Statistical Rethinking Notes

Tyler Burch

Chapter 1 - The Golem of Prague

1.1 - Statistical Golems

"Golem" analogy to clay robots that just do what they're told without thinking, leading to carelessness. Common in statistics too, consider flowcharts with many different tests at the end. Need a more generalized approach (think engineering - don't start with building bridges, start with physics), rethinking inference as a set of strategies, not a set of tools.

1.2 - Statistical Rethinking

Many have approach that the objective of inference is to test null hypotheses, but science isn't described by falsification standard.

Hypotheses are not models

- Models can correspond to multiple hypotheses, multiple hypotheses to a model
- All models are false, so what does it mean to falsify a model?

Progression: - Hypotheses - Statements - Process Models - Causal structure model that formalize cause/effect relationships - Statistical Model - Don't embody causal relationships; express associations among variables

How to get from statistical model back to process model? Derive a expected frequency distribution of some quantity - "statistic" from the causal model. Histogram, for example. Unfortunately, other models can imply the same statistical model, reinforcing the many-to-many relationships:

- Any statistical model can correspond to many process models
- Any hypothesis can correspond to multiple process models
- Any statistical model can correspond to multiple hypotheses

So what to do? If you have multiple process models that all make similar predictions, then you search for a description where the processes look different.

Measurement matters

Logic of falsification is have hypothesis, look for observation. If not found, then hypothesis is false. If we formalize H_0 to "All swans are white", no number of observations can prove it, one observation disproves - powerful but prone to observation errors, and quantitative hypotheses have degrees of existence

Observation Error - Doubted observations, measurement/instrumentation error

Continuous Hypotheses - Not trying to disprove, but understand a distribution

Falsification is consensual

Communities argue toward consensus about the meaning of evidence, can be messy

1.3 - Tools for Golem Engineering

If falsification isn't the way, we can model. Models then can be made to testing procedures, as well as designs, forecasts, and arguments. This text focuses on several tools: - Bayesian Data Analysis - Model comparison - Multilevel models - Graphical models

Bayesian data analysis

Takes question in form of a model and uses logic to produce an answer in the form of probability distributions. Literally just counting how the data may look according to our assumptions. Compare to frequentest, which is defined by the frequencies of events in large samples, based on imaginary data resampling.

Bayesian approaches treat randomness as a property of information, not of the world.

Model comparison and prediction

If multiple models, how to choose? Cross validation and information criteria.

Help in 3 ways: provide expectations of accuracy, give an estimate of overfitting, help spot influential observations

Multilevel Models

Parameters all the way down - parameters support inference. Multiple levels of uncertainty feed into the next, a multilevel (hierarchical, random effects, varying effects, mixed effects) model. Help us with overfitting, by exploiting partial pooling, which can be used to adjust estimates for repeat sampling, imbalance, variation, and to avoid averaging.

Diverse models turn out to be multilevel: models for missing data (imputation), measurement, factor analysis, some time series models, and types of spatial and network regression are all special applications of multilevel.

Fitting and interpreting multilevel models can be harder than traditional

Graphical Causal Models

Statistical models are association engines, detect (not infer!) association between cause and effect. Due to overfitting though, causally incorrect models can make better predictions than causally correct ones, can't focus on just prediction.

So instead, causal model that can be used to design one or more statistical models for causal identification. "Graphical Causal Model" represents a causal hypothesis, the most simple being a Directed Acyclic Graph (DAG).

1.4 - Summary

4 parts to book

1. Bayesian inference (ch 2,3)

2. Multiple linear regression (ch 4-9)
3. Generalized linear models, with MCMC and maximum entropy (ch 9-12)
4. Multilevel models, specialized models (ch 13-17)

Chapter 2 - Small Worlds and Large Worlds

Christopher Columbus thought the world was 10,000 km smaller than it is, his prediction made him think he could have enough supplies to circle it - contrast between model and reality.

Small world - self-contained logical world of model

Large World - larger context where model is deployed

2.1 - The Garden of Forking Data

Bayesian inference is really just counting and comparing possibilities. Cannot guarantee a correct answer, but can guarantee the best possible answer given information provided.

Use example of blue/white marbles out of bag, examine each path of pulls to exclude different "paths" through the data. Introduce a "prior," if all routes are equally likely, can base prior on the number of counts to an outcome. Update prior after a pull, or when new information is added (e.g. the company says blue is rare). The example outlines the following terms:

1. Conjectured proportion of marbles (p in example) is a "parameter" value
2. Number of ways a value can produce data is a "likelihood" - derived by enumerating all data sequences that could have happened and eliminate those inconsistent with data
3. Prior plausibility of p is the "prior probability"
4. Updated plausibility of p is the "posterior probability"

2.2 - Building a model

Design loop with 3 steps:

1. Data story: motivate model by narrating how the data might arise
2. Update: educate model by feeding it data
3. Evaluate: All models require supervision, possibly leading to model revision

Example in chapter - calculate the amount of water on earth by throwing globe up in the air, see if right thumb is on water or land.

Data Story

May be *descriptive*, specifying associations to predict outcomes given observations; may be *causal*, a theory of how some events produce others. Generally causal stories are easily descriptive, descriptive stories may be hard to be causal. Can motivate by explaining how each piece of data is born. The value of the story is to more strongly define hypotheses and resolve ambiguities.

Example: true proportion of water is p, single toss has p chance of producing water W and $1 - p$ of land L, each toss is independent.

Bayesian Updating

Model begins with one set of plausibilities assigned to each possibility. As data is collected, posteriors are produced.

Example: Give uniform prior. First case lands on water, $p = 0$ is excluded, since there is no longer a possibility of no water. Continue tossing and distribution shifts and closes more and more.

One benefit of Bayesian approach is that estimates are valid for any sample size. Of course better with more data.

Evaluate

Because of differences between the model and real world, no guarantee of large world performance. Keep in mind two principles. Model's certainty is no guarantee the model is good - this is telling you that, given this model, plausible values are in some range. Second, it's important to supervise and critique the work - in the example, order of tosses shouldn't change final curve, but may indirectly affect it because data depends on order, so check on data it does not know about.

2.3 - Components of the Model

Variables

Symbols that can take different values - for globe example: target p (proportion of water) cannot be observed. Unobserved are called "parameters." Other variables are count of water W and count of land L, these are observed.

Definitions

In defining, we build a model relating variables to one another. For each value of unobserved, need to define the number of ways/probability that the values of each observed could arise. For each unobserved also need a prior.

Using example:

Observed Variables - For specific p, need to define how plausible combinations of W and L would be, using a mathematical function called a likelihood. In this case, if tosses are independent and probability are the same, we use binomial distribution.

$$\Pr(W, L|p) = \frac{(W+L)!}{W!L!} p^W (1-p)^L$$

In R:

```
dbinom( 6 , size=9 , prob=0.5)
```

```
## [1] 0.1640625
```

This gives the relative number of ways to get 6 water results for $p = 0.5$ after 9 total tosses ($N = W + L = 9$).

Unobserved Variables - Distributions for observed variables typically have own variables; p not observed, so a parameter. Many common data questions are answered directly by parameters, e.g. average difference between groups, association strength, covariate dependence, variation. For every parameter, you must define a prior.

Some schools of thought that emphasize choosing priors on personal belief, known as "subjective Bayesian." If you don't have a strong argument for any prior, try different ones.

Model is born

Can summarize as the following:

$$W \sim \text{Binomial}(N, p)$$
$$p \sim \text{Uniform}(0, 1)$$

Telling us W is binomial, and p is flat over the range 0 to 1.

2.4 - Making the Model Go

Once you have named all the variables, definitions, update prior to posterior - the relative plausibility of parameter values conditional on fdata and model, for our example $\Pr(p|W, L)$.

Bayes Theorem

Mathematical definition of posterior arises from Bayes. Joint probability $\Pr(W, L, p) = \Pr(W, L|p)\Pr(p)$. The right side can also be reversed $\Pr(p|W, L)\Pr(W, L)$, which can be solved to

$$\Pr(p|W, L) = \frac{\Pr(W, L|p)\Pr(p)}{\Pr(W, L)}$$

This is Bayes theorem but can really just be said

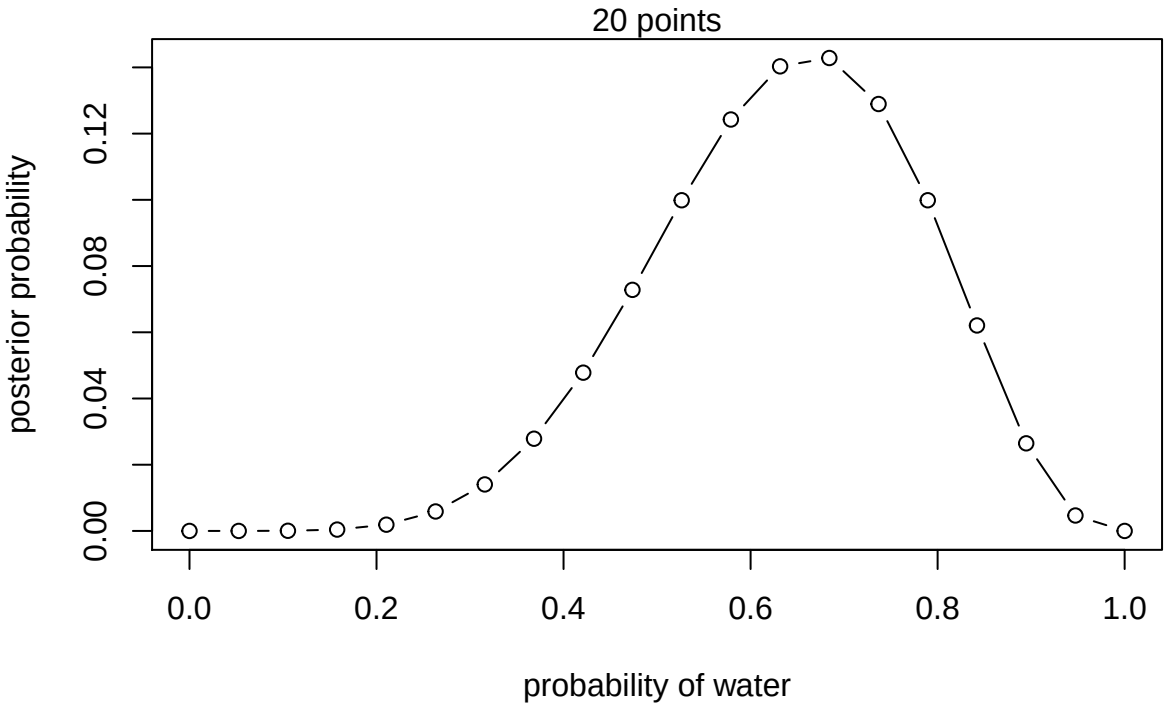
$$\text{Posterior} = \frac{\text{Probability of data} \times \text{Prior}}{\text{Average probability of the data}}$$

Denominator is averaged over the prior - meant to standardize the posterior to make the sum one.

Three different numerical techniques for computing posterior: grid approximation, quadratic approximation, MCMC.

Grid Approximation - Consider a finite number of values, compute posterior by multiplying prior by likelihood, repeat until getting an approximate picture of the posterior. Mostly a pedagogical tool, since not typically practical.

```
# define grid
p_grid <- seq( from=0 , to=1 , length.out=20 )
# define prior
prior <- rep( 1 , 20 )
# compute likelihood at each value in grid
likelihood <- dbinom( 6 , size=9 , prob=p_grid )
# compute product of likelihood and prior
unstd.posterior <- likelihood * prior
# standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
plot(p_grid , posterior , type="b" ,
     xlab="probability of water" , ylab="posterior probability")
mtext( "20 points" )
```



Quadratic Approximation - More parameters make grid approximations tough (N^p for p parameters and N data points). Use quadratic approximation when the region near the peak of the posterior will be Gaussian in shape, easy because it can be described by just mean and variance.

1. Find posterior mode
2. Estimate curvature, either analytically or computationally.

For this book use `quap` from rethinking programming package

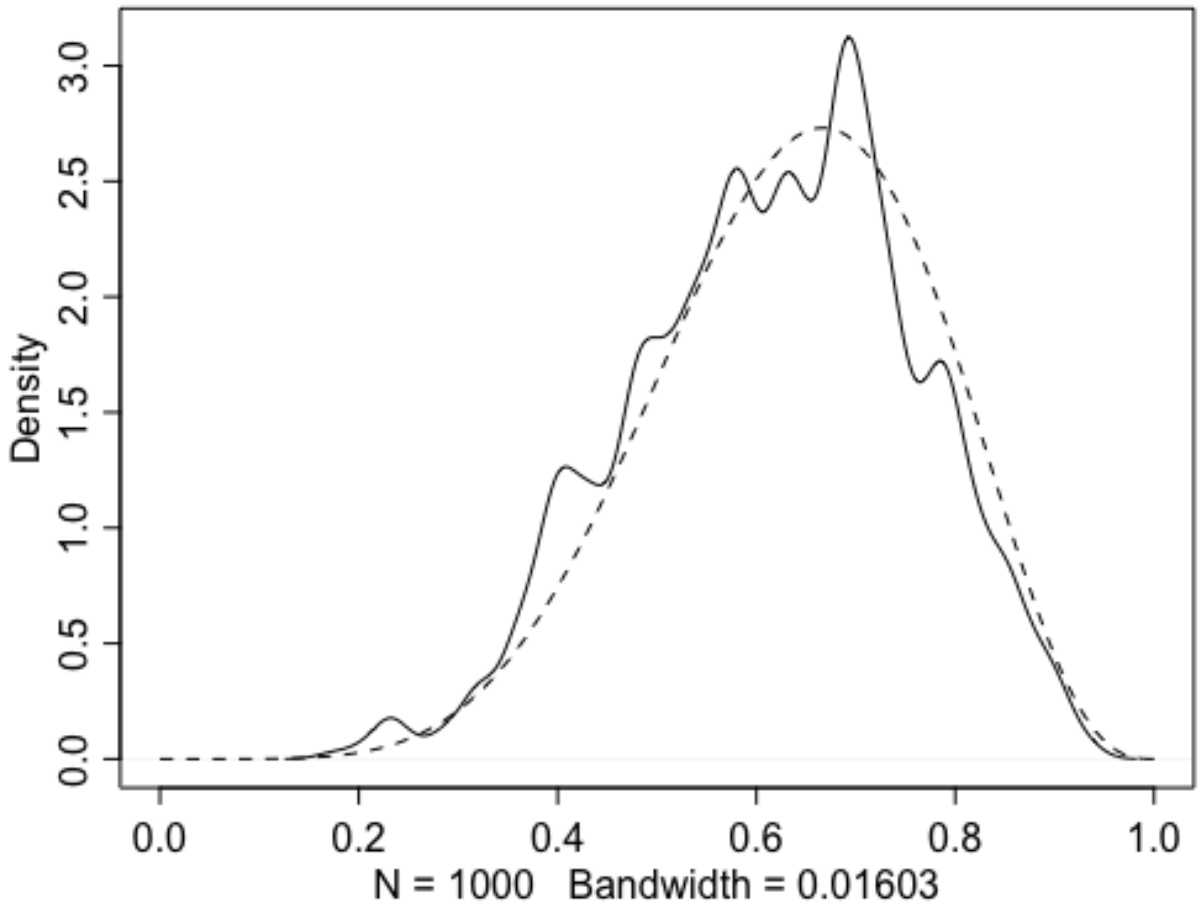
```
library(rethinking)
globe.qa <- quap(
  alist(
    W ~ dbinom (W+L, p), # Binomial
    p ~ dunif(0,1) # Uniform
  ), data = list(W=6, L=3)
)
precis(globe.qa)
```

```
##        mean        sd      5.5%     94.5%
## p 0.6666664 0.1571338 0.4155362 0.9177966
```

The quadratic approximation is often equivalent to a Maximum Likelihood Estimate and its standard error.

note - quadratic is solved by computing the Hessian, a square matrix of second derivatives of the log of posterior probability wrt parameters. Derivatives sufficient to describe a Gaussian. Std is typically computed from Hessian, which can occasionally cause problems in computation.

Markov chain Monte Carlo (MCMC) - Many models, like multilevel/mixed-effects don't work for grid approximation (many parameters) or quadratic (non-Gaussian posterior). Function to maximize isn't known, computed in pieces via MCMC. Rather than computing or approximating posterior, MCMC draws samples, a collection of parameter values.



2.5 - Summary

Looked at conceptual ideas in Bayesian data analysis. Models are composite of variables and distributional definitions, fit to data using numerical techniques

Chapter 3 - Sampling the Imaginary

Given example of test to check for vapmirism 0 highly accurate, but makes false positives at the rate of $\Pr(\text{positive test}|\text{mortal}) = 0.01$. Vampirism is rare, 0.1% of the population.

To solve given a positive test the likelihood that they are a vampire,

$$\Pr(\text{positive}) = \Pr(\text{positive}|\text{vampire})\Pr(\text{vampire}) + \Pr(\text{positive}|\text{mortal})(1 - \Pr(\text{vampire}))$$

and

$$\Pr(\text{vampire}|\text{positive}) = \frac{\Pr(\text{positive}|\text{vampire})\Pr(\text{vampire})}{\Pr(\text{positive})}$$

```
Pr_Positive_Vampire <- 0.95
Pr_Positive_Mortal <- 0.01
Pr_Vampire <- 0.001
Pr_Positive <- Pr_Positive_Vampire * Pr_Vampire +
  Pr_Positive_Mortal * ( 1 - Pr_Vampire )
( Pr_Vampire_Positive <- Pr_Positive_Vampire*Pr_Vampire / Pr_Positive )
```

```
## [1] 0.08683729
```

Gives an 8.6% chance they're actually a vampire, despite positive test. This is a canonical problem, broader in statistics - despite using Bayes' theorem, not uniquely Bayesian. Reframe using *natural frequencies*:

1. In a population of 100,000 people, 100 are vampires
2. Of 100, 95 test positive
3. Of 99,900 mortals, 999 test positive

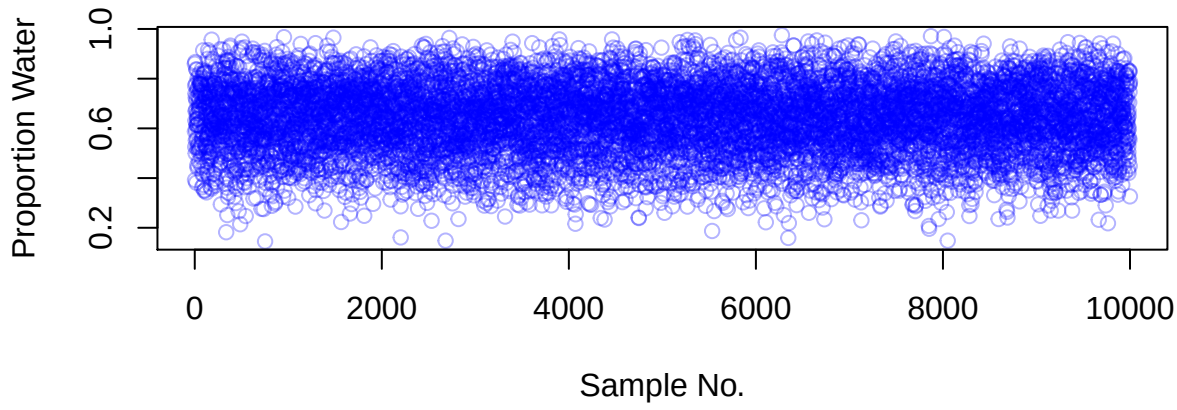
$$\begin{aligned}\Pr(\text{vampire}|\text{positive}) &= \frac{\text{true positives}}{\text{all positives}} \\ &= \frac{95}{1094} \approx 0.087\end{aligned}$$

Chapter is focused on working from samples from posterior, to make sense of model output.

3.1 - Sampling from a grid approximate posterior

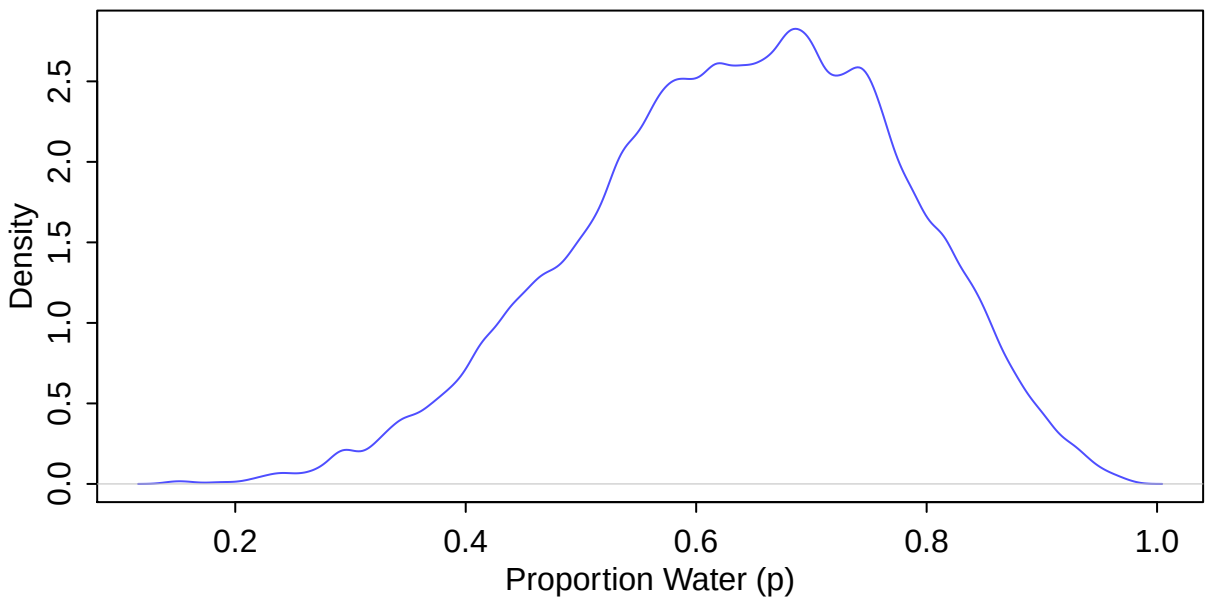
Rerun code for grid approximation posterior, then draw 10,000 samples.

```
samples <- sample(p_grid, prob=posterior, size=1e4, replace=TRUE)
plot(samples,col=alpha("blue",0.3), xlab = "Sample No.", ylab = "Proportion Water")
```



Can also draw a density plot:

```
dens(samples, col=alpha("blue",0.7),
     xlab = "Proportion Water (p)", ylab = "Density")
```



3.2 - Sampling to Summarize

Can ask many questions using your posterior: How much lies below a parameter value, or between two parameter values? Which parameter value marks the lower X%? What range contains most the posterior probability? Which parameter values are most likely?

Intervals of defined boundaries -

Address probability that proportion of water is less than 0.5. Directly from grid:

```
sum( posterior[ p_grid < 0.5 ] )
```

```
## [1] 0.1718746
```

However, if not using grid, you can use samples and get a nearly identical result:

```
sum( samples < 0.5 ) / 1e4
```

```
## [1] 0.1662
```

Intervals of Defined Mass -

"Confidence intervals" commonly used, but we work with "credible interval" or "compatibility interval." To get the 80% "percentile interval" (PI):

```
quantile( samples , c( 0.1 , 0.9 ) )
```

```
##       10%       90%
## 0.4504505 0.8128128
```

`PI` function in rethinking package does this as well. Also `HPDI` is the "Highest posterior density interval," the narrowest interval containing specified probability mass. Generally this interval best represents parameter values consistent with the data. Note HPDI is more computationally intensive and suffers from variance on number of samples drawn.

Point Estimates -

Given the entire posterior, what number to report? Can do *maximum a posteriori* (MAP) by taking the mode. Other point estimates (mean, median) can also work, but often worse in terms of loss function.

3.3 - Sampling to Simulate Prediction

Useful for:

1. Model design - sampling from the prior can help understand implications
2. Model checking - see if the fit worked correctly
3. Software validation - does the model fitting software work alright? check by recovering parameter values
4. Research design - simulate observations from hypothesis, can evaluate weather research design is effective, *power analysis*.
5. Forecasting - simulate new predictions for the future

Dummy data

Bayesian models are always generative, capable of simulating predictions. For the globe example

```
dbinom( 0:2 , size=2 , prob=0.7 )
```

```
## [1] 0.09 0.42 0.49
```

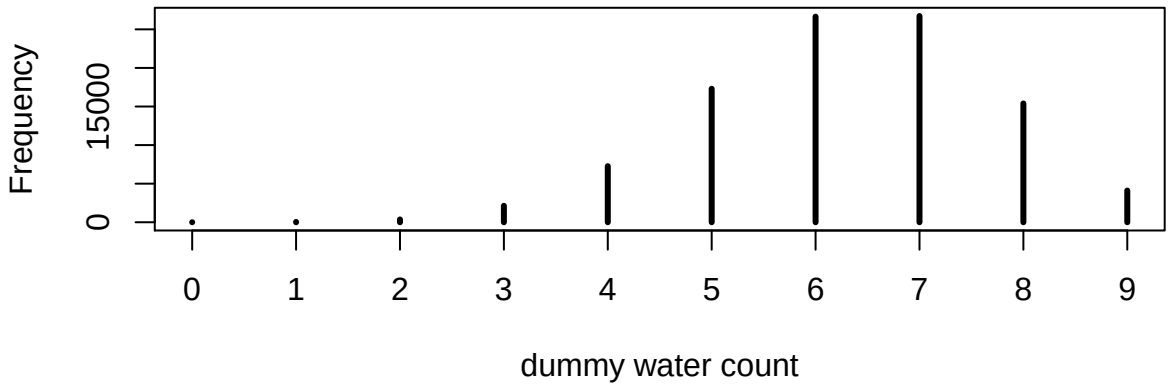
simulates 0, 1, 2 "water" results; 9% chance of not landing on water at all. Can simulate many dummy observations:

```
trials <-1e5
dummy_w <- rbinom( trials , size=2 , prob=0.7 ) # r for random, 10
table(dummy_w)/trials
```

```
## dummy_w
##       0       1       2
## 0.08978 0.42176 0.48846
```

which are close to analytical solution. Also can plot to make sure it looks binomial, now using 9 tosses

```
trials <-1e5
dummy_w <- rbinom( trials , size=9 , prob=0.7 )
simplehist( dummy_w , xlab="dummy water count" )
```



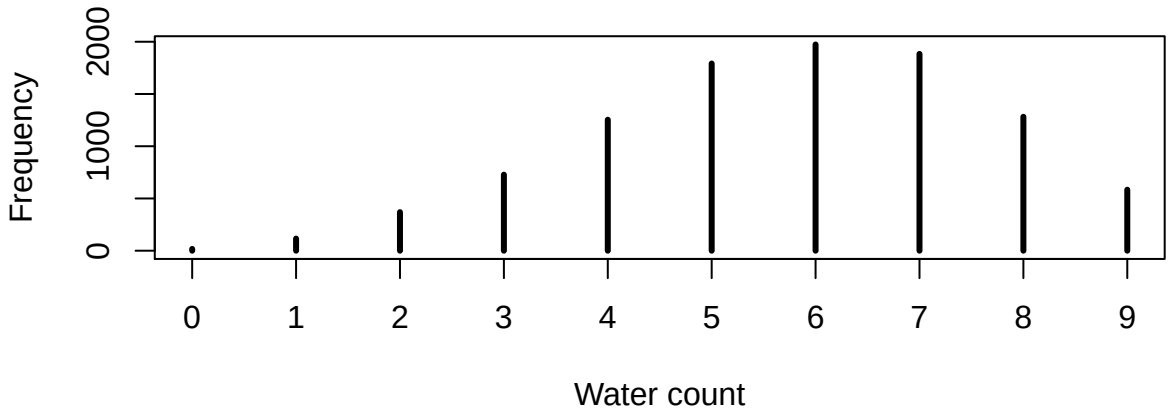
Model Checking

1. Ensure the fitting worked correctly
2. Evaluate the adequacy of the model for a purpose

Don't test whether assumptions are "true," assess exactly how it fails to describe the data. Basic model checks using samples from full posterior (not point estimates!).

- Observation uncertainty: sample variation - globe tossing, even if you know p exactly, you won't know the next globe toss results
- Parameter uncertainty: posterior distribution embodies this, will interact with sampling variation. Want to propagate this as evaluating predictions; computing sampling distribution at each value of p, averaging together, gets a "posterior predictive distribution

```
 w <- rbinom( 1e4 , size=9 , prob=samples )
simplehist( w , xlab="Water count" )
```



Here, for each posterior sample, a random binomial dataset is created. Wide spread, but arises from the binomial process itself. Can consider other metrics, like the longest consecutive Water results (mode=3, obs=3) or number of switches between water/land (mode=4, obs=6).

3.4 - Summary

Given basic procedures for manipulating posterior distributions, can be used for intervals, point estimates, posterior predictive checks, simulations. Encapsulate uncertainty about parameters with uncertainty about outcomes.

Chapter 4 - Geocentric Models

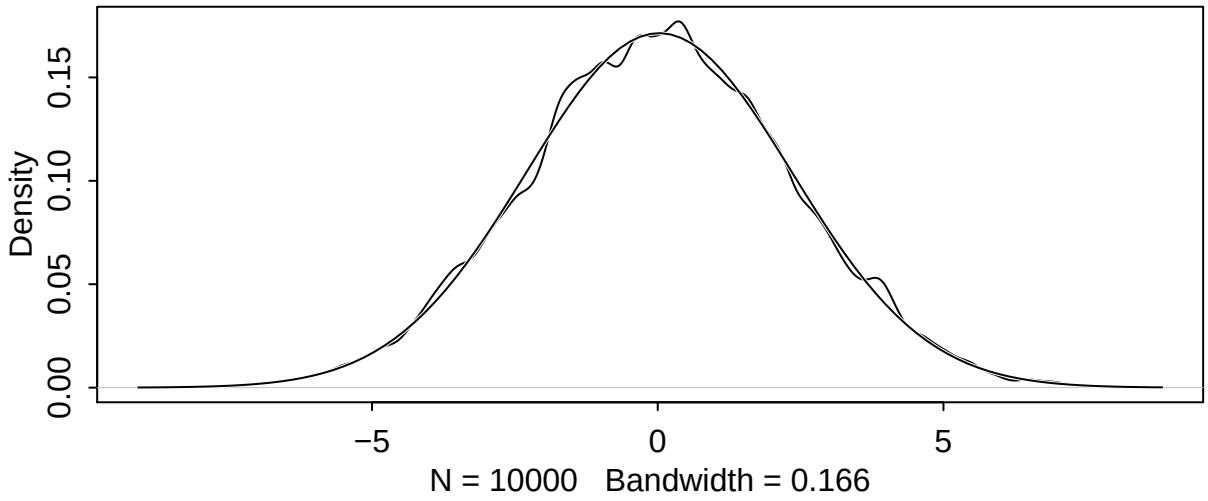
Ptolemy came up with an extremely accurate geocentric model of our solar system in 1st century, and used for over 1000 years. Based on epicycle - circles on circles, which generalize to a Fourier series with enough circles. Comparison to linear regression - don't want to read too literally, but useful.

4.1 Why normal distributions are normal

Normal by Addition

Introduce from a random walk perspective:

```
pos <- replicate( 10000 , sum( runif(16,-1,1) ) )
dens(pos,  norm.comp=TRUE)
```

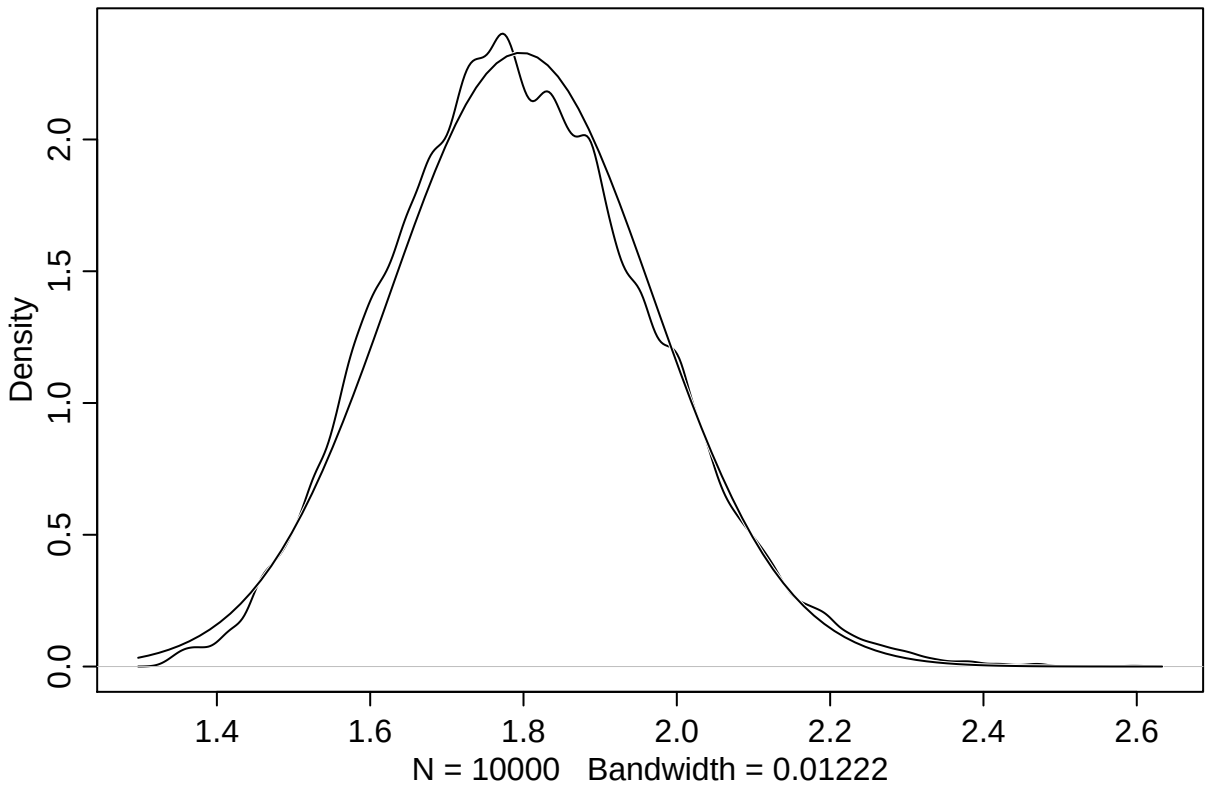


Gaussian structure ultimately emerges - processes that add together random values result in Gaussians.

