EXST 7014, Lab 3: Simple Linear Regression: Regression Diagnostics and Assumptions Test

OBJECTIVES

Simple linear regression (SLR) is a common analysis procedure, used to describe the significant relationship a researcher presumes to exist between two variables: the dependent (or response) variable, and the independent (or explanatory) variable. This lab will familiarize you with how to perform SLR using the lm command in R.

You might notice that a single observation that is substantially different from all other observations can make a large difference in the results of your regression analysis. I f a single observation (or small group of observations) substantially changes your results, you would want to know about this and investigate further. In this lab exercise, we will use appropriate regression diagnostics to detect influential observations.

In this lab exercise, you will get familiar with and understand the following:

1) Use appropriate regression diagnostics to detect influential observations
2) Evaluate the assumptions of SLR using Normality test

Part I. Housekeeping Statements

Before we dive into the main part of the code it is good to create a pre-amble in which we will load all the necessary packages for R to execute the following tasks. Since we will be graphing the scatterplots we need the library "ggplot2". If you have it installed already great. If not, you can install it using the "packages" tab on the bottom right panel. Click install, and put the name of the package you want on the "install packages" window that pops up. The default setting is installing the packages from the CRAN repository where most "mainstream" packages can be found.

To activate the package, use the following command:

```
library(ggplot2)
```

Since we will be analyzing some measures of influence we will also use the library olsrr. Again install it if it is missing and activate it with the command:

```
library(olsrr)
```

Dataset

The data is from your textbook, chapter 7, problem 6 and you can attain it through the link: http://www.stat.lsu.edu/exstweb/statlab/datasets/fwdata97/FW07P06.txt.

The latitude (LAT) and the mean monthly range (RANGE), which is the difference between mean monthly maximum and minimum temperatures, are given for a selected set of US cities. The following program performs a SLR using RANGE as the dependent variable and LAT as the independent variable.

Download and save the data in the same folder as the R-script you are working on. From the tab "Session" on R-Studio select "Set Working Directory" to "Source File location". This instructs R to look for the datafiles in the same directory as the R-script. Then use the command:

```
fw07p06=read.table("data_lab1.txt", header = FALSE, sep = "", dec = ".")
```

The name of the dataset is then **fw07p06** and it is read from the file **data_lab1.txt** using the command **read.table**. The argument **header=FALSE** instructs R to see the first line of the table (the header) as yet another datapoint. If it is set to **header=TRUE** then R will read the first line as the names of the variables corresponding to each column. Our dataset does not have names for the columns and hence we choose header to be false and we will then create the names of the columns ourselves.
The argument **sep=""** tells R to use spaces as separators between each column/variable. It can be changed to **","** if one is using comma separated values (CSV) and others. Finally, the **dec="."** forces R to use a dot **.** as a separator for decimal points. Again this can be changed to commas, semicolons and more. After you run the line above you should have a dataset named **fw07p06** in your data environment on the up right part of R-studio.

To set the variable names we use the command **colnames** as follows:

```
colnames(fw07p06)=c("CITY","STATE","LAT","RANGE")
```

This forces R to rename (or just name) the 4 columns of the dataset using the labels inside the c() argument. Don't forget the c() is used to create an ordered list of elements or a vector.

You can view the dataset by either clicking on it, or by using the command

```
View(fw07p06)
```

Creating a Scatter Plot
When performing a regression analysis, it is always advisable to look at scatter plots of the data in order to get an idea of the type of relationship that exists between the response variable and the explanatory variables. The following commands will create a scatterplot between the two variables.

```
plot1=ggplot(fw07p06, aes(x = LAT, y = RANGE)) +
        geom_point()+
        theme_classic()
plot1
ggsave("Scatter1.pdf",plot1)
```

Let's explore what each line does. The first one utilizes the command ggplot that is used to create good plots using R. Inside the argument of ggplot, we define the dataset which we will use to create the plot to be the dataframe fw07p06 we created earlier. Then we need to define the two

variables in the aes (short for aesthetics) argument. The x axis for us will be the LAT variable (independent) and the y axis will be the RANGE variable (dependent).

