Notes on and Solutions for Statistical Rethinking

Alexander Pastukhov

2020-12-13

Contents

1 Precis **5**

2 Loss functions **7**

2.1 Loss function, the concept 7

2.2 L0 (mode) 8

2.3 L1 (median) 9

2.4 L2 (mean) 12

2.5 L1 (median) vs. L2 (mean) 13

2.6 Choosing a likelihood 14

2.7 Gaussian in frenquentist versus Bayesian statistics 16

3 Multiple Regression and Causal Reasoning **17**

3.1 Peering into a black box 17

3.2 Turning unconditional dependence into conditional independence 19

3.3 Magic of multiple regression 23

3.4 Multicollinearity 24

3.5 Back to spurious association 26

3.6 Warning 29

4 Solutions for Chapter 2 **31**

5 Solutions for Chapter 3 **43**

6 Solutions for Chapter 4 **67**

Chapter 1

Precis

This is a collection of solutions for exercises but also of notes that attempt to provide further details and intuition for some topics, such as information theory, information criteria, MCMC algorithms, etc.

Chapter 2

Loss functions

The purpose of this comment is to give you an intuition about loss functions, mentioned in chapter 3. In particular, I want you to understand why different loss functions (L0, L1, and L2) correspond to different point-estimates (mode, median, and mean). Plus, I want you to understand that you can view a choice of a likelihood function, as in picking Gaussian in chapter 4, as being analogous to picking a loss function.

I am afraid that the easiest way to explain why an *L2* loss results in *mean* is via a derivative. So, if you are not confident in your basic calculus skill, it might be useful for you to first watch a few episodes of Essense of Calculus series by Grant Sanderson, a.k.a. 3Blue1Brown. I would suggest watching at least the first three episodes (actually, I would recommend to watch the whole series) but if you are short on time watch only episode 2[1].

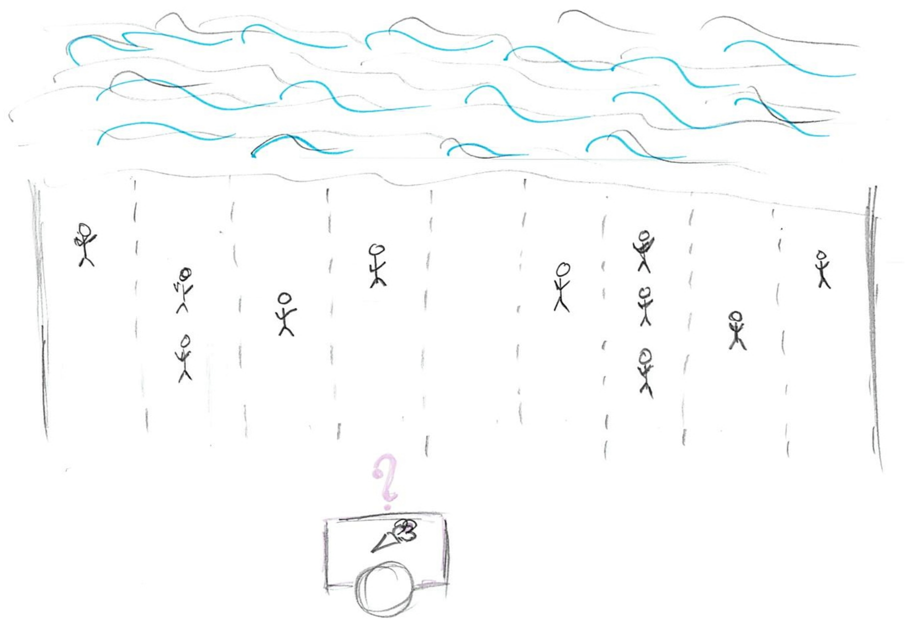
2.1 Loss function, the concept

Imagine that you are selling ice-cream on a beach, so we can assume it is a narrow strip of sand and, therefore, a one-dimensional problem. It is hot, so *everyone* wants ice-cream (obviously) and you want to maximize the number of ice-creams you sell (obviously). People are distributed in some random (not necessarily uniform or symmetric) way along the beach, so the question is: Where do you put your *single* ice-cream stand to maximize your profits? The answer depends on your choice of the *loss function* that describes how distance between a particular person and your stand influences whether person will buy your ice-cream. In other words, it describes the *cost* of getting to your stand, i.e. walking all-the-way through the sand in that heat. This *cost* clearly depends on the

[1]Although, if you skip episode 1, you won't know why it is *obvious* that area of a circle is $\pi \cdot r^2$

distance and in the simplest case, it is linearly proportional to the distance: If you need to walk twice the distance, your costs for getting an ice-cream are twice as high. However, the relationship between the distance and cost does not have to be so simple and linear and this is why we have many different *loss* / *cost* functions.

We can write a loss/cost function more formally as $L(stand, person_i)$ where `stand` is the location of your stand and `person_i` is a location of a particular ith person. The cost can be either zero or positive, i.e., we assume there is no benefit in walking all the way, only no or some cost. So, where should you put your ice-cream stand?

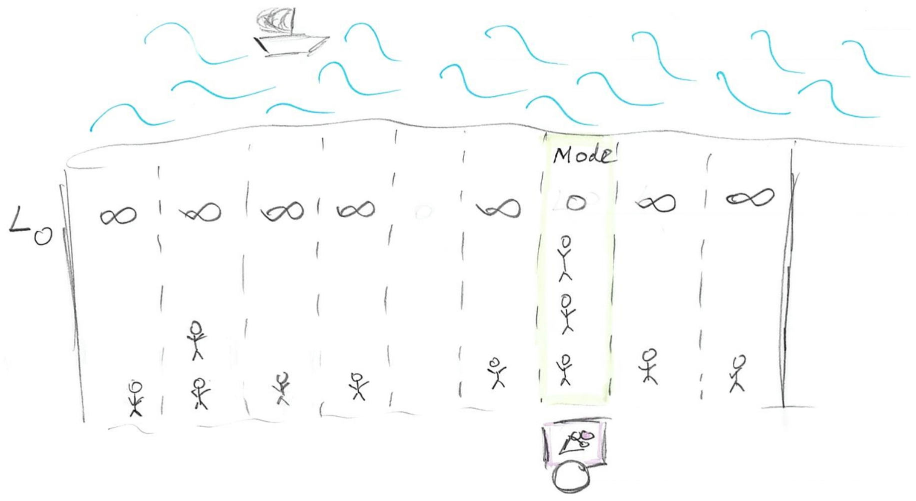


2.2 L0 (mode)

The simplest loss function is

$$L0(stand, person_i) = \begin{cases} 0, stand == person_i \\ \infty, stand \neq person_i \end{cases}$$

This function assumes that everybody hates walking so much, that *any* walk is unbearable and should be avoided. Thus, there is no cost for getting your ice-cream only for people who are positioned right next to your stand. For everybody else, even one meter away, the costs of walking are infinite, so they won't bother and, therefore, won't buy your ice-cream. Still, we are in the business of selling one, so where do we put our stand given how lazy our customers are? Well, we just find the biggest group of people and put our stand next to them. No one else will come but at least you got the biggest group of customers you could. If you look at the *distribution* of your customers along the beach this is the highest peak (that you peak) and it is called the *mode* of the distribution.

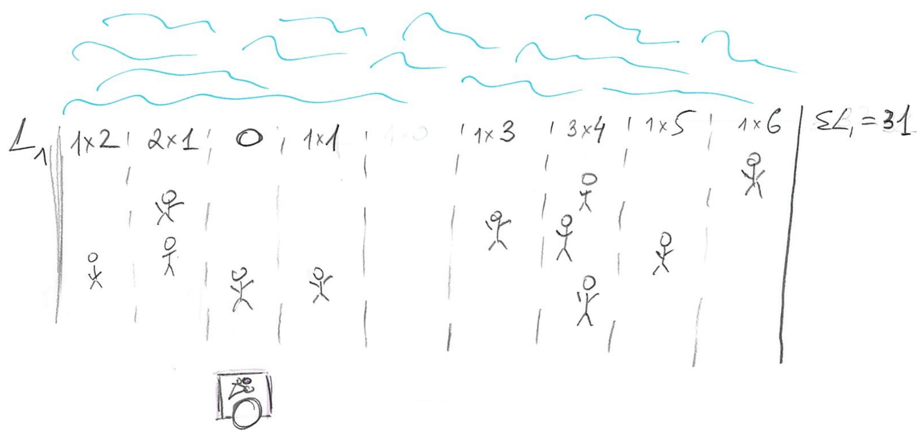


2.3 L1 (median)

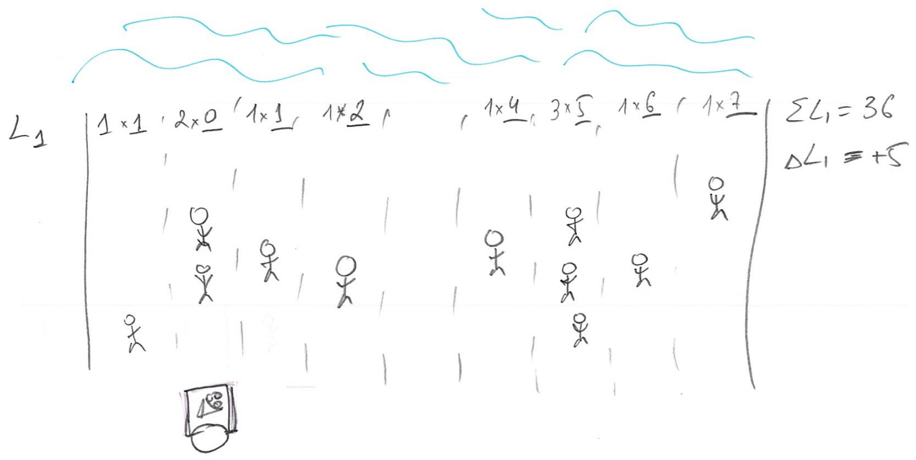
The next loss function, that I already mentioned, assumes a simple linear relationship between the distance and the cost

$$L1(stand, person_i) = |person_i - stand|$$

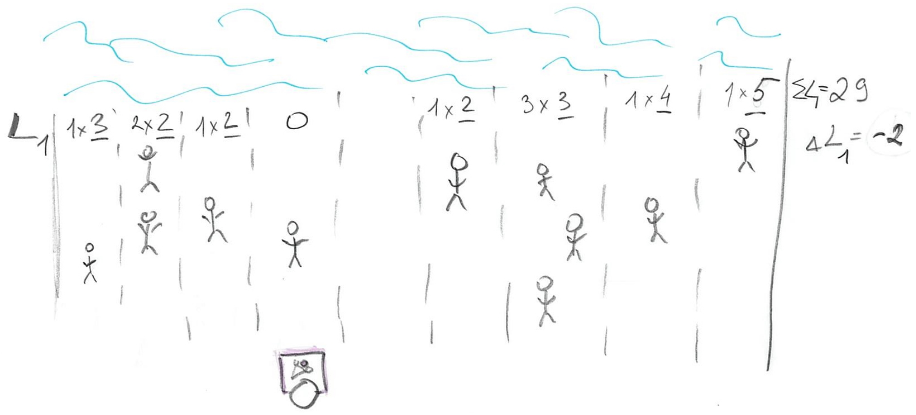
In other words, the cost is equal to distance (we need `| |` to get an absolute value, because the person could be "to the left of" of stand, in which case `person - stand` distance will be negative). So, where should we put our stand? Let us start at a fairly random location so that 3 of our customers are on the left and 7 are on the right.



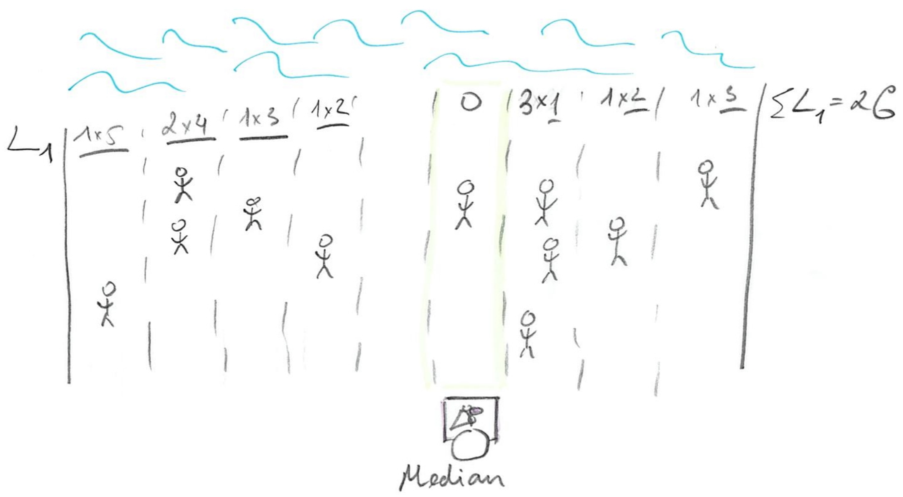
We can, in principle, compute the actual cost but it is simpler to ask the question of whether we can *improve* on that cost by moving somewhere else? Imagine that we move to the left where *minority* of our customers are. Now we have 1 on the left and 8 on the right (plus 2 more at our location).



The problem is, we moved *away* from the majority of the people so our total cost is *original cost - 3 (improvement due to moving close to minority) + 8 (increase in loss due to moving away from majority)*, so $\Delta L1 = +5$. Oops, we made it worse! How about moving to the *right*?



Now that we move *towards* the majority of customers, we have four on the left and six on the right (plus one at our location). The change in cost is *original cost + 4 (loss due to moving away from minority) - 6 (improvement due to moving towards majority)*, so $\Delta L1 = -2$. Which gives us an idea: we should try to get even closer to that majority by keeping walking to the right! Eventually, you will get to point of the 50/50. Should you keep moving to the right? Should you move to the left? Should you move at all?



There is no point in moving to the left. You just came from where because moving to the right made things better. However, if you keep moving to the right, you will keep passing people, so that majority now will be on the left and you would be walking *away* from the majority, raising the costs (and your losses). So, once you get to point where half of your customers are on the left and half are on the right, you cannot do any better. Any movement that gets you from 50/50 means there are more customers on one side (say left, if you moved to the right) and, as we already figured out, your best strategy is to move towards the majority, which gets you back where you started at 50/50 point. That 50/50 points split, when half of customers / probability mass is on one side and half is on the other, is called *median*.

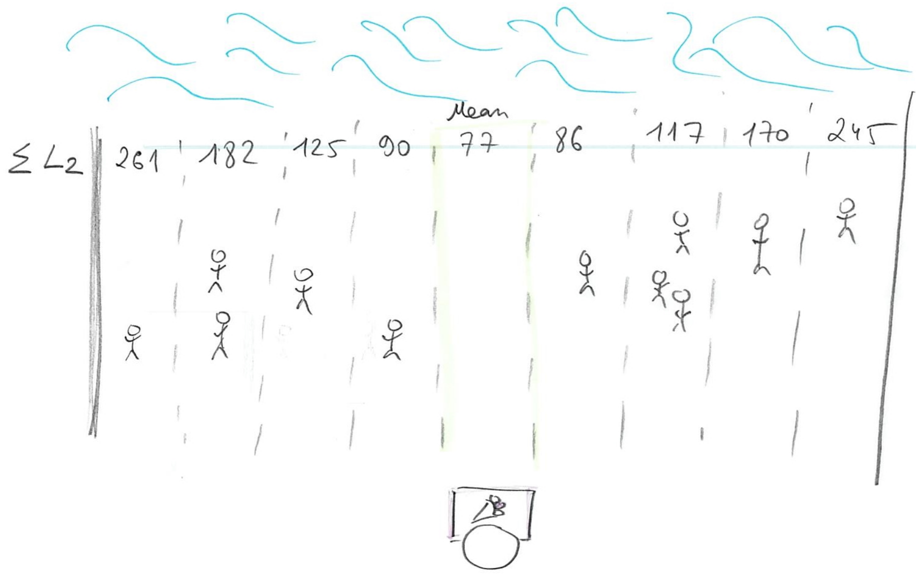
2.4 L2 (mean)

The classic loss function is Euclidean distance

$$L2(stand, person_i) = (person - stand)^2$$

Here, every next step becomes progressively harder for our customers. The cost of walking 1 meter is 1 (unit of effort). But walking 2 is $2^2 = 4$ and is $3^2 = 9$ for 3 meters. Thus, the penalty (cost/loss) for being further away from your stand increases as a power law. Still, one needs to sell ice-cream, so one needs to find the best spot where total cost is minimal

$$L2(stand, person) = \sum_{i=1}^{N} (person_i - stand)^2$$



Or, we can compute the minimal *average* cost by dividing the sum by the total number of customers `N`:

$$< L2(stand, person) >= \frac{1}{N} \sum_{i=1}^{N} (person_i - stand)^2$$

Conceptually, you find that minimum by walking along the beach in the direction that reduces the cost until you hit the point where it start going up again. This strategy is called *gradient descent* and, generally speaking, this is how computer finds minima computationally: They make steps in different directions to see which way is down and keep going until things start going up. However, in one-dimensional well-behaving case we have here things are even simpler as you can use calculus to figure out the solution analytically. If you watched the videos I advertised above, you'll know that the *derivative* of the function is zero at the extrema (minima or maxima), so we just need to differentiate our average *L2* over position of the stand and find where it is zero[2].

$$\frac{\partial L2}{\partial stand} = -\frac{2}{N}\sum_{i=1}^{N}(person_i - stand)$$

As we want

$$\frac{\partial L2}{\partial stand} = 0$$

we state

$$\frac{2}{N}\sum_{i=1}^{N}(person_i - stand) = 0.$$

Opening up brackets and rearranging we get

$$-\frac{2}{N}\sum_{i=1}^{N}person_i + \frac{2\cdot N}{N}\cdot stand = 02\cdot stand = \frac{2}{N}\sum_{i=1}^{N}person_i stand = \frac{1}{N}\sum_{i=1}^{N}person_i$$

So, the optimal location of your stand is the *mean*: an average location of all people on the beach.

2.5 L1 (median) vs. L2 (mean)

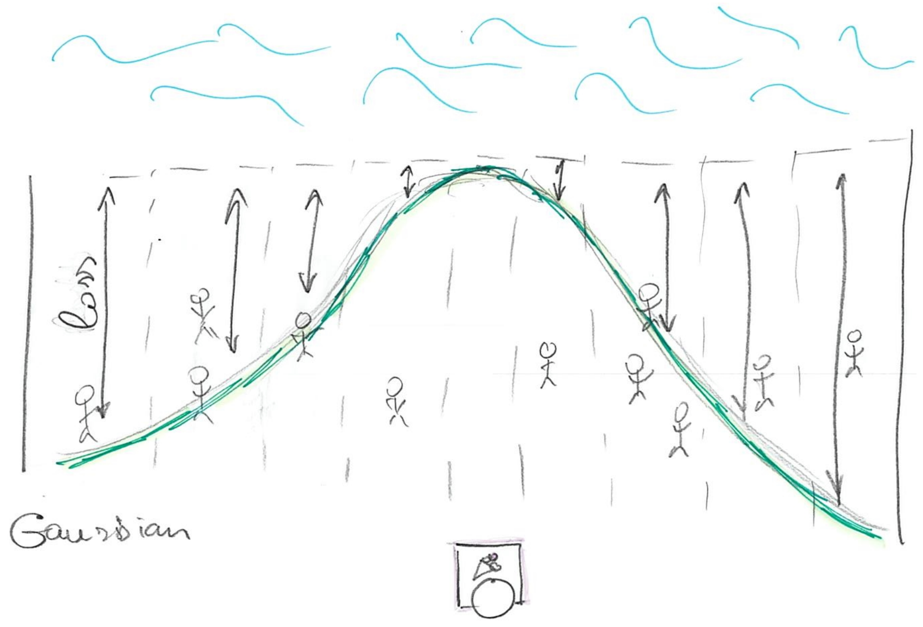
One problem about the *mean* is that it is sensitive to outliers. Because the costs grow as a power law, this approach favors a lot of medium-sized distances over lots of smalls ones plus one really large one. Thus, a single person at a far side of the beach would have a big influence on your stand's location (you already saw the difference in the example above). In data analysis, this means that those outliers will pull your estimates away from the majority of responses. Which is why it might be a good idea to consider using `median` rather than `mean`. If you distribution is symmetric, the difference will be negligible but in presence of outliers `median`, as a point-estimate, is more robust.