Normal by Multiplication

Consider an organism with growth rate influenced by local interactions; multiplicative effect. Sample growth rates between 1.0 and 1.1:

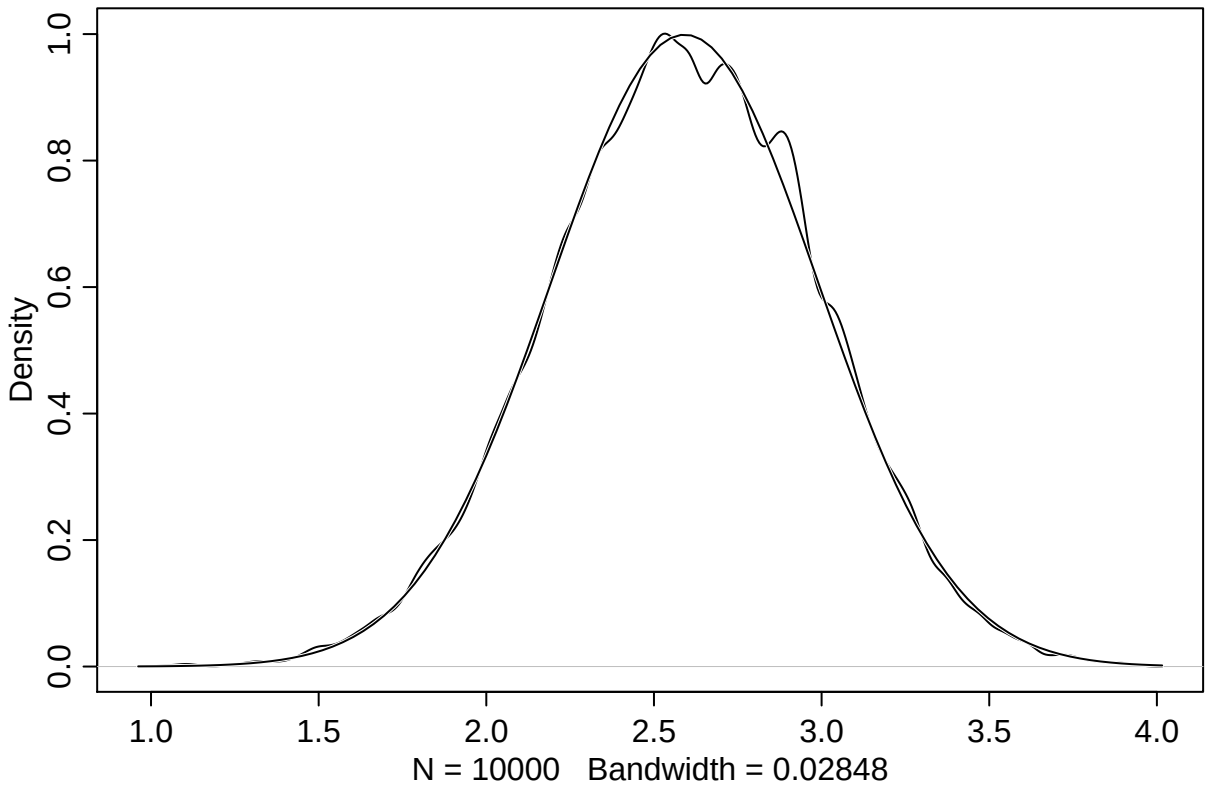
```
growth <- replicate( 10000 , prod( 1 + runif(12,0,0.1) ) )
dens( growth , norm.comp=TRUE )
```



Normal by log-multiplication

Large deviates multiplied don't produce Gaussians, but do produce Gaussians at log scale

```
log.big <- replicate( 10000 , log(prod(1 + runif(12,0,0.5))) )
dens(log.big, norm.comp=TRUE )
```



This is because adding logs is the equivalent to multiplying original numbers.

Using Gaussian Distributions

Rest of chapter will be using Gaussians as a skeleton; 2 justifications

- Ontological - World is full of approximate Gaussians. Gaussian is a member of the "exponential family," all of which are important for science and seen in the natural world.
- Epistemological - Represents a state of ignorance: all we know or are willing to say is mean and variance. Premised on Information theory and maximum entropy (chapters 7 and 10).

4.2 - A language for describing models

1. Recognize a set of variables to understand. Observable ones are data, unobservable are parameters.
2. For each variable define in terms of other variables or a probability distribution.
3. The combination of variables and probability distributions defines a *joint generative model*.

Then summarize the model, e.g.

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \beta x_i \\
\beta &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Exponential}(1) \\
x_i &\sim \text{Normal}(0, 1)
\end{aligned}
$$

Immense flexibility with this - don't need to worry about conditions like homoscedasticity since in the model definition, natural ways to change assumptions so not stuck in fixed model type.

Re-describing globe tossing

$$W \sim \text{Binomial}(N, p)$$
$$p \sim \text{Uniform}(0, 1)$$

Said: "Count W is distributed binomially, sample size N, probability p. Prior for p is uniform between 0 and 1.

Both are stochastic, which is indicated by $\sim$, and implies not known with certainty, just probabilistic.

4.3 - Gaussian model of height

Building a linear regression model, consider all Gaussians, so posterior is a distribution of Gaussians distributions

Data

1960s partial census data from foraging population of !Kung San

```
library(rethinking)
data(Howell1)
d<- Howell1 #dataframe object
d2 <- d[ d$age >=18 , ] # Create new, just adult one
# precis(d2)  ### run this - but doesn't compile on r-markdown
```

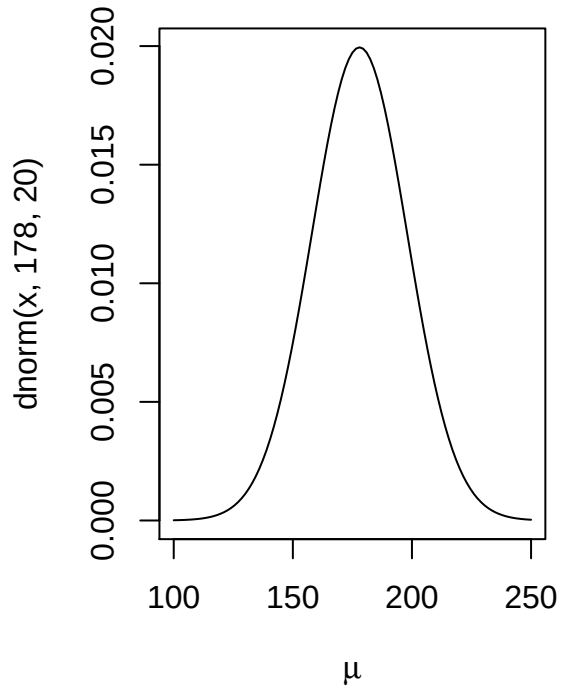
The model

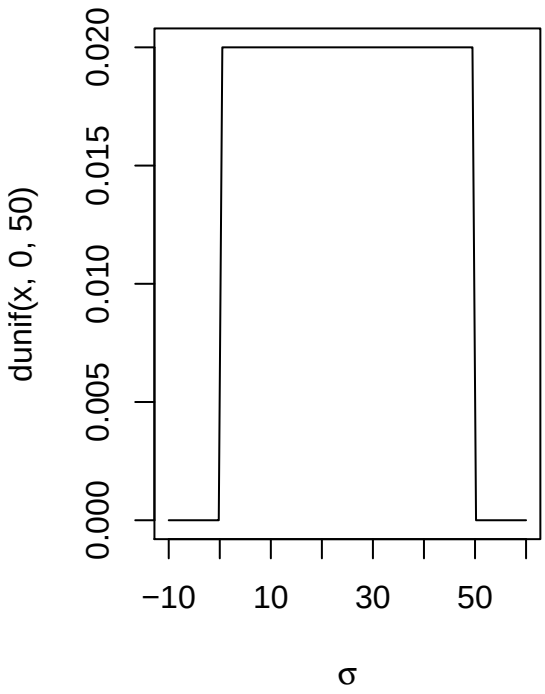
Adult heights from a single population are approximately normal, so model using Gaussian, but which one? Model is:

$$h_i \sim \text{Normal}(\mu, \sigma)$$
$$\mu \sim \text{Normal}(178, 20)$$
$$\sigma \sim \text{Uniform}(0, 50)$$

Estimate on mean of μ comes from author height being 178 cm, so 178 ± 40 seems reasonable. Sigma a 0 because standard deviations need to be positive. Plotting priors:

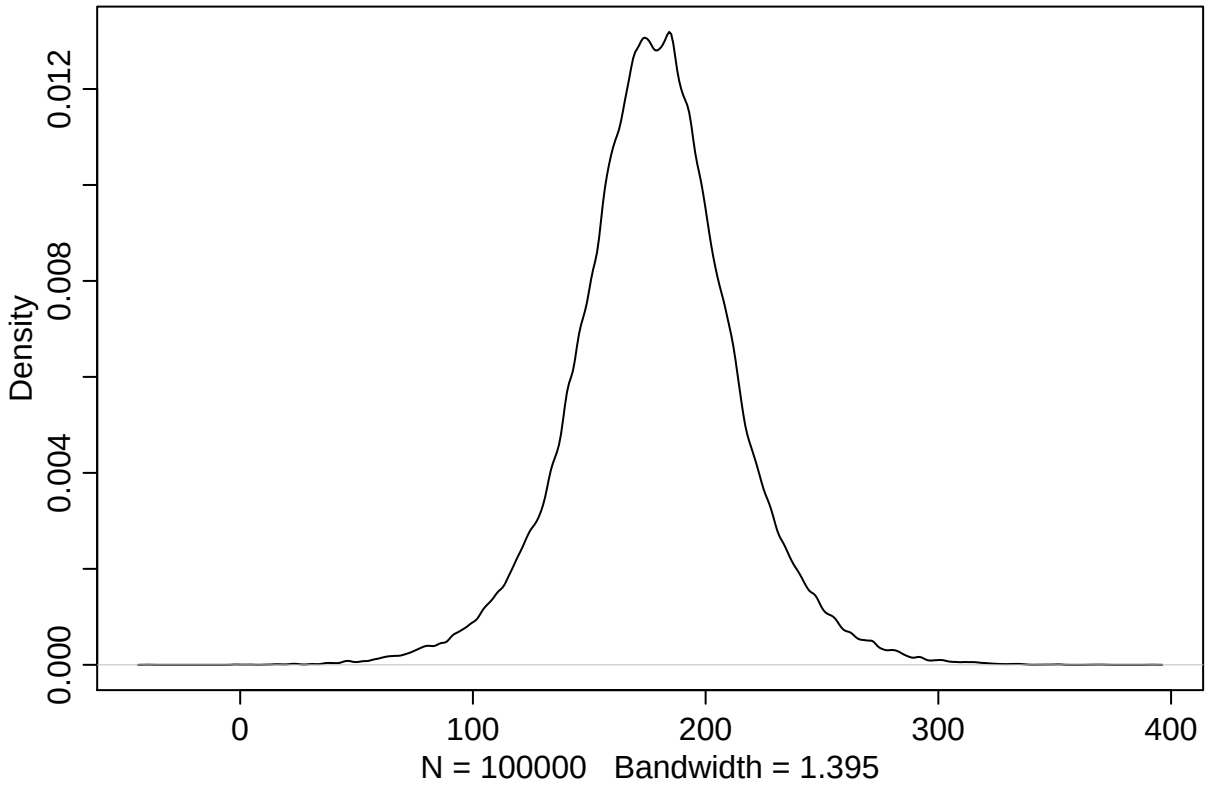
```
par(mfrow=c(1,2))
curve( dnorm( x , 178 , 20 ) , from=100 , to=250, xlab=expression(mu))
curve( dunif( x , 0 , 50 ) , from=-10 , to=60, xlab=expression(sigma))
```





Prior predictive simulation should be run, to see implications of prior choices.

```
sample_mu <- rnorm( 1e4 , 178 , 20 )
sample_sigma <- runif( 1e4 , 0 , 50 )
prior_h <- rnorm( 1e5 , sample_mu , sample_sigma )
dens( prior_h )
```

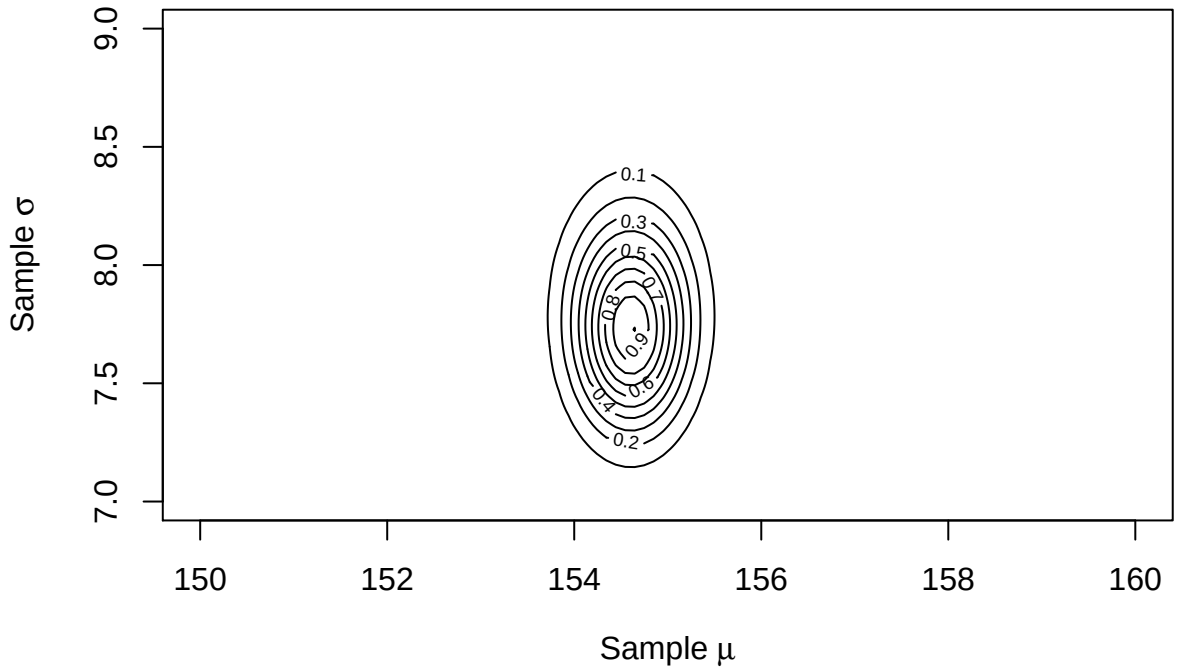


You could consider a less informative prior, e.g. $\mu \sim \text{Normal}(178, 100)$, but this will give you odd things like negative heights, or people taller than the record tallest person - this is why these tests are useful.

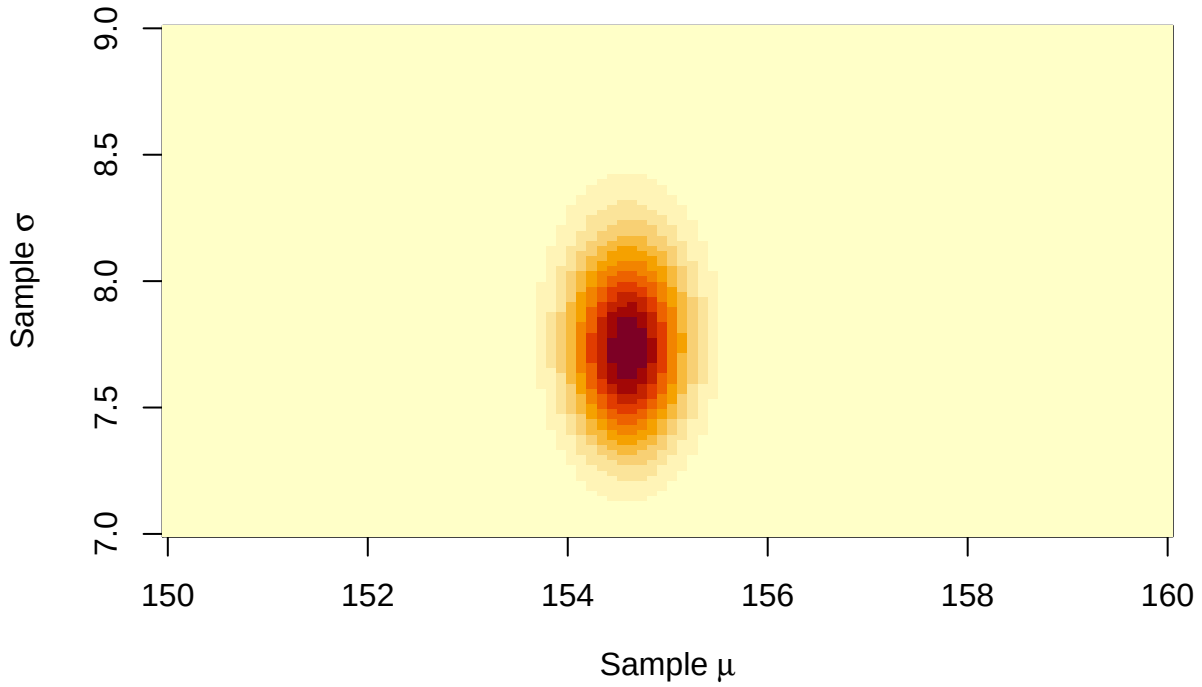
Grid approximation of posterior

Worth running posterior since first multi-parameter model and first Gaussian. Code given without explanation :)

```
mu.list <- seq( from=150, to=160 , length.out=100 )
sigma.list <- seq( from=7 , to=9 , length.out=100 )
post <- expand.grid( mu=mu.list , sigma=sigma.list )
post$LL <- sapply( 1:nrow(post) , function(i) sum(
  dnorm( d2$height , post$mu[i] , post$sigma[i] , log=TRUE ) ) )
post$prod <- post$LL + dnorm( post$mu , 178 , 20 , TRUE ) +
    dunif( post$sigma , 0 , 50 , TRUE )
post$prob <- exp( post$prod - max(post$prod) )
contour_xyz( post$mu , post$sigma , post$prob,
            xlab=expression(paste("Sample ",mu)),
            ylab=expression(paste("Sample ",sigma)))
```

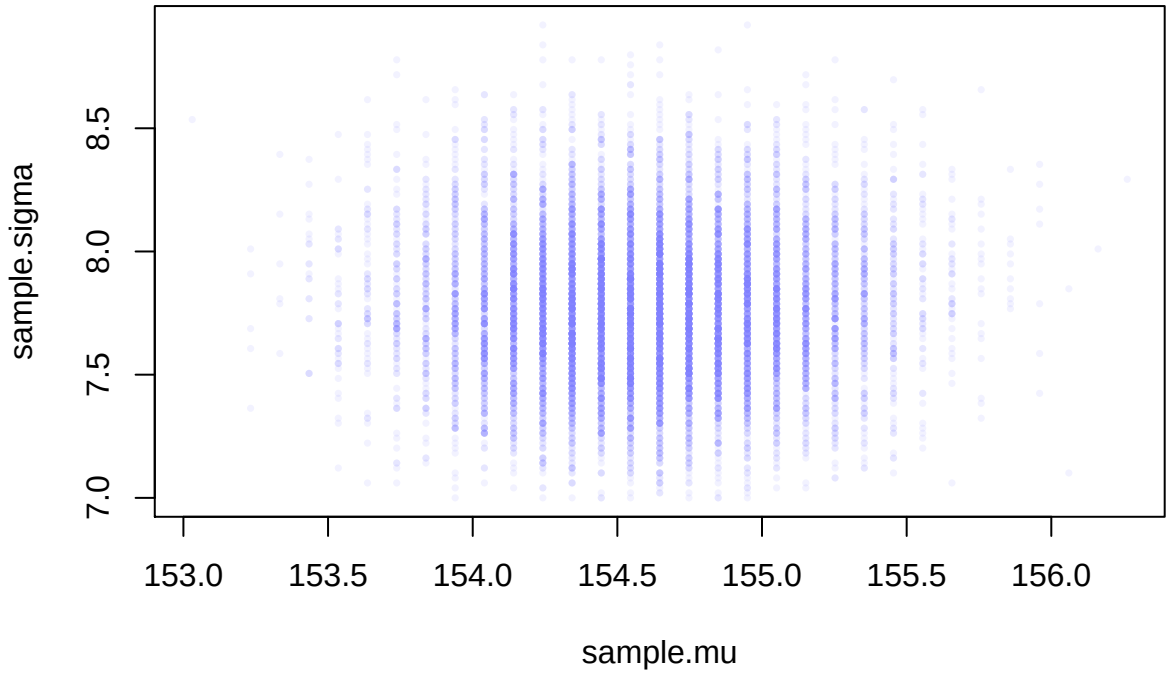



```
image_xyz( post$mu , post$sigma , post$prob ,
            xlab=expression(paste("Sample ",mu)),
            ylab=expression(paste("Sample ",sigma)))
```



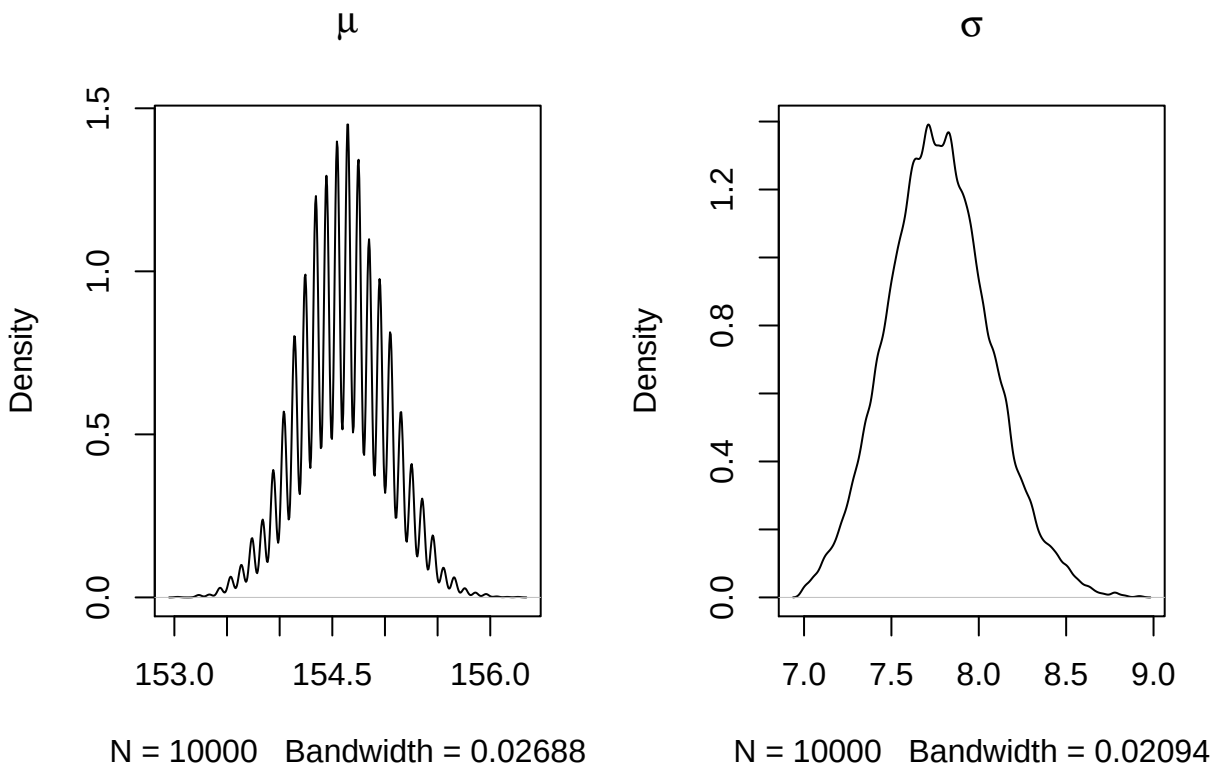
Sampling posterior

```
sample.rows <- sample( 1:nrow(post) , size=1e4 , replace=TRUE , prob=post$prob )
sample.mu <- post$mu[ sample.rows ]
sample.sigma <- post$sigma[ sample.rows ]
plot( sample.mu , sample.sigma , cex=0.5 , pch=16 , col=col.alpha(rangi2,0.1) )
```



(note - the text has a different plot for this, theirs has no horizontal spaces, oddly)

```
par(mfrow=c(1,2))
dens( sample.mu , main=expression(mu))
dens( sample.sigma, main=expression(sigma) )
```



Here we can see something has gone odd with the posterior μ. Regardless, this is sampling of the *marginal* posterior density, meaning averaging over other parameters.

Finding the posterior with quap

Moving onto quadratic approximation (repeating some code for clarity).

```
# Load Data
data(Howell1)
d <- Howell1
d2 <- d[ d$age >= 18 , ] # Select Adults

flist <- alist(
  height ~ dnorm( mu , sigma ), # height ~ Normal(mu,sigma)
  mu ~ dnorm( 178 , 20 ),  # mu ~ Normal (178, 20)
  sigma ~ dunif( 0 , 50 ) #sigma ~ Uniform(0,50)
)

# Fit the model to the data in the frame
start <- list( # Give nice starting location
  mu=mean(d2$height),
  sigma=sd(d2$height)
)
m4.1 <- quap( flist , data=d2 , start=start)
# Look at posterior
precis( m4.1 , prob=0.95)
```

```
##             mean        sd       2.5%      97.5%
```

```
## mu    154.607024 0.4119947 153.799529 155.414518
## sigma   7.731333 0.2913860   7.160227   8.302439
```

This shows the marginal distribution of each parameter -

Sampling from a quap

When R constructs quadratic, it calculates covariances, which are sufficient to make Gaussians. Variance-covariance matrix given by:

```
vcov( m4.1 )
```

```
##                 mu        sigma
## mu    0.1697396109 0.0002180307
## sigma 0.0002180307 0.0849058224
```

Decomposition:

```
diag( vcov( m4.1 ) )
```

```
##         mu      sigma
## 0.16973961 0.08490582
```

```
cov2cor( vcov( m4.1 ) )
```

```
##                mu       sigma
## mu    1.000000000 0.001816174
## sigma 0.001816174 1.000000000
```

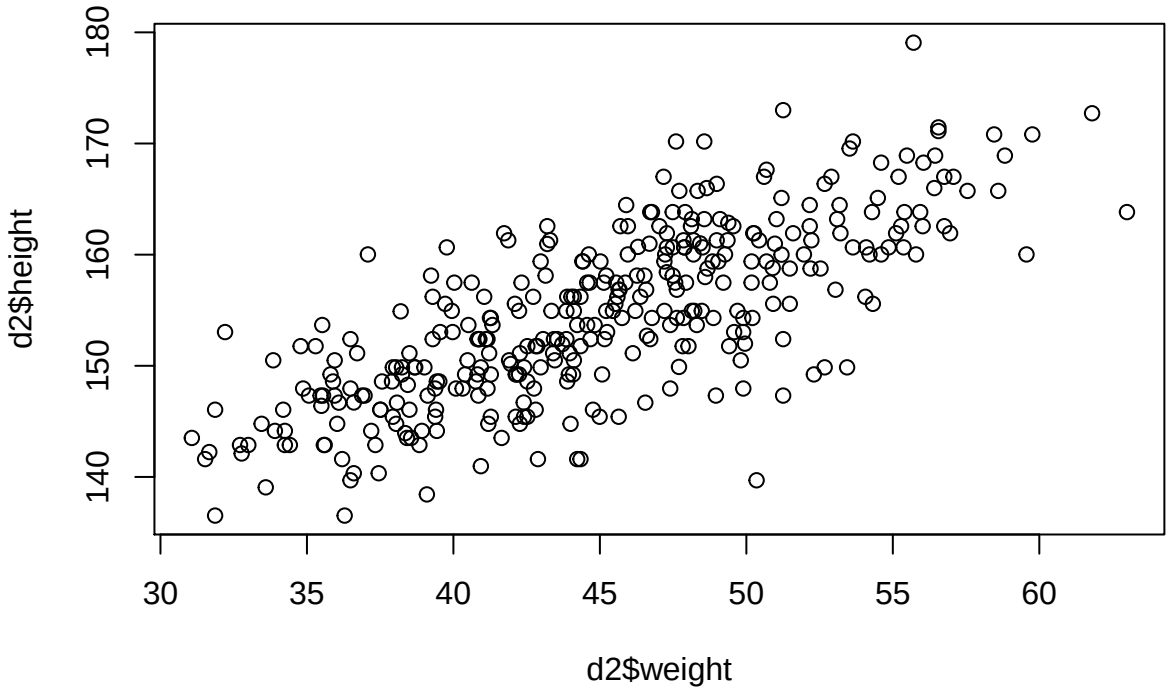
Sample vectors from of values from multi-dimensional Gaussian distribution (provided by rethinking package)

```
post <- extract.samples( m4.1 , n=1e4 )
#precis(post)
```

4.4 - Linear Prediction

Typically interested in how outcome is related to a *predictor variable*. Plot height and weight to see how well they covary.

```
plot( d2$height ~ d2$weight )
```



Linear Model Strategy

Strategy - make parameter for the mean of a Gaussian μ into a linear function of the predictor variable and other new parameters we invent. Instructs the analysis to assume that the predictor has a **constant** and **additive** relationship to the mean of the outcome.

Basically - "consider all lines that relate one variable to the other, rank in order of plausibility given the data"

Before our Gaussian model was:

$$\begin{aligned} h_i &\sim \text{Normal}(\mu, \sigma) \\ \mu &\sim \text{Normal}(178, 20) \\ \sigma &\sim \text{Uniform}(0, 50) \end{aligned}$$

To get weight into the Gaussian model we instead relate the two, letting x be the column of weight measurements, with average $\bar{x}$

$$\begin{aligned} h_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \beta(x_i - \bar{x}) \\ \alpha &\sim \text{Normal}(178, 20) \\ \beta &\sim \text{Normal}(0, 10) \\ \alpha &\sim \text{Uniform}(0, 50) \end{aligned}$$

Step by step:

$h_i \sim \text{Normal}(\mu_i, \sigma)$, the probability of the observed height, this is basically unchanged, except μ has been replaced by μ_i, implying mean depends on row.

$\mu_i = \alpha + \beta(x_i - \bar{x})$, the linear model. No longer estimate μ, construct from other parameters. Notice, not stochastic (=, not $\sim$), meaning this is deterministic; once we know α and β and x_i, we know mu_i. *alpha* and β are made up, just devices for manipulating μ; think of them as targets of learning.

The rest are priors. All have been seen before except β. If we look at many values of β:

We can immediately see that this is barely constrained, even negative values. Try swapping it for something more useful, $\beta \sim \text{Log-Normal}(0, 1)$

Much better constraints.

Finding the posterior

Let's define full model:

```
data(Howell1)
d <- Howell1
d2 <- d[ d$age >= 18 , ]
# define the average weight, x-bar
xbar <- mean(d2$weight)
# fit model
m4.3 <- quap(
alist(
  height ~ dnorm( mu , sigma ) ,
  mu <- a + b*( weight - xbar ) ,
  a ~ dnorm( 178 , 20 ) ,
  b ~ dlnorm( 0 , 1 ) ,
  sigma ~ dunif( 0 , 50 ) ),
data=d2 )
```

Interpreting the posterior

Table approach:

```
precis(m4.3, prob=.95)
```

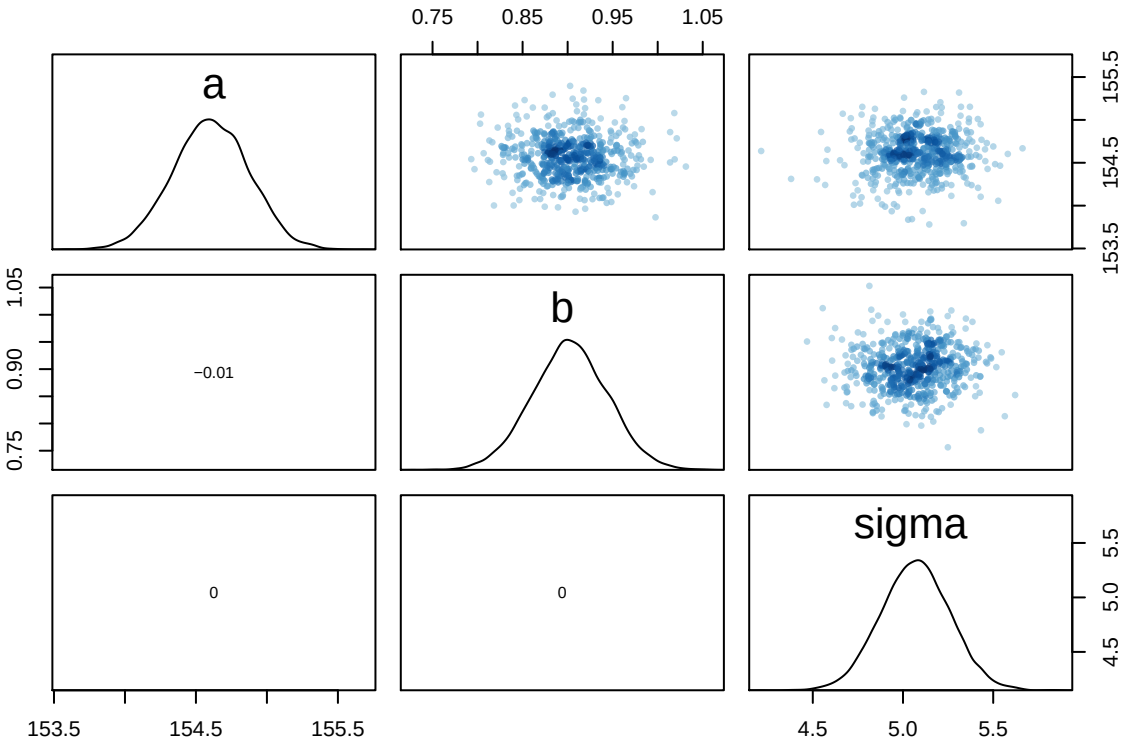
```
##              mean         sd        2.5%      97.5%
## a     154.6019647 0.27030139 154.0721837 155.131746
## b       0.9032842 0.04192265   0.8211173   0.985451
## sigma   5.0717621 0.19114360   4.6971276   5.446397
```

The new parameter, β, is a slope - interpret as "a person 1 kg heavier is expected to be .9 cm taller." 95% of the posterior is between 0.82 and 0.99, so values around 0 or higher than 1 are highly incompatible with these data and model. This is *not* evidence the relationship is linear, but saying "if you commit to lines, ones around 0.9 slope are reasonable."

```
 round( vcov( m4.3 ) , 3 )
```

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

```
pairs(m4.3)
```

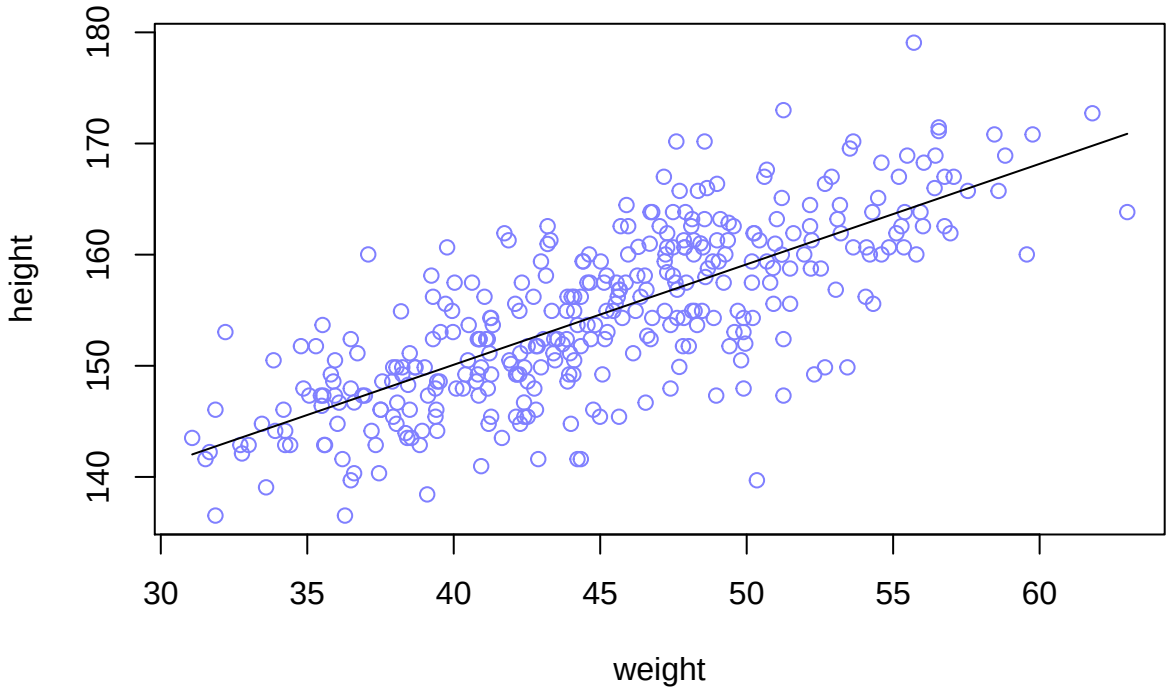


Low covariation.

