Quantitative analysis

Quantitative analysis - week 5

Categories of Analysis

Statistical tests are primarily used to investigate sample data - the most imporant use beinh finding out if the sample data is representative of the population

Simple Hypothesis Testing

Statistical tests are used to test simple hypotheses sample data.

- "Sample X is from a different population to sample Y"
- "Sample X has a different mean to the expected mean μ"

With non-sample data, these are easy to test; we could just compare the means of the data sets.

With sample data, we need to use statistical tests to determine if the difference is significant.

For example: if I measure the heights of every person in class A, and every person in class B, and find that the average height in class A is 1.5m, and the average height in class B is 1.6m, is this difference significant?

But what if I only measure 10 randomly chosen people from each class of 40 people?

The sample means I can calculate from this data will be different from the population means.

- Even if sample Mean A is higher than sample Mean B, this could be due to random variation in the sample data. It doesnt necessarily mean that the population means are different.

Sample vs Population Means

We can usually only observe sample means - but we use sample data to calculate the probability overall population mean is within a certain range.

We can never be totally certain, but we can estimate how confident we are that a set of samples comes from a given population.

By using confidence levels like 95% or 99% confidence, we can then reject or accept hypotheses about our data.

1 time in 20 with 95% confidence we will be wrong, and with 99% confidence, 1 time in 100 we will be wrong.

Alot of talk about histograms and normal distributions, with an example of class A and B with height.

Of a normal distribution when you calculate the probability of someone fitting the distribution, you can say example $p = 0.15$ which is 18% chance of someone being 1.5m tall for example.

Social scientists usually use a significance level of 0.05 or 0.01 for p-values, which means that if the probability of the data being from the population is less than 0.05, then we can reject the null hypothesis. If it is less than 0.01, we can reject the null hypothesis with 99% confidence. Some social scientists might also be manipulating this value to get the results they want. Which is called p-hacking.

If we reject the null hypothesis, we can say that the data is significantly different from the population.

If we accept it we can say that the data is not significantly different from the population. If the p-value is higher, we retain the null hypothesis.

But what if we are looking at a set of samples that comes from an expected distribution?

To do this we use Students t-test, which uses the t-distribution to compare the means of two samples. This is similar to the normal population but it changes shape depending on the number of samples. (The degrees of freedom)

The t-distribution is more uncertain you you have few observations: as yo get more data is gets closer to the normal distribution.

Running t-tests

We can use a two-sample t-test to compare the means of two samples. This will find out if theres a significant difference between the two samples. Or if they come from populations with different distributions.

Lets move to Rstudio

```
library(tidyverse)
```

```
Warning: package 'tidyr' was built under R version 4.3.3
```

```
Warning: package 'readr' was built under R version 4.3.3
```

```
Warning: package 'dplyr' was built under R version 4.3.3
```

```
Warning: package 'stringr' was built under R version 4.3.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

First of all, let's load in the population data we've been looking at.

```
height.data <- read_csv(file = "class_heights.csv")
```

```
Rows: 80 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): class
dbl (1): height

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Let's examine the data we just loaded.
glimpse(height.data)
```

```
Rows: 80
Columns: 2
$ class  <chr> "classA", "classB", "classA", "classB", "classA", "classB", "cl~
$ height <dbl> 185.7, 173.5, 170.6, 174.5, 175.7, 170.5, 174.0, 168.9, 166.4, ~
```

```r
summary(height.data)
```

```
    class               height     
 Length:80          Min.   :162.0  
 Class :character   1st Qu.:170.5  
 Mode  :character   Median :174.9  
                    Mean   :175.0  
                    3rd Qu.:179.2  
                    Max.   :192.7  
```

Of course, we won't actually be looking at the entire population datafor these groups - we need to take a sample.

Tidyverse has a function slice_sample() which will do exactly that for us - it will randomly select the required number of samples from a larger data set.

Since we need 8 samples from each class, we do two sampling operations, and combine them into a single data frame with the bind_rows() function.

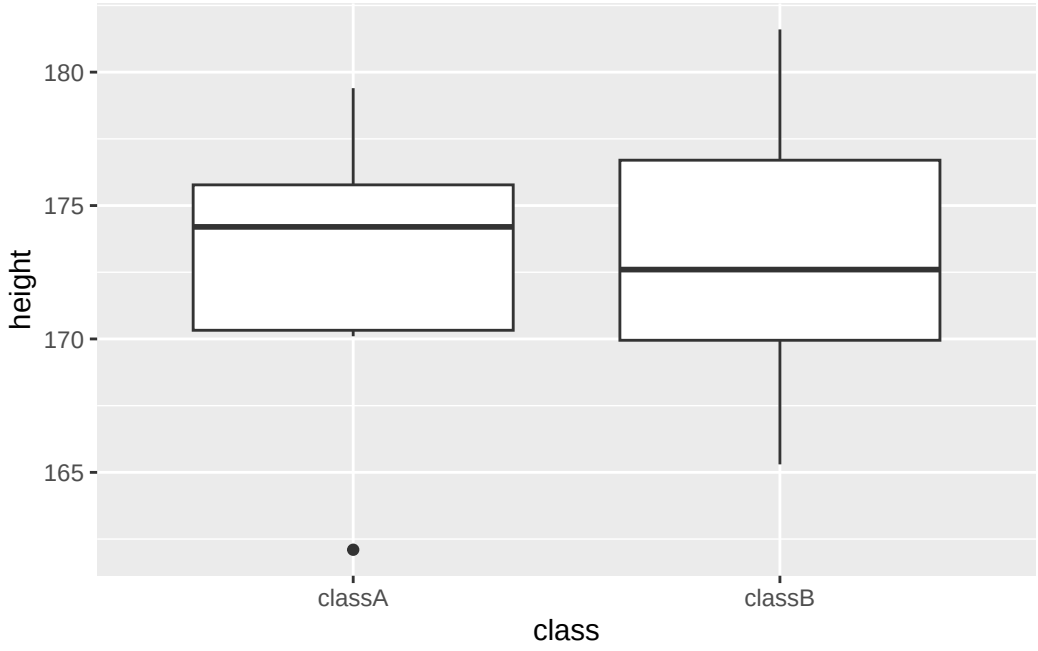
```r
height.samples <- bind_rows(slice_sample(subset(height.data, class == "classA"), n = 8),
                            slice_sample(subset(height.data, class == "classB"), n = 8))

