MATH 3070 Lab Project 12

Your Name

November 16, 2016

Contents

Remember: I expect to see commentary either in the text, in the code with comments created using `#`*, or (preferably) both!* ***Failing to do so may result in lost points!***

Because randomization is used in this assignment, I set the seed here, in addition to beginning each code block. ***Do not change the seed!***

```
set.seed(6222016)
```

Problem 1

The data set `DDT` *(**MASS**) contains measurements of the pesticide DDT in kale, in parts per million. Use bootstrapping to estimate a 95% confidence interval for the mean ppm of DDT in kale. Do the same with the standard deviation. Use 2000 replications each.*

```
##     2.5%    97.5%
## 3.145317 3.572717
```

```
# Your code here
```

Problem 2

An inspector receives a batch of widgets which will be used to manufacture a new product. The batch will be rejected and sent back to the manufacturer if the proportion of defective widgets in the batch exceeds 10%. The inspector selected thirty widgets from the batch and tested them. He found that of the thirty widgets he tested, two were defective.

Let the proportion of defective widgets in the batch, $p \in \{0.1, 0.2, 0.3, ..., 0.9\}$, and assign a uniform prior to p (that is, according to the prior distribution, $P(p = p_i) = P(p = p_j)$ for every $p_i, p_j \in \{0.1, 0.2, 0.3, ..., 0.9\}$; in words, each p is equally likely). Given the results of the inspection:

1. *Compute the posterior distribution of p. Plot both the prior and posterior distribution for p. (You may borrow code from the lecture as appropriate; for instance, I suggest using my function* `plot_pmf()`*.)*

```r
# Your code here
# plotting function code
# I will be plotting a lot of pmf's in this document, so I create a function
# to help save effort.  The first argument, q, represents the quantiles of
# the random variable (the values that are possible). The second argument
# represents the value of the pmf at each q (and thus should be of the same
# length); in other words, for each q, p is the probability of seeing q
plot_pmf <- function(q, p, main = "pmf") {
    # This will plot a series of horizontal lines at q with height p, setting
    # the y limits to a reasonable heights
    plot(q, p, type = "h", xlab = "x", ylab = "probability", main = main, ylim = c(0,
        max(p) + 0.1))
    # Usually these plots have a dot at the end of the line; the point function
    # will add these dots to the plot created above
    points(q, p, pch = 16, cex = 1.5)
}

# Change plot settings
old_par <- par()
par(mfrow = c(2, 1))

# Possible probabilities
(p <- seq(0, 1, by = 0.1))
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```r
# The prior distribution of p
(prior <- rep(1/length(p), times = length(p)))
```

```
##  [1] 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
##  [7] 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
```

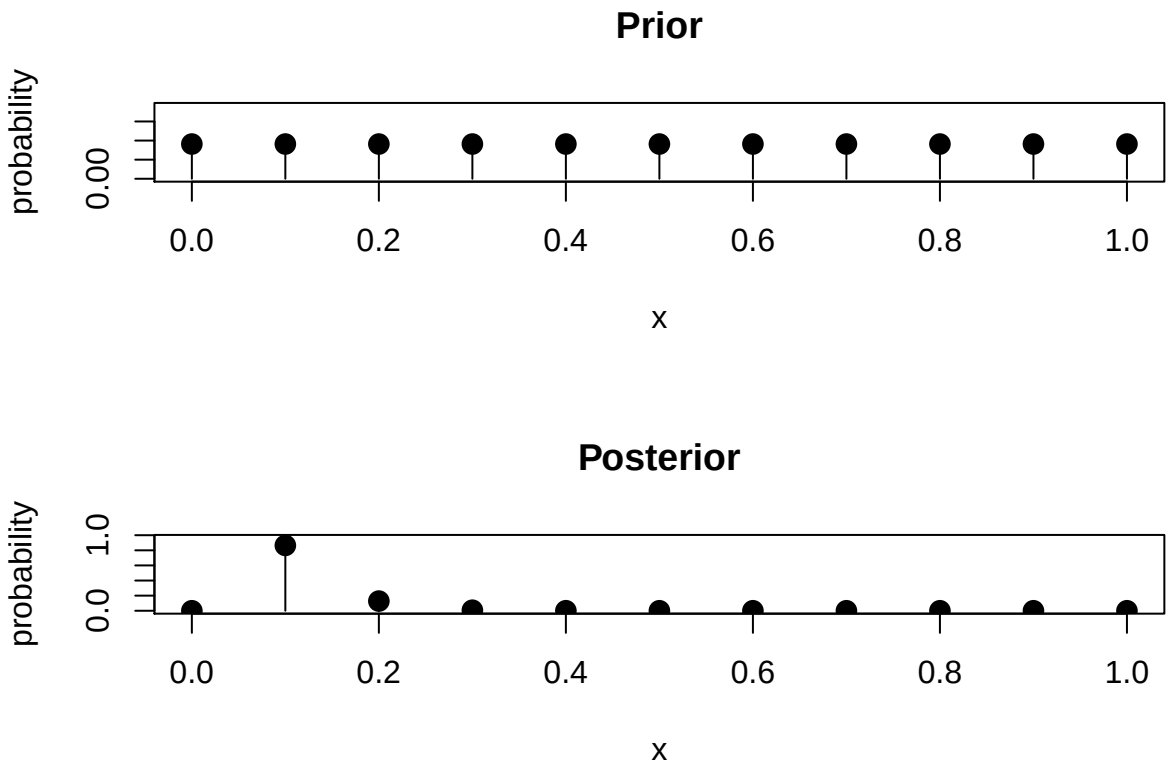
```r
# Plot the prior
plot_pmf(p, prior, main = "Prior")

# Compute the likelihood function
n <- 30  # Number of widgets
s <- 2  # Number of defective widgets
(like <- p^s * (1 - p)^(n - s))
```

```
##  [1] 0.000000e+00 5.233476e-04 7.737125e-05 4.139879e-06 9.825508e-08
##  [6] 9.313226e-10 2.594073e-12 1.120963e-15 1.717987e-20 8.100000e-29
## [11] 0.000000e+00
```

```r
# With the likelihood and prior computed, we now obtain the posterior
# distribution.
post <- like * prior
(post <- post/sum(post))  # Normalize; make proper probabilities
```

```
##  [1] 0.000000e+00 8.650975e-01 1.278953e-01 6.843251e-03 1.624164e-04
##  [6] 1.539483e-06 4.288023e-09 1.852960e-12 2.839845e-17 1.338936e-25
## [11] 0.000000e+00
```

```
plot_pmf(p, post, main = "Posterior")
```



