Loan Aggregator with Liner Regression in R

In this project we will use Linear regression to predict the most optimal loan for our clients.

Project Scenario

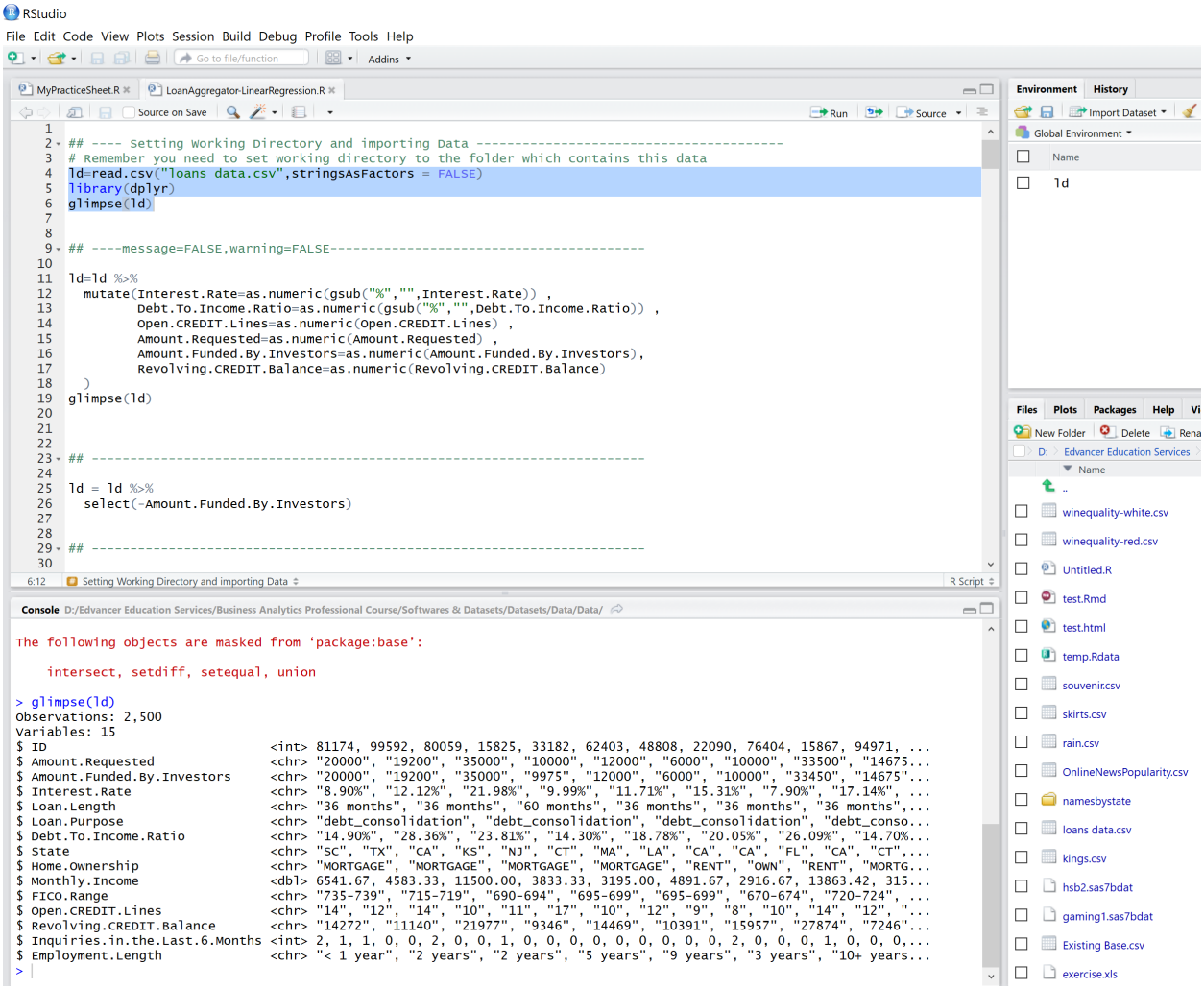
We are a loan aggregation organization. We receive applications seeking loan from different clients. Based on their information like income, etc., we determine which financial institution's loan would be the best for them in terms of interest rate. Usually this is done by people working at the institution but the process is not very efficient. We want to predict the interest rates that will be provided to an application based on its features so that we can narrow down our search to top 5 or so loans. This will make to process faster and consistent, thus bringing in more customers.

In this project, we will focus on the data of a specific financial institution and what kind of rates it will provide to an application.

Data Preparation & Manipulation

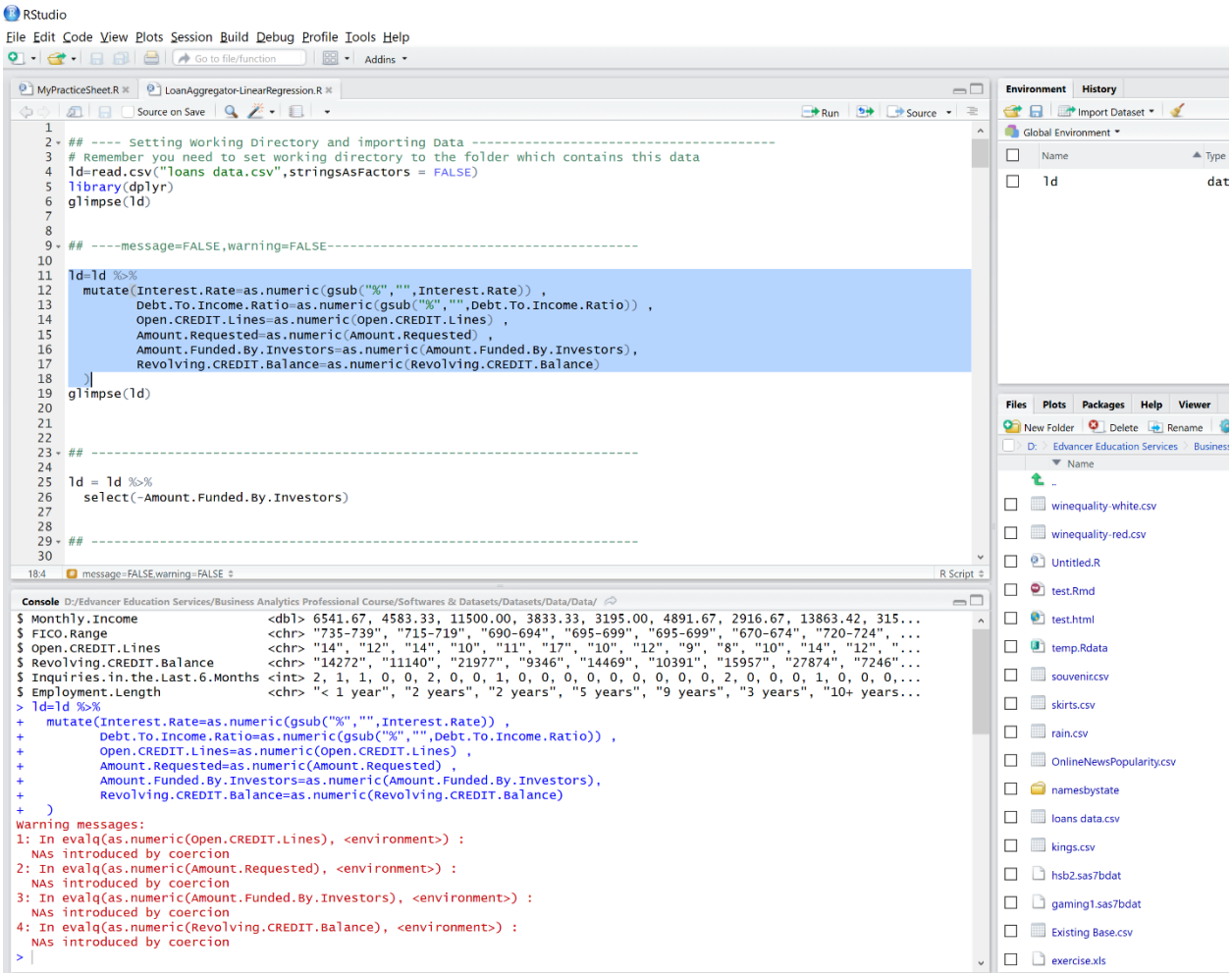
After setting the path and importing the dataset, we see that a lot of the attributes (columns) are characters.

We can also see that many columns like Amount.Requested, Amount.Funded, Interest.Rate, etc. are numbers. They have been converted to characters because some of their rows may have been "NA" because of lack of values or contain "%".

