Specifically, I talked a couple of times about binomial regression , which is used to predict (read: recreate with a set of variables significantly related to) a binary outcome. The data example I used involved my dissertation data and the binary outcome was verdict: guilty or not guilty. A regression model returns the linear correction applied to the predictor variables to reproduce the outcome, and will highlight whether a predictor was significantly related to the outcome or not. But a big question you may be asking of your binomial model is: how well does it predict the outcome? Specifically, how can you examine whether your regression model is correctly classifying cases?

Code Chunks – GLM Function

GLM stands for general linear model, which is the basis for many statistical analyses, including regression and structural equation modeling. It provides a mathematical method of relating predictor variables to outcomes in terms of an equation, converting values on the predictor variable(s) to values on the outcome variable.  
  
Using the glm function is very similar to using the lm function for a linear model - you need to symbolically represent your equation:  
  
Y ~ x1 + x2 + ... + lastx  
  
And you need some kind of data object for R to apply the equation to. A key difference in the glm function is that you reference the kind of regression model to use, through the "family" argument. Here are the different family options:

* Binomial
* Gaussian
* Gamma
* Inverse.Gaussian
* Poisson
* Quasi
* Quasibinomial
* Quasipoisson

The family refers to the distribution that best represents your outcome data. For instance, Gaussian refers to the normal distribution - unless you specify some additional arguments in your glm function, these results will be essentially the same as using the lm function. Binomial is used for two-level outcomes, which is what I'll demonstrate below. Poisson is used for count data. Quasibinomial is used when you have additional variance in your outcome not explained by the binomial distribution alone (that is, there is more variance than if it followed the binomial distribution perfectly); it provides similar results as binomial, except with an additional error term. Quasipoisson is similar for count data. We won't worry about those for now though - just remember that anytime you have to estimate something extra, you need to provide more data, so each additional term increases how much data you must collect. We'll come back to this topic when we get to power analysis.  
  
So let's have some fun using the GLM function. First up, let's run a binomial regression. But we'll need some binary data first.

I've been sitting on my dissertation data for several years now and have gotten some fun presentations out it. To briefly summarize, I conducted my dissertation, which if you're so inclined you can : participants were exposed to a certain kind of pretrial publicity at random or to a control condition with no biasing information, then they completed a measure of justice attitudes, read a trial transcript, rendered a verdict, and provided a guilt rating. I included both verdict and guilt rating because of my meta-analysis, which included some studies that used guilt ratings instead of or in addition to verdicts. I was curious how these two measures related to each other, and what impact attitudes might have. So in addition to my planned analyses on pretrial publicity, I conducted some exploratory analyses using the attitude items.  
  
The measure I used provided many attitude items, but based on a literature search, I picked out a handful of specific attitudes that appeared to impact how people make decisions about guilt versus innocence. The theoretical background for my dissertation had to do with the tenuous relationship between attitudes and behaviors. Attitudes are poor predictors of behaviors, though more specific attitudes are better predictors than more general attitudes. My hypothesis was that pretrial publicity was not in itself biasing, unless you hold an attitude that a specific piece of information implies a defendant is guilty. And I conducted my exploratory analysis with much the same framework, that specific attitudes relating to the case and factors of the trial would be better predictors of the outcome.  
  
First, I ran a regression with a handful of attitudes: belief that courts should be able to use illegal evidence (e.g., obtained without a search warrant), beliefs about circumstantial evidence, belief that the defendant should be required to testify (even though defendants don't have to do anything, even mount a defense), belief that police abuse power, an attitude that police should be allowed to arrest anyone who seems "suspicious", belief that police are "overzealous", and finally an attitude that upstanding citizens have nothing to fear from police. I conducted this analysis once with verdict as the outcome, and again with guilt rating as the outcome.  
  
Since we want to get started with our glm function, let's run a binomial regression with verdict as the outcome.

dissertation<-**read.delim**("dissertation\_data.txt",

header=TRUE)

verdict\_binomial<-**glm**(verdict ~ illev + circumst + deftest + policepower +

suspicious + overzealous + upstanding, family=binomial,

data=dissertation)

**summary**(verdict\_binomial)

##

## Call:

## glm(formula = verdict ~ illev + circumst + deftest + policepower +

## suspicious + overzealous + upstanding, family = binomial,

## data = dissertation)

##

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -2.0267 -1.0100 -0.5999 1.0981 2.0656

##

## Coefficients:

## Estimate Std. Error z value Pr(>|z|)

## (Intercept) -0.57140 0.81612 -0.700 0.483834

## illev 0.13836 0.10659 1.298 0.194280

## circumst -0.43640 0.12770 -3.417 0.000632 \*\*\*

## deftest 0.09870 0.10859 0.909 0.363398

## policepower -0.13188 0.11256 -1.172 0.241354

## suspicious 0.37546 0.12157 3.088 0.002013 \*\*

## overzealous -0.01246 0.09991 -0.125 0.900717

## upstanding 0.22768 0.10820 2.104 0.035362 \*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## (Dispersion parameter for binomial family taken to be 1)

##

## Null deviance: 490.08 on 354 degrees of freedom

## Residual deviance: 445.22 on 347 degrees of freedom

## (1 observation deleted due to missingness)

## AIC: 461.22

##

## Number of Fisher Scoring iterations: 4

You would interpret the output in much the same way you would linear regression - significant variables indicate those significantly related to the outcome. I could use this resulting equation to predict a person's verdict based on their attitudes on these items. As I found in my dissertation analysis, beliefs about circumstantial evidence, police ability to arrest "suspicious people", and that upstanding (law-abiding) citizens have no reason to fear the police significantly affected whether the participant convicted in this particular case. The next step in my analysis would be to test how well the equation does at predicting verdict: how often it correctly and incorrectly classified people. (Spoiler alert: I did this as part of my dissertation and found that, though 3 indicators were significant, this equation wasn't great at predicting verdict, doing so correctly only about 55 percent of the time.)  
  
