LEARN Center
EPFL

Analysis of Third Year Data

Academic year : 2019-20 (Year 3)

Himanshu Verma, Cécile Hardebolle

Flipped Classroom Project

2 juillet 2020

Table des matières

1 Background and Rationale

In this document, we will do the following :

1. ...

2 Packages

We will import all the necessary R packages here :

```
library(readxl)
library(dplyr)
library(tidyr)
library(ggplot2)
library(scales)
library(gridExtra)
library(gplots)
library(RColorBrewer)
library(FactoMineR)
library(factoextra)
library(nlme)
library(rcompanion)
library(here)
library(corrplot)
library(wesanderson)
library(psych)
library(ggpubr)
```

We will also define the two functions to Anonymize/De-Anonymize SCIPERs. These functions are `GenerateSCIPER(str x)` and `GenerateUniqueID(str x)`. Please note that these functions are masked from being shown in the PDF.

3 Importing and Preparing Data

In this section, we will import the data of volunteers, and scores of all students in the final semester exam to be able to assess the impact of flipped condition on the students' academic performance.

3.1 Volunteer Data

Firstly, we import the data of all the volunteers.

```
# Defining the path.
path = paste(here(), "/Data/Personal/", sep = "")

# Reading the .xlsx file.
# Flipped data
t.flipped = read_excel(paste(path,
                             "VolunteersList-Y3.xlsx",
                             sep = ""),
                       sheet = "Pilote")

## New names:
## * `<10 : Masculin, >9 Femini` -> `<10 : Masculin, >9 Femini...14`
## * `<10 : Masculin, >9 Femini` -> `<10 : Masculin, >9 Femini...17`

# Control data
t.control = read_excel(paste(path,
                             "VolunteersList-Y3.xlsx",
                             sep = ""),
                       sheet = "Controle")

## New names:
## * random -> random...15
## * random -> random...16
```

There are some columns which we will not need, so it would be good to simply remove them :

```
# Removing a few unnessary columns.
# Flipped data
t.flipped[, c(12:19)] = NULL

#Control data
t.control[, c(12:18)] = NULL
```

Next, we will rename the columns and add the `Condition` columns before merging the data.

```
# Renaming the column names.
newColumns = c("Full.Name", "Reached.Adulthood", "Gender",
               "SCIPER", "Section", "Email",
               "Creation.Time", "Code.BA", "Diploma.Obtained",
               "Category", "Country.Diploma")
names(t.flipped) = newColumns
names(t.control) = newColumns
```

... Creating a new column called `Condition` :

```
# New column: Condition
t.flipped$Condition = "Flipped"
t.control$Condition = "Control"
```

... and merging the *flipped* and *control* datasets into a single dataset :

```
# Combining the two datasets into a single one.
dt.y3 = rbind(t.flipped, t.control)

# Cleaning up.
rm(t.flipped, t.control, newColumns)
```

... in the following sub-sections, we will go through the individual variables and check if they are uniform.

```
# Displaying the column names.
names(dt.y3)

##  [1] "Full.Name"         "Reached.Adulthood" "Gender"            "SCIPER"
##  [5] "Section"           "Email"             "Creation.Time"     "Code.BA"
##  [9] "Diploma.Obtained"  "Category"          "Country.Diploma"   "Condition"
```

3.1.1 Gender

... showing the levels of this variable and recoding if necessary.

```
# Showing the values in this column.
levels(as.factor(dt.y3$Gender))

## [1] "F" "M"
```

... summarizing the students based on `Condition` and `Gender` :

```
# Summarizing.
dt.y3 %>% group_by(Condition, Gender) %>%
  summarise(Count = n())

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 4 x 3
## # Groups:   Condition [2]
##   Condition Gender Count
##   <chr>     <chr>  <int>
```

```
## 1 Control   F        111
## 2 Control   M        221
## 3 Flipped   F         69
## 4 Flipped   M        133
```

3.1.2 Code.BA

... showing the levels of this variable and recoding if necessary.

```
# Showing the values in this column.
levels(as.factor(dt.y3$Code.BA))

## [1] "Ex-CMS"      "Ex-MAN"      "Nouveaux"    "Redoublants"

# Recoding this column.
dt.y3$Code.BA = recode(dt.y3$Code.BA,
                       "Ex-CMS" = "Ex-CMS",
                       "Ex-MAN" = "Ex-MAN",
                       "Nouveaux" = "New",
                       "Redoublants" = "Repeating")
```

... summarizing the students' distribution :

```
# Across Code.BA
dt.y3 %>% group_by(Code.BA) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 5 x 2
##   Code.BA   Count
##   <chr>     <int>
## 1 Ex-CMS       12
## 2 Ex-MAN       86
## 3 New         401
## 4 Repeating    34
## 5 <NA>          1

# Across Code.BA, Gender
dt.y3 %>% group_by(Code.BA, Gender) %>%
  summarise(Count = n())

## `summarise()` regrouping output by 'Code.BA' (override with `.groups` argument)

## # A tibble: 9 x 3
## # Groups:   Code.BA [5]
##   Code.BA   Gender Count
##   <chr>     <chr>  <int>
## 1 Ex-CMS    F          3
## 2 Ex-CMS    M          9
## 3 Ex-MAN    F         38
## 4 Ex-MAN    M         48
## 5 New       F        127
## 6 New       M        274
## 7 Repeating F         12
## 8 Repeating M         22
## 9 <NA>      M          1

# Across Condition, Code.BA and Gender
dt.y3 %>% group_by(Condition, Code.BA, Gender) %>%
  summarise(Count = n())

## `summarise()` regrouping output by 'Condition', 'Code.BA' (override with `.groups` argument)
```

```
## # A tibble: 17 x 4
## # Groups:   Condition, Code.BA [9]
##    Condition Code.BA   Gender Count
##    <chr>     <chr>     <chr>  <int>
##  1 Control   Ex-CMS    F          2
##  2 Control   Ex-CMS    M          6
##  3 Control   Ex-MAN    F         24
##  4 Control   Ex-MAN    M         30
##  5 Control   New       F         78
##  6 Control   New       M        171
##  7 Control   Repeating F          7
##  8 Control   Repeating M         14
##  9 Flipped   Ex-CMS    F          1
## 10 Flipped   Ex-CMS    M          3
## 11 Flipped   Ex-MAN    F         14
## 12 Flipped   Ex-MAN    M         18
## 13 Flipped   New       F         49
## 14 Flipped   New       M        103
## 15 Flipped   Repeating F          5
## 16 Flipped   Repeating M          8
## 17 Flipped   <NA>      M          1
```

... we see that there is 1 student in the Flipped, we will remove this student because `Code.BA` is a mandatory variable to look into.

```
# Filtering student with no Code.BA
dt.y3 = dt.y3 %>% filter(!is.na(Code.BA))
```

3.1.3 Course.Year

We will create a new column called `Course.Year` whose value will be "`Y3-2019-20`" :

```
# Course.Year
dt.y3$Course.Year = "Y3-2019-20"
```

3.1.4 Category

... showing the levels of this variable and recoding if necessary.

```
# Showing the values in this column.
levels(as.factor(dt.y3$Category))

## [1] "Autres"         "Autres suisses" "France"         "PAM"

# Recoding this column.
dt.y3$Category = recode(dt.y3$Category,
                        "Autres" = "Etranger.Autres",
                        "Autres suisses" = "Suisse.Autres",
                        "France" = "France",
                        "PAM" = "Suisse.PAM")
```

... summarizing the students' distribution.

```
# Across Category
dt.y3 %>% group_by(Category) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 4 x 2
##   Category        Count
##   <chr>           <int>
```

```
## 1 Etranger.Autres    24
## 2 France            283
## 3 Suisse.Autres     110
## 4 Suisse.PAM        116

# Across Condition, Category
dt.y3 %>% group_by(Condition, Category) %>%
  summarise(Count = n())

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 8 x 3
## # Groups:   Condition [2]
##   Condition Category        Count
##   <chr>     <chr>           <int>
## 1 Control   Etranger.Autres    15
## 2 Control   France            173
## 3 Control   Suisse.Autres      72
## 4 Control   Suisse.PAM         72
## 5 Flipped   Etranger.Autres     9
## 6 Flipped   France            110
## 7 Flipped   Suisse.Autres      38
## 8 Flipped   Suisse.PAM         44

# Across Condition, Category, and Gender
dt.y3 %>% group_by(Condition, Category, Gender) %>%
  summarise(Count = n())

## `summarise()` regrouping output by 'Condition', 'Category' (override with `.groups` argument)

## # A tibble: 16 x 4
## # Groups:   Condition, Category [8]
##    Condition Category        Gender Count
##    <chr>     <chr>           <chr>  <int>
##  1 Control   Etranger.Autres F          7
##  2 Control   Etranger.Autres M          8
##  3 Control   France          F         59
##  4 Control   France          M        114
##  5 Control   Suisse.Autres   F         31
##  6 Control   Suisse.Autres   M         41
##  7 Control   Suisse.PAM      F         14
##  8 Control   Suisse.PAM      M         58
##  9 Flipped   Etranger.Autres F          4
## 10 Flipped   Etranger.Autres M          5
## 11 Flipped   France          F         43
## 12 Flipped   France          M         67
## 13 Flipped   Suisse.Autres   F         15
## 14 Flipped   Suisse.Autres   M         23
## 15 Flipped   Suisse.PAM      F          7
## 16 Flipped   Suisse.PAM      M         37
```

3.1.5 Reached.Adulthood

... we should not analyze minor students' data, as a result, we will filter them out here.

```
# Showing the levels of this variable.
levels(as.factor(dt.y3$Reached.Adulthood))

## [1] "Non" "Oui"

# Summarizing the distribution across Condition.
dt.y3 %>% group_by(Condition, Reached.Adulthood) %>%
  summarise(Count = n())
```

```
## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 4 x 3
## # Groups:   Condition [2]
##   Condition Reached.Adulthood Count
##   <chr>     <chr>             <int>
## 1 Control   Non                  58
## 2 Control   Oui                 274
## 3 Flipped   Non                  40
## 4 Flipped   Oui                 161
```

There are 435 students who reached adulthood before starting their semester, and there are 98 students who were minors.

