GEOG0178 Assessment Data Cleaning & Wrangling

DWKB3

2024-04-17

Settup and general pre-processing.

Load packages.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library("SMMT")
library(xml2)
```

Load data sets

```
Parties19 <- read.csv("2019 votes per party.csv")
Pension_vote<- read.csv("2024 pension referendum.csv")
Income_20<- read.csv("clean 2020 income data.csv", na.strings =
                       c("", "NA", "#VALUE!"))
# Assign NAs to the missing values from the original data file
Data_2022<- read.csv("ML 2022 spatial reference.csv")
Data_2023<- read.csv("ML 2023 spatial reference .csv")
Data_2024<- read.csv("ML 2024 spatial reference.csv")
```

Rename the Municipality_Number column of Income_20 to bfs_nr_old to be able to use mapping tables later on:

```
names(Income_20)[names(Income_20) == "Municipality_Number"] <- "bfs_nr_old"
```

As the pension vote data and Data_2024 have the same spatial reference, we can merge them directly. We do so on their municipality number to avoid mismatches due to spelling abbreviation differences.

```
Data_2024_updated <- merge(Data_2024, Pension_vote, by = "bfs_nr_new")
# Safety check
sum(is.na(Data_2024_updated)) # 0, all good.
```

```
## [1] 0
```

Download and extract the 2024 municipality inventory - this only needs to be executed once as data is available locally afterwards (remove # if need to run)

```
# path_inventory_xml <- download_municipality_inventory(path = getwd())
```

Download the 2024 municipality inventory

```
Municipality_inventory_file <- read_xml("eCH0071_240301.xml")

# import_CH_municipality_inventory imports the Swiss municipality inventory
# from the raw XML resource into R as a tibble.
mutations_object <- import_CH_municipality_inventory(Municipality_inventory_file)
mutations        <- mutations_object$mutations
```

Create all the relevant old and new states before creating the mapping tables

The spatial reference (SR) of the 2024 data, the 2024-01-01, will form the new state of all other data sets as it is the latest in time, hence that with the least municipalities.

```
# New state
new_state_2024 <- as.Date("2024-01-01")

# Define 2019, 2020, 2022, and 2023 old states to use in the mapping tables (MT).
old_state_2019 <- as.Date("2019-01-01")
old_state_2020 <- as.Date("2020-10-18")
old_state_2022 <- as.Date("2022-05-01")
old_state_2023 <- as.Date("2023-01-01")
```

Create two MT-making functions.

This is to account for the merger of municipalities forming Verzasca in 2020, not included in the "*Swiss Municipal Data Merger Tool*".

```
# Set up the pre-2020 MT-making function
clean_mapping_table_maker_pre_2020 <- function(mutations, old_state, new_state) {
  # Create a mapping table
  mapping_object <- map_old_to_new_state(mutations, old_state, new_state)
  mapping_table <- mapping_object$mapped

  # Define lakes to be removed
  lakes_to_remove <- c("Zürichsee (ZH)", "Zürichsee (SZ)", "Zürichsee (SG)",
                       "Thunersee", "Brienzersee",
                       "Bielersee (BE)", "Lac de Bienne (NE)", "Lac de Neuchâtel (BE)",
                       "Lac de Neuchâtel (FR)", "Lac de Neuchâtel (VD)",
                       "Lac de Neuchâtel (NE)", "Baldeggersee", "Sempachersee",
                       "Hallwilersee (LU)", "Hallwilersee (AG)", "Zugersee (LU)",
                       "Zugersee (SZ)", "Zugersee (ZG)","Vierwaldstättersee (LU)",
                       "Vierwaldstättersee (UR)", "Vierwaldstättersee (SZ)",
                       "Vierwaldstättersee (OW)", "Vierwaldstättersee (NW)",
                       "Sihlsee", "Sarnersee", "Walensee (GL)", "Walensee (SG)",
                       "Aegerisee", "Lac de la Gruyère", "Murtensee (FR)",
                       "Lac de Morat (VD)", "Bodensee (SH)", "Bodensee (SG)",
                       "Bodensee (TG)", "Lago di Lugano (o.Camp.)",
                       "Lago Maggiore", "Lac de Joux", "Lac Léman (VD)",
                       "Lac Léman (VS)", "Lac Léman (GE)")
```

