ST 502 R Project 1

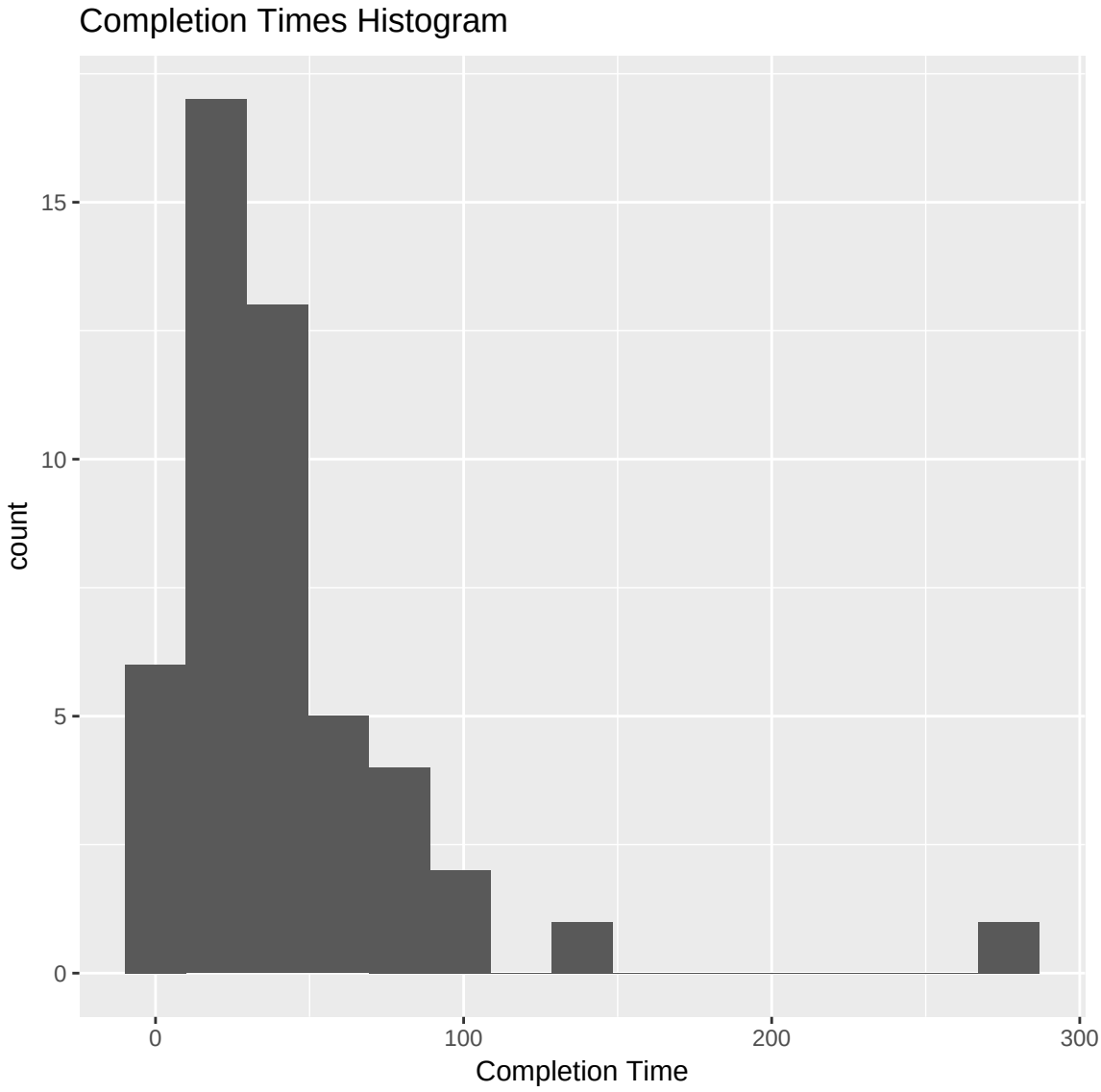
Brandon Sandusky, Bruce Campbell, Kenneth Wright

March 17, 2017

This report considers data that represents a sample of survey completion times. The population distribution is assumed to be an exponential distribution. Inference for the rate parameter λ will be explored here. We will calculate a variety of confidence intervals for the rate parameter λ and will then follow up with Monte Carlo simulation sudies of the convergence rates of the confidence interval procedures. We use $\alpha = 0.05$ for the coverage of all calculated confidence intervals.

Plot the data

```
# install.packages('pander') install.packages('plyr')
# install.packages('dplyr') install.packages('readr')
# install.packages('ggplot2')
library(pander)
library(plyr)
library(dplyr)
library(readr)
library("ggplot2")

completionTimes <- data.frame(as.numeric(unlist(read_csv("Group8Data.csv"))))
colnames(completionTimes) <- c("time")

qplot(completionTimes$time, geom = "histogram", bins = 15, main = "Completion Times Histogram",
    xlab = "Completion Time")
```



Part 1 - calculation of the confidence intervals

(a) Exact interval based on the distribution of the MLE

We calculate the MLE of λ here

$$\mathcal{L}(\lambda, x_1, ..., x_n) = \prod_{i=1}^{n} f(x_i|\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i} = \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

$$\frac{d}{d\lambda} log(\mathcal{L}) = \frac{n}{\lambda} - \sum_{i=1}^{n} x_i$$

From which we get

$$\widehat{\lambda_{MLE}} = \frac{1}{\bar{X}}$$

Recall $Exp(\lambda) \sim Gamma(1, \lambda)$ and $\sum X_i \sim Gamma(n, \lambda)$ for iid $Gamma(1, \lambda)$, and finally that $\frac{1}{n}\sum X_i \sim Gamma(n, n\lambda)$

