Hannah Roach
2/28/2019

Lab Exercise 6: Linear Regression

Purpose:	This lab is designed to investigate and practice the Linear Regression method. After completing the tasks in this lab you should able to: • Use R functions for Linear Regression (Ordinary Least Squares – OLS) • Predict the dependent variables based on the model • Investigate different statistical parameter tests that measure the effectiveness of the model

Tasks:	Tasks you will complete in this lab include: • Use the R –Studio environment to code OLS models • Review the methodology to validate the model and predict the dependent variable for a set of given independent variables • Use R graphics functions to visualize the results generated with the model

References:	References used in this lab are located in your ***Student Resource Guide Appendix.***

Workflow Overview

1
- Set Working directory

2
- Use random number generators to create data for the OLS Model

3
- Generate the OLS model using R function “lm”

4
- Print and visualize the results and review the plots generated

5
- Generate Summary Outputs

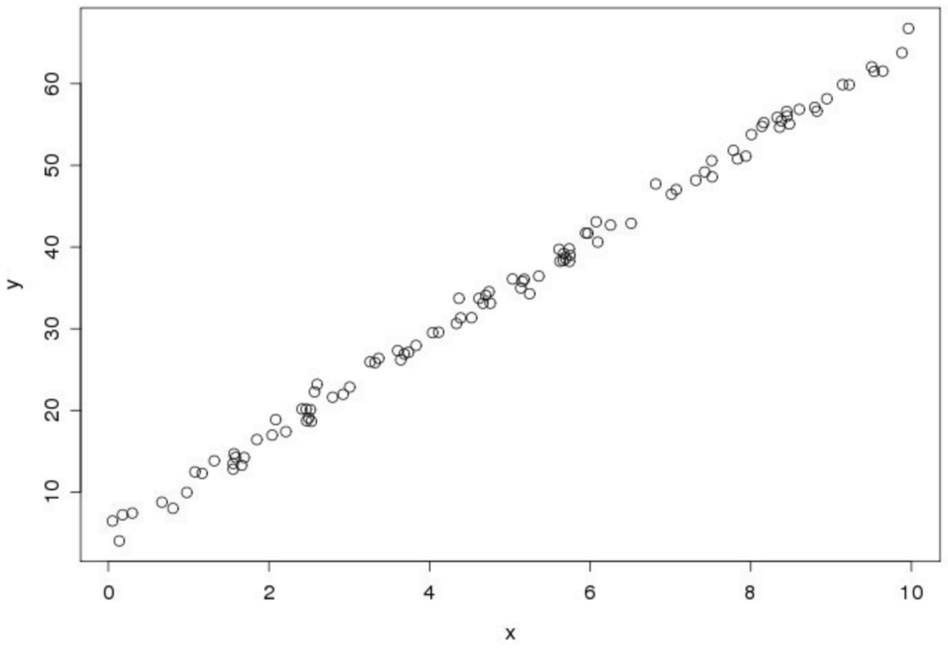
6
- Introduce a slight non-linearity and test the model