[2]I've nicked the derivations from [https://stats.stackexchange.com/a/312997]

2.6 Choosing a likelihood

So far we talked about selling ice-cream on the beach but same question of choosing your loss function applies when you are trying to fit a distribution or a regression line, as in chapter 4. Here, you also have a point-estimate (regression line at each point) and you try to put it in such a way as to minimize the costs of having data points off that line (the distance from the point-estimate of the line and each data point is called a *residual*). The classic way is to use *L2* distance and the approach is called *ordinary least squares*, as you try to minimize squared residuals.

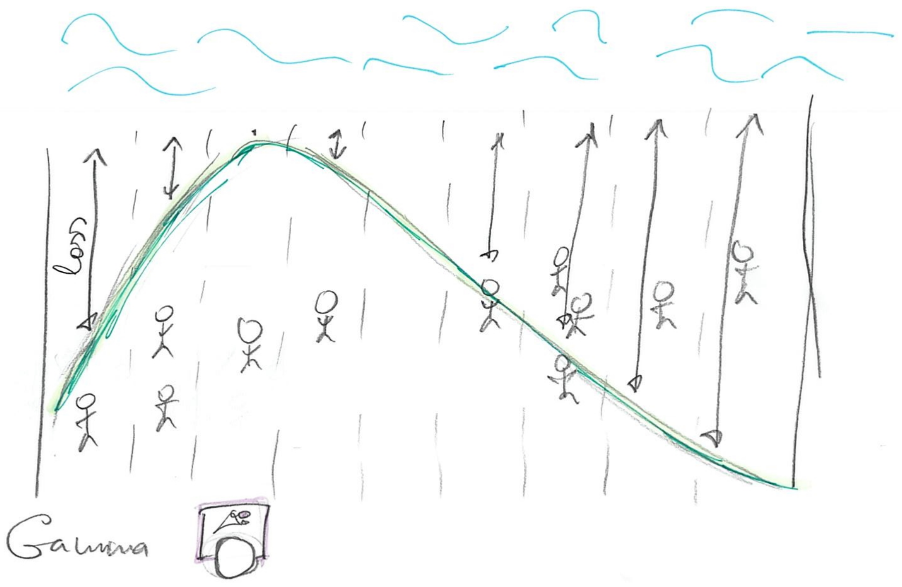
Alternatively, you can express same costs-of-being-off-the-line using a distribution, such as Gaussian. You put its peak (mean) at the (candidate) location of your point estimate (that point has highest probability, so lowest cost) and the loss is computed as a probability of the residual (distance-to-the-point). You can think about it in terms of the probability that a person will go and buy ice-cream from your stand.



The Gaussian is special because it uses L2 distance, see $(x-\mu)^2$ inside the exponential:

$$f(x) = \frac{1}{\sigma\sqrt(2\pi)} e^{\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)}$$

so using it is equivalent to fitting via ordinary least squares. However, as McElreath hinted, you can choose different priors that are different not only in the distance-to-loss formula (like *L1* is different from *L2*) but also in symmetry. Both *L1* and *L2* (and Gaussian) ignore the sign of the distance. It does not matter whether customers are on the left or on the right. Other distributions, such as Beta, Gamma, or Log Normal are not symmetric, so the same distance will cost differently depending on the side the customer is at.



This allows you to think about the choice of your likelihood distribution in terms of choosing a loss function. Both describe how tolerant you are for points to be off the point estimate (regression line). For example, a t-distribution has heavier tails than a Gaussian (if you want to sound like a real mathematician, you say "leptokurtic"), so its losses for outliers (penalty for larger residuals) are lower. Using it instead of a Gaussian would be similar to changing the loss function from L2 to be more like L1 (e.g. $|person_i - stand|^{1.5}$). Conversely, you can pick a symmetric distribution that is narrower than a Gaussian to make residuals penalty even higher (e.g. using $(person_i - stand)^4$). You can also consider other properties: Should it be symmetric? Should it operate only within certain range (1..7 for a Likert scale, 0..1 for proportions, positive values for Gamma)? Should it weight all points equally? As you saw in the examples above, picking a different function moves your cart (regression line), so you should keep in mind that using a different likelihood will move the regression line and produce different estimates and predictions.

How do you pick a likelihood/loss function? It depends on the kind of data you have, on your knowledge about the process that generated the data, robustness of inferences in the presence of outliers, etc. However, most real-life cases you are likely to encounter will be covered by the distributions described in the book (Gaussian, exponential, binomial, Poisson, Gamma, etc.). After finishing the book, you will have a basic understanding of which are most appropiate in typical cases. The atypical cases you'll have to research yourself!

2.7 Gaussian in frenquentist versus Bayesian statistics

Later on in the book McElreath will note that erroneously assuming normal distribution for residuals ruins your inferences in frequentist statistics but not in Bayesian. This is because picking a distribution means different things in frequentist and Bayesian. As I wrote above, in the Bayesian case, likelihood is merely a loss function that translate distance from a data point to a regression line (residual) into a penalty (again, it determines just how tolerant you are for points off the line). Thus, you are using penalties for *observed residuals* and having a bad loss function will make your posterior distribution suboptimal but you still can make inferences because it still is based on your actual residuals.

In contrast, in frequentist statistics, when you are stating that your observed residuals are a sample from a particular distribution, your actual residuals are used to determine parameters of this distribution. Then, however, you make your inferences using *that distribution* not the residuals themselves. This is a very strong conjecture and probably the biggest leap of faith in frequentist statistics saying "I know the true distribution". Problem is, if you got your likelihood function / distribution wrong, your inferences are based on a model that describes *something else* not your data. For example, you have a proportion data but you assume Gaussian distribution for residuals and build a model as if your residuals are always symmetrically distributed (not squashed on one side by floor or celing). That model will not be about your data, it will be about normally distributed *something else*. The numbers for that *something else* may look good (or bad) but they are not the numbers you are interested in. This is a mistake that is remarkably easy to do because computers won't stop you from making it. Think back to Chapter 1: Golems don't care! You can abuse any statistical model/test and they will simply spit out the numbers, even if tests are completely unsuitable for your data. Making sure that distribution is correct and that you are doing the right thing is on you, not the Golem!

Chapter 3

Multiple Regression and Causal Reasoning

This notes are on Chapter 5 "The Many Variables & The Spurious Waffles", specifically, on section 5.1 "Spurious association". If you are to remember one thing from that chapter, it should be "doing multiple regression is easy, understanding and interpreting it is hard".

3.1 Peering into a black box

To better understand the relationship between data and statistical analysis on the one hand and DAGs (directed acyclic graphs) on the other hand, I came up with an electric engineering metaphor[1]. Imagine that you have an electric device that you cannot take apart. However, there are certain points where you can connect your multimeter and check whether current flows between these two points. You also have a schematics for the device but you have no idea whether it is accurate. The names of the connector nodes match but *the connectivity* between them is anybody's guess. So, what do you do? You look at the schematics and identify two nodes where current should *definitely flow* and you measure whether that is the case. Do you have signal? Good, that means that *at least with respect to the connectivity between these two nodes* you schematics is not completely wrong. No signal? Sorry, your schematics is no good, find a different one or try making one yourself. Conversely, you can identify two nodes, so that *no current* should flow between them and measure it *empirically.* No current? Good! Current? Bad, your schematics is wrong.

[1]Dear electric engineers, yes, I know that's not quite how it works but it is still a good metaphor!

The relationship between the device and the schematics is that of the large (device) and small (schematics) world. Your job is to iteratively adjust the latter, so that it matches the former. You need to keep prodding the black box until predictions from your schematics start matching the actual readings. Only then you can say that you understand how device works and you can make predictions about what it will do under different conditions. Although testing based on schematics can be automated, generating such schematics is a creating process. It depends on our knowledge about devices of that kind and about individual exposed nodes.

Hopefully, you can see how it maps on our observational or experimental data (large world, readings from the device) and DAGs (small world, presumed schematics). As a scientist, you *guess*[2] the causal relationships between individual variables and you draw a schematics for that (a DAG). Using this DAG, you identify pairs of variables that must be dependent or independent *assuming your DAG is correct* and check the data on whether this is indeed the case. It is? Good, your DAG is not (entirely) wrong! They are not? Bad news, their causal relationship is different from what you (or others in prior work) came up with. You need to modify DAG or draw a completely different one, just like a engineer must modify the schematics.

Note that this process is the opposite to that in a typical multivariate analysis approach using, say, ANOVA. In the latter case, you throw *everything* together, including some/all interactions between the terms, and try figuring out causal relationship between individual independent and the dependent variable based on magnitude and significance of individual terms. So, *first* you run the statistical analysis and *then* you make your inferences about causal relationships. In causal reasoning using DAG, you *first* formulate causal relationships and *then* you use statistical analysis to check whether these relationship are correct. The second approach is much more powerful, because you can run ANOVA only once but you can formulate many DAGs that describe your data and test them iteratively, refining your understanding step-by-step. Moreover, ANOVA (or any other regression model like that) is about identifying relationships between invididual independent and the signle dependent variable (individual coefficients tell you how independent variables can be used to predict the dependent). DAG-based analysis allows you to predict and test relationship between pairs of *dependent* variables as well, decomposing your complex model into *testable* and *verifyable* chunks. It is a more involved and time-consuming approach but it gives you much deeper understanding of the process you are styding compared to "throw everything into ANOVA and hope it will magically figure it out".

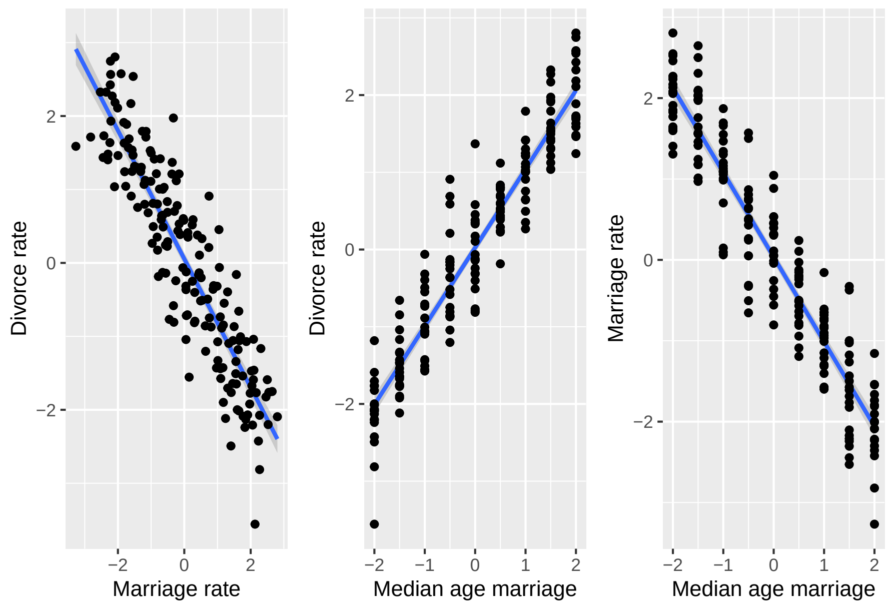
[2]If you feel that science is not about guessing, you are wrong and I have Richard Feynman on my side! You always start by guessing a particular law or rule and then use empirical data to check whether your guess was correct. If you made an *educated* guess, your chances of it being correct are higher, so your job is to study the field and prior work to make your guess as educated as possible. But, at the end, it is still a guess and it can be wrong. Good news, at least in my experience, is that guesses that turn out to be spectacularly wrong are the most informative ones, as they reveal something unexpected and, thus, hitherto unknown about the process.

Moreover, causal calculus via DAGs has another trick up its sleeve. You can use *conditional probabilities* (see below) to flip the relationship between variables. If two variables are independent, they may be dependent when conditioned on some third variable. Or vice versa, depedent variables can become *conditionally independent*. This means that your predictions about connectivity between pairs of variables can be both more specific (e.g., they are related via a third variable or they both independently cause that third variable) and more testable, as you now have two *opposite* predictions for the same pair of variables! You can check that current flows when it should (unconditional probability) and *does not flow* once you condition it on the third variable. Both tests match? DAG is not that bad! One or none match? Back to the drawing board!

3.2 Turning unconditional dependence into conditional independence

Below, you will see how multiple regression can show conditional independence of two variable in case of the fork DAG in divorce data. However, there is another more general way to understand this concept it terms of conditional probabilities $Pr(M|A)$ and $Pr(D|A)$. For this, it helps to understand condition probabilities as *filtering* operation. When we compute conditional probability for a *specific* value of A, this means that we slice the data, keeping only whose values of M and D that correspond to that chosen level of A. It is easier to understand, if we visualize that conditional-probability-as-filtering in synthetic data. For illustration purposes, I will synthesize divorce data, keeping the relationships but I will space marriage age linearly and generate the data so that there are 20 data points for each age (makes it easier to see and understand).

```
set.seed(84321169)
N <- 180
sigma_noise <- 0.5
# we repeat each value of age 10 times to make filtering operation easier to see
sim_waffles <- tibble(MedianAgeMarriage = rep(seq(-2, 2, length.out=9), 20),
                      Divorce = rnorm(N, MedianAgeMarriage, sigma_noise),
                      Marriage = -rnorm(N, MedianAgeMarriage, sigma_noise))

MD_plot <-
  ggplot(data=sim_waffles, aes(x=Marriage, y=Divorce)) +
    geom_smooth(method="lm", formula=y~x) +
    geom_point() +
    xlab("Marriage rate") +
    ylab("Divorce rate")

AD_plot <-
  ggplot(data=sim_waffles, aes(x=MedianAgeMarriage, y=Divorce)) +
    geom_smooth(method="lm", formula=y~x) +
    geom_point() +
    xlab("Median age marriage") +
    ylab("Divorce rate")

AM_plot <-
  ggplot(data=sim_waffles, aes(x=MedianAgeMarriage, y=Marriage)) +
    geom_smooth(method="lm", formula=y~x) +
    geom_point() +
    xlab("Median age marriage") +
    ylab("Marriage rate")

MD_plot | AD_plot | AM_plot
```

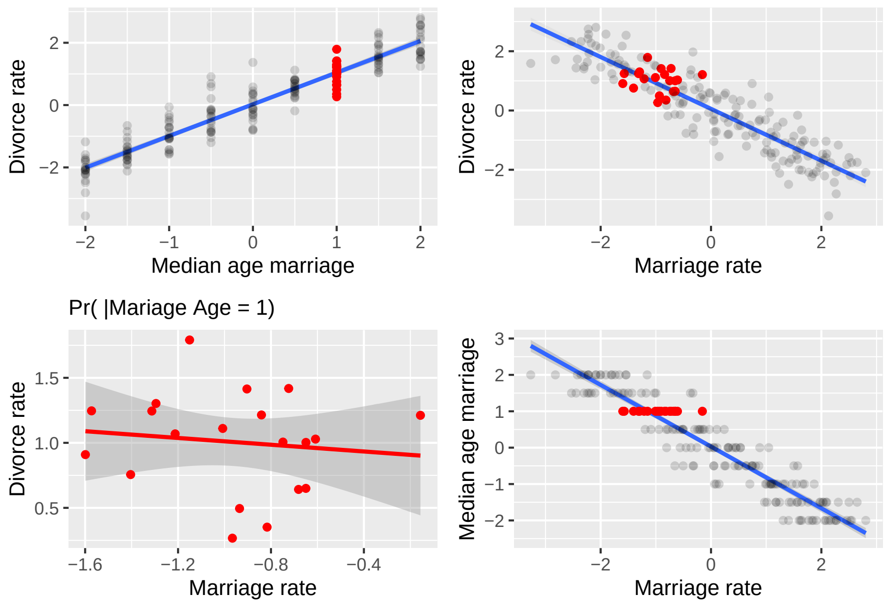


As you can see, all variables being dependent on each other, as in case of the original data. However, let us pick an arbitrary value, say $A = 1$[3] and see which dots on the left plot will be selected via *filtering* on that value.

```
MD_plot <-
  ggplot(data=sim_waffles, aes(x=Marriage, y=Divorce)) +
    geom_smooth(method="lm", formula=y~x, alpha=0.1) +
```

[3]The data is "standardized", therefore, age of 1 is one standard deviation away from the mean marriage rate.

```r
    geom_point(data=sim_waffles %>% filter(MedianAgeMarriage == 1), color="red") +
    geom_point(data=sim_waffles %>% filter(MedianAgeMarriage != 1), alpha=0.15) +
    xlab("Marriage rate") +
    ylab("Divorce rate")
AD_plot <-
  ggplot(data=sim_waffles, aes(x=MedianAgeMarriage, y=Divorce)) +
    geom_smooth(method="lm", formula=y~x) +
    geom_point(data=sim_waffles %>% filter(MedianAgeMarriage == 1), color="red") +
    geom_point(data=sim_waffles %>% filter(MedianAgeMarriage != 1), alpha=0.15) +
    xlab("Median age marriage") +
    ylab("Divorce rate")

AM_plot <-
  ggplot(data=sim_waffles, aes(x=Marriage, y=MedianAgeMarriage)) +
    geom_smooth(method="lm", formula=y~x) +
    geom_point(data=sim_waffles %>% filter(MedianAgeMarriage == 1), color="red") +
    geom_point(data=sim_waffles %>% filter(MedianAgeMarriage != 1), alpha=0.15) +
    ylab("Median age marriage") +
    xlab("Marriage rate")

MD_A1_plot <-
  ggplot(data=sim_waffles %>% filter(MedianAgeMarriage == 1),
         aes(x=Marriage, y=Divorce)) +
    geom_smooth(method="lm", formula=y~x, color="red") +
    geom_point(color="red") +
    xlab("Marriage rate") +
    ylab("Divorce rate") +
    labs(subtitle = "Pr( |Mariage Age = 1)")

(AD_plot | MD_plot) /
(MD_A1_plot |AM_plot)
```



First, take at top-left and bottom-right plots that plot, correspondingly, divorce and marriage rate versus marriage age. Note that I've flipped axes on the bottom-right plot, so that marriage rate is always mapped on x-axis. The *red* dots in each plot correspond to divorce and marriage rates *given that* (filtered on) marriage age is 1. The same dots are marked in red in the top-right plot and are plotted separately at the bottom-right. As you can see, the two variables *conditional on A=1* are uncorrelated. Why? Because both were fully determined by marriage age and any variation (the spread of red dots vertically or horizontally in the top-left and bottom-right plots) was due to noise. Therefore, the plot on the bottom-left effectively plots noise in marriage rate versus noise in divorce rate and, by our synthetic data design, the noise in two variable was independent, hence, lack of correlation.

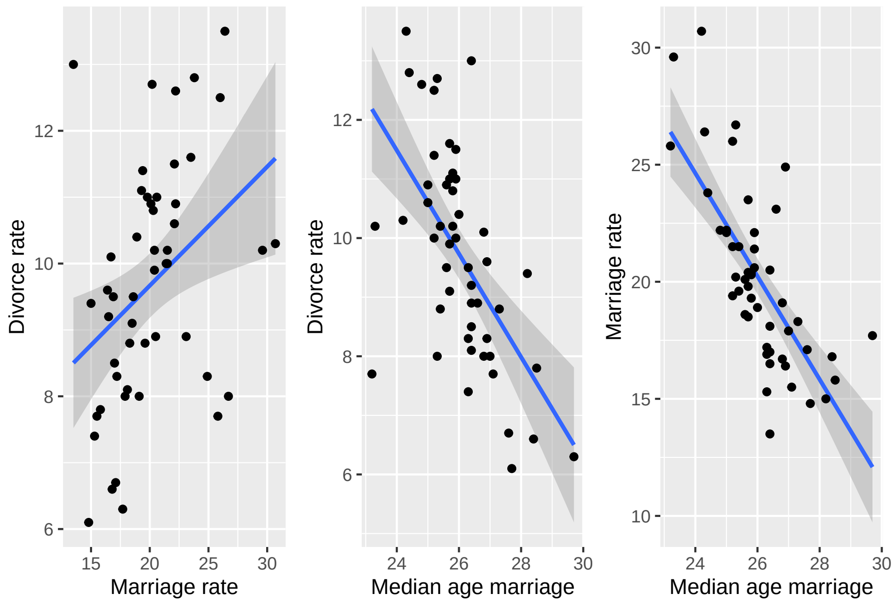
This opportunity to turn dependence into independence by conditioning two variables on the third is at the heart of causal reasoning[4]. You draw a DAG and if it has a fork in it, you know that *given your educated guess about causal relationship is correct* (small world!), your data (large world!) should show dependence *and* conditional independence of the two variables (divorce and marriage rates). What if it does not? E.g., the two variables are always dependent or always independent, conditional or not? As we already discussed above, that just means that your educated guess was wrong and that relationship of the variables is different from how you thought it is. Thus, you need to go back to the drawing board, come up with another DAG, repeat the analysis and see if

[4]As you will learn later, the opposite is also true, so you can turn independence into conditional dependence.

that new DAG is supported by data. If you have more variables, your DAGs will be more complex but the beauty of such causal reasoning is that you can concentrate on *parts* of it, picking three variables and seeing whether your guess about these three variables was correct. This way, you can tinker with your causal model part-by-part, instead of hoping that you got *everything* right on the very first attempt.

