HarvardX PH125.9x - Data Science: Capstone_CYO

Machine Learning - Case study: GEOS-Chem Dataset

Paolo Sebastianelli

2021-07-28

Contents

Preface

This report is part of the requirements for completing the final course in HarvardX's multi-part **Data Science Professional Certificate** series.

Introduction

The dataset used (Fisher, Sebastianelli, and Schofield (2021))[1] to develop this report can be found at Research Data Australia which is the data discovery service of the Australian Research Data Commons (ARDC).

Specifically, the data set is stored at this link.

Inspired by Keller and Evans article (Keller and Evans (2019)), the **goal** of this project is to start the journey in the use of machine learning (hereafter ML) for the study of gas-phase chemistry in atmospheric Chemistry Transport Models (hereafter CTM).

The CTM used to generate the dataset is GEOS-Chem 12.8.1 (Version 12.8.1) (Community (2020)).

This is the official description of GEOS-Chem that can be found at this link:

> *"GEOS-Chem enables simulations of atmospheric composition on local to global scales. It can be used off-line as a 3-D chemical transport model driven by assimilated meteorological observations from the Goddard Earth Observing System (GEOS) of the NASA Global Modeling Assimilation Office (GMAO) . It can also be used on-line as a chemical module coupled to weather and climate models. GEOS-Chem is developed and used by hundreds of research groups worldwide as a versatile tool for application to a wide range of atmospheric composition problems. It is open-access and can be downloaded through github. It is also fully supported for use on the Amazon Web Services cloud."*

The GEOS-Chem Community MISSION can be found in the GEOS-Chem Wiki:

> *"Advance understanding of human and natural influences on the environment through a comprehensive, state-of-the-science, readily accessible global model of atmospheric"* composition.

Data concerning 1 month GEOS-Chem simulation (March 2018) were stored in netCDF files. To process this kind of data the *tidync*, *ncdf4* and other relevant packages have been used. In fact, these kind of files are really big, some of them occupy more than 10Gb. In order to avoid long period of downloading from the official webpage a dataframe has been stored in a .txt file downloadable here, in my GitHub repository. A R script that does take this .txt and returns all the analysis showed in this report can be found in the same repository inside the folder "CAPSTONE_CYO/GEOS_Chem."

Instead of use the netCDF files, a R script user can manipulate the .txt file where the same information is gathered.

The geographic area investigated is a rectangular regional zone characterized by the following latitude and longitude intervals:

$$-89.5° < lat < 30.0°$$

and

$$-180° < lon < 177.5°$$

.

The area is divided in grid cells and each grid cell constitutes one training sample, consisting of 234 variables (the features): transported species concentrations and meteorological variables.

In this project only the predictions of sulfur dioxide $[SO_2]$ concentrations in each grid cell[2] have been calculated using the other 233 predictors. Furthermore, the study has been restricted to the world surface level and the first day of simulation with the future aim of expanding the investigation at all the month of simulation.

[1]This data set conforms to the CCBY Attribution License (http://creativecommons.org/licenses/by/4.0/). Please follow instructions listed in the citation reference provided at http://data.aad.gov.au/aadc/metadata/citation.cfm?entry_id=AAS_4431_CAMMPCAN_GEOS_Chem_Model_AA_2017-18 when using these data.Please also contact Jenny Fisher at the University of Wollongong before using these data. Jennyf at uow dot edu dot au

[2](they are 144*31 = 4464 grid cells, or observations)

The 234 variables are the Dry mixing ratio of the chemical species (units: mol*mol-1 dry air) and the metereological variables included in the GEOS-Chem simulation. The dataset has been analysed numerically and visually and some features has been dropped according the zero variance criteria.

The total dataset has been partitioned in two subsets for training and testing the ML algorithms that have been selected.

The performance of a ML algorithm has been estimated by the calculation of Normalized Root Mean Square Error (hereafter NRMSE), which gives an idea of how far the predictions are from the "true" concentrations (i.e. from the GEOS-Chem chemical concentrations).

The numerical solution of mathematical equations that model the Atmospheric Chemistry are computationally expensive. The idea here is to start a path toward the development of a ML-driven CTM model where the numerical integration steps could be substituted by a ML algorithm, if necessary.

Environment, vegetation and human health suffer the impact of excessive anthropogenic chemical species or induced concentration imbalances in the natural processes. CTM are central tools to study these aspects. For this reason, in this context, a faster implementation of CTM simulations has an enormous chain-effect on the analysis processes.

Data Analysis

The result of a GEOS-Chem simulation is a .nc4 file. Let's describe what kind of files they are.

netCDF Files

NetCDF means Network Common Data Form (NetCDF).

From the official webpage:

> Data in netCDF format is:
>
> -Self-Describing. A netCDF file includes information about the data it contains.
>
> -Portable. A netCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.
>
> -Scalable. Small subsets of large datasets in various formats may be accessed
> efficiently through netCDF interfaces, even from remote servers.
>
> -Appendable. Data may be appended to a properly structured netCDF file without copying the dataset or redefining its structure.
>
> -Sharable. One writer and multiple readers may simultaneously access the same netCDF file.
>
> -Archivable. Access to all earlier forms of netCDF data will be supported by current and future versions of the software.

Visit this site to understand more about this file format for storing array-based data originally developed for the Earth Science field.

The scientific data are thought as a set of related arrays. For some point located at a certain latitude, longitude, vertical level, and time a number of physical properties and info are stored in this multy-array arrangment.

Data manipulation

Unfortunately, the size of the .nc4 files is enormous (i.e. more than 5Gb) and a download at the beginning of a R script could take too much time. For this reason they have been manipulated and a dataframe containing all the required info has been generated as a .txt file.

Here the code that should be used to obtain the .txt:

```r
##### REQUIRED PACKAGES

#### These are the packages and libraries I need to explore and manipulate
#### the data set

if(!require(tidyverse))install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(ncdf4))install.packages("ncdf4")("ncdf4", repos = "http://cran.us.r-project.org")
if(!require(raster))install.packages("raster")("raster", repos = "http://cran.us.r-project.org")
if(!require(rgdal))install.packages("rgdal")("rgdal", repos = "http://cran.us.r-project.org")
if(!require(ggplot2))install.packages("ggplot2")("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(RNetCDF))install.packages("RNetCDF")("RNetCDF", repos = "http://cran.us.r-project.org")
if(!require(tidync))install.packages("tidync")("tidync", repos = "http://cran.us.r-project.org")
if(!require(maps))install.packages("maps")("maps", repos = "http://cran.us.r-project.org")
if(!require(devtools))install.packages("devtools")("devtools", repos = "http://cran.us.r-project.org")
if(!require(stars))install.packages("stars")("stars", repos = "http://cran.us.r-project.org")
if(!require(dplyr))install.packages("dplyr")("dplyr", repos = "http://cran.us.r-project.org")
if(!require(tidyr))install.packages("tidyr")("tidyr", repos = "http://cran.us.r-project.org")
if(!require(readr))install.packages("readr")("readr", repos = "http://cran.us.r-project.org")
if(!require(RCurl))install.packages("RCurl")("RCurl", repos = "http://cran.us.r-project.org")
if(!require(rgeos))install.packages("rgeos")("rgeos", repos = "http://cran.us.r-project.org")
if(!require(glmnet))install.packages("glmnet")("glmnet", repos = "http://cran.us.r-project.org")
if(!require(ranger))install.packages("ranger")("ranger", repos = "http://cran.us.r-project.org")
if(!require(randomForest))install.packages("randomForest")("randomForest", repos = "http://cran.us.r-pr
if(!require(kernlab))install.packages("kernlab")("kernlab", repos = "http://cran.us.r-project.org")
if(!require(e1071))install.packages("e1071")("e1071", repos = "http://cran.us.r-project.org")
if(!require(DataExplorer))install.packages("DataExplorer")("DataExplorer", repos = "http://cran.us.r-pr


library(ncdf4) # package for netcdf manipulation
library(raster) # package for raster manipulation
# library(rgdal) # package for geospatial analysis
library(ggplot2) # package for plotting
library(RNetCDF)
library(tidync)
library(fields)
library(maps)
library(dplyr)
library(ncmeta)
library(viridis)
library('rnaturalearth')
library("rnaturalearthdata")
# library("rnaturalearthhires")
library(tidyverse)
library(dslabs)
library(caret)
library(randomForest)
library(rpart)
library(matrixStats)
library(corrplot)
library(ggcorrplot)
library(Rborist)
library("plot3D")
library(ranger)
library(xgboost)
library(plyr)
library(readr)
library(RCurl)
library(rgeos)
library(glmnet)
library(kernlab)
library(e1071)
library(DataExplorer)
```

