PSTAT 105 HW 7

Matthew Xu (5752811)

7 March 2021

```
# libraries
library(tidyverse)

# scan in signal data
signal <- read_delim("andro2.txt", " ", col_names = T)
```

Please complete these questions and submit the answers on GauchoSpace. Your calculations should be done in R, and your results should be typed up to include R input and output as well as clear explanations for your answers

1. The file andro2.txt contains the record of biological signal taken on a mouse subject. We are going to model this data as noisy measurements around an unknown mean function.

(a) Esimate the value of the mean function at time 200 by first averaging the 35 points closest to that time, and then average the 105 points closest to time 200. Give 95% confidence intervals for each of these estimates using a t statistic and estimating the variance from the data used in the mean.

```
n = 35
signal$dist_200 = abs(signal$Time - 200)
signal_35 = signal %>% arrange(dist_200) %>% top_n(-35, dist_200)
x_bar = mean(signal_35$Signal)

var = var(signal_35$Signal)
var
```

```
## [1] 0.01035496
```

```r
t_stat = qt(0.975, n - 1)
t_stat
```

```
## [1] 2.032245
```

```r
s = sum((signal_35$Signal - x_bar)^2)
# including zero includes the mean and lower/upper bounds 95% CI
CI = x_bar + c(-t_stat, 0, t_stat) * sqrt(s/n)
CI
```

```
## [1] -0.18125269  0.02257143  0.22639554
```

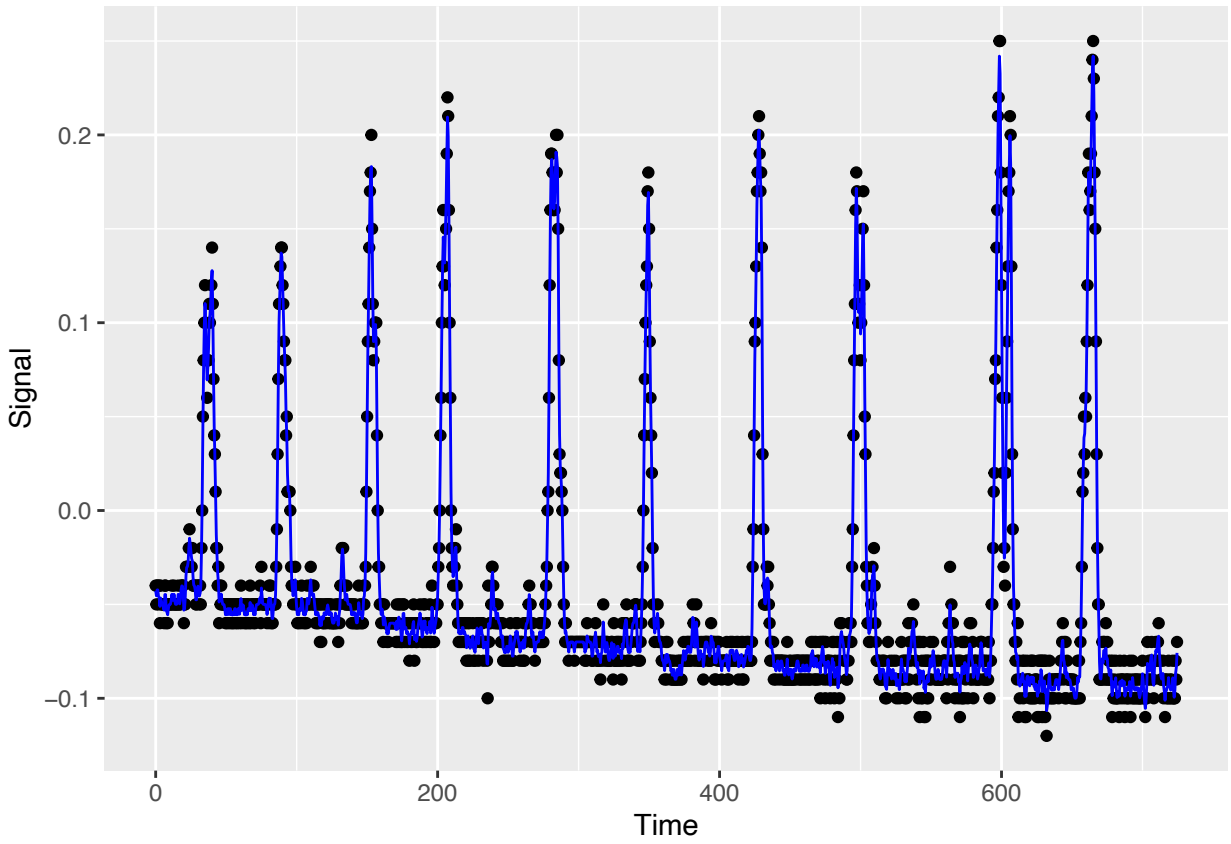
95% of the confidence can be explained between -0.18125269 and 0.22639554 with the true mean being within those bounds. THe variance of the mean is about 0.01035496. The test statistic is about 2.032245.

