Natural Disasters in France (Clay Soils Movement)

HarvardX - Data Science : Capstone

Meryll Mercadier

2025-05-04

Summary

1 Preface

This project is an assessment, part of the online course HarvardX: Data Science: Capstone from Harvard University on Edx platform.
The HarvardX: Data Science: Capstone" is the last part of a global Professional Certificate Program in Data Science.
The purpose of this project is to autonomously put into practice all the knowledge learned during the past courses.
The following code will be highly commented in order to facilitate its readability and evaluation.

As English is not my native language, part of the comments meaning can be lost due to a wrong translation. Please take it in consideration by evaluating this project.

2 Introduction

2.1 Context

Superficial clay soils undergo volume variations depending on their water content, which is influenced by weather conditions. During drought periods, they shrink, while they swell when the rains return. Although these changes occur gradually, their magnitude can cause damage to buildings located on such soils.

This shrink-swell phenomenon of clays is responsible for numerous damages each year, primarily affecting individual houses.

In France, municipalities can request recognition of natural disasters from the government.
When the government declares a state of natural disaster in a municipality, affected individuals become eligible for compensation through their insurance.

2.2 Project

Based on public data, can we make predictions about the recognition and requests for natural disasters related to clay movements in French municipalities?

2.3 Data

By reading the information note on the risks of clay shrinkage and swelling on the GEORISQUE website, we obtain all the information allowing us to define the data we need.
Click here for more informartion.

To carry out this project, we therefore need the following data:

- A history of requests and acknowledgements of natural disasters linked to clay soil movements.
- Mapping of clay soils in France.
- All soil drought histories (or weather data that can induce droughts).
- All soil moisture histories (or proxies from meteorological data).

3 Data collection and preparation

Here are the public databases we will use to carry out this project :

- History of requests and acknowledgement of "natural disasters" per ciy, linked to clay movements.
  Imported from "www.data.gouv.fr" (Open platform for French public data), from the following link :
  "https://www.data.gouv.fr/fr/datasets/sols-argileux-et-catastrophes-naturelles/"
- Cities exposure to climate risks in 2016
  Imported from "www.statistiques.developpement-durable.gouv.fr", from the following link :
  "https://www.statistiques.developpement-durable.gouv.fr/risques-climatiques-six-francais-sur-dix-sont-dores-et-deja-concernes"
- "Map of exposure to the phenomenon of shrinkage-swelling of clays covers metropolitan France (excluding the city of Paris)"
  Imported from "www.georisques.gouv.fr", from the following link :
  https://www.georisques.gouv.fr/donnees/bases-de-donnees/retrait-gonflement-des-argiles
- GPS positions of all French cities
  Imported from "www.data.gouv.fr", from the following link :
  https://www.data.gouv.fr/fr/datasets/villes-de-france/#/community-resources
- Monthly basic climatological data by department (from 1950 to 2023)
  Imported from "www.data.gouv.fr" (data collected by "Météo-France"), from the following link:
  https://meteo.data.gouv.fr/datasets/6569b3d7d193b4daf2b43edc
- Population size per city per year
  Imported from "www.insee.fr", from the following link:
  https://www.insee.fr/fr/statistiques/3698339
- History of droughts by municipality
  Imported from "www.data.gouv.fr", from the following link:
  https://www.data.gouv.fr/fr/datasets/donnee-secheresse-vigieau/#/resources

3.1 Importing the History of requests and acknowledgement of "natural disasters"

```
# ->
# 'https://www.data.gouv.fr/fr/datasets/sols-argileux-et-catastrophes-naturelles/'
data_base_1 <- "data_base_1"
if (!file.exists(data_base_1)) download.file(
↪  "https://www.ccomptes.fr/sites/default/files/2022-02/20220215-sols-argileux-catastrophes-naturelles
↪  ,
    data_base_1)
```

Preparing the table : Natural Disasters by City

```
nat_dis_city <- "D_Base arrêtés Cat Nat brute.csv"
if (!file.exists(nat_dis_city)) unzip(data_base_1,
    nat_dis_city)

nat_dis_city <- as.data.frame(str_split(read_lines(nat_dis_city),
    fixed(";"), simplify = TRUE), stringsAsFactors = FALSE)
```

```r
# Adding colnames
colnames(nat_dis_city) <- nat_dis_city[1, ]
nat_dis_city <- nat_dis_city[-1, ]
```

Keeping only usefull columns

```r
nat_dis_city <- nat_dis_city %>%
    select("N° Insee", "Département", "Nom de la commune",
        "Année évènement", "Décision")
head(nat_dis_city)
```

```
  N° Insee Département Nom de la commune Année évènement     Décision
2     1004          01 AMBERIEU EN BUGEY            2009 Non reconnue
3     1004          01 AMBERIEU EN BUGEY            2018     Reconnue
4     1007          01          AMBRONAY            2015 Non reconnue
5     1007          01          AMBRONAY            2018     Reconnue
6     1012          01             ARANC            2018     Reconnue
7     1012          01             ARANC            2019 Non reconnue
```

Translating the column names to English (Nat_dis stand for “Natural Disaster”)

```r
colnames(nat_dis_city) <- c("insee_code", "department",
    "city", "year", "is_nat_dis")
```

We replace characters with binary data and create the column demand

```r
nat_dis_city <- nat_dis_city %>%
    mutate(is_nat_dis = ifelse(is_nat_dis == "Reconnue",
        1, 0), demand = 1, year = as.numeric(year))
```

As the available weather conditions database are grouped by department,
to facilitate the treatment in this study,
we will only focus on the departments of metropolitan France (from 1 to 96)

```r
colnames(nat_dis_city)
```

```
[1] "insee_code" "department" "city"       "year"       "is_nat_dis"
[6] "demand"
```

```r
studied_dep <- data.frame(department = seq(1:96))
studied_dep <- studied_dep %>%
    mutate(department = as.character(ifelse(nchar(department) <
        2, paste(0, department, sep = ""), department)))
```

It seem’s we have several lines for a same city, so we keep only one row per city

```r
nat_dis_city <- nat_dis_city %>%
    mutate(city_year = paste(insee_code, "-", year)) %>%
    group_by(insee_code, year, city_year) %>%
    summarise(demand = ifelse(sum(demand) > 0, 1, 0),
        nat_dis = ifelse(sum(is_nat_dis) > 0, 1, 0),
        .groups = "drop")
```

3.2 Importing the cities gps position

Downloading public database containing gps positions of all French cities (https://www.data.gouv.fr/fr/datasets/villes-de-france/#/community-resources)

```
cities_gps <- "cities_gps"
if (!file.exists(cities_gps)) download.file(
↪  "https://www.data.gouv.fr/fr/datasets/r/51606633-fb13-4820-b795-9a2a575a72f1",
    cities_gps)

cities_gps <- as.data.frame(str_split(read_lines(cities_gps),
    fixed(","), simplify = TRUE), stringsAsFactors = FALSE)
colnames(cities_gps) <- cities_gps[1, ]
cities_gps <- cities_gps[-1, ]
cities_gps <- cities_gps[!duplicated(cities_gps$insee_code),
    ]  #Keeping only unique values

head(cities_gps)
```

```
  insee_code           city_code zip_code               label     latitude
2      25620       ville du pont    25650       ville du pont 46.999873398
3      25624      villers grelot    25640      villers grelot 47.361512085
4      25615 villars les blamont    25310 villars les blamont 47.368383721
5      25619       les villedieu    25240       les villedieu 46.713906258
6      25622       villers buzon    25170       villers buzon 47.228558434
7      25625    villers la combe    25510    villers la combe 47.240809828
    longitude department_name department_number             region_name
2 6.498147193           doubs                25 bourgogne-franche-comté
3 6.235167025           doubs                25 bourgogne-franche-comté
4 6.871414913           doubs                25 bourgogne-franche-comté
5  6.26583065           doubs                25 bourgogne-franche-comté
6 5.852186748           doubs                25 bourgogne-franche-comté
7 6.473842387           doubs                25 bourgogne-franche-comté
      region_geojson_name
2 Bourgogne-Franche-Comté
3 Bourgogne-Franche-Comté
4 Bourgogne-Franche-Comté
5 Bourgogne-Franche-Comté
6 Bourgogne-Franche-Comté
7 Bourgogne-Franche-Comté
```

Insee_code correction in “nat_dis_city” for good matching with “citites_gps”

```
nat_dis_city <- nat_dis_city %>%
    mutate(insee_code = as.character(ifelse(nchar(insee_code) <
        5, paste(0, insee_code, sep = ""), insee_code)))
```

Checking if some cities are in natural disaster dataframe and not in our gps dataframe :

```
nat_dis_city %>%
    filter(!insee_code %in% cities_gps$insee_code) %>%
    pull(insee_code) %>%
    n_distinct()
```

```
[1] 175
```

By analyzing our "nat_dis_city" we can see that 175 cities are are not in gps dataframe.
this probably means that those cities changed their name after the meteorological sampling
To correct this we could replace those cities by the next city according the zip_code order But to go faster in this project we will only filter them out.

```
nrow(nat_dis_city)
```

```
[1] 28136
```

```
nat_dis_city <- nat_dis_city %>%
    filter(insee_code %in% cities_gps$insee_code)
nrow(nat_dis_city)
```

```
[1] 27828
```

Checking if all cities are in our natural disaster dataframe are missing in the cities gps dataframe:

```
nat_dis_city %>%
    filter(!insee_code %in% cities_gps$insee_code) %>%
    pull(insee_code) %>%
    n_distinct()
```

```
[1] 0
```

No cities are missing.

Creating rows when cities didn't make the request for acknowledgement of natural disaster.

```
cities_gps <- cities_gps %>%
    filter(department_number %in% studied_dep$department)  # Filtering cities_gps on
    ↪  department studied here
cities <- data.frame(insee_code = unique(cities_gps$insee_code))  # Creating unique
↪  cities dataframe
years <- data.frame(years = seq(from = min(nat_dis_city$year),
    to = max(nat_dis_city$year), by = 1))  #Creating unique years dataframe
```

Creating all possible request

```
all_possible_demands <- expand.grid(cities$insee_code,
    years$years)
head(all_possible_demands)
```

```
   Var1 Var2
1 25620 2004
2 25624 2004
3 25615 2004
4 25619 2004
5 25622 2004
6 25625 2004
```

```
head(nat_dis_city)
```

```
# A tibble: 6 x 5
  insee_code  year city_year    demand nat_dis
```

```
  <chr>      <dbl> <chr>         <dbl>   <dbl>
1 10010       2018 10010 - 2018      1       1
2 10010       2019 10010 - 2019      1       1
3 10018       2018 10018 - 2018      1       1
4 10018       2019 10018 - 2019      1       1
5 10018       2020 10018 - 2020      1       1
6 10024       2020 10024 - 2020      1       1
```

```
all_possible_demands <- all_possible_demands %>%
    mutate(insee_code = Var1, year = Var2, city_year = paste(Var1,
        "-", Var2), demand = 0, nat_dis = 0) %>%
    select(insee_code, year, city_year, demand, nat_dis) %>%
    filter(!city_year %in% nat_dis_city$city_year)
```

Creating our main table (df) by adding "all_possible_demands" to "nat_dis_city"

```
df <- union(nat_dis_city, all_possible_demands) %>%
    select(-city_year)
```

3.3 Importing the historical weather conditions

Now let's download the weather data for each of the french departments In this report, we will only display the first three data downloads, but you can find the complete list of downloads in the original code.

```
# Department 57
data_weather_57 <- "data_weather_57"
if (!file.exists(data_weather_57)) download.file(
↪  "https://www.data.gouv.fr/fr/datasets/r/b4c28470-3957-4e2c-82bf-dee0b972d770",
    data_weather_57)
weather_57 <- as.data.frame(str_split(read_lines(data_weather_57),
    fixed(";"), simplify = TRUE), stringsAsFactors = FALSE)
colnames(weather_57) <- weather_57[1, ]
weather_57 <- weather_57[-1, ]

# Department 03
data_weather_03 <- "data_weather_03"
if (!file.exists(data_weather_03)) download.file(
↪  "https://www.data.gouv.fr/fr/datasets/r/294c43cd-9071-4ef0-bc8b-5080c5401960",
    data_weather_03)
weather_03 <- as.data.frame(str_split(read_lines(data_weather_03),
    fixed(";"), simplify = TRUE), stringsAsFactors = FALSE)
colnames(weather_03) <- weather_03[1, ]
weather_03 <- weather_03[-1, ]

# Department 71
data_weather_71 <- "data_weather_71"
if (!file.exists(data_weather_71)) download.file(
↪  "https://www.data.gouv.fr/fr/datasets/r/46732f42-be77-41ce-971e-65ff3d63103f",
    data_weather_71)
weather_71 <- as.data.frame(str_split(read_lines(data_weather_71),
    fixed(";"), simplify = TRUE), stringsAsFactors = FALSE)
colnames(weather_71) <- weather_71[1, ]
weather_71 <- weather_71[-1, ]
```

Joining all departments in one dataframe

```
weather_list <- mget(ls(pattern = "^weather_\\d{2}$"))  # Selecting and creating a list
↪  of all objects 'weather_..'
weather <- bind_rows(weather_list)  # Joining all weather objects

# Keeping only interesting columns and renaming
# them
weather <- weather %>%
    select(NUM_POSTE, LAT, LON, ALTI, AAAAMM, RR, QRR,
        NBRR, NBJRR1, NBJRR5, NBJRR10, NBJRR30, NBJRR50,
        NBJRR100, TX, QTX, TMM, QTMM, NBTX, NBJTX0,
        NBJTX25, NBJTX30, NBJTX35, NBJTXI20, NBJTXI27,
        NBJTXS32, TN, QTN, NBTN, NBJTN5, NBJTN10, UMM,
        QUMM, UNAB, QUNAB, UXAB, QUXAB, TSVM, QTSVM,
        ETP, QETP, GLOT, QGLOT)
```

Here is the column detail :
- RR = cumulative monthly precipitation
- QRR = quality of precipitation measurement
- NBRR = number of days with precipitation
- NBJRR1 = Number of days with precipitation of 1 mm
- NBJRR5 = Number of days with precipitation of 5 mm
- NBJRR10 = Number of days with precipitation of 10 mm
- NBJRR30 = Number of days with precipitation of 30 mm
- NBJRR50 = Number of days with precipitation of 50 mm
- NBJRR100 = Number of days with precipitation of 100 mm
- TX = Average maximum temperature of the month (in °C)
- QTX = Quality of maximum temperature measurement
- NBTX = Number of days with a maximum temperature measurement
- NBJTX0 = Number of days with maximum temperatures under or equal to 0°C
- NBJTX25 = Number of days with maximum temperatures exceeding 25°C
- NBJTX30 = Number of days with maximum temperatures exceeding 30°C
- NBJTX35 = Number of days with maximum temperatures exceeding 35°C
- NBJTXI20 = Number of days with maximum temperatures under or equal to 20°C
- NBJTXI27 = Number of days with maximum temperatures under or equal to 27°C
- NBJTXS32 = Number of days with maximum temperatures exceeding 32°C
- TN = Average minimum temperature of the month (in °C)
- QTN = Quality of minimum temperature measurement
- NBTN = Number of days with a minimum temperature measurement
- NBJTN5 = Number of days with minimum temperatures below -5°C
- NBJTN10 = Number of days with minimum temperatures below -10°C
- UMM = Monthly average of daily average humidity (UM) (in %)
- QUMM = Quality of UMM measurement
- UNAB = Absolute monthly minimum of daily minimum relative humidity (UN) (in %)
- QUNAB = Quality of UNAB measurement
- UXAB = Absolute monthly maximum of daily maximum relative humidity (UX) (in %)
- QUXAB = Quality of UXAB measurement
- TSVM = Monthly average of vapor pressure (in hPa and 1/10)
- QTSVM = Quality of QTSVM measurement
- ETP = ETP: sum of decadal Penman ETPs (potential evapotranspiration) (in mm and 1/10)
- QETP = Quality of ETP measurement

- GLOT = Monthly cumulative daily global radiation (in J/cm2)
- QGLOT = Quality of GLOT measurement

```
colnames(weather) <- c("NUM_POSTE", "latitude", "longitude",
    "altitude", "Year_Month", "RR", "QRR", "NBRR",
    "NBJRR1", "NBJRR5", "NBJRR10", "NBJRR30", "NBJRR50",
    "NBJRR100", "TX", "QTX", "TMM", "QTMM", "NBTX",
    "NBJTX0", "NBJTX25", "NBJTX30", "NBJTX35", "NBJTXI20",
    "NBJTXI27", "NBJTXS32", "TN", "QTN", "NBTN", "NBJTN5",
    "NBJTN10", "UMM", "QUMM", "UNAB", "QUNAB", "UXAB",
    "QUXAB", "TSVM", "QTSVM", "ETP", "QETP", "GLOT",
    "QGLOT")
```

We remove the duplicates to avoid any errors.

```
weather <- weather[!duplicated(weather[c("NUM_POSTE",
    "Year_Month")]), ]
```

Creating columns for “year” and “month”

```
weather <- weather %>%
    mutate(year = substr(weather$Year_Month, 1, 4),
        month = substr(weather$Year_Month, 5, 6)) %>%
    select(-Year_Month)
weather <- weather %>%
    mutate(year = as.numeric(year), month = as.numeric(month))
```

We see that the period covered by our weather dataframe is much bigger than in our main dataframe, so we will filter on the period to study.

```
weather <- weather %>%
    filter(year > 2004 & year < 2021)
```

Matching each city from “df”, to the closest post from “weather”
To calculate the distance between 2 gps points we will use the Harvesine formula as following:
Distance(km)<-6371 x 2 x asin(sqrt(sin((Lat2-Lat1) x pi/180/2)^2+cos(Lat1 x pi/180)*cos(Lat2 x pi/180) x sin((Long2-Long1) x pi/180/2)^2))

Are the weather post working every year ?

```
weather %>%
    filter(NUM_POSTE == weather[10112, 1]) %>%
    group_by(year) %>%
    summarise()
```

```
# A tibble: 8 x 1
   year
  <dbl>
1  2005
2  2006
3  2007
4  2008
```

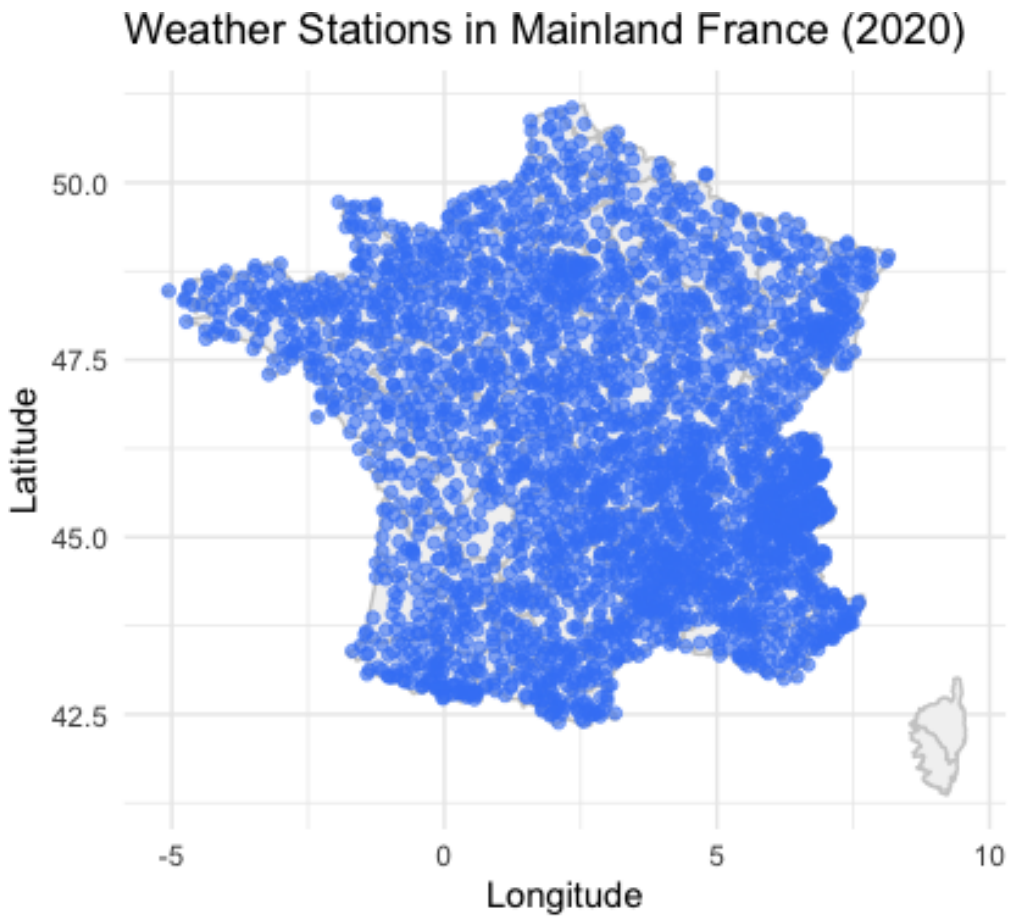
```
5  2009
6  2010
7  2011
8  2012
```

For example we see that the 10112th Post of our dataframe has been only working from 2005 to 2012.
If we want to match cities with the closest weather post, we will so need to take in consideration the year concerned.

```r
unique_cities_year_df <- df %>%
    filter(year > 2004 & year < 2021, insee_code %in%
        cities_gps$insee_code) %>%
    distinct(insee_code, year) %>%
    left_join(cities_gps, by = "insee_code", relationship = "many-to-many") %>%
    select(insee_code, year, latitude, longitude) %>%
    mutate(latitude = as.numeric(latitude), longitude = as.numeric(longitude))

unique_poste_weather <- weather %>%
    filter(year > 2004 & year < 2021) %>%
    distinct(NUM_POSTE, year, longitude, latitude) %>%
    mutate(latitude = as.numeric(latitude), longitude = as.numeric(longitude))
```

Let's check on a 2D map of France,
whether we have successfully retrieved all the weather stations in mainland France

```r
#Filtering year 2020 for exemple
stations_2020 <- unique_poste_weather %>%
  filter(year == 2020)

# 2D map creation
ggplot(stations_2020, aes(x = as.numeric(longitude), y = as.numeric(latitude))) +
  borders("france", colour = "gray80", fill = "gray95")+  # Map
  geom_point(color = "#4285f6", alpha = 0.6, size = 1.5)+    # Circle markers
  coord_fixed(1.3) +  # latitude/longitude proportion
  theme_minimal() +
  labs(
    title = "Weather Stations in Mainland France (2020)",
    x = "Longitude",
    y = "Latitude"
  )
```



The map shows good data coverage

Matching the closest weather station for each city/year.
(!This line of code take around 2 minutes to run!)

```
nearest_poste <- map_df(unique(unique_cities_year_df$year),
    function(a) {
        unique_cities_year_df <- unique_cities_year_df %>%
            filter(year == a)
        unique_poste_weather <- unique_poste_weather %>%
            filter(year == a)
        distances_matrix <- geodist(unique_cities_year_df[,
            c("longitude", "latitude")], unique_poste_weather[,
            c("longitude", "latitude")], measure = "haversine")/1000

        unique_cities_year_df %>%
            mutate(NUM_POSTE = unique_poste_weather[apply(distances_matrix,
                1, which.min), 1], distance_km = apply(distances_matrix,
                1, min))
    })
```

Creatin our dataframe (df1) by adding the nearest weather station to df.

```
nearest_poste <- nearest_poste %>%
    select(-latitude, -longitude)
df1 <- df %>%
    inner_join(nearest_poste, by = c("insee_code",
        "year"), relationship = "many-to-many")
```

Wrangling “weather” dataframe to integrate it in the main dataframe “df”
Transforming all predictors as numeric to facilitate the treatment

```
weather <- weather %>%
    mutate_at(2:ncol(weather), as.numeric) %>%
    mutate_at(1, as.character)
```

The quality of the measurement is evaluated as following:
9: filtered data (the data has passed the first level filters/checks)
0: protected data (the data has been definitively validated by the climatologist)
1: validated data (the data has been validated by automatic check or by the climatologist)
2: doubtful data in the process of being verified (the data has been called into question by automatic check)

You can find further information on the following link : https://donneespubliques.meteofrance.fr/client/document/mensq_descriptif_champs_323.csv

To facilitate the readability we will change the notation as following
0 (ex 0): protected data (the data has been definitively validated by the climatologist)
1 (ex 1): validated data (the data has been validated by automatic check or by the climatologist)
2 (ex 9): filtered data (the data has passed the first level filters/checks)
3 : unmeasured quality (NA)
4 (ex 2): doubtful data in the process of being verified (the data has been called into question by automatic check)

```
weather1 <- weather %>%
    mutate(QRR = ifelse(QRR == 2, 4, ifelse(QRR ==
        9, 2, QRR)), QTX = ifelse(QTX == 2, 4, ifelse(QTX ==
        9, 2, QTX)), QTN = ifelse(QTN == 2, 4, ifelse(QTN ==
        9, 2, QTN)), QTMM = ifelse(QTMM == 2, 4, ifelse(QTMM ==
        9, 2, QTMM)), QUMM = ifelse(QUMM == 2, 4, ifelse(QUMM ==
        9, 2, QUMM)), QUNAB = ifelse(QUNAB == 2, 4,
        ifelse(QUNAB == 9, 2, QUNAB)), QUXAB = ifelse(QUXAB ==
        2, 4, ifelse(QUXAB == 9, 2, QUXAB)), QTSVM = ifelse(QTSVM ==
        2, 4, ifelse(QTSVM == 9, 2, QTSVM)), QETP = ifelse(QETP ==
        2, 4, ifelse(QETP == 9, 2, QETP)), QGLOT = ifelse(QGLOT ==
        2, 4, ifelse(QGLOT == 9, 2, QGLOT)))
weather1$QRR[is.na(weather$QRR)] <- 3
weather1$QTX[is.na(weather$QTX)] <- 3
weather1$QTN[is.na(weather$QTN)] <- 3
weather1$QTMM[is.na(weather$QTMM)] <- 3
weather1$QUMM[is.na(weather$QUMM)] <- 3
weather1$QUNAB[is.na(weather$QUNAB)] <- 3
weather1$QUXAB[is.na(weather$QUXAB)] <- 3
weather1$QTSVM[is.na(weather$QTSVM)] <- 3
weather1$QETP[is.na(weather$QETP)] <- 3
```

```
weather1$QGLOT[is.na(weather$QGLOT)] <- 3
```

Grouping data per years

```
weather_year <- weather1 %>%
    group_by(NUM_POSTE, year) %>%
    summarise(cumul_precipit = sum(RR), qlty_precipit = mean(QRR,
        na.rm = TRUE), n_day_1mm_precipit = sum(NBJRR1,
        na.rm = TRUE), n_day_5mm_precipit = sum(NBJRR5,
        na.rm = TRUE), n_day_10mm_precipit = sum(NBJRR10,
        na.rm = TRUE), n_day_30mm_precipit = sum(NBJRR30,
        na.rm = TRUE), n_day_50mm_precipit = sum(NBJRR50,
        na.rm = TRUE), n_day_100mm_precipit = sum(NBJRR100,
        na.rm = TRUE), mean_temp = mean(TMM, na.rm = TRUE),
        qlty_mean_temp = mean(QTMM, na.rm = TRUE),
        n_day_below_0 = sum(NBJTX0, na.rm = TRUE),
        n_day_above_25 = sum(NBJTX25, na.rm = TRUE),
        n_day_above_30 = sum(NBJTX30, na.rm = TRUE),
        n_day_above_35 = sum(NBJTX35, na.rm = TRUE),
        n_day_below_5 = sum(NBJTN5, na.rm = TRUE),
        n_day_below_10 = sum(NBJTN10, na.rm = TRUE),
        n_day_below_20 = sum(NBJTXI20, na.rm = TRUE),
        n_day_below_27 = sum(NBJTXI27, na.rm = TRUE),
        n_day_above_32 = sum(NBJTXS32, na.rm = TRUE),
        mean_temp_spring = mean(TMM[month %in% c(3,
            4, 5)], na.rm = TRUE), mean_temp_summer = mean(TMM[month %in%
            c(6, 7, 8)], na.rm = TRUE), mean_temp_autumn = mean(TMM[month %in%
            c(9, 10, 11)], na.rm = TRUE), mean_temp_winter = mean(TMM[month %in%
            c(12, 1, 2)], na.rm = TRUE), mean_humid = mean(UMM,
            na.rm = TRUE), qlty_mean_humid = mean(QUMM,
            na.rm = TRUE), mean_min_humid = mean(UNAB,
            na.rm = TRUE), qlty_mean_min_humid = mean(QUNAB,
            na.rm = TRUE), mean_max_humid = mean(UXAB,
            na.rm = TRUE), qlty_mean_max_humid = mean(QUXAB,
            na.rm = TRUE), mean_vap_press = mean(TSVM,
            na.rm = TRUE), qlty_mean_vap_press = mean(QTSVM,
            na.rm = TRUE), mean_ET = mean(ETP, na.rm = TRUE),
        qlty_mean_ET = mean(QETP, na.rm = TRUE), mean_radiation = mean(GLOT,
            na.rm = TRUE), qlty_mean_radiation = mean(QGLOT,
            na.rm = TRUE))
```

3.4 Importing every year population size for each city

```
dat_pop_size <- "dat_pop_size"
if (!file.exists(dat_pop_size)) download.file(
↪  "https://www.insee.fr/fr/statistiques/fichier/3698339/base-pop-historiques-1876-2022.xlsx"
↪  ,
    dat_pop_size)

pop_size <- read_excel("dat_pop_size", sheet = 1)
pop_size <- pop_size[5:34943, 1:21]
```

```
colnames(pop_size) <- pop_size[1, ]
pop_size <- pop_size[-1, ]
colnames(pop_size) <- c("insee_code", "REG", "DEP",
    "LIBGEO", "2022", "2021", "2020", "2019", "2018",
    "2017", "2016", "2015", "2014", "2013", "2012",
    "2011", "2010", "2009", "2008", "2007", "2006")
pop_size <- pop_size %>%
    select(-REG, -DEP, -LIBGEO)
head(pop_size)
```

```
# A tibble: 6 x 18
  insee_code `2022` `2021` `2020` `2019` `2018` `2017` `2016` `2015` `2014`
  <chr>      <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>
1 01001      859    832    806    779    771    776    767    767    767
2 01002      273    267    262    256    253    248    243    241    239
3 01004      15554  14854  14288  14134  14204  14035  14081  14127  14022
4 01005      1917   1897   1782   1751   1720   1689   1671   1619   1627
5 01006      114    113    113    112    112    111    110    109    109
6 01007      2828   2833   2827   2800   2763   2726   2684   2615   2570
# i 8 more variables: `2013` <chr>, `2012` <chr>, `2011` <chr>, `2010` <chr>,
#   `2009` <chr>, `2008` <chr>, `2007` <chr>, `2006` <chr>
```

```
pop_size <- gather(pop_size, "year", "population",
    -insee_code) %>%
    mutate_at(2, as.numeric)
head(pop_size)
```

```
# A tibble: 6 x 3
  insee_code  year population
  <chr>      <dbl> <chr>
1 01001       2022 859
2 01002       2022 273
3 01004       2022 15554
4 01005       2022 1917
5 01006       2022 114
6 01007       2022 2828
```

Preparing cities informations

```
colnames(cities_gps)
```

```
 [1] "insee_code"          "city_code"           "zip_code"
 [4] "label"               "latitude"            "longitude"
 [7] "department_name"     "department_number"   "region_name"
[10] "region_geojson_name"
```

```
cities_names <- cities_gps %>%
    select(insee_code, city_code, zip_code, department_name,
        department_number, region_name, latitude, longitude)
head(cities_names)
```

```
  insee_code           city_code zip_code department_name department_number
1      25620       ville du pont    25650           doubs                25
2      25624      villers grelot    25640           doubs                25
```

```
3      25615 villars les blamont    25310           doubs                25
4      25619       les villedieu    25240           doubs                25
5      25622       villers buzon    25170           doubs                25
6      25625    villers la combe    25510           doubs                25
              region_name     latitude   longitude
1 bourgogne-franche-comté 46.999873398 6.498147193
2 bourgogne-franche-comté 47.361512085 6.235167025
3 bourgogne-franche-comté 47.368383721 6.871414913
4 bourgogne-franche-comté 46.713906258  6.26583065
5 bourgogne-franche-comté 47.228558434 5.852186748
6 bourgogne-franche-comté 47.240809828 6.473842387
```

3.5 Importing "Mapping of the exposure of individual houses to clay movements" dataset

```
houses_exposure <- "houses_exposure"
if (!file.exists(houses_exposure)) download.file(
↪  "https://www.statistiques.developpement-durable.gouv.fr/media/4553/download?inline",
    houses_exposure)

houses_exposure <- read_excel("houses_exposure", sheet = 2)
head(houses_exposure)
```

```
# A tibble: 6 x 26
  insee_com nb_logement nb_logement_alea_moyen_fort nb_logement_alea_faible
  <chr>           <dbl>                       <dbl>                   <dbl>
1 01001             341                           8                     333
2 01002             169                         157                       9
3 01004            3262                        1444                    1818
4 01005             527                           0                     527
5 01006              63                           0                      63
6 01007            1085                         702                     383
# i 22 more variables: nb_logement_sans_alea <dbl>,
#   nb_logement_alea_moyen_fort_avant_1920 <dbl>,
#   part_logement_alea_moyen_fort_avant_1920 <dbl>,
#   nb_logement_alea_moyen_fort_1920_1945 <dbl>,
#   part_logement_alea_moyen_fort_1920_1945 <dbl>,
#   nb_logement_alea_moyen_fort_1945_1975 <dbl>,
#   part_logement_alea_moyen_fort_1945_1975 <dbl>, ...
```

```
colnames(houses_exposure)
```

```
 [1] "insee_com"
 [2] "nb_logement"
 [3] "nb_logement_alea_moyen_fort"
 [4] "nb_logement_alea_faible"
 [5] "nb_logement_sans_alea"
 [6] "nb_logement_alea_moyen_fort_avant_1920"
 [7] "part_logement_alea_moyen_fort_avant_1920"
 [8] "nb_logement_alea_moyen_fort_1920_1945"
 [9] "part_logement_alea_moyen_fort_1920_1945"
[10] "nb_logement_alea_moyen_fort_1945_1975"
[11] "part_logement_alea_moyen_fort_1945_1975"
[12] "nb_logement_alea_moyen_fort_apres_1975"
```

```
[13] "part_logement_alea_moyen_fort_apres_1975"
[14] "nb_logement_alea_faible_avant_1920"
[15] "part_logement_alea_faible_avant_1920"
[16] "nb_logement_alea_faible_1920_1945"
[17] "part_logement_alea_faible_1920_1945"
[18] "nb_logement_alea_faible_1945_1975"
[19] "part_logement_alea_faible_1945_1975"
[20] "nb_logement_alea_faible_apres_1975"
[21] "part_logement_alea_faible_apres_1975"
[22] "surface_commune"
[23] "surface_alea_faible_commune"
[24] "part_alea_faible_commune"
[25] "surface_alea_moyen_fort_commune"
[26] "part_alea_moyen_fort_commune"
```

We will only keep the latest data (updated in 2019) as following :

```
houses_exposure <- houses_exposure %>%
    select(insee_com, nb_logement, nb_logement_alea_faible,
        nb_logement_alea_moyen_fort, surface_commune,
        part_alea_faible_commune, part_alea_moyen_fort_commune)
```

Renaming and translating columns names

```
colnames(houses_exposure) <- c("insee_code", "n_houses",
    "n_low_risk_houses", "n_high_risk_houses", "city_area",
    "percent_low_risk_city_area", "percent_high_risk_city_area")
```

Creating new predictors

```
houses_exposure <- houses_exposure %>%
    mutate(n_houses_risk = (n_low_risk_houses + n_high_risk_houses),
        percent_low_risk_houses = (n_low_risk_houses/n_houses) *
            100, percent_high_risk_houses = (n_high_risk_houses/n_houses) *
            100, percent_houses_risk = (n_houses_risk/n_houses) *
            100)
```

Transforming values into digital data

```
houses_exposure <- houses_exposure %>%
    mutate_at(2:ncol(houses_exposure), as.numeric)
str(houses_exposure)
```

```
tibble [34,839 x 11] (S3: tbl_df/tbl/data.frame)
 $ insee_code                : chr [1:34839] "01001" "01002" "01004" "01005" ...
 $ n_houses                  : num [1:34839] 341 169 3262 527 63 ...
 $ n_low_risk_houses         : num [1:34839] 333 9 1818 527 63 ...
 $ n_high_risk_houses        : num [1:34839] 8 157 1444 0 0 ...
 $ city_area                 : num [1:34839] 15619934 9175479 24508833 16014205 6030856 ...
 $ percent_low_risk_city_area : num [1:34839] 91.36 2.79 29.49 98.93 54.08 ...
 $ percent_high_risk_city_area: num [1:34839] 8.64 85.53 70.11 0 40.06 ...
 $ n_houses_risk             : num [1:34839] 341 166 3262 527 63 ...
```

```
 $ percent_low_risk_houses    : num [1:34839] 97.65 5.33 55.73 100 100 ...
 $ percent_high_risk_houses   : num [1:34839] 2.35 92.9 44.27 0 0 ...
 $ percent_houses_risk        : num [1:34839] 100 98.2 100 100 100 ...
```

3.6 Importing "Typology of the vulnerability of municipalities to climate risks" dataset

```
risks_exposure <- "risks_exposure"
if (!file.exists(risks_exposure)) download.file(
↪  "https://www.statistiques.developpement-durable.gouv.fr/media/3514/download?inline",
    risks_exposure)

risks_exposure <- read_excel("risks_exposure", sheet = 2)
head(risks_exposure)
```

```
# A tibble: 6 x 7
  CLASSE codgeo exp_aval exp_atm exp_ino exp_mvt exp_feu
  <chr>  <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 0      01005         0       0       0       0       0
2 0      01006         0       0       0       0       0
3 0      01008         0       0       0       0       0
4 0      01011         0       0       0       0       0
5 0      01012         0       0       0       0       0
6 0      01013         0       0       0       0       0
```

```
colnames(risks_exposure)
```

```
[1] "CLASSE"   "codgeo"   "exp_aval" "exp_atm"  "exp_ino"  "exp_mvt"  "exp_feu"
```

Filtering on the data wee need (updated in 2016) as following :

```
risks_exposure <- risks_exposure %>%
    select(codgeo, exp_atm, exp_ino, exp_mvt, exp_feu)
```

Renaming and translating columns names

```
colnames(risks_exposure) <- c("insee_code", "exp_atmospheric",
    "exp_floods", "exp_land_mvt", "exp_fire")
head(risks_exposure)
```

```
# A tibble: 6 x 5
  insee_code exp_atmospheric exp_floods exp_land_mvt exp_fire
  <chr>                <dbl>      <dbl>        <dbl>    <dbl>
1 01005                    0          0            0        0
2 01006                    0          0            0        0
3 01008                    0          0            0        0
4 01011                    0          0            0        0
5 01012                    0          0            0        0
6 01013                    0          0            0        0
```

Transforming values into digital data

```
risks_exposure <- risks_exposure %>%
    mutate_at(2:ncol(risks_exposure), as.numeric)
str(risks_exposure)
```

```
tibble [35,889 x 5] (S3: tbl_df/tbl/data.frame)
 $ insee_code     : chr [1:35889] "01005" "01006" "01008" "01011" ...
 $ exp_atmospheric: num [1:35889] 0 0 0 0 0 0 0 0 0 0 ...
 $ exp_floods     : num [1:35889] 0 0 0 0 0 0 0 0 0 0 ...
 $ exp_land_mvt   : num [1:35889] 0 0 0 0 0 0 0 0 0 0 ...
 $ exp_fire       : num [1:35889] 0 0 0 0 0 0 0 0 0 0 ...
```

3.7 Importing "History of drought states" dataset

Since the public dataset was an 8.8GB JSON file :https://www.data.gouv.fr/fr/datasets/donnee-secheresse-vigieau/#/resources,
I processed this document in parallel to make it lighter.
In this study, we will therefore download the processed version of the dataset,
which is publicly accessible on my GitHub.

```
droughts <- "droughts.csv"
if (!file.exists(droughts)) download.file(
↪  "https://raw.githubusercontent.com/meryllm/natural_disasters_clay_movement/refs/heads/main/drought_
↪  ,
    droughts)
droughts <- read.csv(droughts)
head(droughts)
```

	insee_code	year	n_AEP	n_SOU	n_SUP	n_tot
1	01001	2015	0	43	0	43
2	01001	2017	0	115	27	142
3	01001	2018	0	117	14	131
4	01001	2019	0	41	27	68
5	01001	2020	0	0	78	78
6	01002	2015	0	0	43	43

```
str(droughts)
```

```
'data.frame':   79477 obs. of  6 variables:
 $ insee_code: chr  "01001" "01001" "01001" "01001" ...
 $ year      : int  2015 2017 2018 2019 2020 2015 2017 2018 2019 2020 ...
 $ n_AEP     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ n_SOU     : int  43 115 117 41 0 0 53 117 231 104 ...
 $ n_SUP     : int  0 27 14 27 78 43 62 59 115 0 ...
 $ n_tot     : int  43 142 131 68 78 43 115 176 346 104 ...
```

Transforming values into digital data

```
droughts <- droughts %>%
    mutate_at(2:ncol(droughts), as.numeric)
str(droughts)
```

```
'data.frame':   79477 obs. of  6 variables:
```

```
 $ insee_code: chr  "01001" "01001" "01001" "01001" ...
 $ year      : num  2015 2017 2018 2019 2020 ...
 $ n_AEP     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ n_SOU     : num  43 115 117 41 0 0 53 117 231 104 ...
 $ n_SUP     : num  0 27 14 27 78 43 62 59 115 0 ...
 $ n_tot     : num  43 142 131 68 78 43 115 176 346 104 ...
```

Adding all datasets to "df1"

```
df1 <- df1 %>%
    inner_join(weather_year, by = c("NUM_POSTE", "year"),
        relationship = "many-to-many") %>%
    left_join(cities_names, by = "insee_code", relationship = "many-to-many") %>%
    left_join(pop_size, by = c("insee_code", "year"),
        relationship = "many-to-many") %>%
    left_join(houses_exposure, by = "insee_code", relationship = "many-to-many") %>%
    left_join(risks_exposure, by = "insee_code", relationship = "many-to-many") %>%
    left_join(droughts, by = c("insee_code", "year"),
        relationship = "many-to-many")
```

Reorganizing dataframe columns

```
df1 <- df1 %>%
    select(insee_code, year, population, city_code,
        zip_code, department_name, department_number,
        region_name, latitude, longitude, everything(),
        -demand, -nat_dis, demand, nat_dis)
```