7
- Perform In-database Analysis of Linear Regression

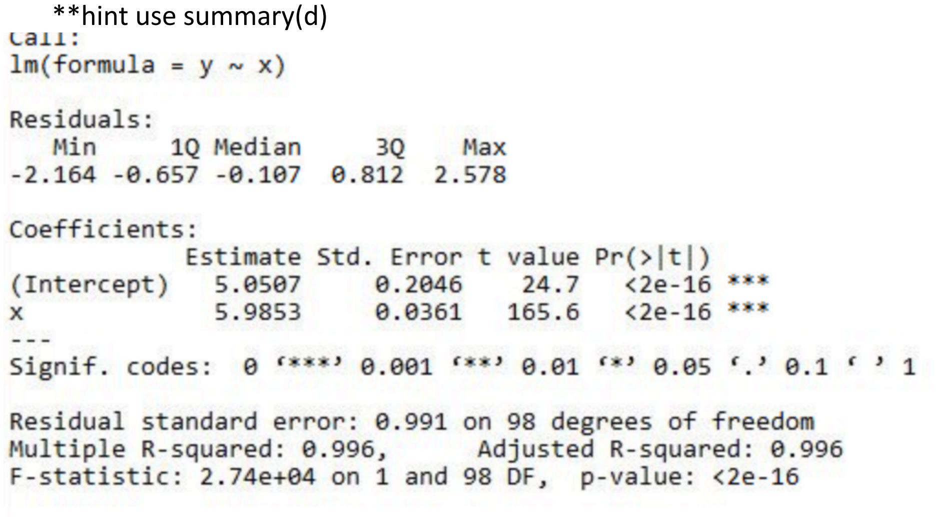
LAB Instructions

Step	Action
1	Log in with GPADMIN credentials on to R-Studio.
2	**Set Working directory** Set the working directory to ~/LAB06/ by executing the command: `setwd("~/LAB06")` • (Or using the "Tools" option in the tool bar in the RStudio environment).

Ste p	Action

3	**<u>Use random number generators to create data for the OLS Model :</u>**
	
	1. Run the “runif” function in R which generates random deviates within the specified minimum and maximum range.
	
	`x <- runif(100, 0, 10)`
	
	This generates 100 random values for “x” in the range 0 to 10.
	2. Create the dependent variable “y” with the “beta” values as 5 and 6 and the “sigma” = 1 (generated with the “rnorm” function, random generation for the normal distribution with mean =0 and SD= 1.)
	
	`y <- 5 + 6*x + rnorm(100)`
	
	3. Plot it
	
	`> plot(x,y)`
	`**SCREENSHOT`
	
	
	
	4. Review the results in the graphics window

Ste p	Action

4	**Generate the OLS model using R function "lm":**
	
	An OLS Model is generated with an R function call "lm".
	You can learn about "lm" with the following command on the console:
	
	`?lm`
	
	1. Generate an OLS Model using the following command:
	`d <- lm(y ~ x)`
	
	2. Use the following command to display the structure of the object "d" created with the function call "lm"
	`str(d)`
	
	3. You can see the details of the fitted model. What are the values of the coefficients (x) in the model?
	**hint use summary(d)
	

```
Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max 
-2.164 -0.657 -0.107  0.812  2.578 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   5.0507     0.2046    24.7   <2e-16 ***
x             5.9853     0.0361   165.6   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.991 on 98 degrees of freedom
Multiple R-squared: 0.996,	Adjusted R-squared: 0.996 
F-statistic: 2.74e+04 on 1 and 98 DF,  p-value: <2e-16
```

