Data Science and Society Master's Thesis - Alzheimer's Prediction

Shadi Saee

27 - 09 - 2019

Loading Packages

```
#install.packages(corrplot)
#install.packages("table1")
#install.packages("Hmisc")  #label function for tables
#install.packages("splitstackshape")  stratifiation


library(ggplot2)
library(dplyr)
library(corrplot)
library(table1)
library(Hmisc)
library(stats)
library(splitstackshape)
```

Loading Data

```
adnimerge <- read.delim("ADNIMERGE.csv", sep = ',', stringsAsFactors = FALSE)
vitals <- read.delim("VITALS.csv", sep = ',', stringsAsFactors = FALSE)
postproc <- read.delim("Biospecimen_Results/ADMC_BA_POSTPROC_06_28_18.csv",
              sep = ',', stringsAsFactors = FALSE)
```

Data Preparation and Cleaning: Subsetting the Relevant Variables and Instances

All the loaded datasets contain variables that are irrelevant for the current task. I subset only the relevant variables. The adnimerge dataset contains many key variables like demographic information, diagnosis, biomarkers, and neuropsychological test results. I select diagnosis (+ baseline diagnosis which has a seperate variable), age, gender, education, APOE4 genotype, and biomarker information at baseline. The vitals dataset contains information on weight and height of the participants neccessary to create the BMI variable. The postproc dataset contains the log-transformed BA measurements taken at baseline and information about the extraction process. I only keep the measurements of the BAs and their respective ratios. In all datasets I also keep the unique identifier RID and the "VISCODE" variable that encodes which visit the instance belongs too. Each visit entails a new instance.

I also combine the diagnoses late MCI and early MCI to "MCI" and turn the gender variable into a factor, as it was previously encoded as character.

```r
# Discard irrelevant variables
vitals2 <- vitals %>% select(RID, VISCODE, VSWEIGHT, VSWTUNIT, VSHEIGHT, VSHTUNIT)
postproc2 <- postproc[ ,-c(1,5:9,34:57)]
adnimerge2 <- adnimerge[ ,c(1,3,8:11,15,60,66,103:105,109:110,112)] %>%
  mutate(DX_bl = ifelse(DX_bl == "LMCI"|DX_bl == "EMCI", 'MCI', DX_bl)) %>%
  rename("GENDER" = PTGENDER, "EDUCATION" = PTEDUCAT,
  "ADAS13" = ADAS13_bl) %>%
  mutate(GENDER = factor(GENDER), APOE4 = factor(APOE4)) %>%
  arrange(RID, VISCODE)
```

I subset and manipulate the datasets individually at first and merge them together later.

I exclude patients with subjective memory complaints and include only patients who have converted to AD within three years. I do this in two steps: first I filter for patients that have diagnostic information available at three years. In a second step I check for patients who converted from MCI to AD earlier than three years after BL.

```r
# Remove patients with SMC and include only patients with data available for
#three years
adnimerge3 <- adnimerge2 %>%
  filter(DX_bl != 'SMC')  %>%
  filter(Years_bl <= 3.2) %>%
  group_by(RID) %>%
  mutate(max_data = (max(M))) %>%
  filter(max_data == 36) %>%
  select(-c(max_data, Years_bl)) %>%
  ungroup(RID)

# Also include MCI patients who converted to dementia earlier than three years:
#First, get all RIDs of people who converted from MCI to dementia within three #years. Then, filter the
# who were already in the adnimerge3 object

`%not in%` <- function (x, table) is.na(match(x, table, nomatch=NA_integer_))
# Source: https://stackoverflow.com/questions/5831794/opposite-of-in

MCI_converters_RID <- adnimerge2 %>%
  filter(DX_bl == "MCI") %>%
  filter(Years_bl <= 3.2) %>%
  filter(DX == "Dementia") %>%
  pull(RID)

MCI_converters <- adnimerge2 %>%
  filter(RID %in% MCI_converters_RID) %>%
  filter(RID %not in% adnimerge3$RID) %>%
  select(-Years_bl)

# Check  how many patients are in the MCI_converters that were excluded in adnimerge3
length(unique(MCI_converters$RID))
```

```
## [1] 86
```

```r
#Check if both dataframes have same variables to be able to append them
all_equal(names(adnimerge3),names(MCI_converters))
```

```
## [1] TRUE
```

```r
# Append MCI_converters to adnimerge3
adnimerge4 <- rbind(adnimerge3, MCI_converters)
```

