Methods Homework 1

Tim Vigers

September 9, 2018

Exercise 1

a.

Calculate the probability that 2.5% of Patagonians have the disease, assuming a sample size of 120 and population prevalence of 1%. Use both the exact binomial probability and the Poisson approximation of it. Compare the two.

```
# Calculate the probability using the binomial PMF.
# With a sample size 120, 2.5% is equal to three cases.
choose(120,3) * (0.01)^3 * (1 - 0.01)^(120-3)
```

```
## [1] 0.08665163
```

```
# Double check with dbinom().
dbinom(x = 3,size = 120,prob = 0.01)
```

```
## [1] 0.08665163
```

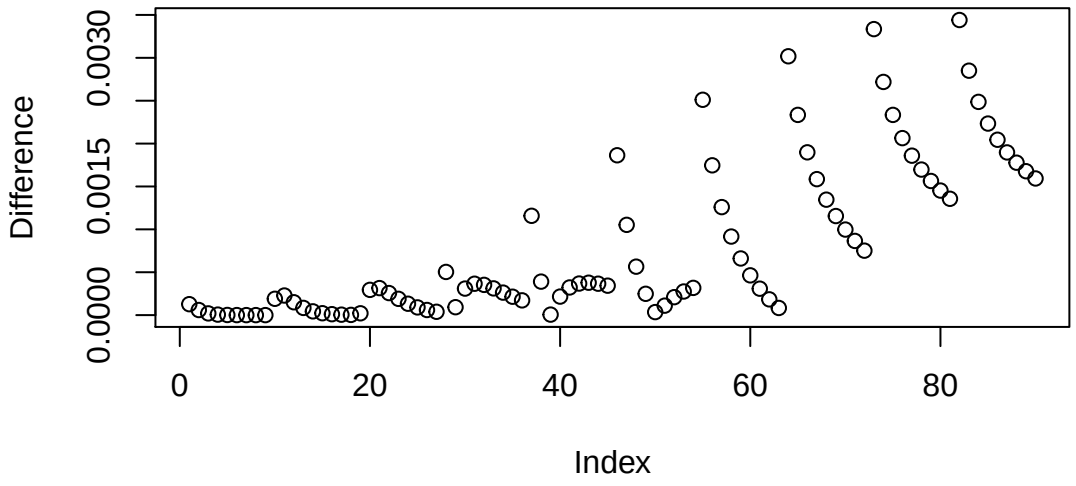
```
# For the Poisson disribution, lambda = np. Check the probability using dpois().
dpois(x = 3, lambda = (120 * 0.01))
```

```
## [1] 0.08674393
```

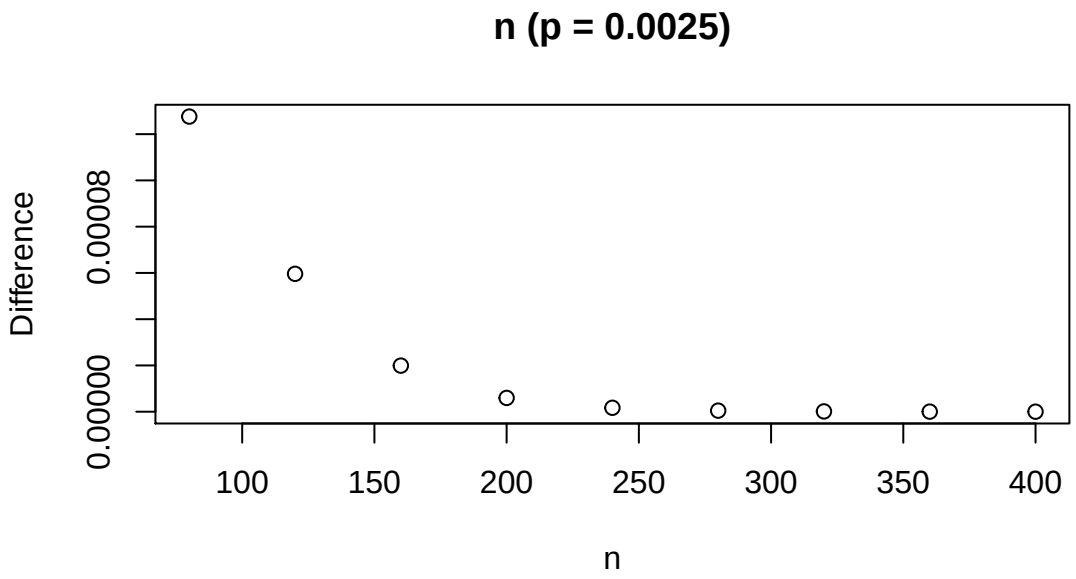
It looks like the Poisson approximation works well for this case (it also fits Rosner's rule where n >= 100 and p <=0.01).

