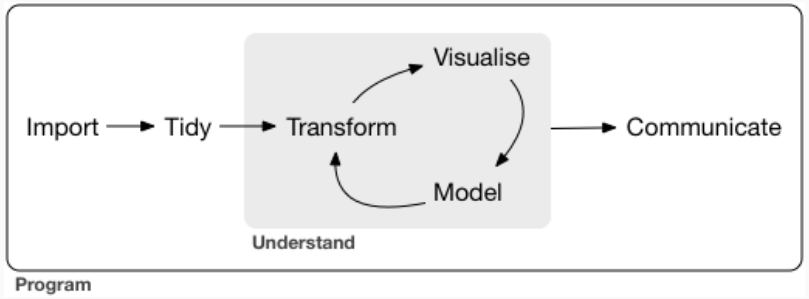
Cleaning, Summarizing, and Visualizing Data

Jeroen Mahieu
jeroen.mahieu@vu.nl
Updated 2022-02-14

The Data Science Pipeline

- Quantitative Research is about numeric `data`



Cleaning (Tidying) Data

- According a to 2014 NYTimes article, "data scientists [...] spend from ***50 percent to 80 percent of their time*** mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets."
- Luckily we have some powerful tools to help us out.
- Here, we will focus on `dplyr` which is part of the `tidyverse`
- (When you work with large datasets (+100k rows with many columns) learn to use `data.table` which is much faster but has more difficult syntax)

`dplyr` Overview

- You are ***highly encouraged*** to read through Hadley Wickham's chapter. It's clear and concise.
- Also check out this great "cheatsheet" here.
- The package is organized around a set of **verbs**, i.e. *actions* to be taken.
- We operate on `data.frames` or `tibbles` (*nicer looking* data.frames.)
- All *verbs* work as follows:

$$\text{verb}(\underbrace{\text{data.frame}}_{\text{1st argument}}, \underbrace{\text{what to do}}_{\text{2nd argument}})$$

- Alternatively you can (should) use the `pipe` operator `%>%`:

$$\underbrace{\text{data.frame}}_{\text{1st argument}} \quad \underbrace{\%>\%}_{\text{``pipe'' operator}} \quad \text{verb}(\underbrace{\text{what to do}}_{\text{2nd argument}})$$

Main `dplyr` Verbs

`filter()`: Choose observations based on a certain condition (i.e. subset)

`arrange()`: Reorder rows

`select()`: Select variables by name

`mutate()`: Create new variables out of existing ones

`summarise()`: Collapse data to a single summary

`group_by()`: All the above can be used in conjunction with `group_by()` to use function on groups rather than entire data

Data: Red Wine Quality

This dataset contains information about Portuguese "Vinho Verde" wine (more details | download link). Each row represents one wine sample.

```
library(tidyverse)
wine <- read_csv("data/winequality-red.csv") #import data as dataframe "wine"
head(wine[,1:6]) # show first 6 lines of first 6 variables
```

```
## # A tibble: 6 x 6
##   `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
##             <dbl>              <dbl>         <dbl>            <dbl>     <dbl>
## 1             7.4               0.7             0              1.9     0.076
## 2             7.8               0.88            0              2.6     0.098
## 3             7.8               0.76            0.04           2.3     0.092
## 4            11.2               0.28            0.56           1.9     0.075
## 5             7.4               0.7             0              1.9     0.076
## 6             7.4               0.66            0              1.8     0.075
## # ... with 1 more variable: `free sulfur dioxide` <dbl>
```

Data: Red Wine Quality

What variables does this dataset contain?

```
str(wine)
```

```
## spec_tbl_df [1,599 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ fixed acidity       : num [1:1599] 7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile acidity    : num [1:1599] 0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric acid         : num [1:1599] 0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual sugar      : num [1:1599] 1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num [1:1599] 0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free sulfur dioxide : num [1:1599] 11 25 15 17 11 13 15 15 9 17 ...
##  $ total sulfur dioxide: num [1:1599] 34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num [1:1599] 0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num [1:1599] 3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num [1:1599] 0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num [1:1599] 9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : num [1:1599] 5 5 5 6 5 5 5 7 7 5 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `fixed acidity` = col_double(),
##   ..   `volatile acidity` = col_double(),
```

Filtering observations

```
filter()
```

Example: Which wines have at least a alchohol percentage of 10?

