

Day 4: Describing Posterior Distributions

Stephen R. Proulx

1/13/25

How to describe and make use of your posterior probability distribution

We've already seen how we can use Bayesian updating to approximate a posterior distribution using the "grid approximation" method. There are multiple ways to generate a posterior distribution, you could do the exact math (often statistics courses focus on this method), use a grid approximation, approximate everything with normal distributions (this is the quadratic approximation), or use MCMC methods to sample the posterior without ever really defining it (I know, it sounds magical).

Later in the course, we will use these MCMC methods and these involve having random samples from the posterior. For now, we are going to learn how to use R to sample from a posterior even if we did not generate it via MCMC methods.

Today's objectives:

- * Refresh our memory of the Bayesian algorithm using the globe tossing example.
- * Learn how to sample from a probability distribution
- * Learn how to sample from a grid approximation
- * Quantify the posterior distribution using mean, median, quantiles, and HPDI
- * Use a posterior distribution to simulate data: posterior-predictive simulations

Globe tossing: binomial style

The general algorithm for Bayesian updating: 1. Write your likelihood function: This is the model or "data story" and reflects your knowledge and assumptions about the biological system.

2. Identify the parameters, these are the part of the likelihood function that are unknown. Decide what the range of possible parameter values is. (there may be variables that you decide are known, like how many samples you have collected. It is a variable, but we won't consider it to be a parameter.)
3. Create a "grid" of the parameters. You have the ability to make this grid regular (i.e. each point is the same distance from its neighbor) or scaled some other way. This will really be a list of points in the grid, i.e. a table where each row is a set of parameters to be considered.
4. Calculate the likelihood of the data for each value of the parameters in the grid and store this value.
5. Decide on a prior and define it for each of the points in the grid.
6. Multiply the likelihood and the prior and store this "raw posterior" value.
7. Normalize the raw posterior to get the posterior probability.
8. Take a break and bask in your success

In the last class, we sequentially added information about the fraction of the earth covered in water. We did this using the posterior probability distribution from the first toss in order to apply Bayes rule after conducting the second toss. Another way to analyze the data, which will give precisely the same result, is to consider all the tosses at once, and use the **binomial distribution** as our likelihood function. The binomial distribution is a distribution that calculates the possibility of i out of n “successes” given a probability of success of p . Instead of updating our posterior n times, we will use Bayes rule once to calculate the posterior distribution for all of the data.

The binomial distribution formula:

$$Pr(i \text{ successes}) = \binom{n}{i} p^i (1-p)^{(n-i)}$$

You will write the code to do this yourself (in groups) and plot the figure. You definitely should re-use code from the last class, the only change will be that the likelihood function is now using the binomial formula, which is a built in function of R: *dbinom*

