MACS Introduction to Machine Learning

Final Project: Predictive Modeling of Wildfire

Borui Sun[1]

Bohan Yin[2]

Shuai Yuan[3]

[1] Methodology and Code Development
[2] Data Retrieve and Feature Selection
[3] Data Retrieve and Result Analysis

Introduction

Wildfire is a natural phenomenon, and it has many undesirable impacts on human safety, health and regional economic development. Wildfires have inflicted major damage to life and property and posed a great ecological threat to numerous places in the world. In October 2019, the news that Amazon rainforest fires have been burning over three weeks rapidly occupied almost all of the press main pages overnight. The long-lasting fire is considered highly destructive and harmful for both human beings and the environment.

Therefore, early detection and prevention will be of utmost importance in reducing the loss triggered by wildfire. To achieve this, this research project aims to leverage multiple machine learning methods to develop a predictive modeling of wildfires by using data collected from satellite images and weather stations throughout 2019. There are 5 sections throughout the paper. Section I briefly discusses previous literature and findings. Section II introduces the two major datasets used for this research project. Section III discusses the research design, which includes data preprocessing, feature selection, two-stage sampling strategy and spatial fixed effects. Section IV evaluates the model with LDA, Logistic, and Naive Bayes. Only these three linear models are included due to computation difficulty. Section V concludes the findings of the project.

Given the significance of wildfire prevention, the authors believe that the success of this project can provide valuable insights into the early detection of wildfires and early detection mechanism for future wildfire prevention and management.

I. Literature Review

Wildfires occur in different climatic zones and across different land use types, and factors contributing to its outbreak can be very complicated. Mario et.al (2015) identify potential 28 factors corresponding with wildfire ignition and categorizes them into climatic (precipitation, temperature, wind), topographic (slope, aspect, altitude), in-situ (fuel type, soil mixture), historical and anthropogenic factors.[4] This comprehensive research serves as a theoretical guideline for this project to determine possible variables included in the prediction model.

As for machine learning techniques used on this topic, multiple researches have set great examples. Rishickesh et al (2019) choose logistic regression, support vector machine, random forest, K-nearest neighbors, both with and without PCA to predict the eventuality of forest fires in Portugal.[5] Castelli et al. applies SVM, random forests, logistic regression and KNN to estimating burned areas in a wildfire.[6] Sayad et al. also uses similar methods but they don't choose a database but real-time satellite images.[7]

With reference to Mario's research as the theoretical guideline, this project chooses all possible climatic factors (precipitation, temperature, wind), the data of which can be easily accessed and quantified. Inspired by all those excellent previous researches, this project decides to use real-

[4] Mhawej Mario, Faour, and Jocelyne, "Wildfire Likelihood's Elements: A Literature Review," MDPI (Multidisciplinary Digital Publishing Institute, December 8, 2015), https://www.mdpi.com/2078-1547/6/2/282)
[5] Rishickesh.R et al. "Predicting Forest Fires Using Supervised and Ensemble Machine Learning Algorithms," *International Journal of Recent Technology and Engineering 2* 8 (2019): pp. 3697-3705)
[6] Castelli, M., et al. "Predicting Burned Areas of Forest Fires: An Artificial Intelligence Approach." *Fire Ecol* 11 (2015): pp. 106–118)
[7] Sayad, Younes Oulad, et al. "Predictive Modeling of Wildfires: A New Dataset and Machine Learning Approach." *Fire Safety Journal* 104 (2019): pp. 130–146)

time satellite data from NASA, and will try to include as many models as possible to improve accuracy work.

