Lab 10: Analyzing the 2015 Major League Baseball (MLB) Season
Math 32, Fall 2019, Prof. Bhat

The starting point for this lab will be the `mlb2015.RData` data set that has been uploaded to the Files/Lab10/ section of the CatCourses web site. Once you download this data set and set your R working directory to the right place, you can load the file and verify you have the right data set by running:

```
> load('mlb2015.RData')
> class(mlb2015)

[1] "data.frame"

> dim(mlb2015)

[1] 30 63
```

Let us explain a few facts about this data set and about baseball!

- This is real data from the 2015 Major League Baseball (MLB) regular season.
- There are 30 MLB teams. Each team corresponds to a row of the data set.
- The first column of the data set gives you the team name. All other columns are numeric.
- Each row of the data set contains aggregate statistics on how the team performed in a variety of categories, both offensive (batting) and defensive (pitching) categories.
- If you know nothing else about baseball, know that it is a game played between two teams. When the game starts, the score is 0 to 0. There are no "points" in baseball—the unit of scoring is a "run." The team with the most runs at the end of the game is the winner.
- There is no clock in baseball. Instead, the teams take turns on offense and defense. A complete cycle where both teams have taken one turn each on offense and defense is called an "inning." A typical baseball game consists of nine innings. If the score is tied after nine innings, the teams continue to play one inning at a time until the score is not tied.
- For further information on the meaning of each of the columns, check out the following web page: `http://www.baseball-reference.com/leagues/MLB/2015.shtml`. I pulled the data from the tables labeled "Team & League Standard Batting" and "Team & League Standard Pitching." If you let your mouse hover over one of the column headings (for the tables on the web site), it will show you the meaning of the column name. For instance, if I hover over "RBI" using the mouse, then I learn that "RBI" means "Runs Batted In."

Now, in the regular season, each team is supposed to play 162 games. Occasionally, games are cancelled due to bad weather. To see how many games each team played, try this:

```
> mlb2015$G

 [1] 162 162 162 162 162 162 162 161 162 161 162 162 162 162 162 162 162 162
[18] 162 162 162 162 162 162 162 162 162 162 162 162 162
```

As you can see, some teams played only 161 games due to weather. To determine each team's winning percentage, we can do the following:

```
> mlb2015$W / mlb2015$G

 [1] 0.4876543 0.4135802 0.5000000 0.4814815 0.5987654 0.4691358
 [7] 0.3950617 0.5031056 0.4197531 0.4596273 0.5308642 0.5864198
[13] 0.5246914 0.5679012 0.4382716 0.4197531 0.5123457 0.5555556
[19] 0.5370370 0.4197531 0.3888889 0.6049383 0.4567901 0.4691358
[25] 0.5185185 0.6172840 0.4938272 0.5432099 0.5740741 0.5123457
```

Now here's the thing: this statistic is already present in the data set:

```
> mlb2015$W.Lpct

 [1] 0.488 0.414 0.500 0.481 0.599 0.469 0.395 0.503 0.420 0.460 0.531
[12] 0.586 0.525 0.568 0.438 0.420 0.512 0.556 0.537 0.420 0.389 0.605
[23] 0.457 0.469 0.519 0.617 0.494 0.543 0.574 0.512
```

Suppose that winning percentage (`W.Lpct`) is our response variable, i.e., the thing we are trying to predict. Then there are two things we should definitely avoid doing:

1. First, we should avoid using `W` or `L` as predictor variables. That is because there is an obvious relationship between winning percentage and the total number of wins! The exact relationship is:

   $$\text{winning percentage} = \text{number of games won}/\text{number of games played}$$

   Because the number of games played is almost always 162, we can approximate by saying

   $$\text{winning percentage} \approx \text{number of games won}/162$$

   This is a linear model! Does the data show this? Let's check—by the way, please notice that in this entire set of notes, we use `<-` instead of `=`. If you want to use the equals sign, that is perfectly fine!

   ```
   > lm.test = lm(W.Lpct ~ 0+W,data=mlb2015)
   > coefficients(lm.test)

             W
   0.006175558
   ```

```
> print(1/162)

[1] 0.00617284
```

The `0+` syntax tells R to not include an intercept. Note that the resulting model has a slope that is very close to 1/162. Please make sure you understand why that is the case. Once you do, you can check the MSE (mean squared error) of the model as follows:

```
> mean(lm.test$residuals^2)

[1] 6.631962e-07
```

The error is nearly 0. *The fundamental point here is that we already knew that the winning percentage is going to be extremely close to the number of wins divided by* 162*, and so we have learned nothing from the statistical analysis.* If we want the model to teach us something we do not already know, we should not use the `W` variable to try to predict the `W.Lpct`.

Similarly, we should not use the `L` variable, because the number of losses is simply the number of games played minus the number of wins! And so, we could equally well have written:

winning percentage $\approx$ (162$-$number of games lost)/162 = 1$-$number of games lost/162

Just to see that linear regression is capable of picking up this relationship, try this:

```
> lm.test = lm(W.Lpct ~ L,data=mlb2015)
> coefficients(lm.test)

 (Intercept)           L
 0.999815213 -0.006172687
```

As we expected, we get an intercept that is very close to 1 and a slope that is very close to $-1/162$. Again, make sure you understand why this happened.