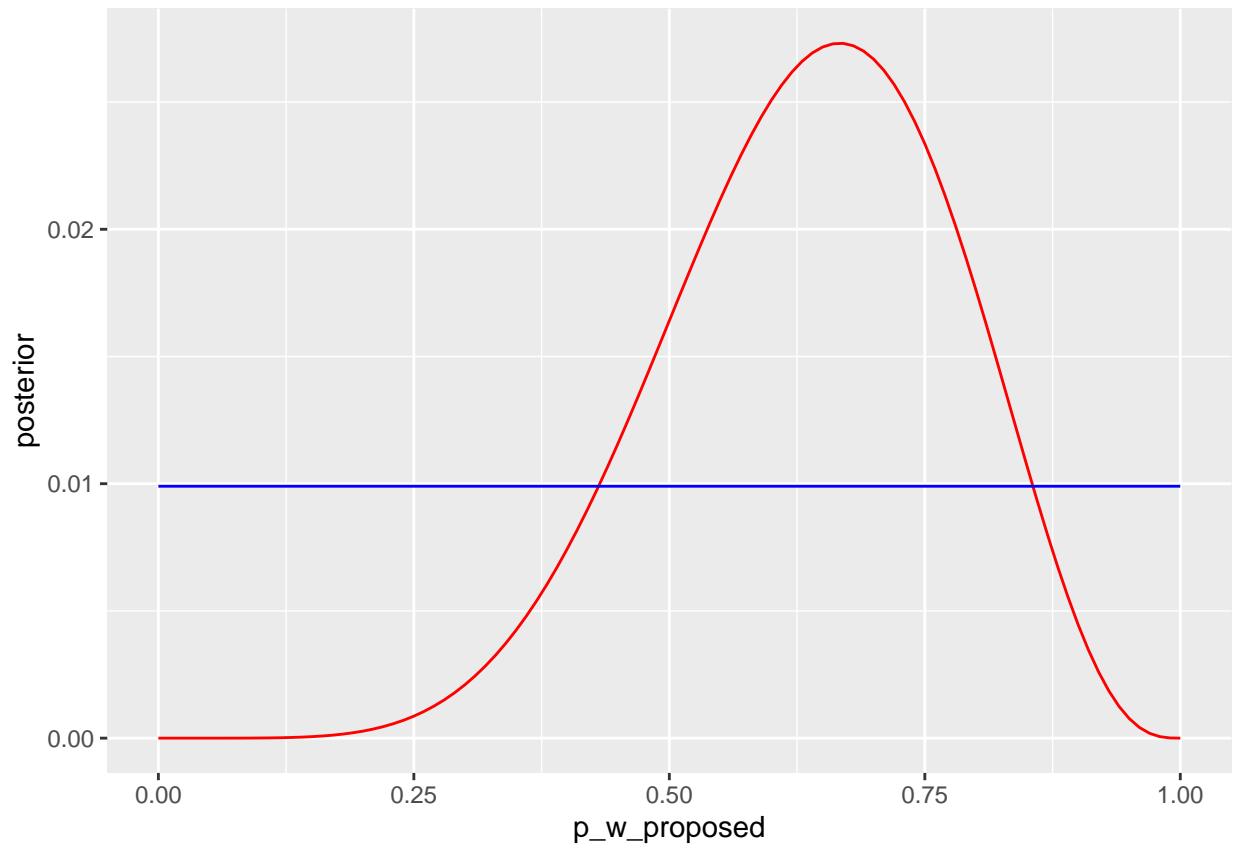
This additional useful piece of code to describe the binomial likelihood: `likelihood = dbinom(numW,size=numT,prob=p_w_p` where you will need to replace `numT` with the number of times the globe has been tossed and `numW` with the number of times it came up W.

```
posterior_table =tibble(p_w_proposed = seq(from=0, to = 1, by = 0.01) )

numW=6
numL=3
numT=numW+numL

posterior_table <- mutate(posterior_table , likelihood =dbinom(numW, size=numT, prob=p_w_proposed) ,
                          prior=1/n(),
                          raw_posterior = likelihood * prior ,
                          posterior = raw_posterior/sum(raw_posterior))

ggplot(data=posterior_table, aes(x=p_w_proposed, y=posterior))+
  geom_line(color="red")+
  geom_line(aes(x=p_w_proposed, y=prior) , color="blue")
```