II. Data

Daily Wildfire Occurrence

The data of the target variable are collected by the Fire Information for Resource Management System (FIRMS), developed by the University of Maryland with funds from NASA and UN FAO. These fire data are collected through satellite observation from NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) and NASA's Visible Infrared Imaging Radiometer Suite (VIIRS). Both data include a confidence value to each fire detection, because there have been false fires set off by heated smoke plume in the past, particularly during nighttime. In the VIIRS fire data, the confidence values are set to low, nominal and high, where low confidence are typically associated with areas of sun glint and relatively low temperature anomaly, nominal confidence are those free of potential sun glint and with strong temperature anomaly and high confidence are saturated pixels.[8] The confidence value in MODIS ranges from 0% to 100%, which can be assigned to one of the three fire classes. To reduce the false alarm rate, low-confidence fires (< 40 in MODIS) are removed from our data. Nominal-confidence and high-confidence fires are treated as the same. Fire detections from both satellites are used to reduce the number of neglected fire detection due to cloud obscuration.[9] The FIRMS fire data

[8] https://earthdata.nasa.gov/faq/firms-faq#ed-viirs-375m-product
[9] A fire may be blocked by cloud in one satellite but captured by the other, since two satellites are orbiting at different angles.

also serves as one of the main sources for our predictor variables. We believe that precedent wildfire affects the likelihood of future wildfire occurrence.

Daily Climate Features

The second type of data is climatic data, which are sourced from the Earth System Research Laboratory (ESRL) from National Oceanic and Atmospheric Administration website. The climate data archives a wide range of precipitation, temperature, wind rose, ranging from gridded climate datasets extending hundreds of years to real-time wind profiler data at a single location. The spatial coverage of each climate data contains 2.5 degree by 2.5 degree global grids. Considering computational machine's maximum capacity, we set the time range as the whole year of 2019, with temporal coverage of each climate data at daily level.

III. Method

Data Structure and Preprocessing

The FIRMS fire data reports the coordinates of each fire detection. As fires are observed through satellite images, the coordinates are not necessarily the coordinates of the actual fire. Instead, they are the center point of the pixel flagged as containing one or more fires. The pixel is approximately 1km for MODIS and 375m for VIIRS. The actual fire size is often smaller than the pixel size.

Similar to the structure of FIRMS fire data, instead of predicting the likelihood of wildfire occurrence in a specific location, our model predicts the wildfire likelihood in a given region. We create a raster layer at 0.1 degree by 0.1 degree (~11km) resolution within the minimum-perimeter bounding box enclosing South America. The raster layer contains 388,627 grid cells in total and 160,959 after removing non-land regions. This raster layer serves as the basis for data cleaning and preprocessing.

The FIRMS fire locations are mapped into each grid cell in the raster layer. Pixels without any wildfire occurrences are labeled as 0 and pixels that have one or more occurrences are labeled as 1 (since our pixels are at a lower resolution than the two satellite fire data, one grid cell may contain more than one wildfire occurrence). Other variables- precipitation, wind and temperature- are also matched into the raster layer by calculating the area-weighted mean.

Another important feature is that the raster layer is not an equal-area grid. As the Earth is an ellipsoid that is flattened at the poles and bulges at the Equator, grid cells closer to the Equator have larger areas, while grid cells closer to the pole are smaller in size. We do not convert our raster layer into an equal area datum because our data and most of the data available are set in the equal-degree WGS84 reference system. In our model, area of grid cells ranges from 69.44 km^2 to 123.09 km^2.

Feature Selection

The target variable in our model is a dummy variable of whether or not the wildfire will occur in the near future. In this research, we select “the day after tomorrow” as the target day. Our selection is from a pragmatist perspective. As wildfire prevention requires preparation beforehand, we leave a one-day gap in our predictions so that stakeholders can have enough time to react. Based on literature review and research, we select fire precedents, precipitation, temperature, and wind rose as our predictors that could impose potential impact on wildfire occurrence. Here we highlight the selection of fire precedence as one key feature. Given the fact that wildfires could last multiple days, while some prolong more than a month, it is reasonable to assume that past fire instances are highly correlated with whether or not wildfires will occur on a target date in the future.

Historical factors- wildfire occurrences in the past- are proven to have a strong impact on future wildfire occurrences. [10] However, there is little agreement on to what extent of the past we should track back to. Therefore, by applying a LASSO penalized regression, we are able to narrow down precedence date ranges by only selecting dates that are most significant in influencing the wildfire occurrence.

