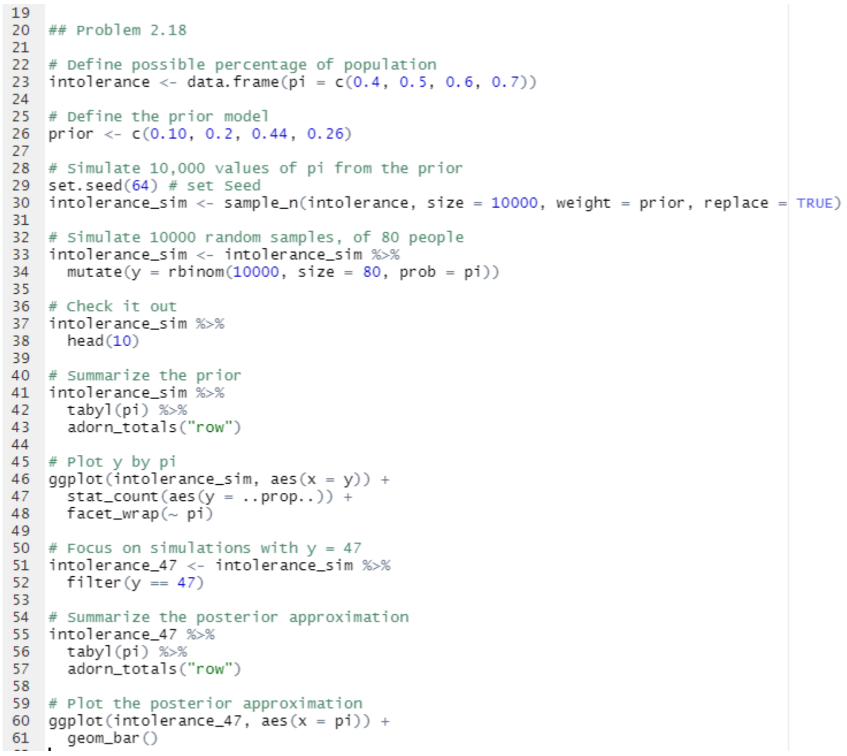
Programming Task 2.1

The code is based on the one explained on the book, on chapter 2.3.7 Posterior simulation

Problem 1.

This is problem 2.18

Here we can see the code, as well as in the R file that I uploaded as well.



```r

## Problem 2.18

# Define possible percentage of population
intolerance <- data.frame(pi = c(0.4, 0.5, 0.6, 0.7))

# Define the prior model
prior <- c(0.10, 0.2, 0.44, 0.26)

# Simulate 10,000 values of pi from the prior
set.seed(64) # set Seed
intolerance_sim <- sample_n(intolerance, size = 10000, weight = prior, replace = TRUE)

# Simulate 10000 random samples, of 80 people
intolerance_sim <- intolerance_sim %>%
  mutate(y = rbinom(10000, size = 80, prob = pi))

# Check it out
intolerance_sim %>%
  head(10)

# Summarize the prior
intolerance_sim %>%
  tabyl(pi) %>%
  adorn_totals("row")

# Plot y by pi
ggplot(intolerance_sim, aes(x = y)) +
  stat_count(aes(y = ..prop..)) +
  facet_wrap(~ pi)

# Focus on simulations with y = 47
intolerance_47 <- intolerance_sim %>%
  filter(y == 47)

# Summarize the posterior approximation
intolerance_47 %>%
  tabyl(pi) %>%
  adorn_totals("row")

# Plot the posterior approximation
ggplot(intolerance_47, aes(x = pi)) +
  geom_bar()
```

The first thing that we need to do is to define the prior model, for that we need to indicate the π and the f(π). We then simulate 10,000 values in order to have a big sample size to be able to approximate the results. But before simulating we fixed the seed to '64' to make it repeatable.

