LEARN Center
EPFL

Data Pre-Processing

Private Data (Year 1 & Year 2)

Himanshu Verma, Cécile Hardebolle

Flipped Classroom Project

May 27, 2020

Contents

1 R Package Imports

In this section, we will import all the required packages for importing, cleaning, and pre-processing the data.

```
library(readxl)
library(dplyr)
library(tidyr)
library(ggplot2)
library(scales)
library(gridExtra)
library(gplots)
library(here)
library(stringi)
```

2 Background

In this document we will do the following:

1. Initially, parse the .xlsx files for both volunteers and non-volunteers.
2. Perform the data cleaning and filtering operations.
3. Restructure and re-name the different variables, and generate some new variables (columns).
4. Remove questions with low DI.
5. Compute Scores for different parts (A, B, and C).
6. Finally, compute the normalized scores.

3 Reading Volunteer Data

As the flipped classrooms were conducted in the academic years 2017-18 (Year1) and 2018-19 (Year2), we will use this notation henceforth.

The raw data is in the form of Excel files – one for each year. As a first step, we will import the data and save it in a dataframe. Followed by the cleaning and pre-processing, and finally, exporting the data as a CSV.

```
# Specifying the temporary path where these files can be found.
# Please note that this path is temporary as data exists on an external storage.
# Furthermore the filenames have been modified to include the Year information as mentioned above.

path = paste(here(), "/Data/Personal/", sep = "")
dt.y1 = read_excel(paste(path, "VolunteersList-Y1.xlsx", sep = ""), sheet = "Tous")
dt.y2 = read_excel(paste(path, "VolunteersList-Y2.xlsx", sep = ""), sheet = "Liste")
```

4 Data Cleaning

In this section, we clean the data by removing the unnecessary columns. We will not filter out specific cases (rows) yet, which will follow in the next sections.

Since the two datasets have their own specificities in terms of formatting and organization, we will deal with them individually in the next sub-sections.

4.1 Year1 (2017-18)

As the first step, we remove the unnecessary columns, especially related to the calculations done in the table at the extreme right of the `.xlsx` file.

```
dt.y1[,c(18:23)] = NULL
```

As the column names contain characters with accents, we will first rename all the columns.

```
names(dt.y1) <- c("Serial.No", "Full.Name", "Gender", "SCIPER",
                  "Section", "Semester", "Email", "Immatriculation",
                  "Code.BA", "Diploma.Obtained", "Category", "Country.Diploma",
                  "Temp.Code.1", "Temp.Code.2", "Temp.Code.3", "Selected",
                  "Random")
```

Next, we select the useful columns.

```
# The first column with the serial numbers is not necessary.
# Name, Email, Immatriculation (Drop-outs), Semester, and Random are not needed.
# Finally, the columns specifying the numeric codes for Code BA will also not be needed.

dt.y1 = dt.y1 %>% select(Gender, SCIPER, Section,
                         Code.BA, Diploma.Obtained, Category,
                         Country.Diploma, Selected)
```

4.1.1 Gender

Since, the names used in the two files are different. Therefore, we will standardize some of the variables.

```
# Gender variable will have two values: F or M
dt.y1$Gender = ifelse(dt.y1$Gender == "Masculin", "M", "F")
```

4.1.2 Code.BA

`Code.BA` values are represented in singular/plural form in the two files. We will standardize them as well using the `recode` function.

```
# Code.BA variable will have four values: New, Repeating, Ex-CMS, and Ex-MAN
dt.y1$Code.BA = recode(dt.y1$Code.BA,
                       "ex-CMS" = "Ex-CMS",
                       "ex-MAN" = "Ex-MAN",
                       "Nouveau" = "New",
                       "Redoublant" = "Repeating")

# Overview of the variable:
summary(factor(dt.y1$Code.BA))

##    Ex-CMS    Ex-MAN       New Repeating      NA's
##        13        46       415        44         1
```

4.1.3 Diploma.Obtained

`Diploma.Obtained` values have some inconsistencies and contain acented characters. I will recode them next.

NOTE: In the first year, there are two less categories for the variable `Diploma.Obtained`.

```
# Recoding the values for the variable Diploma.Obtained.
# The new 12 levels can be found in the code below.

dt.y1$Diploma.Obtained = recode(dt.y1$Diploma.Obtained,
                                "Bacc. ?tranger" = "Bacc.Etranger",
                                "Bachelor ?tranger" = "Bachelor.Etranger",
                                "Mat. Commission suisse de maturit?" =
                                  "Mat.Commission.Suisse",
                                "Mat. reconnue opt. arts visuels" =
                                  "Mat.Opt.Arts.Visuels",
                                "Mat. reconnue opt. biologie et chimie" =
                                  "Mat.Opt.Biologie.Chimie",
                                "Mat. reconnue opt. ?conomie et droit" =
                                  "Mat.Opt.Economie.Droit",
                                "Mat. reconnue opt. langue moderne" =
                                  "Mat.Opt.Langue.Moderne",
                                "Mat. reconnue opt. langues anciennes" =
                                  "Mat.Opt.Langue.Ancienne",
                                "Mat. reconnue opt. philo / psycho / p?dagogie" =
                                  "Mat.Opt.Philosophy.Psychology.Pedagogy",
                                "Mat. reconnue opt. physique et math" =
```

```
                                  "Mat.Opt.Physique.Math",
                                "Maturit? professionnelle" = "Mat.Prof",
                                "Passerelle maturit? professionnelle - hautes ?coles universitaires" =
                                  "Mat.Prof.Haute.Ecole")

# The following code will give an overview of the different categories
# and how many students are part of each.
summary(factor(dt.y1$Diploma.Obtained))

##                                                  Bacc. Ã‰tranger
##                                                             312
##                                               Bachelor Ã‰tranger
##                                                               1
##                                 Mat. Commission suisse de maturitÃ©
##                                                               6
##                               Mat. reconnue opt. Ã‰conomie et droit
##                                                              19
##                       Mat. reconnue opt. philo / psycho / pÃ©dagogie
##                                                               3
##                                             Mat.Opt.Arts.Visuels
##                                                               3
##                                          Mat.Opt.Biologie.Chimie
##                                                              56
##                                          Mat.Opt.Langue.Ancienne
##                                                               7
##                                           Mat.Opt.Langue.Moderne
##                                                              12
##                                            Mat.Opt.Physique.Math
##                                                              94
##                                        MaturitÃ© professionnelle
##                                                               2
## Passerelle maturitÃ© professionnelle - hautes Ã‰coles universitaires
##                                                               4
```

4.1.4 Reached.Adulthood

Since the Year1 table did not have any values for the `Reached.Adulthood` variabale, we will simply add the column to maintain homogeneity and populate the column with NAs.

```
# Adding NA values for the following variable.

dt.y1$Reached.Adulthood = NA
```

4.1.5 Course.Year

Now, we will add a *categorical* variable to indicate if the data belongs to Year1 or Year2.

```
# Adding a categorical variable for the course year.

dt.y1$Course.Year = "Y1-2017-18"
```

4.1.6 Category

Here, we will also recode the `Category` variable so that the data is homogeneous across Year1 and Year2.

```
# Recoding the values for the variable Category.
dt.y1$Category = recode(dt.y1$Category,
                        "Autres" = "Etranger.Autres",
                        "Autres suisses" = "Suisse.Autres",
                        "France" = "France",
                        "PAM" = "Suisse.PAM")
```

```
# Overview of the changes.
summary(factor(dt.y1$Category))

## Etranger.Autres          France   Suisse.Autres      Suisse.PAM
##              51             262             112              94
```

4.1.7 Selected

This variable illustrates whether the subjects were part of the flipped condition `(value == 1)` or control condition `(value == 0)`. So, we will recode this variable with string values and subsequently create a variable that shows the experimental condition.

```
# Recoding the values for the variable 'Selected'
dt.y1$Selected = recode(dt.y1$Selected,
                        "0" = "Non",
                        "1" = "Oui")

# Creating a new variable 'Condition'
dt.y1$Condition = ifelse(dt.y1$Selected == "Non", "Control", "Flipped")

# Overview of the newly created variable.
summary(factor(dt.y1$Condition))

## Control Flipped
##     410     109
```

4.2 Year2 (2018-19)

As the first step, we remove the unnecessary columns, especially related to the calculations done in the table at the extreme right of the `.xlsx` file.

```
dt.y2[,c(17:22)] = NULL
```

As the column names contain characters with accents, we will first rename all the columns.

```
names(dt.y2) <- c("Full.Name", "Gender", "Reached.Adulthood", "SCIPER",
                  "Section", "Email", "Subscription.Time", "Code.BA",
                  "Diploma.Obtained", "Category", "Country.Diploma", "Temp.Code.1",
                  "Temp.Code.2", "Temp.Code.3", "Random", "Selected")
```

Next, we select the useful columns.

```
# The first column with the serial numbers is not necessary.
# Name, Email, Subscription Time, and Random are not needed.
# Finally, the columns specifying the numeric codes for Code BA will also not be needed.

dt.y2 = dt.y2 %>% select(Gender, SCIPER, Section,
                         Code.BA, Diploma.Obtained, Category,
                         Country.Diploma, Selected,
                         Reached.Adulthood)
```

4.2.1 Code.BA

`Code.BA` values are represented in singular/plural form in the two files. So, we will standardize them similar to Year 1 by using the `recode` function.

```
# Code.BA variable will have four values: New, Repeating, Ex-CMS, and Ex-MAN
dt.y2$Code.BA = recode(dt.y2$Code.BA,
                       "Ex-CMS" = "Ex-CMS",
                       "Ex-MAN" = "Ex-MAN",
                       "Nouveaux" = "New",
```

```
                        "Redoublants" = "Repeating")

# Overview of the variable:
summary(factor(dt.y2$Code.BA))

##    Ex-CMS    Ex-MAN       New Repeating
##         7        38       305        22
```

4.2.2 Diploma.Obtained

`Diploma.Obtained` values have some inconsistencies and contain acented characters. I will recode them next:

```
# Recoding the values for the variable Diploma.Obtained.
# The new 12 levels can be found in the code below.

dt.y2$Diploma.Obtained = recode(dt.y2$Diploma.Obtained,
                                "Bacc. ?tranger" = "Bacc.Etranger",
                                "Mat. Commission suisse de maturit?" =
                                  "Mat.Commission.Suisse",
                                "Mat. reconnue opt. musique" =
                                  "Mat.Opt.Musique",
                                "Mat. reconnue opt. arts visuels" =
                                  "Mat.Opt.Arts.Visuels",
                                "Mat. reconnue opt. biologie et chimie" =
                                  "Mat.Opt.Biologie.Chimie",
                                "Mat. reconnue opt. ?conomie et droit" =
                                  "Mat.Opt.Economie.Droit",
                                "Mat. reconnue opt. langue moderne" =
                                  "Mat.Opt.Langue.Moderne",
                                "Mat. reconnue opt. langues anciennes" =
                                  "Mat.Opt.Langue.Ancienne",
                                "Mat. reconnue opt. philo / psycho / p?dagogie" =
                                  "Mat.Opt.Philosophy.Psychology.Pedagogy",
                                "Mat. reconnue opt. physique et math" =
                                  "Mat.Opt.Physique.Math",
                                "Maturit? professionnelle" = "Mat.Prof",
                                "Dipl?me HES" = "Diploma.HES",
                                "Autre certificat suisse" = "Autre.Suisse",
                                "Passerelle maturit? professionnelle - hautes ?coles universitaires" =
                                  "Mat.Prof.Haute.Ecole")

# The following code will give an overview of the different categories
# and how many students are part of each.
summary(factor(dt.y2$Diploma.Obtained))

##                                                 Autre.Suisse
##                                                            2
##                                               Bacc. Ã‰tranger
##                                                          226
##                                                DiplÃ´me HES
##                                                            1
##                                 Mat. Commission suisse de maturitÃ©
##                                                            1
##                                Mat. reconnue opt. Ã©conomie et droit
##                                                           10
##                          Mat. reconnue opt. philo / psycho / pÃ©dagogie
##                                                            1
##                                             Mat.Opt.Arts.Visuels
##                                                            2
##                                          Mat.Opt.Biologie.Chimie
##                                                           37
##                                          Mat.Opt.Langue.Ancienne
##                                                            4
```

```
##                                                Mat.Opt.Langue.Moderne
##                                                                     8
##                                                      Mat.Opt.Musique
##                                                                     2
##                                                 Mat.Opt.Physique.Math
##                                                                    70
##                                               MaturitÃ\u0089 professionnelle
##                                                                     6
## Passerelle maturitÃ\u0089 professionnelle - hautes Ã\u0089coles universitaires
##                                                                     2
```

