Lecture 14

Paul Holaway, Abhi Thanvi

July 14th, 2022

Lecture 13 Review

Before we move on to the new material, we will do a quick review of Lecture 13 content. Last time we learned about hypothesis testing and how it is so useful. If you go into *any* type of science, you are almost guaranteed to be using hypothesis testing at some point. Let's do a quick review example.

Example 1; Hypothesis Testing Review

Suppose you are taking a sample of people where you ask two things...

1. Do they have siblings? (0 = No, 1 = Yes)
2. How many siblings do they have?

The data is below.

```
#DO NOT DELETE THIS CODE!!
set.seed(143572)
sibling = sample(0:1, size = 15, replace = TRUE)
number = rep(0,15)
for(i in 1:15){
  if(sibling[i] != 0){
    number[i] = sample(1:5, size = 1, replace = TRUE, prob = c(0.55,0.25,0.10,0.08,0.02))
  } else {
    number[i] = 0
  }
}
data = data.frame(sibling,number)
data
```

```
##    sibling number
## 1        1      4
## 2        1      1
## 3        1      2
## 4        0      0
## 5        1      2
## 6        1      1
## 7        0      0
## 8        1      1
## 9        1      1
## 10       1      2
```

```
## 11       1       1
## 12       0       0
## 13       0       0
## 14       0       0
## 15       1       2
```

Your friend had told you before you took the survey that on average a person will have one other sibling. Run a hypothesis test to see if a person does indeed have on average one sibling or not.

$$H_0 : \mu = 1$$

$$H_a : \mu \neq 1$$

Question: Do we use Z or t for the test statistic here?

Answer: t because we do not know the population standard deviation and we do not have a large sample size.

```
xbar = mean(data$number)
n = length(data$number)
df = n - 1
s = sd(data$number)
tts = (xbar - 1)/(s / sqrt(n))
tts
```

```
## [1] 0.4588315
```

```
2*pt(abs(tts),df,lower.tail = FALSE)
```

```
## [1] 0.6533977
```

Question: Do we reject or fail to reject H_0?

Answer: We would fail to reject H_0 because the p-value > 0.05.

Okay, now we can move onto the next portion of lecture content.

Simple Linear Regression

In our final lecture, we will go over a basic modeling technique in statistics known as linear regression. We will be specifically focusing on simple linear regression as anything more would require multivariate calculus and linear algebra to understand. That is clearly beyond the scope of this course. Regression is the study of dependence. It is used to answer interesting questions about how one or more predictors influence a response. Simple linear regression is when we have only one predictor, while multiple linear regression is when you have more than one predictor. So we will be looking at how one predictor influences a response. Remember making those best fit lines in algebra class and thinking wondering when you would actually use them in a real world application? This is it here.
The *simple linear regression model* consists of a mean function and variance function. The model takes the averages of the values and uses that to predict what the true value will be. The formula for a simple linear regression model is ...

$$\mathbb{E}(Y|X = x) = \beta_0 + \beta_1 x$$

$$Var(Y|X = x) = \sigma^2$$

You can read this as, given x is a certain value, then you expect the dependent variable y to be this value, with a variance of σ^2. Now let's look at the two parameters. The first one, β_0 is the intercept of the line, and β_1 is the slope (rate of change) of the line. We can get all possible straight lines by varying these two parameters. The values are usually unknown and must be estimated by the data we have collected. Due to estimation and trying to make a best fit line for our data, the line will not perfectly predict the data. This results in statistical error (e_i) for each data point i. Using this, we can then augment our model to be ...

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

This would make modeling more tricky if we could not make some assumptions about the errors.

1. $E(e_i|x_i) = 0$, meaning on average, the errors are 0, so they do not affect the model.
2. All e_i are independent, which means that the value of the error for one case give us no information about the value of the error for another case.

DISCLAIMER: You will have to check that these assumptions hold when you are making a model. However, those techniques are mostly beyond the scope of this course and fixing the model to make the errors follow these two assumptions is. However, there is one thing you can do to check to make sure your errors are following these two assumptions. We will look at that later when we go through an example.

Example 2; How Much Do Yards Count?

