DATA2002

Data Analytics: Learning from Data – Notes

Brendan Chow

Student at the University of Sydney
Website
LinkedIn
GitHub

brendanchow@xagill.com

Based on the lecture notes by A/Prof Garth Tarr

Contents

1
Getting started with R

1.1 R packages

- While R by itself is very powerful, we will use many different packages.
- A **package** is a library of functions that add additional functionality.
- Most packages are avaiable from CRAN and can be installed at the R console using `install.packages("package_name")`
- R will also check if you have the required dependencies on your machine and if needed install any additional packages required.
- To use a function from an installed package, you load the package using `library("package_name")` and then you can refer to `function_name()` or you can explicitly use `package_name::function_name()`.

1.2 Palmer penguins

The `penguins` data set was collected and made available by Dr. Kristen Gorman and the PalmerStation, Antarctica LTER, a member of the Long Term Ecological Research Network. It is available in the palmerpenguins package.

```
library(palmerpenguins)
```

To find out more about the `penguins` data set:

```
help(penguins_raw, package = "palmerpenguins")
# or more simply
?penguins
```

A quick look at the data

The `glimpse()` function from the **pillar** package (and re-exported by **dplyr**) gives us a quick overview of the data frame.

```
library(tidyverse)
dplyr::glimpse(penguins_raw)
```

```
Rows: 344
Columns: 17
$ studyName             <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0708", "PAL
    ~
$ `Sample Number`       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
    ~
$ Species               <chr> "Adelie Penguin (Pygoscelis adeliae)", "Adelie P
    ~
$ Region                <chr> "Anvers", "Anvers", "Anvers", "Anvers", "Anvers"
    ~
$ Island                <chr> "Torgersen", "Torgersen", "Torgersen", "Torgerse
    ~
$ Stage                 <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg Stage", "Adu
    ~
$ `Individual ID`       <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1", "N3A2",
    ~
$ `Clutch Completion`   <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No",
    ~
$ `Date Egg`            <date> 2007-11-11, 2007-11-11, 2007-11-16, 2007-11-16,
    ~
```

```
$ `Culmen Length (mm)`  <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34
    ~
$ `Culmen Depth (mm)`   <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18
    ~
$ `Flipper Length (mm)` <dbl> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190,
    ~
$ `Body Mass (g)`       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 34
    ~
$ Sex                   <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMALE", "MALE"
    ~
$ `Delta 15 N (o/oo)`   <dbl> NA, 8.94956, 8.36821, NA, 8.76651, 8.66496, 9.18
    ~
$ `Delta 13 C (o/oo)`   <dbl> NA, -24.69454, -25.33302, NA, -25.32426, -25.298
    ~
$ Comments              <chr> "Not enough blood for isotopes.", NA, NA, "Adult
    ~
```

Cleaning the Palmer penguins column names

The janitor package has a bunch of functions to help with standardising and summarising data frames. Here we standardise the column names: all lower case and no spaces.

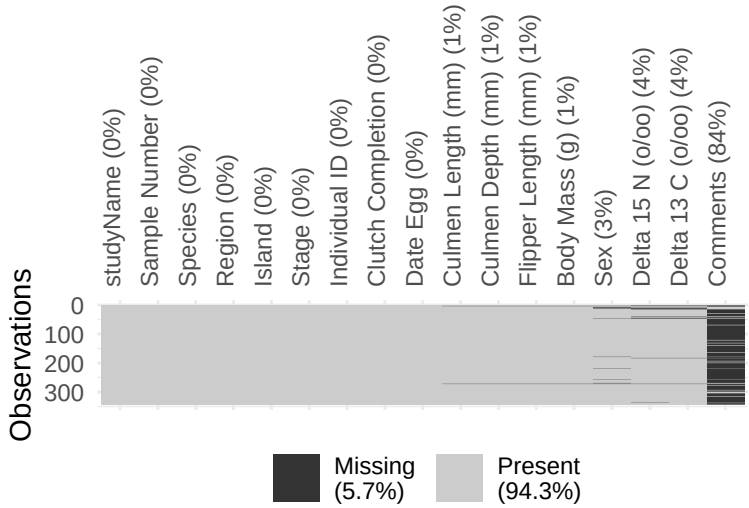
```
penguins_clean = janitor::clean_names(penguins_raw)
dplyr::glimpse(penguins_clean)
```

```
Rows: 344
Columns: 17
$ study_name        <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0708", "PAL0708
    ~
$ sample_number     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1
    ~
$ species           <chr> "Adelie Penguin (Pygoscelis adeliae)", "Adelie Pengu
    ~
$ region            <chr> "Anvers", "Anvers", "Anvers", "Anvers", "Anvers", "A
    ~
$ island            <chr> "Torgersen", "Torgersen", "Torgersen", "Torgersen",
    ~
$ stage             <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg Stage", "Adult,
    ~
$ individual_id     <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1", "N3A2", "N4A
    ~
$ clutch_completion <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", "No"
    ~
$ date_egg          <date> 2007-11-11, 2007-11-11, 2007-11-16, 2007-11-16, 200
    ~
$ culmen_length_mm  <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1,
    ~
$ culmen_depth_mm   <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1,
    ~
$ flipper_length_mm <dbl> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186
    ~
$ body_mass_g       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475,
    ~
$ sex               <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMALE", "MALE", "F
    ~
$ delta_15_n_o_oo   <dbl> NA, 8.94956, 8.36821, NA, 8.76651, 8.66496, 9.18718,
    ~
$ delta_13_c_o_oo   <dbl> NA, -24.69454, -25.33302, NA, -25.32426, -25.29805,
    ~
$ comments          <chr> "Not enough blood for isotopes.", NA, NA, "Adult not
    ~
```

Missing data?

We can use the visdat package to give a visual overview of the missingness in a data frame.

```
library(ggplot2)
visdat::vis_miss(penguins_raw) + theme(axis.text.x = element_text(angle = 90))
```



For simplicity we'll remove any observations (rows) that have missing values.

```
penguins_clean = tidyr::drop_na(penguins_clean, sex)
```

Compare the number of observations between the original and the clean data frame:

```
nrow(penguins)
```

```
[1] 344
```

```
nrow(penguins_clean)
```

```
[1] 333
```

We have made two changes to `penguins_raw` to get to `penguins_clean`. We can summarise this in an easy to read **pipeline**:

```
penguins_clean = penguins_raw |>
  janitor::clean_names() |>
  tidyr::drop_na(sex)
```

Palmer penguins cross tabulation

The **janitor** package has a range of functions that help with importing data and doing quick tabulations

```
library(janitor)
penguins_clean |>
  janitor::tabyl(species, sex) |>
  janitor::adorn_totals(where = c("row", "col"))
```

```
                              species FEMALE MALE Total
      Adelie Penguin (Pygoscelis adeliae)     73   73   146
 Chinstrap penguin (Pygoscelis antarctica)     34   34    68
        Gentoo penguin (Pygoscelis papua)     58   61   119
                                    Total    165  168   333
```

Visualising the Palmer penguins data

We'll use the **ggplot2** package extensively. There are three key components:

- input a data frame
- mapping aesthetics `aes()` where you specify what goes on the axes, how to colour variables, what the groups are, etc.
- geometries `geom_****()` that you add to build up the plot

Mutating a variable

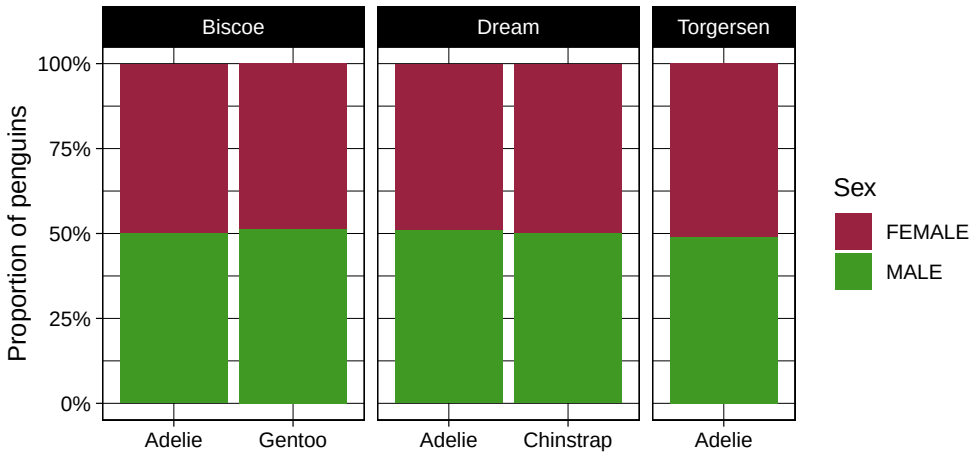
The `species` variable is a bit long. For visualisation purposes we only really need the first word of the species variable. The code below adds to the **pipeline** we saw earlier by creating a new variable, `species_short`, that extracts the first word from the species variable.

```
penguins_clean = penguins_raw |>
  janitor::clean_names() |>
  tidyr::drop_na(sex) |>
  dplyr::mutate(
  species_short = stringr::word(species, start = 1, end = 1)
  )
penguins_clean |>
  dplyr::select(species, species_short) |>
  dplyr::distinct()
```

```
# A tibble: 3 x 2
  species                                 species_short
  <chr>                                   <chr>
1 Adelie Penguin (Pygoscelis adeliae)     Adelie
2 Gentoo penguin (Pygoscelis papua)       Gentoo
3 Chinstrap penguin (Pygoscelis antarctica) Chinstrap
```

- The `ggplot()` function knows about the data frame `penguins`.
- We add (`+`) the bar chart geometry, `geom_bar()`, to our blank canvas.
- Make the bars represent **proportions** within each species:
- Increase the font size and change the theme.
- Tidy up the axis labels.
- What if we want percent on the y-axis?
- Is it the same across all islands?
- Remove empty categories from the x axis.

```
ggplot(data = penguins_clean) +
  aes(x = species_short, fill = sex) +
  geom_bar(position = "fill") +
  theme_linedraw() +
  scale_fill_manual(values=c("#992240", "#409922")) +
  labs(x = "", y = "Proportion of penguins", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  facet_grid(cols = vars(island), scales = "free_x", space = "free_x")
```



2

Collecting data

2.1 Sample surveys

Polling fail

- 1936 US Presidential election
- Franklin D. Roosevelt was completing his term of office
- America was struggling with high unemployment (16%) following the Great Depression
- Literary Digest polled **10 million** people (mail survey)
- 24% response rate (**2.4 million** people reply)
- They had correctly predicted the winner at every election since 1916
- Predicted victory for Landon

Gallup poll

- George Gallup was setting up his survey organisation.
- He drew 3000 people and predicted the Digest results.
- He also drew 50,000 people and correctly predicted Roosevelt victory. The actual prediction was off by a bit: 56% predicted instead of 62%

What went wrong?!?

Why not observe the whole population?

Typical limitations

- Hard to observe the population
- Not enough time
- Not enough money
- Not enough resources

The solution is for us to draw samples and hope or expect to make general statement about the entire population.

Sampling

Definition

Sampling is the process of selecting a subset of representative observations from a population of interest so that characteristics from the subset (sample) can be used to draw conclusion or making inference about the entire population.

Why sample?

- Reduce the number of measurements
- Save time, money and resources
- Might be essential in destructive testing

Sampling procedure

- What sample size is needed for my study?
- How the design will affect the sample size?
- Appropriate **survey design** provides the best estimation with high reliability at the lowest cost with the available resources.
  - What survey design is appropriate for my study?
  - How survey will be conducted/implemented?

Types of biases

- Selection bias
- Recall bias
- Sensitive questions
- Misinterpret the questions
- Wording of question
- Other attributes of the interview as a source of bias…

Getting an opinion

Phrasing 1
Should a woman have control over her own body, including her reproductive system?

Phrasing 2
Should a doctor be allowed to murder unborn children who can't defend themselves?

Measurement bias

Schuman & Converse (1971) performed a study to check whether or not the race of the interviewer inuenced responses after major racial riots in 1968 in Detroit. A sample of 395 African American were asked:

> "Do you personally feel that you can trust most white people, some white people, or none at all"

- White interviewer: 35% responded "most" ($n = 165$)
- African American interviewer: 7% responded "most" ($n = 330$)

Back to the 1936 US election

- The 2.4 million responses didn't even represent the 10 million people who were sent the surveys let alone the general voting population.
- Non-response bias: the people who didn't respond were different to those that did respond.
- Selection bias: addresses sourced from car registration and phone books (skewed towards wealthy Americans).

When a selection procedure is biased, taking a larger sample DOES NOT help. This just repeats the basic mistake at a larger scale.

Bias

Definition

Bias is any factor that favours certain outcomes or responses, or inuences an individual's responses. Bias may be unintentional (accidental), or intentional (to achieve certain results).

When looking at data from a survey think about:

- **Selection bias / sampling bias:** the sample does not accurately represent the population. Example: Attendees at a Star Trek convention may report that their favorite genre is science fiction
- **Non-response bias:** Certain groups are under-represented because they elect not to participate. Example: a restaurant may give each table a "customer satisfaction" survey with their bill.
- **Measurement or designed bias:** Bias factors in the sampling method influence the data obtained. Example: a respondent may answer questions in the way she thinks the questioner wants her to answer.

2.2 Controlled experiments

Randomised controlled double-blind trials

What is a randomised controlled double-blind study? Why is it good but rare?

1. Investigators obtain a representative sample of subjects.
2. Investigators randomly allocate the subjects into a **treatment** group and a **control** group.

3. The **control group** is given a placebo, but neither the subjects nor the investigators know the identity of the 2 groups (double-blind).
4. Investigators compare the responses of the 2 groups.
5. The design is good because we expect the 2 groups to be similar, hence any difference in the responses is likely to be caused by the treatment

2.3 Observational studies

Does smoking cause cancer?

> "Tobacco smoking is the largest preventable cause of cancer, responsible for more cancer deaths in Australia than any other single factor. It is also directly responsible for many heart and lung diseases." – Australian Cancer Council

> In a hearing to the Australian High Court in 2012 disputing the introduction of cigarette plain packaging with health warnings, while British American Tobacco was prepared to accept that there are serious health consequences caused by smoking, Imperial Tobacco responded **some people say that...**
> – The Sydney Morning Herald

The need for observational studies

- By necessity, many research questions require an **observational study**, rather than a controlled experiment.
- For example, with a study on the effects of smoking, investigators cannot choose which subjects will be in the **treatment group** (smoking). Rather, they must **observe** medical results for the 2 groups.
- Smilarly, most **educational research** is based on observational studies.
- The conclusions of observational studies require great care.

Observational studies can not establish causation.

- A good **randomised controlled experiment** can establish causation, an **observational study** can only establish association.
- An observational study may *suggest* causation, but it can't *prove* causation.

Misleading hidden confounders

- Confounding occurs when the **treatment group** and **control group** differ by some third variable (other than the treatment) which influences the response that is studied.
- Confounders can be hard to find, and can mislead about a cause and effect relationship.
- Confounding (or lurking) variables can be introduced into a randomised study if any of the subjects drop out, causing **selection bias** or **survivor bias**. Similarly, if not all subjects keep taking the treatment or placebo, we get the confounding of **adherers** and **non-adherers**.

Lung cancer associations

A study finds that having yellow fingertips is associated with lung cancer. Does having yellow fingertips cause lung cancer?

A study finds that smokers tend to have higher rates of lung cancer. Does smoking cause lung cancer?

Strategy for dealing with confounders

Sometimes we can make the groups more comparable by dividing them into subgroups with respect to the confounder. For example, if alcohol consumption is a potential confounding factor for smoking's affect on liver cancer, we can divide our subjects into 3 groups:

- heavy drinkers
- medium drinkers
- light drinkers

This is called **controlling** for alcohol consumption.

Controlling for confounding

We can control for confound by making 3 separate comparisons:

- heavy drinking: smokers vs non-smokers
- medium drinking: smokers vs non-smokers
- light drinking: smokers vs non-smokers

What are the limitations of this strategy?

Is smoking good for your longevity?

A famous study by Appleton et al. (1996) considered data on female subjects 20 years apart. Two studies:

- initial data from a 1 in 6 survey from an electoral roll in a mixed urban and rural area near Newcastle upon Tyne UK Tunbridge et al. (1977).
- follow-up data 20 years later Vanderpump et al. (1995).

The study concentrated on the 1314 women who were either smokers or non- smokers (in the full data, only 162 had stopped smoking and only 18 did not record their status).

```
x = read_csv("https://git.io/fNGXk")
x
```

```
# A tibble: 4 x 3
  status     survival count
  <chr>      <chr>    <dbl>
1 Smoker     Died       139
2 Smoker     Survived   443
3 Non-smoker Died       230
4 Non-smoker Survived   502
```

```
x_long = tidyr::uncount(x, weights = count)
dim(x_long)
```

```
[1] 1314    2
```

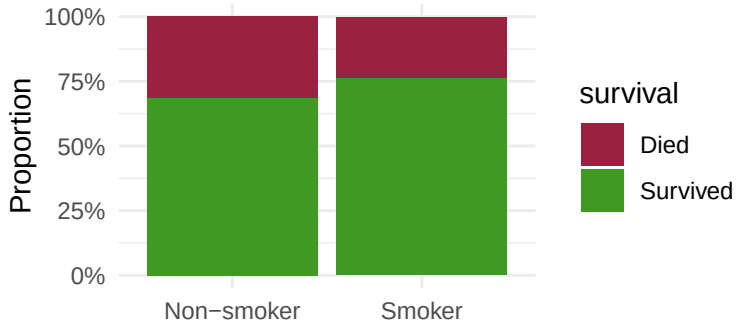
```
x_long |>
  dplyr::group_by(status) |>
  dplyr::summarise(
                rate = sum(survival == "Died")/n()
        )
```

```
# A tibble: 2 x 2
  status      rate
  <chr>      <dbl>
1 Non-smoker 0.314
2 Smoker     0.239
```

```
x_long |>
  dplyr::summarise(
                rate = sum(survival == "Died")/n()
  )
```

```
# A tibble: 1 x 1
   rate
  <dbl>
1 0.281
```

```
ggplot(x) +
  aes(x = status,
  y = count,
  fill = survival) +
  geom_bar(stat = "identity",
  position = "fill") +
  scale_y_continuous(
  labels = scales::percent_format()) +
  labs(x = "", y = "Proportion") +
  theme_minimal() +
  scale_fill_manual(values=c("#992240", "#409922"))
```



2.4 Simpson's Paradox

What is Simpson's Paradox?

- Simpson's Paradox was first mentioned by British statistician Udny Yule in 1903.
- It is named after Edward H. Simpson (Simpson, 1951)
- Sometimes there is a clear trend in individual groups of data that disappears when the groups are pooled together.
- It occurs when relationship between percentages in subgroups are reversed when the subgroups are combined, because of a confounding or lurking variable.
- The association between a pair of variables (X, Y) reverses sign upon conditioning of a third variable Z, regardless of the value taken by Z.

> Observational studies with a confounding variable can lead to **Simpson's paradox**

Mortality by Age Group

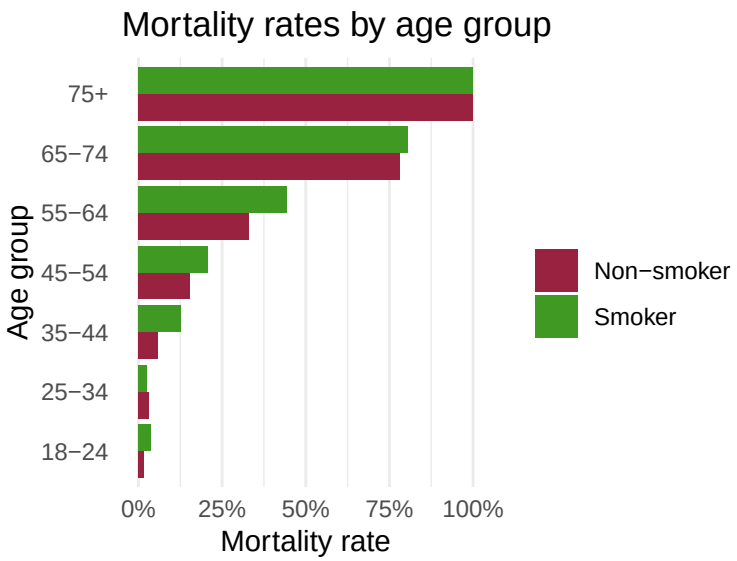
```
y = readr::read_csv("https://git.io/fNGSi")
dplyr::glimpse(y, width = 80)
```

```
Rows: 28
Columns: 4
$ status    <chr> "Smoker", "Smoker", "Smoker", "Smoker", "Smoker", "Smoker",
    ~
$ survival  <chr> "Died", "Died", "Died", "Died", "Died", "Died", "Died", "Sur
    ~
$ age_group <chr> "18-24", "25-34", "35-44", "45-54", "55-64", "65-74", "75+",
    ~
$ count     <dbl> 2, 3, 14, 27, 51, 29, 13, 53, 121, 95, 103, 64, 7, 0, 1, 5,
    ~
```

```
ytab = y |>
  tidyr::pivot_wider(id_cols = age_group,
                     names_from = c(status, survival),
                     values_from = count,
                     names_sep = ": ")
ytab |>
  knitr::kable(booktabs = TRUE) |>
  kable_styling(position="center", latex_options = "hold_position")
```

age_group	Smoker: Died	Smoker: Survived	Non-smoker: Died	Non-smoker: Survived
18-24	2	53	1	61
25-34	3	121	5	152
35-44	14	95	7	114
45-54	27	103	12	66
55-64	51	64	40	81
65-74	29	7	101	28
75+	13	0	64	0

```
mortality = y |>
  uncount(weights = count) |>
  group_by(status, age_group) |>
  summarise(rate = mean(survival=="Died"))
mortality |>
  ggplot() +
  aes(x = age_group, y = rate, fill = status) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  scale_fill_manual(values=c("#992240", "#409922")) +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "Mortality rates by age group",
           y = "Mortality rate",
           x = "Age group",
  fill = "") +
  theme(panel.grid.major.y = element_blank()) +
  coord_flip()
```

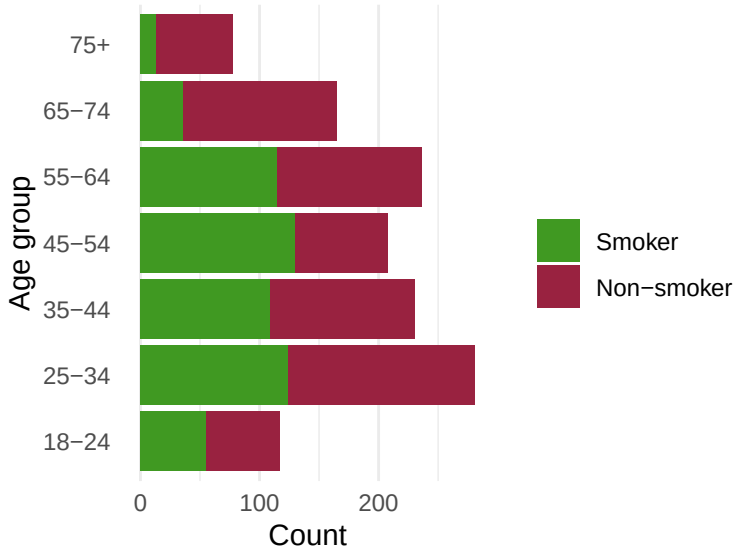


- Not many young people died.
- Most old people died.
- In the middle age groups, smokers tended to have higher mortality rates than non-smokers

How did we got the wrong overall conclusion?

Consider the distribution of samples by smoking status across age groups.

```
# convert into "long" form
mortality_long = y |> uncount(weights = count)
mortality_long |>
  ggplot() +
  aes(x = age_group, fill = status) +
  geom_bar() +
  theme_minimal() +
  scale_fill_manual(values=c("#992240", "#409922")) +
  labs(y = "Count",
           x = "Age group",
  fill = "") +
  theme(panel.grid.major.y = element_blank(),) +
  guides(fill = guide_legend(reverse = TRUE)) +
  coord_flip()
```



- As there are many more young women who smoked than older women, and as younger women are expected to live longer than older women, adding all the groups together makes smoking appear to be beneficial.
- This is a classic example of **Simpson's paradox**: a trend present within multiple groups can reverse when the groups are combined.

3

Chi-squared tests

3.1 Genetic linkage

In a backcross experiment to investigate the **genetic linkage** between two genes A and B in a species of flower. Researchers classified 400 offspring by phenotype:

Phenotype	AB	ab	aB	ab
Count	128	86	74	112

- A might be pink flowers and a might be yellow flowers
- B might be smooth leaves and b might be wrinkled leaves
  - AB means a plant with pink flowers and smooth leaves
  - ab means a plant with pink flowers and wrinkled leaves
  - aB means a plant with yellow flowers and smooth leaves
  - ab means a plant with yellow flowers and wrinkled leaves

Under the *no linkage* model, the four phenotypes are equally likely. So we would expect to see equal proportions for each phenotype:

Phenotype	AB	ab	aB	ab
Expected proportion	0.25	0.25	0.25	0.25

If linkage is in the *coupling phase*, the probabilities of the four phenotypes are given by

Phenotype	AB	ab	aB	ab
Expected proportion	$\frac{1}{2}(1-p)$	$\frac{1}{2}p$	$\frac{1}{2}p$	$\frac{1}{2}(1-p)$

where p is the **recombination fraction**. We will need to estimate p as the overall proportion of observed Ab and aB

3.2 No linkage model

No linkage model

- **Null hypothesis**: each of the phenotypes are equally likely.
- **Alternative hypothesis**: the phenotypes are not equally likely.

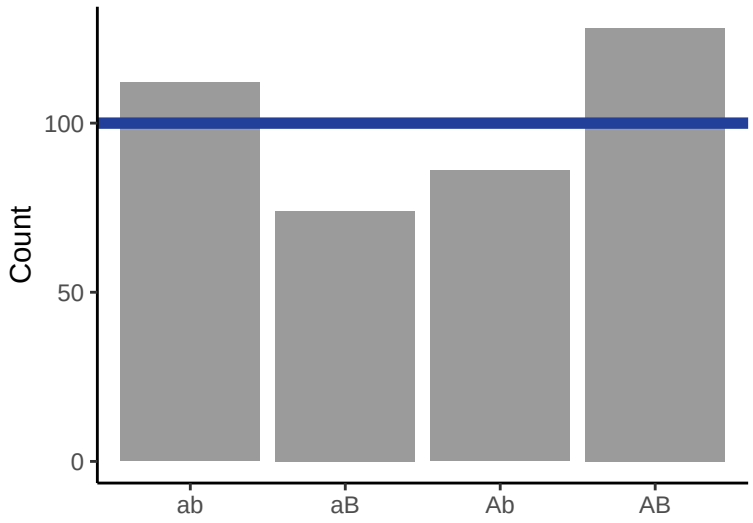
Let p_i be the probability of being in the ith phenotype $i = AB, Ab, aB, ab$.
Under the null hypothesis (i.e. assuming that the no linkage model is correct) the counts are uniformly distributed across the 4 categories,

$$p_i = 0.25 \text{ for all } i$$

```
# observed counts
y = c(128, 86, 74, 112)
n = sum(y)
# hypothesised proportions
p = c(1/4, 1/4, 1/4, 1/4)
# expected counts
e = p*n
names = c("AB", "Ab", "aB", "ab")
df = tibble(
  phenotype = names,
  # observed counts
  y = y,
  # hypothesised proportions
  p = p,
  # expected counts
  e = n * p
)
df
```

```
# A tibble: 4 x 4
  phenotype     y     p     e
  <chr>     <dbl> <dbl> <dbl>
1 AB          128  0.25   100
2 Ab           86  0.25   100
3 aB           74  0.25   100
4 ab          112  0.25   100
```

```
df |> ggplot() +
  aes(x = phenotype, y = y) +
  geom_col(alpha = 0.6) +
  geom_hline(yintercept = 100,
  colour = "#224099",
  linewidth = 2) +
  labs(x = "", y = "Count") +
  theme_classic()
```



Differences between observed counts and expected counts:

```
df = df |>
  mutate(d = y - e)
df
```

```
# A tibble: 4 x 5
  phenotype     y     p     e     d
  <chr>     <dbl> <dbl> <dbl> <dbl>
1 AB          128  0.25   100    28
2 Ab           86  0.25   100   -14
3 aB           74  0.25   100   -26
4 ab          112  0.25   100    12
```

```
df |>
        summarise(avg_diff = mean(d))
```

```
# A tibble: 1 x 1
  avg_diff
     <dbl>
1        0
```

Test statistic

The average of the differences doesn't tell us much. Let's take the squared differences, and "normalise" by dividing by the expected cell counts:

$$t_0 = \sum_{i=1}^{k} \frac{(y_i - e_i)^2}{e_i}$$

where k is the number of categories (groups).

```
df = df |> mutate(
  squared_discrepency = (y-e)^2,
  contribution = (y-e)^2/e
  )
t0 = sum(df$contribution)
t0
```

```
[1] 18
```

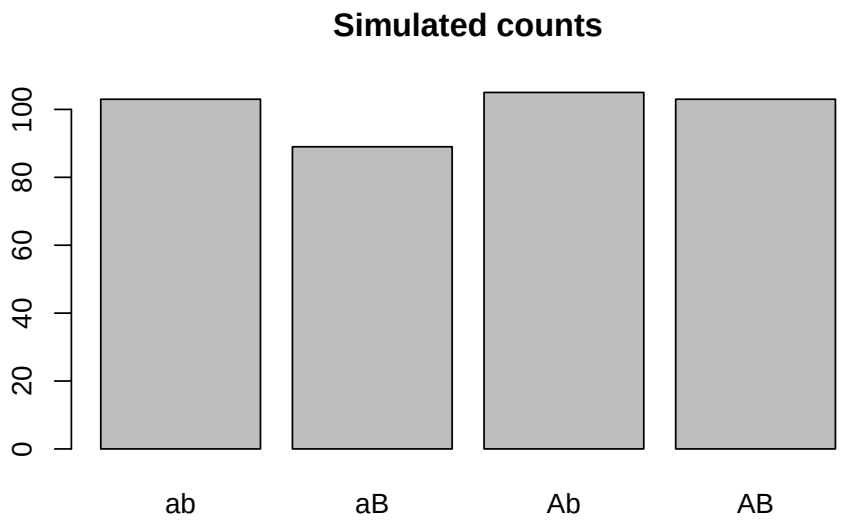
Simulate

Under the null hypothesis, the counts are uniformly distributed across the 4 categories. Fixing the sample size at $n = 400$ we can **simulate** data assuming the null hypothesis is true.

```
n = 400
set.seed(88)
sim1 = sample(
  x = names,
  size = n,
  replace = TRUE,
  prob = c(0.25,0.25,0.25,0.25)
)
table(sim1)
```

```
sim1
 ab  aB  Ab  AB
103  89 105 103
```

```
barplot(table(sim1), main = "Simulated counts")
```



Our test statistic for that simulated sample is:

```
sim_y = table(sim1)
sum((sim_y - e)^2/e)
```

```
[1] 1.64
```

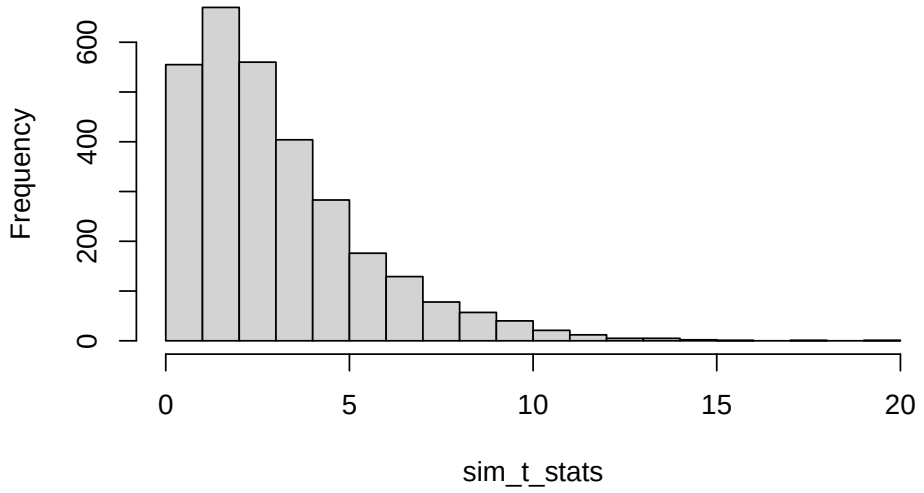
which is alot smaller than what we observed on our **actual data**:

```
t0
```

```
[1] 18
```

```
B = 3000
sim_t_stats = vector(mode = "numeric", length = B)
for(i in 1:B){
  sim = sample(x = names, size = n,
  replace = TRUE, prob = c(0.25,0.25,0.25,0.25))
  sim_y = table(sim)
```

```
  sim_t_stats[i] = sum((sim_y - e)^2/e)
}
hist(sim_t_stats, main = "", breaks = 20)
```



- Now we have a pretty good idea about the shape of the **distribution** of the test statistic when the null hypothesis is true.
- We can compare the test statistic that we calculated on the original data to the "null distribution"
- One way to do this is to ask the question:
  Given that the null hypothesis is true, how likely is it that we observe a test statistic as or more extreme than that we calculated from our original sample?

```
# sum(sim_t_stats >= t0)/B
mean(sim_t_stats >= t0)
```

```
[1] 0.0003333333
```

In 0.1% of samples when the null hypothesis is true, we got a simulated sample that was "more extreme" than our original sample.

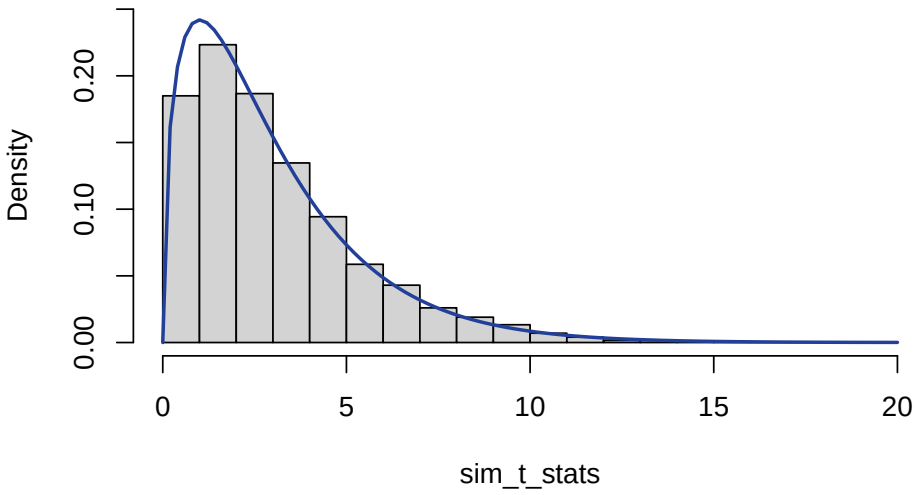
Is there way to do it without simulation?

Yes! A χ^2 test!
In this example the test statistic,

$$T = \sum_{i=1}^{k} \frac{(Y_i - e_i)^2}{e_i} \sim \chi^2_{k-1},$$

approximately, where k is the number of groups.
Let's compare this distribution to the simulated test statistic distribution.

```
hist(sim_t_stats, main = "", breaks = 20, probability = TRUE, ylim = c(0, 0.25
    ))
curve(dchisq(x, df = 3), add = TRUE, col = "#224099", lwd = 2)
```



χ^2 test degrees of freedom

- The **degrees of freedom** is $k-1$ because the first $k-1$ observations y_i contain all the information and the last observation is fixed by $y_k = n - \sum_{i=1}^{k-1} y_i$ adding no extra information.
- In general, the test statistic takes the form,

$$T = \sum_{i=1}^{k} \frac{(Y_{i-e_i})^2}{e_i} \sim \chi^2_{k-1-q}$$

  where q is the number of parameters that need to be estimated from the sample.
- In the no linkage example, $q = 0$ because we do not need to estimate any parameters (i.e. we don't need to estimate the hypothesised proportions, all $p_{i0} = 0.25$).
- The approximation will only be accurate if *no expected frequency* is too small, as a rule of thumb we require all $e_i \geqslant 5$. Otherwise, we need to pool adjacent categories so that the expected frequencies are always $\geqslant 5$.

Workflow: Chi-squared goodness of fit test

One categorical variable from a single population and want to see if it follows a hypothesised distribution.

- **Hypothesis**: H_0: the proportions in each category (p_i) are equal to the corresponding hypothesised proportions (p_{i0}), i.e. $p_1 = p_{10}, p_2 = p_{20}, \dots, p_k = p_{k0}$ vs H_1: at least proportion is not equal to the hypothesised proportion, i.e. at least one equality does not hold.
- **Assumptions**: independent observations and $e_i = np_{i0} \geqslant 5$
- **Test statistic**: $T = \sum_{i=1}^{k} \frac{(Y_i - e_i)^2}{e_i}$. Under $H_0, T \sim \chi^2_{k-1-q}$ approximately, where k is the number of groups and q is the number of parameters that needs to be estimated from the data.
- **Observed test statistic**: $\sum_{i=1}^{k} \frac{(Y_{i-e_i})^2}{e_i}$.

- **P-value**: $P(T \geqslant t_0) = P(\chi^2_{k-1-q} \geqslant t_0)$
- **Decision**: Reject H_0 if the p-value $< \alpha$, otherwise do not reject H_0.

Table for calculating the test statistic

The calculations can be summarised in the following table:

Group i	y_i	p_{i0}	$e_i = np_{i0}$	$y_i - e_i$	$\frac{(y_i-e_i)^2}{e_i}$
1	y_1	p_{10}	np_{10}	$y_1 - e_1$	$(y_1 - e_1)^2/e_1$
2	y_2	p_{20}	np_{20}	$y_2 - e_2$	$(y_2 - e_2)^2/e_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	y_k	p_{k0}	np_{k0}	$y_k - e_k$	$(y_k - e_k)^2/e_k$
Sum	n	1	n	0	t_0

- y_i are the observed counts
- p_{i0} are the hypothesised probabilities
- e_i are the expected counts *assuming the null hypothesis is true*
- t_0 is the observed test statistic

No linkage model

Under the *no linkage model* we assume that the observations are uniformly distributed across the four categories, i.e. the hypothesised probabilities are $p_{i0} = \frac{1}{4}$ for $i = 1, 2, 3, 4$:

- **Hypothesis**: The null hypothesis is a no linkage model with uniform probabilities,H_0 : $p_{AB} = p_{Ab} = p_{aB} = p_{ab} = \frac{1}{4}$. The alternative is anything other than a no linkage model, H_1: at least one equality does not hold.
- **Assumptions**: independent observations and expected cell counts at least 5, $e_i = np_{i0} \geqslant 5$
- **Test statistic**: $T = \sum_{i=1}^{k} \frac{(Y_i-e_i)^2}{e_i}$. Under $H_0, T \sim \chi^2_3$ approximately.
- **Observed test statistic**: $t_0 = 18$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_3 \geqslant 18) = 0.0004$
- **Decision**: Since the p-value is much smaller than 0.05, there is strong evidence in the data against H_0. Hence the four phenotypes are not equally likely and we reject the no linkage model.

Type	i	y_i	$e_i = np_{i0}$	$y_i - e_i$	$\frac{(y_i-e_i)^2}{e_i}$
AB	1	128	$400 \times \frac{1}{4} = 100$	$128 - 100 = 28$	$\frac{(28)^2}{100} = 7.84$
Ab	2	86	$400 \times \frac{1}{4} = 100$	$86 - 100 = -14$	$\frac{(-14)^2}{100} = 1.96$
aB	3	74	$400 \times \frac{1}{4} = 100$	$74 - 100 = -26$	$\frac{(-26)^2}{100} = 6.76$
ab	4	112	$400 \times \frac{1}{4} = 100$	$112 - 100 = 12$	$\frac{(12)^2}{100} = 1.44$
Total		400	$400 \times \frac{1}{4} = 100$	0	$t_0 = 18.00$

```
1 - pchisq(18, df = 3)
```

```
[1] 0.0004398497
```

```
(ey=n*p) # expected counts
```

```
[1] 100 100 100 100
```

```
ey >= 5 # test e_i >= 5
```

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

The `chisq.test()` function in R can do it all for us. We give it the vector of observed counts and the vector of hypothesised probabilities:

```
chisq.test(y, p = p)
```

```
        Chi-squared test for given probabilities

data:  y
X-squared = 18, df = 3, p-value = 0.0004398
```

3.3 Linkage model

Under the *coupling phase* linkage model, the probabilities of each of the four phenotype outcomes are given by

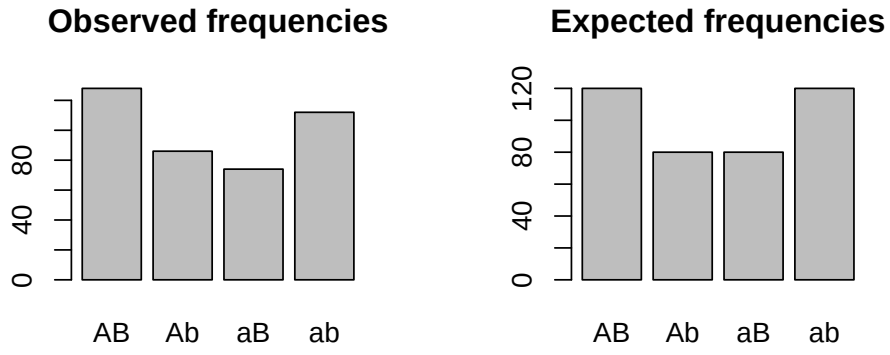
Phenotype	AB	ab	aB	ab
Observed count	128	86	74	112
Expected proportion	$\frac{1}{2}(1-p)$	$\frac{1}{2}p$	$\frac{1}{2}p$	$\frac{1}{2}(1-p)$

To test the linkage model hypothesis, we need to estimate the parameter p, the recombination fraction from the observed data: $\hat{p}$ is the proportion of observed offspring with phenotype Ab or aB,

$$\hat{p} = \frac{86 + 74}{400} = 0.4$$

Hence the four (estimated) hypothesised probabilities are,

```
y = c(128, 86, 74, 112)
p = c(0.3, 0.2, 0.2, 0.3)
names = c("AB", "Ab", "aB", "ab")
par(mfrow = c(1, 2)) # set up a graphics device with 1 row and 2 columns
barplot(y, names.arg = names, main = 'Observed frequencies')
barplot(p * sum(y), names.arg = names, main = 'Expected frequencies')
```



Test statistic

The form of the observed test statistic is the same as in the no linkage model,

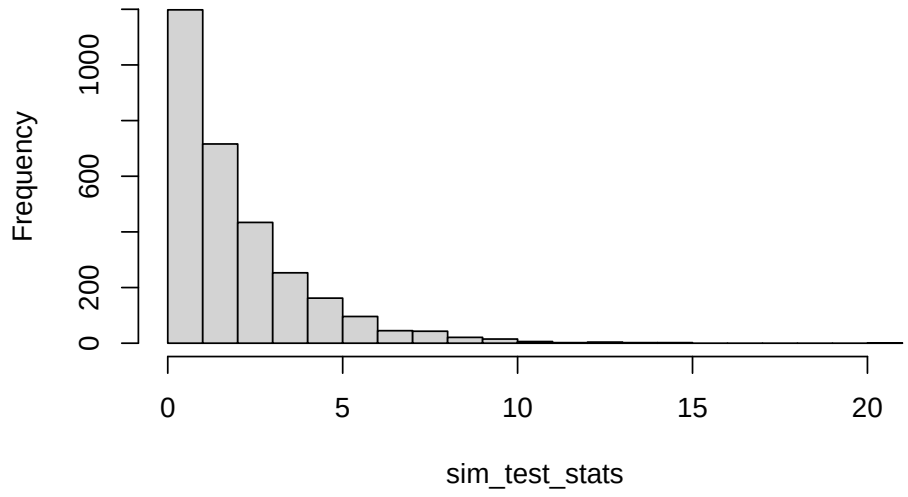
$$t_0 = \sum_{i=1}^{4} \frac{(y_i - e_i)^2}{e_i}$$

but now the e_i are now calculated using the new hypothesised proportions.

Type	y_i	$e_i = np_{i0}$	$y_i - e_i$	$\frac{(y_i-e_i)^2}{e_i}$
AB	128	$400 \times \frac{3}{4} = 120$	$128 - 120 = 8$	$\frac{(8)^2}{120} = 7.84$
Ab	86	$400 \times \frac{2}{4} = 80$	$86 - 80 = 6$	$\frac{(6)^2}{80} = 1.96$
aB	74	$400 \times \frac{2}{4} = 80$	$74 - 80 = -6$	$\frac{(-6)^2}{80} = 6.76$
ab	112	$400 \times \frac{3}{4} = 120$	$112 - 120 = -8$	$\frac{(-8)^2}{120} = 1.44$
Total	400	400	0	$t_0 = 1.96$

Simulated null distribution

```
n = 400
hyp_probs = c(0.3, 0.2, 0.2, 0.3)
B = 3000
sim_test_stats = vector(mode = "numeric", length = B)
for(i in 1:B){
  sim = sample(x = names, size = n, replace = TRUE, prob = hyp_probs)
  sim_y = table(sim)
  # estimated probability
  p_e = sum(table(sim)[2:3])/n
  # expected values using estimated probabilities
  e = 400*c(1 - p_e, p_e, p_e, 1 - p_e)/2
  sim_test_stats[i] = sum((sim_y - e)^2/e)
}
hist(sim_test_stats, main = "", breaks = 20)
```



> We re-estimate the recombination fraction and expected cell counts *within* the loop replicating what we did to calculate the original test statistic. This is important because it changes the shape of the null distribution. Try for yourself to see what happens when the hypothesised proportions and expectations are fixed.

Significance

Let's compare this with the distribution of test statistics that we simulated assuming the null hypothesis is true.

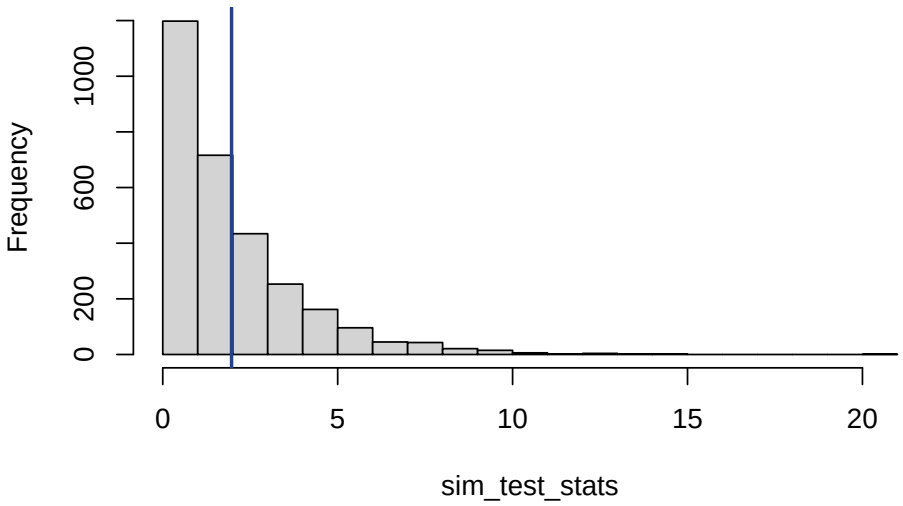
```
n = 400
hyp_probs = c(0.3, 0.2, 0.2, 0.3)
expected_counts  = hyp_probs * n
t0 = sum((y-expected_counts)^2/expected_counts)
t0
```

```
[1] 1.966667
```

```
mean(sim_test_stats >= t0)
```

```
[1] 0.3683333
```

```
hist(sim_test_stats, main = "", breaks = 20)
abline(v = t0, col = "#224099", lwd = 2)
```



- **Hypothesis**: The null hypothesis is a coupling linkage model,$H_0 : p_{AB} = p_{ab} = \frac{1-p}{2}$ and $p_{Ab} = p_{aB} = \frac{p}{2}$. The alternative that the proportions do not follow a coupling phase linkage model, i.e H_1 : at least one equality does not hold.
- **Assumptions**: independent observations and expected cell counts at least 5, $e_i = np_{i0} \geqslant 5$
- **Test statistic**: $T = \sum_{i=1}^{k} \frac{(Y_i - e_i)^2}{e_i}$. Under $H_0, T \sim \chi^2_2$ approximately.
- **Observed test statistic**: $t_0 = 18$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_2 \geqslant 1.97) = 0.37$
- **Decision**: Since the p-value is much smaller than 0.05, so we do not reject the null hypothesis and conclude that the data are consistent with the "coupling phase" linkage model.

In R

```
chisq.test(y, p = p)
```

```
        Chi-squared test for given probabilities

data:  y
X-squared = 1.9667, df = 3, p-value = 0.5794
```

> **Note the incorrect degrees of freedom!**

```
n = sum(y)
k = length(y)
(ey = n*p)
```

```
[1] 120  80  80 120
```

```
ey >= 5 # check e_i >= 5
```

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

```
(t0=sum((y-ey)^2/ey))
```

```
[1] 1.966667
```

```
(pval=1 - pchisq(t0, df = k - 1 - 1))
```

```
[1] 0.3740621
```

4

Goodness of fit tests for discrete distributions

Radiation exposure

The goal in biological dosimetry is to estimate the dose of **ionizing radiation**, absorbed by an exposed individual by using chromosome damage in peripheral lymphocytes. When radiation exposure occurs, the damage in DNA is randomly distributed between cells producing chromosome aberrations. The outcome of interest is the number of aberrations observed. The number of aberrations typically follows a Poisson distribution, the rate of which depends on the dose.
The table below shows the number of chromosome aberrations from a patient exposed to radiation after the nuclear accident of Stamboliyski (Bulgaria) in 2011

Number of aberrations	0	1	2	3	4	5	6	7
Frequency	117	94	51	15	0	0	0	1

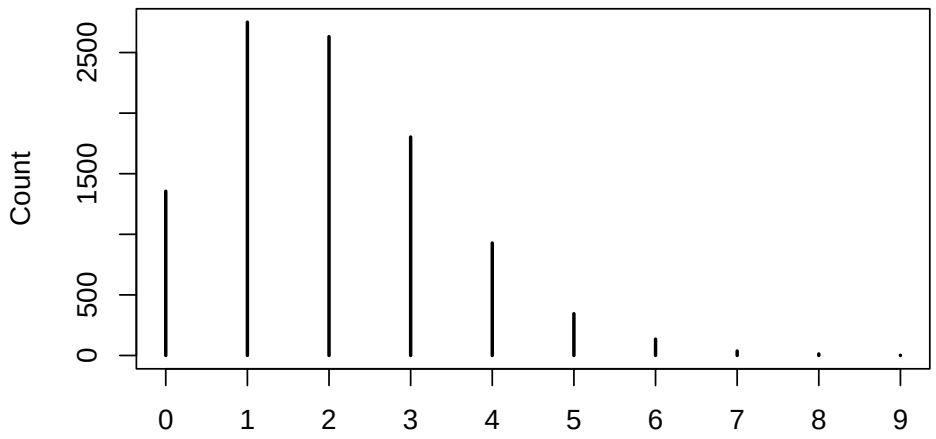
How can we check if the random variable generating this data follows a Poisson distribution.

Poisson distribution

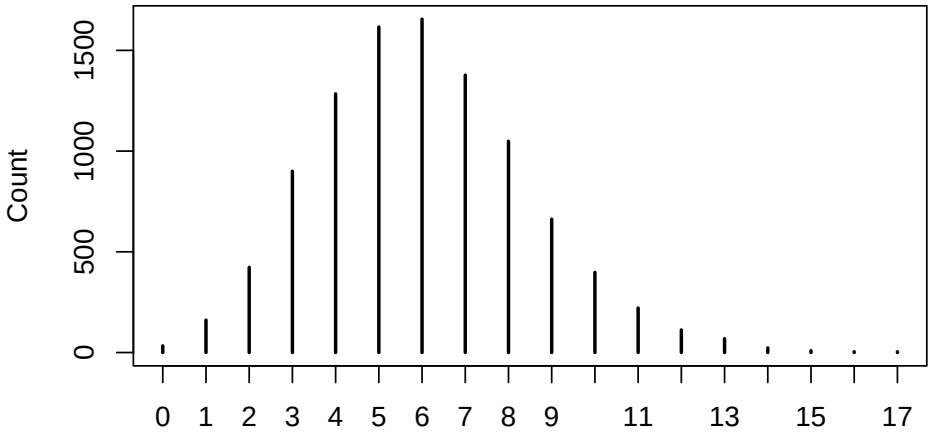
A **Poisson** random variable represents the probability of a given number of events occurring in a fixed interval (e.g. number of events in a fixed period of time) if these event occur independently with some known average rate λ per unit of time.
If X is a Posson random variable with rate parameter λ, the probability mass function is:

$$P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}, \quad k = 0, 1, 2, \dots,$$

```
set.seed(8)
rpois(n=10000, lambda = 2) |>
  table() |>
plot(ylab = "Count")
```




```
rpois(n=10000, lambda = 6) |>
  table() |>
plot(ylab = "Count")
```



Chi-squared tests for discrete distributions

Suppose we have a sample of observations $x_1, x_2, \dots, x_n$.
We want to test whether the sample is taken from a population with a given distribution function $F_0(x \mid \theta_1, \theta_2, \dots, \theta_h)$ where θ_l are parameters of the distribution.
We summarise the observed data by tabulating the observed frequencies y_i for each possible outcome and compare them to the corresponding expected frequencies, e_i calculate using the expected probabilities, p_i, from the hypothesised distribution, $F_0(x \mid \theta_1, \theta_2, \dots, \theta_h)$.
This is a *general* chi-squared goodness-of-fit test with test statistic,

$$T = \sum_{i=1}^{k} \frac{(Y_i - e_i)^2}{e_i} = \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i},$$

where $i = 1, 2, \dots, k$ iterates over the distinct outcomes.
However the model parameters $\theta_1, \theta_2, \dots, \theta_h$ are usually **unknown** and have to be estimated from the sample.
In theis case, the expected probabilities p_i are replaced by their estimates $\hat{p}_i$.
Then the obseved test statistic is

$$t_0 = \sum_{i=1}^{k} \frac{(y_i - n\hat{p}_i)^2}{n\hat{p}_i}$$

and the approximate p-value is

$$P(\chi^2_{k-1-q} \geqslant t_0).$$

Note the degrees of freedom are $k - 1 - q$ where q is the number of parameters we need to estimate.

Radiation exposure

- Let X be a random variable such that $X \sim$ Poisson (λ).

- Let y_i be the observed frequency of outcome i.
- The expected probabilities p_i are given by the probability mass function,

$$P(X = i) = p_i = e^{-\lambda}\frac{\lambda^i}{i!} \quad \text{for } i = 0, 1, 2, \dots,$$

  where p_i denote the probability of the number of chromosome aberrations in the ith category.
- Note that for a Posson distribution both $\mathrm{E}(X)$ and $\mathrm{Var}(X)$ are equal to the parameter λ.
- We need to estimate λ by the sample mean: $\hat{\lambda} = \overline{x} = 248/278 = 0.89$.

i	y_i	$\hat{p}_i = e^{-\hat{\lambda}}\hat{\lambda}^i/i!$	$\hat{e}_i = n\hat{p}_i$	$(y_i - \hat{e}_i)^2/\hat{e}_i$
0	117	$\frac{0.89^0 e^{-0.89}}{0!} = 0.4098$	$278 \times 0.4098 = 113.92$	$\frac{(117 - 113.92)^2}{113.92} = 0.08$
1	94	$\frac{0.89^1 e^{-0.89}}{1!} = 0.3656$	$278 \times 0.3656 = 101.63$	$\frac{(94 - 101.63)^2}{101.63} = 0.57$
2	51	$\frac{0.89^2 e^{-0.89}}{2!} = 0.1631$	$278 \times 0.1631 = 45.33$	$\frac{(51 - 45.33)^2}{45.33} = 0.71$
3	15	$\frac{0.89^3 e^{-0.89}}{3!} = 0.0485$	$278 \times 0.0485 = 13.48$	$\frac{(15 - 13.48)^2}{13.48} = 0.17$
4	0	$\frac{0.89^4 e^{-0.89}}{4!} = 0.0108$	$278 \times 0.0108 = 3.01$	$\frac{(0 - 3.01)^2}{3.01} = 3.01$
5	0	$\frac{0.89^5 e^{-0.89}}{5!} = 0.0019$	$278 \times 0.0019 = 0.54$	$\frac{(0 - 0.54)^2}{0.54} = 0.54$
6	0	$\frac{0.89^6 e^{-0.89}}{6!} = 0.0003$	$278 \times 0.0003 = 0.08$	$\frac{(0 - 0.08)^2}{0.08} = 0.08$
$\geqslant 7$	1	0.00004	$278 \times 0.00004 = 0.01$	$\frac{(1 - 0.01)^2}{0.01} = 96.40$
Total	278	1	278	101.56

- But wait! There are a number of cells where the expected number of counts is less than 5 which violates an assumption.
- We combine the last five classes so that the final group is $\geqslant 3$ with observed frequency $y_{\geqslant 3} = 15 + 1 = 16$, the expected count is $\hat{e}_{\geqslant 3} = 13.48 + 3.01 + 0.54 + 0.08 + 0.01 = 17.12$:

i	y_i	$\hat{p}_i = e^{-\hat{\lambda}}\hat{\lambda}^i/i!$	$\hat{e}_i = n\hat{p}_i$	$(y_i - \hat{e}_i)^2/\hat{e}_i$
0	117	$\frac{0.89^0 e^{-0.89}}{0!} = 0.4098$	$278 \times 0.4098 = 113.92$	$\frac{(117 - 113.92)^2}{113.92} = 0.08$
1	94	$\frac{0.89^1 e^{-0.89}}{1!} = 0.3656$	$278 \times 0.3656 = 101.63$	$\frac{(94 - 101.63)^2}{101.63} = 0.57$
2	51	$\frac{0.89^2 e^{-0.89}}{2!} = 0.1631$	$278 \times 0.1631 = 45.33$	$\frac{(51 - 45.33)^2}{45.33} = 0.71$
$\geqslant 3$	16	$1 - 0.4098 - 0.3656 - 0.1631 = 0.0615$	$278 \times 0.015 = 17.12$	$\frac{(16 - 17.12)^2}{17.12} = 0.07$
Total	278	1	278	1.43

- The final observed test statistic is, $t_0 = 0.08 + 0.57 + 0.71 + 0.07 = 1.43$

- **Hypothesis**: H_0: the data follow a Poisson distribution vs H_1: the data do not follow from a Possion distribution.
- **Assumptions**: The expected frequencies, $e_i = np_i \geqslant 5$. Observations are independent.
- **Test statistic**: $T = \sum_{i=1}^{k} \frac{(Y_i - np_i)^2}{np_i}$. Under H_0, $T \sim \chi^2_2$ approximately.
- **Observed test statistic**: $t_0 = 1.43$
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_2 \geqslant 1.43) = 0.49$
- **Decision**: Since the p-value is greater than 0.05, we do not reject the null hypothesis. The data are consistent with a Poisson distribution.

> We don't conclude that H_0 is true, just that the data are consistent with H_0. This is a subtle distinction.

If we wanted to implement this "manually" in R:

```
y = c(117, 94, 51, 15, 0, 0, 0, 1) # input the observed counts
x = 0:7 # define the corresponding groups
n = sum(y) # total number of samples (sample size)
k = length(y) # number of groups
(lam = sum(y * x)/n) # estimate the lambda parameter
```

```
[1] 0.8920863
```

```
p = dpois(x, lambda = lam) # obtain the p_i from the Poisson pmf
p
```

```
[1] 4.097999e-01 3.655769e-01 1.630631e-01 4.848878e-02 1.081404e-02
[6] 1.929412e-03 2.868670e-04 3.655859e-05
```

```
p[8] = 1 - sum(p[1:7]) # redefine the 8th element P(>=7) NOT P(7)
round(p, 5)
```

```
[1] 0.40980 0.36558 0.16306 0.04849 0.01081 0.00193 0.00029 0.00004
```

```
(ey = n * p) # calculate the expected frequencies
```

```
[1] 113.92436722 101.63037076  45.33153228  13.47988010   3.00630420
[6]   0.53637658   0.07974904   0.01141984
```

```
ey >= 5 #check assumption e_i >= 5 not all satisfied
```

```
[1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

Collapsing categories, so that the assumptions are met:

```
(yr = c(y[1:3], sum(y[4:8])))  # reduced category counts
```

```
[1] 117  94  51  16
```

```
(eyr = c(ey[1:3], sum(ey[4:8])))  # reduced category expected cell counts
```

```
[1] 113.92437 101.63037  45.33153  17.11373
```

```
all(eyr >= 5)  # check that all expected cell counts are >= 5
```

```
[1] TRUE
```

```
(pr = c(p[1:3], sum(p[4:8])))  # reduced category hypothesised probabilities
```

```
[1] 0.40979988 0.36557687 0.16306307 0.06156018
```

```
kr = length(yr)  # number of combined classes
(t0 = sum((yr - eyr)^2/eyr))  # test statistic
```

```
[1] 1.43721
```

```
(pval = 1 - pchisq(t0, df = kr - 1 - 1))  # p-value
```

```
[1] 0.4874317
```

What if we try to `chisq.test()`?

```
chisq = chisq.test(yr, p = pr)
chisq
```

```

        Chi-squared test for given probabilities

data:  yr
X-squared = 1.4372, df = 3, p-value = 0.6968
```

The degrees of freedom are wrong, is there a slightly easier way to get the right answer?

```
chisq$statistic
```

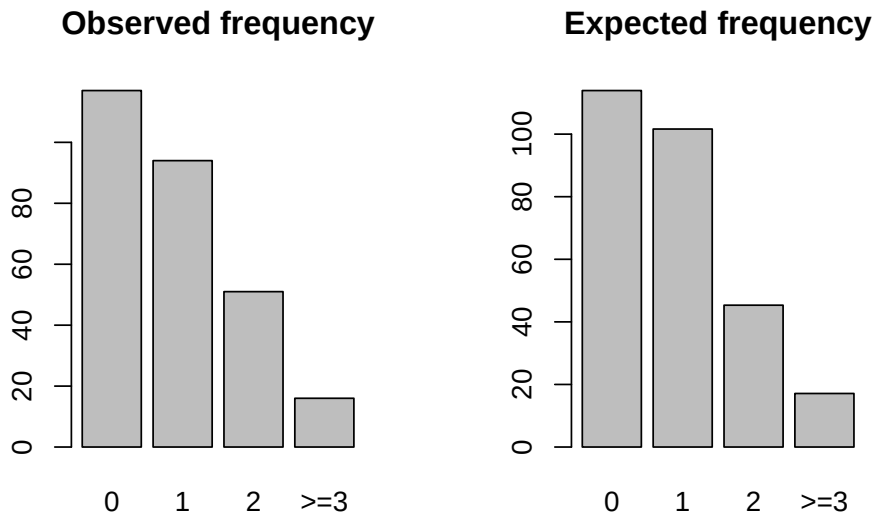
```
X-squared
  1.43721
```

```
pchisq(unname(chisq$statistic), df = 2, lower.tail = FALSE)
```

```
[1] 0.4874317
```

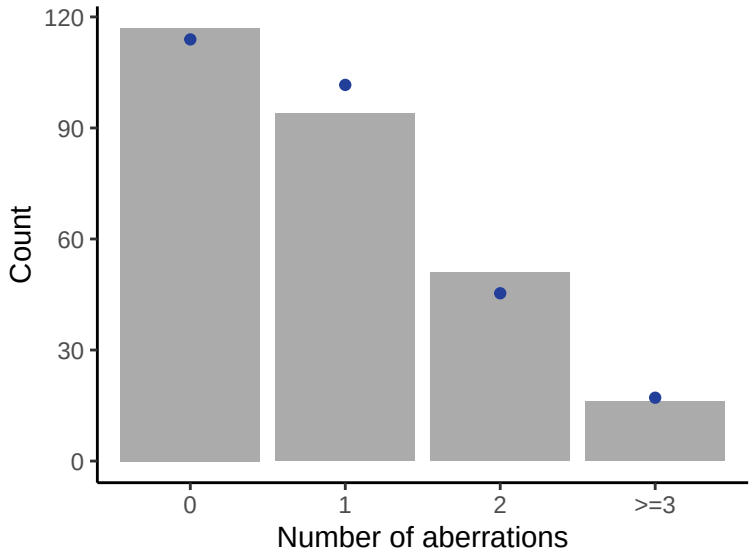
Visualising the comparison between observed and expected counts.

```
par(mfrow = c(1, 2))
xr = c("0", "1", "2", "⩾3")  # group labels
barplot(yr, names.arg = xr, main = "Observed frequency")
barplot(eyr, names.arg = xr, main = "Expected frequency")
```



Visualising the comparison between observed and expected counts.

```
dat = tibble::tibble(
  aberrations = factor(
    xr,
    levels = c("0","1","2","⩾3")
  ),
  observed = yr,
  expected = eyr
)
dat |> ggplot() +
  aes(x = aberrations, y = observed) +
  geom_col(alpha = 0.5) +
  geom_point(aes(y = eyr),
             colour = "#224099") +
  labs(y = "Count",
       x = "Number of aberrations") +
  theme_classic()
```



5

Measures of performance

5.1 Types of errors

Test result vs actual status

In general, **positive** = identified and **negative** = rejected. Therefore: **True positive** = correctly identified
False positive = incorrectly identified
True negative = correctly rejected
False negative = incorrectly rejected
We can summarise this in a table:

	Actual positive	**Actual negative**
Test positive	True positive (TP)	False positive (FP)
Test negative	False negative (FN)	True negative (TN)

Breast cancer

	Have breast cancer	**Does not have breast cancer**
Mammogram returns a positive test result	True positive (TP)	False positive (FP)
Mammogram returns a positive test result	False negative (FN)	True negative (TN)

- **True positive**: Mammogram returns a positive test result and you actually have breast cancer
- **False positive**: Mammogram returns a positive test result but you don't actually have breast cancer
- **False negative**: Mammogram returns a negative test result but you do actually have breast cancer
- **True negative**: Mammogram returns a negative test result and you don't actually have breast cancer

What's worse? False negative (FN) or false positive (FP)?

Extra notation

Let's formalise this. Let,

- D^+ be the event that an individual **has** a particular disease. The **prevalence** is the marginal probability of disease, $P(D^+)$
- D^- be the event that an individual **does not** have a particular disease.
- S^+ represent a positive screening test result or a symptom being present.
- S^- represent a negative screening test result or no symptom.

With this notation, we can start to quantify the influence of a risk factor or screening test (S) in the disease outcome (D).

	Actual positive $D+$	**Actual negative $D-$**
Test positive $S+$	True positive (TP)	False positive (FP)
Test negative $S-$	False negative (FN)	True negative (TN)

	Actual positive $D+$	**Actual negative $D-$**	
Test positive $S+$	a	b	$a+b$
Test negative $S-$	c	d	$c+d$
	$a+c$	$b+d$	$a+b+c+d$

- **False negative rate**: $P(S^- \mid D^+) = \frac{c}{a+c}$
- **False positive rate**: $P(S^+ \mid D^-) = \frac{b}{b+d}$
- **Sensitivity / Recall**: $P(S^+ \mid D^+) = \frac{a}{a+c}$, the probability that the test is positive given that the subject actually has the disease
- **Specificity**: $P(S^- \mid D^-) = \frac{d}{b+d}$, the probability that the test is negative given that the subject does not have the disease

- **Positive predictive value/Precision**: $P(D^+ \mid S^+) = \frac{a}{a+b}$, the probability that the subject has the disease given that the test is positive
- **Negative predictive value**: $P(D^- \mid S^-) = \frac{d}{c+d}$, the probability that the subject does not have the disease given that the test is negative
- **Accuracy**: $\frac{a+d}{a+b+c+d}$

Breast cancer

	Actual positive $D+$	Actual negative $D-$	
Test positive $S+$	9	99	108
Test negative $S-$	1	891	892
	10	990	1000

Prevalence: 1% (10/1000)
Sensitivity: 90% (9/10)
Specificity: 90% (891/990)
Positive predictive value (or precision)

$$P(D^+ \mid s^+) = \frac{9}{9+99} = 0.08$$

5.2 Conditional probability

Motivation

It's been suggested that lawn bowls is one of the most dangerous sports in the world (highest death rates).

> **Given that you're a young adult, should you be worried about playing?**

Conditional on being a young adult, the risk of playing lawn bowls is very small. Even though the probability of dying while playing lawn bowls is higher than many other sports, the overwhelming majority of lawn bowl related deaths are linked to the extreme age of the player.

Definition

- Let B be an event so that $P(B) > 0$
- The conditional probability of an event A given that B has occured is

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

  Rearranging, we also have
  - $P(A \cap B) = P(A \mid B)P(B)$
  - $P(A \cap B) = P(B \mid A)P(a)$
- For the *classical definition of probability* where we have a finite number of equally likely outcomes, then
  - $P(A)$ is the *proportion* of outcomes in the subset A
  - $P(B)$ is the *proportion* of outcomes in the subset B
  - $P(A \mid B)$ is the *proportion of outcomes in B that are also in A*

Bayes' rule

Bayes' rule allows us to reverse the conditioning set provided that we know some marginal probabilities,

$$P(B \mid A) = \frac{P(B \cap A)}{P(A)} = \frac{P(A \mid B)P(B)}{P(A \mid B)P(B) + P(A \mid B^c)P(B^c)}.$$

where B^c is the vent that B doesn't occur.
Note that we've used the law of total probability

$$P(A) = P(A \cap B) + P(A \cap B^c) = P(A \mid B)P(B) + P(A \mid B^c)P(B^c)$$

Efficacy of HIV tests

A study comparing the efficacy of HIV tests, reports on an experiment which concluded that HIV antibody tests have a **sensitivity** of 99.7% and a **specificity** of 98.5%. Suppose that a subject, from a population with a .1% **prevalence** of HIV, receives a positive test result. What is the positive predictive value?

- Mathematically, we want $P(D^+ \mid S^+)$ given the sensitivity, $P(S^+ \mid D^+) = .997$, the specificity, $P(S^- \mid D^-) = 0.985$, and the prevalence $P(D^+) = .001$

$$P(D^+ \mid S^+) = \frac{P(S^+ \mid D^+)P(D^+)}{P(S^+ \mid D^+)P(D^+) + P(S^+ \mid D^-)P(D^-)}$$

Need to find:

- $P(D^-) = 1 - P(D^+) = 1 - 0.001 = 0.999$
- $P(S^+ \mid D^-) = 1 - P(S^- \mid D^-) = 1 - 0.985 = 0.015$

$$P(D^+ \mid S^+) = \frac{P(S^+ \mid D^+)P(D^+)}{P(S^+ \mid D^+)P(D^+) + P(S^+ \mid D^-)P(D^-)} = \frac{.997 \times .001}{.997 \times .001 + .015 \times .999} = 0.062$$

In this population a positive test result only suggests a 6.2% probability that the subject has the disease I.e. the positive predictive value is 6.2% for this test.

- The low positive predictive value is due to low prevalence of disease and the somewhat modest specificity
- Suppose it was known that the subject was an intravenous drug user and routinely had intercourse with an HIV infected partner
- The evidence implied by a positive test result does not change because of the prevalence of disease in the subject's population, only our interpretation of that evidence changes

5.3 Evaluating classification models

Evaluating a classification model

Imagine a very simple classification model looking to classify a particular type of deformation, kyphosis, in children who have had corrective spinal surgery Chambers & Hastie (1991). The **outcome** is absence or presence of kyphosis after the operation.
The **predictor** is the number of the topmost vertebra operated on.

- If the highest vertebrae operated on was 9th or higher, we predict that kyphosis will be absent after surgery.

- If the highest vertebrae operated on was 8th or lower, we predict that kyphosis will be present after surgery.

The data can be found in the `kyphosis` data frame that comes with the **rpart** package.

```
data(kyphosis, package = "rpart")
dplyr::glimpse(kyphosis)
```

```
Rows: 81
Columns: 4
$ Kyphosis <fct> absent, absent, present, absent, absent, absent, absent, abse
    ~
$ Age      <int> 71, 158, 128, 2, 1, 1, 61, 37, 113, 59, 82, 148, 18, 1, 168,
    ~
$ Number   <int> 3, 3, 4, 5, 4, 2, 2, 3, 2, 6, 5, 3, 5, 4, 3, 3, 6, 5, 5, 4, 2
    ~
$ Start    <int> 5, 14, 5, 1, 15, 16, 17, 16, 16, 12, 14, 16, 2, 12, 18, 16, 1
    ~
```

```
kyphosis = kyphosis |>
  dplyr::mutate(
    prediction = if_else(Start ⩾ 9,
                         "Predict absent",
                         "Predict present")
  )
table(kyphosis$prediction, kyphosis$Kyphosis)
```

```
                absent present
  Predict absent      56       6
  Predict present      8      11
```

For our prediction model, define D^-, D^-, S^+, S^- and find the

- accuracy
- sensitivity
- specificity
- false negative rate
- false positive rate
- positive predictive value
- negative predictive value

Reordering the rows and columns in the table might help!

We can use the **yardstick** package from the **tidymodels** suite of packages to do this for us.

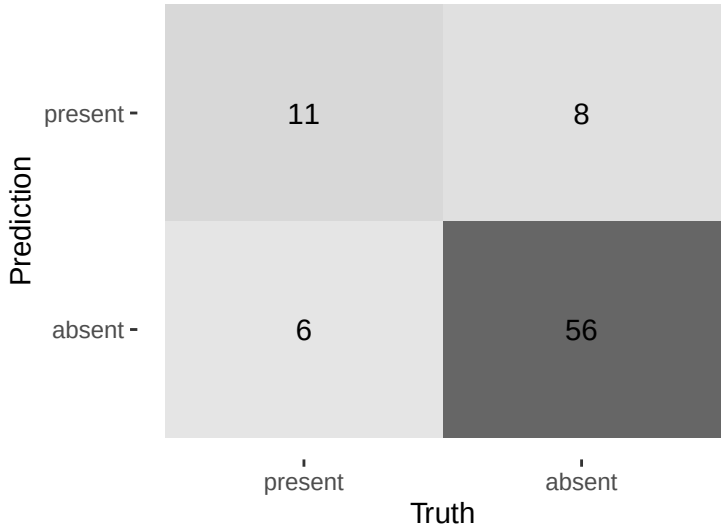
```
conf_mat = kyphosis |>
  dplyr::mutate(
    prediction = if_else(Start ⩾ 9,
                         "absent",
                         "present"),
    Kyphosis = factor(Kyphosis, levels = c("present", "absent")),
    prediction = factor(prediction, levels = c("present", "absent"))
  ) |>
  yardstick::conf_mat(truth = Kyphosis, estimate = prediction)
conf_mat
```

```
          Truth
Prediction present absent
   present      11      8
   absent        6     56
```

```
summary(conf_mat) |>
  select(-2) |>
  gt::gt() |>
  gt::fmt_percent(columns = 2,
                  decimals = 1)
```

.metric	.estimate
accuracy	82.7%
kap	50.0%
sens	64.7%
spec	87.5%
ppv	57.9%
npv	90.3%
mcc	50.2%
j_index	52.2%
bal_accuracy	76.1%
detection_prevalence	23.5%
precision	57.9%
recall	64.7%
f_meas	61.1%

```
autoplot(conf_mat, type = "heatmap")
```



6
Measures of risk

6.1 Motivating examples

Asthma and hay fever

A large group of infants with mild respiratory problems (but asthma free) is split into those with a family history of hay fever, and those without.
Random samples of 85 from the first group and 405 from the second group are selected for special study. Of these, the number diagnosed with asthma by the age of 12 are 25 and 70 respectively.

	Asthma	No Asthma	
Hay fever	25	60	85
No hay fever	70	335	405
	95	395	

Does a family history of hay fever increase the risk of developing asthma?

Hodgkin's disease and tonsillectomies

Vianna et al. (1971) collected data on a group of **101 patients suffering from Hodgkin's disease** and a comparable control group of **107 non-Hodgkin's patients**.
They were interested in the effect of tonsil tissue as a barrier to Hodgkin's disease. They found that in the Hodgkin's disease group, there had been 67 tonsillectomies. The corresponding figure for the non-Hodgkin's patients was 43.

	Hodgkin's disease		
Tonsillectomy	Yes	No	
Yes	67	43	110
No	34	64	98
	101	107	

Does having a tonsillectomy increase your risk of developing Hodgkin's disease?

Notation

How a study is conducted over time will affect various conditional probabilities.
To illustrate this we will use the following symbols.

- D^+ is the event that an individual **has** a particular disease.
- D^- is the event that an individual **does not have** a particular disease.
- R^+ is the event that an individual **has** a risk factor.
- R^- is the event that an individual **does not have** a risk factor.

6.2 Prospective studies

Prospective (or cohort study) studies

A study design where one or more samples (called cohorts) are followed prospectively and subsequent status evaluations with respect to a disease or outcome are conducted to determine which initial participants exposure characteristics (risk factors) are associated with it.
As the study is conducted, an outcome from participants in each cohort is measured and relationships with specific characteristics determined.

- A **prospective study** is based on subjects who are initially identified as *disease-free* and classified by presence or absence of a *risk factor*.
- A random sample from each group is followed in time (prospectively) until eventually classified by disease outcome.

Fictious example

- A **prospective study** was designed to assess the impact of sun exposure on skin damage in beach volleyball players.
- During a weekend tournament, players from one team wore waterproof, SPF 35 sunscreen, while players from the other team did not wear any sunscreen.
- At the end of the volleyball tournament players' skin from both teams was analyzed for texture, sun damage, and burns.
- Comparisons of skin damage were then made based on the use of sunscreen.
- The analysis showed a significant difference between the cohorts in terms of the skin damage.

Asthma

A large group of infants with mild respiratory problems (but asthma free) is split into those with a family history of hay fever, R^+, and those without, R^-. Random samples of 85 from the first group and 405 from the second group are selected for special study. Of these, the number diagnosed with asthma by the age of 12, D^+, are 25 and 70 respectively.

	D^+ Asthma by age 12	D^- No Asthma by age 12	Total
R^+ Family History of Hay fever	25	60	85
R^- No family history of hay fever	70	335	405
Total	**95**	**395**	**490**

- Risk factor is **family history of hay fever**
- Disease is **asthma by age 12**

In a prospective study, the **number of participants in each of the risk factor groups (row totals)** is fixed by design.

6.3 Retrospective studies

Retrospective (or case control) stzudies

A study that compares patients who have a disease or outcome of interest (cases) with patients who do not have the disease or outcome (controls), and looks back retrospectively to compare how frequently the exposure to a risk factor is present in each group to determine the relationship between the risk factor and the disease.

> A **retrospective study** is based on random samples from each of the two *outcome categories* which are followed back (retrospectively) to determine the presence or absence of the *risk factor* for each individual.

Fictious example

- There is a suspicion that zinc oxide, the white non-absorbent sunscreen traditionally worn by lifeguards is more effective at preventing sunburns that lead to skin cancer than absorbent sunscreen lotions.
- A **retrospective study** was conducted to investigate if exposure to zinc oxide is a more effective skin cancer prevention measure.
- The study involved comparing a group of former lifeguards that had developed cancer on their cheeks and noses (cases) to a group of lifeguards without this type of cancer (controls) and assess their prior exposure to zinc oxide or absorbent sunscreen lotions.
- This study would be **retrospective** in that the former lifeguards would be asked to recall which type of sunscreen they used on their face and approximately how often.

Hodgkin's disease

Vianna et al. (1971) collected data on a group of 101 patients suffering from Hodgkin's disease, D^+, and a comparable control group of 107 non-Hodgkin's patients, D^-. They were interested in the effect of tonsil tissue as a barrier to Hodgkin's disease. They found that in the D^+ group, there had been 67 tonsillectomies, R^+. The corresponding figure for the D^- group was 43.

	D^+ Hodgkin's	D^- Non-Hodgkin's	Total
R^+ Tonsillectomy	67	43	110
R^- No Tonsillectomy	34	64	98
Total	101	107	208

- Risk factor is **tonsillectomy**
- Disease is **Hodgkin's disease**

In a retrospective study the numbers **highlighted in green** are fixed by design.

6.4 Estimating population proportions

Suppose

- we have a large (but finite) population containing objects/individuals of two different types (say type 0 and type 1);
- it is desired to determine or at least estimate the overall proportion of type 1 but it is not feasible to examine every object/individual.

If we can take a random sample from the population then we can use the sample proportion of type 1 as an estimate of the population proportion of type 1.
Extending this idea, consider two events A and B,

- If we can take a random sample from the whole population, we can estimate $P(A)$ using the observed sample proportion with attribute A.
- If we can take a random sample from the subpopulation defined by B, we can estimate $P(A \mid B)$ using the observed sample proportion (of the subpopulation) with attribute A.

Application to prospective and retrospective studies

In both kinds of study we have

- a population;
- a subpopulation/attribute determined by a risk factor R^+ (with complementary subpopulation/attribute R^-);
- an subpopulation/attribute determined by having/developing the disease D^+ (with complementary subpopulation/attribute D^-)

The labels "subpopulation" and "attribute" here are mathematically equivalent (they both mean event).
The main difference between prospective and retrospective studies are which (sub)populations we can sample from.

Prospective study

In a **prospective study** we take two random samples:

- one from the risk factor group (subpopulation) R^+;
- another from the non-risk factor group R^-.
- We then (wait to) see how many in each group develop the disease.
- We can thus estimate $P(D^+ \mid R^+)$ as well as $P(D^- \mid R^-)$.
- We cannot however estimate $P(R^+ \mid D^+)$ or $P(R^- \mid D^-)$ since we did not take random samples from the disease group.

Retrospective studies

In a **retrospective study** we take two random samples:

- one from the disease group (subpopulation) D^+;
- another from the non-disease group (subpopulation) D^-.
- We then (look back to) see how many in each group were exposed to the risk factor.
- We can thus estimate $P(R^+ \mid D^+)$ as well as $P(R^- \mid D^-)$.
- We cannot however estimate $P(D^+ \mid R^+)$ or $P(D^- \mid R^-)$ since we did not take random samples from the risk factor group.

6.5 Relative risk

Measures of risk

These are different ways to measure the association between a risk factor/treatment and the disease outcome.
How the data is **sampled** will greatly impact the ways in which these methods are applicable and interpretable.

Relative risk

The relative risk is defined as a ratio of two conditional probabilities,

$$RR = \frac{P(D^+|R^+)}{P(D^+|R^-)}.$$

Since probabilities are bounded between 0 and 1,

$$RR = \frac{P(D^+|R^+)}{P(D^+|R^-)} \to \infty \quad \text{as} \quad P(D^+|R^-) \to 0,$$
$$RR = \frac{P(D^+|R^+)}{P(D^+|R^-)} \to 0 \quad \text{as} \quad P(D^+|R^+) \to 0,$$

and $RR \approx 1$ when $P(D^+ \mid R^+) \approx P(D^+ \mid R^-)$.

Note

If D and R are **independent** then,

$$P(D \mid R) = P(D),$$

and so,

$$\begin{aligned} RR &= \frac{P(D^+|R^+)}{P(D^+|R^-)} \\ &= \frac{P(D^+)}{P(D^+)} \\ &= 1. \end{aligned}$$

Interpretation

$$RR = \frac{P(D^+ \mid R^+)}{P(D^+ \mid R^-)}$$

The relative risk is the ratio of the probability of having the disease in the group with the risk factor to the probability of having the disease in the group without the risk factor.

- $RR = 1$ means there is **no difference** between the two groups.
- $RR < 1$ implies the disease is **less likely** to occur in the group with the risk factor.
- $Rr > 1$ implies the disease is **more likely** to occur in the group with the risk factor.

Prospective studies

	$D+$	$D-$	Total
$R+$	a	b	$\mathbf{a+b}$
$R-$	c	d	$\mathbf{c+d}$
	$a+c$	$b+d$	$a+b+c+d$

Given data from a **prospective study** or **from a sample of completed records** we can estimate these

- $P(D^+ \mid R^+) = \frac{a}{a+b}$
- $P(D^+ \mid R^-) = \frac{c}{c+d}$

- **Relative risk**: $\widehat{RR} = \frac{P(D^+|R^+)}{P(D^+|R^-)} = \frac{a(c+d)}{c(a+b)}$

Retrospective studies

	$D+$	$D-$	Total
$R+$	a	b	$a+b$
$R-$	c	d	$c+d$
	a + c	**b + d**	$a+b+c+d$

- In a **retrospective study**, D^+ and D^-, and we retrospectively assess each group them for their risk status, R^+ and R^-.
- The **column totals** are fixed by design.
- We "sampled on the outcome", choosing subjects based on D and then observing R.
- Due to the design, we cannot extract any information about the incidence of D in the population because the proportions of cases with D^+ and D^- were decided by the investigator. I.e. we cannot estimate $P(D^+ \mid R^+), P(D^+ \mid R^-)$, or $RR =$

Aspirin (relative risk)

Steering Committee of the Physicians' Health Study Research Group (1988) provide data on a 5 year (blind) study into the effect of taking aspirin every second day on the incidence of heart attacks.

	Myocardial infarction $D+$	No myocardial infarction $D-$	Total
Aspirin $R+$	104	10,933	11,037
Placebo $R-$	189	10,845	11,034
Total	293	21,778	22,071

Estimates for the proportion of each group (subpopulation) having heart attacks:

$$P(D^+ \mid R^+) = \frac{104}{10,933 + 104} = 0.0094$$
$$P(D^+ \mid R^-) = \frac{189}{10,845 + 189} = 0.0171$$

The estimated relative risk is:

$$\widehat{RR} = \frac{0.0094}{0.0171} = 0.55$$

> Participants are roughly half as likely to have myocardial infarction if they take aspirin.

6.6 Odds ratio

> **A common alternative to the relative risk is the odds ratio, denoted** OR

Odds are a ratio of probabilities. The **odds** are used as an alternative way of measuring the likelihood of an event occurring. If the probabilityof event A is $P(A)$ the **odds** of

event A is defined as

$$O(A) = \frac{P(A)}{1 - P(A)}.$$

In the risk/disease setting, the probability of disease for R^+ patients is $P(D^+ \mid R^+)$ and so the odds is,

$$O(D^+|R^+) = \frac{P(D^+|R^+)}{1 - P(D^+|R^+)} = \frac{P(D^+|R^+)}{P(D^-|R^+)}.$$

Equivalent definitions of odds ratio

The ratio of the odds of a disease for R^+ patients to the corresponding odds for R^- patients is the odds ratio, OR:

$$\text{Definition 1: } OR = \frac{O(D^+|R^+)}{O(D^+|R^-)} = \frac{P(D^+|R^+)}{P(D^-|R^+)} \Big/ \frac{P(D^+|R^-)}{P(D^-|R^-)}.$$

We can show that this ratio is identical to

$$\text{Definition 2: } \quad OR = \frac{O(R^+|D^+)}{O(R^+|D^-)} = \frac{P(R^+|D^+)}{P(R^-|D^+)} \Big/ \frac{P(R^+|D^-)}{P(R^-|D^-)}.$$

This means that OR can be found from both **prospective** and **retrospective** studies, unlike RR.

Invariance

Consider the table

	D^+	D^-	Total
R^+	a	b	$a + b$
R^-	c	d	$c + d$
	$a + c$	$b + d$	$a + b + c + d$

$$\text{Def. 1: } OR = \frac{P(D^+|R^+)}{P(D^-|R^+)} \Big/ \frac{P(D^+|R^-)}{P(D^-|R^-)} = \left(\frac{\frac{a}{a+b}}{\frac{b}{a+b}}\right) \Big/ \left(\frac{\frac{c}{c+d}}{\frac{d}{c+d}}\right) = \frac{ad}{bc}$$

$$\text{Def. 2: } OR = \frac{P(R^+|D^+)}{P(R^-|D^+)} \Big/ \frac{P(R^+|D^-)}{P(R^-|D^-)} = \left(\frac{\frac{a}{a+c}}{\frac{c}{a+c}}\right) \Big/ \left(\frac{\frac{b}{b+d}}{\frac{d}{b+d}}\right) = \frac{ad}{bc}$$

Same no matter which definition is used.

Interpretation

$$OR = \frac{P(D^+|R^+)}{P(D^-|R^+)} \Big/ \frac{P(D^+|R^-)}{P(D^-|R^-)} = \frac{ad}{bc},$$

If D and R are independent then $P(D \mid R) = P(D)$ and

$$OR = \frac{P(D^+|R^+)}{P(D^-|R^+)} \Big/ \frac{P(D^+|R^-)}{P(D^-|R^-)} = \frac{P(D^+)}{P(D^-)} \Big/ \frac{P(D^+)}{P(D^-)} = 1.$$

It can be shown that $OR = 1$ if and only if D and R are independent (there is no relationship between risk and disease).

Large odds ratios ($OR > 1$) implies increased risk of disease and small odd ratios ($OR < 1$) implies decreased risk of disease.

Aspirin (odds ratio)

	Myocardial infarction $D+$	No myocardial infarction $D-$	Total
Aspirin $R+$	104	10,933	11,037
Placebo $R-$	189	10,845	11,034
Total	293	21,778	22,071

The **odds ratio** is

$$OR = \frac{104 \times 10,845}{189 \times 10,933} = 0.55.$$

Interpretation

The estimated odds of heart attack for patients taking aspirin is 0.55 times the estimated odds for those taking the placebo.

Compare this with the relative risk of 0.55. These are similar because the disease is rare (low prevalence).

6.7 Standard errors and confidence intervals for odds ratios

Standard errors

The **odds ratio** estimator, OR, has a skewed distribution on $0, \infty$, with the neutral value being 1.
the **log odds ratio** estimator, $\log(OR)$, has a more symmetric distribution centred at 0 if there is no difference between the two groups.
The **asymptotic standard error** for the log odds ratio estimator, $\log(\widehat{OR})$, is

$$\text{SE}(\log(\widehat{OR})) = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}.$$

Standard errors and confidence intervals

A large sample 95% confidence interval for $\log \theta$ is approximately

$$\log(\widehat{OR}) \pm 1.96 \times \text{SE}(\log(\widehat{OR})).$$

Exponentiating the lower and upper ends of the $\log(OR)$ confidence interval gives us an approximate a confidence interval for the odds-ratio,

$$\left[\exp\left(\log(\widehat{OR}) - 1.96\text{SE}(\log(\widehat{OR}))\right), \exp\left(\log(\widehat{OR}) + 1.96\text{SE}(\log(\widehat{OR}))\right)\right].$$

Note that these should only be applied if a, b, c and d are *reasonably large* (so that asymptotics hold).

Aspirin

	Myocardial infarction $D+$	No myocardial infarction $D-$	Total
Aspirin $R+$	104	10,933	11,037
Placebo $R-$	189	10,845	11,034
Total	293	21,778	22,071

$$\widehat{OR} = \frac{104 \times 10,845}{189 \times 10,933} = 0.55 \quad \text{and} \quad \log(\widehat{OR}) = -0.6$$
$$\text{SE}(\log(\widehat{OR})) = \sqrt{\frac{1}{104} + \frac{1}{189} + \frac{1}{10933} + \frac{1}{10845}} = 0.12.$$

A 95% confidence interval for the log odds-ratio is

$$-0.6 \pm 1.96 \times 0.12 \approx (-0.84, -0.36).$$

A 95% confidence interval for the odds-ratio is

$$(e^{-0.84}, e^{-0.36}) \approx (0.43, 0.69).$$

6.8 Examples revisited

Hodgkin's disease

Vianna et al. (1971) collected data on a group of 101 patients suffering from Hodgkin's disease, and a comparable control group of 107 non-Hodgkin's patients. They were interested in the effect of tonsil tissue as a barrier to Hodgkin's disease.

	D^+ Hodgkin's	D^- Non-Hodgkin's	Total
R^+ Tonsillectomy	67	43	**110**
R^- No Tonsillectomy	34	64	**98**
Total	**101**	**107**	**208**

The estimated odds-ratio is,

$$\widehat{OR} = \frac{67 \times 64}{43 \times 34} = 2.93.$$

Hence, the odds of a tonsillectomy patient having Hodgkin's disease are three times the odds of a non-tonsillectomy patient having Hodgkin's.
The log odds ratio is

$$\log(\widehat{OR}) = \log(2.93) = 1.07$$

with standard error,

$$\text{SE}(\log(\widehat{OR})) = \sqrt{\tfrac{1}{67} + \tfrac{1}{43} + \tfrac{1}{34} + \tfrac{1}{64}} = 0.29.$$

A 95% confidence interval for the log odds-ratio is

$$1.07 \pm 1.96 \times 0.29 \approx (0.51, 1.63)$$

and a 95% confidence interval for the odds-ratio is

$$(e^{0.51}, e^{1.63}) \approx (1.66, 5.10)$$

Asthma

A large group of infants with mild respiratory problems (but asthma free) is split into those with a family history of hay fever, R^+, and those without, R^-. Random samples of 85 from the first group and 405 from the second group are selected for special study. Of these, the number diagnosed with asthma by the age of 12, D^+, are 25 and 70 respectively.

	D^+ Asthma by age 12	D^- No Asthma by age 12	Total
R^+ Family History of Hay fever	25	60	85
R^- No family history of hay fever	70	335	405
Total	**95**	**395**	**490**

The odds ratio, log odds ratio and the standard error of the log odds ratio are,

$$\widehat{OR} = \frac{25 \times 335}{60 \times 70} = 1.99, \qquad \log(\widehat{OR}) = 0.69,$$
$$\text{and} \quad \text{SE}(\log(\widehat{OR})) = \sqrt{\frac{1}{25} + \frac{1}{60} + \frac{1}{70} + \frac{1}{335}} = 0.27.$$

A 95% confidence interval for the log odds-ratio is

$$0.69 \pm 1.96 \times 0.27 \approx (0.16, 1.22)$$

and a 95% confidence interval for the odds-ratio is

$$(e^{0.16}, e^{1.22}) \approx (1.17, 3.39)$$

7

Testing for homogeneity

7.1 Testing for homogeneity in 2×2 tables

COVID treatment

Liu et al. (2020) performed a study where 39 plasma recipients were propensity-score matched to 156 control patients to assess the effectiveness of convalescent plasma therapy in patients with severe or life-threatening COVID-19 at The Mount Sinai Hospital in New York City.

	Died	Discharged	Censored	
Plasma Treatment	5	28	6	39
No Plasma Treatment	38	104	14	156
	43	132	20	195

Test of homogeneity

- Suppose that observations are sampled from two independent populations, each of which is categorised according to the same set of outcomes.
- We want to test whether the distribution (proportions) of the outcomes are the same across the different populations.

In our COVID-19 treatment example, we will consider the proportions of patients treated with plasma who died or were discharged and (separately) the proportion of patients who were not treated with plasma who died or were discharged.

	Outcome 1: Died	Outcome 2: Discharged	Row Total (fixed)
Population 1: Plasma Treatment	p_{11}	p_{12}	$p_{11} + p_{12} = 1$
Population 2: No Plasma Treatment	p_{21}	p_{22}	$p_{11} + p_{22} = 1$

Under the null hypothesis of **homogeneity** the proportion of patients who died is the same in both populations $p_{11} = p_{21}$, and the proportion of patients who were discharged is the same in both populations p_{12}, p_{22}.

Two way contigency table

	Died	Discharged	
Plasma Treatment	5	28	33
No Plasma Treatment	38	104	142
	43	132	175

- A contingency table allows us to tabulate data from multiple categorical variables.
- Contingency tables are heavily used in health, survey research, business, intelligence, engineering and scientific research.
- The above table is a two-way a contingency table, specifcally a 2×2 contingency table.

Two way contigency table in R

```
# covid_data = readxl::read_excel("extra/covid_treatment_outcomes.xlsx") |>
#   janitor::clean_names()
raw_dat = read_csv("https://raw.githubusercontent.com/DATA2002/data/master/
    covidplasma.csv")


dat = raw_dat |>
          mutate(subject = as.character(subject)) |>
          filter(outcome != "Censored") |>
          mutate(treatment = factor(treatment, levels = c("Plasma", "No plasma
              ")),
```

```
          outcome = factor(outcome, levels = c("Died", "Discharged")))
dat |>
  janitor::tabyl(treatment,outcome) |>
gt::gt()
```

treatment	Died	Discharged
Plasma	5	28
No plasma	38	104

Notation

In 2×2 contingency tables, for column c and row r let,

$$y_{\bullet c} \sum_{j=1}^{2} y_{jc} \quad \text{and} \quad y_{r\bullet} \sum_{j=1}^{2} y_{rj}$$

	Died	Discharged	Row total (fixed)
Plasma Treatment	y_{11}	y_{12}	$y_{\bullet 1} = n_1$
No Plasma Treatment	y_{21}	y_{22}	$y_{\bullet 2} = n_2$
Column Total	$y_{\bullet 1}$	$y_{\bullet 2}$	$n = n_1 + n_2$

Under the null hypothesis of homogeneity we have $p_{11} = p_{21}$ and $p_{12} = p_{22}$ so our best estimate of the proportion in each category is the column total divided by the overall sample size, $\hat{p}_{ij} = \frac{y_{\bullet j}}{n}$.
Under H_0, the expected counts are $e_{ij} = n_i \hat{p}_{ij} = y_{i\bullet} \frac{y_{\bullet j}}{n}$

Chi-squared test of homogeneity

Using our observed counts and expected counts for each cell, we can construct a chi-squared test for homogeneity,

$$T = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}} \sim \chi_1^2 \quad \text{approximately.}$$

The expected cell counts are,

$$e_{ij=n_i \hat{p}_{ij}} = y_{i\bullet} \frac{y_{\bullet j}}{n} = \frac{\text{Row } i \text{ total} \times \text{Column } j \text{ total}}{\text{Overall total}}$$

Workflow: Chi-squared test of homogeneity for a 2×2 contigency table

- **Hypothesis**: $H_0 : p_{11} = p_{21}$ and $p_{12} = p_{22}$ vs $H_1 : p_{11} \neq p_{21}$ $p_{12} \neq p_{22}$, or equivalently, H_0 : the proportions for the two outcomes are the same across the two populations vs H_1 : the proportions for the two outcomes are different across the two populations.
- **Assumptions**: observations randomly sampled from two independent populations and $e_{ij} = y_{i\bullet} y_{\bullet j} / n \geqslant 5$
- **Test statistic**: $T = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi_1^2$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}}$.
- **P-value**: $P(T \geqslant t_0) = P(\chi_1^2 \geqslant t_0)$

- **Decision**: Reject H_0 if the p-value $< \alpha$.

COVID treatment

	Died	Discharged	Row Total
Plasma Treatment	5	28	39
No Plasma Treatment	38	104	156
Column Total	43	132	195

	Died	Discharged	Row Total
Plasma Treatment	$\frac{33\times43}{175} = 8.11$	$\frac{33\times43}{132} = 24.89$	33
No Plasma Treatment	$\frac{142\times43}{175} = 34.89$	$\frac{142\times43}{132} = 107.11$	142
Column Total	43	132	175

$$
\begin{aligned}
t_0 &= \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{(Y_{ij}-e_{ij})^2}{e_{ij}} \\
&= \frac{(5-8.11)^2}{8.11} + \frac{(28-24.89)^2}{24.89} + \frac{(38-34.89)^2}{34.89} + \frac{(104-107.11)^2}{107.11} \\
&= 1.9471
\end{aligned}
$$

- **Hypothesis**: $H_0 : p_{11} = p_{21}$ and $p_{12} = p_{22}$ vs $H_1 : p_{11} \neq p_{21}\ p_{12} \neq p_{22}$, **or** H_0 : death and discharge outcomes are homogenous across both the plasma and non-plasma populations vs H_1 : the proportion of death and discharge outcomes are different across both the plasma and non-plasma populations.
- **Assumptions**: observations randomly sampled from two independent populations and $e_{ij} = y_{i\bullet}y_{\bullet j}/n \geqslant 5$
- **Test statistic**: $T = \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{(Y_{ij}-e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi^2_{k-1-q}$ approximately.
- **Observed test statistic**: $t_0 = 1.9471$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_1 \geqslant 1.9471) = 0.16$
- **Decision**: Do not reject H_0 as the p-value is quite large, i.e. there is no evidence to suggest there is a significant difference in the proportion of dead and discharged patients between the plasma and control groups.

