CS 251 Statistical Computing

HOP 7: R for statistical project

4/15/2020 Developed by Aya Khalil

5/12/2020 Reviewed by Kim Nguyen

School of Technology and Computing @City University of Seattle (CityU)

**Before You Start**

* If you already finished this module through any CityU Technology Institute (TI) courses,  
  just skim this module and skip it.
* Version numbers may not match with the guide. But that should be fine.  
  If given the option to choose between stable release (long-term support) or most recent, please choose the stable release.
* This guide targets Windows OS users. So, MacOS users may have different commands to input in the shell/terminal.
* We cannot explain every step. **This cookbook always needs your own creative judgement.**
* **For your working directory, use your course number.** The hands-on tutorial may use a different course number as an example.

**Learning Outcomes**

* Linear Regression

**Resource**

* Hui, E. G. M. (2019). [*Learn R for applied statistics: With data visualization, regressions, and statistics*](https://login.proxy.cityu.edu/sso/skillport?context=144516). Apress.
* Data Science and Machine Learning BootCamp with R online course

**Machine Learning**

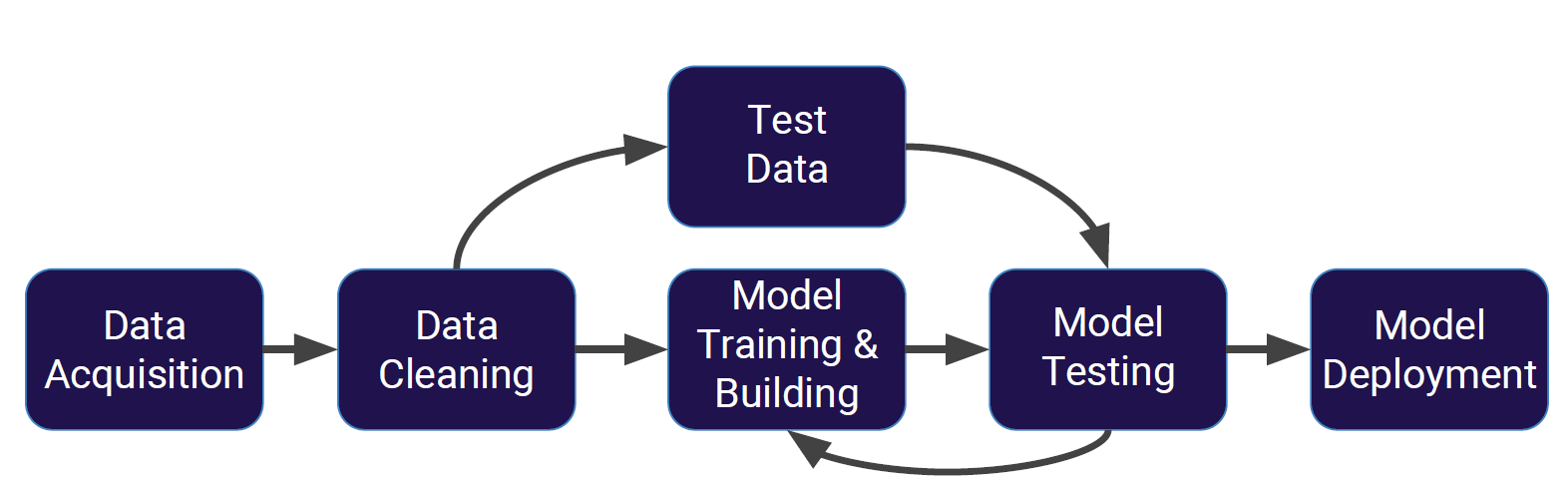
Machine learning is a method of data analysis that automates analytical model building.

Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.

Machine learning has so many applications, for example:

* Fraud detection.
* Web search results.
* Real-time ads on web pages
* Credit scoring and next-best offers.
* Prediction of equipment failures.
* New pricing models.
* Network intrusion detection.
* Recommendation Engines
* Customer Segmentation
* Text Sentiment Analysis
* Predicting Customer Churn
* Pattern and image recognition.
* Email spam filtering.
* Financial Modeling

The general machine learning process:



This is the process we’re going to implement as we learn how to do machine learning with R

**Data Acquisition**: In this step, we need to acquire our data. These data will be the CSV files for the most part or built in data frames or data sets.

**Data Cleaning**: Once we acquired the data, we will need to clean that data and will need to do some exploratory data analysis.

After cleaning the data, we will split that clean data into test data and training data.

**What’s the difference between test data and training data?**

**Training Data**

The observations in the training set form the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

**Test Data**

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

**For example**, you have a dataset of students with their demographics, hours spent practicing for the SAT, books they’ve read throughout the year, times spent per day studying for the test, and the SAT results for years 2005–2016. This is your training dataset, since the results of their test scores already known. The algorithm will train to predict each score, based on the students’ parameters.

You have a similar dataset of students, with the same data, except for the SAT scores. This is your testing dataset, and the algorithm will predict the outcome, based on the historical data for similar students.

Back to the previous graph, after splitting the data into test data and training data, we will train a machine learning model on our training data. Once we’ve trained the model what we are going to do is test our model on the test data that we went ahead and split off your clean data.

This allows you to accurately evaluate your model science that machine learning model hasn’t actually seen that test data yet. Then, we can repeat this process to our model until we’re satisfied with it. Once we finish testing our model, we can deploy our model.

**Let’s talk about the** **three major types of machine learning algorithms**

**Supervised learning**:

Supervised learning algorithms are trained using labeled examples, such as an input where the desired

output is known.

* For example, a piece of equipment could have data points labeled either “F” (failed) or “R” (runs).

The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly.

Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data.

Supervised learning is commonly used in applications where historical data predicts likely future events.

**For example**, it can anticipate when credit card transactions are likely to be fraudulent or which insurance customer is likely to file a claim.

Or it can attempt to predict the price of a house based on different features for houses for which we have historical price data.

**Unsupervised learning**

Unsupervised learning is used against data that has no historical labels. The system is not told the "right answer." The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within.

Or it can find the main attributes that separate customer segments from each other.

● Popular techniques include self-organizing maps, nearestneighbor mapping, k-means clustering and singular value decomposition.

**Reinforcement learning**

Reinforcement learning is often used for robotics, gaming and navigation.

With reinforcement learning, the algorithm discovers through trial and error which actions yield the greatest rewards.

This type of learning has three primary components: the agent (the learner or decision maker), the environment (everything the agent interacts with) and actions (what the agent can do).

The objective is for the agent to choose actions that maximize the expected reward over a given amount of time. The agent will reach the goal much faster by following a good policy.

So, the goal in reinforcement learning is to learn the best policy.

Let’s use R for machine learning. Our first topic is Linear Regression. If you want to understand the linear regression in details, Chapters 2 & 3 of Introduction to Statistical Learning By Gareth James, et al, book are recommended to read, you can get the free pdf book from the following link: <http://faculty.marshall.usc.edu/gareth-james/ISL/> then, choose Download the book PDF

**Linear Regression**

**History**

This all started in the 1800s with a guy named Francis Galton. Galton was studying the relationship between parents and their children. In particular, he investigated the relationship between the heights of fathers and their sons.

