Script for generating dummy data

Dietmar H. and Fabian P.

Aug 2019

Introduction

The aim of this script is to generate dummy data for analysis using K-means algorithms. The data generated in this process is similar to data that can be found in insurance companies and has three columns:

- **sex:** The sex of the insured person.
- **age:** The age of the insured person.
- **sum_assured:** The sum insured assigned to the contract.
- **ID:** A unique contract-ID for every policy. ### General Settings The pacman package is an R package management tool for loading and installing packages if necessary. The following packages are used for the data generation:
- **data.table:** For filtering, grouping and transforming the data as well as fast read and write opterations. This package is particularly suitable for the fast processing of large amounts of data.
- **dplyr:** This package is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges. We need it especially for the use of the pipe operator.
- **rmarkdown:** R Markdown provides an authoring framework for data science. One can use a single R Markdown file to save and execute code and generate high quality reports that can be shared with an audience. We suppress the warning for the *pacman* library because the message that the package was built under a different R-versions should not bother the user by beeing displayed on the console in red. Then the required packages are loaded and installed if necessary.

```
suppressWarnings(if (!require("pacman")) install.packages("pacman"))
pacman::p_load(data.table, dplyr, rmarkdown)
```

For the generation of the test data we use functions which work with random numbers. To make the results reproducible we initialize the random number generator with an arbitrary number (here 100) to ensure that the random numbers are the same at every program run.

```
set.seed(100)
```

It is defined that each cluster created should contain 100 data points. Of course, it is also possible to randomly determine the cluster size which could be a further enhancement of the code.

```
count_per_group <- 100
```

Clusters for both men and women are defined and then merged into a data set. Two variables are defined for each cluster, namely age and sum assured. For each of these variables, both the average and the standard deviation are specified.
For example, the first cluster of men is defined as :

$$age \sim \mathcal{N}(20, 3^2) \quad \text{and} \quad sum\_assured \sim \mathcal{N}(10000, 2000^2)$$

```
mean_age_male <- c(20   , 20   , 45  , 45   , 70   , 70)
sd_age_male   <- c(3    , 4    , 4.2 , 3    , 3.2  , 6)
```

```
mean_vs_male  <- c(10000, 35000, 8000, 30000, 10000, 45000)
sd_vs_male    <- c(2000 , 3000 , 1000, 4000 , 1000 , 6000)
```

After all parameters for the 6 clusters have been defined, the individual data points can be created. For this purpose, one loops over the parameter set and 100 data points with the corresponding parameters are randomly generated in each pass. The randomly generated values for the age as well as the sum insured are then rounded to obtain integer values. The datasets of the 6 clusters are returned in separate data.tables and then converted using rbindlist into a dataset containing all male policyholders.

```
males <- lapply(seq_along(mean_age_male), function(i) {
  data.table(
    sex = "m",
    age = rnorm(count_per_group, mean = mean_age_male[i], sd = sd_age_male[i]) %>%
      round(),
    sum_assured = rnorm(count_per_group, mean = mean_vs_male[i], sd = sd_vs_male[i]) %>%
      round()
  )
}) %>% rbindlist()
```

As before for men, the parameters for women are now defined. The first of the six cluster for females is defined as :

$$age \sim \mathcal{N}(25, 3^2) \quad \text{and} \quad sum\_assured \sim \mathcal{N}(40000, 4000^2)$$

```
mean_age_female <- c(25    , 32   , 45    , 57    , 70    , 70)
sd_age_female   <- c(2     , 3.2  , 2.2   , 3     , 1.5   , 4)
mean_vs_female  <- c(40000, 9000, 42000, 10000, 10000, 35000)
sd_vs_female    <- c(4000 , 1000, 3500 , 1500 , 1000 , 4200)
```

The data set of the females can then be generated with the already defined parameters.

```
females <- lapply(seq_along(mean_age_male), function(i) {
  data.table(
    sex = "f",
    age = rnorm(count_per_group, mean = mean_age_female[i], sd = sd_age_female[i]) %>%
      round(),
    sum_assured = rnorm(count_per_group, mean = mean_vs_female[i], sd = sd_vs_female[i]) %>%
      round()
  )
}) %>% rbindlist()
```

Once both sub-portfolios for males and females have been generated, they can be merged into a final data set holding all dummy policies. At the end the record is extended by the column ID which represents a unique insurance policy.

```
policies <- rbind(males, females)[, ID := sample(.N)][order(ID)]
```

Many data sets available in practice have for various reasons partly incorrect or missing data points. This behavior is simulated by setting some entries in the age column to invalid entries (NA). This prepared data set is in the last step exported as a text file with the name *insurance_portfolio.txt*.

```
policies %>%
  {sample(1:nrow(.), size = 10, replace = FALSE)} %>%
  {policies[., age := NA_integer_]} %>%
  fwrite("insurance_portfolio.txt")
```