We can use the pivotal quantity $n\frac{\lambda}{\lambda MLE} \sim Gamma(n, 1)$ and work out that the exact $1 - \alpha$ CI for the MLE estimator of λ is

$$(\frac{1}{n}\widehat{\lambda_{MLE}}\ g_{\frac{\alpha}{2}}, \frac{1}{n}\widehat{\lambda_{MLE}}\ g_{1-\frac{\alpha}{2}})$$

Now we calculate the CI

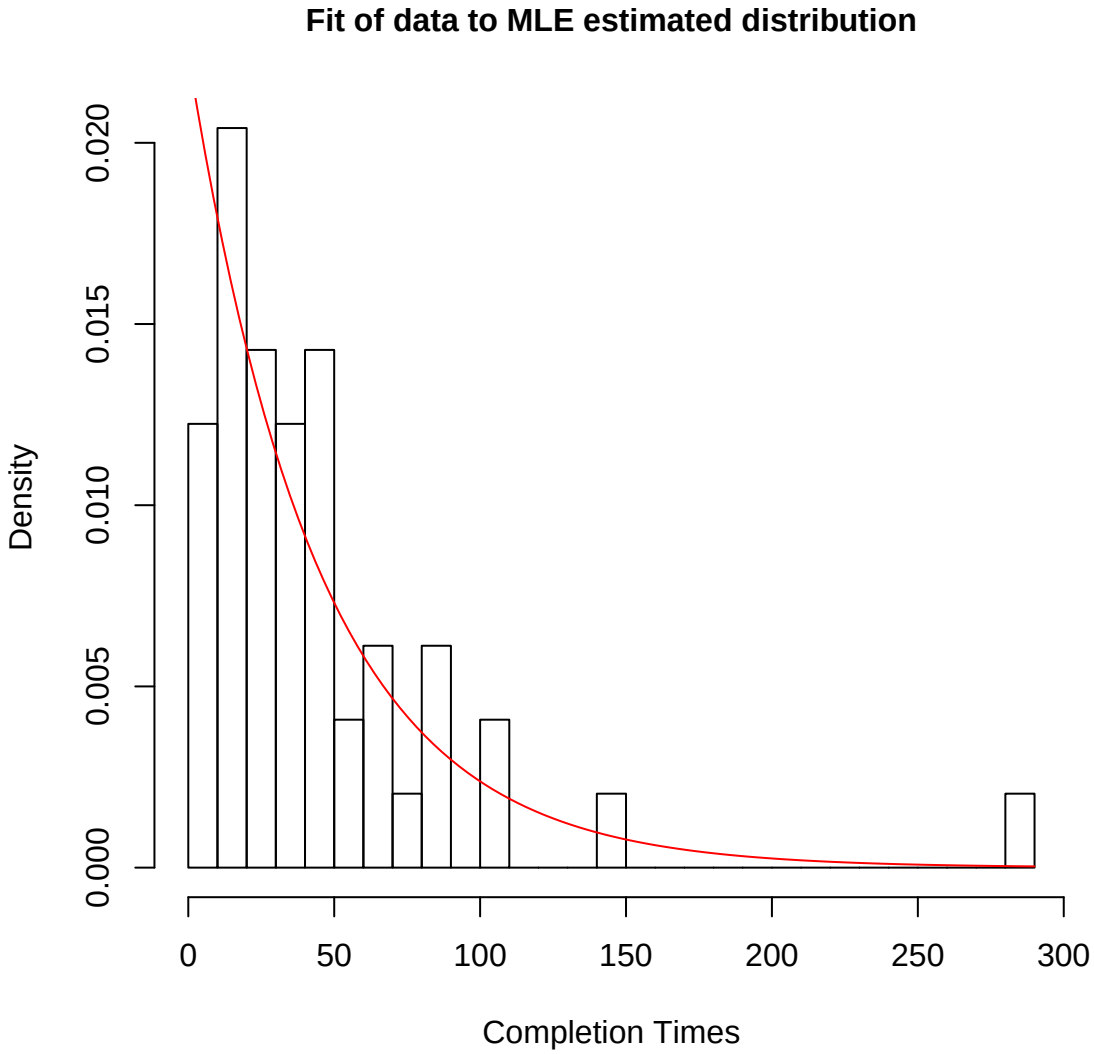
```
DataLength <- nrow(completionTimes)
MLEMean <- 1/mean(completionTimes$time)

ExactInterval <- cbind((MLEMean/DataLength) * qgamma(0.025, DataLength, 1),
    (MLEMean/DataLength) * qgamma(0.975, DataLength, 1))

pander(data.frame(left = ExactInterval[1], right = ExactInterval[2]), caption = "Exact CI from MLE sampl
```

Table 1: Exact CI from MLE sampling distribution

left	right
0.0166	0.02914

The interpretation of the confidence interval is that if we were to perform this procedure many times that approximately 95% of such intervals would contain the true value of the parmeter. Since we have one sample we are 95% confidant that the true rate paramter value lies between left and right values above.

```
hist(as.numeric(completionTimes$time), 20, freq = FALSE, cex.main = 1, main = "Fit of data to MLE estima
    xlab = "Completion Times")
curve(dgamma(x, 1, MLEMean), add = TRUE, col = "red")
```



Using the Fisher information and the CLT, we derive the asymptotic distribution

$$\widehat{\lambda_{MLE}} \sim N(\lambda, \frac{\lambda^2}{n})$$

This approximation works for larger n (n>30). Since our sample size is greater than this we expect the approximation to work well and be in close agreement with the exact results.

