**Linear Regression**

**with Prof. Rafael Irizarry**

**Table of Contents**

[1. Introduction to Regression 2](#_Toc26418549)

[1.1. : Baseball as a Motivating Example 2](#_Toc26418550)

[1.1.1. Motivating Example: Moneyball 2](#_Toc26418551)

[1.1.2. Baseball Basics 2](#_Toc26418552)

[1.1.3. Bases on Balls or Stolen Bases? 3](#_Toc26418553)

[1.2. : Correlation 7](#_Toc26418554)

[1.2.1. Correlation 7](#_Toc26418555)

[1.2.2. Correlation Coefficient 8](#_Toc26418556)

[1.2.3. Sample Correlation is a Random Variable 10](#_Toc26418557)

[1.3. : Stratification and Variance Explained 12](#_Toc26418558)

[1.3.1. Anscombe's Quartet/Stratification 12](#_Toc26418559)

[1.3.2. Bivariate Normal Distribution 18](#_Toc26418560)

[1.3.3. Variance Explained 20](#_Toc26418561)

[1.3.4. There are Two Regression Lines 20](#_Toc26418562)

[2. Linear Models 21](#_Toc26418563)

[2.1. Introduction to Linear Models 21](#_Toc26418564)

[2.1.1. Confounding: Are BBs More Predictive? 21](#_Toc26418565)

[2.1.2. Stratification and Multivariate Regression 21](#_Toc26418566)

[2.1.3. Linear Models 25](#_Toc26418567)

[2.2. Least Squares Estimates 27](#_Toc26418568)

[2.2.1. Least Squares Estimates (LSE) 27](#_Toc26418569)

[2.2.2. The lm Function 28](#_Toc26418570)

[2.2.3. LSE are Random Variables 30](#_Toc26418571)

[2.2.4. Advanced Note on LSE 32](#_Toc26418572)

[2.2.5. Predicted Variables are Random Variables 32](#_Toc26418573)

[2.3. Tibbles, do, and broom 35](#_Toc26418574)

[2.3.1. Advanced dplyr: Tibbles 35](#_Toc26418575)

[2.3.2. Tibbles: Differences from Data Frames 37](#_Toc26418576)

[2.3.3. Do 39](#_Toc26418577)

[2.3.4. Broom 43](#_Toc26418578)

[2.4. Regression and Baseball 47](#_Toc26418579)

[2.4.1. Building a Better Offensive Metric for Baseball 47](#_Toc26418580)

[2.4.2. Building a Better Offensive Metric for Baseball: Linear Programming 55](#_Toc26418581)

[2.4.3. On Base Plus Slugging (OPS) 57](#_Toc26418582)

[2.4.4. Regression Fallacy 58](#_Toc26418583)

[2.4.5. Measurement Error Models 63](#_Toc26418584)

[3. Correlation is Not Causation 67](#_Toc26418585)

[3.1. Spurious Correlation 67](#_Toc26418586)

[3.2. Correlation is Not Causation: Outliers 70](#_Toc26418587)

[3.3. Correlation is Not Causation: Reversing Cause and Effect 72](#_Toc26418588)

[3.4. Correlation is Not Causation: Confounders 73](#_Toc26418589)

[3.5. Simpson's Paradox 78](#_Toc26418590)

**List of Figures**

[Figure 1. Scatterplot of Home Run(HR) per game vs Run per game (R) 4](#_Toc26418591)

[Figure 2. Run per game(R ) vs stolen bases per game (SB) 5](#_Toc26418592)

[Figure 3. Bases on balls per game (BB) vs Run per games (R) 6](#_Toc26418593)

[Figure 4. Scatterplot of father and son heights in galton data set 8](#_Toc26418594)

[Figure 5. Examples of data set with correlation coefficient between -0.99 to 0.99 9](#_Toc26418595)

[Figure 6. Histogram of Monte-Carlo simulation results of correlation between father and son in GaltonFamilies data set 10](#_Toc26418596)

[Figure 7. QQ plot of Normal distribution approximation of Galton data set with N=25 11](#_Toc26418597)

[Figure 8. Anscombe’s quartet visualization 12](#_Toc26418598)

[Figure 9. Boxplots of father taller than 72 inches (rounded to the closest inch) and sons 13](#_Toc26418599)

[Figure 10. Scatter plot of son height conditioned on the height of their father 14](#_Toc26418600)

[Figure 11. Standardized heights of son versus father with a line that has a slope equal to the correlation 15](#_Toc26418601)

[Figure 12. Father and son height with regression line 16](#_Toc26418602)

[Figure 13. Father and son height normalized with regression line 17](#_Toc26418603)

[Figure 14. Son height stratify by standardized father heights 19](#_Toc26418604)

[Figure 15. Scatterplot of Runs vs Bases on balls stratified by Home runs 22](#_Toc26418605)

[Figure 16. Scatter plot of Home run vs Runs stratified by Base on Balls 24](#_Toc26418606)

[Figure 17. Beta1 vs RSS 28](#_Toc26418607)

[Figure 18. Histogram of the estimated beta0’s and beta1’s 30](#_Toc26418608)

[Figure 19. Predicted son heights based on their father height with confidence intervals 33](#_Toc26418609)

[Figure 20. Y\_hat versus father height 34](#_Toc26418610)

[Figure 21. Home runs prediction with confidence interval by strata 46](#_Toc26418611)

[Figure 22. Prediction of 2002 runs 49](#_Toc26418612)

[Figure 23. Players metric by salary 53](#_Toc26418613)

[Figure 24. Players metric vs salary for players that started before 1997 54](#_Toc26418614)

[Figure 25. OPS vs prediction of runs 57](#_Toc26418615)

[Figure 26. performance of 2013 vs 2014 62](#_Toc26418616)

[Figure 27. Measurement of the ball position versus time 63](#_Toc26418617)

[Figure 28. Observed data versus predicted data 65](#_Toc26418618)

[Figure 29. Correlation between Divorce rates and margarine consumption 67](#_Toc26418619)

[Figure 30. Distribution of the Monte Carlo simulation 69](#_Toc26418620)

[Figure 31. High correlated data x and y 70](#_Toc26418621)

[Figure 32. Plot of the ranks plotted against each other 71](#_Toc26418622)

[Figure 33. total percent admitted to a major versus the percent of women that make up the applicants 75](#_Toc26418623)

[Figure 34. percentage of applicants that were accounted by gender 76](#_Toc26418624)

[Figure 35. Gender acceptance stratify by major 77](#_Toc26418625)

[Figure 36. Example of a scatter plot of 2 variables x and y 78](#_Toc26418626)

[Figure 37. Example of the same scatter plot stratify by a confounder z 78](#_Toc26418627)

1. Introduction to Regression
   1. : Baseball as a Motivating Example
      1. Motivating Example: Moneyball

As motivation for this course we’ll go back to 2002 and try to build a baseball team with a limited budget. Note that in 2002, the Yankees payroll was almost $130 million, and had more than tripled the Oakland A’s $40 million budget. Statistics have been used in baseball since its beginnings. Note that the data set we will be using, included in the Lahman Library, goes back to 19th century. For example, a summary of statistics we will describe soon, the batting average (BA), has been used to summarize a batter’s success for decades. Other statistics such as home runs (HR) , runs batted in (RBI) and stolen bases (SB), we’ll describe all this soon, are reported for each player in the game summaries included in the sports section of newspapers. Players are rewarded for high numbers. Although summary statistics were widely used in baseball, data analysis per se was not.

These statistics were arbitrarily decided on without much thought as to whether they actually predicted or were related to helping a team win. This all changed with Bill James. In the late 1970s, this aspiring writer and baseball fan started publishing articles describing more in-depth analysis of baseball data. He named the approach of using data to predict what outcomes best predict if a team wins sabermetrics. Until Billy Beane made sabermetrics the center of his baseball operations. Bill James’ work was mostly ignored by the baseball world. Today, pretty much every team uses the approach, and it has gone beyond baseball into other sports.

In this course, to simplify the example we use, we’ll focus on predicting scoring runs. We will ignore pitching and fielding, although those are important as well. We will see how regression analysis can help develop strategies to build a competitive baseball twam with a constrained budget. The approach can be divided into two separate data analyses. In the first, we determine which recorded player specific statistics predict runs. In the second, we examine if players were undervalued based on what our first analysis predicts.

* + 1. Baseball Basics

We actually don’t need to understand all the details about the game of baseball, which has over 100 rules, to see how regression will help us find undervalued players. Here, we distill the sport to the basic knowledge one needs to know effectively to attack the data science challenge.

The goal of a baseball game is to score more runs, they’re like points, than the other team. Each team has 9 batters that bat in a predetermined order. After the 9th batter hits, we start with the first again. Each time they come to bat, we call it a plate appearance PA. At each plate appearance the other team’s pitcher throws the ball an you try to hit it. The plate appearance ends with a binary outcome; you either make an out, a failure, or you don’t, a success, and you get to run around the bases and potentially score a run. Each team get nine tries, referred to as innings, to score runs. Each inning ends after three outs, after you’ve failed three times. From these examples we see how luck is involved in the process. When you bat you want to hit the ball hard. If you hit it hard enough, it’s a home run, the best possible outcome as you get at least one automatic run. But sometimes, due to chance, you hit the ball very hard and a defender catches it, which makes it an out, a failure. In contrast, sometimes you hit the ball softly but it lands just in the right place. You get a hit which is a success. The fact that there is chance involved hints at why probability models will be involved in all this. Now there are several ways to succeed, understanding this distinction will be important for our analysis. When you hit the ball you want to pass as many bases as possible. There are 4 bases with the 4th one called home plate.

Home plate is where you start, where you try to hit, so the bases form a cycle. If you get home, you score a run. By simplifying, there is 5 ways you can succeed, in other words not making an out:

* First one is called a base on balls, this is when the pitcher does not pitch well and you get to go to first base.
* A single is when you hit the ball and you get to first base
* A double is when you hit the ball and you go past first base to second
* Triple is when you do that but go to third.
* A home run is hen you hit the ball and go all the way home and score a run.

If you get to a base, you still have a chance of getting home and scoring a run if the next batter hit successfully. While you are on base, you can also try to steal a base. If you run fast enough, you can try to go from first to second or from second to third without the other team tagging you.

Now historically, the batting average has been considered the most important offensive statistic. To define this average, we define a hit (H) and an at bat AB). Singles, doubles, triples and home runs are hits. But remember, there’s a 5th way to be successful, the base on balls. That is not a hit. An at bat is the number of times you either get a hit or make an out, bases on balls are excluded. The batting average is simply hits divided by at bats. And it is considered the main measure of a success rate. Today, in today’s game, this success rates ranges from player to player from about 20% to 38%. We refer to the batting average in thousands. So for example, if your success rate is 25% we say you’re batting 250. One of Bill James’ first important insights is that the bathing average ignores bases on balls but bases on balls is a success. So a player that gets many more bases on balls than the average player might not be recognized if he does not excel in batting average. But is this player not helping produce runs? No award is given to the player with the most bases on balls. In contrast, the total number of stolen bases are considered important and a an award is given to the player with the most. But players with high total of stolen bases also make outs as they do not always succeed. So foes a player with a high stolen base total help produce runs? Can we use data science to determine if it’s better to pay for bases on balls or stolen bases? One of the challenges in this analysis is that it is not obvious how to determine if a player produces runs because so much depends on his teammates. We do keep track of the number of runs scored by our player. But note that if you hit after someone who hits many home runs, you will score many runs. But these runs don’t necessarily happen if we hire this player but not his home run hitting teammate. However, we can examine team level statistics. How do teams with many stolen bases compare to team with few? How about bases on balls? We have data, let’s examine some.

* + 1. Bases on Balls or Stolen Bases?

Let’s start looking at some baseball data and try to answer your questions using these data. First one, do team that hit more home runs score more runs? We know what the answer to this will be, but let’s look at the data anyways. We’re going to examine from 1961 to 2001. We end at 2001 because, remember, we’re back in 2002, getting ready to build a team. We started in 1961, because that year, the league changed from 154 games to 162 games. The visualization of choice when exploring the relationship between two variables like home runs and runs is a scatterplot, the following code shows you how to make that scatterplot:

*library(Lahman)*

*ds\_theme\_set()*

*Teams%>%filter(yearID %in% 1961:2001)%>%*

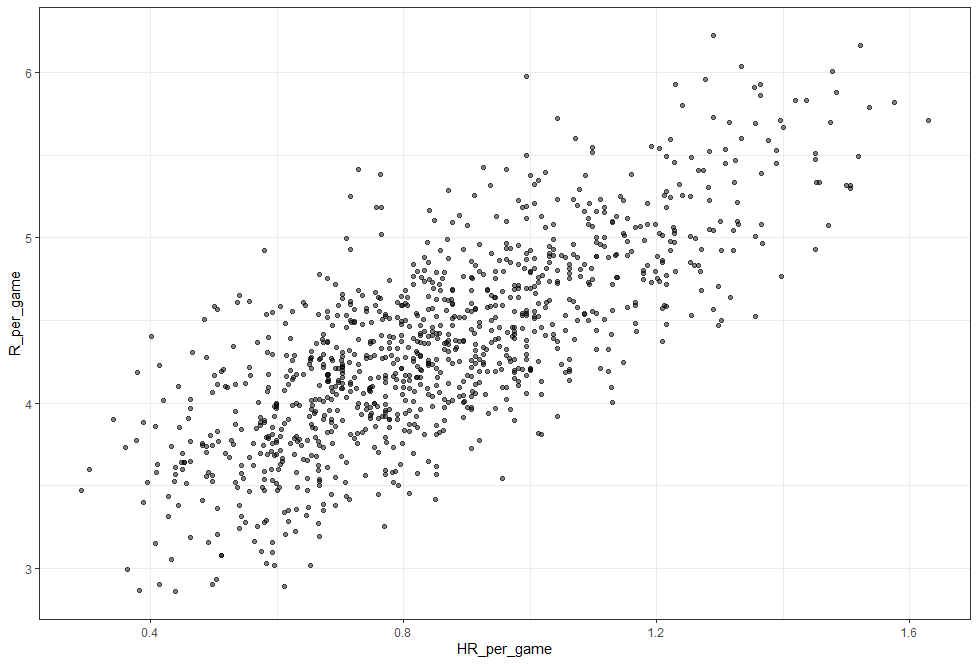
*+ mutate(HR\_per\_game=HR/G,R\_per\_game=R/G)%>%*

*+ ggplot(aes(HR\_per\_game,R\_per\_game))+*

*+ geom\_point(alpha=0.5)*

We start by loading the Lahman library that has all these baseball statistics, and then we simply make a scatterplot using ggplot:

1. Scatterplot of Home Run(HR) per game vs Run per game (R)



The plot shows a very strong association, teams with more home runs tended to score more runs. Now, let’s examine the relationship between stolen bases and wins:

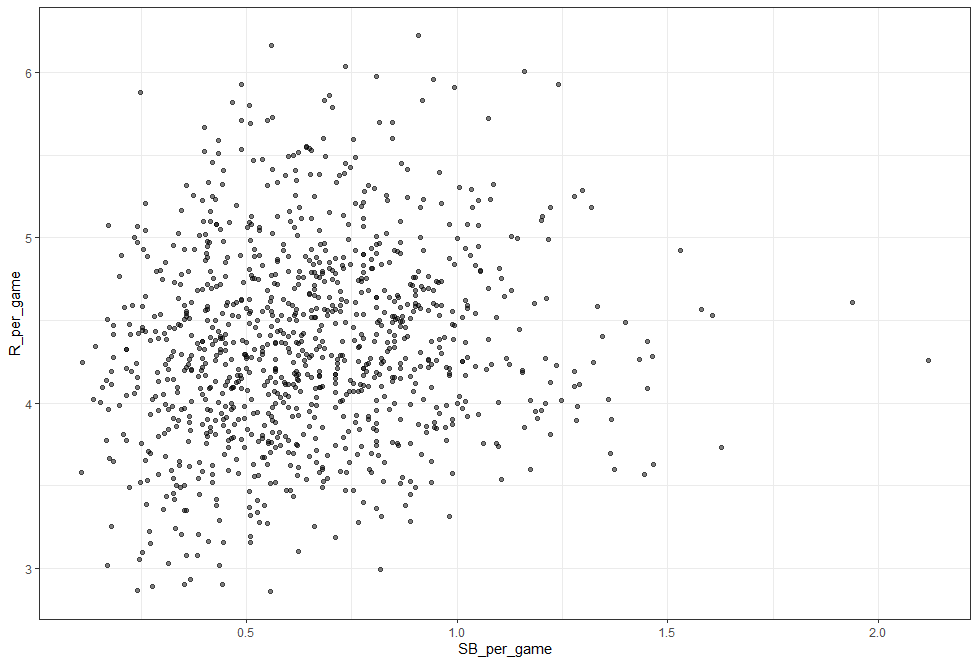
*Teams%>%filter(yearID %in% 1961:2001)%>%*

*+ mutate(SB\_per\_game=SB/G,R\_per\_game=R/G)%>%*

*+ ggplot(aes(SB\_per\_game,R\_per\_game))+*

*+ geom\_point(alpha=0.5)*

1. Run per game(R ) vs stolen bases per game (SB)



Here the relationship is not as clear. Finally, let’s examine the relationship between bases on balls and runs:

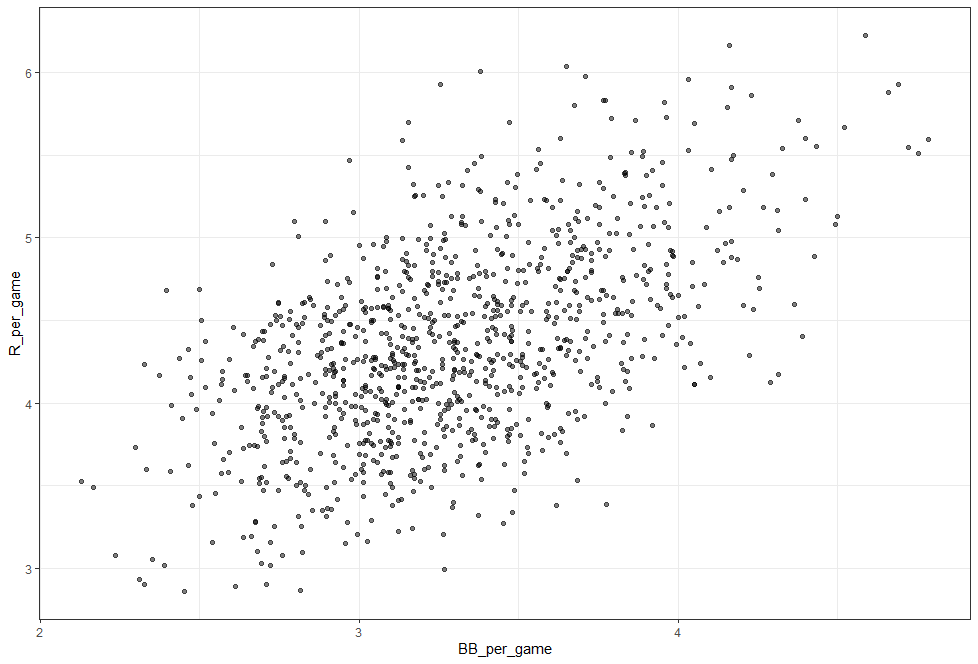
*Teams%>%filter(yearID %in% 1961:2001)%>%*

*+ mutate(BB\_per\_game=BB/G,R\_per\_game=R/G)%>%*

*+ ggplot(aes(BB\_per\_game,R\_per\_game))+*

*+ geom\_point(alpha=0.5)*

1. Bases on balls per game (BB) vs Run per games (R)



Although the relationship is not as string as it was for home run, we do see a pretty strong relationship here. We know that, by definition, homes runs cause runs, because when you hit a home run, at least one run will score. Now it could be that home runs also cause the bases on balls. If you understand the game, you will agree with the previous statement. So, it might appear that a base on ball is causing runs, when in fact, it’s home runs that’s causing both. This is called confounding. An important concept you will learn about. Linear regression will help us parse all this out and quantify the associations. This will then help us determine what players to recruit. Specifically, we will try to predict things like how many more runs will the team score if we increase the number of bases on balls but keep the home runs fixed. Regression will help us answer this question, as well.

