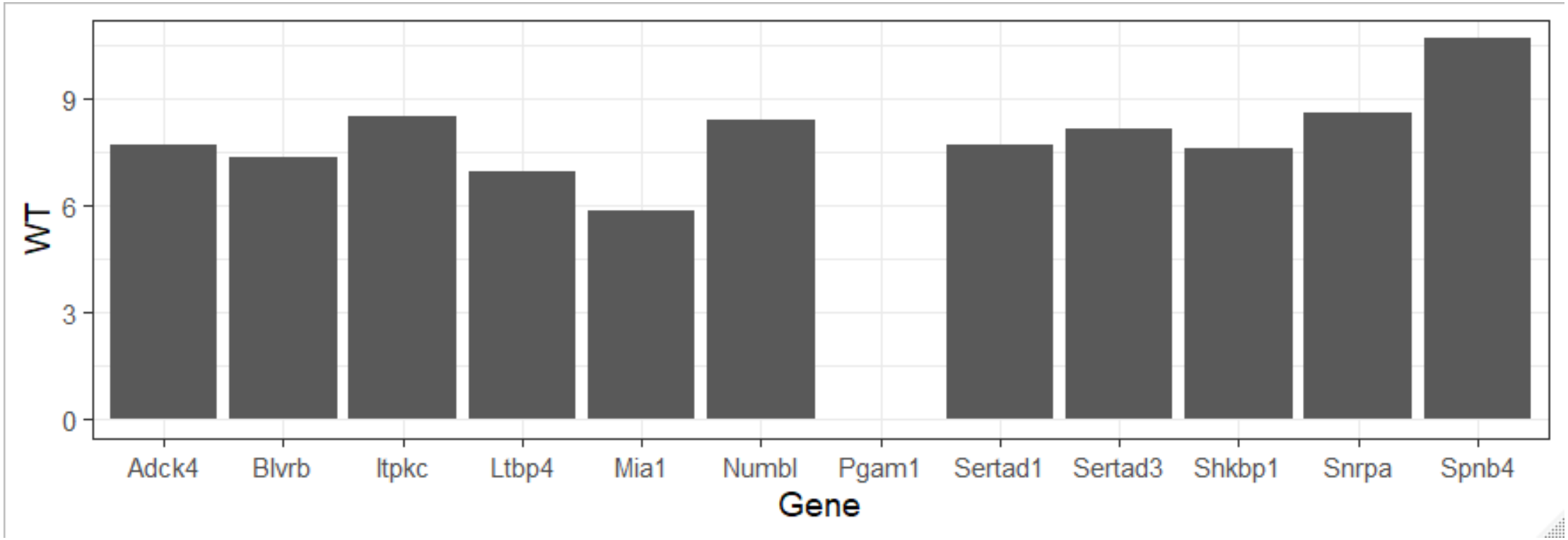
- Plot the expression values for the WT samples for all genes

- What is your X?
- What is your Y?

```
> expression
# A tibble: 12 x 4
   Gene        WT      KO pValue
   <chr>    <dbl>   <dbl>  <dbl>
 1 Mia1      5.83   3.24   0.1
 2 Snrpa     8.59   5.02   0.001
```

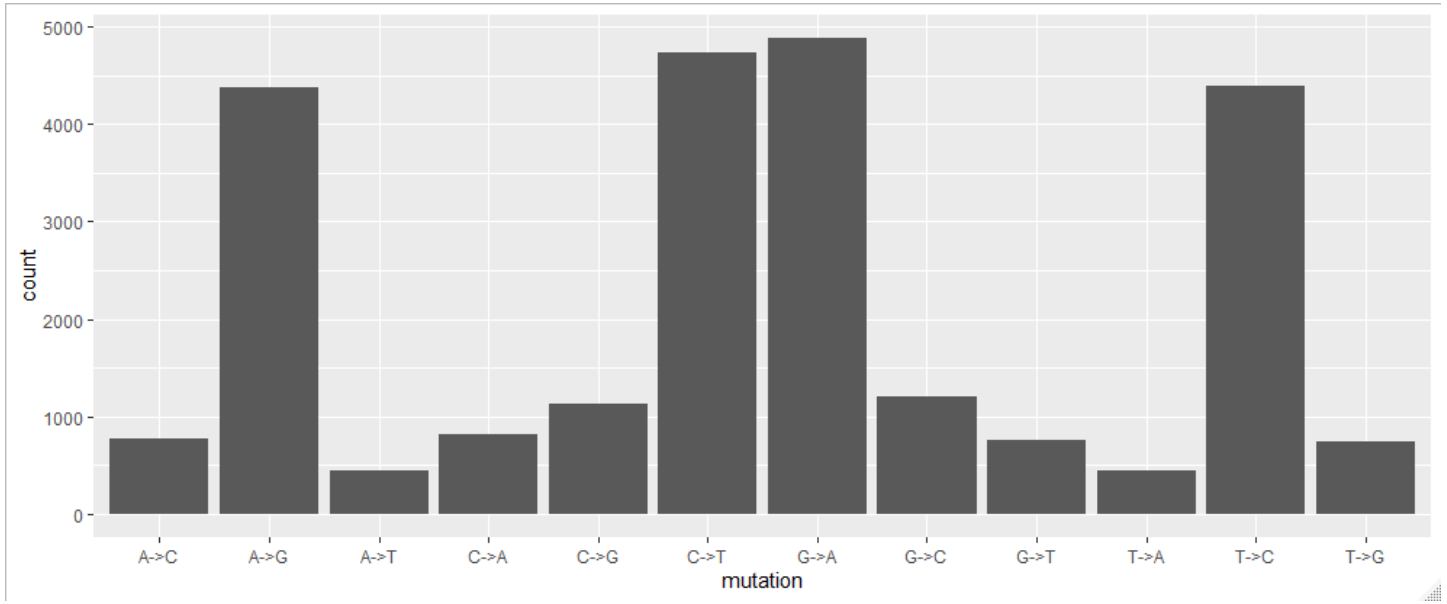
A bar height barplot

```
ggplot(expression, aes(x=Gene, y=WT)) +
geom_col()
```



