Day 3: Bayesian Updating

Stephen R. Proulx

1/18/2023

Calculating posterior probability distribution using grid approximation

Goals for today: * Understand how to write down the formula for the posterior distribution * Use R to perform grid approximation for a single random trial or a binomial likelihood * Perform multiple rounds of Bayesian updating to understand how previous experiments become part of the prior probability.

Recipe for doing Bayesian updating by "grid approximation"

Remember that the basic formula is

$$\text{Posterior}(p|d) = \frac{\text{Likelihood}(d|p) * \text{prior}(p)}{\text{norm(d)}}$$

The function norm(d) is a function of the data: This is because it integrates over all possible values of the parameters. So while this does not depend on the parameters, it does depend on the choice of possible parameters. We could also call this the total probability of the data, which really means the total likelihood of observing the data given the "small world" that we have constructed.

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 the store this "raw posterior" value.
7. Normalize the raw posterior to get the posterior probability.
8. Take a break and bask in your success

Putting it into practice, globe tossing sequential data aqcuisition

Here we will do the globe tossing model from chapter 2 by updating our posterior distribution with one piece of data at a time.

1. Let's write out our likelihood. It has one parameter, the probability of water, p_w. We could write

$$\Pr(\text{water}|p_w) = p_w$$

$$\Pr(\text{land}|p_w) = 1 - p_w$$

2. The only parameter in sight is p_w which can be between 0 and 1. We may want to assume that values of exactly 0 or exactly 1 are not possible (i.e. our prior for values of 0 or 1 is vanishingly small).
3. Our grid will be one-dimensional, with values ranging from 0 to 1. I'll do this by starting a tibble:

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

I'm calling this column `p_w_proposed` because I want to emphasize that this is not the value of p_w used to generate the data.

4. Now we calculate our likelihood. We're going to set the value of `d_now` (for "data now") and pass this value to our likelihood function. I'll use the coding of 1 for water and 0 for land. Note there are lots of ways to do this, I am doing it right inside the mutate function itself. The trick here is to use the fact that `(1==1)` returns 1, while `1==0` returns 0.

```
d_now=1

posterior_table <- mutate(posterior_table , likelihood = (d_now==1)*p_w_proposed + (d_now==0)*(1-p_w_pro
```

5. We'll start with a uniform prior, meaning we believe all values of p_w are equally likely. Because we are discretizing the prior density function, I'm going to go ahead and ensure that the probabilities add up to 1 by setting our prior probability to 1 divided by the number of points in our grid.

Again we add this using mutate, and we use the fact that `n()` gives the number of elements in our grid.

```
posterior_table <- mutate(posterior_table , prior=1/n())
```

You can check to see that the prior sums to 1:

```
sum(posterior_table$prior)
```

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

6. Multiply the likelihood and the prior and the store this "raw posterior" value. Again `mutate` does the trick.

```
posterior_table <- mutate(posterior_table , raw_posterior = likelihood * prior)
```

7. Normalize the raw posterior to get the posterior probability. Check to see what it sums to:

```
posterior_table <- mutate(posterior_table , posterior = raw_posterior/sum(raw_posterior))
```

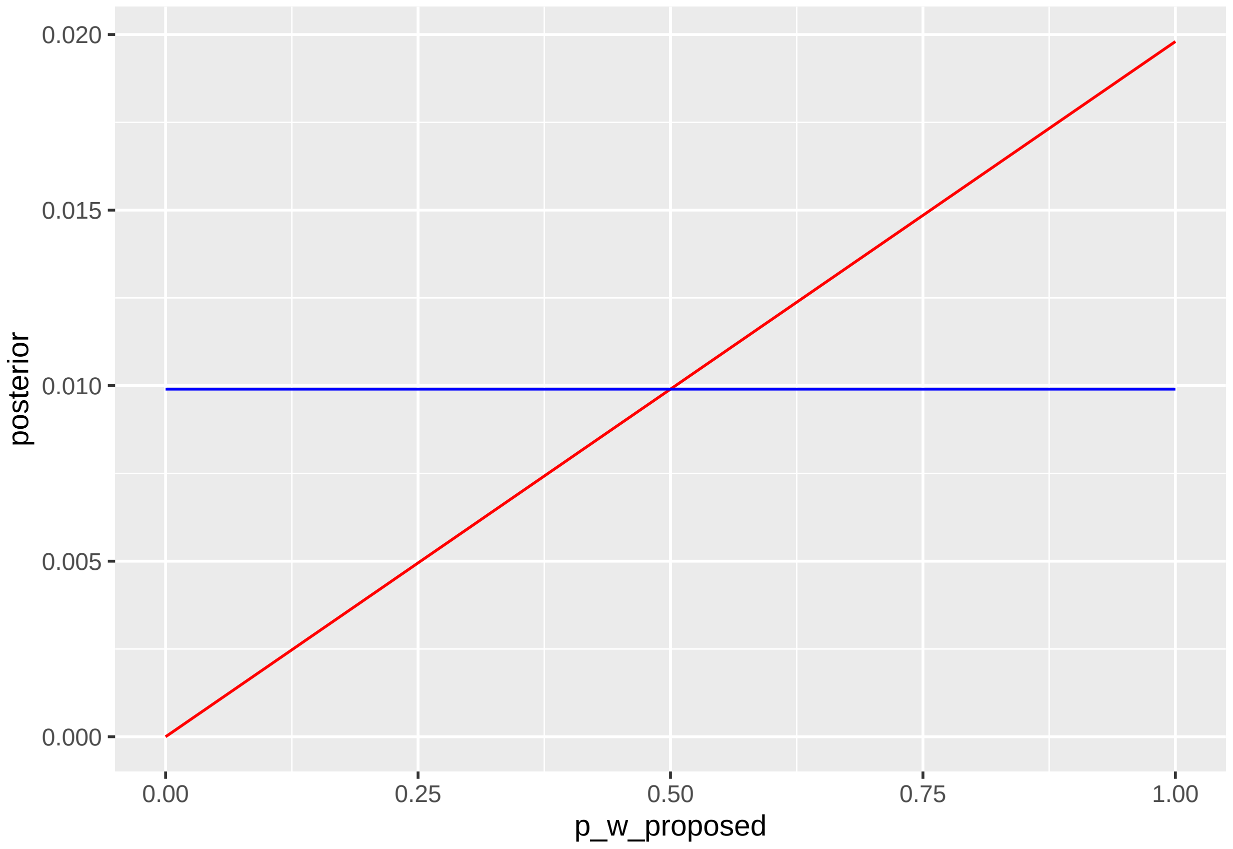
Chec to see that we have normalized it successfuly:

```
sum(posterior_table$posterior)
```

