Handout 6

Put your name here!

For this week only, the questions at the end will not be handed in. Instead please hand in the code from the examples in an R Script. We will do the questions in precept.

In this handout, we will learn how to calculate a single loop.

Topics and Concepts Covered

- Building a loop
- Bootstrapping
- Using a loop to adjust rows or columns of a matrix
- Creating a nested loop
- *Indicator variable*: A variable that takes on a value of 1 if a particular condition is met, and 0 otherwise
- *Standard error of sample mean*: If $\overline{Y}$ is the sample mean of some set of observations, let $\widehat{\sigma}_Y$ denote the sample standard deviation of Y.

The standard error of the sample mean, $\overline{Y}$ is

$$\frac{\widehat{\sigma}_Y}{\sqrt{n}}$$

and the 95% confidence interval, using the Normal Approximation, is

$$\left[\overline{Y} - 1.96 \times \frac{\widehat{\sigma}_Y}{\sqrt{n}}, \overline{Y} + 1.96 \times \frac{\widehat{\sigma}_Y}{\sqrt{n}}\right]$$

R Commands Covered

- Running a loop with the command `for`
- Fitting a chance model for the mean using `lm(y ~ 1)`
- Calculating a difference-in-means using `lm(y ~ x)`
- Using `summary` to return the output from `lm`
- Using `rowMeans` and `colMeans` to return the means of the rows or columns of a matrix
- Testing conditions using `if`
- Creating a pdf with `pdf`
- Running all the code in a file with `source`
- Loading all the objects in an R data file (`.Rdata`) with `load`

Before beginning this handout, do not forget to set your working directory!

Using a Loop to Create a Bootstrapped Confidence Interval

We will be using the command `lm` in this class with some frequency. `lm` stands for "linear model" and gives a way of estimating chance models. For example, let's assume we have some outcome, Y_i, for a simple random sample of observations, i from 1 to N. Next, assume we want to fit a model of the following form:

$$Y_i = \beta_0 + \epsilon_i$$

In this model, we are assuming that our observed value Y_i is equal to some expected value (β_0) plus some chance error (ϵ_i). Though you may have realized that β_0 is simply the expected value of Y_i, and so its best estimate is the mean, $\overline{Y}$, we are going to spend some time incorporating this into the framework implemented by the command `lm`.

Fitting a Chance Model to World-Wide GDP

In the data folder you will find the file `GDPNorAm.csv`. This file contains 184 observations, which consist of the country name, 2008 GDP (according to the World Bank), and an *indicator variable* that takes on a value of 1 if the country is in North America.

```
Data.gdp <- read.csv("data/GDP_NorAm_Partial.csv", header = TRUE)
head(Data.gdp)
summary(Data.gdp)
```

Next, note that GDP is in dollars. Let's make a new variable that turns it from dollars to billions of dollars. Then, let's add the logged value of GDP:

```
Data.gdp$gdp2 <- Data.gdp$gdp / 1e9 # 1e9 returns a 1 with 9 zeroes, i.e., one billion
Data.gdp$log.gdp2 <- log(Data.gdp$gdp2)
```

Now that we have the data, let's fit a linear model (what FPP calls a "chance model") to log GDP. The linear model will look like

$$log.gdp2 = \beta_0 + \epsilon_i$$

which we can implement in R using the following code:

```
lm1 <- lm(Data.gdp$log.gdp2 ~ 1)
summary(lm1)
```

Using this summary function, you can focus on the part that says `Coefficients` and (for now) ignoring the rest. The output is telling us that our estimate for β_0, the mean of Y_i, is 3.2384. We can check this with

```
lm1$coef
mean(Data.gdp$log.gdp2)
```

and we get the same answer. The next column gives the standard error. Recall that the standard error of the sample mean is $\widehat{\sigma}_Y/\sqrt{n}$, where $\widehat{\sigma}_Y$ is the sample standard deviation of Y. We can calculate the standard error using

```
n <- length(Data.gdp$log.gdp2)
se <- sd(Data.gdp$log.gdp2) / sqrt(n)
se
```

