Notes on and Solutions for Statistical Rethinking

Alexander Pastukhov

2021-05-17

Contents

Chapter 1

Precis

This is a collection of notes that attempt to provide further details and intuition for some topics, such as loss functions, information theory, information criteria, MCMC algorithms, etc. This material is **free to use** and is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives V4.0 International License.

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].

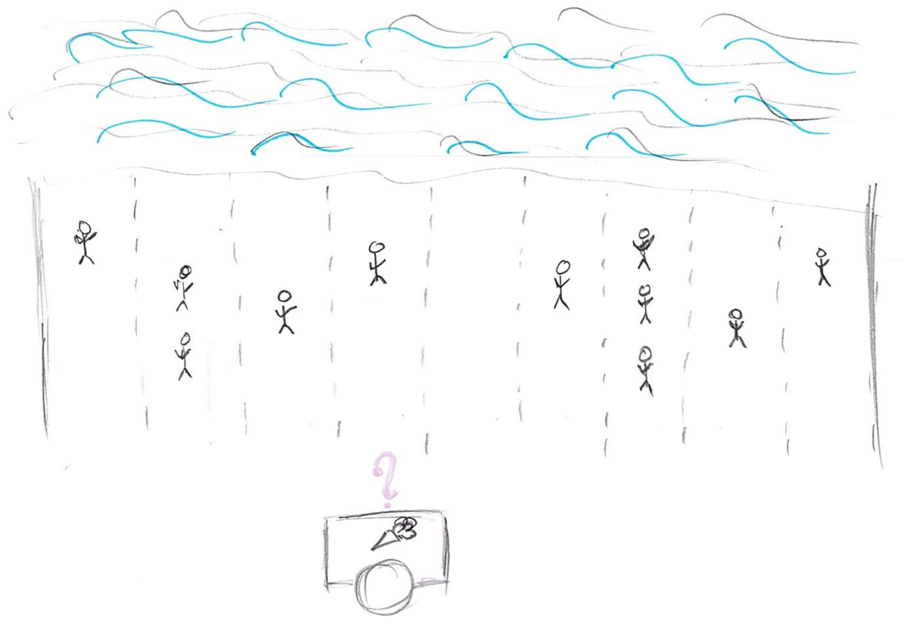
2.1 Loss function, the concept

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?

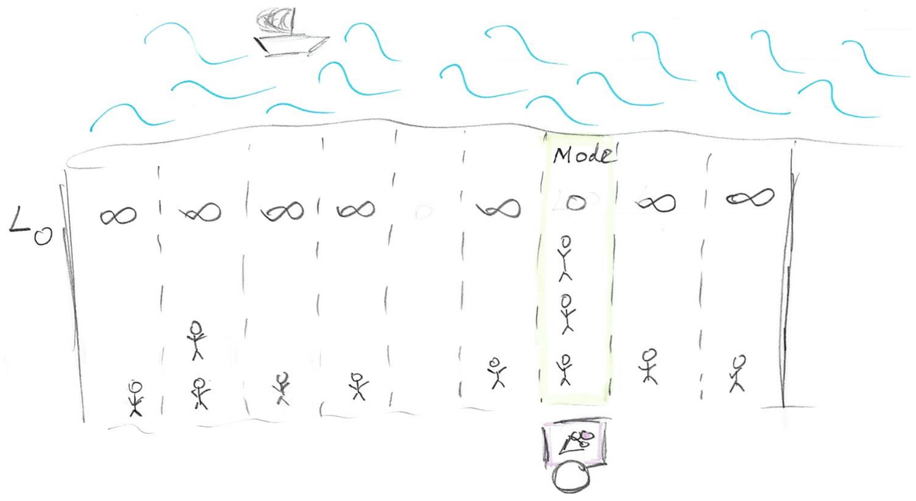


2.2 L0 (mode)

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.

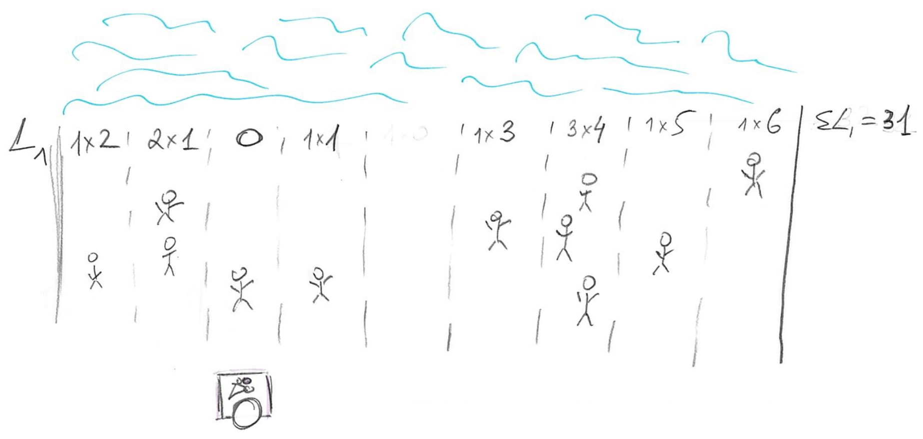


2.3 L1 (median)

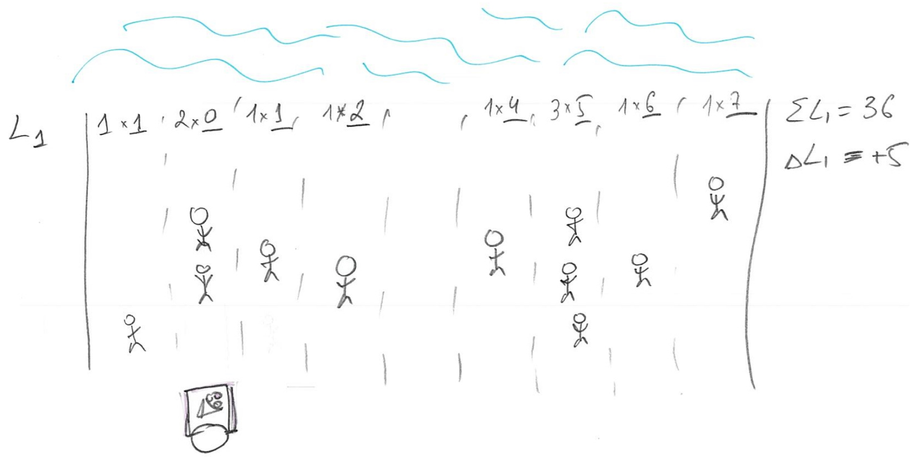
The next loss function, that I already mentioned, assumes a simple linear relationship between the distance and the cost

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

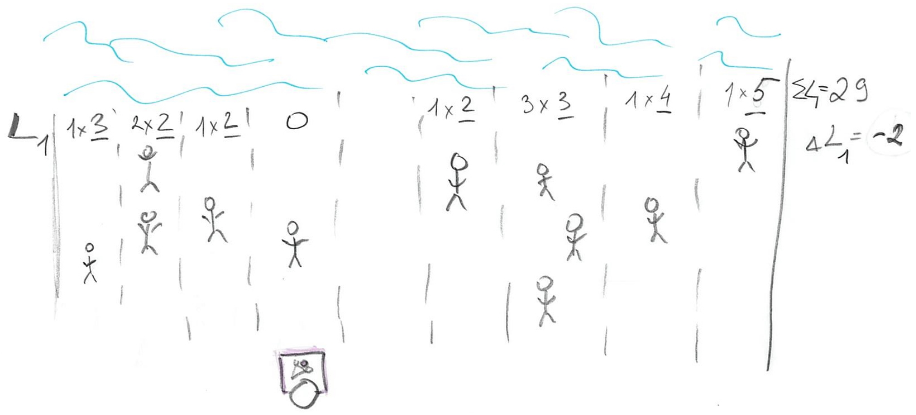
In other words, the cost is equal to distance (we need `| |` to get an absolute value, because the person could be "to the left of" of stand, in which case `person - stand` distance will be negative). So, where should we put our stand? Let us start at a fairly random location so that 3 of our customers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



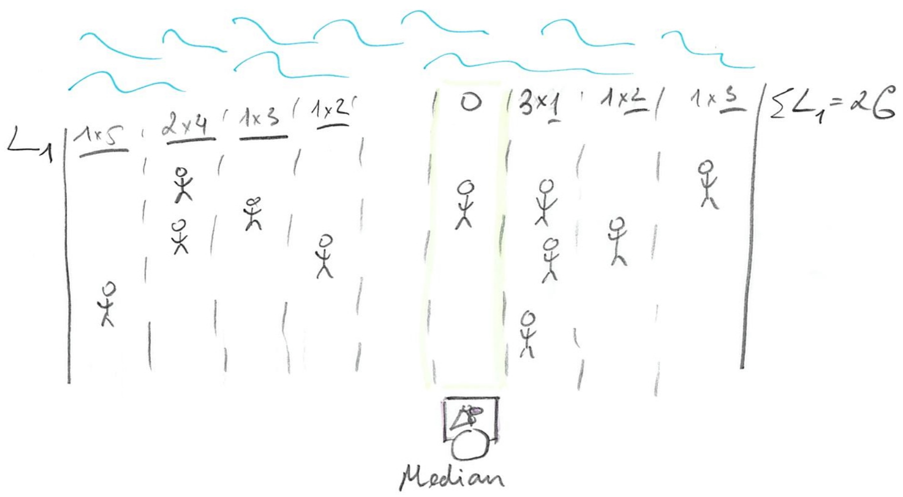
We can, in principle, compute the actual cost but it is simpler to ask the question of whether we can *improve* on that cost by moving somewhere else? Imagine that we move to the left where *minority* of our customers are. Now we have 1 on the left and 8 on the right (plus 2 more at our location).



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*.