Figure 1: λ for LASSO

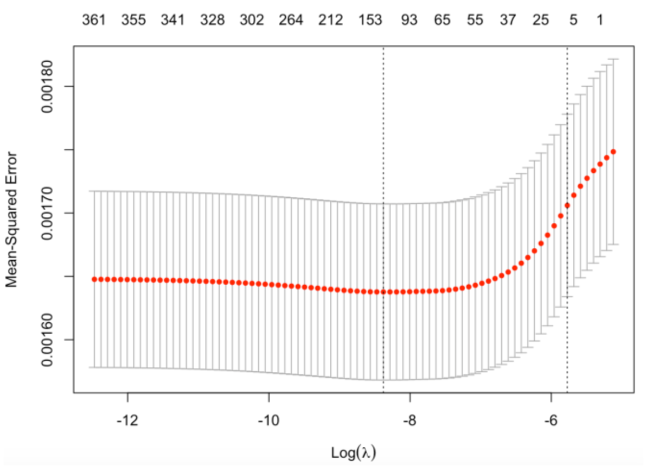


Table 1: LASSO Selected Precedence Dates

Days_before_20191231	Coef
1	0.0013854
2	0.0011299
3	0.0524864
4	0.0148393
15	0.0007564
26	0.0100894
333	0.0026858
338	0.0014268
339	0.0010313
352	0.0013337
360	0.0142647
361	0.0003715

We selected December 31st, 2019 as the target date. After LASSO regression, only the first 26 days and days after day 333 survived. It is worth to mention that coefficients of all climate factors including wind, temperature and precipitation shrink to zero, which means that they are eliminated through the selection process. We speculate the reason behind this is that resolutions of climate data and fire data are not compatible with each other. The spatial coverage of each

[10] Mhawej Mario, Faour, and Jocelyne, "Wildfire Likelihood's Elements: A Literature Review," MDPI (Multidisciplinary Digital Publishing Institute, December 8, 2015), https://www.mdpi.com/2078-1547/6/2/282)

climate data contains 2.5 degree by 2.5 degree global grids, whereas the raster layer that we created for the fire occurrence data contains 0.1 x 0.1 degree grids. The low resolution of climate data fails to match with high resolution of fire occurrence data. This inconsistency could cause that each fire occurrence grid cell does not have compatible climate data, showing subtle climatic variance across all regions in South America. As a result, all climate features are dropped during penalized feature selection.

In terms of lambda selection, we chose the 1se lambda λ, which lands at the value of 0.003109. We choose using $\lambda 1.se$ because it hedges against overfitting by selecting a larger λ value than the min. Table 1 shows the detailed result of coefficients that survived after penalized regression. Based on the result, the range of 26 days before the target day seems to include days that have significant impact on the target day's wildfire occurrence. Therefore, we manually set the cutoff point at 20 days prior to the target day (which is 2019-12-31). This cutoff will be tested and validated in the later section of this paper when discussing the cross-validation test results.

It is also worth to mention that 333 days to 361 days prior to the target day also survived after the penalized regression. Excluding them when setting the cutoff point does not mean that we deny their correlation with the target day wildfire occurrence. On the contrary, it demonstrates a solid relationship between these days and the target day, but in reverse causation roles. If we extend 4 days before the earliest significant day (day 361 in Table 1), we will arrive at the exact same day with the target day, but one year before (which is 2018-12-31). Next, if we start from this day (2018-12-31), and push forward roughly 30 days, we will land at the days between 2019-01-20 to 2019-02-02. This could be interpreted that the wildfire instances on 2018-12-31 have on-going significant impacts on wildfire occurrences on days between 2019-01-20 to 2019-02-02.

Meanwhile, this range is where the range between 333 days to 339 days before the target day falls into. Considering the possibility that the two same days from two sequent years would have similar feature and environment, we speculate that the impacts from the day 2018-12-31 on its following 30 days are imposed (transferred) on the day 2019-12-31, causing LASSO regression to mistreat day 333 to day 361 as coefficients that have significant effect on the target day wildfire occurrence. In fact, day 333 to day 361 only have correlation with the target day, but not causation.