Plotting posterior inference against the data

Always useful to plot posterior over data to check fit worked, and to interpret posterior.

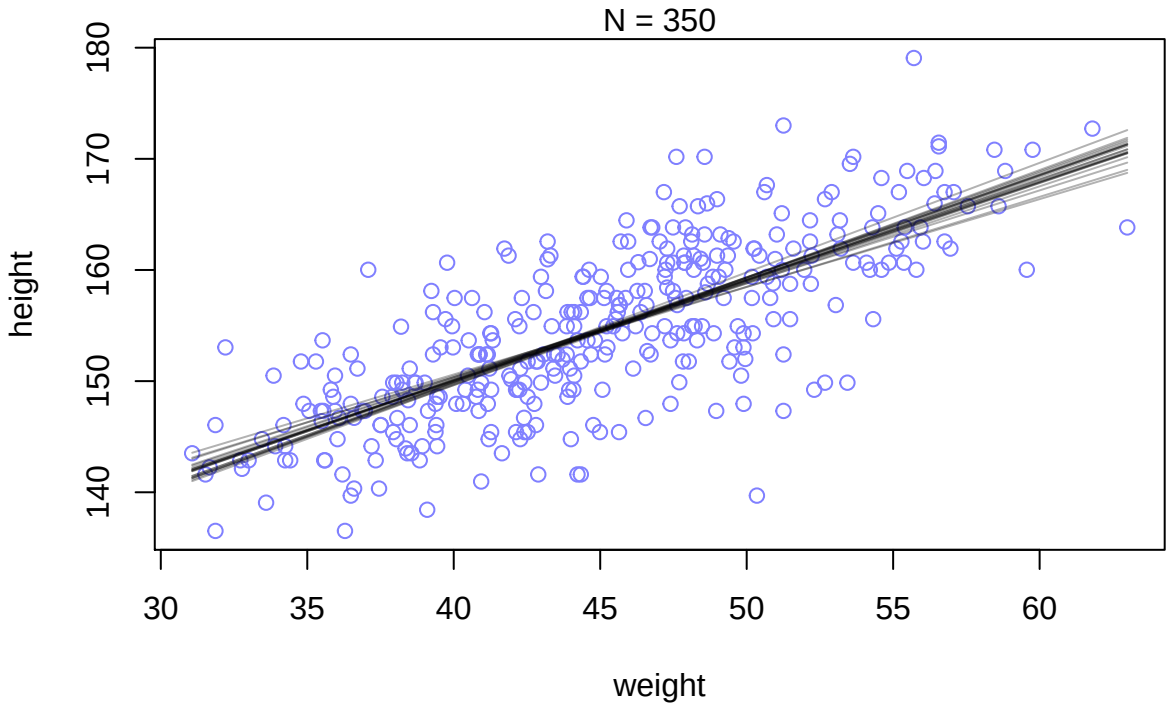
```
plot( height ~ weight , data=d2 , col=rangi2 )
post <- extract.samples( m4.3 )
a_map <- mean(post$a)
b_map <- mean(post$b)
curve( a_map + b_map*(x - xbar) , add=TRUE )
```



Adding uncertainty around mean

While that's the most plausible line, many others are possible, looking at these are good for communicating uncertainty.

```
N <- 350
dN <- d2[ 1:N , ]
mN <- quap(
  alist(
  height ~ dnorm( mu , sigma ) ,
  mu <- a + b*( weight - mean(weight) ) ,
  a ~ dnorm( 178 , 20 ) ,
  b ~ dlnorm( 0 , 1 ) ,
  sigma ~ dunif( 0 , 50 )
  ) , data=dN )

# extract 20 samples from the posterior
post <- extract.samples( mN , n=20 )
# display raw data and sample size
plot( dN$weight , dN$height ,
    xlim=range(d2$weight) , ylim=range(d2$height) ,
    col=rangi2 , xlab="weight" , ylab="height" )
mtext(concat("N = ",N))
# plot the lines, with transparency
for ( i in 1:50 )
  curve( post$a[i] + post$b[i]*(x-mean(dN$weight)) ,
         col=col.alpha("black",0.3) , add=TRUE )
```

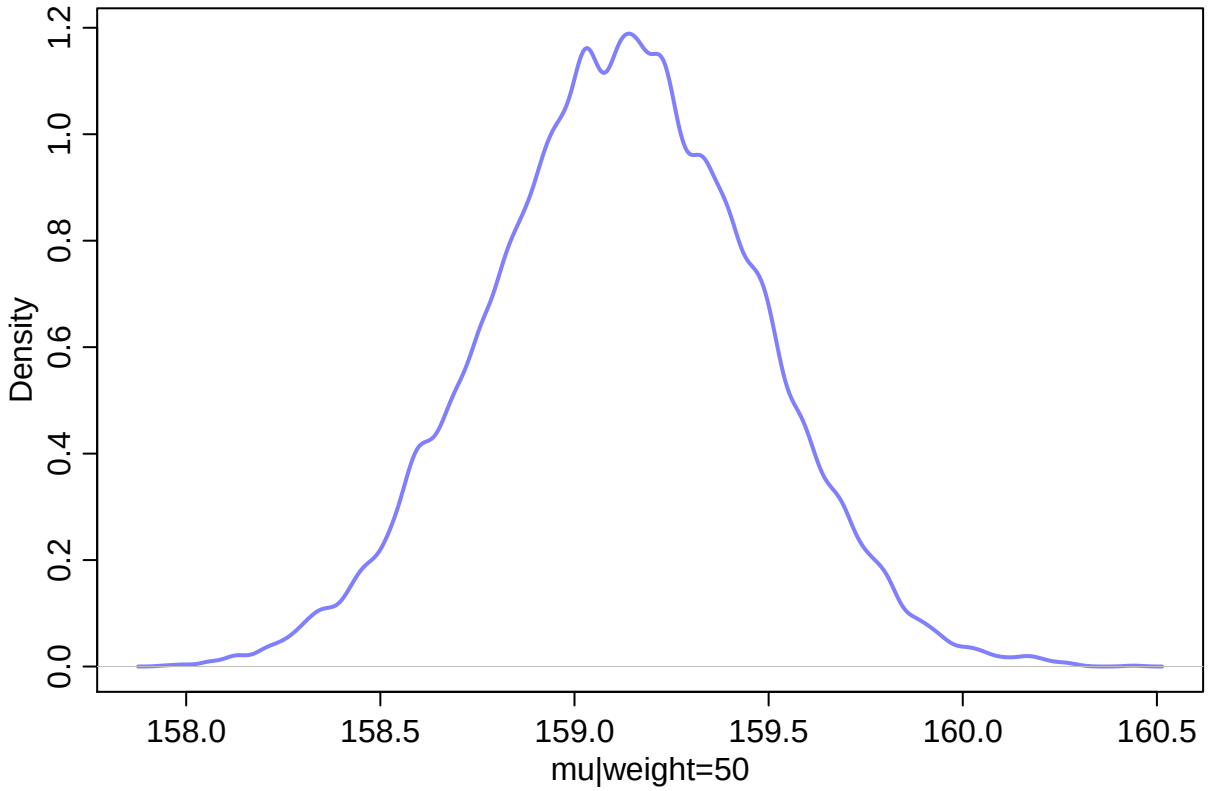


For 350 points, 50 possible lines.

Plotting regression intervals and contours

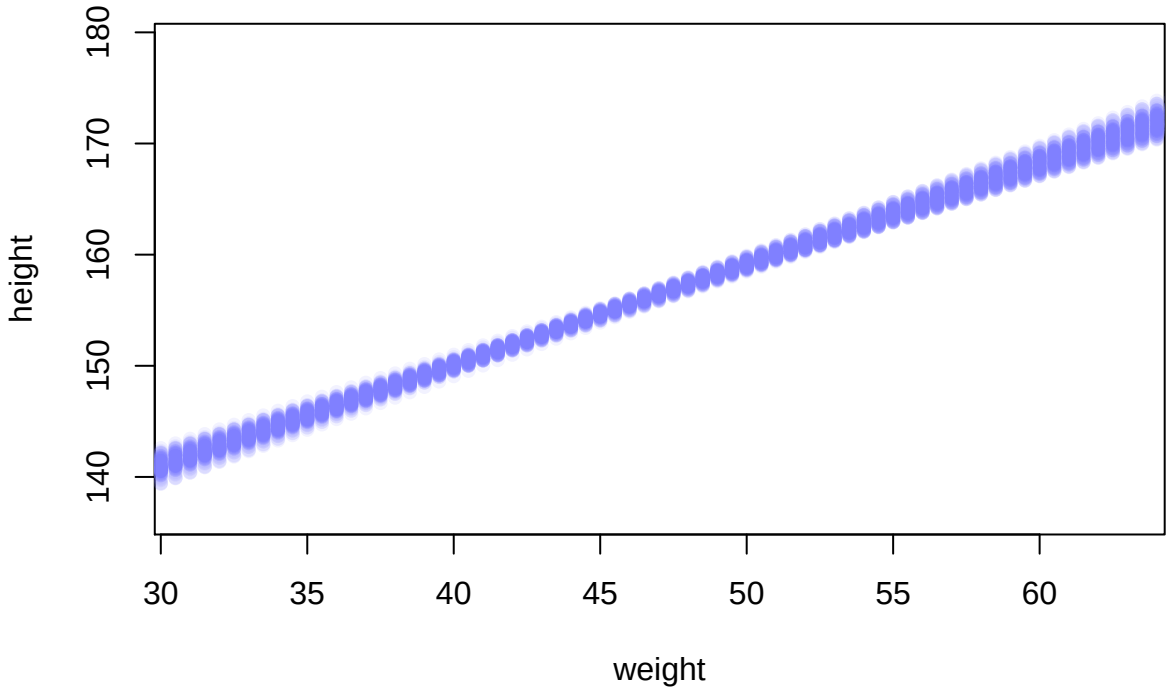
For a single value (e.g. 50 kg), we can make a distribution

```
post <- extract.samples( m4.3 )
mu_at_50 <- post$a + post$b * ( 50 - xbar )
dens( mu_at_50 , col=rangi2 , lwd=2 , xlab="mu|weight=50" )
```



If we want to evaluate over all the values, we can use the link function, which generates posterior samples (must pass it data to evaluate on). Then can use points to visualize.

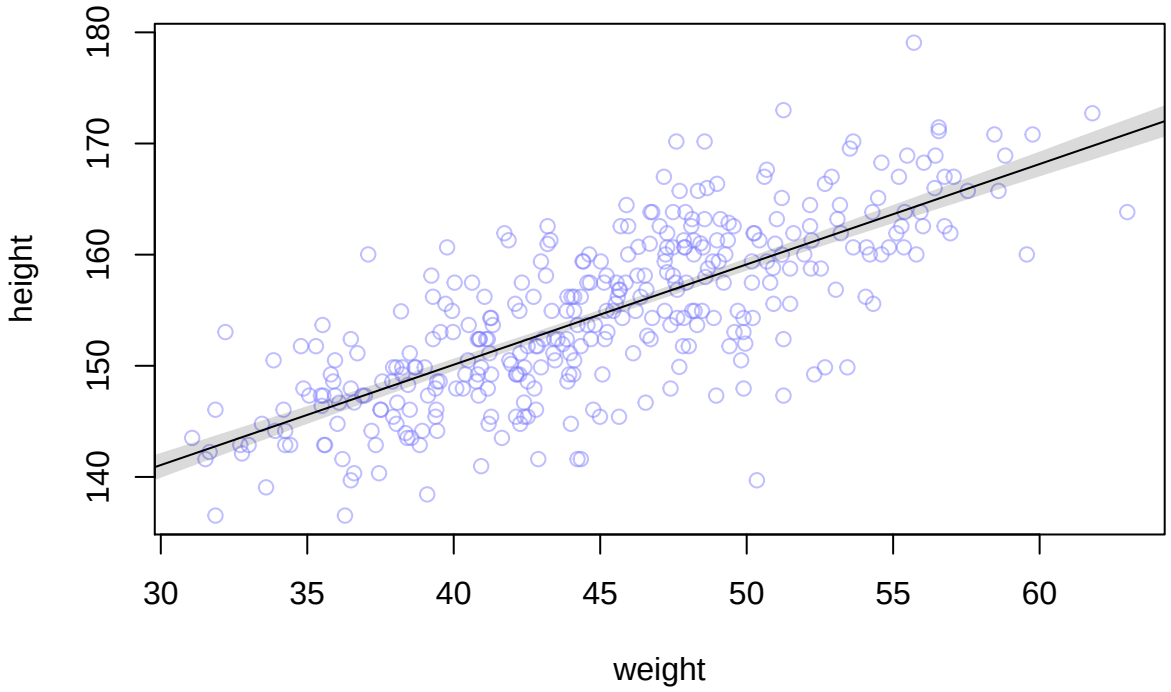
```
# define sequence of weights to compute predictions for # these values will be on the horizontal axis
weight.seq <- seq( from=25 , to=70 , by=.5 )
# use link to compute mu
# for each sample from posterior
# and for each weight in weight.seq
mu <- link( m4.3 , data=data.frame(weight=weight.seq) )

# use type="n" to hide raw data
plot( height ~ weight , d2 , type="n" )
# loop over samples and plot each mu value
for ( i in 1:100 )
points( weight.seq , mu[i,] , pch=16 , col=col.alpha(rangi2,0.1) )
```



Alternatively, shade function works too.

```
# summarize the distribution of mu
mu.mean <- apply( mu , 2 , mean )
mu.PI <- apply( mu , 2 , PI , prob=0.89 )

# plot raw data
# fading out points to make line and interval more visible
plot( height ~ weight , data=d2 , col=col.alpha(rangi2,0.5) )
# plot the MAP line, aka the mean mu for each weight
lines( weight.seq , mu.mean )
# plot a shaded region for 89% PI
shade( mu.PI , weight.seq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```



Prediction Intervals

Important - we've only considered the range of μ, not included σ. We can do that via the sim function, which simulates heights. Then we can plot all the distributions we've seen so far, the data, the MAP, the PI, and the HPDI.

```
sim.height <- sim( m4.3 , data=list(weight=weight.seq) )
height.PI <- apply( sim.height , 2 , PI , prob=0.89 )

# plot raw data
plot( height ~ weight , d2 , col=col.alpha(rangi2,0.6) )
# draw MAP line
lines( weight.seq , mu.mean ,col="red")
# draw HPDI region for line
mu.HPDI <- apply( mu , 2 , HPDI , prob=0.89)
shade( mu.HPDI , weight.seq, col=col.alpha('cornflowerblue',0.4) )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```

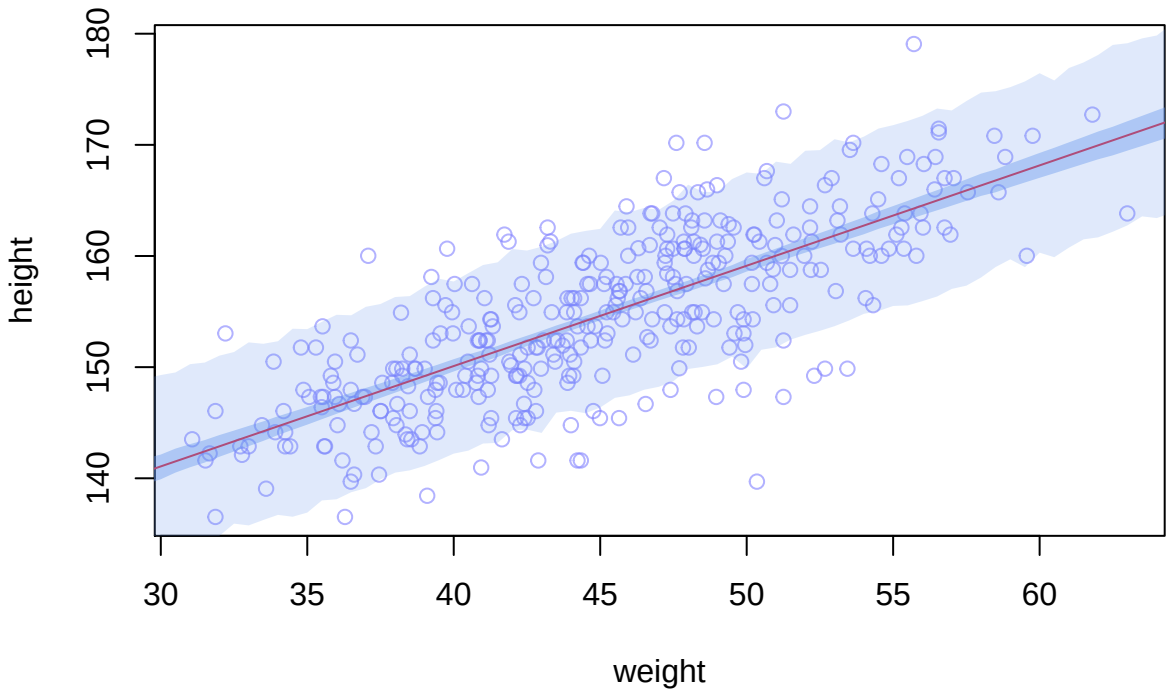
```
# draw PI region for simulated heights
shade( height.PI , weight.seq , col=col.alpha('cornflowerblue',0.2) )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```



Curves From Lines

No a priori reason why linear models are special, just that they're simple. Can consider a polynomial model:

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 \\
\alpha &\sim \text{Normal}(178, 20) \\
\beta_1 &\sim \text{Log-Normal}(0, 1) \\
\beta_2 &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

```
library(rethinking)
data(Howell1)
d <- Howell1

d$weight_s <- ( d$weight - mean(d$weight) )/sd(d$weight)
d$weight_s2 <- d$weight_s^2
m4.5 <- quap(
  alist(
  height ~ dnorm( mu , sigma ) ,
```

```
  mu <- a + b1*weight_s + b2*weight_s2 , a ~ dnorm( 178 , 20 ) ,
  b1 ~ dlnorm( 0 , 1 ) ,
  b2 ~ dnorm( 0 , 1 ) ,
  sigma ~ dunif( 0 , 50 )
),
data=d )
precis( m4.5 )
```

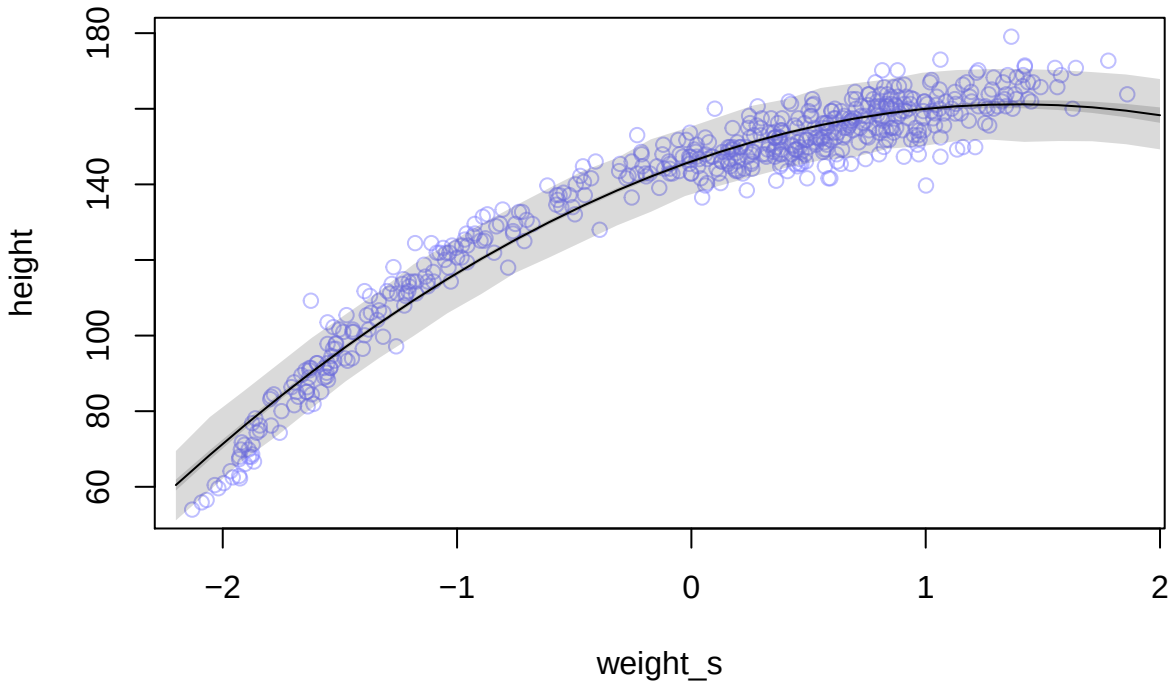
```
##             mean        sd       5.5%      94.5%
## a     146.056120 0.3690237 145.466349 146.645891
## b1     21.733752 0.2889245  21.271995  22.195509
## b2     -7.802221 0.2742210  -8.240480  -7.363963
## sigma   5.775145 0.1765175   5.493036   6.057254
```

```
weight.seq <- seq( from=-2.2 , to=2 , length.out=30 )
pred_dat <- list( weight_s=weight.seq , weight_s2=weight.seq^2 )
mu <- link( m4.5 , data=pred_dat )
mu.mean <- apply( mu , 2 , mean )
mu.PI <- apply( mu , 2 , PI , prob=0.89 )
sim.height <- sim( m4.5 , data=pred_dat )
height.PI <- apply( sim.height , 2 , PI , prob=0.89 )

plot( height ~ weight_s , d , col=col.alpha(rangi2,0.5) )
lines( weight.seq , mu.mean )
shade( mu.PI , weight.seq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```

```
shade( height.PI , weight.seq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```



Can consider even higher orders (cubic, quartic, etc.), but just because they fit the sample better doesn't make them better models, moreover it doesn't have biological information, so no causal relationships can be found. These are addressed later.

Next can also consider spline models, looking at the cherry blossom dataset.

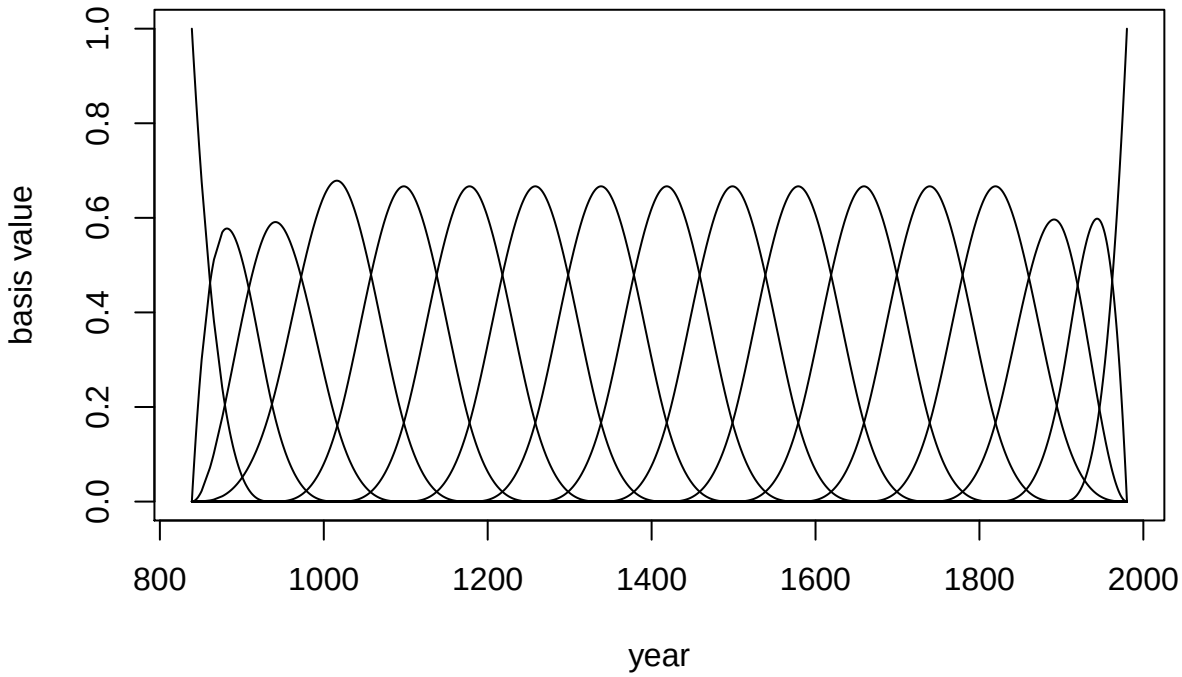
```
library(rethinking)
data(cherry_blossoms)
d <- cherry_blossoms
#precis(d)
```

Splines basically split up a predictor into parts, then has weights for when they turn on/turn off:

$$\mu_i = \alpha + w_1 B_{i,1} + w_2 B_{i,2} + w_3 B_{i,3} + \dots$$

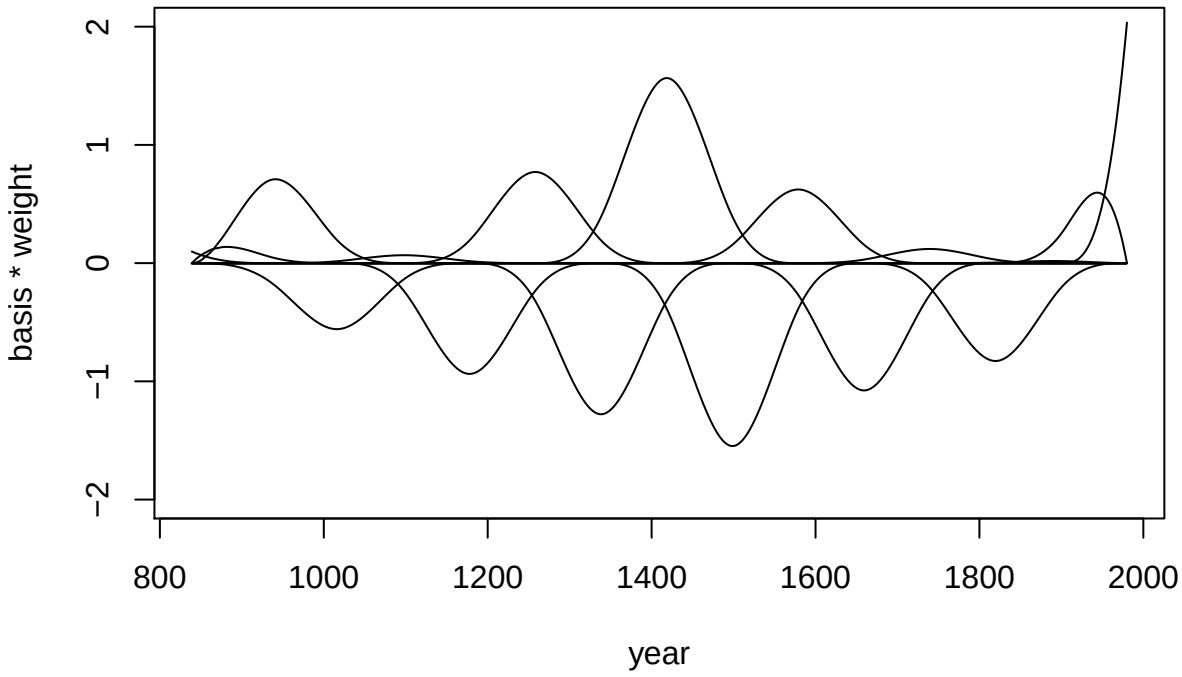
Pivot points are called knots. Here we make a model with 15 knots, which are of degree 3.

```
d2 <- d[ complete.cases(d$temp) , ] # complete cases on temp
num_knots <- 15
knot_list <- quantile( d2$year , probs=seq(0,1,length.out=num_knots) )

library(splines)
B <- bs(d2$year,
        knots=knot_list[-c(1,num_knots)] ,
        degree=3 , intercept=TRUE )

plot( NULL , xlim=range(d2$year) , ylim=c(0,1) , xlab="year" , ylab="basis value" )
for ( i in 1:ncol(B) ) lines( d2$year , B[,i])
```



Next we fit to data and plot the knots times their weight values

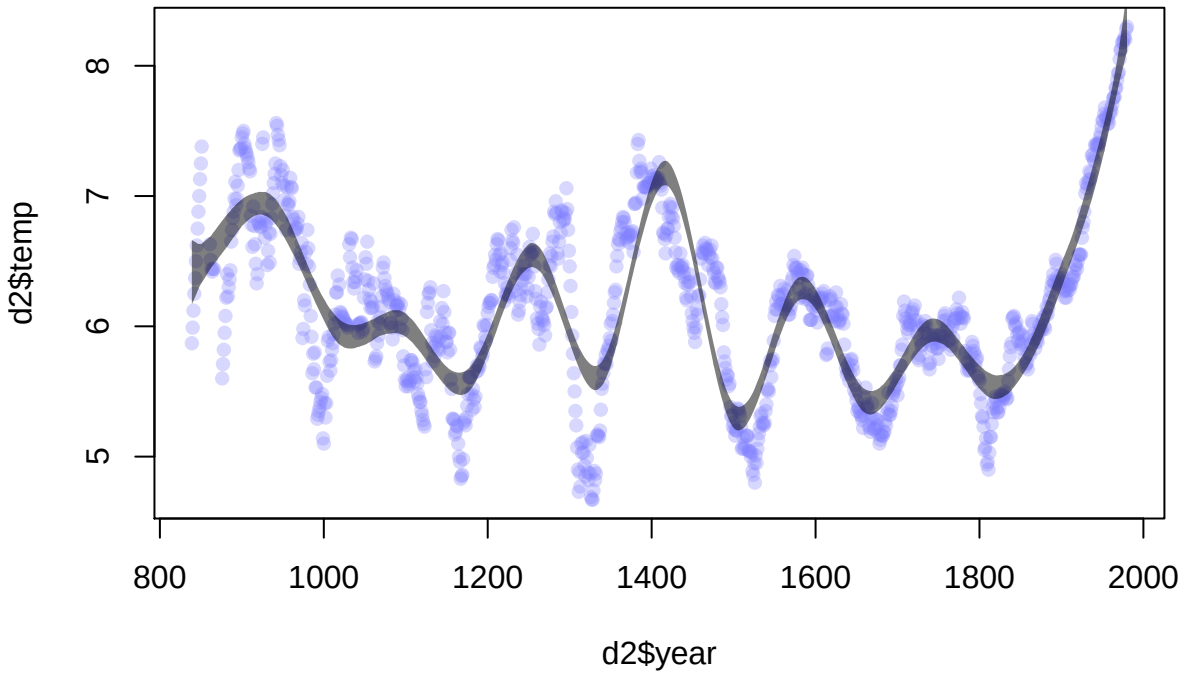
```
m4.7 <- quap(
  alist(
    T ~ dnorm( mu , sigma ) ,
    mu <- a + B %*% w ,
    a ~ dnorm(6,10),
    w ~ dnorm(0,1),
    sigma ~ dexp(1)
    ),
    data=list( T=d2$temp , B=B ) ,
    start=list( w=rep( 0 , ncol(B) ) ) )

post <- extract.samples(m4.7)
w <- apply( post$w , 2 , mean )
plot( NULL , xlim=range(d2$year) , ylim=c(-2,2) ,
    xlab="year" , ylab="basis * weight" )
for ( i in 1:ncol(B) ) lines( d2$year , w[i]*B[,i] )
```



Last we can fit the full splined model to the data

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

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```



This model is:

$$
\begin{aligned}
T_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \sum_{k=1}^{K} w_k B_{k,i} \\
\alpha &\sim \text{Normal}(6, 10) \\
w_j &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

Summary

Looked a linear regression, estimating association between a predictor and outcome. The likelihood is comprised of a Gaussian.

Chapter 5 - The Many Variables & The Spurious Waffles

Title explanation: Waffle Houses are an index to disaster severity, opening quickly after severe events. Also correlated to divorce rates, this is a spurious correlation - it's a southern establishment, and the south has high divorce rates. Need to distinguish correlations from causation.

Multiple Regression - using multiple predictors to model an outcome. Useful for:

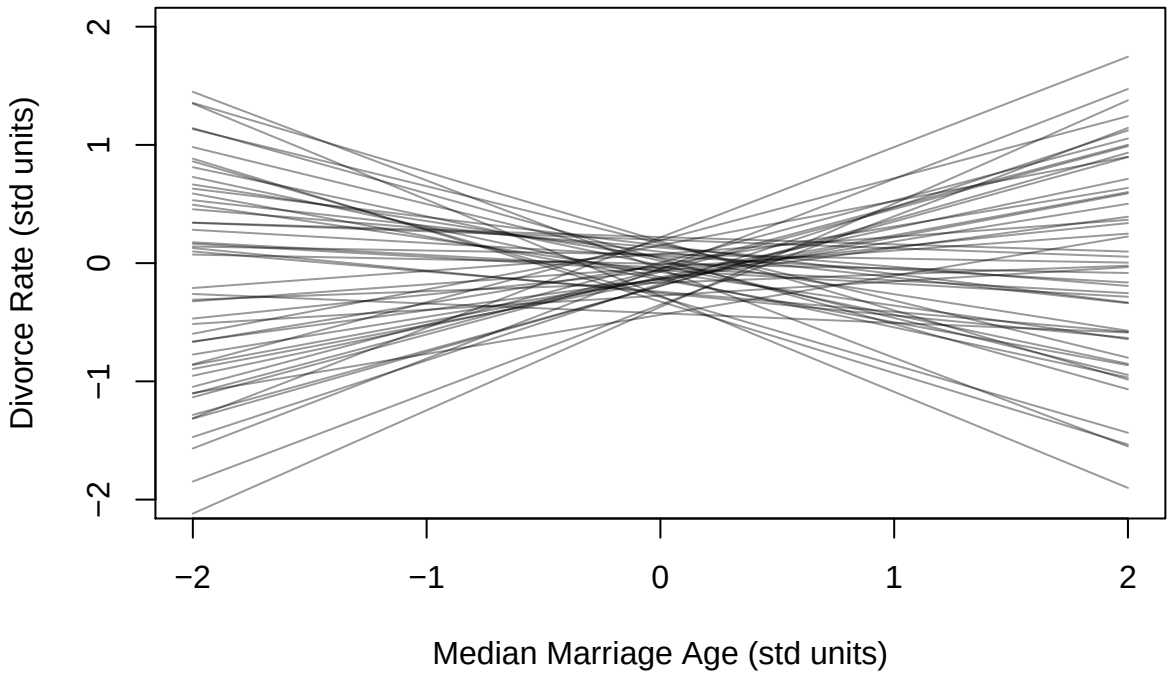
- Acting as a statistical control for confounds
- Multiple causation
- Interactions between variables (not dealt with in this chapter)

Also looking at **causal inference** this chapter, using graphs.

5.1 - Spurious Association

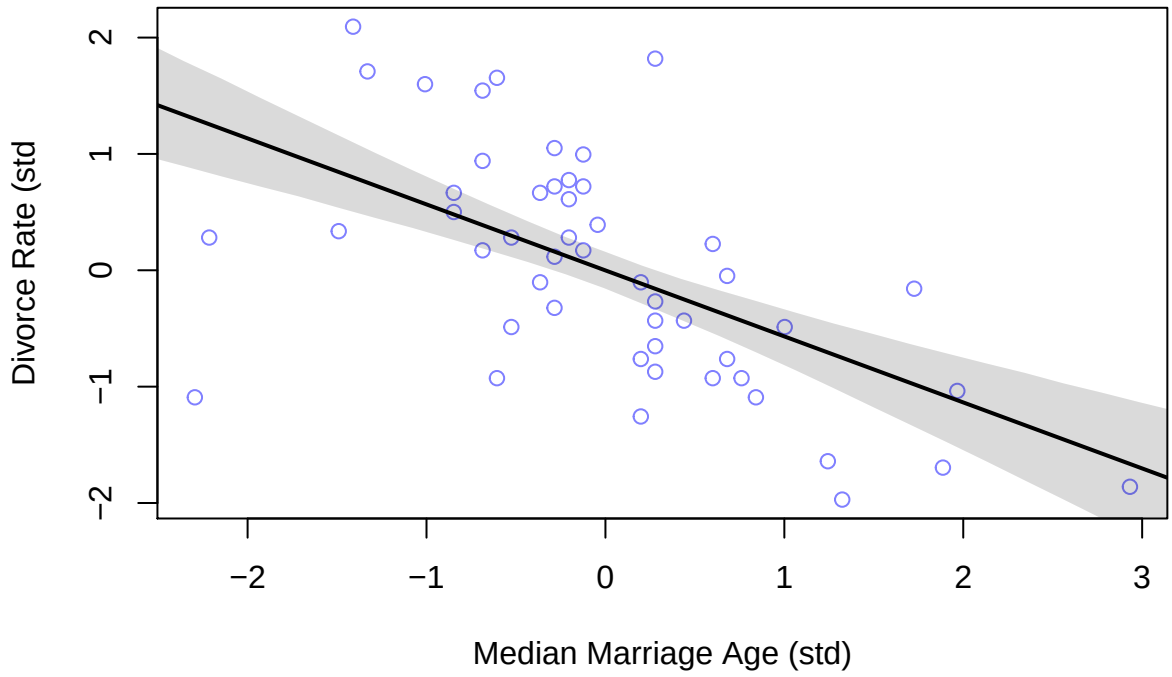
Test problem now divorce as a function of marriage rate, median marriage age. Straightforward linear regression.