For further analysis, we will remove the minors from our analysis. At the same time, we will keep a dataset with minors in case of future analysis requirements.

```
# Data without minors.
dt.sm = dt.y3 %>% filter(Reached.Adulthood == "Oui")
```

3.1.6 Anonymizing Data

... we won't need the `Full.Name`, `Email`, `Creation.Time` :

```
# Removing some personal columns.
colNames = c("Full.Name", "Email", "Creation.Time")
dt.y3 = dt.y3 %>% select(-one_of(colNames))
dt.sm = dt.sm %>% select(-one_of(colNames))
```

... convert the `SCIPER` to `ID.Anon` (Please not that we will remove the `SCIPER` column when exporting the data).

```
# Anonymizing SCIPERs.
dt.y3$ID.Anon = lapply(dt.y3$SCIPER, GenerateUniqueID)
dt.y3$ID.Anon = unlist(dt.y3$ID.Anon)

dt.sm$ID.Anon = lapply(dt.sm$SCIPER, GenerateUniqueID)
dt.sm$ID.Anon = unlist(dt.sm$ID.Anon)
```

3.2 Non-Volunteer Data

In this section, we will import and prepare the data containing students' scores. This data will be merged with the students' personal data.

```
# Importing score data for Linear Algebra.

# Path to the file
path = paste(here(),
             "/Data/Personal/FinalExam-Y3.xlsx",
             sep = "")

# Reading the data individually for each teacher.
t.friedli = read_excel(path, sheet = "data")[1:243,]
t.deparis = read_excel(path, sheet = "data")[245:420,]
t.maddocks = read_excel(path, sheet = "data")[422:671,]
t.hess = read_excel(path, sheet = "data")[673:902,]
t.urech = read_excel(path, sheet = "data")[904:1241,]
t.jecker = read_excel(path, sheet = "data")[1243:1527,]
t.zuleta = read_excel(path, sheet = "data")[1529:1709,]
t.semmler = read_excel(path, sheet = "data")[1711:1876,]
```

... setting the first line of above datasets as the header :

```
# Defining new headers.
colnames(t.deparis) = as.character(unlist(t.deparis[1,]))
t.deparis = t.deparis[-1,]
colnames(t.friedli) = as.character(unlist(t.friedli[1,]))
t.friedli = t.friedli[-1,]
colnames(t.hess) = as.character(unlist(t.hess[1,]))
t.hess = t.hess[-1,]
colnames(t.jecker) = as.character(unlist(t.jecker[1,]))
t.jecker = t.jecker[-1,]
colnames(t.maddocks) = as.character(unlist(t.maddocks[1,]))
t.maddocks = t.maddocks[-1,]
colnames(t.semmler) = as.character(unlist(t.semmler[1,]))
t.semmler = t.semmler[-1,]
colnames(t.urech) = as.character(unlist(t.urech[1,]))
t.urech = t.urech[-1,]
colnames(t.zuleta) = as.character(unlist(t.zuleta[1,]))
t.zuleta = t.zuleta[-1,]
```

... defining the columns which we will retain in the above datasets :

```
# Defining the list of columns.
colNames = c("TEACHER", "PRESENT", "GRADE", "POINTS 80%", "SCIPER",
             "MC-calc-moindre-carres", "MC-calc-syst-lineaire",
             "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", "q:MC-calc-matrice-matrice",
             "q:MC-calc-ortho-diag", "q:MC-calc-passage",
             "q:MC-calc-proj-ortho", "q:MC-calc-rank",
             "q:MC-calc-span", "q:MC-theory-det",
             "q:MC-theory-diagonalisable", "q:MC-theory-diagonalisation",
             "q:MC-theory-matrice-orthogonale", "q:MC-theory-moindres-carres",
             "q:MC-theory-sous-espaces", "q:MC-theory-syst-lineaire",
             "MC-calc-moindre-carres-A18", "q:MC-calc-matrice-poly")
```

... selecting the relevant columns for datasets for each professor :

```
# Deparis
t.deparis = t.deparis[,1:67] %>% select(one_of(colNames))

## Warning: Unknown columns: ‘MC-calc-moindre-carres-A18‘, ‘q:MC-calc-matrice-poly‘

# Friedli
t.friedli = t.friedli[,1:67] %>% select(one_of(colNames))

## Warning: Unknown columns: ‘MC-calc-moindre-carres-A18‘, ‘q:MC-calc-matrice-poly‘

# Hess
t.hess = t.hess[,1:63] %>% select(one_of(colNames))

## Warning: Unknown columns: ‘MC-calc-moindre-carres-A18‘, ‘q:MC-calc-matrice-poly‘

# Jecker
t.jecker = t.jecker[,1:65] %>% select(one_of(colNames))

## Warning: Unknown columns: ‘MC-calc-moindre-carres-A18‘, ‘q:MC-calc-matrice-poly‘

# Maddocks
t.maddocks = t.maddocks[,1:77] %>% select(one_of(colNames))

## Warning: Unknown columns: ‘MC-calc-moindre-carres‘, ‘q:MC-calc-matrice-matrice‘

# Semmler
t.semmler = t.semmler[,1:67] %>% select(one_of(colNames))

## Warning: Unknown columns: ‘MC-calc-moindre-carres-A18‘, ‘q:MC-calc-matrice-poly‘
```

```
# Urech
t.urech = t.urech[,1:67] %>% select(one_of(colNames))

## Warning: Unknown columns: `MC-calc-moindre-carres-A18`, `q:MC-calc-matrice-poly`

# Zuleta
t.zuleta = t.zuleta[,1:69] %>% select(one_of(colNames))

## Warning: Unknown columns: `MC-calc-moindre-carres-A18`, `q:MC-calc-matrice-poly`
```

Each teacher asked 22 common questions from the original set of 24. We also observe that all excpet one teacher did not ask two questions : `MC-calc-moindre-carres-A18` and `q:MC-calc-matrice-poly`. However, one professor removed two other questions : `MC-calc-moindre-carres` and `q:MC-calc-matrice-matrice`.

After a quick discussion with Simone, we will consider these questions as equivalent.

```
# Renaming the two questions for t.maddocks.
names(t.maddocks)[names(t.maddocks) == "MC-calc-moindre-carres-A18"] <- "MC-calc-moindre-carres"
names(t.maddocks)[names(t.maddocks) == "q:MC-calc-matrice-poly"] <- "q:MC-calc-matrice-matrice"
```

... combining all these datasets into a single dataset.

```
# Defining Column Names
colNames = c("TEACHER", "PRESENT", "GRADE", "POINTS 80%", "SCIPER",
             "MC-calc-moindre-carres", "MC-calc-syst-lineaire",
             "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", "q:MC-calc-matrice-matrice",
             "q:MC-calc-ortho-diag", "q:MC-calc-passage",
             "q:MC-calc-proj-ortho", "q:MC-calc-rank",
             "q:MC-calc-span", "q:MC-theory-det",
             "q:MC-theory-diagonalisable", "q:MC-theory-diagonalisation",
             "q:MC-theory-matrice-orthogonale", "q:MC-theory-moindres-carres",
             "q:MC-theory-sous-espaces", "q:MC-theory-syst-lineaire")

# Combining datasets.
dt.y3.exam = rbind(
  t.deparis %>% select(one_of(colNames)),
  t.friedli %>% select(one_of(colNames)),
  t.hess %>% select(one_of(colNames)),
  t.jecker %>% select(one_of(colNames)),
  t.maddocks %>% select(one_of(colNames)),
  t.semmler %>% select(one_of(colNames)),
  t.urech %>% select(one_of(colNames)),
  t.zuleta %>% select(one_of(colNames))
)
```

... removing all the individual datasets except `t.deparis`, which we will use later to examine how many people changed classes in the middle of the semester.

```
# Cleaning up.
rm(colNames, path,
   t.friedli, t.hess, t.jecker,
   t.maddocks, t.semmler, t.urech,
   t.zuleta)
```

... renaming all the columns :

```
# Renaming all the columns.
colNames = c("Teacher.Name", "Exam.Present",
                       "Final.Grade", "Points.80.Percent",
                       "SCIPER", "MC.Calc.Moindre.Carres",
                       "MC.Calc.Syst.Lineaire", "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",
                        "Q.MC.Calc.Matrice.Matrice", "Q.MC.Calc.Ortho.Diag",
                        "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
                        "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
                        "Q.MC.Theory.Det", "Q.MC.Theory.Diagonalisable",
                        "Q.MC.Theory.Diagonalisation", "Q.MC.Theory.Matrice.Orthogonale",
                        "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces",
                        "Q.MC.Theory.Syst.Lineaire")
names(dt.y3.exam) = colNames
names(t.deparis) = colNames
```

... we convert all the character scores into numerical values.

```
# Defining the column names which must be converted to integer.
colNames = c("MC.Calc.Moindre.Carres",
                        "MC.Calc.Syst.Lineaire", "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",
                        "Q.MC.Calc.Matrice.Matrice", "Q.MC.Calc.Ortho.Diag",
                        "Q.MC.Calc.Passage", "Q.MC.Calc.Proj.Ortho",
                        "Q.MC.Calc.Rank", "Q.MC.Calc.Span",
                        "Q.MC.Theory.Det", "Q.MC.Theory.Diagonalisable",
                        "Q.MC.Theory.Diagonalisation", "Q.MC.Theory.Matrice.Orthogonale",
                        "Q.MC.Theory.Moindres.Carres", "Q.MC.Theory.Sous.Espaces",
                        "Q.MC.Theory.Syst.Lineaire")
# Converting character scores to integer.
# Full Dataset
dt.y3.exam[,colNames] = apply(dt.y3.exam[,colNames],
                              2, # Column Operation
                              function(x) as.numeric(as.character(x)))

# Deparis Dataset
t.deparis[,colNames] = apply(t.deparis[,colNames],
                             2, # Column operation
                             function(x) as.numeric(as.character(x)))

# Cleaning up.
rm(colNames)
```

... next, we will remove questions which have a **Discriminatory Index** value of less than `0.33`.

3.2.1 Identifying Questions with Low DI

We will import the global DI values and remove questions which have a DI value less than 0.33.

```
# Path variable
path = paste(
  here(),
  "/Data/Question-DI/Y3-DI.csv",
  sep = "")

# Loading the .csv file.
t.di = read.csv(path, header = TRUE)

# Setting the first row as the column headers.
colnames(t.di) = as.character(unlist(t.di[1,]))
t.di = t.di[-1,]
```

... let's check which questions have a value less than 0.33.

```r
# Selecting relevant columns.
t.di = t.di %>% select("question_id", "DI")
t.di$DI = as.numeric(as.character(t.di$DI))

# Filtering rows which have a value less than 0.33
t.di %>% filter(DI <= 0.33)

##         question_id        DI
## 1 q:MC-calc-inverse 0.3276956
```

We see that there is only one question "`Q.MC.Calc.Inverse`" which has a global DI less than 0.33. Please note that this criteria of choosing 0.33 is based on the first 2 iterations of the study.

```r
# Cleaning up.
rm(t.di)
```

3.2.2 Removing Absentees

In this section, we will remove all the absentees who were absent (or possibly dropped out of) during the linear algebra exam. It is worth noting that we will remove absentees for `dt.y3.exam` and `t.deparis` datasets.

First, let's summarise how many students were actually absent in exam.

```r
# Summarizing the number of absentees.
dt.y3.exam %>% filter(Exam.Present == 0) %>%
  select(SCIPER, Teacher.Name) %>%
  group_by(Teacher.Name) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Deparis         10
## 2 Friedli         10
## 3 Hess            13
## 4 Jecker          10
## 5 Maddocks        16
## 6 Semmler          8
## 7 Urech           27
## 8 Zuleta          13
```

In summary, there are in total 107 absentees. Which we will remove below.

```r
# Removing Absentees
# Full dataset
dt.y3.exam = dt.y3.exam %>% filter(Exam.Present == 1)

# Deparis dataset
t.deparis = t.deparis %>% filter(Exam.Present == 1)
```

3.2.3 Computing Total Score

Since this year's course was completely flipped, so there is no need to categorize the questions belonging to the parts 'A', 'B', or 'C'. We will instead use the complete score.

```r
# Computing Total Score
# Full Dataset
dt.y3.exam$Total.Score = rowSums(dt.y3.exam[, c(6:27)])
# With DI
dt.y3.exam$Total.Score.DI = rowSums(dt.y3.exam[, c(6:12, 14:27)])
```

```r
# Deparis dataset
t.deparis$Total.Score = rowSums(t.deparis[, c(6:27)])
# With DI
t.deparis$Total.Score.DI = rowSums(t.deparis[, c(6:12, 14:27)])
```

3.2.4 Anonymizing Data

We will convert the SCIPERs to a new variable `ID.Anon`. However, we will not remove the SCIPER as of now because we will use it for the join operation later. We will remove the SCIPER when we are exporting the datasets.

```r
# Anonymizing datasets
# Full Dataset
dt.y3.exam$ID.Anon = lapply(dt.y3.exam$SCIPER, GenerateUniqueID)
dt.y3.exam$ID.Anon = unlist(dt.y3.exam$ID.Anon)

# Deparis Dataset
t.deparis$ID.Anon = lapply(t.deparis$SCIPER, GenerateUniqueID)
t.deparis$ID.Anon = unlist(t.deparis$ID.Anon)
```

3.3 Merging Personal and Exam Datasets

In this section, we will merge the Personal and Exam datasets. In addition, we will separate the *non-volunteer* and *volunteer* datasets so that we can compute the normalized scores later.