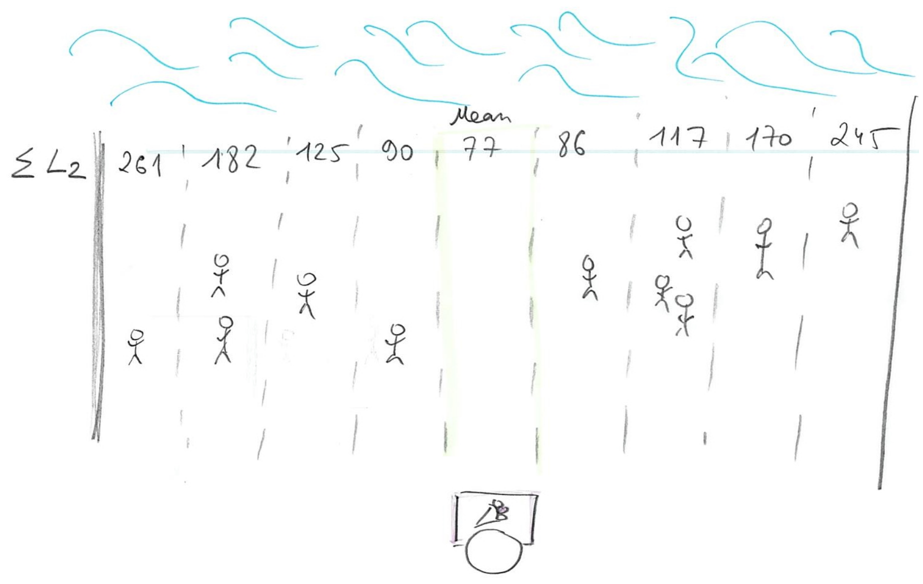
2.4 L2 (mean)

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

$$-\frac{2}{N}\sum_{i=1}^{N} person_i + \frac{2 \cdot N}{N} \cdot stand = 0 2 \cdot stand = \frac{2}{N}\sum_{i=1}^{N} person_i stand = \frac{1}{N}\sum_{i=1}^{N} person_i$$

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)

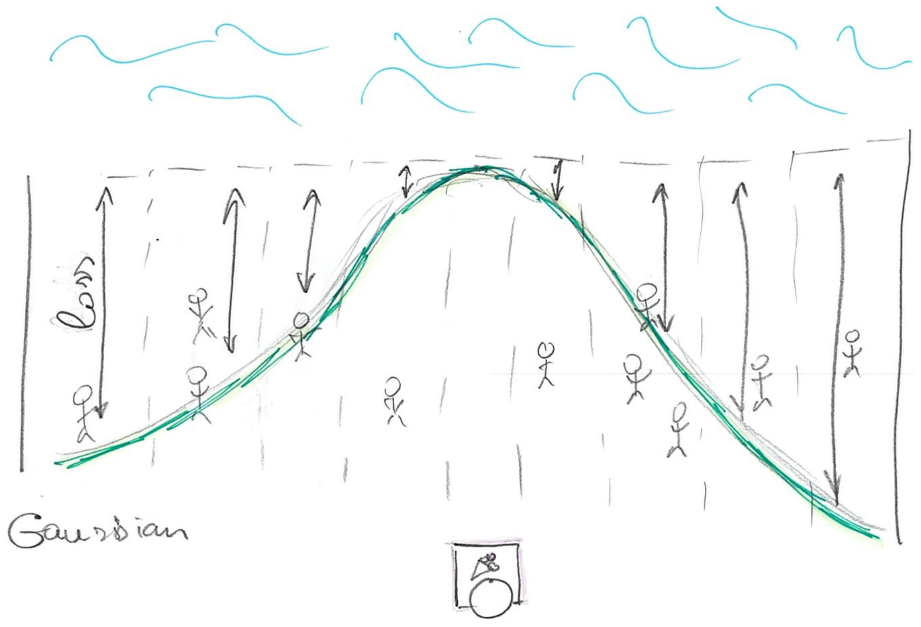
One problem about the *mean* is that it is sensitive to outliers. Because the costs grow as a power law, this approach favors a lot of medium-sized distances over lots of smalls ones plus one really large one. Thus, a single person at a far side of the beach would have a big influence on your stand's 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.

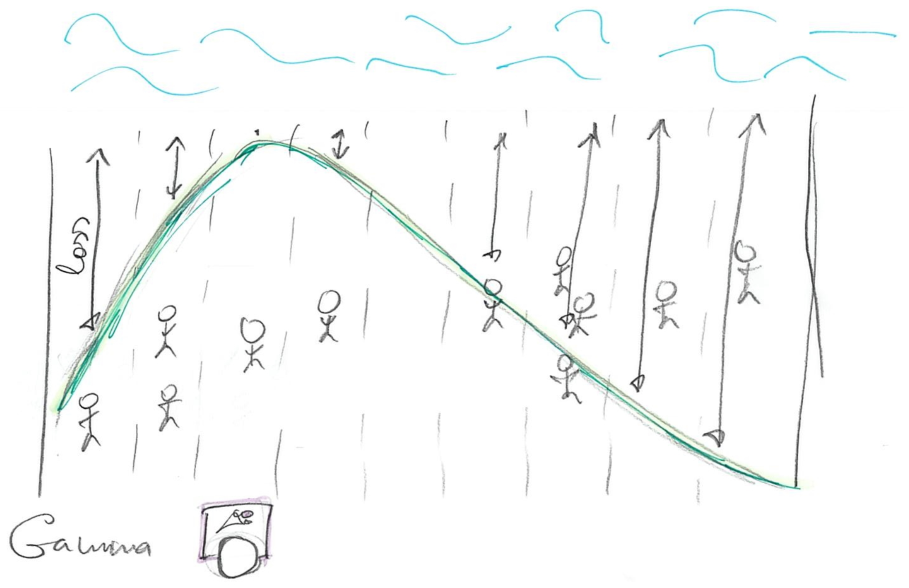
Alternatively, you can express same costs-of-being-off-the-line using a distribution, such as Gaussian. You put its peak (mean) at the (candidate) location of your point estimate (that point has highest probability, so lowest cost) and the loss is computed as a probability of the residual (distance-to-the-point). 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

Directed Acyclic Graphs and Causal Reasoning

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.

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

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

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".

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

[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.

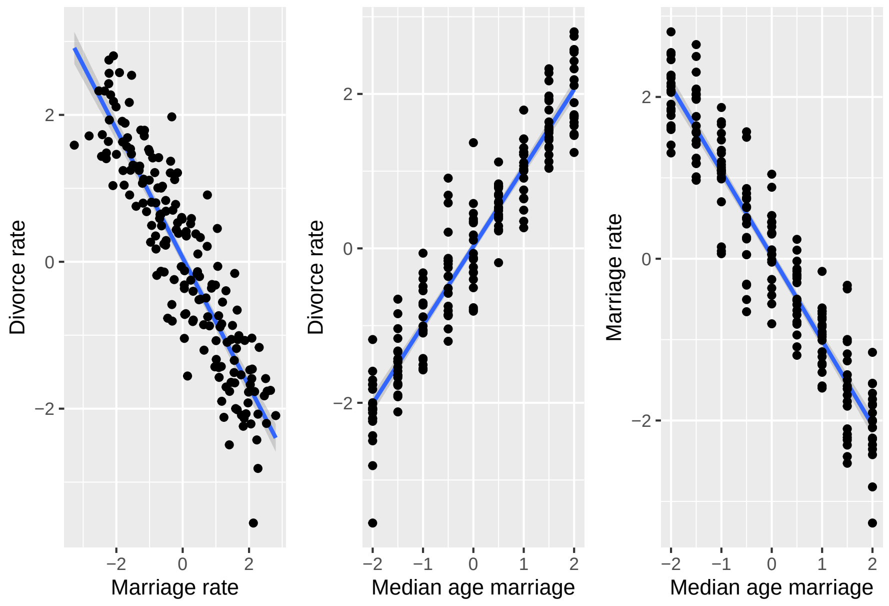
of variables can be both more specific (e.g., they are related via a third variable or they both independently cause that third variable) and more testable, as you now have two *opposite* predictions for the same pair of variables! You can check that current flows when it should (unconditional probability) 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).

```r
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
```

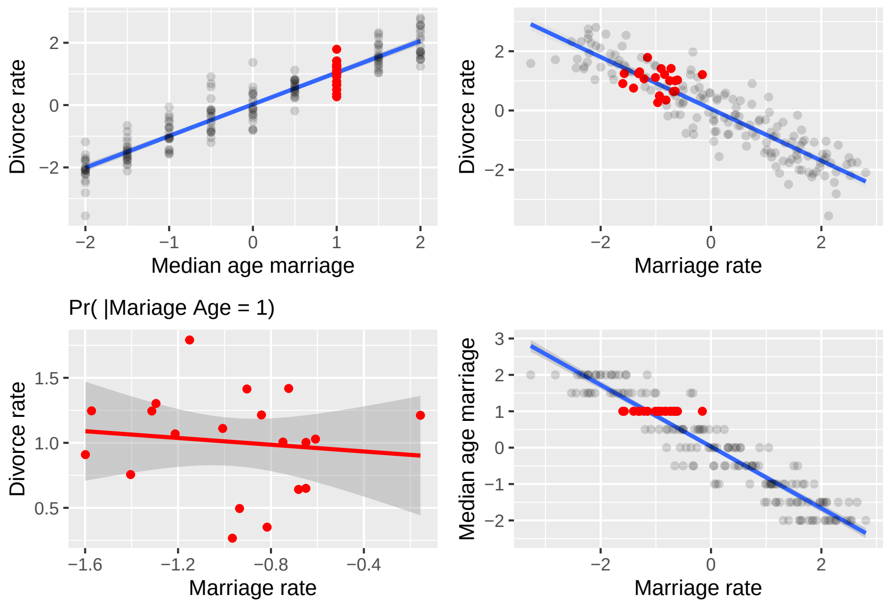


As you can see, all variables being dependent on each other, as in case of the original data. However, let us pick an arbitrary value, say $A = 1$[3] and see which dots on the left plot will be selected via *filtering* on that value.

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

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

```
  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.

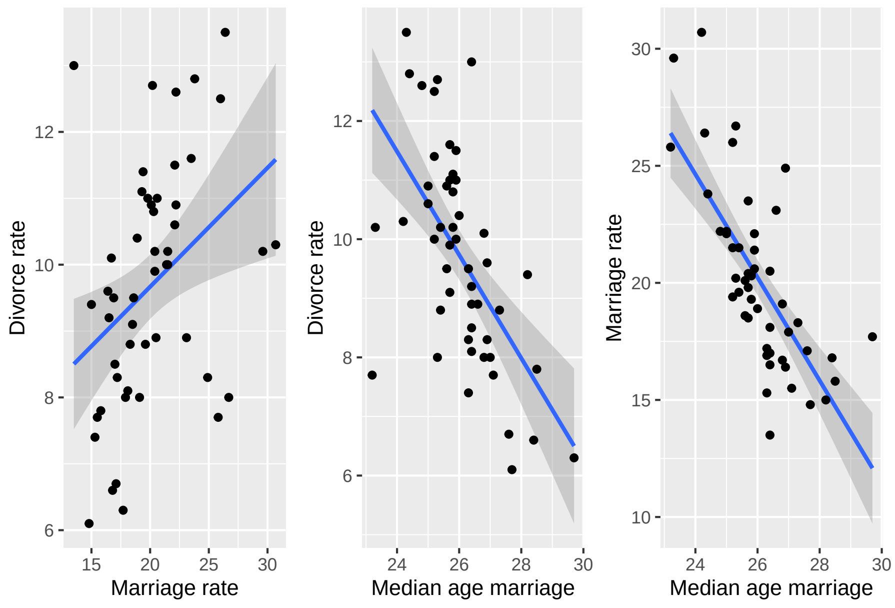
that new DAG is supported by data. If you have more variables, your DAGs will be more complex but the beauty of such causal reasoning is that you can concentrate on *parts* of it, picking three variables and seeing whether your guess about these three variables was correct. This way, you can tinker with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

Chapter 4

Collider bias

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”. As you will see below, fully understanding relationship between variables is complicate even when we are working with a minimal multiple regression with just two predictors.

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[1], 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 marriage = marriage rate + *extra information* + noise` but `marriage rate = median age marriage + noise`. In a nutshell, both variables code the same information but marriage rate is a noisier version of it, so it 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*.

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

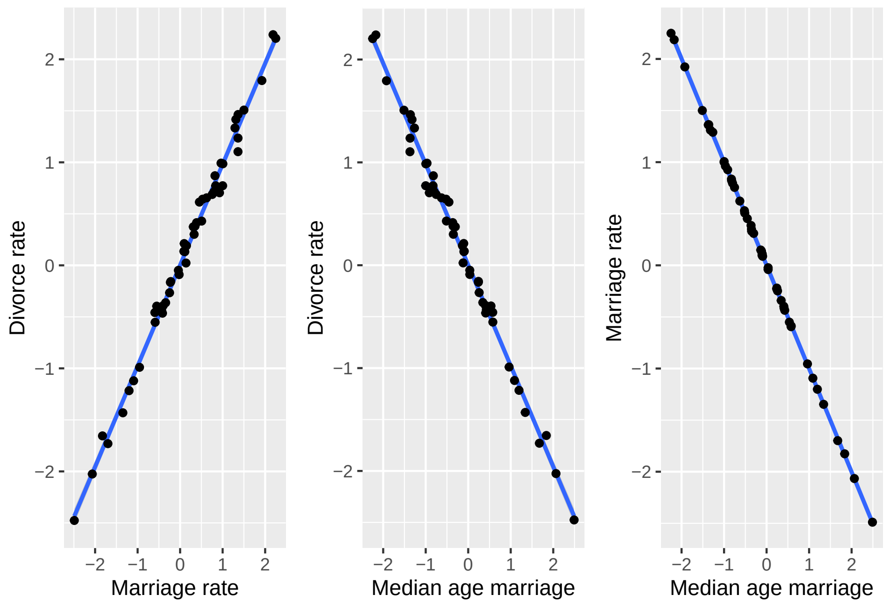
4.1 Multicollinearity

To make things easier to understand, let us use simulated data. Imagine that both marriage and divorce rate are both *caused by* marriage age and are almost perfectly linearly dependent it, so that $D_i = \beta_A^{true} \cdot A_i$ (for the sake of simplicity $\beta_A^{true} = -1$) and $M_i = -A_i$. The *causal* relationship that we are modeling is called a fork:

$$Marriage\ rate \leftarrow Age\ of\ marriage \rightarrow Divorce\ rate$$

. We pretend our variables are already standardized, so the plots would look something like this.