```
# load data and copy
data(WaffleDivorce)
d <- WaffleDivorce
# standardize variables
d$A <- scale( d$MedianAgeMarriage )
d$D <- scale( d$Divorce )

# Build model and sample
m5.1 <- quap(
  alist(
    D ~ dnorm( mu , sigma ) ,
    mu <- a + bA * A ,
    a ~ dnorm( 0 , 0.2 ) ,
    bA ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) , data = d )

set.seed(10)
prior <- extract.prior( m5.1 )
mu <- link( m5.1 , post=prior , data=list( A=c(-2,2) ) )
plot( NULL , xlim=c(-2,2) , ylim=c(-2,2),
      xlab="Median Marriage Age (std units)", ylab="Divorce Rate (std units)" )
for ( i in 1:50 ) lines( c(-2,2) , mu[i,] , col=col.alpha("black",0.4) )
```



Looking at posterior

```
# compute percentile interval of mean
A_seq <- seq( from=-3 , to=3.2 , length.out=30 )
mu <- link( m5.1 , data=list(A=A_seq) )
mu.mean <- apply( mu , 2, mean)
mu.PI <- apply( mu , 2, PI)

# plot it all
plot( D ~ A , data=d , col=rangi2 , xlab="Median Marriage Age (std)", ylab="Divorce Rate (std")
lines( A_seq , mu.mean , lwd=2 )
shade( mu.PI , A_seq )
```



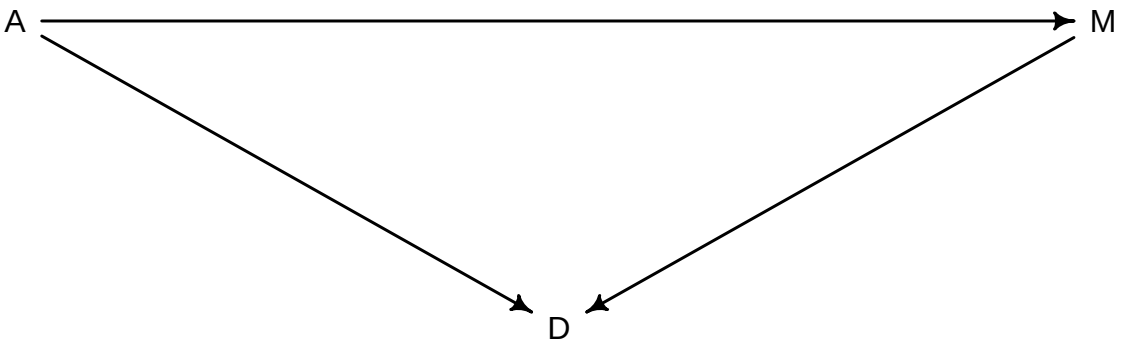
To make sense, we need to work on causation.

Think before you regress

Three variables: Divorce rate (D), marriage rate (M), median age of marriage (A).

Set up a Directed Acyclic Graph (DAG)

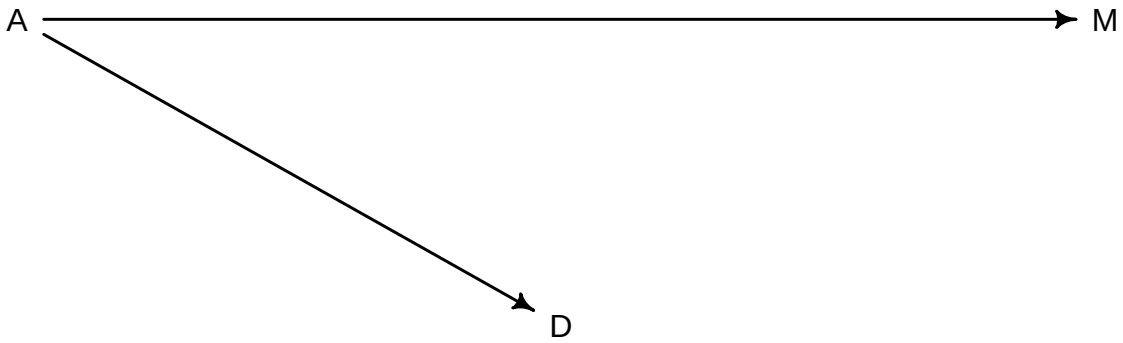
```
library(dagitty)
dag5.1 <- dagitty( "dag {
  A -> D
  A -> M
  M -> D
}")
coordinates(dag5.1) <- list( x=c(A=0,D=1,M=2) , y=c(A=0,D=1,M=0) )
drawdag( dag5.1 )
```



Depicts the directions of influence - Age affects both marriages and divorce, marriages affect divorces. Have to account for each path, $A \rightarrow D$ and $A \rightarrow M \rightarrow D$.

$A \rightarrow M \rightarrow D$ path does little work, we know marriage is positively associated with divorce. The graph could also be something like this:

```
dag5.1 <- dagitty( "dag { D <- A -> M}")
coordinates(dag5.1) <- list( x=c(A=0,D=1,M=2) , y=c(A=0,D=1,M=0) )
drawdag( dag5.1 )
```



Also plausible - Carefully consider each DAG to know which is correct.

Testable Implications -

Compare the two DAGs, the second gives the implication that once we've conditioned on A, M tells us nothing more about D; D is independent of M conditional of A ($D \perp\!\!\!\perp M|A$)

Code version of that:

```
DMA_dag2 <- dagitty('dag{ D <- A -> M }')
impliedConditionalIndependencies( DMA_dag2 )
```

```
## D _||_ M | A
```

Compared to:

```
DMA_dag1 <- dagitty('dag{ D <- A -> M -> D }')
impliedConditionalIndependencies( DMA_dag1 )
```

(no conditional indpendencies, so no output)

To test: need a model that conditions on A, so we can see weather that renders D independent of M, multiple regression can do this.

Answers the question "Is there any additional value in knowing a variable, once I already know all of the other predictor variables?"

Note - often this question is framed as "controlling for one variable while estimating another," but statistical control is different from experimental so a bit sloppy.

Notation

Strategy:

1. Nominate predictor variables you want in linear model
2. For each, make a parameter that will measure it's association
3. Multiply

e.g.

$$
\begin{aligned}
D_i &\sim \text{Normal}(\mu, \sigma) \\
\mu_i &= \alpha + \beta_M M_i + \beta_A A_i \\
\alpha &\sim \text{Normal}(0, 0.2) \\
\beta_M &\sim \text{Normal}(0, 0.5) \\
\beta_A &\sim \text{Normal}(0, 0.5) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

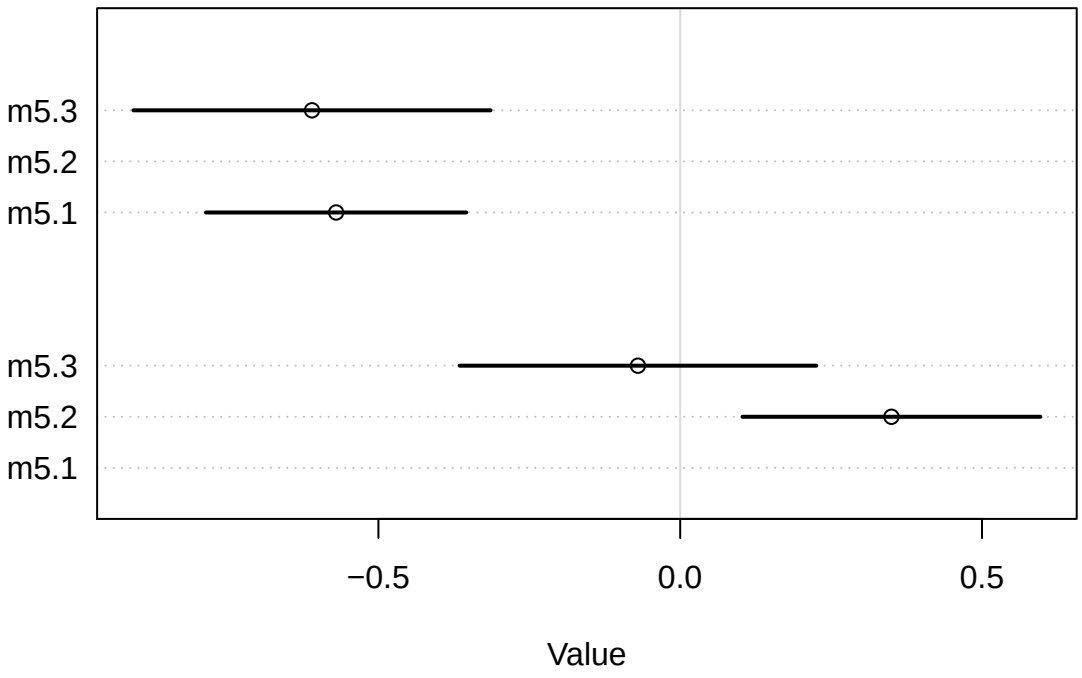
M for marriage rate, A for age of marriage. One sample way to read this is "a divorce rate is a function of its marriage rate or median age at marriage" (read + as "or").

```
d$M <- scale( d$Marriage )
m5.3 <- quap(
  alist(
    D ~ dnorm( mu , sigma ) ,
    mu <- a + bM*M + bA*A ,
    a ~ dnorm( 0 , 0.2 ) ,
    bM ~ dnorm( 0 , 0.5 ) ,
    bA ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
    ) , data = d )
precis( m5.3 )
```

```
##                mean         sd       5.5%      94.5%
## a     -2.828642e-05 0.09707123 -0.1551669  0.1551103
## bM    -6.553086e-02 0.15076312 -0.3064794  0.1754177
## bA    -6.136370e-01 0.15097351 -0.8549218 -0.3723521
## sigma  7.850672e-01 0.07783076  0.6606786  0.9094558
```

Visualization of posterior

```
plot( coeftab(m5.1,m5.2,m5.3), par=c("bA","bM"))
```



89% compatibility shown, top is bA, bottom is bM.

- bA doesn't move much, uncertainty grows
- bM only associated with divorce when age is missing from the model

So there is little to no additional predictive power in knowing the rate of marriage in a state/no direct causal path from marriage rate to divorce rate. Meaning, model 2 is the better one.

Plotting multivariate posteriors

Predictor residual plots - useful for understanding statistical model

Model has 2 predictors: M, marriage rate and A, median age. For residuals, you use the other predictor to model it. Residuals are found by subtracting observed from model.

$$
\begin{aligned}
M_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_A A_i \\
\alpha &\sim \text{Normal}(0, 0.2) \\
\beta &\sim \text{Normal}(0, 0.5) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

```
m5.4 <- quap(
  alist(
    M ~ dnorm( mu , sigma ) ,
    mu <- a + bAM * A ,
    a ~ dnorm( 0 , 0.2 ) ,
    bAM ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) , data = d )

mu <- link(m5.4)
mu_mean <- apply( mu , 2 , mean )
mu_resid <- d$M - mu_mean
```

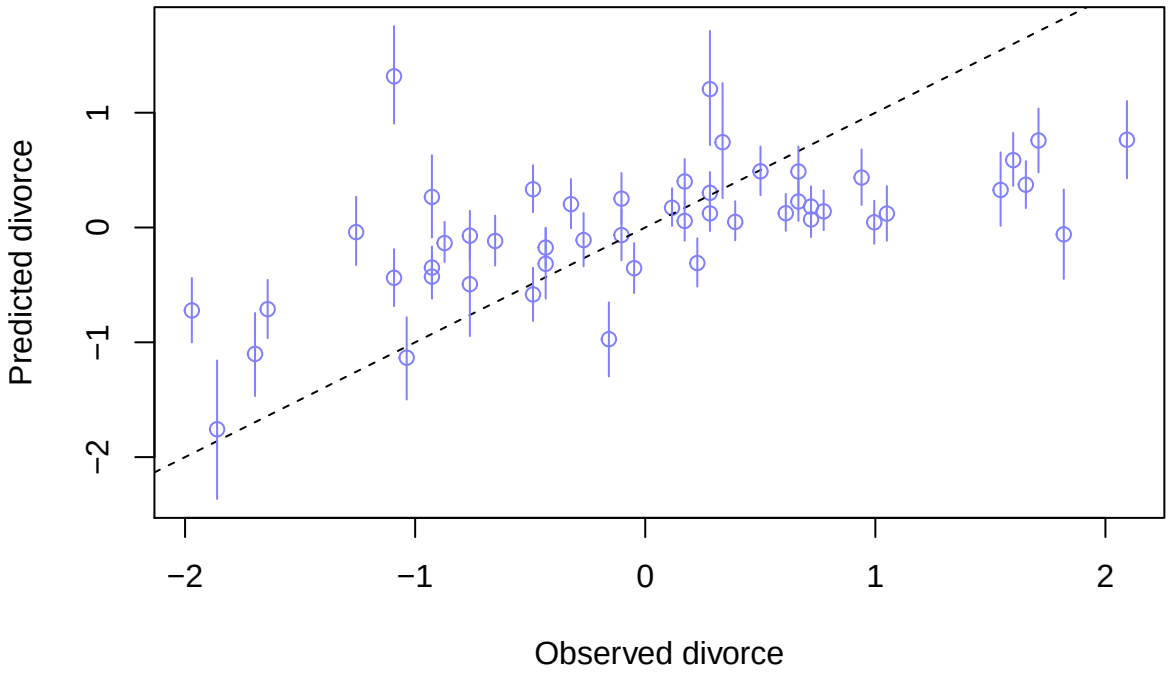
Brings home message that regression models measure the remaining association of each predictor with the outcome, after knowing the other predictors - computing the residuals shows this yourself, but with the unified model it happens automatically.

2. Posterior prediction plots - check fit and assess predictions, not causal tools

Just plot posterior against data, more easily diagnoses errors or why it could fail.

```
# call link without specifying new data # so it uses original data
mu <- link( m5.3 )
# summarize samples across cases
mu_mean <- apply( mu , 2 , mean )
mu_PI <- apply( mu , 2 , PI )
# simulate observations
# again no new data, so uses original data
D_sim <- sim( m5.3 , n=1e4 )
D_PI <- apply( D_sim , 2 , PI )

plot( mu_mean ~ d$D , col=rangi2 , ylim=range(mu_PI) ,
      xlab="Observed divorce" , ylab="Predicted divorce" )
```

```
abline( a=0 , b=1 , lty=2 )
for ( i in 1:nrow(d) ) lines( rep(d$D[i],2) , mu_PI[,i] , col=rangi2 )
```



From this, understand that model under-predicts states with high divorce rates, and over-predicts those with low rates (89% CI shown). Several problematic ones, e.g. Idaho and Utah, high above the mean (high Latter Day Saints population with low divorce rate, can consider demographic compositions).

3. Counterfactual plots - explore causal implications of manipulating variables

Display implied predictions of the model - pick an intervention variable, define a range for it, simulate other values including outcome. Consider the DAG where M affects D.

```
data(WaffleDivorce)
d <- list()
d$A <- standardize( WaffleDivorce$MedianAgeMarriage )
d$D <- standardize( WaffleDivorce$Divorce )
d$M <- standardize(WaffleDivorce$Marriage )
m5.3_A <- quap(
  alist(
    ## A -> D <- M
    D ~ dnorm( mu , sigma ) ,
    mu <- a + bM*M + bA*A ,
    a ~ dnorm( 0 , 0.2 ) ,
    bM ~ dnorm( 0 , 0.5 ) ,
    bA ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 ),
    ## A -> M
    M ~ dnorm( mu_M , sigma_M ),
    mu_M <- aM + bAM*A,
    aM ~ dnorm( 0 , 0.2 ),
    bAM ~ dnorm( 0 , 0.5 ),
    sigma_M ~ dexp( 1 )
) , data = d )
```

```
precis(m5.3_A)
```

```
##                   mean         sd       5.5%      94.5%
## a       -2.192238e-06 0.09707342 -0.1551443  0.1551399
## bM      -6.524040e-02 0.15076987 -0.3061998  0.1757190
## bA      -6.134339e-01 0.15097997 -0.8547290 -0.3721387
## sigma    7.850904e-01 0.07783661  0.6606924  0.9094883
## aM      -3.647809e-05 0.08685053 -0.1388404  0.1387674
## bAM     -6.947411e-01 0.09573042 -0.8477368 -0.5417454
## sigma_M  6.817629e-01 0.06758648  0.5737467  0.7897792
```

This model shows M and A are strongly negatively associated, so manipulate A for 30 observations around 2 sigma of the mean.

```
A_seq <- seq( from=-2 , to=2 , length.out=30 )
# prep data
sim_dat <- data.frame( A=A_seq )
# simulate M and then D, using A_seq
s <- sim( m5.3_A , data=sim_dat , vars=c("M","D") )

# display counterfactual predictions
plot( sim_dat$A , colMeans(s$D) , ylim=c(-2,2) , type="l" ,
      xlab="manipulated A" , ylab="counterfactual D" )
shade( apply(s$D,2,PI) , sim_dat$A )
```

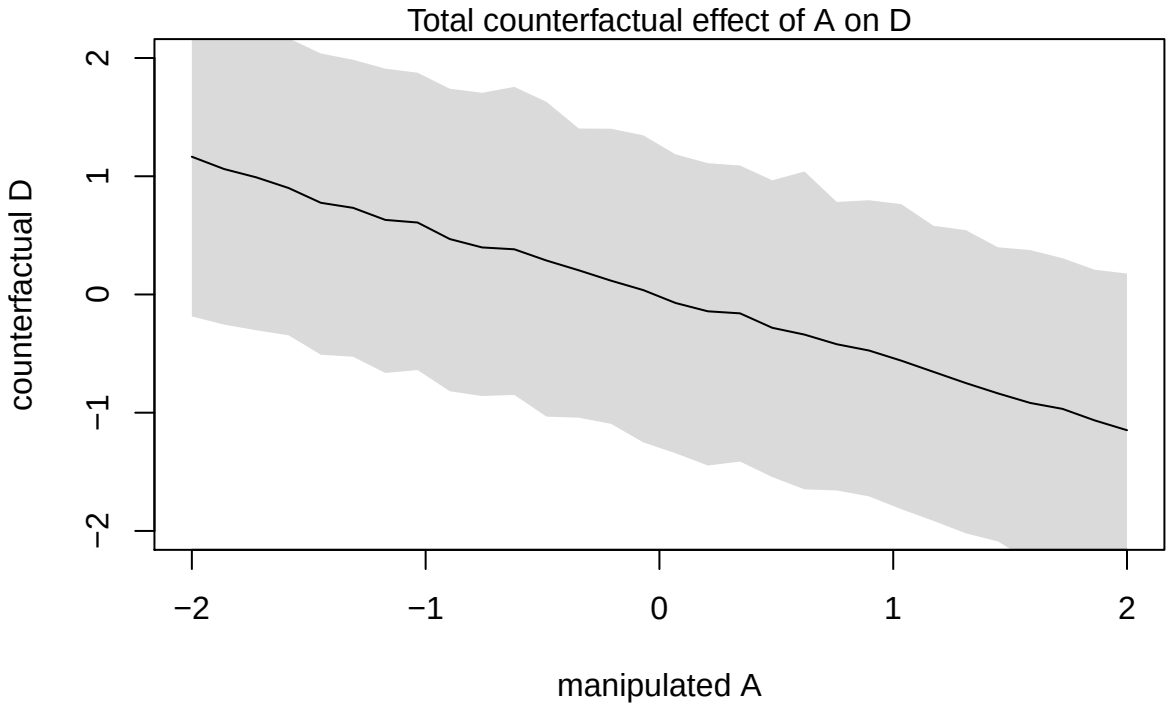
```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```

```
mtext( "Total counterfactual effect of A on D" )
```




```
plot( sim_dat$A , colMeans(s$M) , ylim=c(-2,2) , type="l" ,
      xlab="manipulated A" , ylab="counterfactual M" )
shade( apply(s$M,2,PI) , sim_dat$A )
```

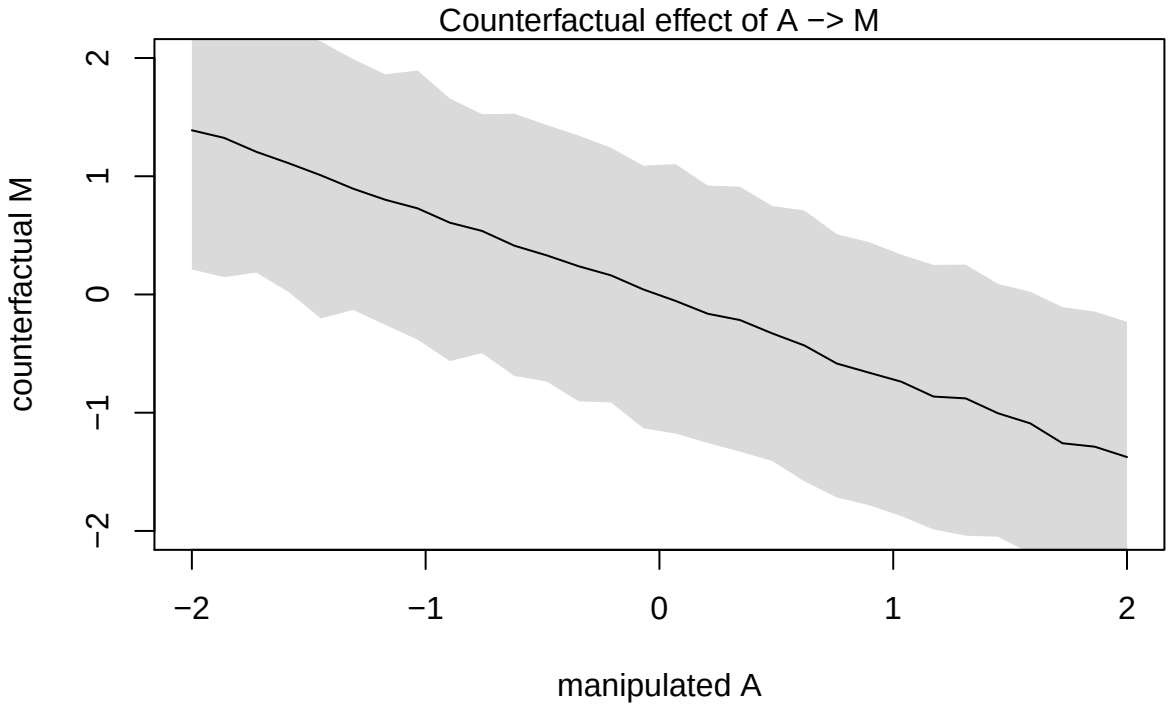
```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used
```

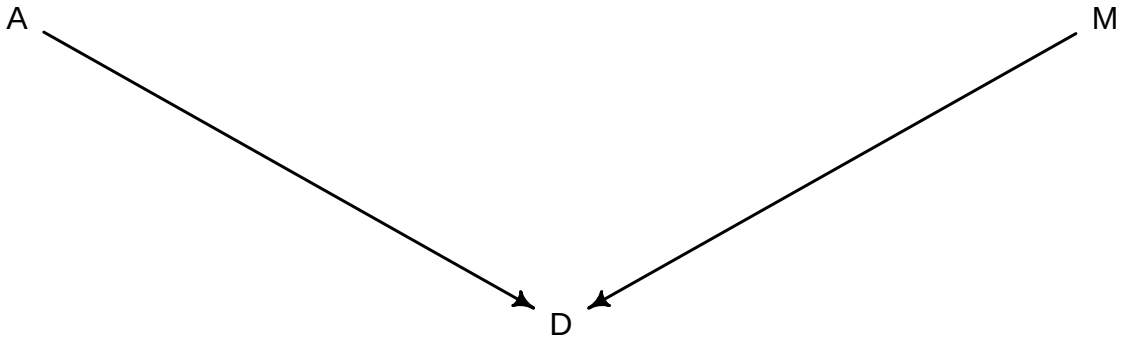
```
## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```

```
mtext( "Counterfactual effect of A -> M" )
```



The trick: when you manipulate a variable, you break the causal influence of other variables on it. For example, try breaking the causal chain from A->M.

```
dag_2 <- dagitty( "dag { A -> D <- M}")
coordinates(dag_2) <- list( x=c(A=0,D=1,M=2) , y=c(A=0,D=1,M=0) )
drawdag( dag_2 )
```

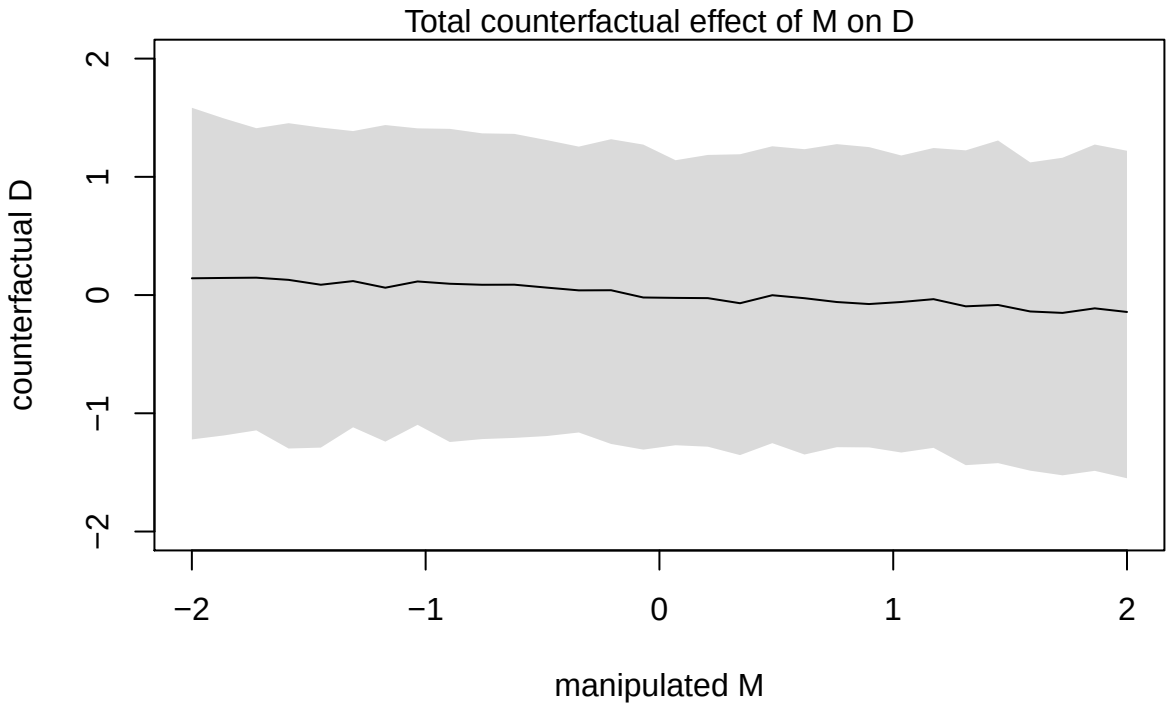



```
sim_dat <- data.frame( M=seq(from=-2,to=2,length.out=30) , A=0 )
s <- sim( m5.3_A , data=sim_dat , vars="D" )

plot( sim_dat$M , colMeans(s) , ylim=c(-2,2) , type="l" ,
      xlab="manipulated M" , ylab="counterfactual D" )
shade( apply(s,2,PI) , sim_dat$M )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```

```
mtext( "Total counterfactual effect of M on D" )
```



Only simulate D, not A since M doesn't influence it - notice the effect is not strong.

5.2 - Masked Relationship

Previous section showed multiple predictors are useful for dealing with spurious correlation. Next, look at measuring direct influences of multiple factors on an outcome when none is apparent from bivariate relationships - occurs when predictors are correlated, one positive, one negative.

New dataset - composition of milk across species.

```
library(rethinking)
data(milk)
d <- milk
str(d)
```

```
## 'data.frame':    29 obs. of  8 variables:
##  $ clade         : Factor w/ 4 levels "Ape","New World Monkey",..: 4 4 4 4 4 2 2 2 2 2 ...
##  $ species       : Factor w/ 29 levels "A palliata","Alouatta seniculus",..: 11 8 9 10 16 2 1 6 28 2
##  $ kcal.per.g    : num  0.49 0.51 0.46 0.48 0.6 0.47 0.56 0.89 0.91 0.92 ...
##  $ perc.fat      : num  16.6 19.3 14.1 14.9 27.3 ...
##  $ perc.protein  : num  15.4 16.9 16.9 13.2 19.5 ...
##  $ perc.lactose  : num  68 63.8 69 71.9 53.2 ...
##  $ mass          : num  1.95 2.09 2.51 1.62 2.19 5.25 5.37 2.51 0.71 0.68 ...
##  $ neocortex.perc: num  55.2 NA NA NA NA ...
```

Hypothesis - larger brain -> more energetic milk. Look at K, calories per g of milk, M, body mass, and N, neocortex mass percent of the brain.

