ITEC 621 - Homework 1

R Basics and Foundations

Mohamed Seifeldin

5th Feb 2024

Abstract

This Quarto file contains Homework 1 for ITEC 621, which includes R refresher, statistical foundations and OLS regression modeling.

Table of contents

## Overview – Read this carefully

The goal of this homework is to practice a bit more with R, R Studio, Quarto and with simple statistical analysis. Open the Quarto template file **HW1\_YourLastName.Qmd**, re-name it using **your actual last name** and copy over your work to the corresponding code chunk sections in the Quarto template.

## Rendering (up to 10 pts.)

You are required to **render ALL** your Quarto homework files into a **Word** (preferred) or PDF file. Learning how to build your models in R and report your analysis and results in the same document conduct, which is what Quarto enables you to do is an important learning objective of this course. You are expected to submit your homework in a properly rendered document with **business-like** formatting and appearance. **No rendering, inadequate rendering and/or improper formatting of the document will carry point deductions up to 10 points.**

**Important Notes about Rendering and Formatting:**

* Your **R code** must be **visible** in your rendered document. This means that your Quarto file **MUST** have the attribute echo: true in YAML so that we can evaluate your R code. The template provided for the homework usually has the echo: true setting, but it is your responsibility to ensure that it is set correctly.
* The knitted file must have a table of contents that include all Heading 1 (#) and Heading 2 (##) entries. Please review your Quarto file to ensure that these headings are the **only** text with # or ## tags. Including these hash tags anywhere else will cause your narrative text to be be improperly formatted (with large blue font) and the text will also appear in the table of contents, which is not appropriate for a business document. **Technical note:** the YAML attribute toc: true instructs Quarto to generate a table of contents. The YAML attribute toc-depth: 2 instructs Quarto to include in the table of contents all text prefix with # and ##.
* Enter your narrative answers to interpretation questions in the text areas (without # tags), where **Answer:** is noted, not in the R code chunks. It is OK to enter text in the R code chunks with a # tag, but these should be used to make comments and annotations about your script, not for interpretations or other report narratives.
* Also note the YAML attribute code-overflow: wrap above. This attribute makes your code and # marked text to wrap. If you omit this or if you use code-overflow: scroll the text will extend beyond the right margin, which is not what you want in a report.
* Overall, anything that would not be acceptable to a business or client audience is not acceptable in rendered documents for this class.

## Interpretations:

The goal in this course is NOT to make you proficient in R, although you will get a lot of R practice in this class. The main goal is for you to develop the ability to extract meaningful business insights from your analysis. As such, all **interpretation questions** will be graded rigorously in every homework. Please think through every interpretation question and respond concisely, but accurately. Your analysis must demonstrate that you understand how to interpret the output of your models.

## Submission:

I will always display the solution output (not the code) in the homework instructions, so that you can compare your results against the solution. In questions involving random sampling, your outputs may differ slightly from the solution. This is OK, but if in doubt, please ask a TA or me. Once done, submit your rendered document in Canvas.

## Q1. Functions (10 pts.)

Write a function to compute and return() the normalized values of a vector using the **Min-Max** method.

**Technical Note:** Certain predictive modeling methods require that we normalize the data to a 0-1 scale. The **Min-Max** method is a simple normalization method that transforms the data to a 0-1 scale and it is one of the most popular normalization methods to train **neural network** models (see Ch.11 in the PAML4B book). We will illustrate this method with a vector of values. The transformation consists in taking each value and subtracting from it the minimum value in the vector and then dividing that value by the range of the data, which is the maximum value minus the minimum value. For example, take a vector x and subtract from it min(x) and then divide that by max(x) - min (x).

You first need to **define** the function, before you can use it:

Define function named **normalize** and assign to it normalize <- function(x) {. This is the first line of the definition of a function named normalize(). The function will be applied over any value of **x** you you enter in the function. The open bracket { denotes the begining of the function definition. The next line needs to contain the value the function will return (note: complex functions have many lines, we are only writing 1 line here for simplicity). Indent the second line a couple of spaces and type return(). Inside the return function, enter the Min-Max function (x - min(x)) / (max(x) - min(x)). In the next line, close the function definition with a close bracket }. Your function has now been defined and it is ready to be used.