```r
data("WaffleDivorce")
set.seed(1212)
N <- nrow(WaffleDivorce)
sim_waffles <- tibble(MedianAgeMarriage = rnorm(N),
                      Divorce = -rnorm(N, MedianAgeMarriage, 0.1),
                      Marriage = -rnorm(N, MedianAgeMarriage, 0.01))

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
```



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 regression line. Still, by definition of *how we constructed the data*, both marriage and divorce rate are *caused* (computed from) median age and, importantly, 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     0.002034578 0.012914402 -0.01860513 0.02267429
## bA    0.267403927 1.188973843 -1.63280591 2.16761377
## bM    1.245792079 1.187211226 -0.65160076 3.14318492
## sigma 0.090129176 0.008996557  0.07575094 0.10450741
```

Oh no, we broke it! β_M is now about `1.25` rather than zero and β_A is around `0.27` rather than `-1`, as it should. Also note the uncertainty associated with both values, as they both overlap heavily with zero[2]. So, the data generation process is the same (`Divorce rate ← Age → Marriage rate`) and the model is the same (changes to priors have no particular impact in this case) but the "magic" of inferring the lack an "causal arrow" `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 ($M = -A$), that means that you can substitute one for another. Thus, we can rewrite[3]

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

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[4] of coefficients times one of the variables and does not matter which one, since they are identical. We used `A` because we know that it causes `M`. If you look at the precis output above, you will see that we did fit the β_A^{true}! Since `bA = 0.27` and `bM = 1.25`, so plugging them in gives us

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

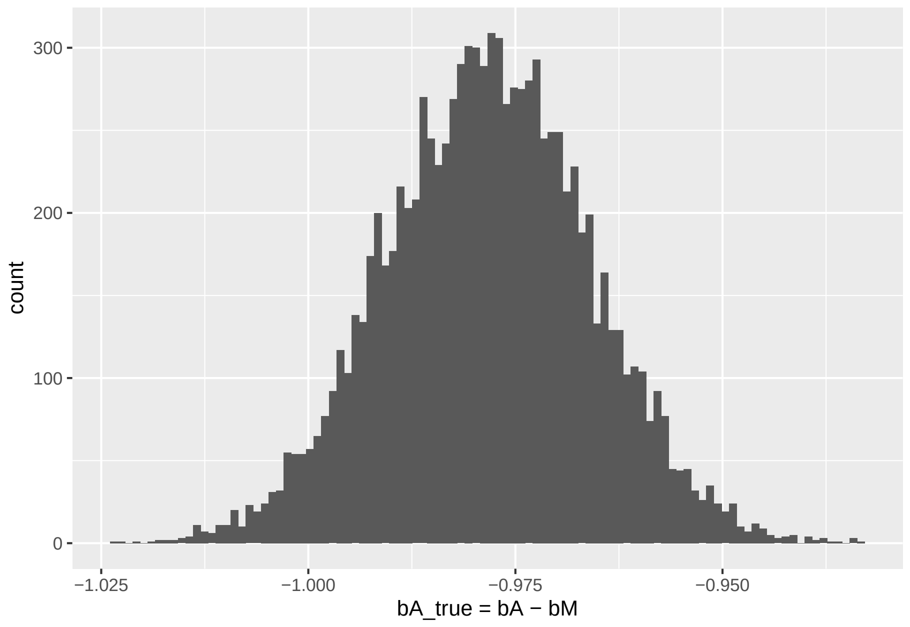
Hey, that is the slope that we used to construct divorce rate, so fitting does work! Moreover, we can see that there is very little uncertainty about β_A^{true}

[2]I've made priors for both betas broad, so that they are not pushed towards zero too aggressively and uncertainty about them is more evident

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

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

```
posterior_samples <-
  rethinking::extract.samples(sim_waffles_fit) %>%
  mutate(bA_true = bA - bM)

ggplot(posterior_samples, aes(x=bA_true)) +
  geom_histogram(bins=100) +
  xlab("bA_true = bA - bM")
```



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` in our case, of course). There are infinite number of pairs of numbers whose difference would give -1: $1-2$, $2-3$, $(-200)-(-199)$, $999.41-1000.41$, etc. All of them add up (subtract to) -1, 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.

4.2 Back to spurious association

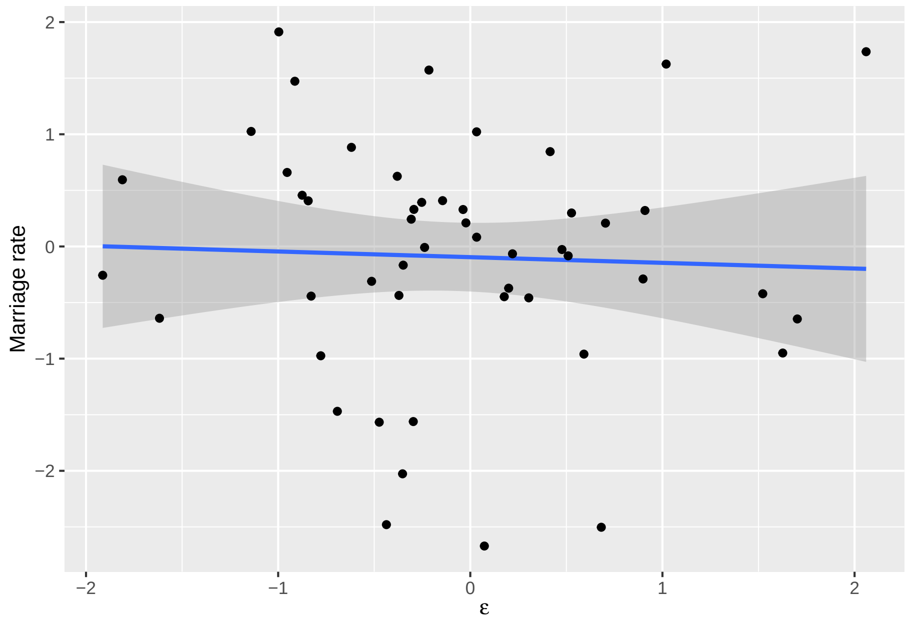
Above, you have learned that if two variable have the same information, you can only fit *both* of them but cannot get individual slopes. But wasn't that the case for real data we started with? Marriage age and rate *are* correlated, so why fitting used one (age) and not their sum? The answer is *extra 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 extra noise change our linear model for divorce rate?

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

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

```r
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")
```



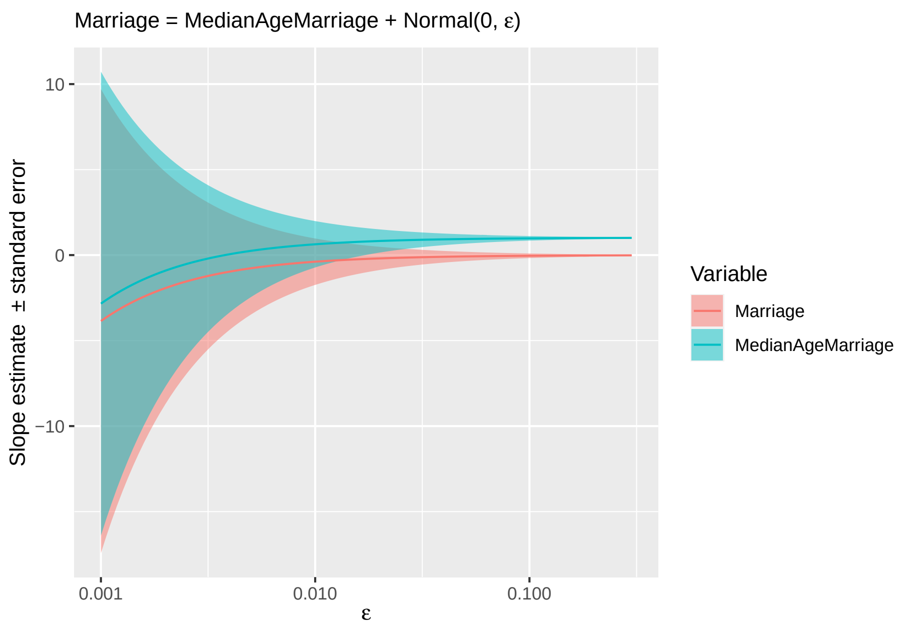
But we are not fitting it alone, as the coefficient β_M appears at *twice*:

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

The latter part, $\beta_M \cdot \epsilon$, pushes β_M towards zero slope, which is the best solution for pure noise, as you saw in the plot above. 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 (extra residuals) and the only way to reduce residuals is to ignore the *uncorrelated* 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[5].

```
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
```

[5]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 relationship[6].