I also conducted a linear regression using guilt rating. Just for fun, let's conduct an lm as well as a glm of the Gaussian family to see how results match up:

guilt\_lm<-**lm**(guilt ~ illev + circumst + deftest + policepower +

suspicious + overzealous + upstanding,

data=dissertation)

**summary**(guilt\_lm)

##

## Call:

## lm(formula = guilt ~ illev + circumst + deftest + policepower +

## suspicious + overzealous + upstanding, data = dissertation)

##

## Residuals:

## Min 1Q Median 3Q Max

## -3.0357 -0.7452 0.1828 0.9706 2.5013

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 4.16081 0.38966 10.678 < 2e-16 \*\*\*

## illev 0.11111 0.05816 1.911 0.05689 .

## circumst -0.08779 0.06708 -1.309 0.19147

## deftest -0.02020 0.05834 -0.346 0.72942

## policepower 0.02828 0.06058 0.467 0.64090

## suspicious 0.17286 0.06072 2.847 0.00468 \*\*

## overzealous -0.03298 0.04792 -0.688 0.49176

## upstanding 0.08941 0.05374 1.664 0.09706 .

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 1.169 on 347 degrees of freedom

## (1 observation deleted due to missingness)

## Multiple R-squared: 0.07647, Adjusted R-squared: 0.05784

## F-statistic: 4.105 on 7 and 347 DF, p-value: 0.0002387

guilt\_gaus<-**glm**(guilt ~ illev + circumst + deftest + policepower +

suspicious + overzealous + upstanding,

family="gaussian", data=dissertation)

**summary**(guilt\_gaus)

##

## Call:

## glm(formula = guilt ~ illev + circumst + deftest + policepower +

## suspicious + overzealous + upstanding, family = "gaussian",

## data = dissertation)

##

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -3.0357 -0.7452 0.1828 0.9706 2.5013

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 4.16081 0.38966 10.678 < 2e-16 \*\*\*

## illev 0.11111 0.05816 1.911 0.05689 .

## circumst -0.08779 0.06708 -1.309 0.19147

## deftest -0.02020 0.05834 -0.346 0.72942

## policepower 0.02828 0.06058 0.467 0.64090

## suspicious 0.17286 0.06072 2.847 0.00468 \*\*

## overzealous -0.03298 0.04792 -0.688 0.49176

## upstanding 0.08941 0.05374 1.664 0.09706 .

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## (Dispersion parameter for gaussian family taken to be 1.366385)

##

## Null deviance: 513.40 on 354 degrees of freedom

## Residual deviance: 474.14 on 347 degrees of freedom

## (1 observation deleted due to missingness)

## AIC: 1128.2

##

## Number of Fisher Scoring iterations: 2

As you can see, the regression results are the same, though the output is slightly different between the two. The lm function gives you your R-squared and F-test for the regression (test that any indicators are significant), while the glm function gives you dispersion parameters and AIC. For this reason, if you're conducting a linear regression, use the lm function. The glm function only makes sense here if you need to specify additional arguments to use a different member of the Gaussian family. The results may be the same, but the applications are slightly different.  
  
I also did a second set of binomial regressions for my dissertation, this one dealing with variable that might affect how a person translates their guilt rating (which was on a scale of 1 to 7, the likelihood that the defendant actually committed the crime) to a verdict. That is, I found that while most people didn't select a verdict of guilty unless they also selected a guilt rating of 6 or 7, some convicted at a much lower rating, while others didn't select guilty even when they gave a guilt rating of 7, which translated to "definitely committed the crime". So I conducted two binomial regressions: one with criminal justice attitudes (e.g., how they think "reasonable doubt" should be defined, belief in the fairness of juries, and so on) as well as guilt rating, and one that also looked at how each attitude item interacts with guilt rating. This lets us see if the impact of guilt rating on verdict *depends* on a person's attitude. To make things easier to read, I'm going to set up my two equations separately from the glm function.

predictors\_only<-'verdict ~ obguilt + reasdoubt + bettertolet + libertyvorder +

jurevidence + guilt'

pred\_int<-'verdict ~ obguilt + reasdoubt + bettertolet + libertyvorder +

jurevidence + guilt + obguilt\*guilt + reasdoubt\*guilt +

bettertolet\*guilt + libertyvorder\*guilt + jurevidence\*guilt'

model1<-**glm**(predictors\_only, family="binomial", data=dissertation)

**summary**(model1)

##

## Call:

## glm(formula = predictors\_only, family = "binomial", data = dissertation)

##

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -2.4814 -0.6050 -0.1296 0.6419 2.7575

##

## Coefficients:

## Estimate Std. Error z value Pr(>|z|)

## (Intercept) -10.21158 1.33072 -7.674 1.67e-14 \*\*\*

## obguilt 0.39024 0.16010 2.437 0.0148 \*

## reasdoubt -0.12315 0.11591 -1.062 0.2880

## bettertolet -0.10054 0.11406 -0.881 0.3781

## libertyvorder -0.01425 0.14961 -0.095 0.9241

## jurevidence 0.08448 0.09897 0.854 0.3933

## guilt 1.81769 0.20222 8.989 < 2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## (Dispersion parameter for binomial family taken to be 1)

##

## Null deviance: 490.08 on 354 degrees of freedom

## Residual deviance: 309.66 on 348 degrees of freedom

## (1 observation deleted due to missingness)

## AIC: 323.66

##

## Number of Fisher Scoring iterations: 5

model2<-**glm**(pred\_int, family="binomial", data=dissertation)