(b) Large-sample interval based on the asympotic distribution of the MLE

```
# Get the Large Sample Interval
LargeSampleCI <- cbind(qnorm(0.025, MLEMean, MLEMean/sqrt(DataLength)), qnorm(0.975,
    MLEMean, MLEMean/sqrt(DataLength)))

LargeSampleCI <- data.frame(LargeSampleCI)
```

```r
colnames(LargeSampleCI) <- c("left", "right")
pander(LargeSampleCI, caption = " 95% CI from large sample approximation")
```

Table 2: 95% CI from large sample approximation

left	right
0.01615	0.02872

The bootstrap procedure here is to use the point estimates to simulate a collection SRS's from a distribution with the point estimates as parameters. We can use the empirical distribution of the bootstrap estimate to approximate properties of the sampling distribution of the estimator. Below we make plots of the bootstrap sample for $\widehat{\lambda_{MLE}}$, and then use the quantiles of the empirical distribution for estimating the CI.

```r
library(pander)
set.seed(123)

# set up the sample sizes and numbers of samples for our bootstrap process
n <- DataLength
B <- 5000

# Now do Parametric Intervals
ParametricStats <- Lambda.hat <- rep(0, B)
for (i in 1:B) {
    Parametric.Data <- rgamma(n, 1, MLEMean)  #Get the sample
    Lambda.hat[i] <- 1/mean(Parametric.Data)  #Get lambda for the sample
    bootstrapSE <- sqrt(Lambda.hat[i]^2/n)  #Get standard Error of the sample
    ParametricStats[i] <- (Lambda.hat[i] - MLEMean)/bootstrapSE  #Boot t stat
}
```

Parametric Raw Percentile interval

```r
ParametricBootstrap.RawQuantile <- quantile(Lambda.hat, c(0.025, 0.975))
ParametricBootstrap.RawQuantile
```

```
##       2.5%      97.5%
## 0.01720542 0.03017154
```

Parametric Reflected Percentile interval

```r
Parametric.ReflectedInterval <- c(MLEMean - (quantile(Lambda.hat, 0.975) - MLEMean),
    MLEMean - (quantile(Lambda.hat, 0.025) - MLEMean))
Parametric.ReflectedInterval
```

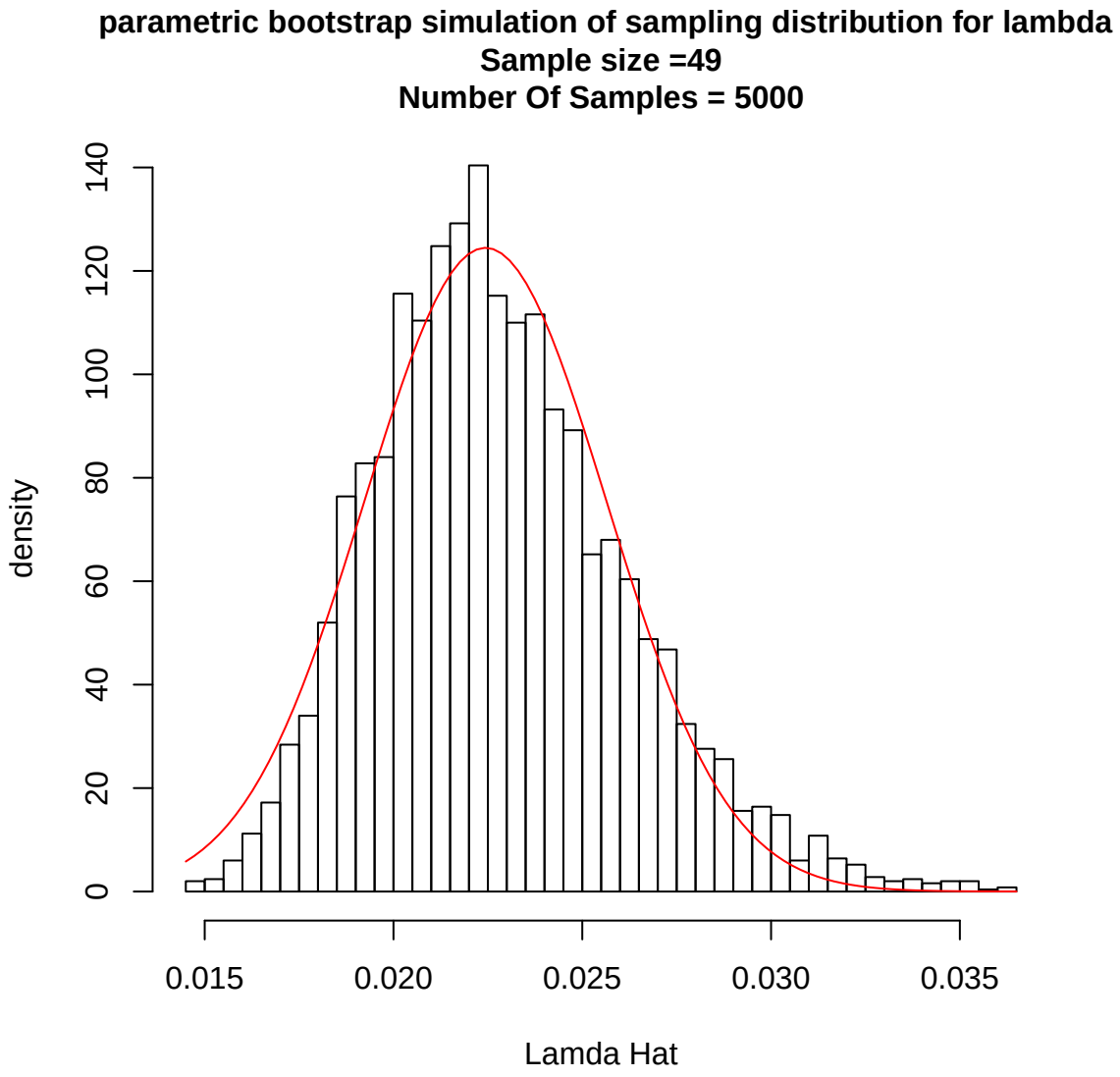
```
##      97.5%       2.5%
## 0.01470025 0.02766638
```

Parametric bootstrap t interval

```
Parametric.bootstrapSE <- sqrt((1/B) * sum((Lambda.hat - mean(Lambda.hat))^2))
Parametric.Bootstrap.TInterval <- c(MLEMean - quantile(ParametricStats, 0.975) *
    Parametric.bootstrapSE, MLEMean - quantile(ParametricStats, 0.025) * Parametric.bootstrapSE)
Parametric.Bootstrap.TInterval
```

```
##      97.5%       2.5%
## 0.01648615 0.02949056
```

```
titleString <- c(" parametric bootstrap simulation of sampling distribution for lambda",
    paste("Sample size =", n, sep = ""), paste("Number Of Samples = ", B, sep = ""))
hist(as.numeric(Lambda.hat), 40, freq = FALSE, main = titleString, cex.main = 1,
    xlab = "Lamda Hat", ylab = "density")
curve(dnorm(x, MLEMean, sqrt(MLEMean^2/n)), col = "red", add = TRUE)
```



We see a good fit to the asymptotic sampling distribution, and now we calculate the CI.