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

8. Before you rest: let's see what it looks like.

```
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")
```



Bayesian updating as new information arises

Hope you had a good break! Now let's see how we can turn the crank over and over again.

First, use the posterior you just generated as your new prior.

```
posterior_table_next <- mutate(posterior_table , prior = posterior)
```

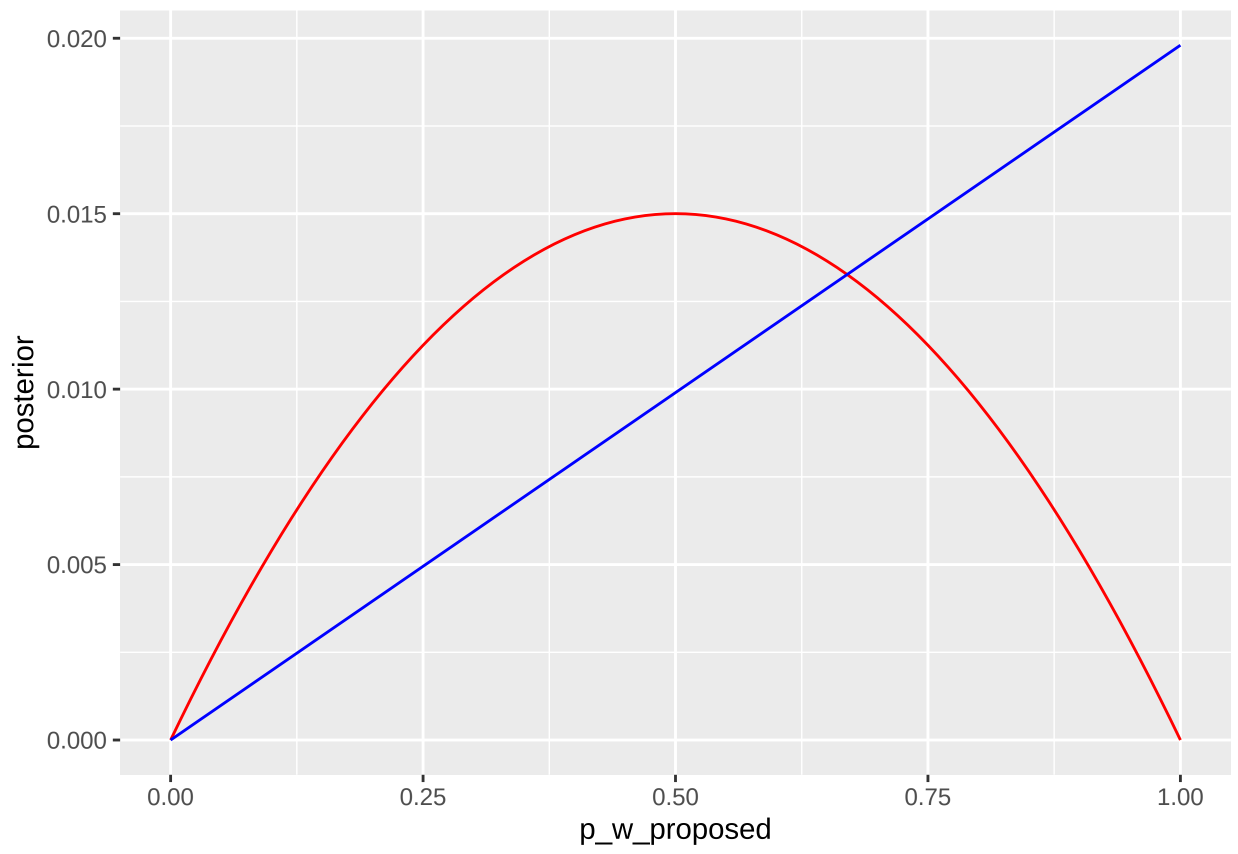
Decide what piece of data to add:

```
d_now=0
```

Turn the crank on the Bayesian updating.

```
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                               raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



Continue updating with the full sequence given in the book

We want to see what happens when we have a sequence of draws as listed in the book: W L W W W L W L W

Do this yourself by cutting and pasting the crank-turning code. Be sure to include the part where we take the posterior from the last iteration as the prior for the next.

First, use the posterior you just generated as your new prior.

