Using R in Data mining for the masses, Chapter 3

Ulrik Hørlyk Hjort

April 5, 2014

1 Data Scrubbing

In the following text "the book" will refer to the the book: "Data Mining for the masses"

1.1 Importing data into an data frame

Import an csv file by using the **read.csv()** function. Use the help function in R - *help(read.csv)* or just *?read.csv* - to see a full description of this function and the arguments it takes.

Start the R commandline interpreter environment and type the following command to inport the dataset for chapter 3 (It is assumed that the data set file is located in the directory where the R interprenter is executed):

```
data = read.csv(‘‘Chapter03DataSet.csv’’, sep=’’,’’,header = TRUE)
```

This line imports the data tables from the csv file into the data frame object **data** and is equivalent to the action that add a data set to a process in RapidMiner.

To get a overview of the data set - like the **Result Perspective** in RapidMiner - simply type the name of the data frame, which in this case is **data**. We then get the following output of the data set where the top line of the table, called the header, contains the column names. Each horizontal line

afterward denotes a data row, which begins with the name of the row, and then followed by the actual data. Each data member of a row is called a cell. Observe that since the number of columns exceed the screen witdh the coloums will be split up in the view.

Output of the data frame view:

```
> data
   Gender              Race Birth_Year Marital_Status Years_on_Internet
1       M             White       1972              M                 8
2       M          Hispanic       1981              S                14
3       F African American       1977              S                 6
4       F             White       1961              D                 8
5       M             White       1954              M                 2
6       M African American       1982              D                15
7       M African American       1981              D                11
8       M             White       1977              S                 3
9       F African American       1969              M                 6
10      M             White       1987              S                12
11      F          Hispanic       1959              D                12
```

	Hours_Per_Day	Preferred_Browser	Preferred_Search_Engine	Preferred_Email
1	1	Firefox	Google	Yahoo
2	2	Chrome	Google	Hotmail
3	2	Firefox	Yahoo	Yahoo
4	6	Firefox	Google	Hotmail
5	3	Internet Explorer	Bing	Hotmail
6	4	Internet Explorer	Google	Yahoo
7	2	Firefox	Google	Yahoo
8	3	Internet Explorer	Yahoo	Yahoo
9	2	Firefox	Google	Gmail
10	1	Safari	Yahoo	Yahoo
11	5	Chrome	Google	Gmail

	Read_News	Online_Shopping	Online_Gaming	Facebook	Twitter
1	Y	N	N	Y	N
2	Y	N	N	Y	N
3	Y	Y		Y	N
4	N	Y	N	N	Y
5	Y	Y	N	Y	N
6	Y	N	Y	N	N
7		Y		Y	Y
8	Y			Y	99
9	N	Y	N	N	N
10	Y		Y	Y	N
11	Y	N	N	Y	N

	Other_Social_Network
1	
2	
3	
4	
5	
6	
7	LinkedIn
8	LinkedIn
9	
10	MySpace
11	Google+

1.2 Data Reduction

The data reduction example in the book reduces the data set by filtering out the row in which the **Online_Shopping** attribute is undefined. In R it is done in the following way:

```
data <- data[data$Online_Shopping==''Y'' | data$Online_Shopping==''N'',]
```

The inverted filter function is easily done by negate the logical expression:

```
data <- data[!data$Online_Shopping==''Y'' & !data$Online_Shopping==''N'',]
```

1.2.1 Sample Of A Data Frame

Reducing the size of the data set by takeing a sample can be done in the following way in R. First we create a function to generate an random sample on a data frame:

```
random_sample = function(data_frame,n) {
     return (df[sample(nrow(data_frame), n),])
}
```

The random sample function takes the data frame as first argument and the number of row we wish in the reduced data set as second argument. We can generate a new data frame with 5 random rows from the original one by:

```
reduced_data_frame<-randomSample(data, 5)
```

where **reduced_data_frame** is the new data frame containing the 5 row reduced data set.

1.3 Handling Inconsistent Data

In R we can show a summary view of the data frame which display the modes, means, medians and factor levels for the data rows. Looking at the summary data frame view we can see the inconsistent value 99 for the **Twitter** attribute:

```
> summary(data)
 Gender               Race     Birth_Year   Marital_Status Years_on_Internet
 F:4    African American:4   Min.   :1954   D:4            Min.   : 2.000
 M:7    Hispanic        :2   1st Qu.:1965   M:3            1st Qu.: 6.000
        White           :5   Median :1977   S:4            Median : 8.000
                             Mean   :1973                  Mean   : 8.818
                             3rd Qu.:1981                  3rd Qu.:12.000
                             Max.   :1987                  Max.   :15.000
 Hours_Per_Day           Preferred_Browser Preferred_Search_Engine
 Min.   :1.000   Chrome           :2       Bing  :1
 1st Qu.:2.000   Firefox          :5       Google:7
 Median :2.000   Internet Explorer:3       Yahoo :3
 Mean   :2.818   Safari           :1
 3rd Qu.:3.500
 Max.   :6.000
 Preferred_Email Read_News Online_Shopping Online_Gaming Facebook Twitter
 Gmail  :2        :1        :2              :3           N:3      99:1
 Hotmail:3       N:2       N:4             N:6          Y:8      N :8
 Yahoo  :6       Y:8       Y:5             Y:2                   Y :2

 Other_Social_Network
        :7
 Google+ :1
 LinkedIn:2
 MySpace :1
```

We wish to change this value to a valid one e.g “Y” or “N”. Since the mode of “N” is 80% we choose that as the replacement value. The replacement is done in the following way in R:

```
data$Twitter[!data$Twitter == "Y" & !data$Twitter == "N"] <- "N"
```

It is ly possible to replace with values already in the range and the following will result in an error message:

```
#data$Twitter[!data$Twitter == ‘‘Y’’ & !data$Twitter == ‘‘N’’] <- ‘‘Q’’

Error Message:
In ‘[<-.factor‘(‘*tmp*‘, !d1$Twitter == ‘‘Y’’ & !d1$Twitter == ‘‘N’’,  :
  invalid factor level, NAs generated
```

If we want to assign a value that is not currently a factor level, you first would need to create the additional level like:

```
levels(data$Twitter) <- c(levels(data$Twitter), ‘‘Q’’)
```

1.4 Handling Inconsistent Data

Reducing the data set to a subset data frame containing the attributes **"Birth_Year"**, **"Gender"**, **"Martial_Status"**, **"Race"** and **"Years_on_Internet"** is done in the following way in R:

```
reduced_data_frame<-subset(data, select = c("Gender", "Birth_Year",
                                            "Marital_Status", "Race",
                                            "Years_on_Internet"))
```

If we instead wish to remove attributes from the original data set we can do the following which delete the attributes **"Twitter"**, **"Gender"** and **"Race"**:

```
data$Twitter <- NULL
data$Gender <- NULL
data$Race <- NULL
```