ITEC 621 Exercise 4 - Data Pre-Processing

Transformations

Johnson Odejide

February 17, 2023

Table of Contents

[General Instructions 1](#_Toc127872834)

[Data Work - SoftwareData.csv 2](#_Toc127872835)

[1. Categorical (Factor) Predictors 2](#_Toc127872836)

[2. Log-Linear Model 5](#_Toc127872837)

[3. Standardized Regression 14](#_Toc127872838)

[4. Multivariate Time Series 16](#_Toc127872839)

## General Instructions

Download the **Ex4\_DataPreProcessing\_YourLastName.Rmd** R Markdown file and save it with your own **last name** and **date**. Complete all your work in that template file.

**IMPORTANT:**

**Knitting to HTML:** We normally knit our R Markdown scripts to a Word file, but in this exercise I’m asking you to practice with the knitting to an HTML file, which you could publish on a web site. Knit your .Rmd file into an **HTML** file (the output html\_document header is already provided for you in the template). Your knitted document **must display your R commands**. Knitting and formatting is worth up to **3 points** in this and all exercises.

**Formatting:** Please ensure that all your text narratives are fully visible (if I can’t see the text, I can’t grade it). Also, please ensure that your **Table of Contents** is visible and properly formatted. Also, please prepare your R Markdown file with a **professional appearance**, as you would for top management or an important client. Please, write all your interpretation narratives in the text area, outside of the R code chunks, with the appropriate formatting and businesslike appearance. **Note:** I write all my interpretation solutions inside of the R code chunk to suppress their display until I print the solution, but don’t need to do this. I will read your submission as a report to a client or senior management. Anything unacceptable to that audience is unacceptable to me.

**Important Formatting Tip About the # Tag:** Many students submit their knitted file with text narratives embedded in the table of contents and with the text in the main body in large blue font. This is **NOT** proper business formatting. This is the issue: if you want to write comments inside an R code chunk, you need to use the # tag, which tells R that that line should not be executed and it is there as a comment only. However, if you use the # tag in the text area, R Markdown treats this as **Heading 1** text and ## as **Heading 2** text. Heading text will appear in the table of contents and in large blue font in the main text. Please **DO NOT** use # tags in the main text, except for actual headers and sub-headers in your document.

**Submission**: Submit your knitted homework HTML file in Canvas. There is no need to submit the .Rmd file, just your knitted file.

## Data Work - SoftwareData.csv

The first part of this exercise will be done with the This analysis will be done with the **SoftwareData.csv** data set

Read the **SoftwareData.csv** data file into a data frame.This file is a subset from data I collected from the configuration management system of a large software repository that keeps statistics about software modifications for this company. Each observation represents a software **modification request (MR)**. Once approved, the MR number becomes the modification identifier. An MR record contains the following information:

* **dev.days**: number of days it took to complete the MR
* **priority**: categorical - VeryHigh, High, Medium or Low
* **num.modifs**: number of file modifications made in the MR
* **num.modules**: number of software modules affected by the MR
* **teams.size**: number of software developers who worked on the MR

Use the read.table() function with the header = T , sep = ",", stringsAsFactors = T parameters. If you are using R vesion 3.xx the **priority** variable will be read correctly as a **factor**. But version 4.xx will read it as a **character**, so stringsAsFactors = T will read that variable correctly as a factor. Store the data read in a data frame named **swd**. Briefly review the data set outside of the script (in the R Console or from the Environment tab).

Also, set the options(scipen = 4) to minimize the use of scientific notation.

# Done for you  
swd <- read.table("../../Dataset/SoftwareData.csv",   
 header = T,   
 sep = ",",   
 stringsAsFactors = T)  
  
options(scipen = 4)

## 1. Categorical (Factor) Predictors

1.1 The **priority** variable is categorical (i.e., factor). Firs, list the class() of this variable and ensure that it is a **factor** variable. Also, the levels of this factor variable using the levels() function.

class(swd$priority)

## [1] "factor"

levels(swd$priority)

## [1] "High" "Low" "Medium" "VeryHigh"

1.2 Fit an OLS model using **lm()** to predict the time to completion of the MR, **dev.days**, using **priority, num.modifs, num.modules** and **team.size** as predictors. Store your **lm()** object results in **lm.fit**. Then display the summary() results.

lm.fit <- lm(dev.days ~ priority + num.modifs + num.modules + team.size, data = swd)  
summary(lm.fit)

