Learning Goals

1. Counting Statistics vs Rate Statistics
2. Evaluating Statistics
3. Measuring individual contributions on a team.
4. Four factors in basketball.
5. Run a multiple regression in R (find linear weights).
6. Interpreting the output of a multiple regression.
7. Testing the model.

Topic 3: A few Statistics and Multiple Regression

Sports Stats Sports statistics are used in many ways. Some are used to evaluate or rank players and teams, by coaches, recruiters, fantasy sports players, mangers and bettors. Some statistics suit our needs better than others. Clearly if we wish to predict winning percentage, a statistic that is well correlated with winning percentage is better than one that is not. We may not wish to rely entirely on a single statistic for prediction, it makes more sense to include a variety of statistics that measure various aspects of the strengths and weaknesses of a team.

Counting Statistics vs Rate Statistics Many of the statistics on sports websites are counting statistics which count the number of points, Field Goals Scored, Turnovers, etc... Although it is essential to count these important statistics, they miss some important subtleties in measuring performance, especially the efficiency of a team or player. For example, a basketball player may have scored very few three pointers in a game or a season compared to a teammate, but the number of three pointers scored for each player does not take into account the opportunities they had to score. A better statistic in measuring performance here is 3 point percentage (3pointers made/3 pointers attempted). The team statistic, Effective Field Goal Percentage (eFG.P) discussed below is a more sophisticated example of a rate statistic which measures shooting effectiveness and efficiency for a team. It can also be calculated as a measure of efficiency for a player. It is calculated as

$$eFG.P = \frac{FGM + (0.5)3PtFGM}{FGA}$$

and takes into account that Three point Field Goals Made (3PtFGM) are worth 3 pts whereas all others are worth 2 points.

Evaluating statistics A good statistic should be reliable. A statistic that measures a player's/team's ability should correlate well from season to season, give or take a few anomalies due to injury and random fluctuation about average performance. One would expect that players who are relatively strong in a particular skill in one season will be relatively strong in the next season. If we are using a statistic for prediction, then it should give good predictions on the existing data and also should give good predictions on future data. When making models or creating statistics, one often makes the model using half of the data available and then tests the model on the other half.

Team Stats vs. Player Stats When a player is part of a team, it is difficult to separate and measure their effectiveness from that of their teammates. For example, in soccer, a great supporting player who continually creates opportunities for his/her teammates may never actually score a goal. One way to measure a player's contribution is to identify play sequences and attribute credit/fault for the success or failure of the sequence. There can be a lot of subjectivity involved in attributing credit/fault in these

statistics and a deep knowledge of the game is required in calculating them. Another popular method is with **plus/minus statistics**, which divides the blame/credit among the players. In general one looks at net team points (points for - points against) while the player is on the team. If a seven player team scores 10 points against a seven-player opposing team who scores 8 points, then every player on the team gets a +2 while every player on the opposing team gets a -2. Clearly a player on a good team will have a high +/− rating no matter how bad they are and vice-versa. One can fix this obvious flaw to some extent by calculating the difference in the teams net score when the player is in the game vs when the player is not in the game. For example Melinda gets a +4 rating if her team is outscored by 10 points when she is out of the game and outscored by 6 points when she is in the game. As a result Melinda gets a + rating even though her team lost. On the other hand if the coach always puts Melinda and Melissa in the game at the same time, and Melissa does all the work, Melinda gets the points for nothing. (As with most statistics/methods of rating players, this is not perfect).

Basketball's Four Factors

The last four variables in the data frame `BBdata` are all "rate" statistics. They give measures of the four important factors shooting, turnovers, rebounding and free throws respectively. They can be calculated for each game for both teams, or they can be calculated from cumulative statistics for the season for the teams. They were introduced to basketball analysis by Dean Oliver and there are slight variations in their calculations. More background and details are given in [1]. They are defined as:

Effective Field Goal Percentage (`eFG.P`) ; this statistic adjusts for the fact that a 3-point field goal is worth one more point than a 2-point field goal. The formula is given by

```
eFG% = eFG.P = (FG+0.5*3P)/FGA
```

Turnover Percentage (`TOV.P`) ; an estimate of turnovers per 100 plays. The formula is given by

```
TOV% = TOV.P = TOV/Poss (Poss = FGA-OR+TO+0.44*FTA)
```