Creation of our column containing the 3 states of natural disaster to predict

```
df1 <- df1 %>%
    mutate(natural_disaster = ifelse(nat_dis == 1,
        2, ifelse(demand == 1, 1, 0)))
df1 <- df1 %>%
    mutate(natural_disaster = factor(natural_disaster,
        levels = c("0", "1", "2"), labels = c("No Natural Disaster",
            "Requested", "Acknowledged")))
```

4 Data Exploration & Analysis

```
str(df1)
```

```
tibble [553,848 x 68] (S3: tbl_df/tbl/data.frame)
 $ insee_code                : chr [1:553848] "10010" "10010" "10018" "10018" ...
 $ year                      : num [1:553848] 2018 2019 2018 2019 2020 ...
 $ population                : chr [1:553848] "48" "47" "1012" "1023" ...
 $ city_code                 : chr [1:553848] "arrembecourt" "arrembecourt" "auxon" "auxon" ...
 $ zip_code                  : chr [1:553848] "10330" "10330" "10130" "10130" ...
 $ department_name           : chr [1:553848] "aube" "aube" "aube" "aube" ...
 $ department_number         : chr [1:553848] "10" "10" "10" "10" ...
```

```
$ region_name                : chr [1:553848] "grand est" "grand est" "grand est" "grand est" ...
$ latitude                   : chr [1:553848] "48.543829581" "48.543829581" "48.092371915" "48.0923719
$ longitude                  : chr [1:553848] "4.6059035690000005" "4.6059035690000005" "3.92343338" "
$ NUM_POSTE                  : chr [1:553848] "51135001" "51135001" "10099002" "10099002" ...
$ distance_km                : num [1:553848] 7.46 7.46 7.47 7.47 7.47 ...
$ cumul_precipit             : num [1:553848] 804 750 752 712 713 ...
$ qlty_precipit              : num [1:553848] 1 1 1 1 1 ...
$ n_day_1mm_precipit         : num [1:553848] 127 136 121 130 115 115 127 102 111 115 ...
$ n_day_5mm_precipit         : num [1:553848] 55 56 54 47 44 44 55 45 49 58 ...
$ n_day_10mm_precipit        : num [1:553848] 22 17 24 16 18 18 22 18 23 34 ...
$ n_day_30mm_precipit        : num [1:553848] 2 0 0 1 1 1 2 1 1 3 ...
$ n_day_50mm_precipit        : num [1:553848] 0 0 0 0 0 0 0 0 0 1 ...
$ n_day_100mm_precipit       : num [1:553848] 0 0 0 0 0 0 0 0 0 0 ...
$ mean_temp                  : num [1:553848] NaN NaN 12.1 11.7 12.4 ...
$ qlty_mean_temp             : num [1:553848] 3 3 1 1 1 1 3 3 1 1 ...
$ n_day_below_0              : num [1:553848] 0 0 4 1 0 0 0 0 4 13 ...
$ n_day_above_25             : num [1:553848] 0 0 107 76 82 82 0 0 102 84 ...
$ n_day_above_30             : num [1:553848] 0 0 34 28 26 26 0 0 32 30 ...
$ n_day_above_35             : num [1:553848] 0 0 5 7 9 9 0 0 5 4 ...
$ n_day_below_5              : num [1:553848] 0 0 7 5 1 1 0 0 7 20 ...
$ n_day_below_10             : num [1:553848] 0 0 1 0 0 0 0 0 1 2 ...
$ n_day_below_20             : num [1:553848] 0 0 199 224 213 213 0 0 202 204 ...
$ n_day_below_27             : num [1:553848] 0 0 292 308 310 310 0 0 301 304 ...
$ n_day_above_32             : num [1:553848] 0 0 14 22 19 19 0 0 12 14 ...
$ mean_temp_spring           : num [1:553848] NaN NaN 11.47 9.83 11.67 ...
$ mean_temp_summer           : num [1:553848] NaN NaN 20.4 20.1 19.5 ...
$ mean_temp_autumn           : num [1:553848] NaN NaN 11.8 12 12.1 ...
$ mean_temp_winter           : num [1:553848] NaN NaN 4.87 4.73 6.43 ...
$ mean_humid                 : num [1:553848] NaN NaN NaN NaN NaN ...
$ qlty_mean_humid            : num [1:553848] 3 3 3 3 3 3 3 3 1.75 2 ...
$ mean_min_humid             : num [1:553848] NaN NaN NaN NaN NaN ...
$ qlty_mean_min_humid        : num [1:553848] 3 3 3 3 3 3 3 3 1 1 ...
$ mean_max_humid             : num [1:553848] NaN NaN NaN NaN NaN ...
$ qlty_mean_max_humid        : num [1:553848] 3 3 3 3 3 3 3 3 1 1 ...
$ mean_vap_press             : num [1:553848] NaN NaN NaN NaN NaN ...
$ qlty_mean_vap_press        : num [1:553848] 3 3 3 3 3 3 3 3 2 2 ...
$ mean_ET                    : num [1:553848] NaN NaN NaN NaN NaN ...
$ qlty_mean_ET               : num [1:553848] 3 3 3 3 3 3 3 3 1 1 ...
$ mean_radiation             : num [1:553848] NaN NaN NaN NaN NaN ...
$ qlty_mean_radiation        : num [1:553848] 3 3 3 3 3 ...
$ n_houses                   : num [1:553848] 34 34 478 478 478 ...
$ n_low_risk_houses          : num [1:553848] 0 0 118 118 118 ...
$ n_high_risk_houses         : num [1:553848] 34 34 360 360 360 ...
$ city_area                  : num [1:553848] 7195055 7195055 25586603 25586603 25586603 ...
$ percent_low_risk_city_area : num [1:553848] 0 0 18.5 18.5 18.5 ...
$ percent_high_risk_city_area: num [1:553848] 99.1 99.1 80.6 80.6 80.6 ...
$ n_houses_risk              : num [1:553848] 34 34 478 478 478 ...
$ percent_low_risk_houses    : num [1:553848] 0 0 24.7 24.7 24.7 ...
$ percent_high_risk_houses   : num [1:553848] 100 100 75.3 75.3 75.3 ...
$ percent_houses_risk        : num [1:553848] 100 100 100 100 100 100 100 100 100 100 ...
$ exp_atmospheric            : num [1:553848] 0 0 0 0 0 0 0 0 0 0 ...
$ exp_floods                 : num [1:553848] 0 0 0 0 0 0 0 0 1 1 ...
$ exp_land_mvt               : num [1:553848] 1 1 1 1 1 1 1 1 1 1 ...
$ exp_fire                   : num [1:553848] 0 0 0 0 0 0 0 0 0 0 ...
```

```
 $ n_AEP                      : num [1:553848] NA NA NA NA NA NA NA NA NA NA ...
 $ n_SOU                      : num [1:553848] NA NA NA NA NA NA NA NA NA NA ...
 $ n_SUP                      : num [1:553848] NA NA NA NA NA NA NA NA NA NA ...
 $ n_tot                      : num [1:553848] NA NA NA NA NA NA NA NA NA NA ...
 $ demand                     : num [1:553848] 1 1 1 1 1 1 1 1 1 1 ...
 $ nat_dis                    : num [1:553848] 1 1 1 1 1 1 1 1 1 0 ...
 $ natural_disaster           : Factor w/ 3 levels "No Natural Disaster",..: 3 3 3 3 3 3 3 3 3 2 ...
```

Creating a personalized theme we will apply on all our visualizations, to avoid repetitions :

```
standard_theme <- theme_bw() + theme(plot.title = element_text(size = 14,
    face = "bold"), legend.position = "bottom", axis.text = element_text(size = 10),
    axis.title = element_text(size = 12, face = "bold"),
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 10), strip.text = element_text(size = 10,
        face = "bold"))
```

As we see there where only few acknowledged natural disaster from 2004 to 2008, so we can filter out this period

```
df1 %>%
    filter(year < 2009 & nat_dis == 1) %>%
    nrow()
```

```
[1] 10
```

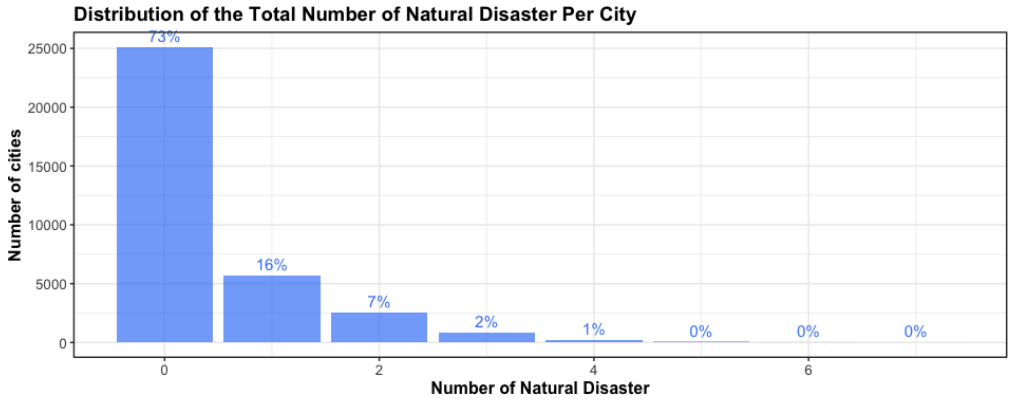
```
df1 %>%
    filter(year < 2010 & nat_dis == 1) %>%
    nrow()
```

```
[1] 595
```

```
dataframe_trend <- df1 %>%
    filter(year > 2007)  # We keep informations about year 2008 in order to calculate
    ↪  trend on from 2008 to 2009 later
df1 <- df1 %>%
    filter(year > 2008)
```

By quickly analyzing the distribution of the number of natural disaster per city, we see that a lot of cities never had natural disaster :

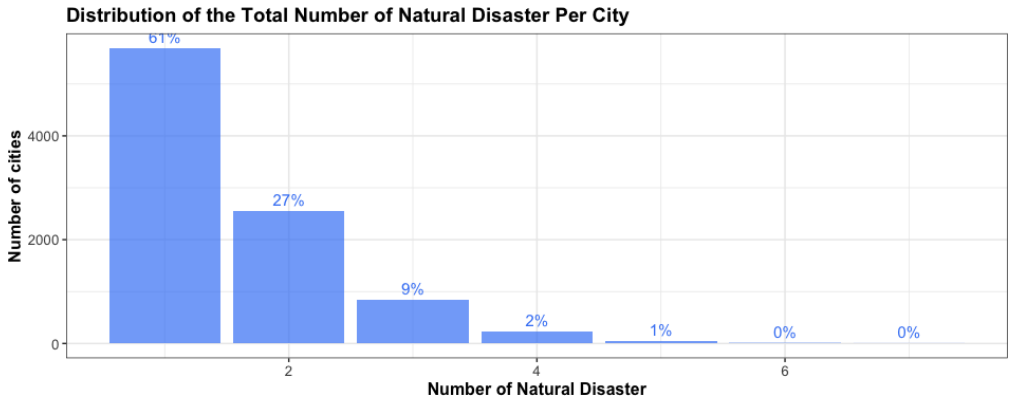
```
df1 %>%
    group_by(insee_code) %>%
    summarise(n_nat_dis = sum(nat_dis)) %>%
    ggplot(aes(n_nat_dis)) + geom_bar(alpha = 0.7,
    fill = "#4285f6") + geom_text(stat = "count", aes(label =
    ↪  scales::percent(..count../sum(..count..),
    accuracy = 1)), vjust = -0.5, size = 4, color = "#4285f6") +
    labs(title = "Distribution of the Total Number of Natural Disaster Per City",
        x = "Number of Natural Disaster", y = "Number of cities") +
    standard_theme + coord_cartesian(clip = "off")
```



Since, cities that have never had natural disasters are significantly not on clay soils, or not impacted yet about the environement.
We can remove them from our study because they risk biasing the results.

Let's explore the distribution of the number of natural disaster per city, by filtering out the cities that nerver had natural disasters.

```
df1 %>%
    group_by(insee_code) %>%
    summarise(n_nat_dis = sum(nat_dis)) %>%
    filter(n_nat_dis > 0) %>%
    ggplot(aes(n_nat_dis)) + geom_bar(alpha = 0.7,
    fill = "#4285f6") + geom_text(stat = "count", aes(label =
    ↪  scales::percent(..count../sum(..count..),
    accuracy = 1)), vjust = -0.5, size = 4, color = "#4285f6") +
    labs(title = "Distribution of the Total Number of Natural Disaster Per City",
        x = "Number of Natural Disaster", y = "Number of cities") +
    standard_theme
```



We see that, from 2009 to 2020 (when city with no natural disaster are filtered out) :
- 61% of the cities had only 1 Natural disaster,
- 88% of the cities had 2 natural disaster or less,
- 97% of the cities had 3 natural disaster or less,
- 99% of the cities had 4 natural disaster or less.

Regarding this we can create different natural disaster group exposure, as following :

```r
nat_dis_group <- df %>%
    group_by(insee_code) %>%
    summarise(nat_dis_group = sum(nat_dis)) %>%
    select(insee_code, nat_dis_group)
```

In this study, we will assume that these cities in group "nat_dis_group" = 0, are not affected by clay soils. Consequently, we will arbitrarily consider that they will never be impacted by natural disasters in our predictions.

Of course, we adopt this assumption to limit the study's scope and duration.
It is possible that some of these cities do sit on critical clay soils,
but that the specific meteorological conditions required to trigger a natural disaster have simply not occurred yet,
meaning the city has not been impacted or has never filed for official disaster recognition.

To better anticipate risks in such cities, more detailed information on clay soil distribution would be required.

However, in our study, we will go as far as completely excluding these cities from the modeling process, in order to obtain a more meaningful and reliable "global accuracy" indicator.

By filtering cities with no natural disaster, we are considerably reducing the number of rows

```r
nrow(df1)
```

```
[1] 415807
```

```r
df1 %>%
    left_join(nat_dis_group, by = "insee_code") %>%
    filter(!nat_dis_group == 0) %>%
    nrow()
```

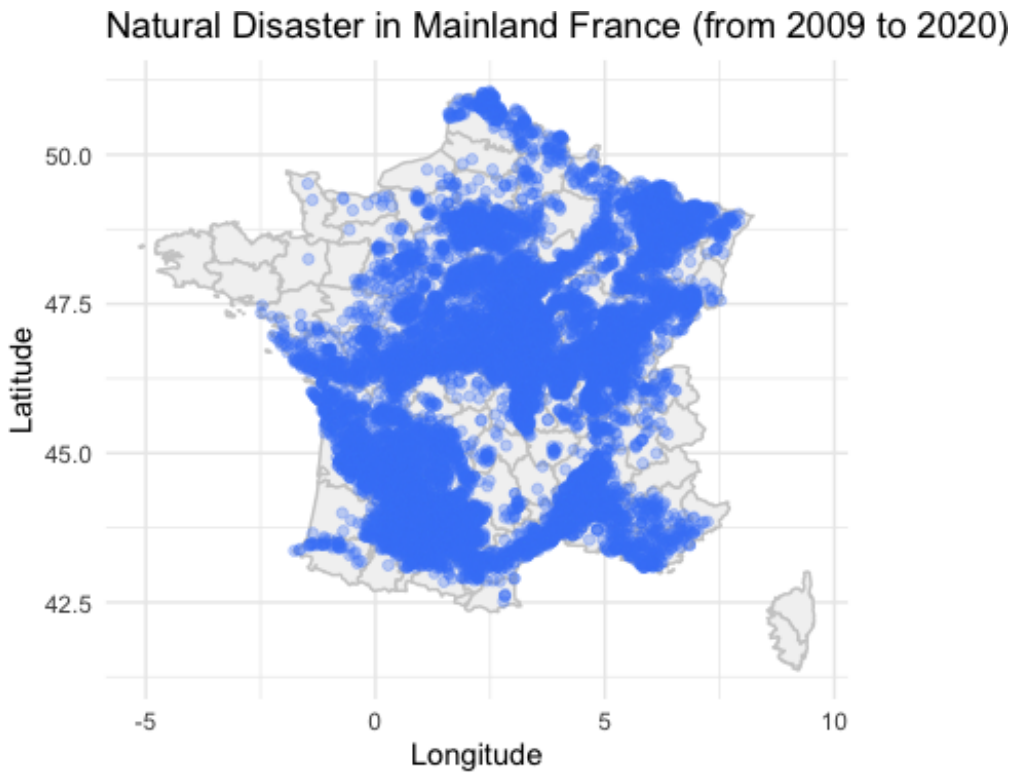
```
[1] 114378
```

```r
df2 <- df1 %>%
    left_join(nat_dis_group, by = "insee_code") %>%
    filter(nat_dis_group != 0)
```

Let's check on a 2D map of France, if whether the mapping of natural disaster cases corresponds to the distribution of clay soil areas

```r
df2_filtred_nat_dis<-df2%>%
          filter(nat_dis==1)
# 2D map creation
ggplot(df2_filtred_nat_dis, aes(x = as.numeric(longitude), y = as.numeric(latitude))) +
  borders("france", colour = "gray80", fill = "gray95") +  # Map
  geom_point(color = "#4285f6", alpha = 0.3, size = 1.5) +    # Circle markers
  coord_fixed(1.3) +  # latitude/longitude proportion
  theme_minimal() +
  labs(
    title = "Natural Disaster in Mainland France (from 2009 to 2020)",
    x = "Longitude",
```

```
    y = "Latitude"
  )
```

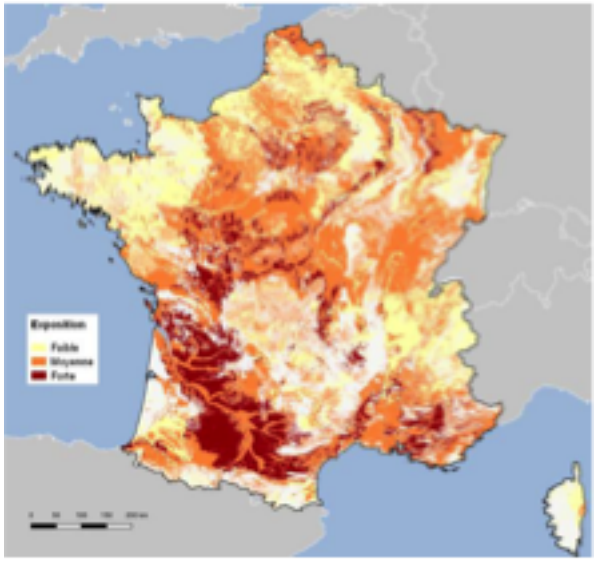


Let's compare it with a French map of clay soil distribution

```
url <-
↪  "https://www.ain.gouv.fr/var/ide_site/storage/images/8/7/6/1/131678-1-fre-FR/2020cartefranceargiles
destfile <- "france_clay_soil_map.jpg"
GET(url, write_disk(destfile, overwrite = TRUE))
```

```
Response [https://www.ain.gouv.fr/var/ide_site/storage/images/8/7/6/1/131678-1-fre-FR/2020cartefrancear
  Date: 2025-05-07 10:37
  Status: 200
  Content-Type: image/jpeg
  Size: 59.8 kB
<ON DISK>  /Users/meryllmercadier/Rstudio/Harvard courses/9. Capstone/Final_project/france_clay_soil_ma
```

```
image <- image_read("france_clay_soil_map.jpg")
plot(image)
```



We observe that the distribution of our points with natural disasters is consistent, as it closely aligns with the clay soil map.

4.1 Analyzing NAs

```
df2_nas <- round(colSums(is.na(df2))/nrow(df2) * 100)
df2_nas <- as.data.frame(df2_nas)
colnames(df2_nas) <- c("na_percent")
df2_nas %>%
    mutate(remaining_lines = round((nrow(df2) * (1 -
        (na_percent/100))))) %>%
    arrange(desc(na_percent)) %>%
    filter(na_percent > 0)
```

	na_percent	remaining_lines
mean_radiation	90	11438
mean_ET	89	12582
n_AEP	82	20588
n_SOU	82	20588
n_SUP	82	20588
n_tot	82	20588
mean_humid	73	30882
mean_min_humid	73	30882
mean_max_humid	73	30882
mean_vap_press	73	30882
mean_temp_spring	62	43464
mean_temp_summer	61	44607
mean_temp_autumn	61	44607
mean_temp	60	45751
mean_temp_winter	60	45751
cumul_precipit	3	110947

By analyzing the NAs in our dataframe,
We see that 9 predictors have around 60%-73% NAS

We see that 6 predictors have around 82%-90% NAS
By filtering out all NAs in our dataframe we will miss a large amount of data,
We will see later if we obtain more information by keeping the predictors with a large amount of NAs

As we have a lot of predictors we will have to filter them, by methodically select those which are most likely to help us with our prediction algorithm.

As a reminder, correlation with between predictors do not justify causality.
We just use this method to make a first filter in our predictors before to studying them in detail.

To do so, we will first focus on the correlation between "natural_disaster" and other predictors, using 3 methods :
- Pearson : To measure the linear correlation between predictors (does not seem to be the most suitable method in our case, but could still reveal information)
- Spearman : To measure a nonlinear monotonic correlation between the predictors (seems more appropriate in our study, to hightlight outliers)

Adapting our mane dataframe to apply correlation calculations

```
correlation_df <- df2 %>%
    select(-city_code, -zip_code, -department_name,
        -region_name, -exp_atmospheric, -n_AEP)
correlation_df <- correlation_df %>%
    mutate_at(1:ncol(correlation_df), as.numeric)
```

Calculating correlation with different methods

```
cor_pearson <- cor(correlation_df, method = "pearson",
    use = "pairwise.complete.obs")
cor_spearman <- cor(correlation_df, method = "spearman",
    use = "pairwise.complete.obs")
```

Focusing on "Natural_disaster" correlation

```
natdis_cor_pearson <- cor_pearson["natural_disaster",
    ]
natdis_cor_spearman <- cor_spearman["natural_disaster",
    ]
```

Filtering our relevant predictors for an absolute correlation greater than or equal to 0.17.

```
selected_pearson <- names(natdis_cor_pearson[abs(natdis_cor_pearson) >=
    0.16])
selected_spearman <- names(natdis_cor_spearman[abs(natdis_cor_spearman) >=
    0.16])
```

Keeping each predictors present in the filtred predictors

```
selected_predictors <- unique(c(selected_pearson, selected_spearman))
```

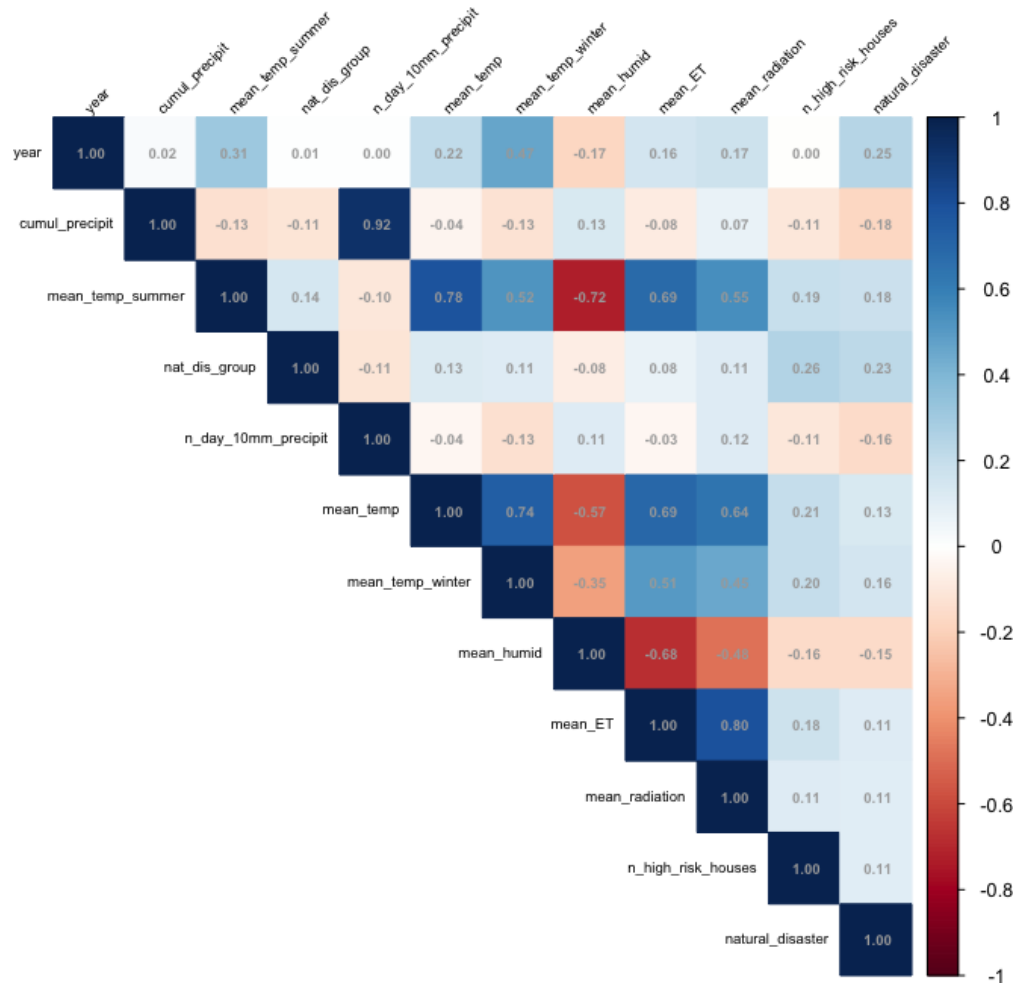
Placing "Natural_disaster" into last position and filtering out demand and natural_disaster

```
selected_predictors <- c(selected_predictors[-c(which(selected_predictors ==
    "natural_disaster"), which(selected_predictors ==
    "demand"), which(selected_predictors == "nat_dis"))],
    "natural_disaster")
length(selected_predictors)  # We still have 12 predictors to work on 'Natural_disaster'
↪  prediction
```

```
[1] 12
```

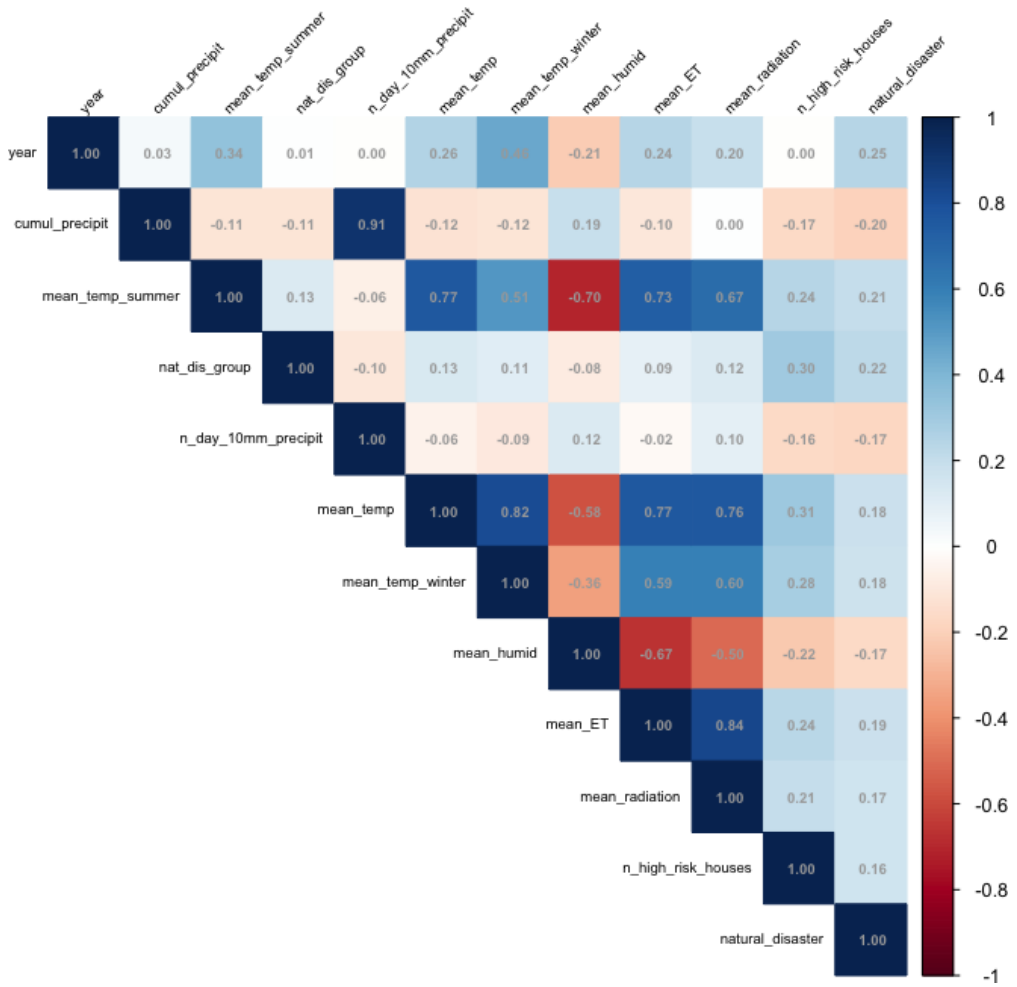
Let's see if we can filter our predictors even better by removing predictors that report the same information. Creating Pearson correlation matrix on selected predictors

```
cor_pearson[, selected_predictors][selected_predictors,
    ] %>%
    corrplot(method = "color", type = "upper", order = "original",
        tl.col = "black", tl.srt = 45, tl.cex = 0.6,
        addCoef.col = "darkgrey", number.cex = 0.6,
        main = "Correlation Matrix - Pearson Method",
        mar = c(0, 0, 6, 0))
```



Creating Spearman correlation matrix on selected predictors

```
cor_spearman[, selected_predictors][selected_predictors,
    ] %>%
    corrplot(method = "color", type = "upper", order = "original",
        tl.col = "black", tl.srt = 45, tl.cex = 0.6,
        addCoef.col = "darkgrey", number.cex = 0.6,
        main = "Correlation Matrix - Spearman Method",
        mar = c(0, 0, 6, 0))
```



These correlation matrices allow us to see that there are certain correlations between the predictors
We will use those matrix to study each predictors individually later.

We see a strong correlation between “cumul_precipit” and “n_day_10mm_precipit”.
This makes sense since “cumul_precipit” takes into account part of “n_day_10mm_precipit.”
But if these two predictors are too similar, it could impact the performance of our future models.
It’s better to keep only one of the two, in our case “cumul_precipit,” which contains more information.

```
selected_predictors <- selected_predictors[-which(selected_predictors ==
    "n_day_10mm_precipit")]
```

Let’s continue with a quick overview of all the selected predictors

```
overview_predictors <- df2[, selected_predictors]
colnames(overview_predictors)
```

```
 [1] "year"              "cumul_precipit"    "mean_temp_summer"
 [4] "nat_dis_group"     "mean_temp"         "mean_temp_winter"
```

```
 [7] "mean_humid"         "mean_ET"            "mean_radiation"
[10] "n_high_risk_houses" "natural_disaster"
```

```
overview_predictors <- overview_predictors %>%
    mutate_all(as.numeric) %>%
    pivot_longer(cols = mean_temp_summer:natural_disaster,
        names_to = "predictors", values_to = "values")

overview_predictors <- df2[, selected_predictors]
selected_predictors
```

```
 [1] "year"               "cumul_precipit"     "mean_temp_summer"
 [4] "nat_dis_group"      "mean_temp"          "mean_temp_winter"
 [7] "mean_humid"         "mean_ET"            "mean_radiation"
[10] "n_high_risk_houses" "natural_disaster"
```

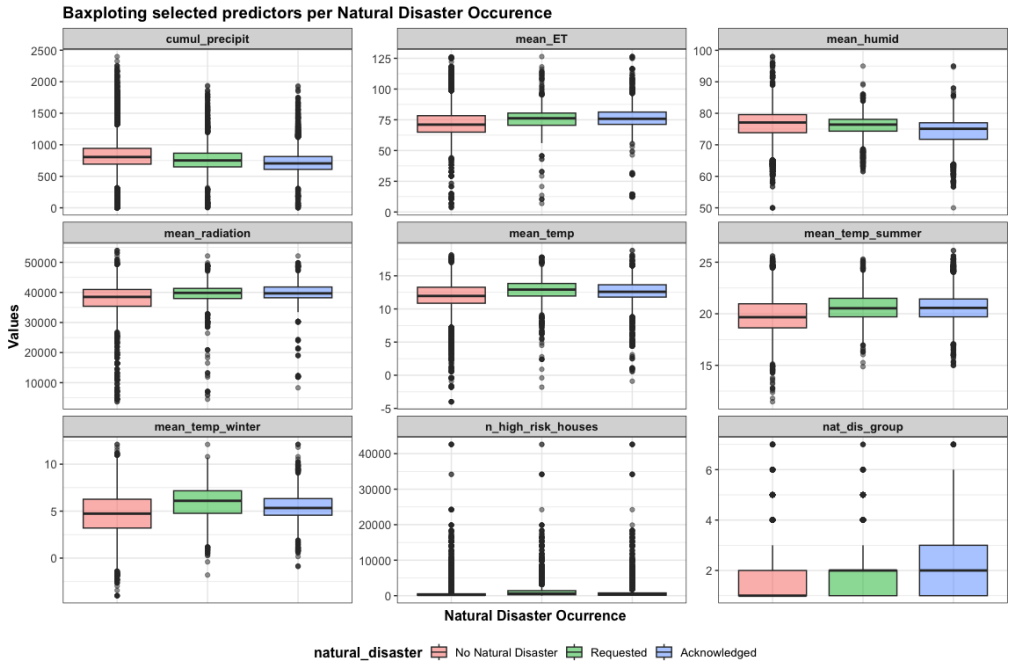
```
colnames(overview_predictors)
```

```
 [1] "year"               "cumul_precipit"     "mean_temp_summer"
 [4] "nat_dis_group"      "mean_temp"          "mean_temp_winter"
 [7] "mean_humid"         "mean_ET"            "mean_radiation"
[10] "n_high_risk_houses" "natural_disaster"
```

```
overview_predictors <- overview_predictors %>%
    pivot_longer(cols = 2:(ncol(overview_predictors) -
        1), names_to = "predictors", values_to = "values")
```

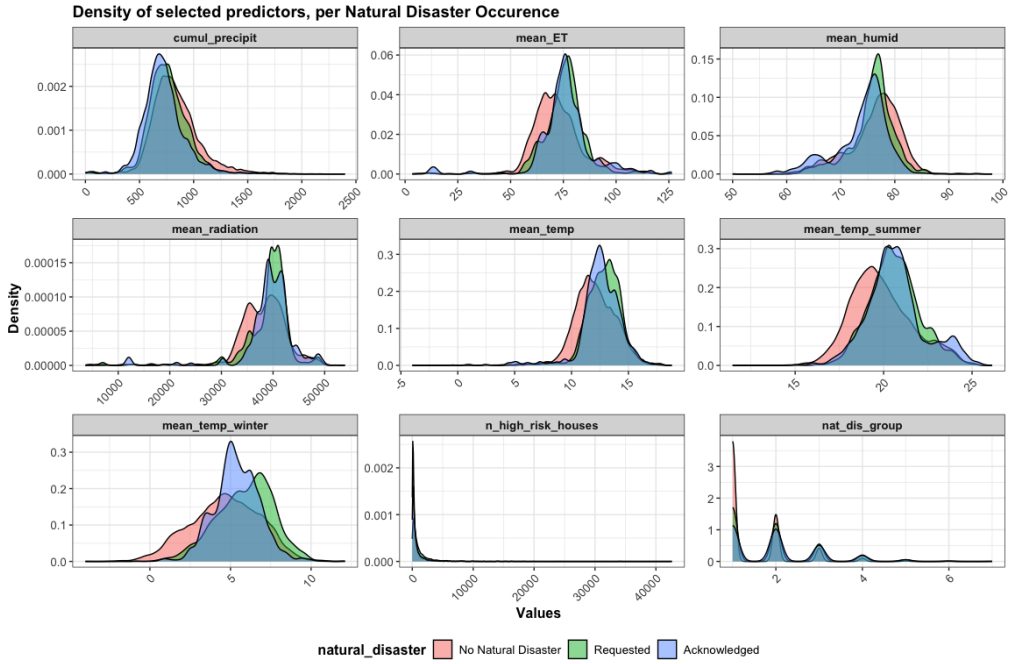
What is the boxplot chart of selected predictors grouped by natural disaster occurrence ?

```
overview_predictors %>%
    ggplot(aes(x = natural_disaster, y = values, fill = natural_disaster)) +
    geom_boxplot(alpha = 0.5) + facet_wrap(~predictors,
    scales = "free_y") + labs(title = "Baxploting selected predictors per Natural
    ↪  Disaster Occurence",
    x = "Natural Disaster Ocurrence", y = "Values") +
    standard_theme + theme(axis.text.x = element_blank(),
    axis.ticks.x = element_blank())
```



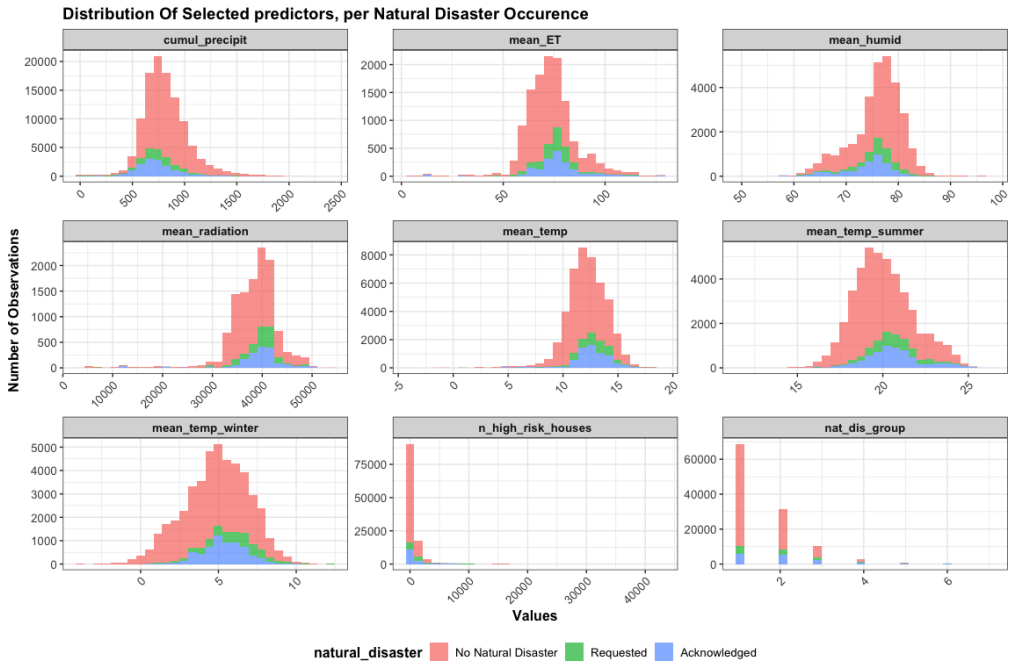
What is the density chart of selected predictors grouped by natural disaster occurrence ?

```
overview_predictors %>%
    ggplot(aes(x = values, fill = natural_disaster)) +
    geom_density(alpha = 0.5) + facet_wrap(~predictors,
    scales = "free") + labs(title = "Density of selected predictors, per Natural Disaster
    ↪  Occurence",
    x = "Values", y = "Density") + standard_theme +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



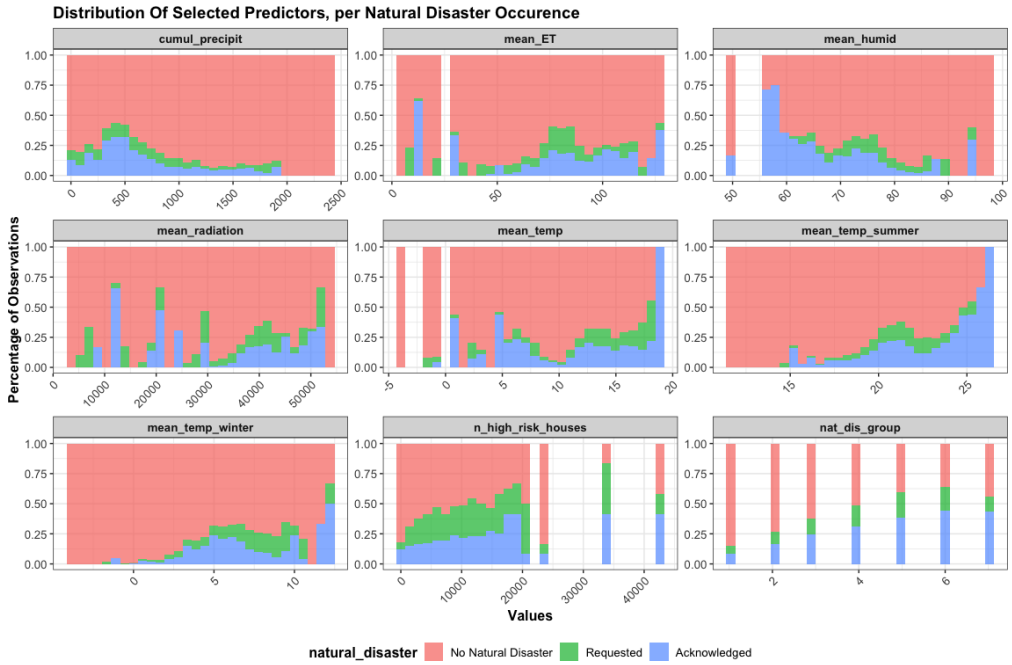
What is the histogram chart of selected predictors grouped by natural disaster occurrence ?