**summary**(model2)

##

## Call:

## glm(formula = pred\_int, family = "binomial", data = dissertation)

##

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -2.6101 -0.5432 -0.1289 0.6422 2.2805

##

## Coefficients:

## Estimate Std. Error z value Pr(>|z|)

## (Intercept) -12.84571 6.10651 -2.104 0.0354 \*

## obguilt -0.34506 1.13742 -0.303 0.7616

## reasdoubt 1.56658 0.83360 1.879 0.0602 .

## bettertolet -0.21819 0.86374 -0.253 0.8006

## libertyvorder 0.88325 0.95459 0.925 0.3548

## jurevidence -0.62756 0.94042 -0.667 0.5046

## guilt 2.43022 1.19628 2.031 0.0422 \*

## obguilt:guilt 0.13062 0.21752 0.600 0.5482

## reasdoubt:guilt -0.33930 0.16323 -2.079 0.0376 \*

## bettertolet:guilt 0.01426 0.16662 0.086 0.9318

## libertyvorder:guilt -0.17482 0.18666 -0.937 0.3490

## jurevidence:guilt 0.14006 0.18359 0.763 0.4456

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## (Dispersion parameter for binomial family taken to be 1)

##

## Null deviance: 490.08 on 354 degrees of freedom

## Residual deviance: 300.66 on 343 degrees of freedom

## (1 observation deleted due to missingness)

## AIC: 324.66

##

## Number of Fisher Scoring iterations: 6

This matches what I found initially: that there was a significant interaction between guilt rating and attitudes about reasonable doubt. So depending on how an individual defined reasonable doubt, they would convict at lower or higher guilt ratings.

Code Chunks – Z Score and Standardizing

Let's revisit that GLM analysis. I was predicting verdict (guilty, not guilty) with binomial regression. I did one analysis where I used a handful of attitude items and the participant's guilt rating, and a second analysis where I created interactions between each attitude item and the guilt rating. The purpose was to see if an attitude impacts the threshold - how high the guilt rating needed to be before a participant selected "guilty". When working with interactions, the individual variables are highly correlated with the interaction variables based on them, so we solve that problem, and make our analysis and output a bit cleaner, by centering our variables and using those centered values to create interactions.

dissertation<-**read.delim**("dissertation\_data.txt",header=TRUE)

predictors<-**c**("obguilt","reasdoubt","bettertolet","libertyvorder",

"jurevidence","guilt")

**library**(psych)

**## Warning: package 'psych' was built under R version 3.4.4**

**describe**(dissertation[predictors])

## vars n mean sd median trimmed mad min max range skew

## obguilt 1 356 3.50 0.89 4 3.52 0.00 1 5 4 -0.50

## reasdoubt 2 356 2.59 1.51 2 2.68 1.48 -9 5 14 -3.63

## bettertolet 3 356 2.36 1.50 2 2.38 1.48 -9 5 14 -3.41

## libertyvorder 4 355 2.74 1.31 3 2.77 1.48 -9 5 14 -3.89

## jurevidence 5 356 2.54 1.63 2 2.66 1.48 -9 5 14 -3.76

## guilt 6 356 4.80 1.21 5 4.90 1.48 2 7 5 -0.59

## kurtosis se

## obguilt -0.55 0.05

## reasdoubt 26.92 0.08

## bettertolet 25.47 0.08

## libertyvorder 34.58 0.07

## jurevidence 25.39 0.09

## guilt -0.54 0.06

dissertation<-**subset**(dissertation, !**is.na**(libertyvorder))

R has a built-in function that will do this for you: scale. The scale function has 3 main arguments - the variable or variables to be scaled, and whether you want those variables centered (subtract mean) and/or scaled (divided by standard deviation). For regression with interactions, we want to both center and scale. For instance, to center and scale the guilt rating:

dissertation$guilt\_c<-**scale**(dissertation$guilt, center=TRUE, scale=TRUE)

I have a set of predictors I want to do this to, so I want to apply a function across those specific columns:

dissertation[46:51]<-**lapply**(dissertation[predictors], **function**(x) {

y<-**scale**(x, center=TRUE, scale=TRUE)

}

)

Now, let's rerun that binomial regression, this time using the centered variables in the model.

pred\_int<-'verdict ~ obguilt.1 + reasdoubt.1 + bettertolet.1 + libertyvorder.1 +

jurevidence.1 + guilt.1 + obguilt.1\*guilt.1 + reasdoubt.1\*guilt.1 +

bettertolet.1\*guilt.1 + libertyvorder.1\*guilt.1 + jurevidence.1\*guilt.1'

model2<-**glm**(pred\_int, family="binomial", data=dissertation)

**summary**(model2)

##

## Call:

## glm(formula = pred\_int, family = "binomial", data = dissertation)

##

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -2.6101 -0.5432 -0.1289 0.6422 2.2805

##

## Coefficients:

## Estimate Std. Error z value Pr(>|z|)

## (Intercept) -0.47994 0.16264 -2.951 0.00317 \*\*

## obguilt.1 0.25161 0.16158 1.557 0.11942

## reasdoubt.1 -0.09230 0.20037 -0.461 0.64507

## bettertolet.1 -0.22484 0.20340 -1.105 0.26899

## libertyvorder.1 0.05825 0.21517 0.271 0.78660

## jurevidence.1 0.07252 0.19376 0.374 0.70819

## guilt.1 2.31003 0.26867 8.598 < 2e-16 \*\*\*

## obguilt.1:guilt.1 0.14058 0.23411 0.600 0.54818

## reasdoubt.1:guilt.1 -0.61724 0.29693 -2.079 0.03764 \*

## bettertolet.1:guilt.1 0.02579 0.30123 0.086 0.93178

## libertyvorder.1:guilt.1 -0.27492 0.29355 -0.937 0.34899

## jurevidence.1:guilt.1 0.27601 0.36181 0.763 0.44555

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## (Dispersion parameter for binomial family taken to be 1)

##

## Null deviance: 490.08 on 354 degrees of freedom

## Residual deviance: 300.66 on 343 degrees of freedom

## AIC: 324.66

##

## Number of Fisher Scoring iterations: 6

The results are essentially the same; the constant and slopes of the predictors variables are different but the variables that were significant before still are. So standardizing doesn't change the results, but it is generally recommended because it makes results easier to interpret. The variables are centered around the mean, so negative numbers are below the mean, and positive numbers are above the mean.

