Chapter 3 – Sampling The Imaginary

3.1 Sampling from a grid-approximate posterior

- R Code 3.2:

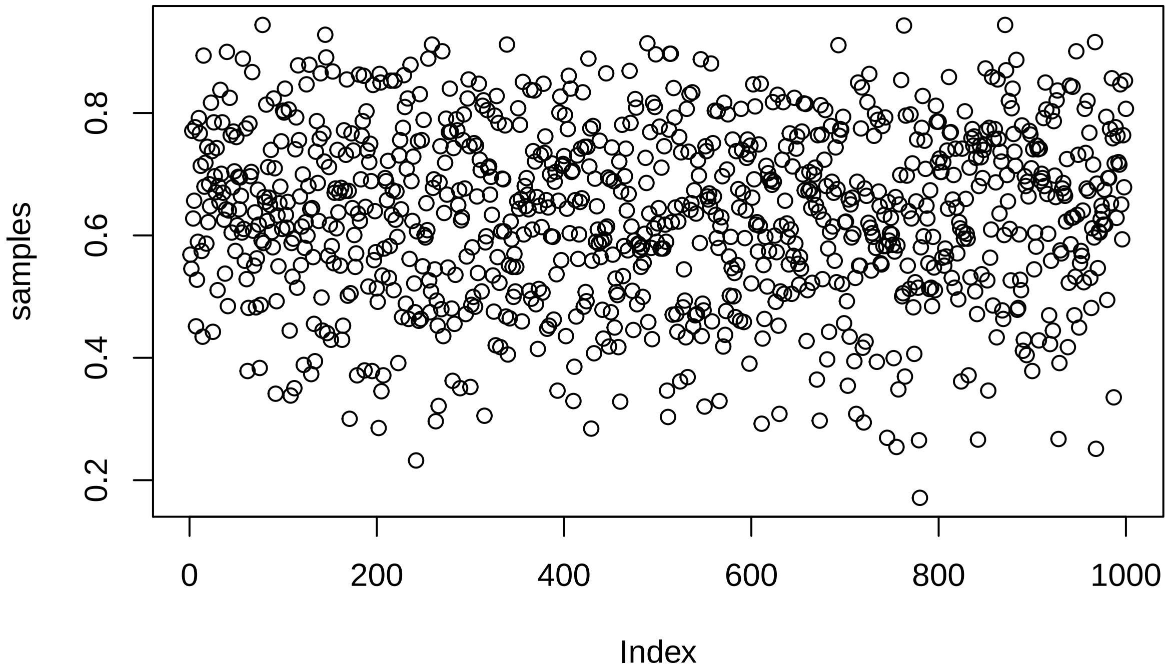
```
n = 1000
p_grid <- seq(from=0, to=1, length.out=n)
prior <- rep(1, n)
likelihood <- dbinom(x=6, size=9, prob=p_grid)
posterior_notnorm <- likelihood * prior
posterior <- posterior_notnorm / sum(posterior_notnorm)
```

Draw 10,000 samples: * R Code 3.3:

```
samples <- sample(p_grid, prob=posterior, size=n, replace=T)
```