```
covid_tab = table(dat$treatment, dat$outcome)
covid_tab
```

```
           Died Discharged
  Plasma      5         28
  No plasma  38        104
```

```
chisq.test(covid_tab, correct = FALSE)
```

```
        Pearson's Chi-squared test

data:  covid_tab
X-squared = 1.9471, df = 1, p-value = 0.1629
```

7.2 Testing for homogeneity in general tables

Voter sentiment

A survey of voter sentiment was conducted in Labor and Liberal to compare the fraction of voters favouring a new tax reform package. Random samples of 100 voters were polled in each of the two parties, with results as follows:

	Approve	Not Aprrove	No Comment	Row Total
Labor	63	29	9	100
Liberal	47	46	7	100
Column Total	109	75	16	200

A general two-way contigency table

	Category 1	Category 2	…	Category c	Row Total (fixed)
Population 1	y_{11}	y_{12}	$\cdots$	y_{1c}	$y_{1\bullet}$
Population 2	y_{21}	y_{22}	$\cdots$	y_{2c}	$y_{2\bullet}$
⋮	⋮	⋮		⋮	⋮
Population r	y_{r1}	y_{r2}	$\cdots$	y_{rc}	$y_{1\bullet}$
Column Total	$y_{\bullet 1}$	$y_{\bullet 2}$	$\cdots$	$y_{\bullet c}$	$y_{\bullet\bullet}$

- A contingency table allows us to tabulate data from multiple categorical variables.
- We call the above table a two-way a contingency table, specifically a $r \times c$ contigency table
- There are rc categories and either row or column totals are fixed (therefore, n is also fixed)

	Category 1	Category 2	…	Category c	Row Total
Population 1	p_{11}	p_{12}	$\cdots$	p_{1c}	$p_{1\bullet} = 1$
Population 2	p_{21}	p_{22}	$\cdots$	p_{2c}	$p_{2\bullet} = 1$
⋮	⋮	⋮		⋮	⋮
Population r	p_{r1}	p_{r2}	$\cdots$	p_{rc}	$p_{1\bullet} = 1$

Under the null hypothesis of **homogeneity** $p_{11} = p_{21} = \cdots = p_{r1}$, $p_{12} = p_{22} = \cdots = p_{r2}$, and $p_{1c} = p_{2c} = \cdots = p_{rc}$

Test of homogeneity

	Approve	Not Approve	No Comment	Row Total
Labor	p_{11}	p_{12}	p_{13}	$p_{1\bullet} = 1$
Liberal	p_{21}	p_{22}	p_{23}	$p_{2\bullet} = 1$
Category Total	$p_{\bullet 1}$	$p_{\bullet 2}$	$p_{\bullet 3}$	n

Under the null hypothesis of homogeneity, $p_{11} = p_{21}$, $p_{12} = p_{22}$ and $p_{13} = p_{23}$.
We don't know p_{ij}, so estimate using category proportions, $\hat{p}_{ij} = \frac{y_{\bullet j}}{n}$.

Under H_0, the expected counts are,

$$e_{ij} = n_i \hat{p}_{ij} = y_{i\bullet} = \frac{y_{\bullet j}}{n} = \frac{\text{Row } i \text{ total} \times \text{Column } j \text{ total}}{\text{Overall total}},$$

and the test statistic is, $T = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}} \sim \chi^2_{(r-1)(c-1)}$ approximately.

Degrees of freedom

	Approve	Not Approve	No Comment	Row Total
Labor	y_{11}	y_{12}	y_{13}	$y_{1\bullet} = n_1$
Liberal	y_{21}	y_{22}	y_{23}	$y_{2\bullet} = n_2$
Category Total	$y_{\bullet 1}$	$y_{\bullet 2}$	$y_{\bullet 3}$	n

- We need to estimate 3 parameters $\hat{p}_{\bullet 1}, \hat{p}_{\bullet 2}$ and $\hat{p}_{\bullet 3}$.
- The degrees of freedom for a 2×3 table is 2
- More generally the degrees of freedom for a $r \times c$ table is $(r-1)(c-1)$.

Workflow: Chi-squared test of homogeneity for a $r \times c$ contigency table

- **Hypothesis**: $H_0 : p_{1j} = p_{2j} = \cdots = p_{rj}$ for $j = 1, 2, \ldots, c$ vs H_1 : Not all equalities hold, or, H_0 : the distribution of outcomes is the same across all r populations vs H_1 : the distribution outcomes differ across the r populations.
- **Assumptions**: $e_{ij} = y_{i\bullet} y_{\bullet j}/n \geqslant 5$ and independent observations sampled from the r populations.
- **Test statistic**: $T = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi^2_{(r-1)(c-1)}$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}}$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_{(r-1)(c-1)} \geqslant t_0)$
- **Decision**: Reject H_0 if the p-value $< \alpha$.

Voter sentiment

```
y = c(62, 47, 29, 46, 9, 7)
n = sum(y)
c = 3
r = 2
voter_tab = matrix(y, nrow = r, ncol = c)
# default is to fill by column
colnames(voter_tab) = c("Approve",
                        "Not approve",
                        "No comment")
rownames(voter_tab) = c("Labor", "Liberal")
voter_tab
```

```
        Approve Not approve No comment
Labor        62          29          9
Liberal      47          46          7
```

```
chisq.test(voter_tab, correct = FALSE)
```

```
        Pearson's Chi-squared test
```

```
data:  voter_tab
X-squared = 6.1676, df = 2, p-value = 0.04579
```

- **Hypothesis**: $H_0 : p_{1j} = p_{2j}$ for $j = 1, 2, 3$ vs H_1 : Not all equalities hold, or H_0 : the proportions of approve, not approve and no comment are homogenous across Liberal and Labour voters vs H_1 : the proportions of approve, not approve and no comment are not the same across Liberal and Labour voters.
- **Assumptions**: independent observations and $e_{ij} = y_{i\bullet}y_{\bullet j}/n \geqslant 5$.
- **Test statistic**: $T = \sum_{i=1}^{2}\sum_{j=1}^{3}\frac{(Y_{ij}-e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi^2_2$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{2}\sum_{j=1}^{3}\frac{(Y_{ij}-e_{ij})^2}{e_{ij}} = 6.1676$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_2 \geqslant 6.1676)$
- **Decision**: The p-value is less than 0.05, therefore at the 5% level of significance, we reject the null hypothesis and conclude that voter preferences about the new tax reform package are not homogenous across Liberal and Labour voters.

8

Testing for independence

8.1 Testing for independence in 2×2 tables

Titanic

- The `Titanic` dataset comes preloaded in R.
- This data set provides information on the fate of passengers on the fatal maiden voyage of the ocean liner Titanic, summarized according to economic status (class), sex, age and survival.

```
titanic_df = as.data.frame(Titanic)
head(titanic_df)
```

```
  Class    Sex   Age Survived Freq
1   1st   Male Child       No    0
2   2nd   Male Child       No    0
3   3rd   Male Child       No   35
4  Crew   Male Child       No    0
5   1st Female Child       No    0
6   2nd Female Child       No    0
```

```
y_mat = xtabs(Freq ~ Sex + Survived,
              data = titanic_df)
y_mat
```

```
        Survived
Sex        No  Yes
  Male   1364  367
  Female  126  344
```

Tests for independence

- There are times where a sample from a population may be categorised according to two categorical variables.
- Each categorical variable has various levels (factors).
- It is of interest to know whether the categorical variables are independent.
- We summarise what we know in a contingency table:

	Surive	Did Not Survive	Row Total
Male	y_{11}	y_{12}	$y_{1\bullet} = n_1$
Female	y_{21}	y_{22}	$y_{2\bullet} = n_2$
Column Total	$y_{\bullet 1}$	$y_{\bullet 2}$	n

Here the two variables are: **survival status** and **gender**.

Table of proportions

Let p_{ij} denote the probability of an observation falling in the $(i, j)^{\text{th}}$ category. The marginal row and column probabilities are respectively:

$$p_{i\bullet} = \sum_{j=1}^{2} p_{ij} \quad \text{and} \quad p_{\bullet j} \sum_{i=1}^{2} p_{ij}$$

	Surive	Did Not Survive	Row Total
Male	p_{11}	p_{12}	$p_{1\bullet} = n_1$
Female	p_{21}	p_{22}	$p_{2\bullet} = n_2$
Column Total	$p_{\bullet 1}$	$p_{\bullet 2}$	1

Test statistic

Under the null hypothesis of independence, the expected frequencies are $e_{ij} = np_{ij} = np_{i\bullet}p_{\bullet j}$. A large test statistic,

$$T = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(Y_{ij} - e_{ij})^2}{e_{ij}} = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(Y_{ij} - np_{i\bullet}p_{\bullet j})^2}{np_{i\bullet}p_{\bullet j}}$$

indicates that we should reject H_0.

- However, T includes unknown parameters $p_{i\bullet}$ and $p_{\bullet j}$.
- Estimate $p_{i\bullet}$ and $p_{\bullet j}$ by $\hat{p}_{i\bullet} = y_{i\bullet}/n$ and $\hat{p}_{\bullet j} = y_{\bullet j}/n$.
- Calculate the observed test statistic,

$$t_0 = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(y_{ij} - n\hat{p}_{i\bullet}\hat{p}_{\bullet j})^2}{n\hat{p}_{i\bullet}\hat{p}_{\bullet j}} = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(y_{ij} - y_{i\bullet}y_{\bullet j}/n)^2}{y_{i\bullet}y_{\bullet j}/n}$$

Workflow: Chi-squared test of independence for a 2×2 contigency table

- **Hypothesis**: $H_0 : p_{ij} = p_{i\bullet}p_{\bullet j}$ for $i = 1, 2$ and $j = 1, 2$ vs H_1 : Not all equalities hold, or H_0 : variable 1 is independent of variable 2 vs H_1 : the two variables are not independent.
- **Assumptions**: observations randomly sampled from two independent populations and $e_{ij} = y_{i\bullet}y_{\bullet j}/n \geqslant 5$

- **Test statistic**: $T = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(Y_{ij}-e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi^2_1$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(y_{ij}-y_{i\bullet}y_{\bullet j}/n)^2}{y_{i\bullet}y_{\bullet j}/n}$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_1 \geqslant t_0)$
- **Decision**: Reject H_0 if the p-value $< \alpha$.

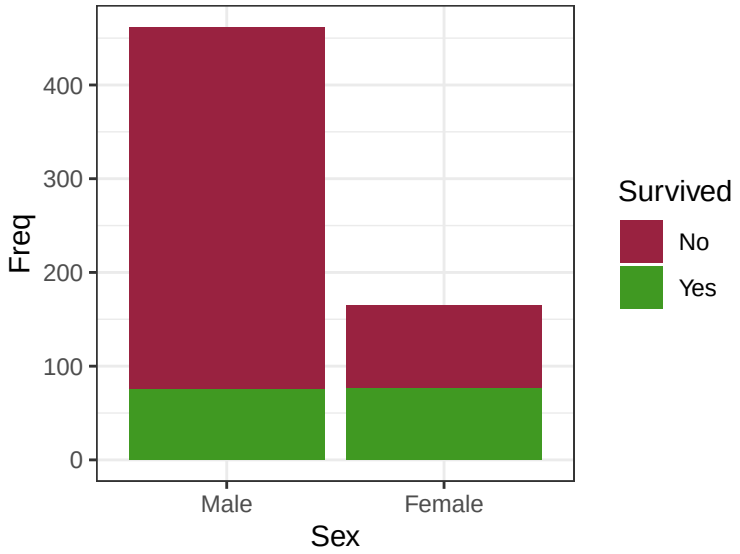
Titanic

Looking at Adults in the Third Class:

```
t3a = titanic_df |>
  filter(Class == "3rd", Age == "Adult")
y_mat = xtabs(Freq ~ Sex + Survived, data = t3a)
y_mat
```

```
        Survived
Sex       No Yes
  Male   387  75
  Female  89  76
```

```
t3a |>
  ggplot() +
  aes(x = Sex, y = Freq, fill = Survived) +
  geom_col() +
  scale_fill_manual(values=c("#992240", "#409922")) +
  theme_bw()
```




```
chisq.test(y_mat, correct = FALSE)
```

```
        Pearson's Chi-squared test

data:  y_mat
X-squared = 59.159, df = 1, p-value = 1.454e-14
```

- **Hypothesis**: H_0 : $p_{ij} = p_{i\bullet}p_{\bullet j}$ for $i = 1, 2$ and $j = 1, 2$ vs H_1 : Not all equalities hold, or H_0 : gender is independent of survival for adults in third class vs H_1 : gender and survival are not independent for adults in third class.
- **Assumptions**: $e_{ij} = y_{i\bullet}y_{\bullet j}/n \geqslant 5$, do you think we have independent obs?

- **Test statistic**: $T = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(Y_{ij}-e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi_1^2$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(y_{ij}-y_{i\bullet}y_{\bullet j}/n)^2}{y_{i\bullet}y_{\bullet j}/n} = 59.159$.
- **P-value**: $P(T \geqslant t_0) = P(\chi_1^2 \geqslant 59.159) < 0.001$
- **Decision**: We reject the null hypothesis that sex is independent of survival for adults in third class as the p-value is very small (much smaller than 0.05). Hence, there is evidence to suggest that survival status of passengers on the Titanic is related to the sex of the passenger.

8.2 Testing for independence in general tables

Advertisement

200 randomly sampled people are classified according to their income level and their reactions to an advertisement for a product (positive, negative, no opinion).

	Positive	Negative	No Opinon	Total
High Income	24	46	38	108
Low Income	32	22	38	92
Total	56	68	76	200

General 2-Way contigency tables

We now turn our attention from 2×2 table to $r \times c$ tables to $r \times c$ tables with classifying factors R at r levels and S at c levels.
Let y_{ij} be the observed count in the (i, j)th cell. Summarise this in a contingency table,

	$S=1$	$S=2$	$\ldots$	$S=c$	Total
$R=1$	y_{11}	y_{12}	$\cdots$	y_{1c}	$y_{1\bullet}$
$R=2$	y_{21}	y_{22}	$\cdots$	y_{2c}	$y_{2\bullet}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$R=r$	y_{r1}	y_{r2}	$\cdots$	y_{rc}	$y_{1\bullet}$
Total	$y_{\bullet 1}$	$y_{\bullet 2}$	$\cdots$	$y_{\bullet c}$	$y_{\bullet\bullet}$

$$y_{\bullet\bullet} = \sum_{i=1}^{r}\sum_{j=1}^{c} y_{ij} = n$$

Estimating p_{ij} under independence

Suppose we have a completely random sample of size n classified by R and S. We want to test H_0 : R and S are independent.
If R and S are independent, by definition of independence,

$$p_{ij} = P(R = i, S = j) = P(R = i)P(S = j)$$

We estimate the marginal probabilities $P(R = i)$ and $S = j$ via

- $(P = i) = p_{i\bullet} = \sum_{j=1}^{c} p_{ij}$ estimated by $\hat{p}_{i\bullet} = \frac{1}{n}\sum_{j=1}^{c} y_{ij} = \frac{y_{i\bullet}}{n}$, and
- $(S = j) = p_{j\bullet} = \sum_{i=1}^{r} p_{ij}$ estimated by $\hat{p}_{\bullet j} = \frac{1}{n}\sum_{i=1}^{r} y_{ij} = \frac{y_{\bullet j}}{n}$,

Under the independence hypothesis, the expected frequency in the (i, j)th cell is

$$e_{ij} = n\hat{p}_{ij} = n\hat{p}_{i\bullet}\hat{p}_{\bullet j} = \frac{y_{i\bullet}y_{\bullet j}}{n}$$

Advertisement

	Positive	Negative	No Opinon	Row Total
High Income	p_{11}	p_{12}	p_{13}	$p_{1\bullet}$
Low Income	p_{21}	p_{22}	p_{23}	$p_{2\bullet}$
Column Total	$p_{\bullet 1}$	$p_{\bullet 2}$	$p_{\bullet 3}$	1

Recall: X and Y are said to be independent if

$$P(X = x \mid Y = y) = P(X = x)P(Y = y)$$

Let X be a random variable representing opinion and let Y be a random variable representing income. Under independence,

$$\begin{aligned} p_{12} &= P(X = \text{Negative}, Y = \text{High Income}) \\ &= P(X = \text{Negative})P(Y = \text{High Income}) \\ &= p_{\bullet 2}p_{1\bullet} \end{aligned}$$

Test statistic

Under H_0 of independence, the expected frequencies are $e_{ij} = np_{ij} = np_{i\bullet}p_{\bullet j}$ for $i = 1, 2, \ldots, r$ and $j = 1, 2, \ldots, c$. Hence

$$T = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(Y_{ij} - e_{ij})^2}{e_{ij}} = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(Y_{ij} - np_{i\bullet}p_{\bullet j})^2}{np_{i\bullet}p_{\bullet j}}$$

will be large if we should reject H_0.
However T includes unknown parameters $p_{i\bullet}$ and $p_{\bullet j}$. We estimate $p_{i\bullet}$ and $p_{\bullet j}$ with

$$\hat{p}_{i\bullet} = y_{i\bullet}/n \quad \hat{p}_{\bullet j} = y_{\bullet j}/n$$

Hence, we may calculate the observed test statistic as

$$t_0 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(Y_{ij} - n\hat{p}_{i\bullet}\hat{p}_{\bullet j})^2}{n\hat{p}_{i\bullet}\hat{p}_{\bullet j}} = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(y_{ij} - y_{i\bullet}y_{\bullet j}/n)^2}{y_{i\bullet}y_{\bullet j}/n}$$

Degrees of freedom

	Positive	Negative	No Opinion	Total
High Income	y_{11}	y_{12}	y_{13}	$y_{1\bullet} = n_1$
Low Income	y_{21}	y_{22}	y_{23}	$y_{2\bullet} = n_2$
Total	$y_{\bullet 1}$	$y_{\bullet 2}$	$y_{\bullet 3}$	n

- The degrees of freedom for a $r \times c$ table is $(r - 1)(c - 1)$.

$$(rc - 1) - (r - 1) - (c - 1) = rc - r - c + 1 = (r - 1)(c - 1)$$

- The degrees of freedom for a 2×3 table is 2

Workflow: Chi-squared Test of independence between two factors

- **Hypothesis**: $H_0 : p_{ij} = p_{i\bullet}p_{\bullet j}, i = 1, 2, \ldots, r, j = 1, 2, \ldots, c$ vs H_1 : Not all equalities hold.
- **Assumptions**: Independent observations and $e_{ij} = y_{i\bullet}y_{\bullet j}/n \geqslant 5$.
- **Test statistic**: $T = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(Y_{ij}-e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi^2_{(r-1)(c-1)}$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(y_{ij}-y_{i\bullet}y_{\bullet j}/n)^2}{y_{i\bullet}y_{\bullet j}/n}$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_{(r-1)(c-1)} \geqslant t_0)$
- **Decision**: Reject H_0 if the p-value $< \alpha$.

As noted in the assumptions, this test should only really be applied when all cells have expected counts greater than 5, $e_{ij} \geqslant 5$. . If there are expected cell counts less than 5, consider using Fisher's exact test (2×2) or simulation (general $r \times c$ case).

Advertisement

200 randomly sampled people are classified according to their income level and their reactions to an advertisement for a product (positive, negative, no opinion).

	Positive	Negative	No Opinon	Total
High Income	24	46	38	108
Low Income	32	22	38	92
Total	56	68	76	200

The test for independence between factors of *income level* and *opinion* is

- **Hypothesis**: $H_0 : p_{ij} = p_{i\bullet}p_{\bullet j}, i = 1, 2, j = 1, 2, e$ vs H_1 : Not all equalities hold **or** H_0 : income level is independent of opinion vs H_1: income level is not independent of opinion.
- **Assumptions**: independent observations (satisfied, they were randomly sampled) $e_{ij} = y_{i\bullet}y_{\bullet j}/n \geqslant 5$ (satisfied, checked with calculations).
- **Test statistic**: $T = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(Y_{ij}-e_{ij})^2}{e_{ij}}$. Under $H_0, T \sim \chi^2_2$ approximately.
- **Observed test statistic**: $t_0 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(y_{ij}-y_{i\bullet}y_{\bullet j}/n)^2}{y_{i\bullet}y_{\bullet j}/n} = 8.39$.
- **P-value**: $P(T \geqslant t_0) = P(\chi^2_2 \geqslant 8.39) = 0.015$
- **Decision**: Since the p-value is less than 0.05, the data provide evidence against H_0. There is evidence to suggest that there is an association between income level and opinion.

```
y = c(24,32,46,22,38,38)
n = sum(y)
c = 3
r = 2
y_mat = matrix(y, nrow = r, ncol = c)
colnames(y_mat) = c("Positive", "Negative", "No opinion")
rownames(y_mat) = c("High income", "Low income")
y_mat
```

```
           Positive Negative No opinion
High income       24       46         38
Low income        32       22         38
```

```
chisq.test(y_mat, correct = FALSE)
```

```
        Pearson's Chi-squared test

data:  y_mat
X-squared = 8.3871, df = 2, p-value = 0.01509
```

9

Testing in small samples

9.1 Fisher's exact test

Lady tasting tea

Given a cup of tea with milk, a lady claims she can discriminate as to whether milk or tea was first added to the cup.

> **How could we test this claim? What information would we need?**

Fisher proposed preparing 8 cups of tea

- 4 cups where tea was added before milk
- 4 cups where milk was added before tea

The lady would then be randomly given the cups of tea and asked to identify the 4 where tea was added before milk.
We would need to record:

- Which cups had tea or milk added first (**truth**).
- Which cups the lady claimed had tea or milk added first (**predicted**).

```
truth = c("milk", "tea", "tea", "milk", "tea", "tea", "milk", "milk")
predicted = c("milk", "tea", "tea", "milk", "tea", "tea", "milk", "milk")
tea_mat = table(truth, predicted)
tea_mat
```

```
      predicted
truth  milk tea
  milk    4   0
  tea     0   4
```

```
chisq.test(tea_mat, correct = FALSE)
```

```
        Pearson's Chi-squared test

data:  tea_mat
X-squared = 8, df = 1, p-value = 0.004678
```

Fisher's exact test

- The χ^2 approximation for the test statistic is only reasonable when n is sufficiently large.

- I.e. we need the expected cell frequencies to all be 5 or more.
- If this is not the case, then we need to take care and maybe consider **exact** tests, i.e. calculating the exact p-value for the test statistic.
- All that we really need to assume is that we have independent observations (and we condition on knowing the row and column totals).
- In R the function `fisher.test()` is available to carry out these calculations both for 2×2 tables and general contingency tables.

Formulating the null hypothesis

The simplest exact test for contingency tables is Fisher's test for 2×2 tables. Consider the table:

	A_1	A_2	**Total**
B_1	y_{11}	y_{12}	$y_{1\bullet}$
B_2	y_{21}	y_{12}	$y_{2\bullet}$
Total	$y_{\bullet 1}$	$y_{\bullet 2}$	n

Let θ be the odds ratio.
Fisher's exact test can be thought of as testing the null hypothesis,

$$H_0 : \theta = 1$$

against the alternative $H_1 : \theta > 1$ or $H_1 : \theta < 1$ or $H_1 : \theta \neq 1$.
The test is based on the observed value of y_{11} given the marginal totals.
The **test statistic** is Y_{11} and the **observed test statistic** is y_{11}.

The hypergeometric distrubution

For a 2×2 if we know the row and column and y_{11} then the table is completely specified.
If H_0 is true and we know the $y_{1\bullet}, y_{\bullet 1}$ and n values we expect $y_{\bullet 1} \times \frac{y_{1\bullet}}{n}$ in the $(1,1)$th cell.
To obtain the distribution of y_{11} given the marginal values we note that this is like selecting $y_{1\bullet}$ values from n where $y_{1\bullet}$ and ype B_1 and $y_{2\bullet}$ are type B_2.
This motivates the use of the hypergeometric distribution:

$$P(Y_{11} = y_{11}) = \frac{\binom{y_{1\bullet}}{y_{11}}\binom{y_{2\bullet}}{y_{\bullet 1} - y_{11}}}{\binom{n}{y_{\bullet 1}}} e.$$

p-values

To calculate the p-value for a particular table we need to:

- enumerate all tables, as extreme, or more extreme than the observed table **with the same marginal totals**; and
- sum up the probability of each of these tables.

Lady tasting tea

```
truth = c("milk", "tea", "tea", "milk", "tea", "tea", "milk", "milk")
predicted = c("milk", "tea", "tea", "milk", "tea", "tea", "milk", "milk")
tea_mat = table(truth, predicted)
tea_mat
```

```
      predicted
truth  milk tea
  milk    4   0
  tea     0   4
```

For Fisher's exact test we:

1. Consider all possible permutations of the 2×2 contingency table with the same *marginal totals* (in this case $y_{i\bullet} = y_{\bullet j} = 4$).
2. Calculate how many of these were equal to or" more extreme" than what we observed.

Truth	Predicted Milk	Tea	**Total**
Milk	4	0	$y_{1\bullet} = 4$
Tea	0	4	$y_{2\bullet} = 4$
Total	$y_{\bullet 1} = 4$	$y_{\bullet 2} = 4$	$y_{\bullet\bullet} = n = 8$

How do we define more extreme?

Let us define a test statistic

T = number of cups of tea before milk that she got correct.

This test statistic has 5 outcomes: $\{0, 1, 2, 3, 4\}$.
Given that there are 8 cups of tea, there are $\binom{8}{4} = 70$ ways that we could predict which cups had tea added before milk.
We can look at all 70 ways and calculate how often we see a test statistic of 0, 1, 2, 3 or 4.

Number correct: t_i	0	1	2	3	4	
How many ways: f_i	$\binom{4}{0}\binom{4}{4} = 1$	$\binom{4}{1}\binom{4}{3} = 16$	$\binom{4}{2}\binom{4}{2} = 36$	$\binom{4}{3}\binom{4}{1} = 16$	$\binom{4}{4}\binom{4}{0} = 1$	70
Corresponding probability: p_i	$\frac{1}{70}$	$\frac{16}{70}$	$\frac{36}{70}$	$\frac{16}{70}$	$\frac{1}{70}$	1

$P(\text{Getting at least 4 correct}) = P(T = 4) = \frac{1}{70} = 0.014$

```
fisher.test(tea_mat)
```

```
        Fisher's Exact Test for Count Data

data:  tea_mat
p-value = 0.02857
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.339059      Inf
sample estimates:
odds ratio
       Inf
```

```
fisher.test(tea_mat, alternative = "greater") # correct in this setting
```

```
        Fisher's Exact Test for Count Data

data:  tea_mat
p-value = 0.01429
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 2.003768      Inf
```

```
sample estimates:
odds ratio
       Inf
```

Cancer of the larynx

Mendenhall et al. (1984) report the results of a study comparing radiation therapy with surgery in treating cancer of the larynx.

	Cancer controlled	Cancer bot controlled	Total
Surgery	21	2	23
Radiation therapy	15	3	18
Total	36	5	41

Suppose that we wish to test $H_0 : \theta = 1$ (both treatments equally effective) against $H_1 : \theta > 1$ (surgery more effective).
First we need to enumerate all tables which are as extreme or more extreme than the observed table. These are:
The original table:

	Cancer controlled	Cancer bot controlled	Total
Surgery	21	2	**23**
Radiation therapy	15	3	**18**
Total	**36**	**5**	**41**

And the tables where surgery controlled more than 21 (i.e. 22 or 23) patients, holding the **margins** constant.

	Cancer controlled	Cancer bot controlled	Total
Surgery	22	1	**23**
Radiation therapy	15	3	**18**
Total	**36**	**5**	**41**

	Cancer controlled	Cancer bot controlled	Total
Surgery	23	0	**23**
Radiation therapy	15	3	**18**
Total	**36**	**5**	**41**

Let X be the number of surgery cases where cancer is controlled. Applying Fisher's

approach

$$\begin{aligned}
\text{p-value} &= P(X \geqslant 21 \mid \text{marginal totals}) \\
&= P(X = 21, 22, 23 \mid \text{marginal totals}) \\
&= P(X = 21 \mid \text{marginal totals}) \\
&\quad + P(X = 22 \mid \text{marginal totals}) \\
&\quad + P(X = 23 \mid \text{marginal totals}) \\
&= \frac{\binom{23}{21}\binom{18}{15}}{\binom{41}{36}} + \frac{\binom{23}{22}\binom{18}{14}}{\binom{41}{36}} + \frac{\binom{23}{23}\binom{18}{13}}{\binom{41}{36}} + \\
&= 0.3808.
\end{aligned}$$

```
y_mat = matrix(c(21, 15, 2, 3), ncol = 2)
colnames(y_mat) = c("Controlled", "Not controlled")
rownames(y_mat) = c("Surgery", "Radiation therapy")
y_mat
```

```
                 Controlled Not controlled
Surgery                  21              2
Radiation therapy        15              3
```

```
fisher.test(y_mat, alternative = "greater")
```

```

        Fisher's Exact Test for Count Data

data:  y_mat
p-value = 0.3808
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 0.2864828       Inf
sample estimates:
odds ratio
  2.061731
```

Drawbacks

Why don't we use Fisher's exact test all the time?

- The calculation of the p-value requires conditioning on row and column margins being fixed.
- Computationally difficult for large samples.
- It can be generalized to $r \times c$ two-way contingency tables but is very difficult to compute. Generally requires use of Monte Carlo (i.e. random permutation).

X Y	y_1	y_2	y_3	**Row total**
x_1	a	b	c	$a+b+c$
x_2	d	e	f	$d+e+f$
Row total	$a+d$	$b+e$	$c+f$	$n = a+b+c+d+e+f$

9.2 Yates' chi-squared test

Yates' Corrected χ^2 Test

Yates (1934) modified the standard chi-squared test with a continuity correction. It is usually more accurate when counts in each cell are small. Yates' statistic for 2×2

tables is:

$$T = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(|Y_{ij} - e_{ij}| - 0.5)^2}{e_{ij}}$$

which approximately follows a χ^2_1 distribution under H_0.

Logic behind continuity corrections In general, if we have an *integer-valued random variable* X which we would like to approximate with a continuous random variable Y then

$$P(X \leqslant x) \approx P(Y \leqslant x + 0.5) \quad \text{and} \quad P(X \geqslant x) \approx P(Y \geqslant x - 0.5).$$

```
chisq.test(tea_mat, correct = TRUE)
```

```
Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
    incorrect
```

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  tea_mat
X-squared = 4.5, df = 1, p-value = 0.03389
```

9.3 Permutation testing

Fingerprints

The study of Galton (1892) marked one of the first formal statistical examinations of association for contingency tables. His work involved determining the association of fingerprint characteristics of 105 fraternal (or dizygotic) male twins. One male twin was "earmarked" as twin A and his brother was "earmarked" as twin B. For each twin, the number of Arches, Loops and Whorls was counted and summarised in the 3×3 contigency table of Galton (1892, Table XXII, pg. 175).

TABLE XXII.

Observed Fraternal Couplets.

B children.	A children.			Totals in B children.
	Arches.	Loops.	Whorls.	
Arches . . .	5	12	2	*19*
Loops . . .	4	42	15	*61*
Whorls . . .	1	14	10	*25*
Totals in A children	*10*	*68*	*27*	105

```
galton.dat ← matrix(c(5, 4, 1, 12, 42, 14, 2, 15, 10), 3, 3)
rownames(galton.dat) = c("Arches-B", "Loops-B", "Whorls-B")
colnames(galton.dat) = c("Arches-A", "Loops-A", "Whorls-A")
galton.dat
```

```
         Arches-A Loops-A Whorls-A
Arches-B        5      12        2
Loops-B         4      42       15
Whorls-B        1      14       10
```

```
chisq.test(galton.dat)
```

```
Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
    incorrect
```

```
        Pearson's Chi-squared test

data:  galton.dat
X-squared = 11.17, df = 4, p-value = 0.02472
```

Monte Carlo simulation

The **Monte Carlo simulation procedure** is as follows:

- Analyse the sample as one would normally do in a hypothesis test (up to, and including, the calculation of the test statistic)
- From the original sample being analysed, resample it LOTS of times
- The test statistic of interest is calculated for each of the resamples (so that we have the sampling distribution of the test statistic)
- This leads to LOTS of test statistics that will be used to calculate p-values for the observed statistic.

Monte Carlo p-values are calculated by determining the proportion of the resampled test statistics as or more extrame than the observed test statistic. **No assumptions** are made about the underlying distribution of the population.

Fingerprints

Monte Carlo p-values may be obtained by randomly generating contingency tables given that the margins are assumed fixed.
To randomly generate a contingency table with the same margins as the original table we use the `r2dtable()` function in R.
`r2dtable()` generates a **list** of random 2-way tables given marginals.

```
row_totals = rowSums(galton.dat)
col_totals = colSums(galton.dat)
B = 10000
set.seed(123)
x_list = r2dtable(n = B,
                  r = row_totals,
                  c = col_totals)
x_list[[1]]
```

```
     [,1] [,2] [,3]
[1,]    2   10    7
[2,]    7   43   11
[3,]    1   15    9
```

```
chisq.test(x_list[[1]])
```

```
Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
    incorrect
```

```
        Pearson's Chi-squared test

data:  x_list[[1]]
X-squared = 5.2367, df = 4, p-value = 0.2639
```

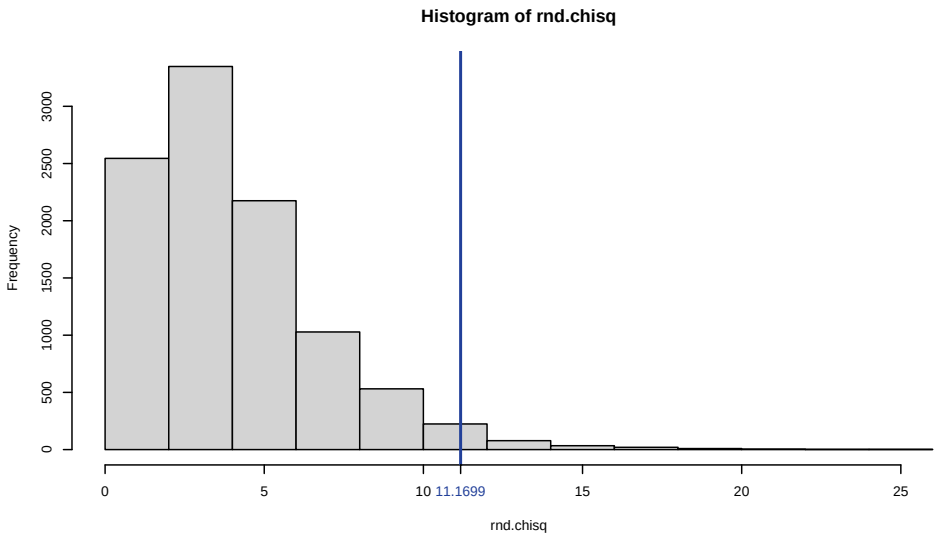
The observed test statistic for the first random sample is 5.24.
For each of the 10,000 randomly generated contingency tables, we can record their test statistic then determine what proportion of them are equal to (or exceed) the observed test statistic.

```
rnd.chisq = numeric(B) # initialise an empty vector
for (i in 1:B){ # loop over B iterations
  # each time save the test statistic
  rnd.chisq[i] = chisq.test(x_list[[i]])$statistic
}
# what proportion of times did we observe a test statistic
# as or more extreme than what we observed?
sum(rnd.chisq ⩾ 11.1699)/B
```

```
[1] 0.022
```

Here, the Monte Carlo p-value is 0.022 (comparable to theoretical p-value of 0.0247).

```
par(cex = 0.6)
hist(rnd.chisq)
abline(v = 11.1699, col = "#224099", lwd = 2)
axis(1, 11.1699, col.axis = "#224099")
```



`chisq.test(..., simulate.p.value = TRUE)` calculates Monte Carlo p-values and does so using `r2dtable()` internally.

```
chisq.test(galton.dat, simulate.p.value = TRUE)
```

```
        Pearson's Chi-squared test with simulated p-value (based on 2000
        replicates)

data:  galton.dat
X-squared = 11.17, df = NA, p-value = 0.01999
```

The default is 2000 contingency tables. We could use 10,000 resamples by specifying `B = 10000`.

```
chisq.test(galton.dat, simulate.p.value = TRUE, B = 10000)
```

```
        Pearson's Chi-squared test with simulated p-value (based on 10000
        replicates)

data:  galton.dat
X-squared = 11.17, df = NA, p-value = 0.0232
```

10
Testing Means

10.1 General t-test background

What is it?

Some basic probability facts about samples from *normal populations* will prove useful.

1. The sample mean from a normal sample is itself normally distributed.
2. The sample variance from a normal sample has a *scaled* χ^2 distribution.
3. The sample mean and sample variance from a normal sample are *statistically independent*.
4. If $Z \sim \mathcal{N}(0,1)$ is independent of a χ^2_d random variable, the quantity

$$\frac{Z}{\sqrt{\chi^2_d/d}} \sim t_d,$$

   a t-distribution with d degrees of freedom.

The t-statistic

- Given a population mean μ, the sample mean and variance $\overline{X}$ and S^2, the ratio

$$\frac{\overline{X}-\mu}{S/\sqrt{n}} = \frac{\sqrt{n}(\overline{X}-\mu)/\sigma}{S/\sigma} \sim t_{n-1}.$$

  - the numerator is $\mathcal{N}(0,1)$;
  - the denominator is $\sqrt{\chi^2_{N-1}/n-1}$, independent of the numerator.
- In many statistical applications we have a model whereby a certain statistic has this general form:
  - some estimator of some parameter is normally distributed;
  - a standard error based on the data has a distribution like $\sqrt{\chi^2_d/d}$ times the true SD of the estimator (for some d) and is *independent* of the estimator;
  - then the ratio $\frac{\text{estimator}-\text{true value}}{\text{standard error}} \sim t_d$.

10.2 One sample t-test

Beer contents

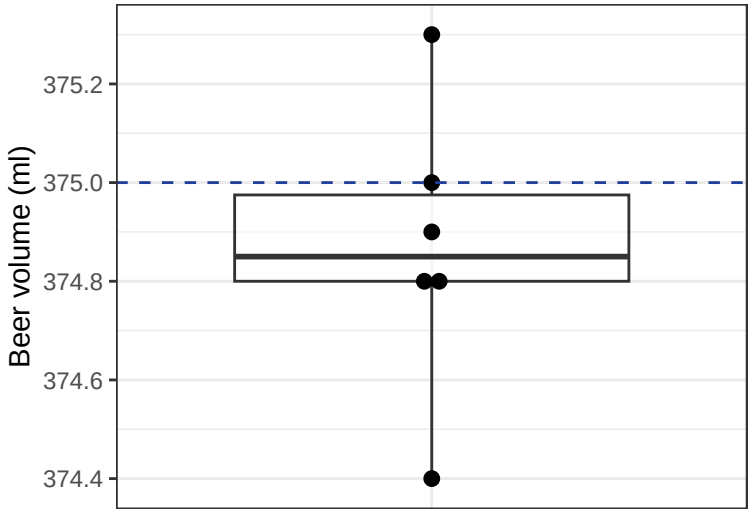
Beer contents in a pack of six bottles (in millilitres) are:

```
y = c(374.8, 375.0, 375.3, 374.8, 374.4, 374.9)
```

Is the mean beer content less than the 375 ml claimed on the label?

```
library("ggplot2")
df = data.frame(y)
set.seed(88)
ggplot(df, aes(x = "", y = y)) +
  geom_boxplot(alpha = 0.5, coef = 10) +
  geom_dotplot(binaxis = 'y', stackdir = 'center') +
  geom_hline(yintercept = 375, colour = "#224099", linetype = "dashed") +
  labs(y = "Beer volume (ml)", x = "") +
  theme_bw() +
```

```
theme(axis.ticks.x = element_blank(), axis.text.x = element_blank())
```



Hypothesis testing

Workflow: General hypothesis test

- **Hypotheses**: $H_0 : \theta = \theta_0$ vs $H_1 : \theta > \theta_0$ or $\theta < \theta_0$ or $\theta \neq \theta_0$
- **Assumptions**: $X_1, X_2, \cdots, X_n \sim F_\theta$
- **Test Statistic**: $T = f(X_1, X_2, \cdots, X_n)$
- **Observed Test Statistic**: $t_0 = f(x_1, x_2, \ldots, x_n)$.
- **Significance**: p-value = $P(T \geqslant t_0)$ or $P(T \leqslant t_0)$ or $2P(T \geqslant |t_0|)$
- **Decision**: If the p-value is less than α, there is ecidence against H_0.

Hypothesis

- The statement against which you search for evidence is called the null hypothesis, and is denoted by H_0. It is generally a "no difference" statement.
- The statement you claim is called the alternative hypothesis, and is denoted by H_1.

$$H_0 : \theta = \theta_0 \quad \text{vs} \quad \begin{cases} H_1 : \theta > \theta_0 & \text{(upper-side alternative)} \\ H_1 : \theta < \theta_0 & \text{(lower-side alternative)} \\ H_1 : \theta \neq \theta_0 & \text{(two-sided alternative)} \end{cases}$$

Assumptions

- Each observation $X_1, X_2, \ldots, X_n$ is chosen at random from a population.
- We say that such random variables are *idd* (independently and identically distributed).
- Each test we consider will have its own assumptions.

Test statistic

- Since observations X_i vary from sample to sample we can never be sure whether H_0 is true or not.
- We use a test statistic $T = f(X_1, \ldots, X_n)$ to test if the data are consistent with H_0 such that the distribution of T is known assuming H_0 is true.

The **observed test statistic**, t_0, is where we plug our observed data into the formula for the test statistic.
Large (positive or negative depending on H_1) observed test statistic values is taken as evidence of poor agreement with H_0

Significance

The p-value is defined as the probability of getting a test statistic, T, *as or more extreme* than the value we observed, t_0, *assuming* that H_0 is true.
Typical p-value statements:

- For $H_1 : \theta > \theta_0$, p-value = $P(T \geqslant t_0)$
- For $H_1 : \theta < \theta_0$, p-value = $P(T \leqslant -t_0)$
- For $H_1 : \theta \neq \theta_0$, p-value = $2P(T \geqslant |t_0|)$

Decision

An observed *large* positive or negative value of t_0 and hence small p-value is taken as evidence of poor agreement with H_0.

- If the p-value is small, then either H_0 is true and the poor agreement is due to an unlikely event, or H_0 is false. Therefore…
- the smaller the p-value, the stronger the evidence against H_0 in favour of H_1.
- A large p-value does not mean that there is evidence that H_0 is true
- The level of significance, α, is the strength of evidence needed to reject H_0 (often $\alpha = 0.05$)

One sample t-test

Suppose we have a sample $X_1, X_2, \ldots, X_n$ of size n drawn from a normal population with an unknown variance σ^2. Let $x_1, x_2, \ldots, x_n$ be the observed values. We want to test the population mean μ.

Workflow: One Sample t-test

- **Hypotheses**: $H_0 : \mu = \mu$ vs $H_1 : \mu > \mu_0, \mu < \mu_0$ or $\mu \neq \mu_0$
- **Assumptions**: X_i are iid random variables and follow $\mathcal{N}(\mu, \sigma^2)$
- **Test Statistic**: $T = \frac{\overline{X}-\mu_0}{S/\sqrt{n}}$. Under $H_0, T \sim t_{n-1}$
- **Observed Test Statistic**: $t_0 = \frac{\overline{x}-\mu_0}{s/\sqrt{n}}$.
- **p-value**: $P(t_{n-1} \geqslant t_0)$ or $P(t_{n-1} \leqslant t_0)$ or $2P(t_{n-1} \geqslant |t_0|)$
- **Decision**: Reject H_0 in favor of H_1 if the p-value is less then α.

Beer contents

Beer contents in a pack of six bottles (in millilitres) are:

$$374.8, 375.0, 375.3, 374.8, 374.4, 374.9$$

Is the mean beer content less than the 375 ml claimed on the label?

```
x = c(374.8 , 375.0 , 375.3 , 374.8 , 374.4 , 374.9)
mean(x)
```

```
[1] 374.8667
```

```
sd(x)
```

```
[1] 0.294392
```

- **Hypotheses**: $H_0 : \mu = 375$ vs $H_1 : \mu < 375$
- **Assumptions**: X_i are iid random variables and follow $\mathcal{N}(\mu, \sigma^2)$
- **Test Statistic**: $T = \frac{\overline{X}-\mu_0}{S/\sqrt{n}}$. Under $H_0, T \sim t_{n-1}$
- **Observed Test Statistic**: $t_0 = \frac{375.87-375}{0.29/\sqrt{6}} = -1.11$.
- **p-value**: $P(t_5 \leqslant -1.11) = 0.16$
- **Decision**: The data is consistent with the null hypothesis H_0.

```
t.test(x, mu = 375, alternative = "less")
```

```
        One Sample t-test

data:  x
t = -1.1094, df = 5, p-value = 0.1589
alternative hypothesis: true mean is less than 375
95 percent confidence interval:
     -Inf 375.1088
sample estimates:
mean of x
 374.8667
```

10.3 Two sample t-test

What if you have two samples?

There are times that we want to test if the population means of two samples are different.
Here we are left with two possible scenarios

- Two independent samples
- Two related samples (dependent samples or repeated measures)

Smokers and blood platelet aggregation

Blood samples are taken from 11 smokers and 11 non-smokers to measure aggregation of blood platelets.

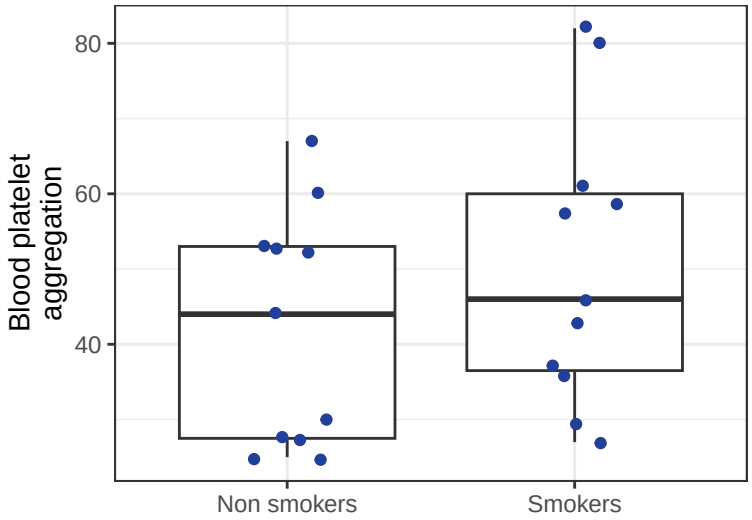
```
non_smokers = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
smokers =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
dat = data.frame(
  platelets = c(non_smokers, smokers),
  status = c(rep("Non smokers", length(non_smokers)),
  rep("Smokers", length(smokers)))
)
library(dplyr)
sum = dat |>
  group_by(status) |>
        summarise(Mean = mean(platelets),
        SD = sd(platelets),
        n = n())
knitr::kable(sum, digits = 1, booktabs=TRUE) |>
  kable_styling(position="center", latex_options = "hold_position")
```

status	Mean	SD	n
Non smokers	42.2	15.6	11
Smokers	50.6	18.9	11

Visualising Blood Platelet Aggregation

Is the aggregation of blood platelets affected by smoking?

```
library(ggplot2)
ggplot(dat) + aes(x = status, y = platelets) +
  geom_boxplot() +
  geom_jitter(width = 0.15, colour = "#224099") +
  labs(x = "", y = "Blood platelet\naggregation") +
theme_bw()
```



Workflow: Two sample t-test

- **Hypotheses**: $H_0 : \mu_x = \mu_y$ vs $H_1 : \mu_x > \mu_y, \mu_x < \mu_y$ or $\mu_x \neq \mu_y$
- **Assumptions**: $X_1, \ldots, X_{n_x}$ are iid $\mathcal{N}(\mu_X, \sigma^2)$, $Y_1, \ldots, Y_{n_y}$ are idd $\mathcal{N}(\mu_Y, \sigma^2)$ and X_i's are independent of Y_i's.
- **Test Statistic**: $T = \frac{\overline{X}-\overline{Y}}{S_p\sqrt{\frac{1}{n_x}+\frac{1}{n_y}}}$ where $S_p^2 = \frac{(n_x-1)S_x^2+(n_{y-1})S_y^2}{n_x+n_y-2}$. Under $H_0, T \sim t_{n_x+n_y-2}$
- **Observed Test Statistic**: $t_0 = \frac{\overline{x}-\overline{y}}{s_p\sqrt{\frac{1}{n_x}+\frac{1}{n_y}}}$ where $s_p^2 = \frac{(n_x-1)s_x^2+(n_{y-1})s_y^2}{n_x+n_y-2}$.
- **p-value**: $P(t_{n_x+n_y-2} \geqslant t_0)$ or $P(t_{n_x+n_y-2} \leqslant t_0)$ or $2P(t_{n_x+n_y-2} \geqslant |t_0|)$.
- **Decision**: If the p-value is less than α, there is evidence against H_0. If the p-value is greater than α, the data are consistent with H_0.

Let X_i be the blood platelet aggregation levels for the ith non-smoker and Y_j the level for the jth smoker. Let μ_S and μ_N be the population mean platelet aggregation levels for smokers and non-smokers respectively.

- **Hypotheses**: $H_0 : \mu_S = \mu_N$ vs $H_1 : \mu_S \neq \mu_N$
- **Assumptions**: $X_1, \ldots, X_{n_x}$ are iid $\mathcal{N}(\mu_X, \sigma^2)$, $Y_1, \ldots, Y_{n_y}$ are idd $\mathcal{N}(\mu_Y, \sigma^2)$ and X_i's are independent of Y_i's.
- **Test Statistic**: $T = \frac{\overline{X}-\overline{Y}}{S_p\sqrt{\frac{1}{n_x}+\frac{1}{n_y}}}$ Under $H_0, T \sim t_{n_x+n_y-2}$
- **Observed Test Statistic**: $t_0 = \frac{50.6-42.2}{17.3\sqrt{\frac{1}{11}+\frac{1}{11}}} = 1.14$.
- **p-value**: $2P(t_{20} \geqslant |1.14|) = 0.27$.
- **Decision**: Large p-value so the data are consistent with H_0. There does not appear to be evidence that blood platelet aggregation levels are different in smokers.

```
t.test(smokers, non_smokers,
       alternative = "two.sided",
       var.equal = TRUE)
```

```
        Two Sample t-test

data:  smokers and non_smokers
t = 1.144, df = 20, p-value = 0.2661
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -6.961816 23.870907
sample estimates:
mean of x mean of y
 50.63636  42.18182
```

The equal variance assumption

- We assume that the two underlying normal populations *have the same variance*.
- In this example, does this seem reasonable?

Status	Mean	SD	n
Non Smokers	42.2	15.6	11
Smokers	50.6	18.9	11

- These are a little different: are they so different that the "equal underlying population variances" assumption is not reasonable?

The Welch two-sample t-test

- Welch developed an alternative test which does **not** assume equal population variances. If
  - the X_i's are $\mathcal{N}(\mu_X, \sigma_X^2)$ and
  - the Y_i's are $\mathcal{N}(\mu_Y, \sigma_Y^2)$ then the variance of the sample mean difference is

$$\text{Var}(\overline{X} - \overline{Y}) = \text{Var}(\overline{X}) + \text{Var}(\overline{Y}) = \frac{\sigma_X^2}{n_X} + \frac{\sigma_Y^2}{n_Y}$$

- The standard error, $\text{SE}(\overline{X}-\overline{Y})$, is obtained by plugging in the two sample variances and taking the square root (note: we do not need to compute a "pooled" estimate of the common variance!).
- This gives the *Welch Statistic*, $T = \frac{\overline{X}-\overline{Y}}{\sqrt{\frac{S_X^2}{n_x}+\frac{S_T^2}{n_y}}}$.

Workflow: Welch two sample t-test

- **Hypotheses**: $H_0 : \mu_x = \mu_y$ vs $H_1 : \mu_x > \mu_y, \mu_x < \mu_y$ or $\mu_x \neq \mu_y$
- **Assumptions**: $X_1, \ldots, X_{n_x}$ are iid $\mathcal{N}(\mu_X, \sigma^2)$, $Y_1, \ldots, Y_{n_y}$ are idd $\mathcal{N}(\mu_Y, \sigma^2)$ and X_i's are independent of Y_i's.
- **Test Statistic**: $T = \frac{\overline{X}-\overline{Y}}{\sqrt{\frac{S_X^2}{n_x}+\frac{S_T^2}{n_y}}}$ where S_x^2 and S_y^2 are the sample variance of the X and Y samples, respectively. Under $H_0, T \sim t_\nu$ approximately, where the degrees of freedom parameter, ν, is estimated from the data.
- **Observed Test Statistic**: $t_0 = \frac{\overline{x}-\overline{y}}{\sqrt{\frac{1}{n_x}+\frac{1}{n_y}}}$.
- **p-value**: $P(t_\nu \geqslant t_0)$ or $P(t_\nu \leqslant t_0)$ or $2P(t_\nu \geqslant |t_0|)$.

- **Decision**: If the p-value is less than α, there is evidence against H_0. If the p-value is greater than α, the data are consistent with H_0.

Welch statistic not a proper t-statistic

- Technically, this statistic is not a "usual" t-statistic since the denominator is not a scaled χ^2 independent of the numerator.
- However, the statistic still has an approximate t-distribution where the degrees of freedom is not necessarily a whole number, and is estimated from the data.
- R does this for us using `var.equal = FALSE` (which is the default):

```
t.test(smokers, non_smokers, alternative = "two.sided")
```

```
        Welch Two Sample t-test

data:  smokers and non_smokers
t = 1.144, df = 19.313, p-value = 0.2666
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -6.997031 23.906122
sample estimates:
mean of x mean of y
 50.63636  42.18182
```

10.4 Paired Samples t-test

Smoking and aggregation (paired)

Blood samples from 11 individuals **before and after** they smoked a cigarette are used to measure aggregation of blood platelets.

```
before = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
after =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
df = data.frame(before, after, difference = after - before)
df
```

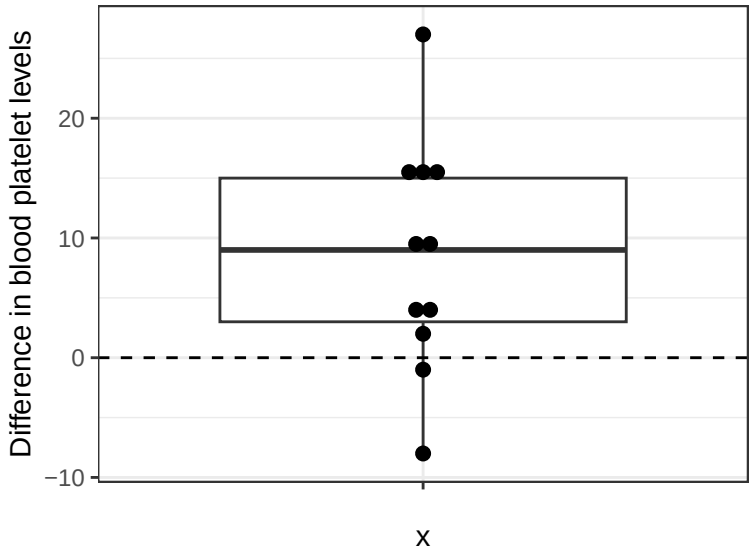
```
   before after difference
1      25    27          2
2      25    29          4
3      27    37         10
4      44    36         -8
5      30    46         16
6      67    82         15
7      53    57          4
8      53    80         27
9      52    61          9
10     60    59         -1
11     28    43         15
```

Is the aggregation affected by smoking?

```
df |>
summarise(across(.cols = c(before, after),
  .fns = list(Mean = mean,
  SD = sd,
  n = length))) |>
pivot_longer(cols = everything(),
  names_sep = "_",
  names_to = c("time", ".value"))
```

```
# A tibble: 2 x 4
  time    Mean    SD     n
  <chr>  <dbl> <dbl> <int>
1 before  42.2  15.6    11
2 after   50.6  18.9    11
```

```
ggplot(df) +
aes(x = "", y = difference) +
geom_boxplot() +
geom_dotplot(binaxis = "y", stackdir = "center") +
geom_hline(yintercept = 0, linetype = "dashed") +
labs(y = 'Difference in blood platelet levels')+
theme(axis.title.x = element_blank(),
  axis.text.x = element_blank(),
  axis.ticks.x = element_blank()) +
theme_bw()
```




```
t.test(df$after, df$before, paired = TRUE)
```

```

        Paired t-test

data:  df$after and df$before
t = 2.9065, df = 10, p-value = 0.01566
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
  1.97332 14.93577
sample estimates:
mean difference
       8.454545
```

```
t.test(df$difference)
```

```

        One Sample t-test

data:  df$difference
t = 2.9065, df = 10, p-value = 0.01566
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  1.97332 14.93577
sample estimates:
mean of x
```

```
8.454545
```

Let X_i and Y_i be the blood platelet aggregation levels for the i^{th} person before and after smoking, respectively. Define the change in person i's platelet aggregation as $D_i = Y_i - X_i$ and the popultion mean change in platelet aggregation as μ_d

- **Hypotheses**: $H_0 : \mu_d = 0$ vs $H_1 : \mu_d \neq 0$
- **Assumptions**: D_i are iid $\mathcal{N}(\mu_d, \sigma^2)$.
- **Test Statistic**: $T = \frac{\overline{D}}{\sqrt{S_d}/\sqrt{n}}$. Under $H_0, T \sim t_{n-1}$
- **Observed Test Statistic**: $t_0 = \frac{8.45}{9.64/\sqrt{11}} = 2.91$.
- **p-value**: $2P(t_{10} \geqslant |2.91|) = 0.016$.
- **Decision**: Small p-value so we have to reject the null hypothesis. There is evidence that blood platelet aggregation levels change after smoking.

11
Critical values, rejection regions and confidence intervals

11.1 Random variables basics

- A random variable can be thought of as a mathematical object which takes certain values with certain probabilities.
- We have *discrete* and *continuous* random variables, although we can always "approximate" a continuous one with a discrete one (taking values on a suitably fine grid).
- A simple discrete random variable X can be described as a single random draw from a "box" containing tickets, each with numbers written on them.
- In this case,
  - $\mathrm{E}(X) = \mu$ (the average of the numbers in the box),
  - $\mathrm{Var}(X) = \sigma^2$ (the *population variance* of the numbers in the box), and
  - $\mathrm{SD}(X) = \sigma$.

Random sample with replacement

- Next, consider taking a random sample of size n *with Replacement*, denote the values $X_1, X_2, \ldots, X_n$.
- This means, one of *all possible samples of size n* is chosen in such a way that each is equally likely.
- If there are N tickets in the box, how many such samples are there?
- It turns out that these X_i's are *independent and identically distributed*. This means
  - each X_i has the same distribution as a single draw,
  - the X_i's are all mutually independent.
- Consider now taking the total $T = \sum_{i=1}^{n} X_i$.

Expectation of sums of random variables

The **expectation** of a sum is *always* the sum of the expectations.
Recall if $\mathrm{E}(X) = \mu$,

$$\begin{aligned}\mathrm{E}(T) &= \mathrm{E}(X_1 + \cdots + X_n)\\ &= \mathrm{E}(X_1) + \cdots + \mathrm{E}(X_n)\\ &= \underbrace{\mu + \cdots + \mu}_{n \text{ terms}}\\ &= n\mu\end{aligned}$$

Multiplying by a constant: for any random variable X and any constant c,

$$\mathrm{E}(cX) = c\mathrm{E}(X)$$

Variance of sums of random variables

The variance of a sum is *not always* the sum of the variances.
However, it *is* if the X_i's are *independent*. So,

$$\begin{aligned}\mathrm{Var}(T) &= \mathrm{Var}(X_1 + \cdots + X_n)\\ &= \mathrm{Var}(X_1) + \cdots + \mathrm{Var}(X_n)\\ &= \underbrace{\sigma^2 + \cdots + \sigma^2}_{n \text{ terms}}\\ &= n\sigma^2.\end{aligned}$$

Multiplying by a constant: for any random variable X and any constant c,

$$\mathrm{Var}(cX) = c^2\,\mathrm{Var}(X)$$

Sample mean

Consider the sample mean $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i = \frac{1}{n}T$.
What is $\mathrm{E}(\overline{X})$? What is $Var(\overline{X})$?
Since $\overline{X} = \frac{1}{n}T$,

$$\mathrm{E}(\overline{X}) = \mathrm{E}\left(\frac{1}{n}T\right) = \frac{1}{n}\mathrm{E}(T) = \frac{1}{n}n\mu = \mu$$

$$\mathrm{Var}(\overline{X}) = \mathrm{Var}\left(\frac{1}{n}T\right) = \frac{1}{n}^2\,\mathrm{Var}(T) = \frac{1}{n^2}n\sigma^2 = \frac{\sigma^2}{n}$$

Estimating μ

- In many applications, we model data $x_1, \ldots, x_n$ as values taken by such a sample $X_1, \ldots, X_n$ and we are interested in "estimating" or "learning" μ (which is an "unknown population mean").
- In this case the *estimator* is the sample mean $\overline{X}$ (regarded as a *random variable*).
- The *estimate* is $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, the observed value of the mean of the data (this is *conceptually different* to $\overline{X}$!!)
- An important theoretical quantity is the *standard error*, the *standard deviation of the estimator*:
$$\mathrm{SE} = \mathrm{SD}(\overline{X}) = \sqrt{\mathrm{Var}(\overline{X})} = \frac{\sigma}{\sqrt{n}}$$
- The standard error is (in general) also an *unknown parameter*.

Importance of the standard error

- The standard error (standard deviation of the estimator) is important to know, since it tells us the "likely size of the estimation error".
- An estimate on its own is not very useful, we need to also know how accurate or reliable the estimate is.
  - This is what the standard error provides.
- *Unfortunately* in most contexts the standard error is also *unknown*,
  - but we can usually (also) estimate the standard error!

Estimating the standard error

- The standard error (at least when estimating a population mean μ) involves the (usually unknown) population variance σ^2:

$$\mathrm{SE} = \frac{\sigma}{\sqrt{n}}$$

- Fortunately, we can usually estimate σ^2 using the *sample variance*

$$\mathrm{S}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

- The corresponding estimated standard error is

$$\widehat{\mathrm{SE}} = \frac{s}{\sqrt{n}}$$

where $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$. is the observed value of the sample variance.

11.2 Critical values and confidence intervals

More precise inference

- Usually, we want to know if a given value μ_0 is a "plausible value" for the unknown μ, based on observed data $x_1, \ldots, x_n$.
- Roughly speaking, we do this by
  1. computing the value of the estimate $\overline{x}$,
  2. computing the value of the *estimated standard error* $\frac{s}{\sqrt{n}}$,
  3. seeing if the discrepancy $\overline{x} - \mu_0$ is "large" compared to the standard error.
- The various procedures we look at:
  - t-tests (with corresponding p-values)
  - confidence intervals
  - rejection regions

  are all variations on this single idea.

What kind of discrepancies are of interest?

- We need to have it very clear in our minds which kind of discrepancies $\overline{x} - \mu_0$ we are interested in:
  - positive
  - negative
  - both
- Another way to think about it is, given a fixed μ_0 of interest and an observed sample mean $\overline{x}$, which of the following questions we are asking:
  1. Is $\overline{x}$ significantly *more* then μ_0? (*one-sided*)

2. Is $\overline{x}$ significantly *less* then μ_0? (*one-sided*)
3. Is $\overline{x}$ significantly *different* then μ_0? (*two-sided*)

Beer example

Beer contents in a pack of six bottles (in millilitres) are:

$$374.8, 375.0, 375.3, 374.8, 374.4, 374.9$$

> Does the mean beer content differ from the 375 ml claimed on the label?

```
x = c(374.8, 375.0, 375.3, 374.8, 374.4, 374.9)
mean(x)
```

```
[1] 374.8667
```

```
sd(x)
```

```
[1] 0.294392
```

- In the beer example there are different possible points of view.
- For *consumers*, the results will only be "interesting" if $\overline{x}$ is significantly *less* than 375:
  - in this case the company is "ripping consumers off".
- However for the *beer* producers, both positive and negative discrepancies might be of interest:
  - if they are *underfilling*, consumers will be unhappy,
  - if they are *overfilling*, they are "wasting" some of their product.
- Thus both a *one-sided* and *two-sided* point of view are conceivable even for this example.

Two-sided discrepancies of interest

When two-sided discrepancies are of interest we are basically asking: for a given μ_0, is the *absolute value* $|\overline{x} - \mu_0|$ large, compared to the standard error $\frac{s}{\sqrt{n}}$?

t-test Approach

Declare hypothesised mean value μ_0 **not plausible** if

$$|\overline{x} - \mu_0| > c\frac{s}{\sqrt{n}}$$

for some suitably chosen c.

Confidence Interval Approach

Set of **plausible values** for the unknown μ is

$$\overline{x} \pm c\frac{s}{\sqrt{n}},$$

for some suitably chosen c.

> **If the same c is chosen in both approaches, the set of plausible values *is the same*, i.e. μ_0 in the confidence interval $\Leftrightarrow |\overline{x} - \mu_0| \leqslant \frac{cs}{\sqrt{n}}$**

How to choose the constant c?

- The constant c can be chosen in a sensible way in each context.
- **Testing:** control the **false alam rate**.

- **Confidence Intervals**: control the **coverage probability**.
  - the coverage probability is commonly also called the confidence level and expressed as a percentage.

False alarm rate

- A "**false alarm**" is when we "reject incorrectly".
- Using our current language it is when we "reject a given value μ_0" when we shouldn't
- That is, we declare μ_0 "not plausible" when in fact it is the true value!
- We pick choose small $0 \leqslant \alpha \leqslant 1$ for the desired "**false alarm rate**" e.g. 0.05, 0.1.
- Choose c such that (if possible)

$$P\left(|\overline{X} - \mu_0| > c\frac{S}{\sqrt{n}}\right) = \alpha$$

- If this isn't possible then just try to ensure that this probability does not exceed α!
- The **false alarm rate** is also called the **significance level**.

Normal population: use the t-distribution

Under the special statistical model where the data are modelled as iid normal random variables, we know that *if the true population mean is indeed* $\mathbf{mu}_0$ then the ratio

$$\frac{\overline{X} - \mu_0}{S/\sqrt{n}} \sim t_{n-1},$$

and therefore we choose c such that,

$$P\left(|\overline{X} - \mu_0| > c\frac{S}{\sqrt{n}}\right) = P\left(\frac{|\overline{X} - \mu_0|}{S/\sqrt{n}} > c\right) = P(|t_{n-1}| > c) = \alpha$$

Finding quantiles in R

In R, we get quantiles using the `qDISTRIBUTION()` range of functions, for example, `qt(p, df = k)`, `qnorm(p)`, `qchisq(p, df = k)` for normal and χ^2 distributions respectively.

```
qt(0.05, df = 5)
```

```
[1] -2.015048
```

```
qnorm(0.05)
```

```
[1] -1.644854
```

Using `qt()`

Note that if $P(|t_{n-1}| > c) = \alpha$ then

$$P(|t_{n-1}| \leqslant c) = P(-c \leqslant t_{n-1} \leqslant c) = 1 - \alpha$$

and

$$P(t_{n-1} < -c) + P(t_{n-1} > c) = 2P(t_{n-1} \leqslant c) = 1 - \frac{\alpha}{2}.$$

For example,

- $\alpha = 0.05$, we need c such that $P(t_{n-1} \leqslant c) = 1 - 0.025 = 0.975$ so we use `c = qt(0.975, df = n-1)`
- $\alpha = 0.1$, we need c such that $P(t_{n-1} \leqslant c) = 1 - 0.005 = 0.995$ so we use `c = qt(0.995, df = n-1)`

Beer example

- Recall we have observations

```
x = c(374.8, 375.0, 375.3, 374.8, 374.4, 374.9)
```

- Here the sample size $n = 6$ so if
  - $\alpha = 0.05$ we neec c such that $P(t_5 < c) = 0.975$,
  - $\alpha = 0.1$ we neec c such that $P(t_5 < c) = 0.995$
- These are given by

```
qt(0.975, df = 5)
```

```
[1] 2.570582
```

```
qt(0.995, df = 5)
```

```
[1] 4.032143
```

- The sample mean is

```
xbar = mean(x)
xbar
```

```
[1] 374.8667
```

- The standard error is

```
se = sd(x)/sqrt(6)
se
```

```
[1] 0.120185
```

- The discrepancy from the "given value" 375 is

```
discrep=abs(xbar-375)
discrep
```

```
[1] 0.1333333
```

- This is only slightly more than 1 (estimated) standard error.
- We need it to be at least 2.57 standard errors to "reject at the 0.05 **false alarm rate**":
- Therefore we cannot reject H_0, so 375 is a plausible value (in this two-sided sense).

Coverage probability

- For a **confidence interval**, the **coverage probability** is simply the probability that the "true" value of the unknown parameter lies inside (is "covered by") the **confidence interval**.
- This is a *long run property* and should be interpreted in the context of *repeated experiments*.
- We choosea (small) *non-coverage* probability α, say 0.05 or 0.01,
  - then the **coverage probability** is $1 - \alpha$.
- Thus, under some statistical model we choose c so tht the **coverage probability** *under the model* satisfies (with μ the *true population mean*):

$$P\left(\overline{X} - c\frac{S}{\sqrt{n}} \leqslant \mu \leqslant \overline{X} + c\frac{S}{\sqrt{n}}\right) = P\left(|\overline{X} - \mu| \leqslant c\frac{S}{\sqrt{n}}\right) = 1 - \alpha$$

Equivalent to false alarm rate condition for t-test

- The **coverage probability** condition on the previous slide is an equivalent statement to the **false alarm rate** condition for the t-test (for the same α).
- Thus if the desired **coverage probability** is
  - 0.95 (i.e. non-coverage probability $\alpha = 0.05$) then we need c such that

$$P(t_{n-1} \leqslant c) = 1 = 0.025 = 0.975$$

  - 0.99 (i.e. non-coverage probability $\alpha = 0.1$) then we need c such that

$$P(t_{n-1} \leqslant c) = 1 = 0.005 = 0.995$$

Beer example

For a 95% confidence interval for μ we thus choose c via

```
c_95 = qt(0.975, df = 5)
c_95
```

```
[1] 2.570582
```

giving

```
xbar + c(-1,1) * c_95 * se
```

```
[1] 374.5577 375.1756
```

Note that this includes the "special value" 375 and so is consistent with our 0.05 **false-alarm rate** test earlier.

For a 99% confidence interval for μ we thus choose c via

```
c_99 = qt(0.995, df = 5)
c_99
```

```
[1] 4.032143
```

giving

```
xbar + c(-1,1) * c_99 * se
```

```
[1] 374.3821 375.3513
```

As we'd expect, this CI is wider, and also includes 375.

Using `t.test`

Compare our 'manual' computations above with the output of the R function `t.test()`:

First the default:

```
t.test(x, mu = 375)
```

```
        One Sample t-test

data:  x
t = -1.1094, df = 5, p-value = 0.3177
alternative hypothesis: true mean is
    not equal to 375
95 percent confidence interval:
 374.5577 375.1756
sample estimates:
mean of x
 374.8667
```

Setting `conf.level-0.99`

```
t.test(x, mu = 375, conf.level = 0.99)
```

```
        One Sample t-test

data:  x
t = -1.1094, df = 5, p-value = 0.3177
alternative hypothesis: true mean is
    not equal to 375
99 percent confidence interval:
 374.3821 375.3513
sample estimates:
mean of x
 374.8667
```

Note the default in R is *two-sided*.

One-sided discrepancies of interest

- The "two-sided" approach just outlined would be of interest to the beer producers, but not necessarily the beer consumers.
- Let us consider the point of view of the consumers now.
- t-test approach: declare (for some suitable value of c),

  - μ_0 not plausible if $\overline{x} - \mu_0 < -c\frac{s}{\sqrt{n}} \Leftrightarrow \overline{x} < \mu_0 - c\frac{s}{\sqrt{n}}$
- **Confidence interval** approach: set of plausible values for the unknown μ are those "not too much bigger than $\overline{x}$" i.e.

$$\left(-\infty, \overline{x} + c\frac{s}{\sqrt{n}}\right]$$

for a "suitably chosen" constant c
  - the upper endpoint is sometimes called an "upper confidence limit"
  - it can be interpreted as "the largest value consistent with the data"

Same set of plausible values

- Again, note that for the same c these two approaches give the *same set of plausible values* for μ:
  - μ_0 is in the one-sided **confidence interval** $\Leftrightarrow \overline{x} \geqslant \mu_0 - c\frac{s}{\sqrt{n}}$.

Controlling the (one-sided) false alarm rate

- We use a similar approach to the two-sided case, but with a crucial difference!
- Under the iid normal model, $T = \frac{\overline{X}-\mu}{S/\sqrt{n}} \sim t_{n-1}$
- We choose c so that if μ_0 is the true value,

$$P\left(\overline{X} \leqslant \mu_0 - c\frac{S}{\sqrt{n}}\right) = P\left(\frac{\overline{X} - \mu_0}{S/\sqrt{n}}\right) = P(t_{n-1} < -c) = \alpha$$

- By symmetry we also have $P(t_{n-1} > c) = \alpha$ or $P(t_{n-1} < -c) = 1 - \alpha$
- For **false alarm rate**
  - 0.05 we need c such that $P(t_{n-1} \leqslant c) = 1 - 0.05 = 0.95$,
  - 0.01 we need c such that $P(t_{n-1} \leqslant c) = 1 - 0.01 = 0.99$.

Beer example

For the $\alpha = 0.05$ **false alarm rate**, since $n = 6$ we need

```
c_05 = qt(.95, df = 5)
c_05
```

```
[1] 2.015048
```

Note this is *smaller* than the two-sided version. We have already seen that the discrepancy is only slightly more than 1 standard error:

```
c(xbar - 375, se)
```

```
[1] -0.1333333  0.1201850
```

For the $\alpha = 0.01$ **false alarm rate**, since $n = 6$ we need

```
c_01 = qt(.99, df = 5)
c_01
```

```
[1] 3.36493
```

Note that this is also smaller than the two-sided version.
This makes the one-sided tests "more sensitive" than the two-sided versions.

One-sided confidence intervals

Again we fix the **coverage probability** $1 - \alpha$:

$$\begin{aligned} P\left(\mu_0 \leqslant \overline{x} + c\frac{S}{\sqrt{n}}\right) &= P\left(\frac{\overline{X} - \mu_0}{S/\sqrt{n}} \geqslant -c\right) \\ &= P(t_{n-1} \geqslant -c) \\ &= P(t_{n-1} \leqslant +c) \\ &= 1 - \alpha \end{aligned}$$

which is again the same as the corresponding **false alarm rate** condition.
Thus for non-coverage probability

- 0.05 we need c such that $P(t_{n-1} \leqslant c) = 1 - 0.05 = 0.95$,
- 0.01 we need c such that $P(t_{n-1} \leqslant c) = 1 - 0.01 = 0.99$.

Beer example

The 95% "upper confidence limit" is

```
c_05 = qt(.95, df = 5)
xbar + c_05 * se
```

```
[1] 375.1088
```

which gives the one-sided **confidence interval**

```
c(-Inf, xbar + c_05 * se)
```

```
[1]     -Inf 375.1088
```

For 99%

```
c_01 = qt(.99, df = 5)
c(-Inf, xbar + c_01 * se)
```

```
[1]     -Inf 375.2711
```

These both include 375!

Using `t.test()`

```
t.test(x, mu = 375, alternative = "less")
```

```

        One Sample t-test

data:  x
t = -1.1094, df = 5, p-value = 0.1589
alternative hypothesis: true mean is less than 375
95 percent confidence interval:
     -Inf 375.1088
sample estimates:
mean of x
 374.8667
```

```
t.test(x, mu = 375, alternative = "less", conf.level = 0.99)
```

```

        One Sample t-test

data:  x
```

```
t = -1.1094, df = 5, p-value = 0.1589
alternative hypothesis: true mean is less than 375
99 percent confidence interval:
     -Inf 375.2711
sample estimates:
mean of x
 374.8667
```

Observed significance level: the p-value

- Finally, to tie all of this together we relate it all to the p-value.
- The observed significance level (or p-value) is the value of α for which the observed data is "right on the edge".
- More precisely that is
  - the smallest **false alarm rate** for which we would "reject" a given value μ_0,
  - the *non-coverage probability* (i.e. 1− confidence level) for which μ_0 is on the boundary of the **confidence interval**

Beer example

Using the (two sided) p-value in our level of confidence gives us a confidence interval "right on the edge".

```
t.test(x, mu = 375, conf.level = 1 - 0.3177)
```

```
        One Sample t-test

data:  x
t = -1.1094, df = 5, p-value = 0.3177
alternative hypothesis: true mean is not equal to 375
68.23 percent confidence interval:
 374.7333 375.0000
sample estimates:
mean of x
 374.8667
```

Using the (one sided) p-value in our level of confidence gives us a confidence interval "right on the edge".

```
t.test(x, mu = 375, alternative = "less", conf.level = 1 - 0.1589)
```

```
        One Sample t-test

data:  x
t = -1.1094, df = 5, p-value = 0.1589
alternative hypothesis: true mean is less than 375
84.11 percent confidence interval:
 -Inf  375
sample estimates:
mean of x
 374.8667
```

11.3 Rejection regions

Decision rules

- To test a hypothesis, we previously defined a **decision rule** to reject H_0. That is when the p-value is less than certain fixed preassigned levels, say p-value $\leqslant \alpha$ where $\alpha = 0.05, 0.10$ etc

- In other words, we reject or do not reject or do not reject H_0 according to whether the p-value is less than or greater than α.
- The α is called the significance level of the test, which is the boundary between rejecting and not rejecting H_0.

Notation

Let $t_{n-1}(\alpha)$ be the **critcal value** (or quantile given by)

$$P(t_{n-1} \leqslant t_{n-1}(\alpha)) = \alpha,$$

or if we are using the standard normal distribution $Z \sim \mathcal{N}(0,1)$ then $z(\alpha)$ is defined by $P(Z \leqslant z(\alpha)) = \alpha$.

Critical value decision rule

The critical value depends on the level of significance, α, and the distribution of T under H_0, t_{n-1}

For a test of $H_0 : \mu = \mu_0$ vs $H_1 : \mu > \mu_0$, the **decision rule** at level α is:
- reject H_0 if $t_0 \geqslant t_{n-1}(1-\alpha)$ or equivalently reject H_0 if $t_0 \geqslant |t_{n-1}(\alpha)|$

For a test of $H_0 : \mu = \mu_0$ vs $H_1 : \mu < \mu_0$, the **decision rule** at level α is:
- reject H_0 if $t_0 \leqslant |t_{n-1}(\alpha)|$

For a test of $H_0 : \mu = \mu_0$ vs $H_1 : \mu \neq \mu_0$, the **decision rule** at level α is:
- reject H_0 if $|t_0| \geqslant |t_{n-1}(\alpha/2)|$
- do not reject H_0 if $|t_0| < |t_{n-1}(\alpha/2)|$

Rejection region for test statistics

Workflow: σ^2 unknown

- **Hypothesis**: $H_0 : \mu = \mu_0$ vs $H_1 : \mu > \mu_0, \mu < \mu_0, \mu \neq \mu_0$
- **Assumptions**: X_i are idd $\mathcal{N}(\mu, \sigma^2)$, where σ^2 is **unknown**.
- **Test statistic**: $T = \frac{\overline{X}-\mu_0}{S/\sqrt{n}} \sim t_{n-1}$
- **Observed test statistic**: $t_0 = \frac{\overline{x}-\mu_0}{s/\sqrt{n}}$
- **Rejection region**:
  - $H_1 : \mu \leqslant \mu_0 : t_0 \leqslant t_{n-1}(\alpha)$ or $t_0 \geqslant |t_{n-1}(\alpha)|$
  - $H_1 : \mu \neq \mu_0 : |t_0| \geqslant |t_{n-1}(\alpha/2)|$
- **Decision**: We reject H_0 if t_0 is in the rejection region

Workflow: σ^2 known

- **Hypothesis**: $H_0 : \mu = \mu_0$ vs $H_1 : \mu > \mu_0, \mu \neq \mu_0$
- **Assumptions**: X_i are idd $\mathcal{N}(\mu, \sigma^2)$, where σ^2 is **known**.
- **Test statistic**: $T = \frac{\overline{X}-\mu_0}{\sigma/\sqrt{n}} \sim t_{n-1}$
- **Observed test statistic**: $z_0 = \frac{\overline{x}-\mu_0}{\sigma/\sqrt{n}}$
- **Rejection region**:
  - $H_1 : \mu \leqslant \mu_0 : z_0 \leqslant z(\alpha)$ or $z_0 \geqslant |z(\alpha)|$
  - $H_1 : \mu \neq \mu_0 : |z_0| \geqslant |z(\alpha/2)|$
- **Decision**: We reject H_0 if z_0 is in the rejection region

Beer contents: testing using critical value

We have $n = 6, \overline{x} = 374.87, s = 0.29, t_0 = -1.11$.

- **Hypothesis**: $H_0 : \mu = 375$ vs $H_1 : \mu < 375$
- **Assumptions**: X_i are idd rv $\mathcal{N}(\mu, \sigma^2)$
- **Test statistic**: $T = \frac{\overline{X}-\mu_0}{\sigma/\sqrt{n}} \sim t_{n-1}$
- **Observed test statistic**: $t_0 = \frac{374.87-375}{0.29/\sqrt{6}} = -1.11$
- **Critical value**: $t_5 = (0.05) =$ `qt(0.05, df = 5)` $= -2.015$ i.e. reject if is less than -2.015
- **Decision**: the observed test statistic, $t_0 = -1.11$, is greater than -2.015, so do not reject H_0

11.4 Rejection region on the data scale

Smoking and blood platelet aggregation

Blood samples from 11 individuals before and after they smoked a cigarette are used to measure aggregation of blood platelets.

```
before = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
after =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
df = data.frame(before, after, difference = after - before)
```

- This is a match-pair sample.
- We reduce the data to one sample by considering the aggregation difference.
- Let X_i and Y_i be the blood platelet aggregation levels for the i^{th} person before and after smoking, respectively.
- Define the change in person i's platelet aggregation levels as $D_i = Y_i - X_i$ and the population mean change in platelet aggregation levels as μ_d

> **Is blood platelet aggregation affected by smoking?**

Rejection region for sample mean

The rejection regions for the test using test statistic

$$t_0 = \frac{\overline{x} - \mu_0}{s/\sqrt{n}} \geqslant t_{n-1}(\alpha)$$

on the standardised scale can be transformed to the measurement scale. We can do this because…

$$\begin{aligned}
\alpha &= P\left(\frac{\overline{x} - \mu_0}{s/\sqrt{n}} \geqslant t_{n-1}(\alpha)\right) \\
&= P\left(\overline{x} - \mu_0 \geqslant t_{n-1}(\alpha)s/\sqrt{n}\right) \\
&= P\left(\overline{x} \geqslant t_{n-1}(\alpha)s/\sqrt{n} + \mu_0\right)
\end{aligned}$$

Which means we can define a rejection region on the measurement scale

$$\{\overline{x} : \overline{x} \geqslant k_0 = \mu_0 + t_{n-1}(\alpha)s/\sqrt{n}\} \quad \text{for} \quad H_1 : \ \mu > \mu_0.$$

We have $n = 11, \overline{d} = 8.45, s_d = 9.65, t_0 = 2.91$

- **Observed test statistic**: $t_0 = \frac{\overline{d}}{s_d/\sqrt{n}} = \frac{8.45}{9.65/\sqrt{11}} = 2.91$

- **Rejection region**: $\left|\frac{\overline{d}-\mu_d}{s_d/\sqrt{n}}\right| > t_{10}(0.025) = 2.228$, rearranging,

$$\begin{aligned} \overline{d} &< \mu_d - t_{n-1}(0.025)\, s_d/\sqrt{n} & \overline{d} &> \mu_d + t_{n-1}(0.025)\, s_d/\sqrt{n} \\ \overline{d} &< 0 - 2.228 \times 9.65/\sqrt{11} & \overline{d} &> 0 + 2.228 \times 9.65/\sqrt{11} \\ \overline{d} &< -6.48 & \overline{d} &> 6.48 \end{aligned}$$

- **Decision**: If $\overline{d} < -6.48$ or $\overline{d} > 6.48$ then reject H_0. In this case, $\overline{d} = 8.45 > 6.48$ so we reject H_0.

```
before = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
after =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
df = data.frame(before, after, difference = after-before)
s_d = sd(df$difference)
s_d
```

```
[1] 9.647421
```

```
n = nrow(df)
mu0 = 0
crit_val = qt(0.975, df = n-1)
crit_val
```

```
[1] 2.228139
```

```
rrlower = mu0 - crit_val * s_d / sqrt(n)
rrupper = mu0 + crit_val * s_d / sqrt(n)
c(rrlower, rrupper) |> round(2)
```

```
[1] -6.48  6.48
```

Beer contents

We have $n = 6, \overline{d} = 374.87, s_d = 0.29, t_0 = -1.11$. Hypothesis test using rejection region with $\alpha = 0.05$.