The following part of the used R code is shown only for illustrative purposes. It is in the R script accompanying this report but it is commented, since the txt version is used to perform the data analysis.

```
# ####################################### PART ONE ###########################
# ### All this "Part ONE" is about the nc4 file opening procedure and the   ##
# ### extraction of data from the netCDF file with the aim of building a    ##
# ### new dataframe, if the the nc4 files are on the local system.
#####
##### NOTE: Since the nc4 files are really big (more than 5Gb) in some
#####       case I processed them to obtain a csv that can be read directly
#####       from my github repository.
#####
#####       Link: https://raw.githubusercontent.com/PaoloSebas/
#####                    DATA_SCIENCE/main/GC_tot_Dat3.txt
###
# ############################################################################
#
# ### OPENING DATA FILE using **nc_open** function from [ncdf4] library
# ### This step is done to extract the values of latitude and longitude
# ### and storing them in th object "Data_ncdf4_type"
#
# Data_ncdf4_type <- nc_open('GEOSChem.SpeciesConc.20180301_regional.nc4')
# class(Data_ncdf4_type)  ### "ncdf4"
#
# #### Extracting the values for longitude and latitude using ncvar_get function
# #### from ncdf4 package. Storing them in two arrays
#
# lon <- ncvar_get(Data_ncdf4_type, "lon")  ### between -180 and 177.5 degrees
# lat <- ncvar_get(Data_ncdf4_type, "lat")  ### between -89.5 and -30.0  degrees
#
# ##############################################################################
############
# ## OPTIONAL #################################################################
############
# ###### If one wants it is possible subsetting the longitude and latitude values
# ###### using indexes., In this example a piece of the world map is taken between
# ###### Australia and Antartic
# #
# # lonIdx <- which(Data_ncdf4_type$dim$lon$vals > 100 &
# #                 Data_ncdf4_type$dim$lon$vals < 170)
# # latIdx  <- which(Data_ncdf4_type$dim$lat$vals > -90 &
# #                 Data_ncdf4_type$dim$lat$vals < 0)
# #
# # lon <- ncvar_get(Data_ncdf4_type,
# #                  "lon", start=c(min(lonIdx)),
# #                   count=c((max(lonIdx)-min(lonIdx)+1)))
# # nlon <- dim(lon)
# # nlon # 27
# #
# # lat <- ncvar_get(Data_ncdf4_type, "lat",
# #                   verbose = F, start=c(min(latIdx)),
# #                   count=c((max(latIdx)-min(latIdx)+1)))
# # nlat <- dim(lat)
# # nlat # 45
# ##############################################################################
#
# ### Tidync package is a better choice to extract the data one wants
# ### Opening data file using functions from tidync library
#
# ### Source file for CHEMICAL SPECIES CONCENTRATION values
#
# source <- tidync('GEOSChem.SpeciesConc.20180301_regional.nc4')
# class(source)
#
# ### METEREOLOGICAL STATE description
#
# state_met <- tidync('GEOSChem.StateMet.20180301_regional.nc4')
#
# ################################
# ###    SOURCE - DESCRIPTION ####
# ################################
#
# class(source)  ### "tidync"
#
# print(source2)
#
# ### There are 5 dimension, 4 of them are active:
# ### (D0) time, (D1) lon, (D2) lat, (D3)lev (level)
# ### The time spans 31 days of simulation
# ### There are 47 levels to be analyzed, starting from the world surface
# ### 31 latitudes points from -89.5 to 30 degrees
# ### 144 longitudes points from -180 to 177.5
# ### 203 variables
# ### 4464 observations (144*31)
#
# ####################################
# ###    STATE MET  - DESCRIPTION ####
# ####################################
#
# print(state_met)
#
# ### There are 5 dimension, 4 of them are active:
# ### (D0) time, (D1) lon, (D2) lat, (D3)lev (level)
# ### The time spans 31 days of simulation
# ### There are 47 levels to be analyzed, starting from the world surface
# ### 31 latitudes points from -89.5 to 30 degrees
# ### 144 longitudes points from -180 to 177.5
# ### 36 variables
# ### 4464 observations (144*31)
#
# ### Activating, again, the dimensions that are of my interest (just in case)
#
# source <- tidync('GEOSChem.SpeciesConc.20180301_regional.nc4') %>%
#                  activate("D1,D2,D3,D0")
# state_met <- tidync('GEOSChem.StateMet.20180301_regional.nc4') %>%
#                     activate("D1,D2,D3,D0")
#
# ############################################################################
# #############                                          #####################
# #############           SLICING DATA                   #####################
# ############################################################################
#
# ### The data set is sliced to obtain information
# ### about the "surface" level (lev = 0.992) during the "first" (time = 720)
# ### day of simulation
#
# source_slice <- source %>%
#                 hyper_filter(time = time == 7.2e+2,
#                              lev = lev > 0.980,
#                              lat =  dplyr::between(index,1, 31),
#                              lon =  dplyr::between(index, 1, 144))
# state_met_slice <- state_met %>%
#                   hyper_filter(time = time == 7.2e+2,
#                                lev = lev > 0.980,
#                               lat =  dplyr::between(index, 1, 31),
#                                lon =  dplyr::between(index, 1, 144))
#
# ### The data set is sliced to obtain information
# ### about the "surface" level (lev = 0.992) during the "second" (time = 2160)
# ### day of simulation
#
# source_slice2 <- source %>%
#                   hyper_filter(time = time == 2160  ,
#                   lev = lev > 0.980,
#                   lat =  dplyr::between(index, 1, 31),
#                   lon =  dplyr::between(index, 1, 144))

# state_met_slice2 <- state_met %>%
#                     hyper_filter(time = time == 2160,
#                     lev = lev > 0.980,
#                     lat =  dplyr::between(index, 1, 31),
#                     lon =  dplyr::between(index, 1, 144))
#
# ### If you want, you can check the time values with this commented cod3
# ##  Data_ncdf4_type$dim$time
#
# #### In case of subsetting one can change lat and lon
# # source_slice <- source %>%
#                 hyper_filter(time = time == 7.2e+2,
#                  lev = lev > 0.980,
#                  lat =  dplyr::between(index, 1, 45),
#                  lon =  dplyr::between(index, 114, 140))
#
# ### If you want see the difference between source and source_slice:
# # print(source_slice)
# # print(state_met_slice)
#
# #### CREATING DATA FRAMES FROM TIDYNC OBJECTS
# #### using hyper_tibble() function from the package tidync
#
# src_slc_dataframe <- source_slice %>% hyper_tibble()
# stm_slc_dataframe <- state_met_slice %>% hyper_tibble()
#
# # src_slc_dataframe2 <- source_slice2 %>% hyper_tibble()
# # stm_slc_dataframe2 <- state_met_slice2 %>% hyper_tibble()
#
# ### Binding the data frames.
# ### They share the same lat, lon, time and lev values.
#
# total_dataframe <- cbind(src_slc_dataframe,
#                          stm_slc_dataframe, deparse.level = 1)
# total_dataframe2 <- cbind(src_slc_dataframe2,
#                          stm_slc_dataframe2, deparse.level = 1)
# #
# #### If you want transform the tidync object in an array you can use
#     hyper_array() function
# #### src_slc_array <- source_slice %>% hyper_array()
#
# ###Check if there are duplicated columns
#
# duplicated(t(total_dataframe)) #lon, lat and lev are a duplicated column
#
# ## Eliminating duplicated columns
# total_dataframe <- total_dataframe[!duplicated(as.list(total_dataframe))]
###
### Eliminating lon,lat,lev duplicated columns
#
# # total_dataframe2 <- total_dataframe2[!duplicated(as.list(total_dataframe2))]
### Eliminating lon,lat,lev duplicated columns
#
# #### Are they columns with NA values?
#
# names(which(sapply(total_dataframe, anyNA)))  #### None
#
# write.table(total_dataframe,"GC_tot_Dat3.txt",sep="\t",row.names=FALSE)
```

Since some duplicated columns have been found, a data cleaning procedure has been implemented, as can be noticed in the previous chunk code.

