Course Name

February, 2024

Table of Contents

# About this Course

## 0.1 Available course formats

This course is available in multiple formats which allows you to take it in the way that best suites your needs. You can take it for certificate which can be for free or fee.

* The material for this course can be viewed without login requirement on this [Bookdown website](LINK%20HERE). This format might be most appropriate for you if you rely on screen-reader technology.
* This course can be taken for [free certification through Leanpub](LINK%20HERE).
* This course can be taken on [Coursera for certification here](LINK%20HERE) (but it is not available for free on Coursera).
* Our courses are open source, you can find the [source material for this course on GitHub](LINK%20HERE).

## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)

## Loading required package: MASS

## Warning: package 'MASS' was built under R version 4.0.3

## Loading required package: HistData

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

##   
## Attaching package: 'UsingR'

## The following object is masked from 'package:survival':  
##   
## cancer

## Warning: package 'reshape' was built under R version 4.0.3

# 1 Week 01

## 1.1 Introduction

### 1.1.1 Welcome to Regression Models

I am happy that you’ve chosen to take Regression Models, part of the Johns Hopkins Data Science Specialization on Coursera! This course presents the fundamentals of regression modeling that you will need for the rest of the specialization and ultimately for your work in the field of data science.

We believe that the key word in Data Science is “science”. Our course track is focused on providing you with three things: (1) an introduction to the key ideas behind working with data in a scientific way that will produce new and reproducible insight, (2) an introduction to the tools that will allow you to execute on a data analytic strategy, from raw data in a database to a completed report with interactive graphics, and (3) on giving you plenty of hands on practice so you can learn the techniques for yourself.

Regression Models represents a both fundamental and foundational component of the series, and it presents the single most practical data analysis toolset. Using only a bare minimum of mathematics, we will attempt to provide you with the fundamentals for the application and practice of regression.

We are excited about the opportunity to attempt to scale Data Science education. We intend for the courses to be self-contained, fast-paced, and interactive, and we intend to run them frequently to give people with busy schedules the opportunity to work on material at their own pace.

### 1.1.2 Some Basics

A couple of first week housekeeping items. First, make sure that you’ve had [R Programming](https://www.coursera.org/learn/r-programming) , the [Data Scientist’s Toolbox](https://www.coursera.org/learn/data-scientists-tools), [Reproducible Research](https://www.coursera.org/learn/regression-models/supplement/uCPA0/welcome-to-regression-models) and [Statistical Inference](https://www.coursera.org/learn/statistical-inference) before taking this class. At a minimum you must know: very basic git, basic R and most of the Statistical Inference Coursera class. The small amount of knitr that you need for the project you can pick up quickly.

An important aspect of this class is to peruse the materials in the github repository. All of the most up to date material can be found [here](https://github.com/bcaffo/courses/tree/master/07_RegressionModels). You should clone this repository as your first step in this class and make sure to fetch updates periodically. (Please issue pull requests so that we may improve the materials!) It is one of the most essential components of the Specialization that you start to use Git frequently. We’re practicing what we preach as well by using the tools in the series to create the series, especially git. Note my [GitHub repo](https://github.com/bcaffo/courses) will generally be more up to date than the Data Science Specialization Repo.

The lectures are in the index.Rmd lecture files. In [Developing Data Products](https://www.coursera.org/learn/data-products), we cover how to create these sorts of slides. However, for the time being, you should be able to open them in R Studio and look at their contents. You will see all of the R code to recreate the lectures. Going through the R code is the best way to familiarize yourself with the lecture materials.

#### 1.1.2.1 YouTube

If you’d prefer to watch the videos on YouTube, you can find them [here](https://www.youtube.com/playlist?list=PLpl-gQkQivXjqHAJd2t-J_One_fYE55tC) and [here](https://www.youtube.com/playlist?list=PLpl-gQkQivXhdgUCdaUQcdb31CRe8Mm2y). If you’d like to keep up with the instructors I’m ([**bcaffo?**](#ref-bcaffo)) on twitter, Roger is ([**rdpeng?**](#ref-rdpeng)) and Jeff is ([**jtleek?**](#ref-jtleek)). The Department of Biostat here is ([**jhubiostat?**](#ref-jhubiostat)).

### 1.1.3 Syllabus (xxx)

Course Title: Regression Models

Course Instructor(s):The primary instructor of this class is [Brian Caffo](https://sites.google.com/view/bcaffo/home/). Brian is a professor at Johns Hopkins Biostatistics and co-directs the [SMART working group](https://www.smart-stats.org).

This class is co-taught by Roger Peng and Jeff Leek. In addition, Sean Kross and Nick Carchedi have been helping greatly.

#### 1.1.3.1 Course Description:

Linear models, as their name implies, relates an outcome to a set of predictors of interest using linear assumptions. Regression models, a subset of linear models, are the most important statistical analysis tool in a data scientist’s toolkit. This course covers regression analysis, least squares and inference using regression models. Special cases of the regression model, ANOVA and ANCOVA will be covered as well. Analysis of residuals and variability will be investigated. The course will cover modern thinking on model selection and novel uses of regression models including scatterplot smoothing.

#### 1.1.3.2 Course Content

This class has three main components:

* Least squares and linear regression
* Multivariable regression
* Generalized linear models

The full list of topics are as follows:

* Module 1, least squares and linear regression
  + 01\_01 Introduction
  + 01\_02 Notation
  + 01\_03 Ordinary least squares
  + 01\_04 Regression to the mean
  + 01\_05 Linear regression
  + 01\_06 Residuals
  + 01\_07 Regression inference
* Module 2, Multivariable regression
  + 02\_01 Multivariate regression
  + 02\_02 Multivariate examples
  + 02\_03 Adjustment
  + 02\_04 Residual variation and diagnostics
  + 02\_05 Multiple variables
* Module 3, Generalized linear models
  + 03\_01 GLMs
  + 03\_02 Binary outcomes
  + 03\_03 Count outcomes
  + 03\_04 Olio
* Module 4, Logistic Regression and Poisson Regression
  + 04\_01 Logistic Regression
  + 04\_02Poisson Regression
  + 04\_03 Hodgepodge

#### 1.1.3.3 Book: Regression Models for Data Science in R.

A companion book is available [here](https://leanpub.com/regmods). The book is published via leanpub, and the suggested price is $14.99. You can get it for free or pay what you feel it is worth.

#### 1.1.3.4 Quizzes

There are four weekly quizzes. You must earn a grade of at least 80% to pass a quiz. You may attempt each quiz up to 3 times in 8 hours. The score from your most successful attempt will count toward your final grade.

#### 1.1.3.5 Course Project

The Course Project is an opportunity to demonstrate the skills you have learned during the course. It is graded through peer assessment. You must earn a grade of at least 80% to pass the peer assessment.

#### 1.1.3.6 Grading Policy

You must score at least 80% on all assignments (Quizzes & Project) to pass the course.

Your final grade will be calculated as follows:

Quiz 1 = 15% Quiz 2 = 15% Quiz 3 = 15% Quiz 4 = 15% Course Project = 40%

##### 1.1.3.6.1 swirl Programming Assignment (optional)

In this course, you have the option to use the [swirl](https://swirlstats.com) R package to practice some of the concepts we cover in lectures.

While these lessons will give you valuable practice and you are encouraged to complete as many as possible, please note that they are completely optional and you can get full marks in the class without completing them.

#### 1.1.3.7 Differences of opinion

Keep in mind that currently data analysis is as much art as it is science - so we may have a difference of opinion - and that is ok! Please refrain from angry, sarcastic, or abusive comments on the message boards. Our goal is to create a supportive community that helps the learning of all students, from the most advanced to those who are just seeing this material for the first time.

### 1.1.4 Data Science Specialization Community Site

Since the beginning of the Data Science Specialization, we’ve noticed the unbelievable passion students have about our courses and the generosity they show toward each other on the course forums. A couple students have created quality content around the subjects we discuss, and many of these materials are so good we feel that they should be shared with all of our students.

We’re excited to announce that we’ve created a site using [GitHub Pages](http://datasciencespecialization.github.io/) to serve as a directory for content that the community has created. If you’ve created materials relating to any of the courses in the Data Science Specialization, please send us a pull request so we can add a link to your content on our site. You can find out more about contributing [here.](https://github.com/DataScienceSpecialization/DataScienceSpecialization.github.io#contributing)

We can’t wait to see what you’ve created and where the community can take this site!

### 1.1.5 Where to get more advanced material

If you want more advanced material, I’ve been working on another version of this class. Eventually I hope to have a second Coursera class as well. Currently, you can get the E-Book in progress [here](https://leanpub.com/lm) (it’s variable pricing including free!)

In addition, you can watch the videos as they’re being developed [here.](https://www.youtube.com/playlist?list=PLpl-gQkQivXhdgUCdaUQcdb31CRe8Mm2y)

## 1.2 Introduction to regression and least squares

Regression models are the workhorse of data science. They are the most well described, practical and theoretically understood models in statistics. A data scientist well versed in regression models will be able to solve an incredible array of problems.

Perhaps the key insight for regression models is that they produce highly interpretable model fits. This is unlike machine learning algorithms, which often sacrifice interpretability for improved prediction performance or automation. These are, of course, valuable attributes in their own rights. However, the benefit of simplicity, parsimony and intrepretability offered by regression models (and their close generalizations) should make them a first tool of choice for any practical problem.

### 1.2.1 Introduction to Regression

Hello, I’m Brian Caffo, and I’d like to welcome you to the introduction to regression lecture in the regression Coursera class, part of our data science specialization. Co-taught by my colleagues Jeff Leek and Roger Peng, we all belong to the Department of Biostatistics at the Johns Hopkins Bloomberg School of Public Health.

Regression is a cornerstone for data scientists. Before delving into complex machine learning, linear regression or its generalization, linear models, are often the go-to procedures. The roots of regression trace back to Francis Galton, who coined the term and concept, along with correlation, closely tied to linear regression.

Galton’s prediction of a child’s height from a parent’s height remains historically significant. Jeff Leek [highlights](https://www.nature.com/articles/ejhg20095) its continued relevance in modern genetic analysis, comparing it to Victorian Era measurements. Moving to a more contemporary example, a blog post by Rafael Irazarry on Simply Statistics explores the relationship between Kobe Bryant’s ball-hogging and the Lakers’ performance, utilizing linear regression.

In a modern example, [Simply Statistics](https://simplystatistics.org) blog talks about “[the Lakers wins](https://simplystatistics.org/posts/2013-01-28-data-supports-claim-that-if-kobe-stops-ball-hogging-the-lakers-will-win-more/)” that Data supports claim that if Kobe stops ball hogging the Lakers will win more.The heart of our class is understanding how to formulate and interpret statements like for example in the Simply Statistics blog post “Linear regression suggests an increase of 1% in the percent of shots taken by Kobe results in a drop of 1.16 points.” We’ll delve into good statistical practices, including providing standard errors.

We might want to find a parsimonious and easily described mean relationships between the parent’s and child’s height. So we don’t want anything complicated. We want the simplest possible relationship, and that is what regression is best at. While machine learning and other techniques generate highly elaborate, in many cases, accurate prediction models, they tend to not be parsimonious. They tend not to explain the data, and they tend not to generate new parsimonious knowledge, whereas this is what regression is good at. This is what regression is in fact best at. We can talk about variation that’s unexplained by the regression model. The so called residual variation.

We’re going to connect the results back to the subject of inference. How do we take our data, which is just a sample, it only talks about that data set, and try to figure out what assumptions are needed to extrapolate it to a larger population. This is a deep subject called statistical inference. We have a whole another course of Statistical Inference as part of data science specialization. But we’re going to apply the tools of inference, which we are hoping most of you will have had as a prerequisite. We’re going to apply the tools of inference to this new subject of regression.

Let’s look at Francis Galton’s data, he first used this data in 1885. He’s really an interesting character in history, in general and definitely in the history of statistics. You need to run install.packages("UsingR"). Here UsingR is the package for the book, [Using R for Introductory Statistics](https://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf). It is a great book, and they’ve very kindly packaged all these data sets together in a single R package. So you need to use UsingR then the library UsingR to get a lot of the data sets that we are going to talk about. So let’s first look at the marginal distribution of the parents. In other words, distribution of the parents disregarding children. And the marginal distribution of the children, disregarding parents.

install.packages("UsingR")

Parent distribution is all heterosexual couples, correcting for sex by multiplying the female heights by 1.08.

library(UsingR); data(galton); library(reshape); long<-melt(galton);

## Using as id variables

g<- ggplot(long, aes(x=value, fill=variable))   
g<- g+ geom\_histogram(color='black', binwidth=1)  
g<- g+ facet\_grid(.~variable)  
  
g



On the left, we have the children’s heights. The X-axis is in inches, the scale goes from 60 inches to 75. The Y-axis is the count, the number of children that fall in each bin of heights. On the right in the more bluish teal color, we have the parents heights. We’ve broken the association by the children and the parents by not doing a scatter plot, and only looking at the marginal distribution of the children, and the marginal distribution of the parents by themselves. We would like to use these distributions to introduce least squares, and then we’ll build on the bivaried association after that. So consider only the child’s height,forget for the moment about using the parent’s height to predict the child’s heights. We just want to find maybe the best prediction of the child’s heights without any other information. Well, probably the best predictor would be the middle and how could one define the middle?

One definition, let , be the height for child , where in this dataset . So the middle is the value of that minimizes

That’s how we define the middle. It’s also related to physics in this so called physical center of mass of the histogram that we showed on the previously. Imagine of those bars as being physical entities, having weight and you are trying to figure out where you would put your finger to balance it out. That would be the physical center of mass. You might have guessed that the center of the data has to be the mean.

Let’s use our studio’s manipulate function to experiment with trying to find that center of mass.

library(manipulate)   
myHist<-function(mu){  
 mse<-mean((galton$child - mu)^2)  
 g <- ggplot(galton, aes(x = child)) + geom\_histogram(fill = "salmon", colour = "black", binwidth=1)  
 g <- g + geom\_vline(xintercept = mu, size = 3)  
 g <- g + ggtitle(paste("mu = ", mu, ", MSE = ", round(mse, 2), sep = ""))  
 g  
}  
manipulate(myHist(mu),mu=slider(62,74,step=0.5))

fig xxx

Because we’re using manipulate we can move the slider around and monitor the value of and the mean squared error, that is the sum of the squared distances between the observed data points and that particular value of . If you move the slider around, you would notice notice as we get toward the center of the histogram, the mean squared error is going down and if you keep moving the slider way up, it get’s up large again. You can see is the point that balanced out this histogram.

**Notice** For those that are interested, we cover some simple proofs of some of the statements made. If this isn’t your thing, just skip these sections. However, if you’re interested, get a pencil and paper to work along!

The equations above show for any value of , the function is larger than or equal to the specific case when we plug in . Therefore, has to be the unique minimizer of that equation.

At this stage, we haven’t utilized the parent’s heights in our analysis. The initial step in examining this type of data is to construct a scatter plot of child heights against parent heights. Here we employ ggplot, but the plot has several shortcomings.

ggplot(galton, aes(x = parent, y = child)) + geom\_point()



Notably, there’s over-plotting due to numerous parent-child pairs sharing the same x, y values. To address this, we provide an improved plot where the point size reflects the number of parent-child combinations at a specific x, y location. Additionally, color indicates frequency, with lighter colors representing higher frequencies.

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:reshape':  
##   
## rename

## The following objects are masked from 'package:Hmisc':  
##   
## src, summarize

## The following object is masked from 'package:MASS':  
##   
## select

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

freqData <- as.data.frame(table(galton$child, galton$parent))  
names(freqData) <- c("child", "parent", "freq")  
freqData$child <- as.numeric(as.character(freqData$child))  
freqData$parent <- as.numeric(as.character(freqData$parent))  
g <- ggplot(filter(freqData, freq > 0), aes(x = parent, y = child))  
g <- g + scale\_size(range = c(2, 20), guide = "none" )  
g <- g + geom\_point(colour="grey50", aes(size = freq+20, show\_guide = FALSE))

## Warning: Ignoring unknown aesthetics: show\_guide

g <- g + geom\_point(aes(colour=freq, size = freq))  
g <- g + scale\_colour\_gradient(low = "lightblue", high="white")   
g



In order to find the best line, all we have to find is the slope. Well, here’s how we could potentially do that. We would want to find the slope beta that minimizes the sum of the squared distances between the observed data points the and the fitted data points on the line, . We’ll square that distance and add them up and this is directly analogous to finding the least squares mean. This is sort of using the origin as a pivot point and picking the line that minimizes the sum of the squared vertical distances between the points and the line. Notice that there is a point in regression to the origin is useful for explaining things, because we only have one parameter, the slope and we don’t have two parameters, the slope and the intercept. But it’s generally bad practice to force regression lines through the point (0, 0). So, an easy way around this is to subtract the mean from the parent’s heights and the mean from the child’s heights, so that the zero, zero point is right in the middle of the data and that will make this solution a little bit more palatable.

y <- galton$child - mean(galton$child)  
x <- galton$parent - mean(galton$parent)  
freqData <- as.data.frame(table(x, y))  
names(freqData) <- c("child", "parent", "freq")  
freqData$child <- as.numeric(as.character(freqData$child))  
freqData$parent <- as.numeric(as.character(freqData$parent))  
myPlot <- function(beta){  
 g <- ggplot(filter(freqData, freq > 0), aes(x = parent, y = child))  
 g <- g + scale\_size(range = c(2, 20), guide = "none" )  
 g <- g + geom\_point(colour="grey50", aes(size = freq+20, show\_guide = FALSE))  
 g <- g + geom\_point(aes(colour=freq, size = freq))  
 g <- g + scale\_colour\_gradient(low = "lightblue", high="white")   
 g <- g + geom\_abline(intercept = 0, slope = beta, size = 3)  
 mse <- mean( (y - beta \* x) ^2 )  
 g <- g + ggtitle(paste("beta = ", beta, "mse = ", round(mse, 3)))  
 g  
}  
  
myPlot(0.5)

## Warning: Ignoring unknown aesthetics: show\_guide



We can find the slope of the line very quickly in R using the lm function. The lm function stands for linear model. We’re going to regress the child’s height on the parent’s height. We’re going to subtract the mean from the child’s height and the mean from the parent’s height, to make sure line is going through the origin. Doing so will give us a line that has slope of 0.646.

lm(I(child - mean(child))~ I(parent - mean(parent)) - 1, data = galton)

##   
## Call:  
## lm(formula = I(child - mean(child)) ~ I(parent - mean(parent)) -   
## 1, data = galton)  
##   
## Coefficients:  
## I(parent - mean(parent))   
## 0.6463

Now what we’re going to do in subsequent sections is to talk about how we get these values? What is the motivation behind it and all the things we can do with this fitted line, we’re going to spend maybe the next several sections talking about this. You have actually learned a lot of material in this very first part, well done!

## 1.3 Linear least squares

### 1.3.1 Notations and background

Ordinary least squares (OLS) is the workhorse of statistics. It gives a way of taking complicated outcomes and explaining behavior (such as trends) using linearity. The simplest application of OLS is fitting a line through some data. In the next few sections, we cover the basics of linear least squares. We start with defining our notation. These are things you probably already saw in the prerequisite for this course in a Statistical Inference course in Data Science Specialization. However, because they’re so fundamental to regression, we’re going to cover them again, so they’re fresh in our minds. We will try to minimize the amount of mathematics that’s required for this class. Throughout the course we will neither require calculus nor linear algebra. And when it does get a little bit more mathematical, we will let you know when you can skip over those sections.

We might write to describe data points. As an example, consider the data set , where , , and in this case is 3. There’s nothing in particular about the letter . We could have just as easily described to . The last bit of notation that’s important, is we’re typically going to use Greek letters for things we don’t know, such as for a population mean and we’ll use non Greek letters or regular letters to denote things that we can observe. So, is something we can observe. is something we can’t observe and would like to estimate. We can define the empirical mean as

Notice if we subtract the mean from data points, we get data that has mean 0. That is, if we define The mean of the is 0.

* This process is called “centering” the random variables. Recall from the previous section that the mean is the least squares solution for minimizing .

Since we talked about means, let’s talk about variances. The variances is usually denoted by . It’s defined as

This is nothing other than basically the average squared deviation of the observations around the mean. The empirical standard deviation is defined as . Notice that the standard deviation has the same units as the data. It’s nice to work with standard deviations because the variance is expressed in whatever units has squared, whereas the standard deviation is just expressed in the normal units of . Another interesting fact related to standard deviation is scaling, so if we subtract a mean off from every observation, we get a resulting data set that has mean 0. If we divide every observation by the standard deviation, the resulting data set will have standard deviation 1. This is called **scaling** the data. If we take our original data now and subtract off , then take the resulting centered data and scale it by . We get a new data set, let’s call them .

This process of centering and then scaling is called *normalizing* the data. As an example, if something has a value 2 from normalized data, that means that the data point was 2 standard deviations larger than the mean. As its name would suggest, normalization is an attempt to make non-comparable data sets comparable.

The empirical covariance is the most central quantity in regression. Imagine we have two vectors, and , and they’re lined up. So might be the BMI and might be the blood pressure for subject . You could meaningfully do a scatter plot. Then we just define the covariance between X and Y as:

The correlation is defined as:

where and are the estimates of standard deviations for the observations and observations, respectively. In other words, the correlation is simply the covariance then standardized into a unitless quantity. So, the correlation is the covariance of and , which has units, basically units of X times units of Y.

Some facts about correlation: \* \* \* and only when the or observations fall perfectly on a positive or negative sloped line, respectively. \* measures the strength of the linear relationship between the and data, with stronger relationships as heads towards -1 or 1. \* implies no linear relationship.

### 1.3.2 Linear Least Squares

Consider again, when we’re looking at the scatter plot of the parent’s heights by the child’s heights from the Galton data, the size of the circle represents the frequency of that particular x, y combination.

## Warning: Ignoring unknown aesthetics: show\_guide



We’d like to use the parent’s heights to explain the child’s heights and we’re going to do it using linear regression. We’re going to use our notation that we developed in our last section. So let’s let be the child’s height and be the parents’ height. Now we want to find the best line, where we want the line to look like child’s height is an intercept. Child’s Height = + Parent’s Height , and are parameters we would like to know that we don’t know. Well, we need a criteria for the term best. We need to figure out what we mean by the best line that fits the data. Well, one criteria is the famous least squares criteria. And the basic gist of the equation is we want to minimize the sum of the squared vertical distances between the data points, the height of the data points, the child’s heights and the points on the line, on the fitted line. And we can write this as

This is the sum of the squared vertical distances between the data points and the fitted line. We want to minimize this quantity. We want to find the and that minimize this quantity. This is called the least squares criteria. We put little hats over and to indicate the estimated values. The least squares model fit to the line through the data pairs with as the outcome obtains the line where

The solution works out to be is the correlation between and times the standard deviation of divided by the standard deviation of . The estimated intercept . So let’s go through a couple of consequences of this being the result.

* has the units of , has the units of . We can see this because the correlation is a unitless quantity.
* The line passes through the point )
* The slope of the regression line with as the outcome and as the predictor is .
* The slope is the same one you would get if you centered the data, , and did regression through the origin.
* If you normalized the data, , the slope is .

### 1.3.3 Linear Least Squares Coding Example

Here we will go through a coding example to show how to calculate the least squares estimates. We plot the Galton parents’ height and childrens’ height data that we are going to look at.

library(dplyr)  
freqData <- as.data.frame(table(galton$child, galton$parent))  
names(freqData) <- c("child", "parent", "freq")  
freqData$child <- as.numeric(as.character(freqData$child))  
freqData$parent <- as.numeric(as.character(freqData$parent))  
g <- ggplot(filter(freqData, freq > 0), aes(x = parent, y = child))  
g <- g + scale\_size(range = c(2, 20), guide = "none" )  
g <- g + geom\_point(colour="grey50", aes(size = freq+20, show\_guide = FALSE))

## Warning: Ignoring unknown aesthetics: show\_guide

g <- g + geom\_point(aes(colour=freq, size = freq))  
g <- g + scale\_colour\_gradient(low = "lightblue", high="white")   
g

 Now we indicate that the solution that we specified is the same solution that R will give you with its built in regression function. The function lm in R stands for linear model. Regression is a component of linear models, and so, this function is the general function whether you want regression or you want some of the more elaborate versions of regression that we’re going to cover later on. So we want lm, the outcome , the predictor . coef takes the output of the linear model and just grabs the coefficients.

beta1 <- cor(y, x) \* sd(x) / sd(y)  
beta0 <- mean(x) - beta1 \* mean(y)  
rbind(c(beta0, beta1), coef(lm(x ~ y)))

## (Intercept) y  
## [1,] 8.207028e-16 0.3256475  
## [2,] 1.258492e-15 0.3256475

As we expected you see we get the same numbers, 23.94 and 0.64, 0.65. Very briefly now, we just want to mention that if we reverse the and relationship the formula, of course holds but now with standard deviation of in the numerator and standard deviation of in the denominator.

beta1 <- cor(y, x) \* sd(x) / sd(y)  
beta0 <- mean(x) - beta1 \* mean(y)  
rbind(c(beta0, beta1), coef(lm(x ~ y)))

## (Intercept) y  
## [1,] 8.207028e-16 0.3256475  
## [2,] 1.258492e-15 0.3256475

If we concatenate these slope and intercept estimates with those that you get with lm where is on the left hand side of the ~ and is on the right hand side of ~, reversed from what it was previously.

So our formula is correct and we know how to use it and we know what happens when we reverse the relationship. Another point that was made thus far in the course was that regression through the origin yielded the same slope as linear regression with a not necessarily zero intercept. If you mean centered the ’s and mean centered the ’s first. So let’s just check that computationally. Recall that the regression to the origin equation for the slope was just the sum of the variable times the variable, divided by the sum of the variable squared. So, let’s run that and get our coefficient that is estimated through a regression to the origin.

yc <- y - mean(y)  
xc <- x - mean(x)  
beta1 <- sum(yc \* xc) / sum(xc ^ 2)  
c(beta1, coef(lm(y ~ x))[2])

## x   
## 0.6462906 0.6462906

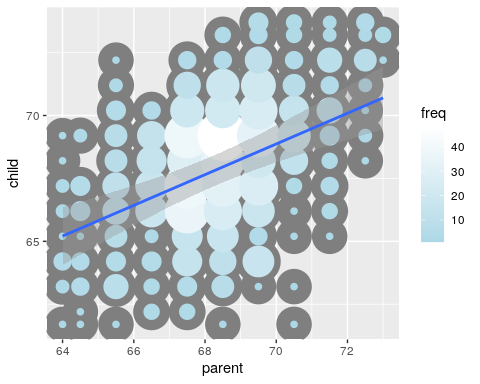
We want to very briefly also just show you how you can actually do regression to the origin. In this case I’ll get the same number if I take the centered and use the centered as a predictor, to subtract out the intercept, you put a minus one to get rid of the intercept.

Another point that was made before, was that if we were to normalize the or the so that they have standard deviation one, the slope would be the correlation. So let’s just double check that quickly. Here, We normalize the child’s heights by subtracting off the mean and dividing by the standard deviation. We do the same thing for variables. We have gotten rid of the, the original units, the inches.

yn <- (y - mean(y))/sd(y)  
xn <- (x - mean(x))/sd(x)  
c(cor(y, x), cor(yn, xn), coef(lm(yn ~ xn))[2])

## xn   
## 0.4587624 0.4587624 0.4587624

## Warning: Ignoring unknown aesthetics: show\_guide



Here we are showing the somewhat fancy plot for this data. We would also note that ggplot2 does a very good thing for us on our behalf. It automatically gives us a confidence interval around the line. We’ll talk about how to generate this confidence interval later on in the lecture. But it’s very nice that they’re thinking of statistical uncertainty automatically.

### 1.3.4 Mathematical Details (Optional) XXX

## 1.4 Regression to the Mean

Regression to the mean was an important milestone in the discovery of regression. So we’re going to talk about it. It was discovered by Francis Galton. Regression to mean asks questions like this.

* Why is it that the children of tall parents tend to be tall, but not as tall as their parents?
* Why do children of short parents tend to be short, but not as short as their parents?
* Why do parents of very short children, tend to be short, but not a short as their child? And the same with parents of very tall children?

We can try this with anything that is measured with error. Why do the best performing athletes this year tend to do a little worse the following? Why do the best performers on hard exams always do a little worse on the next hard exam?

These phenomena are all examples of so-called regression to the mean. Regression to the mean, was invented by Francis Galton in the paper “Regression towards mediocrity in hereditary stature” The Journal of the Anthropological Institute of Great Britain and Ireland , Vol. 15, (1886). The idea served as a foundation for the discovery of linear regression.

Regression to the mean often comes up in sports. If you have a player who has a phenomenal year, the next year they tend to do a little bit worse. If you have a player who has a terrible year, the next year they tend to do a little bit better. Another example would be often people talk about stocks in the same way. Some of the best performing stocks tend to go down. These phenomena could all be examples of so called regression to the mean. We will talk about why these happen and whether or not something is intrinsic or whether it is a regression to the mean effect. Regression to the mean was invented by Francis Galton. We like to think of regression to the mean by thinking of the case where it’s a 100% regression to the mean. So imagine if we were to simulate pairs of standard normals, i.e. they have nothing to do with one another, they’re independent standard normals. If we were to take the largest one, the chance that its pair in the second vector is smaller will be high. And this is simply saying that the probability that is less than , given is going to get bigger as heads to very large values. The same thing in other words, is that probability is greater than . Given that equals is going to get bigger as heads to smaller values. This extreme version of regression in the mean where there’s 100% regression to the mean is what we like to think about.

* gets bigger as heads into the very large values.
* gets bigger as heads to very small values.

However, in most cases there’s some blend of some, some intrinsic component, and a noise. For example, consider a scenario where every student in this class takes two very challenging quizzes. While those at the top likely have a better understanding of the material, quizzes are imperfect instruments, introducing inherent error or noise. This means that even the top performers might benefit from some luck or randomness. Consequently, a top performer, who probably knows the material a bit better than others, may experience a slight dip in performance on the second quiz due to this inherent variability. Conversely, even the worst performers might fare a bit better on one quiz due to chance. This concept extends beyond academics. It’s intriguing to reflect on how much of the discussion about sports revolves around the idea of regression to the mean. For instance, a baseball player with a phenomenal batting average one year might experience a slightly lower average the next year, illustrating the natural tendency for extreme performances to move closer to the average over time. The question is are these examples of just regression to the mean? If so, it would be nice to figure out how to quantify it. This is what Francis Galton did with regression in the first treatment of regression to the mean.

Let’s delve into how Francis Galton employed the concept of regression, particularly using correlation, which is intimately related to linear regression. The goal is to quantify regression to the mean, and I’ll illustrate this with a visual representation. Before delving into the R code, let me outline the setup.

In this case, I’m assigning to be the child’s height and to be the parent’s height. I’m using a dataset where the parent is a single parent, specifically the father. Both the and values have been normalized, meaning they have a mean of 0 and a variance of 1. Assuming you’re familiar with this normalization process, the regression line will pass through the point (0, 0). Notably, regardless of whether the child’s height is the outcome or the parent’s height is the outcome, the slope of the regression line is simply the correlation.

