LEARN Center
EPFL

Analysis of Prior Level in Reinforced Mathematics

Data for Only "New" Students (Year 1 & Year 2)

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 R 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(RMariaDB)
library(corrplot)
library(wesanderson)
library(psych)
library(ggpubr)
```

3 Functions for Anonymization

Below, we have added the code to *anonymize* and *deanonymizing* the SCIPER values. Please note, that the code has been masked to appear in the compiled PDF. Still, the functions are called : `GenerateUniqueID(x)` and `GenerateSCIPER(x)`.

4 Data Import

First step would be to import the data. The data has been prepared with the help of student assistants, who examined the high-school transcripts of volunteers and entered their scores in french, mathematics, physics, and option specifique.

This data is initially in MS Excel (.xlsx) format, which must be imported and re-formatted before we can conduct the analysis.

```
# Importing the data.
# Setting the path.
path = paste(here(), "/Data/Prior-Level-Maths/", sep = "")

# Pre Data for all Volunteers
dt.pre = read_excel(paste(path, "High-School-Data.xlsx", sep = ""), sheet = "Saisie")

# Cleaning up
rm(path)
```

As a *second* step, we will change the column names so that it is easier to navigate within the data.

```
names(dt.pre) = c("Serial.Year.SCIPER", "SCIPER", "Last.Name",
                  "First.Name", "Gender", "Country.Diploma",
                  "Continent.Diploma", "Non.State.Certificate", "Canton.School",
                  "Institution", "Year.Diploma", "Title.Diploma",
                  "Reinforced.Maths", "Avg.French.Bac", "Coder",
                  "Language.Instruction", "Label.French", "Rating.French",
```

```
                  "Scale.French", "Label.Maths", "Rating.Maths",
                  "Scale.Maths", "Label.Physics", "Rating.Physics",
                  "Scale.Physics", "Label.OS", "Rating.OS",
                  "Scale.OS")
```

The first column (`Serial.Year.SCIPER`) is a concatenation of the **Serial #, Course.Year, SCIPER**. We will, next, tokenize the string to just extract the `Course.Year` information.

```
# Defining a new variable called the Enroll.Year :
# The Academic Year into which the student was enrolled.

dt.pre$Enroll.Year = lapply(dt.pre$Serial.Year.SCIPER,
                            function(x) {
                              unlist(strsplit(
                                unlist(strsplit(x, ","))[2],
                                "\""))[2]})
dt.pre$Enroll.Year = unlist(dt.pre$Enroll.Year)
```

As the *third* step, we will remove the columns corresponding to the `First.Name` and `Last.Name`. We don't need them for the further analysis.

```
# Remove First and Last Name Columns.

dt.pre$First.Name = NULL
dt.pre$Last.Name = NULL
```

As a next step, we will do an overview of the different variables. Simultaneously, we will also fine-tune the variables for consistency.

4.1 Overview of Variables

4.1.1 SCIPER

There will be unique SCIPER values for the different students. However, it is important to check if there are <**NA**> or **Duplicate** (where a person is enrolled for both the course years) values.

Checking for NA's :

```
# Checking if there are NA's in SCIPER
nrow(dt.pre %>% filter(is.na(SCIPER)))

## [1] 0
```

... we see that there are no NA's, which is a good thing. Next, we will see for duplicates.

```
# Checking if SCIPERs are duplicated.
t.rep = dt.pre %>% group_by(SCIPER) %>%
  summarise(Count = n()) %>%
  filter(Count == 2)

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

t.rep$SCIPER

##  [1] "270605" "282145" "282752" "282966" "283111" "283613" "283796" "286898" "287325"
## [10] "287386" "287757" "287781" "287786" "287957" "288078" "288206" "288275" "288477"
## [19] "289583"
```

... we see that there are 19 individuals who have repeated, and so we will remove them from our analysis.

```
# Removing the SCIPERs who attended both the courses.
dt.pre = dt.pre %>% filter(!(dt.pre$SCIPER %in% t.rep$SCIPER))

# Cleaning up
rm(t.rep)
```

4.1.2 Gender

```
# Summary of the variable.
dt.pre %>% group_by(Enroll.Year, Gender) %>%
  summarise(Count = n())

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

## # A tibble: 5 x 3
## # Groups:   Enroll.Year [2]
##   Enroll.Year Gender Count
##   <chr>       <chr>  <int>
## 1 Y1-2017-18  F        175
## 2 Y1-2017-18  M        324
## 3 Y1-2017-18  N/A        1
## 4 Y2-2018-19  F        118
## 5 Y2-2018-19  M        235
```

We see that there is 1 student whose Gender is "N/A". So, we will remove this person.

```
# Filtering out the person whose Gender is unavailable.
dt.pre = dt.pre %>% filter(!(Gender == "N/A"))
```

4.1.3 Country.Diploma & Continent.Diploma

```
# Summary of the variables.
t.loc = dt.pre %>% group_by(Enroll.Year, Continent.Diploma, Country.Diploma) %>%
  summarise(Count = n()) %>%
  arrange(Enroll.Year, Continent.Diploma) %>%
  spread(Continent.Diploma, Count)

## `summarise()` regrouping output by 'Enroll.Year', 'Continent.Diploma' (override with `.groups` argument

t.loc

## # A tibble: 27 x 6
## # Groups:   Enroll.Year [2]
##    Enroll.Year Country.Diploma Afrique Amérique Europe International
##    <chr>       <chr>             <int>    <int>  <int>         <int>
##  1 Y1-2017-18  Apatrides            NA       NA     15             2
##  2 Y1-2017-18  Belgique             NA       NA      5            NA
##  3 Y1-2017-18  Bulgarie             NA       NA      1            NA
##  4 Y1-2017-18  Canada               NA        3     NA            NA
##  5 Y1-2017-18  Chypre               NA       NA      1            NA
##  6 Y1-2017-18  El Salvador          NA        1     NA            NA
##  7 Y1-2017-18  Espagne              NA       NA      3            NA
##  8 Y1-2017-18  Etats-Unis           NA        4     NA            NA
##  9 Y1-2017-18  France               NA       NA    247            NA
## 10 Y1-2017-18  Italie               NA       NA      4            NA
## # ... with 17 more rows
```

... since we are only interested in the Swiss and French students, we will filter them out for further analysis.

```
# Filter Swiss and French Student
dt.stat = dt.pre %>% filter(Country.Diploma %in% c("France", "Suisse"))

# Cleaning up
rm(t.loc)
```

4.1.4 Non.State.Certificate

```
# Summary of the variable.
dt.pre %>% group_by(Non.State.Certificate) %>%
  summarise(Count = n())

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

## # A tibble: 3 x 2
##   Non.State.Certificate           Count
##   <chr>                           <int>
## 1 European Baccalaureate (EB)        16
## 2 International Baccalaureate (IB)    4
## 3 <NA>                              832
```

... for the Swiss & French student, this value will be <**NA**>.

4.1.5 Canton.School, Institution, Year.Diploma, Title.Diploma, & Language.Instruction

... `Canton.School` is only valid for Swiss schools.

```
# Canton.School
dt.pre %>% group_by(Enroll.Year, Canton.School) %>%
  summarise(Count = n()) %>%
  spread(Canton.School, Count)

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

## # A tibble: 2 x 18
## # Groups:   Enroll.Year [2]
##   Enroll.Year    BE    BL    BS    FR    GE    JU    LU    NE    SG    SO    SZ    TI
##   <chr>       <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 Y1-2017-18     11    NA    NA    17    44     3     1     6     3    NA     1     7
## 2 Y2-2018-19     10     1     1    10    29     5    NA     8     1     2    NA     4
## # ... with 5 more variables: VD <int>, `VD/ FR` <int>, VS <int>, ZH <int>, `<NA>` <int>
```

... we see that there are <**NA**> values, which to my understanding correspond to the "*missing values*". On the other hand, the presence of **N/A** values corresponds to the fact that the value is not applicable, i.e. for French students.

Next, we will simply provide a summary of the `Institution` (high-school), `Year.Diploma` (gratuation year from high-school), `Title.Diploma` (Bacc. or Matu), and `Language.Instruction` (the language of teaching at high-school).

```
# Institution
levels(as.factor(dt.pre$Institution))

##  [1] "CANT. FRIBOURG"
##  [2] "CANT. VALAIS"
##  [3] "CANT. VAUD"
##  [4] "Centre d'Enseignement Professionnel Technique et Artisanal CEPTA"
##  [5] "Collège Calvin, Genève"
##  [6] "Collège Claparèdes, Conches"
##  [7] "Collège de Candolle, Genève"
##  [8] "Collège de Gambach, Fribourg"
##  [9] "Collège de l'Abbaye, Saint-Maurice"
## [10] "Collège de Saussure, Petit-Lancy"
## [11] "Collège du Sud, Bulle"
## [12] "Collège et Ecole de Commerce André-Chavanne, Genève"
## [13] "Collège et Ecole de Commerce Emilie-Gourd, Genève"
## [14] "Collège Madame de Stäel, Carouge Genève"
## [15] "Collège pour adultes Alice-Rivaz, Genève"
## [16] "Collège Rousseau, Genève"
## [17] "Collège Sismondi, Genève"
## [18] "Collège St-Michel / Kollegium St.Michael, Fribourg"
## [19] "Collège Ste-Croix / Kollegium Heiligkreuz, Fribourg"
## [20] "Collège Voltaire, Genève"
## [21] "Collegio Pio XII - Liceo Diocesano, Breganzona"
## [22] "Columbia University, New York"
```

```
## [23] "École Ardévaz, Sion, Monthey"
## [24] "École des Arches, Lausanne"
## [25] "Ecole inconnue ou étrangère"
## [26] "Ecole Lemania, Lausanne"
## [27] "Ecole Moser, Genève"
## [28] "Ecole Moser, Nyon"
## [29] "Ecole professionnelle technique des métiers, Sion"
## [30] "Ecole Technique et des Métiers, Lausanne"
## [31] "EPSU (Ecole de préparation et soutien universitaire), Genève"
## [32] "Freies Gymnasium Bern"
## [33] "Gymnase Auguste-Piccard, Lausanne"
## [34] "Gymnase d'Yverdon, Cheseaux-Noréaz"
## [35] "Gymnase de Beaulieu, Lausanne"
## [36] "Gymnase de Burier, La Tour-de-Peilz"
## [37] "Gymnase de Chamblandes, Pully"
## [38] "Gymnase de la Cité, Lausanne"
## [39] "Gymnase de Morges, Morges"
## [40] "Gymnase de Nyon, Nyon"
## [41] "Gymnase de Renens, Lausanne"
## [42] "Gymnase du Bugnon, Lausanne"
## [43] "Gymnase du Soir, Lausanne"
## [44] "Gymnase Français, Bienne"
## [45] "Gymnase Intercantonal de la Broye, Payerne"
## [46] "Gymnase Provence, Lausanne"
## [47] "Gymnasium Biel-Seeland, Biel/Bienne"
## [48] "Gymnasium Kirschgarten, Basel"
## [49] "Gymnasium Lebermatt"
## [50] "Gymnasium Neufeld, Bern"
## [51] "Gymnasium Oberwil"
## [52] "Gymnasium Thun"
## [53] "Haute Ecole de gestion HEG Genève"
## [54] "Haute Ecole de musique HEMU de Lausanne"
## [55] "INSTITUT FLORIMONT, PETIT-LANCY"
## [56] "Kantonale Maturitätsschule für Erwachsene (KME), Zürich"
## [57] "Kantonsschule Alpenquai Luzern"
## [58] "Kantonsschule am Burggraben St.Gallen"
## [59] "Kantonsschule Sargans"
## [60] "Kantonsschule Solothurn"
## [61] "Kantonsschule Wattwil"
## [62] "Kollegium Spiritus Sanctus, Brig"
## [63] "Liceo cantonale, Locarno"
## [64] "Liceo cantonale, Mendrisio"
## [65] "Liceo di Lugano 1, Lugano"
## [66] "Liceo di Lugano 2, Savosa"
## [67] "Lycée Blaise-Cendrars, La Chaux-de-Fonds"
## [68] "Lycée cantonal, Porrentruy"
## [69] "Lycée Denis-de-Rougemont, Neuchâtel"
## [70] "Lycée-Collège de la Planta, Sion"
## [71] "Lycée-Collège des Creusets, Sion"
## [72] "Schweizerschule, Bogota"
## [73] "Stiftsschule Einsiedeln"
## [74] "Université de Genève"
```