- **Hypothesis**: $H_0 : \mu = 375$ vs $H_1 : \mu < 375$
- **Assumptions**: X_i are idd rv $\mathcal{N}(\mu, \sigma^2)$
- **Test statistic**: $T = \frac{\overline{X}-\mu_0}{\sigma/\sqrt{n}} \sim t_{n-1}$
- **Rejection region (on the data scale)**:

$$\begin{aligned} \frac{\overline{X}-\mu}{s/\sqrt{n}} &< t_{n-1}(0.05) \\ \overline{X} &< \mu + t_{n-1}(0.05)\, s/\sqrt{n} \\ \overline{X} &< 375 - 2.015 \times 0.29/\sqrt{6} \\ \overline{X} &< 374.74 \end{aligned}$$

- **Decision**: the observed test statistic, $\overline{x} = 374.9$, is greater than 374.9, so do not reject H_0.

12
Sample Size Calculations and Power

12.1 Power and sample size

Errors in hypothesis testing

	H_0 **true (innocent)**	H_0 **false (guilty)**
Don't reject H_0 (acquit)	Correct decision	Type II error (β)
Reject H_0 (guilty)	Type I error (α)	Correct decision ($1-\beta$)

- Type I errors: level of significance, $\alpha = P(\text{reject } H_0 \mid H_0 \text{ true})$
- Type II errors: call it β
- Power: $1-\beta = P(\text{reject } H_0 \mid H_1 \text{ true})$

General testing setup

- Interested in inference concerning an unknown population mean μ
- We are considering a fixed value μ_0 ("**hypothesised value**")
- We then observe the data $x_1, \ldots, x_n$, obtaining
  - the sample mean $\overline{x}$
  - the sample sd s and thus the *estimated standard error* (se) $s/\sqrt{n}$
- We decide to perform a (say, two-sided) t-test, that is to say if the *observed discrepancy* $\overline{x} - \mu_0$ is *large* compared to the se, we will "reject" the value μ_0 as "implausible"

$$\text{Reject if } |\overline{x} - \mu_0| > c\frac{s}{\sqrt{n}},$$

  where c is chosen so that the false alarm rate is some fixed, small value α (e.g. 0.05, 0.01).

Model assumptions

- The **false alarm rate** determination can only be made if a suitable statistical model is assumed for the data.
- If we model the data $x_1, \ldots, x_n$ as values taken by iid $\mathcal{N}(\mu, \sigma^2)$ random variables $X_1, \ldots, X_n$ (with μ and σ^2 *both unknown*), then whatever the true value μ is we know, $\frac{\overline{X}-\mu}{S/\sqrt{n}} \sim t_{n-1}$.
- The **false alarm rate** is

$$P_{\mu_0}\left(\left|\overline{X} - \mu_0\right| > c\frac{S}{\sqrt{n}}\right) = P_{\mu_0}\left(\frac{\left|\bar{X} - \mu_0\right|}{S/\sqrt{n}} > c\right) = P(|t_{n-1}| > c) = 2P(t_{n-1} > c)$$

  where the final equality follows by symmetry of the $t_n - 1$ distribution.
- The $P_{\mu_0}(\cdot)$ indicates probability when the true value is μ_0, i.e. the value we specified in the null hypothesis.

Beer example

- Suppose we have $n = 6$ and choose a **false alarm rate** of $\alpha = 0.05$.
- Then the constant c needs to satisfy

$$2P(t_5 > c) = \alpha$$

so

$$P(t_5 > c) = \alpha/2 = 0.025$$

and thus

$$P(t_5 \leqslant c) = 0.975.$$

```
c_05 = qt(0.975, df = 5)
c_05
```

```
[1] 2.570582
```

Why allow false alarms at all?

- A fair question is: *why would you set things up to have 5% **false alarm rate**?*
- Why not make it *really small*, like 10^{-6}?
- *Answer*: because then you would never reject anything, even if you should!
- The technical reason: because then the test would have no **power**.

Definition

The **power** of a test is the probability that the test rejects the null hypothesis, H_0 when a **specific** alternative hypothesis H_1 is true.

$$\text{Power} = P(\text{reject } H_0 \mid H_1 \text{is true}).$$

Statistical power in one sample t-test

- Consider the probability of "rejecting" as a function of the true population mean μ:

$$P_\mu(\text{reject } H_0) = P_\mu\left(\left|\overline{X} - \mu_0\right| > c\frac{S}{\sqrt{n}}\right) = P_\mu\left(\frac{\left|\overline{X} - \mu_0\right|}{S/\sqrt{n}} > c\right).$$

- This is the **statistical power function** of the test.
- To determine this we need to know the *distribution* of the t-statistic for testing μ_0:

$$\frac{\overline{X} - \mu_0}{S/\sqrt{n}}$$

when the true population mean μ is not necessarily equal to μ_0 (the hypothesised population mean)!

Beer example: power calculations

- Suppose the sample sd is *indicative* of the "true" population standard deviation.
- We want to plot the power function of the test as a function of μ.
- First let us assume the "true" σ is equal to the sample value.

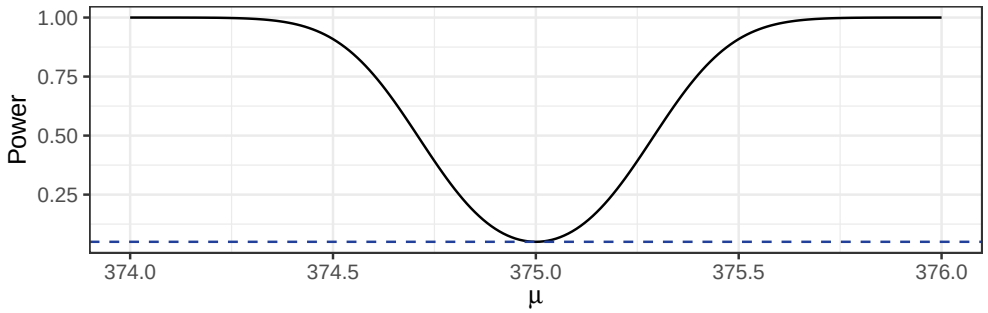
```
x = c(374.8, 375.0, 375.3, 374.8, 374.4, 374.9)
sig = sd(x)
sig
```

```
[1] 0.294392
```

Power as a function of true mean μ

Given that we pick $H_0 : \mu = 375$

```
mu = seq(374, 376, by = 0.01)
nu = (mu-375)/(sig/sqrt(6))
power_mu = pt(c_05, df = 5, ncp = nu, lower.tail = FALSE) + pt(-c_05, df = 5,
    ncp = nu)
power_df = data.frame(mu, nu, power = power_mu)
g1 = ggplot(data = power_df, aes(x=mu, y=power)) +
  geom_line() +
  labs (x = expression(mu),
        y = "Power") +
  geom_hline(yintercept = 0.05, colour = "#224099", linetype = "dashed") +
  theme_bw()
g1
```



The further the true mean μ is from the value μ_0 picked for the null hypothesis the more likely we are to reject.

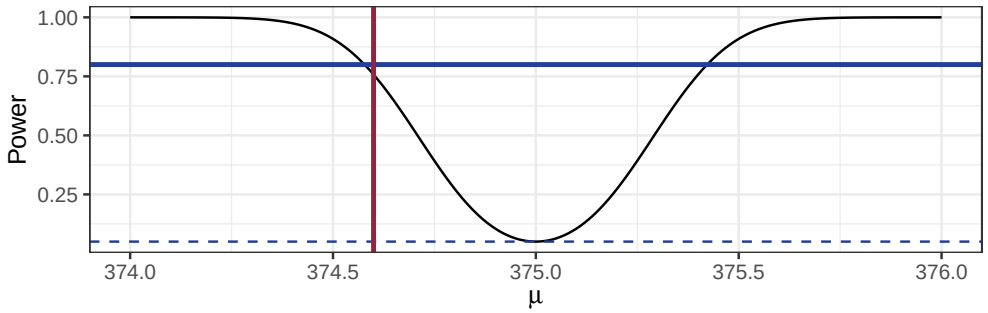
Discrepancy needed to achieve a certain power

Assume:

- population sd $\sigma = 0.294$
- sample size of $n = 6$
- $H_0 : \mu = 375$ vs $\mu \neq 375$

How much lower than 375 does μ need to be for us to be 80% sure of "detecting" that $\mu \neq 375$ with a two-sided test which has false alarm rate 0.05?

```
g1 +
geom_hline(yintercept = 0.8, colour = "#224099",linewidth = 1) +
geom_vline(xintercept = 374.6, colour = "#992240",linewidth = 1)
```



- The μ value with power 80% is around 374.6.

Comparing to false alarm rate of 10^{-6}

For a two sided test with $n = 6$ and a **false alarm rate** of $\alpha = 10^{-6}$, we need a critical value of

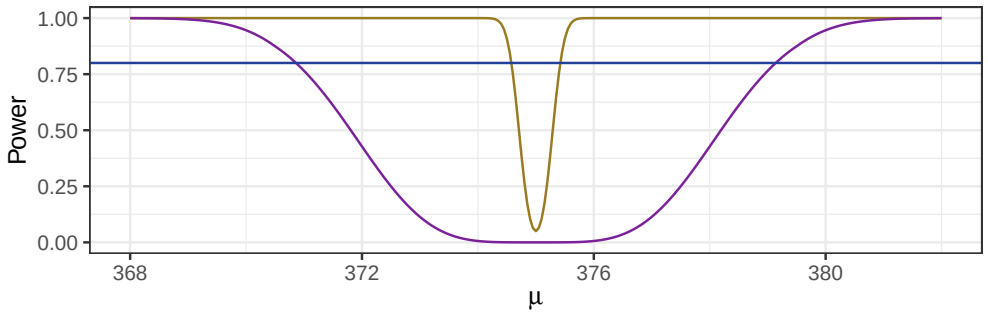
```
crit_val = qt(1 - (1e-6) / 2, df = 5)
crit_val
```

```
[1] 28.47847
```

So we would need a discrepancy equal to more than 28 standard errors before we would reject 375!

Power as a function of α

```
mu = seq(368, 382, by = 0.05)
nu = (mu-375)/(sig/sqrt(6))
power1 = pt(c_05, df = 5, ncp = nu, lower.tail = FALSE) + pt(-c_05, df = 5,
    ncp = nu)
power2 = pt(crit_val, df = 5, ncp = nu, lower.tail = FALSE) + pt(-crit_val, df
     = 5, ncp = nu)
power_df = data.frame(mu, nu, power1, power2)
ggplot(data = power_df, aes(x=mu)) +
  geom_line(aes(y = power1), colour = "#997b22") +
  geom_line(aes(y = power2), colour = "#7b2299") +
  labs (x = expression(mu),
        y = "Power") +
  geom_hline(yintercept = 0.8, colour = "#224099") +
  theme_bw()
```



To achieve a power of about 80% with such a small false alarm rate, we would require a true μ lower than 371 or more than 379!

Power as a function of n

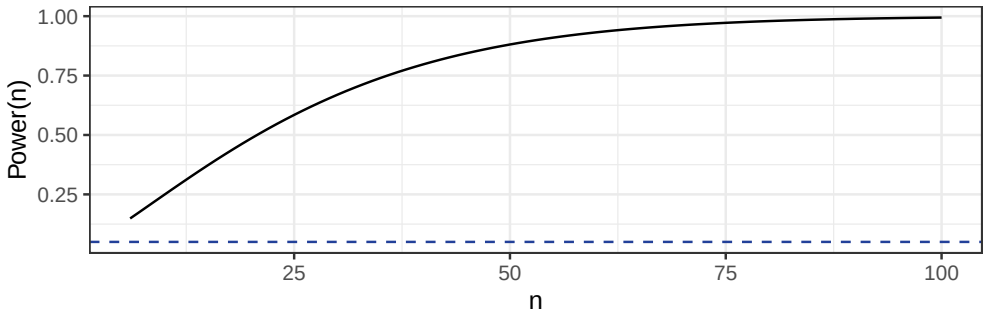
- Now let us suppose that *both* the sample mean and sample sd are indicative of the "true" values μ and σ

```
xbar = mean(x)
c(xbar, sig)
```

```
[1] 374.866667   0.294392
```

- Can we see how the power ought to behave as a function of n?
- *Note* the degrees of freedom, and thus the critical value, change with n.

```
power_n = data.frame(
  n = 6:100
) |> mutate(
  nu = (xbar-375)/(sig/sqrt(n)),
  c_05_n = qt(0.975, df = n-1),
  power = pt(c_05_n, df = n-1, ncp = nu, lower.tail = FALSE) +
    pt(-c_05_n, df = n-1, ncp = nu)
)
ggplot(data = power_n, aes(x = n, y = power)) +
  geom_line() +
  labs (x = "n",
        y = "Power(n)") +
  geom_hline(yintercept = 0.05, colour = "#224099", linetype = "dashed") +
  theme_bw()
```



Again, this is assuming the "true" values μ and σ equal the sample values $\bar{x}$ and s. But it is still useful!

12.2 How to do it in R

There's a package for that

The **pwr** package (Champely, 2020).

```
library(pwr)
```

The key functions:

- `pwr.t.test()` t-tests (one sample, 2 sample, paired)
- `pwr.t2n.test()` t-test (two samples with unequal n)

Cohen's d

Rather than specifying a null mean and an alternative mean and standard deviation, the **pwr** functions take as an input "Cohen's d":

$$d = \frac{|\mu_1 - \mu_2|}{\sigma}$$

Cohen suggests that d values of 0.2, 0.5, and 0.8 represent small, medium, and large effect sizes respectively.

Beer example

- Suppose the population sd $\sigma = 0.294$, with a sample size of $n = 6$ how much lower than 375 does μ need to be for us to be 80% sure of "detecting" that $\mu \neq 375$ with a *two-sided* test which has **false alarm rate** 0.05?

```
res = pwr.t.test(n = 6,
                 d = NULL,
                 sig.level = 0.05,
```

```
                power = 0.8,
                type = "one.sample",
                alternative = "two.sided")
                res$d * 0.294 # d * sigma gives the difference between
                    means
```

```
[1] 0.4217541
```

$$d = \frac{|\mu_1 - \mu_2|}{\sigma} = 1.43 \implies |\mu_1 - \mu_2| = 1.43\sigma = 0.42$$

- Suppose the population $\mu = 374.87$ and $\sigma = 0.294$, what sample size n would be needed to be at least 80% sure of detecting that $\mu \neq 375$ with a *two-sided* test which has **false alarm rate** 0.05?

```
res = pwr.t.test(n = NULL,
                 d = (374.87-375)/0.294,
                 sig.level = 0.05,
                 power = 0.8,
                 type = "one.sample",
                 alternative = "two.sided")
res
```

```
     One-sample t test power calculation

              n = 42.10456
              d = 0.4421769
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

We would need at least 43 observations (round up from 42.1 because if we round down then the power will be less than 80%).

13

Sign test

13.1 Checking for normality

Normality

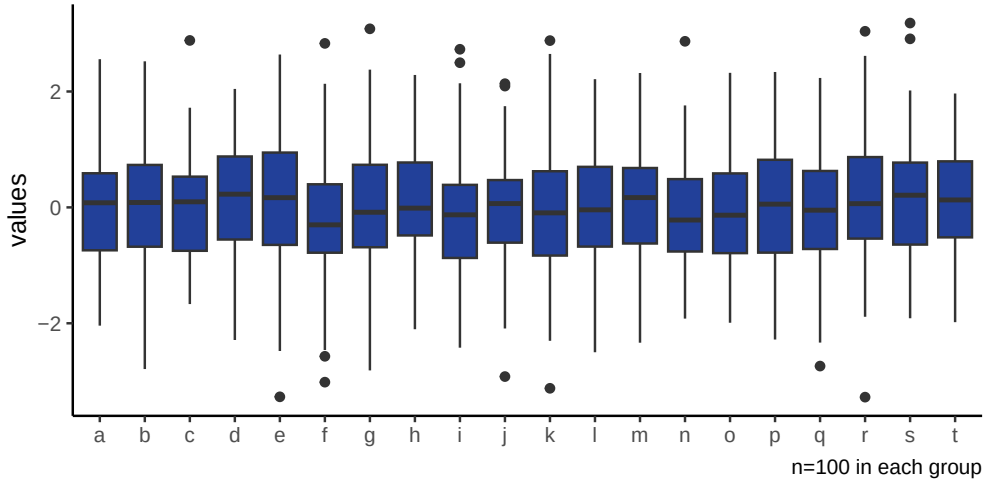
- The assumption that your data are sampled from a normal population arises quite often.
- If you have a **large enough** sample size, then the normality assumption is not as important as you can usually rely on the central limit theorem to ensure your test statistic at least approximately follows a t-distribution
- In small samples it can be difficult to tell whether or not your sample comes from a normal population!

You can check the normality assumption using:

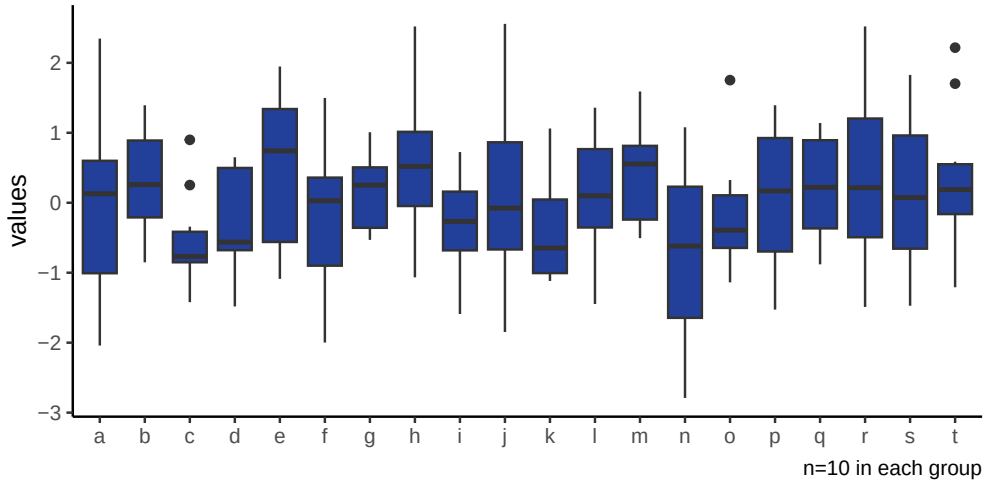
- **QQ plots** (this is the best way to visually assess normality)
- **Boxplots** (most useful when only looking for symmetry)
- Formal hypothesis tests?

Checking for normality: boxplots

```
set.seed(88)
n_groups = 20
n_obs = 100
dat_100 = data.frame(
  values = rnorm(n_groups * n_obs),
  group = rep(letters[1:n_groups], each = n_obs)
)
ggplot(dat_100) + aes(x = group, y = values) +
  geom_boxplot(fill = "#224099") +
  labs(x = NULL, caption = "n=100 in each group") +
  theme_classic()
```



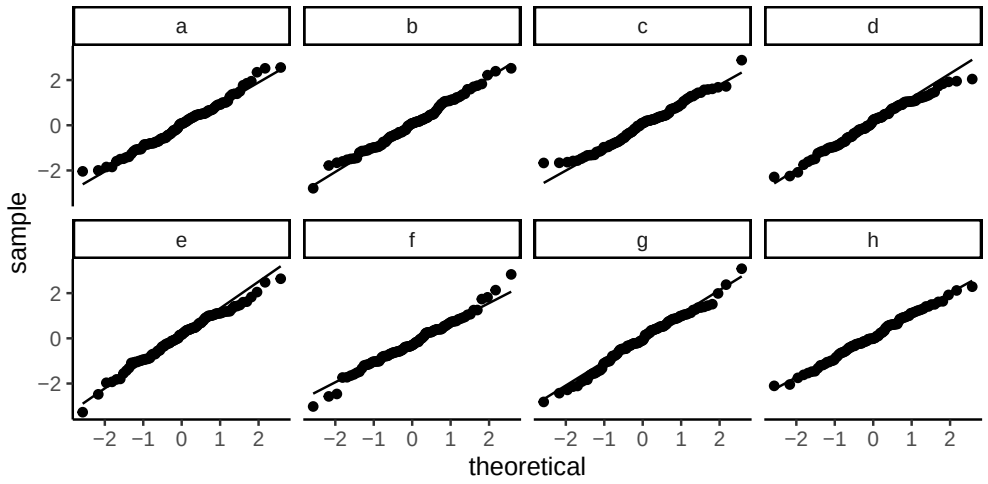

```
set.seed(88)
n_groups = 20
n_obs = 10
dat_10 = data.frame(
  values = rnorm(n_groups * n_obs),
  group = rep(letters[1:n_groups], each = n_obs)
)
ggplot(dat_10) + aes(x = group, y = values) +
  geom_boxplot(fill = "#224099") +
  labs(x = NULL, caption = "n=10 in each group") +
  theme_classic()
```



> To check for normality in a boxplot, we're mostly looking for **symmetry**. But this is **hard**, particularly in small samples.

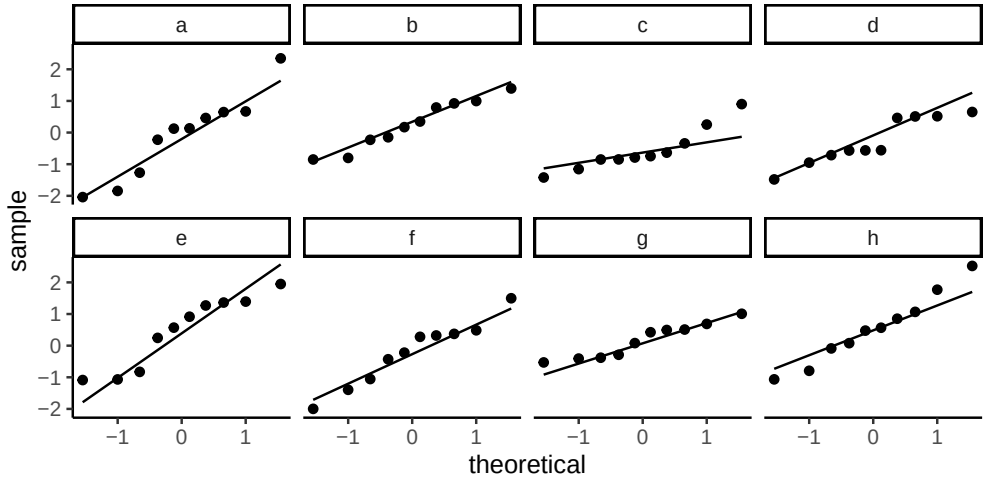
Checking for normality: QQ plots

```
dat_100 |> dplyr::filter(group <= "h") |>
  ggplot() +
  aes(sample = values, group = group) +
  geom_qq_line() + geom_qq() +
  facet_wrap(~group, nrow = 2) +
  theme_classic()
```



These are QQ-plots, each with 100 observations drawn from a normal population.

```
dat_10 |> dplyr::filter(group <= "h") |>
  ggplot() +
  aes(sample = values, group = group) +
  geom_qq_line() + geom_qq() +
  facet_wrap(~group, nrow = 2) +
  theme_classic()
```

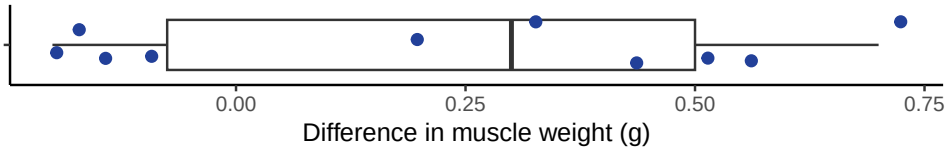


These are QQ-plots, each with 10 observations drawn from a normal population.
If we're checking for normality in a QQ plot, then we're mostly looking for **points that lie reasonably close to the line**.

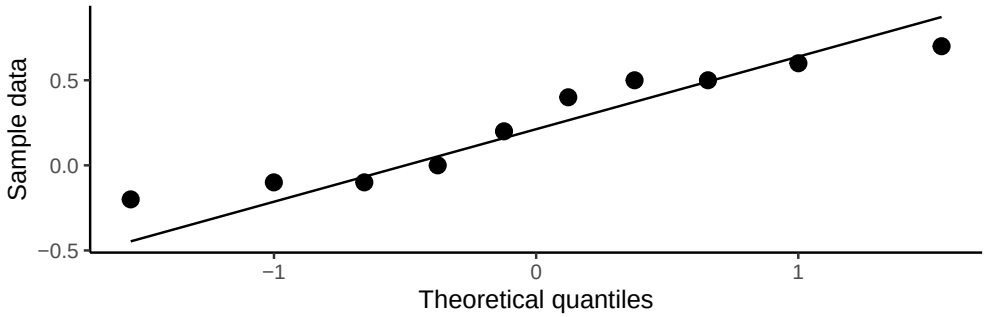
Rat data: normality

```
rat = data.frame(
  bio = c(1.7, 2.0, 1.7, 1.5, 1.6,
```

```
          2.4, 2.3, 2.4, 2.4, 2.6),
  pla = c(2.1, 1.8, 2.2, 2.2, 1.5,
          2.9, 2.9, 2.4, 2.6, 2.5)
) |> mutate(d = pla - bio)
ggplot(rat) + aes(y = "", x = d) +
  geom_boxplot() +
  geom_jitter(width = 0.1,
              size = 2,
              colour = "#224099") +
  labs(y = "",
       x = "Difference in muscle weight (g)") +
  theme_classic()
```




```
ggplot(rat, aes(sample = d)) +
  geom_qq(size = 3) +
  geom_qq_line() +
  labs(x = "Theoretical quantiles",
       y = "Sample data") +
  theme_classic()
```

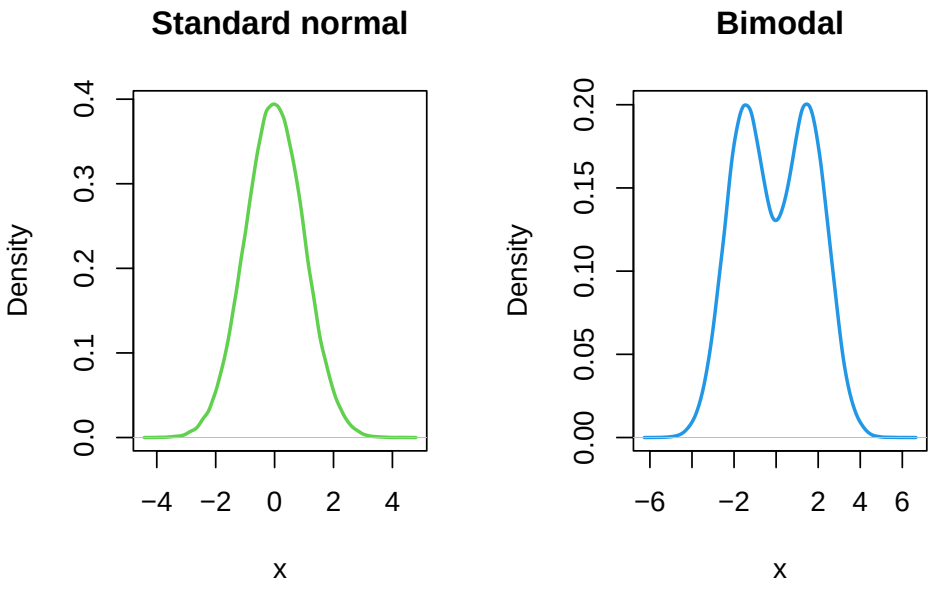


13.2 Sign test

- Suppose a sample $X_1, \ldots, X_n$ are independently sampled from a continuous distribution with mean μ.
- We want to test $H_0 : \mu = \mu_0$.
- If the distribution is **symmetric** about μ_0 under H_0, then $D_i = X_i - \mu_0$ should scatter around 0
  - i.e. D_i is equally likely to be positive or negative.
- If H_0 is true, the probability p_+ of getting a positive D_i is 0.5.
- The **sign test** reduces to a binomial test of proportions.
- The sign test is a **nonparametric** test as no assumption on the data distribution is made except symmetry (though we do still require the independence assumption).

Symmetric distributions

```
set.seed(88)
n = 100000
par(mfrow = c(1,2))
plot(density(rnorm(n)),
     main = 'Standard normal', ylab = 'Density', xlab = 'x', col = 3, lwd = 2)
plot(density(c(rnorm(n, mean = -1.5), rnorm(n, mean = 1.5))),
     main = 'Bimodal', ylab = 'Density', xlab = 'x', col = 4, lwd = 2)
```



14

Wilkcoxon signed-rank test

14.1 Wilkcoxon signed-rank test

What was wrong with the sign test?

- The sign test ignores a lot of information (inefficient use of data; low power).
- How can we use more information than just the sign for data with a symmetric, but possibly non-normal, distribution?
- Suppose the sample $X_1, X_2, \ldots, X_n$ are drawn from a population symmetric with respect to mean μ (or median)
- We test the hypotheses: $H_0 : \mu = \mu_0$ vs $H_1 : \mu > \mu_0 \mu < \mu_0, \mu \neq \mu_0$.
- The t-test and Z-test assume a normal distribution without a long tail (outliers).
- They use all magnitude information.
- On the other hand, the sign test discards all data information on magnitude and hence it has low power.

Ranks to the rescue

Many non-parametric tests are based not on the data, but on their ranks. To find the ranks for a set od data:

- Arrange the data in ascending order
- Assign a rank of 1 to the smallest observation, 2 to the second smallest, etc.
- For tied observations (in blue or red in the table below), assign each the average of the corresponding ranks

Sample	8	5	**10**	2	5	8	8	6
Ordered sample	2	5	5	6	8	8	8	10
Successive Ranks	1	2	3	4	5	6	7	8
Assigned Ranks	1	2.5	2.5	4	6	6	6	8

Thinking about magnitude

- Under the symmetric distribution assumption with mean μ_0 from H_0, half of the $d_i - x_i - \mu_0$ should be negative and half positive and the expected counts are both $\frac{n}{2}$.
- Under the null hypothesis, the positive and negative d_i should be of similar magnitude and occur with equal probability (on average).
- If we rank the absolute values of d_i in ascending order, the expected rank sums for the negative and positive d_i should be nearly equal.

Wilcoxon signed-rank test

We need to define the following quantities:

- $D_i = X_i - \mu_0$ for $i - 1, 2, \dots, n$
- $R_i, \dots, R_n$ be the ranks of $|D_1|, |D_2|, \dots, |D_n|$
- W^+ be the sum of the ranks R_i corresponding to positive D_i
- W^- be the sum of the ranks R_i corresponding to negative D_i
- Let $W = \min(W^+, W^-)$

Calculations of w^+ and w

When we observe the data we have $d_i - x_i - \mu_0$ with ranks (of the absolute values), $r_1, \dots, r_n$ for $|d_1|, \dots, |d_n|$.

$$w^+ = \sum_{i:d_i>0} r_i \quad \text{and} \quad w^- = \sum_{i:d_i<0} r_i$$

We should

- reject $H_0 : \mu = \mu_0$ in favour of $H_1 : \mu > \mu_0$ if w^+ is large enough
- reject $H_0 : \mu = \mu_0$ in favour of $H_1 : \mu < \mu_0$ if w^+ is small enough
- reject $H_0 : \mu = \mu_0$ in favour of $H_1 : \mu \neq \mu_0$ if w^+ if $w = \min(w^+, w^-)$ is small enough

Suppose $X_i, \dots, X_n$ are drawn from some population that follows a symmetric distribution. Given a significance level α, we want to test on the population mean, μ.

Workflow: Wikcoxon signed-rank test

- **Hypotheses**: $H_0 : \mu = \mu_0$ vs $H_1 : \mu > \mu_0, \mu < \mu_0$ or $\mu \neq \mu_0$
- **Assumptions**: X_i are independently sampled from a symmetric distribution.
- **Test Statistic**: $W^+ = \sum_{i:D_i>0} R_i$ for one-sided or $W = \min(W^+, W^-)$ for two-sided
- **Observed Test Statistic**: w^+ for one-sided or $w = \min(w^+, w^-)$ for two-sided
- **p-value**:
  - $P(W^+ \geqslant w^+)$ for $H_1 : \mu > \mu_0$
  - $P(W^+ \leqslant w^+)$ for $H_1 : \mu < \mu_0$
  - $P(W^+ \leqslant w)$ for $H_1 : \mu \neq \mu_0$
- **Decision**: If the p-value is less then α, there is evidence against H_0. If the p-value is greater than α, the data are consistent with H_0.

14.2 Normal approximation to the wilcoxon signed-rank test statistic

Normal approximation

For large enough n we can use a normal distribution to approximate the distribution of the Wilcoxon sign rank test statistic.
i.e. in large samples (without ties),

$$W^+ \sim \mathcal{N}\left(\frac{n(N+1)}{4}, \frac{n(n+1)(2n+1)}{24}\right), \text{approximately}$$

Hence the large sample test statistic is,

$$T = \frac{W^+ - \mathrm{E}(W^+)}{\sqrt{\mathrm{Var}(W^+)}} \sim \mathcal{N}(0,1)$$

where $\mathrm{E}(W^+) = \frac{n(n+1)}{4}$ and $\mathrm{Var}(W^+) = \frac{n(n+1)(2n+1)}{24}$.

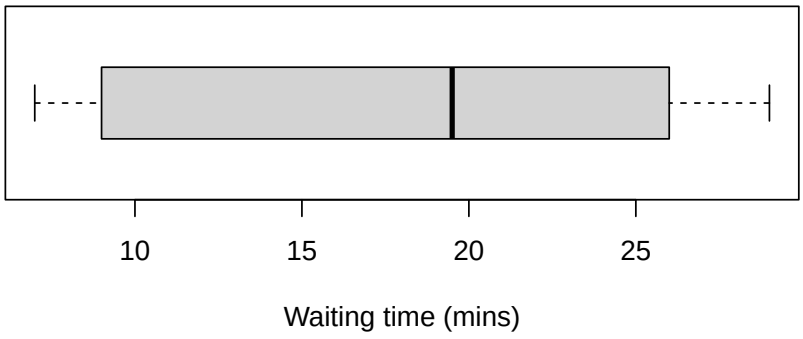
Bus Waiting Times

The following data are waiting times for the 370 bus in minutes for 10 randomly selected passengers:

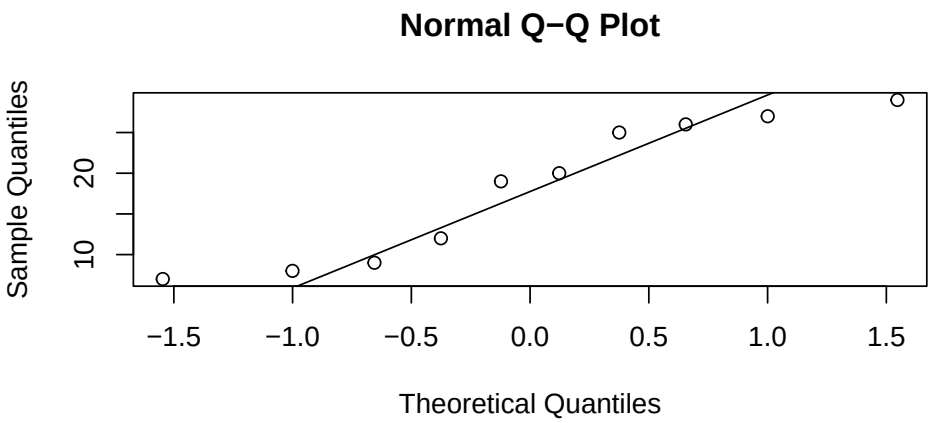
```
bus = c(25, 19, 9, 27, 8, 7, 26, 12, 29, 20)
```

The bus authority claims a typical wait time of 15 minutes. Do these data suggest a different typical wait time? The standard approach is a one-sample t-test to test $H_0 : \mu = 15$

```
boxplot(bus, horizontal = TRUE,
        xlab = "Waiting time (mins)")
```




```
qqnorm(bus); qqline(bus)
```



- **Hypotheses**: $H_0 : \mu = 15$ vs $H_1 : \mu \neq 15$
- **Assumptions**: X_i are independently sampled from a symmetric distribution.
- **Test Statistic**: $W = \min(W^+, W^-)$ where $W^+ = \sum_{i:D_i>0} R_i, W^- = \sum_{i:D_i<0} R_i, D_i = X_i - 15$ and R_i are the ranks of $|D_1|, |D_2|, \ldots, |D_n|$. Under $H_0, W^+ \sim WSR(10)$ a symmetric distribution with mean $\mathrm{E}(W^+) = \frac{n(n+1)}{4} = 27.5$ and $\mathrm{Var}(W^+) = \frac{n(n+1)(2n+1)}{24} = 96.25$
- **Observed Test Statistic**: $t : 0 = \frac{w-\mathrm{E}(W^+)}{\sqrt{\mathrm{Var}(W^+)}} = \frac{16-27.5}{\sqrt{96.25}} = -1.172$
- **p-value**: $2P(W^+ \leqslant 16) = 0.2754$
- **Decision**: Large p-value so the data are consistent with H_0. Therefore there is no evidence to dispute the bus authority's claim of a typical wait time of 15 minutes.

Normal approximation with ties

We can approximate W^+ by a normal distribution. The p-value is approximately given by,

$$\text{p-value} \approx P\left(Z \geqslant \frac{w^+ - \mathrm{E}(W^+)}{\sqrt{\mathrm{Var}(W^+)}}\right) \quad \text{for } H_1 : \mu > \mu_0$$

$$\text{p-value} \approx P\left(Z \geqslant \frac{w^+ - \mathrm{E}(W^+)}{\sqrt{\mathrm{Var}(W^+)}}\right) \quad \text{for } H_1 : \mu < \mu_0$$

$$\text{p-value} \approx P\left(Z \geqslant \left|\frac{w^+ - \mathrm{E}(W^+)}{\sqrt{\mathrm{Var}(W^+)}}\right|\right) \quad \text{for } H_1 : \mu \neq \mu_0,$$

where **in general**, $\mathrm{E}(W^+) = \frac{1}{2}\sum_{i:d_i\neq 0} r_i$ and $\mathrm{Var}(W^+) = \frac{1}{4}\sum_{i:d_{i\neq 0}} r_i^2$.

Smoking and Blood Platelets Aggregation

Blood samples from 11 individuals before and after they smoked a cigarette are used to measure aggregation of blood platelets.

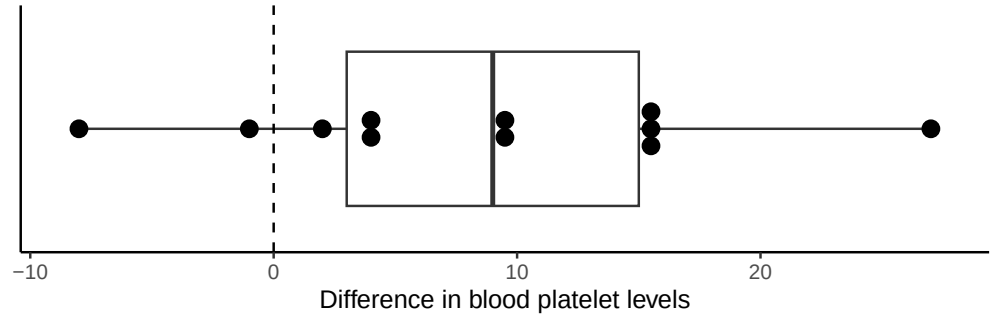
```
before = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
after =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
df = data.frame(before, after,
                difference = after-before)
df
```

```
   before after difference
1      25    27          2
2      25    29          4
```

```
3       27     37          10
4       44     36          -8
5       30     46          16
6       67     82          15
7       53     57           4
8       53     80          27
9       52     61           9
10      60     59          -1
11      28     43          15
```

> Is the aggregation of blood platelets affected by smoking?

```
p = df |> ggplot() +
  aes(y = "", x = difference) +
  geom_boxplot() +
  geom_dotplot(binaxis = "x", stackdir = "center", dotsize = 0.6) +
  geom_vline(xintercept = 0, linetype='dashed') +
  labs(x = 'Difference in blood platelet levels') +
  theme_classic() +
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
p
```




```
df = df |> dplyr::mutate(absDif = abs(difference),
                         rankAbsDif = rank(absDif),
                         srank = sign(difference)*rank(abs(difference)))
df
```

```
   before after difference absDif rankAbsDif srank
1      25    27          2      2        2.0   2.0
2      25    29          4      4        3.5   3.5
3      27    37         10     10        7.0   7.0
4      44    36         -8      8        5.0  -5.0
5      30    46         16     16       10.0  10.0
6      67    82         15     15        8.5   8.5
7      53    57          4      4        3.5   3.5
8      53    80         27     27       11.0  11.0
9      52    61          9      9        6.0   6.0
10     60    59         -1      1        1.0  -1.0
11     28    43         15     15        8.5   8.5
```

```
(w_p = sum(df$srank[df$srank > 0]))
```

```
[1] 60
```

```
(w_m = sum(-df$srank[df$srank < 0]))
```

```
[1] 6
```

```
(w = min(w_p, w_m))
```

```
[1] 6
```

- **Hypotheses**: $H_0 : \mu_d = 0$ vs $H_1 : \mu_d \neq 0$
- **Assumptions**: D_i are independently sampled from a symmetric distribution.
- **Test Statistic**: $W+ = \sum_{i:D_i>0} R_i$ where R_i are the ranks of $|D_1|, |D_2|, \ldots, |D_n|$. Under $H_0, W^+ \sim WSR'(11)$ and the set of ties as given.
- **Observed Test Statistic**: $w = \min(w^+, w^-) = 6$ because $w^+ = 60, w^- = 6$
- **p-value**: SInce the $WSR'(11)$ distribution is unknown, it is approximated by normal with $E(W^+) = \frac{n(n+1)}{4} = \frac{11(11+1)}{4} = 33$ and $\text{Var}(W^+) = \frac{1}{4}\sum_{i=1}^{11} r_i^2 = \frac{1}{4}[(2)^2 + \cdots + (8.5)^2] = \frac{506}{4} = 126.25$. The p-value $= 2P(W^+ \leqslant 6) \simeq 2P(Z \leqslant \frac{6-33}{\sqrt{126.25}}) = 2P(Z \leqslant -2.403) = 2 \times 0.008 = 0.016$
- **Decision**: the p-value is less than 0.05, hence there is evidence against H_0.

```
ew = sum(df$rankAbsDif)/2
varw = sum((df$rankAbsDif)^2)/4
c(w, ew, varw)
```

```
[1]   6.00  33.00 126.25
```

```
t0 = (w - ew)/sqrt(varw)
p_value = 2 * pnorm(t0)
c(t0, p_value)
```

```
[1] -2.40296846  0.01626259
```

```
wilcox.test(df$difference)
```

```
        Wilcoxon signed rank test with continuity correction

data:  df$difference
V = 60, p-value = 0.01835
alternative hypothesis: true location is not equal to 0
```

```
wilcox.test(df$difference, correct = FALSE)
```

```
        Wilcoxon signed rank test

data:  df$difference
V = 60, p-value = 0.01626
alternative hypothesis: true location is not equal to 0
```

```
t.test(df$difference, alternative = "two.sided")
```

```
        One Sample t-test

data:  df$difference
t = 2.9065, df = 10, p-value = 0.01566
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  1.97332 14.93577
sample estimates:
mean of x
 8.454545
```

```
binom.test(c(2,9), 0.5)
```

```
        Exact binomial test

data:  c(2, 9)
number of successes = 2, number of trials = 11, p-value = 0.06543
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.0228312 0.5177559
sample estimates:
probability of success
             0.1818182
```

Final notes

A few extra notes about the Wilcoxon signed-rank test:

- Since we assume that the distribution is symmetric, the hypotheses can also be stated in terms of the **median** (rather than the mean).
- The p-value from a Wilcoxon signed-rank test will typically be smaller than the p-value of a sign test on the same data. Using the information in the ranks, the test becomes much more **powerful** in detecting differences from μ_0 and almost as powerful as the one sample t-test

15

Wilcoxon rank-sum test

15.1 Wilcoxon rank-sum test

It is also known as the Also known as Mann-Whitney U test or Mann-Whitney-Wilcoxon test or Wilcoxon-Mann-Whitney test

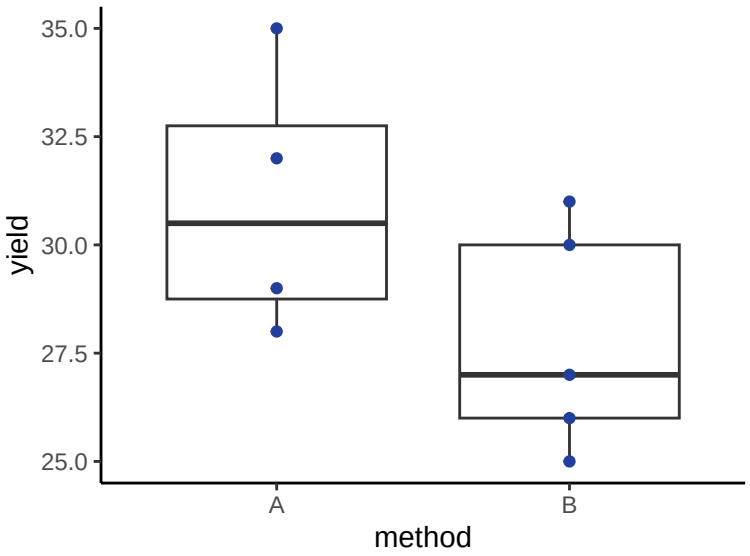
Yield

The following data yield measurements by two different methods.

```
A = c(32, 29, 35, 28)
B = c(27, 31, 26, 25, 30)
dat = data.frame(
        yield = c(A, B),
        method = c(rep("A", length(A)),
                   rep("B", length(B)))
)
```

If the normality assumptions are in doubt, does the data present sufficient evidence to indicate a difference between method A and B?

```
ggplot(dat, aes(x = method, y = yield)) +
geom_boxplot() +
theme_classic() +
geom_point(colour = "#224099")
```



Wilcoxon rank-sum test

- A non-parametric test to compare means of two independent samples
- Relaxes normality assumption (like sign and Wilcoxon sign-rank tests)
- Also relaxes the assumption of symmetry

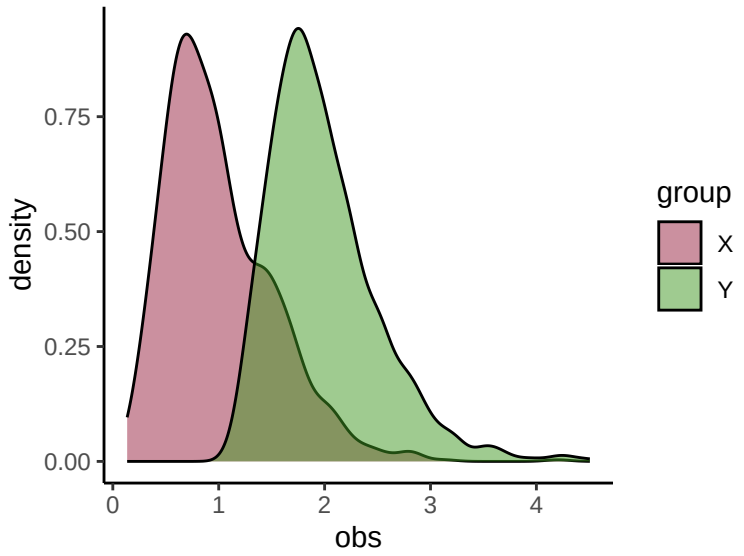
Same same but different (location shift)

Suppose the samples $X_1, X_2, \ldots, X_{n_x}$ and $Y_1, Y_2, \ldots, Y_{n_y}$ are taken from two distinct populations that follow the same kind of distribution but differ in location. That is, $\mu_y = \mu_x + \theta$, where μ_y is the population mean of Y, μ_x is the population mean of X and θ is a **location shift** parameter.

```
N = 1000
theta = 1
loc = data.frame(
obs = c(rgamma(N, 4, 4),
                                rgamma(N, 4, 4)+theta),
group = rep(c("X", "Y"), each = N)
)
loc |> group_by(group) |>
          summarise(Mean = mean(obs),
       SD = sd(obs))
```

```
# A tibble: 2 x 3
  group  Mean    SD
  <chr> <dbl> <dbl>
1 X      1.01 0.510
2 Y      1.99 0.510
```

```
loc |> ggplot() +
aes(x = obs, fill = group) +
geom_density(alpha = 0.5) +
scale_fill_manual(values=c("#992240", "#409922")) +
theme_classic()
```




```
labs(x = "Observed values",
         y = "Density",
         fill = "Sample")
```

```
$x
[1] "Observed values"

$y
[1] "Density"

$fill
[1] "Sample"

attr(,"class")
[1] "labels"
```

Wilcoxon rank-sum test

Let $R_1, R_2, \ldots, R_N$ with $N = n_x + n_y$ be the ranks of combined sample: $X_1, X_2, \ldots, X_{n_x}, Y_1, Y_2, \ldots, Y_{n_Y}$

- For one sample **Wilcoxon signed-rank** test, the ranks re summed over positive side of the differences
- For two sample **Wilcoxon rank-sum** test, the ranks are summed over one of the samples.
- i.e. $W = R_1 + R_2 + \cdots + R_{n_X}$

If H_0 is true, then W should be close to its expected value

$$\mathrm{E} = (W) = \text{Proportion} \times \text{Total rank sum} = \frac{n_x}{N} \times \frac{N(N+1)}{2} = \frac{n_x(N+1)}{2}$$

If W is small (large), we expect $\mu_x < \mu_y (\mu_x > \mu_y)$.

Workflow: Wikcoxon rank-sum test

- **Hypotheses**: $H_0 : \mu_x = \mu_y$ vs $H_1 : \mu_x > \mu_y, \mu_x < \mu_y$ or $\mu_x \neq \mu_y$
- **Assumptions**: X_i and Y_i are independent and follow the same distribution but differ by a shift.
- **Test Statistic**: $W = R_1 + R_2 + \cdots + R_{n_x}$. Under H_0, W follows the $WRS(n_X, n_Y)$ distribution.
- **Observed Test Statistic**: $w = r_1 + r_2 + \cdots + r_{n_x}$.

- **p-value**: $P(W \geqslant w)$ for $H_1 : \mu_x > \mu_y$ or $P(W \leqslant w)$ for $H_1 : \mu_x < \mu_y$ or

$$2P(W \geqslant w) \text{ if } w > \frac{n_x(N+1)}{2} \text{ and } H_1 : \mu_x \neq \mu_y$$
$$2P(W \leqslant w) \text{ if } w < \frac{n_x(N+1)}{2} \text{ and } H_1 : \mu_x \neq \mu_y$$

- **Decision**: If the p-value is less then α, there is evidence against H_0. If the p-value is greater than α, the data are consistent with H_0.

Yield example

The following data yield measurements by two different methods.

```
A = c(32, 29, 35, 28)
B = c(27, 31, 26, 25, 30)
dat = data.frame(
  yield = c(A,B),
  method = c(rep("A", length(A)),
                        rep("B", length(B)))
)
```

If the normality assumptions are in doubt, does the data present sufficient evidence to indicate a difference in the methods A and B?

```
dat = dat |> mutate(rank = rank(yield))
dat
```

```
  yield method rank
1    32      A    8
2    29      A    5
3    35      A    9
4    28      A    4
5    27      B    3
6    31      B    7
7    26      B    2
8    25      B    1
9    30      B    6
```

```
w_A = dat |>
filter(method == "A") |>
pull(rank) |>
sum()
w_A
```

```
[1] 26
```

- **Hypotheses**: $H_0 : \mu_x = \mu_y$
- **Assumptions**: A_i and B_i are independent and follow the same distribution but differ by a shift.
- **Test Statistic**: $W = R_1 + R_2 + \cdots + R_{n_A}$. Under H_0, W follows the $WRS(4, 5)$ distribution.
- **Observed Test Statistic**: $w = 26$. (sum of the ranks associated with method A).
- **p-value**: $2P(W \geqslant w) = 0.19$ because $w = 26 > \frac{n_A(N+1)}{2} = 20$ so we're lookingin the upper tail
- **Decision**: As the p-value is greater than 0.05, the data are consistent with H_0.

```
wilcox.test(A, B) # wilcox.test(yield ~ method, data = dat)
```

```
        Wilcoxon rank sum exact test
```

```
data:  A and B
W = 16, p-value = 0.1905
alternative hypothesis: true location shift is not equal to 0
```

```
t.test(A, B) # t.test(yield ~ method, data = dat)
```

```

        Welch Two Sample t-test

data:  A and B
t = 1.633, df = 5.8232, p-value = 0.1551
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.630468  8.030468
sample estimates:
mean of x mean of y
     31.0      27.8
```

Normal approximation

- We can use a normal approximation to the distribution of test statistic:

$$T = \frac{W - \mathrm{E}(W)}{\sqrt{\mathrm{Var}(W)}} \sim \mathcal{N}(0,1) \text{ approximately}$$

  where $\mathrm{E}(W) = \frac{n_x(N+1)}{2}$ and $\mathrm{Var}(W) = \frac{n_x n_y}{N(N-1)}\left(\sum_{i=1}^{N} r_i^2 - \frac{N(N+1)^2}{4}\right)$
- Our p-value calculations are:
  - p-value $\approx P\left(Z \geqslant \frac{W-\mathrm{E}(W)}{\sqrt{\mathrm{Var}(W)}}\right)$ for $H_1 : \mu_x > \mu_y$
  - p-value $\approx P\left(Z \leqslant \frac{W-\mathrm{E}(W)}{\sqrt{\mathrm{Var}(W)}}\right)$ for $H_1 : \mu_x > \mu_y$
  - p-value $\approx P\left(Z \geqslant \left|\frac{W-\mathrm{E}(W)}{\sqrt{\mathrm{Var}(W)}}\right|\right)$ for $H_1 : \mu_x \neq \mu_y$

Latent heat of fusion

Natrella (1963, pp. 3–23) presents data from two methods that were used in a study of the latent heat of fusion of ice. Both method A (digital method) and Method B (method of mixtures) were conducted with the specimens cooled to -0.72°C. The data represent the change in total heat from -0.72°C to water at 0°C, in calories per gram of mass.

```
A = c(79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04, 79.97,
        80.05, 80.03, 80.02, 80.00, 80.02)
B = c(80.02, 79.94, 79.98, 79.97, 79.97, 80.03, 79.95, 79.97)
heat = data.frame(
        energy = c(A,B),
        method = rep(c("A","B"), c(length(A), length(B))))
```

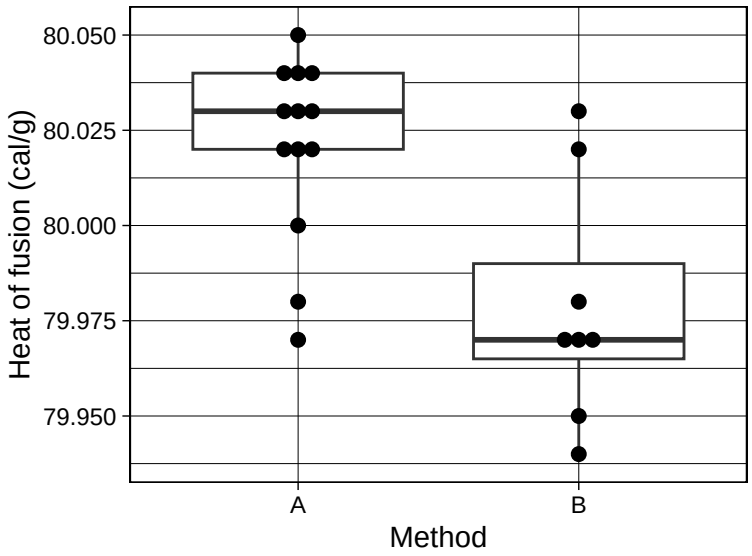
> Does the data support the hypothesis that the electrical method (method A) gives larger results?

```
heat |>
dplyr::group_by(method) |>
dplyr::summarise(
  w = sum(r),
  xbar = mean(energy),
  s = sd(energy),
  n = n()
```

```
) |>
knitr::kable(booktabs=TRUE, digits = 3) |>
 kable_styling(position="center", latex_options = "hold_position")
```

method	w	xbar	s	n
A	2	80.021	0.024	13
B	2	79.979	0.031	8

```
ggplot(heat, aes(x = method, y = energy)) +
  geom_boxplot() +
  geom_dotplot(stackdir = "center",
    binaxis = "y") +
  theme_linedraw() +
  labs(y = "Heat of fusion (cal/g)",
       x = "Method")
```



We have $n_A = 13, n_B = 8$ and $N = 21$

- **Hypotheses**: $H_0 : \mu_x = \mu_y$
- **Assumptions**: A_i and B_i are independent and follow the same distribution but differ by a shift.
- **Test Statistic**: $W = R_1 + R_2 + \cdots + R_{n_A}$. (the sum of the ranks of observations in method A). Under H_0, W follows the $WRS'(13, 8)$ distribution with sizes 13, 8 and with ties as shown.
- **Observed Test Statistic**: $w = r_1 + r_2 + \cdots + r_{n_A} = 180$.
- **p-value**: As the exact $WRS'(13, 8)$ distribution with ties is unknown, we use a normal approximation to this distribution with $\mathrm{E}(W) = \frac{n_x(N+1)}{2} = \frac{13\times13+8+1}{2} = 143$ and $\mathrm{Var}(W) = \frac{n_x n_y}{N(N-1)}\left(\sum_{i=1}^{N} r_i^2 - \frac{N(N+1)}{4}\right) = \frac{13(8)(3293.5-2541)}{21(20)} = 186.33$

$$P(W \geqslant w) \simeq P\left(Z \geqslant \frac{w - \mathrm{E}(W)}{\sqrt{\mathrm{Var}(W)}}\right) = P\left(Z \geqslant \frac{180 - 143}{\sqrt{186.33}}\right) = P(Z > 2.7) = 0.003$$

- **Decision**: As the p-value is greater than 0.05, the data are consistent with H_0.

```
wilcox.test(A, B, alternative = 'greater', correct = FALSE)
```

```
        Wilcoxon rank sum test
```

```
data:  A and B
W = 89, p-value = 0.003359
alternative hypothesis: true location shift is greater than 0
```

```
t.test(A, B, alternative = 'greater')
```

```

        Welch Two Sample t-test

data:  A and B
t = 3.2499, df = 12.027, p-value = 0.00347
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.01897943        Inf
sample estimates:
mean of x mean of y
 80.02077  79.97875
```

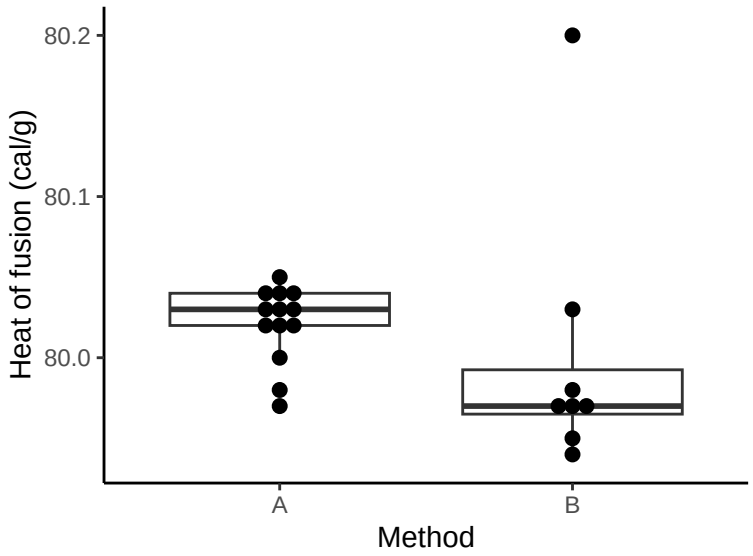
Robustness properties

What happens if there is an outlier in the data?

```
# change the first value for the B method
heat1 = heat
heat1$energy[14] = 80.20 # instead of 80.02
# recalculate ranks
heat1 = heat1 |> dplyr::mutate(
        r = rank(energy)
)
heat1 |>
dplyr::group_by(method) |>
dplyr::summarise(
  w = sum(r),
  Mean = mean(energy),
  SD = sd(energy),
  n = n()
) |>
knitr::kable(digits = 3, booktabs = TRUE) |>
  kable_styling(position="center", latex_options = "hold_position")
```

method	w	Mean	SD	n
A	171.5	80.021	0.024	13
B	59.5	80.001	0.085	8

```
ggplot(heat1) +
  aes(x = method, y = energy) +
  geom_boxplot() +
  theme_classic() +
  geom_dotplot(stackdir = "center",
        binaxis = "y") +
  labs(y = "Heat of fusion (cal/g)",
    x = "Method")
```




```
wilcox.test(energy ~ method, data = heat1, alternative = 'greater', correct =
    FALSE)
```

```
        Wilcoxon rank sum test

data:  energy by method
W = 80.5, p-value = 0.01859
alternative hypothesis: true location shift is greater than 0
```

```
t.test(energy ~ method, data = heat1, alternative = 'greater')
```

```
        Welch Two Sample t-test

data:  energy by method
t = 0.63712, df = 7.6977, p-value = 0.2713
alternative hypothesis: true difference in means between group A and group B
    is greater than 0
95 percent confidence interval:
 -0.03774242         Inf
sample estimates:
mean in group A mean in group B
       80.02077        80.00125
```

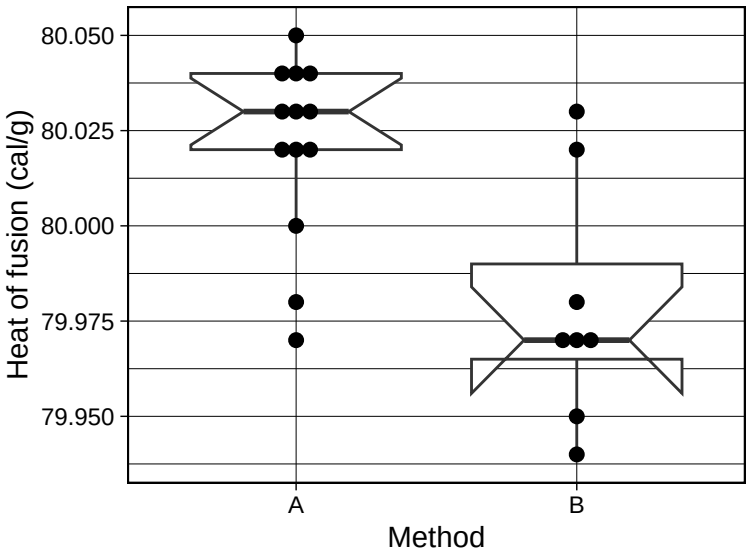
A heuristic for testing for differences: notched boxplot

The upper and lower edges of the notches are at

$$\text{median} \pm 1.58 \times \frac{\text{IQR}}{\sqrt{n}}$$

Rule of thumb:: If the notches of two boxes do not overlap, this suggests that the medians are significantly different Mcgill et al. (1978).

```
ggplot(heat, aes(x = method, y = energy)) +
geom_boxplot(notch = TRUE) +
geom_dotplot(stackdir = "center",
                 binaxis = "y") +
theme_linedraw() +
labs(y = "Heat of fusion (cal/g)",
     x = "Method")
```



16
Permutation tests

16.1 Lady tasting tea

Lady tasting tea

Given a cup of tea with milk, a lady claims she can discriminate as to whether milk or tea was first added to the cup.

How could we test this claim? What information would we need?

Fisher proposed preparing 8 cups of tea
- 4 cups where tea was added before milk
- 4 cups where milk was added before tea

The lady would then be randomly given the cups of tea and asked to identify the 4 where tea was added before milk.
We would need to record:
- Which cups had tea or milk added first (**truth**).
- Which cups the lady claimed had tea or milk added first (**predicted**).

Lady tasting tea - hypothesis

For Fisher's experiment we were left with two categorical variables.

Truth = Milk, Tea, Tea, Milk, Tea, Tea, Milk, Milk
Prediction = Milk, Tea, Tea, Milk, Tea, Tea, Milk, Milk

Asking the question: **Are the predictions independent of the truth?** or

H_0 : Lady cannot take the difference vs H_1 : Lady can taste the difference

Our **test statistic** is the number of predictions she gets correct,

T = Number of tea before milk cups correctly identified

Calculating significance

The order of the cups was random, therefore there are

$$8! = 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 40,320$$

The order of the cups was random, therefore there are $\binom{8}{4} = 70$ ways to select which 4 cups of tea had the tea added before milk (when order doesn't matter).
We can look at all 70 ways and calculate how often we see a test statistic of 0,1,2,3 or 4

Number of correct: t_i	0	1	2	3	4	
How many ways: f_i	$\binom{4}{0}\binom{4}{4} = 1$	$\binom{4}{1}\binom{4}{3} = 16$	$\binom{4}{2}\binom{4}{2} = 36$	$\binom{4}{1}\binom{4}{3} = 16$	$\binom{4}{0}\binom{4}{4} = 1$	70
Corresponding probability: p_i	$\frac{1}{70}$	$\frac{16}{70}$	$\frac{36}{70}$	$\frac{16}{70}$	$\frac{1}{70}$	1

$$P(\text{Getting at least 4 correct}) = P(T = 4) = \frac{1}{70} = 0.014$$

Permutations

We could also consider all 40,320 different orderings (permutations) of 8 cups of tea.

```
library(arrangements)
permute_8 = permutations(8)
head(permute_8, 6) # look at the "first" 6 permutations
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    2    3    4    5    6    7    8
[2,]    1    2    3    4    5    6    8    7
[3,]    1    2    3    4    5    7    6    8
[4,]    1    2    3    4    5    7    8    6
[5,]    1    2    3    4    5    8    6    7
[6,]    1    2    3    4    5    8    7    6
```

```
tail(permute_8, 6) # look at the "last" 6 permutations
```

```
         [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[40315,]    8    7    6    5    4    1    2    3
[40316,]    8    7    6    5    4    1    3    2
[40317,]    8    7    6    5    4    2    1    3
[40318,]    8    7    6    5    4    2    3    1
[40319,]    8    7    6    5    4    3    1    2
[40320,]    8    7    6    5    4    3    2    1
```

If you wanted to roll your own function to calculate all possible permutations:

```
permutation_fn = function(n){
  if(n==1){
    return(matrix(1))
  } else {
    sp ← permutation_fn(n-1)
    p ← nrow(sp)
    A ← matrix(nrow = n*p, ncol = n)
```

```
    for(i in 1:n){
      A[(i-1)*p + 1:p, ] ← cbind(i, sp + (sp ⩾ i))
    }
    return(A)
  }
}
```

Use the `permutations()` function on the `truth` vector:

```
truth = c("milk","tea","tea","milk","tea","tea","milk","milk")
permute_guess = permutations(truth)
permute_guess[92, ] # 92nd permutation
```

```
[1] "milk" "tea"  "tea"  "milk" "milk" "milk" "tea"  "tea"
```

We can check if a particular sequence of tea cups is **identical** to the true sequence:

```
identical(truth,truth)
```

```
[1] TRUE
```

```
identical(permute_guess[92,], truth)
```

```
[1] FALSE
```

Exact p-value

We can calculate the exact p-value by looking across all permutations:

```
B = nrow(permute_guess)
check_correct = vector("numeric", length = B)
for(i in 1:B) {
  check_correct[i] = identical(permute_guess[i,], truth)
}
c(sum(check_correct), mean(check_correct))
```

```
[1] 576.00000000   0.01428571
```

The p-value is the same as we get using Fisher's exact test!

```
truth = c("milk", "tea", "tea", "milk", "tea", "tea", "milk", "milk")
predicted = c("milk", "tea", "tea", "milk", "tea", "tea", "milk", "milk")
tea_mat = table(truth, predicted)
fisher.test(tea_mat, alternative = "greater")$p.value
```

```
[1] 0.01428571
```

Approximate p-value

Often it's not feasible to consider all $n!$ permutations, so we can `sample()` a selection of them.

```
set.seed(88)
truth = c("milk","tea","tea","milk","tea","tea","milk","milk")
B = 10000
result = vector(length = B) # initialise outside the loop
for(i in 1:B){
  guess = sample(truth, size = 8, replace = FALSE) # does the permutation
  result[i] = identical(guess, truth)
}
mean(result)
```

```
[1] 0.0141
```

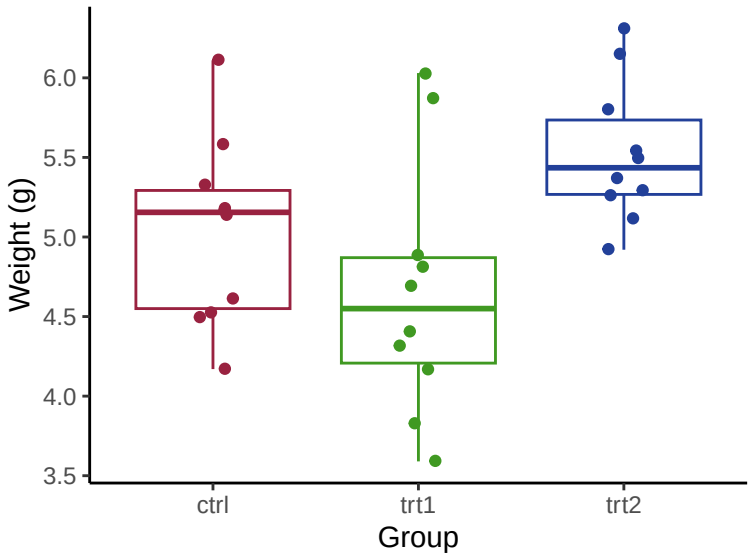
Pretty close to the exact p-value.

16.2 Permutation test: two independent samples

Plant growth

The `Plantgrowth` data has results from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment conditions (Dobson, 1983, Table 7.1).

```
# built into R, make it available
data("PlantGrowth")
PlantGrowth |> ggplot() +
  aes(y = weight, x = group,
      colour = group) +
  geom_boxplot(coef = 10) +
  geom_jitter(width = 0.1) +
  theme_classic() +
  scale_colour_manual(values=c("#992240","#409922","#224099")) +
  theme(legend.position = "none") +
  labs(y = "Weight (g)", x = "Group")
```

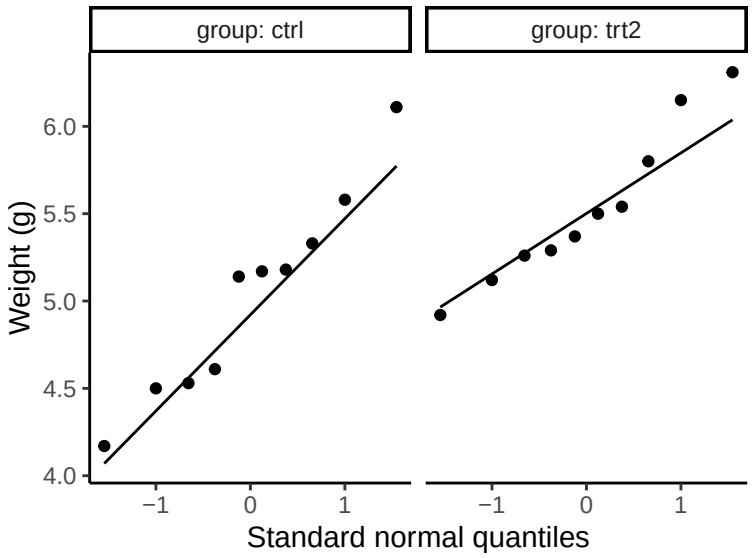


We want to compare the **control** group to the **treatment** 2 group.

```
dat = PlantGrowth |> filter(group %in% c("ctrl", "trt2"))
```

Checking for normality: QQ plot

```
dat |>
  ggplot() + aes(sample = weight) +
  geom_qq() + geom_qq_line() +
  theme_classic() +
  facet_grid(cols = vars(group), labeller = label_both) +
  labs(y = "Weight (g)", x = "Standard normal quantiles")
```



What do our "standard" methods say?

Two-sample t-test

```
t.test(weight ~ group, data = dat, var.equal = TRUE)
```

```
        Two Sample t-test

data:  weight by group
t = -2.134, df = 18, p-value = 0.04685
alternative hypothesis: true difference in means between group ctrl and group
    trt2 is not equal to 0
95 percent confidence interval:
 -0.980338117 -0.007661883
sample estimates:
mean in group ctrl mean in group trt2
             5.032              5.526
```

Wilcoxon Rank-Sum Test

```
wilcox.test(weight ~ group, data = dat)
```

```
        Wilcoxon rank sum exact test

data:  weight by group
W = 25, p-value = 0.06301
alternative hypothesis: true location shift is not equal to 0
```

Extracting information from `t.test` objects

```
tt = t.test(weight ~ group, data = dat, var.equal = TRUE)
tt
```

```
        Two Sample t-test

data:  weight by group
t = -2.134, df = 18, p-value = 0.04685
alternative hypothesis: true difference in means between group ctrl and group
    trt2 is not equal to 0
95 percent confidence interval:
```

```
 -0.980338117 -0.007661883
sample estimates:
mean in group ctrl mean in group trt2
             5.032              5.526
```

```
names(tt)
```

```
 [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
 [6] "null.value"  "stderr"      "alternative" "method"      "data.name"
```

```
tt$statistic
```

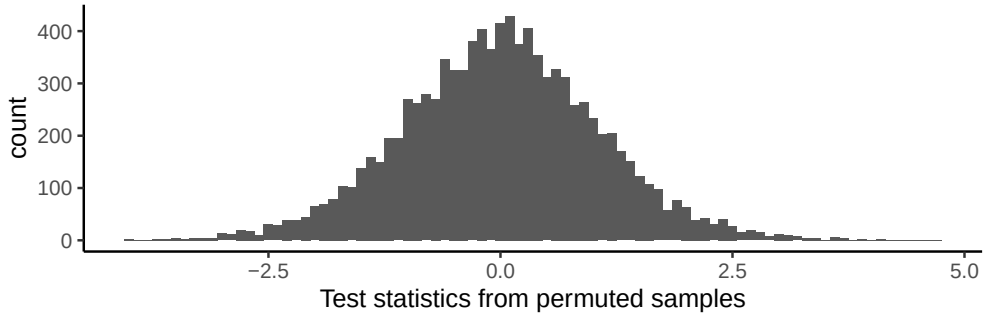
```
      t
-2.13402
```

Permutation test

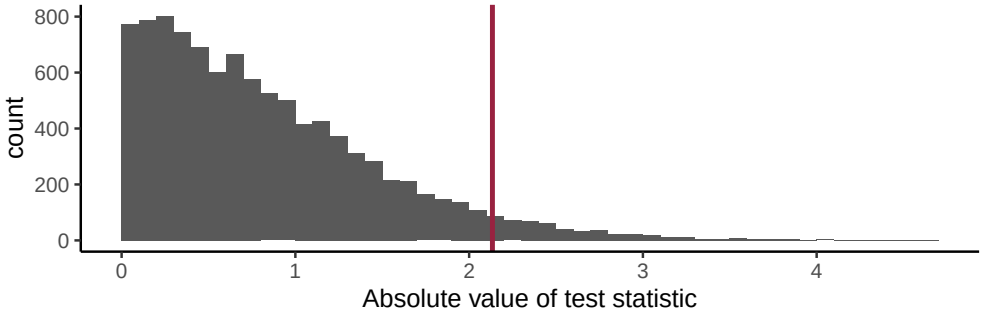
Permute the **class labels** (many times) and see what values we get for the t-test statistic

```
B = 10000 # number of permuted samples we will consider
permuted_dat = dat # make a copy of the data
t_null = vector("numeric", B) # initialise outside loop
for(i in 1:B) {
  permuted_dat$group = sample(dat$group) # this does the permutation
  t_null[i] = t.test(weight ~ group, data = permuted_dat)$statistic
}
```

```
t_null |> data.frame() |>
  ggplot() +
  aes(x = t_null) +
  theme_classic() +
  geom_histogram(binwidth = 0.1) +
  labs(
    x = "Test statistics from permuted samples"
  )
```




```
data.frame(abs_t_null = abs(t_null)) |>
  ggplot() +
  aes(x = abs_t_null) +
  theme_classic() +
  geom_histogram(binwidth = 0.1,
                 boundary = 0) +
  geom_vline(
    xintercept = abs(tt$statistic),
    col = "#992240", lwd = 1) +
  labs(
    x = "Absolute value of test statistic"
  )
```



What proportion of test statistics from randomly permuted data are more extreme than the test statistic we observed?

```
mean(abs(t_null) >= abs(tt$statistic))
```

```
[1] 0.0501
```

This is our permutation test p-value

Permutation tests

- The two-sample t-test and the permutation test gave similar p-values, but this won't always be the case.
- The two-sample t-test is a **parametric** test where the test statistic is assumed to follow some distribution ($t_{n_x+n_y-2}$ distribution)
- The permutation test considers the $(n_1 + n_2)!$ permutations of the labels (or a random subset to save computation time) from a single instance of the data (the $n_1 + n_2$ observations).
- The permutation test only assumes that the observations $X_1, X_2, \ldots, X_{n_x}, Y_1, Y_2, \ldots, Y_{n_y}$ are *exchangeable* under H_0, that is, swapping labels on observations keeps the data just as likely as the original.
- The permutation test may use the t-test **test statistic** but it does not use the t **distribution**

Latent heat of fusion

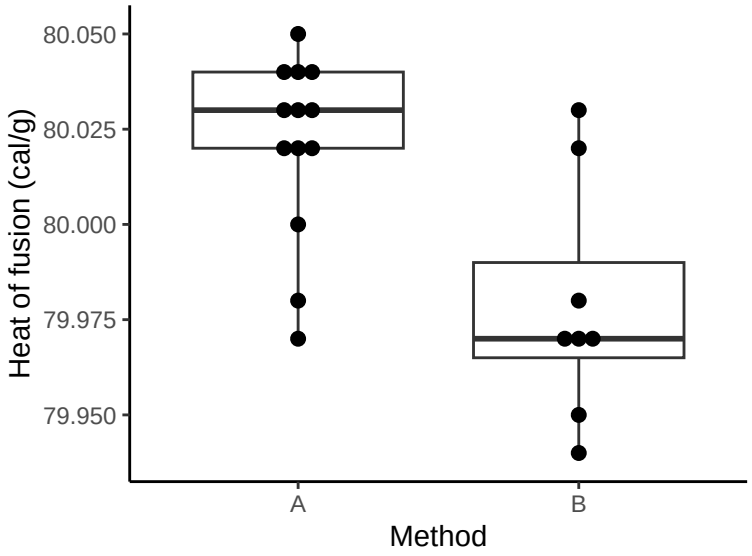
Natrella (1963, pp. 3–23) presents data from two methods that were used in a study of the latent heat of fusion of ice. Both method A (digital method) and Method B (method of mixtures) were conducted with the specimens cooled to -0.72°C. The data represent the change in total heat from -0.72°C to water at 0°C, in calories per gram of mass.

> **Does the data support the hypothesis that the electrical method (method A) gives larger results?**

```
A = c(79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04,
      79.97, 80.05, 80.03, 80.02, 80.00, 80.02)
B = c(80.02, 79.94, 79.98, 79.97, 79.97, 80.03, 79.95,
      79.97)
heat = data.frame(
  energy = c(A,B),
  method = rep(c("A","B"), c(length(A), length(B)))
)
```

```
heat |> ggplot() +
  aes(x = method, y = energy) +
  geom_boxplot(coef = 10) +
  geom_dotplot(stackdir = "center", binaxis = "y") +
```

```
  theme_classic() +
  labs(y = "Heat of fusion (cal/g)", x = "Method")
```




```
tt = t.test(energy ~ method, data = heat, alternative = "greater")
tt
```

```
        Welch Two Sample t-test

data:  energy by method
t = 3.2499, df = 12.027, p-value = 0.00347
alternative hypothesis: true difference in means between group A and group B
    is greater than 0
95 percent confidence interval:
 0.01897943        Inf
sample estimates:
mean in group A mean in group B
       80.02077        79.97875
```

```
t0_original = tt$statistic
t0_original
```

```
       t
3.249867
```

How many permutations of the class label are there?

```
n = nrow(heat)
n
```

```
[1] 21
```

```
factorial(n)
```

```
[1] 5.109094e+19
```

```
B = 10000 # number of permuted samples we will consider
permuted_heat = heat # make a copy of the data
t_null = vector("numeric", B) # initialise outside loop
for(i in 1:B) {
```

```
  permuted_heat$method = sample(heat$method) # this does the permutation
  t_null[i] = t.test(energy ~ method, data = permuted_heat)$statistic
}
mean(t_null >= t0_original)
```

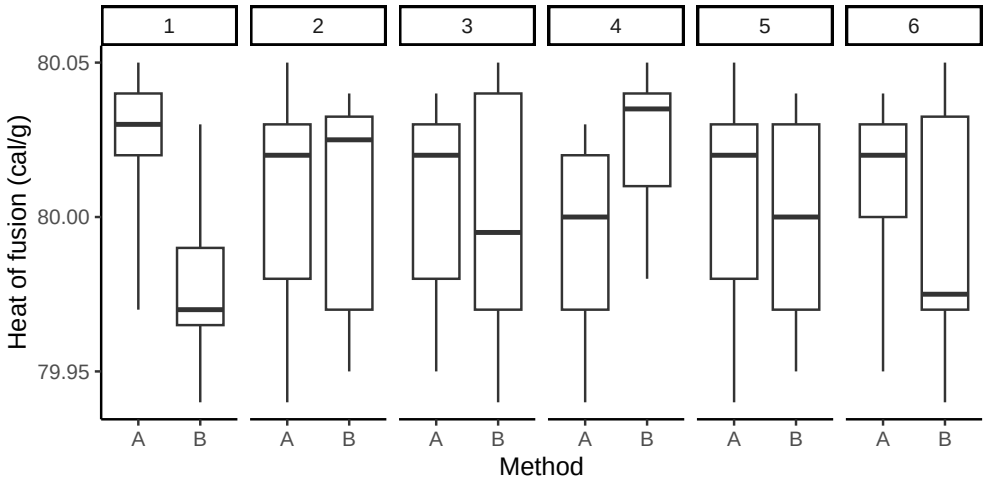
```
[1] 0.0026
```

A closer look at a few permutated samples

```
perm_heat = heat
perm_heat$id = 1
for(i in 2:6){
  temp = heat
  temp$method = sample(temp$method)
  temp$id = i
  perm_heat = rbind(perm_heat, temp)
}
perm_heat |>
  group_by(id) |>
  summarise(
    t_stat = t.test(
      energy[method=="A"],
      energy[method=="B"])$statistic
  )
```

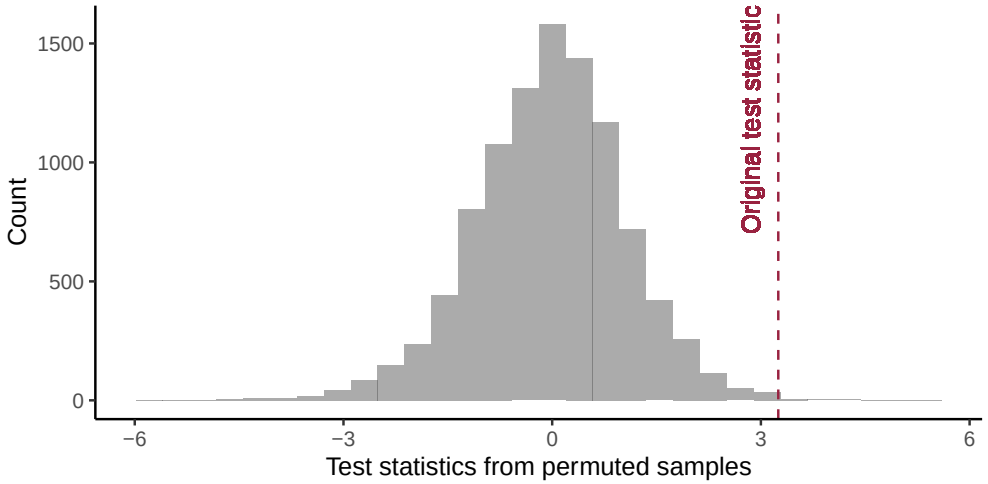
```
# A tibble: 6 x 2
     id t_stat
  <dbl>  <dbl>
1     1  3.25
2     2 -0.152
3     3  0.456
4     4 -2.14
5     5  0.629
6     6  1.09
```

```
perm_heat |> ggplot() +
        aes(x = method, y = energy) +
  geom_boxplot(coef = 10) +
  theme_classic() +
  scale_y_continuous(n.breaks = 3) +
  facet_wrap(vars(id), ncol = 6) +
  labs(y = "Heat of fusion (cal/g)",
       x = "Method")
```




```
t_null |> data.frame() |> ggplot() +
        aes(x = t_null) +
  geom_histogram(alpha=0.5) +
```

```
  geom_vline(xintercept = t0_original, colour = "#992240", linetype = "dashed"
      ) +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "#992240", angle = 90, hjust = 1, vjust = -1) +
  theme_classic() +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



What about outliers?

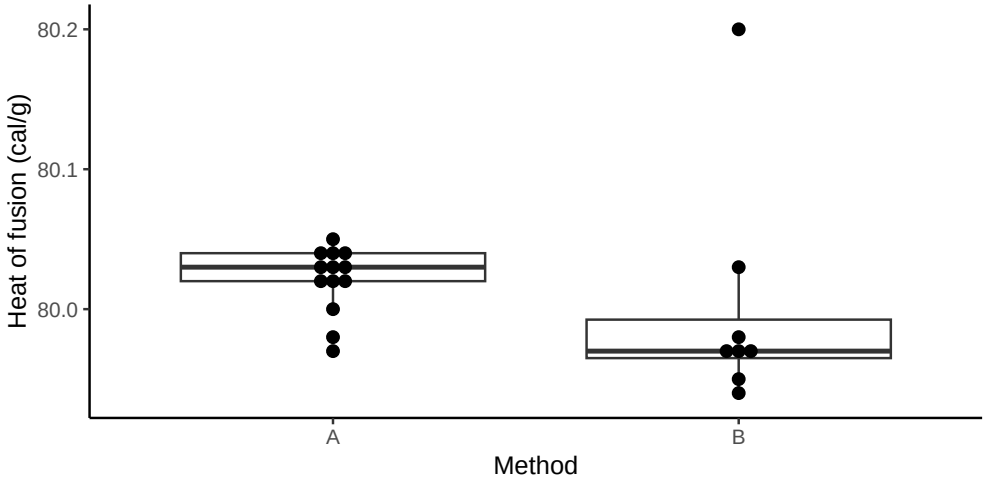
What happens if there is an outlier in the data? Below we change the first value for the B method from 80.02 to 80.20.

```
heat1 = heat
heat1$energy[14]
```

```
[1] 80.02
```

```
# edit 80.02 -> 80.20
heat1$energy[14] = 80.20
```

```
ggplot(heat1) + aes(x = method, y = energy) +
  geom_boxplot() +
  geom_dotplot(stackdir = "center",
               binaxis = "y") +
  theme_classic() +
  labs(y = "Heat of fusion (cal/g)",
       x = "Method")
```



Our "standard" test eesults

Wilcoxon Rank-Sum Test

```
wilcox.test(energy ~ method, data = heat1, alternative = "greater", correct =
    FALSE)
```

```
        Wilcoxon rank sum test

data:  energy by method
W = 80.5, p-value = 0.01859
alternative hypothesis: true location shift is greater than 0
```

Welsh Two-Sample t-test

```
t.test(energy ~ method, data = heat1, alternative = "greater")
```

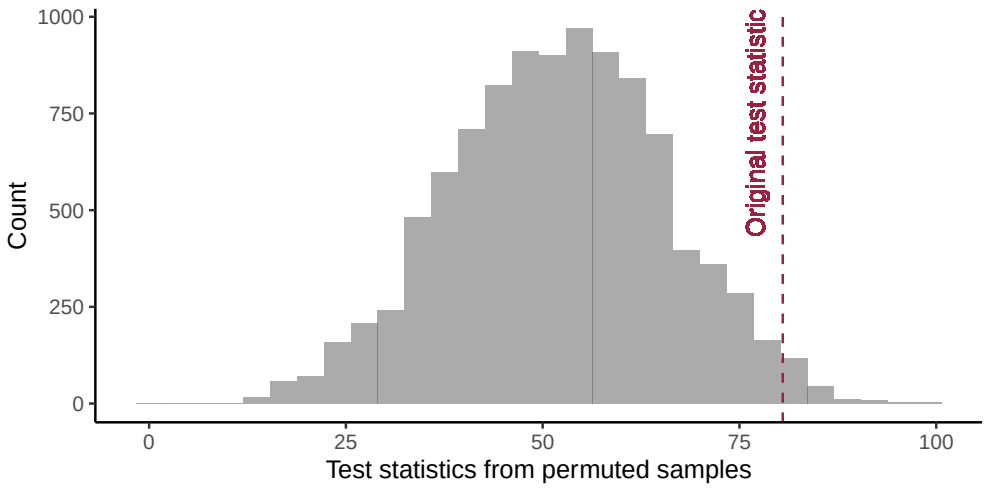
```
        Welch Two Sample t-test

data:  energy by method
t = 0.63712, df = 7.6977, p-value = 0.2713
alternative hypothesis: true difference in means between group A and group B
    is greater than 0
95 percent confidence interval:
 -0.03774242         Inf
sample estimates:
mean in group A mean in group B
       80.02077        80.00125
```

Permutation test using the Wilcoxon rank-sum statistic

```
t0_original = wilcox.test(energy ~ method, data = heat1)$statistic
set.seed(88)
B = 10000
permuted_heat1 = heat1
t_null = vector("numeric", B)
for(i in 1:B){
  permuted_heat1$method = sample(heat1$method)
  t_null[i] = wilcox.test(energy ~ method, data = permuted_heat1)$statistic
}
mean(t_null >= t0_original)
```

```
[1] 0.019
```

```
t_null |> data.frame() |> ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "#992240", linetype = "dashed"
      ) +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "#992240", angle = 90, hjust = 1, vjust = -1) +
  theme_classic() +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



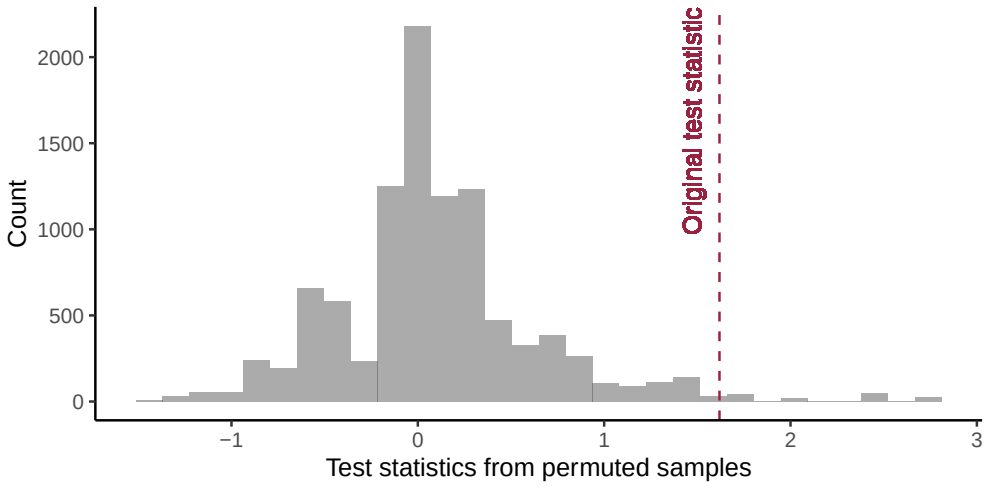
Robustly standardised difference in medians

$$T = \frac{\tilde{x} - \tilde{y}}{\text{MAD}(x) + \text{MAD}(y)}$$

```
median_a = median(heat1$energy[heat1$method=="A"])
median_b = median(heat1$energy[heat1$method=="B"])
mad_a = mad(heat1$energy[heat1$method=="A"])
mad_b = mad(heat1$energy[heat1$method=="B"])
t0_original = (median_a - median_b)/(mad_a + mad_b)

B = 10000
t_null = vector("numeric", B)
for(i in 1:B){
  permuted_heat1$method = sample(heat1$method)
  median_a = median(permuted_heat1$energy[permuted_heat1$method=="A"])
  median_b = median(permuted_heat1$energy[permuted_heat1$method=="B"])
  mad_a = mad(permuted_heat1$energy[permuted_heat1$method=="A"])
  mad_b = mad(permuted_heat1$energy[permuted_heat1$method=="B"])
  t_null[i] = (median_a - median_b)/(mad_a + mad_b)
}
mean(t_null >= t0_original)
```

```
[1] 0.0165
```

```
t_null |> data.frame() |> ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "#992240", linetype = "dashed"
      ) +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "#992240", angle = 90, hjust = 1, vjust = -1) +
  theme_classic() +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



16.3 Permutation Test: Paired Samples

Paired sample tests?

