Statistical Rethinking

Notes working through the book

Daniela Palleschi

2023-09-25

Table of contents

Preface

This is a Quarto book containing my notes as I work through McElreath (2020), using also the accompanying YouTube lecture videos (2023), and Kurz (2023). I'm working through the book as part of a bi-weekly meeting with members of the ZAS Berlin, with shared materials available on GitHub.

Part I

Overview

Set-up

Packages

To be run prior to the other scripts.

```
install.packages("pacman")
pacman::p_load(devtools)

# install developer packages
remotes::install_github("bnicenboim/bcogsci")
remotes::install_github("stan-dev/cmdstanr")
install.packages(c("coda","mvtnorm","devtools","loo","dagitty","shape"))
devtools::install_github("rmcelreath/rethinking")
```

```
pacman::p_load(
MASS,
## be careful to load dplyr after MASS
dplyr,
tidyr,
purrr,
extraDistr,
ggplot2,
loo,
bridgesampling,
brms,
bayesplot,
tictoc,
hypr,
bcogsci,
lme4,
rethinking,
rstan,
# This package is optional, see https://mc-stan.org/cmdstanr/:
cmdstanr,
```

```
# This package is optional, see https://hyunjimoon.github.io/SBC/:
SBC,
SHELF,
rootSolve
)

# could be added to the top of scripts:
## Save compiled models:
rstan_options(auto_write = FALSE)
## Parallelize the chains using all the cores:
options(mc.cores = parallel::detectCores())
# To solve some conflicts between packages:
select <- dplyr::select
extract <- rstan::extract
```

Writing tips

To render documents with equations in HTML *and* PDF, specifically with alignment and numbering of multiple equations in one chunk, avoid using `$$` *and* `\begin{align}`. I like `$$` because it gives you a preview of your equation, but it crashes when rendering a PDF with `\begin{align}`, because pandoc reads this as doubly opening an equation/mathmode (as far as I understand it). So, the solution is to just use `\begin{align}` when I want to align more than one equation.

The following will not allow PDF to render:

```
# will render in HTML but not PDF:
$$
\begin{align}
y &\sim Normal(\mu,\sigma) \label{eq-normal} \\
y &\sim binomial(N = 1, p) \label{eq-binomial}
\end{align}
$$
```

The following will allow PDF to render, and will be printed as in **?@eq-normal1** and **?@eq-binomial2**

```
# will render in both HTML and PDF:
\begin{align}
y &\sim Normal(\mu,\sigma) \label{eq-normal} \\
```

```
y &\sim binomial(N = 1, p) \label{eq-binomial}
\end{align}
```

$$y \sim Normal(\mu, \sigma) \tag{0.1}$$
$$y \sim binomial(N = 1, p) \tag{0.2}$$

It's still helpful to use `$$` when writing so you can check that your equation will be printed as you intend. Just remember to remove them/comment them out before rendering.

Part II

Chapter notes

1 Small Worlds and Large Worlds

Chapter 2

```
# install if needed
# install.packages("pacman")
# devtools::install_github("rmcelreath/rethinking")
```

1.1 Terms and Concepts

term	definition
conjecture	possible outcomes (like the 5 possible proportions of blue/white marbles in the ba
plausibility	things that can happen more than one way are plausible; we want to find out whi
likelihood (general)	relative number of ways that a value p can produce the data; derived by enumerat
likelihood (Bayesian)	distributions of variables; the observed data
prior probability	prior plausibility of any specific p
prior	distribution of prior plausibility
posterior probability	new, updated plausibility of any specific p given priors + data
Bayesian updating	updating prior plausibilties in light of the data to produce posterior plausibilties
variables	symbols that can tkae on a different value; e.g., the coutns of water and land in th
parameters	unobserved variables; e.g., the proportion of water on the globe (in the land/wate
maximal entropy	the distrubtion contains no additional information other then: tehre are 2 events,
Bayes' theorem	$$Pr(p\|W,L) = \frac{Prob\ of\ data\ x\ Prior}{Average\ Prob\ of\ data}$$
Grid approximation	compute the posterior probability for any and every particular value of a paramte
quadratic approximation	a Guassian approximation is quadratic approximation because the log of the Guas

1.2 Code from chapter

1.2.1 Plausibilties

How probable is each possible combination of white/blue marbles? (divide each # of ways each combination could produce the observed data by sum of all ways. Ways: all white = 0 ways, 1 blue = 3, 2 blue = 8, etc.)

```
# R code 2.1: compute plausibilities
ways <- c(0,3,8,9,0)
ways/sum(ways)
```

```
[1] 0.00 0.15 0.40 0.45 0.00
```

So, given that we have taken 2 blue and 1 white out of the bag is 0, the plausibility of:

- all marbles being white (0 ways) = 0
- 1 blue, 3 white (3 ways) = 0.5
- 2 blue, 2 white (8 ways) = 0.4
- 3 blue, 1 white (9 ways) = .5
- 4 blue, 0 white = 0

1.2.2 Binomial distribution (`dbinom`)

In the water/land world-tossing example, what is the lieklihood of the data (where we observed 6 water and 3 land) if we assume the probability of observing 'water' is 0.5?

```
# R code 2.2: binomial distribution
dbinom(6, size = 9, prob = .5)
```

```
[1] 0.1640625
```

1.2.3 Grid approximation

Build a grid approximation for the model we've built so far using the following steps:

1) Define the grid (decide how many points to use in estimateing the posterior, and make a list of the parameter value on the grid)
2) Compute the value of the prior at each parameter value on the grid.
3) Compute the likelihood at each parameter value.

4) Compute the unstandardized posterior at each parameter value, by multiplying the prior by the likelihood.
5) Finally, standardize the posterior, by dividing each value by the sum of all values.

Here we will make a grid of just 20 points:

```
# R code 2.4: grid approximation for 20 points

# 1) define grid
p_grid <- seq( from=0 , to=1 , length.out=20 )

# 2) define prior
prior <- rep( 1 , 20 )

# 3) compute likelihood at each value in grid
likelihood <- dbinom( 6 , size=9 , prob=p_grid )

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

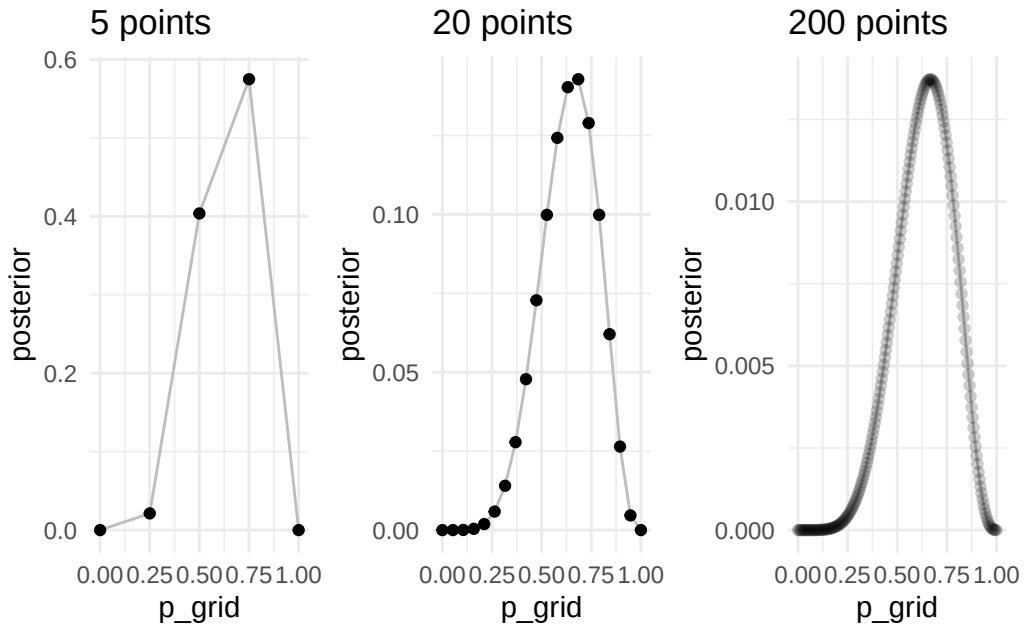
```
# R code 2.4 (my tidyverse version)

fig_grid_20 <-
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "20 points")
```

```
## Repeat for 200 points

# 1) define grid
p_grid <- seq( from=0 , to=1 , length.out=200 )

# 2) define prior
```

```r
prior <- rep( 1 , 200 )

# 3) compute likelihood at each value in grid
likelihood <- dbinom( 6 , size=9 , prob=p_grid )

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```r
## Repeat for 5 points

# 1) define grid
p_grid <- seq( from=0 , to=1 , length.out=5 )

# 2) define prior
prior <- rep( 1 , 5 )

# 3) compute likelihood at each value in grid
likelihood <- dbinom( 6 , size=9 , prob=p_grid )

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```r
fig_grid_5 + fig_grid_20 + fig_grid_200
```

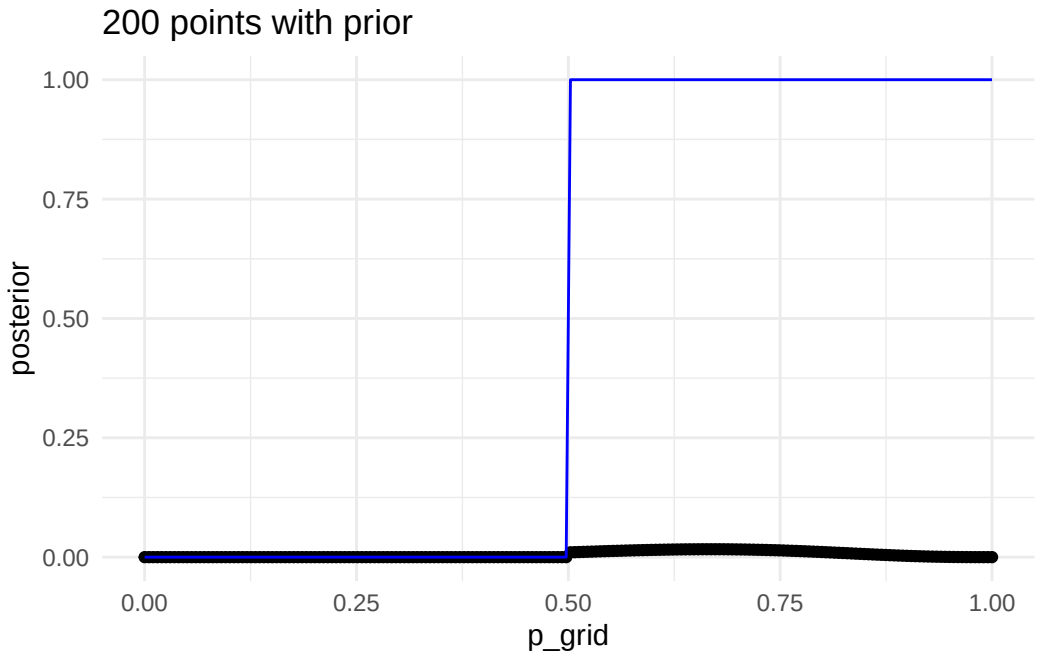


1.2.4 Adjust priors

```
## Repeat for 5 points

# 1) define grid
p_grid <- seq( from=0 , to=1 , length.out=200 )

# 2) define prior
prior <- ifelse( p_grid < 0.5 , 0 , 1 )
# prior <- exp( -5*abs( p_grid - 0.5 ) )

# 3) compute likelihood at each value in grid
likelihood <- dbinom( 6 , size=9 , prob=p_grid )

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```
# fig_grid_5_prior <-
  cbind(p_grid, posterior, prior) |>
```

```
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  geom_line(aes(y = prior), colour = "blue") +
  theme_minimal() +
  labs(title = "200 points with prior")
```



Okay a bit wonky (different y-scales), but we get the point.

1.2.5 Quadratic approximation

We use the `quap` function from the `rethinking` package. This function takes a formula which defines the probability of the data in the prior.