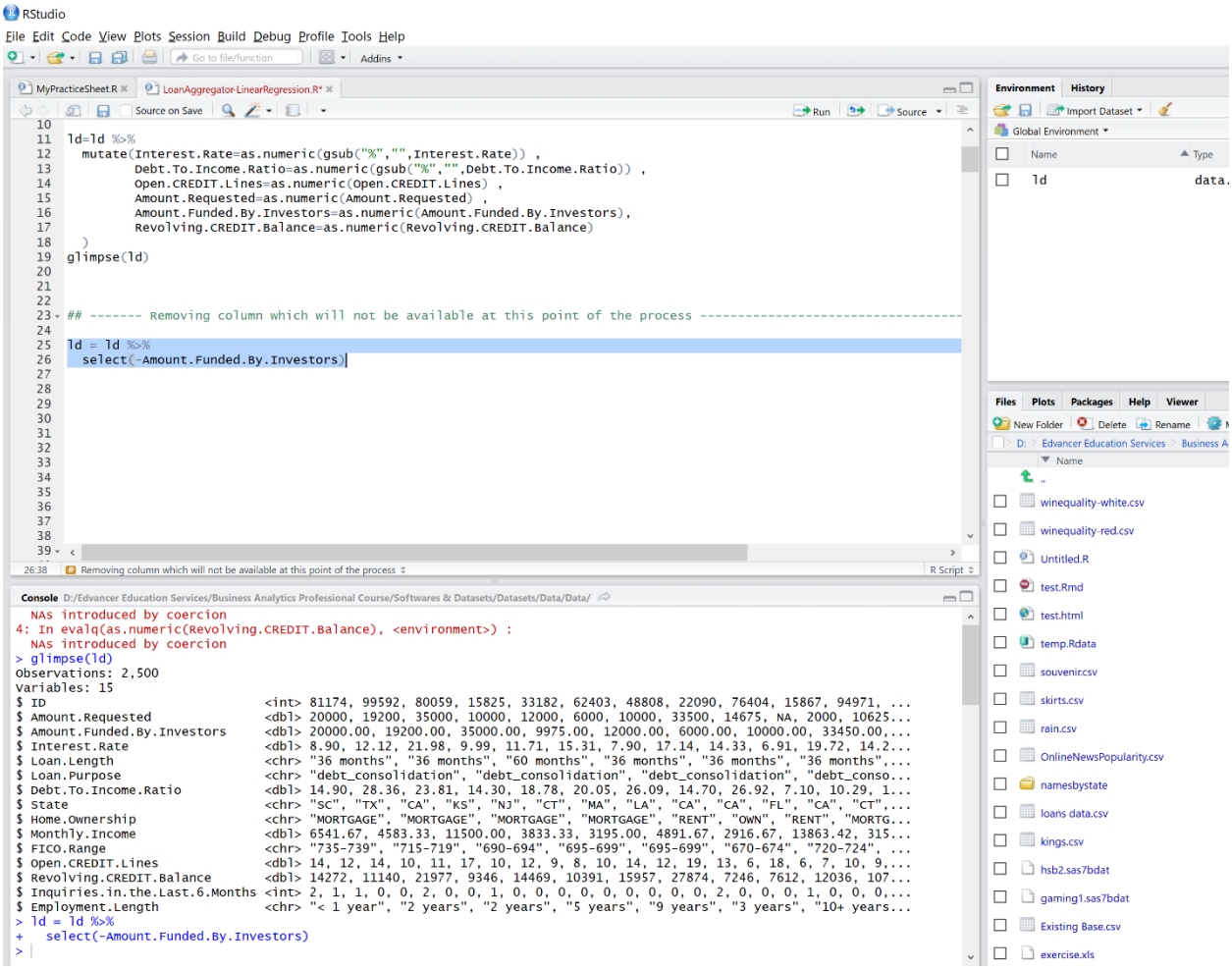


So, we will remove the "%" from the values of some of the columns like Interest Rate and Debt-to-Income ratio, also we will convert some of the character columns into numeric. We may get a warning because we are converting values that have "NA", but they can be ignored.

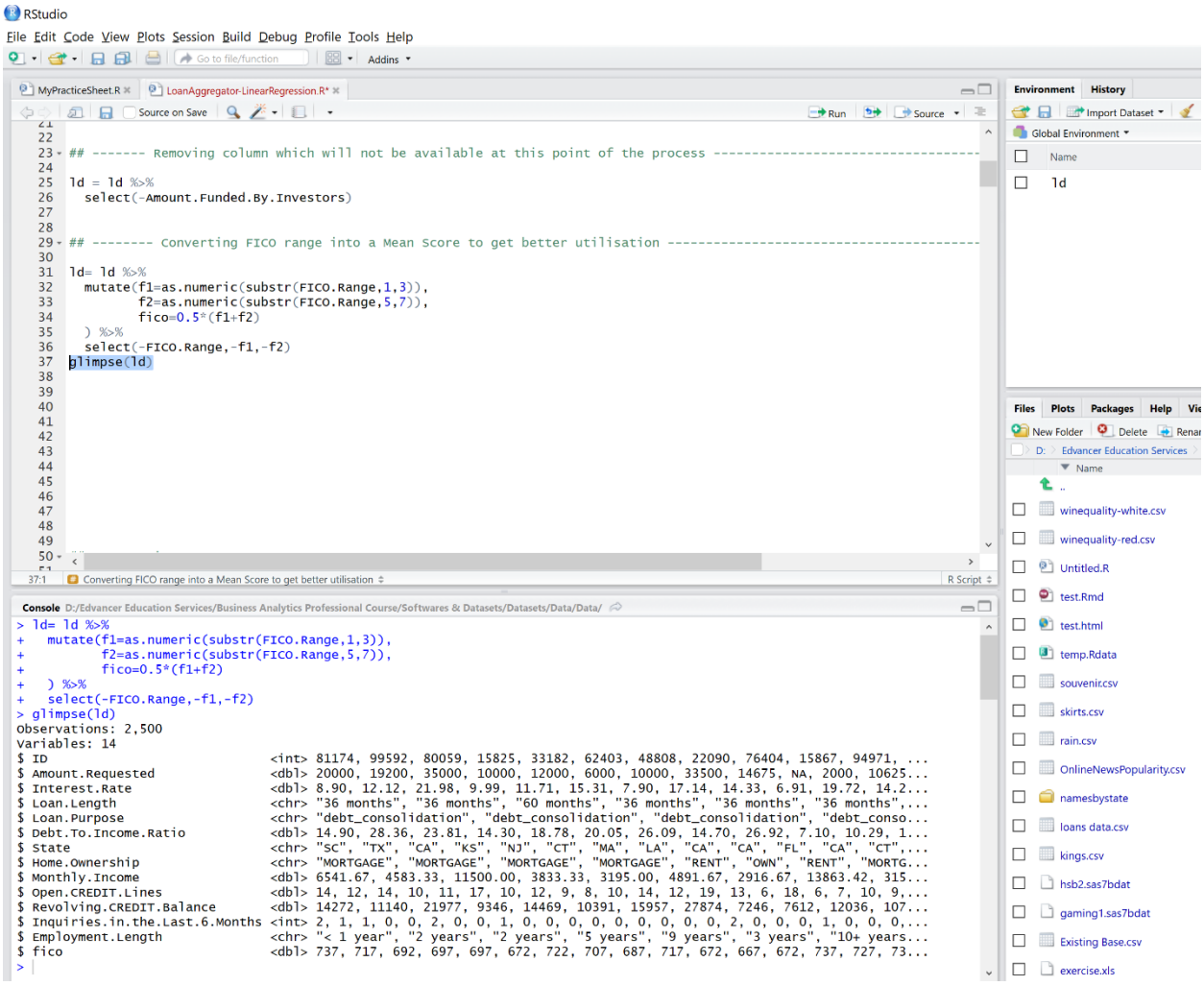
We can use glimpse to see that the rows we had targeted have become numbers (or double).



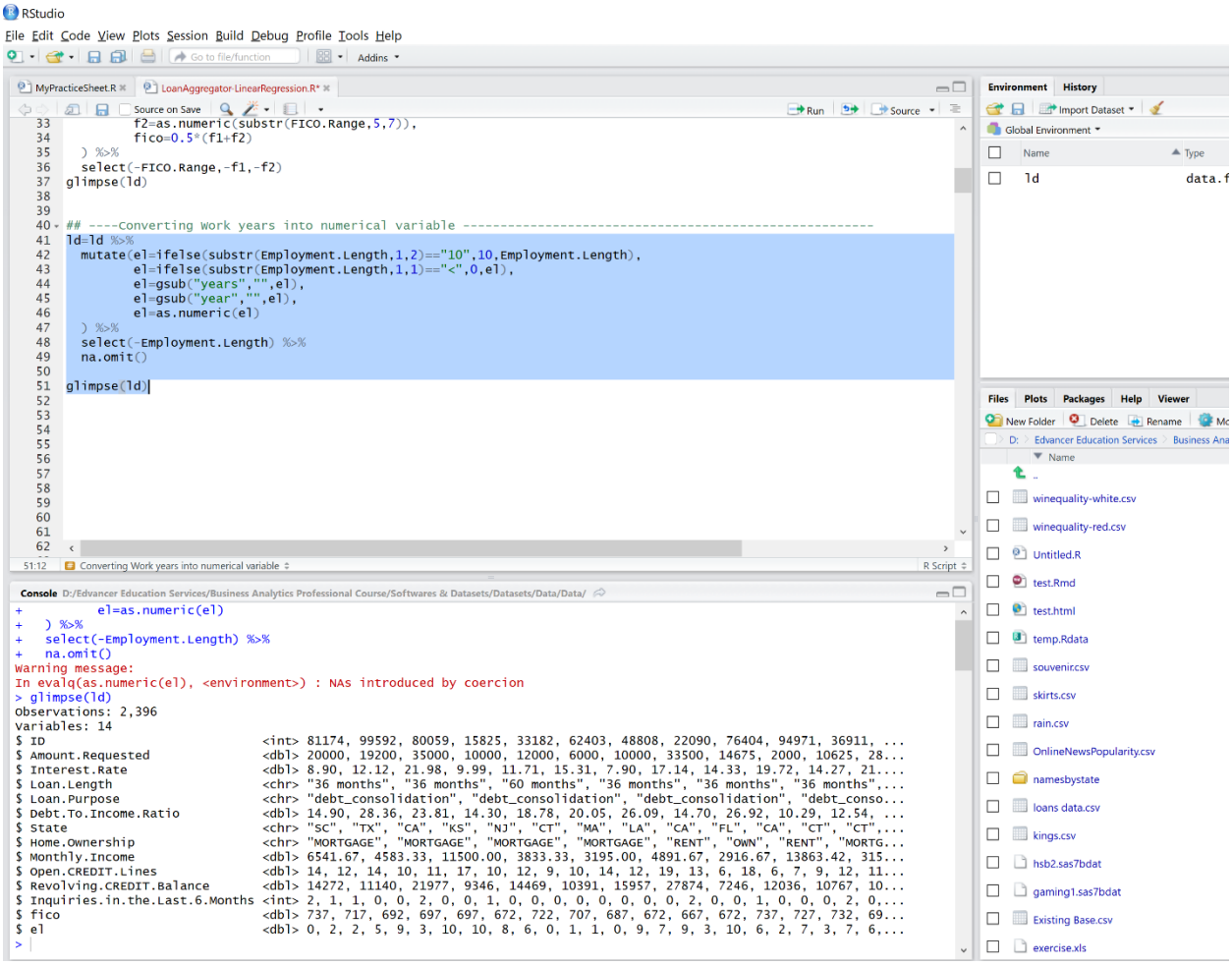
We have a column Amount Funded by Investors, but this is not a part of our process right now as we are assuming that the application is new and not historical data. So, we will remove it from our dataframe.



