EC3305 Programming Tools for Economics

SEMESTER II 2020-2021

Group Project Report

Group: W01-03

Andy Low Wei Liang [redacted]

[redacted]

Project Report

Objective: To assign brand names to a list of automobile names.

Before we started on the project, we first had to perform exploratory data analysis on both of *model_and_brand.csv* and *autoswithout.csv*. Exploratory data analysis is important in providing a gauge of how the datasets look like and check for any abnormalities, guiding our steps ahead.

Exploratory Data Analysis

- **model_and_brand.csv**
  - *brands*
    - 39 unique brands excluding NA, 2 observations of NA
    - All lower case with no special characters
  - *models*
    - 203 unique models excluding NA, 31 observations of NA
    - All lower case with '_' being the separator, absence of brand in model
- **autos_without.csv**
  - *name*
    - 4759 observations of names to be tagged to a brand, 0 observation of NA
    - Mixed case with special characters, lack of special characters as affixes

Problems and solutions

Problem 1: Similarity between merce and mercedes_benz

In this particular case, "merce" only has one model, NA. In addition, there is an absence of brands in model names. Hence, when we iterate through *autoswithout$name*, if "merce" is present, we can safely assign "mercedes_benz" instead.

Problem 2: Models belonging to more than one brand (1_reihe, 3_reihe, 5_reihe, andere)

Instead of randomly assigning brands to duplicate models, our group would like to eliminate such "wrongly-tagged" possibility. As there is no way for us to differentiate between such models, we assigned 'NA" to allow for manual processing by the informed to ensure accuracy.

Hence, we created a data frame called *unique_models_df* counting the number of occurrences of a particular model name in *model_brand*. This allowed us to identify problematic models.

Problem 3: Common abbreviations of brands present in *autowithout$name* ("Mercedes", "VW")

For abbreviations that we are able to identify, we assigned appropriate matches from brand to abbreviation.

Methodology Overview

Step 1: Creating *unique_models_df* from the list of *unique_models* excluding NA and a list of 0s (Line 33 to 37)

Step 2: Iterate through *model_brand* counting the number of occurrences of a model name and updating *unique_models_df* accordingly (Line 45 to 47)

Step 3: Removing rows of problematic models from *model_brand* and naming it *unique_models_df_final* (Line 50 to 55)

Step 4: Iterate through *autowithout$name*, logging names with assigned brands in *assigned_brand* and unassigned ones to *unassigned_name* (Line 67 to 114)

1. Check if *name* matches any *brand* in *unique_brands*
    a. If *name* matches "merce", log "mercedes_benz"
    b. If else, *name* matches *brand*, log matching *brand*
2. Check if *name* matches "Mercedes", the short form of "mercedes_benz"
    a. If *name* matches, log "mercedes_benz"
3. Check if *name* matches "Vw", the short form for "volkswagen"
    a. If *name* matches, log "volkswagen"
4. Check if *name* matches any *model* in *unique_models_df_final*
    a. If *name* matches, log *brand* of *model*
    b. Else, log NA and log *name* into *unassigned_name* (All models identified to be problematic in *unique_models_df* are in this category)

Step 5: *returnNA* data frame is created by converting *unassigned_name* into a dataframe, innerjoin() with *autoswithout* to get the whole data frame (Line 120 to 123)

Method Evaluation

- In reality, datasets will be way larger and it will become significantly harder to expect manual tagging for such "special" cases identified in Problem 2 by informed personnel.
  In such cases, regular expressions can be utilised with a more extensive *model_and_brand.csv* to conduct unigrams, bigrams and even trigrams on *autoswithout$name* after tokenizing by '_', to produce only one outcome for brands.
- For shorter model names, there is a tendency of brand mistag. However, we cannot limit model names to (*_model | model_*) as models may not exist as a token on it's own. Hence, with extensive information, a better method can be conceived.
- In cases whereby the special characters in *autoswithout$name* breaks up the string of the brand or name itself (affixes), more preprocessing will be required such as substituting whole tags,
 with ""
- Language proficiency in German may be helpful for the dataset as we realised later on that the "special" cases such as *andere* and *reihe* refers to "other" and "line" respectively.

EC3305 Project Flowchart

To load given data

```
autoswithout = read.csv( "autoswithout.csv" )
model_brand = read.csv( "model_and_brand.csv" )
```

To convert model_brand$model and model_brand$brand to character

```
model_brand$model = as.character( model_brand$model )
model_brand$brand = as.character( model_brand$brand )
autoswithout$name = as.character( autoswithout$name )
```

Obtain the unique brands

```
unique_brands = unique( model_brand$brand )
unique_brands = unique_brands[ !is.na(unique_brands ) ]
```

We obtain the unique values of brand from the model_brand.