```r
  # Remove the lakes from the mapping table
  mapping_table <- mapping_table[!mapping_table$name_old %in% lakes_to_remove, ]

  # Handle the special case for the last lake with the same name as a municipality,
  # as well as 3 erroneous municipalities
  mapping_table <- mapping_table[!mapping_table$bfs_nr_old
                                 %in% c(9040, 2391, 5391, 5394), ]

  # Remove the original Verzasca row (without valid old BFS numbers)
  mapping_table <- mapping_table[!mapping_table$bfs_nr_new == 5399 |
                                   !is.na(mapping_table$bfs_nr_old), ]

  # Manually add the municipalities merging to form Verzasca in 2020
  verzasca_mergers <- data.frame(
    bfs_nr_new = rep(5399, 4),
    name_new = rep("Verzasca", 4),
    bfs_nr_old = c(5095, 5129, 5102, 5135),
    name_old = c("Brione (Verzasca)", "Sonogno", "Corippo", "Vogorno")
  )

  # Combine the manually added mergers with the existing mapping table
  mapping_table <- rbind(mapping_table, verzasca_mergers)

  # Correct NA entries for unchanged municipalities whose "old data" is missing
  na_indices <- which(is.na(mapping_table$bfs_nr_old))
  mapping_table[na_indices, "bfs_nr_old"] <- mapping_table[na_indices, "bfs_nr_new"]
  mapping_table[na_indices, "name_old"] <- mapping_table[na_indices, "name_new"]

  return(mapping_table)
}
```

```r
# Set up the MT-making function for 2020 and after
clean_mapping_table_maker_from_2020 <- function(mutations, old_state, new_state) {
  # Create a mapping table
  mapping_object <- map_old_to_new_state(mutations, old_state, new_state)
  mapping_table <- mapping_object$mapped

  # Define lakes to be removed
  lakes_to_remove <- c("Zürichsee (ZH)", "Zürichsee (SZ)", "Zürichsee (SG)",
                       "Thunersee", "Brienzersee", "Bielersee (BE)",
                       "Lac de Bienne (NE)", "Lac de Neuchâtel (BE)",
                       "Lac de Neuchâtel (FR)", "Lac de Neuchâtel (VD)",
                       "Lac de Neuchâtel (NE)", "Baldeggersee", "Sempachersee",
                       "Hallwilersee (LU)", "Hallwilersee (AG)",
                       "Zugersee (LU)",  "Zugersee (SZ)", "Zugersee (ZG)",
                       "Vierwaldstättersee (LU)", "Vierwaldstättersee (UR)",
                       "Vierwaldstättersee (SZ)", "Vierwaldstättersee (OW)",
                       "Vierwaldstättersee (NW)", "Sihlsee", "Sarnersee",
                       "Walensee (GL)", "Walensee (SG)", "Aegerisee",
```

```r
                       "Lac de la Gruyère", "Murtensee (FR)", "Lac de Morat (VD)",
                       "Bodensee (SH)", "Bodensee (SG)", "Bodensee (TG)",
                       "Lago di Lugano (o.Camp.)", "Lago Maggiore", "Lac de Joux",
                       "Lac Léman (VD)", "Lac Léman (VS)", "Lac Léman (GE)")

  # Remove the lakes from the mapping table
  mapping_table <- mapping_table[!mapping_table$name_old %in% lakes_to_remove, ]

  # Handle the special case for the last lake with the same name as a municipality,
  # as well as 3 erroneous municipalities
  mapping_table <- mapping_table[!mapping_table$bfs_nr_old
                                 %in% c(9040, 2391, 5391, 5394), ]