```
overview_predictors %>%
    ggplot(aes(x = values, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + facet_wrap(~predictors,
    scales = "free") + labs(title = "Distribution Of Selected predictors, per Natural
    ↪  Disaster Occurence",
    x = "Values", y = "Number of Observations") + standard_theme +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



What is the proportion chart of selected predictors grouped by natural disaster occurrence ?

```
overview_predictors %>%
    ggplot(aes(x = values, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "fill") +
    facet_wrap(~predictors, scales = "free") + labs(title = "Distribution Of Selected
    ↪  Predictors, per Natural Disaster Occurence",
    x = "Values", y = "Percentage of Observations") +
    standard_theme + theme(axis.text.x = element_text(angle = 45,
    hjust = 1))
```

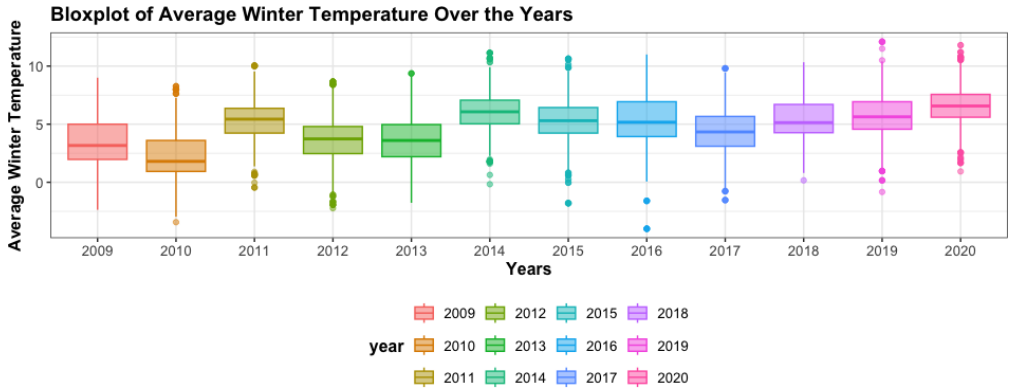


Now let's analyse each predictor individually following "selected_predictors" order :
selected_predictors
- "year" "cumul_precipit" "mean_temp_summer"
- "nat_dis_group" "mean_temp" "mean_temp_winter"
- "mean_humid" "mean_ET" "mean_radiation"
- "n_high_risk_houses" "natural_disaster"

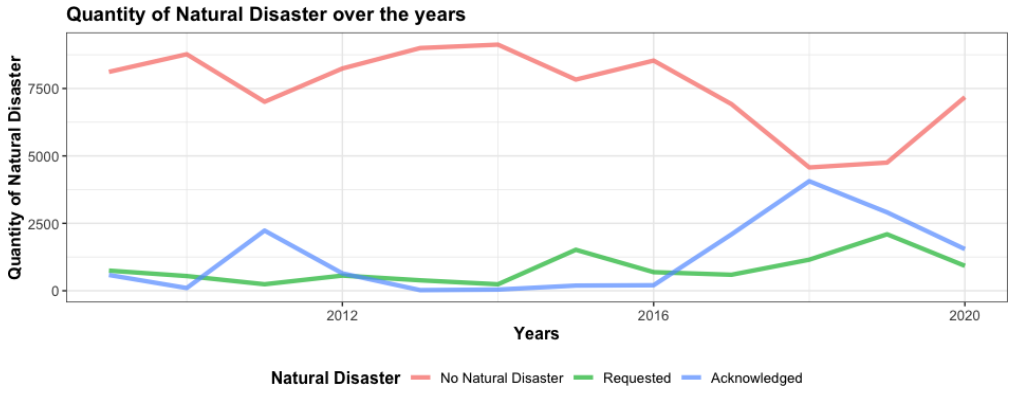
4.2 Analyzing "Year"

We saw earlier a positive correlation between year and average seasonal temperatures
For example, the further we go in the years, the higher the average temperature in winter:

```
df2 %>%
    mutate(year = as.factor(year)) %>%
    ggplot(aes(year, mean_temp_winter, group = year,
        color = year, fill = year)) + geom_boxplot(alpha = 0.5) +
    labs(title = "Bloxplot of Average Winter Temperature Over the Years",
        x = "Years", y = "Average Winter Temperature") +
    standard_theme
```




```
df2 %>%
    group_by(year, natural_disaster) %>%
    summarise(n_natural_disaster = n()) %>%
    ggplot(aes(year, n_natural_disaster, group = natural_disaster,
        color = natural_disaster)) + geom_line(alpha = 0.7,
    size = 1.5) + labs(title = "Quantity of Natural Disaster over the years",
    x = "Years", y = "Quantity of Natural Disaster",
    color = "Natural Disaster") + standard_theme
```



It seems consistent that the "year" predictor is correlated with the occurrence of natural disasters. However, since the goal of this study is to predict the occurrence of natural disasters for future years, We will not use this predictor when training the models.

```
selected_predictors <- selected_predictors[-which(selected_predictors ==
    "year")]
selected_predictors
```

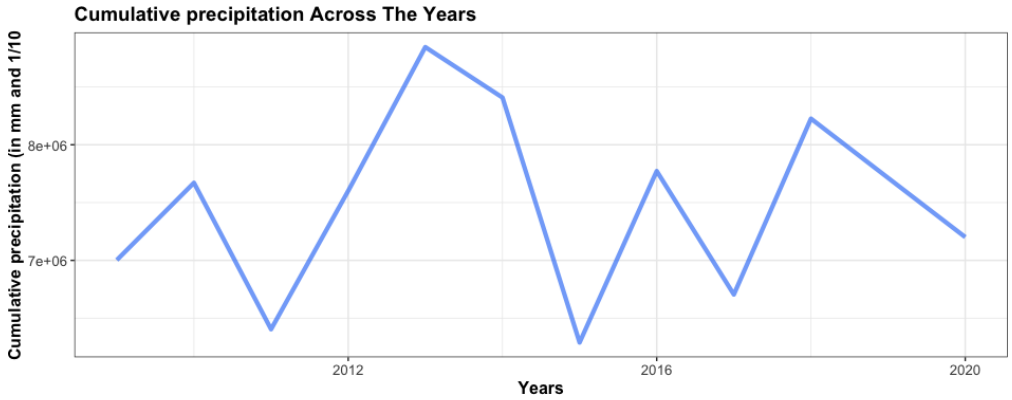
```
 [1] "cumul_precipit"     "mean_temp_summer"   "nat_dis_group"
 [4] "mean_temp"          "mean_temp_winter"   "mean_humid"
 [7] "mean_ET"            "mean_radiation"     "n_high_risk_houses"
[10] "natural_disaster"
```

```
str(selected_predictors)
```

```
 chr [1:10] "cumul_precipit" "mean_temp_summer" "nat_dis_group" "mean_temp" ...
```

4.3 Analyzing "cumul_precipit"

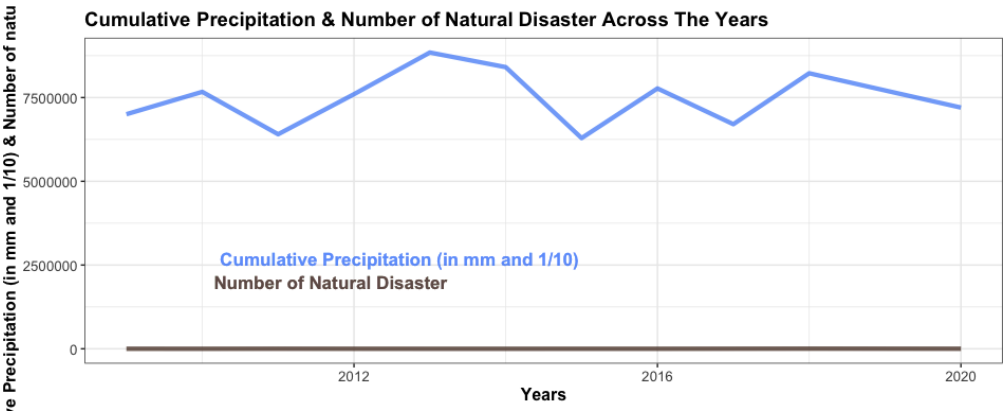
```
df2 %>%
    group_by(year) %>%
    summarise(sum_cumul_precipit = sum(cumul_precipit,
        na.rm = TRUE)) %>%
    ggplot(aes(year, sum_cumul_precipit)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Cumulative precipitation Across The Years",
    x = "Years", y = "Cumulative precipitation (in mm and 1/10") +
    standard_theme
```



We see that the cumulative precipitation is globaly increasing over the years

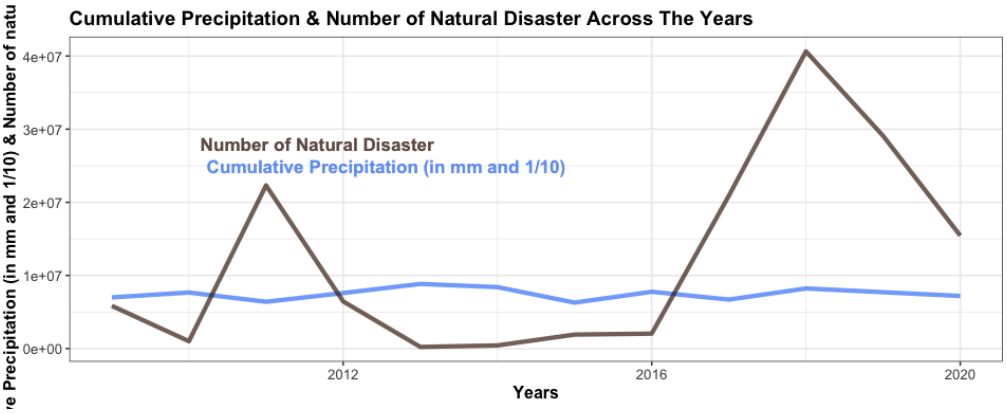
Let's compare average cumulative precipitation with the number of natural disaster over the years

```
df2 %>%
    group_by(year) %>%
    summarise(sum_cumul_precipit = sum(cumul_precipit,
        na.rm = TRUE), n_nat_dis = sum(nat_dis)) %>%
    ggplot(aes(x = year)) + geom_line(aes(y = sum_cumul_precipit),
    color = "#4285f6", alpha = 0.7, size = 1.5) + geom_line(aes(y = n_nat_dis),
    color = "#421", alpha = 0.7, size = 1.5) + geom_text(aes(x = 2010,
    y = 2700000, label = "Cumulative Precipitation (in mm and 1/10)"),
    color = "#4285f6", alpha = 0.12, hjust = -0.05,
    size = 4.5, fontface = "bold") + geom_text(aes(x = 2010,
    y = 2e+06, label = "Number of Natural Disaster"),
    color = "#421", alpha = 0.12, hjust = -0.05, size = 4.5,
    fontface = "bold") + labs(title = "Cumulative Precipitation & Number of Natural
    ↪  Disaster Across The Years",
    x = "Years", y = "Cumulative Precipitation (in mm and 1/10) & Number of natural
    ↪  Disaster") +
    standard_theme
```



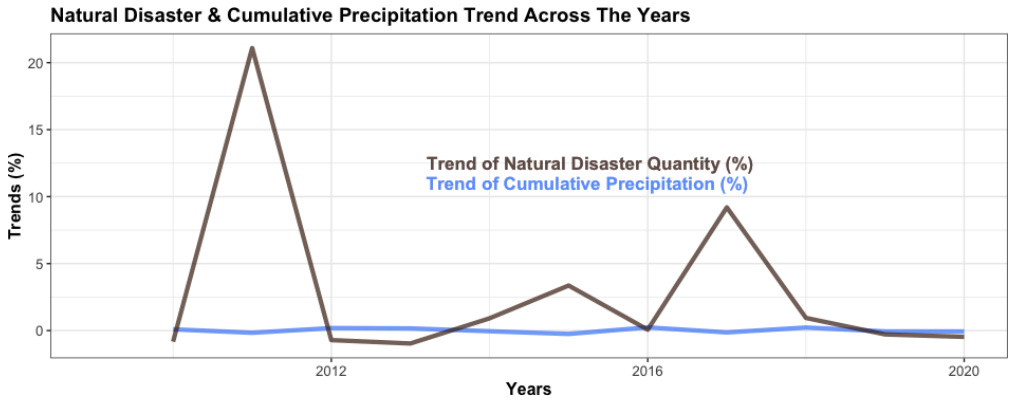
As values are not on the same scale this doesn't gives us pertinent information.
Let's apply a multiplication coefficient on the "Number of Natural Disaster"

```
df2 %>%
    group_by(year) %>%
    summarise(sum_cumul_precipit = sum(cumul_precipit,
        na.rm = TRUE), n_nat_dis = sum(nat_dis)) %>%
    ggplot(aes(x = year)) + geom_line(aes(y = sum_cumul_precipit),
    color = "#4285f6", alpha = 0.7, size = 1.5) + geom_line(aes(y = n_nat_dis *
    10000), color = "#421", alpha = 0.7, size = 1.5) +
    geom_text(aes(x = 2010, y = 2.5e+07, label = "Cumulative Precipitation (in mm and
    ↪  1/10)"),
        color = "#4285f6", alpha = 0.12, hjust = -0.05,
        size = 4.5, fontface = "bold") + geom_text(aes(x = 2010,
    y = 2.8e+07, label = "Number of Natural Disaster"),
    color = "#421", alpha = 0.12, hjust = -0.05, size = 4.5,
    fontface = "bold") + labs(title = "Cumulative Precipitation & Number of Natural
    ↪  Disaster Across The Years",
    x = "Years", y = "Cumulative Precipitation (in mm and 1/10) & Number of natural
    ↪  Disaster") +
    standard_theme
```



It seems that there is some correlation but as it stands,
this graph does not allow us to clearly draw any conclusions.
Can we identify a better correlation between these two predictors based
on their year-to-year trends rather than their direct values?

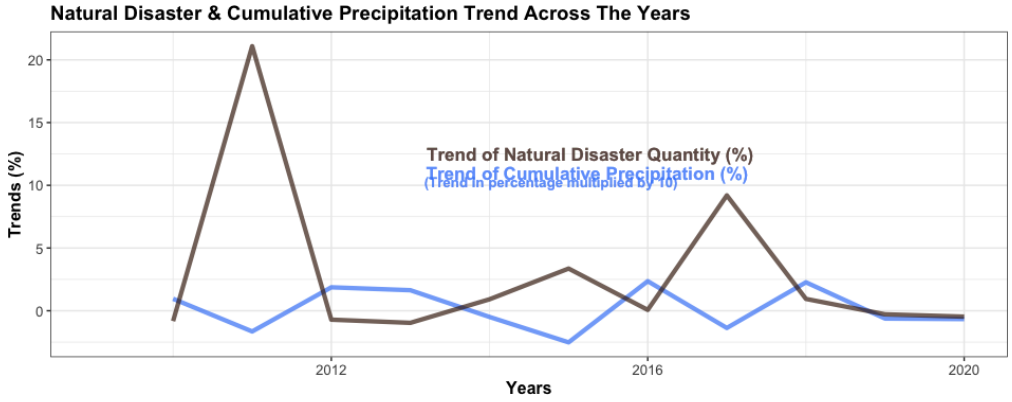
```
df2 %>%
    group_by(year) %>%
    summarise(sum_cumul_precipit = sum(cumul_precipit,
        na.rm = TRUE), n_nat_dis = sum(nat_dis)) %>%
    mutate(trend_cumul_precipit = (sum_cumul_precipit -
        lag(sum_cumul_precipit))/lag(sum_cumul_precipit),
        trend_nat_dis = (n_nat_dis - lag(n_nat_dis))/lag(n_nat_dis)) %>%
    ggplot(aes(x = year)) + geom_line(aes(y = trend_cumul_precipit),
    color = "#4285f6", alpha = 0.7, size = 1.5) + geom_line(aes(y = trend_nat_dis),
    color = "#421", alpha = 0.7, size = 1.5) + geom_text(aes(x = 2013,
    y = 11, label = "Trend of Cumulative Precipitation (%)"),
    color = "#4285f6", alpha = 0.12, hjust = -0.05,
    size = 4.5, fontface = "bold") + geom_text(aes(x = 2013,
    y = 12.5, label = "Trend of Natural Disaster Quantity (%)"),
    color = "#421", alpha = 0.12, hjust = -0.05, size = 4.5,
    fontface = "bold") + labs(title = "Natural Disaster & Cumulative Precipitation Trend
    ↪  Across The Years",
    x = "Years", y = "Trends (%)") + standard_theme
```



These two trends do not have the same scale at all,
let's apply a multiplier coefficient to the trend of the average spring temperature

```
df2 %>%
    group_by(year) %>%
    summarise(sum_cumul_precipit = sum(cumul_precipit,
        na.rm = TRUE), n_nat_dis = sum(nat_dis)) %>%
    mutate(trend_cumul_precipit = ((sum_cumul_precipit -
        lag(sum_cumul_precipit))/lag(sum_cumul_precipit)) *
        10, trend_nat_dis = (n_nat_dis - lag(n_nat_dis))/lag(n_nat_dis)) %>%
    ggplot(aes(x = year)) + geom_line(aes(y = trend_cumul_precipit),
    color = "#4285f6", alpha = 0.7, size = 1.5) + geom_line(aes(y = trend_nat_dis),
    color = "#421", alpha = 0.7, size = 1.5) + geom_text(aes(x = 2013,
    y = 11, label = "Trend of Cumulative Precipitation (%)"),
    color = "#4285f6", alpha = 0.12, hjust = -0.05,
    size = 4.5, fontface = "bold") + geom_text(aes(x = 2013,
    y = 10.3, label = "(Trend in percentage multiplied by 10)"),
    color = "#4285f6", alpha = 0.12, hjust = -0.055,
    size = 3.5, fontface = "bold") + geom_text(aes(x = 2013,
    y = 12.5, label = "Trend of Natural Disaster Quantity (%)"),
    color = "#421", alpha = 0.12, hjust = -0.05, size = 4.5,
```

```
    fontface = "bold") + labs(title = "Natural Disaster & Cumulative Precipitation Trend
    ↪  Across The Years",
    x = "Years", y = "Trends (%)") + standard_theme
```



Let's analyze the correlation of these two predictors

```
df2 %>%
    group_by(year) %>%
    summarise(sum_cumul_precipit = sum(cumul_precipit,
        na.rm = TRUE), n_nat_dis = sum(nat_dis)) %>%
    mutate(trend_cumul_precipit = (sum_cumul_precipit -
        lag(sum_cumul_precipit))/lag(sum_cumul_precipit),
        trend_nat_dis = (n_nat_dis - lag(n_nat_dis))/lag(n_nat_dis)) %>%
    summarise(pearson_cor = cor(trend_cumul_precipit,
        trend_nat_dis, method = "pearson", use = "pairwise.complete.obs"),
        spearman_cor = cor(trend_cumul_precipit, trend_nat_dis,
            method = "spearman", use = "pairwise.complete.obs"))
```

```
# A tibble: 1 x 2
  pearson_cor spearman_cor
        <dbl>        <dbl>
1      -0.542       -0.518
```

We see a strong negative correlation between Natural disaster trend and cumulative precipitation. This suggests that a variation of a few mm precipitation from one year to the next, would have a significant impact on the number of natural disasters.

Now let's see if we can extract more insights by analyzing the precipitation trend per city per year.

```
dataframe_trend_precipit<-dataframe_trend%>% # We are using the dataframe we prepared
↪  before, that contains the year 2008 to permit us to calculate 2009's trend.
  mutate(natural_disaster=as.factor(natural_disaster))%>%
  group_by(insee_code)%>%
  arrange(year, .by_group = TRUE) %>%
  mutate(trend_cumul_precipit =
           ((cumul_precipit-lag(cumul_precipit)) / lag(cumul_precipit)))%>%
  ungroup()%>%
  filter(year>2008&
           !is.na(trend_cumul_precipit)&
           is.finite(trend_cumul_precipit))%>%
```

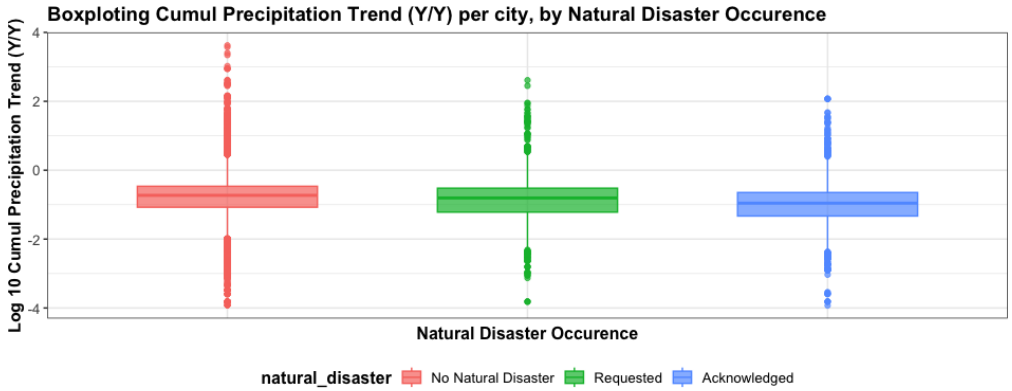
```
  left_join(nat_dis_group,by="insee_code")%>%
  filter(!nat_dis_group==0)%>%select(insee_code,year,trend_cumul_precipit)
```

We are adding the trend to our dataframe

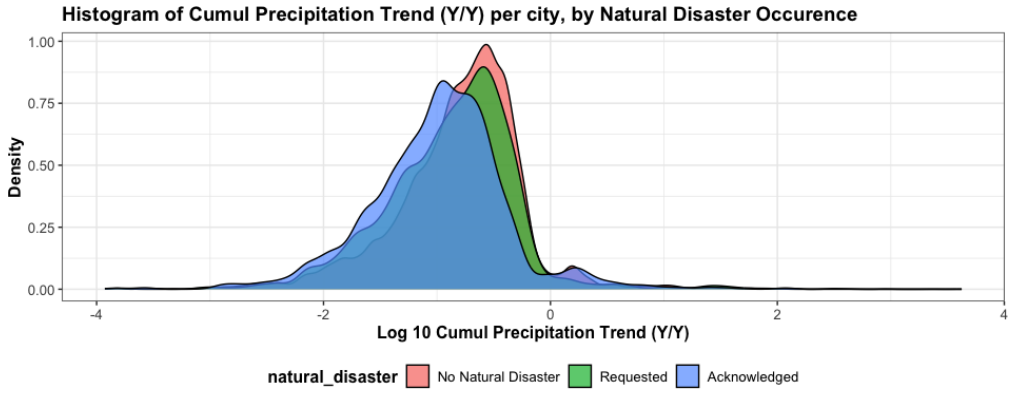
```
df2 <- df2 %>%
    left_join(dataframe_trend_precipit, by = c("insee_code",
        "year"), relationship = "many-to-many")
```

We are now analyzing the trend predictor

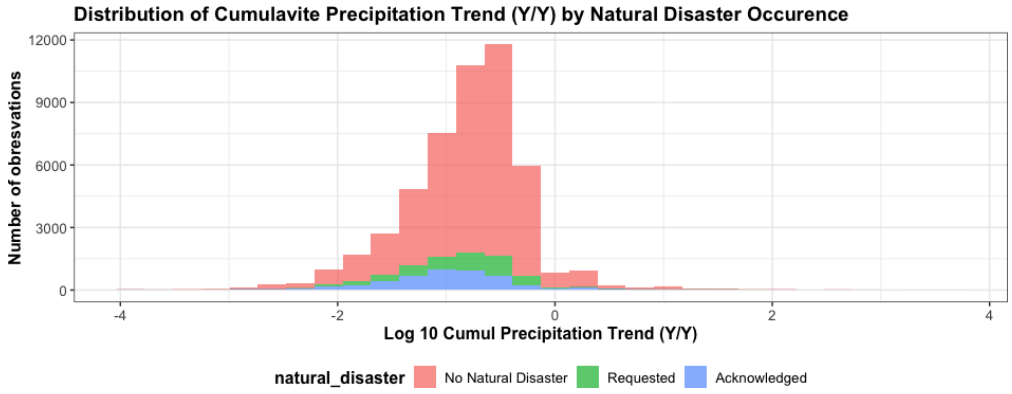
```
df2%>%
  filter(trend_cumul_precipit>-5)%>% # Filtering the outliers to make the Boxplot more
  ↪  readable
  ggplot(aes(natural_disaster,log10(trend_cumul_precipit),
             fill=natural_disaster,group=natural_disaster, color=natural_disaster))+
  geom_boxplot(alpha=.7,  width = 0.6)+
  labs(title = "Boxploting Cumul Precipitation Trend (Y/Y) per city, by Natural Disaster
  ↪  Occurence",
       x = "Natural Disaster Occurence",
       y = "Log 10 Cumul Precipitation Trend (Y/Y)")+
  standard_theme+
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```



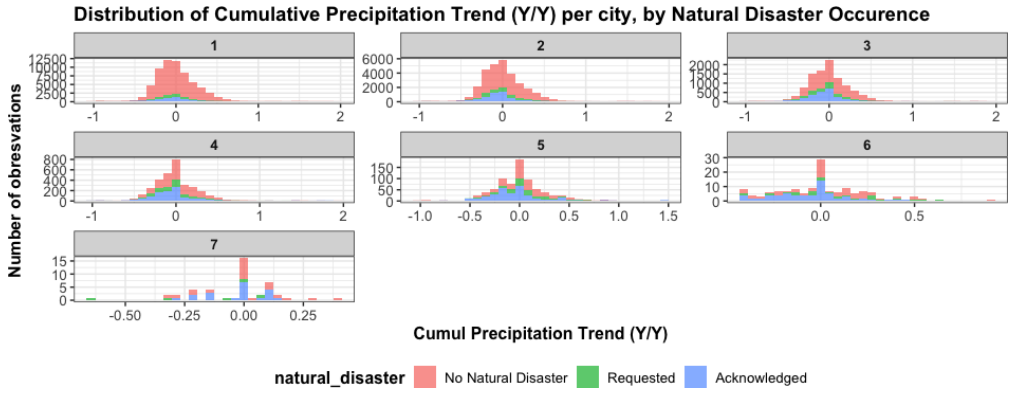

```
df2%>%
  filter(trend_cumul_precipit>-5)%>% # Filtering the outliers to make the Boxplot more
  ↪  readable
  ggplot(aes(log10(trend_cumul_precipit),fill=natural_disaster))+
  geom_density(alpha=.7)+
  labs(title = "Histogram of Cumul Precipitation Trend (Y/Y) per city, by Natural
  ↪  Disaster Occurence",
       x = "Log 10 Cumul Precipitation Trend (Y/Y)",
       y = "Density")+
  standard_theme
```




```
df2%>%
  filter(trend_cumul_precipit>-5)%>% # Filtering the outliers to make the Boxplot more
  ↪  readable
  ggplot(aes(log10(trend_cumul_precipit),fill=natural_disaster))+
  geom_histogram(alpha=.7)+
  labs(title = "Distribution of Cumulavite Precipitation Trend (Y/Y) by Natural Disaster
  ↪  Occurence",
       x = "Log 10 Cumul Precipitation Trend (Y/Y)",
       y = "Number of obresvations")+
  standard_theme
```




```
df2%>%
  filter(trend_cumul_precipit<2)%>%
  ggplot(aes(trend_cumul_precipit,fill=natural_disaster))+
  geom_histogram(alpha=.7,position="stack")+
  facet_wrap(~nat_dis_group,scales="free")+
  labs(title = "Distribution of Cumulative Precipitation Trend (Y/Y) per city, by Natural
  ↪  Disaster Occurence",
       x = "Cumul Precipitation Trend (Y/Y)",
       y = "Number of obresvations")+
  standard_theme
```



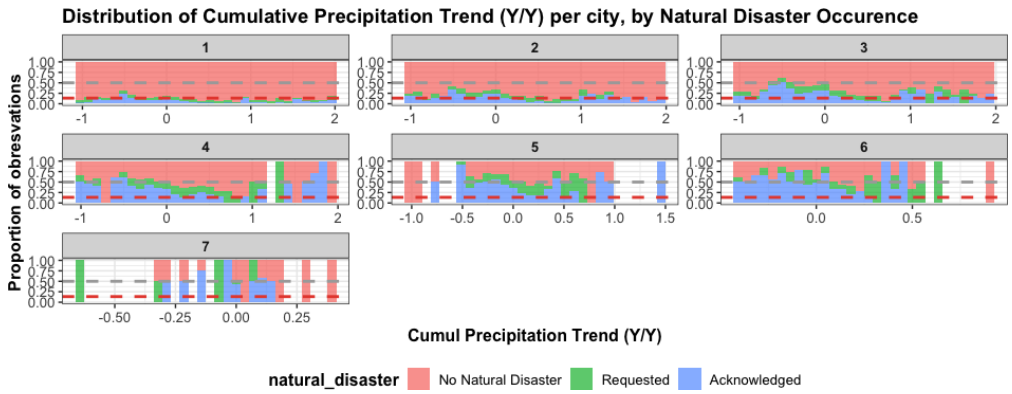
As a reminder we know that the majority of lines in df2 is not acknowledged natural disaster

```
global_mean_nat_dis <- mean(df2$natural_disaster ==
    "Acknowledged")
global_mean_nat_dis
```

```
[1] 0.1345124
```

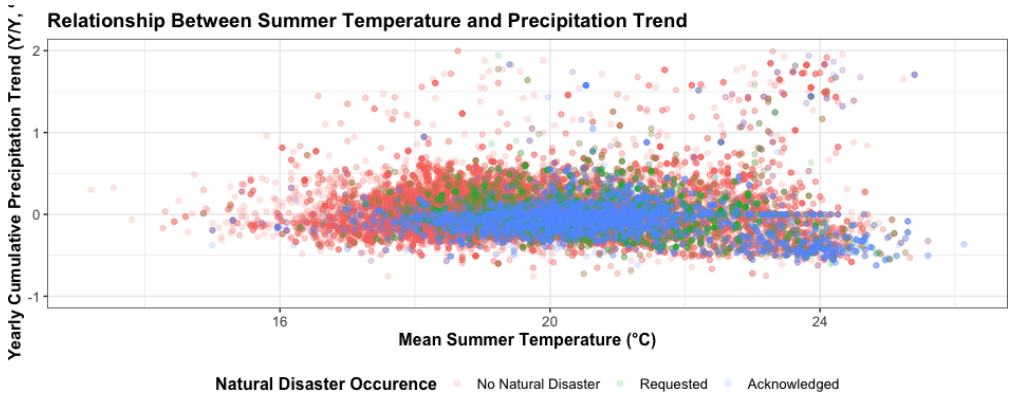
We have only 12.9% of the lines akwnoledged as natural disaster
(As a reminder, df2 is filtered on all cities having made at least one request for natural disasters)

```
df2 %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(trend_cumul_precipit, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "fill") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Distribution of
    ↪  Cumulative Precipitation Trend (Y/Y) per city, by Natural Disaster Occurence",
    x = "Cumul Precipitation Trend (Y/Y)", y = "Proportion of obresvations") +
    standard_theme + geom_hline(yintercept = global_mean_nat_dis,
    color = "#E74C3C", linetype = "dashed", size = 1) +
    geom_hline(yintercept = 0.5, color = "darkgrey",
        linetype = "dashed", size = 1)
```

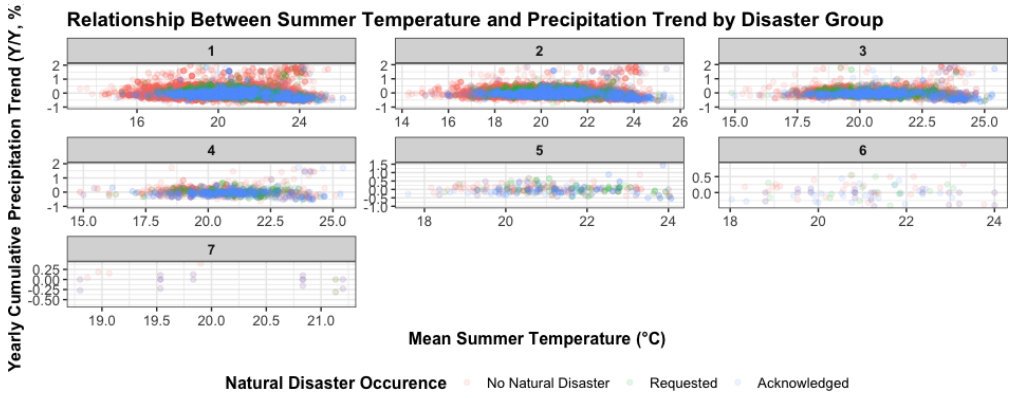



```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_temp_summer, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
```

```
    labs(title = "Relationship Between Summer Temperature and Precipitation Trend",
        x = "Mean Summer Temperature (°C)", y = "Yearly Cumulative Precipitation Trend
        ↪  (Y/Y, %)",
        color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_temp_summer, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.1) +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Relationship Between
    ↪  Summer Temperature and Precipitation Trend by Disaster Group",
    x = "Mean Summer Temperature (°C)", y = "Yearly Cumulative Precipitation Trend (Y/Y,
    ↪  %)",
    color = "Natural Disaster Occurence") + standard_theme
```



These graphs allow us to make the following observations:
- The “acknowledged” cases have a normal precipitation trend distribution centered around -1 (compared to -0.5 for cases without natural disaster requests and denied cases)
- Following the trends, the disaster rates deviate from the average

This supports the relevance of both the trend predictor and the city group segmentation.
It also suggests that decision tree-based classification could be a suitable modeling approach.

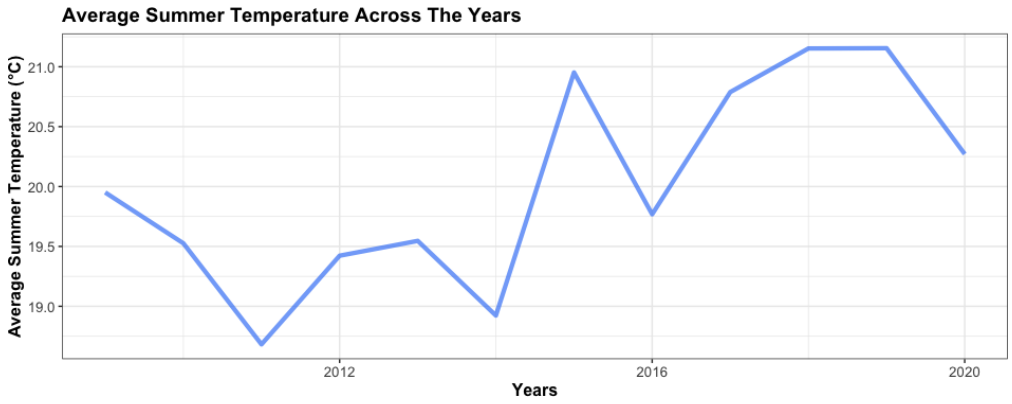
Let’s add the newly created predictor:

```
selected_predictors <- c(selected_predictors, "trend_cumul_precipit")
selected_predictors
```

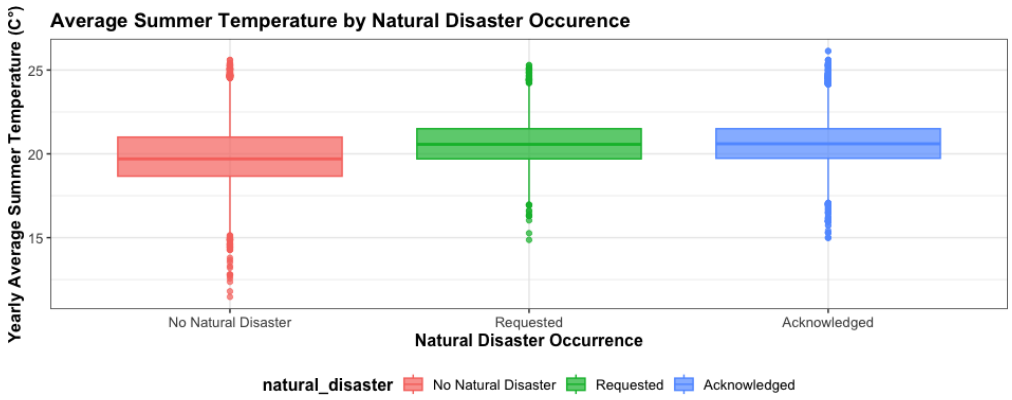
```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp"            "mean_temp_winter"     "mean_humid"
 [7] "mean_ET"              "mean_radiation"       "n_high_risk_houses"
[10] "natural_disaster"     "trend_cumul_precipit"
```

4.4 Analyzing "mean_temp_summer"

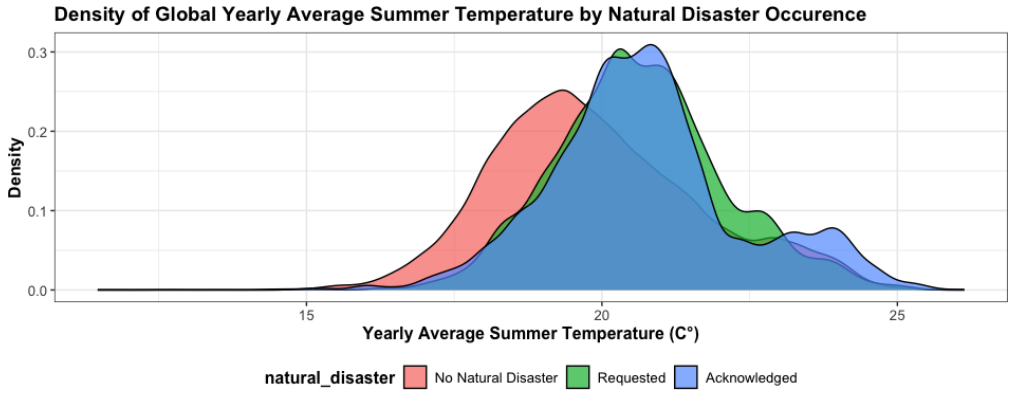
```
df2 %>%
    group_by(year) %>%
    summarise(yearly_mean_temp_summer = mean(mean_temp_summer,
        na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_mean_temp_summer)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Average Summer Temperature Across The
    ↪  Years",
    x = "Years", y = "Average Summer Temperature (°C)") +
    standard_theme
```



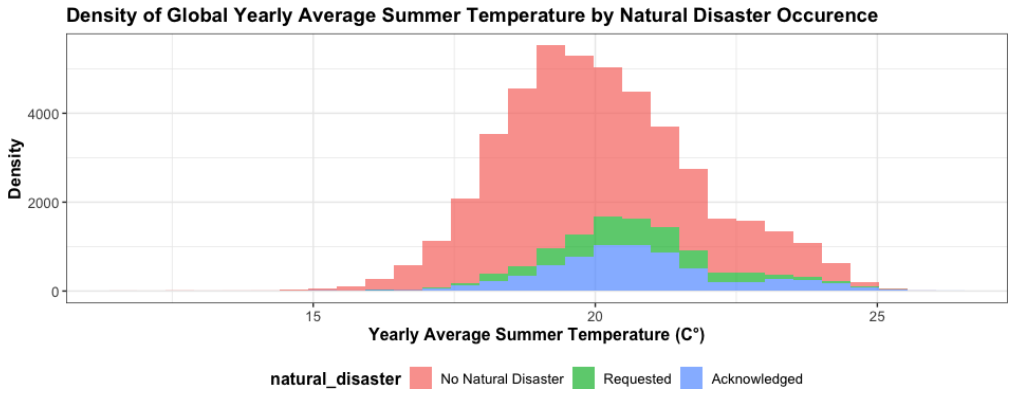

```
df2 %>%
    ggplot(aes(natural_disaster, mean_temp_summer,
        fill = natural_disaster, color = natural_disaster)) +
    geom_boxplot(alpha = 0.7) + labs(title = "Average Summer Temperature by Natural
    ↪  Disaster Occurence",
    x = "Natural Disaster Occurrence", y = "Yearly Average Summer Temperature (C°)") +
    standard_theme
```




```
df2 %>%
    ggplot(aes(mean_temp_summer, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Global Yearly Average Summer
    ↪  Temperature by Natural Disaster Occurence",
    x = "Yearly Average Summer Temperature (C°)",
    y = "Density") + standard_theme
```

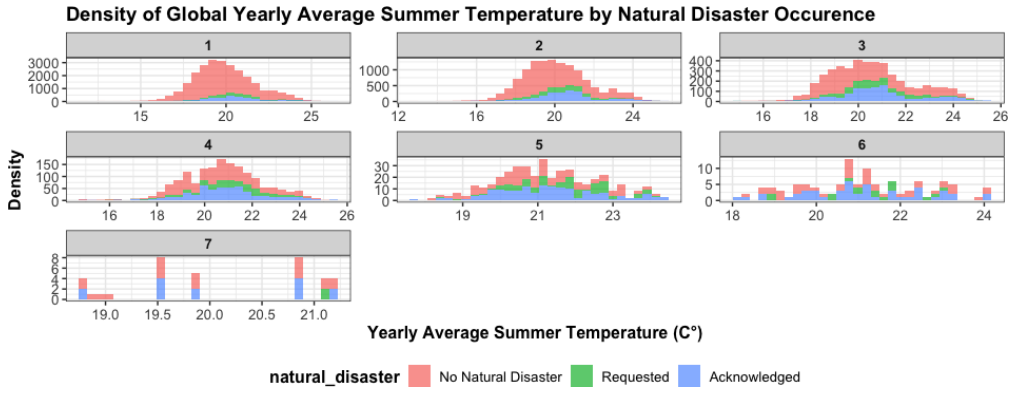



```
df2 %>%
    ggplot(aes(mean_temp_summer, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + labs(title = "Density of Global Yearly Average Summer
    ↪  Temperature by Natural Disaster Occurence",
    x = "Yearly Average Summer Temperature (C°)",
    y = "Density") + standard_theme
```

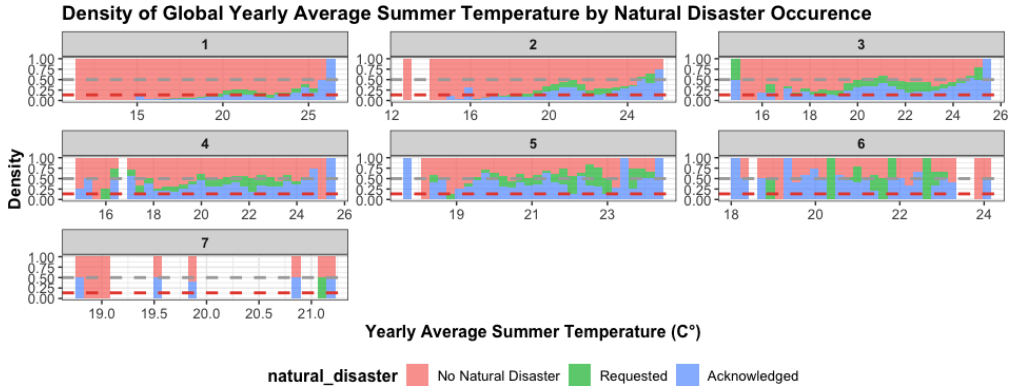



