Introduction to the Bootstrap

STA 325: Lab 6, Fall 2018

Today's agenda: Introduction to the Bootstrap

In this lab, we'll investigate the bootstrap using linear regression.

Let's first read in the dataset `bootstrap-data.csv`

```
my.data <- read.csv('data/bootstrap-data.csv', header=TRUE)
head(my.data)
```

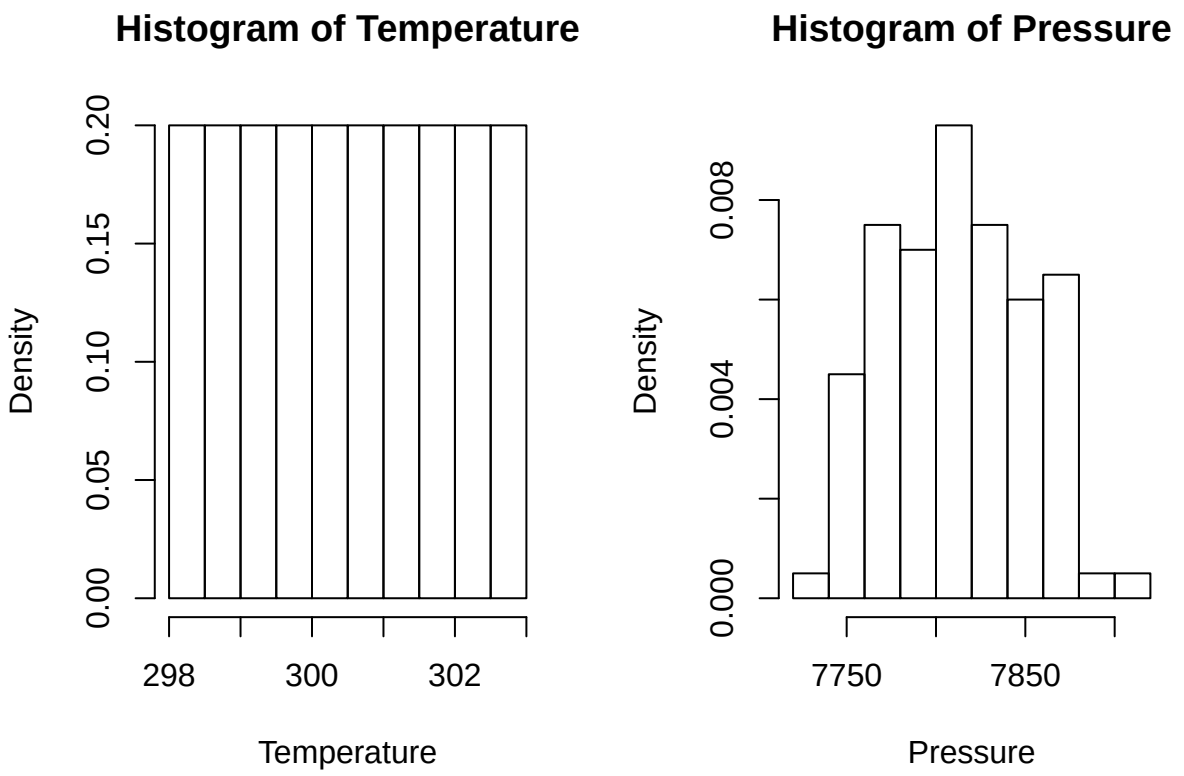
```
##   temperature pressure
## 1    298.0000 7767.519
## 2    298.0505 7744.627
## 3    298.1010 7762.836
## 4    298.1515 7756.751
## 5    298.2020 7760.371
## 6    298.2525 7753.842
```

```
suppressWarnings(library(xtable))
```

Task 1

We begin by examining histograms for both variables.