  # Correct NA entries for unchanged municipalities whose "old data" is missing
  na_indices <- which(is.na(mapping_table$bfs_nr_old))
  mapping_table[na_indices, "bfs_nr_old"] <- mapping_table[na_indices, "bfs_nr_new"]
  mapping_table[na_indices, "name_old"] <- mapping_table[na_indices, "name_new"]

  return(mapping_table)
}
```

Create all necessary MTs

```r
# MT for 2019 data
mapping_table_2019 <- clean_mapping_table_maker_pre_2020(mutations,
                                                         old_state_2019,
                                                         new_state_2024)

# # MT for 2020 data
mapping_table_2020 <- clean_mapping_table_maker_from_2020(mutations,
                                                          old_state_2020,
                                                          new_state_2024)

# MT for 2022 data
mapping_table_2022 <- clean_mapping_table_maker_pre_2020(mutations,
                                                         old_state_2022,
                                                         new_state_2024)

# MT for 2023 data
mapping_table_2023 <- clean_mapping_table_maker_pre_2020(mutations,
                                                         old_state_2023,
                                                         new_state_2024)
```

Investigate the presence of NAs in the MTs

```r
sum(is.na(mapping_table_2019))
```

```
## [1] 0
```

```r
sum(is.na(mapping_table_2020))
```

```
## [1] 0
```

```r
sum(is.na(mapping_table_2022))
```

```
## [1] 0
```

```r
sum(is.na(mapping_table_2023))
```

```
## [1] 0
```

```r
# No NA at all.
```

Simultaneously cleaning and updating data sets to 2024 boundaries

2019 data

```r
# 2019 :
Parties19_updated <- left_join(mapping_table_2019, Parties19, by = "bfs_nr_old") %>%
  mutate(Number_of_votes_19 = as.numeric(gsub("[[:punct:][:space:]]", "",
                                              Number_of_votes_19)),
         # Convert to numeric, remove punctuation
         Number_Liberal_votes_19 = as.numeric(gsub("[[:punct:][:space:]]", "",
                                              Number_Liberal_votes_19)), # idem
         Number_PDC_votes_19 = as.numeric(gsub("[[:punct:][:space:]]", "",
                                              Number_PDC_votes_19)), # idem
         Number_SP_votes_19 = as.numeric(gsub("[[:punct:][:space:]]", "",
                                              Number_SP_votes_19)),# idem
         Number_SPD_votes_19 = as.numeric(gsub("[[:punct:][:space:]]", "",
                                              Number_SPD_votes_19))) %>%  # idem
  group_by(bfs_nr_new) %>%
  summarise(Number_of_votes_19 = sum(Number_of_votes_19, na.rm = TRUE),
            Number_Liberal_votes_19 = sum(Number_Liberal_votes_19, na.rm = TRUE),
            Number_PDC_votes_19 = sum(Number_PDC_votes_19, na.rm = TRUE),
            Number_SP_votes_19 = sum(Number_SP_votes_19, na.rm = TRUE),
            Number_SPD_votes_19 = sum(Number_SPD_votes_19, na.rm = TRUE),
            .groups = "drop") %>%
  mutate(Liberal_percent_19 = Number_Liberal_votes_19 / Number_of_votes_19 * 100,
    PDC_percent_19 = Number_PDC_votes_19 / Number_of_votes_19 * 100,
    SP_percent_19 = Number_SP_votes_19 / Number_of_votes_19 * 100,
    SPD_percent_19 = Number_SPD_votes_19 / Number_of_votes_19 * 100) %>%
  select(-Number_Liberal_votes_19, -Number_PDC_votes_19, -Number_SP_votes_19,
         -Number_SPD_votes_19)
```

```
## Warning: There were 4 warnings in `mutate()`.
## The first warning was:
## i In argument: `Number_Liberal_votes_19 =
##   as.numeric(gsub("[[:punct:][:space:]]", "", Number_Liberal_votes_19))`.
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.
```

```r
Parties19_updated[is.na(Parties19_updated$bfs_nr_new), ] # 0
```

```
## # A tibble: 0 x 6
## # i 6 variables: bfs_nr_new <dbl>, Number_of_votes_19 <dbl>,
## #   Liberal_percent_19 <dbl>, PDC_percent_19 <dbl>, SP_percent_19 <dbl>,
## #   SPD_percent_19 <dbl>
```

