Chapter 3 – Sampling The Imaginary

3.1 Sampling from a grid-approximate posterior

- R Code 3.2:

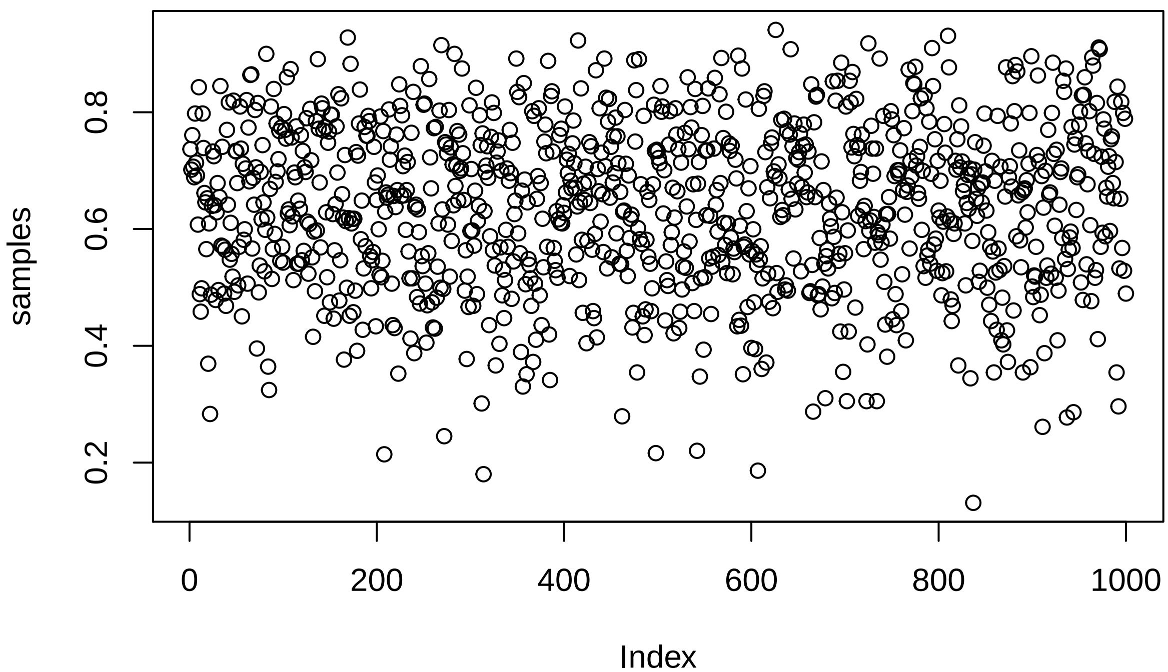
```
n = 1000
p_grid <- seq(from=0, to=1, length.out=n)
prior <- rep(1, n)
likelihood <- dbinom(x=6, size=9, prob=p_grid)
posterior_notnorm <- likelihood * prior
posterior <- posterior_notnorm / sum(posterior_notnorm)
```

Draw 10,000 samples: * R Code 3.3:

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

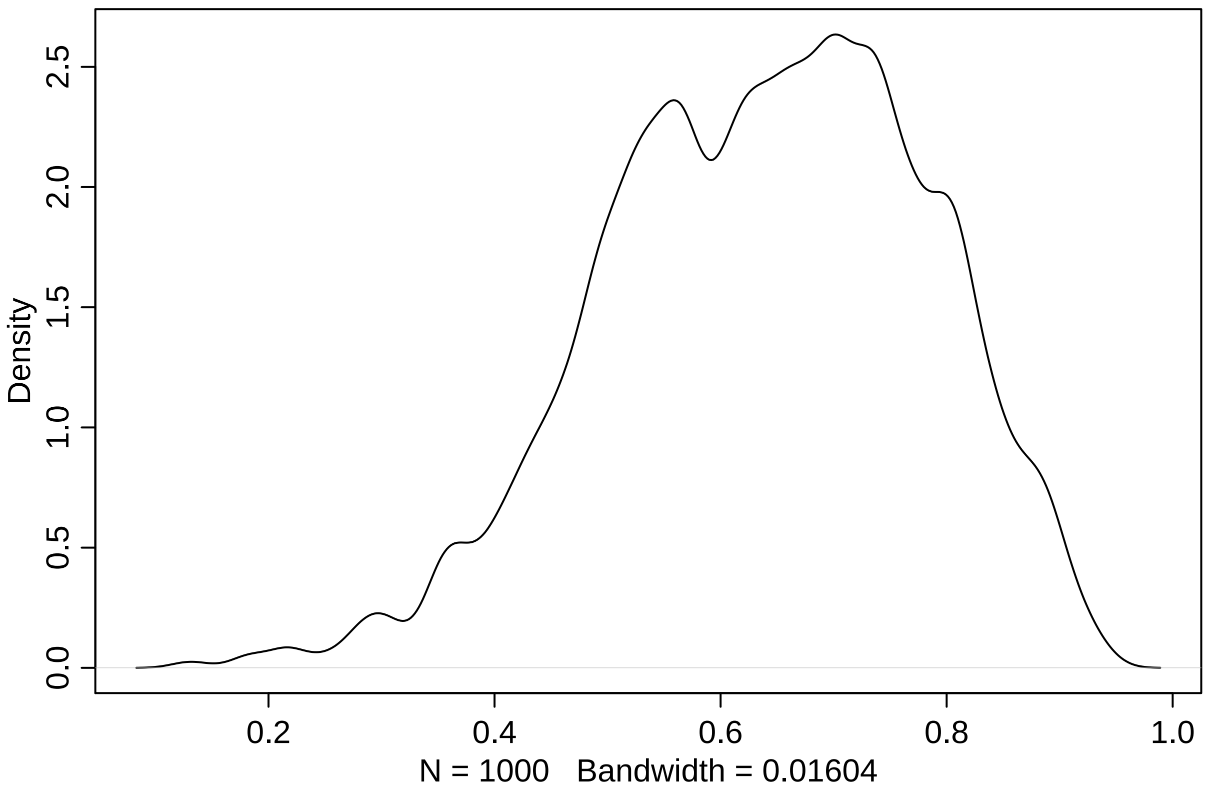
- 3.4:

```
plot(samples)
```



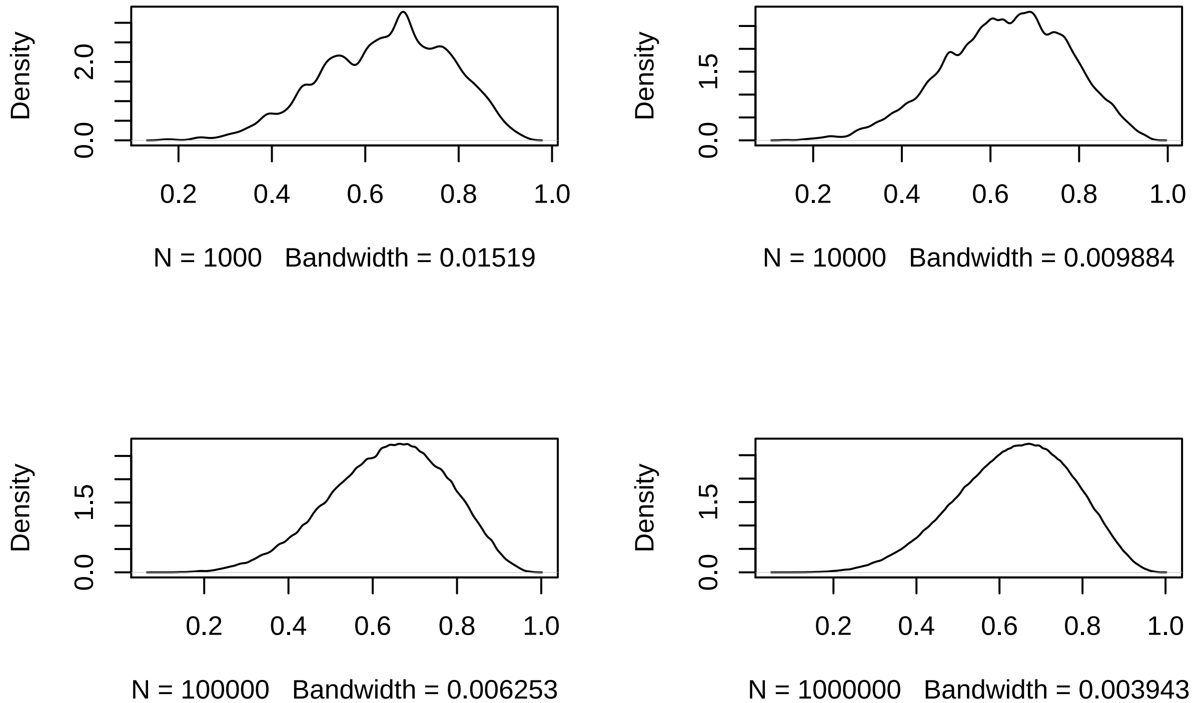
- 3.5:

```
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] 0.7367367 0.7017017 0.7607608 0.7067067 0.6886887 0.7977978 0.7137137
##   [8] 0.6916917 0.6076076 0.8428428 0.4884885 0.4584585 0.4984985 0.7977978
##  [15] 0.7387387 0.6616617 0.6456456 0.5655656 0.6526527 0.3693694 0.6096096
##  [22] 0.2832833 0.4874875 0.6396396 0.7327327 0.7247247 0.6406406 0.4784785
##  [29] 0.6286286 0.6786787 0.4954955 0.6546547 0.8448448 0.5715716 0.7407407
##  [36] 0.5695696 0.4894895 0.5645646 0.4684685 0.7697698 0.6416416 0.8168168
##  [43] 0.5475475 0.6106106 0.5575576 0.5185185 0.8198198 0.4924925 0.7337337
##  [50] 0.7327327 0.6786787 0.5035035 0.5695696 0.8098098 0.7377377 0.4504505
##  [57] 0.7117117 0.5815816 0.5995996 0.7037037 0.8208208 0.5065065 0.7737738
##  [64] 0.6816817 0.8638639 0.8648649 0.5665666 0.6876877 0.6416416 0.8038038
##  [71] 0.7057057 0.3953954 0.8158158 0.4914915 0.5375375 0.6976977 0.6166166
##  [78] 0.7437437 0.6456456 0.5275275 0.5985986 0.8998999 0.6196196 0.3643644
##  [85] 0.3243243 0.6686687 0.8098098 0.5145145 0.5655656 0.8398398 0.5915916
##  [92] 0.6806807 0.7807808 0.6986987 0.7197197 0.7677678 0.5455455 0.7737738
##  [99] 0.5695696 0.5425425
```