4.2.3 Course.Year

Now, we will add a Categorical Variable to indicate if the data belongs to Year1 or Year2.

```
# Adding a categorical variable for the course year.

dt.y2$Course.Year = "Y2-2018-19"
```

4.2.4 Category

Here, we will also recode the `Category` variable so that the data is homogeneous across Year1 and Year2.

```
# Recoding the values for the variable Category.
dt.y2$Category = recode(dt.y2$Category,
                        "Autres" = "Etranger.Autres",
                        "Autres suisses" = "Suisse.Autres",
                        "France" = "France",
                        "PAM" = "Suisse.PAM")

# Overview of the changes.
summary(factor(dt.y2$Category))
```

```
## Etranger.Autres          France   Suisse.Autres      Suisse.PAM
##              16             210              76              70
```

4.2.5 Selected

This variable illustrates whether the subjects were part of the flipped condition `(value > 0)` or control condition `(value == NULL)`. So, we will recode this variable with string values and subsequently create a variable that shows the experimental condition.

```
# Recoding the values for the variable 'Selected'
dt.y2$Selected = ifelse(is.na(dt.y2$Selected), "Non", "Oui")

# Creating a new variable 'Condition'
dt.y2$Condition = ifelse(dt.y2$Selected == "Non", "Control", "Flipped")

# Overview of the newly created variable.
summary(factor(dt.y2$Condition))
```

```
## Control Flipped
##     171     201
```

5 Volunteers: Filtering of Undesirable Rows

In this section, we will remove all the rows from the different datasets corresponding to Year1 and Year2.

Following subsections elaborate the different steps for each year.

5.1 Year1 (2017-18)

5.1.1 Missing Values in Code.BA

`Code.BA` is an important variable, and from the summary in Section 4.1.2 we see that there is one *missing value* in the dataset. So, we will filter this out in the following code segment.

```
dt.y1.filt = dt.y1 %>% filter(!is.na(Code.BA))
```

We see that there are in total 518 students with available Code.BA values out of total 519.

5.1.2 Minors

Since we don't have this information for Year 1 students, we will not do anything with respect to minors for Year 1.

5.2 Year2 (2018-19)

5.2.1 Minors

All the students who were still minors at the time the study started in Fall 2018 will be filtered out in the following code segment:

```
# Only students who reached adulthood.
dt.y2.filt = dt.y2 %>% filter(Reached.Adulthood == "Oui")

# Only students who were minors.
dt.y2.minors = dt.y2 %>% filter(Reached.Adulthood == "Non")
```

There are in total 68 minors in Year2.

6 Subsetting & Merging DataSets (Flipped vs. Control Condition)

In this section, we will split the data across the two conditions, and export the files as CSVs.

6.1 Year1 (2017-18)

```
# Subsetting
dt.y1.flip = dt.y1.filt %>% filter(Condition == "Flipped")
dt.y1.cont = dt.y1.filt %>% filter(Condition == "Control")
```

6.2 Year2 (2018-19)

```
# Subsetting (without minors)
dt.y2.flip = dt.y2.filt %>% filter(Condition == "Flipped")
dt.y2.cont = dt.y2.filt %>% filter(Condition == "Control")

# With Minors
dt.y2.min.flip = dt.y2 %>% filter(Condition == "Flipped")
dt.y2.min.cont = dt.y2 %>% filter(Condition == "Control")
```

6.3 Merge Datasets

In this section, we will merge the two datasets and prepare clean dataframes. We will not export the datasets until they are anonymized, which we will do in the next section.

```
# Merge datasets (without minors)
dt.flip = rbind(dt.y1.flip, dt.y2.flip)
dt.cont = rbind(dt.y1.cont, dt.y2.cont)
dt.total = rbind(dt.flip, dt.cont)

# With Minors
```

```
dt.min.flip = rbind(dt.y1.flip, dt.y2.min.flip)
dt.min.cont = rbind(dt.y1.cont, dt.y2.min.cont)
dt.min.total = rbind(dt.min.flip, dt.min.cont)
```

7 Anonymizing Data

7.1 Function Definitions

In this Section, we will anonymize the `SCIPER` values to have an anonymized identifier, which although unique and reversible, will prevent the case of direct identification of students from the dataset. To accomplish this, we will define two functions which will create the unique identifier and re-generate SCIPER values from the unique ID.

The two methods that we will use are called '`GenerateUniqueID(str)`' and '`GenerateSCIPER(str)`'. I have masked these functions from the PDF.

7.2 Generate Unique IDs from SCIPER

Next, we will create a column that contains a unique and anonymized ID for each row (subject) by using the aforementioned User-Defined Function.

Since the newly-generated `ID.Anon` variable is of type `list`, we have to unlist it before we can export them as .csv files.

```
# Generate Unique IDs and remove the SCIPER Column

# Year1
dt.y1.cont$ID.Anon = lapply(dt.y1.cont$SCIPER, GenerateUniqueID)
dt.y1.cont$SCIPER = NULL
dt.y1.cont$ID.Anon = unlist(dt.y1.cont$ID.Anon)

dt.y1.flip$ID.Anon = lapply(dt.y1.flip$SCIPER, GenerateUniqueID)
dt.y1.flip$SCIPER = NULL
dt.y1.flip$ID.Anon = unlist(dt.y1.flip$ID.Anon)

# Year2 -- Without Minors
dt.y2.cont$ID.Anon = lapply(dt.y2.cont$SCIPER, GenerateUniqueID)
dt.y2.cont$SCIPER = NULL
dt.y2.cont$ID.Anon = unlist(dt.y2.cont$ID.Anon)

dt.y2.flip$ID.Anon = lapply(dt.y2.flip$SCIPER, GenerateUniqueID)
dt.y2.flip$SCIPER = NULL
dt.y2.flip$ID.Anon = unlist(dt.y2.flip$ID.Anon)

# Year2 -- With Minors
dt.y2.min.cont$ID.Anon = lapply(dt.y2.min.cont$SCIPER, GenerateUniqueID)
dt.y2.min.cont$SCIPER = NULL
dt.y2.min.cont$ID.Anon = unlist(dt.y2.min.cont$ID.Anon)

dt.y2.min.flip$ID.Anon = lapply(dt.y2.min.flip$SCIPER, GenerateUniqueID)
dt.y2.min.flip$SCIPER = NULL
dt.y2.min.flip$ID.Anon = unlist(dt.y2.min.flip$ID.Anon)

# Aggregated Data -- without minors
dt.total$ID.Anon = lapply(dt.total$SCIPER, GenerateUniqueID)
dt.total$SCIPER = NULL
dt.total$ID.Anon = unlist(dt.total$ID.Anon)

dt.cont$ID.Anon = lapply(dt.cont$SCIPER, GenerateUniqueID)
dt.cont$SCIPER = NULL
dt.cont$ID.Anon = unlist(dt.cont$ID.Anon)

dt.flip$ID.Anon = lapply(dt.flip$SCIPER, GenerateUniqueID)
dt.flip$SCIPER = NULL
```

```r
dt.flip$ID.Anon = unlist(dt.flip$ID.Anon)

# Aggregated data -- With Minors
dt.min.total$ID.Anon = lapply(dt.min.total$SCIPER, GenerateUniqueID)
dt.min.total$SCIPER = NULL
dt.min.total$ID.Anon = unlist(dt.min.total$ID.Anon)

dt.min.cont$ID.Anon = lapply(dt.min.cont$SCIPER, GenerateUniqueID)
dt.min.cont$SCIPER = NULL
dt.min.cont$ID.Anon = unlist(dt.min.cont$ID.Anon)

dt.min.flip$ID.Anon = lapply(dt.min.flip$SCIPER, GenerateUniqueID)
dt.min.flip$SCIPER = NULL
dt.min.flip$ID.Anon = unlist(dt.min.flip$ID.Anon)
```

8 Dataset Description

In this section, we will briefly describe the different variables in the datasets, their respective types, and what values can be expected in these variables.

- **Gender** (\ categorical \): This variable takes a value of either **M** or **F.**

- **SCIPER** (\ unique, numerical \): A unique 6-digit number which is unique to every student enrolled at EPFL. This column was later redacted from the datasets and replaced by variable `ID.Anon` which is an anonymized encoding of the SCIPER values.

- **Section** (\ categorical \): There are 11 sections to which the students are registered. The values of these sections are **CGC**, **DH**, **EL**, **GC**, **GM**, **IN**, **MT**, **MX**, **SC**, **SIE** and **SV**. Although, there was only 1 student from DH who was registered in the Master programme and was filtered out.

- **Code.BA** (\ categorical \): This is a unique code which indicates the status of the students when they register for the course. You can be either a *new* student. Or you are *repeating*. Also, within repeating, there are different means: *Ex-MAN* and *Ex-CMS*. It is worth noting that some of the students who failed their exams in Year1 are repeatinbg again in Year2.

- **Diploma.Obtained** (\ categorical \): The background of the students before they started at EPFL. There are 14 different categories of students' diploma, which can be aggregated into less categories based on the country of origin and the subjects studied during High School.

- **Category** (\ categorical \): This variable is an aggregation (so to speak) of the country and the subjects taken during High School. There are 4 categories: `Swiss.PAM` (Swiss Baccalaureat with Physics and Maths), `Swiss.Autres` (Swiss Baccalaureat with other backgrounds as described in `Diploma.Obtained` in detail), `France` (French Baccalaureat with maximum score of 20 and admittance at 16), and `Etranger.Autres` (Baccalaureat from other countries).

- **Country.Diploma** (\ categorical \): It is simply the country where Baccalaureat was obtained.

- **Selected** (\ binary, categorical \): This variable enables differentiation amongst *volunteers* and *non-volunteers* (people who were part of the control condition). A value of "`Oui`" indicates that they were *volunteers* and belonged to the *flipped* condition.

- **Reached.Adulthood** (\ binary, categorical \): Whether students were legally adults when they enrolled for the course. This data is only available for Year2.

- **Course.Year** (\ categorical \): The value is either `Y1-2017-18` or `Y2-2018-19` corresponding to either Year1 or Year2.

- **Condition** (\ categorical \): The experimental condition: either `Flipped` or `Control`. This variable was derived from the `Selected` variable.

- **ID.Anon** (\ unique, string \): This variable is a unique encoding of the `SCIPER` variable. This is an alphanumeric value but unique.