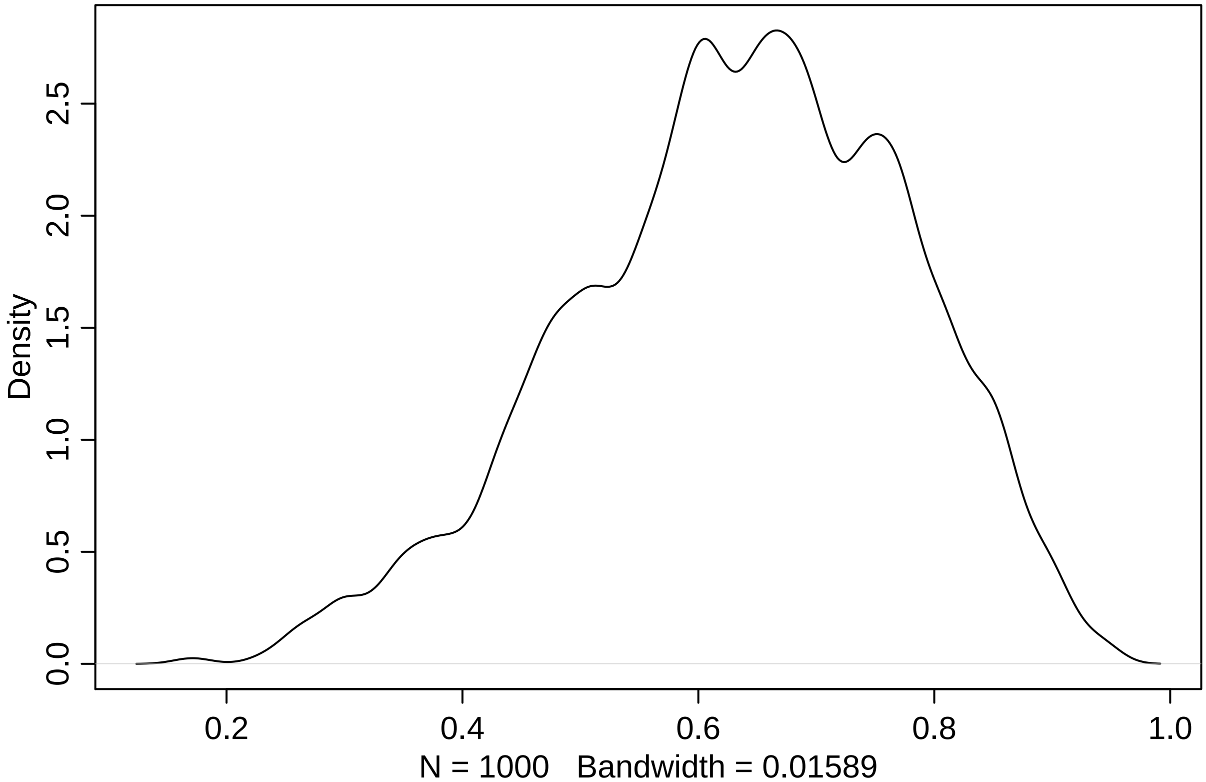
- 3.4:

```
plot(samples)
```



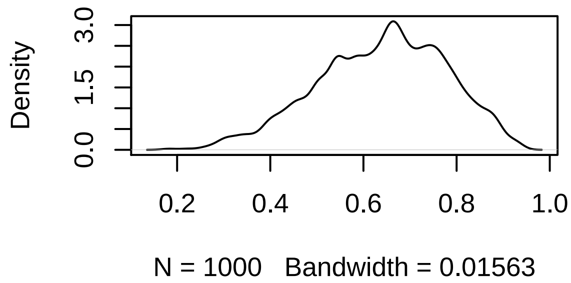
- 3.5:

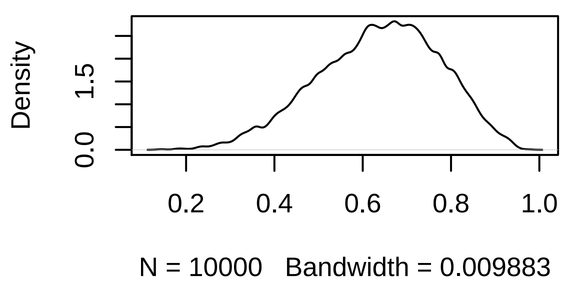
```
dens(samples)
```

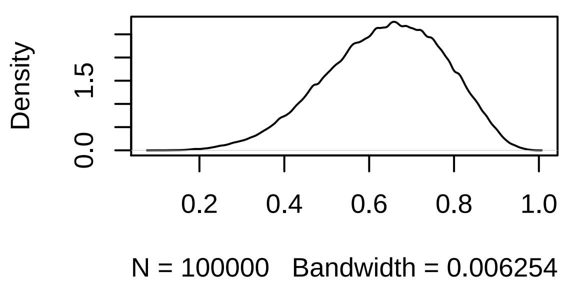


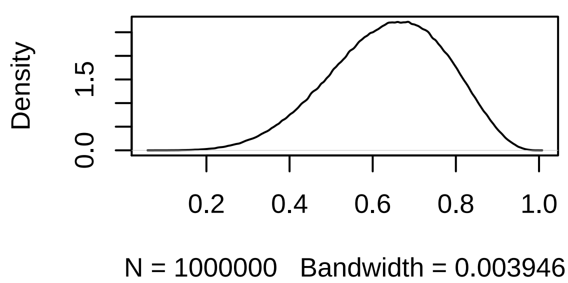
Let's try more samples:

```
par(mfrow=c(2, 2))
dens(sample(p_grid, prob=posterior, size=1e3, replace=T))
dens(sample(p_grid, prob=posterior, size=1e4, replace=T))
dens(sample(p_grid, prob=posterior, size=1e5, replace=T))
dens(sample(p_grid, prob=posterior, size=1e6, replace=T))
```









