Overview of Data Preparation Procedures

Noushin Nabavi

2018-10-16

To first achieve the goals and meet expectations of how data assets can benefit us, we need to first determine how we can execute a data analytics workflow successfully. As good practice, a data scientist needs to first understand the overall goal of a project and what each member or stakeholder is looking to achieve from the dataset. A good way to start would be to have all interested parties who are involved start throwing ideas around and brainstorm on possibilities of what datasets can provide. Following a brainstorming session, the manual part of the data cleaning process starts and can be achieved by various softwares. In general, data cleaning is the process of detecting or removing corrupt and inaccurate records from a record set, table, or database and replacing, modifying, or deleting the dirty, missing, or coarse data. Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting. Regardless of which software is used, the procedures need to be closely documented, monitored, and inconsistencies reviewed, and as such a protocol for data cleaning is imperative. This is important because incorrect or inconsistent data can lead to false conclusions and misdirected investments on both public and private scales. For instance, the government may want to analyze population income figures to decide which regions require further spending and investment on infrastructure and services. In this case, it will be important to have access to reliable data to avoid erroneous fiscal decisions.
Below are eight broad steps and examples for preparing the data with various softwares so that it is repeatable on different platforms and that the data and results are accurate.

Data Cleaning Steps

1. Install softwares, dependencies, and necessary libraries:

Softwares could include installing new packages or loading existing libraries to be able to **read** the data, **transform** or **reshape** the data, find **errors** in the data, and finally **analyze** and **tabulate** the data. This includes softwares that enables reading the data, transforming or reshaping the data, finding errors in the data, and finally analyzing and tabulating the data. This could include using SQL, Python, R, SAS, etc.

2. Import the data and read into chosen softwares:

One may need to convert the data upstream outside of a chosen software to have a ready format file in order to read into the chosen software. The steps here include (i) staging or locating the files, (ii) loading or importing the data, and (iii) recoding and transforming the data format to be compatible with other software. One could also check the top 5 rows of the table to ensure proper input. An instance is shown below:

```
setwd("~/Exploratory_R")
getwd()
```

```
data <- fread("Life_Expectancy.csv", sep = ",")
head(data, 5)
```

```
##    YEAR    GEO Age_group  Sex
## 1: 1980 Canada         0 Both
## 2: 1980 Canada         0 Both
## 3: 1980 Canada         0 Both
## 4: 1980 Canada         0 Both
## 5: 1980 Canada         0 Both
##                                          Element  AVG_VALUE
```

```
## 1:                   Number of survivors at age x (lx) 1.0000e+05
## 2:         Number of deaths between age x and x+1 (dx) 9.7600e+02
## 3:        Death probability between age x and x+1 (qx) 9.7600e-03
## 4: Margin of error of the death probability (m.e.(qx)) 1.8000e-04
## 5:  Probability of survival between age x and x+1 (px) 9.9024e-01
```