3.3.1 Non-Volunteers

In order to get the list of non-volunteers, we will do a join operation with `dt.y3.exam` data-frame and then filter out the rows where the teacher is Simone.

```r
# Outer (right) join to get the list of non-volunteers.
# Without Minors
y3.non.vol.sm = merge(x = dt.sm,
                      y = dt.y3.exam,
                      by = "SCIPER",
                      all.y = "TRUE")
# With Minors
y3.non.vol.am = merge(x = dt.y3,
                      y = dt.y3.exam,
                      by = "SCIPER",
                      all.y = "TRUE")

# All rows with values for Gender, Course.Year, Condition, etc. are
# non-volunteers.
y3.non.vol.sm = y3.non.vol.sm %>% filter(is.na(Condition))
y3.non.vol.am = y3.non.vol.am %>% filter(is.na(Condition))

# Removing ID.Anon columns and renaming one of them so that we
# have only one ID.Anon column.
y3.non.vol.am$ID.Anon.y = NULL
y3.non.vol.sm$ID.Anon.y = NULL

names(y3.non.vol.am)[names(y3.non.vol.am) == "ID.Anon.x"] = "ID.Anon"
names(y3.non.vol.sm)[names(y3.non.vol.sm) == "ID.Anon.x"] = "ID.Anon"

y3.non.vol.am$ID.Anon = lapply(y3.non.vol.am$SCIPER, GenerateUniqueID)
y3.non.vol.am$ID.Anon = unlist(y3.non.vol.am$ID.Anon)
y3.non.vol.sm$ID.Anon = lapply(y3.non.vol.sm$SCIPER, GenerateUniqueID)
y3.non.vol.sm$ID.Anon = unlist(y3.non.vol.sm$ID.Anon)
```

Now, we will check if there are some students who initially registered with other teachers but took exam with Deparis.

```
# With Minors
y3.non.vol.am %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 7 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Friedli        201
## 2 Hess           150
## 3 Jecker         224
## 4 Maddocks       180
## 5 Semmler        112
## 6 Urech          243
## 7 Zuleta         137

# Without Minors
y3.non.vol.sm %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Deparis         28
## 2 Friedli        209
## 3 Hess           158
## 4 Jecker         235
## 5 Maddocks       188
## 6 Semmler        118
## 7 Urech          261
## 8 Zuleta         145
```

In the `y3.non.vol.sm` (without minors) dataset, we see that there are 28 students who took exam with Deparis.

Let's examine if these 28 students are present in the initial personal datasets :

```
# To examine the students who took exam with Deparis,
# and did not initially volunteer.
y3.exam.with.deparis = y3.non.vol.sm %>%
  filter(Teacher.Name == "Deparis")

# Inner join with the personal data.
# With Minors.
t.stat.am = merge(x = dt.y3,
                  y = y3.exam.with.deparis,
                  by = "SCIPER")
nrow(t.stat.am)

## [1] 28

# Without Minors.
t.stat = merge(x = dt.sm,
               y = y3.exam.with.deparis,
               by = "SCIPER")
nrow(t.stat)

## [1] 0

# Cleaning up.
rm(t.stat, t.stat.am, y3.exam.with.deparis)
```

We see that these students are the ones who are minors, and also volunteered for the study. So, we will remove them from the non-volunteer dataset `y3.non.vol.sm` because these were volunteers and should not be part of the non-volunteer dataset.

```
# Removing some Deparis students from non-volunteer dataset.
y3.non.vol.sm = y3.non.vol.sm %>% filter(!(Teacher.Name == "Deparis"))
```

3.3.2 Volunteers

In this section, we will generate the datset of volunteers. These are the students who volunteered to participate in the experiment.

```
# Inner join operation.
# With Minors
y3.vol.am = merge(x = dt.y3,
                  y = dt.y3.exam,
                  by = "SCIPER")

# Without Minors
y3.vol.sm = merge(x = dt.sm,
                  y = dt.y3.exam,
                  by = "SCIPER")

# Removing ID.Anon columns and renaming one of them so that we
# have only one ID.Anon column.
y3.vol.am$ID.Anon.y = NULL
y3.vol.sm$ID.Anon.y = NULL

names(y3.vol.am)[names(y3.vol.am) == "ID.Anon.x"] = "ID.Anon"
names(y3.vol.sm)[names(y3.vol.sm) == "ID.Anon.x"] = "ID.Anon"

y3.vol.am$ID.Anon = lapply(y3.vol.am$SCIPER, GenerateUniqueID)
y3.vol.am$ID.Anon = unlist(y3.vol.am$ID.Anon)
y3.vol.sm$ID.Anon = lapply(y3.vol.sm$SCIPER, GenerateUniqueID)
y3.vol.sm$ID.Anon = unlist(y3.vol.sm$ID.Anon)
```

... let's just summarize these datasets with `Teacher.Name`

```
# Summarizing by Teacher.Name
# With Minors
y3.vol.am %>% group_by(Teacher.Name) %>%
  summarise(Count = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Deparis        165
## 2 Friedli         31
## 3 Hess            66
## 4 Jecker          50
## 5 Maddocks        53
## 6 Semmler         45
## 7 Urech           67
## 8 Zuleta          30
```

```
# Without Minors
y3.vol.sm %>% group_by(Teacher.Name) %>%
  summarise(Count = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Deparis        137
## 2 Friedli         23
## 3 Hess            58
## 4 Jecker          39
## 5 Maddocks        45
## 6 Semmler         39
## 7 Urech           49
## 8 Zuleta          22
```

In `y3.vol.am` (with minors) dataset, the number of volunteers who took exam with Deparis is the same as the number of students in the `t.deparis` exam dataset. So, we will just remove the `t.deparis` dataset.

The number of volunteers who were absent/dropped out is simply the difference of initial and final datasets, and the values are 26 (with minors) and 23 (without minors).

... let's just summarize the number of absentee volunteers across `Condition`.

```
# Absentee volunteers across condition.
# With Minors
y3.absentee.am = merge(x = dt.y3,
                       y = dt.y3.exam,
                       by = "SCIPER",
                       all.x = "TRUE")
y3.absentee.am = y3.absentee.am %>% filter(is.na(Teacher.Name))
# Summarizing...
y3.absentee.am %>% group_by(Condition) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

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

# Withot Minors
y3.absentee.sm = merge(x = dt.sm,
                       y = dt.y3.exam,
                       by = "SCIPER",
                       all.x = "TRUE")
y3.absentee.sm = y3.absentee.sm %>% filter(is.na(Teacher.Name))
# Summarizing ...
y3.absentee.sm %>% group_by(Condition) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 2
##   Condition Count
##   <chr>     <int>
## 1 Control      11
## 2 Flipped      12
```

... one final check we would like to do is to examine if there are certain students who registered as volunteers and assigned to Flipped condition, but then took exam with some other teacher besides Deparis :

```
# Flipped students with other teacher.
# With minors.
t.am = y3.vol.am %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")
```

```
# ... filter out these students.
y3.vol.am = y3.vol.am %>%
  filter(!(SCIPER %in% t.am$SCIPER))

# Without Minors.
t.sm = y3.vol.sm %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name != "Deparis" & Condition == "Flipped")
# ... filter out these students.
y3.vol.sm = y3.vol.sm %>%
  filter(!(SCIPER %in% t.sm$SCIPER))
```

We see that there are 23 (with minors) and 12 (without minors) students who were initially assigned to the Flipped condition but took an exam with other teacher beside Simone.

... let's also check if there are students in the Control condition who took exam with Simone.

```
# Control students with Simone.
# With Minors.
t.am = y3.vol.am %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name == "Deparis" & Condition == "Control")

# Withour Minors.
t.sm = y3.vol.sm %>% group_by(Teacher.Name) %>%
  filter(Teacher.Name == "Deparis" & Condition == "Control")
```

We see that there are 0 (with minors) and 0 (without minors) students who were initially assigned to the Control condition but took an exam with Simone.

... basic clean up of datasets that are not necessary.

```
# Cleaning up.
rm(t.deparis,
   y3.absentee.am,
   y3.absentee.sm,
   t.am, t.sm)
```

3.3.3 Normalized Score

In this section, we will normalize the score of volunteers based on the non-volunteer students.

```
# Normalization
# With minors.
y3.vol.am$Nor.Score = (y3.vol.am$Total.Score - mean(y3.non.vol.am$Total.Score)) /
  sd(y3.non.vol.am$Total.Score)
# With DI
y3.vol.am$Nor.Score.DI = (y3.vol.am$Total.Score.DI - mean(y3.non.vol.am$Total.Score.DI)) /
  sd(y3.non.vol.am$Total.Score.DI)

# Without Minors.
y3.vol.sm$Nor.Score = (y3.vol.sm$Total.Score - mean(y3.non.vol.sm$Total.Score)) /
  sd(y3.non.vol.sm$Total.Score)
# With DI
y3.vol.sm$Nor.Score.DI = (y3.vol.sm$Total.Score.DI - mean(y3.non.vol.sm$Total.Score.DI)) /
  sd(y3.non.vol.sm$Total.Score.DI)
```

3.4 Exporting Data

In this section, we will export the datasets :

```
# Personal data
path = paste(here(), "/Data/Personal/Filtered/", sep = "")
# With minors.
#dt.y3$SCIPER = NULL
```

```
write.csv(dt.y3,
          paste(path, "Year3-Personal-With-Minors.csv", sep = ""))
# Without minors.
#dt.sm$SCIPER = NULL
write.csv(dt.sm,
          paste(path, "Year3-Personal-Sans-Minors.csv", sep = ""))

# Exam Data
dt.y3.exam$SCIPER = NULL
write.csv(dt.y3.exam,
          paste(path, "Year3-Exam.csv", sep = ""))

# Volunteer Data
path = paste(here(), "/Data/Scores/Normalized-Volunteer-Data/", sep = "")
# With minors.
y3.vol.am$SCIPER = NULL
write.csv(y3.vol.am,
          paste(path, "Year3-Normalized-Score-With-Minors.csv", sep = ""))
# Without minors.
y3.vol.sm$SCIPER = NULL
write.csv(y3.vol.sm,
          paste(path, "Year3-Normalized-Score-Sans-Minors.csv", sep = ""))

# Non-Volunteer Data
path = paste(here(), "/Data/Non-Volunteers/", sep = "")
# With minors.
y3.non.vol.am$SCIPER = NULL
write.csv(y3.non.vol.am,
          paste(path, "Year3-Non-Vol-With-Minors.csv", sep = ""))
# Without minors.
y3.non.vol.sm$SCIPER = NULL
write.csv(y3.non.vol.sm,
          paste(path, "Year3-Non-Vol-Sans-Minors.csv", sep = ""))

# Cleaning up.
rm(path)
```

