Estimation Statistics

Dan MacLean

2022-05-01

Table of contents

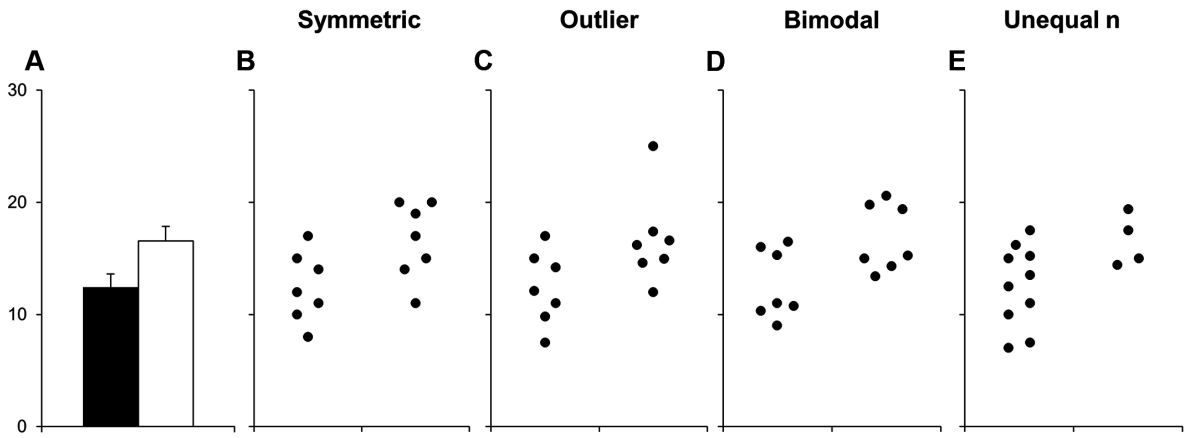
Motivation

Variability in measurements

Variability in measurements is a thing that happens as a natural consequence of working with complex systems that are affected by many variables in stochastic ways. Biological systems are some of the most variable we know. The variability in our experiments could be a function of the behaviour of the system yet it is common practice to hide that variability when we start to analyse our data by using summary plots like box-plots. Ultimately, that's bad news for our science, because the variability could be telling us something.

Summarising your data can lead to wrong conclusions

We all know that when you create a bar chart and put some error bars on it, you're really only representing two numbers, usually a mean and standard deviation. People create bar plots out of habit and without thinking about it, and in doing so can miss important stuff. Look at this figure from Weissgerber (2015):



Test			p value	
T-test: Equal var.	0.035	0.050	0.026	0.063
T-test: Unequal var.	0.035	0.050	0.026	0.035
Wilcoxon	0.054	0.073	0.128	0.103

Figure 1: Weissgerber et al

The bar chart in panel A is one that emerges if we use each of those data sets in the other panels. But creating the bar chart really hides some important stuff, like the fact the numbers are clearly separating into two groups in panel D, or that the two samples have different sizes in panel E.

Worse than any of these is that the significant difference in the t-test is coming from just one point in panel C. From this data set you might be tempted to conclude that there is a significant difference in the two samples and if you relied on the bar chart as a visualisation then you'd never suspect there was something funny.

Some enthusiastic young science communicators have even started a Kickstarter to lobby journals to stop using, in particular, bar charts! These people, calling themselves Bar Barplots, have a nice video on one of the main problems with bar charts. Have a look at this page on Kickstarter . Kickstarter - Barbarplots, especially this video Kickstarter - Barbarplots video.

Ignoring your data visualisation and just making bar plots could be an error! It's important that you spend a little time getting to know, and presenting your data as clearly and thoroughly as possible.

p - one value to fool them all?

A lot of researchers get the impression that *t*-tests, ANOVAs and other hypothesis tests tell you whether something is significant with probability p. This is quite a misinterpretation. They do no such thing. More accurately but still rather informally and in general p can be taken as follows

> **!** What p is
>
> p is the likelihood of seeing a difference of the size observed in the Null Model

Remembering that the Null Model is the default situation where the difference is 0 then this is resolutely not the same as saying they are definitely different. Just that they're not likely to be the same. The p in p value is usually taken to mean 'probability', but if it stands for anything it should be 'probably not the same'.

Hypothesis testing like this has been criticised for being weak inference, and not without reason. All this means is that we need to be awake to the limitations of our methods.

p-values may well be the most abused statisic in the world, and it is probably time to retire it as a go-to statistical tool. In the end won't a *p*-value help you see real differences and make this all easy? Sadly, that isn't true. Let's do an experiment to test that.

Ten Thousand Random Numbers

Below is a set of figures that show different views of the same set of data. Every frame of the 100 frames shows a different sampling from the same pool of 10,000 random normally distributed numbers.

Step-by-step, here's how these figures are made.

1. Generate a pool of 10,000 random numbers (mean 5, SD 1)
2. From that, select 10 and call it sample 1.
3. Select another 10, call it sample 2.
4. Draw plots comparing each sample
5. Do an independent *t*-test on the sample 1 and sample 2 to test for significant differences in means.

The figures are plotted with a red border if p comes up less than 0.05. The thing is, the samples are from the same background pool, so intuitively you might suspect that none should be different from the others. The reason that some of them do is because a p value only states that the difference observed occurs by chance in p of all events, so for 100, we'd expect 5 to be marked out by chance. In this run of the experiment we get three. Here's a couple:

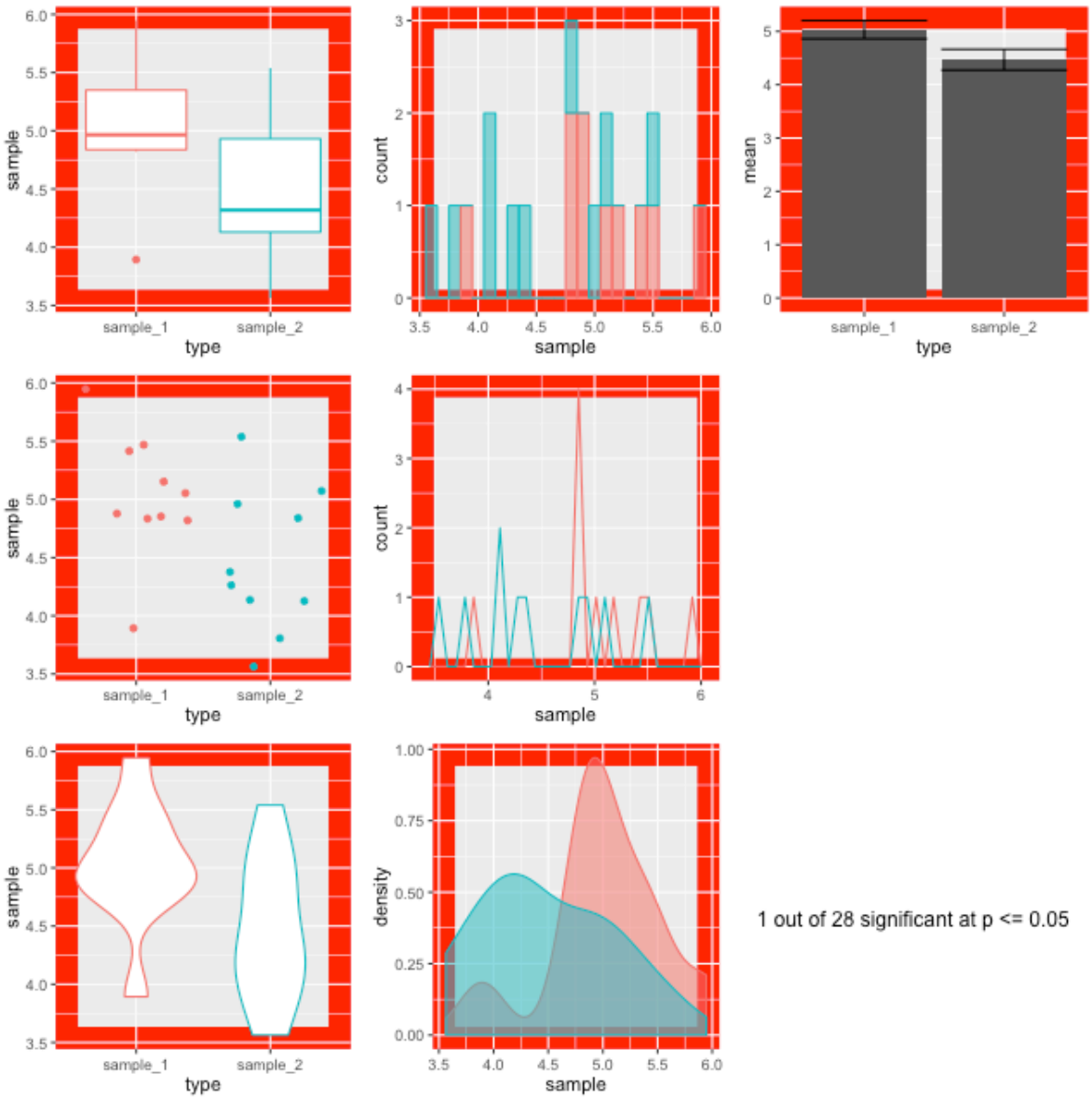


Figure 2: Run 28

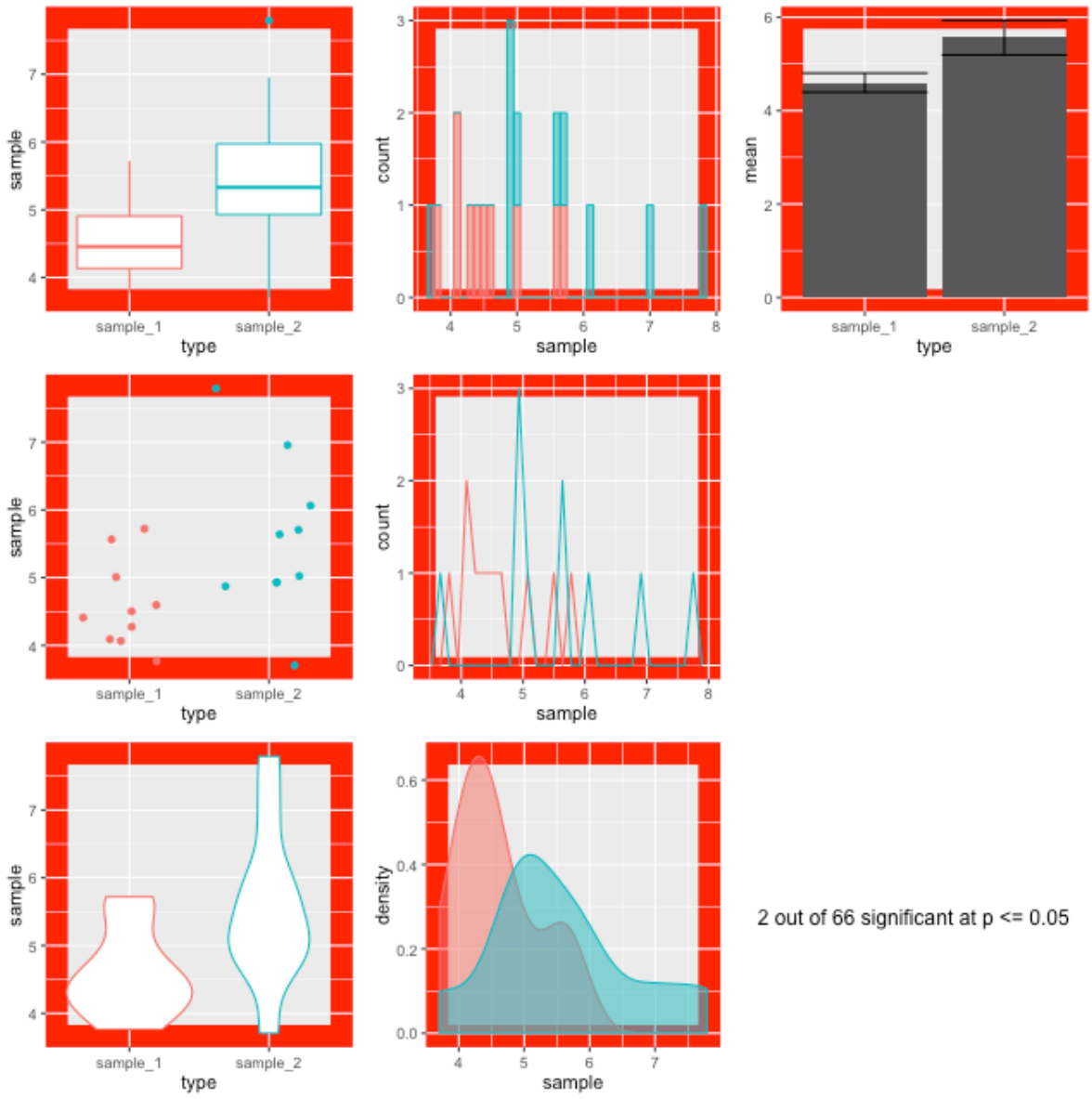


Figure 3: Run 66

Look at the different plots for each. It is observable that for all these the barplots look very convincingly different. But in the context with the other plots its clear that they aren't showing the whole of the story (or in fact much of it). The boxplots (top left) do a good job of showing the range and the violin and density plots (bottom row) do a good job of showing the shape. It is only really the point plot (first column, middle row) that reveals the positions of the data points and shows that the conclusion of the p value is likely skewed by one or two points in

each sample. Concluding differences on this basis is *really* unsafe.

Hence, the conclusion from this is that a range of visualisations is necessary to allow us to have confidence in our p values and understand the shapes of our data. Drawing box plots and sticking to p religiously is going to make us wrong more than we'd like!

Estimation Statistics

So what can we do about this apparent problem? One strategy is to take a much less hypothesis test-centric approach to statistical analysis and apply other ways of thinking about the problems, instead we can use Estimation Statistics to look at the parameters of our data and get an idea of magnitude of an effect or difference and an associated confidence interval that estimates how believable that estimate is.

> **i** Are you repeating this?
>
> Hey, isn't all this rationale in the ggplot book? Umm, it is a bit, yeah! Graphical methods and Estimation Statistics are closely related because plots can show the parameters very well

In this course we'll take a look at some of the main features of Estimation Statistics with the intention that they'll be useful in your future research. The same rule about taking a p-value as your only determinant of whether a statistical claim is important or significant applies to the Estimation Statistics we'll learn here. At the end of this book we will examine the integration of the different estimate statistics covered.

References

1 Effect Size

1. Questions

- How do we tell whether a difference between groups is big or small?

2. Objectives

- Understand what an effect size is
- Understand how to compute effect sizes for different data

3. Keypoints

- Effect sizes tell you whether a difference between groups is significantly big

The usual p-value based statistics are an attempt to determine whether a sample is significant in terms of how frequently one like it occurs. If it doesn't occur often, it is significant. But that isn't the only way to think about a sample being significant. We can also ask whether the difference is meaningfully large, this is what the effect size hopes to capture

Effect size is of the three central ideas in Estimation Statistics. In essence an effect size is just the 'size of the effect' or in other words the size of the difference between groups or the strength of a relationship. It is an essential component in assessing the strength of a relationship between variables and a critical tool in working out whether a statistical claim (like significance) is valid or not.

You will already be familiar with the most common effect sizes, these are common metrics like the sample mean difference, the regression coefficient (r) from a regression analysis or the likelihood of an event.

1.1 Difference in sample means

The most commonly seen effect size is the difference between sample means. Despite the apparent simplicity and to an extent - obviousness - of the difference in the means of two groups this parameter is a very useful tool in determining whether or not a claim of difference or significance is valid or not.

One advantage and disadvantage of using the difference in sample means is that to have a good intuition about the claim of significance we must have some prior experience about whether

the observed effect size is a big or small one. Large effect sizes are more significant, all other things being equal, but the absolute effect size is highly dependent on the specifics of the domain we are working in. For example 10g weight difference between two groups of adult humans is not going to be seen as significant, however 10g in difference in groups of adult domestic mice is quite large.

If we're going to be formal about it, then we can write the absolute effect size as follows: Where n_i is the sample size for group x_i

$$\frac{\sum x_1}{n_1} - \frac{\sum x_2}{n_2}$$

Which looks more complicated than it is. The formula just describes the mean of the second group subtracted from the mean of the first group. We can simplify each term thus

$$\bar{x_i} = \frac{\sum x_i}{n_i}$$

Such that the formula for effect size becomes

$$\bar{x_1} - \bar{x_2}$$

And $\bar{x}$ which is pronounce 'x bar' is a pretty standard name for a sample mean in the statistical literature.

1.1.1 Standardised Effect Sizes

It would be extremely difficult to say whether even a 100g difference in human weight is as of as great significance as 10g in mice. Given that the real world meaningfulness of the mean sample difference is dependent upon the context we often need a mechanism through which we can compare effect sizes. Standardised effect sizes can help with this, and they work by taking into account the pooled standard deviation of the two samples, expressing the mean difference in terms of the variation in the numbers that make up the mean. If s is the pooled standard deviation then our standardised effect size d is

$$d = \frac{\bar{x_1} - \bar{x_2}}{s}$$

Computing this for our mouse and human data would enable us to make reasonable comparisons between the significance of the effect size in the different experiments.

The standardised effect size was introduced by Jacob Cohen and the number is known as Cohen's d. Cohen also suggested descriptions of the different values of d and the effect size:

Effect.Size	d
Very Small	0.01
Small	0.20
Medium	0.50
Large	0.80
Very Large	1.20
Huge	2.00

So if we get a d of 0.7, we have a medium effect size.

> **i** Computing s in real life
>
> Computing s is done according to a formula called the pooled standard deviation for two independent samples. If you don't look at lots of formulae it looks a little scary at first glance so I haven't included it here, I mention it just to point out that it *isn't* the same as adding up the two individual sample standard deviations.

1.1.2 Calculating Mean Difference Effect Sizes in Practice

Thankfully there are functions in R that we can use to calculate each of the quantities we have mentioned straight from the data we collect, so we don't need to know all the formulae off the top of our heads. We can use the `effectsize` library for this. First though, let's generate some samples of data from a random normal distribution with mouse sized and human sized means (measuring their weight in grams), but a standard deviation that is the same proportion of the mean in each

Domestic mice are about 20 g in mass.

```
set.seed(123) # ensure random numbers are identical every time
library(effectsize)
x_mouse <- rnorm(10, mean = 20, sd = 20 / 3 )
y_mouse <- rnorm(10, mean = 10, sd = 10 / 3 )
cohens_d(x_mouse, y_mouse)
```

```
Cohen's d |       95% CI
------------------------
1.91      | [0.82, 2.97]

- Estimated using pooled SD.
```

So we get an effect size which is pretty large!

Now let's try the human data, humans are about 65 kg in mass (depending on where you measure!)

```
x_human <- rnorm(10, mean = 65000, sd = 65000 / 3 )
y_human <- rnorm(10, mean = 32500, sd = 32500 / 3 )

cohens_d(x_human, y_human)
```

```
Cohen's d |       95% CI
------------------------
1.34      | [0.34, 2.30]

- Estimated using pooled SD.
```

The human effect size is of similar magnitude to that of the mouse, that is 'large', from intuition on both sets of measurements we can see that the halving of mass is a big effect, so it matches up. Let's try the human measurements at a mouse size change

```
x_human <- rnorm(10, mean = 65000, sd = 65000 / 3 )
y_human <- rnorm(10, mean = 64990, sd = 64990 / 3 )
cohens_d(x_human, y_human)
```

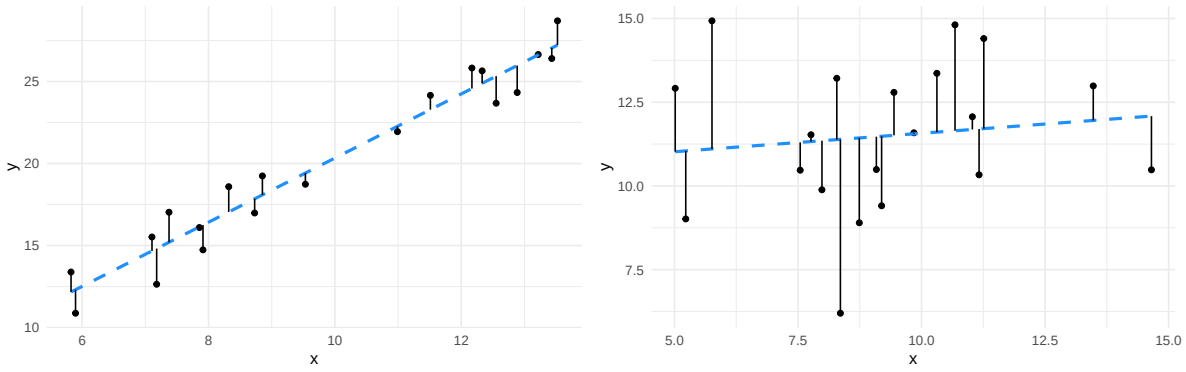
```
Cohen's d |        95% CI
-------------------------
-0.24     | [-1.11, 0.65]

- Estimated using pooled SD.
```

As expected the effect size reduces to small.