3.2 Sampling to Summarize

3.2.1. Intervals of defined boundaries.

The posterior probability that the proportion of water is less than 0.5:

- 3.6:

```
p_grid < 0.5
```

```
##    [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [34]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [45]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [56]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [67]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [78]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [89]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [100]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [111]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [122]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [144]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [155]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [166]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
##  [177]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [188]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [199]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [210]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [221]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [232]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [243]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [254]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [265]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [276]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [287]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [298]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [309]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [320]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [331]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [342]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [353]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [364]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [375]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [386]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [397]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [408]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [419]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [430]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [441]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [452]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [463]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [474]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [485]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [496]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [507] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [518] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [529] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [540] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [551] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [562] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [573] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [584] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [595] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [606] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [617] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [628] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [639] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [650] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [661] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [672] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [683] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [694] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [705] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [716] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [727] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [738] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [749] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [760] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [771] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [782] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [793] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [804] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [815] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [826] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [837] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [848] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [859] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [870] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [881] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [892] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [903] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [914] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [925] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [936] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [947] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [958] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [969] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [980] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [991] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
sum(posterior[p_grid < 0.5])
```

```
## [1] 0.1718746
```

Samples array:

```r
head(samples, 100)
```

```
##   [1] 0.5685686 0.5455455 0.7707708 0.6276276 0.6566567 0.7767768 0.4514515
##   [8] 0.5275275 0.5895896 0.7917918 0.7667668 0.7137137 0.5745746 0.4344344
##  [15] 0.8938939 0.6796797 0.7187187 0.5865866 0.7447447 0.6216216 0.6836837
##  [22] 0.6476476 0.8168168 0.7377377 0.4424424 0.7847848 0.6966967 0.6706707
##  [29] 0.7427427 0.5105105 0.6806807 0.6546547 0.8378378 0.7007007 0.7847848
##  [36] 0.6696697 0.6256256 0.5375375 0.6426426 0.8998999 0.4844845 0.6406406
##  [43] 0.8248248 0.7637638 0.6046046 0.6786787 0.7697698 0.7047047 0.5745746
##  [50] 0.7607608 0.6166166 0.6956957 0.6426426 0.6946947 0.6646647 0.6046046
##  [57] 0.8888889 0.6106106 0.5585586 0.7747748 0.5285285 0.3783784 0.4814815
##  [64] 0.7827828 0.6966967 0.7037037 0.8668669 0.6076076 0.5505506 0.6376376
##  [71] 0.4824825 0.5625626 0.6746747 0.6176176 0.3833834 0.4864865 0.5915916
##  [78] 0.9439439 0.5875876 0.6626627 0.6546547 0.8138138 0.6236236 0.7117117
##  [85] 0.6616617 0.6496496 0.7397397 0.6056056 0.5805806 0.8238238 0.7097097
##  [92] 0.3413413 0.4924925 0.6326326 0.5495495 0.6526527 0.6796797 0.7537538
##  [99] 0.6106106 0.8038038
```

The same calculation using samples. Add up all samples that lie in the grid < 0.5, and divide by the total number of samples to get the frequency ~ probability:

- 3.7:

```r
n = 1e4
samples = sample(p_grid, prob=posterior, size=n, replace=T)
sum(samples < 0.5) / n
```

```
## [1] 0.1652
```

How much probability lies between 0.5 and 0.75: * 3.8:

```r
sample_points = sum(samples > 0.5 & samples < 0.75)
sample_points
```

```
## [1] 6076
```

```r
sample_points / n
```

```
## [1] 0.6076
```

3.2.2. Intervals of defined mass.

Boundaries of the lower 80% posterior probability lies:

- 3.9:

```r
quantile(samples, probs = .8)
```

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

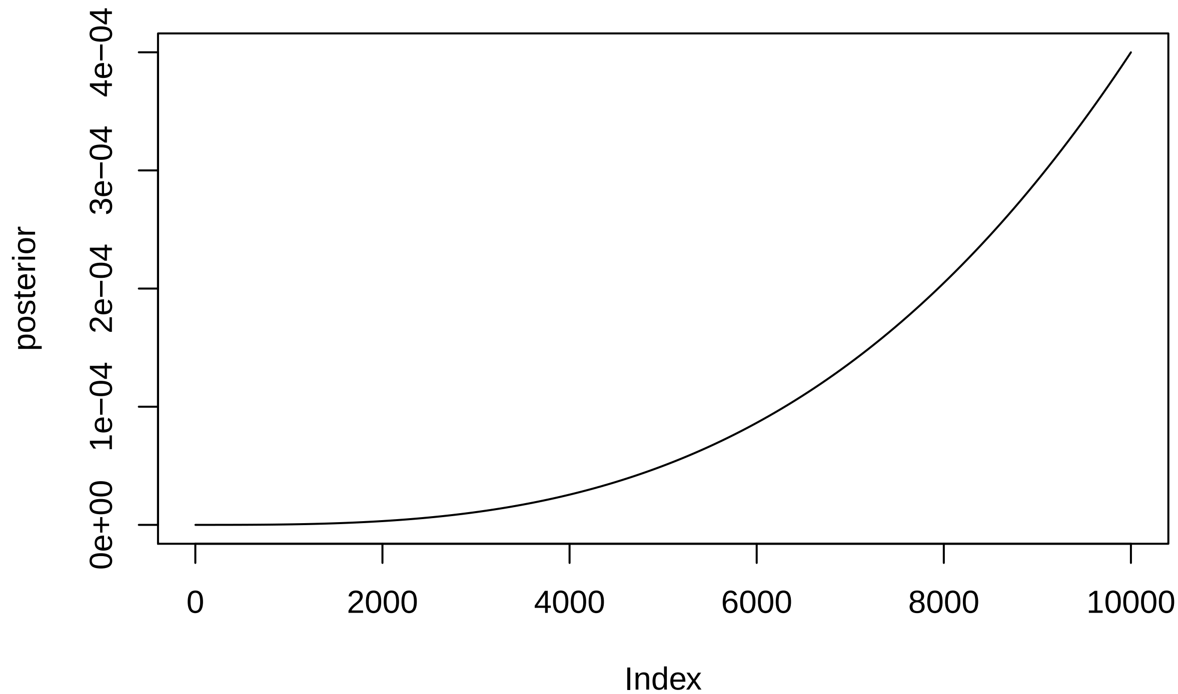
Middle 80%, i.e. lying between 10% and 90%:

```r
# 3.10
quantile(samples, probs = c(0.1, 0.9))
```

```
##       10%       90%
## 0.4514515 0.8148148
```

The above are PERCENTILE INTERVALS. Percentiles can be misleading if the distribution is highly skewed.

```r
# 3.11
n <- 10000
p_grid <- seq(0, 1, length.out = n)
prior <- rep(1, n)
likelihood <- dbinom(3, size=3, prob=p_grid)
posterior_notnorm <- likelihood * prior
posterior <- posterior_notnorm / sum(posterior_notnorm)
samples <- sample(p_grid, size=1e5, replace=T, prob=posterior)
plot(posterior, type='l')
```




```
# 3.12
PI(samples, prob=0.5)
```

```
##       25%       75%
## 0.7076708 0.9306931
```

Highest Posterior Density Interval described the distribution better. It's the *narrowest* interval containing the specified probability mass, e.g. 50%.

```
# 3.13
HPDI(samples, prob=0.5)
```

```
##     |0.5     0.5|
## 0.840484 1.000000
```

3.2.3. Point Estimates

A parameter with the highest posterior probability is called *a maximum a posteriori* estimate, or *MAP*.

```
# 3.14
which.max(posterior)
```

```
## [1] 10000
```

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

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

Use samples to get the same (or similar) result:

```r
# 3.15
chainmode(samples, adj=0.01)
```

```
## [1] 0.9948255
```

```r
# 3.16
mean(samples)
```

```
## [1] 0.7999697
```

```r
median(samples)
```

```
## [1] 0.840484
```

If the loss function is the absolute difference, then the posterior loss for p = 0.5 is

```r
# 3.17
sum(posterior * abs(0.5 - p_grid))
```

```
## [1] 0.3125375
```

```r
# 3.18
loss <- sapply(p_grid, function(d) sum(posterior * abs(d - p_grid)))
```

```r
# 3.19
which.min(loss)
```

```
## [1] 8410
```

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

```
## [1] 0.8409841
```

The posterior median minimizes the abs loss function. Let's test the quadratic loss function:

```r
loss2 <- sapply(p_grid, function(d) sum(posterior * (d - p_grid)^2))
which.min(loss2)
```

```
## [1] 8001
```

```r
p_grid[which.min(loss2)]
```

```
## [1] 0.80008
```

This is a mean.