* 1. : Correlation
     1. Correlation

Up to now in this series, we have focused mainly on univariate variables. However, in data science application it is very common to be interested in the relationship between two or more variables. We saw this in our baseball example in which we were interested in the relationship, for example between bases on balls and runs. We’ll come back to this example, but we introduce the concepts of correlation and regression using a simpler example. It is the dataset from which regression was born. We examine an example from genetics. Francis Galton studied the variation and heredity of human traits. Among many other traits, Galton collected and studied height data from families to try to understand heredity. While doing this, he developed the concepts of correlation an regression, and a connection to pairs of data that follow a normal distribution. Note that, at the time his data was collected, what we know today about genetics was not yet understood. A very specific question Galton tried to answer was, how much of a son’s height can I predict with parents’ height. Note that this is like predicting runs with bases on balls. We have access to Galton’s family data through the HistData package. HistData stands for historical data. We will create a data set with the heights of fathers and the first sons. The actual data Galton used to discover and define regression. So we have the father and son height data:

*galton\_heights<-GaltonFamilies%>%*

*+ filter(childNum==1 & gender=="male")%>%*

*+ select(father,childHeight)%>%*

*+ rename(son=childHeight)*

Suppose we were to summarize these data. Since both distributions are well approximated by normal distributions, we can use the two averages and two standard deviations as summaries, like this:

*galton\_heights%>%*

*+ summarize(mean(father),sd(father),mean(son),sd(son))*

*mean(father) sd(father) mean(son) sd(son)*

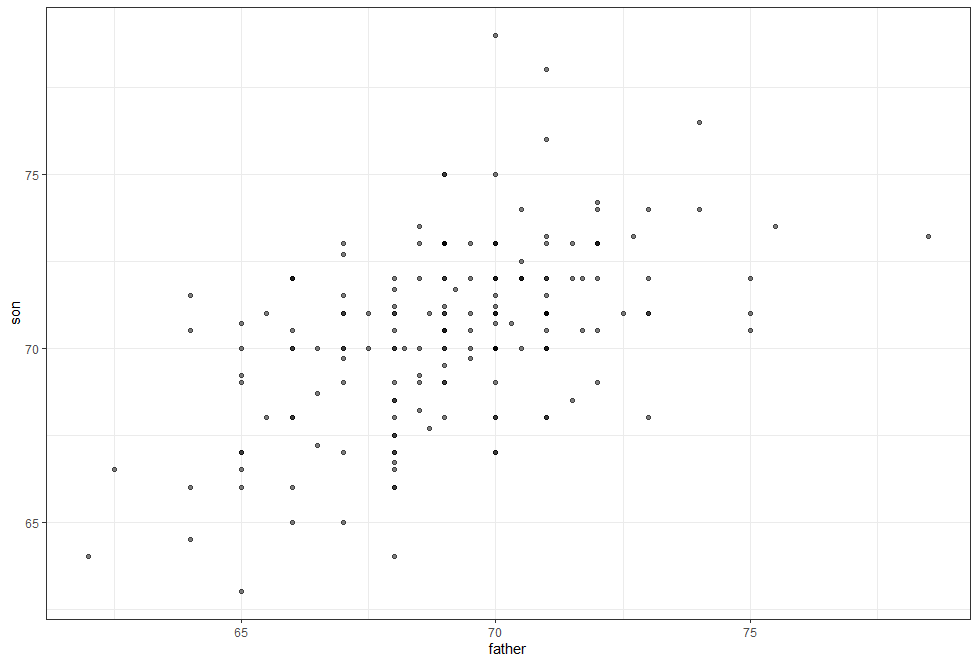
*1 69.09888 2.546555 70.45475 1 2.557061*

You can see the average heights for fathers is 69 inches. The standard deviation is 2.54. For sons, they’re a little taller, because it’s the next generation. The average height is 70.45 inches, and the standard deviation is 2.55 inches. However, this summary fails to describe a very important characteristic of the data that you can see in this figure.:

*galton\_heights%>%ggplot(aes(father,son))+*

*+ geom\_point(alpha=0.5)*

1. Scatterplot of father and son heights in galton data set



The trend that the taller the father, the taller the son, is not described by the summary statistics of the average and the standard deviation. We will learn that the correlation coefficient is a summary of this trend.

* + 1. Correlation Coefficient

The correlation coefficient is defined for a list of pairs with the formula:

Where and are the averages of x and y respectively and , are the standard deviations. The Greek letter (rho) is commonly used in the statistics books to denote this correlation. The reason is that rho is the Greek later for r, the first letter of the word regression. Soon, we will learn about the connection between correlation and regression. To understand why this equation does, in fact, summarize how two variables move together, consider the i-th entry of x is xi minus mu x divided by sigma x SDs away from the average. Similarly, the y which is paired with the xi is yi minus my y divided by sigma y SDs away from the average y. If x an y are unrelated the the product of these two quantities will be positive. That happens when they are both positive or when they are both negative as often as they will be negative. That happens when one is positive and the other is negative, or the other way around. This will average to about 0. The correlation is this average, and therefore, unrelated variables will have a correlation of about 0. If instead the quantities vary together, then we are averaging mostly positive products. Because they’re going to be either positive times positive or negative times negative, and we get a positive correlation. If they vary in opposite direction,w e get a negative correlation. Another thing to know is that we can show mathematically that the correlation is always between -1 and 1. To see this, consider that we can have higher correlation than when we compare a list to itself. That would be perfect correlation. In this case, the correlation is given by this equation:

Which we can show is equal to 1. A similar argument with x and its exact opposite, negative x, proves that the correlation has to be greater or equal to -1. The correlation between father and sons’ height is about 0.5, as shown by this code:

*library(HistData)*

*> data("GaltonFamilies")*

*> galton\_heights<-GaltonFamilies%>%*

*+ filter(childNum==1 & gender=="male")%>%*

*+ select(father,childHeight)%>%*

*+ rename(son=childHeight)*

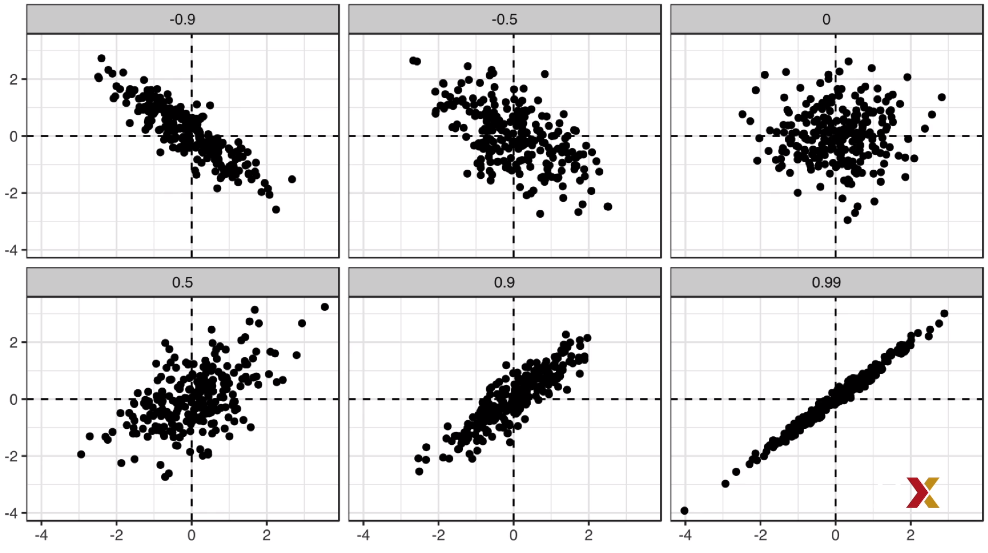
*> galton\_heights%>%summarize(cor(father,son))*

*cor(father, son)*

*1 0.5007248*

We saw what the data looks like when the correlation is 0.5. To see what data looks like for other values of rho, here are 6 examples:

1. Examples of data set with correlation coefficient between -0.99 to 0.99



When the correlation is negative, we see that they go in opposite direction. As x increase, y decreases. When the correlation gets either closer to 1 or -1, we see the cloud of points getting thinner and thinner. When the correlation is 0, we just see a big circle of points.

* + 1. Sample Correlation is a Random Variable

Before we continue describing regression, let’s go over a reminder about random variability. In most data science applications, we do not observe the population, but rather a sample. As with the average and standard deviation, the sample correlation is the most commonly used estimate of the population correlation. This implies that the correlation we compute and use as a summary is a random variable. As an illustration, let’s assume that the 179 pairs of fathers and sons is our entire population. A less fortunate geneticist can only afford to take random sample of 25 pairs. The sample correlation for this random sample can be compute using this code:

*R<-sample\_n(galton\_heights,25,replace=TRUE)%>%*

*+ summarize(cor(father,son))*

Here, the variable R is the random variable. We can run a monte-carlo simulation to see the distribution of this random variable, here we recreate R 1000 times and plot its histogram:

*B<-1000*

*> N<-25*

*> R<-replicate(B,{*

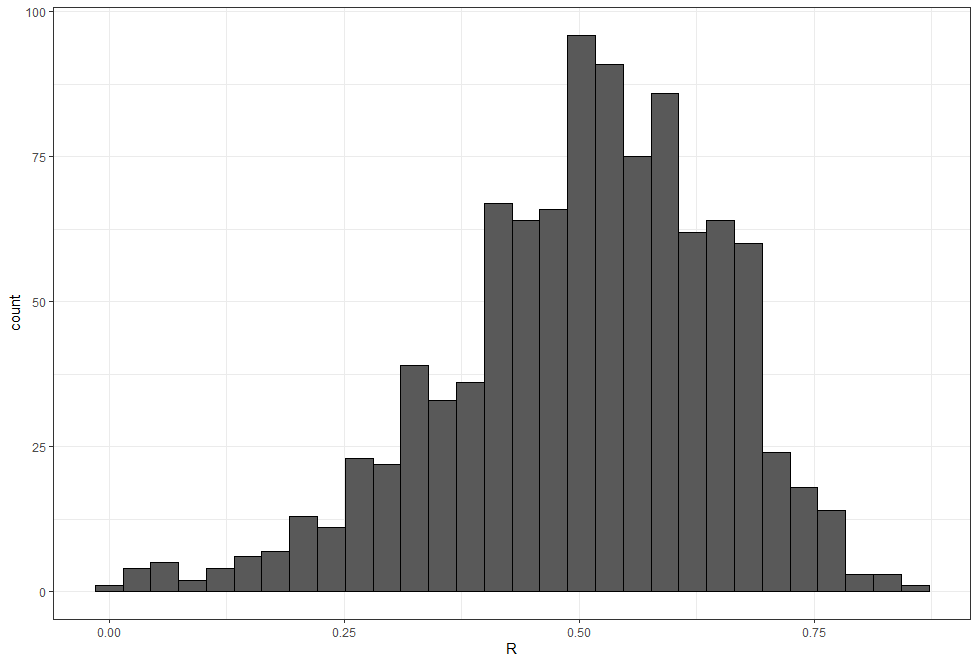
*+ sample\_n(galton\_heights,N,replace=TRUE)%>%*

*+ summarize(r=cor(father,son))%>%.$r*

*+ })*

*data.frame(R)%>%ggplot(aes(R))+geom\_histogram(bindwidth=0.05,color="black")*

1. Histogram of Monte-Carlo simulation results of correlation between father and son in GaltonFamilies data set



We see that the expected value is the population correlation, the mean of these R is 0.5:

*mean(R)*

*[1] 0.5040874*

And that it has a relatively high standard error relative to its size:

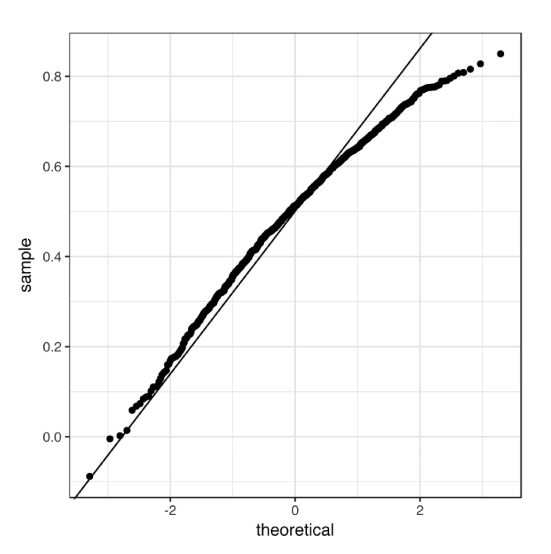
*sd(R)*

*[1] 0.1439084*

This is something to keep in mind when interpreting correlations. It is a random variable, and it can have a pretty large standard error. Also note that because the sample correlation is an average of independent draws, the Central Limit Theorem applies. Therefore, for a large enough sample size N, the distribution of these Rs is approximately normal. The expected value we know is the population correlation. The standard deviation is somewhat more complex but is given by this formula:

In our example N equals to 25 does not appear to be large enough to make the approximation a good one, as we see in this QQ plot

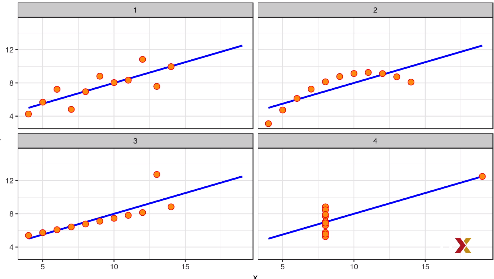
1. QQ plot of Normal distribution approximation of Galton data set with N=25



* 1. : Stratification and Variance Explained
     1. Anscombe's Quartet/Stratification

Correlation is not always a good summary of the relationship between two variables. A famous exampled used to illustrate this are the following 4 artificial data sets, referred to as Anscombe’s quartet:

1. Anscombe’s quartet visualization



All of these pairs have a correlation of 0.82.

Correlation is only meaningful in a particular, to help us understand when it is that correlation is meaningful as a summary statistic, we’ll try to predict the son’s height using the father’s height. This will help motivate and define linear regression. We start by demonstrating how correlation can be useful for prediction. Suppose we are asked to guess the height of a randomly selected son. Because of the distribution of the son height is approximately normal, we know that the average height of 70.5 inches is a value with the highest proportion and would be the prediction with the chances of minimizing the error. But what if we are told father is 72 inches? Do we still guess 70.5 inches for the son? The father is 1.14 standard deviations taller than the average father. So shall we predict that the son is also 1.14 standard deviations taller than the average son? It turns out that this would be an overestimate. To see this, we look at all the sons with fathers who about 72 inches. We do this by stratifying the father’s height. We call this a conditional average, since we are computing the average son height conditioned on the father being 72 inches tall. A challenge when using this approach in practice is that we don’t have many fathers that are exactly 72. In our data set, we only have eight. If we change the number to 72.5, we would only have one father who is that height. This would result in averages with large standard errors, and they won’t be useful for prediction for this reason. But for now, what we’ll do is we’ll take an approach of creating strata of fathers with very similar heights. Specifically, we will round fathers’ height to the nearest inch. This gives us the following prediction for the son of a father that is approximately 72 inches tall. We can use this code and get our answer, which is 71.84:

condtional\_avg<-galton\_heights%>%filter(round(father)==72)%>%

+ summarize(avg=mean(son))%>%.$avg

> condtional\_avg

[1] 71.83571

This is 0.54 standard deviation larger than the average son, a smaller number than the 1.14 standard deviations taller that the father was above the average father. Stratification followed by box plots lets us see the distribution of each group, here is the plot:

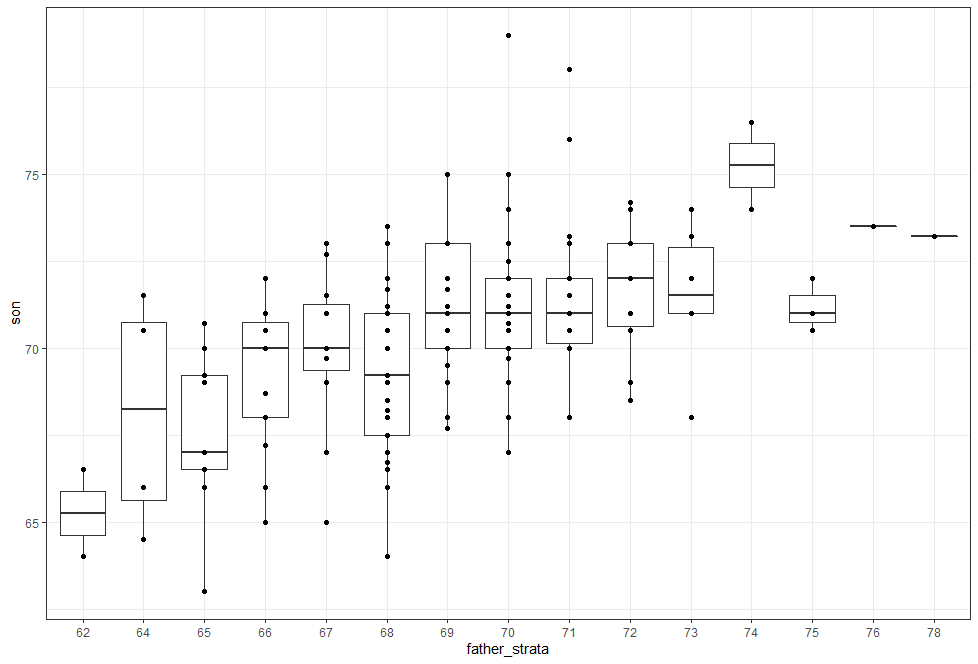
*galton\_heights%>%mutate(father\_strata=factor(round(father)))%>%*