summary(height.samples)
```

```
    class               height     
 Length:16          Min.   :162.1  
 Class :character   1st Qu.:170.1  
 Mode  :character   Median :174.2  
                    Mean   :173.1  
                    3rd Qu.:176.0  
                    Max.   :181.6  
```

We can check out the summary statistics for our samples. Note that because sample() picks a random set of observations, the results from now on will be slightly different for each of you, because we are all working with different random samples from the population.

```r
ggplot(data = height.samples, mapping = aes(x = class, y = height)) +
  geom_boxplot()
```



BOX PLOT

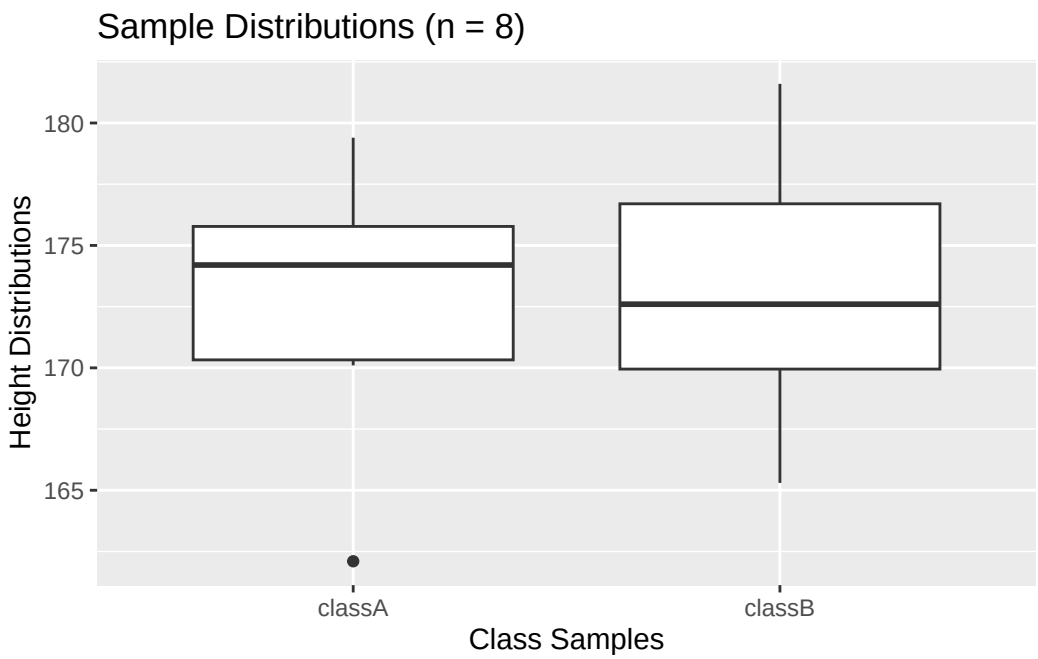
Arguably the simplest visualisation for a set of sample data, the box plot is also very easy to create.

In the ggplot2 package, which is the most popular and flexible way to create graphs in R, we first create a plot with the ggplot() function, giving it an aesthetics object aes() that tells it which data we want to use for the parts of the graph. Then, we can add different kinds of charts to the plot using various functions, or alter things like the axis labels, colour themes, and so on by using other commands. We string these commands together similarly to the piped functions in Tidyverse, but in ggplot2 we use the plus symbol + to add new elements to a plot.

To create a box plot, we give ggplot() the data we want to use, and then specify the mapping - an aes() object telling it which columns to use for the axes. In this case, we want a vertical box plot, so the "class" column (categorical) is on the X-axis, and the "height" column (continuous) is on the Y-axis.

Try swapping the mappings of the x and y axes here, and see what happens?

