Data in Public and Social Services - 1st practical exercise class

25th March 2024

by Davide Chicco
davide.chicco@unimib.it

Practical exercise class lecturer: Vasco Coelho v.coelho@campus.unimib.it

The goals of this practical exercise class are the following:

A. Learn how to use R and how to execute its most common commands
B. Apply the concepts on data cleaning and data preparation seen during the first three class lectures on an EHRs dataset:
    a. Check about data privacy and documentation
    b. Understand a bit about the disease, and if it is rare or common
    c. Have a look at the dataset
    d. Duplicate detection
    e. Error detection
    f. Inconsistency detection
    g. Outlier detection
    h. Data transformation (feature names, categories)
    i. Missing data handling
    j. unbalanced data handling

Introduction to R main commands

0) Put the following commands as header of your script:

```
setwd(".") #we use the current folder of the script as working directory
options(stringsAsFactors = FALSE) # we set the input strings not to be considered
set.seed(10) # we set a seed to be able to replicate our tests
options(repos = list(CRAN="http://cran.rstudio.com/")) # we set the URL where to download the packages
```

1) Install the pacman library for an easier installation and loading of the libraries:

```
install.packages("pacman", dependencies = TRUE)
library("pacman")
```

Load and/or install other packages we need

```
p_load("dplyr", "ggplot2", "pastecs")
```

Load the iris dataset that is built-in in R

```
data(iris)
```

Use the head(), dim() and str() commands to "have a look" on this dataset
head() shows the header of the dataframe
dim() prints the dimensions of the dataset
colnames() prints the names of the columns
str() provides information about the structure of it
Some variables in R can be factors. A factor is a data structure used for fields that takes only a predefined, finite number of values (categorical data). For example: a data field such as marital status may contain only values from single, married, separated, divorced, or widowed.

```
a <- 2
a = 2

head(iris)
iris %>% head()

iris %>% dim()

iris %>% colnames()
iris %>% colnames() %>% sort()

iris %>% str()
```

What does the output of str() say?

Use the summary() command to print the descriptive statistics of this dataframe

```
iris %>% summary()
```

Print the 3rd row, the 4th column, and then the "Petal.Length" column of the iris dataset
The print the content of the cell of the 3rd row and 4th column

```
iris[3,]
iris[,4]
iris$Petal.Length

iris[3,4]
```

Create a contingency table of the values of the column "Petal.Length" and print it

```
iris$Petal.Length %>% table()
iris$Petal.Length %>% table() %>% sort()
```

```
iris %>% select(Petal.Length) %>% table() %>% sort()
```

Print a variable with cat()

```
cat("the number of rows of iris is ", nrow(iris), "\n", sep="")
```

Discover the number of missing data (NAs) in the dataframe by using is.na() and then sum()
Then check the number of missing data in the first column and in the second row

```
iris %>% is.na() %>% sum()
iris[1,] %>% is.na() %>% sum()
iris[,2] %>% is.na() %>% sum()
```

Select only the numeric columns of the iris dataframe and put it in a dataframe called iris_num

```
iris_num <- iris %>% select(where(is.numeric))
iris_num <- select(iris, where(is.numeric))
```

Check the content of iris_num with dim(), summary(), str(), and stat.desc()
What do you notice?

```
iris_num %>% dim()
iris_num %>% summary()
iris_num %>% str()
stat.desc(iris_num)
```

Multiply the content of iris_num by 10, divide it by 9, add 8 to it, and subtract 5 to it

```
iris_num * 10
iris_num / 9
iris_num + 8
iris_num - 5
```

Application of the main data cleaning and data preparation steps to a dataset of electronic health records of patients with diabetes type 1 from Japan

2) Apply the concepts on data cleaning and data preparation seen during the first three class lectures on a real EHRs dataset:.

2.1) Takashi 2019 diabetes type 1 dataset: to download

Takashi2019_diabetes_type1_MANIPULATED.csv file to download

Takashi Y, Ishizu M, Mori H, Miyashita K, Sakamoto F, Katakami N, et al. (2019) “Circulating osteocalcin as a bone-derived hormone is inversely correlated with body fat in patients with type 1 diabetes”. PLOS ONE 14(5): e0216416. https://doi.org/10.1371/journal.pone.0216416

Cerono G, Chicco D. 2024, “Ensemble machine learning reveals key features for diabetes duration from electronic health records”. PeerJ Computer Science 10:e1896 https://doi.org/10.7717/peerj-cs.1896

A. Check about data privacy and documentation

   The first paper contains a link to the URL with the license

B. Understand a bit about the disease, and if it is rare or common. Is it more common among kids, young people, or old people? Men or women?

