When working with a prediction model, like a linear regression, there are a few Ys you need to concern yourself with: the ys (observed outcome variable), the y-hats (predicted outcome variables based on the equation), and the [residuals](http://www.deeplytrivial.com/2018/02/statistics-sunday-what-are-residuals.html) (y minus y-hat). Today, I’ll dig into the different flavors of y and how you might work with them when conducting linear regression.

As my data example, I’ll use my [dissertation](https://www.dropbox.com/s/683w91a6aft0ecf/Locatelli_dissertation.pdf?dl=0) dataset and the linear model I introduced in the [GLM post](http://www.deeplytrivial.com/2018/04/g-is-for-glm-function.html).

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)

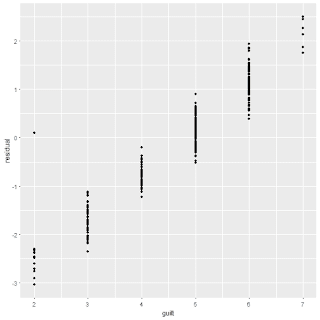
Now I can plot my observed values and residuals.

library(ggplot2)

##   
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':  
##   
## %+%, alpha

qplot(guilt,residual,data=dissertation)

[](https://i2.wp.com/3.bp.blogspot.com/-79EBNOn8WAI/WuDstFWYnfI/AAAAAAAALZI/O9mMncr0530sYsEhk0wgyo-XbFduR-_uACLcBGAs/s1600/unnamed-chunk-6-1.png?ssl=1)

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, or even to determine if I should let you in to the super-elite [Monstersori school](https://en.wikipedia.org/wiki/Avenue_Q) I want to create.

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

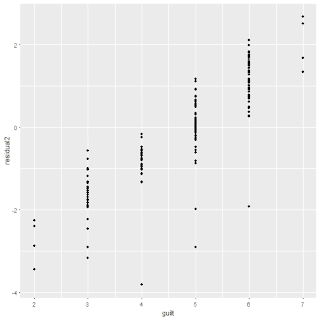
I can use my predict function to predict scores for the testing dataset. Remember, all this function needs is the linear model name and a dataset to use for the prediction function – and it can be any dataset, as long as it contains the same variables from the model.

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://i2.wp.com/4.bp.blogspot.com/-l3aOP8fleH4/WuDsxTvw4KI/AAAAAAAALZM/7vIhJ6CBJjIJAktOOBtDod5_TM_fOg41QCLcBGAs/s1600/unnamed-chunk-10-1.png?ssl=1)

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 equations 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.