Bayesian Inference with Bayes Factors

Dan MacLean

2022-05-01

Table of contents

Motivation

Mr. Micawber's rule of scientific success

We really do need to move away from p-values as a gold-standard of truth in experimental science. The ruinous role of the p-value in modern science can not be overstated. This one value is responsible for happiness and despair in equal measure. They say money is the root of all unhappiness and Dicken's Mr. Micawber had this to say about the role of money in life:

> 'Annual income 20 pounds, annual expenditure 19 [pounds] 19 shillings and six pence, result happiness. Annual income 20 pounds, annual expenditure 20 pounds ought and six, result misery.'

in science a corollary exists:

> 'p below 0.05, result success, papers, grants, and tenure. p above 0.05 result failure, misery, ignominy, and rejection.'

The truth is that $p < 0.05$ is an entirely arbitrary cut-off and is not in itself a helpful or meaningful value. Various scientific communities, led by publishing requirements, have accepted $p < 0.05$ as a gold standard of truth against sense and often against rigorousness. With Bayesian tests we will be able to completely do away with p-values and confidence intervals and in their place use a more evidence based approach to making inferences.

Learning to select hypotheses using Bayesian approaches

The sort of statistics that most experimental science students are taught are called 'Frequentist Statistics'. They include the t-tests, ANOVA and χ^2-tests and the linear models that we have studied already.

The inferential approach (how we make decisions about data) in the Frequentist paradigm is often criticised for being weak and is often abused. Although the abuse is as much a consequence of convention in the scientific literature and in scientific publishing, the misinterpretation of p-values by generations of scientists as it is the philosophical weakness of the methods themselves, the weaknesses persist and over time other paradigms have emerged.

We have seen an alternative in Estimation Statistics, in this course we will look at another - Bayesian Inference. We will use Bayes Factors to compare levels of evidence for one hypothesis over another, rather than just accepting or rejecting a simplistic null hypothesis.

The advantage of this will be that we can much more directly select between specific hypotheses that might describe our data. This will give us a much clearer idea about a question that we instinctively want to answer when we do statistics - 'Which hypothesis is most likely true?', we will see that we can formulate this in lots of ways, but in general the hypotheses we want to compare will be something along the lines of some measured quantity being different in different samples. With Frequentist Inference we can only ask the roundabout question, 'How often does the difference we observe occur by chance?' and if it isn't likely, say so. With Bayes Factors we will be able to compare directly competing hypotheses and reject the least likely absolutely.

Part I

Background

1 Bayesian Inference

1.1 About this chapter

1. Questions
    - What is probability?
    - What does Bayes Theorem do?
    - How can we compare hypotheses about data?
2. Objectives
    - Understand the differences between Frequentist and Bayesian probability
    - Get an appreciation of Bayes Theorem
    - Understand what a Bayes Factor represents
3. Keypoints
    - Probability can be based on frequency of events *or* the level of knowledge we have about a thing
    - Bayes Theorem gives a likelihood based on evidences that can change
    - Bayes Factors are useful in comparing hypothesis about the same evidence

1.2 Frequentist and Bayesian Interpretations of Probability

It may seem like a strange question to ask, but what, exactly, is probability? Whatever it is it certainly isn't a solid thing that we could carry in a bucket. Probability is a strange and often ill-defined concept that can get very confusing when one starts to think deeply about it. When asked what probability is people will generally start to talk about vague concepts like chance or likelihood or randomness or fate, even. Most people will give examples of coins being thrown or dice being rolled. This ephemerality is no good when we want to use probability so when it comes to working with probability statisticians needed to develop very precise definitions. It turns out that different ways of thinking about likelihoods can result in very different definitions of probability.

The two definitions that we will consider are those called the Frequentist and the Bayesian definitions

1.2.1 Frequentist Probability

The Frequentist definition of probability is based on the frequency of occurrence of events. This is a definition that is most similar to the coin toss or dice throw intuition about probability. A probability can be stated thus

$P(Event) = \frac{\text{number of ways event can happen}}{\text{number of all possible outcomes}}$

So in a coin toss, we might get the following probability of getting 'heads'

$P(heads) = \frac{\text{number of heads on the coin}}{\text{number of sides to the coin}}$

which of course, computes as

$P(heads) = \frac{1}{2}$

Thinking of probabilities in this way is similar to a gambler who plays games of chance like roulette or craps, where the odds of winning are entirely based on the outcome of simple random process.

This is so simple and intuitive that we might be tempted to think it's the natural way to think about probabilities, but there are other definitions.

1.2.2 Bayesian Probablity

The Bayesian definition of probability is different, it takes probability to be a reasonable expectation of an event, depending on the knowledge that the observer has. You might understand these probabilities similarly to a gambler that bets on horse races and changes their assessment of a horse's winning ability based on the conditions of the ground and the weight of the jockey. These are trickier to understand than the Frequentist definition but an example can be helpful.

Consider that you and a friend are playing cards and that your friend claims to be able to guess the identity of a card that you draw and replace. A frequentist probability would say that the probability of this was $P(correct) = \frac{1}{52}$. However, you know that your friend is an amateur magician, so you expect that the probability of a correct guess would be much higher than that. That is to say that you have a different reasonable expectation because you have incorporated prior knowledge into your working. Bayesian Probability is based on this prior knowledge and updating of belief based on that knowledge to come up with a posterior likelihood of an event.