3.5 Analysis I Data

In this section, we will import and prepare the data containing students' scores from the "Analyse-I" exam. Later, we will join this data with the students' personal data.

```
# Importing score data for Analysis-I.

# Path to the file.
path = paste(
  here(),
  "/Data/Year-3-Analysis-I/Analysis-1-Year-3.xlsx",
  sep = "")

# Reading the data individually for each teacher.
t.lachowska = read_excel(path, sheet = "data")[1:358,]
t.arevalo = read_excel(path, sheet = "data")[360:576,]
t.friedli = read_excel(path, sheet = "data")[578:863,]
t.kressner = read_excel(path, sheet = "data")[865:898,]
t.patakfalvi = read_excel(path, sheet = "data")[900:1039,]
t.wittwer = read_excel(path, sheet = "data")[1041:1359,]
t.buffa = read_excel(path, sheet = "data")[1361:1559,]
t.favi = read_excel(path, sheet = "data")[1561:1859,]
```

... setting the first line of above datasets as the header :

```
# Defining new headers.
colnames(t.lachowska) = as.character((unlist(t.lachowska[1,])))
t.lachowska = t.lachowska[-1,]
colnames(t.arevalo) = as.character((unlist(t.arevalo[1,])))
t.arevalo = t.arevalo[-1,]
colnames(t.friedli) = as.character((unlist(t.friedli[1,])))
t.friedli = t.friedli[-1,]
colnames(t.kressner) = as.character((unlist(t.kressner[1,])))
t.kressner = t.kressner[-1,]
colnames(t.patakfalvi) = as.character((unlist(t.patakfalvi[1,])))
t.patakfalvi = t.patakfalvi[-1,]
colnames(t.wittwer) = as.character((unlist(t.wittwer[1,])))
t.wittwer = t.wittwer[-1,]
colnames(t.buffa) = as.character((unlist(t.buffa[1,])))
t.buffa = t.buffa[-1,]
colnames(t.favi) = as.character((unlist(t.favi[1,])))
t.favi = t.favi[-1,]
```

... defining the columns which we will retain in the above datasets.

```
colNames = c("TEACHER", "PRESENT", "GRADE", "POINTS 80%", "SCIPER",
             "QCM-complexes-A", "QCM-cont-vs-derivab-A",
             "QCM-contin-deriv-C1-B", "QCM-dev-limite-B",
             "QCM-inf-sup-A", "QCM-int-generalisee-B",
             "QCM-integrale-first-A", "QCM-integrale-second-B",
             "QCM-limite-prolongmt-A", "QCM-limsup-liminf-B",
             "QCM-serie-B", "QCM-serie-entiere-B",
             "QCM-serie-parametre-B", "QCM-suites-convergence-C",
             "QCM-suites-recurrence-A", "QCM-suites-recurrence-B",
             "QCM-theo-accr-finis-B-NEW", "QCM-val-intermed-image-interv-B",
             "TF-complexes-B", "TF-cont-deriv-C1-A",
             "TF-derivabilite-discussion-B", "TF-dev-limite-C",
             "TF-fonction-etc-A", "TF-induction-suites-limites-B",
             "TF-integrale-A", "TF-limite-continuite-B",
             "TF-serie-B", "TF-serie-entiere-A")
```

... selecting the relevant columns from above datasets for each professor.

```
# Arevalo
t.arevalo = t.arevalo[,1:78] %>% select(one_of(colNames))
# Buffa
t.buffa = t.buffa[,1:78] %>% select(one_of(colNames))
# Favi
t.favi = t.favi[,1:78] %>% select(one_of(colNames))
# Friedli
t.friedli = t.friedli[,1:78] %>% select(one_of(colNames))
# Kressner
t.kressner = t.kressner[,1:78] %>% select(one_of(colNames))

## Warning: Unknown columns: `TF-induction-suites-limites-B`, `TF-serie-B`

# Lachowska
t.lachowska = t.lachowska[,1:78] %>% select(one_of(colNames))
# Patakfalvi
t.patakfalvi = t.patakfalvi[,1:78] %>% select(one_of(colNames))
# Wittwer
t.wittwer = t.wittwer[,1:78] %>% select(one_of(colNames))
```

We see that all teachers except Kressner ask all the common questions. However, Kressner does not ask 2 questions "`TF-induction-suites-limites-B`" and "TF-serie-B". Therefore, in order to compute the common score for all students, we will simply remove these questions from other teachers in order to compute the `Total.Score`.

... updating the set of questions, and combining all the datasets.

```
# Updated header.
colNames = c("TEACHER", "PRESENT", "GRADE", "POINTS 80%", "SCIPER",
             "QCM-complexes-A", "QCM-cont-vs-derivab-A",
             "QCM-contin-deriv-C1-B", "QCM-dev-limite-B",
             "QCM-inf-sup-A", "QCM-int-generalisee-B",
             "QCM-integrale-first-A", "QCM-integrale-second-B",
             "QCM-limite-prolongmt-A", "QCM-limsup-liminf-B",
             "QCM-serie-B", "QCM-serie-entiere-B",
             "QCM-serie-parametre-B", "QCM-suites-convergence-C",
             "QCM-suites-recurrence-A", "QCM-suites-recurrence-B",
             "QCM-theo-accr-finis-B-NEW", "QCM-val-intermed-image-interv-B",
             "TF-complexes-B", "TF-cont-deriv-C1-A",
             "TF-derivabilite-discussion-B", "TF-dev-limite-C",
             "TF-fonction-etc-A",
             "TF-integrale-A", "TF-limite-continuite-B",
             "TF-serie-entiere-A")

# Combining datasets.
dt.analyse = rbind(
  t.arevalo %>% select(one_of(colNames)),
  t.buffa %>% select(one_of(colNames)),
  t.favi %>% select(one_of(colNames)),
  t.friedli %>% select(one_of(colNames)),
  t.kressner %>% select(one_of(colNames)),
  t.lachowska %>% select(one_of(colNames)),
  t.patakfalvi %>% select(one_of(colNames)),
  t.wittwer %>% select(one_of(colNames))
)

# Removing the individual datasets.
rm(colNames, path,
   t.arevalo, t.buffa,
   t.favi, t.friedli,
   t.kressner, t.lachowska,
   t.patakfalvi, t.wittwer)
```

... renaming all the columns :

```
# Renaming all the columns.
colNames = c("Teacher.Name", "Exam.Present", "Final.Grade",
             "Points.80.Percent", "SCIPER",
             "QCM.Complexes.A", "QCM.Cont.Vs.Derivab.A",
             "QCM.Contin.Deriv.C1.B", "QCM.Dev.Limite.B",
             "QCM.Inf.Sup.A", "QCM.Int.Generalisee.B",
             "QCM.Integrale.First.A", "QCM.Integrale.Second.B",
             "QCM.Limite.Prolongmt.A", "QCM.Limsup.Liminf.B",
             "QCM.Serie.B", "QCM.Serie.Entiere.B",
             "QCM.Serie.Parametre.B", "QCM.Suites.Convergence.C",
             "QCM.Suites.Recurrence.A", "QCM.Suites.Recurrence.B",
             "QCM.Theo.Accr.Finis.B.NEW", "QCM.Val.Intermed.Image.Interv.B",
             "TF.Complexes.B", "TF.Cont.Deriv.C1.A",
             "TF.Derivabilite.Discussion.B", "TF.Dev.Limite.C",
             "TF.Fonction.Etc.A",
             "TF.Integrale.A", "TF.Limite.Continuite.B",
             "TF.Serie.Entiere.A")

names(dt.analyse) = colNames
```

... we convert all the character scores into numerical values.

```
# Defining the columns which must be converted to integer
colNames = c("QCM.Complexes.A", "QCM.Cont.Vs.Derivab.A",
             "QCM.Contin.Deriv.C1.B", "QCM.Dev.Limite.B",
```

```
              "QCM.Inf.Sup.A", "QCM.Int.Generalisee.B",
              "QCM.Integrale.First.A", "QCM.Integrale.Second.B",
              "QCM.Limite.Prolongmt.A", "QCM.Limsup.Liminf.B",
              "QCM.Serie.B", "QCM.Serie.Entiere.B",
              "QCM.Serie.Parametre.B", "QCM.Suites.Convergence.C",
              "QCM.Suites.Recurrence.A", "QCM.Suites.Recurrence.B",
              "QCM.Theo.Accr.Finis.B.NEW", "QCM.Val.Intermed.Image.Interv.B",
              "TF.Complexes.B", "TF.Cont.Deriv.C1.A",
              "TF.Derivabilite.Discussion.B", "TF.Dev.Limite.C",
              "TF.Fonction.Etc.A",
              "TF.Integrale.A", "TF.Limite.Continuite.B",
              "TF.Serie.Entiere.A")

# Converting character scores to integer.
dt.analyse[,colNames] = apply(dt.analyse[,colNames],
                              2, # Column operation
                              function(x) as.numeric(as.character(x)))
# Cleaning up.
rm(colNames)
```

3.5.1 Removing Absentees

In this section, we will remove all the students who were absent during the exam.
... summarizing the number of absentees.

```
# Summary of absentees
dt.analyse %>% filter(Exam.Present == 0) %>%
  select(SCIPER, Teacher.Name) %>%
  group_by(Teacher.Name) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Arevalo         10
## 2 Buffa           12
## 3 Favi            14
## 4 Friedli          8
## 5 Kressner         3
## 6 Lachowska       17
## 7 Patakfalvi      10
## 8 Wittwer         10
```

In summary there are 84 absentees, which we will remove below.

```
# Removing absentees.
dt.analyse = dt.analyse %>% filter(Exam.Present == 1)
```

3.5.2 Computing Total Score

... we will create a new variable called the `Total.Score` :

```
dt.analyse$Total.Score = rowSums(dt.analyse[, c(6:31)])
```

3.5.3 Merging Personal and Exam Datasets : Non-Volunteers

In order to get the list of non-volunteers, we will do a join operation with `dt.analyse`, and then filter out the rows where the `Condition` value is NULL.

```r
# Outer (right) join to get the list of non-volunteers.
# Sans Minors
an.non.vol.sm = merge(x = dt.sm,
                      y = dt.analyse,
                      by = "SCIPER",
                      all.y = TRUE)

# Avec Minors
an.non.vol.am = merge(x = dt.y3,
                      y = dt.analyse,
                      by = "SCIPER",
                      all.y = TRUE)

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

... let's just summarize the non-volunteers according to `Teacher.Name`

```r
# Avec Minors
an.non.vol.am %>% group_by(Teacher.Name) %>%
  summarise(Count = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Arevalo        174
## 2 Buffa          147
## 3 Favi           170
## 4 Friedli        211
## 5 Kressner        23
## 6 Lachowska      249
## 7 Patakfalvi     103
## 8 Wittwer        184
```

```r
# Sans Minors
an.non.vol.sm %>% group_by(Teacher.Name) %>%
  summarise(Count = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Arevalo        182
## 2 Buffa          157
## 3 Favi           186
## 4 Friedli        226
## 5 Kressner        24
## 6 Lachowska      268
## 7 Patakfalvi     106
## 8 Wittwer        208
```

3.5.4 Merging Personal and Exam Datasets : Volunteers

In this section, we will generate the dataset for volunteers.