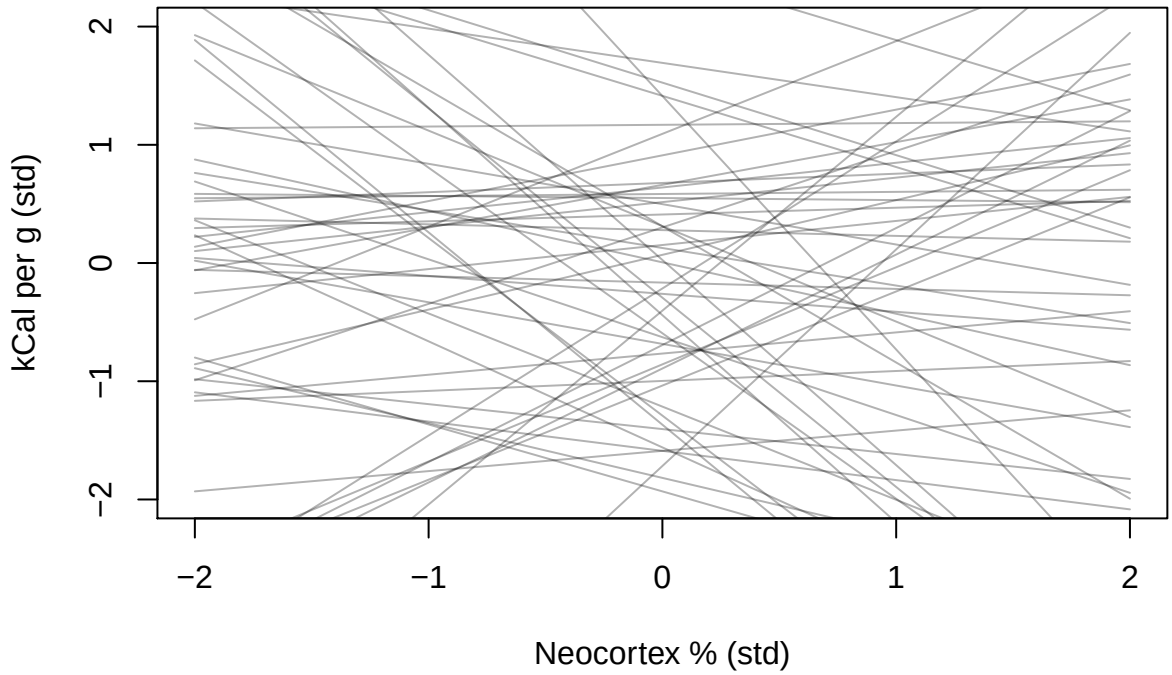
```
d$K <- scale( d$kcal.per.g )
d$N <- scale( d$neocortex.perc )
d$M <- scale( log(d$mass) )
```

Model:

$$K_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta_N N_i$$

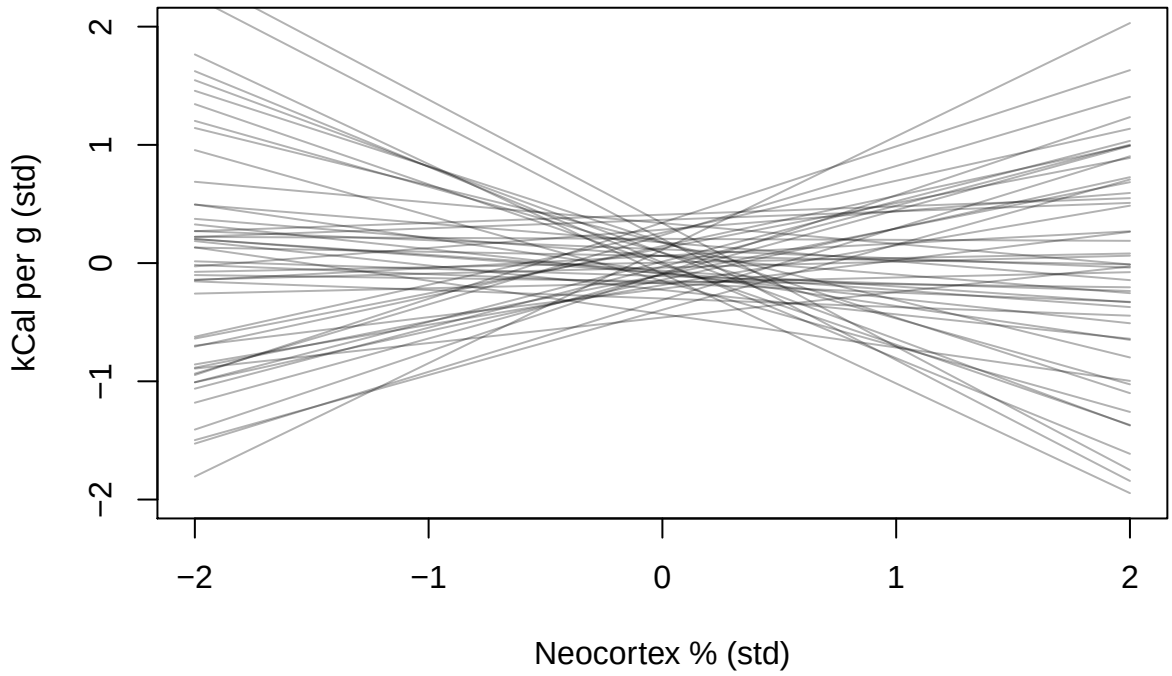
Text asks us to try a model that will fail, giving an error that "initial value in 'vmmin' is not finite." This comes from NANs in neocortex size. We'll do drops now.

```
 dcc <- d[ complete.cases(d$K,d$N,d$M) , ]
m5.5_draft <- quap(
    alist(
      K ~ dnorm( mu , sigma ) ,
      mu <- a + bN*N ,
      a ~ dnorm( 0 , 1 ) ,
      bN ~ dnorm( 0 , 1 ) ,
      sigma ~ dexp( 1 )
) , data=dcc )
prior <- extract.prior( m5.5_draft )
xseq <- c(-2,2)
mu <- link( m5.5_draft , post=prior , data=list(N=xseq) )
plot( NULL , xlim=xseq , ylim=xseq ,
      xlab="Neocortex % (std)", ylab="kCal per g (std)")
for ( i in 1:50 ) lines( xseq , mu[i,] , col=col.alpha("black",0.3) )
```



Crazy priors, let's tighten them.

```
m5.5 <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bN*N ,
    a ~ dnorm( 0 , 0.2 ) ,
    bN ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
) , data=dcc )
prior <- extract.prior( m5.5 )
xseq <- c(-2,2)
mu <- link( m5.5 , post=prior , data=list(N=xseq) )
plot( NULL , xlim=xseq , ylim=xseq ,
      xlab="Neocortex % (std)", ylab="kCal per g (std)")
for ( i in 1:50 ) lines( xseq , mu[i,] , col=col.alpha("black",0.3) )
```



Better, now looking at posterior:

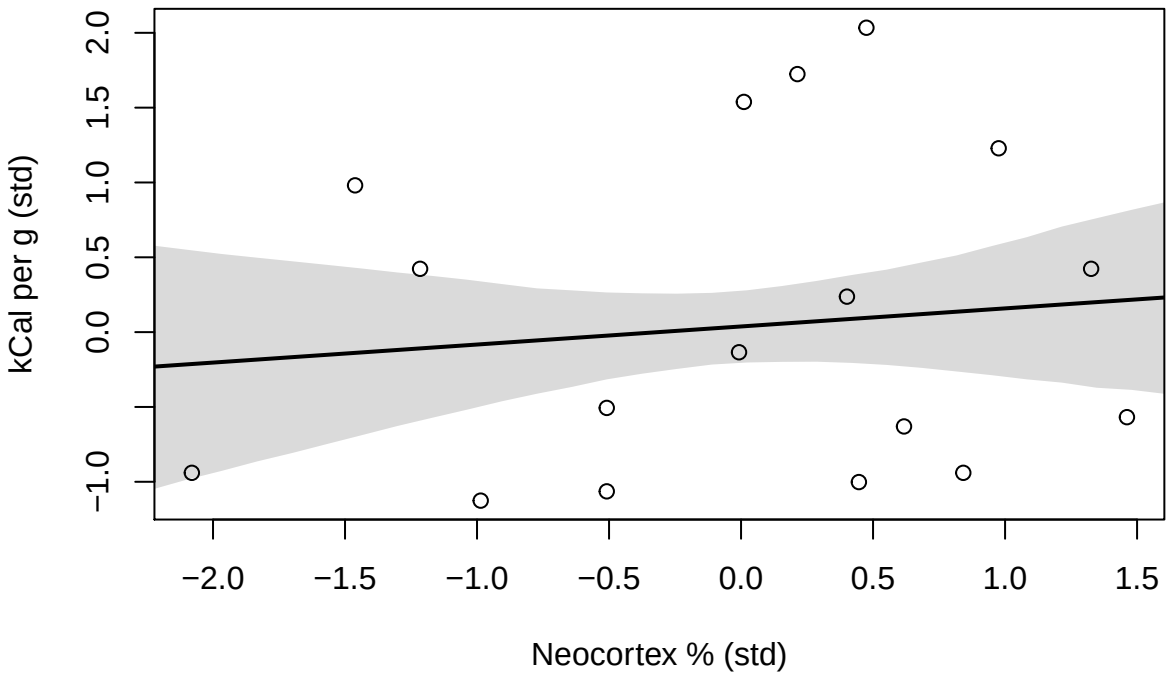
```
precis( m5.5 )
```

```
##             mean        sd       5.5%     94.5%
## a     0.03993788 0.1544907 -0.2069680 0.2868438
## bN    0.13323392 0.2237465 -0.2243562 0.4908241
## sigma 0.99981830 0.1647072  0.7365844 1.2630523
```

Some things to takeaway: neither strong nor precise, std twice mean.

```
xseq <- seq( from=min(dcc$N)-0.15 , to=max(dcc$N)+0.15 , length.out=30 )
mu <- link( m5.5 , data=list(N=xseq) )
mu_mean <- apply(mu,2,mean)
mu_PI <- apply(mu,2,PI)
plot( K ~ N , data=dcc ,xlab="Neocortex % (std)", ylab="kCal per g (std)")
lines( xseq , mu_mean , lwd=2 )
shade( mu_PI , xseq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```



Also look at kCal v body mass independently

```
m5.6 <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bM*M ,
    a ~ dnorm( 0 , 0.2 ) ,
    bM ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
) , data=dcc )
precis(m5.6)
```

```
##              mean        sd       5.5%      94.5%
## a      0.04654131 0.1512800 -0.1952334 0.28831597
## bM    -0.28253547 0.1928817 -0.5907977 0.02572676
## sigma  0.94927893 0.1570613  0.6982646 1.20029327
```

```
xseq <- seq( from=min(dcc$M)-0.15 , to=max(dcc$M)+0.15 , length.out=30 )
mu <- link( m5.6 , data=list(M=xseq) )
mu_mean <- apply(mu,2,mean)
mu_PI <- apply(mu,2,PI)
plot( K ~ M , data=dcc ,xlab="log body mass (std)", ylab="kCal per g (std)")
lines( xseq , mu_mean , lwd=2 )
```

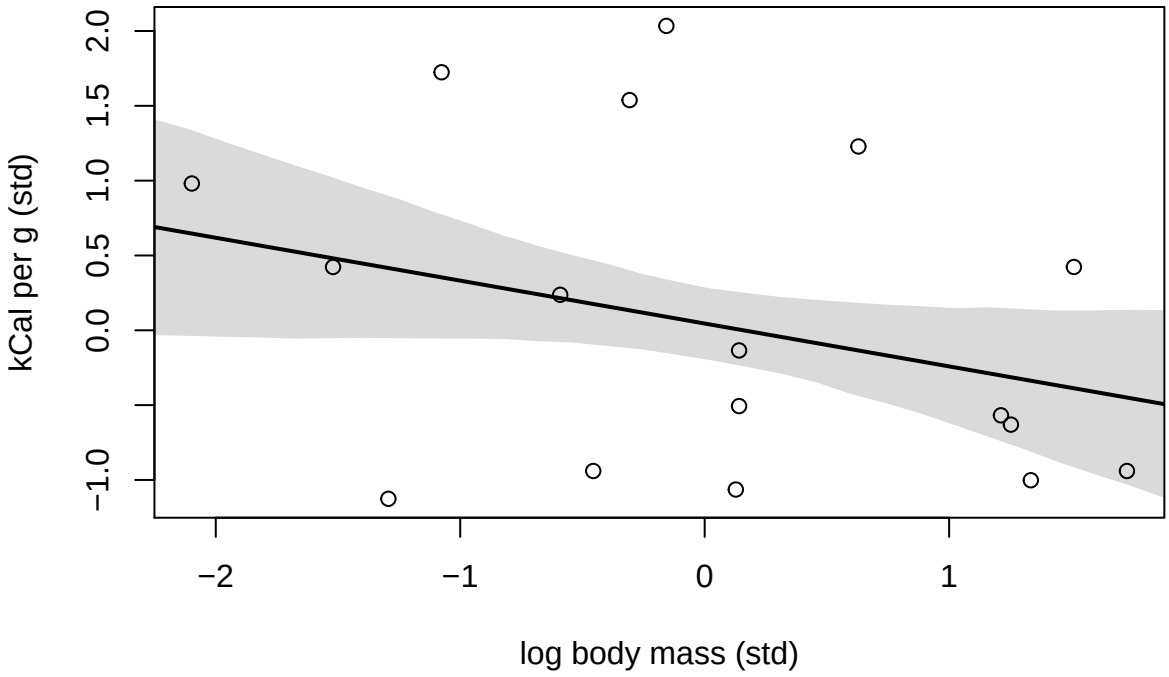
```
shade( mu_PI , xseq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used
```

```
## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```



Multivariate model:

$$
\begin{aligned}
K_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_N N_i + \beta_M M_i \\
\alpha &\sim \text{Normal}(0, 0.2) \\
\beta_n &\sim \text{Normal}(0, 0.5) \\
\beta_m &\sim \text{Normal}(0, 0.5) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

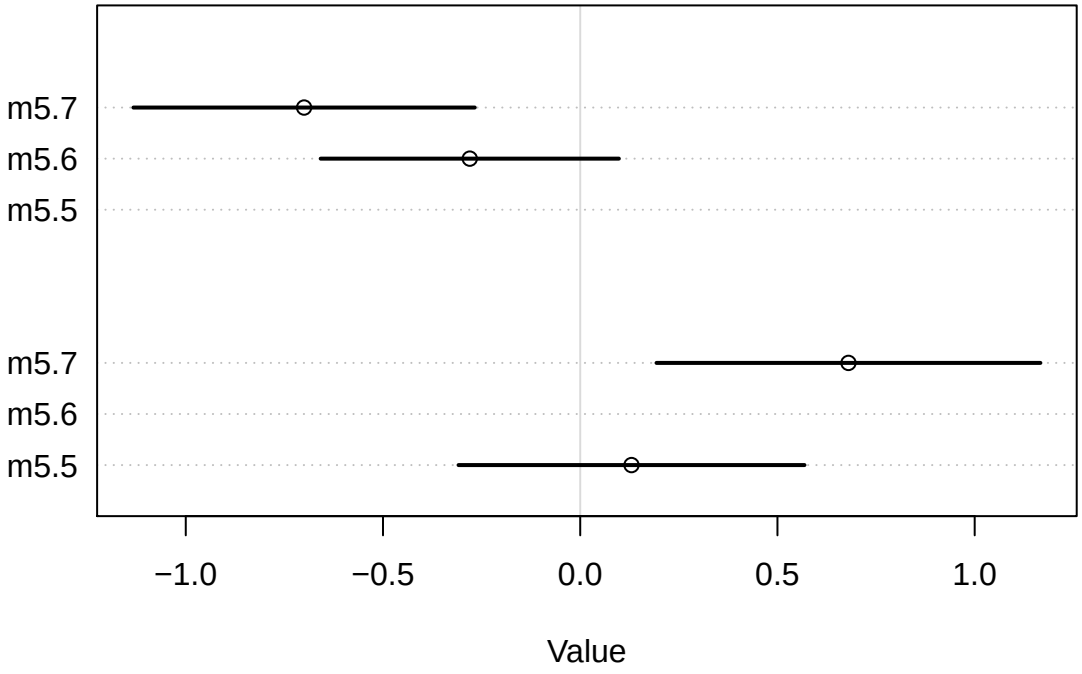
```
m5.7 <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bN*N + bM*M ,
    a ~ dnorm( 0 , 0.2 ) ,
    bN ~ dnorm( 0 , 0.5 ) ,
    bM ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
```

```
) , data=dcc )
precis(m5.7)
```

```
##              mean        sd       5.5%      94.5%
## a      0.06799183 0.1339987 -0.1461639  0.2821476
## bN     0.67511803 0.2482985  0.2782890  1.0719470
## bM    -0.70299097 0.2207870 -1.0558512 -0.3501308
## sigma  0.73801409 0.1324617  0.5263146  0.9497135
```

Posterior of both with the outcome increased.

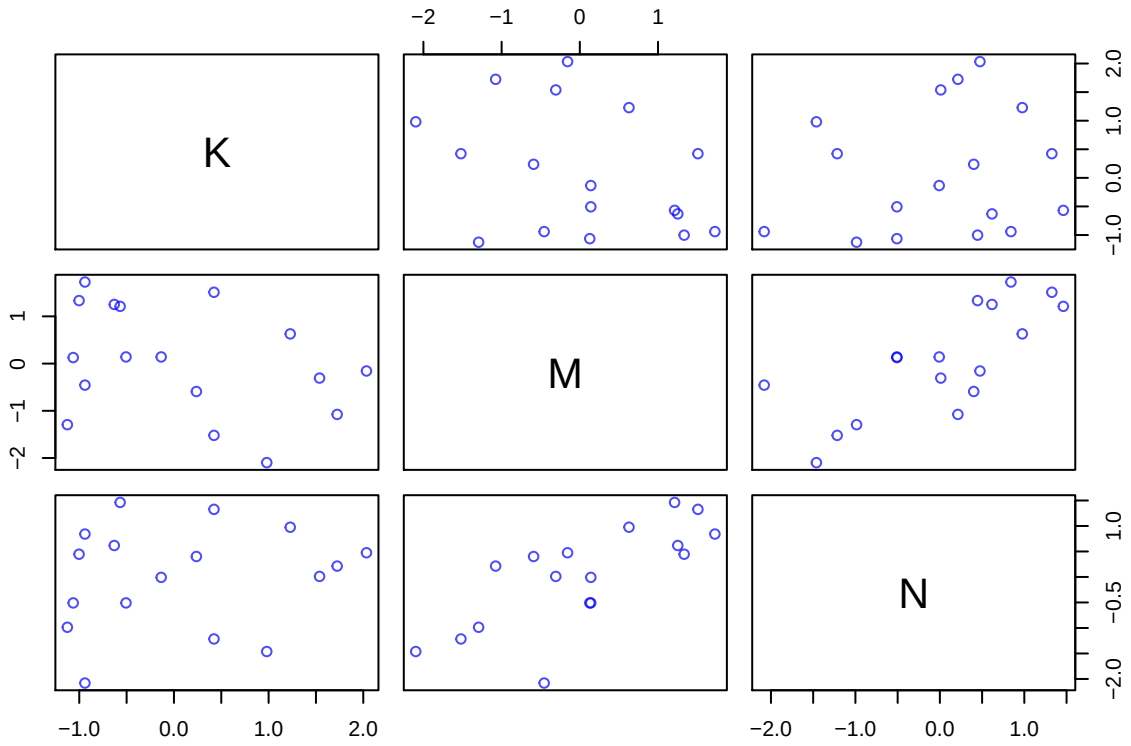
```
plot( coeftab( m5.5 , m5.6 , m5.7 ) , pars=c("bM","bN") )
```



(Top is bM, bottom is bN)

Mean for bM, the neocortex percent, increased by a factor of 5 (bottom), the mean for the log body mass is now much larger too.

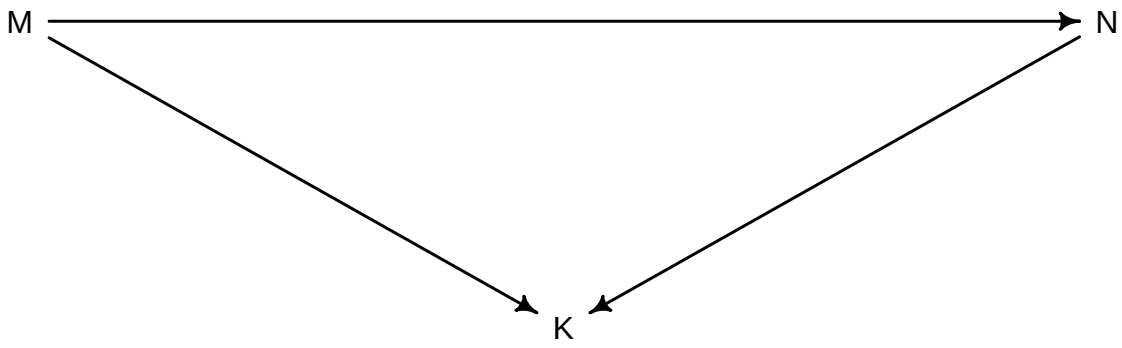
```
pairs( ~K + M + N, dcc, col=col.alpha("blue",0.7))
```



We can see that M and N are positively correlated to each other, so they cancel out. Multiple regression is helpful - asking if a species has a high neocortex percent *for their body mass*; similarly if they have a high body mass *for their neocortex percent*.

Three DAGs we'll explore:

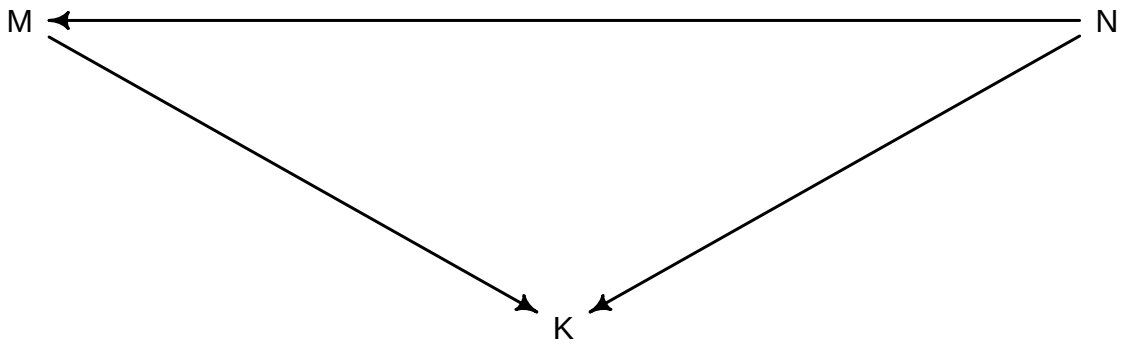
```
dag1 <- dagitty( "dag {
  M -> N
  M -> K
  N -> K
}")
coordinates(dag1) <- list( x=c(M=0,K=1,N=2) , y=c(M=0,K=1,N=0) )
drawdag( dag1 )
```



Body mass influences neocortex percent, both influence calories

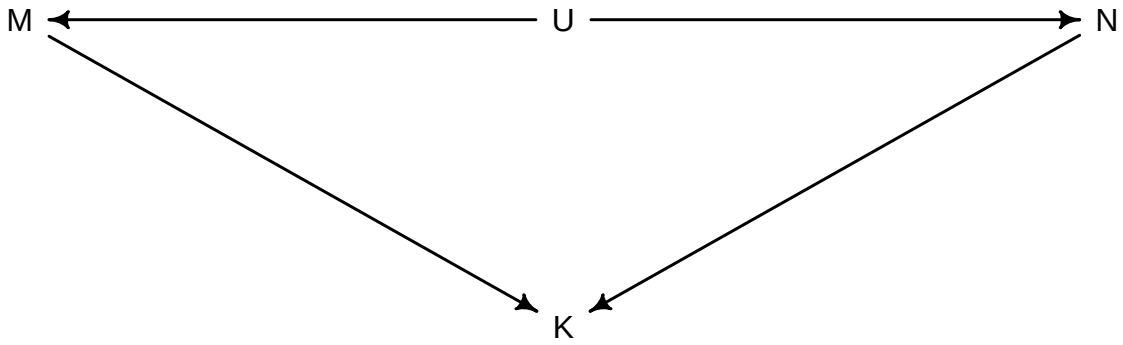
```
dag2 <- dagitty( "dag {
  N -> M
  M -> K
  N -> K
}")
```

```
coordinates(dag2) <- list( x=c(M=0,K=1,N=2) , y=c(M=0,K=1,N=0) )
drawdag( dag2 )
```



Neocortex percent influences body mass, both influence calories.

```
dag3 <- dagitty( "dag {
  M <- U -> N
  M -> K
  N -> K
}")
coordinates(dag3) <- list( x=c(M=0,K=1,U=1,N=2) , y=c(M=0,K=1,U=0,N=0) )
drawdag( dag3 )
```



Some unobserved variable U influences both M and N - unobserved variables will be explored more in next chapter.

Can't tell which of the 3 is correct because the *conditional dependencies* are the same between them; each suggests all variables are associated regardless of what we condition on - this is known as a "Markov Equivalence set."

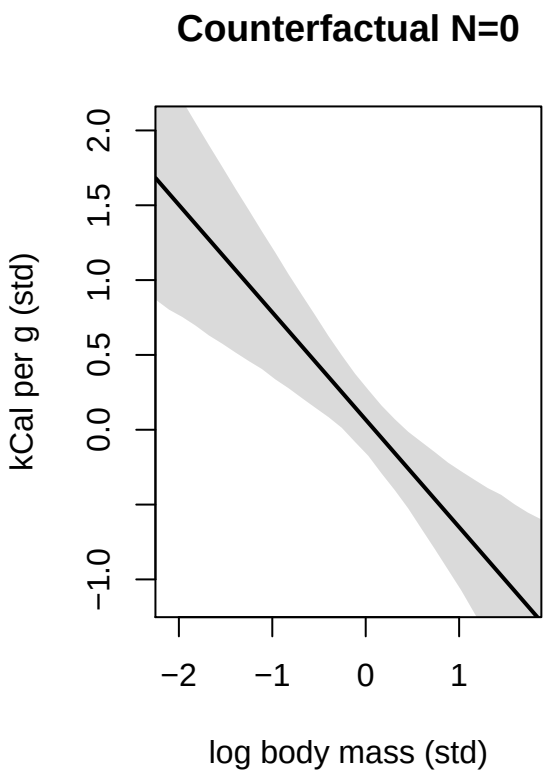
Last some counterfactual plots:

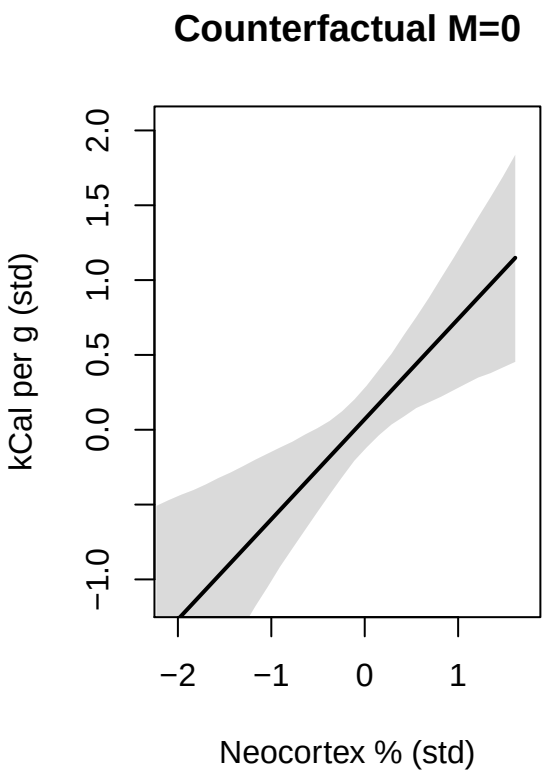
```
# Plot 1
xseq <- seq( from=min(dcc$M)-0.15 , to=max(dcc$M)+0.15 , length.out=30 )
mu <- link( m5.7 , data=data.frame( M=xseq , N=0 ) )
mu_mean <- apply(mu,2,mean)
mu_PI <- apply(mu,2,PI)

par(mfrow=c(1,2))
plot( NULL , xlim=range(dcc$M) , ylim=range(dcc$K),
      xlab="log body mass (std)", ylab="kCal per g (std)", main="Counterfactual N=0")
lines( xseq , mu_mean , lwd=2)
shade( mu_PI , xseq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```

```
#plot2
xseq <- seq( from=min(dcc$N)-0.15 , to=max(dcc$N)+0.15 , length.out=30 )
mu <- link( m5.7 , data=data.frame( N=xseq , M=0 ) )
mu_mean <- apply(mu,2,mean)
mu_PI <- apply(mu,2,PI)

plot( NULL , xlim=range(dcc$M) , ylim=range(dcc$K),
           xlab="Neocortex % (std)", ylab="kCal per g (std)", main="Counterfactual M=0")
lines( xseq , mu_mean , lwd=2 )
shade( mu_PI , xseq )
```

```
## Warning in if (class(object) == "formula") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "density") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (class(object) == "matrix" & length(dim(object)) == 2) {: the
## condition has length > 1 and only the first element will be used

## Warning in if (class(object) == "matrix") {: the condition has length > 1 and
## only the first element will be used
```





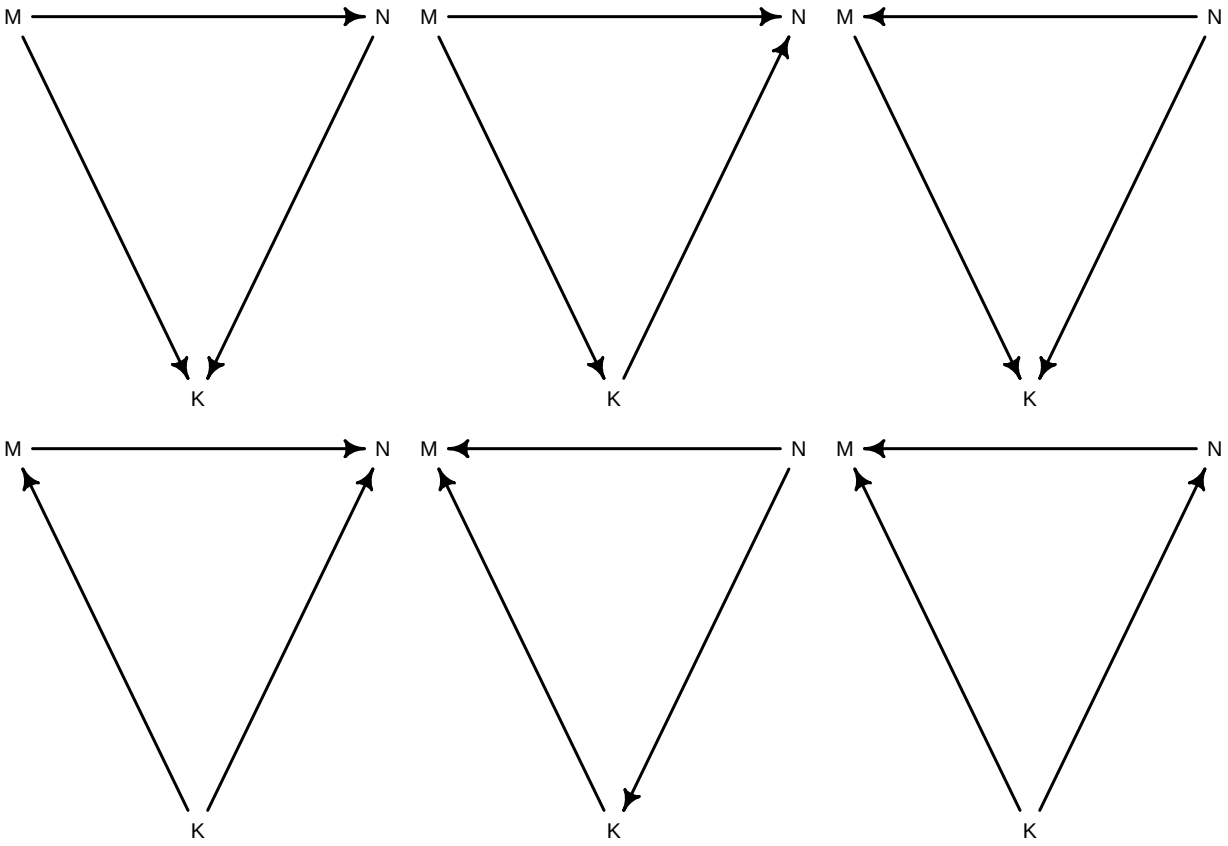
Overthinking section - simulating DAGs

```
# M -> K <- N
# M -> N
n <- 100
M <- rnorm( n )
N <- rnorm( n , M )
K <- rnorm( n , N - M )
d_sim <- data.frame(K=K,N=N,M=M)

# M -> K <- N
# N -> M
n <- 100
N <- rnorm( n )
M <- rnorm( n , N )
K <- rnorm( n , N-M )
d_sim2 <- data.frame(K=K,N=N,M=M)

# M -> K <- N
# M <- U -> N
n <- 100
U <- rnorm( n )
N <- rnorm( n , U )
M <- rnorm( n , U )
K <- rnorm( n , N-M)
d_sim3 <- data.frame(K=K,N=N,M=M)
```

This code simulates data from all 3 dags into dataframes

```
dag5.7 <- dagitty( "dag{
  M -> K <- N
```

```
  M -> N }" )
coordinates(dag5.7) <- list( x=c(M=0,K=1,N=2) , y=c(M=0.5,K=1,N=0.5) )
MElist <- equivalentDAGs(dag5.7)
drawdag(MElist)
```



All possible Markov Equivalent dags.

5.3 - Categorical Variables

Also known as *factors*, can be put into linear models.

Binary categories

Male/female example

```
data(Howell1)
d <- Howell1
str(d)
```

```
## 'data.frame':    544 obs. of  4 variables:
##  $ height: num  152 140 137 157 145 ...
##  $ weight: num  47.8 36.5 31.9 53 41.3 ...
##  $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
##  $ male  : int  1 0 0 1 0 1 0 1 0 1 ...
```

"male" is an *indicator* or dummy variable, indicates the category. Make a model focused on sex, where m is male indicator.

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_m m_i \\
\alpha &\sim \text{Normal}(178, 20) \\
\beta_m &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

So for males you get a linear model, $\mu_i = \alpha + \beta_m m_i$ and for females you get just $\mu_i = \alpha$. This makes β_m the expected difference between males and females; α no longer interpreted as average sample height but average female height.

