Chapter 3 – Sampling The Imaginary

3.1 Sampling from a grid-approximate posterior

- R Code 3.2:

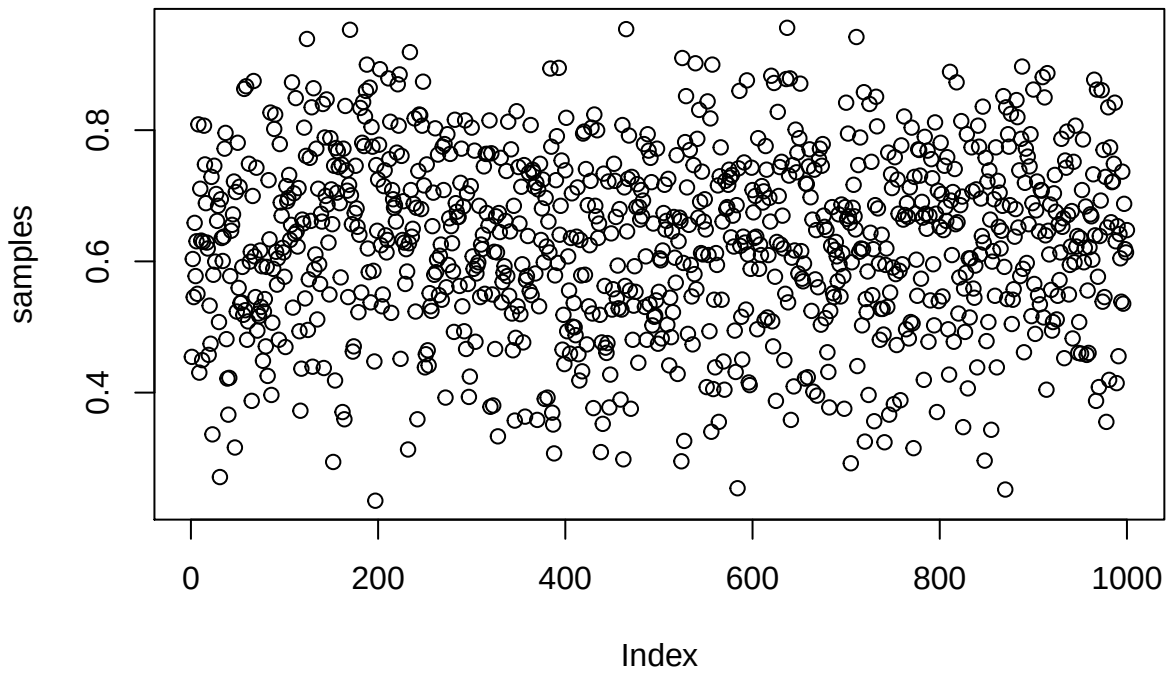
```
n = 1000
p_grid <- seq(from=0, to=1, length.out=n)
prior <- rep(1, n)
likelihood <- dbinom(x=6, size=9, prob=p_grid)
posterior_notnorm <- likelihood * prior
posterior <- posterior_notnorm / sum(posterior_notnorm)
```

Draw 10,000 samples: * R Code 3.3:

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

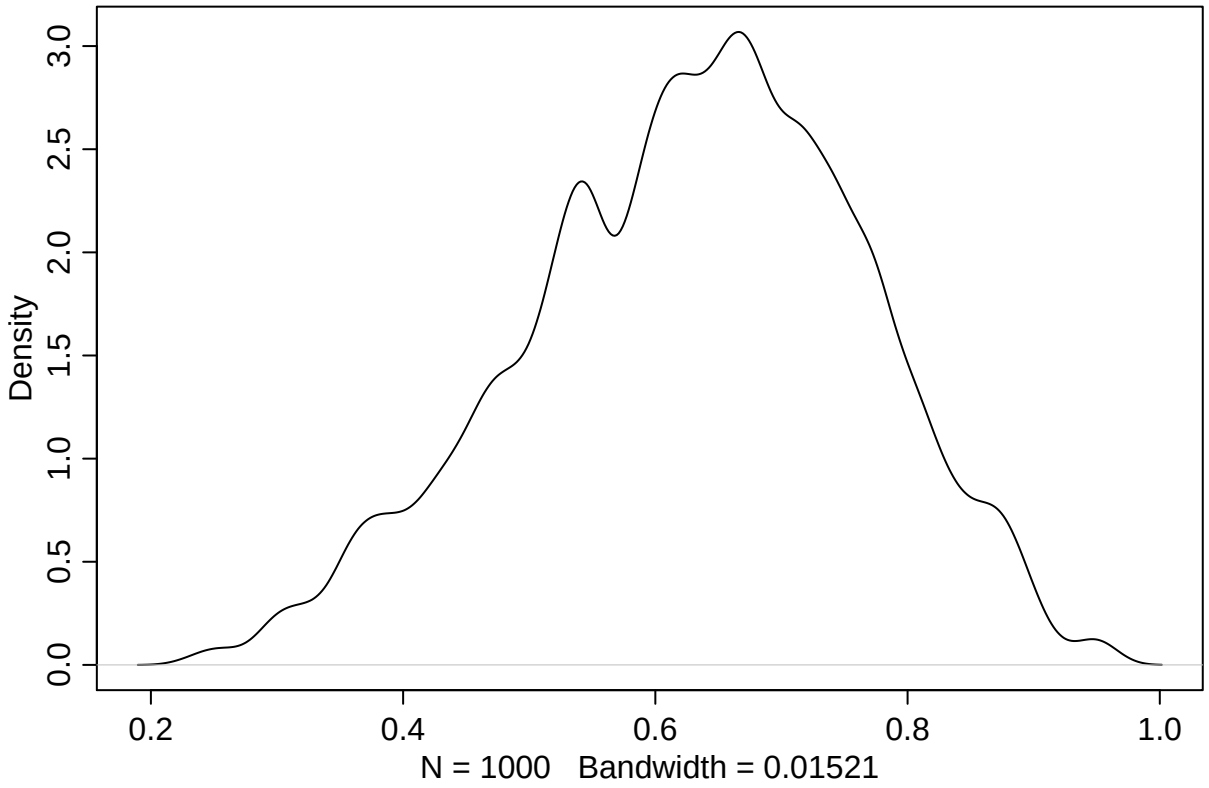
- 3.4:

```
plot(samples)
```



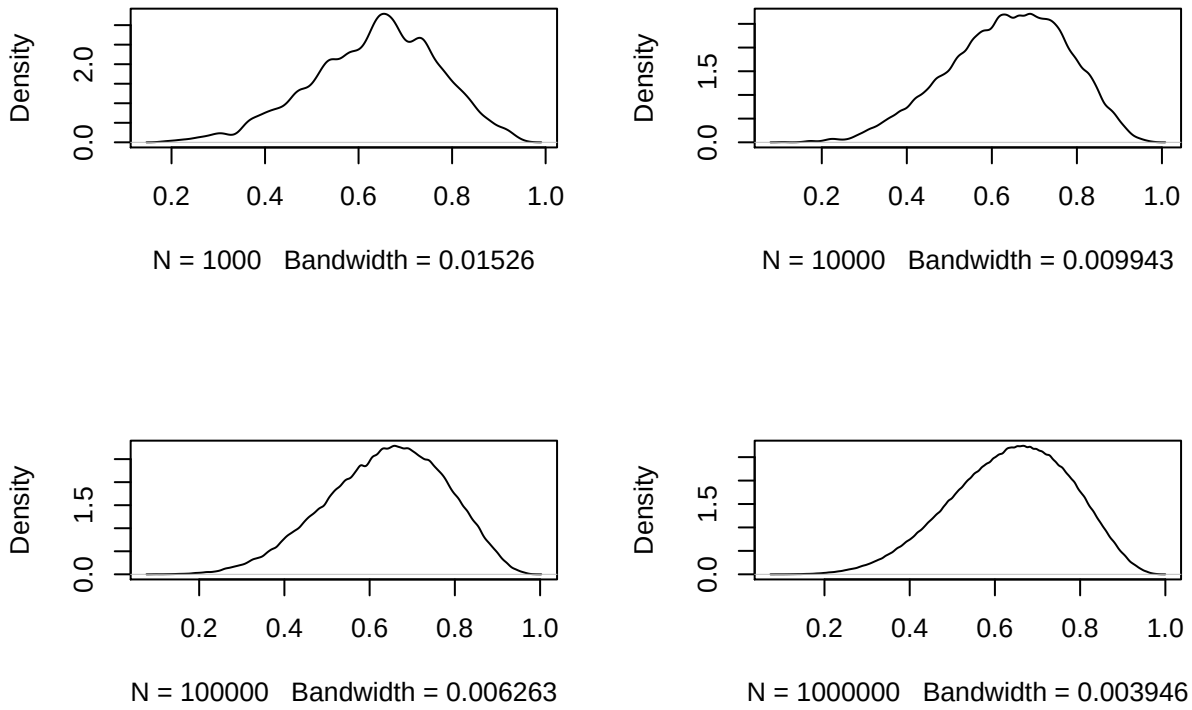
- 3.5:

```
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
```