```
# R code 2.6
globe.qa <- quap( alist( W ~ dbinom( W+L ,p) , # binomial likelihood
                         p ~ dunif(0,1) # uniform prior
                         ), data=list(W=6,L=3) )
# display summary of quadratic approximation
```

```
precis( globe.qa )
```

```
      mean        sd      5.5%     94.5%
p 0.6666663 0.1571339 0.4155361 0.9177966
```

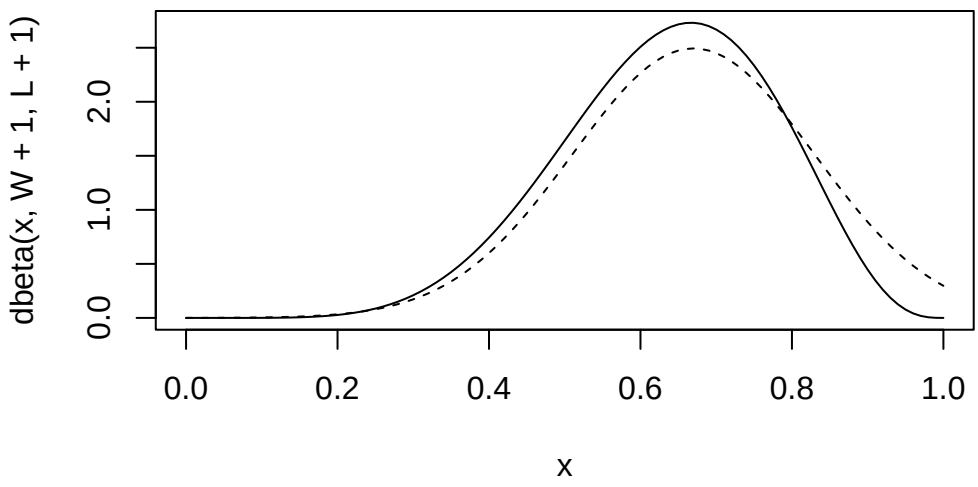
Print output with `precis()`:

- `mean`: posterior mean value of p (peak of the curvature)
- `sd`: the curvature; standard deviation of the posterior distribution
- `5.5%`-94.5%: 89% percentile

This output can be read *Assuming the posterior is Gaussian, it is maximized at 0.67, and its standard deviation is 0.16.*

1.2.6 Check curvature

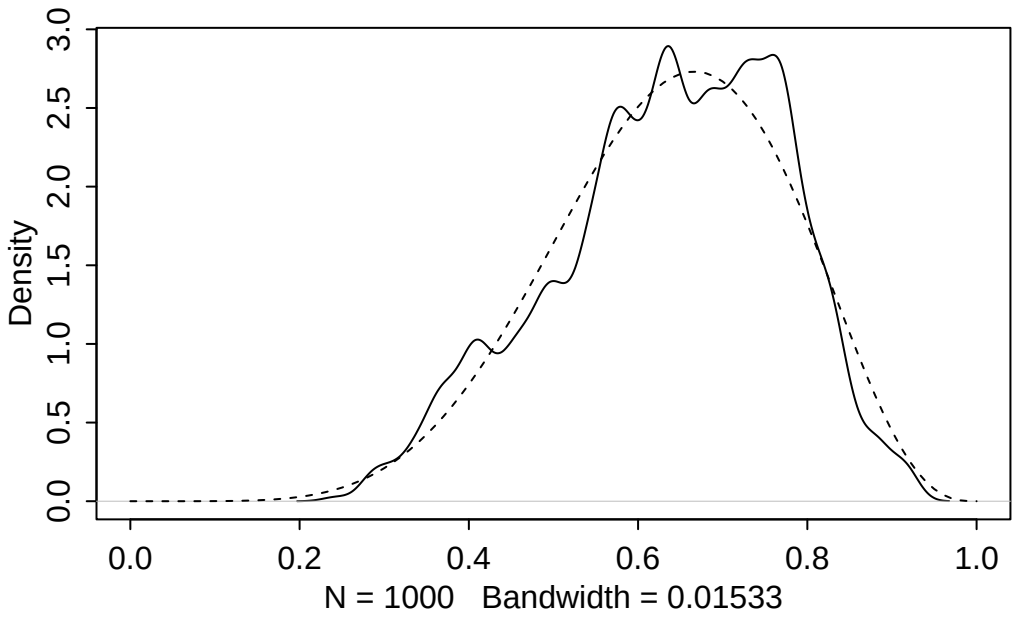
```
# analytical calculation
W <- 6
L <- 3
curve( dbeta( x , W+1 , L+1 ) , from=0 , to=1 )
# quadratic approximation
curve( dnorm( x , 0.67 , 0.16 ) , lty=2 , add=TRUE )
```



1.2.7 Markov chain Monte Carlo

```
# R code 2.8

n_samples <- 1000
p <- rep(NA , n_samples)
p[1] <- 0.5
W <- 6
L <- 3

for (i in 2:n_samples) {
    p_new <- rnorm(1 , p[i - 1] , 0.1)
    if (p_new < 0)  p_new <- abs(p_new)
    if (p_new > 1) p_new <- 2 - p_new
    q0 <- dbinom(W , W + L , p[i - 1])
    q1 <- dbinom(W , W + L , p_new)
    p[i] <- ifelse(runif(1) < q1 / q0 , p_new , p[i - 1])
  }
```

```
# R code 2.9
dens( p , xlim=c(0,1) )
curve( dbeta( x , W+1 , L+1 ) , lty=2 , add=TRUE )
```



1.3 Practice

1.3.1 Easy

2E1: $\frac{Pr(rain,Monday)}{Pr(Monday)}$ is read *the probability of rain on Monday*

2E2: $Pr(Monday|rain)$ is read *the probability that it is Monday, given that it is raining*

2E3: $Pr(Monday|rain)$ is read *the probability that it is Monday, given that it is raining*

2E4: *The probability of water in 0.7* means that...

1.3.2 Medium

1.3.2.1 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.

1. W W W
2. W W W L
3. L W W L W W W

W W W

```
# R code 2.4: grid approximation for 20 points

# 1) define grid
p_grid <- seq( from=0 , to=1 , length.out=20 )

# 2) define prior
prior <- rep( 1 , 20 )

# 3) compute likelihood at each value in grid
likelihood <- dbinom(3 , size=3 , prob=p_grid )
# 3 waters (successes) with 3 tosses (size)

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```
# R code 2.4 (my tidyverse version)

fig_grid_www <-
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "W W W")
```

W W W L

```
# R code 2.4: grid approximation for 20 points

# 3) compute likelihood at each value in grid
likelihood <- dbinom(3 , size=4 , prob=p_grid )
# 3 waters (successes) with 3 tosses (size)

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

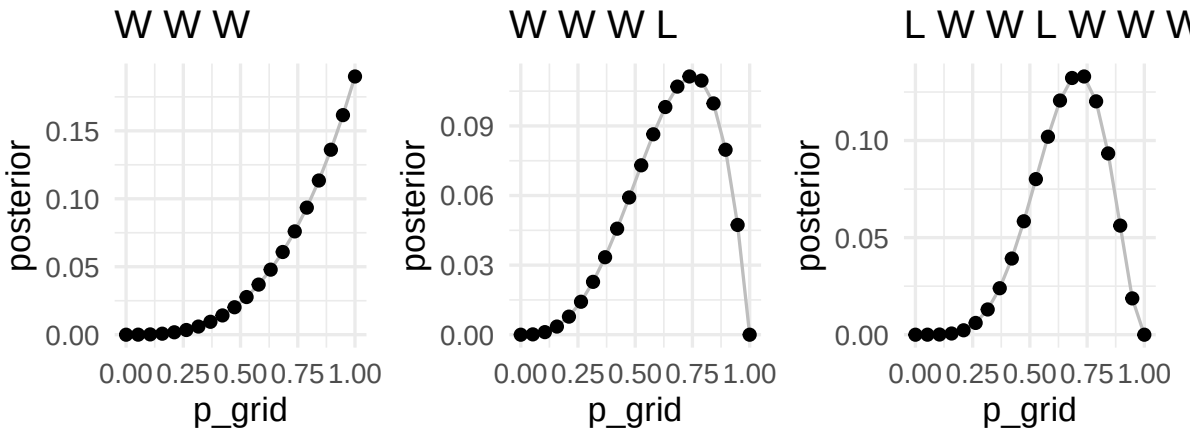
```
# R code 2.4 (my tidyverse version)

fig_grid_wwwl <-
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "W W W L")
```

L W W L W W W

```r
# R code 2.4: grid approximation for 20 points

# 3) compute likelihood at each value in grid
likelihood <- dbinom(5 , size=7 , prob=p_grid )
# 3 waters (successes) with 3 tosses (size)

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```r
# R code 2.4 (my tidyverse version)

fig_grid_lwwlwww <-
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "L W W L W W W")
```

```r
library(patchwork)
fig_grid_www + fig_grid_wwwl + fig_grid_lwwlwww
```



1.3.2.2 2M2:

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

```
# R code 2.4: grid approximation for 20 points

# 1) define grid
p_grid <- seq( from=0 , to=1 , length.out=20 )

# 2) define prior
prior <- ifelse(p_grid < .5, 0, 1)

# 3) compute likelihood at each value in grid
likelihood <- dbinom(3 , size=3 , prob=p_grid )
# 3 waters (successes) with 3 tosses (size)

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```
# R code 2.4 (my tidyverse version)

fig_grid_www <-
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "W W W")
```

W W W L

```
# R code 2.4: grid approximation for 20 points

prior <- ifelse(p_grid < .5, 0, 1)
```

```
# 3) compute likelihood at each value in grid
likelihood <- dbinom(3 , size=4 , prob=p_grid )
# 3 waters (successes) with 3 tosses (size)

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```
# R code 2.4 (my tidyverse version)

fig_grid_wwwl <-
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "W W W L")
```

