## 

Lab 7: Regression

Michael Surdek

Southern New Hampshire University

## **Part A: 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***](http://snhu-media.snhu.edu/files/course_repository/graduate/dat/dat510/dat_510_student_resource_guide_appendix.pdf)***.*** |

### Workflow Overview

### LAB Instructions

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | Download the following file from D2L:   * Ols.R |
| 2 | **Set Working directory**  Set the working directory to <YOUR DIRECTORY> by executing the command:  **setwd("<YOUR DIRECTORY>")**   * (Or using the “Tools” option in the tool bar in the RStudio environment). |
| 3 | **Use random number generators to create data for the OLS Model :**   * 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)**  4. Review the results in the graphics window |
| 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)**   1. You can see the details of the fitted model. What are the values of the coefficients (Beta) in the model?   Beta0 = 4.806  Betax = 6.004   1. Use the following command to display the structure of the object “d” created with the function call “lm”   **str(d)** |
| 5 | **Print and visualize the results and review the plots generated**   1. Get the compact results of the model with the following command:   **print(d)**   1. Visualize the model with the command:   **par(mfrow=c(2,2))**  **plot(d)**  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.  **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.   1. 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)**  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")**   1. Note the result you see on the console in the space below:   OLS gave slope of 6.004345 and an R-sqr of 0.9972704 |
| 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)**   1. Generate the model   **> m <- lm(y1 ~ x1)**   1. Create the test data set   **> x2 <- runif(100)**  **> y2 = 5 + 6\*x2 + 0.1\*x2\*x2 + rnorm(100)**   1. Repeat steps 5 and 6 for model “m” and compare what you observe with the "ideal" in the earlier steps. 2. What's the R2 on the model?   R-sqr = 0.7603  Notice (from the ypred v. y plot) that the model tends to under-predict for higher values of y.   1. Use the predict function on the test data we generated:   **y2pred <- predict(m,data.frame(x2))**   1. compare y2pred with the true outcomes y2 2. What did you observe from the comparison?   The points (y2, y2pred) seem to fill up most of the plot. They are spread out along most of the entire x and y axes. In general, points between true y2 = 4 and 6 have a predicted y2 that is above the y2,y2 line. Points with a true y2 > 9 have a predicted y2 that is below the y2,y2 line.   * The code for this part of the lab is available at the following location: home/gpadmin/LAB07/ols.R |

*End of Lab Exercise*

## **Part B: Logistic Regression**

|  |  |
| --- | --- |
| **Purpose:** | This lab is designed to investigate and practice the Logistic Regression method. After completing the tasks in this lab you should able to:   * Use R functions for Logistic Regression – *also known as Logit*) * 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 R –Studio environment to code Logit 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