*+ ggplot(aes(father\_strata,son))+*

*+ geom\_boxplot()+*

*+ geom\_point()*

1. Boxplots of father taller than 72 inches (rounded to the closest inch) and sons



We can see that the centers of these groups are increasing with the height, not surprisingly. The means of each group appear to follow a linear relationship. We can make that plot like this:

galton\_heights%>%

+ mutate(father=round(father))%>%

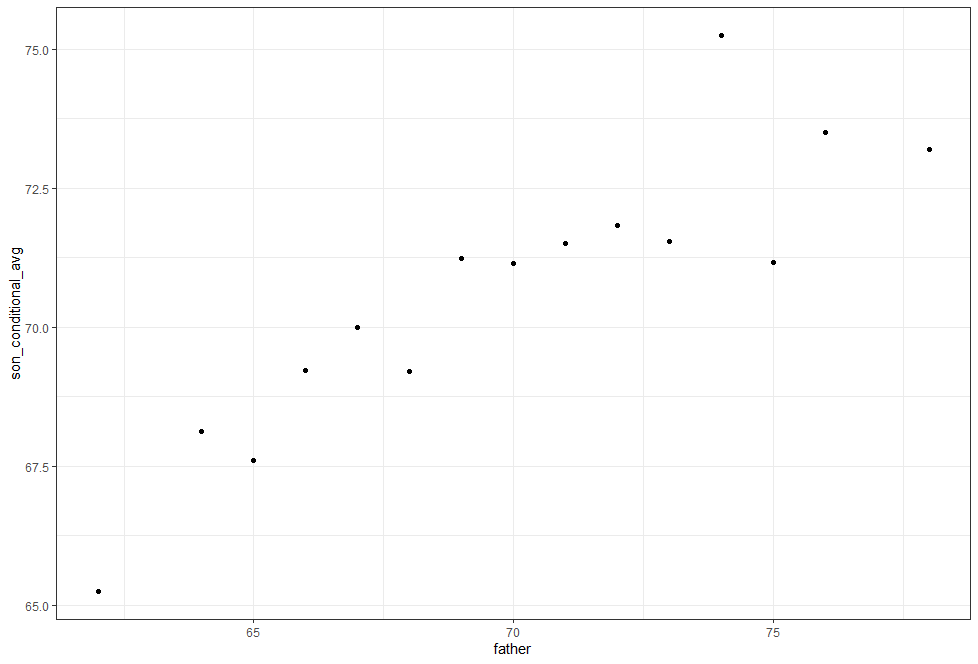
+ group\_by(father)%>%

+ summarize(son\_conditional\_avg=mean(son))%>%

+ ggplot(aes(father,son\_conditional\_avg))+

+ geom\_point()

1. Scatter plot of son height conditioned on the height of their father



The plot seems to follow a line. The slope of this line appears to be about 0.5, which happens to be the correlation between father and son heights. This is not a coincidence. To see this connection, let’s plot the standardized heights against each other, son versus father, with a line that has a slope equal to the correlation:

*r<-galton\_heights%>%summarize(r=cor(father,son))%>%.$r*

*> galton\_heights%>%*

*+ mutate(father=round(father))%>%*

*+ group\_by(father)%>%*

*+ summarize(son=mean(son))%>%*

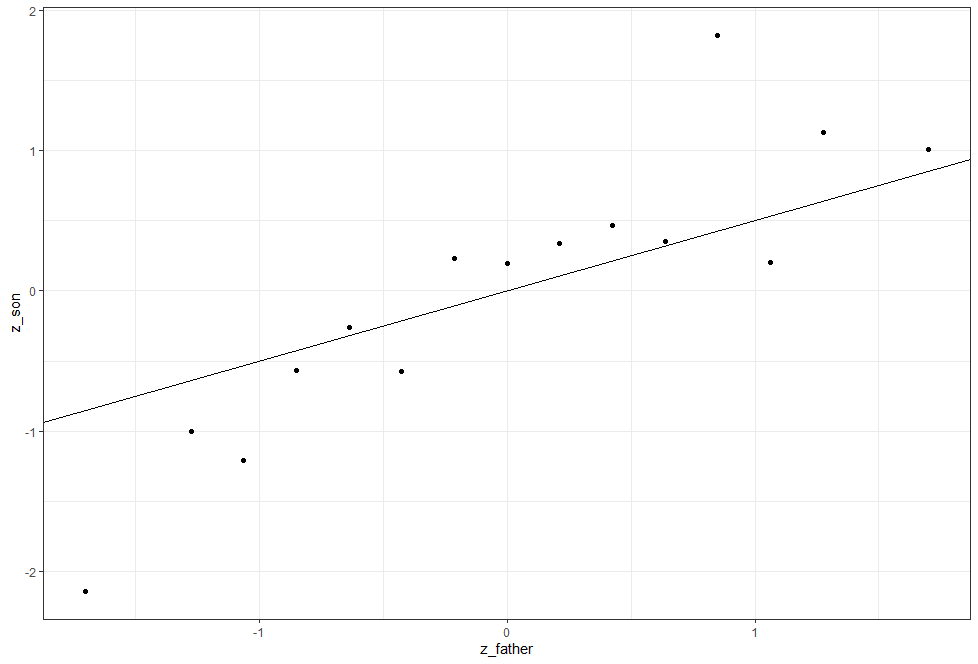
*+ mutate(z\_father=scale(father),z\_son=scale(son))%>%*

*+ ggplot(aes(z\_father,z\_son))+*

*+ geom\_point()+*

*+ geom\_abline(intercept = 0,slope=r)*

1. Standardized heights of son versus father with a line that has a slope equal to the correlation



This line is what we call the regression line. In a later video, we will describe Galton’s theoretical justification for using this line to estimate conditional means. Here, we define it and compute it for the data at hand. The regression line for two variables, x and y, tell us that for every standard deviation sigma x increase above the average mu x. For x, y grows rho standard deviation sigma y above the average mu y . The formula for the regression line is therefore this one:

If there’s perfect correlation, we predict an increase that is the same number of SDs. If there’s zero correlation, then we don’t use x at all for the prediction of y. For values between 0 and 1, the prediction is somewhere in between. If the correlation is negative, we predict a reduction instead of an increase. It is because when the correlation is positive but lower than 1 that we predict something closer to the mean, that we call this regression. The son regresses to the average height. In fact, the title of Galton’s paper was “Regression Towards Mediocrity in Hereditary Stature”. Note that if we write this in the standard form of a line:

The regression line has:

So, if we standardized the variable so they have average 0 and standard deviation 1. Then the regression line has intercept 0 and slope equal to the correlation rho. Let’s look at the original data, father son data, and add the regression line. We can compute the intercept and the slope, here’s the code:

*mu\_x<-mean(galton\_heights$father)*

*mu\_y<-mean(galton\_heights$son)*

*s\_x<-sd(galton\_heights$father)*

*s\_y<-sd(galton\_heights$son)*

*r<-cor(galton\_heights$father,galton\_heights$son)*

*m<-r\*s\_y/s\_x*

*b<-mu\_y-m\*mu\_x*

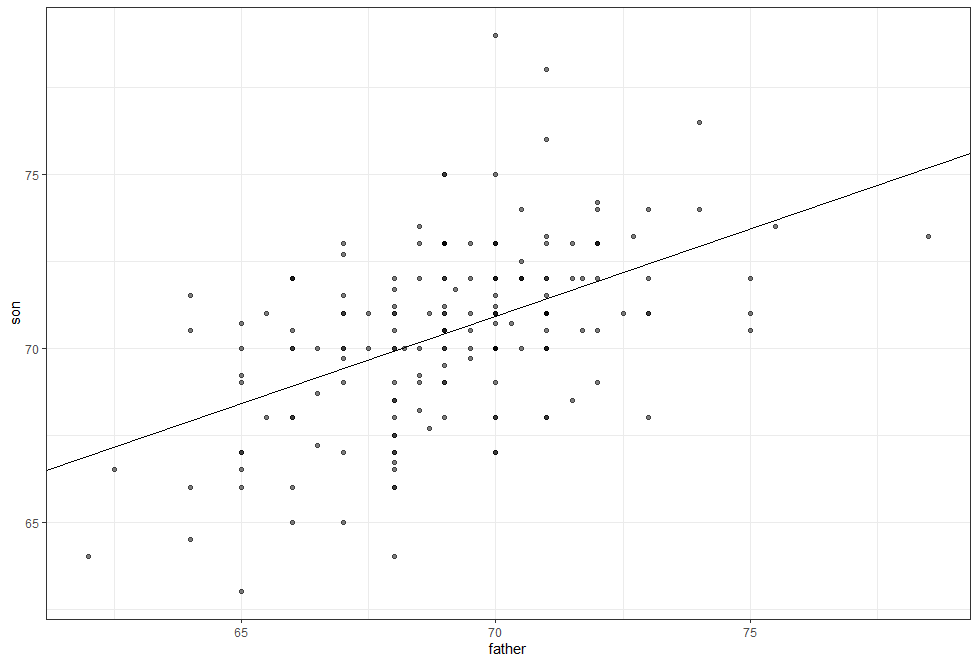
*galton\_heights%>%*

*+ ggplot(aes(father,son))+*

*+ geom\_point(alpha=0.5)+*

*+ geom\_abline(intercept=b,slope=m)*

1. Father and son height with regression line



If we plot the data in standard units, then, as we discussed, the regression line intercept 0 and slope rho:

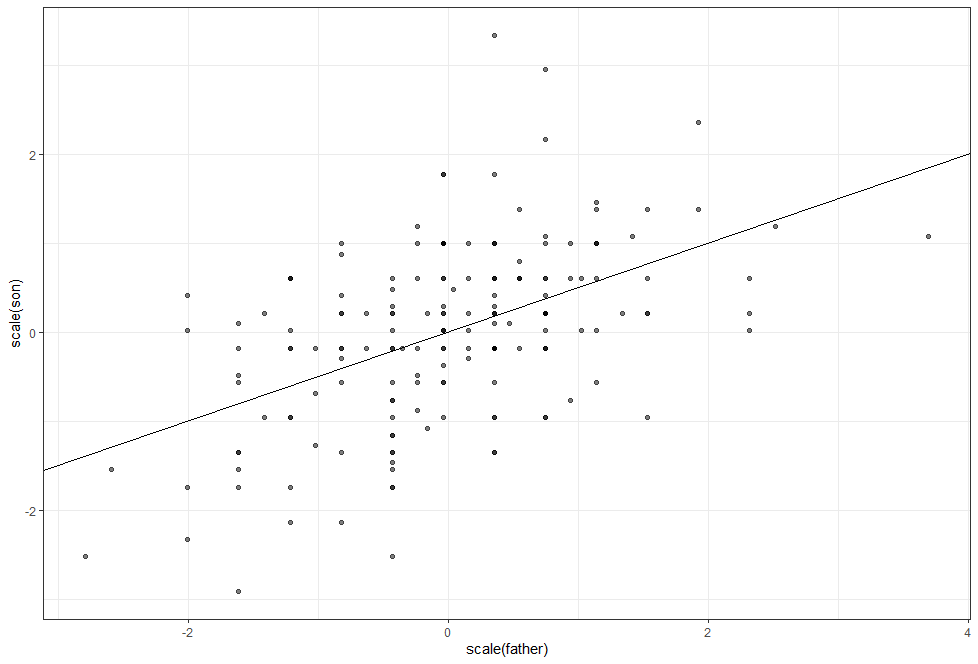
galton\_heights%>%

+ ggplot(aes(scale(father),scale(son)))+

+ geom\_point(alpha=0.5)+

+ geom\_abline(intercept=0,slope=r)

1. Father and son height normalized with regression line



We started by saying that we wanted to use the conditional means to predict the heights of the sons. But then we realized that there were very few data points in each strata. But then we realized that there were very few data points in each strata. When we did this approximation of rounding off the height of the fathers, we found that these conditional means appear to follow a line. And we ended up with the regression line. So the regression line give us the prediction. An advantage of using the regression line is that we used all the data to estimate just two parameters, the slope and the intercept. This makes it much more stable. When we do conditional means, we had fewer data points, which made the estimates have a large standard error, and therefore be unstable. So this is going to give us a much more stable prediction using the regression line. However, are we justified in using the regression line to predict? Galton give us the answer.

* + 1. Bivariate Normal Distribution

Correlation and the regression line are widely used summary statistics. But it is often misused or misinterpreted. Anscombe`s example provided a toy example of data sets in which summarizing with a correlation would be a mistake. But we also see it in the media and in scientific literature as well. The main way we motivate the use of correlation involve what is called the bivariate normal distribution. When a pair of random variables is approximated by a bivariate normal distribution, the scatterplot looks like ovals, like American footballs. They can be thin, when they have high correlation, all the way up to a circle shape when they have no correlation. We saw some examples previously.

A more technical way to define the bivariate normal distribution is the following. First, this distribution is defined for pairs. So we have 2 variables x and y and they have paired values. They are going to be bivariate normally distributed if the following happens:

**If X and Y are normally distributed random variables, and for any group of X, X=x, Y is approximately normal in that group, then the pair is approximately bivariate nomal.**

When we fix x in this way, we then refer to the resulting distribution of the y’s, in the group defined by setting x that way, as the conditional distribution of y given x, written like this:

If we think the height data is well-approximated by the bivariate normal distribution, then we should see the normal distribution hold for each grouping. Here, we stratify the son height by the standardized father heights and see that the assumption appears to hold:

*galton\_heights%>%*

*+ mutate(z\_father=round((father-mean(father))/sd(father)))%>%*

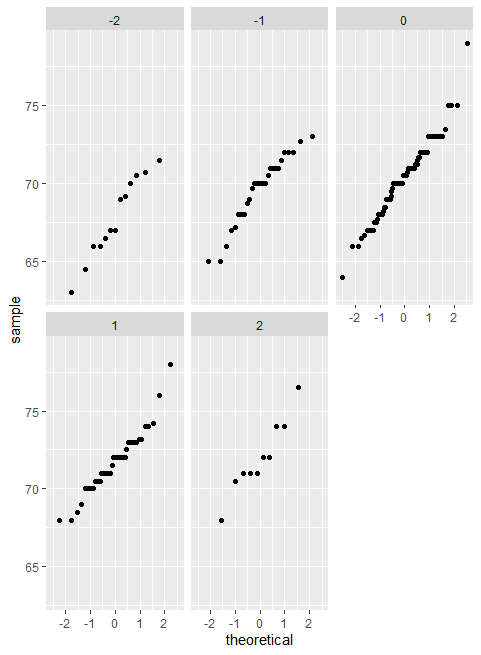
*+ filter(z\_father %in% -2:2)%>%*

*+ ggplot()+*

*+ stat\_qq(aes(sample=son))+*

*+ facet\_wrap(~z\_father)*

1. Son height stratify by standardized father heights



Now, we come back to defining correlation. Galton showed using mathematical statistics that when 2 variables follow a bivariate normal distribution, then for any given x the expected value of the y in pairs for which x is set at that value is:

Note that this is a line with slope:

And intercept:

And therefore, this is the same as the regression line we saw in the last part, that can be written like this:

So in summary, if our data is approximately bivariate, then the conditional expectation which is the best prediction for y given that we know the value of x is given by the regression line.

* + 1. Variance Explained

The theory we’ve been describing also tells us that the standard deviation of the conditional distribution that we described in a previous video is:

This is where statements like x explains such and such percent of the variation in y comes from. Note that the variance of y is sigma squared. That’s where we start if we condition on x, then the variance goes down to:

So, from there, we can compute how much the variance has gone down. It has gone down by rho squared time 100%. So the correlation and the amount of variance explained are related to each other. But it is important to remember that the variance explained statement only makes sense when then data is approximated by a bivariate normal distribution.

* + 1. There are Two Regression Lines

We computed a regression line to predict the son’s height from the father’s height. We used these calculations:

*mu\_x<-mean(galton\_heights$father)*

*mu\_y<-mean(galton\_heights$son)*

*s\_x<-sd(galton\_heights$father)*

*s\_y<-sd(galton\_heights$son)*

*r<-cor(galton\_heights$father,galton\_heights$son)*

*m<-r\*s\_y/s\_x*

*b<-mu\_y-m\*mu\_x*

To get the slope and intercept. This give us the function that the conditional expectation of y given x is:

So, what if we wanted to predict the father’s height based on the son’s? It is important to know that this is not determined by computing the inverse function of what we just saw, which would be this equation here:

We need to compute the expected value of x given y. This give us another regression function together, with slope and intercept computed like this:

*m<-r\*s\_x/s\_y*

*b<-mu\_x-m\*mu\_y*

So now we get that the expected value of x given y, or the expected value of the father’s height given the son’s height is:

A different regression line . In summary, it’s important to remember that the regression line comes from computing expectations, and these give you two different lines, depending on if you compute the expectation of y given x or x given y.

1. Linear Models
   1. Introduction to Linear Models
      1. Confounding: Are BBs More Predictive?

In a previous video, we found that the slope of the regression line for predicting runs form base on balls was 0.735. So does this mean that if we go and hire low salary players with many bases on balls that increases the number of walks per game by 2 for our team? Our team will score 1.47 more runs per game? We are again reminded that association is not causation. The data does provide strong evidence that a team with 2 more bases on balls per game than the average team scores 1.47 more runs per game, but this does not mean that bases on balls are the cause. If we compute the regression line slope for singles, we get 0.449, a lower value. Note that a single gets you to first base just like a base on balls. Those that know a little bit more about baseball will tell you that with a single, runners that are one base have a better chance of scoring than with a base on balls. So how can base on balls be more predictive of runs? The reason this happens is because of confounding. Note the correlation between homeruns, bases on balls, and singles:

*Teams%>%filter(yearID %in% 1961:2001)%>%*

*+ mutate(Singles=(H-HR-X2B-X3B)/G,BB=BB/G,HR=HR/G)%>%*

*+ summarize(cor(BB,HR),cor(Singles,HR),cor(BB,Singles))*

*cor(BB, HR) cor(Singles, HR) cor(BB, Singles)*

*1 0.4039313 -0.1737435 -0.05603822*

We see that the correlation between bases on balls and homeruns is quite high compared to the other two pairs. It turns out that pitchers, afraid of homeruns, will sometimes avoid throwing strikes to homerun hitters. As a result, homerun hitters tend to have more bases on balls. Thus, a team with many homeruns will also have more bases on balls than average, and as a result, it may appear that bases on balls cause runs. But it is actually the homeruns that causes the runs. In this case we say that bases on balls are confounded with homeruns. But could it be that bases on balls still help? To find out, we somehow have to adjust for the homerun effect. Regression can help with this.