```
ggplot(data = height.samples, mapping = aes(x = class, y = height)) +
  geom_boxplot() +
  xlab("Class Samples") +
  ylab("Height Distributions") +
  ggtitle("Sample Distributions (n = 8)")
```

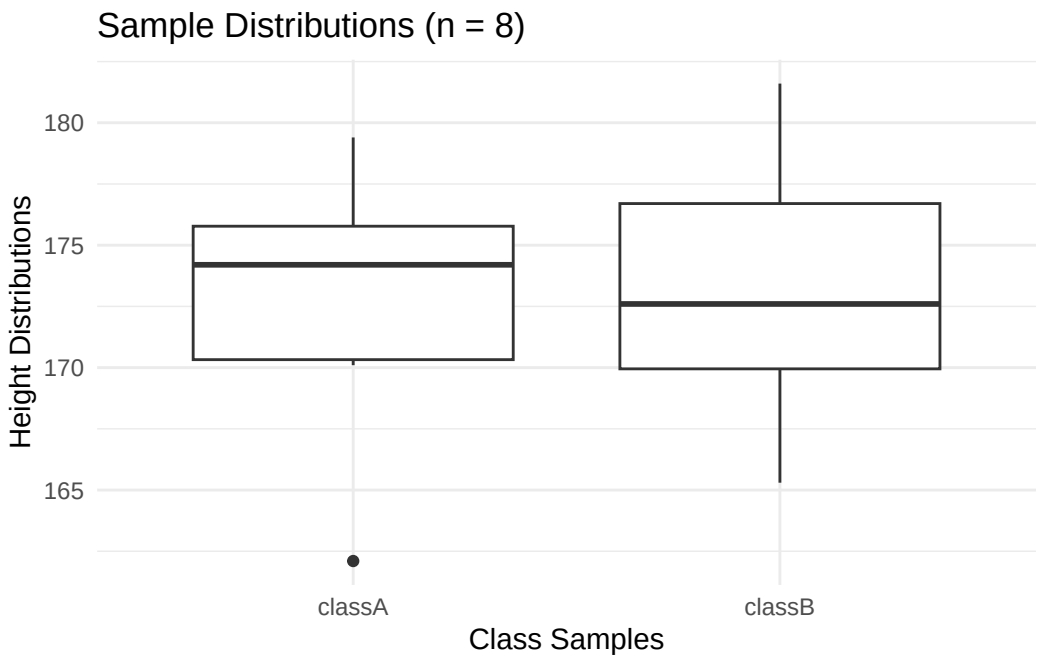


There are tons of other settings you can change or elements you can add in ggplot - we'll see a few more of them as we create different plots in future.

One really useful one that's worth knowing from the start is that there are a set of theme_ functions which apply a consistent theme to your charts.

The default one being used right now is theme_grey(), and others include theme_classic(), theme_dark(), theme_bw(), and theme_light(). I like the theme_minimal() version personally, but try playing with the others and see which one appeals to you or (more importantly) looks good with your data.

```
ggplot(data = height.samples, mapping = aes(x = class, y = height)) +
  geom_boxplot() +
  xlab("Class Samples") +
  ylab("Height Distributions") +
  ggtitle("Sample Distributions (n = 8)") +
  theme_minimal()
```

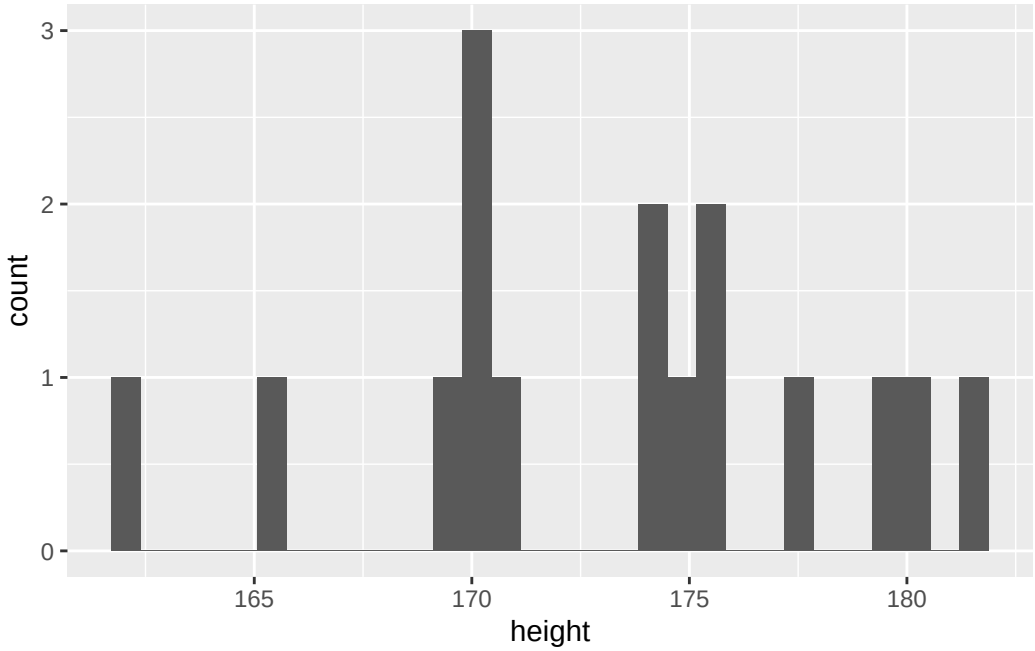


HISTOGRAM

Another really simple and effective way to look at our data is with a histogram that shows the density of observations across the scale of the data.

```
ggplot(data = height.samples, mapping = aes(x = height)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