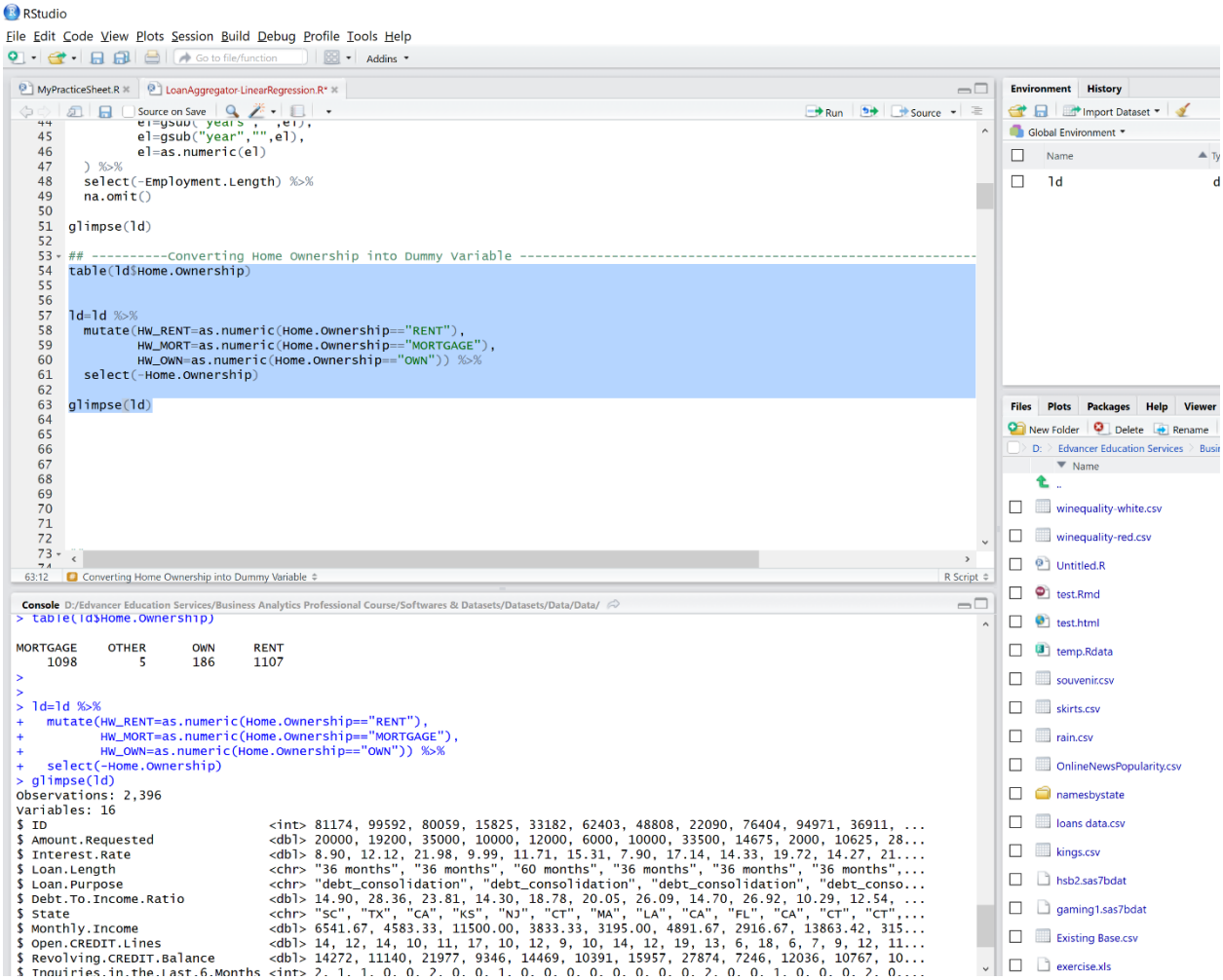
We have another column called FICO Range, which basically gives the range of Fico Score that an applicant has. This is a categorical data, but it has a numerical importance too i.e. higher range means better financial status of the client. So, we cannot completely ignore this aspect of the column too. The solution to this is to convert the range given for each applicant into a mean Fico score. After calculation, we can remove the intermediate and parent column.



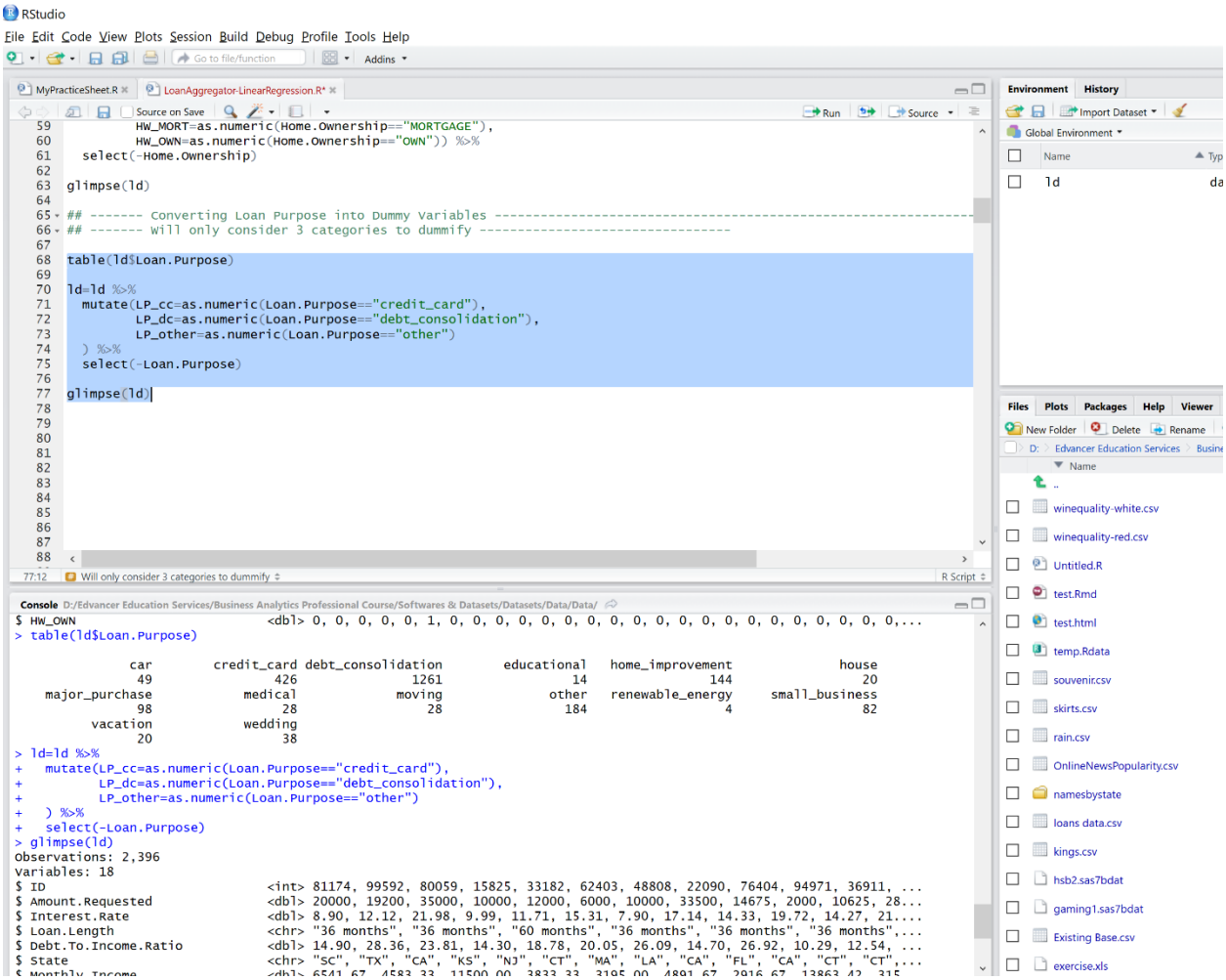
We have another column Employment Length, which contains number of years the applicant has been employed. It is a categorical variable, with columns like "<1 years" and "10+ years". We will convert this into a numerical column and convert the "<1 years" and "10+ years" columns into 0 and 10 respectively. At the same time we will omit the parent column and also remove rows with No Values.



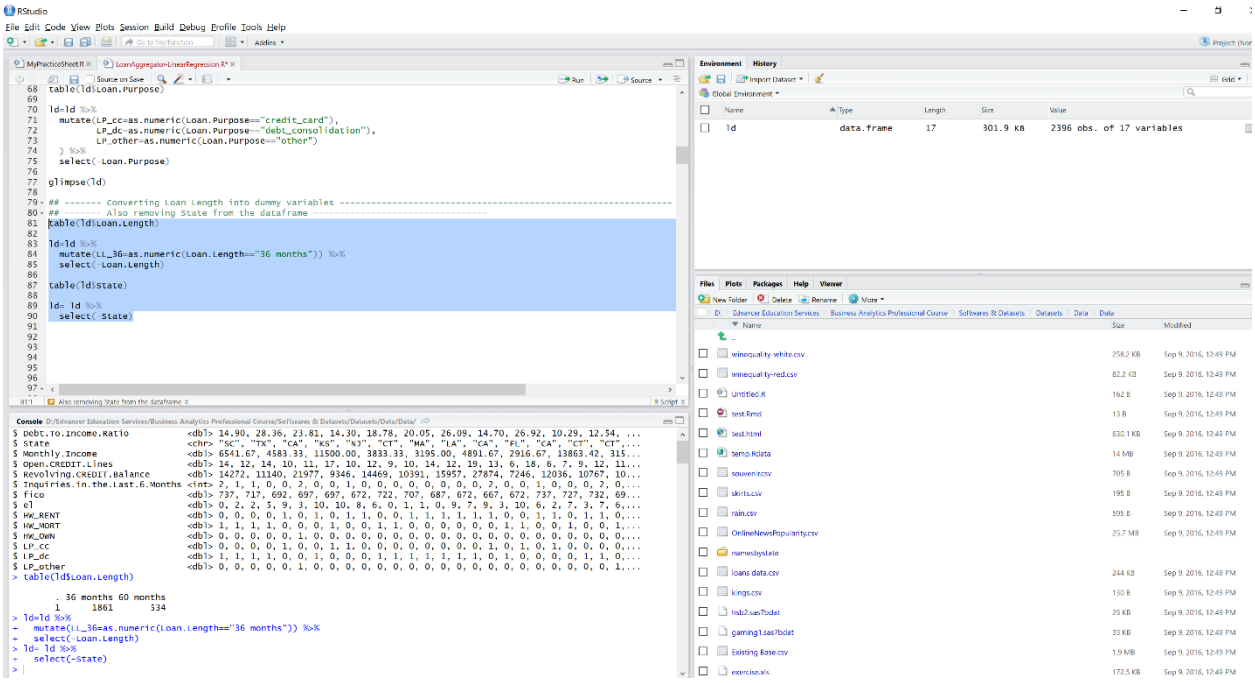
We have a categorical variable Home Ownership which has 3 categories - Mortgage, Rent, Own and Other. for these we will create dummy variables, where the value will be 1 for that category and 0 otherwise. We will not consider the Other category as it could be anything and has less impact than the others.



