Statistical Modeling, Winter 2025

Sampling from the posterior distribution

Motivation

Our goal in the Bayesian framework is to estimate the posterior probability of our model (θ), given the data (d) - which is proportional to the probability of the data (i.e., the likelihood) $\times$ the prior probability of the model:

$$[\text{posterior}] \propto [\text{likelihood}][\text{prior}]$$
$$[\theta|d] \propto [d|\theta][\theta]$$

The typical way of calculating the posterior for most applied questions is a tricky sleight of hand: we sample from the product on the right hand side of the above equation, and then perform summary statistics. The proportionality in the above equation permits this approach. Why do we do this? Because it is relatively straightforward to calculate the likelihood and provide a prior probability distribution, and then get their product.

We can approximate the posterior distribution using **grid approximation**, **quadratic approximation**, and **Markov chain Monte Carlo methods**.

Once we have the product, how can we use it to communicate our results?

- We could use calculus to calculate the denominator in Bayes theorem (and thus retrieve a posterior probability)…but this is hard or impossible
- We could summarize samples drawn randomly from the posterior
    - MCMC produces only samples
    - Easier to think with samples

Recipe

1. Compute or approximate the posterior
2. Sample with replacement from posterior
3. Compute stuff from samples

Approximate the posterior: globe tossing

Data for the globe tossing model arise from the binomial likelihood. If we let w be a count of water and N be the number of tosses, the binomial likelihood is:

$$\Pr(w|N,p) = \frac{N!}{w!(N-w)!}p^w(1-p)^{N-w}$$

We start by setting up:

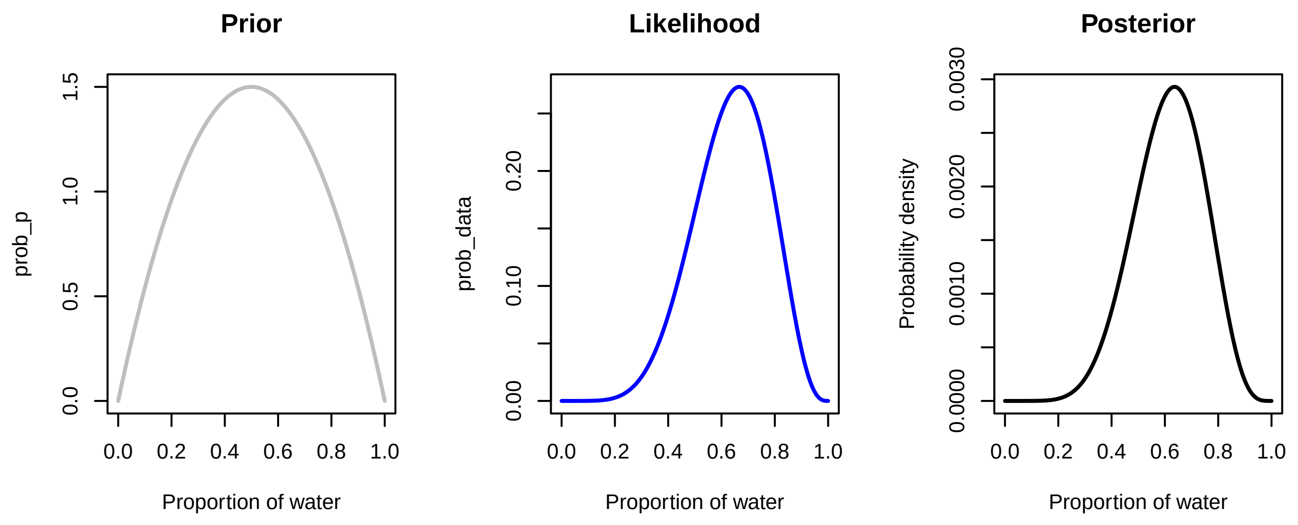
- a sequence of values for p (`p_grid`)
- an informative beta prior for p (`prob_p`)
- the probability of the data (6 w's of 9 tosses) conditional on p, i.e., the likelihood (`prob_data`)

- the posterior probability of the data (`posterior`)

```r
p_grid <- seq(from = 0, to = 1, length.out = 1000)
prob_p <- dbeta(p_grid, shape1 = 2, shape2 = 2)
prob_data <- dbinom(6, size = 9, prob = p_grid)
posterior <- prob_data * prob_p
posterior <- posterior / sum(posterior)
```