In rough terms the answer - a 'posterior probability' is arrived at by combining a 'prior probability' and 'evidence'. In the card guess example the 'prior probability' was the raw chance based probability that anyone would guess the card $\frac{1}{52}$, the 'evidence' was the fact that your friend was an amateur magician and the 'posterior probability' was the updated 'prior probability' that the chance of guessing was higher than $\frac{1}{52}$.

One problem we might spot is how exactly do we update our probability to actually get a measure of the posterior? A formula known as Bayes Theorem lets us do the calculation, but it can be very hard to get the actual numbers we need for evidence and this can be a barrier to using Bayes in the real world. However, let's look work one calculation through with some assumed numbers to get a feel.

1.3 Bayes Theorem by Rough Example

The mathematical basis of calculating a posterior belief or likelihood is done with a formula called Bayes Theorem. Which, using our card example defines the posterior as

$P(correct|magician)$

which reads as the probability of a guess being correct once you know you are working with a magician.

It defines the prior as

$P(correct)$

which reads as the probability of being correct in a random guess (which we know to be $\frac{1}{52}$)

And it defines the evidence as

$P(magician|correct)$

which reads as the probability of the person being a magician given a guess was correct. This is the number which can be hardest to work out in general though in this case we might say it is quite high, say 0.9.

Bayes Theorem then works out the posterior probability given these numbers. There is a very famous formula for this, that I won't include here for simplicity sake, but it is very interesting. We can take a short cut and use R to work out the posterior from the prior and the evidence as follows

```
library(LaplacesDemon)
prior <- c(51/52,1/52)
evidence <- c(0.9, 0.1)

BayesTheorem(prior, evidence)
```

```
[1] 0.997826087 0.002173913
attr(,"class")
[1] "bayestheorem"
```

as it is the first reported number we want, we can see that we get a 99% posterior probability that the guess will be correct if we know that the 90% of correct guesser's are magicians.

The key thing to take away here is that the Bayesian Probability allows us to modify our view based on changes in the evidence. This is a key attribute as we can use it to compare the resulting posteriors from different evidences. In other words it allows us to compare different hypotheses based on different evidence to see which is the more likely.

1.4 Hypotheses in Frequentist and Bayesian Statistics

Now that we know Bayes Statistics allow for updating our beliefs in the light of different evidence we can look at how we can formulate hypotheses to take advantage of this and do something very different with Bayes than we do with Frequentist ideas.

Let' recap the logic of hypothesis tests in Frequentist statistics.

1.4.1 Frequentist Hypotheses

You may recall that the first step of doing a hypothesis test like a t-test is to set up our hypotheses. The first H_0 is the null hypothesis which represents the situation where there is no difference and H_1 is the alternative. Next we select a Null model that represents the Null hypothesis, this step is usually implicit at the operator level and comes as part of the linear model or t-test that we choose to use, and usually is based on the Normal Distribution. Our hypothesis represent the situation as follows

- $H_0 : \bar{x}_1 - \bar{x}_2 = 0$ IE, the sample means are equal.
- $H_1 : \bar{x}_2 - \bar{x}_2 \neq 0$ IE, the sample means are not equal.

We test H_0 (the Null Hypothesis and Model) to see how likely the observed result is under that and if it is unlikely at some level (p) then we reject H_0 and accept H_1.

We criticised this for being weak inference in the Linear Model course. Let's do that again. In this framework haven't we accepted H_1 without analysing it? Here it means that we have had to set up hypotheses that are binary and not compare them directly. We have a take or leave approach to hypotheses.

We haven't, for example been able to ask whether $\bar{x}_1 > \bar{x}_2$ because that wouldn't be askable under our single test, binary paradigm. That's a limitation. As scientists we should be able to collect data and compare models or hypotheses about that data directly.

1.4.2 Bayesian Hypotheses

In the Bayesian Framework we can formulate hypotheses as we wish and compare them directly, using Bayesian probabilities to examine models with different evidences and priors. So if the evidence shows that H_1 isn't any more believable than H_0 we wouldn't falsely fall into the trap of believing H_1 was somehow more correct.

Bayesian Hypotheses can be a bit more like this

- $H_0 : \bar{x}_1 < \bar{x}_2$ IE sample 1 has a lower mean than sample 2
- $H_1 : \bar{x}_1 > \bar{x}_2$ IE sample 1 has a higher mean than sample 2.

which is often much more intellectually satisfying and can lead to clearer answers than the more binary Frequentist hypotheses.

A significant limitation of the approach is the need to select and quantify the prior and the evidence, which can be crucial and lead to very different outcomes if different values are chosen.

Selection of the prior knowledge itself is very difficult and no suitable data may exist. Getting the right data is subjective in many cases and there is no one right way. Domain knowledge is important and often crucial but this can easily lead to bias. An unwitting, uncareful (or say it quietly - unscrupulous) operator could select a prior that would bias the result in favour of a preferred hypothesis. This is a form of confirmation bias or interpretation of the data in a way that confirms your prior beliefs.

For these reasons Frequentist approaches are often the most pragmatic and *a priori* transparent method, though if the priors and evidence can be collected in a non-biased way Bayesian approaches offer us excellent alternatives.

1.5 Bayes Factors

We can use Bayesian Inference through a tool known as Bayes Factors. Bayes Factors are a method of directly comparing the posteriors of different models with different evidences and priors.