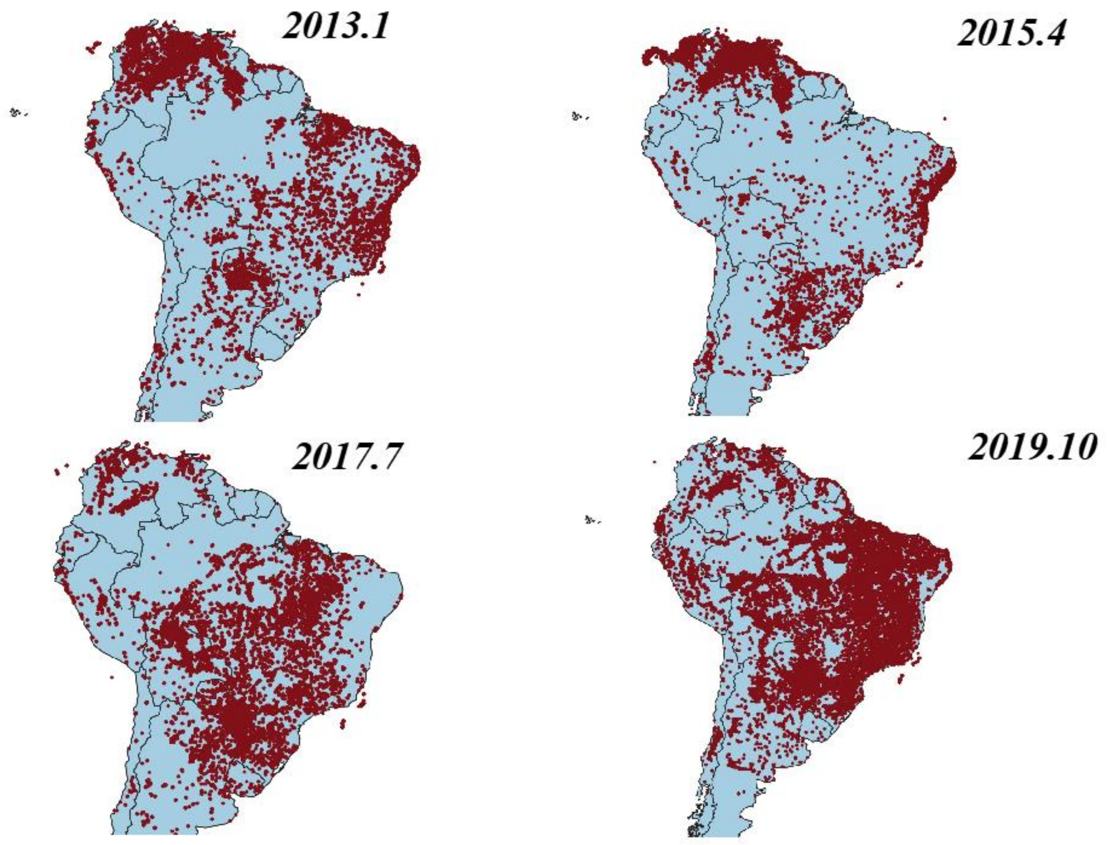
Unsupervised Clustering and Fixed Effect

Systematic spatial variation presents in many variables. Adjacent locations may exhibit more similar values than those further apart. Failing to control for such variation may cause bias in machine learning models, even if the spatial variation is not of interest in itself.[11] Manually looking through FIRMS fire maps across the past 5 years, we are able to observe a spatial pattern of wildfire occurrence. Fig.2 shows the distributions of wildfire happened within the first week of the four months from different years. Although wildfire incidents are more concentrated in the northern area during the summer (Dec-May) and migrate to the south during winter (Jun-Nov), certain regions are frequently flagged with occurrence across the year. It is possible that there are some unobservables that distinguish regions with high frequency of wildfire occurrences and regions with low frequency of wildfire occurrences. Government regulations and local policies on natural reserves, for example, may significantly influence the probability of wildfire

[11] Avald Sommeroll and Dag E. Sommervoll, "Learning from Man or Machine: Spatial Fixed Effects in Urban Econometrics," ScienceDirect, July 2019, https://www.sciencedirect.com/science/article/pii/S016604621830303X)

occurrence. As such information is difficult to identify and collect, spatial fixed effects are used to prevent omitted variable bias by controlling for time-invariant, within-group variations.