The next line instructs R to create the pairs of values as geometrical points (geom_point) there are many features in the geom_point that are left to the user, like shape, size and color of the points. Check the following link for more information:

http://www.cookbook-r.com/Graphs/Scatterplots_(ggplot2)/

The Theme_classic() argument creates the plot with the default settings for ggplot 2. Again the possibilities of customization are endless. Examples of the different themes can be found here: https://ggplot2.tidyverse.org/reference/ggtheme.html

As you might have noticed in the first line, we named our plot **"plot1"**. In order for us to see the plot we need to call it by its name in the script hence the line plot1. Finally, the command **ggsave(Scatter1.pdf,plot1)** saves the plot we created as a pdf file named Scatter1 in the same folder as the script we are working on. This is very handy if one wants to use that for another document. You can also save it as a jpg using the command **ggsave(Scatter1.jpg,plot1)**
Note that the "+" at the end of each line inside the ggplot argument are needed in order for R to consider all three of the lines as one thing.

The statement above will create a scatter plot of RANGE vs. LAT. The graph is not fancy, but is sufficient for getting an idea of how RANGE and LAT are related.
To create more professional graphics, you can explore the R cookbook and ggplot2 in depth here:

http://www.cookbook-r.com/

Part II. Fitting Simple Linear Regression
Model statement with options of INFLUENCE
OUTPUT statement (Predicted values of Y, residual, rstudent, dffits etc)

Based on the scatter plot produced above, we assume that an appropriate regression model relating RANGE and LAT is the liner model given by:

$$y = \beta_0 + \beta_1 \chi + \varepsilon$$

where Y is the RANGE, X is the LAT, and ε is a random error term that is normally distributed with mean 0 and unknown variances σ^2. β_0 is the estimate of Y-intercept, and β_1 is the estimate of the slope coefficient.

R has many different procedures to compute that linear regression but the simples on is the lm method. The command is simply

```
model1 <- lm(RANGE ~ LAT,  data=fw07p06)
```

With this we are creating the object **model1** by invoking the **linear regression model (lm)** between the variables RANGE and LAT (the first is the dependent and the second is the independent). The argument **data=fw07p06** tells R which dataset to use in order to find those variables.

The output of lm is very detailed and provides a lot of information. The following command :

```
summary(model1)
```

Gives us the table:

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6.4793     5.5481  -1.168    0.249
LAT           0.7515     0.1438   5.228 4.79e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The coefficient of LAT (corresponding to β_1) is 0.7515 and the constant term, β_0 is then -6.4793. To find the R squared value we can look at the next output of the summary :

```
Residual standard error: 5.498 on 43 degrees of freedom
Multiple R-squared:  0.3886,   Adjusted R-squared:  0.3744
F-statistic: 27.33 on 1 and 43 DF,  p-value: 4.786e-06
```

The coefficient of determination is labeled Multiple R-squared in the output of summary lm. Also, if one works with more complicated models, the adjusted R-square is preferable, which is similar to R-square but penalizes the use of many explanatory variables. More on that later.
Notice that summary is a very useful generic R command that we will use for other processes as well.

In order for us to get various diagnostics statistics we need to ask R to apply them in the model we have created earlier using the **influence.measures** command

```
influence.measures(model1)
```

This will list the Hat Diag H, the dffits and dfbetas. It also has cook’s d bar and cov.r. These statistics are usually used to detect possible outliers.