* + 1. Stratification and Multivariate Regression

To try to determine if bases on balls is still useful for creating runs, a first approach is to keep home runs fixed at a certain value and then examine the relationship between runs and bases on balls. As we did when we stratified fathers by rounding to the closest inch, here, we can stratify home runs per game to the closest 10th, we filtered our strata with few points, We use this code to generate an informative data set:

*dat<-Teams%>%filter(yearID %in% 1961:2001)%>%*

*+ mutate(HR\_strata=round(HR/G,1),*

*+ BB\_per\_game=BB/G,*

*+ R\_per\_game=R/G)%>%*

*+ filter(HR\_strata>=0.4 & HR\_strata<=1.2)*

And then, we can make a scatterplot of runs versus bases on balls:

*dat%>%*

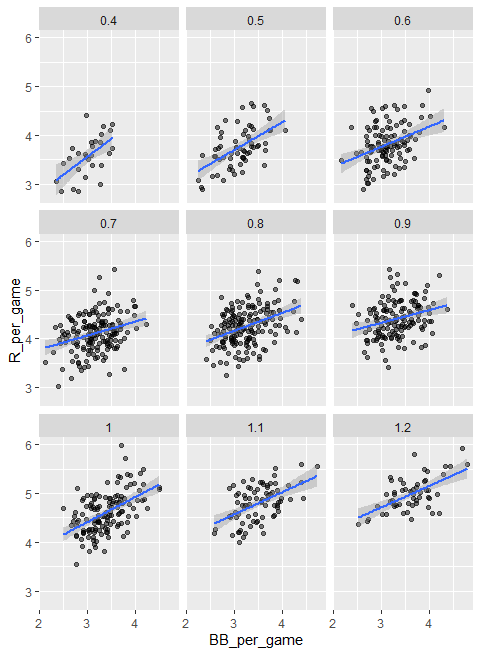
*+ ggplot(aes(BB\_per\_game,R\_per\_game))+*

*+ geom\_point(alpha=0.5)+*

*+ geom\_smooth(method="lm")+*

*+ facet\_wrap(~HR\_strata)*

1. Scatterplot of Runs vs Bases on balls stratified by Home runs



Remember that the regression slope for predicting runs with bases on balls when we ignore home runs was 0.735. But once we stratify by home runs, these slopes are substantially reduced. We can actually see what the slopes are by using this code:

*dat%>%*

*+ group\_by(HR\_strata)%>%*

*+ summarize(slope=cor(BB\_per\_game,R\_per\_game)\*sd(R\_per\_game)/sd(BB\_per\_game))*

*# A tibble: 9 x 2*

*HR\_strata slope*

*<dbl> <dbl>*

*1 0.4 0.734*

*2 0.5 0.566*

*3 0.6 0.412*

*4 0.7 0.285*

*5 0.8 0.365*

*6 0.9 0.261*

*7 1 0.511*

*8 1.1 0.454*

*9 1.2 0.440*

These values are closer to the slope we obtained from singles, which is 0.449. Which is more consistent with our intuition. Since both singles and bases on ball get us to first base, they should have about the same predictive power. Now, although our understanding of the application, our understanding of baseball, tell us that home runs cause bases on balls and not the other way around, we can still check if, after stratifying by base on balls, we still see a home run effect or if it goes down. We use the same code that we just used for bases on balls. But now we swap home runs for bases on balls to get this plot:

*dat2<-Teams%>%filter(yearID %in% 1961:2001)%>%*

*+ mutate(BB\_strata=round(BB/G,1),*

*+ HR\_per\_game=HR/G,*

*+ R\_per\_game=R/G)%>%*

*+ filter(BB\_strata>=2.8 & BB\_strata<=3.9)*

*> dat2%>%*

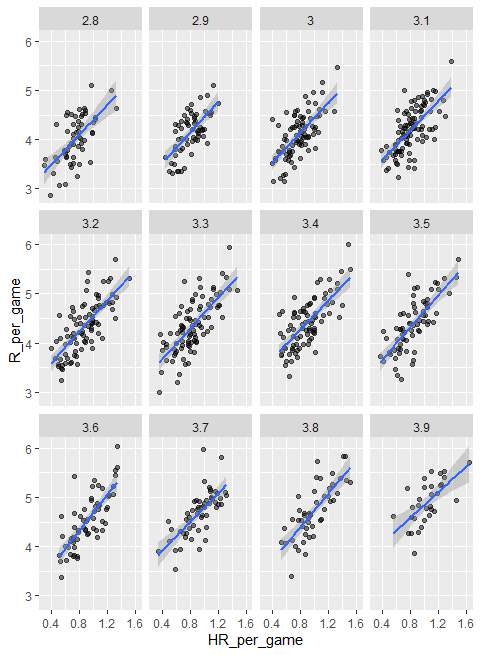
*+ ggplot(aes(HR\_per\_game,R\_per\_game))+*

*+ geom\_point(alpha=0.5)+*

*+ geom\_smooth(method="lm")+*

*+ facet\_wrap(~BB\_strata)*

1. Scatter plot of Home run vs Runs stratified by Base on Balls



In this cases the slopes are the following:

*dat2%>%*

*+ group\_by(BB\_strata)%>%*

*+ summarize(slope=cor(HR\_per\_game,R\_per\_game)\*sd(R\_per\_game)/sd(HR\_per\_game))*

*# A tibble: 12 x 2*

*BB\_strata slope*

*<dbl> <dbl>*

*1 2.8 1.52*

*2 2.9 1.57*

*3 3 1.52*

*4 3.1 1.49*

*5 3.2 1.58*

*6 3.3 1.56*

*7 3.4 1.48*

*8 3.5 1.63*

*9 3.6 1.83*

*10 3.7 1.45*

*11 3.8 1.70*

*12 3.9 1.30*

You can see they are all between 1.5 and 1.7. So they do not change that much from the original slope estimate, which was 1.84. Regardless, it seems that if we stratify by home runs, we have an approximately bivariate normal distribution for runs vs bases on balls. Similarly, if we stratify by bases on balls, we have an approximately normal bivariate distribution for runs versus home runs.

It is somewhat complex to be computing regressions lines for each strata. We’re essentially fitting this model that you can see in this equation with the slopes for x1 changing for different values of x2 and vice versa:

Note that if we take random variability into account, the estimated slopes by strata don’t appear to change that much. If these slopes are in fact the same, this implies that this function and are actually constant, which, in turn, implies that the expectation of runs condition on home runs and bases on balls can be written in this simpler model:

This model implies that if the number of home runs is fixed, we observed a linear relationship between runs and bases on balls and that the slope of that relationship does not depend on the number of home runs. Only the slope changes as the home runs increase. The same is true if we swap home runs and bases on balls. In this analysis, referred to as multivariate regression, we say that the bases on balls slope beta 1 is adjusted for the home run effect. If this model is correct, then confounding has been accounted for. But how do we estimate and from the data? For this we’ll learn about linear models and least squares estimates.

* + 1. Linear Models

Since Galton’s original development, regression has become one of the most widely used tools in data science. One reason for this has to do with the fact that regression permit us to find relationships between two variables while adjusting for others, as we have just shown for bases on balls and home runs. This has been particularly popular in fields where randomized experiments are hard to run, such as economics and epidemiology. When we’re not able to randomly assign each individual to a treatment or control group, confounding is particularly prevalent. For example, consider estimating the effect of fast foods on life expectancy using data collected from a random sample of people in some jurisdiction. Fast food consumers are more likely to be smokers, drinkers and have lower incomes. Therefore, a naïve regression model may lead to an overestimate of a negative health effect of fast foods. So how do we adjust for confounding in practice? We can use regression. We have described how, if data is bivariate normal, then the conditional expectation follow a regression line, that the conditional expectation as a line is not an extra assumption, but rather a result derived from the assumption, that they are approximately bivariate normal. However, in practice it is common to explicitly write down a model that describes the relationship between two or more variables using what is called a linear model. We know that linear here does not refer to lines exclusively, but rather to the fact that the conditional expectation is a linear combination of known quantities. Any combination that multiplies them by a constant and then adds them up with, perhaps a shift, for example:

Is a linear combination of x, y and z, so:

Is a linear combination of and . The simplest model is a constant . The second simplest is a line . For Galton’s data we would denote n observed father’s height with Then we model n son heights we are trying to predict with the following model:

Here, represent the father’s heights, which are fixed nor random, due to the conditioning. We’ve conditioned on these values, then the is the random son’s height that we want to predict. We further assume that the errors that are denoted with the Greek letter epsilon(), are independent from each other, have expected value of 0 and standard deviation, which is usually called sigma, does not depend on i .It`s the same for every individual. We know that the but to have a useful model for prediction, we need and . We estimate these from the data. Once we do, we can predict the son’s height, x. Note that if we further assume that the epsilons are normally distributed, then this model is exactly the same one we derived earlier for the bivariate normal distribution. A somewhat nuanced difference is that in the first approach, we assumed the data was a bivariate normal, and the linear model was derived not assumed. In practice, linear models are just assumed without necessarily assuming normality. The distribution of the epsilon is not specified. But nevertheless, if your data is bivariate normal, then you will need to have other ways of justifying the model. One reason linear models are popular is that they are interpretable. In the case of Galton’s data, we can interpret the data like this:

Due to inherited genes, the son’s height prediction grows by for each inches we increase the father’s height x

Because not all sons with father of height x are of equal height, we need the term epsilon, which explains the remaining variability. This remaining variability includes the mother’s genetic effect, environment factors and other biological randomness. Note that given how we wrote the model the intercept is not very interpretable as it is the predicted height of a son with a father with no height. Due to regression to the mean, the prediction will usually be a bit larger than 0, which is really not very interpretable. To make the intercept parameter more interpretable, we can rewrite the model slightly in the following way:

Here, we have changed to . We have centered our covariate . In this case, the intercept, would be the predicted height for the average father for the case where =

* 1. Least Squares Estimates
     1. Least Squares Estimates (LSE)

For linear models to be useful, we have to estimate the unknown parameters, the betas. The standard approach in science is to find the values that minimize the distance of the fitted model to the data. To quantify, this we use the least squares equation. For Galton’s data, we would write something like this:

This quantity is called the Residual Sum of Squares (RSS). Once we find the values that minimize the RSS, we call the values the Least Squares Estimate (LSE), and denote them, in this case, and . Let’s write the function that computed the RSS for any pair of values:

*rss <- function(beta0, beta1, data){*

*+ resid <- galton\_heights$son - (beta0+beta1\*galton\_heights$father)*

*+ return(sum(resid^2))*

*+ }*

So for any pair of values, we get an RSS. So, this is a 3-dimension plot with beta1 and beta2 as x and y and the RSS as z. To find the minimum you would have to look at this 3-dimension plot. Here, we’re just going to make a 2-dimension version by keeping beta0 fixed at 25. So it will be function of the RSS as a function of beta1. We can use this code to produce this plot:

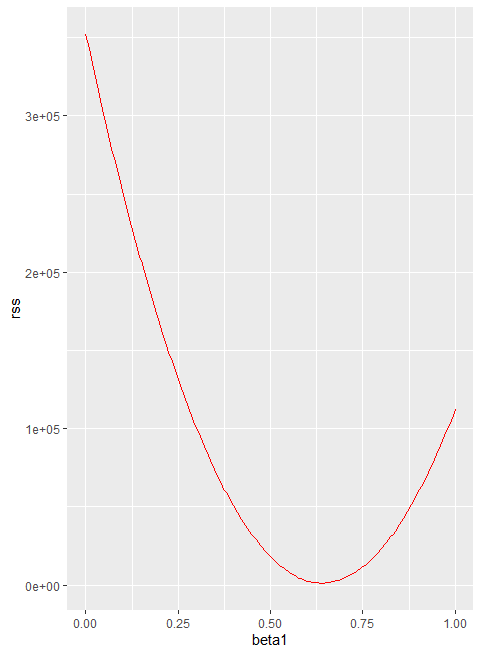
*beta1=seq(0,1,len=nrow(galton\_heights))*

*results<-data.frame(beta1=beta1,*

*+ rss=sapply(beta1,rss,beta0=25))*

*results%>%ggplot(aes(beta1,rss))+geom\_line()+geom\_line(aes(beta1,rss),col=2)*

1. Beta1 vs RSS



We can see a clear minimum for beta1 at around 0.65. So you could see how we would pick the least squares estimates. However, this minimum for beta1 when beta 0 is fixed at 25. But we don’t know if that’s the minimum for beta 0. We don’t know if (25,0.65) minimizes the equation across all pairs. We could use trial and error, but it’s really not going to work here. Instead we will use calculus. We’ll take the partial derivatives, set them equal to 0, and solve for beta1 and beta 0. Of course, if we have many parameters, these equations can get rather complex. But there are functions in R that do these calculations for us. We will learn these soon. To learn the mathematics behind this, you can consult the book on linear models.

* + 1. The lm Function

In R, we can obtain the least squares estimates using the lm function. To fit the following model where Yi is the son’s height and Xi is the father height, we would write the following piece of code:

*fit<-lm(son~father,data=galton\_heights)*

*fit*

*Call:*

*lm(formula = son ~ father, data = galton\_heights)*

*Coefficients:*

*(Intercept) father*

*38.7646 0.4411*

This give us the least squares estimates. The general way we use lm is by using the tilde character to let lm know which is the value we’re predicting that’s on the left side of the tilde, and which variables we’re using to predict, those will be on the right side of the tilde. The intercept is added automatically to the model. So you don’t have to include it when you write it. The object fit that we just computed includes more information about the least squares fit. We can use the function summary to extract more this information, like this:

*summary(fit)*

*Call:*

*lm(formula = son ~ father, data = galton\_heights)*

*Residuals:*

*Min 1Q Median 3Q Max*

*-9.4228 -1.7022 0.0333 1.5670 9.3567*

*Coefficients:*

*Estimate Std. Error t value*

*(Intercept) 38.76457 5.41093 7.164*

*father 0.44112 0.07825 5.637*

*Pr(>|t|)*

*(Intercept) 2.03e-11 \*\*\**

*father 6.72e-08 \*\*\**

*---*

*Signif. codes:*

*0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’*

*0.1 ‘ ’ 1*

*Residual standard error: 2.659 on 177 degrees of freedom*

*Multiple R-squared: 0.1522, Adjusted R-squared: 0.1474*

*F-statistic: 31.78 on 1 and 177 DF, p-value: 6.719e-08*

To understand some the information included in this summary, we need to remember that the LSE are random variables. Mathematical statistics give us some ideas of the distribution of these random variables

* + 1. LSE are Random Variables

The LSE are derived from the data, Y1 through Yn, which are random. This implies that our estimates are random variables. To see this, we can run a Monte Carlo simulation in which we assume that the son and father height data that we have defines an entire population, and we’re going to take random samples of size 50 and compute the regression slope for each one:

*lse<-replicate(B,{*

*+ sample\_n(galton\_heights,N,replace=TRUE) %>%*

*+ lm(son~father,data=.)%>%.$coef*

*+ })*

*lse<-data.frame(beta\_0=lse[1,],beta\_1=lse[2,])*

We can see the variability of the estimates by plotting their distribution:

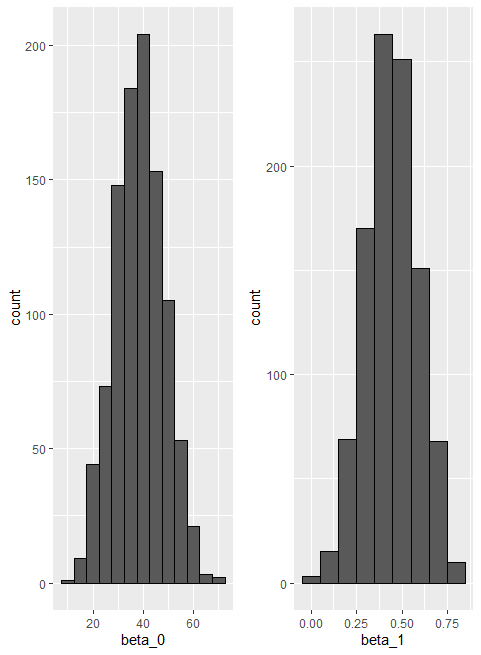
*library(gridExtra)*

*p1<-lse%>%ggplot(aes(beta\_0))+geom\_histogram(binwidth = 5,color="black")*

*p2<-lse%>%ggplot(aes(beta\_1))+geom\_histogram(binwidth = 0.1,color="black")*

*grid.arrange(p1,p2,ncol=2)*

1. Histogram of the estimated beta0’s and beta1’s



The reason these look normal is because the central limit theorem applies here as well. For large enough N, the least squares estimates will be approximately normal with expected value beta0 and beta1 respectively. The standard errors are a bit complicated to compute, but mathematical theory does allow us to compute them, and they are included in the summary provided by the lm function:

*sample\_n(galton\_heights,N,replace=TRUE)%>%*

*+ lm(son~father,data=.)%>% summary*

*Call:*

*lm(formula = son ~ father, data = .)*

*Residuals:*

*Min 1Q Median 3Q Max*

*-8.575 -2.200 0.697 2.050 5.223*

*Coefficients:*

*Estimate Std. Error t value*

*(Intercept) 47.8483 11.8173 4.049*

*father 0.2982 0.1712 1.742*

*Pr(>|t|)*

*(Intercept) 0.000187 \*\*\**

*father 0.087855 .*

*---*

*Signif. codes:*

*0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’*

*0.1 ‘ ’ 1*

*Residual standard error: 3.076 on 48 degrees of freedom*

*Multiple R-squared: 0.05948, Adjusted R-squared: 0.03989*

*F-statistic: 3.036 on 1 and 48 DF, p-value: 0.08785*

You can see the estimated standard errors for one our simulated data sets, they are in the second column in the coefficients table. You can see that the standard errors estimate reported by the summary function are closed to the standard errors we obtain from our Monte Carlo simulation. The summary function also reports t-statistics, this is the t value column, and p-value, Pr bigger than absolute value of t column. The t-statistic is not actually based on the central limit theorem, but rather on the assumption that the epsilons follow a normal distribution. Under this assumption, mathematical theory tell us that the LSE divided by their standard error, which we can see here and here:

Follow a t distribution with N-p degree of freedom, with p the number of parameters in our model, which in this case is 2. The 2p values are testing the null hypothesis that beta 0 is 0 and beta 1 is 0. Note that as we described previously, for large enough N, the central limit works, and the t distribution becomes almost the same as a normal distribution. So if either you assume the errors are normal and use the t distribution or if you assume that N is large enough to use the central limit theorem, you can construct confidence intervals for your parameters. We know here that although we will not show examples in this video, hypothesis testing for regression models is very commonly used in, for example, epidemiology and economics, to male statements such as the effect of A and B was statistically significant after adjusting for X, Y and Z. But it’s very important to note that several assumptions, we just described some of them, have to hold for these statements to hold.