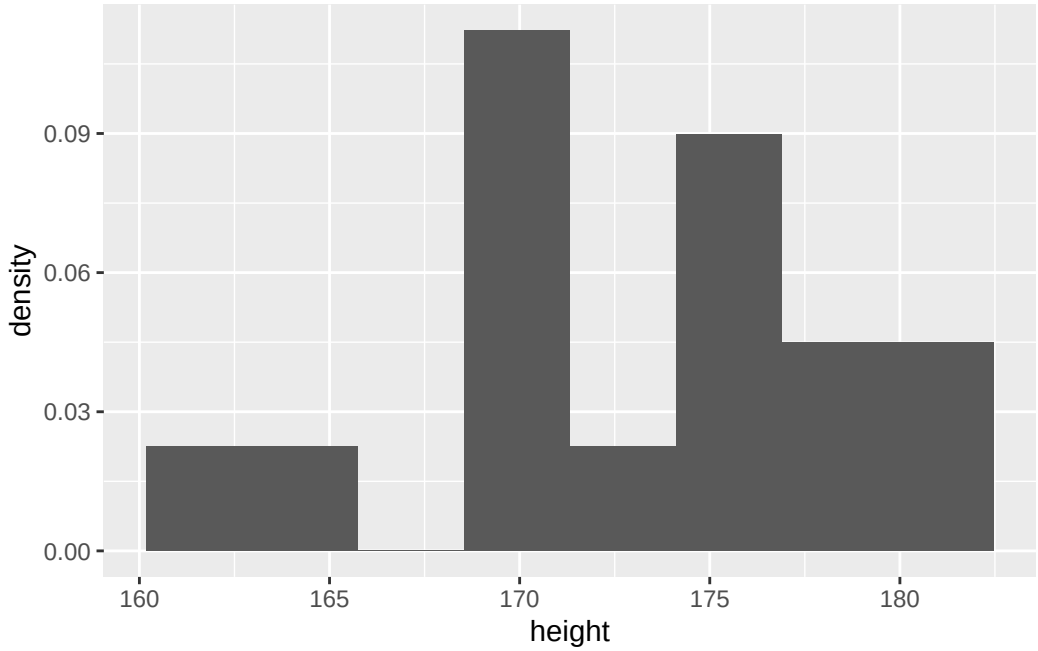
That doesn't look great. The warning message from R hints at the reason; the default setting for histograms is to have 30 "bins", i.e. to divide up the range of the variable into 30 slices and use those as histogram columns.

Since we only have 8 observations of each group, that isn't ideal. We can manually set the number of bins to something more sensible - let's try 10.

Also, ggplot has a special function called ..density.. which will calculate the histogram's scale as proportional density of observations, rather than count of observations. This is usually preferable. We can specify it in another aesthetic aes() object, which this time we give specifically to the geom_histogram() function rather than applying it to the whole ggplot().

```
ggplot(data = height.samples, mapping = aes(x = height)) +
  geom_histogram(aes(y = ..density..), bins = 8)
```

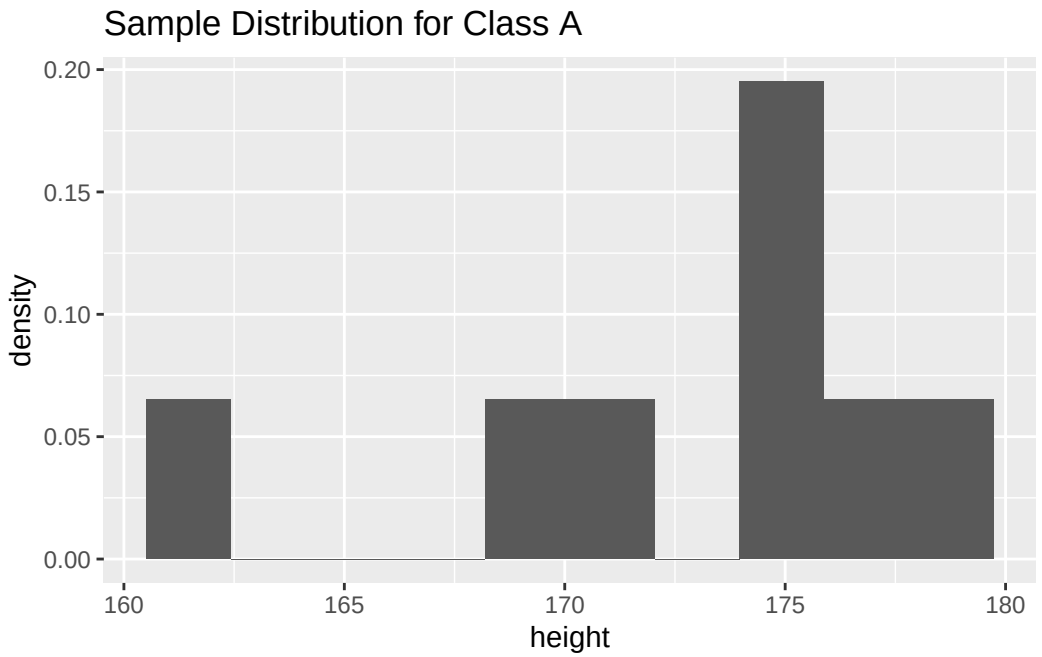
```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```



That's better, and we can see the normal distribution a bit more clearly now. However, there's a problem: we have made a histogram for our entire data set, including both Class A and Class B, rather than for the two classes seperately.

We could fix that problem by using Tidyverse to make a subset of the data before passing it on to ggplot:

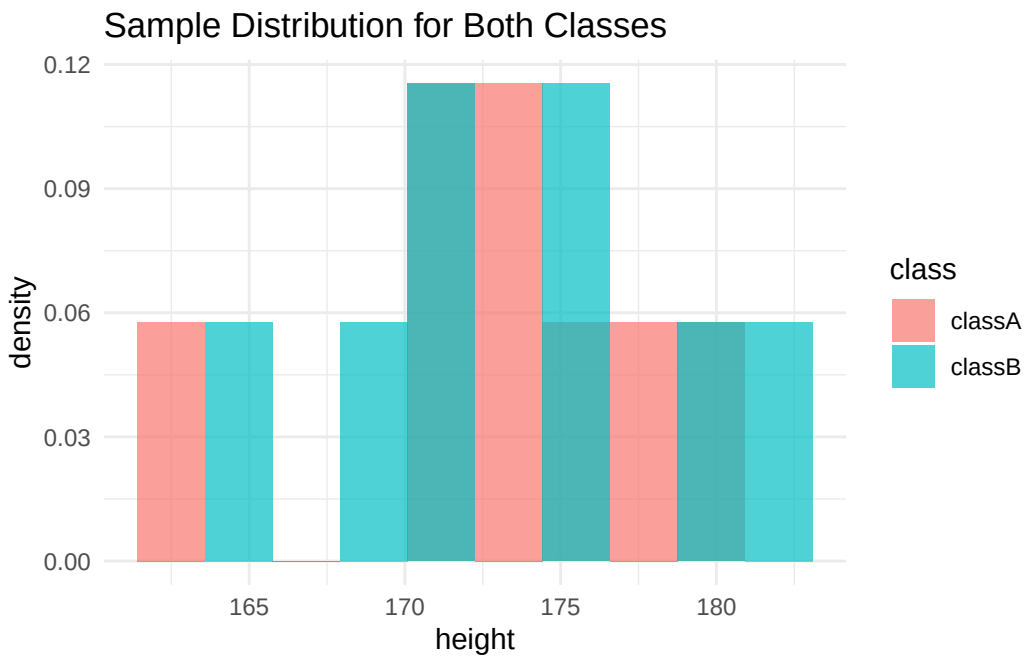
```
height.samples %>%
  filter(class == "classA") %>%
  ggplot(mapping = aes(x = height)) +
  geom_histogram(aes(y = ..density..), bins = 10) +
  labs(title = "Sample Distribution for Class A")
```



Note that the pipe symbol %>% changes to a + when we enter the ggplot part of this chain! It's easy to confuse these when you're mixing tidyverse and ggplot commands, and it will cause an error.

A more elegant way to handle this situation would be to tell ggplot how to divide up the two samples. Here, we tell ggplot that it should set the "fill" of the histogram (i.e. the colour to fill the bars with) according to the class column in the data. Then, when we plot the geom_histogram() object, we tell it to position the two histograms on top of each other by setting the position parameter to "identity", and make each of them 30% transparent so we can see both data sets by setting the alpha parameter to 0.7.