```r
# Inner join operation.
# Avec Minors.
an.vol.am = merge(x = dt.y3,
```

```
                  y = dt.analyse,
                  by = "SCIPER")
# Sans Minors.
an.vol.sm = merge(x = dt.sm,
                  y = dt.analyse,
                  by = "SCIPER")
```

... summarizing these datasets according to `Teacher.Name` :

```
# Summary.
# Avec Minors.
an.vol.am %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Arevalo         32
## 2 Buffa           39
## 3 Favi           114
## 4 Friedli         66
## 5 Kressner         7
## 6 Lachowska       91
## 7 Patakfalvi      26
## 8 Wittwer        124

# Sans Minors.
an.vol.sm %>% group_by(Teacher.Name) %>%
  summarise(Count = n())

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 8 x 2
##   Teacher.Name Count
##   <chr>        <int>
## 1 Arevalo         24
## 2 Buffa           29
## 3 Favi            98
## 4 Friedli         51
## 5 Kressner         6
## 6 Lachowska       72
## 7 Patakfalvi      23
## 8 Wittwer        100
```

3.5.5 Computing Normalized Score

In this section, we will compute the normalized score for Analyse I.

```
# Normalization.
# Avec Minors.
an.vol.am$Nor.Score = (an.vol.am$Total.Score - mean(an.non.vol.am$Total.Score)) /
  sd(an.non.vol.am$Total.Score)

# Sans Minors.
an.vol.sm$Nor.Score = (an.vol.sm$Total.Score - mean(an.non.vol.sm$Total.Score)) /
  sd(an.non.vol.sm$Total.Score)
```

4 Linear Algebra : Visualization and Analysis

In this section, we will study the differences in `Nor.Score` across `Condition`, `Gender`, etc.

4.1 Keeping Only New and Swiss/French Students

We will keep only the students who are New and Swiss + French :

```
dt = y3.vol.sm
# Keeping only NEW students.
dt = dt %>% filter(Code.BA == "New")

# Keeping only Swiss and French Students.
dt = dt %>% filter(!(Category == "Etranger.Autres"))
```

This is the data that we will use for further analysis in this section.

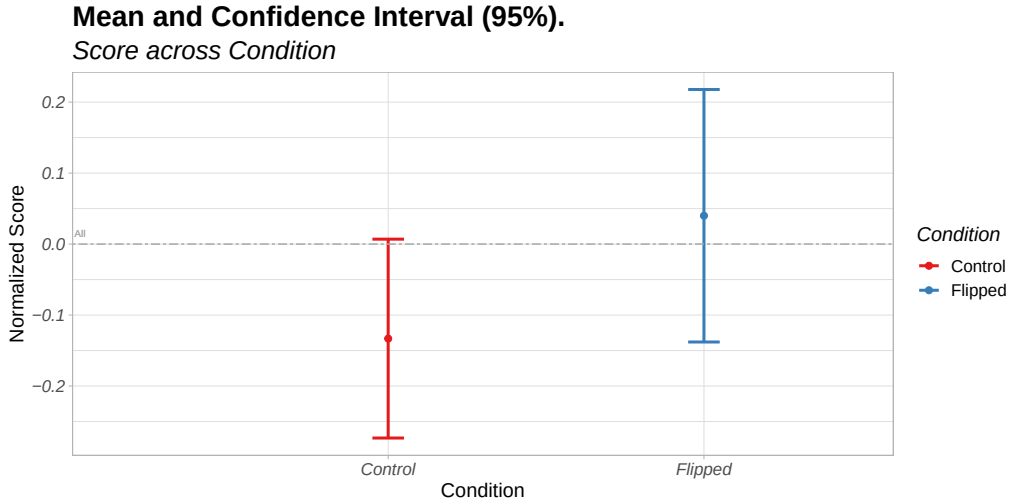
4.2 Differences Across Condition

Summarizing the data :

```
# Summary Across Condition.
dt %>% group_by(Condition) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.DI),
            SD = sd(Nor.Score.DI))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 4
##   Condition     N    Mean    SD
##   <chr>     <int>   <dbl> <dbl>
## 1 Control     178 -0.133  0.954
## 2 Flipped      91  0.0399 0.866
```

We observe that the score is slightly lower in the Control Condition.
... visualizing the differences across `Condition` :

```
# Defining the mean score.
y3.mean = mean(dt$Nor.Score.DI)
```



... now, we do an ANOVA to examine the difference in the `Nor.Score` across `Condition`.

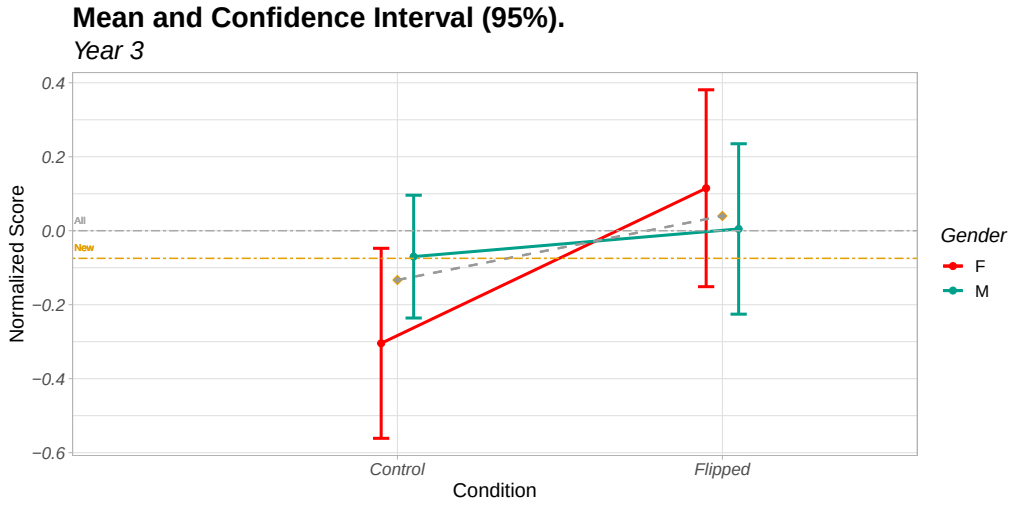
```
# Difference in Normalized Score scross Condition.
oneway.test(dt$Nor.Score.DI~dt$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Nor.Score.DI and dt$Condition
## F = 2.2448, num df = 1.00, denom df = 197.62, p-value = 0.1357
```

4.3 Difference Across Condition and Gender

... summarizing the data.

```
# Summary.
dt %>% group_by(Condition, Gender) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.DI),
            SD = sd(Nor.Score.DI))

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 4 x 5
## # Groups:   Condition [2]
##   Condition Gender     N      Mean    SD
##   <chr>     <chr>  <int>     <dbl> <dbl>
## 1 Control   F         48 -0.304   0.908
## 2 Control   M        130 -0.0700  0.966
## 3 Flipped   F         29  0.115   0.731
## 4 Flipped   M         62  0.00485 0.925
```

... visualizing the data.



... ANOVA to examine the difference in `Nor.Score` across `Gender`.

```
# Statistics.
oneway.test(dt$Nor.Score.DI~dt$Gender)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Nor.Score.DI and dt$Gender
## F = 0.70161, num df = 1.00, denom df = 153.19, p-value = 0.4035
```

... examining the difference only for Female students – `Nor.Score` across the `Conditions`.

```
# Difference in female students across condition
t.stat = dt %>% filter(Gender == "F")
oneway.test(t.stat$Nor.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score.DI and t.stat$Condition
## F = 4.9353, num df = 1.000, denom df = 68.863, p-value = 0.0296

rm(t.stat)
```

... examining the difference only for Male students – `Nor.Score` across the `Conditions`.

```r
# Difference in male students across condition
t.stat = dt %>% filter(Gender == "M")
oneway.test(t.stat$Nor.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score.DI and t.stat$Condition
## F = 0.26663, num df = 1.00, denom df = 124.87, p-value = 0.6065

rm(t.stat)
```

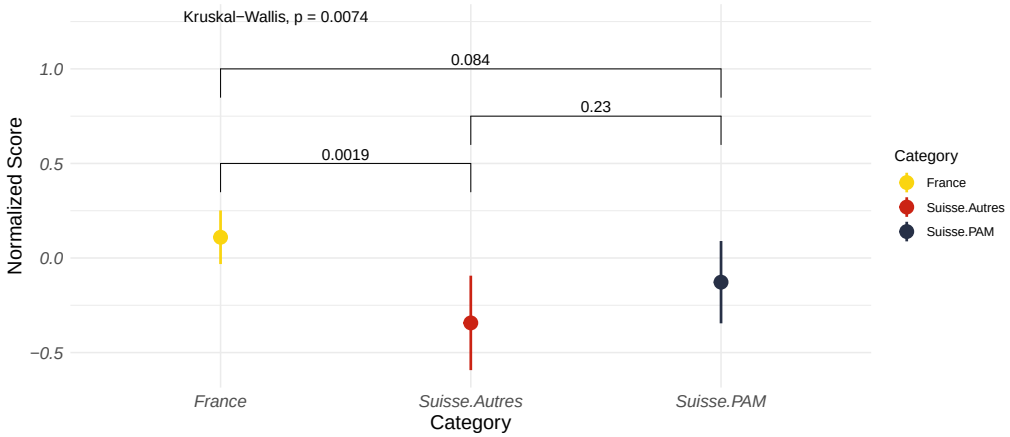
4.4 Difference Across Condition and Category

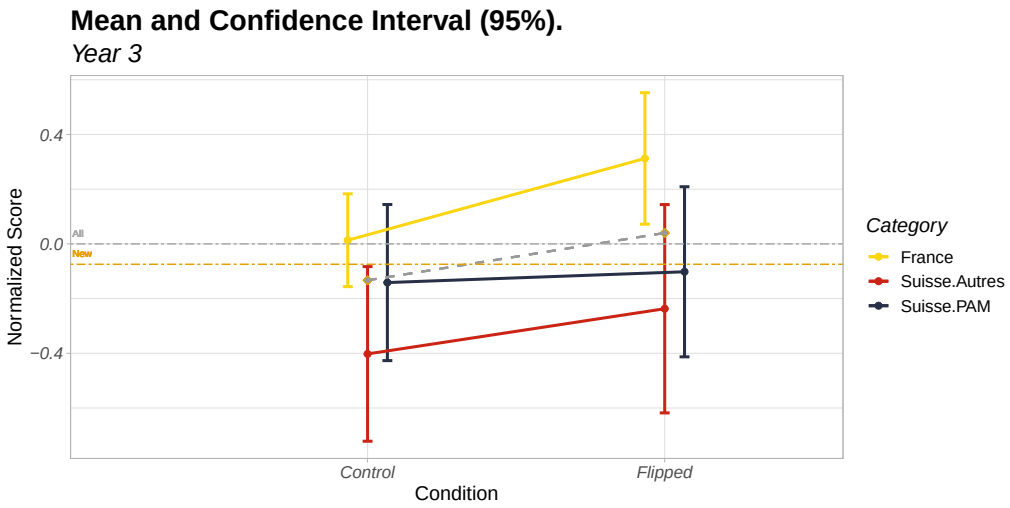
... summarizing the data.

```r
# Summary.
dt %>% group_by(Condition, Category) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.DI),
            SD = sd(Nor.Score.DI))

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 6 x 5
## # Groups:   Condition [2]
##   Condition Category          N    Mean    SD
##   <chr>     <chr>         <int>   <dbl> <dbl>
## 1 Control   France           82  0.0133 0.784
## 2 Control   Suisse.Autres    43 -0.402  1.07
## 3 Control   Suisse.PAM       53 -0.141  1.06
## 4 Flipped   France           39  0.312  0.766
## 5 Flipped   Suisse.Autres    24 -0.237  0.952
## 6 Flipped   Suisse.PAM       28 -0.102  0.839
```

... visualizing the data.

Mean and Confidence Interval (95%).
Year 3 (2019−20).





... and the ANOVA to examine the difference in `Nor.Score` across students' `Category` (first plot in this section).