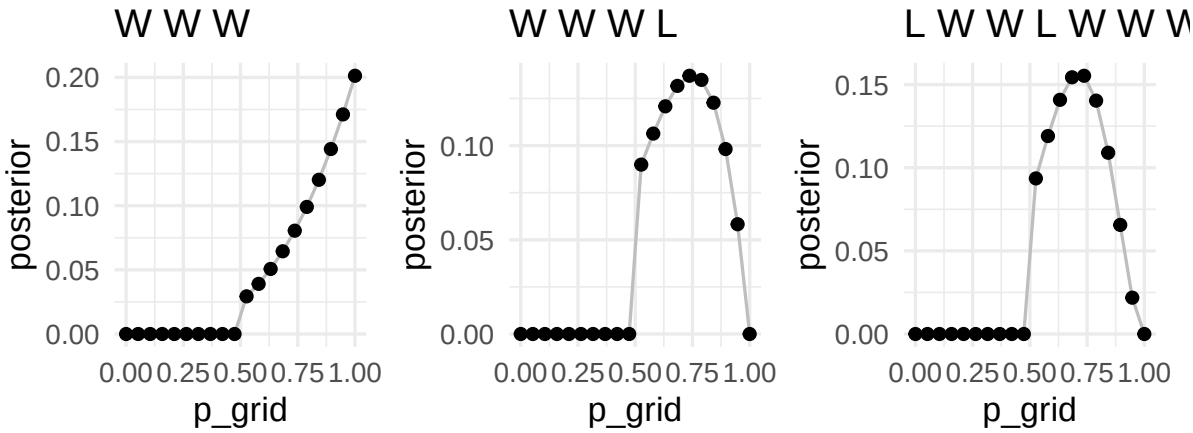
L W W L W W W

```
# R code 2.4: grid approximation for 20 points

prior <- ifelse(p_grid < .5, 0, 1)

# 3) compute likelihood at each value in grid
likelihood <- dbinom(5 , size=7 , prob=p_grid )
# 3 waters (successes) with 3 tosses (size)

# 4) compute product of likelihood and prior
unstd.posterior <- likelihood * prior

# 5) standardize the posterior, so it sums to 1
posterior <- unstd.posterior / sum(unstd.posterior)
```

```
# R code 2.4 (my tidyverse version)

fig_grid_lwwlwww <-
```

```
  cbind(p_grid, posterior) |>
  as_tibble() |>
  ggplot() +
  aes(x = p_grid, y = posterior) +
  geom_line(colour = "grey") +
  geom_point() +
  theme_minimal() +
  labs(title = "L W W L W W W")
```

```
library(patchwork)
fig_grid_www + fig_grid_wwwl + fig_grid_lwwlwww
```



1.3.2.3 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.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.1}$$

$$P(Earth|land) = \frac{P(land|Earth)P(Earth)}{P(land)} \tag{1.2}$$

$$0.23 = \frac{.3 * .5}{P(land)} \tag{1.3}$$

$$0.23 = \frac{.3 * .5}{P(land|Earth) * P(land|Mars)} \tag{1.4}$$

$$0.23 = \frac{.3 * .5}{(0.3 * 0.5) + (1 * 0.5)} \tag{1.5}$$

$$0.23 = \frac{.3 * .5}{((0.3 + 1)/2)} \tag{1.6}$$

```
0.3*0.5/((0.3+1)/2) #0.2307692
```

```
[1] 0.2307692
```

```
#prob(earth|land) = (prob(land/earth)*prob(earth))/prob(land)
```

2M4:

2M5:

2M6:

2M7:

1.3.3 Hard

2 Sampling the Imaginary

Chapter 3

Also following Solomon Kurz's E-book.

```
options(scipen=999)
```

```
pacman::p_load(
  tidyverse,
  brm,
  rethinking,
  knitr,
  kableExtra
)
```

2.1 Thinking in probabilities vs. frequencies

- let's say there's a blood test that detects vampirism 95% of the time
  - so **Pr(positive test result|vampire) = 0.95**
- there is also an error rate, in that it makes falsep ositive 1% of the time
  - so **Pr(positive test result|mortal) = 0.01**
- also, vampires are rare, comprising of only 0.1% of the population
  - **Pr(vampire) = 0.001**
- using Bayes' theorem to find the probability of correctly identifying a vampire with this test (**Pr(vampire|position)**), we use Equation 2.1

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

Table 2.1: Probabilties

pr_positive_vampire	0.950
pr_positive_mortal	0.010
pr_vampire	0.001
pr_positive	0.011
pr_vampire_positive	0.087

Table 2.2: Frequencies

pr_vampire	0.001
pr_positive_vampire	0.950
pr_positive_mortal	0.001
pr_positive	1094.000
pr_vampire_positive	0.087

- we can compute this using these known probabilities
- or to look at observed frequencies
- McElreath points out that most people find it more intuitive to think about counts, rather than p;ugging probabilities into the right place in Bayes' theorem, which is often called the *frequency format* or *natural frequencies*

  Working with samples transforms a problem in calculus into a problem in data summary, into a frequency format problem. - p. 51

Table 2.3: Useful terms and functions

term/function	definition
‘rbinom(n, size, prob)‘	
‘PI(distribution, prob)‘	rethinking package: compute percentile compatiability interval from posterior
‘HPDI(distribution, prob)‘	produces interval that best representats the parameter values most consistent wi

2.2 Sampling from a grid-like approximate posterior

2.3 Sapling to summarise

2.4 Sampling to simulate prediction

2.5 Summary

2.6 Terms and concepts

2.7 Practice

2.7.1 Easy

> The Easy problems use the samples from the posterior distribution for the globe tossing example. This code will give you a specific set of samples, so that you can check your answers exactly…Use the values in samples to answer the questions that follow. - p. 68

```
# R code 3.27
p_grid <- seq( from=0 , to=1 , length.out=1000 )
prior <- rep( 1 , 1000 )
likelihood <- dbinom( 6 , size=9 , prob=p_grid )
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

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

```
hist(samples)
```

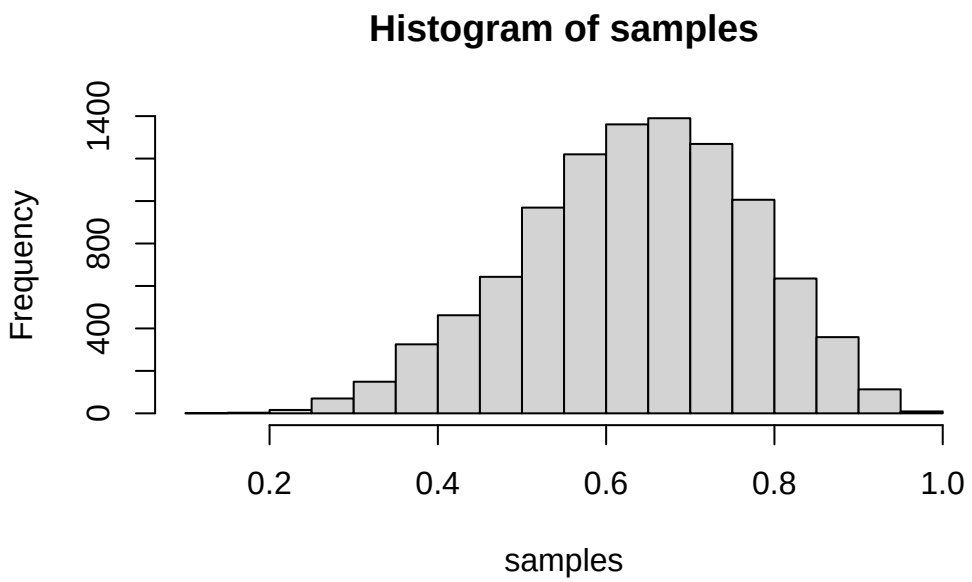


Figure 2.1: Histogram of samples

2.7.1.1 Easy 1

How much posterior probability lies below p = 0.2?

```
length(samples[samples < 0.2])/length(samples)
```

```
[1] 0.0004
```

```
# or
sum(samples < .2)/length(samples)
```

```
[1] 0.0004
```

2.7.1.2 Easy 2

How much posterior probability lies above p = 0.8?

```
sum(samples > .8)/length(samples)
```

```
[1] 0.1116
```

2.7.1.3 Easy 3

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

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

```
[1] 0.888
```

2.7.1.4 Easy 4

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

```
quantile(samples, .2)
```

```
      20%
0.5185185
```

2.7.1.5 Easy 5

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

```
quantile(samples, .8)
```

```
      80%
0.7557558
```

2.7.1.6 Easy 6

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

I used the `rethinking::PI()` function, which is wrong:

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

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

Should've used the `rethinking::HDPI` function, because we wanted the *highest posterior density interval*.

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

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

```
pacman::p_load(tidybayes)
tidybayes::mode_hdi(samples, .width = .66)
```

```
          y      ymin      ymax .width .point .interval
1 0.6559935 0.5205205 0.7877878   0.66   mode       hdi
```

2.7.1.7 Easy 7

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

I used the `rethinking::HDPI()` function, which is wrong:

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

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

I should've used the `rethinking::PI()` function, because e wanted the *conventional* percentile interval.

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

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

2.7.2 Medium

2.7.2.1 Medium 1

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.

```
# R code 3.27
p_grid <- seq( from=0 , to=1 , length.out=1000 )
prior <- rep( 1 , 1000 )
likelihood <- dbinom( 8 , size=15 , prob=p_grid )
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

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

```
hist(samples_m)
```

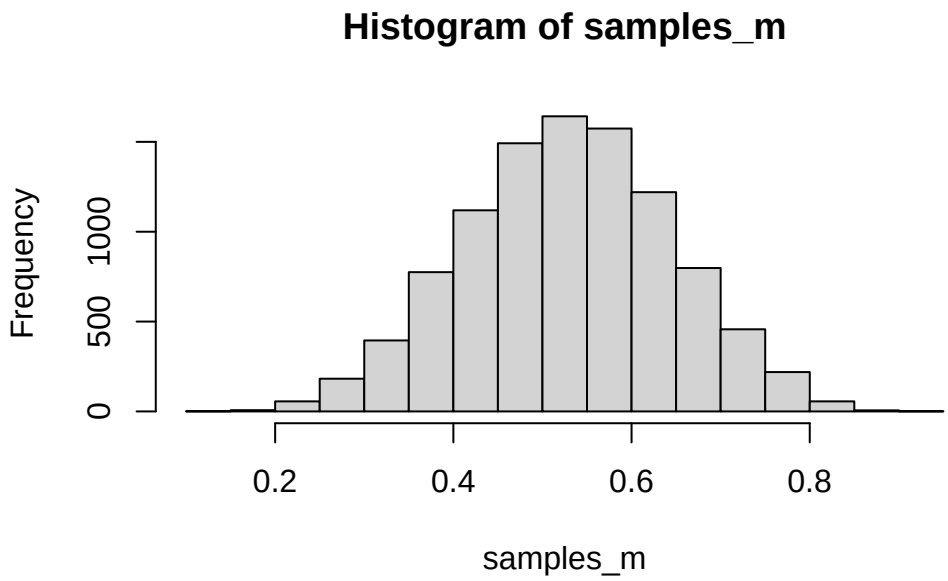
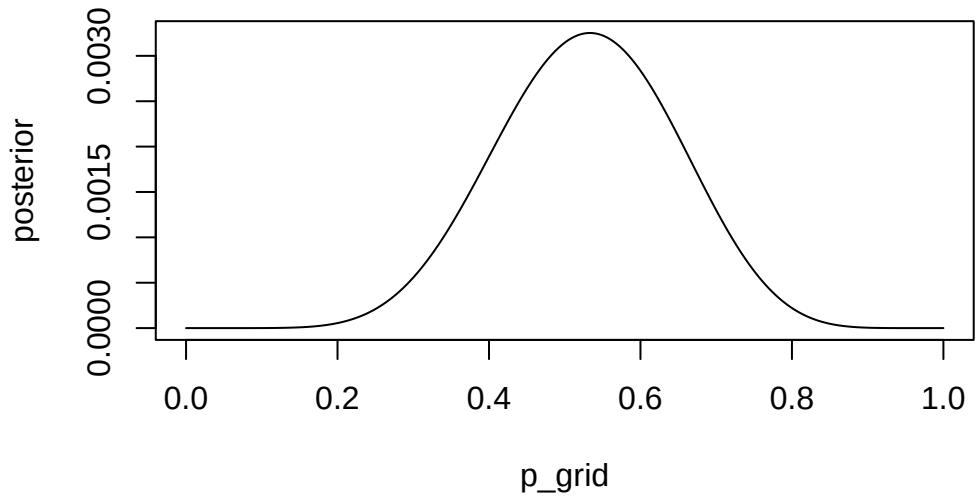


Figure 2.2: Histogram of samples

Correct, but McElreath uses this plot:

```
plot(posterior ~ p_grid, type = "l")
```



2.7.2.2 Medium 2

> 3M2. Draw 10,000 samples from the grid approximation from above. Then use the samples to calculate the 90% HPDI for p.