Offensive Rebound Percentage (`ORB.P`) ; an estimate of the percentage of available offensive rebounds grabbed by a player on the team. The formula is given by

```
ORB%= ORB.P = ORB/ORB+DRB
```

Free Throws Per Field Goal Attempt (`FTPFGA`) .The formula is given by

```
FTR= FTM/FGA
```

Multiple Regression

Many ratings for individual athletes and teams combine multiple statistics into one rating. For example, if we were to rate basketball teams, we would like to take into account their offensive strengths and weaknesses as well as their defensive strengths and weaknesses. Lets suppose we wish to make a rating for the basketball teams in our data frame `BBdata`, the purpose of which is to predict performance in the remaining part of basketball season, and we want the rating to be a formula based on the four factors from the previous discussion. There are many possible types of formulas. If our four factors were labelled x_1, x_2, x_3 and x_4, and R denotes our rating, then we could make a **linear formula** of the type $R = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5$, where a_1, a_2, a_3 and a_4 are numbers called coefficients or **weights**. We could also allow powers of the variables or mixed multiplications(interactions) as in a formula of the type $a_1x_1^2 + a_2x_1x_2 + a_3x_3^2 + a_4x_1 + \ldots$. We can also allow formulas based on exponentials and logarithms etc... We will make a linear formula.

Multiple Regression When we wish to predict the values of a variable Y (called the dependent variable) from an independent set of variables $x_1, x_2, x_3, \ldots, x_n$, using multiple regression, we find weights(or coefficients), $a_1, a_2, a_3, \ldots, a_n$, and a constant term, c, where

$$Y = a_1x_1 + a_2x_2 + \cdots + a_nx_n + c$$

is the best fit model from our data of that form. In other words it the the formula of that type that minimizes the sum of the squared errors in our data when we fit the model. (The linear models with a single independent variable in the previous lecture is a special example of this with one independent variable). We can create such a model very easily in R with the command `fit <- lm(Y` $x_1 + x_2 + x_3 + \cdots + x_n$`, data=mydata)` if our variables are stored in the file `mydta`.

Warning: There are several underlying assumptions about the variables that should be checked in order to use this method to create a valid model. Essentially they boil down to requiring that the errors in the model (actual values in the data minus predicted values) should have similar bell shaped distributions and that the independent variables should not be highly correlated. This second condition essentially means that you do not replicate the information supplied in your model. For example If I were including 2 pt Field Goals Made (2PFGM) and 3 pt FGM (3PFGM) in my formula to explain point differential, I would not want the global statistic FGM = 2PFGM + 3PFGM because I already have that information in my formula. It is not a bad idea to check the correlations between the statistics you are using in your model, before running your model. You can also run tests on a model to see if correlation between factors has caused a problem or if errors are not properly distributed.

Example Suppose we want to predict Overall win-loss percentage, `OWL.P` for the rest of the season based on the four factors Effective Field Goal Percentage, `eFG.P`, Turnover Percentage `TOV.P`, Offensive Rebound Percentage `ORB.P` and Free Throws Per Field Goal Attempt `FTPFGA`. We could create a multiple regression model based on the data we have to date from the season of the form

$$\texttt{OWL.P} = a_1\texttt{eFG.P} + a_2\texttt{TOV.P} + a_3\texttt{ORB.P} + a_4\texttt{FTPFGA} + c$$

using R as follows:

```
BBdata <- read.csv(file="BBStats.csv", header=TRUE, sep=",")
model<-lm(OWL.P ~ eFG.P+TOV.P+ORB.P+FTPFGA, data = BBdata)
model
```

```
##
## Call:
## lm(formula = OWL.P ~ eFG.P + TOV.P + ORB.P + FTPFGA, data = BBdata)
##
## Coefficients:
## (Intercept)        eFG.P        TOV.P        ORB.P       FTPFGA
##    -0.87829      2.71452     -0.03768      0.01486      0.89632
```

This tells us that the model of this type that fits the data best (least squares solution) is of the form

$$\texttt{OWL.P} = 2.71452\ \texttt{eFG.P} - 0.03768\ \texttt{TOV.P} + 0.01486\ \texttt{ORB.P} + 0.89632\ \texttt{FTPFGA} - 0.87829$$

```
summary(model)
```

```
##
## Call:
## lm(formula = OWL.P ~ eFG.P + TOV.P + ORB.P + FTPFGA, data = BBdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31054 -0.08005  0.00477  0.07389  0.34321
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.878286   0.140712  -6.242 1.26e-09 ***
## eFG.P        2.714518   0.198872  13.650  < 2e-16 ***
## TOV.P       -0.037677   0.003557 -10.592  < 2e-16 ***
## ORB.P        0.014856   0.001568   9.472  < 2e-16 ***
## FTPFGA       0.896318   0.160338   5.590 4.59e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1218 on 348 degrees of freedom
## Multiple R-squared:  0.6251, Adjusted R-squared:  0.6208
## F-statistic: 145.1 on 4 and 348 DF,  p-value: < 2.2e-16
```