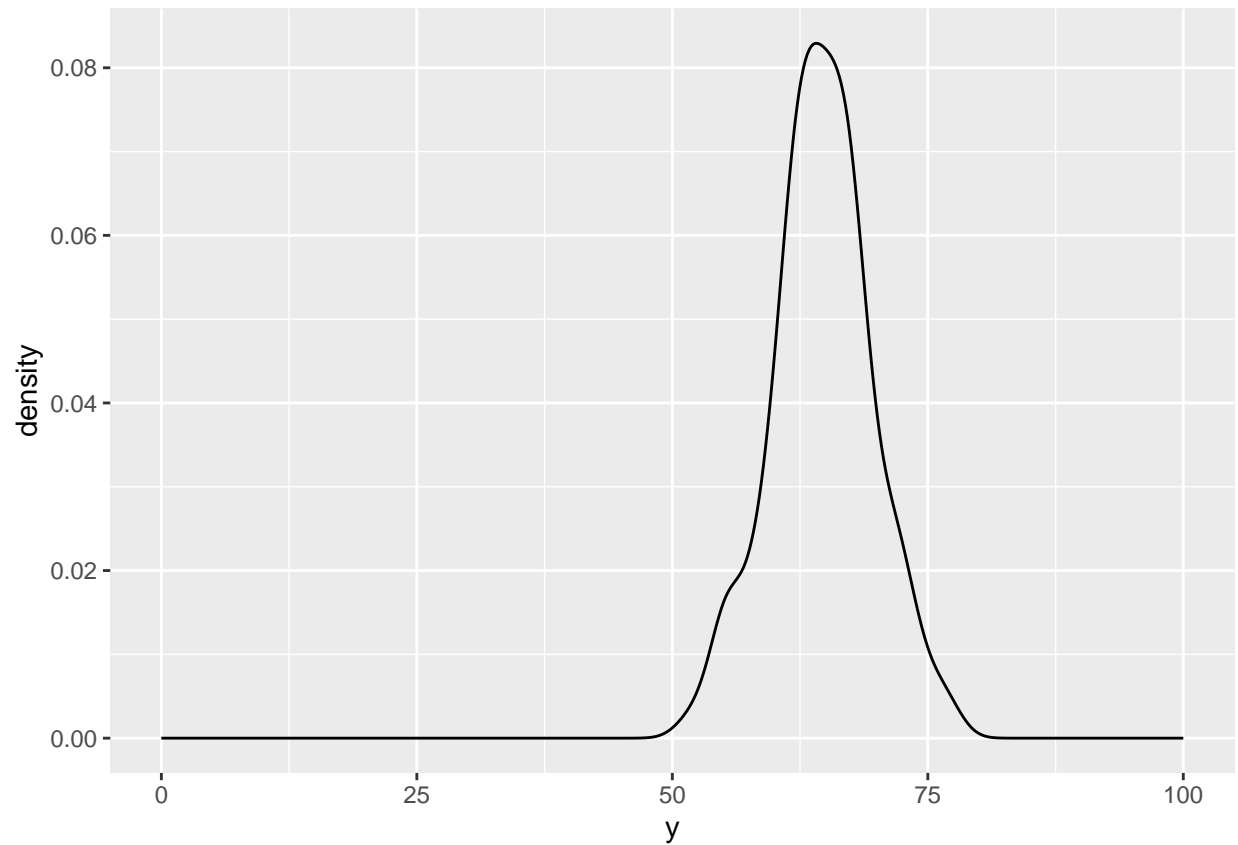


Methods of sampling distributions

- Use built in R function We can do this if we have a mathematical description based on “standard” probability distributions.

```
samples_binomial <- tibble( y = rbinom(100 , size=100, prob=0.65))
```

```
ggplot(data=samples_binomial , aes(x=y)) +  
  geom_density()+  
  scale_x_continuous(limits=c(0,100))
```



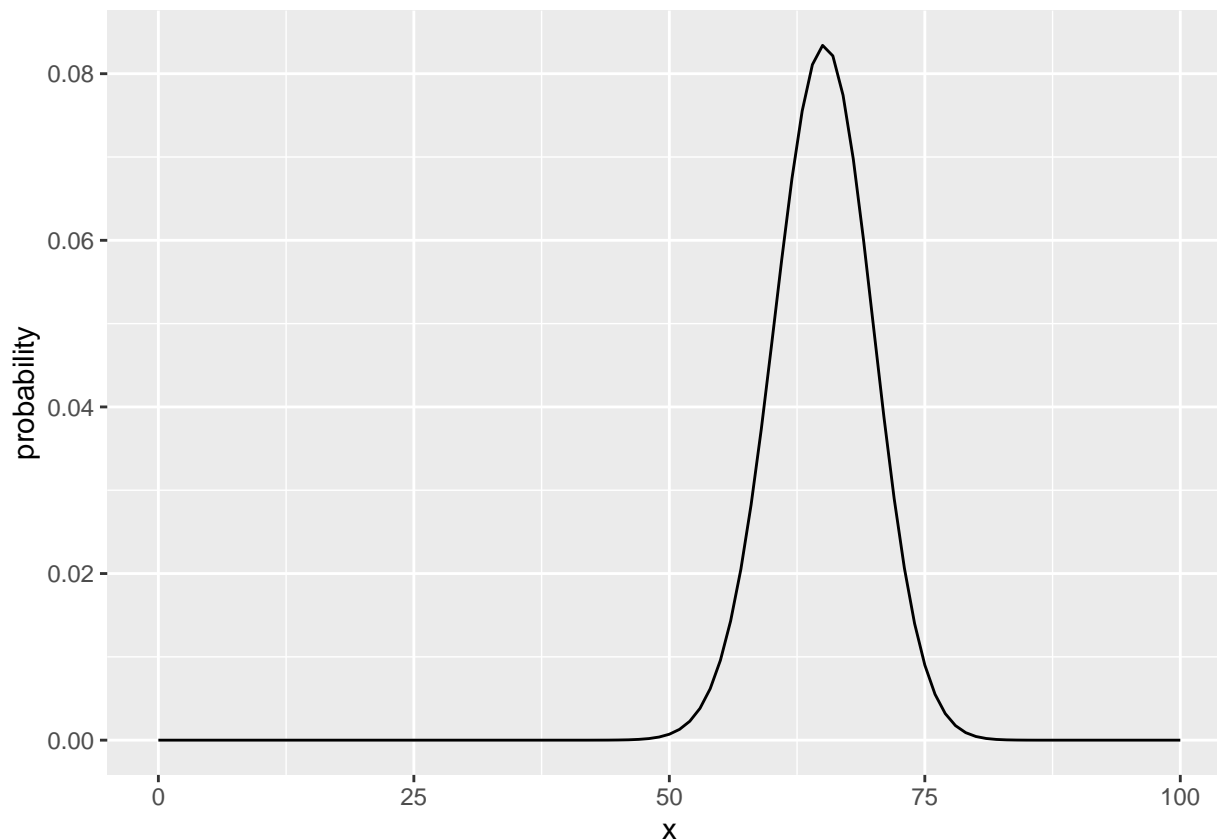
- Generate a discretized probability density function and sample from it