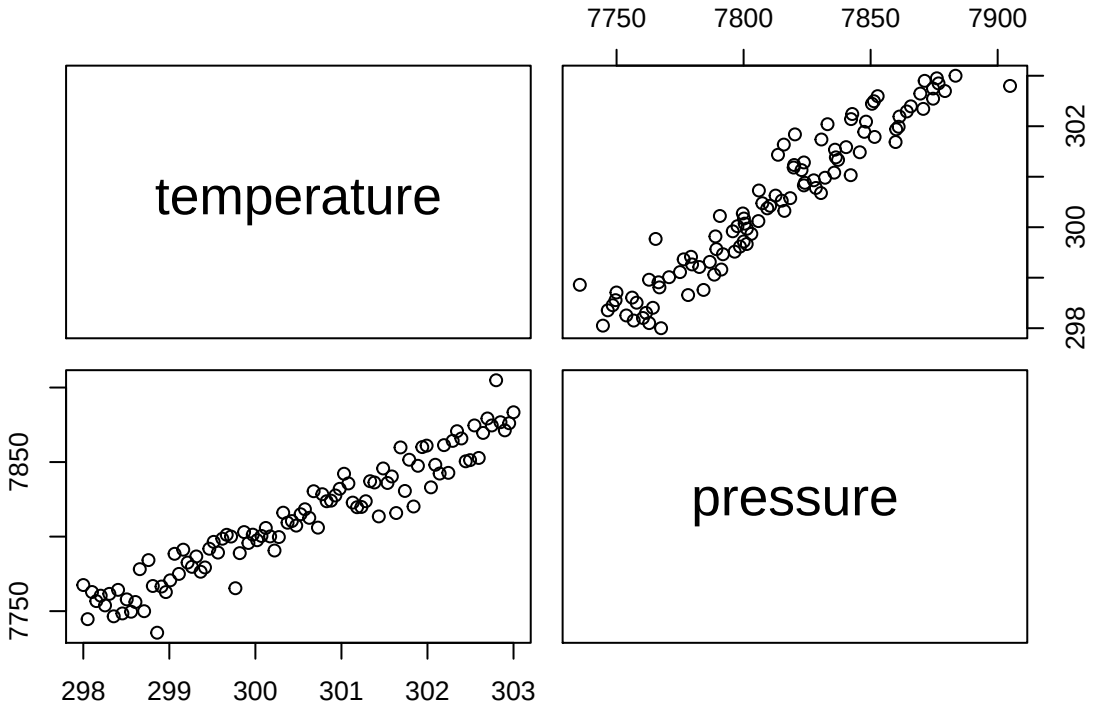
```
par(mfrow = c(1,2))
hist(my.data$temperature,
     freq = FALSE,
     main = "Histogram of Temperature",
     xlab = "Temperature")
hist(my.data$pressure,
     freq = FALSE,
     main = "Histogram of Pressure",
     xlab = "Pressure")
```



Temperature appears to have a uniform distribution; it is clearly not a normal distribution. Pressure could possibly be normally distributed, but we have a small simple size and cannot confirm this. The distribution appears to be somewhat symmetric and unimodal.

We can also look at the relationship between these variables in a scatter plot.

```
pairs(my.data)
```