1.2 Explained variation

Another common type of effect size is that computed for correlation style data (like that we see in linear regression models), so when we have a continuous x (explanatory variable) and a response . These effect sizes are based on the amount of the variance that is captured by the model (like a linear model). In other terms we've thought about explained variation as the fit to the model.



We can visualise this in a scatter plot. The greater the explained variation, the better the fit to the line.

It is clear to see that the line (and therefore model) fits the data in the left panel much better than the line (model) fits the data in the right panel. The effect size of the better fit model is going to be larger.

1.2.1 Pearson's r

Pearson's r value is one we are likely familiar with from correlation analysis and is a simple effect size we can use with x and y continuous data. It runs from -1 to 1 with values around 0 indicating a smaller effect size and values at -1 or 1 indicating larger effect sizes. The sign (+/-) of r indicates only the nature of the correlation, not the effect sizes, so -0.3 and + 0.3 are equivalently sized negative and positive correlations.

The value of this effect size is interpreted differently from that of Cohen's d, as the values can only run from -1 to 1, here's a brief table of categories

Effect.Size	r
Small	0.1
Medium	0.3
Large	0.5

These values vary from domain to domain, in some domains we would expect a much stronger correlation and correspondingly larger values of r to give the same description of an effect size. Consider correlations between performances of machines like car engines, it'd be very surprising if they didn't correlate in the very high 0.9s, whereas correlations of biological measurements would be good at a much lower level. The values given in the table above were stated by Cohen (again) for the Social Sciences. You'll need to make conclusions judiciously

and with an informed mind (and again in conjunction with other measures) for the domain you are working in.

As it is fundamental, calculating Pearson's r is easy in R. Let's use the data from the plots above to run through it. The left plot data is in a dataframe called `df1`, the right plot data is in a dataframe called `df2`. The function we need is `cor()`, which is part of base R.

```
cor(df1$x, df1$y)
```

```
[1] 0.9716088
```

Which gives a high correlation indeed. For the less well fitting data we see this

```
cor(df2$x, df2$y)
```

```
[1] 0.1213444
```

a much lower Pearson's r.

1.2.2 r^2

A related effect size is r^2 which is literally $r \times r$. It measures the proportion of variance shared between the two variables under examination, so can be interpreted as the amount of variance explained. This one naturally runs between values of 0 and 1 so loses the information about direction of correlation.

1.3 Using the effect size

At the beginning of this course we cautioned about using p-values as the sole measure to decide whether a claim about differences is significant or important. The same caution applies to the effect size, whether you use Cohen's d, Pearson's r or some other measure. There's no cut-off that always makes sense, so never use it on it's own. Use your expertise and other measures we'll see later (and including but not limited to hypothesis tests and p-values) to make data informed interpretations about your results and never rely on arbitrary cut-offs. At the end of this book we will examine the integration of the different estimate statistics covered.

1.4 The assumptions of effect sizes

As we've discovered before, statisticians make assumptions about data when discovering statistics. The standard assumption is that the data and the variance are Normally Distributed (so follow the Normal Distribution). The effect sizes we'be discussed here make that assumption in the calculations too. Practically, it means that the further you get away from a 'Normal' situation the less use the named effect size formula will be. There are other effect sizes for non-Normal data, notably the Spearman's Rank r and Kendall's τ for ranked and categorical correlations. Sometimes the raw mean difference is the only practical measure of effect size.

Although it is important that you are aware whether your data are close to Normal or not, the problem isn't always drastic and in combination with other measures we can reduce misuse of single statistics. In the next steps will look at ways of dealing with arbitrary distributions with Estimation Statistics.

> **i** Roundup
>
> Effect Sizes are a good estimation statistic to use to give you an idea of whether an effect (change) is significant in terms of magnitude (rather than merely frequency of occurence). In conjunction with other statistics it can help fill out a story about your groups of data

2 Confidence Intervals

1. Questions

- How do I estimate what range a statistic (e.g a mean) lies in most of the time?

2. Objectives

- Understand that statistics like sample means are estimates from small sets of observations
- Understand how to compute confidence intervals for means from normal and other distributions

3. Keypoints

- A confidence interval is an estimate of the position of your statistic
- Overlaps between different groups confidence intervals suggest similarity of the statistic

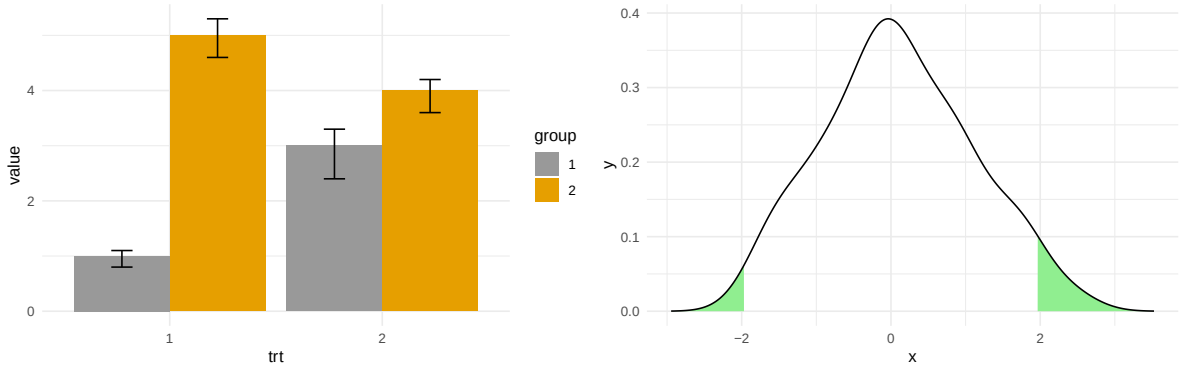
A commonly forgotten fact about measured statistics of a sample like means or medians is that they are only estimates of a real (but somehow unknowable) quantity. Consider trying to calculate the mean height of all trees, this would be highly difficult so we would take a sample and calculate the mean of that hoping that the sample mean would be a good enough representation. In other words we would hope that the sample mean estimates the real mean well enough. But how sure can we be of the sample mean? We can use a confidence interval to give us an idea of the likely range that the real mean falls into.

A confidence interval (CI) is a range estimate of a statistical parameter (a number or measure of some sort) that is computed from the data. It gives a range that the parameter falls in with a certain confidence (the confidence value), so a 95% Confidence Interval of the sample mean tells us the range in which we believe the sample mean will lie.

We often see them rendered like this

But wait! Isn't the first one error bars? Isn't the second one percentiles of a distribution? That's right! They might be. In fact the visual glyph used for confidence intervals and error bars or confidence intervals and percentiles are the same. This is because they are each a kind of numeric range and we can show them on figures in the same way. We must be careful not to confuse them because they seldom show the same thing. To be explicit again confidence intervals show the likely range that a parameter (like a mean) falls in.

Error bars usually show a sample mean plus or minus a fixed number like the standard deviation, so they show us the spread of the data but this doesn't necessarily say anything about



the confidence we have that the mean falls in that range. Confidence Intervals are explicit about this.

Similarly, percentiles show how much of the observed data fall in a range, e.g the 10th percentile shows where 10% of the data fall, but they again don't necessarily say anything about the confidence we have about where a particular parameter falls.

2.1 Confidence intervals on Normally distributed data

A common use of confidence intervals is in conjunction with the Normal Distribution as a Null Model. In which we mean we assume that the variability of a measurement, e.g mouse mass is Normally distributed. As the Normal distribution is well understood we can use its characteristics to get a CI for the mean of a sample.

Let's invent some mouse weights and put them in a vector `x`.

```
mouse_wts
```

```
 [1] 19.57635 29.59349 23.89572 17.75500 11.47902 29.90031 26.04292 30.18206
 [9] 30.28894  8.09662
```

We can use the `qnorm()` function to get the information we need to calculate the limits of a given CI on a standardised Normal Distribution. We don't provide the CI size (that would be too easy!)

Instead we use `qnorm()` to get the value at a given percentile.

As we want a 95% CI and because a 95% confidence interval has a total of 5% that must be equally shared outside of each end of the interval, we use half of 5% (2.5%) as the limit of the percentile. To confuse matters further we must provide it as a probability to this function so the actual percentile we ask for is 0.975.

We can then plug this into a standard formula to get the half width of the CI using the mean and standard deviation of the sample.

Finally, we can add and subtract this value from the sample mean to get the lower and upper bounds of the CI we want.

This isn't quite as complicated as it reads, here's the code

```
x_bar <- mean(mouse_wts)
s <- sd(mouse_wts)
n <- length(mouse_wts)

half_width <- qnorm(0.975) * s / sqrt(n)

left_bound <- x_bar - half_width
right_bound <- x_bar + half_width

left_bound
```

```
[1] 17.625
```

```
right_bound
```

```
[1] 27.73708
```

So we calculated that the 95% CI of the mean of mouse weights runs from 17.6250015 to 27.7370849.

The derivation of this formula is generally available in every statistics text book, so I won't repeat it here, I'll leave it to the interested reader to follow up if they wish.