10000 samples is the same thing, no? `1e4` in scientific notation = 10000. I'll re-run it, but since we've set our seed it shouldn't be any different.

```
samples_10g <- sample( p_grid , prob=posterior , size=1e4 , replace=TRUE )
```

And caldulate the 90% HPDI for p:

```
HPDI(samples_m, prob = .9)
```

```
    |0.9      0.9|
0.3343343 0.7217217
```

After checking teh answers, I was right, it is the same as above, it's just that question M1 didn't also ask me to draw the random samples.

2.7.2.3 Medium 3

3 Geocentric Models

Chapter 4 in McElreath (2020)

My notes for this chapter are a bit muddled, I did a mix of reading the textbook and watching the YouTube video. Unfortuantely the YouTube video doesn't map onto the chapter very transparently and even has code that isn't in the current textbook version, so some concepts might be repeated/out of order.

Resources

- YouTube lecture (2023): Geocentric Models
- Solomon Kurz: Linear models

Set-up

```
pacman::p_load(
  rethinking,
  tidyverse,
  janitor,
  knitr,
  kableExtra
)
```

```
theme_set(theme_bw())
```

Permission to be confused

From ~ 20 minute mark from the YouTue video

- you don't need to understand everything all at once
  - math, concepts, code, installation problems etc.
- it's important to keep some *flow*
  - keep moving forward despite the resistance of confusion/struggle
  - things shouldn't be too hard, because then you stop moving
  - if it's too easy, you don't feel any resistance
  - feeling confused means you're paying attention :)

The Geocentric Model

- planets have been observed to have a zig-zag orbit from the perspective of Earth
- Geocentric model/Telomeric model
  - used to predict the pattern of planets orbit, with the prior of Earth at the centre of the solar system
- the geocentric model was very accurate
  - however, the orbits of the planets are elliptical and orbit around the *sun*, not the earth
- so models built on inaccurate assumptions can still be very accurate in their predictions
- the orbits of the planets are embedded witin each other (based on their proximity to the sun), so there is a point when two planets' orbits are closer to each other, i.e., when their orbits have them closer together
  - which is how the perception of Mars' zig-zag orbit (and other planets, but it's more stricking for Mars)
- in 1809: Bayesian argument for normal error and least-squares estimation from Gauss
- **Geocentric**: describes association, makes predictions, but mechanisticalistically wrong
  - the problem is not the model itself, rather some external causal model we project onto the model
  - geocentric models are still extremely useful, what's wrong is believing they're *true* reflections of e.g., the solar system

  - there's also nothing wrong with linear regression itself, as long as we don't believe they're an accurate mechanistic model of the system we're studying

- **Gaussian**: abstracts from generative error model, replaces with normal distribution, mechanistically silent

  - usually a pretty good approximation of associations
  - we will be using gaussian error models

- useful when handled *with care*

  - many special cases: ANOVA, ANCOVA, t-test, etc.

3.1 Why normal distributions are normal

3.1.1 Normal by multiplication

3.1.2 Normal by log-multiplication

3.1.3 Using Gaussian distributions

3.2 A language for describing models

3.3 A Gaussian model of height

```
# load Howell1 data from rethinking package
data(Howell1)
df_weight <- Howell1
```

```
head(df_weight)
```

```
   height   weight age male
1 151.765 47.82561  63    1
2 139.700 36.48581  63    0
3 136.525 31.86484  65    0
4 156.845 53.04191  41    1
5 145.415 41.27687  51    0
6 163.830 62.99259  35    1
```

x	x	x	x	x
138.3	27.6	81.1	165.7	
35.6	14.7	9.4	54.5	
29.3	20.7	1.0	66.1	
0.5	0.5	0.0	1.0	

```
summary(df_weight)
```

```
     height           weight            age              male        
 Min.   : 53.98   Min.   : 4.252   Min.   : 0.00   Min.   :0.0000  
 1st Qu.:125.09   1st Qu.:22.008   1st Qu.:12.00   1st Qu.:0.0000  
 Median :148.59   Median :40.058   Median :27.00   Median :0.0000  
 Mean   :138.26   Mean   :35.611   Mean   :29.34   Mean   :0.4724  
 3rd Qu.:157.48   3rd Qu.:47.209   3rd Qu.:43.00   3rd Qu.:1.0000  
 Max.   :179.07   Max.   :62.993   Max.   :88.00   Max.   :1.0000  
```

- age ranges from babies to seniors
- sex as binary
- height in cm
- weight in kg

`precis()` function from rethinking package:

```
precis(df_weight) |>
  as_tibble() |>
  kable(digits = 1) |>
  kable_styling()
```

To define heights as normally distributed, we write 3.1, where i stands for index. 3.1 is saying that all the golem (i.e., our model) knows about each observed height measurement is defined by the same normal distribution with mean μ and standard deviation σ.

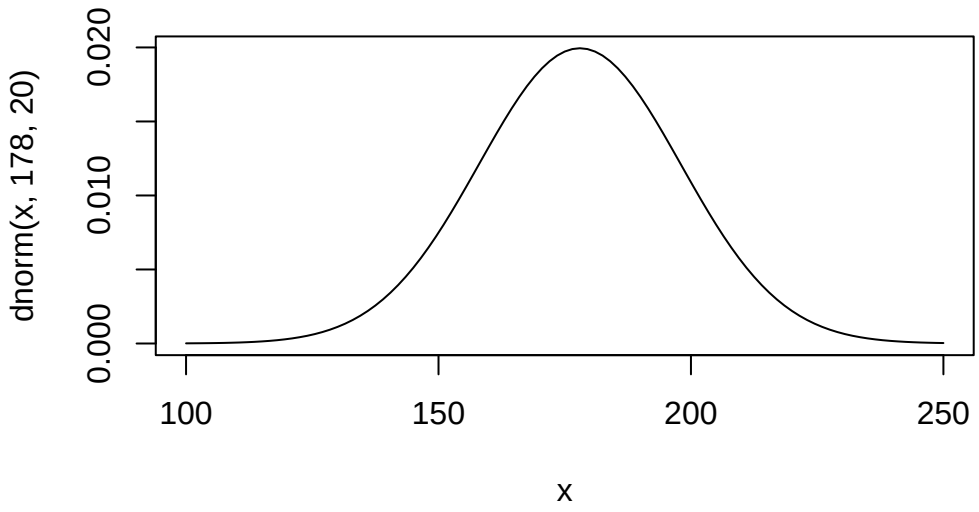
$$h_i \sim Normal(\mu, \sigma) \tag{3.1}$$
$$\mu \sim Normal(178, 20) \tag{3.2}$$
$$\sigma \sim Uniform(0, 50) \tag{3.3}$$

3.2 represents our *prior* for μ, where our assumed mean is 178 cm (because Richard McElreath is 178cm tall), with a 95% probability between +/- 40cm (because if the SD is 20cm, 20 x 1.96 is approximately 20 x 2).

```r
# R chunk 4.10

# subset data for adults only
d2 <- df_weight |> filter(age > 18)
```

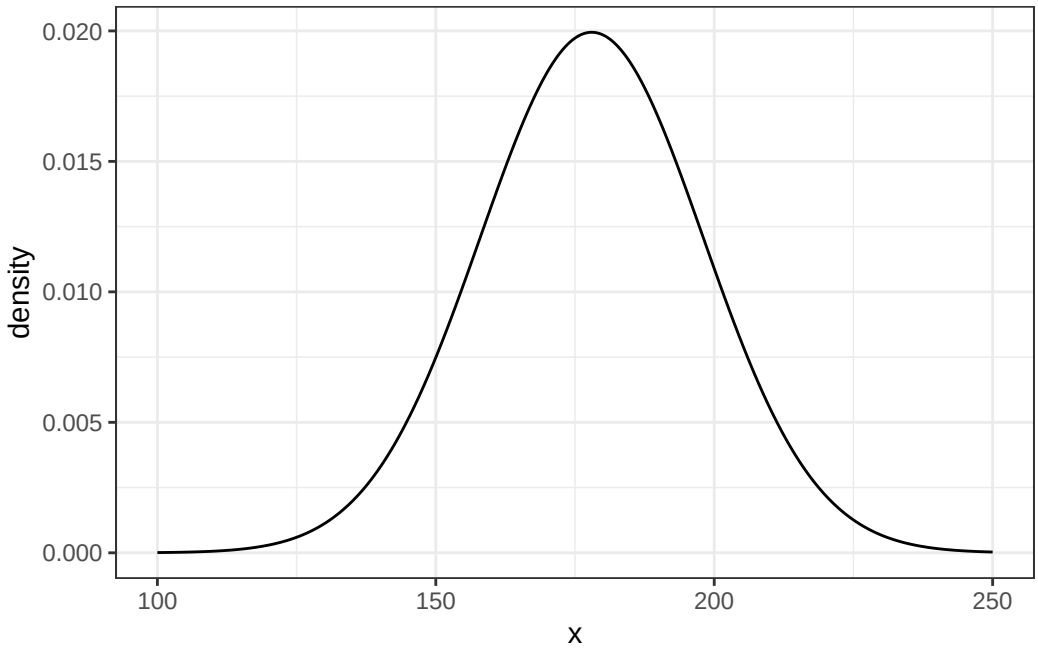
We can plot our priors:

```r
# R code chunk 4.12
curve(dnorm(x, 178, 20), from = 100, to = 250)
```



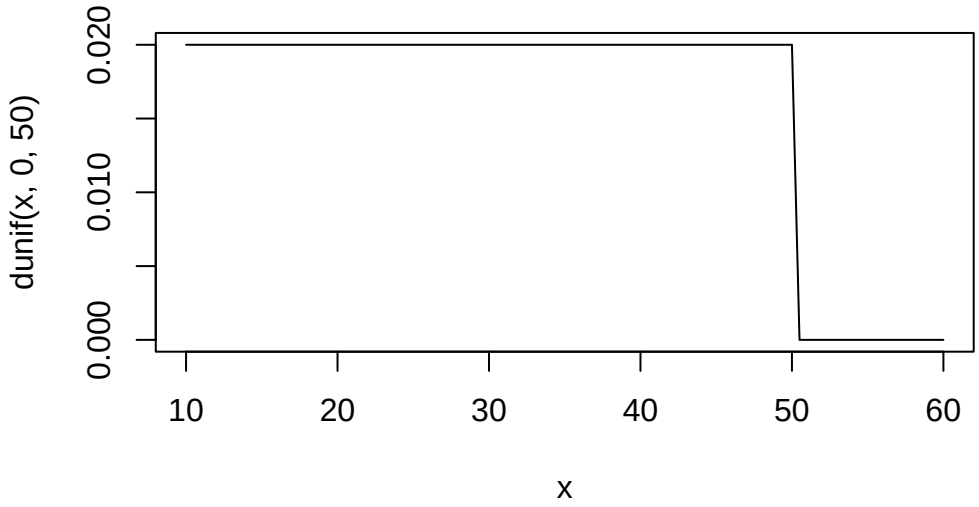
Or, with ggplot (from Solomon Kurz):

```r
ggplot(data = tibble(x = seq(from = 100, to = 250, by = .1)),
       aes(x = x, y = dnorm(x, mean = 178, sd = 20))) +
  geom_line() +
  ylab("density")
```