(b) In measuring the performance of a nonparametric regression estimator, we are looking at the variance and the bias of the estimate. Which of these two is not accounted for in your 95% confidence intervals from part (a)? Explain.

If the non paramaeric funtion is not normal, in reference to the t test statistic, the variance is usually underestimated. Confidence intervals require variance to be calculated, therefore accounting variance. Bias is not considered when calcuating confidence interval in performance of a nonparamteric regression estimator.

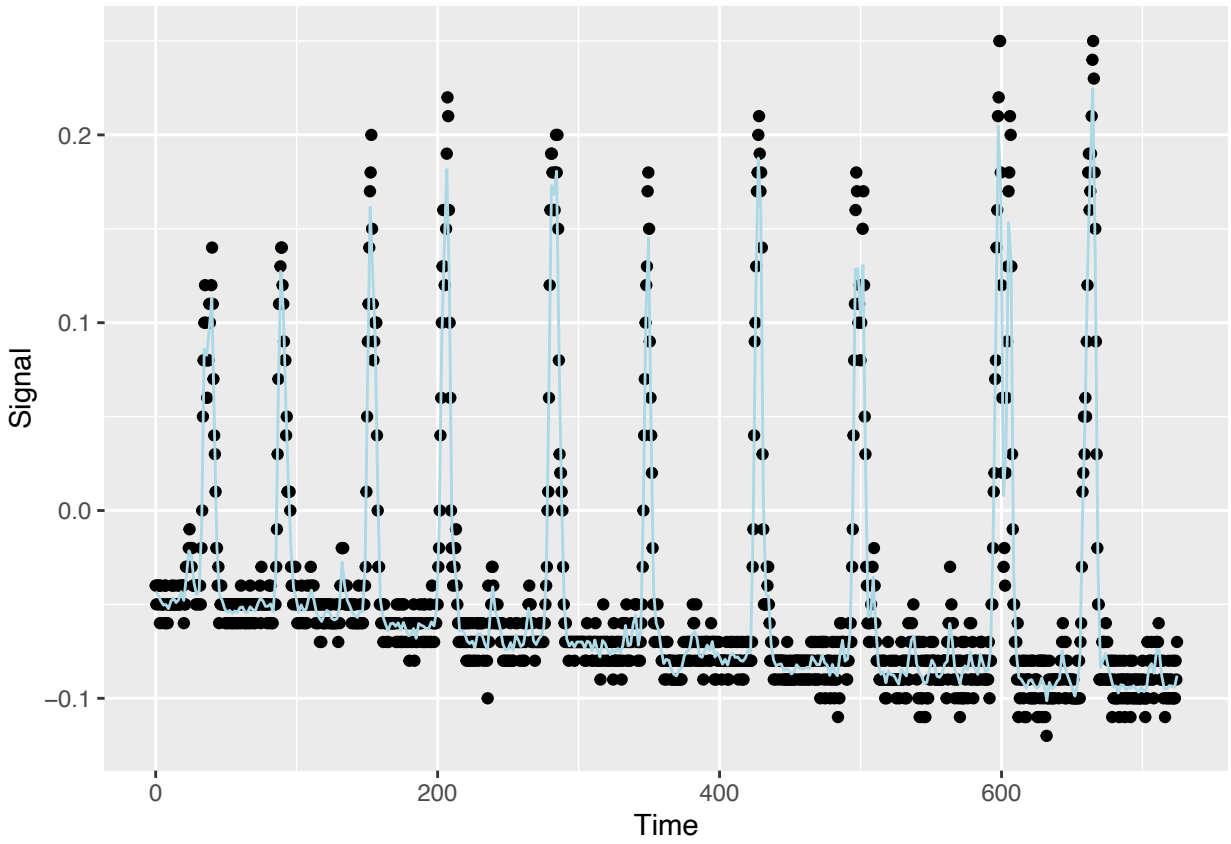
(c) Plot the data with time as the x axis and signal as the y axis. Calculate the Nadaraya-Watson kernel estimate using a Gaussian kernel, and plot the resulting estimate of a smooth mean function. Use a bandwdith that you think works well.

```r
k_est = ksmooth(signal$Time, signal$Signal, kernel = "normal", bandwidth = 1)
k_est_d = data.frame(k_est$x, k_est$y)

x = ggplot(signal, aes(x = Time, y = Signal)) + geom_point()
x + geom_line(data = k_est_d, aes(x = k_est.x, y = k_est.y), col = "blue")
```