filter

Example: Which wines have at least a alchohol percentage of 10?

```
wine %>%
  filter(alcohol > 10)
```

```
## # A tibble: 852 x 12
##    `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
##              <dbl>              <dbl>         <dbl>            <dbl>     <dbl>
##  1             7.5              0.5            0.36              6.1     0.071
##  2             7.5              0.5            0.36              6.1     0.071
##  3             8.5              0.28           0.56              1.8     0.092
##  4             6.7              0.675          0.07              2.4     0.089
##  5             6.9              0.685          0                 2.5     0.105
##  6             7.8              0.6            0.14              2.4     0.086
##  7             7.3              0.45           0.36              5.9     0.074
##  8             7.3              0.45           0.36              5.9     0.074
##  9             7.5              0.49           0.2               2.6     0.332
## 10             8.1              0.66           0.22              2.2     0.069
## # ... with 842 more rows, and 7 more variables: `free sulfur dioxide` <dbl>,
## #   `total sulfur dioxide` <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <dbl>
```

Operators

Standard comparison operators:

- `>` : greater than,
- `<` : smaller than,
- `>=` : greater than or equal to,
- `<=` : smaller than or equal to,
- `!=` : not equal to,
- `==` : equal to.

Logical operators:

1. `x & y` : `x` **and** `y`
2. `x | y` : `x` **or** `y`
3. `!y` : **not** `y`

`R` has the convenient `x %in% y` operator (conversely `!x %in% y`), `TRUE` if `x` is *a member of* `y`.

```
3 %in% 1:3
```

```
## [1] TRUE
```

```
c(2,5) %in% 2:10  # also vectorized
```

```
## [1] TRUE TRUE
```

```
c("V","Uni") %in% c("Vrije","Universiteit")  # also strings
```

```
## [1] FALSE FALSE
```

`filter` with a logical operator

Example: Which wines have at least an alchohol percentage of 10 ***and*** a quality score `<` 6?

`filter` with a logical operator

Example: Which wines have at least an alchohol percentage of 10 ***and*** a quality score `<` 6?

```
wine %>%
  filter(alcohol > 10 & quality < 6)
```

```
## # A tibble: 235 x 12
##    `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
##              <dbl>              <dbl>         <dbl>            <dbl>     <dbl>
##  1             7.5              0.5           0.36             6.1       0.071
##  2             7.5              0.5           0.36             6.1       0.071
##  3             6.7              0.675         0.07             2.4       0.089
##  4             7.3              0.45          0.36             5.9       0.074
##  5             7.3              0.45          0.36             5.9       0.074
##  6             8.1              0.66          0.22             2.2       0.069
##  7             4.6              0.52          0.15             2.1       0.054
##  8             7.2              0.725         0.05             4.65      0.086
##  9             7.2              0.725         0.05             4.65      0.086
## 10             6.6              0.705         0.07             1.6       0.076
## # ... with 225 more rows, and 7 more variables: `free sulfur dioxide` <dbl>,
## #   `total sulfur dioxide` <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
```

Creating new variables

```
mutate()
```

Example: What is the total acidity, defined as `fixed acidity` + `volatile acidity` ?

🚨 Note the use of `` ` ` `` when you refer to a variable consisting of two or more words. This is quite confusing for inexperienced users. Therefore it is good practice to use one-word variable names (e.g., `fixed_acidity`)

mutate

Example: What is the total acidity, defined as `fixed acidity` + `volatile acidity` ? Save the new dataset as `wine2`

```
wine2 <- wine  %>%
  mutate(`total acidity`= `fixed acidity` + `volatile acidity`)

print(wine2[, c('fixed acidity', 'volatile acidity', 'total acidity')])
```

```
## # A tibble: 1,599 x 3
##    `fixed acidity` `volatile acidity` `total acidity`
##              <dbl>              <dbl>           <dbl>
##  1             7.4               0.7             8.1
##  2             7.8               0.88            8.68
##  3             7.8               0.76            8.56
##  4            11.2               0.28           11.5
##  5             7.4               0.7             8.1
##  6             7.4               0.66            8.06
##  7             7.9               0.6             8.5
##  8             7.3               0.65            7.95
##  9             7.8               0.58            8.38
## 10             7.5               0.5             8
## # ... with 1,589 more rows
```

`mutate`

Example: Create a new logical variable `low_alcohol` which is `TRUE` if the alcohol percentage `<=` 10

mutate

Example: Create a new logical variable `low_alcohol` which is `TRUE` if the alcohol percentage `<=` 10. Save the new dataset as `wine2`

```
wine2 <- wine  %>%
  mutate(low_alcohol = (alcohol <= 10))

print(wine2[, c("alcohol", 'low_alcohol')])
```

```
## # A tibble: 1,599 x 2
##    alcohol low_alcohol
##      <dbl> <lgl>
##  1     9.4 TRUE
##  2     9.8 TRUE
##  3     9.8 TRUE
##  4     9.8 TRUE
##  5     9.4 TRUE
##  6     9.4 TRUE
##  7     9.4 TRUE
##  8    10   TRUE
##  9     9.5 TRUE
## 10    10.5 FALSE
```