We see in the summary that the p-values for all factors are significant and we include all in our model. (If a factor is not significant, you should drop it from your model and rerun the model. You will get a new set of coefficients in the new model.) We also see in the summary that the value of R-squared is 0.6208, which says that about 62% of the variation in wins and losses is explained by these 4 statistics.

The model tells us that an increase in `eFG.P` by 1 unit, increases the overall win loss percentage `OWL.P` for the team by about 2.7 times the increase in `eFG.P` (in the presence of the other factors in the model). As we would expect and increase in `TOV.P` by 1 unit for a team tends to decrease the `OWL.P` by about 0.04 times that unit (in the presence of the other factors in the model) etc... If we include different factors in our model, these variables may have a different effect on the win-loss percentage as we will see below.

Improving the model Clearly the model does not account for all of the variation in the win-loss percentage, perhaps we can improve it by taking into account the strength of schedule and average point differential in the `SRS` (Simple Rating System) statistic which is a rating that takes into account average point differential and strength of schedule (Its formula is not given in the data). The rating is denominated in points above/below average, where zero is average. Non-Division I games are excluded from the ratings. Here are the results of a linear model that includes `SRS` and the four factors:

```r
BBdata <- read.csv(file="BBStats.csv", header=TRUE, sep=",")
model1<-lm(OWL.P ~ eFG.P+TOV.P+ORB.P+FTPFGA+SRS, data = BBdata)
model1
```

```
##
## Call:
## lm(formula = OWL.P ~ eFG.P + TOV.P + ORB.P + FTPFGA + SRS, data = BBdata)
##
## Coefficients:
## (Intercept)        eFG.P        TOV.P        ORB.P       FTPFGA
##   -0.089880     1.322149    -0.019415     0.006559     0.344008
##         SRS
##    0.010213
```

```r
summary(model1)
```

```
##
## Call:
## lm(formula = OWL.P ~ eFG.P + TOV.P + ORB.P + FTPFGA + SRS, data = BBdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32145 -0.07255 -0.00041  0.07047  0.29745
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0898800  0.1295380  -0.694   0.4882
## eFG.P        1.3221490  0.1936899   6.826 3.91e-11 ***
## TOV.P       -0.0194145  0.0032185  -6.032 4.15e-09 ***
## ORB.P        0.0065595  0.0014272   4.596 6.03e-06 ***
## FTPFGA       0.3440082  0.1375084   2.502   0.0128 *
## SRS          0.0102134  0.0007734  13.205  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09953 on 347 degrees of freedom
## Multiple R-squared:  0.7505, Adjusted R-squared:  0.7469
## F-statistic: 208.8 on 5 and 347 DF,  p-value: < 2.2e-16
```

We see that the coefficients have changed, but all factors are still significant in the model. we also see that the R-squared statistic is higher than before, showing that the new model explains roughly 75% of the variation in the overall win-loss percentage. Our new (rounded) formula is:

$$\texttt{OWL.P} = 1.3221\ \texttt{eFG.P} - 0.0194\ \texttt{TOV.P} + 0.0066\ \texttt{ORB.P} + 0.3440\ \texttt{FTPFGA} + 0.0102\ \texttt{SRS} - 0.0899$$

The coefficient of `eFG.P` has dropped in our model and if we check, we will see that our new variable and `eFG.P` are correlated: `cor(BBdata$SRS,BBdata$eFG.P) = 0.5643039`. Although they are not

entirely independent, we should probably include both in our model since the new model explains more of the variation. A more refined model would allow for the interaction between these variables.