We’ll start by loading/setting up the data and rerunning the binomial regression with interactions.

dissertation<-read.delim("dissertation\_data.txt",header=TRUE)  
dissertation<-dissertation[,1:44]  
predictors<-c("obguilt","reasdoubt","bettertolet","libertyvorder",  
 "jurevidence","guilt")  
dissertation<-subset(dissertation, !is.na(libertyvorder))  
  
dissertation[45:50]<-lapply(dissertation[predictors], function(x) {  
 y<-scale(x, center=TRUE, scale=TRUE)  
 }  
)  
  
pred\_int<-'verdict ~ obguilt.1 + reasdoubt.1 + bettertolet.1 + libertyvorder.1 +   
 jurevidence.1 + guilt.1 + obguilt.1\*guilt.1 + reasdoubt.1\*guilt.1 +  
 bettertolet.1\*guilt.1 + libertyvorder.1\*guilt.1 + jurevidence.1\*guilt.1'  
model<-glm(pred\_int, family="binomial", data=dissertation)  
summary(model)

##   
## Call:  
## glm(formula = pred\_int, family = "binomial", data = dissertation)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.6101 -0.5432 -0.1289 0.6422 2.2805   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.47994 0.16264 -2.951 0.00317 \*\*   
## obguilt.1 0.25161 0.16158 1.557 0.11942   
## reasdoubt.1 -0.09230 0.20037 -0.461 0.64507   
## bettertolet.1 -0.22484 0.20340 -1.105 0.26899   
## libertyvorder.1 0.05825 0.21517 0.271 0.78660   
## jurevidence.1 0.07252 0.19376 0.374 0.70819   
## guilt.1 2.31003 0.26867 8.598 < 2e-16 \*\*\*  
## obguilt.1:guilt.1 0.14058 0.23411 0.600 0.54818   
## reasdoubt.1:guilt.1 -0.61724 0.29693 -2.079 0.03764 \*   
## bettertolet.1:guilt.1 0.02579 0.30123 0.086 0.93178   
## libertyvorder.1:guilt.1 -0.27492 0.29355 -0.937 0.34899   
## jurevidence.1:guilt.1 0.27601 0.36181 0.763 0.44555   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 490.08 on 354 degrees of freedom  
## Residual deviance: 300.66 on 343 degrees of freedom  
## AIC: 324.66  
##   
## Number of Fisher Scoring iterations: 6

The predict function, which I introduced below, can also be used for the binomial model. Let’s have R generate predicted scores for everyone in the dissertation sample:

Code Chunks - Predict Function

dissertation<-**read.delim**("dissertation\_data.txt",header=TRUE)

guilt\_lm\_full<-**lm**(guilt ~ illev + circumst + deftest + policepower +

suspicious + overzealous + upstanding,

data=dissertation)

**options**(scipen = 999)

**summary**(guilt\_lm\_full)

##

## Call:

## lm(formula = guilt ~ illev + circumst + deftest + policepower +

## suspicious + overzealous + upstanding, data = dissertation)

##

## Residuals:

## Min 1Q Median 3Q Max

## -3.0357 -0.7452 0.1828 0.9706 2.5013

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 4.16081 0.38966 10.678 < 0.0000000000000002 \*\*\*

## illev 0.11111 0.05816 1.911 0.05689 .

## circumst -0.08779 0.06708 -1.309 0.19147

## deftest -0.02020 0.05834 -0.346 0.72942

## policepower 0.02828 0.06058 0.467 0.64090

## suspicious 0.17286 0.06072 2.847 0.00468 \*\*

## overzealous -0.03298 0.04792 -0.688 0.49176

## upstanding 0.08941 0.05374 1.664 0.09706 .

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 1.169 on 347 degrees of freedom

## (1 observation deleted due to missingness)

## Multiple R-squared: 0.07647, Adjusted R-squared: 0.05784

## F-statistic: 4.105 on 7 and 347 DF, p-value: 0.0002387

In this model, the outcome variable is a guilt rating, ranging from 1 to 7. This is our y, which our regression model is trying to recreate through the linear relationship between our x's and our y. Using the coefficients in the output, we could compute the predicted y (y-hat) - what a person's score would be if the linear model fit the data perfectly. Fortunately, R has a built-in function that will compute y-hat for a dataset: predict. This function requires two arguments: a regression model and the dataset to use to predict values. Let's have R predict values for the dissertation dataset, and add it on as a new variable.

dissertation$predicted<-**predict**(guilt\_lm\_full, dissertation)

In this application, we don't care as much about the predicted values - we will later on in this post - but we probably do care about the residuals: the difference between the observed value and the predicted value. This gives us an idea of how our model is doing and whether it fits reasonably well. It can also tell us if the model falls apart at certain values or ranges of values.  
  