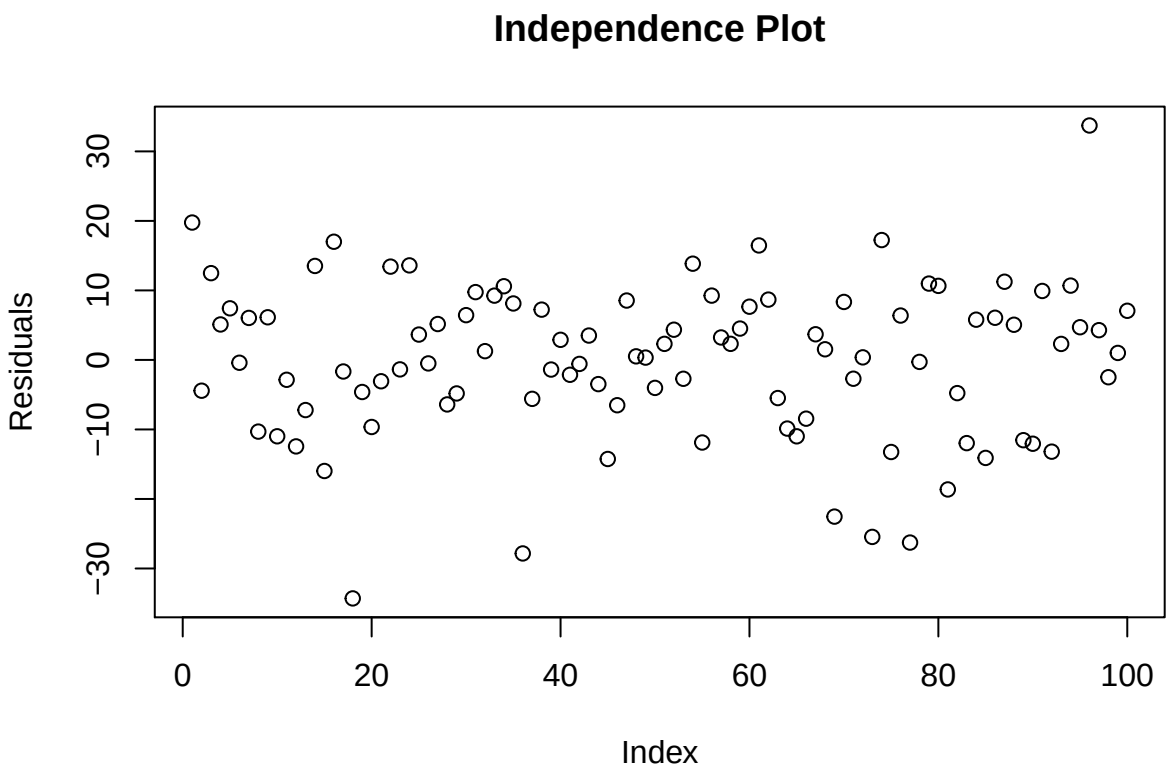
It appears that the variables exhibit a fairly strong linear relationship. Based on the plots, there is no apparent need to consider more complex (i.e. quadratic) models.

Now we run a linear regression.

```
lin.reg <- lm(pressure ~ temperature, data = my.data)
```

First we will create an independence plot for the residuals to analyze the fit of the model.

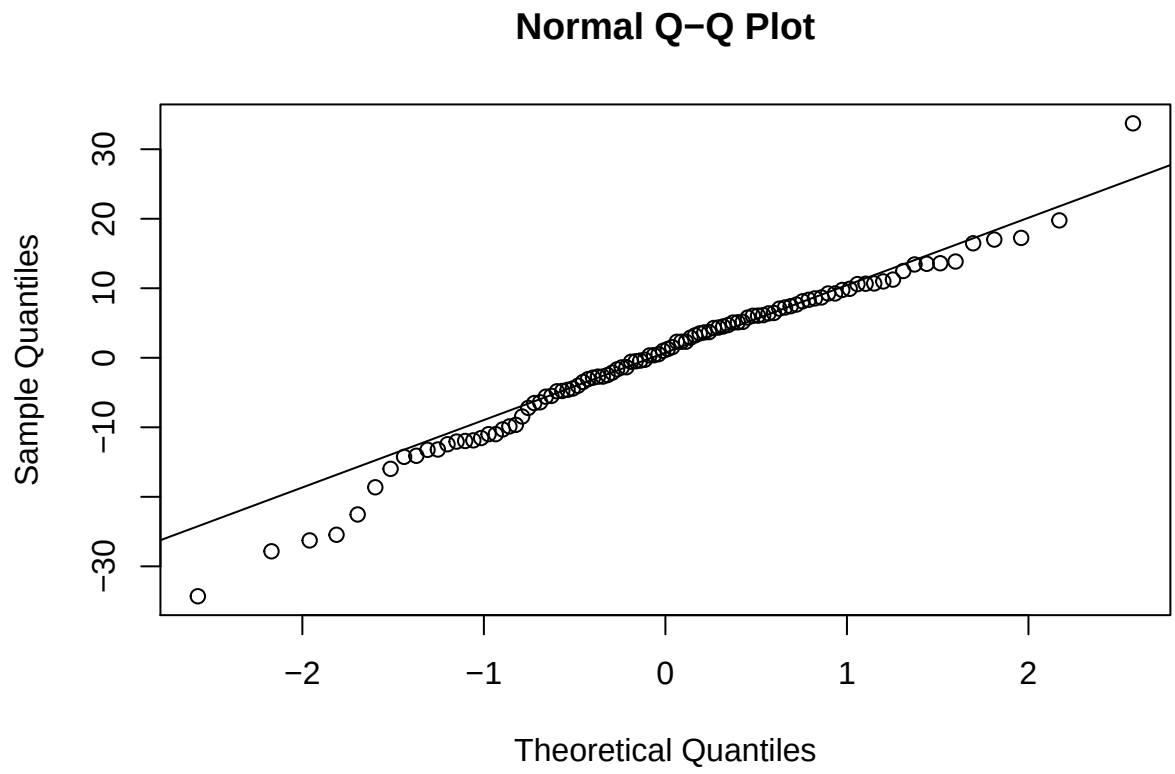
```
plot(lin.reg$residuals, main = "Independence Plot", ylab = "Residuals")
```