Non-Parametric Raw Percentile interval

```
MLE.Estimate.Vector <- NonParametricStats <- rep(0, B)

# Create B=5000 NonParametric Bootstrap Samples
for (i in 1:B) {
    NonParametricData <- sample(x = completionTimes$time, size = n, replace = TRUE)  #Sample from data
    MLE.Estimate.Vector[i] <- 1/mean(NonParametricData)  #Find Bootstrap MLE of each est.
    bootstrapSE <- sqrt(MLE.Estimate.Vector[i]^2/n)  #Get standard Error of each est
    NonParametricStats[i] <- (MLE.Estimate.Vector[i] - MLEMean)/bootstrapSE  #Boot t stats
}

# Find Raw Quantile
NonParBootstrap.RawQuantile <- quantile(MLE.Estimate.Vector, c(0.025, 0.975))
NonParBootstrap.RawQuantile
```

```
##       2.5%      97.5%
## 0.01691974 0.02973346
```

Non Parametric Refected Percentile interval

```
NonParReflectedInterval <- c(MLEMean - (quantile(MLE.Estimate.Vector, 0.975) -
    MLEMean), MLEMean - (quantile(MLE.Estimate.Vector, 0.025) - MLEMean))
NonParReflectedInterval
```

```
##      97.5%       2.5%
## 0.01513833 0.02795205
```

Non-Parametric Bootstrap t-interval

```
NonParbootstrapSE <- sqrt((1/B) * sum((MLE.Estimate.Vector - mean(MLE.Estimate.Vector))^2))
NonParBootstrap.TInterval <- c(MLEMean - quantile(NonParametricStats, 0.975) *
    bootstrapSE, MLEMean - quantile(NonParametricStats, 0.025) * bootstrapSE)
NonParBootstrap.TInterval
```

```
##      97.5%       2.5%
## 0.01664850 0.03012354
```

Part 2

```
# Set up the basics/parameters
n <- 49  # our sample sizes - choose 49 to match the raw data sample size we were provided
numSamples <- 1000  # how many samples we will calculate intervals for
lambda <- 0.0207  # true lambda we are working with
B <- 500  # number of bootstrap samples we will use

run.CI <- function(n, numSamples, lambda, B) {
    NR <- numSamples
    CIs <- data.frame(MLEMean = numeric(NR), ExactLeft = numeric(NR), ExactRight = logical(NR),
        ExactContainsP = logical(NR), LSampLeft = numeric(NR), LSampRight = numeric(NR),
```

```
    LSampContainsP = logical(NR), BootLeft = numeric(NR), BootRight = numeric(NR),
    BootContainP = logical(NR), ReflLeft = numeric(NR), ReflRight = numeric(NR),
    ReflContainP = logical(NR), TLeft = numeric(NR), TRight = numeric(NR),
    TcontainsP = logical(NR), NonParBootLeft = numeric(NR), NonParBootRight = numeric(NR),
    NonParBootContainP = logical(NR), NonParReflLeft = numeric(NR), NonParReflRight = numeric(NR),
    NonParReflContainP = logical(NR), NonParTLeft = numeric(NR), NonParTRight = numeric(NR),
    NonParTcontainsP = logical(NR))

for (i in 1:numSamples) {
    # Get the data for part 2 analyses, and its MLEMean
    data <- rgamma(n = n, 1, lambda)
    MLEMean <- 1/mean(data)
    CIs[i, ]$MLEMean <- MLEMean

    # Get the Exact Interval
    ExactInterval <- cbind((MLEMean/(2 * n)) * qchisq(0.025, 2 * n), (MLEMean/(2 *
        n)) * qchisq(0.975, 2 * n))

    CIs[i, ]$ExactLeft <- ExactInterval[1]
    CIs[i, ]$ExactRight <- ExactInterval[2]
    CIs[i, ]$ExactContainsP <- CIs[i, ]$ExactLeft <= lambda & lambda <=
        CIs[i, ]$ExactRight

    # Get the Large Sample Interval
    LargeSampleCI <- cbind(qnorm(0.025, MLEMean, MLEMean/sqrt(n)), qnorm(0.975,
        MLEMean, MLEMean/sqrt(n)))

    CIs[i, ]$LSampLeft <- LargeSampleCI[1]
    CIs[i, ]$LSampRight <- LargeSampleCI[2]
    CIs[i, ]$LSampContainsP <- CIs[i, ]$LSampLeft <= lambda & lambda <=
        CIs[i, ]$LSampRight

    # Get the Parametric Bootstrap Intervals Start with getting B bootstrap
    # samples for each of our data samples
    ParametricStats <- ParametricLambda.hat <- rep(0, B)
    for (j in 1:B) {
        Parametric.Data <- rgamma(n, 1, MLEMean)
        ParametricLambda.hat[j] <- 1/mean(Parametric.Data)
        bootstrapSE <- sqrt(ParametricLambda.hat[j]^2/n)  #Get standard Error of each est
        ParametricStats[j] <- (ParametricLambda.hat[j] - MLEMean)/bootstrapSE  #Boot t stats
    }

    # Parametric Raw Percentile Interval
    ParametricBootstrap.RawQuantile <- quantile(ParametricLambda.hat, c(0.025,
        0.975))

    CIs[i, ]$BootLeft <- as.numeric(ParametricBootstrap.RawQuantile[1])
    CIs[i, ]$BootRight <- as.numeric(ParametricBootstrap.RawQuantile[2])
    CIs[i, ]$BootContainP <- CIs[i, ]$BootLeft <= lambda & lambda <= CIs[i,
        ]$BootRight

    # Parametric Reflected Percentile Interval
    Parametric.ReflectedInterval <- c(MLEMean - (quantile(ParametricLambda.hat,
```