4.3 Chain DAG

So, a bit of noise will fix everything and we can *know* causal relationships between the three variables? Not necessarily! Consider another possible causal diagram:

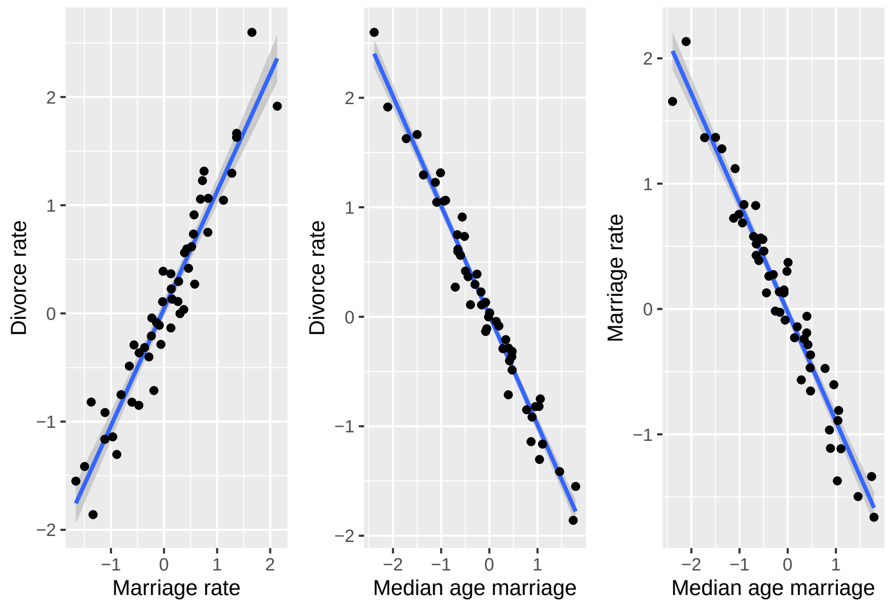
$$Marriage\ rate \rightarrow Age\ of\ marriage \rightarrow Divorce\ rate$$

Now marriage rate causes age of marriage that, in turn, causes divorce rate. Again, let us use synthetic data, so that we can be sure what causes what. However, we will add a fair amount of noise to make it more like real data and avoid multicoliniarity.

```
data("WaffleDivorce")
set.seed(8973791)
N <- nrow(WaffleDivorce)
sim_waffles <- tibble(Marriage = rnorm(N),
```

[6]It is true only in a sense that it matches the processes of creating the data. It is *not* necessarily truly true for real data!

```
                      MedianAgeMarriage = -rnorm(N, Marriage, 0.2),
                      Divorce = -rnorm(N, MedianAgeMarriage, 0.2))

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
```



The plots look very similar to those that we had before, so let us run the model.

```
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       0.01141493 0.02370058 -0.02646318  0.04929304
## bA    -1.04399582 0.11019913 -1.22011532 -0.86787631
## bM    -0.05011269 0.12299879 -0.24668851  0.14646313
## sigma  0.16656407 0.01661272  0.14001374  0.19311440
```

The model fit is virtually identical to that for the real data and, effectively, for a noisy simulated fork DAG. Yet, remember, that the underlying causal relationship between marriage rate and marriage age are now *opposite*. In the fork DAG, age was causing marriage rate. Here, in the chain DAG, marriage rate for causing age. Moreover, the way we generated synthetic data, marriage rate causes divorce rate, although its effect is *mediated* via marriage age. Yet, looking at the parameter values for β_M we might be tempted to conclude that marriage rate has no effect on divorce rate. To understand why, think about the relationship between marriage rate and age again. We designed it, so that $A = -M + \epsilon$ and $D = -A$ (we ignore the noise in the latter part for clarity). Substituting, we get $D = M - \epsilon$ or, since we designed noise to be symmetric, we can also write $D = M + \epsilon$. To put it differently, divorce rate are based on *actual* values of age, which include noise. So, somewhat paradoxically, the cleaner version of the original variable is less correlated. If it still unclear, let me try with a metaphor. Imagine your friend sings you a song she heard. $original \rightarrow friend \rightarrow you$. She remembered it wrong, so her version is somewhat different from the original. But because you are used to *her* version of the song, the original, once you finally hear it, sounds wrong, as it does not have the distortions introduced by your friend.

4.4 Take-home message

So, two DAGs, two *differently* generated data sets, yet, one statistical model. Both DAGs agree that there is a direct causal path between marriage age and divorce but have opposite assumptions about causal relationship between marriage age and rate. What should we conclude from this? That it might be impossible to understand causal relationships between all variables even for the simplest case of just two predictors. The only thing to do is to embrace this ambiguity and be aware of it whenever you interpret regression models. So, you should use DAGs exactly for this purpose of understanding in how many ways can you generate the observed data. I understand that a simple unambiguous story is far more appealing[7] but as Jochen Braun of Magdeburg says: “Do not fall for your own propaganda!”. Selecting just one story / DAG does not make them true. Considering all possible DAGs will likely lead to insights based on predictions they make. Even if your current data cannot decide between them, their different predictions for future experiment will help to solve the puzzle eventually.

Another take home message: Regression models cannot do magic. They can quantify correctional relationship and their signal-to-noise ratios but they do

[7]“Just tell me how things are!”

not know causality and they won't tell you which model is the "true" model. Imagine that we never measured marriage age, we would model divorce rate from marriage rate and would be quite pleased with the results. And, given how noisy the real data is, we probably did not consider some other *very important* variables, those presence might render age as irrelevant as the marriage rate is now. Again, assuming the chain DAG

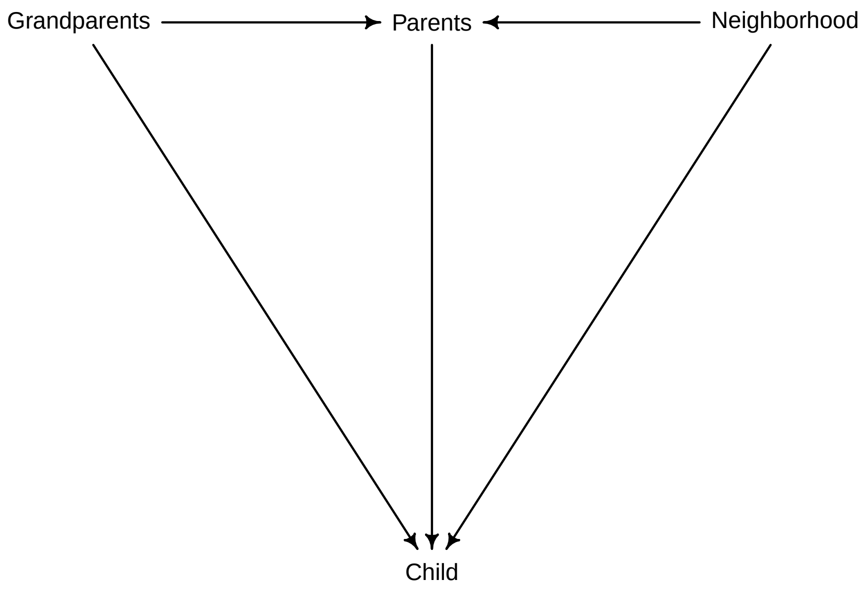
$$\textit{marriage rate} \rightarrow \textit{age} \rightarrow \textit{variable we missed} \rightarrow \textit{divorce rate}$$

Statistical models only work in a small world you created for them. Models are golems, they cannot and do not know about the large world. You do, so it is on you to understand both their limitations and power. Models can help you understand the process you are investigating but they won't understand it for you!

Chapter 5

The haunted DAG

These are notes on section **6.3.2 The haunted DAG** that demonstrates how collider bias can arise, if one of the variables is unobserved and we do not realize that the have a collider in our DAG. The DAG itself is below, I've only changed `Unobserved` into `Neighborhood`.



The example in the book shows that if we include `Grandparents` while "controlling for" influence of `Parents` but ignore influence of the `Neighborhood`, the influence of `Grandparents` is *negative* rather than *zero*. It is zero in the book but it can originally be positive or negative, point is it is different from what is inferred by the model. And, importantly, an estimated effect of `Parents` is also different from its true value. The book shows it visually but I found that in this case algebra is helpful to understand it at the level of "how does a regression model know".

Let us start with a *full* model. For the sake of simplicity, we will ignore coefficients for computing `Parents` variable from `Grandparents` and `Neighborhood` but this does not change the general picture.

$$P = G + N$$

Thus, the linear model for child education is

$$C = \beta_P P + \beta_G G + \beta_N N$$

Substituting \$ P = G + N\$ gives us

$$C = \beta_P (G + N) + \beta_G G + \beta_N N$$

Rearranging terms a little

$$C = (\beta_P + \beta_G) G + (\beta_P + \beta_N) N$$

Note that this means that we do not fit individual coefficients, in both cases we fit a *sum* of two. And, as with multicollinearity, individual coefficients can be in a wrong place and unreliable as long as they add up to the "true" coefficient value. Thus, ignoring the noise and concealing the effect of `Parents`, we might as well fit

$$C = \beta'_G G + \beta'_N N$$

where, $\beta'_G = \beta_P + \beta_G$ and $\beta'_N = \beta_P + \beta_N$ are the total effect of grandparents and neighborhood of child education.

What if we ignore `Neighborhood`? This means that we explicitly set $\beta_N = 0$ and that is the point when the sum of coefficients starts causing us problems. Recall that the model fits $\beta_P + \beta_N$ and not each term separately. Thus, setting one of them to 0 does not upset the model, as it can always compensate with the *other* coefficient. Here, that *other* coefficient is β_P, so its value now is that of the "true" sum: $\beta_P = \beta_P^{true} + \beta_N^{true}$.

Unfortunately for us, β_P appears at *two* places, as it is also used to quantify effect of grandparents:

$$(\beta_P + \beta_G) G$$

Originally, it reflected only the influence of parents, so it was not a problem. But now it is artificially inflated [1] as it stands for influence of both parents *and*

[1] In general, changed depending on the effect signs of individual effects.

neighborhood. Problem? Not for a model that fits a *sum*. How do you make sure that *the sum* still adds up? You change *other* coefficients! Here, we can still wiggle β_G so that everything adds up. Given that

$$\beta_P = \beta_P^{true} + \beta_N^{true}$$

model just needs to subtract that same β_N^{true} and get our sum back. So

$$\beta_P + \beta_G = (\beta_P^{true} + \beta_N^{true}) + (\beta_G^{true} - \beta_N^{true})$$

Thus, if we do not explicitly model the effect of neighborhood, it sneaks in nonetheless, hiding inside *both* parent and grandparent coefficients. What makes it really problematic and confusing is that the effect is *opposite* for the two terms: if we *add* β_N^{true} at one place, we must *subtract* it at the other.

So, what are you supposed to do with this new knowledge? You do not take fitted model coefficients at their face value. You always have a fine-print "Only assuming that my small world is correct and it probably is not" at the back of your head. You think of several models and think of ways to tease out the true relationship. DAGs and statistical models can help you, but they cannot do magic and tell the "real truth" by themselves.

Chapter 6

Information Criteria

These are notes on information criteria, presented in chapter 7 of the Statistical Rethinking book. The purpose of this note is to provide some intuition about the information criteria presented in the "Statistical Rethinking" book. I deliberately oversimplified and overgeneralized certain aspects to paint a "bigger picture".