```
# Year.Diploma
dt.pre %>% group_by(Year.Diploma) %>%
  summarise(Count = n())

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

## # A tibble: 9 x 2
##   Year.Diploma Count
##   <chr>        <int>
```

```
## 1 2005             1
## 2 2011             1
## 3 2012             3
## 4 2013             3
## 5 2014             6
## 6 2015            40
## 7 2016           168
## 8 2017           374
## 9 2018           256
```

```
# Title.Diploma
levels(as.factor(dt.pre$Title.Diploma))
```

```
##  [1] "Autre certificat suisse"
##  [2] "Bacc. étranger"
##  [3] "Bachelor étranger"
##  [4] "Diplôme HES"
##  [5] "Mat. Commission suisse de maturité"
##  [6] "Mat. reconnue opt. arts visuels"
##  [7] "Mat. reconnue opt. biologie et chimie"
##  [8] "Mat. reconnue opt. économie et droit"
##  [9] "Mat. reconnue opt. langue moderne"
## [10] "Mat. reconnue opt. langues anciennes"
## [11] "Mat. reconnue opt. musique"
## [12] "Mat. reconnue opt. philo / psycho / pédagogie"
## [13] "Mat. reconnue opt. physique et math"
## [14] "Maturité professionnelle"
## [15] "Passerelle maturité professionnelle - hautes écoles universitaires"
```

```
# Language.Instruction
levels(as.factor(dt.pre$Language.Instruction))
```

```
##  [1] "Allemand"                      "Anglais"
##  [3] "Arabe"                         "Bilingue Allemand-Anglais"
##  [5] "Bilingue allemand-espagnol"    "Bilingue Arabe-Français"
##  [7] "Bilingue Français-Allemand"    "Bilingue Français-Anglais"
##  [9] "Bilingue Français-Neerlandais" "Bilingue Italien-Français"
## [11] "Bulgare"                       "Espagnol"
## [13] "Français"                      "Grec"
## [15] "Hongrois"                      "Italien"
## [17] "Macédonien"                    "N/A"
## [19] "Norvégien"                     "Plusieurs (Bac. Eur.)"
## [21] "Portugais"                     "Turque"
```

4.1.6 Reinforced.Maths

This variable just provides a summary if students' took reinforced mathmatics in their high-school.

```
# Reinforced.Maths
dt.pre %>% group_by(Reinforced.Maths) %>%
  summarise(Count = n())
```

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

```
## # A tibble: 2 x 2
##   Reinforced.Maths Count
##   <chr>            <int>
## 1 Oui                199
## 2 <NA>               653
```

However, it is important to note that this variable in inself is not very reliable because even though their certificate does not show "Reinforced Mathematics" as a subject, they may have taken it and the subject may be written using some other terminology.

4.1.7 Avg.French.Bac

This variable applies only to the FRENCH students who come through a different system as compared to other students.

```
# Avg.French.Bac
summary(as.double(dt.pre$Avg.French.Bac))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   14.02   16.36   17.10   17.23   17.89   20.00     411
```

4.1.8 Label.French, Rating.French, & Scale.French

... `Label.French` corresponds to the name written on the high-school certificate next to the french exam they took.

```
# Label.French
levels(as.factor(dt.pre$Label.French))
```

```
##  [1] "Atelier Français"                      "C1"
##  [3] "Certificat de langue française B2"     "Certificat de scolarisation en france"
##  [5] "DALF C1"                               "DALF C2"
##  [7] "DELF B2"                               "Français"
##  [9] "Français 5 périodes"                   "Français écrit"
## [11] "French language and literature"        "Langue 1"
## [13] "Langue 2"                              "Langue maternelle : français"
## [15] "N/A"
```

... `Scale.French` is the maximum score that a student can receive for their french exam.

```
# Scale.French
levels(as.factor(dt.pre$Scale.French))
```

```
## [1] "10.0"                    "100.0"                   "140.0"
## [4] "20.0"                    "300.0"                   "6.0"
## [7] "7.0"                     "N/A"                     "Unsatisfactory-Excellent"
```

We see that different scales are used for different persons. Some are on a quantitative scale, and some follow a categoriacal scale.

... `Rating.French` is the actual score that the students' received for their french level.

```
summary(as.double(dt.pre$Rating.French))
```

```
## Warning in summary(as.double(dt.pre$Rating.French)): NAs introduced by coercion
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    3.00    4.50   10.00   10.99   14.00  264.00      37
```

4.1.9 Label.Maths, Rating.Maths, & Scale.Maths

... `Label.Maths` is the name written on the high-school certificate corresponding to maths.

```
# Label.Maths
levels(as.factor(dt.pre$Label.Maths))
```

```
##  [1] "Algèbre linéaire (pour CMS)"     "Mathématiques"
##  [3] "Mathématiques (niveau renforcé)" "Mathématiques (niveau standard)"
##  [5] "Mathématiques 2 (avancé)"        "Mathématiques 5 périodes"
##  [7] "Mathématiques 6 périodes"        "Mathématiques 6h"
##  [9] "Mathématiques fortes"            "Mathématiques niveau 1"
## [11] "Mathématiques niveau 2"          "Mathématiques niveau avancé"
## [13] "Mathématiques standard"          "Maths 2"
## [15] "N/A"                             "option mathématiques"
```

... `Scale.Maths` is the maximum score that someone receives for a maths exam.

```
# Scale.Maths
levels(as.factor(dt.pre$Scale.Maths))
```

```
## [1] "10.0"                    "100.0"                   "140.0"
## [4] "20.0"                    "360.0"                   "6.0"
## [7] "7.0"                     "N/A"                     "Unsatisfactory-Excellent"
```

... `Rating.Maths` is the actual score that the students got on their high-school maths exam.

```
# Rating.Maths
summary(as.double(dt.pre$Rating.Maths))
```

```
## Warning in summary(as.double(dt.pre$Rating.Maths)): NAs introduced by coercion
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    3.00    5.00   15.00   12.77   18.00  297.00      14
```

4.1.10 Label.Physics, Rating.Physics, & Scale.Physics

... `Label.Physics` is the name written on the high-school certificate corresponding to Physics.

```
# Label.Physics
levels(as.factor(dt.pre$Label.Physics))
```

```
##  [1] "N/A"                    "option physique"        "Physique"
##  [4] "Physique (pour CMS)"    "Physique 3"             "Physique et astronomie"
##  [7] "Physique niveau avancé" "Physique pour PAM"      "Physique-Chimie"
## [10] "Sciences physiques"
```

... `Scale.Physics` is the maximum score that someone receives for a physics exam.

```
# Scale.Physics
levels(as.factor(dt.pre$Scale.Physics))
```

```
## [1] "10.0"                    "100.0"                   "120.0"
## [4] "20.0"                    "6.0"                     "7.0"
## [7] "Echelles"                "N/A"                     "Unsatisfactory-Excellent"
```

... `Rating.Physics` is the actual score that the students got on their high-school physics exam.

```
# Rating.Physics
summary(as.double(dt.pre$Rating.Physics))
```

```
## Warning in summary(as.double(dt.pre$Rating.Physics)): NAs introduced by coercion
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    3.00    5.00   15.00   12.48   18.00  111.00      28
```

4.1.11 Label.OS, Rating.OS, & Scale.OS

.. `Label.OS` is the name written on the high-school certificate corresponding to Specific Options.

```
# Label.OS
levels(as.factor(dt.pre$Label.OS))
```

```
##  [1] "Allemand"                          "Anglais"
##  [3] "Arts visuels"                      "Biologie et chimie"
##  [5] "Economie et droit"                 "Espagnol"
##  [7] "Grec"                              "Italien"
##  [9] "Latin"                             "Musique"
## [11] "N/A"                               "Philosophie et psychologie"
## [13] "Physique et applications des maths"
```

... `Scale.OS` is the maximum score that someone receives for a Specific Option exam.

```
# Scale.OS
levels(as.factor(dt.pre$Scale.OS))

## [1] "6.0"      "Echelles" "N/A"
```

... `Rating.OS` is the actual score that the students got on their high-school OS exam.

```
# Rating.OS
summary(as.double(dt.pre$Rating.OS))

## Warning in summary(as.double(dt.pre$Rating.OS)): NAs introduced by coercion

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   3.000   4.500   4.500   4.701   5.000   6.000     526
```

In the next sections, we will separate the data for the **Swiss** students and the **French** students, followed by their progress in the flipped class.

5 Swiss High-School Students

In this section, we will create a separate dataset for **Swiss** high-school students. Next, we will assign them into categories (Low, High scorers) based on their Maths scores.

5.1 Data Pre-Processing & Overview

We will use the `dt.stat` data-frame, which is a filtered dataset after the repeated SCIPERs were removed.

```
# Filtering out the Swiss students.
dt.swiss = dt.stat %>% filter(Country.Diploma == "Suisse")
```

5.1.1 Mathematics

... *First* step would be to see if all the students have a grade for mathematics (`Rating.Maths`), if yes, what is the maximum grade (`Scale.Maths`) :

```
# Checking the validity of Rating.Maths
summary(as.double(dt.swiss$Rating.Maths))

## Warning in summary(as.double(dt.swiss$Rating.Maths)): NAs introduced by coercion

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   3.000   4.500   5.000   4.752   5.000   6.000       3

# Also, checking the validity of Scale.Maths
summary(as.double(dt.swiss$Scale.Maths))

## Warning in summary(as.double(dt.swiss$Scale.Maths)): NAs introduced by coercion

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       6       6       6       6       6       6       3
```

From the results above, we can see that, there are **3** students who have **NULL** values for their `Rating.Maths` and `Scale.Maths`. Let's have a deeper look at these students :

```
# Students with NULL Maths Scores
t.math.na = dt.swiss %>% filter(Rating.Maths == "N/A")

# Summary of their Institution and Title.Diploma
t.math.na$Institution

## [1] "Haute Ecole de musique HEMU de Lausanne" "Université de Genève"
## [3] "Haute Ecole de gestion HEG Genève"

t.math.na$Title.Diploma
```

```
## [1] "Diplôme HES"             "Autre certificat suisse" "Autre certificat suisse"

# Summary of grades in French, Maths, Physics, and OS
t.math.na$Rating.French

## [1] "N/A" "N/A" "N/A"

t.math.na$Rating.Maths

## [1] "N/A" "N/A" "N/A"

t.math.na$Rating.Physics

## [1] "N/A" "N/A" "N/A"

t.math.na$Rating.OS

## [1] "N/A" "N/A" "N/A"

# Cleaning up
rm(t.math.na)
```

We see that these students do not come from conventional high-school backgrounds and their grades for French, Maths, Physics, and OS are not available. Therefore, we will remove them in the next step.

... *Second*, we will remove the students with **N/A** grades and convert numeric grades to double values.