##   
## Call:  
## lm(formula = dev.days ~ priority + num.modifs + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -115.69 -37.99 -25.88 0.68 521.36   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.48742 7.27956 1.166 0.24387   
## priorityLow 20.63773 6.70379 3.079 0.00213 \*\*   
## priorityMedium 3.24761 5.40330 0.601 0.54792   
## priorityVeryHigh -9.90756 12.25952 -0.808 0.41916   
## num.modifs 0.04607 0.03336 1.381 0.16757   
## num.modules 3.12954 0.55206 5.669 0.0000000179 \*\*\*  
## team.size 12.41606 2.66888 4.652 0.0000036399 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 79.72 on 1230 degrees of freedom  
## Multiple R-squared: 0.0829, Adjusted R-squared: 0.07843   
## F-statistic: 18.53 on 6 and 1230 DF, p-value: < 2.2e-16

1.3 Briefly interpret both, the **priorityLow coefficient** and its **p-value** (3 lines max)

**Answer:**

*The effect of* ***priorityLow*** *is positive and it is significant at the p-value=0.002. On average, holding everything else constant, the number of days it takes to complete low priority modification requests (MRs) is about 20.64 longer than the number of days taken to complete High priority MRs (which is the reference level).*

1.4 Notice that the reference level (i.e., the excluded category) is **High**, which is the first alphabetically, but is in the middle of the scale so it is probably not a very useful reference level. Let’s try using **Low** as the reference level, which is more useful for comparisons. Use the relevel() function to re-level the **swd$priority** factor variable to set **“Low”** (ref = "Low") as the reference level and then fit the same regression model above, but this time store it in **lm.fit.rlv**. Then, display the model results with the summary() function.

swd$priority <- relevel(swd$priority, ref = "Low")  
  
lm.fit.rlv <- lm(dev.days ~ priority + num.modifs + num.modules + team.size, data = swd)  
summary(lm.fit.rlv)

##   
## Call:  
## lm(formula = dev.days ~ priority + num.modifs + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -115.69 -37.99 -25.88 0.68 521.36   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 29.12515 8.70014 3.348 0.00084 \*\*\*  
## priorityHigh -20.63773 6.70379 -3.079 0.00213 \*\*   
## priorityMedium -17.39012 6.10306 -2.849 0.00445 \*\*   
## priorityVeryHigh -30.54528 12.77431 -2.391 0.01695 \*   
## num.modifs 0.04607 0.03336 1.381 0.16757   
## num.modules 3.12954 0.55206 5.669 0.0000000179 \*\*\*  
## team.size 12.41606 2.66888 4.652 0.0000036399 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 79.72 on 1230 degrees of freedom  
## Multiple R-squared: 0.0829, Adjusted R-squared: 0.07843   
## F-statistic: 18.53 on 6 and 1230 DF, p-value: < 2.2e-16

1.5 Briefly explain how the 3 priority coefficients changed (in 3 to 4 lines)

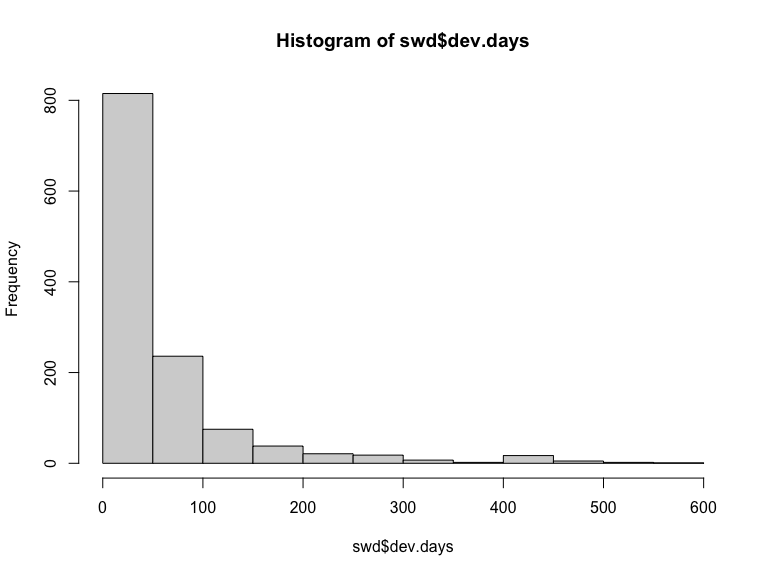
**Answer:** *Since* ***priorityLow*** *is now the reference, it compares the effect of the other priorities relative to Low. Hence, the 3 priority coefficients are now negative and significant. On average, holding everything else constant, it takes about 20.64, 17.39, and 30.55 days faster to perform high, medium, and very high priority modifications respectively than a Low priority modification.*

