Mulling Over McElreath's Statistical Rethinking

Dan Burrell

2020-08-22

Contents

Chapter 1

Before you start, you should do these things

We're working through the second edition of McElreath's Statistical Rethinking text. We need to access the associated R pacakge `rethinking` and also load the `tidyverse`. Use the following code:

```
if(!require(pacman)) install.packages("pacman")
library(pacman)

#install.packages("rstan", repos = "https://cloud.r-project.org/", dependencies = TRUE)
#pkgbuild::has_build_tools(debug = TRUE) # Check the C++ toolchain
#library("rstan")
#options(mc.cores = parallel::detectCores())
#rstan_options(auto_write = TRUE)
p_load(coda, mvtnorm, loo, dagitty, tidyverse, remotes)
remotes::install_github("rmcelreath/rethinking")
#remotes::install_github("stan-dev/cmdstanr")
```

Below is an example of the generic format for STANcode. The blocks need to occur in the order specified, however only the `model` block is necessary (the others are optional, depending on the needs of the modeler).

```
modelString = "

    data {
        ...declarations... // This is a comment
    }

    transformed data {
        ...declarations ... statements ...
```

```
    }

    parameters {
        ...declarations...
    }

    transformed parameters {
        ... declarations ... statemetns ...
    }

    model {
        ...declarations ... statements ...
    }

    generated quantities {
        ...declarations ... statements ...
    }
" # Close quote for modelString
```

Translate the model into C++ code and compile into an executable dynamic shared object (DSO) using `stan_model()` from the `rstan` package:

```
p_load(rstan)
stanDSO = stan_model( model_code=modelString )
```

Once the DSO is created, it can be used for generating a Monte Carlo sample from the posterior distribution. For example:

```
# Create some fictitious data:
N = 50; z = 10; y = c(rep(1,z), rep(0,N-z))
dataList = list( y=y, N=N)
stanFit = sampling( object=stanDSO, data=dataList,
                    chains=3, iter=1000, warmup=200, thin=1)

# Load rjags, coda, and DBDA2E functions
source("DBDA2E-utilities.R")
```

Chapter 2

The Golem of Prague

Statistical models are like powerful yet dangerous robots. They will do their task, but they have no discernment as to whether their task is appropriate. For their particular task, they are good. But set to work on a task they're not intended for, they yield untrustworthy results.

Classical statistical methods are fragile and inflexible. They are inflexible in that they are exhibit very limited adaptability to unique research contexts. They are fragile in that they fail in unpredictable ways when applied to new contexts. This is important because at the cutting edge of research it's rarely ever clear which procedure is appropriate.

McElreath's point is that classical statistical tools are not diverse enough to handle many common research problems. Moreover, statistical tools on their own only understand association, and can tell us nothing about cause and effect. So, rather than having a tool kit of statistical "robots" and risk their misapplication, what is needed is a unified set of engineering principles for designing, constructing and refining purpose-built statistical procedures.

Some thoughts on this:

- You need to understand how a statistical procedure processes information in order to be able to reasonably interpret its output;
- detailed knowledge is a requirement;
- getting greasy under the hood is how you cultivate this sort of deep understanding — doing the computations the hard way (at least initially);
- we also need a statistical epistemology — an appreciation of how statistical models relate to hypotheses and the natural mechanisms of interest.
- we need to do away with the null hypothesis significance testing mindset that is born from unscrupulous application of Popper's falsificationism; good science is done by developing statistical procedures that can falsify hypotheses, but this ought to be done thoughtfully and with adequate

insight into the limitations of deductive falsification.

McElreath argues that deductive falsification is impossible in nearly every scientific context because:

1. Hypotheses are not models. There is not a one-to-one correspondence between hypotheses and models, so that strict falsification is impossible.
2. Measurement matters. Even when we think our data do falsify a model, the trustworthiness (or representativeness) of the data is always up for debate.

The upshot is that the scientific method cannot be reduced to a statistical procedure; and hence our scientific methods ought not pretend. That's the arrow shot at the heart of null hypothesis significance testing (NHST), which is often identified with falsificationist, or Popperian, philosophy of science. NHST is used to falsify a null hypothesis, not the actual research hypothesis, so the "falsification" doesn't even truly pertain to the explanatory model; a kind of reversal of Popper's philosophy.

2.1 Hypotheses are not models

We may wish to test a hypothesis but we actually substitute the hypothesis with a surrogate: a model that attempts to make operational, insofar as is possible, the essence of the hypothesis. Models are not truth, they merely attempt to capture aspects of nature so as to reflect back useful information. If models are false (albeit perhaps useful), what does it even mean to falsify a model? By working with models we are in no position to conclude anything about the truth or falsity of the hypotheses underlying them; we can only say that a model supports the hypothesis or it doesn't.

A model of the process that moves one from a hypothesis to a statistical model capable of being confronted with data is as follows:

1. Hypothesis: characterized by vague boundaries that begin as verbal conjectures
2. Process model: making choices about competing processes that fall within the vague boundaries of the verbal conjecture. Importantly, process models express causal structure.
3. Statistical model: to challenge a process model with data it needs to be translated into statistical models, and a side-effect of this is that statistical models do not embody specific causal relationships, only associations among variables.This translation from process to statistical model is many-to-one: many different process models may be consistent with a single statistical model. Moreover, statistical models can be confused by unobserved variables (mediators) and sampling bias. Process models allow us to design statistical models with these problems in mind - we need both process and statistical model.

Uncomfortable lesson:

1. Any given statistical model (M) may correspond to more than one process model (P).
2. Any given hypothesis (H) may correspond to more than one process model (P).
3. Any given statistical model (M) may correspond to more than one hypothesis (H).

Many common distributions — exponential family distributions — are maximum entropy distributions.

2.2 Measurement matters

The logic of falsificationism is simple: we have a hypothesis H, and we show that it entails some observation D. Then we look for D. If we don't find it, we must conclude that H is false — logically this is *modus tollens* (the method of destruction). In contrast, find D tells us nothing about H, because other hypotheses might also predict D.

Seeking disconfirming evidence is important, but it suffers other problems in addition to the correspondence problems among hypotheses and models discussed earlier. In particular, observations are prone to error, and most hypotheses live on a continuum of degree, rather than on a discrete dichotomous space of absolutes.

2.2.1 Observaqtion error

Especially at the edges of scientific knowledge, the ability to measure a hypothetical phenomenon is often in question as much as the phenomenon itself. Finding disconfirming cases is complicated by the difficulties of observation: false positives (mistaken confirmations) and false negatives (mistaken disconfirmations). This induces a key dilemma: whethr or not a falsification is real or spurious. Measurement is complicated in both cases, but in quite different ways, rendering both true-detection and false-detection plausible.

2.2.2 Continuous hypotheses

Most interesting hypotheses are not of an absolute nature, but concern degrees: not "all swans are white", but "80% of swans are white", or "balck swans are rare". The task is not to prove or disprove but to estimate and explain the distribution of swan coloration as accurately as we can. This problem prevents strict application of *modus tollens*. Nearly everyone agrees that it is a good practice to design experiments and observations that can differentiate competing hypotheses, but in many cases, the comparison must be probabilistic, a matter of degree, not kind.