Bayes Factors make a ratio of the result of one model or hypothesis over another, resulting in a single quantity that we can examine. Consider that our hypotheses above have been put through the process and a result gained thus

- $H_0 : \bar{x}_1 < \bar{x}_2 \rightsquigarrow Posterior = 0.2$
- $H_1 : \bar{x}_1 > \bar{x}_2 \rightsquigarrow Posterior = 0.6$

We can clearly see that H_1 has 3 times more support than H_0 and we would want to accept that as a better explanation of our data.

Bayes Factors are just that, the ratio of the relative goodness of the hypotheses. From this we can make statements about the support for hypotheses. Wagenmakers et al. (2011) created a table of thresholds indicating interpretations for different Bayes Factors on two hypotheses.

Bayes.Factor	Interpretation
>100	Extreme evidence for H_0 compared to H_1
30..100	Very Strong evidence for H_0 compared to H_1
10..30	Strong evidence for H_0 compared to H_1
3..10	Substantial evidence for H_0 compared to H_1
1..3	Anecdotal evidence for H_0 compared to H_1
1	No evidence
1..1/3	Anecdotal evidence for H_1 compared to H_0
1/3..1/10	Substantial evidence for H_1 compared to H_0
1/10..1/30	Strong evidence for H_1 compared to H_0
1/30..1/100	Very Strong evidence for H_1 compared to H_0
<1/100	Extreme evidence for H_1 compared to H_0

These are extremely useful especially when used with other measures and interpretations like estimation statistics to allow us to make statistical claims.

In the next chapters we will look at how to use Bayes Factors in place of common frequentist hypothesis tests.

> **Note**
>
> The Wagenmakers et al. (2011) article is fun if you can get hold of it. It's a commentary on an earlier article in which the researchers conclude that people have the ability to see into the future! Which they arrive at by misapplying statistics the same way that researchers across all fields do. Wagenmakers *et al* reperform the analysis with Bayes Factors and show that the original conclusions are unsound.

1.6 References

Part II

Tests with Bayes Factors

2 Bayes Factor t-tests

2.1 About this chapter

1. Questions
    - How can I do compare two continuous samples with Bayes Factors?
    - How can I specify a directional hypothesis?
    - How much difference does the prior make?
2. Objectives
    - Understand how a Bayes Factor t-test can be done in R
    - Consider how p and the Bayes Factor are not contradictory
    - Understand that hypothesis and prior selection is important
3. Keypoints
    - The `BayesFactor` package provides functions for Bayes Factor analysis
    - Bayes Factors and p-values ask very different questions
    - One-tailed tests are possible and may be better options

In this section we'll look at how we can do a t-test-like two sample comparison with Bayes Factors. The process is surprisingly straight forward but does need us to pay attention to the weaknesses of the Bayes method - specifically choosing the prior probability distribution. To actually do the tests we'll use the `ttestBF()` in the `BayesFactor` package.

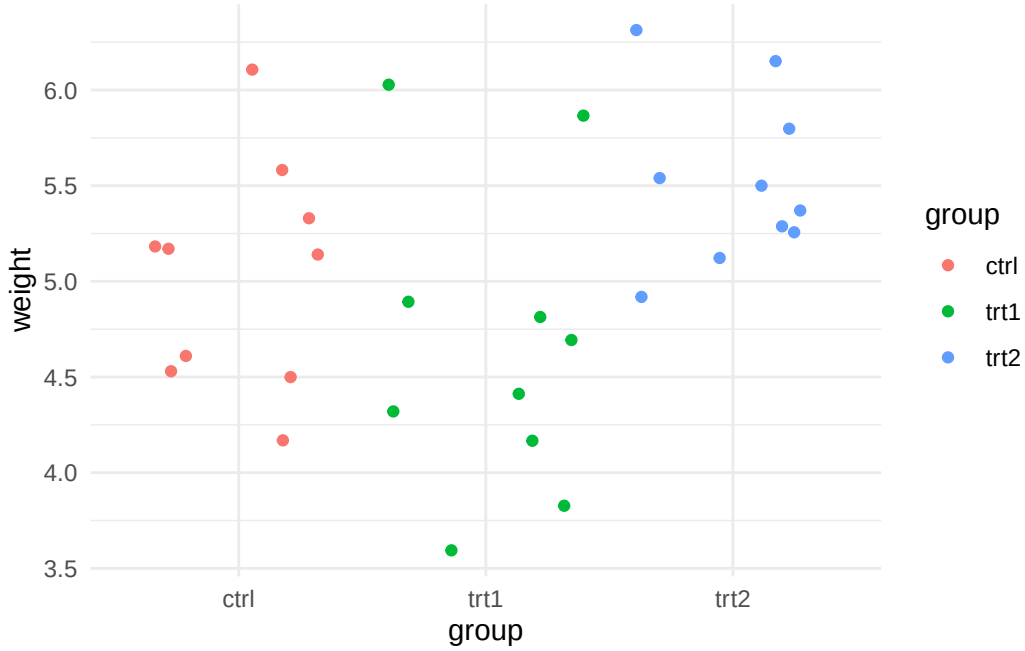
2.2 A Frequentist t-test

To begin we'll first do a normal t-test with a sample data set as a basis for later comparison

2.2.1 The Plant Growth data set

You may recall the Plant Growth data set we used in the Linear Models course, here's a reminder

```
    weight       group
 Min.   :3.590   ctrl:10
 1st Qu.:4.550   trt1:10
 Median :5.155   trt2:10
 Mean   :5.073
 3rd Qu.:5.530
 Max.   :6.310
```