The Loan Purpose also contains 14 different categorical variables. We can convert them all to Dummy Variables but that would be unnecessary because most of these categories have very few rows. In our project, we will consider and dummify only 3 categories - Credit Card, Debt Consolidation and Other.

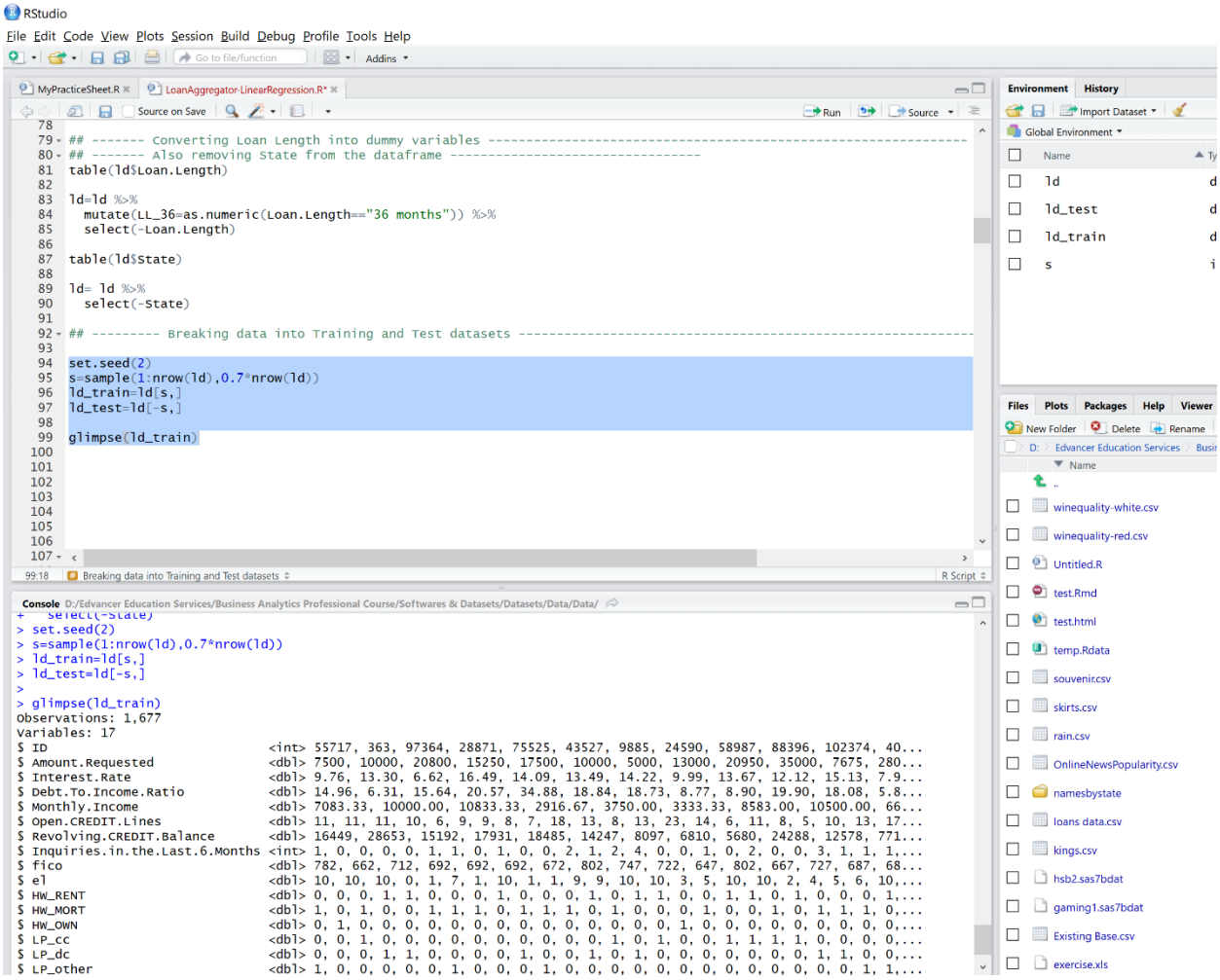


Loan Length has just 2 Categories - "36 months" and "60 months". There is another category but it is just a missing value. Because there are just 2 categories, we need just 1 dummy variable. The State variable does not contain important information so we will discard it in this project.



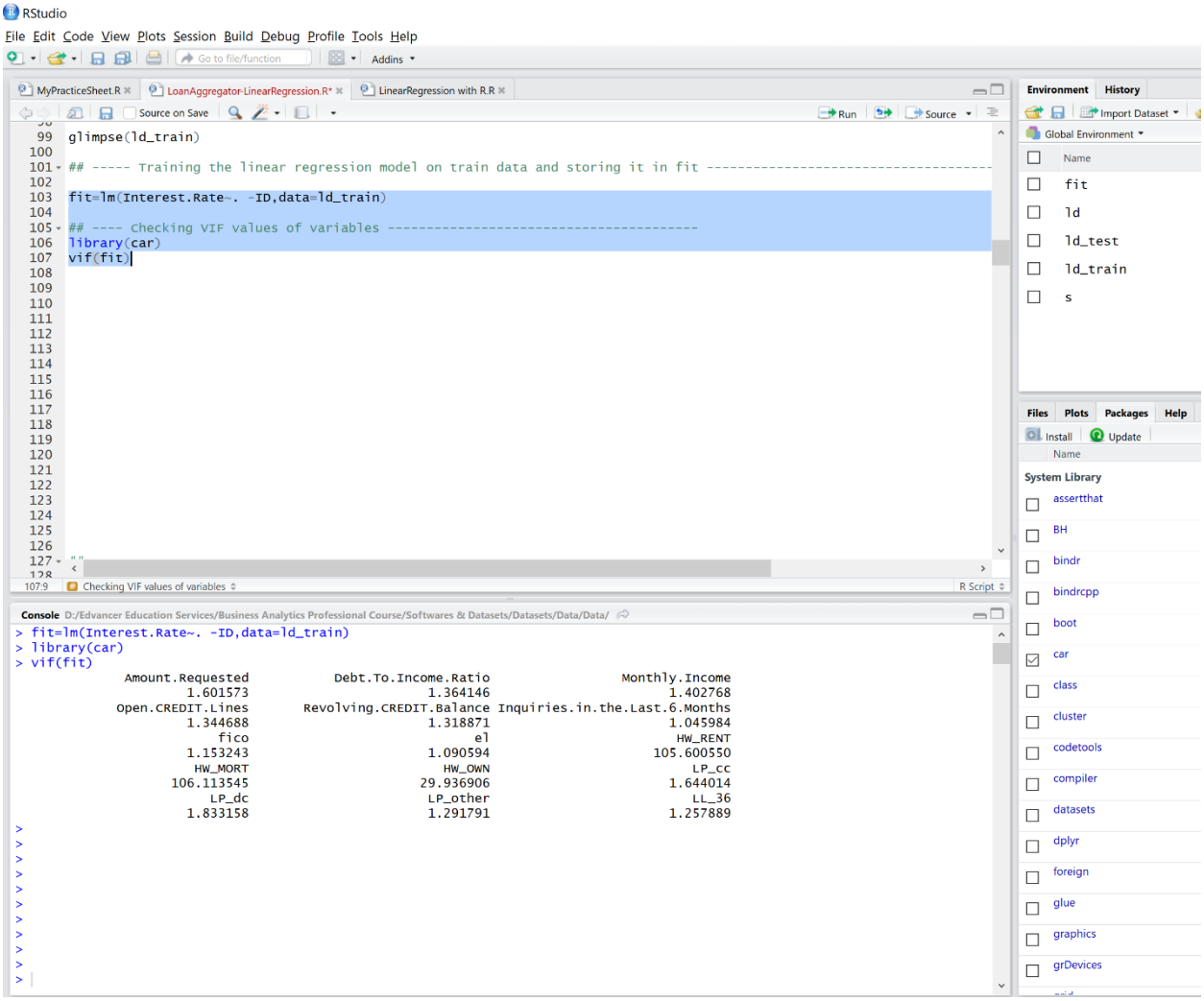
Training the Model

Before we start our analysis, we need to divide our dataset into Training and Testing parts. The division that we are using is 70:30.