In the residuals post, I showed that you can easily request residuals from the model. As we did with predicted, let's create a new variable in the dataset that contains our residuals.

dissertation$residual<-**resid**(guilt\_lm\_full)

**## Error in `$<-.data.frame`(`\*tmp\*`, residual, value = structure(c(0.0326393185592984, : replacement has 355 rows, data has 356**

Ruh-roh, we got an error. Our dataset contains 356 observations, but we only have 355 residuals. This is because someone has a missing value on one of the variables in the regression model and was dropped from the analysis. There are a variety of ways we could find out which case is missing a value, but since I'm only working with a handful of variables, I'll just run descriptives and look for the variable with only 355 values.

**library**(psych)

**## Warning: package 'psych' was built under R version 3.4.4**

**describe**(dissertation[**c**(13,15,18,21,27,29,31,44)])

## vars n mean sd median trimmed mad min max range skew

## illev 1 356 2.98 1.13 3 3.02 1.48 1 5 4 -0.17

## circumst 2 356 2.99 0.95 3 2.97 1.48 1 5 4 0.13

## deftest 3 356 3.39 1.46 4 3.57 0.00 -9 5 14 -5.25

## policepower 4 355 3.86 1.41 4 4.02 0.00 -9 5 14 -6.40

## suspicious 5 356 2.09 1.14 2 2.01 1.48 -9 5 14 -1.97

## overzealous 6 356 3.34 1.34 4 3.41 1.48 -9 5 14 -4.49

## upstanding 7 356 3.09 1.29 3 3.11 1.48 -9 5 14 -2.31

## guilt 8 356 4.80 1.21 5 4.90 1.48 2 7 5 -0.59

## kurtosis se

## illev -1.04 0.06

## circumst -0.51 0.05

## deftest 40.74 0.08

## policepower 55.05 0.08

## suspicious 23.52 0.06

## overzealous 38.44 0.07

## upstanding 19.66 0.07

## guilt -0.54 0.06

The variable policepower is the culprit. I can drop that missing value then rerun the residual code.

dissertation<-**subset**(dissertation, !**is.na**(policepower))

dissertation$residual<-**resid**(guilt\_lm\_full)

Now I can plot my observed values and residuals.

**library**(ggplot2)

*##*

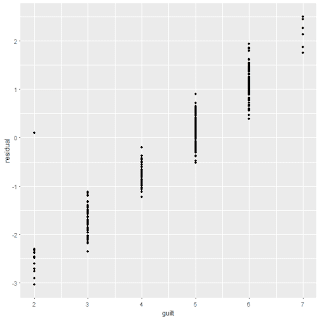
*## Attaching package: 'ggplot2'*

*## The following objects are masked from 'package:psych':*

*##*

*## %+%, alpha*

**qplot**(guilt,residual,data=dissertation)

[](https://3.bp.blogspot.com/-79EBNOn8WAI/WuDstFWYnfI/AAAAAAAALZI/O9mMncr0530sYsEhk0wgyo-XbFduR-_uACLcBGAs/s1600/unnamed-chunk-6-1.png)

We want our residuals to fall around 0, which is only happening for guilt ratings near the midpoint of the scale. This suggests that ordinary least squares regression may not be the best fit for the data, as the model shows a tendency to overpredict (negative residual) guilt rating for people with lower observed ratings and underpredict (positive residual) for people with higher observed ratings.  
  
But, as I often do on this blog for the sake of demonstration, let's pretend the model is doing well. One way we could use a regression model is to predict scores in a new sample. For instance, there are multiple rumors that different graduate schools have prediction equations they use to predict a candidate's anticipated graduate school GPA, based on a combination of factors asked about in the application packet, to determine if a person is grad school-ready (and ultimately, to decide if they should be admitted). Schools generally won't confirm they do this, nor would they ever share the prediction equation, should such an equation exist. But this is one way regression results from a training sample could be used to make decisions on a testing sample. So let's do that.  
  
Unfortunately, I don't have a second dissertation dataset laying around that I could apply this equation to, but I could take a note from the data science playbook, and randomly divide my sample into training and testing datasets. I use the training dataset to generate my equation, and I use the testing dataset to apply my equation and predict values. Since I have outcome variable data in the testing dataset too, I can see how well my model did. Once I have a well-performing model, I could then apply it to new data, maybe to predict how highly you'll rate a book or movie, or to generate recommendations.  
  
First, I'll split my dataset in half.

smp\_size <- **floor**(0.50 \* **nrow**(dissertation))

**set.seed**(42)

train\_ind <- **sample**(**seq\_len**(**nrow**(dissertation)), size = smp\_size)

train <- dissertation[train\_ind, ]

test <- dissertation[-train\_ind, ]

Now I have a train dataset, with 177 observations, and a test dataset with 178. I can rerun the same linear model as before, this time only using the training data.

guilt\_lm\_train<-**lm**(guilt ~ illev + circumst + deftest + policepower +

suspicious + overzealous + upstanding,

data=train)

**summary**(guilt\_lm\_train)

##

## Call:

## lm(formula = guilt ~ illev + circumst + deftest + policepower +

## suspicious + overzealous + upstanding, data = train)

##

## Residuals:

## Min 1Q Median 3Q Max

## -2.9420 -0.8359 0.1641 0.9371 2.3151

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 5.28874 0.77150 6.855 0.000000000128 \*\*\*

## illev 0.08866 0.08485 1.045 0.29759

## circumst -0.13018 0.09917 -1.313 0.19109

## deftest -0.25726 0.10699 -2.405 0.01727 \*

## policepower 0.01758 0.12316 0.143 0.88665

## suspicious 0.25716 0.08857 2.903 0.00419 \*\*

## overzealous -0.11683 0.08240 -1.418 0.15807

## upstanding 0.10371 0.07574 1.369 0.17273

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 1.194 on 169 degrees of freedom