Now, a quirk worth mentioning when creating the plot is that if is the outcome and you happen to plot it on the horizontal axis, the slope of the line needs to be 1 over the correlation. This is due to the specific orientation of the axes. Keep this in mind as we proceed with the *R* code. In the code below we are using the dataset from the usingR library, specifically the father.son data. Here’s how we define the variables:

* Y: Son’s heights, normalized by subtracting the mean and dividing by the standard deviation.
* X: Father’s heights, similarly normalized.

Now, both X and Y should have a mean of 0 and a variance of 1.

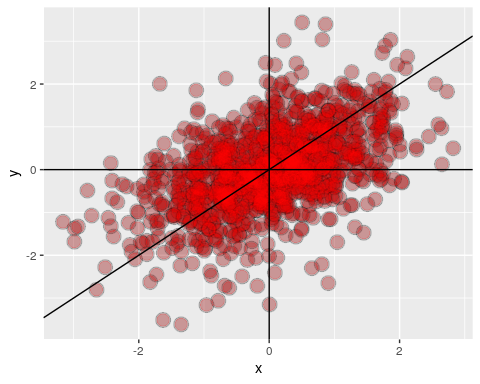
We use the Greek letter (rho) to represent the correlation between and . If you would check the value of rho, turns out to be about 0.5. This indicates a correlation of 0.5 between the father’s height and the son’s height.

Now, let’s create the plot. After loading the ggplot2 library, we assign the ggplot to the variable g and adding points with a black background and salmon-colored foreground. The use of alpha blending makes the points somewhat transparent. We set the x-axis and y-axis limits to be -4 to +4 on both axes. This range is chosen as it should cover most of the data, considering the extremely low probability of standardized random variables being below -4 or above +4. Chebyshev’s theorem supports this choice, especially if you’ve covered it in the Statistical Inference course. Next, we add a layer for the identity line. Afterward, we’ll add the horizontal and vertical axes.

library(UsingR)  
data(father.son)  
y <- (father.son$sheight - mean(father.son$sheight)) / sd(father.son$sheight)  
x <- (father.son$fheight - mean(father.son$fheight)) / sd(father.son$fheight)  
rho <- cor(x, y)  
library(ggplot2)  
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))  
g = g + geom\_point(size = 6, colour = "black", alpha = 0.2)  
g = g + geom\_point(size = 4, colour = "salmon", alpha = 0.2)  
g = g + xlim(-4, 4) + ylim(-4, 4)  
g = g + geom\_abline(intercept = 0, slope = 1)  
g = g + geom\_vline(xintercept = 0)  
g = g + geom\_hline(yintercept = 0)  
g = g + geom\_abline(intercept = 0, slope = rho, size = 2)  
g = g + geom\_abline(intercept = 0, slope = 1 / rho, size = 2)  
g = ggplot(data.frame(x, y), aes(x = x, y = y))  
g = g + geom\_point(size = 5, alpha = .2, colour = "black")  
g = g + geom\_point(size = 4, alpha = .2, colour = "red")  
g = g + geom\_vline(xintercept = 0)  
g = g + geom\_hline(yintercept = 0)  
g = g + geom\_abline(position = "identity")

## Warning: Ignoring unknown parameters: position

g

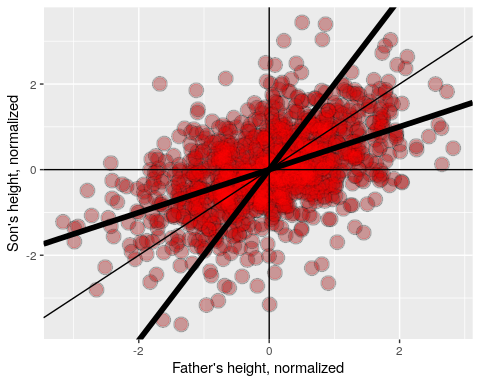


Now, let’s create two lines. First, we’ll treat the son’s height as the outcome and the father’s height as the predictor. Then, we’ll add the line treating the son’s height as the predictor and the father’s height as the outcome. Since the axes are rotated, the slope needs to be 1 over rho.

library(UsingR)  
data(father.son)  
y <- (father.son$sheight - mean(father.son$sheight)) / sd(father.son$sheight)  
x <- (father.son$fheight - mean(father.son$fheight)) / sd(father.son$fheight)  
rho <- cor(x, y)  
library(ggplot2)  
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))  
g = g + geom\_point(size = 6, colour = "black", alpha = 0.2)  
g = g + geom\_point(size = 4, colour = "salmon", alpha = 0.2)  
g = g + xlim(-4, 4) + ylim(-4, 4)  
g = g + geom\_abline(intercept = 0, slope = 1)  
g = g + geom\_vline(xintercept = 0)  
g = g + geom\_hline(yintercept = 0)  
g = g + geom\_abline(intercept = 0, slope = rho, size = 2)  
g = g + geom\_abline(intercept = 0, slope = 1 / rho, size = 2)  
g = ggplot(data.frame(x, y), aes(x = x, y = y))  
g = g + geom\_point(size = 5, alpha = .2, colour = "black")  
g = g + geom\_point(size = 4, alpha = .2, colour = "red")  
g = g + geom\_vline(xintercept = 0)  
g = g + geom\_hline(yintercept = 0)  
g = g + geom\_abline(position = "identity")

## Warning: Ignoring unknown parameters: position

g = g + geom\_abline(intercept = 0, slope = rho, size = 2)  
g = g + geom\_abline(intercept = 0, slope = 1 / rho, size = 2)  
g = g + xlab("Father's height, normalized")  
g = g + ylab("Son's height, normalized")  
g



Now, let’s discuss regression to the mean in relation to this plot. If the observations perfectly aligned on a line, it would be the identity line, given that both and have been normalized. The father’s height is plotted as the variable, and the son’s height is plotted as the variable. For instance, if we had a father’s height of 2 with no noise, the prediction for the son’s height would also be 2, representing 2 standard deviations above the mean for both fathers and sons.

However, in the presence of noise, the prediction deviates from 2 but falls on the regression line. This prediction is obtained by multiplying the father’s height (=2) by the slope (=correlation). The result is a prediction between 2 and 0, precisely 2 multiplied by the correlation. This phenomenon is known as regression to the mean. The extent to which this correlation is shrunk towards the horizontal line indicates the degree of regression to the mean.

Consider the extreme cases for better understanding. In a scenario with no noise, the line would fall perfectly on the identity line. Conversely, if there was only noise, indicating no informative relationship between father’s and son’s heights (correlation = 0), the line would lie on the horizontal axis, predicting a constant height of 0 for sons based on fathers. This concept holds when considering the son’s height as the predictor and the father’s height as the outcome. The regression to the mean is observed in how much the line is shrunk towards the vertical axis. This notion, introduced by Francis Galton, played a pivotal role in the development of modern regression. Although it remains a fundamental idea, regression to the mean continues to have significance in statistical analyses, particularly in the study of longitudinal data where it’s crucial to consider this phenomenon.

In summary: \* If you had to predict a son’s normalized height, it would be \* If you had to predict a father’s normalized height, it would be \* Multiplication by this correlation shrinks toward 0 (regression toward the mean) \* If the correlation is 1 there is no regression to the mean (if father’s height perfectly determine’s child’s height and vice versa) \* Note, regression to the mean has been thought about quite a bit and generalized

## 1.5 Practical R Exercises in swirl

During this course we’ll be using the [swirl](https://swirlstats.com) software package for R in order to illustrate some key concepts. The swirl package turns the R console into an interactive learning environment. Using swirl will also give you the opportunity to construct and explore your own regression models.

1. Install R

swirl requires R 3.0.2 or later. If you have an older version of R, please update before going any further. If you’re not sure what version of R you have, type R.version.string at the R prompt. You can download the latest version of R from <https://www.r-project.org/>.

Optional but highly recommended: Install RStudio. You can download the latest version of RStudio at <https://www.rstudio.com/products/rstudio/>.

1. Install swirl

Since swirl is an R package, you can easily install it by entering a single command from the R console:

* + If you are on a Linux operating system, please visit our Installing swirl on Linux page for special instructions: install.packages("swirl")
  + If you’ve installed swirl in the past make sure you have version 2.2.21 or later. You can check this with: packageVersion("swirl")

1. Load swirl

Every time you want to use swirl, you need to first load the package. From the R console: library(swirl).

1. Install the Regression Models course

swirl offers a variety of interactive courses, but for our purposes, you want the one called Regression Models. If this is your first time using swirl, it will prompt you to install the Regression Models course automatically. If you’ve used swirl in the past, you will need to type the following from the R prompt: install\_course("Regression Models").

1. Start swirl and complete the lessons

Type the following from the R console to start swirl:

For the first part of this course you should complete the following lessons: - Introduction - Residuals - Least Squares Estimation

Good luck and have fun!

## 1.6 Week 1 Quiz

1. Consider the data set given by the R code x <- c(0.18, -1.54, 0.42, 0.95)and weights given by w <- c(2, 1, 3, 1) give the value of that minimizes the least squares equation .
2. Consider the following data set fit the regression through the origin and get the slope treating yas the outcome and x as the regressor. (Hint, do not center the data since we want regression through the origin, not through the means of the data.)

x <- c(0.8, 0.47, 0.51, 0.73, 0.36, 0.58, 0.57, 0.85, 0.44, 0.42)  
y <- c(1.39, 0.72, 1.55, 0.48, 1.19, -1.59, 1.23, -0.65, 1.49, 0.05)

1. Do data(mtcars) from the datasets package and fit the regression model with mpg as the outcome and weight as the predictor. What is the slope coefficient?
2. Consider data with an outcome () and a predictor (). The standard deviation of the predictor is one half that of the outcome. The correlation between the two variables is .5. What value would the slope coefficient for the regression model with as the outcome and as the predictor?
3. Students were given two hard tests and scores were normalized to have empirical mean 0 and variance 1. The correlation between the scores on the two tests was 0.4. What would be the expected score on Quiz 2 for a student who had a normalized score of 1.5 on Quiz 1?
4. Consider the data given by x <- c(8.58, 10.46, 9.01, 9.64, 8.86). What is the value of the first measurement if x were normalized (to have mean 0 and variance 1)?
5. Consider the following data set (used above as well). What is the intercept for fitting the model with x as the predictor and y as the outcome?

x <- c(0.8, 0.47, 0.51, 0.73, 0.36, 0.58, 0.57, 0.85, 0.44, 0.42)  
y <- c(1.39, 0.72, 1.55, 0.48, 1.19, -1.59, 1.23, -0.65, 1.49, 0.05)

1. You know that both the predictor and response have mean 0. What can be said about the intercept when you fit a linear regression?
2. It must be identically 0.
3. It is undefined as you have to divide by zero.
4. It must be exactly one.
5. Nothing about the intercept can be said from the information given.
6. Consider the data given by x <- c(0.8, 0.47, 0.51, 0.73, 0.36, 0.58, 0.57, 0.85, 0.44, 0.42). What value minimizes the sum of the squared distances between these points and itself?
7. Let the slope having fit as the outcome and as the predictor be denoted as . Let the slope from fitting as the outcome and as the predictor be denoted as . Suppose that you divide by ; in other words consider . What is this ratio always equal to?

# 2 Week 02

## 2.1 Statistical linear regression models

Up to this point, we’ve only considered estimation. Estimation is useful, but we also need to know how to extend our estimates to a population. This is the process of statistical inference. Our approach to statistical inference will be through a statistical model. At the bare minimum, we need a few distributional assumptions on the errors. However, we’ll focus on full model assumptions under Gaussianity.

### 2.1.1 Statistical Linear Regression Models

Finding a good regression line using least squares is a mathematical procedure. However, we’d like to do statistics. We’d like to draw emphasis based on our data. In other words we’d like to generalize from our data to a population using statistical models. Consider the probabilistic model for linear regression

The values of are the population parameters that we would like to estimate. is a collection of explanatory variables that we do know, and is iid Gaussian errors.

* Here the are assumed iid .

Understanding independent errors in regression can be approached in various ways. One relatively straightforward interpretation is to consider them as the cumulative effect of unmodeled variables that might collectively influence the response. These unmodeled variables act on the response in a manner that can be statistically modeled as independent and identically distributed Gaussian errors.

Setting aside the complexities of interpretation, let’s focus on the mechanics of working with statistical inference for regression. It’s important to note that the expected value of the response given a specific value of the regressor is simply the line at that regressor, represented as . Additionally, the variance of the response at any given value of the regressor is denoted as σ². It’s crucial to clarify that this variance pertains to the variation around the regression line and not the overall response variance. Conditioning on X reduces the variation, making it lower than the unconditional response variance.

* Note,
* Note, .

Both the expected value and variance mentioned here are population quantities. Although there are sample analogs that estimate these values, it’s essential to recognize that, at this point, we are referring to population quantities—these are the estimands that we ideally want to know.

Now that we have a formal statistical framework, we can interpret our regression coefficients with respect to that framework. Take for example, the intercept. It is the expected value given that the regressor is 0.

Note that the regressor being equal to zero is often not of interest in the study. For example, if the regression variable is blood pressure, probably you’re not interested in the response for among people with blood pressure of zero. However, there is an easy fix for this. Consider just shifting our regression variable by a constant .

We see a new regression line with a new intercept and the same slope. So, shifting your values by value changes the intercept, but not the slope.

* Often is set to so that the intercept is interpretted as the expected response at the average value.

For slope, we can interpret it as the expected change in response for a 1 unit change in the predictor.

\* Consider the impact of changing the units of .

We see a new regression line with a new slope and the same intercept. So, multiplying your values by value changes the slope, but not the intercept. For example, is height in and is weight in . Then is . Converting to implies multiplying by . To get in the right units, we have to divide by to get it to have the right units.

If we would like to guess the outcome at a particular value of the predictor, say , the regression model guesses

This doesn’t mean that we can only predict at the fitted values. We can predict at any value of by plugging in the value of into the equation. However, we’re going to have more reasonable predictions if the value of that we plug in is in the cloud of data that we used to build the model. Later on, we’ll also talk about how to account for that kind of uncertainty with prediction intervals. But for the time being, let’s just talk about how we get a prediction. Let’s go through an example to interpret the regression coefficients and show running of the regression coefficient. The dataset is the diamond dataset from the UsingR package. The data is diamond prices in Singapore dollars and diamond weight in carats, which is a standard measure of diamond mass.

library(UsingR)

## Loading required package: MASS

## Warning: package 'MASS' was built under R version 4.0.3

## Loading required package: HistData

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##   
## Attaching package: 'Hmisc'

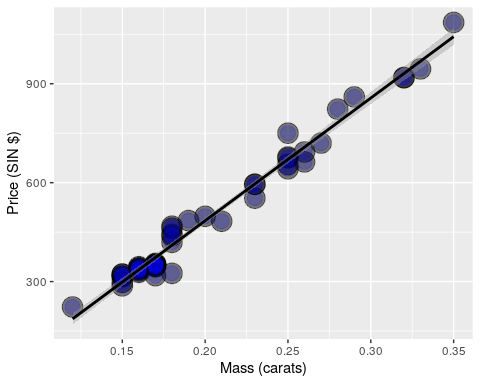
## The following objects are masked from 'package:base':  
##   
## format.pval, units

##   
## Attaching package: 'UsingR'

## The following object is masked from 'package:survival':  
##   
## cancer

data(diamond)  
library(ggplot2)  
g = ggplot(diamond, aes(x = carat, y = price))  
g = g + xlab("Mass (carats)")  
g = g + ylab("Price (SIN $)")  
g = g + geom\_point(size = 7, colour = "black", alpha=0.5)  
g = g + geom\_point(size = 5, colour = "blue", alpha=0.2)  
g = g + geom\_smooth(method = "lm", colour = "black")  
g

## `geom\_smooth()` using formula 'y ~ x'



In this code we assign variable g to the ggplot, the dataset is diamond, the aesthetic has the horizontal axis variable as carat and the y-axis variable as price, we add a layer where the xlab is Mass in carats and the y label price in Singapore dollars. We also add the points of the black background and then a light alpha blending color on top. Afterwards we add a layer that is geom\_smooth where method = "lm" will add the regression line. If you omit any arguments, it’s just going to assume the regression line with as the outcome and as the predictor. Finally, we indicate the color of the regression line as black and call the plot.

Notice what we are plotting is the fitted line, the line that minimizes the sum of the squared vertical distances between the points and the lines. By default, lm includes an intercept, if you don’t want an intercept, you have to explicitly force it in the model. We also want the dataset to be the diamond dataset in other words, we have to give it the data frame. Otherwise, lm looks in the regular R environment for variables in the model. After running the code it basically just prints out the coefficients , which are the intercept and labels it as Intercept and the regression variable for the carat, the slope for the carat regression variable.

fit <- lm(price ~ carat, data = diamond)  
coef(fit)

## (Intercept) carat   
## -259.6259 3721.0249

Let’s look at this variable and try to interpret it. It’s saying that we have an expected Singapore dollar increase in price for every carat increase in mass of the diamond. The intercept, is the expected price of a carat diamond not very interesting, because we’re not interested in zero carat diamonds.

**A side note**, if you want a much more detailed printout by doing summary(fit) which is the summary of the outputted variable from lm and you get this more elaborate printout.

##   
## Call:  
## lm(formula = price ~ carat, data = diamond)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -85.159 -21.448 -0.869 18.972 79.370   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -259.63 17.32 -14.99 <2e-16 \*\*\*  
## carat 3721.02 81.79 45.50 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 31.84 on 46 degrees of freedom  
## Multiple R-squared: 0.9783, Adjusted R-squared: 0.9778   
## F-statistic: 2070 on 1 and 46 DF, p-value: < 2.2e-16

If we mean center our variable, so that the intercept is on a more interpretable scale. Here we assign the output to a different variable, fit2 instead of fit, because we don’t want to overwrite the original fit.

fit2 <- lm(price ~ I(carat - mean(carat)), data = diamond)  
coef(fit2)

## (Intercept) I(carat - mean(carat))   
## 500.0833 3721.0249

As you notice in code: lm is again the linear model procedure, the outcome stays the same and we use carat - mean(carat), and the I is to indicate that we want to do arithmetic on the variable. So, we want to subtract the mean of the carat variable from the carat variable. This is a way to mean center the variable. As we expected the slope stays the same, 3,721, but the intercept has changed to , meaning , Singapore dollars is the expected price of the average sized diamond. In this case, the average diamond is about 0.2 carats. A one carat increase is actually kind of big. What about changing the units to one-tenth of a carat? We can do this just by dividing the coefficient by ten. So we know that we would expect to see a increase in price for every of a carat increase in the mass of a diamond.

fit3 <- lm(price ~ I(carat/10), data = diamond)  
coef(fit3)

## (Intercept) I(carat/10)   
## -259.6259 37210.2485

In the linear model fit instead of putting in carat, we put in , the units of this new variable is one-tenth of a carat. The data is of course, still the diamond dataset.

Imagine if someone came to you with three new diamonds that they had 0.16 carats, 0.27 carats and 0.35 carats, and they wanted to know what you would estimate the price would be. Well, you could do it manually by grabbing the two coefficients in multiplying the intercept or adding the intercept plus the slope times these new values. Let’s do that:

newx <- c(0.16, 0.27, 0.34)  
coef(fit)[1] + coef(fit)[2] \* newx

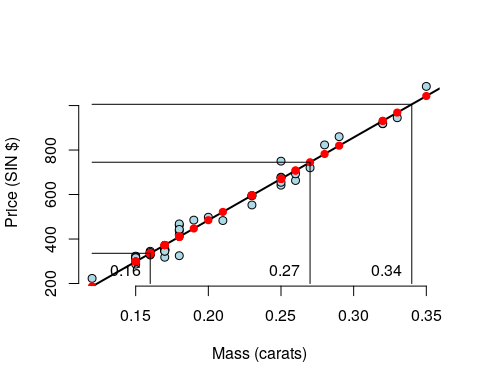
## [1] 335.7381 745.0508 1005.5225

predict(fit, newdata = data.frame(carat = newx))

## 1 2 3   
## 335.7381 745.0508 1005.5225

Often, you don’t want to do even that much coding, you want to more general method, especially when you get lots of regression variables. So there’s this general method called predict that will take the output from several different kinds of model fits. Linear models are one example, but predict is a generic function, and it applies to several different prediction models. The new data is a data.frame(catar=newx) that has the new values of for the carat variable. Then when we do that, what you’ll see is the same answer. The difference is that it scales up when we have lots of regressors in much more complicated settings. In general, we want to predict using the predict function. If you omit this new data statement if you just do predict fit, it predicts at the observed values, so it gives you the values. If you want it at new values, you have to give it this new data argument.

data(diamond)  
plot(diamond$carat, diamond$price,   
 xlab = "Mass (carats)",   
 ylab = "Price (SIN $)",   
 bg = "lightblue",   
 col = "black", cex = 1.1, pch = 21,frame = FALSE)  
abline(fit, lwd = 2)  
points(diamond$carat, predict(fit), pch = 19, col = "red")  
lines(c(0.16, 0.16, 0.12),   
 c(200, coef(fit)[1] + coef(fit)[2] \* 0.16,  
 coef(fit)[1] + coef(fit)[2] \* 0.16))  
lines(c(0.27, 0.27, 0.12),   
 c(200, coef(fit)[1] + coef(fit)[2] \* 0.27,  
 coef(fit)[1] + coef(fit)[2] \* 0.27))  
lines(c(0.34, 0.34, 0.12),   
 c(200, coef(fit)[1] + coef(fit)[2] \* 0.34,  
 coef(fit)[1] + coef(fit)[2] \* 0.34))  
text(newx, rep(250, 3), labels = newx, pos = 2)



To illustrate, here’s our observe data points in blue. The fitted values when we do the predict command, the fitted values in red all of the observed values and their associated fitted points on the line. These are if we were to draw vertical lines from the observed data points on to the fitted line, they would occur on these red points. When we predicted a new value of , we’re finding a point along this horizontal axis. In this example we want, 0.16, 0.27 and 0.34. We’re drawing a line up to the fitted regression line and then over to dollars and those are our predicted dollar amounts.

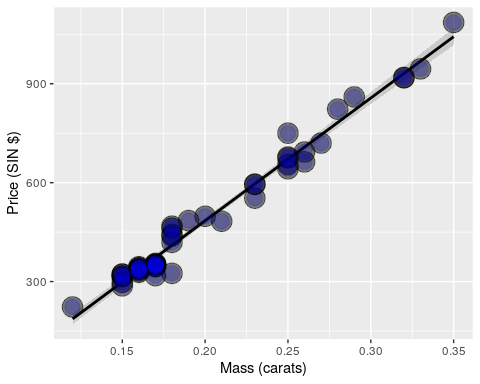
## 2.2 Residuals

Residuals represent variation left unexplained by our model. We emphasize the difference between residuals and errors. The errors unobservable true errors from the known coefficients, while residuals are the observable errors from the estimated coefficients. In a sense, the residuals are estimates of the errors.

To begin, let’s delve into our illustrative example featuring the diamond dataset. It’s important to recall that in this dataset, the diamonds are priced in Singapore dollars. The key variable under consideration is the weight of the diamonds, expressed in carats. Our objective is to explore the correlation between the weight of diamonds and their corresponding prices, seeking to understand how variations in diamond prices can be elucidated by their mass.

library(UsingR)  
data(diamond)  
library(ggplot2)  
g = ggplot(diamond, aes(x = carat, y = price))  
g = g + xlab("Mass (carats)")  
g = g + ylab("Price (SIN $)")  
g = g + geom\_smooth(method = "lm", colour = "black")  
g = g + geom\_point(size = 7, colour = "black", alpha=0.5)  
g = g + geom\_point(size = 5, colour = "blue", alpha=0.2)  
g

## `geom\_smooth()` using formula 'y ~ x'



Now, our focus is on elucidating the price (on the vertical axis) through the mass (on the horizontal axis). Without taking mass into account, we’d have a scatter of points projecting onto the vertical axis, displaying considerable variation. Disregarding mass would result in a notable amount of unexplained variation. However, when we factor in mass, the variation diminishes, as we’re now examining the variation around the regression line.

This remaining variation around the regression line is termed residual variation. It represents the portion of variation that persists even after accounting for mass. Initially, there is substantial variation, a significant portion of which is clarified by the linear relationship with mass. Nonetheless, there remains some residual variation. These residual distances are referred to as residuals, and they constitute the focal point of today’s lecture. Residuals prove to be valuable for various diagnostic purposes, including assessing model fit. Let’s refresh our memory regarding the model under consideration.

The outcome in our example,price, is , which we’re assuming is a line. Observed outcome is at predictor value , predicted outcome is at predictor value is . Residual, the between the observed and predicted outcome , which is the vertical distance between the observed data point and the regression line where least squares minimizes . In essence, it was minimizing the sum of the squared residual, summation squared. One way to think about the residuals are as an estimate of , though, you have to be careful with that, because as we will see later on, we can decrease the residuals just by adding irrelevant regressors into the equation. Let’s talk about some aspects of residuals that will help us interpret them.

* . (Their population’s expected value is zero.)
* If an intercept is included, (Their empirical sum, hence the empirical mean also, is zero if you include an intercept.If you don’t include an intercept, this property doesn’t have to hold.)
* If a regressor variable, , is included in the model . (The generalization of this property is, if you include any regression term in linear regression, the sum of the residuals times that regression variable has to be zero.)
* Residuals are useful for investigating poor model fit. (We can create plots that highlight the aspects of poor model fit.)
* Positive residuals are above the line, negative residuals are below.
* Residuals can be thought of as the outcome () with the linear association of the predictor () removed. (A common use of residuals is to think of them as the outcome with the linear influence of the predictor having been removed. For example, if we wanted to in some subsequent model or some subsequent analysis diamond prices, but in a way that has already been adjusted for their weight, calibrating all the diamond prices to be on the same scale regardless of their weight, we would take those residuals from the model fit that has diamond prices as the outcome, and weight as the predictor.)
* One differentiates residual variation (variation after removing the predictor) from systematic variation (variation explained by the regression model). (It’s very common to take residuals and carry them forward in a later analysis where you want to think of them as the, the new outcome, having removed the predictor at that point. But, remember with linear regression, you’re only removing the linear component of the predictor. One should differentiate between residual variation, which is variation that is left over after the explanatory variable has been accounted for in a linear fashion, from systematic variation, which is variation explained by the regression model. Again, residual plots can highlight poor model fit. And, we are going to go through some residual plots.)
* Residual plots highlight poor model fit.

Let’s walk through calculating residuals in this example we’re going to use the diamond dataset.

data(diamond)  
y <- diamond$price; x <- diamond$carat; n <- length(y)  
fit <- lm(y ~ x)  
e <- resid(fit)  
yhat <- predict(fit)  
max(abs(e -(y - yhat)))

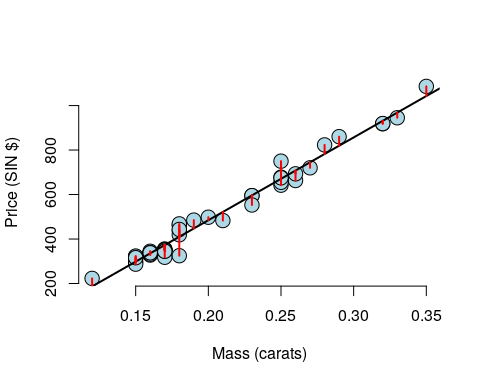
## [1] 8.242296e-13

max(abs(e - (y - coef(fit)[1] - coef(fit)[2] \* x)))

## [1] 8.242296e-13

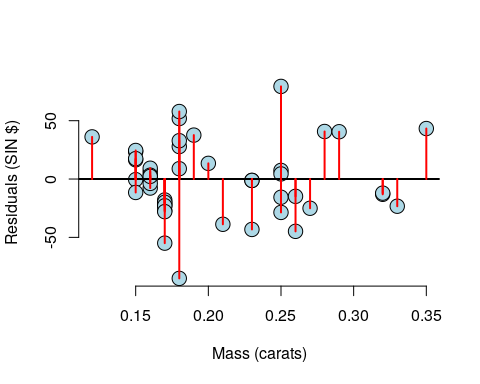
We redefine price as y and x as carat, n as the length of the number of pairs. We assign the linear regression object from lm to variable fit. To get the residuals we resid(fit), and we assign that to e. We also get the fitted values by predict(fit) and assign that to yhat. We can check that the residuals are the difference between the observed outcome and the predicted outcome. We can also check that the residuals are the difference between the observed outcome and the intercept plus the slope times the predictor. To show you that residual’s calculated via resid() functions are the same as the residuals that we calculate manually we take the absolute difference between y - yhat and e and find the one is on the scale of i.e, up to numerical precision, it’s the same thing. Then lastly, we want to show that the residuals are the difference between the observed outcome and the intercept plus the slope times the predictor, again up to numeric precision, exactly the same. To obtain the residuals, the preferred method is to use resid(). However, by demonstrating an alternative code, we aim to shed light on the underlying process of “res” and the specific computation performed by resid(). Ultimately, we would like to demonstrate that the total sum of the residuals equals zero. Technically, it’s , which is sufficiently close to zero. Additionally, the sum of the residuals multiplied by the price variable `x`` must also be zero—albeit at . Therefore, in numerical terms, both cases effectively amount to zero. These residuals represent the magnitudes of the deviations depicted by the red line in the accompanying plot.

plot(diamond$carat, diamond$price,   
 xlab = "Mass (carats)",   
 ylab = "Price (SIN $)",   
 bg = "lightblue",   
 col = "black", cex = 2, pch = 21,frame = FALSE)  
abline(fit, lwd = 2)  
for (i in 1 : n)   
 lines(c(x[i], x[i]), c(y[i], yhat[i]), col = "red" , lwd = 2)



Notice all of the blank space in the graph, making the plot kind of useless for that purpose, why don’t we plot the residuals on the vertical axis versus mass on the horizontal axis?