2.3 Tools for golem engineering

We want to use our models for several distinct purposes: designing inquiry, extracting information from data, and making predictions. Tools for this are:

1. Bayesian data analysis: essentially counting the number of ways the data could happen, according to our assumptions; things that can happen more ways are more plausible.
2. Model comparison: based on expected predictive accuracy (cross-validation and information criteria); overfitting issue
3. Multilevel models: It's parameters all the way down: any particular parameter can be usefully regarded as a placeholder for a missing model; given some model of how the parameter gets its value, it's simple enough to embed the new model inside the old one — resulting in a model with multiple levels of uncertainty that trickle down the levels (also called hierarchical; random effects; varying effects; mixed effects models); incidentally, while corss-validation and information criteria measure overfitting risk, multilevel models help do something about it via partial pooling of information across the data in order to produce better estimates for all units. Multilevel regression deserves to be the default form of regression.
4. Graphical causal models: A statistical model is an amazing association engine, but is insufficient for inferring cause. Models that are causally incorrect can make better predictions than those that are causally correct. Hence, focussing on prediction can systematically mislead us. We need a causal model that can be used to design one or more statistical models for the purpose of causal identification. Graphical causal models allow us to deduce which statistical models can provide valid causal inferences, assuming the DAG is true.

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter **??**. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package

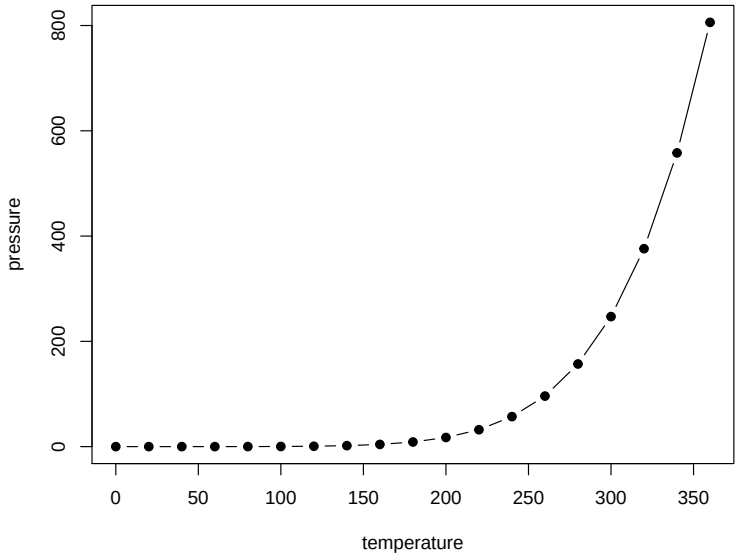


Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

(**?**) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 3

Small Worlds and Large Worlds

Differentiate between the small world — the self-contained logical world of the model — and the large world — the broader context in which one deploys a model. Statistical models pertain to the small world, and insofar as their assumptions reflect reality, they are useful in the large world. Bayesian learning is optimal in the small world in terms of how it makes use of information in the data, but that optimality only transfers to the large world inasmuch as the small world is an accurate description of the large world.

The small world is about probability theory and the stylized components of a Bayesian statistical model for learning from data.

We collect a sequence of observations — data. Building a model involves making assumptions:

1. Data story: how the data arise
2. Update: educate your model by feeding it data
3. Evaluate: supervise, critique and often rebuild.

We need to name all the variables and define them. Variables are symbols that can take on different values. They can be unobserved variables (i.e. parameters) or observed variables (things we can measure and use to infer the unobserved variable values).

3.1 End of chapter exercises

3.1.1 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?

Solution

```
#2H1
prior = c(0.5, 0.5)
likelihood = c(0.1, 0.2)
posterior_unst = likelihood * prior
posterior = posterior_unst / sum(posterior_unst)
# probability next birth is twins:
posterior[1] * 0.1 + posterior[2] * 0.2
```

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

3.1.2 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.

Solution

```
# 2H2
prior1 = c(0.5, 0.5) # A, B
likelihood1 = c(0.1, 0.2) #A, B
posterior_unst1 = likelihood1 * prior1
posterior1 = posterior_unst / sum(posterior_unst)
# probability she is species A
posterior1[1]
```

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

```
posterior1
```

```
## [1] 0.3333333 0.6666667
```

3.1.3 2H3.

Continuing on from the previous problem, suppose the same panda mother has a second birth and that it is not twins, but a singleton infant. Compute the posterior probability that this panda is species A. **Solution** Want: $P(\neg t2|t1, A)$.

```
# 2H3
prior2 = posterior1
likelihood2 = 1 - likelihood1
posterior_unst2 = prior2 * likelihood2
posterior2 = posterior_unst2 / sum(posterior_unst2)
posterior2
```

```
## [1] 0.36 0.64
```

```
# Prob of singleton given species A and twins
posterior2[1]
```

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

```
pr = c(0.5, 0.5)
lik = c(0.1*0.9, 0.2*0.8)
po = pr * lik
po = po/sum(po)
po
```

```
## [1] 0.36 0.64
```

3.1.4 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.

Solution Know: $P(testA|A) = 0.8$ and $P(testB|B) = 0.65$, and can infer $P(testA|B) = 1 - 0.65 = 0.35$ and $P(testB|A) = 1 - 0.8 = 0.2$.

Want: $P(A|testA) = \frac{P(testA|A)P(A)}{P(testA)}$

$$\begin{aligned} P(testA) &= P(testA|A)P(A) + P(testA|B)P(B) \\ &= 0.8 \times 0.5 + 0.35 \times 0.5 \\ &= 0.575 \end{aligned}$$

```
# 2H4
# No birth information
prior3 = c(0.5, 0.5)
likelihood3 = c(0.8, 0.35)
posterior_unst3 = prior3 * likelihood3
posterior3 = posterior_unst3 / sum(posterior_unst3)
posterior3
```

```
## [1] 0.6956522 0.3043478
```

```
# Birth information
prior4 = posterior2
likelihood4 = c(0.8, 0.35)
posterior_unst4 = prior4 * likelihood4
posterior4 = posterior_unst4 / sum(posterior_unst4)
posterior4
```

```
## [1] 0.5625 0.4375
```

Chapter 4

Sampling the imaginary

There is a blood test that correctly detects vampirism 95% of the time and incorrectly diagnoses normal people as vampires 1% of the time. Letting v denote being a vampire and h denote being human (not vampire) and $+$ denote testing positive for vampirism, and $-$ denote testing negative for vampirism, then we have:

$$\begin{aligned}\mathbb{P}(+|v) &= 0.95\\ \mathbb{P}(-|v) &= 0.05\\ \mathbb{P}(+|h) &= 0.01\\ \mathbb{P}(-|h) &= 0.99\end{aligned}$$

Suppose that vampires are quite rare, comprising only 0.1% of the population. That is $\mathbb{P}(v) = 0.001$.

Now suppose someone tests positive for vampirism. What's the actual probability that he/she is really a vampire?

Using Bayes' theorem this is simple. We want $\mathbb{P}(v|+)$ and we can find it through probability inversion as follows:

$$\begin{aligned}\mathbb{P}(v|+) &= \frac{\mathbb{P}(+|v)\mathbb{P}(v)}{P(+)}\\ &= \frac{\mathbb{P}(+|v)\mathbb{P}(v)}{\mathbb{P}(+|v)\mathbb{P}(v) + \mathbb{P}(+|h)\mathbb{P}(h)}\\ &= \frac{0.95 \times 0.001}{0.95 \times 0.001 + 0.01 \times 0.999}\\ &= 0.08683729\end{aligned}$$

The calculations can be carried out easily in R:

```
p_pos_v = 0.95
p_pos_h = 0.01
p_neg_v = 1 - p_pos_v
p_neg_h = 1 - p_pos_h
p_v = 0.001
p_h = 1 - p_v

p_pos = p_pos_v * p_v + p_pos_h * p_h
p_v_pos = p_pos_v * p_v / p_pos

print(p_v_pos)
```

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

There is (only) an 8.7% that the tested person is actually a vampire. Whenever the condition of interest is very rare, having a test that finds all the true cases is still no guarantee that a positive result carries much information at all. Why? Because most positive tests are false positives, even when all the true positives are detected correctly.