What he discovered was that a man's son tended to be roughly as tall as his father. However, Galton's breakthrough was that the son's height **tended to be closer to the overall average** height of all people.

**Example:**

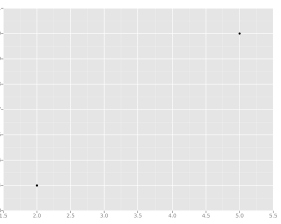
Let's take Shaquille O'Neal as an example. Shaq is really tall:7ft 1in (2.2 meters). If Shaq has a son, chances are he'll be pretty tall too. However, Shaq is such an anomaly that there is also a very good chance that his son will be not be as tall as Shaq.

Turns out this is the case:

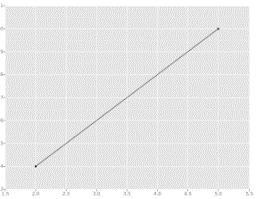
Shaq's son is pretty tall (6 ft 7 in), but not nearly as tall as his dad. Galton called this phenomenon regression, as in "A father's son's height tends to regress (or drift towards) the mean (average) height."

**Example**

Let's take the simplest possible example: calculating a regression with only 2 data points.



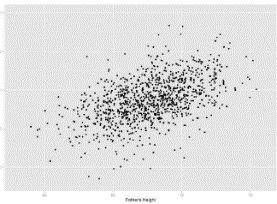
All we're trying to do when we calculate our regression line is draw a line that's as close to every dot as possible. For classic linear regression, or "Least Squares Method", you only measure the closeness in the "up and down" direction



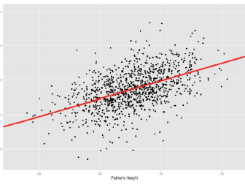
**Example**:

Now wouldn't it be great if we could apply this same concept to a graph with more than just two data points?

By doing this, we could take multiple men and their son's heights and do things like tell a man how tall we expect his son to be...before he even has a son!



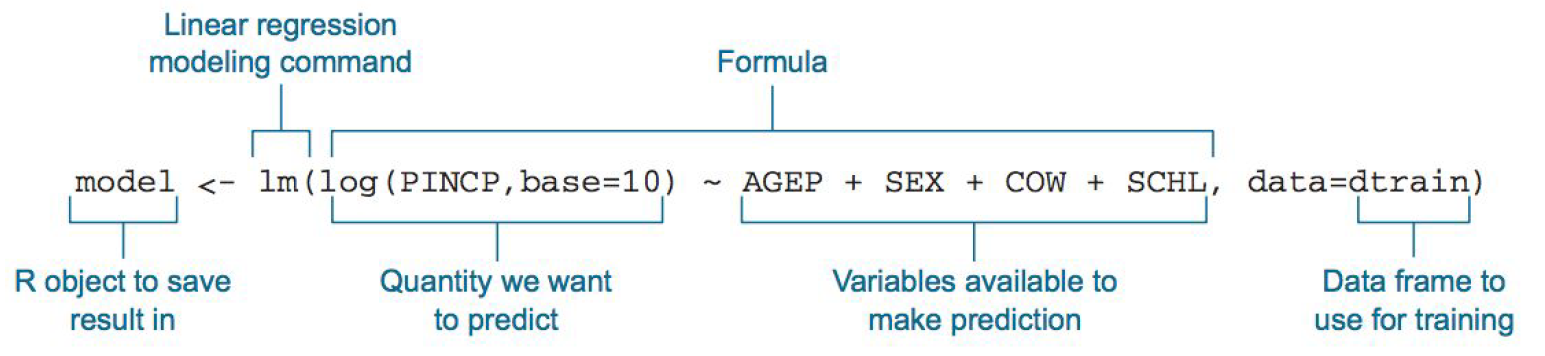
Our goal with linear regression is to minimize the vertical distance between all the data points and our line. So in determining the best line, we are attempting to minimize the distance between all the points and their distance to our line.

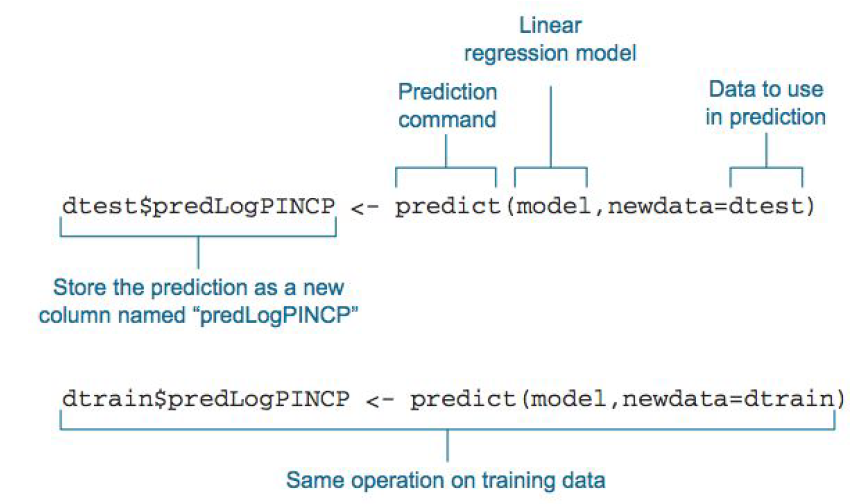


There are lots of different ways to minimize this, (sum of squared errors, sum of absolute errors, etc), but all these methods have a general goal of minimizing this distance.

**Using R for Linear Regression**

Formulas in R take the form (y ~ x). To add more predictor variables, just use the + sign. i.e.(y ~ x + z).





**Remember that Linear Regression is a supervised learning algorithm, meaning we'll have labeled data and try to predict new labels on unlabeled data. We'll explore some of the following concepts:**

* Get our Data
* Exploratory Data Analysis (EDA)
* Clean our Data
* Review of Model Form
* Train and Test Groups
* Linear Regression Model

**Please watch the following video to understand the concept of linear regression**

*“Machine Learning Algorithms | Machine Learning Tutorial | Data Science Training | Edureka”*

at <https://www.youtube.com/watch?v=Up6KLx3m2ww&list=PL9ooVrP1hQOEaSSxHVGRUdridqxgVDdEV&index=2>

*“Linear Regression in R | Linear Regression in R With Example | Data Science Algorithms | Simplilearn”*  at <https://www.youtube.com/watch?v=2Sb1Gvo5si8>

**Setup Working Environment for Module7**

1. Open VS Code.
2. Go to your project folder

* **online student:** Open CS251 \_Fall\_2020/**ON**/FirstnameLastname /. ( File > Open )
* **onsite student:** Open CS251 \_ Fall \_2020/**IN**/FirstnameLastname. ( File > Open )