```
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

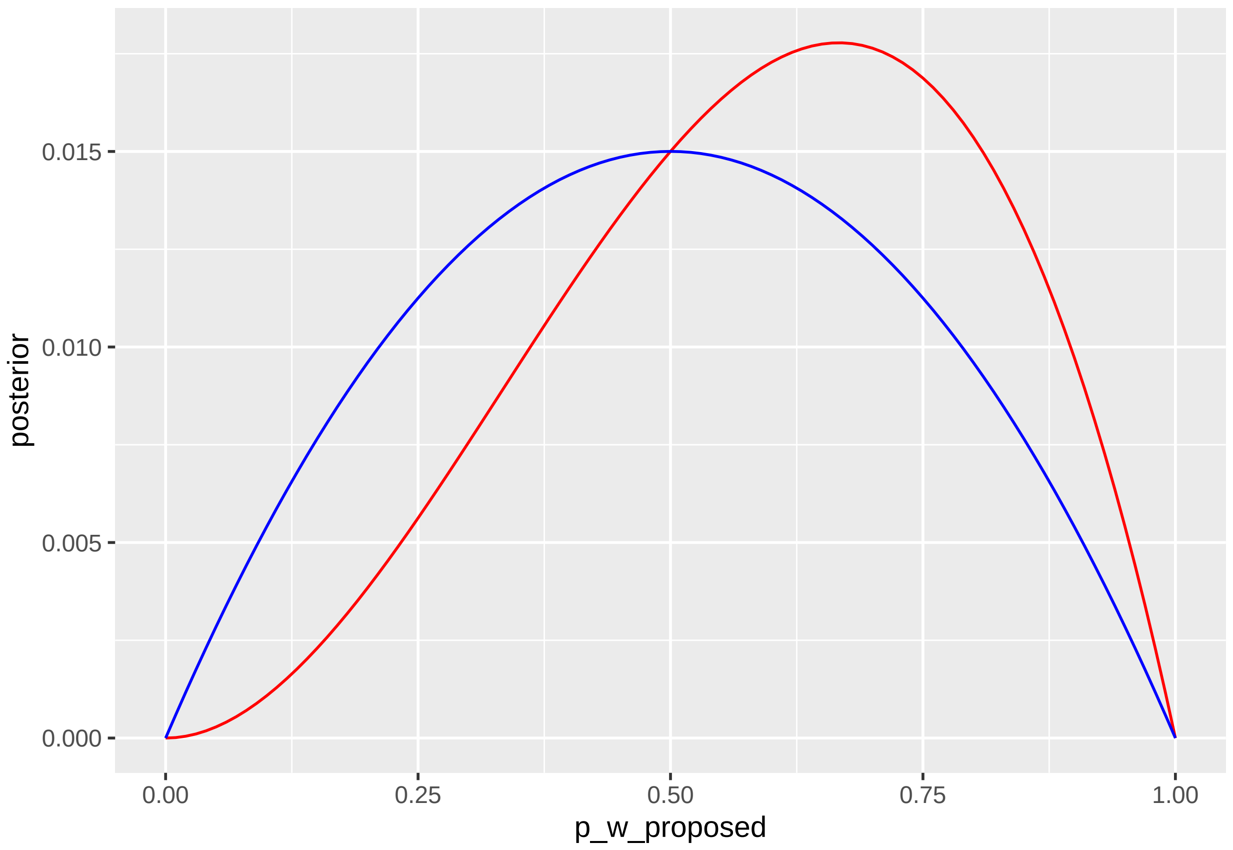
Decide what piece of data to add:

```
d_now=1
```

Turn the crank on the Bayesian updating.

```
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                               raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



```
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

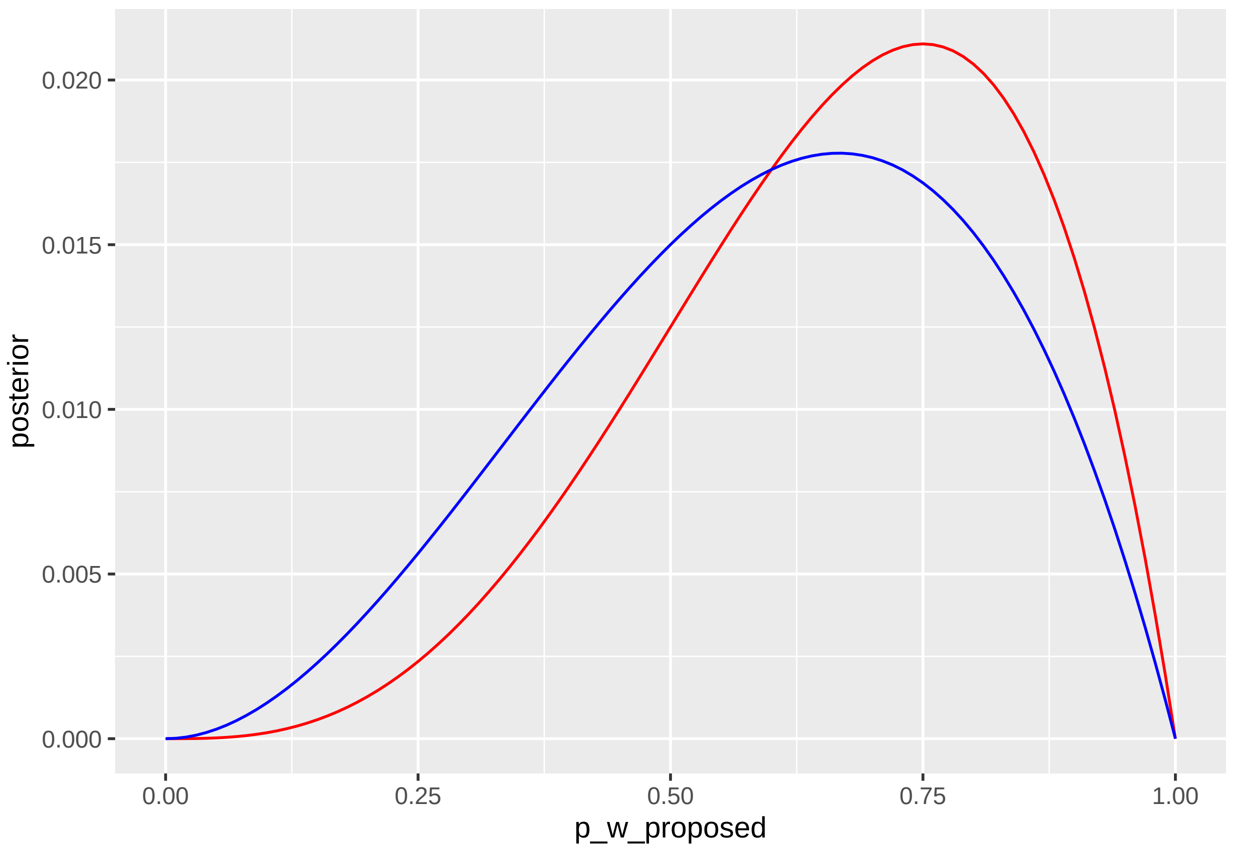
Decide what piece of data to add:

```
d_now=1
```