```r
Parties19_updated[is.na(Parties19_updated$Number_of_votes_19), ] # 0 (also 0 votes)
```

```
## # A tibble: 0 x 6
```

```
## # i 6 variables: bfs_nr_new <dbl>, Number_of_votes_19 <dbl>,
## #   Liberal_percent_19 <dbl>, PDC_percent_19 <dbl>, SP_percent_19 <dbl>,
## #   SPD_percent_19 <dbl>
```

```
Parties19_updated[is.na(Parties19_updated$Liberal_percent_19), ] #5, x3 parties = 20
```

```
## # A tibble: 5 x 6
##   bfs_nr_new Number_of_votes_19 Liberal_percent_19 PDC_percent_19 SP_percent_19
##        <dbl>              <dbl>              <dbl>          <dbl>         <dbl>
## 1        389                  0                NaN            NaN           NaN
## 2        408                  0                NaN            NaN           NaN
## 3        422                  0                NaN            NaN           NaN
## 4        535                  0                NaN            NaN           NaN
## 5        877                  0                NaN            NaN           NaN
## # i 1 more variable: SPD_percent_19 <dbl>
```

```
sum(Parties19$Number_of_votes_19==0)
```

```
## [1] 0
```

```
# The reason why we these municipalities fail to appear is because there is no
# vote data on them. Consequently, their rows do not exist in the Parties19
# dataset and there is nothing to return for them.
# Importantly, these are the same 5 hamlets with between 50 and 500 citizens which
# vote within another municipality in the referendum, so
# they will not be in the 2024 referendum data either.

# Merging with the main 2024 data set
Data_2024_updated <- merge(Data_2024_updated, Parties19_updated, by = "bfs_nr_new")


# 4/5 NAs do disappear, municipality 422 remains. It is the only municipality
# with NAs in the data set so far. To avoid double counting the 2019 vote data of
# the residents of municipality 422, and because it is a single municipality in
# the entire data set, we exclude it.
Data_2024_updated <- Data_2024_updated %>%  filter(bfs_nr_new != 422)
sum(is.na(Data_2024_updated)) # 0, it worked.
```

```
## [1] 0
```

2020 (income) data

```
# 2020 (income) data
Income_20_updated <- left_join(mapping_table_2020, Income_20, by = "bfs_nr_old") %>%
  mutate(Number_of_taxpayers_cleaned = as.numeric(gsub("[[:punct:][:space:]]",
                                                       "", Number_of_taxpayers)),
         # Convert 'Number_of_taxpayers' column to numeric, remove punctuation
         Municipality_total_revenue_cleaned = as.numeric(gsub("[[:punct:][:space:]]",
                                     "", Municipality_total_revenue))) %>% #idem
  group_by(bfs_nr_new) %>%
  summarise(Number_of_taxpayers = sum(Number_of_taxpayers_cleaned, na.rm = TRUE),
            Municipality_total_revenue = sum(Municipality_total_revenue_cleaned,
                                             na.rm = TRUE),
            .groups = "drop") %>%
  mutate(Average_income = Municipality_total_revenue / Number_of_taxpayers) %>%
  select(-Municipality_total_revenue, -Number_of_taxpayers)
```

```r
# For more meaningful models, we re-express income in thousands
Income_20_updated$Average_income <- round(Income_20_updated$Average_income / 1000, 2)

# Merging with the main 2024 data set
Data_2024_updated <- merge(Data_2024_updated, Income_20_updated, by = "bfs_nr_new")

# Checking NAs: there should be less NAs than initially, as NAs were caused by data
# protection to preserve the privacy of residents of small municipality. These
# municipalities are precisely those most likely to merge.
sum(is.na(Income_20)) #150
```