Can we use permutation tests if we are testing for a shift in location by sampling from one population?

- For paired tests we think about the differences, $d_i = x_i - y_i$.
- For the Wilcoxon signed-rank test we had a test statistic involving

$$\sum_{i:d_i>0} r_i \times \text{sign}(d_i)$$

- We could also think of a statistic where we used the values of the differences,

$$\sum_{i=1}^{n} |d_i| \times \text{sign}(d_i)$$

- For a permutation test permute all possible $\text{sign}(d_i)$.

Smoking

Blood samples from 11 individuals before and after they smoked a cigarette are used to measure aggregation of blood platelets.
Is the aggregation affected by smoking?

```
before = c(25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28)
after =  c(27, 29, 37, 36, 46, 82, 57, 80, 61, 59, 43)
d = after - before
t.test(d)
```

```
        One Sample t-test

data:  d
t = 2.9065, df = 10, p-value = 0.01566
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  1.97332 14.93577
sample estimates:
mean of x
 8.454545
```

There are $2^{11} = 2048$ ways to permutations of sign of 11 differences.

```
sign_permute = permutations(c(-1,1), 11, replace = TRUE)
dim(sign_permute)
```

```
[1] 2048   11
```

```
head(sign_permute)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
[1,]   -1   -1   -1   -1   -1   -1   -1   -1   -1    -1    -1
[2,]   -1   -1   -1   -1   -1   -1   -1   -1   -1    -1     1
[3,]   -1   -1   -1   -1   -1   -1   -1   -1   -1     1    -1
[4,]   -1   -1   -1   -1   -1   -1   -1   -1   -1     1     1
[5,]   -1   -1   -1   -1   -1   -1   -1   -1    1    -1    -1
[6,]   -1   -1   -1   -1   -1   -1   -1   -1    1    -1     1
```

```
(t0_original = mean(d)/sd(d)*sqrt(length(d)))
```

```
[1] 2.906534
```

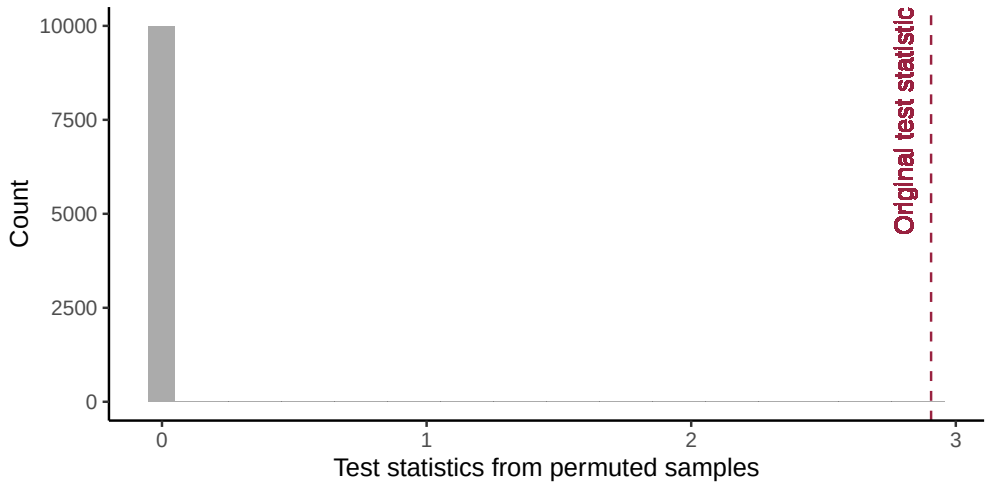
```
n = length(d)
B = nrow(sign_permute)
t_null = vector("numeric", B)
for(i in 1:nrow(sign_permute)){
  d_permute = d*sign_permute[i,]
  t_null[i] = mean(d_permute)/sd(d_permute)*sqrt(n)
}
mean(abs(t_null) >= abs(t0_original))
```

```
[1] 0.01660156
```

```
t.test(d)$p.value
```

```
[1] 0.01565739
```

```
t_null |> data.frame() |> ggplot() + aes(x = t_null) +
  geom_histogram(alpha=0.5) +
  geom_vline(xintercept = t0_original, colour = "#992240", linetype = "dashed"
      ) +
  geom_text(aes(x = t0_original, label = "Original test statistic", y = Inf),
            colour = "#992240", angle = 90, hjust = 1, vjust = -1) +
  theme_classic() +
  labs(x = "Test statistics from permuted samples", y = "Count")
```



17
Bootstrap

17.1 Speed of light

Speed of light

- Simon Newcomb measured the time required for light to travel from his laboratory on the Potomac River to a mirror at the base of the Washington Monument and back, a total distance of about 7400 meters.
- He performed this experiment 66 times.
- These measurements were used to estimate the speed of light.

Newcomb's measurements of the passage time of light, made July 24, 1882 to September 5, 1882. The values $\times^{-3}+24.8$ are Newcomb's measurements recorded in millionths of a second for observations of light passing over a distance of 3721 m and back, from Fort Myer on the west bank of the Potomac to a fixed mirror at the base of the Washington monument. The "true" value is 33.02. Stigler (1977, Table 5)

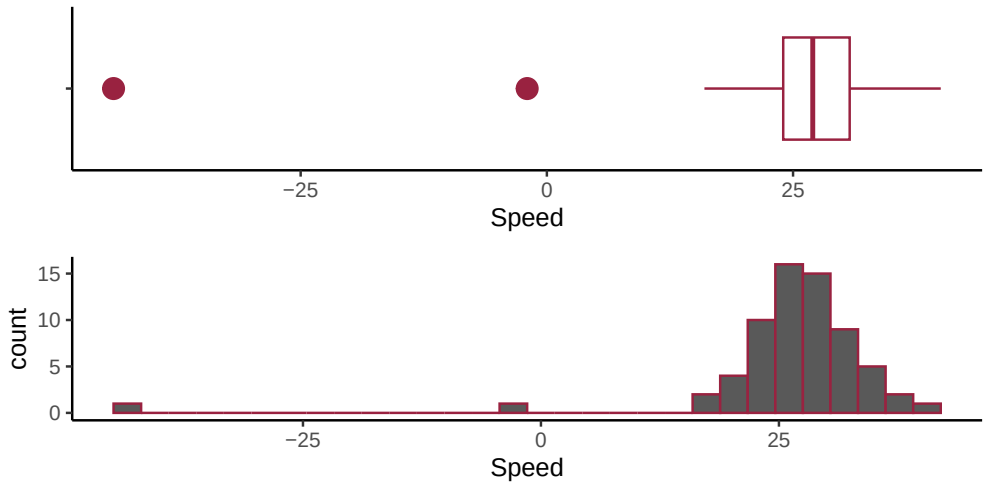
```
library(readr)
speed_file = read_csv("https://raw.githubusercontent.com/DATA2002/data/master/
    speed_of_light.txt")
speed = speed_file$Speed_of_Light
mean(speed)
```

```
[1] 26.21212
```

```
median(speed)
```

```
[1] 27
```

```
library(cowplot)
p1 = ggplot(speed_file) + aes(x = "", y = Speed_of_Light) +
  geom_boxplot(colour = "#992240", outlier.size = 4) +
  theme_classic() +
  labs(x = "", y = "Speed") + coord_flip()
p2 = ggplot(speed_file) + aes(x = Speed_of_Light) +
  geom_histogram(colour = "#992240") +
  theme_classic() +
  labs(x = "Speed")
cowplot::plot_grid(p1, p2, ncol = 1, align = "v")
```



17.2 Confidence Intervals

Estimation vs hypothesis testing

Estimation

- A population parameter is unknown.
- Use the sample statistics to generate estimates of the population parameter.

Hypothesis Testing

- Explicit statement (or hypothesis) regarding the population parameter.
- Test statistics are generated which will either support or reject the null hypothesis.

Confidence intervals

- We should avoid reporting just a point estimate for a sample.
- We should always include a measure of variability:

$$\hat{\theta} \pm \text{margin of error}$$

  where $\hat{\theta}$ is the point estimate (e.g. sample mean, $\overline{X}$).
- The margin of error usually takes the form

$$\text{margin of error} = \text{critical value} \times \text{SE}(\hat{\theta})$$

  where the critical value is some quantile from an appropriate distribution (e.g. $z_{\frac{\alpha}{2}}$ or $t_\nu(\frac{\alpha}{2})$) and $\text{SE}(\theta)$ is the standard error of the point estimate (e.g. $\text{SE}(\overline{X}) = \sigma/\sqrt{n}$).
- The *point* estimate $\hat{\theta}$ (say $\overline{x}$) of a parameter θ (say μ) does not show its variability across samples.
- To show such estimation precision, we should find an *interval* estimate.

Definition

Let $\hat{\theta}_L$ and $\hat{\theta}_R$ be two statistics. If

$$P(\hat{\theta}_L \leqslant \theta \leqslant \hat{\theta}_R) = 1 - \alpha,$$

then the random interval $[\hat{\theta}_L, \hat{\theta}_R]$ is called a $100(1-\alpha)\%$ *confidence interval* (CI) for θ, and $100(1-\alpha)\%$ is called the *confidence level* of the interval.

- In general, the α may be chosen to be 0.01, 0.05, 0.10, etc, and then we get 99%, 95%, 90% confidence interval accordingly.

Confidence intervals for the mean

Let $X_1, X_2, \ldots, X_n$ be a random sample from normal population and $X_i \sim \mathcal{N}(\mu, \sigma^2)$, where σ^2 is unknown.
Then $\frac{\overline{X}-\mu}{S/\sqrt{n}} \sim t_{n-1}$ and

$$P\left(-c < \frac{\overline{X}-\mu}{S/\sqrt{n}} < c\right) = 1-\alpha$$
$$P\left(-\frac{cS}{\sqrt{n}} < \mu - \overline{X} < \frac{cS}{\sqrt{n}}\right) = 1-\alpha$$
$$P\left(\overline{X} - \frac{cS}{\sqrt{n}} < \mu < \overline{X} + \frac{cS}{\sqrt{n}}\right) = 1-\alpha$$

Meaning of the confidence interval

Suppose a 95% confidence interval for the mean μ is (a, b).

- This does **not** mean that 95% of the means μ are in (a, b), that is $P(a < \mu < b) = 0.95$ since μ is a **fixed** but unknown parameter.
- It also does **not** mean $P(a < \overline{X} < b) = 0.95$, where $\overline{X}$ is the sample mean since the CI is for the true mean μ not the sample mean $\overline{X}$.

It **does** mean that if we draw a large number of random samples and compute for each sample a 95% CI, about 95% of these CIs will contain μ. It **can** be described as a **range of possible values** for the population parameter.

Confidence intervals

A confidence interval is a statement about the underlying population parameter.

- Formally, for a 95% confidence interval, 95% of all possible random samples will contain the population mean, leading to 95% **confidence** that a single interval contains the true mean.
- In reality, a specific interval either contains the mean or it doesn't. We just do not know which one is true; but we have 95% **confidence** that it does.

In this context **confidence** isn't the same as **probability**.

Distributional assumptions

> **What happens if your data does not follow a normal distribution?**

1. Guess the distribution of the data and use this distribution to calculate critical values for confidence levels. **Risky**.
2. Use **bootstrap resampling** to empirically model the distribution of the data.

17.3 Bootstrapping

Bootstrapping resampling

Bootstrapping is a computational process that allows us to as make inferences about the population where no information is available about the population. Bootstrap methods take their name from the idea of "lifting yourself up by your bootstraps" -

moving up without any additional outside help. The name was introduced by Efron (1979).

> "in the absence of any other knowledge about a population, the distribution of values found in a sample of size n from the population is the best guide to the distribution in the population. Therefore, to approximate what would happen if the population was resampled it is sensible to resample the sample." Manly (2007, p. 41)

The classic approach to bootstrapping is to **repeatedly resample** from the sample (with replacement).

Speed of light

- Simon Newcomb measured the time required for light to travel from his laboratory on the Potomac River to a mirror at the base of the Washington Monument and back, a total distance of about 7400 meters.
- He performed this experiment 66 times (66 observations).
- These measurements were used to estimate the speed of light.
- What if we approximated **sampling from the population** by **sampling from this sample**?

Bootstrapping speed of light measurements

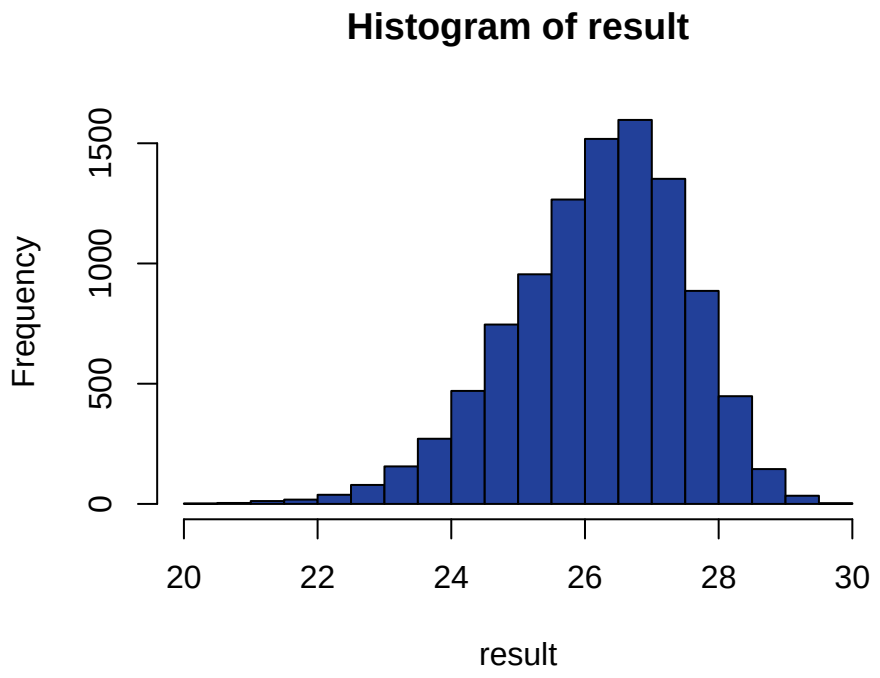
```
mean(speed)
```

```
[1] 26.21212
```

```
set.seed(88)
B = 10000
result = vector("numeric", length = B)
for(i in 1:B){
  newData = sample(speed, replace = TRUE)
  result[i] = mean(newData)
}
round(head(result), 2)
```

```
[1] 22.98 26.15 27.38 24.67 26.11 25.20
```

```
hist(result, col = "#224099")
```



Bootstrap confidence intervals

Efron (1979) proposed that the bootstrap confidence interval be the quantiles from the bootstrap distribution. In general, (θ_L^*, θ_U^*) are the bounds of the $100(1-\alpha)$ bootstrap CI where θ_L^* is the $\frac{\alpha}{2}$ quantile from the bootstrap distribution and θ_U^* is the $1-\frac{\alpha}{2}$ quantile from the bootstrap distribution.
If `result` has our bootstrap estimates then we can get a 95% confidence interval using:

```
quantile(result, c(0.025, 0.975))
```

```
    2.5%    97.5%
23.36364 28.37917
```

Speed of light

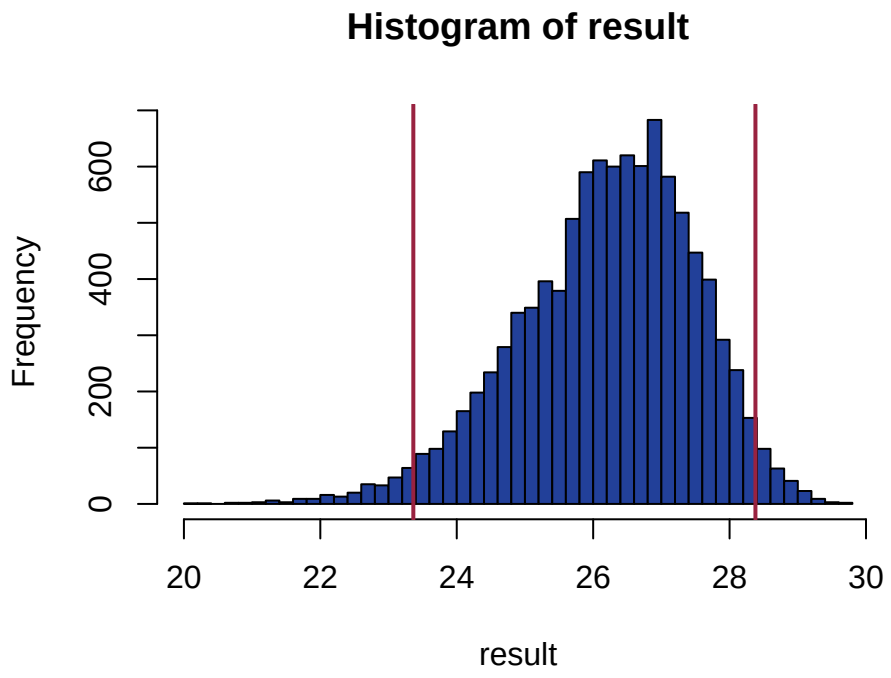
```
CI = quantile(result, c(0.025, 0.975))
CI
```

```
    2.5%    97.5%
23.36364 28.37917
```

```
CI - mean(speed)
```

```
     2.5%     97.5%
-2.848485  2.167045
```

```
hist(result, breaks = 50,
     col = "#224099")
abline(v = CI, col = "#992240", lwd = 2)
```



Compare with the confidence interval using the t distribution.

```
xbar = mean(speed)
n = length(speed)
se = sd(speed)/sqrt(n)
c(xbar, n, se)
```

```
[1] 26.212121 66.000000  1.322658
```

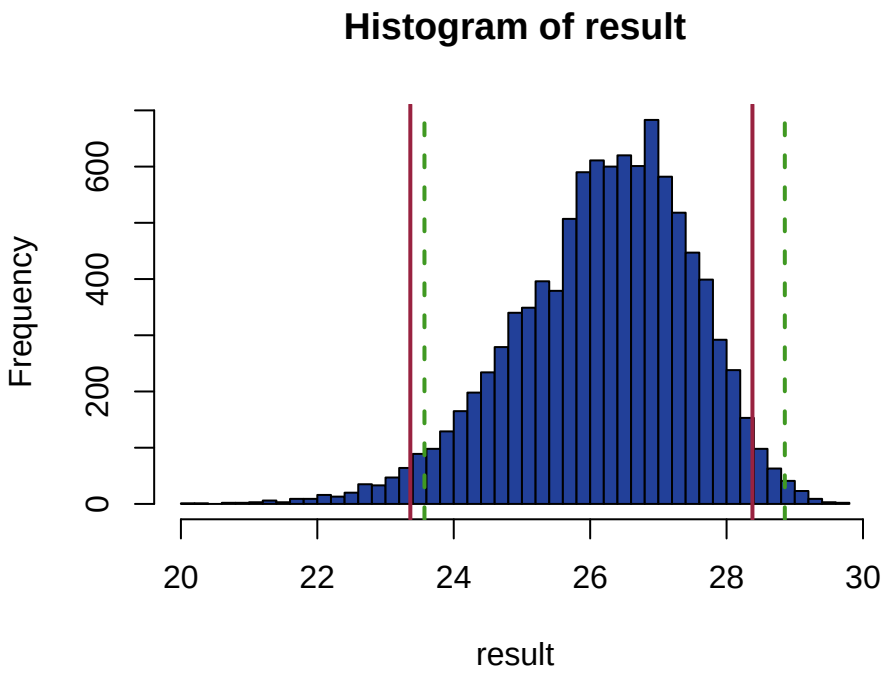
```
critical_values = qt(c(0.025,0.975),
                     df = n-1)
critical_values
```

```
[1] -1.997138  1.997138
```

```
CI_t = xbar + critical_values*se
CI_t
```

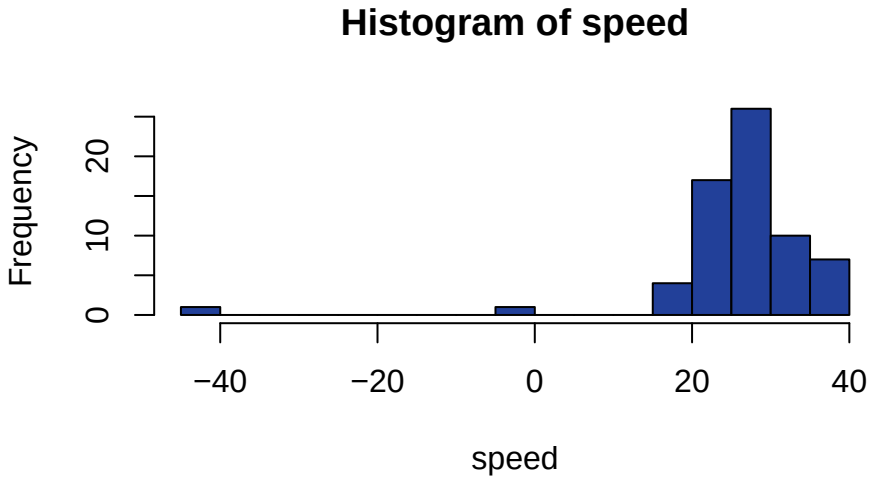
```
[1] 23.57059 28.85365
```

```
hist(result, breaks = 50,
     col = "#224099")
abline(v = CI, col = "#992240", lwd = 2)
abline(v = CI_t, col = "#409922",
       lwd = 2, lty = 2)
```



What if we trimmed the data?

```
hist(speed, col = "#224099",
     breaks = 15)
```



Keep only the positive speeds.

```
speed1 = speed[speed>0]
mean(speed)
```

```
[1] 26.21212
```

```
mean(speed1)
```

```
[1] 27.75
```

Speed of light

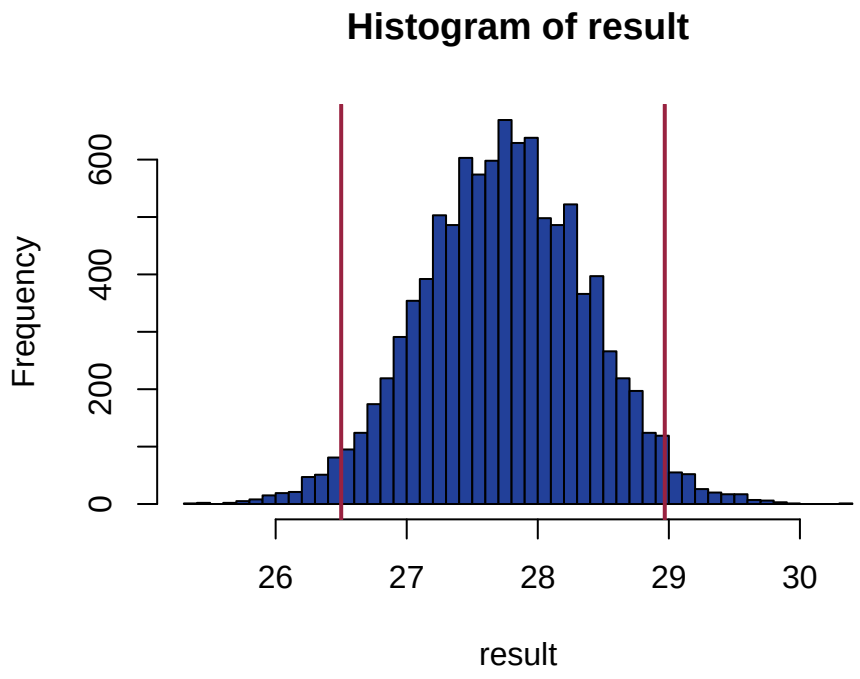
```
B = 10000
result = vector("numeric", length = B)
for(i in 1:B){
  newData = sample(speed1, replace = TRUE)
  result[i] = mean(newData)
}
CI = quantile(result, c(0.025, 0.975))
CI
```

```
    2.5%    97.5%
26.50000 28.96875
```

```
CI - mean(speed1)
```

```
    2.5%    97.5%
-1.25000  1.21875
```

```
hist(result, breaks = 50, col = "#224099")
abline(v = CI, col = "#992240", lwd = 2)
```



Much more symmetric. Compare with the confidence interval using the t distribution.

```
xbar = mean(speed1)
n = length(speed1)
se = sd(speed1)/sqrt(n)
c(xbar, n, se)
```

```
[1] 27.7500000 64.0000000  0.6354289
```

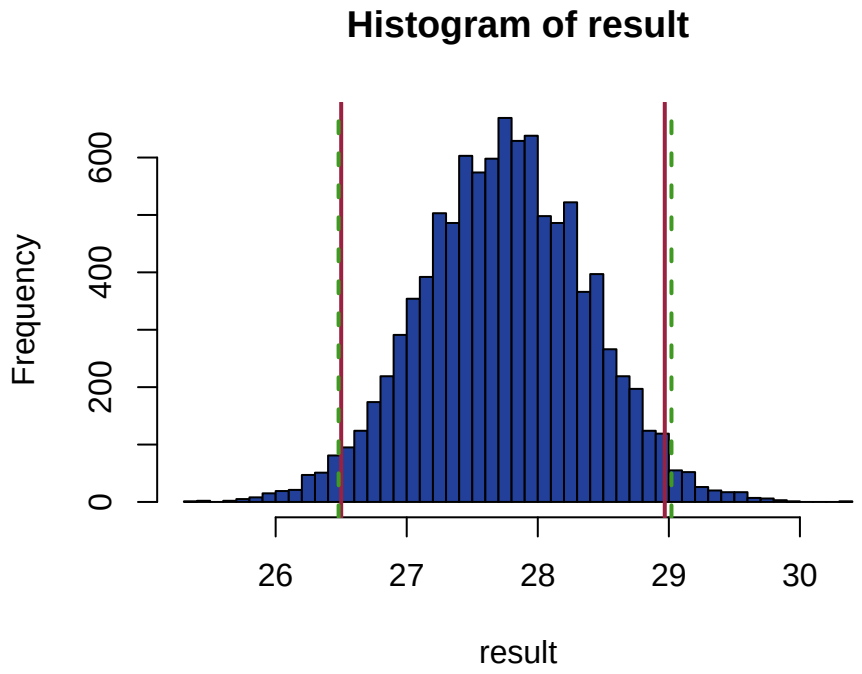
```
critical_values = qt(c(0.025,0.975), df = n-1)
critical_values
```

```
[1] -1.998341  1.998341
```

```
CI_t = xbar + critical_values*se
CI_t
```

```
[1] 26.4802 29.0198
```

```
hist(result, breaks = 50,
     col = "#224099")
abline(v = CI, col = "#992240", lwd = 2)
abline(v = CI_t, col = "#409922",
       lwd = 2, lty = 2)
```



17.4 Example: flight departure delays

Flights data set

```
library(nycflights13)
glimpse(flights)
```

```
Rows: 336,776
Columns: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2
    ~
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
    ~
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
    ~
$ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558,
    ~
```

```
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600,
    ~
$ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1
    ~
$ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,
    ~
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,
    ~
$ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1
    ~
$ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "
    ~
$ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4
    ~
$ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394
    ~
$ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",
    ~
$ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",
    ~
$ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1
    ~
$ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733,
    ~
$ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6
    ~
$ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0
    ~
$ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0
    ~
```

New York City to San Fransisco

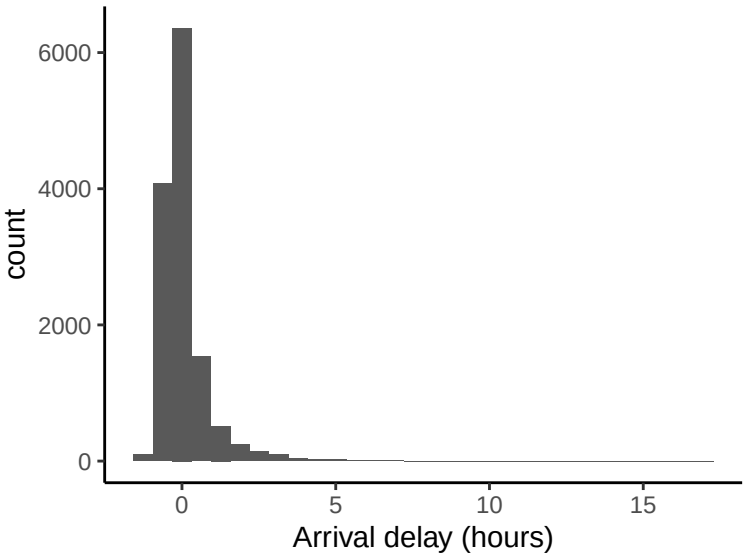
Let's restrict attention to flights between NYC and San Francisco (SFO).

```
sfo = flights |> filter(dest == "SFO")
```

This is the **population** of flights in 2013. Let's look at the distribution of arrival delays:

```
sfo |> ggplot() + aes(x = arr_delay/60) +
  geom_histogram() +
  theme_classic() +
  labs(x = "Arrival delay (hours)")
```

```
Warning: Removed 158 rows containing non-finite values (`stat_bin()`).
```



Travel policy

An organisation regularly flies staff from NYC to SFO. It decides that it is acceptable for staff to be late 2% of the time. How early should they book their flights to ensure that staff arrive on time?

```
quantile(sfo$arr_delay, p = 0.98, na.rm = TRUE)
```

```
98%
153
```

The 98th percentile of the arrival delay distribution is about 2.5 hours, so we should send them on a flight about 2.5 hours early.

> **What if we didn't have the population data?**

Sample of flights

If all we had access to was a sample of 100 flights from 2013, this is our point estimate of the 98th percentile.

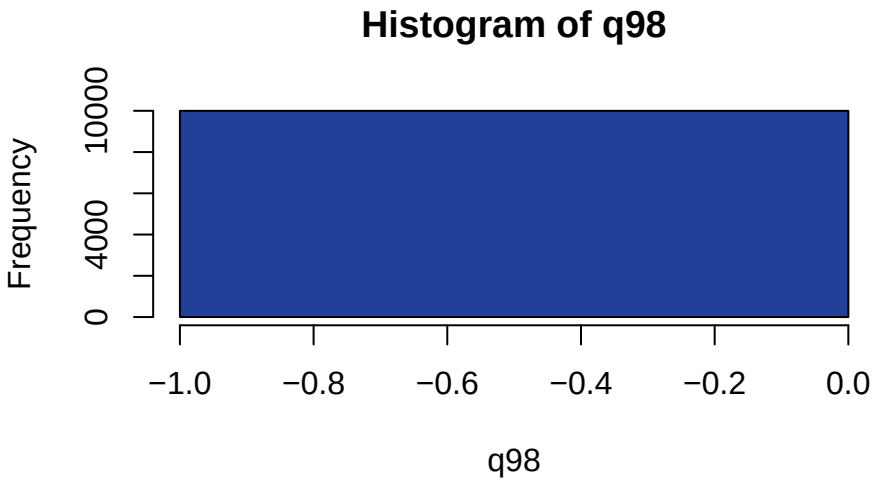
```
set.seed(88)
sfo_sample = sfo |> filter(!is.na(arr_delay)) |> sample_n(size = 100, replace
    = FALSE)
quantile(sfo_sample$arr_delay, p = 0.98)
```

```
  98%
94.72
```

> **How reliable is that point estimate?**

Bootstrap CI for quantiles

```
B = 10000
q98 = vector("numeric", length = B)
for(i in 1:B) {
  resample = sample(sfo_sample$arr_delay,
                    replace = TRUE)
  q98[i] = quantile(resample, probs = 0.98)
}
hist(q98, col = "#224099")
```



A 95% confidence interval for this quantile:

```
quantile(q98, c(0.025,0.975))
```

```
 2.5% 97.5%
 38.6 182.0
```

Based on our sample our (bootstrap) 95% confidence interval is between 1 hour and 3 hours.

Final remarks

Bootstrapping is useful when

- the theoretical distribution of a statistic is complicated or unknown (e.g. coefficient of variation, quantile regression parameter estimates, etc.)
- the sample size is too small to make any sensible parametric inferences about the parameter

Advantages

- Bootstrapping frees us from making parametric assumptions to carry out inferences
- Provides answers to problems for which analytic solutions are impossible
- Can be used to verify, or check the stability of results
- Asymptotically consistent

18
Linear Models

18.1 One Sample

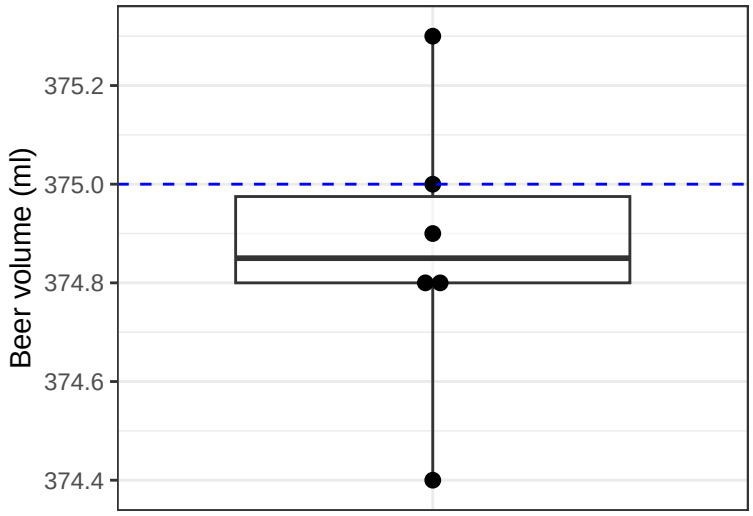
Beer contents

Beer contents in a pack of six bottles (in millilitres) are:

```
y = c(374.8, 375.0, 375.3, 374.8, 374.4, 374.9)
```

Is the mean beer content less than the 375 mL claimed on the label?

```
df = data.frame(y)
set.seed(124)
ggplot(df, aes(x = "", y = y)) +
  geom_boxplot(alpha = 0.5, coef = 10) +
  geom_dotplot(binaxis = 'y',
               stackdir = 'center') +
  geom_hline(yintercept = 375,
             colour = "blue",
             linetype = "dashed") +
  labs(y = "Beer volume (ml)", x = "") +
  theme_bw() +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())
```



Beer contents: one sample t-test

```
t.test(df$y, mu = 375)
```

```
        One Sample t-test

data:  df$y
t = -1.1094, df = 5, p-value = 0.3177
alternative hypothesis: true mean is not equal to 375
```

```
95 percent confidence interval:
 374.5577 375.1756
sample estimates:
mean of x
 374.8667
```

Linear model:

$$Y = \beta_0 + \varepsilon$$

```
beer_lm = lm(y~1, data = df)
confint(beer_lm)
```

```
              2.5 %   97.5 %
(Intercept) 374.5577 375.1756
```

Or alternatively:

$$(Y - \mu_0) = \beta_0 + \beta_1 x + \varepsilon$$

```
beer_lm0 = lm(y-375 ~ 1, data = df)
summary(beer_lm0)$coefficients |> round(4)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1333     0.1202 -1.1094   0.3177
```

Beer contents: Wilcoxon signed-rank test

```
wilcox.test(df$y, mu = 375, exact = FALSE)
```

```

        Wilcoxon signed rank test with continuity correction

data:  df$y
V = 4, p-value = 0.4164
alternative hypothesis: true location is not equal to 375
```

Define a function that converts the data to signed-ranks:

```
signed_rank = function(x) sign(x)*rank(abs(x))
```

Linear model:

$$\text{signed\_rank}(Y - \mu_0) = \beta_0 + \beta_1 x + \varepsilon$$

```
beer_lm = lm(signed_rank(y-375) ~ 1, data = df)
summary(beer_lm)$coefficients %>% round(3)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.667      1.558   -1.07    0.334
```

Not exact, but ballpark similar p-value. The approximation gets better as the sample size increases.
Let's generate a larger sample to see if the approximation works.

```
set.seed(88)
y_new = rnorm(n = 25,
              mean = mean(y),
              sd = sd(y))
wilcox.test(y_new, mu = 375)
```

```

        Wilcoxon signed rank exact test

data:  y_new
V = 90, p-value = 0.05158
alternative hypothesis: true location is not equal to 375
```

Linear model:

$$\text{signed\_rank}(Y - \mu_0) = \beta_0 + \beta_1 x + \varepsilon$$

```
beer_lm = lm(signed_rank(y_new - 375) ~ 1)
summary(beer_lm)$coefficients |> round(4)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     -5.8      2.794 -2.0758   0.0488
```

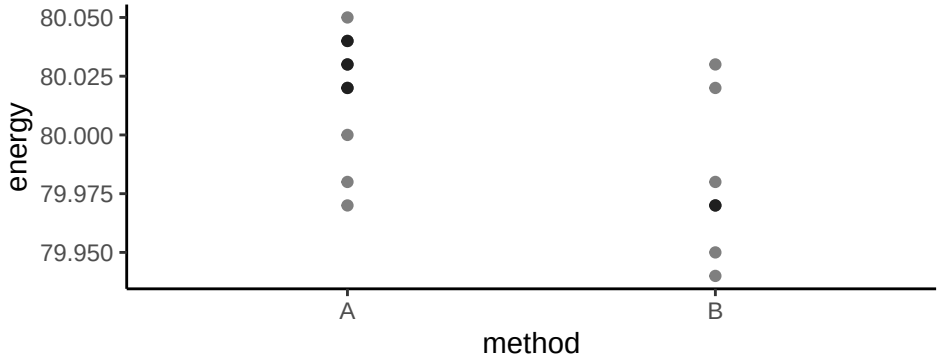
That's a closer approximate p-value.

18.2 Two Sample

Latent Heat of Fusion

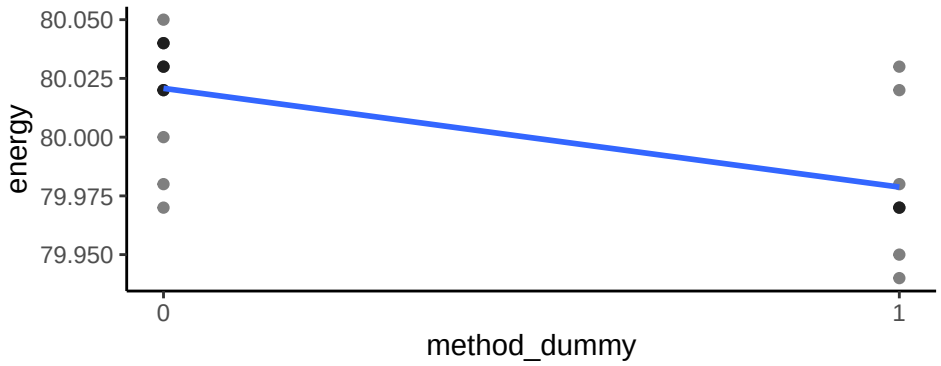
Two methods labelled A and B are used to measure the latent heat of fusion of ice. Does the data support the assumption that A gives larger results?

```
A = c(79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04,
      79.97, 80.05, 80.03, 80.02, 80.00, 80.02)
B = c(80.02, 79.94, 79.98, 79.97, 79.97, 80.03, 79.95,
      79.97)
heat = data.frame(
  energy = c(A,B),
  method = rep(c("A","B"), c(length(A), length(B)))
)
heat = heat |> dplyr::mutate(
  r = rank(energy)
)
heat |> ggplot() +
  aes(x = method, y = energy) +
  geom_point(alpha = 0.5) +
  theme_classic()
```



Recode `A` as `0` and `B` as `1`:

```
heat = heat |> mutate(
  method_dummy = if_else(
    method == "A", 0, 1))
heat |> ggplot() +
  aes(x = method_dummy,
      y = energy) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm",
              se = FALSE) +
  scale_x_continuous(breaks = c(0,1)) +
  theme_classic()
```



Latent heat of fusion: two-sample t-test

```
tt = t.test(
  energy ~ method,
  data = heat,
  var.equal = TRUE)
tt$statistic |> round(4)
```

```
     t
3.4722
```

```
tt$p.value |> round(4)
```

```
[1] 0.0026
```

```
tt$conf.int |> round(3)
```

```
[1] 0.017 0.067
attr(,"conf.level")
[1] 0.95
```

Linear model:

$$Y = \beta_0 + \beta_1 x + \varepsilon$$

```
heat_lm = lm(energy ~ method, data = heat)
summary(heat_lm)$coefficients |> round(4)
```

```
            Estimate Std. Error    t value Pr(>|t|)
(Intercept)  80.0208     0.0075 10713.4567   0.0000
methodB      -0.0420     0.0121    -3.4722   0.0026
```

```
confint(heat_lm) |> round(3)
```

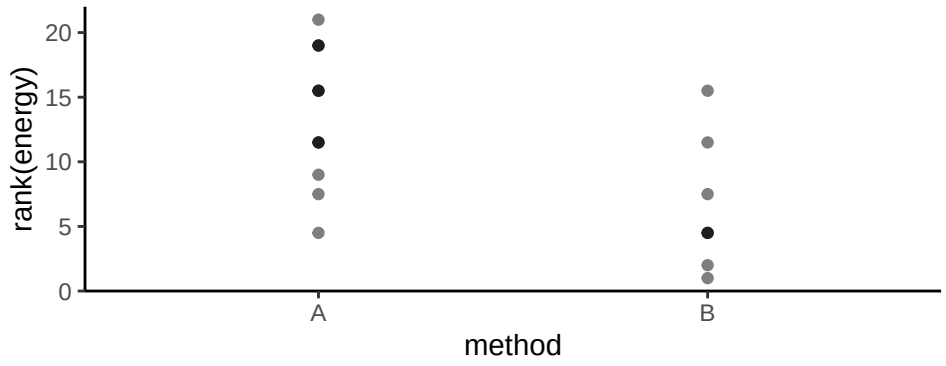
```
             2.5 % 97.5 %
(Intercept) 80.005 80.036
methodB     -0.067 -0.017
```

```
c(tt$est, unname(diff(tt$est))) |> round(4)
```

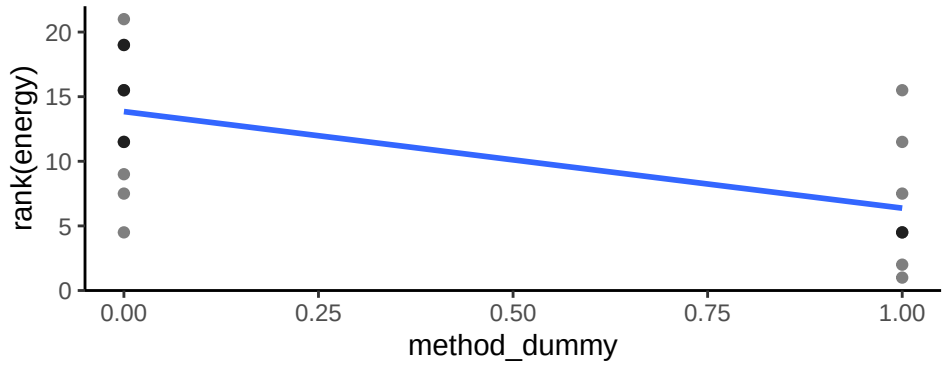
```
mean in group A mean in group B
        80.0208         79.9788         -0.0420
```

Latent heat of fusion: ranks

```
heat |> ggplot() +
  aes(x = method, y = rank(energy)) +
  geom_point(alpha = 0.5) +
  theme_classic()
```




```
heat |> ggplot() +
  aes(x = method_dummy,
      y = rank(energy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm",
              se = FALSE) +
  theme_classic()
```



Latent heat of fusion: Wilcoxon rank-sum test

```
wilcox.test(energy ~ method, data = heat)$p.value
```

```
Warning in wilcox.test.default(x = DATA[[1L]], y = DATA[[2L]], ...): cannot
    compute exact p-value with ties
```

```
[1] 0.007497146
```

Linear model:

$$\text{rank}(Y) = \beta_0 + \beta_1 x + \varepsilon$$

```
heat_lm = lm(rank(energy) ~ method, data = heat)
summary(heat_lm)$coefficients |> round(3)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   13.846      1.388   9.973    0.000
methodB       -7.471      2.249  -3.322    0.004
```

19

Multiple Testing

19.1 microRNA and Alzheimer's Disease

microRNA and Alzheimer's Disease

MicroRNA are small non-coding RNA molecules that regulate gene expression.

> **Is there any evidence that microRNA behaviour in the brain might be associated with Alzheimer's disease? (Patrick et al., 2017)**

- Experiment by measuring the amount of 309 microRNAs in 701 subjects.
- Test for significant differences between the means of subjects with and without Alzheimer's disease for each microRNA.

What does this microRNA data look like?

We start with a `RData` file that has two objects in it: `AD` and `microRNA_Data`.

```
load("data/microRNA_full.RData")
```

The `AD` object is a named vector where the names correspond to the subject (person) and the values correspond to the presence or absence of Alzheimer's disease.

```
str(AD)
```

```
 Named int [1:701] 1 0 1 1 1 1 0 0 1 1 ...
 - attr(*, "names")= chr [1:701] "20264936" "50105725" "20730959" "11229148"
     ...
```

```
head(AD)
```

```
20264936 50105725 20730959 11229148 20151388 11259716
       1        0        1        1        1        1
```

Convert this to a data frame and move the names into their own column:

```
disease_status = data.frame(AD) |>
  tibble::rownames_to_column("subject")
str(disease_status)
```

```
'data.frame':   701 obs. of  2 variables:
 $ subject: chr  "20264936" "50105725" "20730959" "11229148" ...
 $ AD     : int  1 0 1 1 1 1 0 0 1 1 ...
```

The measurements are in the `microRNA_Data` object where columns are subjects and rows are the miRNAs.

```
microRNA_Data[1:4,1:3]
```

```
           20264936 50105725 20730959
hsa-let-7a 11.97670 12.61142 13.34787
hsa-let-7b 12.24761 11.77856 11.44760
hsa-let-7c 11.07564 11.68883 11.79726
hsa-let-7d 11.30709 11.50804 11.37125
```

Reshape the microRNA data and merge in the disease status information:

```
mirna = microRNA_Data |>
  tibble::rownames_to_column("microRNA") |>
  tidyr::pivot_longer(cols = -1, names_to = "subject", values_to = "value") |>
  dplyr::left_join(disease_status)
head(mirna, n = 4)
```

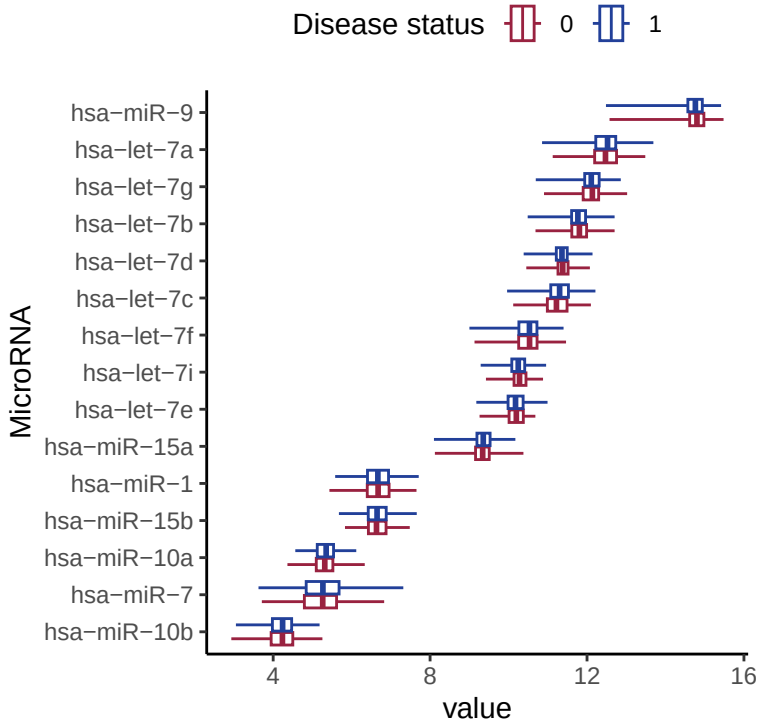
```
# A tibble: 4 x 4
  microRNA   subject  value    AD
  <chr>      <chr>    <dbl> <int>
1 hsa-let-7a 20264936  12.0     1
2 hsa-let-7a 50105725  12.6     0
3 hsa-let-7a 20730959  13.3     1
4 hsa-let-7a 11229148  12.5     1
```

How many patients have Alzheimer's?

```
library(janitor)
mirna |> select(subject, AD) |>
  distinct() |>
  janitor::tabyl(AD) |>
  janitor::adorn_pct_formatting()
```

```
 AD   n percent
  0 273   38.9%
  1 428   61.1%
```

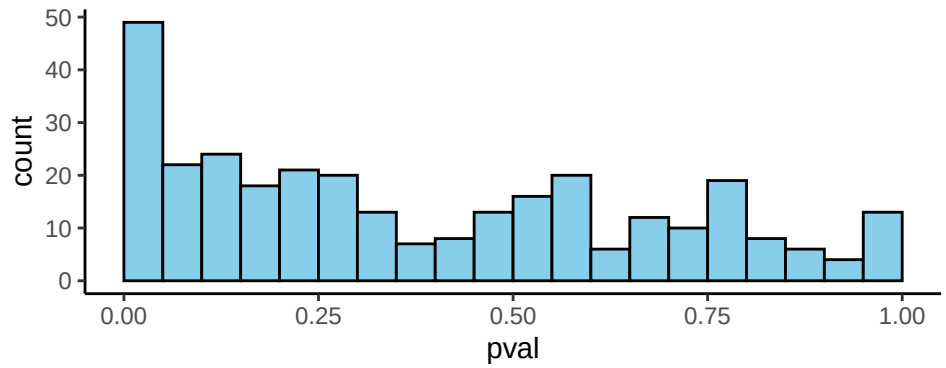
```
mirna |>
  group_by(microRNA) |>
  nest() |>
  ungroup() |>
  slice(1:15) |> # extract first 15 groups
  unnest(cols = everything()) |>
  ggplot() +
  aes(y = reorder(microRNA, value),
      x = value, colour = factor(AD)) +
  geom_boxplot(coef = 10) +
  scale_color_manual(values=c("#992240", "#224099")) +
  theme_classic() +
  theme(legend.position = "top") +
  labs(colour = "Disease status",
       y = "MicroRNA")
```



Lots of *t*-tests

Let's use Welch two-sample t-tests to compare the mean for people with Alzheimer's to people without Alzheimer's for all 309 microRNA.

```
mirna_res = mirna |>
  group_by(microRNA) |>
  summarise(pval = t.test(value~AD)$p.value)
mirna_res |> ggplot() + aes(x = pval) +
  theme_classic() +
  geom_histogram(boundary = 0, binwidth = 0.05,
                 fill = "skyblue",
                 colour = "black")
```




```
sum(mirna_res$pval < 0.05)
```

```
[1] 49
```

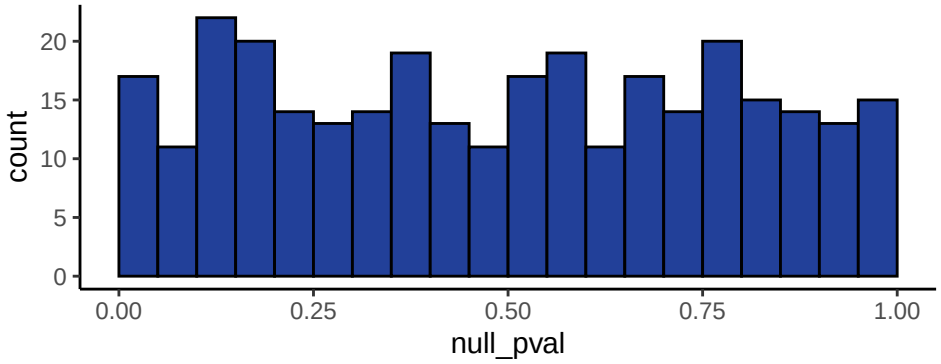
Of the 309 microRNA tested, 49 have p-values less than 0.05.

Are all of these "statistically significant" differences important?

Lots of t-tests when H_0 is TRUE

- If there was **no association** between any microRNAs and Alzheimer's disease our p-values follow a uniform distribution.
- We can generate a set of p-values *knowing* that there is no association and visualise this.

```
set.seed(88)
mirna_res = mirna_res |>
  mutate(null_pval = runif(n = n(),
                           min = 0,
                           max = 1))
mirna_res |> ggplot() + aes(x = null_pval) +
  theme_classic() +
  geom_histogram(boundary = 0,
                 binwidth = 0.05,
                 fill = "#224099",
                 colour = "black")
```




```
sum(mirna_res$null_pval < 0.05)
```

```
[1] 17
```

When we **know** that there are no truly important microRNAs, we still see 15 "significant" p-values in this simulated example.

The Reality of the Situation

- We never really know what is a real association.
- A small p-value provides some evidence against the null bit it could still be a false positive.
- Type 1 error ($\alpha = 0.05$)

For every model we evaluate at $\alpha = 0.05$, we accept that there is a 5% chance that we reject the null hypothesis when the null hypothesis is actually true.

Types of Errors

Suppose you are testing a hypothesis that a parameter θ equals zero versus the alternative that it does not equal zero.

- **Type I error or false positive** (V)
  Conclude that θ does not equal zero when it does.

- **Type II error or false negative** (T)
  Conclude that θ equals zero when it doesn't.

Possible outcomes from a series of m hypothesis tests.

Outcomes	Truth $\theta = 0$	Truth $\theta \neq 0$	Number of tests
Conclusion: $\theta = 0$	U	T	$m - R$
Conclusion: $\theta \neq 0$	V	S	R
Number of tests	m_0	$m - m_0$	m

Error Rates

False positive rate: the rate at which null results ($\theta = 0$) are called significant,

$$\mathrm{E}\left[\frac{V}{m_0}\right]$$

Family wise error rate (FWER): the probability of at least one false positive,

$$P(V \geqslant 1)$$

False discovery rate (FDR): the rate at which claims of significance are false,

$$\mathrm{E}\left[\frac{V}{R}\right]$$

Accounting for Multiple Testing

- If p-values are correctly calculated calling all p-values less than α significant will control the false positive rate at level α, on average.
- You perform 10,000 tests and the reality is that $\theta = 0$ for all of them.

> **In what sort of situation might you be doing 10,000 hypothesis tests?**

- Suppose that you call all p-values less than 0.05 significant.
- The expected number of false positives is: $10,000 \times 0.05 = 500$ false positives

> **How do we avoid so many false positives?**
> – Control the **Family-Wise Error Rate (FWER)**
> – Control the **False Discovery Rate (FDR)**

19.2 Controlling the Family-Wise Error Rate

Family-Wise Error Rate

Family-Wise Error Rate (FWER): the probability of at least one false positive.
Let $T_1, \ldots, T_m$ be m test statistics for null hypothesis $H_{01}, \ldots, H_{0m}$.
The FWER is the probability of falsely rejecting one or more H_{0i},

$$\text{FWER} = P(V \geqslant 1).$$

If the **null hypothesis is always true** but we conduct m tests each significance level α then…

- The probability of at least one false positive is $1 - (1 - \alpha)^m$
- e.g. if $m = 20$ then the FWER is 64%

```
m = 20
alpha = 0.05
1 - (1 - alpha)^m
```

```
[1] 0.6415141
```

Bonferroni Correction

The Bonferroni correction is the oldest multiple testing correction. Given that the number of false positives for m tests is $m\alpha$ then consider defining a new threshold for significance:

$$\alpha^* = \frac{\alpha}{m}$$

- This is "conservative" but keeps FWER $\leqslant \alpha$
- For $m = 20$,

$$1 - (1 - \alpha^*)^m = 1 - (1 - \frac{0.05}{20})^{20} \approx 0.0488.$$

```
m = 20
alpha = 0.05
1 - (1 - alpha/m)^m
```

```
[1] 0.04883012
```

Basic Idea

- Suppose you do m tests
- You want to control FWER at level α so $P(V \geqslant 1) \leqslant \alpha$
  - That is, the probability of making one or more type I errors is controlled at level α.
- Calculate p-values in the usual way
- Set $\alpha^* = \frac{\alpha}{m}$ (or alternatively calculate adjusted p-values: p^* = p-value $\times$ m)
- Call all p-values less than α^* significant (or all adjusted p-values less than α significant)

Pros: Easy to calculate, conservative
Cons: May be very "conservative"

10 microRNA p-values: Bonferroni Method

For the sake of illustration, we're going to control the error rates at $\alpha = 0.2$. Let's also say that we are only interested in $m = 10$ of the microRNA.

```
alpha = 0.2
m = 10
M = nrow(mirna_res)
set.seed(88, sample.kind = "Rounding")
```

```
Warning in set.seed(88, sample.kind = "Rounding"): non-uniform 'Rounding'
    sampler used
```

```
sample_rows = sample(1:M, size = m)
mirna10 = mirna_res |>
  select(microRNA, pval) |>
  slice(sample_rows) |>
  mutate(p_bonferroni = pmin(pval*m, 1)) |>
  arrange(pval)
```

We compare the original p-values to $\frac{0.2}{m} = 0.02$ or the adjusted p-values `p_bonferroni` to $\alpha = 0.02$. We get the same conclusion either way.

```
mirna10 |>
  knitr::kable(digits = 4, booktabs=TRUE) |>
    kable_styling(position="center", latex_options = "hold_position")
```

microRNA	pval	p_bonferroni
hsv1-miR-H3	0.0785	0.7851
hsv1-miR-H8	0.2251	1.0000
hsa-miR-451	0.2308	1.0000
hsa-miR-518e	0.2590	1.0000
hsa-miR-30c	0.2842	1.0000
hsa-miR-222	0.4863	1.0000
hsa-let-7a	0.5538	1.0000
hsa-miR-92b	0.7672	1.0000
hsa-miR-124	0.7698	1.0000
hsa-miR-516a-3p	0.9980	1.0000

19.3 Controlling the False Discovery Rate

False Discovery Rate

Aim: to keep the expected proportion of false positives in your rejected tests (FDR) close to α Let

- R = total number of H_{0i} rejected
- V = number of H_{0i} falsely rejected
- FRD = $\mathrm{E}\left(\frac{V}{R}\right)$

Controlling False Discovery Rate (FDR)

The Benjamini–Hochberg procedure is the most popular correction when performing *lots* of tests say in genomics, imaging, astronomy, or other signal-processing disciplines. **Basic idea**:

- Suppose you do m tests and want to **control FDR** at level α
- Calculate p-values normally
- Order the p-values from smallest to largest $p_{(1)} \leqslant p_{(2)} \leqslant \cdots \leqslant p_{(m)}$
- Find $j^* = \max j$ such that $p_{(j)} \leqslant \frac{j}{m}\alpha$
- Reject all H_{0i} where $p_{(i)} \leqslant \frac{j^*}{m}\alpha$

Pros: Still pretty easy to calculate, less conservative (maybe much less)
Cons: Allows for more false positives, may behave strangely under dependence

10 microRNA p-values: BH method

Controlling the FDR rates at $\alpha = 0.2$. Take a the same sample of 10 p-values from the microRNA experiment.

```
alpha = 0.2
m = 10
p_vals = sort(mirna10$pval)
# BH procedure
# j=1: smallest p-value < 1*alpha/m?
p_vals[1] < 1*alpha/m
```

```
[1] FALSE
```

```
# j=2:
# second smallest p-value < 2*alpha/m?
p_vals[2] < 2*alpha/m
```

```
[1] FALSE
```

```
# j=3:
# third smallest p-value < 3*alpha/m?
p_vals[3] < 3*alpha/m
```

```
[1] FALSE
```

```
# j=4:
# fourth smallest p-value < 4*alpha/m?
p_vals[4] < 4*alpha/m
```

```
[1] FALSE
```

```
# j=5:
# fifth smallest p-value < 5*alpha/m?
p_vals[5] < 5*alpha/m
```

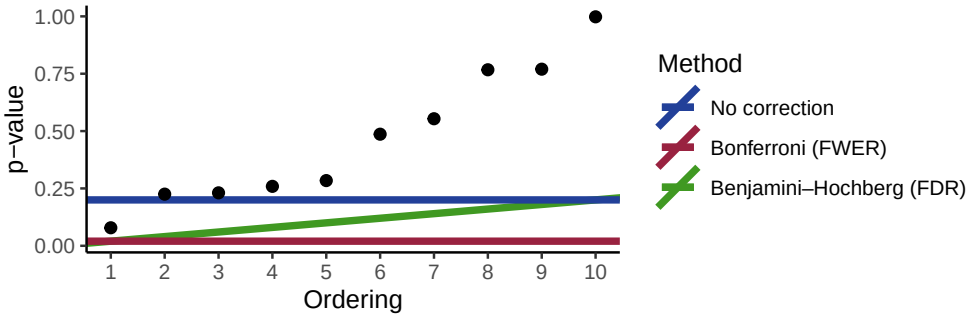
```
[1] FALSE
```

```
# and so on ...
```

10 microRNA p-values: BH vs Bonferonni

Controlling all error rates at $\alpha = 0.20$ and using our sample of 10 microRNAs.
The lines are the significance thresholds for the three methods. If a point is below the line, the method would consider it "significant".

```
alpha = 0.2; m = 10
pvaldf = data.frame(p_vals) |> mutate(rank = rank(p_vals, ties.method = "
    random"))
pvaldf |> ggplot() + aes(y = p_vals, x = rank) +
  geom_abline(aes(intercept = 0, slope=alpha*1/m, colour = '-BenjaminiHochberg
      (FDR)'), linewidth = 1.5) +
  geom_hline(aes(yintercept = alpha, colour = 'No correction'), linewidth = 1
      .5) +
  geom_hline(aes(yintercept = alpha/m, colour = 'Bonferroni (FWER)'),
      linewidth = 1.5) +
  geom_point(size = 2) +
  scale_x_continuous(breaks = 1:10) +
  scale_colour_manual(name='Method',
                      breaks=c('No correction', 'Bonferroni (FWER)', '
                          -BenjaminiHochberg (FDR)'),
                      values=c('No correction'='#224099',
                               'Bonferroni (FWER)'='#992240',
                               '-BenjaminiHochberg (FDR)'='#409922')) +
  theme_classic() +
  labs(y = "p-value", x = "Ordering")
```



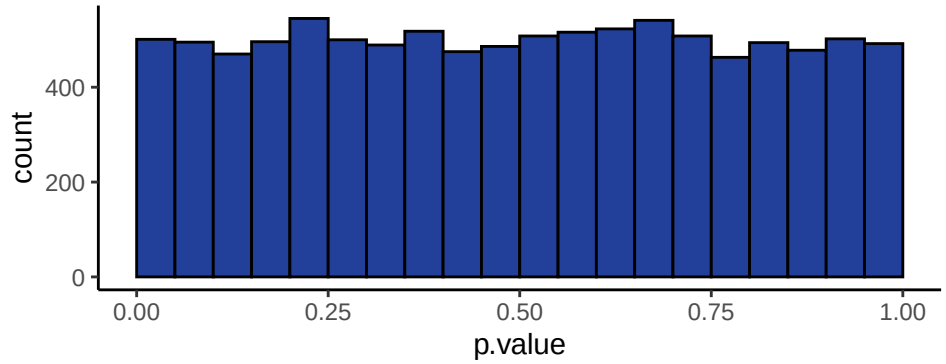
19.4 Simulation Experiments

Case Study I: No True Positives

```
set.seed(88)
p_vals = rep(NA, 1000)
B = 10000
case1 = tibble(experiment = 1:B) |>
  group_by(experiment) |>
  summarise(x_sample = rnorm(20),
            y_sample = rnorm(20)) |>
  nest() |>
  mutate(
    test = map(data,
               ~t.test(.$x_sample,
                       .$y_sample,
                       var.equal = TRUE) |>
                 broom::tidy())) |>
  unnest(test) |>
  ungroup()
mean(case1$p.value < 0.05)
```

```
[1] 0.0501
```

```
case1 |> ggplot() + aes(x = p.value) +
  theme_classic() +
  geom_histogram(boundary = 0, binwidth = 0.05,
                 fill = "#224099", colour = "black")
```



Get R to do the corrections for us using the `p.adjust()` function:

```
case1 = case1 |>
  mutate(
    p_bonf = p.adjust(p.value,
                      "bonferroni"),
    p_bh = p.adjust(p.value, "BH")
  )
case1 |> select(experiment, p.value,
                p_bonf, p_bh) |>
  head()
```

```
# A tibble: 6 x 4
  experiment p.value p_bonf  p_bh
       <int>   <dbl>  <dbl> <dbl>
1          1   0.612      1 0.989
2          2   0.385      1 0.989
3          3   0.310      1 0.989
4          4   0.126      1 0.989
5          5   0.385      1 0.989
6          6   0.710      1 0.989
```

Proportion of "significant" results

```
case1 |> ungroup() |>
  summarise(
    original = mean(p.value < 0.05),
    bonferroni = mean(p_bonf < 0.05),
    bh = mean(p_bh < 0.05)
  )
```

```
# A tibble: 1 x 3
  original bonferroni    bh
     <dbl>      <dbl> <dbl>
1   0.0501          0     0
```

Case study II: 50% true positives

```
set.seed(88)
B = 10000
case2 = tibble(experiment = 1:B) |>
  group_by(experiment) |>
  summarise(x_sample = rnorm(20), y_sample = rnorm(20)) |>
  rowwise() |>
  mutate(truth = if_else(experiment <= B/2, "mu1 - mu2 = 0", "mu1 - mu2 = 2"),
         y_sample = if_else(truth == "mu1 - mu2 = 2", y_sample + 2, y_sample))
              |>
  ungroup() |>
  nest(data = c(x_sample, y_sample)) |>
  mutate(test = map(data, ~t.test(.$x_sample, .$y_sample, var.equal = TRUE) |>
                      broom::tidy())) |>
  unnest(test) |> ungroup() |>
  mutate(
    prediction = if_else(p.value < 0.05, "reject H0", "don't reject H0"),
    p_bonf = p.adjust(p.value, method = "bonferroni"),
    p_bh = p.adjust(p.value, method = "BH"),
    pred_bonf = if_else(p_bonf < 0.05, "reject H0", "don't reject H0"),
    pred_bh = if_else(p_bh < 0.05, "reject H0", "don't reject H0"))
```

```
# no adjustment
case2 |> tabyl(prediction, truth) |>
  gt::gt()
```

prediction	mu1 - mu2 = 0	mu1 - mu2 = 2
don't reject H0	4751	0
reject H0	249	5000

```
# Bonferroni: controls FWER
case2 |> tabyl(pred_bonf, truth) |>
  gt::gt()
```

pred_bonf	mu1 - mu2 = 0	mu1 - mu2 = 2
don't reject H0	5000	900
reject H0	0	4100

```
# BH: controls FDR
case2 |> tabyl(pred_bh, truth) |>
  gt::gt()
```

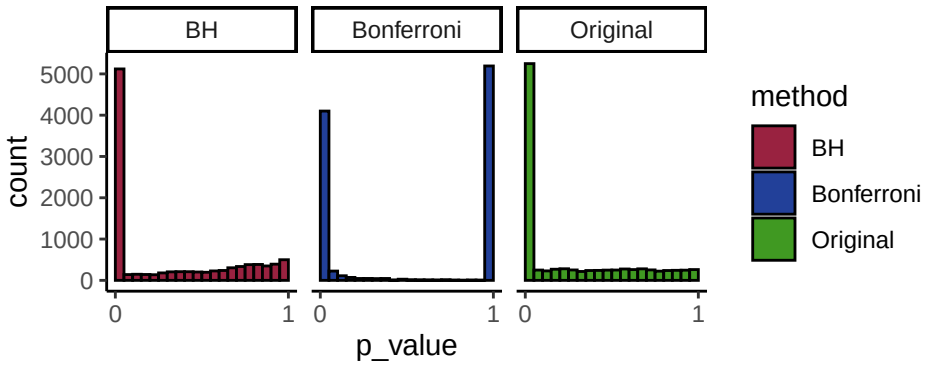
pred_bh	mu1 - mu2 = 0	mu1 - mu2 = 2
don't reject H0	4880	0
reject H0	120	5000

- Bonferroni is controlling FWER

$$\text{FWER} = P(V \geqslant 1)$$

- BH is controlling FDR = $\text{E}\left(\frac{V}{R}\right)$

```
pval2 = case2 |> select(experiment, p.value, p_bonf, p_bh) |>
  pivot_longer(cols = c(p.value, p_bonf, p_bh),
               names_to = "method",
               values_to = "p_value") |>
  mutate(method = recode(method,
                         "p.value" = "Original",
                         "p_bh" = "BH",
                         "p_bonf" = "Bonferroni"))
pval2 |> ggplot() +
  aes(x = p_value, fill = method) +
  geom_histogram(boundary = 0, binwidth = 0.05, colour = "black") +
  facet_grid(~method) +
  scale_fill_manual(values=c("#992240","#224099","#409922")) +
  scale_x_continuous(breaks = c(0,1)) +
  theme(legend.position = "none") +
  theme_classic ()
```



19.5 MicroRNA Revisited

All microRNA p-values

```
mirna_res = mirna_res |>
  mutate(
    p_bonf = p.adjust(pval, method = "bonferroni"),
    p_bh = p.adjust(pval, method = "BH")
  )
mirna_res |>
  summarise(original_n_sig = sum(pval < 0.05),
            bonf_n_sig = sum(p_bonf < 0.05),
            bh_n_sig = sum(p_bh < 0.05))
```

```
# A tibble: 1 x 3
  original_n_sig bonf_n_sig bh_n_sig
           <int>      <int>    <int>
1             49          4        7
```

```
library(gt)
mirna_res |> arrange(pval) |>
  select(-null_pval, `Original p-value` = pval,
         `Bonferroni p-value` = p_bonf,  `BH p-value` = p_bh) |>
  head(n = 10) |>
  gt() |> fmt_scientific(columns = 2:4) |>
  tab_style(
    style = list(
      cell_fill(color = "#F9E3D6"),
      cell_text(style = "italic")),
    locations = list(
      cells_body(columns = `Original p-value`,
                 rows = `Original p-value` < 0.05),
      cells_body(columns = `Bonferroni p-value`,
                 rows = `Bonferroni p-value` < 0.05),
      cells_body(columns = `BH p-value`, rows = `BH p-value` < 0.05)
    ))
```

microRNA	Original p-value	Bonferroni p-value	BH p-value
hsa-miR-132	1.28×10^{-12}	3.95×10^{-10}	3.95×10^{-10}
hsa-miR-129-5p	3.31×10^{-7}	1.02×10^{-4}	5.12×10^{-5}
hsa-miR-1260	6.33×10^{-5}	1.96×10^{-2}	6.52×10^{-3}
hsa-miR-200a	1.31×10^{-4}	4.05×10^{-2}	1.01×10^{-2}
hsa-miR-34c-5p	6.54×10^{-4}	2.02×10^{-1}	4.03×10^{-2}
hsa-miR-744	8.65×10^{-4}	2.67×10^{-1}	4.03×10^{-2}
hsa-miR-129-3p	9.13×10^{-4}	2.82×10^{-1}	4.03×10^{-2}
hsa-miR-504	1.60×10^{-3}	4.93×10^{-1}	5.56×10^{-2}
ebv-miR-BART8	1.62×10^{-3}	5.00×10^{-1}	5.56×10^{-2}
hsa-miR-874	2.20×10^{-3}	6.79×10^{-1}	5.97×10^{-2}

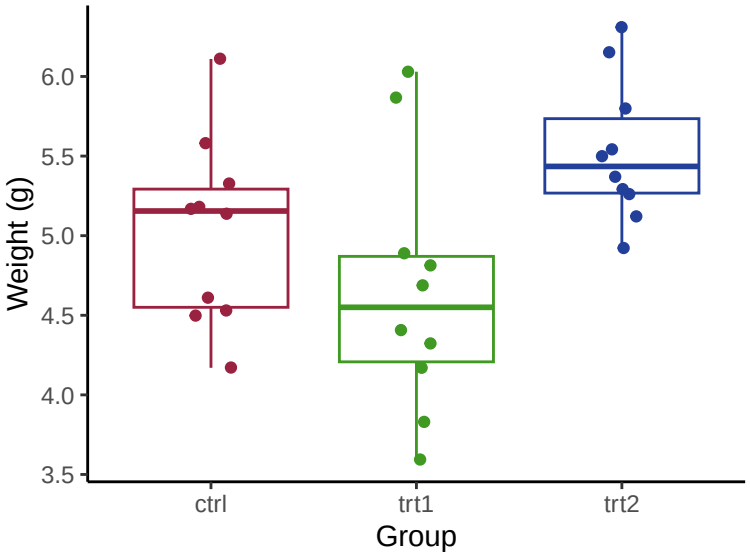
20
ANOVA

20.1 What is ANOVA?

Plant growth

The `Plantgrowth` data has results from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment conditions (Dobson, 1983, Table 7.1).

```
# built into R, make it available
data("PlantGrowth")
PlantGrowth |> ggplot() +
  aes(y = weight, x = group,
      colour = group) +
  geom_boxplot(coef = 10) +
  geom_jitter(width = 0.1) +
  theme_classic() +
  scale_colour_manual(values=c("#992240","#409922","#224099")) +
  theme(legend.position = "none") +
  labs(y = "Weight (g)", x = "Group")
```



We want to compare the means of the three different groups, the control, treatment 1 and treatment 2 groups.

What does ANOVA stand for?

- The term **ANOVA** is an abbreviation of the term **Analysis of Variance**.
- The term "variance", as well as the ANOVA procedure, is mainly due to Fisher from the 1920's, in particular the book "Statistical Methods for Research Workers" (Fisher, 1925).

Yeah, but what is "Analysis of Variance"?

- In its "simplest" form, Analysis of Variance is a generalisation of a *two-sided* two-sample t-test to 3 or more samples.
- Of the 3 different two-sample t-test, the Classical test is the one that requires the most assumptions:
- One could almost "do away" with it:
  - a Welch test could always be used instead (Welch test is the default in R);
  - the paired test can also be used if the sample sizes are equal!
    - In that case the differences are still iid normal!
    - The paired test suffers a *minor* loss of power (due to the lower degrees of freedom only) but is robust against positive correlation.
- **But** the Classical test is the one that generalises to ANOVA.
- We must always be aware of these *key assumptions*:
  - independence between samples;
  - equal variance.

20.2 The General ANOVA Decomposition

ANOVA (in the case of g groups)

Workflow: One-Way ANOVA

- **Hypotheses**: $H_0 : \mu_1 = \mu_2 = \cdots = \mu_g$ vs H_1 : at least $\mu_i \neq \mu_j$
- **Assumptions**: Observations are independent within each of the g samples. Each of the g populations have the same variance, $\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_g^2 = \sigma$ Each of the g populations are normally distributed (or the sample sizes are large enough such that you can rely on the central limit theorem).
- **Test Statistic**: $T = \frac{\text{Treatment Mean Sq}}{\text{Residual Mean Sq}}$. Under $H_0, T \sim F_{g-1,N-g}$ where g is the number of groups.
- **Observed Test Statistic**: $t_0 = \frac{\text{Treatment Mean Sq}}{\text{Residual Mean Sq}}$.
- **p-value**: $P(T \geqslant t_0) = P(F_{g-1,N-g} \geqslant t_0)$. *Note: always looking in the upper tail.*
- **Decision**: If the p-value is less then α we reject the null hypothesis and conclude that the population mean of at least one group is significantly different to the others. If the p-value is larger than α we do not reject the null hypothesis and conclude that there is no significant difference between the population means.

Double Subscript Notation

- We shall now introduce some new notational conventions to enable analysis of multi-sample problems.
- We shall denote the data with two subscripts:
  - i to index the samples ("groups")
  - j to index observations within samples/groups.
- The samples may have possibly different sizes across groups.
- We let g denote the number of groups.
- We thus let y_{ij} denote the observation on individual j in the sample for group i, $j = 1, 2, dotsc, n_i, i = 1, 2, \ldots, g$.

The Normal Model

- We model y_{ij} (for each $j = 1, 2, \dots, n_i$ and $i = 1, 2, \dots, g$) as the value taken by a random variable
$$Y_{ij} \sim \mathcal{N}(\mu_i, \sigma^2),$$
and that all random variables are independent.
- Thus we have g different iid samples, the sample for group i (of size n_i) being iid $\mathcal{N}(\mu_i, \sigma^2)$.
- In other words, for each $i = 1, 2, \dots, g$, $Y_{i1}, \dots, Y_{in_i}$ are iid $\mathcal{N}(\mu_i, \sigma^2)$ random variables.

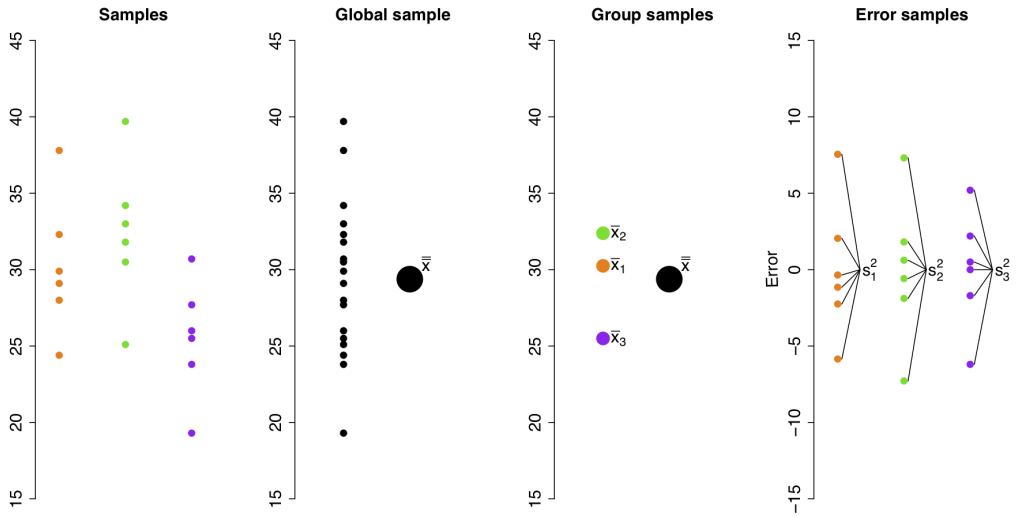
The Dreaded "Dot" Notation

- When working with double subscripts it is convenient to introduce the **dot** notation:
  - replacing either (or both) subscript(s) with a dot means **adding** over that/those subscript(s);
  - replacing either (or both) subscript(s) with a dot a**nd writing a bar over the letter** means **averaging** over that/those subscript(s).
- For example:
  - total for sample i is $\sum_{j=1}^{n_i} y_{ij} = y_{i\bullet}$
  - average for sample i is $\frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij} = \overline{y}_{i\bullet}$
  - grand total of all the observations is $\frac{1}{N}\sum_{i=1}^{g}\sum_{y=1}^{n_i} y_{ij} = \overline{y}_{\bullet\bullet}$
  - here $N = n_1 + \cdots + n_g$ is the total number of observations.

The General ANOVA Decomposition

- The "weighted average" decomposition introduced earlier for the two-sample $t-test$ is a special case of a more general decomposition.
- It is most easily explained by considering the so-called **Total Sum of Squares**:
$$\sum_{i=1}^{g}\sum_{j=1}^{n_i}(y_{ij} - \overline{y}_{\bullet\bullet})^2$$
which is $(N-1)$ times the *combined* sample variance of all observations,
$$\hat{\sigma}_0^2 = \frac{\text{Total SS}}{N-1} = \frac{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(y_{ij} - \overline{y}_{\bullet\bullet})^2}{N-1}$$
- Compare with the i-th group's sample variance: $s_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(y_{ij} - \overline{y}_{i\bullet})^2$.

$$\begin{aligned}
\sum_{i=1}^{g}\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{\bullet\bullet})^2 &= \sum_{i=1}^{g}\sum_{j=1}^{n_i}\left[(y_{ij}-\overline{y}_{i\bullet})+(\overline{y}_{i\bullet}-\overline{y}_{\bullet\bullet})\right]^2 \\
&= \sum_{i=1}^{g}\sum_{j=1}^{n_i}\left[(y_{ij}-\overline{y}_{i\bullet})^2+2(y_{ij}-\overline{y}_{i\bullet})(\overline{y}_{i\bullet}-\overline{y}_{\bullet\bullet})+(\overline{y}_{i\bullet}-\overline{y}_{\bullet\bullet})^2\right] \\
&= \sum_{i=1}^{g}\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{i\bullet})^2+2\sum_{i=1}^{g}(\overline{y}_{i\bullet}-\overline{y}_{\bullet\bullet})\underbrace{\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{i\bullet})}_{=0}+\sum_{i=1}^{g}(\overline{y}_{i\bullet}-\overline{y}_{\bullet\bullet})^2\underbrace{\sum_{j=1}^{n_i}1}_{=n_i} \\
&= \underbrace{\sum_{i=1}^{g}\underbrace{\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{i\bullet})^2}_{=(n_i-1)s_i^2}}_{\text{sample variances}}+\underbrace{\sum_{i=1}^{g}n_i(\overline{y}_{i\bullet}-\overline{y}_{\bullet\bullet})^2}_{\text{sample means}} \\
&= \text{Residual SS} + \text{Treatment SS}
\end{aligned}$$



Residual Sum of Squares; Residual Mean Square

- The first term, viewed as a random variable under the normal model, can be written as
$$\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij}-\overline{Y}_{i\bullet})^2 = \sum_{i=1}^{g}\underbrace{(n_i-1)S_i^2}_{\sim\sigma^2\chi^2_{n_i-1}} \sim \sigma^2\chi^2_{N-g}$$
noting that $\sum_{i=1}^{g}(n_i-1)=N-g$. This is called the **Residual Sum of Squares**.
- Dividing by $N-g$ we obtain an unbiased estimator of σ^2, the generalisation of the pooled estimate of the variance, known as the **Residual Mean Square**:
$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij}-\overline{Y}_{i\bullet})^2}{N-g} \sim \left(\frac{\sigma^2}{N-g}\right)\chi^2_{N-g}.$$

Treatment Sum of Squares

The full "random variable" version of the decomposition looks like

$$\underbrace{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij}-\overline{Y}_{\bullet\bullet})^2}_{\sim\sigma^2\chi^2_{N-1}\text{ under }H_0} = \underbrace{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij}-\overline{Y}_{i\bullet})^2}_{\sim\sigma^2\chi^2_{N-g}\text{ always}} + \underbrace{\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet}-\overline{Y}_{\bullet\bullet})^2}_{\sim\ ????}$$

- When H_0 is true, the blue final term must have a $\sigma^2\chi^2_{g-1}$ distribution;
- If the true group means are not all equal, the blue final term will tend to get bigger.
- The blue final term is the **Treatment Sum of Squares**.
- The ratio $\frac{\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet}-\overline{Y}_{\bullet\bullet})^2}{g-1}$ is the **Treatment Mean Square**

Treatment? Huh?

- The term "Treatment" dates back to the beginnings of Analysis of Variance, where R.A. Fisher applied these techniques to agricultural trials, notably concerning fertiliser treatments.
- The **Treatment Sum of Squares** is the generalisation of the term $\left(\frac{\overline{X}-\overline{Y}}{\sqrt{\frac{1}{m}+\frac{1}{n}}}\right)^2$ in the analysis of the two-combined-sample variance.
- It measures the variability of the sample means in a certain sense.

$$\text{Treatment Mean Square} = \frac{\text{Treatment SS}}{g-1} = \frac{\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet}-\overline{Y}_{\bullet\bullet})^2}{g-1}$$

The "Ratio of Variances" Test

- Continuing the analogy to the two-sample t-test, we can consider the ratio of variance estimates as a test statistic to test the null hypothesis

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_g$$

  against the alternative that they are not all equal.
- The estimate under the null hypothesis is just the "combined" sample variance

$$\hat{\sigma}_0^2 = \frac{1}{N-1}\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij}-\overline{Y}_{\bullet\bullet})^2.$$

- The estimate under the alternative or "full model" is the **Residual Mean Square**

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij}-\overline{Y}_{i\bullet})^2}{N-g}.$$

The F Statistic

- A sensible test statistic considers the ratio of these two ways of estimating σ^2:

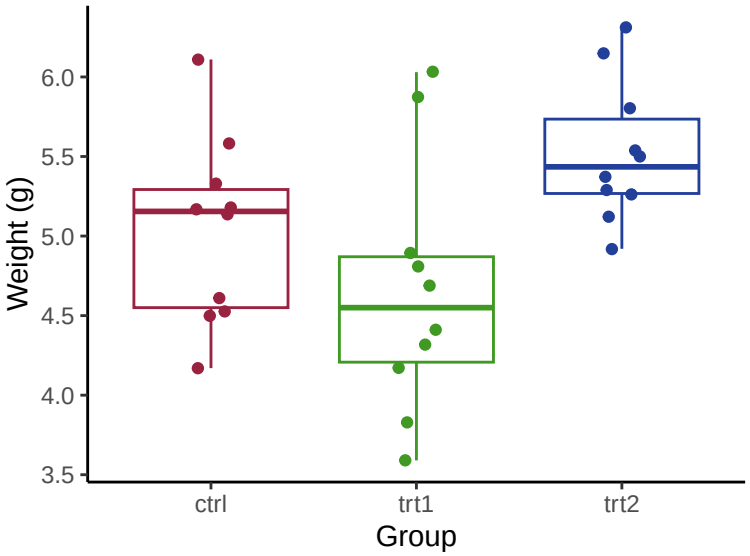
$$\begin{aligned}\frac{\text{Treatment Mean Square}}{\text{Residual Mean Square}} &= \frac{\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet} - \overline{Y}_{\bullet\bullet})^2/(g-1)}{\hat{\sigma}^2} \\ &= \frac{\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet} - \overline{Y}_{\bullet\bullet})^2/(g-1)}{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_{i\bullet})^2/(N-g)} \\ &\sim \frac{\chi^2_{g-1}/(g-1)}{\chi^2_{N-g}/(N-g)} \text{ (both independent)} \\ &\sim F_{g-1,N-g} \text{ under } H_0.\end{aligned}$$

- the denominator is always an unbiased estimator of σ^2 regardless of whether H_0 is true or not
- the numerator is only an unbiased estimator of σ^2 if H_0 is true, otherwise **it tends to get bigger**

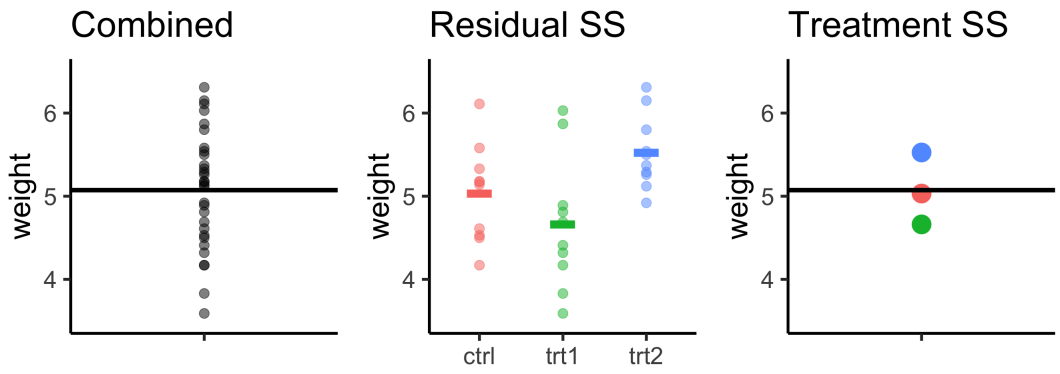
Plant growth

The `Plantgrowth` data has results from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment conditions (Dobson, 1983, Table 7.1).

```
PlantGrowth |> ggplot() +
  aes(y = weight, x = group,
      colour = group) +
  geom_boxplot(coef = 10) +
  geom_jitter(width = 0.1) +
  theme_classic() +
  scale_colour_manual(values=c("#992240","#409922","#224099")) +
  theme(legend.position = "none") +
  labs(y = "Weight (g)", x = "Group")
```



We want to compare the means of the three different groups, the control, treatment 1 and treatment 2 groups.