```
# Filtering out students with N/A grades.
dt.swiss = dt.swiss %>% filter(Rating.Maths != "N/A")

# Converting maths scores to double values.
dt.swiss$Rating.Maths = as.double(dt.swiss$Rating.Maths)
dt.swiss$Scale.Maths = as.double(dt.swiss$Scale.Maths)
```

... *Third*, we will filter out students who have taken Mathematics Renforcé :

```
# Summary of Math Labels
levels(as.factor(dt.swiss$Label.Maths))

##  [1] "Algèbre linéaire (pour CMS)"     "Mathématiques"
##  [3] "Mathématiques (niveau renforcé)" "Mathématiques (niveau standard)"
##  [5] "Mathématiques 2 (avancé)"        "Mathématiques fortes"
##  [7] "Mathématiques niveau 1"          "Mathématiques niveau 2"
##  [9] "Mathématiques niveau avancé"     "Mathématiques standard"
## [11] "Maths 2"

# Filtering out students with Reinforced Mathematics
# We will only consider students with the following labels:
# Mathématiques (niveau renforcé)   Mathématiques 2 (avancé)
# Mathématiques fortes              Mathématiques niveau 2
# Mathématiques niveau avancé       Maths 2
dt.swiss.math = dt.swiss %>% filter(Label.Maths %in% c(
  "Mathématiques (niveau renforcé)", "Mathématiques 2 (avancé)",
  "Mathématiques fortes", "Mathématiques niveau 2",
  "Mathématiques niveau avancé", "Maths 2"
))
```

... *Fourth*, we will summarise the main scores for different `Gender` and `Label.Maths` :

```
# Summary of Rating.Maths
dt.swiss.math %>% group_by(Label.Maths, Gender) %>%
  summarise(Count = n(),
            Mean = mean(Rating.Maths),
            SD = sd(Rating.Maths))

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

```
## # A tibble: 11 x 5
## # Groups:   Label.Maths [6]
##    Label.Maths                      Gender Count  Mean     SD
##    <chr>                            <chr>  <int> <dbl>  <dbl>
##  1 Mathématiques (niveau renforcé)  F         42  4.71  0.606
##  2 Mathématiques (niveau renforcé)  M         71  4.66  0.652
##  3 Mathématiques 2 (avancé)         M          3  5.17  0.289
##  4 Mathématiques fortes             F          4  4.88  0.629
##  5 Mathématiques fortes             M          7  4.71  0.699
##  6 Mathématiques niveau 2           F          2  4.75  0.354
##  7 Mathématiques niveau 2           M          6  4.83  0.753
##  8 Mathématiques niveau avancé      F          5  5     0.612
##  9 Mathématiques niveau avancé      M         22  4.77  0.572
## 10 Maths 2                          F          1  4    NA
## 11 Maths 2                          M          2  4.5   0.707
```

5.1.2 Physics

... *First* step would be to see if all the students have a grade for physics (`Rating.Physics`), if yes, what is the maximum grade (`Rating.Physics`) :

```
# Checking the validity of Rating.Physics
summary(as.double(dt.swiss$Rating.Physics))

## Warning in summary(as.double(dt.swiss$Rating.Physics)): NAs introduced by coercion

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   3.000   4.500   5.000   4.915   5.500   6.000       9

# Also, checking the validity of Scale.Physics
levels(as.factor(dt.swiss$Scale.Physics))

## [1] "6.0"      "Echelles" "N/A"

summary(as.double(dt.swiss$Scale.Physics))

## Warning in summary(as.double(dt.swiss$Scale.Physics)): NAs introduced by coercion

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       6       6       6       6       6       6      11

# Checking the Label.Physics
levels(as.factor(dt.swiss$Label.Physics))

## [1] "N/A"                    "Physique"               "Physique (pour CMS)"
## [4] "Physique niveau avancé" "Physique pour PAM"
```

From the results above, we can see that, there are **9** students who have **NULL** values for their `Rating.Physics`.

Furthermore, there are **11** students who have **Echelles** values for their `Scale.Physics`. Let's have a deeper look.

```
# Students with NULL Physics Scores
t.physics.na = dt.swiss %>% filter(Scale.Physics == "N/A")
t.physics.ech = dt.swiss %>% filter(Scale.Physics == "Echelles")

# Summary of their Institution and Title.Diploma
t.physics.na$Institution

## [1] "Collège pour adultes Alice-Rivaz, Genève"
## [2] "Gymnase Provence, Lausanne"
## [3] "Collège Madame de Stäel, Carouge Genève"
## [4] "Collège St-Michel / Kollegium St.Michael, Fribourg"
## [5] "Gymnase Provence, Lausanne"
## [6] "Gymnase de Beaulieu, Lausanne"
```

```
## [7] "Collège de Saussure, Petit-Lancy"
## [8] "Gymnase de Renens, Lausanne"
## [9] "CANT. VAUD"

t.physics.na$Title.Diploma

## [1] "Passerelle maturité professionnelle - hautes écoles universitaires"
## [2] "Mat. reconnue opt. langue moderne"
## [3] "Mat. reconnue opt. langue moderne"
## [4] "Passerelle maturité professionnelle - hautes écoles universitaires"
## [5] "Passerelle maturité professionnelle - hautes écoles universitaires"
## [6] "Maturité professionnelle"
## [7] "Mat. reconnue opt. physique et math"
## [8] "Passerelle maturité professionnelle - hautes écoles universitaires"
## [9] "Passerelle maturité professionnelle - hautes écoles universitaires"

t.physics.ech$Institution

## [1] "Gymnase de Burier, La Tour-de-Peilz" "Liceo di Lugano 1, Lugano"

t.physics.ech$Title.Diploma

## [1] "Mat. reconnue opt. langue moderne"    "Mat. reconnue opt. langues anciennes"

# Summary of grades in French, Maths, Physics, and OS
t.physics.na$Rating.French

## [1] "3.5" "4.5" "4.0" "4.0" "4.5" "4.0" "5.5" "5.5" "5.0"

t.physics.na$Rating.Maths

## [1] 4.5 4.0 4.5 5.0 5.0 5.5 5.0 5.5 5.5

t.physics.na$Rating.Physics

## [1] "N/A" "N/A" "N/A" "N/A" "N/A" "N/A" "N/A" "N/A" "N/A"

t.physics.na$Rating.OS

## [1] "N/A" "N/A" "4.5" "N/A" "N/A" "N/A" "4.5" "N/A" "N/A"

t.physics.ech$Rating.French

## [1] "4.0" "5.0"

t.physics.ech$Rating.Maths

## [1] 4.5 6.0

t.physics.ech$Rating.Physics

## [1] "5.0" "5.5"

t.physics.ech$Rating.OS

## [1] "5.5" "5.0"

# Cleaning up
rm(t.physics.na, t.physics.ech)
```

... *Second*, we will remove the students with **N/A** grades and convert numeric grades to double values.

```
# Filtering out students with N/A grades.
dt.swiss.Physics = dt.swiss %>% filter(Rating.Physics != "N/A")

# Converting maths scores to double values.
dt.swiss.Physics$Rating.Physics = as.double(dt.swiss.Physics$Rating.Physics)
```

```r
dt.swiss.Physics$Scale.Physics = as.double(dt.swiss.Physics$Scale.Physics)

## Warning: NAs introduced by coercion
```

... *Third*, we will summarise the main scores for different `Gender` and `Label.Physics` :

```r
# Summary of Rating.Physics
dt.swiss.Physics %>% group_by(Label.Physics, Gender) %>%
  summarise(Count = n(),
            Mean = mean(Rating.Physics),
            SD = sd(Rating.Physics))

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

## # A tibble: 6 x 5
## # Groups:   Label.Physics [4]
##   Label.Physics          Gender Count  Mean     SD
##   <chr>                  <chr>  <int> <dbl>  <dbl>
## 1 Physique               F        115  4.94  0.610
## 2 Physique               M        207  4.93  0.592
## 3 Physique (pour CMS)    M          7  4.32  0.863
## 4 Physique niveau avancé F          1  4.5   NA
## 5 Physique niveau avancé M          3  4.67  0.289
## 6 Physique pour PAM      F          1  5.5   NA
```

5.2 Exporting Data

In this section, we will export this data for further analysis in a different document.

```r
# Setting the path.
path = paste(here(), "/Data/Prior-Level-Maths/Processed/", sep = "")

# Exporting the dt.pre
write.csv(dt.pre,
          paste(path, "Pre-Score-All.csv", sep = ""))

# Exporting the dt.stat
write.csv(dt.stat,
          paste(path, "Pre-Score-F-S-All.csv", sep = ""))

# Exporting the dt.swiss
write.csv(dt.swiss,
          paste(path, "Pre-Score-Swiss-All.csv", sep = ""))

# Exporting the dt.swiss.math
write.csv(dt.swiss.math,
          paste(path, "Pre-Score-Swiss-Math-Reinf.csv", sep = ""))

# Exporting the dt.swiss.Physics
write.csv(dt.swiss.Physics,
           paste(path, "Pre-Score-Swiss-Phy.csv", sep = ""))

# Cleaning up
rm(path)
```

6 French High-School Students

6.1 Data Pre-Processing & Overview

We will use the `dt.stat` data-frame, which is a filtered dataset after the repeated SCIPERs were removed.

```r
# Filtering out the Swiss students.
dt.french = dt.stat %>% filter(Country.Diploma == "France")
```

6.1.1 Mathematics

... *First* step would be to see if all the students have a grade for mathematics (`Rating.Maths`), if yes, what is the maximum grade (`Scale.Maths`) :

```r
# Checking the validity of Rating.Maths
summary(as.double(dt.french$Rating.Maths))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   16.00   18.00   17.37   19.00   20.00

# Also, checking the validity of Scale.Maths
summary(as.double(dt.french$Scale.Maths))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   20.00   20.00   19.98   20.00   20.00
```

From the results above, we can see that, there is **1** student for whome the `Scale.Maths` is 10.0. So, we will create a variable which computes the proportional score in mathematics (`Prop.Rating.Maths`) :

```r
# Converting math scores to double values.
dt.french$Rating.Maths = as.double(dt.french$Rating.Maths)
dt.french$Scale.Maths = as.double(dt.french$Scale.Maths)

# Creating Proportionl Score
dt.french$Prop.Rating.Maths = (dt.french$Rating.Maths / dt.french$Scale.Maths) * 100
```

... *Second*, we will explore the `label.Maths` for the french students.

```r
# Summary of Math Labels
levels(as.factor(dt.french$Label.Maths))

## [1] "Mathématiques"
```

We observe that there is only one label, and we can assume that in their curriculum, they all had Reinforced Mathematics.

... *Third*, we will summarise the main scores for `Gender` :

```r
# Summary of Prop.Rating.Maths
dt.french %>% group_by(Gender, Enroll.Year) %>%
  summarise(Count = n(),
            Mean = mean(Prop.Rating.Maths),
            SD = sd(Prop.Rating.Maths))

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

## # A tibble: 4 x 5
## # Groups:   Gender [2]
##   Gender Enroll.Year Count  Mean    SD
##   <chr>  <chr>       <int> <dbl> <dbl>
## 1 F      Y1-2017-18     78  86.2  8.86
## 2 F      Y2-2018-19     64  85.1  9.28
## 3 M      Y1-2017-18    169  86.7  9.18
## 4 M      Y2-2018-19    132  88.6  8.46
```

6.1.2 Physics

... *First* step would be to see if all the students have a grade for physics (`Rating.Physics`), if yes, what is the maximum grade (`Scale.Physics`) :

```r
# Checking the validity of Rating.Physics
summary(as.double(dt.french$Rating.Physics))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     9.0    16.0    18.0    17.3    19.0    20.0
```

```r
# Also, checking the validity of Scale.Physics
summary(as.double(dt.french$Scale.Physics))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   20.00   20.00   19.98   20.00   20.00
```

From the results above, we can see that, there is **1** student for whom the `Scale.Physics` is 10.0. So, we will create a variable which computes the proportional score in physics (`Prop.Rating.Physics`) :

```r
# Converting physics scores to double values.
dt.french$Rating.Physics = as.double(dt.french$Rating.Physics)
dt.french$Scale.Physics = as.double(dt.french$Scale.Physics)

# Creating Proportionl Score
dt.french$Prop.Rating.Physics = (dt.french$Rating.Physics / dt.french$Scale.Physics) * 100
```

... *Second*, we will explore the `Label.Physics` for the french students.

```r
# Summary of Math Labels
levels(as.factor(dt.french$Label.Physics))
```

```
## [1] "Physique"        "Physique-Chimie"
```

```r
# Overview of different labels.
dt.french %>% group_by(Label.Physics, Gender) %>%
  summarise(Count = n(),
            Mean = mean(Prop.Rating.Physics),
            SD = sd(Prop.Rating.Physics))
```