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

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

Samples array:

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

```
##   [1] 0.4544545 0.6036036 0.5455455 0.6586587 0.5775776 0.6306306 0.5505506
##   [8] 0.8088088 0.4304304 0.7107107 0.6316316 0.4494494 0.6296296 0.8068068
##  [15] 0.7477477 0.6886887 0.6286286 0.6196196 0.4574575 0.5325325 0.4744745
##  [22] 0.7287287 0.3363363 0.5795796 0.7457457 0.5995996 0.7027027 0.6616617
##  [29] 0.6846847 0.5075075 0.2712713 0.6946947 0.6356356 0.6006006 0.6376376
##  [36] 0.7717718 0.7957958 0.4814815 0.4214214 0.3663664 0.4224224 0.5775776
##  [43] 0.6466466 0.6556557 0.6716717 0.7217217 0.3163163 0.7057057 0.5235235
##  [50] 0.7807808 0.5595596 0.7137137 0.5365365 0.5375375 0.5915916 0.5185185
##  [57] 0.8628629 0.5255255 0.8668669 0.4804805 0.5085085 0.7487487 0.5995996
##  [64] 0.6146146 0.3873874 0.6996997 0.8748749 0.6076076 0.5455455 0.7427427
##  [71] 0.4944945 0.5205205 0.5165165 0.6166166 0.5305305 0.5925926 0.4484484
##  [78] 0.5235235 0.5425425 0.4704705 0.5915916 0.4254254 0.7237237 0.6336336
##  [85] 0.8268268 0.3963964 0.5065065 0.5885886 0.8018018 0.8238238 0.6146146
##  [92] 0.5645646 0.6046046 0.4804805 0.7787788 0.6896897 0.6696697 0.6206206
##  [99] 0.6136136 0.5765766
```

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 $\sim$ probability:

- 3.7:

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

```
## [1] 0.1705
```

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

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

```
## [1] 6044
```

```
sample_points / n
```

```
## [1] 0.6044
```

3.2.2. Intervals of defined mass.

Boundaries of the lower 80% posterior probability lies:

- 3.9:

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

```
##       80%
## 0.7607608
```