9 Checking for Volunteering Students in Both Year1 and Year2

Since, it is quite likely that some students who were in the `Course.Year` YEAR1 are also present in the YEAR2. This can happen as a result of the following:

- Some students from YEAR1 fail their first *semester* (**17-18-BA1**) and then spend the next semester in **MAN** before starting again in the first semester of YEAR2 (**18-19-BA1**). These students will belong to `Code.BA` EX-MAN.
- Some students from YEAR1 fail their first *year* (**17-18-BA1** and **17-18-BA2**), and then start the first semester of YEAR2 (**18-19-BA1**) as REPEATING value for `Code.BA`.

These students are a special case in the study design, and therefore they must be examined closely. In this section, we will examine this class of students.

```
# Extract all the students who attended both Year1 and Year2
# Simply, count the number of instances of ID.Anon
# If count == 2 then the person was in both Course.Years.
temp = dt.total %>% group_by(ID.Anon) %>%
  summarise(Count = n())
temp = subset(temp, temp$Count == 2)
```

We see that there are 19 students who appeared in both YEAR1 and YEAR2.

Now, if we join the `temp` dataframe with the combined (Year 1 and Year 2) dataframe, we will get all information about these students.

```
# Perform a "Join" operation of the previously generated data frame
# with the dt.total data frame to get other information.
temp.1 = merge(dt.total,
               temp,
               by = "ID.Anon")

# From the above data frame, select Course.Year and Condition
temp.2 = temp.1 %>% select(ID.Anon, Course.Year, Condition)

# Tabulating the above data frame in an interpretable manner (making it wider).
temp.3 = spread(temp.2, Course.Year, Condition)
temp.3
```

```
##      ID.Anon Y1-2017-18 Y2-2018-19
## 1   1121T616    Control    Control
## 2   1289T717    Control    Flipped
## 3   1606T906    Control    Flipped
## 4  1951T1113    Control    Control
## 5  2199T1262    Control    Flipped
## 6  2504T1445    Flipped    Control
## 7  2961T1719    Flipped    Flipped
## 8  3493T2038    Control    Flipped
## 9  3565T2085    Flipped    Control
## 10 3631T2122    Control    Flipped
## 11 4324T2538    Flipped    Control
## 12 4359T2558    Control    Flipped
## 13 4479T2630    Control    Control
## 14 4504T2645    Control    Control
## 15 4546T2671    Control    Flipped
## 16 5062T2980    Control    Flipped
## 17 5359T3158    Control    Flipped
## 18 5394T3180    Control    Control
## 19   966T522    Flipped    Control
```

From the table above, we can see that there is only 1 student who participated in both YEAR1 and YEAR2 in the FLIPPED `Condition`. Following is the code to extract this person.

```
# People who participated in both Course.Year and in "Flipped" Condition
temp.id = subset(temp.3, temp.3[,2] == "Flipped" & temp.3[,3] == "Flipped")[1,1]
```

```
# Look up other details for this person.
dt.total %>% filter(ID.Anon == as.character(temp.id))

## # A tibble: 2 x 11
##   Gender Section Code.BA Diploma.Obtained Category Country.Diploma Selected
##   <chr>  <chr>   <chr>   <chr>            <chr>    <chr>           <chr>
## 1 F      SV      New     Bacc. Ã\tranger   France   France          Oui
## 2 F      SV      Ex-MAN  Bacc. Ã\tranger   France   France          Oui
## # ... with 4 more variables: Reached.Adulthood <chr>, Course.Year <chr>, Condition <chr>,
## #   ID.Anon <chr>
```

We see that this student is a FEMALE from the SC `Section` with her diploma from FRANCE.

Next, we look specifically at students who were assigned to FLIPPED condition in YEAR1 and CONTROL condition in YEAR2.

```
# People who participated in "Flipped" followed by "Control" condition.
temp.id = subset(temp.3, temp.3[,2] == "Flipped" & temp.3[,3] == "Control")[,1]

# Look up other details for these students.
dt.total %>% filter(ID.Anon %in% temp.id) %>%
  select(ID.Anon, Gender, Section, Code.BA, Category) %>%
  arrange(ID.Anon)

## # A tibble: 8 x 5
##   ID.Anon   Gender Section Code.BA   Category
##   <chr>     <chr>  <chr>   <chr>     <chr>
## 1 2504T1445 M      MT      New       Suisse.PAM
## 2 2504T1445 M      MT      Repeating Suisse.PAM
## 3 3565T2085 M      GM      Ex-CMS    Etranger.Autres
## 4 3565T2085 M      GM      Repeating Etranger.Autres
## 5 4324T2538 F      GM      New       France
## 6 4324T2538 F      GM      Ex-MAN    France
## 7 966T522   F      SC      New       France
## 8 966T522   F      SC      Ex-MAN    France
```

Next, we will remove these people, who were selected in the FLIPPED condition in the YEAR1 and then either participated in the FLIPPED or CONTROL condition in the YEAR2. (YEAR1: FLIPPED and YEAR2: FLIPPED) and (YEAR1: FLIPPED and YEAR2: CONTROL). We do this to ensure biases due to the *primining effects* by taking the course in FLIPPED manner in the first year.

```
# People who participated in "Flipped" condition in Year1
# Doesn't matter if they were part of "Control" or "Flipped" condition in Year2.
temp.id = subset(temp.3, temp.3[,2] == "Flipped")[,1]
temp.id

## [1] "2504T1445" "2961T1719" "3565T2085" "4324T2538" "966T522"

# We will keep this list, as these people might also be removed from the
# non-volunteer list.

# Remove these IDs from datasets.
dt.total = dt.total %>% filter(!(ID.Anon %in% temp.id)) # -10
# Flipped
dt.flip = dt.flip %>% filter(!(ID.Anon %in% temp.id)) # -6
dt.y1.flip = dt.y1.flip %>% filter(!(ID.Anon %in% temp.id)) # -5
dt.y2.flip = dt.y2.flip %>% filter(!(ID.Anon %in% temp.id)) # -1
# Control
dt.cont = dt.cont %>% filter(!(ID.Anon %in% temp.id)) # -4
dt.y1.cont = dt.y1.cont %>% filter(!(ID.Anon %in% temp.id)) # No change
dt.y2.cont = dt.y2.cont %>% filter(!(ID.Anon %in% temp.id)) # -4
```

Cleanin up:

```
# Cleaning up.
rm(temp, temp.1, temp.2, temp.3)
```

10 Export Anonymized Datasets

In this section, we will export the anonymized (without SCIPERs) datasets as .csv files.

```
# Export path
path = paste(here(), "/Data/Personal/Filtered/", sep = "")
path.minor = paste(here(),
                   "/Data/Personal/Filtered/Minor/",
                   sep = "")

# Year1
write.csv(dt.y1.flip, paste(path, "Year1-Flipped.csv", sep = ""))
write.csv(dt.y1.cont, paste(path, "Year1-Control.csv", sep = ""))

# Year2
write.csv(dt.y2.flip, paste(path, "Year2-Flipped.csv", sep = ""))
write.csv(dt.y2.cont, paste(path, "Year2-Control.csv", sep = ""))

# With Minor
write.csv(dt.y2.minors, paste(path.minor, "Year2-Minor-List.csv", sep = ""))
write.csv(dt.y2.min.cont, paste(path.minor,
                                "Year2-Control-Minors.csv",
                                sep = ""))
write.csv(dt.y2.min.flip, paste(path.minor,
                                "Year2-Flipped-Minors.csv",
                                sep = ""))

# Aggregated Data -- Without Minors
write.csv(dt.total, paste(path, "Data-Total.csv", sep = ""))
write.csv(dt.flip, paste(path, "Data-Flipped.csv", sep = ""))
write.csv(dt.cont, paste(path, "Data-Control.csv", sep = ""))

# With Minors
write.csv(dt.min.total, paste(path.minor,
                              "Data-Total-Minor.csv",
                              sep = ""))
write.csv(dt.min.cont, paste(path.minor,
                             "Data-Control-Minors.csv",
                             sep = ""))
write.csv(dt.min.flip, paste(path.minor,
                             "Data-Flipped-Minors.csv"))
```

10.1 Clean-Up

In this section, we will clean-up the intermediate data frames which contain non-anonymized data.

```
# Year1
rm(dt.y1.filt, dt.y1)

# Year2
rm(dt.y2.filt, dt.y2)

# Path
rm(path, path.minor)
```

11 Data about Exam Scores

We will aggregate the exam scores from both Year1 and Year2 into the personal datasets, which we prepared in the previous sections. In order to achieve this, we will start with importing the data first, followed by their cleaning, and then aggregation into the personal datasets. Furthermore, the combined dataframe (`dt.total`) contains some individuals who were part of both `Course.Years`, therefore we will integrate them separately for different years.

11.1 Data Import and Cleaning

11.1.1 Importing Data and Basic Cleaning

```
# Specifying the temporary path where these files can be found.
# Please note that this path is temporary as data exists on an external storage.
# Furthermore the filenames have been modified to include the Year information as mentioned above.

# Year1

path = paste(here(), "/Data/Personal/", sep = "")

# Reading rows for different teachers separately.
t.y1.raum = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[1:285,]
t.y1.friedli = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[287:513,]
t.y1.pach = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[515:737,]
t.y1.hess = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[739:943,]
t.y1.deparis = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[945:1040,]
t.y1.scherer = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[1042:1340,]
t.y1.zuleta = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[1342:1535,]
t.y1.shokrollahi = read_excel(paste(path, "FinalExam-Y1.xlsx", sep = ""), sheet = "data")[1537:1821,]

# Year2
t.y2.friedli = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[1:195,]
t.y2.deparis = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[197:368,]
t.y2.scherer = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[370:643,]
t.y2.pach = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[645:834,]
t.y2.hess = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[836:1097,]
t.y2.jecker = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[1099:1406,]
t.y2.semmler = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[1408:1678,]
t.y2.zuleta = read_excel(paste(path, "FinalExam-Y2.xlsx", sep = ""), sheet = "data")[1680:1860,]
```

Before we proceed, we will set the first line of each of the previous data frames as their headers.

```
# Year1
# Setting the first row as the column headers.
colnames(t.y1.raum) = as.character(unlist(t.y1.raum[1,]))
t.y1.raum = t.y1.raum[-1,]
colnames(t.y1.friedli) = as.character(unlist(t.y1.friedli[1,]))
t.y1.friedli = t.y1.friedli[-1,]
colnames(t.y1.pach) = as.character(unlist(t.y1.pach[1,]))
t.y1.pach = t.y1.pach[-1,]
colnames(t.y1.hess) = as.character(unlist(t.y1.hess[1,]))
t.y1.hess = t.y1.hess[-1,]
colnames(t.y1.deparis) = as.character(unlist(t.y1.deparis[1,]))
t.y1.deparis = t.y1.deparis[-1,]
colnames(t.y1.scherer) = as.character(unlist(t.y1.scherer[1,]))
t.y1.scherer = t.y1.scherer[-1,]
colnames(t.y1.zuleta) = as.character(unlist(t.y1.zuleta[1,]))
t.y1.zuleta = t.y1.zuleta[-1,]
colnames(t.y1.shokrollahi) = as.character(unlist(t.y1.shokrollahi[1,]))
t.y1.shokrollahi = t.y1.shokrollahi[-1,]

# Year2
colnames(t.y2.friedli) = as.character(unlist(t.y2.friedli[1,]))
t.y2.friedli = t.y2.friedli[-1,]
```

```
colnames(t.y2.deparis) = as.character(unlist(t.y2.deparis[1,]))
t.y2.deparis = t.y2.deparis[-1,]
colnames(t.y2.scherer) = as.character(unlist(t.y2.scherer[1,]))
t.y2.scherer = t.y2.scherer[-1,]
colnames(t.y2.pach) = as.character(unlist(t.y2.pach[1,]))
t.y2.pach = t.y2.pach[-1,]
colnames(t.y2.hess) = as.character(unlist(t.y2.hess[1,]))
t.y2.hess = t.y2.hess[-1,]
colnames(t.y2.jecker) = as.character(unlist(t.y2.jecker[1,]))
t.y2.jecker = t.y2.jecker[-1,]
colnames(t.y2.semmler) = as.character(unlist(t.y2.semmler[1,]))
t.y2.semmler = t.y2.semmler[-1,]
colnames(t.y2.zuleta) = as.character(unlist(t.y2.zuleta[1,]))
t.y2.zuleta = t.y2.zuleta[-1,]
```