Even with a mathematically defined probability density function we can discretize and sample from it.

```
binom_probs <- tibble( x=seq(from = 0 , to = 100, by=1)) %>%
  mutate(probability = dbinom(x,size=100,prob=0.65))
```

Let's just see what we did:

```
ggplot(data=binom_probs , aes(x=x,y=probability)) +
  geom_line()+
  scale_x_continuous(limits=c(0,100))
```



And we can sample from it as follows using base R methods:

```
samples_binomial <- sample(binom_probs$x, prob = binom_probs$probability, size=1e4, replace=TRUE)
```

or using `dplyer` methods with the function `sample_n`. This function returns full rows of a tibble, so after we get the rows we will remove extra columns we no longer need to see (using `select`).

```
samples_binomial <- sample_n(binom_probs, weight =probability, size=1e4, replace=TRUE) %>%
  select(x)
```

Methods for quantifying PDFs.

We'll start with our rather trivial `samples_binomial` which is just a sample from a binomial with 100 trials and a probability of success of 0.65.

```
binomial_summary <- tibble(mean=mean(samples_binomial$x),
  median=median(samples_binomial$x),
  median2=quantile(samples_binomial$x,0.5),
  q10=quantile(samples_binomial$x,0.1),
  q90=quantile(samples_binomial$x,0.9),
  hdp10=HPDI(samples_binomial$x,0.8)[[1]],
  hdp190=HPDI(samples_binomial$x,0.8)[[2]])
```

```
binomial_summary
```

```
## # A tibble: 1 x 7
##   mean median median2   q10   q90 hdpi10 hdpi90
##   <dbl>  <dbl>   <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1  65.0    65     65    59    71    58    70
```

Let's write a quick function to calculate these summary statistics for later use, `dist-quant` for distribution quantify.

```
dist_quant <- function(vec_name){
  tibble(mean=mean(vec_name),
          median=median(vec_name),
          median2=quantile(vec_name,0.5),
          q10=quantile(vec_name,0.1),
          q90=quantile(vec_name,0.9),
          hdpi10=HPDI(vec_name,0.8)[[1]],
          hdpi90=HPDI(vec_name,0.8)[[2]])
}
```