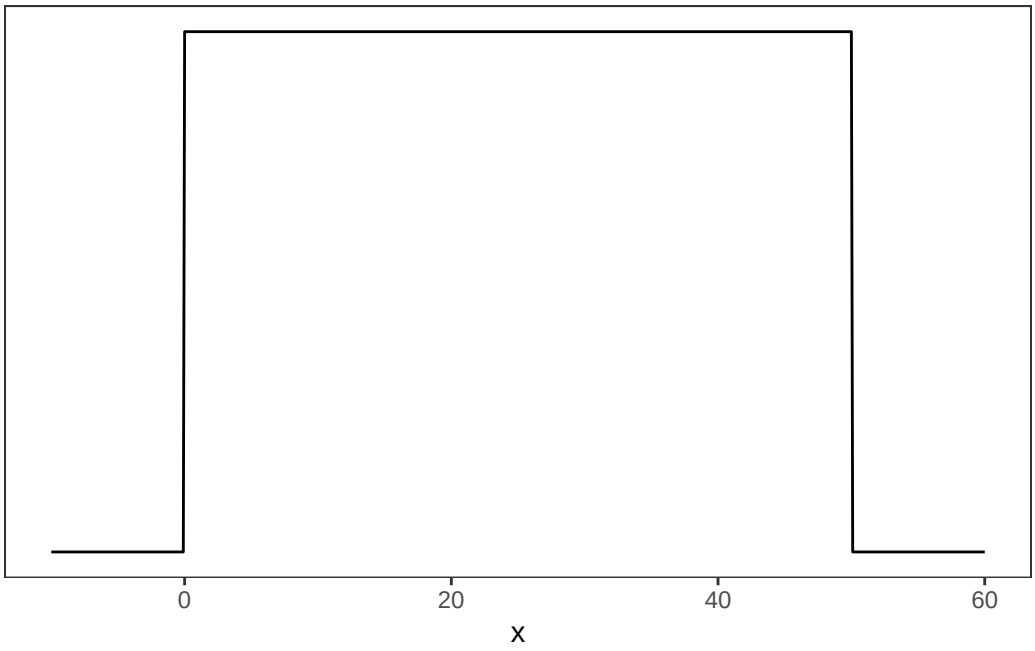
We can also view our prior for σ, which is a flat (uniform) prior:

```
# R code chunk 4.13
curve(dunif(x, 0, 50), from = 10, to = 60)
```



And with ggplot (from Solomon Kurz):

```
tibble(x = seq(from = -10, to = 60, by = .1)) %>%

  ggplot(aes(x = x, y = dunif(x, min = 0, max = 50))) +
  geom_line() +
  scale_y_continuous(NULL, breaks = NULL) +
  theme(panel.grid = element_blank())
```

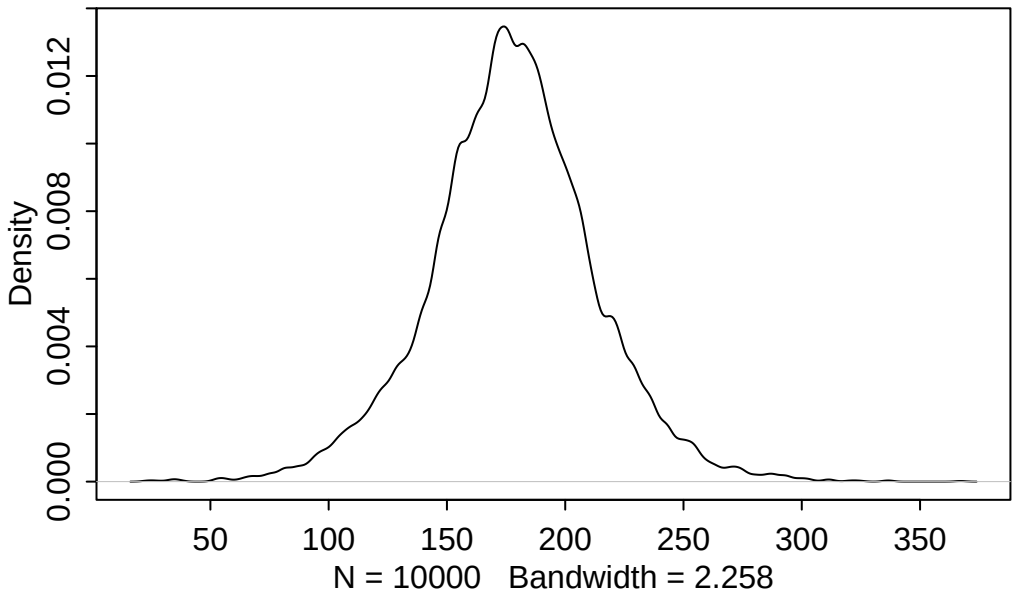


A standard deviation must be positiion, and so we've bound it at zero. Picking the upperbound is another question: in our case a sd of 50cm would imply that 95% of individual heights lie within 100cm of the average height (again, because SD * 1.96), which is a very large range.

Now it's time for a ~prior predictive~ simulation, for which we need our chosen priors for h, μ, and σ, which imply a joint prior distribution of individual heights.

Let's quickly simiulate heights by sampling from our prior. We can process priors just like posteriors, because every posterior is also potentially a prior for a subsequent analysis.

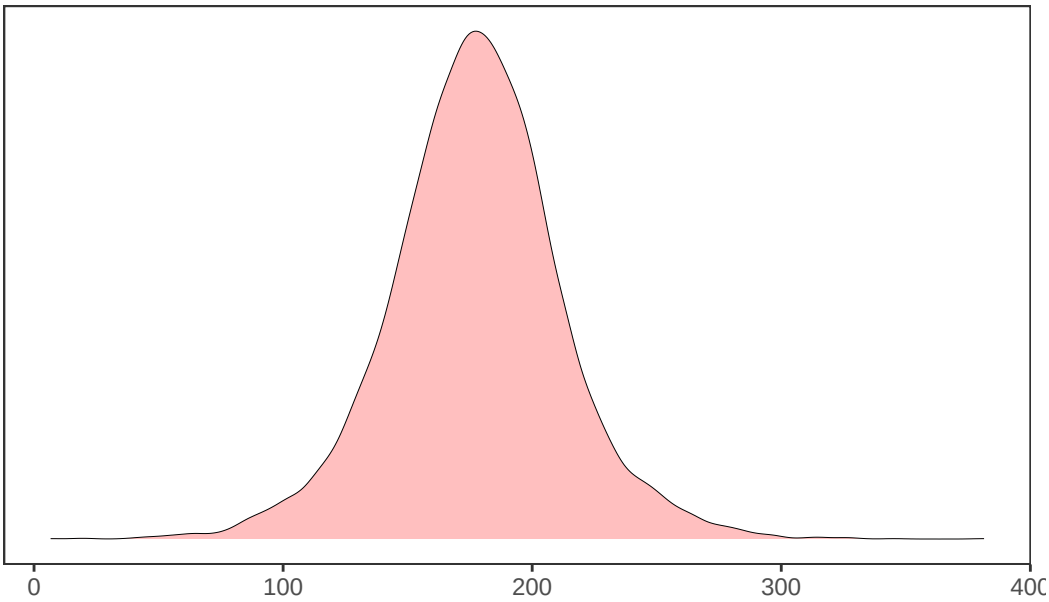
```
# R code 4.14
sample_mu <- rnorm(1e4, 178, 20)
sample_sigma <- runif(1e4, 0, 50)
prior_h <- rnorm(1e4, sample_mu, sample_sigma)
dens(prior_h)
```



And sampling from both μ and σ priors to get a prior probability of heights:

```
n <- 1e4

set.seed(4)
tibble(sample_mu    = rnorm(n, mean = 178,        sd  = 20),
       sample_sigma = runif(n, min  = 0,          max = 50)) %>%
  mutate(x = rnorm(n, mean = sample_mu, sd  = sample_sigma)) %>%

  ggplot(aes(x = x)) +
  geom_density(fill = "red", linewidth = 0, alpha = .25) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = expression(Prior~predictive~distribution~"for"~italic(h[i])),
       x = NULL) +
  theme(panel.grid = element_blank())
```

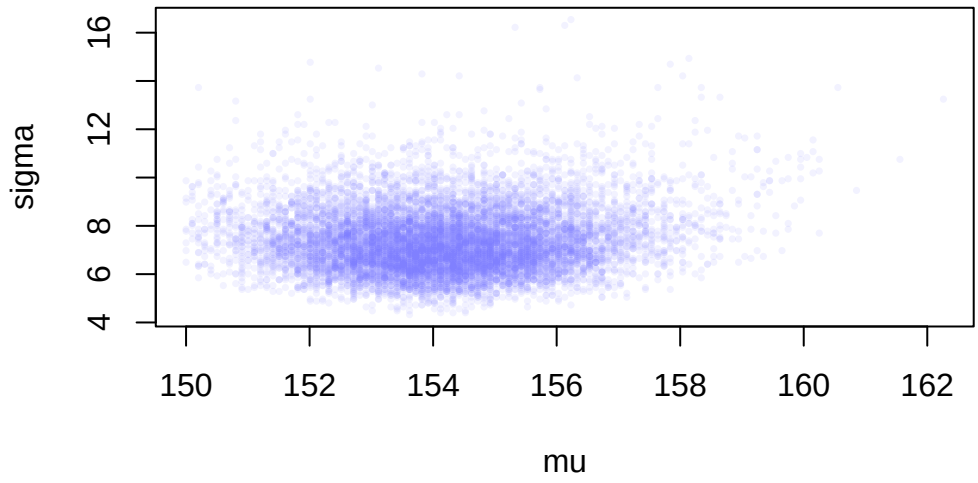
Prior predictive distribution for h_i



3.3.1 Finding the posterior distribution with `quap`

```
# R chunk 2.23
d3 <- sample( d2$height , size=20 )
```

```
# R chunk 4.24
mu.list <- seq( from=150, to=170 , length.out=200 )
sigma.list <- seq( from=4 , to=20 , length.out=200 )
post2 <- expand.grid( mu=mu.list , sigma=sigma.list )
post2$LL <- sapply( 1:nrow(post2) , function(i) sum( dnorm( d3 , mean=post2$mu[i] , sd=pos
post2$prod <- post2$LL + dnorm( post2$mu , 178 , 20 , TRUE ) + dunif( post2$sigma , 0 , 50
post2$prob <- exp( post2$prod - max(post2$prod) )
sample2.rows <- sample( 1:nrow(post2) , size=1e4 , replace=TRUE , prob=post2$prob )
sample2.mu <- post2$mu[ sample2.rows ]
sample2.sigma <- post2$sigma[ sample2.rows ]
plot( sample2.mu , sample2.sigma , cex=0.5 , col=col.alpha(rangi2,0.1) , xlab="mu" , ylab=
```




```
# R code 4.26
data(Howell1)
d <- Howell1
d2 <- d[d$age >= 18 , ]
```

And define our model and place it into an `alist`.

```
# R code 4.27
flist <-
  alist(height ~ dnorm(mu , sigma) ,
        mu ~ dnorm(178 , 20) ,
        sigma ~ dunif(0 , 50)
        )
```

Fit a model to the data in `d2`.

```
# R code 4.28
m4.1 <- quap(flist, data = d2)
```

Take a look.