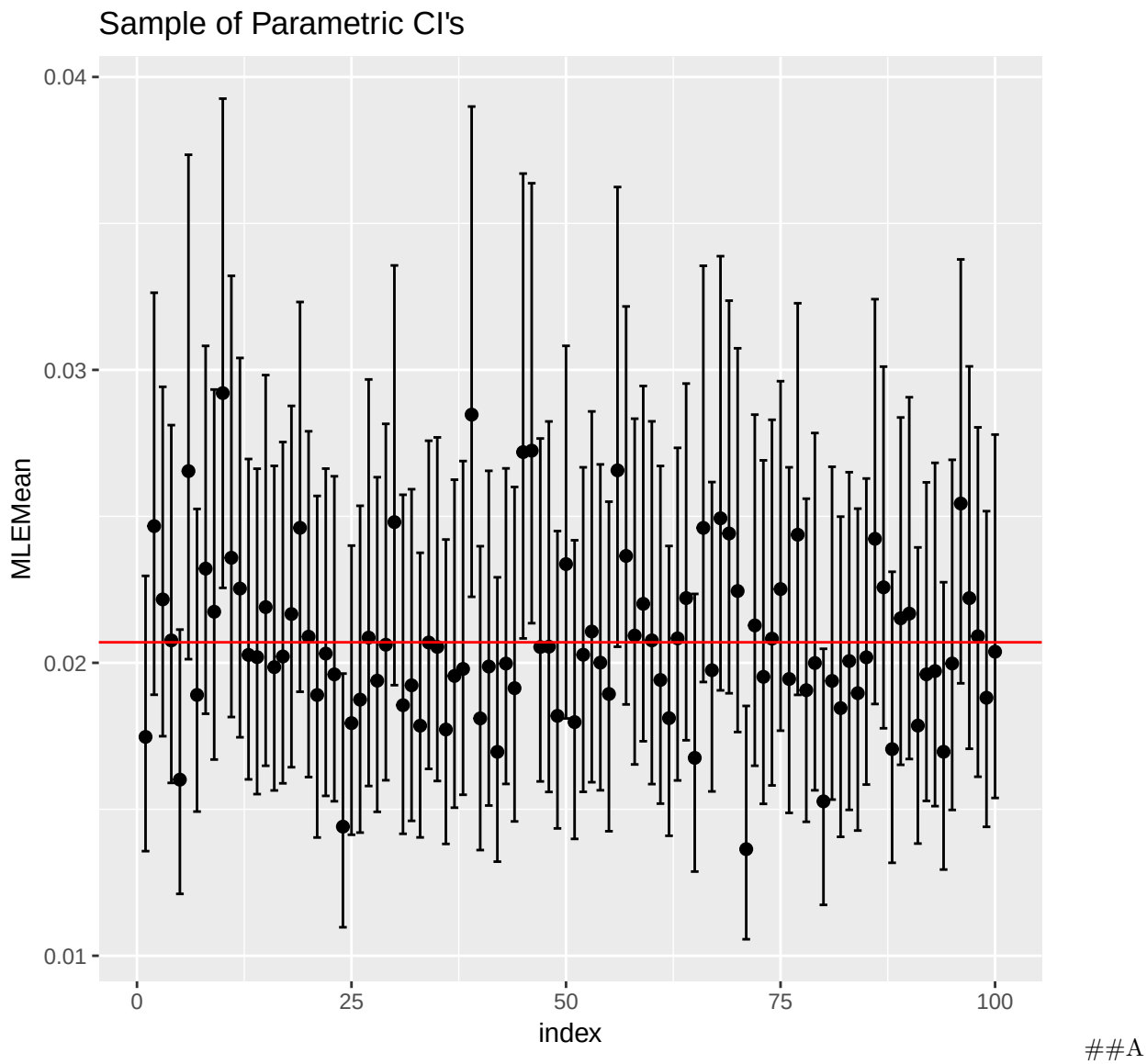
```
    0.975) - MLEMean), MLEMean - (quantile(ParametricLambda.hat, 0.025) -
    MLEMean))

CIs[i, ]$ReflLeft <- Parametric.ReflectedInterval[1]
CIs[i, ]$ReflRight <- Parametric.ReflectedInterval[2]
CIs[i, ]$ReflContainP <- CIs[i, ]$ReflLeft <= lambda & lambda <= CIs[i,
    ]$ReflRight

# Parametric T Interval
Parametric.BootstrapSE <- sqrt((1/B) * sum((ParametricLambda.hat - mean(ParametricLambda.hat))^
Parametric.Bootstrap.TInterval <- c(MLEMean - quantile(ParametricStats,
    0.975) * Parametric.BootstrapSE, MLEMean - quantile(ParametricStats,
    0.025) * Parametric.BootstrapSE)

CIs[i, ]$TLeft <- Parametric.Bootstrap.TInterval[1]
CIs[i, ]$TRight <- Parametric.Bootstrap.TInterval[2]
CIs[i, ]$TcontainsP <- CIs[i, ]$TLeft <= lambda & lambda <= CIs[i, ]$TRight

# Get the NonParametric Bootstrap Intervals Start with getting B bootstrap
# samples for each of our data samples
NonParametricStats <- NonParametricLambda.hat <- rep(0, B)
for (j in 1:B) {
    NonParametricData <- sample(x = data, size = n, replace = TRUE)  #Sample from data w/replac
    NonParametricLambda.hat[j] <- 1/mean(NonParametricData)  #get lambdahat for this sample
    bootstrapSE <- sqrt(NonParametricLambda.hat[j]^2/n)  #Get standard Error of each sample
    NonParametricStats[j] <- (NonParametricLambda.hat[j] - MLEMean)/bootstrapSE  #Boot t stats
}

# NonParametric Raw Percentile Interval
NonParametricBootstrap.RawQuantile <- quantile(NonParametricLambda.hat,
    c(0.025, 0.975))

CIs[i, ]$NonParBootLeft <- NonParametricBootstrap.RawQuantile[1]
CIs[i, ]$NonParBootRight <- NonParametricBootstrap.RawQuantile[2]
CIs[i, ]$NonParBootContainP <- CIs[i, ]$NonParBootLeft <= lambda & lambda <=
    CIs[i, ]$NonParBootRight

# NonParametric Reflected Percentile Interval
NonParReflectedInterval <- c(MLEMean - (quantile(NonParametricLambda.hat,
    0.975) - MLEMean), MLEMean - (quantile(NonParametricLambda.hat,
    0.025) - MLEMean))

CIs[i, ]$NonParReflLeft <- NonParReflectedInterval[1]
CIs[i, ]$NonParReflRight <- NonParReflectedInterval[2]
CIs[i, ]$NonParReflContainP <- CIs[i, ]$NonParReflLeft <= lambda & lambda <=
    CIs[i, ]$NonParReflRight

# NonParametric T Interval
NonParametric.bootstrapSE <- sqrt((1/B) * sum((NonParametricLambda.hat -
    mean(NonParametricLambda.hat))^2))
NonParametric.Bootstrap.TInterval <- c(MLEMean - quantile(NonParametricStats,
    0.975) * NonParametric.bootstrapSE, MLEMean - quantile(NonParametricStats,
    0.025) * NonParametric.bootstrapSE)
```

```
        CIs[i, ]$NonParTLeft <- NonParametric.Bootstrap.TInterval[1]
        CIs[i, ]$NonParTRight <- NonParametric.Bootstrap.TInterval[2]
        CIs[i, ]$NonParTcontainsP <- CIs[i, ]$NonParTLeft <= lambda & lambda <=
            CIs[i, ]$NonParTRight

    }

    return(CIs)

}

df <- run.CI(n, numSamples, lambda, B)
```