Combined sample variance (estimate under the null hypothesis): $\hat{\sigma}_0^2 = \frac{1}{N-1}\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_{\bullet\bullet})^2$
Residual mean square (estimate under the alternative hypothesis): $\hat{\sigma}^2 = \frac{1}{N-g}\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_{i\bullet})^2$
Treatment mean square: $\frac{1}{g-1}\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet} - \overline{Y}_{\bullet\bullet})^2$

Decomposition

```
PlantGrowth = PlantGrowth |>
  mutate(overall_mean = mean(weight)) |>
  group_by(group) |>
  mutate(group_mean = mean(weight))
PlantGrowth |> slice(1:2)
```

```
# A tibble: 6 x 4
# Groups:   group [3]
  weight group overall_mean group_mean
   <dbl> <fct>        <dbl>      <dbl>
1   4.17 ctrl          5.07       5.03
2   5.58 ctrl          5.07       5.03
3   4.81 trt1          5.07       4.66
4   4.17 trt1          5.07       4.66
5   6.31 trt2          5.07       5.53
6   5.12 trt2          5.07       5.53
```

```
N = nrow(PlantGrowth)
g = 3
```

Treatment mean square: $\frac{1}{g-1}\sum_{i=1}^{g} n_i(\overline{Y}_{i\bullet} - \overline{Y}_{\bullet\bullet})^2$
group means vs overall mean

```
treat_ss = sum((PlantGrowth$group_mean - PlantGrowth$overall_mean)^2)
treat_ms = treat_ss/(g-1)
c(treat_ss, treat_ms)
```

```
[1] 3.76634 1.88317
```

Residual mean square: $\hat{\sigma}^2 = \frac{1}{N-g}\sum_{i=1}^{g}\sum_{j=1}^{n_i}(Y_{ij} - \overline{Y}_{i\bullet})^2$
observations vs their group means

```
resid_ss = sum((PlantGrowth$weight - PlantGrowth$group_mean)^2)
resid_ms = resid_ss/(N-g)
c(resid_ss, resid_ms)
```

```
[1] 10.4920900  0.3885959
```

```
plant_anova = aov(weight ~ group, data = PlantGrowth)
plant_anova
```

```
Call:
   aov(formula = weight ~ group, data = PlantGrowth)

Terms:
                    group Residuals
Sum of Squares    3.76634  10.49209
Deg. of Freedom         2        27

Residual standard error: 0.6233746
Estimated effects may be unbalanced
```

```
summary(plant_anova)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
group        2  3.766  1.8832   4.846 0.0159 *
Residuals   27 10.492  0.3886
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Hypotheses**: $H_0 : \mu_1 = \mu_2 = \mu_3$ vs H_1 : at least $\mu_i \neq \mu_j$ for $i \neq j$
- **Assumptions**: Observations are independent within each of the 3 samples. Each of the 3 populations are normally distributed with the common variance σ.
- **Test Statistic**: $T = \frac{\text{Treatment Mean Sq}}{\text{Residual Mean Sq}}$. Under $H_0, T \sim F_{g-1,N-g}$ where $g = 3$ is the number of groups.
- **Observed Test Statistic**: $t_0 = \frac{1.8832}{0.3886} = 4.846$.
- **p-value**: $P(T \geqslant 4.8) = P(F_{2,\,27} \geqslant 4.846) = 0.0159$. Manually in R: `pf(4.846, df1 = 2, df2 = 27, lower.tail = FALSE)`
- **Decision**: As the p-value is less than α we reject the null hypothesis and conclude that the population mean of at least one group is significantly different to the others.

Which are different? Ctrl vs Trt1? Ctrl vs Trt2? Trt1 vs Trt2?

21
ANOVA Contrasts

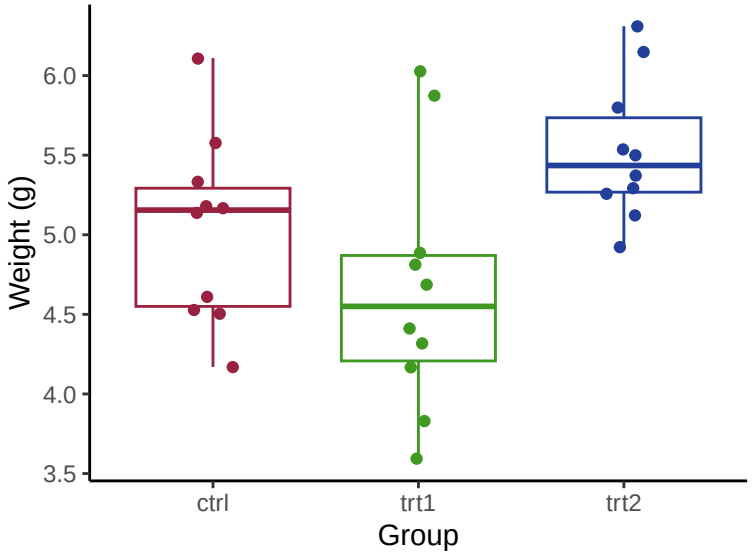
21.1 Contrasts

Plant growth

The `Plantgrowth` data has results from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment conditions (Dobson, 1983, Table 7.1).

```
PlantGrowth |> ggplot() +
  aes(y = weight, x = group,
      colour = group) +
  geom_boxplot(coef = 10) +
```

```
  geom_jitter(width = 0.1) +
  theme_classic() +
  scale_colour_manual(values=c("#992240","#409922","#224099")) +
  theme(legend.position = "none") +
  labs(y = "Weight (g)", x = "Group")
```



We want to compare the means of the three different groups, the control, treatment 1 and treatment 2 groups. The null hypothesis is $H_0 : \mu_1 = \mu_2 = \mu_3$

```
plant_anova = aov(weight ~ group, data = PlantGrowth)
summary(plant_anova)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
group        2  3.766  1.8832   4.846 0.0159 *
Residuals   27 10.492  0.3886
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value, $P(F_2, 27 \geqslant 4.846 = 0.0159)$ is less than 0.05, so we reject the null hypothesis at the 5% level of significance and conclude there is evidence to suggest that at least one of the groups has a significantly different mean yield to the others.

Which means are different?

- Is it `ctrl` vs `trt1`?
- Is it `ctrl` vs `trt2`?
- Is it `trt1` vs `trt2`?
- Or is it some other linear combination of the means that is different?

Beyond ANOVA: Contrasts

- Recall our model: for $i = 1, 2, \ldots, g$ and $j = 1, 2, \ldots, n_i$ with j-th observation in the $i - th$ sample is modelled as the value taken by

$$Y_{ij} \sim \mathcal{N}(\mu_i, \sigma^2)$$

  and all random variables are assumed independent.
- We can rewrite this as:

$$Y_{ij} = \mu_i + \varepsilon_{ij},$$

where $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ is the error term.
- The ANOVA F-test is a test of the hypothesis $H_0 : \mu_1 = \mu_2 = \cdots = \mu_g$.
- If this hypothesis is "rejected", then what?
- Further analysis reduces to the study of **contrasts**

Constrasts

- A **contrast** is a **linear combination** where the coefficients **add to zero**.
- In an ANOVA context, a contrast is a linear combination of **means**.
- We make the distinction between two kinds of contrast:
  - **population contrasts**: contrasts involving the population group means i.e. the μ_i's;
  - **sample contrasts**: contrasts involving the sample group means i.e. the $\overline{y}_{i,\bullet}$'s and $\overline{Y}_{i,\bullet}$'s

Distribution of sample contrasts

- Any $c_1, \cdots, c_g$ with $c_\bullet = \sum_{i=1}^g c_i = 0$ defines a *sample contrast*, $\sum_{i=1}^g c_i \overline{Y}_{i\bullet}$
- Under our *normal-with-equal-variances* model, this random variable has distribution given by
$$\sum_{i=1}^g c_i \overline{Y}_{i\bullet} \sim \mathcal{N}\left(\sum_{i=1}^g c_i \mu_i,\ \sigma^2 \sum_{i=1}^g \frac{c_i^2}{n_i}\right).$$
- The corresponding *population contrast* is the expected value of the (random) sample contrast.
- Conversely, the (observed) *sample contrast* $\sum_{i=1}^g c_i \overline{y}_{i\bullet}$ is an *estimate* of the corresponding *population contrast*;
  - the (random) sample contrast $\sum_{i=1}^g c_i \overline{Y}_{i\bullet}$ is the corresponding *estimator*.

Behaviour of contrasts under the null hypothesis

- Under the "ANOVA null hypothesis" $H_0 : \mu_1 = \cdots = \mu_g (= \mu, \text{say})$
  - all population contrasts are zero:
$$\sum_{i=1}^g c_i \mu_i = \sum_{i=1}^g c_i \mu = \mu \sum_{i=1}^g c_i = 0;$$
  - all (random) sample contrasts have expectation zero:
$$E\left(\sum_{i=1}^g c_i \overline{Y}_{i\bullet}\right) = \sum_{i=1}^g c_i \mu_i = 0.$$
  - Therefore the "ANOVA null hypothesis" can be rephrased as "all population contrasts are zero".

Maybe not what we want

- Thus in some examples, in a particular sense, the "ANOVA null hypothesis" may be "too strong":
  - we may only wish to test one (or more) "special" population contrasts are zero.
- Also, the "ANOVA null hypothesis" may not be rejected for the reason we want:
  - some contrasts may be non-zero, but are they the ones we are interested in?

t-tests for individual contrasts

- Suppose we really only want to test that $H_0 : \sum_{i=1}^{g} c_i\mu_i = 0$ for some "special contrast" given by $c_1, \ldots, c_g$ (with $\sum_{i=1}^{g} c_i = 0$)
- We can of course perform the ANOVA Mean-Square Ratio F-test, but can we possibly do better?
- We can perform a more "targeted" F-test using the corresponding sample contrast and the **residual mean square**.
- The corresponding (random) sample contrast

$$\sum_{i=1}^{g} c_i\overline{Y}_{i\bullet} \sim \mathcal{N}\left(\sum_{i=1}^{g} c_i\mu_i, \sigma^2 \sum_{i=1}^{g} \frac{c_i^2}{n_i}\right).$$

- The *standardised version*

$$\frac{\sum_{i=1}^{g} c_i\overline{Y}_{i\bullet} - \sum_{i=1}^{g} c_i\mu_i}{\sigma\sqrt{\sum_{i=1}^{g} \frac{c_i^2}{n_i}}} \sim \mathcal{N}(0,1).$$

- replacing σ in the denominator with $\hat{\sigma} = \sqrt{\text{Residual MS}} \sim \sqrt{\chi^2_{N-g}/(N-g)}$ (independent of the $\overline{Y}_{i\bullet}$'s) gives

$$\frac{\sum_{i=1}^{g} c_i\overline{Y}_{i\bullet} - \sum_{i=1}^{g} c_i\mu_i}{\hat{\sigma}\sqrt{\sum_{i=1}^{g} \frac{c_i^2}{n_i}}} \sim t_{N-g}.$$

- Thus a t-statistic for testing the hypothesis that $\sum_{i=1}^{g} c_i\mu_i = 0$ is

$$\frac{\sum_{i=1}^{g} c_i\overline{Y}_{i\bullet}}{\hat{\sigma}\sqrt{\sum_{i=1}^{g} \frac{c_i^2}{n_i}}}$$

- This *generalises* the two-sample t-statistic.

Yield

Let μ_1, μ_2 and μ_3 represent the population means of treatment 1, treatment 2 and the control group, respectively.
Let's consider if there is a difference between treatment 1, `trt1` and treatment 2, `trt2`, this corresponds to contrast coefficients $c_1 = 1, c_2 = -1$ and $c_3 = 0$

```
plant_summary = PlantGrowth |>
  mutate(group = factor(group, levels = c("trt1","trt2", "ctrl"))) |>
  group_by(group) |>
  summarise(n = n(), mean_weight = mean(weight)) |>
  mutate(contrast_coefficients = c(1, -1, 0),
         c_ybar = mean_weight * contrast_coefficients)
plant_summary
```

```
# A tibble: 3 x 5
  group     n mean_weight contrast_coefficients c_ybar
  <fct> <int>       <dbl>                 <dbl>  <dbl>
1 trt1     10        4.66                     1   4.66
2 trt2     10        5.53                    -1  -5.53
3 ctrl     10        5.03                     0   0
```

```
sum(plant_summary$c_ybar) # sample contrast
```

```
[1] -0.865
```

Residual standard error

Recall the ANOVA analysis from earlier:

```
plant_anova = aov(weight ~ group, data = PlantGrowth)
summary(plant_anova)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
group        2  3.766  1.8832   4.846 0.0159 *
Residuals   27 10.492  0.3886
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The "Residual Mean Sq" is the estimate of σ^2 the (population) variance (within each group)
- The "Residual standard error" is $\sqrt{\text{Residual Mean Sq}}$, the esitmate of σ, the (population) standard deviation (within each group)

We use the `tidy()` function from the **broom** package to help extract these terms from the anova object.

```
broom::tidy(plant_anova)
```

```
# A tibble: 2 x 6
  term         df sumsq meansq statistic p.value
  <chr>     <dbl> <dbl>  <dbl>     <dbl>   <dbl>
1 group         2  3.77  1.88       4.85  0.0159
2 Residuals    27 10.5   0.389     NA    NA
```

```
resid_ms = broom::tidy(plant_anova)$meansq[2]
resid_ms
```

```
[1] 0.3885959
```

```
resid_se = sqrt(resid_ms)
resid_se
```

```
[1] 0.6233746
```

Calculating the test statistic

- The observed test statistic for the contrast is,

$$t_0 = \frac{\sum_{i=1}^{g} c_i \overline{y}_{i\bullet}}{\hat{\sigma}\sqrt{\sum_{i=1}^{g} \frac{c_i^2}{n_i}}} = \frac{\overline{y}_1 - \overline{y}_2}{\sqrt{\text{Residual Mean Sq}} \times \sqrt{\frac{1^2}{10} + \frac{(-1)^2}{10} + \frac{0}{10}}}$$

- From our earlier summary,

```
(n_i = plant_summary |> pull(n))
```

```
[1] 10 10 10
```

```
(ybar_i = plant_summary |> pull(mean_weight))
```

```
[1] 4.661 5.526 5.032
```

```
(c_i =  plant_summary |> pull(contrast_coefficients))
```

```
[1]  1 -1  0
```

```
(se = sqrt(resid_ms * sum((c_i^2) / n_i)))
```

```
[1] 0.2787816
```

```
(t_stat = sum(c_i * ybar_i)/se)
```

```
[1] -3.102787
```

Calculating the p-value

```
# observed test statistic
(t_stat = sum(c_i * ybar_i)/se)
```

```
[1] -3.102787
```

- The test statistic has a t_{N-g} distribution if $\sum_{i=1}^{g} c_i \mu_i = 0$.
- This is the same degrees of freedom as the denominator (residual) degrees of freedom from the ANOVA!

```
plant_anova$df.res
```

```
[1] 27
```

- A (two-sided) p-value is obtained using

```
2*pt(abs(t_stat), df = plant_anova$df.res, lower.tail = FALSE)
```

```
[1] 0.004459236
```

Why is this better than an ordinary two-sample t-test
(Potentially) a smaller standard error! Better estimate of σ.

21.2 Confidence intervals

Confidence interval

- A confidence interval for a population contrast can be obtained in the usual way, based on the t-statistic
- Suppose the "multiplier", or critical value, $t^\star$ satisfies $P(-t^\star \leqslant t_{N-g} \leqslant t^\star) = 0.95$
- Then whatever be the "true" values of the μ_i's since the quantity

$$\frac{\sum_{i=1}^{g} c_i \overline{Y}_{i\bullet} - \sum_{i=1}^{g} c_i \mu_i}{\hat{\sigma}\sqrt{\sum_{i=1}^{g} \frac{c_i^2}{n_i}}} \sim t_{N-g}$$

we have, using the usual confidence interval-type manipulations,

$$P\left(\sum_i c_i \overline{Y}_{i\bullet} - t^\star \hat{\sigma}\sqrt{\sum_i \frac{c_i^2}{n_i}} \leqslant \sum_i c_i \mu_i \leqslant \sum_i c_i \overline{Y}_{i\bullet} + t^\star \hat{\sigma}\sqrt{\sum_i \frac{c_i^2}{n_i}}\right) = 0.95.$$

"Observed value" of confidence interval

For observed sample means $\overline{y}_{1\bullet}, \ldots, \overline{y}_{g\bullet}$, a 95% confidence interval for the "true" population contrast $\sum_i c_i \mu_i$ is given by

$$\underbrace{\sum_i c_i \overline{y}_{i\bullet}}_{\text{estimate}} \pm t^\star \underbrace{\hat{\sigma}\sqrt{\sum_i \frac{c_i^2}{n_i}}}_{\text{std error}}$$

where, as above, $\hat{\sigma}$ denotes the square root of the **residual mean square**

- A two-sided 95% confidence interval for the "special contrast" considered above is given as follows:
  - the "multiplier" $t\star$ is determined via:

```
t_star = qt(0.975, df = 27)
t_star
```

```
[1] 2.051831
```

  - the interval is obtained using

```
sum(c_i * ybar_i) + c(-1,1) * t_star * se
```

```
[1] -1.4370126 -0.2929874
```

22

ANOVA Post Hoc Tests

22.1 Using residuals to Check for normality

Critical flicker frequency

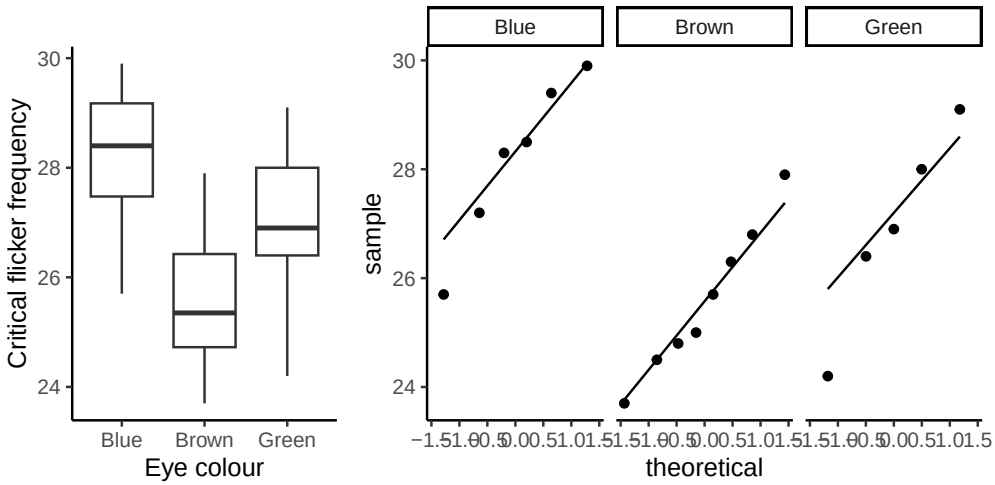
- If a light is flickering but at a very high frequency, it appears to not be flickering at all. Thus there exists a "critical flicker frequency" where the flickering changes from "detectable" to "not detectable" and this varies from person to person.
- The critical flicker frequency and iris colour for 19 people were obtained as part of a study into the relationship between critical frequency flicker and eye colour. They are given in the file `flicker.txt`.

```
path = "https://raw.githubusercontent.com/DATA2002/data/master/flicker.txt"
flicker = read_tsv(path)
glimpse(flicker)
```

```
Rows: 19
Columns: 2
$ Colour  <chr> "Brown", "Brown", "Brown", "Brown", "Brown", "Brown", "Brown",
    ~
$ Flicker <dbl> 26.8, 27.9, 23.7, 25.0, 26.3, 24.8, 25.7, 24.5, 26.4, 24.2, 28
    ~
```

Checking assumptions

```
p1 = ggplot(flicker) + aes(x = Colour, y = Flicker) +
  geom_boxplot() +
  labs(y = "Critical flicker frequency", x = "Eye colour") +
  theme_classic()
p2 = ggplot(flicker) + aes(sample = Flicker) +
  geom_qq_line() + geom_qq() + facet_wrap(~ Colour) +
  theme(axis.text.x = element_blank()) +
  theme_classic()
cowplot::plot_grid(p1, p2, ncol = 2, rel_widths = c(1.1,2),axis = "lrtb",
    align = "hv")
```



Checking for normality with residuals

The population model is,

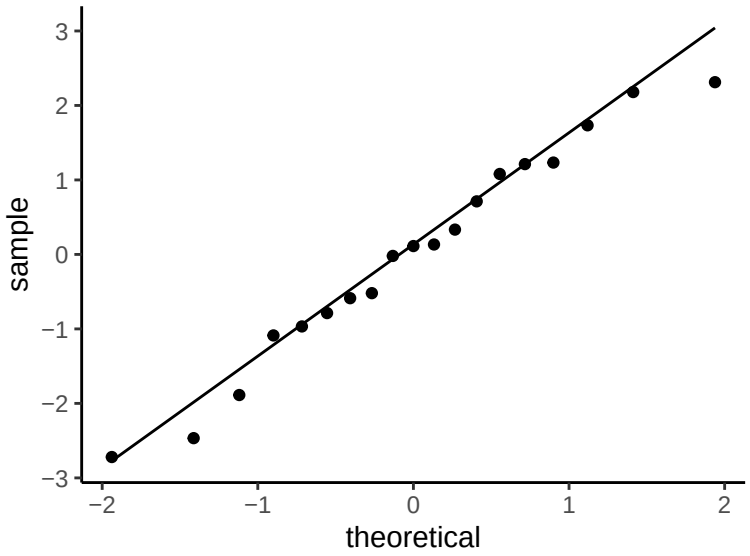
$$Y_{ij} = \mu_i + \varepsilon_{ij},$$

where $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$.
Rather than looking at QQ plots for each sample, we can instead consider the ANOVA residuals,

$$r_{ij} = y_{ij} - \overline{y}_{i\bullet}$$

> If the ANOVA assumptions hold true, then the residuals should be normally distributed.

```
flicker_anova = aov(Flicker ~ Colour, data = flicker)
flicker_resid = flicker_anova$residuals
ggplot(data.frame(flicker_resid)) +
  aes(sample = flicker_resid) +
  geom_qq_line() + geom_qq() +
  theme_classic()
```



Summary statistics

Let's generate some summary statistics which will be useful later.

```
sum_stat = flicker |> group_by(Colour) |>
  summarise(n_i = n(),
            ybar_i = mean(Flicker),
            v_i = var(Flicker))
sum_stat
```

```
# A tibble: 3 x 4
  Colour   n_i ybar_i   v_i
  <chr>  <int>  <dbl> <dbl>
1 Blue       6   28.2  2.33
2 Brown      8   25.6  1.86
3 Green      5   26.9  3.40
```

```
n_i = sum_stat |> pull(n_i)
n_i
```

```
[1] 6 8 5
```

```
ybar_i = sum_stat |> pull(ybar_i)
ybar_i
```

```
[1] 28.16667 25.58750 26.92000
```

```
v_i = sum_stat |> pull(v_i)
v_i
```

```
[1] 2.334667 1.864107 3.397000
```

ANOVA results

```
summary(flicker_anova)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
Colour       2  23.00  11.499   4.802 0.0232 *
Residuals   16  38.31   2.394
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The small p-value suggests that we should reject the ANOVA null hypothesis, $H_0 : \mu_1 = \mu_2 = \mu_3$.
- We can consider a "contrast" to check which mean is different to the others.
- In general, there may be more than one "contrast of interest".

22.2 Multiple Comparisons: Simultaneous Confidence Intervals

All pairwise differences

- When no single group is "special" or notable, so that each pairwise difference is equally interesting, we can consider each pairwise difference as a contrast of interest.
- In this case,
  - a t-statistic can be constructed for each pairwise difference;
  - a t-based confidence interval can be constructed for each pairwise "population" difference (contrast).
  - Let's focus on confidence intervals for the moment.

Individual 95% confidence intervals

- We now construct 95% confidence intervals for each pairwise comparison individually:
- the standard error for $y_{i\bullet} - \overline{y}_{h\bullet}$ is $\hat{\sigma}\sqrt{\frac{1}{n_i} + \frac{1}{n_h}}$

```
N = length(flicker_resid)
g = 3
sig_sq_hat = sum(flicker_resid^2)/(N-g) # Mean square resiudal
sig_sq_hat
```

```
[1] 2.39438
```

```
# alternatively
# sig_sq_hat = sum((n_i - 1) * v_i)/sum(n_i - 1)
t_star = qt(.975, df = sum(n_i - 1))
t_star
```

```
[1] 2.119905
```

Blue vs Brown

```
se.Bl.Br = sqrt(sig_sq_hat * ((1/n_i[1]) + (1/n_i[2])))
(int.Bl.Br.95.indiv = ybar_i[1] - ybar_i[2] + c(-1,1) * t_star * se.Bl.Br)
```

```
[1] 0.8076044 4.3507289
```

Blue vs Green

```
se.Bl.Gr = sqrt(sig_sq_hat * ((1/n_i[1]) + (1/n_i[3])))
(int.Bl.Gr.95.indiv = ybar_i[1] - ybar_i[3] + c(-1, 1) * t_star * se.Bl.Gr)
```

```
[1] -0.7396511  3.2329845
```

Green vs Brown

```
se.Gr.Br = sqrt(sig_sq_hat*((1/n_i[2]) + (1/n_i[3])))
(int.Gr.Br.95.indiv = ybar_i[2] - ybar_i[3] + c(-1, 1) * t_star * se.Gr.Br)
```

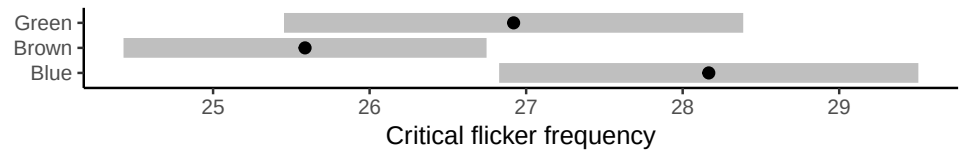
```
[1] -3.2025564  0.5375564
```

The emmeans package

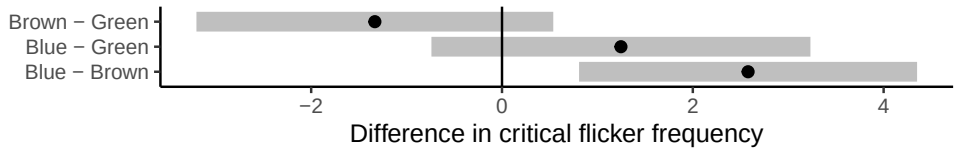
```
library(emmeans)
flicker_anova = aov(Flicker ~ Colour, data = flicker)
flicker_em = emmeans(flicker_anova, ~ Colour)
confint(flicker_em, adjust = "none")
```

```
 Colour emmean    SE df lower.CL upper.CL
 Blue     28.2 0.632 16     26.8     29.5
 Brown    25.6 0.547 16     24.4     26.7
 Green    26.9 0.692 16     25.5     28.4

Confidence level used: 0.95
```

```
confint(flicker_em, adjust = "none") |> plot(colors = "black") +
  labs(y = "", x = "Critical flicker frequency") +
  theme_classic()
```



```
contrast(flicker_em, method = "pairwise", adjust = "none") |> confint()  |>
  plot(colors = "black") + geom_vline(xintercept = 0) +
  labs(y = "", x = "Difference in critical flicker frequency") +
  theme_classic()
```



Summary of *individual* intervals

- So it would appear that individually the only "significantly different" pair is Blue and Brown.
- **However**, we have constructed each interval *without taking any regard of the others*.
- More precisely:
  - each interval has been constructed using a procedure so that when the model is correct, the probability that the "correct" population contrast is covered is 0.95…*individually*.
- **But**, what is the probability that **all** intervals cover their corresponding true values **simultaneously**?

22.3 Bonferroni method

The Bonferroni method

- Let A_1, A_2 and A_3 denote the events where each of the 3 intervals above cover the corresponding "true" value.
- Then, under our normal-equal-variance model, we have

$$P(A_1) = P(A_2) = P(A_3) = 0.95.$$

- However, what is $P(A \cap A_2 \cap A_3)$?
- This is "a bit hard", but we can derive a lower bound a bit more easily using the relation

$$P(A \cap A_2 \cap A_3)^c + A_1^c \cup A_2^c \cup A_3^c.$$

- Recall that $P(A \cup B) \leqslant P(A) + P(B)$, so we get

$$\begin{aligned} 1 - P(A_1 \cap A_2 \cap A_3) = P\left\{(A_1 \cap A_2 \cap A_3)^c\right\} &= P(A_1^c \cup A_2^c \cup A_3^c) \\ &\leqslant P(A_1^c) + P(A_2^c) + P(A_3^c) \\ &= 0.05 + 0.05 + 0.05 = 0.15. \end{aligned}$$

- Therefore, $P(A_1 \cap A_2 \cap A_3) \geqslant 0.85$

The *simultaneous coverage probability* of all 3 intervals is therefore at least 85%.

> **Take home message**
>
> When doing a Bonferroni correction, we take our original significance level and divide by the number of tests being performed.

Make the individual intervals a *little bit wider*

- This method shows us how to get a lower bound of 0.95:

– make each interval have *individual* coverage probability 1−(0.05)/3 = 59/60 = 0.98.3 (this requires the 1 − (0.05/6) quantile!):

```
t_simul = qt(1 - (0.05)/6, df = sum(n_i - 1))
t_simul
```

```
[1] 2.673032
```

emmeans package

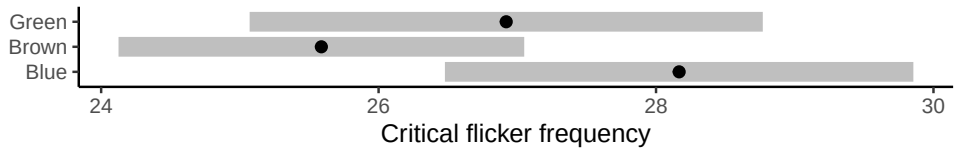
```
flicker_em = emmeans(flicker_anova, ~ Colour)
confint(flicker_em, adjust = "bonferroni")
```

```
 Colour emmean    SE df lower.CL upper.CL
 Blue     28.2 0.632 16     26.5     29.9
 Brown    25.6 0.547 16     24.1     27.0
 Green    26.9 0.692 16     25.1     28.8

Confidence level used: 0.95
Conf-level adjustment: bonferroni method for 3 estimates
```

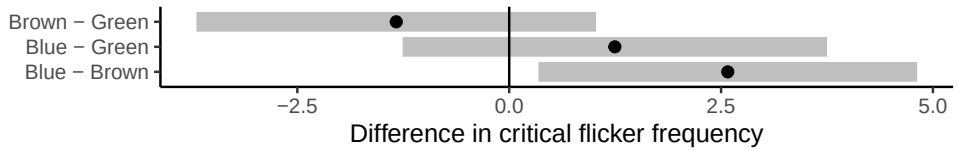
```
contrast(flicker_em, method = "pairwise", adjust = "bonferonni") |> confint()
```

```
 contrast      estimate    SE df lower.CL upper.CL
 Blue - Brown      2.58 0.836 16    0.345     4.81
 Blue - Green      1.25 0.937 16   -1.258     3.75
 Brown - Green    -1.33 0.882 16   -3.690     1.03

Confidence level used: 0.95
Conf-level adjustment: bonferroni method for 3 estimates
```

```
confint(flicker_em, adjust = "bonferroni") |> plot(colors = "black") +
  labs(y = "", x = "Critical flicker frequency") +
  theme_classic()
```




```
contrast(flicker_em, method = "pairwise", adjust = "bonferroni") |> confint()
    |>
  plot(colors = "black") + geom_vline(xintercept = 0) +
  labs(y = "", x = "Difference in critical flicker frequency") +
  theme_classic()
```



"Simultaneous" conclusions

- So, even though we "adjusted for multiplicity", the "Blue–Brown" difference is *still significant*, in the sense that the corresponding interval does **not** include zero.
- By increasing the confidence level of each *individual* comparison, we are able to make "simultaneous" valid statments about them all.

The general Bonferroni approach for k simultaneous comparisons

- In general, if we have k confidence intervals that we wish to have simultaneous coverage probability of (*at least*) $100(1-\alpha)\%$, we can achieve this (possibly conservatively!) by constructing each interval to have *individual* coverage probability $100(1-\alpha/k)\%$
- If we have g groups, then there are $k = \binom{g}{2} = \frac{g(g-1)}{2}$ possible pairs.
- For moderate-to-large g, this grows "quadratically" i.e. like g^2;
- other approaches e.g. Tukey's method, Scheffé's method can give "less conservative" (i.e. smaller) multipliers.

22.4 Multiple comparisons: pairwise t-tests

Pairwise t-tests

A general t tests for a contrast takes the form:

$$t_0 = \frac{\sum_{i=1}^{g} c_i \overline{y}_{i\bullet}}{\hat{\sigma}\sqrt{\sum_{i=1}^{g} c_i^2/n_i}}$$

Blue vs Brown

$$t_0 = \frac{\overline{y}_{1\bullet} - \overline{y}_{2\bullet}}{\hat{\sigma}\sqrt{1/n_1 + 1/n_2}}$$

```
se.Bl.Br = sqrt(sig_sq_hat *
                  ((1/n_i[1]) + (1/n_i[2])))
t_stat.Bl.Br = (ybar_i[1]-ybar_i[2])/se.Bl.Br
2*(1-pt(abs(t_stat.Bl.Br), df = sum(n_i-1)))
```

```
[1] 0.007079982
```

Blue vs Green

```
t_stat.Bl.Gr=(ybar_i[1]-ybar_i[3])/se.Bl.Gr
2*(1-pt(abs(t_stat.Bl.Gr),df=sum(n_i-1)))
```

```
[1] 0.2020033
```

Brown vs Green

```
t_stat.Gr.Br=(ybar_i[2]-ybar_i[3])/se.Gr.Br
2*(1-pt(abs(t_stat.Gr.Br),df=sum(n_i-1)))
```

```
[1] 0.1504046
```

Pairwise t-tests using emmeans

No adjustment

```
contrast(flicker_em, method = "pairwise", adjust = "none")
```

```
 contrast      estimate    SE df t.ratio p.value
 Blue - Brown      2.58 0.836 16   3.086  0.0071
 Blue - Green      1.25 0.937 16   1.331  0.2020
 Brown - Green    -1.33 0.882 16  -1.511  0.1504
```

Bonferroni adjustment (multiply unadjusted p-values by 3)

```
contrast(flicker_em, method = "pairwise", adjust = "bonferroni")
```

```
 contrast      estimate    SE df t.ratio p.value
 Blue - Brown      2.58 0.836 16   3.086  0.0212
 Blue - Green      1.25 0.937 16   1.331  0.6060
 Brown - Green    -1.33 0.882 16  -1.511  0.4512

P value adjustment: bonferroni method for 3 tests
```

Overall p-value from pairwise tests

Contrast	Unadjusted p-value	Adjusted p-value
Blue–Brown	0.0071	0.0212
Blue–Green	0.2020	0.6060
Brown–Green	0.1504	0.4512
Overall p-value		0.0212

```
summary(flicker_anova)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
Colour       2  23.00  11.499   4.802 0.0232 *
Residuals   16  38.31   2.394
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

22.5 Tukey's method

Tukey's method

- John Tukey derived the *exact* multiplier needed for simultaneous confidence intervals for all pairwise comparisons **when the sample sizes are equal**.
- It was later shown that when sample sizes are *unequal*, Tukey's procedure is *conservative* (simultaneous confidence level greater than $1 - \alpha$)
- Multiplicity-adjusted p-values can be obtained by inverting the intervals.
- The "overall ANOVA null hypothesis" can be tested using the *smallest* of these pairwise test p-values.
- Tukey named his method "Honest Significant Differences".
- It is implemented in the function `TukeyHSD()`, which takes as argument an `aov()` fit or using the `emmeans` package

```
contrast(flicker_em, method = "pairwise", adjust = "tukey") |> confint()
```

```
 contrast      estimate    SE df lower.CL upper.CL
 Blue - Brown      2.58 0.836 16    0.423    4.735
 Blue - Green      1.25 0.937 16   -1.171    3.664
 Brown - Green    -1.33 0.882 16   -3.609    0.944

Confidence level used: 0.95
Conf-level adjustment: tukey method for comparing a family of 3 estimates
```

```
contrast(flicker_em, method = "pairwise", adjust = "tukey")
```

```
 contrast      estimate    SE df t.ratio p.value
 Blue - Brown      2.58 0.836 16   3.086  0.0184
 Blue - Green      1.25 0.937 16   1.331  0.3994
 Brown - Green    -1.33 0.882 16  -1.511  0.3124

P value adjustment: tukey method for comparing a family of 3 estimates
```

22.6 Scheffé's method

Scheffé's simultaneous confidence interval method

- If we choose the special multiplier

$$t^{\star}_{\text{Sch}}(\alpha) = \sqrt{(g-1)F_{g-1,N-g}(\alpha)} = \sqrt{(g-1)*\texttt{qf}(1-\alpha, g-1, N-g)}$$

  and construct simultaneous confidence intervals for all possible contrasts according to

$$\sum_{i=1}^{g} c_i \overline{Y}_{i\bullet} \pm t^{\star}_{\text{Sch}}(\alpha)\, \hat{\sigma} \sqrt{\textstyle\sum_{i=1}^{g} \frac{c_i^2}{n_i}}$$

  then the probability that **all** sample contrasts include their true population values is **exactly** $1-\alpha$
- We effectively compare each contrast t-statistic to the $\sqrt{(g-1)F}$ distribution
- Any which exceeds *that* critical value is significant in this "simultaneous" sense.
- The *smallest* such p-value is exactly the ANOVA F-test p-value!

Scheffé's simultaneous confidence intervals using emmeans

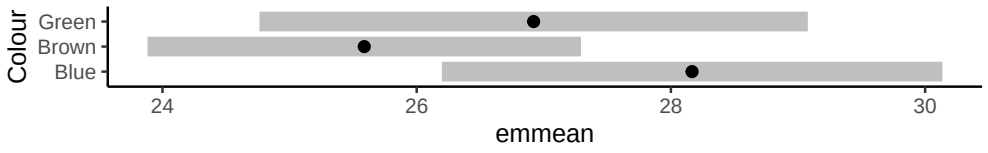
```
contrast(flicker_em, method = "pairwise", adjust = "scheffe") |> confint()
```

```
 contrast      estimate    SE df lower.CL upper.CL
 Blue - Brown      2.58 0.836 16    0.326     4.83
 Blue - Green      1.25 0.937 16   -1.279     3.77
 Brown - Green    -1.33 0.882 16   -3.711     1.05

Confidence level used: 0.95
Conf-level adjustment: scheffe method with rank 2
```

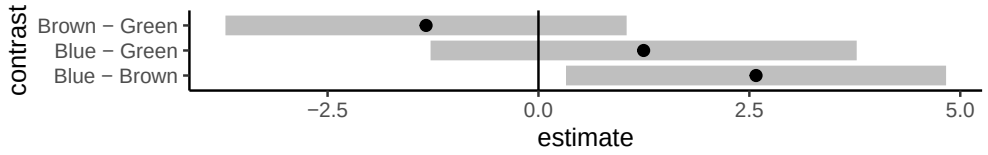
```
contrast(flicker_em, method = "pairwise", adjust = "scheffe")
```

```
 contrast      estimate    SE df t.ratio p.value
 Blue - Brown      2.58 0.836 16   3.086  0.0238
 Blue - Green      1.25 0.937 16   1.331  0.4319
 Brown - Green    -1.33 0.882 16  -1.511  0.3442

P value adjustment: scheffe method with rank 2
```

```
confint(flicker_em, adjust = "scheffe") |> plot(colors = "black") +
    theme_classic()
```




```
contrast(flicker_em, method = "pairwise", adjust = "scheffe") |> confint() |>
    plot(colors = "black") +
   geom_vline(xintercept = 0) + theme_classic()
```



Concluding remarks

- The ANOVA F-test alone may or may not address the important scientific questions in each example.
- Depending on the context, a test based on the most significant contrast(s) may be more useful than a straight F-test
- Bonferroni procedures are in general *conservative* i.e. p-values and confidence intervals may be larger than they really need to be.
  - alternative methods which may be more accurate i.e. less conservative exist: e.g. Tukey's method.
- Any contrasts must be decided upon before looking at the data. Otherwise we are data snooping.
- If we 'snoop' until we find a significant contrast, we must take account of that:
  - Scheffé's method permits unlimited data snooping
  - If we snoop only across k fixed contrasts e.g. all pairwise comparisons, we can use the Bonferroni method to adjust for that (but for large k Tukey's method or Scheffé's method may give smaller intervals).

23

What can we do when ANOVA assumptions fail?

23.1 What happens when assumptions fail?

Assumptions underlying ANOVA (and related methods)

We have learnt much about the comparison of several samples:

- The F-test (with p-value computed using the F-distribution)
- **contrasts** (and associated methods, all based on the t-distribution)
- **multiple comparisons**
  - Bonferroni
  - Tukey
  - Scheffé

However, underlying all of these are the assumptions that

- each sample is from a **normal population**;
- all **population variances are equal**.
  - so all populations are identical up to possible location shifts

What do we do if these assumptions are not reasonable?

Possible violations

- There are various ways the assumptions might be violated:
  - the normality might be ok, but **equal variances** might not be;
  - the normality might **not** be ok, but the "identical up to location shifts" assumption might be ok.

- There are a few tools we can appeal to:
  - simulation
  - resampling (together with *conditioning*).

23.2 Relaxing the equal variance assumption

Wolf River data

- The data in the file `wolfriver.csv` contains 30 measurements (10 at 3 depths: Bottom = 1; Middepth = 2; Surface = 3) on each of two chemicals, `Aldrin` and `HCB`.
  - these were taken downstream from an abandoned dump site near the Wolf River in Tennessee (Jaffe et al., 1982).
- It was believed the concentration might not be constant across different depths.
- We shall be *mainly* interested in each chemical separately, although considering them jointly is also a bit interesting…

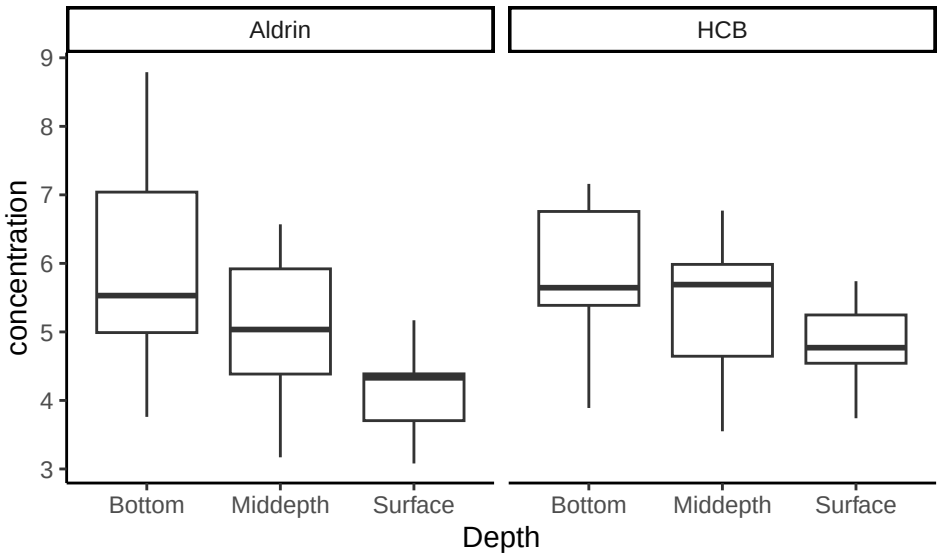
```
wolf = read_csv("https://raw.githubusercontent.com/DATA2002/data/master/
    wolfriver.csv")
glimpse(wolf)
```

```
Rows: 30
Columns: 3
$ Aldrin <dbl> 3.08, 3.58, 3.81, 4.31, 4.35, 4.40, 3.67, 5.17, 5.17, 4.35, 5.1
    ~
$ HCB    <dbl> 3.74, 4.61, 4.00, 4.67, 4.87, 5.12, 4.52, 5.29, 5.74, 5.48, 6.0
    ~
$ Depth  <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,
    ~
```

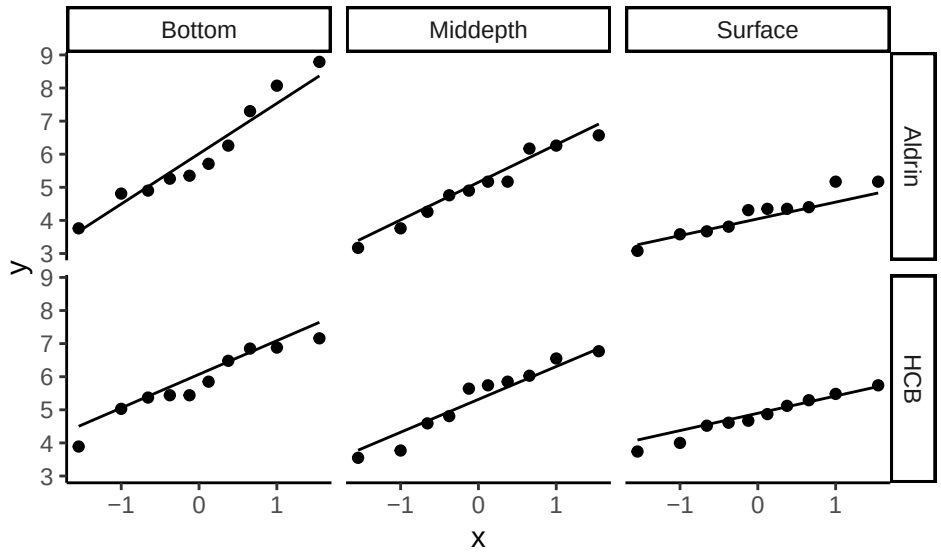
```
wolf = wolf |> mutate(
  Depth = case_when(
    Depth == 1 ~ "Bottom",
    Depth == 2 ~ "Middepth",
    Depth == 3 ~ "Surface"
  )
)
wolf |> count(Depth)
```

```
# A tibble: 3 x 2
  Depth        n
  <chr>    <int>
1 Bottom      10
2 Middepth    10
3 Surface     10
```

```
wolf_long = wolf |>
  pivot_longer(cols = Aldrin:HCB, names_to = "chemical",
          values_to = "concentration")
ggplot(wolf_long, aes(x = Depth, y = concentration)) +
  geom_boxplot() +
  facet_wrap(~chemical) +
  theme_classic()
```



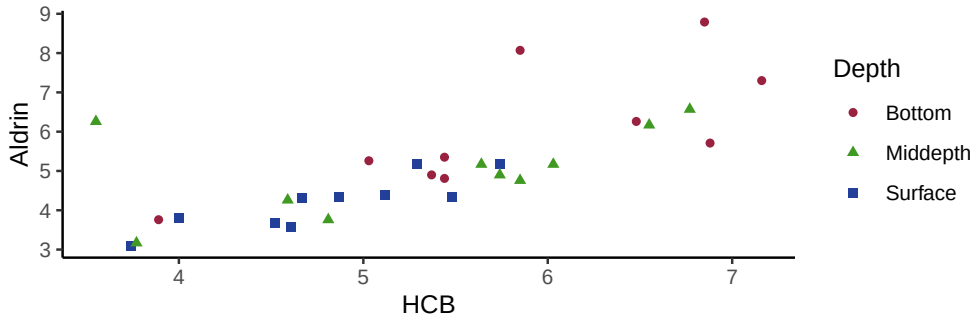

```
ggplot(wolf_long, aes(sample = concentration)) +
  geom_qq() + geom_qq_line() +
  facet_grid(chemical ~ Depth) +
  theme_classic()
```



Assumptions?

- *Both sets* of boxplots suggest a different spread in each group
  - normality is probably ok (points close to line in QQ plots)
- A *scatterplot* (tracking both chemicals together) reveals something interesting:

```
ggplot(wolf) + aes(x = HCB, y = Aldrin, shape = Depth, colour = Depth) +
  geom_point() +
  scale_colour_manual(values=c("#992240","#409922","#224099")) +
  theme_classic()
```



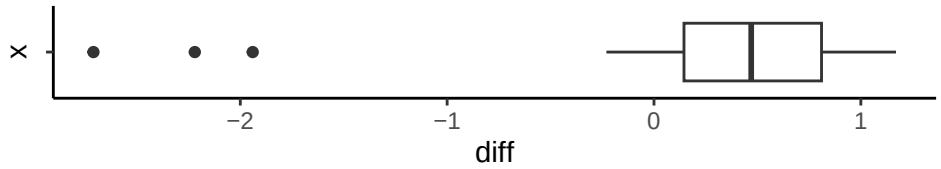
Outliers ?!

- There are 3 possible "outliers" (actually bivariate outliers, really):

```
wolf |> arrange(HCB-Aldrin)
```

```
# A tibble: 30 x 3
   Aldrin   HCB Depth
    <dbl> <dbl> <chr>
 1   6.26  3.55 Middepth
 2   8.07  5.85 Bottom
 3   8.79  6.85 Bottom
 4   5.26  5.03 Bottom
 5   7.3   7.16 Bottom
 6   5.35  5.44 Bottom
 7   5.17  5.29 Surface
 8   3.76  3.89 Bottom
 9   3.81  4    Surface
10   6.57  6.77 Middepth
# i 20 more rows
```

```
wolf |> mutate(
  diff = HCB - Aldrin
) |>
  ggplot() + aes(x = "", y = diff) +
  geom_boxplot() +
  coord_flip() +
  theme_classic()
```



…they might not have been so apparent when considering separately.

Relaxing the "common variance" assumption: all pairwise comparisons

- We could consider assuming normality, but dropping the "common variance" assumption.
- A simple way to do so is to consider all pairwise **Welch tests**, and apply a Bonferroni correction.
- Recall **Welch tests** only assume that each sample is normal, with possibly different variances σ_X^2 and σ_Y^2 and different means, and all random variables are independent.

$$T = \frac{\overline{X} - \overline{Y}}{\sqrt{\frac{S_X^2}{m} + \frac{S_Y^2}{n}}},$$

which is **approximately** $t_{d^*(m,n,\sigma_X,\sigma_Y)}$ under H_0, for a known function $d^*(\dots)$.
- The **Welch tests** is the what R defaults to in the `t.test()` function.

Welch test pairwise comparisons (unadjusted)

Middepth vs Surface

```
t.test(wolf$Aldrin[wolf$Depth=="Middepth"],
       wolf$Aldrin[wolf$Depth=="Surface"])$p.value
```

```
[1] 0.06053252
```

Middepth vs Bottom

```
t.test(wolf$Aldrin[wolf$Depth=="Middepth"],
       wolf$Aldrin[wolf$Depth=="Bottom"])$p.value
```

```
[1] 0.119901
```

Surface vs Bottom

```
t.test(wolf$Aldrin[wolf$Depth=="Surface"],
       wolf$Aldrin[wolf$Depth=="Bottom"])$p.value
```

```
[1] 0.005471484
```

- Since we are doing 3 pairwise comparisons, we multiply the "unadjusted" p-values by 3 to get "adjusted-for-multiplicity" p-values.
- The smallest of these can be used as a test that all (population) means are equal:

```
t.test(wolf$Aldrin[wolf$Depth=="Surface"],
       wolf$Aldrin[wolf$Depth=="Bottom"])$p.value
```

```
[1] 0.005471484
```

```
3 * t.test(wolf$Aldrin[wolf$Depth=="Surface"],
           wolf$Aldrin[wolf$Depth=="Bottom"])$p.value
```

```
[1] 0.01641445
```

- This is a perfectly valid p-value for testing the "global" or 'overall' hypothesis that all means are equal (*assuming all 3 populations are normal, but with possibly different variances*).

```
pairwise.t.test(wolf$Aldrin, wolf$Depth,
                p.adjust.method = "none", pool.sd = FALSE)
```

```
        Pairwise comparisons using t tests with non-pooled SD

data:  wolf$Aldrin and wolf$Depth

         Bottom Middepth
Middepth 0.1199 -
Surface  0.0055 0.0605

P value adjustment method: none
```

```
pairwise.t.test(wolf$Aldrin, wolf$Depth,
                p.adjust.method = "bonferroni", pool.sd = FALSE)
```

```
        Pairwise comparisons using t tests with non-pooled SD

data:  wolf$Aldrin and wolf$Depth

         Bottom Middepth
Middepth 0.360  -
Surface  0.016  0.182

P value adjustment method: bonferroni
```

Simultaneous confidence intervals

- To obtain a set of 3 simultaneous Bonferroni-style 95% confidence intervals, we compute 3 individual $(1 - (0.05/3)) \times 100 = 98.3\%$ intervals

Middepth vs Surface

```
t.test(
  wolf$Aldrin[wolf$Depth=="Middepth"],
  wolf$Aldrin[wolf$Depth=="Surface"],
  conf.level = 1-(0.05/3))$conf.int
```

```
[1] -0.2721228  1.9321228
attr(,"conf.level")
[1] 0.9833333
```

Middepth vs Bottom

```
t.test(
  wolf$Aldrin[wolf$Depth=="Middepth"],
  wolf$Aldrin[wolf$Depth=="Bottom"],
  conf.level = 1-(0.05/3))$conf.int
```

```
[1] -2.6317072  0.6277072
attr(,"conf.level")
[1] 0.9833333
```

Surface vs Bottom

```
t.test(
  wolf$Aldrin[wolf$Depth=="Surface"],
  wolf$Aldrin[wolf$Depth=="Bottom"],
  conf.level = 1-(0.05/3))$conf.int
```

```
[1] -3.3395315 -0.3244685
attr(,"conf.level")
[1] 0.9833333
```

- Of course, this does not include 0 because the *adjusted* p-value <0.05!

23.3 Relaxing the normality assumption

A p-value for the "global" hypothesis under weaker assumptions

- Under the formal ANOVA assumptions:
  - each population is normal
  - variances are the same

  the null hypothesis reduces to

> **All observations come from the same normal distribution**

A weaker set of assumptions *at least under the null hypothesis* is that

All observations come from the same distribution

The powerful tool of *conditioning*

- A common tool in testing is to condition on an "ancillary" statistic:
  - "ancillary statistic" just means a statistic that does not tell us anything useful.
- A familiar example is the **sign test**:
  - we usually *condition* on the number N of non-zeros (i.e. we *ignore the ties*)
- Then, p-values are in fact *conditional* probabilities, e.g. for a one-sided sign test based on the number S of positive signs, the p-value is

$$P(S \geqslant s \mid N = n) = P(B(n, 0.5) \geqslant s)$$

Conditioning on the combined sample

- If we combine all the groups into one combined sample (i.e. throw away the labels) then the remaining "data" tells us **nothing** about differences between groups i.e. what we are interested in.'
- In this sense, the *combined sample* is an "ancillary statistic".
- Once we condition on the *combined sample*, the only remaining "randomness" is the *allocation of observations to groups*.
- Under the null hypothesis of "no differences between groups" **all possible allocations are equally likely**.

Enumerating all possible allocations: exact p-values

- We can (in principle) compute an *exact conditional* p-value for **any "sensible" statistic** under this particular null hypothesis.
- There are actually

$$\frac{N!}{n_1! n_2! \dots n_g!}$$

  different possible allocations of the N total observations into groups of size $n_1, n_2, \dots, n_g$.
- We can (in principle) compute the value of the statistic under *each possible allocation*.
- Since each such value is *equally likely under the null hypothesis*, we can use this "sampling distribution" to compute a p-value.
- Suppose the statistic is T, the observed value is t_0 and larger values indicate more evidence against the null hypothesis.
- The *exact* conditional p-value is a *simple proportion*:

$$P(T \geqslant t_0 \mid \text{combined sample}) = \frac{\text{no. allocations with } T \geqslant t_0}{\text{total no. allocations}}.$$

- Unfortunately, unless the sample sizes are *very small*,
  - the total number of allocations is **MASSIVE**;
  - computing the value of the statistic over all possible allocations is not feasible.
- Fortunately, we can *estimate* this proportion by taking a sufficiently large random sample from the "population of all possible allocations".

Permutation tests

- In R, if the data is represented as a data frame with

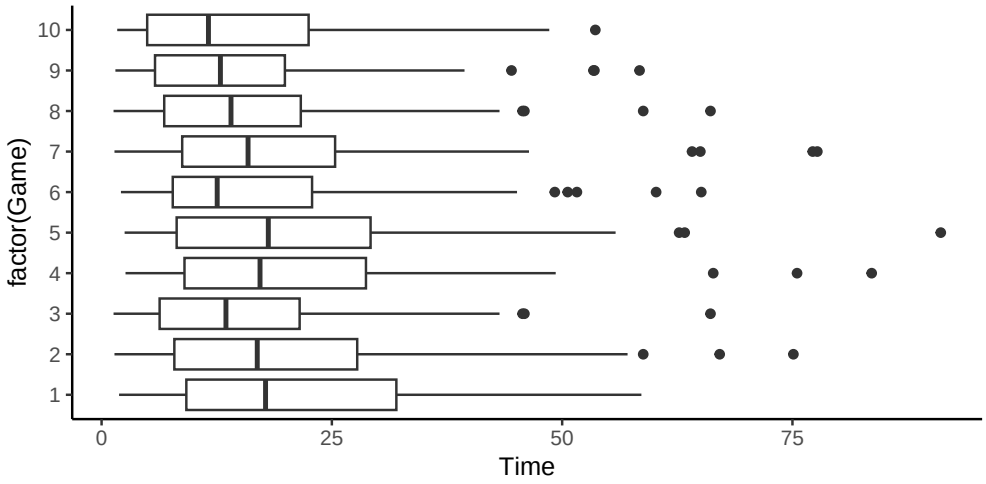
  - observations in one column and
  - groups indicated by a factor in another column

  then is it **easy** to obtain a "random" allocation:
  - simply randomly permute the observation vector, keeping the factor vector fixed.
- Do this a large number of times.
- The "observed proportion" of the times the statistic exceeds t_0 becomes an *estimate* of the "exact" p-value.
- This general procedure is known as a *permutation test*.

Rugby analysis

- In the early 1990's the rules for International Rugby were changed, apparently to make the game "more continuous", that is, for passages of "play" to be longer between stoppages.
- The lengths of time of passages of play were recorded in 10 games featuring the New Zealand national team (the "All Blacks"):
  - the first 5 games under the old rules
  - the last 5 games under the new rules

```
rugby = read_tsv("http://www.statsci.org/data/oz/rugby.txt")
ggplot(rugby) + aes(x = factor(Game), y = Time) +
  geom_boxplot() + coord_flip() +
  theme_classic()
```



Looks skewed, not normal…

F-test

- Let's try doing a permutation test using the *F*-statistic
- It is convenient to use the R function `anova(aov(...))` or `broom::tidy(aov(...))` instead of `summary(aov(...))` since the ANOVA table is returned as numbers in a matrix:

```
rugby_anova = aov(Time ~ factor(Game), data = rugby)
anova(rugby_anova)
```

```
Analysis of Variance Table

Response: Time
             Df Sum Sq Mean Sq F value    Pr(>F)
factor(Game)  9   6904  767.08  3.8867 7.335e-05 ***
Residuals   969 191241  197.36
```

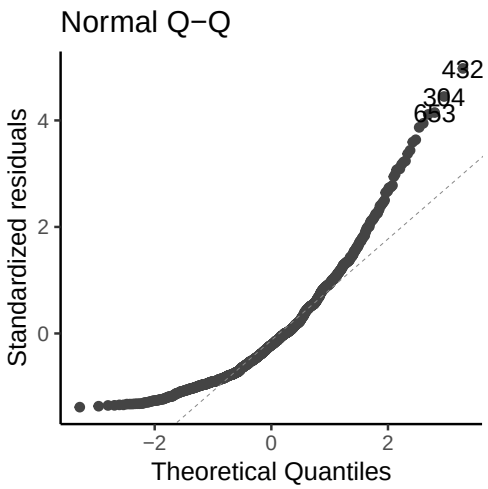
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(rugby_anova)[1,4]
```

```
[1] 3.886706
```

Check for normality

```
library(ggfortify)
autoplot(rugby_anova, which = 2) +
theme_classic()
```



The broom package

```
rugby_anova = aov(Time~factor(Game), data = rugby)
mod_sum = broom::tidy(rugby_anova)
mod_sum
```

```
# A tibble: 2 x 6
  term            df    sumsq meansq statistic    p.value
  <chr>        <dbl>    <dbl>  <dbl>     <dbl>      <dbl>
1 factor(Game)     9    6904.   767.      3.89  0.0000733
2 Residuals      969 191241.   197.     NA    NA
```

```
mod_sum$statistic[1]
```

```
[1] 3.886706
```

```
mod_sum |> kable(digits = c(0,0,0,1,3,4), booktabs=TRUE) |>
  kable_styling(position="center", latex_options = "hold_position")
```

term	df	sumsq	meansq	statistic	p.value
factor(Game)	9	6904	767.1	3.887	1e-04
Residuals	969	191241	197.4	NA	NA

Permutation tests

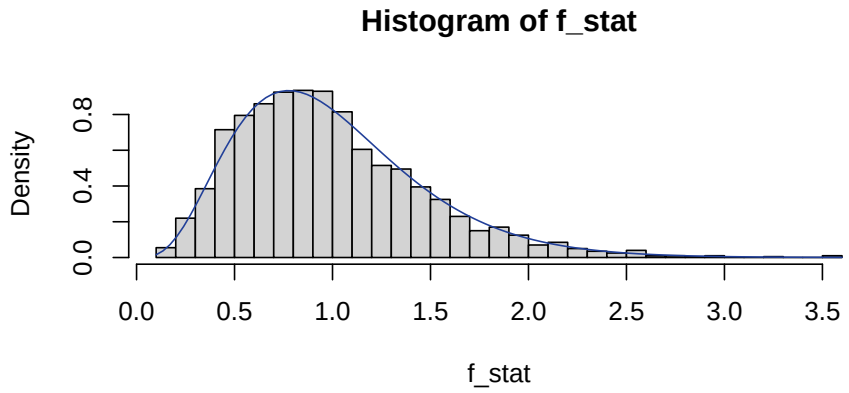
- The R function `sample()`, with only one vector argument, returns a permutation of that vector:

```
set.seed(88)
x = 1:5
sample(x)
```

```
[1] 3 1 5 4 2
```

- The following loop takes a sample of size B from all possible permutations and computes the value of the F- statistic

```
B = 2000
f_stat = vector(mode = "numeric", length = B)
for (i in 1:B){
  permuted_anova = aov(sample(rugby$Time) ~ factor(rugby$Game))
  f_stat[i] = broom::tidy(permuted_anova)$statistic[1]
}
hist(f_stat, probability = TRUE, breaks = 40)
curve(df(x, 9, 969), add = TRUE, col = "#224099")
```



- The F-distribution density is drawn over the histogram and the fit looks pretty good.
- Our *estimate* of the *exact conditional p-value* is obtained as follows:

```
t_0 = broom::tidy(rugby_anova)$statistic[1]
t_0
```

```
[1] 3.886706
```

```
mean(f_stat >= t_0)
```

```
[1] 0
```

- So of the 2000 random permutations 0 gave an F ratio bigger than (or equal to) 3.8867058.
- **We have avoided making any normality assumption here!**
- This says a lot about the *robustness* of the F-test…
  - the *conditional* distribution is very close to the corresponding F-distribution.

23.4 Using Ranks

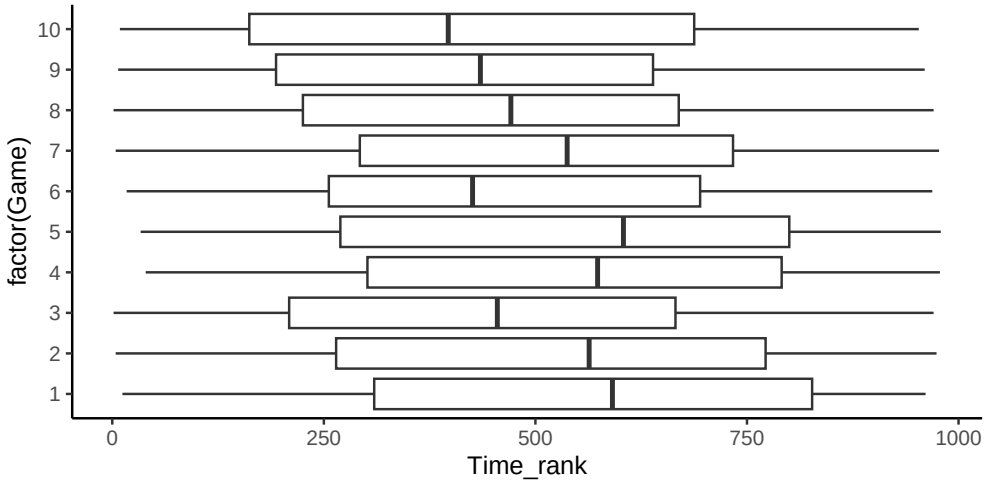
Kruskal-Wallis test

- This is performed by

  - replacing each observation by its "global" rank;
  - then computing the F-ratio as usual *on the ranks*.
- A p-value can be obtained
  - using a permutation test approach **or**
  - a "large-sample" χ^2 approximation can also be used

Rugby data (ranks)

```
rugby = rugby |> ungroup() |> mutate(Time_rank = rank(Time))
ggplot(rugby, aes(x = factor(Game), y = Time_rank)) +
  geom_boxplot() + coord_flip() +
  theme_classic()
```



Rugby data (ANOVA on the ranks)

Perform the usual ANOVA on the ranks.

```
rugby_kw = aov(Time_rank ~ factor(Game), data = rugby)
broom::tidy(rugby_kw)
```

```
# A tibble: 2 x 6
  term            df     sumsq  meansq statistic   p.value
  <chr>        <dbl>     <dbl>   <dbl>     <dbl>     <dbl>
1 factor(Game)     9  2452917. 272546.      3.49  0.000296
2 Residuals      969 75737977.  78161.     NA    NA
```

Kruskal-Wallis test statistic

- The traditional approach to the Kruskal-Wallis test uses a test statistic that is computed as a ratio (like the F-test)
  - the numerator is *exactly* the Treatment Sum of Squares of the ranks
  - the denominator is the *sample variance of all the ranks*!
    - this denominator is not random (it is the same regardless of the allocation).

$$T = \frac{\text{Treatment SS of the ranks}}{\text{Variance of all the ranks}}$$

- When H_0 is true, it has an *approximate* χ^2_{g-1} distribution.

Performing the KW test

- "Manually" computing the test statistic:

```
rugby_rank_anova = aov(Time_rank ~ factor(Game), data = rugby)
rank_treat_SS = broom::tidy(rugby_rank_anova)$sumsq[1]
t0 = rank_treat_SS/var(rugby$Time_rank)
t0
```

```
[1] 30.68072
```

```
pchisq(t0, df = 10 - 1, lower.tail = FALSE)
```

```
[1] 0.0003357678
```

- This statistic is computed (and a resultant "approximate" p-value is obtained) via the R function `kruskal.test()`:

```
kruskal.test(Time ~ factor(Game), data = rugby)
```

```
        Kruskal-Wallis rank sum test

data:  Time by factor(Game)
Kruskal-Wallis chi-squared = 30.681, df = 9, p-value = 0.0003358
```

Workflow: Kruskal-Wallis test

- **Hypotheses**: H_0 : the response variable is distributed identically for all g groups vs H_1 : the response variable is systematically higher for at least one group
- **Assumptions**: Observations are independent within each group and groups are independent of each other. The different groups follow the same distribution (differing only by the location parameter).
- **Test Statistic**: Under the null hypothesis the Kruskal-Wallis test statistic approximately follows a χ^2 distribution with $g-1$ degrees of freedom
- **p-value**: $P(T \geqslant t_0) = P(\chi^2_{g-1} \geqslant t_0)$.
- **Decision**: If the p-value is less then α we reject the null hypothesis and conclude that the population mean of at least one group is significantly different to the others. If the p-value is larger than α we do not reject the null hypothesis and conclude that there is no significant difference between the population means.

Permuted ranks

- The permutation test approach is valid for any "sensible" statistic;
  - it only assumes the same distribution in each group under the null hypothesis.
- What of the "sensible statistic"?
  - If the data are truly normal, the F-statistic makes sense
  - is it still "sensible" if the normality assumption is being relaxed?
  - could also do a permutation test using the Kruskal-Wallis statistic

Rugby data (KW permutation test)

```
set.seed(88)
B = 2000
Game = rugby$Game
```

```
Time = rugby$Time
kw_stat = vector("numeric", length = B)
for (i in 1:B){
  aov_rank = aov(rank(sample(Time)) ~ factor(Game))
  kw_stat[i] = broom::tidy(aov_rank)$statistic[1]
}
original_rank_mod = aov(rank(Time)~factor(Game))
t0 = broom::tidy(original_rank_mod)$statistic[1]
t0
```

```
[1] 3.486988
```

```
mean(kw_stat >= t0)
```

```
[1] 0.0015
```

24

Two-way ANOVA

24.1 Two-way ANOVA: adjusting for "blocks"

Electrode testing

Berry (1987) presents data on skin resistance.

- 5 types of **electrode** were attached to each of 16 **subjects** and the resistance measured
- **Aim:** do all 5 types perform similarly?
- There may be differences **between the electrode** types and/or **between the subjects**.
  - If there is, this will "add" to the overall variation. Can we adjust for this?

Outliers?

Berry (1987) notes that there may have been interference by a hairy arm:

> After obtaining the results the experimenters decided that the reason for the two large readings on subject 15 was the excessive amount of hair on those parts of the subject's arm. They concluded that this subject's data should be deleted. Whether these readings are contaminants is not clear; the amount of hair present for the other 78 readings was not assessed relative to these two and no such assessment was made independent of the results.

Read in the data

```
library(tidyverse)
resist = read_tsv("https://raw.githubusercontent.com/DATA2002/data/master/
    resist.txt")
glimpse(resist)
```

```
Rows: 16
Columns: 6
$ Subject <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
$ E1      <dbl> 500, 660, 250, 72, 135, 27, 100, 105, 90, 200, 15, 160, 250, 1
    ~
$ E2      <dbl> 400, 600, 370, 140, 300, 84, 50, 180, 180, 290, 45, 200, 400,
    ~
$ E3      <dbl> 98, 600, 220, 240, 450, 135, 82, 32, 220, 320, 75, 300, 50, 23
    ~
$ E4      <dbl> 200, 75, 250, 33, 430, 190, 73, 58, 34, 280, 88, 300, 50, 20,
    ~
$ E5      <dbl> 250, 310, 220, 54, 70, 180, 78, 32, 64, 135, 80, 220, 92, 150,
    ~
```

```
# convert Subject from integer to factor
resist$Subject = factor(resist$Subject)
```

Wide form data

Our data is currently in a "wide" format: each person is in a row but the "treatments" (electrodes) are in separate columns.

```
resist
```

```
# A tibble: 16 x 6
   Subject    E1    E2    E3    E4    E5
   <fct>   <dbl> <dbl> <dbl> <dbl> <dbl>
 1 1         500   400    98   200   250
 2 2         660   600   600    75   310
 3 3         250   370   220   250   220
 4 4          72   140   240    33    54
 5 5         135   300   450   430    70
 6 6          27    84   135   190   180
 7 7         100    50    82    73    78
 8 8         105   180    32    58    32
 9 9          90   180   220    34    64
10 10        200   290   320   280   135
11 11         15    45    75    88    80
12 12        160   200   300   300   220
13 13        250   400    50    50    92
14 14        170   310   230    20   150
15 15         66  1000  1050   280   220
16 16        107    48    26    45    51
```

Reshaping the data

To analyse the data in R we need it in "long" format. I.e. we want a data frame with 3 columns:

- one with the **response**;
- a factor indicating **treatment** (i.e. electrode type);
- *another* factor indicating the **Subject**.

```
resist_long = resist |>
  pivot_longer(cols = E1:E5,
               names_to = "electrode",
               values_to = "resistance")
glimpse(resist_long)
```

```
Rows: 80
Columns: 3
$ Subject    <fct> 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4,
    ~
$ electrode  <chr> "E1", "E2", "E3", "E4", "E5", "E1", "E2", "E3", "E4", "E5",
    ~
```

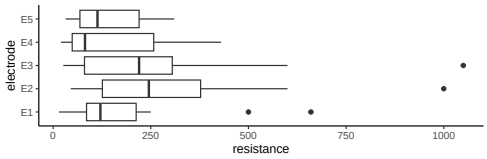
```
$ resistance <dbl> 500, 400, 98, 200, 250, 660, 600, 600, 75, 310, 250, 370, 2
    ~
```

Log transformation

- Let's look at box plots of each **electrode type** ignoring **subject**.

Linear Scale

```
p1 = ggplot(resist_long) +
  aes(x = electrode, y = resistance) +
  geom_boxplot() + coord_flip() +
  theme_classic()
p1
```



Log Scale

```
p1 + scale_y_log10()
```

- The log-transformed data looks a little better: let's use that (for the moment).
- Note: `scale_y_log10()` doesn't change the data…it only transforms the plot axis.

Formulas in R

The basic structure of a formula in R is:

```
y ~ x # it's a tilde (squiggly line) between y and x
```

This can be read as "y is a function of x" or "y against x" or "y by x" or "y twiddles x". Examples we've seen so far:

```
t.test(y ~ group)
ggplot(df) + facet_wrap( ~ group) # one sided formula
ggplot(df) + facet_grid(group1 ~ group2)
aov(y ~ group)
```

Sometimes we might want to consider multiple "explanatory" variables:

```
y ~ x1 + x2 + x3
```

- Define a new variable `y` in our data frame `resist_long` that is the log of the resistance measurement:

  ```
  resist_long$y = log(resist_long$resistance)
  ```

- Let's start with an ordinary (one-way) ANOVA ignoring **Subject**:

  ```
  fit1 = aov(y ~ electrode, data = resist_long)
  summary(fit1)
  ```

  ```
              Df Sum Sq Mean Sq F value Pr(>F)
  electrode    4   5.09  1.2719   1.503   0.21
  Residuals   75  63.48  0.8464
  ```

- This is clearly "not significant".

Adjusting for Subject

We can add Subject as an extra factor variable in our **formula** to indicate that it should be used to help "explain" `y`. The formula that we use in the `aov()` is now: `y ~ Subject + electrode`

```
fit2 = aov(y ~ Subject + electrode, data = resist_long)
summary(fit2)
```

```
           Df Sum Sq Mean Sq F value   Pr(>F)
Subject    15  33.27  2.2180   4.405 1.77e-05 ***
electrode   4   5.09  1.2719   2.526     0.05 *
Residuals  60  30.21  0.5036
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The electrode p-value is a lot smaller than before. What is going on here?

Compare the two tables carefully:

```
summary(fit1) # y ~ electrode
```

```
           Df Sum Sq Mean Sq F value Pr(>F)
electrode   4   5.09  1.2719   1.503   0.21
Residuals  75  63.48  0.8464
```

```
summary(fit2) # y ~ Subject + electrode
```

```
           Df Sum Sq Mean Sq F value   Pr(>F)
Subject    15  33.27  2.2180   4.405 1.77e-05 ***
electrode   4   5.09  1.2719   2.526     0.05 *
Residuals  60  30.21  0.5036
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The **Residual** sum of squares from `fit1` (63.48, on 75 df) is being *decomposed* into two components:
  - the `fit2` **Subject** sum of squares (33.27, on **15** df) and
  - the `fit2` **Residual** sum of squares (30.21, on **60** df)

Decomposition of the residual sum of squares

- For `fit2`, the Residual sum of squares is much smaller than for `fit1`, but the degrees of freedom is only a little less:
  - this gives a much smaller **Residual Mean Square** (0.5036, compared to 0.8464 for `fit1`);
  - this in turn gives a bigger (*treatment-to-residual*) F-ratio (2.526, compared to 1.503 for `fit1`);
  - **crucially** the p-value has been reduced from 0.21 to 0.05:
    - the effect is now (at least mildly) significant!
- We are now ready to study the mathematics of what we have done here;
  - but first, we return briefly to ordinary ANOVA to learn how to change parameters.

Changing parameters

- For **ordinary one-way ANOVA** we have have written the model as: for $i = 1, \dots, g,\ j = 1, \dots, n_i$

$$Y_{ij} \sim \mathcal{N}(\mu_i, \sigma^2)$$

  - There are g unknown mean-value parameters, and 1 unknown variance parameter.
- Another way to write this is, for $i = 1, \dots, g,\ j = 1, \dots, n_i$,

$$Y_{ij} = \mu_i + \varepsilon_{ij}$$

  where the ε_{ij}'s are idd $\mathcal{N}(0, \sigma^2)$.

  - **Again**, there are g unknown mean-value parameters, and 1 unknown variance parameter.

Yet another way...

- A *third* way to write the model is based on expressing each μ_i as
  - an *overall mean* μ (with no subscript) plus
  - an *adjustment* α_i for i-th level of the treatment:

$$\mu_i = \mu + \alpha_i$$

- THis leads to the model: for $i = 1, \ldots, g,\ j = 1, \ldots, n_i$,

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij}.$$

- Note there are now $g + 1$ "mean" parameters (sort of): $\mu, \alpha_1, \ldots, \alpha_g$
  - we have "created" another parameter

An extra constraint

- In fact, depending on how μ is defined, the α_i's necessarily obey a certain constraint.
- The overall mean is defined as some kind of (weighted) average of the μ_i's:

$$\mu = \sum_{i=1}^{g} w_i \mu_i$$

  - Then each $\alpha_1 = \mu_i = \mu$
- Necessarily, the same weighted average of the α_i's is:

$$\sum_{i=1}^{g} w_i \alpha_i = \sum_{i=1}^{g} w_i (\mu_i - \mu) = \left(\sum_{i=1}^{g} w_i \mu_i \right) - \mu \sum_{i=1}^{g} w_i = \mu - \mu = 0.$$

- SO in fact, knowing $g - 1$ of the α_i's means you also know the final one.

Estimating these new "parameters"

- A common choice for the "weighted average" is

$$\mu = \frac{1}{N} \sum_{i=1}^{g} n_i \mu_i = \frac{\sum_{i=1}^{g} n_i \mu_i}{\sum_{i=1}^{g} n_i},$$

  which is the the *expectation of the grand mean* $\overline{Y}_{\bullet\bullet}$
- This can be estimated using the *observed* grand mean $\overline{y}_{\bullet\bullet}$
- Each α_i represents the difference between each group mean and the overall mean,
  - it's thus naturally estimated using the difference

$$\hat{\alpha}_i = \overline{y}_{i\bullet} - \overline{y}_{\bullet\bullet}$$

The two-way ANOVA model

- The model we shall fit to the electrode data is the following:

$$Y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}$$

where

μ = overall mean
α_i = adjustment for electrode type i for $i = 1, 2, \ldots, g$
β_j = adjustment for subject j for $j = 1, 2, \ldots, n$

and n is the common sample (block) size and the ε_{ij}'s are idd $\mathcal{N}(0, \sigma^2)$.

- So each Y_{ij} has a possibly different expectation $\mu_{ij} = \mu + \alpha_i + \beta_j$, but these have an **additive structure**:
  - the ng different means are explained by $1 + (g-1) + (n-1) = g + n - 1$ (free) parameters.

Estimating parameters

- As all "sample sizes" are the same, the overall mean can be thought of as just the mean of the μ_i's
  - it is naturally estimated using the overall mean $\overline{y}_{\bullet\bullet}$
- Also, each α_i, the "adjustment" for **electrode type** i, is naturally estimated using the difference

$$\overline{y}_{i\bullet} - \overline{y}_{\bullet\bullet}.$$

- Similarly, each β_j, the adjustment for **subject** j, is naturally estimated using the difference

$$\overline{y}_{\bullet j} - \overline{y}_{\bullet\bullet}.$$

The two-way decomposition

- Each observation therefore, may be notionally split up into 4 pieces:

$$y_{ij} = \underbrace{\overline{y}_{\bullet\bullet}}_{\hat{\mu}} + \underbrace{(\overline{y}_{i\bullet} - \overline{y}_{\bullet\bullet})}_{\hat{\alpha}_i} + \underbrace{(\overline{y}_{\bullet j} - \overline{y}_{\bullet\bullet})}_{\hat{\beta}_j} + \underbrace{(y_{ij} - \overline{y}_{i\bullet} - \overline{y}_{\bullet j} + \overline{y}_{\bullet\bullet})}_{\hat{\varepsilon}_{ij}}$$

- The final part $\hat{\varepsilon}_{ij}$ is the (i, j) **residual** or estimated error.
- We can "analyse the variance" here in the same way as the ordinary "one-way" ANOVA model.

Decomposing the total sum of squares

$$\begin{aligned}
\sum_{i=1}^{g}\sum_{j=1}^{n}(y_{ij} - \overline{y}_{\bullet\bullet})^2 &= \sum_{i=1}^{g}\sum_{j=1}^{n}\left\{(\overline{y}_{i\bullet} - \overline{y}_{\bullet\bullet}) + (\overline{y}_{\bullet j} - \overline{y}_{\bullet\bullet}) + (y_{ij} - \overline{y}_{i\bullet} - \overline{y}_{\bullet j} + \overline{y}_{\bullet\bullet})\right\}^2 \\
&= \sum_{i=1}^{g} n(\overline{y}_{i\bullet} - \overline{y}_{\bullet\bullet})^2 + \sum_{j=1}^{n} g(\overline{y}_{\bullet j} - \overline{y}_{\bullet\bullet})^2 + \\
&\quad \sum_{i=1}^{g}\sum_{j=1}^{n}(y_{ij} - \overline{y}_{i\bullet} - \overline{y}_{\bullet j} + \overline{y}_{\bullet\bullet})^2 + \\
&\quad \text{cross-product terms which are all zero} \\
&= \text{Treatment sum of squares} + \\
&\quad \text{Block sum of squares} + \\
&\quad \text{Residual sum of squares}
\end{aligned}$$

24.2 The two-way ANOVA table

The two-way ANOVA table

- With the above sum of squares definitions, the "two-way ANOVA" (not the best name) table is given as follows:

Source of Variation	Sum of squares	df	Mean square	F-ratio
Blocks	Block SS	$n-1$		
Treatments	Trt SS	$g-1$	Trt MS $= \dfrac{\text{Trt SS}}{g-1}$	$\dfrac{\text{Trt MS}}{\text{Res MS}}$
Residual	Res SS	$(n-1)(g-1)$	Res MS $= \dfrac{\text{Res SS}}{(n-1)(g-1)}$	
Total	Total SS	$ng-1$		

- The total sum of squares is $\sum_{i=1}^{g}\sum_{j=1}^{n}(y_{ij}-\overline{y}_{\bullet\bullet})^2$, and the total sample size is $N = ng$.
- Once appropriately coded using R, this can be obtained using `summary(aov(...))`, `anova(aov(...))` or even `anova(lm(...))` or using `broom::tidy(aov(...))`.
- We compare the observed value of the (Treatment-to-Residual) F-ratio to the $F_{g-1,(n-1)(g-1)}$ distribution.

The purpose of blocking

- A two-way ANOVA with blocking can be thought of as a generalisation of the paired t-test where each **pair** is a **block**
- In the paired t-test, the idea is to *remove* the variation "between pairs", to more accurately compare the two treatment levels *within each pair*.
  - the "within pair" difference is then averaged over all pairs to get the "treatment effect".
- We are not interested in "testing for a Block effect", we are only interested in comparing **Treatments**.
- We are nonetheless adjusting for **Blocks**, to more accurately compare **Treatments**
- Although the **Treatments sum of squares** and **Blocks sum of squares** are *mathematically* identical, they are playing very different *scientific* roles.

24.3 Behaviour of sums of squares under the two-way ANOVA model

Two-way ANOVA sums of squares behaviour

Recall y_{ij} (the observation in block j receiving treatment level i) is modelled as the value taken by

$$Y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}$$

for $i - 1, 2, \ldots, g$ and $j = 1, 2, \ldots, n$ where μ is the overall mean,

$$\begin{aligned}\alpha_i &= \text{ adjustment for treatment level } i,\\ \beta_j &= \text{ adjustment for block } j,\\ \varepsilon_{ij} &\sim \mathcal{N}(0, \sigma^2),\end{aligned}$$

all random variables are independent and the following constraints are satisfied,

$$\sum_{i=1}^{g} \alpha_i = 0 \text{ and } \sum_{j=1}^{n} \beta_j = 0.$$

Treatment sum of squares

- The treatment sum of squares is

$$\sum_{i=1}^{g} n(\overline{Y}_{i\bullet} - \overline{Y}_{\bullet\bullet})^2 = n \sum_{i=1}^{g} (\alpha_i + \overline{\varepsilon}_{i\bullet} - \overline{\varepsilon}_{\bullet\bullet})^2 .$$

- Under the null hypothesis $H_0 : \alpha_1 = \cdots = \alpha_g = 0$, this is

$$n \underbrace{\sum_{i=1}^{g} (\overline{\varepsilon}_{i\bullet} - \overline{\varepsilon}_{\bullet\bullet})^2}_{\sim \frac{\sigma^2}{n}\chi^2_{g-1}} \sim n\left(\frac{\sigma^2}{n}\chi^2_{g-1}\right) \sim \sigma^2\chi^2_{g-1}$$

  since under the model the $\overline{\varepsilon}_{i\bullet}$'s are idd normal with variance $\frac{\sigma^2}{n}$.
  - This is the same as for one-way ANOVA.

Residual sum of squares

How is the residual sum of squares distributed?

- We have the identity (with $N = ng$),

$$\underbrace{\sum_{i=1}^{g}\sum_{j=1}^{n} (\varepsilon_{ij} - \overline{\varepsilon}_{i\bullet})^2}_{\sim\sigma^2\chi^2_{N-g}} = \underbrace{\sum_{j=1}^{n} g(\overline{\varepsilon}_{\bullet j} - \overline{\varepsilon}_{\bullet\bullet})^2}_{\sim\sigma^2\chi^2_{n-1}} + \underbrace{\sum_{i=1}^{g}\sum_{j=1}^{n} (\varepsilon_{ij} - \overline{\varepsilon}_{i\bullet} - \overline{\varepsilon}_{\bullet j} + \overline{\varepsilon}_{\bullet\bullet})^2}_{\sim ???}$$

  Roughly speaking this is

  One-way Res SS = Block SS of Errors + Two-way Res SS

  - It can be shown that the two terms on the RHS are independent, so the last double sum **must be** $\sigma^2\chi^2_{N-g-(n-1)} \sim \sigma^2\chi^2_{(n-1)(g-1)}$

Two-way ANOVA F-ratio

- In summary:
  - the residual sum of squares always follows a $\sigma^2\chi^2_{(n-1)(g-1)}$ distribution (regardless of whether the null hypothesis is true or not);
  - *if the null hypothesis of "no treatment effect" is true*, the treatment sum of squares follows a $\sigma^2\chi^2_{g-1}$ distribution.

  Therefore, if the *null hypothesis is true*, the F-ratio

$$\frac{\text{Treatment mean square}}{\text{Residual mean square}} \sim \frac{\chi^2_{g-1}/(g-1)}{\chi^2_{(n-1)(g-1)}/(n-1)(g-1)} \sim F_{g-1,(n-1)(g-1)},$$

  Otherwise, it tends to take *larger* values (as it does for one-way ANOVA).

24.4 Post hoc tests and multiple comparisons methods

Bonferroni method

- Under our "new" parametrisation, each "treatment difference" e.g. $\alpha_1 - \alpha_2$ is still naturally estimated using the corresponding treatment level mean difference.
- For example, we would estimate
  - α_1 using $\overline{y}_{1\bullet} - \overline{y}_{\bullet\bullet}$
  - α_2 using $\overline{y}_{2\bullet} - \overline{y}_{\bullet\bullet}$
  - $\alpha_1 - \alpha_2$ using $\overline{y}_{1\bullet} - \overline{y}_{2\bullet}$
- Under this two-way model, the corresponding random mean difference is distributed as

$$\overline{Y}_{1\bullet} - \overline{Y}_{2\bullet} \sim \mathcal{N}\left(\alpha_1 - \alpha_2,\ \frac{2\sigma^2}{n}\right),$$

since all treatment groups have a common sample size n.

An *individual* t-test and confidence interval

Assume we're testing with a significance level, α.

- We estimate the standard error $\sigma\sqrt{\frac{2}{n}}$ by plugging in $\hat{\sigma}^2$ (the **Residual Mean Square**) as the estimate of σ^2.
- Suppose $c(\alpha)$ satisfies $P(-c(\alpha) \leqslant t_{(n-1)(g-1)} \leqslant (\alpha)) = 1 - \alpha$.
- An **individual** level α t-test for comparing groups 1 and 2 would therefore reject for

$$\frac{|\overline{y}_{1\bullet} - \overline{y}_{2\bullet}|}{\hat{\sigma}\sqrt{\frac{2}{n}}} > c(\alpha).$$

- An **individual** $100(1-\alpha)\%$ confidence interval for $\alpha_1 - \alpha_2$ would be given by

$$\overline{y}_{1\bullet} - \overline{y}_{2\bullet} \pm c(\alpha)\,\hat{\sigma}\sqrt{\frac{2}{n}}.$$

Adjusting for multiplicity

- If we are performing k simultaneous comparisons, we replace α with α/k.
- This means that we simply replace $c(\alpha)$ with $c(\alpha/k)$ and
  - perform each t-test at "level α/k"
  - construct each confidence interval at the $100(1-\alpha/k)\%$ confidence level.
- Then the "overall performance" of the k procedures taken "simultaneously" is "at level α": under the model,
  - the probability of incorrectly rejecting any of the t-tests is no more than α (family wise error rate)
  - the probability that all true values are included in the corresponding confidence interval is $1 - \alpha$

Multiplicity-adjusted p-values

- If we are doing k simultaneous t-tests, we reject each one at "overall level α" if and only if each *individual unadjusted* p-value is less than α/k
- This is equivalent to rejecting if k **times the unadjusted p-value** exceeds α.
- We thus *define* each "adjusted" p-value as k times the corresponding unadjusted p-value.

The Bonferroni test

- If we are doing *all pairwise comparisons* across g groups then there are $\binom{g}{2}$ of these.
- If **none** of the tests end up rejecting (after adjusting for multiplicity) then this means the *smallest individual p-value* (corresponding to the *most significant pairwise difference*) was greater than $\alpha/\binom{g}{2}$
- Therefore the p-value for the Bonferroni test is simply $\binom{g}{2}$ times the smallest unadjusted p-value.
- If this test rejects, we can "post hoc" identify which comparisons are significant by identifying which *adjusted* p-values are less than α

Electrode data

Recall the two-way ANOVA table for the electrode data:

```
fit2 = aov(y ~ Subject + electrode, data = resist_long)
anova(fit2)
```

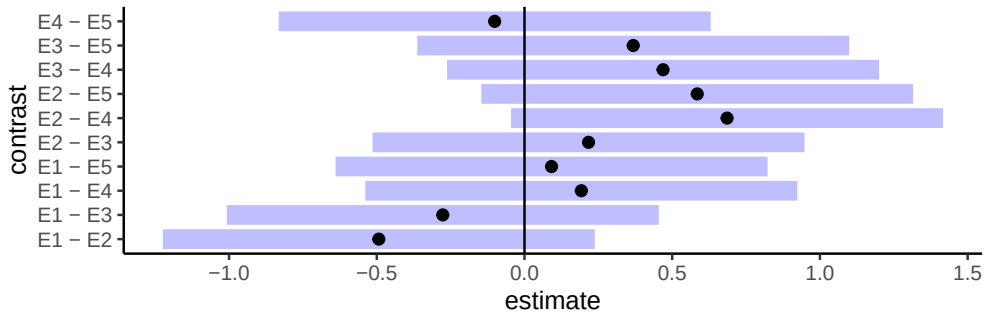
```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
Subject   15 33.269 2.21797  4.4047 1.768e-05 ***
electrode  4  5.087 1.27185  2.5258   0.04996 *
Residuals 60 30.213 0.50355
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The emmeans package

```
fit2_emmeans = emmeans(fit2, ~ electrode)
contrast(fit2_emmeans, method = "pairwise", adjust = "bonferroni")
```

```
 contrast estimate    SE df t.ratio p.value
 E1 - E2   -0.4932 0.251 60  -1.966  0.5394
 E1 - E3   -0.2765 0.251 60  -1.102  1.0000
 E1 - E4    0.1925 0.251 60   0.767  1.0000
 E1 - E5    0.0915 0.251 60   0.365  1.0000
 E2 - E3    0.2167 0.251 60   0.864  1.0000
 E2 - E4    0.6857 0.251 60   2.733  0.0823
 E2 - E5    0.5847 0.251 60   2.331  0.2315
 E3 - E4    0.4690 0.251 60   1.869  0.6645
 E3 - E5    0.3681 0.251 60   1.467  1.0000
 E4 - E5   -0.1010 0.251 60  -0.402  1.0000

Results are averaged over the levels of: Subject
P value adjustment: bonferroni method for 10 tests
```

```
contrast(fit2_emmeans, method = "pairwise", adjust = "bonferroni") |>
  plot() + geom_vline(xintercept = 0) +
  theme_classic()
```



Tukey's method

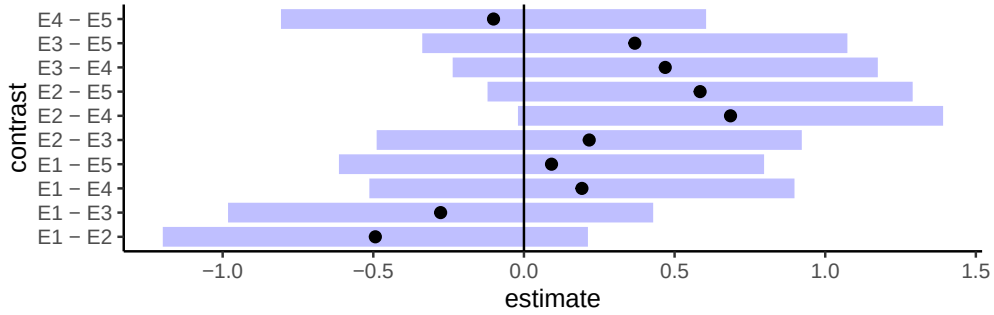
- In this case Tukey's method can be applied, and it is exact.