### LAB Instructions

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | Download files from D2L:   * Logit.R * Survey.csv |
| 2 | **Set the Working Directory**  Set the working directory to <YOUR DIRECTORY> by executing the command:  **setwd(“<YOUR DIRECTORY>")**   * (Or using the “Tools” option in the tool bar in the RStudio environment). |
| 3 | **Define the problem and review input data**  **Logistic Regression**, also known as ***Logit***, is typically used in models where the dependent variables have a binary outcome (True/False, which is coded with 1/0).  You model the log odds of the outcome as a linear combination of predictor variables).  Marketing Survey Data  In this lab you use hypothetical marketing survey data in which customers:   * Responded to the question:   + Would you choose a product based on a “pricing” factor (three “Price” ranges 10, 20 and 30)? * Response options:   + “1” for “yes” and “0” for “no” * The survey also collected information such as “Age” and “Income” of the respondent.   Business Need  The marketing campaign team wants to send special offers to those respondents with the highest probability of purchase.  You should have downloaded the data file from D2L. If you have not done this yet, do so now: survey.csv.   1. Review the survey.csv file. 2. How many responses to the survey does the file contain?   750 observations   1. What is the main purpose of building this model?   The main purpose of building this model to identify the probability that a particular customer would buy a product at a certain price based on information such as their age and income. |
| 4 | **Read in and Examine the Data:**   1. The first step in the modeling process is to examine the data and determine if there are any outliers. To do this you must read in the survey data, use the following command:   **Mydata <- read.csv("survey.csv",header=TRUE,sep=",")**   1. With the following command, explore the data further:   **> table(Mydata$MYDEPV)**  **> with(Mydata, table(Price,MYDEPV))**  **> summary(Mydata$Age)**  **> cor.mat <- cor(Mydata[,-1])**  **> cor.mat**  3. Review the results on the console  **Note:** The general rule **is not** to include variables in your model that are too highly correlated with other predictors. For example, including two variables that are correlated by 0.85 in your model may prevent the true contribution of each variable from being identified by the statistical algorithm. Confirm that the variables in our survey do not fall in this category. |
| 5 | **Build and Review Logistic Regression Model:**   1. Use the “glm” function for logit modeling. Type in the following command:   **mylogit <- glm(MYDEPV ~ Income + Age + as.factor(Price) ,**  **data=Mydata,family=binomial(link="logit"),**  **na.action=na.pass)**   1. Review the model by typing the “summary” and “plot” functions:   **summary (mylogit)**  Review the results of the summary command, for the fitted model, on the console. Results you should see:   * The first line provides the model you specified. * Next, you should see the **deviance residuals**, which provide the measure of the model fit. * The next part of the output shows the **coefficients,** their standard errors, the **z-statistic** (sometimes called a Wald z-statistic), and the associated **p-values.** * Both **Income** and **Age** are statistically significant, as are the two terms for **Price**. * The **logistic regression coefficients** show the change in the **log odds** of the outcome for a one unit increase in the predictor variable. * Residual deviance: analogous to the Residual Sum of Squares of a linear model; that is, it is related to the "total error" of the fit. It is twice the negative log likelihood of the model. * Null deviance: the deviance associated with the "null model" -- that is the model that returns just the global probability of TRUE for every x. The quantity 1 - (Residual deviance/Null deviance) is sometimes called "pseudo-R-squared"; you use it to evaluate goodness of fit in the same way that R-sqr is used for linear models.   The interpretation of the results are as follows:   1. Review the “Estimate” column. For every one unit change in **Income**, the log odds of Purchase (versus no-Purchase) increases by 0.12876. 2. Record the number that describes how much one unit increase in **Age** increases the log odds of purchase: The indicator variables for **Price** are interpreted differently. **For** example, Purchase decision at a **Price** of 20, compared with a **Price** of **10,** decreases the log odds of admission by 0.74418   For every one unit change in **Age**, the log odds of Purchase (versus no-Purchase) increases by 0.03506.   1. Record the log oddsat Price point 30 compared to Price point 10 below:   At price point 30 compared to price point 10 below, the log odds of Purchase (versus no-Purchase) decreases by 2.21028.  The summary then shows the table of coefficients that are “fit indices”, including the null and deviance residuals and the AIC. |
| 6 | **Review the results and interpret the coefficients**   1. Use the “confint” function to obtain the confidence intervals of the coefficient estimates:     **confint(mylogit)**   1. Review the results on the console.  * You can also exponentiate the coefficients and interpret them as odds-ratios. * To get the exponentiated coefficients, use (**exp( )**) * The object you want to exponentiate is called coefficients and it is part of mylogit (**mylogit$coefficients**).   **exp(mylogit$coefficients)**  You can observe that for every unit change in income, the odd-ratio of Purchase increases by a multiplicative factor of 1.137 (and remember a multiplicative factor of 1 corresponds to no change).  This is actually a bit more intuitive than the log odds explanation you reviewed in the previous step. Observe that that Age does not appear to be a very strong factor in this model, and the price factor of 30 has a stronger effect than a price factor of 20.   1. Pseduo R2 with first obtaining the names of the class members of the model “mylogit” and then using the formula (1 – deviance/null deviance)   **attributes(mylogit)**  **1- with(mylogit, deviance/null.deviance)** |
| 7 | **Visualize the Model Using the Plot Function:**  **plot(mylogit)**  You should see multiple plots generated on the graphics window. |
