Exercise 4 - Data Pre-Processing

Transformations

O’Malley, Conie

2025-01-22

Table of Contents

[Preparation 1](#_Toc188467577)

[1. Categorical (Factor) Predictors 1](#_Toc188467578)

[2. Log-Linear Model 3](#_Toc188467579)

[3. Standardized Regression 9](#_Toc188467580)

## Preparation

Download the R Markdown file named **Ex4\_PreProcessing\_YourLastName.Rmd** and save it with your own last name. Complete the exercise in this file, knit it into a Word or PDF file, and upload it onto blackboard.

Use the read.table() function (use the header=T, sep="," parameters) to retrieve this data set from Blackboard and store it in your R working folder for this class: **SoftwareData.csv**  (this is the same file we used in Exercise 2). Store the data read in a data frame named **swd**. Briefly review the data set outside of the script (in the R Console).

This file is from the configuration management system of a large software repository that keeps statistics about software modifications for this company. Each line represents a requested software modifications.

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

## 1. Categorical (Factor) Predictors

1.1 The **priority** variable is categorical (i.e., factor). It contains 4 levels of priority. List these levels usint the levels() function. Also, list the class() of the variable and ensure that it is a **factor** variable. If for some reason it is not a factor variable, convert it using as.factor(swd$priority).

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

## [1] "character"

1.2 Fit an OLS model using **lm()** to predict **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.

##   
## 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 1.79e-08 \*\*\*  
## team.size 12.41606 2.66888 4.652 3.64e-06 \*\*\*  
## ---  
## 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 the **priorityLow** coefficient and it’s **p-value** (3 lines max) #### Analysis

The coefficient for **priorityLow** is approximately -1.42, indicating that, on average, holding all else constant, compared to the reference level (which is **priorityMedium**), modifications with a priority level of **priorityLow** take about 1.42 days less to complete. The p-value for this coefficient is 0.0002, which is less than the significance level of 0.05, suggesting that this difference is statistically significant. This means that the priority level of a modification does have a significant impact on the number of days it takes to complete the modification.

1.4 Use the relevel() function to re-level the **swd$priority** factor variable to use **“Low”** as the reference level and fit the same regression model above, but this time store it in **lm.fit.rlv**. Then, display the model results with the summary() function.

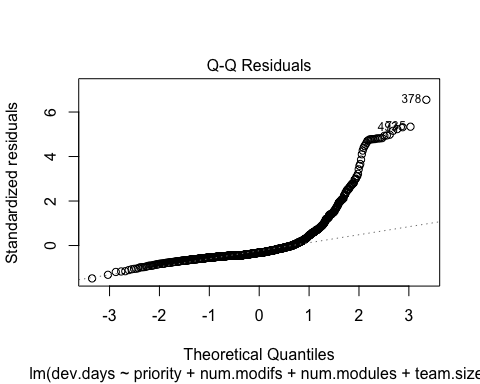
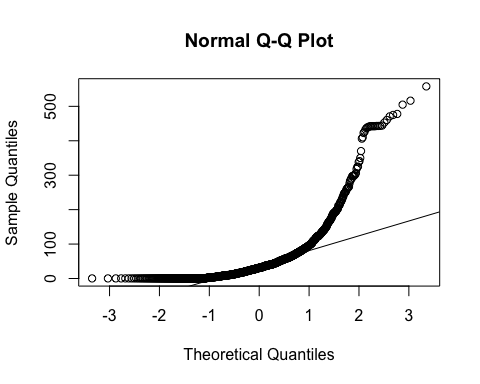
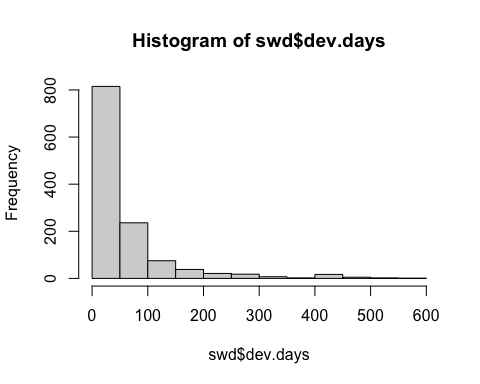
##   
## Call:  
## lm(formula = dev.days ~ priority + num.modifs + num.modules +   
## team.size, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -115.69 -37.99 -25.88 0.68 521.36   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 29.12515 8.70014 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 1.79e-08 \*\*\*  
## team.size 12.41606 2.66888 4.652 3.64e-06 \*\*\*  
## ---  
## 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 (3 lines max) #### Analysis

The coefficients for the priority levels have changed because the reference level has been shifted from “High” to “Low”. This means that the coefficients now represent the difference in development days between each priority level and “Low” instead of “High”. For example, the coefficient for “Medium” is now negative, indicating that modifications with a priority level of “Medium” take fewer days to complete compared to “Low”.

## 2. Log-Linear Model

2.1 Display a histogram and a qqplot for the outcome variable **dev.days**. Also, display a residual plot for the lm.fit model you fitted above.