```
df2 %>%
    ggplot(aes(mean_temp_summer, fill = natural_disaster)) +
```

```
    geom_histogram(alpha = 0.7, position = "stack") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Density of Global Yearly
    ↪  Average Summer Temperature by Natural Disaster Occurence",
    x = "Yearly Average Summer Temperature (C°)",
    y = "Density") + standard_theme
```



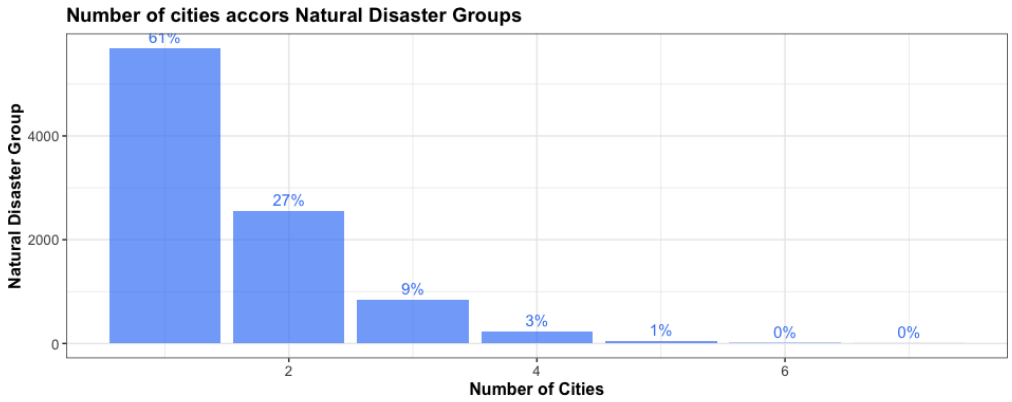

```
df2 %>%
    ggplot(aes(mean_temp_summer, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "fill") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Density of Global Yearly
    ↪  Average Summer Temperature by Natural Disaster Occurence",
    x = "Yearly Average Summer Temperature (C°)",
    y = "Density") + standard_theme + geom_hline(yintercept = global_mean_nat_dis,
    color = "#E74C3C", linetype = "dashed", size = 1) +
    geom_hline(yintercept = 0.5, color = "darkgrey",
        linetype = "dashed", size = 1)
```



These graphs allow us to make the following observations:
- The average summer temperature is generally constantly increasing
- That the “requested” and “acknowledged” cases have a normal distribution centered on an average summer temperature around 20.5°C (compared to 19°C for cases without natural disasters)
- Above 22.5°C on average in summer, the “acknowledged” cases are above average.

4.5 Analysing "nat_dis_group"

```
df2 %>%
    group_by(nat_dis_group) %>%
    summarise(city_count = n_distinct(insee_code)) %>%
    mutate(percentage = city_count/sum(city_count)) %>%
    ggplot(aes(nat_dis_group, city_count)) + geom_bar(fill = "#4285f6",
    alpha = 0.7, stat = "identity") + labs(title = "Number of cities accors Natural
    ↪  Disaster Groups",
    x = "Number of Cities", y = "Natural Disaster Group") +
    geom_text(aes(label = scales::percent(percentage,
        accuracy = 1)), vjust = -0.5, size = 4, color = "#4285f6") +
    standard_theme
```

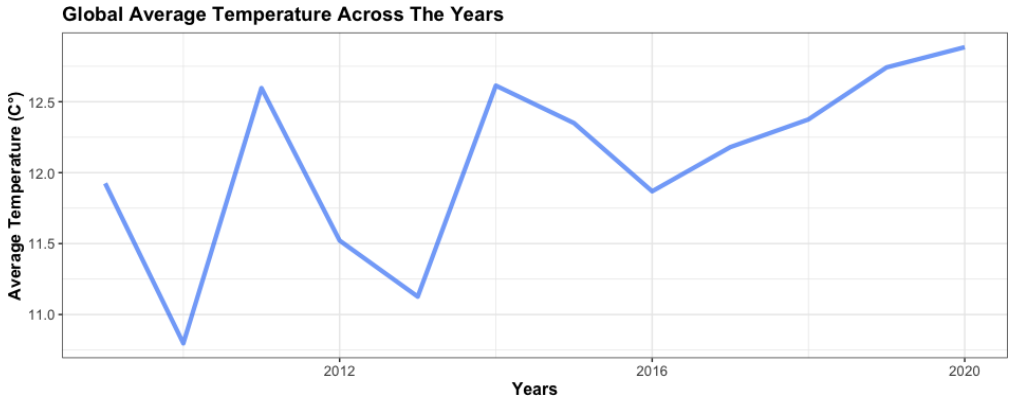


As saw earlier, from 2009 to 2020 (when city with no natural disaster are filtered out) : - 61% of the cities had only 1 Natural disaster,
- 88% of the cities had 2 natural disaster or less,
- 97% of the cities had 3 natural disaster or less,
- 99% of the cities had 4 natural disaster or less.

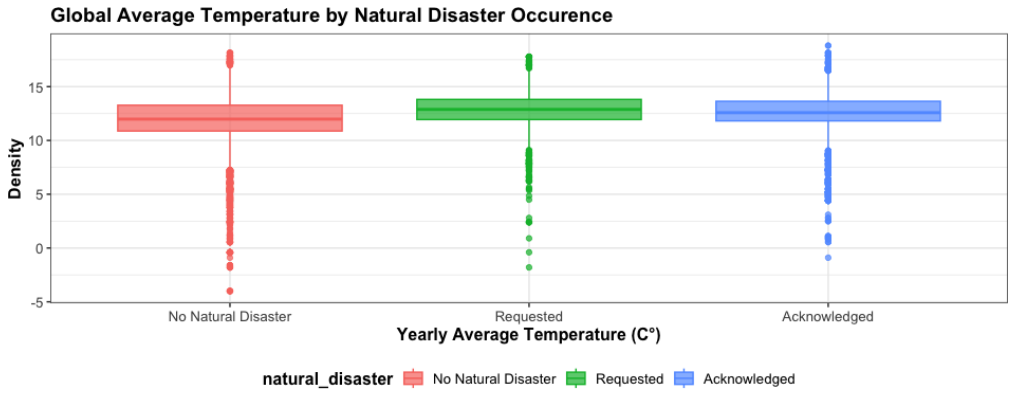
Since "nat_dis_group" is created from the data we are trying to predict, when creating our future "train_set" and "test_set", we will need to recalculate this predictor in each set to avoid using test data to train our model.

4.6 Analyzing "mean_temp"

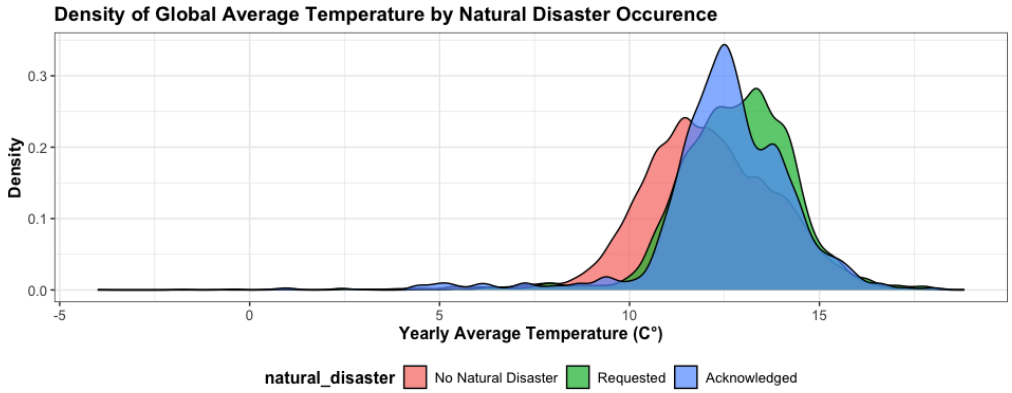
```
df2 %>%
    group_by(year) %>%
    summarise(yearly_mean_temp = mean(mean_temp, na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_mean_temp)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Global Average Temperature Across The
    ↪  Years",
    x = "Years", y = "Average Temperature (C°)") +
    standard_theme
```




```
df2 %>%
    ggplot(aes(natural_disaster, mean_temp, fill = natural_disaster,
        color = natural_disaster)) + geom_boxplot(alpha = 0.7) +
    labs(title = "Global Average Temperature by Natural Disaster Occurence",
        x = "Yearly Average Temperature (C°)", y = "Density") +
    standard_theme
```

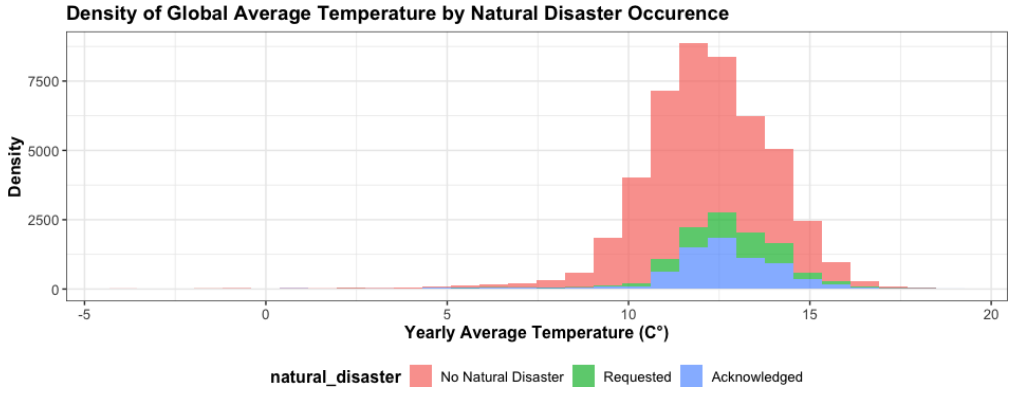



```
df2 %>%
    ggplot(aes(mean_temp, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Global Average Temperature by
    ↪  Natural Disaster Occurence",
    x = "Yearly Average Temperature (C°)", y = "Density") +
    standard_theme
```

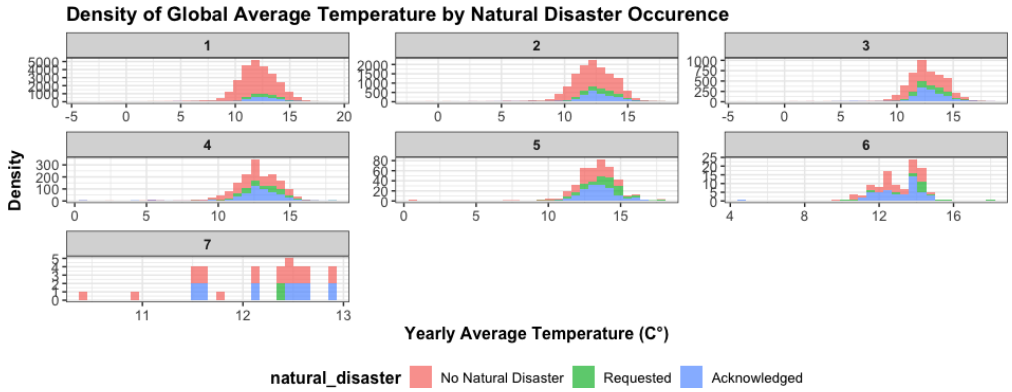



```
df2 %>%
    ggplot(aes(mean_temp, fill = natural_disaster)) +
```

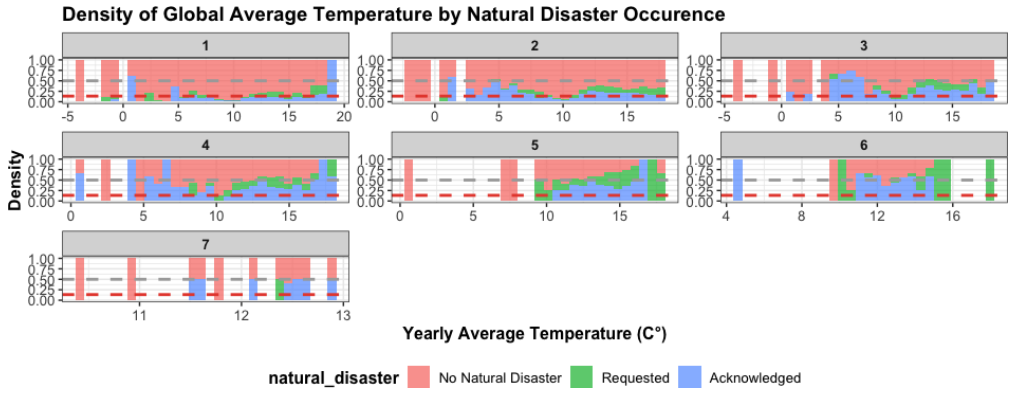
```
    geom_histogram(alpha = 0.7) + labs(title = "Density of Global Average Temperature by
    ↪  Natural Disaster Occurence",
    x = "Yearly Average Temperature (C°)", y = "Density") +
    standard_theme
```



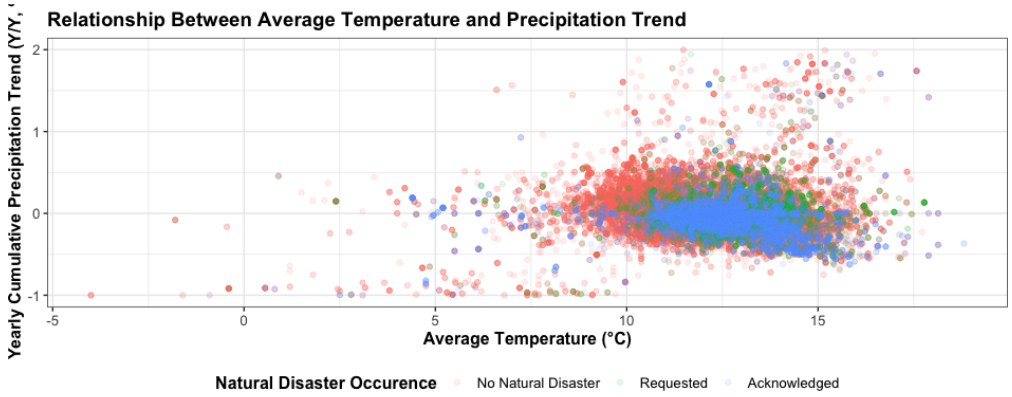

```
df2 %>%
    ggplot(aes(mean_temp, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "stack") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Density of Global Average
    ↪  Temperature by Natural Disaster Occurence",
    x = "Yearly Average Temperature (C°)", y = "Density") +
    standard_theme
```



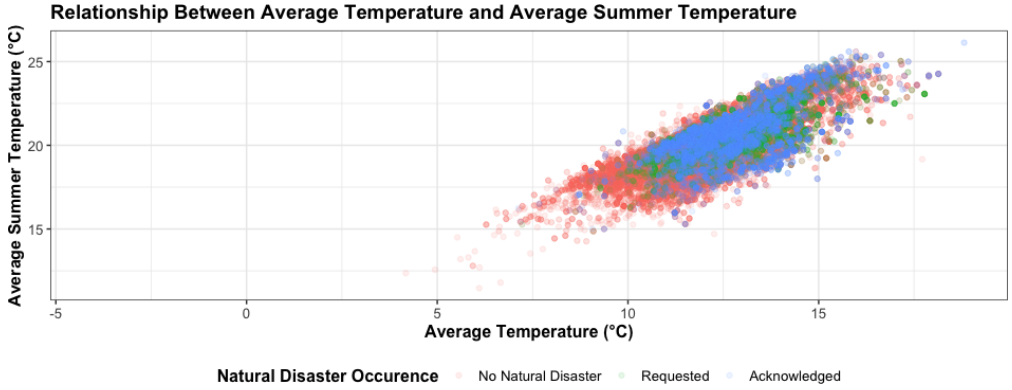

```
df2 %>%
    ggplot(aes(mean_temp, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "fill") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Density of Global Average
    ↪  Temperature by Natural Disaster Occurence",
    x = "Yearly Average Temperature (C°)", y = "Density") +
    standard_theme + geom_hline(yintercept = global_mean_nat_dis,
    color = "#E74C3C", linetype = "dashed", size = 1) +
    geom_hline(yintercept = 0.5, color = "darkgrey",
        linetype = "dashed", size = 1)
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_temp, trend_cumul_precipit, color = natural_disaster)) +
    geom_point(alpha = 0.1) + labs(title = "Relationship Between Average Temperature and
    ↪  Precipitation Trend",
    x = "Average Temperature (°C)", y = "Yearly Cumulative Precipitation Trend (Y/Y, %)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_temp, mean_temp_summer, color = natural_disaster)) +
    geom_point(alpha = 0.1) + labs(title = "Relationship Between Average Temperature and
    ↪  Average Summer Temperature",
    x = "Average Temperature (°C)", y = "Average Summer Temperature (°C)",
    color = "Natural Disaster Occurence") + standard_theme
```



This predictor gives us much less information than "mea_temp_summer", it seems more sensible to remove it from our selected predictors to avoid less relevant redundancy in our models.

```
selected_predictors
```

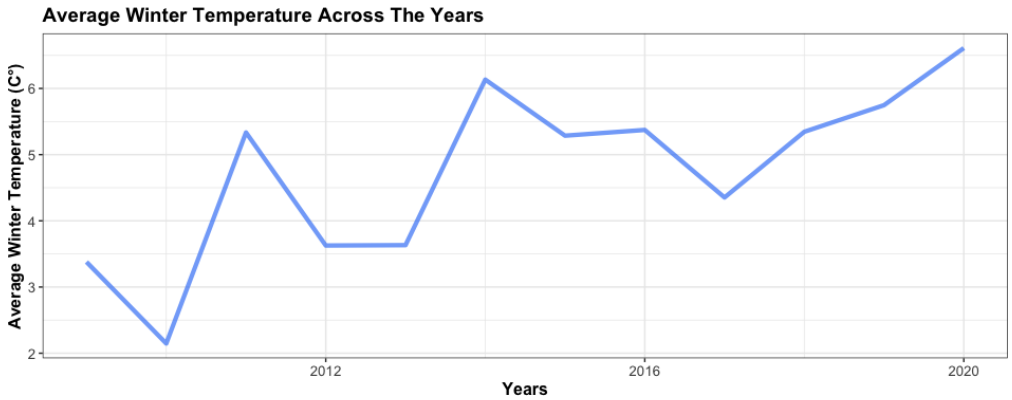
```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp"            "mean_temp_winter"     "mean_humid"
 [7] "mean_ET"              "mean_radiation"       "n_high_risk_houses"
[10] "natural_disaster"     "trend_cumul_precipit"
```

```
selected_predictors <- selected_predictors[-which(selected_predictors ==
    "mean_temp")]
selected_predictors
```

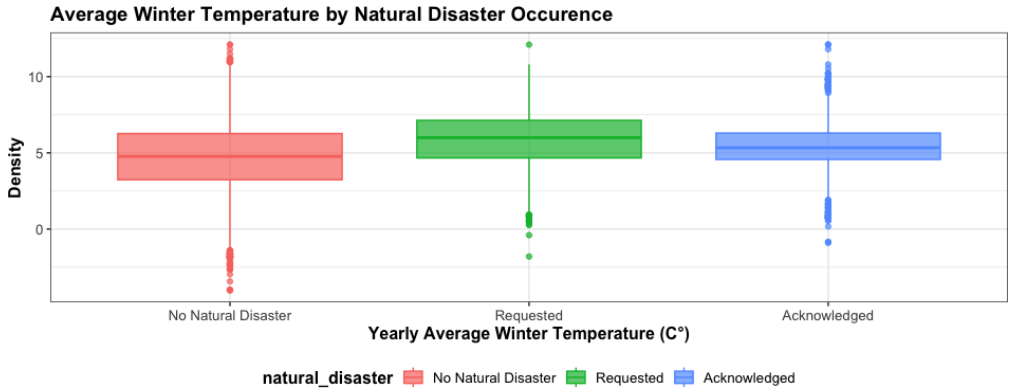
```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp_winter"     "mean_humid"           "mean_ET"
 [7] "mean_radiation"       "n_high_risk_houses"   "natural_disaster"
[10] "trend_cumul_precipit"
```

4.7 Analyzing "mean_temp_winter"

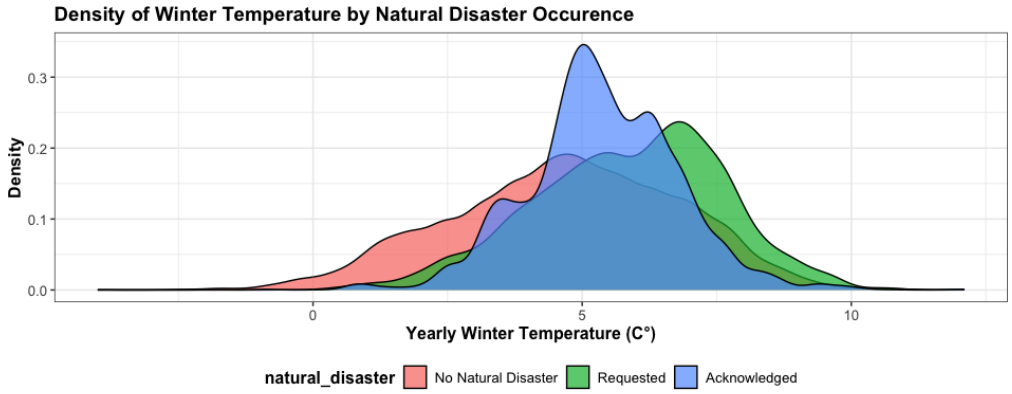
```
df2 %>%
    group_by(year) %>%
    summarise(yearly_mean_temp_winter = mean(mean_temp_winter,
        na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_mean_temp_winter)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Average Winter Temperature Across The
    ↪  Years",
    x = "Years", y = "Average Winter Temperature (C°)") +
    standard_theme
```



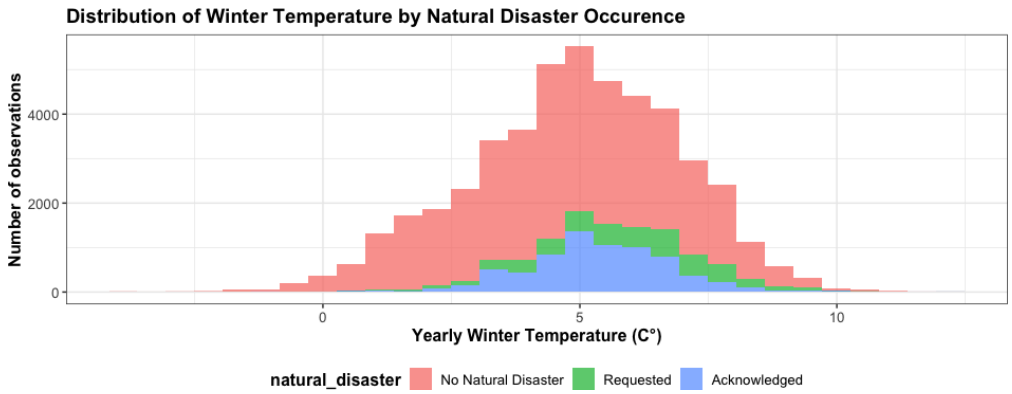

```
df2 %>%
    ggplot(aes(natural_disaster, mean_temp_winter,
        fill = natural_disaster, color = natural_disaster)) +
    geom_boxplot(alpha = 0.7) + labs(title = "Average Winter Temperature by Natural
    ↪  Disaster Occurence",
    x = "Yearly Average Winter Temperature (C°)",
    y = "Density") + standard_theme
```




```
df2 %>%
    ggplot(aes(mean_temp_winter, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Winter Temperature by Natural
    ↪  Disaster Occurence",
    x = "Yearly Winter Temperature (C°)", y = "Density") +
    standard_theme
```

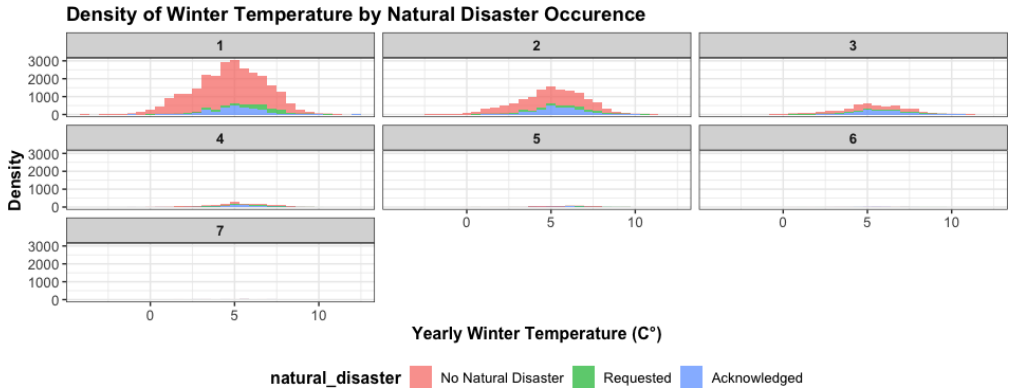



```
df2 %>%
    ggplot(aes(mean_temp_winter, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + labs(title = "Distribution of Winter Temperature by
    ↪  Natural Disaster Occurence",
    x = "Yearly Winter Temperature (C°)", y = "Number of observations") +
    standard_theme
```

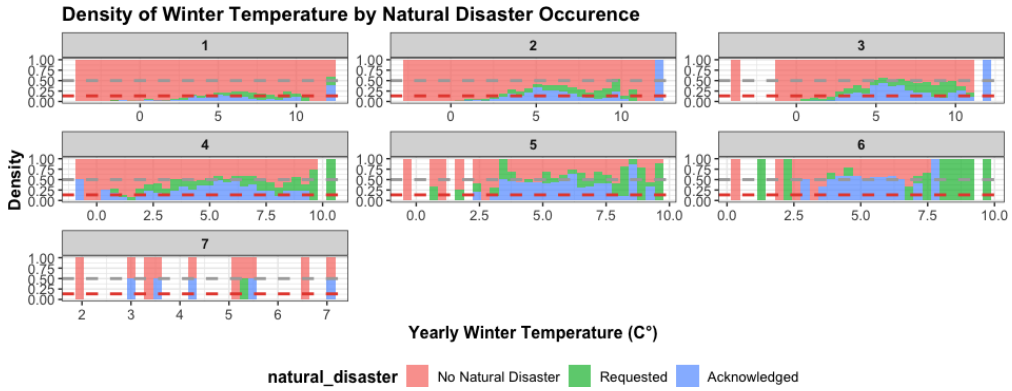



```
df2 %>%
    ggplot(aes(mean_temp_winter, fill = natural_disaster)) +
```

```
    geom_histogram(alpha = 0.7, position = "stack") +
    facet_wrap(~nat_dis_group) + labs(title = "Density of Winter Temperature by Natural
    ↪  Disaster Occurence",
    x = "Yearly Winter Temperature (C°)", y = "Density") +
    standard_theme
```

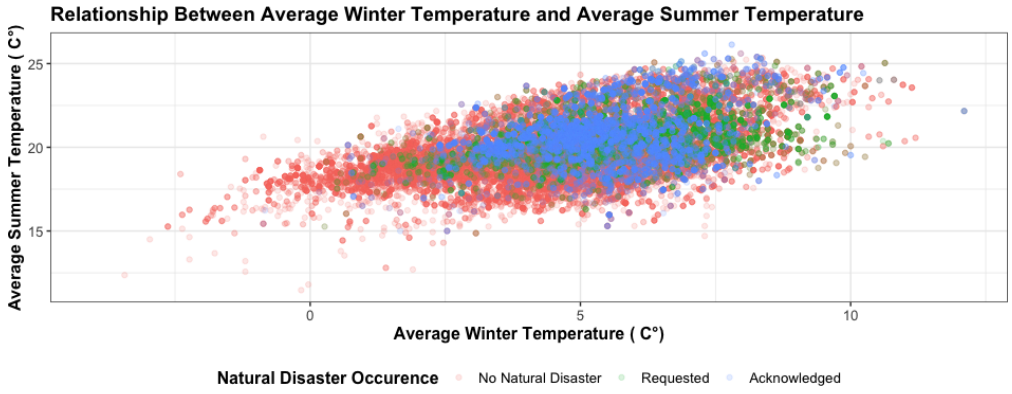



```
df2 %>%
    ggplot(aes(mean_temp_winter, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "fill") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Density of Winter
    ↪  Temperature by Natural Disaster Occurence",
    x = "Yearly Winter Temperature (C°)", y = "Density") +
    standard_theme + geom_hline(yintercept = global_mean_nat_dis,
    color = "#E74C3C", linetype = "dashed", size = 1) +
    geom_hline(yintercept = 0.5, color = "darkgrey",
        linetype = "dashed", size = 1)
```

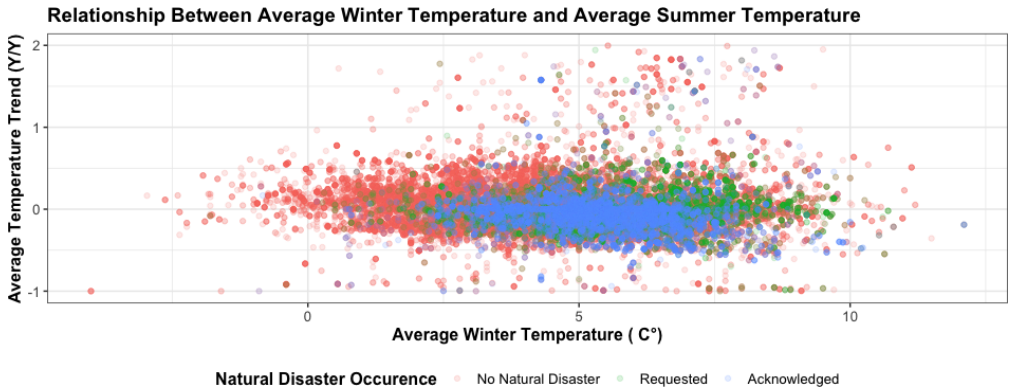


Let's compare this predictor with those already studied previously

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_temp_winter, mean_temp_summer,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    labs(title = "Relationship Between Average Winter Temperature and Average Summer
    ↪  Temperature",
        x = "Average Winter Temperature ( C°)", y = "Average Summer Temperature ( C°)",
        color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_temp_winter, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    labs(title = "Relationship Between Average Winter Temperature and Average Summer
    ↪  Temperature",
        x = "Average Winter Temperature ( C°)", y = "Average Temperature Trend (Y/Y)",
        color = "Natural Disaster Occurence") + standard_theme
```

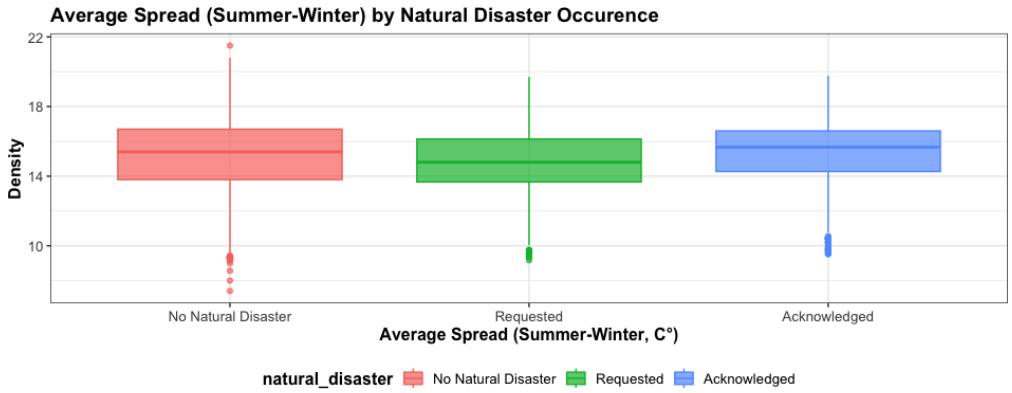


This predictor gives us a lot of information, but can we work with it to get more information out of it?

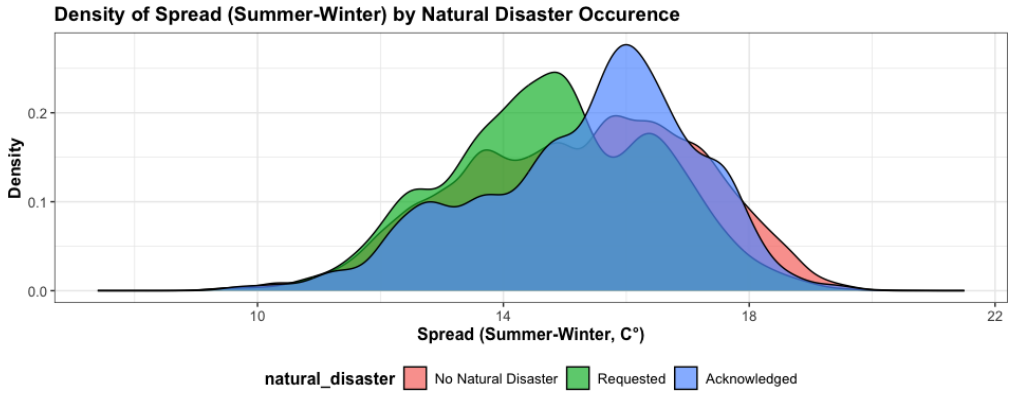
For example by calculating its difference with the average summer temperature?

```
df2 %>%
    group_by(year) %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    summarise(yearly_spread_summer_winter = mean(spread_summer_winter,
        na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_spread_summer_winter)) +
    geom_line(color = "#4285f6", alpha = 0.7, size = 1.5) +
    labs(title = "Average Spread (Summer-Winter) Across The Years",
        x = "Years", y = "Average Spread (Summer-Winter, C°)") +
    standard_theme
```

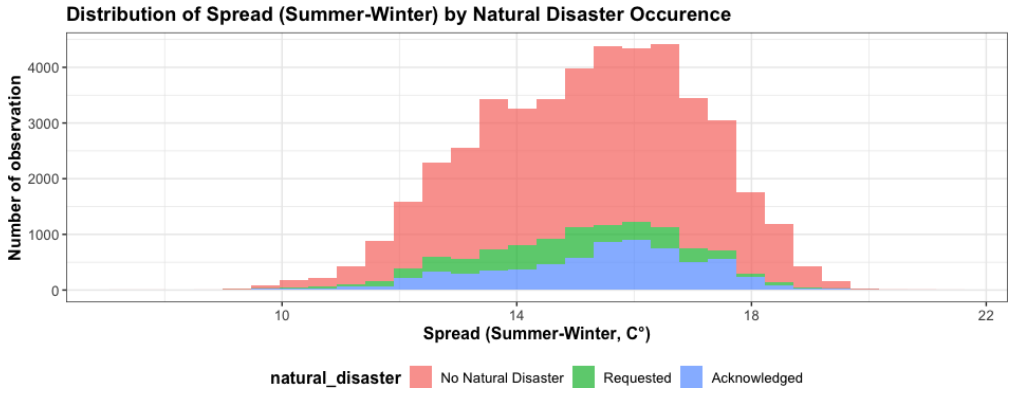

```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    ggplot(aes(natural_disaster, spread_summer_winter,
        fill = natural_disaster, color = natural_disaster)) +
    geom_boxplot(alpha = 0.7) + labs(title = "Average Spread (Summer-Winter) by Natural
    ↪  Disaster Occurence",
    x = "Average Spread (Summer-Winter, C°)", y = "Density") +
    standard_theme
```




```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    ggplot(aes(spread_summer_winter, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Spread (Summer-Winter) by
    ↪  Natural Disaster Occurence",
    x = "Spread (Summer-Winter, C°)", y = "Density") +
    standard_theme
```



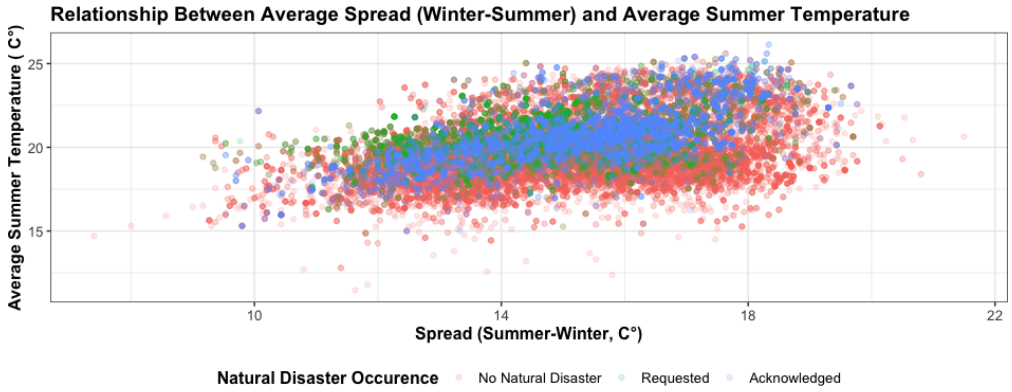

```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    ggplot(aes(spread_summer_winter, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + labs(title = "Distribution of Spread (Summer-Winter) by
    ↪  Natural Disaster Occurence",
    x = "Spread (Summer-Winter, C°)", y = "Number of observation") +
    standard_theme
```



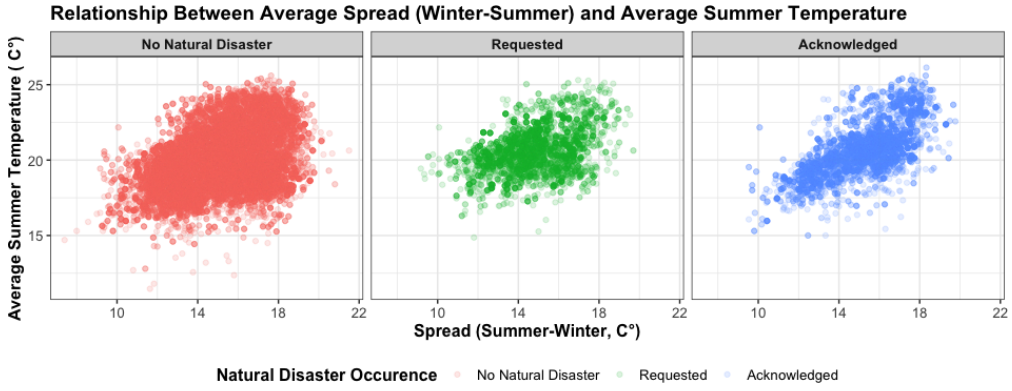
Remarque que la distribution des 'Acknowledged est centré sur un spread plus élevé.

Comparons maintenant ce nouveau paramètre avec les paramètre étudiés précedement

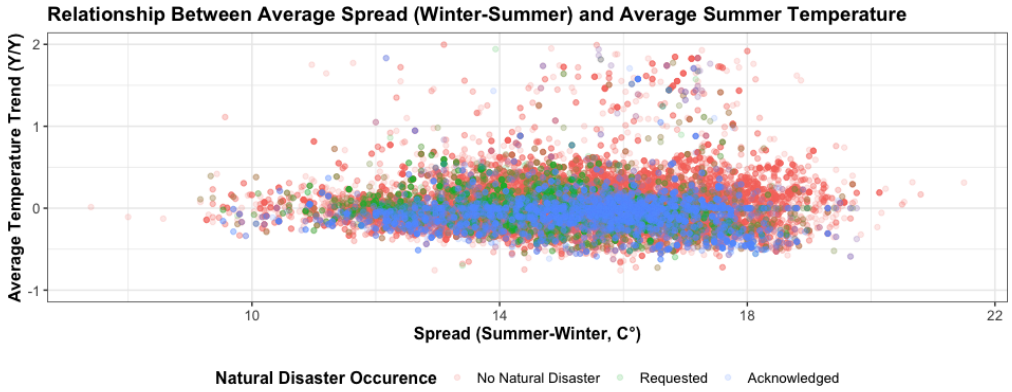
```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(spread_summer_winter, mean_temp_summer,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    labs(title = "Relationship Between Average Spread (Winter-Summer) and Average Summer
    ↪  Temperature",
        x = "Spread (Summer-Winter, C°)", y = "Average Summer Temperature ( C°)",
        color = "Natural Disaster Occurence") + standard_theme
```



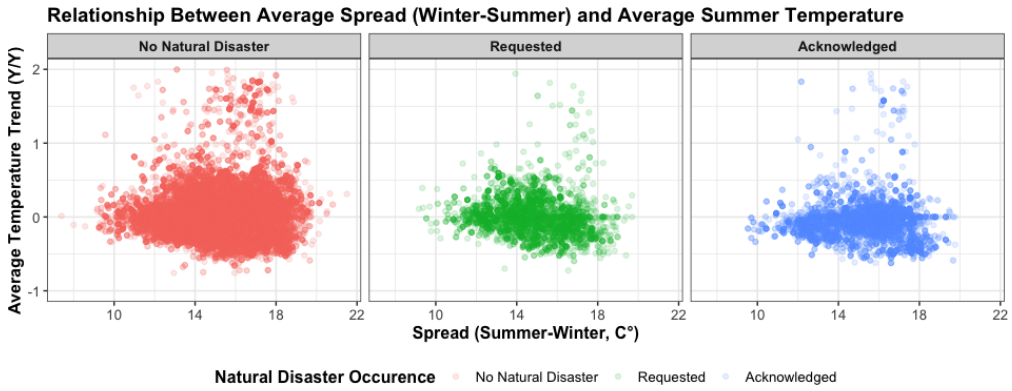

```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(spread_summer_winter, mean_temp_summer,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    facet_wrap(~natural_disaster) + labs(title = "Relationship Between Average Spread
    ↪  (Winter-Summer) and Average Summer Temperature",
    x = "Spread (Summer-Winter, C°)", y = "Average Summer Temperature ( C°)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(spread_summer_winter, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    labs(title = "Relationship Between Average Spread (Winter-Summer) and Average Summer
    ↪  Temperature",
        x = "Spread (Summer-Winter, C°)", y = "Average Temperature Trend (Y/Y)",
        color = "Natural Disaster Occurence") + standard_theme
```

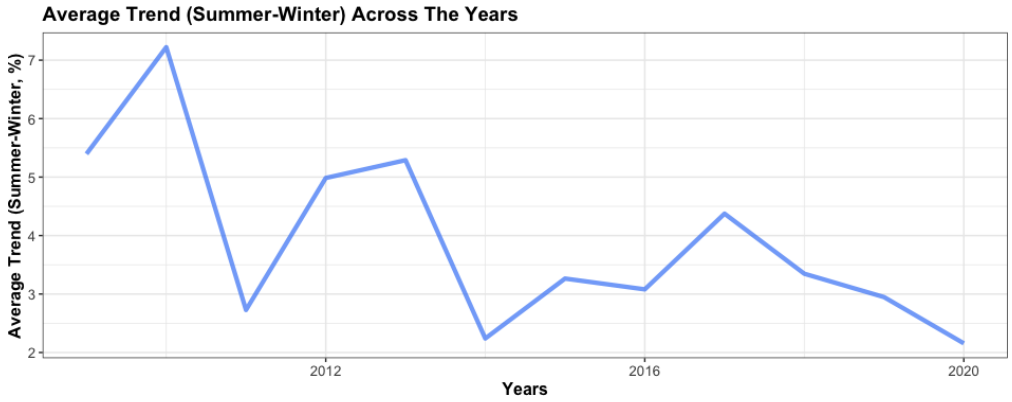



```
df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter) %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(spread_summer_winter, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    facet_wrap(~natural_disaster) + labs(title = "Relationship Between Average Spread
    ↪  (Winter-Summer) and Average Summer Temperature",
    x = "Spread (Summer-Winter, C°)", y = "Average Temperature Trend (Y/Y)",
    color = "Natural Disaster Occurence") + standard_theme
```

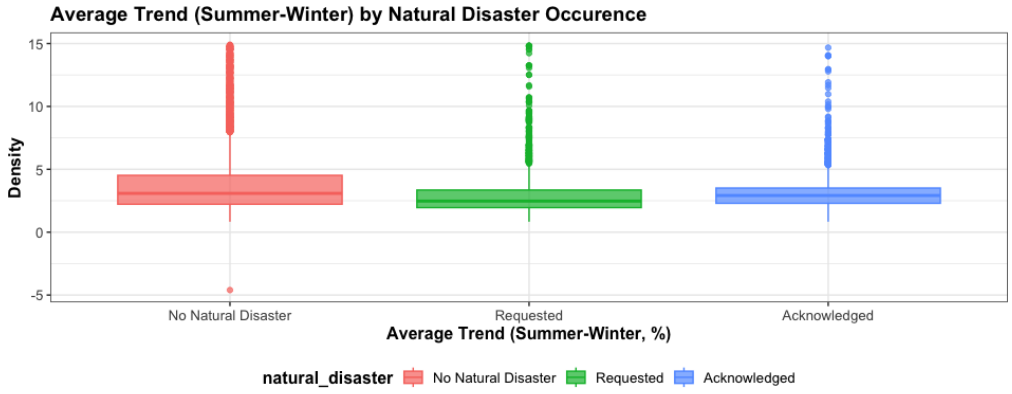


Can we get more information from the trend between summer and winter?