3.3. Sampling to Simulate Prediction

3.3.1. Dummy Data

```r
# 3.20
dbinom(0:2, size=2, prob=0.7)
```

```
## [1] 0.09 0.42 0.49
```

We can sample from this distribution:

```r
# 3.22
rbinom(10, size=2, prob=0.7)
```

```
##  [1] 1 1 2 2 1 1 2 2 2 1
```

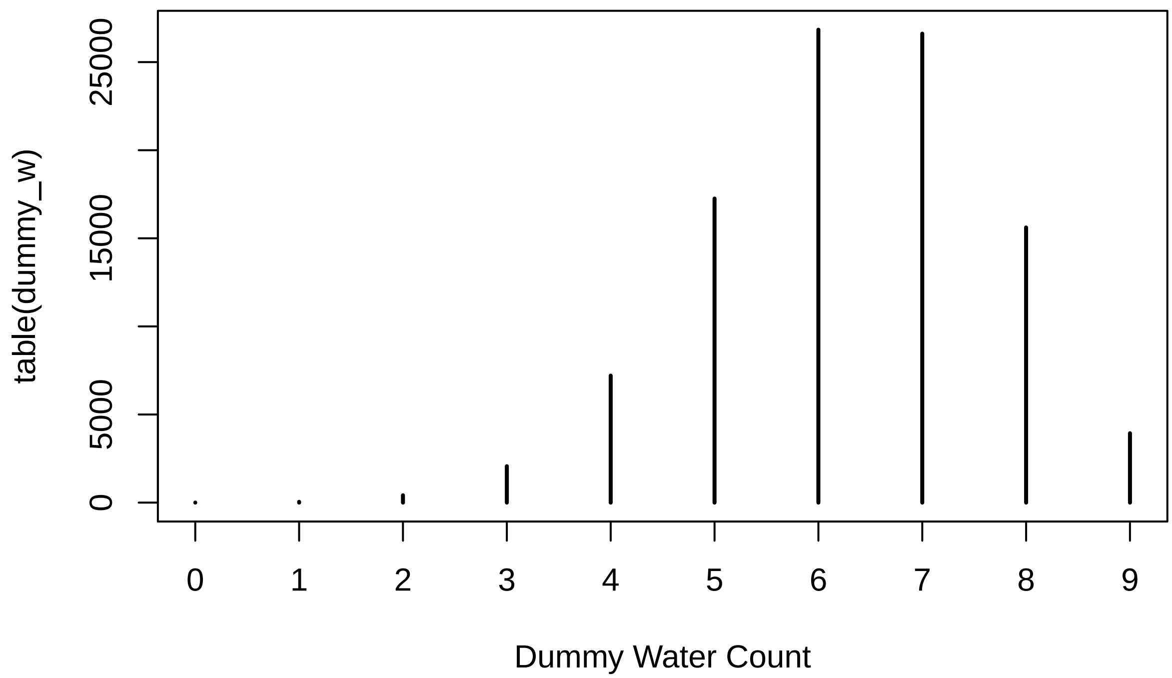
Let's generate 100,000 dummy observations to verify that each values 0, 1, and 2 appear in proportion to its likelihood:

```r
# 3.23
dummy_w <- rbinom(1e5, size=2, prob=0.7)
table(dummy_w) / 1e5
```

```
## dummy_w
##       0       1       2
## 0.08945 0.41974 0.49081
```

Let's simulate the sample with 9 tosses:

```r
# 3.24
dummy_w <- rbinom(1e5, size=9, prob=0.7)
plot(table(dummy_w), xlab="Dummy Water Count")
```




```r
table(dummy_w)
```

```
## dummy_w
##     0     1     2     3     4     5     6     7     8     9
##     2    46   417  2066  7206 17257 26838 26615 15616  3937
```

```r
dummy_w[1:100]
```

```
##   [1] 5 7 8 8 8 6 3 5 4 6 6 7 6 9 6 8 6 8 8 8 7 6 8 7 9 5 6 6 7 7 7 7 6 7 4
##  [36] 7 6 7 6 4 7 7 7 8 7 5 6 8 7 8 6 6 3 5 5 8 8 6 5 6 5 7 5 5 8 5 5 8 7 8
##  [71] 7 6 8 7 6 8 7 6 7 9 7 7 7 6 7 7 4 3 7 6 8 7 8 5 6 6 7 4 6 6
```