test

Jeremy

March 27, 2021

Overview

Linear regression provides a means to fit a line to represent the relationship shared between variables. If the true relationship is linear, then this learned linear function can be used as an accurate predictor. Another perspective . . . Linear regression allows us to calculate the conditional mean of the outcome at *every* value of the predictor variable(s). If the predictor takes on just a few values, then that's the number of conditional means that will be calculated. If the predictor is continuous and takes on a large number of values, we'statll still be able to calculate the conditional mean at every one of those values.

The model we posit for regression is as follows:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...\beta_k x_k + \epsilon$$

It's just a linear, additive model. Y increases or decreases as a function of x, with multiple x's included. ϵ is the extent to which an individual value is above or below the line created.

Let's say that you've got some student data and you want to target those students that may struggle in math. The intervention could be targeted based on what we know about students, much of which reflects broader inequalities in our education system, such as the relationship between SES and parental education and test scores. By intervening early we may help to reduce those inequalities.

We're going to be working with data on high school students from the Educational Longitudinal Study. Our goal will be to predict their math scores based on student characteristics.

```
## Warning: package 'tidyverse' was built under R version 3.6.3

## -- Attaching packages --------------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.6.3

## Warning: package 'tibble' was built under R version 3.6.3

## Warning: package 'tidyr' was built under R version 3.6.3

## Warning: package 'purrr' was built under R version 3.6.3

## Warning: package 'dplyr' was built under R version 3.6.3

## Warning: package 'stringr' was built under R version 3.6.3

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

## Warning: package 'ModelMetrics' was built under R version 3.6.3

##
## Attaching package: 'ModelMetrics'
## The following object is masked from 'package:base':
##
##     kappa
```

```
## Warning: package 'modelr' was built under R version 3.6.3
```

```
##
## Attaching package: 'modelr'
```

```
## The following objects are masked from 'package:ModelMetrics':
##
##     mae, mse, rmse
```

The ELS dataset is called `els_train`. I'll explain the "train" part in a bit– it refers to a training dataset.

```r
load("els_train.Rdata")
```

Bivariate regression

Our dependent variable will be math scores, stored in this dataset as `bynels2m`. Let's take a look at this variable

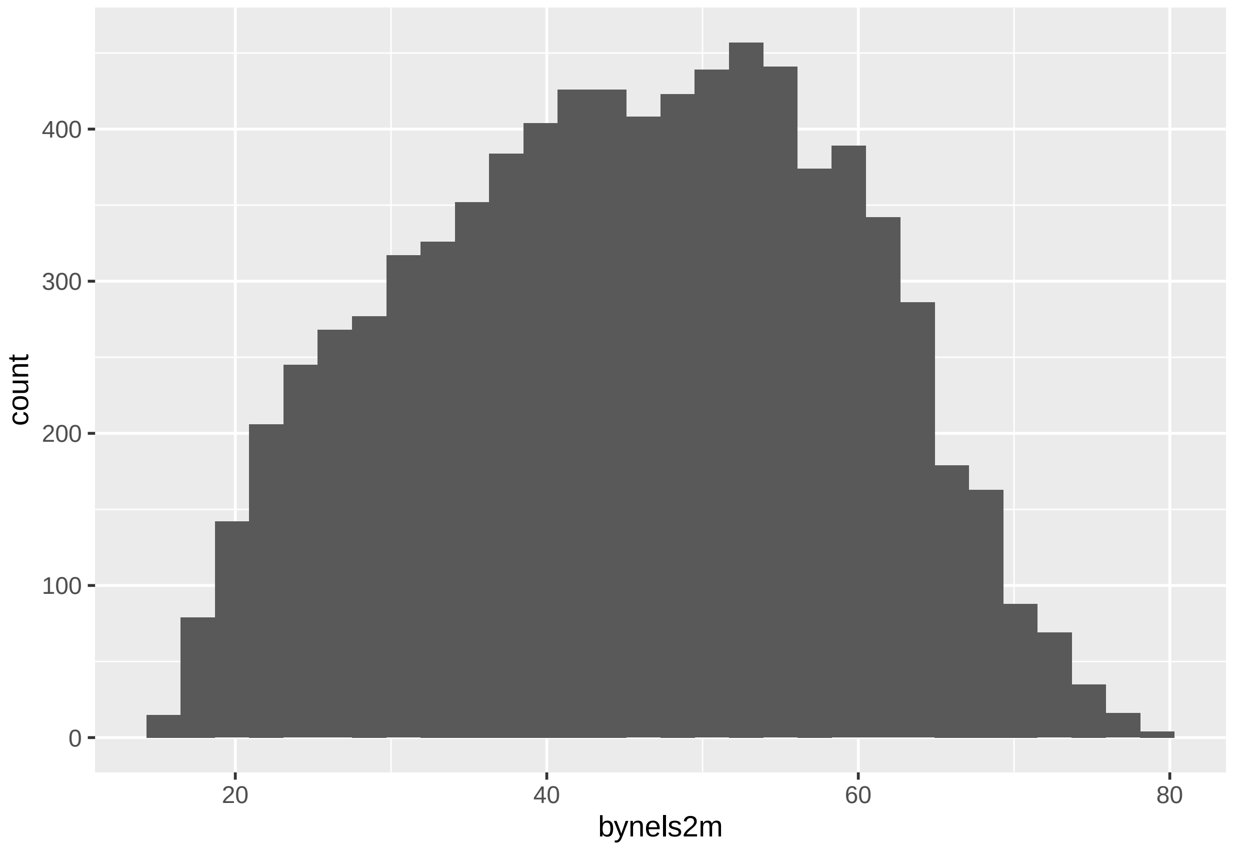
```r
els_train%>%summarize(mean(bynels2m,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   `mean(bynels2m, na.rm = TRUE)`
##                            <dbl>
## 1                           45.3
```

```r
gg<-ggplot(els_train,aes(x=bynels2m))
gg<-gg+geom_histogram()
gg
```

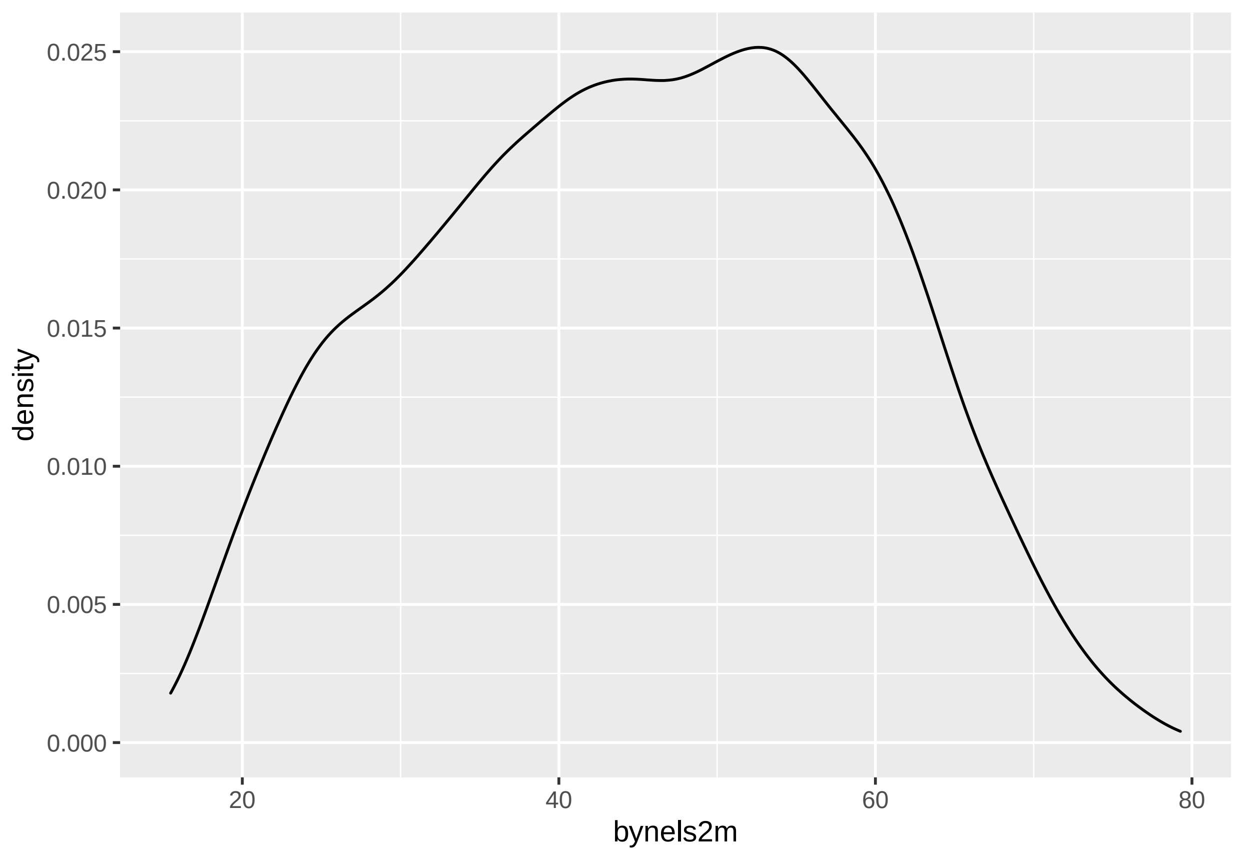
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 164 rows containing non-finite values (stat_bin).
```



```r
gg<-ggplot(els_train,aes(x=bynels2m))
gg<-gg+geom_density()
gg
```

```
## Warning: Removed 164 rows containing non-finite values (stat_density).
```



This variable has a nice symmetric distribution. It looks approximately normal, which will help in interpreting the results.

