```
---
title: "HW3"
author: "STAT 28"
output:
  pdf_document
header-includes:
- \usepackage{framed}
- \usepackage{xcolor}
- \let\oldquote=\quote
- \let\endoldquote=\endquote
- \colorlet{shadecolor}{orange!15}
- \renewenvironment{quote}{\begin{shaded*}\begin{oldquote}}
{\end{oldquote}\end{shaded*}}
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, tidy.opts=list(width.cutoff=60), tidy=TRUE)
```
```

1. Answer the following TRUE/FALSE questions, and explain your reason.
      a. (5 points) Bootstrap confidence intervals are better than parametric confidence intervals based on the t/normal distribution because bootstrap confidence intervals do not make any assumptions.

      > False. The big assumption of the bootstrap is that our sample distribution is a good estimate of the true distribution. It will not necessarily. For example, when the sample size is too small or the data is not SRS, the test might fail.

      b. (5 points) Assume I make a 99% confidence interval of the difference in the means between two groups, based on the parametric normality assumptions. If this confidence interval does not cover 0, then I can reject the null hypothesis that the means are equal at level 0.05.

      > True. If you create a 95% confidence interval, and then decide to reject a specific null hypothesis only when it does not fall within the confidence interval, then this will exactly correspond to a test with level 0.05. A 99% confidence interval would contain the 95% confidence interval. If the 99% confidence interval does not cover 0, neither will the 95% confidence interval. Thus, at level 0.05, you would reject the null.

      c. (5 points) Suppose I test whether the means of two groups are equal, and get a p-value of 0.05. Then the probability that the means are not equal is 0.95.

      > False. The null hypothesis: the mean of two groups are equal. The null is either true or false, and there is no randomness in the correctness of the null. A p-value of 0.05 tells us that when the null is true, the probability of getting a statistic as extreme as the observation is 0.05.

2. (5 points) Consider the question of comparing the difference between two groups. We decide to use the difference in the means to evaluate whether there is any significant difference between the two groups. One of the histograms below is the histogram of difference in the medians from the permutations in the permutation test comparing these two groups. The other histogram is the histogram of the difference in the medians from bootstrap resampling of the data to create confidence intervals. Which is which? (explain your reasoning)

 > Plot A corresponds to the bootstrap and plot B corresponds to the permutation test. The distribution of bootstrapped statistics will center around the mean difference and the permuted statistics would center around zero.

```{r bringInBlindData}
    blindData<-read.table("blindData.txt",sep="\t",header=FALSE)
    par(mfrow=c(1,2))
    hist(blindData[,1],breaks=100,main="Plot A",xlab="")
    hist(blindData[,2],breaks=100,main="Plot B",xlab="")
```

3. Consider the bootstrap function I wrote in class for getting bootstrap confidence intervals for the results of fitting a line to the data using squared error.

```{bootstrapLmFunction}
    bootstrapLM <- function(y,x, repetitions, confidence.level=0.95){
      stat.obs <- coef(lm(y~x))
      bootFun<-function(){
            sampled <- sample(1:length(y), size=length(y),replace = TRUE)
            coef(lm(y[sampled]~x[sampled]))
      }
      stat.boot<-replicate(repetitions,bootFun())
      #this next line is advanced, but simply finds the name of the input x value.
      nm <-deparse(substitute(x))
      row.names(stat.boot)[2]<-nm
      level<-1-confidence.level
      confidence.interval <- apply(stat.boot,1,quantile,probs=c(level/2,1-level/2))  ###LINE 12
      out<-cbind("lower"=confidence.interval[1,],"estimate"=stat.obs,"upper"=confidence.interval[2,])
      return(list(confidence.interval = out, bootStats=stat.boot))
    }
```

a. (5 points) Explain what the function `bootFun` does. Specifically, what does the two lines of code within `bootFun` do, what type of output does `bootFun` return (a character string? A number? A vector? A list? A matrix?)

> The first line of `bootFun` samples without replacement from the indices of the response variable. `y[sampled]` and `x[sampled]` are the bootstrapped sample. The second line of `bootFun` fits the linear regression model using the bootstrapped sample and return the estimated coefficients (which is a numeric vector of length 2).

b. (5 points) Explain what type of object `stat.boot` is (a character string? A number? A vector? A list? A matrix?) and describe what are its actual entries.

> `stat.boot` is a matrix with 2 rows and number of columns equals to `repetition`. The first row is the bootstrapped intercept and the second row is the bootstrapped slope.

c. (5 points) Explain what LINE 12 of the code does (marked above).

> The `apply` function calculate the `level/2` and `1-level/2` quantile for each row of the `stat.boot` matrix (which is the confidence intercal). Thus, this line of code calculates the bootstrapped confidence intervals for the intercept and slope.

4. In the file `fitbit.csv` we have information from a fitbit device for a single

person over the year. A fitbit device is something someone wears that records information regarding your activites throughout the day: your level of activity, the number of calories burned, the amount slept, etc (the company's website is www.fitbit.com).

The data that we have here is summarize per day, so that each entry of the data corresponds to a particular date.

The following code should read in the data and print out data from the first 5 rows of the first 8 columns.