Turn the crank on the Bayesian updating.

```
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                               raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



```r
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

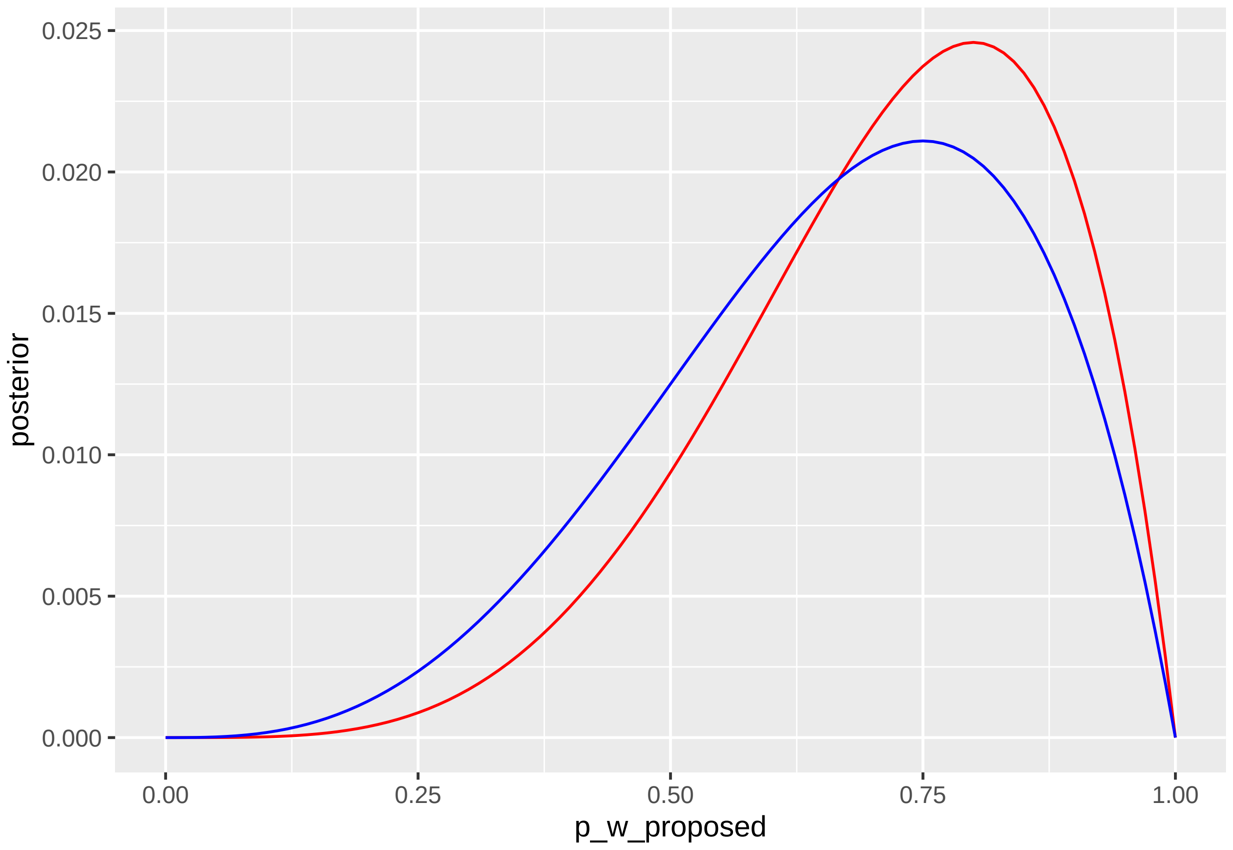
Decide what piece of data to add:

```r
d_now=1
```

Turn the crank on the Bayesian updating.

```r
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                       raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