```
ggplot(data = height.samples, mapping = aes(x = height, fill = class)) +
  geom_histogram(aes(y = ..density..), bins = 10, position = "identity", alpha = 0.7) +
  labs(title = "Sample Distribution for Both Classes") +
  theme_minimal()
```



The other options for the position parameter are "stack", which is the default and will stack the values on top of each other vertically, and "dodge" which will print columns from each sample side by side. Try changing the position to "dodge" in the above example and see what difference it makes.

To check the probability that these two samples came from the same population distribution, we use the Two Sample t Test. R has a simple function called t.test() which performs this test.

After all that complicated explanation, it's actually very simple to use - you just show it the samples you want to compare by telling it which column in the data frame has the observations, and which has the category showing which sample they came from.

```
t.test(height ~ class, data = height.samples)
```

```
    Welch Two Sample t-test

data:  height by class
t = -0.20128, df = 13.978, p-value = 0.8434
alternative hypothesis: true difference in means between group classA and group classB is not
95 percent confidence interval:
 -6.411517  5.311517
sample estimates:
```

```
mean in group classA mean in group classB
            172.8625             173.4125
```

The main value we are interested in here is the p-value.

A high p-value means that we cannot reject the null hypothesis, which is that the two samples are from the same distribution, and thus the real, unobserved difference between the population means is 0.

A low p-value - below the significance levels of 0.05 for 95% confidence, or 0.01 for 99% confidence - would indicate that we can reject the null hypothesis, meaning that we are confident our samples are actually different from each other.

In this case, you probably got a high p-value, indicating that the samples really are from the same population.

Note also the 95% Confidence Interval values. This is essentially saying that statistically, there is a 95% probability based on these samples that the true difference in means between the two populations is between these two values. Generally, if the confidence interval includes 0, that means we retain the null hypothesis, because we haven't been able to exclude the possibility that the real difference is zero.

There are two other forms of t test to introduce here.

Firstly, the ONE SAMPLE T TEST.

Up until now, we've been talking about comparing two samples to see if they are significantly different (i.e. from different populations).

What if we instead want to compare one sample against a prior expectation; for example, we have some new survey data, and want to see if it differs significantly from the findings of previous surveys?

In that case, we can t test the sample against an expected mean value, instead of another sample.

Let's say we have been told the average height of students in this faculty is 170 centimetres. Does our sample fit with that expectation?

```r
# Let's test the class A samples...
height.samples %>%
  filter(class == "classA") %>% pull(height) %>%
  t.test(mu = 170.0)
```

```
    One Sample t-test

data:  .
t = 1.5121, df = 7, p-value = 0.1743
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 168.3862 177.3388
sample estimates:
mean of x
 172.8625
```

```r
# ... and the class B samples...
height.samples %>%
  filter(class == "classB") %>% pull(height) %>%
  t.test(mu = 170.0)
```

```
    One Sample t-test

data:  .
t = 1.7318, df = 7, p-value = 0.1269
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 168.7529 178.0721
sample estimates:
mean of x
 173.4125
```

Depending on your samples, you probably got some pretty low p-values for those tests - perhaps even lower than 0.05 (95% confidence) or 0.01 (99% confidence). It seems like our samples contradict the expected mean value!

Remember: the t-test is sensitive to the number of observations you have of your variables, and gets more confident with larger sample sizes.

Let's try combining the observed samples for class A and class B (since they're both valid samples for the faculty overall) and see what happens.