For each of the 10,000 prior possible values we need to simulate 80 people from which some will be lactose intolerance. For thar we can use a binomial model using 'rbinom()'. To see if everything has worked correctly we can print the first 10 rows of the 10,000 to see how it looks using 'head(10)'.

```
> intolerance_sim %>%
+   head(10)
    pi  y
1  0.6 43
2  0.5 46
3  0.6 45
4  0.4 26
5  0.6 58
6  0.5 37
7  0.4 31
8  0.7 53
9  0.7 61
10 0.7 58
```

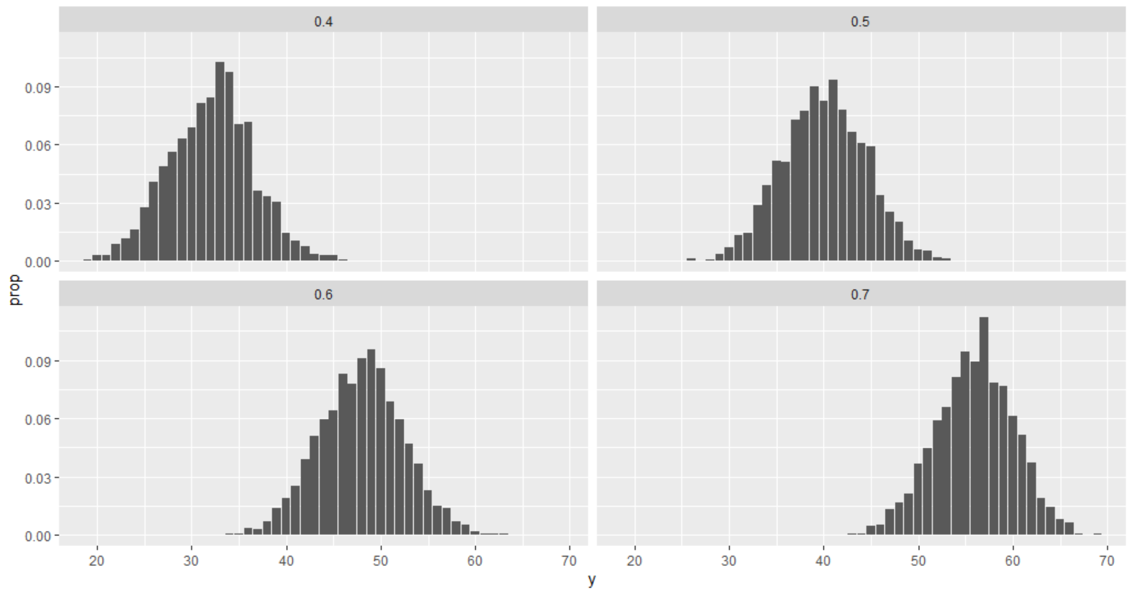
We can see the different values of ‘π’ and the corresponding number of adults that are lactose intolerant in the ‘y’ column, we can also see that the higher the ‘π’ the higher number of adults with intolerance gets.

To see it more clearly, we can print the count for each of the values of ‘π’, in the ‘percent’ column we can see that the values are almost the same as in the original problem. 0.1 for the ‘π’ 0.4; 0.2 for the ‘π’ 0.5; 0.44 for the ‘π’ 0.6; and 0.26 for the ‘π’ 0.7.

```
> # Summarize the prior
> intolerance_sim %>%
+   tabyl(pi) %>%
+   adorn_totals("row")
    pi     n percent
   0.4  1046  0.1046
   0.5  2060  0.2060
   0.6  4334  0.4334
   0.7  2560  0.2560
 Total 10000  1.0000
> 
```