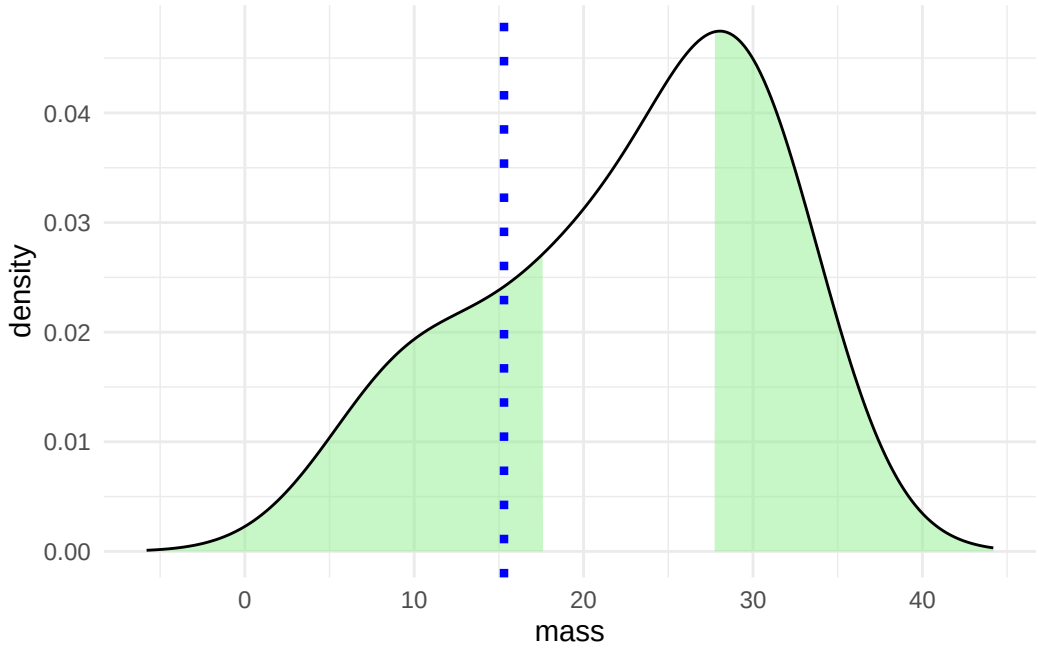
2.2 Using a CI

Now that we have a CI constructed we can use it. One fact about CIs calculated on a sample is that the true mean of the population (E.G the weights of all mice in the world, not just the ones we sampled) falls within an x% CI x% of the time. We can use this fact to estimate whether another measurement like the mean of another sample falls within a CI. If it *doesn't* then we can say the mean of the second sample only occurs in samples from the population (100 - x%) of the time. So if another sample mean falls outside of our 95% CI we can say that two samples with means this far apart only occur by chance less than 5% of the time.

This is actually analogous to how a hypothesis test works, particularly t-tests. So the interpretation of a difference in populations is analogous.

Here's a graphical representation of that on our mouse data

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



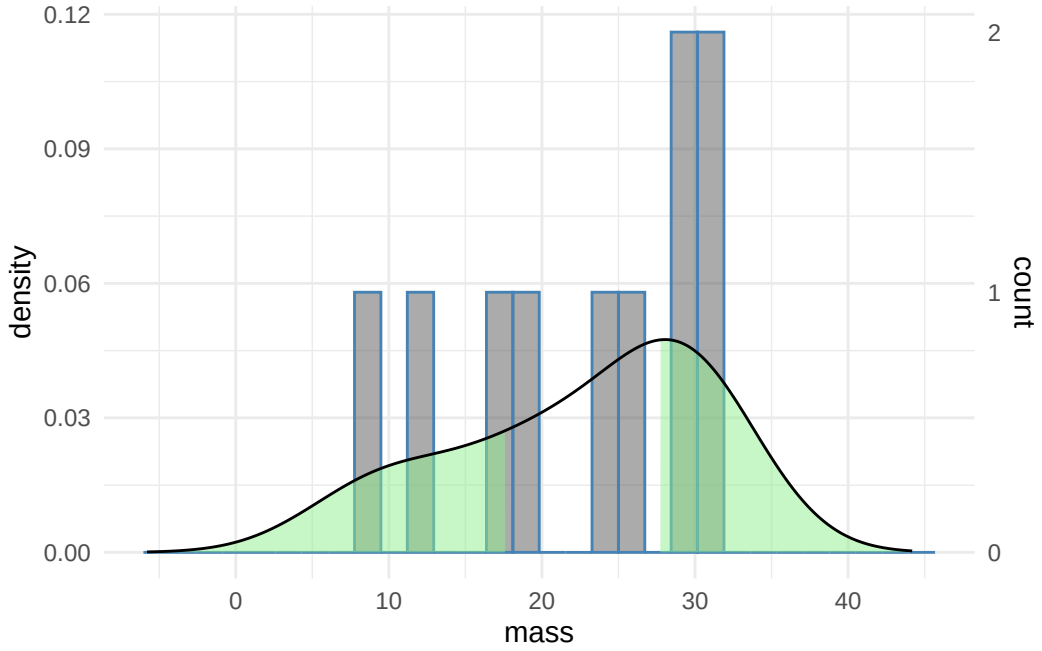
The dotted blue line represents a measurement of a single mouse, the green areas in the density plot are those outside the 95% CI we computed. So the specific measurement looks like it wouldn't come from our population very often.

2.2.1 Confidence Intervals in practice

Of course, looking at the density curve we see that it is not very convincing as the curve of a Normal distribution and the areas outside the CI seem wider than we might have expected from textbook renderings of a Normal distribution. The numbers for mouse weights were drawn randomly from a Normal distribution and the off pattern is due to the small sample size, we only used 10 mouse weight measurement. As a result the estimation of the curve is quite lumpy. Such a phenomenon not only highlights the need for sensibly sized samples but also indicates the importance of careful examination of the data on which we are creating confidence intervals. To repeat our familiar refrain you can't use a single measure on its own. Other measures should be integrated and confidence intervals are best interpreted along side

full representations of the data. Here is the same plot with a histogram that shows more finely the actual data

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



We can see from the histogram that the data are pretty gappy. Most bins have just one measurement in them (see the right hand scale), so the sparse sampling makes our curve density plot of the data less Normal looking. Putting the histogram data on with the confidence interval gives a much clearer view of the data and improves the interpretability of the data.

2.3 Bootstrap Estimation of Confidence Intervals

What can we do about estimating a confidence interval of a parameter like a sample mean when we don't know or can't reasonably assume that the data are drawn from a Normal distribution? We can't apply the formula we saw above as that is tied to assumptions of Normality, but we can use brute force computer power to create a bootstrap confidence intervals.

Bootstrapping is a resampling technique that builds a bigger data set out of a smaller one and hopes to work out what the spread of data would be. Crucially, it doesn't rely on assumptions and can be used on any sort of data.

The logic of bootstrapping goes as follows:

1. We have a set of numbers in a sample that came from a population, so we know these appear in the population
2. If we take lots of samples from our sample, and each time recalculate the parameter we are trying to estimate we can get a distribution of parameter estimates.
3. The distribution of parameter estimates tells us how likely a given parameter estimate is based on the data

In summary it means we can create a confidence interval for our parameter from the data we have.

> **Note**
>
> This 'magic' step is where bootstrapping gets its name from - the phrase "pulling yourself up by your bootstraps" which basically means to create something for yourself from nothing - literally lift yourself up by grabbing hold and pulling on your boots

Let's work through this step-by-step to make it clearer. Starting with the `mouse_wts` data, lets make a random sample of the values of `mouse_wts` of equal length to `mouse_wts` (IE 10) and get the mean of that

```
sample_1 <- sample(mouse_wts, 10, replace = TRUE)

x_bar_sample_1 <- mean(sample_1)
```

Note that we sample 'with replacement' (we effectively put each number back to be selected again if needed) if we didn't we'd end up with just the same set of numbers as in the original `mouse_wts` which is what we don't want - we want a sample of the original numbers. So conceivably we could get the same number from `mouse_wts` all the time. This is actually a desirable feature. Compare `mouse_wts` and `sample_1`

```
mouse_wts
```

```
 [1] 19.57635 29.59349 23.89572 17.75500 11.47902 29.90031 26.04292 30.18206
 [9] 30.28894  8.09662
```

```
sample_1
```

```
 [1] 19.57635 29.59349 26.04292 30.18206 30.28894 11.47902 26.04292 29.90031
 [9] 11.47902 23.89572
```

Some of the same numbers do indeed repeat and some numbers do not appear. The mean of the `sample_1` was 23.848075, this is different to the mean of `mouse_wts` which was 22.6810432

We get a different set of numbers and a different mean each time we iterate this process. If we iterate this many times (like thousands, usually) we get a distribution of the parameter of interest, the mean.

The `bootstrap` function in the `resample` package takes care of this for us. We give it the data we want to sample from, the name of the statistic we want to compute (here `mean`) and the number of replicates to do.

```
library(resample)
bstrap_estimates <- bootstrap(mouse_wts, statistic=mean, R=10000)
```

We can get the values of the CI with the `CI.percentile()` function, passing in the lower and upper bound for the CI with the `probs` argument.

```
CI.percentile(bstrap_estimates, probs=c(0.025, 0.975))
```

```
        2.5%   97.5%
mean 16.6212 27.9609
```

So, this is our 95% CI for the mean of `mouse_wts`. How does it compare with our earlier computed values? It's actually really close.

```
left_bound
```

```
[1] 17.625
```

```
right_bound
```

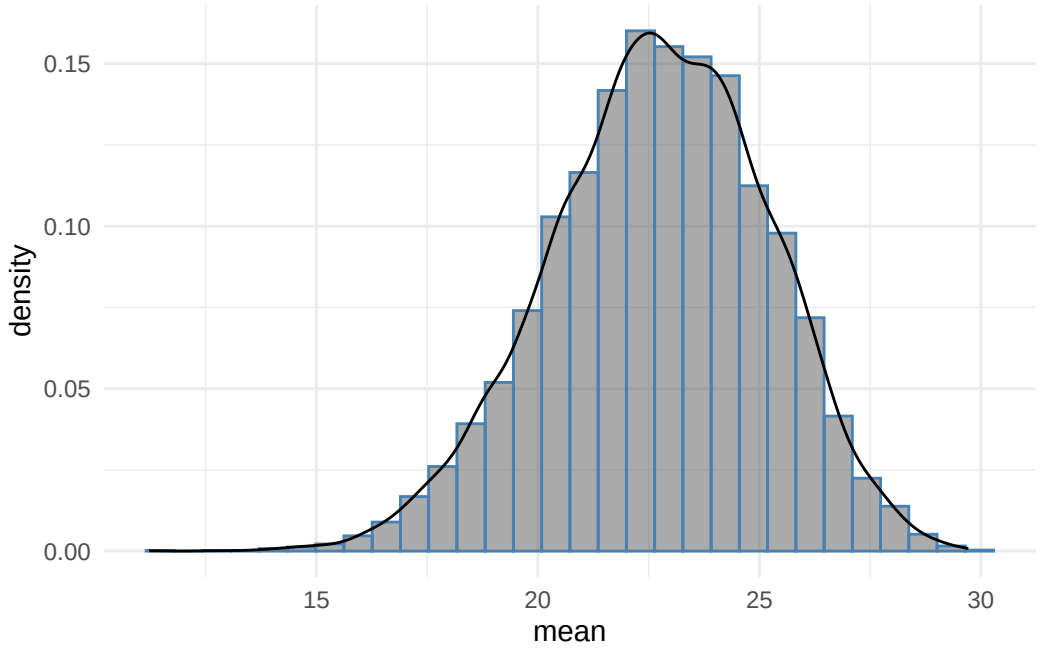
```
[1] 27.73708
```

2.4 Plotting the bootstrap distribution

We would often want to view the bootstrap distribution to understand the CI of our mean.