```
mu_female <- rnorm(1e4,178,20)
mu_male <- rnorm(1e4,178,20) + rnorm(1e4,0,10)
#precis( data.frame( mu_female , mu_male ) )
```

Male prior wider than female, but we're not less certain, so use an index variable instead

```
d$sex <- ifelse( d$male==1 , 2 , 1 )
str( d$sex )
```

```
##  num [1:544] 2 1 1 2 1 2 1 2 1 2 ...
```

for 1 female, 2 male, making the model:

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{SEX}[i]} \\
\alpha_j &\sim \text{Normal}(178, 20), \text{for } j = 1..2 \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

Making the model:

```
m5.8 <- quap(
  alist(
    height ~ dnorm( mu , sigma ) ,
    mu <- a[sex] ,
    a[sex] ~ dnorm( 178 , 20 ) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d )
precis( m5.8 , depth=2 ) #depth=2 -> show vectors
```

```
##            mean        sd      5.5%     94.5%
## a[1]  134.91135 1.6069053 132.34320 137.47949
## a[2]  142.57804 1.6974422 139.86520 145.29088
## sigma  27.30946 0.8280046  25.98615  28.63277
```

```
post <- extract.samples(m5.8)
post$diff_fm <- post$a[,1] - post$a[,2]
#precis( post , depth=2)
```

precis output will have a diff_fm row - *contrast*.

Many categories

Going back to milk example:

```
data(milk)
d <- milk
unique(d$clade)
```

```
## [1] Strepsirrhine    New World Monkey Old World Monkey Ape
## Levels: Ape New World Monkey Old World Monkey Strepsirrhine
```

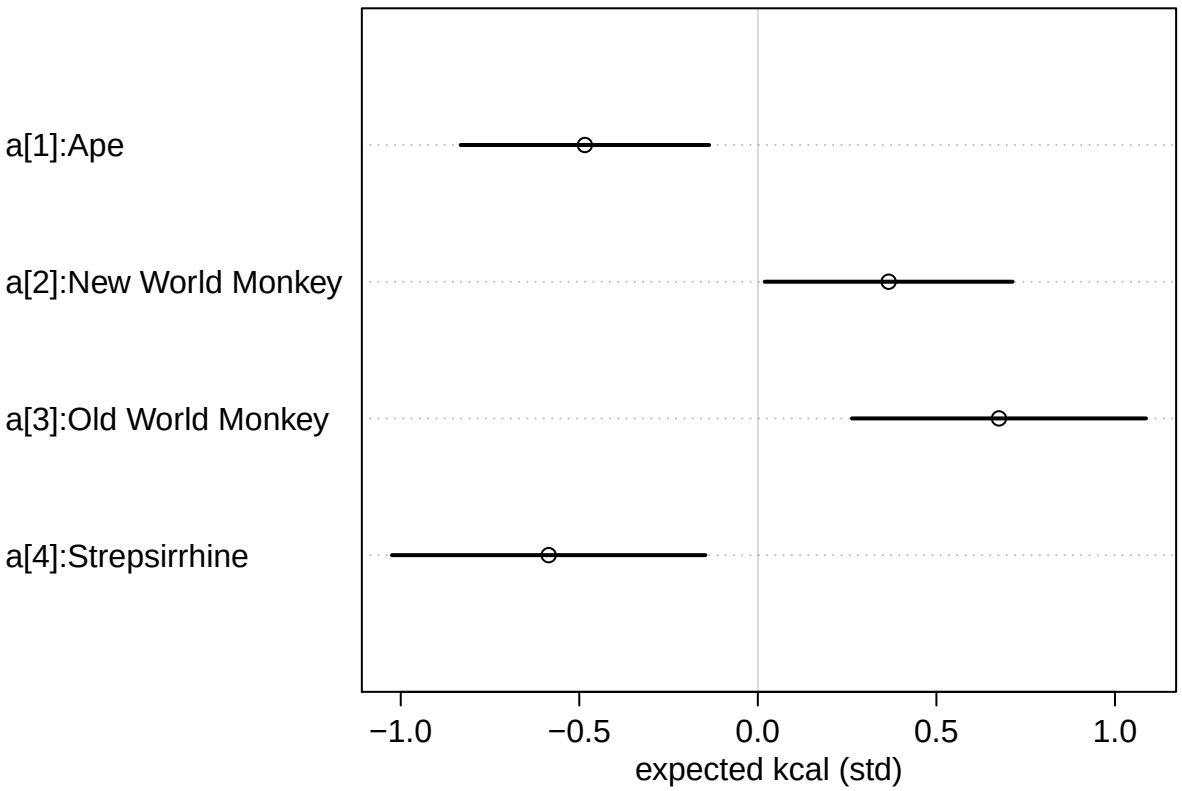
auto-index categories

```
d$clade_id <- as.integer( d$clade)
```

Model:

$$
\begin{aligned}
K_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha_{\text{CLADE}[i]} \\
\alpha_j &\sim \text{Normal}(0, 0.5), \text{for } j = 1..4 \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

```
d$K <- scale( d$kcal.per.g )
m5.9 <- quap(
  alist(
    K ~ dnorm( mu , sigma ),
    mu <- a[clade_id],
    a[clade_id] ~ dnorm( 0 , 0.5 ),
    sigma ~ dexp( 1 )
  ) , data=d )
labels <- paste( "a[" , 1:4 , "]:" , levels(d$clade) , sep="" )
plot( precis( m5.9 , depth=2 , pars="a" ) , labels=labels ,
    xlab="expected kcal (std)" )
```



Rethinking block - common error in interpretation of parameter estimates is that because a parameter is far from zero and another isn't (is/isn't significant) that the difference is significant. If you want to know the distribution of the difference, calculate the contrast between the two parameters, not investigate independently.

5.4 - Summary

Multiple regression can construct models with more than one predictor - letting us answer the question "what is the value of knowing each predictor, once others are known."

1. Focuses on value of predictors as a description, not a forecast only
2. Assumption that value of predictors do not depend on values of other predictors (later confronted)

Chapter 6 - The Haunted DAG & The Causal Terror

Take two uncorrelated variables (newsworthiness and trustworthiness), select the top 10%, and draw a linear regression and there will be be a negative correlation - *Berkson's Paradox* or *selection-distortion* effect.

This can happen within multiple regression, if adding a predictor induces selection, known as *collider bias*.

In this chapter - warnings about adding random variables without an idea of a causal model: multicollinearity, post-treatment bias, collider bias.

6.1 Multicollinearity

"Very strong correlation between two or more predictor variables"

Technically nothing wrong with it, will work well for prediction, just hard to understand.

Return to primate milk data for data.

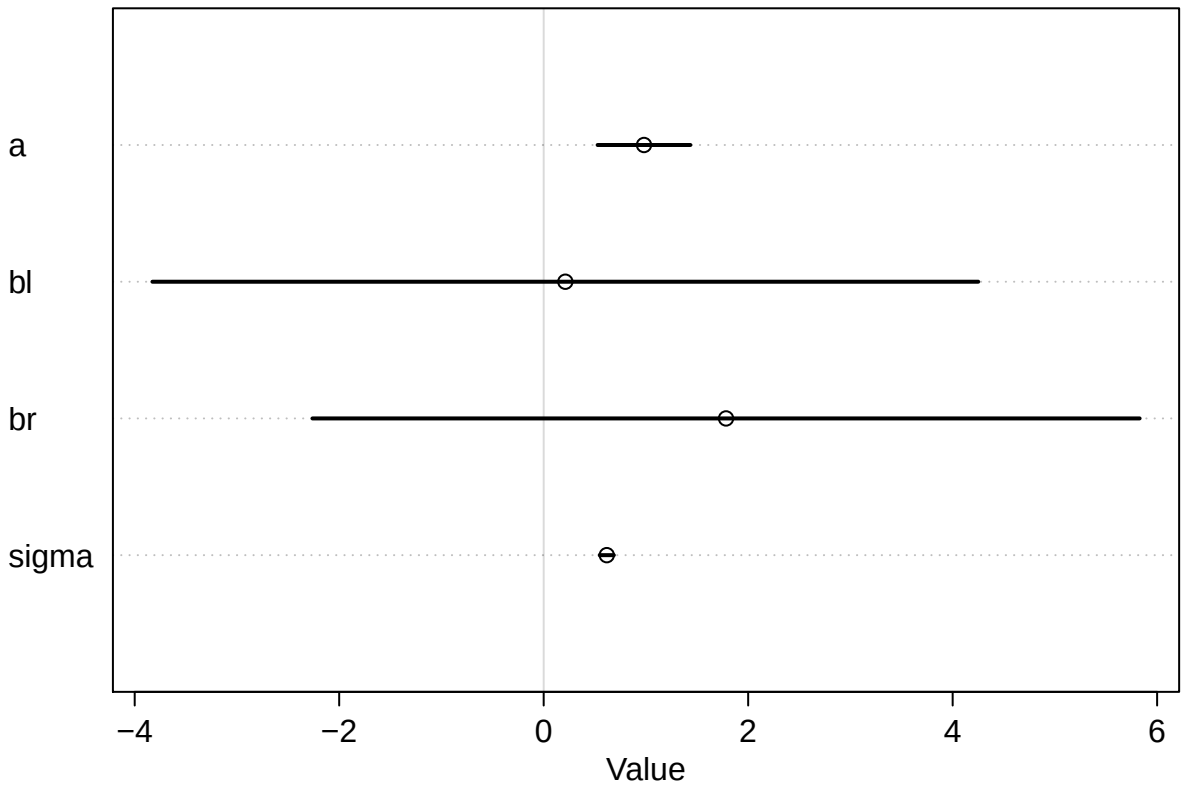
Multicollinear legs

Using both leg lengths (left and right) as a predictor

```
# Generate dataframe
N <- 100
set.seed(909)
height <- rnorm(N,10,2)
leg_prop <- runif(N,0.4,0.5)
leg_left <- leg_prop*height +
  rnorm( N , 0 , 0.02 )
leg_right <- leg_prop*height +
  rnorm( N , 0 , 0.02 )
d <- data.frame(height,leg_left,leg_right)

# Make model
m6.1 <- quap(
  alist(
    height ~ dnorm( mu , sigma ) ,
    mu <- a + bl*leg_left + br*leg_right ,
    a ~ dnorm( 10 , 100 ) ,
    bl ~ dnorm( 2 , 10 ) ,
    br ~ dnorm( 2 , 10 ) ,
    sigma ~ dexp( 1 )
  ), data=d )
precis(m6.1)
```

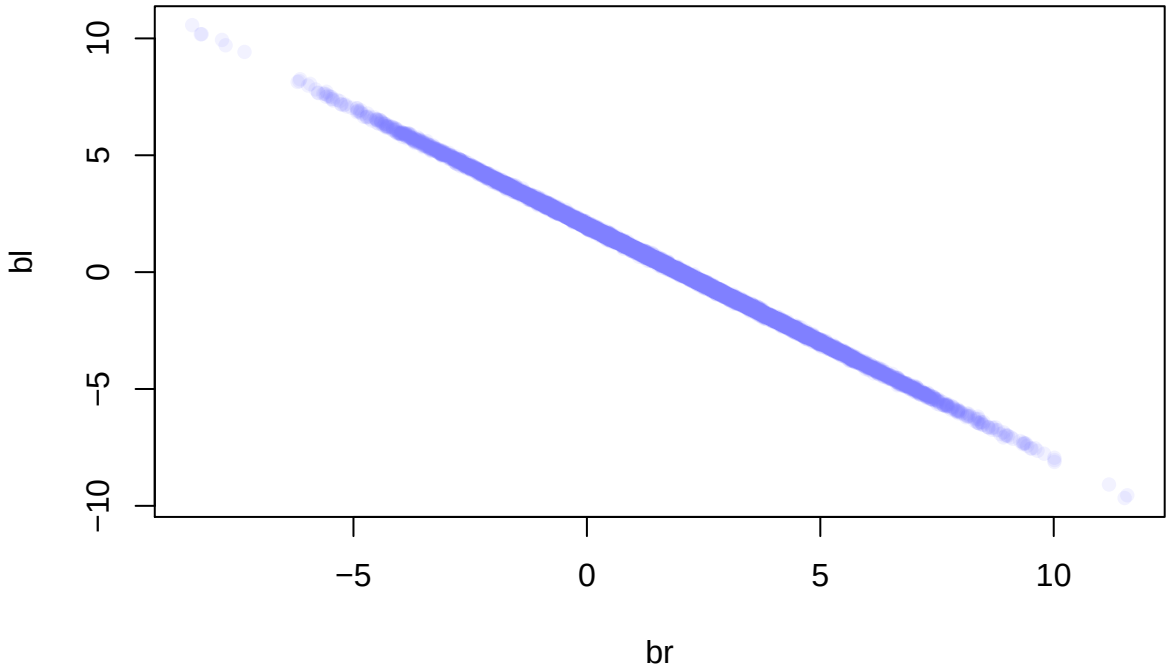
```
##            mean         sd       5.5%     94.5%
## a     0.9812791 0.28395540  0.5274635 1.4350947
## bl    0.2118585 2.52703706 -3.8268348 4.2505518
## br    1.7836774 2.53125061 -2.2617500 5.8291047
## sigma 0.6171026 0.04343427  0.5476862 0.6865189
```

```
plot(precis(m6.1))
```



Gigantic means and standard deviations. Question posed in last chapter is "what's the value of knowing a second leg length after knowing the first?"

```
post <- extract.samples(m6.1)
plot( bl ~ br , post , col=col.alpha(rangi2,0.1) , pch=16 )
```



Posterior for left vs right very highly correlated - as β_l is large, β_r is small, they carry the same variation.

Our model development:

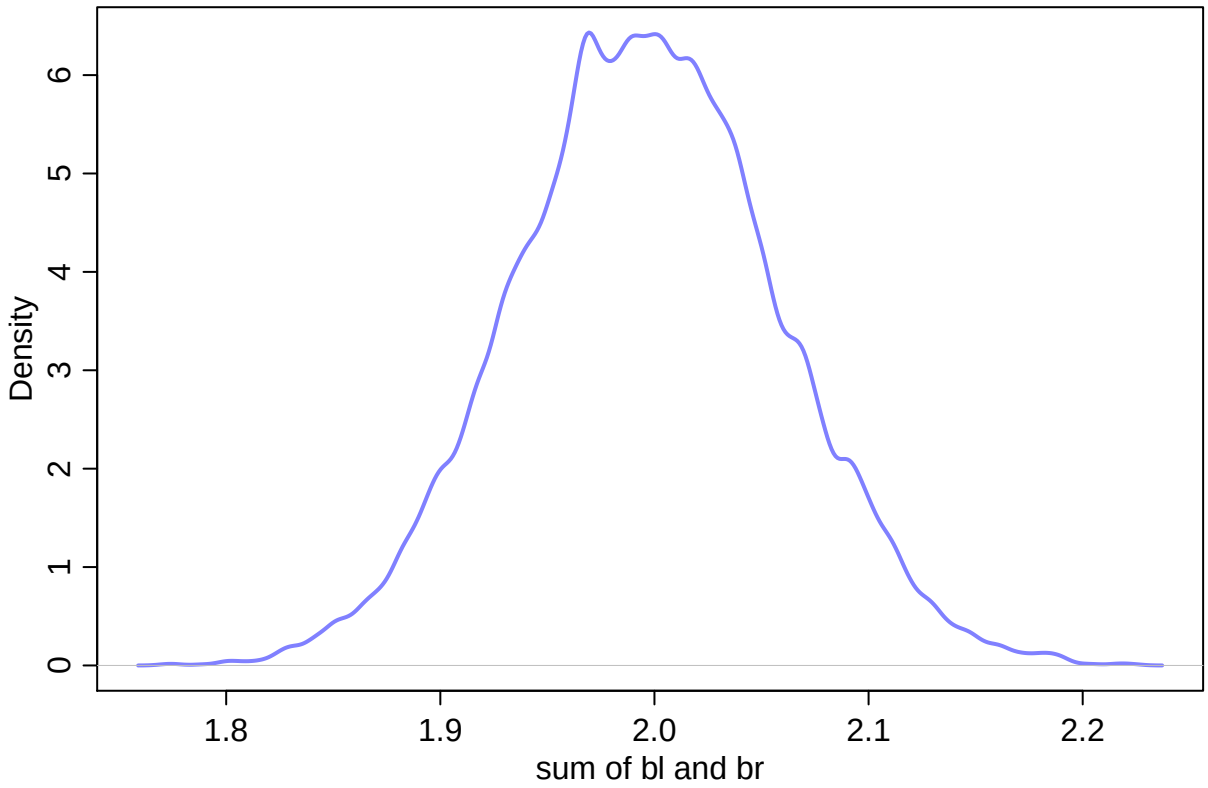
$$y_i \sim \text{Normal}(\mu, \sigma)$$
$$\mu_i = \alpha + \beta_1 x_l + \beta_2 x_r$$

But due to covariation (x is basically the same thing), the computer basically sees:

$$y_i \sim \text{Normal}(\mu, \sigma)$$
$$\mu_i = \alpha + (\beta_1 + \beta_2) x_i$$

As in only the sum of $\beta_1 + \beta_2$ influences μ.

```
sum_blbr <- post$bl + post$br
dens( sum_blbr , col=rangi2 , lwd=2 , xlab="sum of bl and br" )
```



Using just one predictor:

```
m6.2 <- quap(
  alist(
    height ~ dnorm( mu , sigma ) ,
    mu <- a + bl*leg_left,
    a ~ dnorm( 10 , 100 ) ,
    bl ~ dnorm( 2 , 10 ) ,
    sigma ~ dexp( 1 )
  ),
```

```
  data=d )
precis(m6.2)
```

```
##            mean         sd      5.5%    94.5%
## a     0.9979326 0.28364620 0.5446112 1.451254
## bl    1.9920676 0.06115704 1.8943269 2.089808
## sigma 0.6186038 0.04353998 0.5490185 0.688189
```

Which is basically identical to the other model - when the two predictor variables are strongly correlated, including both may lead to confusion.

Multicollinear milk

Legs example is contrived, something more elaborate

```
data(milk)
d <- milk
d$K <- scale( d$kcal.per.g )
d$F <- scale( d$perc.fat )
d$L <- scale( d$perc.lactose )
```

Looking at bivariate regressions:

```
# kcal.per.g regressed on perc.fat
m6.3 <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bF*F ,
    a ~ dnorm( 0 , 0.2 ) ,
    bF ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )

# kcal.per.g regressed on perc.lactose
m6.4 <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bL*L ,
    a ~ dnorm( 0 , 0.2 ) ,
    bL ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )

m6.5 <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bF*F + bL*L ,
    a ~ dnorm( 0 , 0.2 ) ,
    bF ~ dnorm( 0 , 0.5 ) ,
    bL ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ),data=d )

precis( m6.3 )
```

```
##                mean         sd       5.5%     94.5%
## a      1.535526e-07 0.07725195 -0.1234634 0.1234637
## bF     8.618970e-01 0.08426088  0.7272318 0.9965621
## sigma  4.510179e-01 0.05870756  0.3571919 0.5448440
```

```
precis( m6.4 )
```

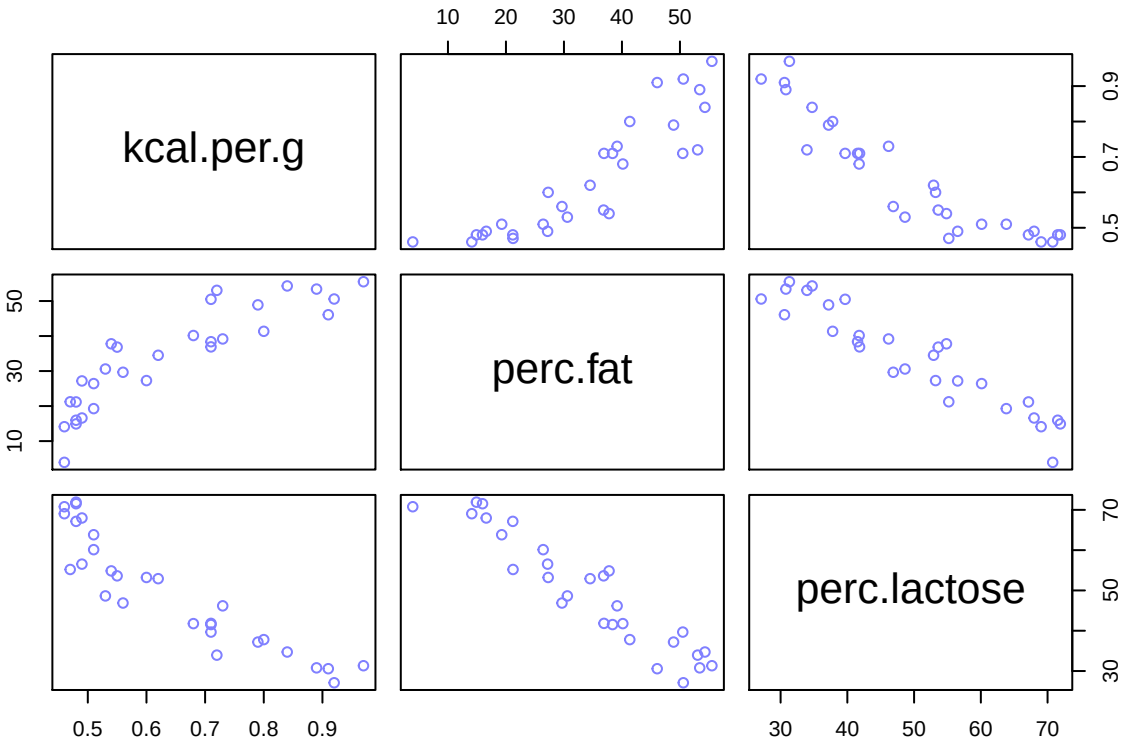
```
##                mean         sd       5.5%      94.5%
## a       7.438895e-07 0.06661633 -0.1064650  0.1064665
## bL     -9.024550e-01 0.07132848 -1.0164517 -0.7884583
## sigma   3.804653e-01 0.04958259  0.3012227  0.4597078
```

```
precis( m6.5 )
```

```
##                mean         sd        5.5%      94.5%
## a      -3.172136e-07 0.06603577 -0.10553823  0.1055376
## bF      2.434983e-01 0.18357865 -0.04989579  0.5368925
## bL     -6.780825e-01 0.18377670 -0.97179320 -0.3843719
## sigma   3.767418e-01 0.04918394  0.29813637  0.4553472
```

Percent fat (perc.fat) and percent lactose (perc.lactose) are mirrors of each other (m6.3 and 6.4). Putting both in a model (6.5) gives posterior means of both closer to 0 than either individually - mutual information.

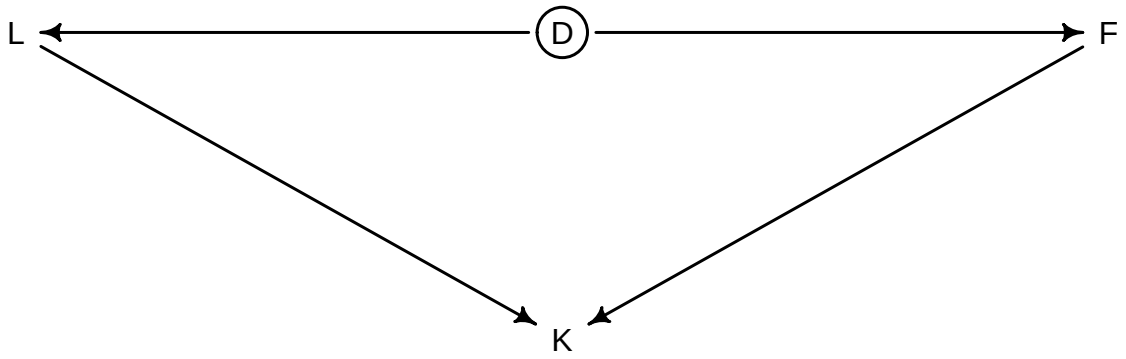
```
pairs( ~ kcal.per.g + perc.fat + perc.lactose , data=d , col=rangi2 )
```



Very obvious from pairs plot, either helps, but neither does if you already know the other. Causal approach is the best approach.

```
dag0 <- dagitty( "dag {
  D [unobserved]
  L <- D -> F
  L -> K
  F -> K
}")
```

```
coordinates(dag0) <- list( x=c(L=0,D=1,K=1,F=2) , y=c(L=0,D=0,K=1,F=0) )
drawdag( dag0 )
```



Central trade-off is on D, the density of milk (non-observed) - fat and lactose are from that (mediators).

Non-Identifiability - structure of data and model do not make it possible to estimate a parameter's value. Can be coding, more often though just nature making it difficult.

6.2 Post-treatment Bias

Omitted Variable Bias - Mistaken inferences that arise from omitting predictor variables

Post-treatment Bias - Mistaken inference that arise from including variables that are the consequences of others.

Name "post-treatment" stems from experimental design, adding variables to model that are a result of the experiment, not truly independent. E.g. plant growing experiment, outcome of interest is final height but some plants grow fungus during the experiment - should not include this.

```
set.seed(71)
# number of plants
N <- 100
# simulate initial heights
h0 <- rnorm(N,10,2)
# assign treatments and simulate fungus and growth
treatment <- rep( 0:1 , each=N/2 )
fungus <- rbinom( N , size=1 , prob=0.5 - treatment*0.4 )
h1 <- h0 + rnorm(N, 5 - 3*fungus)
# compose a clean data frame
d <- data.frame( h0=h0 , h1=h1 , treatment=treatment , fungus=fungus )
#precis(d)
```

A prior is born

Best approach is to pretend you don't have the data generating process.

For this example, height at $t = 1$ is height than that at $t = 0$, so put parameters on a scale of *proportion* of initial height - allows easier prior setting.

$$h_{1,i} \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = h_{0,i} \times p$$

So $p = 2$ means at $t = 1$ it doubles in height, $p = 1$ means same height. Set this prior to 1, to allow for negative growth, force $p > 0$ since a proportion - use Log-Normal since it's always positive.

```
sim_p <- rlnorm( 1e4 , 0 , 0.25 )
#precis( data.frame(sim_p) )
```

Covers everything from 40% shrinking to 50% growth, centered at 1.

```
m6.6 <- quap(
  alist(
    h1 ~ dnorm( mu , sigma ),
    mu <- h0*p,
    p ~ dlnorm( 0 , 0.25 ),
    sigma ~ dexp( 1 )
), data=d )
precis(m6.6)
```

```
##           mean         sd     5.5%    94.5%
## p     1.426626 0.01760992 1.398482 1.454770
## sigma 1.793286 0.12517262 1.593236 1.993336
```

On average, about 40% growth.

Now the model we said would be bad before - adding fungus growth

$$h_{1,i} \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = h_{0,i} \times p$$
$$p = \alpha + \beta_T T_i + \beta_F F_i$$
$$\alpha \sim \text{Log-Normal}(0, 0.25)$$
$$\beta_T \sim \text{Normal}(0, 0.5)$$
$$\beta_F \sim \text{Normal}(0, 0.5)$$
$$\sigma \sim \text{Exponential}(1)$$

Sample model, priors on slopes may be too flat (between -1 and +1), but otherwise proportion of growth defined.

```
m6.7 <- quap(
  alist(
    h1 ~ dnorm( mu , sigma ),
    mu <- h0 * p,
    p <- a + bt*treatment + bf*fungus,
    a ~ dlnorm( 0 , 0.2 ) ,
    bt ~ dnorm( 0 , 0.5 ),
    bf ~ dnorm( 0 , 0.5 ),
    sigma ~ dexp( 1 )
  ), data=d )
precis(m6.7)
```

```
##               mean         sd        5.5%       94.5%
## a      1.481391468 0.02451069  1.44221865  1.52056429
## bt     0.002412222 0.02986965 -0.04532525  0.05014969
## bf    -0.266718915 0.03654772 -0.32512923 -0.20830860
## sigma  1.408797442 0.09862070  1.25118251  1.56641237
```

Nearly same posterior. β_T is 0, tight interval - not associated. β_F is negative - hurts growth. What happened?

Blocked by consequence

Fungus is a consequence of treatment - "post-treatment variable." Controlling for fungus, model answers the question: "Once we already know if a plant developed fungus, does soil treatment matter?" Which, no, it's 0. But we care about treatment.

```
m6.8 <- quap(
  alist(
    h1 ~ dnorm( mu , sigma ),
    mu <- h0 * p,
    p <- a + bt*treatment,
    a ~ dlnorm( 0 , 0.2 ),
    bt ~ dnorm( 0 , 0.5 ),
    sigma ~ dexp( 1 )
  ), data=d )
precis(m6.8)
```

```
##             mean         sd       5.5%     94.5%
## a     1.38035767 0.02517554 1.34012229 1.4205931
## bt    0.08499924 0.03429718 0.03018573 0.1398128
## sigma 1.74631655 0.12191552 1.55147200 1.9411611
```

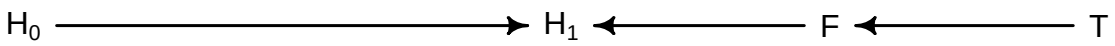
Now the model is a lot more sensible

- Makes sense to control for pre-treatment effects (e.g. initial height) as they might mask causal influence
- Post-treatment inclusion can mask the treatment itself

Fungus and *d*-separation

Draw the DAG:

```
plant_dag <- dagitty( "dag {
  H_0 -> H_1
  F -> H_1
  T -> F
}")
coordinates( plant_dag ) <- list( x=c(H_0=0,T=2,F=1.5,H_1=1) ,
                                  y=c(H_0=0,T=0,F=0,H_1=0) )
drawdag( plant_dag )
```



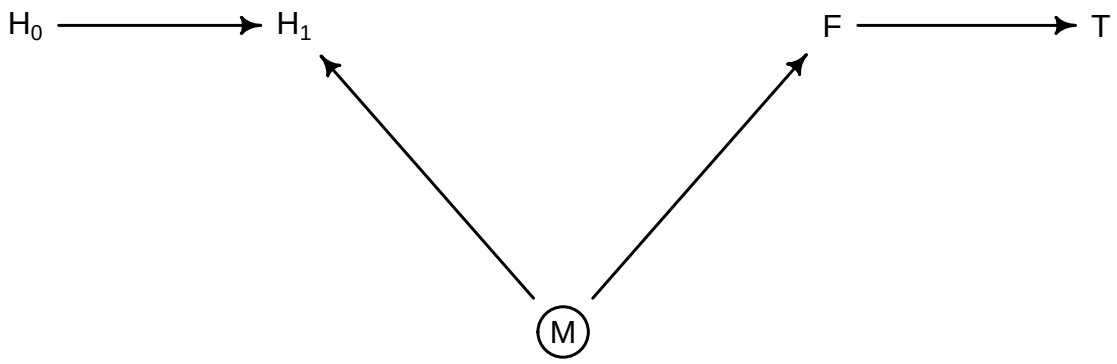
T influences F which influences H_1. By including F, you "block" T, or another way to say this is that conditioning on F induces *d-separation* (d for directional). This means that some variables on a directed graph are independent of others, no path connecting them. Going back to *conditional independencies:*

```
impliedConditionalIndependencies(plant_dag)
```

```
## F _||_ H_0
## H_0 _||_ T
## H_1 _||_ T | F
```

The last one is important - our outcome is independent of T, given conditioning on F.

```
moisture_dag <- dagitty( "dag {
  M [unobserved]
  H_0 -> H_1
  H_1 <- M -> F
  F -> T
}")
coordinates( moisture_dag ) <- list( x=c(H_0=0,T=2,F=1.5,H_1=0.5,M=1) ,
                                     y=c(H_0=0,T=0,F=0,H_1=0,M=0.5) )
drawdag( moisture_dag )
```



Another DAG - Now, an unobserved moisture variable affects both height and fungus growth. In this case (with hypothetical plant unaffected by a fungus), $H1$ and T will appear to be not associated, but then the conditioning on F suddenly fools you into thinking there's an association.

```
set.seed(71)
N <- 1000
h0 <- rnorm(N,10,2)
treatment <- rep( 0:1 , each=N/2 )
M <- rbern(N)
fungus <- rbinom( N , size=1 , prob=0.5 - treatment*0.4 + 0.4*M )
h1 <- h0 + rnorm( N , 5 + 3*M )
d2 <- data.frame( h0=h0 , h1=h1 , treatment=treatment , fungus=fungus )

m6.7_2 <- quap(
  alist(
    h1 ~ dnorm( mu , sigma ),
    mu <- h0 * p,
    p <- a + bt*treatment + bf*fungus,
    a ~ dlnorm( 0 , 0.2 ) ,
    bt ~ dnorm( 0 , 0.5 ),
    bf ~ dnorm( 0 , 0.5 ),
    sigma ~ dexp( 1 )
  ), data=d2 )
precis(m6.7_2)
```

```
##             mean         sd       5.5%      94.5%
## a     1.52211420 0.01360385 1.50037263 1.54385578
## bt    0.04859313 0.01415624 0.02596872 0.07121754
```

```
## bf    0.14276270 0.01415774 0.12013590 0.16538949
## sigma 2.10262855 0.04694249 2.02760537 2.17765172
```

```
m6.8_2 <- quap(
  alist(
    h1 ~ dnorm( mu , sigma ),
    mu <- h0 * p,
    p <- a + bt*treatment,
    a ~ dlnorm( 0 , 0.2 ),
    bt ~ dnorm( 0 , 0.5 ),
    sigma ~ dexp( 1 )
  ), data=d2 )
precis(m6.8_2)
```

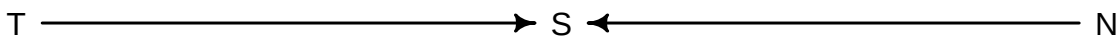
```
##              mean          sd        5.5%      94.5%
## a      1.62401319 0.009546625  1.60875584 1.63927054
## bt    -0.01051596 0.013511945 -0.03211066 0.01107874
## sigma  2.20520300 0.049231869  2.12652096 2.28388504
```

Next section fleshes this effect out more.

6.3 Collider Bias

The initial claim in the chapter of negative correlation between newsworthiness and trustworthiness resulting from selection process is a type of *collider bias*.

```
collider_dag <- dagitty( "dag {
  T -> S <- N
}")
coordinates( collider_dag ) <- list( x=c(T=0,S=1,N=2) ,
                                     y=c(T=0,S=0,N=0) )
drawdag( collider_dag )
```

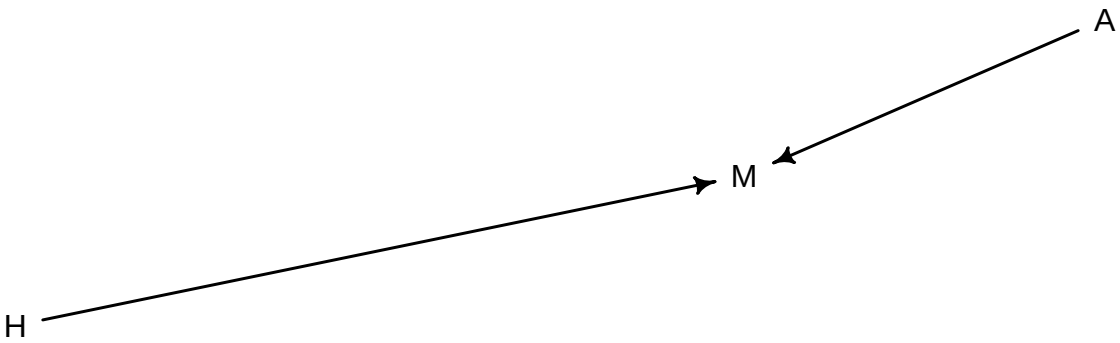


A "collider" is when 2 arrows enter, as in S for that DAG. Conditioning on colliders have statistical, not causal, associations between causes - learning a proposal was selected for trustworthiness provides information about newsworthiness if you know a proposal was selected (if low T, high N).

Collider of false sorrow

Turn to problem of happiness and age. Say happiness, H, is a fixed parameter at birth, but influences events - happier is more likely to get married M. Age, A, is also correlated to marriage (older = more likely), so get same collider dag:

```
collider_dag <- dagitty( "dag {
  H -> M <- A
}")
coordinates( collider_dag ) <- list( x=c(T=0,S=1,N=2) ,
                                     y=c(T=0,S=0,N=0) )
drawdag( collider_dag )
```



Even though no causal association between happiness and age, if conditioned on marriage, induces an association.

```
library(rethinking)
d <- sim_happiness( seed=1977 , N_years=1000 )
#precis(d)
```

Simulation of that data: 1300 people from 0-65. Want to ask if age is related to happiness - try multiple regression with linear model:

$$\mu_i = \alpha_{\text{MID}[i]} + \beta_A A_i$$

MID here is marriage status, 1 is single and 2 is married. Only want older than 18 years old, so correct it 0 to 1, 0 is 18 and 1 is 65. Happiness is arbitrary between -2 and 2.