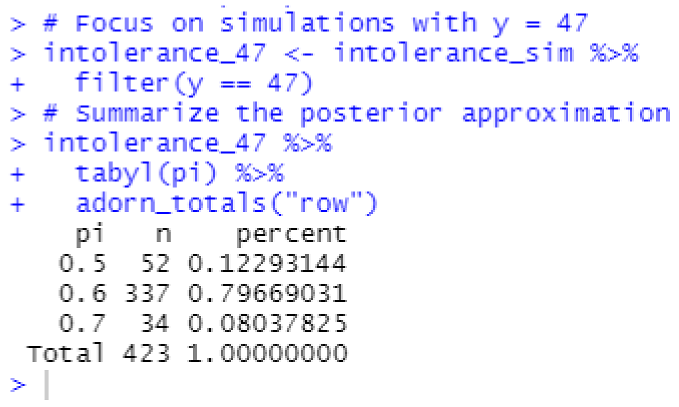
We can plot the number of adults with intolerance for each of the different ‘π’, using:

```
> ggplot(intolerance_sim, aes(x = y)) +
+   stat_count(aes(y = ..prop..)) +
+   facet_wrap(~ pi)
> 
```



We can see the different distributions, as expected, the lower the ‘π’ the lower number of adults with intolerance, and the higher the ‘π’ the higher the number.

Now according to the problem, we need to focus on the ones with 47 adults with lactose intolerance, and now we can print the values, and we get wats the posterior approximation.

