Towards Data Mining – Exercise 5: missing data

To pass this exercise, you need to follow these guidelines and submit answers to questions that are underlined into Moodle.

Most of the needed code is given to you (ex5_2020_code.R).

You can get help on any of the commands in R by typing *?commandname*. You can refer to a variable in a data.frame with the dollar sign: *my.data.frame$my.variable*. You might find the first weeks exercise also helpful.

Missing data

1. Copy the exercise files from Moodle to your computer. Start RStudio (or plain R) and start a new project in the same folder where you copied the data files. Import the file "iris.csv" with the *read.table* function. Remember to check that you are using the right field and decimal separator characters as well as treating the first line as a header (check the data file format with e.g. notepad). What field separator character was used? How about decimal separator?
2. Plot the data so that Petal.Length is on the x-axis and Petal.Width is on the y-axis. It should be a scatter plot.
3. Fit a linear regression model to the data. This can be done with the *lm* command. *lm* uses the formula interface to describe the model to be fit. In this case, the formula should look something like this: *lm.fit <- lm(Petal.Width ~ Petal.Length, data = my.data.frame).* After fitting the regression model, you can see details of the model by typing: *summary(lm.fit).* What is the intercept coefficient of the regression line?
4. Add a regression line to the plot. You can use the function *abline*. Your model parameters can be found in *lm.fit$coefficients.*
5. Calculate the mean for both variables using the *mean* function. Why does the function return *NA*? Try looking at *summary(my.data.frame)*. How many missing values are there in each variable? See *?mean* on how it handles missing values.
6. Notice that *lm* (should have) worked without problems on the same data set. How does *lm* handle missing values (see *?lm*)? Tip: you will need to use *getOption* (see ?options) to check the settings used on your computer.
7. How many complete cases are there in the data set? *complete.cases* function is a tool you can use. Tip: *sum* can be used with Boolean values.
8. Calculate now the means of each variable by ignoring any missing values. Save the values in variables for later. To save the result of an expression and print it in one command, enclose the expression in parenthesis. E.g. *"(variable.name <- mean(values))"* (without the quotes).
9. What happens if you try to calculate the covariance (*cov*) of the variables? Change the value of the parameter *use* so that calculation is possible. Which option did you choose? What's the difference between *complete.obs* and *pairwise.complete.obs* options (in general, here they give identical result)?
10. Calculate the mean of each variable in each of the following scenarios (because there are only a few missing values, you can do the imputation manually. You can e.g. make a copy of the original data.frame for each scenario (*mean.imputation <- my.data.frame*) and manually fill in the imputations (*mean.imputation$Petal.Width[index.of.missing] <- petal.width.mean*)):
    a. Using complete data only (done above)
    b. After imputing the mean. What is the mean of Petal.Length after mean imputation?

c. After imputing a stratified mean (mean of the complete observations that belong to the same class, i.e. Species. Use e.g. *subset* for the stratification). What is the mean of Petal.Length after stratified mean imputation?
d. After imputing with regression imputation (with the *lm.fit* model from above, use *predict(lm.fit, data.frame.to.predict)* where *data.frame.to.predict* contains the rows where *Petal.Width* is missing. Build a model to predict *Petal.Length* when *Petal.Width* is known and use *predict()* again for the missing *Petal.Length* values). What is the mean of Petal.Length after regression imputation?
e. By using the maximum likelihood estimates for the missing values (instructions below). What is the mean of Petal.Width after ML imputation? How about Petal.Length?
f. By using multiple imputation (instructions below). What is the mean of Petal.Width after multiple imputation? How about Petal.Length?

11. ML estimation. See the lecture slides for the principle.
    a. Install the *mclust* package with the *install.packages* command.
    b. Load the package with the *library* command.
    c. Use the complete cases of the data to estimate the probability densities using the *densityMclust* function. Note: Use only the columns Petal.Length and Petal.Width. See an example of how to do that from the first plot command below.
    d. You can plot the density maps with the following commands (assuming your density object is called *dens* and your training data is in data.frame called *trainingdata*):
        i. plot(dens, what = "density", data = trainingdata[,c(1,2)])
        ii. plot(dens, what = "density", type = "persp")
    e. To get the ML estimate for Petal.Width when Petal.Length is known (this is simplified just to make this example understandable), calculate the likelihood for different values of Petal.Width while keeping the Petal.Length value fixed (repeat for each missing value):
        i. widths <- seq(0,4,0.1)
        ii. lengths <- rep(iris$Petal.Length[index.of.missing], length(widths))
        iii. seq(0,4,0.1)[which.max(predict(dens, data.frame(Petal.Length = lengths, Petal.Width = widths)))]
    f. To get the ML estimate for Petal.Length when Petal.Width is known, do the same but now keeping the Petal.Width value fixed. Notice that you should use a different range for the Petal.Length values than you did for the Petal.Widths.

12. Multiple imputation.
    a. Install and load package "*mice*".
    b. Use the default imputation methods and do 10 imputations. Seed is used for reproducibility. *imp <- mice(my.data.frame, m = 10, print = FALSE, seed = 121018)*
    c. Calculate the pooled estimate for the mean

```
m <- imp$m
Q <- rep(NA, m)
U <- rep(NA, m)
for (i in 1:m) {
  Q[i] <- mean(complete(imp, i)$Petal.Width)
  U[i] <- var(complete(imp, i)$Petal.Width) / nrow(my.data.frame)
}
est.pw <- pool.scalar(Q, U, n = nrow(my.data.frame), k = 1)
```

    d. The pooled estimate for the mean can be found in*: est.pw$qbar.* You can also find the error estimates from *est.pw.* See the lecture slides to see what they mean.

    e. Repeat for Petal.Length

13. Compare the mean estimates gotten with each method with the complete data. It is in file *iris_complete.csv.*