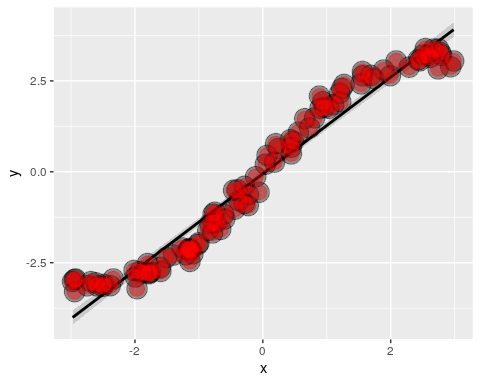
plot(x, e,   
 xlab = "Mass (carats)",   
 ylab = "Residuals (SIN $)",   
 bg = "lightblue",   
 col = "black", cex = 2, pch = 21,frame = FALSE)  
abline(h = 0, lwd = 2)  
for (i in 1 : n)   
 lines(c(x[i], x[i]), c(e[i], 0), col = "red" , lwd = 2)



Now we can see the residual variation much more clearly. One important point is: the residuals should be mostly patternless. Also, remember that if you include an intercept, residuals have to sum to zero. We can see some interesting patterns by honing in on the residual plot here. For example, we can see that there were lots of diamonds of exactly the same mass which gets lost in the scatter plot. Next, we want to go through some pathological residual plots, just to highlight what residual plots can do for us.

x = runif(100, -3, 3); y = x + sin(x) + rnorm(100, sd = .2);   
library(ggplot2)  
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))  
g = g + geom\_smooth(method = "lm", colour = "black")  
g = g + geom\_point(size = 7, colour = "black", alpha = 0.4)  
g = g + geom\_point(size = 5, colour = "red", alpha = 0.4)  
g

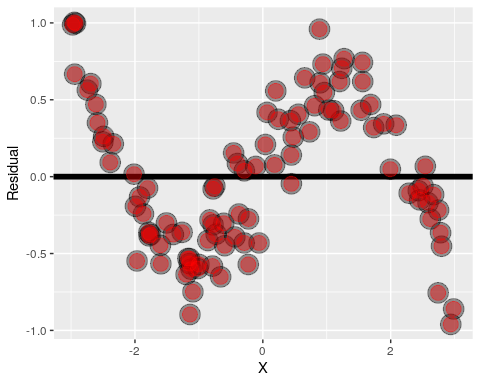
## `geom\_smooth()` using formula 'y ~ x'



Here X is just going to be uniform , y is equal to x, so it’s an identity line, but then we add another term that’s sin(x). This looks like an identity line, but kind of oscillating around it a little bit with some normal noise on top of it. Before we move on to the residual plot, let us make a comment. This model is actually not the correct model for this data and this might happen in practice. This doesn’t mean that this model is unimportant, right? There is a linear trend and the model is accounting for it, it’s just not accounting for the secondary variation in the sin term. To emphasize just because you aren’t fitting the actually correct model, that doesn’t mean the model is itself useless, in regression, having the exact right model is not always the primary goal. You can get meaningful information about trends from incorrect models.

Let’s me plot the residuals’ versus the x variable.

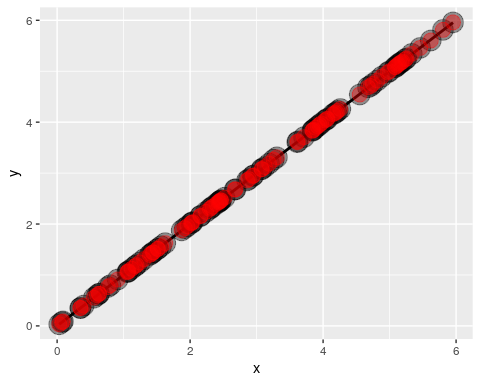
g = ggplot(data.frame(x = x, y = resid(lm(y ~ x))),   
 aes(x = x, y = y))  
g = g + geom\_hline(yintercept = 0, size = 2);   
g = g + geom\_point(size = 7, colour = "black", alpha = 0.4)  
g = g + geom\_point(size = 5, colour = "red", alpha = 0.4)  
g = g + xlab("X") + ylab("Residual")  
g



You can see that the sin term is now extremely apparent. This is what the residual plot has done highlighting the model inadequacy. Another example is the following plot, where by appearances, the plot falls perfectly on a line.

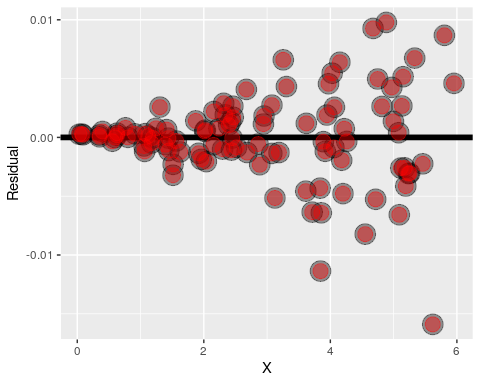
x <- runif(100, 0, 6); y <- x + rnorm(100, mean = 0, sd = .001 \* x);   
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))  
g = g + geom\_smooth(method = "lm", colour = "black")  
g = g + geom\_point(size = 7, colour = "black", alpha = 0.4)  
g = g + geom\_point(size = 5, colour = "red", alpha = 0.4)  
g

## `geom\_smooth()` using formula 'y ~ x'



But when you highlight the residuals, it looks quite different.

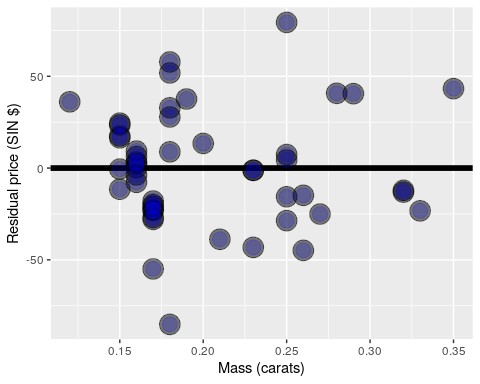
g = ggplot(data.frame(x = x, y = resid(lm(y ~ x))),   
 aes(x = x, y = y))  
g = g + geom\_hline(yintercept = 0, size = 2);   
g = g + geom\_point(size = 7, colour = "black", alpha = 0.4)  
g = g + geom\_point(size = 5, colour = "red", alpha = 0.4)  
g = g + xlab("X") + ylab("Residual")  
g



Plotting the residuals shows the trend toward greater variability as you head along the x variable. That property, where the variability increases with the x variables called heteroscedasticity. Heteroscedasticity is one of those things that residual plots are quite good at diagnosing and you couldn’t see it.

Let’s run the residual plot for the diamond data.

diamond$e <- resid(lm(price ~ carat, data = diamond))  
g = ggplot(diamond, aes(x = carat, y = e))  
g = g + xlab("Mass (carats)")  
g = g + ylab("Residual price (SIN $)")  
g = g + geom\_hline(yintercept = 0, size = 2)  
g = g + geom\_point(size = 7, colour = "black", alpha=0.5)  
g = g + geom\_point(size = 5, colour = "blue", alpha=0.2)  
g



The x-label is Mass in carats, the y-label is Residual price and just to emphasize the residuals have the same units as the ys. There doesn’t appear to be a lot of pattern in the plot, meaning it’s a pretty good fit.

Let us illustrate something about variability in a diamond dataset that will help us set the stage for defining some new properties about our regression model fit. So we create two residual vectors. The first residual vector is the one where we just fit an intercept, so the residuals are just the deviations around the average price. The second is the variation around the regression line with carats as the explanatory variable and price as the outcome. Then we create a factor variable that labels the set of residuals. The first one is labeled as a bunch of intercept only model residuals and the second set is labeled as a bunch of intercept and slope residuals.

e = c(resid(lm(price ~ 1, data = diamond)),  
 resid(lm(price ~ carat, data = diamond)))  
fit = factor(c(rep("Itc", nrow(diamond)),  
 rep("Itc, slope", nrow(diamond))))  
g = ggplot(data.frame(e = e, fit = fit), aes(y = e, x = fit, fill = fit))  
g = g + geom\_dotplot(binaxis = "y", size = 2, stackdir = "center", binwidth = 20)

## Warning: Ignoring unknown parameters: size

g = g + xlab("Fitting approach")  
g = g + ylab("Residual price")  
g



What we see on the left-hand plot with just the intercept is the variation in diamond prices around the average diamond price. What we’re seeing in the rightmost plot is displaying the variation around the regression line. So we have explained a lot of the variation with the relationship with mass. We’re going to talk about , which basically says, we can decompose the total variation, the variation explained by the regression model and the variation that’s left over after accounting for the regression model.

Residual variation is the variation around the regression line ( where ). The residuals are the vertical distances between the outcomes and the fitted regression line. If we include an intercept, the residuals have to sum to zero, which means their mean is zero. The variance of the residuals, is the average squared residual ( is ). Most people use , they instead of so that . The way to think about that is, we include the intercept the residuals have to sum to zero, that puts a constraint. If you know n minus one of them, then, you know the if you have a line term in there, if you have a co-variant in there, then, that puts a second constrain on the residuals. So, you lose two degrees of freedom. If you put another regression variable in there, you have another constraint, you lose three degrees of freedom. So in that sense you really don’t have n residuals, you have of them, because if you knew of them you could figure out the last two. And that’s why it’s one over . You can grab the residual variation out of the lm fit and assign it to a variable.

y <- diamond$price; x <- diamond$carat; n <- length(y)  
fit <- lm(y ~ x)  
summary(fit)$sigma

## [1] 31.84052

sqrt(sum(resid(fit)^2) / (n - 2))

## [1] 31.84052

If you want to grab it as an object that you can assign to something, just put dollar sign sigma. Then you can assign sigma to any other variable. The line sqrt(sum(resid(fit)^2) / (n - 2)) will result in the value and is showing what the lm function is doing behind the scenes.

Now let’s go back to the following plot where we look at the total variability in diamond prices, and compare what happens to the variability when we explain some of that variability with a regression line.

e = c(resid(lm(price ~ 1, data = diamond)),  
 resid(lm(price ~ carat, data = diamond)))  
fit = factor(c(rep("Itc", nrow(diamond)),  
 rep("Itc, slope", nrow(diamond))))  
g = ggplot(data.frame(e = e, fit = fit), aes(y = e, x = fit, fill = fit))  
g = g + geom\_dotplot(binaxis = "y", size = 2, stackdir = "center", binwidth = 20)

## Warning: Ignoring unknown parameters: size

g = g + xlab("Fitting approach")  
g = g + ylab("Residual price")  
g



The total variability is just the deviations of the data, the average squared deviation of the data around its mean. To make things easy, let’s forget about the denominator and just talk about the sum of the squared deviations. We might call the regression variability as the component of that variability that then gets explained away by the regression line. We would take the points on the regression line, the heights, which is the variability in the response and explained by the regression line, . The error variability is what’s leftover around the regression line . The interesting identity is that the total variability disregarding everything except for where they’re centered at is equal to the regression variability, that is the variability explained by the model plus the residual variability, the variability left over and not explained by the model.

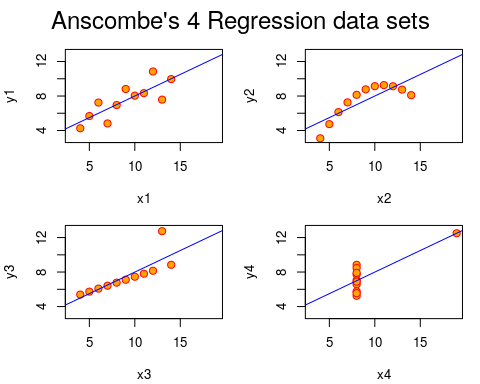
Because the residual variation and the regression model variation add up to the total variation we can define a quantity that represents the percentage of the total variation that’s represented by the model. This is called the coefficient of determination, . R squared is the percentage of the total variability that is explained by the linear relationship with the predictor

So R squared for our diamond example, is the percentage of the variation in diamond price, that is explained by the regression relationship with mass.

**Some facts about :**

* is the percentage of variation explained by the regression model.
* (because the regression variability and the error variability and the sums of the squares add up to the total sums of squares, and they are all positive)
* is the sample correlation squared. (If we define R as the sample correlation between the predictor and the outcome, then R squared is literally that sample correlation R, squared.)
* can be a misleading summary of model fit. (For example, if you have somewhat noisy data and delete a lot of the points in the middle you can get a much higher R squared. Or if you just add arbitrary regression variables into a linear model fit, you increase R squared and decrease mean squared error)
  + Deleting data can inflate .
  + (For later.) Adding terms to a regression model always increases .

Anscombe created a particularly stark example of a bunch of data sets with an equivalent R squared, equivalent mean, and variances in the x’s and the y’s, and identical regression relationships, but when you look at the scatter plots, you can see that the fit has very different meanings in each of the cases.

 The first is a nice regression line, exactly sort of along the lines of what we think of, when we think of just a slightly noisy x,y relationship. In the second one clearly there’s a missing term in order to address some of the curvature in the data. In the third one, there’s an outlier. Finally, in the fourth one, all the data stacked up at one particular location and there’s one point way out at the end. So you could imagine getting this if you had the first example and you deleted a lot of the points in the middle. In all these cases you have an equivalent R squared. But the summary to the single number certainly has thrown out a lot of the important information that you get from a simple scatter plot.

### 2.2.1 Optional reading How to derive R squared:

#### 2.2.1.1 Scratch work

#### 2.2.1.2 The relation between R squared and r

Recall that so that

Since, recall,

So, is literally squared.

## 2.3 Inference in regression

Inference is the process of drawing conclusions about a population using a sample. In statistical inference, we must account for the uncertainty in our estimates in a principled way. Hypothesis tests and confidence intervals are among the most common forms of statistical inference.

These statements apply generally, and, of course, to the regression setting that we’ve been studying. In the next few lectures, we’ll cover inference in regression where we make some Gaussian assumptions about the errors.

Before we begin talking about inference, let’s just revisit our model so that it’s fresh in our mind. where . For the time being, we’re going to assume that the true model is known, and this will be the basis for most of this class. We also assume that you’ve seen confidence intervals and hypothesis tests before. If you feel the need, you should go back and review them. Also, remember , .

We would like to review some of the basic concepts from statistical inference. Statistics like often have the following properties.

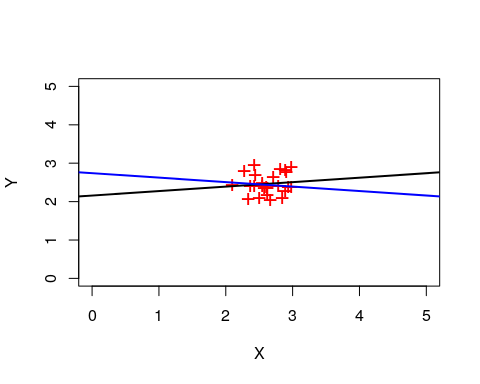
1. Is normally distributed and has a finite sample Student’s T distribution if the variance is replaced with a sample estimate (under normality assumptions).
2. Can be used to test versus .
3. Can be used to create a confidence interval for via where is the relevant quantile from either a normal or T distribution. (For example, if our is 5%, so we want a 95% confidence interval, we take the quantile.)

In the case of regression with iid sampling assumptions and normal errors, our inferences will follow very similarily to what you saw in your inference class. We won’t cover asymptotics for regression analysis, but suffice it to say that under assumptions on the ways in which the values are collected, the iid sampling model, and mean model, the normal results hold to create intervals and confidence intervals. In other words, it’s not mandatory for the errors to be Gaussian for our statistical inferences in regression to hold. You can appeal to large sample theory, though it’s a little bit more complicated.

The variance of our regression slope is actually a highly informative formula.

This is variance of , showing how variable the points are around the true regression line, , and how variable my X’s are. The numerator, how variable the points are around the regression line, is somewhat understandable as to why that would get better estimates of the regression slope if that were smaller. However, it’s maybe less intuitive to understand why we want more variance in our predictor in order to get lower variance in our regression slope. To understand it imagine a dataset where the regressors, the predictors, are all packed in very tightly, closely together, then it’s clear we’re not going to estimate a very good line. It could sort of bend around that cloud of points very easily and get equivalent fits.

# Generate a dataset with more random points around 2.5, 2.5  
set.seed(102)  
num\_points <- 25  
X <- runif(num\_points, min = 2, max = 3)  
Y <- runif(num\_points, min = 2, max = 3)  
  
# Plot the dataset  
plot(X, Y, xlim = c(0, 5), ylim = c(0, 5), xlab = "X", ylab = "Y", pch = "+", col = "red", cex=1.75)  
  
model <- lm(Y ~ I(X))  
abline(model, col = 'black',lwd = 2)  
  
model <- lm(Y ~ I(-X+5))  
abline(model, col = 'blue',lwd = 2)



On the other hand, if we spread our axis out, we will get a better fitted regression line with lower variance for the slope. It turns out the lowest you can make that variance is to push half the observations to one end and the other half of the observations to another end; however, you’re banking on having a line in between those two because you haven’t collected any data to evaluate that property.

The variance of the intercept, which is maybe a little less informative because intercepts are often a little less of interest than the slopes.

In practice, is replaced by its estimate. It’s probably not surprising that under iid Gaussian errors follows a distribution with degrees of freedom and a normal distribution for large . This can be used to create confidence intervals and perform hypothesis tests.

In the following example we demonstrate the formulas we are giving are exactly the formulas that R is using when it performs its calculations.

library(UsingR); data(diamond)  
y <- diamond$price; x <- diamond$carat; n <- length(y)  
beta1 <- cor(y, x) \* sd(y) / sd(x)  
beta0 <- mean(y) - beta1 \* mean(x)  
e <- y - beta0 - beta1 \* x  
sigma <- sqrt(sum(e^2) / (n-2))   
ssx <- sum((x - mean(x))^2)  
seBeta0 <- (1 / n + mean(x) ^ 2 / ssx) ^ .5 \* sigma   
seBeta1 <- sigma / sqrt(ssx)  
tBeta0 <- beta0 / seBeta0; tBeta1 <- beta1 / seBeta1  
pBeta0 <- 2 \* pt(abs(tBeta0), df = n - 2, lower.tail = FALSE)  
pBeta1 <- 2 \* pt(abs(tBeta1), df = n - 2, lower.tail = FALSE)  
coefTable <- rbind(c(beta0, seBeta0, tBeta0, pBeta0), c(beta1, seBeta1, tBeta1, pBeta1))  
colnames(coefTable) <- c("Estimate", "Std. Error", "t value", "P(>|t|)")  
rownames(coefTable) <- c("(Intercept)", "x")  
coefTable

## Estimate Std. Error t value P(>|t|)  
## (Intercept) -259.6259 17.31886 -14.99094 2.523271e-19  
## x 3721.0249 81.78588 45.49715 6.751260e-40

We again use the diamond dataset in the UsingR library. Let’s define the variables y, x, n like before, and . The residuals are response y minus the predicted values, . We create the two t-statistics if you’re testing a hypothesis that is zero or is zero, that is the estimate. Here’s the estimate divided by its standard error. We don’t have to subtract off the true value, because the true value is assumed to be zero under this hypothesis. Next we calculate the two p values. If you’ve taken the inference class, then you know how to go from a t-statistic to a p value. In next step, we create the coefficient table created manually without having done any lm or any built in higher level R function. We specify the row names and column names.

However, there is an easy way to do the same thing in R.

coefTable

## Estimate Std. Error t value P(>|t|)  
## (Intercept) -259.6259 17.31886 -14.99094 2.523271e-19  
## x 3721.0249 81.78588 45.49715 6.751260e-40

fit <- lm(y ~ x);   
summary(fit)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -259.6259 17.31886 -14.99094 2.523271e-19  
## x 3721.0249 81.78588 45.49715 6.751260e-40

You’ll see everything is exactly the same.

Next we want to get a confidence interval for the intercept and the slope.

sumCoef <- summary(fit)$coefficients  
sumCoef[1,1] + c(-1, 1) \* qt(.975, df = fit$df) \* sumCoef[1, 2]

## [1] -294.4870 -224.7649

(sumCoef[2,1] + c(-1, 1) \* qt(.975, df = fit$df) \* sumCoef[2, 2]) / 10

## [1] 355.6398 388.5651

Here we just need the table part of the summary, just the coefficient. With 95% confidence, we estimate that a 0.1 carat increase in diamond size results in a 355.6 to 388.6 increase in price in (Singapore) dollars, which is we estimate that a 0.1 carat increase in diamond size results in a 356 to 389 increase in price in Singapore dollars.

### 2.3.1 Prediction

Prediction is a central concept for the data scientist. In fact, we have an entire course, Practical Machine Learning on advanced prediction techniques. However, regression and generalized linear models which we will cover later on in the course are some of the most core techniques for performing prediction, they often produce very good predictions, they’re parsimonious and interpretable, and as an added bonus we can get inference on top of our predictions without doing any sort of data re sampling. By inference we mean you can get predictions, confidence intervals around the predictions to evaluate the uncertainty in those predictions, so that’s very easy in regression and pretty easy in generalized linear models and quite difficult in some more advanced machine learning algorithms, you may have to do data resampling or other techniques. We might want to predict a response, which might be the price of a diamond at a particular mass, in carats, or we might want to predict a child’s height for a particular value of the parent’s height. The obvious estimate in both cases is just take the , the predictor value multiply it by the relevant estimated slope, and then add the intercept.

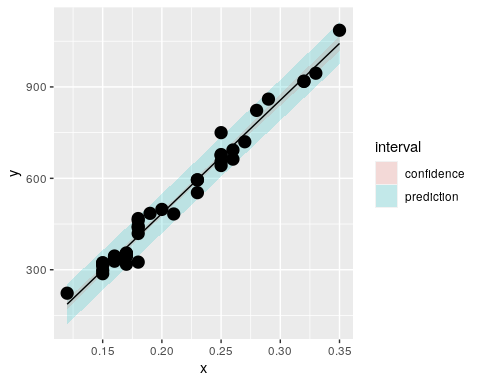
* The obvious estimate for prediction at point is

Being a good statisticians requires us to evaluate some uncertainty in the prediction, and it is nice to have a prediction interval. There’s a small intricacy between trying to predict a regression line at a particular point, and trying to predict a future at that same point. Those are two different ideas.

* Line at ,
* Prediction interval at ,

What we have here, and it makes sense that our prediction variance first relates around how variable the points are around our regression line, and we have the term that also kind of makes sense. Typically our standard errors decrease at some rate, . If we’re predicting a new , then we have the added 1 out front, so we get a wider interval. If we want to predict a new value at a specific point versus trying to predict what the regression line is at that point. We will talk more about that later, for now let’s focus on the very end term that on both equations: consider the numerator of this statistic our prediction error is going to be the lowest when not is equal to the prediction variance is smallest when we predict at the average mass of a diamond or at the average height of the parents. The denominator is basically how variable the Xs are. The more variable the Xs are, the smaller this term becomes and the lower the prediction error is. Like the slope estimate where the more variable the regressors were, the less variable the slope estimate was. The same thing happens in prediction error, and is an essential part of using regression for prediction, where we get easy and convenient prediction uncertainty associated with the parsimonious predictors.

library(ggplot2)  
newx = data.frame(x = seq(min(x), max(x), length = 100))  
p1 = data.frame(predict(fit, newdata= newx,interval = ("confidence")))  
p2 = data.frame(predict(fit, newdata = newx,interval = ("prediction")))  
p1$interval = "confidence"  
p2$interval = "prediction"  
p1$x = newx$x  
p2$x = newx$x  
dat = rbind(p1, p2)  
names(dat)[1] = "y"  
  
g = ggplot(dat, aes(x = x, y = y))  
g = g + geom\_ribbon(aes(ymin = lwr, ymax = upr, fill = interval), alpha = 0.2)   
g = g + geom\_line()  
g = g + geom\_point(data = data.frame(x = x, y=y), aes(x = x, y = y), size = 4)  
g



In predict function we provide the output of lm. For a lot of prediction algorithms, especially linear models and generalized linear models, random forests in R, the predict function is a generic method that applies to them, interval = ("confidence") indicates that we want the confidence interval, not a prediction interval, that’s R’s Code for creating the interval around the estimated line at that particular value of x not for a potential new y at that particular value of x, if we want an interval for potential new Y at that particular value of X, we change the interval = ("confidence") to interval = ("prediction"), as we do on the fourth line.

The blue is the prediction interval, this is for predicting a new line, and the salmon color is for prediction of the line at those particular values of x. The confidence interval is much narrower than the prediction interval. It is because of that 1 plus for the prediction interval.

Imagine if we collected an infinite amount of data at all different values of x along this line. Well, then, we would pretty much know the regression line exactly, if that were the case, we would be extremely confident about predictions on the line, where the line was at a particular x value. As we collected more and more data, that salmon colored confidence interval will get narrower and narrower around the line to the point where it was just the line itself. That’s what we would expect to happen. That’s just the idea of statistical sampling working. On the other hand, the prediction interval, there’s variability in the Ys, that has nothing to do with how well we estimated and in fact, if I were given the correct . There would still be variability in the Ys, because of the error term. Consequently, if we wanted to predict a new y there would be some uncertainty that would be inherent in that prediction. That’s why the prediction interval is always going to be wider than the confidence interval. It doesn’t go away with N. It doesn’t go away as we collect more X’s or anything like that. It’s inherent, and that’s why the prediction interval has a certain amount of width that’s never going to go away.

The last thing as you may notice both of the intervals get narrower toward the center of the data cloud and then get wider as you head out into the tails. That’s just simply saying that we’re more confident in our predictions closer to the mean of the X’s. Because of that one plus term in prediction intervals this phenomena is less obvious in blue color than the salmon one. If we were to go well beyond where we collected data, then these intervals would really become a lot wider which is what we’d want, because we would be extrapolating and we want to predict where we did not collect data.

**Summary**

* Both intervals have varying widths.
  + Least width at the mean of the Xs.
* We are quite confident in the regression line, so that interval is very narrow.
  + If we knew and this interval would have zero width.
* The prediction interval must incorporate the variabilibity in the data around the line.
  + Even if we knew and this interval would still have width.

## 2.4 For the project

You need to know a little bit of knitr. In this video, which you may have to refer back to when you start the project, will get you started on knitr.

In this section we will learn how to use knitr to create reproducible reports. We will also learn how to use R Markdown to create reproducible documents. You’ll need a little bit of knitr to create your R project. We open the go File> New File> R Markdown, this will populate a simple knitr document. Here we can run R commands in a code block, which is defined as three right tick marks followed by {r} if you insert a comma after the r you will open up a bunch of options, cache tells R whether or not to keep it, eval= tells whether or not it should evaluate the code and echo= where echo TRUE shows the code and echo FALSE does not show the code. Once you are done with the document click Knit HTML and it will knit and create an HTML document. That’s just a standard HTML document and you can bring up the document in a browser window. And that’s knitr in a nutshell.

## 2.5 Practical R Exercises in swirl

During this week of the course you should complete the following lessons in the Regression Models swirl course:

1. Residual Variation
2. Introduction to Multivariable Regression
3. MultiVar Examples

## 2.6 Week 2 Quiz

1. Consider the following data with x as the predictor and y as as the outcome. Give a P-value for the two sided hypothesis test of whether from a linear regression model is 0 or not.

x <- c(0.61, 0.93, 0.83, 0.35, 0.54, 0.16, 0.91, 0.62, 0.62)  
y <- c(0.67, 0.84, 0.6, 0.18, 0.85, 0.47, 1.1, 0.65, 0.36)

1. Consider the previous problem, give the estimate of the residual standard deviation.
2. In the mtcars data set, fit a linear regression model of weight (predictor) on mpg (outcome). Get a 95% confidence interval for the expected mpg at the average weight. What is the lower endpoint?
3. Refer to the previous question. Read the help file for mtcars. What is the weight coefficient interpreted as?
4. Consider again the mtcars data set and a linear regression model with mpg as predicted by weight (1,000 lbs). A new car is coming weighing 3000 pounds. Construct a 95% prediction interval for its mpg. What is the upper endpoint?
5. Consider again the mtcars data set and a linear regression model with mpg as predicted by weight (in 1,000 lbs). A “short” ton is defined as 2,000 lbs. Construct a 95% confidence interval for the expected change in mpg per 1 short ton increase in weight. Give the lower endpoint.
6. If my X from a linear regression is measured in centimeters and I convert it to meters what would happen to the slope coefficient?
7. I have an outcome, Y and a predictor, X and fit a linear regression model with to obtain . What would be the consequence to the subsequent slope and intercept if I were to refit the model with a new regressor, for some constant ?
8. Refer back to the mtcars data set with mpg as an outcome and weight (wt) as the predictor. About what is the ratio of the the sum of the squared errors, when comparing a model with just an intercept (denominator) to the model with the intercept and slope (numerator)?
9. Do the residuals always have to sum to 0 in linear regression?

## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)

# 3 Week 03

## 3.1 Multi-variable regression

We now extend linear regression so that our models can contain more variables. A natural first approach is to assume additive effects, basically extending our line to a plane, or generalized version of a plane as we add more variables. Multi-variable regression represents one of the most widely used and successful methods in statistics.

If you’re utilizing predictor X to forecast a response Y and discover a meaningful relationship, there is a potential issue if the predictor hasn’t been randomly assigned to the subjects or units being observed. In such cases, there is always a concern that there might be another variable, whether known or unknown, that could account for the observed relationship. For example, imagine if you had a friend who downloaded some data, where they had all sorts of health information from people and also their dietary information. This person claims to have found an interesting relationship: breath mint usage has a significant regression relationship with forced expiratory volume(FEV), a measure of lung function. You would be skeptical there is very little basis for a biological relationship there. Breath mints are just sugar! But maybe, but what you’ve really be thinking is what other variables might explain this relationship? You might have two hypotheses: this person dug through lots and lots of variables and just found the one that was significant, and it’s just a chance of association, which is the problem of multiplicity. In addition it is likely, you would think the real problem is smokers tend to use more breath mints, and smoking has this relationship with lung function. It’s well-established that chronic exposure to a smoker, even second-hand smoke has negative impacts on lung function. So it’s probably smoking it probably has nothing to do with the breath mints, it’s a indirect effect of breath mints through smoking, not a direct effect of breath mints on lung function. This would be the hypothesis. To establish that there is a breath mint effect beyond smoking we could consider smokers by themselves, and see whether their lung function differs by their breath mint usage, and consider non-smokers by themselves, and see whether their lung function differs by breath mint usage, where we conditioned on smoking status. This way we would compare like with like. Multivariable regression is sort of automated way to do that in a linear fashion. It makes fair enough assumptions, in automated way. In this section we will explain how it works and we will also talk a little bit about its limitations.

Multivariable regression is trying to look at the relationship of a predictor and a response, while having, at some level, accounted for other variables. Moreover, multivariable regression is actually a good prediction model. For example, a Kaggle competition wanted to predict the number of days a person would be in the hospital in subsequent years given their claims history and number of days they were in the hospital in previous years. The insurance companies seek to harness an extensive dataset derived from claims, aiming to predict a singular numerical outcome. However, the conventional approach of simple linear regression would be insufficient when confronted with multiple predictors. How can we extend the scope of simple linear regression to accommodate a multitude of regressors for predictive purposes? The procedure is similar to simple linear regression where there is more predictor terms, X values. For example, might be the number of insurance claims in the previous year, and might be whether or not the person had a particular cardiac problem, and so on. The first variable is typically just a constant one, so there is an intercept that’s included, a term that’s just by itself. Interestingly in this competition, we found that multivariable regression could get people very close to the winning entry, while other machine learning methods like random forest, and boosting only improved the results minorly on top of multivariable regression.

Note: in case of breath mint study, one of the predictors, might be breath mint usage (a binary variable), and might be how much a person smoked.

* The general linear model extends simple linear regression (SLR) by adding terms linearly into the model.
* Where typically, the are the coefficients of the model.
* Least squares (and hence ML estimates under iid Gaussianity of the errors) minimizes
* Note, the important linearity is linearity in the coefficients. Thus
* is still a linear model. (We’ve just squared the elements of the predictor variables.)

### 3.1.1 How to get the coefficients, derivation of formulas

Here we will go through the derivation of formulas to show how the least squares estimates are obtained. This derivation is not required for the course, but it may be helpful for those who are interested in understanding how the estimates are obtained.