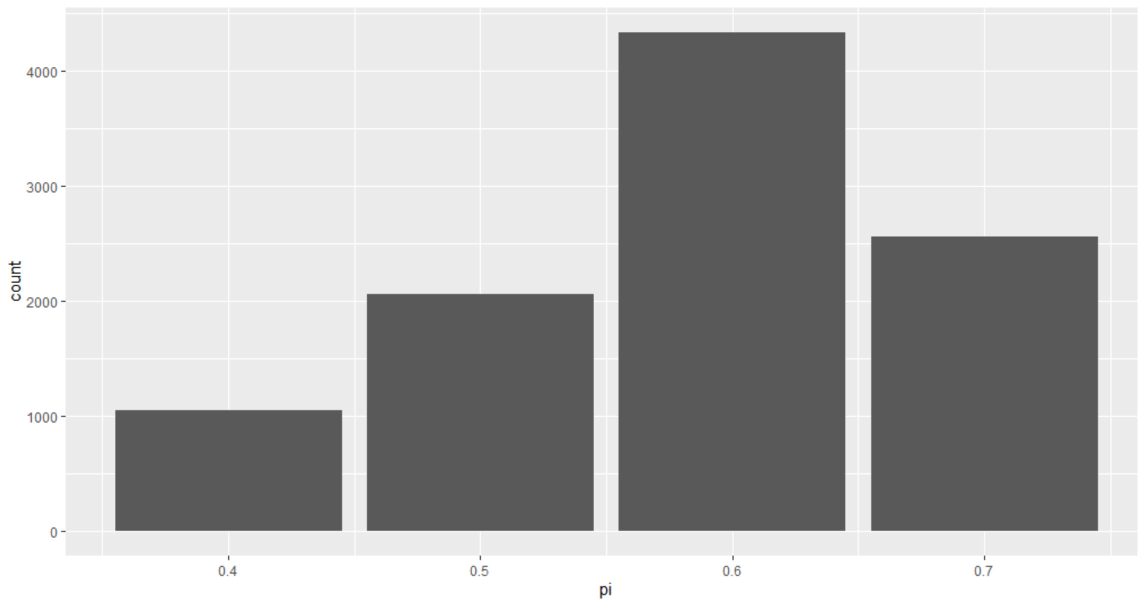


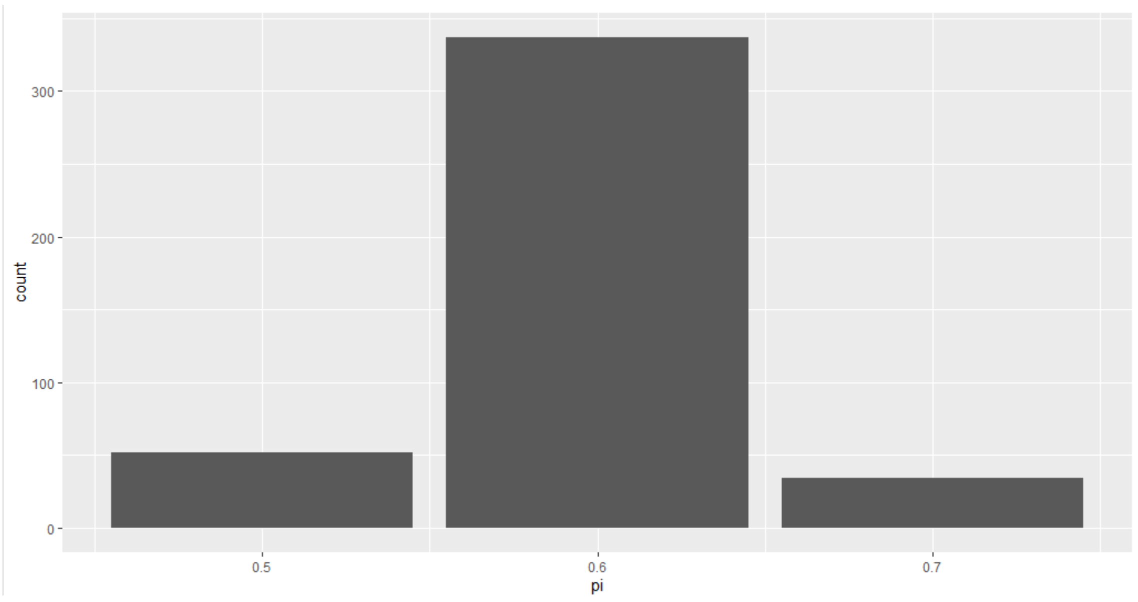
```
> # Focus on simulations with y = 47
> intolerance_47 <- intolerance_sim %>%
+   filter(y == 47)
> # Summarize the posterior approximation
> intolerance_47 %>%
+   tabyl(pi) %>%
+   adorn_totals("row")
     pi   n    percent
    0.5  52 0.12293144
    0.6 337 0.79669031
    0.7  34 0.08037825
  Total 423 1.00000000
> 
```

We can create the table to see the evolution of the prior and posterior values.

π	**0.4**	**0.5**	**0.6**	**0.7**	**Total**
f(π)	0.1	0.2	0.44	0.26	1.0
f(π|y=47)	0.0	0.12	0.80	0.08	1.0

Just to compare them, first we see the prior, and second, we see the posterior.





We can say that after knowing the value to be 47 out of the 80 people, the ‘π’ for the 0.6 went to almost 80%, while for the case of 0.4 went to 0% and for the 0.5 and 0.7 near 1%.

Now for the last part of the problem, for part C of 2.13, we are asked what happened if we have 800 adults and 470 had lactose intolerance, for that we need to update the numbers in the code to 800 in the ‘rbinom()’ and the 470 in the filter.