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

```
## # A tibble: 4 x 5
## # Groups:   Label.Physics [2]
##   Label.Physics   Gender Count  Mean    SD
##   <chr>           <chr>  <int> <dbl> <dbl>
## 1 Physique        F          3  85    5
## 2 Physique        M          3  88.3  5.77
## 3 Physique-Chimie F        139  85.1  9.35
## 4 Physique-Chimie M        298  87.3  8.87
```

... *Third*, we will summarise the main scores for `Gender` :

```r
# Summary of Prop.Rating.Physics
dt.french %>% group_by(Label.Physics, Gender, Enroll.Year) %>%
  summarise(Count = n(),
            Mean = mean(Prop.Rating.Physics),
            SD = sd(Prop.Rating.Physics))
```

```
## `summarise()` regrouping output by 'Label.Physics', 'Gender' (override with `.groups` argument)
```

```
## # A tibble: 6 x 6
## # Groups:   Label.Physics, Gender [4]
##   Label.Physics   Gender Enroll.Year Count  Mean    SD
##   <chr>           <chr>  <chr>       <int> <dbl> <dbl>
## 1 Physique        F      Y1-2017-18      3  85    5
## 2 Physique        M      Y2-2018-19      3  88.3  5.77
## 3 Physique-Chimie F      Y1-2017-18     75  84.7  9.79
## 4 Physique-Chimie F      Y2-2018-19     64  85.5  8.85
## 5 Physique-Chimie M      Y1-2017-18    169  86.7  9.44
## 6 Physique-Chimie M      Y2-2018-19    129  88.0  8.04
```

6.2 Exporting Data

In this section, we will export this data for further analysis in a different document.

```
# Setting the path.
path = paste(here(), "/Data/Prior-Level-Maths/Processed/", sep = "")

# Exporting the dt.french
write.csv(dt.french,
          paste(path, "Pre-Score-France.csv", sep = ""))

# Cleaning up
rm(path)
```

7 Splitting Students into Performance Label

In this section, we will assign the FRENCH and SWISS students into either *high* or *low* performance labels based on their scores in Reinforced maths exam at the high-school level.

Before, we proceed onto this step, we will import the normalized score values (`Nor.Score`) and merge it with their pre-scores.

7.1 Importing Normalized Score Data

In this section, we will import the `.csv` file which contains the Normalized Score of the Linear Algebra semester exam.

```
# Importing the data about Linear Algebra Course.
path = paste(here(),
             "/Data/Scores/Normalized-Volunteer-Data/",
             sep = "")
dt.score = read.csv(paste(path, "Total-Data-Normalized-Score-DI-Filtered.csv", sep = ""),
                    header = TRUE)
dt.score$X = NULL

# Convert ID.Anon to Character.
dt.score$ID.Anon = as.character(dt.score$ID.Anon)
dt.score$SCIPER = lapply(dt.score$ID.Anon, GenerateSCIPER)
dt.score$SCIPER = unlist(dt.score$SCIPER)

# Cleaning up
rm(path)
```

7.2 Threshold for Splitting

In this section, we will try two different thresholds which were discussed together with Simone and Cécile. Based on the distribution of students, we will consider which threshold must we choose for the further analysis.

7.2.1 Threshold 1

Swiss : `Low: [0, 4.0]` and `High: (4.0, 6.0]`

French : `Low: [0, 16]` and `High: (16, 20]`

Next, we will compare this threshold with the *median cut* and to see which is a better fit.

Swiss Students : Median Cut

```
t.stat = dt.swiss.math
# Splitting by Median
t.stat$Performance.Level = ifelse(t.stat$Rating.Maths >= median(t.stat$Rating.Maths),
                                                                "High",
                                                                "Low")
# Summary
```

```
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 29    56
## 2 Low                  25    55
```

Swiss Students : Threshold 1

```
t.stat = dt.swiss.math
# Splitting by Fixed Threshold
t.stat$Performance.Level = ifelse(t.stat$Rating.Maths > 4.0, "High", "Low")

# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 42    83
## 2 Low                  12    28
```

French Students : Median Cut

```
t.stat = dt.french
# Splitting by Median
t.stat$Performance.Level = ifelse(t.stat$Prop.Rating.Maths >= median(t.stat$Prop.Rating.Maths),
                                                                "High",
                                                                "Low")


# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 64   168
## 2 Low                  78   133
```

French Students : Threshold 1

```
t.stat = dt.french
# Splitting by Fixed Threshold
t.stat$Performance.Level = ifelse(t.stat$Prop.Rating.Maths > 80.0, "High", "Low")

# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 93   231
## 2 Low                  49    70
```

7.2.2 Threshold 2

Swiss : `Low: [0, 4.5]` and `High: (4.5, 6.0]`

French : `Low: [0, 17]` and `High: (17, 20]`

Next, we will compare this threshold with the *median cut* and to see which is a better fit.

Swiss Students : Median Cut

```
t.stat = dt.swiss.math
# Splitting by Median
t.stat$Performance.Level = ifelse(t.stat$Rating.Maths >= median(t.stat$Rating.Maths),
                                                              "High",
                                                              "Low")

# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 29    56
## 2 Low                  25    55
```

Swiss Students : Threshold 2

```
t.stat = dt.swiss.math
# Splitting by Fixed Threshold
t.stat$Performance.Level = ifelse(t.stat$Rating.Maths > 4.5, "High", "Low")

# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 29    57
## 2 Low                  25    54
```

French Students : Median Cut

```
t.stat = dt.french
# Splitting by Median
t.stat$Performance.Level = ifelse(t.stat$Prop.Rating.Maths >= median(t.stat$Prop.Rating.Maths),
                                                              "High",
                                                              "Low")

# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 64   168
## 2 Low                  78   133
```

French Students : Threshold 2

```
t.stat = dt.french
# Splitting by Fixed Threshold
t.stat$Performance.Level = ifelse(t.stat$Prop.Rating.Maths > 85.0, "High", "Low")

# Summary
t.stat %>% group_by(Performance.Level, Gender) %>%
  tally() %>%
  spread(Gender, n)

## # A tibble: 2 x 3
## # Groups:   Performance.Level [2]
##   Performance.Level     F     M
##   <chr>             <int> <int>
## 1 High                 65   168
## 2 Low                  77   133

# Clean-up
rm(t.stat)
```

7.3 Deciding which thresold to use (Threshold 1 vs. Threshold 2)

Comparison of the above two thresholds with the median cut reveals that **Thresold 2** (illustrated below) is a much better fit, and is equivalent to the Median Cut. So we will use this threshold for splitting and further analysis.

Swiss : `Low: [0, 4.5]` and `High: (4.5, 6.0]`

French : `Low: [0, 17]` and `High: (17, 20]`

7.4 Swiss Students : Splitting

Splitting the data into new `Performance.Levels` :

```
fixed.swiss.math = dt.swiss.math
fixed.swiss.math$Performance.Level = ifelse(fixed.swiss.math$Rating.Maths > 4.5,
                                            "High",
                                            "Low")
```

7.5 Swiss Students : Merging Semester Scores

We will perform a merge operation with the dataset containing students' scores.

```
# Joining the two datasets.
swiss.stat = merge(x = fixed.swiss.math,
                   y = dt.score,
                   by = "SCIPER")
```

We see that the number of students in the merged dataset is slightly less (149 as compared to 443 for French students with reinforced mathematics).

7.6 French Students : Splitting

Splitting the data into new `Performance.Levels` :

```
fixed.french = dt.french
fixed.french$Performance.Level = ifelse(fixed.french$Prop.Rating.Maths > 85.0,
                                        "High",
                                        "Low")
```

7.7 French Students : Merging Semester Scores

We will perform a merge operation with the dataset containing students' scores.

```r
# Joining the two datasets.
french.stat = merge(x = fixed.french,
                    y = dt.score,
                    by = "SCIPER")
```

We see that the number of students in the merged dataset is slightly less (360 as compared to 443 for French students with reinforced mathematics).

8 Combinging French and Swiss Datasets

Before we go on and analyze the data, we will join the Swiss and the French students' datasets into a single dataset.

```r
# Combining Swiss and French students
t.all = rbind(fixed.swiss.math %>% select(SCIPER, Gender, Performance.Level),
                fixed.french %>% select(SCIPER, Gender, Performance.Level))

# Now merging with the Score dataset.
all.stat = merge(x = t.all,
                 y = dt.score,
                 by = "SCIPER")
```

9 Analysis and Visualization

Let's start with the visualizations (box and confidence interval plots) that give an overwiew of the data across different categorical variables (primarily `Gender`, `Course.Yeaar`, and `Condition`).

First, we will analyze the French, and Swiss students individually, and then I will focus on their combined analysis.

9.1 French Student

Analysis and visualization for the French students.

```r
# Assigning French students to a temporary variable.
dt.stat = french.stat
```

9.1.1 Tabular Summary

Firstly, summarizing the `Performance.Level` across `Course.Year` :

```r
# Counts
dt.stat %>% group_by(Course.Year, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 2 x 3
## # Groups:   Course.Year [2]
##   Course.Year  High   Low
##   <fct>       <int> <int>
## 1 Y1-2017-18    122   113
## 2 Y2-2018-19     65    60

# Mean and SD
dt.stat %>% group_by(Course.Year, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

```
## # A tibble: 4 x 5
## # Groups:   Course.Year [2]
##   Course.Year Performance.Level     N   Mean    SD
##   <fct>       <chr>             <int>  <dbl> <dbl>
## 1 Y1-2017-18  High                122 0.453  0.745
## 2 Y1-2017-18  Low                 113 0.0762 0.933
## 3 Y2-2018-19  High                 65 0.129  0.893
## 4 Y2-2018-19  Low                  60 0.128  0.740
```

Secondly, summarizing the `Performace.Level` across `Course.Year, Gender` :

```
# Counts
dt.stat %>% group_by(Course.Year, Gender.y, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 4 x 4
## # Groups:   Course.Year, Gender.y [4]
##   Course.Year Gender.y  High   Low
##   <fct>       <fct>    <int> <int>
## 1 Y1-2017-18  F           37    37
## 2 Y1-2017-18  M           85    76
## 3 Y2-2018-19  F           19    30
## 4 Y2-2018-19  M           46    30

# Mean and SD
dt.stat %>% group_by(Course.Year, Gender.y, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

## `summarise()` regrouping output by 'Course.Year', 'Gender.y' (override with `.groups` argument)

## # A tibble: 8 x 6
## # Groups:   Course.Year, Gender.y [4]
##   Course.Year Gender.y Performance.Level     N   Mean    SD
##   <fct>       <fct>    <chr>             <int>  <dbl> <dbl>
## 1 Y1-2017-18  F        High                 37 0.459  0.628
## 2 Y1-2017-18  F        Low                  37 0.0716 0.949
## 3 Y1-2017-18  M        High                 85 0.450  0.794
## 4 Y1-2017-18  M        Low                  76 0.0785 0.931
## 5 Y2-2018-19  F        High                 19 0.175  0.823
## 6 Y2-2018-19  F        Low                  30 0.239  0.676
## 7 Y2-2018-19  M        High                 46 0.110  0.928
## 8 Y2-2018-19  M        Low                  30 0.0173 0.796
```

Thirdly, summarizing the `Performance.Level` across `Course.Year, Gender, Condition` :

```
# Counts
dt.stat %>% group_by(Course.Year, Gender.y, Condition, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 8 x 5
## # Groups:   Course.Year, Gender.y, Condition [8]
##   Course.Year Gender.y Condition  High   Low
##   <fct>       <fct>    <fct>     <int> <int>
## 1 Y1-2017-18  F        Control      31    31
## 2 Y1-2017-18  F        Flipped       6     6
## 3 Y1-2017-18  M        Control      66    63
## 4 Y1-2017-18  M        Flipped      19    13
## 5 Y2-2018-19  F        Control       7    16
## 6 Y2-2018-19  F        Flipped      12    14
## 7 Y2-2018-19  M        Control      24    15
## 8 Y2-2018-19  M        Flipped      22    15
```