```
#Model 1: simple bivariate regression

mod1<-lm(bynels2m~byses1,data=els_train) #outcome on left, predictor on right

summary(mod1)
```

```
##
## Call:
## lm(formula = bynels2m ~ byses1, data = els_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.849  -8.930   0.375   9.052  39.206
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  44.9912     0.1409  319.20   <2e-16 ***
## byses1        8.1366     0.1893   42.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.31 on 7640 degrees of freedom
##   (502 observations deleted due to missingness)
## Multiple R-squared:  0.1946, Adjusted R-squared:  0.1945
## F-statistic:  1847 on 1 and 7640 DF,  p-value: < 2.2e-16
```

```
confint(mod1)
```

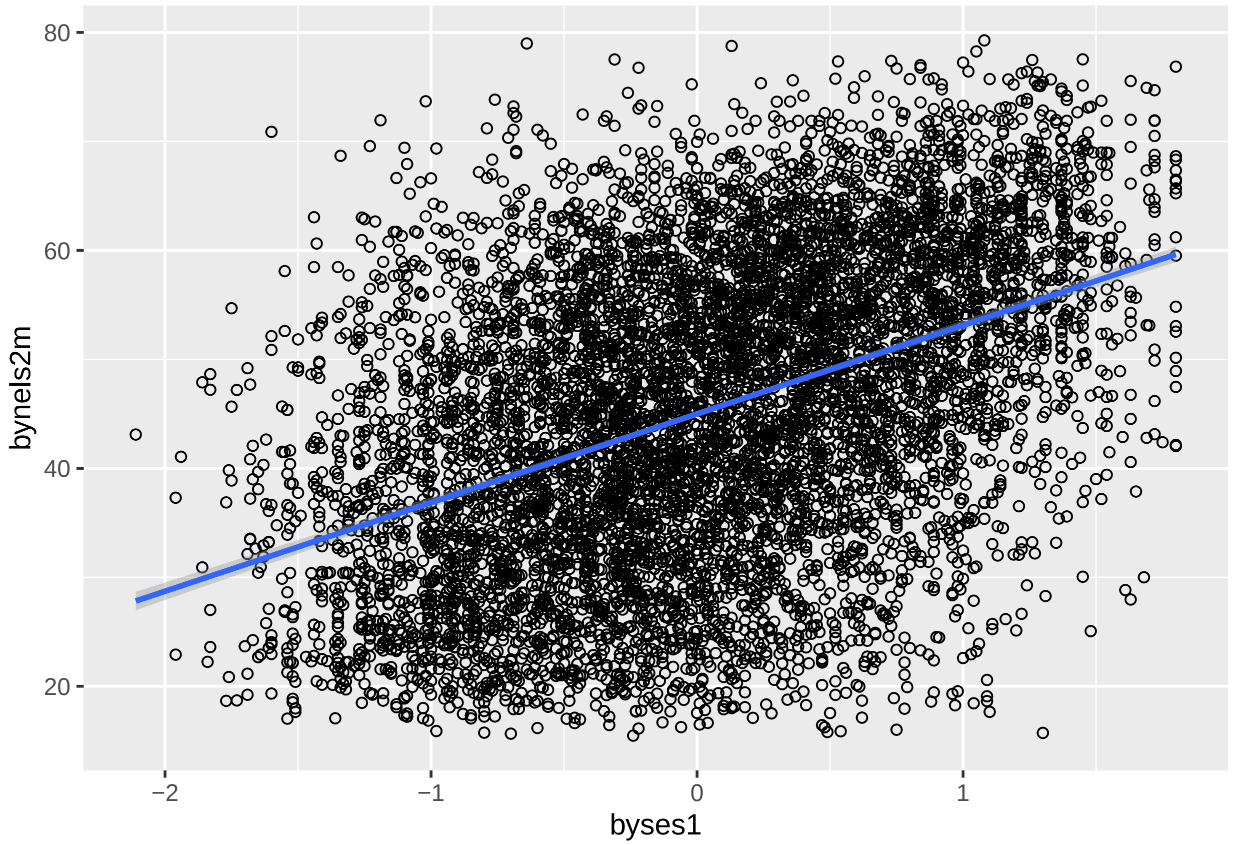
```
##                 2.5 %    97.5 %
## (Intercept) 44.714929 45.267526
## byses1       7.765411  8.507759
```

```
g1<-ggplot(els_train, aes(x=byses1,y=bynels2m))+ #specify data and x and y
          geom_point(shape=1)+ #specify points
          geom_smooth(method=lm) #ask for lm line
g1
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 502 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 502 rows containing missing values (geom_point).
```



```
els_train<-els_train%>%add_predictions(mod1)%>%dplyr::rename(pred1=pred) #predict using data in memory

## RMSE
rmse_1<-modelr::rmse(mod1,els_train);rmse_1
```

```
## [1] 12.30348
```

What this shows is that as socio-economic status increases, math scores are predicted to increase. For every one unit increase in SES, math scores are predicted to increase by $8. The rmse of 12 gives us a sense of how wrong the model tends to be when using just this one predictor.

Quick Exercise Run a regression using a different predictor. Calculate rmse and see if you can beat my score.

Multiple Regression.

Okay, so we can see that this is somewhat predictive, but we can do better. Let's add in a second variable: the parent's level of education.

```
#Part 2: Multiple regression

mod2<-lm(bynels2m~as.factor(bypared)+
           byses1,
          data=els_train)

summary(mod2)
```

```
##
## Call:
## lm(formula = bynels2m ~ as.factor(bypared) + byses1, data = els_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.782  -9.033   0.339   9.010  38.715
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          46.5216     0.6881  67.609  < 2e-16 ***
## as.factor(bypared)2  -0.5464     0.6733  -0.812 0.417093
## as.factor(bypared)3  -2.5776     0.7780  -3.313 0.000927 ***
## as.factor(bypared)4  -0.8523     0.8024  -1.062 0.288181
## as.factor(bypared)5  -1.6261     0.7996  -2.034 0.042018 *
## as.factor(bypared)6  -2.0766     0.8301  -2.502 0.012383 *
## as.factor(bypared)7  -1.5467     0.9682  -1.598 0.110172
## as.factor(bypared)8  -3.5970     1.1014  -3.266 0.001096 **
## byses1                8.9059     0.3350  26.584  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.29 on 7633 degrees of freedom
##   (502 observations deleted due to missingness)
## Multiple R-squared:  0.1975, Adjusted R-squared:  0.1966
## F-statistic: 234.8 on 8 and 7633 DF,  p-value: < 2.2e-16
```

```
els_train<-els_train%>%add_predictions(mod2)%>%dplyr::rename(pred2=pred)

rmse_2<-modelr::rmse(mod2,els_train); rmse_2
```

```
## [1] 12.28195
```

This finding reflects the basic inequity in our education system: lower income students score lower on math scores. This holds true even if we control for parental education.

Quick Exercise Add another variable to your model from above and see what difference it makes. How is your RMSE?

Transformations

The `byses` variable is a little hard to interpret. It's on a scale from -2 to 2, which you should remember as the scale for a standardized variable or Z score. Let's transform it to be on a percentile scale from 0-100.