## 2. Log-Linear Model

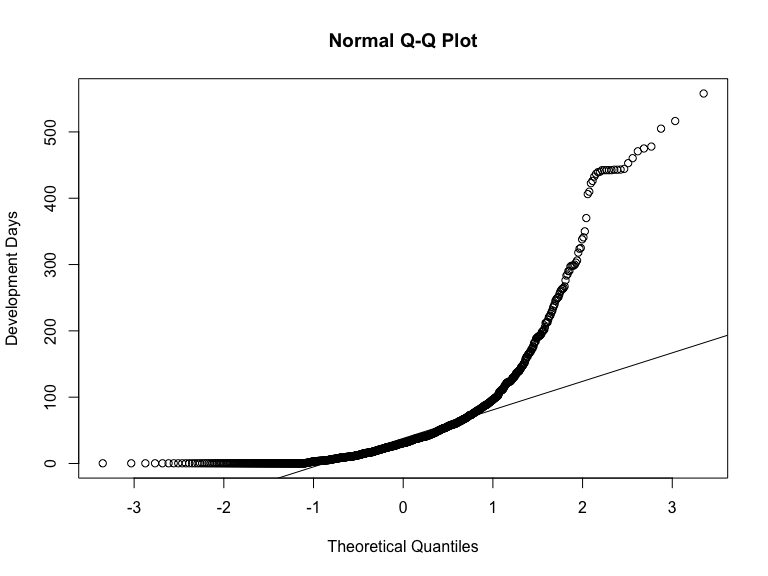
2.1 Display a **histogram** and a **qqplot** for the outcome variable **swd$dev.days**. To make the plot more informative, add a label in the vertical axis of qqnorm() with the parameter ylab = "Development Days".

Also, display a residual plot() for the **lm.fit** model above, using the parameter which = 2 to display only the second plot only, which shows the qqplot for residuals.

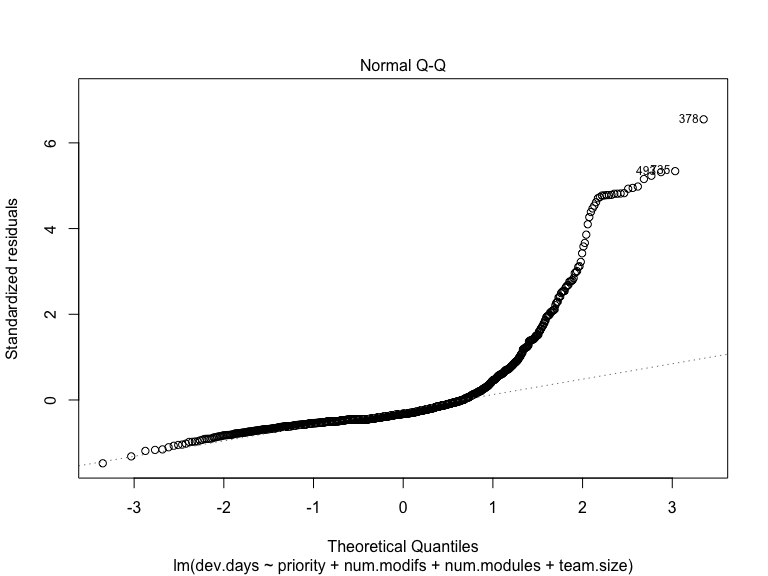
hist(swd$dev.days)



qqnorm(swd$dev.days, ylab = "Development Days")  
qqline(swd$dev.days)



plot(lm.fit, which = 2)



2.2 Do you think we should **log-transform** the **dev.days** variable? Why or why not? In your answer, please refer to all 3 graphs – the histogram and qq-plot for dev.days and the residual plot for lm.fit.

**Answer:**

*The histogram is skewed to the right indicating the need to transform. Furthermore, both qq-plots shows deviation from the qq-line indicating that neither dev.days nor the residuals are normally distributed. Hence, we can try to log-transform the outcome variable, that is, dev.days*

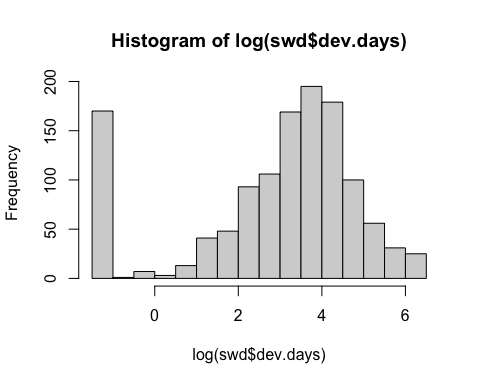
2.3 Fit a **log-linear** model and store the results in an object named **lm.fit.log**. Display the summary() results.