5

<u>**Print and visualize the results and review the plots generated**</u>

1. Get the compact results of the model with the following command:

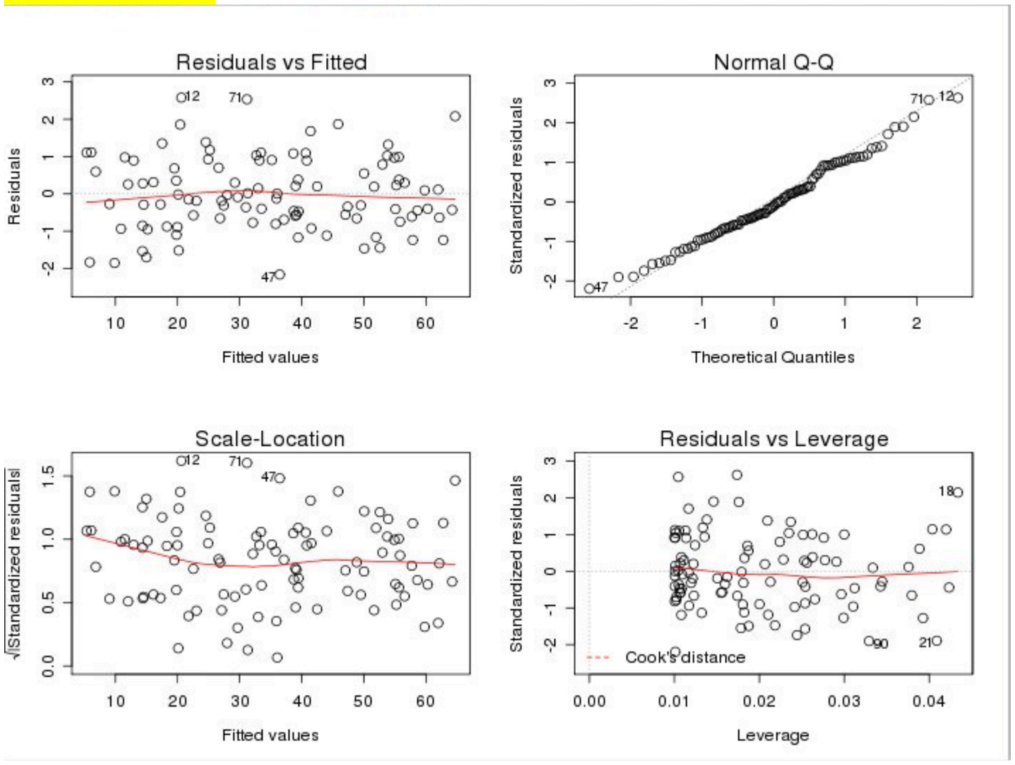
```
print(d)
```

what is the value of the intercept coefficient? 5.05

2. Visualize the model with the command:

```
par(mfrow=c(2,2))
plot(d)
```

****SCREENSHOT**



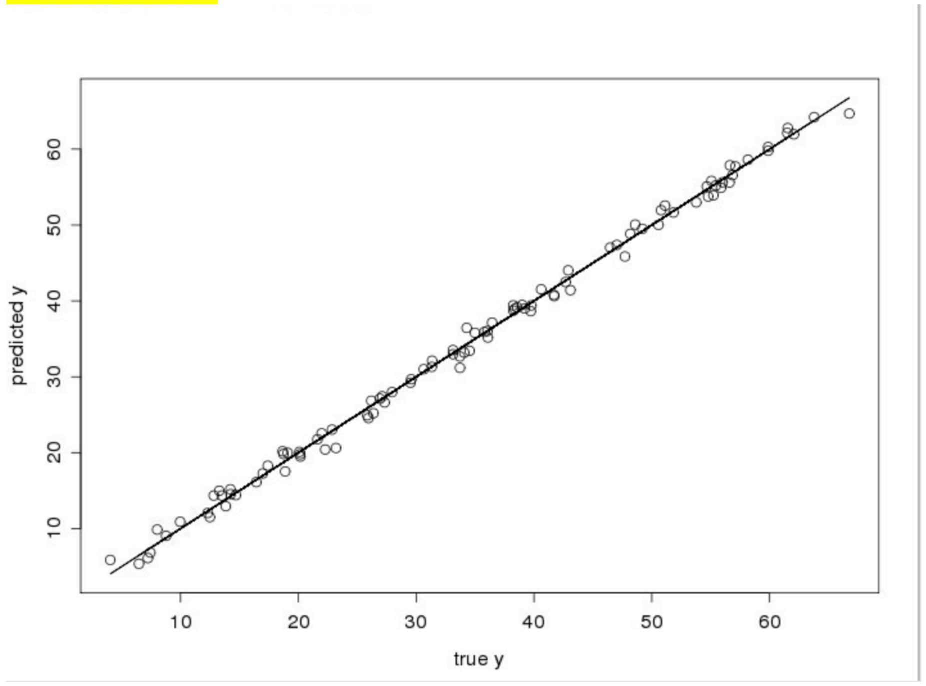
The explanation of the plots are as follows:

Residuals vs. Fitted: you want to make sure that the errors are evenly distributed over the entire range of fitted values; if the errors are markedly larger (or biased either positively or negatively) in some range of the data, this is evidence that this model may not be entirely appropriate for the problem.