If you are worried that your independent variables are too highly correlated, R has a test you can run to check for this. The test gives you a measure of the variance inflation factor (VIF) for each variable. You must install and open the package "car" before running the test. The VIFs of the linear regression indicate the degree that the variances in the regression estimates are increased due to multicollinearity. VIF values higher than 10 indicate that multicollinearity is a problem. You should remove variables with VIF values higher than 10 from your model. For our model, we get

```{r}
#install.packages("car") you need to download this
package before the vif test
library(car)
vif(model1)
```

```
   eFG.P    TOV.P    ORB.P   FTPFGA      SRS
1.584267 1.347983 1.261614 1.136588 2.358864
```

We see that we do not need to throw anything out of our model.
You can also test for normality with the Shapiro-Wilks test. Here a p-value less that 0.05 indicates a problem with normality of residuals. In our case there is no problem.

```{r}
shapiro.test(residuals(model1))
```

```
        Shapiro-Wilk normality test

data:  residuals(model1)
W = 0.99762, p-value = 0.8972
```

Full Season Stats If we are going to use the above model for prediction of the win-loss percentage, it would be better to use statistics from a full season to make the model for prediction. this would take into account the difference in how teams play early in the season vs. later in the season. If we are trying to predict outcomes for games in the March Madness tournament, it would probably be much more accurate if we could make a model where we look at every game in a season and create a linear model for the point differential based on the difference between the current seasonal values of our four factors for both teams. We do not have that data available, but you can find in-game statistics for the four factors along with the point differential in the game-logs for each team on this website https://www.sports-reference.com/cbb/.

Below we show the results of our model for the (season ending in) 2017 statistics, stored in the file `BBStats17.csv`, with name adjustments for use in R as in the file `BBStats.csv`.

Testing the accuracy of the model It is also customary to create a model on one half of the data and then test how well it predicts values of the dependent variable to the other half. Alternatively, we can make the model on one season and test how well it predicts results for the next season. Let's

re-create our model on the data for all of (season ending in) 2017 and then test the model on the data for (season ending in) 2018. I've downloaded the full stats for these years and stored them in the files `BBStats17.csv` and `BBStats18.csv`. I have also changed the names of the variables as in the file `BBStats.csv`, so that we have legitimate variable names in R when we import. We first construct our model on the 2017 data:

```
BBdata17 <- read.csv(file="BBStats17.csv", header=TRUE, sep=",")
model2<-lm(OWL.P ~ eFG.P+TOV.P+ORB.P+FTPFGA+SRS, data = BBdata17)
model2
```

```
##
## Call:
## lm(formula = OWL.P ~ eFG.P + TOV.P + ORB.P + FTPFGA + SRS, data = BBdata17)
##
## Coefficients:
## (Intercept)        eFG.P        TOV.P        ORB.P       FTPFGA
##   -0.751948     2.206535    -0.016432     0.009160     0.575757
##         SRS
##    0.006616
```

```
summary(model2)
```

```
##
## Call:
## lm(formula = OWL.P ~ eFG.P + TOV.P + ORB.P + FTPFGA + SRS, data = BBdata17)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.262782 -0.058405 -0.001024  0.065533  0.217666
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.7519484  0.1312839  -5.728 2.22e-08 ***
## eFG.P        2.2065354  0.2012736  10.963  < 2e-16 ***
## TOV.P       -0.0164323  0.0035153  -4.674 4.23e-06 ***
## ORB.P        0.0091604  0.0013382   6.845 3.50e-11 ***
## FTPFGA       0.5757569  0.1453690   3.961 9.08e-05 ***
## SRS          0.0066159  0.0006581  10.053  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09256 on 345 degrees of freedom
## Multiple R-squared:  0.7127, Adjusted R-squared:  0.7085
## F-statistic: 171.1 on 5 and 345 DF,  p-value: < 2.2e-16
```

Our (rounded) model for predicting the values of `OWL.P` made using all of the 2017 season data is

$$\texttt{OWL.P} = 2.2065\ \texttt{eFG.P} - 0.0164\ \texttt{TOV.P} + 0.0092\ \texttt{ORB.P} + 0.5758\ \texttt{FTPFGA} + 0.0066\ \texttt{SRS} - 0.7519$$