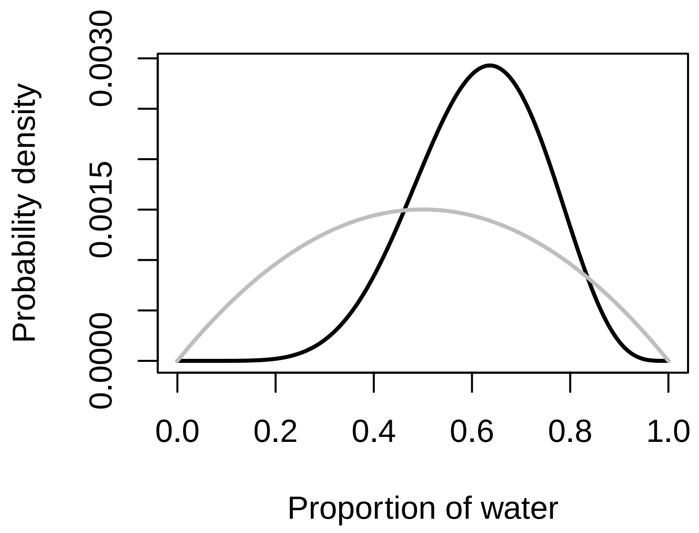
Let's visualize all three elements from Bayes' theorem:

```r
par(mfrow = c(1,3))
plot(p_grid, prob_p, type = "l", lwd = 2, col = "gray",
     xlab = "Proportion of water", main = "Prior")
plot(p_grid, prob_data, type = "l", lwd = 2, col = "blue",
     xlab = "Proportion of water", main = "Likelihood")
plot(p_grid, posterior, type = "l", lwd = 2, col = "black",
     xlab = "Proportion of water", ylab = "Probability density", main = "Posterior")
```



It is often helpful to visualize the prior together with posterior:

```r
par(mfrow = c(1,1))
plot(p_grid, posterior, type = "l", lwd = 2, col = "black",
     xlab = "Proportion of water", ylab = "Probability density")
lines(p_grid, prob_p / sum(prob_p), type = "l", lwd = 2, col = "gray")
```

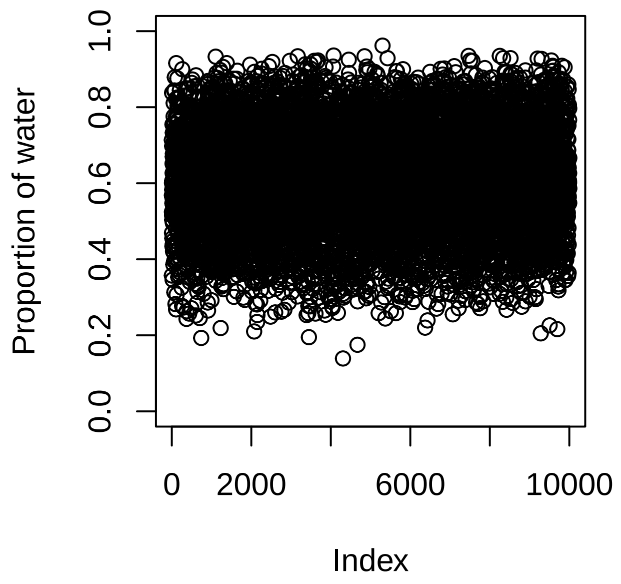


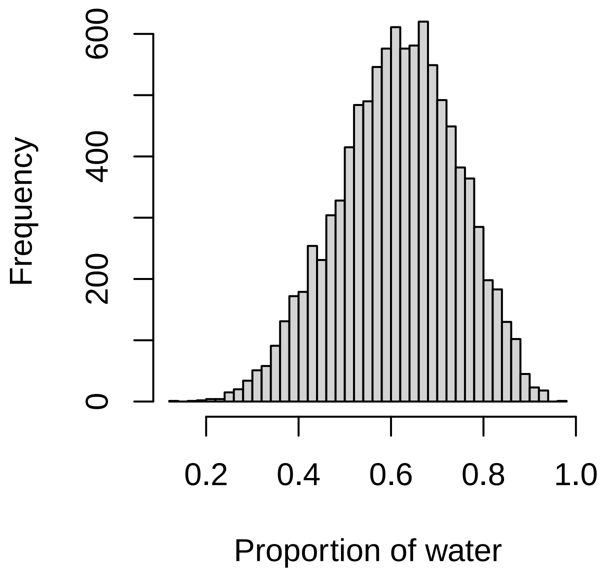
Sampling to summarize

Even though we have already answered our question - that is, we've already estimated the posterior distribution of our hypothesis, conditional on our data and model - we are going to practice *sampling* from the posterior. Why? Because in the future, we will be sampling from the posterior distribution using Markov chain Monte Carlo. It is good to practice, but also, it is useful to realize that getting *point estimates* and *uncertainty intervals* is a simple data summary problem in the Bayesian framework.