**normalize <- function(x) {return((x - min(x)) / (max(x) - min(x)))}**

**vector <- c(2, 5, 10, 8, 4) normalized\_vector <- normalize(vector) print(normalized\_vector)**

In the next line, create a vector using the c() function, c(4, -3, -4, 1, 5, 12, 7) and save it in a vector named **MyVect**. Then normalize the values in this vector using the function we just created, normalize(MyVect) and store the results in a new vector named **MyVect.n**. This will cause the values in **MyVect** to replace the generic **x’s** in the function definition and return the respective Min-Max values and store them in a new vector named **MyVect.n**.

Then use the cbind() function to display MyVect and MyVect.n side by side. Notice that the Min-Max value of the smallest raw value is 0, the largest is 1 and everything else is between 0 and 1.

**normalize <- function(x) { return((x - min(x)) / (max(x) - min(x)))} MyVect <- c(4, -3, -4, 1, 5, 12, 7) MyVect.n <- normalize(MyVect) result <- cbind(MyVect, MyVect.n) print(result)**

## Q2. Data Work (10 pts.)

Let’s analyze the **mtcars** in the library {datasets}. This library loads by default when you start R, so there is no need to load in in your script. The data set is there already. You can inspect the variables in this data set with ?mtcars (notice in the template that I set the parameter #| eval: false to prevent the code chunk from running when rendering your document).

# Done for you  
?mtcars

2.1 Display the object class() for the **mtcars** data frame and for the vectors **mpg** (i.e., mtcars$mpg), **cyl** and **am**.

**class\_mtcars <- class(mtcars)**

**print(class\_mtcars)**

**class\_mpg <- class(mtcars$mpg)**

**print(class\_mpg)**

**class\_cyl <- class(mtcars$cyl)**

**print(class\_cyl)**

**class\_am <- class(mtcars$am)**

print(class\_am)2.2 The **am** (automatic-manual) variable is numeric and it contains the values 0 and 1 for automatic and manual, respectively, which is a bit cryptic. It is best to convert these values to “factor” and use more representative values. Use the as.factor() function and pass the value ifelse(mtcars$am == 0, "Auto", "Manual") function. The ifelse() function will create the text values of “Auto” and “Manual” for 0 and 1 respectively, and the as.factor() function will convert the results into factors. Save the results in an object named mtcars$am.f, which will add this vector as a new column in the **mtcars** data frame. Then display the class of this new variable. Then display the first six values of mtcars$am.f using the head() function. Mtcarsam == 0, “Auto”, “Manual”))

**mtcars$am.f <- as.factor(ifelse(mtcars$am == 0, "Auto", "Manual"))**

**class\_am\_f <- class(mtcars$am.f)**

**print(class\_am\_f)**

**head\_am\_f <- head(mtcars$am.f)**

**print(head\_am\_f)**

Notice that the display now shows the levels Auto and Manual for the factor variable.

2.3 Then create a matrix called **mtcars.mat** that contains only some of the quantitative variables. Use the as.matrix() function and feed these quantitative variables as.matrix(mtcars[ ,c("mpg", "cyl", "disp", "hp", "wt")]). Then display the class() of the **mtcars.mat** object and then list the first 6 rows of this matrix using the head() function, just to ensure you did the right thing.

**mtcars.mat <- as.matrix(mtcars[, c(“mpg”, “cyl”, “disp”, “hp”, “wt”)]) class\_mtcars\_mat <- class(mtcars.mat) print(class\_mtcars\_mat) head\_mtcars\_mat <- head(mtcars.mat) print(head\_mtcars\_mat)**

## Q3. Descriptive Statistics (10 pts.)  
  
Let's analyze the data quantitatively. First get a`summary()` of the \*\*mtcars\*\* data frame and inspect the frequencies. Then load the `{psych}` library and display the descriptive statistics for the data set using the `describe()` function, but only for the first 8 variables of \*\*mtcars\*\* and the first 5 columns of descriptive statistics, `(mtcars)[1:8, 1:5]` (you can inspect all variables and additional statistics on your own).   
  
**summary\_mtcars <- summary(mtcars)  
print(summary\_mtcars)  
describe\_mtcars <- describe(mtcars[, 1:8])  
print(describe\_mtcars)**  
## Q4. Correlation Analysis (10 pts.)  
  