3.3 Magic of multiple regression

When reading section 5.1 "Spurious association", I found relationships between the *marriage age*, *marriage rate*, and *divorce rate* to be both clear and mysterious. On the one hand, everything is correlated with everything.



On the other hand, once we fit linear model to predict divorce rate based on both median age marriage and marriage rate, the latter is *clearly* irrelevant (output of code 5.11 shows that its coefficient is effectively zero, meaning that it is ignored) and, therefore, it has no causal influence on divorce rate.

If you are like me[5], you said "Huh! But how does the model know that?". And, at least for me, explanations in the chapter did not help much. The key figure is 5.4, that shows that (omitting intercept and slope symbols) `median age`

[5]Don't be like me, be better!

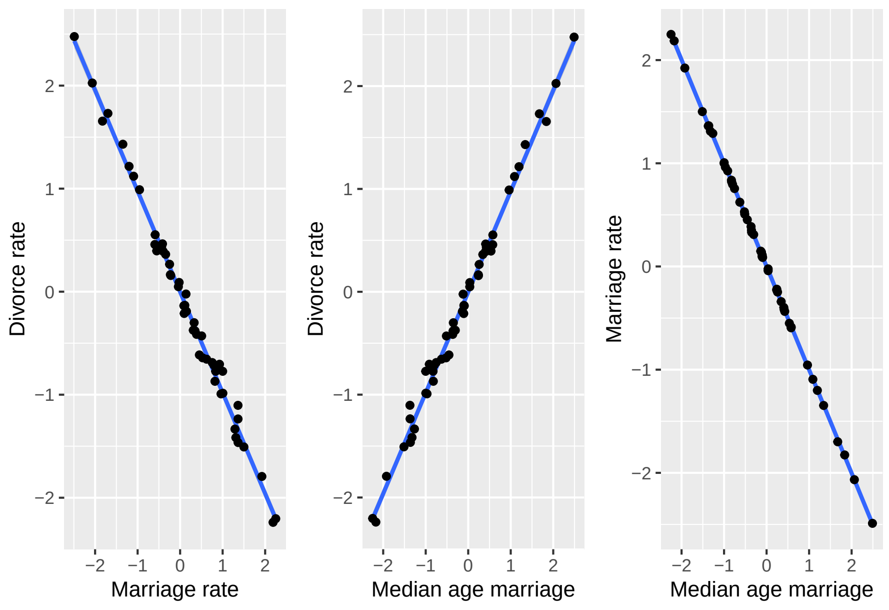
`marriage = marriage rate + *extra information*` but `marriage rate = median age marriage`. In a nutshell, both variables code the same information but there is *more* of it in median age than in marriage rate, so the latter is ignored. Unfortunately, the answer "but how?" still stands. The figure 5.4, which shows fits on residuals is illustrative, but we do not fit residuals, we fit both variables at the same time *without* fitting them on each other! Nowhere in the model 5.1.4 do we find

$$\mu_i^M = \alpha_{AM} + \beta_{AM} * A_i$$

So, what's going on? *How does it know?* To understand this, let us start with an issue of *multicollinearity*.

3.4 Multicollinearity

To make things easier to understand, let us use simulated data. Imagine that both marriage and divorce rate are almost perfectly linearly dependent on marriage rate, so that $D_i = \beta_A^{true} \cdot A_i$ and $M_i = -A_i$. For the sake of simplicity $\beta_A^{true} = 1$. We pretend our variables are already standardized, so the plots would look something like this.



The relationship is the same as in the plots above but, as we assumed an almost perfect correlation, there is not much spread around the line. Still, by definition of how we constructed the data, both marriage and divorce rate are *caused* (computed from) median age and marriage rate is never used to compute the divorce rate. What happens if we analyze this simulated data using the same model 5.1.3, will it be able to figure “marriage rate does not matter” again?

```
sim_waffles <-
  sim_waffles %>%
  mutate(A = MedianAgeMarriage,
         M = Marriage,
         D = Divorce)

sim_waffles_fit <- quap(
  alist(
    D ~ dnorm(mu , sigma) ,
    mu <- a + bM*M + bA*A,
    a ~ dnorm(0, 0.2),
    bA ~ dnorm(0, 10),
    bM ~ dnorm(0, 10),
    sigma ~ dexp(1)
  ),
  data = sim_waffles,
)

precis(sim_waffles_fit)
```

```
##                mean          sd        5.5%      94.5%
## a      -0.002040905 0.012914885 -0.02268138 0.01859958
## bA     -0.271365691 1.189014206 -2.17164004 1.62890866
## bM     -1.249741284 1.187251535 -3.14719854 0.64771597
## sigma   0.090132561 0.008997396  0.07575298 0.10451214
```

Oh no, we broke it! β_M is now about `-1.25` rather than zero and β_A is around `-0.27` rather than one, as it should. Also note the uncertainty associated with both values, as they both totally overlap with zero[6]. So, the data generation process is the same (`Divorce rate ← Age → Marriage rate`) and the model is the same (prior changes have no particular impact in this case) but the “magic” of inferring the lack of `Divorce rate → Marriage rate` is gone! The *only* difference between the two data sets is extra variance (noise) in marriage rate variable, so let us see how the absence of that extra noise in simulated data breaks the magic.

When two variables, marriage age and rate in our case, are (almost) perfectly correlated, that means that you can substitute one for another. Thus, when we

[6]I’ve made priors for both betas broad, so that they are not pushed towards zero too aggressively and their uncertainty is more evident

write[7]

$$D = \beta_A \cdot A + \beta_M \cdot M$$

because `M = -A` (that's how we generated the data!), we can substitute `-A` for `M`

$$D = \beta_A \cdot A - \beta_M \cdot AD = (\beta_A - \beta_M) \cdot AD = \beta_A^{true} \cdot A$$

where

$$\beta_A^{true} = (\beta_A - \beta_M)$$

That last bit is the curse of multicollinearity, because if two variable have *the same* information, you are, effectively, fitting their *sum*! This is equivalent to fitting the sum[8] of coefficients times one of the variables (does not matter which one, since they are identical, we used `A` but could have used `M`). If you look at the precis output above, you will see exactly that! `bA = -0.27` and `bM = -1.25`, so plugging them in gives us

$$\beta_A^{true} = (\beta_A - \beta_M) = (-0.27 - (-1.25)) = 0.98$$

Hey, that is the slope that we used to construct divorce rate, so fitting does work! But what about uncertainty for *individual* slopes? It stems directly from the fact that $\beta_A^{true} = (\beta_A - \beta_M) = 1$ (it is `1` just in our case, of course), as there are infinite number of pairs of numbers whose difference would give 1: `2-1`, `3-2`, `(-200)-(-201)`, `1000.41-999.41`, etc. All of them add up (subtract to) one, so the fitting procedure cannot settle on any specific region for each parameter and any specific pair of values. Any number will do, as long as the *other one* differs by one. This phenomenon of *masked relationship* is the focus of section 5.2, so you will learn more about it soon.

3.5 Back to spurious association

Above, you have learned that if two variable have the same information, you can only fit *both* of them but cannot get individual slopes. But wasn't that the case for real data we started with? Marriage age and rate *are* correlated, so why fitting used one (age) and not their sum? The answer is *extra noise* in marriage rate. In the real data marriage rate is age *plus some noise*: $M = -(A + \epsilon)$, where ϵ is traditionally used to denote "some noise". How does that change our linear model for divorce rate?

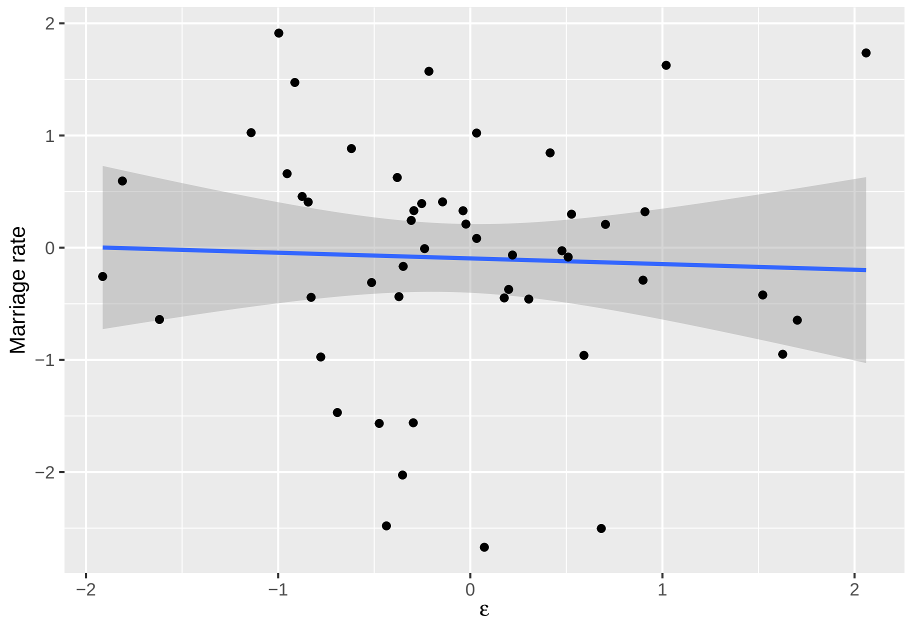
$$D = \beta_A \cdot A - \beta_M \cdot AD = \beta_A \cdot A - \beta_M(A + \epsilon)D = (\beta_A - \beta_M) \cdot A - \beta_M \cdot \epsilon$$

By definition, ϵ is *pure noise* and has zero predictive value with respect to divorce rate. This, if we would fit it *alone*, we would expect to get a slope near zero (that is your "no significant relationship").

[7]I've dropped likelihood and variance only to compress formulas and shed unimportant details. Adding them does not change the essence.

[8]in our case, the difference, because we defined that `M = -A`

```
set.seed(1231455)
sim_waffles <- tibble(MedianAgeMarriage = rnorm(N),
                      Divorce = rnorm(N, MedianAgeMarriage, 0.1),
                      Marriage = -rnorm(N, MedianAgeMarriage, 0.01),
                      epsilon = rnorm(N))

ggplot(sim_waffles, aes(x=epsilon, y=Divorce)) +
  geom_smooth(method="lm", formula=y~x) +
    geom_point() +
    xlab(expression(epsilon)) +
    ylab("Marriage rate")
```



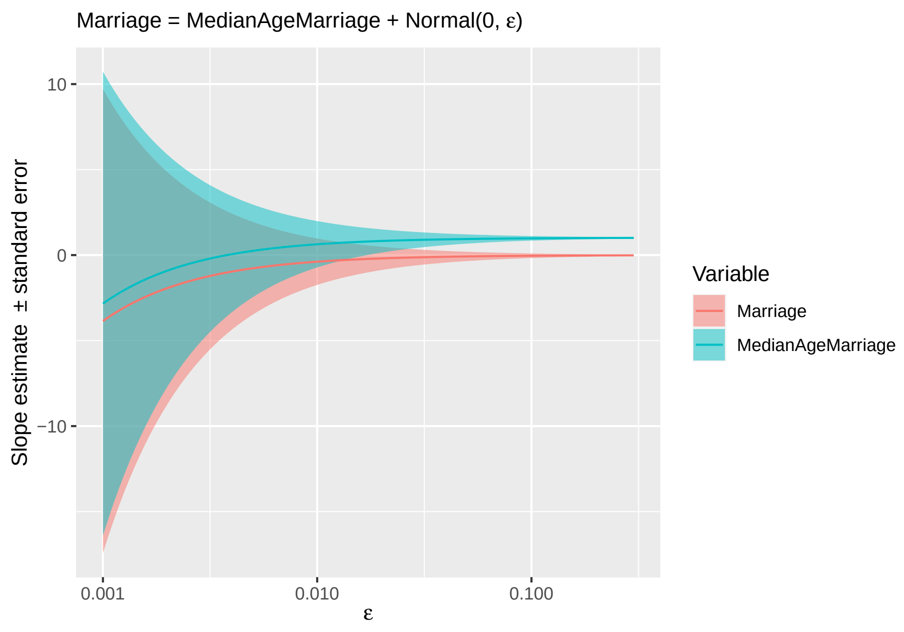
But we are not fitting it alone, as the coefficient β_M appears at *twice*:

$$D = (\beta_A - \beta_M) \cdot A - \beta_M \cdot \epsilon$$

The latter part, $\beta_M \cdot \epsilon$, pushes β_M towards zero (model ignores pure noise by assigning slope of zero). But the former part, $(\beta_A - \beta_M)$ only needs to add up to β_A^{true}, so however we fix β_M, β_A can accommodate. Thus the closer β_M to zero, the closer is β_A to β_A^{true}. And that's how the magic works! If one variable is other variable plus noise, that *plus noise* induces extra penalty and the only way to reduce residuals is to ignore the noise by setting the slope to zero. Therefore, you ignore the variable as well, because it is merely a noisy twin of a better variable. You can see how added noise "disambiguates" the causal relationship[9].

```r
simulate_waffles <- function(sigma_noise){
  # generate same data but for noise in Marraige from Age relationship
  set.seed(169084)
  sim_df <- sim_waffles <- tibble(MedianAgeMarriage = rnorm(N),
                                  Divorce = rnorm(N, MedianAgeMarriage, 0.1),
                                  Marriage = -rnorm(N, MedianAgeMarriage, sigma_noise)

  # fit data using OLS and pulling out two slope coefficients
  lm(Divorce ~ Marriage + MedianAgeMarriage, data=sim_df) %>%
    summary() %>%
    .$coefficients %>%
    data.frame() %>%
    rownames_to_column("Variable") %>%
    slice(-1) %>%
    mutate(LowerCI = Estimate - Std..Error,
           UpperCI = Estimate + Std..Error) %>%
    select(Variable, Estimate, LowerCI, UpperCI)
}

simulated_noise <-
  tibble(epsilon =exp(seq(log(0.001), log(0.3), length.out = 100))) %>%
  group_by(epsilon) %>%
  do(simulate_waffles(.$epsilon))

ggplot(simulated_noise, aes(x=epsilon, y=Estimate)) +
  geom_ribbon(aes(ymin=LowerCI, ymax=UpperCI, fill=Variable), alpha= 0.5) +
  geom_line(aes(color=Variable)) +
  scale_x_log10(name=expression(epsilon)) +
  ylab("Slope estimate  ± standard error") +
  labs(subtitle = expression(paste("Marriage = MedianAgeMarriage + Normal(0, ", epsilo
```

[9]I've used ordinary least squares just to make simulations faster. You will get the same result using Bayesian fittings procedures.



The stripes show uncertainty (estimate ± standard error) and you can appreciate how quickly it is reduced as marriage rate becomes noisier and just how little noise is required for "magic" to start working and converge on the true causal relationships.

3.6 Warning

So, a bit of noise will fix everything? Not necessarily! If *both* marriage age and rate depend on some *third latent variable*, it will depend, for example, on which variable is noisier. The book will continue with this theme, so you will learn more but, in the meantime, the key take-home message is still "doing multiple regression is easy, understanding and interpreting it is hard".

Chapter 4

Solutions for Chapter 2

```
library(tidyverse)
library(rethinking)
```

2E1

Which of the expressions below correspond to the statement: the probability of rain on Monday?

1. Pr(rain)
2. **Pr(rain | Monday)**
3. Pr(Monday | rain)
4. Pr(rain, Monday) / Pr(Monday)

2E2

Which of the following statements corresponds to the expression: Pr(Monday|rain)?

1. The probability of rain on Monday.
2. The probability of rain, given that it is Monday.
3. **The probability that it is Monday, given that it is raining.**
4. The probability that it is Monday and that it is raining.

2E3

Which of the expressions below correspond to the statement: the probability that it is Monday, given that it is raining?

1. **Pr (Monday | rain)**
2. Pr(rain | Monday)
3. Pr(rain | Monday) Pr(Monday)
4. **Pr(rain | Monday) Pr(Monday) / Pr(rain)**
5. Pr(Monday | rain) Pr(rain) / Pr(Monday)

2E4

The Bayesian statistician Bruno de Finetti (1906– 1985) began his 1973 book on probability theory with the declaration: "PROBABILITY DOES NOT EXIST." The capitals appeared in the original, so I imagine de Finetti wanted us to shout this statement. What he meant is that probability is a device for describing uncertainty from the perspective of an observer with limited knowledge; it has no objective reality. Discuss the globe tossing example from the chapter, in light of this statement. What does it mean to say "the probability of water is 0.7"?

He meant that, at least at macro level, there are no truly random events. At that scale, all events, such as motion of a globe throughout its flight, are described by deterministic laws of physics. Therefore, the outcome of globe tossing is deterministic and predictable if we have a complete knowledge about its initial momentum and about other forces at play (e.g., air movement). In most cases, we do not have the full knowledge and, therefore, our predictions are likely to diverge from the observed outcome and be mostly but not always correct. Hence, the concept of probability. At the end, noise is never truly random, noise is information that we did not include into our model explicitly.