```
els_train<-els_train%>%mutate(byses_p=percent_rank(byses1)*100)
els_train%>%dplyr::summarize(mean(byses_p,na.rm=TRUE))
```

```
## # A tibble: 1 x 1
##   `mean(byses_p, na.rm = TRUE)`
##                           <dbl>
## 1                          49.8
```

```
mod3<-lm(bynels2m~byses_p+
         as.factor(bypared),
         data=els_train
         );summary(mod3)
```

```
##
## Call:
## lm(formula = bynels2m ~ byses_p + as.factor(bypared), data = els_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.857  -9.055   0.281   9.180  39.208
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         34.547239   0.573118  60.279   <2e-16 ***
## byses_p              0.222994   0.008538  26.117   <2e-16 ***
## as.factor(bypared)2  0.798336   0.663053   1.204    0.229
## as.factor(bypared)3 -1.174077   0.760833  -1.543    0.123
## as.factor(bypared)4  0.471181   0.784935   0.600    0.548
## as.factor(bypared)5 -0.396732   0.782402  -0.507    0.612
## as.factor(bypared)6 -1.134680   0.814131  -1.394    0.163
## as.factor(bypared)7 -0.299730   0.944575  -0.317    0.751
## as.factor(bypared)8 -1.330383   1.052509  -1.264    0.206
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.31 on 7633 degrees of freedom
##   (502 observations deleted due to missingness)
## Multiple R-squared:  0.1951, Adjusted R-squared:  0.1942
## F-statistic: 231.3 on 8 and 7633 DF,  p-value: < 2.2e-16
```

This changes the coefficient AND its interpretation. Now, for every one percent increase in SES, math scores are predicted to increase by 0.22. Linear transformations will not change the statistical significance (t value), but non linear transformations like the one we just did will, as you can see. Does this change the RMSE?

```
rmse_3<-modelr::rmse(mod3,els_train)
```

It looks like it actually increases it a bit.

Testing and Training

The essence of prediction is discovering the extent to which our models can predict outcomes for data that *does not come from our sample.* Many times this process is temporal. We fit a model to data from one time period, then take predictors from a subsequent time period to come up with a prediction in the future. For instance, we might use data on team performance to predict the likely winners and losers for upcoming soccer games.

This process does not have to be temporal. We can also have data that is out of sample because it hadn't yet been collected when our first data was collected, or we can also have data that is out of sample because we designated it as out of sample.

The data that is used to generate our predictions is known as *training* data. The idea is that this is the data used to train our model, to let it know what the relationship is between our predictors and our outcome. So far, we have only worked with training data.

That data that is used to validate our predictions is known as *testing* data. With testing data, we take our trained model and see how good it is at predicting outcomes using out of sample data.

One very simple approach to this would be to cut our data in half. We could then train our model on half the data, then test it on the other half. This would tell us whether our measure of model fit (e.g. rmse, auc) is similar or different when we apply our model to out of sample data. That's what we've done today: we have only been working with half of our data– the training half.

The testing data (which is a random half of the original dataset) is stored as `els_test`. Since we transformed a variable in the training dataset, we'll need to do the same in the testing dataset.

```
load("els_test.Rdata")
els_test<-els_test%>%mutate(byses_p=percent_rank(byses1)*100)
```

Now we can use the model we trained (model 3) on the testing data.

```
## Generate a prediction from the testing dataset
rmse_test_1<-modelr::rmse(mod1,els_test);rmse_test_1
```

```
## [1] 12.24225
```

```
rmse_test_2<-modelr::rmse(mod2,els_test);rmse_test_2
```

```
## [1] 12.24041
```

```
rmse_test_3<-modelr::rmse(mod3,els_test);rmse_test_3
```

```
## [1] 12.26706
```

Notice that this is different than the value for our training dataset.

Thinking about regression for prediction

You MUST remember: correlation is not causation. All you can pick up on using this tool is associations, or common patterns. You can't know whether one thing causes another. Remember that the left hand side variable could just as easily be on the right hand side.

Scatterplots

Scatterplots are a great way to present data that has a continuous response variable. When creating scatterplots, the idea is to show ALL of the data, and then show how your model is summarizing the relationships in that data.

Setup

The code for today starts with the normal set of preliminaries, opening up the `els.RData` dataset and creating a codebook.

Bivariate Regression

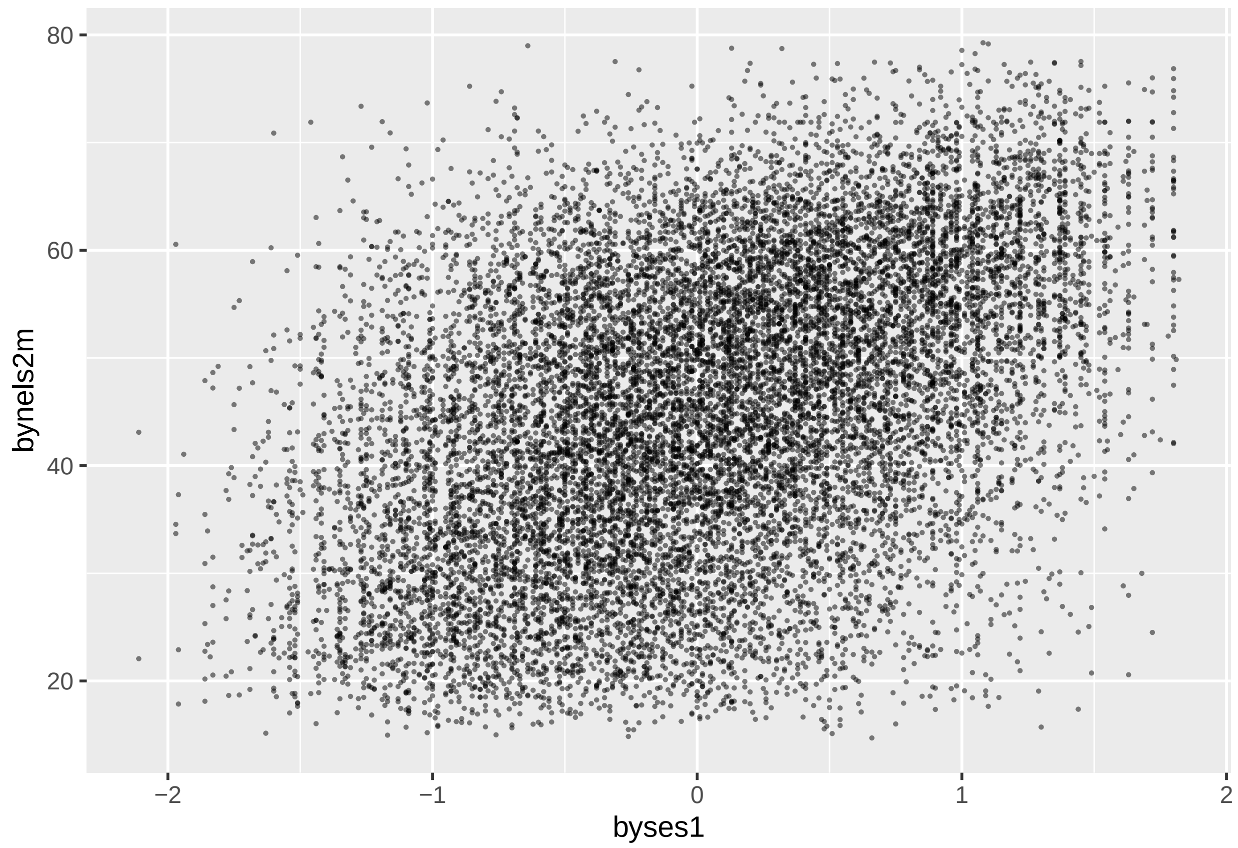
We begin with a simple model of test scores as a function of socio-economic status.

Basics of Creating a Scatterplot

Our first step should be to plot the data. Today, we'll be using the `ggplot2` library, which is a highly functional implementation of what's known as the grammar of graphics. In a very small nutshell, the grammar of graphics refers to laying out a graphic in a series of layers. For our first scatterplot, we first specify the data that we'll be drawing on, then the "aesthetic" of the graphic, which will be based on our x and y variables from our regression. We then specify the first layer, which is a series of points defined by the intersection of the x and y variables.

```
g1<-ggplot(data=els,
           aes(x=byses1,y=bynels2m)
           )