The above results seems counter-intuitive. But there's a way to frame it — in terms of frequencies — that makes it far easier. Suppose we have 100000 people in the population and 100 of them are vampires. Of the 100 vampires, 95 of them would test positive for vampirism. Of the 99900 humans, 999 of them would test positive for vampirism.

Now, suppose we test the whole population, what proportion of those who test positive do you expect to actually be vampires? It's simple now right:

$$\mathbb{P}(v|+) = \frac{95}{1094} \approx 0.087$$

Using frequencies makes things easier — and we exploit this fact when we sample from a posterior distribution.

4.1 Sampling from a grid-approximate posterior

A globe-tossing model: You have a globe representing Earth and you're curious about how much of the surface is covered in water. You will toss the globe in the air and catch it, and record whether or not the surface under your right index finger is water or land. Repeat the procedure some number of times and then look at the relative proportions of water to land. The first 9 observations are

$$W\ L\ W\ W\ W\ L\ W\ L\ W$$

Before we can work with samples, we need to generate them.

The data story is simply that the true proportion of water covering the globe is p, a single toss of the globe has probability p of producing water and probability $1-p$ of producing land, and each toss of the globe is independent of the others.

To translate the data story into a probability model: the counts of "water" and "land" are distributed binomially, with probability p of water on each toss:

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

A Bayesian model specifies a likelihood:

$$W \sim \text{Binomial}(N, p); \quad N = W + L$$

and a prior:

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

To actually process the data and model, to produce a posterior, we need an engine. Most real-world models don't allow us to find analytical (pen and paper) solutions. Instead for small models we can use grid approximation, for moderately larger models we canuse quadratic approximation, and for proper real-world models we can use MCMC.

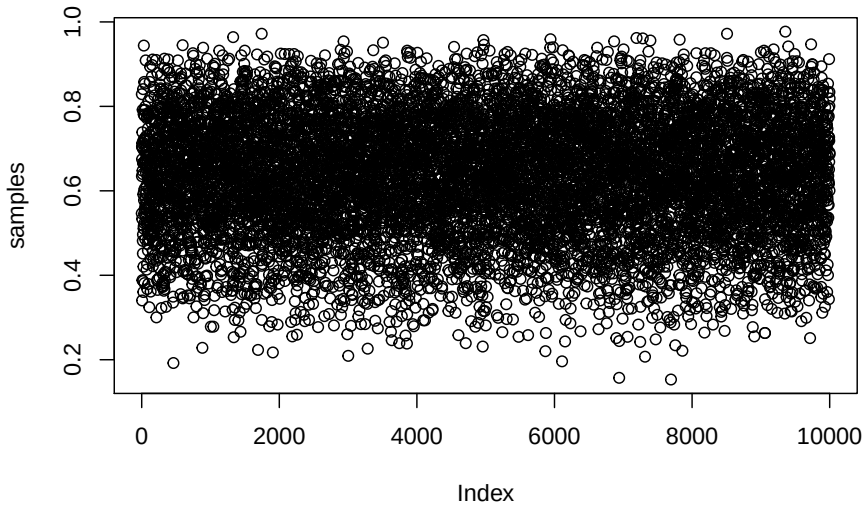
```
if(!require(pacman)) install.packages(pacman)
```

```
## Loading required package: pacman
```

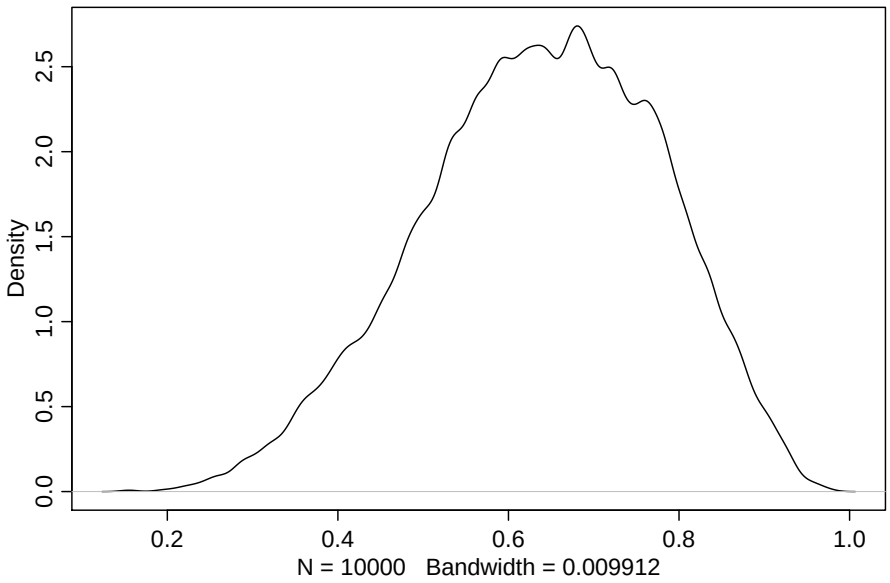
```
library(pacman)
```

```
n_grid = 1000
grid_limits = c(0,1)
p_grid = seq( from=grid_limits[1], to=grid_limits[2], length.out=n_grid)
prob_p = rep( 1, 1000)
prob_data = dbinom( 6, size=9, prob=p_grid )
posterior = prob_data * prob_p
posterior = posterior / sum(posterior)
```

```
pacman::p_load(rethinking)

samples = sample( p_grid, prob=posterior, size=1e4, replace=TRUE )
plot( samples )
```




```
dens( samples )
```



Sampling to summarize

4.1.1 Intervals of defined boundaries

To get the proportion of water less than some value of `p_grid`:

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

```
## [1] 0.1706
```

```
sum( samples > 0.5 & samples < 0.75 ) / 1e4
```

```
## [1] 0.5998
```

```
quantile( samples, 0.8)
```

```
##       80%
## 0.7627628
```

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

```
##       10%       90%
## 0.4494494 0.8128128
```

```
n_grid = 1000
grid_limits = c(0,1)
p_grid = seq( from=grid_limits[1], to=grid_limits[2], length.out=n_grid)
prior = rep( 1, n_grid)
likelihood = dbinom( 3, size=3, prob=p_grid )
posterior = likelihood * prior
posterior = posterior / sum(posterior)
samples = sample( p_grid, size=1e4, replace=TRUE, prob=posterior )
```

```
PI( samples, prob=0.5 )
```

```
##       25%       75%
## 0.7027027 0.9319319
```

```
HPDI( samples, prob=0.5 )
```

```
##      |0.5      0.5|
## 0.8428428 1.0000000
```

```
p_grid[ which.max(posterior) ]
```

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

```
chainmode( samples, adj=0.01 )
```

```
## [1] 0.9972145
```

```
mean( samples )
```

```
## [1] 0.8001998
```

```
median( samples )
```

```
## [1] 0.8428428
```

```
sum( posterior * abs( 0.5 - p_grid ) )
```

```
## [1] 0.3128752
```

```
pacman::p_load(purrr)

loss = map_dbl(
    p_grid,
    function(d){ sum( posterior * abs( d - p_grid ) ) } )

p_grid[ which.min(loss) ]
```

```
## [1] 0.8408408
```

Generating dummy data: suppose we have two tosses and the probability of water is 0.7: $(N = 2, p = 0.7)$. To compute the probability of the possible outcomes:

```
N = 9
p = 0.7
w = 0:N
probs = dbinom( w, size=N, prob=p)
```