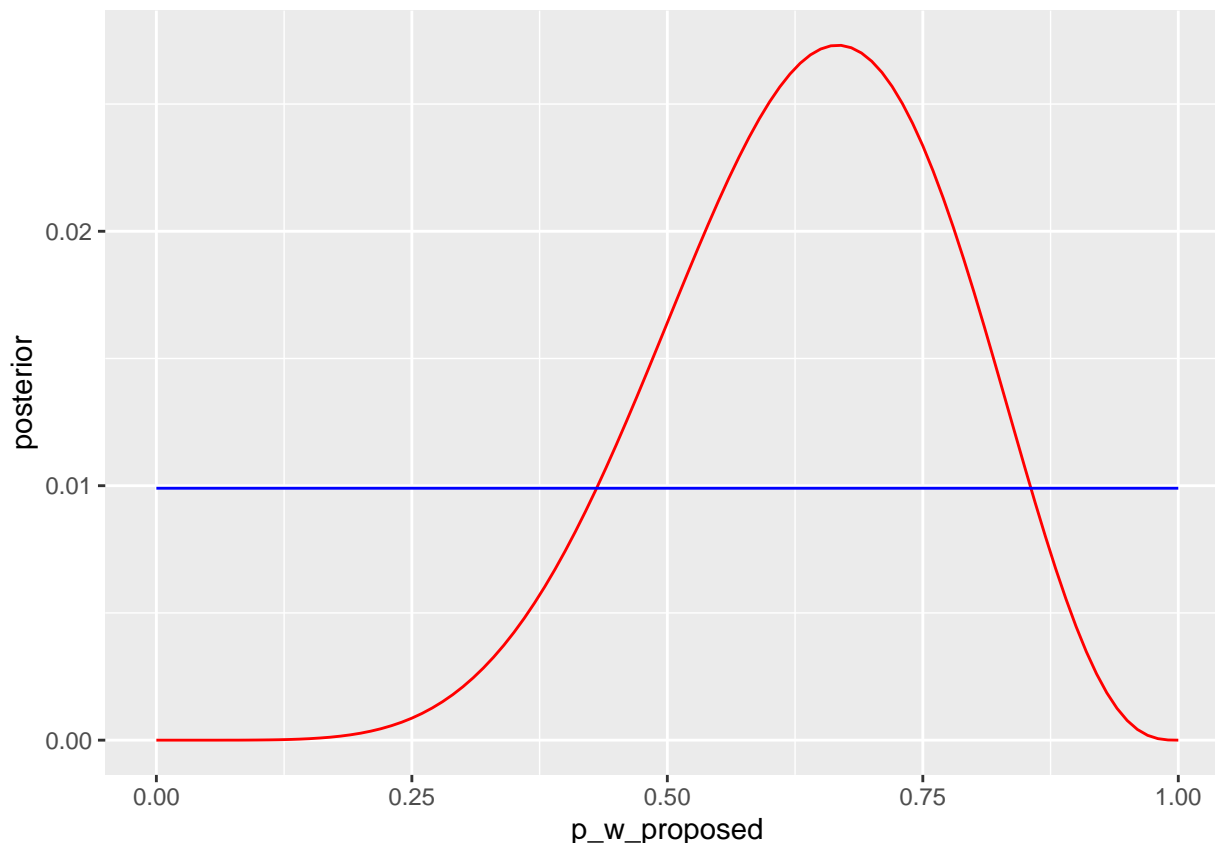
Quantifying our grid approximation of globe tossing

Here is the code to create a grid approximation of the posterior probability density for the parameter `p_w` in the globe tossing example. The data are 9 tosses with 6 observations of water.

```
data=tibble(Ws=6,Ts=9)

posterior_table <- tibble(p_w_proposed=seq(from=0, to=1, by=0.01))%>%
  mutate( likelihood= dbinom(data$Ws,size=data$Ts,prob=p_w_proposed),
          prior=1/n(),
          raw_posterior = likelihood * prior ,
          posterior = raw_posterior/sum(raw_posterior))
```

```
ggplot(data=posterior_table, aes(x=p_w_proposed, y=posterior))+
  geom_line(color="red")+
  geom_line(aes(x=p_w_proposed, y=prior) , color="blue")
```



And now we use the same procedure to sample parameter values from the posterior distribution:

```
samples_globe_tossing <- sample_n(posterior_table, weight =posterior, size=1e4, replace=TRUE) %>%
  select(p_w_proposed) %>%view()
```

We can quantify the distribution, and see that the mean and median are close to the 2/3 that was observed.

```
(globe_toss_stats <- dist_quant(samples_globe_tossing$p_w_proposed))
```

```
## # A tibble: 1 x 7
##   mean median median2   q10   q90 hdpi10 hdpi90
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0.637 0.645   0.645 0.45 0.81 0.48 0.84
```