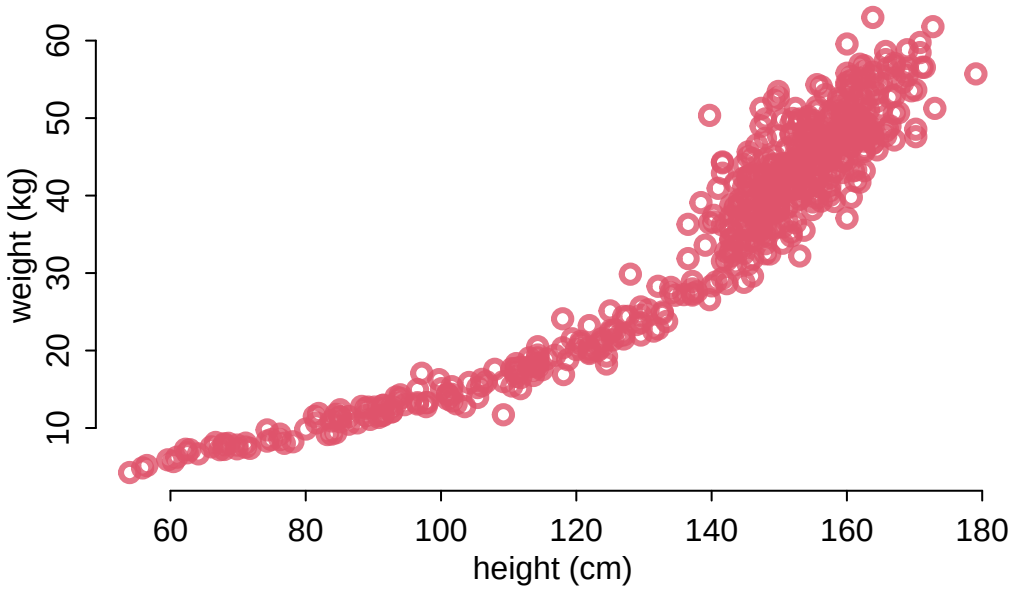
```
# R code 4.29
precis(m4.1)
```

```
            mean        sd       5.5%      94.5%
mu    154.606899 0.4119494 153.948524 155.265273
sigma   7.730482 0.2913059   7.264919   8.196045
```

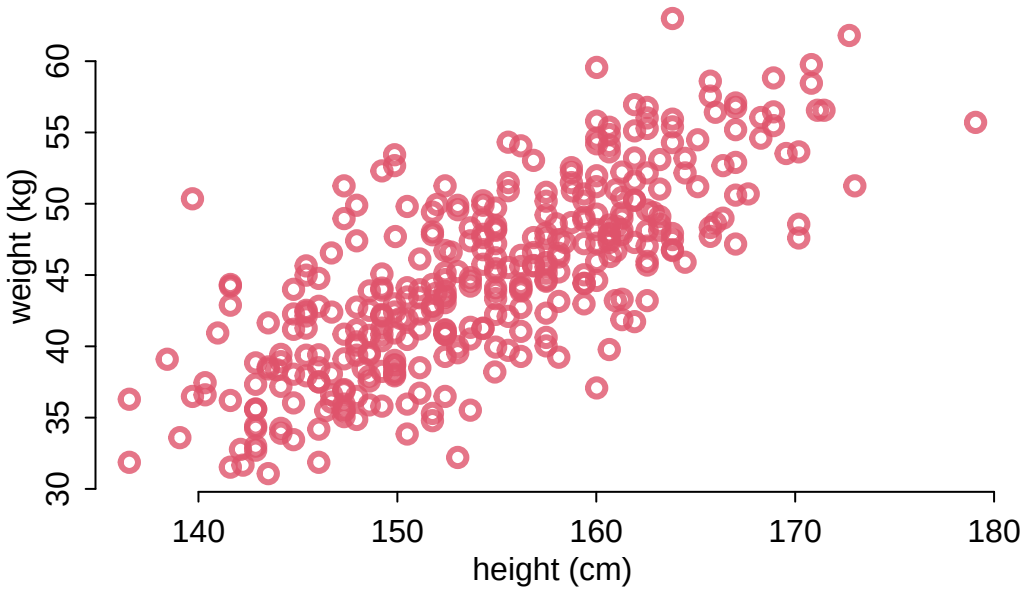
3.3.2 Simulate weights

From the GitHub script 03_howell etc.

```
data(Howell1)
d <- Howell1

# draw data

blank(bty="n")

col2 <- col.alpha(2,0.8)
plot( d$height , d$weight , col=col2 , lwd=3 , cex=1.2 , xlab="height (cm)" , ylab="weight
```




```
d <- d[ d$age>=18 , ]
plot( d$height , d$weight , col=col2 , lwd=3 , cex=1.2 , xlab="height (cm)" , ylab="weight
```



3.3.3 Describing models

(video c. 31:00)

Conventional statistical model notaton:

1. list the variables in the model
2. define each variable as a deterministic or distributional function of the other variables

- where does each variable get its value?

```
# function to simulate weights of individuals from height
sim_weight <- function(H,b,sd) {
  U <- rnorm(length(H), 0, sd)
  W <- b*H + U
  return(W)
}
```

- this is a function of unobserved U, which has normal error
- where W is a deterministic function of H and U together
- this can be re-written with standard statistical notation as followed:

$$W_i = \beta H_i + U_i \tag{3.4}$$
$$U_i \sim Normal(0, \sigma) \tag{3.5}$$
$$H_i \sim Uniform(130, 170) \tag{3.6}$$

Each of these is mirrored in a line of code. Along the left of the equations are the variables, along the right are their definitions. The subscripts indicate individuals (indexing).

- $=$ indicates a **deterministic** relationship
- $\sim$ indicates a **distributional** relationship

Equation 3.4 is the equation for expected weight. Once we know the values to the right of this equation we can assign values to W. Equation 3.5 represents Gaussian error with standard deviation sigma. Equation 3.6 defines height as uniformly distributed from 130cm to 170cm. These are all defined in the code chunk above, but written in the opposite order. In the code it's written in order of execution, but the statistical conventional notation sometimes has arbitrary order, but conventionally is stated in the opposite direction as the code is written because you start with the most general definition (3.4) which depends on the other variables, and then you see how the variables W depends upon are defined. There's a hierarchy that makes sense.

3.3.4 Estimator

We want to estimate how the average weight changes with height, i.e., how the average weight changes with height.

$$E(W_i|H_i) = \alpha + \beta H_i \tag{3.7}$$

Where $E(W_i|H_i)$ is the average weight conditional on height, α is the intercept and β is our slope. So each height has a different average weight.

3.3.5 Posterior distribution

Good Bayesians estimate a posterior. The equation below is the same as that for when we tossed the globe, but we just have more unknowns now.

$$Pr(\alpha, \beta, \sigma|H_i, W_i) = \frac{Pr(W_i|H_i, \alpha, \beta, \sigma)Pr(\alpha, \beta, \sigma)}{Z} \tag{3.8}$$

- $Pr(\alpha, \beta, \sigma | H_i, W_i)$ is the posterior probability of a specific regression line, which is defined by , , and (unobserved variables). H and W are our *observed* data, we don't need a posterior distribution for them, they're observed (the likelihood).
- $Pr(W_i | H_i, \alpha, \beta, \sigma)$: garden of forking data
- $Pr(\alpha, \beta, \sigma)$: our priors
- Z: a normalising constant that we don't need to pay too much attention to, it ensures the lefthand side of the equation is a proper probability

W is normally distributed with mean that is a a linear function of H.

These models are typically written as below:

$$W_i = Normal(\mu_i, \sigma) \tag{3.9}$$
$$\mu_i \sim \alpha + \beta H_i \tag{3.10}$$

3.3.6 Grid approximat posterior

All posterior distributions begin with a single observations. There's no minimum number of observations in Bayesian inference.

3.3.7 Quardratic approximation

- we'll approximate the posterior distribution as a multivariate Gaussian distribution. We can do that because posterior distributions are typically multivariate Gaussian distributions if there are enough observations. A tool that does this for us is called `quap()` in the `statrethinking` package.

The Gaussian distribution is quadratic, and in the log-space the Gaussian distribution is a perfect parabola (i.e., quadratic).

When there are no observations, what does the model believe? These are our priors with no likelihood.

```
# this wasn't working...
m3.1 <-
  quap(
    alist(
      W ~ dnorm(mu, sigma),
      mu <- a + b*Hm,
      a ~ dnorm(0,10),
      b ~ dunif(0,1),
```

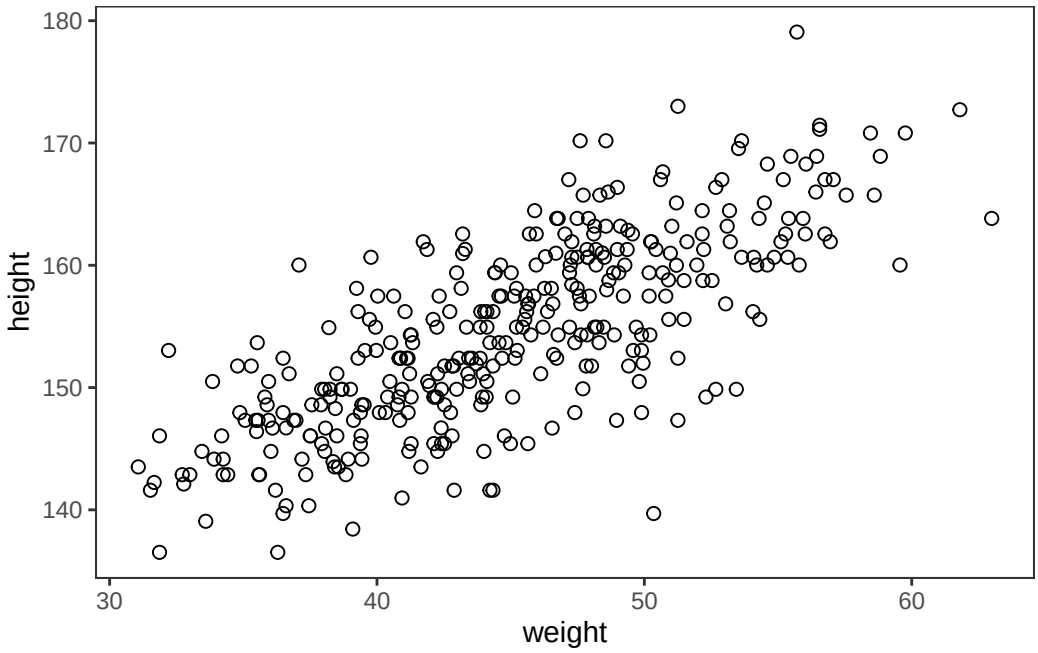
```
      sigma ~ dunif(0, 10)
    ),
    data = list(W=W, H=H)
  )
```

3.4 Linear Prediction: Adding a predictor

We'll now add a predictor to our model: weight. So we'll look at how height covaries with weight (our predictor).

