Stat 6021 R Tutorial: Regression with Categorical Predictors

In this tutorial we will learn how to carry out multiple linear regression with categorical predictors in R. We will use the data set "wine.txt". The data set contains ratings of various wines produced in California. For this tutorial, we will focus on the response variable $y = Quality$ (average quality rating), $x_1 = Flavor$ (average flavor rating), and *Region* indicating which of three regions in California the wine is produced in. The regions are coded 1 = North, 2 = Central, and 3 = Napa.

1. Type of variable.

   (a) One of the predictors for this example is *Region*, which we know is categorical. Quite often, the variable may not be recorded the way we want, and R might not recognize the type of variable correctly. Type `is.numeric(Region)`. What is the output from R?

   (b) To have R treat *Region* as a categorical predictor, we use the `factor()` function. For example, we can type `Region<-factor(Region)`. Next, check by typing `is.factor(Region)`. What is the output from R?

2. Dummy coding in R. The default coding scheme used by R is dummy coding. To see the coding scheme that R will use, type `contrasts(Region)`. What is the reference class here?

3. Naming classes.

   (a) To check the names of the classes in the categorical variable, type `levels(Region)`. I suggest giving descriptive names to the various classes for the categorical variable you are working with, instead of leaving the numerical codes. For example,

   ```
   levels(Region) <- c("North", "Central", "Napa")
   ```

   You need to ensure that you are entering the names in the correct sequence. Next, type `contrasts(Region)` to check the coding scheme.

   (b) Changing reference class. You can change the reference class. For example, if you want to set $Region = Napa$ as the reference class, type

```
Region<-relevel(Region, ref = "Napa")
```

Now type `contrasts(Region)` to confirm.

4. Scatterplot, split by classes. It is often useful to create a scatterplot using the two quantitative variables, *Quality* against *Flavor*, and use separate colors and regression lines for each class of *Region*. To do so,

   (a) Subset the data frame into each of the classes. For example,

   ```
   a1<-subset(data,Region=="1")
   a2<-subset(data,Region=="2")
   a3<-subset(data,Region=="3")
   ```

   So we now have three subsets of the dataframe, according to the class of *Region*.

   (b) Next, carry out separate regressions for each subset. We do this because we eventually want to overlay the estimated regression lines for each region on the scatterplot.

   ```
   reg1<-lm(Quality~Flavor,data=a1)
   reg2<-lm(Quality~Flavor,data=a2)
   reg3<-lm(Quality~Flavor,data=a3)
   ```

   (c) Now, we are ready to create the scatterplot of *Quality* against *Flavor*, with separate plots for each region.

   ```
   plot(Flavor,Quality, main="Quality Rating against Flavor Rating, by Region")
   points(a2$Flavor,a2$Quality, pch=2, col="red")
   points(a3$Flavor,a3$Quality, pch=12, col="blue")
   ```

   The commands after `plot()` will give different plotting symbols for each Region. Arguments like `pch` and `col` should be supplied to `plot()` to give different plotting symbols and colors, respectively, for each region. Check `http://www.statmethods.net/advgraphs/parameters.html` for the different symbols.

   (d) Next, we can overlay the estimated regression lines for each region.

   ```
   abline(reg1,lty=1)
   abline(reg2,lty=2, col="red")
   abline(reg3,lty=3, col="blue")
   ```

   The argument `lty` gives different patterns for the regression lines. You want to ensure that the colors specified in these few lines match up with the colors specified under `points()`.

   (e) Finally, we should add a legend to the plot.

   ```
   legend("topleft", c("North","Central","Napa"), lty=c(1,2,3),
   pch=c(1,2,12), col=c("black","red","blue"))
   ```

The most important thing for `legend()` is that when you specify the arguments `lty`, `pch`, and `col`, the order of the values is consistent.

5. Fitting model with interactions. Since the slopes for each region appear to be different, we fit a model with interactions. Type `result<-lm(Quality~Region*Stay)` to fit a model with interaction between *Region* and *Stay*. Carry out the relevant test to see if the interaction terms can be dropped from the model.

6. Assessing regression assumptions.

   (a) Just like we've done before, we need to generate the residual plot, ACF plot of residuals, and normal probability plot of residuals to check if the regression assumptions are met.

   (b) When we have a categorical predictor, we also assume the model variance is constant across all classes. To assess this:

   - Boxplot of the response variable, *Quality*, by the categorical predictor, *Region*. Type `boxplot(Quality~Region)`. What do you see?
   - Levene's test of equality of variances. The function for Levene's test is found in the `lawstat` library. Make sure you install it before typing the following

     ```
     library(lawstat)
     levene.test(Quality,Region)
     ```

     What are the null and alternative hypotheses for Levene's test? What is the result of this test for our model?

7. Pairwise (multiple) comparisons. Suppose we want to compare the means in *Quality* between each pair of regions, for given values of *Flavor*, i.e., North & Central, North & Napa, Central & Napa. The `glht` function in the `multcomp` library will enable us to make all these comparisons, by typing

   ```
   library(multcomp)
   pairwise<-glht(reduced, linfct = mcp(Region= "Tukey"))
   summary(pairwise)
   ```

   This uses the Tukey method, which is less conservative (but a bit more complicated) than the Bonferroni method. Suppose you want to calculate these by hand, type

   ```
   reduced$coef
   vcov(reduced)
   ```

   to obtain the estimated coefficients and the variance-covariance matrix of the estimated coefficients.