```
## [1] 150
```

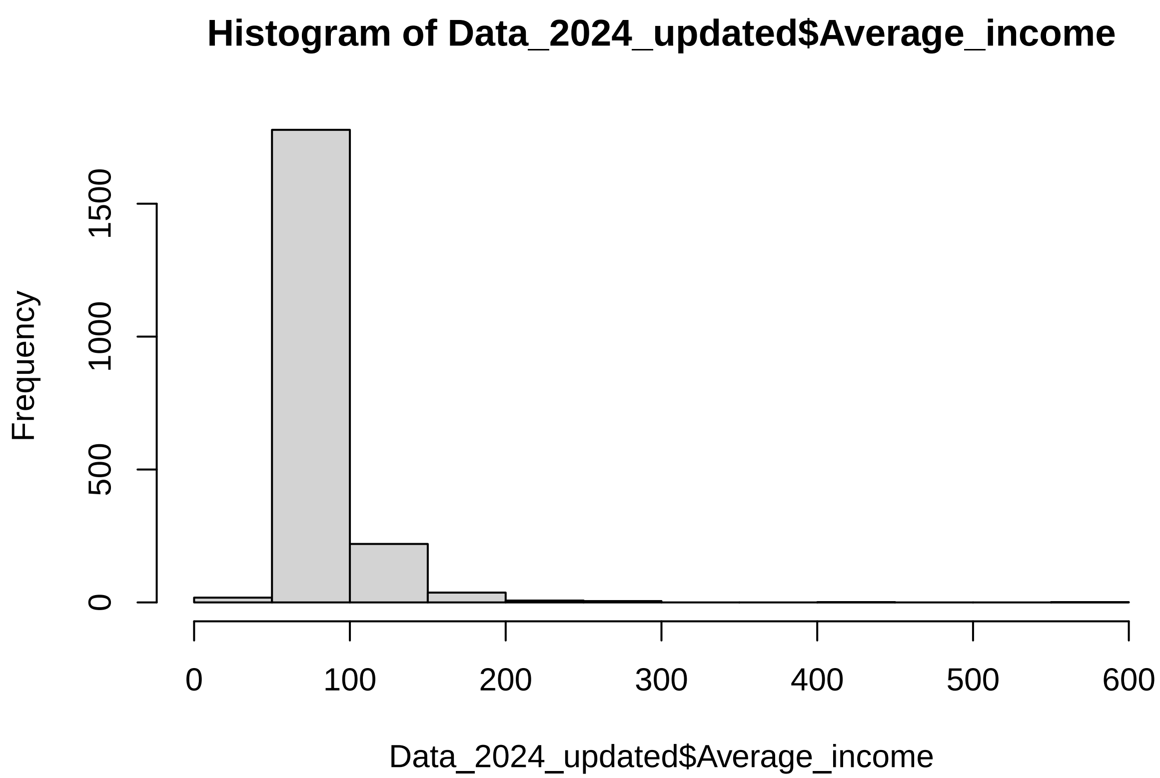
```r
sum(is.na(Data_2024_updated$Average_income)) # 58
```

```
## [1] 58
```

```r
# all in order



# We impute the NAs to avoid losing information.
hist(Data_2024_updated$Average_income)
```




```r
# The histogram shows that average income is extremely skewed. To minimise the
# influence of outliers, we impute based on the median average income of
# municipalities of the same type and which are in the same canton,
# maximizing similarities.
```

```r
# Calculate median income for each combination of Municipality_types and
# Cantonal_affiliation
median_incomes <- Data_2024_updated %>%
  group_by(Municipality_types, Cantonal_affiliation) %>%
  summarise(Median_Income = median(Average_income, na.rm = TRUE), .groups = 'drop')

# Join the median incomes back to the original dataset
Data_2024_updated <- Data_2024_updated %>%
  left_join(median_incomes, by = c("Municipality_types", "Cantonal_affiliation"))

# Impute missing Average_income values using Median_Income
Data_2024_updated <- Data_2024_updated %>%
  mutate(Average_income = ifelse(is.na(Average_income), Median_Income, Average_income))