Summarising variables

```
summarise()
```

Example: What is the mean `fixed acidity` ?

summarise

Example: What is the mean `fixed acidity` ?

```
wine  %>%
  summarise(`mean acidity`= mean(`fixed acidity`))
```

```
## # A tibble: 1 x 1
##   `mean acidity`
##            <dbl>
## 1           8.32
```

Summarising variables by group

```
group_by()
summarise()
```

Example: What is the median `fixed acidity` per `quality` score?

`summarise` with `group_by`

Example: What is the median `fixed acidity` per `quality` score?

```
wine  %>%
  group_by(quality) %>%
    summarise(`median acidity`= median(`fixed acidity`))
```

```
## # A tibble: 6 x 2
##   quality `median acidity`
##     <dbl>            <dbl>
## 1       3             7.5
## 2       4             7.5
## 3       5             7.8
## 4       6             7.9
## 5       7             8.8
## 6       8             8.25
```

Chaining 🔗 Commands Together

Works for all `dplyr` verbs:

```
wine  %>%
  mutate(`total acidity`= `fixed acidity` + `volatile acidity`) %>%
  filter(`total acidity` > 8)
```

```
## # A tibble: 1,024 x 13
##    `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
##              <dbl>              <dbl>         <dbl>            <dbl>     <dbl>
##  1             7.4              0.7           0                 1.9      0.076
##  2             7.8              0.88          0                 2.6      0.098
##  3             7.8              0.76          0.04              2.3      0.092
##  4            11.2              0.28          0.56              1.9      0.075
##  5             7.4              0.7           0                 1.9      0.076
##  6             7.4              0.66          0                 1.8      0.075
##  7             7.9              0.6           0.06              1.6      0.069
##  8             7.8              0.58          0.02              2        0.073
##  9             7.8              0.61          0.29              1.6      0.114
## 10             8.9              0.62          0.18              3.8      0.176
## # ... with 1,014 more rows, and 8 more variables: `free sulfur dioxide` <dbl>,
## #   `total sulfur dioxide` <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
```

Chaining 🔗 Commands Together

Works for all `dplyr` verbs:

```
wine3 <- wine  %>%
            mutate(`total acidity`= `fixed acidity` + `volatile acidity`,
                   alcohol_hi = alcohol > 10)
print(wine3[, c('fixed acidity', 'volatile acidity', 'alcohol', 'total acidity', 'alcohol_hi')])
```

```
## # A tibble: 1,599 x 5
##    `fixed acidity` `volatile acidity` alcohol `total acidity` alcohol_hi
##              <dbl>              <dbl>   <dbl>           <dbl> <lgl>
##  1             7.4              0.7      9.4            8.1  FALSE
##  2             7.8              0.88     9.8            8.68 FALSE
##  3             7.8              0.76     9.8            8.56 FALSE
##  4            11.2              0.28     9.8           11.5  FALSE
##  5             7.4              0.7      9.4            8.1  FALSE
##  6             7.4              0.66     9.4            8.06 FALSE
##  7             7.9              0.6      9.4            8.5  FALSE
##  8             7.3              0.65    10              7.95 FALSE
##  9             7.8              0.58     9.5            8.38 FALSE
## 10             7.5              0.5     10.5            8    TRUE
## # ... with 1,589 more rows
```

Missing Values: `NA`

- Whenever a value is *missing*, we code it as `NA`.

```
x <- NA
```

- `R` propagates `NA` through operations:

```
NA > 5
```

```
## [1] NA
```

```
NA + 10
```

```
## [1] NA
```

- `is.na(x)` function returns `TRUE` if `x` is an `NA`.

```
is.na(x)
```

```
## [1] TRUE
```

Task 1: Data wrangling

10:00

Open the `session1_exercises.R` script

1. Which wines have a `quality` of 3 or 6?
2. Create a new variable called `hi_sugar` which is `TRUE` if `residual sugar >= 2`. Save the new dataset again as 'wine'
3. Calculate the mean chlorides and maximum density by `hi_sugar` group. Use `mean()` and `max()`

Summary statistics with `stargazer`

The `stargazer` package creates well-formattedy summary statistics and regression tables with very little effort in multiple formats.

Using the output table as text (`.txt`) gives a quick view of results. Printing the output table as `.html` , produces tables that can be simply copy-pasted in a Word document.