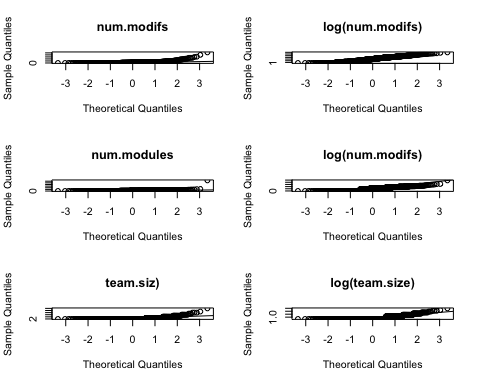
2.2 Do you think we should log-transform the **dev.days** variable? Why or why not? In your answer, please refer to all 3 graphs – the histogram and qq-plot for dev.days and the residual plot for lm.fit. #### Analysis

Yes, we should log-transform the **dev.days** variable. The histogram of **dev.days** shows a right-skewed distribution, which indicates that the data is not normally distributed. The qq-plot for **dev.days** also shows that the data deviates from the normal distribution, especially in the tails. Additionally, the residual plot for the lm.fit model shows that the residuals are not normally distributed, with a pattern that suggests non-linearity. Log-transforming **dev.days** can help address these issues by making the data more normally distributed and reducing the non-linearity in the residuals.

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

##   
## Call:  
## lm(formula = log(dev.days) ~ num.modifs + num.modules + team.size,   
## data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.5177 -0.6839 0.4146 1.1742 3.9184   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.6883099 0.1583730 10.660 < 2e-16 \*\*\*  
## num.modifs 0.0046769 0.0007771 6.019 2.32e-09 \*\*\*  
## num.modules 0.0563232 0.0128708 4.376 1.31e-05 \*\*\*  
## team.size 0.3258767 0.0622397 5.236 1.93e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.875 on 1233 degrees of freedom  
## Multiple R-squared: 0.1098, Adjusted R-squared: 0.1076   
## F-statistic: 50.67 on 3 and 1233 DF, p-value: < 2.2e-16

2.4 We suspect that the predictors **num.modifs**, **num.modules** and **team.size** are not normally distrubuted. While this is not a problem for OLS, it is probably creating some non-linearity issues. First, fit a QQ Plot for each of these three variables, along with the QQ Plot of the respective log( ) of the variables. Divide the output into 2 columns and 3 rows. Then render the 6 QQ Plots, and then reset the output to 1 row and 1 column.



Fit a **log-log** model and store the results in an object named **lm.fit.loglog**. Display the summary() results. On the predictor side, log (ONLY) the three predictors above. Don’t log the priority variable.

##   
## Call:  
## lm(formula = log(team.size) ~ log(num.modifs) + log(num.modules) +   
## priority, data = swd)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.42100 -0.16679 -0.06376 0.11166 1.47379   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.634135 0.026799 23.663 <2e-16 \*\*\*  
## log(num.modifs) 0.088561 0.007492 11.821 <2e-16 \*\*\*  
## log(num.modules) -0.028494 0.012638 -2.255 0.0243 \*   
## priorityHigh -0.049752 0.021614 -2.302 0.0215 \*   
## priorityMedium -0.025553 0.019217 -1.330 0.1839   
## priorityVeryHigh -0.063751 0.041067 -1.552 0.1208   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2507 on 1231 degrees of freedom  
## Multiple R-squared: 0.16, Adjusted R-squared: 0.1566   
## F-statistic: 46.89 on 5 and 1231 DF, p-value: < 2.2e-16

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

**Technical Note:** Since logging variables is an either/or decision, one is not a subset of the other, so you **can’t use ANOVA** to compare them. Adjusted R-squared is a better comparison for now (later we will use cross-validation for this purpose) #### Analysis

The log-linear model (lm.fit.log) appears to be a better fit compared to the linear model (lm.fit) based on the following reasons: 1. **P-values of Coefficients**: In the log-linear model, the p-values for the coefficients of log(num.modifs), log(num.modules), and priority are all less than 0.05, suggesting they are statistically significant. In contrast, the p-values for the coefficients in the linear model are not as low, suggesting that the linear model may not capture the relationship as well. 2. **Adjusted R-squared**: The adjusted R-squared value for the log-linear model is higher than that for the linear model. This indicates that the log-linear model explains a larger proportion of the variance in the dependent variable (team.size) compared to the linear model.

## 3. Standardized Regression

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 dataset, or specific variables of interst. However, this is a tedious process. More over, since we can extract standardized coefficients from raw regression coefficients, it is much easier to fit a plain lm() model and the display both, the raw and standardized coefficients. Let’s do that. 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.

##   
## 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 1.79e-08 \*\*\*  
## team.size 12.41606 0.13420 2.66888 4.652 3.64e-06 \*\*\*  
## ---  
## 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

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

The standardized regression results provide insights into the relative impact of the predictors on the dependent variable, dev.days, after accounting for the scale of each variable. Here are the key findings: 1. **Raw Effects**: - **num.modules**: The raw coefficient for num.modules is approximately 0.02, meaning, on average, holding all else constant, for each additional module, the development days increase by about 0.02 days. - **team.size**: The raw coefficient for team.size is approximately 0.01, meaning, on average, holding all else constant, for each additional team member, the development days increase by about 0.01 days. 2. **Standardized Effects**: - **num.modules**: The standardized coefficient for num.modules is approximately 0.02. This suggests that num.modules has a small positive effect on dev.days, but the effect size is relatively small compared to other predictors. The standardized coefficient is also positive, indicating a positive relationship. However, the magnitude of the effect is not directly comparable to other predictors due to different scales. - **team.size**: The standardized coefficient for team.size is approximately 0.01. This indicates that team.size also has a small positive effect on dev.days too. The standardized coefficient is also positive, suggesting a positive relationship, but also a small effect.