We will use this as an example data set, specifically we'll use `ctrl` and `trt2` data, which we need to extract. Note the mean values for `trt2` look larger than `ctrl`.

```
library(dplyr)
pg_small <- PlantGrowth %>%
  filter(group %in% c("trt2", "ctrl")) %>%
  droplevels()
```

let's calculate too the sample difference mean and the standardised effect size, as it will be important to know these values later

```
library(tidyr)
pg_small %>%
  group_by(group) %>%
  summarise(mean_weight = mean(weight)) %>%
```

```
  pivot_wider( names_from = group, values_from = mean_weight) %>%
  summarise(mean_sample_diff = `trt2` - `ctrl`)
```

```
# A tibble: 1 x 1
  mean_sample_diff
             <dbl>
1            0.494
```

So the mean of `trt2` is bigger than `ctrl` by 0.49 g.

```
library(effectsize)
cohens_d(weight ~ group, data=pg_small)
```

```
Cohen's d |         95% CI
--------------------------
-0.95     | [-1.87, -0.01]

- Estimated using pooled SD.
```

And correspondingly the standardised effect size is large. The effect size is negative because the calculation has been done in the order that the groups appear in the data. `ctrl` comes first so the calculation was `ctrl` - `trt2` which is a negative value. For now, this won't matter. We will need to pay attention to it later.

2.2.2 Two Sample t-test

Let's now do the t-tests. The hypotheses for a test comparing the treatment groups are

- $H_0 : \bar{trt2} - \bar{ctrl} = 0$ IE the mean sample difference is 0
- $H_1 : \bar{trt2} - \bar{ctrl} \neq 0$ IE the mean sample difference is not 0

Using these data to do a t-test is easy, we'll specify a cut-off of 0.05 for rejection of H_0.

```
model <- lm(weight ~ group, data = pg_small)
summary(model)
```

```
Call:
lm(formula = weight ~ group, data = pg_small)

Residuals:
   Min     1Q Median     3Q    Max
-0.862 -0.410 -0.006  0.280  1.078

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.0320     0.1637  30.742   <2e-16 ***
grouptrt2     0.4940     0.2315   2.134   0.0469 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5176 on 18 degrees of freedom
Multiple R-squared:  0.2019,    Adjusted R-squared:  0.1576
F-statistic: 4.554 on 1 and 18 DF,  p-value: 0.04685
```

We get a p-value of 0.046 which is less than our cut-off of 0.05 so we reject H_0 as unlikely and accept H_1 without explicitly testing it. Our conclusion scientifically is that `trt2` has greater `weight` than `ctrl`.

2.3 A Bayesian t-test

Now let's set up a BayesFactor t-test. First we must set our hypotheses. The null hypothesis is similar to that in the frequentist t-test, the idea is that there is no effect which we formulated above as

- $H_0 : \bar{trt}2 - \bar{ctrl} = 0$ IE the mean sample difference is 0

Another way to say this is that the effect size d is 0 so

- $H_0 : d = 0$

Because we need something to compare against we now need to form the alternative hypothesis. By default the `ttestBF()` function tests the alternative hypothesis that the effect size is not 0

- $H_1 : d \neq 0$

and returns the Bayes Factor we need. Performing the test is straightforward

```
library(BayesFactor)
ttestBF(formula =  weight ~ group, data = pg_small)
```

```
Bayes factor analysis
--------------
[1] Alt., r=0.707 : 1.774688 ±0%

Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

We get a clear answer, the output on the line marked `[1]` is a Bayes Factor and states that the data are 1.77 times more likely if H_1 were true than if H_0 were true. In other words the odds of the data favouring the H_1 to H_0 are 1.77:1. Which is the answer we wanted to get, we have explicitly tested H_0 and H_1 and found that H_1 is more likely to fit the data.

2.4 Comparing p and the Bayes Factor for the PlantGrowth data

Comparing to our table of interpretation of Bayes Factors, we see that this corresponds only to 'Anecdotal Evidence' in favour of H_1, which sounds weak and like really there isn't much evidence for the idea that the two samples are different. Do we find this surprising given that the p-value from the t-test was significant? Does this mean that the two methods disagree? Strictly speaking, no, we shouldn't be surprised and no they don't disagree.

It's a bit of an apples and oranges situation. The two values are answers to very different questions.

As we've said before the frequentist p-value only measures the proportion of times a difference of the measured size would occur under some presumed background model. It does not measure the evidence that the hypothesis is true even though that is how many people try to interpret it. p only tells us how often we would be wrong if we reject H_0. As a result, many philosophers have stated that p based significance is a fundamentally uninteresting measure - who cares how often a difference occurs in some ideal world - what is important is the relative fit of the competing hypotheses to the data and that this measure of the strength of evidence per hypothesis is more in line with the interests of researchers.

Taken together our p-value states that the difference between the means of `trt2` and `ctrl` we observed occurs by chance in a normal distribution less than 0.05% of the time and the Bayes Factor tells us that the odds that the data favour the idea that `trt2` is not the same as `ctrl`

are only 1.7 times greater than the idea that `trt2` and `ctrl` are equal. We can see that the two methods do not contradict.

Hopefully this brings home the idea that Bayes Factor is different and arguably closer to what many scientists think they are doing when they do frequentist statistics.

Interpreting these results correctly, then, logically means that a researcher is not likely to be very excited by the results and would not over value the significance of the observed difference.

2.5 Better Hypotheses - One-tailed tests

But looking at the hypotheses we generated, could we ask a better, more informative one? With frequentist tests, no, but with Bayes Factors we can test different hypothesis. Instead of asking whether `trt2` is the same as `ctrl` or not we could ask something more specific. We are likely interested in whether `trt2` is greater than `ctrl`, or in other words that the effect size is greater than 0

- $H_1 : d > 0$

We can specify this H_1 by setting the `nullInterval` argument, this is just the range we expect the effect sizes to be in under the null hypothesis, so we can use 0 to Infinity to cover any increased effect size (and -Infinity to 0 for any decreased effect size.

2.5.1 A data frame based gotcha

Here is where we can run afoul of R's idiosyncarcies - it is important to be careful here because the order of the data in the dataframe can have an effect that can confuse us. Recall that our effect size calculation for these data came out negative because `ctrl` came before `trt2`. Look at the dataframe `pg_small`.