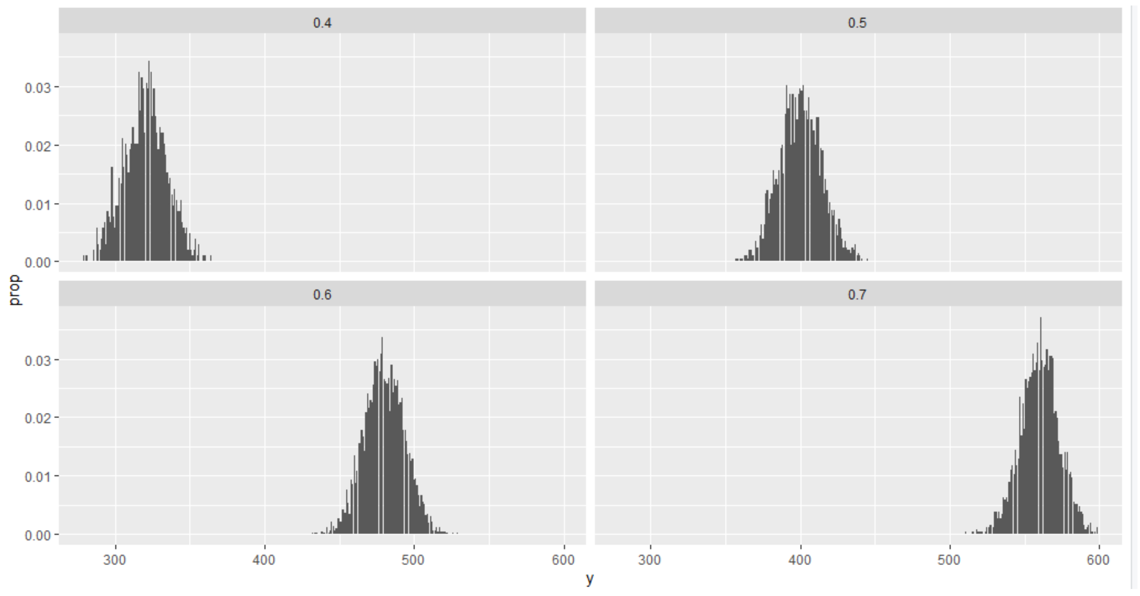
```
# Simulate 10000 random samples, of 80 people
intolerance_sim <- intolerance_sim %>%
  mutate(y = rbinom(10000, size = 800, prob = pi))

# Check it out
intolerance_sim %>%
  head(10)

# Summarize the prior
intolerance_sim %>%
  tabyl(pi) %>%
  adorn_totals("row")

# Plot y by pi
ggplot(intolerance_sim, aes(x = y)) +
  stat_count(aes(y = ..prop..)) +
  facet_wrap(~ pi)

# Focus on simulations with y = 470
intolerance_470 <- intolerance_sim %>%
  filter(y == 470)
```

We can see that the prior is the same distribution:

```
> # Summarize the prior
> intolerance_sim %>%
+   tabyl(pi) %>%
+   adorn_totals("row")
    pi     n percent
   0.4  1046  0.1046
   0.5  2060  0.2060
   0.6  4334  0.4334
   0.7  2560  0.2560
 Total 10000  1.0000
> 
```

Also, if we plot it again, we can see that the shape is the same, but the values are more spread. This is important because now there are no overlays.

```
> # Plot y by pi
> ggplot(intolerance_sim, aes(x = y)) +
+   stat_count(aes(y = ..prop..)) +
+   facet_wrap(~ pi)
> 
```