Below we plot a sample of the CI's from the collection of the parametric and non parametric raw confidence intervals.

A sample of the parametric bootstrap CI's

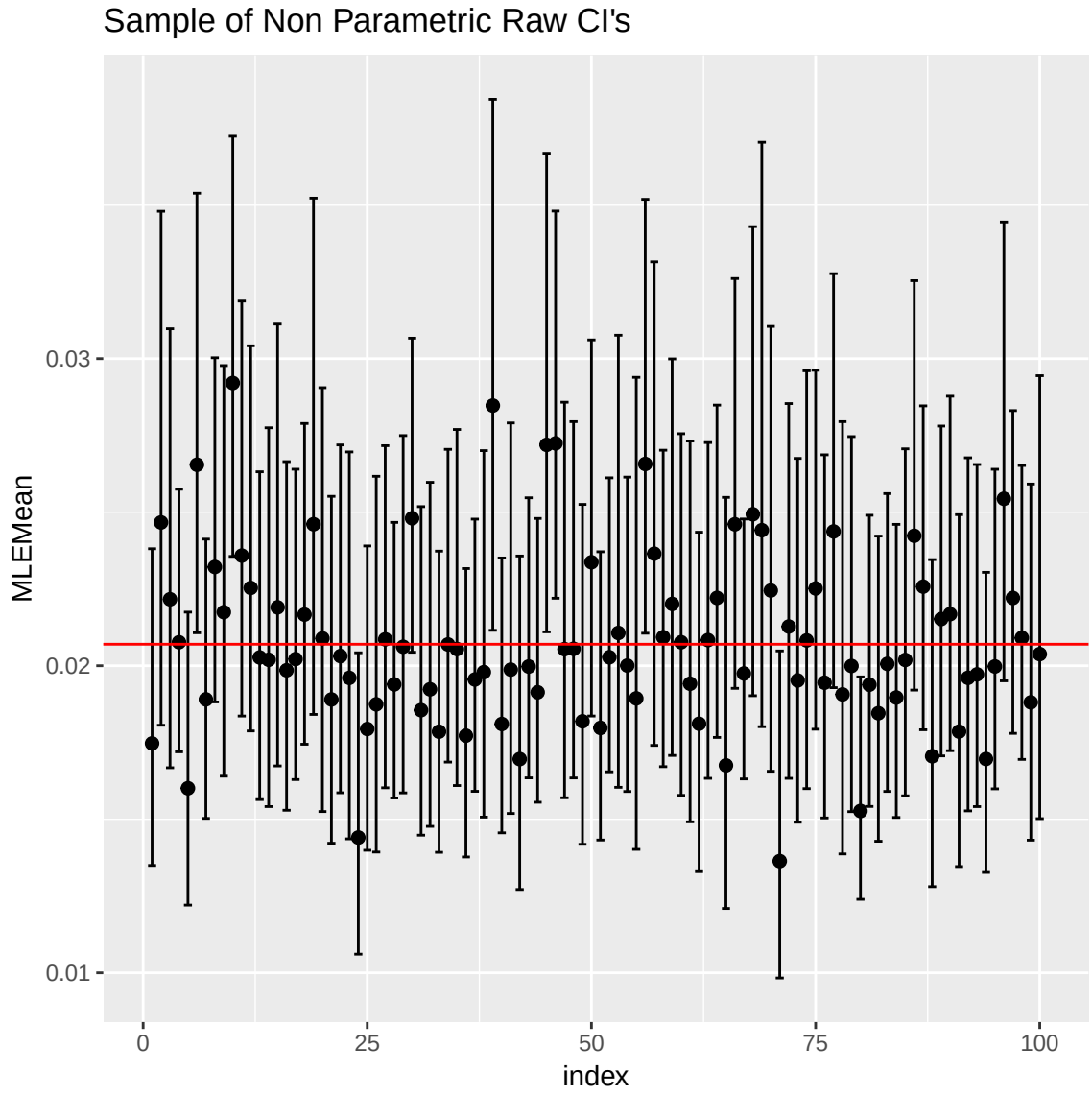
```
df$index <- 1:nrow(df)
require(ggplot2)
titleString <- "Sample of Parametric CI's "
ggplot(df[1:100, ], aes(x = index, y = MLEMean)) + geom_point(size = 2) + geom_errorbar(aes(ymax = BootR
    ymin = BootLeft)) + geom_hline(aes(yintercept = lambda), color = "red") +
    ggtitle(titleString)
```