4.1 Then create a correlation object named \*\*mtcars.cor\*\* using the `cor()` function on the \*\*mtcars.mat\*\* matrix. Then load the \*\*{corrplot}\*\* library and feed this \*\*mtcars.cor\*\* object into the `corrplot()` function. Add the parameter `"order = hclust"` to group the cluster the variables by correlation strength, and the parameters `method = number` to display correlation values. Then run the same `corrplot()` function, but this time use `method = ellipse` to get a graphical display.  
  
**mtcars.cor <- cor(mtcars.mat)  
corrplot(mtcars.cor, method = "number", order = "hclust")  
corrplot(mtcars.cor, method = "ellipse")**  
4.2 Based on the correlation results above, suggest the 2 most promising predictors of \*\*mpg\*\*. Briefly describe why did you select these and why you excluded the other 2.  
  
\*\*Answer:\*\*   
  
**I would say WT and CY becasue they both have strong negative correlations with mpg**  
  
## Q5. Descriptive Analytics: Normality (10 pts.)  
  
5.1 Divide the graph output to 1 row and 2 columns (`par(mfrow = c(1, 2))`). Then draw a histogram for the \*\*mpg\*\* variable. Title your diagram `"Miles per Gallon Histogram"` and label the x axis `"Miles per Gallon"`.   
  
Then draw a \*\*QQ Plot\*\* to inspect the normality of this variable (tip: this is a 2 step process; first draw the QQ Plot with the `qqnorm()` function and give it a main title of `"Miles per Gallon QQ Plot"`), then draw the QQ line with the function `qqline()`.  
  
Also, draw a histogram and a QQ Plot for the \*\*wt\*\* variable. Title the histogram \*\*Weight Histogram"\*\* and label the x axis \*\*1000 lbs\*\*. Title the QQ Plot \*\*Weight QQ Plot"\*\*. Then reset the graph output to 1 row and 1 column.  
  
**par(mfrow = c(1, 2))  
hist(mtcars$mpg, main = "Miles per Gallon Histogram", xlab = "Miles per Gallon")  
qqnorm(mtcars$mpg, main = "Miles per Gallon QQ Plot")  
qqline(mtcars$mpg)  
hist(mtcars$wt, main = "Weight Histogram", xlab = "1000 lbs")  
qqnorm(mtcars$wt, main = "Weight QQ Plot")  
qqline(mtcars$wt)  
par(mfrow = c(1, 1))**

5.2 Briefly answer: Do miles per gallon and weight appear to be normally distributed? Why or why not.

**Answer:**  
**Based on the histogram as well as the plot, it would seem that they are not normally distributed. Thew weigth histogram shows variation by weight while the plot shows multiple outliers.**

## Q6. Descriptive Analytics: Boxplots and ANOVA(10 pts.)

6.1 Divide the graph output to 1 row and 2 columns. Then draw 2 boxplots: (1) one for **mpg** by **am.f** using ylab = "Miles per Gallon" and xlab = "Transmission"and (2) another for **wt** by **am.f** using ylab = "1000 lbs. Weight" and xlab = "Transmission". Then reset the graph output back to 1 row and 1 column.

**par(mfrow = c(1, 2)) boxplot(mpg ~ am.f, data = mtcars, ylab = “Miles per Gallon”, xlab = “Transmission”) boxplot(wt ~ am.f, data = mtcars, ylab = “1000 lbs. Weight”, xlab = “Transmission”) par(mfrow = c(1, 1))**

6.2 Then conduct two \*\*ANOVA\*\* tests using the `aov()` function, one to evaluate if \*\*mpg\*\* (mileage) varies by \*\*am.f\*\* (transmission) and another to evaluate if \*\*wt\*\* (weight) varies \*\*am.f\*\* (transmission). Store the results of the first \*\*ANOVA\*\* test in an object named \*\*aov.mpg\*\* [not aov.brand] and the second one named \*\*aov.wt\*\*. Then display the summary of each of these objects, but write the function `cat("\n")` in between the two summaries to separate the displays with a blank line. \*\*Technical note:\*\* the `cat()` function concatenates and prints strings and `"\n"` is the code for a new line.  
  