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] 0.1786
```

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

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

```
## [1] 6046
```

```
sample_points / n
```

```
## [1] 0.6046
```

3.2.2. Intervals of defined mass.

Boundaries of the lower 80% posterior probability lies:

- 3.9:

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

```
##       80%
## 0.7577578
```

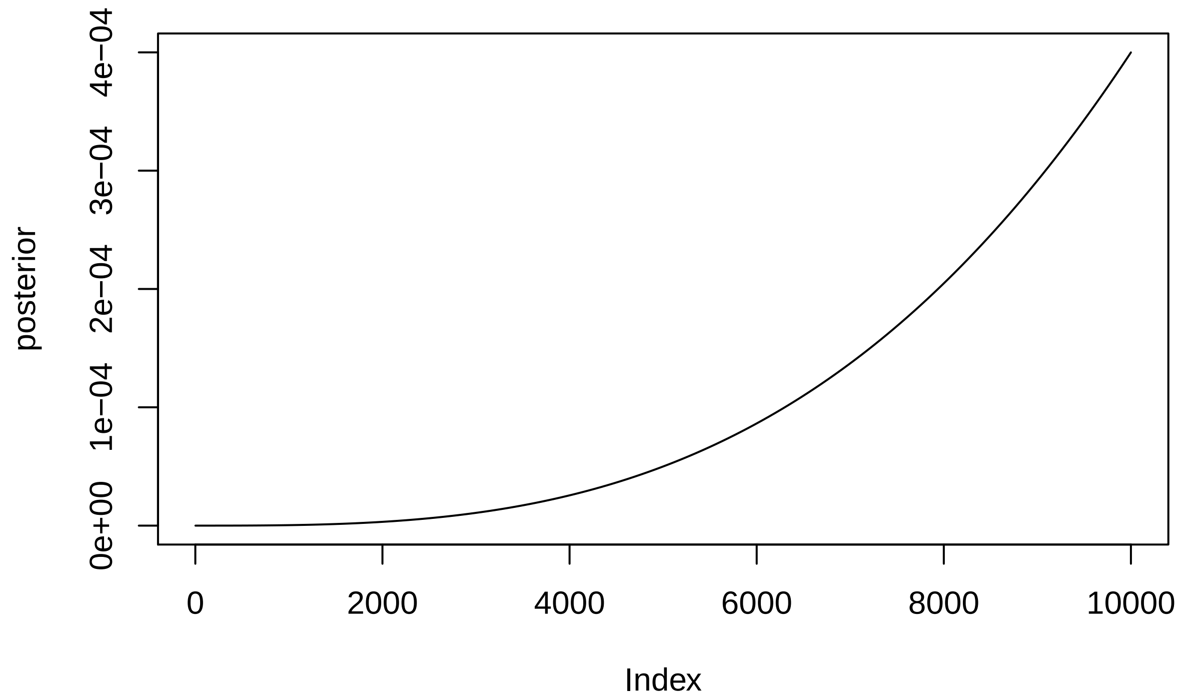
Middle 80%, i.e. lying between 10% and 90%:

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