A summarised barplot (`geom_bar`) - counts

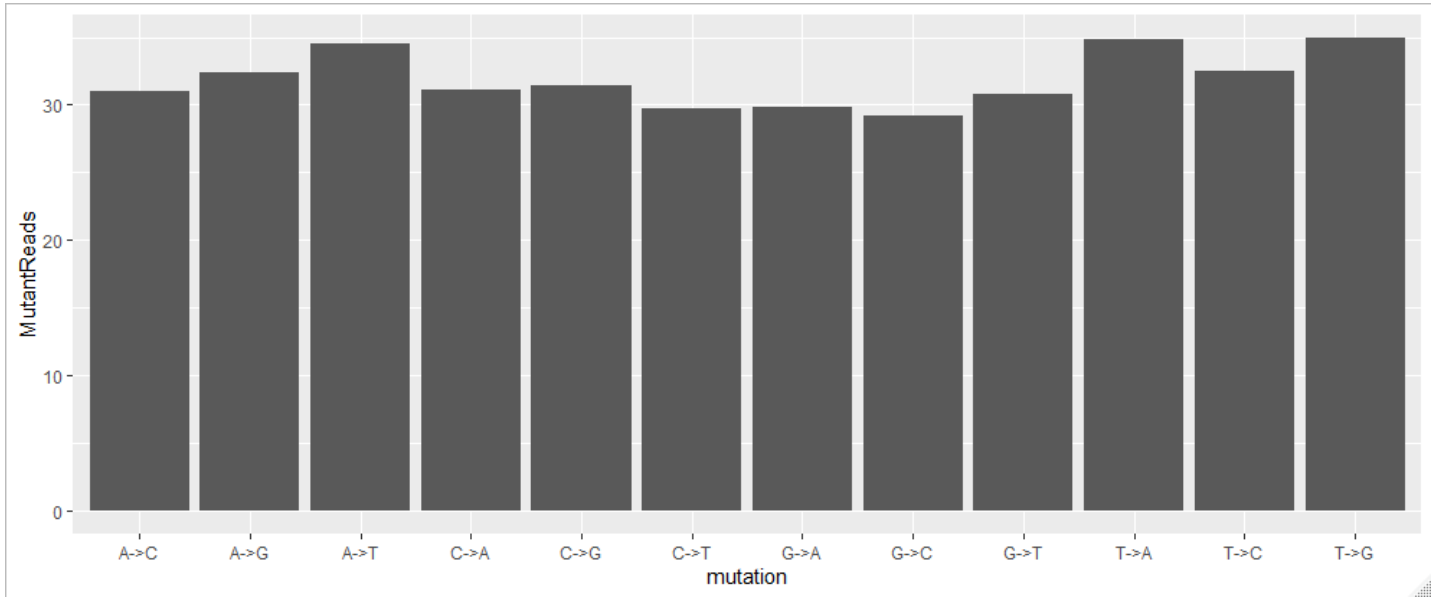
```
mutation.plotting.data %>%
  ggplot(aes(x=mutation)) +
    geom_bar()
```




```
> mutation.plotting.data
# A tibble: 24,686 x 9
   CHR      POS dbSNP      mutation  QUAL GENE   ENST            MutantReads COVERAGE
   <chr>  <dbl> <chr>      <chr>    <dbl> <chr>  <chr>                 <dbl>    <dbl>
 1 1      69270 .          A->G        16 OR4F5  ENST00000335137           3        4
 2 1      69511 rs75062661 A->G       200 OR4F5  ENST00000335137          24       27
 3 1      69761 .          A->T       200 OR4F5  ENST00000335137           8        8
 4 1      69897 rs75758884 T->C        59 OR4F5  ENST00000335137           3        3
 5 1     877831 rs6672356  T->C       200 SAMD11 ENST00000342066          10       11
 6 1     881627 rs2272757  G->A       200 NOC2L  ENST00000327044          52       56
 7 1     887801 rs3828047  A->G       200 NOC2L  ENST00000327044          47       48
 8 1     888639 rs3748596  T->C       200 NOC2L  ENST00000327044          23       24
 9 1     888659 rs3748597  T->C       200 NOC2L  ENST00000327044          17       21
10 1     889158 rs13303056 G->C       200 NOC2L  ENST00000327044          25       28
```

A summarised barplot (`geom_bar`) - means

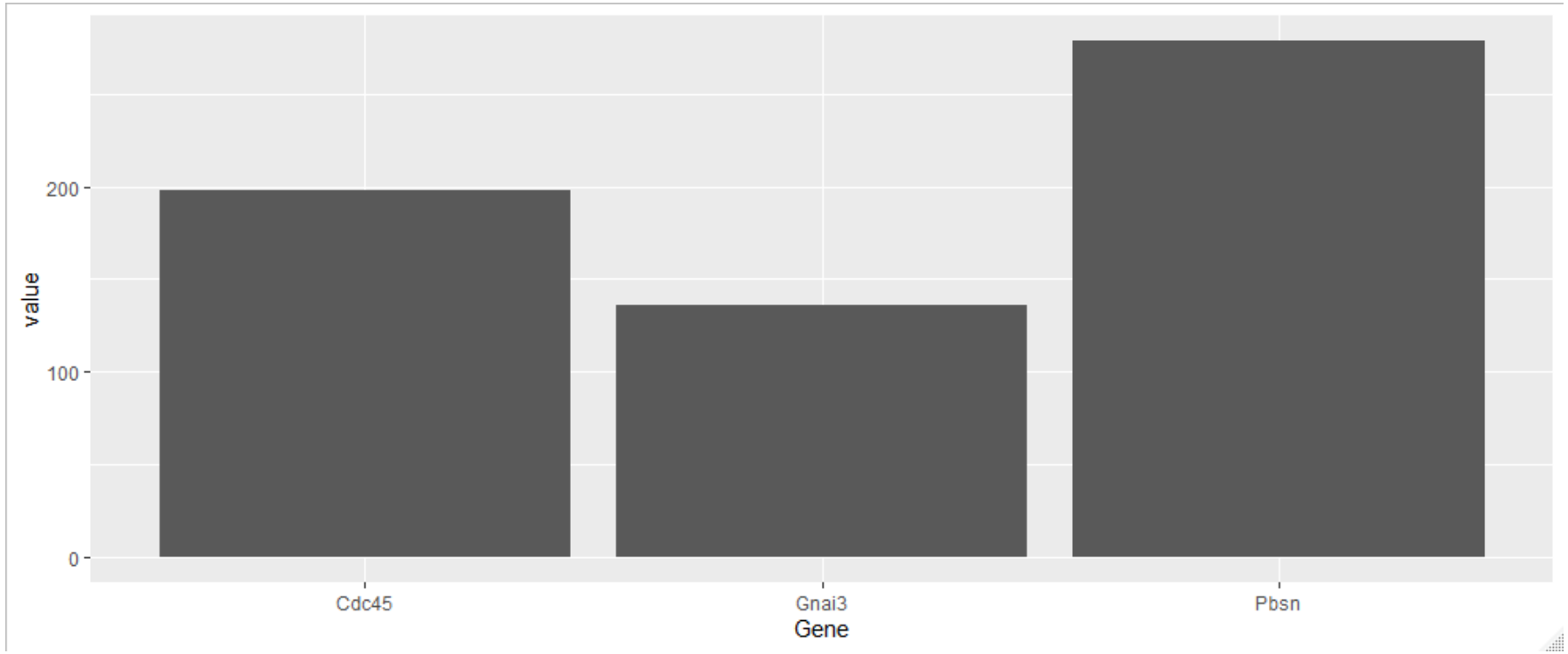
```
mutation.plotting.data %>%
  ggplot(aes(x=mutation, y=MutantReads))+
  geom_bar(stat="summary", fun="mean")
```




```
> mutation.plotting.data
# A tibble: 24,686 x 9
   CHR      POS dbSNP      mutation  QUAL GENE   ENST            MutantReads COVERAGE
   <chr>  <dbl> <chr>      <chr>    <dbl> <chr>  <chr>                 <dbl>    <dbl>
 1 1      69270 .          A->G        16 OR4F5  ENST00000335137           3        4
 2 1      69511 rs75062661 A->G       200 OR4F5  ENST00000335137          24       27
 3 1      69761 .          A->T       200 OR4F5  ENST00000335137           8        8
 4 1      69897 rs75758884 T->C        59 OR4F5  ENST00000335137           3        3
 5 1     877831 rs6672356  T->C       200 SAMD11 ENST00000342066          10       11
 6 1     881627 rs2272757  G->A       200 NOC2L  ENST00000327044          52       56
 7 1     887801 rs3828047  A->G       200 NOC2L  ENST00000327044          47       48
 8 1     888639 rs3748596  T->C       200 NOC2L  ENST00000327044          23       24
 9 1     888659 rs3748597  T->C       200 NOC2L  ENST00000327044          17       21
10 1     889158 rs13303056 G->C       200 NOC2L  ENST00000327044          25       28
```

Stacked and Grouped Barplots

```
bar.group %>%
  ggplot(aes(x=Gene, y=value)) +
  geom_col()
```