So, we'll use the `sample` function to draw randomly from the posterior distribution. Note that `sample` is a vectorized function.

```r
set.seed(100)
samples <- sample(p_grid, prob = posterior, size = 1e4, replace = TRUE)
par(mfrow = c(1,2))
plot(samples, ylim = c(0,1), ylab = "Proportion of water")
hist(samples, breaks = 50, xlab = "Proportion of water", main = "")
```





Compute stuff from samples

- Intervals of *defined boundaries*
- Intervals of *defined probability mass*
- *Point estimates*

Intervals of defined boundary ask:

How much probability mass?
What proportion of the samples are below a specified value p?

```
# (code not run)
(samples < 0.5)[1:5]
sum(samples < 0.5)
sum(samples < 0.5) / length(samples)
```

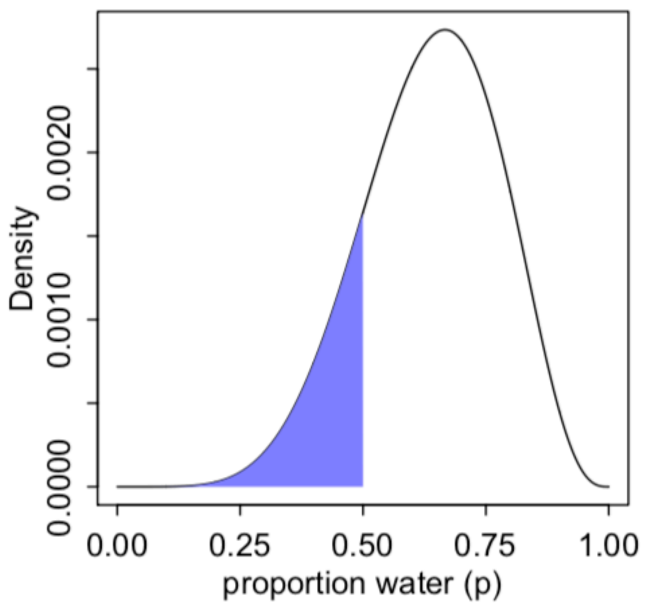


Figure 1: Interval of defined boundary, $p < 0.5$ (McElreath 2020)

Intervals of defined probability mass ask:

Which values?
What value of p will give us a specified probability mass?

```
# (code not run)
quantile(samples, 0.5)
quantile(samples, c(0.1, 0.9))
```

Two kinds of intervals of defined probability mass

Percentile intervals: assign equal probability mass to each tail

Highest posterior density intervals: narrowest interval containing the specified probability mass

Compare intervals for 3 waters in 3 tosses

- which p has the highest probability?

```
## R code 3.11 (code not run)
p_grid <- seq( from=0 , to=1 , length.out=1000 )
prior <- rep(1,1000)
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 )
## R code 3.12
PI( samples , prob=0.5 )
```

```r
## R code 3.13
HPDI( samples , prob=0.5 )
```