Q-Q plot: tests whether or not the errors are in fact distributed approximately normally (as the model formulation assumes). If they are, the Q-Q plot will be along the x=y line. If they aren't, the model may still be adequate, but perhaps a more robust modeling method is suitable. Also, the usual diagnostics (R-squared, t-tests for significance) will not be valid.

Scale-Location: a similar idea to Residuals v. Fitted; you want to make sure that the variance (or its stand-in, "scale") is approximately constant over the range of fitted values.

Ste p	Action

	Residuals vs. Leverage: used for identifying potential outliers and "influential" points. Points that are far from the centroid of the scatterplot in the x direction (high leverage) are influential, in the sense that they may have disproportionate influence on the fit (that doesn't mean they are wrong, necessarily). Points that are far from the centroid in the y direction (large residuals) are potential outliers.
	
	3. Here are some examples of plots that may be a little more intuitive, Type in the following:
	`> ypred <- predict(d)`
	`> par(mfrow=c(1,1))`
	`> plot(y,y, type="l", xlab="true y", ylab="predicted y")`
	`> points(y, ypred)`
	`**SCREENSHOT`
	
	Review the results in the graphics window. The plot of predicted vs. true outcome can be seen there. The plot should be near the x=y line. Where it does not run along the x=y line indicates where the model tends to over-predict or under-predict. You can also use this plot to identify ranges where the errors are especially large. This information is similar to the Residuals vs. Fitted plot, but perhaps is more intuitive to the layperson.
	Note: The "predict" function requires the variables to be named exactly as in the fitted model.

6	**Generate Summary Outputs:**
	
	1. For more detailed results type:
	`d1 <- summary(d)`
	`print(d1)`
	
	Read the explanations given below from the summary output and note the values from the output on the console for each statistic detailed:
	
	coefficients : the estimated value of each coefficient, along with the standard error. coefficient +/- 2*std.error is useful as a quick measure of confidence interval around the estimate.
	
	t-value: coefficient/std.error, or how tight an estimate this is (compared to 0). If the "true" coefficient is zero (meaning this variable has no effect on the outcome), t-value is "small".
	
	Pr(>|t|): the probability of observing this t-value if the coefficient is actually zero. You want this probability to be small. How small depends on the significance desired. Standard significances are given by the significance codes. So, for example "***" means that the probability that this coefficient is really zero is negligible.
	
	R-squared: A goodness of fit measure: the square of the correlation between predicted response and the true response. You want it close to 1. Adjusted R-squared compensates for the fact that having more parameters tends to increase R-squared. Since we only have one variable here, the two are the same.
	
	F-statistic and p-value. Used to determine if this model is actually doing better than just guessing the mean value of y as the prediction (the "null model"). If the linear model is really just estimating the same as the null model, then the F-statistic should be about 1. The p-value is the probability of seeing an F-statistic this large, if the true value is 1. Obviously, you want this value to be very small.
	
	1. Type in the following command:
	`> cat("OLS gave slope of ", d1$coefficients[2,1],`
	`     "and an R-sqr of ", d1$r.squared, "\n")`
	
	2. Note the result you see on the console in the space below:
	OLS gave slope of 5.99 and R-sqr of 0.966

7	**Introduce a slight non-linearity and test the model:**

1. Create a “training” data set

First the training set in which we will introduce a slight non-linearity.

```
> x1 <- runif(100)
> # introduce a slight nonlinearity
> y1 = 5 + 6*x1 + 0.1*x1*x1 + rnorm(100)
```

2. Generate the model

```
> m <- lm(y1 ~ x1)
```

3. Create the test data set

```
> x2 <- runif(100)
> y2 = 5 + 6*x2  + 0.1*x2*x2 + rnorm(100)
```

4. Repeat steps 5 and 6 for model “m” and compare what you observe with the "ideal" in the earlier steps.

5. What's the adjusted R^2 on the model? 0.996

Notice (from the ypred v. y plot) that the model tends to under-predict for higher values of y.