In the next step I process the information from the vitals table to create the BMI variable. This entails first changing all measurements to the same scale (some measurements are in cm and others in inches) and then creating the BMI variable based on weight and height. Three instances were mistakenly coded as inches when they were actually already in cm, resulting in heights above two meters fifty, therefore I mutated those back to the original value.

```r
# Process info for BMI meausure
bmi <- vitals2 %>%
  select(RID, VISCODE,VSHEIGHT, VSHTUNIT, VSWEIGHT, VSWTUNIT) %>%
  mutate(VSHEIGHT = ifelse(VSHTUNIT == 1, VSHEIGHT * 2.54, VSHEIGHT)) %>%
  mutate(VSWEIGHT = ifelse(VSWTUNIT == 1, VSWEIGHT / 2.205, VSWEIGHT)) %>%
  filter(VSHEIGHT != -4 & !is.na(VSHEIGHT)) %>%
  filter(VISCODE %in% c("bl", "sc", "v01")) %>%
  arrange(RID)

# Check if heights are correct
too_tall <- bmi[bmi$VSHEIGHT > 200,]
too_tall
```

```
##       RID VISCODE VSHEIGHT VSHTUNIT  VSWEIGHT VSWTUNIT
## 414   721      sc   208.28        1 121.08844        1
## 628  1094      sc   204.47        1  85.71429        1
## 1918 4981     v01   213.36        1  58.95692        1
## 2298 6134      sc   419.10        1  70.29478        1
## 2859 6782      sc   412.75        1  89.02494        1
```

```r
# Convert 3 instances back to original value
too_tall_ids <- too_tall[3:5,1]

mutate_cond <- function(.data, condition, ..., envir = parent.frame()) {
  condition <- eval(substitute(condition), .data, envir)
  .data[condition, ] <- .data[condition, ] %>% mutate(...)
  .data
}
# Source: https://stackoverflow.com/questions/34096162/dplyr-
#mutate-replace-several-columns-on-a-subset-of-rows

bmi <- bmi %>%
  mutate_cond(RID %in% too_tall_ids, VSHEIGHT = VSHEIGHT/2.54)
bmi[bmi$RID %in% too_tall_ids,]
```

```
##       RID VISCODE VSHEIGHT VSHTUNIT VSWEIGHT VSWTUNIT
## 1918 4981     v01     84.0        1 58.95692        1
## 2298 6134      sc    165.0        1 70.29478        1
## 2859 6782      sc    162.5        1 89.02494        1
```

```r
bmi <- bmi %>%
  mutate(VSHEIGHT = VSHEIGHT/100) %>%
  mutate(BMI = VSWEIGHT/(VSHEIGHT*VSHEIGHT)) %>%
  mutate(VISCODE = "bl") %>%
  select(-c(VSHTUNIT,VSWTUNIT))
```

In the next step I merge all three datasets via left join and then subset the merged dataset to include only

participants with BA meaurements available.

```
# Merge data
total <- adnimerge4 %>%
  left_join(bmi, by = c("RID", "VISCODE")) %>%
  left_join(postproc2, by = c("RID", "VISCODE")) %>%
  arrange(RID, VISCODE)

count(distinct(total,RID)) #851
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   851
```

```
# Only participants with BA info available
final <- total[total$RID %in% postproc2$RID,]

count(distinct(final,RID)) #820
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   820
```

Looking at some distributions

```
# Distribution of Baseline Diagnosis:
bl_distribution <- final %>%
  group_by(DX_bl) %>%
  summarise(n_distinct(RID))
bl_distribution
```

```
## # A tibble: 3 x 2
##   DX_bl `n_distinct(RID)`
##   <chr>             <int>
## 1 AD                   20
## 2 CN                  191
## 3 MCI                 609
```

```
# Distribution of Diagnosis at 3 years:
m36_distribution <- final %>%
  filter(VISCODE == "m36") %>%
  group_by(DX) %>%
  summarise(n_distinct(RID))
m36_distribution
```

```
## # A tibble: 4 x 2
##   DX       `n_distinct(RID)`
##   <chr>                <int>
## 1 ""                      24
## 2 CN                     199
## 3 Dementia               182
## 4 MCI                    338
```

```
# Distribution of Diagnosis of only MCI patients at three years
MCI_patients <- final %>%
  filter(DX_bl == 'MCI')
```

```
MCI_DX_M36 <- MCI_patients %>%
  filter(VISCODE == "m36") %>%
  group_by(DX) %>%
  summarise(n_distinct(RID))
MCI_DX_M36
```

```
## # A tibble: 4 x 2
##   DX       `n_distinct(RID)`
##   <chr>                <int>
## 1 ""                      10
## 2 CN                      36
## 3 Dementia               170
## 4 MCI                    316
```