Since, data corresponding to all the teachers has now been imported, we will go ahead and select only the relevant columns. In order to do so, we will make a list of all the columns (Teacher and Personal data + Question Scores from the Common Part).

```
# Selecting the relevant columns.
# Year1
y1.cols = c("TEACHER", "PRESENT",
            "GRADE", "SCIPER",
            "MC-calc-LU", "MC-calc-systeme-lineaire",
            "MC-calc-moindre-carres", "MC-calc-valeurs-propres",
            "MC-calc-vecteurs-propres", "q:MC-calc-base-im",
            "q:MC-calc-base-ker", "q:MC-calc-det",
            "q:MC-calc-inverse", "q:MC-calc-matrice-polynome",
            "q:MC-calc-matrix", "q:MC-calc-orthonormal-basis",
            "q:MC-calc-passage", "q:MC-calc-proj-ortho",
            "q:MC-calc-rank", "q:MC-calc-span",
            "q:MC-theory-determinant", "q:MC-theory-diag",
            "q:MC-theory-diagonalisable", "q:MC-theory-matrice-orthogonale",
            "q:MC-theory-moindres-carres", "q:MC-theory-sous-espaces",
            "q:MC-theory-systeme-lineaire", "q:MC-theory-valeurs-propres")
# Year2
y2.cols = c("TEACHER", "PRESENT",
            "GRADE", "SCIPER",
            "MC-calc-LU", "MC-calc-systeme-lineaire",
            "MC-calc-moindre-carres", "MC-calc-valeurs-propres",
            "MC-calc-vecteurs-propres", "q:MC-calc-base-im-v2",
            "q:MC-calc-base-ker-3-v2", "q:MC-calc-det-v3",
            "q:MC-calc-diag", "q:MC-calc-inverse-v2",
            "q:MC-calc-matrice-polynome", "q:MC-calc-matrix",
            "q:MC-calc-passage", "q:MC-calc-proj-ortho",
            "q:MC-calc-rank", "q:MC-calc-span",
            "q:MC-theory-determinant", "q:MC-theory-diagonalisable",
            "q:MC-theory-diagonalisation", "q:MC-theory-matrice-orthogonale-v0",
            "q:MC-theory-moindres-carres", "q:MC-theory-sous-espaces-v2",
            "q:MC-theory-systeme-lineaire", "q:MC-theory-valeurs-propres-v2")
```

We will now select the columns defined in the above vectors for each professor and for both `Course.Years`.

```
# Select the relevant columns for each professor.
# Year1

# Raum
t.y1.raum = t.y1.raum[,1:67] %>% select(one_of(y1.cols))
# Friedli
t.y1.friedli = t.y1.friedli[,1:65] %>% select(one_of(y1.cols))
# Pach
t.y1.pach = t.y1.pach[,1:73] %>% select(one_of(y1.cols))
# Hess
```

```
t.y1.hess = t.y1.hess[,1:65] %>% select(one_of(y1.cols))
# Deparis
t.y1.deparis = t.y1.deparis[,1:67] %>% select(one_of(y1.cols))
# Scherer
t.y1.scherer = t.y1.scherer[,1:65] %>% select(one_of(y1.cols))
# Zuleta
t.y1.zuleta = t.y1.zuleta[,1:73] %>% select(one_of(y1.cols))
# Shokrollahi
t.y1.shokrollahi = t.y1.shokrollahi[,1:73] %>% select(one_of(y1.cols))

# Year2

# Friedli
t.y2.friedli = t.y2.friedli[,1:32] %>% select(one_of(y2.cols))
# Deparis
t.y2.deparis = t.y2.deparis[,1:32] %>% select(one_of(y2.cols))
# Scherer
t.y2.scherer = t.y2.scherer[,1:32] %>% select(one_of(y2.cols))
# Pach
t.y2.pach = t.y2.pach[,1:32] %>% select(one_of(y2.cols))
# Hess
t.y2.hess = t.y2.hess[,1:32] %>% select(one_of(y2.cols))
# Jecker
t.y2.jecker = t.y2.jecker[,1:32] %>% select(one_of(y2.cols))
# Semmler
t.y2.semmler = t.y2.semmler[,1:32] %>% select(one_of(y2.cols))
# Zuleta
t.y2.zuleta = t.y2.zuleta[,1:32] %>% select(one_of(y2.cols))
```

Next, we will bind these individual data frames (one for each professor) into a single big data frame (still different for the two academic years).

```
# Combine all the data frames.
# Year1
dt.y1.exam = rbind(t.y1.raum, t.y1.friedli,
                   t.y1.pach, t.y1.hess,
                   t.y1.deparis, t.y1.scherer,
                   t.y1.zuleta, t.y1.shokrollahi)
# Remove the individual data frames.
rm(t.y1.raum, t.y1.friedli, t.y1.pach, t.y1.hess,
   t.y1.deparis, t.y1.scherer, t.y1.zuleta, t.y1.shokrollahi)

# Year2
dt.y2.exam = rbind(t.y2.friedli, t.y2.deparis,
                   t.y2.scherer, t.y2.pach,
                   t.y2.hess, t.y2.jecker,
                   t.y2.semmler, t.y2.zuleta)
# Remove the individual data frames.
rm(t.y2.friedli, t.y2.deparis, t.y2.scherer, t.y2.pach,
   t.y2.hess, t.y2.jecker, t.y2.semmler, t.y2.zuleta)
```

Let's rename the columns as the final step for some consistency.

```
# Year1
names(dt.y1.exam) = c("Teacher.Name", "Exam.Present",
                      "Final.Grade", "SCIPER",
                      "MC.Calc.LU", "MC.Calc.Systeme.Lineaire",
                      "MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
                      "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Base.Im",
                      "Q.MC.Calc.Base.Ker", "Q.MC.Calc.Det",
                      "Q.MC.Calc.Inverse", "Q.MC.Calc.Matrice.Polynome",
                      "Q.MC.Calc.Matrix", "Q.MC.Calc.Orthonormal.Basis",
                      "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
```

```
                        "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
                        "Q.MC.Theory.Determinant", "Q.MC.Theory.Diag",
                        "Q.MC.Theory.Diagonalisable", "Q.MC.Theory.Matrice.Orthogonale",
                        "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces",
                        "Q.MC.Theory.Systeme.Lineaire", "Q.MC.Theory.Valeurs.Propres")

# Year2
names(dt.y2.exam) = c("Teacher.Name", "Exam.Present",
                        "Final.Grade", "SCIPER",
                        "MC.Calc.LU", "MC.Calc.Systeme.Lineaire",
                        "MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
                        "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Base.Im.V2",
                        "Q.MC.Calc.Base.Ker.3.V2", "Q.MC.Calc.Det.V3",
                        "Q.MC.Calc.Diag", "Q.MC.Calc.Inverse.V2",
                        "Q.MC.Calc.Matrice.Polynome", "Q.MC.Calc.Matrix",
                        "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
                        "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
                        "Q.MC.Theory.Determinant", "Q.MC.Theory.Diagonalisable",
                        "Q.MC.Theory.Diagonalisation", "Q.MC.Theory.Matrice.Orthogonale.V0",
                        "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces.V2",
                        "Q.MC.Theory.Systeme.Lineaire", "Q.MC.Theory.Valeurs.Propres.V2")
```

Convert all the score columns into numbers as they are currently character arrays.

```
# Convert scores from characters to numbers.
# Year1
y1.cols = c("MC.Calc.LU", "MC.Calc.Systeme.Lineaire",
            "MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
            "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Base.Im",
            "Q.MC.Calc.Base.Ker", "Q.MC.Calc.Det",
            "Q.MC.Calc.Inverse", "Q.MC.Calc.Matrice.Polynome",
            "Q.MC.Calc.Matrix", "Q.MC.Calc.Orthonormal.Basis",
            "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
            "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
            "Q.MC.Theory.Determinant", "Q.MC.Theory.Diag",
            "Q.MC.Theory.Diagonalisable", "Q.MC.Theory.Matrice.Orthogonale",
            "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces",
            "Q.MC.Theory.Systeme.Lineaire", "Q.MC.Theory.Valeurs.Propres")
dt.y1.exam[,y1.cols] = apply(dt.y1.exam[,y1.cols],
                             2, # Coulmn Operation
                             function(x) as.numeric(as.character(x)))

# We remove 3 questions based on their "Discrimination Index"
# which we will look much further in the analysis.
# Threshold = 0.33
y1.cols.di = c("MC.Calc.Systeme.Lineaire",
            "MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
            "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Base.Im",
            "Q.MC.Calc.Base.Ker", "Q.MC.Calc.Det",
            "Q.MC.Calc.Matrice.Polynome",
            "Q.MC.Calc.Matrix", "Q.MC.Calc.Orthonormal.Basis",
            "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
            "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
            "Q.MC.Theory.Determinant", "Q.MC.Theory.Diag",
            "Q.MC.Theory.Matrice.Orthogonale",
            "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces",
            "Q.MC.Theory.Systeme.Lineaire", "Q.MC.Theory.Valeurs.Propres")

# Here are the questions that must be removed.
rem.ques = c("MC.Calc.LU",
             "Q.MC.Calc.Inverse",
             "Q.MC.Theory.Diagonalisable")
dt.y1.exam.di = dt.y1.exam %>% select(-one_of(rem.ques))
```

```
dt.y1.exam.di[,y1.cols.di] = apply(dt.y1.exam.di[,y1.cols.di],
                              2, # Coulmn Operation
                              function(x) as.numeric(as.character(x)))

# Year2
# Converting scores from character to integer.
y2.cols = c("MC.Calc.LU", "MC.Calc.Systeme.Lineaire",
            "MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
            "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Base.Im.V2",
            "Q.MC.Calc.Base.Ker.3.V2", "Q.MC.Calc.Det.V3",
            "Q.MC.Calc.Diag", "Q.MC.Calc.Inverse.V2",
            "Q.MC.Calc.Matrice.Polynome", "Q.MC.Calc.Matrix",
            "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
            "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
            "Q.MC.Theory.Determinant", "Q.MC.Theory.Diagonalisable",
            "Q.MC.Theory.Diagonalisation", "Q.MC.Theory.Matrice.Orthogonale.V0",
            "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces.V2",
            "Q.MC.Theory.Systeme.Lineaire", "Q.MC.Theory.Valeurs.Propres.V2")
dt.y2.exam[,y2.cols] = apply(dt.y2.exam[,y2.cols],
                              2, # Coulmn Operation
                              function(x) as.numeric(as.character(x)))

# We remove 2 questions based on their "Discrimination Index"
# which we will look much further in the analysis.
# Threshold = 0.33
y2.cols.di = c("MC.Calc.LU", "MC.Calc.Systeme.Lineaire",
            "MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
            "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Base.Im.V2",
            "Q.MC.Calc.Base.Ker.3.V2", "Q.MC.Calc.Det.V3",
            "Q.MC.Calc.Diag",
            "Q.MC.Calc.Matrice.Polynome", "Q.MC.Calc.Matrix",
            "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
            "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
            "Q.MC.Theory.Determinant",
            "Q.MC.Theory.Diagonalisation", "Q.MC.Theory.Matrice.Orthogonale.V0",
            "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces.V2",
            "Q.MC.Theory.Systeme.Lineaire", "Q.MC.Theory.Valeurs.Propres.V2")

# Here are the questions that must be removed.
rem.ques = c("Q.MC.Calc.Inverse.V2",
             "Q.MC.Theory.Diagonalisable")
dt.y2.exam.di = dt.y2.exam %>% select(-one_of(rem.ques))
dt.y2.exam.di[,y2.cols.di] = apply(dt.y2.exam.di[,y2.cols.di],
                              2, # Coulmn Operation
                              function(x) as.numeric(as.character(x)))
```

11.2 Filtering Exam Data: Removing Absentees

From the data frames for the two academic years, we will remove all the absentees.