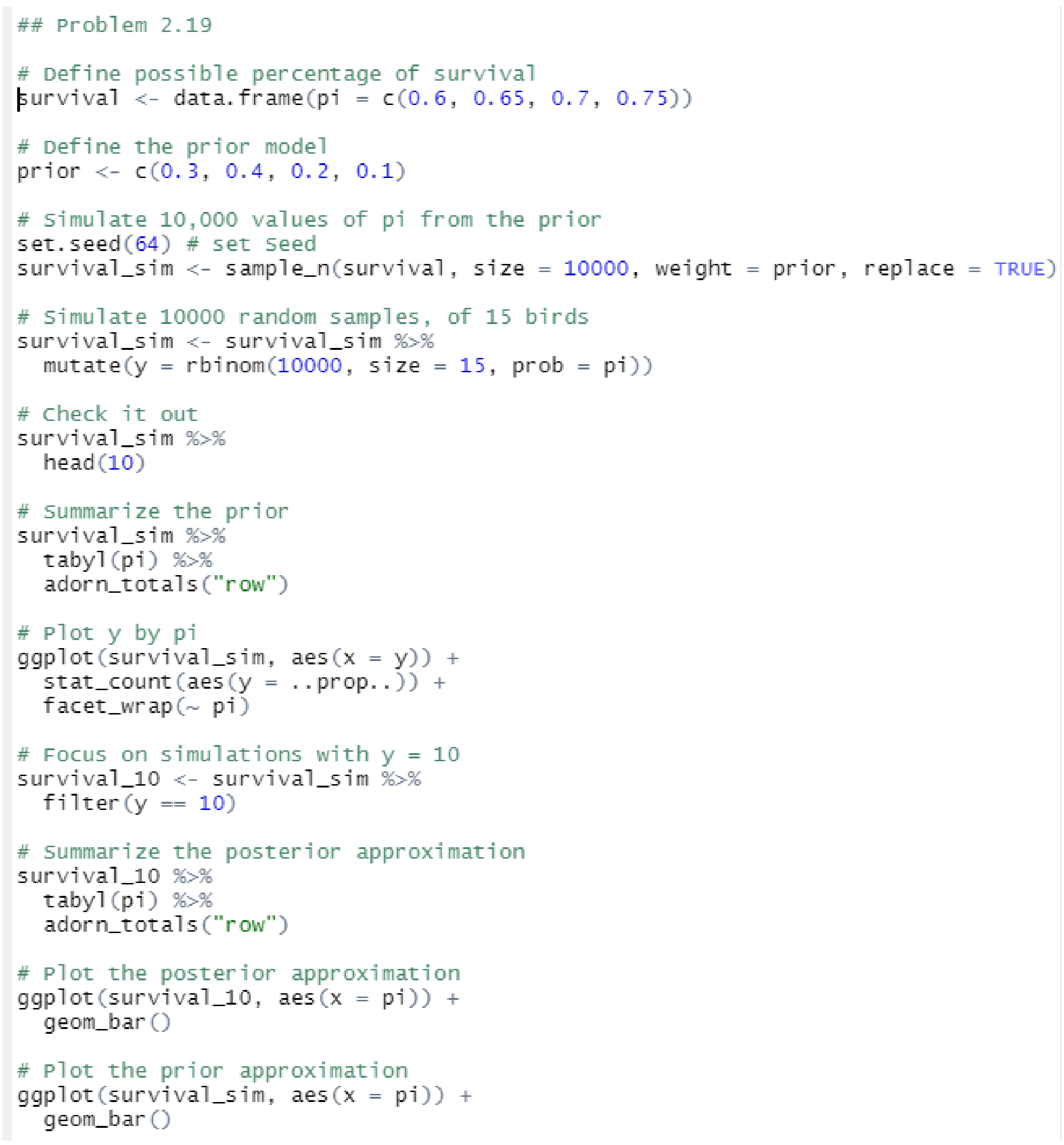
So now if we focus on the value of 470, we can see that there is only one ‘π’, exactly 0.6, while the probability for the rest of ‘π’ are down to 0%. These is because the values are way more spread and there is no overlay.

```
> # Focus on simulations with y = 470
> intolerance_470 <- intolerance_sim %>%
+   filter(y == 470)
> # Summarize the posterior approximation
> intolerance_470 %>%
+   tabyl(pi) %>%
+   adorn_totals("row")
    pi  n percent
   0.6 94       1
 Total 94       1
> 
```

Problem 2.

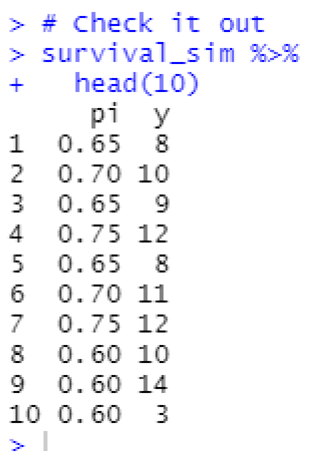
This is problem 2.19

For this problem we can run the same code changing some numbers, we need to modify the ‘π’ and the f(π), to fit the new values. This is done in the survival and prior variables. We also modify the size for the random numbers to 15, since Lisa only studied 15 birds. And finally, when we filter, we do it for the number 10, since 10 survived.