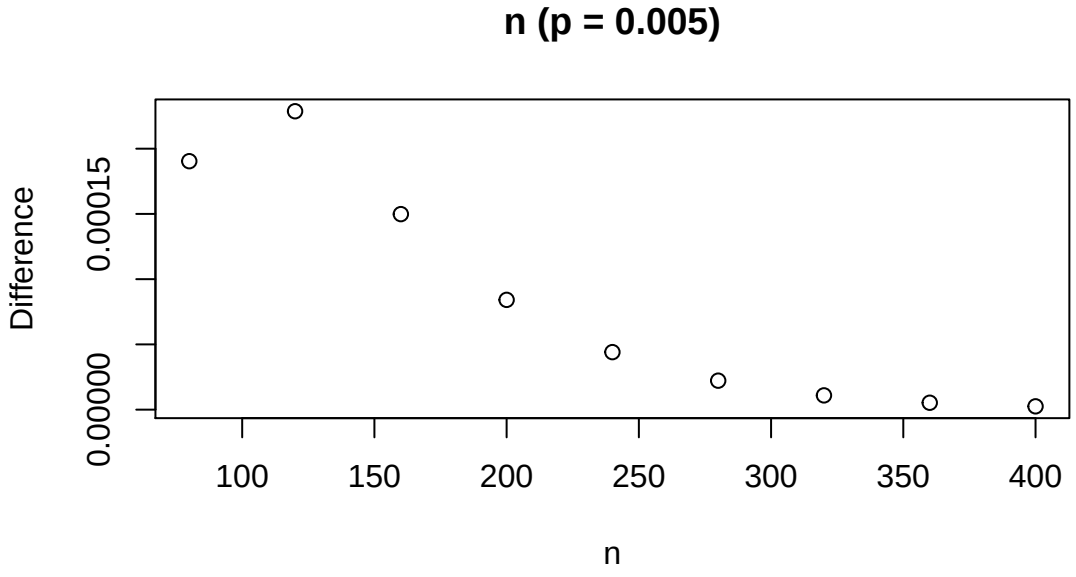
b.

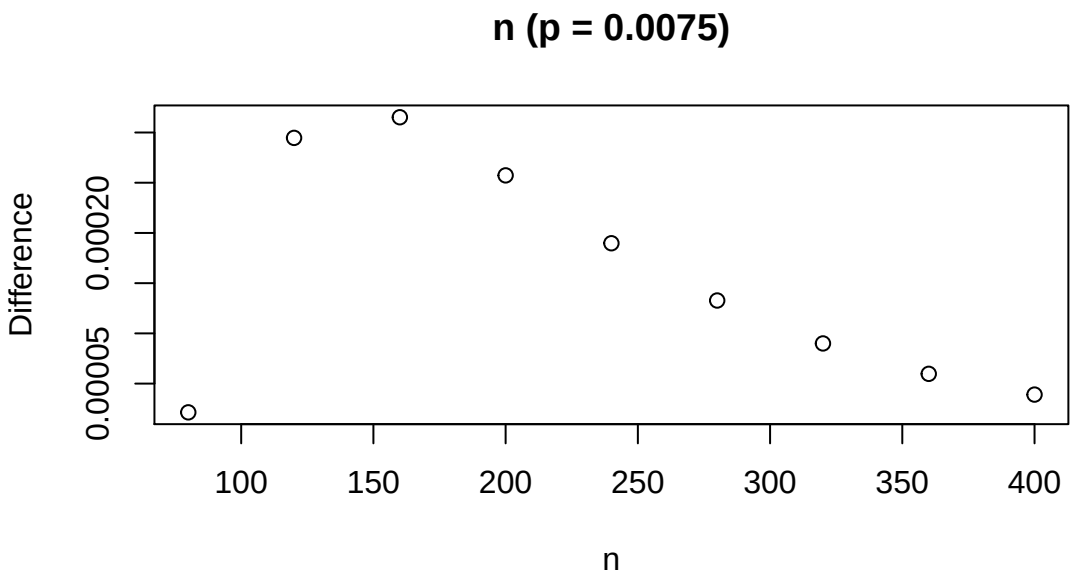
```r
# Use the hint provided.
n=seq(80,400,by=40)
p=seq(0.0025,.025,by=.0025)
np<-expand.grid(n=n,p=p)
# Add a column to np for each of the exact binomial,
# the Poisson approximation of the binomial, and the difference between the two.
np$binom <- dbinom(x = (0.025 * np$n),size = np$n,prob = np$p)
np$poisson <- dpois(x = (0.025 * np$n), lambda = (np$n * np$p))
np$diff <- abs(np$binom - np$poisson)
# Plot everything together.
plot(np$diff,ylab = "Difference")
```