**aov.mpg <- aov(mpg ~ am.f, data = mtcars)  
summary(aov.mpg)  
cat("\n")  
aov.wt <- aov(wt ~ am.f, data = mtcars)  
summary(aov.wt)**

6.3 Briefly answer: Does car mileage vary by transmission? And, does weight vary by transmission? Briefly explain why or why not. Also, comment on Which type of transmission has better mileage and which has more weight. Please refer to **both**, the visual boxplot and the quantitative ANOVA output.

**Answer:**

**Based on both visual inspection and quantitative analysis, manual transmission cars tend to have better mileage, while automatic transmission cars tend to have more weight.**

## Q7. Simple Linear Regression Model (10 pts.)

7.1 Fit a **simple** linear regression model object with the lm() function to predict **mpg** using **wt** as the only predictor. Store your linear model results in an object named **fit.simple**. Then display the summary() results.

**fit.simple <- lm(mpg ~ wt, data = mtcars) summary(fit.simple)**

7.1 Provide a brief \*\*interpretation\*\* of both, the \*\*significance\*\* and \*\*effect\*\* of weight on car mileage.   
  
\*\*Answer:\*\*   
  
**The weight variable is a statistically significant predictor of car mileage in this simple linear regression model. The negative coefficient indicates that an increase in weight is associated with a decrease in predicted mileage.**  
  
## Q8. Linear Regression Model with a Binary Predictor (10 pts.)  
  
8.1 Now fit a larger linear regression model, same as above, but add \*\*am.f\*\* as a predictor. Name the resulting linear model \*\*fit.dummy\*\*. Then display the `summary()` results.   
  
**fit.dummy <- lm(mpg ~ wt + am.f, data = mtcars)  
summary(fit.dummy)**

8.2 Provide a brief **interpretation** of the effect of **am.f** on **mpg**.

**Answer:**  
**Am.f is not a statistically significant independent variable on mpg.**

## Q9. Multivariate Linear Model (10 pts.)

9.1 Now fit a larger linear regression model to predict **mpg**, using **cyl**, **wt** and **am.f** as predictors. Name the resulting linear model **fit.large**. Then display the summary() results.

**fit.large <- lm(mpg ~ cyl + wt + am.f, data = mtcars) summary(fit.large)**

9.2 Then provide a brief \*\*interpretation\*\* of both, the \*\*significance\*\* and \*\*effect\*\* of all three predictors. If any predictor is not significant you don't need to elaborate, just say "has no significant effect".  
  
\*\*Answer:\*\*   
**cyl=significant where the for every increase in cyl values there will be a decrease in mpg  
wyt= same**  
  
  
9.3 Briefly answer: car mileage was significantly different between automatic and manual transmission cars in the ANOVA test in 6.2 above, but its effect is not significant in the two regression models in 8.2 or 9.2. How can you explain these contradictory results?  
  
\*\*Answer:\*\*   
  
**The difference in significance between the ANOVA test and the regression models can be attributed to the inclusion of additional predictors in the regression models. The ANOVA test simply examines the overall mean differences between groups, whereas the regression models consider the individual contributions of each predictor while accounting for other variables. It's possible that when including cyl and wt in the models, the effect of am.f on mpg becomes statistically non-significant, suggesting that the observed differences in means may be explained by other variables in the model.**  
  
## Q10. Residual Plots and Model Evaluation (10 pts.)  
  
10.1 Let's inspect the results and provide some final storytelling. First, `plot()` the \*\*fit.large\*\* object. This function yields 4 residual plots, but for now, we are only interested in the second residual plot, so add the attribute `which = 2`, which renders the QQ Plot of the residuals.   
  
**plot(fit.large, which = 2)**

10.2 Then conduct an **ANOVA** test to compare all 3 models together, \*\*fit.simple, fit.dummy and fit.full.

**models <- list(fit.simple, fit.dummy, fit.large) model\_names <- c(“fit.simple”, “fit.dummy”, “fit.large”) anova\_table <- anova(models[[1]], models[[2]], models[[3]], test = “F”) print(anova\_table)**

```

10.3 Which of the three models is preferred? Briefly explain why.

**Answer:**

**With the current variables in place, the only model that was stastitacly significant was model 3 with cyl, wt, and am.f.**