1. Then create “**Module7**” directory in the VSCode.

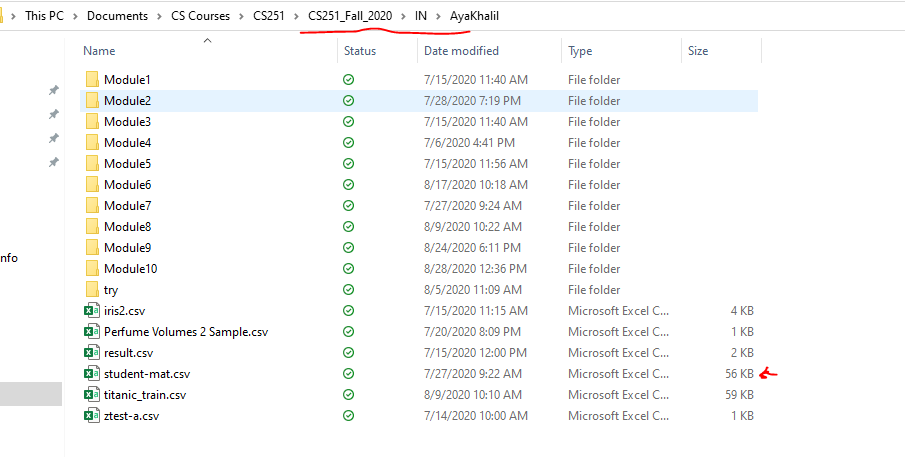
>>>mkdir Module7

**-In Module7** project folder, create new file LinerReg.R

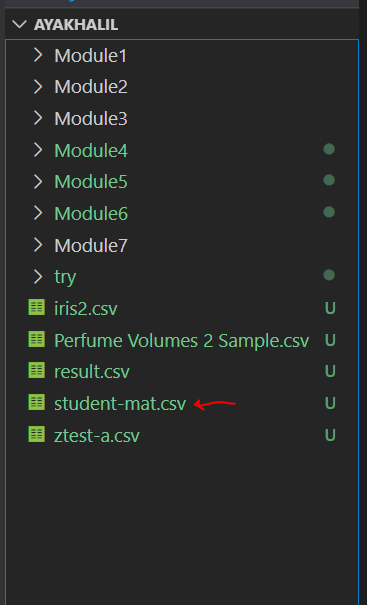
**Get our Data (Data Acquisition)**

We will use the [Student Performance Data Set from UC Irvine's Machine Learning Repository!](https://archive.ics.uci.edu/ml/datasets/Student+Performance)  **Download the supplied csv file that with the HOP. We'll specifically look at the math class (student-mat.csv).** Make sure to take note that the delimiter is a semi-colon.

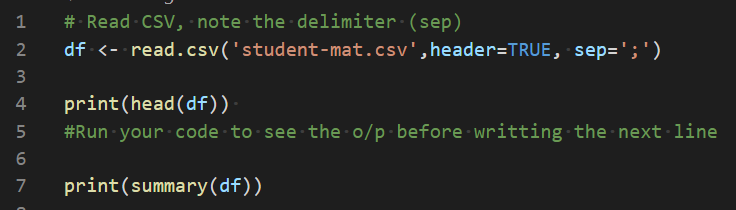
Make sure to add the student-mat.csv file in your working directory



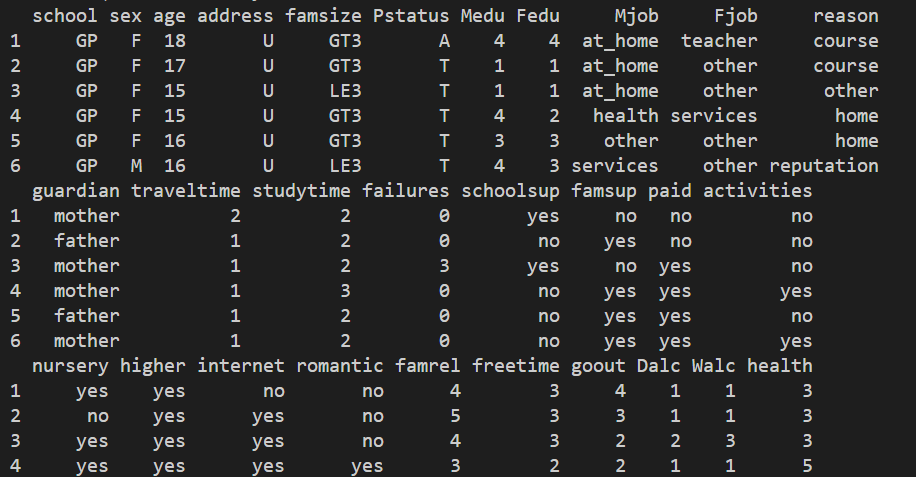
When you add it, you will see it in your vscode files



-Type the following in LinerReg.R file



-Save & Run

-o/p 

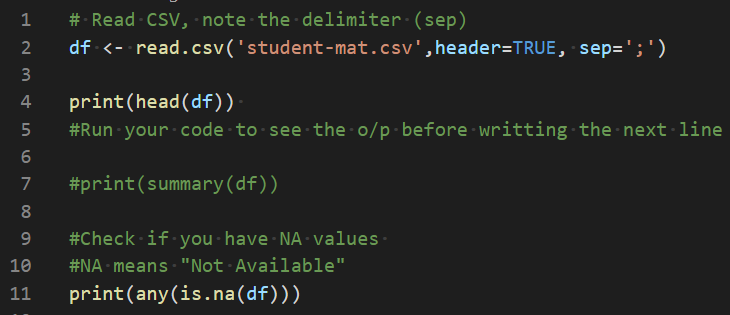
-**Please make sure to read & understand the dataset so what we are going to do in this HOP makes sense to you.**

- You can read the description of the dataset from the link above or you can read the below description of the dataset attributes.

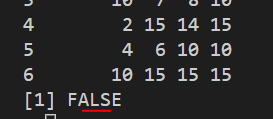


After we have our data (**Data Acqusion**), now we need to see if the data is clean or not and if not, then we need to clean it (**Data Cleaning**).

-Add the following to update LinerReg.R file



-Save & Run’

-o/p: 

-The output FALSE means that our data is clean and we don’t need to clean it. If it’s not clean and we have some variables that’s not available then we need to clean it.

-Most real data sets will probably have NA or Null values, so it is always good to check! It’s up to you how to deal with them, either dropping them if they aren't too many, or imputing other values, like the mean value.

**Exploratory Data Analysis**

Let's use ggplot2 to explore the data a bit.