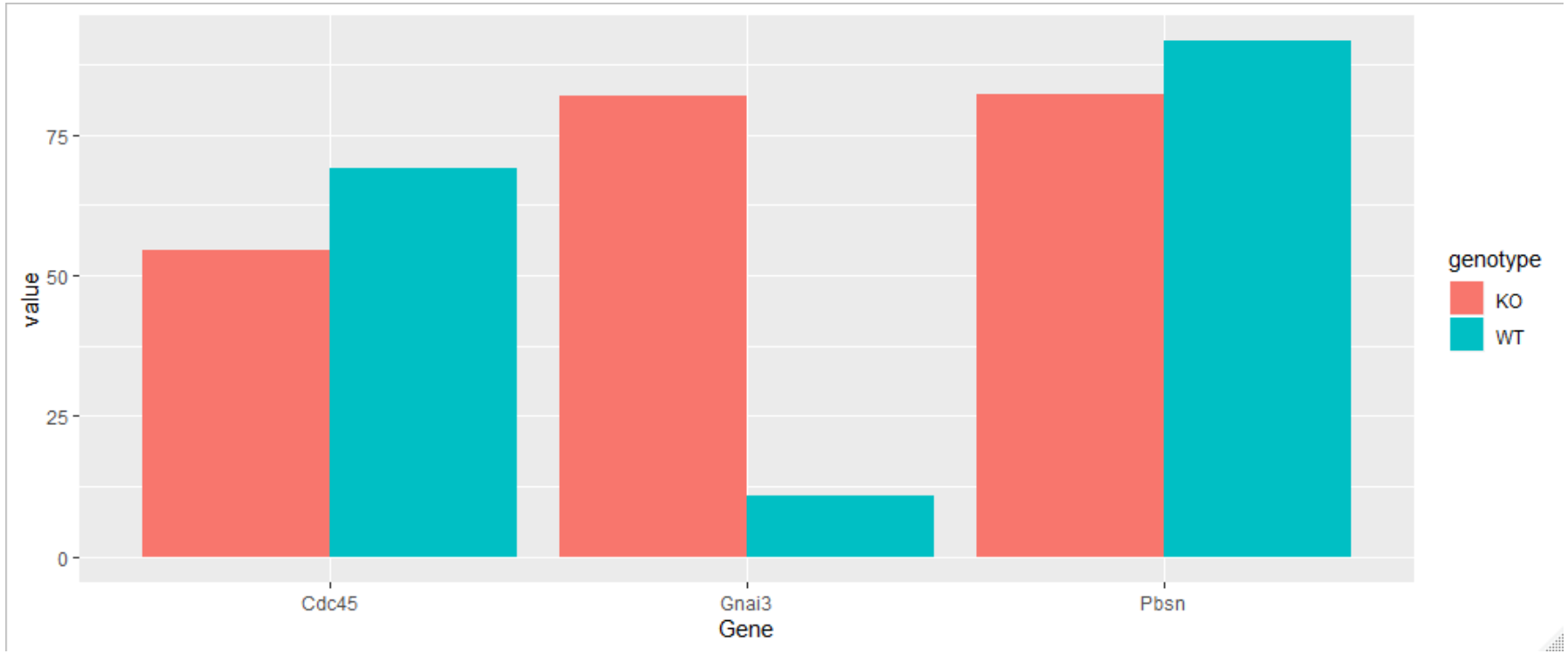

```
> bar.group
# A tibble: 12 x 3
   Gene  genotype value
   <chr> <chr>    <dbl>
 1 Gnai3 WT        9.39
 2 Pbsn  WT       91.7
 3 Cdc45 WT       69.2
 4 Gnai3 WT       10.9
 5 Pbsn  WT       59.6
 6 Cdc45 WT       36.1
 7 Gnai3 KO       33.5
 8 Pbsn  KO       45.3
 9 Cdc45 KO       54.4
10 Gnai3 KO       81.9
11 Pbsn  KO       82.3
12 Cdc45 KO       38.1
```

Sum of values

Stacked and Grouped Barplots

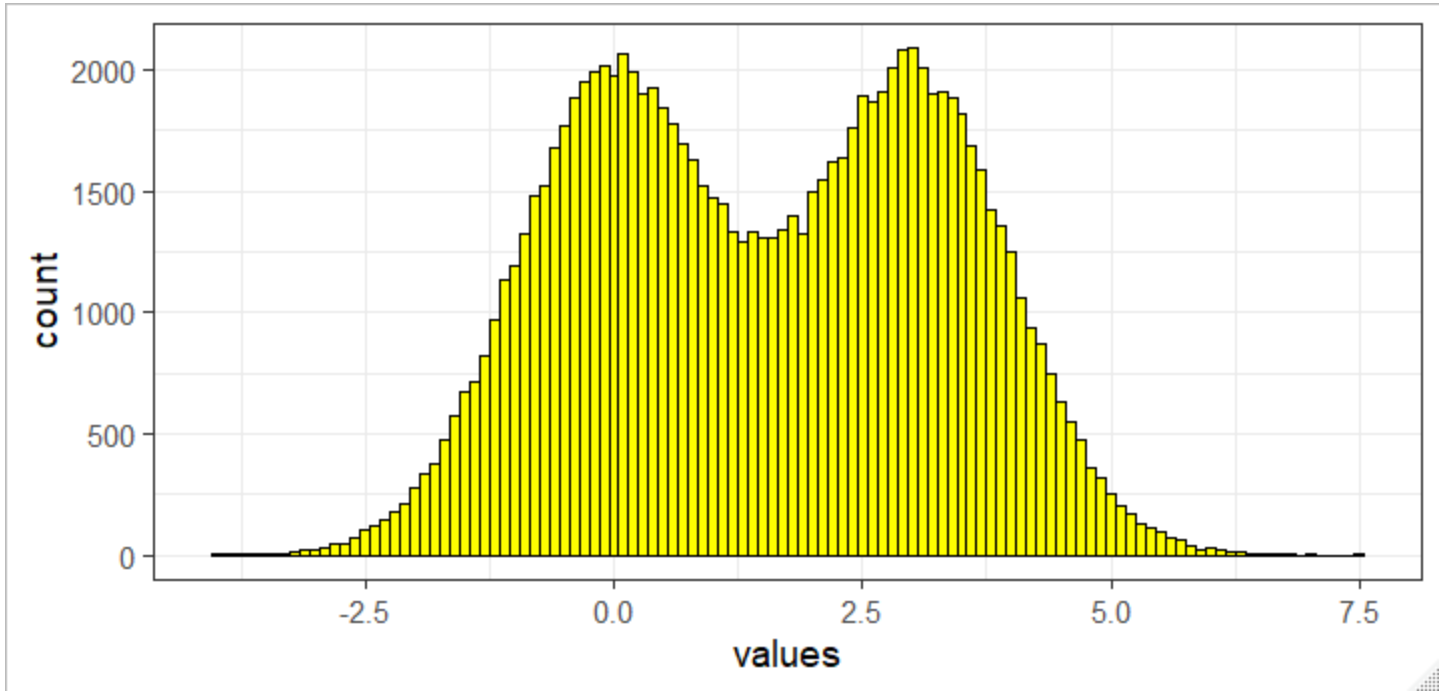
```
> bar.group
# A tibble: 12 x 3
   Gene  genotype value
   <chr> <chr>    <dbl>
 1 Gnai3 WT        9.39
 2 Pbsn  WT       91.7
 3 Cdc45 WT       69.2
 4 Gnai3 WT       10.9
 5 Pbsn  WT       59.6
 6 Cdc45 WT       36.1
 7 Gnai3 KO       33.5
 8 Pbsn  KO       45.3
 9 Cdc45 KO       54.4
10 Gnai3 KO       81.9
11 Pbsn  KO       82.3
12 Cdc45 KO       38.1
```

```
bar.group %>%
  ggplot(aes(x=Gene, y=value, fill=genotype)) +
  geom_col(position="dodge")
```