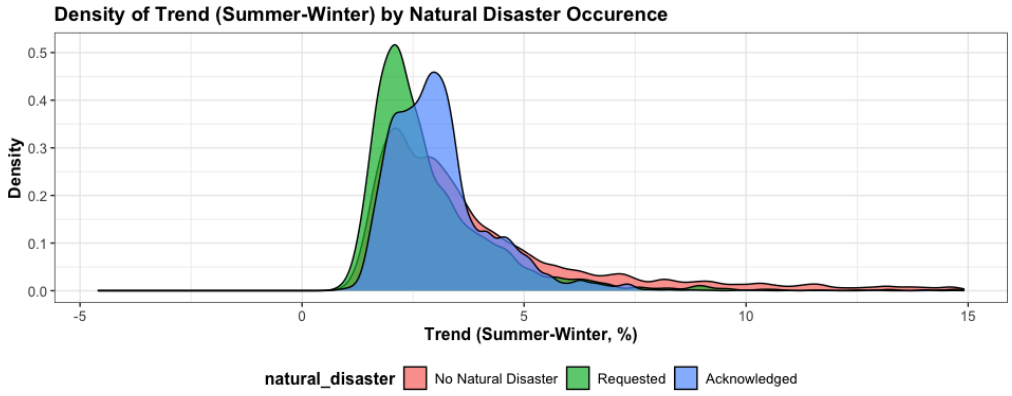
```
df2 %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    group_by(year) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    summarise(yearly_trend_winter_summer = mean(trend_winter_summer,
        na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_trend_winter_summer)) +
    geom_line(color = "#4285f6", alpha = 0.7, size = 1.5) +
    labs(title = "Average Trend (Summer-Winter) Across The Years",
        x = "Years", y = "Average Trend (Summer-Winter, %)") +
    standard_theme
```



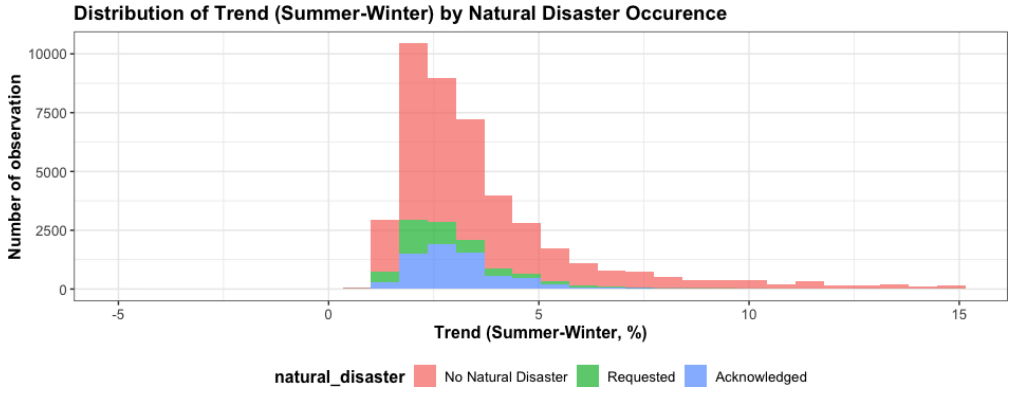

```
df2 %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    ggplot(aes(natural_disaster, trend_winter_summer,
        fill = natural_disaster, color = natural_disaster)) +
    geom_boxplot(alpha = 0.7) + labs(title = "Average Trend (Summer-Winter) by Natural
    ↪  Disaster Occurence",
    x = "Average Trend (Summer-Winter, %)", y = "Density") +
    standard_theme
```



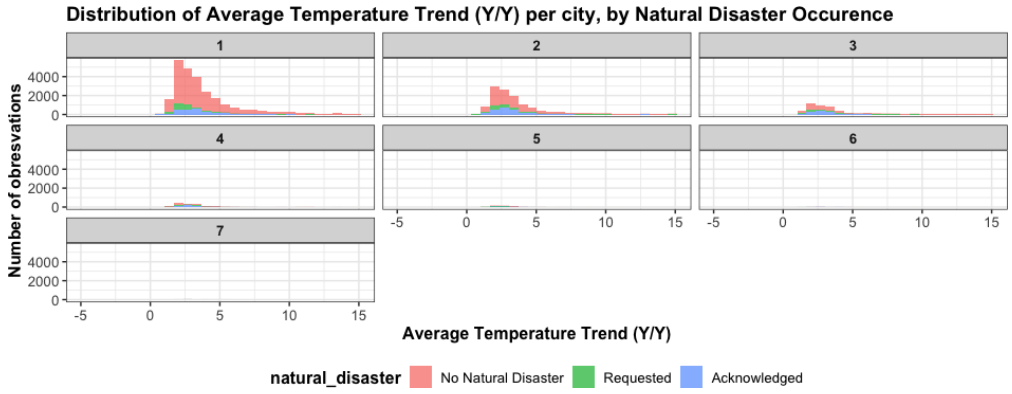

```
df2 %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    ggplot(aes(trend_winter_summer, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Trend (Summer-Winter) by Natural
    ↪  Disaster Occurence",
    x = "Trend (Summer-Winter, %)", y = "Density") +
    standard_theme
```



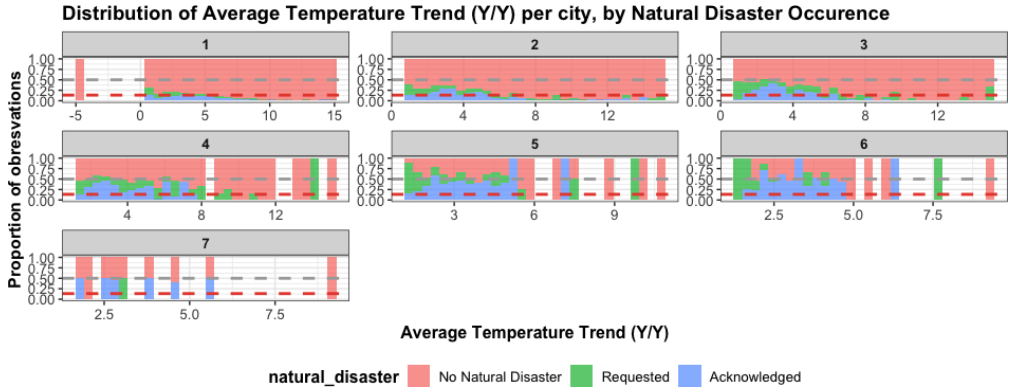

```
df2 %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    ggplot(aes(trend_winter_summer, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + labs(title = "Distribution of Trend (Summer-Winter) by
    ↪  Natural Disaster Occurence",
    x = "Trend (Summer-Winter, %)", y = "Number of observation") +
    standard_theme
```



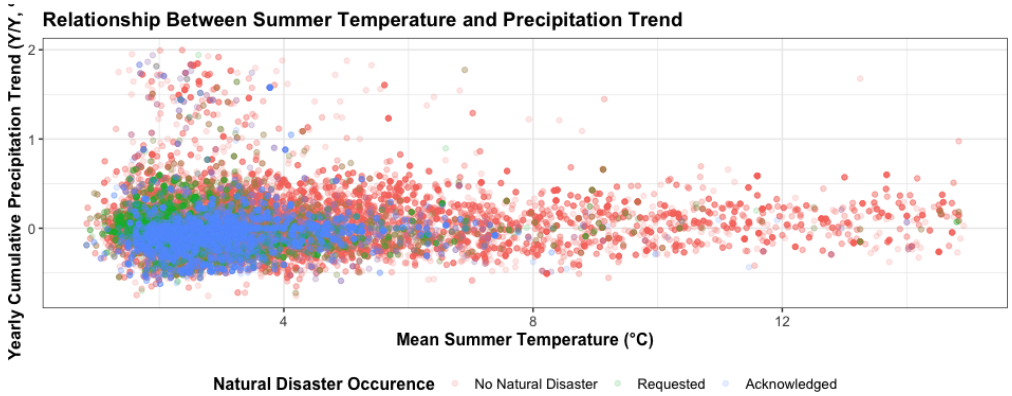

```
df2 %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    ggplot(aes(trend_winter_summer, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "stack") +
    facet_wrap(~nat_dis_group) + labs(title = "Distribution of Average Temperature Trend
    ↪  (Y/Y) per city, by Natural Disaster Occurence",
    x = "Average Temperature Trend (Y/Y)", y = "Number of obresvations") +
    standard_theme
```




```
df2 %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    ggplot(aes(trend_winter_summer, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7, position = "fill") +
    facet_wrap(~nat_dis_group, scales = "free") + labs(title = "Distribution of Average
    ↪  Temperature Trend (Y/Y) per city, by Natural Disaster Occurence",
    x = "Average Temperature Trend (Y/Y)", y = "Proportion of obresvations") +
    standard_theme + geom_hline(yintercept = global_mean_nat_dis,
    color = "#E74C3C", linetype = "dashed", size = 1) +
    geom_hline(yintercept = 0.5, color = "darkgrey",
        linetype = "dashed", size = 1)
```

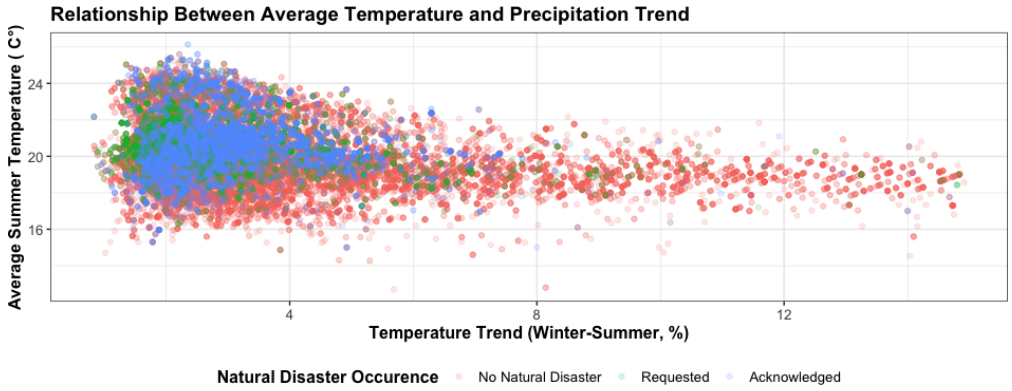



```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(trend_winter_summer, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    labs(title = "Relationship Between Summer Temperature and Precipitation Trend",
        x = "Mean Summer Temperature (°C)", y = "Yearly Cumulative Precipitation Trend
        ↪  (Y/Y, %)",
        color = "Natural Disaster Occurence") + standard_theme
```

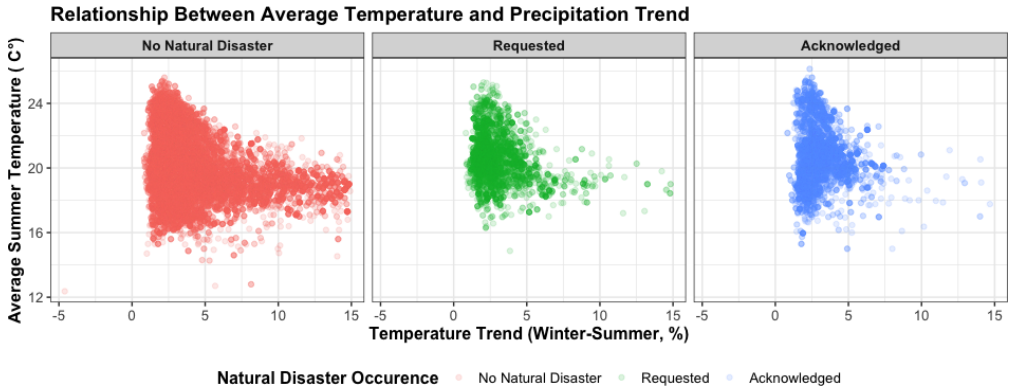


Ce graphique ne nous donne pas beaucoup d'informations

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(trend_winter_summer, mean_temp_summer,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    labs(title = "Relationship Between Average Temperature and Precipitation Trend",
        x = "Temperature Trend (Winter-Summer, %)",
        y = "Average Summer Temperature ( C°)", color = "Natural Disaster Occurence") +
    standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    mutate(trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -5) %>%
    ggplot(aes(trend_winter_summer, mean_temp_summer,
        color = natural_disaster)) + geom_point(alpha = 0.15) +
    facet_wrap(~natural_disaster) + labs(title = "Relationship Between Average
    ↪  Temperature and Precipitation Trend",
    x = "Temperature Trend (Winter-Summer, %)", y = "Average Summer Temperature ( C°)",
    color = "Natural Disaster Occurence") + standard_theme
```



These graphs and predictors allow us to observe a lot:
The spread_summer_winter allows us to distinguish the distribution of "requested" cases
Few natural disaster cases are above +600% trend between winter and summer

Let's add the newly created predictor:

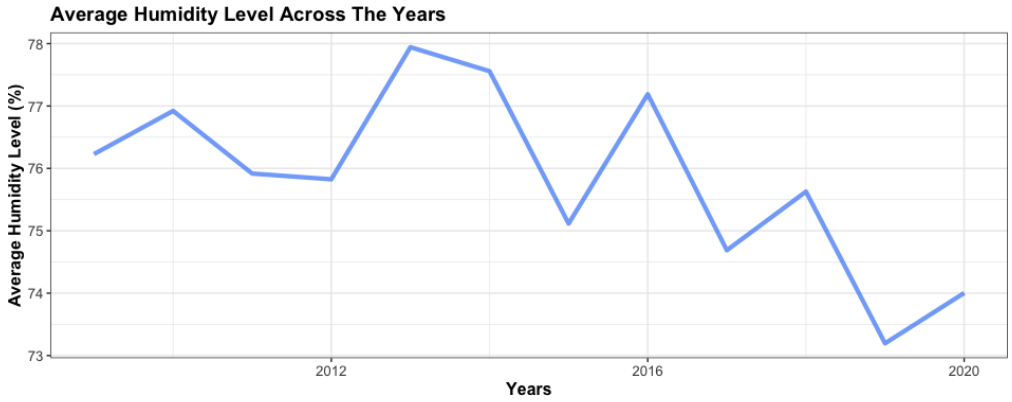
```
selected_predictors
```

```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp_winter"     "mean_humid"           "mean_ET"
 [7] "mean_radiation"       "n_high_risk_houses"   "natural_disaster"
[10] "trend_cumul_precipit"
```

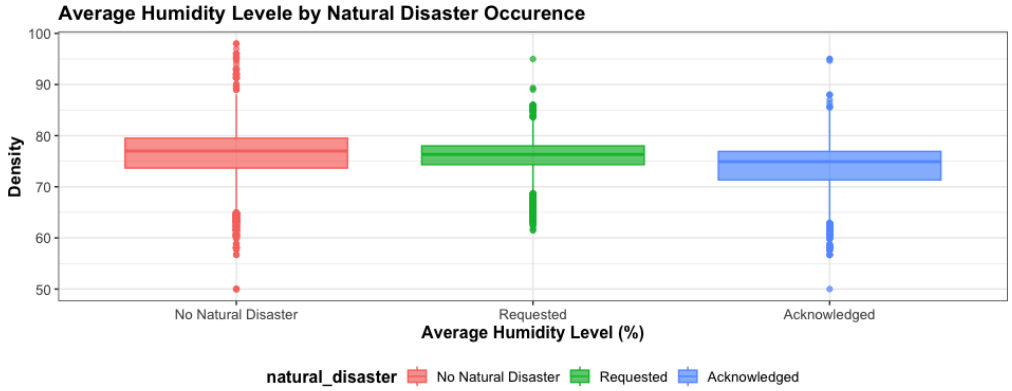
```
selected_predictors <- c(selected_predictors, "spread_summer_winter",
    "trend_winter_summer")

df2 <- df2 %>%
    mutate(spread_summer_winter = mean_temp_summer -
        mean_temp_winter, trend_winter_summer = (mean_temp_summer -
        mean_temp_winter)/mean_temp_winter)
```

4.8 Analyzing "mean_humid"

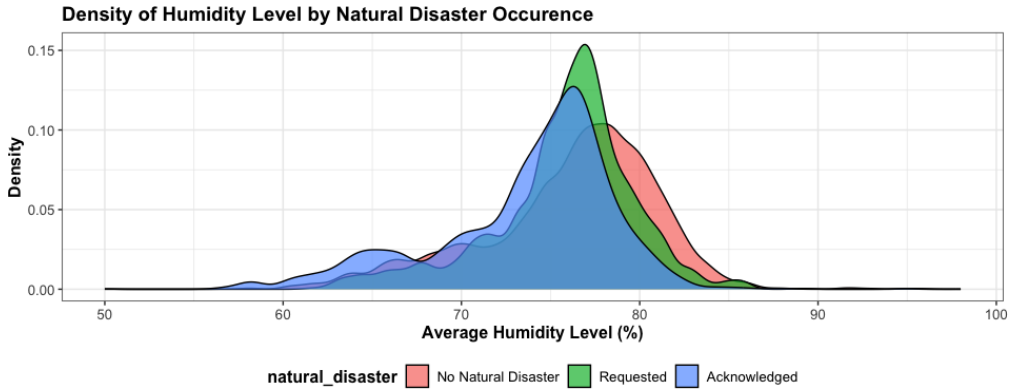
```
df2 %>%
    group_by(year) %>%
    summarise(yearly_mean_humid = mean(mean_humid,
        na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_mean_humid)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Average Humidity Level Across The Years",
    x = "Years", y = "Average Humidity Level (%)") +
    standard_theme
```




```
df2 %>%
    ggplot(aes(natural_disaster, mean_humid, fill = natural_disaster,
        color = natural_disaster)) + geom_boxplot(alpha = 0.7) +
    labs(title = "Average Humidity Levele by Natural Disaster Occurence",
        x = "Average Humidity Level (%)", y = "Density") +
    standard_theme
```

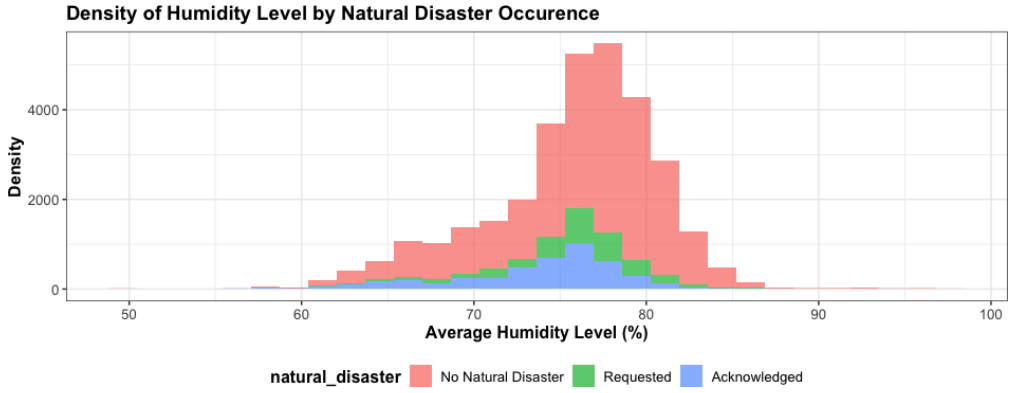



```
df2 %>%
    ggplot(aes(mean_humid, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Humidity Level by Natural
    ↪  Disaster Occurence",
    x = "Average Humidity Level (%)", y = "Density") +
    standard_theme
```

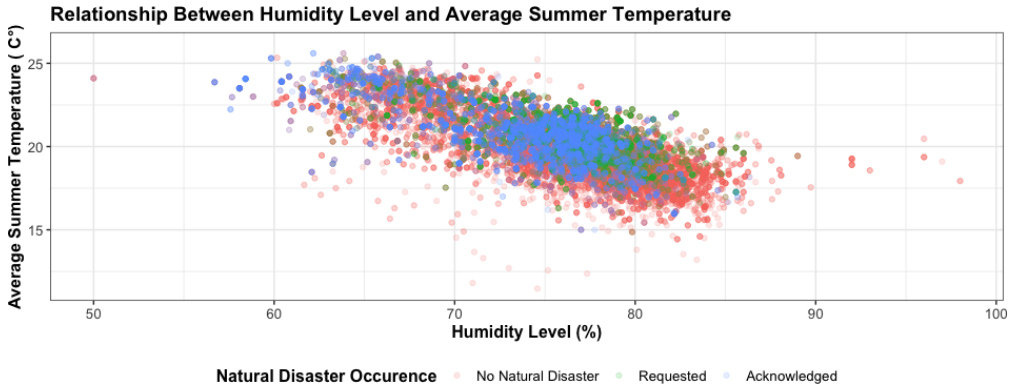



```
df2 %>%
    ggplot(aes(mean_humid, fill = natural_disaster)) +
```

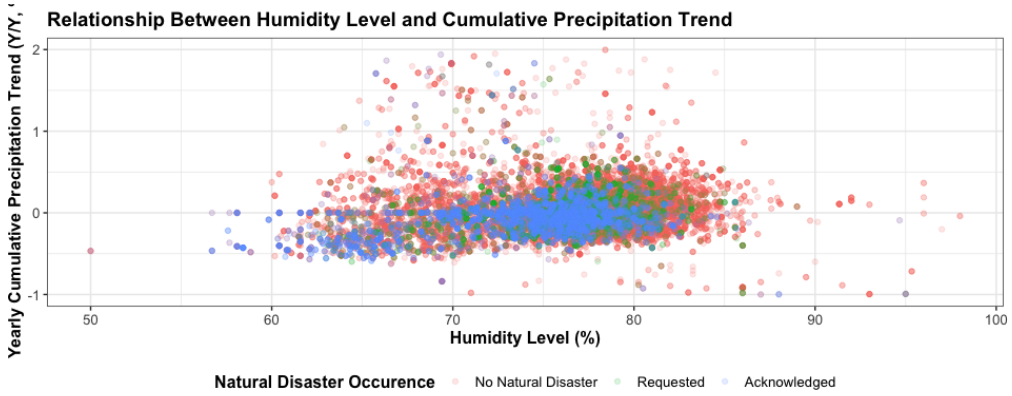
```
    geom_histogram(alpha = 0.7) + labs(title = "Density of Humidity Level by Natural
    ↪  Disaster Occurence",
    x = "Average Humidity Level (%)", y = "Density") +
    standard_theme
```



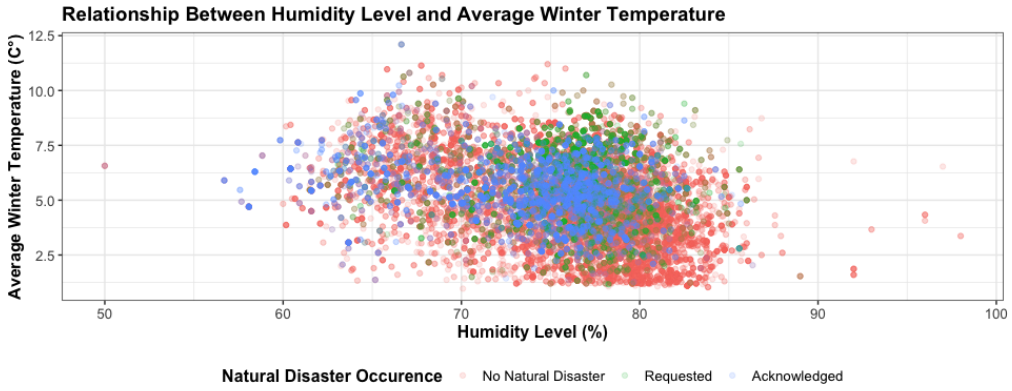

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_humid, mean_temp_summer, color = natural_disaster)) +
    geom_point(alpha = 0.15) + labs(title = "Relationship Between Humidity Level and
    ↪  Average Summer Temperature",
    x = "Humidity Level (%)", y = "Average Summer Temperature ( C°)",
    color = "Natural Disaster Occurence") + standard_theme
```



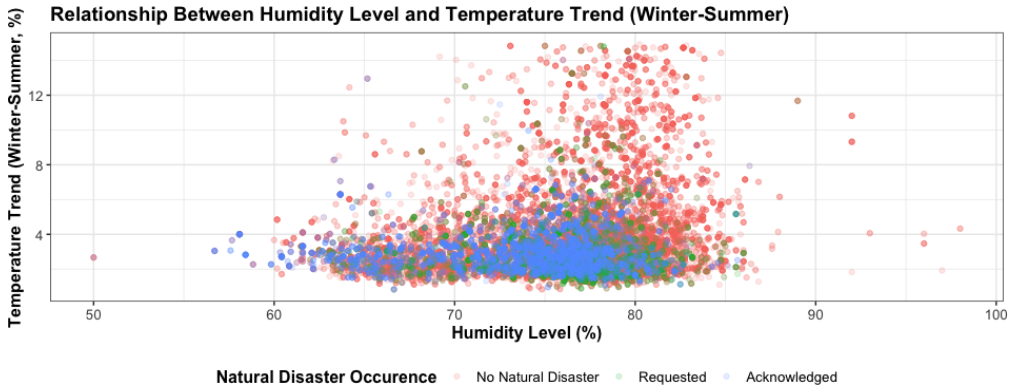

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_humid, trend_cumul_precipit, color = natural_disaster)) +
    geom_point(alpha = 0.15) + labs(title = "Relationship Between Humidity Level and
    ↪  Cumulative Precipitation Trend",
    x = "Humidity Level (%)", y = "Yearly Cumulative Precipitation Trend (Y/Y, %)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(mean_humid, mean_temp_winter, color = natural_disaster)) +
    geom_point(alpha = 0.15) + labs(title = "Relationship Between Humidity Level and
    ↪  Average Winter Temperature",
    x = "Humidity Level (%)", y = "Average Winter Temperature (C°)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(mean_humid, trend_winter_summer, color = natural_disaster)) +
    geom_point(alpha = 0.15) + labs(title = "Relationship Between Humidity Level and
    ↪  Temperature Trend (Winter-Summer)",
    x = "Humidity Level (%)", y = "Temperature Trend (Winter-Summer, %)",
    color = "Natural Disaster Occurence") + standard_theme
```



This predictor shows different distributions depending on the occurrence of natural disasters

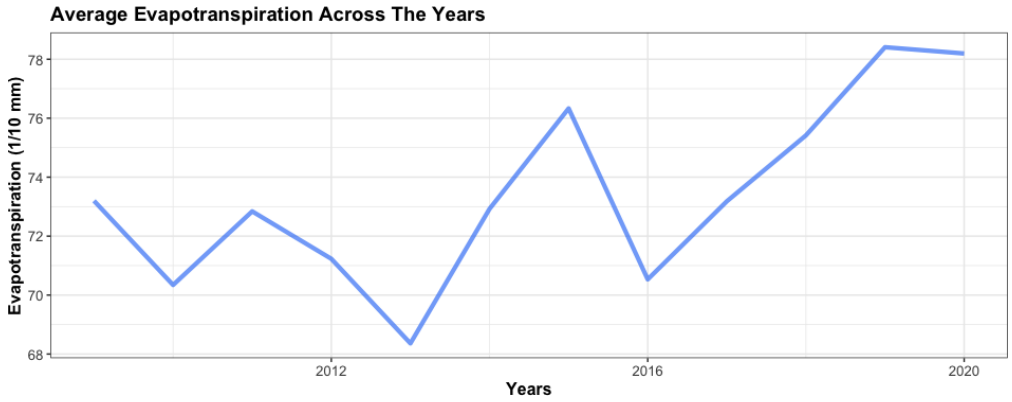
4.9 Analyzing "mean_ET"

As a reminder, we have a very low volume of data for the mean ET predictor (89% of NA)

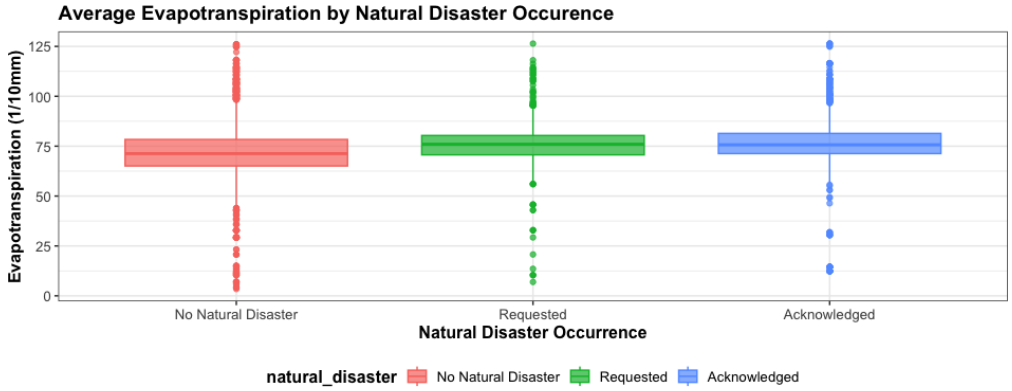
```
df2_nas %>%
    mutate(remaining_lines = round((nrow(df2) * (1 -
        (na_percent/100))))) %>%
    arrange(desc(na_percent)) %>%
    filter(na_percent > 0)
```

	na_percent	remaining_lines
mean_radiation	90	11732
mean_ET	89	12905
n_AEP	82	21118
n_SOU	82	21118
n_SUP	82	21118
n_tot	82	21118
mean_humid	73	31676
mean_min_humid	73	31676
mean_max_humid	73	31676
mean_vap_press	73	31676
mean_temp_spring	62	44582
mean_temp_summer	61	45755
mean_temp_autumn	61	45755
mean_temp	60	46928
mean_temp_winter	60	46928
cumul_precipit	3	113800

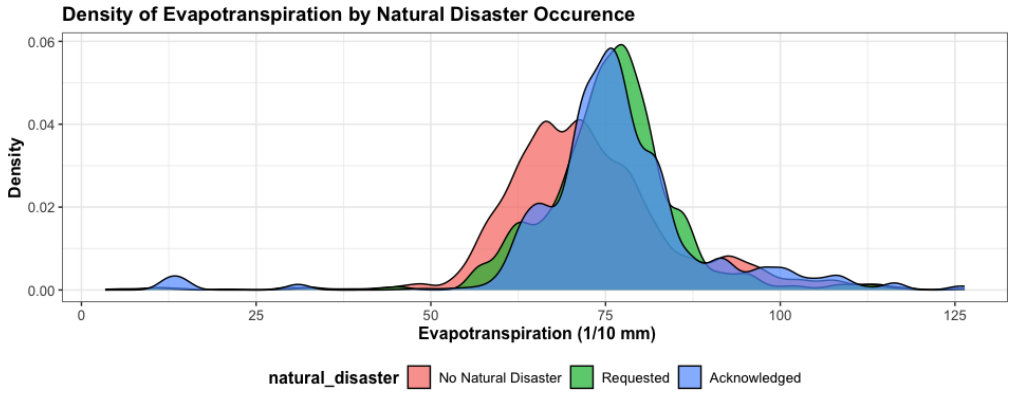
```
df2 %>%
    group_by(year) %>%
    summarise(yearly_mean_ET = mean(mean_ET, na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_mean_ET)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Average Evapotranspiration Across The
    ↪  Years",
    x = "Years", y = "Evapotranspiration (1/10 mm)") +
    standard_theme
```




```
df2 %>%
    ggplot(aes(natural_disaster, mean_ET, fill = natural_disaster,
        color = natural_disaster)) + geom_boxplot(alpha = 0.7) +
    labs(title = "Average Evapotranspiration by Natural Disaster Occurence",
        x = "Natural Disaster Occurrence", y = "Evapotranspiration (1/10mm)") +
    standard_theme
```

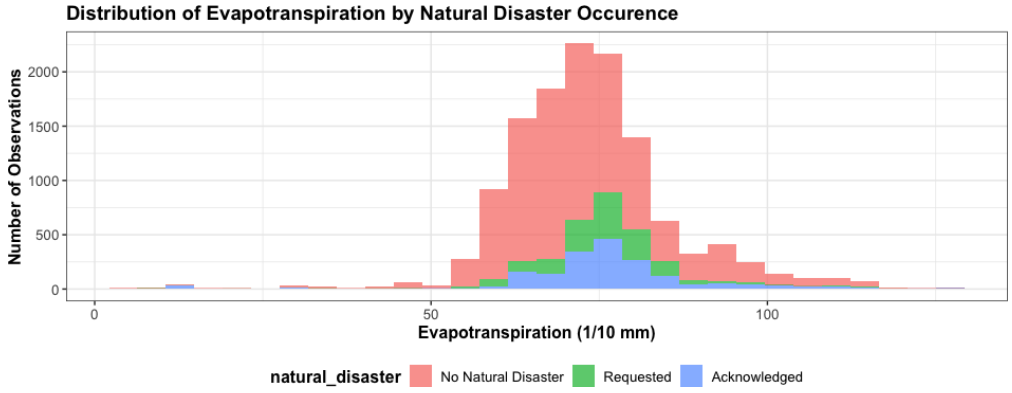



```
df2 %>%
    ggplot(aes(mean_ET, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Evapotranspiration by Natural
    ↪  Disaster Occurence",
    x = " Evapotranspiration (1/10 mm)", y = "Density") +
    standard_theme
```

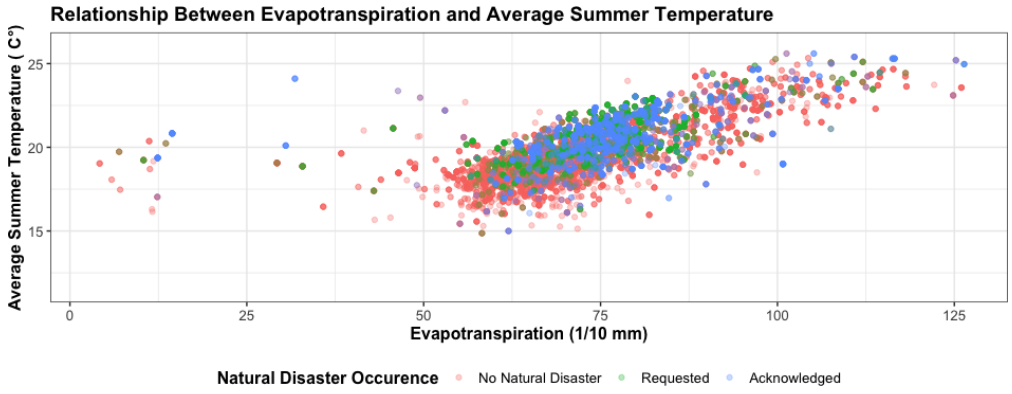



```
df2 %>%
    ggplot(aes(mean_ET, fill = natural_disaster)) +
```

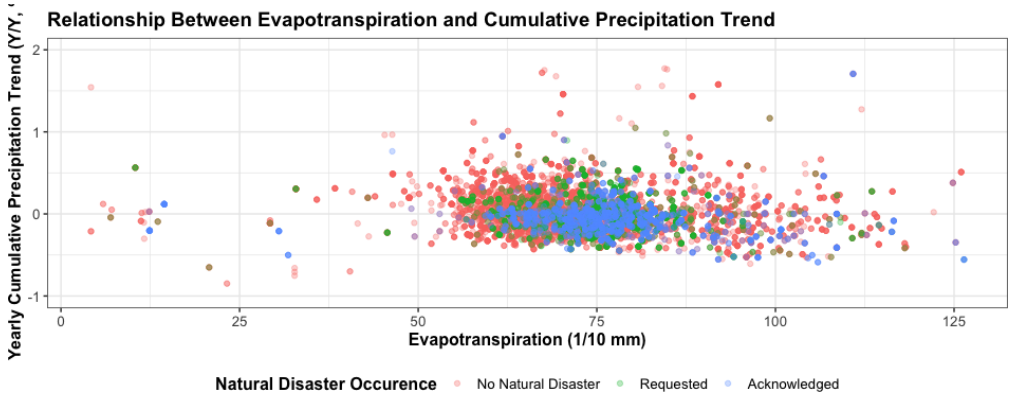
```
    geom_histogram(alpha = 0.7) + labs(title = "Distribution of Evapotranspiration by
    ↪  Natural Disaster Occurence",
    x = "Evapotranspiration (1/10 mm)", y = "Number of Observations") +
    standard_theme
```



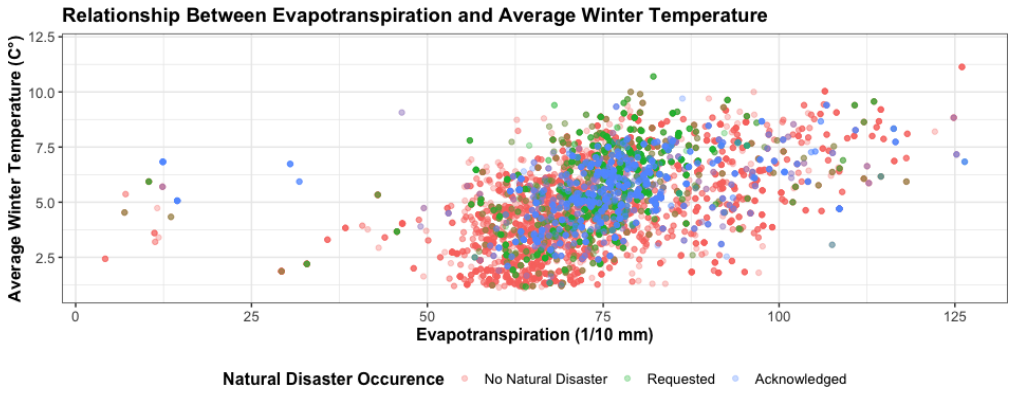

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_ET, mean_temp_summer, color = natural_disaster)) +
    geom_point(alpha = 0.3) + labs(title = "Relationship Between Evapotranspiration and
    ↪  Average Summer Temperature",
    x = "Evapotranspiration (1/10 mm)", y = "Average Summer Temperature ( C°)",
    color = "Natural Disaster Occurence") + standard_theme
```



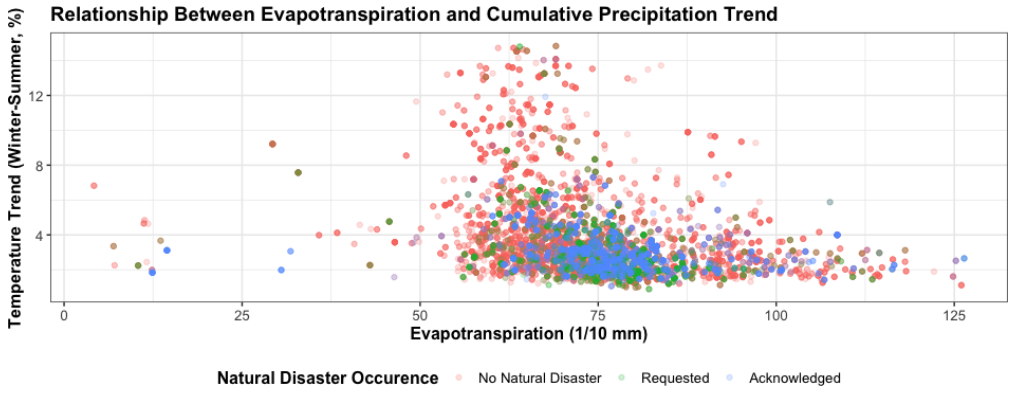

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_ET, trend_cumul_precipit, color = natural_disaster)) +
    geom_point(alpha = 0.3) + labs(title = "Relationship Between Evapotranspiration and
    ↪  Cumulative Precipitation Trend",
    x = "Evapotranspiration (1/10 mm)", y = "Yearly Cumulative Precipitation Trend (Y/Y,
    ↪  %)",
    color = "Natural Disaster Occurence") + standard_theme
```



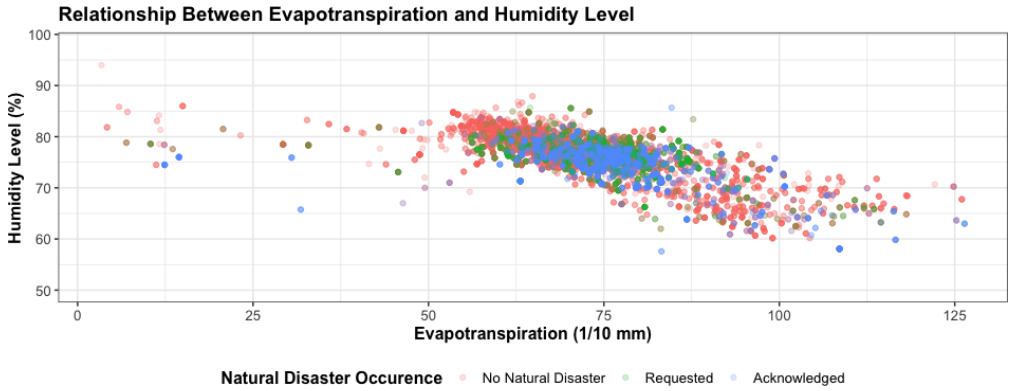

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(mean_ET, mean_temp_winter, color = natural_disaster)) +
    geom_point(alpha = 0.3) + labs(title = "Relationship Between Evapotranspiration and
    ↪  Average Winter Temperature",
    x = "Evapotranspiration (1/10 mm)", y = "Average Winter Temperature (C°)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(mean_ET, trend_winter_summer, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Evapotranspiration and
    ↪  Cumulative Precipitation Trend",
    x = "Evapotranspiration (1/10 mm)", y = "Temperature Trend (Winter-Summer, %)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_ET, mean_humid, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Evapotranspiration and
    ↪  Humidity Level",
    x = "Evapotranspiration (1/10 mm)", y = "Humidity Level (%)",
    color = "Natural Disaster Occurence") + standard_theme
```

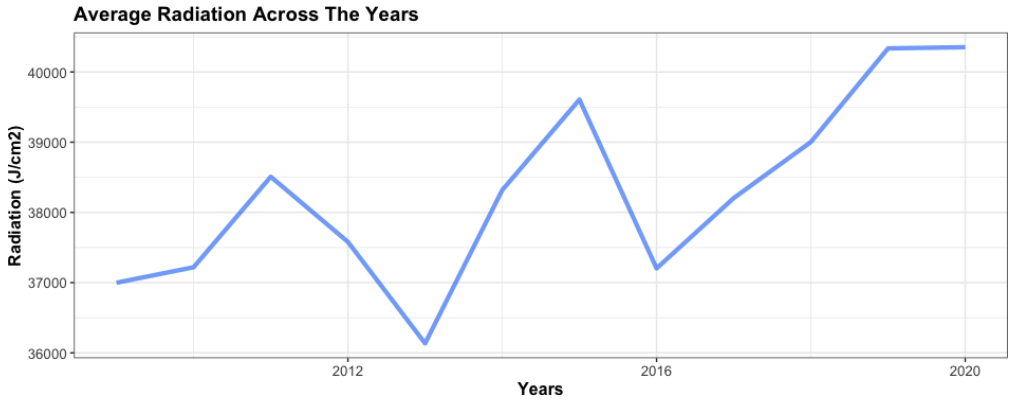


The distribution of this predictor allows us to distinguish cases without natural disasters from other cases.