```
# Difference in Normalized Score across student categories.
oneway.test(dt$Nor.Score.DI~dt$Category)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Nor.Score.DI and dt$Category
## F = 5.4125, num df = 2.00, denom df = 142.49, p-value = 0.005425
```

... examining the difference only for French students – `Nor.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "France")
oneway.test(t.stat$Nor.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score.DI and t.stat$Condition
## F = 3.9679, num df = 1.000, denom df = 76.385, p-value = 0.04995

rm(t.stat)
```

... examining the difference only for Swiss.PAM students – `Nor.Score` across `Condition` :

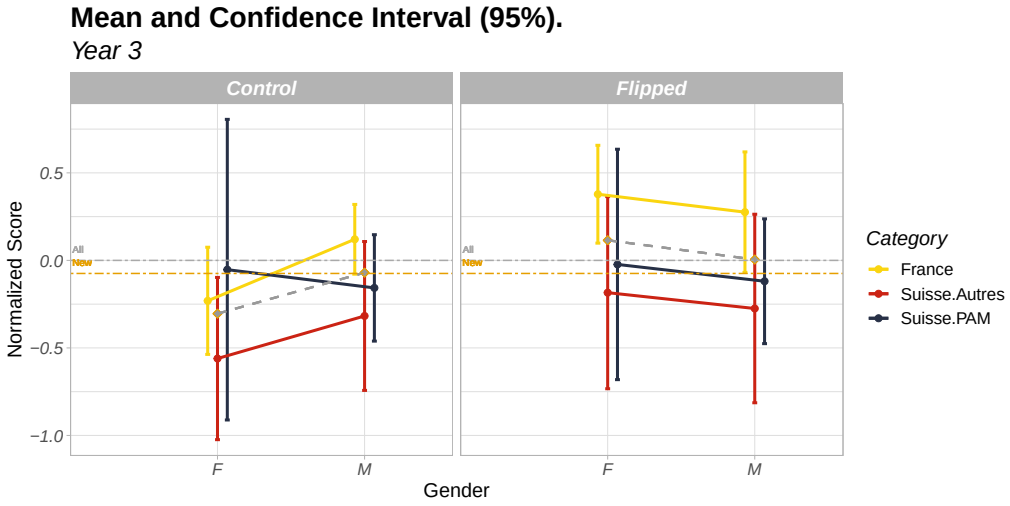
```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "Suisse.PAM")
oneway.test(t.stat$Nor.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score.DI and t.stat$Condition
## F = 0.03349, num df = 1.000, denom df = 66.976, p-value = 0.8553

rm(t.stat)
```

... examining the difference only for Swiss.Other students – `Nor.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "Suisse.Autres")
oneway.test(t.stat$Nor.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
```

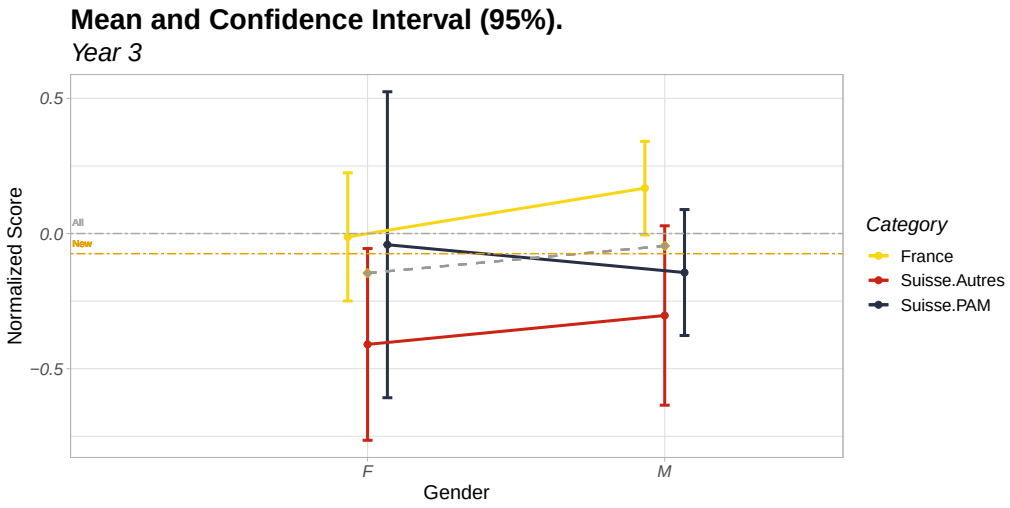
```
##
## data:  t.stat$Nor.Score.DI and t.stat$Condition
## F = 0.42449, num df = 1.00, denom df = 52.48, p-value = 0.5175

rm(t.stat)
```

... visualizing the scores with `Gender` as a third category.



4.5 Difference Across Gender and Category

... visualizing the `Nor.Score` across `Gender` and `Category` as a categorical variable.



5 Linear Algebra : Visualization and Analysis (Actual Score)

In this section, we will study the differences in `Total.Score` across `Condition`, `Gender`, etc.

5.1 Keeping Only New and Swiss/French Students

We will keep only the students who are New and Swiss + French :

```
dt = y3.vol.sm
# Keeping only NEW students.
dt = dt %>% filter(Code.BA == "New")

# Keeping only Swiss and French Students.
dt = dt %>% filter(!(Category == "Etranger.Autres"))
```

This is the data that we will use for further analysis in this section.

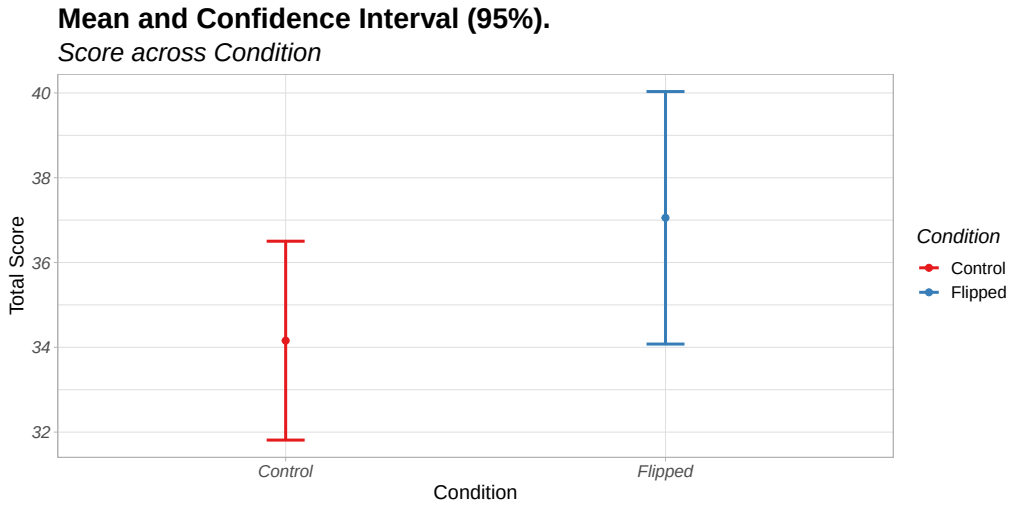
5.2 Differences Across Condition

Summarizing the data :

```
# Summary Across Condition.
dt %>% group_by(Condition) %>%
  summarise(N = n(),
            Mean = mean(Total.Score.DI),
            SD = sd(Total.Score.DI))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 4
##   Condition     N  Mean    SD
##   <chr>     <int> <dbl> <dbl>
## 1 Control     178  34.2  16.0
## 2 Flipped      91  37.1  14.5
```

We observe that the score is slightly lower in the Control Condition.
... visualizing the differences across `Condition` :

```
# Defining the mean score.
y3.mean = mean(dt$Total.Score.DI)
```



... now, we do an ANOVA to examine the difference in the `Total.Score.DI` across `Condition`.

```
# Difference in Normalized Score scross Condition.
oneway.test(dt$Total.Score.DI~dt$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Total.Score.DI and dt$Condition
## F = 2.2448, num df = 1.00, denom df = 197.62, p-value = 0.1357
```

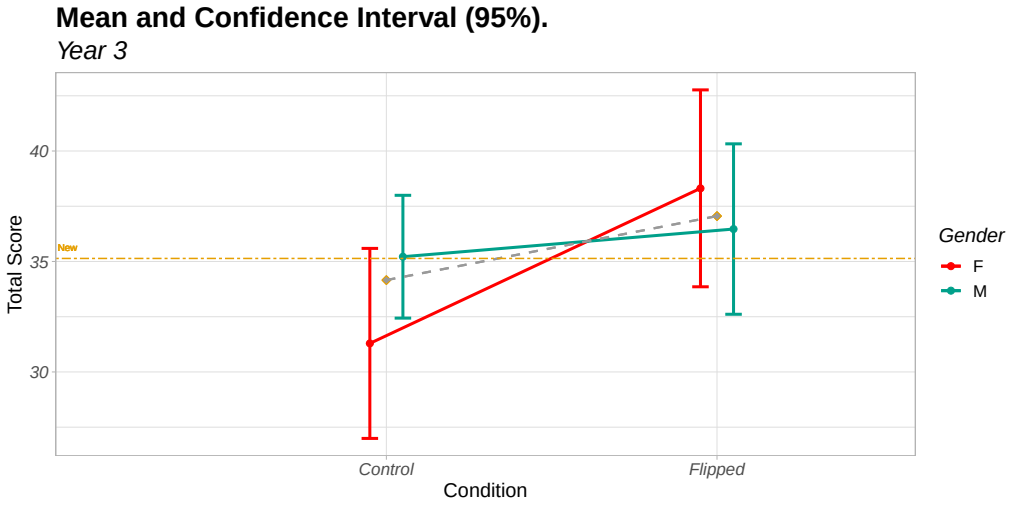
5.3 Difference Across Condition and Gender

... summarizing the data.

```
# Summary.
dt %>% group_by(Condition, Gender) %>%
  summarise(N = n(),
            Mean = mean(Total.Score.DI),
            SD = sd(Total.Score.DI))

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)
```

```
## # A tibble: 4 x 5
## # Groups:   Condition [2]
##   Condition Gender     N  Mean    SD
##   <chr>     <chr>  <int> <dbl> <dbl>
## 1 Control   F         48  31.3  15.2
## 2 Control   M        130  35.2  16.2
## 3 Flipped   F         29  38.3  12.2
## 4 Flipped   M         62  36.5  15.5
```

... visualizing the data.



... ANOVA to examine the difference in `Total.Score` across `Gender`.

```
# Statistics.
oneway.test(dt$Total.Score.DI~dt$Gender)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Total.Score.DI and dt$Gender
## F = 0.70161, num df = 1.00, denom df = 153.19, p-value = 0.4035
```

... examining the difference only for Female students – `Nor.Score` across the `Conditions`.