First, we will also explore and see how many of these absentees were actually the volunteers:

```
# Year1
# Combine the flipped and control condition data
t1 = rbind(dt.y1.cont, dt.y1.flip)
# Compute SCIPER from ID.Anon
t1$SCIPER = lapply(t1$ID.Anon, GenerateSCIPER)
t1$SCIPER = unlist(t1$SCIPER)
# Exam scores data
t2 = dt.y1.exam

# Join operation
t3 = merge(t1,
```

```
            t2,
            by = "SCIPER")

# Summarise the number of absentees based on Condition (only Volunteers)
t3 %>% filter(Exam.Present == 0) %>%
  select(SCIPER, Condition) %>%
  group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control      22
## 2 Flipped       3

# Year2
# Combine the flipped and control condition data
t1 = rbind(dt.y2.cont, dt.y2.flip)
# Compute SCIPER from ID.Anon
t1$SCIPER = lapply(t1$ID.Anon, GenerateSCIPER)
t1$SCIPER = unlist(t1$SCIPER)
# Exam scores data
t2 = dt.y2.exam

# Join operation
t3 = merge(t1,
            t2,
            by = "SCIPER")

# Summarise the number of absentees based on Condition (only Volunteers)
t3 %>% filter(Exam.Present == 0) %>%
  select(SCIPER, Condition) %>%
  group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control       6
## 2 Flipped      12

# Cleaning up.
rm(t1, t2, t3)
```

In YEAR1 there were 113 absentees. And in YEAR2 there were 109 absentees. This includes both the volunteers and the non-volunteers.

```
# Remove all the Absentees.
# Year1
dt.y1.exam = dt.y1.exam %>% filter(Exam.Present == 1)

# Year1 -- With DI Metrics
dt.y1.exam.di = dt.y1.exam.di %>% filter(Exam.Present == 1)

# Year2
dt.y2.exam = dt.y2.exam %>% filter(Exam.Present == 1)

# Year2 -- With DI Metrics
dt.y2.exam.di = dt.y2.exam.di %>% filter(Exam.Present == 1)
```

11.3 New Variables and Anonymization

In this section, we will add new new variables corresponding to the total score, and scores for different course segments (A, B, and C). In addition, we will anonymize the dataframe, and remove the SCIPER values.

11.3.1 Total Score

```
# Compute Total Score
# Year1
dt.y1.exam$Total.Score = rowSums(dt.y1.exam[,y1.cols])

# Year1 -- With DI Metrics
dt.y1.exam.di$Total.Score = rowSums(dt.y1.exam.di[,y1.cols.di])

# Year2
dt.y2.exam$Total.Score = rowSums(dt.y2.exam[,y2.cols])

# Year2 -- With DI Metrics
dt.y2.exam.di$Total.Score = rowSums(dt.y2.exam.di[,y2.cols.di])
```

11.3.2 Grouping Questions into Blocks

In the exam, the questions corresponded to the different lecture weeks. These lecture blocks belonged to three blocks : **A, B, C**. In YEAR1, the block **B** corresponded to the flipped part. However, in YEAR2, blocks **B** and **C** belonged to the flipped part.

First, we do it for YEAR1.

```
# Assigning questions to blocks.
# Year1
# Block A
y1.cols.A = c("MC.Calc.Systeme.Lineaire", "Q.MC.Calc.Base.Im",
              "Q.MC.Calc.Base.Ker", "Q.MC.Calc.Span",
              "Q.MC.Theory.Sous.Espaces", "Q.MC.Theory.Systeme.Lineaire")
# Block B
y1.cols.B = c("MC.Calc.LU", "Q.MC.Calc.Det",
              "Q.MC.Calc.Inverse", "Q.MC.Calc.Matrice.Polynome",
              "Q.MC.Calc.Matrix", "Q.MC.Calc.Passage",
              "Q.MC.Calc.Rank", "Q.MC.Theory.Matrice.Orthogonale")
# Block B -- With DI Metrics
y1.cols.B.di = c("Q.MC.Calc.Det",
                 "Q.MC.Calc.Matrice.Polynome",
                 "Q.MC.Calc.Matrix", "Q.MC.Calc.Passage",
                 "Q.MC.Calc.Rank", "Q.MC.Theory.Matrice.Orthogonale")
# Block C
y1.cols.C = c("MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
              "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Orthonormal.Basis",
              "Q.MC.Calc.Proj.Ortho", "Q.MC.Theory.Determinant",
              "Q.MC.Theory.Diag", "Q.MC.Theory.Diagonalisable",
              "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Valeurs.Propres")
# Block C -- With DI Metrics
y1.cols.C.di = c("MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
                 "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Orthonormal.Basis",
                 "Q.MC.Calc.Proj.Ortho", "Q.MC.Theory.Determinant",
                 "Q.MC.Theory.Diag",
                 "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Valeurs.Propres")
```

Next, for YEAR2.

```
# Assigning questions to blocks.
# Year2
# Block A
y2.cols.A = c("MC.Calc.Systeme.Lineaire", "Q.MC.Calc.Span",
              "Q.MC.Theory.Sous.Espaces.V2", "Q.MC.Theory.Systeme.Lineaire")
# Block B
y2.cols.B = c("MC.Calc.LU", "Q.MC.Calc.Base.Im.V2",
              "Q.MC.Calc.Base.Ker.3.V2", "Q.MC.Calc.Det.V3",
              "Q.MC.Calc.Inverse.V2", "Q.MC.Calc.Matrice.Polynome",
```

```
              "Q.MC.Calc.Matrix", "Q.MC.Calc.Passage",
              "Q.MC.Calc.Rank")
# Block B -- With DI Index
y2.cols.B.di = c("MC.Calc.LU", "Q.MC.Calc.Base.Im.V2",
                 "Q.MC.Calc.Base.Ker.3.V2", "Q.MC.Calc.Det.V3",
                 "Q.MC.Calc.Matrice.Polynome",
                 "Q.MC.Calc.Matrix", "Q.MC.Calc.Passage",
                 "Q.MC.Calc.Rank")
# Block C
y2.cols.C = c("MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
              "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Diag",
              "Q.MC.Calc.Proj.Ortho", "Q.MC.Theory.Determinant",
              "Q.MC.Theory.Diagonalisable", "Q.MC.Theory.Diagonalisation",
              "Q.MC.Theory.Matrice.Orthogonale.V0", "Q.MC.Theory.Moindres.Carres",
              "Q.MC.Theory.Valeurs.Propres.V2")

# Block C -- With DI Index
y2.cols.C.di = c("MC.Calc.Moindre.Carres", "MC.Calc.Valeurs.Propres",
                 "MC.Calc.Vecteurs.Propres", "Q.MC.Calc.Diag",
                 "Q.MC.Calc.Proj.Ortho", "Q.MC.Theory.Determinant",
                 "Q.MC.Theory.Diagonalisation",
                 "Q.MC.Theory.Matrice.Orthogonale.V0", "Q.MC.Theory.Moindres.Carres",
                 "Q.MC.Theory.Valeurs.Propres.V2")
```

11.3.3 Score: Block A

```
# Compute Score for Block A
# Year1
dt.y1.exam$Score.A = rowSums(dt.y1.exam[,y1.cols.A])

# Year1 -- With DI Metrics
dt.y1.exam.di$Score.A = rowSums(dt.y1.exam.di[,y1.cols.A])

# Year2
dt.y2.exam$Score.A = rowSums(dt.y2.exam[,y2.cols.A])

# Year2 -- With DI Index
dt.y2.exam.di$Score.A = rowSums(dt.y2.exam.di[,y2.cols.A])
```

11.3.4 Score: Block B

```
# Compute Score for Block B
# Year1
dt.y1.exam$Score.B = rowSums(dt.y1.exam[,y1.cols.B])

# Year1 -- With DI Metrics
dt.y1.exam.di$Score.B = rowSums(dt.y1.exam.di[,y1.cols.B.di])

# Year2
dt.y2.exam$Score.B = rowSums(dt.y2.exam[,y2.cols.B])

# Year2 -- With DI Index
dt.y2.exam.di$Score.B = rowSums(dt.y2.exam.di[,y2.cols.B.di])
```

11.3.5 Score: Block C

```
# Compute Score for Block C
# Year1
dt.y1.exam$Score.C = rowSums(dt.y1.exam[,y1.cols.C])

# Year1 -- With DI Metrics
dt.y1.exam.di$Score.C = rowSums(dt.y1.exam.di[,y1.cols.C.di])

# Year2
dt.y2.exam$Score.C = rowSums(dt.y2.exam[,y2.cols.C])

# Year2 -- With DI Index
dt.y2.exam.di$Score.C = rowSums(dt.y2.exam.di[,y2.cols.C.di])
```

11.3.6 Score: Block BC (Post-Flipping)

```
# Compute Score for Block BC
# Year1
dt.y1.exam$Score.BC = dt.y1.exam$Score.B + dt.y1.exam$Score.C

# Year1 -- With DI Metrics
dt.y1.exam.di$Score.BC = dt.y1.exam.di$Score.B + dt.y1.exam.di$Score.C

# Year2
dt.y2.exam$Score.BC = dt.y2.exam$Score.B + dt.y2.exam$Score.C

# Year2 -- With DI Index
dt.y2.exam.di$Score.BC = dt.y2.exam.di$Score.B + dt.y2.exam.di$Score.C
```

11.3.7 Anonymizing SCIPERs

```
# Anonymizing the SCIPER followed by their removal.
# Year1
dt.y1.exam$ID.Anon = lapply(dt.y1.exam$SCIPER, GenerateUniqueID)
dt.y1.exam$SCIPER = NULL
dt.y1.exam$ID.Anon = unlist(dt.y1.exam$ID.Anon)

# Year1 -- With DI Metrics
dt.y1.exam.di$ID.Anon = lapply(dt.y1.exam.di$SCIPER, GenerateUniqueID)
dt.y1.exam.di$SCIPER = NULL
dt.y1.exam.di$ID.Anon = unlist(dt.y1.exam.di$ID.Anon)

# Year2
dt.y2.exam$ID.Anon = lapply(dt.y2.exam$SCIPER, GenerateUniqueID)
dt.y2.exam$SCIPER = NULL
dt.y2.exam$ID.Anon = unlist(dt.y2.exam$ID.Anon)

# Year2 -- With DI Index
dt.y2.exam.di$ID.Anon = lapply(dt.y2.exam.di$SCIPER, GenerateUniqueID)
dt.y2.exam.di$SCIPER = NULL
dt.y2.exam.di$ID.Anon = unlist(dt.y2.exam.di$ID.Anon)
```

11.4 Filtering Data: Removing Repeaters

People who were present in both YEAR1 and YEAR2, and especially the ones who were selected in FLIPPED condition in YEAR1 have to be removed from the exam dataset as well. If we do not do so, then with an outer join operation, these people can become part of the NON-VOLUNTEERS, so we will filter them right at the beginning.