## Multiple R-squared: 0.1265, Adjusted R-squared: 0.09027

## F-statistic: 3.495 on 7 and 169 DF, p-value: 0.001586

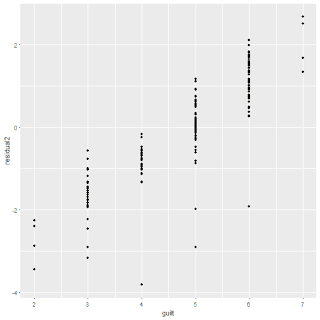
I can use my predict function to predict scores for the testing dataset. Remember, all this function needs is the linear model name and a dataset to use for the prediction function - and it can be any dataset, as long as it contains the same variables from the model.

test$predicted2<-**predict**(guilt\_lm\_train, test)

The original predicted value (from when I was working with the full dataset) is still in this set. I could have replaced values by using the same variable name, but just for fun, decided to keep those values and create a second prediction variable.  
  
Because we have observed and predicted2 for our training dataset, let's see how well our model did, by creating a new residual variable, residual2. We can't use the resid function, because we didn't have R perform a linear regression on the testing dataset, but we can easily generate this variable by subtracting the predicted score from the observed score. Then we can once again plot our observed and residual values.

test$residual2<-test$guilt-test$predicted2

**qplot**(guilt,residual2,data=test)

[](https://4.bp.blogspot.com/-l3aOP8fleH4/WuDsxTvw4KI/AAAAAAAALZM/7vIhJ6CBJjIJAktOOBtDod5_TM_fOg41QCLcBGAs/s1600/unnamed-chunk-10-1.png)

We're still seeing similar issues with the residuals as we did for the full dataset. If we wanted to actually apply our linear model, we'd want to do more research and pilot work to get the best equation we can. As with many things in statistics, the process is heavily determined by what you plan to do with the results. If you want to report variables that have a strong linear relationship with an outcome, we might be fine with these regression results. If we want to build an equation that predicts an outcome with a minimum of error, we'd want to do more exploratory work, pulling in new variables and dropping low-performing ones. Many of the predictors in the model were not significant, so we could start model-building from those results. We may need to build multiple training datasets, to ensure we aren't only picking up chance relationships. And for much larger applications, such as recommendation systems on services like Amazon and Netflix, machine learning may be a better, more powerful method.