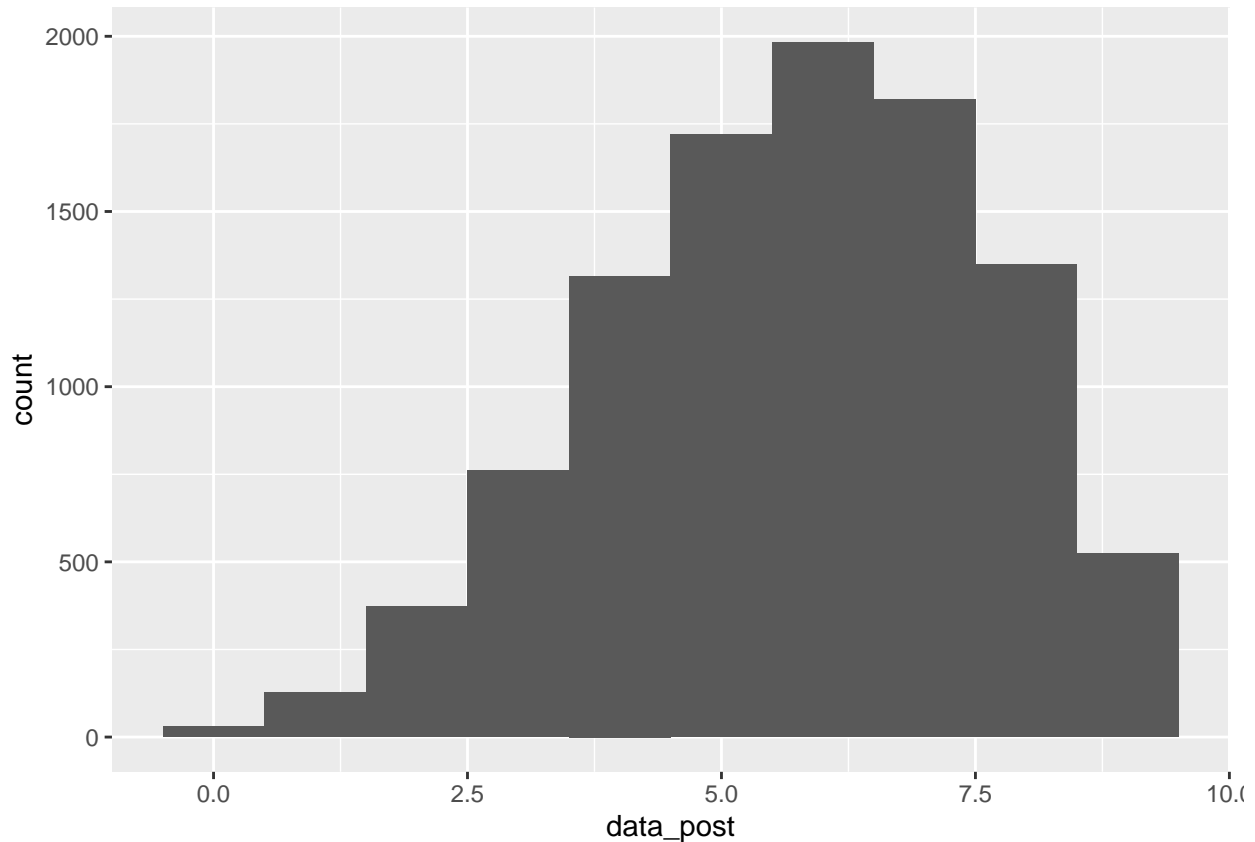
Posterior predictive simulation

Now we want to see what might look like if it came from our posterior distribution. This means that we first pick a value of p_w from the sampled values of $p_w_proposed$ and then use our “sampling distribution”, i.e. the likelihood model, to generate data.

```
posterior_predictive_data <- tibble(p_w=samples_globe_tossing$p_w_proposed) %>%
  mutate(data_post = rbinom(n(),size=9,prob=p_w),
         data_median = rbinom(n(),size=9,prob=globe_toss_stats$median))
```

What happened? Let's look at the overall result.