Individual values

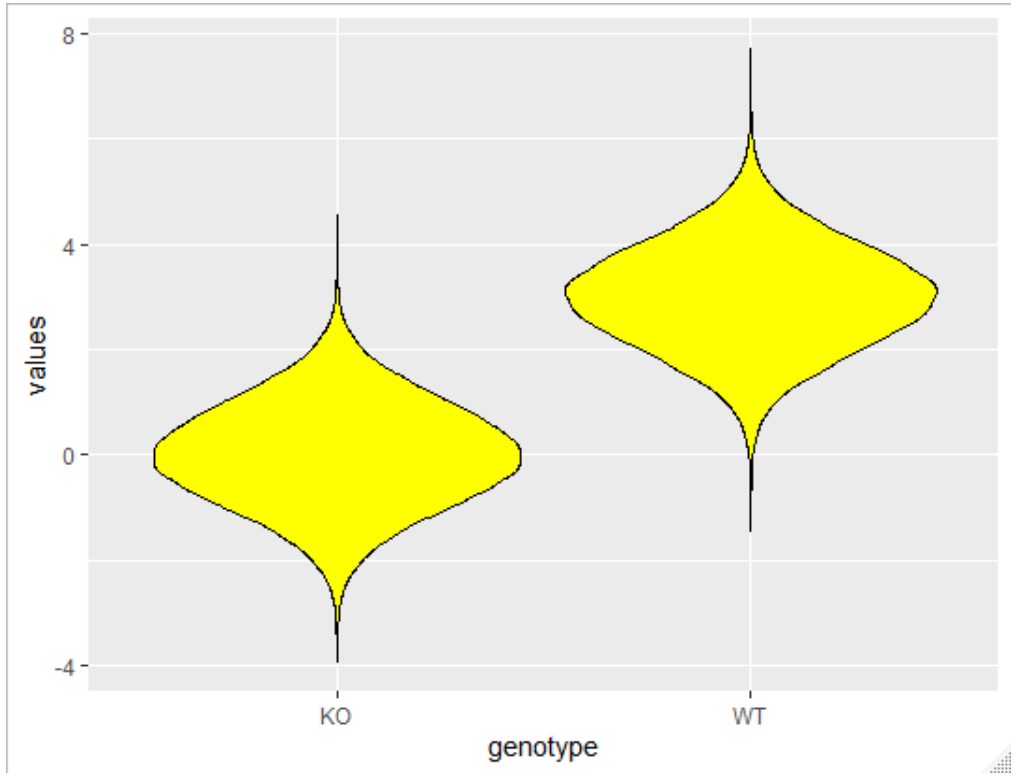
Plotting distributions - histograms




```
> many.values
# A tibble: 100,000 x 2
    values genotype
     <dbl> <chr>
 1  1.90   KO
 2  2.39   WT
 3  4.32   KO
 4  2.94   KO
 5  0.728  WT
 6 -0.280  WT
 7  0.337  WT
 8 -1.31   WT
 9  1.55   WT
10  1.86   KO
```

```
many.values %>%
  ggplot(aes(x=values)) +
  geom_histogram(binwidth = 0.1, fill="yellow", colour="black")
```

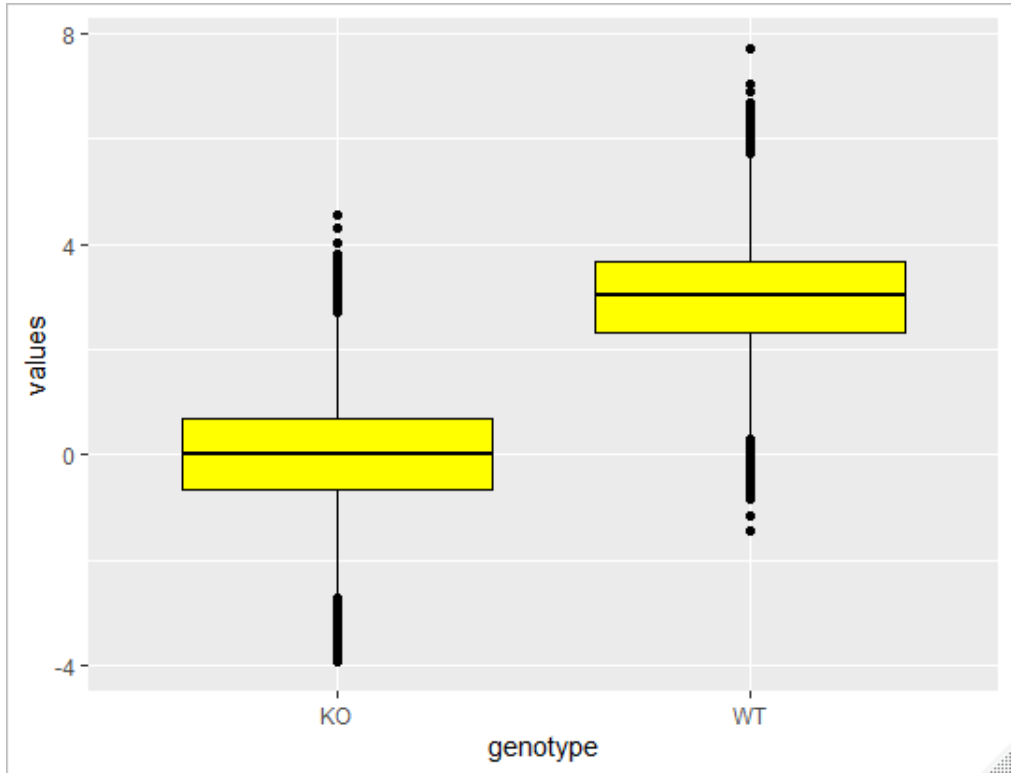
Plotting distributions – violin plots




```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90  KO
 2  2.39  WT
 3  4.32  KO
 4  2.94  KO
 5  0.728 WT
 6 -0.280 WT
 7  0.337 WT
 8 -1.31  WT
 9  1.55  WT
10  1.86  KO
```

```
many.values %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_violin(colour="black", fill="yellow")
```

Plotting distributions – boxplots




```
> many.values
# A tibble: 100,000 x 2
   values genotype
    <dbl> <chr>
 1  1.90  KO
 2  2.39  WT
 3  4.32  KO
 4  2.94  KO
 5  0.728 WT
 6 -0.280 WT
 7  0.337 WT
 8 -1.31  WT
 9  1.55  WT
10  1.86  KO
```

```
many.values %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_boxplot(colour="black", fill="yellow")
```

Annotation, Scaling and Colours

Titles and axis labels

- Can set everything with `labs()`
  - `title="Main title"`
  - `x="X axis"`
  - `y="Y axis"`

- Can use functions to set them individually
  - `ggtitle()`
  - `xlab()`
  - `ylab()`

Changing scaling

- Alter the data before plotting
  - `mutate(value=log(value))`
- Alter the data whilst plotting
  - `ggplot(aes(log(value)))`
- Alter the scale of the plot
  - Add an option to adjust the scaling of the axis

Axis scaling options

- Transforming scales
  - `scale_x_log10()`
  - `scale_x_sqrt()`
  - `scale_x_reverse()`