We all know that in football the more yards your offense can get, the more touchdowns your team will score. However, how much does getting that one extra yard really do? This is something that linear regression is useful for. Let's first look at the relationship using a scatter plot (#Lecture5 & #Lab5).

```
cfb21 <- read.csv("~/Desktop/DPISu22/Data Sets/cfb21.csv", stringsAsFactors=TRUE)
#DO NOT DELETE THIS CODE BELOW!! It fixes the offensive yards column such that it
#can be used in the regression model.
library(tidyverse)
```

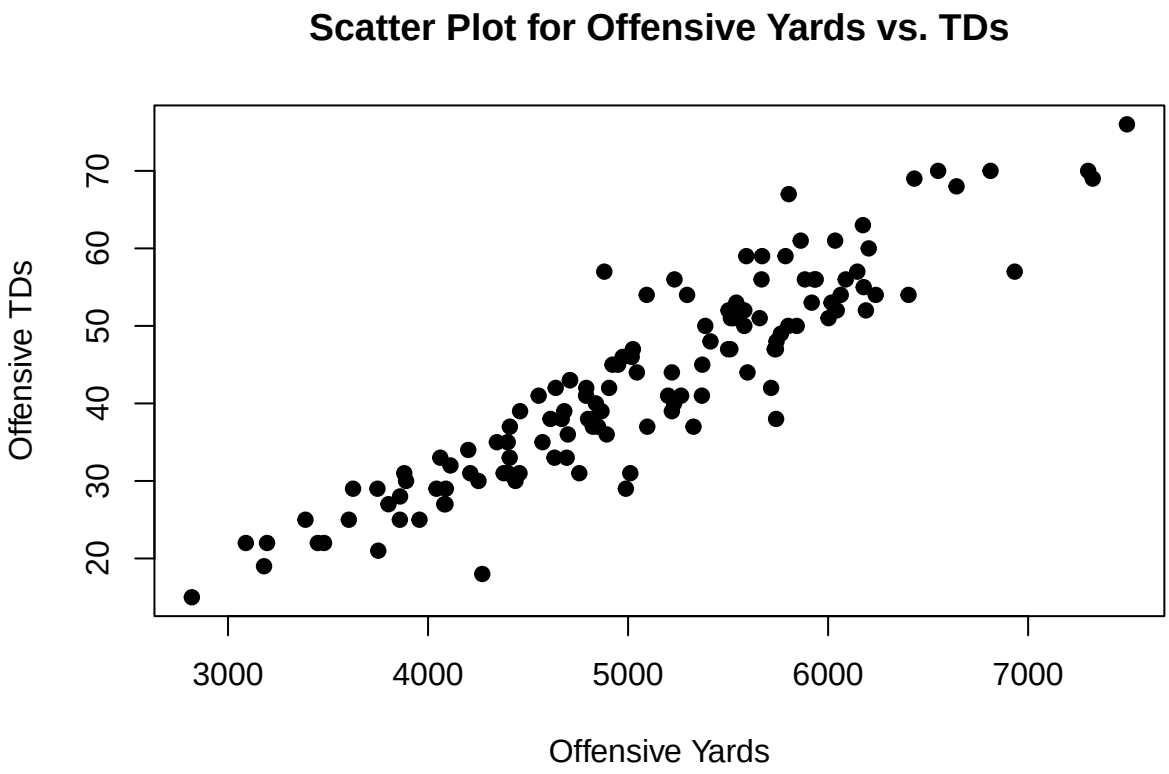
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
cfb21$Off.Yards = str_remove_all(cfb21$Off.Yards, ",")
cfb21$Off.Yards <- as.numeric(cfb21$Off.Yards)
```

```
plot(cfb21$Off.Yards,cfb21$Off.TDs, xlab = "Offensive Yards", ylab = "Offensive TDs",
     main = "Scatter Plot for Offensive Yards vs. TDs", pch = 19)
```



You can see the clear positive linear relationship here. This means it's time to start coding the model. Coding a linear model in `RStudio` is pretty easy, the syntax is ...
`model_name = lm(y ~ x, data = data_name)`

```
football = lm(Off.TDs ~ Off.Yards, data = cfb21)
```

Okay, you have now made it, but how do you see all the information (such as the coefficients)? That can be done using the `summary()` function. You will simply type in `summary(model_name)`.