```
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

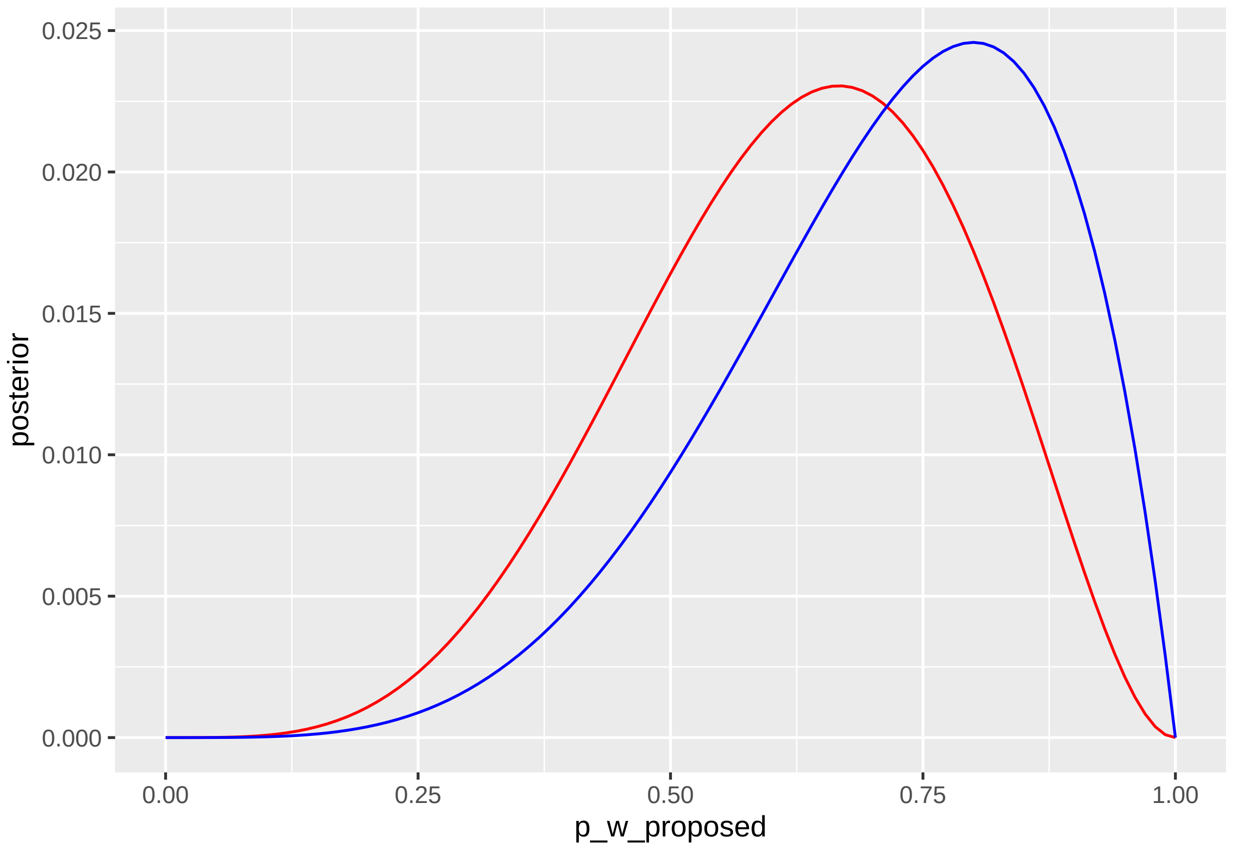
Decide what piece of data to add:

```
d_now=0
```

Turn the crank on the Bayesian updating.

```
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                       raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



```
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

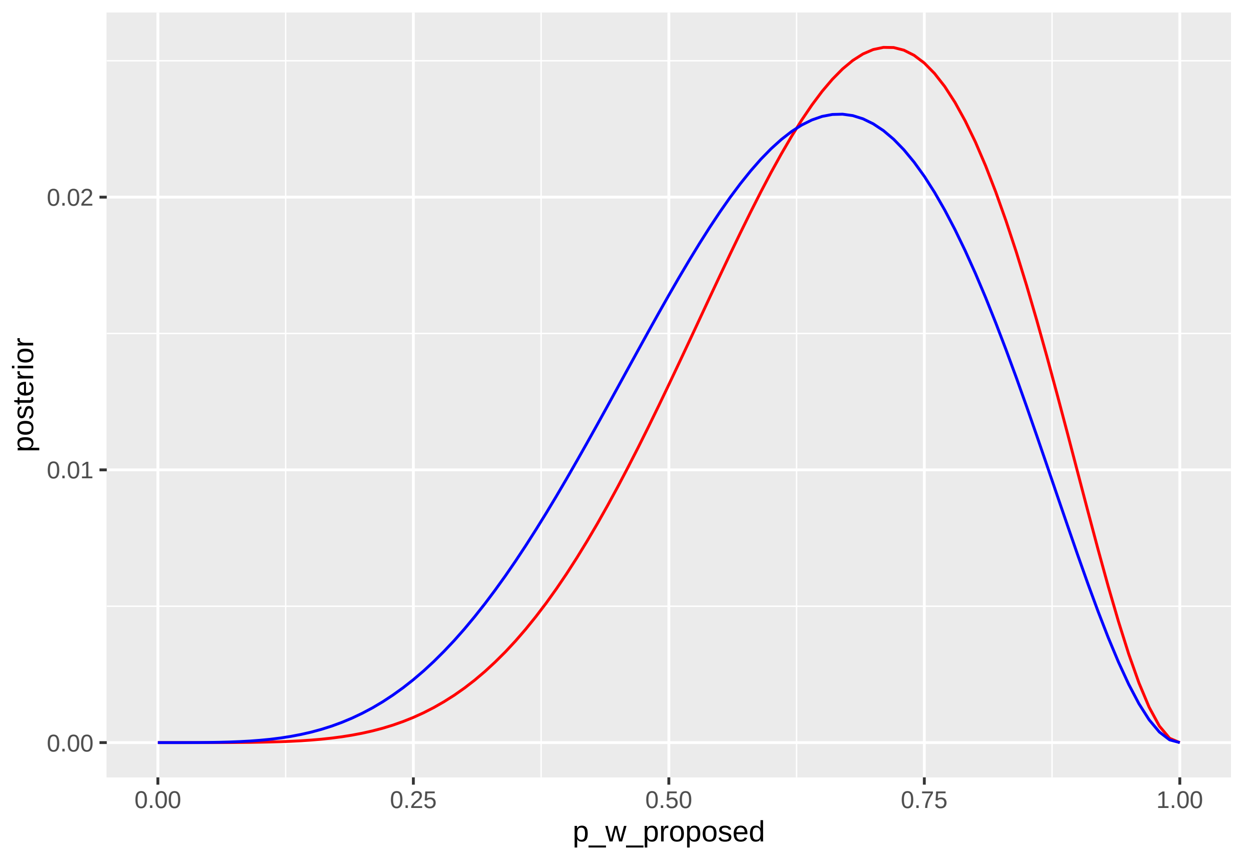
Decide what piece of data to add:

```
d_now=1
```