   You can read information in the Introduction of the two cited articles or online on the website of the World Health Organization (WHO) or on Wikipedia, for general introductory concepts (not for specific information).

C. Have a look at the dataset

```
fileName <- "Takashi2019_diabetes_type1_MANIPULATED.csv"
patients_data <- read.csv(fileName, header = TRUE, sep =",")

patients_data %>% dim()
patients_data %>% summary()
patients_data %>% str()
table(patients_data$gender)
```

D. Duplicate detection

   Are there duplicate features that can be eliminated? Yes, we can create the “weight_kg” feature and eliminate TDD/kg, basal/kg, bolus/kg, since TDD, basal, and bolus are already present

```
patients_data %>% dim()
patients_data %>% colnames() %>% sort()

# weight_kg = TDD * (weight_kg/TDD)

patients_data$weight_kg <- -1
patients_data$weight_kg <- patients_data$TDD * (1/patients_data$TDD.kg)
```

```
patients_data$weight_kg %>% summary()

patients_data$basal.kg <- NULL
patients_data$bolus.kg  <- NULL
patients_data$TDD.kg  <- NULL


patients_data %>% dim()
patients_data %>% colnames() %>% sort()
```

E. Outlier and error detection

There are errors among age, duration of diabetes, and BMI
Double-check against Table 2 of the second article.

Let's see the intervals of age, duration of diabetes, and BMI

```
patients_data$age %>% summary()
patients_data$BMI %>% summary()
patients_data$duration.of.diabetes %>% summary()

patients_data %>% dim()
```

We see in the Table 2 of the article that
age should be between 21 and 48
BMI should be between 17.584 and 35,54
duration of diabetes should be between 10 and 41

```
patients_data %>% dim()

min_age <- 21
max_age <- 48
patients_data_age_checked <- patients_data[which(patients_data$age >=
min_age & patients_data$age <= max_age),]

patients_data_age_checked %>% dim()

min_BMI <- 17.584
max_BMI <- 35.54
patients_data_BMI_age_checked <-
patients_data_age_checked[which(patients_data_age_checked$BMI >=
min_BMI & patients_data_age_checked$BMI <= max_BMI),]

patients_data_BMI_age_checked %>% dim()
```

```
min_diabetes_duration <- 10
max_diabetes_duration  <- 41
patients_data_diabetes_duration_BMI_age_checked <-
patients_data_BMI_age_checked[which(patients_data_BMI_age_checked$durat
ion.of.diabetes >= min_diabetes_duration &
patients_data_BMI_age_checked$duration.of.diabetes <=
max_diabetes_duration),]

patients_data_diabetes_duration_BMI_age_checked %>% dim()

patients_data <- patients_data_diabetes_duration_BMI_age_checked
```

F. Inconsistency detection

There are errors considering age and duration of diabetes together.
Of course, age cannot be lower than diabetes duration

```
patients_data %>% dim()

patients_data_age_greater_diabetes_duration <-
patients_data[which(patients_data$duration.of.diabetes <
patients_data$age),]

patients_data_age_greater_diabetes_duration %>% dim()
patients_data_age_greater_diabetes_duration <- patients_data
```

G. Data transformation (feature names, categories)

The “no” column can be removed.

```
patients_data$no <- NULL
```

The gender and insulin_regimen columns should be converted from binary categories to numerical values.

```
patients_data$insulin_regimen %>% table()
patients_data$insulin_regimen_binary <- -1
patients_data[which(patients_data$insulin_regimen=="CSII"),]$insulin_re
gimen_binary <- 0
patients_data[which(patients_data$insulin_regimen=="MDI"),]$insulin_reg
imen_binary <- 1
patients_data$insulin_regimen <- NULL # we remove the insulin_regimen
original variable

patients_data$gender %>% table()
patients_data$sex_0man_1woman <- -1
```

```
patients_data[which(patients_data$gender=="male"),]$sex_0man_1woman  <-
0
patients_data[which(patients_data$gender=="female"),]$sex_0man_1woman
<- 1
patients_data$gender <- NULL # we remove the gender original variable
```

Some names can be improved (check Table 1 of the second paper):
gender → sex_0man_1woman
insulin_regimen_binary →insulin_regimen_0CSII_1MDI

BMI → body_mass_index
age → age_years
duration of diabetes → diabetes_duration_years
OC → total_osteocalcin
SMI → skeletal_muscle_mass_index
TDD --> total_daily_dose_of_insulin
ucOC --> undercarboxilated_osteocalcin

```
BMI_column_index <- which(colnames(patients_data)=="BMI")
colnames(patients_data)[BMI_column_index] <- "body_mass_index"
```

Age...
Duration of diabetes...
etc...