```{r readInFitbit}
      fitbit<-read.csv("fitbit.csv",header=TRUE)
      head(fitbit[,1:8])
```

a. (5 points) Many of the variables in this data set quantify the number of minutes spent doing activities, including sleeping. In principle, we might assume that these minutes should cover the whole day. However, the device might have been turned off or taken off at some point.

Let's consider this question. Look at the following variables: `MinutesOfSleep`, `minutesOfLightActivity`, `minutesOfModerateActivity`, `minutesOfSedentaryActivities`, and `minutesOfIntenseActivity`. How well do these sum up to cover the entire day?

      > (We give an example as follows, you do not need to follow the code exactly. As long as you calculate the correct total number of minutes, you will get the full credit.) The sum may exceed the total number of minutes in a day. Most days are well covered, but some days are poorly covered.

```{r totalMinutes}
# code to evaluate the total number of minutes
totalNumberOfMinutes <- fitbit$MinutesOfSleep + fitbit$minutesOfLightActivity + fitbit$minutesOfModerateActivity + fitbit$minutesOfSedentaryActivities + fitbit$minutesOfIntenseActivity
summary(totalNumberOfMinutes)
hist(totalNumberOfMinutes / (60*24), main = "Histogram of the proportion of minutes covered",
     xlab = "Proportion of minutes covered", breaks = 50)
```

b. (5 points) Going forward, we are going to consider the question of whether the amount of sedentary activity is predictive of the minutes of sleep per night. We are going to first normalize our data, so that both of these quantities are out of the total amount of minutes recorded in the day with the device. Calculate the total number of minutes recorded each day in the variables from part a, and make new variables `pctAsleep` and `pctSedentary` that are the percentage of all tracked minutes. Print out a `summary` of each of these variables to show that it worked.

```{r makePercentVariables}
      #code to make new variables as percent of total minutes.
pctAsleep <- fitbit$MinutesOfSleep / totalNumberOfMinutes
pctSedentary <- fitbit$minutesOfSedentaryActivities / totalNumberOfMinutes
summary(pctAsleep)
summary(pctSedentary)
```

c. (10 points) Make a plot of `MinutesAsleep` against `minutesOfSedentaryActivites` and another plot of `pctAsleep` against `pctSedentary`. For the minutes plot, convert the minutes into hours. Use `par(mfrow= ...)` to make the two plots side-by-side, and label the two plots appropriately. Comment on the affect of converting them to percentages.

```{r plotAsleepVsSedentary}
      # code to plot the two plots side-by-side
par(mfrow = c(1, 2))
plot(fitbit$minutesOfSedentaryActivities/60, fitbit$MinutesOfSleep/60,
     ylab = "Hours Of Sleep", xlab = "Hours Of Sedentary Activities")
plot(pctSedentary, pctAsleep,
     ylab = "Percentage Of Sleep", xlab = "Percentage Of Sedentary Activities")
```

d. (10 points) Plot the variable `pctAsleep` against `pctSedentary` like above, only now draw in the least squares regression line. Also print out the coefficients of the lines. Interpret the line and plots; be specific to this data -- e.g. as you increase percent of time spent in sedentary minutes by 10%, how is `pctAsleep` predicted to change?

> (Open-ended, but need to show the amount of time increase in sleep when increasing the percentage of sedentary) The fitted line shows that the percentage of sleep is positively correlated with the percentage of sedentary. The coefficients tell that if we increase the percent of time spent in sedentary minutes by 10%, the percentage of sleep will increase by 1.493%.

```{r plotRegression}
      # Code here for plot with regression line
d.fit <- lm(pctAsleep~pctSedentary)
d.fit.coef <- d.fit$coefficients
d.fit.coef
# or
# d.fit.coef <- coef(d.fit)
plot(pctSedentary, pctAsleep,
     ylab = "Percentage Of Sleep", xlab = "Percentage Of Sedentary Activities")
abline(d.fit.coef[1], d.fit.coef[2], col = "red")
```

e. (10 points) We might consider the effect of the (practically) zero levels of sedentary activity. Remove these points from the above analysis in d, and compare the results to that found in d, and draw conclusions as to the effect of these data points. What conclusion would you draw as to whether they should be included in the analysis?

 > The fitted line in d does not seem to across the major trend of the data because the fitting is largely influenced by the points with (practically) zero levels of sedentary activity. After removing those "outliers", we would found that the fitted line becomes more reasonable. Thus, for the purpose of prediction, those points need to be removed.