```
# Mean and SD
dt.stat %>% group_by(Course.Year, Gender.y, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))
```

```
## `summarise()` regrouping output by 'Course.Year', 'Gender.y', 'Condition' (override with `.groups` argument)
```

```
## # A tibble: 16 x 7
## # Groups:   Course.Year, Gender.y, Condition [8]
##    Course.Year Gender.y Condition Performance.Level     N     Mean    SD
##    <fct>       <fct>    <fct>     <chr>             <int>    <dbl> <dbl>
##  1 Y1-2017-18  F        Control   High                 31  0.460   0.656
##  2 Y1-2017-18  F        Control   Low                  31  0.101   0.921
##  3 Y1-2017-18  F        Flipped   High                  6  0.458   0.512
##  4 Y1-2017-18  F        Flipped   Low                   6 -0.0794  1.17
##  5 Y1-2017-18  M        Control   High                 66  0.425   0.790
##  6 Y1-2017-18  M        Control   Low                  63  0.0938  0.948
##  7 Y1-2017-18  M        Flipped   High                 19  0.537   0.827
##  8 Y1-2017-18  M        Flipped   Low                  13  0.00440 0.875
##  9 Y2-2018-19  F        Control   High                  7  0.144   0.845
## 10 Y2-2018-19  F        Control   Low                  16  0.0946  0.671
## 11 Y2-2018-19  F        Flipped   High                 12  0.193   0.847
## 12 Y2-2018-19  F        Flipped   Low                  14  0.403   0.667
## 13 Y2-2018-19  M        Control   High                 24  0.196   0.941
## 14 Y2-2018-19  M        Control   Low                  15  0.198   0.588
## 15 Y2-2018-19  M        Flipped   High                 22  0.0157  0.927
## 16 Y2-2018-19  M        Flipped   Low                  15 -0.163   0.946
```

Fourthly, summarizing the `Performance.Level` across `Course.Year, Condition` :

```
# Counts
dt.stat %>% group_by(Course.Year, Condition, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)
```

```
## # A tibble: 4 x 4
## # Groups:   Course.Year, Condition [4]
##   Course.Year Condition  High   Low
##   <fct>       <fct>     <int> <int>
## 1 Y1-2017-18  Control      97    94
## 2 Y1-2017-18  Flipped      25    19
## 3 Y2-2018-19  Control      31    31
## 4 Y2-2018-19  Flipped      34    29
```

```
# Mean and SD
dt.stat %>% group_by(Course.Year, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))
```

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

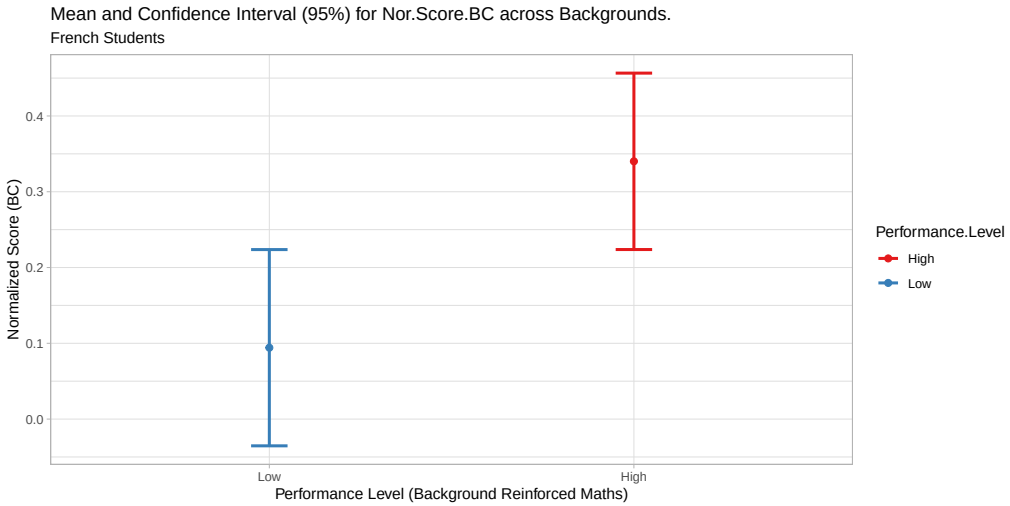
```
## # A tibble: 8 x 6
## # Groups:   Course.Year, Condition [4]
##   Course.Year Condition Performance.Level     N    Mean    SD
##   <fct>       <fct>     <chr>             <int>   <dbl> <dbl>
## 1 Y1-2017-18  Control   High                 97  0.436  0.746
## 2 Y1-2017-18  Control   Low                  94  0.0961 0.934
## 3 Y1-2017-18  Flipped   High                 25  0.518  0.754
## 4 Y1-2017-18  Flipped   Low                  19 -0.0221 0.944
## 5 Y2-2018-19  Control   High                 31  0.184  0.906
## 6 Y2-2018-19  Control   Low                  31  0.144  0.624
## 7 Y2-2018-19  Flipped   High                 34  0.0783 0.891
## 8 Y2-2018-19  Flipped   Low                  29  0.110  0.859
```

Fifthly, summarizing the `Performance.Level` across `Gender, Condition` :

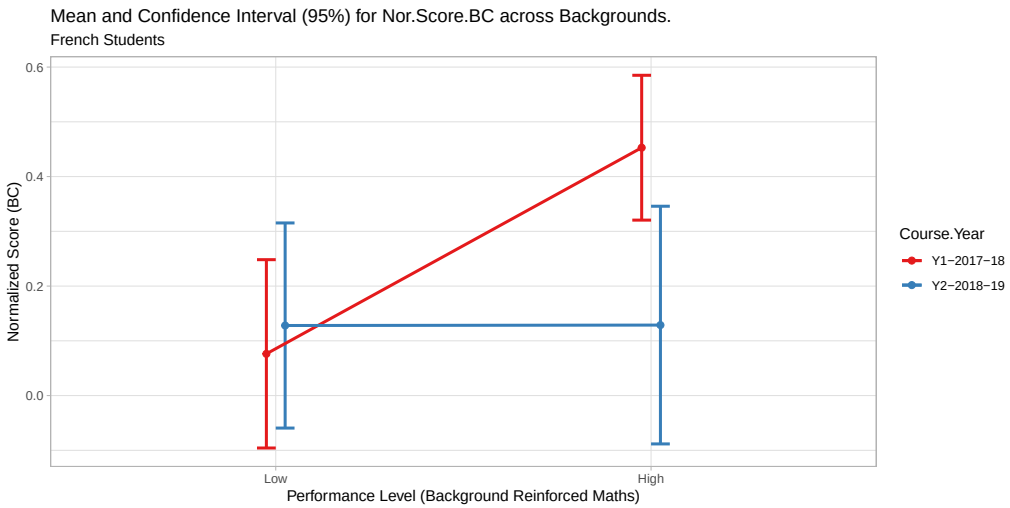
```
# Counts
dt.stat %>% group_by(Gender.y, Condition, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 4 x 4
## # Groups:   Gender.y, Condition [4]
##   Gender.y Condition  High   Low
##   <fct>    <fct>     <int> <int>
## 1 F        Control      38    47
## 2 F        Flipped      18    20
## 3 M        Control      90    78
## 4 M        Flipped      41    28