```
d2 <- d[ d$age>17 , ] # only adults
d2$A <- ( d2$age - 18 ) / ( 65 - 18 )

d2$mid <- d2$married + 1
m6.9 <- quap(
  alist(
    happiness ~ dnorm( mu , sigma ),
    mu <- a[mid] + bA*A,
    a[mid] ~ dnorm( 0 , 1 ),
    bA ~ dnorm( 0 , 2 ),
    sigma ~ dexp(1)
  ) , data=d2 )
precis(m6.9,depth=2)
```

```
##             mean         sd       5.5%      94.5%
## a[1]  -0.2350877 0.06348986 -0.3365568 -0.1336186
## a[2]   1.2585517 0.08495989  1.1227694  1.3943340
## bA    -0.7490274 0.11320112 -0.9299447 -0.5681102
## sigma  0.9897080 0.02255800  0.9536559  1.0257600
```

This model says age is negatively associated with happiness - try omitting marriage

```
m6.10 <- quap(
  alist(
    happiness ~ dnorm( mu , sigma ),
    mu <- a + bA*A,
    a ~ dnorm( 0 , 1 ),
    bA ~ dnorm( 0 , 2 ),
    sigma ~ dexp(1) ) ,
```

```
  data=d2 )
precis(m6.10)
```

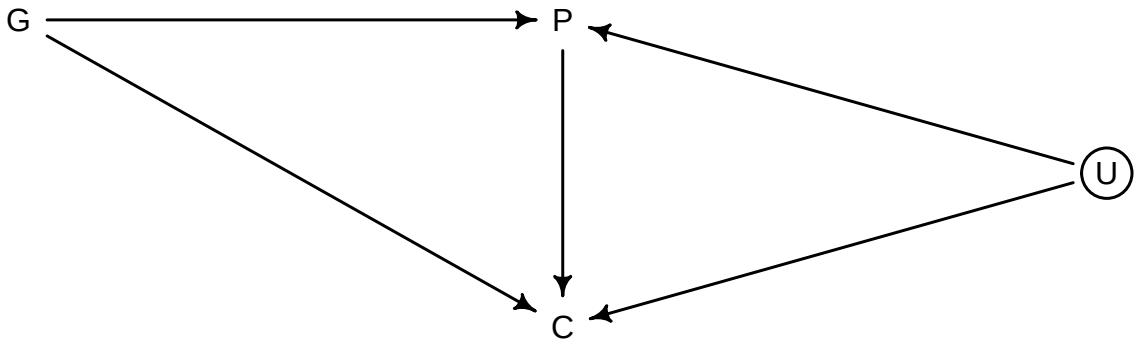
```
##                mean         sd       5.5%     94.5%
## a      1.649248e-07 0.07675015 -0.1226614 0.1226617
## bA    -2.728620e-07 0.13225976 -0.2113769 0.2113764
## sigma  1.213188e+00 0.02766080  1.1689803 1.2573949
```

No association if marriage not included. Highlights selection bias - more people get married as time goes on, so mean among married people approaches population average of zero; inverse for unmarried, happier people migrate over to married population.

The haunted DAG

Not always easy to see a collider because of unmeasured causes, which can still induce it. Example - infer influence of parents P and grandparents G on education of children C. Think a simple dag where G influences P which both influence C. But possibly there's an unobservable U confounder that's not shared by grandparents - say a neighborhood effect after parents moved

```
parent_dag <- dagitty( "dag {
  U [unobserved]
  C <- G -> P
  P -> C
  P <- U -> C
}")
coordinates( parent_dag ) <- list( x=c(G=0,P=1,C=1,U=2) ,
                                   y=c(G=0,P=0,C=1,U=0.5) )
drawdag( parent_dag )
```



If conditioned on P, bias inference on $G \rightarrow$ C$, even without measureing U. Example:

```
N <- 200 # number of grandparent-parent-child triads
b_GP <- 1 # direct effect of G on P
b_GC <- 0 # direct effect of G on C
b_PC <- 1 # direct effect of P on C
b_U<-2 #direct effect of U on P and C

set.seed(1)
U <- 2*rbern( N , 0.5 ) - 1
G <- rnorm( N )
P <- rnorm( N , b_GP*G + b_U*U )
C <- rnorm( N , b_PC*P + b_GC*G + b_U*U )
d <- data.frame( C=C , P=P , G=G , U=U )
```

Set G effect as 0 for effect, U is binary. Try to measure grandparent influence now

```
m6.11 <- quap(
  alist(
    C ~ dnorm( mu , sigma ),
    mu <- a + b_PC*P + b_GC*G,
    a ~ dnorm( 0 , 1 ),
    c(b_PC,b_GC) ~ dnorm( 0 , 1 ),
    sigma ~ dexp( 1 )
  ), data=d )
precis(m6.11)
```

```
##             mean         sd       5.5%       94.5%
## a     -0.1174752 0.09919574 -0.2760091  0.04105877
## b_PC   1.7868915 0.04455355  1.7156863  1.85809664
## b_GC  -0.8389537 0.10614045 -1.0085867 -0.66932077
## sigma  1.4094891 0.07011139  1.2974375  1.52154063
```

Grandparents have huge negative effect, parents have twice as large as it should be. Regression not wrong but causal interpretation is. If you make a plot of C vs G, there's 2 sets, one for good neighborhoods $U = 1$ and one for bad $U = -1$ - but selecting those of similar those with similar education causes a negative trend. Knowing P, learning G invisibly tells us about the neighborhood - how to solve? Measure U.

```
m6.12 <- quap(
  alist(
    C ~ dnorm( mu , sigma ),
    mu <- a + b_PC*P + b_GC*G + b_U*U,
    a ~ dnorm( 0 , 1 ),
    c(b_PC,b_GC,b_U) ~ dnorm( 0 , 1 ),
    sigma ~ dexp( 1 )
  ), data=d )
precis(m6.12)
```

```
##              mean         sd       5.5%        94.5%
## a     -0.12197510 0.07192588 -0.2369265 -0.007023655
## b_PC   1.01161103 0.06597258  0.9061741  1.117047948
## b_GC  -0.04081373 0.09728716 -0.1962974  0.114669941
## b_U    1.99648992 0.14770462  1.7604294  2.232550439
## sigma  1.01959911 0.05080176  0.9384081  1.100790130
```

Which matches the simulated data. This effect is *Simpson's Paradox* - adding a predictor can reverse direction of association between another predictor and outcome.

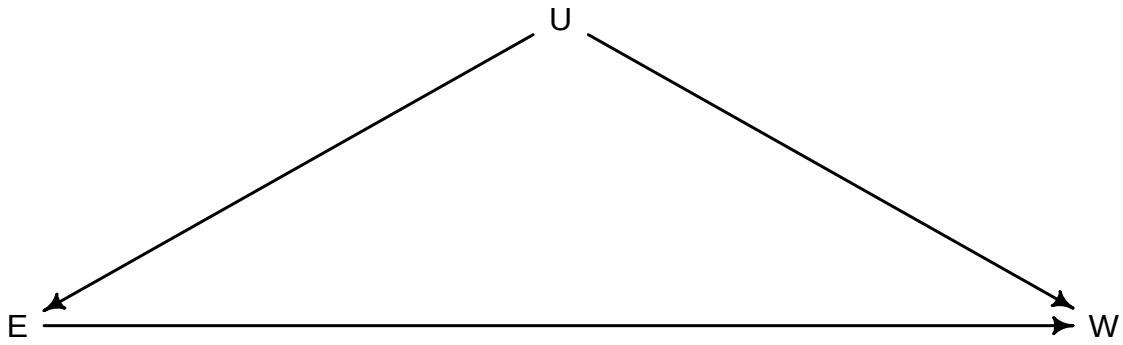
6.4 Confronting Confounding

Multiple regression can both help us deal with confounding, and cause it - controlling for the wrong variables ruins inference.

Confounding - an outcome Y and predictor X are not the same as if we experimentally determined X. Example, educations E, wages W. Many unobserved variables that affect both.

```
wage_dag <- dagitty( "dag {
  E <- U -> W
  E -> W
}")
coordinates( wage_dag ) <- list( x=c(E=0,U=1,W=2) ,
```

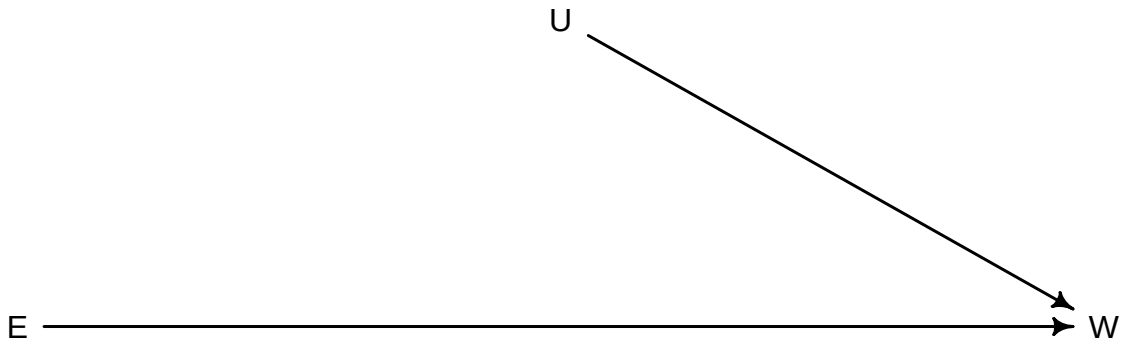
```
                                    y=c(E=1,U=0,W=1) )
drawdag( wage_dag )
```



Regress W on E, U is a confounder, since there are 2 paths connecting E and W. "Paths" in this sense ignore direction of arrows, but still create a statistical association, just not a *causal* one.

How to isolate the causal path? **Experiment** - assign education levels at random, removes the influence of U on E.

```
exp_dag <- dagitty( "dag {
  U -> W
  E -> W
}")
coordinates( exp_dag ) <- list( x=c(E=0,U=1,W=2) ,
                                    y=c(E=1,U=0,W=1) )
drawdag( exp_dag )
```



Other ways to do this though! Add U to the model, condition on it. This *blocks* the flow of information between E and W through U. Once you learn U, learning E provides no additional information about W - consider just $E \leftarrow U \rightarrow W$ in isolation.

(this is important and I want to better understand, so taking verbose notes on it)

Fleshing out - say U is average wealth in a region. High wealth U leads to more education E, leads to better jobs, leads to higher wages. Not knowing region, learning education E gives information about wages W, since these are correlated. But after learning region, assuming no other path between E and W, learning education tells you nothing about W - blocking the path. Becomes a fork.

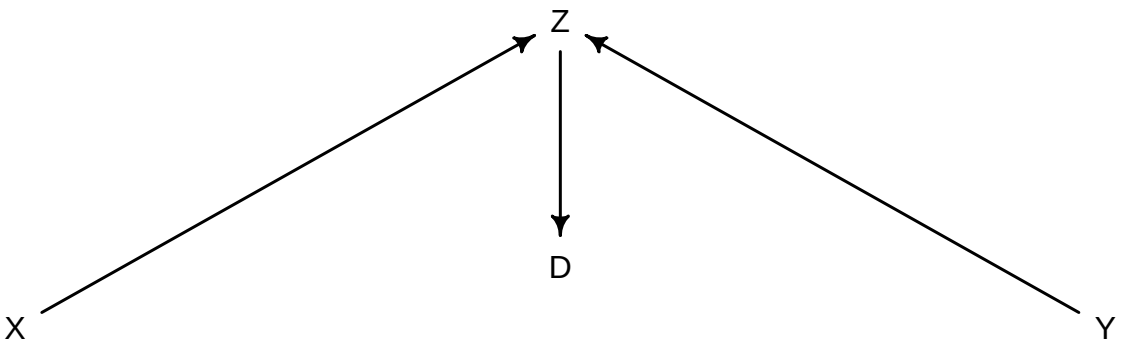
Shutting the backdoor

Blocking paths in this sense is called "shutting the backdoor." Don't want spurious correlation sneaking in through non-causal paths - $E \leftarrow U \rightarrow W$ is a backdoor since it enters E with an arrow and connects E to W.

Causal DAGs make it possible to say which we must control for to shut backdoor paths, also which *not* to. 4 possible variable relations.

1) $X \leftarrow Z \rightarrow Y$ known as the **Fork**. Z is a common cause of X and Y, causing correlation. Conditioning on Z, then learning X tells us nothing about Y. X and Y are independent conditional on Z
2) $X \rightarrow Z \rightarrow Y$ known as the **Pipe**. Seen in plant growth example - treatment$X$ influences fungus Z, which influences growth Y. Conditioning on Z blocks the $X \rightarrow Y$ path.
3) $X \rightarrow Z \leftarrow Y$ known as the **Collider**. Earlier in the chapter - no association between X and Y unless conditioning on Z, which would open the path.
4) The **Descendant**, shown in the DAG below. A variable influenced by another, conditioning on it is a weaker form of conditioning on the variable itself - in the DAG, conditioning on D will control Z, they share some information. This partially opens the path from X to Y since Z is a collider. In the pipe Z scenario, conditioning on the descendant is weakly closing the pipe.

```
descendent <- dagitty( "dag {
  X -> Z
  Y -> Z
  Z -> D
}")
coordinates( descendent ) <- list( x=c(X=0,Z=1,D=1,Y=2) ,
                                   y=c(X=1,Z=0,D=0.8,Y=1) )
drawdag( descendent )
```



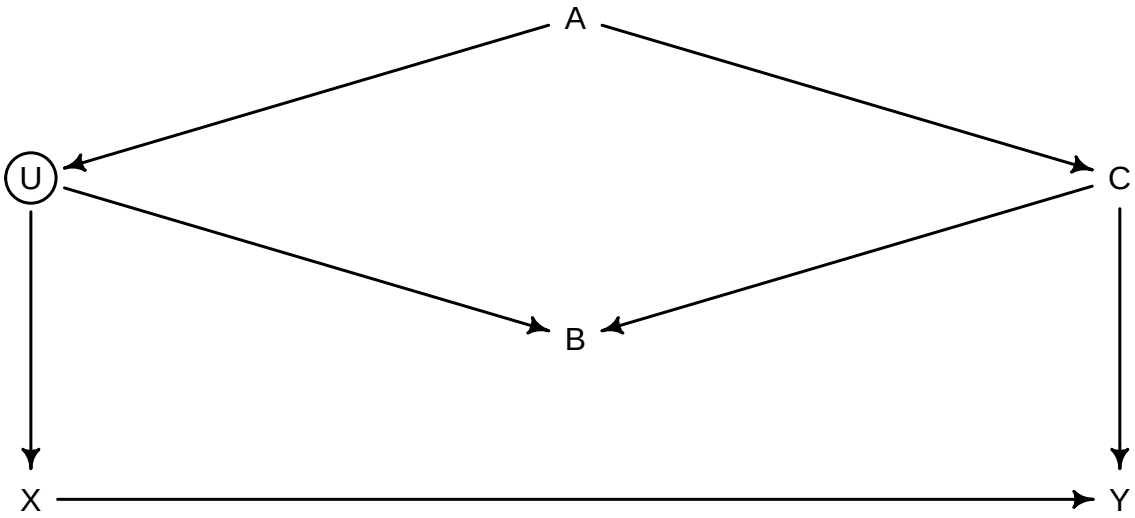
All DAGs are composed of these 4 relations. Here's the approach:

1. List all paths connecting cause X and outcome Y.
2. Classify each as open or closed - open unless it contains a collider.
3. Identify which have backdoor paths - those with an arrow entering X.
4. Decide which variables to condition on to close backdoors.

Examples to follow.

Two Roads

```
tworoads <- dagitty( "dag {
  U [unobserved]
  U <- A -> C
  Y <- X <- U  -> B <- C -> Y
}")
coordinates( tworoads ) <- list( x=c(U=0,X=0,A=1,B=1,C=2,Y=2) ,
                                 y=c(U=0.5,X=1.5,A=0,B=1,C=0.5,Y=1.5) )
drawdag( tworoads )
```



Exposure of interest X, outcome of interest Y, unobservable U, 3 covariates A, B, C.

Interested in bottom path from X to Y, but two other paths - one through U, A, C and one through U, B, C.

Consider A path, no colliders, so a backdoor - need to condition on it.

B path though has a collider, so need to not condition on it.

```
adjustmentSets( tworoads , exposure="X" , outcome="Y" )
```

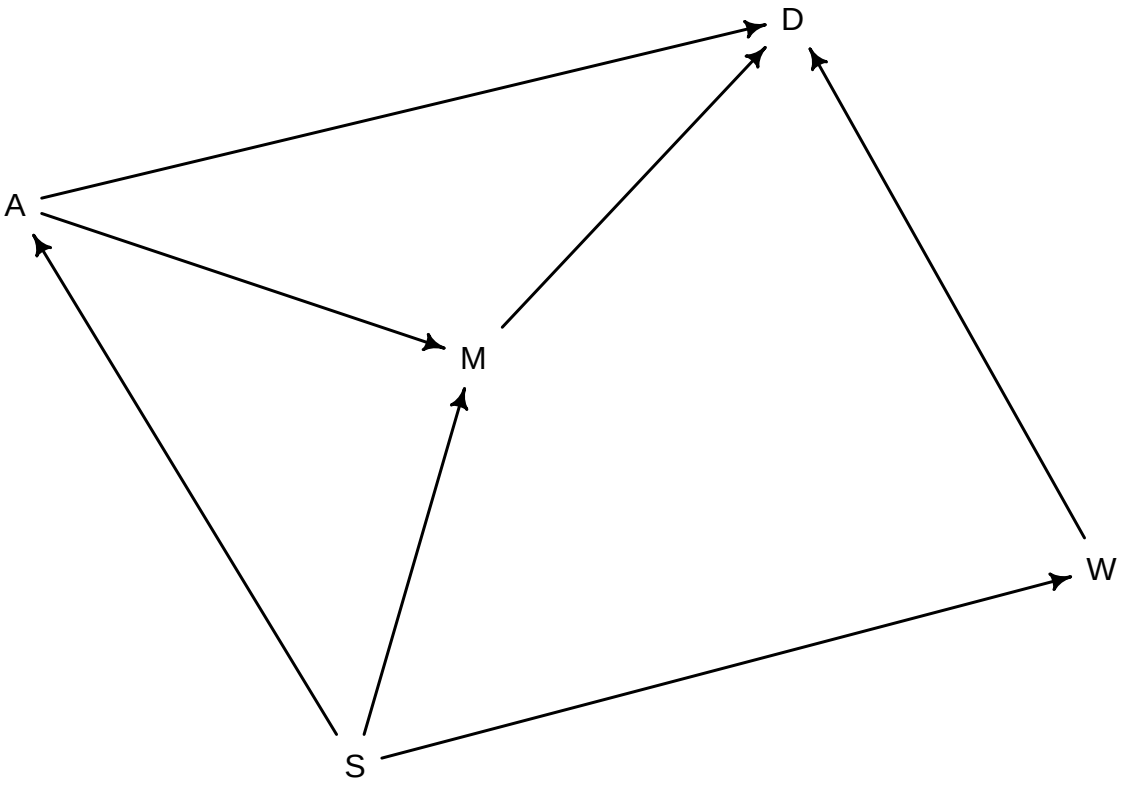
```
##  { C }
##  { A }
```

Daggity tool shows us options to condition on - in this case, C is a better option since it helps with precision of $X \to Y$. U would also work, but of course is unobserved.

Backdoor waffles

Going back to waffle house example. want to find minimal set of covariates, and derive testable implications. Data cannot tell us DAG is right, but can tell us when wrong.

```
waffle <- dagitty( "dag {
  A -> D
  A -> M -> D
  A <- S -> M
  S -> W -> D
}")
drawdag( waffle )
```



S is southern, M is marriage rate, W is waffle houses, D is divorce. Looking between W and D, 3 paths - SMD, SAD, SAMD.

```
adjustmentSets( waffle , exposure="W" , outcome="D" )
```

```
##  { A, M }
##  { S }
```

Can close all paths by conditioning on S alone. This DAG assumes no unobserved confounds, unlikely for this sort of data. Looking at the testable implications, or *conditional independencies*

```
impliedConditionalIndependencies(waffle)
```

```
## A _||_ W | S
## D _||_ S | A, M, W
## M _||_ W | S
```

- Median age of marriage is independent of waffle houses, conditioning on a state being in the south
- Divorce rate is independent of being in the south, conditioning on age, marriage rate, and number of waffle houses
- Marriage rate and waffle houses are independent, conditioning on being in the south

6.5 Summary

Multiple regression describes conditional associations, not causal influences. Common frustrations outlined - multicollinearity, post-treatment bias, collider bias. It is to reach causal valid causal inferences in the absence of experimentation.

Problem sets

Chapter 2 problems

2E1.

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

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

2E2.

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

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

2E3.

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

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

2E4.

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

Based on the tosses that we've performed, on the globe we hold, we can expect 70% of future tosses to also land on water. The source of the uncertainty and limited knowledge is that the point we're landing on is random.

2M1.

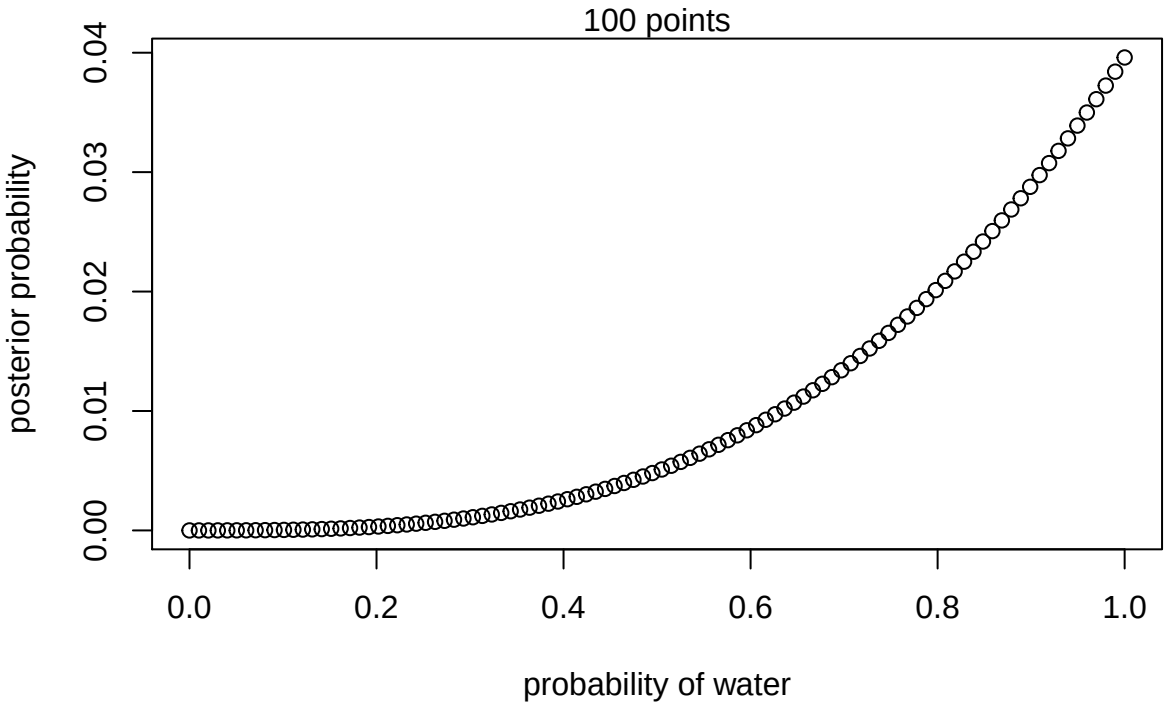
Recall the globe tossing model from the chapter. Compute and plot the grid approximate posterior distribution for each of the following sets of observations. In each case, assume a uniform prior for p.

```
globe_grid <- function(trial_list, grid_size){
  # define grid
  p_grid <- seq( from=0 , to=1 , length.out=grid_size )
  # define prior
  prior <- rep( 1 , grid_size )
  # compute likelihood at each value in grid
  likelihood <- dbinom( sum(trial_list) , size=length(trial_list) , prob=p_grid )
  # compute product of likelihood and prior
```

```
  unstd.posterior <- likelihood * prior
  # standardize the posterior, so it sums to 1
  posterior <- unstd.posterior / sum(unstd.posterior)
  plot(p_grid , posterior , type="b" ,
       xlab="probability of water" , ylab="posterior probability")
  mtext( sprintf("%i points", grid_size ))
}
```

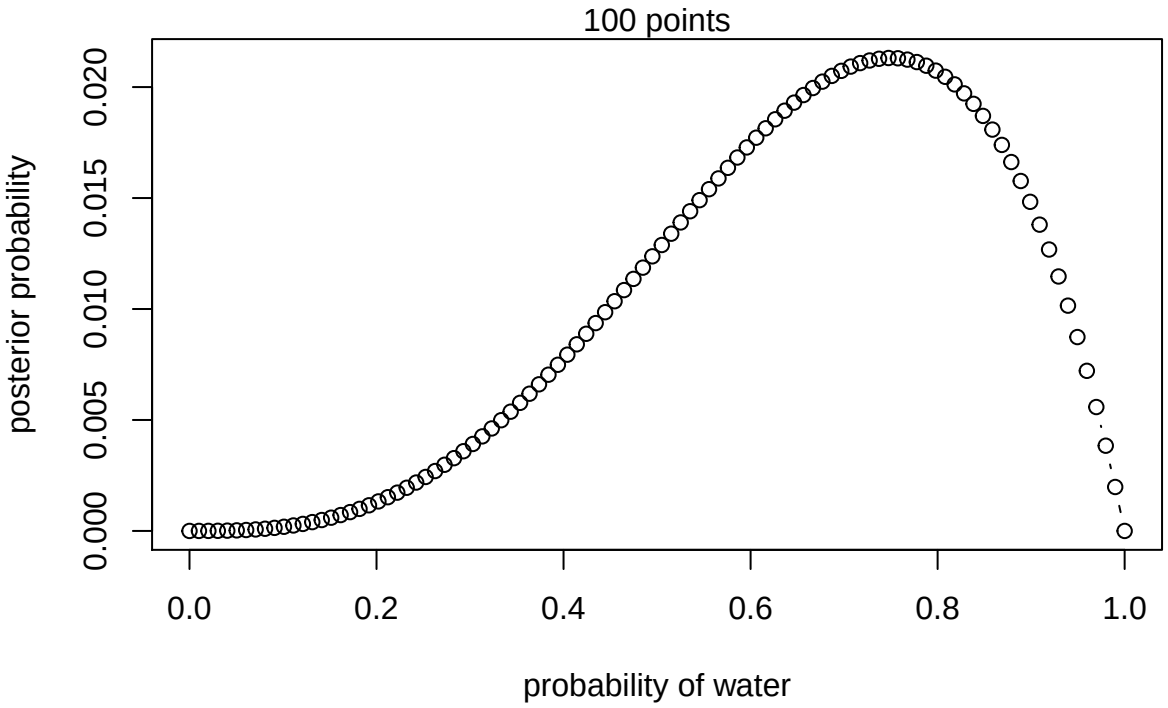
1. W, W, W

```
globe_grid(c(1,1,1),100)
```



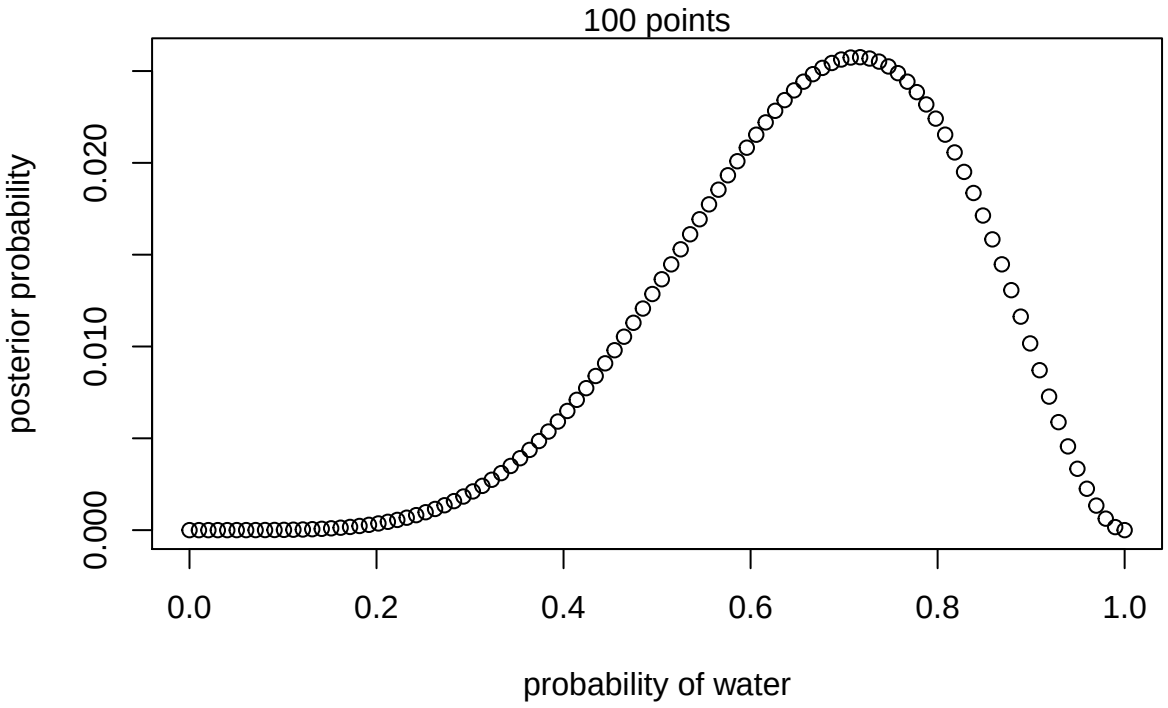
2. W, W, W, L

```
globe_grid(c(1,1,1,0),100)
```



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

```
globe_grid(c(0,1,1,0,1,1,1),100)
```



2M2.

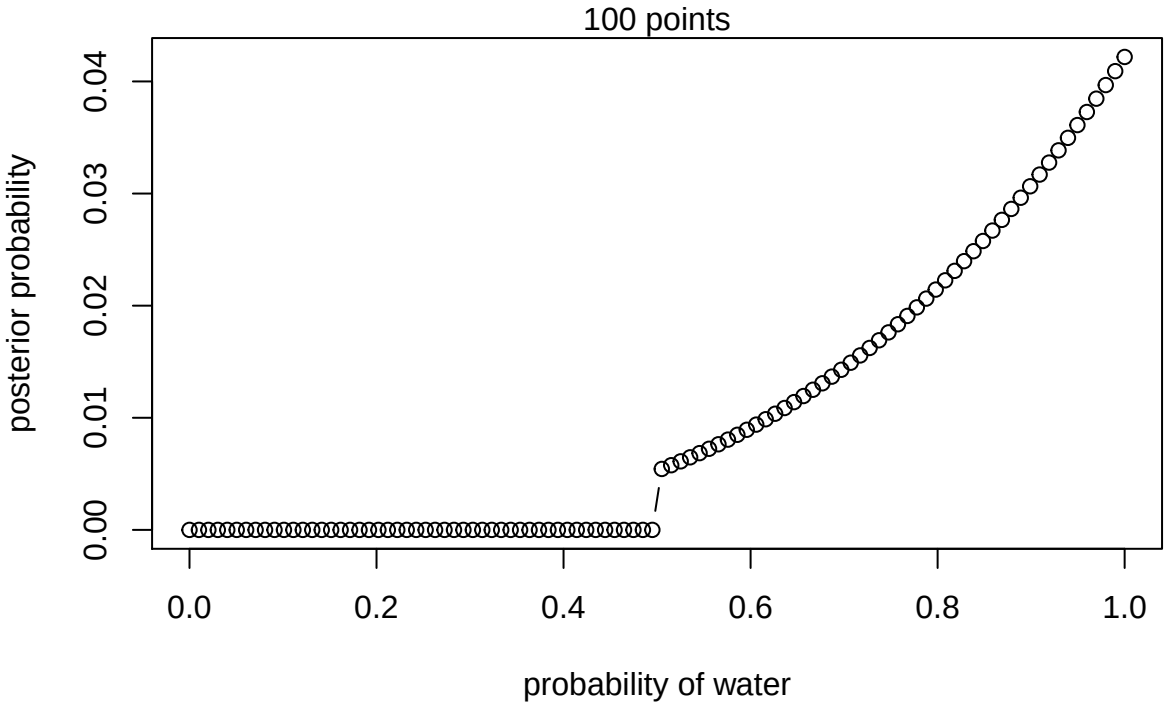
> Now assume a prior for p that is equal to zero when $p < 0.5$ and is a positive constant when $p \geq 0.5$. Again compute and plot the grid approximate posterior distribution for each of the sets of observations in the problem just above.

Change function prior definition to:

```
# define prior
prior <- (p_grid >= 0.5) * prior_const
```

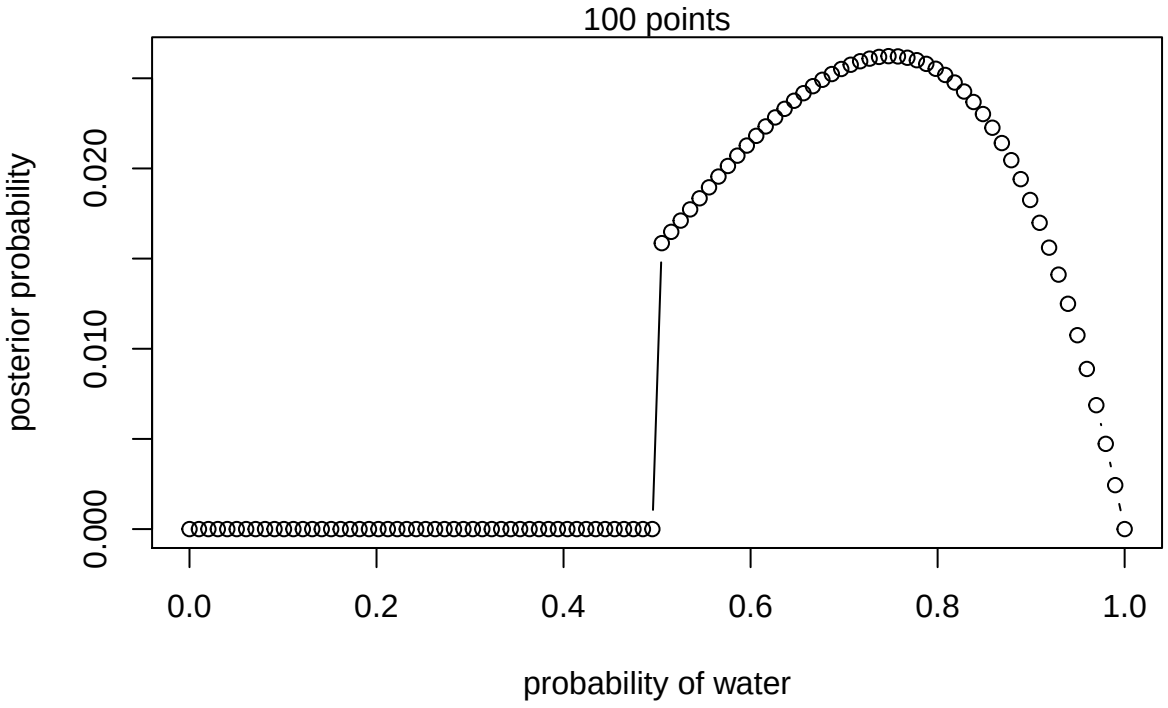
1. W, W, W

```
globe_grid_const(c(1,1,1),100, 1.0)
```



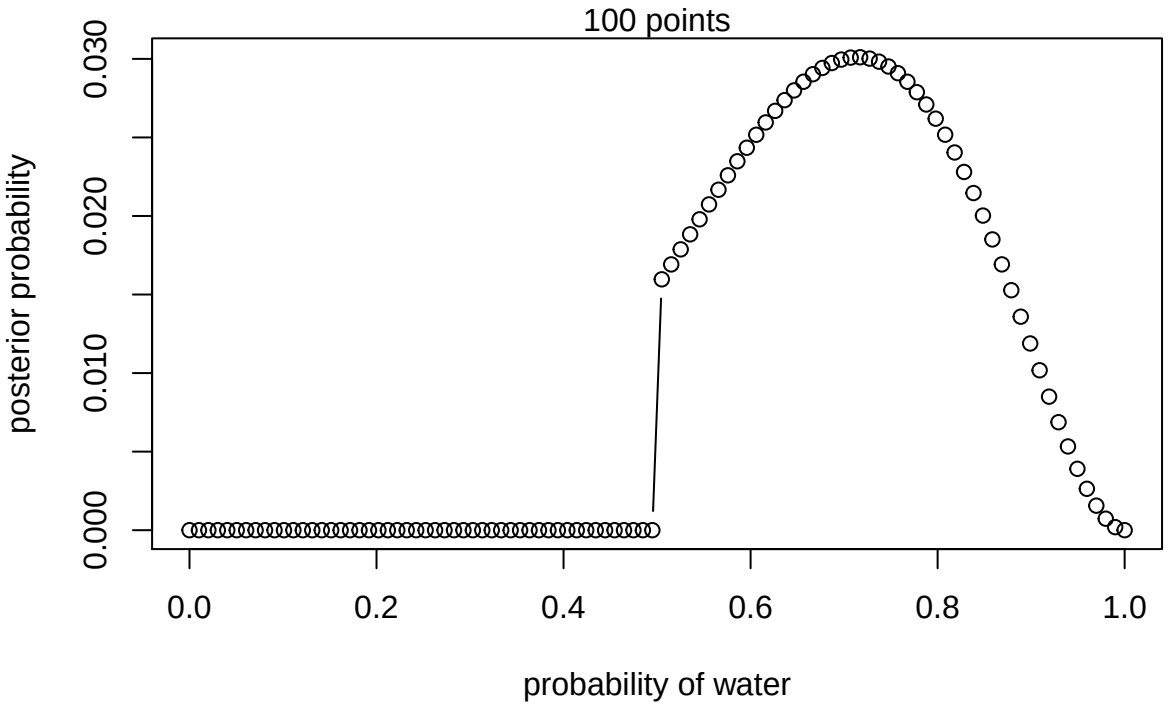
2. W, W, W, L

```
globe_grid_const(c(1,1,1,0),100, 1.0)
```



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

```
globe_grid_const(c(0,1,1,0,1,1,1),100,1)
```



2M3.

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

First find total probability of land:

$$\Pr(\text{land}) = \Pr(\text{land}|\text{Earth})\Pr(\text{Earth}) + \Pr(\text{land}|\text{Mars})\Pr(\text{Mars}) = (0.3)(0.5) + (1.0)(0.5) = 0.65$$

Now solve for probability of Earth, given we have land:

$$\Pr(\text{Earth}|\text{land}) = \frac{\Pr(\text{land}|\text{Earth})\Pr(\text{Earth})}{\Pr(\text{land})} = \frac{(0.3)(0.5)}{0.65} \approx 0.23$$

2M4.

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

Enumerating all possible scenarios (b=black, w=white):

1. w up; w down (w/w)

2. w up; w down (w/w)
3. w up; b down (w/b)
4. b up; w down (w/b)
5. b up; b down (b/b)
6. b up; b down (b/b)

Observation eliminates 1-3, so 3-6 remain. 2/3 of those are b/b card, so this is our solution.

2M5.

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

Add to the prior cases:

7. b up; b down (new b/b)
8. b up; b down (new b/b)

Again we can eliminate 1-3 from observation. This leaves 5 cases (3-8). Of those 4 are b/b, so 4/5.

2M6.

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

Same cases as before, but now amend rates

up	down	card	rate
w	w	w/w	3
w	w	w/w	3
w	b	w/b	2
b	w	w/b	2
b	b	b/b	1
b	b	b/b	1

Now, we can cancel the first three cases, since we've pulled black. 2 b/b options at rate 1, 1 w/b option at rate 2. That means 2 positive chances out of 4 total rate chances, $2/4 = 0.5$.

2M7.

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

Looking at scenarios that match data:

1. b/b, w/b
2. b/b (flipped), w/b
3. b/b, w/w
4. b/b, w/w (flipped)

5. b/b (flipped), w/w
6. b/b (flipped), w/w (flipped)
7. b/w, w/w
8. b/w, w/w (flipped)

1-6 are desired, 7-8 are not; therefore 6/8 or 75%.

2H1.

> Suppose there are two species of panda bear. Both are equally common in the wild and live in the same places. They look exactly alike and eat the same food, and there is yet no genetic assay capable of telling them apart. They differ however in their family sizes. Species A gives birth to twins 10% of the time, otherwise birthing a single infant. Species B births twins 20% of the time, otherwise birthing singleton infants. Assume these numbers are known with certainty, from many years of field research. Now suppose you are managing a captive panda breeding program. You have a new female panda of unknown species, and she has just given birth to twins. What is the probability that her next birth will also be twins?