6.1 Deviance

In the chapter, deviance is introduced as a an estimate for KL-divergence, which in turn is a relative entropy, i.e., the difference between cross-entropy and actual entropy of events. Keep that in mind but you could look at deviance itself as a straightforward goodness-of-fit measure, similar to squared residuals and coefficient of determination R^2. In both cases, you have difference between model prediction (the regression line) and an actual data point. In ordinary least squares (OLS) approach, you quantify this imperfection of prediction by squaring residuals. You sum up all residuals to get the sum of squared residuals (SS_{res}) and then you can compute R^2 by comparing it to the total sum of residuals in the data (SS_{total}):

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}}$$

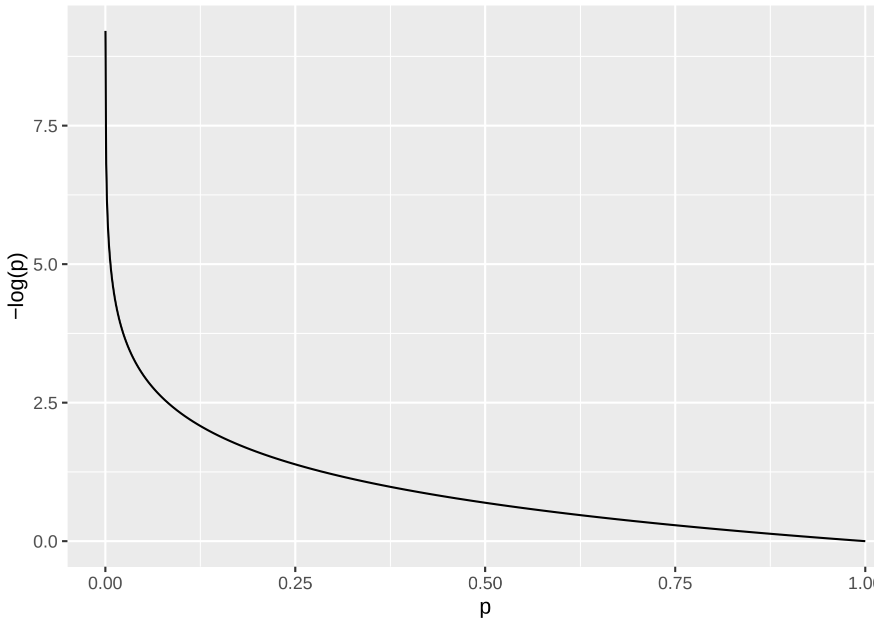
If you use likelihood, you compute the probability that a data point comes from a distribution defined by a model. Then, you compute the (total) joint probability by multiplying all probabilities or, better still, by computing the sum of their log-transform (log likelihood). Then, you can compare the observed log likelihood to the highest theoretically possible log likelihood for a *saturated* *model* (Θ_s), which has as many parameters as there are data points, so that it predicts each point perfectly. This is the original definition of (total) *deviance*:

$$D = -2 \cdot (log(p(y|\Theta)) - log(p(y|\Theta_s)))$$

Recall that $log(\frac{a}{b}) = log(a) - log(b)$, so we can rearrange it and see that it is a log-ratio of likelihoods:

$$D = -2 \cdot log\left(\frac{p(y|\Theta)}{p(y|\Theta_s)}\right)$$

As $p(y|\Theta)$ increases, the fraction inside get closer to 1. The $log()$ bit flips and non-linearly scales it. The minus sign flips it back and we end up with smaller



numbers meaning better fit.

The 2 is there to facilitate significance testing for *nested* models. Models $\Theta 1$ and $\Theta 2$ are nested, if $\Theta 2$ has all predictors of $\Theta 1$ plus k predictors. E.g., model $Divorce = Marriage\ Rate$ is nested inside $Divorce = Marriage\ Rate + Marriage\ Age$ with later model having 1 more parameter. Your actual model Θ is nested inside the saturated model Θ_s that has $k = n - k_{model}$ more parameters, where n is sample size and k_{model} is number of parameters in your model. It turns out that in this case, you can determine whether the difference in goodness-of-fit between two models is significant using chi-squared distribution with k degrees of freedom. The only catch is that log-ratio is half the magnitude, so you need that 2 to match things up[1].

[1] I did not follow the derivation of that correspondence yet, so I cannot comment on how and why.

At this point, you might remember that the definition of deviance in the book was for a single model:

$$D = -2 \cdot log(p(y|\Theta))$$

Unfortunately, same term can be used both ways, to refer to a log likelihood of a single model or, technically more correctly, to the log-ratio you saw above. In reality, you will mostly see deviance defined as in the book because it is used to compare nested models via chi-square distribution as I've described above, with only difference that you compare any two nested models, not just to a saturated one. This is how nested models are frequently compared, for example, see anova() function in R.

Note that a deviance for a single model still expresses the same idea of goodness-of-fit but is merely not normalized. Thus deviance as in the book $D = -2 \cdot log(p(y|\Theta))$ corresponds to sum of residuals (absolute values mean nothing but you can use them to compare two models on the same data), whereas *total* deviance corresponds to the R^2 (values are directly interpretable, theoretically, you can use them to compare models fit on different samples).

6.2 General idea: information criteria as miles-per-gallon

The general formula for all information criteria discussed below is

$$-2 \cdot log\left(\frac{goodness\ of\ fit}{model\ complexity}\right)$$

The goodness-of-fit in the numerator is the likelihood, the joint probability of observing the model given each data point $\prod_i p(\Theta|y_i)$. The denominator expresses model complexity, i.e., its flexibility in fitting the sample and, therefore, its tendency to overfit. Thus, the fraction itself is *goodness-of-fit per unit of model complexity.* This is like miles-per-gallon for car efficiency, so better models are more efficient models, churning out more goodness per complexity.

The numerator is the same for all information criteria discussed below and they differ only in how they compute the model complexity.

6.3 Akaike Information Criterion (AIC)

The formula most commonly used is

$$AIC = -2 \cdot log(p(\Theta|y)) + 2k$$

where k is the number of parameters of the model. If you are not a mathematician who is used to translate back-and-forth using logarithms, you may fail spot to the ratio I was talking about earlier. For this, you need to keep in mind that $log(\frac{a}{b}) = log(a) - log(b)$ and that $a = log(exp(a))$. Let us re-arrange a bit to get the log-ratio back:

$$AIC = -2 \cdot log(p(\Theta|y)) + 2k$$

$$AIC = -2(log(p(\Theta|y)) - k)$$

$$AIC = -2(log(p(\Theta|y)) - log(exp(k))$$

$$AIC = -2 \cdot log\left(\frac{p(\Theta|y)}{exp(k)}\right)$$

And here it is, the log-ratio I've promised! As you can see, AIC assumes that model complexity grows exponentially with the number of parameters. In other words, adding another parameter increases complexity ~2.718 times.

If you are to use AIC, the current recommendation is to *correct it* with an extra penalty for the size of the sample

$$AICc = AIC + \frac{2k^2 + 2k}{n - k - 1}$$

where n is the sample size. I won't do it here but you should be able to work out how it is added into the exponent in the denominator.

6.4 Bayesian information criterion (BIC)

A.k.a. Schwarz information criterion (SIC, SBC, SBIC). The motivation is the same as with AIC but the complexity term, in addition to the number of parameters, also reflects the sample size n.

$$BIC = -2 \cdot log(p(\Theta|y)) + log(n)k$$

Let us do re-arranging again

$$BIC = -2 \cdot log(p(\Theta|y)) + log(n)k$$

$$BIC = -2\left(log(p(\Theta|y)) + log(n)\frac{k}{2}\right)$$

$$BIC = -2\left(log(p(\Theta|y)) - log\left(exp(log(n) \cdot \frac{k}{2}\right)\right)$$

For the complexity term, we need to keep in mind that $exp(a \cdot b) = exp(a)^b$. Thus,

$$exp\left(log(n) \cdot \frac{k}{2}\right) = exp(log(n))^{\frac{k}{2}} = n^{\frac{k}{2}}$$

Putting the complexity term back, we get

$$BIC = -2\left(log(p(\Theta|y)) - log\left(n^{\frac{k}{2}}\right)\right)$$

$$BIC = -2 \cdot log\left(\frac{p(\Theta|y)}{n^{\frac{k}{2}}}\right)$$

Thus, we end up with very similar power law complexity term which uses sample size instead of Euler's number as the base.

6.5 Problem of AIC and BIC: one size may not fit all

Both AIC and BIC assume that model complexity and flexibility, that leads to overfitting, is reflected in the number of parameters *k*. However, this is a fairly indirect measure of a model flexibility, based on how models in general tend to overfit data in general. But you probably want to know how your specific model uses its parameters to fit your specific sample and how much overfitting you should expect in that specific case. Because even if a parameter is present in the model, it may not be able to fully use it in case of regularization or multilevel models.

Regularization, in form of strong priors, lasso/ridge regression, etc., restricts the range of values that a given parameter can take. Thus, a model cannot exploit it as much as *other* parameters and will be less able to use it to improve fit to the sample. Similarly, in hierarchical multilevel modeling, you may have dozens or hundreds of parameters that describe intercepts and/or slopes for individual participants (random factors, in general), but most of them could be trivially average (same as or very similar to the group average) and contribute little to the actual fit. In these cases, a simple raw count, which treats all parameters as equals, will overestimate model complexity.

The desire to go beyond one-size-fits-all approach and be as model- and data-specific led to development of deviance information criterion (DIC) and widely-applicable information criterion (WAIC). Both use posterior distribution of *in-sample* deviance and base their penalty on how *variable* this posterior distribution is. Higher variance, meaning that a model produces very different fits ranging from excellent to terrible, hints that model is too flexible for its own good and leads to higher complexity estimate (penalty for overfitting). Conversely, very similar posterior deviance (low variance) means that model is too restricted to fine-tune itself to the sample and its complexity is low.

If you understand why variance of the posterior distribution of divergence is related to models' flexibility and, therefore, to the number of effective parameters, just skip the next section. If not, I came up with a musical instruments metaphor that I and at least some people I've tested it upon found useful.

6.6 Musical instruments metaphor

Imagine that you are trying to play a song that you have just heard. But the only instrument you have is a triangle. This is not a particularly flexible instruments pitch-wise, so your rendition of that song will not be very good (your model underfits the data). The good news is that even if you do your worst and do not really try, no one will notice because your worst performance will sound very much like your best one. Simply because it is very hard to make songs sound different using a triangle. Thus, if you play that song many times, trying different versions of it with us judging how close you are to the original, the score we will give you will be very similar (low variance of the deviance for fitting to sample).