Once generated the .txt file containing the dataframe info, the following analysis has been performed.

Analysis

Zero variance

First question: Are they columns with 0 variance?

```
##### Loading data from the txt that is in my github page
#####

url <- "https://raw.githubusercontent.com/PaoloSebas/DATA_SCIENCE/main/GC_tot_Dat3.txt"
dest_file <- "GC_tot_Dataframe.txt"
download.file(url, destfile = dest_file)
GC_total_dataframe <- read.table(header = TRUE, dest_file)

### Making an introductory plot
DataExplorer::introduce(GC_total_dataframe)   #### no missing values
```

```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1 4464     234                0                234                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0          4464            1044576      8355368
```

```
#### Are they columns with 0 variance?

nzv <- nearZeroVar(GC_total_dataframe)
col_index <- setdiff(1:ncol(GC_total_dataframe), nzv)
print(nzv)
```

```
## [1] 203 204 225
```

Answer: yes. They are the column 203, 204 and 225.

```
#### Finding the names of the columns that nearzerovar proposes to drop

colnames(GC_total_dataframe)[nzv]
```

```
## [1] "time"             "FracOfTimeInTrop" "Met_DTRAIN"
```

```
#### Filtering the total_dataframe adding
#### another two species "SpeciesConc_H1301", "SpeciesConc_CFC114"

drops <- c("lev","time", "FracOfTimeInTrop", "Met_DTRAIN","SpeciesConc_H1301", "SpeciesConc_CFC114" )
total_dataframe_filtered <- GC_total_dataframe[ , !(names(GC_total_dataframe) %in% drops)]
```

SO2 study

According to Seinfeld and co. (Seinfeld and Pandis (2016)) sulfur dioxide is the predominant anthropogenic sulfur-containing air pollutant. If you want to know more about SO2, read the chapter 6 of the Seinfeld and co. book (Seinfeld and Pandis (2016)).

If the $[SO_2]$ concentration per each grid is stored in a vector and the mean value is calculated, the result is this:

```
### Storing in a vector the SO2 concentration values
### from the total_dataframe_filtered

SO2_total <- total_dataframe_filtered$SpeciesConc_SO2

### Calculating the mean value at surface level
```

```
SO2_mean <- mean(SO2_total)
SO2_mean
```

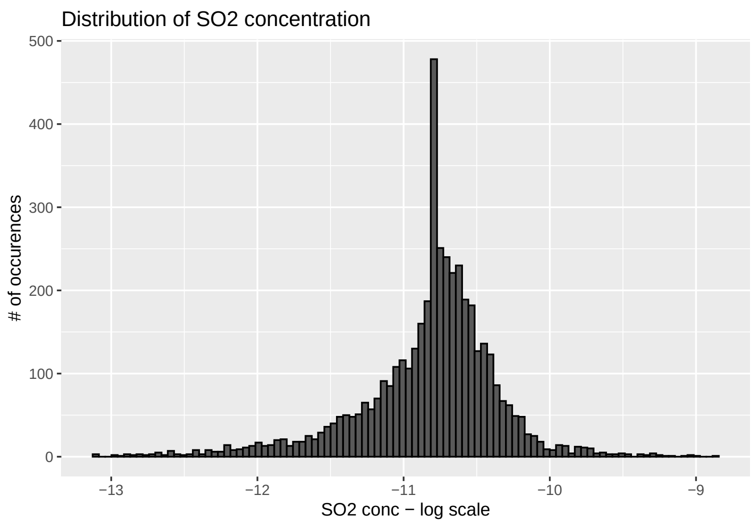
```
## [1] 2.577628e-11
```

```
### The mean Dry mixing ratio of species SO2
### is 2.577628e-11 mol*mol-1 dry air or 25 ppb (parts per billion)
```

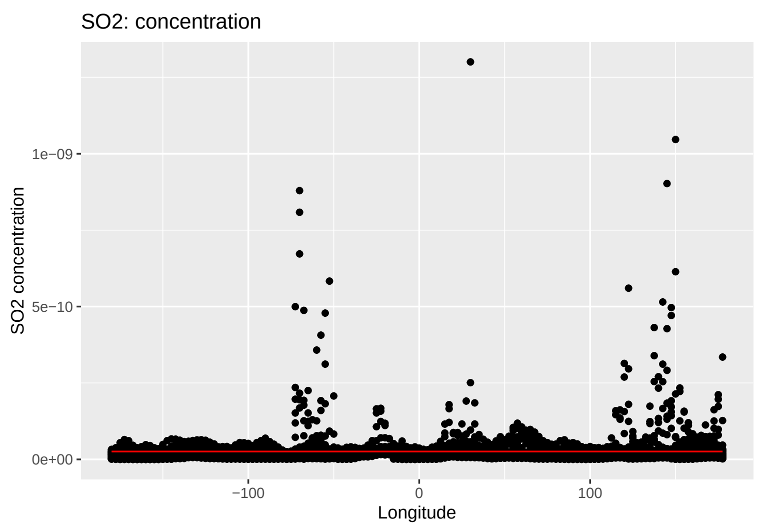
We are interested in the spatial distribution of SO_2 on the surface level.

Firstly, the $[SO_2]$ is depicted in logarithmic scale:

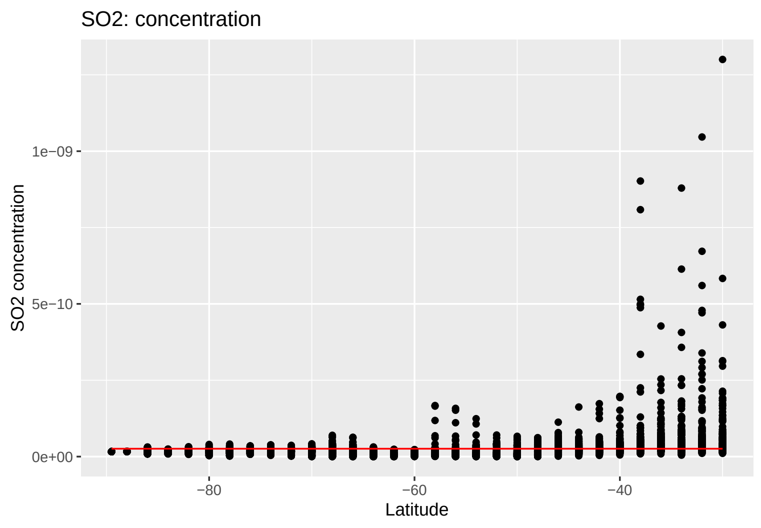
```
### Visualizing the distribution of SO2 values.
### Effectively what is seen is that the distribution
### is centered near the values x*e-11

qplot(log10(SO2_total), bins = 100, color = I("black"),
      main="Distribution of SO2 concentration",
      xlab="SO2 conc - log scale", ylab="# of occurences")
```



```
### According the longitude

total_dataframe_filtered %>% ggplot(aes(x=lon, y = SpeciesConc_SO2)) +
  geom_point() +
  ggtitle("SO2: concentration") +
  labs(x="Longitude", y = "SO2 concentration") +
  geom_line(aes(lon, SO2_mean), col="red")
```