Next I create a categorical conversion variable that enocodes whether the patients converted from MCI to AD within three years or not. First, I create a new variable "Convert" that has value 1 if the diagnosis DX in the given row is "Dementia". Then I create a function that changes all values of the "Convert" variable for each patient to 1 if the diagnosis for that patients equals "Dementia" at any point in time.

```
MCI_patients <- group_by(MCI_patients, RID) %>%
  mutate(Convert = ifelse(DX == "Dementia", 1, 0)) %>%
  ungroup()

create_convert <- function(df){
  for (i in unique(df$RID)){
    if (any(df[df$RID == i,8] == "Dementia")){
      df[df$RID == i,43] <- 1
    }
  }
  df
}

MCI_patients <- create_convert(MCI_patients)
MCI_patients$Convert <- factor(MCI_patients$Convert)

#for(i in unique(MCI_patients$RID)){
#  subgroup <-  filter(MCI_patients, RID == i)
#  print(subgroup %>% select(RID, VISCODE, DX, Convert))
#}

MCI_patients %>% group_by(Convert) %>% summarise(n_distinct(RID))
```

```
## # A tibble: 2 x 2
##   Convert `n_distinct(RID)`
##   <fct>               <int>
## 1 0                     350
## 2 1                     259
```

Now, all the information we need is encoded at baseline visit. Thus, I can subset the MCI patients object to only include the baseline visit and delete some superfuous variables

```
MCIBL <- MCI_patients %>%
  filter(VISCODE == "bl") %>%
  select(-c(VISCODE,DX_bl,DX, Month_bl,M,PLATFORM_ID)) %>%
  select(RID:ADAS13, Convert, ABETA_bl:TDCA_DCA)
```

Summary Statistics and EDA

I create the table with an overview of demographics of the MCI participants stratified by Conversion status using the "table1" package.

```
MCIBL$Convert <- factor(MCIBL$Convert, levels = c(0,1),
                        labels = c("MCI Non-Converter", "MCI Converter"))


labels <- list(variables = list(AGE = "Age (Years)", GENDER = "Sex",
                EDUCATION = "Education", BMI = "BMI (kg/m&sup2;)",                                    ADAS13

strata <- c(list(Total = MCIBL), split(MCIBL, MCIBL$Convert))


render_cont <- function(x) {
    with(stats.apply.rounding(stats.default(x), digits = 2), c("",
        "Mean (SD)"= sprintf("%s (&plusmn; %s)", MEAN, SD)))
}

render_cat <- function(x) {
    c("", sapply(stats.default(x), function(y) with(y,
        sprintf("%d (%0.0f %%)", FREQ, PCT))))
}

table1 <- table1(strata, labels, render.continuous = render_cont, render.categorical = render_cat)
table1
```

```
## [1] "<table class=\"Rtable1\">\n<thead>\n<tr>\n<th class='rowlabel firstrow lastrow'></th>\n<th class
```

```
# Stratified by Convert and Sex
demog_strat_table1 <- table1(~ AGE + EDUCATION + BMI + ADAS13 | GENDER * Convert, data = MCIBL, overall
demog_strat_table1
```

```
## [1] "<table class=\"Rtable1\">\n<thead>\n<tr>\n<th class=\"grouplabel\"></th>\n<th colspan=\"2\" clas
```

```
stratified_df <-stratified(MCIBL,c("GENDER", "Convert"),0.9999)

#table1(MCIBL, GENDER, EDUCATION, BMI, splitby = ~Convert)
```