* + 1. Advanced Note on LSE

Although interpretation is not straight-forward, it is also useful to know that the LSE can be strongly correlated, which can be seen using this code:

lse %>% summarize(cor(beta\_0, beta\_1))

However, the correlation depends on how the predictors are defined or transformed.Here we standardize the father heights, which changes xi to xi− x¯:

*B <- 1000*

*N <- 50*

*lse <- replicate(B, {*

*sample\_n(galton\_heights, N, replace = TRUE) %>%*

*mutate(father = father - mean(father)) %>%*

*lm(son ~ father, data = .) %>% .$coef*

*})*

Observe what happens to the correlation in this case:

cor(lse[1,], lse[2,])

* + 1. Predicted Variables are Random Variables

Once we fit our model, we can obtain predictions of y by plugging the estimates into the regression model. For example, if the father’s height is x, then our prediction for y, which we’ll denote with a hat on top the y, for the son’s height will be the following:

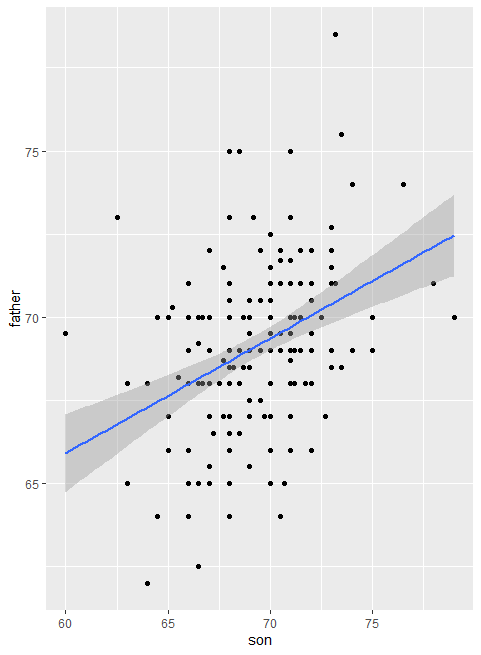
We’re just plugging in the estimated betas into the equation. If we plot y hat versus x, we’ll see the regression line. Note that the prediction y hat is also a random variable, and mathematical theory tell us what the standard error are. If we assume the errors are normal or have a large enough sample size to use the Central Limit Theorem, we can construct confidence intervals for our predictions as well. In fact, the ggplot layer geom\_smooth, when we set method equals to lm, plots confidence intervals around the predicted y hat:

galton\_heights%>%ggplot(aes(son,father))+

+ geom\_point()+

+ geom\_smooth(method = "lm")

1. Predicted son heights based on their father height with confidence intervals



You can see the regression line, in blue, and their confidence interval, the band in grey around it. The R function predict takes an lm object as input and returns these predictions:

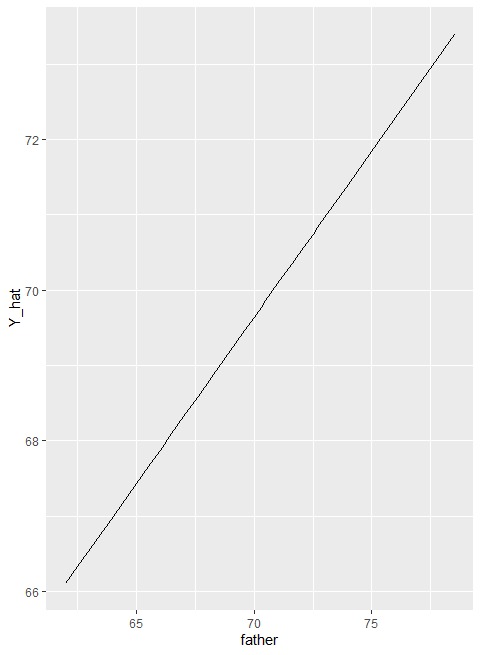
*galton\_heights%>%*

*+ mutate(Y\_hat=predict(lm(son~father,data = .)))%>%*

*+ ggplot(aes(father,Y\_hat))+*

*+ geom\_line()*

1. Y\_hat versus father height



If requested the standard errors and other information from which we can construct confidence intervals can be obtained from the predict function. You can see it by running this code:

*fit<-galton\_heights%>%lm(son~father,data=.)*

*Y\_hat<-predict(fit,se.fit = TRUE)*

*names(Y\_hat)*

*[1] "fit" "se.fit"*

*[3] "df" "residual.scale"*

* 1. Tibbles, do, and broom
     1. Advanced dplyr: Tibbles

Let’s go back to baseball. In a previous example, we estimated the regression lines to predict runs from bases and balls in different home run strata. We first constructed a data frame similar to this:

*dat<-Teams%>%filter(yearID %in% 1961:2001)%>%*

*+ mutate(HR=round(HR/G,1),*

*+ BB=BB/G,*

*+ R=R/G)%>%*

*+ select(HR,BB,R)%>%*

*+ filter(HR>=0.4 & HR<=1.2)*

Then, to compute the regression line in each strata, since we didn’t know the lm function back then, we used the formula directly like this:

*dat%>%*

*+ group\_by(HR)%>%*

*+ summarize(slope=cor(BB,R)\*sd(R)/sd(BB))*

*# A tibble: 9 x 2*

*HR slope*

*<dbl> <dbl>*

*1 0.4 0.734*

*2 0.5 0.566*

*3 0.6 0.412*

*4 0.7 0.285*

*5 0.8 0.365*

*6 0.9 0.261*

*7 1 0.511*

*8 1.1 0.454*

*9 1.2 0.440*

We argued that the slopes are similar and that the differences were perhaps due to random variation. To provide a more rigorous defense of the slopes being the same, which is what led to our multivariate regression model, we could compute confidence intervals for each slope. We have not learned the formula for this, but the lm function provides enough information to construct them.

First, note that if we try to use the lm function to get the estimated slope like this:

*dat%>%*

*+ group\_by(HR)%>%*

*+ lm(R~BB,data=.)%>%*

*+ .$coeff*

*(Intercept) BB*

*2.198 0.638*

we don’t get what we want. The lm function ignored the group\_by. This is expected, because lm is not part of the tidyverse and does not know how to handle the outcome of group\_by which is a group tibble. We’re going to describe tiblles in some details now. When summarize receives the output of group\_by, it somehow knows which rows of the table go with which groups. One question you may ask is where is this information stored in the data frame? Let’s write some code to see the output of a group\_by call:

*dat%>%group\_by(HR)%>%head()*

*# A tibble: 6 x 3*

*# Groups: HR [5]*

*HR BB R*

*<dbl> <dbl> <dbl>*

*1 0.9 3.56 4.24*

*2 0.7 3.97 4.47*

*3 0.8 3.37 4.69*

*4 1.1 3.46 4.42*

*5 1 2.75 4.61*

*6 0.9 3.06 4.58*

Note that there are no columns with the information needed to define the groups. But if you look closely at the output, you notice the line “A tibble: 6 x 3”. We can learn the class of the return object using this line of code:

*dat%>%group\_by(HR)%>%class()*

*[1] "grouped\_df" "tbl\_df" "tbl"*

*[4] "data.frame*

And we see that the class is a “tbl” this is pronounced “tibble”. It is also a tbl\_df . This is equivalent to tibble, the tibble is a special kind of data frame. We have seen them before, because tidyverse functions such as group\_by and also summarize always return this type of data frame. The group\_by function returns a special kind of tibble the gouped tibble. We will say more about the grouped tibbles later, note that the manipulation verbs, select, filter, mutate and arrange, don’t necessarily returns tibbles. They preserve the class of the input. If they receive a tibble, they return a tibble. But tibbles are the default data frame for the tidyverse. Tibbles are very similar to data frames. You can think of them as modern versions of data frames. Next, we’re going to describe briefly three important differences.

* + 1. Tibbles: Differences from Data Frames

In this video, we’re going to describe some of the main differences between tibbles and data frames. First, the print method for tibble is much more readable than that of a data frame. To see this, type teams on your console after loading the Baseball Lahman data set. You will see a very, very long list of columns and rows. It’s barely readable. Team is a data frame with many rows and columns. That’s why you see that. Nevertheless, the output just shows everything wraps around and is hard to read. It is so bad that we don’t even print it here. We’ll let you print it on your own screen. Now if your convert this data frame to a tibble data frame, the output is much more readable. Here’s an example:

*as.tibble(Teams)*

*# A tibble: 2,895 x 48*

*yearID lgID teamID franchID divID Rank*

*<int> <fct> <fct> <fct> <chr> <int>*

*1 1871 NA BS1 BNA NA 3*

*2 1871 NA CH1 CNA NA 2*

*3 1871 NA CL1 CFC NA 8*

*4 1871 NA FW1 KEK NA 7*

*5 1871 NA NY2 NNA NA 5*

*6 1871 NA PH1 PNA NA 1*

*7 1871 NA RC1 ROK NA 9*

*8 1871 NA TRO TRO NA 6*

*9 1871 NA WS3 OLY NA 4*

*10 1872 NA BL1 BLC NA 2*

*# ... with 2,885 more rows, and 42 more*

*# variables: G <int>, Ghome <int>,*

*# W <int>, L <int>, DivWin <chr>,*

*# WCWin <chr>, LgWin <chr>, WSWin <chr>,*

*# R <int>, AB <int>, H <int>, X2B <int>,*

*# X3B <int>, HR <int>, BB <dbl>,*

*# SO <int>, SB <dbl>, CS <dbl>,*

*# HBP <dbl>, SF <int>, RA <int>,*

*# ER <int>, ERA <dbl>, CG <int>,*

*# SHO <int>, SV <int>, IPouts <int>,*

*# HA <int>, HRA <int>, BBA <int>,*

*# SOA <int>, E <int>, DP <int>, FP <dbl>,*

*# name <chr>, park <chr>,*

*# attendance <int>, BPF <int>, PPF <int>,*

*# teamIDBR <chr>, team*

That’s the first difference between tibbles and data frames. A second one is that if you subset the columns of a data frame, you may get back an object that is not a data frame. Here’s an example:

*class(Teams[,20])*

*[1] "integer"*

If we subset the 20th column, we get back an integer, which is not a data frame. With tibbles, this is not the case, Here’s an example:

*class(as\_tibble(Teams)[,20])*

*[1] "tbl\_df" "tbl" "data.frame"*

If we subset a table, we get back a table. This is useful in the tidyverse since functions require data frames as input. Now whenever you want to access the original vector that define a column in a table, for this, you actually have to use the accessor dollar sign. Here’s an example:

*class(as\_tibble(Teams)$HR)*

*[1] "integer"*

A related feature to this is that tibbles will give you a warning if you try to access a column that does not exist. That’s not the case for data frames. For example:

*Teams$hr*

*NULL*

*as\_tibble(Teams)$hr*

*NULL*

*Warning message:*

*Unknown or uninitialised column: 'hr'*

A third difference is that while columns of a data frame need to be a vector of number strings or Boolean, tibbles can have more complex objects, such as lists or functions. Also note that we can create tibbles with the tibble function. For example:

*tibble(id=c(1,2,3),func=c(mean,median,sd))*

*# A tibble: 3 x 2*

*id func*

*<dbl> <list>*

*1 1 <fn>*

*2 2 <fn>*

*3 3 <fn>*

We’re creating a column that has functions in it. Finally the last difference we describe is that tibbles can be grouped. The function group by returns a special kind of tibble, a grouped tibble. This class stores information that lets you know which rows are in which groups. The tidyverse functions, in particular the summarize functions, are aware of the group information. In the example we showed, we saw that the lm function, which is not part of the tidyverse, does not know how to deal with group tibbles. The object is basically converted to regular data frame, and then the function runs ignoring the groups. This is why we only get one pair of estimates as we see here:

*dat%>%*

*+ group\_by(HR)%>%*

*+ lm(R~BB,data=.)*

*Call:*

*lm(formula = R ~ BB, data = .)*

*Coefficients:*

*(Intercept) BB*

*2.198 0.638*

To make these non-tidyverse function better integrate with a tidyverse, we will learn a new function, the function **do.**

* + 1. Do

The tidyverse functions know how to interpret group tibbles. Furthermore, to facilitate stringing commands through the pipe, tidyverse function consistently return data frames. Since this assures that the output of one is accepted as the input of another. But most our function do not recognize group tibbles, nor do they return data frames. The lm function is an example. The do() function serves as bridge between our functions, such as lm() and the tidyvers. The do function understands group tibbles and always returns a data frame. So let’s try to use the do() function to fit a regression line to each home run strata. We would do it like this:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(fit=lm(R~BB,data=.))*

*Source: local data frame [9 x 2]*

*Groups: <by row>*

*# A tibble: 9 x 2*

*HR fit*

*\* <dbl> <list>*

*1 0.4 <lm>*

*2 0.5 <lm>*

*3 0.6 <lm>*

*4 0.7 <lm>*

*5 0.8 <lm>*

*6 0.9 <lm>*

*7 1 <lm>*

*8 1.1 <lm>*

*9 1.2 <lm>*

Notice that we did in fact fit a regression line to each strata. But the do() function would create a data frame with the first column being the strata value, and a column named fit, we chose the name, that column will contain the result of the lm() call. Therefore, the return table has column with lm() objects in the cells, which is not very useful. Also note that if we do not name a column, then do() will return the actual output of lm(), not a data frame, and this will result in an error since do() is expecting a data frame as output:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(lm(R~BB,data=.))*

*Error: Results 1, 2, 3, 4, 5, ... must be data frames, not lm*

For a useful data frame to be constructed the output of the function inside do(), must be a data frame as well. We could build a function that returns only what you want in the form of a data frame, we could write this for example:

*get\_slope<-function(data){*

*+ fit<-lm(R~BB,data=data)*

*+ data.frame(slope=fit$coefficients[2],*

*+ se=summary(fit)$coefficient[2,2])*

*+ }*

And then we use the do function without naming the output, since we are already getting a data frame:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(get\_slope(.))*

*# A tibble: 9 x 3*

*# Groups: HR [9]*

*HR slope se*

*<dbl> <dbl> <dbl>*

*1 0.4 0.734 0.208*

*2 0.5 0.566 0.110*

*3 0.6 0.412 0.0974*

*4 0.7 0.285 0.0705*

*5 0.8 0.365 0.0653*

*6 0.9 0.261 0.0754*

*7 1 0.511 0.0749*

*8 1.1 0.454 0.0855*

*9 1.2 0.440 0.0801*

We get the slope for each strata and the standard error for that slope. If we name the output then we get a column containing the data frame. So if we write this:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(slope=get\_slope(.))*

*Source: local data frame [9 x 2]*

*Groups: <by row>*

*# A tibble: 9 x 2*

*HR slope*

*\* <dbl> <list>*

*1 0.4 <df[,2] [1 x 2]>*

*2 0.5 <df[,2] [1 x 2]>*

*3 0.6 <df[,2] [1 x 2]>*

*4 0.7 <df[,2] [1 x 2]>*

*5 0.8 <df[,2] [1 x 2]>*

*6 0.9 <df[,2] [1 x 2]>*

*7 1 <df[,2] [1 x 2]>*

*8 1.1 <df[,2] [1 x 2]>*

*9 1.2 <df[,2] [1 x 2]>*

We get one of these complex tibbles with a column having a data frame in each cell. Which is again, not very useful. All right. Now we’re going to cover one last feature of do(). If the data frame being returned has more than one row, these will be concatenated appropriately. Here’s an example in which return both estimated parameters. The slope and intercept. We write this piece of code:

*get\_lse<-function(data){*

*+ fit<-lm(R~BB,data=data)*

*+ data.frame(term=names(fit$coefficients),*

*+ slope=fit$coefficients,*

*+ se=summary(fit)$coefficient[,2])*

*+ }*

*> dat%>%group\_by(HR)%>%*

*+ do(get\_lse(.))*

*# A tibble: 18 x 4*

*# Groups: HR [9]*

*HR term slope se*

*<dbl> <fct> <dbl> <dbl>*

*1 0.4 (Intercept) 1.36 0.631*

*2 0.4 BB 0.734 0.208*

*3 0.5 (Intercept) 2.01 0.344*

*4 0.5 BB 0.566 0.110*

*5 0.6 (Intercept) 2.53 0.305*

*6 0.6 BB 0.412 0.0974*

*7 0.7 (Intercept) 3.21 0.225*

*8 0.7 BB 0.285 0.0705*

*9 0.8 (Intercept) 3.07 0.213*

*10 0.8 BB 0.365 0.0653*

*11 0.9 (Intercept) 3.54 0.252*

*12 0.9 BB 0.261 0.0754*

*13 1 (Intercept) 2.88 0.255*

*14 1 BB 0.511 0.0749*

*15 1.1 (Intercept) 3.21 0.300*

*16 1.1 BB 0.454 0.0855*

*17 1.2 (Intercept) 3.40 0.291*

*18 1.2 BB 0.440 0.0801*

We get a very useful tibble giving us the estimates of the slope and intercept as well as the standard errors. Now, if you think this is all a bit too complicated, you’re not alone. To simplify things, we’re going to introduce the broom package, which was designed to facilitate the use of model fitting functions such as lm() with the tidyverse

* + 1. Broom

The original task we ask for was to provide an estimate and a confidence interval for the slope estimates of each strata. The **broom** package will make this quite easy. Broom has three main functions all of which extract information from the object returned by the function lm(), and return it in a tidyverse friendly data frame. These functions are **tidy, glance and augment.** The tidy function returns estimates and related information as a data frame. These functions are tidy, glance and augment. The tidy function returns estimates and related information as a data frame. Here’s an example:

*library(broom)*

*> fit<-lm(R~BB,data=dat)*

*> tidy(fit)*

*A tibble: 2 x 5*

*term estimate std.error statistic p.value*

*<chr> <dbl> <dbl> <dbl> <dbl>*

*1 (Intercept) 2.20 0.113 19.4 1.12e-70*

*2 BB 0.638 0.0344 18.5 1.35e-65*

We can add other important summaries, such as confidence intervals, using arguments like this:

*tidy(fit,conf.int=TRUE)*

*# A tibble: 2 x 7*

*term estimate std.error statistic p.value conf.low conf.high*

*<chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>*

*1 (Intercept) 2.20 0.113 19.4 1.12e-70 1.98 2.42*

*2 BB 0.638 0.0344 18.5 1.35e-65 0.570 0.705*

Because the outcome is a data frame, we can immediately use it with do to string together the commands that produce the table we are after.