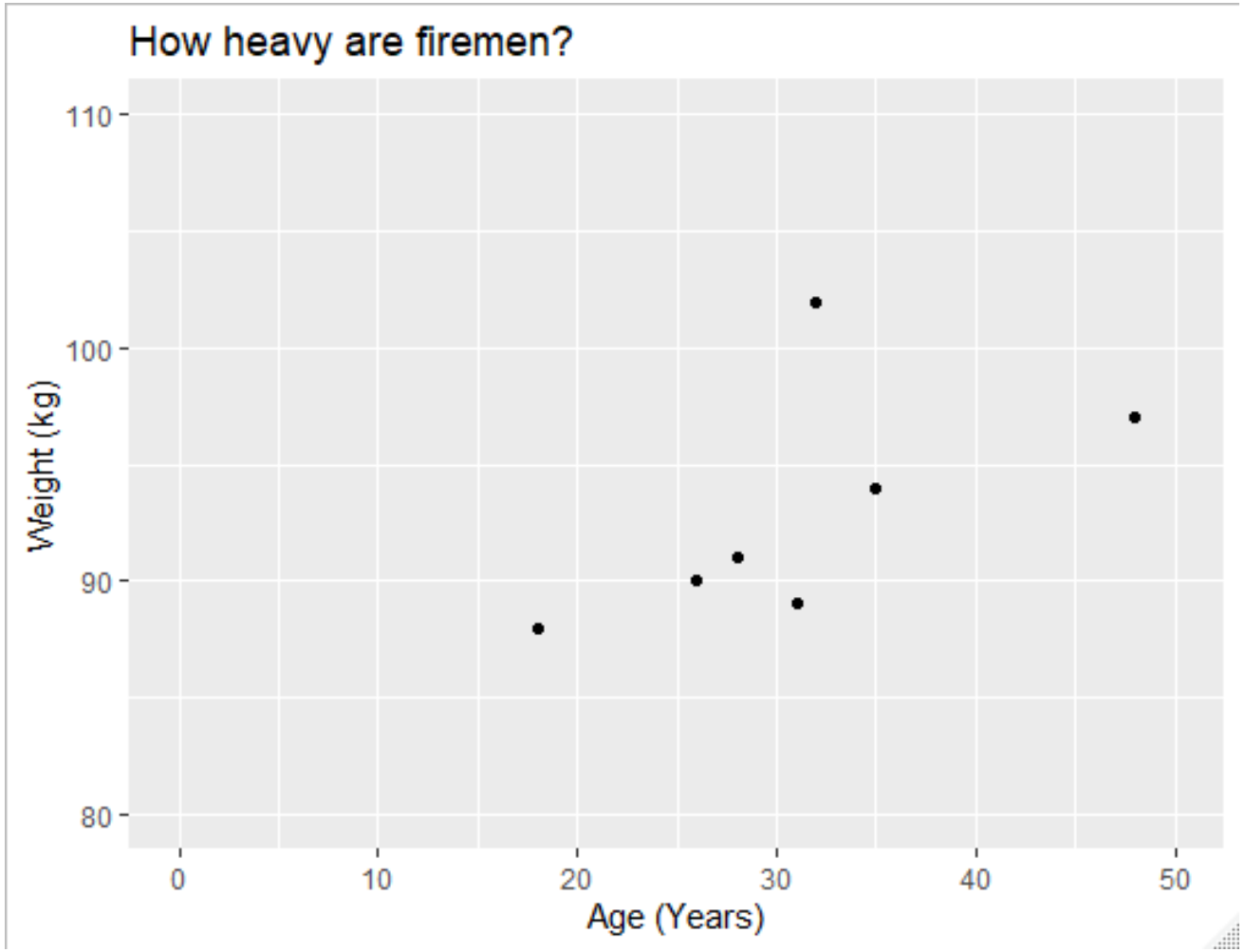
  Equivalent `_y_` versions also exist

- Switching axes
  - `coord_flip()`

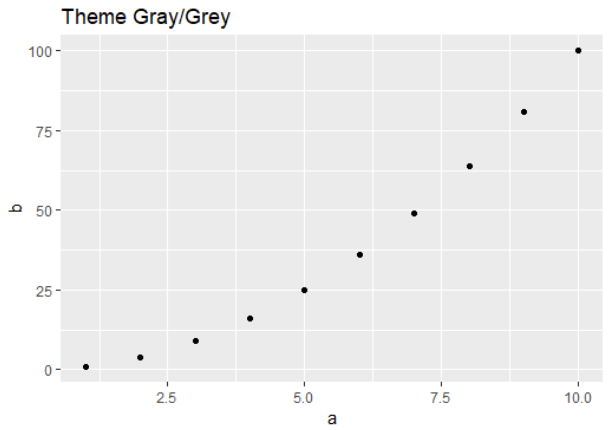
- Adjusting ranges
  - `scale_x_continuous()`
    - `limits=c(-5,5)`
    - `breaks=seq(from=-5,by=2,to=5)`
    - `minor_breaks`
    - `labels`
  - `coord_cartesian()`
    - `xlim=c(-5,5)`
    - `ylim=c(10,20)`

Annotation and scaling example

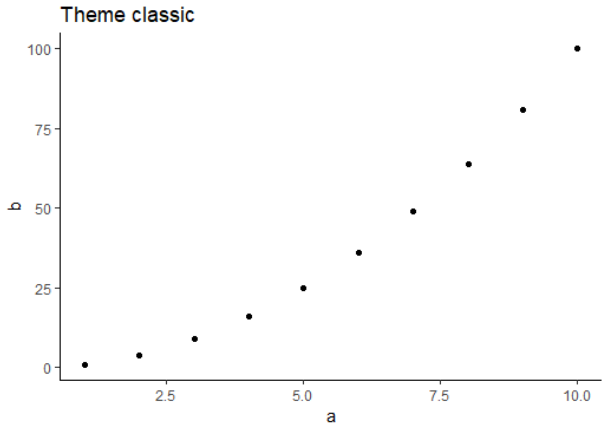
```
trumpton %>%
  ggplot(aes(x=Age, y=Weight))+
  geom_point() +

  xlab("Age (Years)")+
  ylab("Weight (kg)")+
  ggtitle("How heavy are firemen?")+

  coord_cartesian(
    xlim=c(0,50),
    ylim=c(80,110)
  )
```

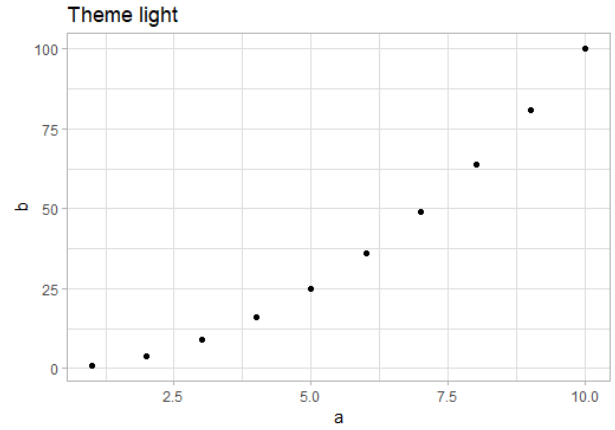


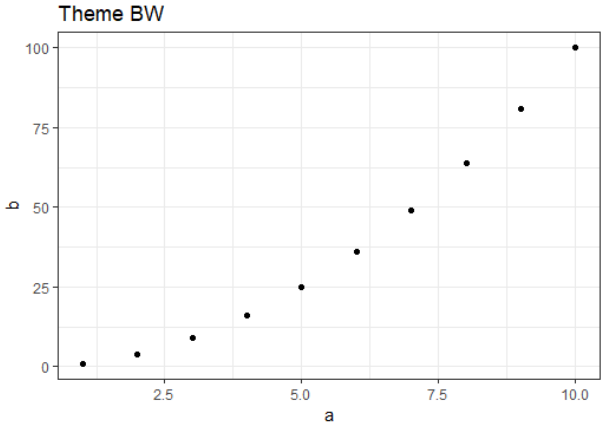
ggPlot Themes

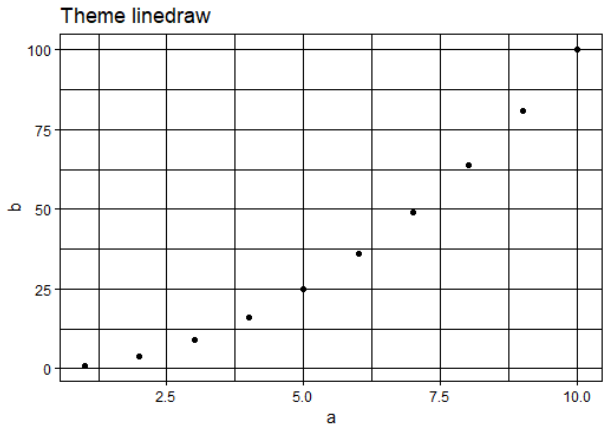


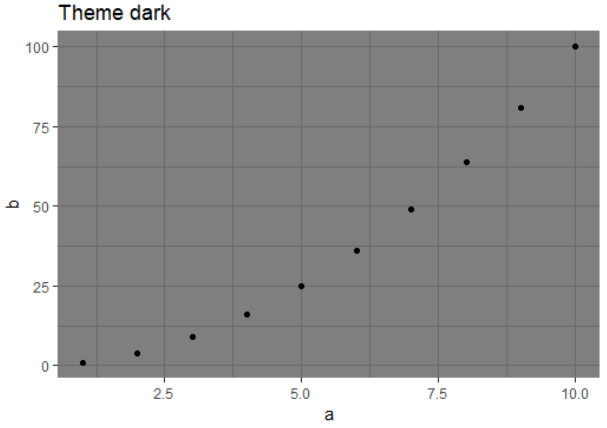
- theme_grey()
- theme_bw()
- theme_dark()
- theme_light()
- theme_minimal()
- theme_classic()
- theme_linedraw()