```
str(pg_small)
```

```
'data.frame':   20 obs. of  2 variables:
 $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
 $ group : Factor w/ 2 levels "ctrl","trt2": 1 1 1 1 1 1 1 1 1 1 ...
```

Note that the `ctrl` level in the `group` factor is first, we need to think of our H_1 more carefully,

- $H_1 : d > 0$

really is in this case

- $H_1 : \bar{\text{trt}}2 - \bar{\text{ctrl}} > 0$

so we need to make sure that `trt2` comes first in the `group` factor. We can use the `$` notation to reorder the factor as we wish

```
pg_small$group <- factor(pg_small$group,
                         levels=c("trt2", "ctrl") )
str(pg_small)
```

```
'data.frame':   20 obs. of  2 variables:
 $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
 $ group : Factor w/ 2 levels "trt2","ctrl": 2 2 2 2 2 2 2 2 2 2 ...
```

2.6 Performing the One-tailed test

With that done we can move back on with the one-sided test, specifying the interval as expected.

```
ttestBF(formula =  weight ~ group, data = pg_small, nullInterval=c(0, Inf))
```

```
Bayes factor analysis
--------------
[1] Alt., r=0.707 0<d<Inf    : 3.387166  ±0%
[2] Alt., r=0.707 !(0<d<Inf) : 0.1622109 ±0%

Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

Performing the test was nice and easy and we get an answer. The first line of the output `[1]` states the odds that the data favour the alternative hypothesis over the null are 3.38:1. The Bayes Factor is increased over the earlier more vague hypothesis, suggesting there is actually substantial evidence for the idea that the effect size is greater than 0.

2.7 Testing the effect of the prior

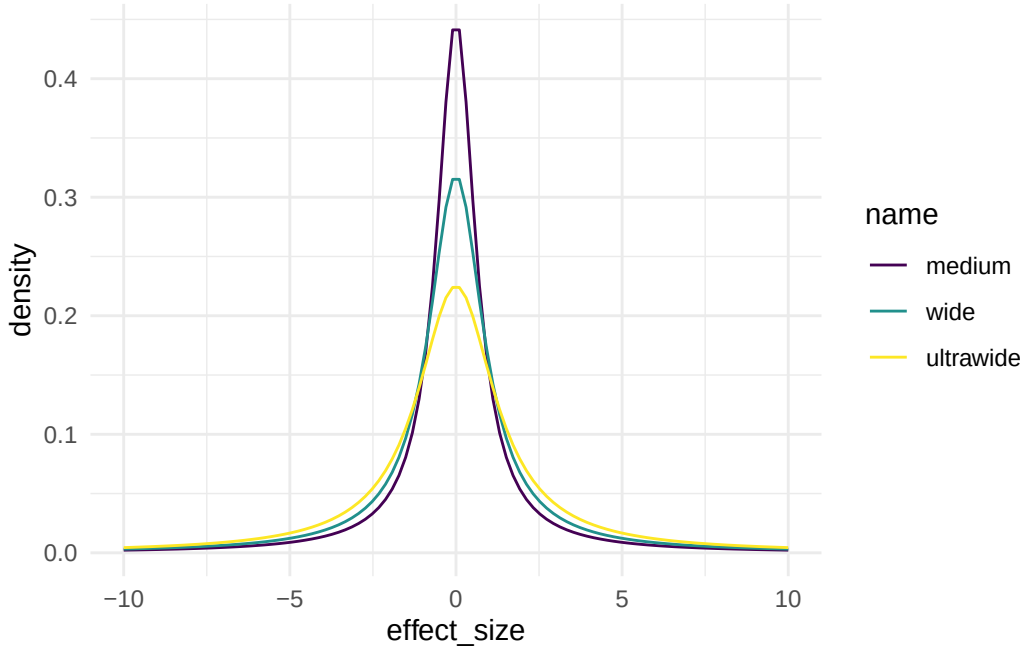
We discussed that one of the limitations of Bayesian Inference was the need to carefully and justifiably select a prior and that doing so was difficult. We'll look at that a little bit now as we did make a decision on this albeit implicitly by allowing the defaults of the `ttestBF()` function.

In our `ttestBF()` function we actually need to provide a prior distribution for the maths to work, not just a single value. We don't want to get into details of those maths as they are out of scope but we do need to know that the prior distribution needs to cover a range of effect sizes that might be plausible if the null hypothesis were false.