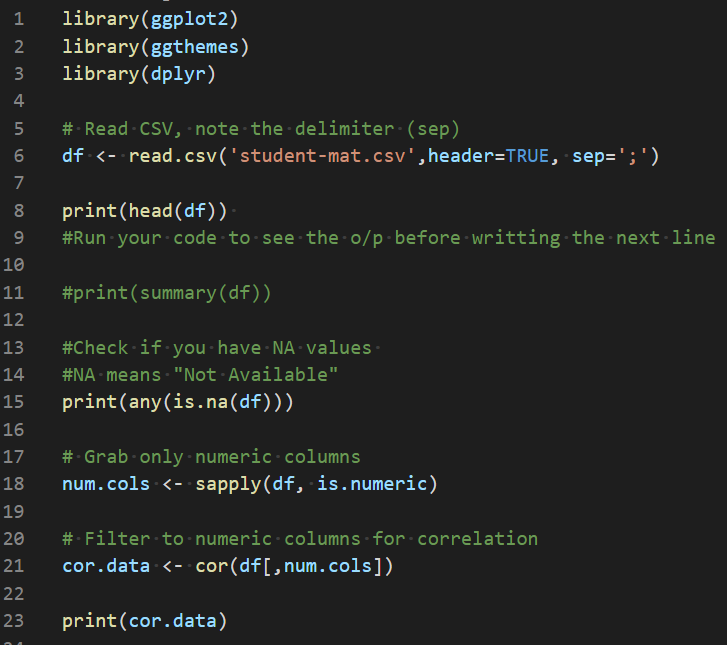
**Correlation and CorrPlots**

From Wikipedia, correlation is defined as:

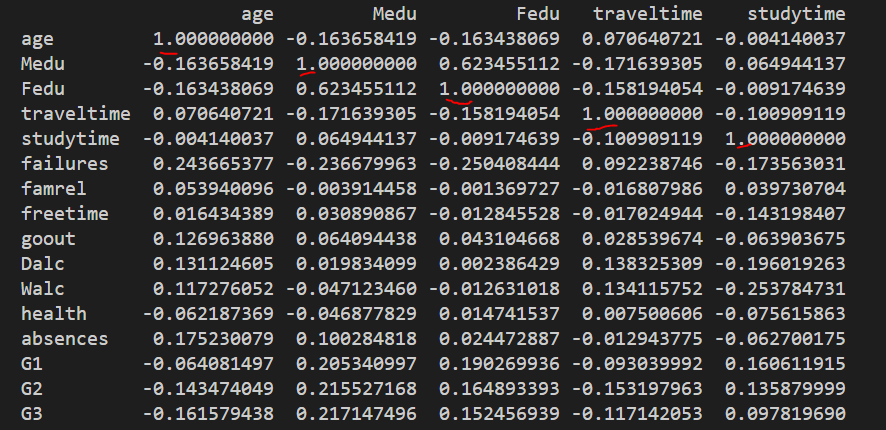
In statistics, dependence or association is any statistical relationship, whether causal or not, between two random variables or two sets of data. Correlation is any of a broad class of statistical relationships involving dependence, though in common usage it most often refers to the extent to which two variables have a linear relationship with each other. Familiar examples of dependent phenomena include the correlation between the physical statures of parents and their offspring, and the correlation between the demand for a product and its price.

Correlation plots are a great way of exploring data and seeing if there are any interaction terms. Let's start off by just grabbing the numeric data (we can't see correlation for categorical data):

-Add the following to update LinerReg.R file



-Save & Run

-O/P

You will see that each value is correlated with itself

**While this is fantastic information, it's hard to take it all in**. Let's visualize all this data. There are lots of amazing 3rd party packages to do this, let's use and install the 'corrgram' package and the corrplot package. This will also install a bunch of dependencies for the package.

**-Type the following in R console to download the packages that will allow us to see the correlation between the values**

install.packages('corrgram',repos = 'http://cran.us.r-project.org')

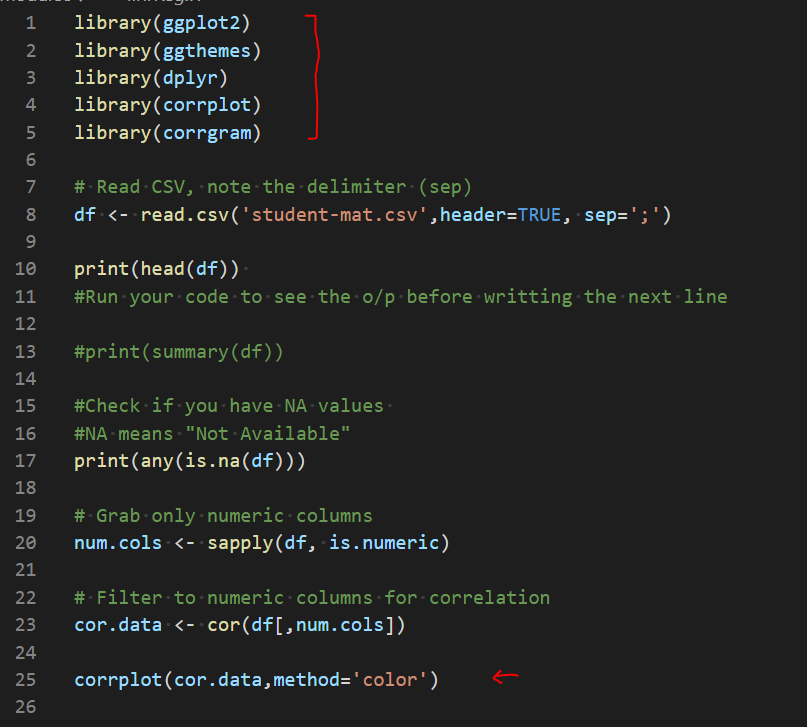


install.packages('corrplot',repos = 'http://cran.us.r-project.org')

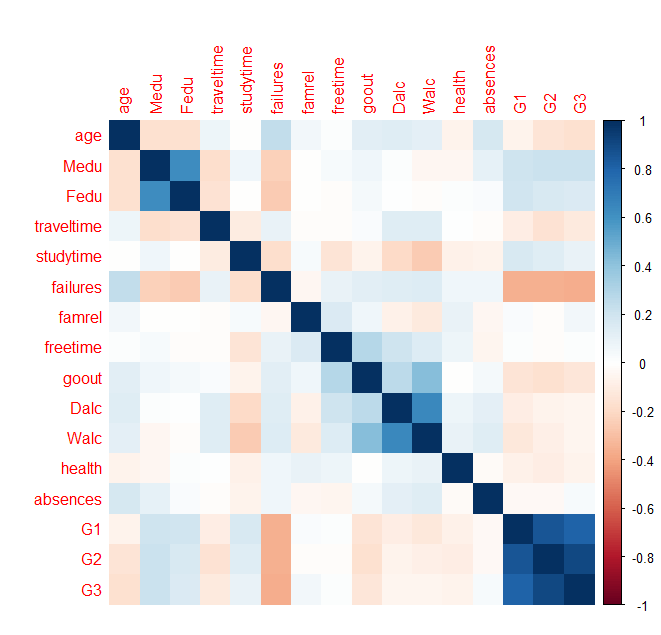


Let's start by using corrplot, the most common one. [Here's a really nice documentation page on the package.](https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html) I encourage you to play around with it.