2. *What is the posterior probability that there are too many defective chips in the batch (that is, $p > .1$)?*

```
# Your code here
sum(post[which(p > .1)])
```

```
## [1] 0.1349025
```

3. *Compute the maximum* a-posteriori *(MAP) estimator for p and a (approximately) 95% credible interval for p using the posterior distribution.*

```
# Your code here

# Getting the MAP estimator for p
p[which.max(post)]
```

```
## [1] 0.1
```

```
# get the cdf
(post_cdf <- cumsum(post))
```

```
##  [1] 0.0000000 0.8650975 0.9929928 0.9998360 0.9999985 1.0000000 1.0000000
##  [8] 1.0000000 1.0000000 1.0000000 1.0000000
```

```r
(a <- p[max(which(post_cdf < 0.025))])
```

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

```r
(b <- p[min(which(post_cdf > 0.975))])
```

```
## [1] 0.2
```

```r
sum(post[which(p >= a & p <= b)])
```

```
## [1] 0.9929928
```

4. *Repeat steps 1-3, but instead of* $p \in \{0.1, 0.2, 0.3, ..., 0.9\}$*, let* $p \in \{0.05, 0.1, 0.15, ..., 0.95\}$*. Do your conclusions from earlier change? How does the new 95% credible interval compare? What does increasing the resolution of possible p values do to the posterior distribution? Which do you prefer?*

```r
# Your code here
# Your code here
# plotting function code
# I will be plotting a lot of pmf's in this document, so I create a function
# to help save effort.  The first argument, q, represents the quantiles of
# the random variable (the values that are possible). The second argument
# represents the value of the pmf at each q (and thus should be of the same
# length); in other words, for each q, p is the probability of seeing q
plot_pmf <- function(q, p, main = "pmf") {
    # This will plot a series of horizontal lines at q with height p, setting
    # the y limits to a reasonable heights
    plot(q, p, type = "h", xlab = "x", ylab = "probability", main = main, ylim = c(0,
        max(p) + 0.1))
    # Usually these plots have a dot at the end of the line; the point function
    # will add these dots to the plot created above
    points(q, p, pch = 16, cex = 1.5)
}

# Change plot settings
old_par <- par()
par(mfrow = c(2, 1))

# Possible probabilities
(p <- seq(0, 1, by = 0.05))
```

```
##  [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
## [16] 0.75 0.80 0.85 0.90 0.95 1.00
```

```r
# The prior distribution of p
(prior <- rep(1/length(p), times = length(p)))
```

```
##  [1] 0.04761905 0.04761905 0.04761905 0.04761905 0.04761905 0.04761905
##  [7] 0.04761905 0.04761905 0.04761905 0.04761905 0.04761905 0.04761905
## [13] 0.04761905 0.04761905 0.04761905 0.04761905 0.04761905 0.04761905
## [19] 0.04761905 0.04761905 0.04761905
```