2. There are more columns than rows in this data set! Even if we exclude the `W` variable, the `L` variable, the response variable, and the name of the team, we still have

```
> ncol(mlb2015) - 4

[1] 59
```

columns in our data set, which is clearly bigger than 30. Hence one thing we should really avoid doing is using 30 or more different variables/columns in our model. Let's see what happens when we use all 59 potential variables in the model.

```
> which(colnames(mlb2015)=="W")
```

```
[1] 33
```

```
> which(colnames(mlb2015)=="L")
```

```
[1] 34
```

```
> which(colnames(mlb2015)=="Tm")
```

```
[1] 1
```

```
> # Now remove the columns associated with the W, L, and Tm variables
> # Store the data set with these columns removed in "mlb"
> mlb = mlb2015[,-c(33,34,1)]
> # Fit a linear model
> lm.overfit = lm(W.Lpct ~ .,data=mlb)
> # Determine MSE
> mean(lm.overfit$residuals^2)
```

```
[1] 0
```

```
> # Coefficients
> coefficients(lm.overfit)
```

(Intercept)	NumBat	BatAge	R.G	G
8.146503e+01	-1.303979e-03	-9.090721e-03	-6.148329e+00	-1.533485e-01
PA	AB	R	H	X2B
-1.868490e-03	-8.008021e-03	3.919036e-02	2.492463e-02	1.389385e-02
X3B	HR	RBI	SB	CS
3.102318e-02	4.413074e-02	-4.742331e-04	-1.700238e-03	3.291890e-03
BB	SO	BA	OBP	SLG
8.639550e-03	-5.055572e-04	-5.431140e+00	1.419436e+01	3.246206e+00
OPS	OPS.	TB	GDP	HBP
-8.054532e+01	2.982785e-03	NA	-1.818034e-04	1.099029e-02
SH	SF	IBB	LOB	NumP
3.016125e-03	-6.371608e-04	2.732495e-03	1.411038e-03	-7.210777e-03
PAge	RAperG	ERA	GS	GF
6.188181e-03	NA	NA	NA	NA
CG	tSho	cSho	SV	IP
NA	NA	NA	NA	NA
H.1	R.1	ER	HR.1	BB.1
NA	NA	NA	NA	NA
IBB.1	SO.1	HBP.1	BK	WP
NA	NA	NA	NA	NA
BF	ERA.	FIP	WHIP	H9
NA	NA	NA	NA	NA
HR9	BB9	SO9	SOperW	LOB.1
NA	NA	NA	NA	NA

Though we have produced a model with zero mean squared error, the model itself is clearly ridiculous. A large number of coefficients are "NA."

Guarding Against Overfitting. If you are in the business of creating regression models for the purposes of prediction, a crucial statistical technique you should learn is cross-validation. Cross-validation is a method to guard against overfitting. In this set of notes, we will explain LOOCV: "leave one out cross-validation."

In the context of our current data set, LOOCV works like this:

- Pick the j-th team in the data set, which corresponds to the j-th row.
- Fit a model, with winning percentage as the response variable, to all data *except* for the data on the j-th row.
- Use the model to predict the winning percentage for the team on the j-th row.
- Calculate the squared error between the predicted and true winning percentages for the team on the j-th row. Call this the j-th "individual error."

We repeat this procedure for each j from 1 to 30. This will give us 30 values of the "individual error." We then take the *average* of these 30 individual values to obtain an overall LOOCV error.

Let us explain this with R code. The code below assumes you have defined `mlb` using the code above.

```
> allmodels = list(NULL)
> n = nrow(mlb)
> indiverrors = numeric(length=n)
> for (i in c(1:30))
+ {
+     allmodels[[i]] = lm(W.Lpct ~ HR, data=mlb[-i,])
+     truewinpct = mlb[i,]$W.Lpct
+     predwinpct = predict(allmodels[[i]],newdata=mlb[i,])
+     indiverrors[i] = (truewinpct-predwinpct)^2
+ }
> mean(indiverrors)

[1] 0.004084733
```

The basic ideas behind leave one out cross-validation (LOOCV) are as follows:

1. If the model is correct, i.e., if it really does give a correct relationship between winning percentage and home runs (or whatever predictors you choose), then it should do a reasonably good job of predicting the winning percentage for all teams, not just the particular teams in our data set.
2. Instead of singling out some teams as "teams we use to build the model" and other teams as "teams we use to test whether the model is right," LOOCV takes a democratic approach. Each team is used once (and only once) as the team used to test whether the model is right.

Your Goal. Let's try to predict a team's winning percentage. As we discussed above, the allowed predictors do *not* include the number of wins or the number of losses. We also do not want to use the team's name as a predictor. Allowed predictors might include nonlinear combinations of existing predictors—within reason. For instance, if you want to use home runs divided by earned run average, go for it. (I just made that up—it's probably not the best predictor, but I mention it only to illustrate the creative freedom that is being gifted to you here.)

So, here are the rules:

- For now, you must use "lm" to build your models.
- You must assess your model's accuracy using LOOCV. Don't overfit!
- Overall, do not use more than 25 predictors. Don't overfit!
- When you find a model you really like, examine the coefficients and try to make sense of them. Figure out if the model makes sense. For instance, if a team allows more runs per game, its winning percentage should decrease, no? Try to see if your model tells a good story about what makes a winning (or losing) baseball team.

Other than that, your goal is to try to devise a model that does the best possible job of predicting a team's winning percentage. Go for it! **The lab is due by Thursday, December 5th at 11:59pm.**