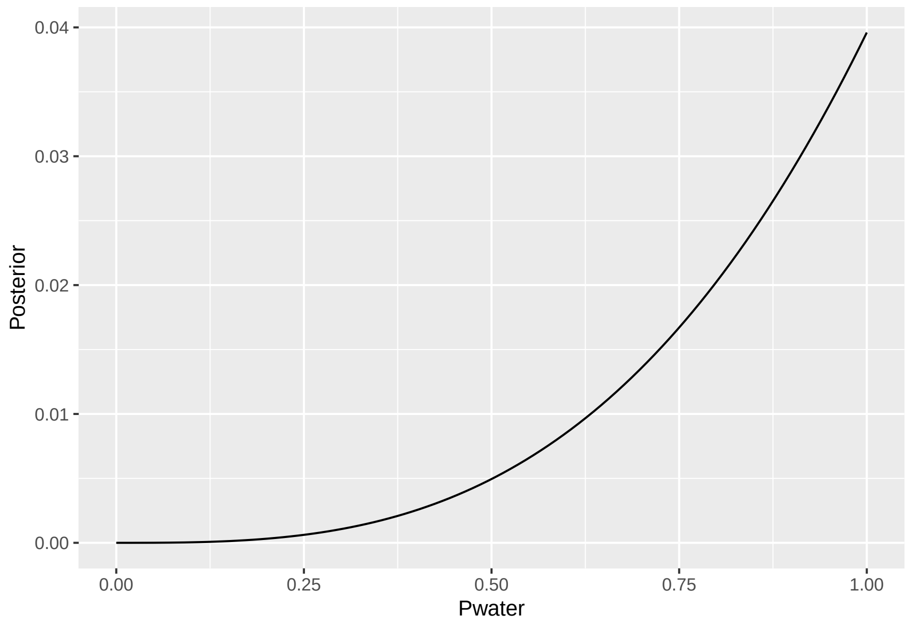
2M1

Recall the globe tossing model from the chapter. Compute and plot the grid approximate posterior distribution for each of the following sets of observations. In each case, assume a uniform prior for p.

```r
#' Computes posterior for water probability assuming binomial likelihood
#'
#' @param observations vector of "W" and "L"
#' @param prior numeric vector, its length determines grid resolution
#'
#' @return tibble with Pwater (from 0 to 1), Posterior (probability of water for given
#'
#' @examples
#' probability_of_water(c("W", "W", "W"), rep(1, 100))
probability_of_water <- function(observations, prior){
  p_grid <- seq(from=0, to=1, length.out=length(prior))
  likelihood <- dbinom(sum(observations == "W"), size=length(observations), prob=p_grid)
  unstandardized_posterior <- likelihood * prior
  tibble(Pwater = p_grid,
         Posterior = unstandardized_posterior / sum(unstandardized_posterior),
         Prior = prior)
}
```

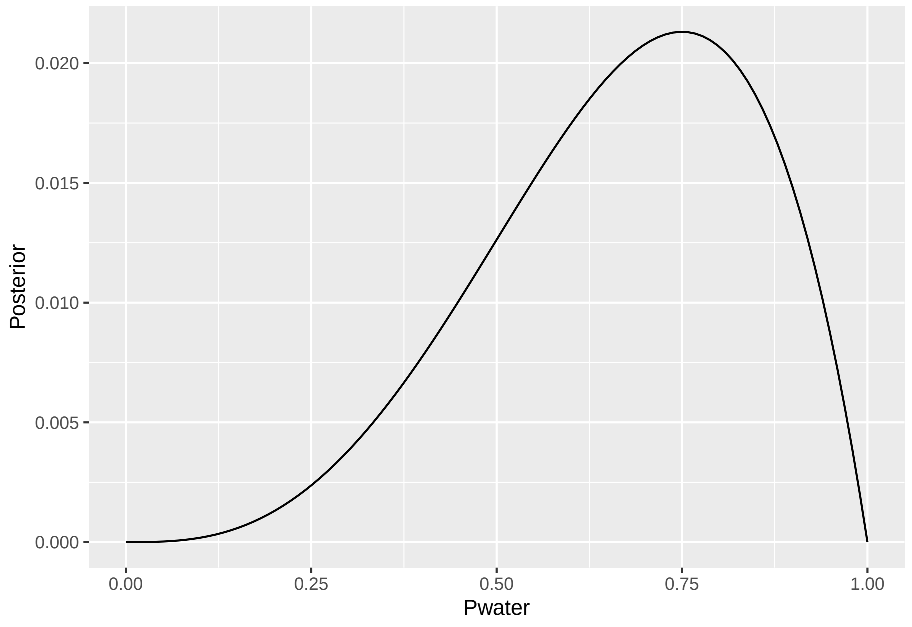
1. W, W, W

```
posterior2M1 <- probability_of_water(c("W", "W", "W"), rep(1, 100))
ggplot(posterior2M1, aes(x=Pwater, y=Posterior)) +
  geom_line()
```



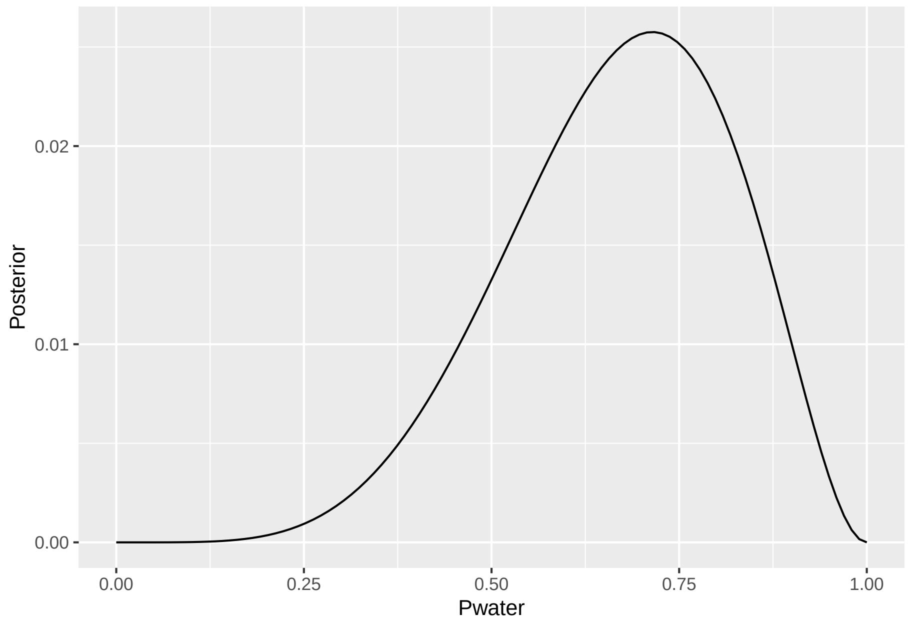
2. W, W, W, L

```
posterior2M2 <- probability_of_water(c("W", "W", "W", "L"), rep(1, 100))
ggplot(posterior2M2, aes(x=Pwater, y=Posterior)) +
  geom_line()
```



3. L, W, W, L, W, W, W

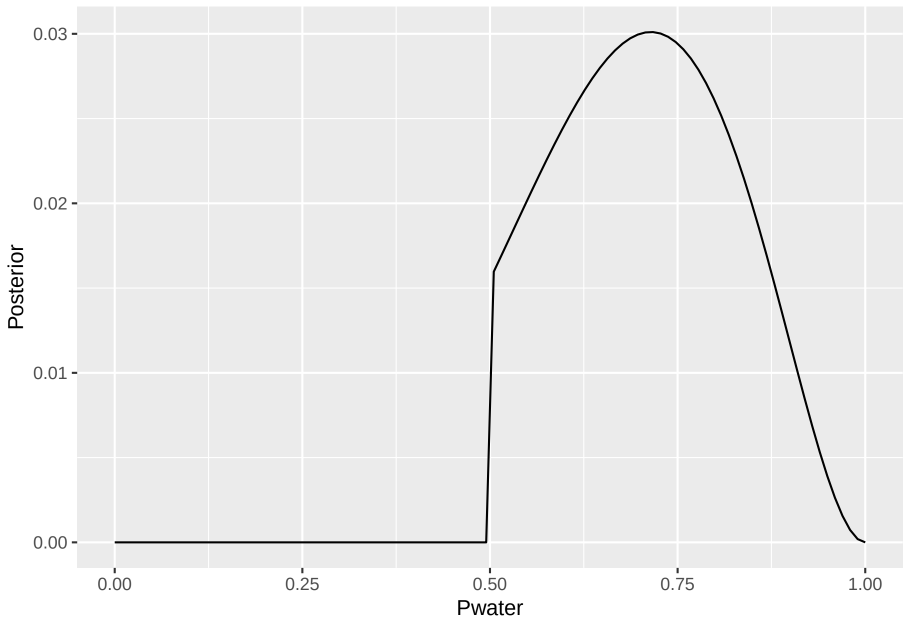
```
posterior2M3 <- probability_of_water(c("L", "W", "W", "L", "W", "W", "W"), rep(1, 100)
ggplot(posterior2M3, aes(x=Pwater, y=Posterior)) +
  geom_line()
```



2M2

Now assume a prior for p that is equal to zero when p<0.5 and is a positive constant when p 0.5. Again compute and plot the grid approximate posterior distribution for each of the sets of observations in the problem just above.

```
p_grid <- seq(from=0, to=1, length.out = 100)
priorM2 <- as.numeric(p_grid >= 0.5)

posterior2M3 <- probability_of_water(c("L", "W", "W", "L", "W", "W", "W"), priorM2)
ggplot(posterior2M3, aes(x=Pwater, y=Posterior)) +
  geom_line()
```



2M3

Suppose there are two globes, one for Earth and one for Mars. The Earth globe is 70% covered in water. The Mars globe is 100% land. Further suppose that one of these globes — you don't know which — was tossed in the air and produced a "land" observation. Assume that each globe was equally likely to be tossed. Show that the posterior probability that the globe was the Earth, conditional on seeing "land" (Pr (Earth | land)), is 0.23.

Bayes Formula tells us that

$$Pr(Planet|land) = \frac{Pr(land|Planet)Pr(Planet)}{Pr(land)}$$

As $Pr(land)$ is a normalization constant, we can ignore it for a moment. Accordingly,

$$uPr(Earth|land) = 0.3 \cdot 0.5 = 0.15 uPr(Mars|land) = 1 \cdot 0.5 = 0.5$$

where $uPr()$ is unstandardized plausibility. Normalizing it, we get

$$Pr(Earth|land) = \frac{0.15}{0.15 + 0.5} = 0.2308$$

2M4

Suppose you have a deck with only three cards. Each card has two sides, and each side is either black or white. One card has two black sides. The second card has one black and one white side. The third card has two white sides. Now suppose all three cards are placed in a bag and shuffled. Someone reaches into the bag and pulls out a card and places it flat on a table. A black side is shown facing up, but you don't know the color of the side facing down. Show that the probability that the other side is also black is 2/3. Use the counting method (Section 2 of the chapter) to approach this problem. This means counting up the ways that each card could produce the observed data (a black side facing up on the table).

- B|B: B(1) → **B(2) : 1**
- B|B: B(2) → **B(1) : 1**
- B|W : B → W : 1
- W|B : 0
- W|W : 0
- W|W : 0

There are three possible outcomes, given the visible side is black and two out of three lead to a black back side: $\frac{2}{3}$.

2M5

Now suppose there are four cards: B/B, B/W, W/W, and another B/B. Again suppose a card is drawn from the bag and a black side appears face up. Again calculate the probability that the other side is black.

- B|B: B(1) → **B(2): 1**
- B|B: B(2) → **B(1) : 1**
- B|W : B → W : 1
- W|B : 0
- W|W : 0
- W|W : 0
- B|B: B(3) → **B(4): 1**
- B|B: B(4) → **B(3) : 1**

There are three possible outcomes, given the visible side is black and four out of five lead to a black back side: $\frac{4}{5}$

2M6

Imagine that black ink is heavy, and so cards with black sides are heavier than cards with white sides. As a result, it's less likely that a card with black sides is pulled from the bag. So again assume there are three cards: B/B, B/W, and W/W. After experimenting a number of times, you conclude that for every way to pull the B/B card from the bag, there are 2 ways to pull the B/W card and 3 ways to pull the W/W card. Again suppose that a card is pulled and a black side appears face up. Show that the probability the other side is black is now 0.5. Use the counting method, as before.

- B|B: B(1) $\rightarrow$ **B(2): 1 $\times$ 1 (prior) = 1**
- B|B: B(2) $\rightarrow$ **B(1) : 1 $\times$ 1 (prior) = 1**
- B|W : B $\rightarrow$ W : 1 $\times$ 2 (prior) = 2
- W|B : 0 $\times$ 2 (prior) = 0
- W|W : 0 $\times$ 3 (prior) = 0
- W|W : 0 $\times$ 3 (prior) = 0

Now the counts are two out of four that other side is black, i.e. 0.5.

2M7

Assume again the original card problem, with a single card showing a black side face up. Before looking at the other side, we draw another card from the bag and lay it face up on the table. The face that is shown on the new card is white. Show that the probability that the first card, the one showing a black side, has black on its other side is now 0.75. Use the counting method, if you can. Hint: Treat this like the sequence of globe tosses, counting all the ways to see each observation, for each possible first card.

Possible card sequences, bold means that back side of the first card is black:

- **B(1)|B(2) $\rightarrow$ W|B : 1**
- **B(2)|B(1) $\rightarrow$ W|B : 1**
- B(1)|B(2) $\rightarrow$ B|W : 0
- B(2)|B(1) $\rightarrow$ B|W : 0
- **B(1)|B(2) $\rightarrow$ W(1)|W(2) : 1**
- **B(2)|B(1) $\rightarrow$ W(1)|W(2) : 1**
- **B(1)|B(2) $\rightarrow$ W(2)|W(1) : 1**
- **B(2)|B(1) $\rightarrow$ W(2)|W(1) : 1**

- B|W → B(1)|B(2) : 0
- W|B → B(1)|B(2) : 0
- B|W → B(2)|B(1) : 0
- W|B → B(2)|B(1) : 0
- B|W → W(1)|W(2) : 1
- W|B → W(1)|W(2) : 0
- B|W → W(2)|W(1) : 1
- W|B → W(2)|W(1) : 0
- W(1)|W(2) → W|B : 0
- W(2)|W(1) → W|B : 0
- W(1)|W(2) → B|W : 0
- W(2)|W(1) → B|W : 0
- W(1)|W(2) → B(1)|B(2) : 0
- W(2)|W(1) → B(1)|B(2) : 0
- W(1)|W(2) → B(2)|B(1) : 0
- W(2)|W(1) → B(2)|B(1) : 0

Total of eight possible path, six of them have black back for the first card: 0.75.

2H1

Suppose there are two species of panda bear. Both are equally common in the wild and live in the same places. They look exactly alike and eat the same food, and there is yet no genetic assay capable of telling them apart. They differ however in their family sizes. Species A gives birth to twins 10% of the time, otherwise birthing a single infant. Species B births twins 20% of the time, otherwise birthing singleton infants. Assume these numbers are known with certainty, from many years of field research.

Now suppose you are managing a captive panda breeding program. You have a new female panda of unknown species, and she has just given birth to twins. What is the probability that her next birth will also be twins?

$uPr(A|twins) = Pr(twins|A) * Pr(A) = 0.1 * 0.5 = 0.05 uPr(B|twins) = Pr(twins|B) * Pr(B) = 0.2 * 0.5 = 0.1$

After normalization $Pr(A|twins) = 1/3$ and $Pr(B|twins) = 2/3$.

Probability that you will see twins again per species is

$Pr(twins|A, twins) = Pr(A|twings)*Pr(twins|A) = 1/3*0.1 = 0.1/3 Pr(twins|B, twins) = Pr(B|twing$

The total probability is $\frac{0.1}{3} + \frac{0.4}{3} = \frac{0.5}{3} \approx 0.167$

2H2

Recall all the facts from the problem above. Now compute the probability that the panda we have is from species A, assuming we have observed only the first birth and that it was twins.

$uPr(A|twins) = Pr(twins|A)*Pr(A) = 0.1*0.5 = 0.05 uPr(B|twins) = Pr(twins|B)*Pr(B) = 0.2*0.5 =$

After normalization $Pr(A|twins) = 1/3$

2H3

Continuing on from the previous problem, suppose the same panda mother has a second birth and that it is not twins, but a singleton infant. Compute the posterior probability that this panda is species A.

$uPr(A|twins) = Pr(twins|A)*Pr(A) = 0.1*0.5 = 0.05 uPr(B|twins) = Pr(twins|B)*Pr(B) = 0.2*0.5 =$

After normalization $Pr(A|twins) = 1/3$ and $Pr(B|twins) = 2/3$.

Given that next birth is a singleton:

$uPr(A|twins, singleton) = Pr(A|twins)*(1-Pr(twins|A)) = 1/3 \cdot 0.9 = 0.9/3 uPr(B|twins, singleton) =$

After normalization $Pr(A|twins, singleton) = 0.36$ (and $Pr(B|twins, singleton) = 0.64$).

2H4

A common boast of Bayesian statisticians is that Bayesian inference makes it easy to use all of the data, even if the data are of different types.

So suppose now that a veterinarian comes along who has a new genetic test that she claims can identify the species of our mother panda. But the test, like all tests, is imperfect. This is the information you have about the test: • The probability it correctly identifies a species A panda is 0.8. • The probability it correctly identifies a species B panda is 0.65.

The vet administers the test to your panda and tells you that the test is positive for species A. First ignore your previous information from the births and compute the posterior probability that your panda is species A. Then redo your calculation, now using the birth data as well.

$uPr(A|test) = Pr(test|A)*Pr(A) = 0.8*0.5uPr(B|test) = (1-Pr(test|B))*Pr(B) = (1-0.65)*0.5 = 0.35*0.5$

After normalization $Pr(A|test) \approx 0.7$ and $Pr(B|test) \approx 0.3$.

Given the twins, this becomes

$uPr(A|test, twins) = Pr(A|test)*Pr(twins|A) = 0.7*0.1 = 0.07uPr(B|test, twins) = Pr(B|test)*Pr(twins|B) = 0.3$

After normalization $Pr(A|test, twins) \approx 0.533$ and $Pr(B|test, twins) \approx 0.467$.

Given the singleton, this becomes

$uPr(A|test, twins, singleton) = Pr(A|test, twins)*(1-Pr(twins|A)) = 0.533*0.9 = 0.48uPr(B|test, twins, singleton$

After normalization $Pr(A|test, twins, singleton) \approx 0.5625$ and $Pr(B|test, twins, singleton) \approx 0.4375$.

Chapter 5

Solutions for Chapter 3

```r
library(tidyverse)
library(rethinking)
```

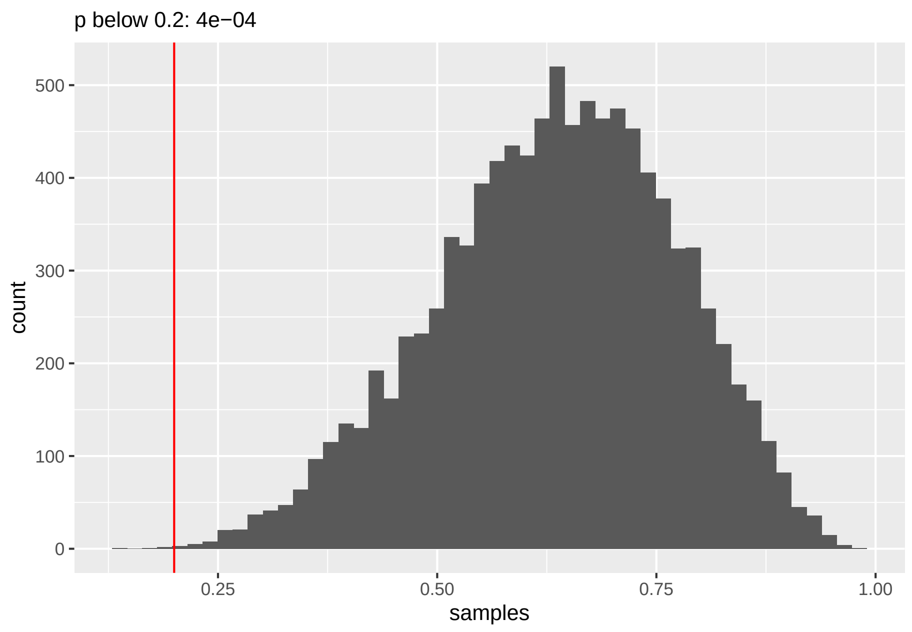
5.0.0.1 Initialization code for easy exercises

```r
p_grid <- seq(from=0, to=1, length.out=1000)
prior <- rep(1,1000)
likelihood <- dbinom(6, size=9, prob=p_grid)
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)
set.seed(100)
samples <- sample(p_grid, prob=posterior, size=1e4, replace=TRUE)
```

3E1

How much posterior probability lies below p = 0.2?

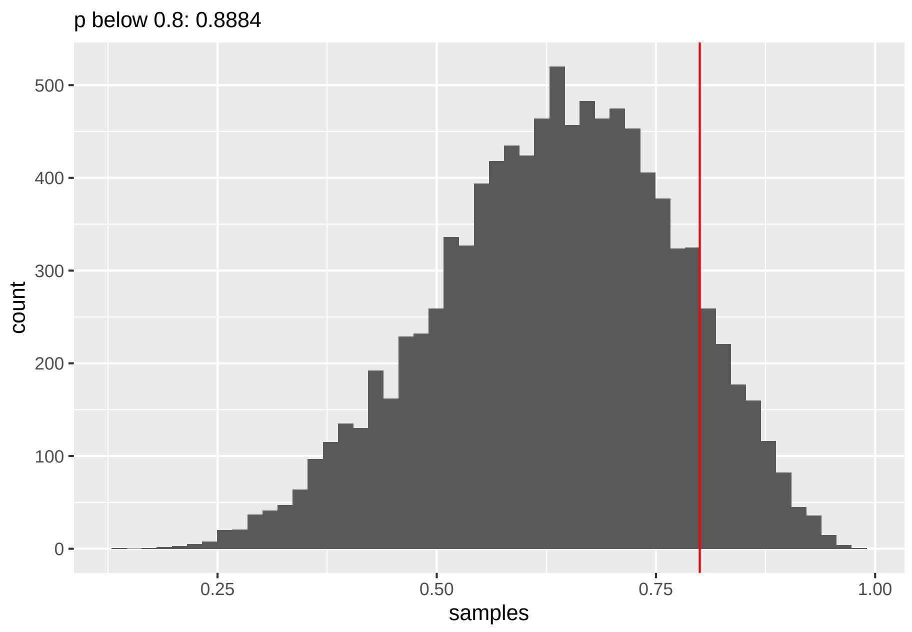
```r
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = 0.2, color="red") +
  labs(subtitle = glue::glue("p below 0.2: {mean(samples < 0.2)}"))
```



3E2

How much posterior probability lies below p = 0.8?