```
summary(football)
```

```
##
## Call:
## lm(formula = Off.TDs ~ Off.Yards, data = cfb21)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.7044  -3.2333   0.2015   3.0961  16.4590
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.216e+01  2.512e+00  -8.824 7.01e-15 ***
## Off.Yards    1.285e-02  4.871e-04  26.373  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 5.154 on 128 degrees of freedom
## Multiple R-squared:  0.8446, Adjusted R-squared:  0.8434
## F-statistic: 695.5 on 1 and 128 DF,  p-value: < 2.2e-16
```

Okay, that is a *lot* of output. Let's break this down so you know what all of the numbers are. The `Call:` is just going to be the code that you typed in above for the model. `RStudio` is just telling you what you typed in. The `Residuals:` are the errors in the model (e_i) and it is giving you a basic five number summary about them. The meat of the output is in the `Coefficients:` part. Notice how it clearly labels the intercept and what the coefficient is. The standard error tells your the standard deviation of the coefficient estimates. We are estimating them, so there will be some variability between what you estimate and the true value of it. They use complicated formulas and their interpretation is a bit beyond the scope of this course, so we will refrain from going over them for now. Next is the `t value` and `Pr(>|t|)`. These are the t test statistic of the estimate and the p-value for the coefficient using the following hypothesis test; $H_0 : \beta_i = 0$ vs. $H_a : \beta_i \neq 0$. The significance codes indicate statistical significance for different levels (of α). You want your coefficients to be as statistically significant as possible because non-significant coefficients are mostly useless in a model. The final number that you have to worry about is the `Multiple R-squared:`. It tells us how much variation for y is explained by using the variable x chosen. Ideally, you want a high R^2, but you do not want it too high. Anything about 0.55 or lower means the model is not good at predicting and anything above 0.95 means the model might be only good at predicting your data, not a real world event. Okay, now let's write out the model using the output above.

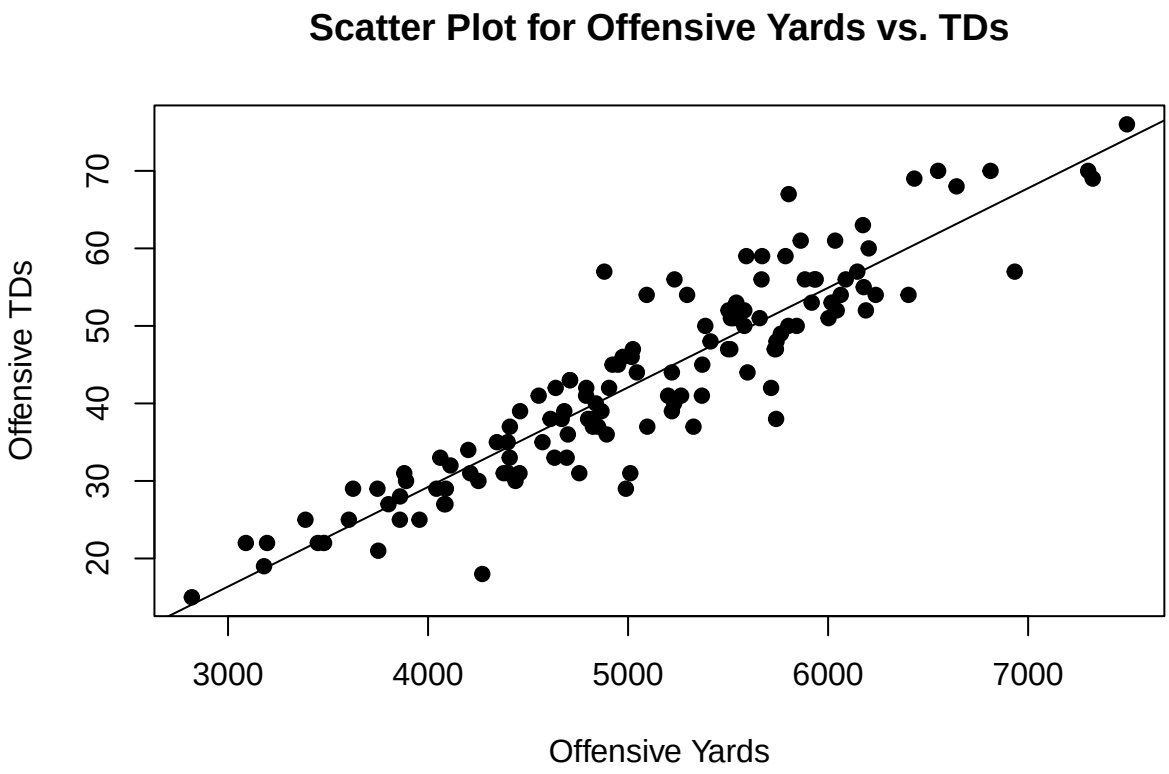
$$OffTDs = -22.16 + 0.01285 OffYards$$

The negative intercept seems strange here. It can mean one of two things.

1. It takes a lot of offensive yards to score a significant number of TDs.
2. There are more variables that go into scoring offensive TDs than just offensive yards.

The second is the likelier option. However, that would fall into multiple linear regression. For now, just take a minute to think about other possible variable that would affect how many offensive TDs a team scores. For now, let's look at how the model looks on our data. You can do this by using the `abline()` function after you make the scatter plot. The syntax is simply `abline(model_name)`.