g1<-g1+geom_point(alpha=.5,size=.25) # Add points at x and y
g1
```

```
## Warning: Removed 964 rows containing missing values (geom_point).
```

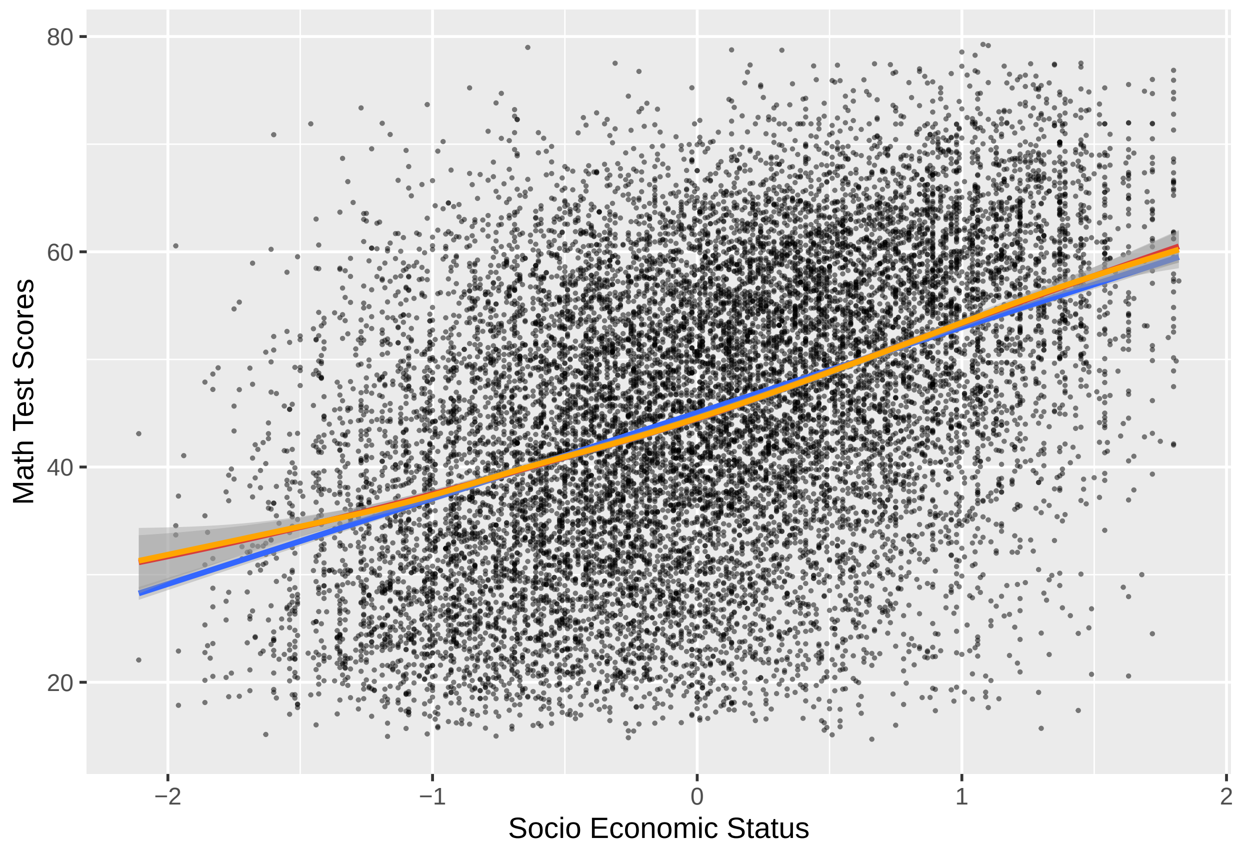


So, this is a bit of a mess. It just looks like a blob. We need to fix it to make it more readable, but let's first get the next element we want, which is the regression line.

```
g1<-g1+geom_smooth(method="lm")
g1<-g1+geom_smooth(method = "loess",color="red")
g1<-g1+geom_smooth(color="orange")
```

```r
g1<-g1+ylab("Math Test Scores")+xlab("Socio Economic Status")
g1
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 964 rows containing non-finite values (stat_smooth).

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 964 rows containing non-finite values (stat_smooth).

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 964 rows containing non-finite values (stat_smooth).

## Warning: Removed 964 rows containing missing values (geom_point).
```



Using Conditional Means to Create Scatterplots

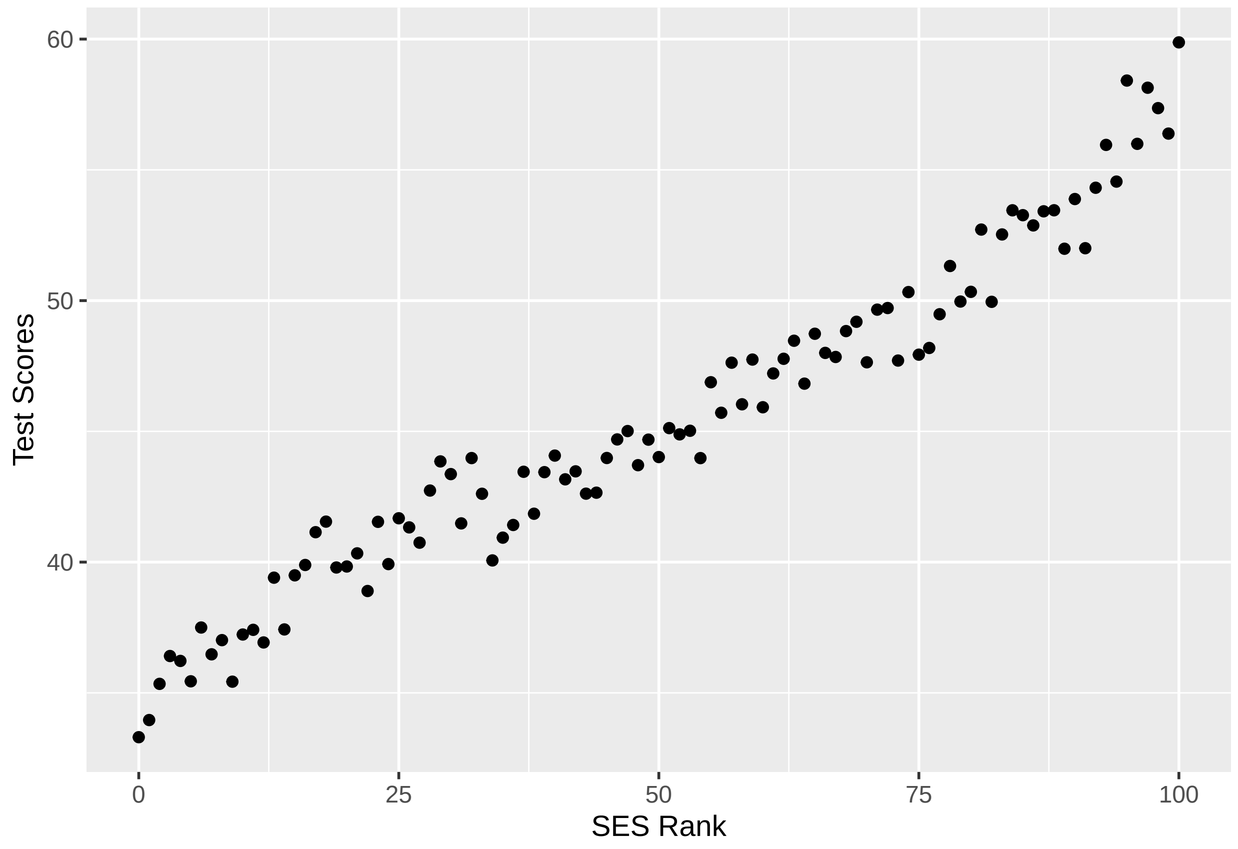
It's also really hard to see. We can use conditional means to help out with that problem. Let's get the average amount of test scores at every percentile level of `byses1`. Notice the use of `round` to get income percentiles that are at two digits only.

```r
els_sum<-els%>%
  mutate(ses_rank=percent_rank(byses1)*100)%>%
  mutate(ses_rank_r=round(ses_rank))%>%
  group_by(ses_rank_r)%>%
  dplyr::summarize(test_mean=mean(bynels2m,na.omit=TRUE))
```

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

```r
g1a<-ggplot(els_sum,aes(x=ses_rank_r,y=test_mean))

g1a<-g1a+geom_point()

g1a<-g1a+ylab("Test Scores")+xlab("SES Rank")