The `BayesFactor` package provides a Cauchy distribution as default. Since the selection of the prior implies that we know something about our dat, using the Cauchy implies that we think the population is normally distributed (which is the same distribution we assume under the standard frequentist statistical tests).

2.7.1 The Cauchy Prior Distribution

The Cauchy is a distribution with a single parameter called `scale` that affects how wide its main humpy bit is. In `BayesFactor` there are three widths we can choose from depending on how big a difference we think we are seeing, that is how big the effect size. When plotted, these distributions look like this



and the `name` corresponds to `scale` values as follows

name	scale
medium	0.71
wide	1.00
ultrawide	1.41

In each of the distributions 50% of the area under the curve falls within +/- the scale value.

Since the scale on the x-axis in our plot is effect size, the choice of scale values says somerthing about what we are expecting our effect sizes to be like. The wider the scale value, the bigger we are expecting our effect sizes to be.

Our effect size in the `PlantGrowth` data was 0.95 so well within the area covered by the `medium` scale Cauchy, much more of that curve falls within the -0.95 to +0.95 effect size range than the other two, so we might think that one would be a better fit. That's why it's the default, it's a good fit for generally found effect sizes.

2.7.2 The effect of changing the prior

As an exercise to help us understand the importance of the prior and explicitly NOT a guide to maximising the odds in favour of one model over another. Let's look at how changing the scale via the `rscale` parameter in `ttestBF()` affects the odds of our one sided model.

```
ttestBF(formula =  weight ~ group, data = pg_small, nullInterval=c(0, Inf), rscale="medium")
```

```
Bayes factor analysis
--------------
[1] Alt., r=0.707 0<d<Inf    : 3.387166  ±0%
[2] Alt., r=0.707 !(0<d<Inf) : 0.1622109 ±0%

Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

```
ttestBF(formula =  weight ~ group, data = pg_small, nullInterval=c(0, Inf), rscale="wide")
```

```
Bayes factor analysis
--------------
[1] Alt., r=1 0<d<Inf    : 3.22134   ±0%
[2] Alt., r=1 !(0<d<Inf) : 0.1189759 ±0.02%

Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

```
ttestBF(formula =  weight ~ group, data = pg_small, nullInterval=c(0, Inf), rscale="ultrawid
```

```
Bayes factor analysis
--------------
[1] Alt., r=1.414 0<d<Inf    : 2.857414   ±0%
[2] Alt., r=1.414 !(0<d<Inf) : 0.08596477 ±0.05%

Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

Indeed we do get stronger odds for the alternative hypothesis in the `medium` scale than the others. Note that it isn't wise to go Bayes Factor fishing by post-hoc selecting the prior in order to maximise the Bayes Factor. This example was an exercise to show that prior selection is important.

Roundup

- Bayes Factor *t*-tests allow us to directly compare hypothesis about data in a way that is analogous to *t*-tests.
- We can compare different hypotheses
- The interpretation of a BayesFactor tells us which of the hypotheses are favoured by the data
- Prior selection is important, but `ttestBF()` restricts us to sensible options for data we assume to be normal

3 Bayes Factor ANOVA

3.1 About this chapter

1. Questions
    - How can I do an ANOVA?
2. Objectives
    - Understand multiplicity is not a problem for Bayes Factors
3. Keypoints
    - The package `simplebf` automates Bayes Factor t-tests for many samples

3.2 The issue of multiplicity in Frequentism and Bayesianism

The ANOVA is often seen to be a catch-all test that can be used for an experiment that has more than two samples in it. Experimenters often understand this to be true on the basis that 'you shouldn't do t-tests for more than two samples by repeating the t-test'. This is quite true and is a strategy for avoidance of the problem of multiplicity.

Multiplicity or multiple testing occurs when we do lots of tests one after the other, in a batch. The more we do, the more likely we are to make an error in our conclusions (not in our working). This happens in Frequentist statistical tests because the p-value expresses a fixed error rate that we are happy to accept.

Recall that the t-test has two hypotheses (of which we test just one)

$H_0 : \bar{x_1} - \bar{x_2} = 0$

$H_1 : \bar{x_1} - \bar{x_2} \neq 0$

and we set a level at which would reject H_0 usually $p < 0.05$. The p reflects the proportion of times that the difference observed is seen in the null model by chance (so we see the difference 1 in 20 times by chance), in other words in a proportion of 0.95 of times we would reject the null correctly. Which is fine for just one comparison.

If we do more than one test we must multiply these probabilities together, giving $0.95 * 0.95 = 0.9025$. This is catastrophic, by doing just two tests we reduce the proportion of times we choose the correct hypothesis to 0.9025, down from 19/20 to 18/20, we make twice as many mistakes! For more tests this gets worse.

Frequentist statistics have lots of corrections for this sort of problem and the ANOVA post-hoc tests are in part a way of doing that. The good news for those using Bayes Factors is that this problem does not exist. Because we don't have a fixed error rate, it doesn't get bigger when we do more tests. We are free to do as many hypothesis comparisons as we wish.

3.3 Automating `BayesFactor::ttest()` for many comparisions

As there isn't a need for a Bayes Factor analogue to the ANOVA and post-hoc tests, we can just use the t-test analogue over and over again. If we have a multiple sample dataset we just need a book-keeping method to pull out the samples of interest.

Let's draft one with `dplyr` and the Plant Growth data set.