| 8 | **Use relevel Function to re-level the Price factor with value 30 as the base reference**.  In the original model that we fitted with the function call:  **mylogit <- glm(MYDEPV ~ Income + Age + as.factor(Price)  ,**  **+            data= Mydata,family=binomial(link="logit"), + na.action=na.pass)**  we obtained the results shown below:  **Coefficients:**  **Estimate Std. Error z value Pr(>|z|)**  **(Intercept) -6.02116 0.53244 -11.309 < 2e-16 \*\*\***  **Income 0.12876 0.00923 13.950 < 2e-16 \*\*\***  **Age 0.03506 0.01179 2.974 0.00294 \*\***  **as.factor(Price)20 -0.74418 0.26439 -2.815 0.00488 \*\***  **as.factor(Price)30 -2.21028 0.31108 -7.105 1.2e-12 \*\*\***  **---**  What does this tell us?  The results of summary(mylogit) tell us the estimated values of the coefficients of the variables in the model, how tight of an estimate each is, and the probability that including the coefficient in the model is no better than guessing.  The odds of MYDEPV decreases when price changes from 10 to 20 and decreases even more when we go from 10 to 30.  1. Now let's use 30 as the reference price, instead of 10. Type in the following:  **Mydata$pricefactor = relevel(as.factor(Mydata$Price), "30")** |
| 8  Cont. | Fit the Model Again (mylogit2) and Display the Summary:  **mylogit2 = glm(MYDEPV ~ Income + Age + pricefactor ,**  **+ data= Mydata,family=binomial(link="logit"),**  **+ na.action=na.pass)**  **summary(mylogit2)**  You will see the results as follows:  **Coefficients:**  **Estimate Std. Error z value Pr(>|z|)**  **(Intercept) -8.23144 0.66180 -12.438 < 2e-16 \*\*\***  **Income 0.12876 0.00923 13.950 < 2e-16 \*\*\***  **Age 0.03506 0.01179 2.974 0.00294 \*\***  **pricefactor10 2.21028 0.31108 7.105 1.20e-12 \*\*\***  **pricefactor20 1.46610 0.29943 4.896 9.76e-07 \*\*\***  **---**  Notice that the intercept has changed (because we changed the reference situation), but the coefficients for Income and Age are the same. The new model tells us that the odds of MYDEPV increase when price decreases from 30 to 10, and less so price decreases from 30 to 20. |
| 9 | **Plot the ROC Curve:**   1. Make sure you have the package ROCR installed and the library included   **install.packages("ROCR", dependencies = TRUE)**  **library(ROCR)**   1. First get all the probability scores on the training data   **pred = predict(mylogit, type="response")**   1. Every classifier evaluation using ROCR starts with creating a prediction object. This function is used to transform the input data (which can be in vector, matrix, data frame, or list form) into a standardized format.   We create the prediction object needed for ROCR as follows:  **predObj = prediction(pred, Mydata$MYDEPV)**   1. All kinds of predictor evaluations are performed using the function “performance”. Read and understand the parameters of the function with   **?performance**   1. We now create the ROC curve object and the AUC object with performance function   **rocObj = performance(predObj, measure="tpr", x.measure="fpr") # creates ROC curve obj**  **aucObj = performance(predObj, measure="auc") # auc object**   1. Extract the value of AUC and display on the console:   **auc = aucObj@y.values[[1]]**  **auc**   1. What is the value of AUC?   0.915272   1. We will plot the ROC curve now   **plot(rocObj, main = paste("Area under the curve:", auc))**   1. Review the curve on the plot window. Review the discussions on ROC in the student resources guide. Record your observations below: |
| 10 | **Predict Outcome given Age and Income:**   1. Use the “predict” function to predict the probability of the purchase outcome given **Age** and **Income.** Start with predicting the probability of the purchase decision at different **Price** points (10, 20, and 30). Create a “data frame” called “newdata1” using the following commands:   **Price <- c(10,20,30)**  **Age <- c(mean(Mydata$Age))**  **Income <- c(mean(Mydata$Income))**  **newdata1 <- data.frame(Income,Age,Price)**  **newdata1**  You are predicting with **Income** and **Age** both set at their mean value and **Price** at 10, 20 and 30.  **Note:** The values of the data frame “newdata1” displayed on the console. The predict function requires the variables to be named exactly as in the fitted model.   1. Create the fourth variable “PurchaseP”.   **newdata1$PurchaseP <- predict (mylogit,newdata=newdata1,type="response")**  **newdata1**   1. What is your observation on the probability of purchase at different **Price** levels?   The probability of purchase decreases as price goes up. When price goes from 10 to 20, the probability of purchase decreases by 18%, and when the price goes from 20 to 30, the probability of purchase decreases by 31%. |
| 11 | **Predict outcome for a sequence of Age values at price 30 and income at its mean:**   1. Keep the **Price** at 30, **Income** at its mean value and select a sequence of values for **Age** starting at a minimum age, incrementing by 2 until the maximum age in our dataset:   **newdata2 <- data.frame(Age=seq(min(Mydata$Age),max(Mydata$Age),2),**  **Income=mean(Mydata$Income),Price=30)**  **newdata2$AgeP<-predict(mylogit,newdata=newdata2,type="response")**  **cbind(newdata2$Age,newdata2$AgeP)**  Newdata2$AgeP stores the predicted variables and you just display the sequence for **Age** you generated and the corresponding probability of the purchase decision using the “cbind” function shown above.   1. Plot and visualize how the “purchase” probability varies with Age:   **plot(newdata2$Age,newdata2$AgeP)** |
| 12 | **Predict outcome for a sequence of income at price 30 and Age at its mean:**   1. Using the same methodology, create a data frame newdata3 with the following characteristics:  * **Income** is a sequence from 20 to 90 in steps of 10 * **Age** is the mean value for the dataset Mydata * **Price** point at 30  1. Predict **newdata3$IncomeP** and display the Income sequence along with the predicted probabilities. 2. Plot the results. |
| 13 | **Use Logistic regression as a classifier:**  Recall the problem statement in Step 3, the marketing campaign team wants to send special offers to those respondents with the highest probability of purchase. They have established a threshold of 0.5 and they want to target customers whose probability of purchase are greater than 0.5.  **Note:** We are assuming that age and income are uniformly distributed in our customer base, and the price factors of our products are also uniformly distributed. Typically in order to run a scenario like this you should understand the demographic distribution of the customers (and the price distribution of the products).   1. You want an idea of how many offers will be sent out, using this threshold, so you test it on a 'random' set of data. First, generate this random set using “runif” functions:   **newdata4 <- data.frame (**  **Age= round(runif(10,min(Mydata$Age),max(Mydata$Age))),**  **Income=round(runif(10,min(Mydata$Income),max(Mydata$Income))),**  **Price = round((runif(10,10,30)/10))\*10)**  **newdata4$Prob <- predict(mylogit,newdata=newdata4,type="response")**  **newdata4**   1. How many samples in your random selection qualify for special offers?   8 samples |

*End of Lab Exercise – Part B*