(d) Make a new scatter plot and then add a local linear regression estimate using the locpoly function from the KernSmooth library. Choose a new bandwidth that fits the data well.

```
library(KernSmooth)
loc = locpoly(signal$Time, signal$Signal, kernel = "normal", bandwidth = 0.5)
loc_d = data.frame(loc$x, loc$y)

x + geom_line(data = loc_d, aes(x = loc.x, y = loc.y), col = "lightblue")
```



(e) The scientists were looking at these measurements for evidence of a sequence of peaks in the levels happening regularly over time. Use your estimates of the mean functions to locate the peaks in the data stream, and calculate the average amount of time between the peaks.

```
library(pracma)
loc_peaks = findpeaks(loc$y, npeaks = 10, threshold = 0.2, sortstr = TRUE)
l_est_peaks = findpeaks(k_est$y, npeaks = 10, threshold = 0.2, sortstr = TRUE)

loc_peaks
```

```
##           [,1] [,2] [,3] [,4]
## [1,] 0.2246474  368  361  371
## [2,] 0.1876849  237  234  244
## [3,] 0.1815803  115  110  120
## [4,] 0.1615326   85   82   92
## [5,] 0.1449079  194  190  200
```

```
l_est_peaks
```

```
##           [,1] [,2] [,3] [,4]
## [1,] 0.2419435 1198 1184 1205
## [2,] 0.2025090  857  845  866
```

```
## [3,] 0.1996350 1213 1205 1223
## [4,] 0.1693522  700  688  709
```

```
time_between_peaks1 = loc_peaks[1, 2] - loc_peaks[2, 2]
time_between_peaks2 = loc_peaks[1, 3] - loc_peaks[2, 3]
time_between_peaks3 = loc_peaks[1, 4] - loc_peaks[2, 4]
time_between_peaks4 = l_est_peaks[1, 2] - l_est_peaks[2, 2]
time_between_peaks5 = l_est_peaks[1, 3] - l_est_peaks[2, 3]
time_between_peaks6 = l_est_peaks[1, 4] - l_est_peaks[2, 4]

time_between_peaks1
```

```
## [1] 131
```

```
time_between_peaks2
```

```
## [1] 127
```

```
time_between_peaks3
```

```
## [1] 127
```

```
time_between_peaks4
```

```
## [1] 341
```

```
time_between_peaks5
```

```
## [1] 339
```

```
time_between_peaks6
```

```
## [1] 339
```

A peak is define here as the highest point of the distrbution using a spesfic kernel for a given amount of time. The peaks here are the high points on the plot after a set of low points. Certain time intervals have mutliple similar high points such as around 600 for the locpoly plot. We can access them thorugh the time intervals for where they occur in. We can use the findpeaks function to find the peaks at certain time points given the number of peaks and the threshold for finding the peaks. The time between peaks in loc is about 130. The time between peaks in l_est is about 340.

2. Use a simulation of one million random trials to approximate the probability for an exponential random variable X with $E(X) = 1$ of the event $A = \{\cos(X) > 0.7\}$. Calculate a margin of error for this approximation by using 95% confidence interval from a binomial experiment.