🚨 `stargazer` works only with `data.frames` not with `tibbles` (I know...). So you have to make sure that your data is in `data.frame` format first

Summary statistics with `stargazer`

```
library(stargazer)
wine <- as.data.frame(wine)

stargazer(wine,
          type = 'text')
```

```
##
## ===================================================
## Statistic              N    Mean  St. Dev.  Min  Pct
## ---------------------------------------------------
## fixed acidity        1,599 8.320   1.741   4.600  7.
## volatile acidity     1,599 0.528   0.179   0.120  0.
## citric acid          1,599 0.271   0.195     0     0
## residual sugar       1,599 2.539   1.410   0.900  1.
## chlorides            1,599 0.087   0.047   0.012  0.
## free sulfur dioxide  1,599 15.875  10.460    1
## total sulfur dioxide 1,599 46.468  32.895    6
## density              1,599 0.997   0.002   0.990  0.
## pH                   1,599 3.311   0.154   2.740  3.
## sulphates            1,599 0.658   0.170   0.330  0.
## alcohol              1,599 10.423  1.066   8.400  9.
## quality              1,599 5.636   0.808     3
## ---------------------------------------------------
```

Summary statistics with `stargazer`

```
library(stargazer)
wine <- as.data.frame(wine)

stargazer(wine,
          type = 'html',
          out='wine_summary.html',
          title= 'Descriptive Statistics')
```

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
fixed acidity	1,599	8.320	1.741	4.600	7.100	9.200	15.900
volatile acidity	1,599	0.528	0.179	0.120	0.390	0.640	1.580
citric acid	1,599	0.271	0.195	0	0.1	0.4	1
residual sugar	1,599	2.539	1.410	0.900	1.900	2.600	15.500
chlorides	1,599	0.087	0.047	0.012	0.070	0.090	0.611
free sulfur dioxide	1,599	15.875	10.460	1	7	21	72
total sulfur dioxide	1,599	46.468	32.895	6	22	62	289
density	1,599	0.997	0.002	0.990	0.996	0.998	1.004
pH	1,599	3.311	0.154	2.740	3.210	3.400	4.010
sulphates	1,599	0.658	0.170	0.330	0.550	0.730	2.000
alcohol	1,599	10.423	1.066	8.400	9.500	11.100	14.900
quality	1,599	5.636	0.808	3	5	6	8

Summary statistics with `stargazer`, selected statistics

```
library(stargazer)
wine <- as.data.frame(wine)

stargazer(wine,
          type = 'html',
          out='wine_summary1.html',
          title= 'Descriptive Statistics',
          summary.stat = c("mean", "median", "min", "max"))
```

Descriptive Statistics

Statistic	Mean	Median	Min	Max
fixed acidity	8.320	7.900	4.600	15.900
volatile acidity	0.528	0.520	0.120	1.580
citric acid	0.271	0.3	0	1
residual sugar	2.539	2.200	0.900	15.500
chlorides	0.087	0.079	0.012	0.611
free sulfur dioxide	15.875	14	1	72
total sulfur dioxide	46.468	38	6	289
density	0.997	0.997	0.990	1.004
pH	3.311	3.310	2.740	4.010
sulphates	0.658	0.620	0.330	2.000
alcohol	10.423	10.200	8.400	14.900
quality	5.636	6	3	8

Summary statistics with `stargazer`, by group

Although there are other ways, an easy way to obtain statistics by group is to first create a new subset dataframe with `filter()`

```
wine_hi_alcohol <- wine %>%
  filter(alcohol > 10)

library(stargazer)
wine_hi_alcohol <- as.data.frame(wine_hi_alcohol)

stargazer(wine_hi_alcohol,
          type = 'html',
          out='wine_hi_alcohol-summary.html',
          title= 'Descriptive Statistics High Alcohol Wines')
```

Descriptive Statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
fixed acidity	852	8.311	1.918	4.600	6.900	9.500	15.900
volatile acidity	852	0.496	0.188	0.120	0.360	0.600	1.580
citric acid	852	0.293	0.202	0.000	0.090	0.460	0.790
residual sugar	852	2.620	1.338	0.900	2.000	2.700	13.900
chlorides	852	0.079	0.027	0.012	0.065	0.087	0.343
free sulfur dioxide	852	15.277	10.298	1	6.8	21	72
total sulfur dioxide	852	39.165	28.872	6	19	51	289
density	852	0.996	0.002	0.990	0.995	0.997	1.003
pH	852	3.334	0.157	2.870	3.230	3.420	4.010
sulphates	852	0.675	0.139	0.370	0.580	0.760	1.360
alcohol	852	11.210	0.856	10.033	10.500	11.700	14.900
quality	852	5.934	0.827	3	5	6	8

Task 2: Summary Statistics

05:00

Open the `session1_exercises.R` script

1. Calculate the number of number of observations (N), the mean, and standard deviation for all wines with a pH `<=` 3.2

Visualising data with `ggplot2`

One of the strengths of `R` is its ability to create elegant graphs of your data with little effort.