Just to review, if you have regression to the origin, you want a line that’s forced to the origin that has no intercepts. You have the single predictor and a single predictor of and you want no intercept, . The slope estimate was . Now lets try to derive the least squares estimate when we have two regressors, which can be generalized to models with more variables. In , Least squares tries to minimize:

Here we try to give a development that is more intuitive than what you would get with something like linear algebra.

$$\Sum(y\_i - X\_{0i} \beta\_0 - X\_{1i} \beta\_1$$

Imagine we knew or fix , then we can write and subsequently $\Sum(\tilde y\_i - X\_{1i} \beta\_1$. This is exactly regression through the origin with just the single regressor. So we can write . Now we can plug this back into the original equation and we get:

This is an equation that only involves and a regression through the origin for . What it works out to be, and this is the interesting part, is that the regression slope for , is exactly what you would obtain if you took the residual of out of , and out of and then just did regression to the origin.

Multivariable regression calculates the coefficient for , , as if you had removed the effect of from both and . Similarly, the regression coefficient for , , is what you would get if you were to remove the effect of from both and . This is why multivariable regression is thought of as having adjusted for the other variables. A coefficient from a multivariable regression is the coefficient where the linear effect of all the other variables on that predictor and response has been removed.

### 3.1.2 Results

In , we have two covariates, .

is what you would get with regression through the origin if you removed the second coefficient . Similarly, the same thing could be said about the coefficient for . is the linear regression where linear effect of out of both the response , and the second predictor, . This is why multivariable regression relationships are considered as having been adjusted for all the other variables.

### 3.1.3 Example with two variables, simple linear regression

where is an intercept term. Notice the fitted coefficient of on is . The residuals are . Thus the fitted coefficient of on is , which is the residuals . We can write:

### 3.1.4 The general case

More generally, multivariate regression estimates are exactly those having removed the linear relationship of the other variables from both the regressor and response. Least squares solutions have to minimize

. The least squares estimate for the coefficient of a multivariate regression model is exactly regression through the origin with the linear relationships with the other regressors removed from both the regressor and outcome by taking residuals. In this sense, multivariate regression “adjusts” a coefficient for the linear impact of the other variables.

### 3.1.5 Examples with multiple-variables

In the following simulation we have 100 observations and want to generate three predictors, x, x2, x3, where they are all just standard normal. When we write y = 1 + x + x2 + x3, all my coefficients are 1, meaning the population model used for simulation, they’re all 1. Next we add some random noise, that’s the error term.

n = 100; x = rnorm(n); x2 = rnorm(n); x3 = rnorm(n)  
y = 1 + x + x2 + x3 + rnorm(n, sd = .1)  
ey = resid(lm(y ~ x2 + x3))  
ex = resid(lm(x ~ x2 + x3))  
sum(ey \* ex) / sum(ex ^ 2)  
coef(lm(ey ~ ex - 1))  
coef(lm(y ~ x + x2 + x3))

Here we want to point out, coef(lm(ey ~ ex - 1)) is the same coefficient as if we regress y on x, x2 and x3, and an intercept coef(lm(y ~ x + x2 + x3)). You see the x term here is exactly the same as the regression through the origin estimate with the residuals.

### 3.1.6 Interpretation of coefficients

The regression predictor, given the collection of covariants take a specific value, to , is just the sum of the .

If one of the predictors, say , is incremented by 1 i.e.  instead of takes , then the regression coefficient is the expected change in the response.

If we subtract the two terms the expected value of the response from the responce where the first co-efficient takes the value of works out to be .

Notice all the other to were held fixed, the interpretation of a multivariate regression coefficient is the expected change in the response per unit change in the regressor, holding all of the other regressors fixed.

The basic components of the linear models are exactly the same as in simple linear regression.

* Model where
* Fitted responses
* Residuals
* Variance estimate (note the degrees of freedom)
* To get predicted responses at new values, , simply plug them into the linear model
* Coefficients have standard errors, , and follows a distribution with degrees of freedom.
* Predicted responses have standard errors and we can calculate predicted and expected response intervals.

These should all be pretty familiar because they’re basically the same as what we did for linear aggression, the difference is we have more terms now. Remember in linear aggression we had two terms, we had an intercept and a covariant now we are just adding more covariants potentially.

One point to note is that the variance estimate is not quite the same as the average squared residuals. In linear regression we divided by , now we divide by . That’s kind of a technical point because if you know of the residuals you implicitly know the last of them due to some linear constraints. That’s a minor point you can think of the residuals variants estimate is nothing other than the average square residuals for the most part with part not withstanding.

In a sense all the things we knew about from linear regression carryover to multi-variable regression.

To end this section, we want to emphasize how important linear models are to the data scientist. Before you do any machine learning or any complex algorithm, linear models should be your first attempt. They offer parsimonious and well understood easily describe relationships between predictors and response. There are some modern machine learning algorithms that can beat some of the properties of linear models, like the imposed linearity. Nonetheless, linear models should always be your starting point. there is some amazing things you can do with linear models that you may not think that would be possible. For example, you can take a time series like a music sound or something like that, and decompose it into its harmonics. This is so-called discrete Fourier transform can be thought of the as the fit from a linear model. You can flexibly fit rather complicated functions and curves and things like that using linear models. You can fit factor variables as predictors. ANOVA and ANCOVA are special cases of linear models. You can uncover complex multivariate relationships within a response and you can build fairly accurate prediction models.

## 3.2 Multi-variable regression tips and tricks

Let’s start this discussion with the famous Swiss Fertility Data. Using the following command you can load the data and see the documentation.

require(datasets);   
data(swiss);   
?swiss

The data shows standardized fertility measure and socio-economic indicators for each of 47 French-speaking provinces of Switzerland at about 1888. A data frame with 47 observations on 6 variables, each of which is in percent, i.e., in [0, 100]. The variables are:

* [,1] Fertility a common standardized fertility measure
* [,2] Agriculture % of males involved in agriculture as occupation
* [,3] Examination % draftees receiving highest mark on army examination
* [,4] Education % education beyond primary school for draftees
* [,5] Catholic % catholic (as opposed to protestant)
* [,6] Infant.Mortality live births who live less than 1 year

All variables but Fertility give proportions of the population.

Visualizing some of the basic scatter plots is always a good practice.

library(GGally)

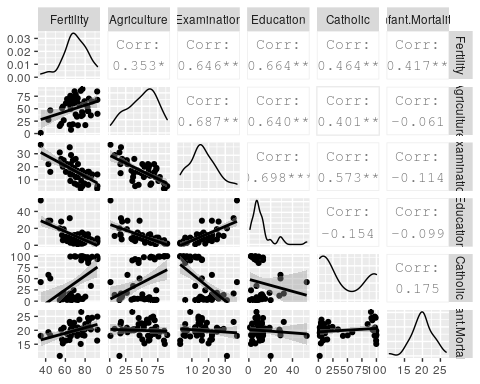
## Loading required package: ggplot2

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

g <- ggpairs(  
 swiss,  
 lower = list(continuous = "smooth"),  
 wrap = function(...) {  
 ggally\_smooth(..., method = "loess")  
 }  
)

## Warning in warn\_if\_args\_exist(list(...)): Extra arguments: 'wrap' are being  
## ignored. If these are meant to be aesthetics, submit them using the 'mapping'  
## variable within ggpairs with ggplot2::aes or ggplot2::aes\_string.

g



In this plot you see fertility is on the x-axis for all the plots in the first column, agriculture is on the x-axis for all of the plots in the second column. Agriculture is also on the y-axis for the first graph. In addition, the corresponding upper triangular part of the matrix gives the correlation between the two variables. For example, fertility and agriculture the relation turns out to be fairly linear with confidence prediction band around it. The correlation between the two is 0.35.

Let’s investigate the relationship where agriculture, the percent of the province that works in the agricultural industry, with fertility.

summary(lm(Fertility ~ . , data = swiss))$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 66.9151817 10.70603759 6.250229 1.906051e-07  
## Agriculture -0.1721140 0.07030392 -2.448142 1.872715e-02  
## Examination -0.2580082 0.25387820 -1.016268 3.154617e-01  
## Education -0.8709401 0.18302860 -4.758492 2.430605e-05  
## Catholic 0.1041153 0.03525785 2.952969 5.190079e-03  
## Infant.Mortality 1.0770481 0.38171965 2.821568 7.335715e-03

Tilde period in lm function is a shorthand for all the other variables in the data frame. The output of the summary function gives the coefficients of the model. The first column gives the estimated coefficients, the second column gives the standard errors of the coefficients, the third column gives the t-statistics, and the fourth column gives the p-values. The p-values are the probability of observing a t-statistic as extreme as the one observed, if the true coefficient were 0.

The number in the Agriculture variable row is interpreted as: we expect a 0.17 decrease, in standardized fertility for every 1% increase in the percentage of males involved in agriculture, holding the other variables constant. Meaning we hold examination and education, percent Catholic and infant mortality constant.

The next column, the standard error 0.07, talks about how precise that coefficient is. It talks about the statistical variability of that coefficient. If we wanted to perform a hypothesis test, we would take the estimate, subtract off the hypothesized value, which in this case is zero, and divide it by the standard error of the estimate. Which is the definition of T-statistic. R conveniently provides it to us, -2.448.

We can calculate the probability of getting a t-statistic as extreme as that. As small as negative 2.448 or smaller, and because we are doing a two-sided test, we would double that p-value. The degrees of freedom are $n - #coefficients$, including the intercept. But again, R does that on our behalf, and that works out to be 0.018. By standard thresholding rules, type one error rate of say 5%, that would be statistically significant.

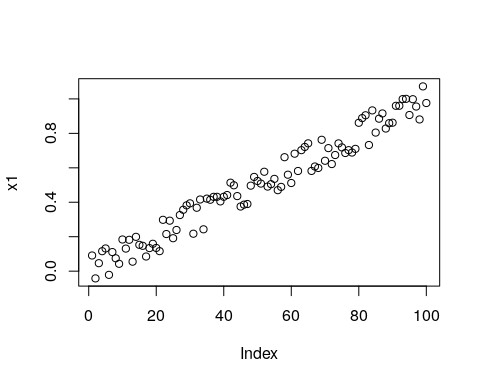
In the following section we will see how the process of model selection changes the estimates. We start by contrasting the model with a model that just has agriculture as predictor, the previous model had all the other variables in this predictor.

summary(lm(Fertility ~ Agriculture, data = swiss))$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 60.3043752 4.25125562 14.185074 3.216304e-18  
## Agriculture 0.1942017 0.07671176 2.531577 1.491720e-02

The agriculture variable is about the same magnitude, 0.19 instead of 0.17 but with changed signs. Instead of agriculture having a negative effect on fertility, it has a positive effect on fertility. Adjusting for the other variables changes the actual direction of the effect of agriculture on fertility. This is the impact of something so-called Simpson’s Paradox. Notice in both cases the agriculture coefficient is strongly statistically significant. We would like to create (via simulation) an example where an effect can reverse itself. It can help us understand Simpson’s paradox could happen. Keep in mind, regression is a dynamic process, where you have to think about what variables to include and why. If there hasn’t been randomization to protect you from confounding, you have to go through a scientific dynamic process of putting confounders in and out and thinking about what they’re doing to your effective interest in order to evaluate it.

n <- 100; x2 <- 1 : n; x1 <- .01 \* x2 + runif(n, -.1, .1); y = -x1 + x2 + rnorm(n, sd = .01)  
plot(x1)



summary(lm(y ~ x1))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 1.231687 1.278821 0.9631424 3.378466e-01  
## x1 96.206352 2.190330 43.9232303 2.801406e-66

summary(lm(y ~ x1 + x2))$coef

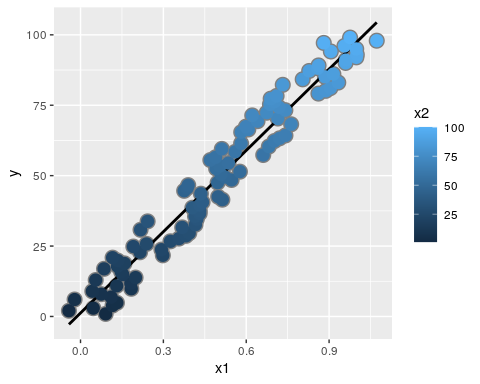
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -0.0001088336 0.0021104605 -0.05156863 9.589785e-01  
## x1 -0.9998822953 0.0165252233 -60.50643190 7.688898e-79  
## x2 0.9999810575 0.0001659207 6026.86147442 3.953311e-272

The second regressor, , is the values , is a variable that depends on and random noise. Think of as something we might measure regularly, like days, and as something like a saving account where the balance goes up with time and random fluctuations. The random fluctuations impact the spending, so the money doesn’t necessarily always just go up. It goes up and down sporadically, but the linear trend is going up. Let’s assume y is happiness with a measure like y = -x1 + x2 + noise. The true generating model y is negatively associated with -x1 suggesting happiness is negatively associated with money and positively associated with x2, so it goes up with time and down with x1 with some random normal noise. We know from the model y = -x1 + x2 + noise the outcome depends negatively on x1 with a coefficient of minus 1, and depends positively on x2 with a coefficient of plus 1. If fit x1 by itself we get an enormous coefficient, 95, which is clearly wrong. It’s nothing near to the negative 1 that it’s supposed to be or that we would hope it would be. It is picking up the residual effect of x2 that’s a big driver of y, but when we fit the correct model, x1 and x2, together we will get the correct coefficients, about minus 1 for x1, and about plus 1 for x2. You can imagine why this would happen by answering: what is regression doing? It’s taking x1 and removing the linear effect of x2.

Let’s do some plots to highlight this, just to show us how it works a little bit.

dat = data.frame(y = y, x1 = x1, x2 = x2, ey = resid(lm(y ~ x2)), ex1 = resid(lm(x1 ~ x2)))  
library(ggplot2)  
g = ggplot(dat, aes(y = y, x = x1, colour = x2))  
g = g + geom\_point(colour="grey50", size = 5) + geom\_smooth(method = lm, se = FALSE, colour = "black")   
g = g + geom\_point(size = 4)   
g

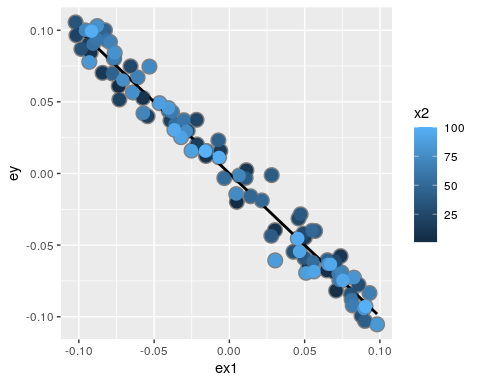
## `geom\_smooth()` using formula 'y ~ x'



There is a clear positive linear relationship, between the x1 and y. However, with x2, which is the color, there is also clear positive gradient. As y goes up, so does x2. And also you can see as x1 goes up, so does x2. So you can see the confounding that’s happening here.

g2 = ggplot(dat, aes(y = ey, x = ex1, colour = x2))   
g2 = g2 + geom\_point(colour="grey50", size = 5) + geom\_smooth(method = lm, se = FALSE, colour = "black") + geom\_point(size = 4)   
g2

## `geom\_smooth()` using formula 'y ~ x'



If we plot the residuals you can see that for the residual y and the residual x1, there is a clear negative linear relationship, and if you stare at it enough, you realize that the slope of this line should be around negative 1. You can also see that the x2 variable is clearly not related to the residual x1 variable.

It is important to remember the above explanation doesn’t mean that throwing every variable into your regression model is the right thing to do. there is consequences to throwing in unnecessary variables. It can make your model less interpretable, it can make your model less stable, and it can make your model less generalizable. It’s important to think about what variables you’re putting in and why.

In the earlier example about Swiss data the agriculture effect reversed itself after we included the other variables in the model. You will find that this happens quite a bit when education and examination are included. Educational attainment is negatively correlated with the percent working in agriculture, a correlation of -0.64. In addition, education and examination are kind of measuring the same thing. Their correlation, those two variables is 0.7. The percent of males in the province working in agriculture is negatively related to educational attainment (correlation of -0.6395225) and Education and Examination (correlation of 0.6984153) are obviously measuring similar things. The question is: is the positive marginal an artifact for not having accounted for, say, Education level? (Education does have a stronger effect, by the way.) At the minimum, anyone claiming that provinces that are more agricultural have higher fertility rates would immediately be open to criticism.

*Notice* What if we include an unnecessary variable? Here we introduce z which adds no new linear information, since it’s a linear combination of variables already included. R just drops terms that are linear combinations of other terms.

z <- swiss$Agriculture + swiss$Education  
lm(Fertility ~ . + z, data = swiss)

## Warning in terms.formula(formula, data = data): 'varlist' has changed (from  
## nvar=6) to new 7 after EncodeVars() -- should no longer happen!

##   
## Call:  
## lm(formula = Fertility ~ . + z, data = swiss)  
##   
## Coefficients:  
## (Intercept) Agriculture Examination Education   
## 66.9152 -0.1721 -0.2580 -0.8709   
## Catholic Infant.Mortality z   
## 0.1041 1.0770 NA

### 3.2.1 Dummy variables are smart

You might be surprised to find out how flexible linear regression models are. For example, you can fit factor variables as regressors and come up with things like analysis of variance as a special case of linear models.

Consider the linear model where each is binary so that it is a 1 if measurement is in a group and 0 otherwise. (Treated versus not in a clinical trial, for example.) The estimated mean for the treated group is the mean of the people who are treated. Then for people in the treated group we can write . is interpreted as the increase, or decrease if it’s negative, in the mean response for those that were treated. Similarly for people without treatment we have .

The LS fits work out to be is the mean for those in the group and is the mean for those not in the group. You see that linear regression provides the fitted values and tell you about the means for both of the groups, in addition it gives you an inference for comparing the two groups automatically.

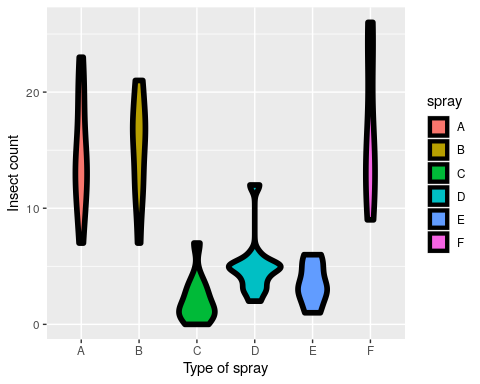
**Note** including a binary variable that is 1 for those not in the group would be redundant. It would create three parameters to describe two means.

We can generalize this to more than two groups. If we have a three-level variable, we can create two binary variables, one for each level, and then we can compare the means of the three groups. For example, imagine you have some outcome but you want to compare it to U.S. political party affiliation. In this case, let’s say you were only considering those who were Democrats, Republicans, or registered Independents. Well, you can do that by having a variable X1, that’s one for Republicans and zero for otherwise, a variable X2 that’s one for Democrats and zero for otherwise, we omit the X3 for Independents because of redundancy. If we know that you’re not Republican and not a Democrat, then you must be an Independent in our data set the way we’ve set things up and having a third variable wouldn’t have any new information.

The mean for the three groups are: \* If is Republican \* If is Democrat . \* If is Independent .

If we compare the means like the mean for the Independents versus the mean for the Republicans, i.e. subtract those two, we get . Which means compares Republicans to Independents, and similarly compares Democrats to Independents and compares Republicans to Democrats. By omitting the regression variable for the Independents, the intercept became the value for the Independents, and all of the other coefficients have become interpreted relative to Independents and shows choice of reference category changes the interpretation. If we had included the regressor for Independents and excluded the one for Republicans, then the intercept would be for Republicans, and the coefficient in front of the Democratic would be Democrats versus Republicans. The coefficient in front of the Independent would be Independent versus Republican. To illustrate how this works we move to R. In this example we look at a factor variable and see how R is treating it the dataset is InsectSprays and we are going to fit a linear model to it.

require(datasets);data(InsectSprays); require(stats); require(ggplot2)  
g = ggplot(data = InsectSprays, aes(y = count, x = spray, fill = spray))  
g = g + geom\_violin(colour = "black", size = 2)  
g = g + xlab("Type of spray") + ylab("Insect count")  
g



Y is the count, the number of insects, X is the spray. We use a violin plot to show the data, which is kind of like a histogram but sort of tilted on its side and repeated on both sides so it looks a little like a violin. It looks like a violin if you’re data cooperates, otherwise, it looks like a blob. As you can see there are eight spray A, B, C, D, E, and F and you can see the insect counts. It’s unfortunate they’re not telling us whether or not the count is the count of the number of alive or the number dead insects. So we don’t know if this is a better spray or a worse spray. However, we still can test the difference between different factor levels in this case using linear models. Wwhen we include insect spray as a linear model and y as an outcome.

summary(lm(count ~ spray, data = InsectSprays))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 14.5000000 1.132156 12.8074279 1.470512e-19  
## sprayB 0.8333333 1.601110 0.5204724 6.044761e-01  
## sprayC -12.4166667 1.601110 -7.7550382 7.266893e-11  
## sprayD -9.5833333 1.601110 -5.9854322 9.816910e-08  
## sprayE -11.0000000 1.601110 -6.8702352 2.753922e-09  
## sprayF 2.1666667 1.601110 1.3532281 1.805998e-01

We get the Intercept, spray B, spray C, spray D, spray E, and spray F, notice that spray A is conspicuously missing. The idea is that everything here is in comparison with spray A. So, 0.833 is the change in the mean between spray B and spray A. In this case, 14.5 is the mean for spray A. You can double check that by looking at the plot. Spray B seems reasonable affected by a little bit from spray A, whereas spray C looks like it’s affected a lot it has a coefficient of . If we wanted to compare spray B and spray C we would have to look at . We wouldn’t have a standard error for that comparison immediately. However, that would give us the estimate.

If we were to take the average count for the sprays, for those with spray A, we would get 14.5 with spray B we would get . What R does is it picks the spray level that’s the lowest alphanumerically, in this case, spray level A, to set as the reference level. Here we like to show you how you can hard code the same model and not rely on R to pick the reference level.

summary(lm(count ~   
 I(1 \* (spray == 'B')) + I(1 \* (spray == 'C')) +   
 I(1 \* (spray == 'D')) + I(1 \* (spray == 'E')) +  
 I(1 \* (spray == 'F'))  
 , data = InsectSprays))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 14.5000000 1.132156 12.8074279 1.470512e-19  
## I(1 \* (spray == "B")) 0.8333333 1.601110 0.5204724 6.044761e-01  
## I(1 \* (spray == "C")) -12.4166667 1.601110 -7.7550382 7.266893e-11  
## I(1 \* (spray == "D")) -9.5833333 1.601110 -5.9854322 9.816910e-08  
## I(1 \* (spray == "E")) -11.0000000 1.601110 -6.8702352 2.753922e-09  
## I(1 \* (spray == "F")) 2.1666667 1.601110 1.3532281 1.805998e-01

Here count is the outcome and we create a variable using the I function which performs the operation inside the regression, inside the model statement. We look at the instances where the spray is equal to B then multiply that by 1 to change it from Boolean to numeric and do the same for the other sprays and add them all together. In this example we included all of the sprays except A which means we forced A to be the reference level. The result is identical to R picking the reference level as we expected.

What happens if we include spray A?

summary(lm(count ~   
 I(1 \* (spray == 'B')) + I(1 \* (spray == 'C')) +   
 I(1 \* (spray == 'D')) + I(1 \* (spray == 'E')) +  
 I(1 \* (spray == 'F')) + I(1 \* (spray == 'A')), data = InsectSprays))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 14.5000000 1.132156 12.8074279 1.470512e-19  
## I(1 \* (spray == "B")) 0.8333333 1.601110 0.5204724 6.044761e-01  
## I(1 \* (spray == "C")) -12.4166667 1.601110 -7.7550382 7.266893e-11  
## I(1 \* (spray == "D")) -9.5833333 1.601110 -5.9854322 9.816910e-08  
## I(1 \* (spray == "E")) -11.0000000 1.601110 -6.8702352 2.753922e-09  
## I(1 \* (spray == "F")) 2.1666667 1.601110 1.3532281 1.805998e-01

Notice it gives an NA in front of the spray A coefficient. We have six means, for six sprays and seven parameters in intercept.

If we do want the coefficients, instead of being interpreted as levels referenced to a control level be the mean for each of the groups? Well, we can do that by removing the intercept.

summary(lm(count ~ spray - 1, data = InsectSprays))$coef

## Estimate Std. Error t value Pr(>|t|)  
## sprayA 14.500000 1.132156 12.807428 1.470512e-19  
## sprayB 15.333333 1.132156 13.543487 1.001994e-20  
## sprayC 2.083333 1.132156 1.840148 7.024334e-02  
## sprayD 4.916667 1.132156 4.342749 4.953047e-05  
## sprayE 3.500000 1.132156 3.091448 2.916794e-03  
## sprayF 16.666667 1.132156 14.721181 1.573471e-22

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

summarise(group\_by(InsectSprays, spray), mn = mean(count))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 6 × 2  
## spray mn  
## <fct> <dbl>  
## 1 A 14.5   
## 2 B 15.3   
## 3 C 2.08  
## 4 D 4.92  
## 5 E 3.5   
## 6 F 16.7

Here count is the outcome and spray is the predictor, but we remove the intercept. Notice we get a different set of coefficients, one for each spray level. It includes A, B, C, D, E and F without dropping any levels. It can do that because it has six parameters, and six means to work with. The coefficients are exactly equal to the means for each spray in the data. If calculated the means for each spray it would work out to be the same numbers.

We want to emphasize this model is no different than the previous model that included an intercept, and just the coefficients have a different interpretation. If we add these together, 14.5 and 0.833 from the model with intercept we should get the mean for spray B, which is the case. In the model with the intercept, the intercept is interpreted as the spray A mean and all the coefficients are interpreted as relative to spray A differences from spray A and when we fit the data without the intercept we get the mean for each spray. The p values are testing whether or not, A is different from B, and A is different from C, and A is different from D, and so on, whereas the p values from the model without intercept are testing whether or not those means are different from 0, which is a very different test. We were trying to illustrate how you play around with factor variables in lm is very important in terms of how you interpret it. It’s not just a conceptual or theoretical thing to worry about it is a very practical thing. What your intercept means changes dramatically depending on what your reference level is.

One more thing that we want to discuss is the idea of re-leveling. You can re-level to have differenct reference level. For example, if you want to compare spray C to spray A, you can re-level the spray variable to have C as the reference level.

spray2 <- relevel(InsectSprays$spray, "C")  
summary(lm(count ~ spray2, data = InsectSprays))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 2.083333 1.132156 1.840148 7.024334e-02  
## spray2A 12.416667 1.601110 7.755038 7.266893e-11  
## spray2B 13.250000 1.601110 8.275511 8.509776e-12  
## spray2D 2.833333 1.601110 1.769606 8.141205e-02  
## spray2E 1.416667 1.601110 0.884803 3.794750e-01  
## spray2F 14.583333 1.601110 9.108266 2.794343e-13

### 3.2.2 Summary of the InsectSprays example

* If we treat Spray as a factor, R includes an intercept and omits the alphabetically first level of the factor.
  + All t-tests are for comparisons of Sprays versus Spray A.
  + Empirical mean for A is the intercept.
  + Other group means are the itc plus their coefficient.
* If we omit an intercept, then it includes terms for all levels of the factor.
  + Group means are the coefficients.
  + Tests are tests of whether the groups are different than zero. (Are the expected counts zero for that spray.)
* If we want comparisons between, Spray B and C, say we could refit the model with C (or B) as the reference level.

**Note** We want to make a few points about the InsectSprays dataset, which we believe is important for the data scientist.

* Counts are bounded from below by 0, violates the assumption of normality of the errors.
  + Also there are counts near zero, so both the actual assumption and the intent of the assumption are violated.
* Variance does not appear to be constant.
* Perhaps taking logs of the counts would help.
  + There are 0 counts, so maybe log(Count + 1)
* Also, we’ll cover Poisson GLMs for fitting count data.

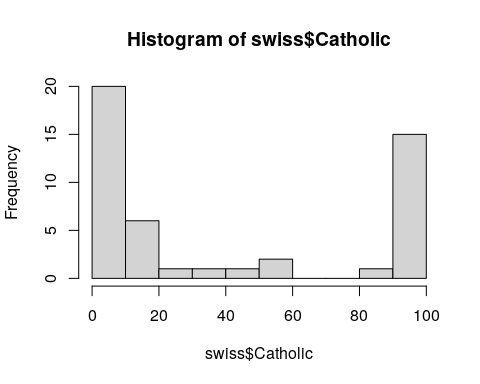
In this section we go through an example that underlies the topic of so called ANCOVA. In this example we will fit multiple lines with different intercepts and different slopes. We will use the swiss dataset, recall we are trying to model fertilities as a linear function of agriculture, which is the percent of that province that was working in agriculture.

library(datasets); data(swiss)  
head(swiss)

## Fertility Agriculture Examination Education Catholic  
## Courtelary 80.2 17.0 15 12 9.96  
## Delemont 83.1 45.1 6 9 84.84  
## Franches-Mnt 92.5 39.7 5 5 93.40  
## Moutier 85.8 36.5 12 7 33.77  
## Neuveville 76.9 43.5 17 15 5.16  
## Porrentruy 76.1 35.3 9 7 90.57  
## Infant.Mortality  
## Courtelary 22.2  
## Delemont 22.2  
## Franches-Mnt 20.2  
## Moutier 20.3  
## Neuveville 20.6  
## Porrentruy 26.6

If we do hist(swiss$Catholic), notice that it’s very bimodal, that’s because most provinces are either majority Catholic or majority Protestant, from this time period.

hist(swiss$Catholic)

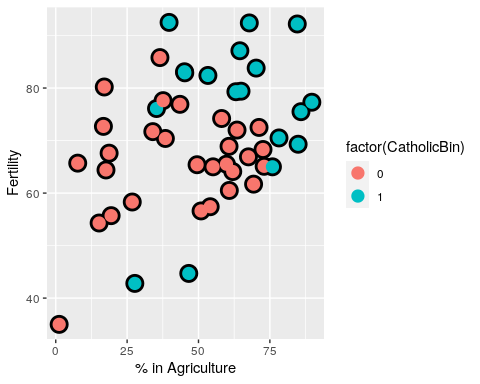


We now create a catholic, binary variable, which is one if the province is majority catholic, and zero if it’s majority protestant.

library(dplyr);   
swiss = mutate(swiss, CatholicBin = 1 \* (Catholic > 50))

We can plot the data to see the two groups, the CatholicBin factor variable, that’s zero for majority Protestant and one for majority Catholic.

g = ggplot(swiss, aes(x = Agriculture, y = Fertility, colour = factor(CatholicBin)))  
g = g + geom\_point(size = 6, colour = "black") + geom\_point(size = 4)  
g = g + xlab("% in Agriculture") + ylab("Fertility")  
g



For the time being, we will ignore the outlaiers and simply work on fitting a line where we want two separate lines. One for the majority Catholic provinces, and one for the majority Protestant provinces. For notations Y is fertility, X1 is the percent of the province working in agriculture, and X2 is a binary variable, where it is one if the province is over 50% catholic, and zero if the province is majority Protestant. Let’s consider model one, where we modeled the expected y, given x1 and x2, is an intercept plus a slope times x1. . This is the line that would disregard the religion of the province entirely. Let’s consider a second model . In the event that X two is equal to zero, i.e. if the province is majority protestant, this works out to be . In the event that x2 is equal to one, i.e. if the province is majority Catholic, this works out to be .