As we did for volunteers in previous section, we will have to remove the 5 subjects from the exam dataset.

The `ID.Anon` of these people are as follows:

```
# Volunteers who repeated in both years with "Flipped" condition in first year.
temp.id
```

```
## [1] "2504T1445" "2961T1719" "3565T2085" "4324T2538" "966T522"
```

Removing them from exam dataset:

```
# Filtering some repeaters.
# Year1
dt.y1.exam = dt.y1.exam %>% filter(!(ID.Anon %in% temp.id))

# Year1 -- With DI Index
dt.y1.exam.di = dt.y1.exam.di %>% filter(!(ID.Anon %in% temp.id))

# Year2
dt.y2.exam = dt.y2.exam %>% filter(!(ID.Anon %in% temp.id))

# Year2 -- With DI Index
dt.y2.exam.di = dt.y2.exam.di %>% filter(!(ID.Anon %in% temp.id))
```

11.5 Data Export and Cleanup

```
# Exporting data as .csv files.
path = paste(here(), "/Data/Personal/Filtered/", sep = "")
#path = "~/Desktop/Work/EPFL-Learn/Analyses/Data/Personal/Filtered/"

# Year1
write.csv(dt.y1.exam, paste(path, "Year1-Exam.csv", sep = ""))

# Year1 -- With DI Metrics
write.csv(dt.y1.exam.di, paste(path, "Year1-Exam-DI-Filtered.csv", sep = ""))

# Year2
write.csv(dt.y2.exam, paste(path, "Year2-Exam.csv", sep = ""))

# Year2 -- With DI Index
write.csv(dt.y2.exam.di, paste(path, "Year2-Exam-DI-Filtered.csv", sep = ""))
```

```
# Some cleanup operations.
rm(y1.cols, y1.cols.A, y1.cols.B, y1.cols.C)
rm(y1.cols.di, y1.cols.B.di, y1.cols.C.di)
rm(y2.cols, y2.cols.A, y2.cols.B, y2.cols.C)
rm(y2.cols.di, y2.cols.B.di, y2.cols.C.di, rem.ques)
rm(path)
```

12 Joining Personal and Exam Data

We will perform the join operation for each year separately. But before we start doing that, we have to create data frames for both Year1 and Year2.

```
# Year1
dt.y1.personal = rbind(dt.y1.cont, dt.y1.flip)
dt.y1.personal$ID.Anon = unlist(dt.y1.personal$ID.Anon)

# Year2
dt.y2.personal = rbind(dt.y2.cont, dt.y2.flip)
dt.y2.personal$ID.Anon = unlist(dt.y2.personal$ID.Anon)

# Year2 -- With Minors
```

```
dt.y2.min.personal = rbind(dt.y2.min.cont, dt.y2.min.flip)
dt.y2.min.personal$ID.Anon = unlist(dt.y2.min.personal$ID.Anon)
```

12.1 Year 1 (2017-18)

12.1.1 Non-Volunteers

First, we will do an Outer (Right) join where all the rows from the `dt.y1.exam` are present. This set of data will provide us with all the people who are in the Non-Volunteers group.

```
# Perform an outer (right) join.
y1.non.vol = merge(x = dt.y1.personal,
                   y = dt.y1.exam,
                   by = "ID.Anon",
                   all.y = TRUE)

# With DI Index
y1.non.vol.di = merge(x = dt.y1.personal,
                      y = dt.y1.exam.di,
                      by = "ID.Anon",
                      all.y = TRUE)

# All rows with NULL values for Gender, Course.Year, Condition, etc. are
# non-volunteers.
y1.non.vol = y1.non.vol %>% filter(is.na(Condition))

# With DI Index
y1.non.vol.di = y1.non.vol.di %>% filter(is.na(Condition))
```

From the Year1 there are 1205 **non-volunteers**.

Checking if there are some students in the Non-Volunteers who might have taken the exam with Deparis (in the Flipped Condition). This means that these set of students did not register initially but took the exam with Deparis.

```
# Any non-volunteers taking exam with Deparis
y1.non.vol %>% group_by(Teacher.Name) %>%
  summarise(Count = n())
```

```
## # A tibble: 7 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Friedli        157
## 2 Hess           135
## 3 Pach           168
## 4 Raum           194
## 5 Scherer        219
## 6 Shokrollahi    198
## 7 Zuleta         134
```

```
# With DI Index
y1.non.vol.di %>% group_by(Teacher.Name) %>%
  summarise(Count = n())
```

```
## # A tibble: 7 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Friedli        157
## 2 Hess           135
## 3 Pach           168
## 4 Raum           194
## 5 Scherer        219
## 6 Shokrollahi    198
## 7 Zuleta         134
```

We see that there are **NO** students in Non-Volunteers who took an exam with Deparis.
NOTE: Please note that this data cannot be exported, and has to be deleted upon the completion of analysis.

12.1.2 Volunteers

In this section, we will generate the dataset of VOLUNTEERS. These are the students who volunteered to participate in the *flipped classroom* study. They were either chosen (FLIPPED CONDITION) or they were not chosen (CONTROL CONDITION). The means of disciminating FLIPPED from CONTROL is to check if the students took the exam with Deparis.

```
# Perform the Inner Join
y1.vol = merge(x = dt.y1.personal,
               y = dt.y1.exam,
               by = "ID.Anon")

# With DI Metrics
y1.vol.di = merge(x = dt.y1.personal,
                  y = dt.y1.exam.di,
                  by = "ID.Anon")
```

We see that there are 483 subjects, which are less than the initial numbers (513) who registered. This could mean that they were absent in the final exam.

```
# Examining the volunteers who did not take exam.
y1.vol.missing = merge(x = dt.y1.personal,
                       y = dt.y1.exam,
                       by = "ID.Anon",
                       all.x = TRUE)

# With DI Metrics
y1.vol.missing.di = merge(x = dt.y1.personal,
                          y = dt.y1.exam.di,
                          by = "ID.Anon",
                          all.x = TRUE)

# Filtering the rows with NULL for Teacher.Name
y1.vol.missing = y1.vol.missing %>%
  filter(is.na(Teacher.Name))
# Giving summary based on condition.
y1.vol.missing %>% group_by(Condition, Teacher.Name) %>%
  summarise(Count = n())
```

```
## # A tibble: 2 x 3
## # Groups:   Condition [2]
##   Condition Teacher.Name Count
##   <chr>     <chr>        <int>
## 1 Control   <NA>            27
## 2 Flipped   <NA>             3
```

```
# With DI Metrics
y1.vol.missing.di = y1.vol.missing.di %>%
  filter(is.na(Teacher.Name))

# Let's see how many absentees were in Control/Flipped conditions.
y1.vol.missing %>% select(ID.Anon, Condition) %>%
  spread(ID.Anon, Condition)
```

```
##   1209T668 1481T832 1533T862 1611T909 1621T917 1653T934 1718T973 2108T1207 2134T1223
## 1  Control  Control  Control  Control  Control  Control  Control   Control   Control
##   2269T1305 2386T1374 2429T1400 2598T1501 2901T1684 2949T1712 2989T1737 3024T1757
## 1   Control   Control   Control   Flipped   Control   Control   Control   Control
##   3038T1766 3625T2120 3927T2299 3956T2317 4202T2464 4300T2525 4539T2666 4596T2700
## 1   Control   Control   Flipped   Control   Control   Control   Control   Control
##   4662T2740 4898T2884 4914T2892 5004T2945 994T539
## 1   Control   Control   Control   Control Flipped
```

```r
# With DI Metrics
y1.vol.missing.di %>% select(ID.Anon, Condition) %>%
  spread(ID.Anon, Condition)

##   1209T668 1481T832 1533T862 1611T909 1621T917 1653T934 1718T973 2108T1207 2134T1223
## 1  Control  Control  Control  Control  Control  Control  Control   Control   Control
##   2269T1305 2386T1374 2429T1400 2598T1501 2901T1684 2949T1712 2989T1737 3024T1757
## 1   Control   Control   Control   Flipped   Control   Control   Control   Control
##   3038T1766 3625T2120 3927T2299 3956T2317 4202T2464 4300T2525 4539T2666 4596T2700
## 1   Control   Control   Flipped   Control   Control   Control   Control   Control
##   4662T2740 4898T2884 4914T2892 5004T2945 994T539
## 1   Control   Control   Control   Control Flipped
```

We see that there are 3 students who were absent in exam from the FLIPPED condition, and 27 learners absent in exams from the CONTROL condition.

```r
# Summarizing the missing individuals.
y1.vol.missing %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control      27
## 2 Flipped       3

# With DI Metrics
y1.vol.missing.di %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control      27
## 2 Flipped       3
```

12.1.3 Flipped Condition Subjects with Other Teachers

It is also important to examine the students, who were initially assigned to the FLIPPED CONDITION, but in the end took exam with a different teacher.

```r
# Examining Flipped-Volunteers with Other Teachers
y1.defect = y1.vol %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")

# With DI Metrics
y1.defect.di = y1.vol.di %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")
```

There are in total 12 students who were initially assigned to the FLIPPED condition but took exam with other teachers besides DEPARIS. These people must be removed from the future analysis.

```r
# Filtering out the people from list: y1.defect
y1.vol.correct = y1.vol %>%
  filter(!(ID.Anon %in% y1.defect$ID.Anon))

# With DI Metrics
y1.vol.correct.di = y1.vol.di %>%
  filter(!(ID.Anon %in% y1.defect.di$ID.Anon))
```

Also, there are now in total **471** volunteers who can be subjected to further analysis. Of these volunteers, following is the distribution in the different conditions:

```
y1.vol.correct %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control     382
## 2 Flipped      89
```

Just for the sake of completion, we will also check the people who were assigned to CONTROL condition, but took an exam with DEPARIS.

```
# Examining Control-Volunteers with Deparis
y1.cont.defect = y1.vol %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name == "Deparis" & Condition == "Control")
```

There are 0 students from the CONTROL condition who took an exam with DEPARIS.

```
# Cleaning up
rm(y1.cont.defect)
```

12.1.4 Exporting Volunteer Data

```
# Exporting volunteer data.
path = paste(here(), "/Data/Personal/Filtered/", sep = "")

write.csv(y1.vol.correct, paste(path, "Year1-Total.csv", sep = ""))

# With DI Metrics
write.csv(y1.vol.correct.di, paste(path, "Year1-Total-DI-Filtered.csv", sep = ""))
```

12.2 Year 2 (2018-19)

12.2.1 Non-Volunteers

First, we will do an Outer (Right) join where all the rows from the `dt.y2.exam` are present. This set of data will provide us with all the people who are in the NON-VOLUNTEERS group.

```
# Perform an outer (right) join.
y2.non.vol = merge(x = dt.y2.personal,
                   y = dt.y2.exam,
                   by = "ID.Anon",
                   all.y = TRUE)

# With the DI Index
y2.non.vol.di = merge(x = dt.y2.personal,
                      y = dt.y2.exam.di,
                      by = "ID.Anon",
                      all.y = TRUE)

# With the DI Index and Minors
y2.non.vol.di.min = merge(x = dt.y2.min.personal,
                          y = dt.y2.exam.di,
                          by = "ID.Anon",
                          all.y = TRUE)

# All rows with NULL values for Gender, Course.Year, Condition, etc. are
# non-volunteers.
y2.non.vol = y2.non.vol %>% filter(is.na(Condition))

# With the DI Index.
```

```
y2.non.vol.di = y2.non.vol.di %>% filter(is.na(Condition))

# With the DI Index and Minors.
y2.non.vol.di.min = y2.non.vol.di.min %>% filter(is.na(Condition))
```

From the YEAR2 there are 1453 **non-volunteers**.

