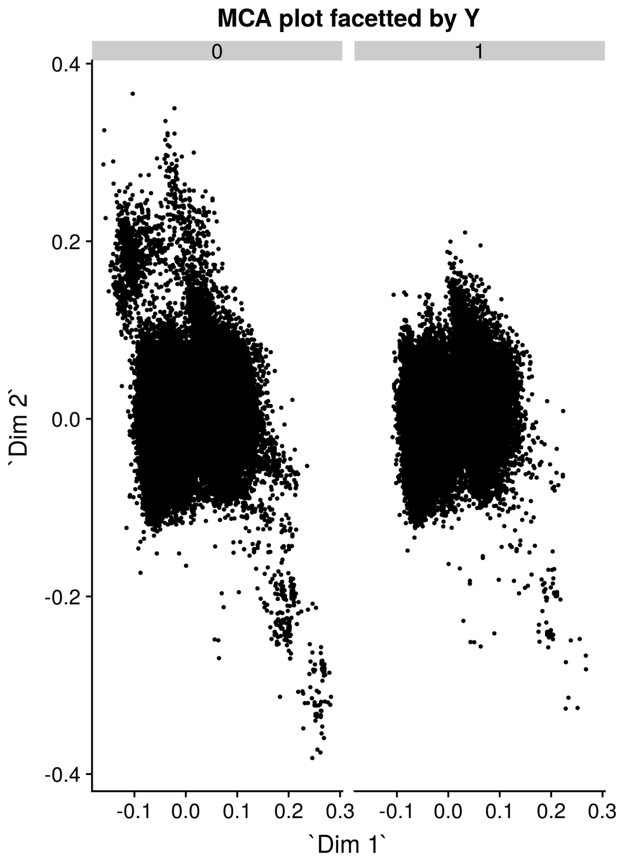
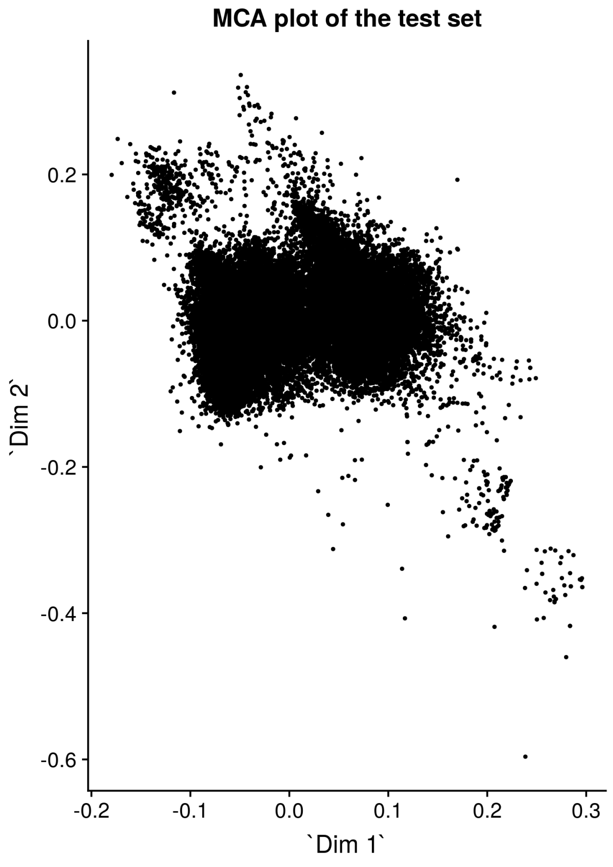
Technical test

Junior Data Scientist position at Datrik Intelligence

Antonio Ortega

October 28, 2018





Contents

1 Introduction

The dataset provided presents a classification problem, whereby the power of several features is to be harnessed to predict a binary label (1/0). A summary of the features available is presented in table 1.

binary	counter	edad	etiqueta	farmaco	identificador	nominal	ordinal	raza	sexo	Y
3	8	1	3	23	1	4	2	1	1	1

Table 1: 46 features are provided together with an identifier variable (*identificador*) and the label to be predicted Y.

Y, the target variable, is to be predicted using the information stored in the remaining features by means of (I) a linear model, and (II) a gradient boosting model. But first, the data needs to be preprocessed into a format that suits these algorithms. This is carried out in the R script `EDA.R`. A guide-through of the code is provided below.