lm.fit.log <- lm(log(dev.days) ~ priority + num.modifs + num.modules + team.size, data = swd)  
  
summary(lm.fit.log)

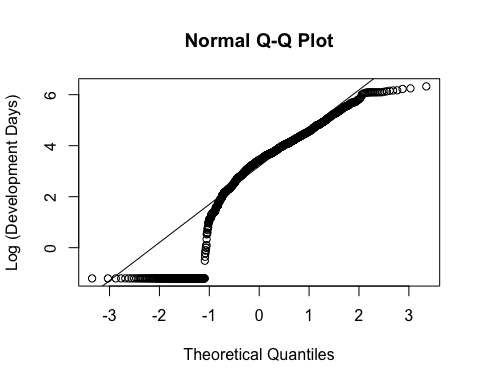
##   
## Call:  
## lm(formula = log(dev.days) ~ priority + num.modifs + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.2705 -0.9233 -0.0657 0.8438 5.0941   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.225738 0.178913 18.030 < 2e-16 \*\*\*  
## priorityHigh -2.430985 0.137859 -17.634 < 2e-16 \*\*\*  
## priorityMedium -0.646383 0.125505 -5.150 3.03e-07 \*\*\*  
## priorityVeryHigh -1.678112 0.262695 -6.388 2.38e-10 \*\*\*  
## num.modifs 0.003583 0.000686 5.223 2.07e-07 \*\*\*  
## num.modules 0.030633 0.011353 2.698 0.007064 \*\*   
## team.size 0.198777 0.054884 3.622 0.000304 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.639 on 1230 degrees of freedom  
## Multiple R-squared: 0.321, Adjusted R-squared: 0.3177   
## F-statistic: 96.9 on 6 and 1230 DF, p-value: < 2.2e-16

It is always a good idea to re-check the normal distribution of the outcome variable and residuals with the logged outcome variable. Repeat the graphs you did in 2.1 but using the log() function this time. As you did in 2.1 above, add a ylab = parameter to qqnorm() to label the vertical axis "Log (Development Days)".

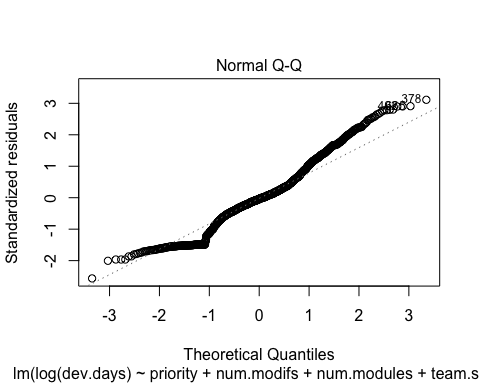
hist(log(swd$dev.days))



qqnorm(log(swd$dev.days), ylab = "Log (Development Days)")  
qqline(log(swd$dev.days))



plot(lm.fit.log, which = 2)



qqline

## function (y, datax = FALSE, distribution = qnorm, probs = c(0.25,   
## 0.75), qtype = 7, ...)   
## {  
## stopifnot(length(probs) == 2, is.function(distribution))  
## y <- as.vector(quantile(y, probs, names = FALSE, type = qtype,   
## na.rm = TRUE))  
## x <- distribution(probs)  
## if (datax) {  
## slope <- diff(x)/diff(y)  
## int <- x[[1L]] - slope \* y[[1L]]  
## }  
## else {  
## slope <- diff(y)/diff(x)  
## int <- y[[1L]] - slope \* x[[1L]]  
## }  
## abline(int, slope, ...)  
## }  
## <bytecode: 0x7f7b5de32830>  
## <environment: namespace:stats>

Please check this and briefly comment on whether the normality issue was corrected.