# Remove the Median_Income column as no longer needed
Data_2024_updated <- select(Data_2024_updated, -Median_Income)

# Verify that the process worked:
sum(is.na(Data_2024_updated$Average_income))
```

```
## [1] 0
```

```r
# no NAs or unwanted 0's; confirming success.
```

2022 data

```r
Data_2022_updated <- left_join(mapping_table_2022, Data_2022, by = "bfs_nr_old") %>%
  mutate(Number_of_65_cleaned = as.numeric(gsub("[[:punct:][:space:]]",
                                     "", Number_of_65.)),
         # Convert 'Linguistic_region' column to numeric, remove punctuation
         Number_of_20_to_39_cleaned = as.numeric(gsub("[[:punct:][:space:]]",
         "", Number_of_20_to_39)) ) %>% #idem
  group_by(bfs_nr_new) %>%
  summarise(Linguistic_region = first(Linguistic_region), # attribute the first
            # linguistic region to which a merged municipality belongs to.
            Total_pop_2022 = sum(Total_pop_2022, na.rm = TRUE),
            Number_of_65 = sum(Number_of_65_cleaned, na.rm = TRUE),
            Number_of_20_to_39 = sum(Number_of_20_to_39_cleaned, na.rm = TRUE),
            .groups = "drop") %>%
  mutate(Percent_age_65_and_over = (Number_of_65 / Total_pop_2022)*100,
         Percent_age_20_39 = (Number_of_20_to_39 / Total_pop_2022)*100) %>%
  select(-Number_of_65, -Number_of_20_to_39)

# Using the first for allocation of linguistic regions should not be a problem
# since there are only 15 mergers between 2022 and 2024 (out of 2k+ municipalities)
# and municipalities only merge with those close to them, i.e., likely in the same
# linguistic region, so the method should also be farely accurate.

# Merging with the main 2024 data set
Data_2024_updated <- merge(Data_2024_updated, Data_2022_updated, by = "bfs_nr_new")

# Checking NAs
sum(is.na(Data_2022)) # 0
```

```
## [1] 0
```

```r
sum(is.na(Data_2024_updated$Total_pop_2022)) # 0
```

```
## [1] 0
```

```r
sum(is.na(Data_2024_updated$Percent_age_65_and_over)) # 0
```

```
## [1] 0
```

```r
sum(is.na(Data_2024_updated$Percent_age_20_39)) # 0
```

```
## [1] 0
```

```r
# all in order
```

2023 data - no cleaning needed

```r
Data_2023_updated <- left_join(mapping_table_2023, Data_2023, by = "bfs_nr_old") %>%
  group_by(bfs_nr_new) %>%
  summarise(Tax_burden_80k = mean(Tax_burden_80k, na.rm = TRUE),
            Tax_burden_150k = mean(Tax_burden_150k,na.rm = TRUE),
            .groups = "drop")

# Given we do not know the number of individuals earning 80k and 150k/municipality
# we cannot account for the relative size of merging municipalities.
setdiff(Data_2023$bfs_nr_old, Data_2024$bfs_nr_new)
```

```
## [1]  947  993 2456 4042 6773 6775
```

```r
# However, the above shows that using the mean should still be fairly accurate given
# only 6 municipalities merged between 2023-2024 (nbr:947, 993, 2456, 4042, 6773,
# 6775). Investigating them shows they are all hamlets with a population between
# 300 and 2500 for the largest, merging with other small municipalities in their
# rural area. Consequently, the value are not expected to differ largely.


# Checking NAs
sum(is.na(Data_2023)) # 0
```

```
## [1] 0
```

```r
sum(is.na(Data_2024_updated$Tax_burden_80k)) # 0
```

```
## [1] 0
```

```r
sum(is.na(Data_2024_updated$Tax_burden_150k)) # 0
```

```
## [1] 0
```

```r
# all in order. Inspecting the dataframe, also no inadvertently introduced 0's.

# Merging with the main 2024 data set
Data_2024_updated <- merge(Data_2024_updated,
                           Data_2023_updated, by = "bfs_nr_new")

# Finally NA check:
sum(is.na(Data_2024_updated)) # No NA at all.
```

```
## [1] 0
```

In anticipation of logit and random forest models, convert dependent variable percentage to a binary yes/no approval of the referendum.