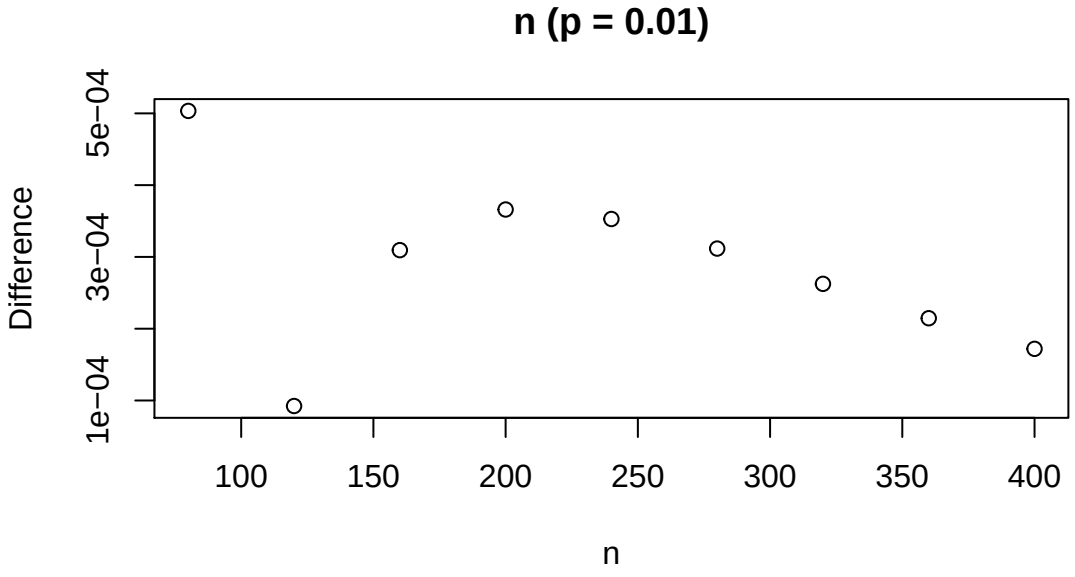


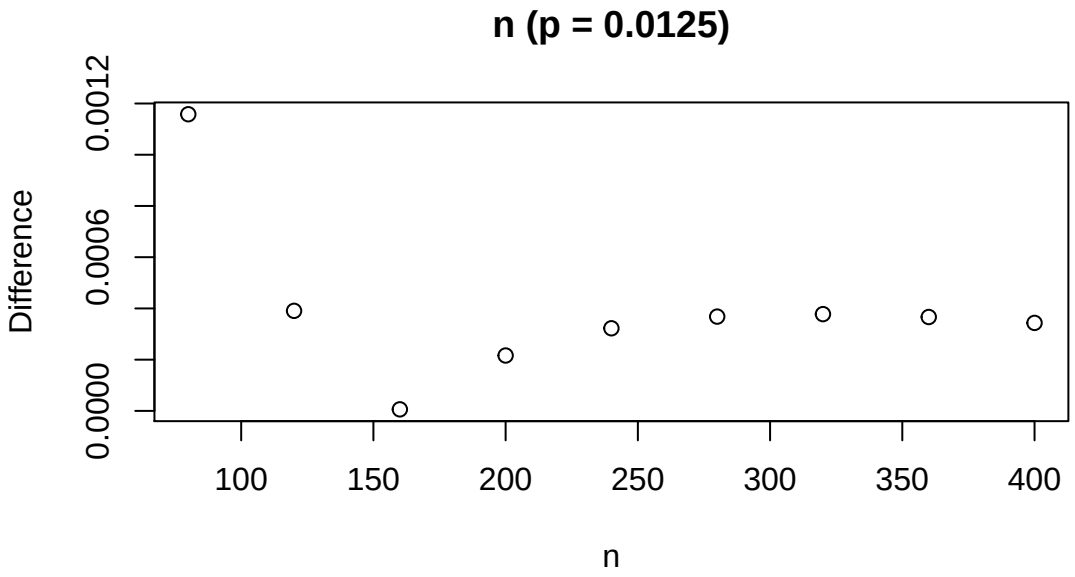
Plot each probability separately.