```
##       10%       90%
## 0.4424424 0.8108108
```

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

```
# 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%
## 0.7084708 0.9311931
```

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|
## 0.8422842 1.0000000
```

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] 0.9966091
```

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

```
## [1] 0.8008572
```

```r
median(samples)
```

```
## [1] 0.8422842
```

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] 2 2 1 1 1 2 1 2 1 1
```

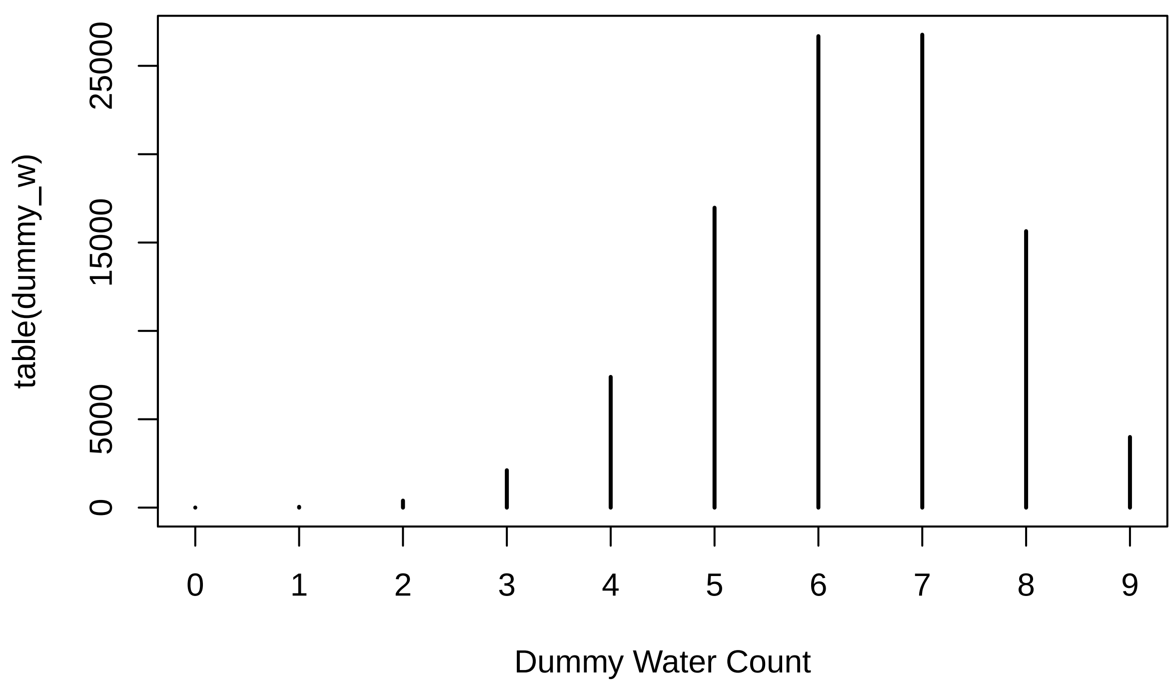
Let's generate 100,000 dummy observations to verify that each values 0, 1, and 2 appear in proportion to its likelihood:

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

```
## dummy_w
##       0       1       2
## 0.09276 0.41841 0.48883
```

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
##     4    44   395  2113  7395 16970 26680 26761 15644  3994
```

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

```
##   [1] 7 3 5 6 2 7 6 5 4 8 7 5 7 9 6 4 5 5 6 6 6 6 7 7 5 4 8 6 4 8 5 4 6 5 4
##  [36] 7 6 5 7 8 7 6 5 7 6 8 5 7 7 8 8 6 9 6 6 7 6 5 2 6 8 9 7 7 5 4 6 7 6 5
##  [71] 7 6 4 7 6 7 9 5 7 8 6 6 4 7 6 6 6 5 6 7 5 6 7 6 5 9 7 8 7 6
```