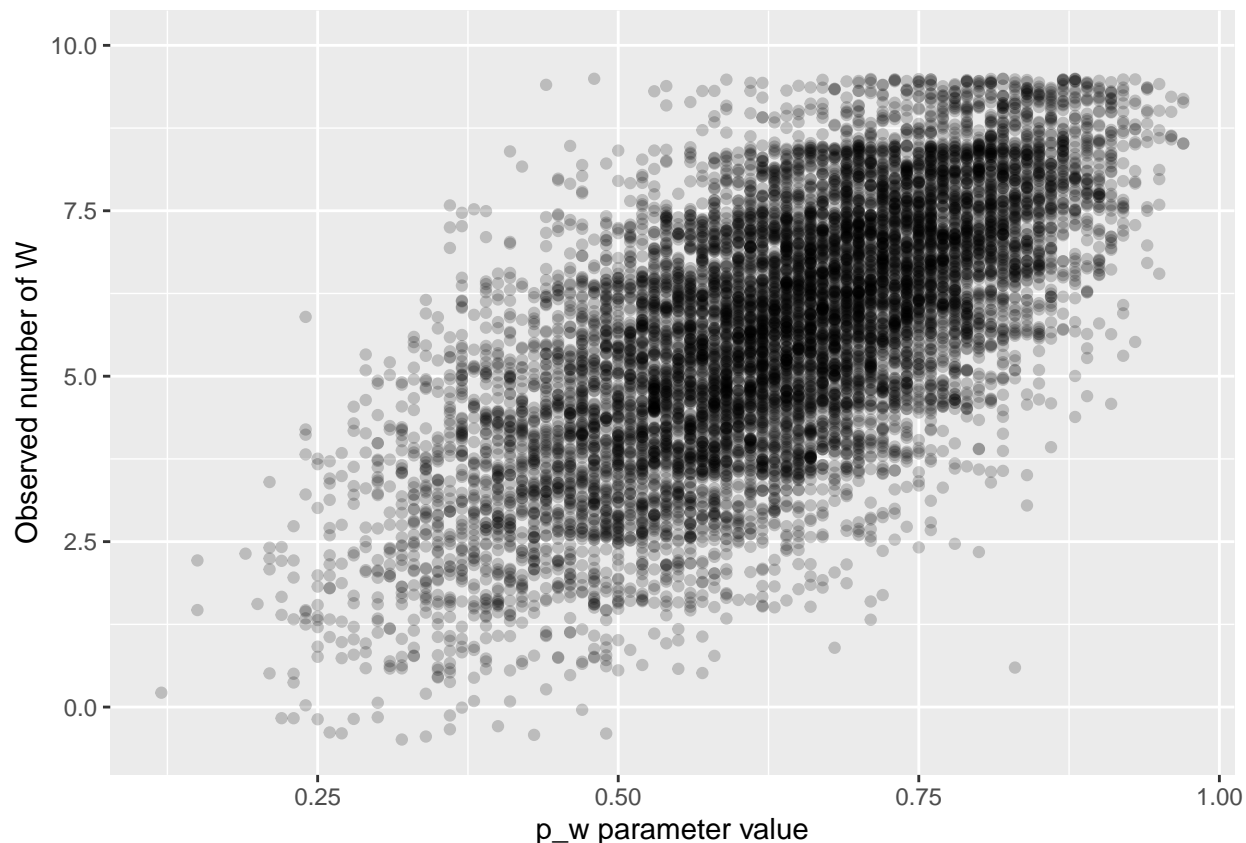
```
ggplot(data=posterior_predictive_data, aes(x=data_post)) + geom_histogram(bins=10)
```



The output is vaguely binomial, it has a mode at 6, which is the actual value we observed.

What did our procedure do, though? It picked a value of `p_w` and then used that to generate a binomial sample. We've retained the parameter from which each sample was generated, so we can plot the data to relate the value of `p_w` with the outcome in terms of observed `Ws`. To make this easier to see we'll use `geom_jitter` to "jitter" the points so they do not all overlap each other.