H. Missing data handling

```
#  Are there missing data? We remove the observation and not the feature.

patients_data %>% is.na() %>% sum()

patients_data_with_missing_data <- patients_data
patients_data %>% dim()
patients_data %>% na.omit() %>% dim()

# how many rows do have missing data?
cat("The number of rows with missing data is ",
sum(!complete.cases(patients_data)), "\n", sep="")

p_load("mice")
MISSING_DATA_IMPUTATION <- TRUE
if(MISSING_DATA_IMPUTATION==TRUE){
```

```
        # missing data imputation
        NUM_DATASETS <- 1
        this_seed <- 500
        max_iterations <- 50
        imputed_data <- mice(patients_data, m=NUM_DATASETS, maxit
        = max_iterations, method = 'pmm', seed = this_seed)
        patients_data <- complete(imputed_data, NUM_DATASETS)
}

patients_data_with_missing_data %>% is.na() %>% sum()
patients_data %>% is.na() %>% sum()
```

I. Unbalanced data handling

We use insulin_regimen as binary target

How much data do we impute? The difference between the two classes of the target variable. Here we use insulin_regimen as target value, so we need to calculate the difference between the number of 0s and the number of 1s.

```
TRAINING_SET_RATIO <- 0.8

p_load("performanceEstimation")

patients_data <- patients_data[sample(nrow(patients_data)),] # shuffle
the rows

patients_training_set_index_start <- 1
patients_training_set_index_end <- round(nrow(patients_data) *
 TRAINING_SET_RATIO)
patients_test_set_index_start <- patients_training_set_index_end + 1
patients_test_set_index_end <- nrow(patients_data)

patients_training_set <-
patients_data[patients_training_set_index_start:patients_training_set_i
ndex_end,]
patients_test_set <-
patients_data[patients_test_set_index_start:patients_test_set_index_end
,]

patients_training_set_original <- patients_training_set

# we check the dimensions before oversampling
patients_training_set %>% dim()

imbal_number <- abs((patients_training_set$insulin_regimen_binary %>%
```

```
 table() %>% as.numeric())[1] -
(patients_training_set$insulin_regimen_binary %>% table() %>%
 as.numeric())[2])

cat("The unbalance in the training set regards ",  imbal_number, " data
observations\n", sep="")

unbalance_numbers <-
patients_training_set_original$insulin_regimen_binary %>% table() %>%
as.numeric() # elements of the two classes
unbalance_augmentation <- round(unbalance_numbers[2] *100 /
unbalance_numbers[1]) # percentage of the majority class over minority
class

new_data <- NULL

  TRAIN_SET_OVERSAMPLING_SYNTHETIC_SMOTE <- TRUE
  if(TRAIN_SET_OVERSAMPLING_SYNTHETIC_SMOTE == TRUE)
      {

           patients_training_set$"insulin_regimen"<-
            as.factor(patients_training_set$"insulin_regimen_binary")

            target_index <-
            which(colnames(patients_data)=="insulin_regimen_binary")

            allFeaturesFormula <-
            as.formula(paste(as.factor(colnames(patients_training_set)[
            target_index]), '.', sep=' ~ ' ))

           cat("SMOTE oversampling\n")
           new_data <- smote(allFeaturesFormula, data =
           patients_training_set,
          perc.over = unbalance_augmentation, perc.under = 1, k=5)
      }

# we check the dimensions after oversampling
patients_training_set_original$insulin_regimen %>% table() * 100 /
nrow(patients_training_set_original)
new_data$insulin_regimen %>% table()  * 100 / nrow(new_data)
```

J. Insert comments for all the previous R commands you used

Application of the main data cleaning and data preparation steps to a dataset of electronic health records of patients with diabetes type 2 from Saudi Arabia

Repeat all the steps of the previous analysis [A, B, C, ..., I]
Convert the file from XLSX to CSV, first.
Use Diabetic retinopathy (DR) as target for the data unbalance phase

For one-hot encoding, use the `one_hot()` function of the `nestedcv` package:
https://search.r-project.org/CRAN/refmans/nestedcv/html/one_hot.html

AlOlaiwi LA, AlHarbi TJ, Tourkmani AM (2018) Prevalence of cardiovascular autonomic neuropathy and gastroparesis symptoms among patients with type 2 diabetes who attend a primary health care center. PLOS ONE 13(12): e0209500.
https://doi.org/10.1371/journal.pone.0209500

Cerono G, Chicco D. 2024, "Ensemble machine learning reveals key features for diabetes duration from electronic health records". PeerJ Computer Science 10:e1896
https://doi.org/10.7717/peerj-cs.1896

2.2) pone.0209500.s001.xlsx file to download