```
plot(cfb21$Off.Yards,cfb21$Off.TDs, xlab = "Offensive Yards", ylab = "Offensive TDs",
     main = "Scatter Plot for Offensive Yards vs. TDs", pch = 19)
abline(football)
```

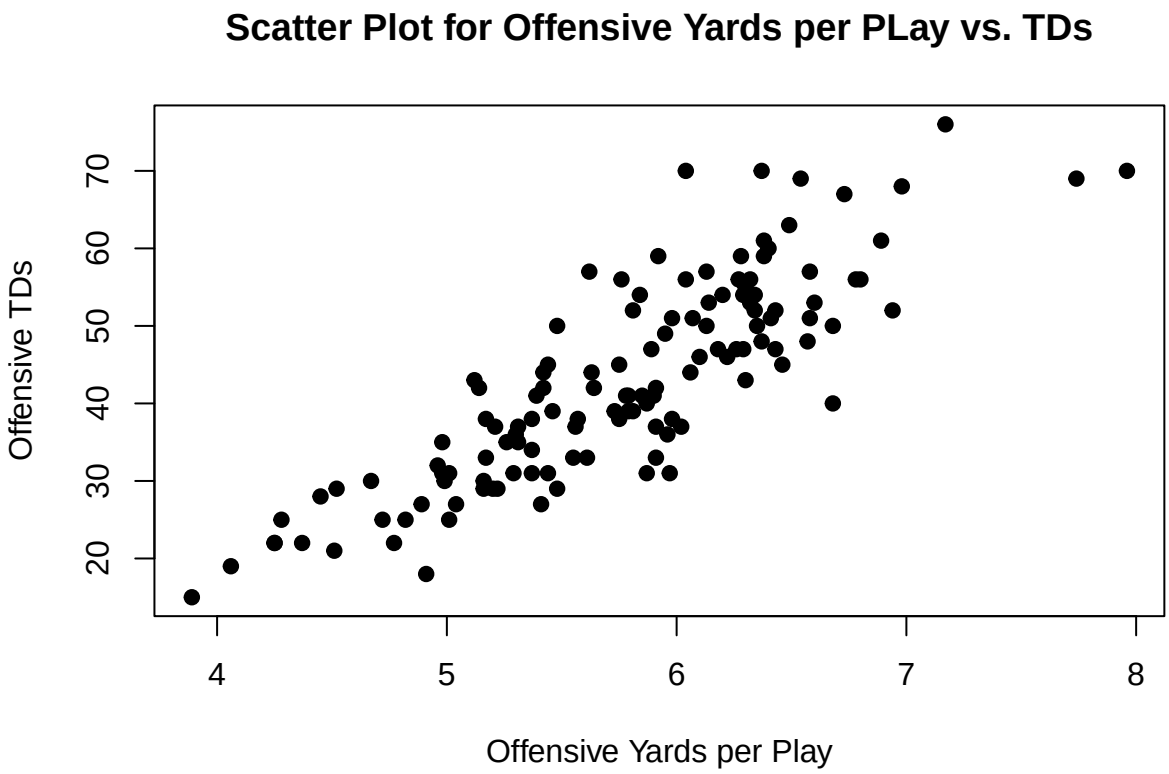


Not bad, but as we said before, there are most likely more factors that affect how many offensive TDs that are scored than just offensive yards. There might even be something better. Let's look at offensive yards per play and see.

Example 3; Another Way?

Let's repeat example 2, but this time using offensive yards per play as our x instead.