```
Data_2024_updated$Referendum_pass <-
  ifelse(Data_2024_updated$Referendum_yes_percent > 50.0, 1, 0)
```

In anticipation of python logit-regressions requiring numerical inputs, we perform one-hot encoding on the linguistic region variable

```
Data_2024_updated <- Data_2024_updated %>%
  mutate(Linguistic_region = factor(Linguistic_region, levels =
                                      c("French", "German", "Italian",
                                        "Romansh"))) %>%
  cbind(model.matrix(~ Linguistic_region - 1, data = .)) %>%
  rename(`French_Region` = `Linguistic_regionFrench`,
         `German_Region` = `Linguistic_regionGerman`,
         `Italian_Region` = `Linguistic_regionItalian`,
         `Romansh_Region` = `Linguistic_regionRomansh`)


# Safety check:Cross-tabulate Linguistic_region with the new dummy variables
table(Data_2024_updated$Linguistic_region, Data_2024_updated$French_Region)
```

```
##
##              0    1
##   French     0  615
##   German  1374    0
##   Italian  121    0
##   Romansh   15    0
```

```
table(Data_2024_updated$Linguistic_region, Data_2024_updated$German_Region)
```

```
##
##              0    1
##   French   615    0
##   German     0 1374
##   Italian  121    0
##   Romansh   15    0
```

```
table(Data_2024_updated$Linguistic_region, Data_2024_updated$Italian_Region)
```

```
##
##              0    1
##   French   615    0
##   German  1374    0
##   Italian    0  121
##   Romansh   15    0
```

```
table(Data_2024_updated$Linguistic_region, Data_2024_updated$Romansh_Region)
```

```
##
##              0    1
##   French   615    0
##   German  1374    0
```

```
##   Italian  121    0
##   Romansh    0   15
```

```r
# Perfect match.

sum(is.na(Data_2024_updated)) # Still no NA.
```

```
## [1] 0
```

Similarly, one-hot encode the Municipality_types variable

```r
Data_2024_updated <- Data_2024_updated %>%
  mutate(Municipality_types = trimws(Municipality_types)) %>%
  mutate(Municipality_types = factor(Municipality_types, levels = c(
    "Urban municipality of a large conurbation",
    "Urban community in a medium-sized conurbation",
    "Small urban community or outside urban area",
    "Low-density suburban community",
    "Medium-density suburban community",
    "High-density suburban community",
    "Centrally located rural municipality",
    "Municipality in a rural center",
    "Rural outskirts"
  )))%>%
  cbind(model.matrix(~ Municipality_types - 1, data = .)) %>%
  rename(
    `Urban_municipality_large_conurbation` =
      `Municipality_typesUrban municipality of a large conurbation`,
    `Urban_municipality_medium_conurbation` =
      `Municipality_typesUrban community in a medium-sized conurbation`,
    `Small_or_outside_urban_municipality` =
      `Municipality_typesSmall urban community or outside urban area`,
    `Low_density_suburban` =
      `Municipality_typesLow-density suburban community`,
    `Medium_density_suburban` =
      `Municipality_typesMedium-density suburban community`,
    `High_density_suburban` =
      `Municipality_typesHigh-density suburban community`,
    `Centrally_located_rural` =
      `Municipality_typesCentrally located rural municipality`,
    `Rural_center` =
      `Municipality_typesMunicipality in a rural center`,
    `Rural_outskirts` = `Municipality_typesRural outskirts`
  )

# For verification, cross-tabulate Municipality_types with each new dummy
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Urban_municipality_large_conurbation)
```

```
##
##                                                   0   1
##   Urban municipality of a large conurbation       0 174
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                430   0
```

```
##   Medium-density suburban community            374   0
##   High-density suburban community              123   0
##   Centrally located rural municipality         388   0
##   Municipality in a rural center                88   0
##   Rural outskirts                              211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Urban_municipality_medium_conurbation)
```