We will use `ggplot2` one of the core members of the `tidyverse`. I recommend reading the chapter of Hadley Wickham's book to learn more

The basic syntax of `ggplot2` looks like this:

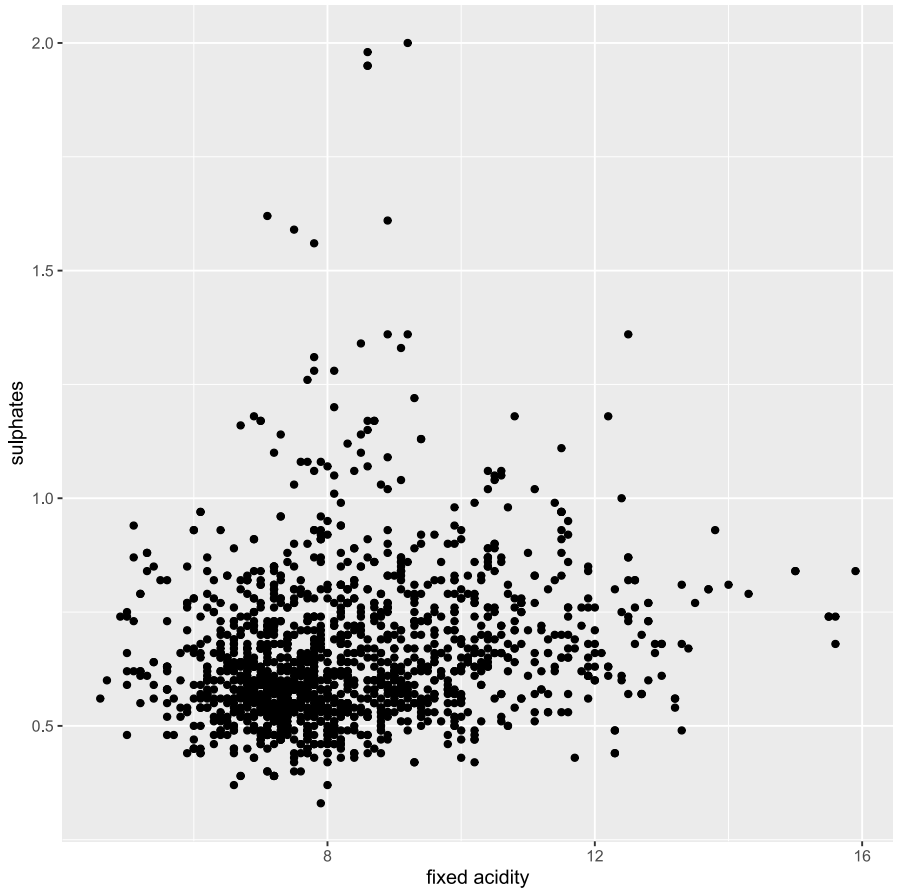
```
ggplot(data = <DATA>)+
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

the geom function is the type of layer you want to add (scatter, line, ...)

the mapping argument defines how variables in your dataset are mapped to visual properties. It is always combined with `aes()`

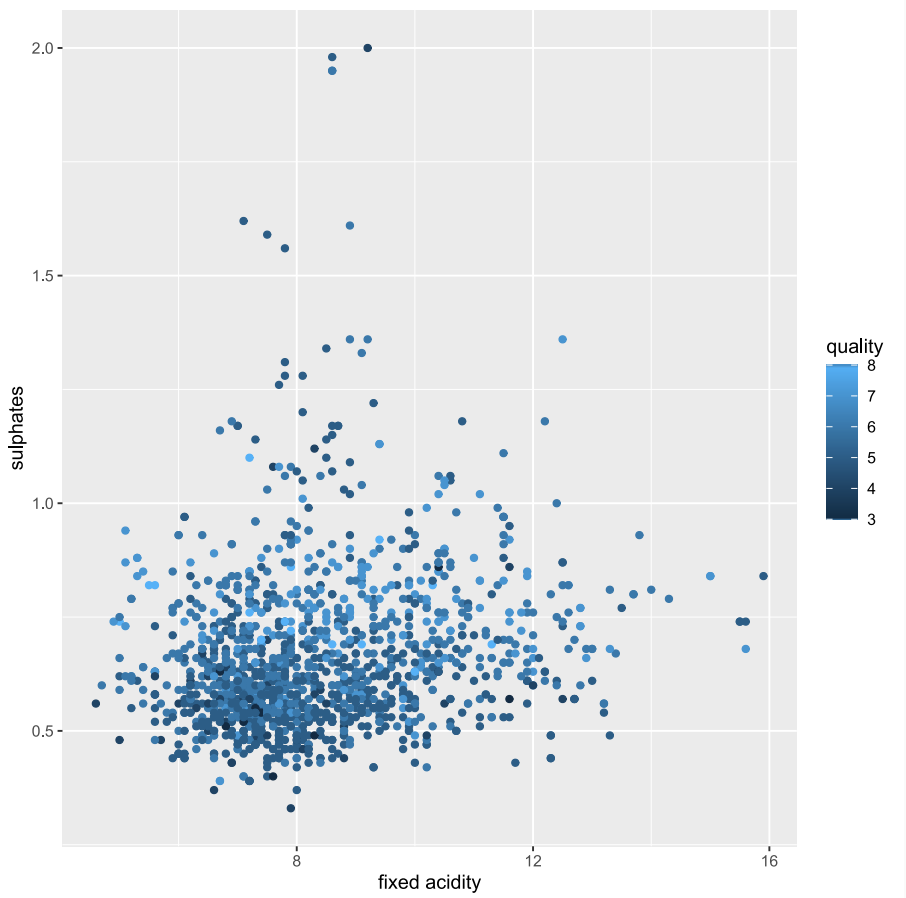
Making a scatterplot with `ggplot2`