To test our model, we import the the test data from the file `BBStats18.csv` and store it in a data frame `BBData18`. We create a new variable in the file called `pred` which give the predictions for the overall win-loss percentage using the formula we just made and the values of the independent variables in the data frame `BBData18`, with the command `predict(model2, BBdata18)`. We see that the values of `OWL.P` predicted by our formula developed on the 2017 data are not exactly the same as the actual

values of OWL.P in 2018 (as we would expect).

```
BBdata18 <- read.csv(file="BBStats18.csv", header=TRUE, sep=",")
BBdata18$pred <- predict(model2, BBdata18)
head(BBdata18)
```

```
##   Rk             School OG OW OL OWL.P    SRS   SOS CW CL HW HL AW AL
## 1  1  Abilene Christian 32 16 16 0.500  -9.14 -6.82  8 10  9  6  6  9
## 2  2          Air Force 31 12 19 0.387  -4.31  1.72  6 12  9  7  3 10
## 3  3              Akron 32 14 18 0.438  -6.82 -1.92  6 12 12  4  2 10
## 4  4        Alabama A&M 31  3 28 0.097 -23.97 -8.04  3 15  2  9  1 18
## 5  5 Alabama-Birmingham 33 20 13 0.606   4.90 -0.65 10  8 14  3  5  6
## 6  6      Alabama State 31  8 23 0.258 -18.55 -8.62  8 10  4  8  4 13
##   Tm.Pt Opp.Pt Pace  ORtg   FTr Th.Par  TS.P TRB.P AST.P STL.P BLK.P eFG.P
## 1  2359   2279 71.6 102.2 0.309  0.350 0.549  49.7  55.4  11.3  11.6 0.521
## 2  2124   2244 67.7 100.8 0.318  0.431 0.527  48.7  60.7   9.5   8.1 0.490
## 3  2296   2411 69.1 102.6 0.319  0.467 0.547  49.2  52.7   8.4   7.5 0.518
## 4  1873   2367 68.3  88.1 0.314  0.354 0.480  48.2  50.5   5.8   3.9 0.450
```



```
## 5  2536   2303 69.5 109.8 0.291  0.334 0.575  54.8  59.3   7.7  11.6 0.545
## 6  2129   2437 70.5  97.0 0.352  0.347 0.496  47.9  41.7   7.7  11.2 0.465
##   TOV.P ORB.P FTPFGA      pred
## 1  17.7  27.8  0.217 0.4259334
## 2  16.3  27.5  0.233 0.4189548
## 3  17.3  27.1  0.222 0.4377021
## 4  20.9  29.4  0.203 0.1251681
## 5  16.5  31.1  0.218 0.6223013
## 6  16.9  31.9  0.227 0.2965726
```

If our model is a good model for prediction, we would expect a high correlation between the predicted values of win-loss percentage stored in BBdata18$pred (calculated using the formula) and the actual values of the statistic OWL.P. We see that the correlation is actually quite high and thus the predictive value of our model is quite high.

```
cor(BBdata18$pred,BBdata18$OWL.P)
```

```
## [1] 0.8494952
```

You might ask if our formula gives us better predictions of the winning percentage for 2018, `BBData18$OWL.P`, than the actual winning percentage for each team in 2017 `BBData17$OWL.P`. To look at the correlation between these two variables I merged the files by team name (in case there were some non-matching teams in the files causing the order not to match exactly). This created two statistics corresponding to `BBData17$OWL.P` and `BBData18$OWL.P` resp. in the resulting file called `new`, namely `new$OWL.P.x` and `new$OWL.P.y`, I then checked the correlation between these statistics. As you can see the results of applying our model to the 2018 data is more highly correlated to the win-loss percentage in 2018 that the 2017 values of `OWL.P`.

```{r}
new<-merge.data.frame(BBdata17,BBdata18,by="School")
cor(new$OWL.P.x,new$OWL.P.y)
```

```
[1] 0.5432826
```

R commands

1. `fit <- lm(y  x1 + x2 + x3, data=mydata)` : creates best fit linear model for predicting values of the dependent variable `y` from values of the independent variables $x_1, x_2, \ldots, x_n$, using the data set `mydata` to make the model.
2. `testdata$pred <- predict(fit, testdata)` applies the model created in `fit` to the relevant variables in the file `testdata` to created a new variable called `pred`, which gives the values predicted by the model.
3. Formulas in the boxed paragraphs are extra to help you check the validity of your model.

References

[1] Kubatko, J., Oliver, D., Pelton K., and Rosenbaum, D. *A Starting Point For Analyzing Basketball Statistics*. Journal of Quantitative Analysis in Sport, **Vol 3, Issue 3, 2007, Article 1.**

[2] Shea, Stephen M., and Baker, Christopher E. *Basketball Analytics*. Advanced Metrics, LLC, Lake St. Louis, MO, 2013.

[3] Winston, Wayne L. *Mathletics* Princeton University Press.