There is not much evidence of autocorrelation; the results resemble random noise. The residuals do not tend to increase or decrease in absolute value as the index changes. There are some points that may be outliers and that we could consider removing.

Next we create a QQ-plot for the residuals.

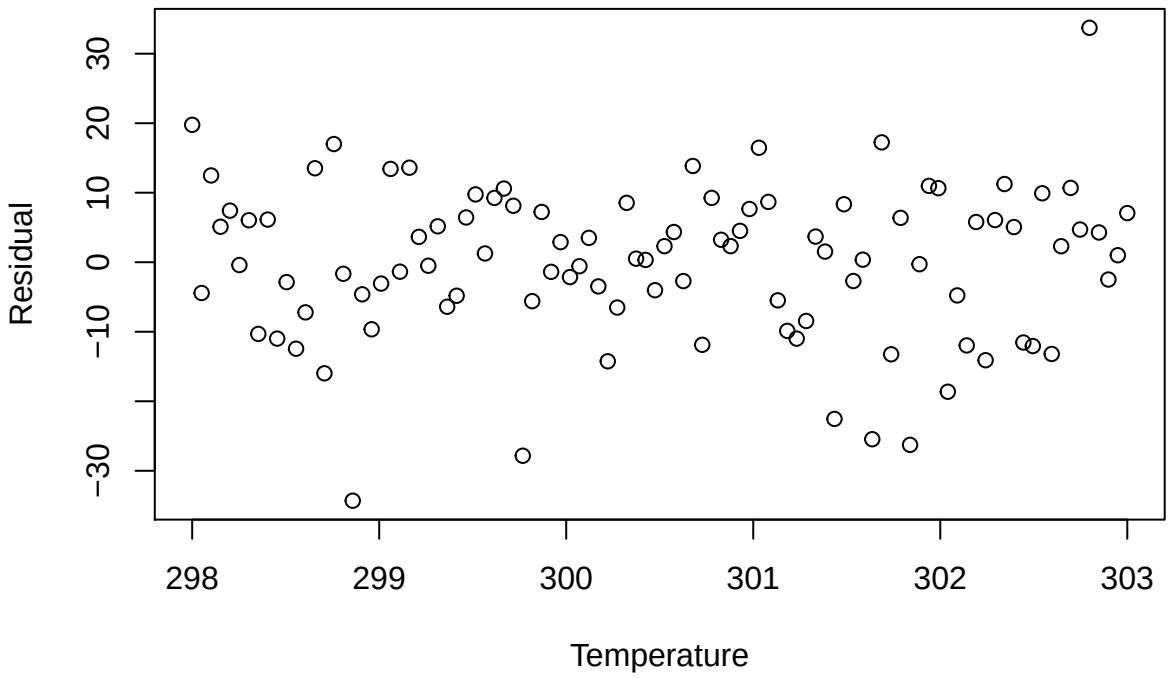
```
qqnorm(lin.reg$residuals)
qqline(lin.reg$residuals)
```



The tails of the sample distribution diverge from the theoretical quantiles of the normal distribution. This means that we have reason to doubt the assumption that the residuals are normally distributed, which is one of the key assumptions for performing inference on a linear model.

Finally we plot residuals vs `temperature`.

```
plot(x = my.data$temperature, y = lin.reg$residuals,
     xlab = "Temperature", ylab = "Residual")
```



Again, this is a fairly good residual plot. There is no trend in the residuals that is easily discernible. There are, however, some apparent outliers that we could consider removing from the data.