```
ggplot(data=wine)+
  geom_point(mapping=aes(x = `fixed acidity`,
                         y = sulphates))
```



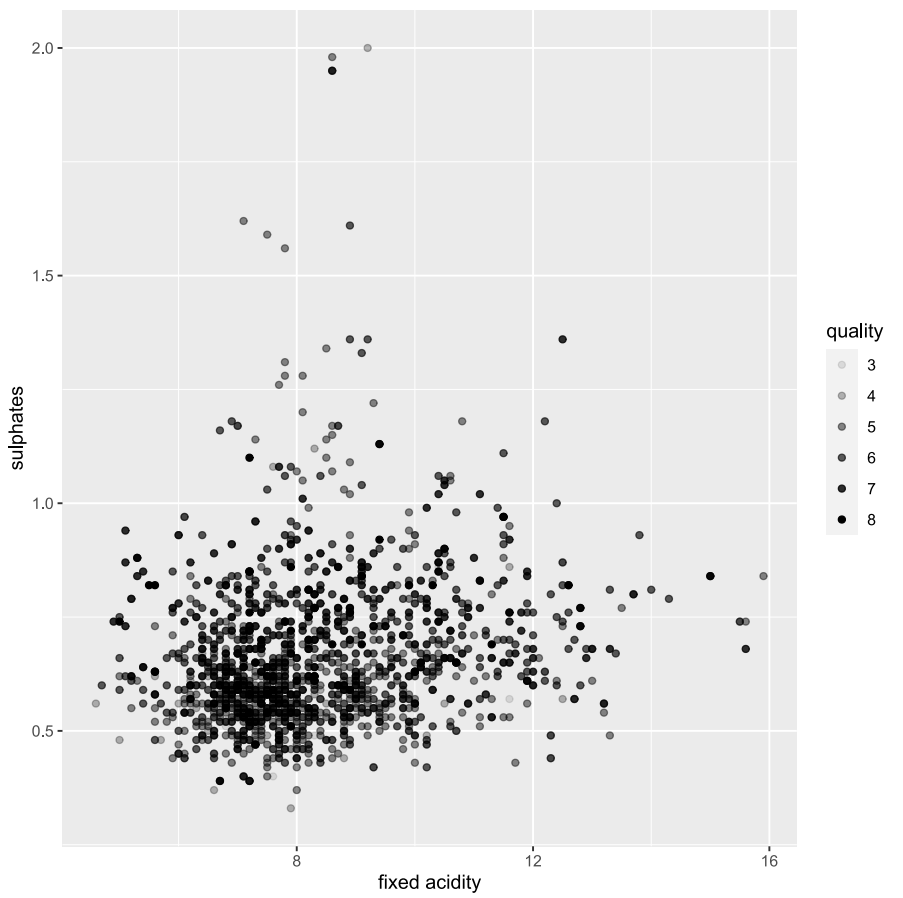
Making a scatterplot with `ggplot2`