# Mean and SD
dt.stat %>% group_by(Gender.y, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

## # A tibble: 8 x 6
## # Groups:   Gender.y, Condition [4]
##   Gender.y Condition Performance.Level     N    Mean    SD
##   <fct>    <fct>     <chr>             <int>   <dbl> <dbl>
## 1 F        Control   High                 38  0.402  0.693
## 2 F        Control   Low                  47  0.0988 0.837
## 3 F        Flipped   High                 18  0.281  0.747
## 4 F        Flipped   Low                  20  0.258  0.846
## 5 M        Control   High                 90  0.364  0.833
## 6 M        Control   Low                  78  0.114  0.888
## 7 M        Flipped   High                 41  0.257  0.910
## 8 M        Flipped   Low                  28 -0.0852 0.901
```

9.1.2 Visualizations of Normalized Score

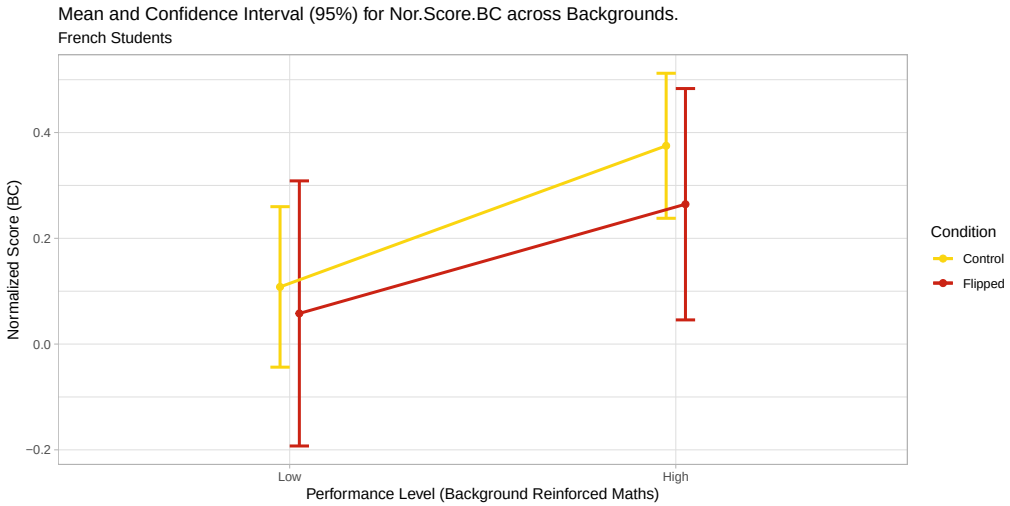
Difference Across Performance Level



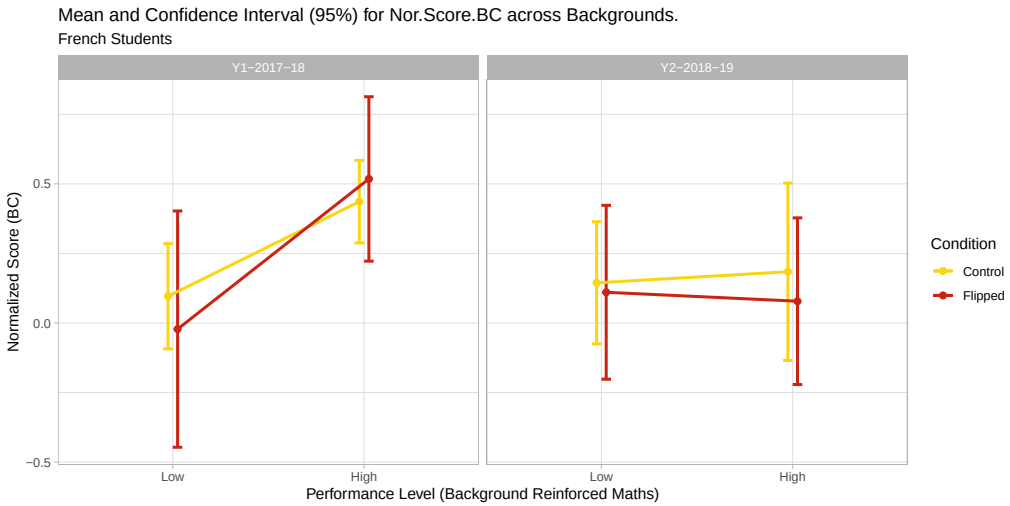
Performance.Level and Course.Year



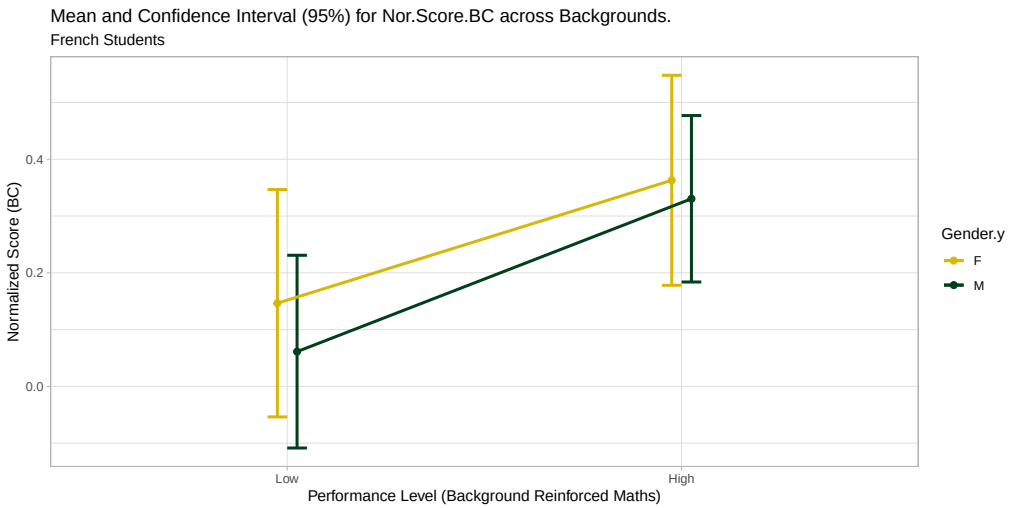
Performance.Level and Condition



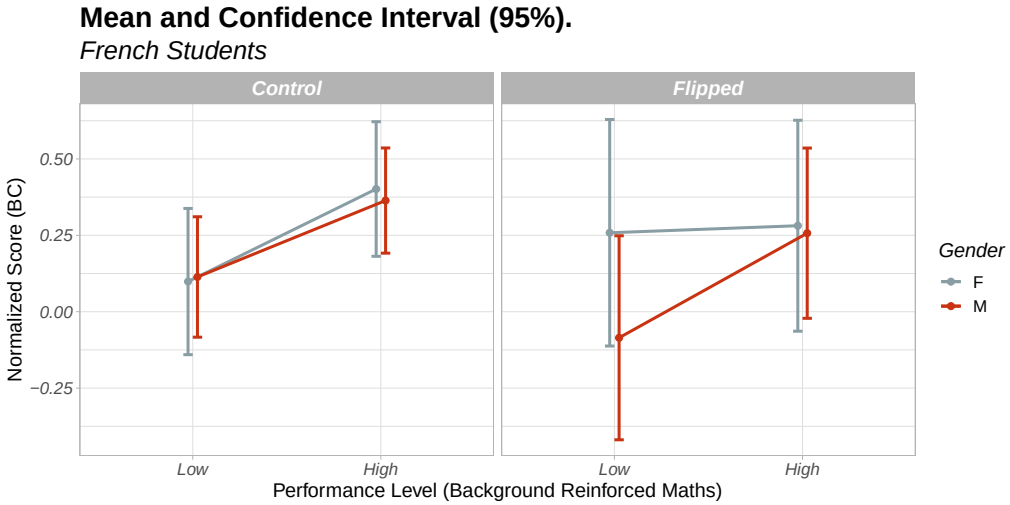
... faceting the above plot with Course Year.



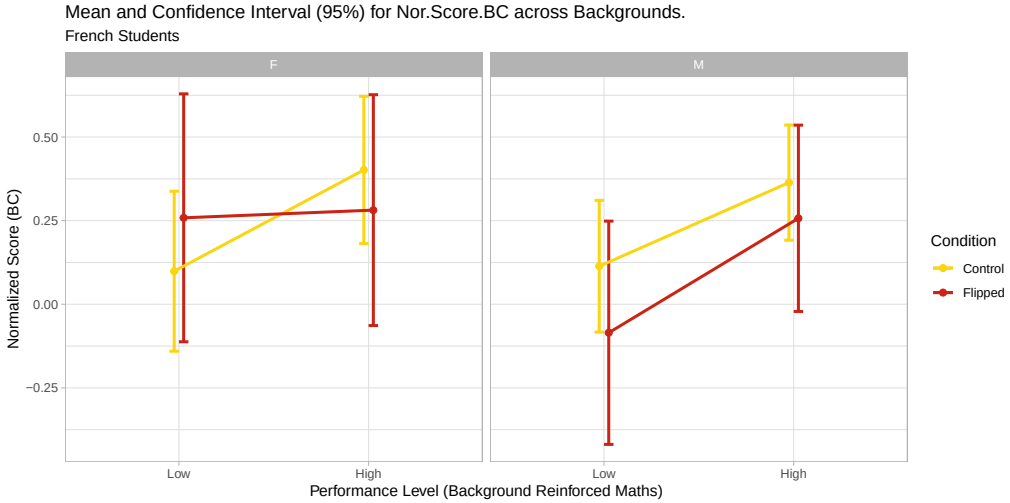
Performance.Level and Gender



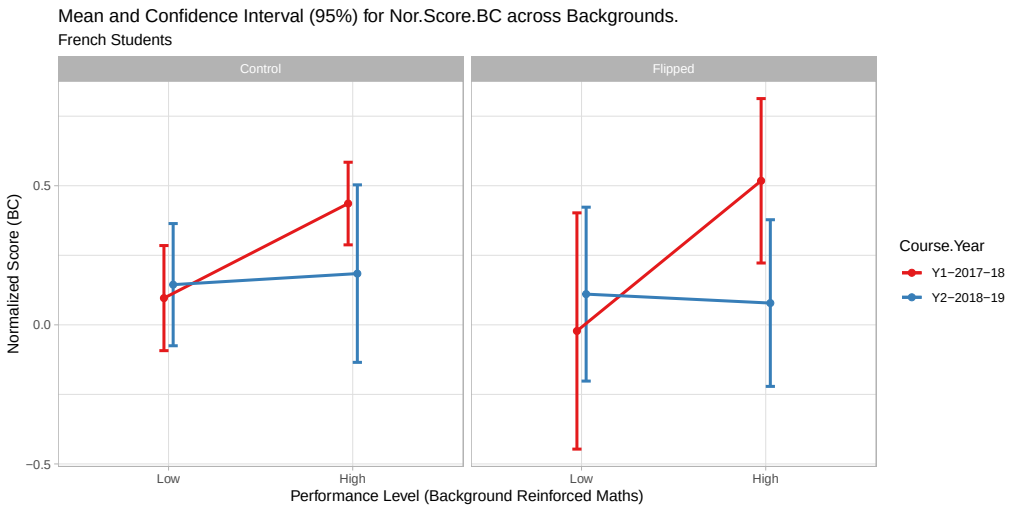
Performance.Level, Condition, and Gender



Performance.Level, Gender, and Condition



Performance.Level, Condition, and Course.Year



9.2 Swiss Students

In this section we will focus on the analysis of Swiss students.

```
# Assigning Swiss students to a temporary variable.
dt.stat = swiss.stat
```

9.2.1 Tabular Summary

Firstly, summarizing the `Performance.Level` across `Course.Year` :

```
# Counts
dt.stat %>% group_by(Course.Year, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 2 x 3
## # Groups:   Course.Year [2]
##   Course.Year  High   Low
##   <fct>       <int> <int>
## 1 Y1-2017-18     54    47
## 2 Y2-2018-19     22    26

# Mean and SD
dt.stat %>% group_by(Course.Year, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

## # A tibble: 4 x 5
## # Groups:   Course.Year [2]
##   Course.Year Performance.Level     N   Mean    SD
##   <fct>       <chr>             <int>  <dbl> <dbl>
## 1 Y1-2017-18  High                 54  0.262 0.837
## 2 Y1-2017-18  Low                  47 -0.345 1.09
## 3 Y2-2018-19  High                 22  0.311 0.678
## 4 Y2-2018-19  Low                  26 -0.536 1.06
```

Secondly, summarizing the `Performace.Level` across `Course.Year, Gender` :

```
# Counts
dt.stat %>% group_by(Course.Year, Gender.y, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)
```

```
## # A tibble: 4 x 4
## # Groups:   Course.Year, Gender.y [4]
##   Course.Year Gender.y  High   Low
##   <fct>       <fct>    <int> <int>
## 1 Y1-2017-18  F           21    17
## 2 Y1-2017-18  M           33    30
## 3 Y2-2018-19  F            4     6
## 4 Y2-2018-19  M           18    20
```

```
# Mean and SD
dt.stat %>% group_by(Course.Year, Gender.y, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))
```

```
## `summarise()` regrouping output by 'Course.Year', 'Gender.y' (override with `.groups` argument)
```

```
## # A tibble: 8 x 6
## # Groups:   Course.Year, Gender.y [4]
##   Course.Year Gender.y Performance.Level     N   Mean    SD
##   <fct>       <fct>    <chr>             <int>  <dbl> <dbl>
## 1 Y1-2017-18  F        High                 21  0.323 0.769
## 2 Y1-2017-18  F        Low                  17 -0.213 0.950
## 3 Y1-2017-18  M        High                 33  0.223 0.887
## 4 Y1-2017-18  M        Low                  30 -0.419 1.17
## 5 Y2-2018-19  F        High                  4  0.244 0.507
## 6 Y2-2018-19  F        Low                   6 -0.526 0.962
## 7 Y2-2018-19  M        High                 18  0.326 0.722
## 8 Y2-2018-19  M        Low                  20 -0.539 1.11
```

Thirdly, summarizing the `Performance.Level` across `Course.Year, Gender, Condition` :

```
# Counts
dt.stat %>% group_by(Course.Year, Gender.y, Condition, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)
```

```
## # A tibble: 8 x 5
## # Groups:   Course.Year, Gender.y, Condition [8]
##   Course.Year Gender.y Condition  High   Low
##   <fct>       <fct>    <fct>     <int> <int>
## 1 Y1-2017-18  F        Control      16    14
## 2 Y1-2017-18  F        Flipped       5     3
## 3 Y1-2017-18  M        Control      25    26
## 4 Y1-2017-18  M        Flipped       8     4
## 5 Y2-2018-19  F        Control       2     4
## 6 Y2-2018-19  F        Flipped       2     2
## 7 Y2-2018-19  M        Control       8    10
## 8 Y2-2018-19  M        Flipped      10    10
```

```
# Mean and SD
dt.stat %>% group_by(Course.Year, Gender.y, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))
```

```
## `summarise()` regrouping output by 'Course.Year', 'Gender.y', 'Condition' (override with `.groups` argument)
```

```
## # A tibble: 16 x 7
## # Groups:   Course.Year, Gender.y, Condition [8]
##    Course.Year Gender.y Condition Performance.Level     N    Mean    SD
##    <fct>       <fct>    <fct>     <chr>             <int>   <dbl> <dbl>
##  1 Y1-2017-18  F        Control   High                 16  0.389  0.825
##  2 Y1-2017-18  F        Control   Low                  14 -0.324  1.01
```

```
##  3 Y1-2017-18  F        Flipped   High                5  0.110  0.571
##  4 Y1-2017-18  F        Flipped   Low                 3  0.303  0.273
##  5 Y1-2017-18  M        Control   High               25  0.314  0.949
##  6 Y1-2017-18  M        Control   Low                26 -0.387  1.17
##  7 Y1-2017-18  M        Flipped   High                8 -0.0617 0.624
##  8 Y1-2017-18  M        Flipped   Low                 4 -0.624  1.33
##  9 Y2-2018-19  F        Control   High                2  0.296  0.726
## 10 Y2-2018-19  F        Control   Low                 4 -0.817  0.769
## 11 Y2-2018-19  F        Flipped   High                2  0.193  0.484
## 12 Y2-2018-19  F        Flipped   Low                 2  0.0561 1.36
## 13 Y2-2018-19  M        Control   High                8  0.390  0.820
## 14 Y2-2018-19  M        Control   Low                10 -0.669  1.21
## 15 Y2-2018-19  M        Flipped   High               10  0.275  0.675
## 16 Y2-2018-19  M        Flipped   Low                10 -0.409  1.04
```

Fourthly, summarizing the `Performance.Level` across `Course.Year, Condition` :