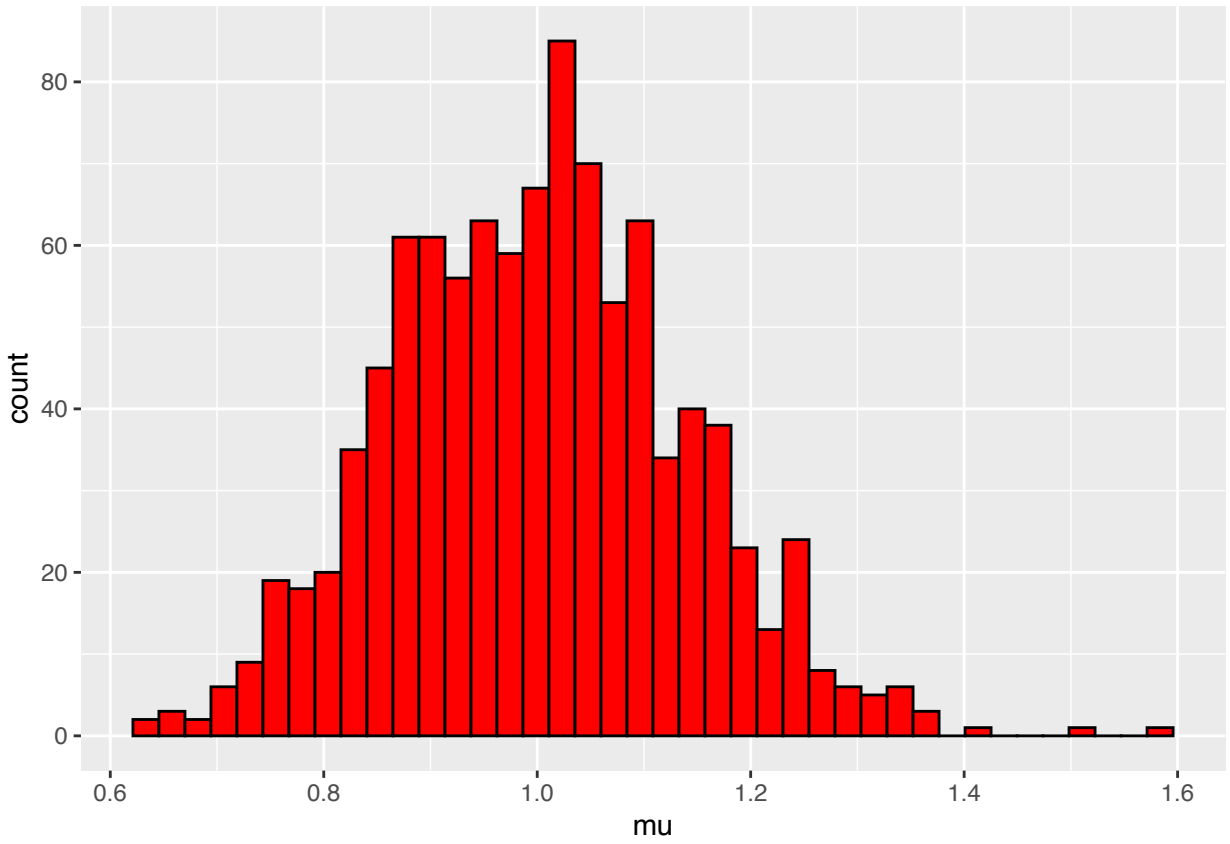
```
B = 100
n = 1e+06
p = NULL
samples = list()

for (i in 1:B) {
    samples[[i]] = rexp(n, 1)
    p[i] = length(which(cos(samples[[i]]) > 0.7))/n
}

mean(p) + c(-1.96, 0, 1.96) * sqrt(mean(p) * (1 - mean(p))/n * B)
```

```
## [1] 0.5421507 0.5518978 0.5616449
```

3. We want to calculate a critical value for a Goodness of Fit test based on the Kolmogorov–Smirnov test for an exponential distribution.