```
# TukeyHSD(fit2, which = "electrode") # an alternative not using emmeans
contrast(fit2_emmeans, method = "pairwise", adjust = "tukey")
```

```
 contrast estimate    SE df t.ratio p.value
 E1 - E2   -0.4932 0.251 60  -1.966  0.2950
 E1 - E3   -0.2765 0.251 60  -1.102  0.8047
 E1 - E4    0.1925 0.251 60   0.767  0.9390
 E1 - E5    0.0915 0.251 60   0.365  0.9961
 E2 - E3    0.2167 0.251 60   0.864  0.9090
 E2 - E4    0.6857 0.251 60   2.733  0.0607
 E2 - E5    0.5847 0.251 60   2.331  0.1495
 E3 - E4    0.4690 0.251 60   1.869  0.3448
 E3 - E5    0.3681 0.251 60   1.467  0.5875
 E4 - E5   -0.1010 0.251 60  -0.402  0.9943

Results are averaged over the levels of: Subject
P value adjustment: tukey method for comparing a family of 5 estimates
```

- The p-value for the "Tukey test" is 0.061.

```
contrast(fit2_emmeans, method = "pairwise", adjust = "tukey") |>
  plot() + geom_vline(xintercept = 0) +
  theme_classic()
```




```
contrast(fit2_emmeans, method = "pairwise", adjust = "tukey") |> confint()
```

```
 contrast estimate    SE df lower.CL upper.CL
 E1 - E2   -0.4932 0.251 60  -1.1988    0.212
 E1 - E3   -0.2765 0.251 60  -0.9822    0.429
 E1 - E4    0.1925 0.251 60  -0.5131    0.898
 E1 - E5    0.0915 0.251 60  -0.6141    0.797
 E2 - E3    0.2167 0.251 60  -0.4889    0.922
 E2 - E4    0.6857 0.251 60  -0.0199    1.391
 E2 - E5    0.5847 0.251 60  -0.1209    1.290
 E3 - E4    0.4690 0.251 60  -0.2366    1.175
 E3 - E5    0.3681 0.251 60  -0.3376    1.074
 E4 - E5   -0.1010 0.251 60  -0.8066    0.605
```

```
Results are averaged over the levels of: Subject
Confidence level used: 0.95
Conf-level adjustment: tukey method for comparing a family of 5 estimates
```

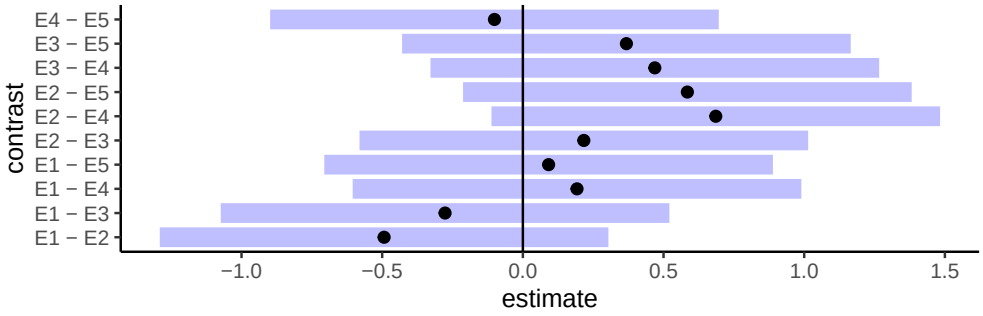
Scheffe's method

Allows for data snooping, can be used with any number and type of contrasts.

```
contrast(fit2_emmeans, method = "pairwise", adjust = "scheffe")
```

```
 contrast estimate    SE df t.ratio p.value
 E1 - E2   -0.4932 0.251 60  -1.966  0.4327
 E1 - E3   -0.2765 0.251 60  -1.102  0.8743
 E1 - E4    0.1925 0.251 60   0.767  0.9636
 E1 - E5    0.0915 0.251 60   0.365  0.9978
 E2 - E3    0.2167 0.251 60   0.864  0.9446
 E2 - E4    0.6857 0.251 60   2.733  0.1279
 E2 - E5    0.5847 0.251 60   2.331  0.2593
 E3 - E4    0.4690 0.251 60   1.869  0.4851
 E3 - E5    0.3681 0.251 60   1.467  0.7083
 E4 - E5   -0.1010 0.251 60  -0.402  0.9968

Results are averaged over the levels of: Subject
P value adjustment: scheffe method with rank 4
```

```
contrast(fit2_emmeans, method = "pairwise", adjust = "scheffe") |>
  plot() + geom_vline(xintercept = 0) +
  theme_classic()
```



Summary: two-way ANOVA normal model

- Apart from the fact that we have a different $\hat{\sigma}$ (and corresponding degrees of freedom), everything is much the same as in the "one-way ANOVA" normal model.
- Once we have "adjusted for blocks" (in order to get a smaller estimate of the error variance), we adjust the degrees of freedom and then proceed as in the one-way case.

Model checking

- It is important to check the assumptions underlying the "additive normal model".
- The main things to check are the **normality** and **constant variance** assumption.
- This is usually done by
  - checking that a boxplot or normal QQ plot of **residuals** "looks normal" (i.e. boxplot: symmetric, not too many outliers; QQ plot: the points are "close" to the diagonal line);
  - plotting **residuals** against **fitted values** to check the common variance assumption.

- **Fitted values**: $\hat{y}_{ij} = \hat{\mu} + \hat{\alpha}_i + \hat{\beta}_j$
- **Residuals**: $r_{ij} = y_{ij} - \hat{y}_{ij}$

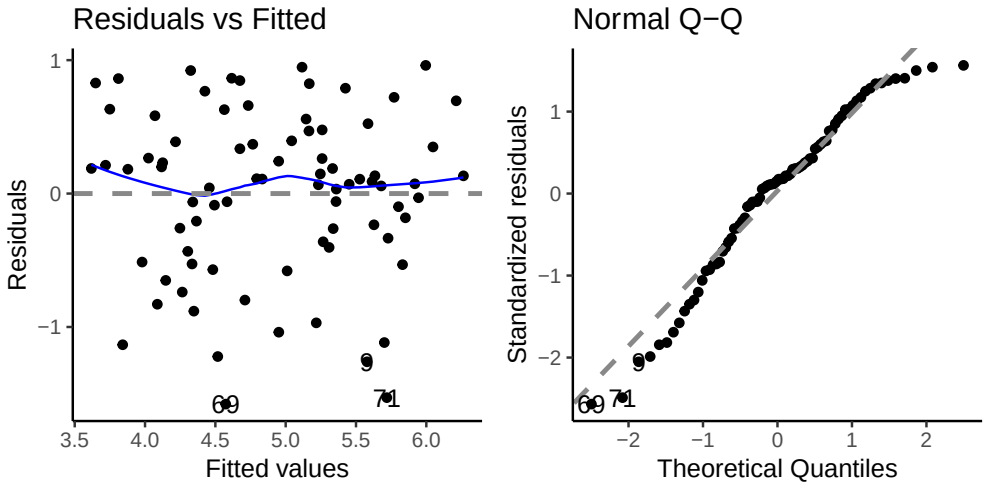
Residual plots

The `fitted.values` and `residuals` can be extracted from the **aov** object:

```
resist_long = resist_long |>
  mutate(fitted = fit2$fitted.values,
         resid = fit2$residuals)
# or using
# broom::augment(fit2)
```

Can use the `autoplot()` function from the **ggfortify** package.

```
autoplot(fit2, which = 1:2, ad.size = 1, colour = "black") +
theme_classic()
```



Possible lack of symmetry?

- The diagnostic plots suggests that *perhaps* the residuals are not symmetrically distributed about zero. [But they're probably fine.]
- This suggests that we might
  - try a different transformation (remember, we transformed the original data!)
  - try using an alternative method that does not require normality assumptions.
- We explore the second option next!

24.5 Adjusting for blocks using ranks

Friedman test

- The χ^2-approximation method:

```
friedman.test(y ~ electrode | Subject, data = resist_long)
```

```
        Friedman rank sum test

data:  y and electrode and Subject
Friedman chi-squared = 5.4522, df = 4, p-value = 0.244
```

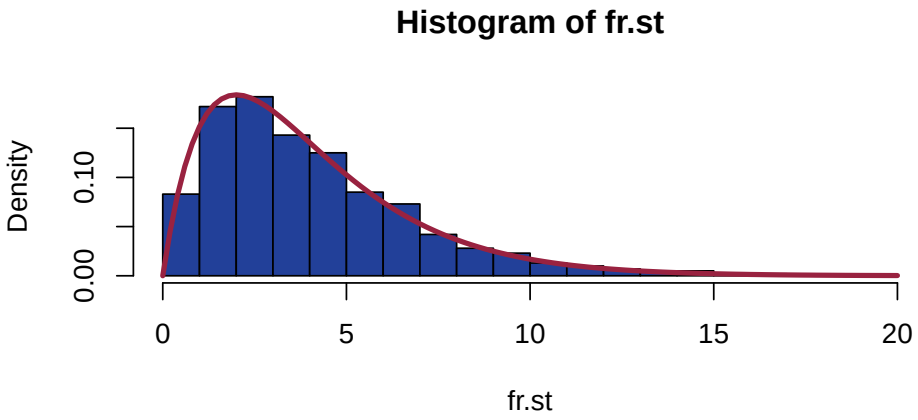
- We can also use a simulation/permutation approach to obtain a p-value:

```
fried.stat = friedman.test(y ~ electrode | Subject, data = resist_long)$
    statistic
B = 1000
fr.st = vector("numeric", length = B)
for(i in 1:B) {
  fr.st[i] = friedman.test(sample(y) ~ electrode | Subject, data =
      resist_long)$statistic
}
mean(fr.st >= fried.stat)
```

```
[1] 0.255
```

Permutation comparison

```
hist(fr.st, breaks = 25, probability = TRUE, col = "#224099")
curve(dchisq(x, 4), col = "#992240", add = TRUE, lwd = 3)
```



25

Two-factor ANOVA with interactions

25.1 Two-factor ANOVA

Multi-factor ANOVA

- In some experiments there is more than one *factor* that might affect the response.
- We can think of such an experiment as a "big one-way ANOVA" where each *combination of factors* is a different "treatment".
- Interest lies in determining not only if there are differences between each treatment, but other questions, such as whether
  - each factor has an effect;
  - the effect of one factor is the same across all levels of the other factor(s).
- We shall focus on the case of **two factors**.

General Setup

We have observations,

$$\{y_{ijk} : \ i = 1, \dots, a; \ j = 1, \dots, b; \ k = 1, \dots, n\},$$

where y_{ijk} is the k-th observation receiving the treatment combination corresponding to level i of factor A and level j of factor B.

- There are n observations receiving each treatment combination.
- The **full model** is that y_{ij} is the value taken by $Y_{ijk} \sim \mathcal{N}(\mu_{ij}, \sigma^2)$.

25.2 Examples

Poisons and antidotes

In a famous paper Box & Cox (1964) analyse an experiment where one of each of 3 poisons and 4 antidotes were administered to a sample of 4 animals, giving 48 observations all together (4 observations on each of the 12 treatment combinations).

- The response was **survival time**.
- They showed the **reciprocal** of the survival time was an appropriate transformation to use on the response.
- The aim of the study was to determine how each `antidote` affected survival in the presence of each `poison`.
- This is a "3-by-4 factorial experiment".

```
data("poison.data", package = "BHH2")
glimpse(poison.data)
```

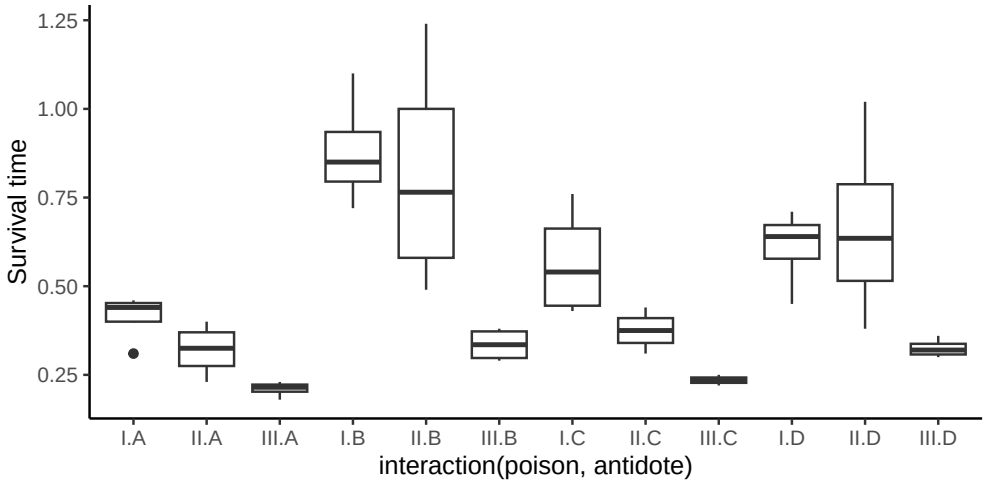
```
Rows: 48
Columns: 3
$ poison <fct> I, I, I, I, II, II, II, II, III, III, III, III, I, I, I, I, II,
    ~
$ treat  <fct> A, A, A, A, A, A, A, A, A, A, A, A, B, B, B, B, B, B, B, B, B,
    ~
$ y      <dbl> 0.31, 0.45, 0.46, 0.43, 0.36, 0.29, 0.40, 0.23, 0.22, 0.21, 0.1
    ~
```

```
# rename treat as antidote to avoid confusion with the general term "treatment
    "
poison.data = poison.data |>
  rename(antidote = treat)
glimpse(poison.data)
```

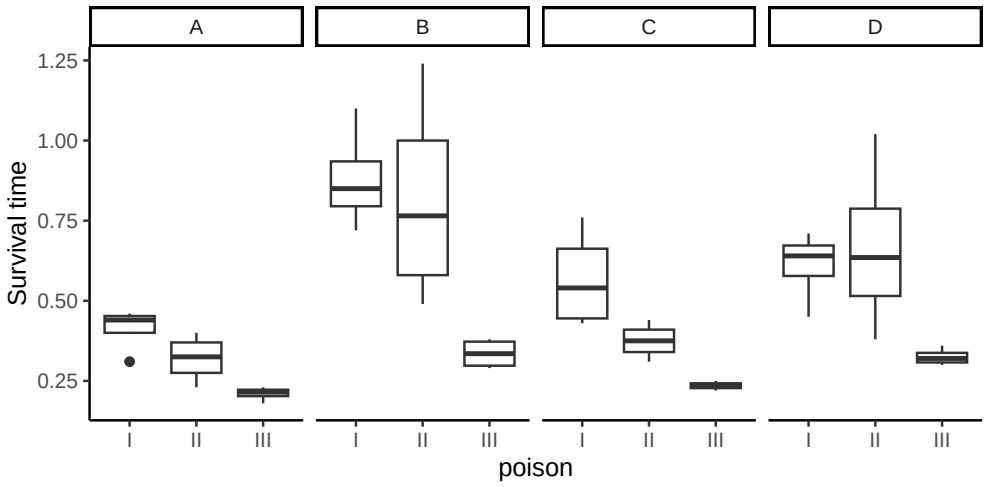
```
Rows: 48
Columns: 3
$ poison   <fct> I, I, I, I, II, II, II, II, III, III, III, III, I, I, I, I, I
    ~
$ antidote <fct> A, A, A, A, A, A, A, A, A, A, A, A, B, B, B, B, B, B, B, B, B
    ~
$ y        <dbl> 0.31, 0.45, 0.46, 0.43, 0.36, 0.29, 0.40, 0.23, 0.22, 0.21, 0
    ~
```

Visualising the data

```
poison.data |> ggplot() +
  aes(x = interaction(poison, antidote), y = y) +
  geom_boxplot() +
  theme_classic() +
  labs(y = "Survival time")
```

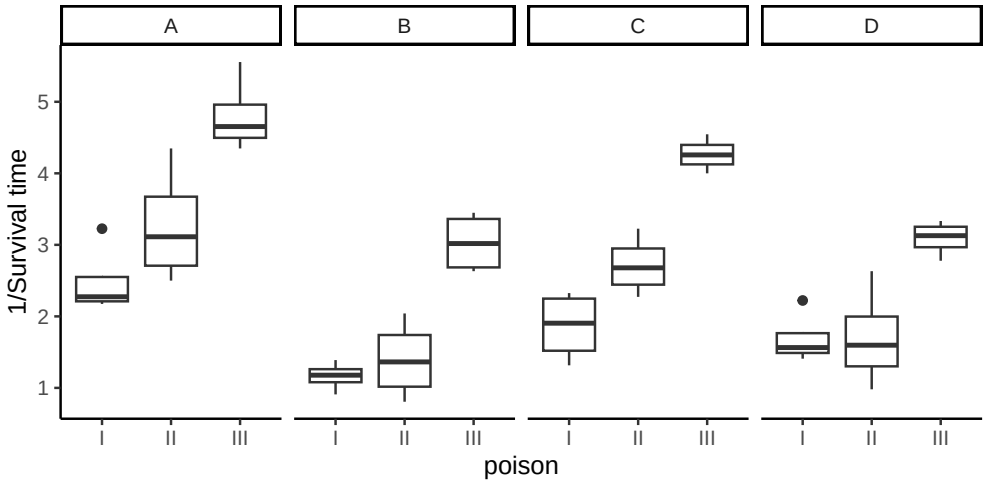



```
poison.data |> ggplot() +
  aes(x = poison, y = y) +
  geom_boxplot() +
  theme_classic() +
  facet_wrap(~antidote, ncol = 4) +
  labs(y = "Survival time")
```



The variance is much less for groups with lower means. A transformation is clearly needed here to satisfy normality, equal variance assumptions.

```
poison.data = poison.data |>
  mutate(inv_survival = 1/y)
poison.data |> ggplot() +
  aes(x = poison, y = inv_survival) +
  geom_boxplot() +
  theme_classic() +
  facet_wrap(~antidote, ncol = 4) +
  labs(y = "1/Survival time")
```



- The reciprocal transformation is much better;
  - while spreads still differ somewhat, they don't get systematically smaller with smaller mean.

Paper planes

Mackisack (1994) has the results from an experiment where statistics students launched paper planes in a controlled environment, controlling for various factors, including

- `Paper` quality: 1 (80gsm) and 2 (50gsm);
- `Plane` design: 1 (High-performance glider) and 2 (Incredibly simple glider).
- The response was distance travelled in mm.
- Four "flights" were conducted at each of the 4 treatment combinations.
- Do either `Paper` or `Plane` (or both) have any impact on the distance "flown"?
- This is a "2-by-2 factorial experiment".

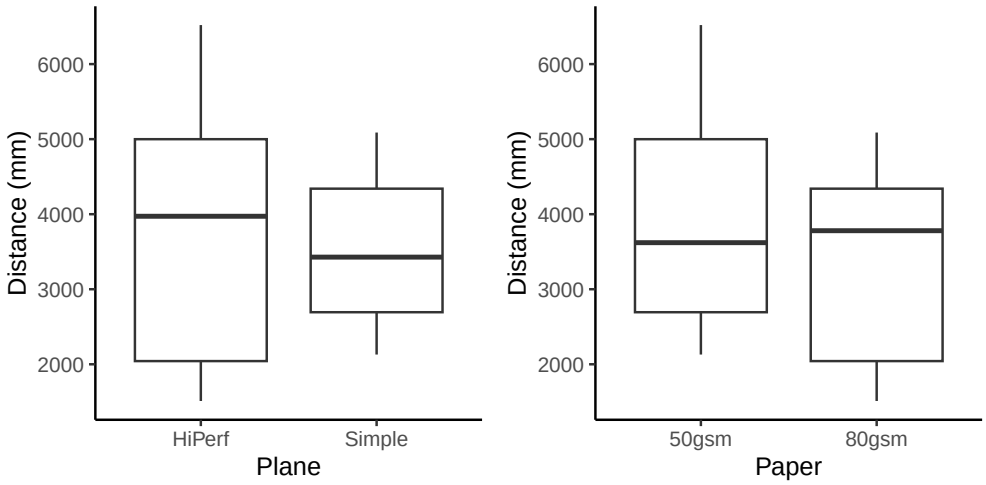
```
planes = read_tsv("https://raw.githubusercontent.com/DATA2002/data/master/
    planes.txt")
glimpse(planes)
```

```
Rows: 16
Columns: 5
$ Distance <dbl> 2160, 1511, 4596, 3706, 3854, 1690, 5088, 4255, 6520, 4091, 2
    ~
$ Paper    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2
$ Angle    <dbl> 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2
$ Plane    <dbl> 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2
$ Order    <dbl> 12, 11, 8, 6, 4, 2, 1, 7, 3, 9, 14, 5, 16, 15, 13, 10
```

```
planes = planes |>
  mutate(
    Paper = case_when(
      Paper == 1 ~ "80gsm",
      Paper == 2 ~ "50gsm"
    ),
    Plane = case_when(
      Plane == 1 ~ "HiPerf",
      Plane == 2 ~ "Simple"
    )
  )
glimpse(planes)
```

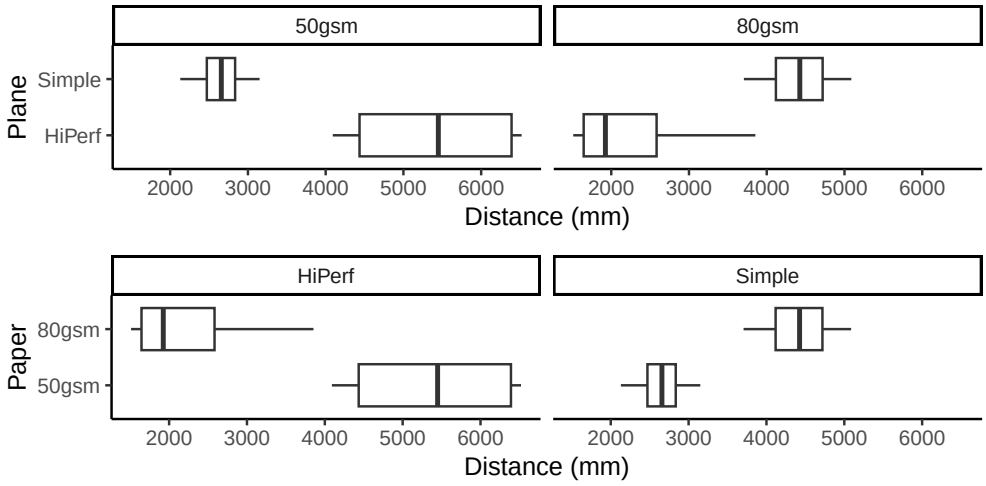
```
Rows: 16
Columns: 5
$ Distance <dbl> 2160, 1511, 4596, 3706, 3854, 1690, 5088, 4255, 6520, 4091, 2
    ~
$ Paper    <chr> "80gsm", "80gsm", "80gsm", "80gsm", "80gsm", "80gsm", "80gsm"
    ~
$ Angle    <dbl> 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2
$ Plane    <chr> "HiPerf", "HiPerf", "Simple", "Simple", "HiPerf", "HiPerf", "
    ~
$ Order    <dbl> 12, 11, 8, 6, 4, 2, 1, 7, 3, 9, 14, 5, 16, 15, 13, 10
```

```
p1 = ggplot(planes, aes(x = Plane, y = Distance)) +
  geom_boxplot() + labs(y = "Distance (mm)") +
  theme_classic()
p2 = ggplot(planes, aes(x = Paper, y = Distance)) +
  geom_boxplot() + labs(y = "Distance (mm)") +
  theme_classic()
gridExtra::grid.arrange(p1, p2, ncol = 2)
```




```
p3 = p1 + facet_wrap(~ Paper) + coord_flip() + theme_classic()
```

```
p4 = p2 + facet_wrap(~ Plane) + coord_flip() + theme_classic()
gridExtra::grid.arrange(p3, p4, ncol = 1)
```



General setup

For each $i = 1, \dots, a,\ j = 1, \dots, b\ k = 1, \dots, n,\ y_{ijk}$ is the k-observation receiving level i of factor A and level j of factor B and is modelled as the value taken by the random variable,

$$Y_{ijk} \sim \mathcal{N}(\mu_{ij}, \sigma^2),$$

and all random variables are assumed independent.

Various "questions"

- We might "ask various questions" about the structure of the μ_{ij}'s (i.e. *test various null hypotheses*), e.g.:
  - $H_0 : \mu_{ij} \equiv \mu$ (i.e. straight 1-way ANOVA between all treatment combinations)
  - $H_0 : \mu_{ij} \equiv \mu_i$ for all j (so factor B has no effect)
  - $H_0 : \mu_{ij} \equiv \mu_j$ for all i (so factor A has no effect)
- We can also ask if an **interaction** is present
  - $H_0 : \mu_{ij} = \mu + \alpha_i + \gamma_j$ for some constants μ, $\alpha_i (i = 1, \dots, a)$, $\gamma_j (j = 1, \dots, b)$, so that the two factors combine **additively**.
  - If this is true then the adjustment for each level of factor A is *the same within each level of factor B*, and vice versa.
  - If this is **not** true, then we say there is some **interaction** between factors A and B.
    - This is (therefore) known as a "test of no interaction".

This is not a two-way ANOVA (in the sense of adjusting for blocks)

- This looks very similar to our earlier model where we were adjusting for blocks.
- However, it is **very different**.
- When adjusting for blocks:
  - the full model is $E(Y_{ij}) = \mu_{ij} = \mu + \alpha_i + \beta_j$ (i.e. treatments and blocks combine additively);
  - the null hypothesis is that $\mu_{ij} = \mu + \beta_j$ (i.e. all "treatment effects" α_i are zero);
  - the β_j's are **not** treatment effects, they are block effects.
- In **this** scenario

- the full model is $E(Y_{ijk}) = \mu_{ij}$ (i.e. unrestricted);
- the null hypothesis is $\mu_{ij} = \mu + \alpha_i + \gamma_j$ (factor A and factor B effects combine additively);
- both α_i and γ_j are related to treatment effects.
- We therefore use γ_j (i.e. a different Greek letter to β_j) to stress this difference.

25.3 Theory

Reparametrisation: contrasts

- We introduce some new "parameters".
- Define
  - $\mu = \overline{\mu}_{\bullet\bullet}$
  - $\alpha_i = \overline{\mu}_{i\bullet} - \overline{\mu}_{\bullet\bullet}$ ("main effect due to i-th level of factor A")
  - $\gamma_j = \overline{\mu}_{\bullet j} - \overline{\mu}_{\bullet\bullet}$ ("main effect due to j-th level of factor B")
  - and the interaction effect between level i of factor A and level j of factor B: $(\alpha\gamma)_{ij} = \mu_{ij} - (\mu + \alpha_i + \gamma_j) = \mu_{ij} - \overline{\mu}_{i\bullet} - \overline{\mu}_{\bullet j} + \overline{\mu}_{\bullet\bullet}$
- Then we may write
$$\mu_{ij} = \mu + \alpha_i + \gamma_j + (\alpha\gamma)_{ij}.$$
- The α_i's, γ_j's and $(\alpha\gamma)_{ij}$'s are all **contrasts** in the μ_{ij}'s!

Main effects: interpretation

- Each α_i is in fact a **contrast** in the (treatment combination) group means that measures in some overall sense how the means for level i of factor A differ from the overall average.
- In exactly the same way, each γ_j is a contrast (in the μ_{ij}'s) that compares (in some overall sense) level j of factor B to the overall average.

Interaction effects

- A similar (but more complicated) calculation can be used to show that each $(\alpha\gamma)_{ij}$ is also a contrast in the μ_{ij}'s.
- Each
$$(\alpha\gamma)_{ij} = \mu_{ij} - (\mu + \alpha_i + \gamma_j)$$
compares a mean μ_{ij} to the corresponding "additive prediction" $\mu + \alpha_i + \gamma_j$.
  - If the factor levels *actually do* combine additively, then each such interaction (population) contrast is zero.
- Therefore a "test for no interaction" can be formulated as the following null hypothesis:
$$H_0 : (\alpha\gamma)_{ij} = 0 \text{ for all } i, j.$$

Constraints: main effects degrees of freedom

- Note that (as with our earlier models) these "effects" satisfy certain constraints:
- Both sets of main effects add to zero:
$$\sum_{i=1}^{a} \alpha_i = \sum_{j=1}^{b} \gamma_j = 0.$$
- There are thus
  - $a - 1$ "free" α_i's and
  - $b - 1$ "free" γ_j's and

- That is to say, there are
  - $a - 1$ degrees of freedom for the factor A main effects;
  - $b - 1$ degrees of freedom for the factor B main effects;

Constraints: interaction effects degrees of freedom

- The interaction terms are such that for each fixed i and j

$$\sum_{i=1}^{a}(\alpha\gamma)_{ij} = \sum_{j=1}^{b}(\alpha\gamma)_{ij} = 0.$$

- There are thus $(a-1)(b-1)$ "free" interaction effects (like in an a by b two-way contingency table where all row and column sums are fixed).
- That is to say, there are $(a-1)(b-1)$ degrees of freedom for the interaction effects.

Fitted values and residuals for each model

- We thus have two models:
  - the full model where μ_{ij} is "unrestricted";
  - the ("no interaction") null hypothesis where $\mu_{ij} = \mu + \alpha_i + \gamma_j$
- Under each model, each observation y_{ijk} is decomposed into two parts:
  - a fitted value $\hat{\mu}_{ij}$;
  - a residual $y_{ijk} - \hat{\mu}_{ij}$.
- Under the full model $\hat{\mu}_{ij} = \overline{y}_{ij\bullet}$, the mean of the (i,j)th **treatment combination**.
- Under the "no interaction" model, the fitted value is

$$\begin{aligned}\hat{\mu}_{ij} &= \hat{\mu} + \hat{\alpha}_i + \hat{\gamma}_j \\ &= \overline{y}_{\bullet\bullet\bullet} + (\overline{y}_{i\bullet\bullet} - \overline{y}_{\bullet\bullet\bullet}) + (\overline{y}_{\bullet j\bullet} - \overline{y}_{\bullet\bullet\bullet}) \\ &= \overline{y}_{i\bullet\bullet} + \overline{y}_{\bullet j\bullet} - \overline{y}_{\bullet\bullet\bullet}\end{aligned}$$

Residual sum of squares for each model

- For the **full model** the (i,j,k)-th residual is $y_{ijk} = \overline{y}_{ij\bullet}$ and the **residual sum of squares** is

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(y_{ijk} - \overline{y}_{ij\bullet}\right)^2.$$

- For the **no interaction model** the (i,j,k)-th residual is $y_{ijk} - \overline{y}_{i\bullet\bullet} - \overline{y}_{\bullet j\bullet} + \overline{y}_{\bullet\bullet\bullet}$ and the **residual sum of squares** is

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(y_{ijk} - \overline{y}_{i\bullet\bullet} - \overline{y}_{\bullet j\bullet} + \overline{y}_{\bullet\bullet\bullet}\right)^2.$$

$$
\begin{aligned}
\text{Total SS} &= \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\overline{y}_{\bullet\bullet\bullet})^2 \\
&= \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left[(\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet\bullet\bullet})+(\overline{y}_{\bullet j\bullet}-\overline{y}_{\bullet\bullet\bullet})+(y_{ijk}-\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet j\bullet}+\overline{y}_{\bullet\bullet\bullet})\right]^2 \\
&= \underbrace{\sum_{i=1}^{a} bn(\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet\bullet\bullet})^2}_{\text{Factor A Sum Sq.}} + \underbrace{\sum_{j=1}^{b} an(\overline{y}_{\bullet j\bullet}-\overline{y}_{\bullet\bullet\bullet})^2}_{\text{Factor B Sum Sq.}} + \underbrace{\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet j\bullet}+\overline{y}_{\bullet\bullet\bullet})^2}_{\text{no interaction model Res Sum Sq.}} \\
&\quad + \text{ cross-product terms (all 0)} \\
&= \sum_{i=1}^{a} bn(\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet\bullet\bullet})^2 + \sum_{j=1}^{b} an(\overline{y}_{\bullet j\bullet}-\overline{y}_{\bullet\bullet\bullet})^2 \\
&\quad + \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left[(\overline{y}_{ij\bullet}-\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet j\bullet}+\overline{y}_{\bullet\bullet\bullet})+(y_{ijk}-\overline{y}_{ij\bullet})\right]^2 \\
&= \sum_{i=1}^{a} bn(\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet\bullet\bullet})^2 + \sum_{j=1}^{b} an(\overline{y}_{\bullet j\bullet}-\overline{y}_{\bullet\bullet\bullet})^2 + \underbrace{\sum_{i=1}^{a}\sum_{j=1}^{b} n(\overline{y}_{ij\bullet}-\overline{y}_{i\bullet\bullet}-\overline{y}_{\bullet j\bullet}+\overline{y}_{\bullet\bullet\bullet})^2}_{\text{Interaction Sum Sq.}} \\
&\quad + \underbrace{\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\overline{y}_{ij\bullet})^2}_{\text{full model Res Sum Sq.}} + \text{a cross-product term which is zero.}
\end{aligned}
$$

F-test for no interaction

- Under the full model, the (random variable version of the) residual sum of squares

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(Y_{ijk}-\overline{Y}_{ij\bullet})^2 \sim \sigma^2\chi^2_{ab(n-1)}\,,$$

  since
  - the total sample size is $N = abn$;
  - the number of groups is $g = ab$ so
  - the degrees of freedom for residuals is $N - g = anb - ab = ab(n-1)$.
- It turns out that *under the no interaction model* (the null hypothesis) the (random variable version of the) interaction sum of squares

$$\sum_{i=1}^{a}\sum_{j=1}^{b} n(\overline{Y}_{ij\bullet}-\overline{Y}_{i\bullet\bullet}-\overline{Y}_{\bullet j\bullet}+\overline{Y}_{\bullet\bullet\bullet})^2 \sim \sigma^2\chi^2_{(a-1)(b-1)}$$

  (we gave already established there are $(a-1)(b-1)$ df for interactions).
- The F-statistic for testing the null hypothesis of no interactions is

$$\frac{\text{Interaction Mean Square}}{\text{(full model) Residual MeanSquare}} = \frac{\text{(Int. SS)}/[(a-1)(b-1)]}{\text{(full model Residual SS)}/[ab(n-1)]} \sim F_{(a-1)(b-1),ab(n-1)}$$

25.4 Back to our examples

Poisons and antidotes

Recall we have two factors, `poison` with 3 levels:

```
levels(poison.data$poison)
```

```
[1] "I"   "II"  "III"
```

and `antidote` with 4 levels:

```
levels(poison.data$antidote)
```

```
[1] "A" "B" "C" "D"
```

`poison:antidote`

- Given the two separate factors `poison` and `antidote`, the R object `poison:antidote` is a new factor which has as its levels every possible `poison:antidote` combination:

```
poison.data$poison:poison.data$antidote
```

```
 [1] I:A   I:A   I:A   I:A   II:A  II:A  II:A  II:A  III:A III:A III:A
     III:A
[13] I:B   I:B   I:B   I:B   II:B  II:B  II:B  II:B  III:B III:B III:B
    III:B
[25] I:C   I:C   I:C   I:C   II:C  II:C  II:C  II:C  III:C III:C III:C
    III:C
[37] I:D   I:D   I:D   I:D   II:D  II:D  II:D  II:D  III:D III:D III:D
    III:D
Levels: I:A I:B I:C I:D II:A II:B II:C II:D III:A III:B III:C III:D
```

- This factor indicates the 12 groups in the "big 1-way ANOVA" where each `poison:antidote` combination is a different treatment;
  - there are 4 observations on each "treatment".

```
poison.data |>
  group_by(poison, antidote) |>
  count()
```

```
# A tibble: 12 x 3
# Groups:   poison, antidote [12]
   poison antidote     n
   <fct>  <fct>    <int>
 1 I      A            4
 2 I      B            4
 3 I      C            4
 4 I      D            4
 5 II     A            4
 6 II     B            4
 7 II     C            4
 8 II     D            4
 9 III    A            4
10 III    B            4
11 III    C            4
12 III    D            4
```

- We could fit the full model (the "big 1-way ANOVA"):

```
a1 = aov(inv_survival ~ poison:antidote, data = poison.data)
anova(a1)
```

```
Analysis of Variance Table

Response: inv_survival
               Df Sum Sq Mean Sq F value    Pr(>F)
poison:antidote 11 56.862  5.1693  21.531 1.289e-12 ***
Residuals       36  8.643  0.2401
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- There is a clear treatment effect
- Note there are 47 df in total:
  - 11 for treatments
  - 36 for residuals

A better approach

- A full, appropriate two-factor ANOVA table can be produced by using an appropriate formula:
  - fit `poison:antidote` *after* fitting the main effects:

```
a3 = aov(inv_survival ~ poison + antidote + poison:antidote,
         data = poison.data)
summary(a3)
```

```
               Df Sum Sq Mean Sq F value   Pr(>F)
poison          2  34.88  17.439   72.64 2.31e-13 ***
antidote        3  20.41   6.805   28.34 1.38e-09 ***
poison:antidote 6   1.57   0.262    1.09    0.387
Residuals       36  8.64   0.240
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The sum of squares for `poison:antidote` here only contains the variation explained by treatments *not already explained by main effects*.
- This provides a convenient *partition* of the "big 1-way ANOVA" **treatment sum of squares** (11 df in total) into 3 contributions:
  - 2 df for `poison` main effect;
  - 3 df for `antidote` (antidote) main effect;
  - 6 df for the interaction effect.

Alternate formula for two-factor ANOVA

- An *equivalent* formula which includes main effects followed by interactions is given as follows:

```
summary(aov(inv_survival ~ poison * antidote, data = poison.data))
```

```
               Df Sum Sq Mean Sq F value   Pr(>F)
poison          2  34.88  17.439   72.64 2.31e-13 ***
antidote        3  20.41   6.805   28.34 1.38e-09 ***
poison:antidote 6   1.57   0.262    1.09    0.387
Residuals       36  8.64   0.240
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- This is what most people use, although the equivalent (but longer) form used initially (i.e. `poison+antidote+poison:antidote`) is perhaps more useful for beginners.
- The `poison` and `antidote` main effects **are significant** (i.e. there is a significant difference between the levels of each treatment group).

Paper planes

- Let us perform a similar analysis on the Paper Planes data.
- First we check the "big 1-way ANOVA":

```
summary(aov(Distance ~ Paper:Plane, data = planes))
```

```
            Df   Sum Sq Mean Sq F value  Pr(>F)
Paper:Plane  3 25491258 8497086   10.66 0.00106 **
Residuals   12  9561029  796752
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

  There is a strong Treatment effect.
- Now we partition the **treatment sum of squares** into main effects and interactions:

```
plane_aov = aov(Distance ~ Paper * Plane, data = planes)
summary(plane_aov)
```

```
            Df   Sum Sq  Mean Sq F value   Pr(>F)
Paper        1  1718721  1718721   2.157 0.167628
Plane        1   385641   385641   0.484 0.499861
Paper:Plane  1 23386896 23386896  29.353 0.000156 ***
Residuals   12  9561029   796752
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- There is a *strong* interaction effect.
- Note the large p-values for the main effects.
  - Are they significant? No.
  - Should they be dropped from the model: **absolutely not**!
- If an interaction is significant, we must retain the **full model**.

25.5 Interaction plots

Interaction plots

- We can graphically examine interactions using **interaction plots**.
- These involve choosing one of the factors as the `x.factor` and one as the `trace.factor`:
  - levels of the `x.factor` are marked, equally spaced, on the "x-axis";
  - then a *trace* of each level of the (other) `trace.factor` is created by
    - plotting mean responses for that level against the corresponding level on the x-axis;
    - joining the mean responses within in each level by a piecewise linear curve.
- If there is **no interaction** the traces should be "roughly parallel".
- If there **is** an interaction, the traces may cross or deviate from parallelism in some other way.

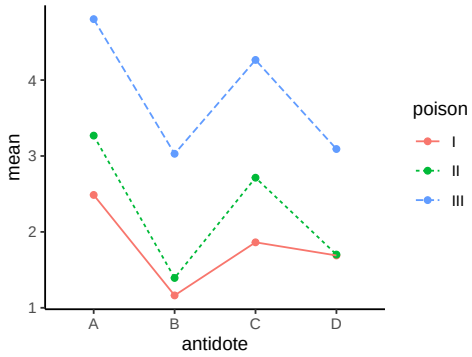
Poisons/antidotes interaction plots

The mean responses for each treatment combination:

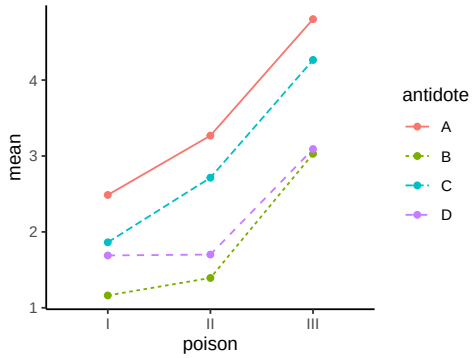
```
sum_dat = poison.data |> group_by(poison, antidote) |>
  summarise(mean = mean(inv_survival),
            n = n())
sum_dat
```

```
# A tibble: 12 x 4
# Groups:   poison [3]
   poison antidote  mean     n
   <fct>  <fct>    <dbl> <int>
 1 I      A         2.49     4
 2 I      B         1.16     4
 3 I      C         1.86     4
 4 I      D         1.69     4
 5 II     A         3.27     4
 6 II     B         1.39     4
 7 II     C         2.71     4
 8 II     D         1.70     4
 9 III    A         4.80     4
10 III    B         3.03     4
11 III    C         4.26     4
12 III    D         3.09     4
```

```
sum_dat |>
  ggplot(aes(x = antidote, y = mean,
             group = poison,
             linetype = poison,
             colour = poison)) +
  geom_point() +
  geom_line() +
  theme_classic()
```



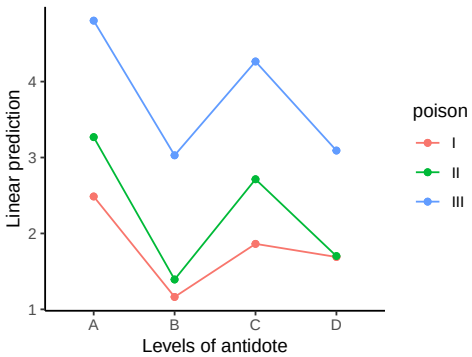

```
sum_dat |>
  ggplot(aes(x = poison, y = mean,
             group = antidote,
             linetype = antidote,
             colour = antidote)) +
  geom_point() +
  geom_line() +
  theme_classic()
```



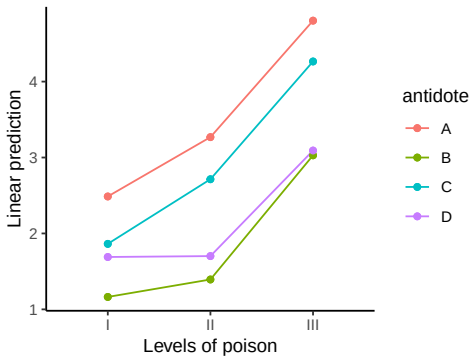
Using emmeans

The `emmip()` function from the **emmeans** package can do this for us:

```
emmip(a1, poison ~ antidote) +
  theme_classic()
```




```
emmip(a1, antidote ~ poison) +
  theme_classic()
```



Paper planes interaction plots

- Recall that there was a strong interaction effect.
- What would be expect to see here?

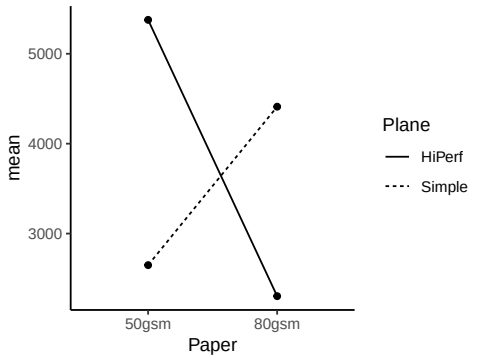
```
plane_sum_dat = planes |>
  group_by(Plane, Paper) |>
  summarise(mean = mean(Distance))
plane_sum_dat
```

```
# A tibble: 4 x 3
# Groups:   Plane [2]
  Plane  Paper  mean
  <chr>  <chr> <dbl>
1 HiPerf 50gsm 5377.
2 HiPerf 80gsm 2304.
3 Simple 50gsm 2649.
4 Simple 80gsm 4411.
```

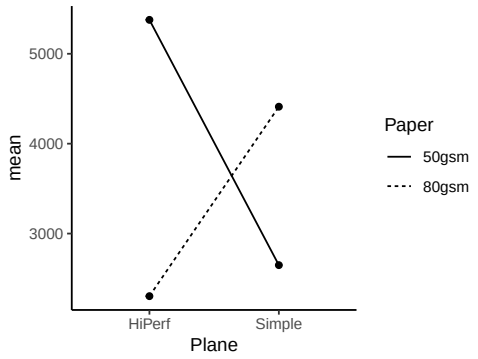
```
plane_sum_dat |>
  spread(key = Plane, value = mean)
```

```
# A tibble: 2 x 3
  Paper HiPerf Simple
  <chr>  <dbl>  <dbl>
1 50gsm  5377.  2649.
2 80gsm  2304.  4411.
```

```
plane_sum_dat |>
  ggplot(aes(x = Paper, y = mean,
             group = Plane,
             linetype = Plane)) +
  geom_point() +
  geom_line() +
  theme_classic()
```



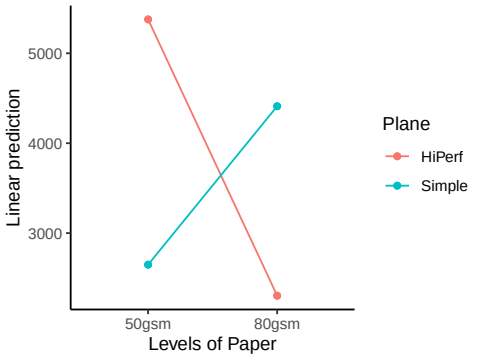

```
plane_sum_dat |>
  ggplot(aes(x = Plane, y = mean,
             group = Paper,
             linetype = Paper)) +
  geom_point() +
  geom_line() +
  theme_classic()
```



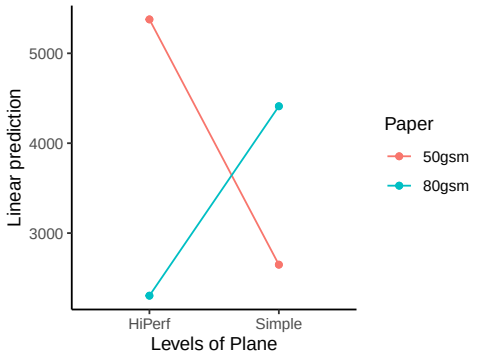
Using emmeans

The `emmip()` function from the **emmeans** package can do this for us:

```
emmip(plane_aov, Plane ~ Paper) +
  theme_classic()
```




```
emmip(plane_aov, Paper ~ Plane) +
  theme_classic()
```



Crossing traces: interaction

- The traces cross dramatically, highlighting the **strong interaction effect**.
- What is going on here?
  - The high performance design flies further with lighter paper;
  - the simple design flies further with heavier paper.

25.6 Post hoc tests

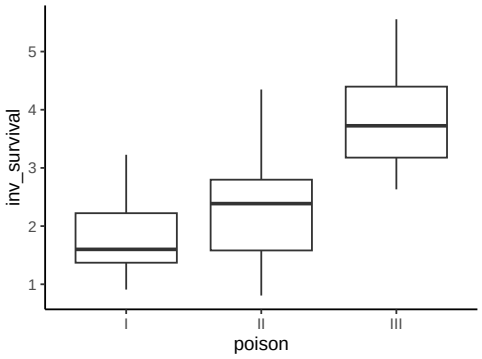
Post hoc comparisons

- As with any analysis of variance, once we establish that there is something going on, we would like to "zoom in" and learn more about what is leading to the significance.
- **However** in this two-factor scenario, exactly which comparisons one might be interested in *may depend on whether there was a significant interaction or not*.
- If there **is a significant interaction**, we might like to know if one factor has a strong effect of the *other* factor within some or all of its levels.
- If there is **no significant interaction**, the levels of factor A can be compared "independently" of levels of factor B. We may wish to determine:
  - which levels of factor A are significantly different from one another, or
  - which levels of factor B are significantly different from one another.
- The main thing to realise is that any within-factor comparison is, at the end of the day, just a *contrast* in the original μ_{ij}'s and so an appropriate multiple comparisons adjustment can be made accordingly.

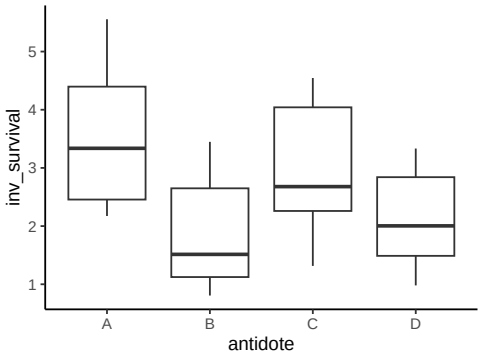
Comparing poisons

- We determined that there is no interaction between `poison` and `antidote`, so we can perhaps go ahead and compare different `poison` (to each other) and also compare antidote treatments (`antidote`) (to each other).

```
poison.data |>
  ggplot(aes(x = poison, y =
      inv_survival)) +
  geom_boxplot() +
  theme_classic()
```




```
poison.data |>
  ggplot(aes(x = antidote, y =
      inv_survival)) +
  geom_boxplot() +
  theme_classic()
```



Pairwise difference t-statistics

```
a2 = aov(inv_survival ~ poison + antidote, data = poison.data)
emmeans(a2, ~ poison) |> contrast(method = "pairwise", adjust = "none")
```

```
 contrast estimate    SE df t.ratio p.value
 I - II     -0.469 0.174 42  -2.688  0.0103
 I - III    -1.996 0.174 42 -11.451  <.0001
 II - III   -1.528 0.174 42  -8.763  <.0001

Results are averaged over the levels of: antidote
```

```
emmeans(a2, ~ antidote) |> contrast(method = "pairwise", adjust = "none")
```

```
 contrast estimate    SE df t.ratio p.value
 A - B       1.657 0.201 42   8.233  <.0001
 A - C       0.572 0.201 42   2.842  0.0069
 A - D       1.358 0.201 42   6.747  <.0001
 B - C      -1.085 0.201 42  -5.391  <.0001
 B - D      -0.299 0.201 42  -1.485  0.1449
 C - D       0.786 0.201 42   3.905  0.0003

Results are averaged over the levels of: poison
```

Assessing significance

- We have a few options, when it comes to comparing these:
  - use a Bonferroni approach;
  - Tukey's method;
  - Scheffe's method.
- There may be some doubt as to the validity of the first two, depending on whether the decision of **which** comparisons to look at came **after the initial hypothesis test** or not.
- **If** the 3+6=9 within-factor comparisons were planned *before looking at the data* then a Bonferroni adjustment could be made as follows
- multiply all *unadjusted* two-sided t-test p-values by 9.

```
p_emm = contrast(emmeans(a2, ~poison), method = "pairwise", adjust = "
    bonferroni")
p_emm
```

```
 contrast estimate    SE df t.ratio p.value
 I - II     -0.469 0.174 42  -2.688  0.0308
 I - III    -1.996 0.174 42 -11.451  <.0001
 II - III   -1.528 0.174 42  -8.763  <.0001

Results are averaged over the levels of: antidote
P value adjustment: bonferroni method for 3 tests
```

```
a_emm = contrast(emmeans(a2, ~antidote), method = "pairwise", adjust = "
    bonferroni")
a_emm
```

```
 contrast estimate    SE df t.ratio p.value
 A - B       1.657 0.201 42   8.233  <.0001
 A - C       0.572 0.201 42   2.842  0.0414
 A - D       1.358 0.201 42   6.747  <.0001
 B - C      -1.085 0.201 42  -5.391  <.0001
 B - D      -0.299 0.201 42  -1.485  0.8693
 C - D       0.786 0.201 42   3.905  0.0020

Results are averaged over the levels of: poison
P value adjustment: bonferroni method for 6 tests
```

Bonferroni method

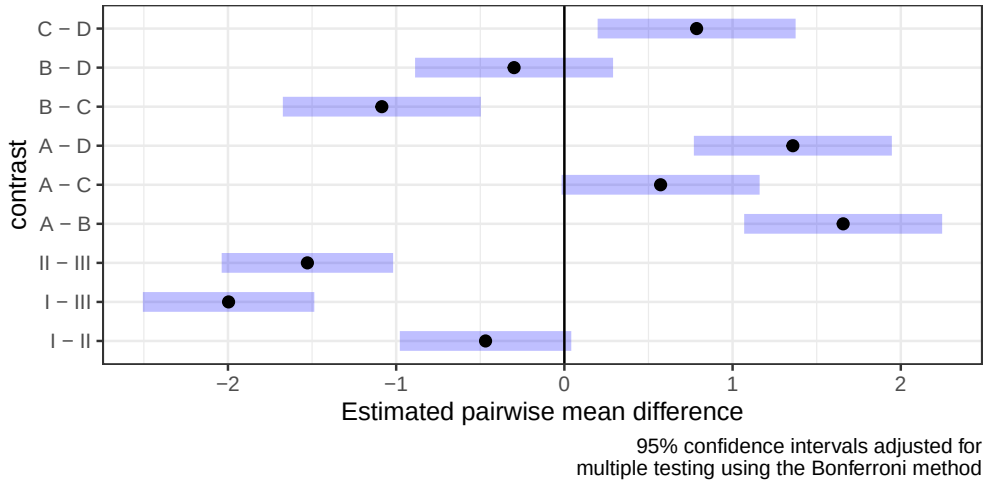
The **emmeans** package didn't know we were looking at **9** tests. But we can tell it that we are:

```
pa_emm = update(p_emm + a_emm)
pa_emm
```

```
 contrast estimate    SE df t.ratio p.value
 I - II     -0.469 0.174 42  -2.688  0.0924
 I - III    -1.996 0.174 42 -11.451  <.0001
 II - III   -1.528 0.174 42  -8.763  <.0001
 A - B       1.657 0.201 42   8.233  <.0001
 A - C       0.572 0.201 42   2.842  0.0620
 A - D       1.358 0.201 42   6.747  <.0001
 B - C      -1.085 0.201 42  -5.391  <.0001
 B - D      -0.299 0.201 42  -1.485  1.0000
 C - D       0.786 0.201 42   3.905  0.0030

Results are averaged over some or all of the levels of: antidote, poison
P value adjustment: bonferroni method for 9 tests
```

- For `poison`: the `I-III` and `II-III` differences are highly significant, while the `I-II` has an adjusted p-value of about 0.09 (not really significant).
- For `antidote`: all differences are highly significant except `B-D` (not at all significant) and `A-C` (p=0.06).

```
plot(pa_emm) + theme_bw() + geom_vline(xintercept = 0) +
  labs(x = "Estimated pairwise mean difference",
       caption = "95% confidence intervals adjusted for\nmultiple testing
           using the Bonferroni method")
```



Paper planes

- For this case, where the full model must be retained, we really do have a "big 1-way ANOVA".
- In such a case, all pairwise comparisons between all treatment combinations, or some other set of comparisons, may be of interest. For example, we can use Tukey's all pairwise comparisons:

```
plane_aov = aov(Distance ~ Plane*Paper, data = planes)
plane_emm = emmeans(plane_aov,  ~ Paper + Plane)
contrast(plane_emm, method = "pairwise", adjust = "tukey")
```

```
 contrast                    estimate  SE df t.ratio p.value
 50gsm HiPerf - 80gsm HiPerf     3074 631 12   4.870  0.0019
 50gsm HiPerf - 50gsm Simple     2728 631 12   4.323  0.0047
 50gsm HiPerf - 80gsm Simple      966 631 12   1.530  0.4508
 80gsm HiPerf - 50gsm Simple     -345 631 12  -0.547  0.9457
 80gsm HiPerf - 80gsm Simple    -2108 631 12  -3.339  0.0262
 50gsm Simple - 80gsm Simple    -1762 631 12  -2.792  0.0677

P value adjustment: tukey method for comparing a family of 4 estimates
```

- Note that all *within-level* comparisons are quite significant.

26
Regression

26.1 Regression

Air pollution

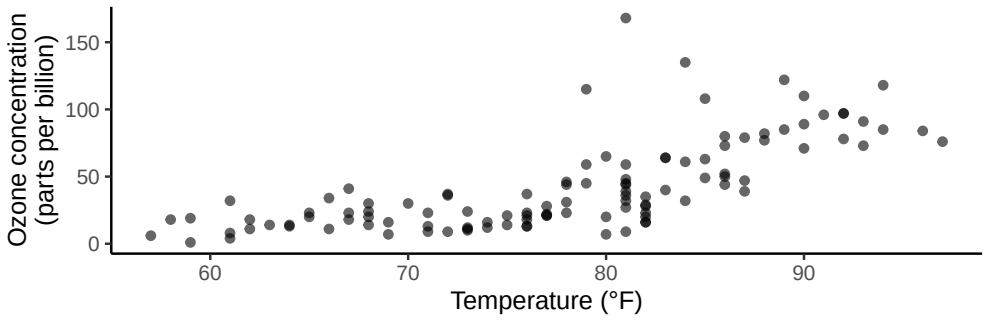
The data frame `environmental` has four environmental variables `ozone`, `radiation`, `temperature` and `wind` taken in New York City from May to September of 1973.

```
data("environmental", package = "lattice")
glimpse(environmental)
```

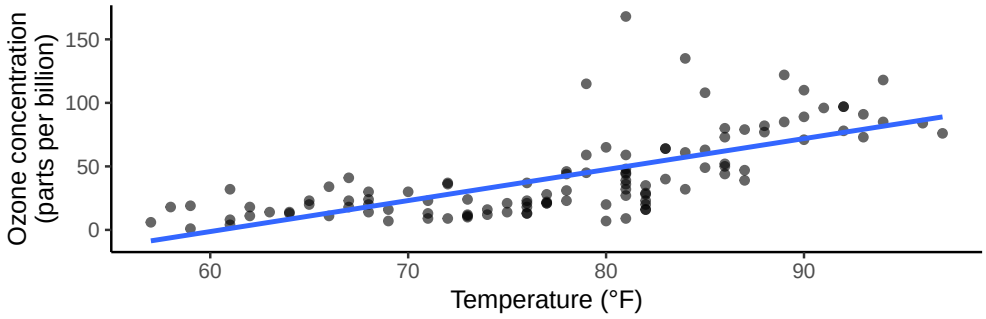
```
Rows: 111
Columns: 4
$ ozone       <dbl> 41, 36, 12, 18, 23, 19, 8, 16, 11, 14, 18, 14, 34, 6, 30,
    ~
$ radiation   <dbl> 190, 118, 149, 313, 299, 99, 19, 256, 290, 274, 65, 334, 3
    ~
$ temperature <dbl> 67, 72, 74, 62, 65, 59, 61, 69, 66, 68, 58, 64, 66, 57, 68
    ~
$ wind        <dbl> 7.4, 8.0, 12.6, 11.5, 8.6, 13.8, 20.1, 9.7, 9.2, 10.9, 13.
    ~
```

We'd like to assess whether the maximum daily temperature has an influence on average ozone concentration.

```
p = ggplot(environmental) + aes(x = temperature, y = ozone) +
  geom_point(alpha = 0.6) +
  labs(x = "Temperature (°F)", y = "Ozone concentration\n(parts per billion)")
       +
  theme_classic()
p
```




```
p + geom_smooth(method = "lm", se = FALSE)
```



Simple linear regression

A **simple linear regression** model aims to predict an outcome variable, Y, using a single predictor variable x,

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

for $i = 1, 2, \ldots, n$ whre n is the number of observations (rows) in the data set.
This is just the equation of a straight line (like $y = mx + b$) plus some additional variation,

- β_0 is the population intercept parameter
- β_1 is the population slope parameter
- ε is the error term and typically assumed to follow $\mathcal{N}(0, \sigma^2)$

Hence,

$$Y_i \sim \mathcal{N}(\beta_0 + \beta_1 x_i,\ \sigma^2).$$

Fitting a straight line by least squares

How to estimate β_0 and β_1? We aim to **minimise the sum of squared residuals**.

Definition

Let

$$r_i = y_i - \hat{y}_i$$

where $\hat{y}_i$ is the fitted value, the value we predict for the ith observation given the ith predictor value:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- The estimated intercept ($\hat{\beta}_0$) and estimated slope ($\hat{\beta}_1$) are found by solving the

following optimisation problem:

$$\underset{\beta_0,\beta_1}{\text{argmin}} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2.$$

- Closed form solutions exist for $\hat{\beta}_0$ and $\hat{\beta}_1$
- R does this for us with the `lm()` function (short for linear model).

Air pollution

```
lm1 = lm(ozone ~ temperature,
         data = environmental)
lm1
```

```
Call:
lm(formula = ozone ~ temperature, data = environmental)

Coefficients:
(Intercept)  temperature
   -147.646        2.439
```

Our estimated model is:

$$\widehat{\text{ozone}} = -147.646 + 2.439 \times \text{temperature}$$

Fitted values and residuals

The fitted values ($\hat{y}$) are obtained by plugging the observed predictor (x) values into our estinated model, $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$.

```
environmental = environmental |>
  mutate(
    fitted = -147.646 + 2.439 * temperature
  )
```

The residuals are the differences between the observed outcome variable (y) and the value the estimated model predicts for that observation (the fitted value $\hat{y}$),

$$r_i = y_i - \hat{y}_i.$$

```
environmental = environmental |>
  mutate(
    resid = ozone - fitted
    )
```

An easier alternative is to extract the residuals and fitted values from the lm1 object directly:

```
environmental = environmental |>
  mutate(
    resid = lm1$residuals,
    fitted = lm1$fitted.values
  )
```

Alternatively we could have used the `augment()` function from the **broom** package to do this:

```
broom::augment(lm1) |> glimpse()
```

```
Rows: 111
Columns: 8
$ ozone       <dbl> 41, 36, 12, 18, 23, 19, 8, 16, 11, 14, 18, 14, 34, 6, 30,
    ~
```

```
$ temperature <dbl> 67, 72, 74, 62, 65, 59, 61, 69, 66, 68, 58, 64, 66, 57, 68
    ~
$ .fitted     <dbl> 15.774291, 27.969841, 32.848061, 3.578742, 10.896071, -3.7
    ~
$ .resid      <dbl> 25.2257087, 8.0301592, -20.8480606, 14.4212582, 12.1039285
    ~
$ .hat        <dbl> 0.020668833, 0.012367934, 0.010448943, 0.033974620, 0.0253
    ~
$ .sigma      <dbl> 23.90523, 24.01815, 23.94597, 23.98922, 24.00176, 23.92627
    ~
$ .cooksd     <dbl> 1.198346e-02, 7.144860e-04, 4.052900e-03, 6.616411e-03, 3.
    ~
$ .std.resid  <dbl> 1.06564582, 0.33780095, -0.87615482, 0.61339917, 0.5125607
    ~
```

The `lm` object

What other hidden treasures does the `lm1` object hold?

```
names(lm1)
```

```
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

E.g. we can extract the coefficients:

```
lm1$coefficients
```

```
(Intercept) temperature
 -147.64607     2.43911
```

Or we can use the `tidy()` function from the **broom** package to do this:

```
lm1 |> broom::tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -148.      18.8       -7.87 2.76e-12
2 temperature     2.44     0.239     10.2  1.55e-17
```

26.2 Checking assumptions

Linear regression assumptions

There are 4 assumptions underling our linear regression model:

1. **Linearity** - the relationship between Y and x is linear
2. **Independence** - all the errors are independent of each other
3. **Homoskedasticity** - the errors have constant variance $\text{Var}(v\varepsilon)_i = \sigma^2$ for all $i = 1, 2, \ldots, n$
4. **Normality** - the errors follow a normal distribution

The last three can be written succinctly as ε_i iid $\mathcal{N}(0, \sigma^2)$.

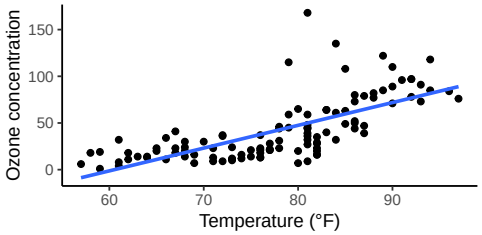
Assumption 1: linearity

> Violations to the linearity assumption are very serious, it means your predictions are likely to be **systematically wrong**

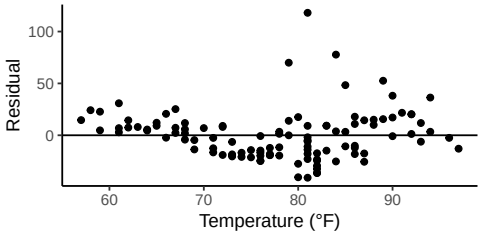
Checking for linearity

1. Before running the regression: plot y against x and look to see if the relationship is approximately linear
2. After running the regression: look at a plot of the residuals against x
    - Residuals should be symmetrically distributed above and below zero
    - A curved pattern in the residuals is evidence for non-linearity, i.e. for some values of x the model regularly overestimates y while in other regions the model regularly underestimates y

```
p1 = environmental |> ggplot() +
  aes(x = temperature, y = ozone) +
  geom_point() +
  labs(x = "Temperature (°F)",
       y = "Ozone concentration") +
  geom_smooth(method="lm", se=FALSE) +
  theme_classic()
p1
```

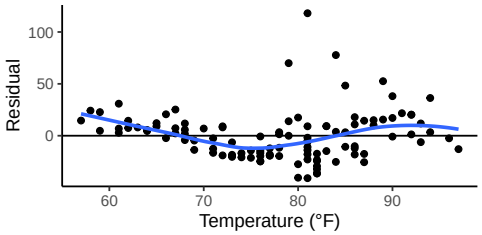



```
p2 = environmental |> ggplot() +
  aes(x = temperature, y = resid) +
  geom_point() +
  labs(x = "Temperature (°F)",
       y = "Residual") +
  geom_hline(yintercept = 0) +
  theme_classic()
p2
```



In the plot below the residuals are above zero for low temperatures, then they go below zero and end up again above zero for high temperatures (as highlighted by the local smoothing curve).

```
p2 + geom_smooth(method = "loess",
                 se = FALSE)
```



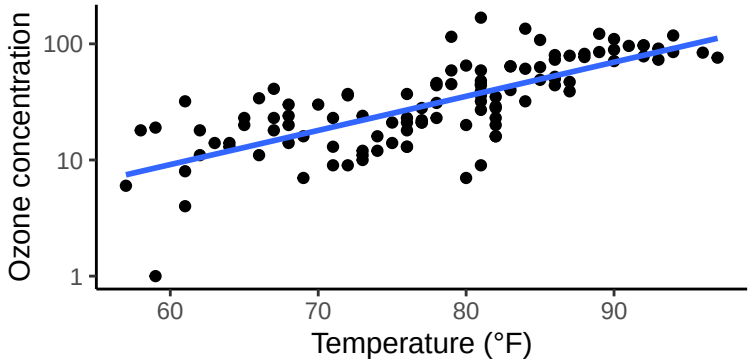
This means that we **underestimate** the ozone level for low and high temperatures and **overestimate** the ozone level at moderate temperatures.
Our predictions are **systematically wrong** for certain ranges of temperature.

> **Note**
>
> If the linearity assumption fails, there's not much point checking the other assumptions because it's not an appropriate prediction model.

Transformations

If we see a non-linear relationship between y and x we might be able to transform the data so that we have a linear relationship between the transformed variable(s).
What if we considered the log of ozone concentration?

```
p1 + scale_y_log10()
```



Air pollution

```
environmental = environmental |>
  mutate(lozone = log(ozone))
lm2 = lm(lozone ~ temperature, data = environmental)
lm2
```

```
Call:
lm(formula = lozone ~ temperature, data = environmental)

Coefficients:
(Intercept)  temperature
   -1.84852      0.06767
```

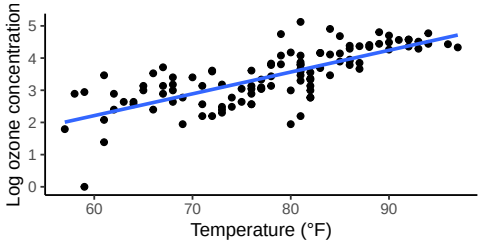
Now the fitted model is:

$$\widehat{\log(\text{ozone})} = -1.84852 + 0.06767 \times \text{temperature}$$

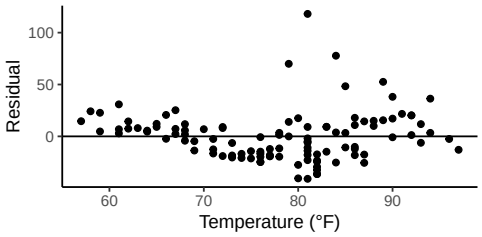
```
environmental = environmental |>
  mutate(
    lfitted = lm2$fitted.values,
    lresid = lm2$residuals
  )
```

Assumption 1: linearity

```
p1 = environmental |> ggplot() +
  aes(x = temperature, y = lozone) +
  geom_point() +
  theme_classic() +
  labs(x="Temperature (°F)",
       y="Log ozone concentration") +
  geom_smooth(method="lm", se=FALSE)
p1
```




```
p2 = environmental |> ggplot() +
  aes(x = temperature, y = resid) +
  geom_point() +
  labs(x = "Temperature (°F)",
       y = "Residual") +
  geom_hline(yintercept = 0) +
  theme_classic()
p2
```



Assumption 2: independence

The assumption of independence between the errors is usually dealt with in the experimental design phase - **before data collection**.

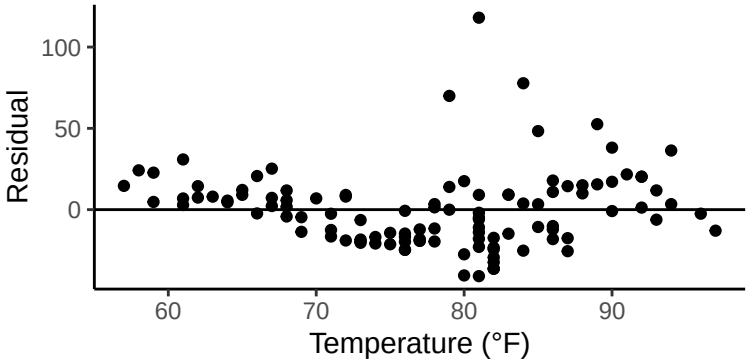
- You aim to design the experiment so that the observations are not related to one another.
- If you don't have a random sample, your estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ may be biased.
- Violations of independence often arise in time series data where observations are measured on the same subject through time and therefore may be related to one another. This is beyond the scope of DATA2002.

In the environmental data, there may be dependence that we haven't accounted as it is a time series data set (though we don't know which days they were taken on and if the records were sequential).

Assumption 3: homoskedasticity

- Homoskedasticity (homo: same, skedasticity: spread)
- Constant error variance is important to ensure the hypothesis tests to give valid results.
- Violations of homoskedasticity, called **heteroskedasticity**, make it difficult to estimate the "true" standard deviation of the errors, resulting in confidence intervals that are too wide or too narrow.
- Heteroskedasticity may also have the effect of giving too much weight to small subset of the data (namely the subset where the error variance was largest) when estimating coefficients.
- You can check for homoskedasticity in plots of residuals versus x. If it appears the residuals are getting more spread-out, that is evidence of heteroskedasticity

```
p2
```

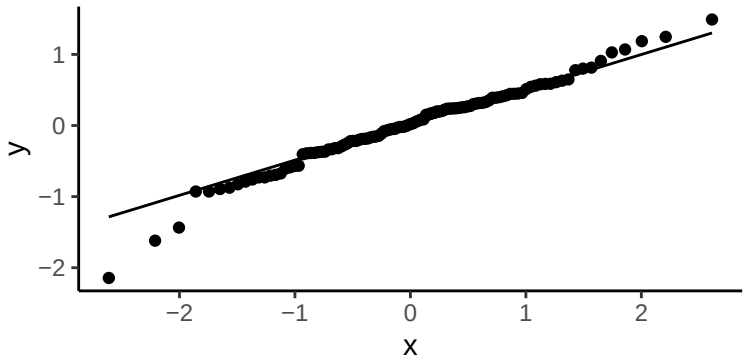


The spread looks reasonably constant over the range of temperature values.
In the region above 85°F, the spread might be somewhat smaller than the spread in the region below 85°F but it's nothing to get too worried about.

Assumption 4: normality

- Violations of normality of the errors can compromise our inferences. The calculation of confidence intervals may be too wide or narrow and our conclusions from our hypothesis tests may be incorrect.
- The best way to check (visually) for normality is a QQ plot of the residuals.
- In some cases, the problem may be due to one or two outliers. Such values should be scrutinised closely: are they genuine, are they explainable, are similar events likely to occur again in the future.
- Sometimes the extreme values in the data provide the most useful information.

```
environmental |> ggplot() +
  aes(sample = lresid) +
  geom_qq() + geom_qq_line() +
  theme_classic()
```

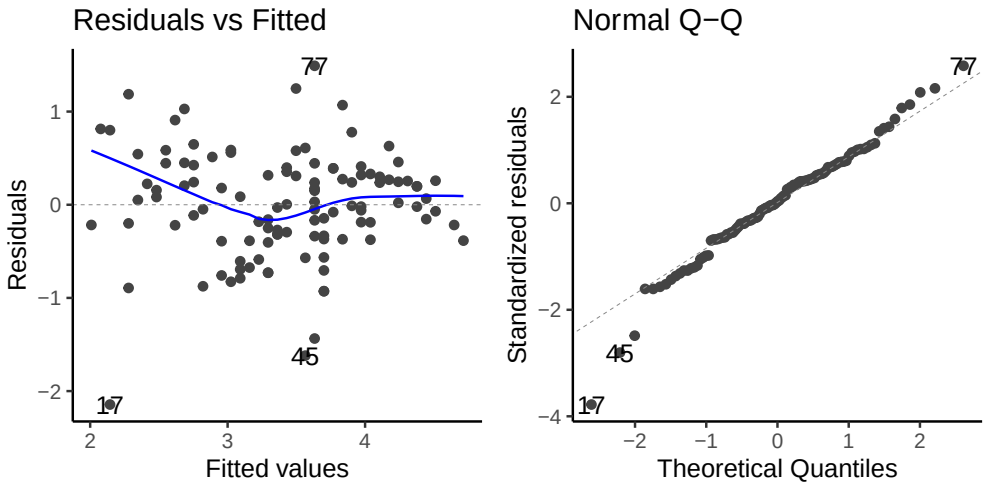


Apart from three points in the lower tail, the majority of the points lie quite close to the diagonal line in the QQ plot. Hence, the normality assumption for the residuals is reasonably well satisfied. Additionally, we have quite a large sample size so we can also rely on the central limit theorem to give us approximately valid inferences.

Autoplot

The **ggfortify** package provides an `autoplot()` method for `lm` objects.

```
autoplot(lm2, which = 1:2) + theme_classic()
```



27

Multiple regression

27.1 Interpreting model coefficients

How can we interpret the estimated coefficients?

$$Y = \beta_0 + \beta_1 x + \varepsilon$$

- The intercept is the expected value of Y when $x = 0$. i.e $\mathrm{E}(Y \mid x = 0) = \beta_0$.
- The slope is the amount we expect Y to change by when x increases one unit. i.e. for a one unit increase in x we expect Y to change by β_1 (could be an increase or decrease depending on the sign).

Recall our fitted model

```
data(environmental, package = "lattice")
environmental = environmental |>
  mutate(lozone = log(ozone))
lm2 = lm(lozone ~ temperature, data = environmental)
lm2
```

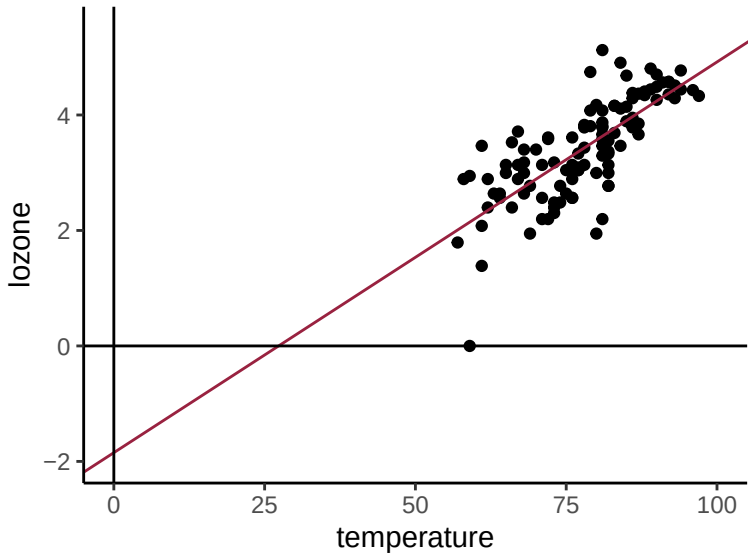
```
Call:
lm(formula = lozone ~ temperature, data = environmental)

Coefficients:
(Intercept)  temperature
   -1.84852      0.06767
```

$$\widehat{\log(\text{ozone})} = -1.84852 + 0.06767 \times \text{temperature}$$

> **How do we interpret this model?**
>
> Does it make sense to interpret the intercept?

Air pollution

```
environmental |> ggplot() + aes(x = temperature, y = lozone) +
  geom_point() + coord_cartesian(xlim = c(0,100), ylim = c(-2, 5.5)) +
  geom_abline(slope = 0.06767, intercept = -1.84852, colour = "#992240",) +
  geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +
  theme_classic()
```



Slope interpretation

$$\widehat{\log(\text{ozone})} = -1.84852 + 0.06767 \times \text{temperature}$$

- Interpreting the slope: a one degree Fahrenheit increase in temperature results in a 0.07 unit **increase** in log ozone, on average.
- A nicer way to interpret this is: a one degree Fahrenheit increase in temperature results in a 7% **increase** in ozone, on average.

Interpreting models with log transformations

Log-linear $\log(Y) = \beta_0 + \beta_1 x$
On average, a one unit increase in x will result in a $\beta_1 \times 100\%$ change in Y.
Linear-log $Y = \beta_0 + \beta_1 \log(x)$
On average, a one percent increase in x will result in a $\beta_1/100\%$ change in Y.
Log-log $\log(Y) = \beta_0 + \beta_1 \log(x)$
On average, a one percent increase in x will result in a $\beta_1\%$ change in Y.

27.2 Inference in regression models

Inference

Recall our simple linear regression population model:

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i.$$

Typically, we are interested in hypotheses of the form, $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ or $\beta_1 > 0$ or $\beta < 0$
To do this we use a t-test:

$$T = \frac{\hat{\beta}_1 - \beta_1}{\text{SE}(\hat{\beta}_1)} \sim t_{n-2}$$

where $\hat{\beta}_1$ and $\text{SE}(\hat{\beta}_1)$ are given in the R output.

```
summary(lm2)
```

```
Call:
lm(formula = lozone ~ temperature, data = environmental)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.14417 -0.32555  0.02066  0.34234  1.49100

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.848518   0.455080  -4.062  9.2e-05 ***
temperature  0.067673   0.005807  11.654  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5804 on 109 degrees of freedom
Multiple R^2:  0.5548,   Adjusted R^2:  0.5507
F-statistic: 135.8 on 1 and 109 DF,  p-value: < 2.2e-16
```

Testing for the significance of the slope parameter β_1

Workflow: Test the significance of β_1

- **Hypothesis**: $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 > 0, \beta_1 < 0, \beta_1 \neq 0$
- **Assumptions**: The residuals ε_i are iid $\mathcal{N}(0, \sigma^2)$ and there is a linear relationship between y and x.
- **Test statistic**: $T = \frac{\hat{\beta}_1}{\text{SE}(\hat{\beta}_1)} \sim t_{n-2}$ under H_0.
- **Observed test statistic**: t_0 (from R)
- **p-value**:
  - $P(t_{n-2} \geqslant t_0)$ for $H_1 > 0$,
  - $P(t_{n-2} \leqslant t_0)$ for $H_1 < 0$, or
  - $2P(t_{n-2} \geqslant t_0)$ for $H_1 \neq 0$
- **Decision:** Reject H_0 if the p-value is less than the level of significance, α.

Air pollution: $\log(Y_i) = \beta_0 + \beta_1 x_i + \varepsilon_i$

- **Hypothesis**: $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$
- **Assumptions**: The residuals ε_i are iid $\mathcal{N}(0, \sigma^2)$ and there is a linear relationship between y and x. (checked previously).
- **Test statistic**: $T = \frac{\hat{\beta}_1}{\text{SE}(\hat{\beta}_1)} \sim t_{n-2}$ under H_0.
- **Observed test statistic**: $t_0 = \frac{0.0677}{0.00581} = 11.65$
- **p-value**: $2P(t_{109} \geqslant 11.95) < 0.0001$
- **Decision:** There is very strong evidence in the data to indicate a linear relationship between temperature and the logarithm of ozone concentration.

```
lm2 |> broom::tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -1.85     0.455       -4.06 9.20e- 5
2 temperature   0.0677   0.00581     11.7  7.17e-21
```

CI for regression coefficients

$100(1-\alpha)\%$ confidence intervals can be constructed for regression coefficients in the usual way:

$$\hat{\beta}_1 \pm t^\star \times \text{SE}(\hat{\beta}_1)$$

where $t^\star$ is the $\alpha/2$ quantile from a t distribution with $n-2$ degrees of freedom.

```
summary(lm2)$coefficients |> round(4)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.8485     0.4551 -4.0620    1e-04
temperature   0.0677     0.0058 11.6539    0e+00
```

```
qt(0.025, df = 109) |> round(3)
```

```
[1] -1.982
```

Plugging in the appropriate values:

$$0.0677 \pm 1.982 \times 0.0058 = (0.056, 0.079).$$

We can also use the `confint()` function

```
confint(lm2) |> round(3)
```

```
             2.5 % 97.5 %
(Intercept) -2.750 -0.947
temperature  0.056  0.079
```

27.3 In-sample performance

Decomposing the error

$$\underbrace{\sum_{i=1}^{n}(y_i - \overline{y})^2}_{TotalSS} = \underbrace{\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2}_{RegSS} + \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{ResidSS}$$

where

- Total SS is the **total** variation in Y
- Reg SS is the sum of squares **explained** by the regression line (regression sum of squares)
- Resid SS = Total SS − Reg SS is the variation in Y remain **unexplained** (residual sum of squares)

Coefficient of determination r^2

The square of correlation coefficient r^2 called the **coefficient of determination** measures the proportion of total variation in Y *explained* by the linear regression model: It is "one minus the proportion of variation not explained by the model":

$$r^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2} = 1 - \frac{\text{Resid SS}}{\text{Total SS}}.$$

Hence the **coefficient of determination** r^2 measures the strength of the linear relationship between x and y by the percentage of variation in y explained by the linear regression model in x.

Air pollution

```
summary(lm2)
```

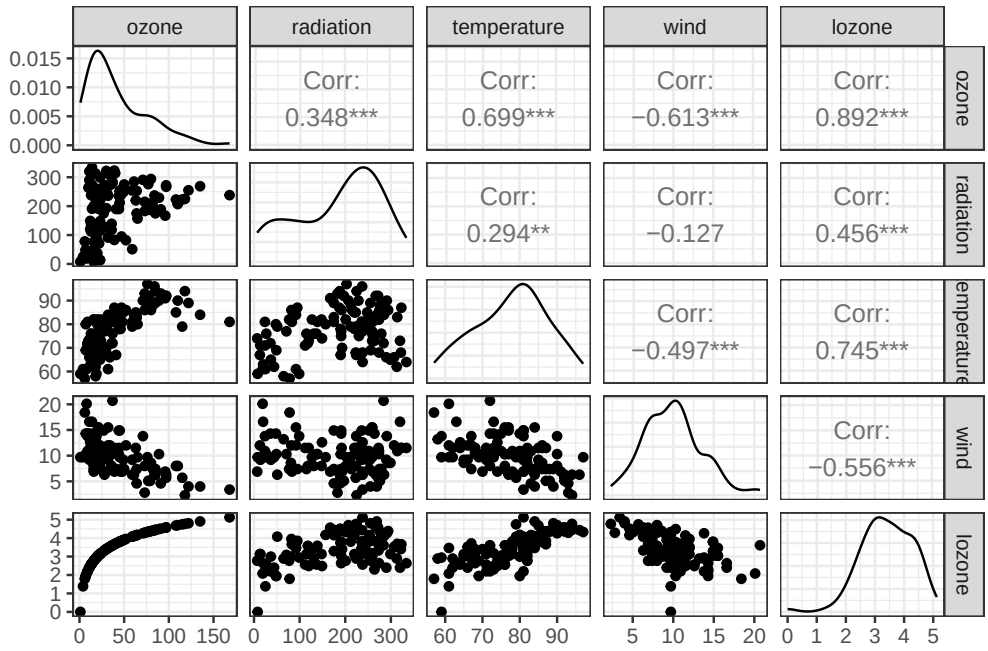
```
Call:
lm(formula = lozone ~ temperature, data = environmental)

Residuals:
     Min       1Q   Median       3Q      Max
-2.14417 -0.32555  0.02066  0.34234  1.49100

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.848518   0.455080  -4.062  9.2e-05 ***
temperature  0.067673   0.005807  11.654  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5804 on 109 degrees of freedom
Multiple R^2:  0.5548,   Adjusted R^2:  0.5507
F-statistic: 135.8 on 1 and 109 DF,  p-value: < 2.2e-16
```

The r^2 in the ozone example is 0.5548.
Interpretation: We can say that temperature explains 55% of the observed variation in the logarithm of ozone concentration.

> **Can we do better if we use more variables to help explain the logarithm of ozone concentration?**

27.4 Multiple regression

```
library(GGally)
GGally::ggpairs(environmental) + theme_bw()
```



What if we extended our question? Can *radiation*, *temperature* and *wind* be used to predict the log of ozone?

$$\log(\text{ozone})_i = \beta_0 + \beta_1 \text{radiation}_i + \beta_2 \text{temperature}_i + \beta_3 \text{wind}_i + \varepsilon_i$$

```
lm3 = lm(lozone ~ radiation + temperature + wind, environmental)
summary(lm3)$coefficients |> round(3)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.261      0.553  -0.472    0.638
radiation      0.003      0.001   4.518    0.000
temperature    0.049      0.006   8.078    0.000
wind          -0.062      0.016  -3.922    0.000
```

Fitted model:

$$\widehat{\log(\text{ozone})} = -0.261 + 0.003 \text{ radiation} + 0.049 \text{ temperature} - 0.062 \text{ wind}$$

Multiple regression is a natural extension of simple linear regression that incorporates multiple explanatory (or predictor) variables. It has the general form,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

Often it's convenient to write the model in matrix format,

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_n)'$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \ldots, \beta_p)'$, $\boldsymbol{\varepsilon} \sim N_n(\mathbf{0}, \sigma^2\mathbf{I})$ and

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1' \\ \mathbf{x}_2' \\ \\ \mathbf{x}_n' \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ & & & & \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix}$$

where $\mathbf{x}_i = (\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{ip}))$ is the vector of predictors for the ith observation. The least squares solution is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}.$$

Interpretation

The estimated coefficients $\hat{\beta}$'s are now interpreted as "conditional on" the other variables - each β_i reflects the predicted change in y associated with a one unit increase in x_i, holding the other variables constant (i.e. a marginal effect).

$$\widehat{\log(\text{ozone})} = -0.261 + 0.003 \text{ radiation} + 0.049 \text{ temperature} - 0.062 \text{ wind}$$

- A one degree Fahrenheit increase in temperature results in a 4.9% **increase** in ozone on average, holding radiation and wind speed constant.
- A one langley increase solar radiation results in a 0.3% **increase** in ozone on average, holding radiation and wind constant.
- A 10 langley increase solar radiation results in a 3% **increase** in ozone on average, holding radiation and wind constant.
- A one mile per hour increase in average wind speed results in a 6.2% **decrease** in ozone on average, holding radiation and temperature constant.

In-sample performance

The r^2 value has the same interpretation: proportion of total variability in Y explained by the regression model.

Simple linear regression model

```
summary(lm2)$r.squared
```

```
[1] 0.5547615
```

"Full" model

```
summary(lm3)$r.squared
```

```
[1] 0.664515
```

```
library(modelsummary)
models = list(
        "Forward Model" = lm2,
        "Backwards Model" = lm3)
modelsummary(models, output = 'kableExtra', gof_omit = 'AIC|BIC|F|L|RMSE',
    statistic = "{p.value}", stars = c('*' = .1, '**' = .05, '***' = 0.01)) |>
  kable_styling(position="center", latex_options = "hold_position")
```

	Forward Model	Backwards Model
(Intercept)	−1.849***	−0.261
	<0.001	0.638
temperature	0.068***	0.049***
	<0.001	<0.001
radiation		0.003***
		<0.001
wind		−0.062***
		<0.001
Num.Obs.	111	111
R2	0.555	0.665
R2 Adj.	0.551	0.655

* p < 0.1, ** p < 0.05, *** p < 0.01

28

Prediction and categorical predictors

28.1 Prediction

Recall our fitted ozone model

```
data(environmental, package = "lattice")
environmental = environmental |>
  mutate(lozone = log(ozone))
lm3 = lm(lozone ~ radiation + temperature + wind, environmental)
lm3
```

```

Call:
lm(formula = lozone ~ radiation + temperature + wind, data = environmental)
```

```
Coefficients:
(Intercept)    radiation  temperature         wind
  -0.261174     0.002515     0.049163    -0.061593
```

$$\widehat{\log(\text{ozone})} = -0.26 + 0.0025 \text{ radiation} + 0.049 \text{ temperature} - 0.0616 \text{ wind}$$

Prediction

Say we want to predict the (log) ozone when the solar radiation is 200 langley, the temperature is 90 degrees Fahrenheit and the average wind speed is 15 miles per hour. We can substitute these into our fitted model:

$$\begin{aligned}\widehat{\log(\text{ozone})} &= -0.26 + 0.0025 \text{ radiation} + 0.049 \text{ temperature} - 0.0616 \text{ wind} \\ &= -0.26 + 0.0025 \times 200 + 0.049 \times 90 - 0.0616 \times 15 \\ &\approx 3.7\end{aligned}$$

Prediction in R

We need to generate a new data frame with the same column names as the original variables:

```
new_obs = data.frame(radiation = 200, temperature = 90, wind = 15)
```

And then feed this into the `predict()` function:

```
predict(lm3, new_obs, interval = "prediction", level = 0.90)
```

```
       fit      lwr      upr
1 3.742554 2.867449 4.617659
```

```
predict(lm3, new_obs, interval = "confidence", level = 0.90)
```

```
       fit      lwr     upr
1 3.742554 3.510278 3.97483
```

We have two options for the interval type: **prediction interval**and **confidence interval**.

Two kinds of "prediction"

Estimate the the mean concentration on days we see certain conditions

Estimate the **average log ozone concentration** when the solar radiation is 200 langley, the temperature is 90 degrees Fahrenheit and the average wind speed is 15 miles per hour and give a 90% confidence interval of this **estimate**.
The confidence interval gives you a range of plausible values for the mean parameter (given certain conditions).

Make a prediction for a specific day

Predict the **log ozone concentration on a day** when solar radiation is 200 langley, the temperature is 90 degrees Fahrenheit and the average wind speed is 15 miles per hour. Give a 90% prediction interval of this **prediction**.
The prediction interval gives you a range of plausible values for a new individual data point to lie (given certain conditions).

Prediction vs confidence intervals

Take a regression model with n observations and p regressors,

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon.$$

Given a new observation vector $\mathbf{x_0}$, the predicted value for that observation is,

$$\mathrm{E}(Y \mid \mathbf{x_0}) = \hat{y}_0 = \mathbf{x}_0'\hat{\boldsymbol{\beta}}.$$

A consistent estimator of the variance of this prediction is,

$$\widehat{\mathrm{Var}}(\hat{y_0}) = \hat{\sigma}^2\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0,$$

where

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N-k} = \frac{\sum_{i=1}^{N} r_i^2}{N-k}.$$

The forecast error for **a particular** y_0 is

$$\hat{e} = y_0 - \hat{y}_0 = (\mathbf{x}_0'\boldsymbol{\beta} + \varepsilon_0) - \hat{y}_0.$$

There is zero covariance between ε_0 and $\hat{\beta}$ so,

$$\mathrm{Var}(\hat{e}) = \mathrm{Var}(\hat{y}_0) + \mathrm{Var}(\varepsilon_0),$$

and a consistent estimator of that is

$$\mathrm{Var}(\hat{\varepsilon}) = \hat{\sigma}^2\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0 + \hat{\sigma}^2.$$

The $1-\alpha$ **confidence** interval will be: $\hat{y}_0 \pm t^\star\sqrt{\mathrm{Var}(\hat{y}_0)}$ where $\mathrm{Var}(\hat{y}_0) = \hat{\sigma}^2\mathbf{x}_0(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0'$
The $1-\alpha$ **prediction** interval will be wider: $\hat{y}_0 \pm t^\star\sqrt{\mathrm{Var}(\hat{e})}$

```
predict(lm3, new_obs,
        interval = "prediction",
        level = 0.90, se.fit = TRUE)
```

```
$fit
       fit      lwr      upr
1 3.742554 2.867449 4.617659

$se.fit
[1] 0.139991

$df
[1] 107

$residual.scale
[1] 0.5085019
```

```
predict(lm3, new_obs,
        interval = "confidence",
        level = 0.90)
```

```
       fit      lwr      upr
1 3.742554 3.510278 3.97483
```

Quantile for 90% intervals:

```
qt(0.95, 107)
```

```
[1] 1.659219
```

Prediction interval $\hat{y}_0 \pm t^\star \sqrt{\text{Var}(\hat{e})}$

$$3.74 \pm 1.659 \times \sqrt{0.14^2 + 0.51^2}$$

Confidence interval $\hat{y}_0 \pm t^\star \sqrt{\text{Var}(\hat{y}_0)}$

$$3.74 \pm 1.659 \times 0.14$$

Effect of variance on intervals

Our population model is:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

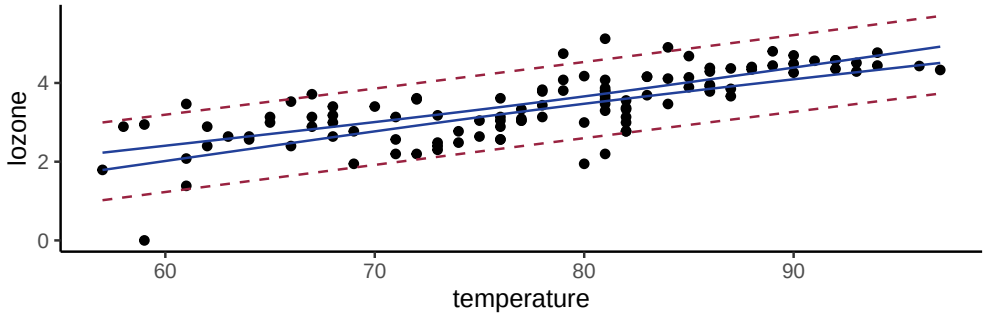
where $\varepsilon \sim N_n(0, \sigma^2)$.

1. The smaller the σ^2, the better the fit and hence the smaller the variances for $\hat{\beta}$ and $\hat{y}_0$.
2. The larger the spread of our x variables, the more information we have about how Y responds to each x variable and hence the smaller the variances for $\hat{\beta}$ and $\hat{y}_0$.
3. The larger the sample size n, the smaller the variances for $\hat{\beta}$ and $\hat{y}_0$.
4. The closer is $\mathbf{x}_0$ to $\overline{\mathbf{x}}$ (the component-wise mean of the design matrix), the smaller the variances of $\hat{y}_0$.

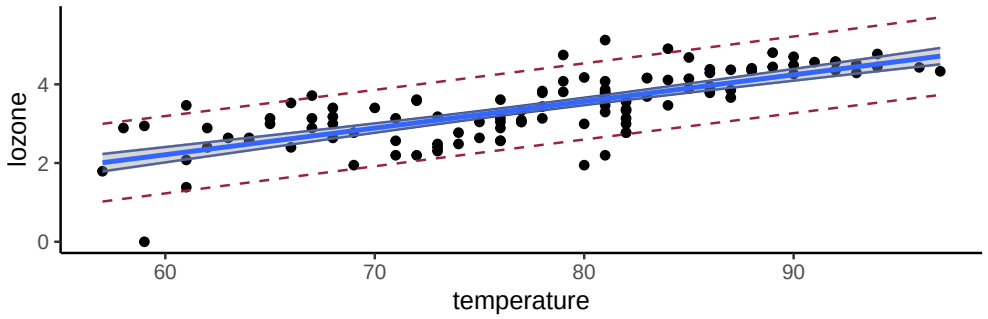
Confidence and prediction intervals

For illustration purposes, let's return to the **simple linear regression** of log ozone on temperature.