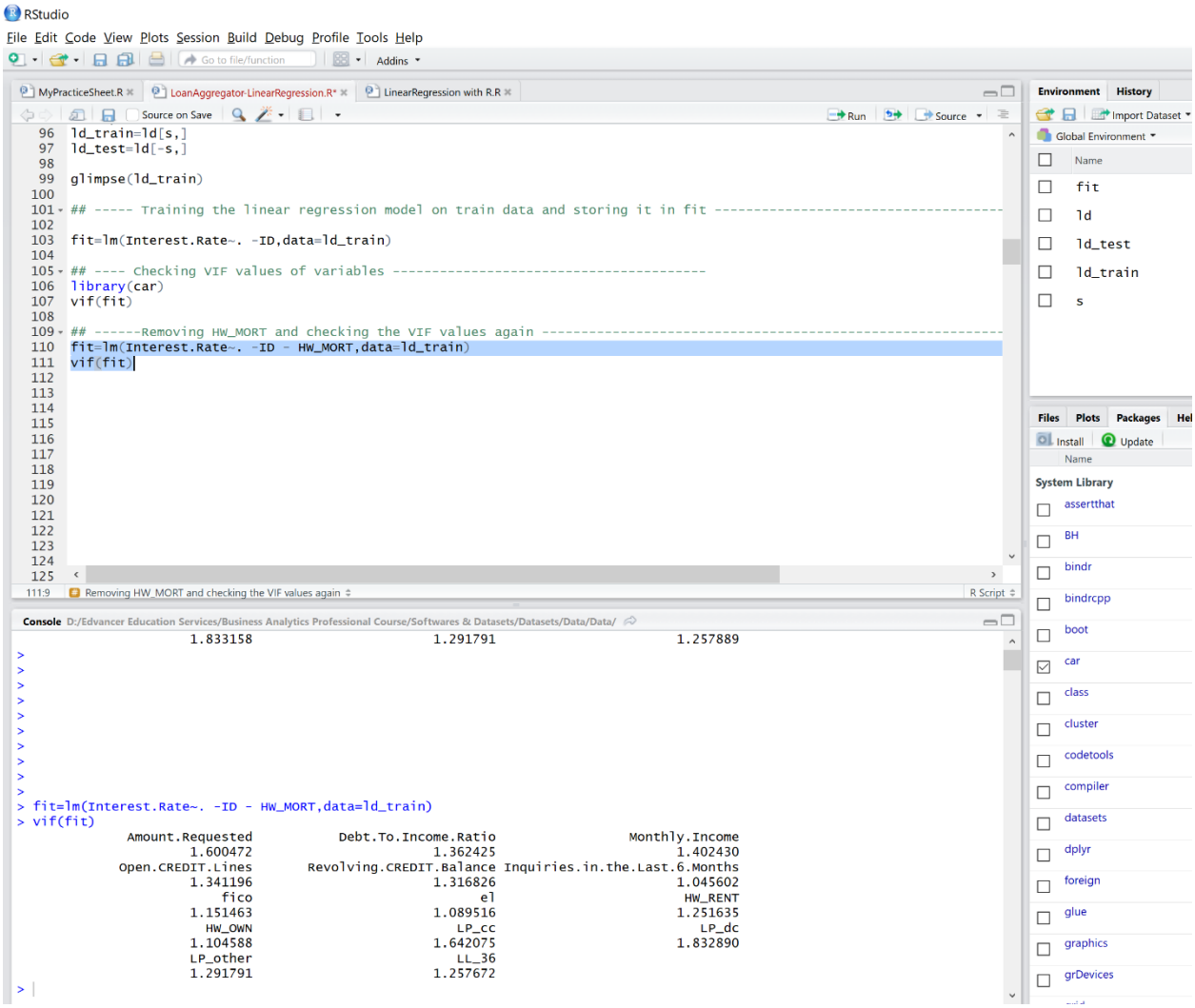


We use the linear regression function with a focus on predicting "Interest Rate" keeping all the predictors except ID, as it will not contribute in any way to our prediction of rates. We specify the dataset on which we have to perform logistic regression (ld_train) and store the output in the variable "fit".

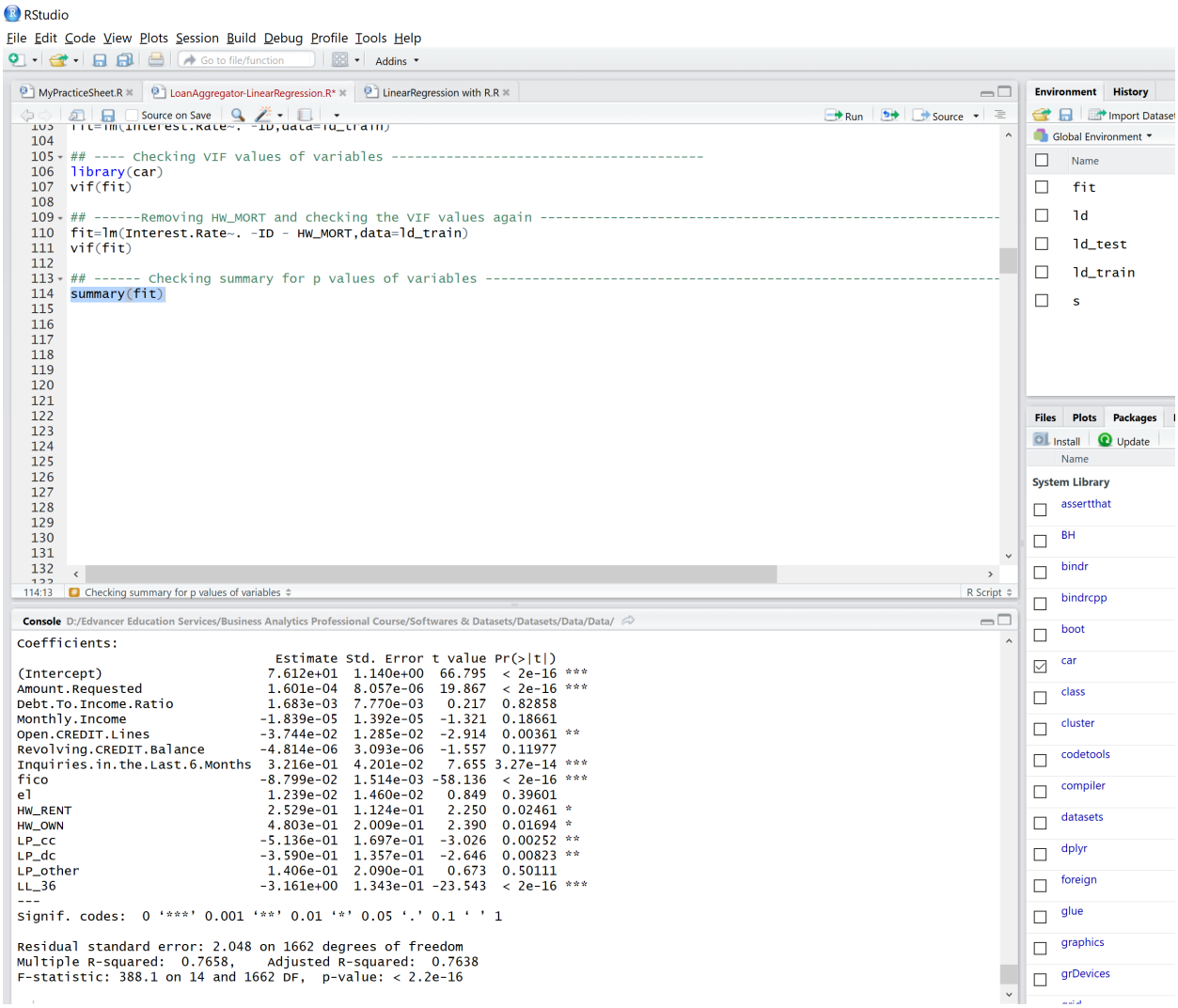
We can check the VIF values of all the predictors using the vif() function present in the car library and use it on the variable fit. (NOTE: You may have to install the car library from CRAN before you use VIF)



The variable HW_MORT has a very high VIF value (more than 5) so we will remove it from the next regression we perform. We then check the VIF again and see that they are well below the cautionary level of 5.



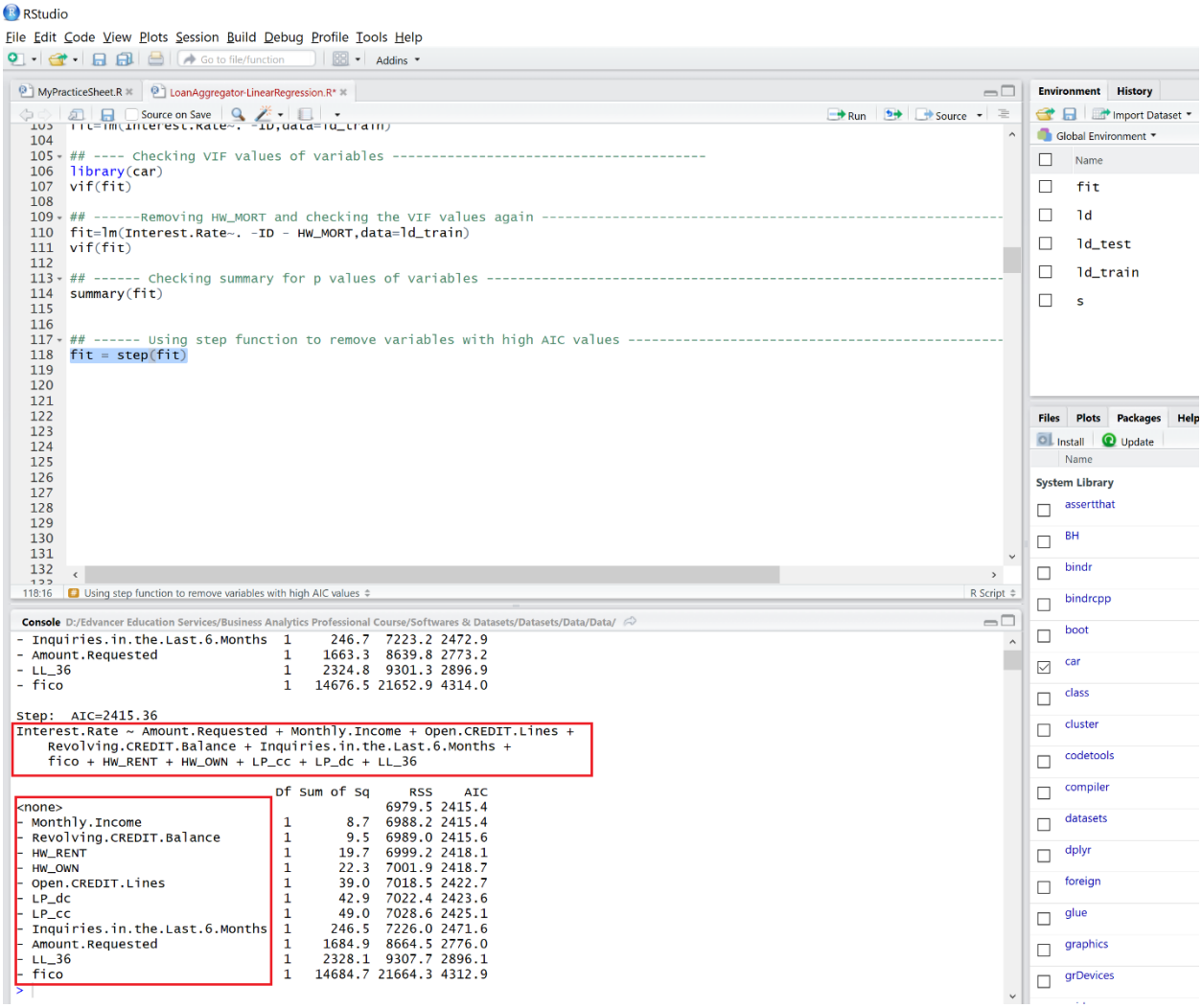
Next we check the p values of all the variables through the summary() function and remove those variables that have a p-value greater than 0.05 (Our acceptance level is still 0.95). This is easy when we have less variables but becomes tedious when we have hundreds of variables to choose from.