-Add the following to update LinerReg.R file



-Save & Run

-o/p 

Cleary we have very high correlation between G1, G2, and G3 which makes sense since those are grades:

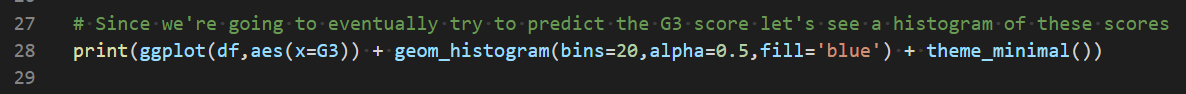
* 31 G1 - first period grade (numeric: from 0 to 20)
* 31 G2 - second period grade (numeric: from 0 to 20)
* 32 G3 - final grade (numeric: from 0 to 20, output target)

Meaning good students do well each period, and poor students do poorly each period, etc. Also, a high G1,G2, or G3 value has a negative correlation with failure (number of past class failures).

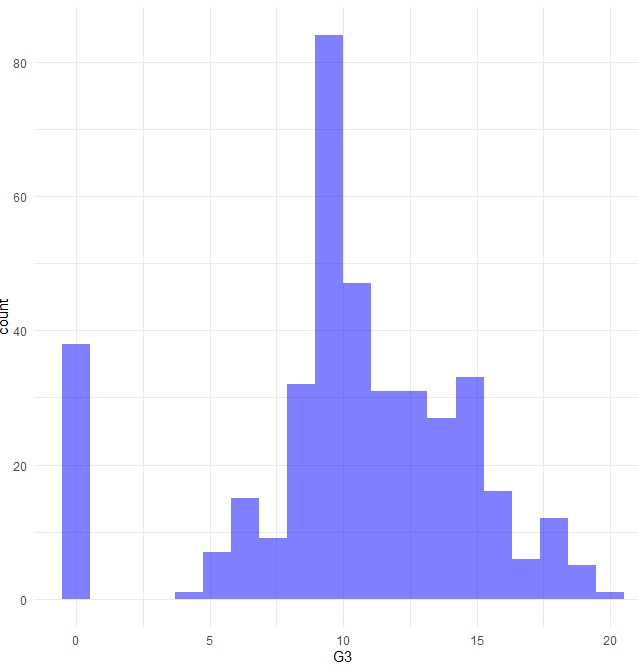
Also, Mother and Father education levels are correlated, which also makes sense.

**Since we're going to eventually try to predict the G3 score let's see a histogram of these scores**

-Add the following to update LinerReg.R file



**-Save & Run**

**o/p:** 

Looks like quite a few students get a zero. This is a good place to ask questions, like are students missing the test? Also why is the mean occurence so high? Is this test curved?

Let's continue by building a model

**Building a Model**

General Form:

The general model of building a linear regression model in R looks like this:

model <- lm(y ~ x1 + x2,data)

or to use all the features in your data

model <- lm(y ~. , data) # Uses all features

**Train and Test Data**

We'll need to split our data into a training set and a testing set in order to test our accuracy. We can do this easily using the caTools library

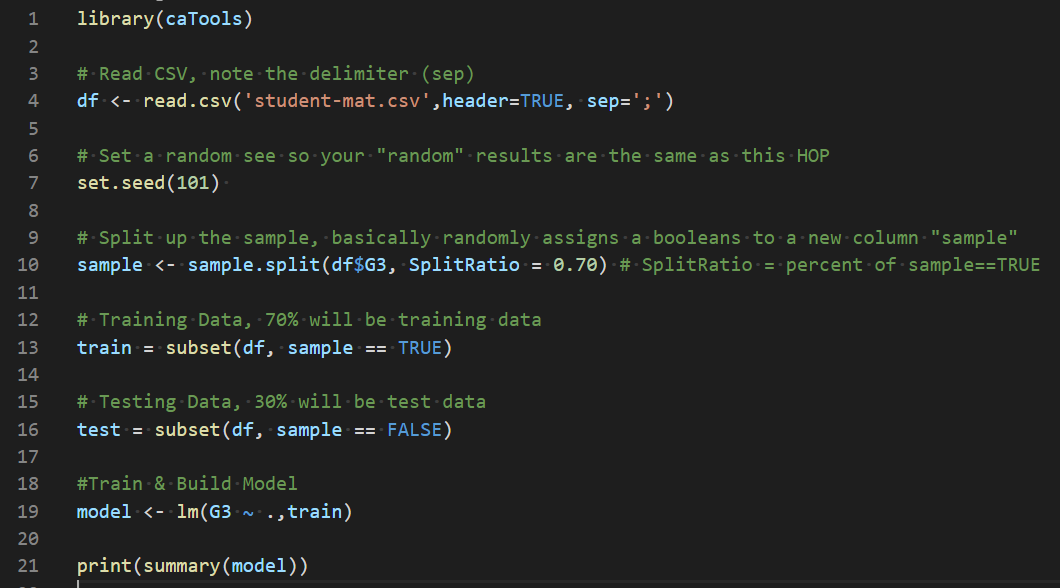
-Type the following in R console to install the caTools package

install.packages('caTools',repos = 'http://cran.us.r-project.org')

**You should be in:**

* **online student:** CS251 \_Fall\_2020/**ON**/FirstnameLastname
* **onsite student:** CS2511 \_ Fall \_2020/**IN**/FirstnameLastname
* **In Module7** project folder, create new file LinerReg2.R

-Type the following in LinerReg2.R

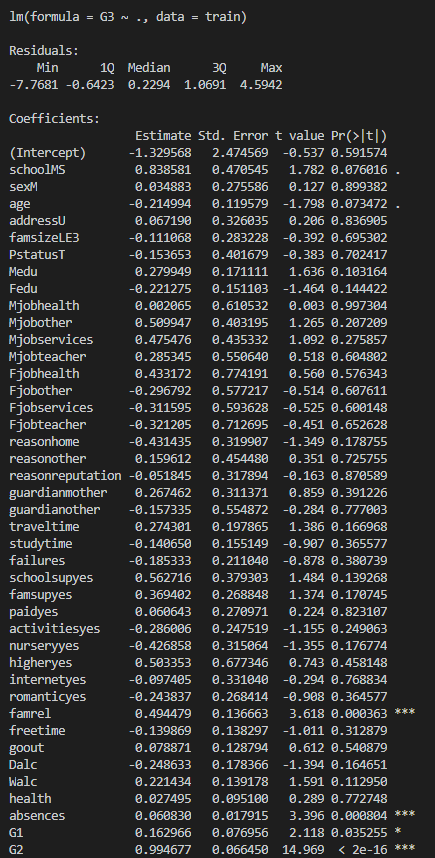


-Code Explanation:

- We wanted to split our data into training set and testing set and the reason for this is that we’re going to train our model on the training set then we’re going to use the test set to have prediction from the train model in order to do this:

1. We installed the caTools package, and this package makes it really easy to randomly split up our data into a training set and a testing set
2. Then we set a seed(101) and the reason we are setting a seed is that since the splits are going to be random, I want you to be able to follow along and have the same results as this HOP, so the seed make sure that the random number that I have on the HOP will be the same as the random numbers of yours.
3. To split the dataset, we created a variable called sample, then we called the function sample.split and what this does it splits the data from a vector into two sets from a predefined ratio and then what you do is you take your data frame and pass in a column of your data frame, so we passed the column that we are trying to predict which is G3. Then we put in the split ratio and the split ratio is going to be the percentage of the stadia that we want to use for training.
4. We used split ratio of 0.7 which means we will use 70 percent of this as training data and 30 percent of it as testing data
5. To build a linear regression model in R, the first step is we’re going to call our model then we used lm which stands for linear model function. Then, we pass in that feature column that we’re trying to predict which is G3 then the tilde sign and we can add up the features that we want to use in order to make the prediction. In our case we used lm(G3 ~ . ,train) we use . because we want to use all the features in our data frame

-Save & Run

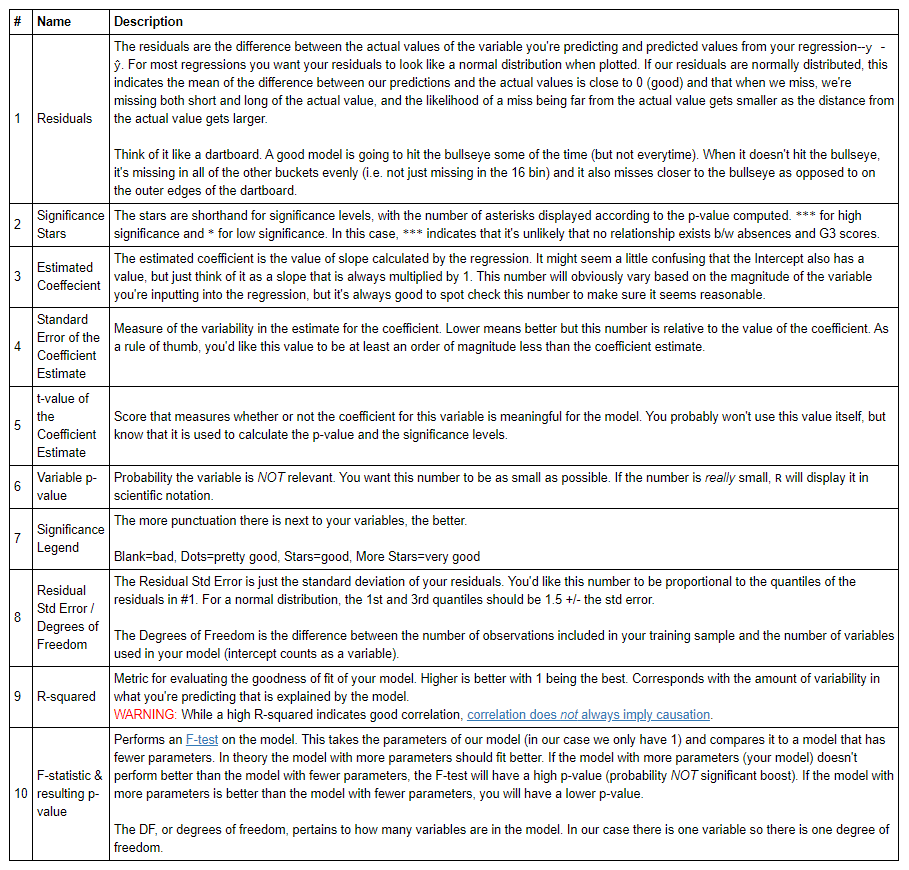
o/p 

output explanation:

The output is the summary of our model. The summary gives us all of this useful mathematical information.

Understanding the model interpretation requires a general understanding of statistics, check out Wikipedia for overviews on some of these topics, as well as the ISLR book. **Here's a quick guide on understanding the model summary:**

**Please read the below table to understand the o/p of the model.**

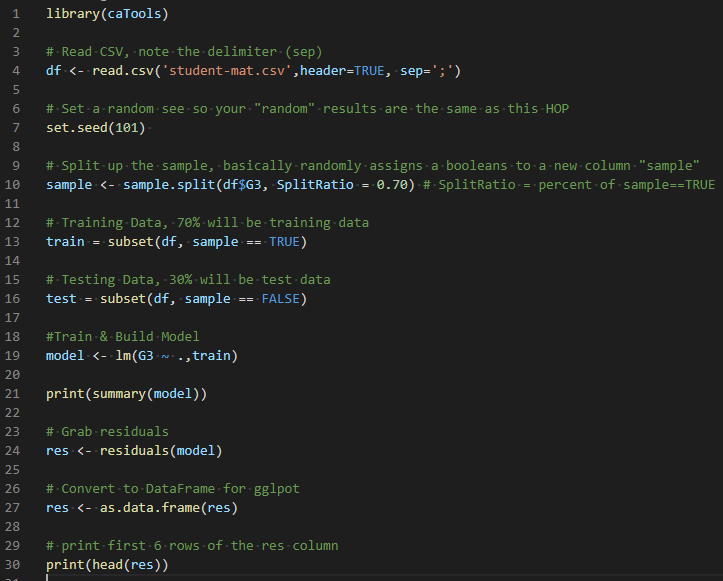


Looks like Absences, G1, and G2 scores are good predictors. With age and activities also possibly contributing to a good model.

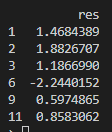
**Visualize our Model**

We can visualize our linear regression model by plotting out the residuals, the residuals are basically a measure of how off we are for each point in the plot versus our model (the error).