2 EDA

1 Load libraries

```r
library(ggplot2)
library(viridis)
library(dplyr)
library(magrittr)
library(waffle)
library(tidyr)
library(tibble)
library(ade4)
library(data.table)
library(stringr)
library(FactoMineR)
library(cowplot)
library(pheatmap)
library(kableExtra)
# library(MASS) library(scales)
plot_dir <- "plots"
output_data <- "proc_data"
rdata_dir <- "RData"
tables_dir <- "tables"
```

2 Load data

Read the `datos.csv` file and split the training/validation set and the test set.

```r
datos <- read.table("datos.csv", sep = ",", header = T, stringsAsFactors = F)
datos <- datos[, c(colnames(datos)[1:4], colnames(datos)[5:(ncol(datos) - 1)] %>%
    sort, "Y")]
write(x = kable(x = datos %>% colnames %>% strsplit(., split = "_") %>% lapply(.,
    function(x) x[[1]]) %>% unlist %>% table %>% t, format = "latex", digits = 2),
    file = file.path(tables_dir, "data_summary.tex"))
```

```
datos$edad_integer <- str_match(string = datos$edad, pattern = "\\[\\d{2}-(\\d{2})\\)") %>%
    .[, 2] %>% as.integer

train_set <- datos[!is.na(datos["Y"]), ]
x_train <- train_set %>% select(-Y)
y_train <- select(train_set, Y) %>% mutate(Y = as.factor(Y))

test_set <- datos[is.na(datos["Y"]), ]
x_test <- test_set %>% select(-Y)
```

3 Visualization of race and age

A general picture of how the potentially relevant categories race and age dataset are distributed across the individuals is shown in figure 1.

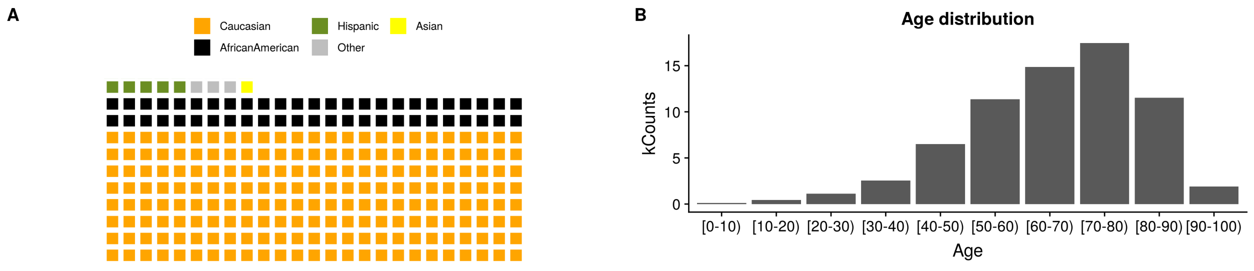


Figure 1: **A** Waffle plot showing the race distribution. A majority of the individuals are defined as Caucasian, with Afro Americans making up a significant though minor proportion. The remanining individuals are Hispanic, Asian and from other groups. **B** A histogram over the age groups reveals a trend for individuals to be aged 50 and 90 years old.

The non uniform distribution of individuals over these categories and eventually many others showcases the existence of some bias in the data.

4 Preprocess the training and test sets

Define a function to preprocess the train dataset and prepare it for the machine learning algorithms. The test set will follow a preprocessing parallel to the train

- Numerical variabels stay the same (quantitative)
- Ordinal variables (categories with order) stay the same (quantitative)
- Farmacos are considered ordinals (quantitative)
- Nominal variables (categories with no defined order) are reformatted to one-hot (qualitiative)
- Binary stays the same (qualitiative)
- Race is reformatted to one-hot (qualitiative)
- Sex stays the same (only made into 1/0) (qualitiative)
- Etiquetas. There is room for 3 etiquetas, but it is only their presence that matters. Therefore they are passed to one-hot encoding. The i,j cell will store how many times the jth etiqueta is present in the ith sample i.e. 0 to 3 times if it appears in none or 3 slots, respectively (qualitiative)

Once the function is defined, it was applied to preprocess the training and test datasets in parallel. See appendix B for source code.