g1a
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

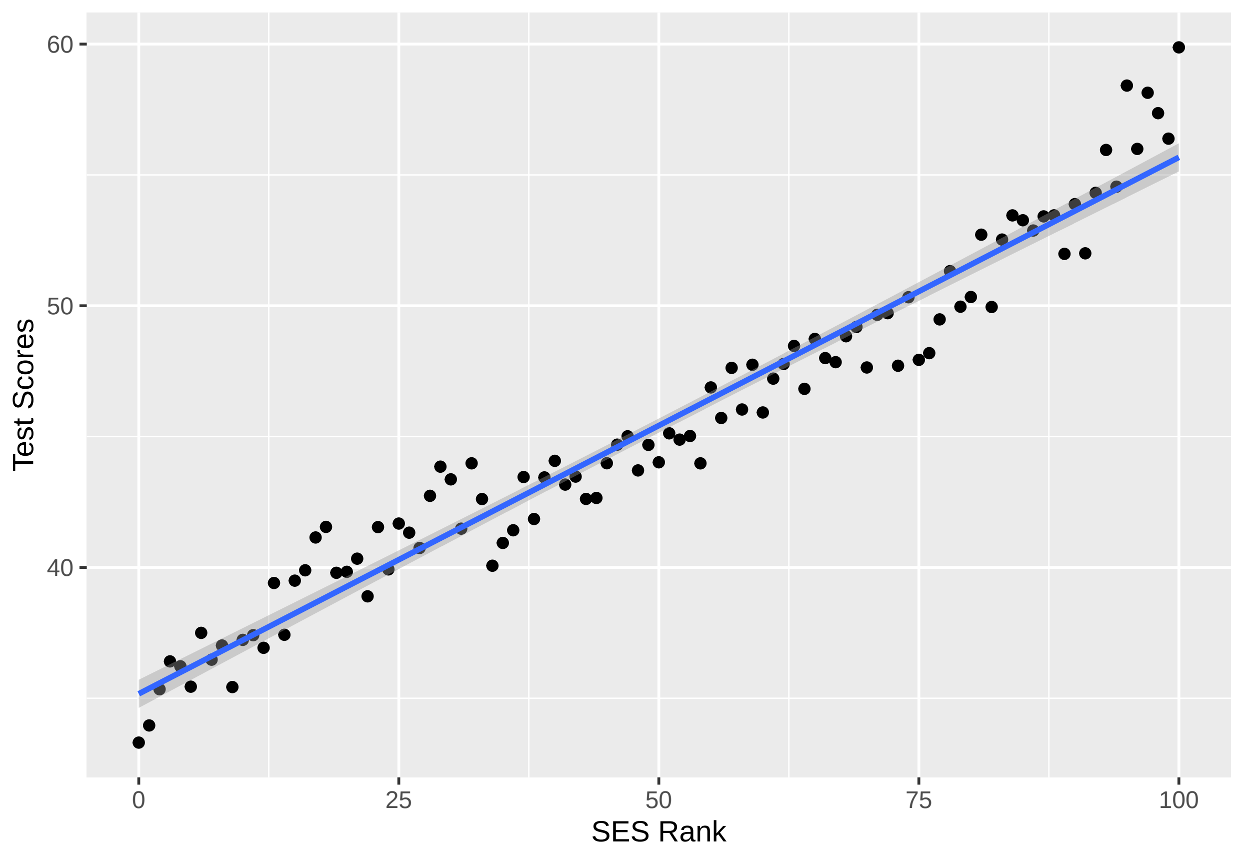


Better! Simplifying data can help.

We can add a regression line to this simpler data

```r
g1b<-g1a+geom_smooth(method="lm") # Add a line
g1b
```

```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```



This summarizes the basic relationship nicely. We're ready to run the model and get results.

```
#First model

mod_1<-lm(bynels2m~byses1,data=els);summary(mod_1)
```

```
##
## Call:
## lm(formula = bynels2m ~ byses1, data = els)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.658  -8.801   0.400   9.048  39.042
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  45.0379     0.0993   453.5   <2e-16 ***
## byses1        7.9535     0.1335    59.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.27 on 15323 degrees of freedom
##   (964 observations deleted due to missingness)
## Multiple R-squared:  0.1882, Adjusted R-squared:  0.1881
## F-statistic:  3552 on 1 and 15323 DF,  p-value: < 2.2e-16
```

Quick Exercise Create a similar graphic, but this time use reading scores as the independent variable.

Presenting Complex Results

The next step is to add covariates. I'll be working with the variable `bypared` which is a factor that summarizes the parental education of respondents. I'm going to set the color of the markers by the `bypared` factor.

```
g2<-ggplot(data=filter(els,is.na(bypared)==FALSE),
            aes(x=byses1,y=bynels2m,
                color=as.factor(bypared) #notice the color option
                ))
## Let's make the dots smaller for readability
g2<-g2+geom_point(size=.25)

## Changing the Legend
g2<-g2+theme(legend.position="bottom"  )#, legend.title =
              # element_blank())

g2<-g2+ylab("Test Scores")+xlab("Socio Economic Status")

g2 <- g2+ scale_color_discrete(name="parental Ed")
```

Our graphic is a bit complex, but shows the intersectionality between SES and parental education: there are very few students with low levels of parental education and/or high levels of SES or test scores.

We can see this same relationship in the model results:

Using Scatterplots to Explain Models

```
#Model 2: with parental education

mod_2<-lm(bynels2m~
            byses1+
            as.factor(bypared),
          data=els); summary(mod_2)
```

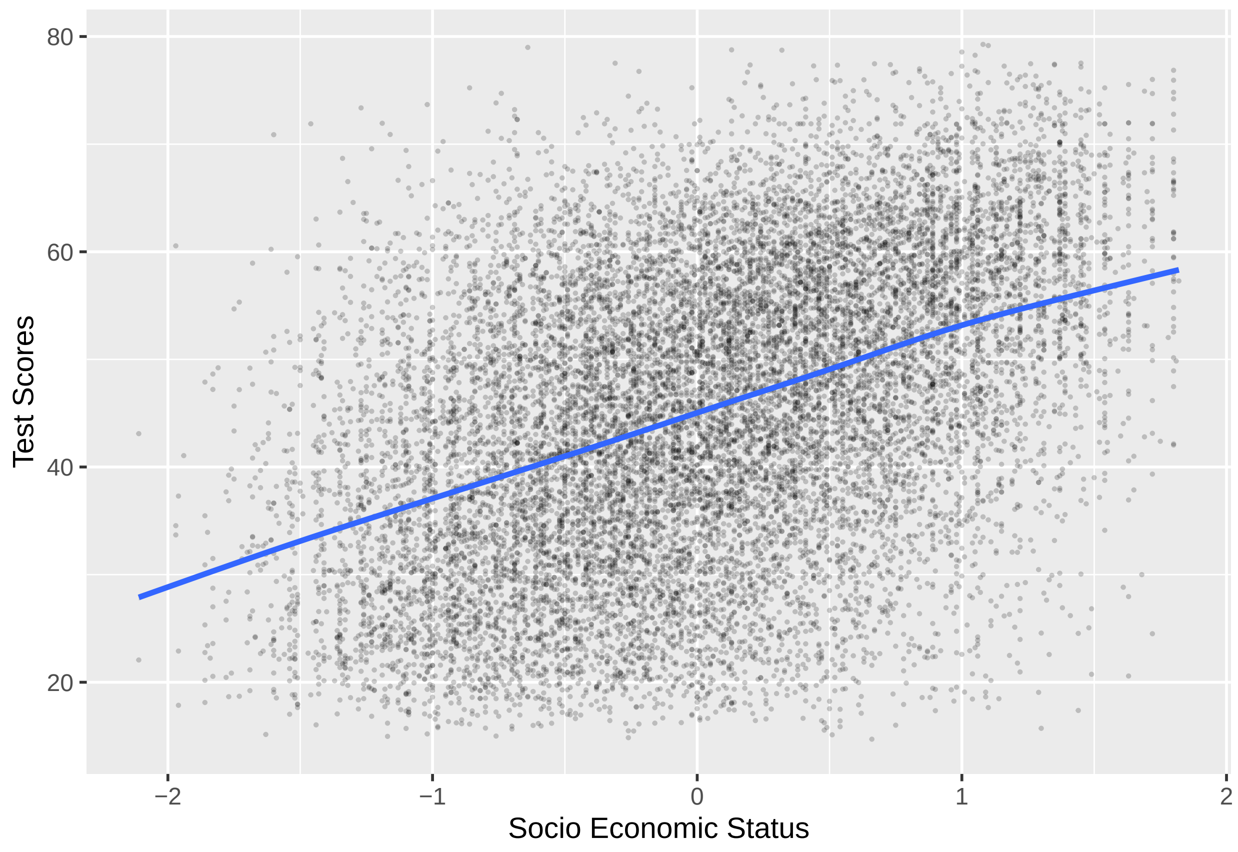
```
##
## Call:
## lm(formula = bynels2m ~ byses1 + as.factor(bypared), data = els)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.518  -8.854   0.348   9.112  38.683
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          46.3023     0.4851  95.447  < 2e-16 ***
## byses1                8.6985     0.2357  36.903  < 2e-16 ***
## as.factor(bypared)2  -0.3953     0.4765  -0.830 0.406712
## as.factor(bypared)3  -1.9105     0.5487  -3.482 0.000499 ***
## as.factor(bypared)4  -0.8594     0.5612  -1.531 0.125676
## as.factor(bypared)5  -1.2607     0.5610  -2.247 0.024653 *
## as.factor(bypared)6  -1.5761     0.5872  -2.684 0.007285 **
## as.factor(bypared)7  -1.5823     0.6822  -2.320 0.020380 *
## as.factor(bypared)8  -3.3724     0.7722  -4.367 1.27e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 12.26 on 15316 degrees of freedom
##   (964 observations deleted due to missingness)
## Multiple R-squared:   0.19,  Adjusted R-squared:  0.1896
## F-statistic: 449.2 on 8 and 15316 DF,  p-value: < 2.2e-16
```

Now let's take a look at this model plotted against the actual data. I'm going to use the `alpha` setting to make the dots more transparent. I'm also going to make the dots smaller via the size specification.