The package olsrr has a complete analysis for them including great graphics, so we will be using that. For more information on it you can visit:

https://cran.r-project.org/web/packages/olsrr/vignettes/influence_measures.html

a) To create Cook’s d bar plot and chart we use the following two simple lines with input our model (model 1)

```
CkDbar=ols_plot_cooksd_bar(model1) # Bar plot
CkDchart=ols_plot_cooksd_chart(model1) # Chart
```

Besides the plot that pops up automatically one can view the created lists CkDbar or CkDchart with the commands **View(CkDbar)** or using the global environment to further analyze the results of the cook analysis if needed.
For example, the command

```
CkDbar$outliers
```

Outputs the table:

```
# A tibble: 3 x 2
  observation cooks_distance
        <int>          <dbl>
1           4          0.166
2          33          0.109
3          43          0.328
```

That contains the outliers computed by this method

b) To create the DFBETAs Panel we again ask ols to apply dfbetas analysis on our model as follows:

```
Dfb=ols_plot_dfbetas(model1) #dfbetas analysis
```

Similarly, you can further explore the Dfb list of outputs.

c) To compute the Dffits one should use:

```
Dff=ols_plot_dffits(model1) #dffites analysis
```

Similarly, you can further explore the Dff list of outputs.

d) For the studentized Residual Plot the command is

```
StRes=ols_plot_resid_stud(model1) #Studentized Residual analysis
```

Similarly, you can further explore the StRes list of outputs.

e) For the Standardized Residual Chart you can use

```
ResSta=ols_plot_resid_stand(model1)
```

Similarly, you can further explore the ResSta list of outputs.

In order to compute 95% confidence intervals for betas, we need to use the following code:

```
Bconfi=confint(model1,level =0.95)
Bconfi
```

Confint is a multi-purpose function in R, and the argument level=0.95 changes the confidence interval level. The default, if nothing is provided, is 95%.

To find a confidence interval for the fitted values the command is:

```
Fitconfi=predict(model1, interval="prediction",level=0.95) #For Fitted values
Fitconfi
```

Again, the level can be adjusted to a desired amount. Also, Fitconfi is a matrix with the corresponding values used for further analysis.

To find a confidence interval for the mean values the command is:

```
Meanconfi=predict(model1, interval = 'confidence',level=0.95) # For mean values
Meanconfi
```

Again, the level can be adjusted to a desired amount. Also, Meanconfi is a matrix with the corresponding values used for further analysis.
Finally, if one wants to find the confidence interval or the confidence interval for the mean of a new datapoint with Latitude 10 the following commands will do the trick

```
predict(results,data.frame(LAT=10),interval="confidence") #confidence for mean
predict(results,data.frame(LAT=10),interval="prediction") #confidence for fitted
```

Part III Evaluate Assumption by Residual Analysis

Ggplot2 has an easy way to create the Residual plot for linear regression using the following lines of code:

```
ggplot(lm(RANGE ~ LAT,  data=fw07p06)) +
  geom_point(aes(x=.fitted, y=.resid)) +
  geom_hline(yintercept=0) +
  labs(x="Fitted Values", y="Residuals", title="Residual Plot")+
  theme_classic()
```

As we can see this command plots the fitted values versus the residuals using the geom_point argument. The rest of the commands add a horizontal line at zero, and creates good labels for the x axis, y axis and the general title. The theme_classic() argument gives us the classic white background.

Finally, to test the residuals for normality we will use the Shapiro Wilk Test, which is a popular statistic to evaluate whether the data is normally distributed. It should be noted that the null

hypothesis of Shapiro-Wilk is that the data is normally distributed. If the p value of the test is less than the significant level of 0.05 the null hypothesis is rejected, and we conclude that the data is not normally distributed. Otherwise the null hypothesis cannot be rejected, and we can conclude that the data is normally distributed.

The residuals are saved in the computation of the model under model1$residuals so the following command will compute the corresponding p value for us:

```
shapiro.test(model1$residuals)
```

LAB ASSIGNMENT

Your assignment is to perform necessary analysis using R and answer the following questions:

1. Is the normality assumption violated? State the name and the value of the statistic that you used to reach your conclusion.
2. Does there appear to be any possible influential observation. State the name and the value of the statistics that you used to reach your conclusion.
3. What is the confidence interval for the mean Range for all cities with latitude 32.3?
4. What is the confidence interval for Range for a randomly selected city with latitude 32.3?

*Remember to attach your R script for the lab report.