Figure 2: Fire Spots Visualization for Different Seasons



Spatial fixed effects rely on spatial aggregation by country, city or county. In our studies, spatial variation does not seem to happen at those levels, as the spatial pattern is not consistent within country, city or county borders. Therefore, unsupervised clustering is used to identify the optimal level of fixed effects to include in our model. Daily wildfire occurrence data throughout 2019 is fed into the K-means clustering algorithm to uncover non-random structures. Even though clustering is an unsupervised machine learning method, since our research uses it as parts of the supervised learning process, cross-validation is used to find the optimal number of clusters.

Two-Stage Sampling and Type I Error

While a large training dataset can improve model performance, due to computational limitations, only a small subset of the full sample is used to train our models. One major concern with using a small training dataset is that our model is learning from partial information and may not be able to fully capture the underlying relationship. If presented are examples having patterns different than the ones our model is trained on, there is high probability the predictions are incorrect. A two-stage cluster sampling strategy is used to reduce the chances of losing important information while shrinking the training data size. We first partition the entire 2019 sample into four non-overlapping clusters, based on the season of fire occurrences, and randomly draw a simple random sample within each season-cluster. In the second stage, we divide each simple random sample further into an occurrence and a non-occurrence group. One simple random sample is then taken from each group.

We believe that the two-stage sampling can improve our model performance and make the predictions more robust for two reasons. To begin with, latent temporal variation exists but is not captured by our model. Similar to the idea of spatial fixed effects, there are seasonal patterns regarding the locations of wildfire occurrence. As shown in Fig.2, wildfire occurrences during summer exhibit different geographical patterns than wildfire happening in the winter. Another form of seasonal patterns is the distinction between high-frequency wildfire seasons and low-frequency wildfire seasons. With only 1 year of wildfire data, it is difficult to statistically identify the longitudinal trend of seasonal variations. Instead, we informally partition 2019 into four seasons and sample independently from each cluster. The first-stage cluster sampling by season can reduce bias by training our model with different seasonal patterns. More importantly, it also makes our results more robust as it no longer highly depends on the random draw of the training sample.

The second-stage partition is designed from a pragmatist perspective. Since the wildfire occurrence-to-non-occurrence ratio is extremely low. Even during peak seasons, there are only less than 2,000 occurrences a day out of 160,959 pixels. If we use a random sampling without stratification, our models can achieve a relatively high accuracy by relentlessly predicting non-occurrences, but we will end up with a very low true positive. In some applications, missing any fire might be especially undesirable. By balancing the rate of occurrence versus non-occurrence in the training data, we are able to improve the true positive rate by a substantial amount. Conversely, such manipulation will decrease the true negative rate, which means that more false alarms are reported. In other applications, especially when opportunity cost of putting out false alarm is taken into account, one might be willing to trade a lower detection rate to gain a lower false alarm rate. In those applications, a stratified sampling is still desirable to prevent the random sampling from drawing non-occurrences only.