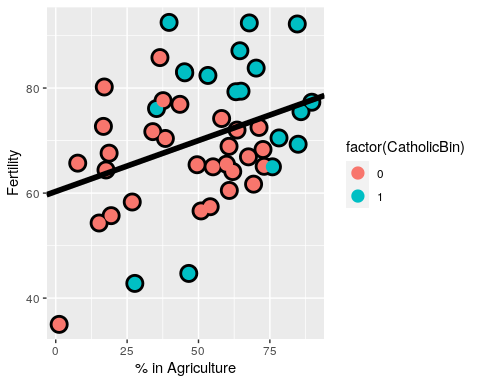
The model that includes X1 and X2, but no interaction fits two models that have the same slope, but they have different intercepts, and then . If we consider one third model , in this model when X2 is zero, it works out to be , and when X2 is 1, which is the case when the province is majority Catholic, we get , now let’s reorganize terms to get .

This model is showing if we omitted that interaction term, we fit two lines the same slope, if we include the interaction term, we get two lines different slopes and different intercepts. And the coefficient in front of the Catholic term is going to change in the intercept going from Protestant to Catholic.

### 3.2.3 Exploring the models in R

We start by fitting the model, where we fit the expected fertility as a linear function of the percent of the province working in agriculture.

fit = lm(Fertility ~ Agriculture, data = swiss)  
g1 = g  
g1 = g1 + geom\_abline(intercept = coef(fit)[1], slope = coef(fit)[2], size = 2)  
g1



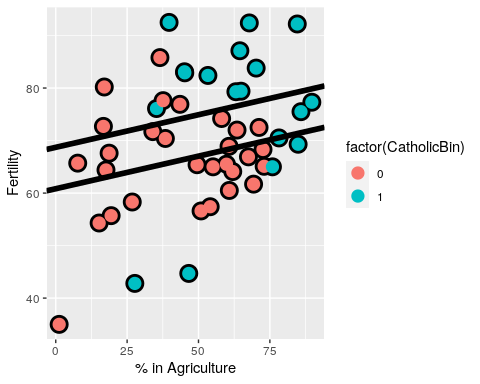
Here we want to save the plot in g1, then we can keep adding different things to it. The coefficients summaries can be obtained by summary(fit)$coef.

summary(lm(Fertility ~ Agriculture + factor(CatholicBin), data = swiss))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 60.8322366 4.1058630 14.815944 1.032493e-18  
## Agriculture 0.1241776 0.0810977 1.531210 1.328763e-01  
## factor(CatholicBin)1 7.8843292 3.7483622 2.103406 4.118221e-02

This just disregards the color of the points. Let’s do one that fits two parallel lines.

fit = lm(Fertility ~ Agriculture + factor(CatholicBin), data = swiss)  
g1 = g  
g1 = g1 + geom\_abline(intercept = coef(fit)[1], slope = coef(fit)[2], size = 2)  
g1 = g1 + geom\_abline(intercept = coef(fit)[1] + coef(fit)[3], slope = coef(fit)[2], size = 2)  
g1



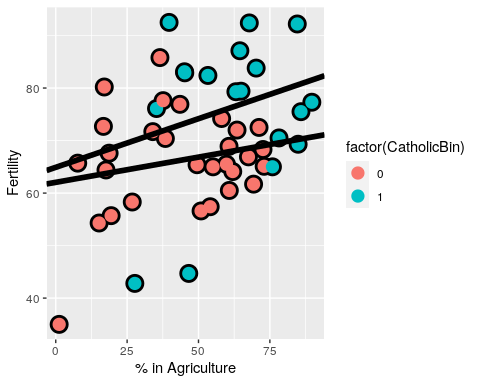
Because the variable CatholicBin is 0 or 1, we don’t actually have to have the factor statement. Because coding a variable of 0 versus 1 treats it as a factor. However we like to call factor variables and the reason is sometimes you have a variable like 0,1,2 for a 3 level variable, and if you don’t call that a factor, R is going to treat that as a continuous regressor. It’s going to say 2 is twice 1, even if 2 is just a numeric coding for representing red hair color and 1 is for brown hair color, where 2 really has nothing to do with being twice 1 in that case.

summary(lm(Fertility ~ Agriculture \* factor(CatholicBin), data = swiss))$coef

## Estimate Std. Error t value  
## (Intercept) 62.04993019 4.78915566 12.9563402  
## Agriculture 0.09611572 0.09881204 0.9727127  
## factor(CatholicBin)1 2.85770359 10.62644275 0.2689238  
## Agriculture:factor(CatholicBin)1 0.08913512 0.17610660 0.5061430  
## Pr(>|t|)  
## (Intercept) 1.919379e-16  
## Agriculture 3.361364e-01  
## factor(CatholicBin)1 7.892745e-01  
## Agriculture:factor(CatholicBin)1 6.153416e-01

We have an intercept and a slope, this intercept is the intercept for mostly Protestant provinces, the slope is the slope for mostly Protestant provinces. This 2.86 plus 62.04 is the intercept for the mostly Catholic provinces and the slope 0.096 + 0.089 is the slope for the mostly Catholic provinces. For lines with different intercepts and different slopes depending on the percent of the province that is Catholic, put an asterisk and do the fit. What happens when you add an asterisks in between two variables in R? It automatically fits the interaction. In general you want to include the main effects, if you include the interaction.

fit = lm(Fertility ~ Agriculture \* factor(CatholicBin), data = swiss)  
g1 = g  
g1 = g1 + geom\_abline(intercept = coef(fit)[1], slope = coef(fit)[2], size = 2)  
g1 = g1 + geom\_abline(intercept = coef(fit)[1] + coef(fit)[3],   
 slope = coef(fit)[2] + coef(fit)[4], size = 2)  
g1

 based on the coefficients all the Catholic terms were positive the line with the slightly higher intercept is the Catholic line and the line for the slightly lower intercept is the mostly Protestant. Now you can probably see that well for the blue dots it is not clear how the two outlier blue dots impacting the fit, so we might want to investigate that. We will do that in the next section where we talk about residuals and influence diagnostics and that sort of thing.

## 3.3 Adjustment

Adjustment, is the idea of putting regressors into a linear model to investigate the role of a third variable on the relationship between another two. Since it is often the case that a third variable can distort, or confound if you will, the relationship between two others.

As an example, consider looking at lung cancer rates and breath mint usage. For the sake of completeness, imagine if you were looking at forced expiratory volume (a measure of lung function) and breath mint usage. If you found a statistically significant regression relationship, it wouldn’t be wise to rush off to the newspapers with the headline “Breath mint usage causes shortness of breath!”, for a variety of reasons. First off, even if the association is sound, you don’t know that it’s causal. But, more importantly in this case, the likely culprit is smoking habits. Smoking rates are likely related to both breath mint usage rates and lung function. How would you defend your finding against the accusation that it’s just variability in smoking habits?

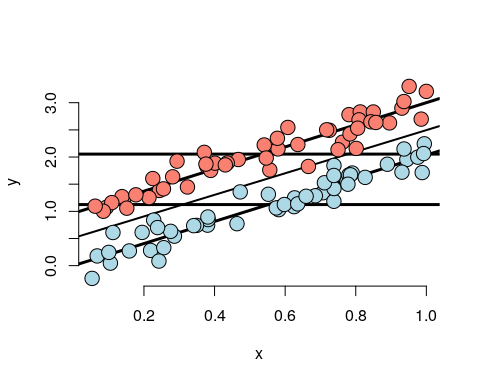
If your finding held up among non-smokers and smokers analyzed separately, then you might have something. In other words, people wouldn’t even begin to believe this finding unless it held up while holding smoking status constant. That is the idea of adding a regression variable into a model as adjustment. The coefficient of interest is interpreted as the effect of the predictor on the response, holding the adjustment variable constant.

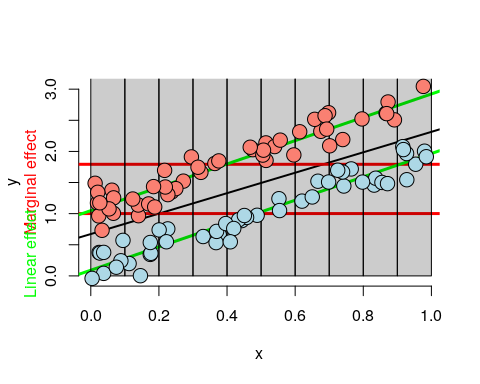
In this lecture, we’ll use simulation to investigate how adding a regressor into a model addresses the idea of adjustment.

### 3.3.1 Adjustment Examples

In this section, we are gonna go over some examples of how adjusting for one variable can impact the apparent relationship of another variable on an outcome. The easiest way to see this is to look at a two group variable. For example, a treatment versus a control or something you might see in an AB test when we are adjusting for a continuous variable. Here’s the first simulation:

n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2), runif(n/2));  
beta0 <- 0; beta1 <- 2; tau <- 1; sigma <- .2  
y <- beta0 + x \* beta1 + t \* tau + rnorm(n, sd = sigma)  
plot(x, y, type = "n", frame = FALSE)  
abline(lm(y ~ x), lwd = 2)  
abline(h = mean(y[1 : (n/2)]), lwd = 3)  
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3)  
fit <- lm(y ~ x + t)  
abline(coef(fit)[1], coef(fit)[2], lwd = 3)  
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)  
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 2)  
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 2)

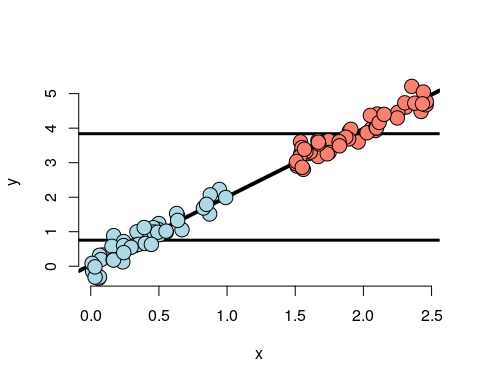


In this case, the red group was the group reciving treatment and the blue group was control. The horizontal lines show the marginal effect of group status disregarding the x. The y was a measure of blood pressure, then we would think that if we hadn’t factored in x, the mean for the group that received the treatment was around 2 and the mean for the control was around 0.8. You notice there is a pretty clear linear relationship between the outcome and the regressor. So what we could do is fit a model that looks like , where is the treatment indicator, with values of . This would fit two parallel lines with representing the change in intercepts between the groups and the common slope that exists across the two groups. 

Taking a closer look at y-axis shows the marginal effect, the effect that we have if we disregard x, and the effect that we have if we incorporate x in a linear model and look at the change in the intercepts, are about the same. In addition, note in this example there is a lot of direct evidence to compare the groups for any given value of x. If we binned x we would have red and blue circles for a direct comparison of the treatment for kind of a fairly isolated level of x.

Let’s try a setting where it’s gonna make a big difference.

n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2), 1.5 + runif(n/2));  
beta0 <- 0; beta1 <- 2; tau <- 0; sigma <- .2  
y <- beta0 + x \* beta1 + t \* tau + rnorm(n, sd = sigma)  
plot(x, y, type = "n", frame = FALSE)  
abline(lm(y ~ x), lwd = 2)  
abline(h = mean(y[1 : (n/2)]), lwd = 3)  
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3)  
fit <- lm(y ~ x + t)  
abline(coef(fit)[1], coef(fit)[2], lwd = 3)  
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)  
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 2)  
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 2)

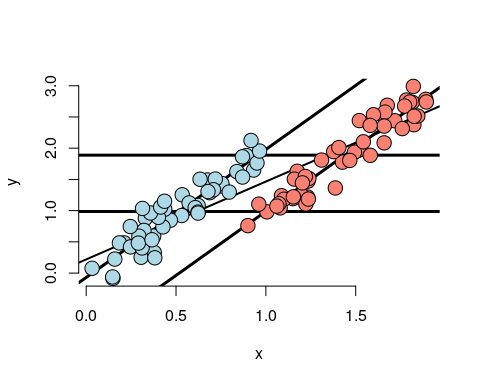


The horizontal lines show marginal difference between the red, treated group, and the blue, control group. However, if we fit that model and look at the change in the intercepts, we’d see a tiny difference. This is a case where we would go from a massive treatment effect to nothing when we accounted for x. Moreover, if we knew that the x value was one or smaller, we know that value is in the blue group, and if it was 1.5 or higher, we know the value was in the treated group. So knowledge of x at some level pretty much gives us perfect knowledge of received treatment. In a randomized setuation it would be very hard to pick what treatment you had based on your x level because the x levels were all jumbled up some of the high x levels went to the treated, some of the high x levels went to the control. However, in this case, we clearly didn’t randomize. The question of which model here is the right one to consider is not the discussion for this section, here we want to show how the inclusion of x can change the estimate. As an example, imagin y, your outcome, was the blood pressure and the x variable was cholesterol or something highly related to whether or not you would’ve gotten prescribed a medication. You could see that adjusting for x is really just adjusting for the same thing that would lead you to have treatment. Again, this is what makes observational data analysis very hard as opposed to instances where you have randomized sample. This is an example where we had a strong marginal effect when we disregarded x, and a very subtle or a non-existent effect when we accounted for x. One more point is there is no value of x we can hold constant and compare red versus blue directly, which is a bad setting, where we are relying very heavily on the model to compare the group.

A summary of the important points about the previous example can be listed as: \* The X variable is highly related to group status \* The X variable is related to Y, the intercept doesn’t depend on the group variable. \* The X variable remains related to Y holding group status constant \* The group variable is marginally related to Y disregarding X. \* The model would estimate no adjusted effect due to group. \* There isn’t any data to inform the relationship between group and Y. \* This conclusion is entirely based on the model.

Another senario is where there is some overlap between data point.

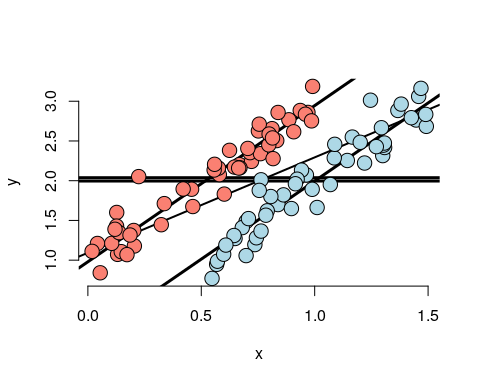
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2), .9 + runif(n/2));  
beta0 <- 0; beta1 <- 2; tau <- -1; sigma <- .2  
y <- beta0 + x \* beta1 + t \* tau + rnorm(n, sd = sigma)  
plot(x, y, type = "n", frame = FALSE)  
abline(lm(y ~ x), lwd = 2)  
abline(h = mean(y[1 : (n/2)]), lwd = 3)  
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3)  
fit <- lm(y ~ x + t)  
abline(coef(fit)[1], coef(fit)[2], lwd = 3)  
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)  
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 2)  
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 2)



There is some direct evidence for comparing the two groups. The marginal mean for the red and blue groups suggests the red is higher than the blue. However, the linear fit in the intercepts shows the blue is higher than the red. This shows the adjusted estimate is significant and the exact opposite of the unadjusted estimate. This phenomenon is often called Simpson’s Paradox, and the idea is things can change to the exact opposite when you perform adjustment.

Next senario is shown in the following plot.

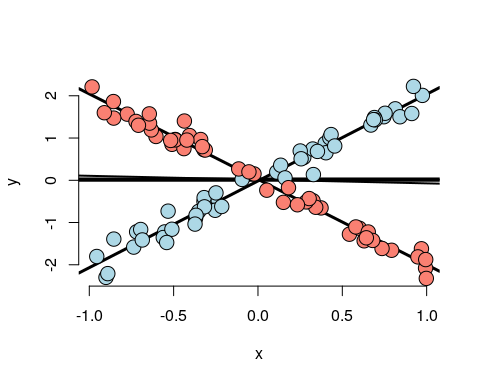
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(.5 + runif(n/2), runif(n/2));  
beta0 <- 0; beta1 <- 2; tau <- 1; sigma <- .2  
y <- beta0 + x \* beta1 + t \* tau + rnorm(n, sd = sigma)  
plot(x, y, type = "n", frame = FALSE)  
abline(lm(y ~ x), lwd = 2)  
abline(h = mean(y[1 : (n/2)]), lwd = 3)  
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3)  
fit <- lm(y ~ x + t)  
abline(coef(fit)[1], coef(fit)[2], lwd = 3)  
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)  
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 2)  
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 2)



Here there is no marginal effect. However, there is a huge effect when we adjust for x. there is no simple rule that says this is always what will happen with adjustment. Pretty much any permutation of going from significant to non-significant, staying both significant, staying non-significant, flipping signs can occur.

As our final example we look at the following graph.

n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2, -1, 1), runif(n/2, -1, 1));  
beta0 <- 0; beta1 <- 2; tau <- 0; tau1 <- -4; sigma <- .2  
y <- beta0 + x \* beta1 + t \* tau + t \* x \* tau1 + rnorm(n, sd = sigma)  
plot(x, y, type = "n", frame = FALSE)  
abline(lm(y ~ x), lwd = 2)  
abline(h = mean(y[1 : (n/2)]), lwd = 3)  
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3)  
fit <- lm(y ~ x + t + I(x \* t))  
abline(coef(fit)[1], coef(fit)[2], lwd = 3)  
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2] + coef(fit)[4], lwd = 3)  
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 2)  
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 2)

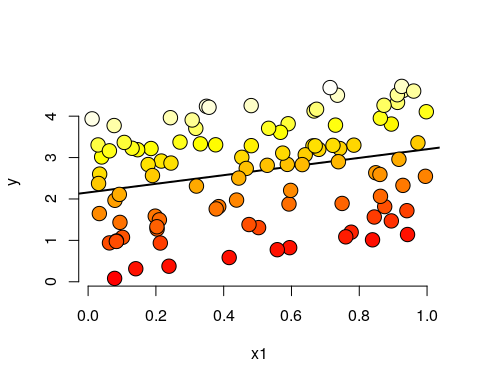


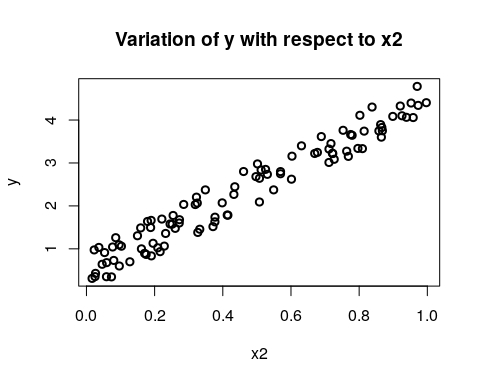
This example considers an instance where assuming the slopes were common across the two groups would be wrong. The linear model would fit two lines with different intercepts and different slopes. Another important thing to ascertain is there is no treatment effect.

If we look at the center there is no evidence of a treatment effect, if we look at the right side of the graph there is a big evidence that blue has a higher outcome than red, and if you look at the left side we see there is a lot of evidence that red has a higher outcome than blue. The interaction is the reason why the main treatment effect doesn’t have a lot of meaning and the coefficient in front of the treated effects is not interpreted as the treatment effect by itself. The treatment effect depends on what level of x you’re at, and you can’t just read the term from the regression output associated with the treatment and act as if it’s a treatment effect because we have an interaction term in the model. Again this shows how adjustment can really change things if you have a setting where you have not just adjustment, but so-called modification.

We want to reiterate that nothing we’ve talked about is specific to having a binary treatment and a continuous x.

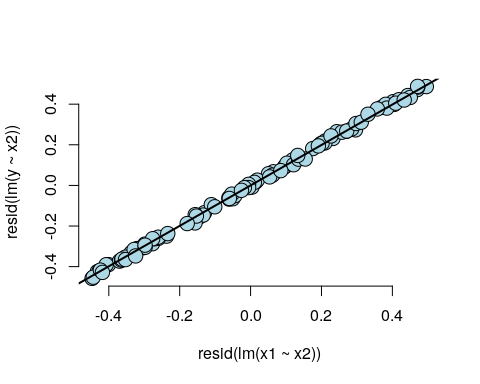
p <- 1  
n <- 100; x2 <- runif(n); x1 <- p \* runif(n) - (1 - p) \* x2  
beta0 <- 0; beta1 <- 1; tau <- 4 ; sigma <- .01  
y <- beta0 + x1 \* beta1 + tau \* x2 + rnorm(n, sd = sigma)  
plot(x1, y, type = "n", frame = FALSE)  
abline(lm(y ~ x1), lwd = 2)  
co.pal <- heat.colors(n)  
points(x1, y, pch = 21, col = "black", bg = co.pal[round((n - 1) \* x2 + 1)], cex = 2)



The above plot shows outcome, y, and the continuous variable and continuous variable color coded, higher lighter values mean higher, and more red darker values means lower. At first glance you might say there is not much of a relationship between y and x1, however, if we look at it in three dimensions you can see that most of the variation of y is explained by its relationship with x2. Here we show the 2D plot of y versus x2. 

An easy way to look at the other variables effect without having to resort to three dimensional plots, which don’t work if you move beyond two variables is to look at residuals.

plot(resid(lm(x1 ~ x2)), resid(lm(y ~ x2)), frame = FALSE, col = "black", bg = "lightblue", pch = 21, cex = 2)  
abline(lm(I(resid(lm(x1 ~ x2))) ~ I(resid(lm(y ~ x2)))), lwd = 2)



Here we show the residual of y after having removed x2, the linear effect of x2, and the residual of x1 having removed linear effect of x2. As you can see there is a very strong relationship left over between y and x1 after having removed the effect of x2.

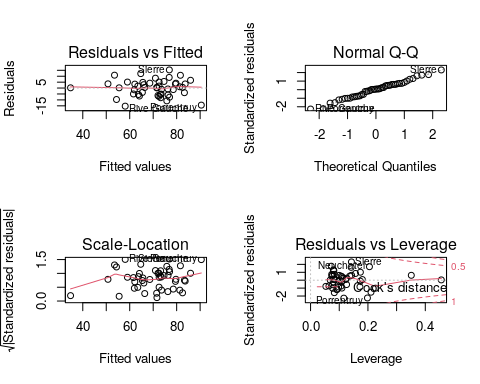
We want to reiterate we haven’t said what exactly is the right model, the best way to think about that is you have to bring in some of the specific subject matter, or clinical scientific subject matter expertise into your model building exercise. If you’re doing model building with a regular data set where you want interpretable coefficients getting the team of people some with the right scientific expertise, some with the statistical expertise, and some with the computing expertise is essential.

## 3.4 Residuals again

Recall from before that the vertical distances between the observed data points and the fitted regression line are called residuals. We can generalize this idea to the vertical distances between the observed data and the fitted surface in multivariable settings.

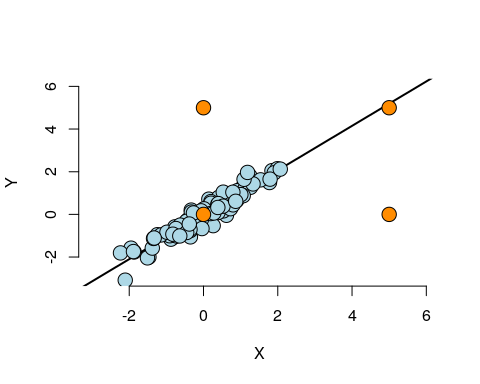
We will start with the model, . Through out this section we will assume errors are normally distribute. We defined the residuals as , and the estimate of residual variation as , the so that . Where we divide by n-p rather than n, for unbiasness. In R we can get a plot of the residuals and several plots that measure aspects of model fit and lack of model fit. Let’s load up the data set swiss and fit the linear model that includes all the terms, using a period after the tilde symbol will include all terms.

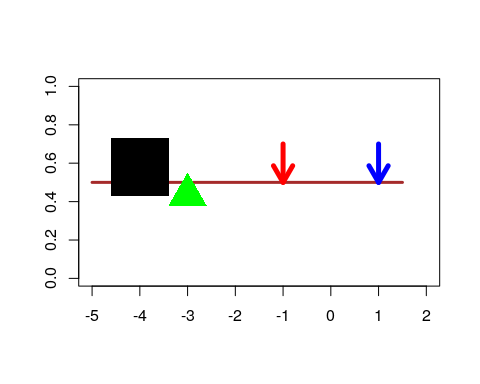
data(swiss); par(mfrow = c(2, 2))  
fit <- lm(Fertility ~ . , data = swiss); plot(fit)



The generic command plot will plot the fit, it will plot and co-create a series of residual and diagnostic plots. We now will go over what the basics of these plots are. To understand these we need to discuss influential points and outlier points. Take a look at the following plot.

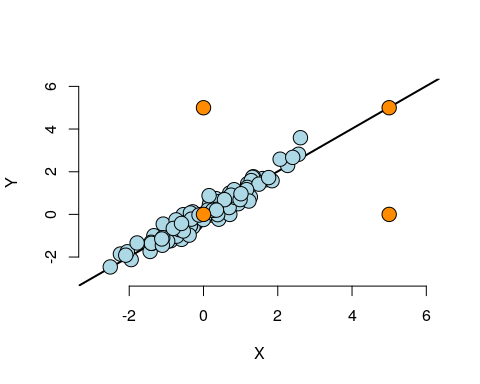
n <- 100; x <- rnorm(n); y <- x + rnorm(n, sd = .3)  
plot(c(-3, 6), c(-3, 6), type = "n", frame = FALSE, xlab = "X", ylab = "Y")  
abline(lm(y ~ x), lwd = 2)  
points(x, y, cex = 2, bg = "lightblue", col = "black", pch = 21)  
points(0, 0, cex = 2, bg = "darkorange", col = "black", pch = 21)  
points(0, 5, cex = 2, bg = "darkorange", col = "black", pch = 21)  
points(5, 5, cex = 2, bg = "darkorange", col = "black", pch = 21)  
points(5, 0, cex = 2, bg = "darkorange", col = "black", pch = 21)



In this plot there is a clear linear relationship between the predictor and the response. However, there are four points in orange that we would like to discuss. The first at the lower left hand side is right in the middle of the cloud of data. It doesn’t really have much influence or leverage. Remember that the line has to go through the average of the y’s and the average of the axis,.In that sense think of the averages as a mid point of fulcrum and that the bar namely the regression line has to move through that fulcrum and like a regular physical fulcrum, that point will have more leverage as you move further out. 

The force to hold the system in balance is higher if we apply in red arrow position and lower if we apply on blue arrow position. The same analogy happens in regression models, if we have a bunch of data, a datapoint that’s far outside of the natural range of the axis would have high leverage and a data point that is right in the middle of the x’s will have low leverage. The concept of leverage is merely a concept of how far away from the center of the axis the data point is. Influence on the other hand is whether or not that point chooses to exert that leverage.

n <- 100; x <- rnorm(n); y <- x + rnorm(n, sd = .3)  
plot(c(-3, 6), c(-3, 6), type = "n", frame = FALSE, xlab = "X", ylab = "Y")  
abline(lm(y ~ x), lwd = 2)  
points(x, y, cex = 2, bg = "lightblue", col = "black", pch = 21)  
points(0, 0, cex = 2, bg = "darkorange", col = "black", pch = 21)  
points(0, 5, cex = 2, bg = "darkorange", col = "black", pch = 21)  
points(5, 5, cex = 2, bg = "darkorange", col = "black", pch = 21)  
points(5, 0, cex = 2, bg = "darkorange", col = "black", pch = 21)



Going back to the plot we created before, the point on the upper right side has high leverage, but it opts not to execute that leverage. On the other hand the lower right hand point, if it was included in the regression line it would probably have a larger effect on it. Because it’s outside of the cloud of points and it also doesn’t adhere to what the rest of the date is saying, we can say this point has high leverage and high influence. One final point to discuss is the upper left point that is in the middle of the cloud or the X values, it has low leverage; however, it doesn’t adhere at all to the relationship meaning while being an outlier, it is not going to exert the same influence on the model that the lower right point is going to.

The idea of calling something an outlier is actually very vague. We want to make the term a little bit more precise. Ourselves outliers can be the result of real or spurious processes. If it is a spurious process we want to remove that point from our fitted model and if it is part of a real process, we don’t want to get rid of a point. We mentioned when outliers can conform to the regression relationship and when they can not. A fairly complete list of default diagnostic measures can be done by the R command ?influence.measures and this will give you a list of potential influence measures. The measures include: \* rstandard - standardized residuals, residuals divided by their standard deviations \* rstudent - standardized residuals, residuals divided by their standard deviations, where the ith data point was deleted in the calculation of the standard deviation for the residual to follow a t distribution \* hatvalues - measures of leverage (measure of influence) \* dffits - change in the predicted response when the point is deleted in fitting the model. \* dfbetas - change in individual coefficients when the point is deleted in fitting the model. \* cooks.distance - overall change in teh coefficients when the point is deleted. \* resid - returns the ordinary residuals \* resid(fit) / (1 - hatvalues(fit)) where fit is the linear model fit returns the PRESS residuals, i.e. the leave one out cross validation residuals - the difference in the response and the predicted response at data point , where it was not included in the model fitting.

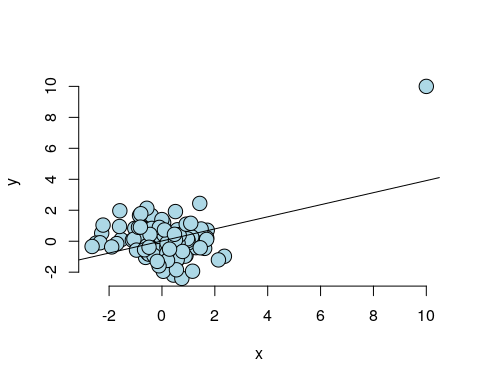
One way which hatvalues are tremendously useful is in finding data entry errors. You might look through the collection of rows of the data and look at the associated hatvalues, if you see some are extremely large not only would they potentially impact the model, but you should look at that row to ascertain whether there was a data entry error. To measuer the actual influence, not just the potential influence, we take out that data point refit the model, and then compare the models with and without the datapoint. dffits tracks how much the fitted value at that X value changes depending on whether or not that point was included. The dfbetas look specifically at the slope coefficients, for the dffits we get one dffit per data point. On the other hand, the dfbetas we get a dfbeta for every coefficient for every data points, if we have a model, with two coefficients, the intercept and the slope dfbetas will be a matrix of two by the number of datapoints. The cooks.distance is just an overall change in the coefficients. In a sense it summarizes the dfbetas. The PRESS residuals are less specifically useful for detecting outliers and determining influence than they are for ascertaining things like module fit, but we put it here because it’s related.