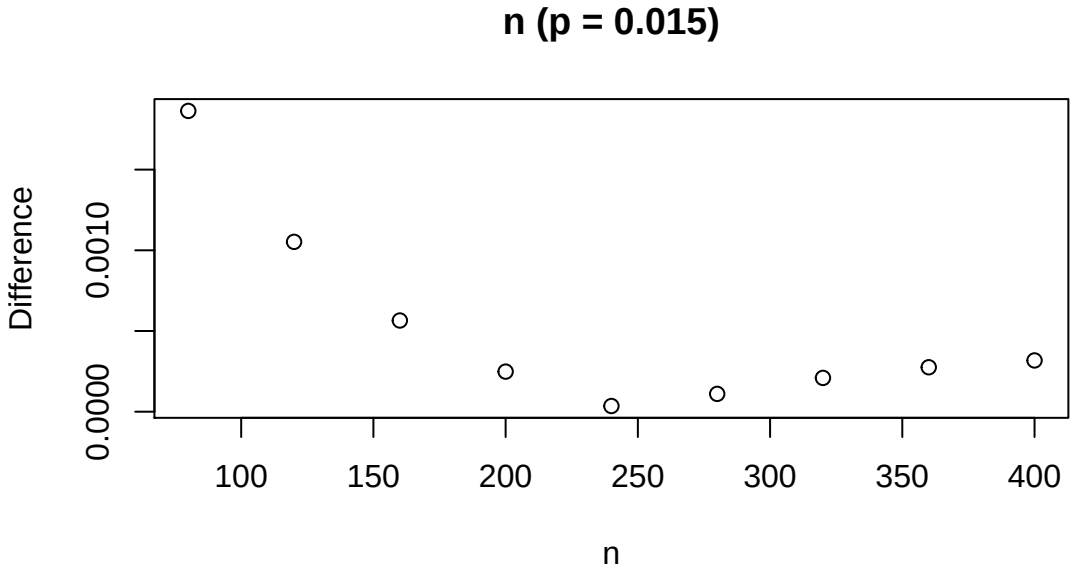


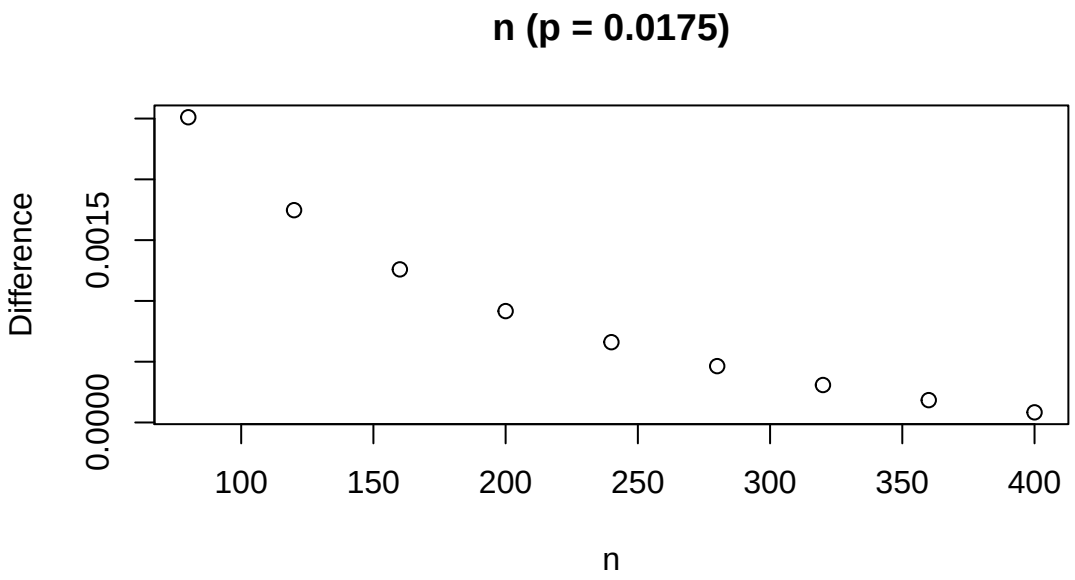


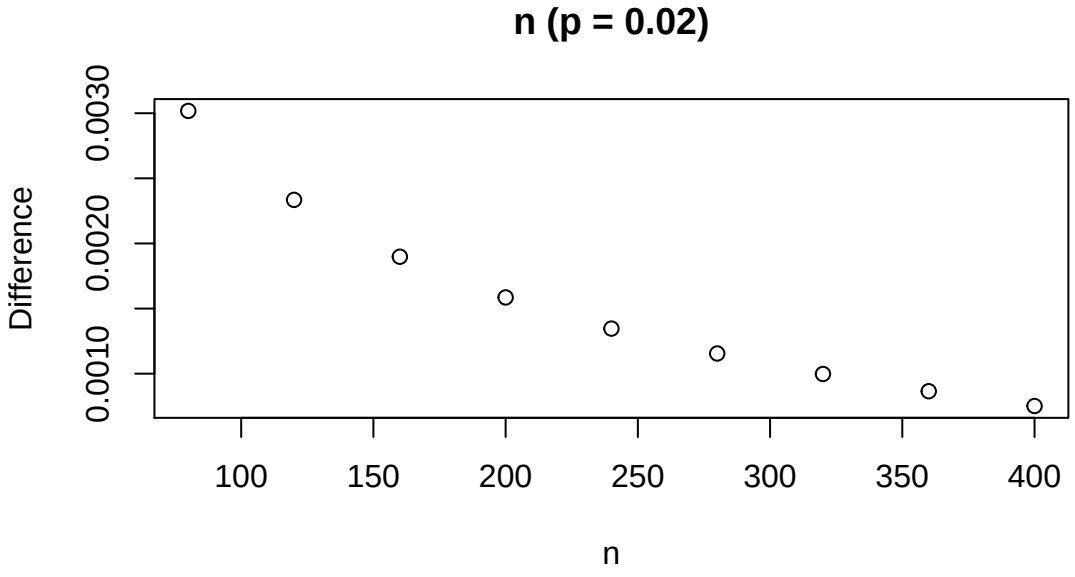

n (p = 0.0075)
Difference
0.00005
0.00020
100
150
200
250
300
350
400
n

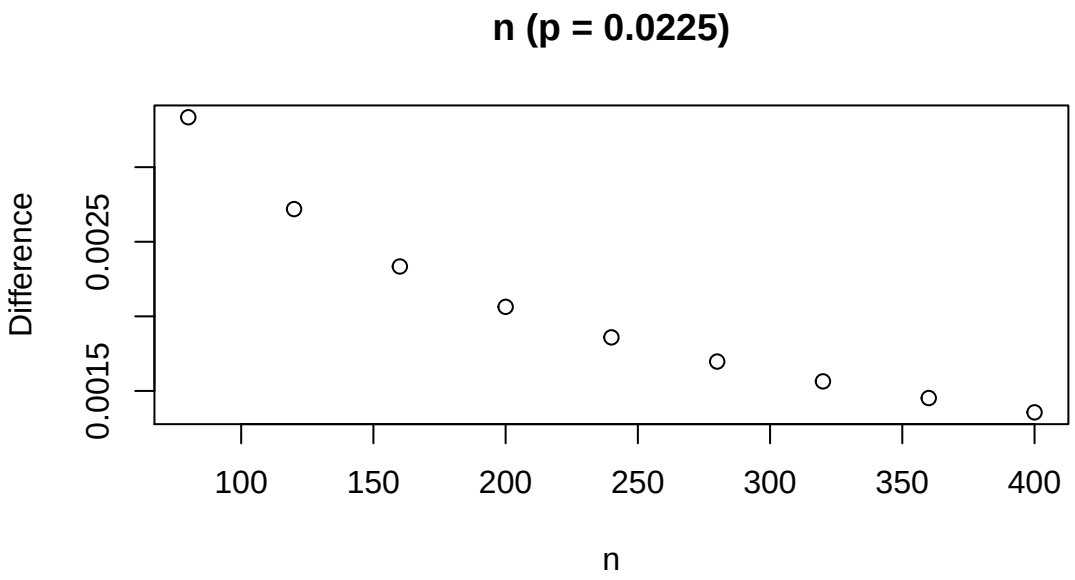

n (p = 0.01)
Difference
1e−04
3e−04
5e−04
100
150
200
250
300
350
400
n

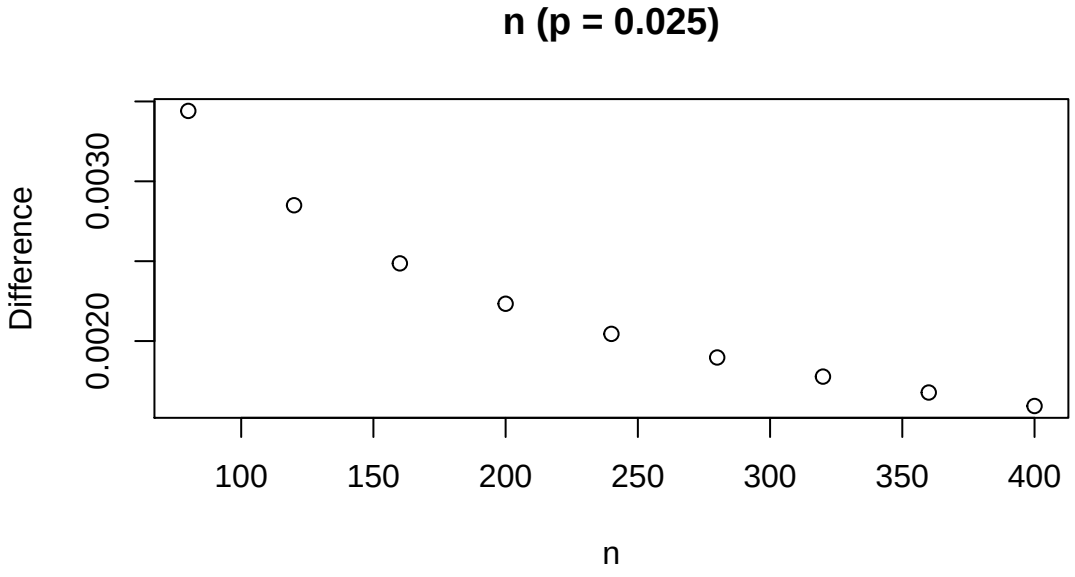

n (p = 0.0125)
Difference
0.0000
0.0006
0.0012
100
150
200
250
300
350
400
n


n (p = 0.015)
Difference
0.0000
0.0010
100
150
200
250
300
350
400
n


n (p = 0.0175)
Difference
0.0000
0.0015
100
150
200
250
300
350
400
n


n (p = 0.02)
Difference
0.0010
0.0020
0.0030
100
150
200
250
300
350
400
n


n (p = 0.0225)
Difference
0.0015
0.0025
100
150
200
250
300
350
400
n


n (p = 0.025)
Difference
0.0020
0.0030
100