A scatter plot of weight by height (with ggplot):

```
ggplot(data = d2,
       aes(x = weight, y = height)) +
  geom_point(shape = 1, size = 2) +
  theme_bw() +
  theme(panel.grid = element_blank())
```



We now have:

$$h_i \sim Normal(\mu_i, \sigma) \quad likelihood \tag{3.11}$$
$$\mu_i \sim \alpha + \beta(x_i - \bar{x}) \quad linear\ model \tag{3.12}$$
$$\alpha \sim Normal(178, 20) \quad \beta\ prior \tag{3.13}$$
$$\beta \sim Normal(178, 20) \quad \mu\ prior \tag{3.14}$$
$$\sigma \sim Uniform(0, 50) \quad \sigma\ prior \tag{3.15}$$

where the average of x values is $\bar{x}$. Our predictor variablle is x, which is a list of measures of the same length as h. We now define μ as a function of the values in x in order to include our predictor, weight, in the model. α is our intercept, β is our slope.

3.5 Polynomial regression

3.6 Solomon Kurz

```
library(brms)
```

```
Loading required package: Rcpp
```

```
Loading 'brms' package (version 2.20.3). Useful instructions
can be found by typing help('brms'). A more detailed introduction
to the package is available through vignette('brms_overview').
```

```
Attaching package: 'brms'
```

```
The following objects are masked from 'package:rethinking':

    LOO, stancode, WAIC
```

```
The following object is masked from 'package:stats':

    ar
```

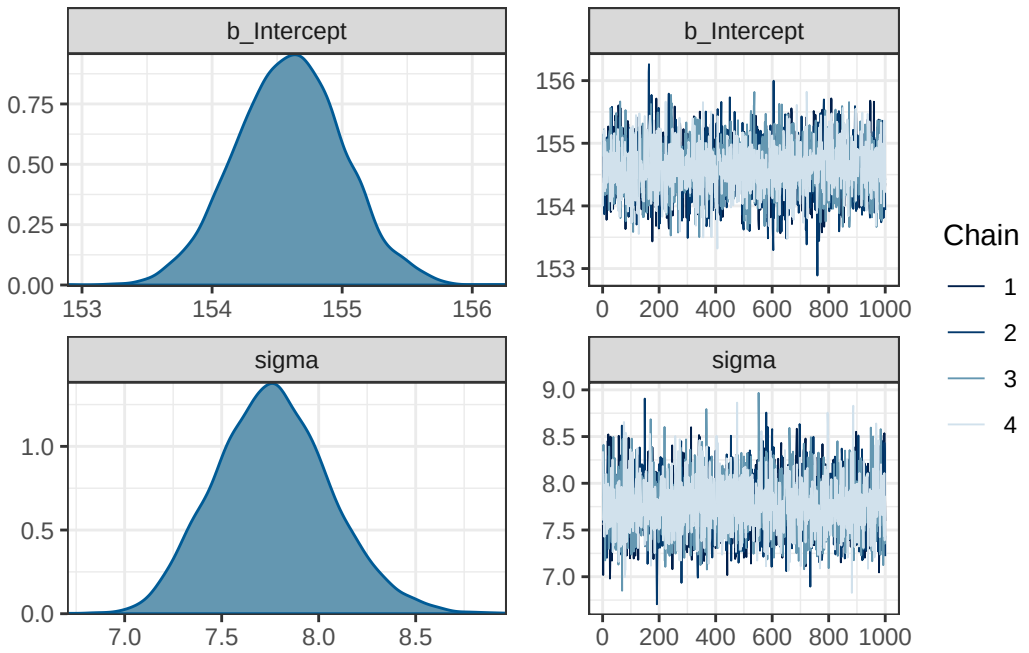
3.6.1 Fitting a model with `brm()`

Instead of `statrethinking::map()`

```
d2 <-
  d %>%
  filter(age >= 18)
```

```
b4.1 <-
  brm(data = d2,
      family = gaussian,
      height ~ 1,
      prior = c(prior(normal(178, 20), class = Intercept),
                prior(uniform(0, 50), class = sigma, ub = 50)),
      iter = 2000, warmup = 1000, chains = 4, cores = 4,
      seed = 4,
      file = here::here("fits/b04.01"))
```

```
plot(b4.1)
```




```
print(b4.1)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: height ~ 1
```

```
   Data: d2 (Number of observations: 352)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   154.60      0.41   153.80   155.43 1.00     2887     2156

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     7.77      0.30     7.22     8.38 1.00     3322     2196

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Or, if we want to use the 89% intervals used in the textbook:

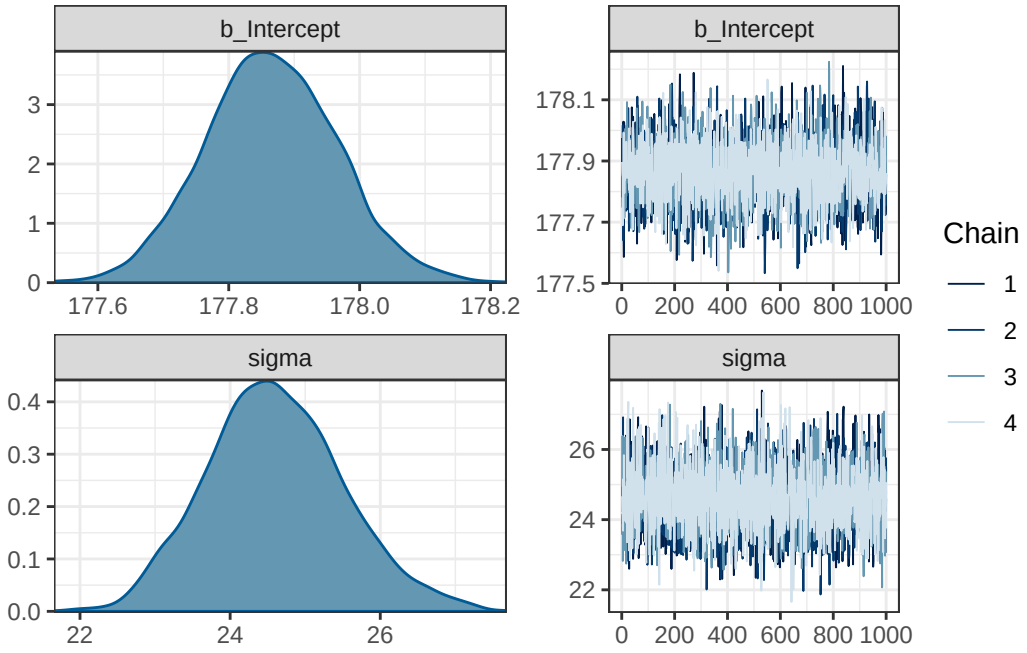
```
summary(b4.1, prob = .89)
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: height ~ 1
   Data: d2 (Number of observations: 352)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
Intercept   154.60      0.41   153.95   155.24 1.00     2887     2156

Family Specific Parameters:
      Estimate Est.Error l-89% CI u-89% CI Rhat Bulk_ESS Tail_ESS
sigma     7.77      0.30     7.31     8.26 1.00     3322     2196

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Now if we want very narrow priors (sd of 0.1!):

```
b4.2 <-
  brm(data = d2, family = gaussian,
      height ~ 1,
      prior = c(prior(normal(178, 0.1), class = Intercept),
                prior(uniform(0, 50), class = sigma, ub = 50)),
      iter = 2000, warmup = 1000, chains = 4, cores = 4,
      seed = 4,
      file = here::here("fits/b04.02"))
```

```
plot(b4.2)
```



We see the chains actually look pretty good. Let's look at the fixed effects to easily compare the estimates of the two models:

```
rbind(
  summary(b4.1)$fixed |>
  as_tibble() |>
  mutate(model = "b4.1"),
summary(b4.2)$fixed |>
  as_tibble() |>
  mutate(model = "b4.2")
)
```

```
# A tibble: 2 x 8
  Estimate Est.Error `l-95% CI` `u-95% CI`  Rhat Bulk_ESS Tail_ESS model
     <dbl>     <dbl>      <dbl>      <dbl> <dbl>    <dbl>    <dbl> <chr>
1     155.     0.411       154.       155.  1.00    2887.    2156. b4.1
2     178.     0.101       178.       178.  1.00    3066.    2437. b4.2
```

3.6.2 Sampleing from a `brm()` fit

Instead of using the `vcov()` function from the `statrethinking` package (`brms` has a `vcov()` function but it only gives the first element in the matrix), we can get it after putting the HMC chains in a data frame (which we do with `as_draws_df()`):

```
post <- as_draws_df(b4.1)

head(post)
```

```
# A draws_df: 6 iterations, 1 chains, and 4 variables
  b_Intercept sigma lprior  lp__
1         155   7.5   -8.5 -1227
2         155   7.0   -8.5 -1230
3         154   7.6   -8.5 -1226
4         154   8.0   -8.5 -1226
5         155   7.6   -8.5 -1226
6         155   7.4   -8.5 -1227
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