5 Principal Component Analysis (PCA)

The distribution of edad (age), raza (race), gender (sexo) and Y (label) across individuals on the 2D plane capturing the most variance can be visualized by means of a PCA. The PCA takes numerical features and rotates them into a new space where variance (information) is maximised on each new feature (principal component). The first two can be used to generate the mentioned 2D plane. This is shown in figure 2

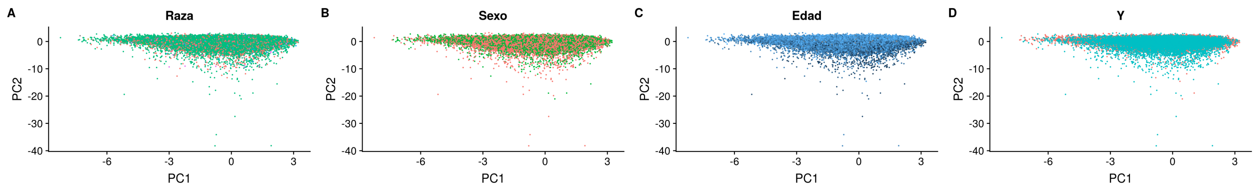


Figure 2: PCA plots for individuals colored by race, gender, age and label. No category seems to be distinguishable on the 2D plane in any of the features probed.

The PCA did not successfully evidence any clear pattern.

6 Multiple Correspondence Analysis (MCA)

An analogous analysis can be performed using categorical variables with the MCA algorithm from the R package FactoMineR. The algorithm was run with and without etiquetas (tags) to explore if saving these ensemble of one-hot features could be spared for computational efficiency. A plot of the contribution of the explored variables to the new 2D plane is shown in figure 3, whereas a plot of the individuals in this plane is shown on figure 4.

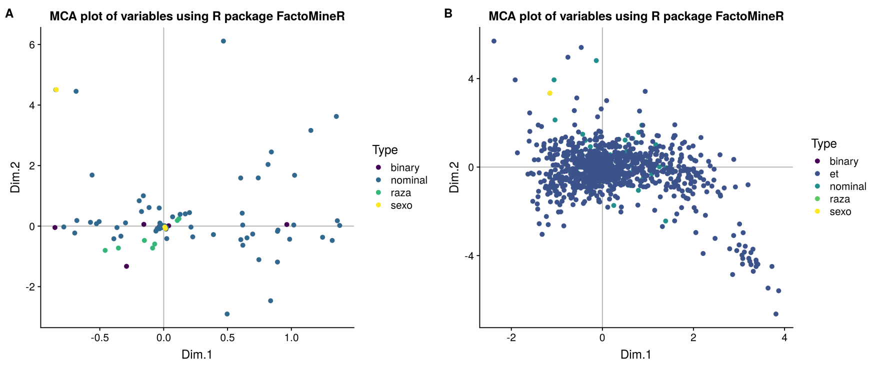


Figure 3: Visualization of the contribution of each categorical variable to the most informative MCA 2D plane. Etiquetas and nominal variables seemed to bring the most information. **A** Without etiquetas. **B** With etiquetas.

The analysis, whose results are shown in figures the mentioned figures, is able to successfuly find a pattern whereby individuals clustering in the top left corner are highly likely to be of class 0 (Y=0). The features extracted in this analysis will be added to the processed data in order to make use of their predictive power in the model training. Remarkably, if MCA is conducted without etiquetas, many individuals are projected on the same spot of the 2D plane. This effect is canceled when etiquetas are included, indicating that indeed etiquetas contribute to each of the individuals' diversity. For this reason, the top 5 MCA features produced when including etiquetas will be used to power the models. Figure 5 shows the single greatest contributions come from nominal variables, even though the whole of the etiquetas might contribute more in the end, given the sheer amount of etiquetas.