```
# Difference in female students across condition
t.stat = dt %>% filter(Gender == "F")
oneway.test(t.stat$Total.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Total.Score.DI and t.stat$Condition
## F = 4.9353, num df = 1.000, denom df = 68.863, p-value = 0.0296

rm(t.stat)
```

... examining the difference only for Male students – `Nor.Score` across the `Conditions`.

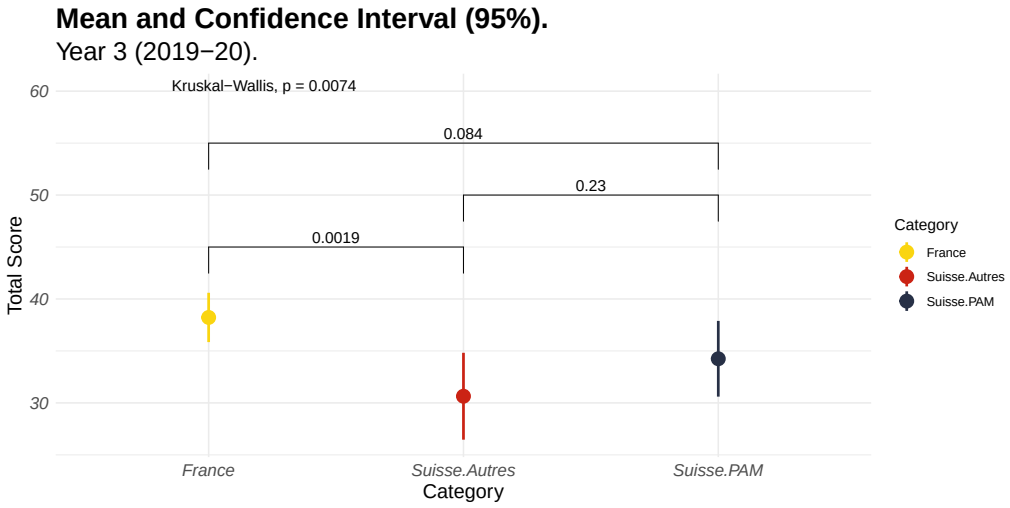
```
# Difference in male students across condition
t.stat = dt %>% filter(Gender == "M")
oneway.test(t.stat$Total.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Total.Score.DI and t.stat$Condition
## F = 0.26663, num df = 1.00, denom df = 124.87, p-value = 0.6065

rm(t.stat)
```

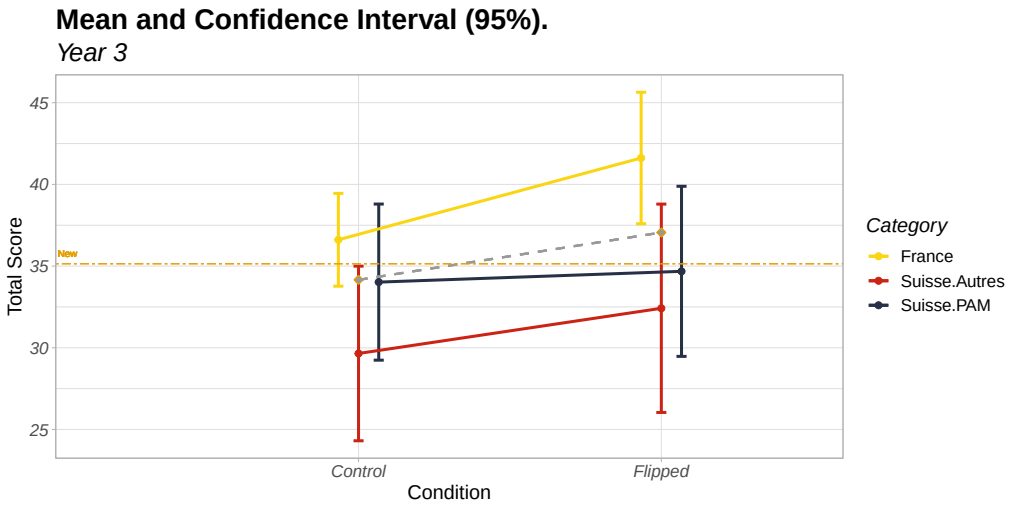
5.4 Difference Across Condition and Category

... summarizing the data.

```
# Summary.
dt %>% group_by(Condition, Category) %>%
  summarise(N = n(),
            Mean = mean(Total.Score.DI),
            SD = sd(Total.Score.DI))

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 6 x 5
## # Groups:   Condition [2]
##   Condition Category          N  Mean    SD
##   <chr>     <chr>         <int> <dbl> <dbl>
## 1 Control   France           82  36.6  13.1
## 2 Control   Suisse.Autres    43  29.7  17.9
## 3 Control   Suisse.PAM       53  34.0  17.7
## 4 Flipped   France           39  41.6  12.8
## 5 Flipped   Suisse.Autres    24  32.4  15.9
## 6 Flipped   Suisse.PAM       28  34.7  14.1
```

... visualizing the data.





... and the ANOVA to examine the difference in `Total.Score` across students' `Category` (first plot in this section).

```
# Difference in Normalized Score across student categories.
oneway.test(dt$Total.Score.DI~dt$Category)

##
##  One-way analysis of means (not assuming equal variances)
```

```
##
## data:  dt$Total.Score.DI and dt$Category
## F = 5.4125, num df = 2.00, denom df = 142.49, p-value = 0.005425
```

... examining the difference only for French students – `Total.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "France")
oneway.test(t.stat$Total.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Total.Score.DI and t.stat$Condition
## F = 3.9679, num df = 1.000, denom df = 76.385, p-value = 0.04995

rm(t.stat)
```

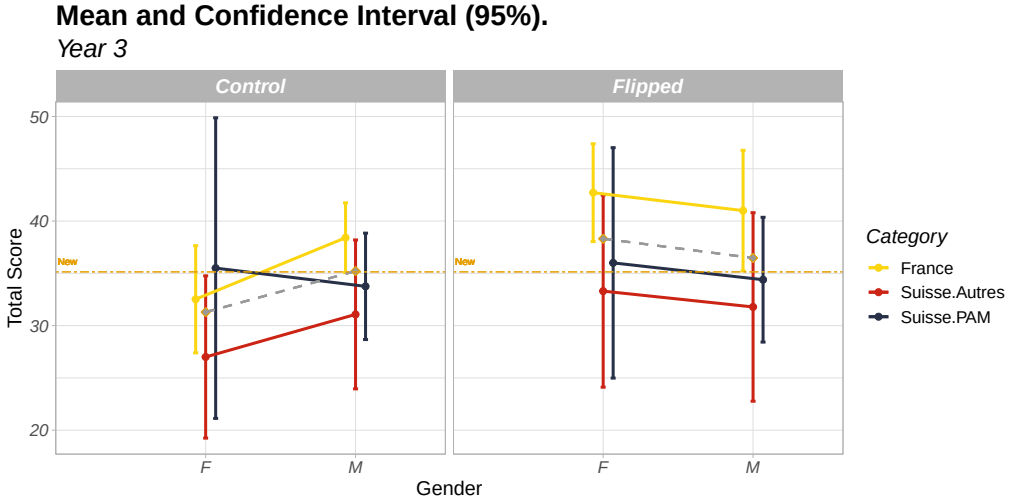
... examining the difference only for Swiss.PAM students – `Total.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "Suisse.PAM")
oneway.test(t.stat$Total.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Total.Score.DI and t.stat$Condition
## F = 0.03349, num df = 1.000, denom df = 66.976, p-value = 0.8553

rm(t.stat)
```

... examining the difference only for Swiss.Other students – `Total.Score` across `Condition` :

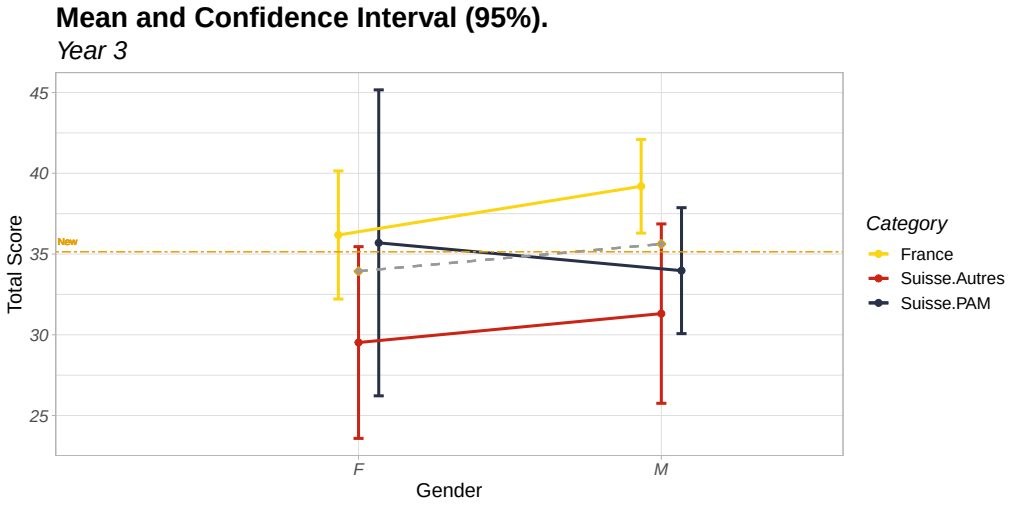
```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "Suisse.Autres")
oneway.test(t.stat$Total.Score.DI~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Total.Score.DI and t.stat$Condition
## F = 0.42449, num df = 1.00, denom df = 52.48, p-value = 0.5175

rm(t.stat)
```

... visualizing the scores with `Gender` as a third category.



5.5 Difference Across Gender and Category

... visualizing the `Total.Score` across `Gender` and `Category` as a categorical variable.



6 Analysis I : Visualization and Analysis

In this section, we will study the differences in `Nor.Score` across `Condition`, `Gender`, etc.

6.1 Keeping Only New and Swiss/French Students

We will keep only the students who are New and Swiss + French :

```
dt = an.vol.sm
# Keeping only NEW students.
dt = dt %>% filter(Code.BA == "New")

# Keeping only Swiss and French Students.
dt = dt %>% filter(!(Category == "Etranger.Autres"))
```

This is the data that we will use for further analysis in this section.

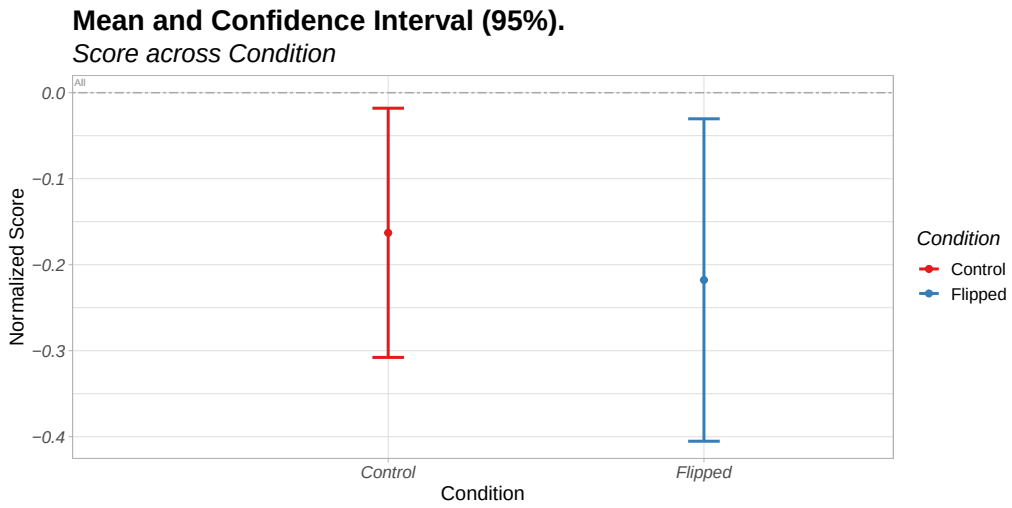
6.2 Differences Across Condition

Summarizing the data :

```
# Summary Across Condition.
dt %>% group_by(Condition) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score),
            SD = sd(Nor.Score))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 4
##   Condition     N   Mean    SD
##   <chr>     <int>  <dbl> <dbl>
## 1 Control     179 -0.163 0.989
## 2 Flipped     102 -0.218 0.966
```

We observe that the score is slightly lower in the Flipped Condition.
... visualizing the differences across `Condition` :