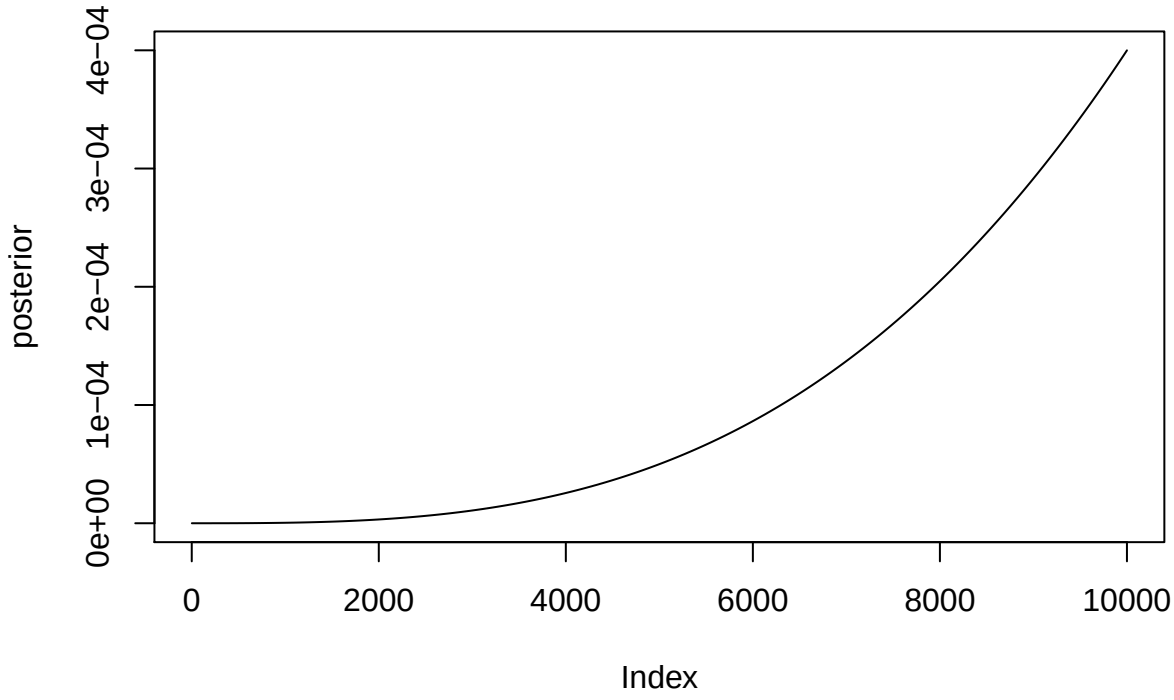
Middle 80%, i.e. lying between 10% and 90%:

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

```
##       10%       90%
## 0.4484484 0.8128128
```

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%
## 0.7099710 0.9314931
```

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|
## 0.8427843 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:

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

```
## [1] 0.9894167
```

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

```
## [1] 0.801751
```

```
median(samples)
```

```
## [1] 0.8427843
```

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

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

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

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

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

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

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

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

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

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

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

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

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

This is a mean.

3.3. Sampling to Simulate Prediction

3.3.1. Dummy Data

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

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

We can sample from this distribution:

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

```
##  [1] 1 1 2 1 1 0 1 2 2 1
```

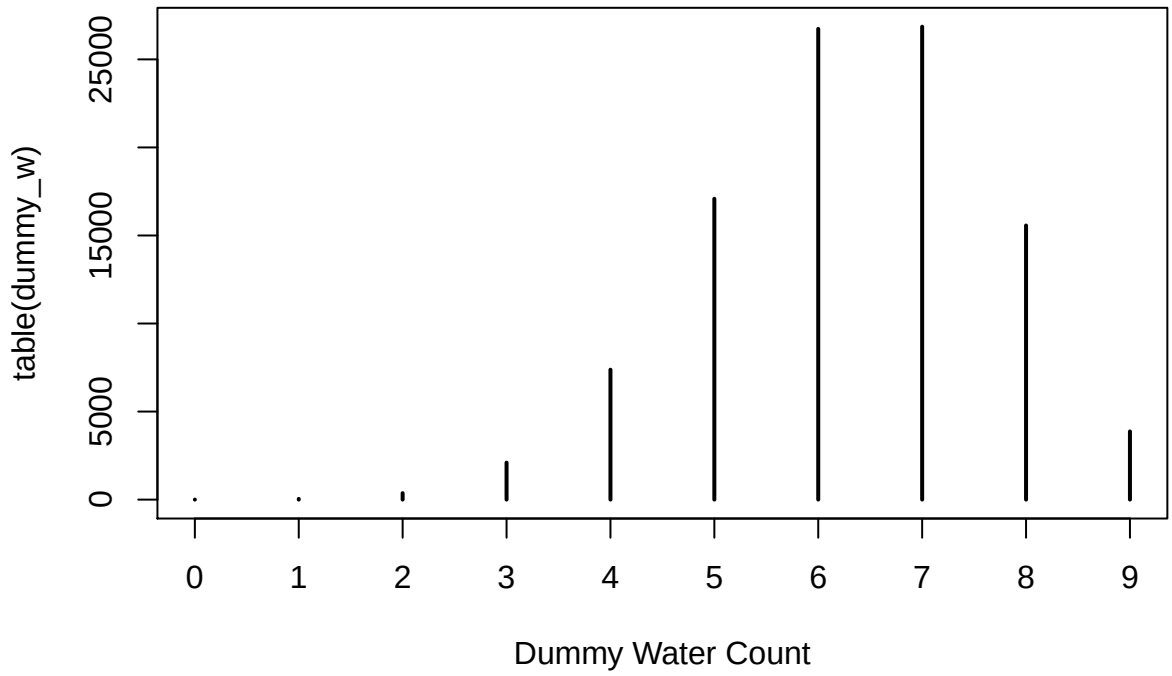
Let's generate 100,000 dummy observations to verify that each values 0, 1, and 2 appear in proportion to its likelihood:

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