The function returns a complex object and the actual computed values are in the `replicates` slot which we can get with the `$` syntax and convert to a data frame and use in a typical plot.

```
results <- as.data.frame(bstrap_estimates$replicates)
ggplot(results) +
  aes(mean, y=..density..) +
  geom_histogram(colour="steelblue", alpha=0.5) +
  geom_density() + theme_minimal()
```



The histogram shows that most of the bootstrap resample means were around 22, further the distribution is smooth and approaches the Normal distribution, reflecting the underlying data from which the original `mouse_wts` values were drawn. The sample size of 1000 bootstrap replicates contributes to this and it is clear that the bootstrap CI is reliable as to the position of the mean of the data from which `mouse_wts` was sampled.

Bootstrap sampling can be a great way to get a confidence interval, it is particularly useful when we have smallish sample sizes and/or don't know the background distribution.

Roundup

- Confidence intervals are excellent tools to help us know a range a particular statistic (like a mean) will lie in in a given percentage of samples.
- For normally distributed data we have simple model based methods to compute a confidence interval
- For any data (including normally distributed data) we have the bootstrap method

3 Putting the pieces together

1. Questions

- How do I combine effect sizes and confidence intervals to create an 'ensemble'?

2. Objectives

- Understand how to use multiple estimation statistics in plots for meaningful interpretations
- Be aware of a range of packages that can aid this

3. Keypoints

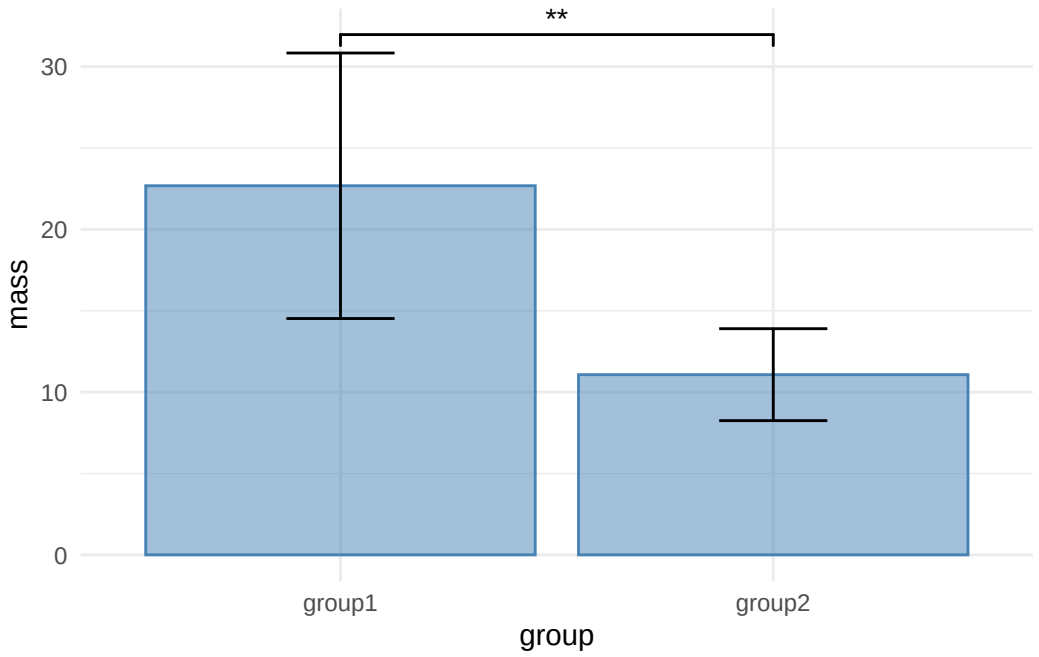
- A good statistical analysis combines p-values, effect sizes and confidence intervals
- A good plot shows the 'ensemble' well

Now that we've studied the two main estimation statistics principles - the effect size and the confidence interval, we should start to look at how to apply them in our work. We should look at bringing them together in an ensemble of statistics to give a clear picture of magnitude of difference (from effect size) and range of the estimate (from a confidence interval). The methods for this are graphical and based around plots.

In this chapter we will look at different plots and plot elements that use our computed estimation statistics clearly and informatively such that we can draw conclusions without hypothesis tests.

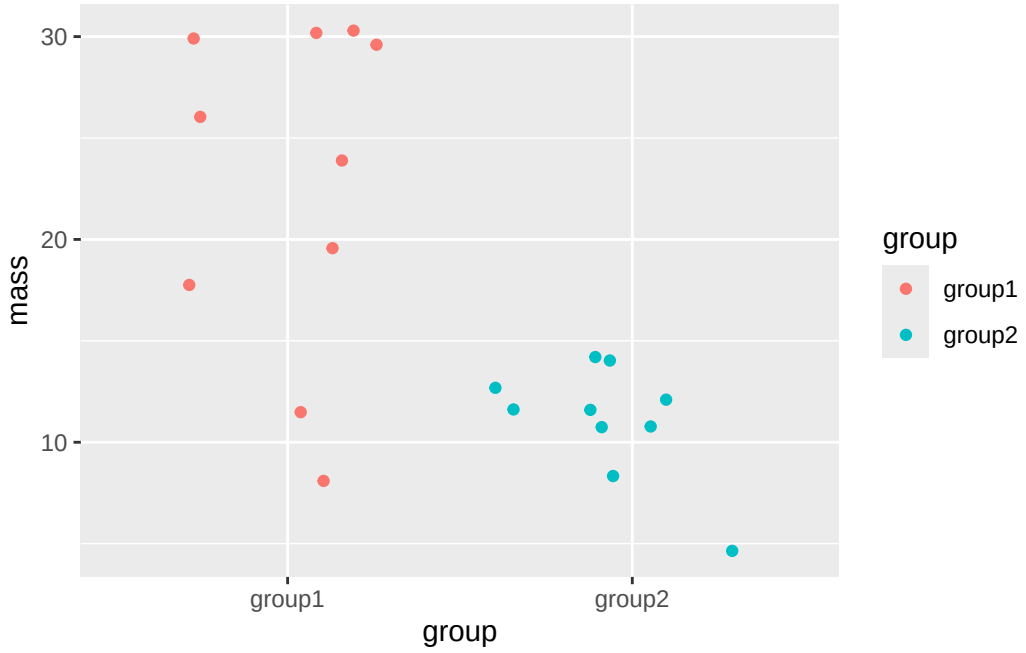
3.1 Incorporating estimation statistics into plots

You are likely familiar with this sort of plot, it is pretty much the most standard plot around, the bar chart with error bars. This one is of our `mouse_wt` data.



It is a very poor plot, for reasons we discussed in the preamble to this course and even in the literature, not least in this article by Weissgerber (2015). Another MAJOR thing about this plot is that it is only appropriate when the y-axis data are continuous. When you have discrete or categorical data on the y-axis, then this plot DOES NOT APPLY AT ALL. You can see a fuller discussion of why here.

Even though we do have continuous data in this example of mouse mass, the bars and error bars serve to hide the spread and patterns in the numbers, hiding the opportunity to understand what is going on in the data as clearly as possible, and if we apply tests blindly, they can lead us to false conclusions. The least we could do to avoid this would be to release the data! Recall how we can show each data point with a simple `ggplot geom_jitter()` call.



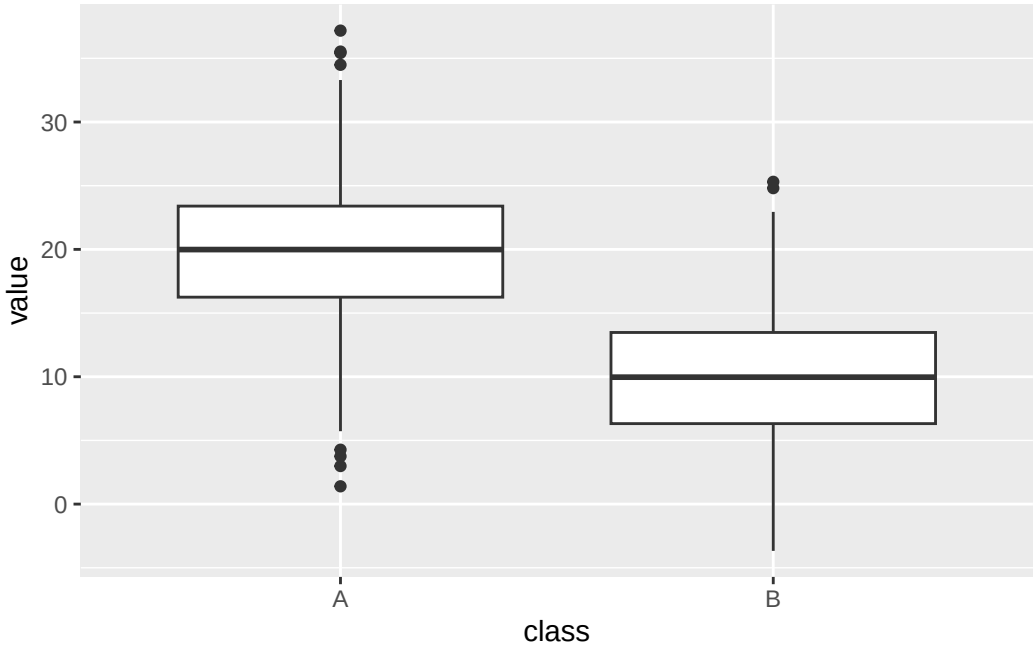
This is much clearer, we can see all the spread of and patterns in the data. This sort of dotted plot should always be a first step in plotting your data.

Let's work towards look at adding a 95% percent CI of the mean on. First, an approach that *doesn't* do this, but that you might be tempted to use as a shortcut.