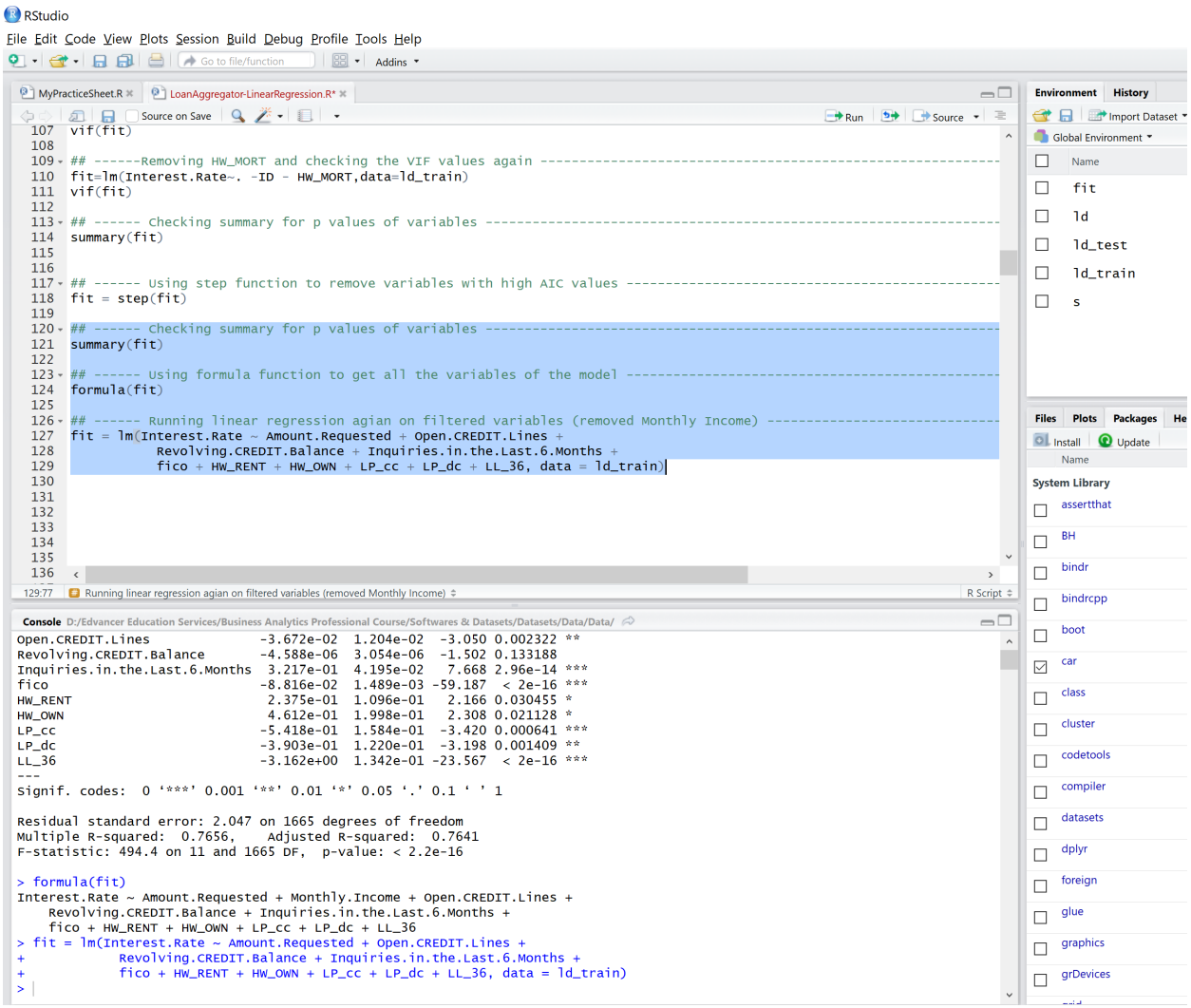
What makes this step automated is the function step(), which checks on the AIC score of each variable.

AIC = <Loss function> + 2 * <No. of variables>

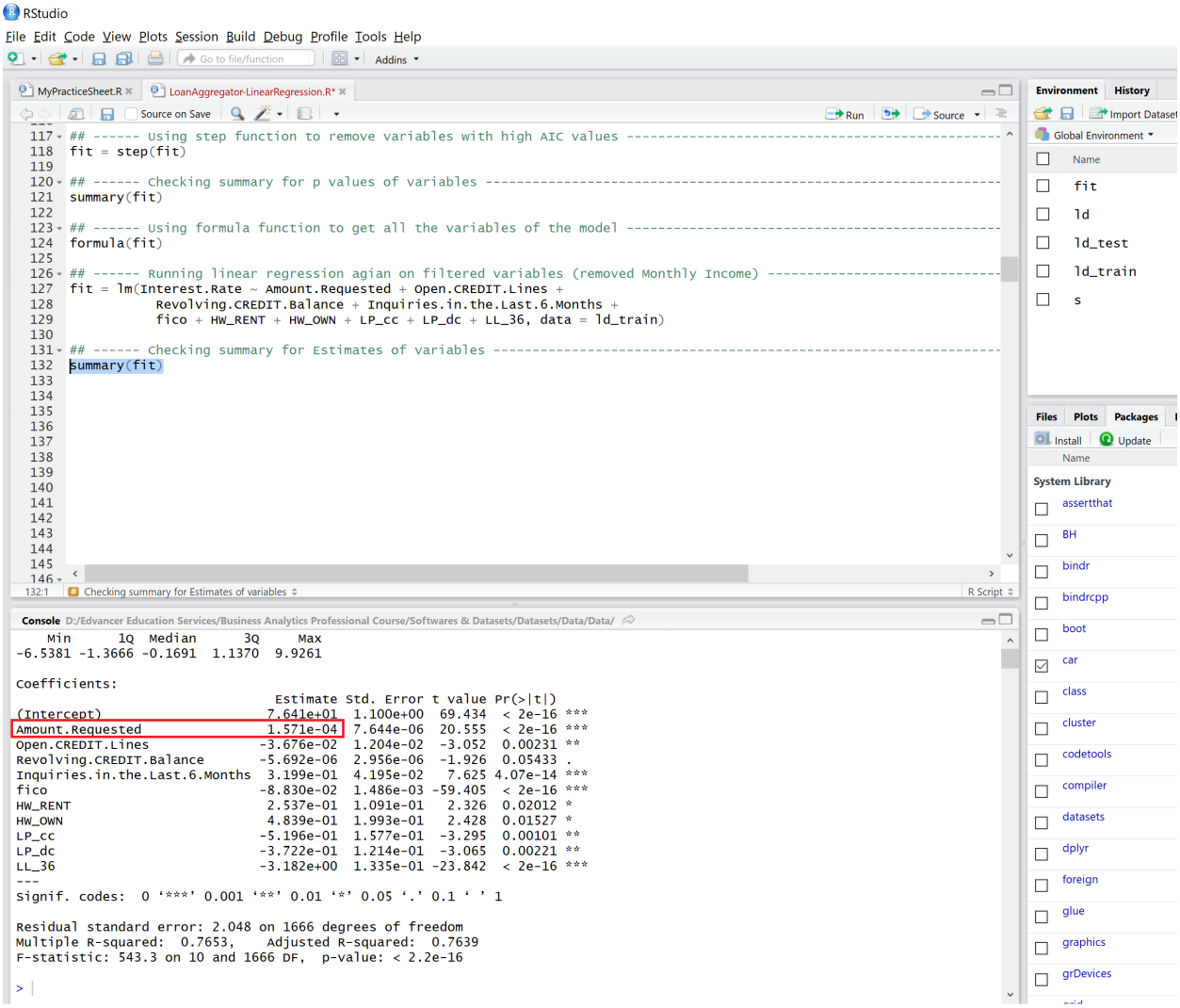
The Loss function we use here is Error Sum of Squares (SSE). Higher number of variables will lead to explanation of variance and thus decrease in Loss Function. In a nutshell, lower AIC value variables will be kept and higher ones will be discarded. Once all the variables have been dropped, we can check the summary again and drop the variables with higher p-values.



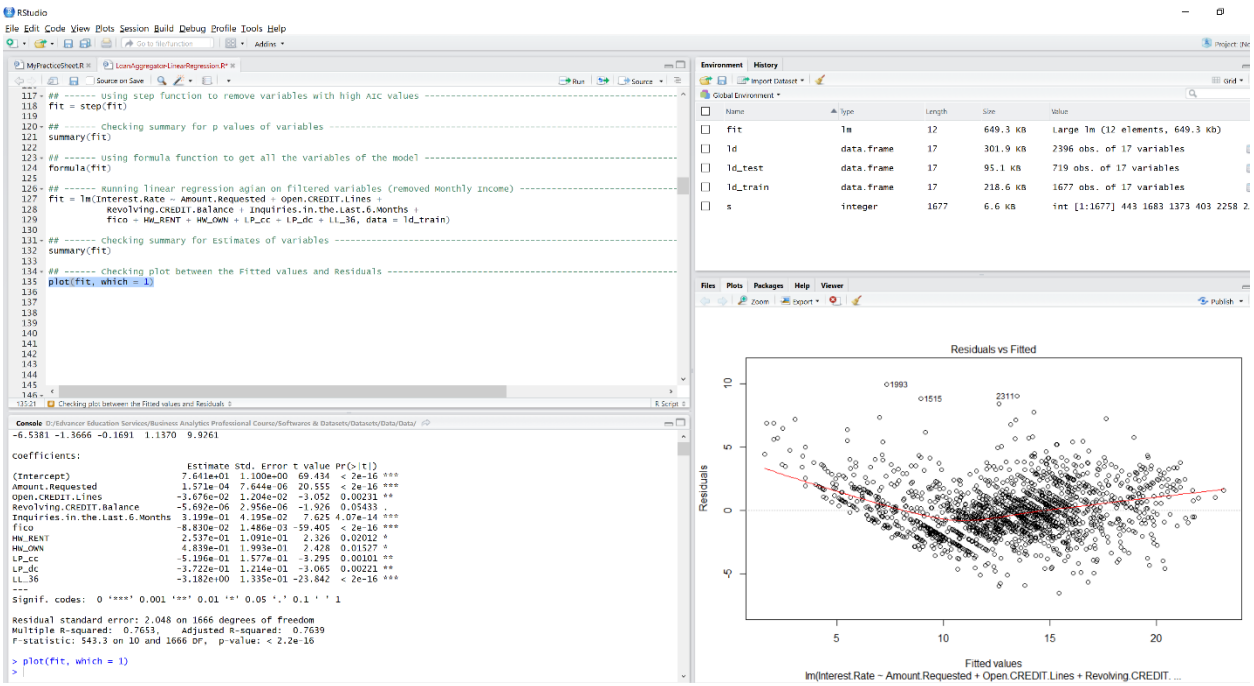
Rather than writing variables manually in the linear regression function, we can use the formula() function to get the dependent and predictor variables. We can copy paste these variables to the linear regression function and then remove the variables with high p values. Here, we will remove the Monthly Income variable, because it has a high p-value.