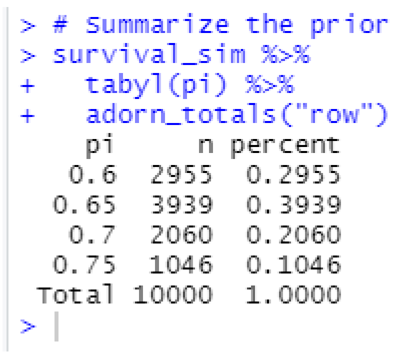
```r
## Problem 2.19

# Define possible percentage of survival
survival <- data.frame(pi = c(0.6, 0.65, 0.7, 0.75))

# Define the prior model
prior <- c(0.3, 0.4, 0.2, 0.1)

# Simulate 10,000 values of pi from the prior
set.seed(64) # set Seed
survival_sim <- sample_n(survival, size = 10000, weight = prior, replace = TRUE)

# Simulate 10000 random samples, of 15 birds
survival_sim <- survival_sim %>%
  mutate(y = rbinom(10000, size = 15, prob = pi))

# Check it out
survival_sim %>%
  head(10)

# Summarize the prior
survival_sim %>%
  tabyl(pi) %>%
  adorn_totals("row")

# Plot y by pi
ggplot(survival_sim, aes(x = y)) +
  stat_count(aes(y = ..prop..)) +
  facet_wrap(~ pi)

# Focus on simulations with y = 10
survival_10 <- survival_sim %>%
  filter(y == 10)

# Summarize the posterior approximation
survival_10 %>%
  tabyl(pi) %>%
  adorn_totals("row")

# Plot the posterior approximation
ggplot(survival_10, aes(x = pi)) +
  geom_bar()

# Plot the prior approximation
ggplot(survival_sim, aes(x = pi)) +
  geom_bar()
```

So now let’s see the different outputs, first we print the first 10 values after doing the ‘rbinom()’, we can see that the higher the ‘π’ the higher the number of surviving birds.



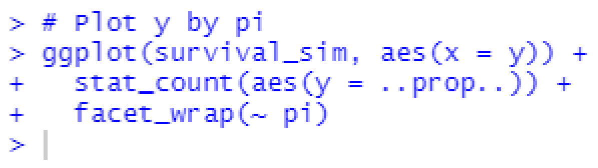
```
> # Check it out
> survival_sim %>%
+   head(10)
     pi  y
1  0.65  8
2  0.70 10
3  0.65  9
4  0.75 12
5  0.65  8
6  0.70 11
7  0.75 12
8  0.60 10
9  0.60 14
10 0.60  3
> 
```

We can check the prior values to see if they are correct according to the Problem. We can see that for ‘π’ of 0.6 a value near 0.3, for ‘π’ of 0.65 a value near 0.4, for ‘π’ of 0.7 a value near 0.2 and for ‘π’ of 0.75 a value near 0.1. So, the numbers are the same as in the ones given by the problem.