```
# Counts
dt.stat %>% group_by(Course.Year, Condition, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 4 x 4
## # Groups:   Course.Year, Condition [4]
##   Course.Year Condition  High   Low
##   <fct>       <fct>     <int> <int>
## 1 Y1-2017-18  Control      41    40
## 2 Y1-2017-18  Flipped      13     7
## 3 Y2-2018-19  Control      10    14
## 4 Y2-2018-19  Flipped      12    12

# Mean and SD
dt.stat %>% group_by(Course.Year, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

## # A tibble: 8 x 6
## # Groups:   Course.Year, Condition [4]
##   Course.Year Condition Performance.Level     N     Mean    SD
##   <fct>       <fct>     <chr>             <int>    <dbl> <dbl>
## 1 Y1-2017-18  Control   High                 41  0.343   0.893
## 2 Y1-2017-18  Control   Low                  40 -0.365   1.11
## 3 Y1-2017-18  Flipped   High                 13  0.00440 0.586
## 4 Y1-2017-18  Flipped   Low                   7 -0.227   1.07
## 5 Y2-2018-19  Control   High                 10  0.371   0.764
## 6 Y2-2018-19  Control   Low                  14 -0.711   1.07
## 7 Y2-2018-19  Flipped   High                 12  0.261   0.629
## 8 Y2-2018-19  Flipped   Low                  12 -0.332   1.04
```

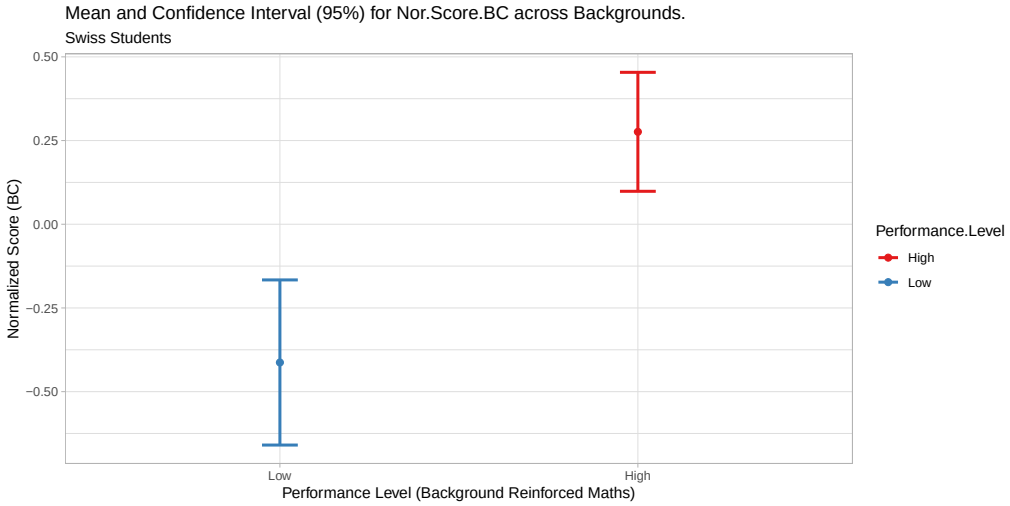
Fifthly, summarizing the `Performance.Level` across `Gender, Condition` :

```
# Counts
dt.stat %>% group_by(Gender.y, Condition, Performance.Level) %>%
  tally() %>%
  spread(Performance.Level, n)

## # A tibble: 4 x 4
## # Groups:   Gender.y, Condition [4]
##   Gender.y Condition  High   Low
##   <fct>    <fct>     <int> <int>
## 1 F        Control      18    18
```

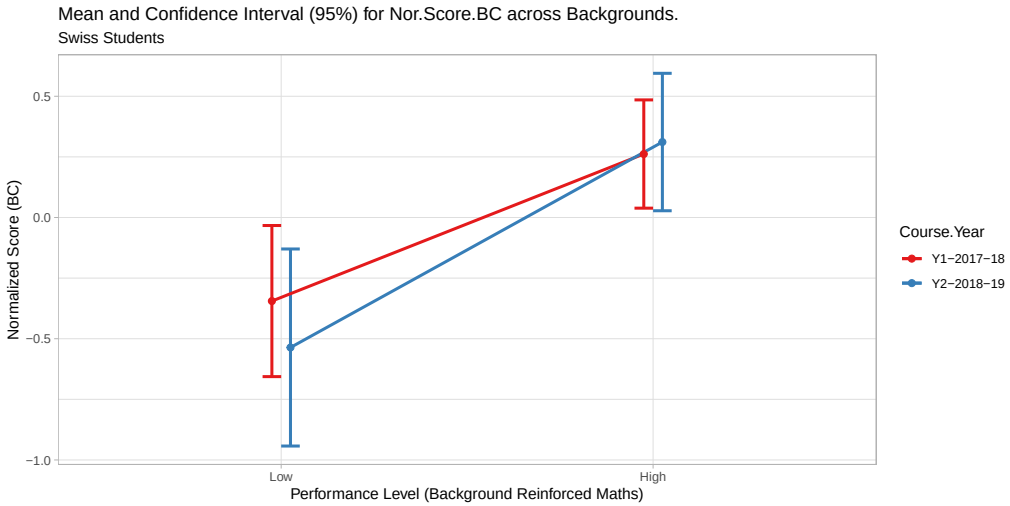
```
## 2 F        Flipped       7     5
## 3 M        Control      33    36
## 4 M        Flipped      18    14

# Mean and SD
dt.stat %>% group_by(Gender.y, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

## # A tibble: 8 x 6
## # Groups:   Gender.y, Condition [4]
##   Gender.y Condition Performance.Level     N   Mean    SD
##   <fct>    <fct>     <chr>             <int>  <dbl> <dbl>
## 1 F        Control   High                 18  0.379 0.796
## 2 F        Control   Low                  18 -0.433 0.966
## 3 F        Flipped   High                  7  0.134 0.508
## 4 F        Flipped   Low                   5  0.204 0.717
## 5 M        Control   High                 33  0.332 0.908
## 6 M        Control   Low                  36 -0.466 1.17
## 7 M        Flipped   High                 18  0.125 0.657
## 8 M        Flipped   Low                  14 -0.471 1.08
```

9.2.2 Visualizations of Normalized Score

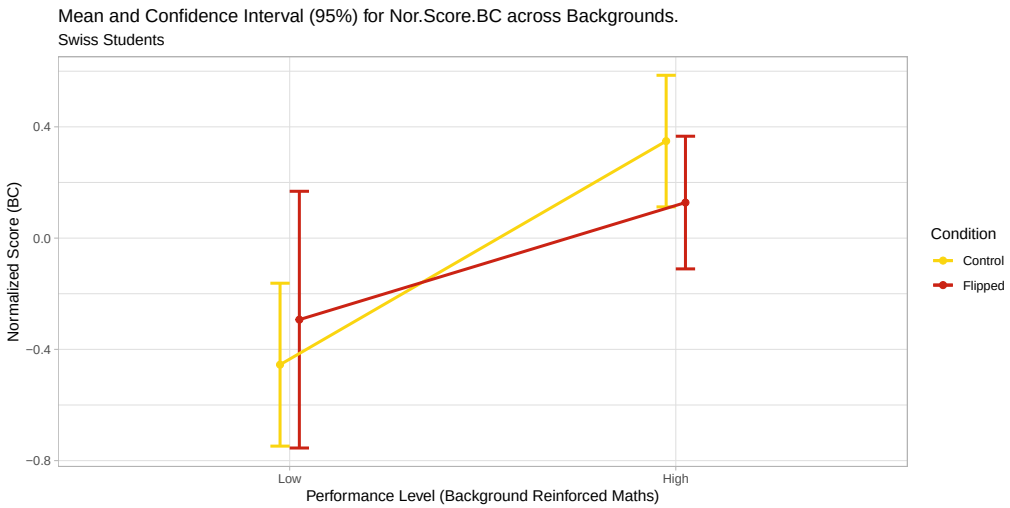
Difference Across Performance Level



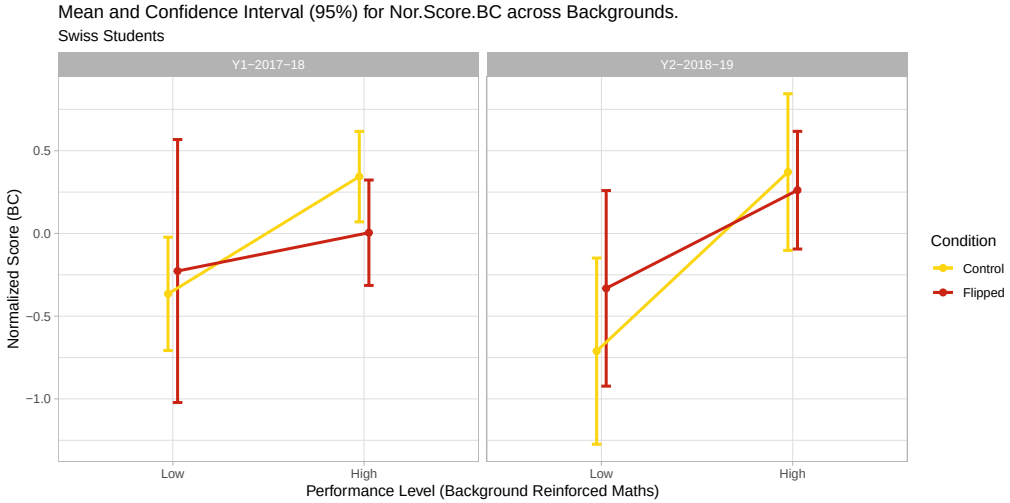
Performance.Level and Course.Year



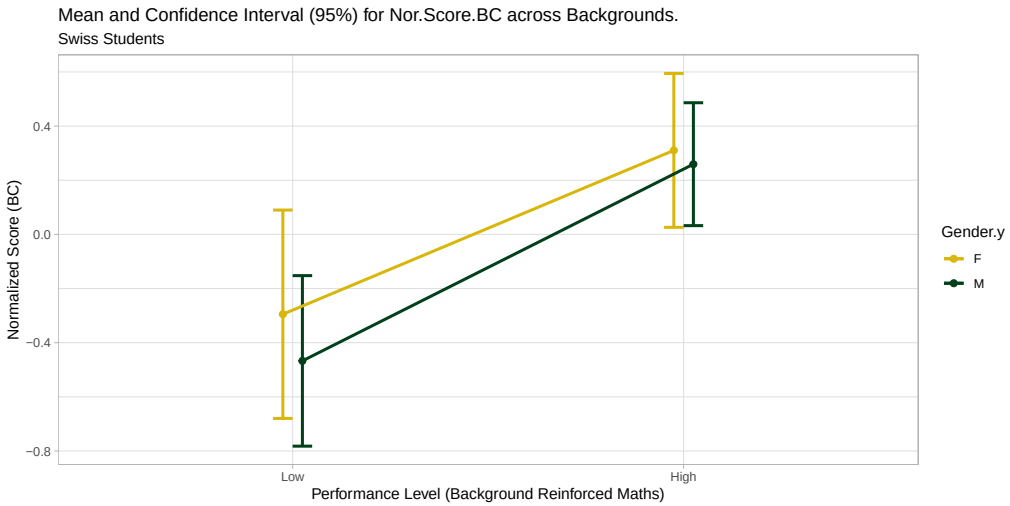
Performance.Level and Condition



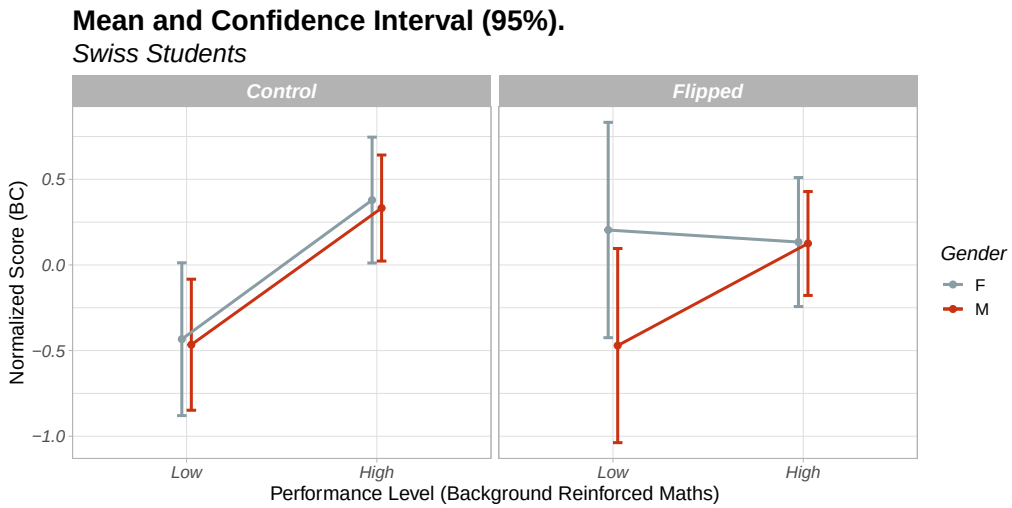
... faceting the above plot by Course Year



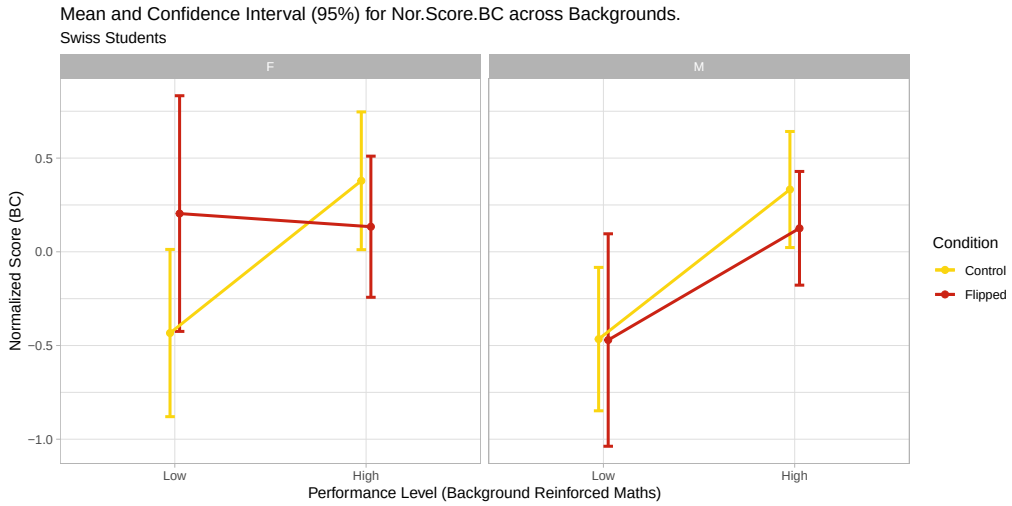
Performance.Level and Gender



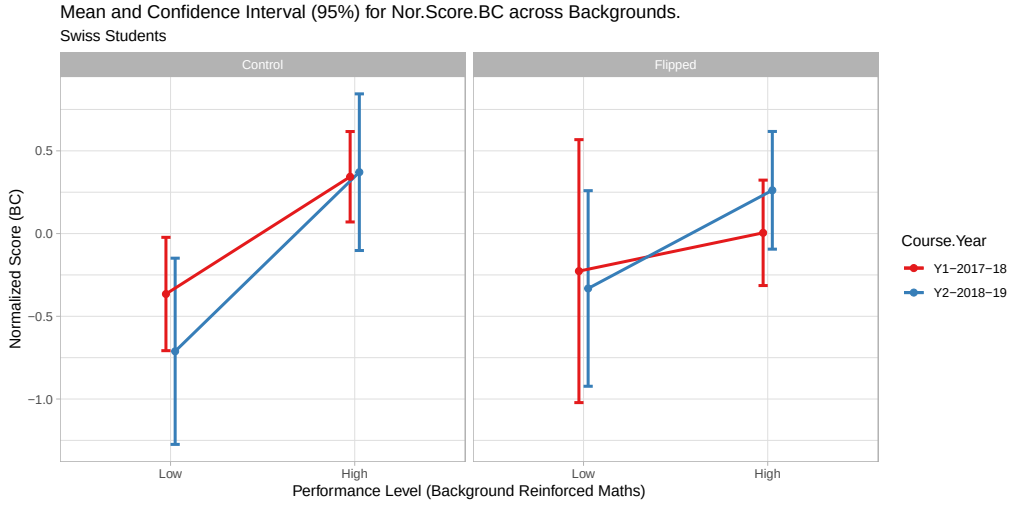
Performance.Level, Condition, and Gender



Performance.Level, Gender, and Condition



Performance.Level, Condition, and Course.Year



10 Combining Swiss & French Students with a Fixed Threshold