dissertation$predver<-predict(model)  
dissertation$predver

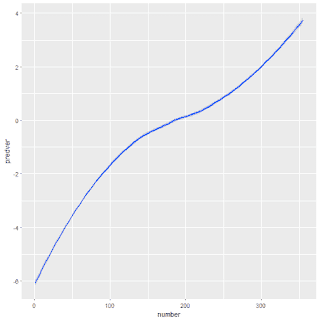
## [1] 0.3907097456 -4.1351129605 2.1820478279 -2.8768390246 2.5804618523  
## [6] 0.4244692909 2.3065468369 -2.7853434926 0.3504760502 -0.2747339639  
## [11] -1.8506160725 -0.6956240161 -4.7860574839 -0.3875950731 -2.4955679446  
## [16] -0.3941516951 -4.5831011509 1.6185480937 0.4971923298 4.1581842900  
## [21] -0.6320531052 -4.8447046319 -2.3974890696 1.8566258698 0.0360685822  
## [26] 2.2151040131 2.3477149003 -2.4493726369 -0.2253481404 -4.8899805287  
## [31] 1.7789459288 -0.0978703861 -3.5541042186 -3.6009218603 0.1568318789  
## [36] 3.7866003489 -0.6371816898 -0.7047761441 -0.7529742376 -0.0302759317  
## [41] -0.1108055330 1.9751810033 0.2373614802 0.0424471071 -0.4018757856  
## [46] 0.0530272726 -1.0763759980 0.0099577637 0.3128581222 1.4806679691  
## [51] -1.7468626219 0.2998282372 -3.6359162016 -2.2200774510 0.3192366472  
## [56] 3.0103216033 -2.0625775984 -6.0179845235 2.0300503627 2.3676828409  
## [61] -2.8971753746 -3.2131490026 2.1349358889 3.0215336139 1.2436192890  
## [66] 0.2885535375 0.2141821004 1.9480686936 0.0438751446 -1.9368013875  
## [71] 0.2931258287 0.5319938265 0.0177643261 3.3724920900 0.0332949791  
## [76] 2.5935500970 0.7571810150 0.7131757400 2.5411073339 2.8499853550  
## [81] 2.8063291084 -0.4500738791 1.4700679077 -0.8659309719 0.0870492258  
## [86] 0.5728074322 0.1476797509 2.4697257261 2.5935500970 -2.2200774510  
## [91] -0.0941827753 1.3708676633 1.4345235392 -0.2407209578 2.4662700339  
## [96] -1.9687731888 -6.7412580522 -0.0006224018 -4.4132951092 -2.8543032695  
## [101] 1.2295635352 2.8194173530 0.1215689324 -3.8258079371 1.8959803882  
## [106] -4.5578801595 2.3754402614 0.0826808026 1.5112359711 -3.5402060466  
## [111] 0.2556657363 0.7054183194 1.4675797244 -2.3974890696 2.6955929822  
## [116] -0.3123518919 -4.8431862346 -2.0132721372 0.4673405434 -2.3053405270  
## [121] 1.9498822386 -0.5164183930 -1.8277820872 -0.0134750769 -2.3013547136  
## [126] -0.2498730859 -4.4281010683 -0.0134750769 -0.2604532514 0.1476797509  
## [131] -2.3392939519 -2.0625775984 -3.5541042186 1.5087477879 -4.6453051124  
## [136] 2.0616474606 -3.2691362859 -7.3752231145 -1.6666447439 1.0532964013  
## [141] -2.0625775984 -0.3355312717 2.2481601983 -2.2200774510 -4.3276959075  
## [146] 0.8685972087 -0.7727065311 1.7511589809 -0.4774548995 0.0008056357  
## [151] 1.7022334970 -0.4202625135 -0.2902646169 2.4409712692 0.0008056357  
## [156] 0.0008056357 -3.6009218603 -0.8567788439 -0.4528474822 0.3517462520  
## [161] 0.1307210605 -3.7843118182 -2.8419024763 -3.5191098774 -0.1460684795  
## [166] 1.8809888141 2.8194173530 -2.4656469123 1.0589888029 0.1659840070  
## [171] 1.4345235392 2.3676828409 1.5749534339 -0.1681557545 2.6406620359  
## [176] 0.1476797509 -2.2135177411 1.9168260534 -3.4993205379 0.4557086940  
## [181] -3.8136089417 -0.1121510987 -3.9772095600 1.3849234171 0.3504760502  
## [186] 2.3807710856 -3.0667307601 2.3040586537 1.7599138086 -0.2083894500  
## [191] 0.6844579761 -0.3552635652 -1.9459392035 -0.6075281598 -2.1663310490  
## [196] 2.3676828409 -1.9205271122 -2.2334295071 -4.4265826710 -1.0117771483  
## [201] -0.0161530548 -0.3072233074 -0.0161530548 -0.7451676752 -7.0351269313  
## [206] 2.6406620359 -3.7523234832 -0.2498730859 2.0222929422 3.2886316225  
## [211] -1.6221457956 2.4749949634 1.7570711677 0.0904873650 -4.7332807307  
## [216] 0.1568318789 -0.0302759317 0.5127229828 1.3097316594 -6.9309218514  
## [221] 0.0515992352 -0.4514194447 -0.2253481404 -4.7652690656 -0.4279866041  
## [226] -4.4136563866 -3.7618312672 0.0156676181 -0.2590252139 2.6076058507  
## [231] 1.6420333133 -3.9985172969 -6.2076483227 0.1632104039 0.1829426974  
## [236] -4.7652690656 -4.4212844958 1.6001906117 0.8579971472 -3.8699110198  
## [241] 0.3022779567 -0.1679979189 1.9421248181 0.6592738895 1.6132788564  
## [246] -0.0366544567 -3.4818233673 -3.9422152187 -0.3473613776 0.4321933815  
## [251] 0.7480288869 -0.2498730859 -1.9861068488 -2.2297920164 -0.7621263656  
## [256] 1.2966434147 0.1632104039 0.2048721368 1.7789459288 0.4926393080  
## [261] 0.4096285430 -1.7794744955 -2.5822853071 2.0413250624 -6.6574350219  
## [266] -0.1277642235 -2.1972434657 -2.5075677545 -0.4482774141 -0.6943740757  
## [271] -0.7821891015 6.3289445390 0.1568318789 0.1165981835 1.4781797859  
## [276] -4.2287015488 -3.6157278195 -0.1511970641 -0.7047761441 2.0935344484  
## [281] -3.8258079371 -4.4231102471 1.3097316594 3.4081542651 -0.4996175382  
## [286] -2.0534397824 0.9783975145 -2.2562634924 3.7196170683 1.1110084017  
## [291] 2.1661785291 -4.2138955896 1.9421248181 2.3065468369 -0.7139282722  
## [296] -4.1431023472 -2.0854115837 2.9389399956 1.7711269214 -0.0302759317  
## [301] -2.6458711124 0.5856241187 -0.1199576611 1.8566258698 -2.2383553905  
## [306] 2.3807710856 -0.2838860920 3.1176953128 2.8499853550 2.8063291084  
## [311] 0.0034011417 -0.4683781352 -3.0377484314 -1.3833686805 1.7764577456  
## [316] 1.7842151661 3.4081542651 0.1165981835 -4.6988069009 -2.6013721641  
## [321] 2.0616474606 -0.2498730859 -4.2207121622 4.1705330009 5.2103776377  
## [326] -4.5406977837 -1.5080855068 -2.5232652805 -5.7259789038 2.5211393933  
## [331] -0.3487069432 -2.5035573312 -2.2764097339 -5.8364854607 -1.8694684539  
## [336] 1.3402996614 0.5728074322 0.3663267540 -0.1603491921 -2.1690805453  
## [341] -1.4105339689 3.0768201201 -5.1065624241 -4.5966850670 -4.5498907729  
## [346] -1.3078399029 -1.0882592824 0.3128581222 -0.3644156933 0.3100845191  
## [351] 2.4774831467 -1.0763759980 2.2151040131 -0.0952748801 -4.6864864366

Now, remember that the outcome variable is not guilty (0) and guilty (1), so you might be wondering – what’s with these predicted values? Why aren’t they 0 or 1?