Upon checking the summary again, we can see a column called estimate, which shows the kind of correlation that variable has with the dependent variable i.e. Interest rate. Amount Requested is positively correlated with Interest Rate: higher the amount requested, higher will be the rate. These are basically the coefficients (beta). So, can we say that 1 coefficient is more valuable than the other based on its value? The answer is NO, and that is because we must take into account the size of the values in the variables as well. Amount Requested has values in thousands whereas Open Credit Lines is at the most in double-digits. We need to put this into perspective and scale up or down the importance of each variable and standardize before comparing coefficients.

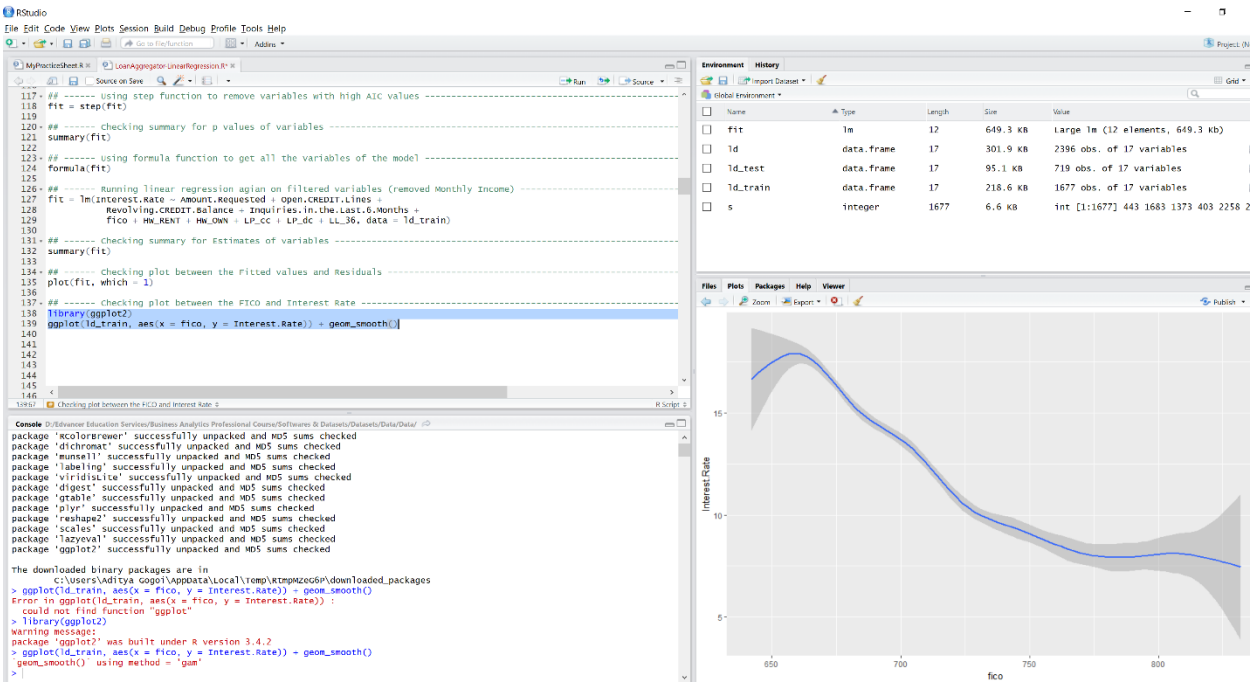


If we use plot() function to check the fitted values and their residuals, we see a non-linear plot. This shows us that one (or more) of the variables are not linearly correlated with Interest Rate. (Check “Residuals VS Fitted plot.png”)

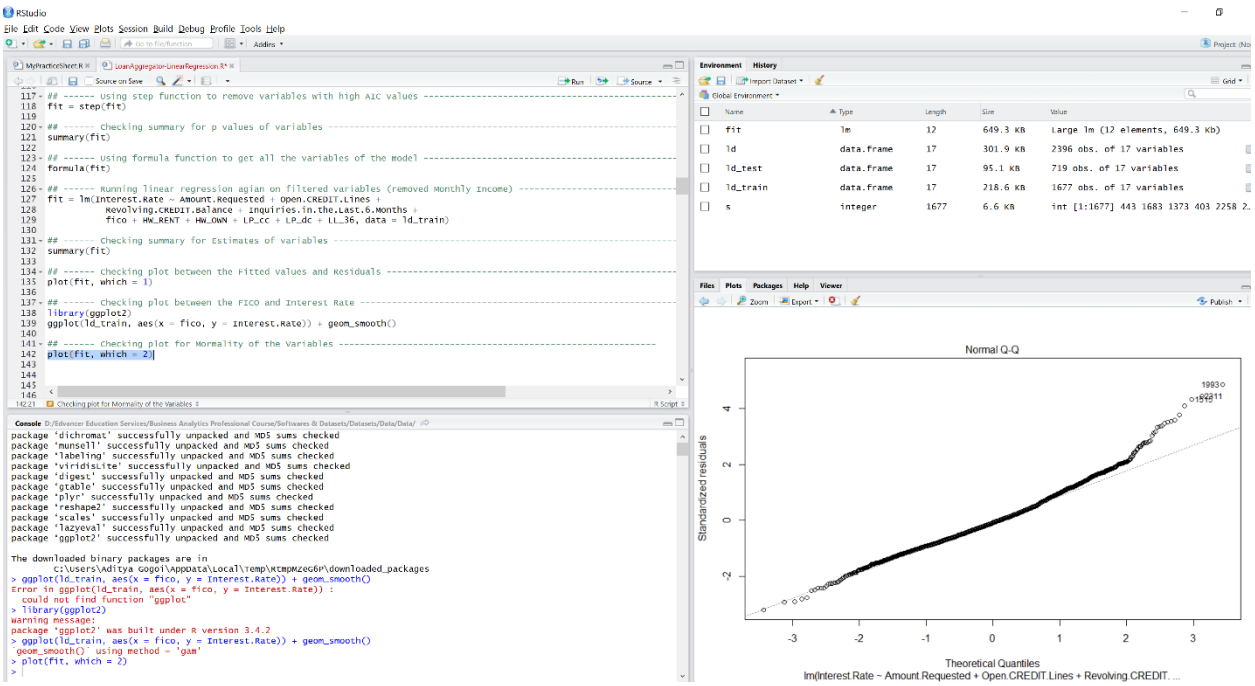


Upon going through the Smooth Line Scatter plot of FICO score VS Interest Rate, we see that, barring the extremes, their relationship is not Linear. Their relationship appears to be almost square like. (Check “Fico VS Interest Rate.png”)

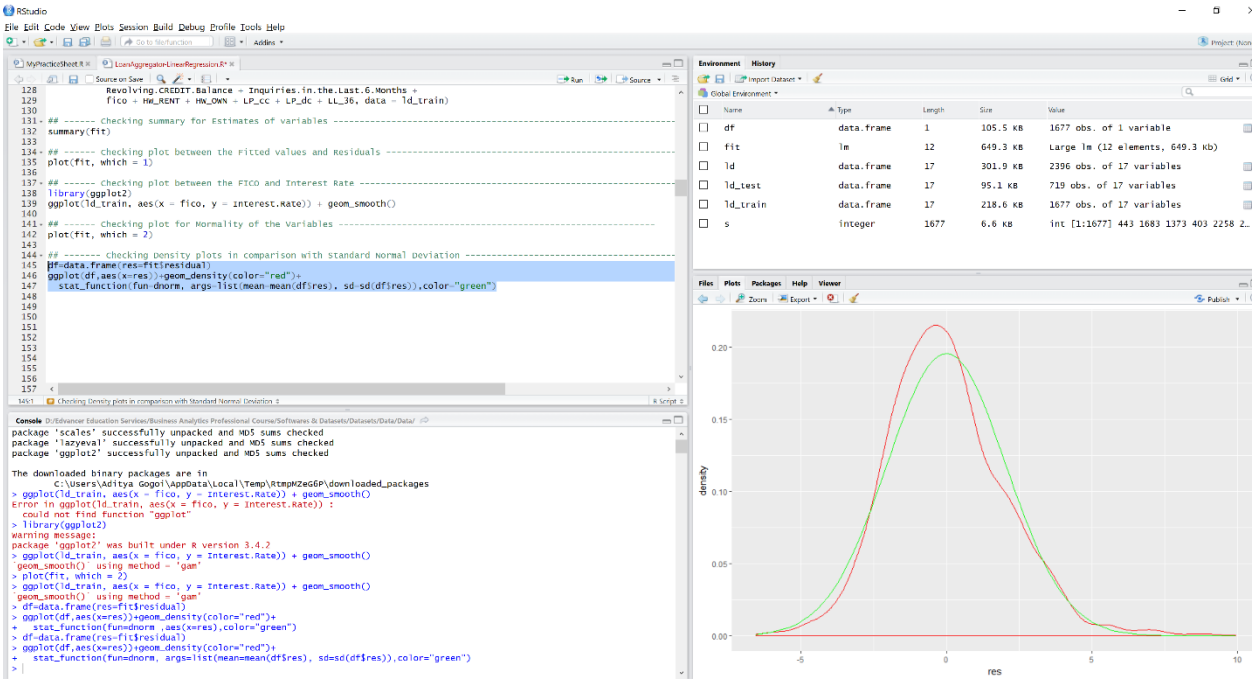
One way out of this would be to convert fico to squared value and then fit the training data on it.