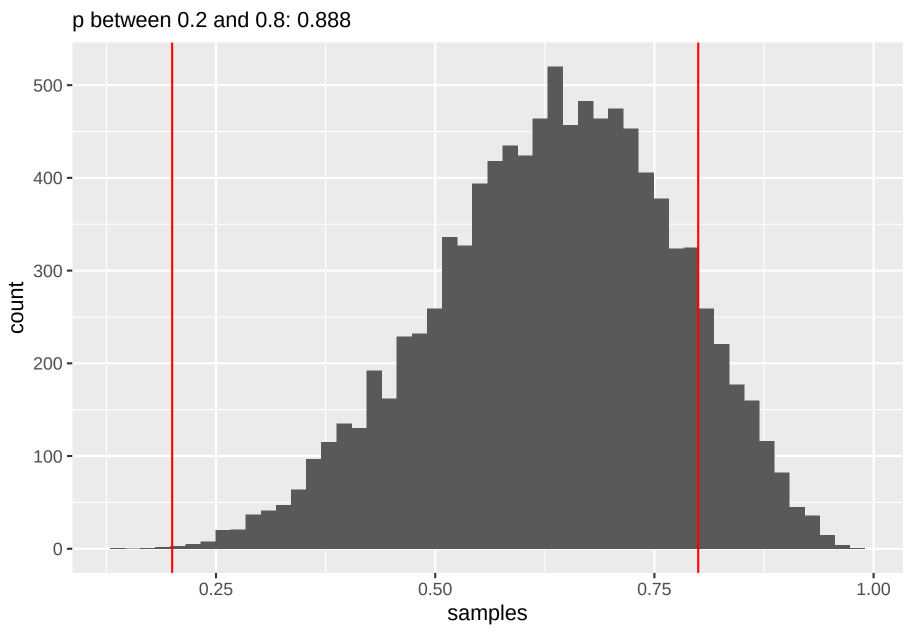
```r
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = 0.8, color="red") +
  labs(subtitle = glue::glue("p below 0.8: {mean(samples < 0.8)}"))
```



3E3

How much posterior probability lies between p = 0.2 and p = 0.8?

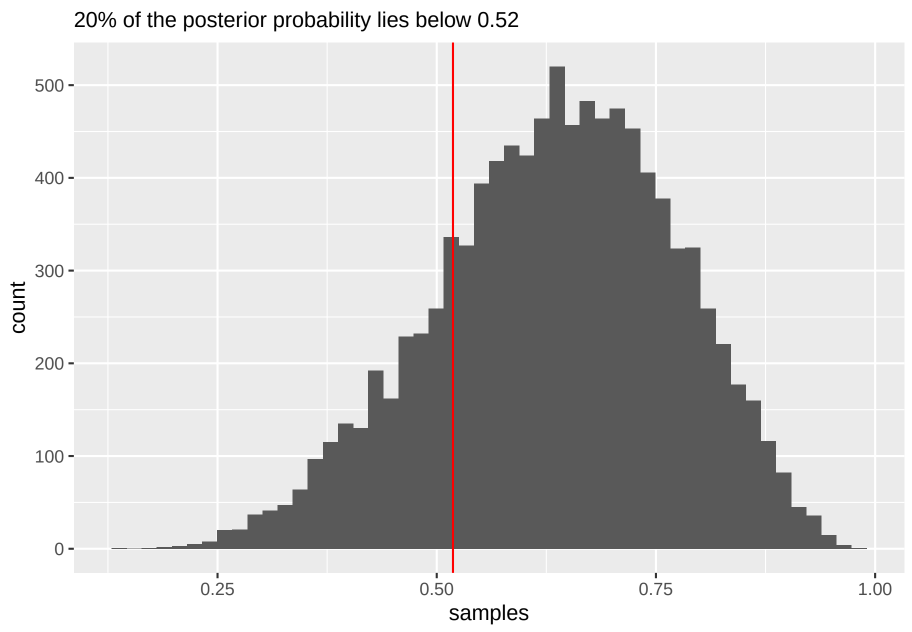
```
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = c(0.2, 0.8), color="red") +
  labs(subtitle = glue::glue("p between 0.2 and 0.8: {mean(samples < 0.8 & samples > 0.2)}"))
```



3E4

20% of the posterior probability lies below which value of p?

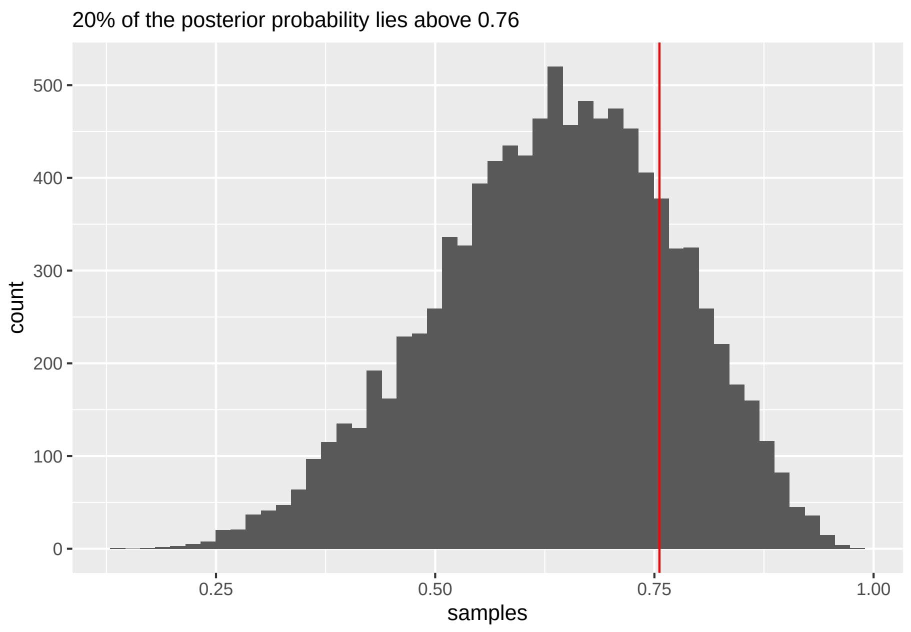
```
q20 <- quantile(samples, 0.2)
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = q20, color="red") +
  labs(subtitle = glue::glue("20% of the posterior probability lies below {round(q20, 2
```



3E5

20% of the posterior probability lies above which value of p?

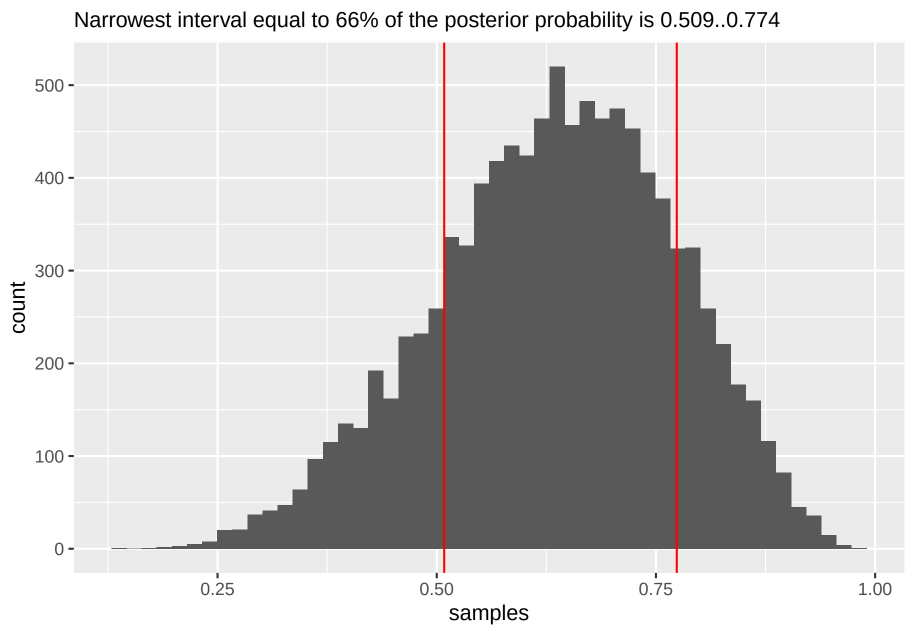
```
q80 <- quantile(samples, 0.8)
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = q80, color="red") +
  labs(subtitle = glue::glue("20% of the posterior probability lies above {round(q80, 2)}"))
```



3E6

Which values of p contain the narrowest interval equal to 66% of the posterior probability?

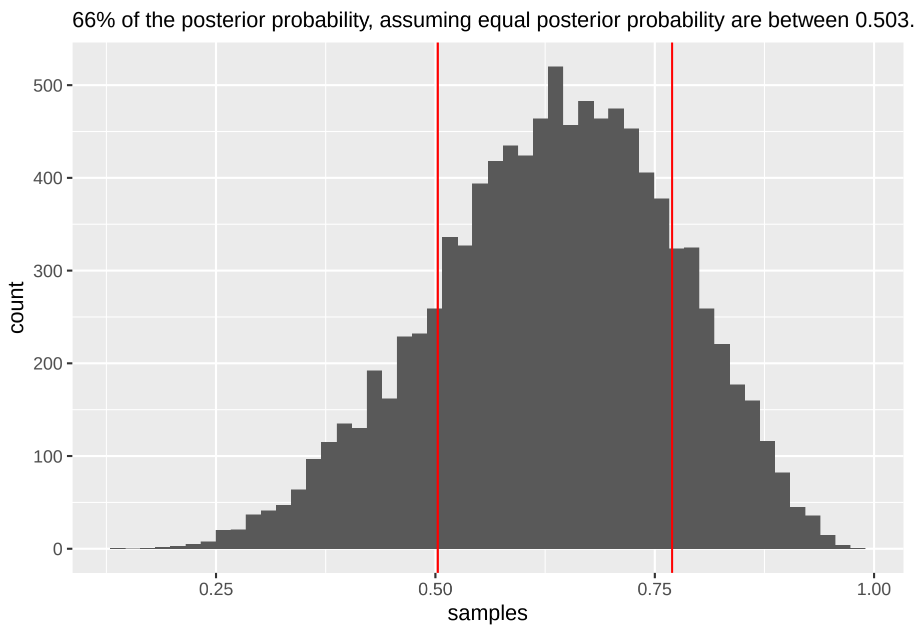
```r
hpdi66 <- HPDI(samples, 0.66)
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = hpdi66, color="red") +
  labs(subtitle = glue::glue("Narrowest interval equal to 66% of the posterior probabi
```



3E7

Which values of p contain 66% of the posterior probability, assuming equal posterior probability both below and above the interval?

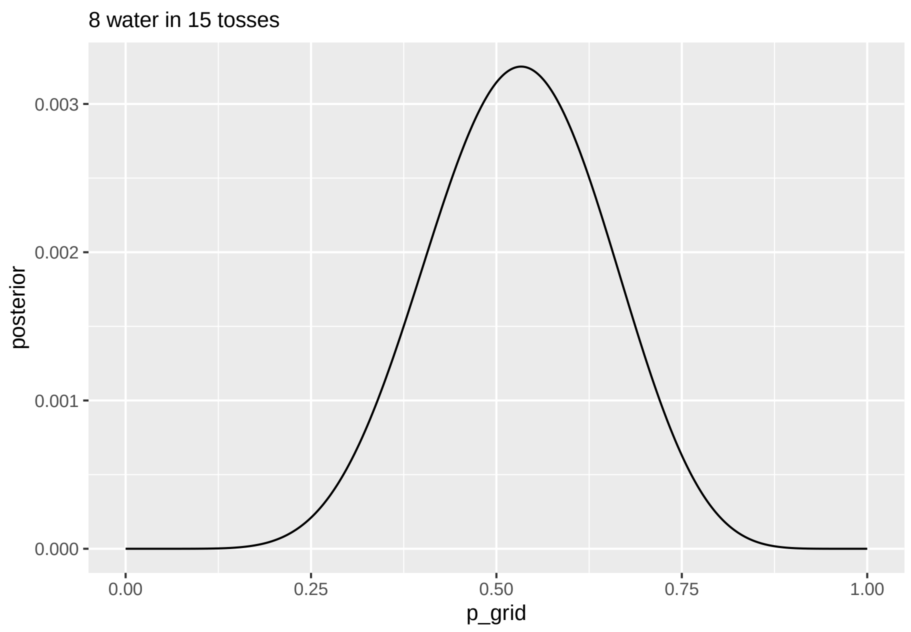
```
pi66 <- PI(samples, 0.66)
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = pi66, color="red") +
  labs(subtitle = glue::glue("66% of the posterior probability, assuming equal posterior probabil
```



3M1

Suppose the globe tossing data had turned out to be 8 water in 15 tosses. Construct the posterior distribution, using grid approximation. Use the same flat prior as before.

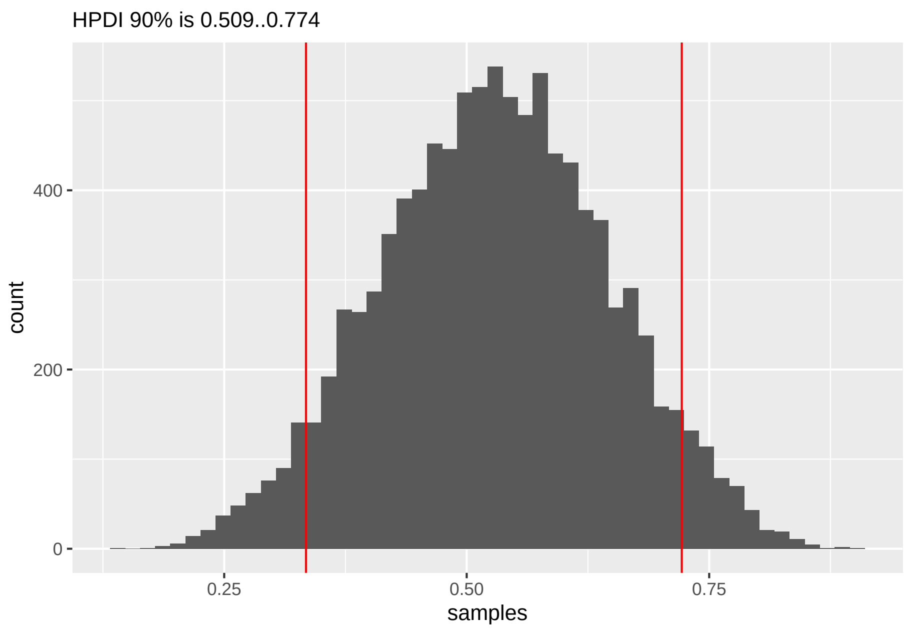
```
p_grid <- seq(from=0, to=1, length.out=1000)
prior <- rep(1,1000)
likelihood <- dbinom(8, size=15, prob=p_grid)
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

ggplot(data=NULL, aes(x=p_grid, y=posterior)) +
  geom_line() +
  labs(subtitle="8 water in 15 tosses")
```



3M2

Draw 10,000 samples from the grid approximation from above. Then use the samples to calculate the 90% HPDI for p.

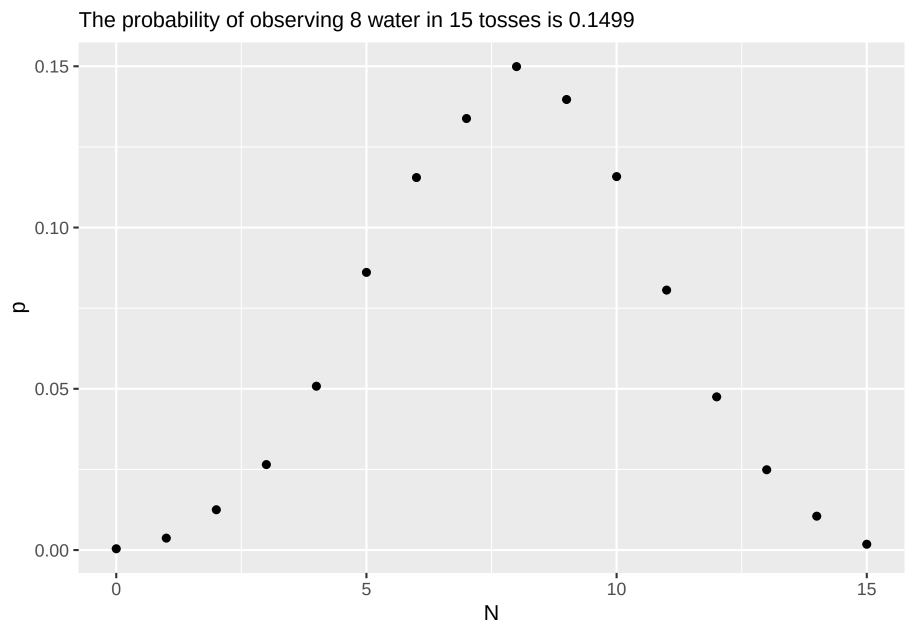
```r
set.seed(100)
samples <- sample(p_grid, prob=posterior, size=1e4, replace=TRUE)

hpdi90 <- HPDI(samples, 0.9)
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = hpdi90, color="red") +
  labs(subtitle = glue::glue("HPDI 90% is {round(hpdi66[1], 3)}..{round(hpdi66[2], 3)}"))
```



3M3

Construct a posterior predictive check for this model and data. This means simulate the distribution of samples, averaging over the posterior uncertainty in p . What is the probability of observing 8 water in 15 tosses?

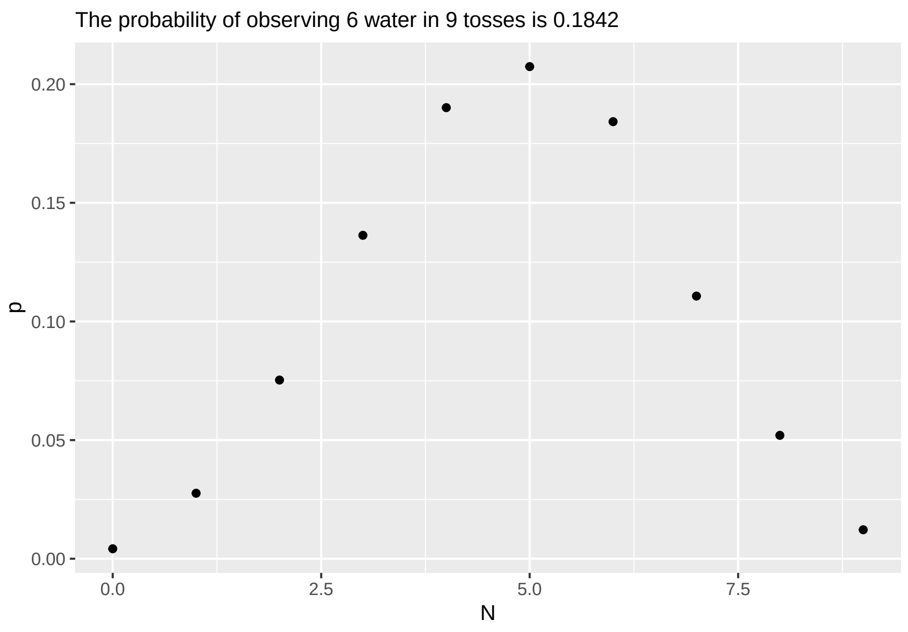
```
predicted_water8_15 <-
  tibble(N = rbinom(length(samples), size=15, prob=samples)) %>%
  group_by(N) %>%
  summarize(count = n(), .groups = 'drop')  %>%
  ungroup() %>%
  mutate(p = count / sum(count))

p8 <-
  predicted_water8_15 %>%
  filter(N == 8) %>%
  pull(p)

ggplot(predicted_water8_15, aes(x=N, y= p)) +
    geom_point() +
  labs(subtitle = glue::glue("The probability of observing 8 water in 15 tosses is {p8
```



3M4

Using the posterior distribution constructed from the new (8/15) data, now calculate the probability of observing 6 water in 9 tosses.