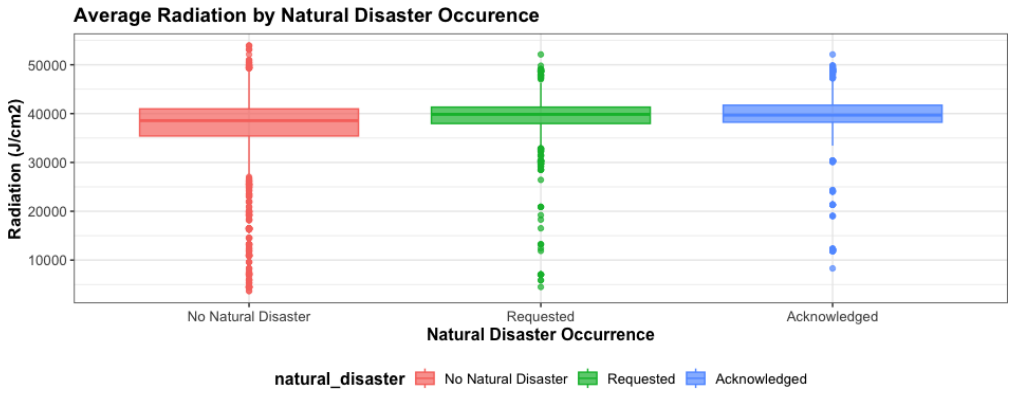
4.10 Analyzing "mean_radiation"

As a reminder, we have a very low volume of data for the mean ET predictor (90% of NA).

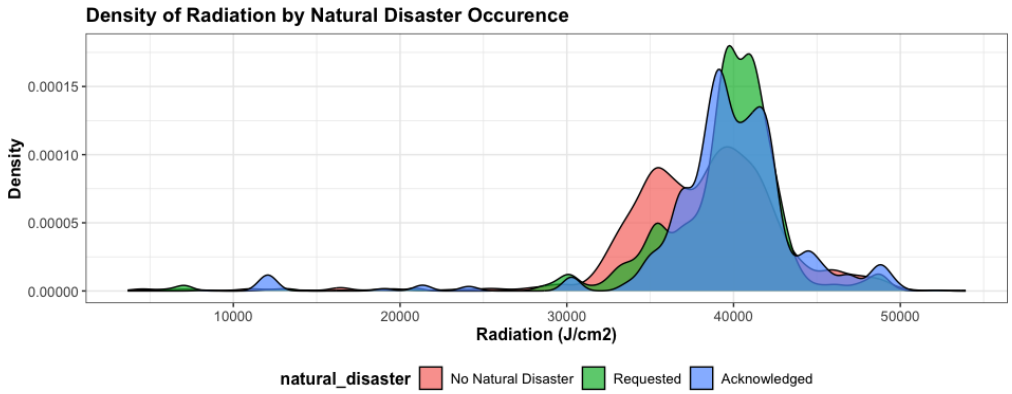
```
df2 %>%
    group_by(year) %>%
    summarise(yearly_mean_radiation = mean(mean_radiation,
        na.rm = TRUE)) %>%
    ggplot(aes(year, yearly_mean_radiation)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Average Radiation Across The Years",
    x = "Years", y = "Radiation (J/cm2)") + standard_theme
```




```
df2 %>%
    ggplot(aes(natural_disaster, mean_radiation, fill = natural_disaster,
        color = natural_disaster)) + geom_boxplot(alpha = 0.7) +
    labs(title = "Average Radiation by Natural Disaster Occurence",
        x = "Natural Disaster Occurrence", y = "Radiation (J/cm2)") +
    standard_theme
```

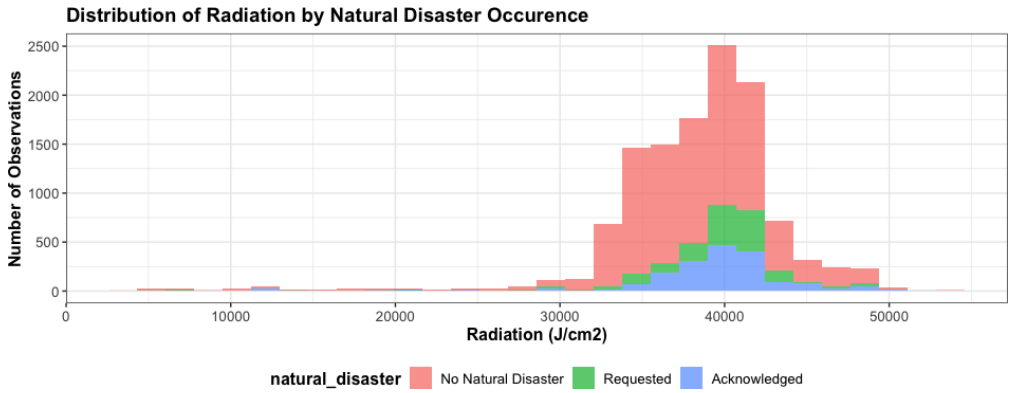



```
df2 %>%
    ggplot(aes(mean_radiation, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Density of Radiation by Natural Disaster
    ↪  Occurence",
    x = "Radiation (J/cm2)", y = "Density") + standard_theme
```

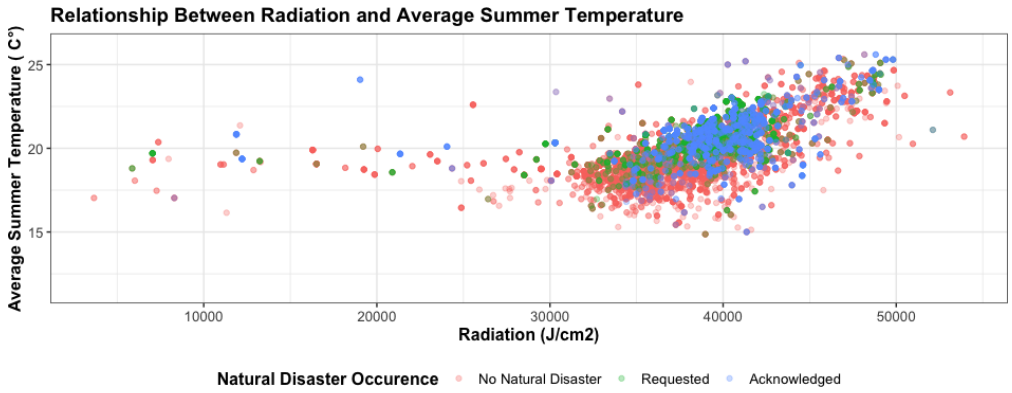



```
df2 %>%
    ggplot(aes(mean_radiation, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + labs(title = "Distribution of Radiation by Natural
    ↪  Disaster Occurence",
```

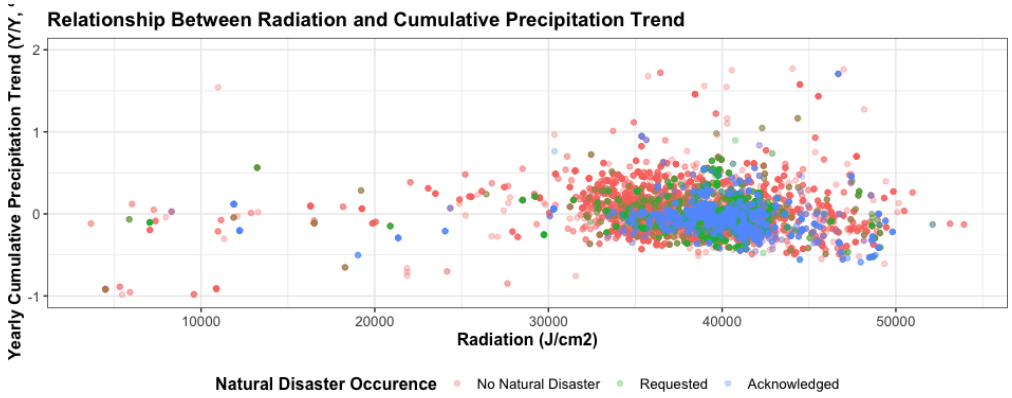
```
    x = "Radiation (J/cm2)", y = "Number of Observations") +
    standard_theme
```



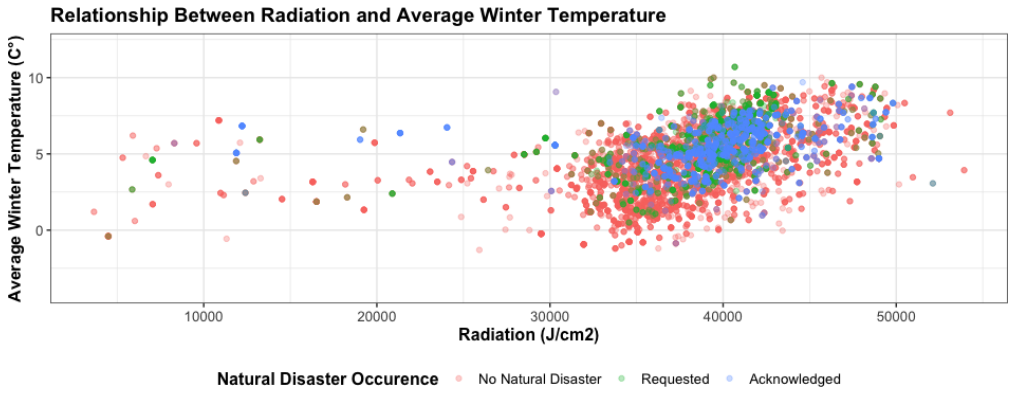

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_radiation, mean_temp_summer, color = natural_disaster)) +
    geom_point(alpha = 0.3) + labs(title = "Relationship Between Radiation and Average
    ↪  Summer Temperature",
    x = "Radiation (J/cm2)", y = "Average Summer Temperature ( C°)",
    color = "Natural Disaster Occurence") + standard_theme
```



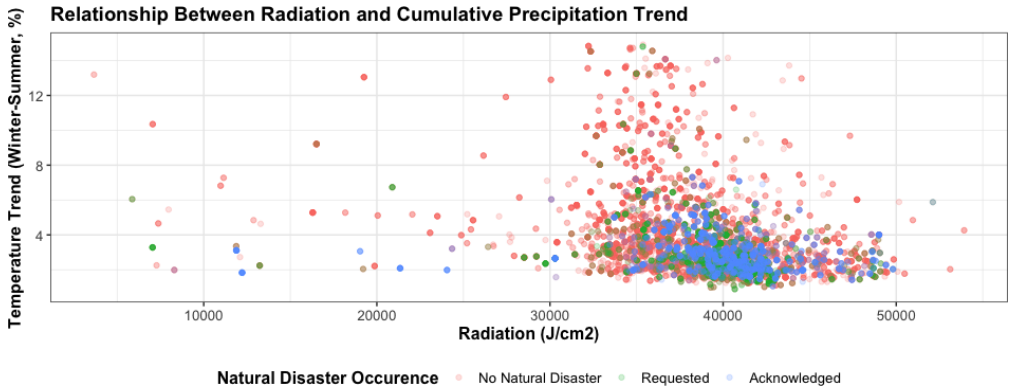

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(mean_radiation, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.3) +
    labs(title = "Relationship Between Radiation and Cumulative Precipitation Trend",
        x = "Radiation (J/cm2)", y = "Yearly Cumulative Precipitation Trend (Y/Y, %)",
        color = "Natural Disaster Occurence") + standard_theme
```



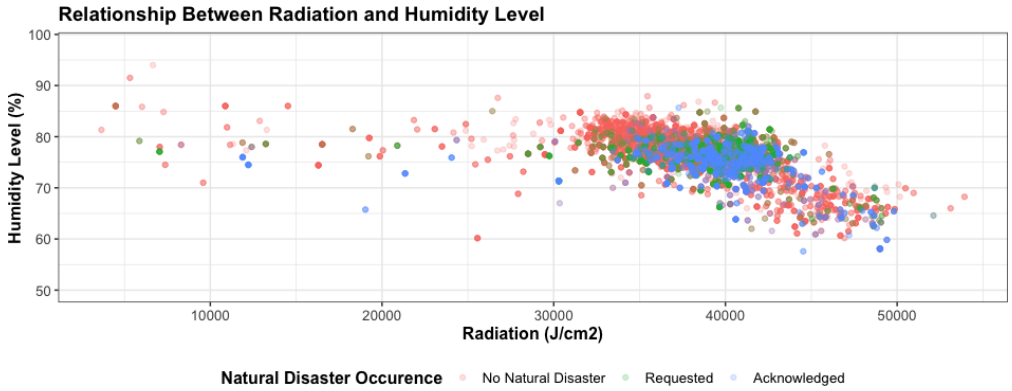

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_radiation, mean_temp_winter, color = natural_disaster)) +
    geom_point(alpha = 0.3) + labs(title = "Relationship Between Radiation and Average
    ↪  Winter Temperature",
    x = "Radiation (J/cm2)", y = "Average Winter Temperature (C°)",
    color = "Natural Disaster Occurence") + standard_theme
```



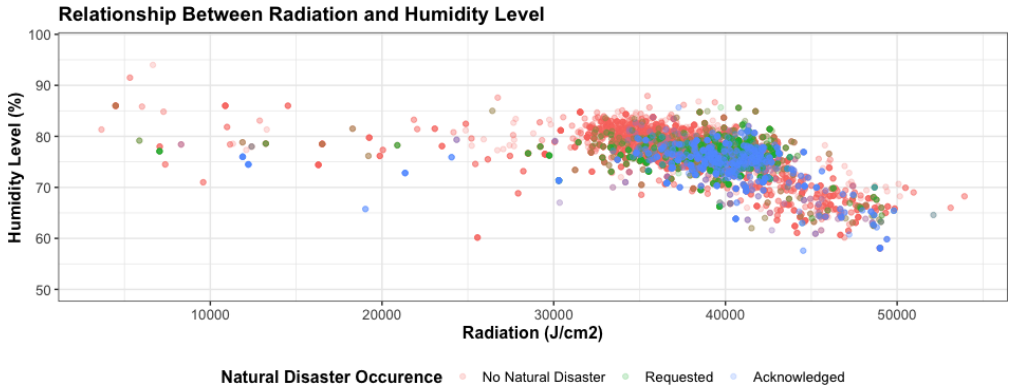

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(mean_radiation, trend_winter_summer,
        color = natural_disaster)) + geom_point(alpha = 0.2) +
    labs(title = "Relationship Between Radiation and Cumulative Precipitation Trend",
        x = "Radiation (J/cm2)", y = "Temperature Trend (Winter-Summer, %)",
        color = "Natural Disaster Occurence") + standard_theme
```



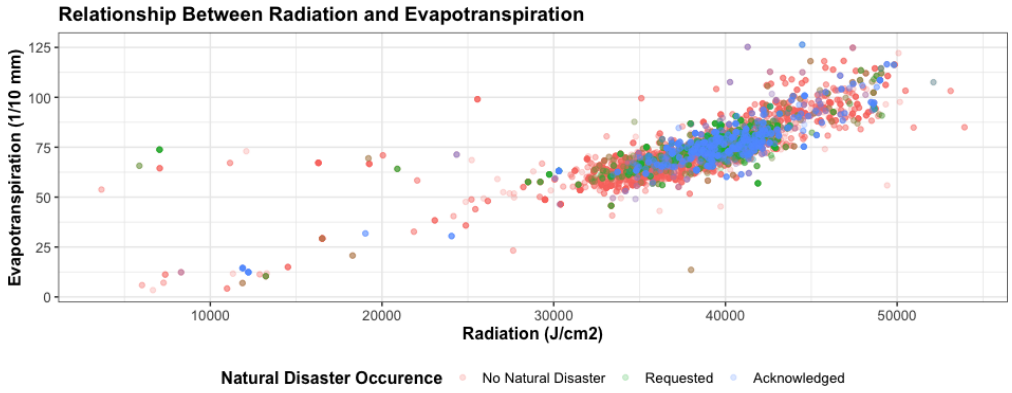

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_radiation, mean_humid, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Radiation and Humidity
    ↪  Level",
    x = "Radiation (J/cm2)", y = "Humidity Level (%)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_radiation, mean_humid, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Radiation and Humidity
    ↪  Level",
    x = "Radiation (J/cm2)", y = "Humidity Level (%)",
    color = "Natural Disaster Occurence") + standard_theme
```



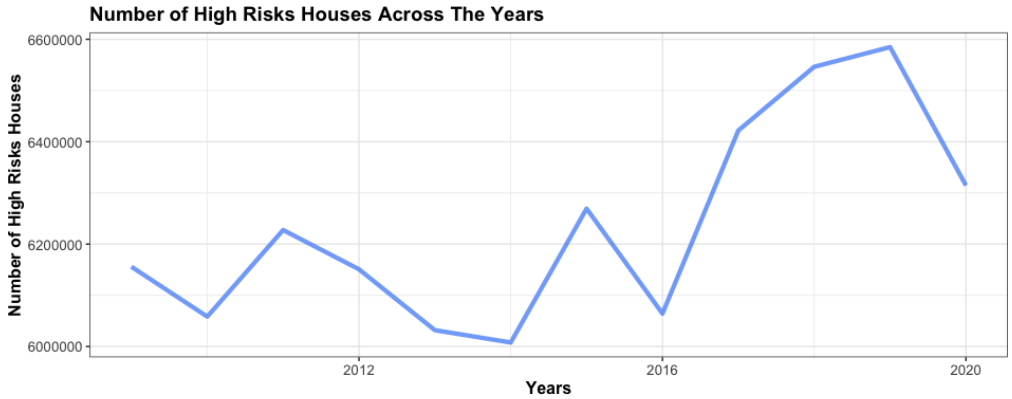
```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    ggplot(aes(mean_radiation, mean_ET, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Radiation and
    ↪  Evapotranspiration",
    x = "Radiation (J/cm2)", y = "Evapotranspiration (1/10 mm)",
    color = "Natural Disaster Occurence") + standard_theme
```



The distribution of this predictor allows us to distinguish cases without natural disasters from other cases.

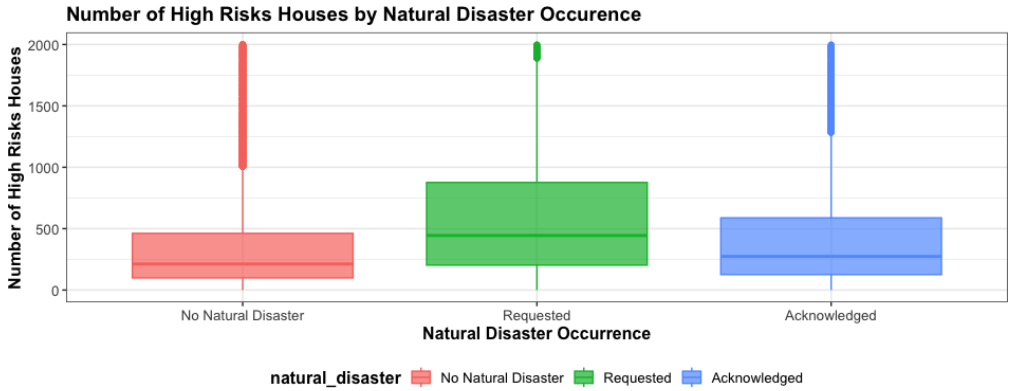
4.11 Analyzing "n_high_risk_houses"

```
df2 %>%
    group_by(year) %>%
    summarise(sum_n_high_risk_houses = sum(n_high_risk_houses,
        na.rm = TRUE)) %>%
    ggplot(aes(year, sum_n_high_risk_houses)) + geom_line(color = "#4285f6",
    alpha = 0.7, size = 1.5) + labs(title = "Number of High Risks Houses Across The
    ↪  Years",
    x = "Years", y = "Number of High Risks Houses") +
    standard_theme
```

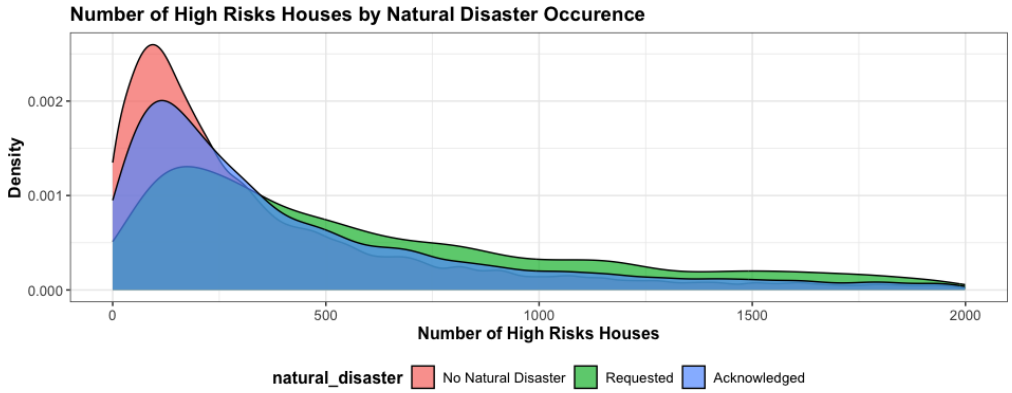



```
df2 %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(natural_disaster, n_high_risk_houses,
```

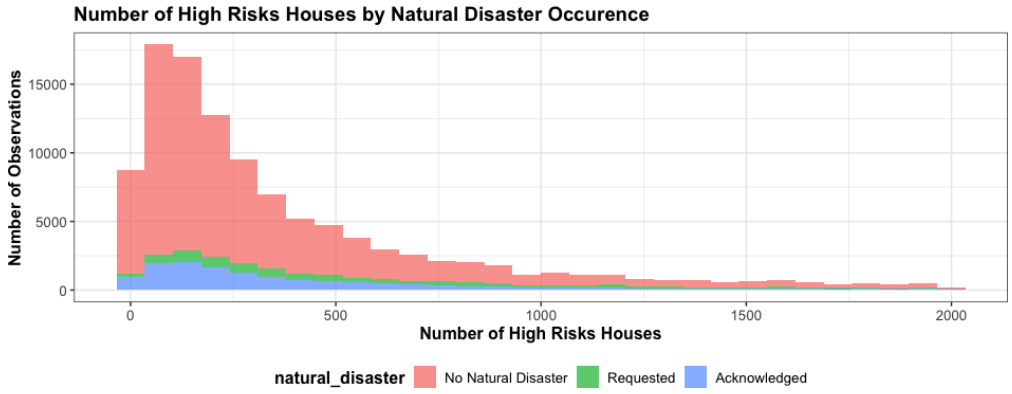
```
        fill = natural_disaster, color = natural_disaster)) +

geom_boxplot(alpha = 0.7) + labs(title = "Number of High Risks Houses by Natural Disaster
↪  Occurence",
    x = "Natural Disaster Occurrence", y = "Number of High Risks Houses") +
    standard_theme
```



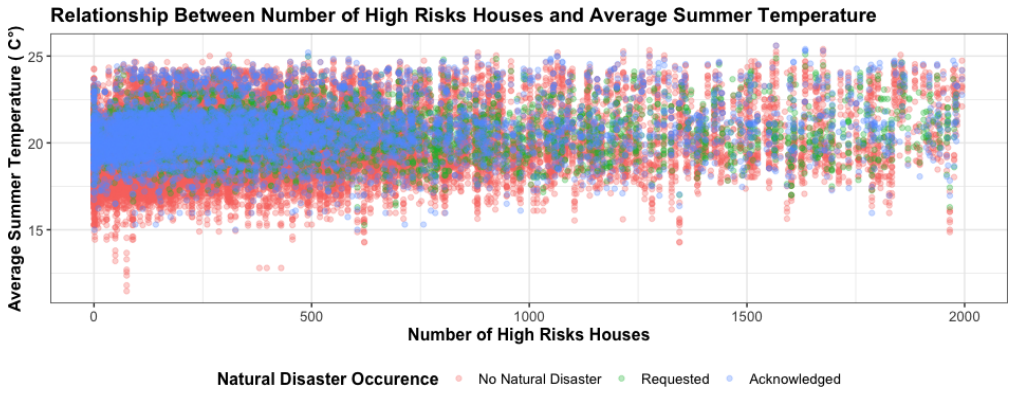

```
df2 %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, fill = natural_disaster)) +
    geom_density(alpha = 0.7) + labs(title = "Number of High Risks Houses by Natural
    ↪  Disaster Occurence",
    x = "Number of High Risks Houses", y = "Density") +
    standard_theme
```



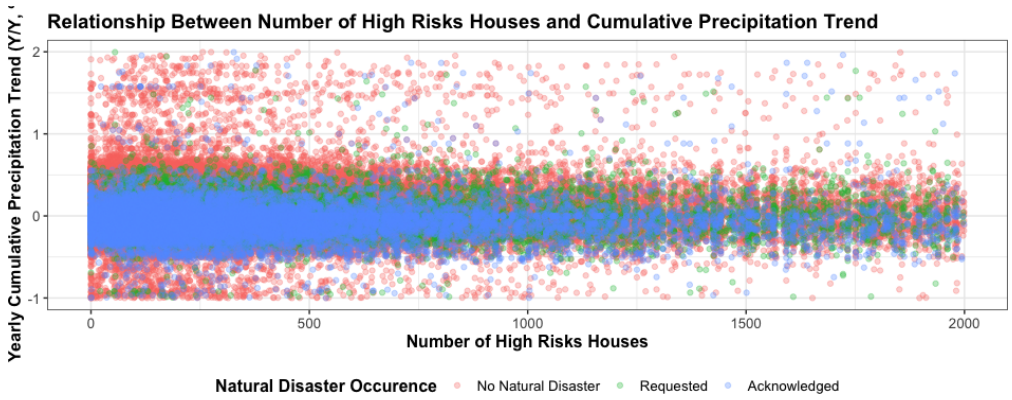

```
df2 %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, fill = natural_disaster)) +
    geom_histogram(alpha = 0.7) + labs(title = "Number of High Risks Houses by Natural
    ↪  Disaster Occurence",
    x = "Number of High Risks Houses", y = "Number of Observations") +
    standard_theme
```



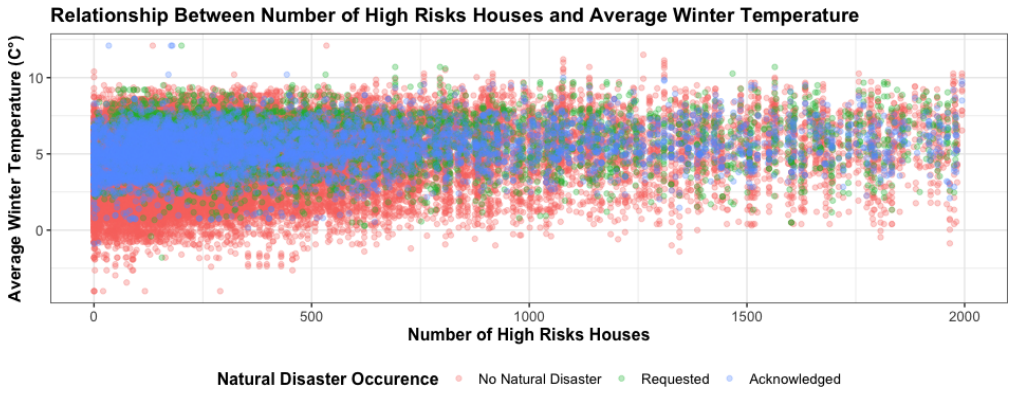

```r
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, mean_temp_summer,
        color = natural_disaster)) + geom_point(alpha = 0.3) +
    labs(title = "Relationship Between Number of High Risks Houses and Average Summer
    ↪  Temperature",
        x = "Number of High Risks Houses", y = "Average Summer Temperature ( C°)",
        color = "Natural Disaster Occurence") + standard_theme
```



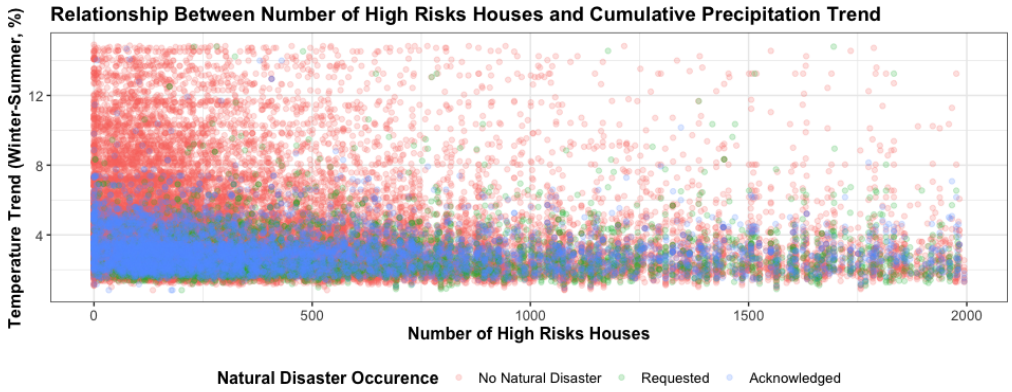

```r
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    filter(trend_cumul_precipit < 2) %>%
    ggplot(aes(n_high_risk_houses, trend_cumul_precipit,
        color = natural_disaster)) + geom_point(alpha = 0.3) +
    labs(title = "Relationship Between Number of High Risks Houses and Cumulative
    ↪  Precipitation Trend",
        x = "Number of High Risks Houses", y = "Yearly Cumulative Precipitation Trend
        ↪  (Y/Y, %)",
        color = "Natural Disaster Occurence") + standard_theme
```



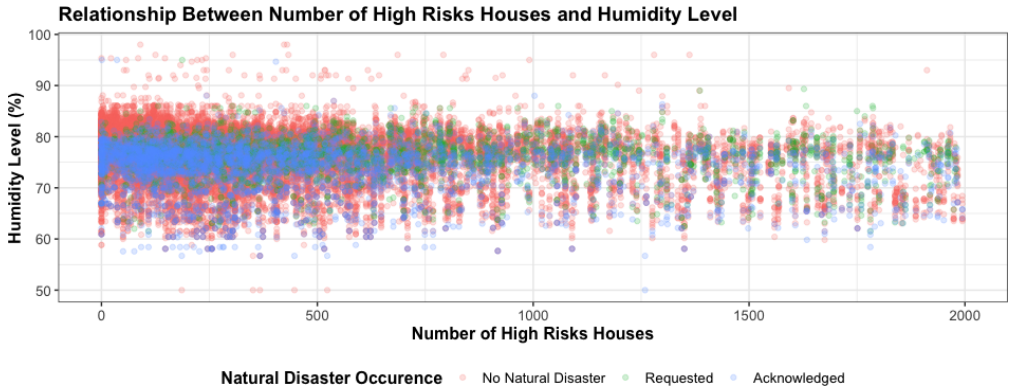

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, mean_temp_winter,
        color = natural_disaster)) + geom_point(alpha = 0.3) +
    labs(title = "Relationship Between Number of High Risks Houses and Average Winter
    ↪  Temperature",
        x = "Number of High Risks Houses", y = "Average Winter Temperature (C°)",
        color = "Natural Disaster Occurence") + standard_theme
```



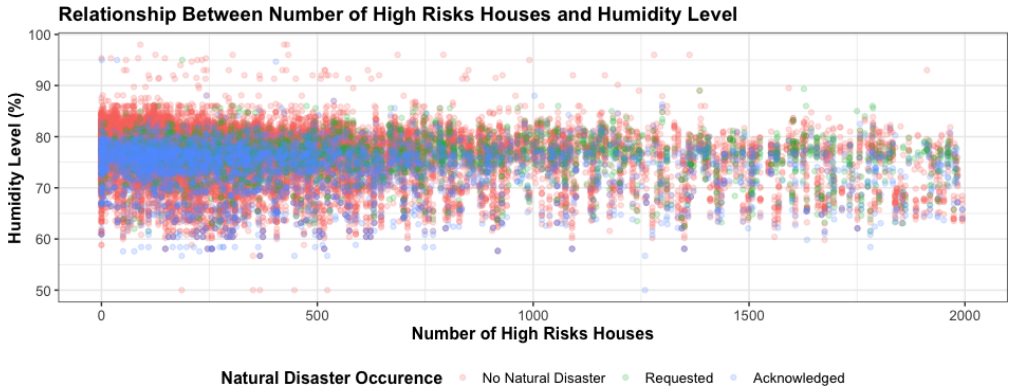

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    filter(trend_winter_summer < 15 & trend_winter_summer >
        -4) %>%
    ggplot(aes(n_high_risk_houses, trend_winter_summer,
        color = natural_disaster)) + geom_point(alpha = 0.2) +
    labs(title = "Relationship Between Number of High Risks Houses and Cumulative
    ↪  Precipitation Trend",
        x = "Number of High Risks Houses", y = "Temperature Trend (Winter-Summer, %)",
        color = "Natural Disaster Occurence") + standard_theme
```



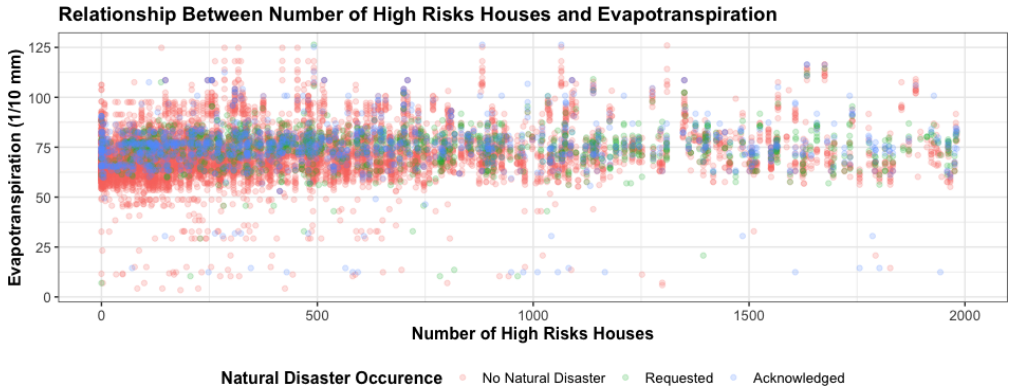

```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, mean_humid, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Number of High Risks
    ↪  Houses and Humidity Level",
    x = "Number of High Risks Houses", y = "Humidity Level (%)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, mean_humid, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Number of High Risks
    ↪  Houses and Humidity Level",
    x = "Number of High Risks Houses", y = "Humidity Level (%)",
    color = "Natural Disaster Occurence") + standard_theme
```




```
df2 %>%
    arrange(-desc(natural_disaster)) %>%
    filter(n_high_risk_houses < 2000) %>%
    ggplot(aes(n_high_risk_houses, mean_ET, color = natural_disaster)) +
    geom_point(alpha = 0.2) + labs(title = "Relationship Between Number of High Risks
    ↪  Houses and Evapotranspiration",
    x = "Number of High Risks Houses", y = "Evapotranspiration (1/10 mm)",
    color = "Natural Disaster Occurence") + standard_theme
```



This predictor doesn't give us much information, we'll remove it from the selected predictors.

```
selected_predictors <- selected_predictors[-which(selected_predictors ==
    "n_high_risk_houses")]
selected_predictors
```

```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp_winter"     "mean_humid"           "mean_ET"
 [7] "mean_radiation"       "natural_disaster"     "trend_cumul_precipit"
[10] "spread_summer_winter" "trend_winter_summer"
```

4.12 Analyzing "natural_disaster"

```
df2 %>%
    group_by(natural_disaster) %>%
    summarise(n = n(), percent = n/nrow(df2) * 100)
```

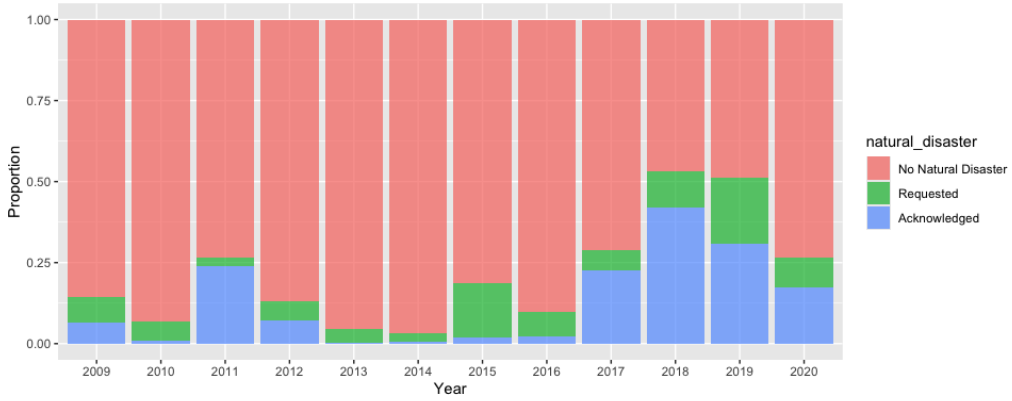
```
# A tibble: 3 x 3
```

```
  natural_disaster        n percent
  <fct>               <int>   <dbl>
1 No Natural Disaster 91519   78.0
2 Requested           10020    8.54
3 Acknowledged        15781   13.5
```

We see that most of the time (in our dataframe) there are not natural disasters (78%).

Let's check the natural disaster ratio over the years.

```
df2 %>%
    ggplot(aes(as.factor(year), fill = natural_disaster)) +
    geom_bar(position = "fill", alpha = 0.7) + labs(x = "Year",
    y = "Proportion")
```



We see that a large minority off cities suffer of natural disaster.
During the worst years the ratio is around 12 and 40%.
The ratio has been in average much higher since 2017.

Nous avions gardé natural disaster dans selected_predictors pour notre matrice de correlation, mais nous pouvons maintenant le retirer.

```
selected_predictors <- selected_predictors[-which(selected_predictors ==
    "natural_disaster")]
selected_predictors
```

```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp_winter"     "mean_humid"           "mean_ET"
 [7] "mean_radiation"       "trend_cumul_precipit" "spread_summer_winter"
[10] "trend_winter_summer"
```

5 MODEL BUIDLING

5.1 Multiclass prediction models

5.1.1 Dataset

As a reminder, we performed a preselection of the following predictors

```
selected_predictors
```

```
 [1] "cumul_precipit"       "mean_temp_summer"     "nat_dis_group"
 [4] "mean_temp_winter"     "mean_humid"           "mean_ET"
 [7] "mean_radiation"       "trend_cumul_precipit" "spread_summer_winter"
[10] "trend_winter_summer"
```

```
str(df2[, selected_predictors])
```

```
tibble [117,320 x 10] (S3: tbl_df/tbl/data.frame)
 $ cumul_precipit      : num [1:117320] 804 750 752 712 713 ...
 $ mean_temp_summer    : num [1:117320] NaN NaN 20.4 20.1 19.5 ...
 $ nat_dis_group       : num [1:117320] 2 2 3 3 3 1 1 1 1 1 ...
 $ mean_temp_winter    : num [1:117320] NaN NaN 4.87 4.73 6.43 ...
 $ mean_humid          : num [1:117320] NaN NaN NaN NaN NaN ...
 $ mean_ET             : num [1:117320] NaN NaN NaN NaN NaN ...
 $ mean_radiation      : num [1:117320] NaN NaN NaN NaN NaN ...
 $ trend_cumul_precipit: num [1:117320] -0.00802 -0.06667 -0.00397 -0.05334 0.00126 ...
 $ spread_summer_winter: num [1:117320] NaN NaN 15.5 15.3 13 ...
 $ trend_winter_summer : num [1:117320] NaN NaN 3.18 3.24 2.03 ...
```

We have a large number of rows: 117,320.

We have a lot of missing data.

```
df2_nas <- round(colSums(is.na(df2[, selected_predictors]))/nrow(df2) *
    100)
df2_nas <- as.data.frame(df2_nas)
colnames(df2_nas) <- c("na_percent")
df2_nas %>%
    mutate(remaining_lines = round((nrow(df2) * (1 -
        (na_percent/100))))) %>%
    arrange(desc(na_percent))
```

```
                     na_percent remaining_lines
mean_radiation               90           11732
mean_ET                      89           12905
mean_humid                   73           31676
mean_temp_summer             61           45755
spread_summer_winter         61           45755
trend_winter_summer          61           45755
mean_temp_winter             60           46928
```

```
trend_cumul_precipit          5          111454
cumul_precipit                3          113800
nat_dis_group                 0          117320
```

If we filter out all rows with NAs in our predictors, we are left with only a few rows: 11,622.