```
lm2 = lm(lozone ~ temperature, data = environmental)
new_temp = data.frame(
  temperature = seq(from = min(environmental$temperature),
                    to = max(environmental$temperature),
                    by = 0.1)
)
pred_int = predict(lm2, new_temp, interval = "prediction", level = 0.90) |>
  data.frame()
conf_int = predict(lm2, new_temp, interval = "confidence", level = 0.90) |>
  data.frame()
interval_df = data.frame(
  pi_upper = pred_int$upr,
  pi_lower = pred_int$lwr,
  ci_upper = conf_int$upr,
  ci_lower = conf_int$lwr,
  temperature = new_temp$temperature
)
environmental |> ggplot() + aes(x = temperature, y = lozone) +
  geom_point() +
  geom_line(data = interval_df, aes(y=pi_lower), color = "#992240", linetype =
        2) +
  geom_line(data = interval_df, aes(y=pi_upper), color = "#992240", linetype =
        2) +
  geom_line(data = interval_df, aes(y=ci_lower), color = "#224099", linetype =
        1) +
  geom_line(data = interval_df, aes(y=ci_upper), color = "#224099", linetype =
        1) +
  theme_classic()
```




```
environmental |> ggplot() + aes(x = temperature, y = lozone) +
  geom_point() +
  geom_line(data = interval_df, aes(y=pi_lower), color = "#992240", linetype =
      2) +
  geom_line(data = interval_df, aes(y=pi_upper), color = "#992240", linetype =
      2) +
  geom_line(data = interval_df, aes(y=ci_lower), color = "#224099", linetype =
      1) +
  geom_line(data = interval_df, aes(y=ci_upper), color = "#224099", linetype =
      1) +
  geom_smooth(method = "lm", se = TRUE) +
  theme_classic()
```



28.2 Categorical predictors

Fuel economy

The `mpg` dataset contains a subset of the fuel economy data collated by the EPA. It contains only models which had a new release every year between 1999 and 2008.

```
data("mpg", package = "ggplot2")
glimpse(mpg)
```

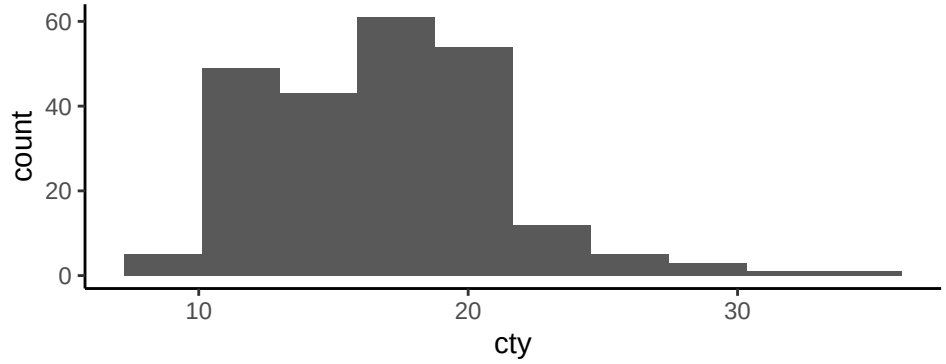
```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "
    ~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "
    ~
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.
    ~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200
    ~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8,
    ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto
    ~
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4
    ~
```

```
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1
   ~
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2
   ~
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p
   ~
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "c
   ~
```

In a regression context, the dependent variable could be either `cty` (city miles per gallon) or `hwy` (highway miles per gallon).

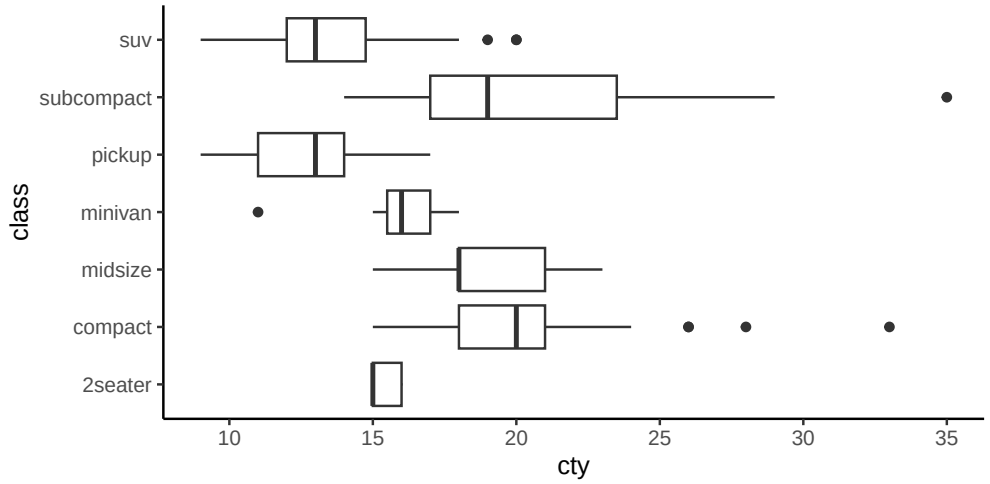
Initial look at the data

```
mpg |> ggplot() +
  aes(x = cty) +
  geom_histogram(bins = 10) +
  theme_classic()
```



Fuel economy by type of car

```
mpg |> ggplot() +
  aes(y = class, x = cty) +
  geom_boxplot() +
  theme_classic()
```



Categorical predictors

Think ANOVA!

```
lm1 = lm(cty ~ class, data = mpg)
summary(lm1)
```

```
Call:
lm(formula = cty ~ class, data = mpg)

Residuals:
    Min      1Q  Median      3Q     Max
-6.3714 -1.7561 -0.1277  1.0000 14.6286

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      15.4000     1.3024  11.824  < 2e-16 ***
classcompact      4.7277     1.3699   3.451 0.000666 ***
classmidsize      3.3561     1.3796   2.433 0.015759 *
classminivan      0.4182     1.5708   0.266 0.790307
classpickup      -2.4000     1.3976  -1.717 0.087304 .
classsubcompact   4.9714     1.3923   3.571 0.000435 ***
classsuv         -1.9000     1.3539  -1.403 0.161884
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.912 on 227 degrees of freedom
Multiple R^2:  0.5438,   Adjusted R^2:  0.5317
F-statistic:  45.1 on 6 and 227 DF,  p-value: < 2.2e-16
```

How does this compare with ANOVA?

Let's compare with a one-way ANOVA:

```
a1 = aov(cty ~ class, data = mpg)
summary(a1)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
class         6   2295   382.5    45.1 <2e-16 ***
Residuals   227   1925     8.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If we apply the `summary()` function to each, they look different. There are some similarities though…

- the `summary(lm1)` mentions a F-statistic of 45.1 on 6 and 227 degrees of freedom…
- the `summary(lm1)` talks about a residual standard error of 2.912 which just happens to be very similar to $\sqrt{8.5}$, the square root of the Residual Mean Sq value above…

What about the `anova()` function?

If we apply the `anova()` function to each, they look are the same!

`lm()` object

```
anova(lm1)
```

```
Analysis of Variance Table

Response: cty
           Df Sum Sq Mean Sq F value
                 Pr(>F)
class       6 2295.0  382.51  45.099 <
     2.2e-16 ***
Residuals 227 1925.3    8.48
---
Signif. codes:  0 '***' 0.001 '**'
    0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`aov()` object

```
anova(a1)
```

```
Analysis of Variance Table

Response: cty
           Df Sum Sq Mean Sq F value
                 Pr(>F)
class       6 2295.0  382.51  45.099 <
     2.2e-16 ***
Residuals 227 1925.3    8.48
---
Signif. codes:  0 '***' 0.001 '**'
    0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What about `emmeans`?

Similarly if we look at `emmeans()` we get the same results,

`lm()` object

```
lm_e1 = emmeans(lm1, ~ class)
lm_e1
```

```
 class      emmean    SE  df lower.CL
     upper.CL
 2seater      15.4 1.302 227     12.8
         18.0
 compact      20.1 0.425 227     19.3
         21.0
 midsize      18.8 0.455 227     17.9
         19.7
 minivan      15.8 0.878 227     14.1
         17.5
 pickup       13.0 0.507 227     12.0
         14.0
 subcompact   20.4 0.492 227     19.4
         21.3
 suv          13.5 0.370 227     12.8
         14.2

Confidence level used: 0.95
```

`aov()` object

```
e1 = emmeans(a1, ~ class)
e1
```

```
 class      emmean    SE  df lower.CL
     upper.CL
 2seater      15.4 1.302 227     12.8
         18.0
 compact      20.1 0.425 227     19.3
         21.0
 midsize      18.8 0.455 227     17.9
         19.7
 minivan      15.8 0.878 227     14.1
         17.5
 pickup       13.0 0.507 227     12.0
         14.0
 subcompact   20.4 0.492 227     19.4
         21.3
 suv          13.5 0.370 227     12.8
         14.2

Confidence level used: 0.95
```

What is the relationship between `emmeans()` and the estimated coefficients?

```
summary(lm1) |>
  broom::tidy() |>
  select(term, estimate)
```

```
# A tibble: 7 x 2
  term            estimate
  <chr>              <dbl>
1 (Intercept)        15.4
2 classcompact        4.73
3 classmidsize        3.36
4 classminivan        0.418
5 classpickup        -2.40
6 classsubcompact     4.97
7 classsuv           -1.90
```

```
lm_e1 |> data.frame() |>
  select(class, emmean) |>
  tibble()
```

```
# A tibble: 7 x 2
  class      emmean
  <fct>       <dbl>
1 2seater      15.4
2 compact      20.1
3 midsize      18.8
4 minivan      15.8
5 pickup       13.0
6 subcompact   20.4
7 suv          13.5
```

```
summary(lm1) |>
  broom::tidy() |>
  select(term, estimate)
```

```
# A tibble: 7 x 2
  term            estimate
  <chr>              <dbl>
1 (Intercept)        15.4
2 classcompact        4.73
3 classmidsize        3.36
4 classminivan        0.418
5 classpickup        -2.40
6 classsubcompact     4.97
7 classsuv           -1.90
```

```
lm_e1 |>
  contrast(method = "revpairwise") |>
  data.frame() |>
  select(contrast, estimate) |>
  filter(
    stringr::str_detect(contrast,"2
        seater")
  ) |> tibble()
```

```
# A tibble: 6 x 2
  contrast             estimate
  <chr>                   <dbl>
1 compact - 2seater        4.73
2 midsize - 2seater        3.36
3 minivan - 2seater        0.418
4 pickup - 2seater        -2.40
5 subcompact - 2seater     4.97
6 suv - 2seater           -1.90
```

Interpreting coefficients of categorical predictors

So how do we interpret these coefficients? The model that is being fit is:

$$\text{cty} = \beta_0 + \beta_1(\text{class}_{\text{compact}}) + \beta_2(\text{class}_{\text{midsize}}) + \beta_3(\text{class}_{\text{minivan}}) + \beta_4(\text{class}_{\text{pickup}}) + \beta_5(\text{class}_{\text{subcompact}}) + \beta_6(\text{class}_{\text{suv}}) + \varepsilon$$

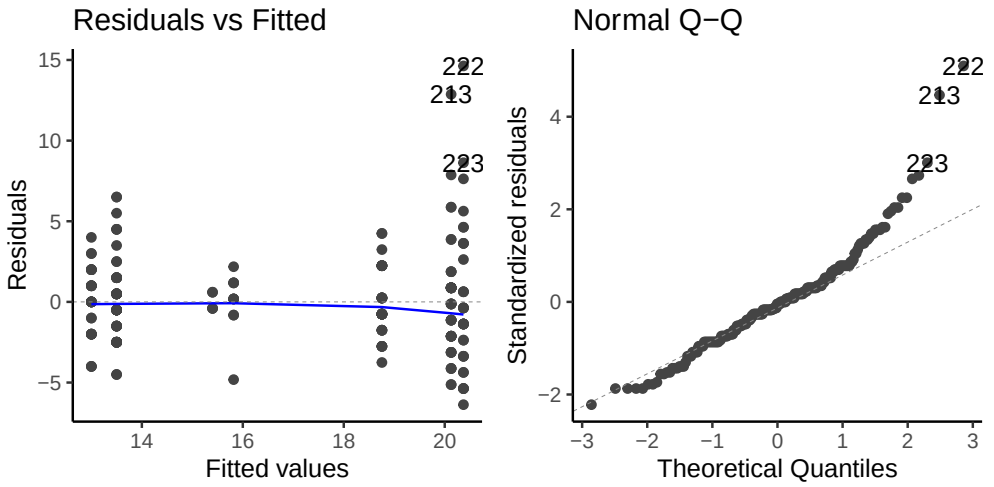
Where's $\text{class}_{\text{2 seater}}$

- The coefficients represent the difference between each level and the "baseline".
- In this case the baseline is the 2seater car type.

Assumption checking

If you only have categorical variables in your regression model, you can check your assumptions in the same way as ANOVA.

```
autoplot(lm1, which = 1:2) + theme_classic()
```



29

Model selection

29.1 Model selection

US crime rate data

- Ehrlich (1973) considered crime data from 47 states of the USA in 1960
- Our aim is to model the crime rate as a function of up to 15 potential explanatory variables
- We will consider **stepwise** and exhaustive search schemes

```
data("UScrime", package = "MASS")
```

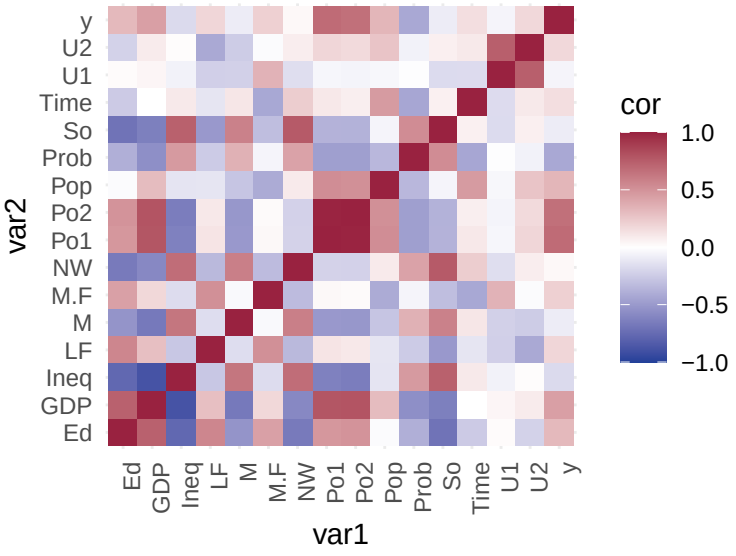
```
dim(UScrime)
```

```
[1] 47 16
```

```
n = 47
k = 15
2^k
```

```
[1] 32768
```

```
cor_mat = cor(UScrime)
melted_cor_mat = cor_mat |>
  data.frame() |>
  rownames_to_column(var = "var1") |>
  pivot_longer(cols = -var1,
               names_to = "var2",
               values_to = "cor")
melted_cor_mat |> ggplot() +
  aes(x=var1, y=var2, fill=cor) +
  geom_tile() +
  theme_minimal() +
  scale_fill_gradient2(
    low = "#224099",
    high = "#992240",
    mid = "white",
    midpoint = 0,
    limit = c(-1,1)) +
  theme(
    axis.text.x = element_text(
      angle = 90, hjust = 1
    )
  )
```



Crime data: variables in the data set

Variable	Description	Variable	Description
M	percentage of males aged 14-24	NW	number of nonwhites per 1000 people
So	indicator variable for a southern state	U1	unemployment rate of urban males 14-24
Ed	mean years of schooling	U2	unemployment rate of urban males 35-39
Po1	police expenditure in 1960	GDP	gross domestic product per head
Po2	police expenditure in 1959	Ineq	income inequality
LF	labour force participation rate	Prob	probability of imprisonment
M.F	number of males per 1000 females	Time	average time served in state prisons
Pop	state population	y	rate of crimes in a particular category per head of population

Crime data: null and full model

```
M0 = lm(y ~ 1, data = UScrime)  # Null model
M1 = lm(y ~ ., data = UScrime)  # Full model
round(summary(M1)$coef, 3)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5984.288   1628.318  -3.675    0.001
M               8.783      4.171   2.106    0.043
So             -3.803    148.755  -0.026    0.980
Ed             18.832      6.209   3.033    0.005
Po1            19.280     10.611   1.817    0.079
Po2           -10.942     11.748  -0.931    0.359
LF             -0.664      1.470  -0.452    0.655
M.F             1.741      2.035   0.855    0.399
Pop            -0.733      1.290  -0.568    0.574
NW              0.420      0.648   0.649    0.521
U1             -5.827      4.210  -1.384    0.176
U2             16.780      8.234   2.038    0.050
GDP             0.962      1.037   0.928    0.361
Ineq            7.067      2.272   3.111    0.004
Prob        -4855.266   2272.375  -2.137    0.041
Time           -3.479      7.165  -0.486    0.631
```

```
res = bind_rows(broom::glance(M1),
                broom::glance(M0))
res$model= c("M1","M0")
res |> pivot_longer(
  cols = -model,
  names_to = "metric",
  values_to = "value") |>
  pivot_wider(
    names_from = "model") |>
  gt::gt() |>
  gt::fmt_number(columns = 2:3,
                 decimals = 2) |>
  gt::sub_missing()
```

metric	M1	M0
r.squared	0.80	0.00
adj.r.squared	0.71	0.00
sigma	209.06	386.76
statistic	8.43	—
p.value	0.00	—
df	15.00	—
logLik	−308.01	−346.20
AIC	650.03	696.40
BIC	681.48	700.10
deviance	1,354,945.77	6,880,927.66
df.residual	31.00	46.00
nobs	47.00	47.00

29.2 Stepwise selection

The `drop1` and `update` command in R

- For a response variable `Y` and explanatory variables `x1`, `...`, `xk` stored in the data frame `dat` consider
- `M1 = lm(Y ~ ., data = dat)`
- The function `drop1(M1, test = "F")` returns a number of information criteria for all explanatory variables used in `M1`
- In particular, this includes the p-values of the F-test for $H_0 : \beta_j = 0$ versus $H_1 : \beta_j \neq 0$ for all $j = 1, \ldots, k$
- To efficiently delete a variable from regression model `M1`, say `x1`, the `update()` function can be used:
- `M2 = update(M1, . ~. - x1)`
- The full stops in the `update()` formula `.~.` stand for "whatever was in the corresponding position in the old formula"

Crime data: `drop1` and `update`

Start with the full model, `M1`

```
drop1(M1, test = "F")
```

```
Single term deletions

Model:
y ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
    GDP + Ineq + Prob + Time
       Df Sum of Sq     RSS    AIC F value   Pr(>F)
<none>              1354946 514.65
M       1    193770 1548716 518.93  4.4333 0.043443 *
So      1        29 1354974 512.65  0.0007 0.979765
Ed      1    402117 1757063 524.86  9.2001 0.004861 **
Po1     1    144306 1499252 517.41  3.3016 0.078892 .
Po2     1     37919 1392865 513.95  0.8676 0.358830
LF      1      8917 1363862 512.96  0.2040 0.654654
M.F     1     31967 1386913 513.74  0.7314 0.398995
Pop     1     14122 1369068 513.14  0.3231 0.573845
NW      1     18395 1373341 513.28  0.4209 0.521279
U1      1     83722 1438668 515.47  1.9155 0.176238
U2      1    181536 1536482 518.56  4.1534 0.050161 .
GDP     1     37613 1392558 513.94  0.8605 0.360754
Ineq    1    423031 1777977 525.42  9.6786 0.003983 **
```

```
Prob     1    199538 1554484 519.11  4.5653 0.040627 *
Time     1     10304 1365250 513.00  0.2357 0.630708
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Remove So (it has the largest F-statistic p-value) and create M2

```
M2 = update(M1, . ~ . - So)
drop1(M2, test = "F")
```

```
Single term deletions

Model:
y ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 + GDP +
    Ineq + Prob + Time
       Df Sum of Sq     RSS    AIC F value   Pr(>F)
<none>               1354974 512.65
M       1    195084 1550059 516.97  4.6072 0.039520 *
Ed      1    403140 1758114 522.89  9.5208 0.004170 **
Po1     1    144302 1499277 515.41  3.4079 0.074152 .
Po2     1     37954 1392929 511.95  0.8964 0.350855
LF      1     10878 1365852 511.03  0.2569 0.615736
M.F     1     32449 1387423 511.76  0.7663 0.387876
Pop     1     14127 1369101 511.14  0.3336 0.567573
NW      1     21626 1376600 511.39  0.5107 0.480002
U1      1     96420 1451395 513.88  2.2771 0.141106
U2      1    189859 1544834 516.81  4.4839 0.042081 *
GDP     1     39223 1394197 511.99  0.9263 0.343041
Ineq    1    488834 1843808 525.13 11.5446 0.001834 **
Prob    1    204463 1559437 517.26  4.8287 0.035344 *
Time    1     10341 1365315 511.01  0.2442 0.624556
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
M3 = update(M2, . ~ . - Time)
drop1(M3, test = "F")
```

```
Single term deletions

Model:
y ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 + GDP +
    Ineq + Prob
       Df Sum of Sq     RSS    AIC F value   Pr(>F)
<none>               1365315 511.01
M       1    186110 1551425 515.01  4.4983 0.041529 *
Ed      1    409448 1774763 521.33  9.8965 0.003497 **
Po1     1    134137 1499452 513.41  3.2421 0.080914 .
Po2     1     28932 1394247 509.99  0.6993 0.409032
LF      1     10533 1375848 509.37  0.2546 0.617215
M.F     1     41784 1407099 510.42  1.0099 0.322234
Pop     1     21846 1387161 509.75  0.5280 0.472563
NW      1     15482 1380797 509.54  0.3742 0.544915
U1      1     91420 1456735 512.05  2.2096 0.146643
U2      1    184143 1549458 514.95  4.4508 0.042553 *
GDP     1     36070 1401385 510.23  0.8718 0.357237
Ineq    1    502909 1868224 523.75 12.1554 0.001406 **
Prob    1    237493 1602808 516.54  5.7403 0.022409 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
M4 = update(M3, . ~ . - LF)
drop1(M4, test = "F")
```

```
Single term deletions

Model:
```

```
y ~ M + Ed + Po1 + Po2 + M.F + Pop + NW + U1 + U2 + GDP + Ineq +
    Prob
       Df Sum of Sq     RSS    AIC F value   Pr(>F)
<none>              1375848 509.37
M       1    217716 1593564 514.27  5.3802 0.026506 *
Ed      1    413254 1789103 519.71 10.2124 0.003008 **
Po1     1    123896 1499744 511.42  3.0617 0.089178 .
Po2     1     21418 1397266 508.09  0.5293 0.471886
M.F     1     31252 1407100 508.42  0.7723 0.385674
Pop     1     27803 1403651 508.31  0.6871 0.412947
NW      1     11675 1387523 507.77  0.2885 0.594679
U1      1     80954 1456802 510.06  2.0005 0.166336
U2      1    190746 1566594 513.47  4.7137 0.036996 *
GDP     1     35035 1410883 508.55  0.8658 0.358685
Ineq    1    500944 1876792 521.96 12.3793 0.001256 **
Prob    1    226971 1602819 514.54  5.6089 0.023696 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
M5 = update(M4, . ~ . - NW)
```

and so on…

Backward variable selection

1. Start with model containing all possible explanatory variables
2. For each variable in turn, investigate effect of removing variable from current model
3. Remove the least informative variable, unless this variable is nonetheless supplying significant information about the response
4. Go to step 2. Stop only if all variables in the current model are important

The Akaike information criterion

- **AIC** is the most widely known and used model selection method (of course this does not imply it is the best/recommended method to use)
- The **AIC** was introduced by Hirotugu Akaike in his seminal 1973 paper and for our linear regression models, is defined as,

$$\text{AIC} = n \log\left(\frac{\text{Residual sum of squares}}{n}\right) + 2p$$

- The smaller the **AIC** the better the model
- Models that differ by less than one or two **AIC** values can be regarded as somewhat equally well fitting

Crime data: backward search using AIC

```
step.back.aic = step(M1,
                     direction = "backward",
                     trace = FALSE)
round(summary(step.back.aic)$coef,3)
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6426.101   1194.611  -5.379    0.000
M               9.332      3.350   2.786    0.008
Ed             18.012      5.275   3.414    0.002
Po1            10.265      1.552   6.613    0.000
M.F             2.234      1.360   1.642    0.109
U1             -6.087      3.339  -1.823    0.076
U2             18.735      7.248   2.585    0.014
```

```
Ineq            6.133      1.396   4.394    0.000
Prob        -3796.032   1490.646  -2.547    0.015
```

Use `trace = TRUE` to see the sequence of models considered by the procedure.

```
step.back.aic |>
  broom::glance() |>
  round(2) |> t()
```

```
                   [,1]
r.squared          0.79
adj.r.squared      0.74
sigma            195.55
statistic         17.74
p.value            0.00
df                 8.00
logLik          -309.66
AIC              639.32
BIC              657.82
deviance      1453067.77
df.residual        38.00
nobs               47.00
```

Crime data: `drop1`

We could try to see if we can drop any variables from the backward stepwise using AIC model:

```
drop1(step.back.aic, test = "F")
```

```
Single term deletions

Model:
y ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob
       Df Sum of Sq     RSS    AIC F value    Pr(>F)
<none>              1453068 503.93
M       1    296790 1749858 510.67  7.7615  0.008281 **
Ed      1    445788 1898855 514.51 11.6580  0.001534 **
Po1     1   1672038 3125105 537.93 43.7264 8.261e-08 ***
M.F     1    103159 1556227 505.16  2.6978  0.108740
U1      1    127044 1580112 505.87  3.3224  0.076217 .
U2      1    255443 1708511 509.55  6.6802  0.013714 *
Ineq    1    738244 2191312 521.24 19.3062 8.633e-05 ***
Prob    1    247978 1701046 509.34  6.4850  0.015053 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Forward variable selection

1. Start with model containing no explanatory variables, i.e. `lm(y ~ 1, data = dat)`.
2. For each variable in turn, investigate effect of adding variable from current model
3. Add the most informative/significant variable, unless this variable is not supplying significant information about the response
4. Go to step 2. Stop only if none of the non-included variables are important

Crime data: forward search using AIC

```
M0 = lm(y ~ 1, data = UScrime)  # Null model
M1 = lm(y ~ ., data = UScrime)  # Full model
step.fwd.aic = step(M0, scope = list(lower = M0, upper = M1),
                    direction = "forward", trace = FALSE)
summary(step.fwd.aic)
```

```
Call:
lm(formula = y ~ Po1 + Ineq + Ed + M + Prob + U2, data = UScrime)

Residuals:
    Min      1Q  Median      3Q     Max
-470.68  -78.41  -19.68  133.12  556.23

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5040.505    899.843  -5.602 1.72e-06 ***
Po1            11.502      1.375   8.363 2.56e-10 ***
Ineq            6.765      1.394   4.855 1.88e-05 ***
Ed             19.647      4.475   4.390 8.07e-05 ***
M              10.502      3.330   3.154  0.00305 **
Prob        -3801.836   1528.097  -2.488  0.01711 *
U2              8.937      4.091   2.185  0.03483 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 200.7 on 40 degrees of freedom
Multiple R^2:  0.7659,   Adjusted R^2:  0.7307
F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

The `add1()` function

- For a response variable `Y` and explanatory variables `x1, …, xk` stored in the data frame dat consider `M0 = lm(Y ~ 1, data = dat)`
- The R function `add1(M0, scope = ~x1+x2+...+xk, data=dat, test="F")` returns a number of information criteria for all variables specified after the option `scope = ~` to model the response variable
- Alternatively you can avoid listing all the variable names by coding up a full model `Mf = lm(Y ~ ., data = dat)` and then using `add1(M0, scope = Mf, data = dat, test = "F")`
- In particular, this includes the p-values of the F-test for $H_0 : \beta_j = 0$ versus $H_1 : \beta_j \neq 0$ for all $j \notin \alpha_{\text{M0}}$

Crime data: `add1()`

```
add1(step.fwd.aic, test = "F", scope = M1)
```

```
Single term additions

Model:
y ~ Po1 + Ineq + Ed + M + Prob + U2
       Df Sum of Sq     RSS    AIC F value Pr(>F)
<none>              1611057 504.79
So      1     17958 1593098 506.26  0.4396 0.5112
Po2     1     25017 1586040 506.05  0.6152 0.4376
LF      1     13179 1597878 506.40  0.3217 0.5739
M.F     1     30945 1580112 505.87  0.7638 0.3875
Pop     1     51320 1559737 505.26  1.2832 0.2642
NW      1       359 1610698 506.78  0.0087 0.9262
U1      1     54830 1556227 505.16  1.3741 0.2482
GDP     1     59910 1551147 505.00  1.5063 0.2271
Time    1      7159 1603898 506.58  0.1741 0.6788
```

```
step.fwd.aic$AIC = AIC(step.fwd.aic)
step.back.aic$AIC = AIC(step.back.aic)
models = list(
        "Forward Model" = step.fwd.aic,
        "Backwards Model" = step.back.aic)
```

```
modelsummary(models, output = 'kableExtra', escape = FALSE, gof_omit = 'BIC|F|
    L|RMSE', statistic = "{p.value}", stars = c('*' = .1, '**' = .05, '***' =
    0.01)) |>
  kable_styling(position="center", latex_options = "hold_position")
```

	Forward Model	Backwards Model
(Intercept)	−5040.505***	−6426.101***
	<0.001	<0.001
Po1	11.502***	10.265***
	<0.001	<0.001
Ineq	6.765***	6.133***
	<0.001	<0.001
Ed	19.647***	18.012***
	<0.001	0.002
M	10.502***	9.332***
	0.003	0.008
Prob	−3801.836**	−3796.032**
	0.017	0.015
U2	8.937**	18.735**
	0.035	0.014
M.F		2.234
		0.109
U1		−6.087*
		0.076
Num.Obs.	47	47
R2	0.766	0.789
R2 Adj.	0.731	0.744
AIC	640.2	639.3

* p < 0.1, ** p < 0.05, *** p < 0.01

Review: stepwise variable selection

1. Start with **some** model, typically null model (with no explanatory variables) or full model (with all variables)
2. For each variable in the current model, **investigate** effect of **removing** it
3. **Remove** the least informative variable, unless this variable is nonetheless supplying significant information about the response
4. For each variable not in the current model, **investigate** effect of **including** it
5. **Include** the most statistically significant variable not currently in model (unless no significant variable exists)
6. Go to step 2. Stop only if no change in steps 2-5

Exhaustive searches

Exhaustive searching the full set of possible models is totally feasible for $p < 100$.

```
library(leaps)
exh = regsubsets(y~., data = UScrime, nvmax = 15)
summary(exh)$outmat
```

```
          M   So  Ed  Po1 Po2 LF  M.F Pop NW  U1  U2  GDP Ineq Prob Time
1  ( 1 )  " " " " " " "*" " " " " " " " " " " " " " " " " "*"  " "  " "
2  ( 1 )  " " " " " " "*" " " " " " " " " " " " " " " " " "*"  " "  " "
3  ( 1 )  " " " " "*" "*" " " " " " " " " " " " " " " " " "*"  " "  " "
```

```
4  ( 1 )  "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  " "  " "
5  ( 1 )  "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  "*"  " "
6  ( 1 )  "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  "*"  " "
7  ( 1 )  "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  "*"  " "
8  ( 1 )  "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  "*"  " "
9  ( 1 )  "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  "*"  " "
10  ( 1 ) "*" " " " *" "*" " " " " " " " " " " " " " " " " " " " " " "*"  "*"  " "
11  ( 1 ) "*" " " " *" "*" "*" " " " " "*" "*" " " " " "*" "*" "*" "*"  "*"  " "
12  ( 1 ) "*" " " " *" "*" "*" " " " " "*" "*" "*" "*" "*" "*" "*" "*"  "*"  " "
13  ( 1 ) "*" " " " *" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"  "*"  " "
14  ( 1 ) "*" " " " *" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"  "*"  "*"
15  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"  "*"  "*"
```

30

Assessing performance

30.1 In sample performance

In sample performance vs out of sample performance

In sample

- e.g. r^2 comparing the simple linear regression to the full model

```
summary(lm2)$r.squared # lozone ~ temperature
```

```
[1] 0.5547615
```

```
summary(lm3)$r.squared # lozone ~ radiation + temperature + wind
```

```
[1] 0.664515
```

- Doesn't protect against over fitting

Out of sample

- How well do we predict observations that we didn't use to build the model?

Out of sample performance

Comparing simple linear regression with the full model

Out of sample performance

We could think about building a **training** set and using it to predict observations from a **test** set.

```
n = nrow(environmental)
n
```

```
[1] 111
```

```
n_train = floor(0.8*n)
n_test = n - n_train
grp_labs = rep(c("Train","Test"), times = c(n_train, n_test))
environmental$grp = sample(grp_labs)
```

```
train_dat = environmental |> filter(grp == "Train")
lm_simple_train = lm(lozone ~ temperature, data = train_dat)
lm_full_train = lm(lozone ~ radiation + temperature + wind, data = train_dat)
test_dat = environmental |> filter(grp == "Test")
simple_pred = predict(lm_simple_train, newdata = test_dat)
full_pred = predict(lm_full_train, newdata = test_dat)
```

Root mean square error How can we compare the predictions from the two models? Compare them to the observed values using the root mean square error:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

```
simple_mse = mean((test_dat$lozone - simple_pred)^2)
sqrt(simple_mse)
```

```
[1] 0.5129889
```

```
full_mse = mean((test_dat$lozone - full_pred)^2)
sqrt(full_mse)
```

```
[1] 0.435352
```

Mean absolute error
An alternative measure of performance, less influenced by outliers is the **mean absolute error**,

$$\text{MAE} = \frac{\sum_{i=1}^{m}|y_i - \hat{y}_i|}{m}$$

```
simple_mae = mean(abs(test_dat$lozone - simple_pred))
simple_mae
```

```
[1] 0.4067837
```

```
full_mae = mean(abs(test_dat$lozone - full_pred))
full_mae
```

```
[1] 0.3661988
```

30.2 Cross-validation

Out of sample performance

***k*-fold cross-validation (CV) estimation**

- Data randomly divided into k subsets of (nearly) equal size
- Estimate your model by leaving one subset out
- Use your estimated model to predict the observations left out
- Compute error rates on the left out set
- Repeat k times (for each of the subsets) Average the error rate over the k runs

Bias-variance tradeoff: smaller k can give larger bias but smaller variance

10-fold cross validation

Step 1: divide our data up into 10 folds there are 111 observations, so we have 9 folds of 11 observations and 1 fold of 12 observations.

```
set.seed(88)
nrow(environmental)
```

```
[1] 111
```

```
environmental$grp = NULL # remove the grp variable we added previously
fold_id = c(1, rep(1:10, each = 11))
environmental$fold_id = sample(fold_id, replace = FALSE)
head(environmental)
```

```
  ozone radiation temperature wind   lozone fold_id
1    41       190          67  7.4 3.713572       5
2    36       118          72  8.0 3.583519       1
3    12       149          74 12.6 2.484907       8
4    18       313          62 11.5 2.890372       5
5    23       299          65  8.6 3.135494      10
6    19        99          59 13.8 2.944439      10
```

Step 2: estimate the model leaving one fold out, make predictions on the test set and calculate the error rate

```
k = 10
simple_mse = full_mse = vector(mode = "numeric", length = k)
simple_mae = full_mae = vector(mode = "numeric", length = k)
for(i in 1:k) {
  test_set = environmental[fold_id == i,]
  training_set = environmental[fold_id != i,]
  simple_lm = lm(lozone ~ temperature, data = training_set)
  simple_pred = predict(simple_lm, test_set)
  simple_mse[i] = mean((test_set$lozone - simple_pred)^2)
  simple_mae[i] = mean(abs(test_set$lozone - simple_pred))
  full_lm = lm(lozone ~ radiation + temperature + wind, data = training_set)
  full_pred = predict(full_lm, test_set)
  full_mse[i] = mean((test_set$lozone - full_pred)^2)
  full_mae[i] = mean(abs(test_set$lozone - full_pred))
}
```

Step 3: aggregate the errors over the 10 folds

```
cv_res = tibble(simple_mse, full_mse,
                simple_mae, full_mae)
cv_res
```

```
# A tibble: 10 x 4
   simple_mse full_mse simple_mae full_mae
        <dbl>    <dbl>      <dbl>    <dbl>
 1     0.437    0.400       0.527    0.552
 2     0.955    0.839       0.784    0.722
 3     0.307    0.244       0.489    0.418
 4     0.348    0.194       0.396    0.297
 5     0.176    0.134       0.360    0.313
 6     0.376    0.190       0.471    0.372
 7     0.389    0.260       0.486    0.395
 8     0.0879   0.0783      0.249    0.206
 9     0.160    0.0975      0.322    0.284
10     0.252    0.251       0.414    0.397
```

Find the averages:

```
cv_sum_res = cv_res |>
  summarise(
    across(.cols = everything(),
           mean)
  )
cv_sum_res |> t()
```

```
                [,1]
simple_mse 0.3489454
full_mse   0.2688823
simple_mae 0.4498899
full_mae   0.3957052
```

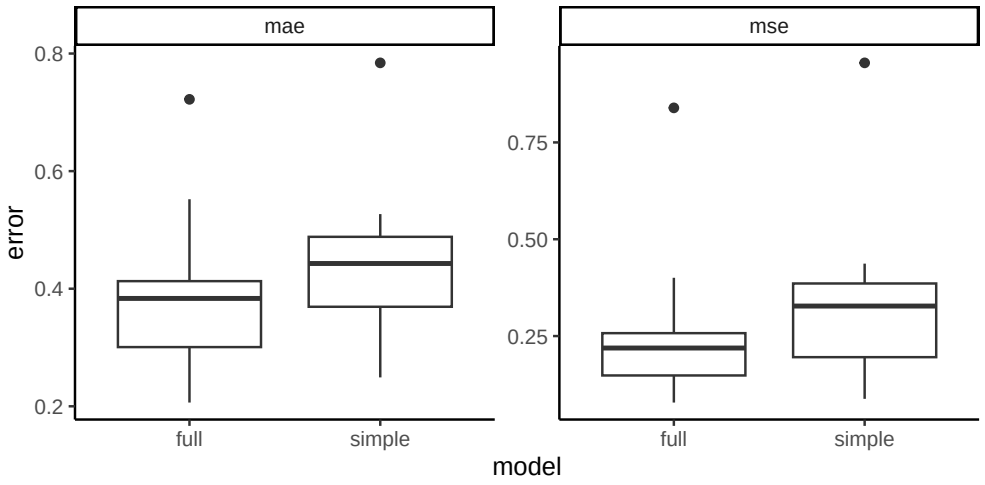
Common to report the root mean square errors:

```
sqrt(cv_sum_res[,1:2])
```

```
# A tibble: 1 x 2
  simple_mse full_mse
       <dbl>    <dbl>
1      0.591    0.519
```

We could visualise the error rates for each of the 10 folds:

```
cv_res |> gather(key = "metric", value = "error") |>
  separate(col = metric, into = c("model","metric")) |>
  ggplot(aes(x = model, y = error)) + facet_wrap(~metric, scales = "free_y") +
  geom_boxplot() +
  theme_classic()
```



`caret` (Classification And REgression Training)

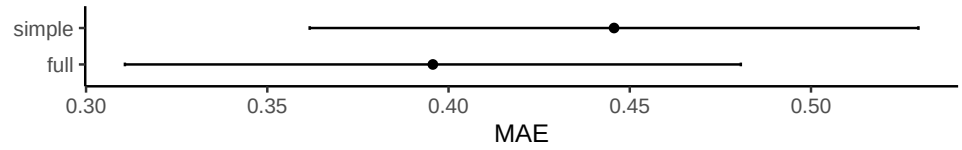
```
library(caret)
tr_ctrl = trainControl(
    method = "cv", number = 10,
    verboseIter = FALSE
  )

cv_full = train(
  lozone ~ radiation + temperature + wind, environmental,
  method = "lm",
  trControl = tr_ctrl
)
cv_full
```

```
Linear Regression

111 samples
  3 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 99, 101, 100, 100, 100, 100, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.5010752  0.7026242  0.395675

Tuning parameter 'intercept' was held constant at a value of TRUE
```

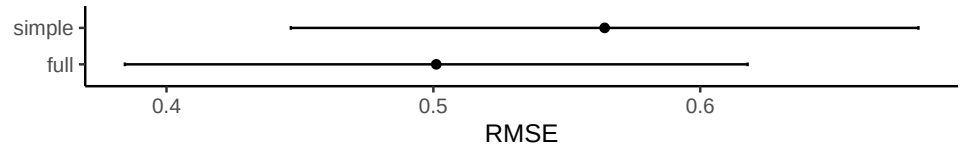
```
cv_simple = train(
  lozone ~ temperature,
  environmental,
  method = "lm",
  trControl = tr_ctrl
)
cv_simple
```

```
Linear Regression

111 samples
  1 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 99, 101, 100, 99, 100, 99, ...
Resampling results:

  RMSE       Rsquared  MAE
  0.5641813  0.584467  0.4456647

Tuning parameter 'intercept' was held constant at a value of TRUE
```

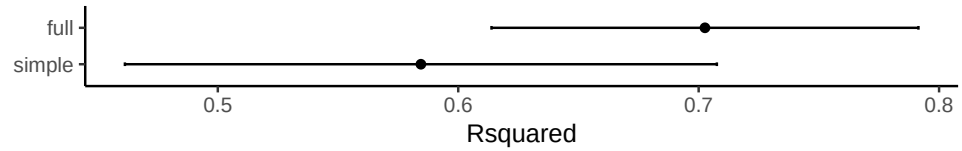
```
results ← caret::resamples(list(simple = cv_simple,  full = cv_full))
ggplot(results, metric = "MAE") +
  labs(y = "MAE") +
  theme_classic()
```




```
ggplot(results, metric = "RMSE") +
  labs(y = "RMSE") +
  theme_classic()
```




```
ggplot(results, metric = "Rsquared") +
  labs(y = "Rsquared") +
  theme_classic()
```



31

Logistic regression

31.1 Logistic regression

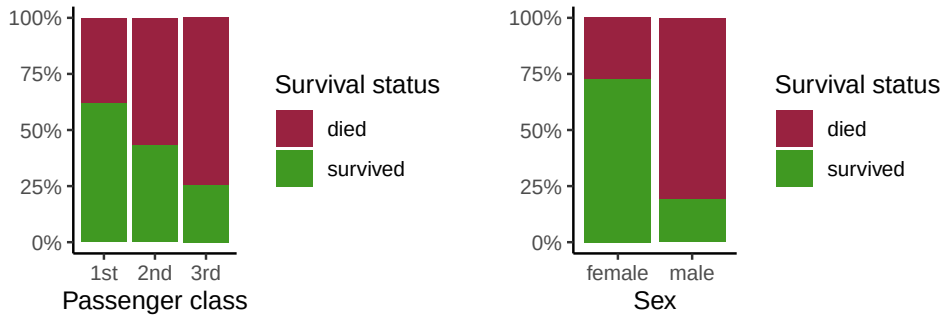
Titanic survival

Data on passengers on the RMS Titanic, excluding the crew and some individual identifier variables.

```
data("Titanicp", package = "vcdExtra")
glimpse(Titanicp)
```

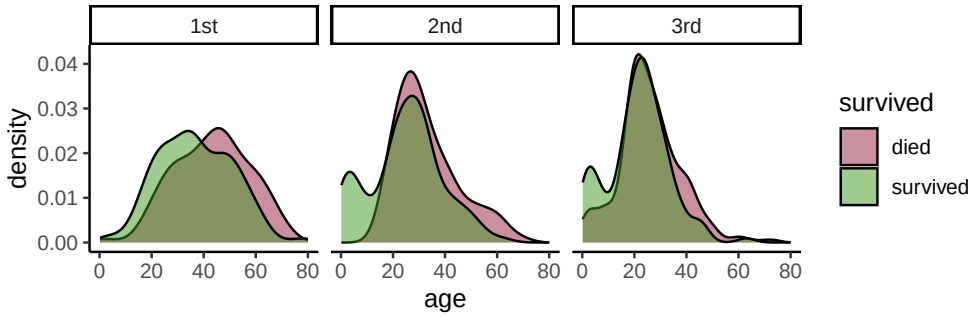
```
Rows: 1,309
Columns: 6
$ pclass   <fct> 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1
   ~
$ survived <fct> survived, survived, died, died, died, survived, survived, die
   ~
$ sex      <fct> female, male, female, male, female, male, female, male, femal
   ~
$ age      <dbl> 29.0000, 0.9167, 2.0000, 30.0000, 25.0000, 48.0000, 63.0000,
   ~
$ sibsp    <dbl> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1
   ~
$ parch    <dbl> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1
   ~
```

```
t1 = Titanicp |> group_by(survived, pclass) |> count() |>
  ggplot(aes(x = pclass, y = n, fill = survived)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "", x = "Passenger class", fill = "Survival status") +
  scale_fill_manual(values=c("#992240", "#409922")) +
  theme_classic()
t2 = Titanicp |> group_by(survived, sex) |> count() |>
  ggplot(aes(x = sex, y = n, fill = survived)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "", x = "Sex", fill = "Survival status") +
  scale_fill_manual(values=c("#992240", "#409922")) +
  theme_classic()
plot_grid(t1,t2)
```




```
Titanicp |> ggplot() +
  aes(x = age, fill = survived) +
  geom_density(alpha = 0.5) +
```

```
facet_grid(~pclass) +
scale_fill_manual(values=c("#992240", "#409922")) +
theme_classic()
```



It seems clear that there is some sort of a relationship between survival, sex, class and perhaps even age.

How do we model this?!

Linear regression

For linear regression we have

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

- Conditional on the vector of predictor variables ($\mathbf{x}$'s), the dependent variable Y also follows a normal distribution, i.e.

$$Y_i \mid \mathbf{x}_i \sim \mathcal{N}(\mathbf{x}_i'\boldsymbol{\beta}, \sigma^2).$$

- If the dependent variable Y is binary, i.e. $Y_i \in \{0, 1\}$, a linear regression doesn't work so well, and we typically to use **logistic regression** instead.

Modelling binary data

We need to think what sort of a (conditional) distribution for $Y_i \mid \mathbf{x}_i$ makes more sense for binary data.
Since Y_i is either 0 or 1 it is natural to model it as a Bernoulli random variable,

$$Y_i \mid \mathbf{x}_i \sim \text{Bernoulli}(p(\mathbf{x}_i, \boldsymbol{\beta}))$$

where the probability that $Y_i = 1$ is given by some function of our predictors, $p(\mathbf{x}_i, \boldsymbol{\beta})$.
We need the probability $p(\mathbf{x}_i, \boldsymbol{\beta})$ to be in $[0, 1]$ and for it to depend on the lienar combination of our predictors $\mathbf{x}_i'\boldsymbol{\beta}$ in some way.
A common choice is the **logistic function**,

$$p(\mathbf{x}_i\boldsymbol{\beta}) = \frac{\exp(\mathbf{x}_i'\boldsymbol{\beta})}{1 + \exp(\mathbf{x}_i'\boldsymbol{\beta})}.$$

Logistic regression

- A logistic regression model begins with,

$$Y_i \mid \mathbf{x}_i \sim \text{Bernoulli}\left(\frac{\exp(\mathbf{x}_i'\boldsymbol{\beta})}{1 + \exp(\mathbf{x}_i'\boldsymbol{\beta})}.\right)$$

- If we had a new observation vector $\mathbf{x}_0$ and we knew the β vector, we could calculate the probability that the corresponding $Y = 1$:

$$P(Y = 1 \mid \mathbf{x}_0) = \frac{\exp(\mathbf{x}_i'\beta)}{1 + \exp(\mathbf{x}_i'\beta)}.$$

- If this probability is greater than 0.5, we would make the prediction $\hat{Y} = 1$, otherwise we would make the prediction $\hat{Y} = 1$ otherwise we would predict $\hat{Y} = 0$.
- BUT we don't know β, we need to estimate the coefficient vector.
- Estiamting $\hat{\beta}$ can be done using iteratively reweighted least squares.

Odds

We introduced **odds** in section 6, they are an alternative way of quantifying the probability of an event.
For some event E,

$$\text{odds}(E) = \frac{P(E)}{1 - P(E)}.$$

If we are told the odds of E are a to b, then

$$\text{odds(E)} = \frac{a}{b} = \frac{\frac{a}{(a+b)}}{\frac{b}{(a+b)}}$$

which implies $P(E) = \frac{a}{(a+b)}$.
Odds feature in **logistic regression**.

Titanic

- Start by converting survival to 0/1 (numeric) variable

```
x = Titanicp |> mutate(survived = ifelse(survived == "survived", 1, 0))
glimpse(x)
```

```
Rows: 1,309
Columns: 6
$ pclass   <fct> 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1
    st, 1~
$ survived <dbl> 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
     0, 1~
$ sex      <fct> female, male, female, male, female, male, female, male,
    femal~
$ age      <dbl> 29.0000, 0.9167, 2.0000, 30.0000, 25.0000, 48.0000,
    63.0000, ~
$ sibsp    <dbl> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
     0, 1~
$ parch    <dbl> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
     0, 1~
```

- We treat `survived` and `died` as successes and failures from a Bernoulli (binomial) distribution where the probability of success is given by a transformation of a linear model of the predictors.

Fit a logistic regression model

```
glm1 = glm(survived ~ pclass + sex + age, family = binomial, data = x)
summary(glm1)
```

```
Call:
glm(formula = survived ~ pclass + sex + age, family = binomial,
    data = x)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.522074   0.326702  10.781  < 2e-16 ***
pclass2nd   -1.280570   0.225538  -5.678 1.36e-08 ***
pclass3rd   -2.289661   0.225802 -10.140  < 2e-16 ***
sexmale     -2.497845   0.166037 -15.044  < 2e-16 ***
age         -0.034393   0.006331  -5.433 5.56e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1414.62  on 1045  degrees of freedom
Residual deviance:  982.45  on 1041  degrees of freedom
  (263 observations deleted due to missingness)
AIC: 992.45

Number of Fisher Scoring iterations: 4
```

Checking for significance

Before we start to interpret our model and make predictions, we might want to know if we can drop any of the variables from the model. Like in the linear regression context, this is equivalent to testing $H_0 : \beta = 0$ against the alternative $H_0 : \beta \neq 0$.
Example, let's test if the coefficient for age is significantly different to zero.

- **Hypotheses**: $H_0 : \beta_{\text{age}} = 0$ vs $H_1 : \beta_{\text{age}} \neq 0$
- **Test statistic**: $T = \frac{\hat{\beta}_{\text{age}} - \beta_{\text{age}}}{\text{SE}(\hat{\beta}_{\text{age}})} \sim \mathcal{N}(0, 1)$
- **Observed test statistic**: $t_0 = \frac{-0.034393}{0.006331} = -5.433$
- **p-value**: $2P(T \geqslant |t_0|) = 2P(Z \geqslant 5.433) < 0.0001$
- **Conclusion**: the p-value is very small (much smaller than 0.05) therefore we reject the null hypothesis and conclude that age is a significant predictor for survival.

Write down the fitted model

```
glm1
```

```
Call:  glm(formula = survived ~ pclass + sex + age, family = binomial,
    data = x)

Coefficients:
(Intercept)    pclass2nd    pclass3rd      sexmale          age
    3.52207     -1.28057     -2.28966     -2.49784     -0.03439

Degrees of Freedom: 1045 Total (i.e. Null);  1041 Residual
  (263 observations deleted due to missingness)
Null Deviance:      1415
Residual Deviance: 982.5        AIC: 992.5
```

The fitted model is,

$$\text{logit}(p) = 3.5 - 1.3 \text{ pclass2nd} - 2.3 \text{ pclass3rd} - 2.5 \text{ sexmale} - 0.03 \text{ Age}$$

What's this logit function?

The **logit** function is our **link** from a linear combination of the predictors to the probability of the outcome being equal to 1.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

- It's the log-odds!
- Our estimated coefficients are therefore interpreted as changes in the log-odds.
- I.e. we can write out fitted model as:

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3\ \text{pclass2nd} - 2.3\ \text{pclass3rd} - 2.5\ \text{sexmale} - 0.03\ \text{Age}$$

Interpreting our coefficients

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3\ \text{pclass2nd} - 2.3\ \text{pclass3rd} - 2.5\ \text{sexmale} - 0.03\ \text{Age}$$

- **Intercept**: the log-odds of survival for an individual travelling in 1st class who is female and aged zero years old.
- Holding sex and age constant, the `pclass2nd` coefficient represents the **difference** in the log-odds between someone travelling in 1st class and someone travelling in 2nd class. It's **negative**, so we're saying that your odds of survival were lower if you travelled in second class, relative to those who travelled in first class.
- Holding class and age constant, the `sexmale` coefficient represents the **difference** in the log-odds between males and females. It is **negative**, so if you were a male, your odds of survival were **lower** than if you were a female.
- The `age` coefficient is also negative, which implies that older people had lower odds of survival than younger people. Specifically, on average, for each additional year older you are, the log-odds of survival decreased by 0.03, holding class and sex constant.

What do our predictions mean?

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3\ \text{pclass2nd} - 2.3\ \text{pclass3rd} - 2.5\ \text{sexmale} - 0.03\ \text{Age}$$

We can predict the log-odds for a newborn male travelling in first class:

- `pclass2nd = 0, pclass3rd = 0, sexmale = 1, age = 0`

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3 \times 0 - 2.3 \times 0 - 2.5 \times 1 - 0.03 \times 0 = 3.5 - 2.5 = 1$$

The log-odds of survival for a newborn male travelling in first class is estimated to be 1.

```
new_data = data.frame(pclass = "1st", sex = "male", age = 0)
predict(glm1, newdata = new_data, type = "link")
```

```
       1
1.024229
```

Can we work out the estimated probability of survival for a newborn male travelling in first class?

$$\begin{aligned}
\log\left(\frac{p}{1-p}\right) &= 1 \\
\left(\frac{p}{1-p}\right) &= \exp(1) \\
p &= \exp(1) - p\exp(1) \\
p + p\exp(1) &= \exp(1) \\
p &= \frac{\exp(1)}{1+\exp(1)} \approx 0.73
\end{aligned}$$

```
new_data = data.frame(pclass = "1st", sex = "male", age = 0)
predict(glm1, newdata = new_data, type = "response")
```

```
        1
0.7357956
```

Note that we've used the **logistic** function to transform back to obtain an estimate of the **probability** (from the output of our model which is an estimate of the log-odds).

Outputting your model coefficients

```
models = list(glm1,glm1)
modelsummary(models, output = 'kableExtra', gof_omit = 'AIC|BIC|L|RMSE',
    estimate = "{estimate} [{conf.low}, {conf.high}]", exponentiate = c(FALSE,
     TRUE), statistic = c("{p.value}"), stars = c('*' = .1, '**' = .05, '***'
    = 0.01)) |>
  kable_styling(position="center", latex_options = "hold_position")
```

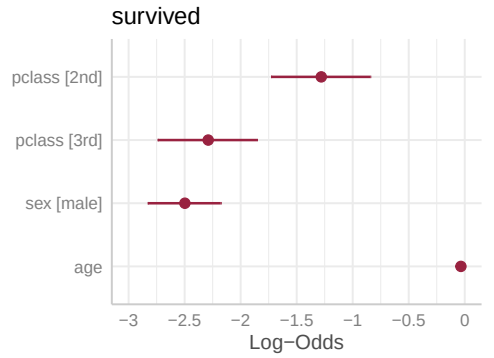
	(1)	(2)
(Intercept)	3.522 [2.897, 4.179]	33.855 [18.115, 65.274]
	<0.001	<0.001
pclass2nd	−1.281 [−1.728, −0.843]	0.278 [0.178, 0.430]
	<0.001	<0.001
pclass3rd	−2.290 [−2.741, −1.855]	0.101 [0.065, 0.157]
	<0.001	<0.001
sexmale	−2.498 [−2.829, −2.178]	0.082 [0.059, 0.113]
	<0.001	<0.001
age	−0.034 [−0.047, −0.022]	0.966 [0.954, 0.978]
	<0.001	<0.001
Num.Obs.	1046	1046

* p < 0.1, ** p < 0.05, *** p < 0.01

Visualising your model coefficients

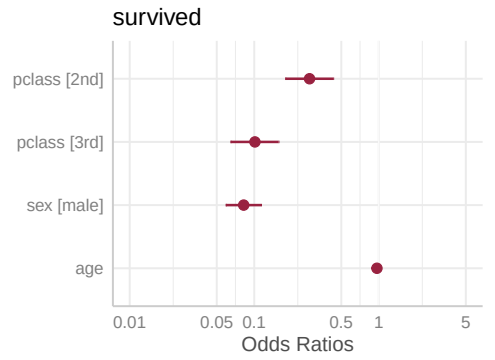
Log-odds scale

```
library(sjPlot)
plot_model(glm1, transform = NULL,
    colors = "#992240") + theme_sjplot
    ()
```



Odds scale

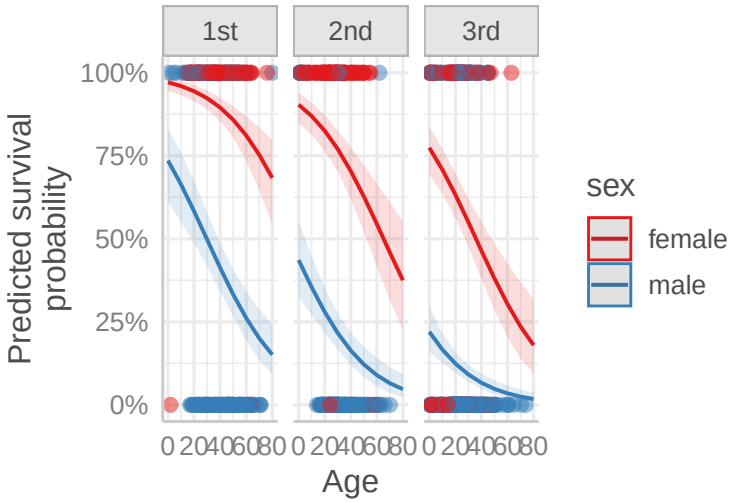
```
plot_model(glm1, colors = "#992240") +
    theme_sjplot()
```



Note the coefficient for age is on a different scale to the categorical variables.

Visualising predictions

```
plot_model(glm1, type = "pred", terms = c("age", "sex", "pclass"), show.data =
    TRUE) +
  labs(title = "", y = "Predicted survival\nprobability", x = "Age", colours =
      c("#992240","#224099")) +
  theme_sjplot()
```



31.2 Evaluating performance

Making predictions

We can make predictions by rounding our predicted probability to 0 or 1.

```
x = x %>% drop_na() %>%
  mutate(pred_prob = predict(glm1, type = "response"),
         pred_surv = round(pred_prob))
head(x, n = 10)
```

```
   pclass survived    sex     age sibsp parch pred_prob pred_surv
1     1st        1 female 29.0000     0     0 0.9258533         1
2     1st        1   male  0.9167     1     2 0.7296211         1
3     1st        0 female  2.0000     1     2 0.9693290         1
4     1st        0   male 30.0000     1     2 0.4981081         0
5     1st        0 female 25.0000     1     2 0.9347616         1
6     1st        1   male 48.0000     0     0 0.3482715         0
7     1st        1 female 63.0000     1     0 0.7949948         1
8     1st        0   male 39.0000     0     0 0.4213810         0
9     1st        1 female 53.0000     2     0 0.8454345         1
10    1st        0   male 71.0000     0     0 0.1950240         0
```

Evaluating (in-sample) performance

How many passengers did we correctly classify? **Resubstitution error rate** The resubstitution error rate is the proportion of observations we predict **incorrectly** when we try to predict all the points we used to fit the model.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i \neq \hat{y}_i)$$

```
mean(x$survived != x$pred_surv)
```

```
[1] 0.2151052
```

We failed to correctly classify 21.5% of the observations.

Confusion matrix

We can examine how our model predicted all the data points, using `confusionMatrix` from the **caret** package. Note, that we are required to put in factor inputs.

```
library(caret)
confusion.glm = confusionMatrix(
  data = as.factor(x$pred_surv),
  reference = as.factor(x$survived))
confusion.glm$table
```

```
          Reference
Prediction   0   1
         0 520 126
         1  99 301
```

Looking at the table output and reading vertically, we can assess model performance.

- Out of the 520+99=619 deaths in our data set, the model successfully predicts 520.
- Out of the 126+301=427 survivors in our data set, the model correctly predicts 301.

```
confusion.glm
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 520 126
         1  99 301

               Accuracy : 0.7849
                 95% CI : (0.7587, 0.8094)
    No Information Rate : 0.5918
```

```
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5504

 Mcnemar's Test P-Value : 0.08304

            Sensitivity : 0.8401
            Specificity : 0.7049
         Pos Pred Value : 0.8050
         Neg Pred Value : 0.7525
             Prevalence : 0.5918
         Detection Rate : 0.4971
   Detection Prevalence : 0.6176
      Balanced Accuracy : 0.7725

       'Positive' Class : 0
```

The **accuracy** is 1 minus the resubstitution error rate. Some of the other performance metrics will be familiar to you from module 1.

- sensitivity
- specificity
- positive predictive value
- negative predictive value

Evaluating out of sample performance

- Often, we want to see how well our model can predict new data points. However, it is often impossible to get completely new data.
- Like with linear regression we can perform k-fold cross validation to evaluate out of sample performance.
- We split our data into training and testing sets to evaluate performance, treating the testing data as new data points.
- For k-fold CV, we split our data into k-folds. The first fold is treated as a testing set, and the method is fit on the remaining $k - 1$ folds.
- The misclassification error rate is then computed on the observations in the held-out fold.
- This procedure is repeated k times; each time, a different group of observations is treated as a testing set.
- The **CV error rate** is then calculated as the average of these k error rates.

How many folds?

- Larger k can take longer to run (computationally more expensive)
- Larger k means each model is trained on more data which should lead to a *lower* prediction error.
- If k is too small you might not have enough training observations to build a sensible model
- Lower k means that there's more of a chance of the test set being "different" in some way to the training set, which may lead to higher prediction errors.

Cross validation for Titanic data

```
x_full = x %>%
  drop_na() %>%
  select(pclass, survived,
         sex, age)
nrow(x_full)
```

```
[1] 1046
```

```
nrow(x_full)/5
```

```
[1] 209.2
```

```
fold_id = c(1, rep(1:5, each = 209))
table(fold_id)
```

```
fold_id
  1   2   3   4   5
210 209 209 209 209
```

```
x_full$fold_id = sample(
  fold_id,
  replace = FALSE)
head(x_full)
```

```
  pclass survived    sex     age fold_id
1    1st        1 female 29.0000       1
2    1st        1   male  0.9167       3
3    1st        0 female  2.0000       2
4    1st        0   male 30.0000       2
5    1st        0 female 25.0000       1
6    1st        1   male 48.0000       4
```

```
cv_error = vector("numeric", length = 5)
for(j in 1:5){
  train = x_full %>% filter(fold_id != j)
  fit = glm(survived ~ pclass + sex + age, data = train)
  test = x_full %>% filter(fold_id == j)
  pred = round(predict(fit, newdata = test, type = "response"))
  cv_error[j] = mean(pred != test$survived)
}
cv_error
```

```
[1] 0.1619048 0.2535885 0.2200957 0.2775120 0.1818182
```

```
mean(cv_error)
```

```
[1] 0.2189838
```

```
1-mean(cv_error) # accuracy
```

```
[1] 0.7810162
```

Cross validation for Titanic data using caret

```
train(factor(survived) ~ pclass + sex + age,
      data = x_full,
      method = "glm",
      family = "binomial",
      trControl = trainControl(
        method = "cv", number = 5,
        verboseIter = FALSE
      ))
```

```
Generalized Linear Model

1046 samples
   3 predictor
   2 classes: '0', '1'
```

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 837, 837, 836, 837, 837
Resampling results:

  Accuracy   Kappa
  0.7839508  0.548324
```

32

Decision trees and random forests

32.1 Decision trees

Decision trees

A decision tree determines the predicted outcome based on series of questions and conditions.

How to build a tree?

- What features do we make our decisions on?
- What is the threshold for classifying each question into a **yes** or **no** answer?

Consider the `iris` dataset

```
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.
    ~
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.
    ~
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.
    ~
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.
    ~
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s
    ~
```

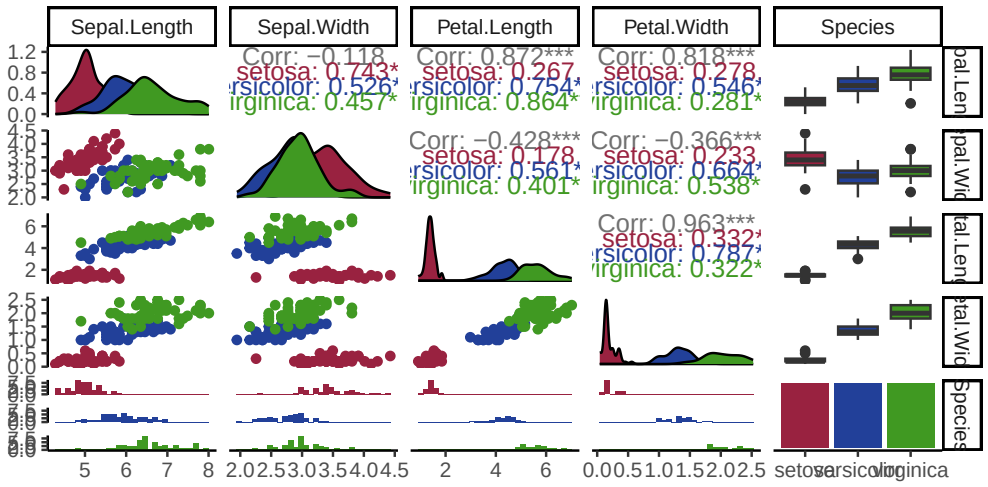
```
table(iris$Species)
```

```
    setosa versicolor  virginica
        50         50         50
```

How can we construct a decision tree to determine the `Species` of `iris` given petal and sepal measurements?

Consider the `iris` dataset

```
ggplot <- function(...) ggplot2::ggplot(...) + scale_color_manual(values=c("
    #992240","#224099","#409922")) + scale_fill_manual(values=c("#992240","
    #224099","#409922"))
unlockBinding("ggplot",parent.env(asNamespace("GGally")))
assign("ggplot",ggplot,parent.env(asNamespace("GGally")))
ggpairs(iris, mapping = aes(col = Species)) + theme_classic()
```



Iris tree

- We can use the **rpart** package (recursive partitioning) to build our decision tree.
- It uses a formula structure, much like `lm()` and `glm()` to identify the dependent variable (what we're trying to predict) and the explanatory variables (what information do we have available to help prediction).

```
library(rpart)
tree = rpart(Species ~ ., data = iris, method = "class")
```

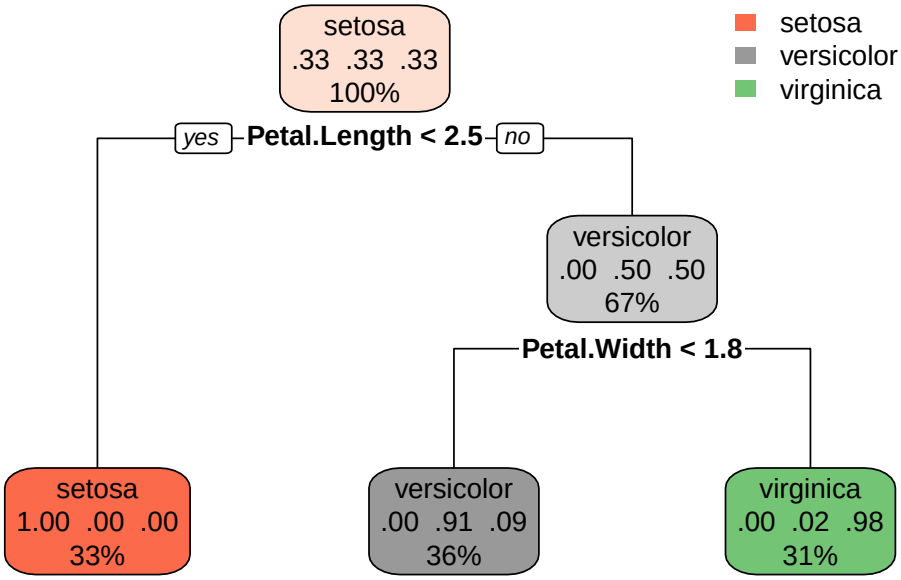
- The formula `Species ~.` in the code above says we want to predict `Species` using all the variables in the data frame `iris`.
- Specifying `method = "class"` means we want to build a classification (decision) tree.

```
library(rpart)
tree = rpart(Species ~ ., data = iris, method = "class")
tree
```

```
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
  2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259)
      *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

Visualising the tree

```
tree = rpart(
  Species ~ .,
  data = iris,
  method = "class")

library(rpart.plot)
rpart.plot(tree, extra = 104)
```

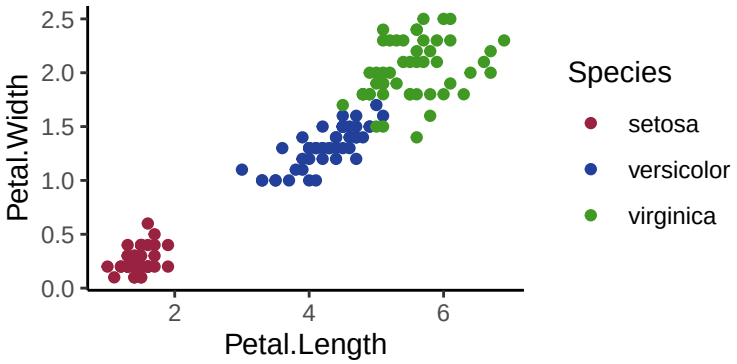


- Using the `rpart.plot()` function from the rpart.plot library, we can visualise the results of our classification tree.
- Each node shows the predicted class, the predicted probability of each class, and the percentage of observations in each node.

How does it work?

In this tree, we only need to consider two variables.

```
p1 = iris |> ggplot() +
  aes(x = Petal.Length,
      y = Petal.Width,
      colour = Species) +
  geom_point() +
  scale_color_manual(values=c("#992240","#224099","#409922")) +
  theme_classic()
p1
```

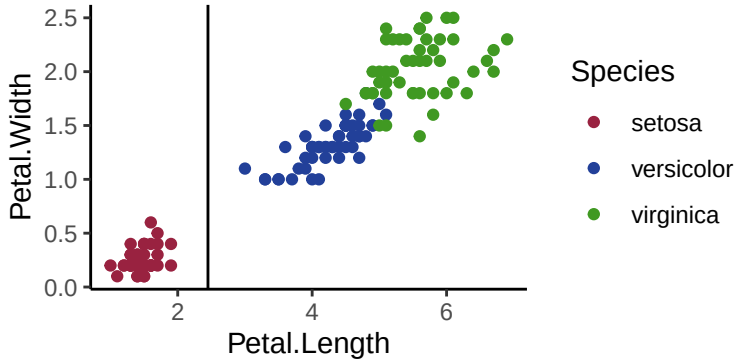


The first branch is done to "best" split the data to create the most "pure" (homogenous) partitions.

`Petal.Length < 2.45`

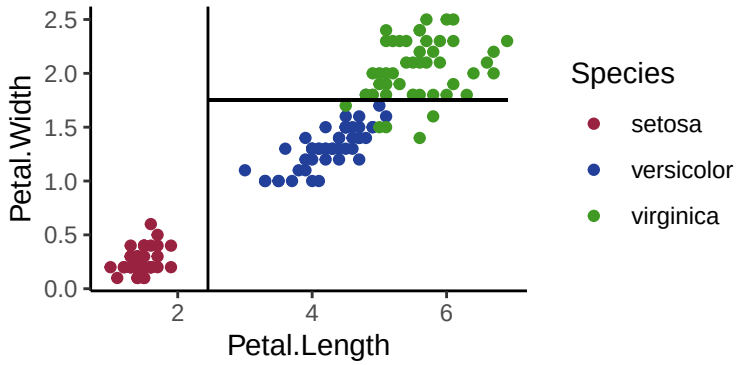
```
p2 = p1 +
  geom_vline(
    aes(xintercept=2.45)
  )
```

```
p2
```



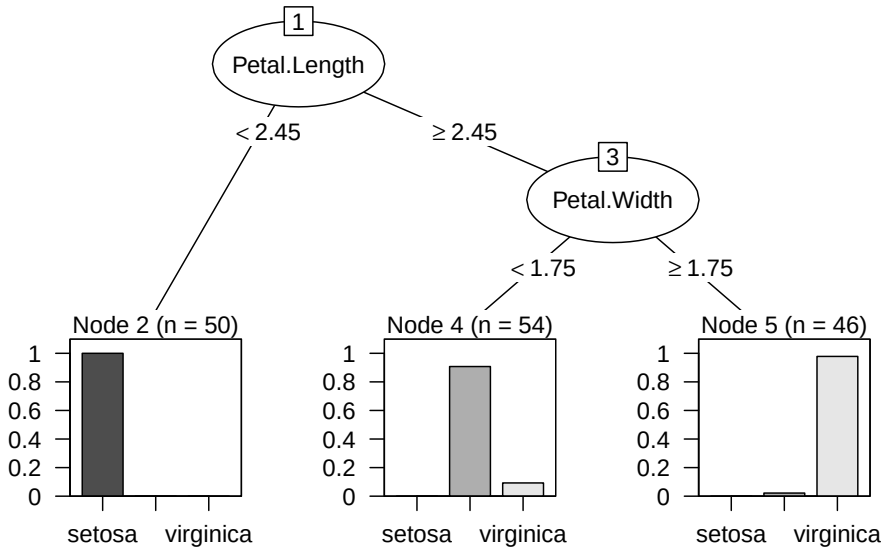
The next branch applies to observations that have `Petal.Length > 2.45` and it tries to find the next best split of the data.
`Petal.width < 1.75`

```
p3 = p2 +
  geom_segment(
    aes(x = 2.45, y = 1.75,
        xend = 6.9, yend = 1.75),
        colour = "black"
  )

p3
```



Alternative visualisation with partykit

```
library(partykit)
plot(as.party(tree))
```



Making a prediction

This was our fitted model:

```
tree ← rpart(Species ~ ., data = iris, method = "class")
```

Using `predict()`, we can predict the class of a new data point, much like for `lm()` and `glm()` objects.

```
new_data = data.frame(Sepal.Length = c(5.0, 5.0),
                      Sepal.Width = c(3.9, 3.9),
                      Petal.Length = c(1.4, 3.4),
                      Petal.Width = c(0.3,0.3),
                      Species = c(NA,NA))
new_data
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1            5         3.9          1.4         0.3      NA
2            5         3.9          3.4         0.3      NA
```

```
predict(tree, new_data, type = "class")
```

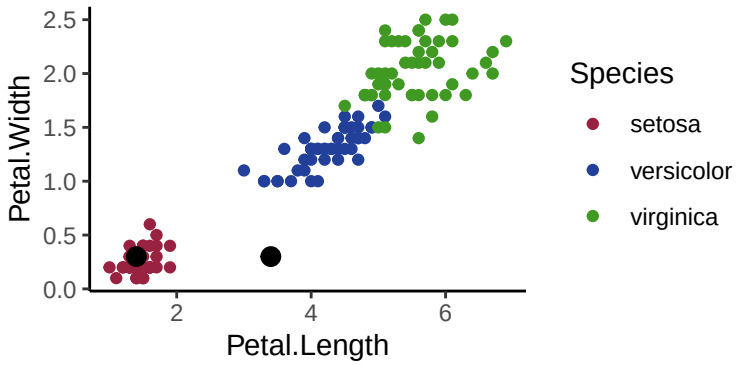
```
         1          2
    setosa versicolor
Levels: setosa versicolor virginica
```

Does it seem reasonable?

```
predict(tree, new_data, type = "class")
```

```
         1          2
    setosa versicolor
Levels: setosa versicolor virginica
```

```
p1 + geom_point(data = new_data, size = 3, colour = "black")
```



Assessing in sample performance in the iris data

```
predicted_species = predict(tree, type = "class")
confusionMatrix(data = predicted_species, reference = iris$Species)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         49         5
  virginica       0          1        45

Overall Statistics

               Accuracy : 0.96
                 95% CI : (0.915, 0.9852)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.94

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9800           0.9000
Specificity                 1.0000            0.9500           0.9900
Pos Pred Value              1.0000            0.9074           0.9783
Neg Pred Value              1.0000            0.9896           0.9519
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3267           0.3000
Detection Prevalence        0.3333            0.3600           0.3067
Balanced Accuracy           1.0000            0.9650           0.9450
```

Model selection

- How did our tree know to know to only use two splits?
- There is a **complexity parameter** that can be used to determine if a proposed new split *sufficiently* improves the predictive power or not.
- A choice is made whether to keep or "prune" a proposed new branch.
- Default is that a branch should decrease the error by 1%.
- This helps to avoid **overfitting**.

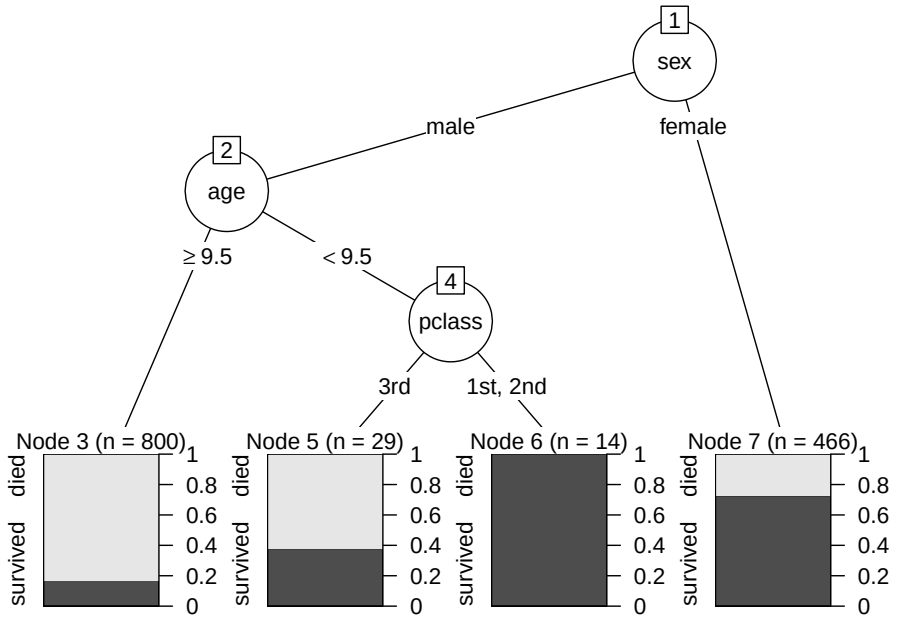
Titanic tree

```
data("Titanicp", package = "vcdExtra")
```

```
titanic_tree = rpart(survived ~ sex + age + pclass, data = Titanicp, method =
    "class")
titanic_tree
```

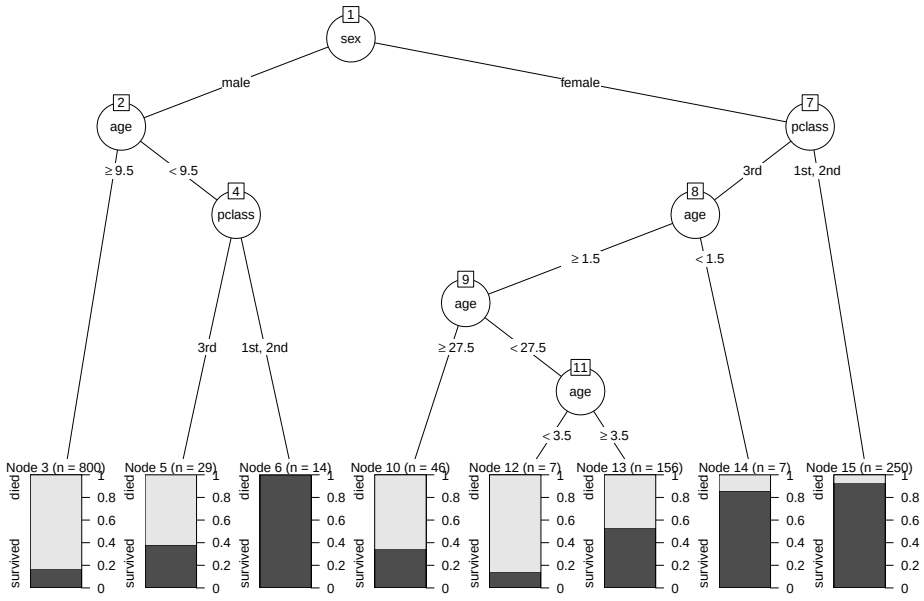
```
n= 1309

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 1309 500 died (0.6180290 0.3819710)
   2) sex=male 843 161 died (0.8090154 0.1909846)
     4) age>=9.5 800 136 died (0.8300000 0.1700000) *
     5) age< 9.5 43  18 survived (0.4186047 0.5813953)
      10) pclass=3rd 29  11 died (0.6206897 0.3793103) *
      11) pclass=1st,2nd 14   0 survived (0.0000000 1.0000000) *
   3) sex=female 466 127 survived (0.2725322 0.7274678) *
```

```
plot(as.party(titanic_tree))
```



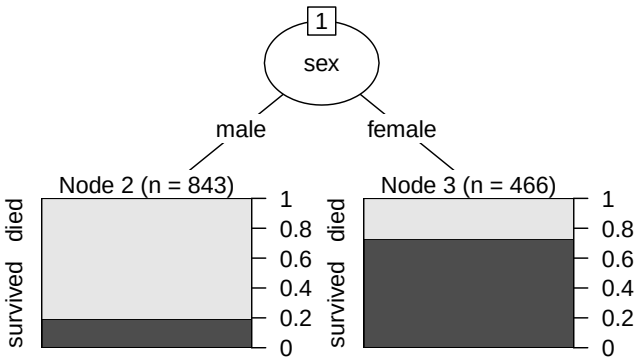
What if we in lower the complexity parameter threshold, so that each new branch only needs to decrease the error by 0.9%?

```
titanic_tree0.9 = rpart(survived ~ sex + age + pclass, data = Titanicp, method
    = "class",
                        control = rpart.control(cp = 0.009))
plot(as.party(titanic_tree0.9))
```



What if we in increase the complexity parameter threshold, so that each new branch needs to decrease the error by 2%?

```
titanic_tree2 = rpart(survived ~ sex + age + pclass, data = Titanicp, method =
    "class",
                      control = rpart.control(cp = 0.02))
plot(as.party(titanic_tree2))
```



Evaluating (in-sample) performance

1% (default)

```
titanic_1_pred = predict(titanic_tree, type = "class")
confusionMatrix(data=titanic_1_pred, reference = Titanicp$survived)$table
```

```
          Reference
Prediction died survived
  died      682      147
  survived  127      353
```

```
confusionMatrix(data=titanic_1_pred, reference = Titanicp$survived)$overall[1]
```

```
 Accuracy
0.7906799
```

0.9%

```
titanic_0.9_pred = predict(titanic_tree0.9, type = "class")
confusionMatrix(data=titanic_0.9_pred,
                reference = Titanicp$survived)$table
```

```
          Reference
Prediction died survived
  died      718      164
  survived   91      336
```

```
confusionMatrix(data=titanic_0.9_pred,
                reference = Titanicp$survived)$overall[1]
```

```
 Accuracy
0.8051948
```

2%

```
titanic_2_pred = predict(titanic_tree2, type = "class")
confusionMatrix(data=titanic_2_pred,
                reference = Titanicp$survived)$table
```

```
          Reference
Prediction died survived
  died      682      161
  survived  127      339
```

```
confusionMatrix(data=titanic_2_pred,
                reference = Titanicp$survived)$overall[1]
```

```
 Accuracy
0.7799847
```

Performance benchmarking

```
table(Titanicp$survived)
```

```
   died survived
    809      500
```

What if our prediction model was just that everyone died? The accuracy would be:

```
809/(809+500)
```

```
[1] 0.618029
```

When considering performance, we should take into account that a "null" model might appear to give quite good performance when we have unbalanced group sizes.

Evaluating (out-of-sample) performance

```
titanic_complete = Titanicp |> select(survived, sex, age, pclass) |> drop_na()
train(survived ~ sex + age + pclass, data = titanic_complete,
      method = "rpart", trControl = trainControl(method = "cv", number = 10))
```

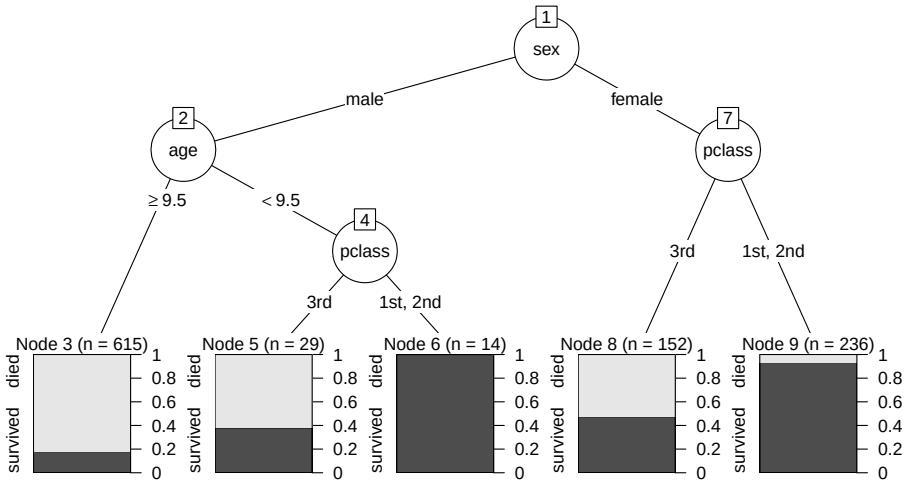
```
CART

1046 samples
   3 predictor
   2 classes: 'died', 'survived'

No pre-processing
```

```
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 941, 941, 941, 941, 941, 942, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.01639344  0.7638239  0.4924266
  0.01873536  0.7638516  0.4923864
  0.45901639  0.6920713  0.2969126

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01873536.
```

Final model

The CV procedure suggested 1.6% for the complexity parameter.

```
titanic_final = rpart(survived ~ sex + age + pclass, data = titanic_complete,
                      control = rpart.control(cp = 0.016))
plot(as.party(titanic_final))
```



Decision tree weaknesses

- Decision trees can become very complex very quickly - without a complexity penalty, it will happily continue until perfect classification (likely massively overfitting the data).
- The selected tree might be very sensitive to the complexity penalty

32.2 Random forests

> In a **random forest** we grow many trees. Each one learns from different (sub)samples of observations and different combinations of variables.

A random forest is constructed by:

1. Choosing the number of decision trees to grow and the number of variables to consider in each tree.
2. Randomly selecting the rows of the data frame **with replacement**.
3. Randomly selecting the appropriate number of variables from the data frame.
4. Building a decision tree on the resulting data set.
5. Repeating this procedure a large number of times.

6. A prediction is be made by **majority rule**, i.e. running your new observation through all the trees in the forest and seeing which class is predicted most often.

- The `randomForest::randomForest()` function in R defaults to 500 trees each trained on `sqrt(p)` variables where `p` is the number of predictors in the full data set.

> **Why do we randomly select the features?**
>
> If one or a few features are very strong predictors for the response variable, then these features will be selected in many of the trees, causing them to become correlated. This could reduce the accuracy of the ensemble.

`randomForest` in R

```
library(randomForest)
iris_rf = randomForest(Species ~ ., iris)
iris_rf
```

```
Call:
 randomForest(formula = Species ~ ., data = iris)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          3        47        0.06
```

Using the `randomForest()` function, we can train our ensemble learning using the same formula we passed to `rpart`.