The linear regression is fairly sound. The residuals, while not necessarily normally distributed, are free from other issues, such as homoscedasticity. The results with the standard significance measures (which may not be appropriate in this case) are displayed below.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	83.3452	228.4621	0.36	0.7160
temperature	25.7195	0.7603	33.83	0.0000

The intercept does not appear to be significantly different from zero, while the coefficient of `temperate` is highly significant. These conclusions depend on assumptions about the distribution of the residuals that may not be satisfied. We will now write code to construct a confidence interval for these estimates.

Task 2

In this task we will write a function that will randomly sample rows (observations) from a data frame with replacement. It will return data frame with the same number of rows as the original data frame.

```
resample <- function(df){
################################################################
# This function resamples rows of a data frame with replacement
#
# Inputs:
#   df:  a data frame which will be resampled
#
# Output:
#  A data frame with the same number of rows as initial data frame
################################################################
  row.sample <- sample(1:nrow(df), # sample row numbers
                       size = nrow(df), # same size as input
                       replace = TRUE) # w/replacement
  new.df <- df[row.sample,] # select random rows
  row.names(new.df) <- 1:nrow(df) # rename rows
  return(new.df) # return result
}
```

Task 3

Now we write a function that takes a data frame, resamples the rows, and runs a the linear model specified by a string formula on the resampled data.

```
reModel <- function(df, str.formula){
################################################################
# This function runs a specified regression on resampled data
#
# Inputs:
#   df:  a data frame to be resampled
#   str.formula:  a string specifying a linear model for the data in df
#
# Outputs:
```

```
#   An lm object, the results of a linear regressoin
#############################################################
  new.df <- resample(df) # use previous function
  formula <- as.formula(str.formula) # convert string to formula
  resam.reg <- lm(formula, data = new.df) # run regression
  return(resam.reg) # return regression object
}
```

Task 4

In this task we will use our previous function to write a new function that will run a specified number of bootstrap replications and return the coefficients for the results in a data frame. Note that the function `reModel` returns an `lm` regression object and we must extract the coefficients from the object.

```
runBoot <- function(df, str.formula, reps){
#############################################################
# This function runs a specified number of bootstrap samples
#
# Inputs:
#   df:  data to be resampled
#   str.formula: string containing model formula
#   reps:  number of bootstrap repetitions to be performed
#
# Outputs
#   A data frame where each row contains the coefficients from a bootstrap
#############################################################
  list.of.regs <- lapply(1:reps, # repeat specified number of times
                         function(x) {
    # run reModel and extract coefficients from regression
    return(reModel(df, str.formula)$coefficients)
                           })
  res.df <- do.call(rbind, list.of.regs) # bind together into df
  return(res.df) # return result
}
```

Task 5

Using the function `runBoot` we will construct confidence intervals at a specified level `alpha`.

```
makeCI <- function(reps, alpha, df, str.formula){
#############################################################
# This function runs a bootstrap a specified number of times and
# constructs confidence intervals at a specified level
#
# Inputs:
#   reps:  number of iterations of the bootstrap
#   alpha:  significance level, type I error rate
#   df:  data frame
#   str.formula:  string representing regression model
#
# Output:
```

```
#   a data frame with a confidence interval in each row
#############################################################
  res.df <- runBoot(df, str.formula, reps) # get bootstrap estimates
  # get original bootstrap estimates
  orig.est <- lm(as.formula(str.formula), data = df)$coefficients
  # next lines compute upper and lower bounds
  lower <- 2*orig.est - apply(res.df, 2, quantile, 1- alpha/2)
  upper <- 2*orig.est - apply(res.df, 2, quantile, alpha/2)
  c.ints <- cbind(lower, upper) # bind together
  return(c.ints) # return result
}
```

We will now run this function on `my.data` for 10,000 repetitions to construct an interval with a Type I error rate of 0.05.

```
# need to specify model as a string
my.formula <- "pressure ~ temperature"
conf.ints <- makeCI(10000, .05, my.data, my.formula)
```

	lower	upper
(Intercept)	-400.30	570.69
temperature	24.09	27.33

As zero is in the confidence interval for the intercept, we cannot conclude that the intercept is not zero. This is the same conclusion we derived from the t-test results given in Part 1. The confidence interval for `temperature` does not contain zero, so we could conclude that `temperature` has some effect on `pressure` and that this effect is most likely positive. Again, this is similar to the result in Part 1, where we observed a very small p-value for the estimate.