Applied Predictive Modeling

Lab 1, Part IV Transcript: Useful Data Preprocessing Functions

Now we turn our attention to some useful functions for Module 1. Some of the functions that we will go over are the pairs, correlation, boxplot and skewness. And these functions will be useful for you in Module 1, answering some of the quiz questions, maybe answering some of the assignment questions. So again, let's start with the dataset. As I said, R comes with default datasets that are available when you download R on your system. So Iris is a very well known dataset that people use for a lot of analysis. So we'll use the same dataset, Iris. So data Iris. And then we can look at the structure of Iris.

Again, we see that it's a data frame 150 observations with five variables, four numeric, and one factor variable. So another useful function in R is the pairs function. When you do the pairs, what it does is it plots pairwise scatter plots. Here you can see we have our five variables plotted against each other, it's a pairwise scatter plot. It's a single function and you can nicely see me bivariate trends between variables using this function. And then we could do similarly a correlation with Iris.

And then you see how I get an error here and it is saying x must be numeric. When you pass a data frame or a matrix to a correlation function, it expects everything to be numeric. But we know that when did the structure of our Iris, the species column was a factor, it's not numeric. What we need to do is remove that column. We have learned that in our previous lab sessions, one way to remove a column is do minus the column number.

Because this is column five, and I'll remove column five from Iris and then run the same. Now you can see that you have a correlation matrix and there are no errors. You can see that these are, again, pairwise correlations. Obviously the diagonals are going to be perfectly correlated. And then if you want to see how each variable is correlated, what the correlation coefficient is, you can see that here, sepal length seems to be highly correlated with petal length and petal width, and it is positively correlated. And similarly, you can look at other things and answer any questions between, hey, which variables in your dataset are highly correlated? So I can come in this way.

Now I come to a box plot function. When I do box plot Iris, what you'll see is previously, we saw the pairwise correlations and pairwise scatter plots when we did pairs. This box plot is very similar, but you'll get box plots for each of the variables. You have the sepal length, sepal width, petal length

and petal width and the species. And then you see what those, how the box plots look. For those of you who are not familiar with the box plot, you have the Q1 here, Q2 here, which is your median, and Q3 is your third quartile. And then these whiskers go, these are called the interquartile range.

And then anything about the interquartile range can be considered as an outlier in some sense. That's how these lines are drawn for the box plot. You can learn more about how it is done, again, by going into the function question mark box plot, and it tells you how each of those are done and how the box plots are generated. You will get more input about how R generates these box plots here. Okay, and then we'll go to some other function. So let's look at this other function.

So here, let's go here. Here we have this new function called par What it does is like, "Hey, from now on, I will create whatever plots are created, I want them to be like, condensed into a single plot." Here, I'm saying, "Condense them into one row and two columns." So if I execute this set of commands here in the console, what you'll see is hey, I generated two box plots, And then those two box plots should be arranged in a single plot as one row and two columns, which is what you see in the plot section here. So if I zoom this here, you can see you have two box plots in a single plot. So this is very useful instead of generating multiple plots, you can aggregate all of your things into a single plot and you can use it in your homework and publications and things like that.

And similarly, I do the same thing here instead of one row and two columns here, I can do one row and three columns. If I want to aggregate three plots into one thing, this is how it's going to look like. So you have three plots now. And if I zoom it, you can see hey, three plots, one row and three columns. You could play with this par function and then arrange your plots in however fashion you want. And then, there is this library Hmisc, which is another useful library for plotting data frames, especially.

When you, when I load this libraryHmisc and then do hist data frame, Iris, it generates a histogram of all the variables that are there in this Iris dataset. You can see we had five columns and it shows the variables, all five variables. And it also tells you how many data points are there and if there are any missing data points here. Here in the Iris dataset, we don't have any missing data point. That's why m corresponds to the missing data and none of them are missing. But if we do the same thing for the air quality, we know that in air quality, there are missing data because we have seen that in our previous lab sessions.