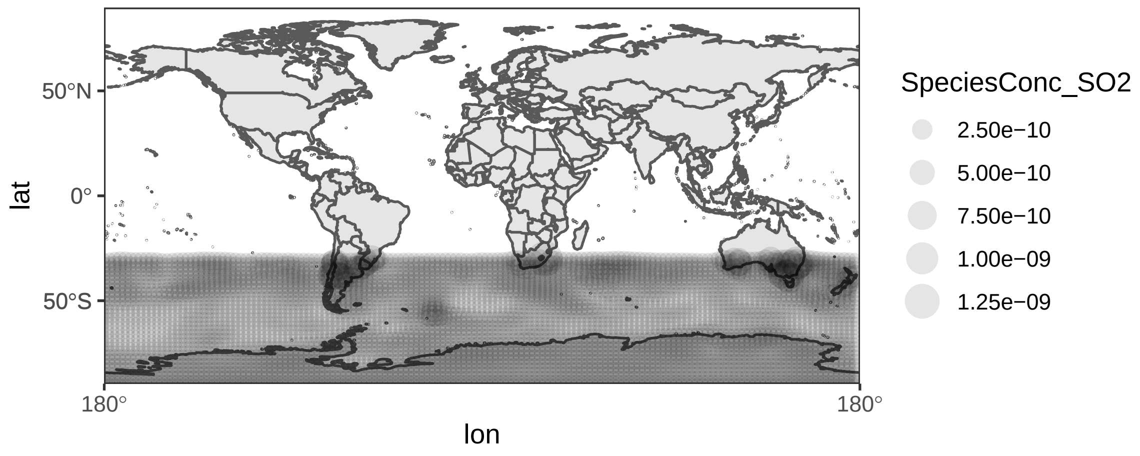


```r
### According the latitude

total_dataframe_filtered %>% ggplot(aes(x=lat, y = SpeciesConc_SO2)) +
  geom_point() +
  ggtitle("SO2: concentration") +
  labs(x="Latitude", y = "SO2 concentration") +
  geom_line(aes(lat, SO2_mean), col="red")
```



To understand better how SO_2 is distributed on the selected geographical area a map is depicted:

```r
world <- ne_countries(scale = "medium", returnclass = "sf")

ggplot(data = world) +
  geom_sf() +
  geom_point(data = total_dataframe_filtered, aes(x = lon, y = lat, size = SpeciesConc_SO2), alpha = 1/
  coord_sf(xlim = c(-180, 180), ylim = c(-89.5,89.5), expand = FALSE)+
  theme_bw()
```



It can be noticed that there are different zones showing concentration values higher than the mean value.

Which one is the location with maximum value of $[SO_2]$?

```
#### Where is the maximum [SO2] concentration?

### The maximum value of [SO2] is at -30° latitude and 30° longitude

which.max(total_dataframe_filtered$SpeciesConc_SO2)  ### row 4405
```

```
## [1] 4405
```

```
total_dataframe_filtered$lat[4405] # -30
```

```
## [1] -30
```

```
total_dataframe_filtered$lon[4405] # 30
```

```
## [1] 30
```

```
total_dataframe_filtered$SpeciesConc_SO2[4405]
```

```
## [1] 1.300319e-09
```

According to this GEOS-Chem simulation, during the first day of March 208 the point with the highest sulfur dioxide concentration was in Ingwe Local Municipality, South Africa.

Correlation

Can be gained some insight about the correlation between variables? Specifically, what kind of correlation exists between SO_2 and the other predictors?

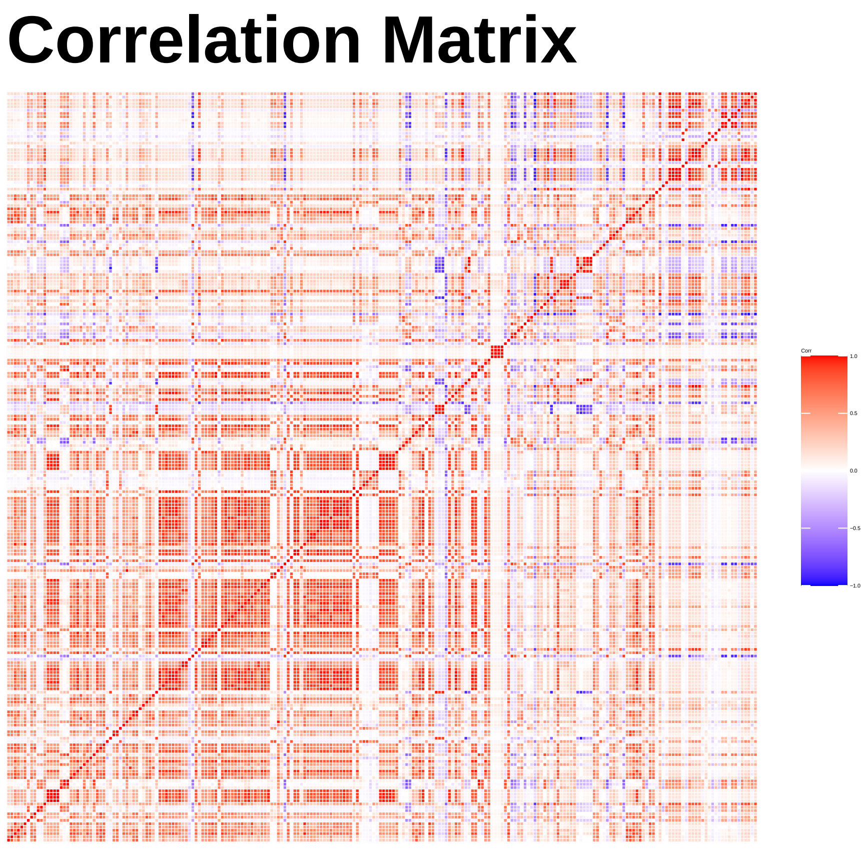
In order to answer this question the follow code chunks have been applied:

```
##############################################################
######
######    CORRELATION BETWEEN VARIABLES
######
##############################################################

sum(is.na(total_dataframe_filtered))  ### no NA value in total_dataframe
```

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

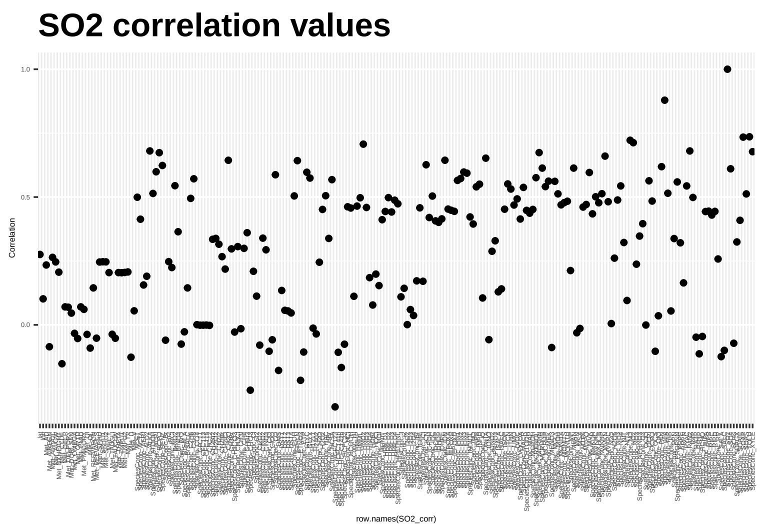
```
### Determining correlation between variables in train_set
corr <- cor(total_dataframe_filtered)
```

```r
### Plotting the correlation matrix
### You have to export the image at high resolution if you want
### appreciate the details.
### The matrix is as png in my github page
### link:

ggcorrplot(corr,  outline.col = "white") +
  theme(text = element_text(size = 2),
        axis.text.x = element_blank(),
        axis.text.y = element_blank()) +
  ggtitle("Correlation Matrix")+
  theme(plot.title = element_text(size = 25, face = "bold"))
```