Creating a Confidence Interval Using the Normal Approximation

When we use the Normal Approximation to create a confidence interval, we combine two different observations:

1. If a variable follows a normal distribution, we expect it to show up within 1.96 standard errors of its mean approximately 95% of the time.
2. Due to the **Central Limit Theorem**, we know that the sample mean, $\overline{Y}$, is approximately normal, with mean $\mathbb{E}(Y)$ and variance σ^2_Y/n, which we sometimes write as

$$\overline{Y} \sim \mathcal{N}\left(\mathbb{E}(Y), \frac{\sigma^2_Y}{n}\right)$$

In the notation above, we are saying that $\overline{Y}$ is distributed ($\sim$) as a Normal random variable, with mean $\mathbb{E}(Y)$ and variance σ^2_Y/n.

When we use the Normal Approximation, we create a 95% confidence interval by taking the interval that ranges from 1.96 standard errors below the sample mean to 1.96 standard errors above the standard mean:

```
c(mean(Data.gdp$log.gdp2) - 1.96 * se, mean(Data.gdp$log.gdp2) + 1.96 * se)
```

We could create a 90% confidence interval

```
c(mean(Data.gdp$log.gdp2) - 1.64 * se, mean(Data.gdp$log.gdp2) + 1.64 * se)
```

Creating a Confidence Interval Using the Bootstrap

Next, we are going to calculate a confidence interval using the bootstrap. In order to implement the bootstrap, we need to use the command `sample` to produce a sample. The syntax for the command is

```
sample(vector, size, replace)
```

where the *vector* is what we are sampling from; the `size` option tells how elements observations to sample from *vector*; and `replace` tells whether we can resample the same element multiple times. The `sample` function is a means of simulating the box-models discussed in the book. To see precisely what the function is doing, look at the following loop:

```
for (i in 1:10) {
  boot.samp <- sample(1:5, size = 5, replace = TRUE)
  print(boot.samp)
}
```

Each of the rows gives the indices for a *bootstrapped sample*. We have more than 5 observations in our data, so we are going to run our bootstrap so that it samples from `1:n`. The code for calculating the bootstrapped sample mean is

```
boot.mean <- NULL
for (i.boot in 1:5000) {
  boot.samp <- sample(1:n, size = n, replace = TRUE)
  Data.boot <- Data.gdp[boot.samp, ]
  boot.mean[i.boot]  <- lm(Data.boot$log.gdp2 ~ 1)$coef
}
```

Now, we are going to use the function `quantile` to recover our 95% confidence interval. The syntax for the command is

```
quantile(vector, probs)
```

where *vector* is the vector of bootstrapped means. For the 95% confidence interval, we want to take the quantiles at the probabilities `c(0.025, 0.975)`, since 0.975 - 0.025 = 0.95 and the two numbers add up to 1. We get

```
quantile(boot.mean, probs = c( 0.025, 0.975))
```

We could also calculate a 90% bootstrapped confidence interval, using

```
quantile(boot.mean, probs = c(0.05, 0.95))
```

The bootstrapped confidence intervals are quite close to that from using the Normal Approximation. This approximation is quite accurate *most* of the time; in the problems, you will explore a situation where the Normal Approximation is not entirely accurate.

Adjusting Rows or Columns of a Data Frame with a Loop

For this problem, we will use the data from an online repository of US Congressional roll call votes. The roll call data for the 112th Session of the House of Representatives (2011-2012), `rollcalls_112.csv`, can be found in the data folder.

```
rollcalls <- read.csv("data/rollcalls_112.csv")
```

The variables in this dataset are:

- `cong`: the number of that Congress
- `id`: a unique identifier for each Congressmember
- `state`: the legislator's state (numeric)
- `ldist`: the legislator's district
- `lstate`: the legislator's state (as a word)
- `party`: the legislator's party (100: Democrat and 200: Republican)
- `V1`, `V2`, etc.: outcome for each vote, with each column a unique vote (0: no vote, 1: yes vote, NA: legislator did not vote)

First, we are going to clean up this data. To do so, we are going to create a new data frame that contains only the votes. This is `rollcalls` without its first 7 columns, and we can do this one of two different ways:

```
votes <- rollcalls[, -c(1:7)]
votes <- rollcalls[, 8:ncol(rollcalls)]
```

To reduce the time it takes to run the loops, we are going to look at only the first 200 votes that session:

```
votes <- votes[, 1:200]
```

Note that, since we are selecting columns, we want to enter the subset condition into the *second* argument of the data frame.