**Answer:**

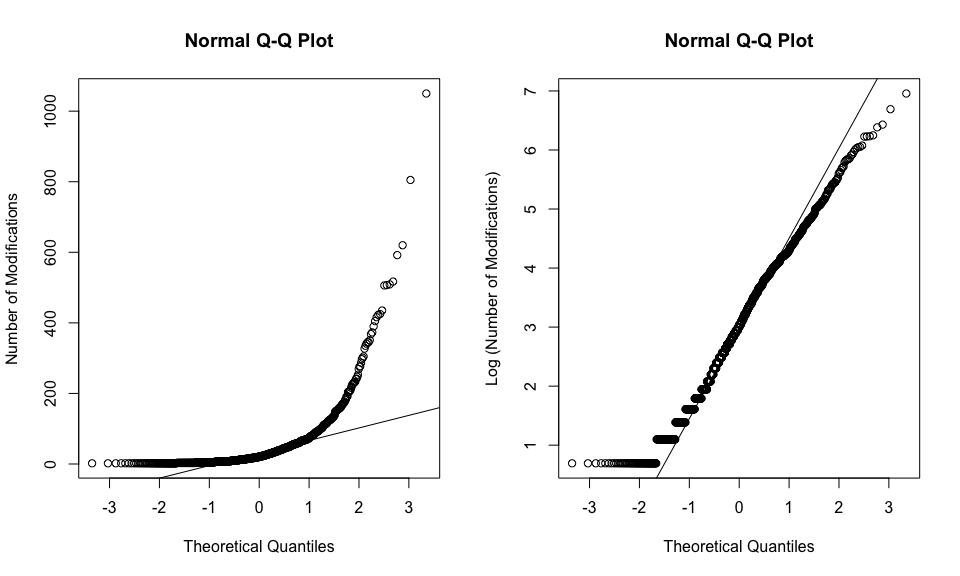
*The distribution of the outcome variable dev.days now looks somewhat normal, although not perfect, it showed some substantial improvement compared to when it was not transformed.*

*Following the qqplot for log(dev.days), the distribution is roughly aligned with the qq-line, although this does not hold true for lower values of dev.days which appear to be off the qq-line.*

*Additionally, the residual plot indicates the normality of distribution at the center with the very high and very low values of dev.days departing from normality.*

2.4 We suspect that the predictor **num.modifs** is not normally distributed. While this is not a problem for OLS, it is probably creating some non-linearity issues with the outcome variable dev.days. First, draw a QQ Plot for this predictor variable, along with the QQ Plot of the respective log( ) of the variable. Divide the output into 2 rows x 2 columns. Then render the 2 QQ Plots side by side. The first plot should have a vertical axis label “Number of Modifications”, and the second plot should have a vertical axis label “Log (Number of Modifications). Then reset the output to 1 row and 1 column.

par(mfrow = c(1, 2))  
  
qqnorm(swd$num.modifs, ylab = "Number of Modifications")  
qqline(swd$num.modifs)  
  
qqnorm(log(swd$num.modifs), ylab = "Log (Number of Modifications)")  
qqline(log(swd$num.modifs))



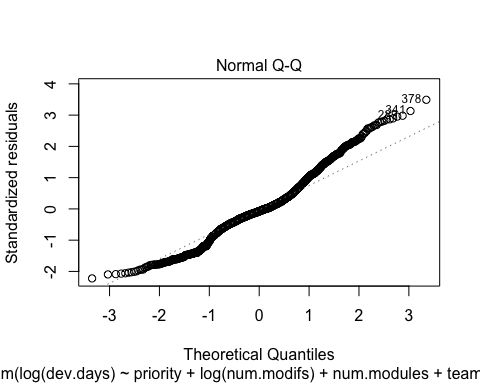
par(mfrow = c(1, 1))

2.5 Fit a **log-log** model and store the results in an object named **lm.fit.loglog**. Log the outcome variable **dev.days** and the predictor **num.modifs**. Include the raw (not logged) predictors **priority**, **num.modules** and **team.size**. Display the summary() results and then plot the **lm.fit.loglog** residual QQ Plot using the which = 2 parameter.

lm.fit.loglog <- lm(log(dev.days) ~ priority + log(num.modifs) + num.modules + team.size, data = swd)  
summary(lm.fit.loglog)

##   
## Call:  
## lm(formula = log(dev.days) ~ priority + log(num.modifs) + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.5653 -0.8963 -0.1227 0.7995 5.6158   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.43325 0.19848 12.259 < 2e-16 \*\*\*  
## priorityHigh -2.23885 0.13873 -16.138 < 2e-16 \*\*\*  
## priorityMedium -0.58833 0.12369 -4.757 2.20e-06 \*\*\*  
## priorityVeryHigh -1.32316 0.26385 -5.015 6.09e-07 \*\*\*  
## log(num.modifs) 0.36258 0.04367 8.303 2.65e-16 \*\*\*  
## num.modules 0.01662 0.01125 1.477 0.140   
## team.size 0.12303 0.05519 2.229 0.026 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.613 on 1230 degrees of freedom  
## Multiple R-squared: 0.3427, Adjusted R-squared: 0.3395   
## F-statistic: 106.9 on 6 and 1230 DF, p-value: < 2.2e-16

plot(lm.fit.loglog, which = 2)



2.4 Which is the best model of the 3, lm.fit (linear), lm.fit.log (log-linear) or lm.fit.loglog (log-log)? In your answer, refer to the p-values of the coefficients and the adjusted R-squares.