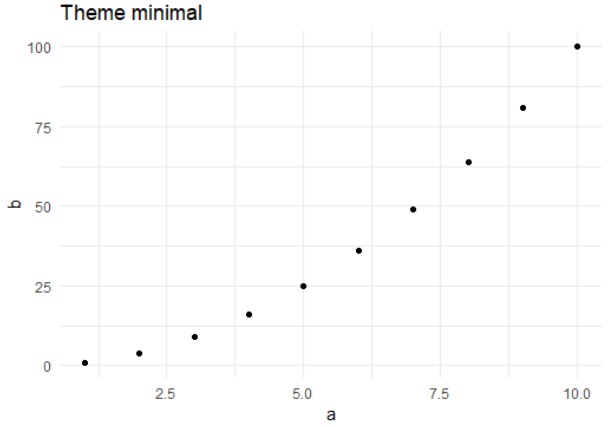












Setting and Customising themes

- Globally

```
theme_set(theme_bw(base_size=14))
theme_update(plot.title = element_text(hjust = 0.5))
```

- In a single plot

```
+theme_dark()
+theme(plot.title = element_text(hjust=0.5))
```

What can you customise?

```
theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x,
  axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left,
  axis.title.y.right, axis.text, axis.text.x, axis.text.x.top,
  axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right,
  axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom,
  axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length,
  axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y,
  axis.line.y.left, axis.line.y.right, legend.background, legend.margin,
  legend.spacing, legend.spacing.x, legend.spacing.y, legend.key,
  legend.key.size, legend.key.height, legend.key.width, legend.text,
  legend.text.align, legend.title, legend.title.align, legend.position,
  legend.direction, legend.justification, legend.box, legend.box.just,
  legend.box.margin, legend.box.background, legend.box.spacing,
  panel.background, panel.border, panel.spacing, panel.spacing.x,
  panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor,
  panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x,
  panel.grid.minor.y, panel.ontop, plot.background, plot.title,
  plot.subtitle, plot.caption, plot.tag, plot.tag.position, plot.margin,
  strip.background, strip.background.x, strip.background.y,
  strip.placement, strip.text, strip.text.x, strip.text.y,
  strip.switch.pad.grid, strip.switch.pad.wrap
```

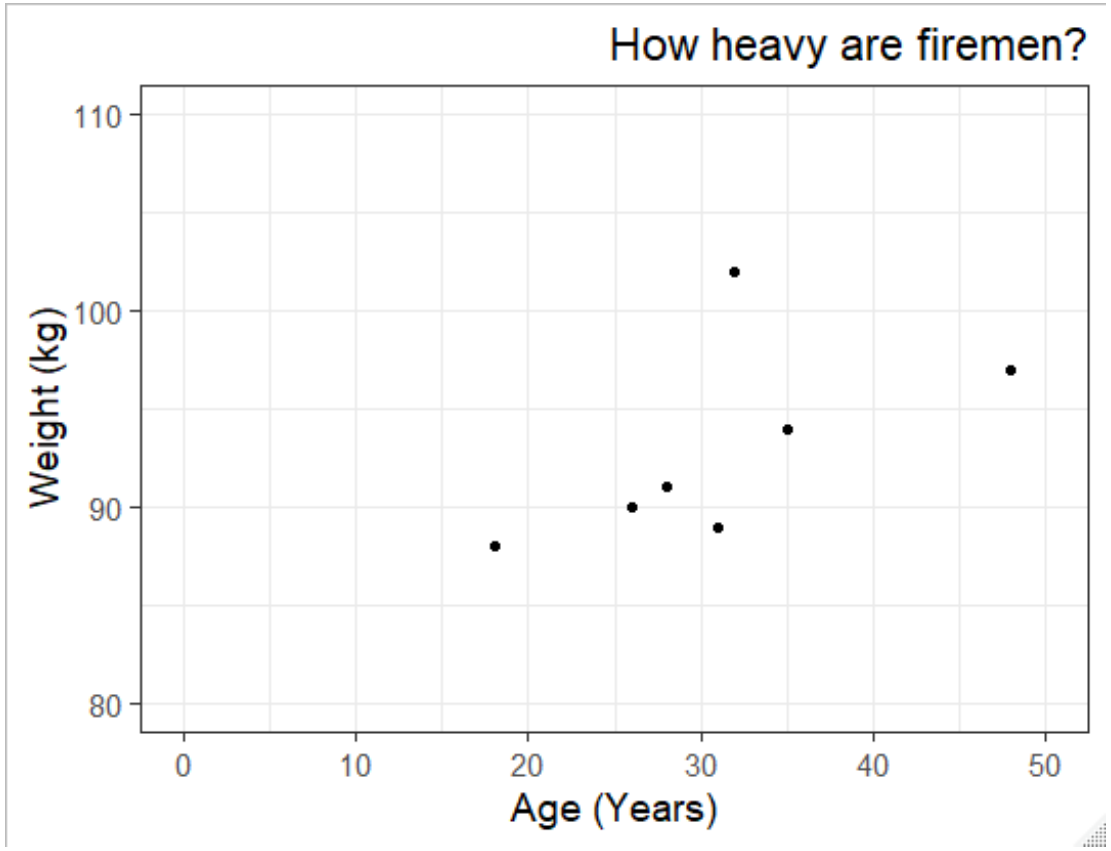
https://ggplot2.tidyverse.org/reference/theme.html

Theme setting example

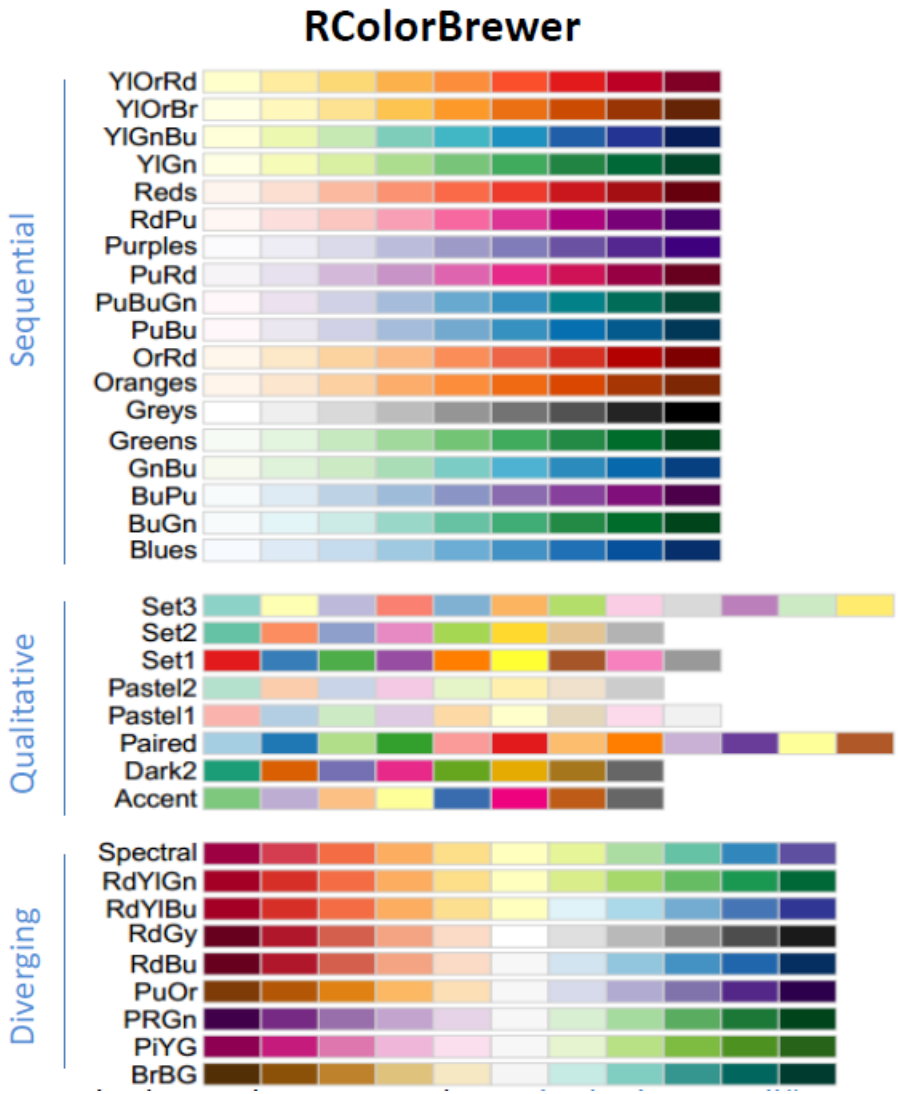
```
theme_set(theme_bw(base_size = 14))
theme_update(plot.title = element_text(hjust=1))
```