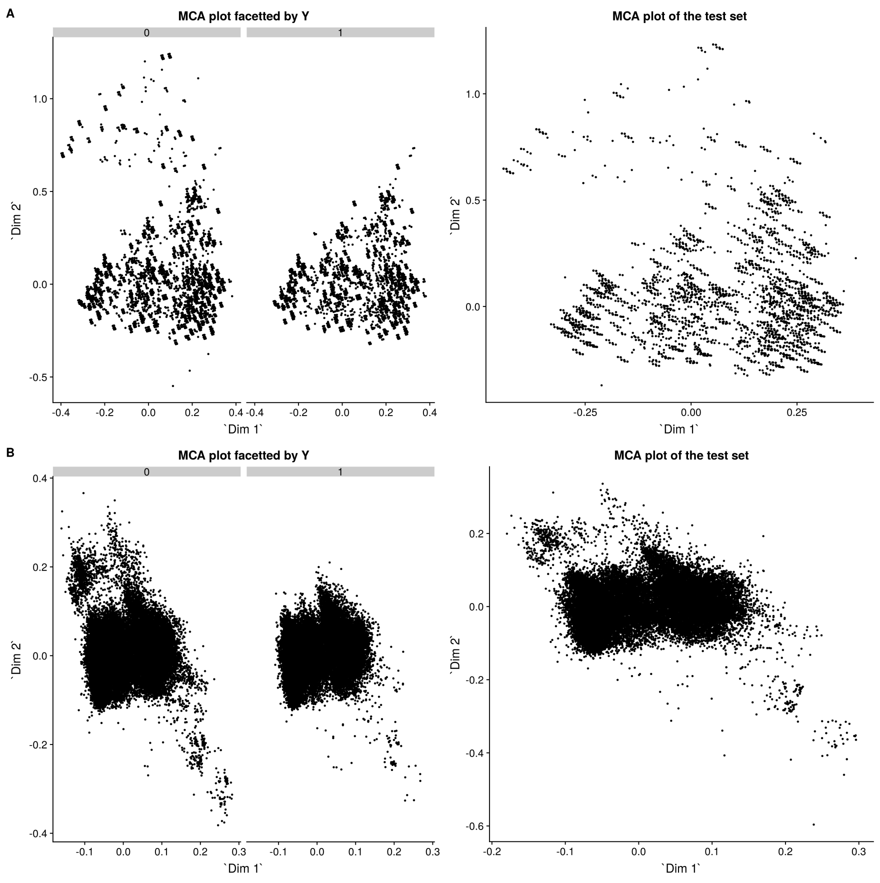


Figure 4: A visualization of the individuals on the MCA 2D plane facetted by label. A clear separation is visible for some individuals, clustering around the top left corner of the plot. A very similar pattern is observed in the test set. **A** Without etiquetas. **B** With etiquetas.

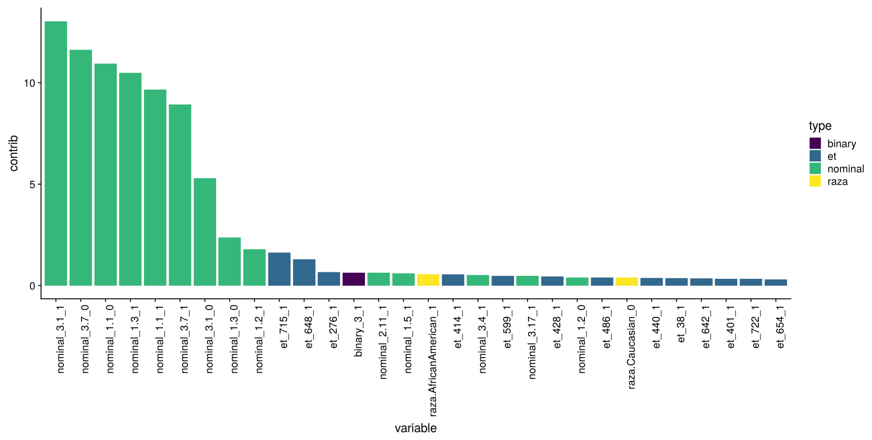


Figure 5: Top 30 variables contributing to the overall variance found by the MCA ordered by its size and coloured by type.

7 Heatmap

Finally, a heatmap (figure 6) is an alternative way to visualize the presence of any patterns/clustering in the processed dataset.