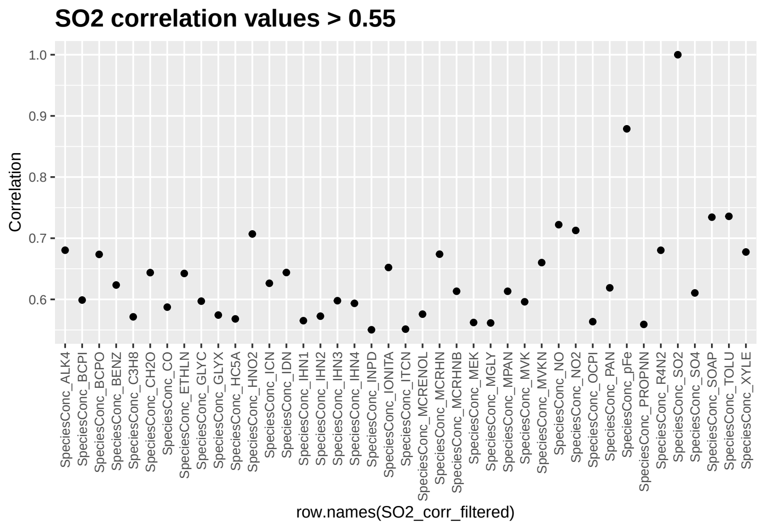
Zooming in the correlation matrix SO_2 row:

```r
### Since we are interested how SO2 is correlated with the others
### Storing of the data for SO2

SO2_corr <- as.data.frame(corr['SpeciesConc_SO2',])
colnames(SO2_corr) <- c("Correlation")

####Plotting the result

ggplot(SO2_corr, aes(x = row.names(SO2_corr), y = Correlation)) +
  geom_point() +
  theme(text = element_text(size = 5),axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  ggtitle("SO2 correlation values")+
  theme(plot.title = element_text(size = 20, face = "bold"))
```



and filtering the result with the condition “Correlation > 0.55”:

```
#### Filtering with correlation > 0.55
SO2_corr_filtered <- SO2_corr %>% filter(Correlation > 0.55)
ggplot(SO2_corr_filtered, aes(x = row.names(SO2_corr_filtered), y = Correlation)) +
  geom_point() +
  theme(text = element_text(size = 10),axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ggtitle("SO2 correlation values > 0.55")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



```
### Storing the variables for future studies
rownames_SO2 <- SO2_corr_filtered %>% row.names()
rownames_SO2
```

```
##  [1] "SpeciesConc_XYLE"    "SpeciesConc_TOLU"    "SpeciesConc_SOAP"
##  [4] "SpeciesConc_SO4"     "SpeciesConc_SO2"     "SpeciesConc_R4N2"
##  [7] "SpeciesConc_PROPNN"  "SpeciesConc_pFe"     "SpeciesConc_PAN"
## [10] "SpeciesConc_OCPI"    "SpeciesConc_NO2"     "SpeciesConc_NO"
## [13] "SpeciesConc_MVKN"    "SpeciesConc_MVK"     "SpeciesConc_MPAN"
## [16] "SpeciesConc_MGLY"    "SpeciesConc_MEK"     "SpeciesConc_MCRHNB"
## [19] "SpeciesConc_MCRHN"   "SpeciesConc_MCRENOL" "SpeciesConc_ITCN"
## [22] "SpeciesConc_IONITA"  "SpeciesConc_INPD"    "SpeciesConc_IHN4"
## [25] "SpeciesConc_IHN3"    "SpeciesConc_IHN2"    "SpeciesConc_IHN1"
## [28] "SpeciesConc_IDN"     "SpeciesConc_ICN"     "SpeciesConc_HNO2"
## [31] "SpeciesConc_HC5A"    "SpeciesConc_GLYX"    "SpeciesConc_GLYC"
## [34] "SpeciesConc_ETHLN"   "SpeciesConc_CO"      "SpeciesConc_CH2O"
## [37] "SpeciesConc_C3H8"    "SpeciesConc_BENZ"    "SpeciesConc_BCPO"
## [40] "SpeciesConc_BCPI"    "SpeciesConc_ALK4"
```

it is possible to detect how the species "pFe" (Anthropogenic iron) is highly correlated to SO_2.

ML Models Implementation

Loss Function

To compare the various models with each other a Loss Function based on the Normalized Root Mean Square Error (NRMSE) has been adopted:

$$\text{NRMSE} = (\frac{\sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}}{\sigma(y)}) * 100$$

Where $\sigma(y)$ is the standard deviation calculated, for $[SO2_2]$ on the *train_set*.

This metric has been chosen because gives an idea of the distance between the predictions and the *true* concentrations (GC concentration), as a percentage.

Models: motivation

Inside the Supervised Learning (SL) approach, being all the variables numeric,
learning regression algorithms have been implemented by the use of specific packages or through the *caret* package and its functions.

They are Linear Regression, Random Forest Regression (*randomforest package* and *ranger*) and Glmnet: Lasso and elastic-net regularized generalized linear model.

Models: Description

A detailed and in-depth description of the used algorithms is beyond the purpose of this report. Only a brief description of the ML models will be provided and the meaning of the tuning parameters explained.

Linear Regression

Since we want to understand the relation between $[SO_2]$ and the others 233 predictors, we implement, in this case, a multiple linear regression.

The model can be described with this formula:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in} + \epsilon_i$$

where y_i is the prediction, β_0 the intercept and the others β the various coefficients associated at the independent variables x.

Random Forest

Two packages have been compared with each other: *randomforest* and *ranger*. The latter has been trained by using the *caret* package.

This algorithm is less prone to over-fitting. It consists of a "forest" of decision trees. These trees split (at the nodes) into their branches whether one of the input features is above a certain value until the "leaves" are found. These are the predictions that will be averaged to obtain the final result.

If you want know more about this model consult this link

Lasso and elastic-net regularized generalized linear model

Elastic net is a *regularized* regression method (i.e. with penalties terms) that overcomes the limitations of the Lasso (least absolute shrinkage and selection operator) during the fitting phase.

From this Wiki-link:

> *"The software Glmnet permits the estimation of generalized linear models with L1 (the Lasso), L2 (Ridge regression) and mixtures of the two penalties (the elastic net) using cyclical coordinate descent, computed along a regularization path".*

RESULTS AND DISCUSSION

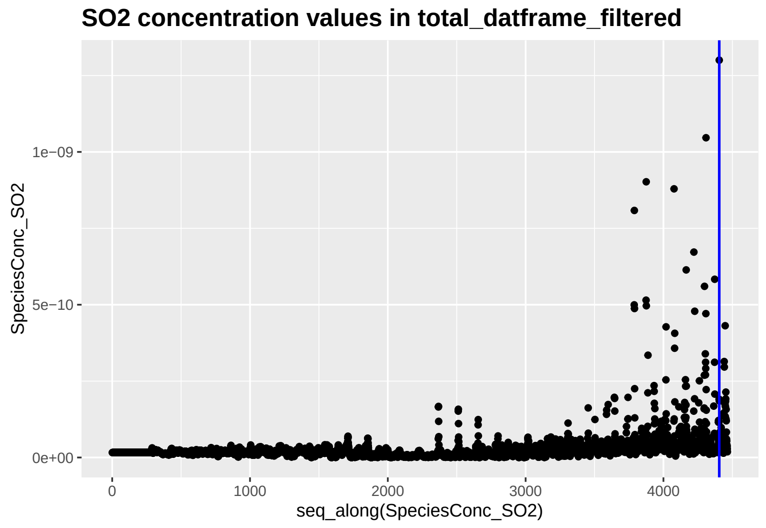
Splitting the dataset

In order to train and validate the models on a test dataset, the *total_dataframe_filtered* has been split in other two subsets:

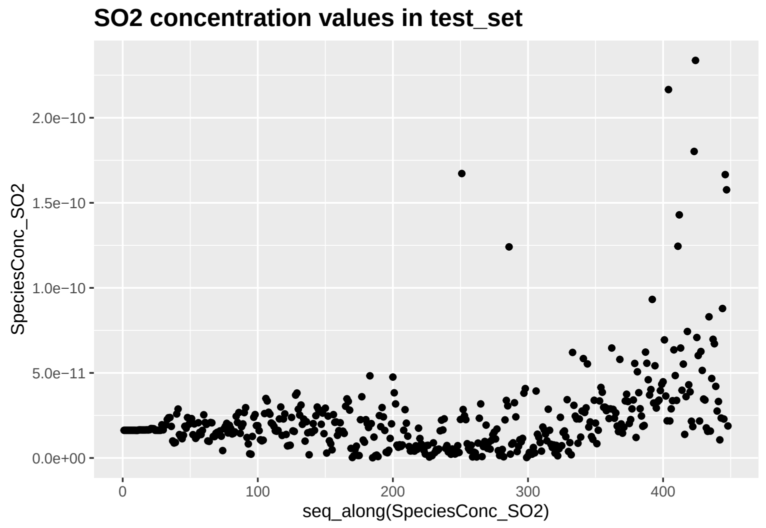
```
##############################################################################
#########                                                       ##############
#########                MACHINE LEARNING                       ##############
##############################################################################