**Technical Note:** Since the logged models are NOT nested within the others, you **can’t use ANOVA** to compare them. Just use the **Adjusted R-squared** for now (we will use cross-validation for this purpose later on).

**Answer:**

*The log-log model is the best because it has the highest adjusted R-square, followed by the log-linear model. Both log models show a significant difference in adjusted R-square when compared to the linear OLS model. There are also more significant coefficients which also have stronger levels of significance.*

## 3. Standardized Regression

3.1 As we discussed, there are two ways to run standardized regressions. The first one is to use the scale() function to standardize either the entire data set, or just specific variables of interest. However, standardizing individual variables is a tedious process. It is much easier to fit a plain OLS regression and then extract standardized regression coefficients from raw coefficients and display both of them side by side. Since you already have a fitted linear model **lm.fit**, load the **{lm.beta}** library and use the lm.beta() function to extract standardized regression results. Store these results in an object named **lm.fit.std** and then display a summary() for this object.

library(lm.beta)  
  
lm.fit.std <- lm.beta(lm.fit)  
summary(lm.fit.std)

##   
## Call:  
## lm(formula = dev.days ~ priority + num.modifs + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -115.69 -37.99 -25.88 0.68 521.36   
##   
## Coefficients:  
## Estimate Standardized Std. Error t value Pr(>|t|)   
## (Intercept) 8.48742 NA 7.27956 1.166 0.24387   
## priorityLow 20.63773 0.10014 6.70379 3.079 0.00213 \*\*   
## priorityMedium 3.24761 0.01950 5.40330 0.601 0.54792   
## priorityVeryHigh -9.90756 -0.02305 12.25952 -0.808 0.41916   
## num.modifs 0.04607 0.04308 0.03336 1.381 0.16757   
## num.modules 3.12954 0.16960 0.55206 5.669 0.0000000179 \*\*\*  
## team.size 12.41606 0.13420 2.66888 4.652 0.0000036399 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 79.72 on 1230 degrees of freedom  
## Multiple R-squared: 0.0829, Adjusted R-squared: 0.07843   
## F-statistic: 18.53 on 6 and 1230 DF, p-value: < 2.2e-16

summary(lm.fit)

##   
## Call:  
## lm(formula = dev.days ~ priority + num.modifs + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -115.69 -37.99 -25.88 0.68 521.36   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.48742 7.27956 1.166 0.24387   
## priorityLow 20.63773 6.70379 3.079 0.00213 \*\*   
## priorityMedium 3.24761 5.40330 0.601 0.54792   
## priorityVeryHigh -9.90756 12.25952 -0.808 0.41916   
## num.modifs 0.04607 0.03336 1.381 0.16757   
## num.modules 3.12954 0.55206 5.669 0.0000000179 \*\*\*  
## team.size 12.41606 2.66888 4.652 0.0000036399 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 79.72 on 1230 degrees of freedom  
## Multiple R-squared: 0.0829, Adjusted R-squared: 0.07843   
## F-statistic: 18.53 on 6 and 1230 DF, p-value: < 2.2e-16

3.2 Please provide a brief interpretation of both, the raw and standardized effects on **dev.days** of each, **num.modules** and **team.size**.

**Answer:**

*The effect of num.modules on dev.days is significant at the level of p<0.01. On average, holding everything else constant, when the number of software modules affected by the modification requested for increases by 1, number of days it takes to complete the modification request (MR) increases by 3.13. When standardized, the coefficient of num.modules show that on average, holding the other predictors constant, when the number of software modules affected by the modification requested for increases by 1 standard deviation, the number of days it takes to complete the MR increases by 0.1696 standard deviations.*

*Similarly, the raw effect of team.size on dev.days is significant at the level of p<0.01, showing that on average, holding all other predictors constant, when the number of software developers who worked on the MR as a team increases by 1, the number of days it takes to complete the MR increases by 12.42. The standardized effect show that on average, holding everything else constant, when the size of the team of software developers who worked on the MR increases by 1 standard deviation, the number of days it takes to complete the MR increases by 0.1342 standard deviations.*

**Some Insights** (about the effect of team.size): Can you figure out why larger teams take longer to develop software? You would think that more people in a team speed up the development, but these results support the opposite. Well, as it turns out, this supports Fred Brooks’ (seminal The Mythical Man-Month book) argument that adding members to a software team brings in more productive resources, but it increases exponentially the coordination challenges, which generally end up increasing the development time. Statistically, you could simply say that keeping priority, number of modifications needed, and number of modules affected, team size does not provide incremental benefits, but on the contrary, it is detrimental to software development speed.