```
df2[, selected_predictors] %>%
    drop_na() %>%
    nrow()
```

```
[1] 11739
```

From the data analysis, we observed that there are no simple interactions between the predictors.

5.1.2 Value to predict

This is a 3-level factor:
- 1 - "No Natural Disaster"
- 2 - "Requested"
- 3 - "Acknowledged"

The distribution is highly imbalanced: 1 = 75.6%, 2 = 10.2%, 3 = 14.2%.
Levels 2 and 3 are significantly underrepresented, so we will need to pay close attention to this when building our models.

5.1.3 Models Selection

Given the nature of our classification problem:
- large number of observations,
- highly imbalanced dataset,
- with complex non-linear interactions between predictors.

We selected decision tree-based models:
- 1. Random Forest (`randomForest`) – a robust and fast option for initial experiments.
- 2. Ranger (`ranger`) – a faster implementation of Random Forest (thanks to multithreading), allowing for quicker testing across configurations.
- 3. XGBoost – chosen for its potential to improve predictive accuracy.

5.1.4 Metrics & Target

In our context, overall accuracy alone is not truly meaningful, as the target variable is highly imbalanced. In other words, if we simply predicted only the "no natural disaster" cases (class "0"), we would still achieve a global accuracy of 75.6%.

Our primary objective is to capture as many natural disaster cases as possible, even at the cost of misclassifying non-disaster cases.

So, in addition to overall accuracy, we will focus on the following metrics:
- Sensitivity (Recall) – indicates the percentage of true positive cases detected, per class (to maximize for class 2 and 3)
- F1-Score – combines precision and recall, helping to balance the evaluation of imbalanced classes.
- Brier Score – measures the quality of the predicted probabilities (the more confident and accurate the model is, the lower the Brier score).

Let's define the list of metrics and our corresponding objectives:

```
results <- data.frame(row.names = "Target", Accrcy = "> 0.84",
    Stivity_2 = "> 0.59", Stivity_3 = "> 0.79", F1_2 = "> 0.49",
    F1_3 = "> 0.49", Brier = "< 0.31")

results
```

```
       Accrcy Stivity_2 Stivity_3   F1_2   F1_3  Brier
Target > 0.84    > 0.59    > 0.79 > 0.49 > 0.49 < 0.31
```

5.1.5 Creating Train & Test Set

```
set.seed(1)
test_index <- createDataPartition(df2$natural_disaster,
    times = 1, p = 0.2, list = FALSE)
test_set <- df2[test_index, ]
train_set <- df2[-test_index, ]
missing_cities <- test_set %>%
    filter(!insee_code %in% train_set$insee_code)
train_set <- train_set %>%
    union(missing_cities)
test_set <- test_set %>%
    filter(insee_code %in% train_set$insee_code)
```

As "nat_dis_group" predictor is created from the all dataset, we need to recreate it only from the train set.

```
train_nat_dis_group <- train_set %>%
    group_by(insee_code) %>%
    summarise(nat_dis_group = sum(nat_dis)) %>%
    select(insee_code, nat_dis_group)

train_set <- train_set %>%
    select(-nat_dis_group) %>%
    left_join(nat_dis_group, by = "insee_code")

test_set <- test_set %>%
    select(-nat_dis_group) %>%
    left_join(train_nat_dis_group, by = "insee_code")
```

5.1.6 Naive Model

To establish a baseline, let's start by creating a "naive" model that randomly generates the target classes (1, 2, 3) while preserving their overall proportions in the dataset.

```
percentage <- train_set %>%
    group_by(natural_disaster) %>%
    summarise(count = n()) %>%
    mutate(percent = count/sum(count))

percentage[which(percentage$natural_disaster == "Requested"),
    ]$percent
```

```
[1] 0.08540834
```

```
set.seed(1)
naive_model <- sample(c("No Natural Disaster", "Requested",
    "Acknowledged"), size = nrow(test_set), replace = TRUE,
    prob = c(percentage[which(percentage$natural_disaster ==
        "No Natural Disaster"), ]$percent, percentage[which(percentage$natural_disaster
        ↪  ==
        "Requested"), ]$percent, percentage[which(percentage$natural_disaster ==
        "Acknowledged"), ]$percent))
naive_model <- factor(naive_model, levels = c("No Natural Disaster",
    "Requested", "Acknowledged"))
y_test <- factor(test_set$natural_disaster, levels = c("No Natural Disaster",
    "Requested", "Acknowledged"))

accuracy <- sum(y_test == naive_model)/length(y_test)
accuracy
```

```
[1] 0.6354144
```

```
results_naive_model <- confusionMatrix(naive_model,
    y_test)

new_result <- data.frame(row.names = "naive_model",
    Accrcy = round(accuracy, 2), Stivity_2 = round(results_naive_model$byClass["Class:
    ↪  Requested",
        "Sensitivity"], 2), Stivity_3 = round(results_naive_model$byClass["Class:
        ↪  Acknowledged",
        "Sensitivity"], 2), F1_2 = round(results_naive_model$byClass["Class: Requested",
        "F1"], 2), F1_3 = round(results_naive_model$byClass["Class: Acknowledged",
        "F1"], 2), Brier = "NA")
new_result <- new_result %>%
    mutate(across(everything(), as.character))

results <- bind_rows(results, new_result)
results
```

```
            Accrcy Stivity_2 Stivity_3   F1_2   F1_3  Brier
Target      > 0.84    > 0.59    > 0.79 > 0.49 > 0.49 < 0.31
naive_model   0.64      0.09      0.14   0.09   0.14     NA
```

5.1.7 M1 - Random Forest - (all selected_predictors)

Let's start by testing Random Forest on the preselected predictors:

We are creating a function to make it easier to repeat the Random Forest model training.

```
rf_model <- function(studied_df, model_name, focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    test_index <- createDataPartition(studied_df$natural_disaster,
        times = 1, p = 0.2, list = FALSE)
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    # As 'nat_dis_group' predictor is created
    # from the all dataset, we need to recreate
    # it only from the train set :
    train_nat_dis_group <- train_set %>%
        group_by(insee_code) %>%
        summarise(nat_dis_group = sum(nat_dis)) %>%
        select(insee_code, nat_dis_group)

    train_set <- train_set %>%
        select(-nat_dis_group) %>%
        left_join(nat_dis_group, by = "insee_code")

    test_set <- test_set %>%
        select(-nat_dis_group) %>%
        left_join(train_nat_dis_group, by = "insee_code")

    model_train_set <- train_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    model_test_set <- test_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    y_train <- model_train_set %>%
        pull(natural_disaster)
    y_train <- as.factor(y_train)

    y_test <- model_test_set %>%
        pull(natural_disaster)
    y_test <- as.factor(y_test)
```

```
    x_train <- model_train_set %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)

    # Random Forest Model
    random_forest_fit_1 <- randomForest(natural_disaster ~
        ., data = model_train_set, importance = TRUE,
        keep.forest = TRUE)
    # Generating predictions & metrics
    y_predicted <- predict(random_forest_fit_1, x_test)

    model_results <- confusionMatrix(y_predicted, y_test)
    accuracy <- sum(y_predicted == y_test)/length(y_test)

    y_predicted_prob <- predict(random_forest_fit_1,
        x_test, type = "prob")
    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 2)

    # Output of the metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 2), Stivity_2 = round(model_results$byClass["Class:
        ↪  Requested",
            "Sensitivity"], 2), Stivity_3 = round(model_results$byClass["Class:
            ↪  Acknowledged",
            "Sensitivity"], 2), F1_2 = round(model_results$byClass["Class: Requested",
            "F1"], 2), F1_3 = round(model_results$byClass["Class: Acknowledged",
            "F1"], 2), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))

    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}
```

Running our function

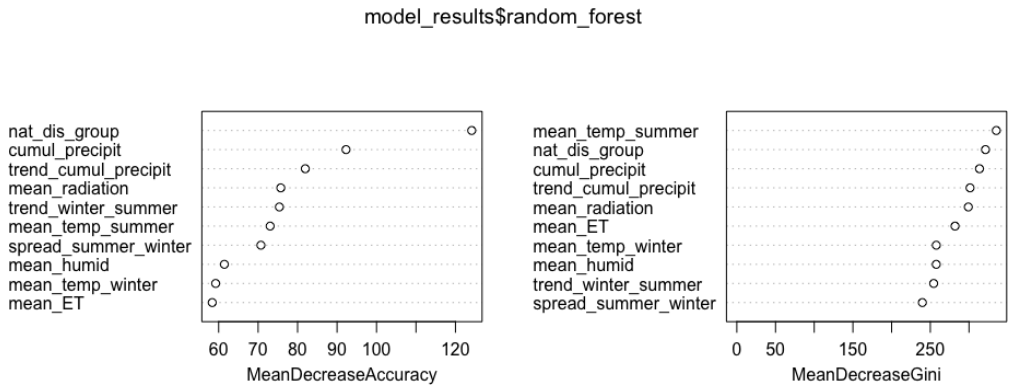
```
model_results <- rf_model(studied_df = df2, model_name = "M1_RF_(All_slctd_prms)",
    focus_predictors = selected_predictors)

results <- bind_rows(results, model_results$new_result)  # Adding the new result to our
↪  historical results
results
```

	Accrcy	Stivity_2	Stivity_3	F1_2	F1_3	Brier
Target	> 0.84	> 0.59	> 0.79	> 0.49	> 0.49	< 0.31
naive_model	0.64	0.09	0.14	0.09	0.14	NA
M1_RF_(All_slctd_prms)	0.74	0.33	0.3	0.4	0.38	0.38

```r
model_results$confusion_matrix  # Showing all the metrics
```

```
Confusion Matrix and Statistics

                     Reference
Prediction            No Natural Disaster Requested Acknowledged
  No Natural Disaster                1542       200          224
  Requested                            89        99            8
  Acknowledged                         79         5           98

Overall Statistics

               Accuracy : 0.7419
                 95% CI : (0.7237, 0.7595)
    No Information Rate : 0.7295
    P-Value [Acc > NIR] : 0.09209

                  Kappa : 0.2955

 Mcnemar's Test P-Value : < 2e-16

Statistics by Class:

                     Class: No Natural Disaster Class: Requested
Sensitivity                              0.9018          0.32566
Specificity                              0.3312          0.95245
Pos Pred Value                           0.7843          0.50510
Neg Pred Value                           0.5556          0.90456
Prevalence                               0.7295          0.12969
Detection Rate                           0.6578          0.04224
Detection Prevalence                     0.8387          0.08362
Balanced Accuracy                        0.6165          0.63905
                     Class: Acknowledged
Sensitivity                      0.29697
Specificity                      0.95829
Pos Pred Value                   0.53846
Neg Pred Value                   0.89269
Prevalence                       0.14078
Detection Rate                   0.04181
Detection Prevalence             0.07765
Balanced Accuracy                0.62763
```

```r
varImpPlot(model_results$random_forest)  # Displaying the importance of each predictor in
↪  the decision tree
```



We observe that a simple Random Forest model clearly outperforms our Naive Model.
However, the model overepresents "No Disaster" cases, showing good sensitivity but poor specificity.
We also notice that "nat_dis_group" has significantly more impact than the other predictors in this prediction.
Could it be that this predictor is causing an overrepresentation of class 1?
Let's see how the model performs when we keep all selected_predictors but remove nat_dis_group.

5.1.8 M2 - (all selected_predictors - minus "nat_dis_group")

```
focus_predictors <- selected_predictors[-which(selected_predictors ==
    "nat_dis_group")]

model_results <- rf_model(studied_df = df2, model_name =
↪  "M2_RF_(M1_minus_nat_dis_group)",
    focus_predictors = focus_predictors)

results <- bind_rows(results, model_results$new_result)  # Adding the new result to our
↪  historical results
results  # Showing all results
```

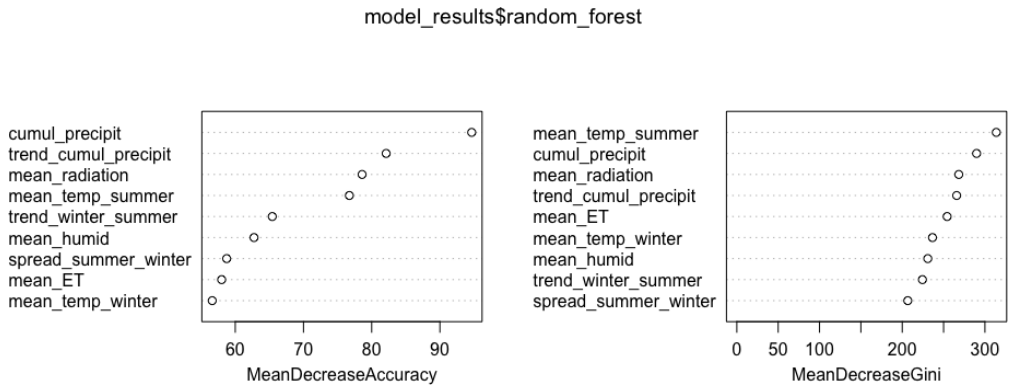
```
                              Accrcy Stivity_2 Stivity_3   F1_2   F1_3  Brier
Target                        > 0.84    > 0.59    > 0.79 > 0.49 > 0.49 < 0.31
naive_model                     0.64      0.09      0.14   0.09   0.14     NA
M1_RF_(All_slctd_prms)          0.74      0.33       0.3    0.4   0.38   0.38
M2_RF_(M1_minus_nat_dis_group)  0.77      0.35      0.52   0.42   0.57   0.35
```

```
model_results$confusion_matrix  # Showing all the metrics
```

```
Confusion Matrix and Statistics

                    Reference
Prediction           No Natural Disaster Requested Acknowledged
  No Natural Disaster                1526       190          153
  Requested                            89       107            6
  Acknowledged                         95         7          171

Overall Statistics

               Accuracy : 0.7696
                 95% CI : (0.752, 0.7865)
```

```
    No Information Rate : 0.7295
    P-Value [Acc > NIR] : 5.003e-06

                  Kappa : 0.4104

 Mcnemar's Test P-Value : 7.228e-11

Statistics by Class:

                     Class: No Natural Disaster Class: Requested
Sensitivity                              0.8924          0.35197
Specificity                              0.4590          0.95343
Pos Pred Value                           0.8165          0.52970
Neg Pred Value                           0.6126          0.90803
Prevalence                               0.7295          0.12969
Detection Rate                           0.6510          0.04565
Detection Prevalence                     0.7974          0.08618
Balanced Accuracy                        0.6757          0.65270
                     Class: Acknowledged
Sensitivity                      0.51818
Specificity                      0.94935
Pos Pred Value                   0.62637
Neg Pred Value                   0.92323
Prevalence                       0.14078
Detection Rate                   0.07295
Detection Prevalence             0.11647
Balanced Accuracy                0.73377
```

```r
varImpPlot(model_results$random_forest)  # Displaying the importance of each predictor in
↪  the decision tree
```



We observe that removing "nat_dis_group" significantly improves the model's performance.

```r
selected_predictors <- selected_predictors[-which(selected_predictors ==
    "nat_dis_group")]
```

5.1.9 Checking missing values impact

Now let's study the impact of missing values: how does the model respond when we remove the predictors with the highest number of missing values?

```
df2_nas %>%
    mutate(remaining_lines = round((nrow(df2) * (1 -
        (na_percent/100))))) %>%
    arrange(desc(na_percent))
```

```
                    na_percent remaining_lines
mean_radiation              90           11732
mean_ET                     89           12905
mean_humid                  73           31676
mean_temp_summer            61           45755
spread_summer_winter        61           45755
trend_winter_summer         61           45755
mean_temp_winter            60           46928
trend_cumul_precipit         5          111454
cumul_precipit               3          113800
nat_dis_group                0          117320
```

We will run a series of tests in 3 phases, based on levels of missing data:/ - Phase 1: remove mean_radiation and mean_ET
- Phase 2: remove mean_radiation, mean_ET, and mean_humid
- Phase 3: remove mean_radiation, mean_ET, mean_humid, mean_temp_summer, spread_summer_winter, trend_winter_summer, and mean_temp_winter.

```
remove_ph1 <- c("mean_radiation", "mean_ET")
focus_predictors_ph1 <- selected_predictors[!selected_predictors %in%
    remove_ph1]
new_result_NA_1 <- rf_model(df2, "Phase 1", focus_predictors_ph1)
new_result_NA_1 <- bind_rows(model_results$new_result,
    new_result_NA_1$new_result)
new_result_NA_1
```

```
                            Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M2_RF_(M1_minus_nat_dis_group)   0.77      0.35      0.52 0.42 0.57  0.35
Phase 1                          0.78      0.27      0.52 0.35 0.56  0.33
```

```
remove_ph2 <- c("mean_radiation", "mean_ET", "mean_humid")
focus_predictors_ph2 <- selected_predictors[!selected_predictors %in%
    remove_ph2]
new_result_NA_2 <- rf_model(df2, "Phase 2", focus_predictors_ph2)
new_result_NA_2 <- bind_rows(new_result_NA_1, new_result_NA_2$new_result)
new_result_NA_2
```

```
                            Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M2_RF_(M1_minus_nat_dis_group)   0.77      0.35      0.52 0.42 0.57  0.35
Phase 1                          0.78      0.27      0.52 0.35 0.56  0.33
Phase 2                          0.78      0.27      0.51 0.35 0.55  0.32
```

```
remove_ph3 <- c("mean_radiation", "mean_ET", "mean_humid",
    "mean_temp_summer", "spread_summer_winter", "trend_winter_summer",
    "mean_temp_winter")
focus_predictors_ph3 <- selected_predictors[!selected_predictors %in%
    remove_ph3]
new_result_NA_3 <- rf_model(df2, "Phase 3", focus_predictors_ph3)
```

```
new_result_NA_3 <- bind_rows(new_result_NA_2, new_result_NA_3$new_result)
new_result_NA_3
```

```
                              Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M2_RF_(M1_minus_nat_dis_group)   0.77      0.35      0.52 0.42 0.57  0.35
Phase 1                          0.78      0.27      0.52 0.35 0.56  0.33
Phase 2                          0.78      0.27      0.51 0.35 0.55  0.32
Phase 3                          0.81      0.31       0.5 0.37 0.56   0.3
```

```
focus_predictors_ph3
```

```
[1] "cumul_precipit"       "trend_cumul_precipit"
```

```
selected_predictors
```

```
[1] "cumul_precipit"       "mean_temp_summer"     "mean_temp_winter"
[4] "mean_humid"           "mean_ET"              "mean_radiation"
[7] "trend_cumul_precipit" "spread_summer_winter" "trend_winter_summer"
```

Through these different tests, we observe that the model tends to overrepresent class 1 at the expense of the other classes.
This seems to explain why removing certain predictors significantly improves sensitivity for classes 2 and 3. However, the model also becomes more confident in its predictions (brier score), which is likely due to the larger amount of available data.

Let's now try a similar phased approach by replacing the missing values with departmental averages:
- Phase 1: replace NAs for "mean_temp_winter", "trend_winter_summer", "spread_summer_winter", and "mean_temp_summer".
- Phase 2: additionally replace NAs for mean_humid, mean_ET, and mean_radiation.

```
# Phase 1
df2_no_nas <- df2 %>%
    group_by(year, department_number) %>%
    mutate(mean_temp_summer = ifelse(is.na(mean_temp_summer),
        mean(mean_temp_summer, na.rm = TRUE), mean_temp_summer),
        mean_temp_winter = ifelse(is.na(mean_temp_winter),
            mean(mean_temp_winter, na.rm = TRUE), mean_temp_winter),
        trend_winter_summer = ifelse(is.na(trend_winter_summer),
            mean(trend_winter_summer, na.rm = TRUE),
            trend_winter_summer), spread_summer_winter =
↪   ifelse(is.na(spread_summer_winter),
            mean(spread_summer_winter, na.rm = TRUE),
            spread_summer_winter)) %>%
    ungroup()

df2_nas <- round(colSums(is.na(df2_no_nas[, selected_predictors]))/nrow(df2_no_nas) *
    100)
df2_nas <- as.data.frame(df2_nas)
colnames(df2_nas) <- c("na_percent")
df2_nas %>%
    mutate(remaining_lines = round((nrow(df2_no_nas) *
        (1 - (na_percent/100))))) %>%
    arrange(desc(na_percent))
```

```
                    na_percent remaining_lines
mean_radiation              90           11732
mean_ET                     89           12905
mean_humid                  73           31676
trend_cumul_precipit         5          111454
cumul_precipit               3          113800
mean_temp_summer             0          117320
mean_temp_winter             0          117320
spread_summer_winter         0          117320
trend_winter_summer          0          117320
```

```
focus_predictors_ph1 <- focus_predictors_ph2
new_result_NA_1 <- rf_model(df2_no_nas, "Phase 1",
    focus_predictors_ph1)
new_result_NA_1 <- bind_rows(model_results$new_result,
    new_result_NA_1$new_result)
new_result_NA_1
```

```
                           Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M2_RF_(M1_minus_nat_dis_group)   0.77      0.35      0.52 0.42 0.57  0.35
Phase 1                          0.81      0.23      0.58 0.31 0.59  0.28
```

```
## Phase 2
df2_no_nas <- df2_no_nas %>%
    group_by(year, department_number) %>%
    mutate(mean_humid = ifelse(is.na(mean_humid), mean(mean_humid,
        na.rm = TRUE), mean_humid), mean_ET = ifelse(is.na(mean_ET),
        mean(mean_ET, na.rm = TRUE), mean_ET), mean_radiation =
        ↪  ifelse(is.na(mean_radiation),
        mean(mean_radiation, na.rm = TRUE), mean_radiation)) %>%
    ungroup()

df2_nas <- round(colSums(is.na(df2_no_nas[, selected_predictors]))/nrow(df2_no_nas) *
    100)
df2_nas <- as.data.frame(df2_nas)
colnames(df2_nas) <- c("na_percent")
df2_nas %>%
    mutate(remaining_lines = round((nrow(df2_no_nas) *
        (1 - (na_percent/100))))) %>%
    arrange(desc(na_percent))
```

```
                    na_percent remaining_lines
trend_cumul_precipit         5          111454
mean_radiation               4          112627
cumul_precipit               3          113800
mean_ET                      3          113800
mean_temp_summer             0          117320
mean_temp_winter             0          117320
mean_humid                   0          117320
spread_summer_winter         0          117320
trend_winter_summer          0          117320
```

```
selected_predictors
```

```
[1] "cumul_precipit"       "mean_temp_summer"     "mean_temp_winter"
```

```
[4] "mean_humid"           "mean_ET"              "mean_radiation"
[7] "trend_cumul_precipit" "spread_summer_winter" "trend_winter_summer"
```

```
focus_predictors_ph2 <- selected_predictors
new_result_NA_2 <- rf_model(df2_no_nas, "Phase 2",
    focus_predictors_ph2)
new_result_NA_2 <- bind_rows(new_result_NA_1, new_result_NA_2$new_result)
new_result_NA_2
```

```
                              Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M2_RF_(M1_minus_nat_dis_group)   0.77      0.35      0.52 0.42 0.57  0.35
Phase 1                          0.81      0.23      0.58 0.31 0.59  0.28
Phase 2                          0.81      0.24      0.52 0.32 0.57  0.28
```

We observe that by replacing the NAs with departmental average values, we obtain a more balanced model:
- Our Global Accuracy exceeds 80%
- We lose slightly in sensitivity for class 3
- We lose more noticeably in sensitivity for class 2
- However, we significantly reduce the model's uncertainty (Brier Score)

5.1.10 M3 - Random Forest - (all selected_predictors + minus " nat_dis_group" + NAs replaced by average department)

Let's integrate this latest result with the previous results:

```
new_result_NA_2 <- new_result_NA_2[3, ]
new_result_NA_2
```

```
        Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
Phase 2   0.81      0.24      0.52 0.32 0.57  0.28
```

```
rownames(new_result_NA_2) <- c("M3_RF_(M2_NAs_replcd_by_avg_dep)")
results <- bind_rows(results, new_result_NA_2)
results
```

```
                               Accrcy Stivity_2 Stivity_3   F1_2   F1_3
Target                         > 0.84    > 0.59    > 0.79 > 0.49 > 0.49
naive_model                      0.64      0.09      0.14   0.09   0.14
M1_RF_(All_slctd_prms)           0.74      0.33       0.3    0.4   0.38
M2_RF_(M1_minus_nat_dis_group)   0.77      0.35      0.52   0.42   0.57
M3_RF_(M2_NAs_replcd_by_avg_dep) 0.81      0.24      0.52   0.32   0.57
                                Brier
Target                         < 0.31
naive_model                       NA
M1_RF_(All_slctd_prms)           0.38
M2_RF_(M1_minus_nat_dis_group)   0.35
M3_RF_(M2_NAs_replcd_by_avg_dep) 0.28
```

We observe that the strong imbalance in our target variable continues to affect our results.
Let's try optimizing our Random Forest function by applying class weights to the minority classes.

5.1.11 M4 - Random Forest - (all selected_predictors + minus " nat_dis_group" + NAs replaced by average department + random_weight)

Here we modify our function to include the following class weighting (randomly chosen):
classwt = c("No Natural Disaster" = 0.60, "Requested" = 0.10, "Acknowledged" = 0.30)

```
rf_model_2 <- function(studied_df, model_name, focus_predictors) {
    set.seed(1)
    test_index <- createDataPartition(studied_df$natural_disaster,
        times = 1, p = 0.2, list = FALSE)
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    model_train_set <- train_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    model_test_set <- test_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    y_train <- model_train_set %>%
        pull(natural_disaster)
    y_train <- as.factor(y_train)

    y_test <- model_test_set %>%
        pull(natural_disaster)
    y_test <- as.factor(y_test)

    x_train <- model_train_set %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)
    # Random Forest Model with Ponreration
    random_forest_fit_1 <- randomForest(natural_disaster ~
        ., data = model_train_set, importance = TRUE,
        keep.forest = TRUE, classwt = c(`No Natural Disaster` = 0.6,
            Requested = 0.1, Acknowledged = 0.3))
    # Generating predictions & metrics
    y_predicted <- predict(random_forest_fit_1, x_test)
    model_results <- confusionMatrix(y_predicted, y_test)
    accuracy <- sum(y_predicted == y_test)/length(y_test)
    y_predicted_prob <- predict(random_forest_fit_1,
        x_test, type = "prob")
    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
```

```
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 2)
    # Output of the metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 2), Stivity_2 = round(model_results$byClass["Class:
        ↪  Requested",
            "Sensitivity"], 2), Stivity_3 = round(model_results$byClass["Class:
            ↪  Acknowledged",
            "Sensitivity"], 2), F1_2 = round(model_results$byClass["Class: Requested",
            "F1"], 2), F1_3 = round(model_results$byClass["Class: Acknowledged",
            "F1"], 2), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}
```

Running the function

```
model_results <- rf_model_2(studied_df = df2_no_nas,
    model_name = "M4_RF_(M3_random_weight)", focus_predictors = selected_predictors)
    ↪  #Running the function
model_results$new_result  # Showing the new result
```

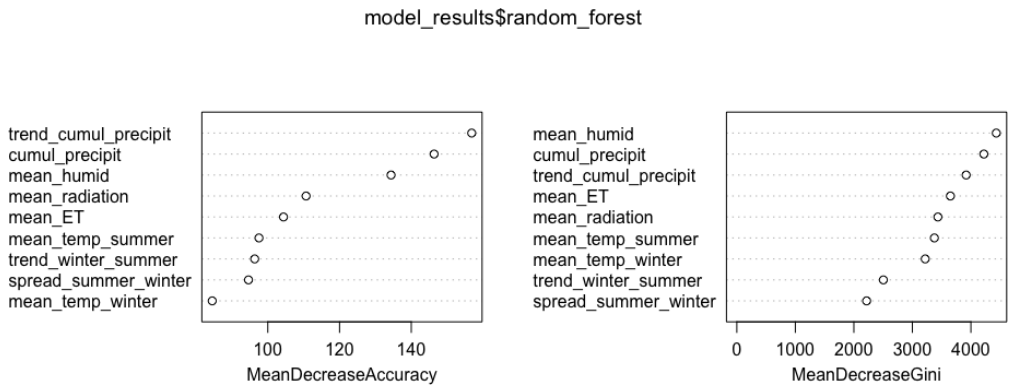
```
                         Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M4_RF_(M3_random_weight)   0.79      0.36      0.78  0.4 0.64  0.31
```

```
model_results$confusion_matrix  # Showing all metrics
```

```
Confusion Matrix and Statistics

                    Reference
Prediction           No Natural Disaster Requested Acknowledged
  No Natural Disaster               14126      1055          614
  Requested                           797       668           31
  Acknowledged                       1804       117         2281

Overall Statistics

               Accuracy : 0.7944
                 95% CI : (0.789, 0.7998)
    No Information Rate : 0.7783
    P-Value [Acc > NIR] : 4.314e-09

                  Kappa : 0.4803

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: No Natural Disaster Class: Requested
Sensitivity                              0.8445          0.36304
```

```
Specificity                            0.6498          0.95787
Pos Pred Value                         0.8943          0.44652
Neg Pred Value                         0.5435          0.94139
Prevalence                             0.7783          0.08561
Detection Rate                         0.6572          0.03108
Detection Prevalence                   0.7349          0.06960
Balanced Accuracy                      0.7472          0.66046
                     Class: Acknowledged
Sensitivity                        0.7796
Specificity                        0.8965
Pos Pred Value                     0.5428
Neg Pred Value                     0.9627
Prevalence                         0.1361
Detection Rate                     0.1061
Detection Prevalence               0.1955
Balanced Accuracy                  0.8380
```

```r
varImpPlot(model_results$random_forest)  # Displaying the importance of each predictor in
↪  the decision tree
```




```r
results <- bind_rows(results, model_results$new_result)  # Adding the new result to
↪  historal results
results  # Showing latest results
```

```
                                Accrcy Stivity_2 Stivity_3   F1_2   F1_3
Target                          > 0.84    > 0.59    > 0.79 > 0.49 > 0.49
naive_model                       0.64      0.09      0.14   0.09   0.14
M1_RF_(All_slctd_prms)            0.74      0.33       0.3    0.4   0.38
M2_RF_(M1_minus_nat_dis_group)    0.77      0.35      0.52   0.42   0.57
M3_RF_(M2_NAs_replcd_by_avg_dep)  0.81      0.24      0.52   0.32   0.57
M4_RF_(M3_random_weight)          0.79      0.36      0.78    0.4   0.64
                                 Brier
Target                          < 0.31
naive_model                        NA
M1_RF_(All_slctd_prms)            0.38
M2_RF_(M1_minus_nat_dis_group)    0.35
M3_RF_(M2_NAs_replcd_by_avg_dep)  0.28
M4_RF_(M3_random_weight)          0.31
```

We observe that by applying class weighting to the minority classes, we achieve very good performance. Ideally, we should run several weighting tests to find the best configuration.
However, we are reaching the limits of our current "randomForest" function, which already takes about 2

minutes to run each time.

Therefore, we will now test a more efficient model that supports parallel processing: "ranger()".

5.1.12 M5 - Ranger - (all selected_predictors + minus " nat_dis_group" + NAs replaced by average department)

```
model_ranger <- function(studied_df, model_name, focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    test_index <- createDataPartition(studied_df$natural_disaster,
        times = 1, p = 0.2, list = FALSE)
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    model_train_set <- train_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    model_test_set <- test_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    y_train <- model_train_set %>%
        pull(natural_disaster)
    y_train <- as.factor(y_train)

    y_test <- model_test_set %>%
        pull(natural_disaster)
    y_test <- as.factor(y_test)

    x_train <- model_train_set %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)
    # Ranger Model
    random_forest_fit_1 <- ranger(formula = natural_disaster ~
        ., data = model_train_set, importance = "impurity",
        probability = TRUE, num.trees = 500, num.threads = parallel::detectCores())
    # Generating predictions & metrics
    y_predicted_prob <- predict(random_forest_fit_1,
        x_test)
    y_predicted_prob <- y_predicted_prob$predictions
```

```
    y_predicted <- apply(y_predicted_prob, 1, function(x)
↪   colnames(y_predicted_prob)[which.max(x)])
    y_predicted <- factor(y_predicted, levels = levels(y_test))

    model_results <- confusionMatrix(y_predicted, y_test)
    accuracy <- sum(y_predicted == y_test)/length(y_test)

    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 2)
    # Output of metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 2), Stivity_2 = round(model_results$byClass["Class:
        ↪  Requested",
            "Sensitivity"], 2), Stivity_3 = round(model_results$byClass["Class:
            ↪  Acknowledged",
            "Sensitivity"], 2), F1_2 = round(model_results$byClass["Class: Requested",
            "F1"], 2), F1_3 = round(model_results$byClass["Class: Acknowledged",
            "F1"], 2), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}
```

We first apply the ranger() function without class weighting.

```
model_results <- model_ranger(studied_df = df2_no_nas,
    model_name = "M5_Ranger_(M3)", focus_predictors = selected_predictors)  # Running the
    ↪  function
results <- bind_rows(results, model_results$new_result)  # Adding new_result to
↪  historical results
results  # Showing latest results
```

```
                                Accrcy Stivity_2 Stivity_3   F1_2   F1_3
Target                          > 0.84    > 0.59    > 0.79 > 0.49 > 0.49
naive_model                       0.64      0.09      0.14   0.09   0.14
M1_RF_(All_slctd_prms)            0.74      0.33       0.3    0.4   0.38
M2_RF_(M1_minus_nat_dis_group)    0.77      0.35      0.52   0.42   0.57
M3_RF_(M2_NAs_replcd_by_avg_dep)  0.81      0.24      0.52   0.32   0.57
M4_RF_(M3_random_weight)          0.79      0.36      0.78    0.4   0.64
M5_Ranger_(M3)                    0.81      0.24      0.55   0.33   0.58
                                 Brier
Target                          < 0.31
naive_model                        NA
M1_RF_(All_slctd_prms)            0.38
M2_RF_(M1_minus_nat_dis_group)    0.35
M3_RF_(M2_NAs_replcd_by_avg_dep)  0.28
M4_RF_(M3_random_weight)          0.31
M5_Ranger_(M3)                    0.26
```

When we compare ranger() with randomForest(), we observe a drop in sensitivity for class 3, but an improvement in the Brier score.

5.1.13 M6 - Ranger - (all selected_predictors + minus " nat_dis_group" + NAs replaced by average department + random_weight)

The parameters of the ranger() function do not allow true class weighting by oversampling, as in randomForest().
Instead, it applies a penalty in the cost function during tree construction.
Therefore, we will need to manually oversample the minority classes to reproduce similar conditions.

```
model_ranger_weighted <- function(studied_df, model_name,
    weight_2, weight_3, focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    test_index <- createDataPartition(studied_df$natural_disaster,
        times = 1, p = 0.2, list = FALSE)
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    # Manual duplication of minority classes (2
    # and 3)
    set.seed(1)
    multiple_class_2 <- train_set %>%
        filter(natural_disaster == "Requested")
    multiple_class_2 <- multiple_class_2 %>%
        sample_n(size = round(nrow(multiple_class_2) *
            weight_2), replace = TRUE)

    multiple_class_3 <- train_set %>%
        filter(natural_disaster == "Acknowledged")
    multiple_class_3 <- multiple_class_3 %>%
        sample_n(size = round(nrow(multiple_class_3) *
            weight_3), replace = TRUE)

    train_set_weighted <- bind_rows(train_set, multiple_class_2,
        multiple_class_3)

    model_train_set_weighted <- train_set_weighted %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()

    model_test_set <- test_set %>%
        select(focus_predictors, natural_disaster) %>%
        drop_na()
```

```
    y_train <- model_train_set_weighted %>%
        pull(natural_disaster)
    y_train <- as.factor(y_train)

    y_test <- model_test_set %>%
        pull(natural_disaster)
    y_test <- as.factor(y_test)

    x_train <- model_train_set_weighted %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)
    # Ranger Model
    random_forest_fit_1 <- ranger(formula = natural_disaster ~
        ., data = model_train_set_weighted, importance = "impurity",
        probability = TRUE, num.trees = 500, num.threads = parallel::detectCores())
    # Generating predictions and metrics
    y_predicted_prob <- predict(random_forest_fit_1,
        x_test)
    y_predicted_prob <- y_predicted_prob$predictions

    y_predicted <- apply(y_predicted_prob, 1, function(x)
↪   colnames(y_predicted_prob)[which.max(x)])
    y_predicted <- factor(y_predicted, levels = levels(y_test))

    model_results <- confusionMatrix(y_predicted, y_test)
    accuracy <- sum(y_predicted == y_test)/length(y_test)

    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 2)

    # Output of the metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 2), Stivity_2 = round(model_results$byClass["Class:
        ↪  Requested",
            "Sensitivity"], 2), Stivity_3 = round(model_results$byClass["Class:
            ↪  Acknowledged",
            "Sensitivity"], 2), F1_2 = round(model_results$byClass["Class: Requested",
            "F1"], 2), F1_3 = round(model_results$byClass["Class: Acknowledged",
            "F1"], 2), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}
```

We will test different weightings for class 3 to evaluate their impact on the results and optimize its performance.

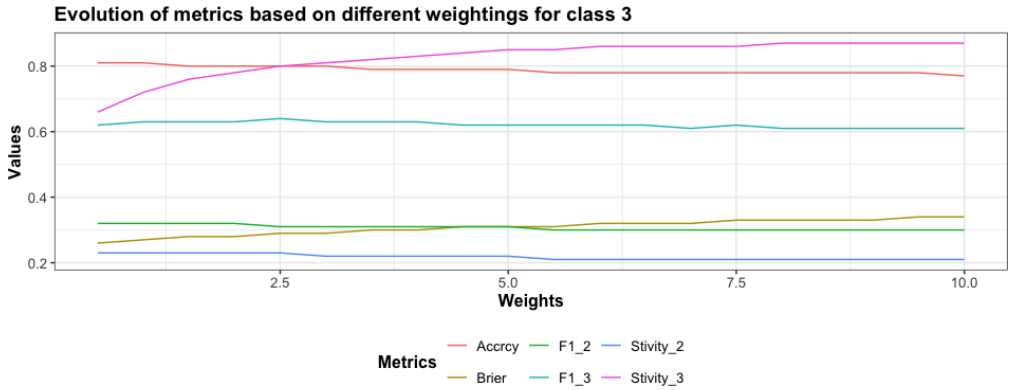
```
weight_3 <- seq(from = 0.5, to = 10, by = 0.5)
length(weight_3)
```

```
[1] 20
```

```
all_w3_test <- map_df(weight_3, function(w) {
    model_results <- model_ranger_weighted(df2_no_nas,
        paste0("Ranger_model_w3: ", w), 0, w, selected_predictors)
    model_results$new_result
})

all_w3_test
```

```
                   Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
Ranger_model_w3: 0.5   0.81      0.23      0.66 0.32 0.62  0.26
Ranger_model_w3: 1     0.81      0.23      0.72 0.32 0.63  0.27
Ranger_model_w3: 1.5    0.8      0.23      0.76 0.32 0.63  0.28
Ranger_model_w3: 2      0.8      0.23      0.78 0.32 0.63  0.28
Ranger_model_w3: 2.5    0.8      0.23       0.8 0.31 0.64  0.29
Ranger_model_w3: 3      0.8      0.22      0.81 0.31 0.63  0.29
Ranger_model_w3: 3.5   0.79      0.22      0.82 0.31 0.63   0.3
Ranger_model_w3: 4     0.79      0.22      0.83 0.31 0.63   0.3
Ranger_model_w3: 4.5   0.79      0.22      0.84 0.31 0.62  0.31
Ranger_model_w3: 5     0.79      0.22      0.85 0.31 0.62  0.31
Ranger_model_w3: 5.5   0.78      0.21      0.85  0.3 0.62  0.31
Ranger_model_w3: 6     0.78      0.21      0.86  0.3 0.62  0.32
Ranger_model_w3: 6.5   0.78      0.21      0.86  0.3 0.62  0.32
Ranger_model_w3: 7     0.78      0.21      0.86  0.3 0.61  0.32
Ranger_model_w3: 7.5   0.78      0.21      0.86  0.3 0.62  0.33
Ranger_model_w3: 8     0.78      0.21      0.87  0.3 0.61  0.33
Ranger_model_w3: 8.5   0.78      0.21      0.87  0.3 0.61  0.33
Ranger_model_w3: 9     0.78      0.21      0.87  0.3 0.61  0.33
Ranger_model_w3: 9.5   0.78      0.21      0.87  0.3 0.61  0.34
Ranger_model_w3: 10    0.77      0.21      0.87  0.3 0.61  0.34
```

```
all_w3_test %>%
    mutate(weight_3 = weight_3) %>%
    mutate_all(as.numeric) %>%
    pivot_longer(cols = -weight_3, names_to = "Metrics",
        values_to = "Values") %>%
    ggplot(aes(x = weight_3, y = Values, color = Metrics)) +
    geom_line() + labs(title = "Evolution of metrics based on different weightings for
    ↪  class 3",
    x = "Weights", y = "Values") + standard_theme
```



We observe that the more we increase the weighting on this class, the more we lose in overall performance and in precision (Brier score).

Now let's study the impact of applying a weight to class 2.

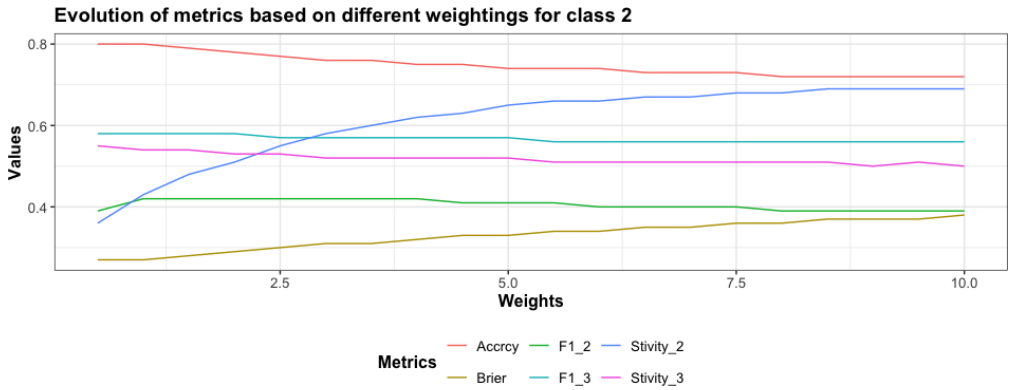
```
weight_2 <- seq(from = 0.5, to = 10, by = 0.5)
length(weight_2)
```

```
[1] 20
```

```
all_w2_test <- map_df(weight_2, function(w) {
    model_results <- model_ranger_weighted(df2_no_nas,
        paste0("Ranger_model_w2: ", w), w, 0, selected_predictors)
    model_results$new_result
})

all_w2_test
```

```
                    Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
Ranger_model_w2: 0.5    0.8      0.36      0.55 0.39 0.58  0.27
Ranger_model_w2: 1      0.8      0.43      0.54 0.42 0.58  0.27
Ranger_model_w2: 1.5   0.79      0.48      0.54 0.42 0.58  0.28
Ranger_model_w2: 2     0.78      0.51      0.53 0.42 0.58  0.29
Ranger_model_w2: 2.5   0.77      0.55      0.53 0.42 0.57   0.3
Ranger_model_w2: 3     0.76      0.58      0.52 0.42 0.57  0.31
Ranger_model_w2: 3.5   0.76       0.6      0.52 0.42 0.57  0.31
Ranger_model_w2: 4     0.75      0.62      0.52 0.42 0.57  0.32
Ranger_model_w2: 4.5   0.75      0.63      0.52 0.41 0.57  0.33
Ranger_model_w2: 5     0.74      0.65      0.52 0.41 0.57  0.33
Ranger_model_w2: 5.5   0.74      0.66      0.51 0.41 0.56  0.34
Ranger_model_w2: 6     0.74      0.66      0.51  0.4 0.56  0.34
Ranger_model_w2: 6.5   0.73      0.67      0.51  0.4 0.56  0.35
Ranger_model_w2: 7     0.73      0.67      0.51  0.4 0.56  0.35
Ranger_model_w2: 7.5   0.73      0.68      0.51  0.4 0.56  0.36
Ranger_model_w2: 8     0.72      0.68      0.51 0.39 0.56  0.36
Ranger_model_w2: 8.5   0.72      0.69      0.51 0.39 0.56  0.37
Ranger_model_w2: 9     0.72      0.69       0.5 0.39 0.56  0.37
Ranger_model_w2: 9.5   0.72      0.69      0.51 0.39 0.56  0.37
Ranger_model_w2: 10    0.72      0.69       0.5 0.39 0.56  0.38
```

```r
all_w2_test %>%
    mutate(weight_2 = weight_2) %>%
    mutate_all(as.numeric) %>%
    pivot_longer(cols = -weight_2, names_to = "Metrics",
        values_to = "Values") %>%
    ggplot(aes(x = weight_2, y = Values, color = Metrics)) +
    geom_line() + labs(title = "Evolution of metrics based on different weightings for
    ↪  class 2",
    x = "Weights", y = "Values") + standard_theme
```



When analyzing the effect of weighting class 2, we observe the following:
- Increasing the weight of class 2 significantly impacts global accuracy and sensitivity for class 1.
- There is a sharp decline, indicating that the model needs to predict many false class 2 cases to correctly identify a few true ones—at the expense of classes 1 and 3.
- The Brier score increases rapidly beyond our target of 0.35, indicating a loss in model confidence.
- As we approach a reasonable sensitivity for class 2, signs of overfitting begin to appear.

This outcome makes sense, as class 2 corresponds to natural disaster requests that were not officially acknowledged by the government. These cases are likely similar to class 1 (no disaster), in terms of city/year characteristics.

This result suggests that we are missing relevant predictors to properly identify class 2 cases.

Let's focus our efforts on classes 1 and 3.
First, let's revisit the weighting for class 3 to determine the most performant configuration.