So this piece of code will generate what we wanted to see:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(tidy(lm(R~BB,data=.),conf.int=TRUE))*

*# A tibble: 18 x 8*

*# Groups: HR [9]*

*HR term estimate std.error statistic p.value conf.low conf.high*

*<dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>*

*1 0.4 (Intercept) 1.36 0.631 2.16 4.05e- 2 0.0631 2.66*

*2 0.4 BB 0.734 0.208 3.54 1.54e- 3 0.308 1.16*

*3 0.5 (Intercept) 2.01 0.344 5.84 2.07e- 7 1.32 2.69*

*4 0.5 BB 0.566 0.110 5.14 3.02e- 6 0.346 0.786*

*5 0.6 (Intercept) 2.53 0.305 8.32 2.43e-13 1.93 3.14*

*6 0.6 BB 0.412 0.0974 4.23 4.80e- 5 0.219 0.605*

*7 0.7 (Intercept) 3.21 0.225 14.3 1.49e-30 2.76 3.65*

*8 0.7 BB 0.285 0.0705 4.05 7.93e- 5 0.146 0.425*

*9 0.8 (Intercept) 3.07 0.213 14.4 5.40e-31 2.65 3.49*

*10 0.8 BB 0.365 0.0653 5.59 9.13e- 8 0.236 0.494*

*11 0.9 (Intercept) 3.54 0.252 14.1 1.37e-28 3.04 4.04*

*12 0.9 BB 0.261 0.0754 3.46 7.12e- 4 0.112 0.410*

*13 1 (Intercept) 2.88 0.255 11.3 5.32e-21 2.37 3.39*

*14 1 BB 0.511 0.0749 6.83 3.00e-10 0.363 0.660*

*15 1.1 (Intercept) 3.21 0.300 10.7 6.46e-17 2.61 3.81*

*16 1.1 BB 0.454 0.0855 5.31 1.03e- 6 0.284 0.624*

*17 1.2 (Intercept) 3.40 0.291 11.7 2.33e-16 2.81 3.98*

*18 1.2 BB 0.440 0.0801 5.50 1.07e- 6 0.280 0.601*

Because a data frame is returned, we can filter and select the rows and columns we want. So this simple piece of code:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(tidy(lm(R~BB,data=.),conf.int=TRUE))%>%*

*+ filter(term=="BB")%>%*

*+ select(HR,estimate,conf.low,conf.high)*

*# A tibble: 9 x 4*

*# Groups: HR [9]*

*HR estimate conf.low conf.high*

*<dbl> <dbl> <dbl> <dbl>*

*1 0.4 0.734 0.308 1.16*

*2 0.5 0.566 0.346 0.786*

*3 0.6 0.412 0.219 0.605*

*4 0.7 0.285 0.146 0.425*

*5 0.8 0.365 0.236 0.494*

*6 0.9 0.261 0.112 0.410*

*7 1 0.511 0.363 0.660*

*8 1.1 0.454 0.284 0.624*

*9 1.2 0.440 0.280 0.601*

gives us exactly the table we asked for. We have filtered away the intercept rows, and only show the columns we care about, the estimate and the confidence intervals. Furthermore, a table like this makes visualization with ggplot quite easy, this piece of code produces this nice plot:

*dat%>%*

*+ group\_by(HR)%>%*

*+ do(tidy(lm(R~BB,data=.),conf.int=TRUE))%>%*

*+ filter(term=="BB")%>%*

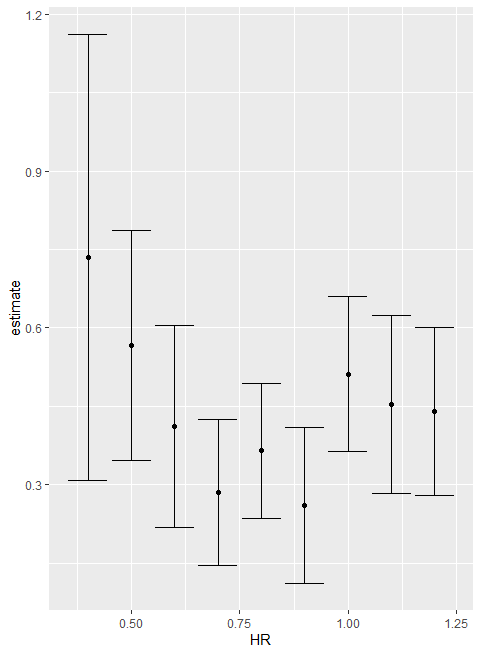
*+ select(HR,estimate,conf.low,conf.high)%>%*

*+ ggplot(aes(HR,y=estimate,ymin=conf.low,ymax=conf.high))+*

*+ geom\_errorbar()+*

*+ geom\_point()*

1. Home runs prediction with confidence interval by strata



Which provides very useful information. Now we return to discussing our original task of determining if slopes change. The plot we just made using do and broom shows that the confidence intervals overlap, which provides a nice visual confirmation that our assumption that the slopes do not change with home run strata, is relatively safe. Earlier we mentioned two other functions from the broom package, glance and augment. Glance relate to model specific outcomes and Augment relate to observation specific outcomes. Here we can see the model fit summary the glance returns:

*glance(fit)*

*# A tibble: 1 x 11*

*r.squared adj.r.squared sigma statistic p.value df logLik AIC BIC deviance*

*<dbl> <dbl> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>*

*1 0.266 0.265 0.454 343. 1.35e-65 2 -596. 1199. 1214. 195.*

You can learn more about these summaries in any regression textbook.

* 1. Regression and Baseball
     1. Building a Better Offensive Metric for Baseball

In trying to answer how well bases on balls predict runs, data exploration led us to this model:

Here, the data is approximately normal and conditional distributions were also normal. Thus, we’re just justified to pose a linear model like this:

With Yi, the runs per game. x1, walks per game and x2 home runs per game. To use lm here we need to let it know that we have two predictor variables. So we use the + symbol as follows:

*fit<-Teams%>%*

*+ filter(yearID %in% 1961:2001)%>%*

*+ mutate(BB=BB/G,HR=HR/G,R=R/G)%>%*

*+ lm(R~BB+HR,data=.)*

Now, we can use the tidy function to see the nice summary:

tidy(fit,conf.int = TRUE)

# A tibble: 3 x 7

term estimate std.error statistic p.value conf.low conf.high

<chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 (Intercept) 1.74 0.0824 21.2 7.62e- 83 1.58 1.91

2 BB 0.387 0.0270 14.3 1.20e- 42 0.334 0.440

3 HR 1.56 0.0490 31.9 1.78e-155 1.47 1.66

When we fit the model with only one variable without the adjustment, the estimated slopes were 0.735 and 1.844 for bases on ball and home runs, respectively. But note that when we fit the multivariate model, both these slopes go down with the base on balls effect decreasing much more. Now, if we want to construct a metric to pick players, we need to consider singles, doubles, and triples as well. Can we build a model that predicts runs bases on all these outcomes? Now, we’re going to take somewhat of a leap of faith and assume that these five variables are jointly normal. This means that if we pick any one of them and hold the other four fixed, the relationship with the outcome, in this case, run per game is linear, and the slopes for this relationship do not depend on the other four values that were held constant.

If this is true, if this model holds true, then a linear model for our data is the following:

With x1, x2, x3, x4, x5 representing bases on balls per game, singles per game, doubles per game, triples per game and home runs per game, respectively. Using lm, we can quickly find the least squared errors for the parameters using this relatively simple piece of code:

*fit<-Teams%>%*

*+ filter(yearID %in% 1961:2001)%>%*

*+ mutate(BB=BB/G,*

*+ singles=(H-X2B-X3B-HR)/G,*

*+ doubles=X2B/G,*

*+ triples=X3B/G,*

*+ HR=HR/G,*

*+ R=R/G)%>%*

*+ lm(R~BB+singles+doubles+triples+HR,data=.)*

We can again use the tidy function to see the coefficients, the standard errors, and confidence intervals:

*coefs<-tidy(fit,conf.int = TRUE)*

*> coefs*

*# A tibble: 6 x 7*

*term estimate std.error statistic p.value conf.low conf.high*

*<chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>*

*1 (Intercept) -2.77 0.0862 -32.1 4.76e-157 -2.94 -2.60*

*2 BB 0.371 0.0117 31.6 1.87e-153 0.348 0.394*

*3 singles 0.519 0.0127 40.8 8.67e-217 0.494 0.544*

*4 doubles 0.771 0.0226 34.1 8.44e-171 0.727 0.816*

*5 triples 1.24 0.0768 16.1 2.12e- 52 1.09 1.39*

*6 HR 1.44 0.0243 59.3 0. 1.40 1.49*

To see how well our metric actually predict runs, we can predict the number of runs for each team in 2002 using the function predict() to make the plot. Note that we did not use the 2002 year to create this metric. We used data from years previous to 2002, here is the plot :

*Teams%>%*

*+ filter(yearID %in% 2002)%>%*

*+ mutate(BB=BB/G,*

*+ singles=(H-X2B-X3B-HR)/G,*

*+ doubles=X2B/G,*

*+ triples=X3B/G,*

*+ HR=HR/G,*

*+ R=R/G)%>%*

*+ mutate(R\_hat=predict(fit,newdata = .))%>%*

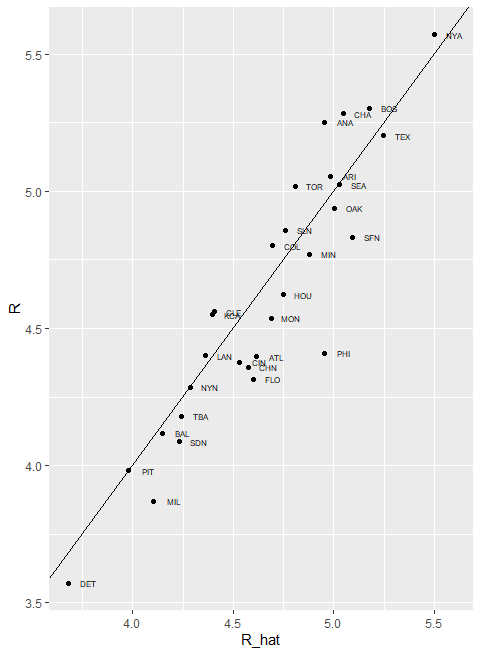
*+ ggplot(aes(R\_hat,R,label=teamID))+*

*+ geom\_point()+*

*+ geom\_text(nudge\_x = 0.1,cex=2)+*

*+ geom\_abline()*

1. Prediction of 2002 runs



Our model does quite a good job, as demonstrated by the fact that points from the observed versus predicted plot fall close to the identity line. So instead of using batting average or just the number of home runs as a measure for picking players, we can use our fitted model to form a more informative metric that relates more directly to run production. Specifically, to define a metric for player A, we imagine a team made up of players just like player A and use our fitted regression model to predict how many runs this team would produce. The formula would look like this:

We`re basically sticking in the estimated coefficients into the regression formula. However, to define a player specific metric, we have a bit more work to do. Our challenge here is that we have derived the metrics for teams based on team level summary statistics. For example, the home run value that is entered into the equation is home runs per game for the entire team. if you compute the home runs per game for a player, it will be much lower. As the total is accumulated by nine batters, not just one. Furthermore, if a player only plays part of the game and gets less opportunity than average, it’s still considered a game played. So, this mean that their rates will be lower than they should be.

For players, a rate that takes into account opportunities is a per-plate-appearance rate. To make the per-game team rate comparable to the per-plate appearance player rate, we compute the average number of team plate appearances per game using this piece of code:

*pa\_per\_game<-Batting%>%filter(yearID==2002)%>%*

*+ group\_by(teamID)%>%*

*+ summarize(pa\_per\_game=sum(AB+BB)/max(G))%>%*

*+ .$pa\_per\_game%>%*

*+ mean*

Now, we’re ready to use our metric. We’re going to compute the per-plate-appearance rates for players available in 2002. But we’re going to use data from 1999-2001. Because remember, we are picking players in 2002. We don’t know what has happened yet. To avoid small sample artifacts, we’re going to filter players with few plate appearances. Here is the calculation of what we want to do in one long line of code using tidyverse:

*players<-Batting%>%filter(yearID %in% 1999:2001)%>%*

*+ group\_by(playerID)%>%*

*+ mutate(PA=BB+AB)%>%*

*+ summarize(G=sum(PA)/pa\_per\_game,*

*+ BB=sum(BB)/G,*

*+ singles=sum(H-X2B-X3B-HR)/G,*

*+ doubles=sum(X2B)/G,*

*+ triples=sum(X3B)/G,*

*+ HR=sum(HR)/G,*

*+ AVG=sum(H)/sum(AB),*

*+ PA=sum(PA))%>%*

*+ filter(PA>=300)%>%*

*+ select(-G)%>%*

*+ mutate(R\_hat=predict(fit,newdata = .))*

So we fit our model, and we have player-specific metrics. The player-specific predicted runs computer here can be interpreted as the number of runs we would predict a team to score if this team was made up of just that player, if that player batted every single time. The distribution shows that there’s wide variability across players, as we can see here. To actually build the teams, we will need to know the players’ salaries, since we have a limited budget. Remember, we are pretending to be the Oakland A’s in 2002 with only a $40 million budget. We also need to know the players’ position. Because, we’re going to need one shortstop, one second baseman, one third baseman etc… For this we’re going to have to do a little bit of data wrangling to combine information that is contained in different tables from the Lehman library.

We start by adding the 2002 salaries for each player using this code:

*players<-Salaries%>%*

*+ filter(yearID==2002)%>%*

*+ select(playerID,salary)%>%*

*+ right\_join(players,by="playerID")*

Next, we’re going to add the defensive position. This is a little bit complicated, because players play more than one position each year. So here. We’re going to pick the one position most played by each player using the top\_n function and to make sure that we only pick one position in the case of ties, we’re going to take the first row if there is a tie. We also remove the OF position. Because this stands for outfielder, which is a generalization of three position: left field, center field, right field. We also remove pitchers, as they don’t bat in the league that the Athletics play. Here is the code:

*playersM<-Fielding%>%filter(yearID==2002)%>%*

*+ filter(!POS %in% c("OF","P"))%>%*

*+ group\_by(playerID)%>%*

*+ top\_n(1,G)%>%*

*+ filter(row\_number(G)==1)%>%*

*+ ungroup()%>%*

*+ select(playerID,POS)%>%*

*+ right\_join(players,by="playerID")%>%*

*+ filter(!is.na(POS)&!is.na(salary))*

Finally, we add their names and last names, so we know who we’re talking about and here’s a code that does that:

*players<-Master%>%*

*+ select(playerID,nameFirst,nameLast,debut)%>%*

*+ right\_join(players,by="playerID")*

So now, we have a table with our predicted run statistic, some other statistics, the players’ name, their position and their salary. If we look at the top 10 players bases on our run production statistic:

*players%>%select(nameFirst,nameLast,POS,salary,R\_hat)%>%*

*+ arrange(desc(R\_hat))%>%*

*+ top\_n(10)*

*Selecting by R\_hat*

*# A tibble: 10 x 5*

*nameFirst nameLast POS salary R\_hat*

*<chr> <chr> <chr> <int> <dbl>*

*1 Todd Helton 1B 5.00e6 8.23*

*2 Jason Giambi 1B 1.04e7 7.99*

*3 Albert Pujols 3B 6.00e5 7.54*

*4 Nomar Garciaparra SS 9.00e6 7.51*

*5 Jeff Bagwell 1B 1.10e7 7.48*

*6 Alex Rodriguez SS 2.20e7 7.44*

*7 Carlos Delgado 1B 1.94e7 7.37*

*8 Rafael Palmeiro 1B 8.71e6 7.26*

*9 Mike Piazza C 1.06e7 7.16*

*10 Jim Thome 1B 8.00e6 7.16*

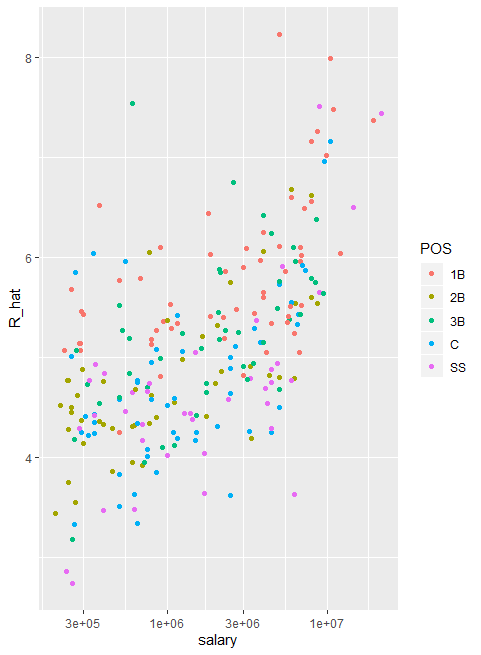
You’re going to recognize some names if you’re a baseball fan. Note that the very high salaries of these players in the top 10. In fact, we see that players with high metric have high salaries. We can see that by making a plot we do see some low-cost players with very high metrics:

*players%>%ggplot(aes(salary,R\_hat,color=POS))+*

*+ geom\_point()+*

*+ scale\_x\_log10()*

1. Players metric by salary



These would be great for our team. Unfortunately, these are likely young players that have not yet been able to negotiate a salary and are not going to be available in 2002. For example, the lowest earner on our top 10 list is Albert Pujols, who was a rookie in 2001. Here’s a plot with players that debuted before 1997:

*players%>%ggplot(aes(salary,R\_hat,color=POS))+*

*+ geom\_point()+*

*+ scale\_x\_log10()*

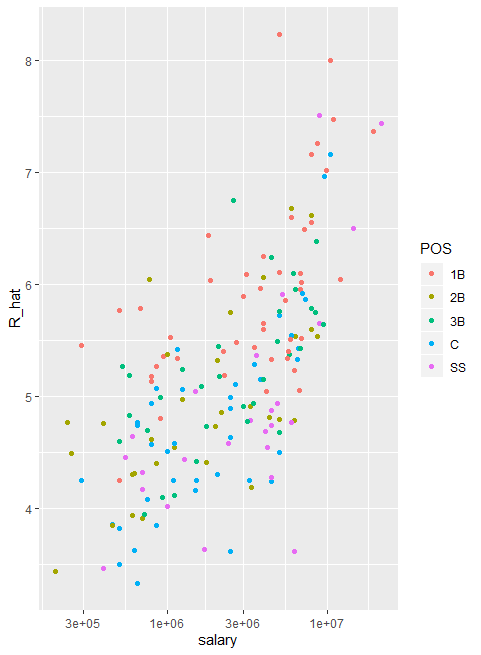
*> players%>%filter(debut<1998)%>%*

*+ ggplot(aes(salary,R\_hat,color=POS))+*

*+ geom\_point()+*

*+ scale\_x\_log10()*

1. Players metric vs salary for players that started before 1997



This remove all the young players. We can now search for good deals by looking at players that produce many more runs and others with similar salaries. We can use this table to decide what players to pick and keep our total salary below the $40 million Billy Beane had to work with.

* + 1. Building a Better Offensive Metric for Baseball: Linear Programming

A way to actually pick the players for the team can be done using what computer scientists call linear programming. Although we don't go into this topic in detail in this course, we include the code anyway:

*library(reshape2)*

*library(lpSolve)*

*players <- players %>% filter(debut <= "1997-01-01" & debut > "1988-01-01")*

*constraint\_matrix <- acast(players, POS ~ playerID, fun.aggregate = length)*

*npos <- nrow(constraint\_matrix)*

*constraint\_matrix <- rbind(constraint\_matrix, salary = players$salary)*

*constraint\_dir <- c(rep("==", npos), "<=")*

*constraint\_limit <- c(rep(1, npos), 50\*10^6)*

*lp\_solution <- lp("max", players$R\_hat,*

*constraint\_matrix, constraint\_dir, constraint\_limit,*

*all.int = TRUE)*

This algorithm chooses these players:

*our\_team <- players %>%*

*filter(lp\_solution$solution == 1) %>%*

*arrange(desc(R\_hat))*

*our\_team %>% select(nameFirst, nameLast, POS, salary, R\_hat)*

*nameFirst nameLast POS salary R\_hat*

*1 Larry Walker RF 12666667 8.146729*

*2 Nomar Garciaparra SS 9000000 7.505063*

*3 Luis Gonzalez LF 4333333 7.397086*

*4 Mike Piazza C 10571429 7.160964*

*5 Jim Edmonds CF 7333333 6.898596*

*6 Phil Nevin 3B 2600000 6.745128*

*7 Greg Colbrunn 1B 1800000 6.438270*

*8 Terry Shumpert 2B 775000 6.044863*

We note that these players generally have above average BB and HR rates while the same is not true for singles.