##A

sample of the nonparametric raw bootstrap CI's

```
df$index <- 1:nrow(df)
require(ggplot2)
titleString <- "Sample of Non Parametric Raw CI's "
ggplot(df[1:100, ], aes(x = index, y = MLEMean)) + geom_point(size = 2) + geom_errorbar(aes(ymax = NonPa
    ymin = NonParBootLeft)) + geom_hline(aes(yintercept = lambda), color = "red") +
    ggtitle(titleString)
```



##Performance Characteristics

```
methodNames <- c("Exact", "LargeSample", "RawBootstrap", "Reflected", "Parametric T",
    "NonParametric Raw", "NonParametric Reflected", "NonParametric T")
numMethods <- length(methodNames)
Methods <- data.frame(Method = character(numMethods), Proportion = numeric(numMethods),
    ProportionAbove = numeric(numMethods), ProportionBelow = numeric(numMethods),
    AverageLength = numeric(numMethods), AverageLengthSE = numeric(numMethods))

Methods$Method <- methodNames

Methods[Methods$Method == "Exact", ]$Proportion = sum(df$ExactContainsP)/numSamples
Methods[Methods$Method == "Exact", ]$ProportionBelow = sum(df$ExactRight < lambda)/numSamples
Methods[Methods$Method == "Exact", ]$ProportionAbove = sum(df$ExactLeft > lambda)/numSamples
Methods[Methods$Method == "Exact", ]$AverageLength = mean(df$ExactRight - df$ExactLeft)
Methods[Methods$Method == "Exact", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$ExactRight - df$ExactLeft) - mean(df$ExactRight - df$ExactLeft))^2))
```

```
Methods[Methods$Method == "LargeSample", ]$Proportion = sum(df$LSampContainsP)/numSamples
Methods[Methods$Method == "LargeSample", ]$ProportionBelow = sum(df$LSampRight <
    lambda)/numSamples
Methods[Methods$Method == "LargeSample", ]$ProportionAbove = sum(df$LSampLeft >
    lambda)/numSamples
Methods[Methods$Method == "LargeSample", ]$AverageLength = mean(df$LSampRight -
    df$LSampLeft)
Methods[Methods$Method == "LargeSample", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$LSampRight - df$LSampLeft) - mean(df$LSampRight - df$LSampLeft))^2))

Methods[Methods$Method == "RawBootstrap", ]$Proportion = sum(df$BootContainP)/numSamples
Methods[Methods$Method == "RawBootstrap", ]$ProportionBelow = sum(df$BootRight <
    lambda)/numSamples
Methods[Methods$Method == "RawBootstrap", ]$ProportionAbove = sum(df$BootLeft >
    lambda)/numSamples
Methods[Methods$Method == "RawBootstrap", ]$AverageLength = mean(df$BootRight -
    df$BootLeft)
Methods[Methods$Method == "RawBootstrap", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$BootRight - df$BootLeft) - mean(df$BootRight - df$BootLeft))^2))

Methods[Methods$Method == "Reflected", ]$Proportion = sum(df$ReflContainP)/numSamples
Methods[Methods$Method == "Reflected", ]$ProportionBelow = sum(df$ReflRight <
    lambda)/numSamples
Methods[Methods$Method == "Reflected", ]$ProportionAbove = sum(df$ReflLeft >
    lambda)/numSamples
Methods[Methods$Method == "Reflected", ]$AverageLength = mean(df$ReflRight -
    df$ReflLeft)
Methods[Methods$Method == "Reflected", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$ReflRight - df$ReflLeft) - mean(df$ReflRight - df$ReflLeft))^2))

Methods[Methods$Method == "Parametric T", ]$Proportion = sum(df$TcontainsP)/numSamples
Methods[Methods$Method == "Parametric T", ]$ProportionBelow = sum(df$TRight <
    lambda)/numSamples
Methods[Methods$Method == "Parametric T", ]$ProportionAbove = sum(df$TLeft >
    lambda)/numSamples
Methods[Methods$Method == "Parametric T", ]$AverageLength = mean(df$TRight -
    df$TLeft)
Methods[Methods$Method == "Parametric T", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$TRight - df$TLeft) - mean(df$TRight - df$TLeft))^2))

Methods[Methods$Method == "NonParametric Raw", ]$Proportion = sum(df$NonParBootContainP)/numSamples
Methods[Methods$Method == "NonParametric Raw", ]$ProportionBelow = sum(df$NonParBootRight <
    lambda)/numSamples
Methods[Methods$Method == "NonParametric Raw", ]$ProportionAbove = sum(df$NonParBootLeft >
    lambda)/numSamples
Methods[Methods$Method == "NonParametric Raw", ]$AverageLength = mean(df$NonParBootRight -
    df$NonParBootLeft)
Methods[Methods$Method == "NonParametric Raw", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$NonParBootRight - df$NonParBootLeft) - mean(df$NonParBootRight -
        df$NonParBootLeft))^2))

Methods[Methods$Method == "NonParametric Reflected", ]$Proportion = sum(df$NonParReflContainP)/numSample
Methods[Methods$Method == "NonParametric Reflected", ]$ProportionBelow = sum(df$NonParReflRight <
```

```
    lambda)/numSamples
Methods[Methods$Method == "NonParametric Reflected", ]$ProportionAbove = sum(df$NonParReflLeft >
    lambda)/numSamples
Methods[Methods$Method == "NonParametric Reflected", ]$AverageLength = mean(df$NonParReflRight -
    df$NonParReflLeft)
Methods[Methods$Method == "NonParametric Reflected", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$NonParReflRight - df$NonParReflLeft) - mean(df$NonParReflRight -
        df$NonParReflLeft))^2))

Methods[Methods$Method == "NonParametric T", ]$Proportion = sum(df$NonParTcontainsP)/numSamples
Methods[Methods$Method == "NonParametric T", ]$ProportionBelow = sum(df$NonParTRight <
    lambda)/numSamples
Methods[Methods$Method == "NonParametric T", ]$ProportionAbove = sum(df$NonParTLeft >
    lambda)/numSamples
Methods[Methods$Method == "NonParametric T", ]$AverageLength = mean(df$NonParTRight -
    df$NonParTLeft)
Methods[Methods$Method == "NonParametric T", ]$AverageLengthSE = sqrt((1/numSamples) *
    sum(((df$NonParTRight - df$NonParTLeft) - mean(df$NonParTRight - df$NonParTLeft))^2))
```