Turn the crank on the Bayesian updating.

```
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                               raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



```
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

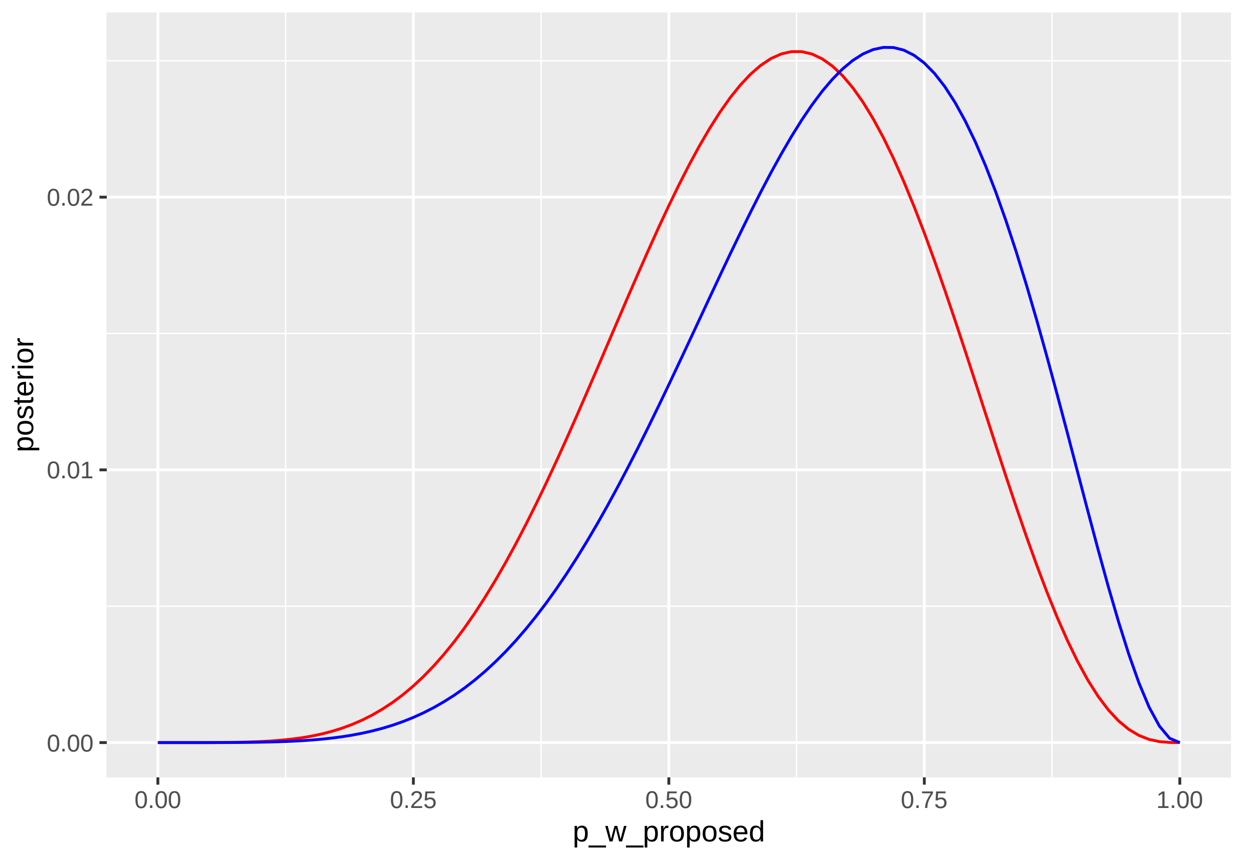
Decide what piece of data to add:

```
d_now=0
```

Turn the crank on the Bayesian updating.

```
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                       raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



```r
posterior_table_next <- mutate(posterior_table_next , prior = posterior)
```

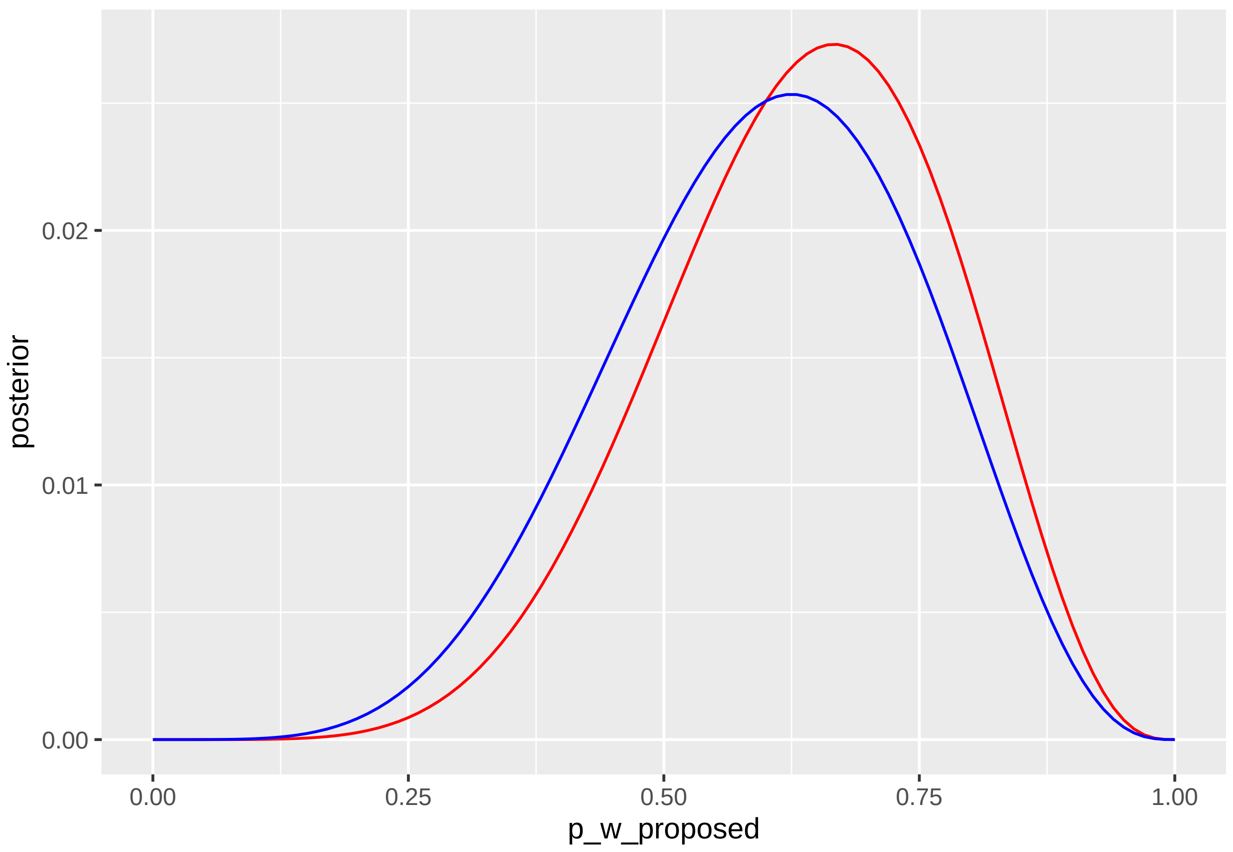
Decide what piece of data to add:

```r
d_now=1
```

Turn the crank on the Bayesian updating.

```r
posterior_table_next <- mutate(posterior_table_next , likelihood= (d_now==1)*p_w_proposed +(d_now==0)*(
                       raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

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



## Try changing the order that you add W's and L's to the list. How does it change the outcome?

Globe tossing binomial style

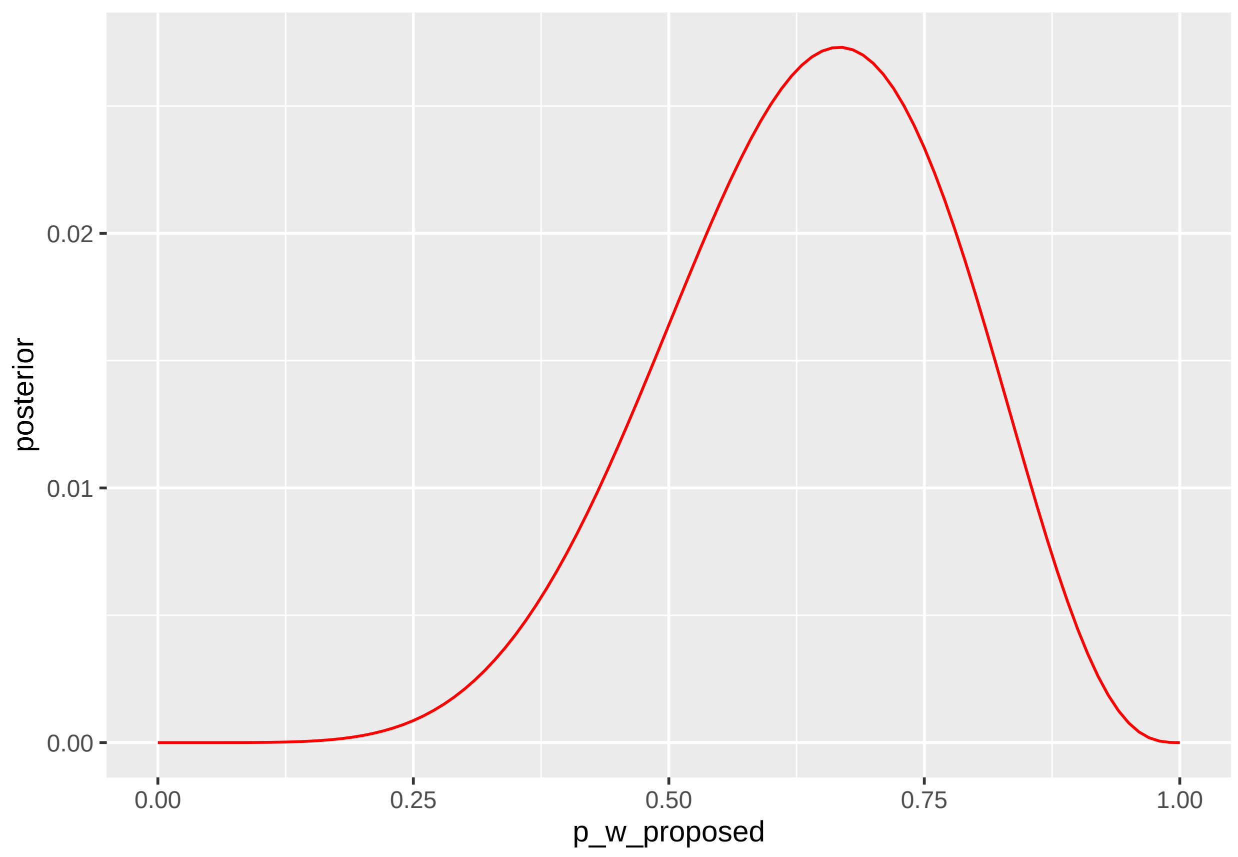
We can also do this by grouping the W's and L's together ahead of time. Because we are assuming that the order of the observations does not matter we can use a binomial likelihood to describe a pool of observations.