### Generating an 'outcome' object for the createDataPartition() function
### total_dataframe_filtered will be partitionated (90:10) in two subsets
### the train_set and test_set

outcome <- total_dataframe_filtered$SpeciesConc_SO2
set.seed(123)
test_index <- createDataPartition(outcome, times = 1, p = 0.1, list = FALSE)
## p = 0.1 means 10% to the test_set

## Building the train and test data sets

test_set <- total_dataframe_filtered[test_index, ]
train_set <- total_dataframe_filtered[-test_index, ]
```

We are interested in knowing how well the various models can predict these $[SO_2]$ concentration values:

```
total_dataframe_filtered %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), y = SpeciesConc_SO2))+
  ggtitle("SO2 concentration values in total_datframe_filtered")+
  theme(plot.title = element_text(size = 15, face = "bold"))+
  geom_vline(xintercept = 4405,
                color = "blue", size=0.75)
```



```
test_set %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), y = SpeciesConc_SO2))+
  ggtitle("SO2 concentration values in test_set")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



The plot represents the sulfur dioxide concentration values concerning the *test_set* dataset. The NRMSE estimation will permit to quantify the performance of each model.

If the mean value of sulfur dioxide concentration is taken as prediction for all the grid cells the results are these:

```
#### Metrics for comparison: RMSE error and NRMSE
### NMRSE will be RMSE / standard deviation of [SO2] in train_set

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

### Calculating the RMSE associated at the "pure average" algorithm
### Just as a baseline
```

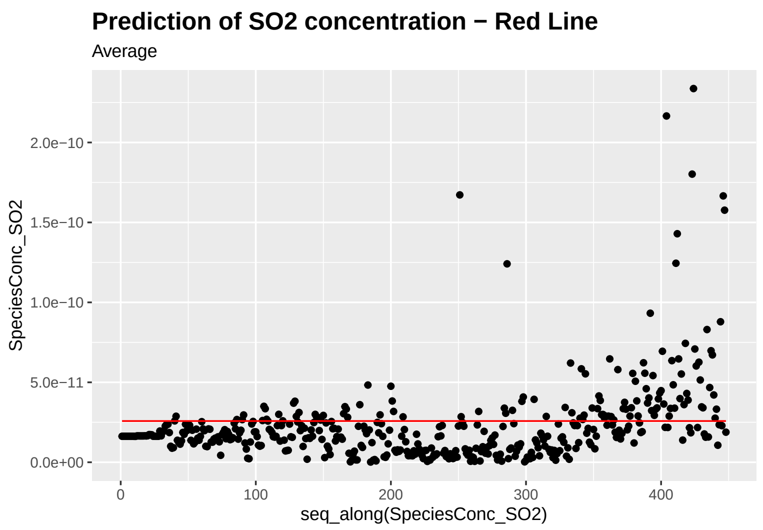
```
RMSE_average <- RMSE(test_set$SpeciesConc_SO2, SO2_mean)
RMSE_average
```

```
## [1] 2.627619e-11
```

```
### 2.627619e-11 the error is bigger than the mean value

##### STANDARD DEVIATION
SO2_sd_train <- sd(train_set$SpeciesConc_SO2)

### Normalized RMSE on the train sd value
NRMSE_average <- (RMSE_average/SO2_sd_train)*100
### This is not a good result: 50% far from the reality

NRMSE_average
```

```
## [1] 49.81961
```

```
test_set %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), SpeciesConc_SO2)) +
  geom_line(aes(seq_along(SpeciesConc_SO2), SO2_mean), col="red")+
  ggtitle("Prediction of SO2 concentration - Red Line", subtitle = "Average")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



Linear Regression

This is the code used for implementing a Linear Regression model:

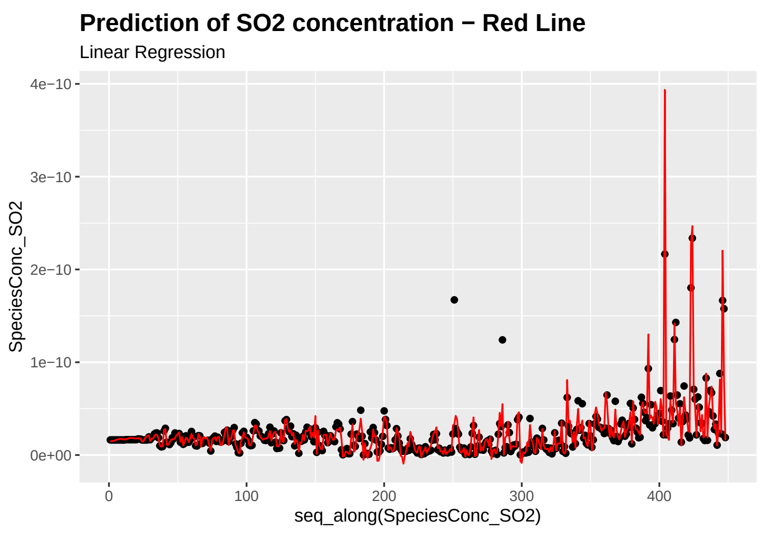
```
### LINEAR REGRESSION
#### Training with caret train() function a "lm" model

### Repeated cross-validation (3 times)

control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 7
set.seed(seed)

fit_lm <- train(SpeciesConc_SO2 ~ .,
                data = train_set,
                method="lm",
                preProcess = c("center","scale"),
                trControl=control)

print(fit_lm)
```

```
## Linear Regression
##
## 4016 samples
##  227 predictor
##
## Pre-processing: centered (227), scaled (227)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 3615, 3615, 3612, 3614, 3615, 3614, ...
## Resampling results:
##
##   RMSE          Rsquared   MAE
##   2.186016e-11  0.8244334  6.371784e-12
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
###Predicting with lm
y_clm = predict(fit_lm, newdata = test_set)

RMSE(y_clm, test_set$SpeciesConc_SO2)
```

```
## [1] 1.389955e-11
```

```
### Normalized RMSE (on standard deviation)
NRMSE_lm <- (RMSE(y_clm, test_set$SpeciesConc_SO2)/SO2_sd_train)*100  ### 25,48%

####Plotting the result

test_set %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), SpeciesConc_SO2)) +
  geom_line(aes(seq_along(SpeciesConc_SO2), y_clm), col="red")+
  ggtitle("Prediction of SO2 concentration - Red Line", subtitle = "Linear Regression")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



```
NRMSE_lm ### 26.35%
```

```
## [1] 26.35352
```

A great improvement is detectable. The NRMSE in this case is 26.35 %.