150
200
250
300
350
400
n

c.

Looking at the plot of everything together, it looks like the differences start to go a little wild at around index 30 (most likely a little before). Index 30 is p = 0.01 and n = 160. The recommendation depends on how conservative you want to be, but I agree with Rosner that p = 0.01 is the maximum probability you'd want to use the Poisson approximation for. I also think that his rule of n >= 100 makes sense based on this plot, since for p = 0.01, n = 80 looks pretty bad.

Exercise 2

a.

The exponential function is only defined for x >= 0, so for the expected value you integrate from 0 to infinity.

$$E[X] = \int_0^\infty 3xe^{-3x}$$

$$u = x \text{ and } dv = 3e^{-3x} \text{ , so } v = -e^{-3x}$$

Therefore:

$$E[X] = -xe^{-3x}\Big|_0^\infty + \int_0^\infty e^{-3x} = (0-0) + (0-(-\frac{1}{3})) = \frac{1}{3}$$

Sally can expect to wait in line for about 20 minutes.

b.

$$Var(X) = E[X^2] - E[X]^2$$

$$E[X]^2 = \frac{1}{9}$$

$$E[X^2] = \int_0^\infty x^2 3e^{-3x}$$

$$u = x^2 \text{ and } dv = 3e^{-3x} \text{ , so } v = -e^{-3x} \text{ and } du = 2xdx$$

$$E[X^2] = -x^2e^{-3x}\Big|_0^\infty + \int_0^\infty e^{-3x} = (0-0) + \int_0^\infty 2xe^{-3x}$$

$$\int_0^\infty 2xe^{-3x}$$

$$u = 2x \text{ and } dv = e^{-3x} \text{ , so } v = -\frac{1}{3}e^{-3x} \text{ and } du = 2dx$$

$$\int_0^\infty 2xe^{-3x} = -2xe^{-3x}\Big|_0^\infty + \frac{2}{3}\int_0^\infty e^{-3x} = \frac{2}{3} * \frac{1}{3} = \frac{2}{9}$$

So:

$$Var(X) = E[X^2] - E[X]^2 = \frac{2}{9} - \frac{1}{9} = \frac{1}{9}$$

c.