From the YEAR2, including the minors, there are 1386 **non-volunteers**. The reason that this number is smaller than the total number of non-volunteers (above) is that: the size of the volunteer set becomes larger after the minors are included, which reduces the size of this (non-volunteer) dataset. Previously, since we removed the minors, pre-emptively, after the outer join operation, they are counted in the non-volunteer dataset.

Checking if there are some students in the NON-VOLUNTEERS who might have taken the exam with Deparis. This means that these set of students did not register initially but took the exam with Deparis.

```
# Any non-volunteers taking exam with Deparis
y2.non.vol %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Deparis         27
## 2 Friedli        159
## 3 Hess           220
## 4 Jecker         269
## 5 Pach           164
## 6 Scherer        227
## 7 Semmler        229
## 8 Zuleta         158

# Also with Minors
y2.non.vol.di.min %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## # A tibble: 7 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Friedli        151
## 2 Hess           214
## 3 Jecker         260
## 4 Pach           162
## 5 Scherer        219
## 6 Semmler        227
## 7 Zuleta         153
```

We see that there are **27** students in Non-Volunteers who took an exam with Deparis. But then if we also include minors, we see that these 27 students are no longer there. Which signifies that these students are basically minors who were counted as non-volunteers due to the outer join operation.

Also, including minors in the above list, we see that there are **0** students in the Non-Volunteers (including minors) who took an exam with Deparis.

Let's look at these students separately:

```
# Subsetting non-volunteers taking exams with Deparis.
y2.non.vol.deparis = y2.non.vol %>% filter(Teacher.Name == "Deparis")

# With the DI Index
y2.non.vol.deparis.di = y2.non.vol.di %>% filter(Teacher.Name == "Deparis")
```

Remove these students: Since these **27** students took an exam with DEPARIS and we do not have any information regarding their background (because they did not register as volunteers), they should be removed from the `y2.non.vol` dataset.

```
# Removing non-volunteers who took exam with Deparis
y2.non.vol = y2.non.vol %>%
  filter(!(ID.Anon %in% y2.non.vol.deparis$ID.Anon))
```

```
# Cleaning up a bit.
rm(y2.non.vol.deparis)

# With the DI Index
y2.non.vol.di = y2.non.vol.di %>%
  filter(!(ID.Anon %in% y2.non.vol.deparis.di$ID.Anon))

# Cleaning up a bit.
rm(y2.non.vol.deparis.di)
```

Just to check (again) if there still remain any students (non-volunteers) who took exam with DEPARIS:

```
# Summarizing the number of non-volunteers for each teacher.
y2.non.vol %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## # A tibble: 7 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Friedli        159
## 2 Hess           220
## 3 Jecker         269
## 4 Pach           164
## 5 Scherer        227
## 6 Semmler        229
## 7 Zuleta         158

# With the DI Index.
y2.non.vol.di %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## # A tibble: 7 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Friedli        159
## 2 Hess           220
## 3 Jecker         269
## 4 Pach           164
## 5 Scherer        227
## 6 Semmler        229
## 7 Zuleta         158
```

NOTE: Please note that this data cannot be exported, and has to be deleted upon the completion of analysis.

12.2.2 Volunteers

In this section, we will generate the dataset of VOLUNTEERS. These are the students who volunteered to participate in the *flipped classroom* study. They were either chosen (FLIPPED CONDITION) or they were not chosen (CONTROL CONDITION). The means of disciminating FLIPPED from CONTROL is to check if the students took the exam with Deparis.

```
# Perform the Inner Join
y2.vol = merge(x = dt.y2.personal,
               y = dt.y2.exam,
               by = "ID.Anon")

# With the DI Index
y2.vol.di = merge(x = dt.y2.personal,
                  y = dt.y2.exam.di,
                  by = "ID.Anon")

# With the DI Index and Minors
y2.vol.di.min = merge(x = dt.y2.min.personal,
```

```
                      y = dt.y2.exam.di,
                      by = "ID.Anon")
```

We see that there are 278 subjects, which are less than the initial numbers (299) who registered. This could mean that they were absent in the final exam.

```
# Examining the volunteers who did not take exam.
y2.vol.missing = merge(x = dt.y2.personal,
                       y = dt.y2.exam,
                       by = "ID.Anon",
                       all.x = TRUE)

# Filtering the rows with NULL for Teacher.Name
y2.vol.missing = y2.vol.missing %>%
  filter(is.na(Teacher.Name))

# Let's see how many of them were in Control/Flipped conditions.
y2.vol.missing %>% select(ID.Anon, Condition) %>%
  spread(ID.Anon, Condition)

##   1489T833 1664T939 1837T1042 1984T1131 2122T1213 2237T1282 2347T1348 2407T1384 2556T1476
## 1  Control  Flipped   Flipped   Flipped   Control   Flipped   Flipped   Control   Control
##   2682T1549 2712T1567 2857T1654 3128T1817 3157T1834 3874T2265 4487T2632 4488T2633
## 1   Flipped   Control   Control   Flipped   Control   Flipped   Flipped   Flipped
##   4730T2779 4988T2938 5522T3253 939T504
## 1   Flipped   Flipped   Flipped Control

# With DI Index
y2.vol.missing.di = merge(dt.y2.personal,
                          dt.y2.exam.di,
                          by = "ID.Anon",
                          all.x = TRUE)

# Filtering the rows with NULL for Teacher.Name
y2.vol.missing.di = y2.vol.missing.di %>%
  filter(is.na(Teacher.Name))

# Let's see how many of them were in Control/Flipped conditions.
y2.vol.missing.di %>% select(ID.Anon, Condition) %>%
  spread(ID.Anon, Condition)

##   1489T833 1664T939 1837T1042 1984T1131 2122T1213 2237T1282 2347T1348 2407T1384 2556T1476
## 1  Control  Flipped   Flipped   Flipped   Control   Flipped   Flipped   Control   Control
##   2682T1549 2712T1567 2857T1654 3128T1817 3157T1834 3874T2265 4487T2632 4488T2633
## 1   Flipped   Control   Control   Flipped   Control   Flipped   Flipped   Flipped
##   4730T2779 4988T2938 5522T3253 939T504
## 1   Flipped   Flipped   Flipped Control

# With DI Index and minors
y2.vol.missing.di.min = merge(dt.y2.min.personal,
                          dt.y2.exam.di,
                          by = "ID.Anon",
                          all.x = TRUE)

# Filtering the rows with NULL for Teacher.Name
y2.vol.missing.di.min = y2.vol.missing.di.min %>%
  filter(is.na(Teacher.Name))

# Let's see how many of them were in Control/Flipped conditions.
y2.vol.missing.di.min %>% select(ID.Anon, Condition) %>%
  spread(ID.Anon, Condition)

##   1489T833 1664T939 1837T1042 1984T1131 2122T1213 2237T1282 2347T1348 2407T1384 2504T1445
## 1  Control  Flipped   Flipped   Flipped   Control   Flipped   Flipped   Control   Control
```

```
##   2556T1476 2682T1549 2712T1567 2857T1654 2961T1719 3128T1817 3157T1834 3565T2085
## 1   Control   Flipped   Control   Control   Flipped   Flipped   Control   Control
##   3774T2204 3874T2265 4324T2538 4487T2632 4488T2633 4730T2779 4988T2938 5522T3253 939T504
## 1   Control   Flipped   Control   Flipped   Flipped   Flipped   Flipped   Flipped Control
##   966T522
## 1 Control
```

We see that there are 13 learners who were absent in exam from the FLIPPED condition, and 8 learners absent in exams from the CONTROL condition. Also, if we include the minors, there are in total 27 absentees (14 Flipped, 13 Control).

```
# Summarizing the missing individuals.
y2.vol.missing %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control       8
## 2 Flipped      13

# With the DI Index
y2.vol.missing.di %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control       8
## 2 Flipped      13

# With the DI Index and minors
y2.vol.missing.di.min %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control      13
## 2 Flipped      14
```

12.2.3 Flipped Condition Subjects with Other Teachers

It is also important to examine the students, who were initially assigned to the FLIPPED CONDITION, but in the end took exam with a different teacher.

```
# Examining Flipped-Volunteers with Other Teachers
y2.defect = y2.vol %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")

# With DI Index
y2.defect.di = y2.vol.di %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")

# With DI Index and Minors
y2.defect.di.min = y2.vol.di.min %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")
```

There are in total 18 students who were initially assigned to the FLIPPED condition but took exam with other teachers besides DEPARIS. These people must be removed from the future analysis.

```
# Filtering out the people from list: y2.defect
y2.vol.correct = y2.vol %>%
```

```
  filter(!(ID.Anon %in% y2.defect$ID.Anon))

# With DI Index
y2.vol.correct.di = y2.vol.di %>%
  filter(!(ID.Anon %in% y2.defect.di$ID.Anon))

# With DI Index and Minors
y2.vol.correct.di.min = y2.vol.di.min %>%
  filter(!(ID.Anon %in% y2.defect.di.min$ID.Anon))
```

Also, there are now in total **260** volunteers who can be subjected to further analysis. Of these volunteers, following is the distribution in the different conditions:

```
# Summarising
y2.vol.correct %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control     126
## 2 Flipped     134

# With DI Index
y2.vol.correct.di %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control     126
## 2 Flipped     134

# With DI Index and Minors
y2.vol.correct.di.min %>% group_by(Condition) %>%
  summarise(Count = n())

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control     158
## 2 Flipped     161
```

Just for the sake of completion, we will also check the people who were assigned to CONTROL condition, but took an exam with DEPARIS.

```
# Examining Control-Volunteers with Deparis
y2.cont.defect = y2.vol %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name == "Deparis" & Condition == "Control")
```

There are 0 students from the CONTROL condition who took an exam with DEPARIS.

```
# Cleaning up
rm(y2.cont.defect)
```

12.2.4 Exporting Volunteer Data

```
# Exporting volunteer data.
path = paste(here(), "/Data/Personal/Filtered/", sep = "")
path.minor = paste(here(),
                   "/Data/Personal/Filtered/Minor/",
                   sep = "")
```

```
write.csv(y2.vol.correct, paste(path, "Year2-Total.csv", sep = ""))

# With DI Index
write.csv(y2.vol.correct.di, paste(path, "Year2-Total-DI-Filtered.csv", sep = ""))

# With DI Index and Minors
write.csv(y2.vol.correct.di.min, paste(path.minor, "Year2-Total-DI-Filtered-Minor.csv", sep = ""))
```

12.3 Combining and Exporting Volunteer Data

```
# List of selected columns.
col.names = c("ID.Anon", "Gender", "Section",
              "Code.BA", "Diploma.Obtained", "Category",
              "Country.Diploma", "Reached.Adulthood", "Course.Year",
              "Condition", "Teacher.Name", "Final.Grade",
              "Total.Score", "Score.A", "Score.B",
              "Score.C")
# Combining data for Year1 and Year2
dt.vol = rbind(y1.vol.correct %>% select(one_of(col.names)),
               y2.vol.correct %>% select(one_of(col.names)))

# With DI Index
dt.vol.di = rbind(y1.vol.correct.di %>% select(one_of(col.names)),
                  y2.vol.correct.di %>% select(one_of(col.names)))

# With DI Index and Minors
dt.vol.di.min = rbind(y1.vol.correct.di %>% select(one_of(col.names)),
                  y2.vol.correct.di.min %>% select(one_of(col.names)))

# Cleaning up.
rm(col.names)
```