```
# Givens
rate_a <- .5
rate_b <- .5
twin_rate_a <- .1
twin_rate_b <- .2
# Need to solve:
# P(twins) = P(twins|A)*P(A) + P(twins|B)P(B)
sum_probability_twins <- rate_a * twin_rate_a + rate_b * twin_rate_b # norm factor

pA_given_twins <- (twin_rate_a * rate_a) / sum_probability_twins
pB_given_twins <- (twin_rate_b * rate_b) / sum_probability_twins
p_twins <- twin_rate_a * pA_given_twins + twin_rate_b * pB_given_twins
p_twins
```

```
## [1] 0.1666667
```

16.7% chance

2H2.

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

```
pA_given_twins
```

```
## [1] 0.3333333
```

33% chance

2H3.

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

```
# Givens
updated_rate_a <- pA_given_twins
updated_rate_b <- pB_given_twins
single_rate_a <- 1-twin_rate_a
single_rate_b <- 1-twin_rate_b

# Repeat calculations for new single birth
# norm factor
```

```
sum_probability_single <- single_rate_a * twin_rate_a +
  single_rate_b * twin_rate_b
# Calculate probabilities
pA_given_single <- (single_rate_a * twin_rate_a) / sum_probability_single
pB_given_single <- (single_rate_b * twin_rate_b) / sum_probability_single
pA_given_single
```

```
## [1] 0.36
```

36% chance

2H4.

> A common boast of Bayesian statisticians is that Bayesian inference makes it easy to use all of the data, even if the data are of different types. So suppose now that a veterinarian comes along who has a new genetic test that she claims can identify the species of our mother panda. But the test, like all tests, is imperfect. This is the information you have about the test: - The probability it correctly identifies a species A panda is 0.8. - The probability it correctly identifies a species B panda is 0.65. The vet administers the test to your panda and tells you that the test is positive for species A. First ignore your previous information from the births and compute the posterior probability that your panda is species A. Then redo your calculation, now using the birth data as well.

Starting with data-free solution:

```
# Givens
a_given_pos <- 0.8
b_given_pos <- 1 - a_given_pos
b_given_neg <- 0.65
b_given_pos <- 1 - b_given_neg

pA_given_test <- a_given_pos * rate_a /
                 (a_given_pos * rate_a + b_given_pos * rate_b)
pA_given_test
```

```
## [1] 0.6956522
```

Probability of A given test returns an "A" reading is 69.6%

Now adding the data:

```
# P(A| positive test, twins, single) =
# P(positive test|A) * P(twins|A) * P(single|A) * P(A) /
# P(postive test, twins, single)
numerator <- a_given_pos * twin_rate_a * single_rate_a * rate_a

# P(positive test, twins, single) =
# P(positive test|A) * P(twins|A) *  P(single|A) * P(A) +
# (positive test|B) * P(twins|B) *  P(single|B) * P(B)
denom <- numerator + b_given_pos * twin_rate_b * single_rate_b * rate_b

numerator/denom
```

```
## [1] 0.5625
```

56.25% chance

Chapter 3 problems

Given:

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

3E1.

How much posterior probability lies below p = 0.2?

```
sum(samples < 0.2) / trial_size
```

```
## [1] 4e-04
```

3E2.

How much posterior probability lies above p = 0.8?

```
sum(samples > 0.8) / trial_size
```

```
## [1] 0.1116
```

3E3.

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

```
sum(samples < 0.8 & samples > 0.2) / trial_size
```

```
## [1] 0.888
```

3E4.

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

```
quantile(samples, 0.2)
```

```
##       20%
## 0.5185185
```

3E5.

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

```
quantile(samples, 1-0.2)
```

```
##       80%
## 0.7557558
```

3E6.

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

```
HPDI(samples,prob=.66)
```

```
##     |0.66     0.66|
## 0.5085085 0.7737738
```

3E7.

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

```
PI(samples,prob=.66)
```

```
##       17%       83%
## 0.5025025 0.7697698
```

3M1.

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

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

3M2.

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

```
samples <- sample(p_grid, prob=posterior, size=1e5, replace=TRUE)
HPDI(samples, prob=0.9)
```

```
##      |0.9      0.9|
## 0.3413413 0.7267267
```

3M3. >Construct a posterior predictive check for this model and data. This means simulate the distribution of samples, averaging over the posterior uncertainty in p. What is the probability of observing 8 water in 15 tosses?

```
simulations <- 1e4
w <- rbinom( simulations, size=15 , prob=samples )
sum(w==8)/simulations
```

```
## [1] 0.1473
```

3M4.

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

```
simulations <- 1e4
w_2 <- rbinom( simulations, size=9 , prob=samples )
sum(w==6)/simulations
```

```
## [1] 0.1108
```

3M5.

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

```
p_grid <- seq( from=0 , to=1 , length.out=1000 )
prior_const <- 1
prior <- (p_grid >= 0.5) * prior_const
prob_data_new <- dbinom( 8 , size=15 , prob=p_grid )
```

```
posterior_new <- prob_data_new * prior
posterior_new <- posterior_new / sum(posterior)
samples_new <- sample(p_grid, prob=posterior_new, size=1e5, replace=TRUE)
```

Tackling all the old problems:

```
print("Problem 2")
```

```
## [1] "Problem 2"
```

```
HPDI(samples_new, prob=0.9)
```

```
##      |0.9      0.9|
## 0.5005005 0.7117117
```

```
print("Problem 3")
```

```
## [1] "Problem 3"
```

```
simulations <- 1e4
w <- rbinom( simulations, size=15 , prob=samples_new )
sum(w==8)/simulations
```

```
## [1] 0.1634
```

```
print("Problem 4")
```

```
## [1] "Problem 4"
```

```
w_2 <- rbinom( simulations, size=9 , prob=samples_new )
sum(w==6)/simulations
```

```
## [1] 0.0659
```

HPDI is far narrower. Likelihood of 8/15 is slightly increased, likelihood of 6/9 increases considerably - effectively we've removed the opportunity for fewer than 50% water cases to be considered, which will subsequently increase the likelihood of all >50% cases.

3M6.

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

```
interval_width <- 1
nSimulations <- 0
p <- 0.7
while (interval_width > 0.05)
{
  nSimulations <- nSimulations + 10
  p_grid <- seq( from=0 , to=1 , length.out=1000 )
  prob_p <- rep( 1 , 1000 )

  # Simulate data
  simulations <- nSimulations
  likelihood <- dbinom( round(simulations*p), size=simulations, prob=p_grid )
  posterior <- likelihood * prob_p
  posterior <- posterior / sum(posterior)
  #print(posterior)
```

```
  trial_size <- 1e4
  #print(trial_size)
  samples <- sample( p_grid , prob=posterior , size=trial_size , replace=TRUE )
  interval_width <- quantile(samples,0.995) - quantile(samples, 0.005)
}
nSimulations
```

```
## [1] 2200
```

About 2200 trials.

3H1.

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

```
all_births <- c(birth1,birth2)
p_grid <- seq( from=0 , to=1 , length.out=1000 )
prior <- rep( 1 , 1000 ) #Uniform
likelihood <- dbinom( sum(all_births) , size=length(all_births) , prob=p_grid )
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)
p_grid[which.max(posterior)]
```

```
## [1] 0.5545546
```

3H2.

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

```
trial_size <- 10000
samples <- sample( p_grid , prob=posterior , size=trial_size , replace=TRUE )
HPDI(samples, prob=.5)
```

```
##      |0.5      0.5|
## 0.5305305 0.5775776
```

```
HPDI(samples, prob=.89)
```

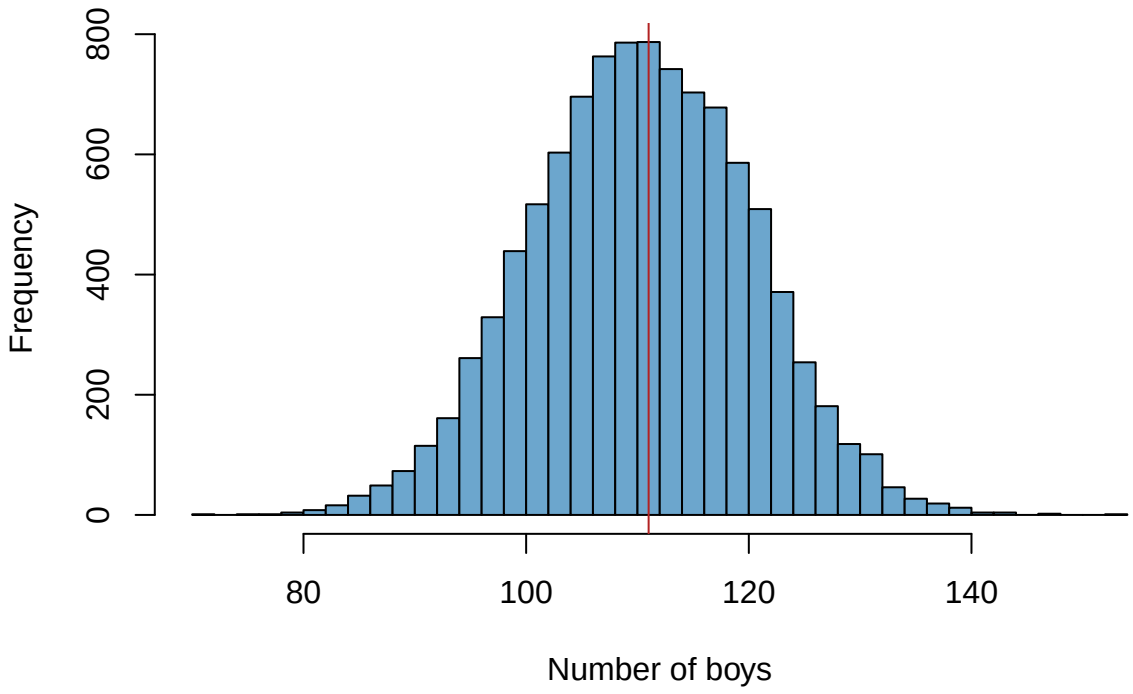
```
##     |0.89     0.89|
## 0.5005005 0.6116116
```

```
HPDI(samples, prob=.97)
```

```
##     |0.97     0.97|
## 0.4794795 0.6296296
```

3H3. Use rbinom to simulate 10,000 replicates of 200 births. You should end up with 10,000 numbers, each one a count of boys out of 200 births. Compare the distribution of predicted numbers of boys to the actual count in the data (111 boys out of 200 births). There are many good ways to visualize the simulations, but the dens command (part of the rethinking package) is probably the easiest way in this case. Does it look like the model fits the data well? That is, does the distribution of predictions include the actual observation as a central, likely outcome?

```
sim <- rbinom(10000, size=200, prob=samples)
hist(sim, c="skyblue3", breaks=50, xlab="Number of boys", main="")
abline(v=sum(all_births), col="firebrick")
```

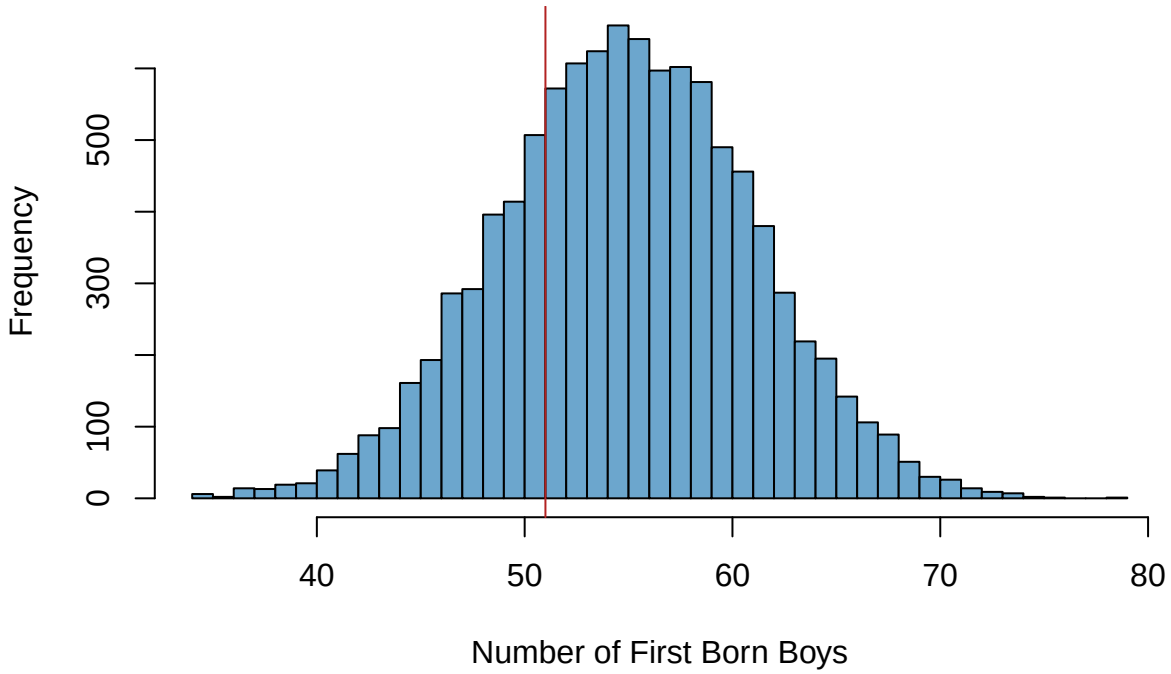


Fits right on mean, this seems like a likely outcome.

3H4.

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

```
sim <- rbinom(10000, size=100, prob=samples)
hist(sim, c="skyblue3", breaks=50, xlab="Number of First Born Boys", main="")
abline(v=sum(birth1), col="firebrick")
```



It's not on the maximum likelihood location, but it's still a reasonable value.

```
sprintf("Value: %i",sum(birth1))
```

```
## [1] "Value: 51"
```

```
PI(sim,prob=0.60)
```

```
## 20% 80%
##  50  61
```

The value is within the inner 60% of posterior density

3H5.

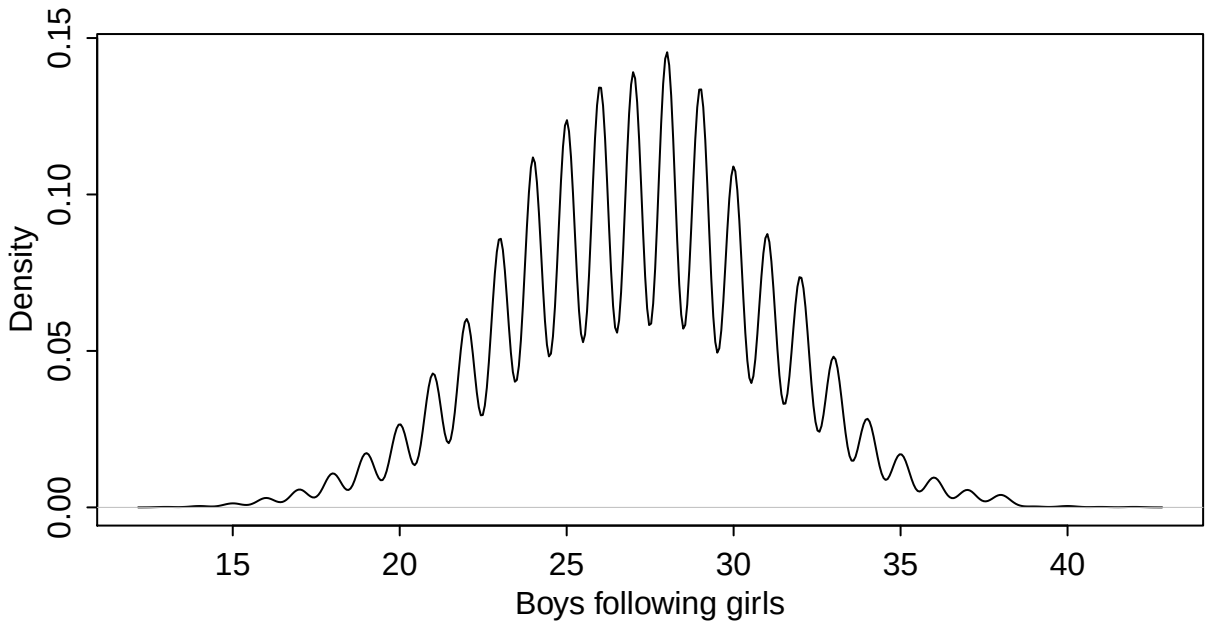
The model assumes that sex of first and second births are independent. To check this assumption, focus now on second births that followed female first borns. Compare 10,000 simulated counts of boys to only those second births that followed girls. To do this correctly, you need to count the number of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

```
boys_after_girls <- birth2[birth1==0 & birth2 ==1]
sum(boys_after_girls)
```

```
## [1] 39
```

39 cases of boys born after a girl

```
count_first_girls <- sum(birth1==0)
sim_girl <- rbinom(10000, size=count_first_girls, prob=samples)
dens(sim_girl, xlab="Boys following girls", main="")
```



This doesn't look like anything normal. The biggest thing is that binomial assumes that trials are independent, and it's very possible these are not.

Chapter 4 problems

4E1.

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

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu, \sigma) \\
\mu &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

Line 1, the y_i definition, is the likelihood.

4E2.

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

2 parameters (μ, σ)

4E3.

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

TODO

4E4.

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

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim \text{Normal}(0, 10) \\
\beta &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Exponential}(2)
\end{aligned}
$$

The second line, μ_i definition, is the linear model.

4E5.

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

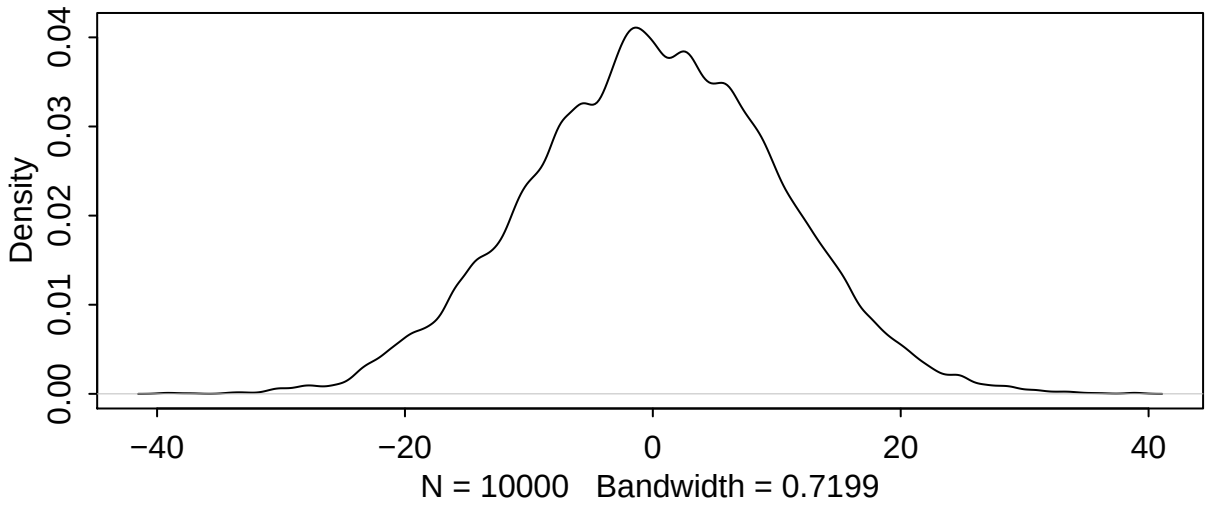
3 parameters (α, β, σ)

4M1.

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

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu, \sigma) \\
\mu &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

```
mu <- rnorm(10000, 0, 10) # normal from 0 to 10
sigma <- rexp(10000, 1) # Exponential rate 1
y_prior <- rnorm(10000, mu, sigma)
dens(y_prior)
```



4M2.

Translate the model just above into a quap formula.

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

4M3.

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

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

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu, \sigma) \\
\mu_i &= a + bx_i \\
a &\sim \text{Normal}(0, 10) \\
b &\sim \text{Uniform}(0, 1) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

4M4.

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

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim \text{Normal}(152, 6) \\
\beta &\sim \text{Uniform}(8, 4) \\
\sigma &\sim \text{Uniform}(0, 30)
\end{aligned}
$$

Defense:

For simplicity, heights given as normal (typically might expect bimodal for male/female). Parameters are μ and σ.

Use a linear regression to get μ, where x_i is year. σ, the deviation, we'll set to uniform. I'd expect about 95% (2σ) to fall within 1.3 ft (e.g. if mean is 5.5 ft, 95% will be between 6.15 ft and 4.85 ft), so that puts σ at 0.65 ft $\sim$ 20 cm, so to be conservative, we'll do a uniform over 0-30 cm.

Set regression α to an average height; not indicating what age "students" are (could be elementary, could be college), so let's say 5 ft ($\sim$ 152 cm), with a 0.65 ft ($\sim$ 20 cm) deviation. Then β is how much growth happens per year. Let's say a high schooler is full grown at 18, at 5.5 ft on average, born just under a foot, so over 18 years grow 4.5 foot, so that's about 3 inches a year, or just under 8 cm, and give that a reasonable distribution of half that.

4M5.

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

The problem suggests that I might want to update β, but we've already considered that students are growing each year, so probably don't need to update the prior. We want to ensure that β is never negative, so if anything we might reduce the standard deviation to remove the possibility of negative values there at the fringe of the tails.

4M6.

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

Variance is σ^2, so that makes the standard deviation for β at most $\sqrt{64} = 8$. I set this value to 4 already, implying my prior is a stronger constraint than the new information given. This might lead me to loosen that constraint a little bit, for safety.

4H1.

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

```
# Load data
data("Howell1")
d <- Howell1

height_model <- map(
  alist(
    height ~ dnorm(a + b*weight, sigma),
    a ~ dnorm(178, 20),
    b ~ dnorm(0, 10),
    sigma ~ dunif(0, 50) # most pirors stolen from chapter
  ),
```

```
  data = d
)
to_evaluate <- c(46.95, 43.72, 64.78, 32.59, 54.63)
simulated_heights <- sim(height_model, data=list(weight=to_evaluate), n=1e5, silent=TRUE)
mean_val <- apply(simulated_heights, # on heights
                  2, # Columns (not rows)
                  mean # Evaluate mean
                  )
print(mean_val)
```

```
## [1] 158.2306 152.5320 189.6199 132.9842 171.8118
```

```
percentile_interval <- apply(simulated_heights,
                             2,
                             PI)
print(percentile_interval)
```

```
##         [,1]     [,2]     [,3]     [,4]     [,5]
## 5%  143.2583 137.5613 174.5426 118.0317 156.8543
## 94% 173.2253 167.4803 204.6575 147.8720 186.7142
```

4H2.

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

```
d_youth <- d[d$age < 18,]
nrow(d_youth)
```

```
## [1] 192
```

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

```
youth_height_model <- quap(
  alist(
    height ~ dnorm(a + b*weight, sigma),
    a ~ dnorm(178, 100),
    b ~ dnorm(0, 10),
    sigma ~ dunif(0, 50)
  ),
  data = d_youth
)

print(youth_height_model)
```

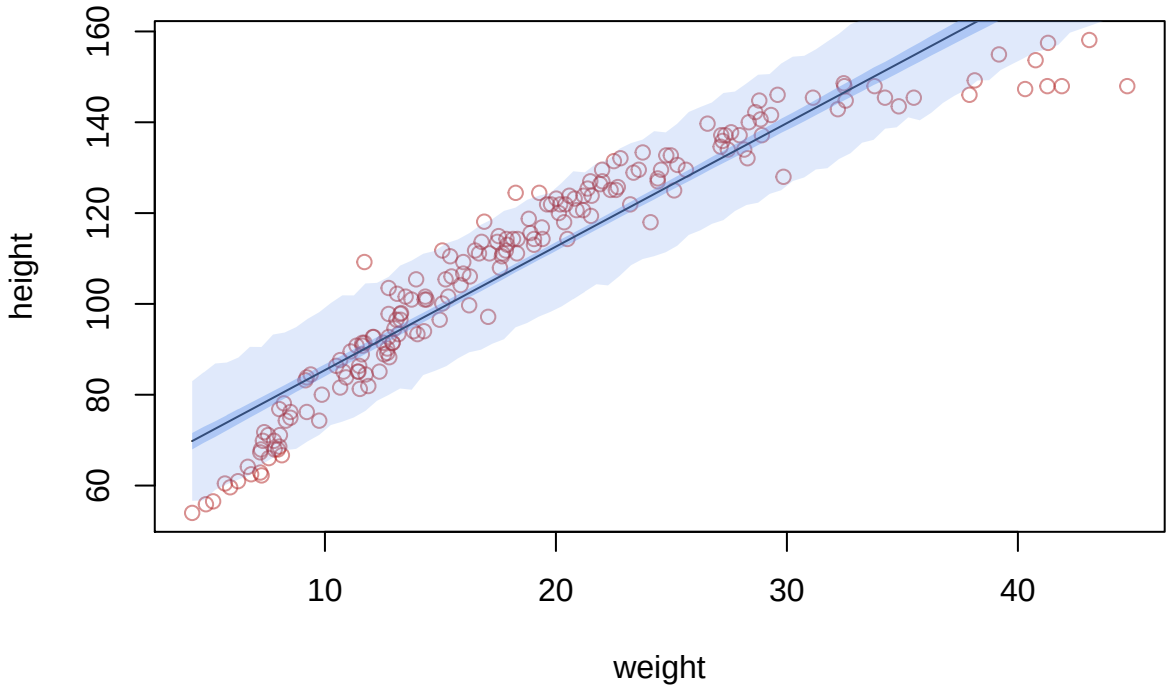
```
##
## Quadratic approximate posterior distribution
##
## Formula:
## height ~ dnorm(a + b * weight, sigma)
## a ~ dnorm(178, 100)
## b ~ dnorm(0, 10)
## sigma ~ dunif(0, 50)
##
## Posterior means:
##         a         b     sigma
```

```
## 58.255103  2.718987  8.437192
##
## Log-likelihood: -681.9
```

For every 10 units of increase in weight, height increases by about 27 cm.

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

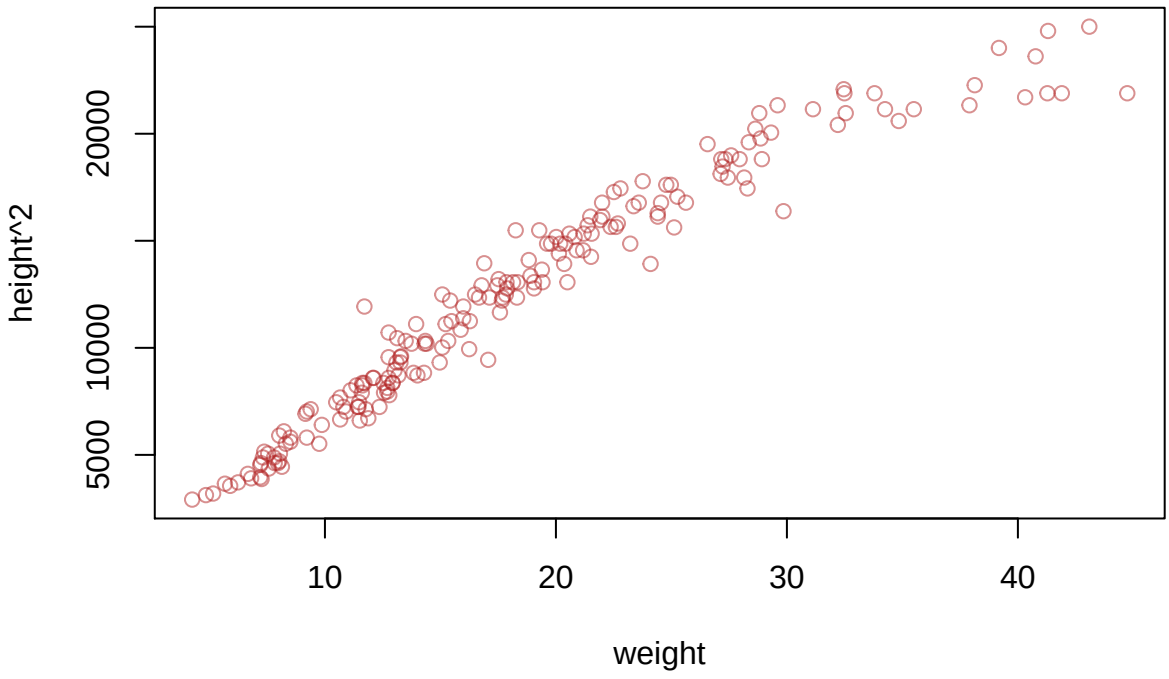
```
plot( height ~ weight , data=d_youth , col=col.alpha("firebrick",0.5) )
x_seq <- seq(from=min(d_youth$weight), to=max(d_youth$weight), by=0.5)

# Sample posterior

# Inital try, didn't work
# mu <- link(youth_height_model, data=data.frame(weight=x_seq))

posterior <- extract.samples(youth_height_model)
mu_link_lambda <- function(weight){
  posterior$a + posterior$b*weight
}
mu <- sapply(x_seq, mu_link_lambda)

mu.mean <- apply(mu, 2, mean )
mu.PI <- apply(mu, 2, PI)
mu.HPDI <- apply(mu, 2, HPDI)

simulated_heights <- sim(youth_height_model,data=list(weight=x_seq))

height.PI <- apply(simulated_heights, 2, PI)

# plot the MAP line, aka the mean mu for each weight
lines( x_seq , mu.mean )
# plot a shaded region for 89% PI
shade(mu.HPDI,x_seq, col=col.alpha('cornflowerblue',0.4))
shade( height.PI , x_seq,col=col.alpha('cornflowerblue',0.2))
```



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

This data doesn't appear to be linear at the tails. Probably what the best method would be is to square the height values then draw a linear model on that:

```
plot( height**2 ~ weight , data=d_youth , col=col.alpha("firebrick",0.5) )
```



Certainly looks more linear than before. Alternatively, a polynomial function would work too.