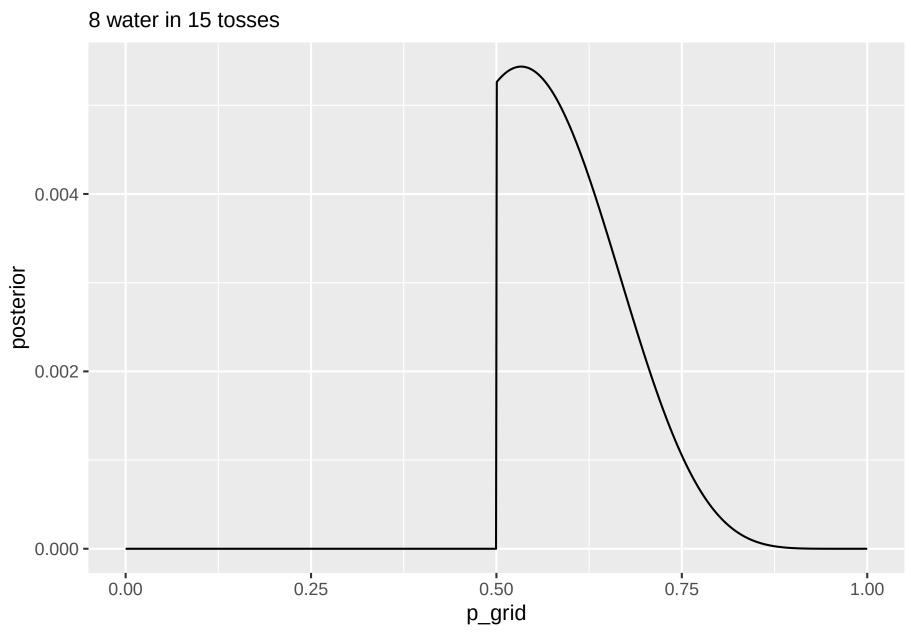
```
predicted_water6_9 <-
  tibble(N = rbinom(length(samples), size=9, prob=samples)) %>%
  group_by(N) %>%
  summarize(count = n(), .groups = 'drop')  %>%
  ungroup() %>%
  mutate(p = count / sum(count))

p6 <-
  predicted_water6_9 %>%
  filter(N == 6) %>%
  pull(p)

ggplot(predicted_water6_9, aes(x=N, y= p)) +
    geom_point() +
  labs(subtitle = glue::glue("The probability of observing 6 water in 9 tosses is {p6}"))
```



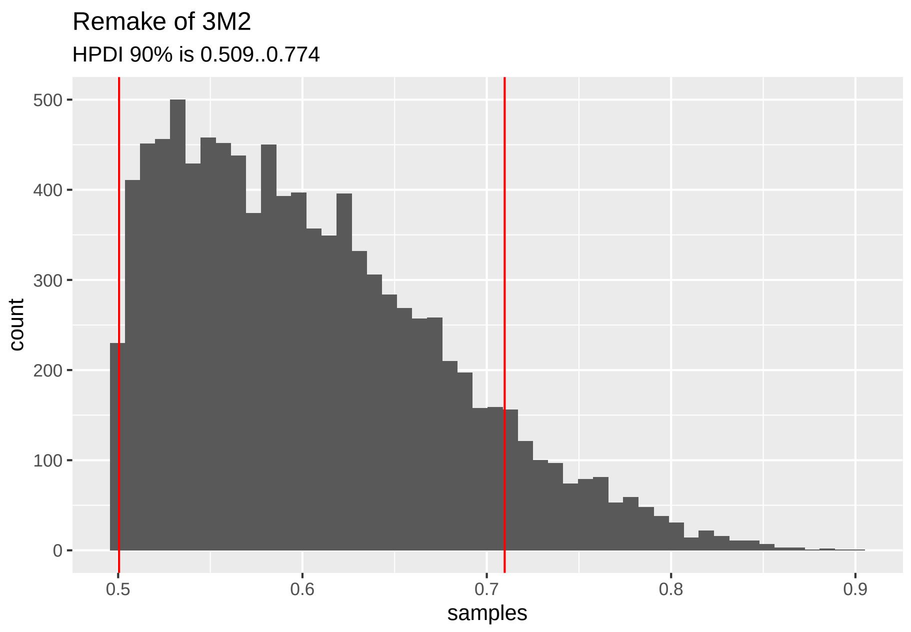
3M5

Start over at 3M1 , but now use a prior that is zero below p = 0.5 and a constant above p = 0.5. This corresponds to prior information that a majority of the Earth's surface is water. Repeat each problem above and compare the inferences. What difference does the better prior make? If it helps, compare inferences (using both priors) to the true value p = 0.7.

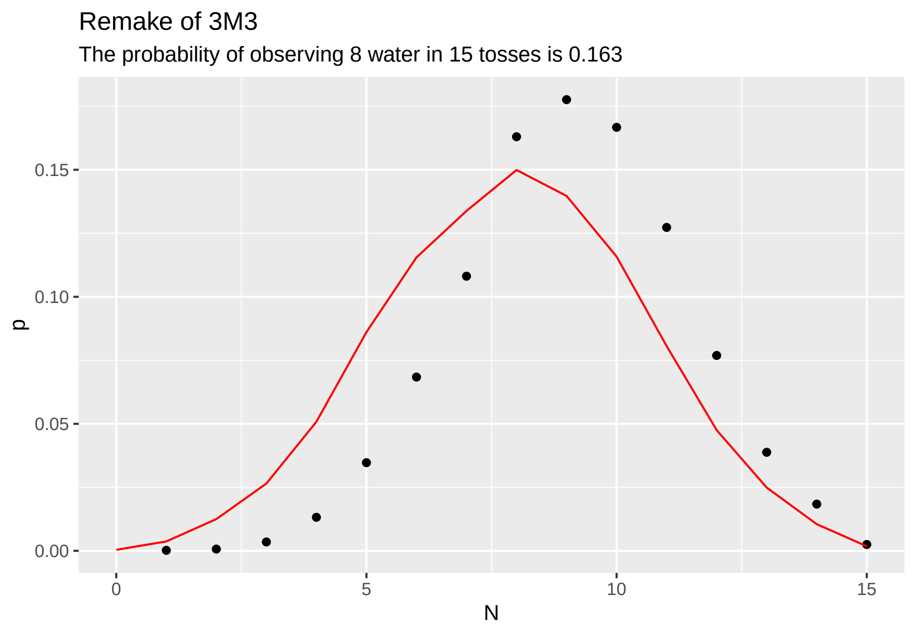
```r
p_grid <- seq(from=0, to=1, length.out=1000)
prior <- as.numeric(p_grid >= 0.5)
likelihood <- dbinom(8, size=15, prob=p_grid)
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

ggplot(data=NULL, aes(x=p_grid, y=posterior)) +
  geom_line() +
  labs(subtitle="8 water in 15 tosses")
```



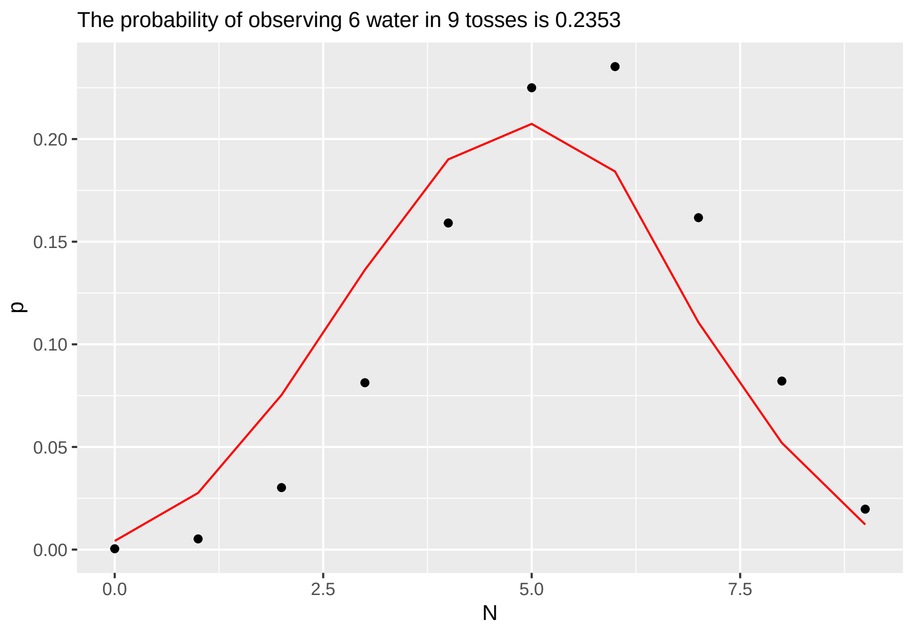
```
set.seed(100)
samples <- sample(p_grid, prob=posterior, size=1e4, replace=TRUE)

hpdi90 <- HPDI(samples, 0.9)
ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=50) +
  geom_vline(xintercept = hpdi90, color="red") +
  labs(title = "Remake of 3M2",
       subtitle = glue::glue("HPDI 90% is {round(hpdi66[1], 3)}..{round(hpdi66[2], 3)}"))
```



```
predicted_water8_15_remake <-
  tibble(N = rbinom(length(samples), size=15, prob=samples)) %>%
  group_by(N) %>%
  summarize(count = n(), .groups = 'drop')  %>%
  ungroup() %>%
  mutate(p = count / sum(count))

p8 <-
  predicted_water8_15_remake %>%
  filter(N == 8) %>%
  pull(p)

ggplot(predicted_water8_15_remake, aes(x=N, y= p)) +
  geom_point() +
  geom_line(data=predicted_water8_15, color="red") +
  labs(title = "Remake of 3M3",
       subtitle = glue::glue("The probability of observing 8 water in 15 tosses is {p8
```



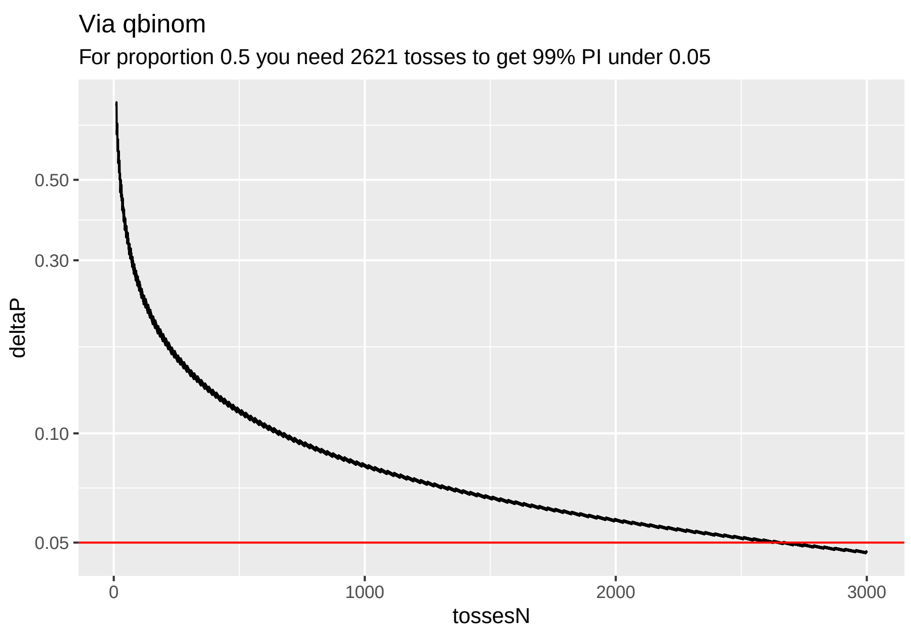
```
predicted_water6_9_remake <-
  tibble(N = rbinom(length(samples), size=9, prob=samples)) %>%
  group_by(N) %>%
  summarize(count = n(), .groups = 'drop')  %>%
  ungroup() %>%
  mutate(p = count / sum(count))

p6 <-
  predicted_water6_9_remake %>%
  filter(N == 6) %>%
  pull(p)

ggplot(predicted_water6_9_remake, aes(x=N, y= p)) +
  geom_point() +
  geom_line(data=predicted_water6_9, color="red") +
  labs(subtitle = glue::glue("The probability of observing 6 water in 9 tosses is {p6}"))
```



3M6

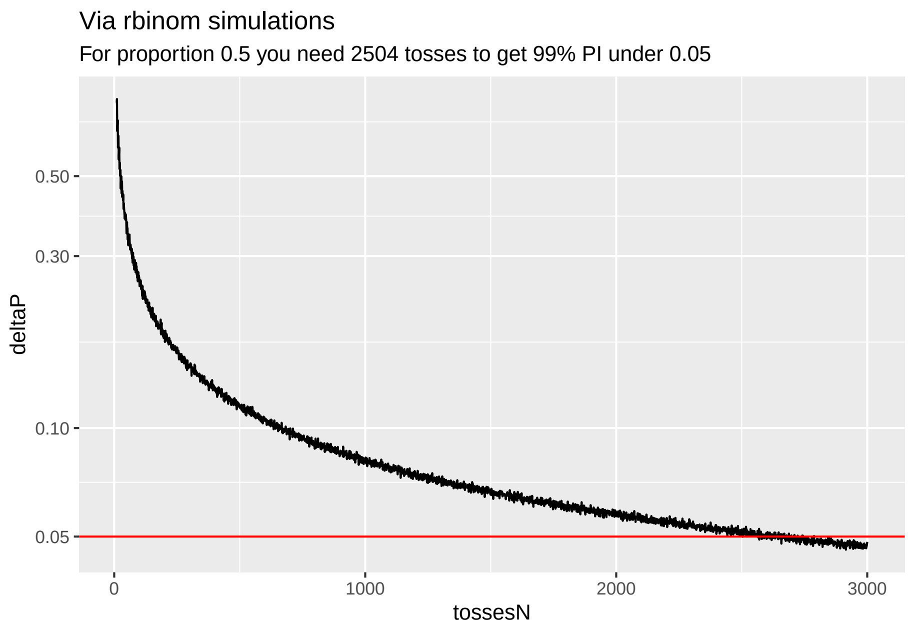
Suppose you want to estimate the Earth's proportion of water very precisely. Specifically, you want the 99% percentile interval of the posterior distribution of p to be only 0.05 wide. This means the distance between the upper and lower bound of the interval should be 0.05. How many times will you have to toss the globe to do this?

```
true_proportion <- 0.5
toss_quantiles <-
  tibble(tossesN = seq(10, 3000, by=1)) %>%
  mutate(lowerN = qbinom(0.005, tossesN, true_proportion),
         upperN = qbinom(0.995, tossesN, true_proportion),
         lowerP = lowerN / tossesN,
         upperP = upperN / tossesN,
         deltaP = upperP - lowerP)

minimal_N <-
  toss_quantiles %>%
  filter(deltaP < 0.05) %>%
  slice(1) %>%
  pull(tossesN)

ggplot(toss_quantiles, aes(x=tossesN, y=deltaP)) +
  geom_line() +
  geom_hline(yintercept = 0.05, color="red") +
  scale_y_log10() +
  labs(title = "Via qbinom",
       subtitle = glue::glue("For proportion {true_proportion} you need {minimal_N} tosses to get
```



```
simulate_and_compute_PI99 <- function(tossN, true_proportion){
  simulated_N <- rbinom(1e4, tossN, true_proportion)
  simulated_P <- simulated_N / tossN
  diff(rethinking::PI(simulated_P, prob=0.99))
}

true_proportion <- 0.5
toss_sims <-
  tibble(tossesN = seq(10, 3000, by=1)) %>%
  rowwise() %>%
  mutate(deltaP =simulate_and_compute_PI99(tossesN, true_proportion))

minimal_N <-
  toss_sims %>%
  filter(deltaP < 0.05) %>%
  slice(1) %>%
  pull(tossesN)
```

```
ggplot(toss_sims, aes(x=tossesN, y=deltaP)) +
  geom_line() +
  geom_hline(yintercept = 0.05, color="red") +
  scale_y_log10() +
  labs(title = "Via rbinom simulations",
       subtitle = glue::glue("For proportion {true_proportion} you need {minimal_N} tos
```



Hard

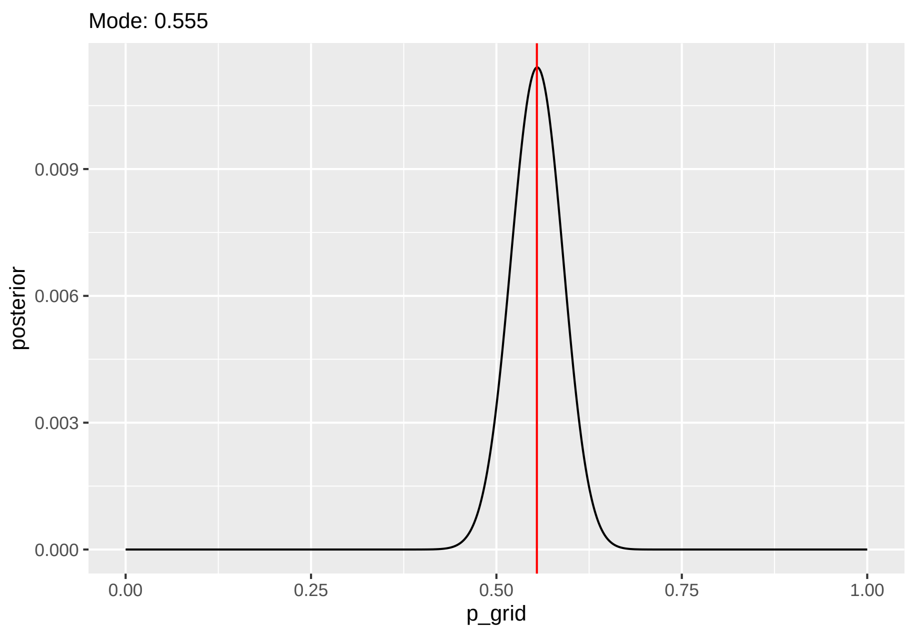
```
data(homeworkch3)
```

3H1

Using grid approximation, compute the posterior distribution for the probability of a birth being a boy. Assume a uniform prior probability. Which parameter value maximizes the posterior probability?

```
birth <- c(birth1, birth2)
birthN <- length(birth)
boyN <- sum(birth)
```

```
p_grid <- seq(from=0, to=1, length.out=1000)
prior <- rep(1,1000)
likelihood <- dbinom(boyN, size=birthN, prob=p_grid)
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

boyP <- p_grid[which.max(posterior)]

ggplot(data=NULL, aes(x=p_grid, y=posterior)) +
  geom_line()+
  geom_vline(xintercept = boyP, color="red") +
  labs(subtitle = glue::glue("Mode: {round(boyP, 3)}"))
```

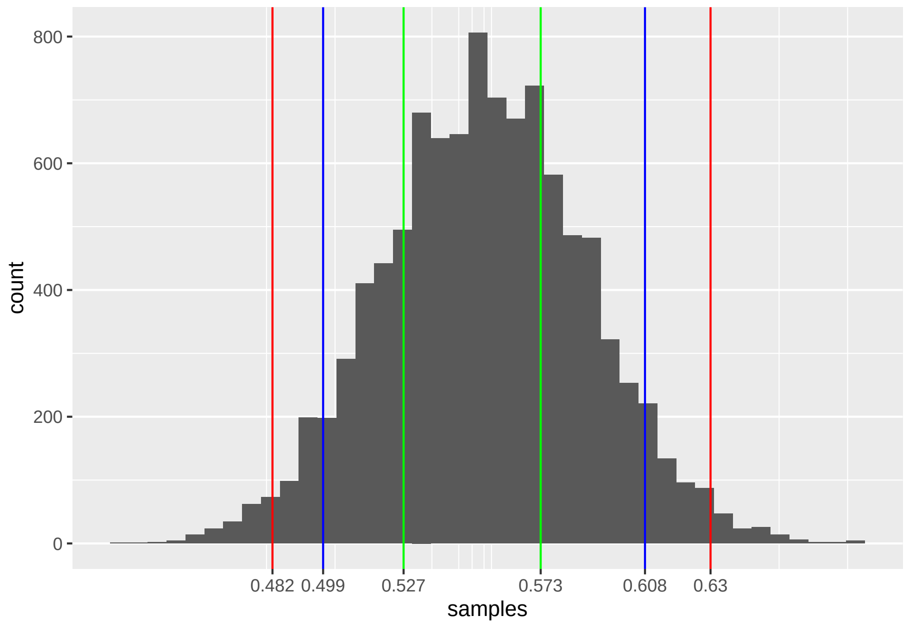


3H2

Using the sample function, draw 10,000 random parameter values from the posterior distribution you calculated above. Use these samples to estimate the 50%, 89%, and 97% highest posterior density intervals.

```
set.seed(100)
samples <- sample(p_grid, prob=posterior, size=1e4, replace=TRUE)

HPDI50 <- rethinking::HPDI(samples, prob = 0.5)
```

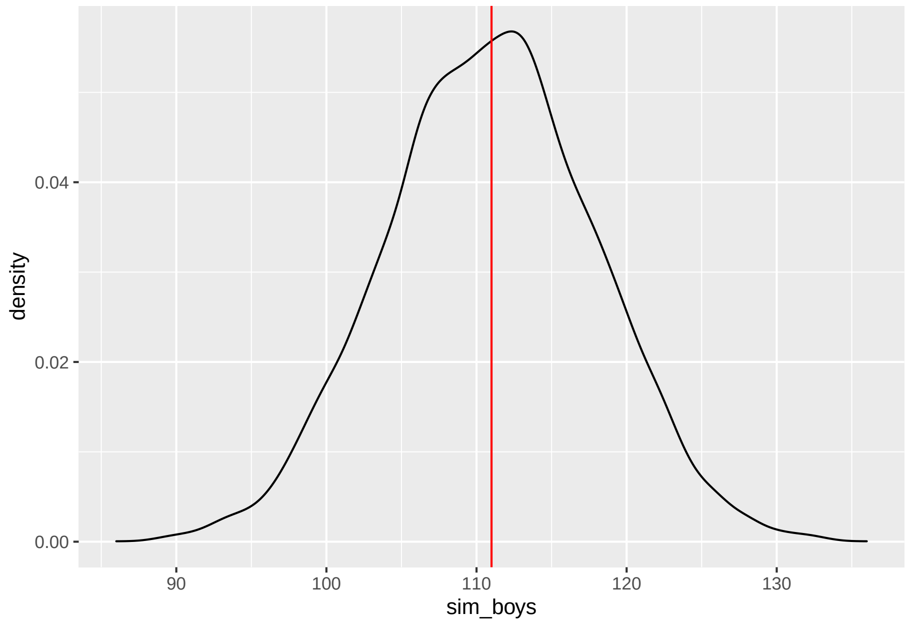
```
HPDI89 <- rethinking::HPDI(samples, prob = 0.89)
HPDI97 <- rethinking::HPDI(samples, prob = 0.97)

all_HPDI <- c(HPDI50, HPDI89, HPDI97)
names(all_HPDI) <- NULL


ggplot(data=NULL, aes(x=samples)) +
  geom_histogram(bins=40) +
  geom_vline(xintercept = HPDI50, color="green") +
  geom_vline(xintercept = HPDI89, color="blue") +
  geom_vline(xintercept = HPDI97, color="red") +
  scale_x_continuous(breaks = all_HPDI, labels = round(all_HPDI, 3))
```



3H3

Use rbinom to simulate 10,000 replicates of 200 births. You should end up with 10,000 numbers, each one a count of boys out of 200 births. Compare the distribution of predicted numbers of boys to the actual count in the data (111 boys out of 200 births). There are many good ways to visualize the simulations, but the dens command (part of the rethinking package) is probably the easiest way in this case. Does it look like the model fits the data well? That is, does the distribution of predictions include the actual observation as a central, likely outcome?