Binomial regression is used for nonlinear outcomes. Since the outcome is 0/1, it’s nonlinear. But binomial regression is based on the general linear model. So how can we apply the general linear model to a nonlinear outcome? Answer: by transforming scores. Specifically, it transforms the outcome into a log odds ratio; the log transform makes the outcome variable behave somewhat linearly and symmetrically. The predicted outcome, then, is also a log odds ratio.

ordvalues<-dissertation[order(dissertation$predver),]  
ordvalues<-ordvalues[,51]  
ordvalues<-data.frame(1:355,ordvalues)  
colnames(ordvalues)<-c("number","predver")  
library(ggplot2)  
ggplot(data=ordvalues, aes(number,predver))+geom\_smooth()

## `geom\_smooth()` using method = 'loess'

[](https://i0.wp.com/3.bp.blogspot.com/-t68kCtjcevo/WueGreJepKI/AAAAAAAALbs/QsSzDQL-QjgO27QRZkuhkXsZrqP9GRTiQCLcBGAs/s1600/unnamed-chunk-3-1.png?ssl=1)

Log odds ratios are great for analysis, but when trying to understand how well your model is predicting values, we want to convert them into a metric that’s easier to understand in isolation and when compared to the observed values. We can convert them into probabilities with the following equation:

dissertation$verdict\_predicted<-exp(predict(model))/(1+exp(predict(model)))

This gives us a value ranging from 0 to 1, which is the probability that a particular person will select guilty. We can use this value in different ways to see how well our model is doing. Typically, we’ll divide at the 50% mark, so anyone with a probability of 0.5 or greater is predicted to select guilty, and anyone with a probability less than 0.5 would be predicted to select not guilty. We then compare this new variable with the observed results to see how well the model did.

dissertation$vpred\_rounded<-round(dissertation$verdict\_predicted,digits=0)  
library(expss)

## Warning: package 'expss' was built under R version 3.4.4

dissertation<- apply\_labels(dissertation,  
 verdict = "Actual Verdict",  
 verdict = c("Not Guilty" = 0,  
 "Guilty" = 1),  
 vpred\_rounded = "Predicted Verdict",  
 vpred\_rounded = c("Not Guilty" = 0,  
 "Guilty" = 1)  
)  
cro(dissertation$verdict,list(dissertation$vpred\_rounded, total()))

|  | **Predicted Verdict** | |  | **#Total** |
| --- | --- | --- | --- | --- |
|  | **Not Guilty** | **Guilty** |  |  |
| **Actual Verdict** | | | | |
| Not Guilty | 152 | 39 |  | 191 |
| Guilty | 35 | 129 |  | 164 |
| #Total cases | 187 | 168 |  | 355 |

One thing we could look at regarding this table, which when dealing with actual versus predicted categories is known as a confusion matrix, is how well the model did at correctly categorizing cases – which we get by adding together the number of people with both observed and predicted not guilty, and people with observed and predicted guilty, then dividing that sum by the total.

accuracy<-(152+129)/355  
accuracy

## [1] 0.7915493

Our model correctly classified 79% of the cases. However, this is not the only way we can determine how well our model did. There are a [variety of derivations](https://en.wikipedia.org/wiki/Sensitivity_and_specificity#Definitions) you can make from the confusion matrix. But two you should definitely include when doing this kind of analysis are sensitivity and specificity. Sensitivity refers to the true positive rate, and specificity refers to the true negative rate.

When you’re working with confusion matrices, you’re often trying to diagnose or identify some condition, one that may be deemed positive or present, and the other that may be deemed negative or absent. These derivations are important because they look at how well your model identifies these different states. For instance, if most of my cases selected not guilty, I could get a high accuracy rate by simply predicting that *everyone* will select not guilty. But then my model lacks sensitivity – it only identifies negative cases (not guilty) and fails to identify any positive cases (guilty). If I were dealing with something even higher stakes, like whether a test result indicates the presence of a condition, I want to make certain my classification is sensitive to those positive cases. And vice versa, I could keep from missing any positive cases by just classifying everyone as positive, but then my model lacks specificity and I may subject people to treatment they don’t need (and that could be harmful).

Just like accuracy, sensitivity and specificity are easy to calculate. As I said above, I’ll consider not guilty to be negative and guilty to be positive. Sensitivity is simply the number of true positives (observed and predicted guilty) divided by the sum of true positives and false negatives (people who selected guilty but were classified as not guilty).

sensitivity<-129/164  
sensitivity

## [1] 0.7865854

And specificity is the number of true negatives (observed and predicted not guilty) divided by the sum of true negatives and false positives (people who selected not guilty but were classified as guilty).

specificity<-152/191  
specificity

## [1] 0.7958115

So the model correctly classifies 79% of the positive cases and 80% of the negative cases. The model could be improved, but it’s functioning equally well across positive and negative cases, which is good.

It should be pointed out that you can select any cutpoint you want for your probability variable. That is, if I want to be very conservative in identifying positive cases, I might want there to be a higher probability that it is a positive case before I classify it as such – perhaps I want to use a cutpoint like 75%. I can easily do that.

dissertation$vpred2[dissertation$verdict\_predicted < 0.75]<-0  
dissertation$vpred2[dissertation$verdict\_predicted >= 0.75]<-1  
dissertation<- apply\_labels(dissertation,  
 vpred2 = "Predicted Verdict (0.75 cut)",  
 vpred2 = c("Not Guilty" = 0,  
 "Guilty" = 1)  
)  
cro(dissertation$verdict,list(dissertation$vpred2, total()))

|  | **Predicted Verdict (0.75 cut)** | |  | **#Total** |
| --- | --- | --- | --- | --- |
|  | **Not Guilty** | **Guilty** |  |  |
| **Actual Verdict** | | | | |
| Not Guilty | 177 | 14 |  | 191 |
| Guilty | 80 | 84 |  | 164 |
| #Total cases | 257 | 98 |  | 355 |

accuracy2<-(177+84)/355  
sensitivity2<-84/164  
specificity2<-177/191  
accuracy2

## [1] 0.7352113

sensitivity2

## [1] 0.5121951

specificity2

## [1] 0.9267016

Changing the cut score improves specificity but at the cost of sensitivity, which makes sense, because our model was predicting equally well (or poorly, depending on how you look at it) across positives and negatives. In this case, a different cut score won’t improve our model. We would need to go back and see if there are better variables to use for prediction. And to keep us from fishing around in our data, we’d probably want to use a training and testing set for such exploratory analysis.