OR

```
my.plot +
theme_bw(base_size = 14) +
theme(plot.title = element_text(hjust=1))
```



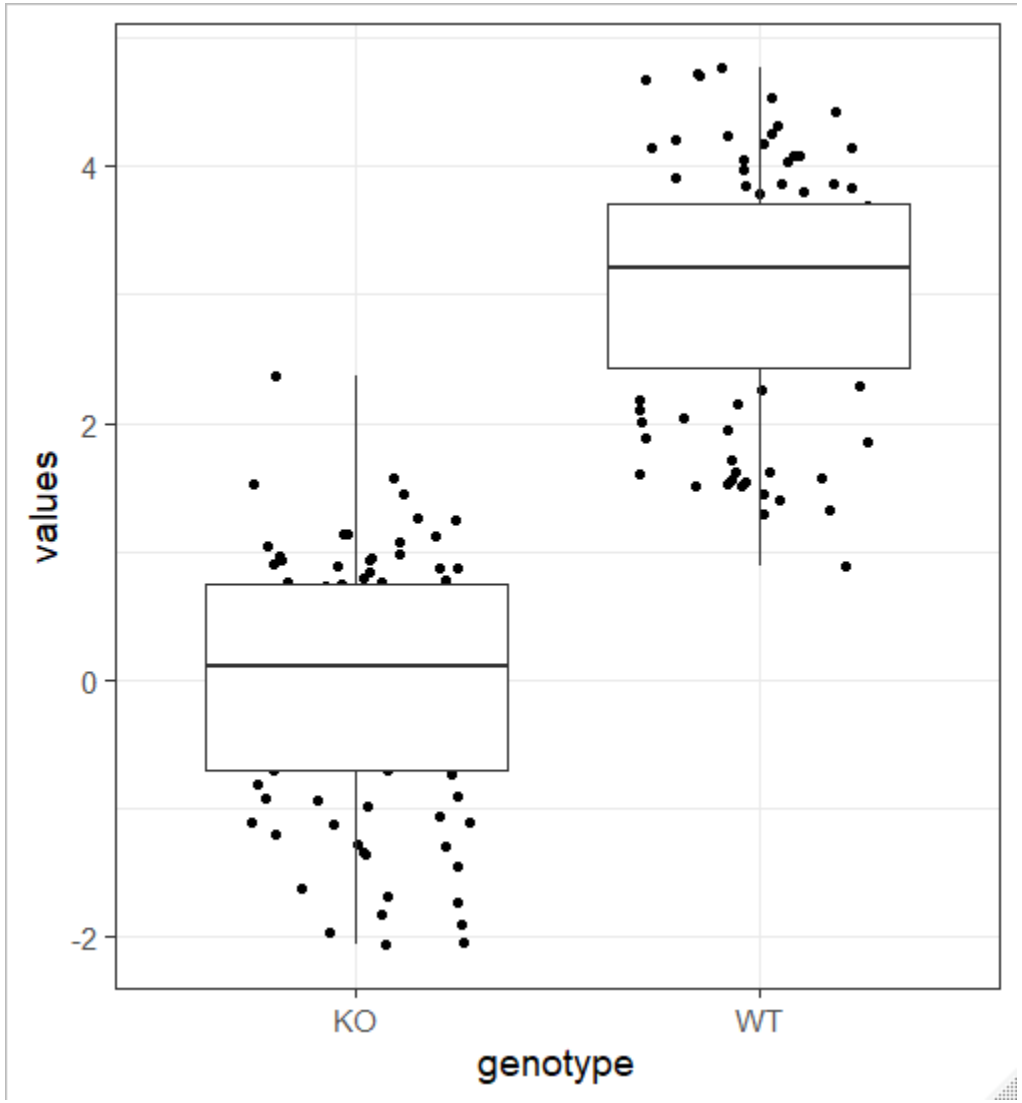
ColorBrewer Scales



`scale_color_brewer` for qualitative
`scale_color_distiller` for quantitative

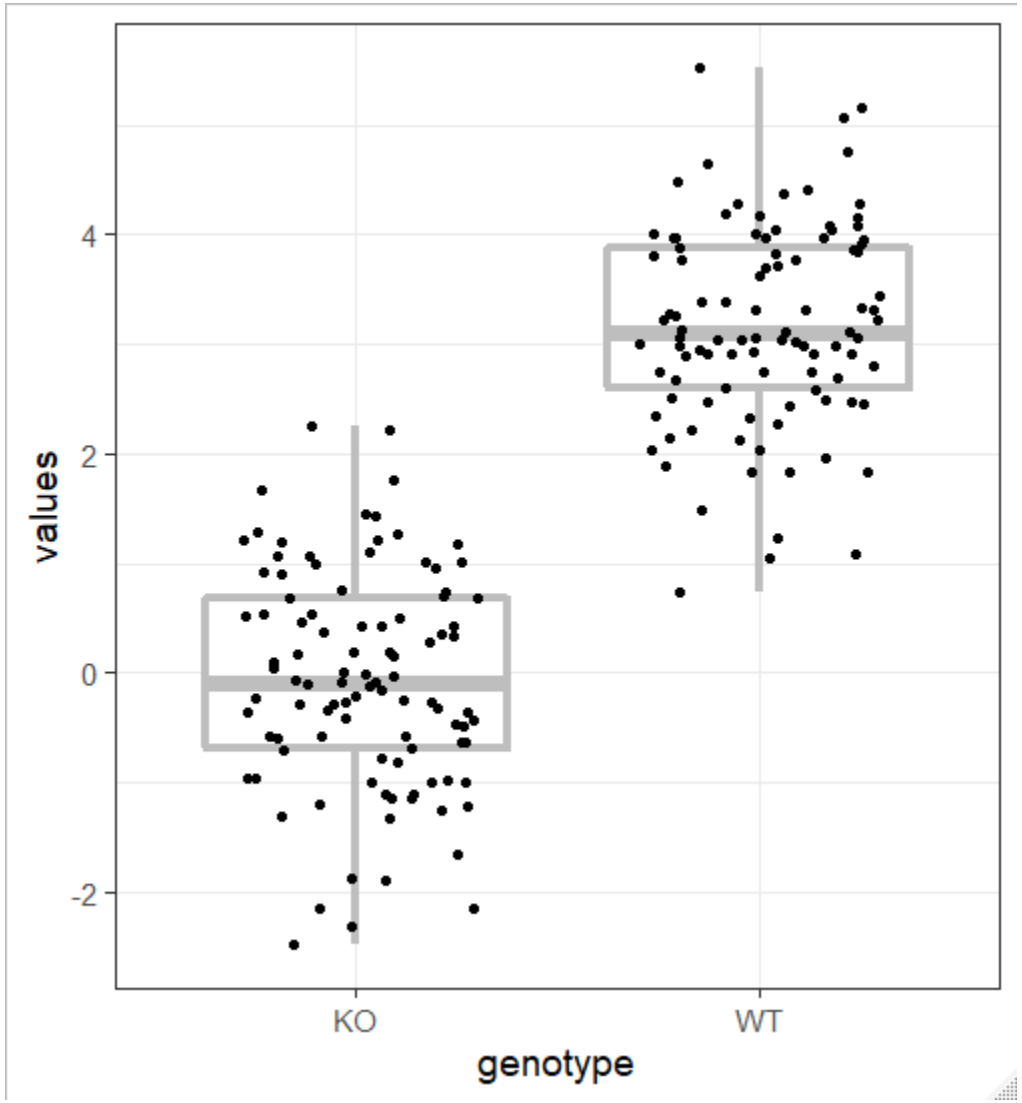
Overlaying raw data and summaries

```
many.values %>%
  group_by(genotype) %>%
  sample_n(100) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_jitter(height=0, width = 0.3) +
  geom_boxplot()
```



Overlaying raw data and summaries

```
many.values %>%
  group_by(genotype) %>%
  sample_n(100) %>%
  ggplot(aes(x=genotype, y=values)) +
  geom_boxplot(size=1.5, colour="grey") +
  geom_jitter(height=0, width = 0.3)
```



Saving plots

ggsave() saves the last plot.