First, we want to reorient the 1's and 0's so that a 1 denotes "voted with the majority of the Democrats", and a 0 means "voted with the majority of the Republicans". That way, we can give some partisan valence to our measure. To do this we are going to loop through column by column. However, we are going to have to deal with the `NA` values in the data. To do this we will use the `na.rm = TRUE` option, which tells R to ignore `NA` values. The syntax of this command when used in combination with the `mean` command is

```
mean(vector, na.rm=TRUE)
```

Since votes are in columns, we are going to loop through columns (the second argument in the data frame) rather than rows (the first):

```
votes_recoded <- votes # Our new data frame
n.votes <- ncol(votes)
for (i.vote in 1:n.votes) {
  # Share of Democrats Voting Yea
  mean.dem <- mean(votes[rollcalls$party == 100, i.vote], na.rm = TRUE)
  # Share of Republicans Voting Yea
  mean.rep <- mean(votes[rollcalls$party == 200, i.vote], na.rm = TRUE)
  if (mean.dem < mean.rep) {
    votes_recoded[, i.vote] <- 1 - votes[, i.vote]
  }
}
```

There are several ways to do this loop, and some require a fewer lines than the code above. As you get more used to loops, you will find your coding growing more efficient naturally.

Note that we are using the `if` command above. The command uses the following syntax:

```
if (test) { function(x) }
```

The `if` command can be useful in programming, and is a more flexible version of `ifelse`. We are now ready to produce a measure of each legislator's ideology, which political scientists refer to as 'ideal points.' A simple estimate of ideology is the mean for each legislator, across rows. We can produce this using the command `rowMeans`, as below:

```
ideal.points <- rowMeans(votes_recoded, na.rm = TRUE)
plot(density(ideal.points[rollcalls$party == 200], cut = 0, na.rm = TRUE),
```

```
     col = "red", xlim = c(0,1),
     main = "Estimated Ideal Points",
     xlab = "Ideal Points")
lines(density(ideal.points[rollcalls$party == 100], cut = 0, na.rm = TRUE),
      col = "blue")
```

The `rowMeans` function takes a matrix and returns the mean of each row; the `colMeans` function does the same, but on columns. Note that in this Congress, we get complete separation between the Democrats and Republicans.

Nested Loops

In the last 15 years, political scientists have produced a vast literature on get out the vote (GOTV) effects. The question is simple: what can be done to increase turnout among the US population? The first modern study in this field was conducted by Alan Gerber and Don Green, of Yale, during a 1998 election (Gerber and Green 2000). The authors assigned residents of New Haven to different groups, whereby the residents received either a personal visit, phone calls, or mailings. Several individuals were also left untreated, to act as a control group.

The study was particularly influential—so much so that one of the original authors headed up the GOTV effort for Gov. Rick Perry of Texas, and one of his students/co-authors headed up the GOTV efforts for Pres. Obama out of Chicago. The work has gotten plenty of attention from the popular press, as well. According to an interview of a campaign operative on the Washington Post:

> Two political scientists at Yale, Donald Green and Alan Gerber, went out and did a field experiment, which was a big deal at the time because political science lagged behind other social sciences in using field experiments to measure cause and effect in elections [...] The campaigns went out and did a bunch more of these comparative effectiveness studies, as opposed to mass media, where it's really hard to isolate voters and implement controls. When you're measuring turnout and registration rates, it's very easy to select some people to get your mail. In that case, the dependent variable is whether they voted or registered, which is an easy thing to track. In the last few years it's moved a lot to the behavioral psychology-informed bent, trying to demonstrate things that have been demonstrated in lab experiments about how to change motivations around voting [...] We have a whole sort of body of research about the contact and the quality of contact that we didn't 15 years ago.