## 4. Multivariate Time Series

4.1 Let’s re-do a slightly different specification of the class example with the **HousingStarts.csv** data set to get some practice with **Durbin-Watson** testing and lagging. In this reformulated example, we are predicting house starts in thousands of units, **KUnits**, with use the variable **T** for time (i.e., month sequence 1, 2, etc.). In the class example, we used dummy variables Q1, Q2, etc. to model the quarter, but this time we will use an alternative specification with a categorical variable **Qtr**, which has a value of Q1 for the first quarter, Q2 for the second quarter, etc. In the R script code that follows, I first read the data, fit the model, and show the summary results.

# Done for you  
HousingStarts <- read.csv("../../Dataset/HousingStarts.csv",   
 header = T,   
 sep = ",",   
 stringsAsFactors = T)  
  
lm.KUnits <- lm(KUnits ~ T + Qtr, data = HousingStarts)  
summary(lm.KUnits)

##   
## Call:  
## lm(formula = KUnits ~ T + Qtr, data = HousingStarts)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -92.559 -18.367 3.326 21.473 71.289   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 100.56739 5.70666 17.623 < 2e-16 \*\*\*  
## T 0.01137 0.03069 0.370 0.711   
## QtrQ2 31.85479 6.09060 5.230 0.000000368 \*\*\*  
## QtrQ3 29.16437 6.16555 4.730 0.000003822 \*\*\*  
## QtrQ4 9.03694 6.16624 1.466 0.144   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 34.18 on 241 degrees of freedom  
## Multiple R-squared: 0.1376, Adjusted R-squared: 0.1233   
## F-statistic: 9.614 on 4 and 241 DF, p-value: 0.0000003145

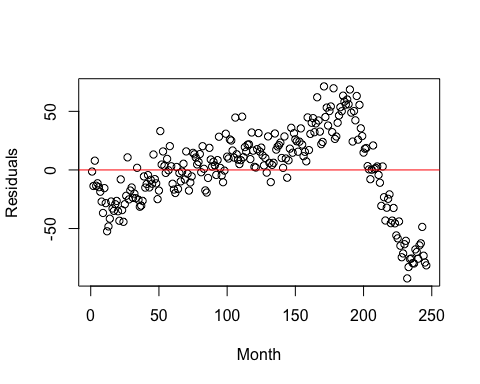
str(HousingStarts)

## 'data.frame': 246 obs. of 9 variables:  
## $ Month : Factor w/ 246 levels "1-Apr","1-Aug",..: 160 149 193 116 204 182 171 127 237 226 ...  
## $ T : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ KUnits: num 99.2 86.9 108.5 119 121.1 ...  
## $ S.P : num 329 332 340 331 361 ...  
## $ Qtr : Factor w/ 4 levels "Q1","Q2","Q3",..: 1 1 1 2 2 2 3 3 3 4 ...  
## $ Q1 : int 1 1 1 0 0 0 0 0 0 0 ...  
## $ Q2 : int 0 0 0 1 1 1 0 0 0 0 ...  
## $ Q3 : int 0 0 0 0 0 0 1 1 1 0 ...  
## $ Q4 : int 0 0 0 0 0 0 0 0 0 1 ...

Now examine the model for the assumption of residual independence. That is, first examine the residual plot to see if you notice a cyclical pattern with the residuals. Plot **HousingStarts$T** against **lm.KUnits$residuals**. Use the xlab = and ylab = parameters to display the X and Y labels shown below. Also use the abline(0, 0) function to draw a horizontal line, but add the col = parameter to make the line red.

Then load the **{lmtest}** library and test for serial correlation with the Durbin-Watson test **dwtest()**.

plot(HousingStarts$T, lm.KUnits$residuals,  
 xlab = "Month",  
 ylab = "Residuals")  
  
abline(0, 0, col = "red")  
  
library(lmtest)



dwtest(lm.KUnits)

##   
## Durbin-Watson test  
##   
## data: lm.KUnits  
## DW = 0.22514, p-value < 2.2e-16  
## alternative hypothesis: true autocorrelation is greater than 0

4.2 Is there serial correlation in the model? Is it positive or negative? Use both, the **residual plot** and the results of the **Durbin-Watson** test to briefly explain why or why not.