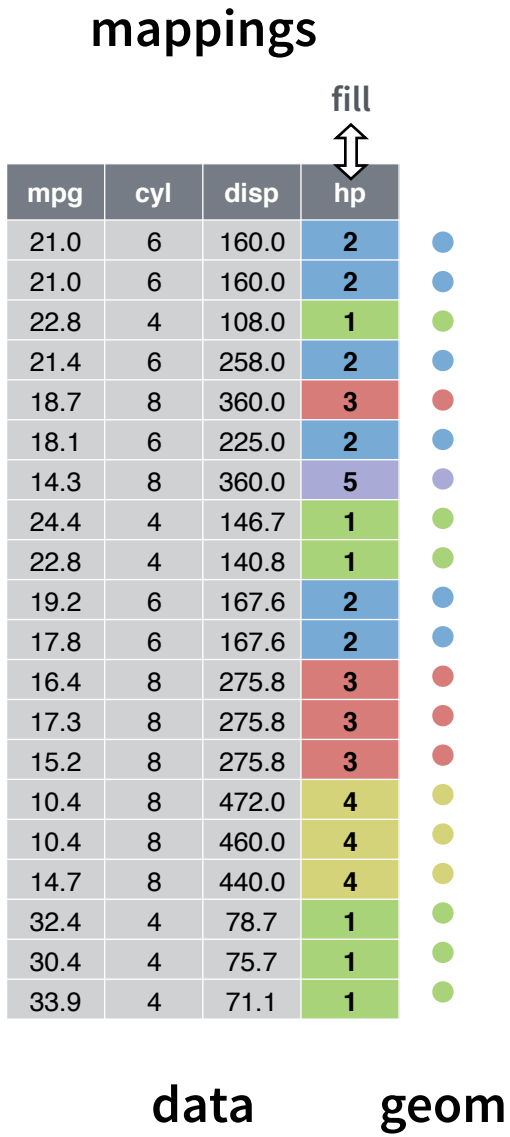
Uses size on screen:

```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height = 6)
```

To make a graph



1. Pick a **data** set

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom** to display cases

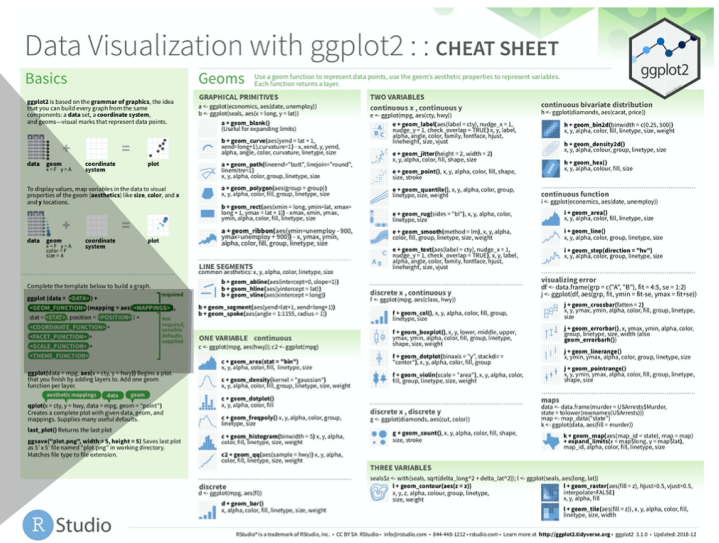
3. **Map** aesthetic properties to variables

Courtesy: Hadley Wickham

A ggplot2 template

Make any plot by filling in the parameters of this template

```
ggplot (data = <DATA>) +                                    ] required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),                  ]
 stat = <STAT>, position = <POSITION>) +                    ] Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +                                     ]
<FACET_FUNCTION> +                                          ]
<SCALE_FUNCTION> +                                          ]
<THEME_FUNCTION>                                            ]
```



Courtesy: Hadley Wickham

https://community.rstudio.com

ggplot2 part of the tidyverse 3.3.2

Search... Reference News Articles Extensions

Overview

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Installation

```
# The easiest way to get ggplot2 is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just ggplot2:
install.packages("ggplot2")

# Or the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/ggplot2")
```

Cheatsheet

Links

Download from CRAN at
https://cloud.r-project.org/package=ggplot2

Browse source code at
https://github.com/tidyverse/ggplot2/

Report a bug at
https://github.com/tidyverse/ggplot2/issues

Learn more at
https://r4ds.had.co.nz/data-visualisation.html

Extensions at
https://exts.ggplot2.tidyverse.org/gallery/

License

Full license

GPL-2 | file LICENSE

Community

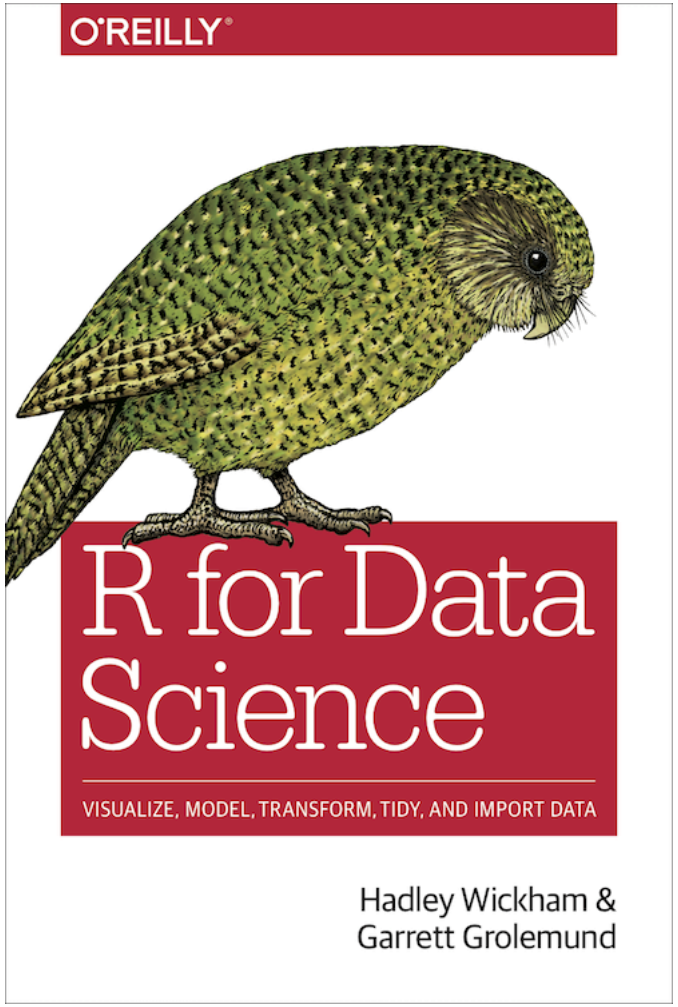
Contributing guide

Code of conduct

Citation

Books

- Elements of Data Analytic Style, Jeff Leek
- R Programming for Data Science, Roger Peng
-The Art of Data Science, Roger Peng
- R Cookbook. Both a website, and a book, Winston Chang
- R for Data Science. Both a website and a book. Hadley Wickham and Garrett Grolemund.



Courtesy: Hadley Wickham


Thank you