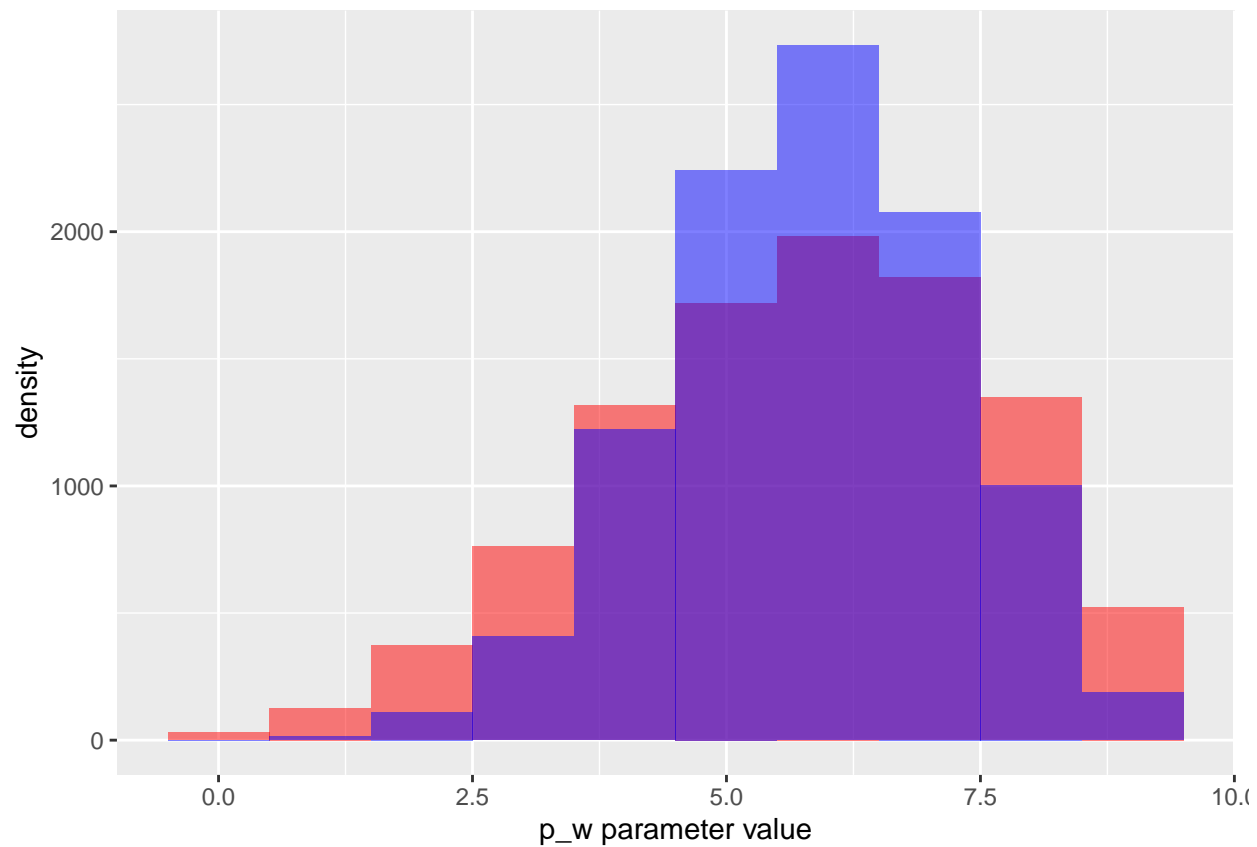
```
ggplot(data=posterior_predictive_data, aes(x=p_w,y=data_post )) +  
  geom_jitter(width=0,height = .5,alpha=.2)+  
  scale_y_continuous(limits = c(-0.5,10))+  
  labs(x="p_w parameter value",  
       y="Observed number of W")
```

What we see here is that the value of the selected parameter has an effect on the observed number of Ws. If we happened to sample a small p_w then few Ws are observed, if we happen to have sampled a large p_w then more Ws are observed.

Now let's plot the posterior-predictive simulation and compare it to a binomial with the median parameter value from the posterior. We'll use both `geom_histogram` and `geom_freqpoly` to visualize this. They generate similar visualizations, choice of which to use is really a personal preference. By setting `alpha=0.5` we get ghost-like bars so we can see through them. Note that we plot the data from the posterior predictive simulation in red and from the median parameter simulation in blue.

```
ggplot(data=posterior_predictive_data , aes(x=data_post)) +
  geom_histogram(bins=10,fill="red",alpha=0.5, aes(x=data_post)) +
  geom_histogram(bins=10,fill="blue",alpha=0.5, aes(x=data_median)) +
  labs(x="p_w parameter value",
       y="density")
```



```
ggplot(data=posterior_predictive_data , aes(x=data_post)) +
  geom_freqpoly(bins=10,color="red",alpha=0.5, aes(x=data_post)) +
  geom_freqpoly(bins=10,color="blue",alpha=0.5, aes(x=data_median)) +
  labs(x="Number of Ws",
       y="density")
```