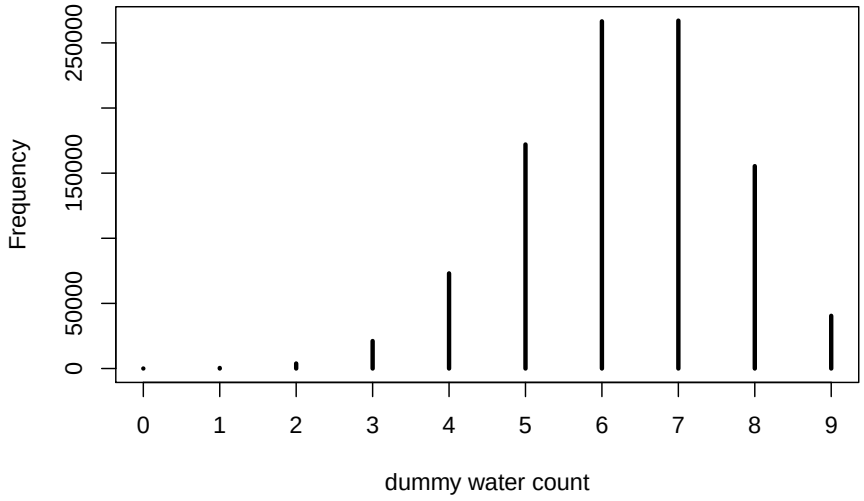
This tells us that there is a 9% chance that $w = 0$, a 42% chance that $w = 1$ and a 49% chance that $w = 2$.

Let's simulate observations now using these probabilities:

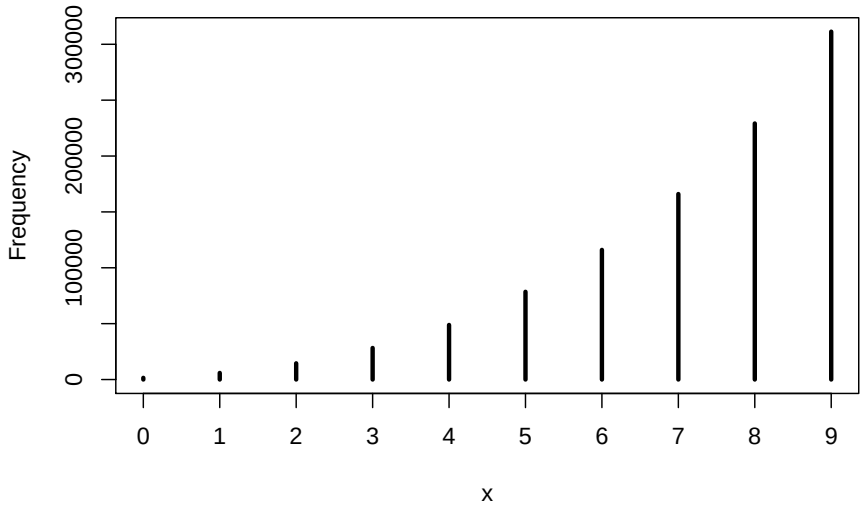
```
n_samples = 1e6
dummy_w = rbinom(n_samples, size=N, prob=p )
table(dummy_w)/n_samples
```

```
## dummy_w
##        0        1        2        3        4        5        6        7
## 0.000018 0.000431 0.003869 0.021115 0.073042 0.172010 0.266576 0.267091
##        8        9
## 0.155379 0.040469
```

```
simplehist( dummy_w, xlab="dummy water count" )
```



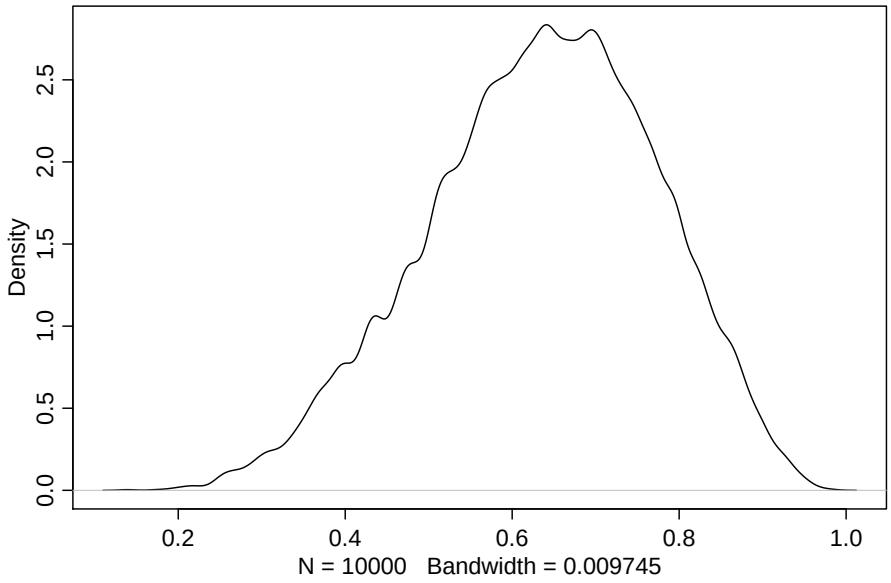

```
w = rbinom( n_samples, size=N, prob=samples)
simplehist(w)
```



4.2 Chapter 3 Practice Problems

The easy problems all use the following code:

```
require(rethinking)
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)
dens(samples)
```



Use the values in `samples` to answer the following questions:

4.2.1 3E1.

How much posterior probability lies below p = 0.2?

Solution This is just asking for the sum of probabilities < 0.2:

```
# Using samples from posterior
sp = 100 * sum( samples < 0.2 ) / 1e4

# cf. grid-approximate posterior
gp = 100 * sum( posterior[ p_grid < 0.2 ])
```

```
paste("The samples give ", round(sp,4), "% less than 0.2, while the grid-approximate posterior in
```

```
## [1] "The samples give  0.04 % less than 0.2, while the grid-approximate posterior indicates  0
```

4.2.2 3E2.

How much posterior probability lies above p = 0.8?

Solution This is just asking for the sum of probabilities on > 0.8

```
# Using samples from posterior
sp = 100 * sum( samples > 0.8 ) / 1e4

# cf. grid-approximate posterior
gp = 100 * sum( posterior[ p_grid > 0.8 ])

paste("The samples give ", round(sp,4), "% greater than 0.8, while the grid-approximate posterior
```

```
## [1] "The samples give  11.16 % greater than 0.8, while the grid-approximate posterior indicate
```

4.2.3 3E3.

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

Solution This is just asking for the sum of probabilities on > 0.2 & < 0.8

```
# samples
sp = 100 * sum( samples > 0.2 & samples < 0.8) / 1e4

# grid-approximate
gp = 100 * sum( posterior[ p_grid > 0.2 & p_grid < 0.8 ])

paste("The samples give", round(sp,4), "% in (0.2, 0.8), while the grid-approximate posterior ind
```

```
## [1] "The samples give 88.8 % in (0.2, 0.8), while the grid-approximate posterior indicates 87.
```

4.2.4 3E4.

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

Solution This is just asking for the 20th percentile:

```
q_20 = quantile( samples, 0.2, lower.tail=TRUE)

paste("The 20th percentile is", round(q_20, 2), "(to 2 dp)")
```

```
## [1] "The 20th percentile is 0.52 (to 2 dp)"
```

4.2.5 3E5.

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

Solution This is just asking for the (100 - 20)th = 80th percentile:

```
# from lower tail
q_80 = quantile( samples, 0.8, lower.tail=TRUE )

paste("The 80th percentile is", round(q_80, 2), "(to 2 dp)")
```

```
## [1] "The 80th percentile is 0.76 (to 2 dp)"
```

4.2.6 3E6.

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

Solution This is just asking for the 66% HPDI:

```
require(rethinking)
HPDI_66 = round( HPDI( samples, prob=0.66 ), 2)
HPDI_66
```

```
## |0.66 0.66|
##  0.51  0.77
```

```
paste("The narrowest interval containing 66% of the probabilitiy mass is the 66% HPDI:
```

```
## [1] "The narrowest interval containing 66% of the probabilitiy mass is the 66% HPDI
```

4.2.7 3E7.