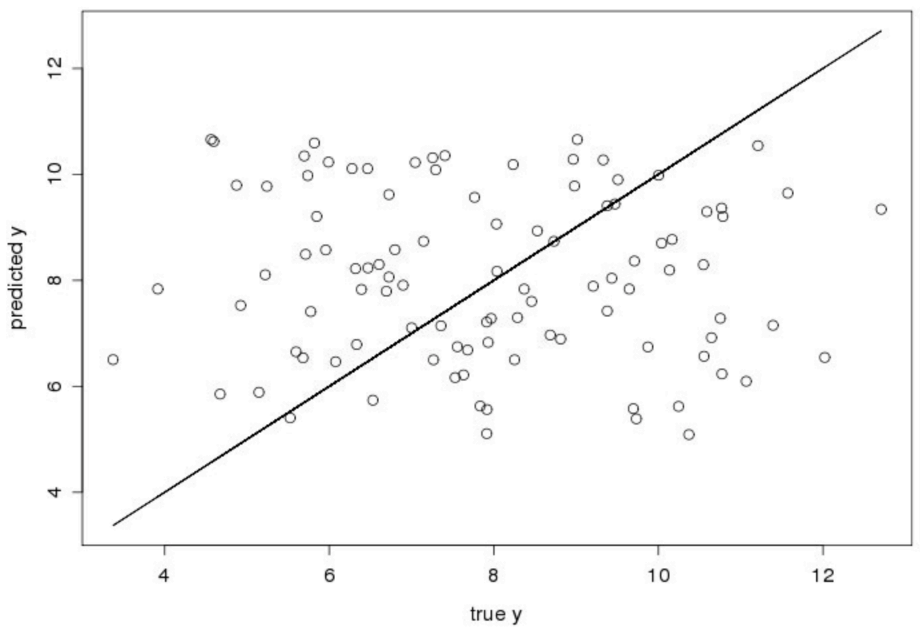
6. Use the predict function on the test data we generated:

```
y2pred <- predict(m,data.frame(x2))
```

7. compare y2pred with the true outcomes y2

**hint use step 5.3 replace ypred with y2pred

8. What did you observe from the comparison?

Ste p	Action
	 The points are not as tightly grouped around the line. This illustrates the predicted versus actual values of y. • The code for this part of the lab is available at the following location: home/gpadmin/LAB06/ols.R

8

Perform In-database Analysis of Linear Regression:

To illustrate an in-database execution of linear regression we make use of
with approximately 64,000 rows of data aggregated from the census data. The following columns are generated for zip code and sex(male/female) tuples:

zip code,

mean age in the zip code,

mean number of years of education,

mean level of employment (Categorical level values were converted to a range from 1 - 3, three being a "high status" job, 1 being "low status"

mean household employment in the zip code

This data is stored in table “zeta” in the database “training2”.

1. Explore the table with Meta commands or with the pgadmin utility
2. Review the MADlib documentation at http://doc.madlib.net/v0.2beta/group__grp__linreg.html and the following code:

```
DROP TABLE IF EXISTS zeta1;
CREATE TABLE zeta1 (
  depvar FLOAT8
  , indepvar FLOAT8[])
DISTRIBUTED BY (depvar)
;

INSERT INTO zeta1 (
  depvar
, indepvar[1]
, indepvar[2]
, indepvar[3]
, indepvar[4]
)
SELECT
  ln(meanhouseholdincome + 1)
, 1
, CASE
    WHEN sex = 'M'  THEN 0
    WHEN sex = 'F' THEN 1
  END AS sex
, meanage
, meanemployment
FROM
  zeta
```

Ste p	Action
	;