3. Monitor errors:

Inspect the dataset thoroughly and get to understand the fields. This includes looking at trends in data, outliers, minimum and maximum values, and missing data or errors. Next step is keeping a log of where most errors are in order to identify and fix the incorrect or corrupt data types, including typographical errors. A harmonization of names and variables is also necessary if one is integrating the data with other datasets. Examining the data quality includes inspecting the validity, accuracy, completeness, consistency, and uniformity of data columns and rows. An example of finding outliers, means, and margins of error. An example of finding outliers, means, and margins of error include deciphering the summary of dataset like belo:

```
class(data)
```

```
## [1] "data.table" "data.frame"
```

```
str(data)
```

```
## Classes 'data.table' and 'data.frame':   1048575 obs. of  6 variables:
##  $ YEAR     : int  1980 1980 1980 1980 1980 1980 1980 1980 1980 1980 ...
##  $ GEO      : chr  "Canada" "Canada" "Canada" "Canada" ...
##  $ Age_group: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Sex      : chr  "Both" "Both" "Both" "Both" ...
##  $ Element  : chr  "Number of survivors at age x (lx)" "Number of deaths between age x and x+1 (dx)"
##  $ AVG_VALUE: num  1.00e+05 9.76e+02 9.76e-03 1.80e-04 9.90e-01 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
nrow(data)
```

```
## [1] 1048575
```

```
ncol(data)
```

```
## [1] 6
```

```
print(data)
```

```
##          YEAR              GEO Age_group  Sex
##       1: 1980           Canada         0 Both
##       2: 1980           Canada         0 Both
##       3: 1980           Canada         0 Both
##       4: 1980           Canada         0 Both
##       5: 1980           Canada         0 Both
##      ---
## 1048571: 2014 British Columbia        96    F
## 1048572: 2014 British Columbia        96    F
## 1048573: 2014 British Columbia        97 Both
## 1048574: 2014 British Columbia        97 Both
## 1048575: 2014 British Columbia        97 Both
##                                                  Element    AVG_VALUE
##       1:                   Number of survivors at age x (lx) 1.000000e+05
##       2:         Number of deaths between age x and x+1 (dx) 9.760000e+02
##       3:        Death probability between age x and x+1 (qx) 9.760000e-03
##       4: Margin of error of the death probability (m.e.(qx)) 1.800000e-04
```

```
##       5:  Probability of survival between age x and x+1 (px) 9.902400e-01
##      ---
## 1048571:            Life expectancy (in years) at age x (ex) 3.133333e+00
## 1048572:   Margin of error of the life expectancy (m.e.(ex)) 1.000000e-01
## 1048573:                   Number of survivors at age x (lx) 8.909667e+03
## 1048574:         Number of deaths between age x and x+1 (dx) 2.273667e+03
## 1048575:        Death probability between age x and x+1 (qx) 2.552200e-01
```

```
head(data)
```

```
##    YEAR    GEO Age_group  Sex
## 1: 1980 Canada         0 Both
## 2: 1980 Canada         0 Both
## 3: 1980 Canada         0 Both
## 4: 1980 Canada         0 Both
## 5: 1980 Canada         0 Both
## 6: 1980 Canada         0 Both
##                                                 Element  AVG_VALUE
## 1:                     Number of survivors at age x (lx) 1.0000e+05
## 2:           Number of deaths between age x and x+1 (dx) 9.7600e+02
## 3:          Death probability between age x and x+1 (qx) 9.7600e-03
## 4:   Margin of error of the death probability (m.e.(qx)) 1.8000e-04
## 5:    Probability of survival between age x and x+1 (px) 9.9024e-01
## 6: Number of life years lived between age x and x+1 (Lx) 9.9152e+04
```

```
dim(data)
```

```
## [1] 1048575       6
```

```
names(data)
```

```
## [1] "YEAR"      "GEO"       "Age_group" "Sex"       "Element"   "AVG_VALUE"
```

```
summary(data)
```

```
##       YEAR          GEO              Age_group          Sex
##  Min.   :1980   Length:1048575     Min.   :  0.00   Length:1048575
##  1st Qu.:1988   Class :character   1st Qu.: 27.00   Class :character
##  Median :1997   Mode  :character   Median : 55.00   Mode  :character
##  Mean   :1997                      Mean   : 54.98
##  3rd Qu.:2006                      3rd Qu.: 83.00
##  Max.   :2014                      Max.   :110.00
##
##    Element            AVG_VALUE
##  Length:1048575     Min.   :      0
##  Class :character   1st Qu.:      0
##  Mode  :character   Median :     16
##                     Mean   : 340560
##                     3rd Qu.:  59880
##                     Max.   :8455458
##                     NA's   :1047
```

4. Scrub for duplicated or erroneous data:

Similar to step #3, as part of the quality assurance, we need to identify duplicates and error schema in the dataset since this will help save time when analyzing. This step also entails scanning for accuracy and reproducibility of data linkage if this has been performed. This is done by cross checking with the files prior to

input and ensuring the number of rows and columns match the metadata before deterministic or probabilistic linkage. This could also include testing the individual column, e.g. for unexpected values like NULL values, non-numeric values that should be numeric, out of range values, as well as data linkage discrepancies. The reiterative procedures in this step can be avoided by researching and investing in data cleaning tools (dependencies, libraries) such that the data can be analyzed in bulk and the process is automated. For instance, in RStudio, quality assurance can be performed with *dplyr* package among others.

5. Standardize and validate the accuracy of both codes and data:

As reiteration of steps 3-4, it is also important to (re)validate the accuracy of the data through scanning for anomalis and contradictions once it has been cleaned. For example, fix the column names by checking with metadata information, transform the dates or genders, etc. as needed and parsing the data to detect all syntax errors by checking with metadata, or transform the dates or genders, etc. as needed according to the metadata and parsing the data to detect all syntax errors. The purpose here is to avoid lengthy and reiterative codes, write more functions instead of loops, and reduce or compact the source codes so that the procedure is more accurate and reliable. Note: keeping the codes simple and functional is key. This step should help with reproducibility of codes and data tables so that analysis is repeatable and reputable. Note: keep the codes simple and functional.

```
setnames(data, old = "Age_group", new = "AGE")
data[, Sex := gsub("M", "Male", Sex)]
data[, Sex := gsub("F", "Female", Sex)]
```

6. Analyze the data:

After the data has been standardized, validated, and scrubbed for duplicates, use reliable fields to analyze the data for high-level descriptive statistical analyses such as values of mean, standard deviation, range, or clustering algorithms. These results could be visualized using simple frequency tables to more sophisticated ggplots and graphs. Compile the data plots to provide more complete information for business intelligence and analytics for operational insights. For instance, *ggplot2* package can be used to visualize a subset of the data extract.

```
#boxplot(data, x=GEO, y=AVG_VALUE)
```

7. Report the data:

The data and the software specifications, codes used to analyze the data, as well as the results are to be closely documented, exported, and saved in related folders so that they can be accessible to anyone else who would want to replicate the data or produce new results from them. Reporting can be done directly with R through rmarkdown.

8. Communicate the results with the team:

Communicate the new standardized cleaning process and preliminary results of analysis to the team and receive feedback because fresh set of eyes and people's insights are almost always beneficial to improving the analysis. This step is also needed to develop and strengthen existing research and policy questions in order to later send more targeted information to stakeholders. Meeting with the team ensures that the team is in line with data cleanup before moving to other parties.