```r
t.test(height.samples$height, mu = 170.0)
```

```
    One Sample t-test

data:  height.samples$height
t = 2.3736, df = 15, p-value = 0.0314
alternative hypothesis: true mean is not equal to 170
95 percent confidence interval:
 170.3201 175.9549
sample estimates:
mean of x
 173.1375
```

Again, the result will depend on your sample - but you probably got a really, really small p-value, meaning that it's extremely unlikely that these samples came from a population with a mean of 170cm.

Can we really say that the 170cm figure for the faculty is wrong, just on the basis of 8 or 16 observations?

If the samples truly were randomly chosen then yes, we can. The t-test and other tests based on population distributions are very powerful tools for this reason: by calculating probabilities that specific sets of observations would appear by chance from a certain population, they allow us to make inferences about large populations from relatively small sets of randomly selected data.

The second variation on the t-test is the PAIRED SAMPLES T TEST.

This is a special case, because it assumes that the two samples you are testing are two observations of the same units / cases.

For example, if you ran an experiment where you measured people's views on a certain topic, then showed them a video, then measured their views again - you would have two observations for each person. Similarly, if you did a panel survey with the same people answering the survey before and after an election, you would have paired samples before and after the election.

In this case, you would use a paired sample t test to see if there is a statistically significant difference between the sample means before and after the "treatment".

For this, we'll need some paired-samples data. Let's load in a dataset of survey data showing respondents' attitudes to a certain political party before and after being shown a campaign ad.

```
survey_scores <- read_csv(file = "survey_experiment.csv")
```

```
Rows: 80 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): survey_wave
dbl (2): subject_id, sentiment

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(survey_scores)
```

```
Rows: 80
Columns: 3
$ subject_id  <dbl> 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, ~
$ survey_wave <chr> "Before", "After", "Before", "After", "Before", "After", "~
$ sentiment   <dbl> 28, 58, 44, 50, 18, 48, 59, 65, 65, 60, 48, 74, 35, 63, 60~
```

```
summary(survey_scores)
```

```
   subject_id    survey_wave          sentiment
 Min.   : 1.00   Length:80          Min.   :  2.00
 1st Qu.:10.75   Class :character   1st Qu.: 48.75
 Median :20.50   Mode  :character   Median : 59.00
 Mean   :20.50                      Mean   : 58.90
 3rd Qu.:30.25                      3rd Qu.: 69.00
 Max.   :40.00                      Max.   :100.00
```

We can see that there are certainly some differences in the samples, but to understand if the treatment actually made a meaningful difference, we have to test for differences on a pairwise basis.

For a paired t-test, we need to have two samples of data - x and y - which are in exactly the same order, so that the observation of case N in x corresponds to the observation of case N in y. This is done easily in R by using the pivot_wider() function from the Tidyverse, which will take a long-form data and turn it into a wide-form data frame with the two samples side by side.

```
survey_scores.wide <- survey_scores %>%
  pivot_wider(names_from = survey_wave, values_from = sentiment)

t.test(survey_scores.wide$Before, survey_scores.wide$After, paired = TRUE)
```

```
    Paired t-test

data:  survey_scores.wide$Before and survey_scores.wide$After
t = -0.85476, df = 39, p-value = 0.3979
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -9.257585  3.757585
sample estimates:
mean difference
          -2.75
```

Here we can see that the p-value is high - well above the 0.05 significance level, meaning that we cannot reject the null hypothesis that these samples are not notably different. It would appear that the treatment did not have a significant effect.

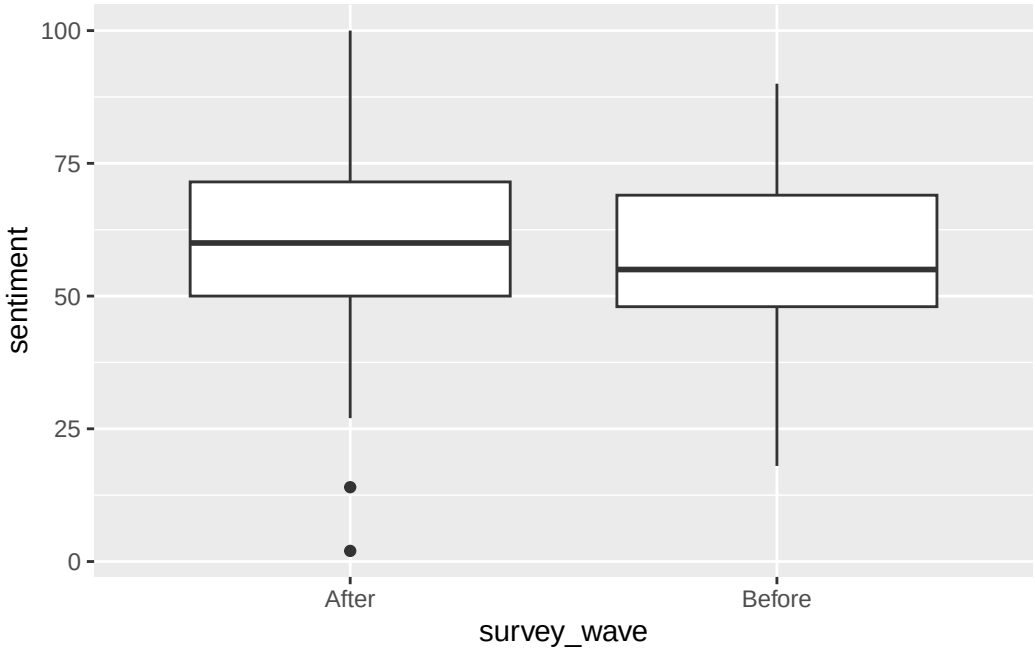
Another Visualisation - PAIRED DATA BOXPLOT

While we're on the topic of paired data, there is a variation of the box plot that is really useful for viewing the change in a population between two observations of the same subjects. This is also a good opportunity to look at how adding elements to a ggplot can gradually build up a complex data visualisation.

Our objective is to create a box plot which also shows lines displaying the change in each data point between the two waves of the survey.

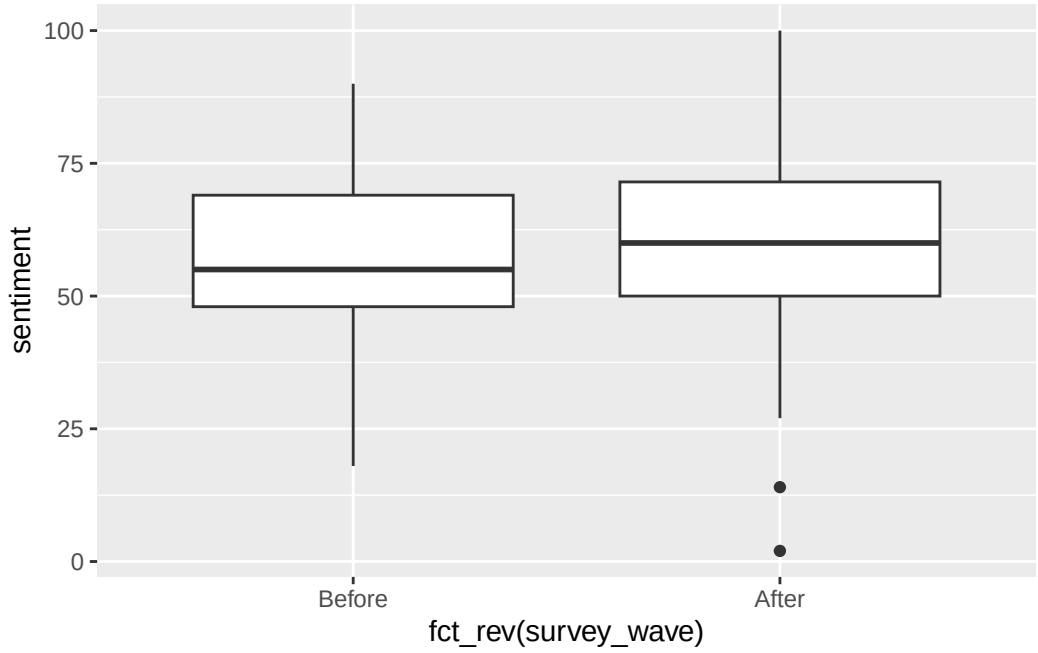
Let's start with a regular box plot:

```
ggplot(data = survey_scores, aes(x = survey_wave, y = sentiment)) +
geom_boxplot()
```

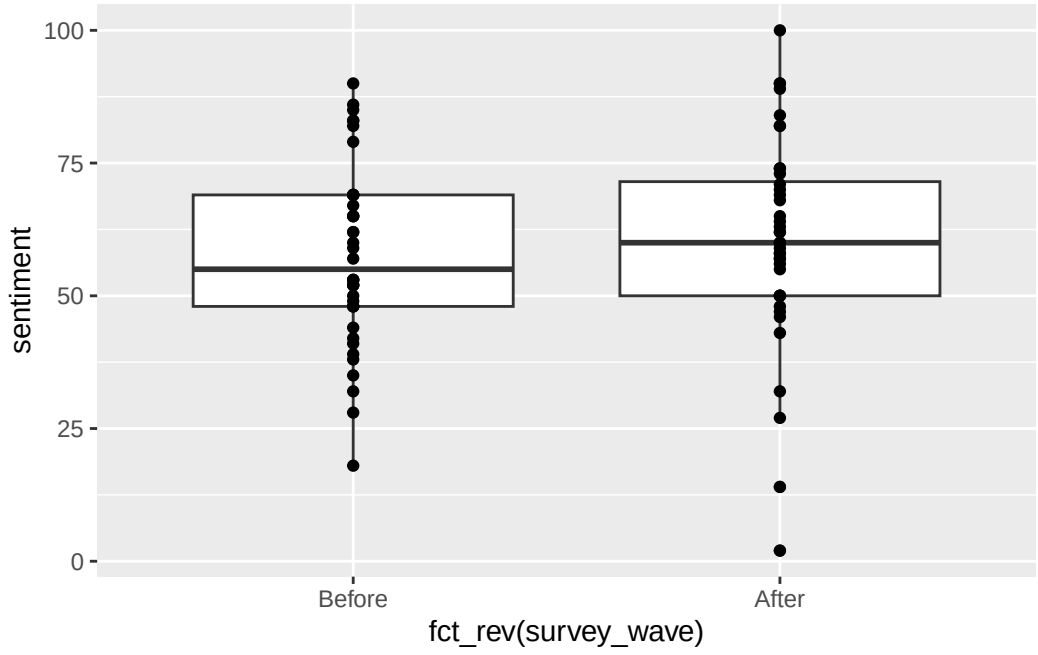


First problem: we want the Before plot on the left, since most people willnaturally read the plot left to right. We can use the fct_rev() function("factor reverse") to turn around the order of this category / factor variable.

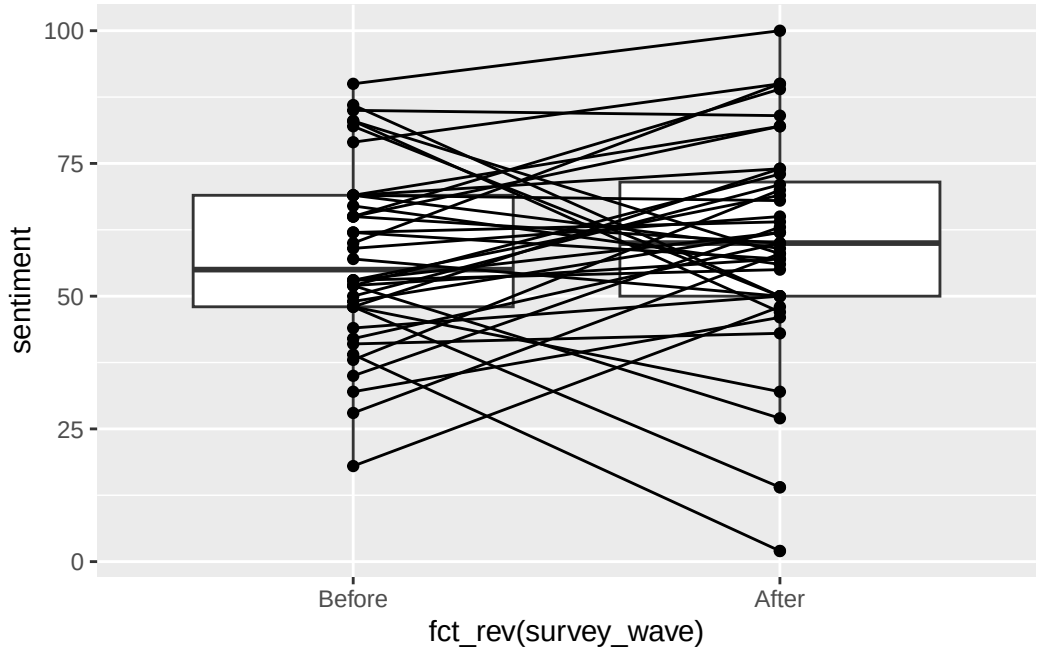
```
ggplot(data = survey_scores, aes(x = fct_rev(survey_wave), y = sentiment)) +
  geom_boxplot()
```