3.1.1 The Box Plot

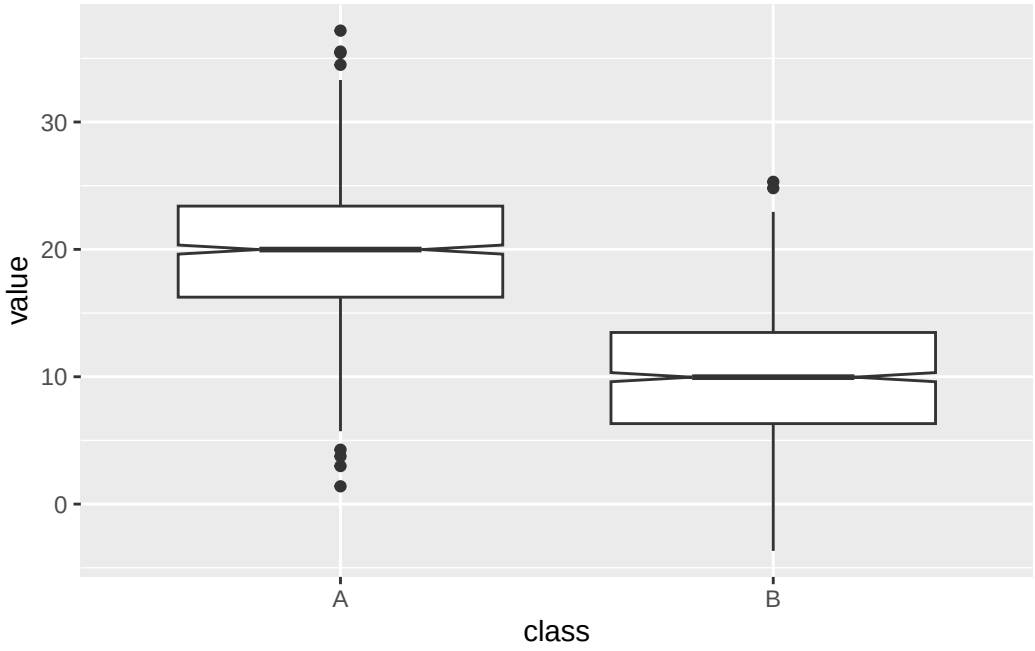
A box plot (or more properly a box and whisker plot) is easy to add to a `ggplot`, and it is a great geom for showing features of the data like mean and spread, but it doesn't do a 95% CI of the mean - not quite.

Here's a general view of one, not our mouse data.



The thick horizontal central line shows the mean, which is useful. The edges of the box show the 25% to 75% IQR (interquartile range), a measure of the spread of the data in which the central 50% is covered by the box. The extent of the vertical lines (whiskers) show the IQR * 1.5 or the most extreme data point (whichever is smallest), the whisker is intended to show the data range. Any points that are beyond the range are drawn in individually and called `outliers` but they have no special statistical significance.

A further feature of the box plot, the notch, looks like this

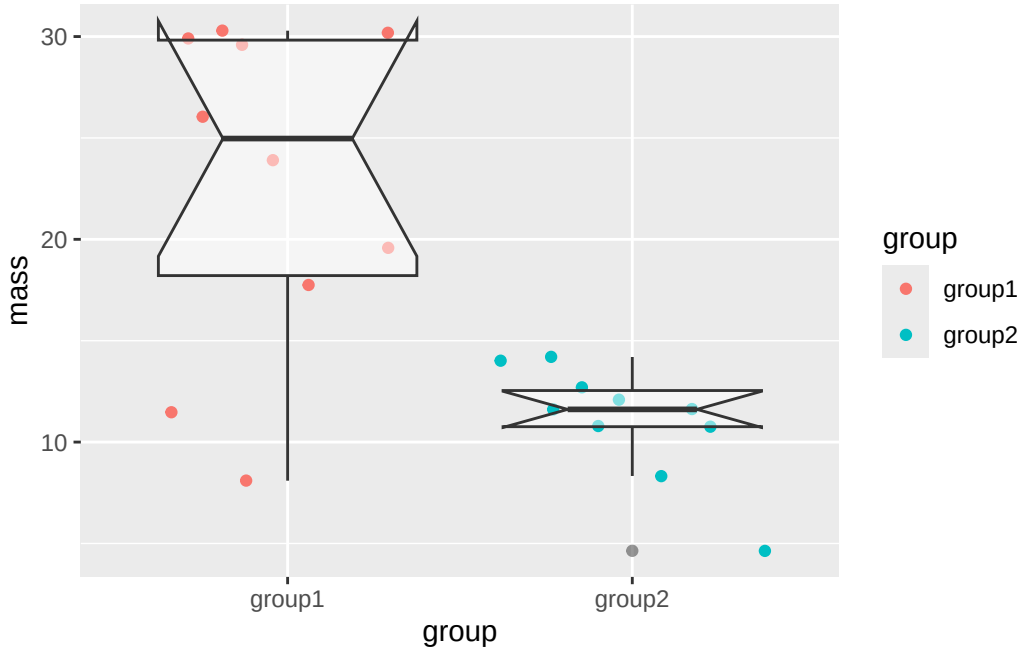


Those little wedges show the extent of `1.58 * IQR / sqrt(n)` which is roughly a 95% CI of the mean in normally distributed data, so it can be useful if you have normally distributed data. In this case (and only in this case) if the notches don't overlap between groups you have a visual clue that there are differences between the groups.

The box plot shouldn't be used on its own though. Here is one with our mouse data, including a notch.

```
base_plot + geom_boxplot(notch = TRUE, alpha=0.5)
```

```
Notch went outside hinges
i Do you want `notch = FALSE`?
Notch went outside hinges
i Do you want `notch = FALSE`?
```



The notch has gone wild in this one! This is because the sample size is so small (10 data points), the range of the notch goes out of the range of the box and the drawing of the lines in the box goes crazy. The box plot is here harder to interpret.

3.1.2 Adding a Normal CI using dplyr

To add a CI we can use the `dplyr` package to calculate the CI limits then add it to a plot. You will recall that it goes like this

1) Use the `group_by` function to group the data into respective groups to work on them one-by-one
2) Use the `summarize` function to get the summary data we need for each group. We'll need the mean mass, SD of mass and number of rows (`n()`).
3) Use the `summarize` function to apply the formula we used for the Normal 95% CI
4) Use the `summarise` function to add the CI upper and lower limits to the mean.

Note that steps 2 to 4 can be done in one pass. The code looks like this

```
ci_df <- mouse_wts %>%
  group_by(group) %>%
  summarize(mean = mean(mass),
            n = n(),
            s = sd(mass),
```

```
            half_width = qnorm(0.975) * s / sqrt(n),
            upper = mean + half_width,
            lower = mean - half_width
  )
ci_df
```

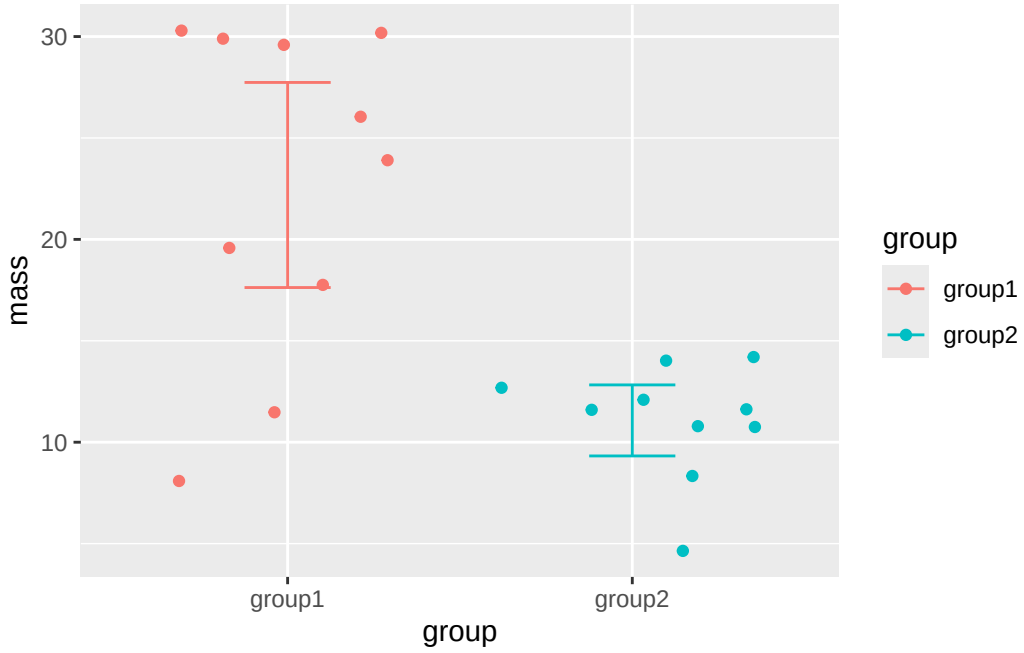
```
# A tibble: 2 x 7
  group   mean     n     s half_width upper lower
  <chr>  <dbl> <int> <dbl>      <dbl> <dbl> <dbl>
1 group1  22.7    10  8.16       5.06  27.7 17.6
2 group2  11.1    10  2.82       1.75  12.8  9.32
```

And the resulting `ci_df` contains the calculated values including the upper and lower values of the CI.

We can apply the summaries to our existing plot using `geom_errorbar()` and the `data` argument, which will use our new data frame as the data source exclusively for this geom.

Because our new data frame doesn't have the original data in it, we have to turn off the original data mapping `aes()` with the `inherit.aes` argument and rename the x axis and colour in a new `aes()`. We should also set the width of the cross lines at the limits of the error bars as they come out comically wide by default.

```
base_plot + geom_errorbar(data = ci_df, inherit.aes = FALSE, aes(x=group, ymax=upper, ymin=lo
```



And there we have a clear view of the 95% CI of the means in our groups.

3.1.3 Non Normal CIs

To add a non-normal CI we need to first calculate the CI using bootstrap resampling. This is a semi-involved process, the `dabestR` package makes it easy, so we'll look at using that. We will use it to make two specific types of plot, the Gardner-Altman (Gardner and Altman 1986) plot and the Cumming plot (Cumming 2012). So far we've concentrated only on continuous y data, so we will also look at making a Derevnina plot (Derevnina et al. 2021), a variant for categoric data, with the `besthr` package.

3.2 The Gardner-Altman Plot

Invented in 1986 by Gardner and Altman, this is a plot for two group continuous data that can show a choice of effect size and the 95% bootstrap CI and distribution in a single panel.