```
# Defining the mean score.
y3.mean = mean(dt$Nor.Score)
```



... now, we do an ANOVA to examine the difference in the `Nor.Score` across `Condition`.

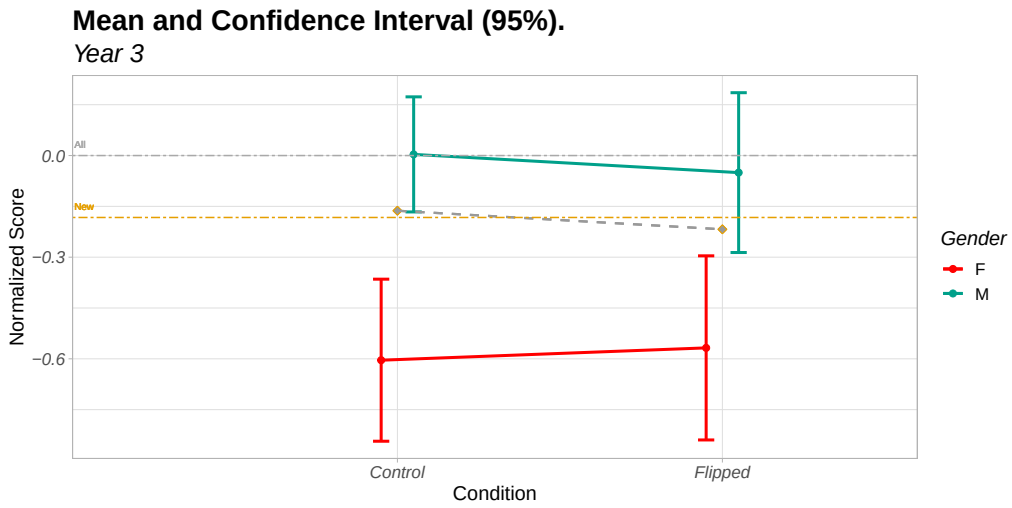
```
# Difference in Normalized Score scross Condition.
oneway.test(dt$Nor.Score~dt$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Nor.Score and dt$Condition
## F = 0.20612, num df = 1.00, denom df = 214.31, p-value = 0.6503
```

6.3 Difference Across Condition and Gender

... summarizing the data.

```
# Summary.
dt %>% group_by(Condition, Gender) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score),
            SD = sd(Nor.Score))

## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 4 x 5
## # Groups:   Condition [2]
##   Condition Gender     N     Mean    SD
##   <chr>     <chr>  <int>    <dbl> <dbl>
## 1 Control   F         49 -0.604   0.854
## 2 Control   M        130  0.00339 0.988
## 3 Flipped   F         33 -0.568   0.796
## 4 Flipped   M         69 -0.0504  1.00
```

... visualizing the data.



... ANOVA to examine the difference in `Nor.Score` across `Gender`.

```
# Statistics.
oneway.test(dt$Nor.Score~dt$Gender)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Nor.Score and dt$Gender
## F = 24.874, num df = 1.00, denom df = 179.39, p-value = 1.437e-06
```

... examining the difference only for Female students – `Nor.Score` across the `Conditions`.

```
# Difference in female students across condition
t.stat = dt %>% filter(Gender == "F")
oneway.test(t.stat$Nor.Score~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score and t.stat$Condition
## F = 0.038733, num df = 1.000, denom df = 71.985, p-value = 0.8445

rm(t.stat)
```

... examining the difference only for Male students – `Nor.Score` across the `Conditions`.

```
# Difference in male students across condition
t.stat = dt %>% filter(Gender == "M")
oneway.test(t.stat$Nor.Score~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score and t.stat$Condition
## F = 0.13159, num df = 1.00, denom df = 137.33, p-value = 0.7173

rm(t.stat)
```

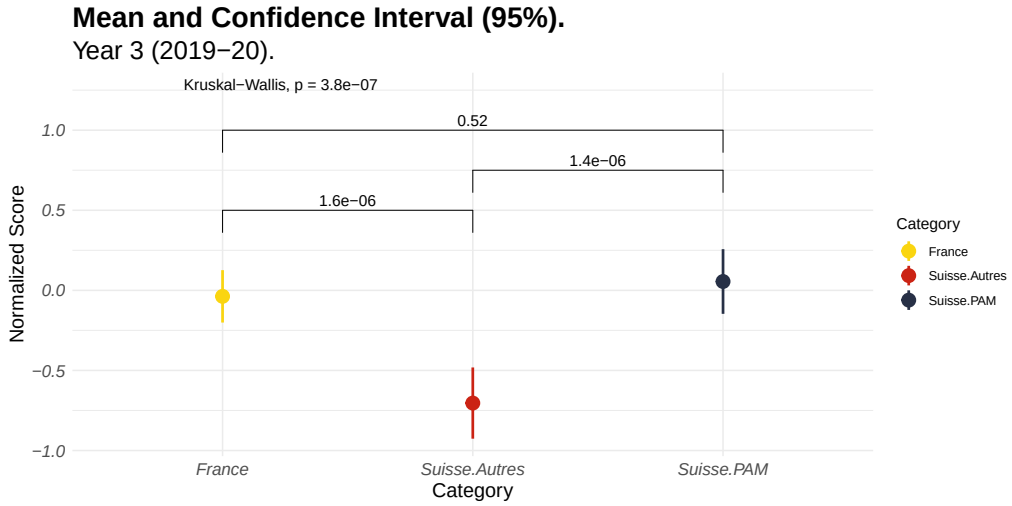
6.4 Difference Across Condition and Category

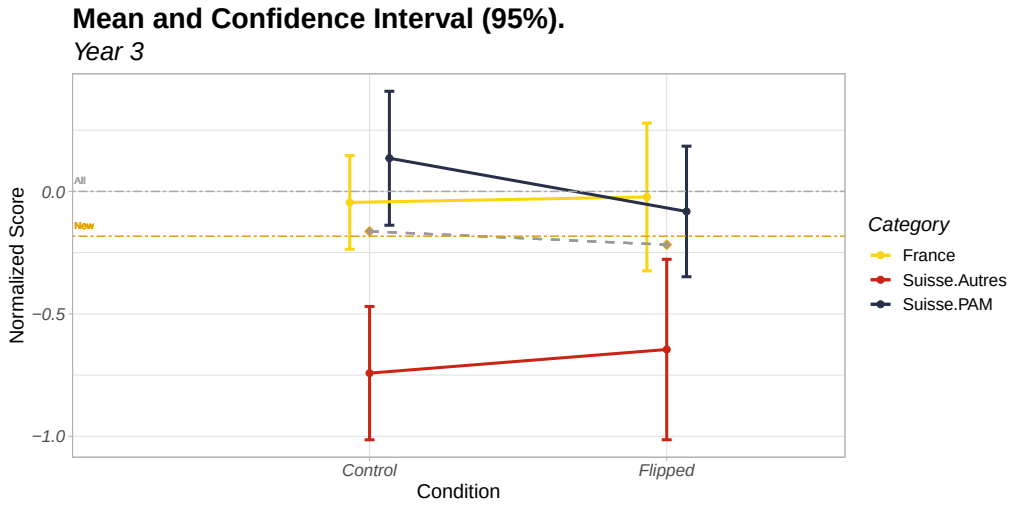
... summarizing the data.

```
# Summary.
dt %>% group_by(Condition, Category) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score),
            SD = sd(Nor.Score))
```

```
## `summarise()` regrouping output by 'Condition' (override with `.groups` argument)

## # A tibble: 6 x 5
## # Groups:   Condition [2]
##   Condition Category          N    Mean    SD
##   <chr>     <chr>         <int>   <dbl> <dbl>
## 1 Control   France           82 -0.0450 0.884
## 2 Control   Suisse.Autres    44 -0.742  0.921
## 3 Control   Suisse.PAM       53  0.135  1.02
## 4 Flipped   France           42 -0.0227 0.998
## 5 Flipped   Suisse.Autres    29 -0.646  1.01
## 6 Flipped   Suisse.PAM       31 -0.0819 0.757
```

... visualizing the data.





... and the ANOVA to examine the difference in `Nor.Score` across students' `Category` (first plot in this section).

```
# Difference in Normalized Score across student categories.
oneway.test(dt$Nor.Score~dt$Category)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  dt$Nor.Score and dt$Category
## F = 15.141, num df = 2.00, denom df = 164.61, p-value = 9.202e-07
```

... examining the difference only for French students – `Nor.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "France")
oneway.test(t.stat$Nor.Score~t.stat$Condition)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score and t.stat$Condition
## F = 0.014998, num df = 1.000, denom df = 74.471, p-value = 0.9029

rm(t.stat)
```

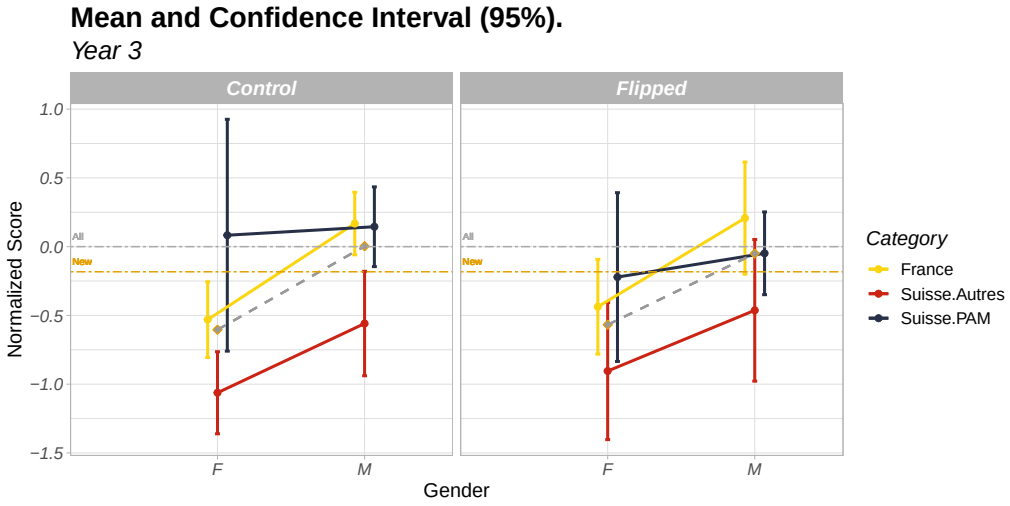
... examining the difference only for Swiss.PAM students – `Nor.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "Suisse.PAM")
oneway.test(t.stat$Nor.Score~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score and t.stat$Condition
## F = 1.2438, num df = 1.000, denom df = 77.106, p-value = 0.2682

rm(t.stat)
```

... examining the difference only for Swiss.Other students – `Nor.Score` across `Condition` :

```
# Difference in Score across Condition for "French" students only.
t.stat = dt %>% filter(Category == "Suisse.Autres")
oneway.test(t.stat$Nor.Score~t.stat$Condition)

##
##  One-way analysis of means (not assuming equal variances)
##
## data:  t.stat$Nor.Score and t.stat$Condition
## F = 0.17024, num df = 1.000, denom df = 56.045, p-value = 0.6815

rm(t.stat)
```

... visualizing the scores with `Gender` as a third category.



6.5 Difference Across Gender and Category

... visualizing the `Nor.Score` across `Gender` and `Category` as a categorical variable.

Mean and Confidence Interval (95%).

Year 3