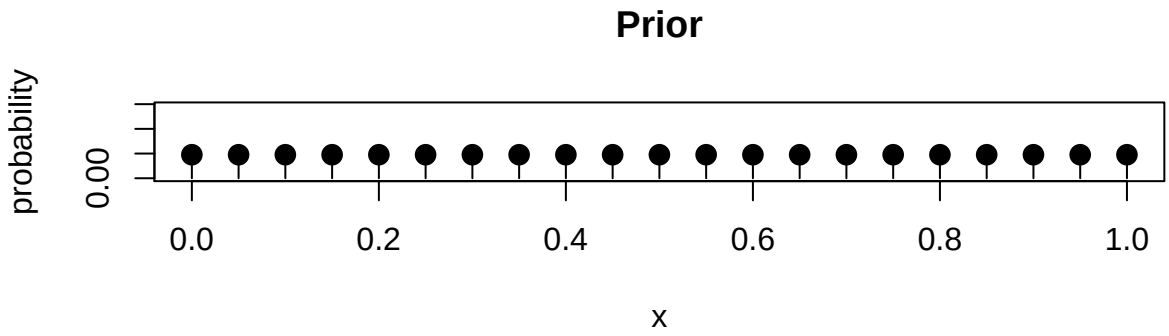
```r
# Plot the prior
plot_pmf(p, prior, main = "Prior")

# Compute the likelihood function
n <- 30  # Number of widgets
s <- 2  # Number of defective widgets
(like <- p^s * (1 - p)^(n - s))
```

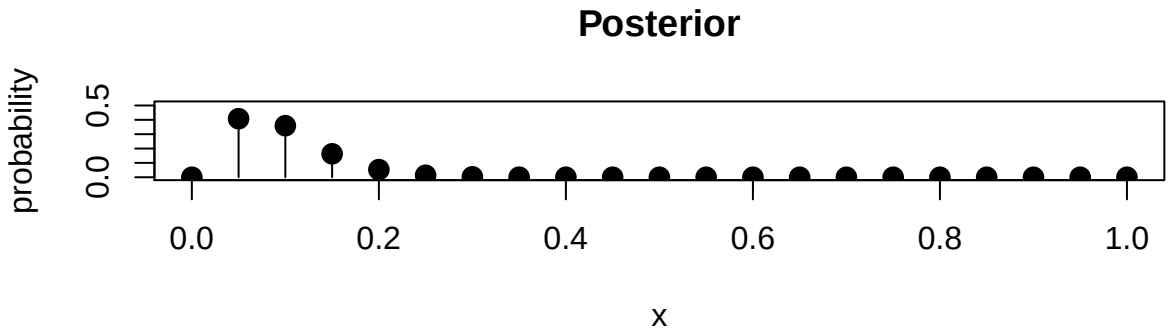
```
##  [1] 0.000000e+00 5.945672e-04 5.233476e-04 2.376361e-04 7.737125e-05
##  [6] 1.984245e-05 4.139879e-06 7.074733e-07 9.825508e-08 1.087878e-08
## [11] 9.313226e-10 5.897606e-11 2.594073e-12 7.239890e-14 1.120963e-15
## [16] 7.806256e-18 1.717987e-20 6.157340e-24 8.100000e-29 3.362074e-37
## [21] 0.000000e+00
```

```r
# With the likelihood and prior computed, we now obtain the posterior
# distribution.
post <- like * prior
(post <- post/sum(post))  # Normalize; make proper probabilities
```

```
##  [1] 0.000000e+00 4.078742e-01 3.590174e-01 1.630188e-01 5.307682e-02
##  [6] 1.361196e-02 2.839964e-03 4.853280e-04 6.740316e-05 7.462865e-06
## [11] 6.388890e-07 4.045768e-08 1.779539e-09 4.966577e-11 7.689825e-13
## [16] 5.355105e-15 1.178542e-17 4.223946e-21 5.556614e-26 2.306389e-34
## [21] 0.000000e+00
```

```r
plot_pmf(p, post, main = "Posterior")
```






```
# compute 95%
# Getting the MAP estimator for p
print("THE 95% FOR THE NEW DATA")
```

```
## [1] "THE 95% FOR THE NEW DATA"
```

```
p[which.max(post)]
```

```
## [1] 0.05
```

```
# get the cdf
(post_cdf <- cumsum(post))
```

```
##  [1] 0.0000000 0.4078742 0.7668916 0.9299104 0.9829872 0.9965992 0.9994391
##  [8] 0.9999245 0.9999919 0.9999993 1.0000000 1.0000000 1.0000000 1.0000000
## [15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
(a <- p[max(which(post_cdf < 0.025))])
```

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

```r
(b <- p[min(which(post_cdf > 0.975))])
```

```
## [1] 0.2
```

```r
sum(post[which(p >= a & p <= b)])
```

```
## [1] 0.9829872
```

```r
# Increasing the resolution lowered my original result from 0.992 to 0.982
# I would prefer the higher resolution because when by lowering the size of the confidence interval we
```

5. *An advantage of Bayesian statistics is it's much more elegant to updating conclusions given prior conclusions and new data. The posterior distribution becomes the prior distribution of the next test, and a new posterior distribution can be computed given new data. Let's say the inspector draws ten more widgets from the batch and discovers that among the widgets in the new sample, five are defective. Repeat part 4, but using the posterior of the first experiment as the prior distribution of the next, and then compute the new posterior distribution. How does the new MAP estimator for p and the new 95% credible interval compare to the old? What is the new probability that there are too many defective widgets? Based on this evidence, should the inspector recommend the batch be rejected?*

```r
# Your code here
```