```
# Combining the Swiss and French Students.
dt.comb = rbind(fixed.french %>% select(SCIPER, Gender, Performance.Level),
                fixed.swiss.math %>% select(SCIPER, Gender, Performance.Level))
# Joining the two datasets.
dt.stat = merge(x = dt.comb,
                y = dt.score,
                by = "SCIPER")
```

... summary of the data.

```
dt.stat %>% group_by(Gender.y, Condition, Performance.Level) %>%
  summarise(N = n()) %>%
  spread(Performance.Level, N)

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

## # A tibble: 4 x 4
## # Groups:   Gender.y, Condition [4]
##   Gender.y Condition  High   Low
##   <fct>    <fct>     <int> <int>
## 1 F        Control      56    65
## 2 F        Flipped      25    25
## 3 M        Control     123   114
## 4 M        Flipped      59    42
```

```
dt.stat %>% group_by(Gender.y, Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

## # A tibble: 8 x 6
## # Groups:   Gender.y, Condition [4]
##   Gender.y Condition Performance.Level     N    Mean    SD
##   <fct>    <fct>     <chr>             <int>   <dbl> <dbl>
## 1 F        Control   High                 56  0.394  0.720
## 2 F        Control   Low                  65 -0.0486 0.899
## 3 F        Flipped   High                 25  0.240  0.682
## 4 F        Flipped   Low                  25  0.248  0.808
## 5 M        Control   High                123  0.355  0.850
## 6 M        Control   Low                 114 -0.0693 1.02
## 7 M        Flipped   High                 59  0.217  0.837
## 8 M        Flipped   Low                  42 -0.214  0.970
```

...summary of their scores.

```
dt.stat %>% group_by(Condition, Performance.Level) %>%
  summarise(N = n(),
            Mean = mean(Nor.Score.BC),
            SD = sd(Nor.Score.BC))

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

## # A tibble: 4 x 5
## # Groups:   Condition [2]
##   Condition Performance.Level     N    Mean    SD
##   <fct>     <chr>             <int>   <dbl> <dbl>
## 1 Control   High                179  0.368  0.810
## 2 Control   Low                 179 -0.0618 0.974
## 3 Flipped   High                 84  0.224  0.790
## 4 Flipped   Low                  67 -0.0416 0.934

# Kruskal-Wallis: Difference across condition.
kruskal.test(dt.stat$Nor.Score.BC~dt.stat$Condition)

##
##  Kruskal-Wallis rank sum test
##
## data:  dt.stat$Nor.Score.BC by dt.stat$Condition
## Kruskal-Wallis chi-squared = 0.45725, df = 1, p-value = 0.4989
```

```r
epsilonSquared(x = dt.stat$Nor.Score.BC,
               g = dt.stat$Condition)

## epsilon.squared
##           9e-04

# Kruskal-Wallis: Differences across performance level.
kruskal.test(dt.stat$Nor.Score.BC~as.factor(dt.stat$Performance.Level))

##
##  Kruskal-Wallis rank sum test
##
## data:  dt.stat$Nor.Score.BC by as.factor(dt.stat$Performance.Level)
## Kruskal-Wallis chi-squared = 19.884, df = 1, p-value = 8.229e-06

epsilonSquared(x = dt.stat$Nor.Score.BC,
               g = dt.stat$Performance.Level)

## epsilon.squared
##          0.0391
```

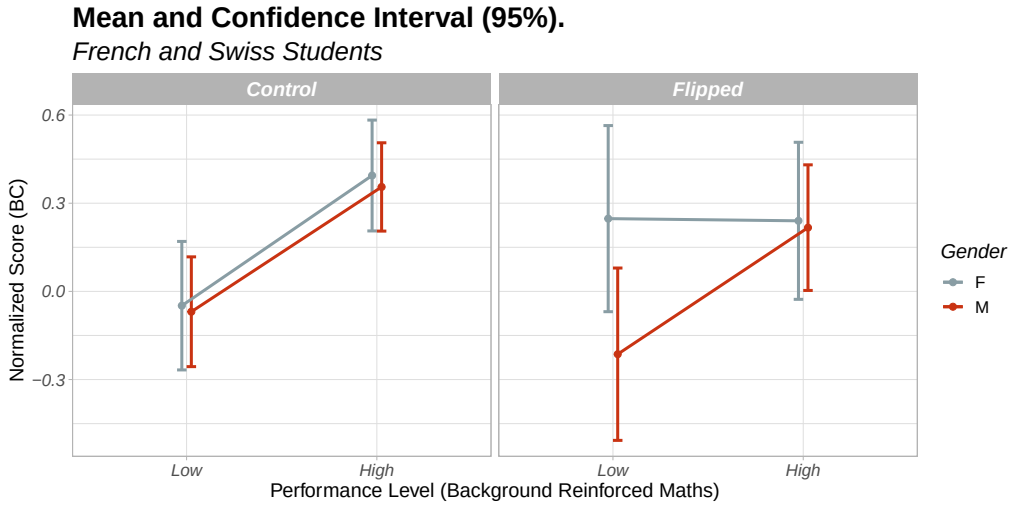
... Kruskal-Wallis (Separately for Conditions)

```r
# Control Condition
t.stat = dt.stat %>% filter(Condition == "Control")
kruskal.test(t.stat$Nor.Score.BC~as.factor(t.stat$Performance.Level))

##
##  Kruskal-Wallis rank sum test
##
## data:  t.stat$Nor.Score.BC by as.factor(t.stat$Performance.Level)
## Kruskal-Wallis chi-squared = 19.31, df = 1, p-value = 1.111e-05

epsilonSquared(x = t.stat$Nor.Score.BC,
               g = as.factor(t.stat$Performance.Level))

## epsilon.squared
##          0.0541

# Flipped Condition
t.stat = dt.stat %>% filter(Condition == "Flipped")
kruskal.test(t.stat$Nor.Score.BC~as.factor(t.stat$Performance.Level))

##
##  Kruskal-Wallis rank sum test
##
## data:  t.stat$Nor.Score.BC by as.factor(t.stat$Performance.Level)
## Kruskal-Wallis chi-squared = 1.9009, df = 1, p-value = 0.168

epsilonSquared(x = t.stat$Nor.Score.BC,
               g = as.factor(t.stat$Performance.Level))

## epsilon.squared
##          0.0127

rm(t.stat)
```

... Kruskal-Wallis Test (Low-performing feamles)

```r
# Preparing Data
t.stat = dt.stat %>% filter(Condition == "Flipped" & Performance.Level == "Low")
# Kruskal-Wallis
kruskal.test(t.stat$Nor.Score.BC~t.stat$Gender.y)

##
##  Kruskal-Wallis rank sum test
```

```
##
## data:  t.stat$Nor.Score.BC by t.stat$Gender.y
## Kruskal-Wallis chi-squared = 3.3915, df = 1, p-value = 0.06553

# Epsilon Squared
epsilonSquared(x = t.stat$Nor.Score.BC,
               g = t.stat$Gender.y)

## epsilon.squared
##          0.0514

# Cleaning up
rm(t.stat)
```



... for the JEE Paper