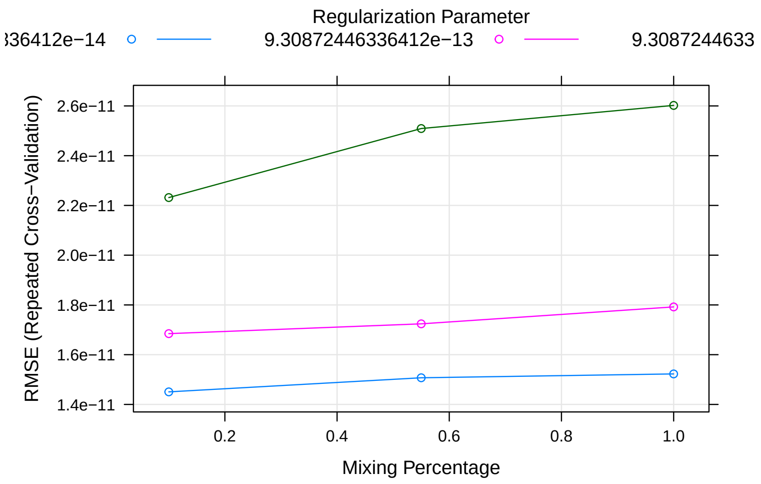
Glmnet model

A slight improvement can be obtain by using a *glmnet* approach, but it does not justify a change of model. The result can be observed using this code:

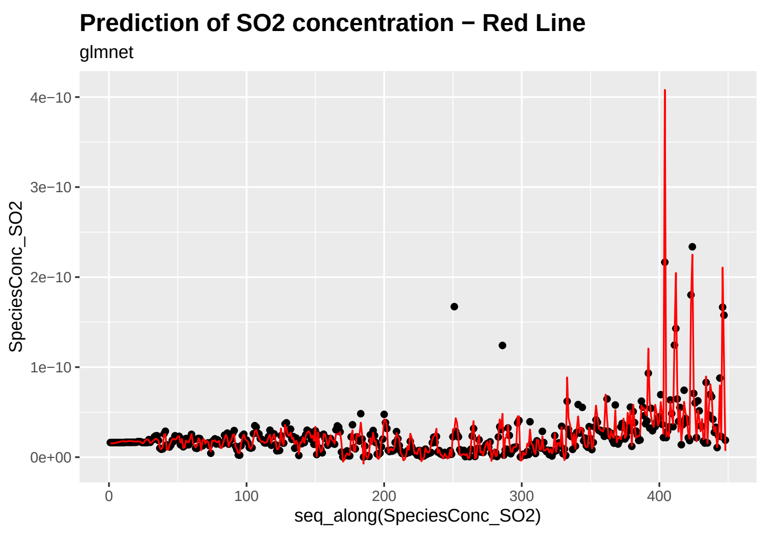
```
#### Training a glmnet model from the caret package

control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 7
metric <- "RMSE"
set.seed(seed)

fit_glmnet <- train(SpeciesConc_SO2 ~ . ,
                         data=train_set,
                         method="glmnet",
                         trControl=control, ### repeated cross-validation
                         preProcess = c("center","scale"),
                         do.trace=TRUE)

## Tuning parameters: • Mixing Percentage (alpha, numeric)
##                    • Regularization Parameter (lambda, numeric)
## as Default

print(fit_glmnet)
```

```
## glmnet
##
## 4016 samples
##  227 predictor
##
## Pre-processing: centered (227), scaled (227)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 3615, 3615, 3612, 3614, 3615, 3614, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda        RMSE          Rsquared   MAE
##   0.10   9.308724e-14  1.450487e-11  0.9060031  5.577482e-12
##   0.10   9.308724e-13  1.684587e-11  0.8785994  6.034006e-12
##   0.10   9.308724e-12  2.231593e-11  0.8025009  7.887959e-12
##   0.55   9.308724e-14  1.507157e-11  0.8990324  5.759633e-12
##   0.55   9.308724e-13  1.723972e-11  0.8583333  6.582770e-12
##   0.55   9.308724e-12  2.509115e-11  0.7397519  1.155065e-11
##   1.00   9.308724e-14  1.522629e-11  0.8954815  5.884211e-12
##   1.00   9.308724e-13  1.792046e-11  0.8433987  6.992155e-12
##   1.00   9.308724e-12  2.602081e-11  0.7197917  1.282682e-11
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1 and lambda = 9.308724e-14.
```

```r
plot(fit_glmnet)
```



```r
#### Predicting with glmnet
y_glmnet = predict(fit_glmnet, newdata = test_set)

####Plotting the result

test_set %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), SpeciesConc_SO2)) +
  geom_line(aes(seq_along(SpeciesConc_SO2), y_glmnet), col="red") +
  ggtitle("Prediction of SO2 concentration - Red Line", subtitle = "glmnet")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



```
##### NRMSE for ridge
NRMSE_glmnet <- (RMSE(y_glmnet,test_set$SpeciesConc_SO2)/SO2_sd_train)*100
```

Random Forest model

For what concern the application of RF model the NRMSE improvement is remarkable.

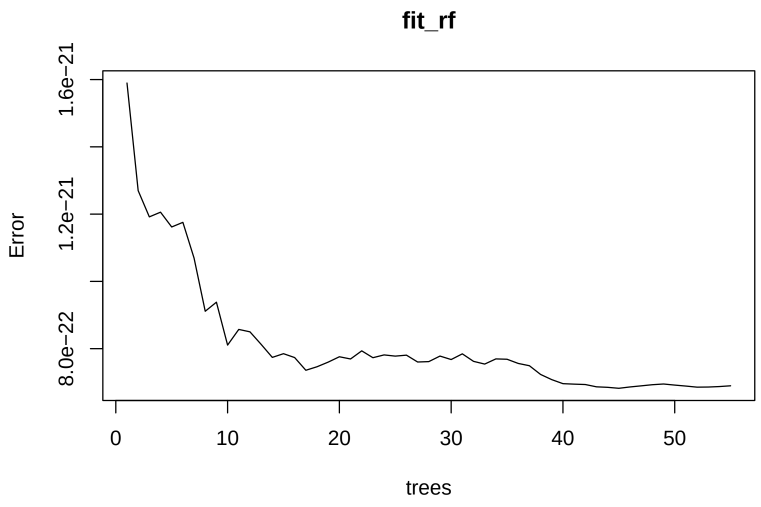
Let’s look at the results.

With the *randomforest* package, a tuning trial and error approach has been applied. The selected tuning parameters are: ntree = 55 (i.e. number of trees), mtry = 72 (i.e. Number of variables randomly sampled as candidates at each split).

```
#### RANDOM FOREST PACKAGE
#### TUNING RF

seed <- 7
metric <- "RMSE"
set.seed(seed)

fit_rf <- randomForest(SpeciesConc_SO2 ~ .,
                       data=train_set,
                       ntree = 55,
                       mtry = 72,
                       preProcess = c("center","scale"),
                       do.trace = TRUE)
```

```
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##    1 | 1.59e-21    57.16 |
##    2 | 1.27e-21    45.66 |
##    3 | 1.192e-21    42.85 |
##    4 | 1.206e-21    43.35 |
##    5 | 1.162e-21    41.77 |
##    6 | 1.175e-21    42.27 |
##    7 | 1.069e-21    38.45 |
##    8 | 9.112e-22    32.76 |
##    9 | 9.382e-22    33.73 |
##   10 | 8.107e-22    29.15 |
##   11 | 8.572e-22    30.82 |
##   12 | 8.501e-22    30.57 |
##   13 | 8.128e-22    29.23 |
##   14 | 7.741e-22    27.83 |
##   15 | 7.851e-22    28.23 |
##   16 | 7.736e-22    27.82 |
##   17 | 7.358e-22    26.46 |
##   18 | 7.462e-22    26.83 |
##   19 |  7.6e-22    27.33 |
##   20 | 7.759e-22    27.90 |
##   21 | 7.694e-22    27.66 |
##   22 | 7.935e-22    28.53 |
##   23 | 7.732e-22    27.80 |
##   24 | 7.814e-22    28.10 |
##   25 | 7.779e-22    27.97 |
##   26 | 7.807e-22    28.07 |
##   27 | 7.604e-22    27.34 |
##   28 | 7.615e-22    27.38 |
##   29 | 7.78e-22    27.98 |
##   30 | 7.677e-22    27.60 |
##   31 | 7.846e-22    28.21 |
##   32 | 7.623e-22    27.41 |
##   33 | 7.542e-22    27.12 |
##   34 | 7.696e-22    27.67 |
##   35 | 7.686e-22    27.63 |
##   36 | 7.561e-22    27.19 |
##   37 | 7.493e-22    26.94 |
##   38 | 7.233e-22    26.01 |
##   39 | 7.08e-22    25.46 |
##   40 | 6.96e-22    25.03 |
##   41 | 6.946e-22    24.97 |
##   42 | 6.934e-22    24.93 |
##   43 | 6.865e-22    24.69 |
##   44 | 6.851e-22    24.63 |
##   45 | 6.822e-22    24.53 |
##   46 | 6.862e-22    24.67 |
##   47 | 6.895e-22    24.79 |
##   48 | 6.929e-22    24.91 |
##   49 | 6.95e-22    24.99 |
##   50 | 6.917e-22    24.87 |
##   51 | 6.888e-22    24.77 |
##   52 | 6.856e-22    24.65 |
##   53 | 6.86e-22    24.66 |
##   54 | 6.874e-22    24.72 |
##   55 | 6.896e-22    24.80 |
```

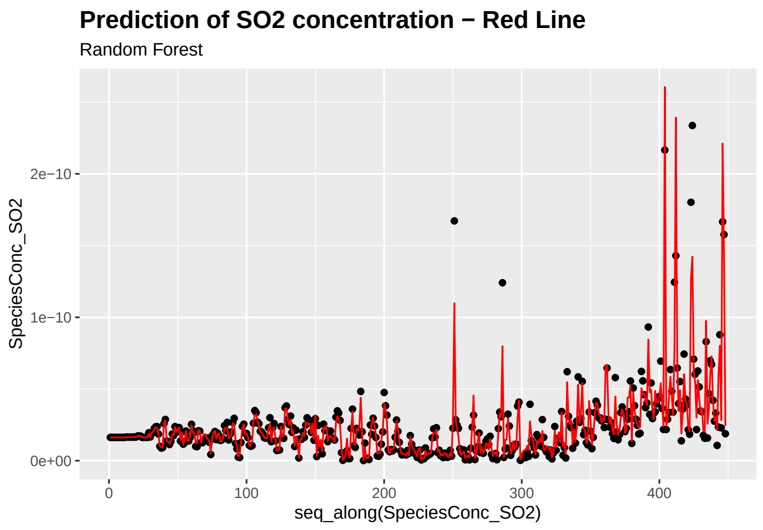
```
plot(fit_rf)
```