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

Solution This is just asking for the 66% PI:

```
require(rethinking)
PI_66 = round( PI( samples, prob=0.66 ), 2)
PI_66
```

```
##  17%  83%
## 0.50 0.77
```

```
paste("The narrowest interval containing 66% of the probabilitiy mass is the 66% PI: "
```

```
## [1] "The narrowest interval containing 66% of the probabilitiy mass is the 66% PI:
```

4.2.8 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.

Solution First write a function so we can input any number of waters in any number of tosses — a bit more generic:

```
globe_water_distn <- function(n_trials, n_water, n_grid=1e3, n_samples=1e4, seed=100, prior=rep(1
    set.seed(seed)
    p_grid = seq( from=0, to=1, length.out=n_grid )
    prior = prior
    likelihood = dbinom( n_water, size=n_trials, prob=p_grid )
    lp = likelihood * prior
    posterior = lp/ sum(lp)

    out = list(p_grid = p_grid, prior = prior, likelihood = likelihood, posterior = posterior)
    return(out)
}
```

Now try it with `n_trials = 15` and `n_water = 8`:

```
out = globe_water_distn(15, 8)
```

4.2.9 3M2.

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

Solution

```
require(rethinking)
n_samples = 1e5
samples = sample( out[["p_grid"]],
                  size=n_samples,
                  replace=TRUE,
                  prob=out[["posterior"]])

hpdi90_1 = HPDI( samples, prob=0.9)
hpdi90_1
```

```
##      |0.9      0.9|
## 0.3313313 0.7167167
```

4.2.10 3M3.

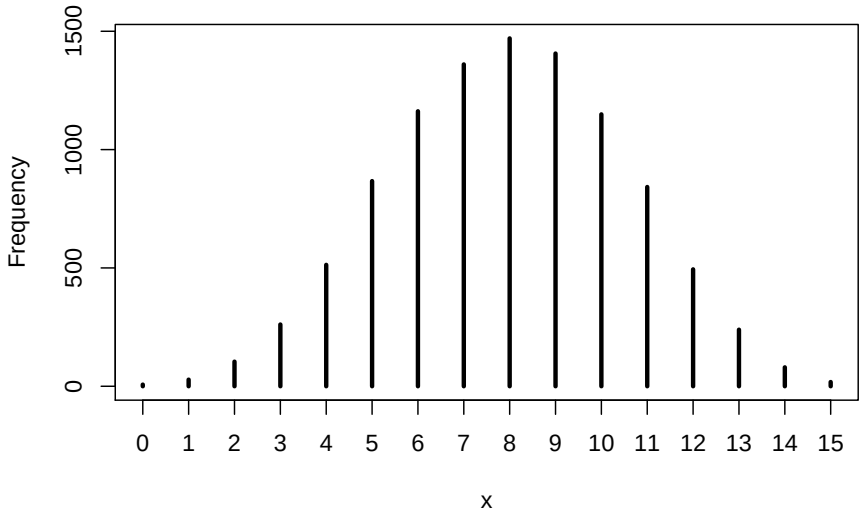
Construct a posterior predictive check for this model and data. This means simulate the distribution of samples, averaging over the posterior uncertainty in

p. What is the probability of observing 8 water in 15 tosses?

Solution

Propagate parameter uncertainty into predictions by using `samples` as the probability weightings:

```
w = rbinom( 1e4, size=15, prob=samples)
simplehist(w)
```




```
prob_8_15 = sum(w == 8)/length(w)
round(100*prob_8_15, 4)
```

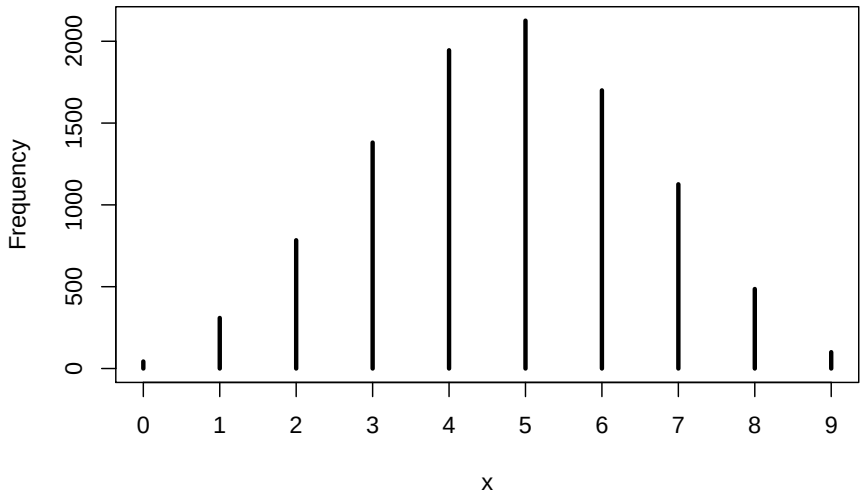
```
## [1] 14.7
```

4.2.11 3M4.

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

Solution Use the posterior from 3M1 as the prior for this run, with `n_trials = 9` and `n_water = 6`:

```
w = rbinom( 1e4, size=9, prob=samples)
simplehist(w)
```




```
prob_6_9 = sum(w == 6)/length(w)
round(100*prob_6_9, 4)
```

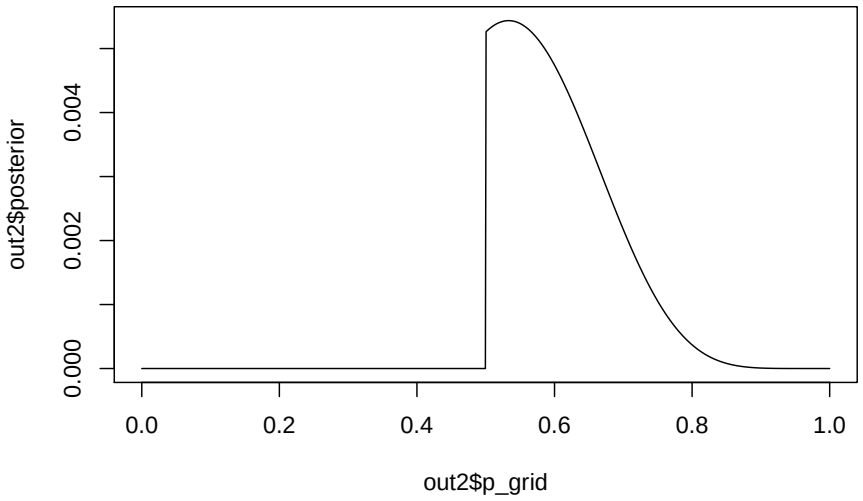
```
## [1] 17
```

4.2.12 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.