Create a dataframe to store unique values of model_brand$model

```
unique_models = unique( model_brand$model )
unique_models = unique_models[ !is.na( unique_models ) ]
count = rep( 0, length( unique_models) )
unique_models_df = data.frame( unique_models,count )
colnames( unique_models_df ) = c( "model", "count" )
```

Dataframe contains all unique values of model_brand$model and a count column. We set the count to 0 initially for all models before we check for the actual count.

Generate counts for each unique models in the unique_models_df

```
model_brand =na.omit(model_brand)
for ( i in 1:nrow( model_brand ) ) {
  unique_models_df[unique_models_df$model == model_brand$model[i], 2 ] = unique_models_df[unique_models_df$model == model_brand$model [ i ], 2 ] + 1
}
```

Using na.omit, we remove the NA in model_brand. For each value of i, we equate a model from model_brand to the first column of unique_models_df. This outputs a logical vector with the same number of rows as unique_models_df, with TRUE indicting a match and FALSE indicating no match. This logical vector is then used to subset unique_models_df and we further subset out the count column, with these rows representing models that matches with ith model in model_brand. Thus, we reassign their count value by adding 1.

Removing models that appear more than one time

```
unique_models_df_check = unique_models_df [ unique_models_df$count  <  2,  ]
```

We dropped models with count more than one since they can have different brands assigned to them.

Match unique models to brand

```
library( dplyr )
unique_models_df_final = inner_join( unique_models_df_check, model_brand, by = "model"
)
```

We matched the unique models with their respective brand using inner_join

Create prediction column

```
assigned_brand = c()
unassigned_name = c()
```

We create two columns assigned_brand and unassigned_name.

EC3305 Project Flowchart

Creating "returnNA" dataframe

```
## This will be our predictions column
assigned_brand = c()
## To track unassigned names
unassigned_name =c()
```

Here we create the returnNA using inner_join by joining unassigned to autoswithout since unassigned contains observations which we fail to find a brand for.

For loop to assign brand for name column from autoswithout

```
for( names in autoswithout$name ) {
  check_brand = FALSE  #boolean check for brand
  check_model = FALSE  #boolean check for model
```

Here we create two boolean variables to indicate a match in different part of the for loop.

Assignment of brand by unique_brands

```
for( brands in unique_brands ) {
    if (grepl( brands,names, ignore.case = TRUE ) ) {
      check_brand = TRUE
      if( brands == "merce" ) {
        assigned_brand = c( assigned_brand, "mercedes_benz" )
        break
      } else {
        assigned_brand = c( assigned_brand, brands )
        break
      }
    }
  }
```

This nested for loop searches for unique brand names inside autoswithout$name. If there is a match, the unique brand name is assigned to the corresponding names and the boolean variable is assigned TRUE.

Assignment for "mercedes" and "vw"

```
if( check_brand == FALSE) {
    if( grepl( "Mercedes", names,ignore.case = TRUE)) {
      assigned_brand = c( assigned_brand, "mercedes_benz")
      check_brand = TRUE
    } else if( grepl( "VW", names, ignore.case = TRUE)) {
      assigned_brand = c( assigned_brand, "volkswagen" )
      check_brand = TRUE
    }
  }
```

Here we manually search for "mercedes" manually since some of the cars are named "mercedes" instead of "mercedes_benz". We also manually search for vw since some of the cars are named "vw" instead of "volkswagen".

Assignment of brand by unique_model

```
if( check_brand == FALSE) {
    for( i in 1 : length( unique_models_df_final$model)) {
      if( grepl( unique_models_df_final$model[i], names, ignore.case = TRUE)) {
        assigned_brand = c( assigned_brand, unique_models_df_final$brand[i] )
        check_model = TRUE
        break
      }
    }
  }
```

The datapoints that have arrived at this condition have not had their brand assigned by unique_brand. Hence, we assign brand by unique_model instead by searching whether the unique_model is in names using grepl. If grepl returns TRUE, we assign the corresponding brand of the matching unique_model to the brand of the name. We also assign TRUE to the boolean variable check_model.

Assignment of NA

```
  if( check_brand == FALSE && check_model == FALSE ){
    assigned_brand = c( assigned_brand, NA )
    unassigned_name = c( unassigned_name, names ) ## keeping to create unassigned dataframe
  }
}
```

The datapoints that have arrived at this condition have not had their brand assigned by unique_brand or unique_model. Hence, we assign NA. We also included the names of these datapoints into the unassigned_name variable to keep track of them.

Creating "returnNA" dataframe

```
unassigned = data.frame( unassigned_name )
colnames( unassigned ) = c( "name" )
unassigned$name = as.character(unassigned$name )
returnNA = inner_join( unassigned, autoswithout, by = "name" )
```

Here we create the returnNA using inner_join by joining unassigned to autoswithout since unassigned contains observations which we fail to find a brand for.