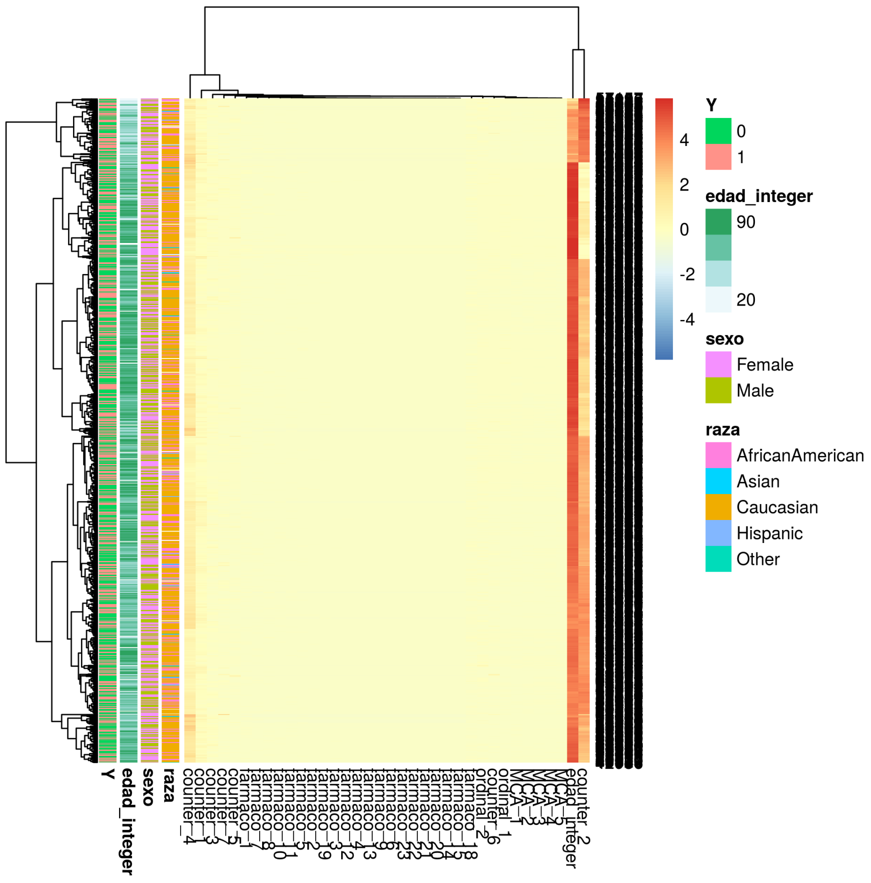


Figure 6: Heatmap of the training set including the top 5 MCA features. As expected, age and the counters with the most levels/categories are the first splitting features. However, the MCA features don't seem to contribute to the clustering.

8 Improvements to the EDA

More insights could be extracted by performing supervised projections of the dataset, where new highly informative planes are found taking into account the label. Moreover, more intensive processing, like the usage of an autoencoder, could be used to perform more powerful dimensionality reduction on the categorical variables, specially the etiquetas.

9 Export features

```r
# Save processed datasets without one-hot encoding data but with mca features
write.table(x = train_sup_proj_data$proj, file = file.path(output_data, "x_train_mca.csv"),
    sep = ",", row.names = x_train$identificador, quote = F, col.names = T)
write.table(x = test_sup_proj_data$proj, file = file.path(output_data, "x_test_mca.csv"),
    sep = ",", row.names = x_test$identificador, quote = F, col.names = T)
```

```r
write.table(x = train_sup_proj_data_full$proj, file = file.path(output_data, "x_train_mca_full.csv"),
    sep = ",", row.names = x_train$identificador, quote = F, col.names = T)
write.table(x = test_sup_proj_data_full$proj, file = file.path(output_data, "x_test_mca_full.csv"),
    sep = ",", row.names = x_test$identificador, quote = F, col.names = T)

# Save label to a separate file
write.table(x = y_train, file = file.path(output_data, "y_train.csv"), sep = ",",
    row.names = x_train$identificador, quote = F, col.names = T)
```