How do we use all of these things? Be wary of simplistic rules for diagnostic plots and measures. It’s better to understand what they are trying to accomplish and use them judiciously. If you’re digging deep into a data analysis, you really want to not think of these in terms of simple thresholding rules for declaring things an outlier. You want to take a more wholistic approach. Like look at the collective residuals relative to the other residuals, look at the collection of hat values relative to the other hat values. Not all of the measures have meaningful absolute scales. You can look at them relative to the values across the data. An important aspect of all these measures is that they probe the data to look at lack of influence, and whether or not something is an outlier in separate ways. So you want to really think of them as diagnostics in the same way a physicians thinks of diagnostics to try and figure out if there is any problem. Patterns in residual plots generally indicate some poor aspect of model fit like heteroskedasticity (non constant variance), missing model terms, temporal patterns (plot residuals versus collection order). Residual QQ plots investigate normality of the errors. Leverage measures (hat values) can be useful for diagnosing data entry errors and finally influence measures get to the bottom line, ‘how does deleting or including this point impact a particular aspect of the model’.

### 3.4.1 Examples in R

Here we will go through some simulation experiments to understand how some of the diagnostic measures work. In this first case, there is a big cloud of uncorrelated data.

n <- 100; x <- c(10, rnorm(n)); y <- c(10, c(rnorm(n)))  
plot(x, y, frame = FALSE, cex = 2, pch = 21, bg = "lightblue", col = "black")  
abline(lm(y ~ x))



What we can see in this case is that there is a strong correlation estimated by the data merely because of the existence of the point (10,10) otherwise the correlation would be estimated to be zero.

fit <- lm(y ~ x)  
round(dfbetas(fit)[1 : 10, 2], 3)

## 1 2 3 4 5 6 7 8 9 10   
## 8.466 0.013 0.011 0.035 -0.018 0.022 -0.035 -0.019 0.000 -0.090

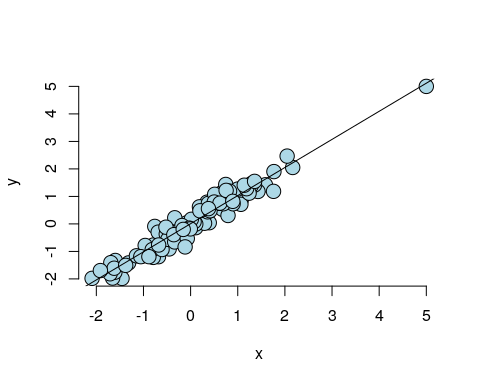
round(hatvalues(fit)[1 : 10], 3)

## 1 2 3 4 5 6 7 8 9 10   
## 0.487 0.010 0.011 0.013 0.011 0.012 0.014 0.010 0.010 0.020

The dfbetas shows the first point (10,10) is orders ofmagnitude larger than the remaining point. The hat value for this point is much larger than the hat values for the remaining points. The hat values have to be between zero and one. Based on thes diagnostics we would single out (10,10).

Second example is a clear regression relationship:

x <- rnorm(n); y <- x + rnorm(n, sd = .3)  
x <- c(5, x); y <- c(5, y)  
plot(x, y, frame = FALSE, cex = 2, pch = 21, bg = "lightblue", col = "black")  
fit2 <- lm(y ~ x)  
abline(fit2)



In this case the outlier adheres very nicely to the regression line.

round(dfbetas(fit2)[1 : 10, 2], 3)

## 1 2 3 4 5 6 7 8 9 10   
## -0.252 -0.005 -0.110 0.092 0.048 0.265 -0.032 -0.026 -0.017 0.052

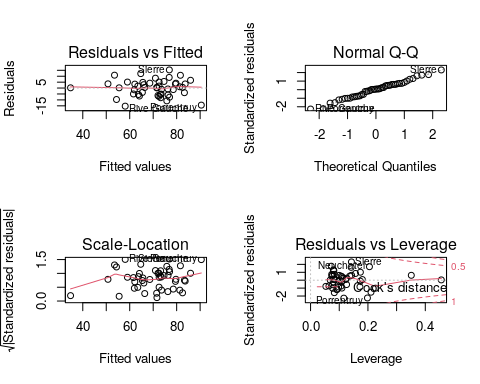
round(hatvalues(fit2)[1 : 10], 3)

## 1 2 3 4 5 6 7 8 9 10   
## 0.231 0.010 0.031 0.013 0.020 0.031 0.015 0.013 0.010 0.036

The diagnostic values of dfbetas for the first point, which was the outlying point. It’s still large but nowhere near as large as in the previous case. It appears to have some influence in the fit, but nothing like in the previous case. However, the hat values has a much larger hat value than and the remaining points. It is because the point is outside of the range of the X values, but it adheres to the direction relationship meaning it will have a large leverage value but not a large dfbetas value.

Now let’s go back to the swiss data and look at the different examples of diagnostic plots that can spit out by default by R.

data(swiss); par(mfrow = c(2, 2))  
fit <- lm(Fertility ~ . , data = swiss); plot(fit)



Earlier we started out with this now we want to interpret these. The residual plot versus the fitted values is trying to find anything that’s systematic. There doesn’t seem to be too many aspects of absence of model departures there. The Q-Q plot is specifically designed to test normality. The Scale-Location plot is plotting the standardized residuals, which are the ordinary residuals but standardize. This is a lot like the residual plot, you’re applying them against the fitted values but, now have changed the scale. That’s potentially useful for looking at the across different experiments. The final one is plot of the Residuals vs Leverage. These are a couple of examples of the kind of plots you might want to look at in this dataset none of them seem too inherently bad.

## 3.5 Model selection

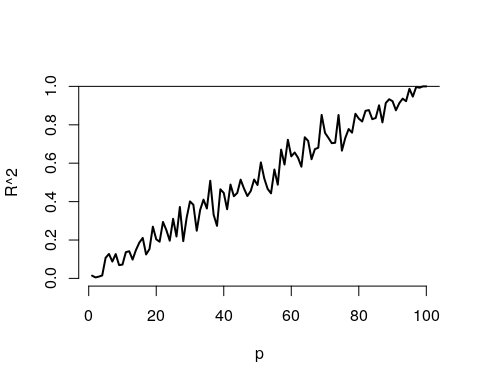
These lectures represents a challenging question: “How do we chose what variables to include in a regression model?”. Sadly, no single easy answer exists and the most reasonable answer would be “It depends.”. These concepts bleed into ideas of machine learning, which is largely focused on high dimensional variable selection and weighting. In the following lectures we cover some of the basics and, most importantly, the consequences of over- and under-fitting a model.

In this part, we will talk about the process of model fitting when you have lots of variables in a regression context. An important point is to distinguish between multivariable regression and the prediction. In prediction we are less concerned with interpretability so we are far more tolerant for complex models and automated search algorithms and rules to come up with the best prediction with respect to a specific loss function. In contrast, in a lot of regression examples, we want simple interpretable models. That means we are going to place a heavy emphasis on the idea of parsimony, which is finding the simplest model to explain what we are looking at, as simple as possible, but no simpler. If a variable in a regression model explains a tiny bit extra about a variation but really harms the interpretability, we are going to omit that variable. Any model that teaches you something valuable about your data is right, as long as it teaches you something true. There are uncountably many ways that our model can be wrong and lead us astray. In this section we will discuss some basics of what happens when we either include unnecessary regressors or exclude the important regressors. A qoute by Donald Rumsfeld *There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don’t know. But there are also unknown unknowns. There are things we don’t know we don’t know.* is a good way to start this section. There are known knowns, these are things we know that we know. Then there are known unknowns that is to say, there are things that we know we don’t know. But there are also unknown unknowns. There are things we don’t know we don’t know. This quote was heavily criticized, but for regression it makes a lot of sense. Known knowns are regressors that we know we should include in our model and we have recorded so we just put them in. There are known unknowns, and these are the things that we know we should include in our regression model, but unfortunately we didn’t collect them. And there are unknown unknowns and these are things that we should include in our regression model, but we didn’t collect them or don’t know whether or not they’re important.

Some general rules for our known knowns, unknown unknowns, and known unknowns: \* Omitting variables results in bias in the coeficients of interest - unless their regressors are uncorrelated with the omitted ones. \* This is why we randomize treatments, it attempts to uncorrelate our treatment indicator with variables that we don’t have to put in the model. \* (If there is too many unobserved confounding variables, even randomization won’t help you.) \* Including variables that we shouldn’t have increases standard errors of the regression variables. \* Actually, including any new variables increasese (actual, not estimated) standard errors of other regressors. So we don’t want to idly throw variables into the model. \* The model must tend toward perfect fit as the number of non-redundant regressors approaches . \* increases monotonically as more regressors are included. \* The SSE decreases monotonically as more regressors are included.

Remember if there is too many confounded then even randomization is not going to help you. So that means omitting variables that are important, wil have negative impact on our regression model fit. Including variables that are unimportant will not have bias issue; however, they will inflate standard error. Additionally, some of the model terms, like, R squared are not necessarily good ways to adjudicate between looking at different models with different regressors. The reason with R squared is as you put in a bunch of new regressors, it’s guaranteed to go up no matter what. In the sum of the squared errors, the mean squared error is guaranteed to go down in regression as you include more regressors. Here is an example where we simulated a bunch of data and redid the simulation for every number of regressors and included them into the model. It shows the R squared going up even though all we are doing is generating a bunch of random normal vectors and putting them into a regression model.

n <- 100  
plot(c(1, n), 0 : 1, type = "n", frame = FALSE, xlab = "p", ylab = "R^2")  
r <- sapply(1 : n, function(p)  
 {  
 y <- rnorm(n); x <- matrix(rnorm(n \* p), n, p)  
 summary(lm(y ~ x))$r.squared   
 }  
 )  
lines(1 : n, r, lwd = 2)  
abline(h = 1)



Let’s go a little bit further in variance inflation. We will explain this using the following code:

n <- 100;   
nosim <- 1000 # number of simulations  
x1 <- rnorm(n); x2 <- rnorm(n); x3 <- rnorm(n);   
betas <- sapply(1 : nosim, function(i){  
 y <- x1 + rnorm(n, sd = .3)  
 c(coef(lm(y ~ x1))[2], # the model that just includes x1  
 coef(lm(y ~ x1 + x2))[2], # the model that includes x1 and x2  
 coef(lm(y ~ x1 + x2 + x3))[2]) # the model that includes x1, x2 and x3  
})  
round(apply(betas, 1, sd), 5)

## x1 x1 x1   
## 0.03317 0.03316 0.03346

We are interested in the standard deviation of the simulated estimated coefficients. The reason we do this by simulation is the variance inflation occurs on the actual standard errors, not the estimated standard errors. This is sort of the ideal setting, where the three regressors don’t have anything to do with one another. What you see is the standard deviation of the coefficients we get across all simulations. They are about same and nothing that bad. The variance inflation by including the extra variables was negligible. The reason is because x2 and x3, have nothing to do x1. We simulated them independently.

In the following case x2 and x3 are heavily dependent on x1.

n <- 100; nosim <- 1000  
x1 <- rnorm(n); x2 <- x1/sqrt(2) + rnorm(n) /sqrt(2)  
x3 <- x1 \* 0.95 + rnorm(n) \* sqrt(1 - 0.95^2);   
betas <- sapply(1 : nosim, function(i){  
 y <- x1 + rnorm(n, sd = .3)  
 c(coef(lm(y ~ x1))[2],   
 coef(lm(y ~ x1 + x2))[2],   
 coef(lm(y ~ x1 + x2 + x3))[2])  
})  
round(apply(betas, 1, sd), 5)

## x1 x1 x1   
## 0.03149 0.04037 0.10649

As expected we see huge amounts of variance inflation, especially for the third model where we include x2 and x3. The general rule is that the more correlated the covariates are to the regressors that you’re interested in, the worse off you’re going to be in terms of paying a penalty for increased standard deviation. For example, if we have diastolic blood pressure data in the model, and we put systolic blood pressure in the model, which is relatively the same thing it is going to make the diastolic blood pressure standard error increase. Another example is if you had height in a model and then you also included weight then you just have to bite the bullet and deal with the extra standard deviation. However, if it’s unnecessary then you don’t want to include weight. If we omit variables that are important, we get bias. If we include variables that are unnecessary, we inflate standard errors. Especially if we include variables that are unnecessary and are correlated with our variable of interest. This again reiterates why randomization is so important it randomizes across the other variables, the known and unknown knowns that we would otherwise need to include in the model, so that we can only include the treatment of interest. The things that we feel we have to include should be relatively balanced, or relatively uncorrelated with the treatment because it has been randomized. To avoid the undesirable increase in variance that would occur otherwise, randomization becomes crucial. This is why randomization holds such significance. Now, let’s delve into the concept of variance inflation factors. Since we lack knowledge of the actual sigma, precise calculation of variance inflation is not feasible. It relies on the true residual variance, which remains unknown. However, by comparing their ratios, we can eliminate the sigma variable and generate metrics such as ratios and variance inflation. These metrics accurately portray the extent to which standard deviation experiences inflation or remains unaffected. A widely used metric is the Variance Inflation Factor (VIF). Essentially, VIF is a comparison, specifically the ratio, of the variance inflation stemming from the existing data. This comparison involves examining the standard error in your dataset against the hypothetical scenario where all other covariates of interest are orthogonal or, in simpler terms, uncorrelated. Recall that in an ideal situation where all other variables are uncorrelated with your area of interest, minimal variance inflation is observed. The variance inflation factor assesses the current scenario in comparison to the ideal scenario where your area of interest is uncorrelated with all other variables. This factor is crucial for understanding the extent of variance inflation. Additionally, a simulation is provided to illustrate how these ratios can be estimated, although the details of the simulation are not presented here.

y <- x1 + rnorm(n, sd = .3)  
a <- summary(lm(y ~ x1))$cov.unscaled[2,2]  
c(summary(lm(y ~ x1 + x2))$cov.unscaled[2,2],  
 summary(lm(y~ x1 + x2 + x3))$cov.unscaled[2,2]) / a

## [1] 1.62661 10.32427

temp <- apply(betas, 1, var); temp[2 : 3] / temp[1]

## x1 x1   
## 1.64394 11.43768

Let’s analyze the variance inflation using the Swiss data.

library(car)

## Warning: package 'car' was built under R version 4.0.3

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.3

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

fit <- lm(Fertility ~ . , data = swiss)  
vif(fit)

## Agriculture Examination Education Catholic   
## 2.284129 3.675420 2.774943 1.937160   
## Infant.Mortality   
## 1.107542

sqrt(vif(fit)) #sd prefered

## Agriculture Examination Education Catholic   
## 1.511334 1.917138 1.665816 1.391819   
## Infant.Mortality   
## 1.052398

We plan to fit the model to explore what occurs if there is a residual effect of the first aspirin when administering the second one. The model will include all variables, denoted by ‘~.’ in the fitting process. Subsequently, we will use the vif(fit) function to obtain the variance inflation factors. Please ensure that you load the required library(car) before proceeding. In the output, the first line displays the variance inflation factor, while the line below presents its square root. Let’s focus on the first line initially, as it is the more conventional approach. For instance, the variance inflation factor for agriculture is 2. This implies that the standard error for the agriculture effect is double what it would be if it were orthogonal to all other regressors. Some factors exhibit high inflation, like examination with a factor of 3.67, almost four times the instance when orthogonal to other regressors. Notably, education and examination are highly related, leading to their elevated variance inflation factors. Consider variables like infant mortality, which has a notably low variance inflation factor. Infant mortality is largely unrelated to other variables, resulting in a minimal inflation factor. This insight provides a glimpse into the impact of including unnecessary variables, where variance inflation factors prove to be valuable tools.

Up until now, our focus has been on the influence of included or excluded variables on regression coefficients. However, there’s another crucial parameter in regression models: the residual variance (sigma squared). Assuming independent and identically distributed errors in our linear model, we can mathematically describe the consequences of omitting or including unnecessary variables. This follows a similar logic to our prior discussion underfitting the model, omitting important variables, leads to biased variance estimates. Because we have attributed to variations that are actually systematically explained by the co-variants that we have omitted. Conversely, if we correctly fit the model by including all relevant terms, or if we overfit the model, the variance estimates remain unbiased. However, including unnecessary variables inflates the variance of the variance estimate. Essentially, the same rule applies here as it did with coefficients: omitting variables leads to bias, while including them results in less reliable estimates. This illustrates a similar impact across both scenarios.

Let’s shift our focus to model selection in general, particularly automated model selection. Initially, we will fit the model with all variables included. This topic, once primarily within the domain of statistics, has largely transitioned into machine learning field. Even for relatively straightforward linear regression models, the space of potential model terms expands rapidly, especially with the inclusion of interactions and polynomial terms. When faced with numerous regressors and the need to streamline this space, techniques such as factor analysis and principal component analysis become invaluable. However, they come with trade-offs; the interpretability of the obtained principal components or factors may be diminished compared to the original data. It’s worth noting that extensive model discussion can often be circumvented with good design practices. Randomization, for instance, can mitigate many issues by ensuring that the variable of interest remains unrelated to unwanted nuisance variables. Additionally, strategies like stratification and randomization within strata can serve similar purposes, addressing potential sources of bias or confounding. The classic example of the importance of experimental design dates back to R.A. Fisher’s work in field crop experiments. In this context, if one were testing different types of seeds, blocking on various areas of the field was crucial. This involved systematically distributing different seeds across the field while incorporating randomization within that design. Experimental design, as illustrated by Fisher’s field crop experiments, is a broad and impactful topic. In biostatistics, a field with frequent applications of experimental design, the cross-over design is commonly employed. This design aims to use each subject as their own control. For instance, in a study comparing two types of aspirin, participants may receive both aspirins at different times, ensuring that each person is compared to themselves. This control strategy helps mitigate biases that might arise from differences between groups. Considerations for experimental design also extend to addressing potential residual effects, as seen in the example of the second aspirin’s impact after the first one. Strategies such as implementing a washout period may be employed to handle such scenarios. Despite these nuances, the overarching goal of experimental design is to compare individuals with themselves, controlling for intrinsic factors. The significance of thoughtful experimental design becomes evident in its ability to alleviate the complexities associated with model building. By strategically planning and executing experiments, researchers can minimize the need for extensive model considerations that would arise in an observational data collection approach.

One automated search model technique that proves to be quite useful involves examining nested models. This approach is particularly handy when you’re focused on a specific variable, such as a treatment, and are keen on understanding how other collected variables may impact it. The idea is to create nested models, where each successive model incorporates all the terms of the previous one. This strategy facilitates easy testing of each model, providing insights into the necessity of including additional terms. Here’s an example using R to demonstrate nested model testing with three linear models fitted to the SWF dataset:

# Fit three linear models  
fit1 <- lm(Fertility ~ Agriculture, data = swiss)  
fit3 <- update(fit, Fertility ~ Agriculture + Examination + Education)  
fit5 <- update(fit, Fertility ~ Agriculture + Examination + Education + Catholic + Infant.Mortality)  
  
# Display the results  
anova(fit1, fit3, fit5)

## Analysis of Variance Table  
##   
## Model 1: Fertility ~ Agriculture  
## Model 2: Fertility ~ Agriculture + Examination + Education  
## Model 3: Fertility ~ Agriculture + Examination + Education + Catholic +   
## Infant.Mortality  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 45 6283.1   
## 2 43 3180.9 2 3102.2 30.211 8.638e-09 \*\*\*  
## 3 41 2105.0 2 1075.9 10.477 0.0002111 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

In this example, we fit models with increasing complexity, starting with a model including only the variable of interest (Agriculture) and gradually adding other relevant variables. The ANOVA function is then used to test whether the inclusion of additional terms is necessary. The output provides information about the models, including degrees of freedom, residual sums of squares, and excess degrees of freedom when moving from one model to the next. By calculating the S statistic and associated P values, you can determine if the inclusion of specific terms is necessary at each stage. This approach allows you to systematically evaluate the impact of adding variables to the model and helps inform decisions about model complexity based on statistical significance. If your data naturally lends itself to a nested model search, especially when you’re interested in a specific variable, then employing this approach is quite reasonable. The nested model search is particularly suitable when you want to analyze a series of models that gradually become more complex while including the terms from the previous models. However, this technique has limitations, it works best when the models you’re examining are nested. If the models are not nested, more advanced methods like information criteria may be necessary, but those are topics for a prediction-focused class.

In this section, you’ve learned a couple of model selection techniques and the basic consequences of including or excluding variables in your models. These consequences affect the coefficients and residual variance estimates, and while we haven’t delved into other model assumptions like linearity and normality, it’s crucial to approach your model with a degree of skepticism, acknowledging that it’s likely to be imperfect. The famous quote by George Box, “All models are wrong, some models are useful,” serves as a guiding principle. Recognize that your model may not perfectly capture the complexity of reality, but if it provides valuable insights and proves useful in understanding your dataset, it serves its purpose. I hope this lecture has equipped you with valuable tools, and I look forward to our next session.

## 3.6 Practical R Exercises in swirl

During this week of the course you should complete the following lessons in the Regression Models swirl course:

1. MultiVar Examples2
2. MultiVar Examples3
3. Residuals Diagnostics and Variation

## 3.7 Week 3 Quiz

1. Consider the mtcars data set. Fit a model with mpg as the outcome that includes number of cylinders as a factor variable and weight as confounder. Give the adjusted estimate for the expected change in mpg comparing 8 cylinders to 4.
2. Consider the mtcarsdata set. Fit a model with mpg as the outcome that includes number of cylinders as a factor variable and weight as a possible confounding variable. Compare the effect of 8 versus 4 cylinders on mpg for the adjusted and unadjusted by weight models. Here, adjusted means including the weight variable as a term in the regression model and unadjusted means the model without weight included. What can be said about the effect comparing 8 and 4 cylinders after looking at models with and without weight included?.
   1. Including or excluding weight does not appear to change anything regarding the estimated impact of number of cylinders on mpg.
   2. Holding weight constant, cylinder appears to have less of an impact on mpg than if weight is disregarded.
   3. Holding weight constant, cylinder appears to have less of an impact on mpg than if weight is disregarded.
   4. Holding weight constant, cylinder appears to have more of an impact on mpg than if weight is disregarded.
3. Consider the mtcarsdata set. Fit a model with mpg as the outcome that considers number of cylinders as a factor variable and weight as confounder. Now fit a second model with mpg as the outcome model that considers the interaction between number of cylinders (as a factor variable) and weight. Give the P-value for the likelihood ratio test comparing the two models and suggest a model using 0.05 as a type I error rate significance benchmark.
   1. The P-value is larger than 0.05. So, according to our criterion, we would fail to reject, which suggests that the interaction terms is necessary.
   2. The P-value is larger than 0.05. So, according to our criterion, we would fail to reject, which suggests that the interaction terms may not be necessary.
   3. The P-value is small (less than 0.05). Thus it is surely true that there is no interaction term in the true model.
   4. The P-value is small (less than 0.05). Thus it is surely true that there is an interaction term in the true model.
   5. The P-value is small (less than 0.05). So, according to our criterion, we reject, which suggests that the interaction term is not necessary.
   6. The P-value is small (less than 0.05). So, according to our criterion, we reject, which suggests that the interaction term is necessary
4. Consider the mtcars data set. Fit a model with mpg as the outcome that includes number of cylinders as a factor variable and weight inlcuded in the model as lm(mpg ~ I(wt \* 0.5) + factor(cyl), data = mtcars). How is the wt coefficient interpretted?
   1. The estimated expected change in MPG per half ton increase in weight for for a specific number of cylinders (4, 6, 8).
   2. The estimated expected change in MPG per one ton increase in weight.
   3. The estimated expected change in MPG per one ton increase in weight for a specific number of cylinders (4, 6, 8).
   4. The estimated expected change in MPG per half ton increase in weight.
   5. The estimated expected change in MPG per half ton increase in weight for the average number of cylinders.
5. Consider the following data set and give the hat diagonal for the most influential point.

x <- c(0.586, 0.166, -0.042, -0.614, 11.72)  
y <- c(0.549, -0.026, -0.127, -0.751, 1.344)

1. Consider the following data set, give the slope dfbeta for the point with the highest hat value.

x <- c(0.586, 0.166, -0.042, -0.614, 11.72)  
y <- c(0.549, -0.026, -0.127, -0.751, 1.344)

1. Consider a regression relationship between Y and X with and without adjustment for a third variable Z. Which of the following is true about comparing the regression coefficient between Y and X with and without adjustment for Z.
   1. The coefficient can’t change sign after adjustment, except for slight numerical pathological cases.
   2. Adjusting for another variable can only attenuate the coefficient toward zero. It can’t materially change sign.
   3. For the the coefficient to change sign, there must be a significant interaction term.
   4. It is possible for the coefficient to reverse sign after adjustment. For example, it can be strongly significant and positive before adjustment and strongly significant and negative after adjustment.

## 3.8 (OPTIONAL) Practice exercise in regression modeling

## 3.9 You are being asked to participate in a research experiment with the purpose of better understanding how people analyze data. If you complete this quiz, you are giving your consent to participate in the study. This quiz involves a short data analysis that gives you a chance to practice the regression concepts you have learned so far. We anticipate that this will take about 15 minutes to complete. You will be receiving feedback on your work immediately after submission. For this reason, we ask that you do not post on the forums about this quiz to maintain the integrity of this experiment. Thank you for helping us learn more about data science! -Brian, Roger, Jeff

Your assignment is to study how income varies across different categories of college majors. You will be using data from a study of recent college graduates. Make sure to use good practices that you have learned so far in this course and previous courses in the specialization.In particular, it is good practice to specify an analysis plan early in the process to avoid the “p-hacking” behavior of trying many analyses to find one that has desired results. If you want to learn more about “p-hacking”, you can visit <https://projects.fivethirtyeight.com/p-hacking/>

Your assignment is to study how income varies across college major categories. Specifically answer: “Is there an association between college major category and income?”

To get started, start a new R/RStudio session with a clean workspace. To do this in R, you can use the q() function to quit, then reopen R. The easiest way to do this in RStudio is to quit RStudio entirely and reopen it. After you have started a new session, run the following commands. This will load a data.frame called college for you to work with.

install.packages("devtools")  
devtools::install\_github("jhudsl/collegeIncome")  
library(collegeIncome)  
data(college)

Next download and install the matahari R package with the following commands:

install.packages("matahari")  
library(matahari)

This package allows a record of your analysis (your R command history) to be documented. You will be uploading a file containing this record to GitHub and submitting the link as part of this quiz. Before you start the analysis for this assignment, enter the following command to begin the documentation of your analysis:

dance\_start(value = FALSE, contents = FALSE)

You can then proceed with the rest of your analysis in R as usual. When you have finished your analysis, use the following command to save the record of your analysis on your desktop:

dance\_save("~/Desktop/college\_major\_analysis.rds")

Please upload this college\_major\_analysis.rds file to a public GitHub repository. In question 4 of this quiz, you will share the link to this file.

A codebook for the dataset is given below: \* rank: Rank by median earnings \* major\_code: Major code \* major: Major description \* major\_category: Category of major \* total: Total number of people with major \* sample\_size: Sample size of full-time, year-round individuals used for income/earnings estimates: p25th, median, p75th \* p25th: 25th percentile of earnings \* median: Median earnings of full-time, year-round workers \* p75th: 75th percentile of earnings \* perc\_men: % men with major (out of total) \* perc\_women: % women with major (out of total) \* perc\_employed: % employed (out of total) \* perc\_employed\_fulltime: % employed 35 hours or more (out of employed) \* perc\_employed\_parttime: % employed less than 35 hours (out of employed) \* perc\_employed\_fulltime\_yearround: % employed at least 50 weeks and at least 35 hours (out of employed and full-time) \* perc\_unemployed: % unemployed (out of employed) \* perc\_college\_jobs: % with job requiring a college degree (out of employed) \* perc\_non\_college\_jobs: % with job not requiring a college degree (out of employed) \* perc\_low\_wage\_jobs: % in low-wage service jobs (out of total)

**Question** Based on your analysis, would you conclude that there is a significant association between college major category and income? Please type a few sentences describing your results. Please upload the file generated by matahari (college\_major\_analysis.rds) to a public GitHub repository.

## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)  
## Installing package into '/usr/local/lib/R/site-library'  
## (as 'lib' is unspecified)

# 4 Week 04

## 4.1 GLM

Generalized linear models (GLMs) were a great advance in statistical modeling. The original manuscript with the GLM framework was from [Nelder and Wedderburn](https://www.jstor.org/stable/2344614) in 1972 in the Journal of the Royal Statistical Society. The [McCullagh and Nelder book](https://books.google.com/books/about/Generalized_Linear_Models_Second_Edition.html?id=h9kFH2_FfBkC) is the famous standard treatise on the subject.

Recall linear models. Linear models are the most useful applied statistical technique. However, they are not without their limitations. Additive response models don’t make much sense if the response is discrete, or strictly positive. Additive error models often don’t make sense, for example, if the outcome has to be positive. Transformations, such as taking a cube root of a count outcome, are often hard to interpret.In addition, there is value in modeling the data on the scale that it was collected. Particularly interpretable transformations, natural logarithms in specific, are not applicable for negative or zero values. The generalized linear model is family of models that includes linear models. By extending the family, it handles many of the issues with linear models, but at the expense of some complexity and loss of some of the mathematical tidiness. A GLM involves three components: \* An exponential family model for the response. \* A systematic component via a linear predictor. \* A link function that connects the means of the response to the linear predictor. The three most famous cases of GLMs are: linear models, binomial and binary regression and Poisson regression. We will go through the GLM model specification and likelihood for all three. For linear models, we have developed them previously. The next two modules will be devoted to binomial and Poisson regression. We will only focus on the most popular and useful link functions.

In this section, we are transitioning from linear additive response models to the broader class of generalized linear models (GLMs). While linear models are immensely useful in applied statistics, they have limitations, especially when dealing with certain types of data. GLMs address some of these challenges, but they may sacrifice some mathematical elegance that linear models offer. Linear models assume additivity, which can be challenging to apply to binary data. Considering binary data as additive might lead to awkward interpretations of the error structure. Additionally, when dealing with strictly positive data, the assumption of additive Gaussian errors can be problematic as it allows for positive mass on negative values. While this might not be an issue in some cases, it becomes problematic if the normal distribution assigns a significant positive probability to negative values, contradicting the known positivity of the response. Transformations are a common strategy to handle strictly positive data. For example, taking the natural logarithm is often used, providing interpretability to the coefficients. However, other transformations, like the arcsine or square root transformation for binomial data, can compromise interpretability. One reason to consider generalized linear models is to avoid the need for transformations and to work directly with the scale in which the data was recorded. GLMs can respect the inherent characteristics of the data, especially when dealing with binary outcomes. The natural log transformation is frequently used, but has challenges when applied to negative or zero values, leading to the development of alternative methods. Generalized linear models were introduced in a 1972 paper by Nelder and Wedderburn, a seminal work in the field of statistics that has been widely read by PhD statisticians. GLMs provide a versatile framework for addressing the limitations of linear models and handling various types of response variables.