```
library(dplyr)
library(BayesFactor)

small_df <- PlantGrowth %>%
    filter(group %in% c("ctrl", "trt1")) %>%
    droplevels()

ttestBF(formula = weight ~ group, data = small_df)
```

This pattern helps you extract the pairs of samples you need, though you would need to repeat it every time you wanted to analyse a new pair. A convenience function for the simple case that allows us to do `BayesFactor::ttestBF()` for all pairs in a specified column in a dataframe exists in the package `simplebf`. It works like this:

```
library(simplebf)
result <- allpairs_ttestbf(PlantGrowth,
                           group_col = "group", data_col = "weight",
                           rscale = "medium",
                           h_1 = "test_greater_than_control")

knitr::kable(result, digits = 4)
```

control_group	test_group	h_0	h_1	BayesFactor	odds_h_1	summary
trt1	ctrl	ctrl equal to trt1	ctrl greater than trt1	1.0834	1:1.0834	Anecdotal evidence for H_1 compared to H_0
trt2	ctrl	ctrl equal to trt2	ctrl greater than trt2	0.1622	1:0.1622	Substantial evidence for H_0 compared to H_1
ctrl	trt1	trt1 equal to ctrl	trt1 greater than ctrl	0.2167	1:0.2167	Substantial evidence for H_0 compared to H_1
trt2	trt1	trt1 equal to trt2	trt1 greater than trt2	0.1363	1:0.1363	Substantial evidence for H_0 compared to H_1
ctrl	trt2	trt2 equal to ctrl	trt2 greater than ctrl	3.3872	1:3.3872	Substantial evidence for H_1 compared to H_0
trt1	trt2	trt2 equal to trt1	trt2 greater than trt1	12.6445	1:12.6445	Strong evidence for H_1 compared to H_0

The results are pretty easy to read. Note we can set `rscale` values as in the `ttestBF()` and we can choose one of three values for H_1 `test_greater_than_control`, `test_less_than_control` and `test_not_equal_to_control`.

Roundup

- Bayes Factors do not need multiple hypothesis corrections
- `simplebf` is a package for automating the comparison of all groups in a single variable in a tidy dataframe

4 Bayes Factor on Contingency Tables and Proportions

4.1 About this chapter

1. Questions
    - How can I do a categoric count based χ^2 or proportional test with Bayes Factors?
2. Objectives
    - Perform χ^2 on contingency tables of any size
3. Keypoints
    - `BayesFactor` and `simplebf` provide functions and automations for categorical count or frequency data
    - These are useful for HR scoring data

The `BayesFactor` package has some functions for performing other types of tests and returning a Bayes Factor. In this section we will briefly look at these.

4.2 Bayes Factor χ^2

A common question is whether proportions of counted things or frequency is different between samples. The one we typically learn first as biologists is Mendel's pea data that led to his genetic insights, like this 2x2 table for flower colour (purple or white). Note that we have the counts of flower colour that were observed and expected counts that would come from a 3:1 Mendelian segregating cross.

```
mendel_data
```

```
           P   W
observed 459 141
expected 450 150
```

The χ^2 test is the classical frequentist test performed to determine differences in proportions in a contingency table, and there is an equivalent Bayesian method in `BayesFactor`. We can run our data through the function `contingencyTableBF()` very easily, but it does need the data to be an R matrix object, not the more typical dataframe. We can change that easily with `as.matrix()`, then run the function.

The arguments are important: `fixedMargin` describes whether the variable of interest is in the rows or columns of the table - here it is in the columns so we use `cols`; `sampleType` describes what the function should do in the Bayesian sampling process as it runs. This is highly technical and out of scope for what we want to discuss, so I'm going to gloss over it. The function documentation has more information if you want it (`?contingencyTableBF`) the option used here `indepMulti` is a good one to start with.

```
mendel_matrix <- as.matrix(mendel_data)

library(BayesFactor)
contingencyTableBF(mendel_matrix, sampleType = "indepMulti", fixedMargin='cols')
```

```
Bayes factor analysis
--------------
[1] Non-indep. (a=1) : 0.1011097 ±0%

Against denominator:
  Null, independence, a = 1
---
Bayes factor type: BFcontingencyTable, independent multinomial
```

The hypotheses that are tested in this example are fixed and simple ones. Strictly H_0 is that the proportions in the table are equal and H_1 is that the proportions are not equal. So in effect the whole table is tested to see whether the observed counts are different to the expected counts. Here we see that the odds are 1:0.101 *against* H_1 so the conclusion is that the proportions are equal, that is our observed flower colour proportions match the expected.

There isn't a way to use different H_1's in the way that we did with the Bayes Factor t-test, so we can't test the explicit hypothesis that one is bigger (or smaller than the other).

4.2.1 Converting a dataframe to a contingency table

In most of our work we've used tidy data (or case based data) in dataframes. The function we just learned uses a contingency table in a matrix, not a dataframe. Sometimes too, we will want to make a contingency table to see it. We can make a contingency table out of a dataframe with the `table` function, we just have to select the columns we want using the `$` notation.

```
hr_df
```

```
# A tibble: 9 x 3
  strain  replicate score
  <chr>       <dbl> <dbl>
1 control         1     1
2 mild            1     3
3 deadly          1     4
4 control         2     2
5 mild            2     3
6 deadly          2     4
7 control         3     1
8 mild            3     3
9 deadly          3     3
```

```
hr_cont_table <- table(hr_df$score,hr_df$strain)
```

4.2.2 Bigger contingency tables

Sometime we'll have a contingency table of counts that is larger than 2 x 2 IE we have more than two samples and more than two levels of a variable. For example we might have this HR scoring table.