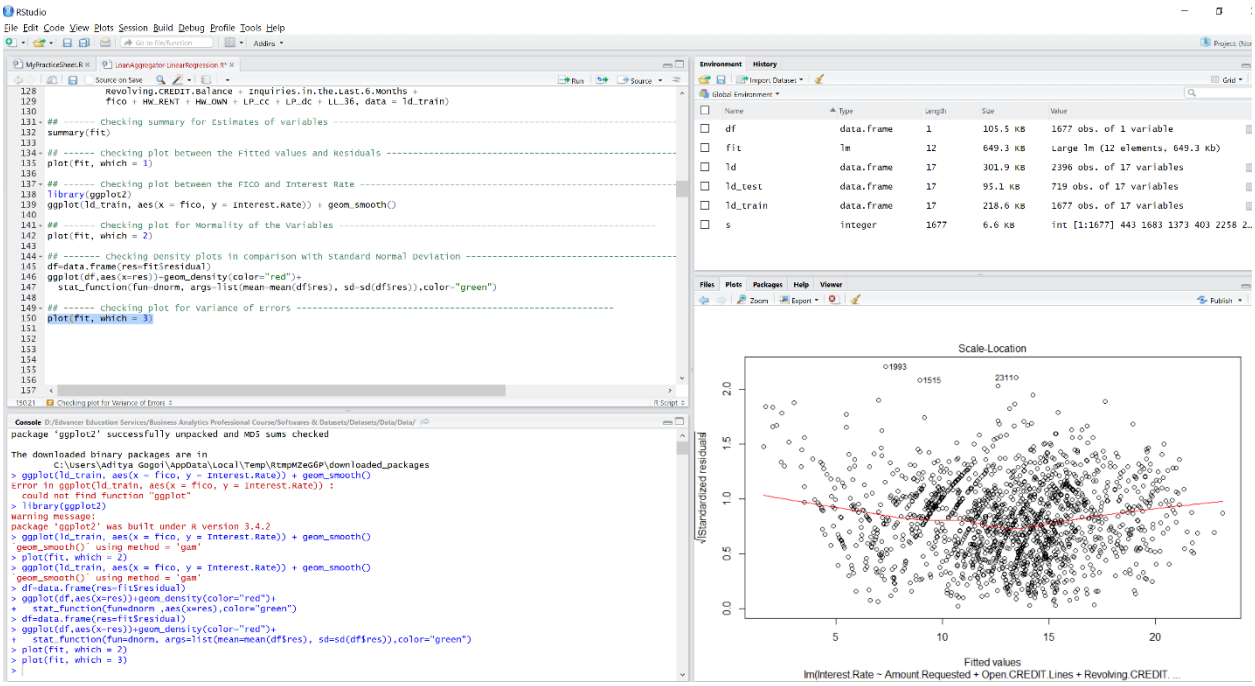
The next plot checks for the Normality of the model. We can see that most of the values fall on the reference line, except for some extreme values. (Check “Normality Plot.png”)



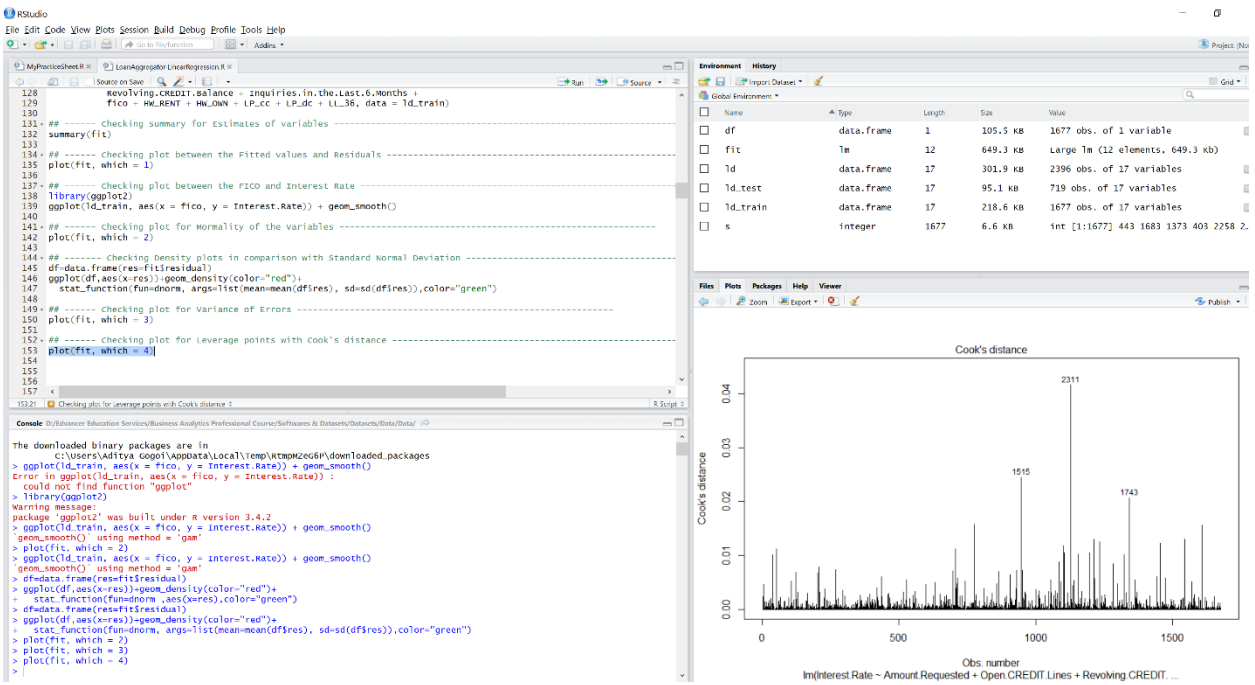
Upon plotting the density function for comparison with a Standard Normal Deviation plot, we see that there is just a slight deviation in our data, which is no cause for concern. (Check "Normality Comparison.png")



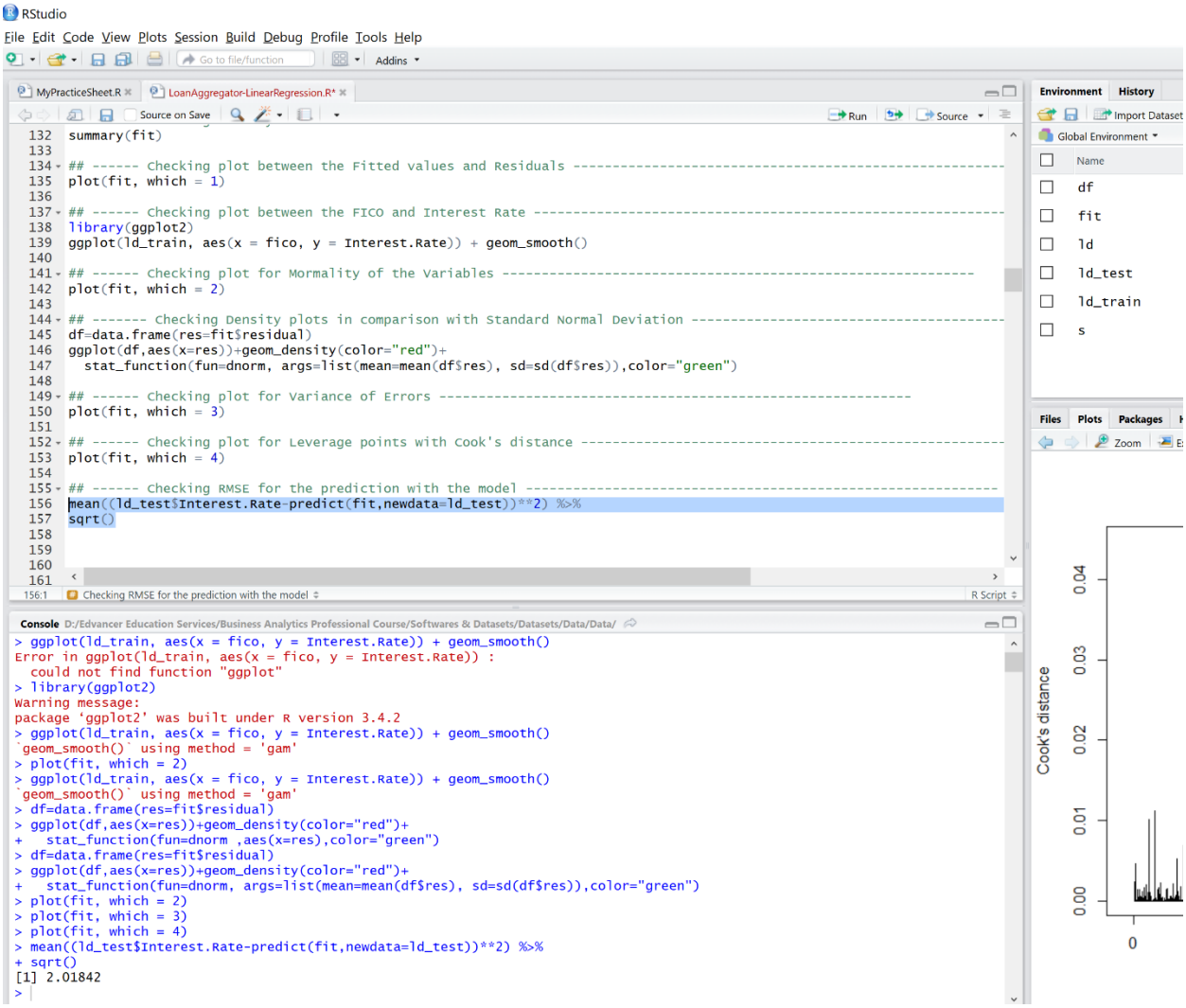
Next, we will check for variance in errors. For all purposes, the plot shows a constant variance in the errors. We see that the scatter is consistent and the thread line is almost linear. Although ideally, it should have been completely flat. (Check "Variance in Errors.png")



We will check for outliers. Certain outliers cause the regression line to shift just slightly from the original incline, because they are already in the same line as the other data points. But some outliers really affect the regression line by a large degree, and these outliers are called 'leverage points'. We need to find out the leverage points and discard them from our dataset. We will use Cook's Distance to see how much does a regression line's inclination change by the absence of a point. If it varies a lot, then that point must be discarded. For our dataset, we see that none of the points have a Cook's distance greater than 1. (Check “Cook’s Distance.png”)



Now that we are sure that the data and model is optimal, we will predict on the test data, get its difference from original values, square it, get its mean and then get its square root. So basically, the Root Mean Square Error (RMSE). The predict() function needs 2 parameters - the model and the data on which I am predicting. The RMSE for this model came at 2.01842.



Conclusion

The regression model we created has a decent RMSE of ~ 2. But this can be improved by including some variables (like state, which might affect the loan interest rate) and making improvements on them.