A generalized linear model (GLM) comprises three essential components:

1. **Distribution (Random Component):** The randomness in the model is described by a specific family of distributions known as the exponential family. This family includes various distributions like normal, binomial, and Poisson. The choice of the distribution depends on the nature of the response variable.
2. **Systematic Component (Linear Predictor):** This component involves the linear predictor, which is the part of the model being estimated. This component corresponds to what we modeled in linear models, where the systematic component was the linear combination of covariates with their coefficients.
3. **Link Function:** The link function connects the linear predictor to the distribution of the response variable from the exponential family. It essentially transforms the linear predictor into a scale that makes sense for the chosen distribution.

Suppose we assume that the response variable is normally distributed with mean . We define the linear predictor to be , and the link function as , identity link function, so that and . This yields the same likelihood model as our additive error Gaussian linear model

where .

To summarize, in this case: - **Distribution:** The exponential family is represented by the normal distribution. - **Systematic Component (Linear Predictor):** The linear predictor, denoted as η (eta), is defined as the sum of covariates (xi) multiplied by their coefficients (beta): ηi = ∑(xi \* beta). - **Link Function:** For simplicity, we can choose the identity link function, which directly connects the linear predictor (η) to the mean (Mu) of the normal distribution. This link function implies that Mu is exactly equal to the linear predictor: Mu = η.

The GLM framework allows us to extend the modeling approach beyond linear models by incorporating different distributions, systematic components, and link functions tailored to the characteristics of the response variable.

Let’s delve into logistic regression, perhaps the most useful variation of generalized linear models when dealing with binary data (0s and 1s). Since binary outcomes don’t conform to a normal distribution, we employ a Bernoulli distribution, assuming that our outcomes (Y) follow this distribution. In a Bernoulli distribution, we model the probability of a head (or success) for each observation. This probability is represented by , where the linear predictor remains the sum of covariates multiplied by their coefficients. The link function used in logistic regression is the logistic link function, which transforms the probability of success () into the log odds (log of the odds ratio). .Mathematically, this link function is expressed as the natural logarithm of the odds: . The logistic link function connects the linear predictor () to the log odds. To go from the log odds back to the probability (), we use the inverse of the logistic link function. The inverse logistic function, sometimes referred to as the “logit,” is given by:

It’s important to note that this transformation applies to the mean or probability of success () rather than directly transforming the binary outcomes (Y) themselves. This distinction is a key aspect of generalized linear models, where the focus is on modeling the underlying probabilities or means associated with the observed outcomes. Given the information the likelihood is:

It’s through this likelihood that parameter estimates are optimized by maximizing it.

Let’s discuss another example, Poisson regression. Poisson distribution is suitable for count data, which is often unbounded and positive. The Poisson regression assumes that the response variable Y follows a Poisson distribution with the expected value . The linear predictor is again the sum of covariates multiplied by their coefficients, . The log link function is commonly used in Poisson regression, . Recall that is the inverse of so that . The inverse of the natural logarithm is applied to go from the mean () to the linear predictor (). Thus, the likelihood is

The likelihood function for Poisson regression is formulated based on this distribution and the linear predictor. In generalized linear models, the parameters are estimated by maximizing the likelihood function. This is akin to finding roots or solutions of equations, similar to least squares in linear regression.

All models achieve their maximum at the root of the so called normal equations

where are the derivative of the inverse of the link function.

The MLE equation for generalized linear models, presented above, involves weights and variances in the denominator, emphasizing that the fitting process is a bit more complex than least squares. While these details are interesting from a statistical perspective, for practical purposes and interpretation of generalized linear models, understanding the conceptual framework, link functions, and likelihood is often sufficient. It’s important to note that for cases like Poisson and binomial, the variance is not constant across observations, unlike the linear model case. In the Poisson case, the variance of a Poisson distribution equals its mean. Therefore, in this scenario, the Poisson distribution has a variance that differs by . This assumption in modeling can be verified. For instance, if you have Poisson data with multiple observations at the same covariance level, the means should be equal. Consequently, the variances of these observations should be approximately equal to the mean. If your data doesn’t demonstrate this pattern, it poses an issue. This is a crucial and practical consideration in generalized linear models, where modeling assumptions often impose a constraint on the relationship between the mean and variance. However, this relationship may not hold in your specific dataset.

To address this issue, there is a method involving a more flexible variance model, albeit at the cost of sacrificing some generalized linear model assumptions. Standard options include quasi-Poisson and quasi-binomial distributions, which are available in statistical languages fitting generalized linear models. These quasi versions offer a more flexible variant structure, accommodating data that deviates from the typical GLM variant structure.

Moving on to the fitting process, it’s essential to note that GLMs require iterative optimization, unlike linear models, which allow for straightforward solutions through linear algebra. Consequently, program failures may occur, especially in cases like binary regression with numerous zeros or ones. Before delving into specific cases, such as Poisson and Binomial, it’s crucial to understand that these equations need iterative solutions. Although we won’t cover the full spectrum of GLMs, the process involves solving equations iteratively. For instance, if you wish to obtain the predicted response, you would multiply the estimated coefficients () by the regressors, yielding the predictive response. It’s important to recognize that this response will be on the log-odds scale for logistic regression or the log scale for Poisson regression. To align it with the original data scale, a conversion to the natural scale is necessary. For example, if you’re modeling coin flips and you obtain regression coefficients to predict responses, those predictions will be on the log-odds scale. To bring them back to the scale of coin flips (binary outcomes between zero or one), you’ll need to invert the log-odds. The coefficients are interpreted similarly to linear regression coefficients, representing the expected change in the response for a unit change in the regressors, with other regressors held constant. However, this interpretation is now done on the scale of the linear predictor: log-odds for the binomial case and log-mean for the Poisson case. Though the interpretation is slightly more complex, the benefit lies in modeling the data on its natural scale without transforming the outcome.

In terms of inference, you lose the convenience of closed-form normal inferences and t-distributions. However, statisticians have determined the appropriate distributions for comparing coefficients and obtaining p-values through the GLM output. The testing and interpretation of coefficients remain similar to linear regression, but the underlying mathematics is more involved. It’s essential to note that these results are based on asymptotics, requiring larger sample sizes. In cases with small sample sizes, p-values may not be applicable.

This overview provides a glimpse into GLMs. In the upcoming sections, the focus will shift to the two crucial cases: binomial and Poisson regression, specifically logistic regression for binary outcomes.

## 4.2 Logistic Regression

Binary GLMs come from trying to model outcomes that can take only two values. Some examples include: survival or not at the end of a study, winning versus losing of a team and success versus failure of a treatment or product. Often these outcomes are called Bernoulli outcomes, from the Bernoulli distribution named after the famous probabilist and mathematician.

If we happen to have several exchangeable binary outcomes for the same level of covariate values, then that is binomial data and we can aggregate the 0’s and 1’s into the count of 1’s. As an example, imagine if we sprayed insect pests with 4 different pesticides and counted whether they died or not. Then for each spray, we could summarize the data with the count of dead and total number that were sprayed and treat the data as binomial rather than Bernoulli.

In this section, the focus is on scenarios where the outcome is a binary random variable, taking values of zero or one. Such situations arise frequently in various analyses, such as survival studies where individuals are either alive or deceased at the study’s conclusion, or in sports analytics, where wins or losses are recorded. The goal is to model the data as a series of coin flips, where the probability of success depends on a set of covariates. When dealing with collections of zeros and ones with a constant success probability and independence, the total number of successes or failures follows a binomial distribution. Binary logistic regression is employed to handle binomial random variables in the special case where the covariate is constant. Here we will cover two cases: binary and binomial, with the focus initially on the binary case. To illustrate, a motivating example is presented using a dataset organized by Jeff Leak, an instructor in the data science specialization.

download.file("https://raw.githubusercontent.com/B7M/Course\_data/main/ravensData.csv"  
 , destfile="ravensData.csv",method="auto")  
ravensData <- read.csv("ravensData.csv")  
head(ravensData)

## ravenWinNum ravenWin ravenScore opponentScore  
## 1 1 W 24 9  
## 2 1 W 38 35  
## 3 1 W 28 13  
## 4 1 W 34 31  
## 5 1 W 44 13  
## 6 0 L 23 24

The dataset involves the Ravens’ football team, where wins and losses are coded as ones and zeros, respectively. The analysis explores the extent to which the Ravens’ score predicts their winning outcomes. Linear regression is not suitable for this type of data. Attempting to model the probability of a Ravens win as a linear progression would not accurately capture the underlying relationships. While fitting binary data with a linear model is sometimes a preliminary step, it is typically followed by more appropriate models like logistic regression. The potential issues with fitting binary data using linear regression include: \* $ RW\_i = b\_0 + b\_1 RS\_i + e\_i $ \* - 1 if a Ravens win, 0 if not \* - Number of points Ravens scored \* - probability of a Ravens win if they score 0 points \* - increase in probability of a Ravens win for each additional point \* - residual variation due

While it’s not the worst thing to fit binary data with linear regression, especially for quick analyses, a preferable approach is to model the odds. In the context of binary outcomes, such as whether the Ravens win, the goal is to model the probability of winning as if it were a series of coin flips. The success probability varies from game to game based on factors like the points the Ravens score. The odds, denoted as the probability over one minus the probability, capture this relationship. The odds can be converted back to probability and vice versa. To quantify the odds, the log odds, also known as the logit, which is introduced as the logarithm of the odds the model is placed on the logit scale. Logistic regression is preferred over linear regression due to the clear violation of linear regression assumptions and the potential inaccuracies in inferences. To draw a connection between the two, linear regression models can be expressed as the expected value of the outcome being a linear relationship. In the case of binary outcomes, the expected value is the probability, highlighting the transition from linear to logistic regression. The logistic regression model involves predicting the log odds, allowing for a more appropriate representation of binary data. The expected value of a fair coin is 0.5, and instead of directly modeling the expected value of the outcome in logistic regression, the approach is to model the expected value, which is the probability, using the equation exp(the linear regression) over 1 + exp(the linear regression). When this equation is inverted, it results in the log of the odds being the linear regression relationship. In binary generalized linear models, particularly in logistic regression, the data is treated as a series of coin flips with changing success probabilities related to covariates. The log of the odds, also known as the logit, is crucial in this modeling process.

The function that goes backward from the log of the odds to the probability is referred to as the xbit, and represents this inversion. The clever aspect of generalized linear models lies in not directly placing the model on the scale but recognizing the dependence on a probability distribution, which, in turn, is influenced by the success probability that is connected to regressors.

Interpreting logistic regression is straightforward.

$$ \log\left(\frac{\rm{Pr}(RW\_i | RS\_i, b\_0, b\_1 )}{1-\rm{Pr}(RW\_i | RS\_i, b\_0, b\_1)}\right) = b\_0 + b\_1 RS\_i $$

If the Ravens score is zero, the log odds () represent the log odds of a Ravens win. The probability of winning can then be calculated using . However, an imperfect aspect arises when b0 is estimated to be non-zero, despite the team not being able to win with zero points. The slope coefficient (b1) is interpreted as the increase or decrease in the log odds of the probability associated with a one-unit increase in the regression variable (in this case, a one-point increase in score). If we exponentiate the difference on the log scale in logistic regression, we obtain a ratio on the original scale. Exponentiating results in being the ratio of the odds, comparing the odds of the Ravens winning for a one-point increase in the score. While there may not be multiple regressors in this particular example, the extension to cases with multiple covariates is direct. The interpretation remains consistent with holding the other regression variables fixed, just like in linear regression. In logistic regression, everything is interpreted under the assumption that other covariates are constant. For instance, if additional covariates, such as whether it was a home or away game, were present, would represent the odds ratio of a Ravens win for a one-unit increase in the score, holding the status of the game (home or away) constant. The interpretability of logistic regression coefficients is straightforward due to the logarithmic function, which allows the inversion of additive effects into ratios. This interpretation in terms of ratios is convenient and aligns well with how people naturally discuss relationships.

A brief historical note on odds: It originates from the concept of a fair game. Consider a scenario where you flip a coin with a success probability p, winning X dollars if it lands heads, and losing Y dollars if it lands tails. To make the game fair, the expected earnings (winning given heads minus losing given tails) should be equal to zero. Solving for fairness leads to the relationship . This equation implies that odds can be seen as how much one would be willing to pay for a p probability of winning a dollar. In summary, the concept of odds is deeply rooted in the idea of a fair game, where when X is set to 1. This ratio determines how much one would be willing to pay for a certain probability of winning a dollar. For instance, at a horse track, odds like 10 to 1 indicate that if you bet a dollar and the horse wins, you receive $10. The setting of horse betting probabilities adapts as bets come in, reflecting a balance of money in and out to ensure the house neither wins nor loses. Over time, the aggregate odds from betting often serve as a fairly accurate representation of the percentage of times a horse wins.

Casinos or tracks make money by taking a fee for every bet, ensuring they are financially neutral regardless of the outcome. This fee ensures that the house is guaranteed to win, making the phrase “the house wins” quite literal. The complexity of the betting system involves various factors, but the basic principle remains: the house sets up odds to be financially neutral and takes a small fee from each bet.

Run this in RStudio which is a demonstration using the manipulate package in R, logistic regression curves are visualized with varying values of beta0 (intercept) and beta1 (slope). At the time of developing this course, the manipulate package was not available in OTTR. However, the code is provided for students to run in RStudio.

library(manipulate)  
x <- seq(-10, 10, length = 1000)  
manipulate(  
 plot(x, exp(beta0 + beta1 \* x) / (1 + exp(beta0 + beta1 \* x)),   
 type = "l", lwd = 3, frame = FALSE),  
 beta1 = slider(-2, 2, step = .1, initial = 2),  
 beta0 = slider(-2, 2, step = .1, initial = 0)  
 )

The logistic regression curve resembles an ‘s’ shape, and changes in beta1 result in alterations in the curve’s shape—becoming more peaked for higher values and flipping around for negative values. The beta0 parameter shifts the curve left or right along the x-axis. If the x-axis represents regressors with zeros and ones, R attempts to find the logistic curve that best aligns with the associated probabilities. The principle behind this is maximum likelihood estimation. The logistic regression model, represented as , is used to convert the logistic curve back to the probability scale.

Essentially, logistic regression aims to find the best-fitting curve by adjusting values. The fitted function captures the logistic regression model on the probability scale. This visualization helps illustrate the underlying process of logistic regression and how R optimizes the logistic curve to match the observed data probabilities.

Now that we have examined the logistics regression curves, let’s explore the outcomes when we apply these curves to our Ravens data. To do this, you use the GLM function to feed the Ravens data.

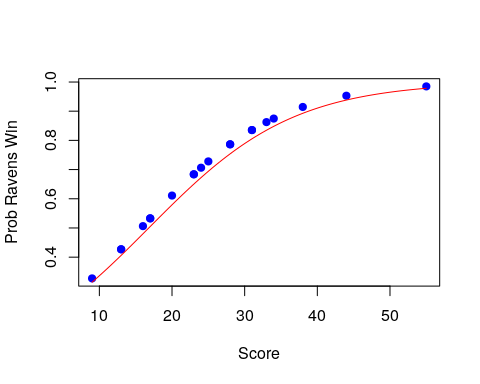
logRegRavens <- glm(ravensData$ravenWinNum ~ ravensData$ravenScore,family="binomial")  
summary(logRegRavens)

##   
## Call:  
## glm(formula = ravensData$ravenWinNum ~ ravensData$ravenScore,   
## family = "binomial")  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.7575 -1.0999 0.5305 0.8060 1.4947   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -1.68001 1.55412 -1.081 0.28  
## ravensData$ravenScore 0.10658 0.06674 1.597 0.11  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 24.435 on 19 degrees of freedom  
## Residual deviance: 20.895 on 18 degrees of freedom  
## AIC: 24.895  
##   
## Number of Fisher Scoring iterations: 5

The glm function function is similarly to the lm function, with the response variable on the left side of the tilde and the predictors on the right side. The key difference here is that we need to specify “family equals binomial” to inform the glm function that we are working with binary data (0,1). If dealing with count data or real binomial data, a sample size would also need to be specified. By default, for binomial and binary cases, the glm function assumes the logistic link function, which is appropriate.

When reviewing the summary output, it mirrors the format used for the lm function. The coefficients for the Ravens data are displayed, such as -1.68 and 0.1 for the intercept and Ravens coefficient, respectively. On the Logit scale, it’s important to examine whether the coefficients are close to zero for variables like the Ravens score, and on the exponentiated scale, whether they are close to one. The summary includes standard errors, Z values, and P values, which are interpreted similarly to a linear model, recognizing the differences in interpretation.

plot(ravensData$ravenScore,logRegRavens$fitted,pch=19,col="blue",xlab="Score",ylab="Prob Ravens Win")  
curve(exp(-1.68 + 0.1\*x)/(1 + exp(-1.68 + 0.1\*x)),add=TRUE,col="red")

 The fitted curve represents the predicted responses converted back to the probability scale. R calculates this by substituting the x values associated with the coefficients, multiplying them by the respective coefficients, adding the estimated intercept, and applying the logistic function. The result is the probability values. It’s crucial to note that the displayed fitted S curve is only a portion, as the component where the data are observed is restricted. The S curve depicted earlier has a starting point at one, resembling a curve that ascends and eventually descends, concluding at 0.4 instead of zero. This signifies the actual curve fitting the data, although the displayed fitted values only show a portion of it. Exponentiating the Ravens coefficients yields 0.1864 for the intercept and 1.1125 for the score, indicating an 11% increase in the probability of winning for each additional point the Ravens score. Interpreting this logistic regression coefficient involves assessing the odds ratios. Confidence intervals for these coefficients are easily obtained using the confint operator. The preferred method for viewing these intervals on the exponentiated scale involves using the x function to exponentiate the two endpoints.

exp(logRegRavens$coeff)

## (Intercept) ravensData$ravenScore   
## 0.1863724 1.1124694

exp(confint(logRegRavens))

## Waiting for profiling to be done...

## 2.5 % 97.5 %  
## (Intercept) 0.005674966 3.106384  
## ravensData$ravenScore 0.996229662 1.303304

The resulting interval, ranging from 0.99 to 1.3, suggests that despite the established influence of scoring points on the Ravens’ game outcomes, this coefficient is not statistically significant.

The ANOVA function operates similarly to lm, taking the output of the fitted model and accommodating multiple models, including nested ones. It performs sequential tests, and in this case, a one Degree of Freedom test is conducted for the score variable, although its utility is limited in this specific example. ANOVA becomes more valuable when assessing several models or dealing with factor variables. It helps determine if a factor variable should be included or excluded, considering all levels collectively, in contrast to the independent testing of each level when using the summary function.

anova(logRegRavens,test="Chisq")

## Analysis of Deviance Table  
##   
## Model: binomial, link: logit  
##   
## Response: ravensData$ravenWinNum  
##   
## Terms added sequentially (first to last)  
##   
##   
## Df Deviance Resid. Df Resid. Dev Pr(>Chi)   
## NULL 19 24.435   
## ravensData$ravenScore 1 3.5398 18 20.895 0.05991 .  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

When interpreting odds ratios, it’s crucial to note that they represent functions of probabilities, not probabilities themselves. An odds ratio of one indicates no difference, while on the Logit scale, a log odds ratio of zero implies no difference. Understanding these nuances is essential in the interpretation of logistic regression coefficients. An odds ratio below 0.5 or above 2 is often considered indicative of a strong effect. However, the significance of these values depends heavily on the context. In fields like epidemiology, where studies often involve noisy factors like nutrition and its impact on health, even a small odds ratio such as 1.01 might be significant in large-scale studies. On the other hand, in tightly controlled experimental clinical trials, larger odds ratios may be required to deem an effect meaningful. While the benchmarks of less than 0.5 or greater than 2 provide some guidance, their utility is limited and varies based on the scientific context.

The relative risk is another metric similar to the odds ratio and is favored by those who think more intuitively in terms of probabilities $\frac{\rm{Pr}(RW\_i | RS\_i = 10)}{\rm{Pr}(RW\_i | RS\_i = 0)}$. It represents the ratio of two probabilities. Unlike odds ratios, relative risk introduces some model constraints, making relative risk regression less common for binary variables. In cases with small probabilities, the relative risk can approximate odds ratios, leading some to fit odds ratios but interpret them as if they were relative risks.

## 4.3 Poisson Regression

### 4.3.1 Data count

Many data take the form of unbounded count data. For example, consider the number of calls to a call center or the number of flu cases in an area or the number of hits to a web site.

In some cases the counts are clearly bounded. However, modeling the counts as unbounded is often done when the upper limit is not known or very large relative to the number of events.

If the upper bound is known, the techniques we are discussing can be used to model the proportion or rate. The starting point for most count analysis is the the Poisson distribution.

In the following chapter, we go over some of the basics of modeling count data.

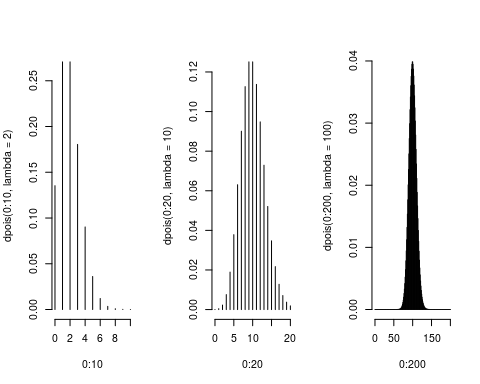
### 4.3.2 Poisson Regression

This section focuses on Poisson Generalized Linear Models (GLMs), and credit is attributed to Jeff Leek for contributing much of the content from a previous version of the class. Count data modeling is a common need in various applications, such as the number of calls to a call center or the occurrences of flu cases. Count data can also be expressed as rates or proportions, like the percentage of people passing a test or the rate of cases over a specific time period, as seen in bio-statistics and public health, for instance, with incidence rates. Count, rates, and proportions all fall under the purview of Poisson GLMs. The Poisson distribution serves as a suitable model for count and rate data, where rates are counts per monitoring time. It is frequently used in modeling incidence rates, web traffic, and other scenarios. The Poisson distribution is also applied to approximate binomial probabilities when the success probability is very small, and the sample size is large. Contingency table data, involving counts of occurrences for different variables, is another area where Poisson models excel. A contingency table, illustrating counts of occurrences for various combinations of variables, finds an elegant framework in Poisson models. The Poisson mass function, presented in the lecture, defines the rate of counts per unit time as lambda, with t representing the total time. This distribution proves versatile for handling a variety of count data scenarios and provides a robust framework for statistical modeling.

The Poisson mass function is defined as: For .

If x represents a positive value with a particular significance, its expected value equals . Consequently, our natural estimate of the rate is the count divided by the total time, denoted as . It’s noteworthy that in this scenario, the , our rate estimate, precisely equals lambda—the desired rate of estimation. The assumption underlying our model is that the variance equals the mean, which is . We can verify this assumption and explore potential solutions if it doesn’t hold. An interesting observation is that the Poisson distribution approaches a normal distribution as the mean increases. This can happen if is fixed and increases, if is fixed and increases, or if both and increase. In various applications, the means may increase differently, but as long as they increase significantly, the Poisson distribution approximates a normal distribution. We illustrate this through simulations, presenting three sets of Poisson random variables as the mean of the distribution grows larger.

par(mfrow = c(1, 3))  
plot(0 : 10, dpois(0 : 10, lambda = 2), type = "h", frame = FALSE)  
plot(0 : 20, dpois(0 : 20, lambda = 10), type = "h", frame = FALSE)  
plot(0 : 200, dpois(0 : 200, lambda = 100), type = "h", frame = FALSE)



The rightmost panel demonstrates the close resemblance to a normal distribution. Additionally, we can theoretically demonstrate that the mean and variance are equal. While this might not be suitable for this class, I encourage experimenting with simulations or exploring mathematical biostatistics courses, such as Brian Caffo’s other courses, for a detailed understanding of the mathematics behind this concept.

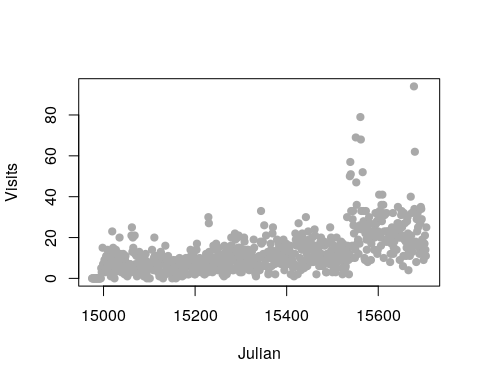
As an example, let’s analyze Jeff Leek’s web traffic on his [website](www.biostat.jhsph.edu/~jleek). The variable of interest in this case is the number of web hits per day, with the unit of time set as equals one day. To interpret the data as web hits per hour, would need to be set to 24. Similarly, for seconds, it would be , and so on.

download.file("https://raw.githubusercontent.com/B7M/Course\_data/main/gaData.csv",destfile="gaData.csv",method="auto")  
gaData <- read.csv("gaData.csv")  
gaData$date <- as.Date(gaData$date)  
gaData$julian <- julian(gaData$date)  
head(gaData)

## date visits simplystats julian  
## 1 2011-01-01 0 0 14975  
## 2 2011-01-02 0 0 14976  
## 3 2011-01-03 0 0 14977  
## 4 2011-01-04 0 0 14978  
## 5 2011-01-05 0 0 14979  
## 6 2011-01-06 0 0 14980

After downloading the data, the date is converted from a standard character date-time format to a Julian date, counting the days since January 1st, 1970. The dataset includes the date, number of visits, and Julian date. Early dates show zero visits. A plot of the dataset illustrates the relationship between Julian date and the number of visits.

plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")



Considering linear regression for modeling count data, as discussed in the last lecture, there are challenges. However, with larger mean counts, concerns decrease due to the tendency of counts towards a normal distribution. Here we will define the following model:

- number of hits to the website

- day of the year (Julian day)

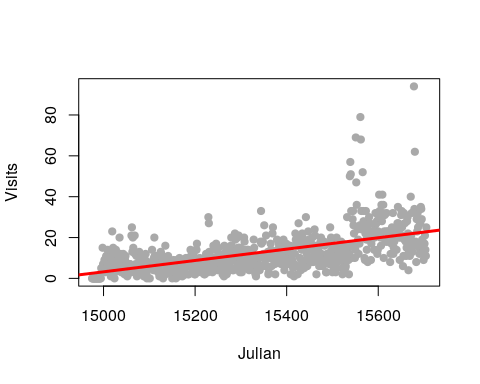
- number of hits on Julian day 0 (1970-01-01)

- increase in number of hits per unit day

- variation due to everything we didn’t measure

And fit a linear regression model.

plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")  
lm1 <- lm(gaData$visits ~ gaData$julian)  
abline(lm1,col="red",lwd=3)



The fitted line shows some curvature, suggesting potential issues. While this initial approach may not be ideal for small counts, it serves as a starting point. Linear models may lack optimal interpretation for count data, prompting exploration of ways to improve interpretation in subsequent slides. Viewing counts as relative scales, rather than linear additive scales, is considered for a more meaningful interpretation of linear models. To this end we take the natural logarithm of the outcome variable. The model is expressed as

where are coefficients, is the Julian date, and represents the error term. The logarithm has a meaningful interpretation in terms of geometric means. The exponential value of the logarithm’s expected value for a random variable is what we refer to as the population geometric mean, . This term is derived from the empirical or geometric mean, which is the product of a sample represented as . To clarify, this is equivalent to the product of to the power of one over n. Taking the logarithm of this yields the arithmetic mean or the average of the log data. In essence, the geometric mean is obtained by exponentiating the arithmetic mean of the log data. As we gather more data in our sample, the arithmetic mean tends to converge to a certain value, and the geometric mean represents what this quantity, the product of the data raised to the power of , converges to. When we apply the natural logarithm to the outcome in a linear regression, the exponentiated coefficients become interpretable in terms of geometric means. For instance, signifies the estimated geometric mean hits on day zero.

round(exp(coef(lm(I(log(gaData$visits + 1)) ~ gaData$julian))), 5)

## (Intercept) gaData$julian   
## 0.00000 1.00231

It’s important to note, as emphasized earlier, that the intercept () might not carry significant meaning since January 1, 1970, is not a relevant date in terms of web hits. To enhance interpretability, a better approach would have been to subtract the earliest date observed in the dataset and begin counting days from there. Consequently, the intercept would then represent the exponentiated value of the estimated intercept as the geometric mean hits on the first day of the dataset. The adjustment in the intercept is a minor modification that doesn’t alter the fitted model’s slope or other aspects. Nevertheless, for those seeking an interpretable intercept such adjustments become relevant. on the other hand, represents the estimated proportional increase or decrease in the geometric mean hits per day. It’s crucial to address an issue with logarithms when dealing with zero counts. Taking the logarithm of zero is undefined, so a common solution is to add a constant, often plus one, to the counts. Thus, we perform the log of the outcome plus one. In a linear model fitted to the log of the outcome plus one against the Julian date, the intercept, as mentioned earlier, may not carry significant meaning. However, the coefficient of 1.002, when exponentiated, indicates a 0.2% daily increase in web traffic according to our model. This interpretation holds if no additional covariates are considered. Introducing other covariates would lead to a 0.02% daily increase, keeping the other covariates constant.

### 4.3.3 Linear vs. Poisson regression

**Linear**

$ NH\_i = b\_0 + b\_1 JD\_i + e\_i $ or $ E[NH\_i | JD\_i, b\_0, b\_1] = b\_0 + b\_1 JD\_i$

**Poisson/log-linear**

$ (E[NH\_i | JD\_i, b\_0, b\_1]) = b\_0 + b\_1 JD\_i $ or $ E[NH\_i | JD\_i, b\_0, b\_1] = (b\_0 + b\_1 JD\_i) $

In Generalized Linear Models (GLMs), the focus isn’t directly on the outcome or its transformation; instead, we concentrate on the transformation of the mean of the outcome. In linear models, the outcome is represented as the linear component plus an error, or equivalently, the expected value of the outcome is the linear component. On the other hand, in a Poisson/ log-linear model, the linear part is the log of the expected outcome. For example, the log of the expected number of web hits per day is expressed as . By exponentiating both sides of this equation, we can state that the mean web hits per day depend on . The key distinction lies in assuming that our data follows a Poisson distribution with a mean defined as . This alteration significantly affects the interpretation of the model, providing a more plausible distribution for observed outcomes. Moreover, since everything is logged, the coefficients are interpreted in a relative sense, similar to when we logged the outcome. This approach mitigates issues encountered in the previous example, such as taking logs of 0. It’s worth emphasizing the utility of taking logs of outcomes, which is a valuable technique for count data and also for general regression. When dealing with positive data, the logarithmic transformation is often one of the most beneficial. It maintains or even enhances the interpretability of coefficients on the log scale. However, when considering other transformations such as square root or cube root, the complexities arise. It’s crucial to remember that the differences observed in transformed data are multiplicative when returning to the natural scale.

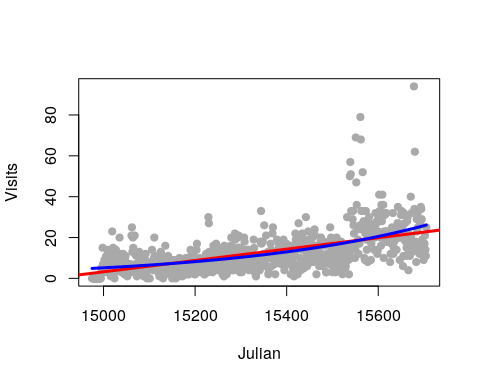
Examining the model where the expected value of the outcome is , by leveraging the properties of expected value, we can factor out .