Exporting the data.

```
path = paste(here(), "/Data/Personal/Filtered/", sep = "")
path.minor = paste(here(),
                   "/Data/Personal/Filtered/Minor/",
                   sep = "")

write.csv(dt.vol, paste(path, "Volunteer-Data-Complete.csv", sep = ""))

# With DI Index.
write.csv(dt.vol.di, paste(path, "Volunteer-Data-Complete-DI-Filtered.csv",
                           sep = ""))

# With DI Index and Minors.
write.csv(dt.vol.di.min, paste(path.minor,
                               "Volunteer-Data-Complete-DI-Filtered-Minor.csv",
                               sep = ""))
```

13 Export Non-Volunteer Data

```
# Exporting non-volunteer data.
path = paste(here(), "/Data/Non-Volunteers/", sep = "")

write.csv(y1.non.vol, paste(path, "Year1-Non-Vol.csv", sep = ""))
write.csv(y2.non.vol, paste(path, "Year2-Non-Vol.csv", sep = ""))
```

```
# With DI Index
write.csv(y1.non.vol.di, paste(path, "Year1-Non-Vol-DI-Filtered.csv",
                               sep = ""))
write.csv(y2.non.vol.di, paste(path, "Year2-Non-Vol-DI-Filtered.csv",
                               sep = ""))
```

13.0.1 Cleaning Up

Before we proceed with the next steps, including the normalization of scores, we will clean up the workspace:

```
# Cleaning up
rm(list = ls())
```

14 Normalization of the Scores

In this section, we will *normalize* our score variables such as `Total.Score, Score.A, Score.B, Score.C, Score.BC`. But, before we proceed we will re-import the cleaned data.

14.1 Re-importing data

Volunteer data, first:

```
# Reading from the .csv files
path = paste(here(), "/Data/Personal/Filtered/", sep = "")

# Year1
dt.y1 = read.csv(paste(path, "Year1-Total.csv", sep = ""), header = TRUE)
dt.y1$X = NULL
# Convert ID.Anon to character.
dt.y1$ID.Anon = as.character(dt.y1$ID.Anon)

# Year1 -- With DI Index
dt.y1.di = read.csv(paste(path,
                          "Year1-Total-DI-Filtered.csv",
                          sep = ""),
                    header = TRUE)
dt.y1.di$X = NULL
# Convert ID.Anon to character.
dt.y1.di$ID.Anon = as.character(dt.y1.di$ID.Anon)

# Year2
dt.y2 = read.csv(paste(path, "Year2-Total.csv", sep = ""), header = TRUE)
dt.y2$X = NULL
# Convert ID.Anon to character.
dt.y2$ID.Anon = as.character(dt.y2$ID.Anon)

# Year2 -- With DI Index
dt.y2.di = read.csv(paste(path,
                          "Year2-Total-DI-Filtered.csv",
                          sep = ""),
                    header = TRUE)
dt.y2.di$X = NULL
# Convert ID.Anon to character.
dt.y2.di$ID.Anon = as.character(dt.y2.di$ID.Anon)
```

Non-volunteer data, second:

```
# Reading from the .csv files.
path = paste(here(), "/Data/Non-Volunteers/", sep = "")

# Year1
```

```
t.y1.nv = read.csv(paste(path, "Year1-Non-Vol.csv", sep = ""), header = TRUE)
t.y1.nv$X = NULL

# Year1 - With DI Index
t.y1.nv.di = read.csv(paste(path,
                            "Year1-Non-Vol-DI-Filtered.csv",
                            sep = ""),
                      header = TRUE)
t.y1.nv.di$X = NULL

# Year2
t.y2.nv = read.csv(paste(path, "Year2-Non-Vol.csv", sep = ""), header = TRUE)
t.y2.nv$X = NULL

# Year2 -- With DI Index
t.y2.nv.di = read.csv(paste(path,
                            "Year2-Non-Vol-DI-Filtered.csv",
                            sep = ""),
                      header = TRUE)
t.y2.nv.di$X = NULL
```

Cleaning up the `path` variable:

```
rm(path)
```

14.2 Normalization Procedure

In this section, we will describe the procedure to normalize the score variables. We will normalize the scores for volunteers, based on the **mean** and **sd** values of non-volunteers. The formula used for normalization is as follows:

$$\frac{X_{Vol} - \mu_{NonVol}}{SD_{NonVol}} \tag{1}$$

Where, μ corresponds to the mean value.

It is noteworthy that the normalization will be conducted separately for each year, and for each type of score as we will see below:

14.2.1 Normalizing Total Score

```
# Normalizing the Total.Score.
# Year1
dt.y1$Nor.Score = (dt.y1$Total.Score - mean(t.y1.nv$Total.Score)) /
  sd(t.y1.nv$Total.Score)

# Year1 -- With DI Index
dt.y1.di$Nor.Score = (dt.y1.di$Total.Score - mean(t.y1.nv.di$Total.Score)) /
  sd(t.y1.nv.di$Total.Score)

# Year2
dt.y2$Nor.Score = (dt.y2$Total.Score - mean(t.y2.nv$Total.Score)) /
  sd(t.y2.nv$Total.Score)

# Year2 -- With DI Index
dt.y2.di$Nor.Score = (dt.y2.di$Total.Score - mean(t.y2.nv.di$Total.Score)) /
  sd(t.y2.nv.di$Total.Score)
```

14.2.2 Normalizing Score 'A'

```
# Normalizing the Score.A.
# Year1
dt.y1$Nor.Score.A = (dt.y1$Score.A - mean(t.y1.nv$Score.A)) /
  sd(t.y1.nv$Score.A)

# Year1 -- With DI Index
dt.y1.di$Nor.Score.A = (dt.y1.di$Score.A - mean(t.y1.nv.di$Score.A)) /
  sd(t.y1.nv.di$Score.A)

# Year2
dt.y2$Nor.Score.A = (dt.y2$Score.A - mean(t.y2.nv$Score.A)) /
  sd(t.y2.nv$Score.A)

# Year2 -- With DI Index
dt.y2.di$Nor.Score.A = (dt.y2.di$Score.A - mean(t.y2.nv.di$Score.A)) /
  sd(t.y2.nv.di$Score.A)
```

14.2.3 Normalizing Score 'B'

```
# Normalizing the Score.B.
# Year1
dt.y1$Nor.Score.B = (dt.y1$Score.B - mean(t.y1.nv$Score.B)) /
  sd(t.y1.nv$Score.B)

# Year1 -- With DI Index
dt.y1.di$Nor.Score.B = (dt.y1.di$Score.B - mean(t.y1.nv.di$Score.B)) /
  sd(t.y1.nv.di$Score.B)

# Year2
dt.y2$Nor.Score.B = (dt.y2$Score.B - mean(t.y2.nv$Score.B)) /
  sd(t.y2.nv$Score.B)

# Year2 -- With DI Index
dt.y2.di$Nor.Score.B = (dt.y2.di$Score.B - mean(t.y2.nv.di$Score.B)) /
  sd(t.y2.nv.di$Score.B)
```

14.2.4 Normalizing Score 'C'

```
# Normalizing the Score.C.
# Year1
dt.y1$Nor.Score.C = (dt.y1$Score.C - mean(t.y1.nv$Score.C)) /
  sd(t.y1.nv$Score.C)

# Year1 -- With DI Index
dt.y1.di$Nor.Score.C = (dt.y1.di$Score.C - mean(t.y1.nv.di$Score.C)) /
  sd(t.y1.nv.di$Score.C)

# Year2
dt.y2$Nor.Score.C = (dt.y2$Score.C - mean(t.y2.nv$Score.C)) /
  sd(t.y2.nv$Score.C)

# Year2 -- With DI Index
dt.y2.di$Nor.Score.C = (dt.y2.di$Score.C - mean(t.y2.nv.di$Score.C)) /
  sd(t.y2.nv.di$Score.C)
```

14.2.5 Normalizing Score 'BC'

```r
# Normalizing the Score.BC.
# Year1
dt.y1$Nor.Score.BC = (dt.y1$Score.BC - mean(t.y1.nv$Score.BC)) /
  sd(t.y1.nv$Score.BC)

# Year1 -- With DI Index
dt.y1.di$Nor.Score.BC = (dt.y1.di$Score.BC - mean(t.y1.nv.di$Score.BC)) /
  sd(t.y1.nv.di$Score.BC)

# Year2
dt.y2$Nor.Score.BC = (dt.y2$Score.BC - mean(t.y2.nv$Score.BC)) /
  sd(t.y2.nv$Score.BC)

# Year2 -- With DI Index
dt.y2.di$Nor.Score.BC = (dt.y2.di$Score.BC - mean(t.y2.nv.di$Score.BC)) /
  sd(t.y2.nv.di$Score.BC)
```

14.2.6 Cleaning Up

Since, we will no longer need the non-volunteer data, we will remove them from our workspace.

```r
# Cleaning up the non-volunteer data.
rm(t.y1.nv, t.y1.nv.di, t.y2.nv, t.y2.nv.di)
```

14.2.7 Selecting Relevant Columns and Combining Y1 and Y2 Data

In this section, we will select only the relevant columns, and then combine Year1 & Year2 data.

```r
# Select Columns
cols = c("ID.Anon", "Gender", "Section",
         "Code.BA", "Diploma.Obtained", "Category",
         "Country.Diploma", "Selected", "Reached.Adulthood",
         "Course.Year", "Condition", "Teacher.Name",
         "Exam.Present", "Final.Grade", "Total.Score",
         "Score.A", "Score.B", "Score.C",
         "Score.BC",
         "Nor.Score", "Nor.Score.A", "Nor.Score.B",
         "Nor.Score.C", "Nor.Score.BC")

# Combining the volunteer data.
dt = rbind(
  select(dt.y1, one_of(cols)),
  select(dt.y2, one_of(cols)))

# With DI Index
dt.di = rbind(
  select(dt.y1.di, one_of(cols)),
  select(dt.y2.di, one_of(cols))
)

# Clean-up
rm(cols)
```

14.2.8 Exporting Normalized Data

The aforementioned normalized data will be used further. So, we will simply export it to .csv files.

```r
# Setting the path.
path = paste(here(), "/Data/Scores/Normalized-Volunteer-Data/", sep = "")

# Export Year-1
```

```r
write.csv(dt.y1, paste(path, "Year1-Normalized-Score.csv", sep = ""))

# Export Year-1 -- With DI Index
write.csv(dt.y1.di, paste(path,
                          "Year1-Normalized-Score-DI-Filtered.csv",
                          sep = ""))

# Export Year-2
write.csv(dt.y2, paste(path, "Year2-Normalized-Score.csv", sep = ""))

# Export Year-2 -- With DI Index
write.csv(dt.y2.di, paste(path,
                          "Year2-Normalized-Score-DI-Filtered.csv",
                          sep = ""))

# Export Total Data
write.csv(dt, paste(path, "Total-Data-Normalized-Score.csv", sep = ""))

# Export Total Data -- With DI Index
write.csv(dt.di, paste(path,
                       "Total-Data-Normalized-Score-DI-Filtered.csv",
                       sep = ""))
```

14.2.9 Final Clean-Up

```r
rm(list = ls())
```