(link) from 11/05/2012.

We are going to analyze the data from Gerber and Green's first study. In the data folder you will find the file `apsr_corrected_latest.txt`, which has the relevant data. The variables we will be using are:

Name	Description
`voted98`	1 if they voted in the 1998 election, 0 otherwise (dependent variable)
`vote96.0`	1 if they abstained from voting in 1996, 0 otherwise
`persons`	Number of people in the household
`mailgrp`	1 if they were sent mail encouraging them to vote; 0 otherwise
`persngrp`	1 if they were assigned to receive a personal visit encouraging them to vote; 0 otherwise
`phonegrp`	1 if they were called and encouraged them to vote; 0 otherwise
`ward`	Voting ward of household (2, 3, ..., 30)
`appeal`	Type of appeal (1, 2, 3)
`mail`	Number of mailings sent (0,1,2,3)
`age`	age of respondent

We are going to come back to this data a few times. First, let's read in the data and look at the baseline voting rate, for those who received no treatment:

```
GG <- read.table("data/apsr_corrected_latest.txt")
head(GG)
summary(GG)

notreat.sub <- (GG$mailgrp == 0) &
               (GG$persngrp == 0) &
               (GG$phonegrp == 0)
control.mean <- mean(GG$voted98[notreat.sub == 1])
control.mean
```

Now, mailing (may?) have an effect that is close to zero: approximately 1.1 percentage points:

```
mail.mean <- mean(GG$voted98[GG$mailgrp == 1])
mail.mean
mail.mean - control.mean
```

A candidate, though, may want to know if a particular combination of appeals and mailings may be more effective. We are going to assess this in the next loop. This loop is a double loop, which is sometimes called a nested loop. The idea is that we loop through two different counters: one for the number of mailings (1, 2, 3) and another for the appeal (1, 2, 3).

The code follows:

```
effects.out <- NULL
for (i.num in 1:3) {
  for (i.appeal in 1:3) {
    mail.sub <- (GG$mailgrp == 1) &
                (GG$appeal == i.appeal) &
                (GG$mailings == i.num)
    outcome.sub <- mean(GG$voted98[mail.sub])
    output <- c(i.num, i.appeal, outcome.sub - control.mean)
    effects.out <- rbind(effects.out, output)
  }
}
colnames(effects.out) <- c("number", "appeal", "effect")
effects.out
```

It appears that receiving 3 mailings of appeal 3 may be the most effective.
If you are wary of this result, though, you are probably correct. First, what we are doing is a type of "data-fishing," in that we cycle through all sub-groups until we find a strong result. Given enough time in a dataset, we can find *something*, though it may be noise. Second, we will use this as an example later to discuss statistical significance, the idea of whether we can differentiate a given effect from 0.

Precept Questions

Please note that *for this week only*, these problems will not be handed in. Please hand in the code from the examples, and we will do the following questions in precept.

Question 1

Using the GDP data calculate the 95% Confidence Interval for `gdp2` using the normal approximation and the bootstrap.

Why do they differ so much? (Hint: Look at the density plots of `gdp2` versus `log.gdp2`

Question 2

Create a new chance model, `lm(Data.gdp$gdp2 ~ Data.gdp$north.america)`

What does each coefficient represent?

Question 3

In the data folder you will find the file `rollcalls_96to110.csv`, which contains roll call votes for the House for the 96th, 98th, 100th, 102nd, 104th, 106th, 108th, and 110th Congresses. Create a `for` loop that does the same analysis as the roll call example above but for multiple congresses. (This will be a nested loop with the outer loop iterating through each Congress and the inner loop iterating through each vote.) The loop should produce a 4 x 2 figure of density plots.

Question 4

In the Gerber and Green data what is the estimated effect of having been contacted by phone?

Question 5

Does the effect vary across whether the respondent voted in the past and the type of appeal they received?

References

Gerber, Alan S., and Donald P. Green. 2000. "The Effects of Canvassing, Telephone Calls, and Direct Mail on Voter Turnout: A Field Experiment." *American Political Science Review* 94 (03): 653–63. https://doi.org/10.2307/2585837.