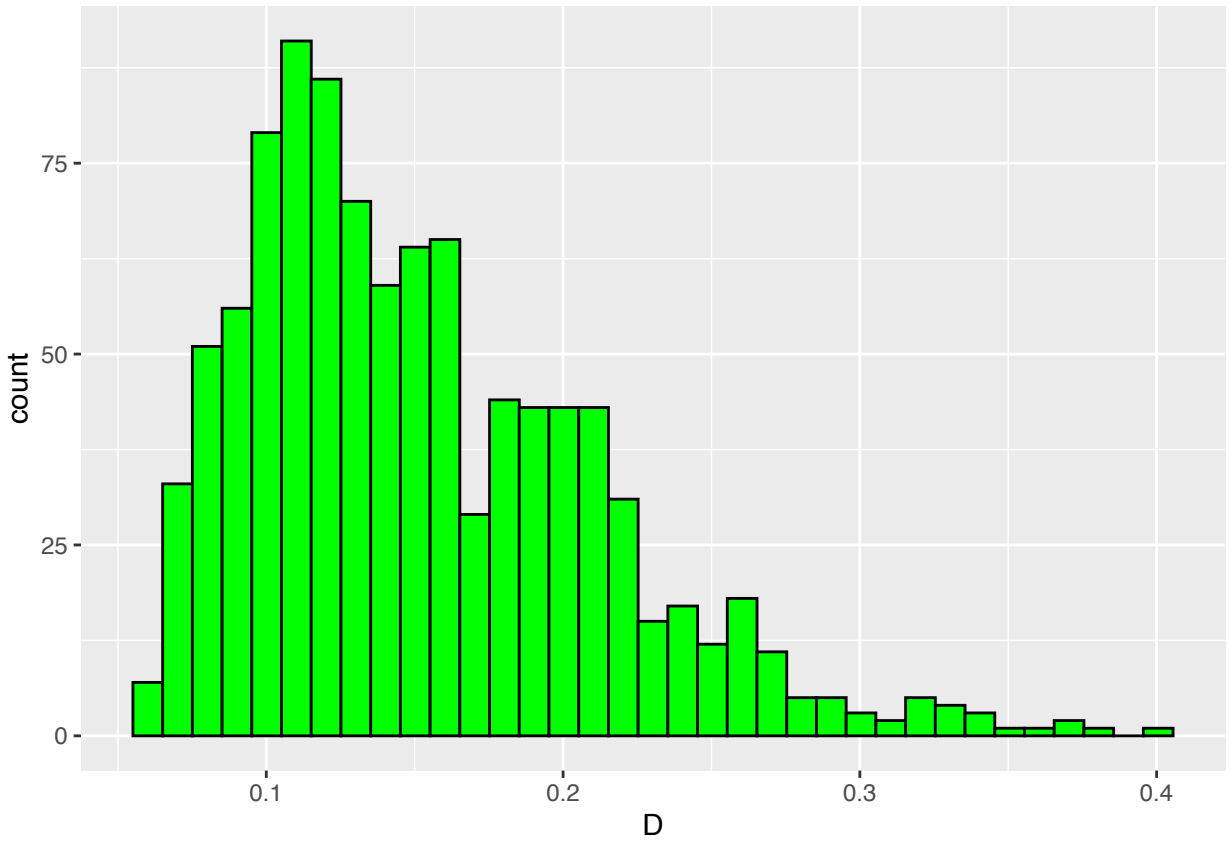
(a) Generate 1000 samples where each consists of 50 independent exponential random variables with mean 1. Estimate the mean of each sample. Draw a histogram of the means.

```
n2 = 50
samples2 = list()
mu = NULL

for (i in 1:1000) {
    samples2[[i]] = rexp(n2, 1)
    mu[i] = mean(samples2[[i]])
}

g = ggplot(data = data.frame(mu), aes(x = mu)) + geom_histogram(bins = 40, fill = "red",
    color = "black")
g
```



(b) Perform a KS test on each sample against the null hypothesis that they are from an exponential random variable with a mean that matches the mean of the data set. Draw a histogram of the 1000 values of D.

```
D = NULL

for (i in 1:1000) {
    D[i] = ks.test(samples2[[i]], "pexp", mu[i])$statistic
}

h = ggplot(data = data.frame(D), aes(x = D)) + geom_histogram(bins = 35, fill = "green",
    color = "black")
h
```



(c) From the simulated values of D, find an critical value c such that only 5% of the time will the test statistic D exceed that critical value.

```
quantile(D, 0.95)
```

```
##       95%
## 0.2604221
```

(d) Write up R code that will perform multiple batches of 1000 samples each consisting of 50 independent draws. Decide how many batches your computer can run in two hours, and run the program. Use the results to generate a more accurate estimate of c.

```
K = 5000
c = NULL
M = matrix(nrow = K, ncol = 1000)
a = Sys.time()
for (i in 1:K) {
    samples3 = list()
    for (j in 1:1000) {
        samples3[[j]] = rexp(50, 1)
        M[i, j] = ks.test(samples3[[j]], "pexp", mean(samples3[[j]]))$statistic
    }
```

```
    c[i] = quantile(M[i, ], 0.95)
}
b = Sys.time()
a - b
```

```
## Time difference of -16.26412 mins
```

```
mean(c)
```

```
## [1] 0.2678972
```

(e) Calculate the critical value that is suggested in Table 1.4 of Stephens(1974) paper. Compare this to the results from your simulations. Do you think this value is significantly different from your simulation results?

```
# n value in D.star is value in rexp not simulation number
(D.star <- 0.895/(sqrt(50) - 0.01 + 0.85/sqrt(50)))
```

```
## [1] 0.1246297
```

```
t.test(c, mu = D.star)
```

```
##
##  One Sample t-test
##
## data:  c
## t = 1685.4, df = 4999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0.1246297
## 95 percent confidence interval:
##  0.2677305 0.2680638
## sample estimates:
## mean of x
## 0.2678972
```

No, they are quite similar numerically with the 95% percentile using D values at about 0.2648041, the simulated 95% of mean at about 0.2668, and the calculated mean using Stephens paper at about 0.2671479. They are very close to each other.