Solution Simply pass a prior of the form `ifelse( p_grid < 0.5, 0, 1)` instead of the uniform prior:

```
out2 = globe_water_distn(15, 8, prior = ifelse(p_grid < 0.5, 0, 1))

n_samples = 1e5
samples = sample( out2[["p_grid"]],
                  size=n_samples,
                  replace=TRUE,
                  prob=out2[["posterior"]])


plot( out2$posterior ~ out2$p_grid, type = "l")
```




```
hpdi90_2 = HPDI( samples, prob = 0.90)
hpdi90_2
```

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

```
print("The uniform prior gave a 90% HPDI of:")
```

```
## [1] "The uniform prior gave a 90% HPDI of:"
```

```
print(round(hpdi90_1, 2))
```

```
## |0.9 0.9|
## 0.33 0.72
```

```
print("The informed prior gave a 90% HPDI of:")
```

```
## [1] "The informed prior gave a 90% HPDI of:"
```

```
print(round(hpdi90_2, 2))
```

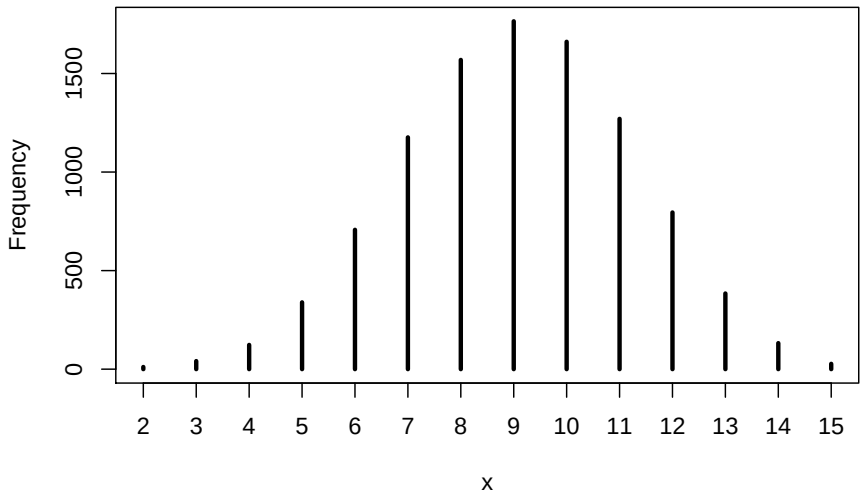
```
## |0.9 0.9|
## 0.50 0.71
```

```
print("The informed prior leads to a narrower HPDI indicating that the informed prior l
```

```
## [1] "The informed prior leads to a narrower HPDI indicating that the informed prior
```

Now the posterior predictive distribution:

```
w = rbinom( 1e4, size=15, prob=samples)
simplehist(w)
```




```
prob_8_15 = sum(w == 8)/length(w)
round(100*prob_8_15, 4)
```

```
## [1] 15.69
```

Now 8/15 is not exactly in the center of the posterior predictive distribution. The informative prior tells the model not to completely trust the data, so the simulated posterior sampling distributions are not quite centered on the data.

4.2.13 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?

Assuming the uniform prior Borrowed from Brian Callander and done using tidyverse functions:

Bayesian models are generative, so we can simulate new datasets according to our prior probabilities. We'll simulate 100 datasets for each value of N of interest. We simulate a dataset by choosing a `p_true` from our prior, then randomly choosing a `w` from the corresponding binomial distribution.

```
pacman::p_load(tidyverse)
m6_prior_predictive = crossing(
    N = 200 * (1:16),
    iter = 1:10
) %>%
    mutate(
        p_true = runif(n(), min=0, max=1),
        W = rbinom(n(), N, p_true)
    )
```

For each of these simulated datasets, we grid approximate the posterior, take posterior samples, then calculate the HPDI

```
granularity = 1000
m6_grid = tibble(p = seq(0, 1, length.out = granularity)) %>%
    mutate(prior = 1)

m6_posteriors = m6_prior_predictive %>%
    crossing(m6_grid) %>%
    group_by(N, p_true, iter) %>%
    mutate(
        likelihood = dbinom(W, N, p),
        posterior = prior * likelihood
    )

m6_samples = m6_posteriors %>%
    sample_n(1000, replace=TRUE, weight=posterior)

m6_hpdi = m6_samples %>%
    summarise(lo = HPDI(p, 0.99)[1], hi = HPDI(p, 0.99)[2]) %>%
    mutate(width = abs(hi - lo))
```

```
## `summarise()` regrouping output by 'N', 'p_true' (override with `.groups` argument)
```

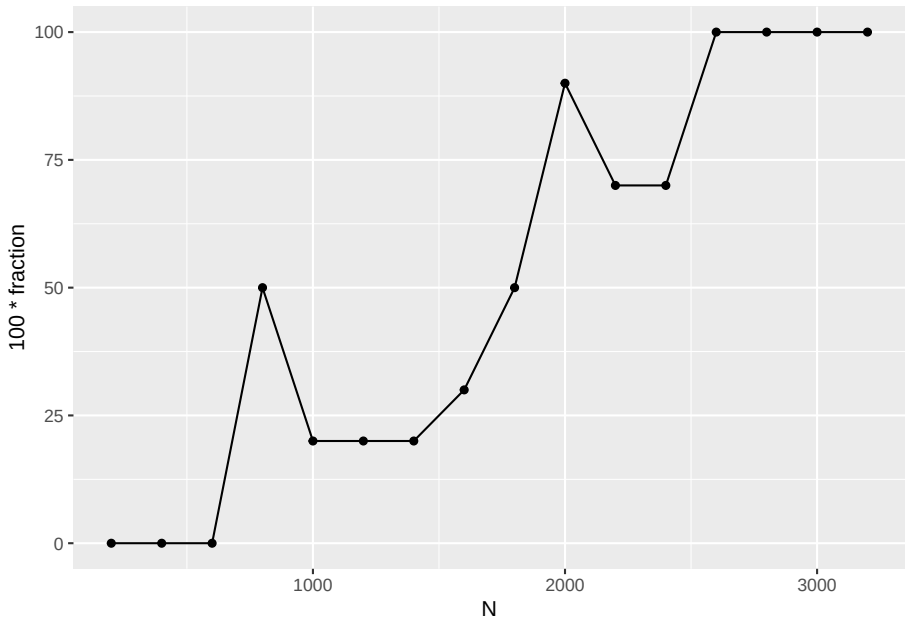
Now for each value of N, we check how many of the intervals have the desired width:

```
m6_n = m6_hpdi %>%
    group_by(N) %>%
    summarise(fraction = mean(width < 0.05))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
gg1 = ggplot(data = m6_n,
             mapping = aes(x = N, y = 100 * fraction)) +
    geom_point() +
    geom_line()
```

```
gg1
```



NB. The hard problems here all use the data below. These data indicate the gender (male=1, female=0) of officially reported first and second born children in 100 two-child families. The data is in the `homeworkch3` data set:

```
pacman::p_load(rethinking)
data(homeworkch3)

df = tibble(
    birth1 = birth1,
    birth2 = birth2
) %>%
    mutate(family = row_number())
```

4.2.14 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?

** Data story**

1. The true proportion of boys births in 2-child families is p
2. A single sample of a 2-child family has probability p of having a boy birth, and probaility $1 - p$ of having a not boy (girl) birth.

3. Each family sampled is independent of the others, and each birth within a family is independent in terms of propensity for a particular gender (this may not be quite valid).

Model

Let B denote the number of boy births and G denote the number of girl births, and assign the data model:

$$B \sim \text{Binomial}(N, p)$$

where $N = B + G$. Suppose we have vague prior information and assume it is uniform:

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

Then we can in this case, $N = 200$ (2 children in each of 100 families).