```
hr_table
```

```
  control deadly mild
1       2      0    0
2       1      0    0
3       0      1    3
4       0      2    0
```

As we can see it shows an HR score in the rows and different strains in the columns. The numbers represent the count of times each score was seen in three replicated experiments. Because it's a contingency table the replicates are merged in together. It is important therefore that the same amount of sampling was done in each strain.

Here we would want to compare the two basic hypotheses of whether the proportions of observed scores are different between the strains are the same or not. Let's go ahead and do that with `contingencyTableBF()`

```
contingencyTableBF(hr_table, sampleType = "indepMulti", fixedMargin = "cols")
```

```
Bayes factor analysis
--------------
[1] Non-indep. (a=1) : 11.55 ±0%

Against denominator:
  Null, independence, a = 1
---
Bayes factor type: BFcontingencyTable, independent multinomial
```

We get a clear answer, the Bayes Factor strongly favours the hypothesis that the proportions of scores across strains are not equal. Which is nice but it doesn't go far enough - it doesn't tell us which are bigger than others and whether the conclusion applies to all the possible pairings of strains. This is the same problem we had with the Bayes Factor *t*-test and the solution is the same. We can just pull out each pair of strains and compare them one pair at a time. All we need is a book-keeping method to do this. The library `simplebf` contains one, so let's use that.

We can use the `allpairs_proportionbf()` function to get a data frame of Bayes Factors. If you pass this function a dataframe it will make the contingency table for you. You must specify which columns to use for the group and the counts. For easy reading we'll send the output to the `knitr::kable()` function.

```
library(simplebf)
allpairs_proportionbf(hr_df,
                      group_col = "strain", count_col = "score",
                      sample_type = "indepMulti") %>%
  knitr::kable()
```

control_group	test_group	h_0	h_1	BayesFactor	odds	bf_summary
control	mild	mild proportions equal to control proportions	mild proportions not equal to control proportions	5.6000	1:5.6	Substantial evidence for H_1 compared to H_0
control	deadly	deadly proportions equal to control proportions	deadly proportions not equal to control proportions	4.2000	1:4.2	Substantial evidence for H_1 compared to H_0
mild	deadly	deadly proportions equal to mild proportions	deadly proportions not equal to mild proportions	2.1875	1:2.1875	Anecdotal evidence for H_1 compared to H_0

So we get a nice set of Bayesian Hypothesis test for proportion or contingency table data on our HR experiment.

> **Roundup**
>
> - Bayes Factors can be used for proportion tests like the χ^2
> - The `BayesFactor` and `simplebf` packages are useful tools implementing these

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`, `Intro to Stats` and `Estimation Statistics` 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: `devtools`, `tidyverse` and `BayesFactor` and `simplebf`

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", "devtools", "BayesFactor"))
```

and these packages should install. Once that is done, type

```
devtools::install_github("danmaclean/simplebf")
```

to install the final package

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]  0.016462236 -0.067227114  1.275945141  0.074917119 -1.888627555
  [6] -1.393397693  1.084217560 -1.571466365  0.136622191 -0.115198118
 [11] -0.187390639  0.463747416 -0.181248005 -0.379680700  0.555015628
 [16] -0.927346830 -0.742599448  0.597151303 -2.123336682  0.349818432
 [21] -0.573431209 -0.332480783 -1.364223819  0.361332577  0.358422704
 [26] -0.003078206 -1.035649220 -1.643442351  0.355391303 -1.809201261
 [31]  1.715503036 -0.937862657 -0.158335281  1.117304153 -0.335464860
 [36]  1.836347205 -0.805181369  1.002967854  0.401403287 -0.556871423
 [41]  0.429921580  0.272820808  0.651281302 -0.943411733  1.552192717
 [46]  0.640611833  1.687426390  0.018507821  0.244251753 -1.148090969
 [51] -0.804283573  1.782516444 -0.826521766 -0.020082032  1.253940965
 [56]  0.083781682  1.137014675  0.704295572  0.668589925  1.735597188
 [61] -0.171815761 -0.551948376  1.523139435 -0.062948817  1.223807298
 [66] -0.295273087  0.688572498 -0.588642576 -1.283534533 -0.318113641
```

```
 [71] -1.350438149 -0.204627078  0.430765033 -0.718295834 -1.186648853
 [76] -1.019526679 -0.162495144  1.763888517 -0.030703047 -1.205553151
 [81]  0.609815807 -1.849825268  0.538986356 -0.559927751  0.528043359
 [86] -0.680607902  1.049143122  0.693643153  0.306646235 -0.766900649
 [91]  0.426951421  0.416344956  1.079778487  0.351290386 -1.401048465
 [96] -0.683954669  0.156395424 -0.452664433 -0.711574704 -1.004393762
```

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] -0.1171882
```

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 68.09373
2 Marty  45 47.84005
3  Alan  11 60.84998
```

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

Wagenmakers, E. J., T. Wetzels, D. Borsboom, and H. L. J van der Maas. 2011. "Why Psychologists Must Change the Way They Analyze Their Data: The Case of Psi: Comment on Bem (2011)." *Journal of Personality and Social Psychology* 100: 426–32. https://doi.org/https://doi.org/10.1037/a0022790.