But what if I give you an instrument that can vary the pitch at least a bit, like a xylophone for children. Now you more expressive freedom and your version of the music will sound much more like the original. But, it also gives you an opportunity to make a mess of it, so your rendition might sound nothing like the music you've just heard. Thus, the instrument increases the difference between the best and the worst possible performance, so the variance of your performances (on how close they are to the original) also increases (higher variance of the deviance). A more flexible instrument will make the difference even bigger. Think violin or trombone which are not restricted to the scale, so you can play any sound in between and you can match the music you just heard exactly. Imagine that the music your just hear has an odd off-the-scale sounds. Was a defect of the turntable, which cannot go at constant speed, so overall pitch wobbles overtime (noise)? Or is it an experimental music piece designed to sound odd (signal)? If you do not know for sure, you will try to play as close to the original music your heard as possible, matching those off-scale sounds. And, because you can play any sound, your range of possible performance is even larger from one-to-one to "please, have mercy and stop!" (even larger variance of deviance).

In short, variance of your performance (posterior divergence) reflects how flexible your instrument is. But why is it indicative of the *effective* number of parameters? Here are regularizing priors in the world of music instrument metaphor. Imagine that in addition to the triangle, I also give you a rubber bell (it was used in a super-secret meeting in Stanislaw Lem's Eleventh Voyage of Ijon Tichy, so that no one would hear when they ring that bell!). Now, technically you have two instruments (your number of parameters is 2) but that bell does not affect your performance (we put very strong regularizing priors so that coefficients are zero or something very-very close to zero). Thus, your ability to play the song did not change and your variance of performance stays the same. Two actual instruments, but only one "effective" one. Or, I give you a piano but allow you to use only one octave and only white keys. Yes, you have a piano but with this regularization it is as complex as as kids' xylophone. The *potential* number of notes you can play is great (AIC and BIC would be very impressed and slap a heavy penalty on you) but the actual "effective" range is small. Or, you regularize yourself to play scale only notes using violin (something you learn to do). In all of these cases, you deliberately restrict yourself. But why? Why not just play as you heard it (why not fit as well as you can?)? Because if the song you heard is short (sample is small), regularization based on your knowledge about real life helps you to ignore the noise that is always present. E.g., you know that song is for kids' xylophone, so even if you heard notes outside of a single octave that was probably a problem with recording. Or, you never heard that piece for violin but you do know other works of this composer and they always use scale-only notes, so you should not use my violin to play off-scale sounds.

Multilevel models also limit the actual use of parameters. Imagine you heard a recording of a symphonic orchestra. Lots of violins but you figured out that most of them actually play the same melody. So you can get away with using one violin score (sample group average) and assume that most violins play like that (most participants are very close to group average). Any deviations from that group melody are probably mistakes by individual musicians, not the actual melody. Same goes if you hear a choir. Again, many people sing (lots of parameters!) but, mostly, in unison, so you do not need to create an individual score sheet (parameter) for each singer, just one per group of singers.

Wrapping up the metaphor, the more flexible your instrument is, the more variable your performance can be, the easier it is for you to mimic noise and imperfections of the recording that have nothing to do with the piece itself. But when you play it next time, matching the recording with all its noise and distortions perfectly, people who know the piece will be puzzled or may not even recognize it (poor out-of-sample predictions). Adopting the melody for a more limited instrument may make it easier for others to recognize the song! Thus, higher variance in performance accuracy (higher variance of deviance) indicates that you can overfit easily with that instrument (model) and you should be extra careful (impose higher penalty for complexity).

6.7 Deviance information criterion (DIC) and widely-applicable information criterion (WAIC)

The two are very similar, as both compute the model complexity based on posterior distribution of log likelihood. The key difference is that DIC sums the log likelihood for each model (sample) first and then computes the variance. WAIC computes variance of log likelihood per point and then sums those variances up. In the musical instrument metaphor, for DIC you perform the piece many times (generate many posterior samples), compute accuracy for each performance (deviance for a single sample), and then compute how variable they are. For WAIC, you go note by note (observation by observation). For each note you compute variance over all sample to see how consistent you are in playing it. Then, you sum this up.

$$DIC = -2 \cdot \left(log(p(y|\Theta)) - var(\sum log(p(y|\Theta_i)))\right)$$
$$WAIC = -2 \cdot \left(log(p(y|\Theta)) - \sum_i var(log(p(y_i|\Theta)))\right)$$

The penalty replace k in AIC and, therefore, will go into the exponent inside the ratio. Again, same idea, that increase in variance of deviance (either per sample in DIC or per point in WAIC) leads to exponentially increasing estimate of complexity.

WAIC is more stable mathematically and is mode widely applicable (that's what statisticians tell us). Moreover, its advantage is that it explicitly recognizes that not all data points in your sample are equal. Some (outliers) are much harder to predict than others. And it is variance of log likelihood for these points that determines how much your model can overfit. An inflexible model will make a poor but consistent job (triangles don't care about pitch!), whereas a complex model can do anything from spot-on to terrible (violins can do anything). In short, you should use WAIC yourself but recognize DIC when you see it and think of it as somewhat less reliable WAIC, which is still better than AIC or BIC when you use regularizing priors and/or hierarchical models.

6.8 Importance sampling

Importance sampling is mentioned in the chapter but is never explained, so here is a brief description. The core idea is to pretend that you sample from a distribution you need (but have no access to) by sampling from another distribution (the one you have access to) and "translating" the probabilities via *importance ratios*. What does this mean?

Imagine that you want to know an average total score for a given die after you throw it ten times. The procedure is as simple as it gets: you toss the die ten times, record the number you get on each throw, sum them up at the end. Repeat the same toss-ten-times-and-sum-it-up as many times as you want and compute your average. But what if you do not have access to that die because it is *the die* and is kept under lock in International Bureau of Weights and Measures? Well, you have *a die* which you can toss and you have a list of *importance ratios* for each number. These *important ratios* tell you how much more likely is the number of *the die* (the one you are after) compared to *a die* you have in your hand. Let's say the importance ratio for *1* (so, number 1 comes up on top) is `3.0`. This means that whenever *your die* gives you *1*, you assume that *the die* came up *1* on **three** throws. If the importance ratio for *2* is 0.5, whenever you see *2* on your die, you record only half the throw (*2* comes up twice as rarely for real die than for your die, so two throws that give you *2* amount to a single throw). This way you can toss your die and every toss equates to different number of throws that generated the same number for *the die*. So, you sample your die but record outcomes for the other die. Funny thing is that you don't even need to know how fair your die is and what is the probability of individual sides. As long as you know the importance ratios, keep tossing it and translating the probabilities, you will get the samples for the die you are interested in.

Note that if you toss your die ten times, the translated number of tosses for *the die* does not need to add up to ten. Imagine that, just by chance, you got *1* four times. Given the importance ratio of `3.0` that alone translates into twelve tosses. Solution? You *normalize* your result by *sum of importance ratios* and get back you ten tosses.

The very obvious catch is, *how do we know the importance ratios*? Well, that is situation specific. Sometimes, we can compute them because we know both distributions, it just that one is easier to sample than the target one (so, we optimize the use of computing power). Sometimes, as in case of PSIS/LOO below, we can use an approximation.

6.9 Pareto-smoothed importance sampling / leave-one-out cross-validation (PSIS/LOO)

The importance sampling I've described above is the key to the PSIS/LOO. The idea is the same, we want to sample from the posterior of the model that was fitted *without* a specific data point y_i (we write it as $p(\Theta_{-i,s}|y_i)$). But we do not really want to refit the model. Instead, we want to use what we already have, samples from the model that was fit on all the data, *including* point y_i. So, we pull of the same importance sampling trick, sampling from $p(\Theta_s|y_i)$ and translating it to $p(\Theta_{-i,s}|y_i)$. The only thing we need are importance factors[2]

$$r_i = \frac{p(\Theta_{-i,s}|y_i)}{p(\Theta_s|y_i)} \propto \frac{1}{p(y_i|\Theta_s)}$$

The importance ratio tells you that decrease in your ability to predict an observation out-of-sample is proportional to difficulty of predicting in-sample. Here is an intuition behind this. *Any* observation will be harder to predict, if it was not included into the data the model was trained on. This is because, in its absence the model will use its parameters to fit the data that is present, including fitting noise, if it has spare parameters. So, you expect that out-of-sample deviance $(p(y_i|\Theta_{-i})$ should be always worse than in-sample deviance for the same observation $(p(y_i|\Theta)$. How much worse depends on how "typical" the observation is.

[2]Unfortunately, I did not yet follow the derivation of this proportionality, so I cannot explain why this is the case mathematically.

If it is typical and “easy” for a model to predict, in its absence the model will still see many similar “typical” observations and will be well prepared to predict it. However, if the observations is atypical, an outlier, the model won’t see too many observations that are alike and will concentrate more on typical points.

I.e., assume that out-of-sample deviance is in-sample deviance squared (the importance ratio we use goes in that direction). Asume that our *in-sample* deviance is 60, 40, and 50 in one case and 10, 10 and 130 in the other case. Both deviances sum up to 150, so from *in-sample* point of view they are equal. However, approximating *out-of-sample* deviance as a square of *in-sample* deviance makes for a dramatic difference. For the first case, it jumps to 7700 but the second case to 17100, more than twice the difference. This is because in second case *out-of-sample* deviance is dominanted by that single very difficult point, which will be terribly difficult out-of-sample.

This explosive powerlaw-like scaling of expected out-of-sample deviance is the reason for warnings that LOO/PSIS generates. A single very difficult observation can be single-handedly responsible for most of the deviance, making it hard to understand whether the model is good or bad on the rest of observations. Thus, you should consider what this extreme outlier means for your understanding of the process. Is it so different because it was recorded wrongly (the responsible person pressed 1 instead of 7)? Is this an extreme response time because a participant was distracted on that trial and started paying attention to the task only 15 seconds after the trial began? Is it an important observations that indicates that you are ignoring a really important predictor? You should think carefully about it and decide on whether you treat that observation differently (e.g., allowing for larger uncertainty for that point because you think this is mostly noise) or you change your model to better account for it.

6.10 Bayes Factor

Not an information criterion. However, it is a popular way to compare Bayesian models. Compared to information criteria, the logic is reversed. In case of the information criteria, we are asking which **model fits data** the best given the penalty we impose for its complexity. In case of Bayes Factor, we already have two models (could be different models with different number of parameters or just with different parameter values) and we are interested how well the **data matches models** we already have.

Let’s start with the Bayes theorem:

$$Pr(M|D) = \frac{\Pr(D|M)\Pr(M)}{\Pr(D)}$$

where, D is data and M is the model (hypothesis). The tricky part is the marginal probability (prior) of data $Pr(D)$. We hardly ever know it for sure, making computing the "correct" value for $Pr(M|D)$ problematic. When using posterior sampling, we side-step the issue by ignoring it and normalizing the posterior by the sum of the posterior distribution. Alternatively, when comparing two models, you can compute their ratio:

$$\frac{\Pr(D|M_1)}{\Pr(D|M_2)} = \frac{\Pr(M_1|D)}{\Pr(M_2|D)} \cdot \frac{\Pr(M_1)}{\Pr(M_2)}$$

here $\frac{\Pr(D|M_1)}{\Pr(D|M_2)}$ are **posterior odds**, $\frac{\Pr(M_1|D)}{\Pr(M_2|D)}$ is **Bayes Factor**, and $\frac{\Pr(M_2)}{\Pr(M_1)}$ are **prior odds**. The common $Pr(D)$ nicely cancels out!