-Add the following to update LinerReg2.R

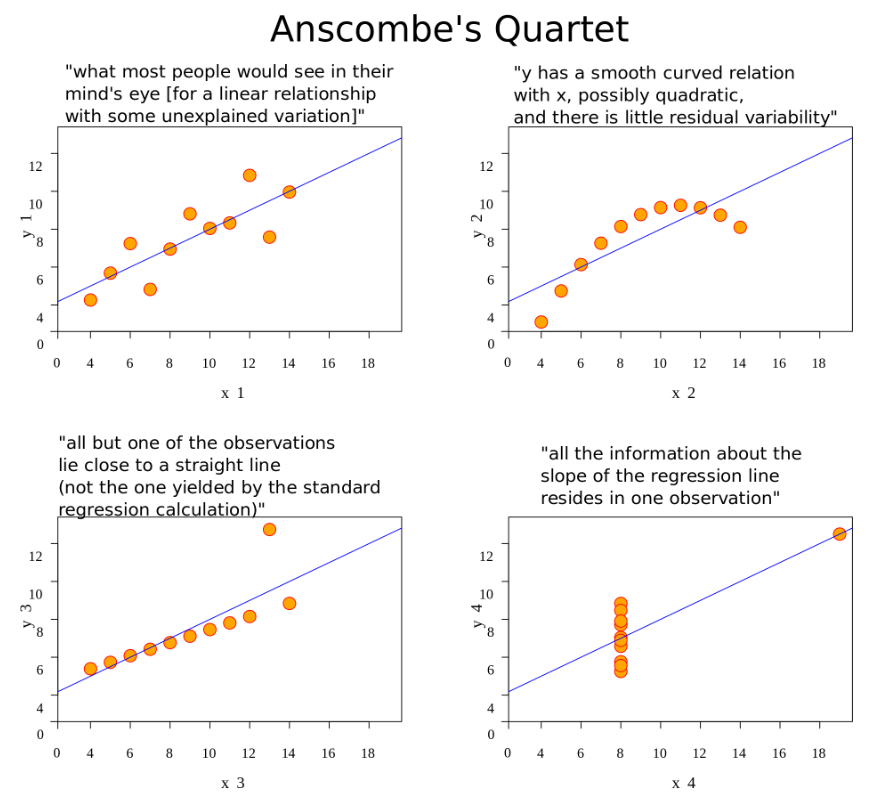


-Save & Run

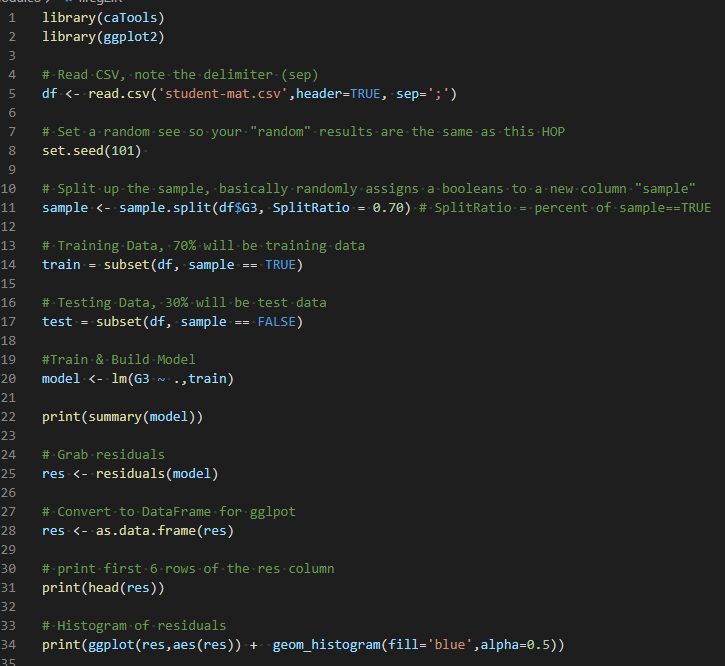
o/p: 

## Why Plot Residuals?

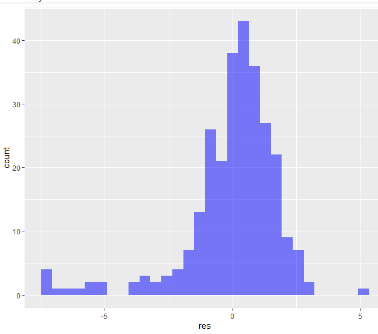
We want a histogram of our residuals to be normally distributed, something with a strong bimodal distribution may be a warning that our data was not a good fit for lienar regression. However, this can also be hidden from out model. A famous example is [Anscombe's Quartet](https://en.wikipedia.org/wiki/Anscombe%27s_quartet)



**-**Add the following to update LinerReg2.R

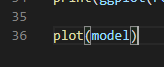


-Save & Run

o/p: 

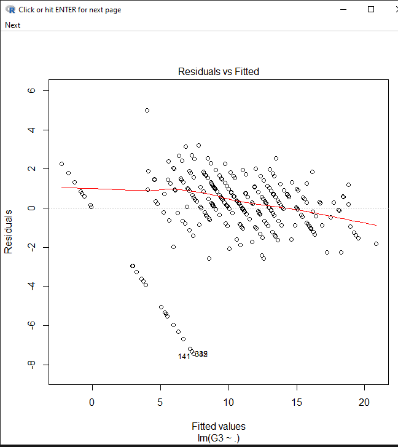
Looks like there are some suspicious residual values that have a value less than -5. We can further explore this by just calling plot on our model. What these plots represent is outside the course of this lecture, but it's covered in ISLR, as well as the Wikipedia page on [Regression Validation](https://en.wikipedia.org/wiki/Regression_validation).

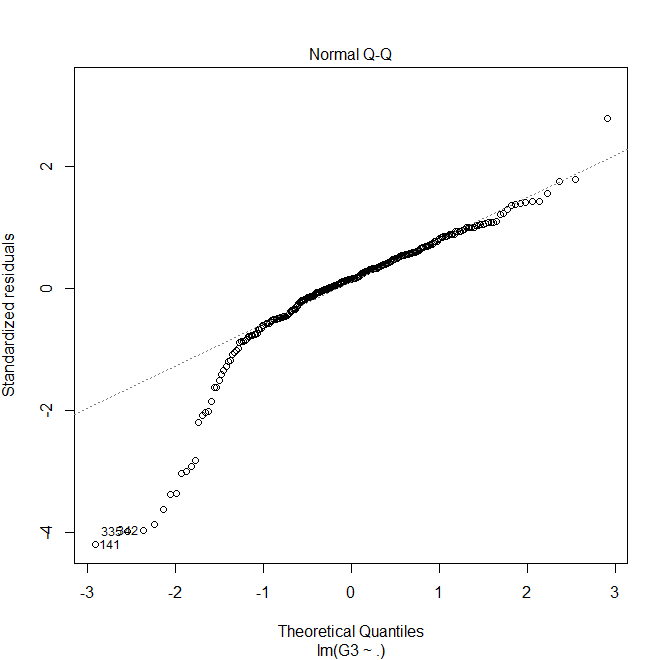
-Add the following line to update LinerReg2.R