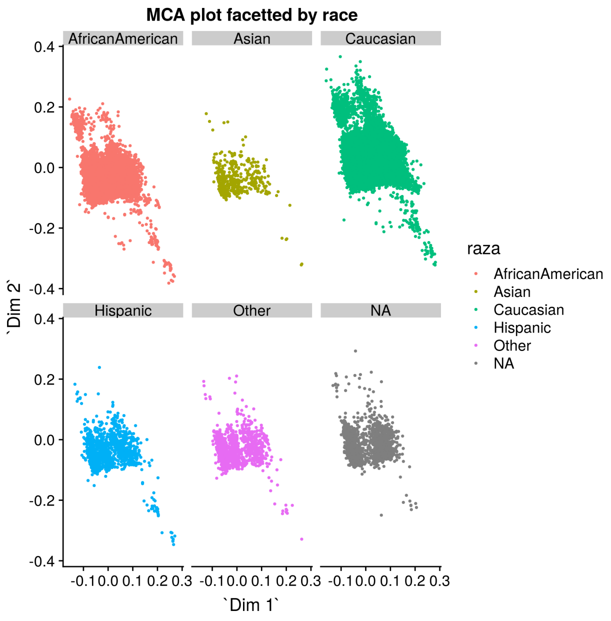
3 Session info

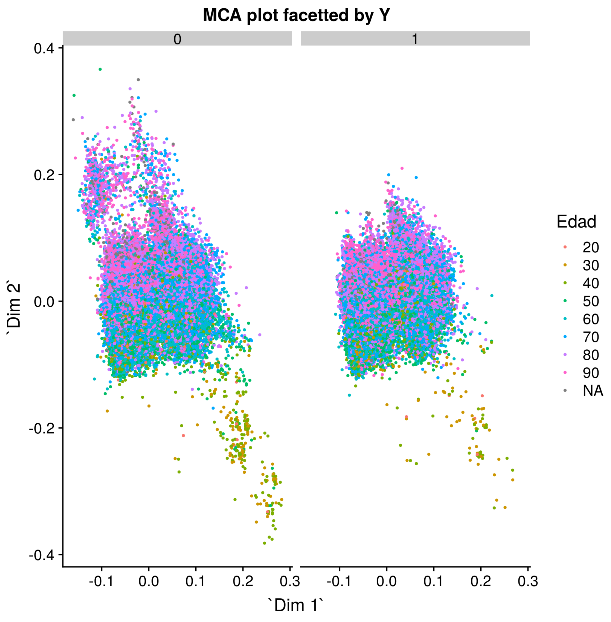
- 12 GB RAM
- 8 processors Intel(R) Core(TM) i7-6700HQ CPU at 2.60GHz