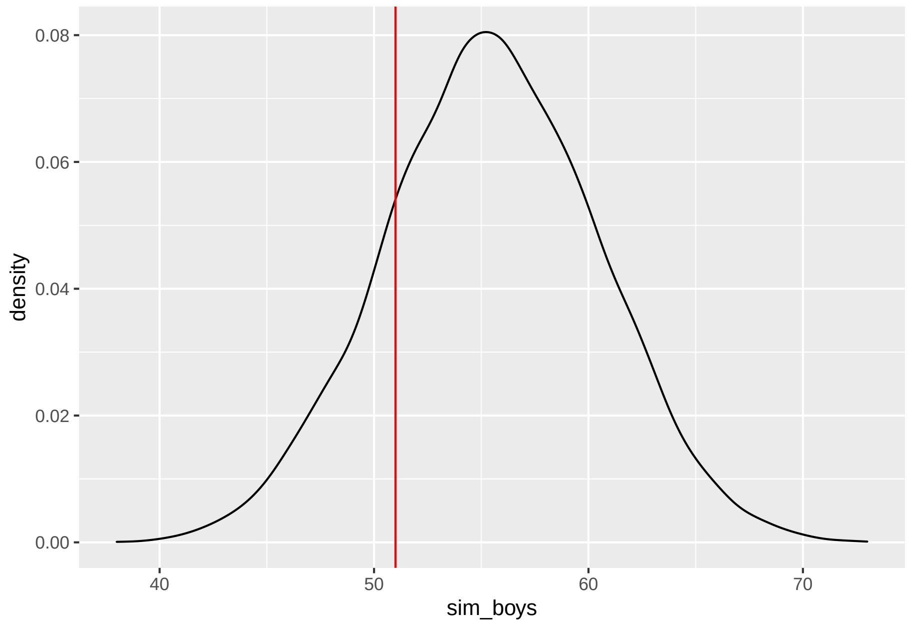
```
sim_boys <- rbinom(1e4, 200, boyN / birthN)

ggplot(data=NULL, aes(x=sim_boys)) +
  geom_density() +
  geom_vline(xintercept = boyN, color="red")
```



3H4

Now compare 10,000 counts of boys from 100 simulated first borns only to the number of boys in the first births, birth1. How does the model look in this light?

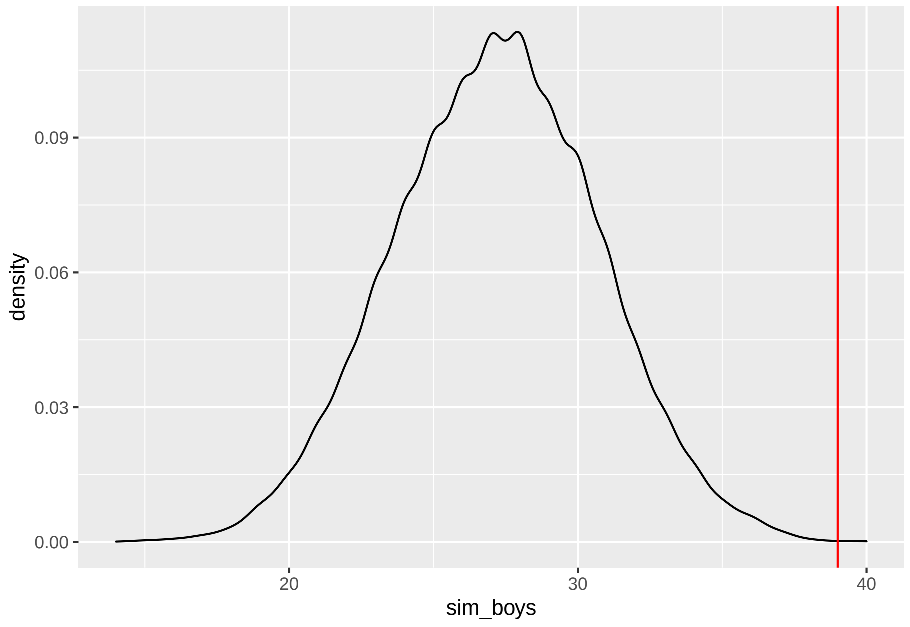
```
sim_boys <- rbinom(1e4, 100, boyN / birthN)

ggplot(data=NULL, aes(x=sim_boys)) +
  geom_density() +
  geom_vline(xintercept = sum(birth1), color="red")
```



3H5

The model assumes that sex of first and second births are independent. To check this assumption, focus now on second births that followed female first borns. Compare 10,000 simulated counts of boys to only those second births that followed girls. To do this correctly, you need to count the number of first borns who were girls and simulate that many births, 10,000 times. Compare the counts of boys in your simulations to the actual observed count of boys following girls. How does the model look in this light? Any guesses what is going on in these data?

```r
birth_after_girl <- birth2[birth1 == 0]
sim_boys <- rbinom(1e4, length(birth_after_girl), boyN / birthN)

ggplot(data=NULL, aes(x=sim_boys)) +
  geom_density() +
  geom_vline(xintercept = sum(birth_after_girl), color="red")
```



0.09
0.06
0.03
0.00
density
20
30
40
sim_boys

Chapter 6

Solutions for Chapter 4

4E1

In the model definition below, which line is the likelihood?

$$\rightarrow\ y_i \sim Normal(\mu, \sigma)\ \leftarrow \mu \sim Normal(0, 10)\sigma \sim Exponential(1)$$

4E2

In the model definition just above, how many parameters are in the posterior distribution?

**Two: μ and σ.

4E3

Using the model definition above, write down the appropriate form of Bayes' theorem that includes the proper likelihood and priors.

$$Pr(\mu, \sigma|y) = \frac{\prod_i Normal(y_i|\mu, \sigma) \cdot Normal(mu|0, 10) \cdot Exponential(\sigma|1)}{\int \int \prod_i Normal(y_i|\mu, \sigma) \cdot Normal(mu|0, 10) \cdot Exponential(\sigma|1) \partial\mu\partial\sigma}$$

4E4

In the model definition below, which line is the linear model?

$$y_i \sim Normal(\mu, \sigma) \rightarrow\ \mu_i = \alpha + \beta x_i\ \leftarrow \alpha \sim Normal(0, 10)\beta \sim Normal(0, 1)\sigma \sim Exponential(2)$$

4E5

In the model definition just above, how many parameters are in the posterior distribution?

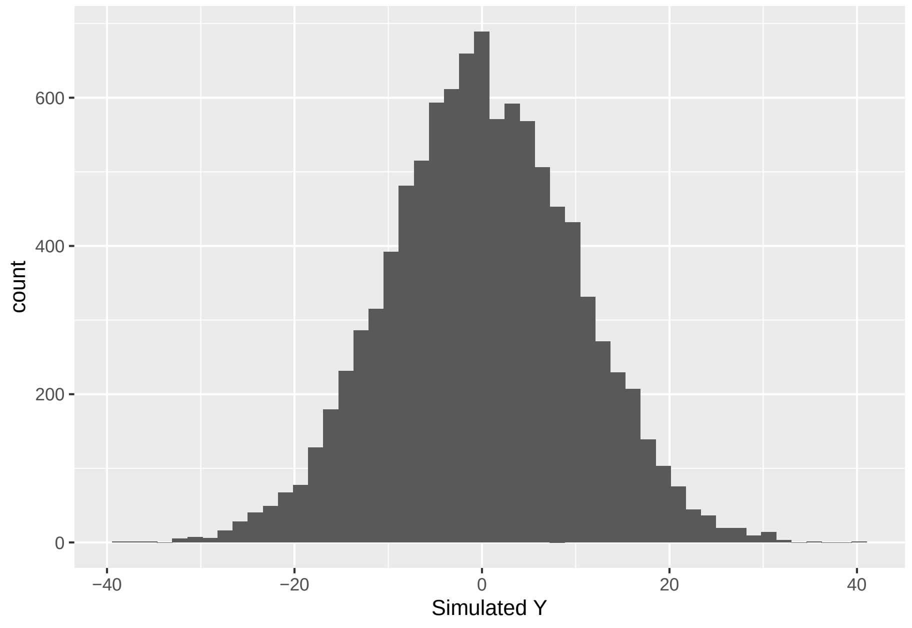
Three, α, β (used to compute μ), and σ.

4M1

For the model definition below, simulate observed y values from the prior (not the posterior).

$$y_i \sim Normal(\mu, \sigma)\mu \sim Normal(0, 10)\sigma \sim Exponential(1)$$

```
library(ggplot2)

samples_N <- 1e4
mu <- rnorm(samples_N, 0, 10)
sigma <- rexp(samples_N, 1)
y_sim <- rnorm(samples_N, mu, sigma)

ggplot(data=NULL, aes(x=y_sim)) +
  geom_histogram(bins=50) +
  xlab("Simulated Y")
```



4M2

Translate the model just above into a quap formula.

```
quap_formula <- alist(
  y ~ dnorm(mu, sigma),
  mu ~ dnorm(0,10),
  sigma ~ dexp(1))
```

4M3

Translate the quap model formula below into a mathematical model definition.

```
y ~ dnorm( mu , sigma ),
mu <- a + b*x,
a ~ dnorm( 0 , 10 ),
b ~ dunif( 0 , 1 ),
sigma ~ dexp( 1 )
```

$$y_i \sim Normal(\mu, \sigma)\mu_i = \alpha + \beta x_i \alpha \sim Normal(0, 10)\beta \sim Uniform(0, 1)\sigma \sim Exponential(1)$$

4M4

A sample of students is measured for height each year for 3 years. After the third year, you want to fit a linear regression predicting height using year as a predictor. Write down the mathematical model definition for this regression, using any variable names and priors you choose. Be prepared to defend your choice of priors.

I assume *students* as in *college students*. Therefore, you would expect an adult height, may be slightly lower, hence `170` for the intercept. You don't expect them to grow much, hence conservative slope for the effect of the year. I assume a fairly broad but strictly positive prior for σ.

$$h_i \sim Normal(\mu, \sigma)\mu_i = \alpha + \beta year_i \alpha \sim Normal(170, 10)\beta \sim Normal(0, 1)\sigma \sim Exponential(0.1)$$

4M5

Now suppose I remind you that every student got taller each year. Does this information lead you to change your choice of priors? How?

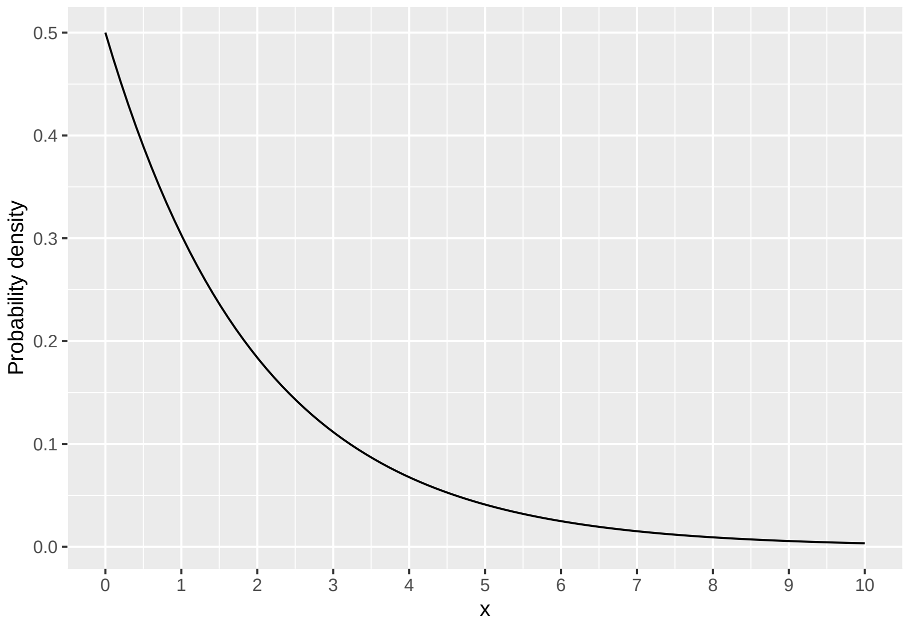
I would ensure that β can be only positive, so $\beta \sim Exponential(1)$.

4M6

Now suppose I tell you that the variance among heights for students of the same age is never more than 64cm. How does this lead you to revise your priors?

Based on the density plot, I would pick a more conservative prior of $\sigma \sim Exponential(0.5)$ that drops off more rapidly and just barely includes $\sqrt(64)$.

```
x <- seq(0, 10, length.out=100)
ggplot(data=NULL, aes(x=x, y= dexp(x, 0.5))) +
  geom_line() +
  scale_x_continuous(breaks = 0:10) +
  ylab("Probability density")
```



4M7

Refit model m4.3 from the chapter, but omit the mean weight xbar this time. Compare the new model's posterior to that of the original model. In particular, look at the covariance among the parameters. What is different? Then compare the posterior predictions of both models.

```
# load data again, since it's a long way back
library(rethinking)
data(Howell1);
```

```
d <- Howell1;
d2 <- d[d$age >= 18,]

# define the average weight, x-bar
xbar <- mean(d2$weight)
# fit model

m4_3c <- quap(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + b * (weight - xbar),
    a ~ dnorm(178, 20) ,
    b ~ dlnorm(0, 1) ,
    sigma ~ dunif(0, 50)) ,
  data=d2)

m4_3 <- quap(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + b * weight,
    a ~ dnorm(178, 20) ,
    b ~ dlnorm(0, 1) ,
    sigma ~ dunif(0, 50)) ,
  data=d2)

precis(m4_3)
```

```
##              mean         sd        5.5%       94.5%
## a     114.5342008 1.89774719 111.5012343 117.5671674
## b       0.8907326 0.04175799   0.8239952   0.9574699
## sigma   5.0727190 0.19124895   4.7670662   5.3783718
```

```
precis(m4_3c)
```

```
##              mean         sd        5.5%      94.5%
## a     154.6013683 0.27030768 154.1693645 155.033372
## b       0.9032809 0.04192363   0.8362788   0.970283
## sigma   5.0718813 0.19115482   4.7663790   5.377384
```

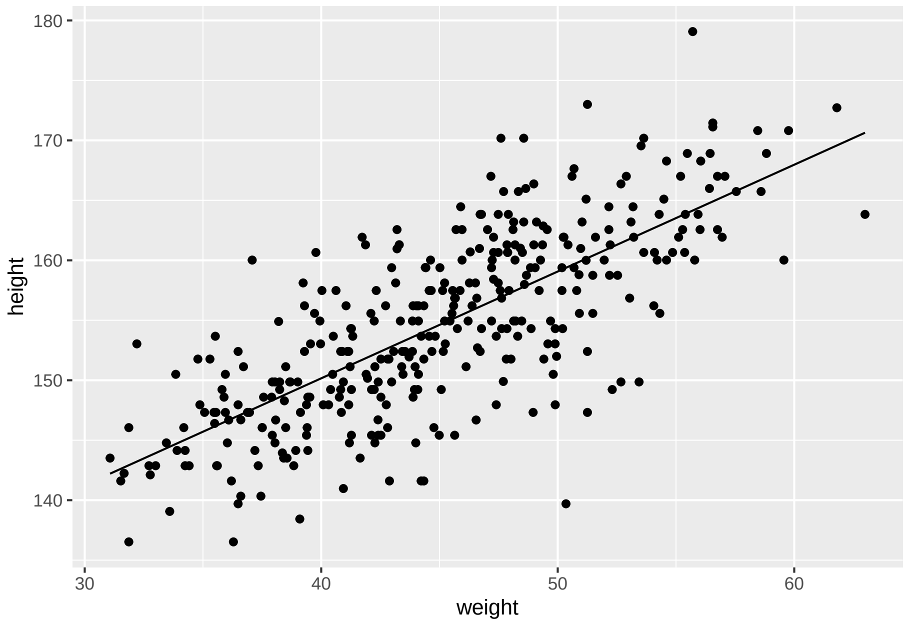
```
round(vcov(m4_3), 3)
```

```
##            a      b sigma
## a      3.601 -0.078 0.009
## b     -0.078  0.002 0.000
## sigma  0.009  0.000 0.037
```

```
round(vcov(m4_3c), 3)
```

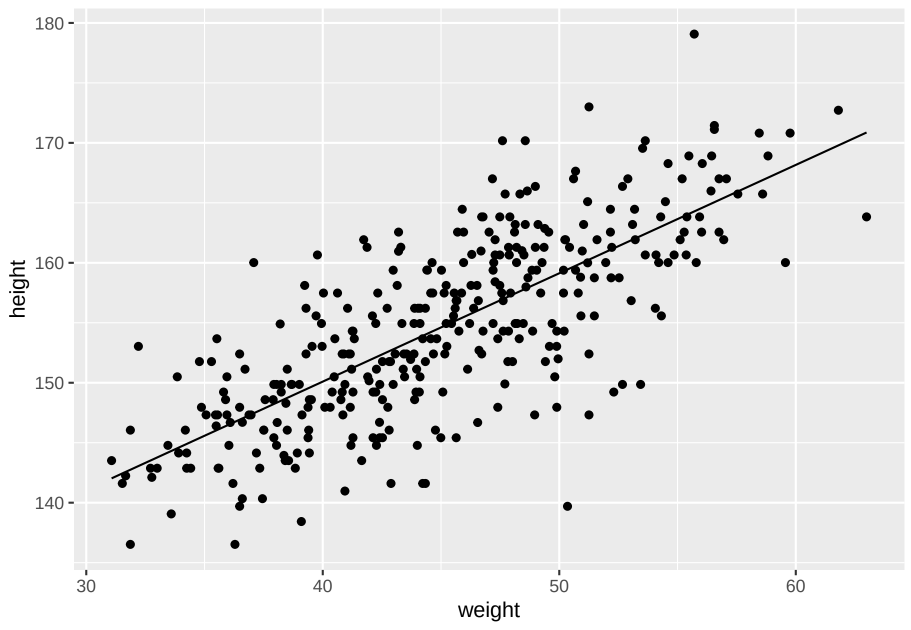
```
##           a     b sigma
## a     0.073 0.000 0.000
## b     0.000 0.002 0.000
## sigma 0.000 0.000 0.037
```

```
post <- extract.samples(m4_3)
a_map <- mean(post$a)
b_map <- mean(post$b)

w <- d2$weight
prediction <- a_map + b_map*w

ggplot(data = d2, aes(x=weight, y=height)) +
  geom_point() +
  geom_line(data = NULL, aes(x=w, y=prediction))
```



```
post <- extract.samples(m4_3c)
a_map <- mean(post$a)
b_map <- mean(post$b)

w <- d2$weight
```

```
prediction <- a_map + b_map*(w - xbar)

ggplot(data = d2, aes(x=weight, y=height)) +
  geom_point() +
  geom_line(data = NULL, aes(x=w, y=prediction))
```