```
ggplot(data=wine)+
  geom_point(mapping=aes(x = `fixed acidity`,
                         y = sulphates,
                         color = quality))
```



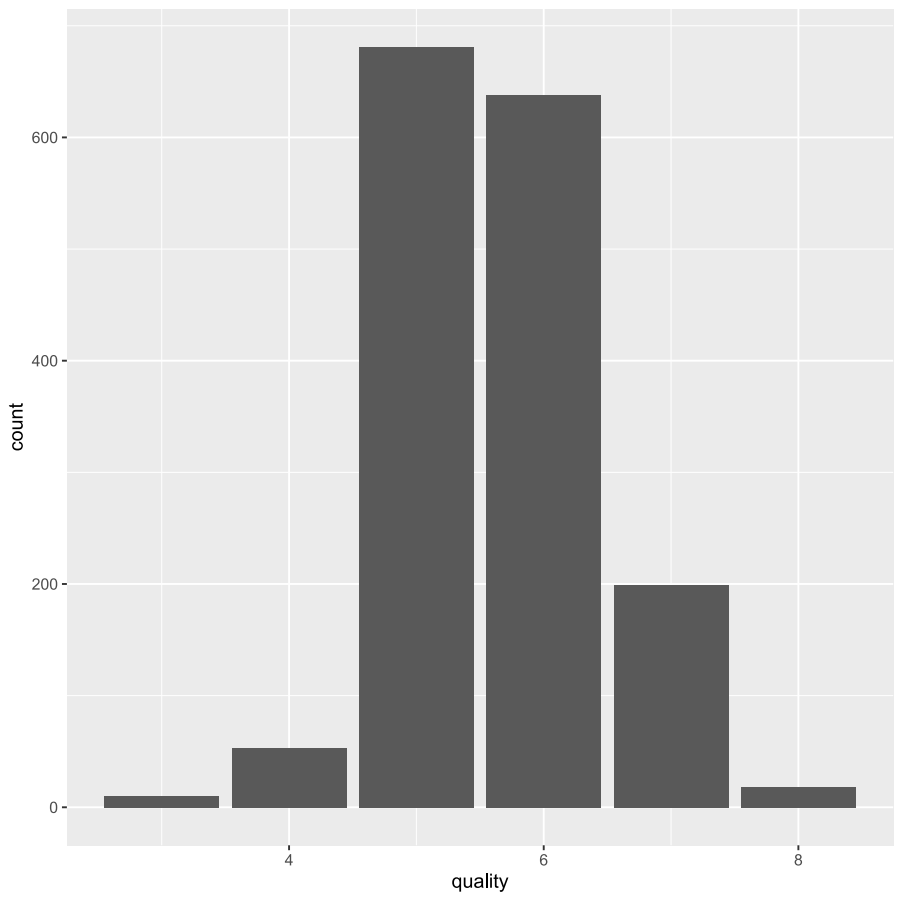
Making a scatterplot with `ggplot2`

```
ggplot(data=wine)+
  geom_point(mapping=aes(x = `fixed acidity`,
                         y = sulphates,
                         alpha = quality))
```



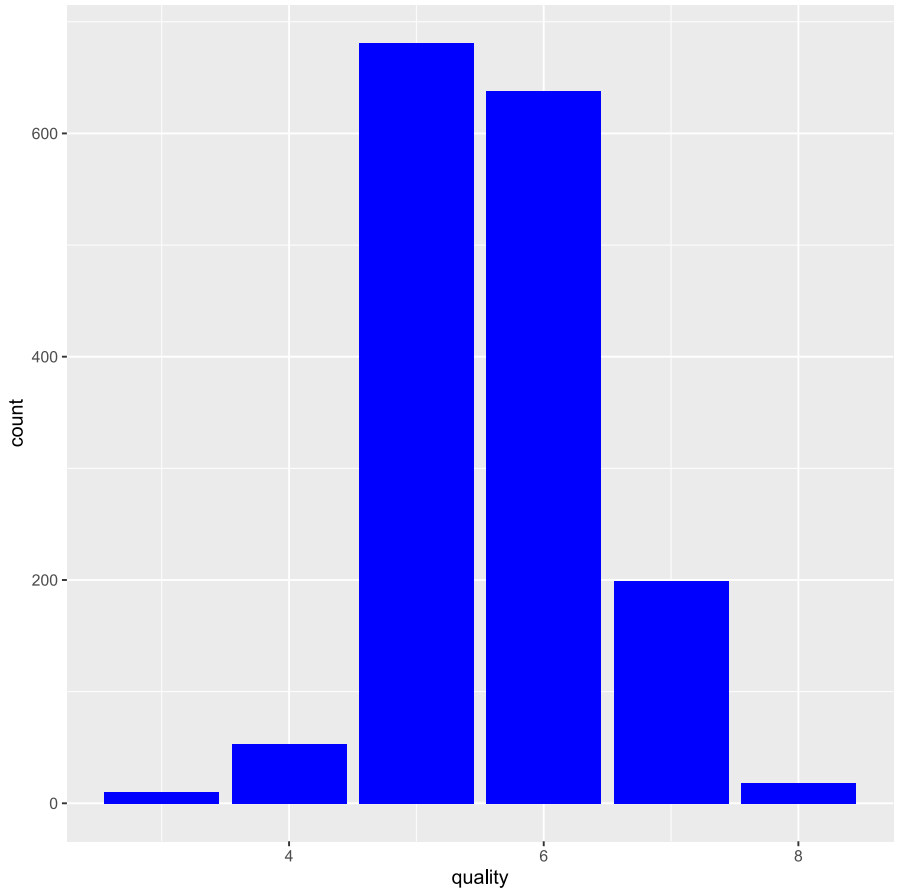
Making a barplot with `ggplot2`