Conclusions

Our objective with this study is to programmatically explore various procedures we can use with sample data to create a confidence interval around a statistic. We then want to evaluate how effective the various procedures are in capturing the true parameter in a reasonable interval.

Our process involved taking many samples (1000) from a known exponential distribution with rate parameter $\lambda = 0.0207$. For every one of these samples we then find the sample statistic $\hat{\lambda}$, and use 8 different procedures to create a 95% confidence interval around that statistic. We created an exact interval using the pivotal quantity and the known distribution. We created a large sample interval based on the asymptotic normal distribution. We also created intervals with 3 procedures (raw, reflected, and T) for both parametric and non-parametric bootstrap samples. The bootstrap methods required 5000 new samples for each of our 1000 samples. Finally we evaluated all 8 procedures for all 1000 samples to see how they performed with regard to capturing the true parameter, and how large the intervals needed to be in order to achieve their capture rate.

Results

There are 3 primary factors we looked at in assessing performance. The proportion of CI's which contain the true parameter, the average length of the CI, and the SE of the average length of the CI.

In considering the proportion it is important to recognize that our target rate is 95%, so the proportion needs to be close to that number to be an effective procedure. At the same time we think it is worth considering that a miss "high" is a good miss, i.e. the procedure is actually outperforming expectation when the proportion is greater than 95%. Similarly a miss "low" is a bad miss, not only does the procedure not meet the expectation but it is also underperforming.

We also look at the length of the intervals. We could achieve a very high proportion if we use very large intervals, so while the accuracy is great the information actually gleaned would be minimal. The standard error of the interval length must be taken into account as well. The larger this standard error is, the more variation you would get in the interval length created for a given sample. Knowing that in practice we will likely have a single sample that we are using to make inference, a smaller interval with minimal variance that also produces the expected proportion is the ideal procedure.

```
pander(Methods[, c("Method", "Proportion", "ProportionBelow", "ProportionAbove")],
    caption = "Performance Metrics - Proportions")
```

Table 3: Performance Metrics - Proportions

Method	Proportion	ProportionBelow	ProportionAbove
Exact	0.955	0.023	0.022
LargeSample	0.957	0.028	0.015
RawBootstrap	0.948	0.016	0.036
Reflected	0.945	0.05	0.005
Parametric T	0.96	0.021	0.019
NonParametric Raw	0.934	0.016	0.05
NonParametric Reflected	0.934	0.055	0.011
NonParametric T	0.933	0.03	0.037

```
pander(Methods[, c("Method", "AverageLength", "AverageLengthSE")], caption = "Performance Metrics - Con
```

Table 4: Performance Metrics - Confidence Intervals

Method	AverageLength	AverageLengthSE
Exact	0.01173	0.00164
LargeSample	0.01175	0.001643
RawBootstrap	0.01205	0.001784
Reflected	0.01205	0.001784
Parametric T	0.01204	0.001896
NonParametric Raw	0.01189	0.002355
NonParametric Reflected	0.01189	0.002355
NonParametric T	0.01186	0.003809

Looking at our results from the table above, we find 3 groupings of these procedures with regard to performance.

The exact and large sample are the best performers in our analysis, with proportions close to the 95% proportion expectation (both missing "high" - a good miss). They also have the smallest confidence interval lengths and standard errors. These procedures both use theoretical means to determine the CI from the MLE, so it makes sense that they perform the best.

Our next best group of performers are the parametric bootstrap results. They are all very close to the expected 95% proportion (or they miss high), have the larger average length for their CI than the other groups, but also significantly smaller SE than the non-parametric procedures. In our judgement the better performance in the "on target" rate along with this lower SE places them ahead of the non-parametric group. This also makes sense to us because we use a more rigorous theoretical approach here than in the non-parametric procedures, because we are pulling all of the bootstrap samples from the known parent distribution - just with the MLE estimator for rate.

Finally our least performant (but still very good) procedures are the non-parametric. They are all more than 1.5% off the target proportion, and they all miss low - underperforming expectation. The CI length is not as good as the exact and large samples, but they are still good, but the standard error values are half again to 2+ times the variation we see in the other methods. We think this bootstrap method is the least "rigorous" of the approaches in that it only samples repeatedly from values available in the initial sample, while the other methods consider all possible values present in the parent distribution. In other words it ignores the known fact that the parent distribution is exponential.