*my\_scale <- function(x) (x - median(x))/mad(x)*

*players %>% mutate(BB = my\_scale(BB),*

*singles = my\_scale(singles),*

*doubles = my\_scale(doubles),*

*triples = my\_scale(triples),*

*HR = my\_scale(HR),*

*AVG = my\_scale(AVG),*

*R\_hat = my\_scale(R\_hat)) %>%*

*filter(playerID %in% our\_team$playerID) %>%*

*select(nameFirst, nameLast, BB, singles, doubles, triples, HR, AVG, R\_hat) %>%*

*arrange(desc(R\_hat))*

*nameFirst nameLast BB singles doubles triples HR AVG R\_hat*

*1 Larry Walker 1.06046632 0.65537604 0.9222382 1.5623740 1.5655295 2.8353966 2.9036338*

*2 Nomar Garciaparra 0.02740822 1.63711529 3.1180186 0.3360131 0.6245544 3.1968196 2.2442192*

*3 Luis Gonzalez 0.70462683 0.00000000 1.4130409 0.5372768 1.3546270 1.8287077 2.1332555*

*4 Mike Piazza 0.31291096 -0.05467554 -0.2423260 -1.2744279 2.0347431 1.2518167 1.8906021*

*5 Jim Edmonds 1.80740051 -1.14093605 0.6744908 -0.6744908 1.2636708 0.5794012 1.6209774*

*6 Phil Nevin 0.49093751 -0.64789374 0.7639165 -1.0981114 1.5479606 0.7282880 1.4632640*

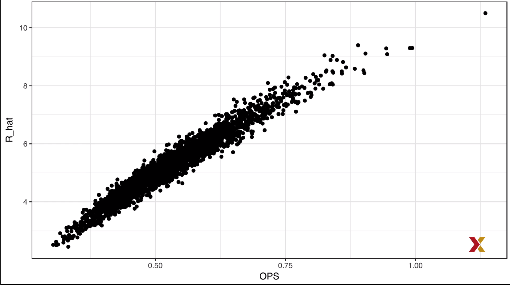
*7 Greg Colbrunn 0.27030518 0.65458272 0.7842985 0.5852173 0.4747056 1.3745471 1.1479184*

*8 Terry Shumpert -0.15757676 0.12209840 1.3257086 3.9081446 -0.1227625 0.8594304 0.7436304*

* + 1. On Base Plus Slugging (OPS)

Since the 1980s sabermetricians have used a summary statistic different form batting average to evaluate players. They realized walks were important, and that doubles, triples, and home runs should be weighted much more than singles and proposed the following metric. They call this on-base-percentage plus slugging percentage or OPS. Today, this statistic has caught on, and you see it in ESPSN and other sports networks. Although the sabermetricians are probably not using regression, this metric is impressively close to what one gets with regression to the summary statistic that we created here is the plot:

1. OPS vs prediction of runs



* + 1. Regression Fallacy

Wikipedia defines the sophomore slump in the following way. A sophomore slump or sophomore jinx or sophomore jitters refers to an instance in which a second, or sophomore, effort fails to live up to the standard of the first effort. It is commonly used to refer to the apathy of students, second year of high school, college, university, performance athletes, second season of series, singer/ bands second album, movies sequel or prequels. We hear about the sophomore slump often in Major League Baseball. This is because in Major League Baseball, the Rookie of the Year, which is an award that’s given to the thirst year player that is judged to have performed the best, usually does not perform as well during their second year. Therefor they call this the sophomore slump. Know, for example, that in a 2016 Fox sports article they asked, will MLB’s tremendous rookie class of 2015 suffer a sophomore slump. Now does the data confirm the existence of a sophomore slump? Let’s take a look and examine the data batting averages to see if the observation holds true. The data is available in the Lehman Library, but we have to do some work to create a table with statistics for all the rookies of the year. Let’s go through then, first we create a table with player ID, their name and their most played position:

*playerInfo<-Fielding%>%*

*+ group\_by(playerID)%>%*

*+ arrange(desc(G))%>%*

*+ slice(1)%>%*

*+ ungroup%>%*

*+ left\_join(Master,by="playerID")%>%*

*+ select(playerID,nameFirst,nameLast,POS)*

Now we will create a table with only the Rookie of the Year Award winners and add their batting statistics. We’re going to filter out pitchers since pitcher are not giving awards for batting, and we’re going to focus on offense. Specifically, we’ll focus on batting average since it is the summary that most pundits talk about when discussing the sophomore slump. So, we write this piece of code to do this:

ROY<-AwardsPlayers%>%

+ filter(awardID=="Rookie of the Year")%>%

+ left\_join(playerInfo,by="playerID")%>%

+ rename(rookie\_year=yearID)%>%

+ right\_join(Batting,by="playerID")%>%

+ mutate(AVG=H/AB)%>%

+ filter(POS!="P")

Now we’ll keep only the rookie and sophomore seasons and remove players that did not play a sophomore season. Remember, now we’re only looking at players that won the Rookie of the Year award. This code achieves what we want:

*ROY<-ROY%>%*

*+ filter(yearID==rookie\_year | yearID==rookie\_year+1)%>%*

*+ group\_by(playerID)%>%*

*+ mutate(rookie=ifelse(yearID==min(yearID),"rookie","sophomore"))%>%*

*+ filter(n()==2)%>%*

*+ ungroup%>%*

*+ select(playerID,rookie\_year,rookie,nameFirst,nameLast,AVG)*

Finally, we will use the spread function to have one column for the rookie and another column for the sophomore years’ batting averages. For that we use this simple line of code:

*ROY<-ROY%>%spread(rookie,AVG)%>%arrange(desc(rookie))*

*ROY*

*playerID rookie\_year nameFirst nameLast rookie sophomore*

*<chr> <int> <chr> <chr> <dbl> <dbl>*

*1 mccovwi01 1959 Willie McCovey 0.354 0.238*

*2 suzukic01 2001 Ichiro Suzuki 0.350 0.321*

*3 bumbral01 1973 Al Bumbry 0.337 0.233*

*4 lynnfr01 1975 Fred Lynn 0.331 0.314*

*5 pujolal01 2001 Albert Pujols 0.329 0.314*

*6 troutmi01 2012 Mike Trout 0.326 0.323*

*7 braunry02 2007 Ryan Braun 0.324 0.285*

*8 olivato01 1964 Tony Oliva 0.323 0.321*

*9 hargrmi01 1974 Mike Hargrove 0.323 0.303*

*10 darkal01 1948 Al Dark 0.322 0.276*

Now we can see the top performers in their first year. These are the Rookie of the Year Award winners and we’re showing their rookie season batting average and their sophomore season batting average. Look closely and you will see the sophomore slump. It definitely appears to be real. In fact, the proportion of players that have a lower batting average their sophomore years is 68%. So is it jitters? Is it a jinx? To answer this question, let’s turn out attention to all players. We’re going to look at the 2013 season and 2014 season. We’re going to look at players that batted at least 130 times.

This is a minimum to win the Rookie of the Year. We’re going to perform a similar operation as we did before to construct this data set:

*two\_years<-Batting%>%*

*+ filter(yearID %in% 2013:2014)%>%*

*+ group\_by(playerID,yearID)%>%*

*+ filter(sum(AB)>=130)%>%*

*+ summarize(AVG=sum(H)/sum(AB))%>%*

*+ ungroup%>%*

*+ spread(yearID,AVG)%>%*

*+ filter(!is.na(`2013`)& !is.na(`2014`))%>%*

*+ left\_join(playerInfo,by="playerID")%>%*

*+ filter(POS!="P")%>%*

*+ select(-POS)%>%*

*+ arrange(desc(`2013`))%>%*

*+ select(-playerID)*

*two\_years*

*# A tibble: 312 x 4*

*`2013` `2014` nameFirst nameLast*

*<dbl> <dbl> <chr> <chr>*

*1 0.348 0.313 Miguel Cabrera*

*2 0.345 0.283 Hanley Ramirez*

*3 0.331 0.332 Michael Cuddyer*

*4 0.324 0.289 Scooter Gennett*

*5 0.324 0.277 Joe Mauer*

*6 0.323 0.287 Mike Trout*

*7 0.321 0.263 Chris Johnson*

*8 0.319 0.288 Freddie Freeman*

*9 0.319 0.296 Yasiel Puig*

*10 0.319 0.282 Yadier Molina*

*# ... with 302 more rows*

Note that the same pattern arises when we look at the top performers. Batting averages go down for the top performers. But these are not rookies. So this can’t be explained with a sophomore slump, also know what happens to the worst performers of 2013:

*arrange(two\_years,`2013`)*

*# A tibble: 312 x 4*

*`2013` `2014` nameFirst nameLast*

*<dbl> <dbl> <chr> <chr>*

*1 0.158 0.219 Danny Espinosa*

*2 0.179 0.149 Dan Uggla*

*3 0.181 0.2 Jeff Mathis*

*4 0.184 0.208 Melvin Upton*

*5 0.190 0.262 Adam Rosales*

*6 0.192 0.215 Aaron Hicks*

*7 0.194 0.229 Chris Colabello*

*8 0.194 0.177 J. P. Arencibia*

*9 0.195 0.241 Tyler Flowers*

*10 0.198 0.218 Ryan Hanigan*

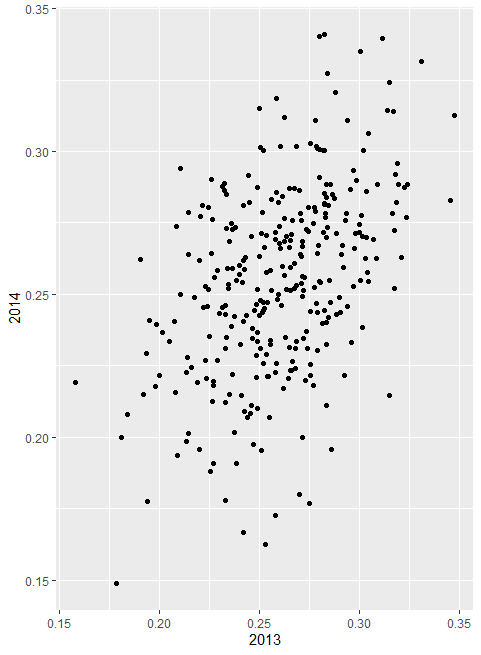
Their batting averages go up in their second season in 2014. Is this some sort of reverse sophomore slump? Nope.

There is no such thing as a sophomore slump. This is all explained with a simple fact. The correlation of performance in two separate years is high but not perfect. Here is the data for 2013 and 2014:

*two\_years%>%ggplot(aes(`2013`,`2014`))+*

*geom\_point()*

1. performance of 2013 vs 2014



You can see it’s correlated. But it’s not perfectly correlated. The correlation is 0.46. The data look very much like a bivariate normal distribution, which means that if we were to predict the 2014 batting average, let’s call it y for any given player that had a 2013 batting average of x, we would use the regression equation, which would be this:

Because a correlation is not perfect, regression tells us that on average, we expect high performers from 2013 to do a little bit worse in 2014. This regression to the mean. It’s not a jinx. It’s just due to chance. The rookies of the year are selected from the top values of x, so it is expected that their y will regress to the mean.

* + 1. Measurement Error Models

Up until now, all our linear regression examples have been applied to two or more random variables. We assume the pairs are bivariate normal and use this to motivate a linear model. This approach covers most of real life examples where linear regression is used. The other major application comes form measurement error models. In these applications, it is common to have a nonrandom covariate, such as time, and randomness is introduced from measurement error rather than sampling or natural variability. To understand these models, we’re going to use a motivation example related to physics. Imagine your are Galileo in the 16th century trying to describe the velocity of a falling object, an assistant climbs the Tower of Pisa and drops a ball. While several record the position at different times. The falling object data set contains an example of what that data would look like. The assistant hands the data to Galileo and this is what he sees:

*falling\_object%>%*

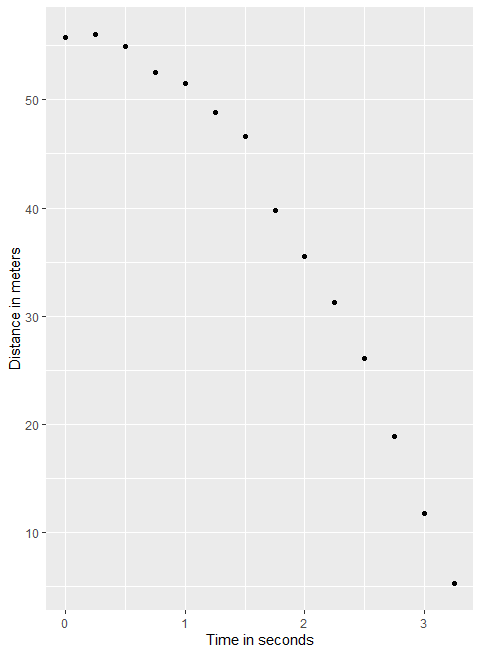
*+ ggplot(aes(time,observed\_distance))+*

*+ geom\_point()+*

*+ ylab("Distance in meters")+*

*+ xlab("Time in seconds")*

1. Measurement of the ball position versus time



Here we see the distance in meters that has dropped on the y-axis and time on the x-axis. Galileo doesn’t know the exact equation, but from data exploration, by looking at the plot, he deduces that the position should follow a parabola, which we can write like this:

The data does not fall exactly on a parabola, but Galileo know that this is due to measurement error. His helpers makes mistakes when measuring the distance the ball has fallen. To account for this, we write this model:

Here, y represents the distance the ball is dropped in meters. Xi represents time in seconds, and epsilon represents measurement error. The measurement error is assumed to be random, independent from each other and having the same distribution from each eye. We also assume that there is no bias, which means that the expected value of epsilon is 0. Note that this is a linear model because it is a linear combination of known quantities. X and x squared are known and unknown parameters, the betas. Unlike our previous example, the x’s are fixed quantities. This is just time. We’re not conditioning. Now to pose a new physical theory and start making predictions about other falling objects, Galileo needs actual numbers, rather than the unknown parameters. The least estimates seem like a reasonable approach. So how do we find the least squares estimates? Note that the LSE calculations do not require the errors to be approximately normal. The lm() function will find the betas and minimize the residual sum of squares, which is what we want:

*fit<-falling\_object%>%*

*+ mutate(time\_sq=time^2)%>%*

*+ lm(observed\_distance~time+time\_sq,data=.)*

*tidy(fit)*

*# A tibble: 3 x 5*

*term estimate std.error statistic p.value*

*<chr> <dbl> <dbl> <dbl> <dbl>*

*1 (Intercept) 56.1 0.569 98.7 1.44e-17*

*2 time -0.265 0.813 -0.327 7.50e- 1*

*3 time\_sq -4.79 0.241 -19.9 5.78e-10*

To check if the estimated parabola fits the data, the broom function augment() let us do this easily:

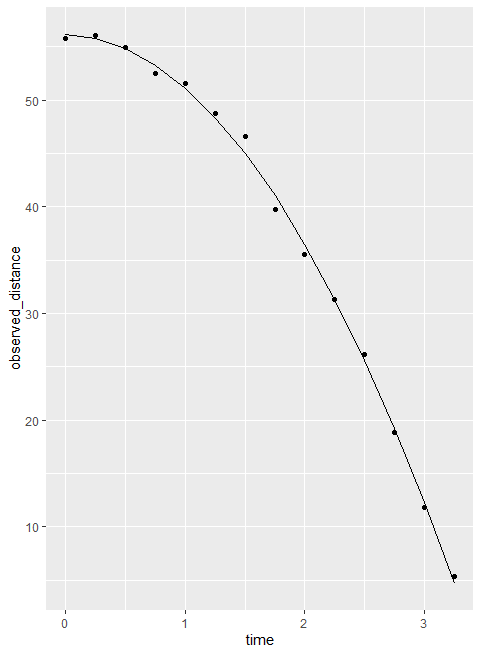
*augment(fit)%>%*

*+ ggplot()+*

*+ geom\_point(aes(time,observed\_distance))+*

*+ geom\_line(aes(time,.fitted))*

1. Observed data versus predicted data



Note that the predicted values go right through the points. Now, thanks to my high school physics teacher. I know that the equation for the trajectory of a falling object is the following:

With h0 and v0 the starting height and starting velocity respectively. The data we use follow this equation and added measurement error to simulate an observation. Dropping the ball, that means the starting velocity is 0 because we start just by dropping if from the Tower of Pisa, which has a height of about 56.67 meters. These known quantities are consistent with the parameters that we estimated, which we can see using the tidy function:

*tidy(fit,conf.int = TRUE)*

*# A tibble: 3 x 7*

*term estimate std.error statistic p.value conf.low conf.high*

*<chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>*

*1 (Intercept) 56.1 0.569 98.7 1.44e-17 54.9 57.4*

*2 time -0.265 0.813 -0.327 7.50e- 1 -2.05 1.52*

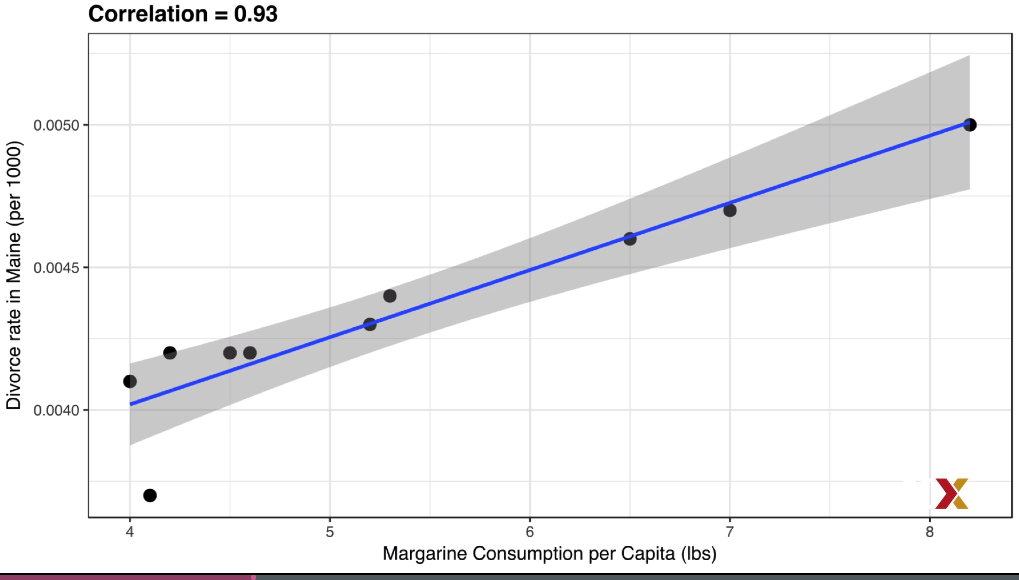
*3 time\_sq -4.79 0.241 -19.9 5.78e-10 -5.32 -4.25*

The tower of Pisa height is within the confidence interval for beta 0. The initial velocity of 0 is in the confidence interval for beta 1. Note that the p value is larger than 0.05, which means we wouldn’t reject the hypothesis that the starting velocity is 0, and so is the acceleration.

1. Correlation is Not Causation
   1. Spurious Correlation

Correlation is not causation is perhaps the most important lesson one learns in a statistics class. In this course, we have described tools useful for quantifying associations between variables, but we must be careful not to over interpret these associations. There are many reasons that a variable x can correlate with a variable y, without wither being a cause for the other. Here we examine common ways that can lead to misinterpreting association. The first example of how we can misinterpret associations are spurious correlations. The following comical example underscores that correlation is not causation. The example shows a very strong correlation between divorce rates and margarine consumption

1. Correlation between Divorce rates and margarine consumption



Does this mean that margarine causes divorces, or do divorces cause people to eat more margarine? Of course, the answer to both these questions is no. This is just an example of what we call spurious correlations. You can see many other examples on this website: <http://tylervigen.com/spurious-correlations> .