$ E[NH\_i | JD\_i, b\_0, b\_1] = (b\_0 )(b\_1 JD\_i) $

If we explore the expected mean for the next day (Julian date plus one), represented as , dividing this by the current expected mean yields . Consequently, the coefficient is interpreted as the relative increase or decrease in the mean for a one-unit change in the regressor. Exponentiating the coefficients allows us to assess whether they are close to 1, while leaving them on the log scale enables an evaluation of their proximity to 0. In the multivariate setting, signifies the expected relative increase or decrease in web traffic, holding the other coefficients constant.

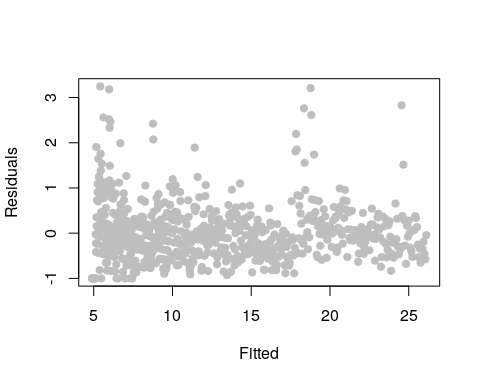
Moving on to the fitted Poisson regression model overlaid onto the data, it closely aligns with the linear model but exhibits desired curvature.

plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")  
glm1 <- glm(gaData$visits ~ gaData$julian,family="poisson")  
abline(lm1,col="red",lwd=3); lines(gaData$julian,glm1$fitted,col="blue",lwd=3)



This curvature could have been achieved in the linear model by introducing a squared term, yet it’s noteworthy that a simpler model with fewer coefficients seems to offer a better fit. One common concern is the requirement for variance to equal the mean. However, when examining the plot of fitted values versus residuals, it becomes apparent that the variance is higher for lower mean values, indicating a potential issue.

plot(glm1$fitted,glm1$residuals,pch=19,col="grey",ylab="Residuals",xlab="Fitted")



Thus, there’s a need for a mechanism to address the non-constant variance. Addressing the non-constant variance in models like the Poisson regression can be approached through various methods. One approach discussed in the book is the quasi-Poisson model, which allows for the variance to be a constant multiple of the mean rather than strictly equal to it. However, in cases like the one presented, where there’s larger variance for lower fitted values, indicating a violation of the Poisson model assumption, alternative solutions are necessary.

Jeff shared code for a model-agnostic standard errors approach using the sandwich package.

library(sandwich)

## Warning: package 'sandwich' was built under R version 4.0.3

confint.agnostic <- function (object, parm, level = 0.95, ...)  
{  
 cf <- coef(object); pnames <- names(cf)  
 if (missing(parm))  
 parm <- pnames  
 else if (is.numeric(parm))  
 parm <- pnames[parm]  
 a <- (1 - level)/2; a <- c(a, 1 - a)  
 pct <- stats:::format.perc(a, 3)  
 fac <- qnorm(a)  
 ci <- array(NA, dim = c(length(parm), 2L), dimnames = list(parm,  
 pct))  
 ses <- sqrt(diag(sandwich::vcovHC(object)))[parm]  
 ci[] <- cf[parm] + ses %o% fac  
 ci  
}

The sandwich variance estimator, famous for generalized estimating equations, originated at Johns Hopkins Biostatistics. This technique, although a bit advanced, is crucial in practice. It involves conducting residual plots to assess model assumptions. If the quasi-Poisson model is deemed suitable, especially when assumptions about variance being a constant multiple of the mean are not strictly met, it can be easily implemented in R.

confint(glm1)

## Waiting for profiling to be done...

## 2.5 % 97.5 %  
## (Intercept) -34.346577587 -31.159715656  
## gaData$julian 0.002190043 0.002396461

confint.agnostic(glm1)

## 2.5 % 97.5 %  
## (Intercept) -36.362674594 -29.136997254  
## gaData$julian 0.002058147 0.002527955

For cases where model assumptions are significantly violated, more advanced solutions may be required. In the example presented, confidence intervals are shown without any adjustment and with the model-agnostic approach. While there’s not a substantial difference in this particular case, it’s essential to explore these options when necessary. Both confidence intervals presented are in their non-exponentiated form. This advanced topic underscores the importance of thoroughly examining model assumptions and employing appropriate techniques to address issues arising in practical applications. Exponentiating coefficients, especially for small ones, essentially adds 1. For instance, if the coefficient is around 0.002, exponentiating it might be approximately 1.002. Continuing with the example, this translates to about a 0.2% increase on the lower end, and a 0.3% increase per day on the high end.

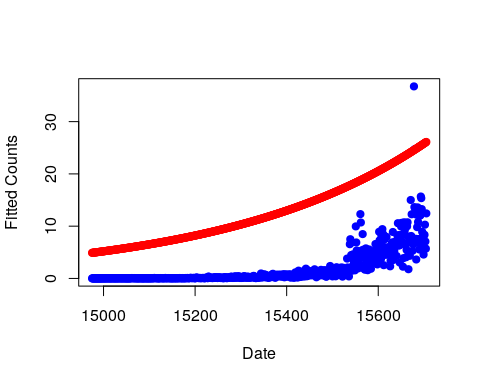
### 4.3.4 How should we handle rates and proportions?

When dealing with rates and proportions, it’s essential to distinguish between the two. Rates involve counts with an associated offset that helps interpret the count relative to some factor, such as time, population size, or sample size. $ E[NHSS\_i | JD\_i, b\_0, b\_1]/NH\_i = (b\_0 + b\_1 JD\_i) $

In the example of web hits, Jeff is interested in the number of web hits from Simply Statistic relative to the total number of web hits, modeling this proportion as . To achieve this in R, the key is to interpret the expected value of the outcome divided by the relative term.

By taking the log of both sides and manipulating the equation, a similar log-linear model is obtained, featuring the log of the outcome as the linear regression part and a log offset without a coefficient. This simple addition of a log offset with no coefficient is all that is required to incorporate a regular proportion into a Poisson Generalized Linear Model (GLM).

glm2 <- glm(gaData$simplystats ~ julian(gaData$date),offset=log(visits+1),  
 family="poisson",data=gaData)  
plot(julian(gaData$date),glm2$fitted,col="blue",pch=19,xlab="Date",ylab="Fitted Counts")  
points(julian(gaData$date),glm1$fitted,col="red",pch=19)

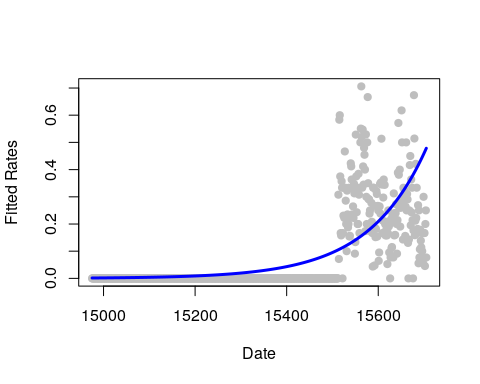


To incorporate the relative denominator count or time into the linear model for rates and proportions, you can easily add it as a log offset. An effective way to do this is by using the term offset = log(visits + 1) in the GLM function. The +1 is added to avoid issues with taking the log of 0. Ensure to specify family = Poisson in the model statement, which assumes a log link by default.

Jeff compares two fitted rates using GLM1 and GLM2. GLM1 is based on the number of web hits, while GLM2 is based on the relative number of web hits originating from Simply Statistics. The blue points, adjusted for the red points, represent the fitted model relative to the data, showing a temporal component with numerous zeros early on. For handling zero inflation in Poisson data, various approaches exist, and it’s crucial to consider the specific characteristics of the dataset. Jeff used a package that aids in modeling zero inflation, addressing concerns about an excessive number of zeros.

In the following example the graph presented depicts the fitted model in relation to the data, showcasing a surge in values after an initial period of numerous zeros. The blue line represents the fitted model, while the gray points correspond to the actual data points.

glm2 <- glm(gaData$simplystats ~ julian(gaData$date),offset=log(visits+1),  
 family="poisson",data=gaData)  
plot(julian(gaData$date),gaData$simplystats/(gaData$visits+1),col="grey",xlab="Date",  
 ylab="Fitted Rates",pch=19)  
lines(julian(gaData$date),glm2$fitted/(gaData$visits+1),col="blue",lwd=3)



### 4.3.5 More information

* [Log-linear models and multiway tables](http://ww2.coastal.edu/kingw/statistics/R-tutorials/loglin.html)
* [Wikipedia on Poisson regression](http://en.wikipedia.org/wiki/Poisson_regression), [Wikipedia on overdispersion](http://en.wikipedia.org/wiki/Overdispersion)
* [Regression models for count data in R](http://cran.r-project.org/web/packages/pscl/vignettes/countreg.pdf)
* [pscl package](http://cran.r-project.org/web/packages/pscl/index.html) - the function *zeroinfl* fits zero inflated models.

## 4.4 Hodgepodge

This section is a bit of mishmash of interesting things that one can accomplish with linear models.

In this final section, we aim to inspire you to continue exploring regression models and linear models, emphasizing their immense importance in data analysis. We will share a couple of interesting insights that extend the capabilities of linear models beyond what we’ve covered so far.

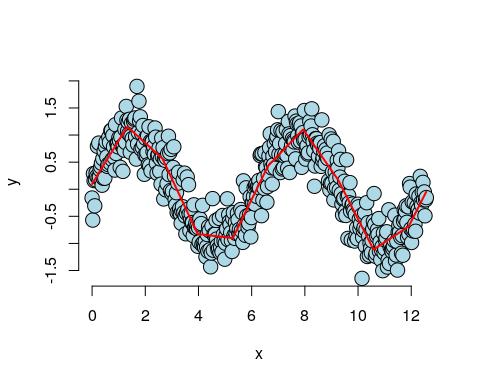
When thinking about extending linear models, the first consideration is fitting complex functions using regression models. While we’ve explored adding squared and cubic terms to capture lower-order functions, what if the function is more intricate, like a sine curve? How can we fit such non-parametric functions using a linear model? One approach is to specify our model as , where represents a potentially complicated function. Surprisingly, there’s a relatively straightforward way to achieve this, and we will introduce you to a simple initial step. Keep in mind that there’s an entire literature on this topic, and we’ll cover some fundamental concepts.

The first step involves using something called regression splines. The model for includes an intercept (), a regular slope term (), and additional terms involving the sum of terms of the form

where if and otherwise and are known knot points.

The idea behind regression splines involves creating kinks or breaks in a stick rather than breaking it apart. Imagine having a stick, and instead of breaking it, you introduce kinks at specific points. Now, if you aim to fit a function to your data, and the data exhibits a pattern like zigzag, you might want to create breaks at certain points to capture the complexity of the function. Mathematically modeling these breaks is precisely what the little plus functions do in the context of regression splines. The process involves introducing these “kinks” at selected points in the data to better represent the underlying function. It’s worth noting that these functions help ensure continuity in the overall model. If you’re interested, you can delve deeper into the mathematical aspects and prove to yourself that the resulting function remains continuous despite the introduced breaks. This continuity is a crucial aspect of the approach, allowing for a more flexible and accurate representation of complex functions in the data. The easiest way to show this would be to go through an example.

n <- 500; x <- seq(0, 4 \* pi, length = n); y <- sin(x) + rnorm(n, sd = .3)  
knots <- seq(0, 8 \* pi, length = 20);   
splineTerms <- sapply(knots, function(knot) (x > knot) \* (x - knot))  
xMat <- cbind(1, x, splineTerms)  
yhat <- predict(lm(y ~ xMat - 1))  
plot(x, y, frame = FALSE, pch = 21, bg = "lightblue", cex = 2)  
lines(x, yhat, col = "red", lwd = 2)



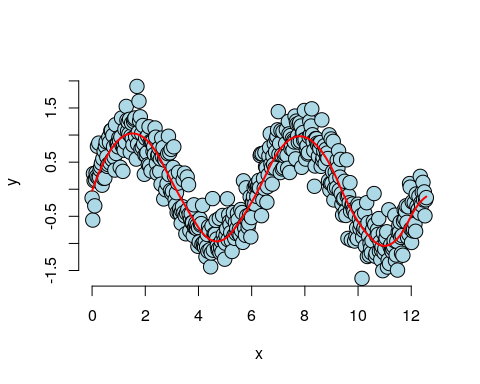
In this example, simulated data is generated, representing a sine curve with added noise, as seen in the blue points on the plot. The goal is to fit a model to this data using regression splines. To achieve this, a set of knots (breakpoints where the line will be continuous but not smooth) is calculated. In this case, 20 knots are evenly spaced along the range of the collected data. The creation of the matrix of knot terms is illustrated, showcasing how to implement the function discussed earlier. This matrix serves as a basis for constructing the x matrix, which includes an intercept term, a slope coefficient term (x by itself), and the spline terms. The model is then fitted using these predictors, with the intercept excluded from the model since it’s already incorporated in the x matrix. The resulting fitted plot looks promising, effectively capturing the sine curve. However, one notable aspect is that the fit appears sharp at the knots. Mathematically, this occurs because although the function used is continuous, it lacks continuity in its derivatives, making it not continuously differentiable. This characteristic may lead to sharp transitions at the knot points, which can be a consideration when applying regression splines to capture smooth functions.

To address the issue of a non-continuous derivative at the knot points and achieve a smoother curve, a simple trick involves adding square terms. Instead of having a function with x minus the knot points, marked with a little plus symbol, you square that difference.

The updated function maintains the same structure as before but incorporating the square term.

Mathematically, the squared term ensures that if the x-coordinate is beyond the knot point, the difference is squared, creating a smooth transition. The function is defined as if x is greater than the knot point, and zero otherwise.

splineTerms <- sapply(knots, function(knot) (x > knot) \* (x - knot)^2)  
xMat <- cbind(1, x, x^2, splineTerms)  
yhat <- predict(lm(y ~ xMat - 1))  
plot(x, y, frame = FALSE, pch = 21, bg = "lightblue", cex = 2)  
lines(x, yhat, col = "red", lwd = 2)



The code remains almost identical to the previous version, with the addition of squaring the difference for points beyond the knot. When fitting the model with this modification, the resulting curve appears smooth, effectively addressing the discontinuity issue at the knot points. It’s important to note that this process involves ordinary regression, showcasing the flexibility of regression models in capturing complex functions.

In this example, we’ve successfully fitted a complex function using ordinary regression by incorporating a set of knots. However, there are certain challenges associated with this basic version of regression splines. One key challenge is determining the precise locations of the knot points. Too many or too few knot points can introduce potential issues. To address these challenges, there are more advanced techniques and solutions. It’s essential to recognize that the collection of sine terms introduced earlier is termed a basis, representing building blocks for functions. While regression splines are one way to create a basis, there are various other approaches. Some popular bases include the Fourier basis (which will be discussed shortly), the wavelet basis, and spline bases like the regression splines we’ve explored. Different bases have distinct strengths and weaknesses, and researchers invest considerable effort in developing and understanding these foundations. If the goal is to fit data with a specific shape, such as a hockey stick, having just one knot point may be sufficient. This approach can capture abrupt changes in the data, like a hockey stick model, provided the location of the knot point is precisely known. It’s important to note that fitting non-linear functions is not limited to linear models; it can also be extended to generalized linear models (GLMs). The process involves specifying the regression in the linear predictor, allowing for the flexibility to handle non-linear relationships in a more generalized framework. One consistent challenge in this approach is the determination of knot points (naught points). Too few or too many knot points can pose problems. A modern solution to address this challenge involves incorporating many knot points and adding a regularization term. The regularization term penalizes coefficients from spline terms, discouraging excessively large coefficients and helping control the number of parameters in the model. While regularization is a more advanced topic, it is commonly covered in advanced linear modeling or regression modeling classes. By understanding the basics of fitting non-linear functions using linear and generalized linear models, you now have a foundation for further exploration. Experimenting with knot placement and the number of knots is a practical starting point. To delve deeper into this topic, consider taking additional classes in linear models to extend your knowledge and skills.

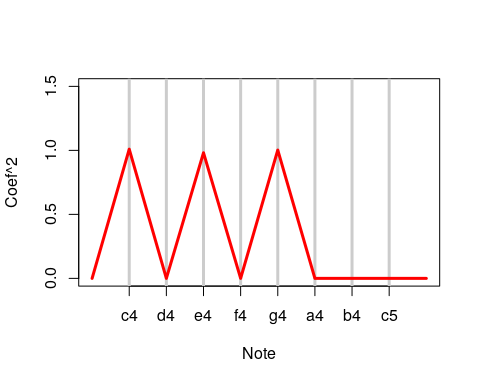
In the next example, the focus is on creating another basis known as the Fourier basis and exploring how to model harmonics using it. The exercise involves imagining a major scale, which consists of eight notes: do, re, mi, fa, so, la, ti, do. A chord, in this context, refers to three notes played together. The question posed is whether, given a continuous chord played for some time and digitized into R, the software would be able to figure out the notes comprising the chord based solely on the sound.

The data for this exercise was generated, and the frequencies for various notes were obtained from the Internet. The frequencies correspond to notes starting from the middle C key on a piano and going up one octave to the C key again. The goal is to take these notes and create digitally sampled time points for analysis. Now, let’s proceed with the analysis of this data to see if R can deduce the notes comprising the chord based on the digitized sound.

##Chord finder, playing the white keys on a piano from octave c4 - c5  
notes4 <- c(261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88, 523.25)  
t <- seq(0, 2, by = .001); n <- length(t)  
c4 <- sin(2 \* pi \* notes4[1] \* t); e4 <- sin(2 \* pi \* notes4[3] \* t);   
g4 <- sin(2 \* pi \* notes4[5] \* t)  
chord <- c4 + e4 + g4 + rnorm(n, 0, 0.3)  
x <- sapply(notes4, function(freq) sin(2 \* pi \* freq \* t))  
fit <- lm(chord ~ x - 1)

The chord is created using three notes: C, E, and G. Each note is generated by creating a sine wave at the corresponding frequency. While R lacks the capability to directly play these frequencies as sound, the resulting chord would sound like a digital representation of a C major chord if played. To analyze the chord, a basis is created using sine functions for every note (eight sine functions in total). A linear model is then fitted with the chord as the outcome and the sine functions as predictors. The intercept is removed since everything is centered to have mean zero. The coefficients obtained from this model are then plotted, revealing that R correctly estimates the coefficients for C, E, and G as being relatively large compared to the other notes.

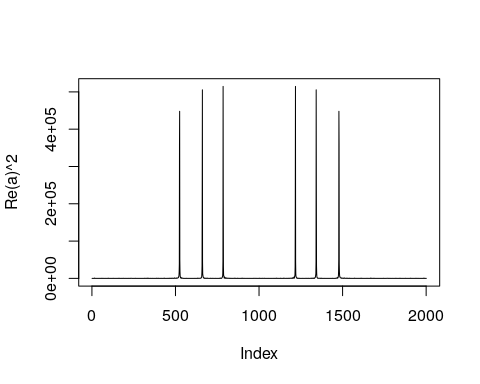
plot(c(0, 9), c(0, 1.5), xlab = "Note", ylab = "Coef^2", axes = FALSE, frame = TRUE, type = "n")  
axis(2)  
axis(1, at = 1 : 8, labels = c("c4", "d4", "e4", "f4", "g4", "a4", "b4", "c5"))  
for (i in 1 : 8) abline(v = i, lwd = 3, col = grey(.8))  
lines(c(0, 1 : 8, 9), c(0, coef(fit)^2, 0), type = "l", lwd = 3, col = "red")



This analysis demonstrates that by utilizing a Fourier basis and fitting a linear model, R can effectively estimate the coefficients for each note, allowing one to identify the chord based on the magnitudes of these coefficients. Here we just used sine waves but in general case we have sine and cosine waves. The Discrete Fourier Transform (DFT) is a mathematical technique used to analyze the frequency content of a time-series signal. It fits a completely saturated linear model by including both sine and cosine terms for all possible frequencies that the given time series allows based on its sampling rate. In the context of a two-second digitally sampled music segment, the DFT calculates coefficients for both sine and cosine components at various frequencies. The “completely saturated model” means that it includes coefficients for all possible frequencies up to the Nyquist frequency, which is half the sampling rate. For example, if there are a thousand time points in the sampled music, the DFT will include a thousand coefficients in the model, covering a range of frequencies. This comprehensive approach allows the DFT to capture both the sine and cosine components of the signal, providing a detailed representation of its frequency spectrum.

To illustrate, the DFT is applied to the chord data, and the square real components of the resulting transform are plotted.

a <- fft(chord);   
plot(Re(a)^2, type = "l")



The plot shows peaks at specific frequencies, and by back-calculating these frequencies, one can identify the notes C, E, and G—the components of the C major chord. This technique is commonly used in music and sound processing, where engineers analyze the spectrum of sound signals using the DFT, essentially performing a linear model with numerous sine and cosine terms as regressors.

We want to emphasize that the discovery of the Discrete Fourier Transform was a significant breakthrough. The reason people prefer using the Fourier Transform over the linear model is due to a discovery by the renowned statistician Tucci. Tucci found a quick way to perform the Fourier transform, known as the fast Fourier transform, which has become the standard method. It’s notably faster than the linear model, making it the preferred choice for sound engineers dealing with Fourier transform problems.

In conclusion, this section has demonstrated two key points. First, we can fit complex functions easily with just a few lines of code. Second, linear models can handle tasks that may initially seem beyond their scope, such as diagnosing notes in a chord. Both of these tasks are not only possible but also quite straightforward with linear models. The examples presented required only a minimal amount of code, showcasing the effectiveness of these techniques. We hope you’ve enjoyed the class and gained valuable insights. As you continue your learning journey, we recommend exploring techniques for generalizing models, which we only briefly touched upon in this class. Additionally, delving into correlated and longitudinal data is essential, as it introduces a crucial aspect not covered in our discussion of independent and exchangeable data. Understanding how to handle correlations is vital for extending linear models to various important topics.

If you decide to pursue further studies, prioritize generalized linear models and longitudinal multi-level data. These areas will enhance your understanding and application of linear models to diverse scenarios. Thank you for participating in the class, and feel free to follow Jeff Leak and Roger Pang on Twitter for updates on new developments in our data science lab.

## 4.5 Practical R Exercises in swirl

During this week of the course you should complete the following lessons in the Regression Models swirl course:

1. Variance Inflation Factors
2. Overfitting and Underfitting
3. Binary Outcomes
4. Count Outcomes

## 4.6 Week 4 Quiz

1. Consider the space shuttle data ?shuttle in the MASS library. Consider modeling the use of the autolander as the outcome (variable name use). Fit a logistic regression model with autolander (variable auto) use (labeled as “auto” 1) versus not (0) as predicted by wind sign (variable wind). Give the estimated odds ratio for autolander use comparing head winds, labeled as “head” in the variable headwind (numerator) to tail winds (denominator).
2. Consider the previous problem. Give the estimated odds ratio for autolander use comparing head winds (numerator) to tail winds (denominator) adjusting for wind strength from the variable magn.
3. If you fit a logistic regression model to a binary variable, for example use of the autolander, then fit a logistic regression model for one minus the outcome (not using the autolander) what happens to the coefficients?
4. Consider the insect spray data InsectSprays. Fit a Poisson model using spray as a factor level. Report the estimated relative rate comapring spray A (numerator) to spray B (denominator).
5. Consider a Poisson glm with an offset, t. So, for example, a model of the form glm(count ~ x + offset(t), family = poisson) where x is a factor variable comparing a treatment (1) to a control (0) and t is the natural log of a monitoring time. What is impact of the coefficient for x if we fit the model glm(count ~ x + offset(t2), family = poisson) where 2 <- log(10) + t? In other words, what happens to the coefficients if we change the units of the offset variable. (Note, adding log(10) on the log scale is multiplying by 10 on the original scale.)
6. Consider the data below. Using a knot point at 0, fit a linear model that looks like a hockey stick with two lines meeting at x=0. Include an intercept term, x and the knot point term. What is the estimated slope of the line after 0?

x <- -5:5  
y <- c(5.12, 3.93, 2.67, 1.87, 0.52, 0.08, 0.93, 2.05, 2.54, 3.87, 4.97)

## 4.7 Course Project

You work for Motor Trend, a magazine about the automobile industry. Looking at a data set of a collection of cars, they are interested in exploring the relationship between a set of variables and miles per gallon (MPG) (outcome). They are particularly interested in the following two questions: \* “Is an automatic or manual transmission better for MPG” \* “Quantify the MPG difference between automatic and manual transmissions”

**Grading Criteria Overview**

*Peer Grading*

* The criteria that your classmates will use to evaluate and grade your work are shown below.
* Each criteria is binary: (1 point = criteria met acceptably; 0 points = criteria not met acceptably)

*Criteria*

* Did the student interpret the coefficients correctly?
* Did the student do some exploratory data analyses?
* Did the student fit multiple models and detail their strategy for model selection?
* Did the student answer the questions of interest or detail why the question(s) is (are) not answerable?
* Did the student do a residual plot and some diagnostics?
* Did the student quantify the uncertainty in their conclusions and/or perform an inference correctly?
* Was the report brief (about 2 pages long) for the main body of the report and no longer than 5 with supporting appendix of figures?
* Did the report include an executive summary?
* Was the report done in Rmd (knitr)?

# About the Authors

These credits are based on our [course contributors table guidelines](https://www.ottrproject.org/more_features.html#giving-credits-to-contributors).

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Lead Content Instructor(s) | [FirstName LastName](link%20to%20personal%20website) |
| Lecturer(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | Delivered the course in some way - video or audio |
| Content Author(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | If any other authors besides lead instructor |
| Content Contributor(s) (include section name/link in parentheses) - make new line if more than one section involved | Wrote less than a chapter |
| Content Editor(s)/Reviewer(s) | Checked your content |
| Content Director(s) | Helped guide the content direction |
| Content Consultants (include chapter name/link in parentheses or word “General”) - make new line if more than one chapter involved | Gave high level advice on content |
| Acknowledgments | Gave small assistance to content but not to the level of consulting |
| **Production** |  |
| Content Publisher(s) | Helped with publishing platform |
| Content Publishing Reviewer(s) | Reviewed overall content and aesthetics on publishing platform |
| **Technical** |  |
| Course Publishing Engineer(s) | Helped with the code for the technical aspects related to the specific course generation |
| Template Publishing Engineers | [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/), [Ava Hoffman](https://www.avahoffman.com/) |
| Publishing Maintenance Engineer | [Candace Savonen](https://www.cansavvy.com/) |
| Technical Publishing Stylists | [Carrie Wright](https://carriewright11.github.io/), [Ava Hoffman](https://www.avahoffman.com/), [Candace Savonen](https://www.cansavvy.com/) |
| Package Developers ([ottrpal](https://github.com/jhudsl/ottrpal)) [Candace Savonen](https://www.cansavvy.com/), [John Muschelli](https://johnmuschelli.com/), [Carrie Wright](https://carriewright11.github.io/) |  |
| **Art and Design** |  |
| Illustrator(s) | Created graphics for the course |
| Figure Artist(s) | Created figures/plots for course |
| Videographer(s) | Filmed videos |
| Videography Editor(s) | Edited film |
| Audiographer(s) | Recorded audio |
| Audiography Editor(s) | Edited audio recordings |
| **Funding** |  |
| Funder(s) | Institution/individual who funded course including grant number |
| Funding Staff | Staff members who help with funding |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value   
## version R version 4.0.2 (2020-06-22)  
## os Ubuntu 20.04.5 LTS   
## system x86\_64, linux-gnu   
## ui X11   
## language (EN)   
## collate en\_US.UTF-8   
## ctype en\_US.UTF-8   
## tz Etc/UTC   
## date 2024-02-09   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] RSPM (R 4.0.5)   
## bookdown 0.24 2023-03-28 [1] Github (rstudio/bookdown@88bc4ea)   
## cachem 1.0.7 2023-02-24 [1] CRAN (R 4.0.2)   
## callr 3.5.0 2020-10-08 [1] RSPM (R 4.0.2)   
## cli 3.6.1 2023-03-23 [1] CRAN (R 4.0.2)   
## crayon 1.3.4 2017-09-16 [1] RSPM (R 4.0.0)   
## desc 1.2.0 2018-05-01 [1] RSPM (R 4.0.3)   
## devtools 2.3.2 2020-09-18 [1] RSPM (R 4.0.3)   
## digest 0.6.25 2020-02-23 [1] RSPM (R 4.0.0)   
## ellipsis 0.3.1 2020-05-15 [1] RSPM (R 4.0.3)   
## evaluate 0.20 2023-01-17 [1] CRAN (R 4.0.2)   
## fastmap 1.1.1 2023-02-24 [1] CRAN (R 4.0.2)   
## fs 1.5.0 2020-07-31 [1] RSPM (R 4.0.3)   
## glue 1.4.2 2020-08-27 [1] RSPM (R 4.0.5)   
## htmltools 0.5.5 2023-03-23 [1] CRAN (R 4.0.2)   
## knitr 1.33 2023-03-28 [1] Github (yihui/knitr@a1052d1)   
## magrittr 2.0.3 2022-03-30 [1] CRAN (R 4.0.2)   
## memoise 2.0.1 2021-11-26 [1] CRAN (R 4.0.2)   
## pkgbuild 1.1.0 2020-07-13 [1] RSPM (R 4.0.2)   
## pkgload 1.1.0 2020-05-29 [1] RSPM (R 4.0.3)   
## prettyunits 1.1.1 2020-01-24 [1] RSPM (R 4.0.3)   
## processx 3.4.4 2020-09-03 [1] RSPM (R 4.0.2)   
## ps 1.4.0 2020-10-07 [1] RSPM (R 4.0.2)   
## R6 2.4.1 2019-11-12 [1] RSPM (R 4.0.0)   
## remotes 2.2.0 2020-07-21 [1] RSPM (R 4.0.3)   
## rlang 1.1.0 2023-03-14 [1] CRAN (R 4.0.2)   
## rmarkdown 2.10 2023-03-28 [1] Github (rstudio/rmarkdown@02d3c25)  
## rprojroot 2.0.3 2022-04-02 [1] CRAN (R 4.0.2)   
## sessioninfo 1.1.1 2018-11-05 [1] RSPM (R 4.0.3)   
## stringi 1.5.3 2020-09-09 [1] RSPM (R 4.0.3)   
## stringr 1.4.0 2019-02-10 [1] RSPM (R 4.0.3)   
## testthat 3.0.1 2023-03-28 [1] Github (R-lib/testthat@e99155a)   
## usethis 1.6.3 2020-09-17 [1] RSPM (R 4.0.2)   
## withr 2.3.0 2020-09-22 [1] RSPM (R 4.0.2)   
## xfun 0.26 2023-03-28 [1] Github (yihui/xfun@74c2a66)   
## yaml 2.2.1 2020-02-01 [1] RSPM (R 4.0.3)   
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library

# 5 References