Reproducibly simulate an Exponential(3) distribution of size 100,000:

```
# Set seed for reproducibility.
set.seed(1017)
# Simulate distribution.
expdist <- rexp(n = 100000, rate = 3)
# Calculate mean and variance.
mean(expdist)
```

```
## [1] 0.3338419
```

```
var(expdist)
```

```
## [1] 0.1106774
```

The mean and variance of the simulated data are very close the theoretical values calculated above.

d.

Because of the memoryless property of the exponential distribution, " 'waiting time' until a certain event does not depend on how much time has already elapsed." So Sally should still plan on waiting 20 minutes, even though she's already waited 10. This definitely sounds like the DMV.

Exercise 3

a.

```
set.seed(1017)
# Simulate the normal distribution.
normdist <- rnorm(n = 100, mean = 70, sd = sqrt(15))
# Find the median of the simulated group.
median(normdist)
```

```
## [1] 69.4482
```

```
# Find the difference.
median(normdist) - 70
```

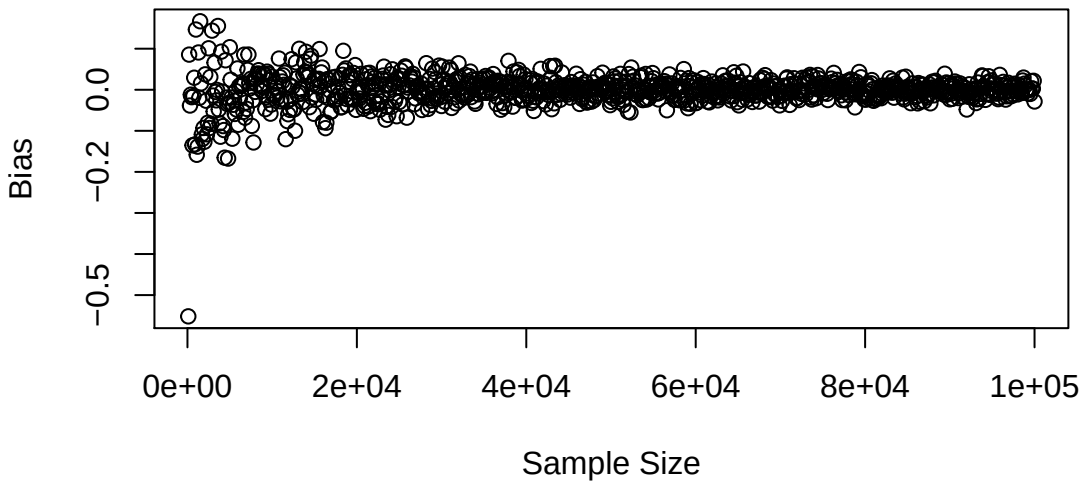
```
## [1] -0.5517957
```

Bias is the difference between the estimator and the population value, so in this case it's -0.5517957.

b.

Increasing sample size by increments of 100:

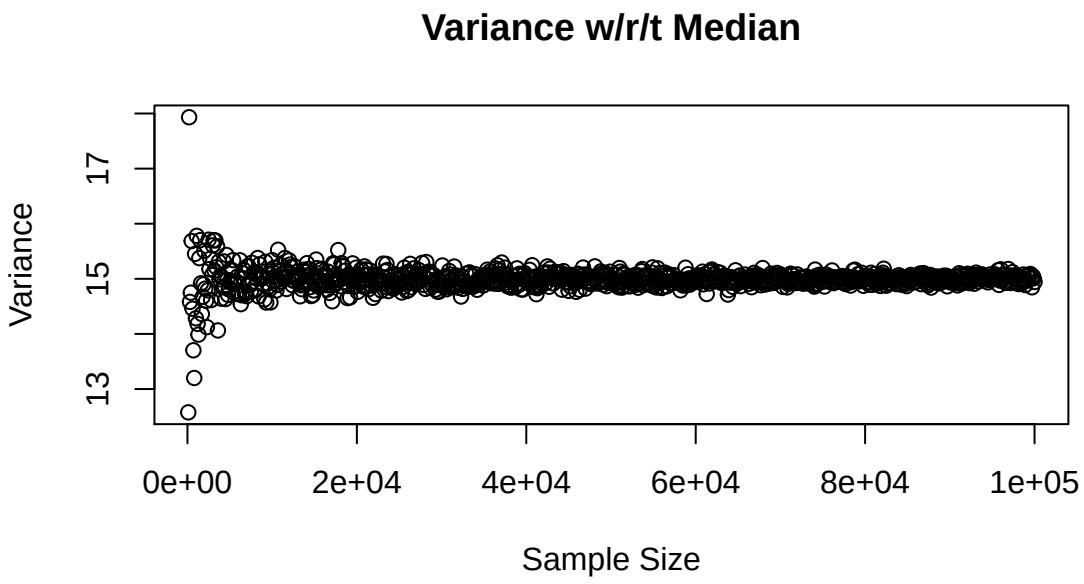
```
set.seed(1017)
# Create a sequence of increasing sample size.
increasingns <- seq(from = 100, to = 100000, by = 100)
# Create an empty vector to store medians.
vector_of_sample_medians <- rep(-9, length(increasingns))
# For each sample size, generate a dataset and find the median. Store in vector of medians.
for (i in 1:length(increasingns)) {
  vector_of_sample_medians[i] <- median(rnorm(n = increasingns[i], mean = 70, sd = sqrt(15)))
}
# Calculate the biases.
vector_of_bias <- vector_of_sample_medians - 70
plot(x = increasingns,y = vector_of_bias,xlab = "Sample Size", ylab = "Bias")
```



Consistency means that "when the sample size becomes infinite, we desire that this bias disappears." Based on the above plot, it seems like this median estimate is pretty consistent, as the bias approaches 0 as n increases.

c.

```
set.seed(1017)
# Make an empty vector for variances.
var_wrt_estimator <- rep(-9, length(increasingns))
# For each sample size, generate a distribution.
for (i in 1:length(increasingns)) {
  dist <- rnorm(n = increasingns[i], mean = 70, sd = sqrt(15))
# Take each value in the distribution, subtract the sample median, square the result, and then divide b
  var_wrt_estimator[i] <-
    sum(((dist - vector_of_sample_medians[i])^2) / increasingns[i])
}
# Plot the variance as n increases.
plot(x = increasingns,y = var_wrt_estimator,main = "Variance w/r/t Median",xlab = "Sample Size", ylab =
```



The variance of the data approaches the theoretical variance of 15 as n increases.

d.

```
# Set seed, determine number of simulations and sample size.
set.seed(1017)
number_of_sims <- 10000
sample_size <- 1000
# Create a vector to store sample means and medians.
vector_of_sample_means <- rep(-9, number_of_sims)
vector_of_sample_medians <- rep(-9, number_of_sims)
# For loop to generate values and store in their respective vectors.
for (i in 1:number_of_sims) {
  vector_of_sample_means[i] <-
    mean(rnorm(n = sample_size, mean = 70, sd = sqrt(15)))
  vector_of_sample_medians[i] <-
    median(rnorm(n = sample_size, mean = 70, sd = sqrt(15)))
}
# Compare the variances of means and medians.
var(vector_of_sample_means)
```

```
## [1] 0.01494572
```

```
var(vector_of_sample_medians)
```

```
## [1] 0.02345415
```

```
var(vector_of_sample_means) / var(vector_of_sample_medians)
```

```
## [1] 0.6372313
```

Based on the code above, it appears that using the mean is a more efficient estimator than than the median, at least at this large sample size.

e.

The Cramér-Rao bound expresses a lower bound on the variance of estimators. In other words, if an estimator "achieves this lower bound [it] is said to be (fully) efficient" (https://en.wikipedia.org/wiki/Cram%C3%A9r%E2%80%93Rao_bound).

In order to test the efficiency of an estimators, you would find the Cramér-Rao bound by taking the inverse of the Fisher information (although I'm not sure how to do that), and dividing that by the estimator's variance. So Using this bound would allow us to compare the efficiencies of our two estimators, in order to figure out which is the best one to use.