The cases presented in the spurious correlation site are all examples of what is generally called data dredging, or data phishing, or data snooping. It’s basically a form of what in the United States, they call cherry picking. An example of data dredging would be if you look through many results produced by a random process, and prick the one that shows a relationship that supports the theory you want to defend. A Monte Carlo simulation can be used to show how data dredging can result in finding high correlations among variables that are theoretically uncorrelated. For example:

*N<-25*

*> G<-1000000*

*> sim\_data<-tibble(group=rep(1:G,each=N),X=rnorm(N\*G),Y=rnorm(N\*G))*

The first column denotes group and we simulated one million groups, each with 25 observations. For each group, we generate 25 observations which are stored in the second and third column. These are just random, independent normally distributed data. So we know, because we constructed the simulation, that x and y are not correlated. Next, we compute the correlation between x and y for each group, and look for the maximum:

*res<-sim\_data%>%*

*+ group\_by(group)%>%*

*+ summarize(r=cor(X,Y))%>%*

*+ arrange(desc(r))*

*> res*

*# A tibble: 1,000,000 x 2*

*group r*

*<int> <dbl>*

*1 60747 0.762*

*2 352546 0.757*

*3 365938 0.754*

*4 534679 0.751*

*5 660910 0.750*

*6 141808 0.749*

*7 854856 0.748*

*8 915529 0.741*

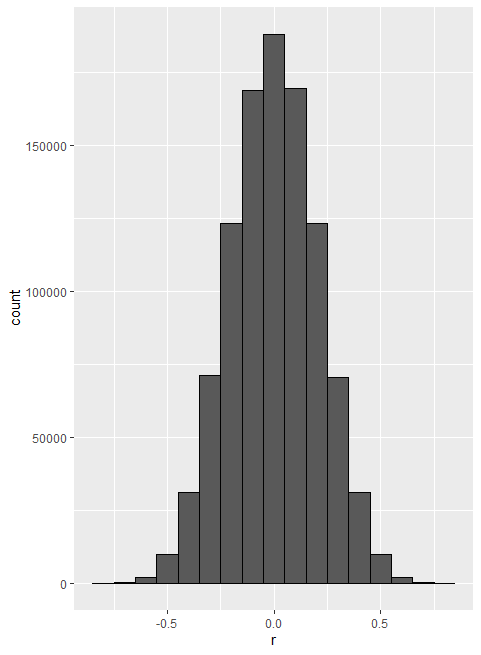
*9 226858 0.739*

*10 924945 0.737*

If we just plot the data from this particular group, it shows a convincing plot that x and y are, in fact, correlated. But remember that the correlations number is a random variable. Here’s the distribution we just generated with our Monte Carlo simulation:

*res%>%ggplot(aes(x=r))+geom\_histogram(binwidth = 0.1,color="black")*

1. Distribution of the Monte Carlo simulation



It is just a mathematical fact that if we observe random correlations that are expected to be 0, but have a standard error of about 0.2, the largest one will be close to 1 if we pick from among one million. Note that if we performed regression on this group and interpreted the p-value, we would incorrectly claim this was a statistically significant relation:

*sim\_data%>%*

*+ filter(group==res$group[which.max(res$r)])%>%*

*+ do(tidy(lm(Y~X,data=.)))*

*# A tibble: 2 x 5*

*term estimate std.error statistic p.value*

*<chr> <dbl> <dbl> <dbl> <dbl>*

*1 (Int~ 0.527 0.170 3.10 5.09e-3*

*2 X 0.844 0.149 5.65 9.48e-6*

This form of data dredging is referred to as p-hacking. P-hacking is a topic of much discussion because it is a problem in scientific publications. Because publishers tend to reward statically significant results over negative results, there’s an incentive to report significant results. In epidemiology in the social sciences for example, researchers may look for associations between an average outcome and several exposures, and report only the one exposure that resulted in a small p-value. Furthermore, they might try fitting several different models to adjust for confounding and pick the one model that yields the smallest p-value. In experimental disciplines, an experiment might be repeated more than once, and only the one that results in a small p-value are reported. This does not necessarily happen due to unethical behavior, but rather to statistical ignorance or wishful thinking.

* 1. Correlation is Not Causation: Outliers

Another way that we can see high correlations when there’s no causation is when we have outliers. Suppose we take measurements from two independent outcomes, x and y, and we standardize the measurements. However, imagine we made a mistake and forgot to standardize entry 23. We can simulate such data using the following code:

*set.seed(1)*

*> x<-rnorm(100,100,1)*

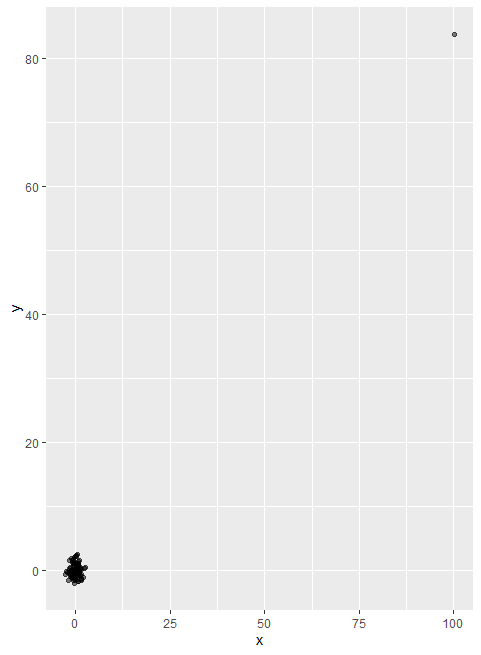
*> y<-rnorm(100,84,1)*

*> x[-23]<-scale(x[-23])*

*> y[-23]<-scale(y[-23])*

*> tibble(x,y)%>%ggplot(aes(x,y))+geom\_point(alpha=0.5)*

1. High correlated data x and y



Not surprisingly, the correlation is very high. That one outlier is making the correlation be as high as 0.99

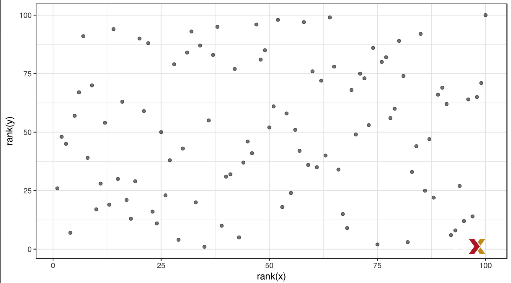
If we remove that outlier, the correlation is greatly reduced to almost 0, which is what it should be. Here’s what we get it remove the entry:

*cor(x[-23],y[-23])*

*[1] -0.001066464*

There is an alternative way to sample correlation for estimating the population correlation that is robust to outliers. It is called Spearman correlation. The ideas is simple, we compute the correlation on the ranks of the values, rather than the values themselves.

1. Plot of the ranks plotted against each other



Note that the one point that’s very large is just at the 100-100 location. It is no longer really out there and pulling the correlation towards 1. So if we compute the correlation of the ranks:

*cor(rank(x),rank(y))*

*[1] 0.06583858*

We get something much closer to 0. Spearman correlation can also be calculated with the correlation function, but using the method argument to tell cor which correlation to compute:

*cor(x,y,method="spearman")*

*[1] 0.06583858*

* 1. Correlation is Not Causation: Reversing Cause and Effect

Another way association are confounded with causation is when the cause and effect are reversed. An example of this is claiming that tutoring makes students perform worse because they test lower than peers that are not tutored. Here, the tutoring is not causing the low test, but the other way around. A form of this claim was actually made in an op ed in the New York Times titled “Parental Involvement is Overrated”. Consider this quote from the article:

“When we examine whether regular help homework had a positive impact on children’s academic performance, we were quite startled by what we found. Regardless of family social class, racial, or ethnic background, or child’s grade level, consistent homework helps almost never improved test scores or grades. Even more surprising to us was that when parents regularly helped with homework, kids usually performed worse”.

A very likely possibility is that children needing regular parental help get this help because they don’t perform well in school. To see another example, we’re going to use one the data sets that we’ve seen in this course. Specifically, we can easily construct an example of cause and effect reversal using the father and son height data. Note that if we fit the following model:

With X representing the father’s height and y the son’s height, we do get a statistically significant result. You can see that with this simple code:

*library(HistData)*

*> data("GaltonFamilies")*

*> GaltonFamilies%>%*

*+ filter(childNum==1 & gender=="male")%>%*

*+ select(father,childHeight)%>%*

*+ rename(son=childHeight)%>%*

*+ do(tidy(lm(father~son,data=.)))*

*# A tibble: 2 x 5*

*term estimate std.error statistic p.value*

*<chr> <dbl> <dbl> <dbl> <dbl>*

*1 (Intercept) 34.0 4.57 7.44 4.31e-12*

*2 son 0.499 0.0648 7.70 9.47e-13*

This model fits the data very well. However, if we look at the mathematical formulation of the model, it could easily be incorrectly interpreted as to suggest that the son being tall caused the father to be tall. But given what we know about genetics and biology, we know it’s the other way around. The model is technically correct. The estimates and p-value were obtained correctly as well. What is wrong here is simply the interpretation.

* 1. Correlation is Not Causation: Confounders

Confounders are perhaps the most common reason that leads to associations being misinterpreted. If X and Y are correlated, we call Z a confounder if changes in Z cause changes in both X and Y. Earlier, when studying baseball data, we saw how home runs was a confounder that resulted in higher correlation than expected when studying the relationship between bases on balls and runs. In some cases, we can use linear models to account for confounders, as we did in the baseball example. But it is not always possible. Incorrect interpretation due to confounders are ubiquitous in the lay press. They are sometimes hard to detect. We examined admission data from UC Berkeley majors from 1973 that showed that more men were being admitted than women. 44% men were admitted compared to 30% women, here’s the data:

*data("admissions")*

*> admissions*

*major gender admitted applicants*

*1 A men 62 825*

*2 B men 63 560*

*3 C men 37 325*

*4 D men 33 417*

*5 E men 28 191*

*6 F men 6 373*

*7 A women 82 108*

*8 B women 68 25*

*9 C women 34 593*

*10 D women 35 375*

*11 E women 24 393*

*12 F women 7 341*

The percent of men and women that got accepted can be computed from this data using this simple piece of code:

*admissions%>%group\_by(gender)%>%*

*+ summarize(percentage=round(sum(admitted\*applicants)/sum(applicants),1))*

*# A tibble: 2 x 2*

*gender percentage*

*<chr> <dbl>*

*1 men 44.5*

*2 women 30.3*

A statistical test, the chi-squared test that we learned about in a previous chapter, clearly rejects the hypothesis that gender and admissions are independent. The p-value is very small:

*admissions%>%group\_by(gender)%>%*

*+ summarize(total\_admitted=round(sum(admitted/100\*applicants)),*

*+ not\_admitted=sum(applicants)-sum(total\_admitted))%>%*

*+ select(-gender)%>%*

*+ do(tidy(chisq.test(.)))*

*# A tibble: 1 x 4*

*statistic p.value parameter method*

*<dbl> <dbl> <int> <chr>*

*1 91.6 1.06e-21 1 Pearson's Chi-squared test with Yates' continuity correction*

But closer inspection shows a paradoxical result. Here are the admissions by major:

*admissions%>%select(major,gender,admitted)%>%*

*+ spread(gender,admitted)%>%*

*+ mutate(women\_minus\_men=women-men)*

*major men women women\_minus\_men*

*1 A 62 82 20*

*2 B 63 68 5*

*3 C 37 34 -3*

*4 D 33 35 2*

*5 E 28 24 -4*

*6 F 6 7 1*

Four out of the six majors favor women. But more importantly, all the differences are much smaller than the 14% difference that we see when examining the totals. The paradox is that analyzing the totals suggest a dependence between admissions and gender. But when the data is grouped by major, this dependence seems to disappear. What’s going on? This actually can happen if an uncounted confounder is driving most of the variability. So let’s define 3 variables:

* X is 1 for men and 0 for women
* Y is 1 for those admitted and 0 otherwise
* Z quantifies how selective the major is

A gender bias claim would be based on the fact that this probability:

Is higher when X is 1 than when X is 0. But Z is an important confounder. Clearly, Z is associated with Y, because the more selective a major, the lower the probability that someone enters that major. But is major selectivity, which we call Z, associated with gender? One way to see this is to plot the total percent admitted to a major versus the percent of women that make up the applicants:

*admissions%>%*

*+ group\_by(major)%>%*

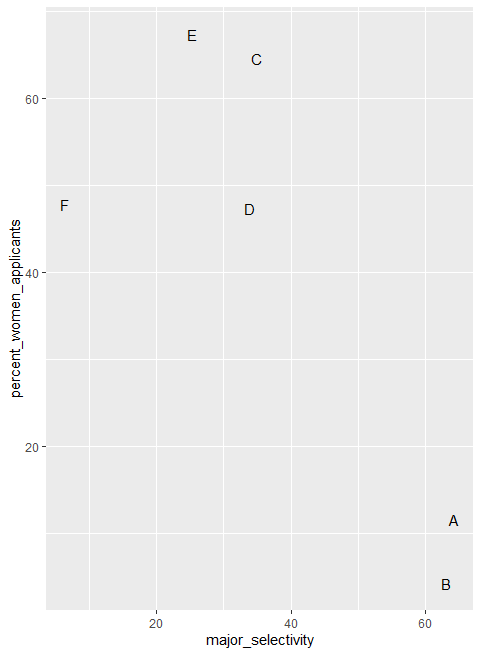
*+ summarize(major\_selectivity=sum(admitted\*applicants)/sum(applicants),*

*+ percent\_women\_applicants=sum(applicants\*(gender=="women")/sum(applicants))\*100)%>%*

*+ ggplot(aes(major\_selectivity,percent\_women\_applicants,label=major))+*

*+ geom\_text()*

1. total percent admitted to a major versus the percent of women that make up the applicants



There seems to be an association. The plot suggest that women were much more likely to apply to the two hard majors. Gender and major selectivity are confounded. Compare, for example, major B and E. Major E is much harder to enter than major B, and over 60% of applicants to major E were women, while less than 30% of the applicants of major B were women. The following plot shows the percentage of applicants that were accounted by gender:

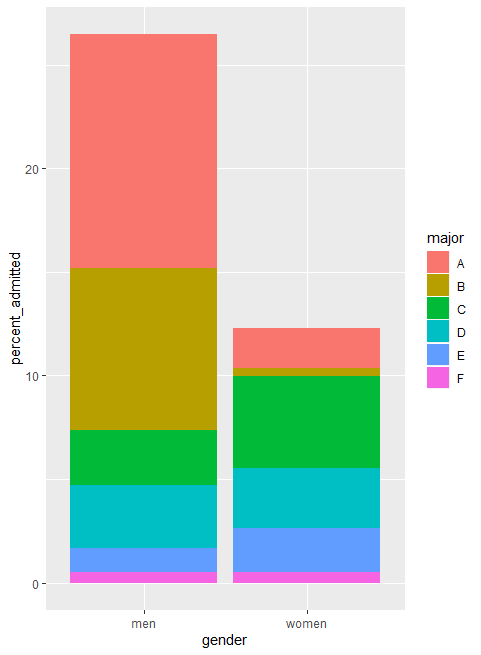
*admissions%>%*

*+ mutate(percent\_admitted=admitted\*applicants/sum(applicants))%>%*

*+ ggplot(aes(gender,y=percent\_admitted,fill=major))+*

*+ geom\_bar(stat="identity",position="stack")*

1. percentage of applicants that were accounted by gender



The color here represents major. It also breaks down the acceptance rates by major. The size of the colored bar represents the percent of each major students that were admitted to. This breakdown let us see that the majority of accepted men came from two majors, A and B. It also let us see that few women apply to these two easy majors.

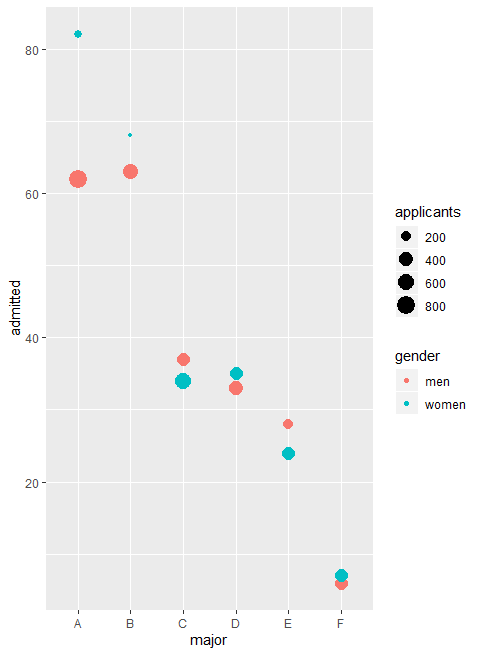
What the plot does not show us is what’s the percent admitted by major. In this plot, we can see that if we condition or stratify by major and then look at the differences, we control for the confounder, and the effect goes away:

*admissions%>%*

*+ ggplot(aes(major,admitted,col=gender,size=applicants))+*

*+ geom\_point()*

1. Gender acceptance stratify by major

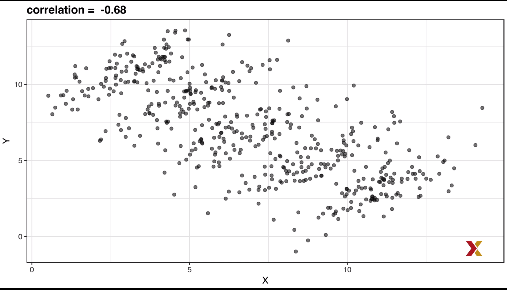


Now we see that major by major, there’s not much difference. The size of the dot represents the number of applicants, and explains the paradox. We see large red dots and small blue dots for the easiest majors A and B. If we first stratify by major, compute the difference and then average, we find that the percent difference is actually quite small. This is an example of something that is called Simpson’s paradox, which we will describe in the next lesson.

* 1. Simpson's Paradox

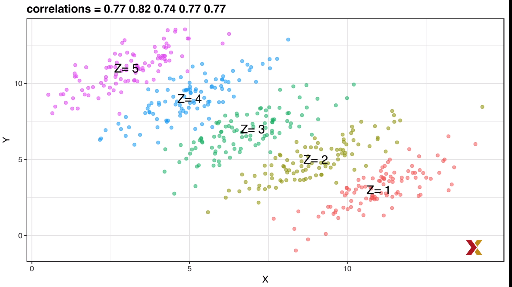
We have just seen an example of Simpsons’ paradox. It is called a paradox because we see the sign of the correlation flip when we computed on the entire population and when computed on specific strata. Now, we’re going to use a very illustrative simulated example, to show you how this can happen. Suppose you have 3 variables, x,y and z. Here’s a scatter plot of y versus x:

1. Example of a scatter plot of 2 variables x and y



You can see that x and y are negatively correlated. However once we stratify by z, the confounder which we haven’t looked at yet, we see that another pattern emerges

1. Example of the same scatter plot stratify by a confounder z



If you compute the correlation in each strata, you see that the correlations are now positive. So, it’s really z that is negatively correlated with x. If we stratify by z, the x and y are actually positively correlated. This is an example of Simpson’s paradox.