If you assume that both hypotheses/models are equally likely (you have flat priors), the prior odds are 1:1 and your posterior odds are equal to Bayes Factor or, vice versa, Bayes Factor is equal to posterior odds. This means you can just pick their likelihoods from the posterior sampled distribution and compute the ratio.

I am not a big fan of Bayes Factor for conceptual reasons. Although it can compare any two models (as long as the sample is the same), it looks a lot like a Bayesian version of a p-value and, therefore, lends itself naturally to the null-hypothesis testing. And, as far as my reading of literature in my field is concerned, this is how people most frequently use it, as a cooler Bayesian way of null-hypothesis testing. You have no worries about multiple comparisons (it is Bayesian, so no need for error correction!) and it can prove null hypothesis (it is the ratio, so flip it and see how much stronger *H0* is)! There is nothing wrong with this per se but the advantage of Bayesian statistics and information criteria is that you do not *need* to think in terms of null hypothesis testing and nested models. Adopting Bayes Factor may prevent you from seeing this and will allow you to continue doing same analysis just in a differently colored wrapper. Again, there is nothing wrong with exploratory analysis using null hypothesis testing until you can formulate a better model. But it should not be the *only* way you approach modeling.

Chapter 7

Bayesian versus Frequentist Statistics

I suspect that many student who read “Statistical Rethinking” have a feeling that it is something completely different from what they have been learning in “traditional” statistics classes. That Bayesian approach is more “hands-on” and complicated, whereas “normal” statistics is simpler and easy to work with even it is “less powerful.”[1] Thus, the purpose of this note is to walk you through a typical statistical analysis and focus on practical differences and, more importantly, similarity of the two approaches.

7.1 Choice of likelihood (both)

The very first we do is to look at the data and decide which distribution we will use to model the data / residuals be it normal, binomial, Poisson, beta, etc. That is the very first line of our models that goes like this

$$y_i \sim Normal(\mu_i, \sigma) \mu_i = \alpha + \beta_{x1} \cdot X1 + \beta_{x2} \cdot X2 + \beta_{x1 \cdot x2} \cdot X1 \cdot X2 \ldots \alpha \sim Normal(0,1) \beta_{x1} \sim Exponential(1) \cdots \sigma \sim Expone$$

This decision is neither Bayesian, nor frequentist. This is a decision about the model that best describes the data, so it is independent of the inference method you will use. This is a decision that you are making even if you are using “prepackaged” statistical tests like the t-test or ANOVA that assume normally distributed residuals[2].

[1] Not really.

[2] Admittedly, in this case people often start with the statistical test and see whether data is suitable rather than the other way around.

7.2 Linear model (both)

Next, you decide on the deterministic part of the model that expresses how a parameter of the distribution you chose on the previous step is computed from various predictors. E.g., for the linear regression with normally distributed residuals, you decide which predictors do you use to compute the mean. The model line would look something like this

$$y_i \sim Normal(\mu_i, \sigma)\mu_i = \alpha + \beta_{x1} \cdot X1 + \beta_{x2} \cdot X2 + \beta_{x1 \cdot x2} \cdot X1 \cdot X2 \ldots \alpha \sim Normal(0, 1)\beta_{x1} \sim Exponenti$$

Again, this is neither Bayesian, nor frequentist decision, it is a linear model decision. Chapters 4-6 and 8 concentrate on how to make this decision using directed-acyclic graphs (DAGs) and introduce concepts of multicollinearity, colliders and bias they can produce, backdoor paths and how to identify them, etc. They explain how you can make educated decision on which predictors to use based on your knowledge of the field or of the problem. At this stage you also decide on whether to normalize data, as it could make interpreting the model easier.

You always have to make this decision. For example, if you use the (repeated measures) ANOVA, you do need to decide which factors to use, whether to include interactions, should you transform the data to make coefficients directly interpretable, do you use a link function, etc.[3]

7.3 Priors (optional for Bayesian)

Priors are a Bayesian way to regularize the model, so this is something you do need to think about when doing Bayesian statistics[4]. In a model this part would look something like this

$$y_i \sim Normal(\mu_i, \sigma)\mu_i = \alpha + \beta_{x1} \cdot X1 + \beta_{x2} \cdot X2 + \beta_{x1 \cdot x2} \cdot X1 \cdot X2 \ldots \alpha \sim Normal(0, 1)\beta_{x1} \sim Exponential(1) \cdots$$

This is probably a decision that students worry about the most as it feels more subjective and arbitrary than other decisions, such as choice of the likelihood or predictors. Chapter 4 through 7 gave you multiple examples that there is nothing particularly arbitrary about these choices and that you can come up with a set of justifiable priors based on what you know about the topic or based on how your pre-processed the data[5].

[3] However, you do see cases when one simply throws all factors and interactions into the pot with little regard for an underlying causal model or interpretability of the coefficients.

[4] Modern packages like *brms* make it easy for you by deducing a set of reasonable priors for you. However, it is always a good idea to double-check them.

[5] In my experience, people tend to worry about priors for data unseen. Some kind of data of which you know absolutely nothing, hence, have trouble deducing priors. In practice, you always know something about the topic and the data. If not, you should read on it instead of using flat priors!

Still, I think for a lot of people "normal" statistics with its flat priors feels simpler and also more objective and, therefore, more trustworthy ("we did not favor any specific range of values!"). If that is the case then use flat priors (but see the side note below) making Bayesian and frequentists models identical! For me, though, writing it down explicitly makes one realize that range $-\infty, +\infty$ is remarkably large to the point of being an obvious overkill

$$y_i \sim Normal(\mu_i, \sigma)\mu_i = \alpha + \beta_{x1} \cdot X1 + \beta_{x2} \cdot X2 + \beta_{x1 \cdot x2} \cdot X1 \cdot X2 \ldots \alpha \sim Uniform(-\infty, +\infty)\beta_{x1} \sim Uniform(-\infty, +\infty) \cdots$$

In short, Bayesian inference gives you an *option* to specify priors. You do not need to take on the option and can use flat frequentist's priors.

Side note. In reality, flat priors are never good priors. If there is sufficient data then, in most cases, the priors (flat or not) do not have much influence. However, if the data is limited then flat priors almost inevitably lead to overfitting as there is no additional information to counteract the effect of noise. This overfitting may feel more "objective" and "data-driven" than a more conservative underfitting of data via strongly regularizing priors but the latter is more likely to lead to better out-of-sample predictions and, therefore, are more likely to be replicated.

7.4 Maximum-likelihood estimate (both)

Once you fitted the model, you get estimates for each parameter you specified. If you opted for flat priors, these estimates will be the same but for very minor differences due to sampling in Bayesian statistics. If you did specify regularizing priors then estimates will be different, although the magnitude of that difference will depend the amount of data: the more data you have, the smaller the influence of the priors, we closer the estimates will be (see also the side note above). Importantly, both types of inferences will produce (very similar) estimates and you interpret them the same way.

7.5 Uncertainty about estimates (different but comparable)

This is the point where the two approach fundamentally diverge. In case of frenquetists statistics you obtain confidence intervals and p-values based on appropriate statistics and degrees of freedom, whereas in case of Bayesian inference you obtain credible/compatibility intervals and can use posterior distribution for individual parameters to compute probability that they are strictly positive, negative, concentrated within a certain region around zero, etc.

These measures are conceptually different but tend to be interpreted similarly and mostly from Bayesian perspective. I think it is a good idea to compute and report all of them. Typically, they would be close, making you more certain about the results. More importantly, whenever they diverge it serves as a warning to investigate the case and what can cause this difference.

7.6 Model comparison via information criteria (both)

Both approaches use information criteria to compare models with Akaike and Bayesian/Schwarz information criteria being developed specifically for the case of flat priors of frenquetist models. Here, Bayesian approach holds an advantage as the full posterior allows for more elaborate information criteria such as DIC, WAIC, or LOO. Still the core idea and the interpretation of the comparison results are the same.

7.7 Generating predictions (both)

You generate predictions using the model definition which is the same for both approach. Hence, you are going to get same predictions, at least for the mean. As the two approach differ in how they characterize uncertainty, the uncertainty of predictions will be different but, typically, comparable.

7.8 Conclusions

As you can see, from *practical* point of view, apart from optional Bayesian priors and different ways to quantify the uncertainty of estimates, the two approaches are the same. They require you making same decisions and the results are interpreted the same way. This lack of difference becomes even more apparent when you use software packages for running your statistical models. E.g., the way you specify your model in `lme4` (frenquetist) and `brms` (Bayesian) is very much the same to the point that in most cases you only need to change the function name (`lmer` to `brm` or vice versa) and leave the rest the same.

Thus, the point is that you should not be choosing between studying or doing frenquetists or Bayesian statistic. I feel more comfortable with Bayesian, mostly because it makes it easier interpret statistical significance. However, my typical approach is to start with frenquetist statistics (fast, good for shooting from a hip), once I am certain about my decisions (likelihood, model) I re-impliement the same model in Bayesian using informative priors and see whether results match (with reason). Then, I report both sets of inferences. This costs me remarkably little time precisely because there is so little difference between the two approaches from practical point of view!

7.9 Take home message

We are not studying something completely different! We are merely approaching it from an unusual angle that leads to deeper understanding and interesting insights in the long run.

Chapter 8

Mixtures

8.1 Beta Binomial

Beta binomial is defined as a product of binomial and beta distributions.

$$BetaBinomial(k|N, p, \theta) = Binomial(k|N, p) \cdot Beta(p|\beta_1, \beta_2),$$

where k is number of successes (e.g., "heads" in a coin toss), N is total number of trials/draws, and p is the probability of success), and β_1 and β_2 determine the shape of the beta distribution. The book uses reparametrized version of the beta distribution, sometimes called *beta proportion*:

$$BetaBinomial(k|N, p, \theta) = Binomial(k|N, p) \cdot Beta(p|p, \theta),$$

where θ is precision parameter. p and θ can be computed from β_1 and β_2 as

$$p = \frac{\beta_1}{\beta_1 + \beta_2} \theta = \beta_1 + \beta_2$$

The latter form makes it more intuitive but if you look at the code of `dbetabinom()`, you will see that you can use `shape1` and `shape2` parameters instead of `probe` and `theta`.