```{r plotRegressionNoZero}
      # Code here for plot with regression line removing zero sedentary
e.fit <- lm(pctAsleep[pctSedentary>=0.2]~pctSedentary[pctSedentary>=0.2]) # you can choose the cutoff your self as long as they amke sence(remove the low activity data but not too large)
e.fit.coef <- e.fit$coefficients
e.fit.coef
```

```
# or
# d.fit.coef <- coef(d.fit)
plot(pctSedentary[pctSedentary>=0.2], pctAsleep[pctSedentary>=0.2],
     ylab = "Percentage Of Sleep", xlab = "Percentage Of Sedentary Activities")
abline(e.fit.coef[1], e.fit.coef[2], col = "red")
```

f. (10 points) Create bootstrap confidence intervals as well as parametric confidence intervals for the coefficients of the regression you found in e. (you can use the code from question 3). Plot the resulting confidence intervals and compare them.

> The confidence interval from bootstrap and parametric test are very close. However, the bootstrapped confidence interval is a bit wider than the parametric one.

```{r bootCI}
# Code here bootstrap and parametric confidence intervals
bootstrapLM <- function(y,x, repetitions, confidence.level=0.95){
        stat.obs <- coef(lm(y~x))
        bootFun<-function(){
              sampled <- sample(1:length(y), size=length(y),replace = TRUE)
              coef(lm(y[sampled]~x[sampled]))
        }
        stat.boot<-replicate(repetitions,bootFun())
        #this next line is advanced, but simply finds the name of the input x value.
        nm <-deparse(substitute(x))
        row.names(stat.boot)[2]<-nm
        level<-1-confidence.level
        confidence.interval <- apply(stat.boot,1,quantile,probs=c(level/2,1-level/2))  ###LINE 12
        out<-cbind("lower"=confidence.interval[1,],"estimate"=stat.obs,"upper"=confidence.interval[2,])
        return(list(confidence.interval = out, bootStats=stat.boot))
}
pctAsleepCleaned <- pctAsleep[pctSedentary>=0.2]
pctSedentaryCleaned <- pctSedentary[pctSedentary>=0.2]
bootLM <- bootstrapLM(pctAsleepCleaned, pctSedentaryCleaned, repetitions = 1000)
permutConfInt <- bootLM$confidence.interval
paramConfInt <- confint(e.fit)

b1CI<-rbind(c(lower=paramConfInt[2,1],
              estimate=unname(coef(e.fit)[2]),
              upper=paramConfInt[2,2]),
              permutConfInt[2,])
print(b1CI)
b0CI<-rbind(c(lower=paramConfInt[1,1],
              estimate=unname(coef(e.fit)[1]),
              upper=paramConfInt[1,2]),
              permutConfInt[1,])
print(b0CI)

# plot
library(gplots)

par(mar=c(8,4,4,1))
plotCI(b1CI[,2],ui=b1CI[,3] ,li=b1CI[,1],main="compare confidence intervals for the
```

```
slope",
       col=c("blue","red"),pch=19,xaxt="n",xlim=c(0,3))
axis(1,at=c(1,2),labels=c("Parameteric","Bootstrap"),las=2)

par(mar=c(8,4,4,1))
plotCI(b0CI[,2],ui=b0CI[,3] ,li=b0CI[,1],main="compare confidence intervals for the
intercept",
       col=c("blue","red"),pch=19,xaxt="n",xlim=c(0,3))
axis(1,at=c(1,2),labels=c("Parameteric","Bootstrap"),las=2)
```

g. (5 points) These data are collected on the same person over a year. What problem does this create in the validity of the parametric confidence intervals? Is this a problem for the validity of the bootstrap intervals?

> Since the data are collected on the same person over a year, it is very likely that the data will be correlated with each other. For example, the amount of sleep is usually influenced by the amount of sleep in the previous day. It is also a problem for the validity of the bootstrap intervals because bootstrap also requires the independence of samples.

h. (5 points) We've seen that we do not always have all of the minutes of a day recorded. What affect could these missing minutes have on our above analysis? Be specific. A good idea is to first try to think of extreme examples of what could be happening in those minutes that might change your conclusion, so you can think about why it might be a problem. Then try to think of the best case scenario where its not affecting your results. Then return to the actual data and think how plausible it is to be worried about this problem.

> Our results will be biased if the missing minutes are not missing at random. For example, there may be some problem with the device, so it only fails to record the sleep but works for other activities. Then the minutes of sleeping would increase dramatically for part of the data. The best case scenario for our regression (using sedentary to predict sleep) is that the missing minutes account for a fixed proportion of the total actual minutes in each activity, so the proportion of sleep and sedentary are not influenced. However, it seems not the case for our Fitbit dataset. Thus the regression results could be biased.