```
ggplot(data=wine)+
  geom_bar(mapping=aes(x = quality))
```



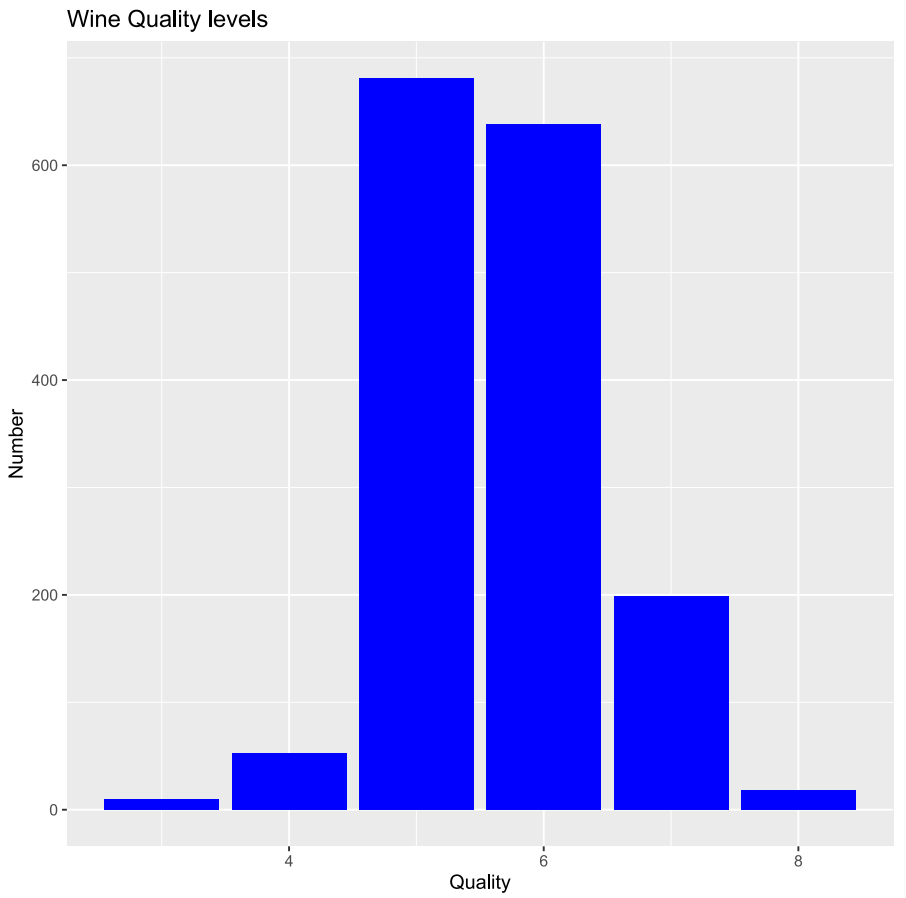
Making a barplot with `ggplot2`

```
ggplot(data=wine)+
  geom_bar(mapping=aes(x = quality),
           fill = 'blue)
```



Making a barplot with `ggplot2`

```
ggplot(data=wine)+
  geom_bar(mapping=aes(x = quality),
           fill = 'blue') +
  labs(title = 'Wine Quality levels',
       x = 'Quality',
       y = 'Number')
```



Task 3: Visualising Data

05:00

Open the `session1_exercises.R` script

1. Make a scatter plot with `residual sugar` on the x-axis and `alcohol` on the y-axis. Title the graph "Residual sugar by alcohol percentage"