```
plot(cfb21$Off.Yards.Play,cfb21$Off.TDs, xlab = "Offensive Yards per Play",
     ylab = "Offensive TDs", main = "Scatter Plot for Offensive Yards per PLay vs. TDs",
     pch = 19)
```



Again, we can see another clear positive, linear relationship. Let's now make the model.

```
football2 = lm(Off.TDs ~ Off.Yards.Play ,data = cfb21)
summary(football2)
```

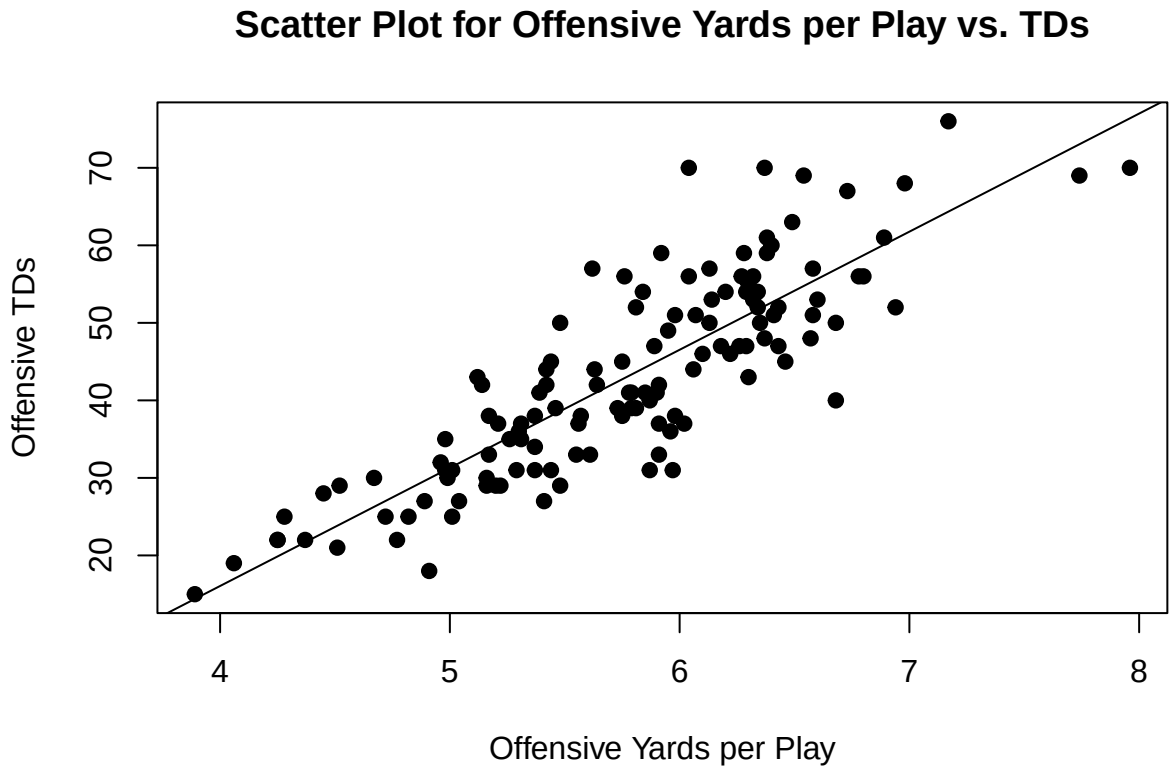
```
##
## Call:
## lm(formula = Off.TDs ~ Off.Yards.Play, data = cfb21)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.885  -4.353  -0.663   4.092  22.864
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -44.8789     4.8116  -9.327 4.21e-16 ***
## Off.Yards.Play  15.2342     0.8276  18.408  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.846 on 128 degrees of freedom
## Multiple R-squared:  0.7258, Adjusted R-squared:  0.7237
## F-statistic: 338.9 on 1 and 128 DF,  p-value: < 2.2e-16
```

Now write out the model.

$$OffTDs = -44.8789 + 15.2342 OffYardsPlay$$

Here we can notice a few things. The first is that the intercept is still negative, so it looks like we would likely be better off making a multiple linear regression model. The second, is that the R^2 is lower in this model by a significant amount. The coefficients are still statistically significant, but it would appear the total offensive yards are better at explaining the variation than the offensive yards per play.