```r
els<-els%>%add_predictions(mod_2)%>%dplyr::rename(pred_mod_2=pred)

g3<-ggplot(els,aes(x=byses1,y=bynels2m))
g3<-g3+geom_point(alpha=.2,size=.25)

g3<-g3+geom_smooth(data=els,(aes(x=byses1,y=pred_mod_2)))

g3<-g3+xlab("Socio Economic Status")+ylab("Test Scores")

g3
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 964 rows containing non-finite values (stat_smooth).
## Warning: Removed 964 rows containing missing values (geom_point).
```



As we add more variables to the model, it can get more difficult to plot relationships. One very good option is to plot lines based on a hypothetical set of data. Below, I create a hypothetical set of data that include values of SES across the range of SES, and includes values for every level of `bypared`. I then run predictions from this hypothetical data to get a prediction line for every level of parental education.

Now, using my estimates from model 2, I predict what would happen to these hypothetical individuals. Once I've got my prediction, I transform it back out of the log scale into the "response" level of dollars.

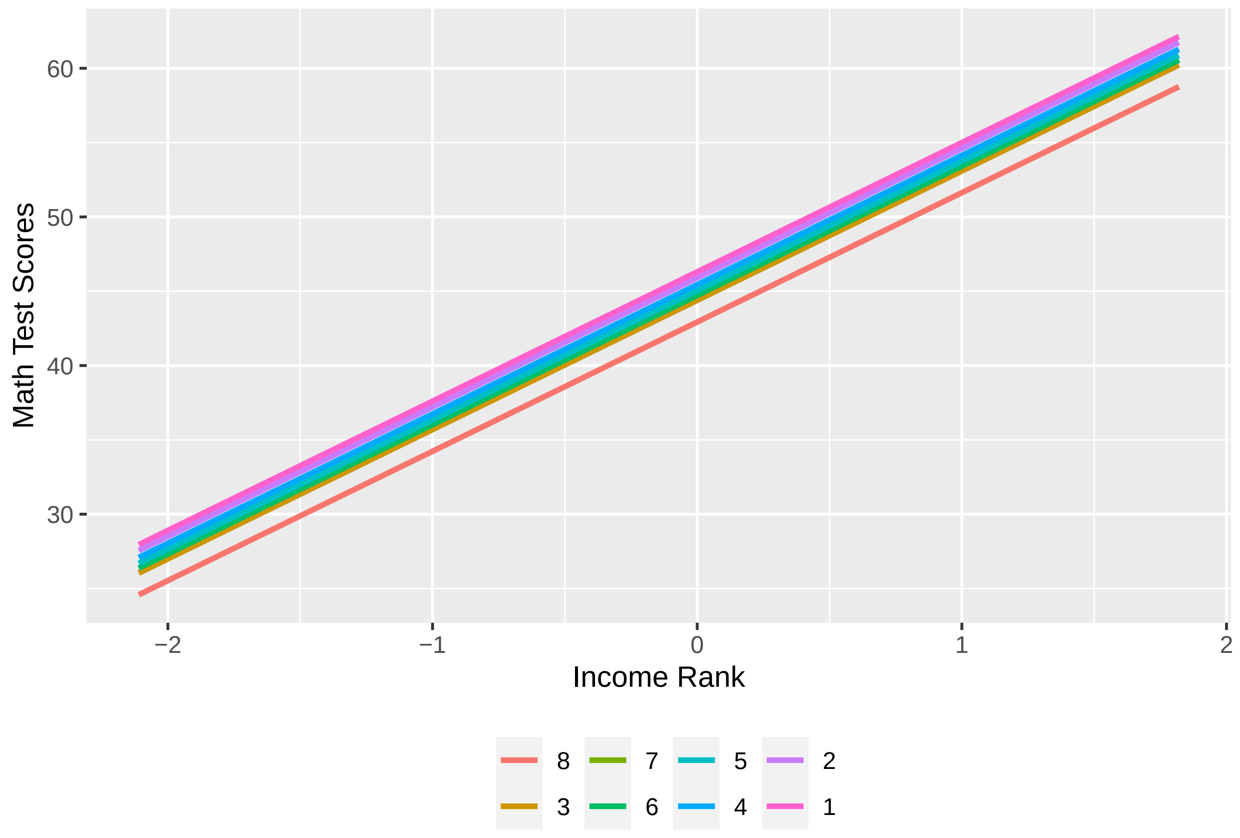
```
hypo_data<-data_grid(els, byses1 = seq_range(byses1,n=100),bypared)  %>% add_predictions(mod_2)
```

Now we can plot the result, using the `geom_smooth` layer to give us lines for every level of `childage`.

```
g4<-ggplot(data=hypo_data,
           aes(x=byses1,
               y=pred,
               color=fct_reorder(.f=as.factor(bypared),pred))) #notice color
g4<-g4+geom_smooth(method=lm,se=FALSE)
g4<-g4+theme(legend.position="bottom",legend.title = element_blank())
g4<-g4+xlab("Income Rank")+ylab("Math Test Scores")
g4
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 100 rows containing non-finite values (stat_smooth).
```

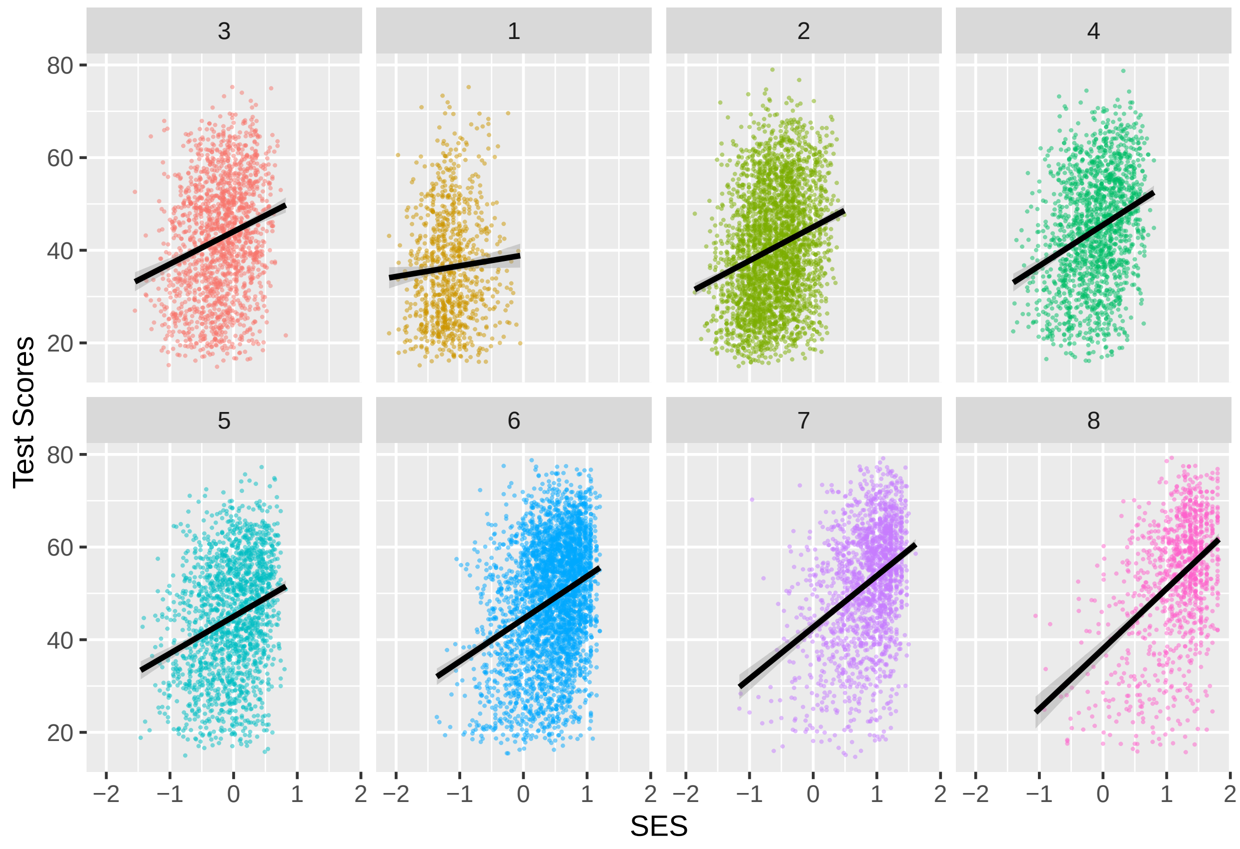


To show this in the data we can break it out for every type of parental education.