Recall the (unnormalized) Bayes rule (p is probability, y is an outcome, ... parameters of the prior distribution):

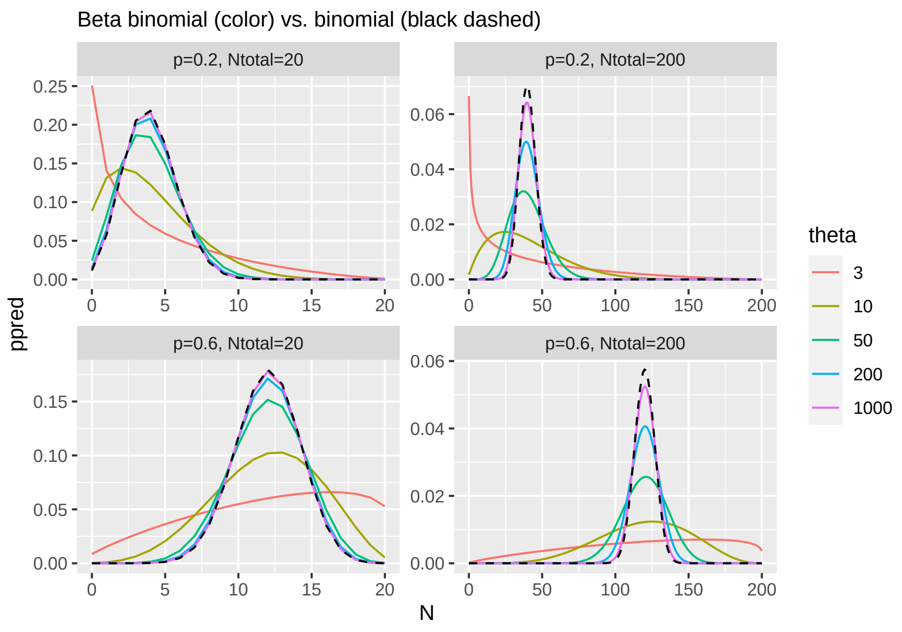
$$Pr(p|y) = Pr(y|p) \cdot Pr(p|...)$$

Examine the formula again and you can see that you can think about beta binomial as a posterior distribution for a binomial likelihood with the beta distribution as prior for parameter p' of the binomial distribution:

$$BetaBinomial(N, p, \theta|k) = Binomial(k|N, p) \cdot Beta(p|p_{mode}, \theta)$$

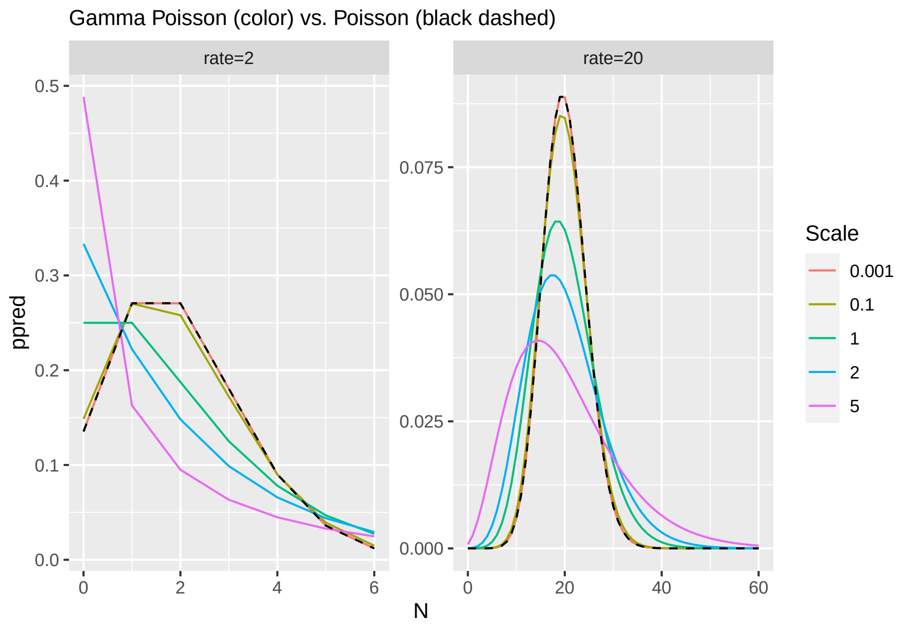
Thus, beta binomial is a combination of *all* binomial distributions weighted by a beta distribution that has its mode at p_{mode} and its width is determined by θ. In other words, when we use binomial distribution alone, we state that we can *compute* the probability directly as $p =$ some linear model. Here, we state that our knowledge is incomplete and, *at best*, we can predict mode of the beta distribution from which this probability comes from and we let data determine variance/precision (θ) of that distribution. Thus, our posterior will reflect *two* uncertainties based on two loss functions: one about the number of observed events (many counts are compatible with a given p but with different probabilities), as for the binomial, plus another one about p itself (many values of p are compatible with given p_{mode} and θ). This allows model to compute a trade-off by considering values of p that are less likely from prior point of view (they are away from p_{mode}) but that result in higher probability of k given that chosen p. We will see the same trick again later in the book, when we will use it to incorporate uncertainty about measured value (i.e., at best, we can say that the actual observed value comes from a distribution with that mean and standard deviation).

In practical terms, this means that parameter θ controls the width of the distribution (see plots below). As θ approaches positive infinity, our prior uncertainty about p is reduced to zero, which means that we now consider only one binomial distribution, where $p' = p$, which is equivalent to the simple binomial distribution. Thus, beta binomial *at most* is as narrow as the binomial distribution.



8.2 Negative binomial, a.k.a. Gamma Poisson

The idea is the same: We do not have enough information to compute the rate of events, so instead, we compute the mean of the Gamma distribution rates come from and let data determine its variance (scale). Again, in practical terms this means that for the smallest scale our uncertainty about the rate is minimal and distribution matches the Poisson processes with a fixed rate. Any increase in uncertainty (larger values for scale parameter), mean broader distribution that is capable to account for more extreme values.

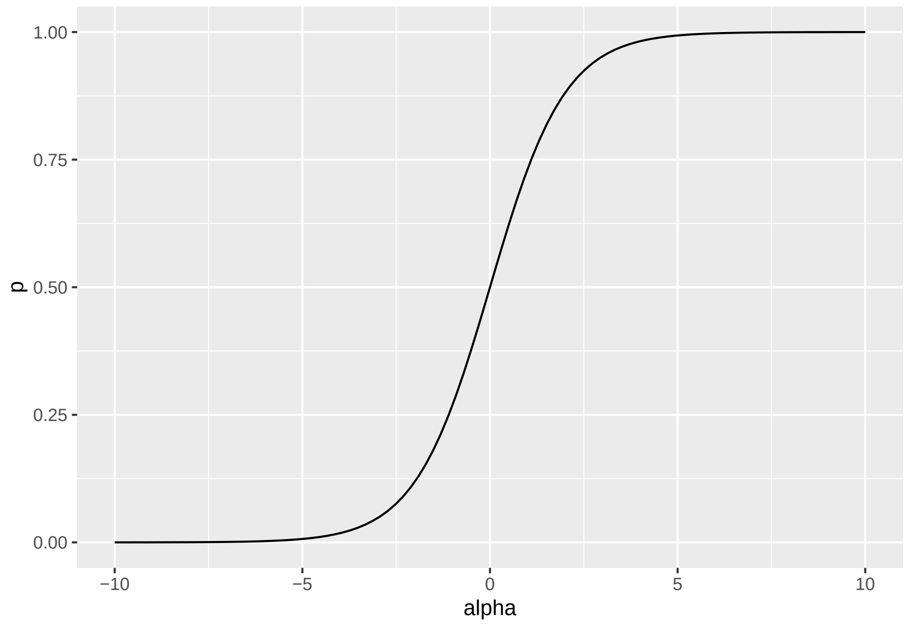


8.3 Ordered categorical

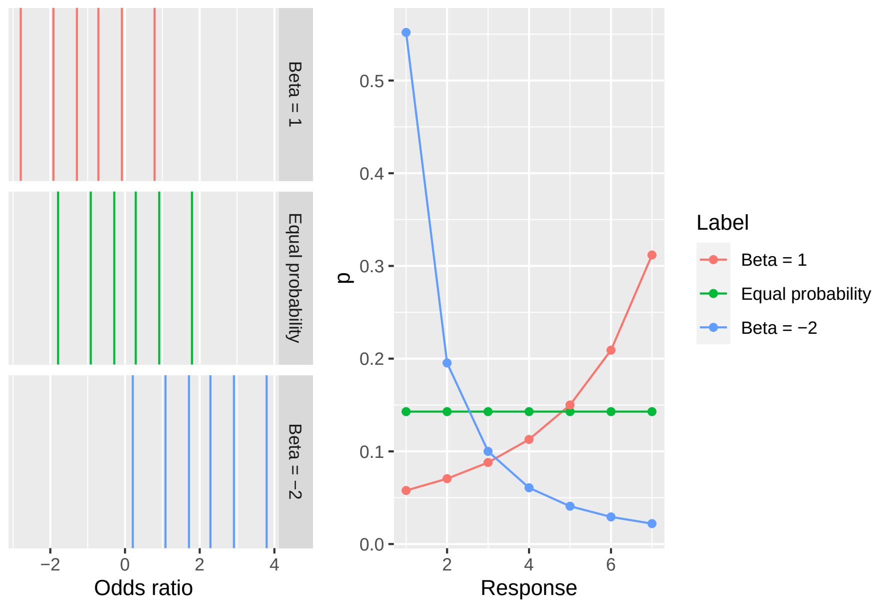
From log odds to logit link.

$$log(\frac{Pr(y_i \leq k)}{1 - Pr(y_i \leq k)}) = \alpha_k \frac{Pr(y_i \leq k)}{1 - Pr(y_i \leq k)} = e^{\alpha_k} Pr(y_i \leq k) = e^{\alpha_k} \cdot (1 - Pr(y_i \leq k)) Pr(y_i \leq k) \cdot (1 + e^{\alpha_k}) = e^{\alpha_k} Pr(y_i \leq k$$

```
df_p <-
  tibble(alpha = seq(-10, 10, length.out=100)) %>%
  mutate(p = exp(alpha) / (1 + exp(alpha)))

ggplot(df_p, aes(x=alpha, y=p)) +
  geom_line()
```



```
equal_probability <- log(((1:6) / 7) /  (1 - (1:6) / 7))

df_ord_cat <-
  tibble(Response = 1:7,
         p = rethinking::dordlogit(1:7, 0, equal_probability),
         Label = "Equal probability") %>%
  bind_rows(tibble(Response = 1:7,
         p = rethinking::dordlogit(1:7, 0, equal_probability - 1),
         Label = "Beta = 1")) %>%
  bind_rows(tibble(Response = 1:7,
         p = rethinking::dordlogit(1:7, 0, equal_probability + 2),
         Label = "Beta = -2")) %>%
  mutate(Label = factor(Label, levels = c("Beta = 1", "Equal probability", "Beta = -2"

df_cuts <-
  tibble(Response = 1:6,
         K = equal_probability,
         Label = "Equal probability") %>%
  bind_rows(tibble(Response = 1:6,
         K = equal_probability - 1,
         Label = "Beta = 1")) %>%
  bind_rows(tibble(Response = 1:6,
         K = equal_probability + 2,
         Label = "Beta = -2")) %>%
  mutate(Label = factor(Label, levels = c("Beta = 1", "Equal probability", "Beta = -2")))


cuts_plot <-
  ggplot(data=df_cuts,) +
  geom_vline(aes(xintercept = K, color=Label), show.legend = FALSE) +
  facet_grid(Label~.) +
  scale_x_continuous("Odds ratio")

prob_plot <-
  ggplot(data=df_ord_cat, aes(x=Response, y=p, color=Label)) +
  geom_point() +
  geom_line()

cuts_plot + prob_plot
```


Beta = 1
Equal probability
Beta = −2
Odds ratio
p
Response
Label
Beta = 1
Equal probability
Beta = −2