Ozone and solar have missing data, that's what it is saying. So we have 116 non missing data and 37 missing data in this case. We have 146 data points and then seven of those are missing, . And 153 data points here, none of them are missing. So this is a nice, useful set of functions that you could use to generate me plots in R. And then next we'll go to Box-Cox transformations.

Box-Cox transformations are used to transform predictor variables that are non normal in nature into a normal distribution. Some of the predictive models that you will use would require the distributions to be Gaussian or normal. In those cases, you may have to transform those variables into normal distribution. Anytime you generate random data, I would recommend you to set the seed.

When you set the seed, you can reproduce your analysis, or if you send your code to me or someone else, they will be able to reproduce it. If not, R will choose the seed by itself. Usually chooses the seed by the current time. And the current time up to a certain position, because there is no way for you to remember the exact time when you generated it, you won't be able to reproduce the analysis, I always recommend people to set the seed whenever you are using random number generators.

Here I'm setting a seed and I'm just creating a random gamma distribution x. So I'm generating thousand values. And if I hover our gamma, it has a shape parameter. And if you don't give a rate, it assumes the rate as one. So I'm giving a shape two, and then I'm going to do, let's see how the histogram looks, right? If you look at the histogram, you know, yeah, this is not perfectly normal, for sure.

It's skewed to the right. So we can certainly improve this distribution to make it look more normal. So, and you can all measure how skewed this distribution is. So in the library, e1071, which has SVM model too. This package has a skewness function that helps you estimate how skewed the distribution is. So let's load the distribution, let's load this package and see how skewed our distribution is. So our skewness is about 1.4. So it's skewed somewhat.

And then if we do the same thing, if we generate the normal distribution, and then we look at the histogram, this looks obviously normal because we generated a random normal distribution, it looks much more normal. And then the skewness obviously going to be close to zero because we generated it from the normal distribution.

Here, the skewness was positively skewed to the right, like we saw in the previous distribution. So let's see how we can and fix the skew distribution of x. So within caret package, we have what is called a Box-Cox trans function that helps us, to change a non-normal distribution to a normal distribution using that Box-Cox transformation.

If I do library , and then apply Box-Cox transformation to x, what we get is a bct. So bct is the variable that holds the Box-Cox transformation, and if we look at it, it's a list with a few parameters, and lambda, if you read the text far, you should know that the lambda is the one that is used to transform the predictor variable into something that is more normal. So here, lambda is estimated to be 0.3, based on the data that was provided.

The way you transform the variable is you use the predict function and say, hey, use the bct that has been calculated by calling the Box-Cox transformation on this x variable. And then you pass again, x. So you predict bct by passing x and then your new x transform should be pretty close to normal. So we can check by plotting the histogram again. So you can clearly see previously it was skewed, now it's much more normal in nature. And you can check the skewness if you want. The skewness, trans. And then see what the skewness value comes out to be.

Previously, we know it was about 1.4. Now you can see it's pretty close to zero. And then another thing you may observe is like, sometimes let's say I create another variable z, where it uses all the values of x that we have previously created, and we add another value zero to it, and let's see what happens. If we do the Box-Cox transformation for this new variable z. So if I look at bct2, what you'll see is lambda is NA. So it cannot estimate the lambda when there are zero values in the variable. So the Box-Cox transformation only works for positive values.

One way you can fix this problem is you can take z and then add a very small number to it. If the z's are in a particular scale where adding this very small number is not going to change it significantly, then you can just do a very, add a very small, positive number to make all the z values positive. And then now run the bct, or the Box-Cox transformation. So now, if you look at the bct2, it was able to estimate, previously it estimated as 0.3, now it estimates it as 0.3. So this is one way to fix the problem if you have zero values in your dataset, then you just add a very small, positive value. In this case, 10 to the power of -4, and then that would fix the Box-Cox transformation problem.