```
## Resort Parental Education for graphic
#els<-els%>%mutate(bypared = factor(bypared))
els<-els%>%mutate(bypared=reorder(x=bypared,bynels2m))
# filter nas!
g5<-ggplot(filter(els, is.na(bypared) == FALSE),aes(x=byses1,y=bynels2m, color=as.factor(bypared)))
g5<-g5+geom_point(alpha=.5,size=.1)
g5<-g5+geom_smooth(method="lm",color="black")
```

```
g5<-g5+facet_wrap(~as.factor(bypared),nrow=2)
g5<-g5+xlab("SES")+ylab("Test Scores")
g5<-g5+theme(legend.position="none") #Suppress legend, not needed

g5
```

```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 68 rows containing non-finite values (stat_smooth).
## Warning: Removed 68 rows containing missing values (geom_point).
```



Classification

Classification is the process of predicting group membership. Understanding which individuals are likely to be members of which groups is a key task for data scientists. For instance, most recommendation engines that are at the hear of consumer web sites are based on classification algorithms, predicting which consumers are likely to purchase which products.

Pizza

Today we'll be working with the pizza dataset, which comes from the subreddit random acts of pizza. Each line represents a post to this subreddit. We have various characteristics of these posts, along with the request text from the post itself. We'll use these characteristics of the posts to predict whether or not the poster received pizza. This lesson is inspired by this article

```r
library(knitr)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.3

##
## Attaching package: 'caret'

## The following objects are masked from 'package:ModelMetrics':
##
##     confusionMatrix, precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
load("za_train.RData")
```

Our goal is to create a classifier that will accurately classify people in a testing dataset as to whether they will receive a pizza or not, based on the content of their post. This is a VERY common task in data science– taking user supplied content and using it to accurately classify that user, typically as someone who will buy a product or service.

## Dependent Variable

Our dependent variable is a binary variable, `got_pizza` that denotes whether the user indicated that someone had sent them a pizza after posting in the subreddit "random acts of pizza". Let's take a look at this and see how many people posted that they got a pizza.

```r
table(za_train$got_pizza)
```

```
##
##    0    1
## 2136  700
```

This tells us the raw numbers. Lots of times we want to know the proportions. The function `prop.table` can do this for us.

```r
prop.table(table(za_train$got_pizza))
```

```
##
##         0         1
## 0.7531735 0.2468265
```

So, 0.2468265 of posts indicate that they were sent a pizza as a result of their post. We're interested in taking information in the posts themselves to see what makes it more or less likely that they would indicate that they received a pizza.

Conditional Means as a Classifier

We'll start by generating some cross tabs and some quick plots, showing the probability of receiving pizza according to several characteristics of the post. We start with a basic crosstab of the dependent variable. We use `prop.table` to change this from raw counts to proportions. I also provide a brief exampl of how to do a table using the `kable` function.

```r
#Cross Tabs
```

```r
za_train%>%
  dplyr::count(got_pizza)%>% # Count numbers getting pizza
  mutate(p=prop.table(n))%>% #mutate for proportions using prop.table
  kable(format="markdown") # output to table
```

got_pizza	n	p
0	2136	0.7531735
1	700	0.2468265

So, about 75% of the sample didn't get pizza, about 25% did.

Next, we cross-tabulate receiving pizza with certain terms. First, if the request mentioned the word "student."

```r
prop.table(table(za_train$student,za_train$got_pizza),margin=1)
```

```
##
##                      0         1
##   No student 0.7590683 0.2409317
##   Student    0.6820276 0.3179724
```

Next, if the request mentioned the word "grateful."

```r
g_table<-table(za_train$grateful,za_train$got_pizza);g_table
```

```
##
##                           0    1
##   Grateful not in post 2071  673
##   Grateful in post       65   27
```

```r
prop.table(g_table,margin=1)
```

```
##
##                                0         1
##   Grateful not in post 0.7547376 0.2452624
##   Grateful in post     0.7065217 0.2934783
```

Crosstabs using binary data are equivalent to generating conditional means, as shown below.

```r
#Predictions using conditional means

za_train%>%group_by(grateful)%>%dplyr::summarize(mean(got_pizza))
```

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

```
## # A tibble: 2 x 2
##   grateful             `mean(got_pizza)`
##   <fct>                            <dbl>
## 1 Grateful not in post             0.245
## 2 Grateful in post                 0.293
```

Note how the mean of got pizza is equivalent to the proportion answering "1" in the previous table.

But, we can also use conditional means to get proportions for very particular sets of characteristics. In this case, what about individuals who included some combination of the terms "grateful","student" and "poor" in their posts?

```r
za_sum<-za_train%>%
  group_by(grateful,student,poor)%>%
  dplyr::summarize(mean_pizza=mean(got_pizza))%>%
  arrange(-mean_pizza)
```

```
## `summarise()` regrouping output by 'grateful', 'student' (override with `.groups` argument)
```

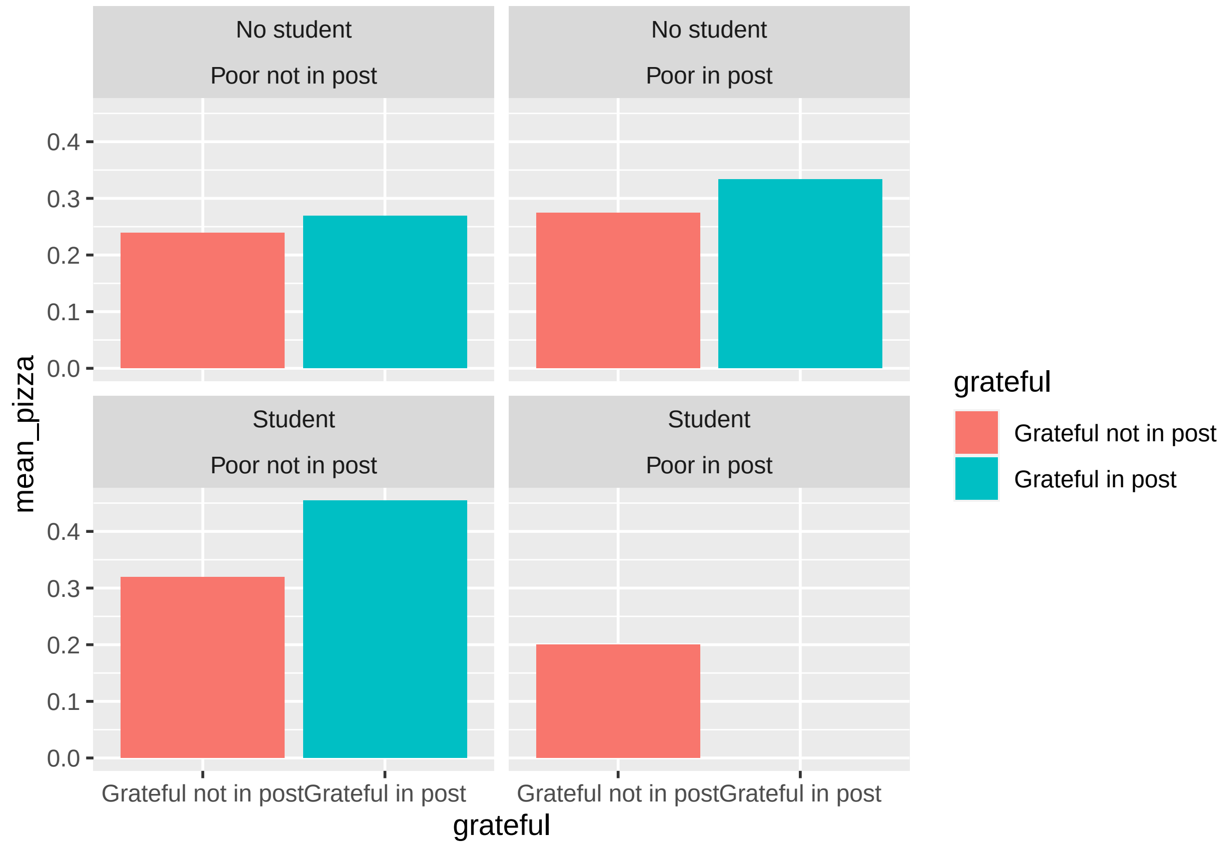
```
za_sum%>%kable()
```

grateful	student	poor	mean_pizza
Grateful in post	Student	Poor not in post	0.4545455
Grateful in post	No student	Poor in post	0.3333333
Grateful not in post	Student	Poor not in post	0.3193717
Grateful not in post	No student	Poor in post	0.2745098
Grateful in post	No student	Poor not in post	0.2692308
Grateful not in post	No student	Poor not in post	0.2392441
Grateful not in post	Student	Poor in post	0.2000000

The initial evidence here makes it look like the posts that included the terms "Grateful" and "student" had the highest probability of receiving a pizza (or at least posting that they received a pizza!).

Probability of Receiving Pizza, Using Various Terms in Post

```
gg<-ggplot(za_sum,aes(x=grateful,y=mean_pizza,fill=grateful))
gg<-gg+geom_bar(stat="identity")
gg<-gg+facet_wrap(~student+poor)
gg
```



Classification Using Linear Probability Model

We can use standard OLS regression for classification. It's not ideal, but most of the time it's actually not too bad, either. Below we model the binary outcome of receiving pizza as a function of karma, total posts, posts on the pizza subreddit, whether or not the poster mentioned the words "student" or "grateful."