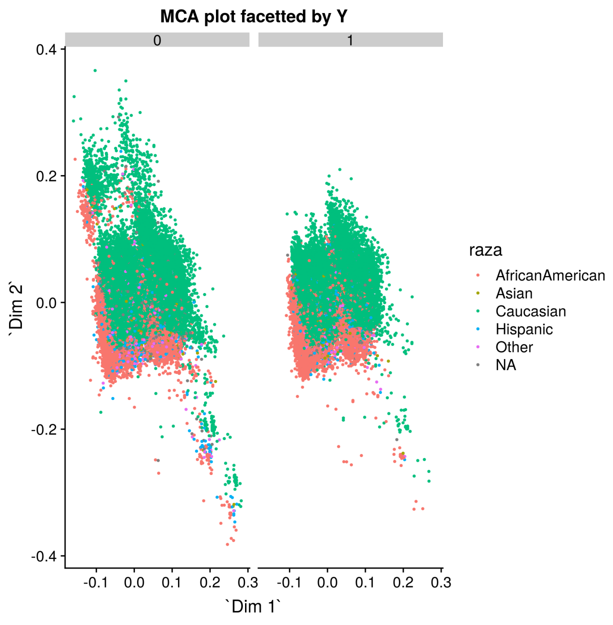
```r
sessionInfo()
```

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=es_ES.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=es_ES.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=es_ES.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] kableExtra_0.9.0  pheatmap_1.0.10   cowplot_0.9.2     FactoMineR_1.41
##  [5] stringr_1.3.1     data.table_1.11.8 ade4_1.7-11       tibble_1.4.2
##  [9] tidyr_0.8.1       waffle_0.8.0      magrittr_1.5      dplyr_0.7.7
## [13] viridis_0.5.1     viridisLite_0.3.0 ggplot2_3.1.0     knitr_1.20
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5     purrr_0.2.5          lattice_0.20-35
##  [4] colorspace_1.3-2     htmltools_0.3.6      rlang_0.3.0.1
##  [7] pillar_1.3.0         glue_1.3.0           withr_2.1.2
## [10] RColorBrewer_1.1-2   bindrcpp_0.2.2       bindr_0.1.1
## [13] plyr_1.8.4           munsell_0.5.0        gtable_0.2.0
## [16] rvest_0.3.2          codetools_0.2-15     leaps_3.0
## [19] evaluate_0.10.1      extrafont_0.17       curl_3.2
## [22] Rttf2pt1_1.3.6       highr_0.7            Rcpp_0.12.19
## [25] readr_1.1.1          scales_1.0.0.9000    backports_1.1.2
## [28] flashClust_1.01-2    formatR_1.5          scatterplot3d_0.3-41
## [31] gridExtra_2.3        hms_0.4.2            digest_0.6.18
## [34] stringi_1.2.4        grid_3.4.4           rprojroot_1.3-2
## [37] tools_3.4.4          lazyeval_0.2.1       cluster_2.0.7-1
## [40] crayon_1.3.4         extrafontdb_1.0      pkgconfig_2.0.2
## [43] MASS_7.3-48          xml2_1.2.0           httr_1.3.1
## [46] rstudioapi_0.8       assertthat_0.2.0     rmarkdown_1.10
## [49] R6_2.3.0             compiler_3.4.4
```

4 Appendix A


MCA plot facetted by race
AfricanAmerican
Asian
Caucasian
Hispanic
Other
NA
`Dim 2`
`Dim 1`
raza
AfricanAmerican
Asian
Caucasian
Hispanic
Other
NA


MCA plot facetted by Y
0
1
`Dim 2`
`Dim 1`
Edad
20
30
40
50
60
70
80
90
NA



5 Appendix B

```r
preprocess_data <- function(x_train, x_test, etiquettes = FALSE) {

    if (etiquettes) {
        ## Preprocess etiquettes
        print("Processing etiquettes")
        train_etiquettes <- x_train[, c("etiqueta_1", "etiqueta_2", "etiqueta_3")]
        test_etiquettes <- x_test[, c("etiqueta_1", "etiqueta_2", "etiqueta_3")]
        rownames(train_etiquettes) <- x_train$identificador
        rownames(test_etiquettes) <- x_test$identificador
        unique_etiquettes <- train_etiquettes %>% unlist %>% unique %>% sort

        etiquettes_template <- rep(0, length(unique_etiquettes))
        names(etiquettes_template) <- unique_etiquettes
        etiquettes_proc <- lapply(list(train = train_etiquettes, test = test_etiquettes),
            function(y) {
                res <- y %>% as.matrix %>% apply(., 1, function(x) {
                  local_count <- table(x)
                  et <- etiquettes_template
                  # drop any etiquette that's not in the training set
                  local_count <- local_count[names(local_count) %in% names(et)]
                  et[names(local_count)] <- local_count
                  return(et)
                }) %>% t %>% unlist
                colnames(res) <- paste0("et_", colnames(res))
                res <- res %>% apply(., 2, as.factor) %>% as.data.frame
                return(res)
            })

        # etiquettes_proc$train %>% View OK
    }
```

```r
## Preprocess nominals
print("Processing nominals")
# Drop nominal 4 for now Should remove the last column produced with each nominal
train_nominals_one_hot <- x_train %>% select(nominal_1:nominal_3) %>% acm.disjonctif() %>%
    apply(., 2, as.factor) %>% as.data.frame
test_nominals_one_hot <- x_test %>% select(nominal_1:nominal_3) %>% acm.disjonctif() %>%
    apply(., 2, as.factor) %>% as.data.frame

missing_nominal_test <- colnames(train_nominals_one_hot)[!colnames(train_nominals_one_hot) %in%
    colnames(test_nominals_one_hot)]
missing_test <- as.data.frame(matrix(0, nrow = nrow(x_test), ncol = length(missing_nominal_test)))
colnames(missing_test) <- missing_nominal_test
test_nominals_one_hot <- cbind(test_nominals_one_hot, missing_test)[colnames(train_nominals_one_hot

nominals_proc <- list(train = train_nominals_one_hot, test = test_nominals_one_hot)

nominals_proc$train %>% class
# OK

## Preprocess drugs (farmacos)
print("Processing drugs")

train_drugs <- x_train[, grep(pattern = "farmaco", x = colnames(x_train))]
test_drugs <- x_test[, grep(pattern = "farmaco", x = colnames(x_test))]