```
print(fit_rf)
```

```
##
## Call:
##  randomForest(formula = SpeciesConc_SO2 ~ ., data = train_set,      ntree = 55, mtry = 72, preProcess
##                Type of random forest: regression
##                      Number of trees: 55
## No. of variables tried at each split: 72
##
##           Mean of squared residuals: 6.896031e-22
##                     % Var explained: 75.2
```

```
y_rf = predict(fit_rf, newdata = test_set)

#### Visualizing the fitting

test_set %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), SpeciesConc_SO2)) +
  geom_line(aes(seq_along(SpeciesConc_SO2), y_rf), col="red") +
  ggtitle("Prediction of SO2 concentration - Red Line", subtitle = "Random Forest")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



```
##### NRMSE for random forest

NRMSE_rf <- (RMSE(y_rf,test_set$SpeciesConc_SO2)/SO2_sd_train)*100 ### 18.25%

NRMSE_rf
```

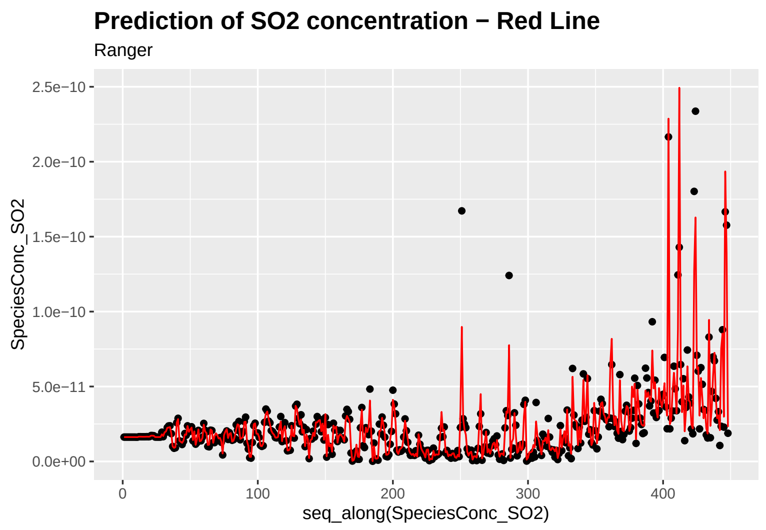
```
## [1] 18.25682
```

The NRMSE associated with this model is 18.25%

The performance is remarkable also using the *ranger* package trained by *caret*.

Here the results:

```
### TUNING ranger

fit_ranger <- ranger(SpeciesConc_SO2 ~ .,
                     data = train_set,
                     num.trees = 55,
                     mtry = 71
                )

y_ranger <- predict(fit_ranger, data = test_set)

NRMSE_ranger <- (RMSE(y_ranger$predictions,test_set$SpeciesConc_SO2)/SO2_sd_train)*100

test_set %>%
  ggplot() +
  geom_point(aes(x = seq_along(SpeciesConc_SO2), SpeciesConc_SO2)) +
  geom_line(aes(seq_along(SpeciesConc_SO2), y_ranger$predictions), col="red") +
  ggtitle("Prediction of SO2 concentration - Red Line", subtitle = "Ranger")+
  theme(plot.title = element_text(size = 15, face = "bold"))
```



This model shows the best NMRSE = 18.06 %.

If the variable importance is calculated for the predictors used during the RF-ML model implementation, the results show again that the "pFe" chemical species (anthropogenic iron) plays an important role on sulfur dioxide concentration.

CONCLUSION

With the aim of using Machine Learning algorithms for the study of gas-phase chemistry in atmospheric Chemistry Transport Models, a series of ML models have been tested on a specific dataset produced by GEOS-Chem software simulation.

The values of sulfur dioxide concentration $[SO_2]$ in the selected geographical area have been predicted using other 233 chemical species concentrations and meteorological parameters (as predictors).

The following table shows a summary of the performances (as %):

```
######################################
### RMSE RESULTS on TEST DATA SET
######################################

RMSE_table <- tibble(method ='Algorithm - Average', NRMSE = 49.82 ) %>%
              add_row(method='Algorithm - Linear Regression', NRMSE = 26.35 )%>%
              add_row(method='Algorithm: glmnet', NRMSE = 26.20 ) %>%
              add_row(method='Algorithm: RF', NRMSE = 18.26 ) %>%
              add_row(method='Algorithm: ranger', NRMSE = 18.06)


RMSE_table
```

```
## # A tibble: 5 x 2
##   method                        NRMSE
##   <chr>                         <dbl>
## 1 Algorithm - Average            49.8
## 2 Algorithm - Linear Regression  26.4
## 3 Algorithm: glmnet              26.2
## 4 Algorithm: RF                  18.3
## 5 Algorithm: ranger              18.1
```

This is only the first step on a long journey, but it has been shown how the Random Forest algorithm could be a future replacement of the chemical integrator in CTM softwares.

At the moment, a limitation of this project can be found in a lack of details for what concern the variables correlation study. Clustering techniques or PCA analysis could help in gaining new insights about the chemical link between the various species and meteorological parameters. This would permit to improve the ML model adding other pieces to the RF algorithm.

Having at our disposal other 30 days of GC simulation, a future objective will be using the RF model to predict $[SO_2]$ day by day to build a data time series.

The next simulation days can be used also as further validation datasets.

Acknowledgments

I would like to thank my colleagues Jenny FISHER and Robyn SCHOFIELD for giving me the opportunity to be part of this investigation, and my family for supporting me in each moment.

REFERENCES

Community, The International GEOS-Chem User. 2020. "Geoschem/Geos-Chem: GEOS-Chem 12.8.1." 2020. https://doi.org/10.5281/zenodo.3837666.

Fisher, J., P. Sebastianelli, and R. Schofield. 2021. "GEOS-Chem Model Output for 2017-2018 CAMMPCAN Aurora Australis Voyages." Ver. 1, Australian Antarctic Data Centre. 2021. https://doi.org/10.26179/h na9-ab45.

Keller, C. A., and M. J. Evans. 2019. "Application of Random Forest Regression to the Calculation of Gas-Phase Chemistry Within the GEOS-Chem Chemistry Model V10." *Geoscientific Model Development* 12 (3): 1209–25. https://doi.org/10.5194/gmd-12-1209-2019.

Seinfeld, J. H., and S. N. Pandis. 2016. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change*. Wiley. https://books.google.com.au/books?id=n/_RmCgAAQBAJ.