8

Cont.

```
SET SEARCH_PATH to madlib,public,myschema;

SELECT (linregr(depvar,indepvar)).r2 FROM zeta1;
SELECT (linregr(depvar,indepvar)).coef FROM zeta1;
SELECT (linregr(depvar,indepvar)).std_err FROM zeta1;
SELECT (linregr(depvar,indepvar)).t_stats FROM zeta1;
SELECT (linregr(depvar,indepvar)).p_values FROM zeta1;
```

We are predicting mean household income with drivers age, sex, and years of employment:

Notice that we take the log of income (refer to the discussions in the student resource guide)

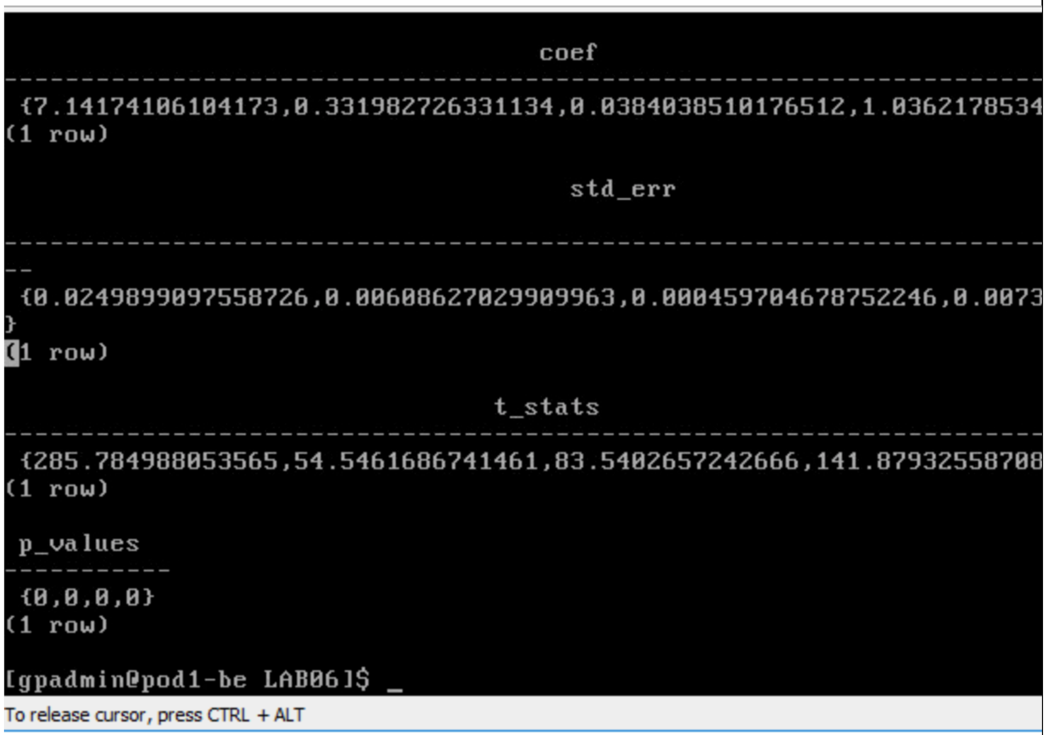
The code is available at

```
/home/gpadmin/LAB06/madliblinear.sql
```

1. You can execute the code with the following command at the command prompt:

```
psql  -d training2 -f madliblinear.sql
```

2. Review the results and note your observations below:



```
                                 coef
-------------------------------------------------------------------------
 {7.14174106104173,0.331982726331134,0.0384038510176512,1.0362178534
(1 row)

                                     std_err

-------------------------------------------------------------------------
--
 {0.0249899097558726,0.00608627029909963,0.000459704678752246,0.0073
}
(1 row)

                               t_stats
-------------------------------------------------------------------------
 {285.784988053565,54.5461686741461,83.5402657242666,141.87932558708
(1 row)

 p_values
-----------
 {0,0,0,0}
(1 row)

[gpadmin@pod1-be LAB06]$ _
```

To release cursor, press CTRL + ALT

Ste p	Action

End of Lab Exercise