```
new_data
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1            5         3.9          1.4         0.3      NA
2            5         3.9          3.4         0.3      NA
```

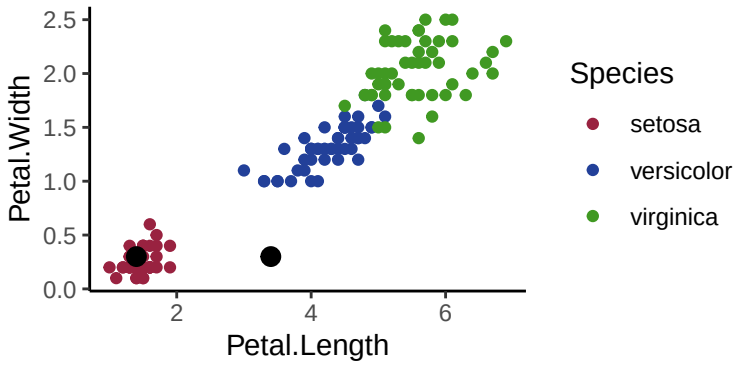
```
predict(iris_rf , new_data)
```

```
     1      2
setosa setosa
Levels: setosa versicolor virginica
```

```
predict(tree , new_data, type = "class")
```

```
         1          2
    setosa versicolor
Levels: setosa versicolor virginica
```

```
p1 + geom_point(data = new_data, size = 3, colour = "black")
```



Titanic random forest

```
titanic_rf = randomForest(survived ~ sex + age + pclass, titanic_complete)
titanic_rf
```

```
Call:
 randomForest(formula = survived ~ sex + age + pclass, data = titanic_complete
     )
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 21.32%
Confusion matrix:
         died survived class.error
died      575       44  0.07108239
survived  179      248  0.41920375
```

```
importance(titanic_rf)
```

```
       MeanDecreaseGini
sex           122.89907
age            21.67346
pclass         41.03750
```

33

Nearest neighbours

33.1 Nearest neighbours

Microchip test data

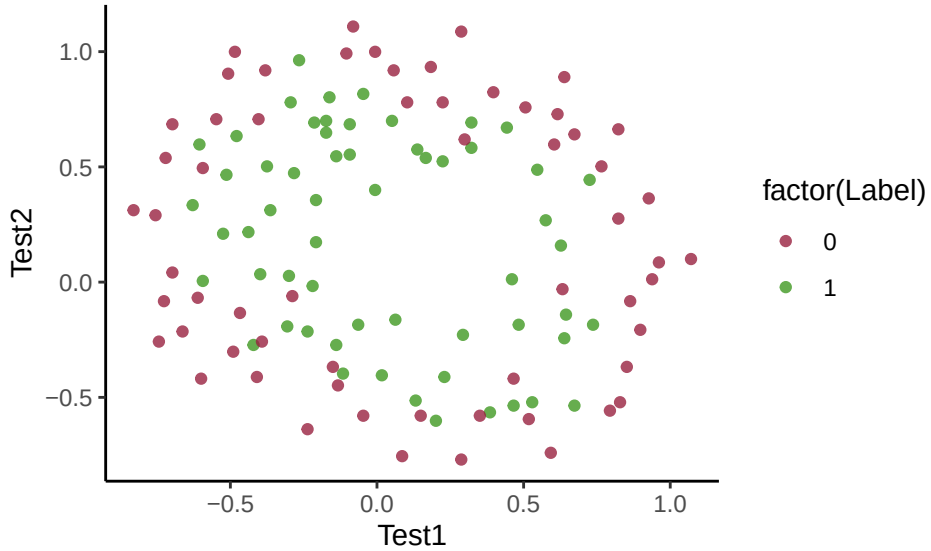
```
data = read_csv("https://raw.githubusercontent.com/DATA2002/data/master/
    Microchips.csv")
glimpse(data)
```

```
Rows: 118
Columns: 3
$ Test1 <dbl> 0.051267, -0.092742, -0.213710, -0.375000, -0.513250, -0.524770,
    ~
$ Test2 <dbl> 0.699560, 0.684940, 0.692250, 0.502190, 0.465640, 0.209800, 0.03
    ~
$ Label <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
    ~
```

The `Microchips` dataset contains test scores `Test1` and `Test2`, and `Label` indicating whether or not the chip passed the test.

> **Why can't we use logistic regression or decision trees to predict the class for a new data point?**

```
ggplot(data) +
  aes(x=Test1, y=Test2,
      color=factor(Label)) +
  geom_point(alpha = 0.8) +
  scale_color_manual(values=c("#992240", "#409922")) +
  theme_classic()
```



Different data types require different approaches

We can use logistic regression to **classify** binary observations, this isn't feasible when:

- We have high class separation in our data (perfect separation)
- We have a non-linear combination of predictors influencing our response

We can use decision trees to **classify** observations into 2 or more classes, but this

- Can be complicated
- Might overfit
- Only draws boundaries parallel to axes

> **So, what other options do we have?**

The k-nearest neighbours algorithm

> Data with p attributes (explanatory variables) are points in a p-dimensional space. We can calculate distances between these points.

- **k-nearest neighbours (kNN)** is a non-parametric algorithm (doesn't assume the data follows any particular shape).
- In kNN, we vote on the class of a new data point by looking at the majority class of the **k** nearest neighbours.

k-nearest neighbours (general formulation)

- Suppose we have a set of training data $(\mathbf{X}, \mathbf{y})$.
- For some positive integer k, a k-nearest neighbour algorithm classifies a new observation $\mathbf{x}_0$ by:

1. Finding the k observations in the training data $\mathbf{X}$, that are closest to $\mathbf{x}_0$ according to some distance metric.
2. Denote the set of k closest observations as $D(\mathbf{x}_0 \subseteq (\mathbf{X}, \mathbf{y}))$
3. Define an aggregation function f (e.g. the majority rules aggregation function) and compute $\hat{y} = f(\mathbf{y})$ for the k values of $\mathbf{y}$ in $D(\mathbf{x}_0)$.

How to measure closeness?

The Euclidean distance between any two points is the square root of the sum of the squared deviations:

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + \ldots + (x_p - z_p)^2}$$

- It makes sense to talk about the distance between two observations (rows in the data frame) in p dimensional space.
- **k-nearest neighbours** uses these distances and the understanding that points "close" to each other probably have similar outcomes.
- Usefulness depends on the geometry of the data: are the points clustered together? What is the distribution of differences among each variable? A wider scale on one variable can dwarf a narrow scale on another variable.

kNN: advantages

- kNN is easy to understand and implement
- The kNN classifier doesn't need to pre-process the training data to make predictions - it can do this on the fly so it adapts easily to new data

- Only requires two "hyperparameters": k and the distance metric
- Analytically tractable and simple implementation
- Performance improves as the sample size grows
- Uses **local** information, and is highly adaptive
- Can be easily parallelised

kNN: disadvantages

- The distance metric only makes sense for quantitative variables (not categorical predictors).
- Does not scale well and predictions can be slow: computationally intensive when predicting new observations, data is processed at that time, lazy algorithm.
- Need to store the full data set (large storage requirements for big data)
- Curse of dimensionality: doesn't perform well with high dimensional inputs, i.e. with lots of predictors
- Easy to overfit/underfit: Small k can overfit the data. High k tend to "smooth out" the predictions, but if k is too high then might underfit.

Choosing k

- An appropriate choice of k will depend on the application and the data.
- Cross validation can be used to optimise the choice of k.
- Misclassification rate increases as k increases. This implies that if you're looking to minimise the misclassification rate on a particular data set, the optimal value of k is 1.

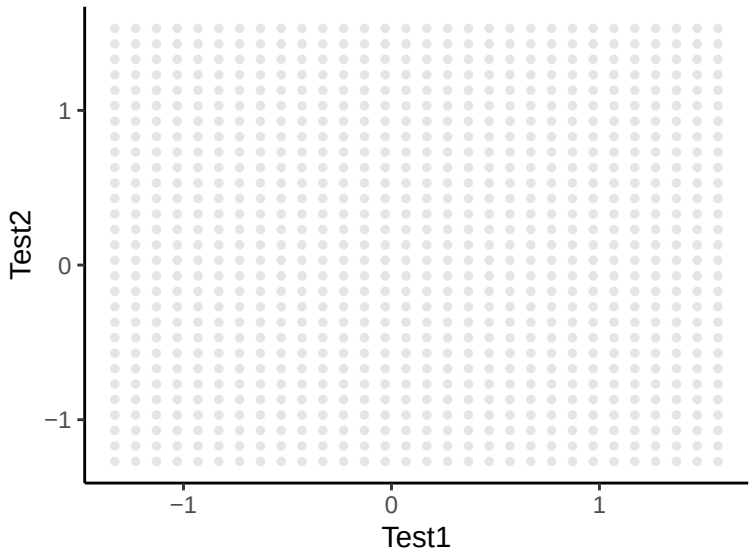
kNN in R

Split our data into **predictors** (X), and response (y),

```
library(class)
X = data |>
  dplyr::select(Test1, Test2)
y = as.factor(data$Label)
```

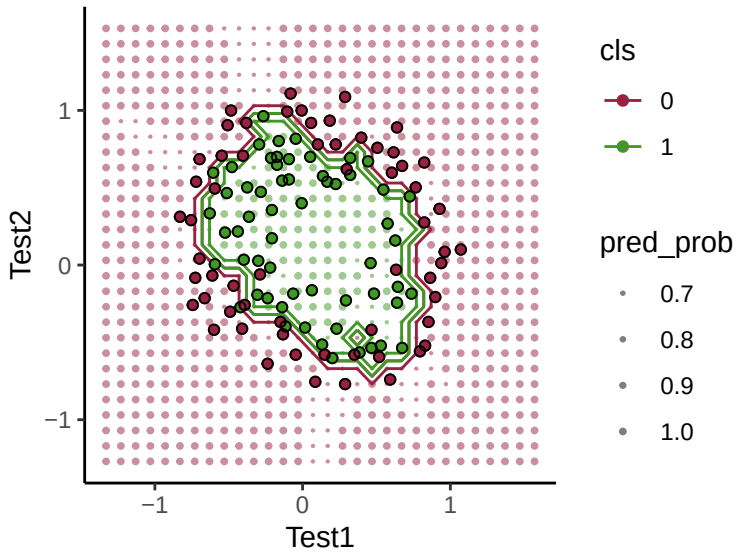
Use knn to predict a grid of new data points:

```
new.X = expand.grid(
  Test1 = seq(min(X$Test1 - 0.5),
              max(X$Test1 + 0.5),
              by = 0.1),
  Test2 = seq(min(X$Test2 - 0.5),
              max(X$Test2 + 0.5),
              by = 0.1)
  )
p = ggplot(new.X) +
  geom_point(aes(x=Test1, y=Test2),
             alpha=0.1, size = 1) +
  theme_classic()
p
```



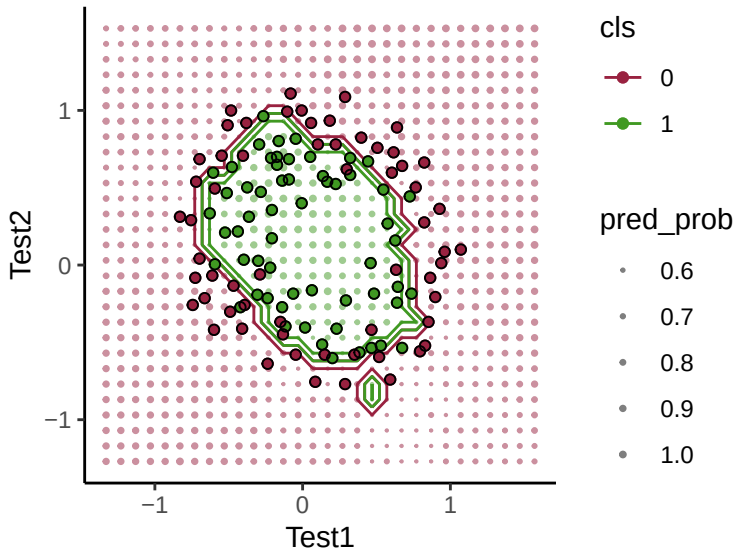
Now we call the knn function, putting in our original observations and their observed the classes as well as the grid of "new points" that we'll be predicting. We also need to specify how many neighbours we want to learn from. For now let $k = 3$

```
pred_k3 = knn(train = X, test = new.X,
              cl = y, k = 3, prob = TRUE)
pred_prob_k3 = attr(pred_k3, "prob")
dataf_k3 = bind_rows(
  mutate(new.X, pred_prob = pred_prob_k3, cls = 1,
         prob_cls = ifelse(pred_k3 == cls, 1, 0)),
  mutate(new.X, pred_prob = pred_prob_k3, cls = 0,
         prob_cls = ifelse(pred_k3 == cls, 1, 0)))
ggplot() +
  geom_point(aes(x = Test1, y = Test2, col = cls, size = pred_prob),
             data = mutate(new.X, cls = pred_k3, pred_prob = pred_prob_k3),
                 alpha=0.5) +
  scale_size(range=c(0.1, 0.7)) +
  geom_contour(aes(x = Test1, y = Test2, z = prob_cls,
                   group = as.factor(cls), color = as.factor(cls)),
               bins = 2, data = dataf_k3) +
  geom_point(aes(x = Test1, y = Test2, col = factor(Label)), data = data) +
  geom_point(aes(x = Test1, y = Test2),shape = 1, data = data) +
  scale_color_manual(values=c("#992240", "#409922")) +
  theme_classic()
```



We can also see how our decisions change based on the number of neighbours we consider. Here, we consider k=9.

```
pred_k9 = knn(train = X, test = new.X,
              cl = y, k = 9, prob = TRUE)
pred_prob_k9 = attr(pred_k9, "prob")
dataf_k9 = bind_rows(
  mutate(new.X, pred_prob = pred_prob_k9, cls = 1,
         prob_cls=ifelse(pred_k9 == cls, 1, 0)),
  mutate(new.X, pred_prob=pred_prob_k9, cls = 0,
         prob_cls = ifelse(pred_k9 == cls, 1, 0)))
ggplot() +
  geom_point(aes(x=Test1, y=Test2, col=cls, size=pred_prob),
             data = mutate(new.X, cls = pred_k9, pred_prob = pred_prob_k9),
                 alpha=0.5) +
  scale_size(range=c(0.1, 0.7)) +
  geom_contour(aes(x=Test1, y=Test2, z = prob_cls,
                   group = as.factor(cls), color=as.factor(cls)),
               bins=2, data=dataf_k9) +
  geom_point(aes(x=Test1, y=Test2, col=factor(Label)), data = data) +
  geom_point(aes(x=Test1, y=Test2), shape=1, data = data) +
  scale_color_manual(values=c("#992240", "#409922")) +
  theme_classic()
```



33.2 Performance assessment

Let's compare k=3 and k=9

In sample confusion matrix

k=3

```
k3 = knn(train = X, test = X,
         cl = y, k = 3)
confusionMatrix(k3, y)$table
```

```
          Reference
Prediction  0  1
         0 50  6
         1 10 52
```

```
confusionMatrix(k3, y)$overall[1] |>
  round(2)
```

```
Accuracy
    0.86
```

k=9

```
k9 = knn(train = X, test = X,
                 cl = y, k = 9)
confusionMatrix(k9, y)$table
```

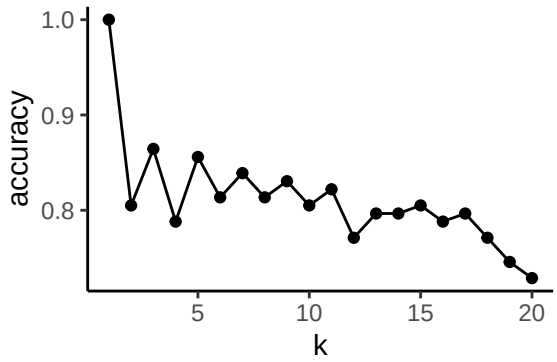
```
          Reference
Prediction  0  1
         0 51 11
         1  9 47
```

```
confusionMatrix(k9, y)$overall[1] |>
  round(2)
```

```
Accuracy
    0.83
```

Let's compare lots of k! (Still in-sample)

```
res = data.frame(
  k = 1:20,
  accuracy = NA
)
for(k in res$k){
  k_temp = knn(train = X, test = X,
               cl = y, k = k)
  cmat = confusionMatrix(k_temp, y)
  res$accuracy[k] = cmat$overall[1]
}
ggplot(res) +
  aes(x = k, y = accuracy) +
  geom_point() + geom_line() +
  theme_classic()
```

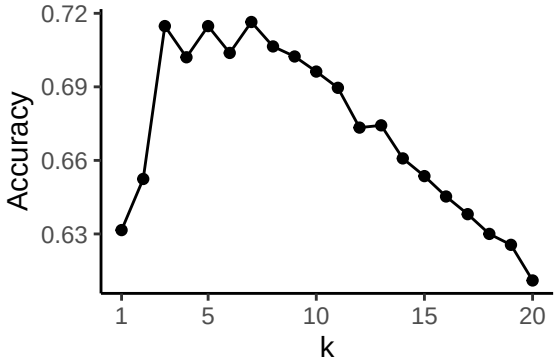


What's with the saw-tooth pattern?!?!

Repeated 5 fold CV

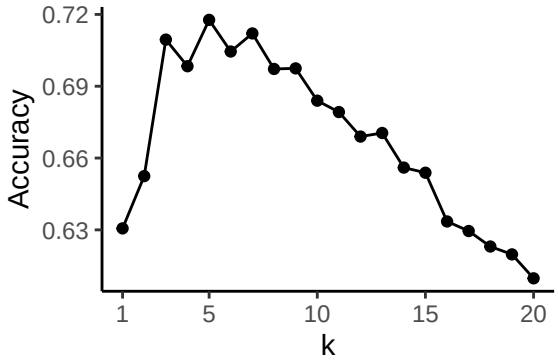
To help choose k

```
set.seed(1)
cv_acc_k = array(dim = c(20, 5, 100))
for(j in 1:100){
  fold_id = sample(c(1,2,3,rep(1:5, each = 23)))
  for(k in 1:20){
    for(i in 1:5){
      k_temp = knn(train = X[fold_id!=i,],
                   test = X[fold_id==i,],
                   cl = y[fold_id!=i], k = k)
      cmat = confusionMatrix(k_temp, y[fold_id==i])
      cv_acc_k[k,i,j] = cmat$overall[1]
    }
  }
}
cv_res = data.frame(
  k = 1:20,
  k_acc = apply(cv_acc_k, 1, mean),
  k_acc_n = apply(cv_acc_k, 1, length)
)
p1 = ggplot(cv_res) + aes(x = k, y = k_acc) +
  geom_point() + geom_line() +
  scale_x_continuous(breaks = c(1,5,10,15,20)) +
  labs(y = "Accuracy") +
  theme_classic()
p1 %+% cv_res
```




```
set.seed(2)
cv_acc_k = array(dim = c(20, 5, 100))
for(j in 1:100){
  fold_id = sample(c(1,2,3,rep(1:5, each = 23)))
```

```
  for(k in 1:20){
    for(i in 1:5){
      k_temp = knn(train = X[fold_id!=i,],
                   test = X[fold_id==i,],
                   cl = y[fold_id!=i],
                   k = k)
      cmat = confusionMatrix(k_temp, y[fold_id==i])
      cv_acc_k[k,i,j] = cmat$overall[1]
    }
  }
}
cv_res = data.frame(
  k = 1:20,
  k_acc = apply(cv_acc_k, 1, mean),
  k_acc_n = apply(cv_acc_k, 1, length)
)
p1 %+% cv_res
```



Using the caret package

```
fitCtrl = trainControl(method = "repeatedcv", number = 5, repeats = 100)
set.seed(1)
knnFit1 = caret::train(factor(Label) ~ ., data = data, method = "knn",
                       trControl = fitCtrl)
knnFit1
```

```
k-Nearest Neighbors

118 samples
  2 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 100 times)
Summary of sample sizes: 94, 94, 95, 95, 94, 94, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7295725  0.4589585
  7  0.7263261  0.4527712
  9  0.7195254  0.4393039

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

34

Clustering

34.1 k-means clustering

k-means clustering - algorithm

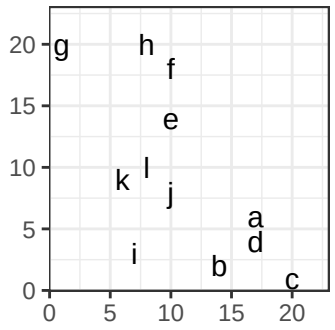
This is an iterative procedure. To use it the number of clusters, k, must be decided first. The stages of the iteration are:

- Initialize by either (a) partitioning the data into k groups, and compute the k group means or (b) an initial set of k points as the first estimate of the cluster means (seed points).
- Loop over all observations reassigning them to the group with the closest mean.
- Recompute group means.
- Iterate steps 2 and 3 until convergence.

```
set.seed(88)
df = tibble(lbl=letters[1:12],
                        x1=sample(1:10, 12, replace=TRUE),
                        x2=sample(1:10, 12, replace=TRUE))
df[1:4,2] = df[1:4,2] + 12
df[5:8,3] = df[5:8,3] + 12
kable(df, booktabs = TRUE) |>
  kable_styling(position="center", latex_options = "hold_position")
```

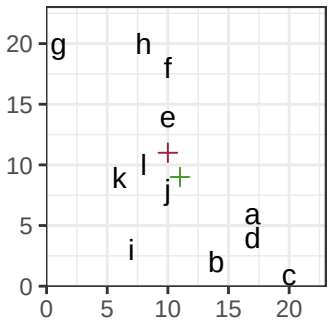
lbl	x1	x2
a	17	6
b	14	2
c	20	1
d	17	4
e	10	14
f	10	18
g	1	20
h	8	20
i	7	3
j	10	8
k	6	9
l	8	10

```
ggplot(data=df, aes(x1, x2)) + geom_text(aes(label=lbl)) +
  xlab("") + ylab("") + theme_bw() +
  xlim(c(1,22)) + ylim(c(1,22))
```



Select $k = 2$, and set initial seed means

```
xb = data.frame(cl = factor(c(1, 2)),
    x1 = c(10,11), x2 = c(11, 9))
mn = data.frame(cl=xb$cl, lbl=c("m1",
    "m2"),
                x1=xb$x1, x2=xb$x2)
ggplot(data=df, aes(x1, x2)) +
    geom_text(aes(label=lbl)) +
  xlab("") + ylab("") + theme_bw() +
  xlim(c(1,22)) + ylim(c(1,22)) +
  geom_point(data=mn, aes(x=x1, y=x2,
      color=cl),
             shape=3, size=2) +
  scale_color_manual(values=c("#992240
      ", "#409922")) +
  theme(aspect.ratio=1,
      legend.position="none")
```



Assign the cluster membership

Compute distances (d_1, d_2) between each observation and each mean.

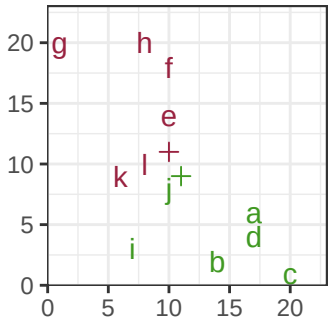
```
# Compute distances between each
    observation and each mean
d1 = apply(df[,2:3], 1, function(x)
    round(sqrt(sum((x-xb[1,2:3])^2))
    ,1))
d2 = apply(df[,2:3], 1, function(x)
    round(sqrt(sum((x-xb[2,2:3])^2))
    ,1))
df.km = cbind(df, d1, d2)
df.km |>
  kable(booktabs = TRUE) |>
  column_spec(4:5, color="myblue")
```

lbl	x1	x2	d1	d2
a	17	6	8.6	6.7
b	14	2	9.8	7.6
c	20	1	14.1	12.0
d	17	4	9.9	7.8
e	10	14	3.0	5.1
f	10	18	7.0	9.1
g	1	20	12.7	14.9
h	8	20	9.2	11.4
i	7	3	8.5	7.2
j	10	8	3.0	1.4
k	6	9	4.5	5.0
l	8	10	2.2	3.2

```
df.km$cl = ifelse(d1 < d2, 1, 2)
df.km$cl = factor(df.km$cl)
kable(df.km, booktabs = TRUE) |>
  column_spec(4:6, color="myblue")
```

lbl	x1	x2	d1	d2	cl
a	17	6	8.6	6.7	2
b	14	2	9.8	7.6	2
c	20	1	14.1	12.0	2
d	17	4	9.9	7.8	2
e	10	14	3.0	5.1	1
f	10	18	7.0	9.1	1
g	1	20	12.7	14.9	1
h	8	20	9.2	11.4	1
i	7	3	8.5	7.2	2
j	10	8	3.0	1.4	2
k	6	9	4.5	5.0	1
l	8	10	2.2	3.2	1

```
ggplot() +
  geom_text(data=df.km, aes(x=x1, y=x2
      , label=lbl, color=cl)) +
  geom_point(data=mn, aes(x=x1, y=x2,
      color=cl),
             shape=3, size=2) +
  scale_color_manual(values=c("#992240
      ", "#409922")) +
  xlab("") + ylab("") + theme_bw() +
  xlim(c(1,22)) + ylim(c(1,22)) +
  theme(aspect.ratio=1,
      legend.position="None")
```



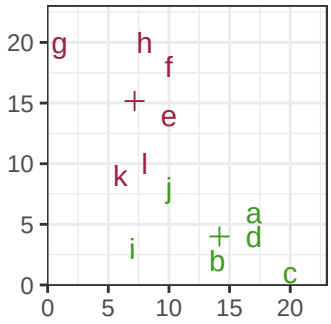
Recompute means, and re-assign the cluster membership

```
xb = df.km |>
  dplyr::group_by(cl) |>
  dplyr::summarise(x1=mean(x1), x2=
      mean(x2))
xb1 = data.frame(x1=xb$x1[1], x2=xb$x2
    [1])
xb2 = data.frame(x1=xb$x1[2], x2=xb$x2
    [2])
d1 = apply(df[,2:3], 1, function(x)
    round(sqrt(sum((x-xb[1,2:3])^2))
    ,1))
d2 = apply(df[,2:3], 1, function(x)
    round(sqrt(sum((x-xb[2,2:3])^2))
    ,1))
df.km$d1 = round(d1, 1)
df.km$d2 = round(d2, 1)
df.km$cl = ifelse(d1 < d2, 1, 2)
df.km$cl = factor(df.km$cl)
kable(df.km, booktabs = TRUE) |>
  column_spec(4:6, color="myblue")
```

lbl	x1	x2	d1	d2	cl
a	17	6	13.4	3.5	2
b	14	2	14.8	2.0	2
c	20	1	19.1	6.6	2
d	17	4	14.9	2.8	2
e	10	14	3.1	10.8	1
f	10	18	4.0	14.6	1
g	1	20	7.8	20.7	1
h	8	20	4.9	17.1	1
i	7	3	12.2	7.2	2
j	10	8	7.7	5.8	2
k	6	9	6.3	9.6	1
l	8	10	5.2	8.6	1

Recompute means, and re-assign the cluster membership

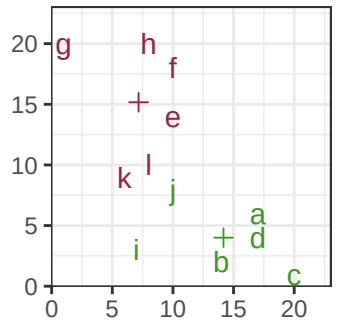
```
mn = data.frame(cl=xb$cl, lbl=c("m1",
    "m2"),
                  x1=xb$x1, x2=xb$x2)
ggplot() +
  geom_text(data=df.km, aes(x=x1, y=x2
      , label=lbl, color=cl)) +
  geom_point(data=mn, aes(x=x1, y=x2,
      color=cl),
             shape=3, size=2) +
  scale_color_manual(values=c("#992240
      ", "#409922")) +
  xlab("") + ylab("") + theme_bw() +
  xlim(c(1,22)) + ylim(c(1,22)) +
  theme(aspect.ratio=1,
      legend.position="None")
```



```
xb = df.km |>
  dplyr::group_by(cl) |>
  dplyr::summarise(x1=mean(x1), x2=
      mean(x2))
xb1 = data.frame(x1=xb$x1[1], x2=xb$x2
    [1])
xb2 = data.frame(x1=xb$x1[2], x2=xb$x2
    [2])
d1 = apply(df[,2:3], 1, function(x)
    round(sqrt(sum((x-xb[1,2:3])^2))
    ,1))
d2 = apply(df[,2:3], 1, function(x)
    round(sqrt(sum((x-xb[2,2:3])^2))
    ,1))
df.km$d1 = round(d1, 1)
df.km$d2 = round(d2, 1)
df.km$cl = ifelse(d1 < d2, 1, 2)
df.km$cl = factor(df.km$cl)
kable(df.km, booktabs = TRUE) |>
  column_spec(4:6, color="myblue")
```

lbl	x1	x2	d1	d2	cl
a	17	6	13.4	3.5	2
b	14	2	14.8	2.0	2
c	20	1	19.1	6.6	2
d	17	4	14.9	2.8	2
e	10	14	3.1	10.8	1
f	10	18	4.0	14.6	1
g	1	20	7.8	20.7	1
h	8	20	4.9	17.1	1
i	7	3	12.2	7.2	2
j	10	8	7.7	5.8	2
k	6	9	6.3	9.6	1
l	8	10	5.2	8.6	1

```
mn = data.frame(cl=xb$cl, lbl=c("m1",
    "m2"),
                                x1=xb
                                  $
                                  x1
                                  ,
                                  x2
                                  =
                                  xb
                                  $
                                  x2
                                  )
ggplot() +
  geom_text(data=df.km, aes(x=x1, y=x2
      , label=lbl, color=cl)) +
  geom_point(data=mn, aes(x=x1, y=x2,
      color=cl),
                          shape=3, size
                              =2) +
  scale_color_manual(values=c("#992240
      ", "#409922")) +
  xlab("") + ylab("") + theme_bw() +
  xlim(c(1,22)) + ylim(c(1,22)) +
  theme(aspect.ratio=1,
      legend.position="None")
```

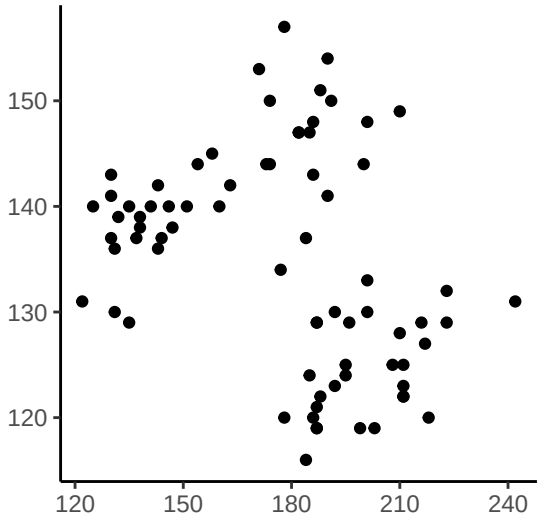


Flea

```
glimpse(flea)
```

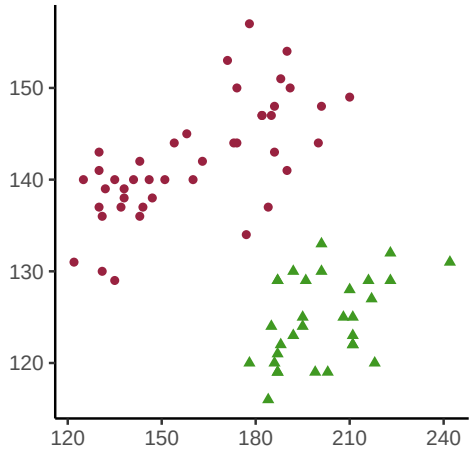
```
Rows: 74
Columns: 7
$ species <fct> Concinna, Concinna, Concinna, Concinna, Concinna, Concinna, Co
    ~
$ tars1   <int> 191, 185, 200, 173, 171, 160, 188, 186, 174, 163, 190, 174, 20
    ~
$ tars2   <int> 131, 134, 137, 127, 118, 118, 134, 129, 131, 115, 143, 131, 13
    ~
$ head    <int> 53, 50, 52, 50, 49, 47, 54, 51, 52, 47, 52, 50, 51, 53, 51, 51
    ~
$ aede1   <int> 150, 147, 144, 144, 153, 140, 151, 143, 144, 142, 141, 150, 14
    ~
$ aede2   <int> 15, 13, 14, 16, 13, 15, 14, 14, 14, 15, 13, 15, 13, 15, 14, 14
    ~
$ aede3   <int> 104, 105, 102, 97, 106, 99, 98, 110, 116, 95, 99, 105, 110, 10
    ~
```

```
ggplot(data=flea) +
  geom_point(aes(x=tars1, y=aede1)) +
  theme(aspect.ratio=1) + xlab("") + ylab("") +
  theme_classic() +
  theme(aspect.ratio=1, legend.position = "none")
```



$k = 2$

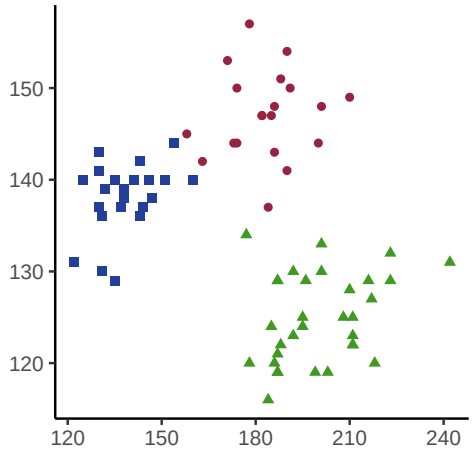
```
set.seed(88)
flea$km2 = kmeans(scale(flea[,c(2,5)]), 2, nstart=5)$cluster
flea$km3 = kmeans(scale(flea[,c(2,5)]), 3, nstart=5)$cluster
flea$km4 = kmeans(scale(flea[,c(2,5)]), 4, nstart=5)$cluster
flea$km5 = kmeans(scale(flea[,c(2,5)]), 5, nstart=5)$cluster
ggplot(data=flea) +
  geom_point(aes(x=tars1, y=aede1, colour=factor(km2),
                 shape=factor(km2))) +
  scale_color_manual(values=c("#992240", "#409922")) +
  xlab("") + ylab("") +
  theme_classic() +
  theme(aspect.ratio=1, legend.position = "none")
```



$k = 3$

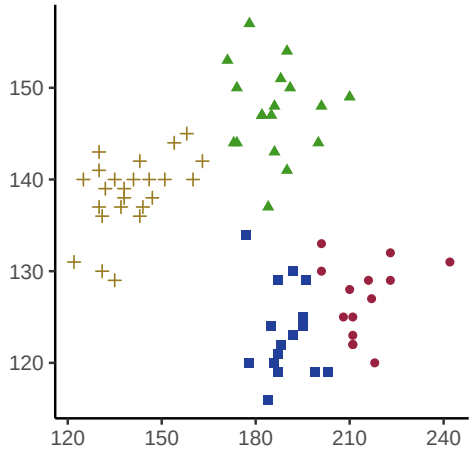
```
ggplot(data=flea) +
  geom_point(aes(x=tars1, y=aede1, colour=factor(km3),
                 shape=factor(km3))) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
```

```
  xlab("") + ylab("") +
  theme_classic() +
  theme(aspect.ratio=1, legend.position = "none")
```



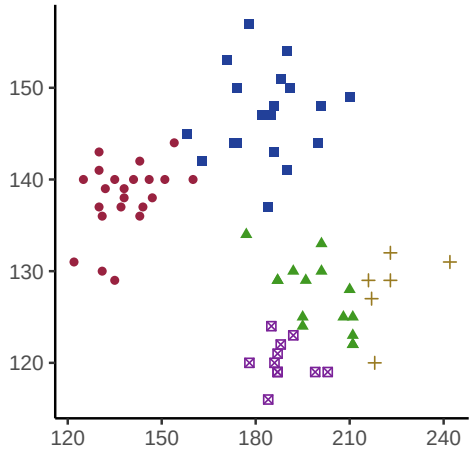
$k = 4$

```
ggplot(data=flea) +
  geom_point(aes(x=tars1, y=aede1, colour=factor(km4),
                 shape=factor(km4))) +
  scale_color_manual(values=c("#992240", "#409922", "#224099", "#997b22")) +
  xlab("") + ylab("") +
  theme_classic() +
  theme(aspect.ratio=1, legend.position = "none")
```



$k = 5$

```
ggplot(data=flea) +
  geom_point(aes(x=tars1, y=aede1, colour=factor(km5),
  shape=factor(km5))) +
  scale_color_manual(values=c("#992240", "#409922", "#224099", "#997b22", "#7
      b2299")) +
  xlab("") + ylab("") +
  theme_classic() +
  theme(aspect.ratio=1, legend.position = "none")
```

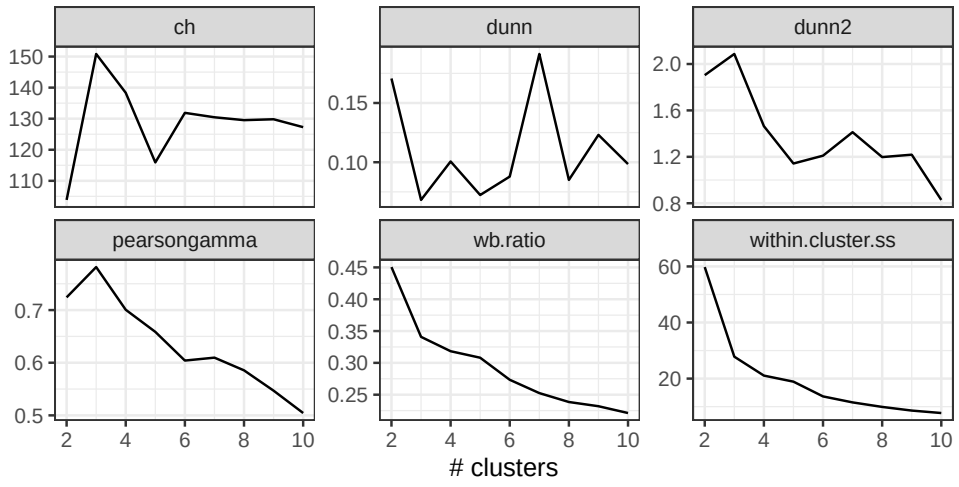


Choosing k

Cluster statistics

- WBRatio: average within/average between want it to be low, but always drops for each additional cluster so look for large drops
- Hubert Gamma: $\frac{s^+ - s^-}{s^+ + s^-}$ where s^+ is sum of number of within < between, s^- sum of number within > between, want this to be high
- Dunn: smallest distance between points from different clusters/maximum distance of points within any cluster, want this to be high
- Calinski-Harabasz Index: $\frac{\sum_{i=1}^{p} B_{ii}/(k-1)}{\sum_{i=1}^{p} W_{ii}/(n-k)}$ want this to be high

```
library(fpc)
set.seed(88)
f.km = NULL; f.km.stats = NULL
for (i in 2:10) {
  cl = kmeans(scale(flea[,c(2,5)]), i, nstart=5)$cluster
  x = cluster.stats(dist(scale(flea[,c(2,5)])), cl)
  f.km = cbind(f.km, cl)
  f.km.stats = rbind(f.km.stats, c(x$within.cluster.ss, x$wb.ratio, x$ch,
                                   x$pearsongamma, x$dunn, x$dunn2))
}
colnames(f.km.stats) = c("within.cluster.ss","wb.ratio", "ch", "pearsongamma",
     "dunn", "dunn2")
f.km.stats = data.frame(f.km.stats)
f.km.stats$cl = 2:10
f.km.stats.m = f.km.stats |>
  gather(stat, value, -cl)
ggplot(data=f.km.stats.m) +
  geom_line(aes(x=cl, y=value)) + xlab("# clusters") + ylab("") +
  facet_wrap(~stat, ncol=3, scales = "free_y") +
  theme_bw()
```



k-means caveats

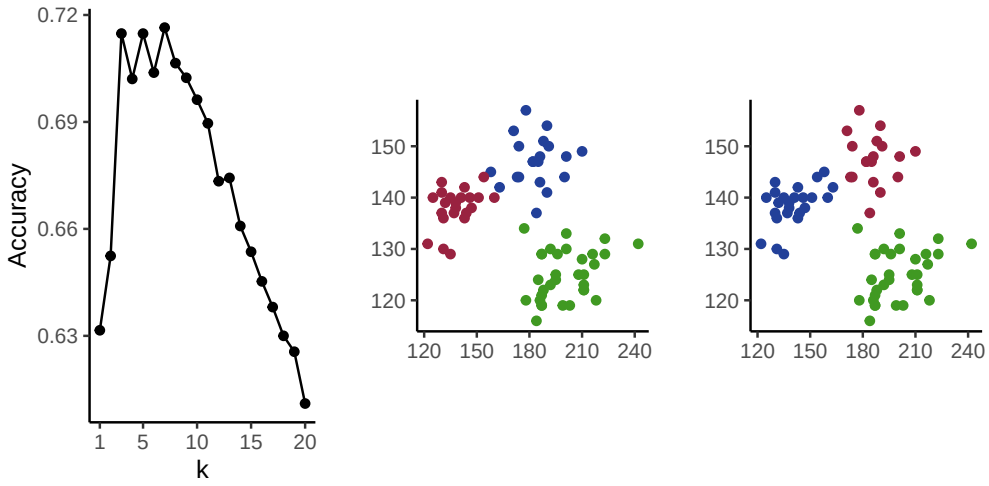
Effect of seed

- The k-means algorithm can yield quite different results depending on the initial seed.
- Example runs used 5 random starts, and used the `within.cluster.ss` metric to decide on the best solution

```
set.seed(2023)
flea$cl1 = kmeans(scale(flea[,c(2,5)]), 3, nstart=1)$cluster
flea$cl2 = kmeans(scale(flea[,c(2,5)]), 3, nstart=1)$cluster
flea$cl3 = kmeans(scale(flea[,c(2,5)]), 3, nstart=1)$cluster
flea$cl4 = kmeans(scale(flea[,c(2,5)]), 3, nstart=1)$cluster
flea$cl5 = kmeans(scale(flea[,c(2,5)]), 3, nstart=1)$cluster
flea$cl6 = kmeans(scale(flea[,c(2,5)]), 3, nstart=1)$cluster
p1 = ggplot(data=flea, aes(x=tars1, y=aede1,
                           colour=factor(cl1))) +
  geom_point() +
  sscale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_classic() +
  theme(aspect.ratio=1, legend.position="none")
```

```
Error in sscale_color_manual(values = c("#992240", "#409922", "#224099")):
    could not find function "sscale_color_manual"
```

```
p2 = ggplot(data=flea, aes(x=tars1, y=aede1,
                           colour=factor(cl2))) +
  geom_point() +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_classic() +
  theme(aspect.ratio=1, legend.position="none")
p3 = ggplot(data=flea, aes(x=tars1, y=aede1,
                           colour=factor(cl5))) +
  geom_point() +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_classic() +
  theme(aspect.ratio=1, legend.position="none")
  gridExtra::grid.arrange(p1, p2, p3, ncol=3)
```



Interpoint distance measures

Euclidean

- Cluster analysis depends on the interpoint distances, points close together should be grouped together
- Euclidean distance was used for the example. Let $A = (x_{a1}, x_{a2}, ..., x_{ap}), B = (x_{b1}, x_{b2}, ..., x_{bp})$

$$d_{EUC}(A,B) = \sqrt{\sum_{j=1}^{p}(x_{aj} - x_{bj})^2}$$
$$= \sqrt{((X_A - X_B)'(X_A - X_B))}$$

Other distance metrics

- Mahalanobis (or statistical) distance

$$\sqrt{((X_A - X_B)^\top S^{-1}(X_A - X_B))}$$

- Manhattan

$$\sum_{j=1}^{p}|(X_{aj} - X_{bj})|$$

- Minkowski

$$(\sum_{j=1}^{p}|(X_{aj} - X_{bj})|^m)^{1/m}$$

Distances for count data

- Canberra

$$\frac{1}{n_z}\sum_{j=1}^{p}\frac{X_{aj} - X_{bj}}{X_{aj} + X_{bj}}$$

- Bray-Curtis

$$\frac{\sum_{j=1}^{p}|X_{aj} - X_{bj}|}{\sum_{j=1}^{p}(X_{aj} + X_{bj})}$$

Interpoint distance measures - Euclidean

Rules for any metric to be a distance

1. $d(A,B) \geqslant 0$
2. $d(A,A) = 0$
3. $d(A,B) = d(B,A)$
4. Metric dissimilarity satisfies $d(A,B) \leqslant d(A,C) + d(C,B)$, and an ultrametric dissimilarity satisfies $d(A,B) \leqslant max\{d(A,C), d(C,B)\}$

34.2 Hierarchical clustering

Hierarchical clustering

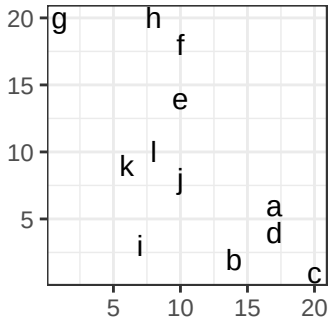
- Agglomeration: Begin with all observations in singleton clusters. Sequentially join points into clusters, until all are in one cluster.
- Divisive: Begin with all observtions in one cluster, and sequentially divide until all observations are in singleton clusters.
- Produces a tree diagram illustrating the process, called a dendrogram.

Some new data:

```
set.seed(88)
df = tibble(lbl=letters[1:12],
            x1=sample(1:10, 12,
                replace=TRUE),
            x2=sample(1:10, 12,
                replace=TRUE))
df[1:4,2] = df[1:4,2] + 12
df[5:8,3] = df[5:8,3] + 12
kable(df, booktabs = TRUE)
```

lbl	x1	x2
a	17	6
b	14	2
c	20	1
d	17	4
e	10	14
f	10	18
g	1	20
h	8	20
i	7	3
j	10	8
k	6	9
l	8	10

```
ggplot(data=df, aes(x1, x2)) +
    geom_text(aes(label=lbl)) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1)
```

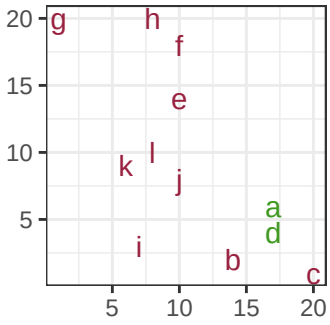


$m \times n$ distance matrix

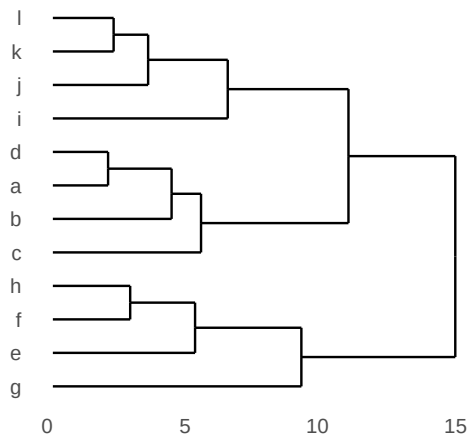
```
d = as.matrix(round(dist(df[,2:3], diag=TRUE, upper=TRUE),1))
colnames(d) = df$lbl
rownames(d) = df$lbl
kable(d, booktabs = TRUE) |>
  kable_styling(position="center", latex_options = "hold_position") |>
  column_spec(c(2,5), color="myblue") |>
  row_spec(c(1,4), color="myblue")
```

	a	b	c	d	e	f	g	h	i	j	k	l
a	0.0	5.0	5.8	2.0	10.6	13.9	21.3	16.6	10.4	7.3	11.4	9.8
b	5.0	0.0	6.1	3.6	12.6	16.5	22.2	19.0	7.1	7.2	10.6	10.0
c	5.8	6.1	0.0	4.2	16.4	19.7	26.9	22.5	13.2	12.2	16.1	15.0
d	2.0	3.6	4.2	0.0	12.2	15.7	22.6	18.4	10.0	8.1	12.1	10.8
e	10.6	12.6	16.4	12.2	0.0	4.0	10.8	6.3	11.4	6.0	6.4	4.5
f	13.9	16.5	19.7	15.7	4.0	0.0	9.2	2.8	15.3	10.0	9.8	8.2
g	21.3	22.2	26.9	22.6	10.8	9.2	0.0	7.0	18.0	15.0	12.1	12.2
h	16.6	19.0	22.5	18.4	6.3	2.8	7.0	0.0	17.0	12.2	11.2	10.0
i	10.4	7.1	13.2	10.0	11.4	15.3	18.0	17.0	0.0	5.8	6.1	7.1
j	7.3	7.2	12.2	8.1	6.0	10.0	15.0	12.2	5.8	0.0	4.1	2.8
k	11.4	10.6	16.1	12.1	6.4	9.8	12.1	11.2	6.1	4.1	0.0	2.2
l	9.8	10.0	15.0	10.8	4.5	8.2	12.2	10.0	7.1	2.8	2.2	0.0

```
df$cl11 = factor(c
    (2,1,1,2,1,1,1,1,1,1,1,1))
ggplot(data=df, aes(x=x1, y=x2, colour
    =cl11)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240
      ", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1,
      legend.position="none")
```




```
library(ggdendro)
df_hc = hclust(as.dist(d), method="
    average")
ggdendrogram(df_hc, rotate = TRUE,
    size = 4)
```

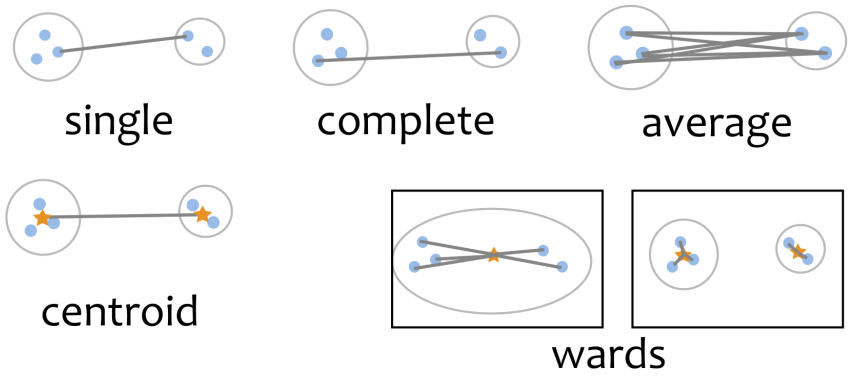


What is the distance between the new cluster (a,d) and all of the other observations?

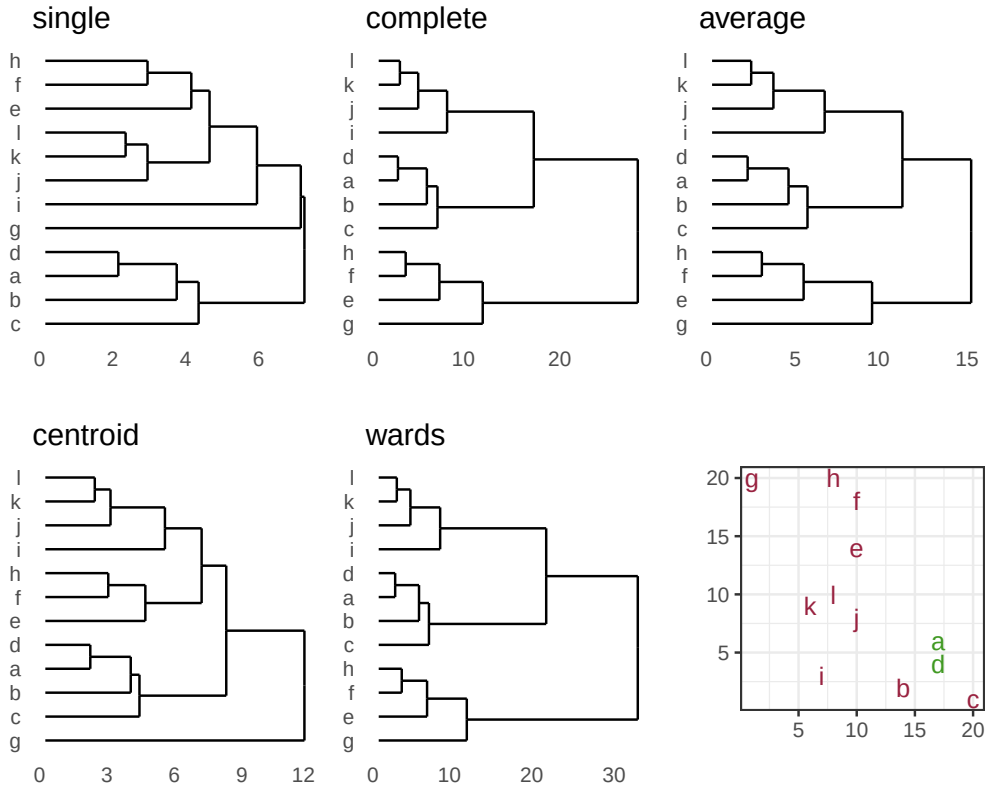
Linkage

Between points in the cluster to points not in the cluster.

- Single: minimum distance between points in the different clusters
- Complete: maximum distance between points in the different clusters
- Average: average of distances between points in the different clusters
- Centroid: distances between the average of the different clusters
- Wards: minimizes the total within-cluster variance




```
df_hc1 = hclust(as.dist(d), method="single")
p1 = ggdendrogram(df_hc1, rotate = TRUE, size = 4) + ggtitle("single")
df_hc2 = hclust(as.dist(d), method="complete")
p2 = ggdendrogram(df_hc2, rotate = TRUE, size = 4) + ggtitle("complete")
df_hc3 = hclust(as.dist(d), method="average")
p3 = ggdendrogram(df_hc3, rotate = TRUE, size = 4) + ggtitle("average")
df_hc4 = hclust(as.dist(d), method="centroid")
p4 = ggdendrogram(df_hc4, rotate = TRUE, size = 4) + ggtitle("centroid")
df_hc5 = hclust(as.dist(d), method="ward.D2")
p5 = ggdendrogram(df_hc5, rotate = TRUE, size = 4) + ggtitle("wards")
p6 = ggplot(data=df, aes(x=x1, y=x2, colour=cl11)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1, legend.position="none")
gridExtra::grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
```

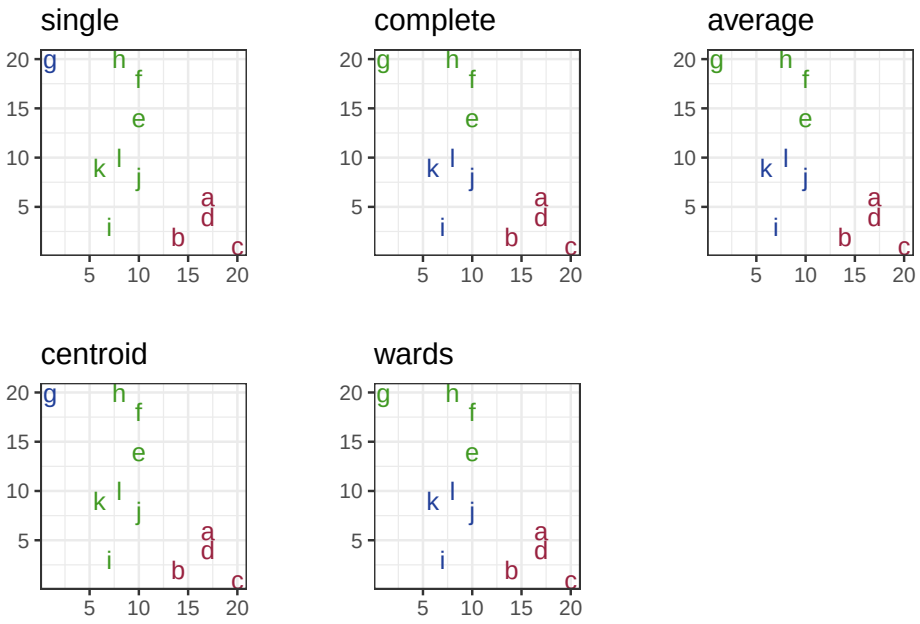


Dendrogram

- Each leaf of the dendrogram represents one observation
- Leaves fuse into branches and branches fuse, either with leaves or other branches.
- Fusions .lower in the tree mean the groups of observations are more similar to each other.

Cut the tree to partition the data into k clusters.

```
df$cl1 = factor(cutree(df_hc1, 3))
df$cl2 = factor(cutree(df_hc2, 3))
df$cl3 = factor(cutree(df_hc3, 3))
df$cl4 = factor(cutree(df_hc4, 3))
df$cl5 = factor(cutree(df_hc5, 3))
p1 = ggplot(data=df, aes(x=x1, y=x2, colour=cl1)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1, legend.position="none") + ggtitle("single")
p2 = ggplot(data=df, aes(x=x1, y=x2, colour=cl2)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1, legend.position="none") +
  ggtitle("complete")
p3 = ggplot(data=df, aes(x=x1, y=x2, colour=cl3)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1, legend.position="none") +
  ggtitle("average")
p4 = ggplot(data=df, aes(x=x1, y=x2, colour=cl4)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1, legend.position="none") +
  ggtitle("centroid")
p5 = ggplot(data=df, aes(x=x1, y=x2, colour=cl5)) +
  geom_text(aes(label=lbl)) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  xlab("") + ylab("") + theme_bw() +
  theme(aspect.ratio=1, legend.position="none") +
  ggtitle("wards")
gridExtra::grid.arrange(p1, p2, p3, p4, p5, ncol=3)
```



Pros and cons

- Single linkage tends to "chain" the data into long stringy clusters, can avoid confusion from nuisance variables but gets confused by "inliers" (outliers between clusters)
- Complete linkage tends to be confused by nuisance variables, but not by inliers
- Wards tends to create spherical homogeneously shaped clusters, a little similar to k-mean

No one perfect method for all problems, but Wards tends to be a good starting point.

35
Dimension reduction

35.1 Principal component analysis

Sub-spaces

- Data will often be confined to a region of the space having lower intrinsic dimensionality.
- The data lives in a low-dimensional subspace.
- The goal is to reduce the dimensionality, to the subspace containing the data.

PCA

- Principal component analysis (PCA) produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- It is an **unsupervised learning** method.

Why use PCA?

- We may have too many predictors for a **regression**. Instead, we can use the first few principal components. PCA regression.
- **Understanding relationships** between variables - similar to a correlation matrix.
- **Data visualisation**: we can plot a small number of variables more easily than a large number of variables.

First principal component

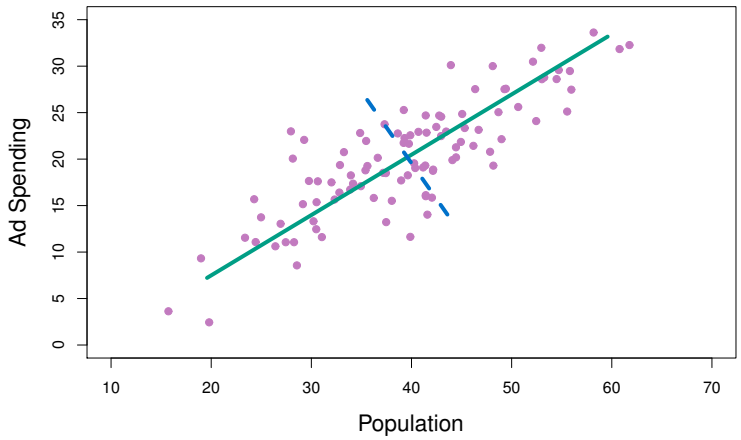
The first principal component of a set of variables $x_1, x_2, \dots, x_p$ is the linear combination

$$z_1 = \varphi_{11}x_1 + \varphi_{21}x_2 + \cdots + \varphi_{p1}x_p$$

that has the largest variance such that

$$\sum_{j=1}^{p} \varphi_{j1}^2 = 1$$

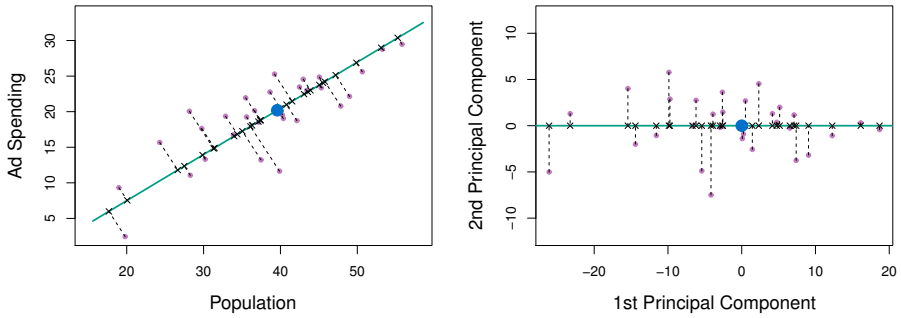
The elements $\varphi_{11}, \dots, \varphi_{p1}$ are the loadings of the first principal component.



First PC; second PC

Geometry

- The loading vector $\varphi_1 = [\varphi_{11}, \dots, \varphi_{p1}]'$ defines direction in feature space along which data vary most.
- If we project the n data points $\mathbf{x}_1, \dots, \mathbf{x}_n$ onto this direction, the projected values are the principal component scores $\mathbf{z}_1 = [z_{11}, \dots, z_{n1}]'$.
- The second principal component is the linear combination $z_{i2} = \varphi_{12}x_{i1} + \varphi_{22}x_{i2} + \dots + \varphi_{p2}x_{ip}$ that has maximal variance among all linear combinations that are *uncorrelated* with $\mathbf{z}_1$.
- Equivalent to constraining φ_2 to be orthogonal (perpendicular) to φ_1 . And so on.
- There are at most $\min(n-1, p)$ PCs.



If you think of the first few PCs like a linear model fit, and the others as the error, it is like regression, except that errors are orthogonal to model.

Total variance

Total variance in data (assuming variables centered at 0):

$$\text{TV} = \sum_{j=1}^{p} \text{Var}(x_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

If variables are standardised, TV=number of variables!
Variance explained by mth principal component:

$$V_m = \text{Var}(z_m) = \frac{1}{n}\sum_{i=1}^{n} z_i^2$$

$$\text{TV} = \sum_{m=1}^{M} V_m \text{where } M = \min(n-1, p).$$

Choosing the number of PCs

- Proportion of variance explained:

$$\text{PVE}_m = \frac{V_m}{\text{TV}}$$

- Choosing the number of PCs that adequately summarises the variation in $\mathbf{X}$ is achieved by examining the cumulative proportion of variance explained.
- Cumulative proportion of variance explained:

$$\text{CPVE}_k = \sum_{m=1}^{k} \frac{V_m}{\text{TV}}$$

and also by a scree plot

Choosing the number of PCs: scree plot

```
df = tibble(npcs=1:10, evl=c(3.5,2.7,2.2,0.5,0.3,0.3,0.2,0.1,0.1,0.1))
ggplot(df, aes(x=npcs, y=evl)) + geom_line() +
  xlab("Number of PCs") + ylab("Eigenvalue") +
  scale_x_continuous(breaks=seq(0,10,1)) + geom_vline(xintercept=4, colour="
      myred", linewidth=1, alpha=0.7) +
  annotate("text", x=4.5, y=3, label="Choose k=4", colour="myred", hjust = 0,
      size=2) +
  theme_classic()
```

```
Error in `geom_vline()`:
! Problem while converting geom to grob.
i Error occurred in the 2nd layer.
Caused by error:
! Unknown colour name: myred
```

National track records

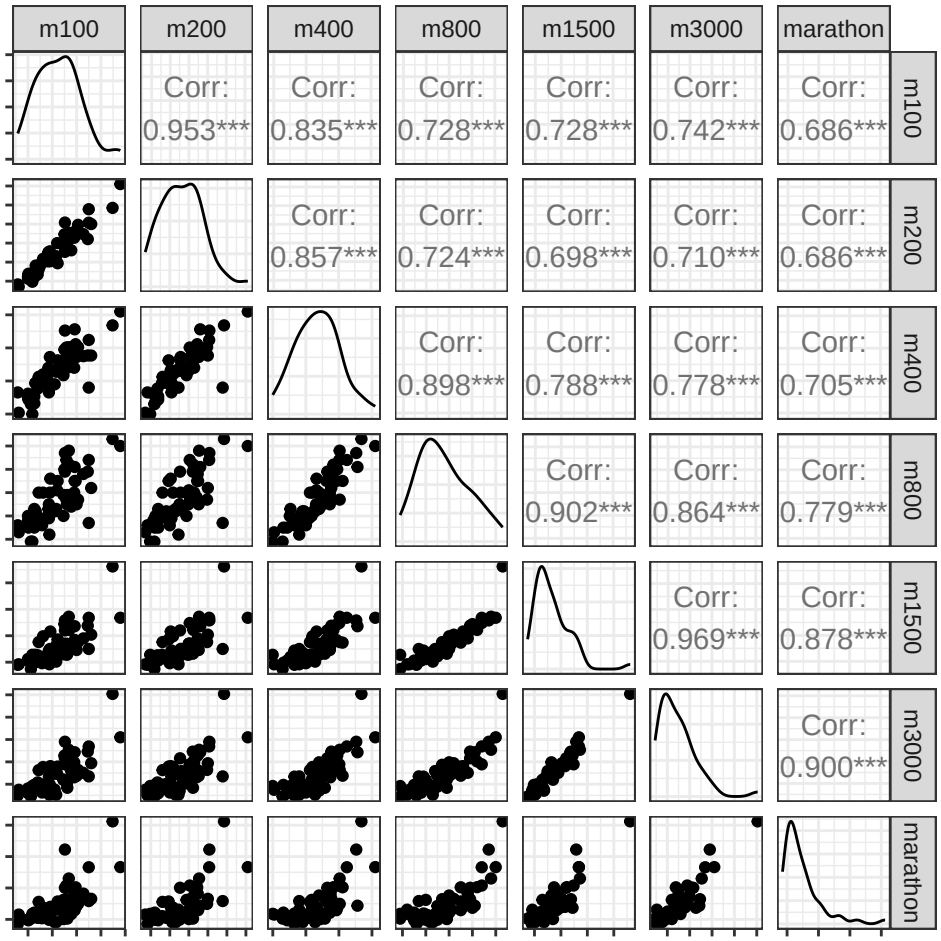
Example: the data on national track records for women (as at 1984).

```
track = readr::read_csv("data/womens_track.csv")
dplyr::glimpse(track)
```

```
Rows: 55
Columns: 8
$ m100     <dbl> 11.61, 11.20, 11.43, 11.41, 11.46, 11.31, 12.14, 11.00, 12.00
    ~
$ m200     <dbl> 22.94, 22.35, 23.09, 23.04, 23.05, 23.17, 24.47, 22.25, 24.52
    ~
$ m400     <dbl> 54.50, 51.08, 50.62, 52.00, 53.30, 52.80, 55.00, 50.06, 54.90
    ~
$ m800     <dbl> 2.15, 1.98, 1.99, 2.00, 2.16, 2.10, 2.18, 2.00, 2.05, 2.08, 2
    ~
$ m1500    <dbl> 4.43, 4.13, 4.22, 4.14, 4.58, 4.49, 4.45, 4.06, 4.23, 4.33, 4
    ~
```

```
$ m3000    <dbl> 9.79, 9.08, 9.34, 8.88, 9.81, 9.77, 9.51, 8.81, 9.37, 9.31, 9
    ~
$ marathon <dbl> 178.52, 152.37, 159.37, 157.85, 169.98, 168.75, 191.02, 149.4
    ~
$ country  <chr> "argentin", "australi", "austria", "belgium", "bermuda", "bra
    ~
```

```
GGally::ggpairs(track[,1:7]) + theme_bw() + theme(axis.text = element_blank())
```



Compute

```
track_pca = prcomp(track[,1:7], center = TRUE, scale = TRUE)
track_pca
```

```
Standard deviations (1, .., p=7):
[1] 2.4094991 0.8084835 0.5476152 0.3542280 0.2319847 0.1976089 0.1498085

Rotation (n x k) = (7 x 7):
                PC1        PC2         PC3         PC4         PC5          PC6
m100      0.3683561  0.4900597 -0.28601157  0.31938631  0.23116950  0.619825234
m200      0.3653642  0.5365800 -0.22981913 -0.08330196  0.04145457 -0.710764580
m400      0.3816103  0.2465377  0.51536655 -0.34737748 -0.57217791  0.190945970
m800      0.3845592 -0.1554023  0.58452608 -0.04207636  0.62032379 -0.019089032
m1500     0.3891040 -0.3604093  0.01291198  0.42953873  0.03026144 -0.231248381
m3000     0.3888661 -0.3475394 -0.15272772  0.36311995 -0.46335476  0.009277159
marathon 0.3670038 -0.3692076 -0.48437037 -0.67249685  0.13053590  0.142280558
                PC7
m100      0.05217655
```

```
m200      -0.10922503
m400       0.20849691
m800      -0.31520972
m1500      0.69256151
m3000     -0.59835943
marathon   0.06959828
```

Assess

```
summary(track_pca)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     2.4095 0.80848 0.54762 0.35423 0.23198 0.19761 0.14981
Proportion of Variance 0.8294 0.09338 0.04284 0.01793 0.00769 0.00558 0.00321
Cumulative Proportion  0.8294 0.92276 0.96560 0.98353 0.99122 0.99679 1.00000
```

```
track_pca_smry = tibble(evl=track_pca$sdev^2) |>
  mutate(p = evl/sum(evl), cum_p = cumsum(evl/sum(evl))) |>t()
colnames(track_pca_smry) = colnames(track_pca$rotation)
rownames(track_pca_smry) = c("Variance", "Proportion", "Cum. prop")
kable(track_pca_smry, digits=2, booktabs = TRUE) |>
  kable_styling(position="center", latex_options = "hold_position")
```

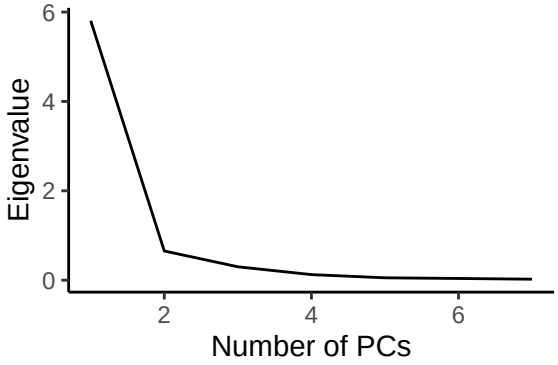
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Variance	5.81	0.65	0.30	0.13	0.05	0.04	0.02
Proportion	0.83	0.09	0.04	0.02	0.01	0.01	0.00
Cum. prop	0.83	0.92	0.97	0.98	0.99	1.00	1.00

```
  row_spec(0, color="white", background = "myblue") |>
  column_spec(1,color="white", background = "myblue") |>
  row_spec(3, color="white", background = "myred")
```

```
Error in row_spec(0, color = "white", background = "myblue"): argument "row"
    is missing, with no default
```

Increase in variance explained large until $k = 3$ PCs, and then tapers off. A choice of 3 PCs would explain 97% of the total variance. Scree plot: Where is the elbow?

```
track_pca_var = tibble(n=1:length(track_pca$sdev), evl=track_pca$sdev^2)
ggplot(track_pca_var, aes(x=n, y=evl)) + geom_line() +
  labs(x = "Number of PCs", y = "Eigenvalue") +
  theme_classic()
```

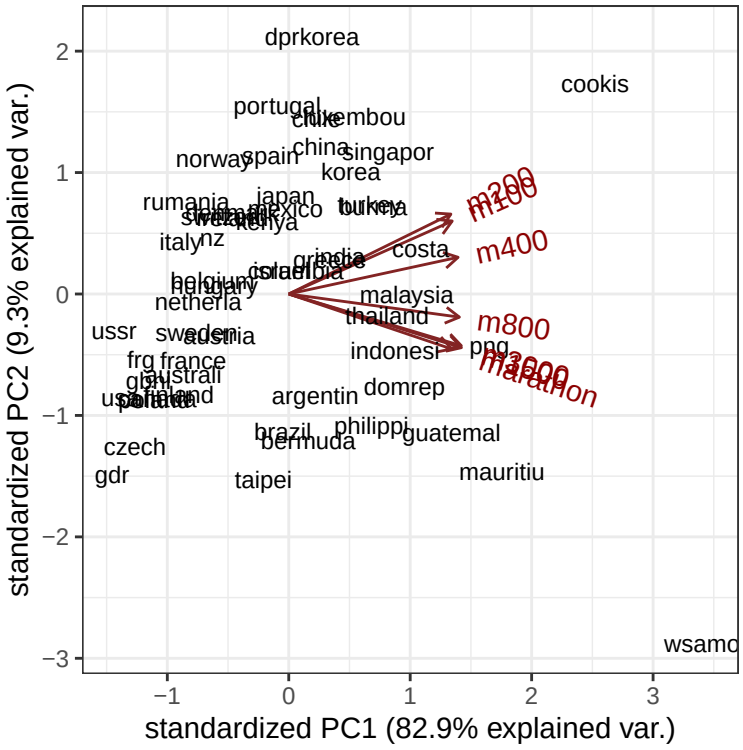


The scree plot suggests just 1 PCs would be sufficient to explain most of the variability.

Visualise

Visualise model using a **biplot**: plot the principal component scores, and also the contribution of the original variables to the principal component.

```
row.names(track) = track$country
library(ggbiplot)
ggbiplot(track_pca,
         labels = rownames(track),
         labels.size = 3,
         varname.size = 4) +
  theme_bw()
```



Interpret

Explain and interpret: using the coefficients of the principal components.

- PC1 measures overall magnitude, the strength of the athletics program. High positive values indicate poor programs with generally slow times across events.
- PC2 measures the contrast in the program between short and long distance events. Some countries have relatively stronger long distance athletes, while others have relatively stronger short distance athletes.
- There are several outliers visible in this plot, `wsamoa`, `cookis`, `dpkorea`. PCA, because it is computed using the variance in the data, can be affected by outliers. It may be better to remove these countries, and re-run the PCA.

Related problems

You have multivariate variables $\mathbf{x}_1, \ldots, \mathbf{x}_n$ so $\mathbf{x}_1 = (x_{11}, \ldots, x_{1p})$

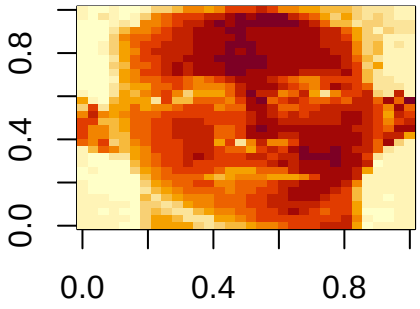
- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is statistical and the second goal is **data compression**.

35.2 Eigenfaces

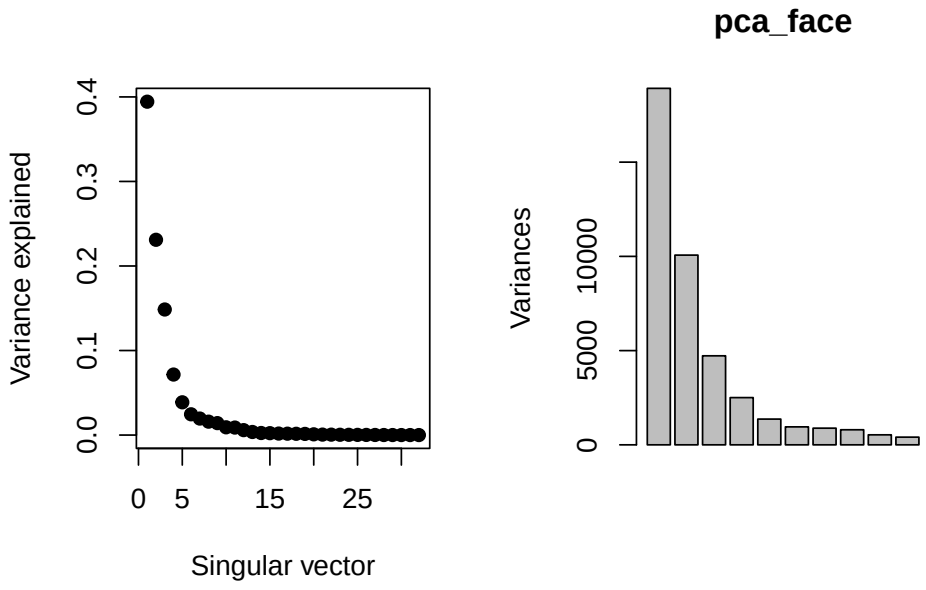
Face example

```
load(url("https://github.com/DATA2002/data/raw/master/face.rda"))
image(t(faceData)[,nrow(faceData):1])
```



Variance explained

```
svd_face = svd(scale(faceData))
pca_face = prcomp(faceData)
par(mfrow=c(1,2))
plot(svd_face$d^2/sum(svd_face$d^2),
     pch=19,
     xlab="Singular vector",
     ylab="Variance explained")
screeplot(pca_face)
```



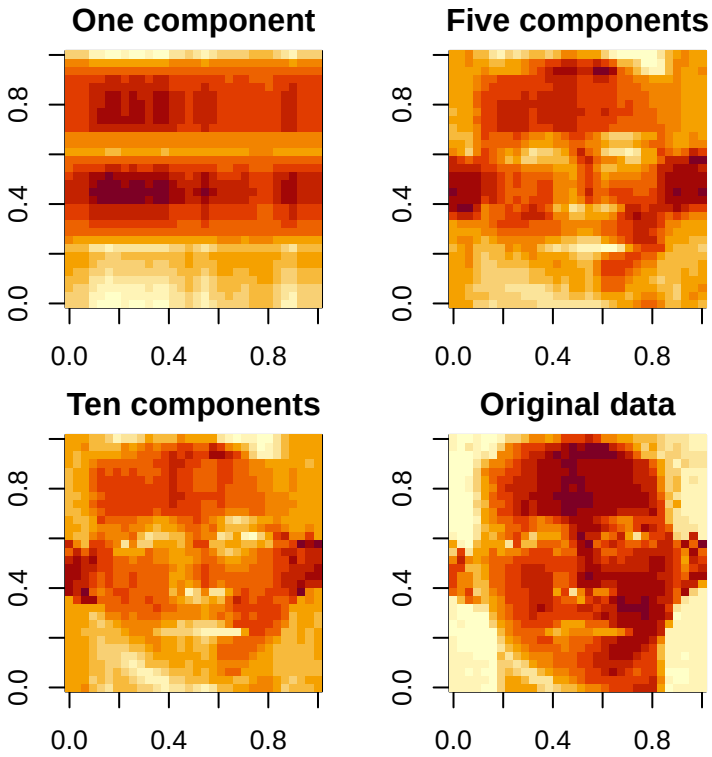
Create approximations

We can reconstruct the face using lower dimensional approximations.

```
svd1 = svd(scale(faceData))
## Note that %*% is matrix multiplication

# Here svd1$d[1] is a constant
approx1 = svd1$u[,1] %*% t(svd1$v[,1]) * svd1$d[1]

# In these examples we need to make the diagonal matrix out of d
approx5 = svd1$u[,1:5] %*% diag(svd1$d[1:5]) %*% t(svd1$v[,1:5])
approx10 = svd1$u[,1:10] %*% diag(svd1$d[1:10]) %*% t(svd1$v[,1:10])
```

Plot approximations

```
par(mfrow=c(2,2), mar = c(2,2,2,2))
image(t(approx1)[,nrow(approx1):1], main = "One component")
image(t(approx5)[,nrow(approx5):1], main = "Five components")
image(t(approx10)[,nrow(approx10):1], main = "Ten components")
image(t(faceData)[,nrow(faceData):1], main = "Original data")  ## Original
    data
```



35.3 Dimension reduction and clustering

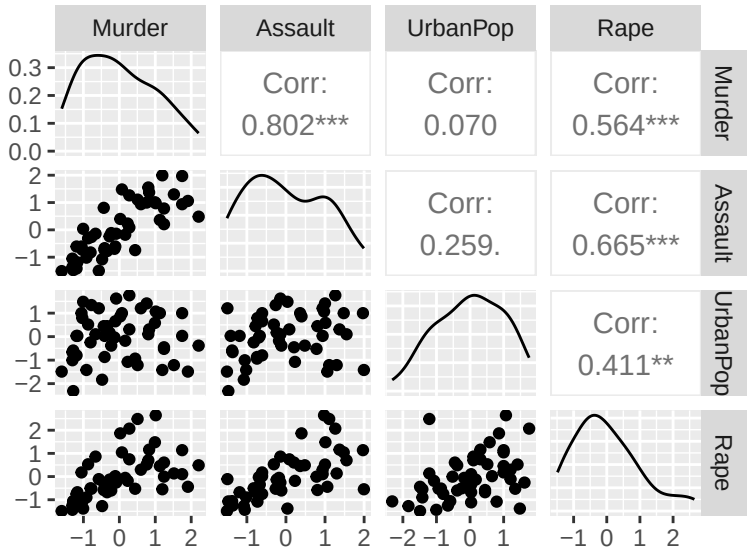
US arrests data

```
data("USArrests")
glimpse(USArrests)
```

```
Rows: 50
Columns: 4
$ Murder   <dbl> 13.2, 10.0, 8.1, 8.8, 9.0, 7.9, 3.3, 5.9, 15.4, 17.4, 5.3, 2.
    ~
```

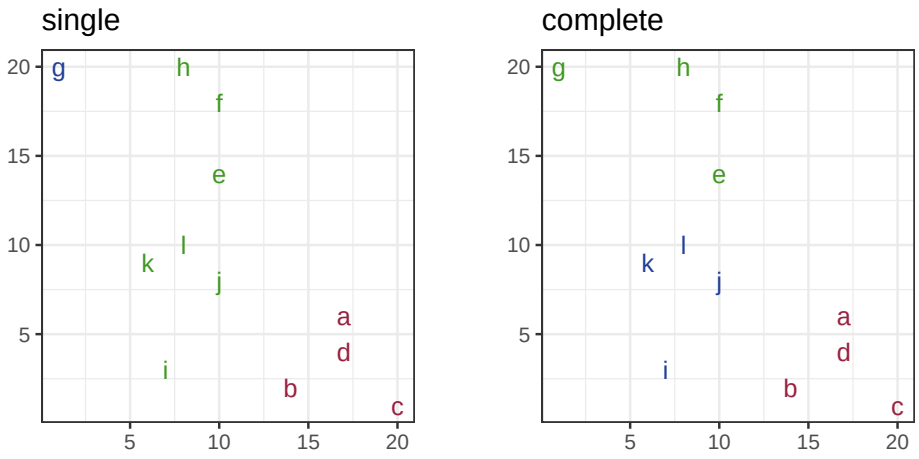
```
$ Assault  <int> 236, 263, 294, 190, 276, 204, 110, 238, 335, 211, 46, 120, 24
    ~
$ UrbanPop <int> 58, 48, 80, 50, 91, 78, 77, 72, 80, 60, 83, 54, 83, 65, 57, 6
    ~
$ Rape     <dbl> 21.2, 44.5, 31.0, 19.5, 40.6, 38.7, 11.1, 15.8, 31.9, 25.8, 2
    ~
```

```
df = data.frame(scale(USArrests))
GGally::ggpairs(df)
```



PCA for US arrests

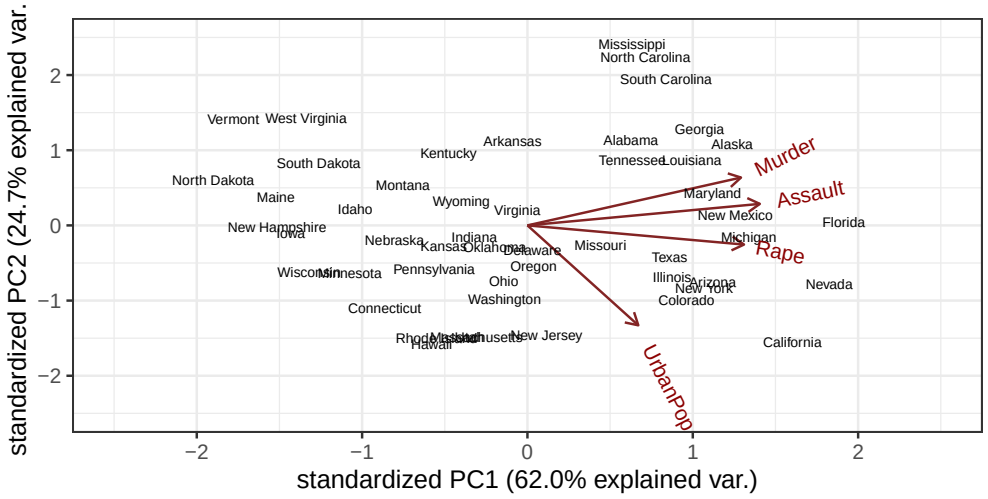
```
pca_arrests = princomp(df)
library(ggbiplot)
p1 = ggscreeplot(pca_arrests) +
  theme_bw() + labs(y = "Prop. of explained variance")
p2 = ggscreeplot(pca_arrests,type = "cev") + theme_bw() + labs(y = "
    Cumulative prop. of explained variance")
gridExtra::grid.arrange(p1, p2, ncol=2)
```



Biplot

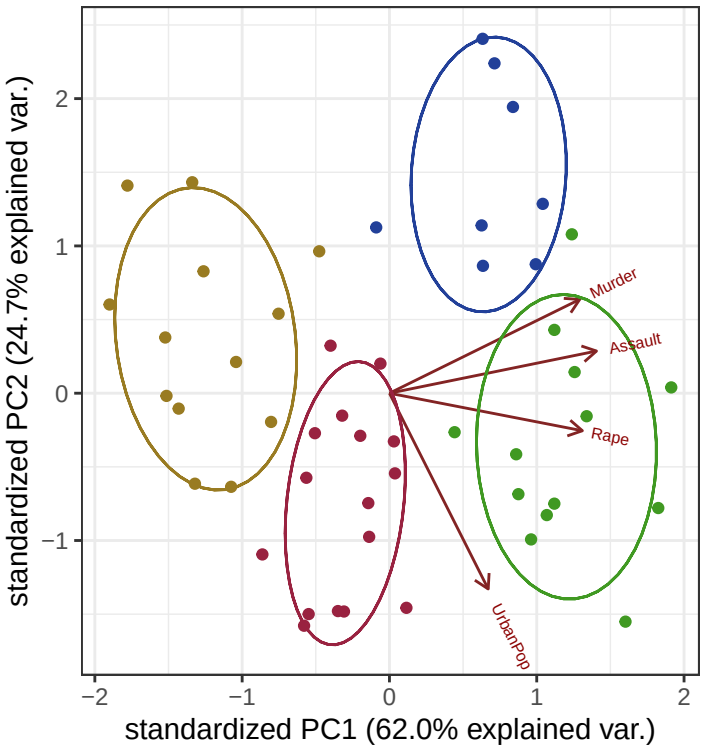
```
ggbiplot(pca_arrests,
         labels = rownames(USArrests),
```

```
        labels.size = 2,
        varname.size = 3) +
 coord_cartesian(ylim = c(-2.5,2.5), xlim = c(-2.5,2.5)) +
 theme_bw()
```

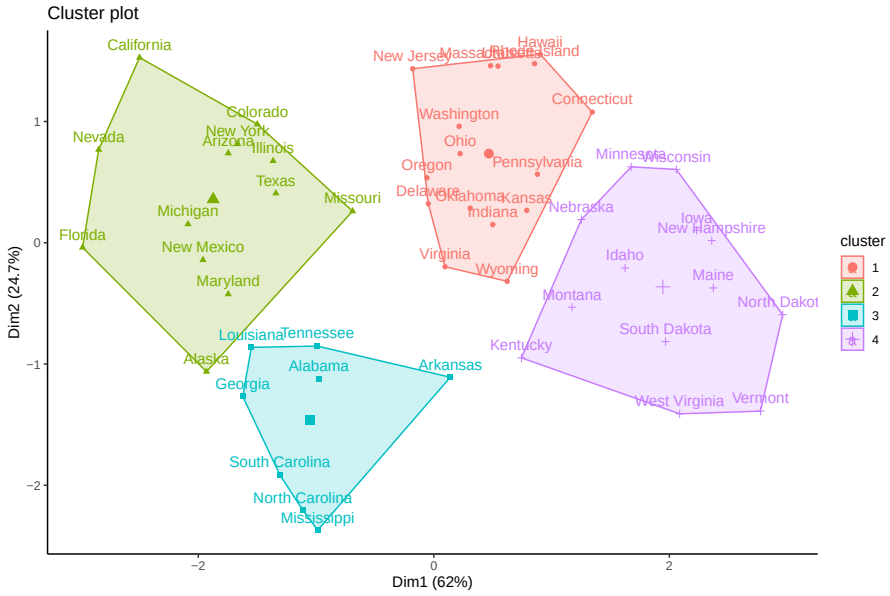


Clustering

```
set.seed(88)
cl_arrests = kmeans(
  df, centers = 4, nstart = 10
)
ggbiplot(
  pca_arrests,
  groups = factor(cl_arrests$cluster),
  ellipse = TRUE,
  varname.size = 2) +
  scale_color_manual(values=c("#992240", "#409922", "#224099", "#997b22")) +
  theme_bw() +
  theme(legend.position = "none")
```




```
library(factoextra)
fviz_cluster(cl_arrests, data = df) + theme_classic()
```



Wine

```
data(wine)
glimpse(wine)
```

```
Rows: 178
Columns: 13
$ Alcohol        <dbl> 14.23, 13.20, 13.16, 14.37, 13.24, 14.20, 14.39, 14.06,
    ~
$ MalicAcid      <dbl> 1.71, 1.78, 2.36, 1.95, 2.59, 1.76, 1.87, 2.15, 1.64, 1
    ~
```

```
$ Ash             <dbl> 2.43, 2.14, 2.67, 2.50, 2.87, 2.45, 2.45, 2.61, 2.17, 2
    ~
$ AlcAsh          <dbl> 15.6, 11.2, 18.6, 16.8, 21.0, 15.2, 14.6, 17.6, 14.0, 1
    ~
$ Mg              <int> 127, 100, 101, 113, 118, 112, 96, 121, 97, 98, 105, 95,
    ~
$ Phenols         <dbl> 2.80, 2.65, 2.80, 3.85, 2.80, 3.27, 2.50, 2.60, 2.80, 2
    ~
$ Flav            <dbl> 3.06, 2.76, 3.24, 3.49, 2.69, 3.39, 2.52, 2.51, 2.98, 3
    ~
$ NonFlavPhenols <dbl> 0.28, 0.26, 0.30, 0.24, 0.39, 0.34, 0.30, 0.31, 0.29, 0
    ~
$ Proa            <dbl> 2.29, 1.28, 2.81, 2.18, 1.82, 1.97, 1.98, 1.25, 1.98, 1
    ~
$ Color           <dbl> 5.64, 4.38, 5.68, 7.80, 4.32, 6.75, 5.25, 5.05, 5.20, 7
    ~
$ Hue             <dbl> 1.04, 1.05, 1.03, 0.86, 1.04, 1.05, 1.02, 1.06, 1.08, 1
    ~
$ OD              <dbl> 3.92, 3.40, 3.17, 3.45, 2.93, 2.85, 3.58, 3.58, 2.85, 3
    ~
$ Proline         <int> 1065, 1050, 1185, 1480, 735, 1450, 1290, 1295, 1045, 10
    ~
```

```
library(ggbiplot)
wine.pca = prcomp(
  wine,
  scale. = TRUE)
ggbiplot(
  wine.pca,
  obs.scale = 1,
  var.scale = 1,
  groups = wine.class,
  ellipse = TRUE,
  circle = TRUE) +
  scale_color_manual(values=c("#992240", "#409922", "#224099")) +
  theme(legend.direction = 'horizontal',
        legend.position = 'top') +
  theme_bw()
```