```
select(post, b_Intercept:sigma) %>%
  cov()
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
             b_Intercept        sigma
b_Intercept  0.168810743 -0.000740092
sigma       -0.000740092  0.087416426
```

3.7 Practice

E1

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

$$y_i \sim Normal(\mu, \sigma) \tag{3.16}$$
$$\mu \sim Normal(0, 10) \tag{3.17}$$
$$\sigma \sim Exponential(1) \tag{3.18}$$

3.16 is the likelihood (3.17 is the prior for μ and 3.18 for σ)

E2

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

There are two parameters in the posterior distribution: μ and σ

E3

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

Bayes' theorem is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.19}$$

Where A is our priors and B is the posterior, i.e., observed data (y) I'm guessing? And so...

$$P(\mu, \sigma|y) = \frac{P(y|\mu, \sigma)P(\mu)P(\sigma)}{P(y)} \tag{3.20}$$

Okay I really don't know...I guess we add in the model parameter definitions?

E4

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

$$y_i \sim Normal(\mu, \sigma) \tag{3.21}$$
$$\mu = \alpha + \beta x_i \tag{3.22}$$
$$\alpha \sim Normal(0, 10) \tag{3.23}$$
$$\beta \sim Normal(0, 1) \tag{3.24}$$
$$\sigma \sim Exponential(2) \tag{3.25}$$

3.22 is the linear model. is the likelihood, the prior for α (intercept), the prior for β (slope), and the prior for sigma (sd).

E5

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

The posterior distribution has the parameters:

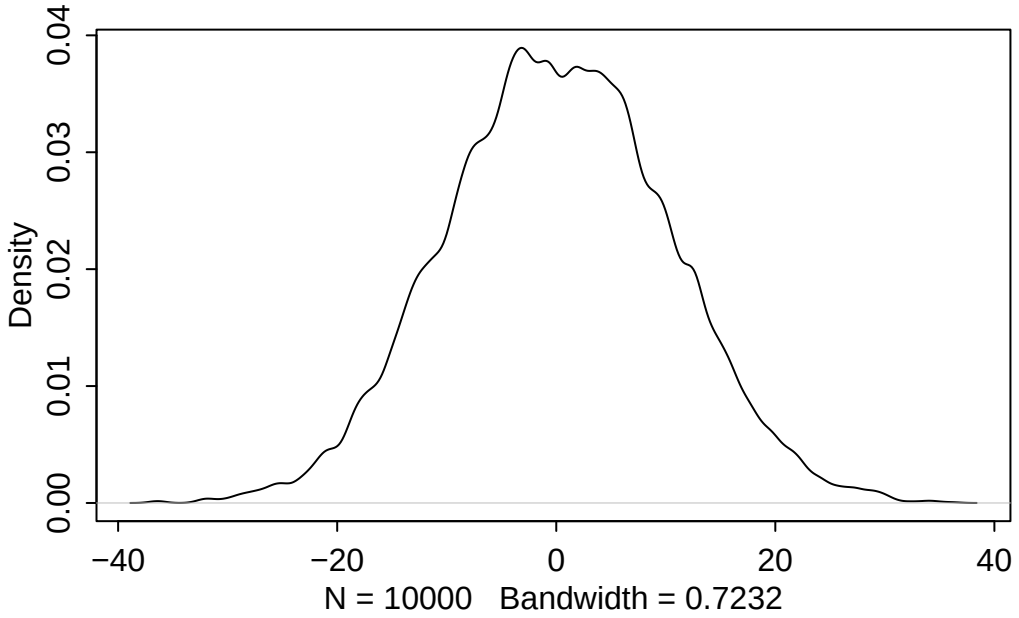
- σ: standard deviation
- α: intercept
- β: slope

M1

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

$$y_i \sim Normal(\mu, \sigma) \tag{3.26}$$
$$\mu \sim Normal(0, 10) \tag{3.27}$$
$$\sigma \sim Exponential(1) \tag{3.28}$$

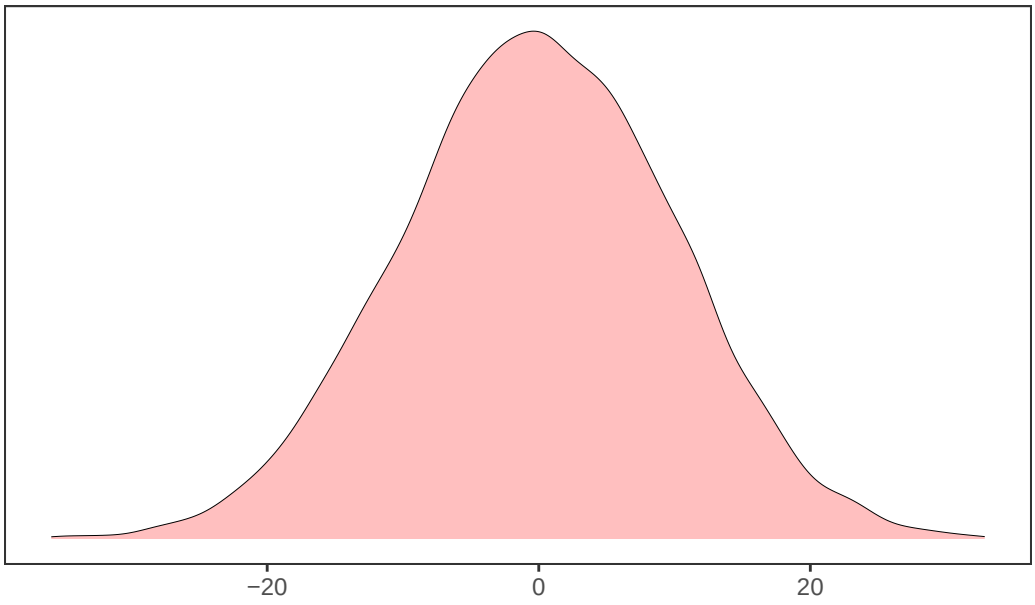
```
# same as R chunk 4.14
sample_mu <- rnorm(1e4, 0, 10)
sample_sigma <- rexp(1e4, 1)
# simulate prior predictive values
prior_h <- rnorm(1e4, sample_mu, sample_sigma)
dens(prior_h)
```



And our prior predictive with ggplot:

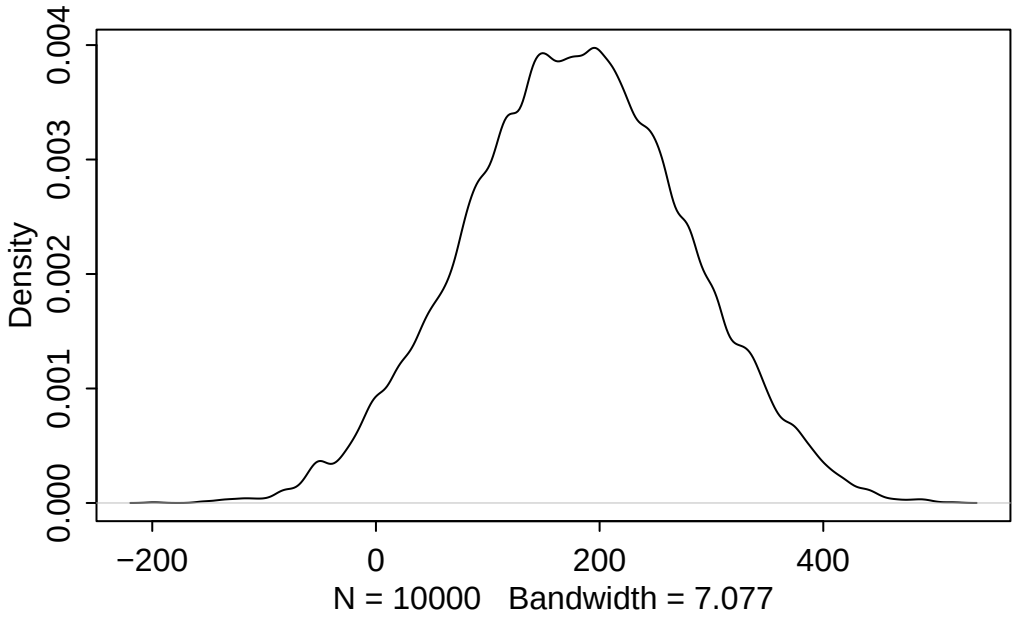
```
n <- 1e4

set.seed(4)
tibble(sample_mu    = rnorm(n, mean = 0,       sd  = 10),
       sample_sigma = rexp(n, 1)) %>%
  mutate(x = rnorm(n, mean = sample_mu, sd  = sample_sigma)) %>%

  ggplot(aes(x = x)) +
  geom_density(fill = "red", linewidth = 0, alpha = .25) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = expression(Prior~predictive~distribution~"for"~italic(h[i])),
       x = NULL) +
  theme(panel.grid = element_blank())
```

Prior predictive distribution for h_i



3.7.1 Sampling from the posterior

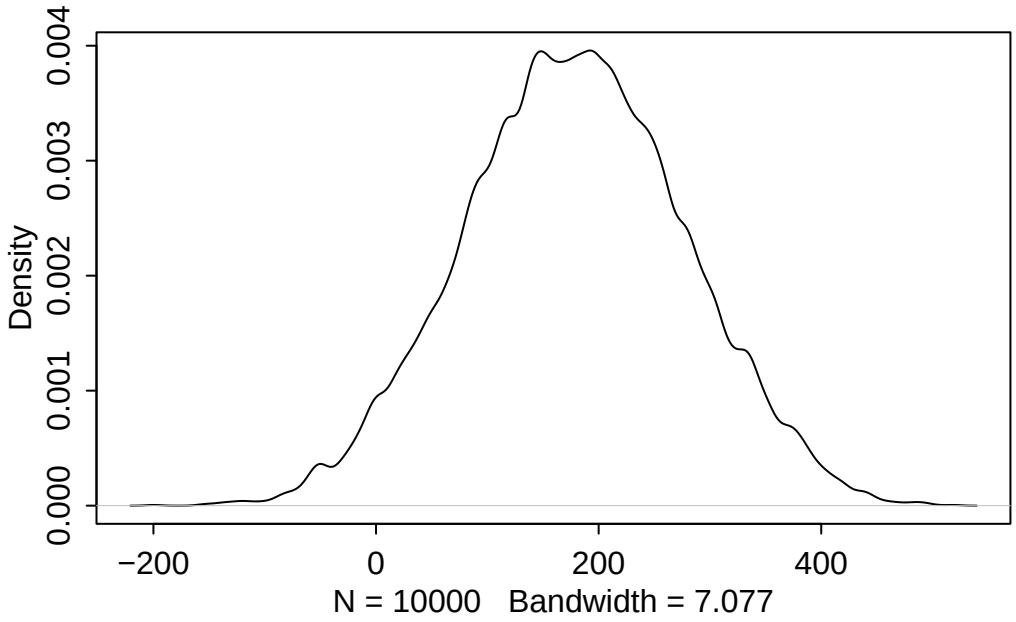
```
# R chunk 4.15
sample_mu <- rnorm( 1e4 , 178 , 100 )
prior_h <- rnorm( 1e4 , sample_mu , sample_sigma )
dens( prior_h )
```



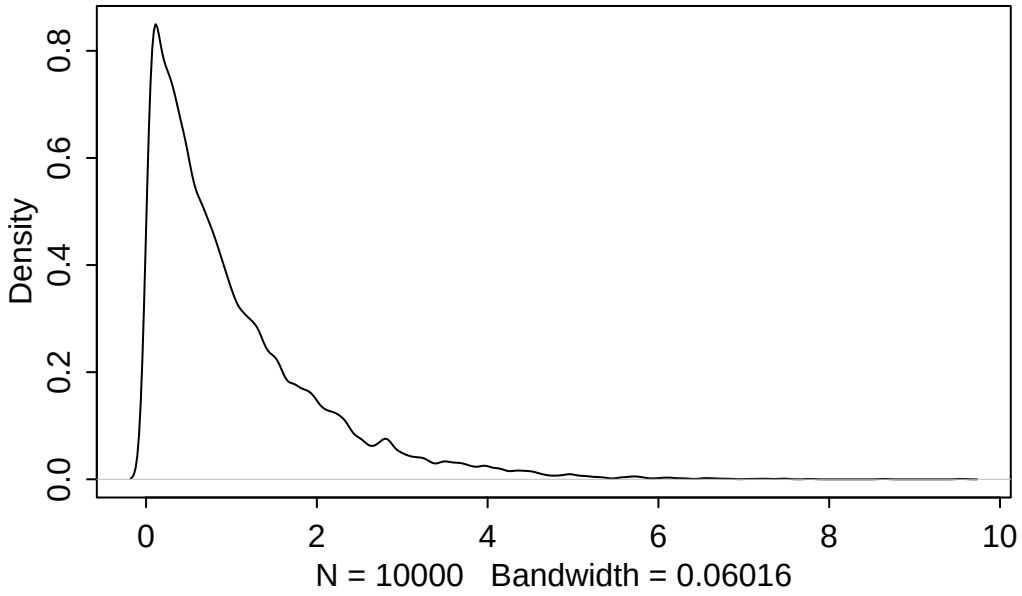

```
# R chunk 4.16
mu.list <- seq( from=150, to=160 , length.out=100 )
sigma.list <- seq( from=7 , to=9 , length.out=100 )
post <- expand.grid( mu=mu.list , sigma=sigma.list )
post$LL <- sapply( 1:nrow(post) , function(i)
  sum( dnorm( d2$height , post$mu[i] , post$sigma[i] , log=TRUE ) ) )

post$prod <- post$LL + dnorm( post$mu , 178 , 20 , TRUE ) + dunif( post$sigma , 0 , 50 , T
post$prob <- exp( post$prod - max(post$prod) )
```

```
# R chunk 4.17
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 ]
```

```
# R chunk 4.21
dens(sample_mu)
```




```
# R chunk 4.21
dens(sample_sigma)
```



M2

Translate the model just above into a quap formula.

Do this as in R chunk 4.26:

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

References

Kurz, A. Solomon. 2023. *Statistical Rethinking with brms, ggplot2, and the tidyverse*. Version 1.3.0. https://bookdown.org/content/3890/.

McElreath, Richard. 2020. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Second edition. Chapman & Hall/CRC Texts in Statistical Science Series. Boca Raton: CRC Press.