```r
weight_3 <- seq(from = 1.4, to = 3.4, by = 0.15)
length(weight_3)
```

```
[1] 14
```

```r
all_w3_test <- map_df(weight_3, function(w) {
    model_results <- model_ranger_weighted(df2_no_nas,
        paste0("Ranger_model_w3: ", w), 0, w, selected_predictors)
    model_results$new_result
})

all_w3_test
```

```
                     Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
Ranger_model_w3: 1.4     0.8      0.23      0.75 0.32 0.63  0.27
Ranger_model_w3: 1.55    0.8      0.23      0.76 0.32 0.63  0.28
Ranger_model_w3: 1.7     0.8      0.23      0.77 0.31 0.64  0.28
Ranger_model_w3: 1.85    0.8      0.22      0.77 0.31 0.63  0.28
Ranger_model_w3: 2       0.8      0.23      0.78 0.32 0.63  0.28
Ranger_model_w3: 2.15    0.8      0.23      0.79 0.32 0.64  0.28
Ranger_model_w3: 2.3     0.8      0.22       0.8 0.31 0.64  0.28
Ranger_model_w3: 2.45    0.8      0.22       0.8 0.31 0.63  0.29
Ranger_model_w3: 2.6     0.8      0.22      0.81 0.31 0.63  0.29
Ranger_model_w3: 2.75    0.8      0.22      0.81 0.31 0.63  0.29
Ranger_model_w3: 2.9     0.8      0.22      0.81 0.31 0.63  0.29
Ranger_model_w3: 3.05    0.8      0.22      0.81 0.31 0.63  0.29
Ranger_model_w3: 3.2    0.79      0.22      0.82 0.31 0.63  0.29
Ranger_model_w3: 3.35   0.79      0.22      0.82 0.31 0.63   0.3
```

```
best_w3 <- all_w3_test %>%
    mutate(weight_3 = weight_3) %>%
    filter(Accrcy > 0.78, Stivity_3 > 0.79, Brier <
        0.31) %>%
    arrange(-desc(Brier)) %>%
    dplyr::slice(1:1) %>%
    pull(weight_3)
best_w3
```

```
[1] 2.3
```

```
# Let's save our new result
model_results <- model_ranger_weighted(df2_no_nas,
    "M6_Ranger_(M5_best_weight) ", 0, best_w3, selected_predictors)
model_results$new_result
```

```
                          Accrcy Stivity_2 Stivity_3 F1_2 F1_3 Brier
M6_Ranger_(M5_best_weight)    0.8      0.22       0.8 0.31 0.64  0.28
```

```
results <- bind_rows(results, model_results$new_result)
results
```

```
                                Accrcy Stivity_2 Stivity_3   F1_2   F1_3
Target                          > 0.84    > 0.59    > 0.79 > 0.49 > 0.49
naive_model                       0.64      0.09      0.14   0.09   0.14
M1_RF_(All_slctd_prms)            0.74      0.33       0.3    0.4   0.38
M2_RF_(M1_minus_nat_dis_group)    0.77      0.35      0.52   0.42   0.57
M3_RF_(M2_NAs_replcd_by_avg_dep)  0.81      0.24      0.52   0.32   0.57
M4_RF_(M3_random_weight)          0.79      0.36      0.78    0.4   0.64
M5_Ranger_(M3)                    0.81      0.24      0.55   0.33   0.58
M6_Ranger_(M5_best_weight)         0.8      0.22       0.8   0.31   0.64
                                Brier
Target                          < 0.31
naive_model                         NA
M1_RF_(All_slctd_prms)            0.38
M2_RF_(M1_minus_nat_dis_group)    0.35
M3_RF_(M2_NAs_replcd_by_avg_dep)  0.28
M4_RF_(M3_random_weight)          0.31
M5_Ranger_(M3)                    0.26
```

```
M6_Ranger_(M5_best_weight)          0.28
```

Now let's focus exclusively on predicting classes 1 and 3 (i.e., y = "nat_dis")

5.2 Binary Prediction Models

5.2.1 Metrics & Target

Let's redefine our objectives given our new context (binary classification)

```
results <- data.frame(row.names = "Target", Accrcy = "> 0.84",
    Stivity = "> 0.79", F1 = "> 0.59", Brier = "< 0.11")
```

5.2.2 Creating train & test set

```
set.seed(1)
test_index <- createDataPartition(df2$natural_disaster,
    times = 1, p = 0.2, list = FALSE)
test_set <- df2[test_index, ]
train_set <- df2[-test_index, ]
missing_cities <- test_set %>%
    filter(!insee_code %in% train_set$insee_code)
train_set <- train_set %>%
    union(missing_cities)
test_set <- test_set %>%
    filter(insee_code %in% train_set$insee_code)
```

5.2.3 Naive Model

To establish a baseline, let's start by creating a "naive" model that randomly generates the target class (0, 1) while preserving their overall proportions in the dataset.

```
percentage <- train_set %>%
    group_by(nat_dis) %>%
    summarise(count = n()) %>%
    mutate(percent = count/sum(count))
percentage
```

```
# A tibble: 2 x 3
  nat_dis count percent
    <dbl> <int>   <dbl>
1       0 81231   0.865
2       1 12624   0.135
```

```
percentage[which(percentage$nat_dis == 0), ]$percent
```

```
[1] 0.8654946
```

```
set.seed(1)
naive_model <- sample(c(0, 1), size = nrow(test_set),
    replace = TRUE, prob = c(percentage[which(percentage$nat_dis ==
        0), ]$percent, percentage[which(percentage$nat_dis ==
        1), ]$percent))

y_test <- test_set$nat_dis
accuracy <- sum(y_test == naive_model)/length(y_test)
accuracy
```

```
[1] 0.7652674
```

```
results_naive_model <- confusionMatrix(factor(naive_model),
    factor(y_test), positive = "1")
results_naive_model$byClass
```

```
         Sensitivity          Specificity       Pos Pred Value
           0.13937282           0.86256648           0.13618075
      Neg Pred Value            Precision               Recall
           0.86572106           0.13618075           0.13937282
                  F1           Prevalence       Detection Rate
           0.13775830           0.13454081           0.01875133
Detection Prevalence    Balanced Accuracy
           0.13769444           0.50096965
```

```
new_result <- data.frame(row.names = "naive_model",
    Accrcy = round(accuracy, 4), Stivity =
    ↪  round(results_naive_model$byClass["Sensitivity"],
        4), F1 = round(results_naive_model$byClass["F1"],
        4), Brier = "NA")
new_result <- new_result %>%
    mutate(across(everything(), as.character))
new_result
```

```
            Accrcy Stivity     F1 Brier
naive_model 0.7653  0.1394 0.1378    NA
```

```
results <- bind_rows(results, new_result)
results
```

```
            Accrcy Stivity     F1  Brier
Target      > 0.84  > 0.79 > 0.59 < 0.11
naive_model 0.7653  0.1394 0.1378     NA
```

5.2.4 M1 - Random Forest - best_predictors

We recreate our randomForest function using the preselected predictors (selected_predictors)

```
rf_model <- function(studied_df, model_name, w3, focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    studied_df <- studied_df %>%
        mutate(nat_dis = as.factor(nat_dis))
    test_index <- createDataPartition(studied_df$natural_disaster,
```

```
    times = 1, p = 0.2, list = FALSE)
test_set <- studied_df[test_index, ]
train_set <- studied_df[-test_index, ]
missing_cities <- test_set %>%
    filter(!insee_code %in% train_set$insee_code)
train_set <- train_set %>%
    union(missing_cities)
test_set <- test_set %>%
    filter(insee_code %in% train_set$insee_code)

# Manual duplication of minority
multiple_class_3 <- train_set %>%
    filter(nat_dis == 1)
multiple_class_3 <- multiple_class_3 %>%
    sample_n(size = round(nrow(multiple_class_3) *
        w3), replace = TRUE)

train_set_weighted <- bind_rows(train_set, multiple_class_3)

model_train_set <- train_set_weighted %>%
    select(focus_predictors, nat_dis) %>%
    drop_na()

model_test_set <- test_set %>%
    select(focus_predictors, nat_dis) %>%
    drop_na()

y_train <- model_train_set %>%
    pull(nat_dis)
y_train <- as.factor(y_train)

y_test <- model_test_set %>%
    pull(nat_dis)
y_test <- as.factor(y_test)

x_train <- model_train_set %>%
    select(focus_predictors)

x_test <- model_test_set %>%
    select(focus_predictors)
# Random Forest Model
random_forest_fit_1 <- randomForest(nat_dis ~ .,
    data = model_train_set, importance = TRUE,
    keep.forest = TRUE)
# Creating predictions & metrics
y_predicted <- predict(random_forest_fit_1, x_test)

model_results <- confusionMatrix(y_predicted, y_test,
    positive = "1")
accuracy <- sum(y_predicted == y_test)/length(y_test)

y_predicted_prob <- predict(random_forest_fit_1,
    x_test, type = "prob")
```

```
    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 4)
    # Outup of metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 4), Stivity =
        ↪  round(model_results$byClass["Sensitivity"],
            4), F1 = round(model_results$byClass["F1"],
            4), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}
```

Running the function

```
best_predictors <- selected_predictors
model_results <- rf_model(df2, "M1_RF_(best_prms)",
    0, best_predictors)  # Running our function
results <- bind_rows(results, model_results$new_result)  # Adding new result to
↪  historical results
results  # Showing latest result
```

```
                 Accrcy Stivity     F1  Brier
Target           > 0.84  > 0.79 > 0.59 < 0.11
naive_model      0.7653  0.1394 0.1378     NA
M1_RF_(best_prms) 0.8869  0.5061 0.5576 0.1746
```

We observe that an unparameterized Random Forest model already provides much more relevant results than our naive model.

5.2.5 M2 - Random Forest - (best_predictors + NA replaced)

```
model_results <- rf_model(df2_no_nas, "M2_RF_(M1_NAs_replcd_by_avg_dep)",
    0, best_predictors)  # Running our function

results <- bind_rows(results, model_results$new_result)  # Adding new result to
↪  historical results
results  # Showing latest result
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
```

As observed earlier, when we replace NAs with departmental averages, we notice a slight loss in sensitivity but a considerable improvement in the Brier score.

5.2.6 M3 - Random Forest - (best_predictors + NA replaced + weighted)

```
model_results <- rf_model(df2_no_nas, "M3_RF_(M2_weighted)",
    best_w3, best_predictors)  # Running our function
results <- bind_rows(results, model_results$new_result)  # Adding new result to
↪  historical results
results  # Showing latest result
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
```

We observe a loss in accuracy and Brier score, but a significant gain in sensitivity.

5.2.7 M4 - Ranger - (best_predictors + NA replaced)

```
model_ranger <- function(studied_df, model_name, w3,
    focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    studied_df <- studied_df %>%
        mutate(nat_dis = as.factor(nat_dis))
    test_index <- createDataPartition(studied_df$natural_disaster,
        times = 1, p = 0.2, list = FALSE)
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    # Manual duplication of minority
    multiple_class_3 <- train_set %>%
        filter(nat_dis == 1)
    multiple_class_3 <- multiple_class_3 %>%
        sample_n(size = round(nrow(multiple_class_3) *
            w3), replace = TRUE)

    train_set_weighted <- bind_rows(train_set, multiple_class_3)

    model_train_set <- train_set_weighted %>%
        select(focus_predictors, nat_dis) %>%
        drop_na()

    model_test_set <- test_set %>%
        select(focus_predictors, nat_dis) %>%
```

```
        drop_na()

    y_train <- model_train_set %>%
        pull(nat_dis)
    y_train <- as.factor(y_train)

    y_test <- model_test_set %>%
        pull(nat_dis)
    y_test <- as.factor(y_test)

    x_train <- model_train_set %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)

    # Ranger Model
    random_forest_fit_1 <- ranger(formula = nat_dis ~
        ., data = model_train_set, importance = "impurity",
        probability = TRUE, num.trees = 500, num.threads = parallel::detectCores())

    # Creating predictions & metrics
    y_predicted_prob <- predict(random_forest_fit_1,
        x_test)
    y_predicted_prob <- y_predicted_prob$predictions

    y_predicted <- apply(y_predicted_prob, 1, function(x)
↪   colnames(y_predicted_prob)[which.max(x)])
    y_predicted <- factor(y_predicted, levels = levels(y_test))

    model_results <- confusionMatrix(y_predicted, y_test,
        positive = "1")
    accuracy <- sum(y_predicted == y_test)/length(y_test)

    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 4)

    # Output of metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 4), Stivity =
        ↪  round(model_results$byClass["Sensitivity"],
            4), F1 = round(model_results$byClass["F1"],
            4), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}
```

Running the function

```
model_results <- model_ranger(df2_no_nas, "M4_Ranger_(M2)",
    0, best_predictors)  # Running our function

results <- bind_rows(results, model_results$new_result)  # Adding new result to
↪  historical results
results  # Showing latest results
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                  0.8931  0.5338 0.5762 0.1385
```

Compared to M2, we achieve better accuracy and Brier score, but with a slight loss in sensitivity.

5.2.8 M5 - Ranger - (best_predictors + NA replaced + weighted)

```
model_results <- model_ranger(df2_no_nas, "M5_Ranger_(M3)",
    best_w3, best_predictors)  # Running our function

results <- bind_rows(results, model_results$new_result)  # Adding new result to
↪  historical results
results  # Showing latest results
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                  0.8931  0.5338 0.5762 0.1385
M5_Ranger_(M3)                  0.8785  0.7878 0.6384 0.1673
```

We obtain slightly lower performance compared to our model 3 (randomForest()), however, we significantly improve the model's confidence (Brier score).

5.2.9 M6 - Ranger - (best_predictors + NA replaced + weighted + threshold)

Let's now see if we can further refine performance by testing different probability thresholds for decision-making.

```
model_ranger <- function(studied_df, model_name, w3,
    x_threshold, focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    studied_df <- studied_df %>%
        mutate(nat_dis = as.factor(nat_dis))
    test_index <- createDataPartition(studied_df$natural_disaster,
```

```
    times = 1, p = 0.2, list = FALSE)
test_set <- studied_df[test_index, ]
train_set <- studied_df[-test_index, ]
missing_cities <- test_set %>%
    filter(!insee_code %in% train_set$insee_code)
train_set <- train_set %>%
    union(missing_cities)
test_set <- test_set %>%
    filter(insee_code %in% train_set$insee_code)

# Manual duplication of minority
multiple_class_3 <- train_set %>%
    filter(nat_dis == 1)
multiple_class_3 <- multiple_class_3 %>%
    sample_n(size = round(nrow(multiple_class_3) *
        w3), replace = TRUE)

train_set_weighted <- bind_rows(train_set, multiple_class_3)

model_train_set <- train_set_weighted %>%
    select(focus_predictors, nat_dis) %>%
    drop_na()

model_test_set <- test_set %>%
    select(focus_predictors, nat_dis) %>%
    drop_na()

y_train <- model_train_set %>%
    pull(nat_dis)
y_train <- as.factor(y_train)

y_test <- model_test_set %>%
    pull(nat_dis)
y_test <- as.factor(y_test)

x_train <- model_train_set %>%
    select(focus_predictors)

x_test <- model_test_set %>%
    select(focus_predictors)

random_forest_fit_1 <- ranger(formula = nat_dis ~
    ., data = model_train_set, importance = "impurity",
    probability = TRUE, num.trees = 500, num.threads = parallel::detectCores())
y_predicted_prob <- predict(random_forest_fit_1,
    x_test)
y_predicted_prob <- y_predicted_prob$predictions

# Creating predictions & metrics
proba_class1 <- y_predicted_prob[, "1"]
y_predicted <- ifelse(proba_class1 > x_threshold,
    1, 0)  # Creating the decision threshold
y_predicted <- factor(y_predicted, levels = levels(y_test))
```

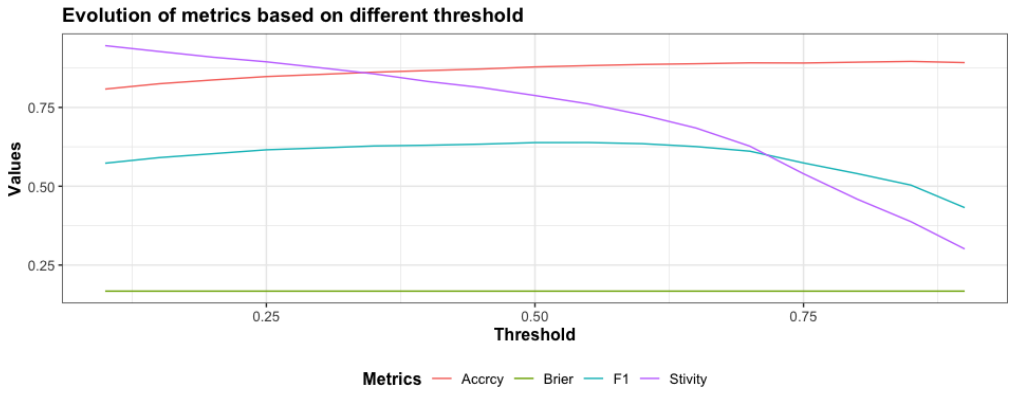
```
    model_results <- confusionMatrix(y_predicted, y_test,
        positive = "1")
    accuracy <- sum(y_predicted == y_test)/length(y_test)

    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 4)

    # Output of the metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 4), Stivity =
        ↪  round(model_results$byClass["Sensitivity"],
            4), F1 = round(model_results$byClass["F1"],
            4), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 3 elements
    return(list(new_result = new_result, confusion_matrix = model_results,
        random_forest = random_forest_fit_1))
}

# Let's test different thresholds
threshold <- seq(from = 0.1, to = 0.9, by = 0.05)
length(threshold)
```

```
[1] 17
```

```
all_threshold_test <- map_df(threshold, function(x_threshold) {
    model_results <- model_ranger(df2_no_nas, paste0("threshold: ",
        x_threshold), best_w3, x_threshold, best_predictors)
    model_results$new_result
})
all_threshold_test
```

```
               Accrcy Stivity     F1  Brier
threshold: 0.1  0.8082   0.946 0.5731 0.1673
threshold: 0.15 0.8251  0.9275 0.5908 0.1673
threshold: 0.2  0.8371  0.9094 0.6032 0.1673
threshold: 0.25 0.8479  0.8947 0.6156 0.1673
threshold: 0.3  0.8547  0.8759 0.6214 0.1673
threshold: 0.35 0.8617  0.8565 0.6278 0.1673
threshold: 0.4  0.8669  0.8325   0.63 0.1673
threshold: 0.45 0.8719  0.8131 0.6335 0.1673
threshold: 0.5  0.8785  0.7878 0.6384 0.1673
threshold: 0.55 0.8827  0.7611 0.6386 0.1673
threshold: 0.6  0.8863  0.7262  0.635 0.1673
threshold: 0.65 0.8886  0.6846 0.6258 0.1673
threshold: 0.7  0.8914  0.6271 0.6112 0.1673
threshold: 0.75 0.8909  0.5396 0.5739 0.1673
threshold: 0.8  0.8936   0.459 0.5401 0.1673
threshold: 0.85  0.896  0.3876 0.5036 0.1673
threshold: 0.9  0.8922  0.3011 0.4321 0.1673
```

```
all_threshold_test %>%
    mutate(threshold = threshold) %>%
    mutate_all(as.numeric) %>%
    pivot_longer(cols = -threshold, names_to = "Metrics",
        values_to = "Values") %>%
    ggplot(aes(x = threshold, y = Values, color = Metrics)) +
    geom_line() + labs(title = "Evolution of metrics based on different threshold",
    x = "Threshold", y = "Values") + standard_theme
```



In our context, we will select the threshold that makes our model:
- the most balanced (with the highest F1 score)
- with a sensitivity greater than 80%

```
best_threshold <- all_threshold_test %>%
    mutate(threshold = threshold) %>%
    filter(Stivity > 0.8) %>%
    arrange(desc(F1)) %>%
    dplyr::slice(1:1) %>%
    pull(threshold)
best_threshold
```

```
[1] 0.45
```

Let's add the best result to the results tracking table

```
new_result <- all_threshold_test %>%
    mutate(threshold = threshold) %>%
    filter(Stivity > 0.8) %>%
    arrange(desc(F1)) %>%
    dplyr::slice(1:1) %>%
    select(-threshold)

rownames(new_result) <- c("M6_Ranger_(M5_threshold)")  # Creating new result name
results <- bind_rows(results, new_result)  # Adding new result to historical results
results  # Showing latest results
```

```
                              Accrcy Stivity     F1  Brier
Target                        > 0.84  > 0.79 > 0.59 < 0.11
naive_model                   0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)             0.8869  0.5061 0.5576 0.1746
```

```
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)              0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                   0.8931  0.5338 0.5762 0.1385
M5_Ranger_(M3)                   0.8785  0.7878 0.6384 0.1673
M6_Ranger_(M5_threshold)         0.8719  0.8131 0.6335 0.1673
```

5.2.10 M7 - Ranger - (best_predictors + NA replaced + weighted + threshold + cross validation)

Let's now check using cross-validation:
- whether there is any overfitting
- and whether we can achieve even better overall performance

```
model_ranger_cv <- function(studied_df, model_name,
    w3, x_threshold, focus_predictors, k_fold) {
    # Creating Train & Test Set
    set.seed(1)
    studied_df <- studied_df %>%
        mutate(nat_dis = as.factor(nat_dis))
    folds <- createFolds(studied_df$nat_dis, k = k_fold,
        list = TRUE, returnTrain = FALSE)

    # Creating cross validation loop for each
    # fold
    results_fold <- lapply(folds, function(test_index) {
        test_set <- studied_df[test_index, ]
        train_set <- studied_df[-test_index, ]
        missing_cities <- test_set %>%
            filter(!insee_code %in% train_set$insee_code)
        train_set <- train_set %>%
            union(missing_cities)
        test_set <- test_set %>%
            filter(insee_code %in% train_set$insee_code)

        # Manual duplication of minority
        multiple_class_3 <- train_set %>%
            filter(nat_dis == 1)
        multiple_class_3 <- multiple_class_3 %>%
            sample_n(size = round(nrow(multiple_class_3) *
                w3), replace = TRUE)

        train_set_weighted <- bind_rows(train_set,
            multiple_class_3)

        model_train_set <- train_set_weighted %>%
            select(focus_predictors, nat_dis) %>%
            drop_na()

        model_test_set <- test_set %>%
            select(focus_predictors, nat_dis) %>%
            drop_na()

        y_train <- model_train_set %>%
```

```
            pull(nat_dis)
        y_train <- as.factor(y_train)

        y_test <- model_test_set %>%
            pull(nat_dis)
        y_test <- as.factor(y_test)

        x_train <- model_train_set %>%
            select(focus_predictors)

        x_test <- model_test_set %>%
            select(focus_predictors)
        # Ranger Model
        model <- ranger(formula = nat_dis ~ ., data = model_train_set,
            importance = "impurity", probability = TRUE,
            num.trees = 500, num.threads = parallel::detectCores())
        # Creating predictions & metrics
        y_predicted_prob <- predict(model, x_test)
        y_predicted_prob <- y_predicted_prob$predictions

        proba_class1 <- y_predicted_prob[, "1"]
        y_predicted <- ifelse(proba_class1 > x_threshold,
            1, 0)  # Creating the decision threshold
        y_predicted <- factor(y_predicted, levels = levels(y_test))

        model_results <- confusionMatrix(y_predicted,
            y_test, positive = "1")
        accuracy <- sum(y_predicted == y_test)/length(y_test)

        y_test_onehot <- model.matrix(~y_test - 1)
        colnames(y_test_onehot) <- colnames(y_predicted_prob)
        brier_score <- round(mean(rowSums((y_predicted_prob -
            y_test_onehot)^2)), 4)
        # Output of the metrics
        data.frame(row.names = model_name, Accrcy = round(accuracy,
            4), Stivity = round(model_results$byClass["Sensitivity"],
            4), F1 = round(model_results$byClass["F1"],
            4), Brier = brier_score)
    })

    # Calculating the average metrics across all
    # folds
    cv_result <- bind_rows(results_fold) %>%
        summarise(across(everything(), mean, na.rm = TRUE)) %>%
        mutate(across(everything(), round, 4)) %>%
        mutate(across(everything(), as.character))
    rownames(cv_result) <- c(model_name)
    # Returning average metrics
    return(cv_result)
}
new_result <- model_ranger_cv(studied_df = df2_no_nas,
    model_name = "M7_Ranger(M6_cv)", w3 = best_w3,
    x_threshold = best_threshold, focus_predictors = best_predictors,
```

```
    k_fold = 5)  # Running our function
results <- bind_rows(results, new_result)  # Adding new result to historical results
results  # Showing latest results
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                  0.8931  0.5338 0.5762 0.1385
M5_Ranger_(M3)                  0.8785  0.7878 0.6384 0.1673
M6_Ranger_(M5_threshold)        0.8719  0.8131 0.6335 0.1673
M7_Ranger(M6_cv)                 0.874  0.8202  0.639 0.1662
```

We observe that our model is stable, indicating no apparent overfitting.
The model's overall performance even turns out to be slightly better.

5.2.11 M8 - XG BOOST (best_predictors + NA replaced + weighted + threshold)

Let's now try an XGBoost model

```
model_xgboost <- function(studied_df, model_name, w3,
    x_threshold, focus_predictors) {
    # Creating Train & Test Set
    set.seed(1)
    studied_df <- studied_df %>%
        mutate(nat_dis = as.factor(nat_dis))
    test_index <- createDataPartition(studied_df$natural_disaster,
        times = 1, p = 0.2, list = FALSE)
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    # Manual duplication of minority
    multiple_class_3 <- train_set %>%
        filter(nat_dis == 1)
    multiple_class_3 <- multiple_class_3 %>%
        sample_n(size = round(nrow(multiple_class_3) *
            w3), replace = TRUE)

    train_set_weighted <- bind_rows(train_set, multiple_class_3)

    model_train_set <- train_set_weighted %>%
        select(focus_predictors, nat_dis) %>%
        drop_na()
```

```
    model_test_set <- test_set %>%
        select(focus_predictors, nat_dis) %>%
        drop_na()

    y_train <- model_train_set %>%
        pull(nat_dis)
    y_train <- as.numeric(as.character(y_train))

    y_test <- model_test_set %>%
        pull(nat_dis)
    y_test <- as.numeric(as.character(y_test))

    x_train <- model_train_set %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)

    x_train <- sparse.model.matrix(~. - 1, data = x_train)
    x_test <- sparse.model.matrix(~. - 1, data = x_test)

    # XGB MODEL
    xgb_model <- xgboost(x_train, y_train, nrounds = 100,
        objective = "binary:logistic", verbose = 0)
    # Creating predictions & metrics
    y_predicted_prob <- predict(xgb_model, x_test)
    y_predicted <- ifelse(y_predicted_prob > best_threshold,
        1, 0)  # Creating the decision threshold

    xgb_model_results <- confusionMatrix(factor(y_predicted),
        factor(y_test), positive = "1")

    model_results <- confusionMatrix(as.factor(y_predicted),
        as.factor(y_test), positive = "1")
    accuracy <- sum(y_predicted == y_test)/length(y_test)

    y_test_onehot <- model.matrix(~y_test - 1)
    colnames(y_test_onehot) <- colnames(y_predicted_prob)
    brier_score <- round(mean(rowSums((y_predicted_prob -
        y_test_onehot)^2)), 4)

    # Output of the metrics
    new_result <- data.frame(row.names = model_name,
        Accrcy = round(accuracy, 4), Stivity =
        ↪  round(model_results$byClass["Sensitivity"],
            4), F1 = round(model_results$byClass["F1"],
            4), Brier = brier_score)
    new_result <- new_result %>%
        mutate(across(everything(), as.character))
    # Returning 2 elements
    return(list(new_result = new_result, confusion_matrix = model_results))
}
```

```
new_result <- model_xgboost(studied_df = df2_no_nas,
    model_name = "M8_XGB_(M6)", w3 = best_w3, x_threshold = best_threshold,
    focus_predictors = best_predictors)

new_result$new_result
```

```
           Accrcy Stivity     F1 Brier
M8_XGB_(M6) 0.8701  0.8421 0.6383 0.083
```

```
results
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                  0.8931  0.5338 0.5762 0.1385
M5_Ranger_(M3)                  0.8785  0.7878 0.6384 0.1673
M6_Ranger_(M5_threshold)        0.8719  0.8131 0.6335 0.1673
M7_Ranger(M6_cv)                 0.874  0.8202  0.639 0.1662
```

```
results <- bind_rows(results, new_result$new_result)
results
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                  0.8931  0.5338 0.5762 0.1385
M5_Ranger_(M3)                  0.8785  0.7878 0.6384 0.1673
M6_Ranger_(M5_threshold)        0.8719  0.8131 0.6335 0.1673
M7_Ranger(M6_cv)                 0.874  0.8202  0.639 0.1662
M8_XGB_(M6)                     0.8701  0.8421 0.6383  0.083
```

We obtain slightly lower accuracy but we gain significantly in sensitivity and confidence and the Brier score has dropped considerably.

5.2.12 M9 - XG BOOST (best_predictors + NA replaced + weighted + threshold + cross validation)

Let's now run cross-validation on our model to check its stability.

```
model_xgboost_cv <- function(studied_df, model_name,
    w3, x_threshold, focus_predictors, k_fold) {

    # Creating Train Set & Test Set
    set.seed(1)
    studied_df <- studied_df %>%
        mutate(nat_dis = as.factor(nat_dis))
    folds <- createFolds(studied_df$nat_dis, k = k_fold,
```

```
    list = TRUE, returnTrain = FALSE)

# Creating cross validation loop for each
# fold
results_fold <- lapply(folds, function(test_index) {
    test_set <- studied_df[test_index, ]
    train_set <- studied_df[-test_index, ]
    missing_cities <- test_set %>%
        filter(!insee_code %in% train_set$insee_code)
    train_set <- train_set %>%
        union(missing_cities)
    test_set <- test_set %>%
        filter(insee_code %in% train_set$insee_code)

    # Manual duplication of minority
    multiple_class_3 <- train_set %>%
        filter(nat_dis == 1)
    multiple_class_3 <- multiple_class_3 %>%
        sample_n(size = round(nrow(multiple_class_3) *
            w3), replace = TRUE)

    train_set_weighted <- bind_rows(train_set,
        multiple_class_3)

    model_train_set <- train_set_weighted %>%
        select(focus_predictors, nat_dis) %>%
        drop_na()

    model_test_set <- test_set %>%
        select(focus_predictors, nat_dis) %>%
        drop_na()

    y_train <- model_train_set %>%
        pull(nat_dis)
    y_train <- as.numeric(as.character(y_train))

    y_test <- model_test_set %>%
        pull(nat_dis)
    y_test <- as.numeric(as.character(y_test))

    x_train <- model_train_set %>%
        select(focus_predictors)

    x_test <- model_test_set %>%
        select(focus_predictors)

    x_train <- sparse.model.matrix(~. - 1, data = x_train)
    x_test <- sparse.model.matrix(~. - 1, data = x_test)

    # XGB MODEL
    xgb_model <- xgboost(x_train, y_train, nrounds = 100,
        objective = "binary:logistic", verbose = 0)
```

```
        # Creating predictions & metrics
        y_predicted_prob <- predict(xgb_model, x_test)
        y_predicted <- ifelse(y_predicted_prob > best_threshold,
            1, 0)  # Creating the decision threshold

        xgb_model_results <- confusionMatrix(factor(y_predicted),
            factor(y_test), positive = "1")

        model_results <- confusionMatrix(as.factor(y_predicted),
            as.factor(y_test), positive = "1")
        accuracy <- sum(y_predicted == y_test)/length(y_test)

        y_test_onehot <- model.matrix(~y_test - 1)
        colnames(y_test_onehot) <- colnames(y_predicted_prob)
        brier_score <- round(mean(rowSums((y_predicted_prob -
            y_test_onehot)^2)), 4)

        # Output of the metrics
        data.frame(row.names = model_name, Accrcy = round(accuracy,
            4), Stivity = round(model_results$byClass["Sensitivity"],
            4), F1 = round(model_results$byClass["F1"],
            4), Brier = brier_score)
    })
    # Calculating the average metrics across all
    # folds
    cv_result <- bind_rows(results_fold) %>%
        summarise(across(everything(), mean, na.rm = TRUE)) %>%
        mutate(across(everything(), round, 4)) %>%
        mutate(across(everything(), as.character))
    rownames(cv_result) <- c(model_name)
    # Returning the average metrics
    return(cv_result)
}
```

Running the function/

```
new_result <- model_xgboost_cv(studied_df = df2_no_nas,
    model_name = "M9_XGB_(M6_cv)", w3 = best_w3, x_threshold = best_threshold,
    focus_predictors = best_predictors, k_fold = 5)  # Running our function

results <- bind_rows(results, new_result)  # Adding new result to historical results
results  # Showing the latest result
```

```
                                Accrcy Stivity     F1  Brier
Target                          > 0.84  > 0.79 > 0.59 < 0.11
naive_model                     0.7653  0.1394 0.1378     NA
M1_RF_(best_prms)               0.8869  0.5061 0.5576 0.1746
M2_RF_(M1_NAs_replcd_by_avg_dep) 0.8936  0.5096  0.566 0.1548
M3_RF_(M2_weighted)             0.8768  0.7898 0.6358 0.2017
M4_Ranger_(M2)                  0.8931  0.5338 0.5762 0.1385
M5_Ranger_(M3)                  0.8785  0.7878 0.6384 0.1673
M6_Ranger_(M5_threshold)        0.8719  0.8131 0.6335 0.1673
M7_Ranger(M6_cv)                 0.874  0.8202  0.639 0.1662
M8_XGB_(M6)                     0.8701  0.8421 0.6383  0.083
```

```
M9_XGB_(M6_cv)                    0.8701    0.84 0.6375 0.0825
```

The results show that the model is robust and there is no apparent overfitting.

6 CONCLUSION

6.1 Context

Professional certificate:
As a reminder, this project is part of a global "Professional Certificate Program in Data Science" from Harvard University on Edx platform
("https://pll.harvard.edu/series/professional-certificate-data-science")
The purpose of this project was to autonomously put into practice all the knowledge learned during the past courses.

Project Selection:
Here is the list of criteria that led me to choose this topic for my project:
- The accessibility of data from public databases
- The accessibility of the subject itself (many French citizens are affected by clay soil movement)
- The ease of communicating the topic to a wider audience (a concrete issue that impacts our immediate environment)
- A problem complex enough to apply the knowledge gained during the training, while also requiring the learning of new tools and working methods.

Hardware and Tools:
- IDE: RStudio
- Language: R (version 4.4.1 - 2024-06-14)
- Operating System: macOS Sonoma 14.4
- Computer: MacBook Pro – Apple M3 Pro – 18 GB RAM

6.2 Results

Databases:
For this study, we first downloaded numerous public datasets:
- History of requests and acknowledgement of "natural disasters" per ciy, linked to clay movements.
- Cities exposure to climate risks in 2016
- Map of exposure to the phenomenon of shrinkage-swelling of clays covers metropolitan France (excluding the city of Paris)"
- GPS positions of all French cities
- Monthly basic climatological data by department (from 1950 to 2023)
- Population size per city per year
- History of droughts by municipality

Data Analysis & Engineering:
Then, through data analysis, we were able to identify the most relevant predictors.
Our data engineering process allowed us to create new, meaningful features.
One-third of the features used in our final model resulted from our data engineering efforts:

trend_cumulative_precipitation, trend_winter_summer, and spread_summer_winter.

Multiclass prediction:
We initially focused on the target variable “natural_disaster”, a 3-level factor: 1 - “No Natural Disaster”, 2 - “Requested”, 3 - “Acknowledged”.
We developed the following models: randomForest and ranger.
We achieved good results for predicting classes 1 and 3 after applying optimal class weighting, without significantly harming the prediction of class 1.
However, we were not able to improve predictions for class 2 without negatively impacting class 1.
Here are the results of our last model: - Accuraccy: 0.79
- Sensitivity Class 2: 0.21
- Sensitivity Class 3: 0.8
- F1 Score Class 2: 0.3
- F1 Score Class 3: 0.63
- Brier score: 0.29

This suggests that there may be a loss of information in our dataset.
We may be missing variables that capture complex relationships within the data.
As a result, we decided to continue the study focusing exclusively on classes 1 and 3.

Binary prediction:
So, in a second phase, we focused on the target variable “nat_dis”, a binary factor with 2 levels: 0 - “No Natural Disaster”, 1 - “Acknowledged”
We developed the following models: randomForest, ranger, and XGBoost.
We optimized the following parameters: class 1 weighting and the ideal decision probability threshold.
After cross-validation, we achieved balanced, stable, and accurate results:
- Accuracy: 0.8688
- Sensitivity: 0.8375
- F1 score: 0.635
- Brier score: 0.0834

6.3 Improvement

Several improvement paths can be considered to optimize our model’s performance:
Exclusion of cities with no natural disasters (nat_dis_group = 0)
For simplicity, we excluded them from the modeling process.
This arbitrary choice may hide true risk exposure due to a lack of historical claims.
It would be interesting to create a dedicated prediction model for these cities, based on those that had no natural disasters for many years but were affected in recent years.

We have only used one method to handle missing data:/ So far, we used simplistic imputation replacing missing values with the average by department.
To enhance robustness, more advanced imputation (for exemple: KNN imputation, multiple imputation, or predictive imputation) could improve accuracy.

Key data is missing:
The model does not integrate soil composition data, which is crucial for predicting shrink-swell risks.
Including such external geotechnical datasets could significantly refine predictions.

More data analysis & engineering:
To move quickly, we focused only on a few features most correlated with our target variable.
However, we are likely missing complex relationships between features and with the target itself.
A more in-depth data analysis and additional feature engineering would likely improve performance by capturing complementary information.
Unsupervised learning methods (e.g., neural networks) or techniques like Singular Value Decomposition (SVD) could help uncover complex relationships between features.

6.4 Next Steps

The objective of this project was to provide a working foundation for the various stakeholders involved in this issue.
This is why the indicators and performance metrics were selected to prioritize balanced results.
Moving forward, each stakeholder will be able to adapt this project to their specific business needs.

Here are a few examples of business-related concerns for the stakeholders involved:
- Insurers: Anticipate the amount of funds to be released and assess risk levels.
- Individuals: Anticipate and initiate insurance and repair procedures only when truly necessary.
- Municipalities/Prefectures: Anticipate the handling of large volumes of disaster declaration requests.

And here are a few ideas for how indicators and model development could be tailored:
- Focus model development on the ability to predict in advance (i.e., using the least amount of data from the target year).
For example, instead of engineering features over the full 12 months of the target year, features could be built using the last 6 months of the previous year and the first 6 months of the current year.
- A risk calculation formula for insurers could be developed:
(theoretical overpricing = false positives $\times$ cost) - (theoretical underpricing = false negatives $\times$ cost)

7 Sources and citations

Ministry of Ecological Transition
“Mapping of individual houses during shrinkage-swelling of clays” (June 2021)
https://www.statistiques.developpement-durable.gouv.fr/media/4551/download?inline

General Commission For Sustainable Development
“The vulnerability of municipalities to climate risks method note for the calculation and typological classification” (january 2020)
https://www.statistiques.developpement-durable.gouv.fr/media/3509/download?inline

General Commission For Sustainable Development
“Clay shrinkage and swelling: more than 4 million houses potentially highly exposed” (otcober 2017)
https://www.statistiques.developpement-durable.gouv.fr/media/612/download?inline

Central Reinsurance Fund (Caisse Centrale de Réassurance - CCR)
“Consequences Of Climate Change On The Cost Of Natural Disasters In France By 2050” (september 2023)
https://www.ccr.fr/documents/35794/1255983/CCR+Etude+climat+BAG+23102023+page+22mo.pdf/6

8b95f6e-8238-4dcc-6c56-025fa410257b?t=1698161402128

Ministry of Ecological Transition
"The vulnerability of municipalities to clay shrink-swell hazard: calculation method and typological classification"
https://www.statistiques.developpement-durable.gouv.fr/media/613/download?inline