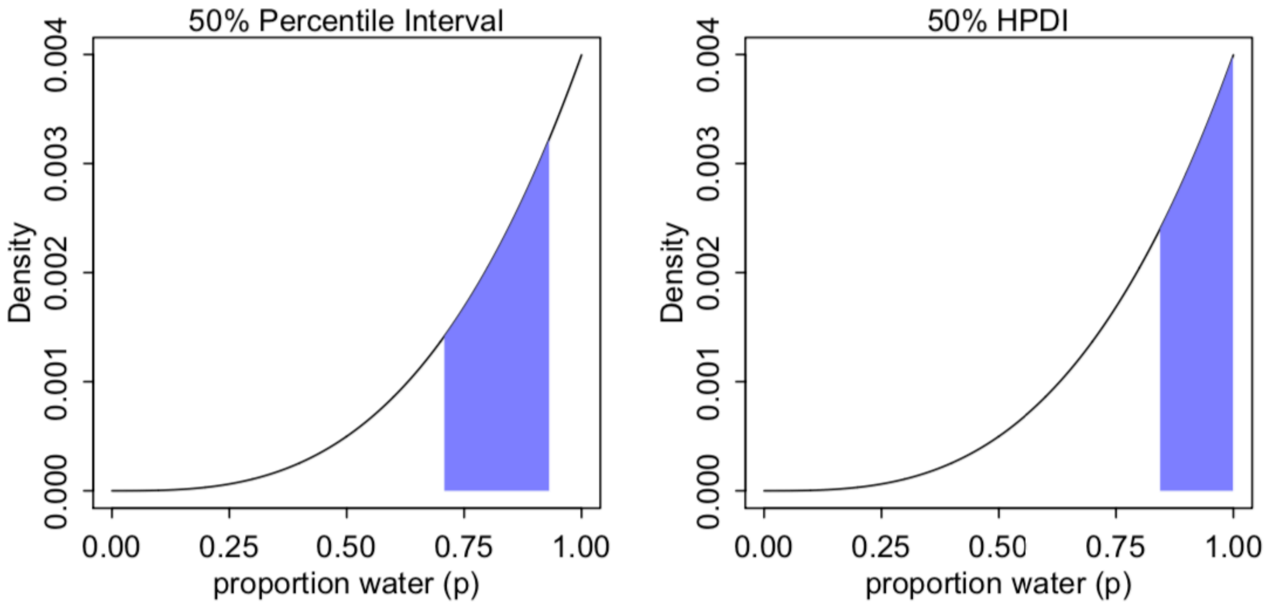


Figure 2: Intervals of defined mass (McElreath 2020)

Interpreting intervals

> What does a xx% confidence (frequentist) interval mean?

What we wish it meant: the interval of values that probably (with xx% probability) contains the true value

What it actually means: xx% of future intervals would contain the true value

Credible (Bayesian) intervals are intuitive

Compatibility intervals: contain the true value that is *compatible* with the data AND the model (McElreath's term; equivalent to credible interval)

Point estimates ask:

What is the single point that *best* describes the posterior information?

- mean
- median
- mode

For a normal distribution, these will be identical.

Otherwise known as *summary statistics* or *measures of central tendency*.

Point estimates imply a function (*estimator*) that minimizes a *loss function*

- the mean minimizes squared error loss

```
## R code 3.11 (code not run)
## R code 3.14
p_grid[ which.max(posterior) ]
## R code 3.15
chainmode( samples , adj=0.01 )
## R code 3.16
mean( samples )
median( samples )
```

Sampling to simulate prediction

Good reasons for simulating predictions (i.e., new data) from the posterior distribution:

1. Model design

- do your priors make sense?
- prior predictive distributions

2. Model checking

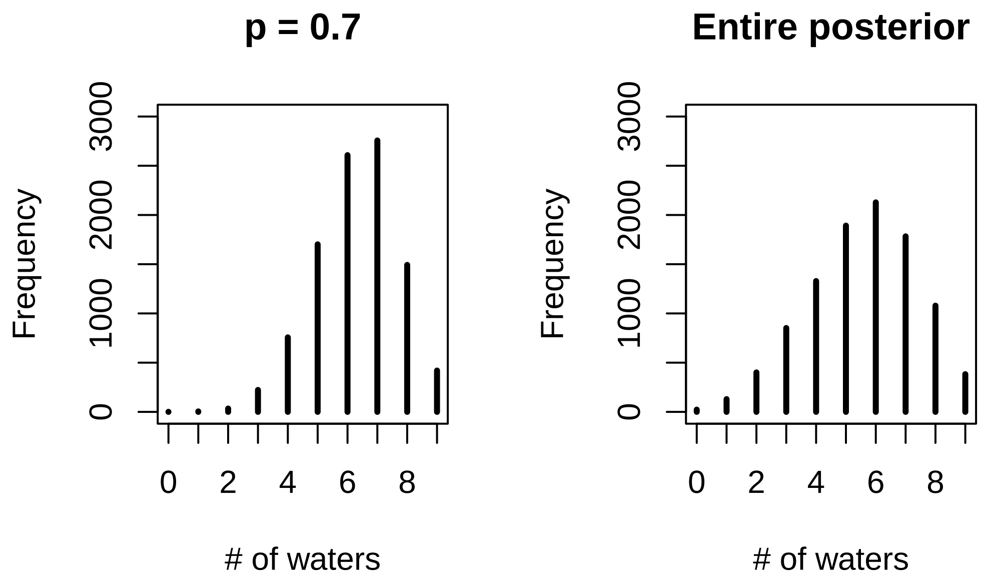
- does your posterior make sense?
- do your predictions match your observations?

3. Software validation

- if you simulate data given some parameters, can your model recover those parameters?

Compare the predictions from a single value of p = 0.7, with predictions sampled from the entire posterior:

```
set.seed(99)
# Sample from a single point estimate
dummy_w <- rbinom(1e4, size = 9, prob = 0.7)
# Sample from the entire posterior
w <- rbinom(1e4, size = 9, prob = samples)
# Plot
par(mfrow = c(1,2))
simplehist(dummy_w, xlab = "# of waters", main = "p = 0.7",
           ylim = c(0, 3000))
simplehist(w, xlab = "# of waters", main = "Entire posterior",
           ylim = c(0, 3000))
```



Practice

Complete the following exercises from *Statistical Rethinking*:

3E1-7, 3M1-5

Attribution

These notes are based on Chapter 3 of *Statistical Rethinking* (McElreath 2020).

McElreath, Richard. 2020. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan.* CRC Press.