Let's make one with `dabestr`

The first step is to load our `mouse_wts` data into a `dabest` object. We must specify the x and y variable names and using the `idx` argument, the two groups to be compared. The first group will be considered the base for comparison group. We must also tell the function whether these data are paired or not.

```r
library(dabestr)

dbest_data <- load(mouse_wts, group, mass,  idx = c("group1", "group2"))

dbest_data
```

```
DABESTR v2023.9.12
==================

Good morning!
The current time is 10:54 AM on Wednesday December 04, 2024.

Effect size(s) with 95% confidence intervals will be computed for:
1. group2 minus group1

5000 resamples will be used to generate the effect size bootstraps.
```

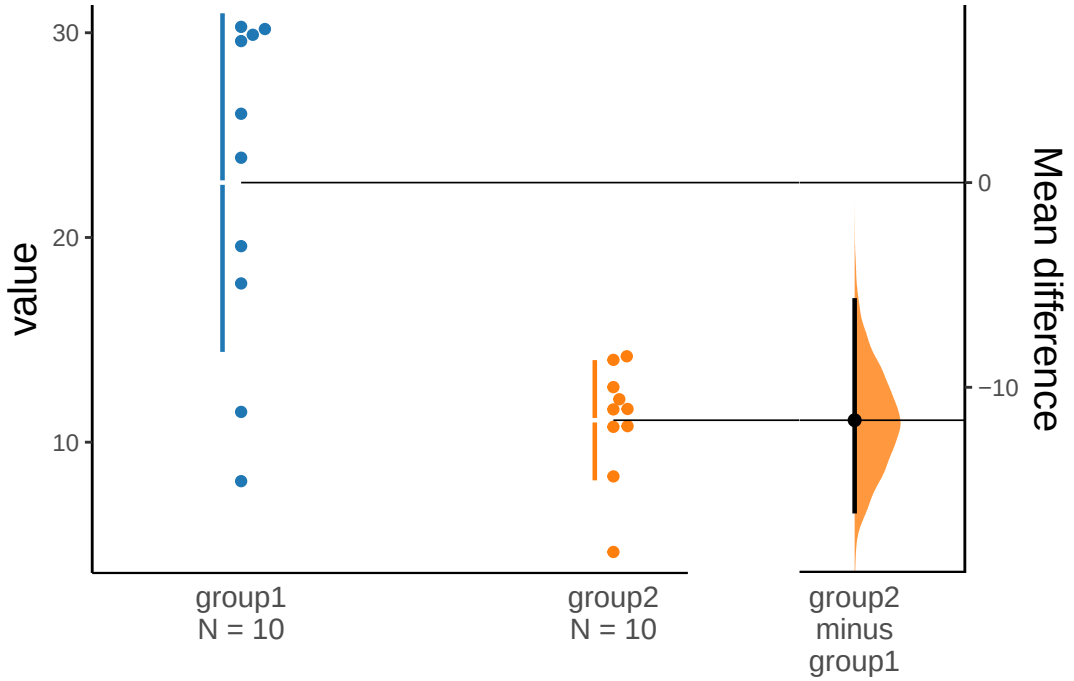
We get a summary that is very friendly and let's us verify whether we entered the data properly.

We can now do an effect size calculation to add to the data object. Here we'll use the mean sample difference, but there are other options, including a `cohens_d()` function.

```r
dbest_with_mean_diff <- mean_diff(dbest_data)
```

We can now draw the plot with the generic `plot()` function.

```r
dabest_plot(dbest_with_mean_diff)
```



This plot is extrememly informative and gives us a super clear indication of our data, the points are shown, the sample size is quoted, the effect size is plotted and the bootstrap CI and distribution are plotted. This is everything we need from a plot and statistical procedure. Importantly, we can conclude that these two samples are not likely to be the same without relying on opaque statistical tests. The plot and the generated numbers also make any argument that the observed differences are significant clearly convincing.

3.3 The Cumming Plot

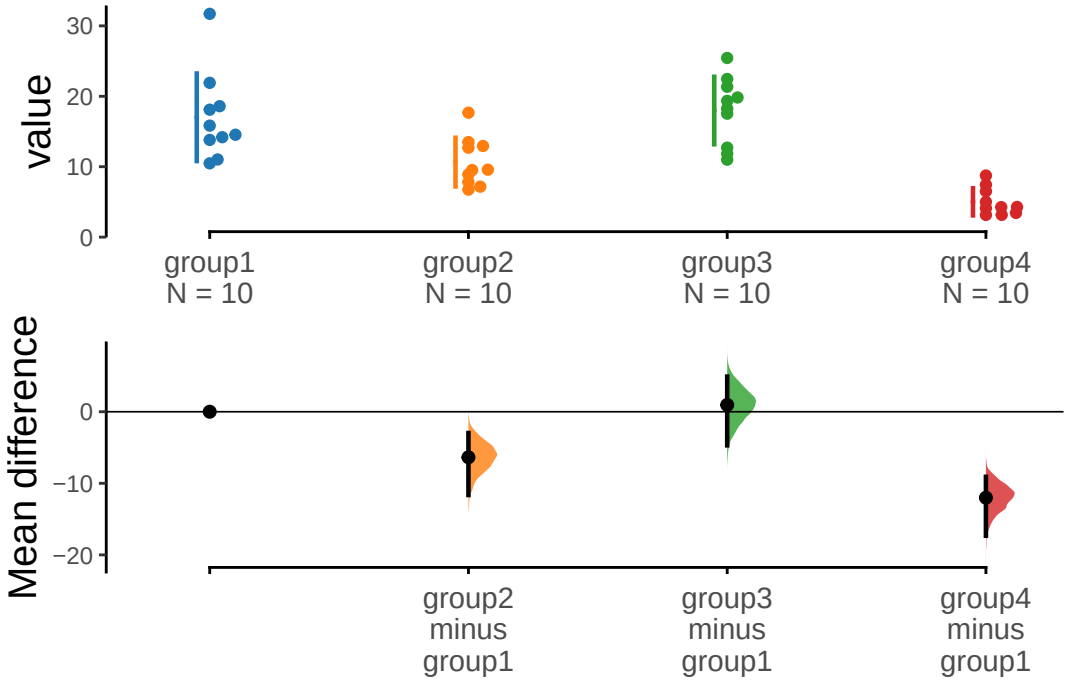
The Cumming plot extends the ideas of the Gardner-Altman plot into more groups and comparisons, named after Cumming who suggested designs like this in his paper (Cumming 2012).

The setup is similar, we will need some data with more groups in it, `mouse_wt_4g`. The othe major difference is that the comparison specification is wrapped in a list of comparisons.

```
library(dabestr)
mouse_wt_4g %>%
  load(group, mass,
         idx = list(
                c('group1', 'group2','group3','group4')
                )
   ) %>%
```

```
  mean_diff() %>%
  dabest_plot()
```

```
Warning in get_plot_component(plot, "guide-box"): Multiple components found;
returning the first one. To return all, use `return_all = TRUE`.
```



The plot shows something similar to the Gardner-Altman in a two-panel layout to allow the bootstrap profiles space. Also the black lines show the extent of the standard deviation, with the mean represented by the gap.

As with the previous plot, this is easy to interpret, rich and informative and convincing. It is clear to see in this plot that the masses of `group1` are substantially and significantly higher than the other groups. We see also that `group2` is not different to `group3` or `group4`, but `group3` and `group4` are different from each other.

3.4 The Derevnina Plot

The Gardner-Altman and Cumming plots work really well on continuous y data. As we have discussed before here, categorical or discrete data need a different representation. An extension of the ideas here for categoric data in general is the Derevnina plot, which was applied specifically to HR experiment score data in her paper (Derevnina et al. 2021).

The basic idea is the same, we have HR score data with e.g strain, replicate and score.

```
hr_scores
```

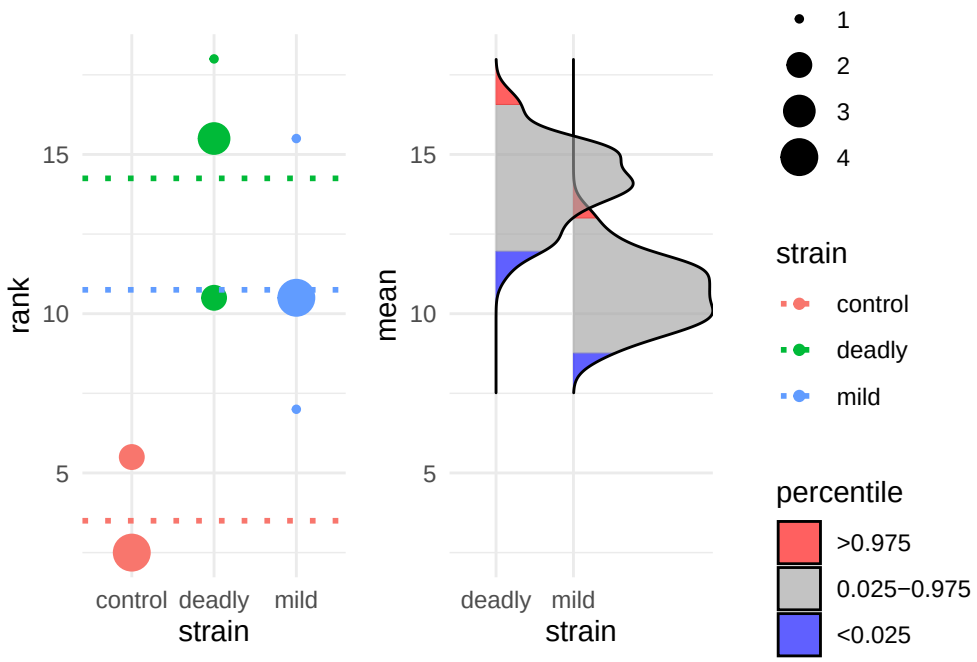
```
    strain replicate score
1  control         1     1
2  control         2     2
3  control         3     1
4  control         4     1
5  control         5     1
6  control         6     2
7     mild         1     3
8     mild         2     4
9     mild         3     4
10    mild         4     4
11    mild         5     5
12    mild         6     4
13  deadly         1     5
14  deadly         2     4
15  deadly         3     6
16  deadly         4     5
17  deadly         5     4
18  deadly         6     5
```

We can use the `besthr` package to create analgous plots for this categoric data. The `estimate()` function requires the data, the y and x variable names and the name of the control in x. It will return an estimation object, which we can plot.

```
library(besthr)

est_1 <- estimate(hr_scores, score, strain, control="control")
plot(est_1)
```

```
Picking joint bandwidth of 0.411
```

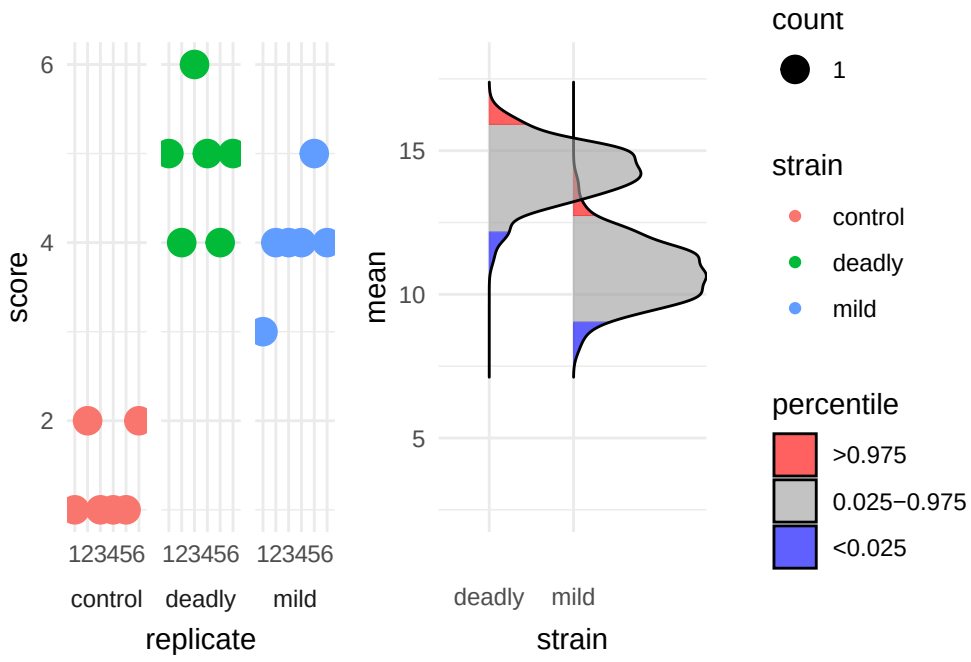


The resulting bootstrap estimations are performed on ranked data, the ranks are computed automatically for us and the resulting rank mean difference shown as dotted lines. The blue areas of the bootstrap distribution show the 95% CI for the mean rank. The plot is again very clear that the `control` has lower mean rank than each of the other two `strains` but the `mild` and `deadly` strains are not much different from each other.