```
# Linear model
lm_mod<-lm(got_pizza~
             karma+
             total_posts+
             raop_posts+
             student+
             grateful,
           data=za_train,y=TRUE,na.exclude=TRUE);summary(lm_mod)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'na.exclude' will be disregarded
```

```
##
## Call:
## lm(formula = got_pizza ~ karma + total_posts + raop_posts + student +
##     grateful, data = za_train, y = TRUE, na.exclude = TRUE)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6932 -0.2288 -0.2260 -0.2172  0.7950
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              2.260e-01  9.298e-03  24.310  < 2e-16 ***
## karma                    5.039e-06  2.403e-06   2.097  0.03609 *
## total_posts             -1.180e-04  2.058e-04  -0.573  0.56640
## raop_posts               1.558e-01  2.566e-02   6.072 1.43e-09 ***
## studentStudent           8.221e-02  3.022e-02   2.720  0.00657 **
## gratefulGrateful in post  4.752e-02  4.536e-02   1.048  0.29489
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4275 on 2830 degrees of freedom
## Multiple R-squared:  0.01913,    Adjusted R-squared:  0.01739
## F-statistic: 11.04 on 5 and 2830 DF,  p-value: 1.521e-10
```

We're going to do something a bit different with the predictions from this model. After creating predictions, we're going to classify everyone with a predicted probablity above .5 as being predicted to get a pizza, while everyone with a predicted probability below .5 is predicted to not get one. We'll compare our classifications with the actual data.

```
#Predictions
za_train<-za_train%>%
  add_predictions(lm_mod)%>% ## Add in predictions from the model
  dplyr::rename(pred_lm=pred)%>% ## rename to be predictions from ols (lm)
  mutate(pred_lm_out=ifelse(pred_lm>=.35,1,0))
```

Let's create a table that shows the predictions of our model against what actually happened

```
pred_table<-table(za_train$got_pizza,za_train$pred_lm_out)
```

```r
pred_table
```

```
##
##        0    1
##   0 2055   81
##   1  638   62
```

```r
prop.table(pred_table)
```

```
##
##               0          1
##   0 0.72461213 0.02856135
##   1 0.22496474 0.02186178
```

```r
rownames(pred_table)<-c("Predicted 0","Predicted 1")
colnames(pred_table)<-c("Actually 0","Actually 1")
```

```r
ModelMetrics::confusionMatrix(za_train$got_pizza,za_train$pred_lm_out)
```

```
##      [,1] [,2]
## [1,] 2055  638
## [2,]   81   62
```

```r
caret::confusionMatrix(as.factor(za_train$got_pizza),as.factor(za_train$pred_lm_out))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2055   81
##          1  638   62
##
##                Accuracy : 0.7465
##                  95% CI : (0.73, 0.7624)
##     No Information Rate : 0.9496
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0691
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.76309
##             Specificity : 0.43357
##          Pos Pred Value : 0.96208
##          Neg Pred Value : 0.08857
##              Prevalence : 0.94958
##          Detection Rate : 0.72461
##    Detection Prevalence : 0.75317
##       Balanced Accuracy : 0.59833
##
##        'Positive' Class : 0
##
```

The confusion matrix generated here is explained here.

We're usually interested in three things: the overall accuracy of a classification is the proportion of cases accurately classified. The sensitivity is the proportion of "ones" that are accurately classified as ones– it's the probability that a case classified as positive will indeed be positive. Specificity is the probability that a case classified as 0 will indeed by classified as 0.

Question: how do you get perfect specificity? How do you get perfect sensitivity?

There are several well-known problems with linear regression as a classification algortihm. Two should give us pause: it can generate probabilites outside of 0,1 and it implies a linear change in probabilities as a function of the predictors which may not be justified given the underlying relationship between the predictors and the probability that the outcome is 1. Logistic regresssion should give a better predicted probability, one that's more sensitive to the actual relationship between the predictors and the outcome.

Logistic regression as a classifier

Logistic regression is set up to handle binary outcomes as the dependent variable. In particular, the predictions will always be a probability, which makes it better than the ironically named linear probability model. The downside to logistic regression is that it is modeling the log odds of the outcome, which means all of the coefficients are expressed as log odds, which no one understands intuitively. In this class, we're going to concentrate on logistic regression's ability to produce probabilities as predictions. Below I run the same model using logistic regression. Note the use of `glm` and the `family` option, which specifies a functional form and a particular link function.

```
#Logisitic model

logit_mod<-glm(got_pizza~
             karma+
             total_posts+
             raop_posts+
             student+
             grateful,
             data=za_train,
            na.action=na.exclude,
            family=binomial(link="logit"),
               y=TRUE)

summary(logit_mod)
```

```
##
## Call:
## glm(formula = got_pizza ~ karma + total_posts + raop_posts +
##     student + grateful, family = binomial(link = "logit"), data = za_train,
##     na.action = na.exclude, y = TRUE)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0224  -0.7192  -0.7146  -0.6478   1.9943
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.235e+00  5.211e-02 -23.699  < 2e-16 ***
## karma                    5.358e-05  2.246e-05   2.385  0.01707 *
## total_posts             -1.939e-03  1.439e-03  -1.347  0.17786
## raop_posts               7.467e-01  1.362e-01   5.482 4.21e-08 ***
## studentStudent           4.171e-01  1.536e-01   2.715  0.00662 **
## gratefulGrateful in post  2.384e-01  2.359e-01   1.010  0.31226
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3169.6  on 2835  degrees of freedom
## Residual deviance: 3118.7  on 2830  degrees of freedom
## AIC: 3130.7
##
## Number of Fisher Scoring iterations: 4
```

With these results in hand we can generate predicted probabilities and see if this model did any better. To get predicted probabilities, we need to specify `type=response` in our prediction call.

```
za_train<-za_train%>%
  mutate(pred_logit=predict(logit_mod,type="response"))
```

We can convert the predictions to a binary variable by setting a "threshold" of .5. Any prediction above .5 is considered to be a 1, anything below, a 0.

```
za_train<-za_train%>%
    mutate(pred_logit_out=ifelse(pred_logit>=.45,1,0))

za_train<-za_train%>%
    mutate(pred_logit_out=as.factor(pred_logit_out))

za_train<-za_train%>%
    mutate(got_pizza=as.factor(got_pizza))
```

Now we create a confusion matrix to see how we did.

```
confusionMatrix(data=as.factor(za_train$pred_logit_out),reference=as.factor(za_train$got_pizza))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2119  676
##          1   17   24
##
##                Accuracy : 0.7556
##                  95% CI : (0.7394, 0.7714)
##     No Information Rate : 0.7532
##     P-Value [Acc > NIR] : 0.3899
##
##                   Kappa : 0.0385
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99204
##             Specificity : 0.03429
##          Pos Pred Value : 0.75814
##          Neg Pred Value : 0.58537
##              Prevalence : 0.75317
##          Detection Rate : 0.74718
##    Detection Prevalence : 0.98554
##       Balanced Accuracy : 0.51316
##
##        'Positive' Class : 0
```

```
##
```

Applying predictions to the testing dataset.

With our new (not very good) classifier, we can now add predictions to the testing dataset, and see how good this classifier is at predicting out of sample information.

```
load("za_test.RData")

za_test<-za_test%>%
  mutate(pred_logit=predict(logit_mod,newdata=.,type="response"))%>%
      mutate(pred_logit_out=ifelse(pred_logit>=.45,1,0))

za_test<-za_test%>%
    mutate(pred_logit_out=as.factor(pred_logit_out))

za_test<-za_test%>%
    mutate(got_pizza=as.factor(got_pizza))


confusionMatrix(data=za_test$pred_logit_out,reference=za_test$got_pizza)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2095  674
##          1   11   23
##
##                Accuracy : 0.7556
##                  95% CI : (0.7393, 0.7714)
##     No Information Rate : 0.7513
##     P-Value [Acc > NIR] : 0.3086
##
##                   Kappa : 0.0407
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9948
##             Specificity : 0.0330
##          Pos Pred Value : 0.7566
##          Neg Pred Value : 0.6765
##              Prevalence : 0.7513
##          Detection Rate : 0.7474
##    Detection Prevalence : 0.9879
##       Balanced Accuracy : 0.5139
##
##        'Positive' Class : 0
##
```

Thinking about classifiers

First, make sure that your dependent variable really is binary. If you're working with a continuous variable (say, income) don't turn it into a binary variable (e.g. low income).

Second, remember that classifiers must always balance sensitivity and specificity. Don't be overly impressed by a high overall percent correctly predicted, nor a high level of either specificity or sensitivity. Instead, look for classifiers that have both.