```
ggplot(data = survey_scores, aes(x = fct_rev(survey_wave), y = sentiment)) +
  geom_boxplot() +
  geom_point()
```

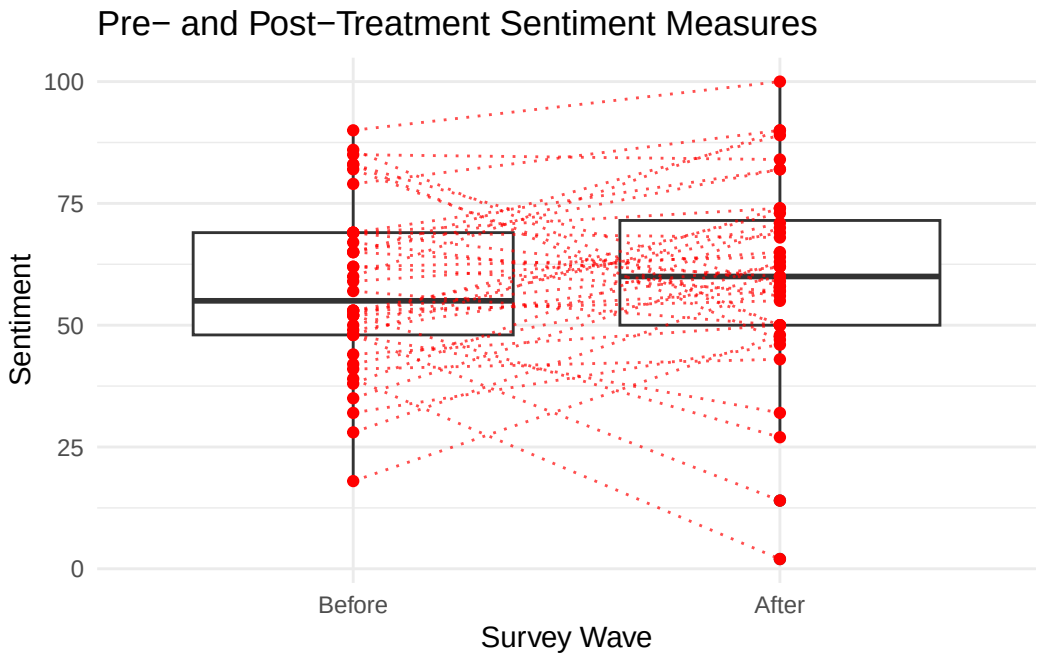


```
ggplot(data = survey_scores, aes(x = fct_rev(survey_wave), y = sentiment)) +
  geom_boxplot() +
  geom_point() +
  geom_line(mapping = aes(group = subject_id))
```



You can see how by putting together three different types of chart - a box plot, a scatter / point graph, and a line graph - we have created a more complex visualisation. The next step would be to play with aspects like theme, colours, and transparency to make the graph readable and attractive. Here's my attempt at it.

```
ggplot(data = survey_scores, aes(x = fct_rev(survey_wave), y = sentiment)) +
  geom_boxplot() +
  geom_point(color = "red") +
  geom_line(mapping = aes(group = subject_id), color = "red", alpha = 0.7, linetype = 'dotte
  labs(title = "Pre- and Post-Treatment Sentiment Measures", x = "Survey Wave", y = "Sentimer
  theme_minimal()
```



Something you can see clearly from this graph, which might not have been so easy to understand just from the statistical test, is that the treatment seems to have been very polarising - it did improve sentiment among most users, but there were a substantial minority of users whose sentiment dropped dramatically after seeing the campaign ad.