We can see technical replicates if we have used them by extending the call to estimate to include these explictly and we can plot them by setting the `which` argument to `just_data`.

```
est_2 <- estimate(hr_scores, score, strain, replicate, control="control")
plot(est_2, which="just_data")
```

```
Picking joint bandwidth of 0.35
```



The plot is clearer now about the spread of scores in different replicates.

> **Roundup**
>
> Ensemble estimation statistics give us a good view of differences in our data without relying solely (or at all) on reductive p-values.

3.5 References

Setting up

The primary purpose of this course is to help you to understand how to use statistics that will help with your research. The course will try to explain a branch of statistics called 'Estimation Statistics' which are complementary to the normal sort of hypothesis test procedures and address some of the criticisms of those methods.

Statistics is a computationally heavy topic, so we'll be making use of the R statistical programming environment to do that side of the work. The rest of this chapter will help you get that set up on your own computer.

Prerequisites

Knowledge prerequisites

There are no specific knowledge prerequisites for this book but it will be very helpful if you have read and worked through the `ggplot` and `Intro to Stats` books and are familiar with R use.

Software prerequisites

You need to install the following stuff for this book:

1. R
2. RStudio
3. Some R packages: `tidyverse`, `effectsize`, `resample`, `dabestr`,`devtools` and `besthr`

Installing R

Follow this link and install the right version for your operating system https://www.stats.bris.ac.uk/R/

Installing RStudio

Follow this link and install the right version for your operating system https://www.rstudio.com/products/rstudio/download/

Installing R packages in RStudio

Standard packages

In the RStudio console, type

```
install.packages(c("tidyverse", "effectsize", "resample", "dabestr","devtools"))
```

and the packages should install.

Development packages

`besthr` is a new package that needs to be installed from the source.

1. In the `Console` tab in the lower left panel of RStudio type `devtools::install_github("danmaclean/best`

You may get asked to install newer versions of packages, select `1. All` for these questions.

R Fundamentals

About this chapter

1. Questions:

- How do I use R?

2. Objectives:

- Become familiar with R syntax
- Understand the concepts of objects and assignment
- Get exposed to a few functions

3. Keypoints:

- R's capabilities are provided by functions
- R users call functions and get results

Working with R

In this workshop we'll use R in the extremely useful RStudio software. For the most part we'll work interactively, meaning we'll type stuff straight into the R console in RStudio (Usually this is a window on the left or lower left) and get our results there too (usually in the console or in a window on the right).

Panels like the ones below mimic the interaction with R and first show the thing to type into R, and below the calculated result from R.

Let's look at how R works by using it for it's most basic job - as a calculator:

```
3 + 5
```

```
[1] 8
```

```
12 * 2
```

```
[1] 24
```

```
1 / 3
```

```
[1] 0.3333333
```

```
12 * 2
```

```
[1] 24
```

Fairly straightforward, we type in the expression and we get a result. That's how this whole book will work, you type the stuff in, and get answers out. It'll be easiest to learn if you go ahead and copy the examples one by one. Try to resist the urge to use copy and paste. Typing longhand really encourages you to look at what you're entering.

As far as the R output itself goes, it's really straightforward - its just the answer with a `[1]` stuck on the front. This `[1]` tells us how many items through the output we are. Often R will return long lists of numbers and it can be helpful to have this extra information.

Variables

We can save the output of operations for later use by giving it a name using the assignment symbol `<-`. Read this symbol as 'gets', so `x <- 5` reads as 'x gets 5'. These names are called variables, because the value they are associated with can change.

Let's give five a name, `x` then refer to the value 5 by it's name. We can then use the name in place of the value. In the jargon of computing we say we are assigning a value to a variable.

```
x <- 5
x
```

```
[1] 5
```

```
x * 2
```

```
[1] 10
```

```
y <- 3
x * y
```

```
[1] 15
```

This is of course of limited value with just numbers but is of great value when we have large datasets, as the whole thing can be referred to by the variable.

Using objects and functions

At the top level, R is a simple language with two types of thing: functions and objects. As a user you will use functions to do stuff, and get back objects as an answer. Functions are easy to spot, they are a name followed by a pair of brackets. A function like `mean()` is the function for calculating a mean. The options (or arguments) for the function go inside the brackets:

```
sqrt(16)
```

```
[1] 4
```

Often the result from a function will be more complicated than a simple number object, often it will be a vector (simple list), like from the `rnorm()` function that returns lists of random numbers

```
rnorm(100)
```

```
  [1] -0.94415052 -1.54867611  0.86946048 -0.10571191 -0.07618756  0.94701580
  [7]  0.82810622 -1.71854073  2.26039765 -0.01594248  0.14851323 -0.80774582
 [13] -0.15993178 -0.23571200  0.30124368  0.48261941 -0.13288265 -0.84565095
 [19] -0.57710289  1.00489952  0.52209382  0.27720237  0.55754176  0.03109452
 [25]  0.63754790 -0.33710422  2.26948090  1.41226651  0.08697529 -2.12743591
 [31]  1.90746929 -0.67264558 -0.34162669 -1.22001911 -1.99324409  1.07810832
 [37] -1.77040885 -0.87456615 -0.45182293 -0.57311716 -0.56347691  0.50016925
 [43] -0.09272024  0.16817909  0.01795391  0.55550683  2.24138647 -0.35431378
 [49]  1.15705344 -0.72653915  0.26548580 -0.37437904 -0.15448324  0.47707109
 [55]  0.91924144 -1.07736657  0.43078888  0.72073148  0.62198423 -1.18623572
 [61]  1.07974151  0.22322203  0.86720555  1.30730541  1.04105973 -1.89632524
 [67] -0.41419233  1.71325314  0.56991074  0.44007080  0.89504064 -0.02535382
 [73]  0.12192371 -0.17383571  1.35940055  0.62975699  1.71753871 -1.04203290
 [79]  0.21329292 -0.66828034  0.49422168  1.21595290 -2.08979113 -0.46606393
```

```
 [85] -0.01759501 -0.25834180  1.87560141  1.00051587 -0.40189461 -0.27572356
 [91] -0.09890294  0.53074427 -1.70354902 -1.37420371  0.31814228  0.46957972
 [97]  1.58665460  0.15322968  0.13853571  0.62571482
```

We can combine objects, variables and functions to do more complex stuff in R, here's how we get the mean of 100 random numbers.

```
numbers <- rnorm(100)
mean(numbers)
```

```
[1] -0.1195863
```

Here we created a vector object with `rnorm(100)` and assigned it to the variable `numbers`. We than used the `mean()` function, passing it the variable `numbers`. The `mean()` function returned the mean of the hundred random numbers.

Dataframes

One of the more common objects that R uses is a dataframe. The dataframe is a rectangular table-like object that contains data, think of it like a spreadsheet tab. Like the spreadsheet, the dataframe has rows and columns, the columns have names and the different columns can have different types of data in. Here's a little one

```
  names age     score
1 Guido  24 63.969352
2 Marty  45 16.582285
3  Alan  11  2.417062
```

Usually we get a dataframe by loading in data from an external source or as a result from functions, occasionally we'll want to hand make one, which can be done with various functions, `data.frame` being the most common.

```
data.frame(
  names = c("Guido", "Marty", "Alan"),
  age = c(24,45,11),
  score = runif(3) * 100
)
```

Packages

Many of the tools we use in will come in R packages, little nuggets of code that group related functions together. Installing new packages can be done using the `Packages` pane of RStudio or the `install.packages()` function. When we wish to use that code we use the `library()` function

```
library(somepackage)
```

Using R Help

R provides a command, called `?` that will display the documentation for functions. For example `?mean` will display the help for the `mean()` function.

```
?mean
```

As in all programming languages the internal documentation in R is written with some assumption that the reader is familiar with the language. This can be a pain when you are starting out as the help will seem a bit obscure at times. Don't worry about this, usually the `Examples` section will give you a good idea of how to use the function and as your experience grows then the more things will make more sense.

> **Roundup**
>
> - R is an excellent and powerful statistical computing environment

> **For you to do**
>
> Complete the interactive tutorial online https://danmaclean.shinyapps.io/r-start

Cumming, Geoff. 2012. *Understanding the New Statistics: Effect Sizes, Confidence Intervals, and Meta-Analysis.* New York: Routledge.

Derevnina, Lida, Mauricio P. Contreras, Hiroaki Adachi, Jessica Upson, Angel Vergara Cruces, Rongrong Xie, Jan Sklenar, et al. 2021. "Plant Pathogens Convergently Evolved to Counteract Redundant Nodes of an NLR Immune Receptor Network." *bioRxiv*. https://doi.org/10.1101/2021.02.03.429184.

Gardner, M J, and D G Altman. 1986. "Confidence Intervals Rather Than p Values: Estimation Rather Than Hypothesis Testing." *BMJ* 292 (6522): 746–50. https://doi.org/10.1136/bmj.292.6522.746.

Weissgerber, Natasa M. AND Winham, Tracey L. AND Milic. 2015. "Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm." *PLOS Biology* 13 (4): 1–10. https://doi.org/10.1371/journal.pbio.1002128.