# Drop drugs with no variability (non-informative)
train_drugs <- train_drugs[, train_drugs %>% apply(., 2, function(x) length(unique(x))) >
    1]
test_drugs <- test_drugs[, colnames(train_drugs)]

# Replace strings with integer
drugs_proc <- list(train = train_drugs, test = test_drugs) %>% lapply(function(x) {
    x[x == "No"] <- "-1"
    x[x == "Down"] <- "0"
    x[x == "Steady"] <- "1"
    x[x == "Up"] <- "2"
    x <- x %>% apply(., 2, as.integer) %>% as.data.frame
})

drugs_proc$train %>% class

## Preprocess ordinal
print("Processing ordinals")

train_ordinal <- x_train %>% select(ordinal_1:ordinal_2)
test_ordinal <- x_test %>% select(ordinal_1:ordinal_2)
ordinals_proc <- lapply(list(train = train_ordinal, test = test_ordinal), function(x) {
    x[x == "None"] <- "0"
    x[x == "Norm"] <- "1"
    x[x == ">7" | x == ">200"] <- "2"
    x[x == ">8" | x == ">300"] <- "3"
    x <- x %>% apply(., 2, as.integer) %>% as.data.frame
    x
```

```
})

ordinals_proc$train %>% class


## Preprocess binary
print("Processing binaries")

train_binary <- x_train %>% select(binary_1:binary_3) %>% apply(., 2, function(x) as.factor(as.integ
    1)) %>% as.data.frame
test_binary <- x_test %>% select(binary_1:binary_3) %>% apply(., 2, function(x) as.factor(as.integer
    1)) %>% as.data.frame
binary_proc <- list(train = train_binary, test = test_binary)

binary_proc$train %>% class

## Preprocess counter
print("Processing counters")

counter_proc <- list(train = x_train %>% select(counter_1:counter_7), test = x_test %>%
    select(counter_1:counter_7))
counter_proc$train %>% class

## Preprocess race
print("Processing race")

race_proc <- list(train = x_train %>% select(raza) %>% acm.disjonctif() %>% apply(.,
    2, as.factor) %>% as.data.frame, test = x_test %>% select(raza) %>% acm.disjonctif() %>%
    apply(., 2, as.factor) %>% as.data.frame)

race_proc$train %>% class

## Preprocess sex
print("Processing sex")

sex_proc <- list(train = x_train %>% select(sexo) %>% apply(., 2, function(x) as.factor(as.integer(a
    1)) %>% as.data.frame, test = x_test %>% select(sexo) %>% apply(., 2, function(x) as.factor(as.i
    1)) %>% as.data.frame)

## Preprocess age
print("Processing age")

train_age <- x_train$edad_integer
train_age_mean <- mean(train_age, na.rm = T)
train_age[is.na(train_age)] <- train_age_mean
test_age <- x_test$edad_integer
test_age[is.na(test_age)] <- train_age_mean

age_proc <- list(train = data.frame(edad_integer = train_age), test = data.frame(edad_integer = test

datasets <- c("train", "test")
if (etiquettes) {
    processed_datasets <- lapply(datasets, function(x) {
```

```r
            cbind(race_proc[[x]], sex_proc[[x]], age_proc[[x]], nominals_proc[[x]],
                counter_proc[[x]], drugs_proc[[x]], ordinals_proc[[x]], binary_proc[[x]],
                etiquettes_proc[[x]])
        })
    } else {
        processed_datasets <- lapply(datasets, function(x) {
            cbind(race_proc[[x]], sex_proc[[x]], age_proc[[x]], nominals_proc[[x]],
                counter_proc[[x]], drugs_proc[[x]], ordinals_proc[[x]], binary_proc[[x]])
        })
    }

    names(processed_datasets) <- datasets
    return(processed_datasets)
}
```