```
##
##                                                  0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation   0 202
##   Small urban community or outside urban area   135   0
##   Low-density suburban community               430   0
##   Medium-density suburban community            374   0
##   High-density suburban community              123   0
##   Centrally located rural municipality         388   0
##   Municipality in a rural center                88   0
##   Rural outskirts                              211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Small_or_outside_urban_municipality)
```

```
##
##                                                  0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area     0 135
##   Low-density suburban community               430   0
##   Medium-density suburban community            374   0
##   High-density suburban community              123   0
##   Centrally located rural municipality         388   0
##   Municipality in a rural center                88   0
##   Rural outskirts                              211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Low_density_suburban)
```

```
##
##                                                  0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                 0 430
##   Medium-density suburban community            374   0
##   High-density suburban community              123   0
##   Centrally located rural municipality         388   0
##   Municipality in a rural center                88   0
##   Rural outskirts                              211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Medium_density_suburban)
```

```
##
##                                                  0   1
##   Urban municipality of a large conurbation     174   0
```

```
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                430   0
##   Medium-density suburban community               0 374
##   High-density suburban community               123   0
##   Centrally located rural municipality          388   0
##   Municipality in a rural center                 88   0
##   Rural outskirts                               211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$High_density_suburban)
```

```
##
##                                                   0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                430   0
##   Medium-density suburban community             374   0
##   High-density suburban community                 0 123
##   Centrally located rural municipality          388   0
##   Municipality in a rural center                 88   0
##   Rural outskirts                               211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Centrally_located_rural)
```

```
##
##                                                   0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                430   0
##   Medium-density suburban community             374   0
##   High-density suburban community               123   0
##   Centrally located rural municipality            0 388
##   Municipality in a rural center                 88   0
##   Rural outskirts                               211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Rural_center)
```

```
##
##                                                   0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                430   0
##   Medium-density suburban community             374   0
##   High-density suburban community               123   0
##   Centrally located rural municipality          388   0
##   Municipality in a rural center                  0  88
##   Rural outskirts                               211   0
```

```r
table(Data_2024_updated$Municipality_types,
      Data_2024_updated$Rural_outskirts)
```

```
##
##                                                      0   1
##   Urban municipality of a large conurbation     174   0
##   Urban community in a medium-sized conurbation 202   0
##   Small urban community or outside urban area   135   0
##   Low-density suburban community                430   0
##   Medium-density suburban community             374   0
##   High-density suburban community               123   0
##   Centrally located rural municipality          388   0
##   Municipality in a rural center                 88   0
##   Rural outskirts                                 0 211
```

```r
# All good.
```

To mitigate the curse of dimensionality, merge subcategories of urban, suburban, and rural municipality dummies (merging 3 sub-categories)

```r
Data_2024_updated <- Data_2024_updated %>%
  mutate(
    Urban_Municipality = Urban_municipality_large_conurbation +
      Urban_municipality_medium_conurbation +
      Small_or_outside_urban_municipality,
    Suburban_Municipality = Low_density_suburban +
      Medium_density_suburban +
      High_density_suburban,
    Rural_Municipality = Centrally_located_rural +
      Rural_center +
      Rural_outskirts
  )


sum(is.na(Data_2024_updated)) # Still no NA.
```

```
## [1] 0
```

Create a clean data set with only the necessary variables (limiting model complexity)

```r
ML_final_clean <- Data_2024_updated %>%
  select(-Municipality_types, -Cantonal_affiliation, -Referendum_yes_percent,
         -Number_of_votes_19, -Linguistic_region, -Urban_municipality_large_conurbation,
         -Urban_municipality_medium_conurbation, -Small_or_outside_urban_municipality,
         -Low_density_suburban, -Medium_density_suburban, -High_density_suburban,
         -Centrally_located_rural, -Rural_center, -Rural_outskirts) %>%
  select(bfs_nr_new, Municipality_name, Referendum_pass, everything()) # Reordering
```

Export the data to use in python for the remainder of the assessment.

```r
# Cleaned data set (remove the next # to export)
# write.csv(ML_final_clean, "ML_final_data_clean.csv", row.names = FALSE)
# End of R section of the assessment.
```