You will write the code to do it yourself and plot the figure. This additional useful piece of code to describe the binomial likelihood: `likelihood = dbinom(numW,size=numT,prob=p_w_proposed)` 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.

```
ws=6
ls=3

posterior_binom =tibble(p_w_proposed = seq(from=0, to = 1, by = 0.01) ) %>%
   mutate( prior=1/n(), likelihood= dbinom(ws,size=ws+ls,prob= p_w_proposed),
                       raw_posterior = likelihood * prior , posterior = raw_posterior/sum(raw_posterior
```

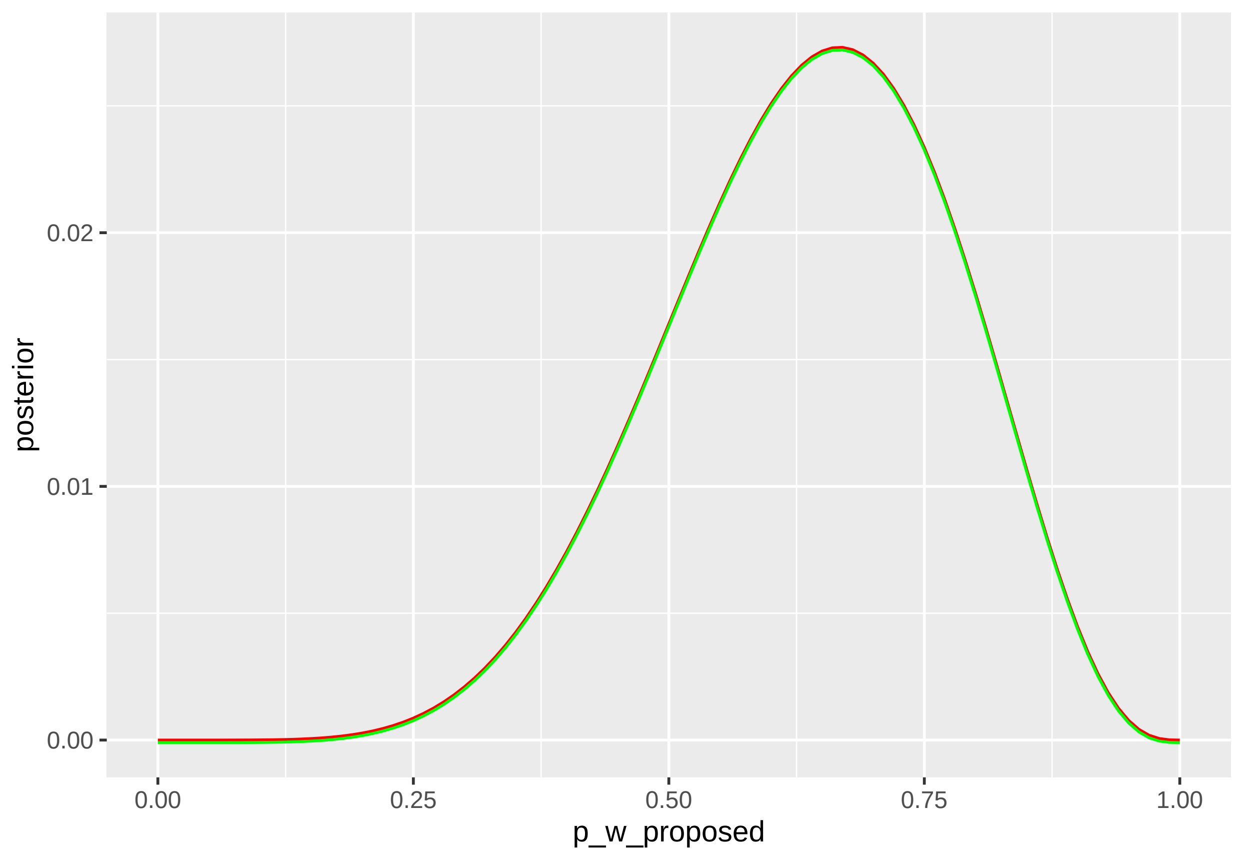
```
ggplot(data=posterior_binom, aes(y=posterior, x=p_w_proposed ))+
  geom_line(color="red")
```



And if we want to compare to the sequential approach (I shifted the sequential answer down by 0.0001 units just so you can see them both)

```
ggplot(data=posterior_binom, aes(y=posterior, x=p_w_proposed ))+
  geom_line(color="red")+
  geom_line(data=posterior_table_next, aes(x=p_w_proposed, y=posterior-0.0001, alpha(0.5)),color="green
```

```
## Warning in geom_line(data = posterior_table_next, aes(x = p_w_proposed, :
## Ignoring unknown aesthetics:
```



Using the same technique to answer other types of Bayesian question

Recall our covid diagnosis question. Assume that the probability of cold symptoms if you have a cold is 100%, while if you have covid it is 25%. 10% of the population has a cold at any given time, and 1% have covid. If you have cold symptoms, what is the probability that you have covid?

Here the prior is the probability of covid in the absence of new data/information, in other words the frequency of covid in the entire population. The "parameter" we are investigating is the patient's covid infection state, 0 if they don't have covid and 1 if they do.

```
dat=1 # you do have symptoms
pcovid=0.01
pcold=0.25
psym_cov=0.25
posterior_covid <- tibble(covid = seq(from=0, to=1, by=1), prior=c(1-pcovid,pcovid)) %>%
  mutate(likelihood = dat*(covid*(pcold*1 + (1-pcold)*psym_cov) + (1-covid)*((1-pcold)*0+pcold*1) ) +
           (1-dat)*(covid *( (1-pcold)*(1-psym_cov) + pcold*0 ) + (1-covid)*((1-pcold)*1+(pcold)*0) ),
         raw_posterior=likelihood*prior,
         posterior=raw_posterior/sum(raw_posterior))%>%
  view()
```

Try this with `dat=0` meaning that the data are that the patient does not have cold symptoms.