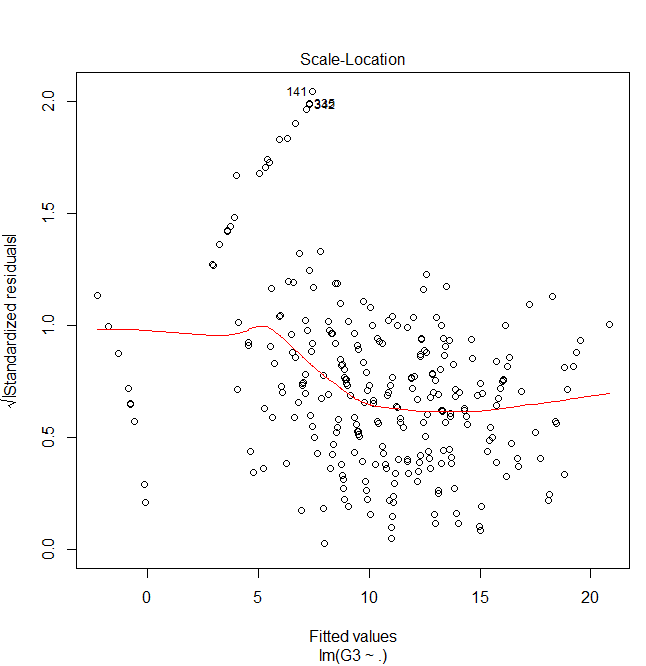


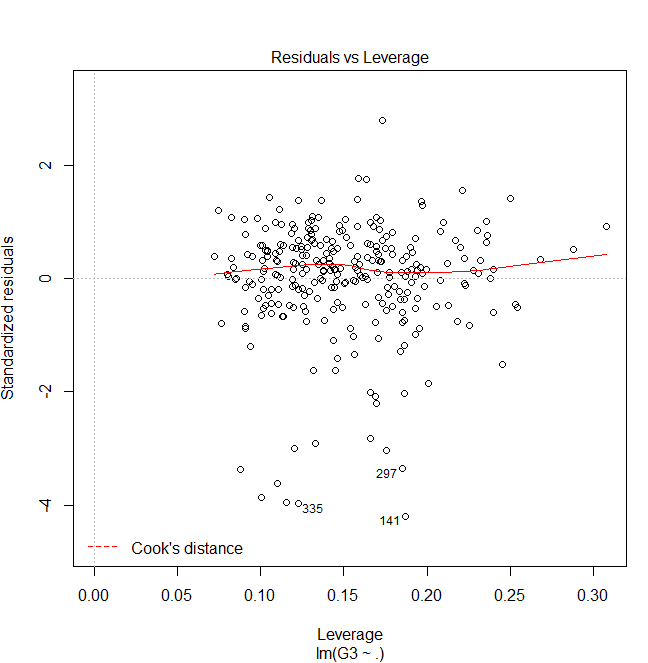
-Save & Run

o/p: make sure to click enter on the graph to see the next









Basically, after looking at these plots what you will realize is that our model (behaving as a continuous line, predicted students would get negative scores on their test! Let's make these all zeros when running our results against our predictions.

**Predictions**

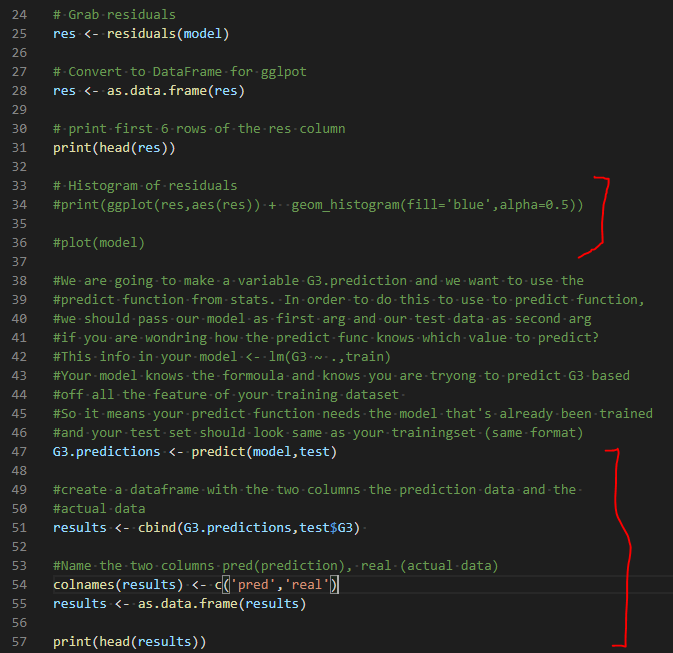
Let's test our model by predicting on our testing set:

In order to see how well our models performing what we want to do is not just a summary of model but use predict to grab prediction off our testing set, and this is basically the whole reason we were using the test train split. We want to train our model on the training data and then test the model for new prediction using the test dataset.

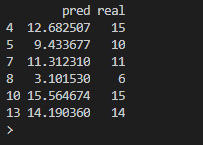
It doesn’t really make sense to predict the new data points off the training set because the model already been trained and has seen all those training points. We want to try to predict new values based off the test set which is something that the models never seen before.

So, let’s go ahead and started with predictions.

**-**Add the following to update LinerReg2.R, and make sure to comment plot(model)



-Save & Run

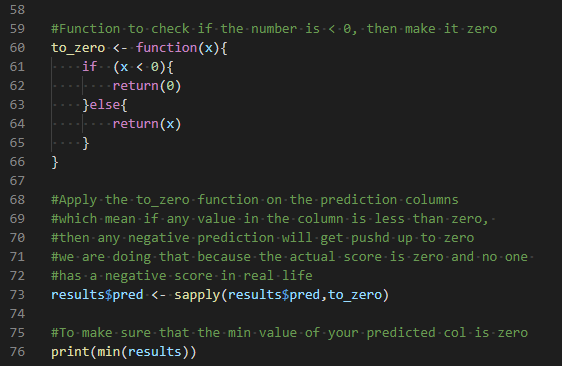
-o/p: 

Output explanation:

* What we have done is created the data frame for each of the test datapoints
* Note that these numbers are in innumerical order because this was a random grab of the data frame that original data and that’s why it’s not 1 2 3 4, and this is acutally datapoint number 4 , 5, 7, ..

We notice in the residual a large amount of larger negative values and the reason for that was because our model was predicting negative final score test values or negative final period values. Now the lowest value you could get is zero, that means we are going to do is take care of those negative predictions.

**-**Add the following to update LinerReg2.R

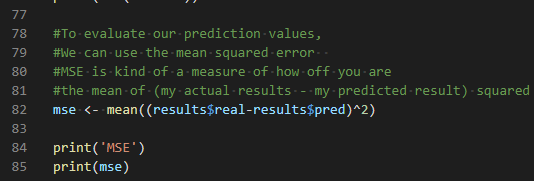


-Save & Run

-You will find that the minimum value of the result dataset is equal to zero

There's lots of ways to evaluate the prediction values, for example the MSE (mean squared error)

**-**Add the following to update LinerReg2.R

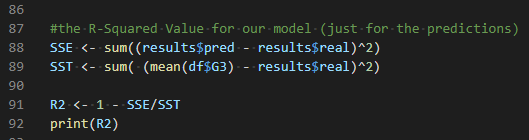


-Save & Run

**Or**

just the R-Squared Value for our model (just for the predictions)

**-**Add the following to update LinerReg2.R



-Save & Run

-o/p of R2 will be 

0.8 is not so bad for the test data it means we’re explaining about 80 percent variance on the test data

**Push your work to GitHub**

**Make sure you are in**

Onsite students: CS251\_ Fall \_2020/**IN**/FirstnameLastname

Online students: CS251\_ Fall \_2020/**ON**/FirstnameLastname

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hit the **control + ~** key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 7”

>>> git push origin YOUR\_BRANCH\_NAME

Note: you should change the YOUR\_BRANCH\_NAME to your own branch name. It should be firstname-lastname (e.g. maria-gracia).