```
plot(cfb21$Off.Yards.Play,cfb21$Off.TDs, xlab = "Offensive Yards per Play",
     ylab = "Offensive TDs", main = "Scatter Plot for Offensive Yards per Play vs. TDs",
     pch = 19)
abline(football2)
```



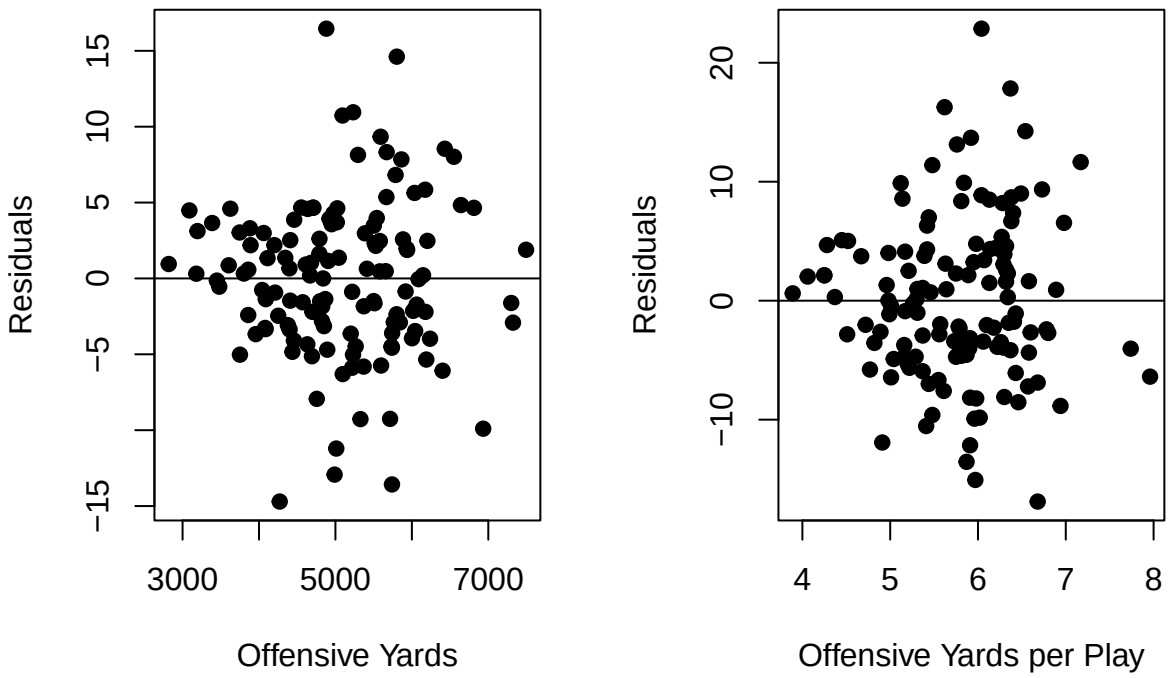
Notice here how there is a distance between the data points and the line. The line does not perfectly predict all of the data. Those vertical distances are the residuals/errors (e_i). Remember earlier when I said we need those two assumptions to hold true and there was a simple way to check them? Well it turns out that `RStudio` calculated those errors for you and they can be easily visualized in a plot.

Example 4; Checking Those Pesky Assumptions

The plot you will be making to check these two assumptions is called a residual plot. To make it, you plot the e_i on the x-axis, and the x on the y-axis. The syntax for calling the residuals from the model and using them to make the plot are as follows . . .
`plot(model_name$residuals, data$x)`

```
par(mfrow = c(1,2))
plot(cfb21$Off.Yards, football$residuals, xlab = "Offensive Yards", ylab = "Residuals",
     pch = 19)
abline(0,0)
```

```
plot(cfb21$Off.Yards.Play, football2$residuals, xlab = "Offensive Yards per Play",
     ylab = "Residuals", pch = 19)
abline(0,0)
```



By looking at the plots, you see absolutely no relationship whatsoever between the residuals and the x variables. This is exactly what we want because it shows that the two assumptions have held up. If you see anything *besides* this kind of plot, then there is a problem with your model. Fixing issues like that fall into model diagnostics, which are meant for a college level course in linear regression. This is just a taste of what it is. Do not be scared about using it for your projects. We will not be grading them that meticulously.

The End

We have now reached the end of lecture content for the summer. Congratulations on being almost done. I hope you have had an enjoyable summer with both myself and Abhi. We both look forward to watching your presentations and reading your project reports.

End of Lecture 14 Notes