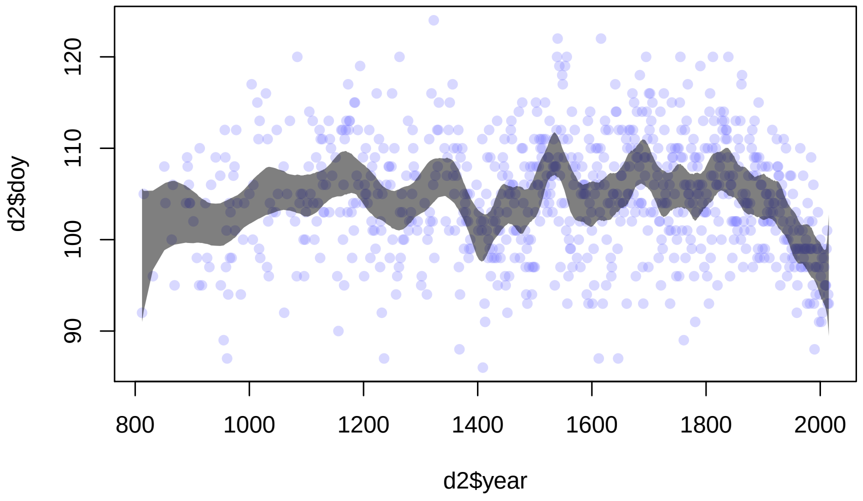


4M8

In the chapter, we used 15 knots with the cherry blossom spline. Increase the number of knots and observe what happens to the resulting spline. Then adjust also the width of the prior on the weights—change the standard deviation of the prior and watch what happens. What do you think the combination of knot number and the prior on the weights controls?

```
data(cherry_blossoms)
d <- cherry_blossoms
d2 <- d[ complete.cases(d$doy) , ] # complete cases on doy
num_knots <- 30
knot_list <- quantile( d2$year , probs=seq(0,1,length.out=num_knots) )
B <- bs(d2$year,
        knots=knot_list[-c(1,num_knots)],
        degree=3 , intercept=TRUE)
```

```
m4.7_30 <- quap(
  alist(
    D ~ dnorm( mu , sigma ) ,
    mu <- a + B %*% w ,
    a ~ dnorm(100,10),
    w ~ dnorm(0,10),
    sigma ~ dexp(1)),
  data=list(D=d2$doy, B=B) ,
  start=list(w=rep(0, ncol(B))))

mu <- link( m4.7_30 )
mu_PI <- apply(mu,2, PI, 0.97)
plot( d2$year , d2$doy , col=col.alpha(rangi2,0.3) , pch=16 )
shade( mu_PI , d2$year , col=col.alpha("black",0.5) )
```



Together, they control just how wiggly vs. smooth the resultant curve is.

4H1

The weights listed below were recorded in the !Kung census, but heights were not recorded for these individuals. Provide predicted heights and 89% intervals for each of these individuals. That is, fill in the table below, using model-based predictions.

```r
predicted_weight <-
  tibble(Individual = 1:5,
         weight = c(46.95, 43.72, 64.78, 32.59, 54.63))

# getting sampled means
posterior_mu <- link(m4_3c, data=predicted_weight)

predicted_weight <- predicted_weight %>%
  mutate(`expected height` = apply(posterior_mu, 2, mean),
         LowerCI = round(apply(posterior_mu, 2, quantile, probs=(1-0.89)/2), 2),
         UpperCI = round(apply(posterior_mu, 2, quantile, probs=1-(1-0.89)/2), 2),
         `89% interval` = glue::glue("{LowerCI}..{UpperCI}"))  %>%
  select(-LowerCI, -UpperCI)

knitr::kable(predicted_weight)
```

Individual	weight	expected height	89% interval
1	46.95	156.3876	155.97..156.83
2	43.72	153.4682	153.05..153.89
3	64.78	172.5032	171.1..173.91
4	32.59	143.4083	142.5..144.37
5	54.63	163.3292	162.57..164.07

4H2

Select out all the rows in the Howell1 data with ages below 18 years of age. If you do it right, you should end up with a new data frame with 192 rows in it.

a. Fit a linear regression to these data, using quap. Present and interpret the estimates. For every 10 units of increase in weight, how much taller does the model predict a child gets?

```r
kung_kids <- filter(Howell1, age < 18)

# define the average weight, x-bar
avg_kid_weight <- mean(kung_kids$weight)

# fit model
kung_kids_fit <- quap(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + b * (weight - avg_kid_weight),
    a ~ dnorm(140, 20) ,
    b ~ dlnorm(0, 1) ,
    sigma ~ dunif(0, 50)) ,
  data=kung_kids)

precis(kung_kids_fit)
```

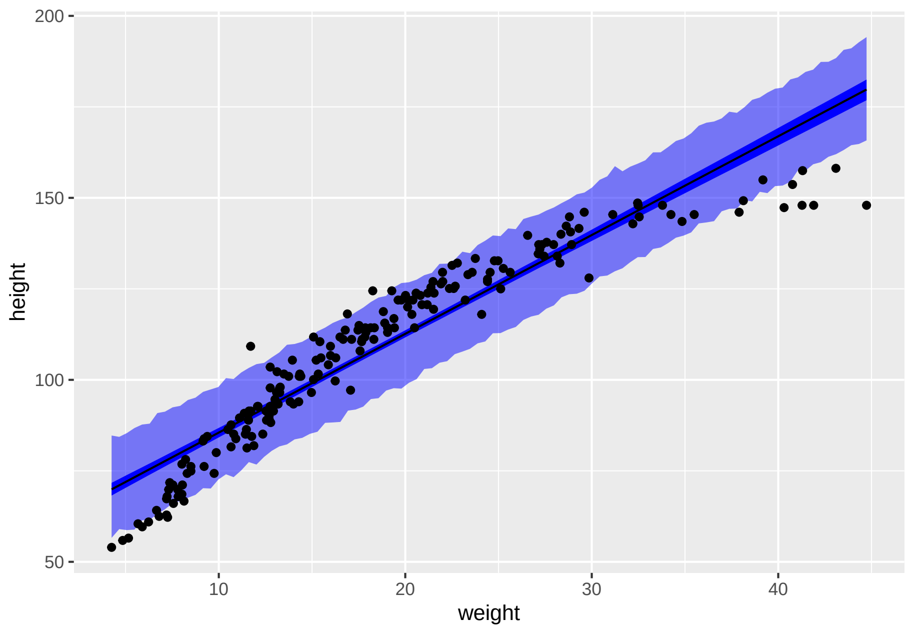
```
##             mean         sd       5.5%      94.5%
## a     108.348193 0.60862837 107.375488 109.320899
## b       2.716657 0.06831574   2.607475   2.825839
## sigma   8.437208 0.43057023   7.749073   9.125342
```

```
b_mean <- precis(kung_kids_fit)['b', 'mean']
```

For every 10 units of increase in weight, the model predict a child will get 27.2 cm taller.

b. Plot the raw data, with height on the vertical axis and weight on the horizontal axis. Superimpose the MAP regression line and 89% interval for the mean. Also superimpose the 89% interval for predicted heights.

```
mean_estimates <- tibble(weight = seq(min(kung_kids$weight), max(kung_kids$weight), ler
sample_means <- link(kung_kids_fit, mean_estimates)
mean_estimates <-
  mean_estimates %>%
  mutate(mu = apply(sample_means, 2, mean),
         lowerCI = apply(sample_means, 2, quantile, probs=(1-0.89)/2),
         upperCI = apply(sample_means, 2, quantile, probs=1-(1-0.89)/2))

simulated_height_samples <- sim(kung_kids_fit, mean_estimates)
simulated_height <-
  mean_estimates %>%
  mutate(lowerCI = apply(simulated_height_samples, 2, quantile, probs=(1-0.89)/2),
         upperCI = apply(simulated_height_samples, 2, quantile, probs=1-(1-0.89)/2))


ggplot(data = kung_kids, aes(x = weight, y=height)) +
  geom_ribbon(data=simulated_height, aes(y=lowerCI, ymin=lowerCI, ymax=upperCI), fill='
  geom_ribbon(data=mean_estimates, aes(y=mu, ymin=lowerCI, ymax=upperCI), fill="blue")+
  geom_point() +
  geom_line(data=mean_estimates, aes(y=mu))
```



c. What aspects of the model fit concern you? Describe the kinds of assumptions you would change, if any, to improve the model. You don't have to write any new code. Just explain what the model appears to be doing a bad job of, and what you hypothesize would be a better model.

The relationship is clearly no linear with rapid growth for smaller children turning into a more slow one for older ones.

4H3

Suppose a colleague of yours, who works on allometry, glances at the practice problems just above. Your colleague exclaims, "That's silly. Everyone knows that it's only the logarithm of body weight that scales with height!" Let's take your colleague's advice and see what happens.

(a) Model the relationship between height (cm) and the natural logarithm of weight (log-kg). Use the entire Howell1 data frame, all 544 rows, adults and non-adults. Can you interpret the resulting estimates?

```
kung <-
  Howell1 %>%
  mutate(logWeight = log(weight))
```

```
# define the average weight, x-bar
avg_log_weight <- mean(kung$logWeight)

# fit model
kung_fit <- quap(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + b * (logWeight - avg_log_weight),
    a ~ dnorm(140, 20) ,
    b ~ dlnorm(0, 1) ,
    sigma ~ dunif(0, 50)) ,
  data=kung)

precis(kung_fit)
```

```
##             mean        sd       5.5%      94.5%
## a     138.263717 0.2201177 137.911927 138.615508
## b      47.072415 0.3826000  46.460947  47.683884
## sigma   5.134294 0.1556374   4.885555   5.383032
```

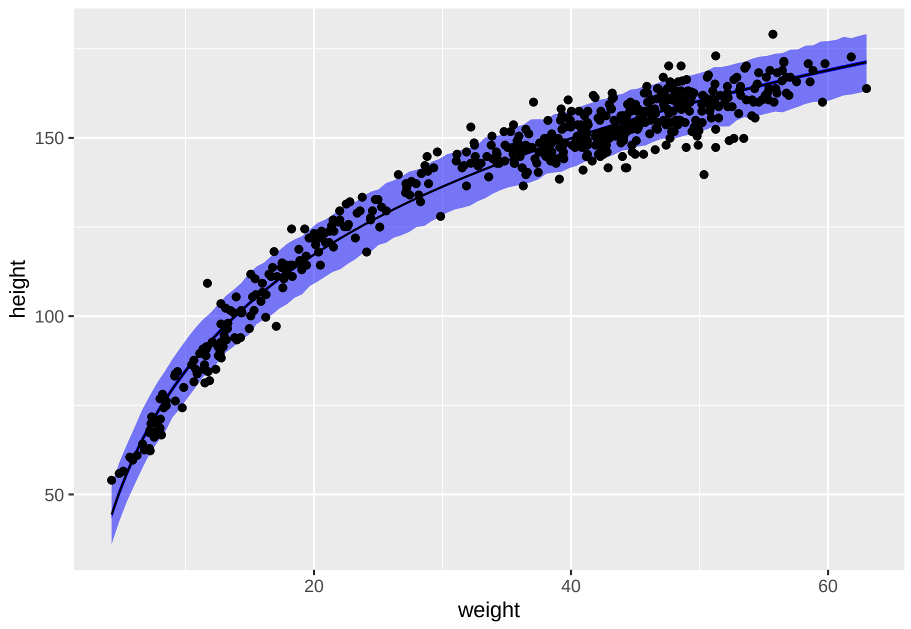
Not really, as `b` now refers to change in height in response to 1 log-Kg of weight.

(b) Begin with this plot: plot(height ~ weight , data=Howell1). Then use samples from the quadratic approximate posterior of the model in (a) to superimpose on the plot:

(1) the predicted mean height as a function of weight,
(2) the 97% interval for the mean, and
(3) the 97% interval for predicted heights.

```
mean_estimates <-
  tibble(weight = seq(min(kung$weight), max(kung$weight), length.out = 100)) %>%
  mutate(logWeight = log(weight))

sample_means <- link(kung_fit, mean_estimates)
mean_estimates <-
  mean_estimates %>%
  mutate(mu = apply(sample_means, 2, mean),
         lowerCI = apply(sample_means, 2, quantile, probs=(1-0.89)/2),
         upperCI = apply(sample_means, 2, quantile, probs=1-(1-0.89)/2))

simulated_height_samples <- sim(kung_fit, mean_estimates)
simulated_height <-
```

```
  mean_estimates %>%
  mutate(lowerCI = apply(simulated_height_samples, 2, quantile, probs=(1-0.89)/2),
         upperCI = apply(simulated_height_samples, 2, quantile, probs=1-(1-0.89)/2))


ggplot(data = kung, aes(x = weight, y=height)) +
  geom_ribbon(data=simulated_height, aes(y=lowerCI, ymin=lowerCI, ymax=upperCI), fill="blue", alp
  geom_ribbon(data=mean_estimates, aes(y=mu, ymin=lowerCI, ymax=upperCI), fill="blue")+
  geom_point() +
  geom_line(data=mean_estimates, aes(y=mu))
```

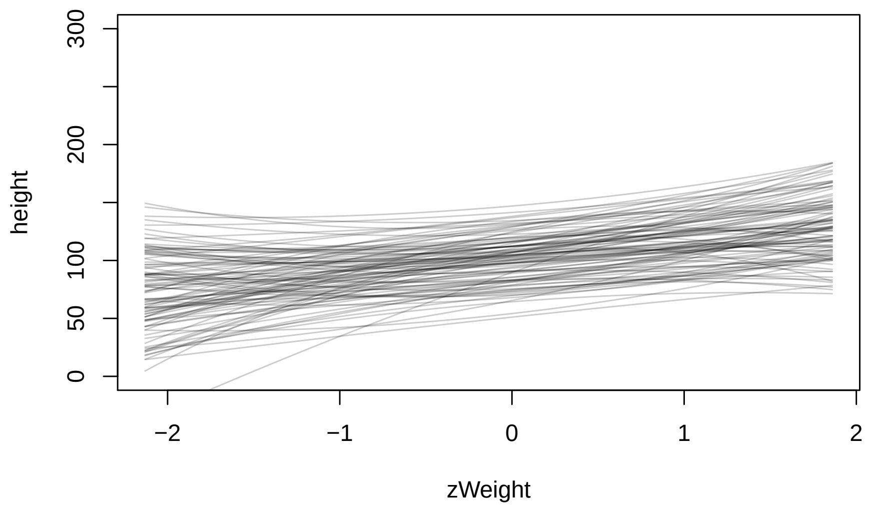


4H4

Plot the prior predictive distribution for the parabolic polynomial regression model in the chapter. You can modify the code that plots the linear regression prior predictive distribution. Can you modify the prior distributions of , 1, and 2 so that the prior predictions stay within the biologically reasonable outcome space? That is to say: Do not try to fit the data by hand. But do try to keep the curves consistent with what you know about height and weight, before seeing these exact data.

```
zWeight <- c(scale(kung$weight))
```

```r
N <- 100
a <- rnorm(N, 100, 20)
b1 <- rlnorm(N, 2.5, 0.5)
b2 <- rnorm(N, 0, 5)
x <- zWeight
x2 <- zWeight^2
plot(NULL, xlim=range(zWeight), ylim=c(0, 300),
    xlab="zWeight", ylab="height")

for (i in 1:N){
  curve( a[i] + b1[i]*x + b2[i]*(x*x),
  from=min(zWeight), to=max(zWeight),
  add=TRUE,
  col=col.alpha("black",0.2))
}
```



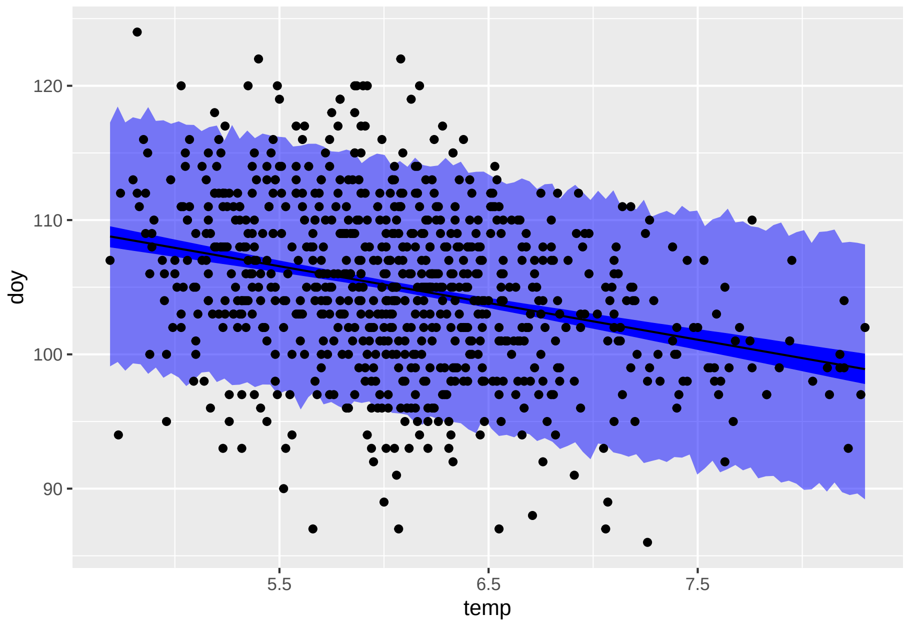
4H5

Return to data(cherry_blossoms) and model the association between blossom date (doy) and March temperature (temp). Note that there are many missing values in both variables. You may consider a linear model, a polynomial, or a spline on temperature. How well does temperature trend predict the blossom trend?

```
sakura <-
  cherry_blossoms %>%
  drop_na(doy, temp)

avg_temp <- mean(sakura$temp)

sakura_fit <- quap(
  alist(
    doy ~ dnorm(mu, sigma),
    mu <- a + b * (temp - avg_temp),
    a ~ dnorm(75, 30), # jan-may is about 150 days, so 75±60 is very conservative spring estimate
    b ~ dnorm(0, 1), # we would assume the relationship to be negative, so higher tempratures lea
    sigma ~ dunif(0, 15)), # ±30 gives us a month
  data=sakura)

precis(sakura_fit)
```

```
##              mean        sd       5.5%      94.5%
## a     104.919743 0.2107589 104.582909 105.256576
## b      -2.733661 0.2951189  -3.205318  -2.262004
## sigma   5.912675 0.1491559   5.674295   6.151055
```

```
mean_estimates <-
  tibble(temp = seq(min(sakura$temp), max(sakura$temp), length.out = 100))

sample_means <- link(sakura_fit, mean_estimates)
mean_estimates <-
  mean_estimates %>%
  mutate(mu = apply(sample_means, 2, mean),
         lowerCI = apply(sample_means, 2, quantile, probs=(1-0.89)/2),
         upperCI = apply(sample_means, 2, quantile, probs=1-(1-0.89)/2))

simulated_doy_samples <- sim(sakura_fit, mean_estimates)
simulated_doy <-
  mean_estimates %>%
  mutate(lowerCI = apply(simulated_doy_samples, 2, quantile, probs=(1-0.89)/2),
         upperCI = apply(simulated_doy_samples, 2, quantile, probs=1-(1-0.89)/2))

ggplot(sakura, aes(x = temp, y=doy)) +
  geom_ribbon(data=simulated_doy, aes(y=lowerCI, ymin=lowerCI, ymax=upperCI), fill="blue", alpha=
  geom_ribbon(data=mean_estimates, aes(y=mu, ymin=lowerCI, ymax=upperCI), fill="blue")+
  geom_point() +
  geom_line(data=mean_estimates, aes(y=mu))
```



Reasonably well, but you can see a lot of spread in the middle, so other factors are clearly at play.

4H6

Simulate the prior predictive distribution for the cherry blossom spline in the chapter. Adjust the prior on the weights and observe what happens. What do you think the prior on the weights is doing?

4H8

The cherry blossom spline in the chapter used an intercept , but technically it doesn't require one. The first basis functions could substitute for the intercept. Try refitting the cherry blossom spline without the intercept. What else about the model do you need to change to make this work?