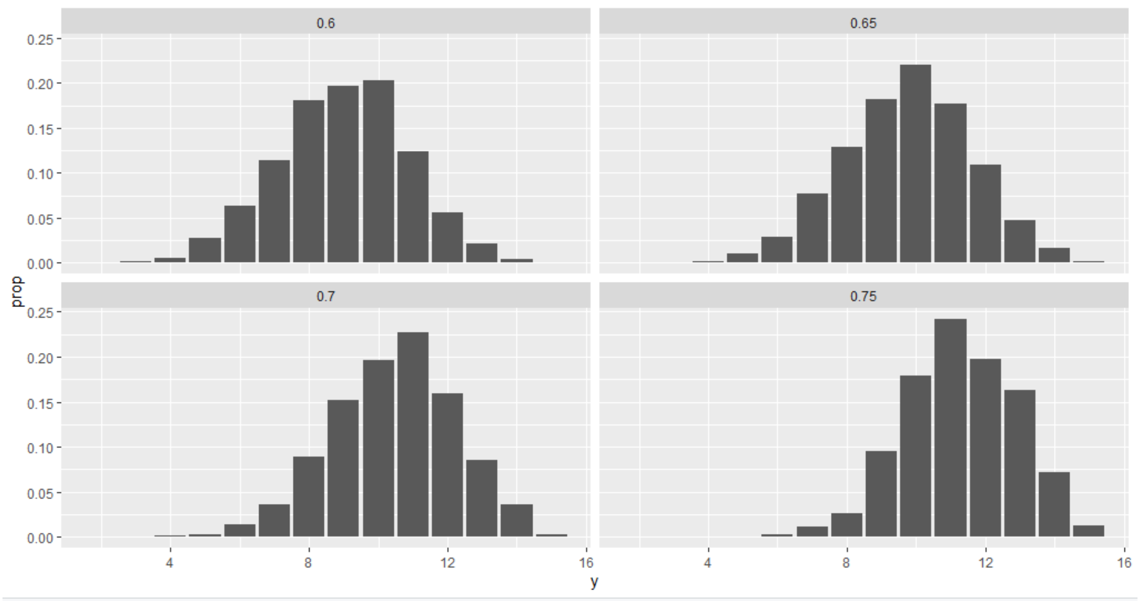


```
> # Summarize the prior
> survival_sim %>%
+   tabyl(pi) %>%
+   adorn_totals("row")
    pi     n percent
   0.6  2955  0.2955
  0.65  3939  0.3939
   0.7  2060  0.2060
  0.75  1046  0.1046
 Total 10000  1.0000
> 
```

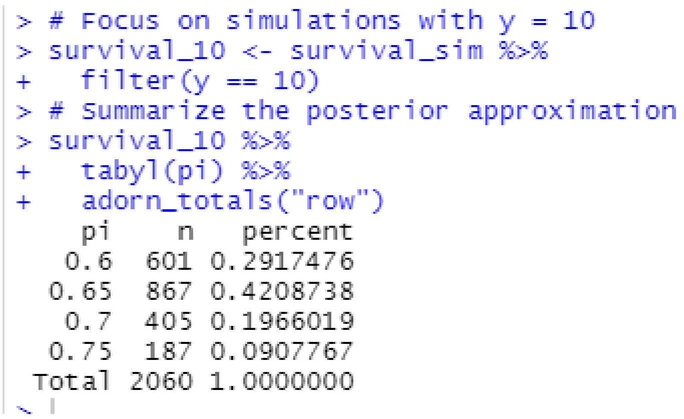
Now we can plot the different distribution for each of the different values of ‘π’. As well as in the previous exercise when the value of ‘π’ is lower the graph is cantered more to lower values.



```
> # Plot y by pi
> ggplot(survival_sim, aes(x = y)) +
+   stat_count(aes(y = ..prop..)) +
+   facet_wrap(~ pi)
> 
```



Now we can focus on the important case, when the number of surviving birds is 10. And we can see the posterior values.



```
> # Focus on simulations with y = 10
> survival_10 <- survival_sim %>%
+   filter(y == 10)
> # Summarize the posterior approximation
> survival_10 %>%
+   tabyl(pi) %>%
+   adorn_totals("row")
    pi    n   percent
   0.6  601 0.2917476
  0.65  867 0.4208738
   0.7  405 0.1966019
  0.75  187 0.0907767
 Total 2060 1.0000000
>
```

We can create the table to see the evolution of the prior and posterior values to make it more clear

π	**0.6**	**0.65**	**0.7**	**0.75**	**Total**
f(π)	0.3	0.4	0.2	0.1	1.0
f(π|y=47)	0.2917	0.4209	0.1966	0.0908	1.0

In this case the posterior values are really similar to the prior values. That is because the prior values make it more likely that the 'π' was 0.65, which is really similar to 10 birds surviving over 15, 10/15=0.66. That is why that probability got slightly higher while the rest got slightly smaller.