```
## dummy_w
##       0       1       2
## 0.08912 0.42157 0.48931
```

Let's simulate the sample with 9 tosses:

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




```
table(dummy_w)
```

```
## dummy_w
##     0     1     2     3     4     5     6     7     8     9
##     2    40   367  2100  7378 17085 26733 26855 15567  3873
```

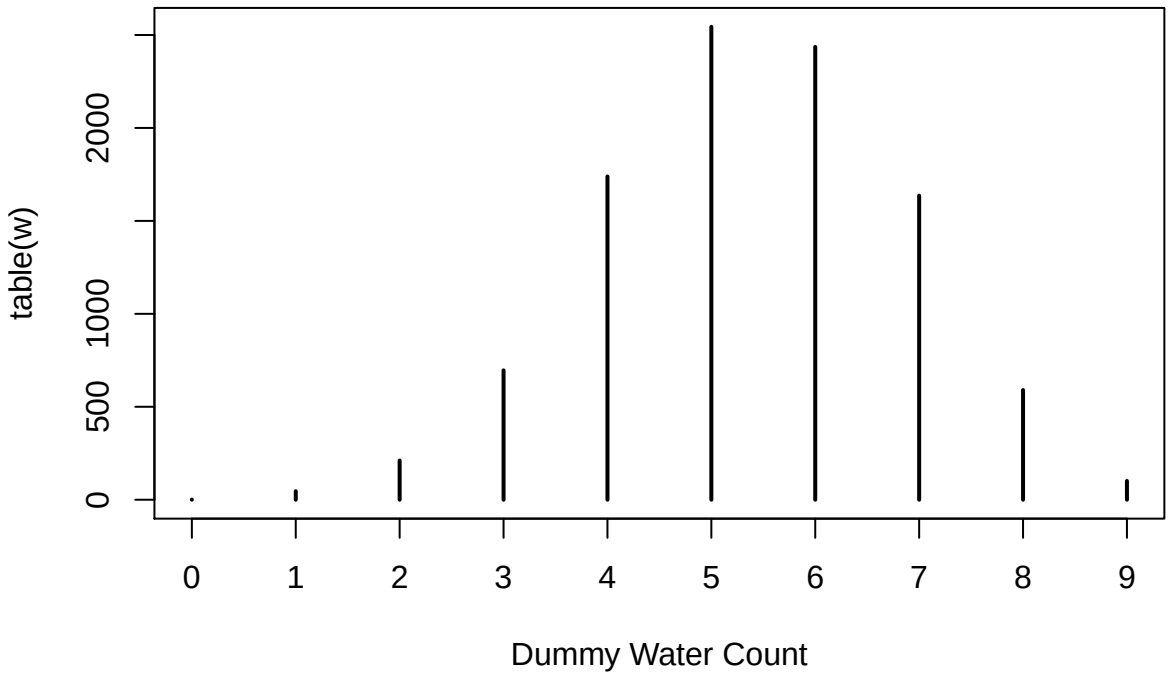
```
dummy_w[1:100]
```

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

3.2. Model Checking

Below is a misleading distribution plot. While p = 0.6 is the likeliest estimate, if we simply use it as a point estimate we will obtain a much more narrow, "overly confident" predictions:

```
# 3.25
w <- rbinom(1e4, size=9, prob=0.6)
plot(table(w), xlab="Dummy Water Count")
```



The correct way to generate predictions is to incorporate our uncertainty about p. We can do it by using sampled values of p, and averaging over all of them. The sampled values will appear with the right frequency, described by our posterior. Samples from the *posterior* distribution of p:

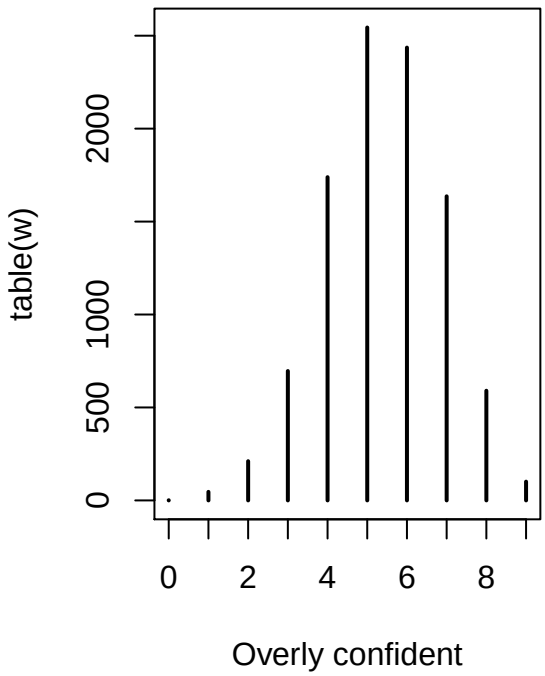
```
samples[1:20]
```

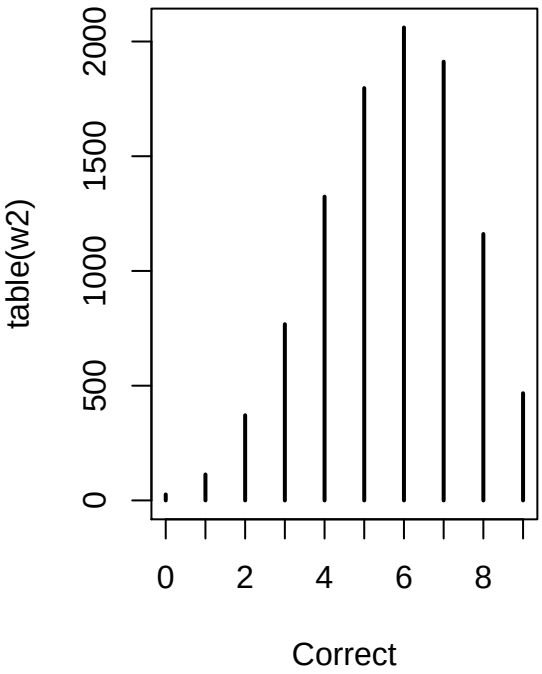
```
##  [1] 0.9022902 0.2151215 0.7202720 0.9247925 0.9942994 0.7457746 0.8035804
##  [8] 0.8390839 0.8042804 0.6967697 0.8024802 0.7568757 0.9796980 0.6420642
## [15] 0.9236924 0.8105811 0.6538654 0.7180718 0.4230423 0.6872687
```

```
# 3.26
w2 <- rbinom(1e4, size=9, prob=samples_orig)
table(w2)
```

```
## w2
##    0    1    2    3    4    5    6    7    8    9
##   26  113  371  768 1324 1797 2061 1912 1161  467
```

```
par(mfrow=c(1, 2))
plot(table(w), xlab ="Overly confident")
plot(table(w2), xlab="Correct")
```


table(w)
0
500
1000
2000
0
2
4
6
8
Overly confident


table(w2)
0
500
1000
1500
2000
0
2
4
6
8
Correct