Let's check the total number of boys and girls:

```
h1_birthcounts = df %>%
    summarise(
        boy_births = sum(c(birth1, birth2)),
        total_births = length(c(birth1, birth2))
    )
h1_birthcounts
```

```
## # A tibble: 1 x 2
##   boy_births total_births
##        <dbl>        <int>
## 1        111          200
```

Now proceed to a grid-approximation of the posterior

```
# define global variables
granularity = 1000

# define grid
h1_p_grid = tibble(p = seq( from=0, to=1, length.out=granularity)) %>%
    mutate(prior = 1)

# grid-approximation of posterior
h1_posterior = h1_p_grid %>%
    mutate(
        likelihood = dbinom(
            x = h1_birthcounts$boy_births,
            size = h1_birthcounts$total_births,
            prob = p),
        posterior = (prior * likelihood)/sum(prior*likelihood)
    )
```

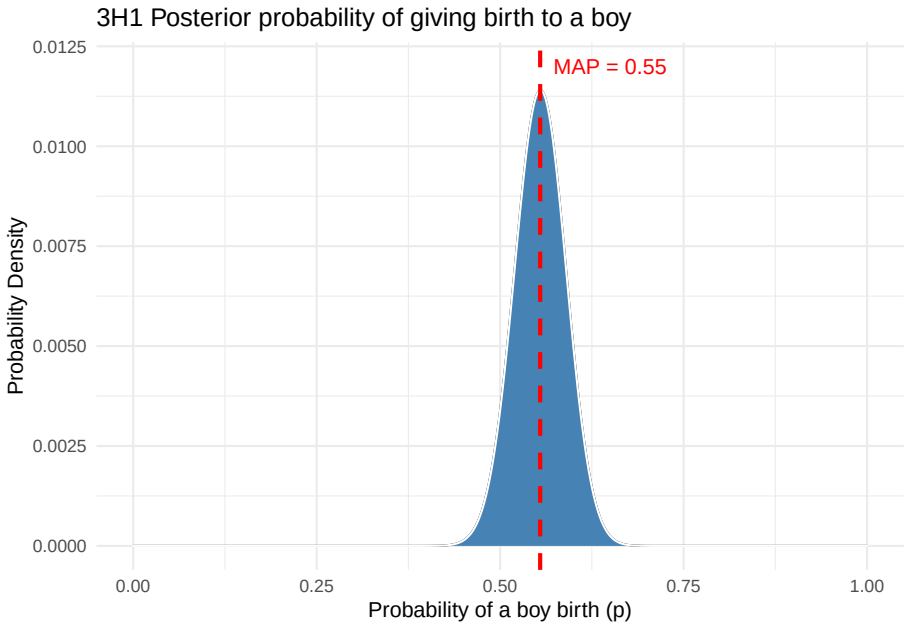
The maximum *a posteriori* (MAP) estimate is:

```
h1_map = h1_posterior %>%
    slice(which.max(posterior)) %>%
    pull(p)

h1_map %>% round(2)
```

```
## [1] 0.55
```

Create a plot to show this:

```
h1_ggp = h1_posterior %>%
    ggplot( aes(x = p) ) +
    geom_line( aes(y = posterior) ) +
    geom_area( aes(y = posterior), colour="white", fill="steelblue" )


h1_ggp + geom_vline(aes(xintercept = h1_map),
                color="red",
                linetype="dashed",
                size=1) +
    annotate(geom = "text",
             x = 0.65, y = 0.012,
             label = paste("MAP =", h1_map %>% round(2)),
             color="red") +
    labs(x = "Probability of a boy birth (p)",
         y = "Probability Density",
         title = "3H1 Posterior probability of giving birth to a boy") +
    theme_minimal()
```



4.2.15 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.

Solution

Draw samples from the grid weighted according to the posterior probabilities. Can do this following the book as:

```
# rethinking approach:
n_samples = 1e4
h2_samples = with(h1_posterior,
                  sample(p, prob = posterior, size=n_samples, replace=TRUE) )
```

and then use `rethinking::HPDI()` to find highest posterior density intervals for the given levels. However, we can also use a `tidyverse` approach:

```
# tidyverse approach:
n_samples = 1e4
h2_samples = h1_posterior %>%
    sample_n(size = n_samples, replace=TRUE, weight=posterior) %>%
    pull(p)

h2_hpdi = h2_samples %>%
    crossing(prob = c(0.5, 0.89, 0.97)) %>%
```

```
    group_by(prob) %>%
    group_map(HPDI)

h2_hpdi
```

```
## [[1]]
##      |0.5      0.5|
## 0.5005005 0.6126126
##
## [[2]]
##     |0.89     0.89|
## 0.4534535 0.6516517
##
## [[3]]
##     |0.97     0.97|
## 0.4454454 0.6746747
```

4.2.16 3H3.

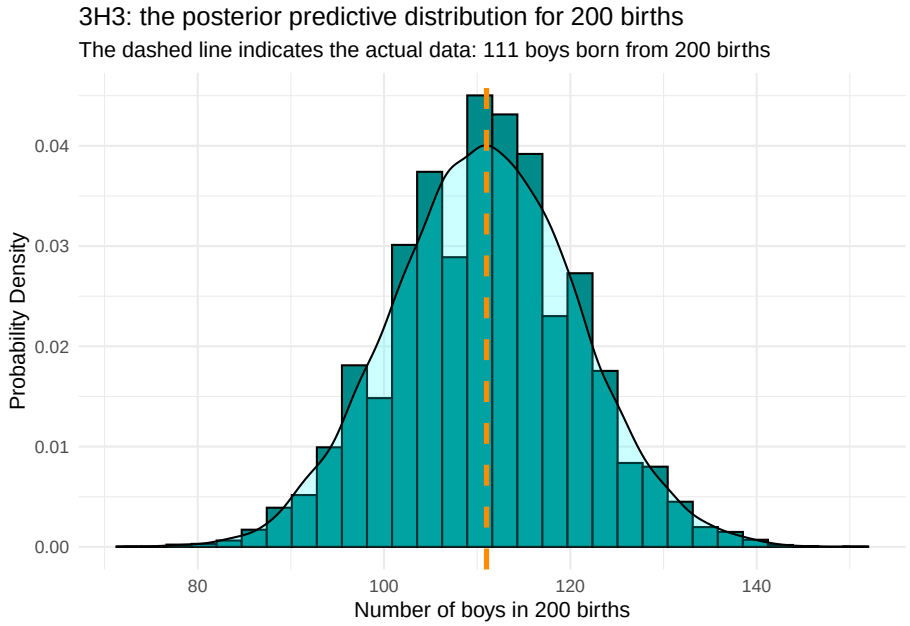
Use rbinom to simulate 10,000 replicates of 200 births. You should end up with 10,000 numbers, each one a count of boys out of 200 births. Compare the distribution of predicted numbers of boys to the actual count in the data (111 boys out of 200 births). There are many good ways to visualize the simulations, 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?

