Lab Assigment 3

Akshay Yadav, Ganesh Krishnan, Vinny Paris, Matt Stuart

The given data on scores for students measured before and after two exams has 2 distinct types of columns. One type is where the variables are expected to have different values for before (**pre**) and after (**post**) taking the exams. The columns like **total** and the columns containing the individual answer scores are included in this type. The other type of columns have variables with values that are expected to remain same before and after the exams. This category includes columns like **Gender, Normalized Changes, Characteristic**.

We have designed a function *read_exam_data()* that will read in the data from excel file and assign missing values. The function performs following operations and returns a dataframe.

- Takes the file name and the sheet number to read the data. Only the columns containing the student-ids, the type of the score *(pre/post)* information, the total scores, the gender information, the two treatment information, the characteristic information and normalized change scores are read in. The first 2 columns are assigned names - *id* and *scoretype*.
- The values in columns total scores, gender, treatment 1, treatment 2, normalized changes and characteristic are given only once, for each student-id in the *pre* section. These values need to be copied for the same student-ids in the *post* section. This can be easily and efficiently done using the full join function on 2 tables: One table containing the previously mentioned one value columns with the student-id as the key and the second table containing the *scoretype* and the *total* score columns with the student ids as the key. The full join on these 2 tables by using the student-ids will copy all the missing values to the post section of the table.
- For getting the first table, for the full join, a small support function *get_unique_values()* that will return unique values for each student-id is designed that can handle senarios like unknown genders and two unique values etc.

```
library(tidyverse)
library(ggplot2)

get_unique_values<-function(x){
  u_len<-length(unique(x[!is.na(x)]))
  if(u_len==0){
    return("NA")
  }else if(u_len==2){
    return(paste(unique(x[!is.na(x)]),collapse=","))
  } else {
    return(unique(x[!is.na(x)]))
  }
}

read_exam_data <- function(file="Spreadsheets/FileOne.xlsx", sem=1, exam=1){
score_table<-readxl::read_excel(path=file, sheet=sem)
names(score_table)[1]<-"id"
names(score_table)[2]<-"scoretype"
colnames(score_table) <- tolower(colnames(score_table))
score_table <- score_table %>% select(id, scoretype, starts_with("gen"),
                  starts_with("treatme"),
                  starts_with("charac"),
                  starts_with("tot"),
                  starts_with("norm"))
```

```
names(score_table)<- c("id", "scoretype", "gender",
                       "treat1", "treat2",
                       "characteristic",
                       "total","normchange")

uval_table<-score_table %>% group_by(id) %>%
  select(id,normchange,gender,treat1,treat2,characteristic) %>%
  summarise(gender=get_unique_values(unique(gender[!is.na(gender)])),
            normchange=get_unique_values(unique(normchange[!is.na(normchange)])),
            characteristic=get_unique_values(unique(characteristic[!is.na(characteristic)])),
            treat1=get_unique_values(unique(treat1[!is.na(treat1)])),
            treat2=get_unique_values(unique(treat2[!is.na(treat2)])))


nuval_table<-score_table %>% select(id,scoretype,total)
new_score_table<-full_join(uval_table,nuval_table,by="id")
new_score_table$semester<-rep(sem,dim(new_score_table)[1])
new_score_table$exam<-rep(exam,dim(new_score_table)[1])

return(new_score_table)
}
```

Calling the *read_exam_data()* function for each sheet (semester) for both the exams to prepare a master data table containing all the data from 8 data tables.

```
master_table<-read_exam_data(file="Spreadsheets/FileOne.xlsx", sem=1, exam=1)
master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileOne.xlsx", sem=2, exam=1))
master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileOne.xlsx", sem=3, exam=1))
master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileOne.xlsx", sem=4, exam=1))

master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileTwo.xlsx", sem=1, exam=2))
master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileTwo.xlsx", sem=2, exam=2))
master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileTwo.xlsx", sem=3, exam=2))
master_table<-rbind(master_table,read_exam_data(file="Spreadsheets/FileTwo.xlsx", sem=4, exam=2))
```

All the character data containing columns need to be converted to lower-case for data consistency

```
master_table<-data.frame(lapply(master_table, function(v) {
  if (is.character(v)) return(tolower(v))
  else return(v)
}))
```

Changing the ids for the forth semester students since they were coded differently for the two exams

```
master_table$id <- ifelse(as.numeric(master_table$id) >= 41000,
                          as.numeric(master_table$id) - 1000,
                          as.numeric(master_table$id))
```

Converting the *gender*, *semester*, and *exam* columns as factor data for consistency and to get distinct colors in the graphs

```
master_table$gender<-as.factor(master_table$gender)
master_table$semester<-as.factor(master_table$semester)
master_table$exam<-as.factor(master_table$exam)
```

Plotting the *pre v/s post* values for both the exams faceted by semester. The different colors represent different exams and the different shapes of the points represent the different genders

```
master_table %>% spread(scoretype,total) %>%
  ggplot(aes(x = pre, y = post,color=exam)) +
  geom_point(aes(shape=gender)) +
  facet_wrap(~semester)
```