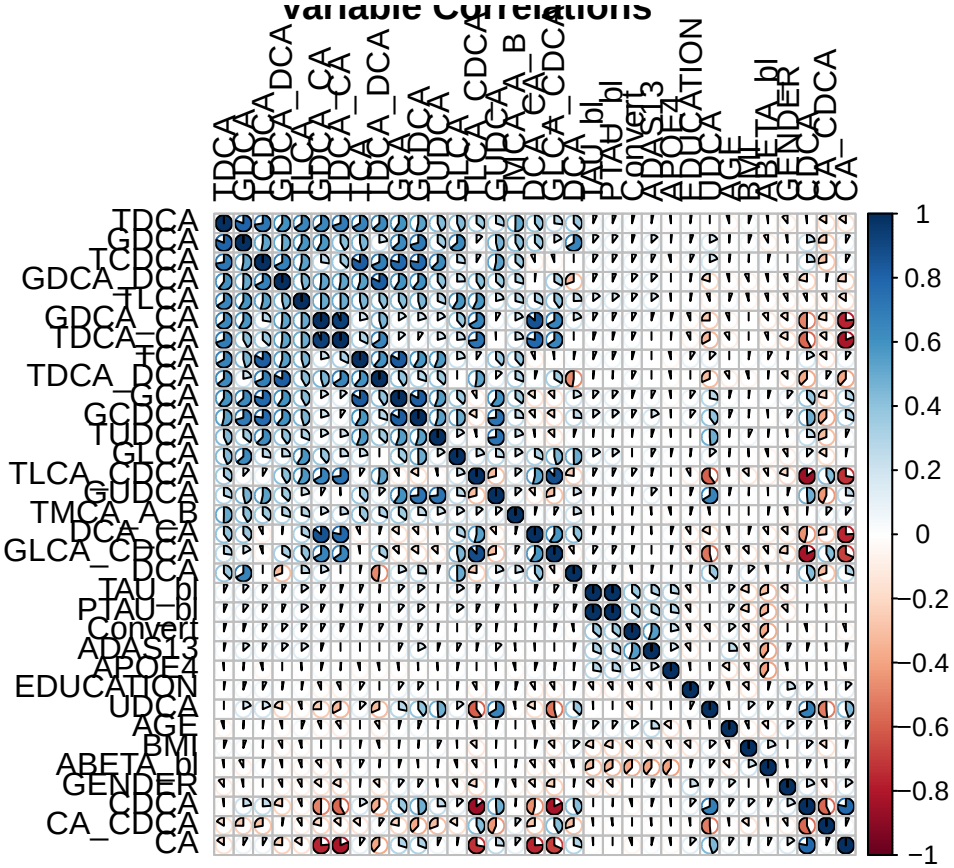
```
BA_strat_table <- table1(~ CA + CDCA+ DCA + GCA + GCDCA + GDCA + GLCA + GUDCA + TCA + TCDCA + TDCA + TL
                         data = MCIBL, overall = "Total")
BA_strat_table

BA_labels <- list(variables = list(TMCA_A_B = "TMCA:A:B", CA_CDCA = "CA:CDA",
                                   DCA_CA = "DCA:CA", GLCA_CDCA = "GLCA:CDDCA",
                                   GDCA_CA = "GDCA:CA", GDCA_DCA = "GDCA:DCA",
                                   TDCA_CA = "TDCA:CA", TLCA_CDCA = "TLCA:CDCA",
                                   TDCA_DCA = "TDCA:DCA"))
BA_strata <- c(list(Total = MCIBL), split(MCIBL, MCIBL$Convert))

BA_table <- table1(BA_strata, BA_labels, render.continuous = render_cont)
BA_table
```

Correlation Matrix

```
# Look at correlation of mean values of each feature #

predictors <- select(MCIBL,AGE:ADAS13,BMI,ABETA_bl:PTAU_bl,CA:TDCA_DCA, Convert)


predictors[] <- lapply(predictors,as.numeric)
```

```
## Warning in lapply(predictors, as.numeric): NAs introduced by coercion
```

```
correlations <- cor(predictors, use = "complete.obs")

correlation_plot <- corrplot(correlations, method = "pie",
                             title = "Variable Correlations", tl.col = "black",
                             order = "FPC", type = "full")
```




```
#correlation_plot


#corr2 <- rcorr(as.matrix(predictors))

#corr2$r # Extract correlations
#corr2$P # extract p values
```