Solution Samples from the posterior predictive distribution are "possible" observed counts of boy births from 200 according to the posterior, and it's easy to generate these (under a uniform prior) by simply setting the `prob` argument of `rbinom()` to values in `h2_samples`, to draw binomial random numbers as follows:

```
# posterior predictive samples
h3_posterior_predictive = tibble(
    predicted_values = rbinom(n=1e4, size=200, prob=h2_samples) )

# plot the posterior predictive distribution and
# draw a vertical line to indicate the actual observed value of 111 from 200.
h3_ggp = h3_posterior_predictive %>%
    ggplot( aes(x = predicted_values) ) +
    geom_histogram(aes(y=..density..), colour="black", fill="cyan4") +
    geom_density( alpha=0.2, fill="cyan") +
    geom_vline(aes(xintercept = h1_birthcounts$boy_births),
               color="darkorange",
```

```
                linetype="dashed",
                size=1.25) +
    labs(x = "Number of boys in 200 births",
         y = "Probability Density",
         title = "3H3: the posterior predictive distribution for 200 births",
         subtitle = "The dashed line indicates the actual data: 111 boys born from 200
        theme_minimal()

h3_ggp
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As can be seen, the actual observed value of 111 boy births from 200 total births is very close to the peak of the posterior predictive distribution, so that indicates that our model is placing most probability in the region that the data indicates is appropriate.

4.2.17 3H4.

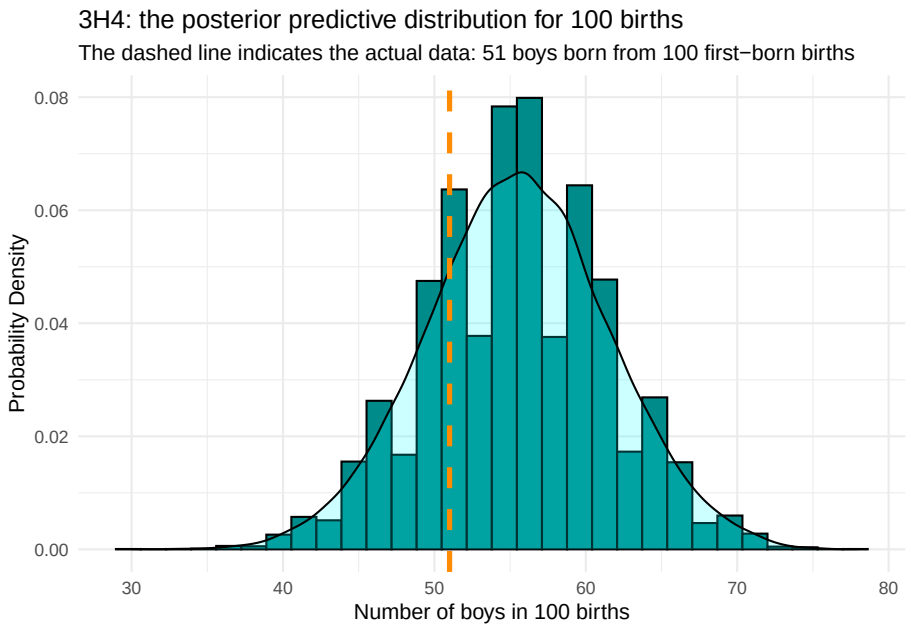
Now compare 10,000 counts of boys from 100 simulated first borns only to the number of boys in the first births, birth1. How does the model look in this light?

Solution So far we've considered getting 111 from 200 births, but these come from only 100 families, and we've so far ignored any influence of birth order.

But we can easily produce a new predictive distribution that only deals with the first 100 births (i.e. the first births in each family):

```
# posterior predictive samples
h4_posterior_predictive = tibble(
    predicted_values = rbinom(n=1e4, size=100, prob=h2_samples) )

# plot the posterior predictive distribution and
# draw a vertical line to indicate the actual observed value of 111 from 200.
h4_ggp = h4_posterior_predictive %>%
    ggplot( aes(x = predicted_values) ) +
    geom_histogram(aes(y=..density..), colour="black", fill="cyan4") +
    geom_density( alpha=0.2, fill="cyan") +
    geom_vline(aes(xintercept = sum(df$birth1)),
               color="darkorange",
               linetype="dashed",
               size=1.25) +
    labs(x = "Number of boys in 100 births",
         y = "Probability Density",
         title = "3H4: the posterior predictive distribution for 100 births",
         subtitle = "The dashed line indicates the actual data: 51 boys born from 100 first-born
        theme_minimal()

h4_ggp
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## What's the probability of between 50 and 52 first births under the posterior ## pre
h4_posterior_predictive %>%
    summarise(prob_51_100 = 100*mean(predicted_values > 50 & predicted_values < 52))
```

```
## # A tibble: 1 x 1
##   prob_51_100
##         <dbl>
## 1        5.14
```

```
## What's the achieved level of significance
h4_posterior_predictive %>%
    summarise(prob_als51_100 = 100*mean(predicted_values <= 51))
```

```
## # A tibble: 1 x 1
##   prob_als51_100
##            <dbl>
## 1           25.2
```

```
## What's the MAP estimate for the predictive distribution
h4_dens = with(h4_posterior_predictive,
               density(predicted_values))

h4_dens$x[ which.max(h4_dens$y) ]
```

```
## [1] 55.71489
```

Now the fit of the model doesn't look quite so good. It's not terrible, but the model MAP estimated number of first-born boys being about 55, and values of 51 or fewer are only expected about 25% of the time. From the predicted values, we only expect to observe 51 first-born male births slightly less than 5% of the time.

4.2.18 3H5.

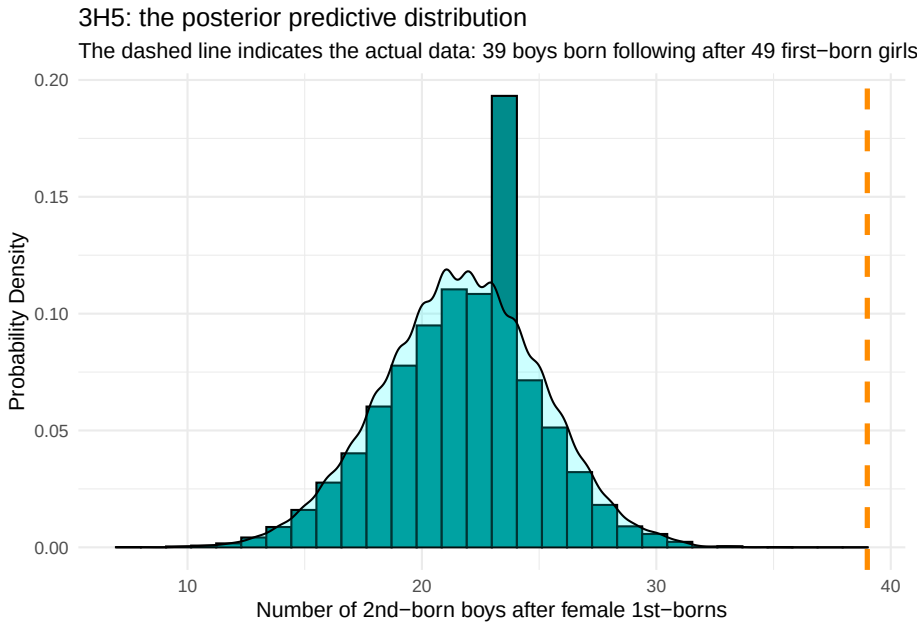
The model assumes that sex of first and second births are independent. To check this assumption, focus now on second births that followed female first borns. Compare 10,000 simulated counts of boys to only those second births that followed girls. To do this correctly, you need to count the number of 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?

Solution Let's check the number of boys born after a girl.

```
h5_birthcounts = df %>%
    filter(birth1 == 0) %>%
    summarise(boy_births = sum(birth2), total_births = n())
```

```
h5_posterior_predictive = tibble(
    predicted_values = rbinom(
        n = 1e4,
        size = h5_birthcounts$boy_births,
        prob = h2_samples)
)

h5_ggp = h5_posterior_predictive %>%
    ggplot( aes(x = predicted_values) ) +
    geom_histogram(aes(y=..density..), colour="black", fill="cyan4") +
    geom_density( alpha=0.2, fill="cyan") +
    geom_vline(aes(xintercept = h5_birthcounts$boy_births),
               color="darkorange",
               linetype="dashed",
               size=1.25) +
    labs(x = "Number of 2nd-born boys after female 1st-borns",
         y = "Probability Density",
         title = "3H5: the posterior predictive distribution",
         subtitle = "The dashed line indicates the actual data: 39 boys born following after 49 f
        theme_minimal()

h5_ggp
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## What's the achieved level of significance
h5_posterior_predictive %>%
    summarise(prob_als39_49 = 100*mean(predicted_values >= 39))
```

```
## # A tibble: 1 x 1
##   prob_als39_49
##           <dbl>
## 1             0
```

The fit here is very poor, indicating that the assumption of independence of birth gender from first born to second born is not adequate. Indeed, none of the 10000 predicted observations were as high as 39 second-born male births following on from 49 first-born female births. We should stratify this model and fit to first and then second and make that distinction the whole way through.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.