**Answer:**

*Yes, there is serial correlation and it is positive. This is indicated by the cyclical pattern over time in the residual plot and the result of the Durbin-Watson is significant at p<0.01 and positive at DW=0.2. This confirms the presence of strong positive correlation.*

4.3 Following the class example, load the **{DataCombine}** library and then use the slide() function to create a new variable called **KUnits.L1** containing the value of KUnits, lagged by one period. Save the new lagged data set with a different name like **HousingStarts.L1**. Then fit a lagged linear model just like the one above (you can copy/edit if you wish), but include the new lagged variable **KUnits.L1** as a predictor, and use the lagged data frame **HousingStarts.L1** instead. Save the results in a linear model object named **lm.KUnits.L1** and display the summary() results of the model.

library(DataCombine)  
  
HousingStarts.L1 <- slide(HousingStarts,   
 Var = "KUnits",   
 NewVar = "KUnits.L1",   
 slideBy = -1)  
  
lm.KUnits.L1 <- lm(KUnits ~ KUnits.L1 + T + Qtr,   
 data = HousingStarts.L1)  
  
summary(lm.KUnits.L1)

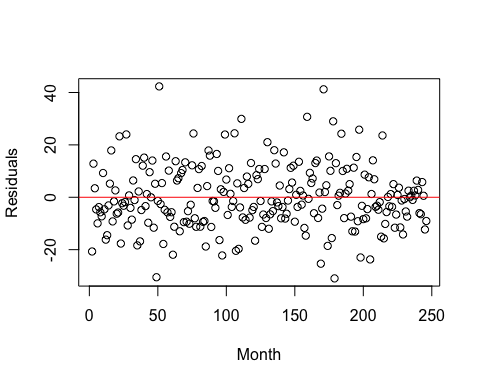
##   
## Call:  
## lm(formula = KUnits ~ KUnits.L1 + T + Qtr, data = HousingStarts.L1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -30.975 -7.910 -1.425 7.162 42.326   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 11.428448 2.979503 3.836 0.00016 \*\*\*  
## KUnits.L1 0.969634 0.023569 41.141 < 2e-16 \*\*\*  
## T -0.008215 0.010918 -0.752 0.45252   
## QtrQ2 -1.036656 2.302820 -0.450 0.65300   
## QtrQ3 -10.652912 2.390878 -4.456 1.29e-05 \*\*\*  
## QtrQ4 -16.678066 2.273843 -7.335 3.44e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 12.07 on 239 degrees of freedom  
## (1 observation deleted due to missingness)  
## Multiple R-squared: 0.8932, Adjusted R-squared: 0.8909   
## F-statistic: 399.6 on 5 and 239 DF, p-value: < 2.2e-16

Then, render a plot with the **residuals** in the vertical axis and time (i.e., **T**) in the horizontal, with the appropriate x and y axis labels shown, and a horizontal red line at 0.

**Technical Tip**: When you lag one period, you lose the first observation because the first observation does not have a lagged value. Consequently, when you plot the residuals, you need to remove the first observation from the time **T** vector. You can accomplish this with the [-1] index right next to **T** shown below, which removes the first observation in the vector.

Also, conduct a **DW test** of the new lagged model **lm.KUnits.L1**.

plot(HousingStarts$T[-1], residuals(lm.KUnits.L1),  
 xlab = "Month",  
 ylab = "Residuals")  
abline(0, 0, col = 'red')



dwtest(lm.KUnits.L1)

##   
## Durbin-Watson test  
##   
## data: lm.KUnits.L1  
## DW = 2.4197, p-value = 0.9989  
## alternative hypothesis: true autocorrelation is greater than 0

4.4 Please indicate if the serial correlation problem was resolved. Use both, the **residual plot** and the results of the **Durbin-Watson** test to briefly explain why or why not.

**Answer:**

*Yes, the serial correlation was resolved. The residual plot does not show any specific pattern. It rather shows a cloud of data. The Durbin-Watson statistic 2.4 which is within the threshold for lack of serial correlation and as observed, the p-value indicates that it is not significant. Hence, we conclude that there is no serial correlation.*

4.5 Please interpret the effect of QtrQ4 and KUnits.L1

**Answer:**

*The effect of Qtr4 is negative and highly significant at the level of p<0.001. On average, holding everything else constant, there are 16.68 thousand fewer units of houses started in the fourth quarter, compared to the first quarter.*

*The effect of KUnits.L1 is positive and also highly significant at p<0.001 level. On average, holding everything else constant, for every one thousand units of houses started in a given period, there are 969 units of houses started in the following period*