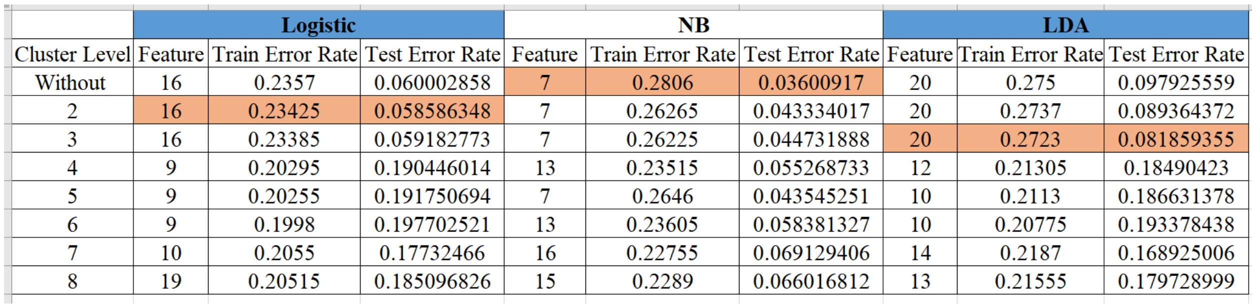
IV. Results

Constrained by computational difficulties, only three linear models could finish running within a reasonable time range (others might take over 2 hours without generating any results): Logistics Regression, Linear Discriminant Analysis and Naïve Bayes. The authors also think of cutting the sample size to resolve this issue, but since two-stage cluster sampling is used to avoid the “zero imputation” problem mentioned above, the richness of machine techniques is compromised for better quality of data.

Main results

After finishing the feature selection by LASSO, the model drops all the other variables but wildfire precedents; however, the accuracy remains incredibly high. This result has implied that fire precedents are likely to have a long-lasting effect (up to 26 days prior) on future possible fire outbreaks, due to the long duration of most wildfires.

Table 2: Model Evaluation Results



	Logistic			NB			LDA		
Cluster Level	Feature	Train Error Rate	Test Error Rate	Feature	Train Error Rate	Test Error Rate	Feature	Train Error Rate	Test Error Rate
Without	16	0.2357	0.060002858	7	0.2806	0.03600917	20	0.275	0.097925559
2	16	0.23425	0.058586348	7	0.26265	0.043334017	20	0.2737	0.089364372
3	16	0.23385	0.059182773	7	0.26225	0.044731888	20	0.2723	0.081859355
4	9	0.20295	0.190446014	13	0.23515	0.055268733	12	0.21305	0.18490423
5	9	0.20255	0.191750694	7	0.2646	0.043545251	10	0.2113	0.186631378
6	9	0.1998	0.197702521	13	0.23605	0.058381327	10	0.20775	0.193378438
7	10	0.2055	0.17732466	16	0.22755	0.069129406	14	0.2187	0.168925006
8	19	0.20515	0.185096826	15	0.2289	0.066016812	13	0.21555	0.179728999

The above table clearly shows the evaluation result for all the three models chosen for this project including models both with and without clusters. The result shows that including clusters into the model does not necessarily improve the accuracy, especially for the Naive Bayes model.

There is not much difference in prediction accuracy across the three models. Naive Bayes model has the lowest test error rate of around 3.6% when the model includes the first 7 days before the target day but includes no cluster, and it performs best among the three. The logistic model has its lowest test error rate of around 5.9% when the model includes the first 16 days before the target day with 2 clusters. And the LDA model has its lowest test error rate of around 8.2% when the model includes the first 20 days before the target day with 3 clusters.

Although these three models disagree on whether the model should include cluster or not nor on the number of days that a precedent fire breaks out prior to the target day, the results have shown that models without cluster or with 2 to 3 clusters perform the best. When cluster size reaches 4, the training error continues to decrease, while the test error starts to rise. This is a clear sign of overfitting.

In terms of the number of precedents to be included, the three models vary substantially. The accuracy of NB model continues to drop when the number of precedents increases, while Logistic model and LDA model generally finds the optimal accuracy at around 20.

Regardless of the optimal cluster size and precedents range, the three models yields a very high accuracy with a limited number of features. One main takeaway from the result is that precedents have the most significant influence on future wildfire occurrences.

Robustness

To test the robustness of the model, several other samples are selected. The results are relatively consistent thanks to the two-stage cluster sampling strategy. Before this strategy is implemented, the accuracy of the model fluctuates drastically between 78% to 98%. It turns out that the fluctuation can be attributed to the change of seed or different distribution of the sample.

Therefore, the two-stage cluster sampling strategy works effectively in improving the robustness of the model.

V. Summary and Conclusions

Findings

Based on the model evaluation result, there are two major take-aways from our projects:

1. Precedent fires have a huge impact on future fire outbreaks, which can be traced back up to 26 days prior to the target day. The most possible reason might be that in most cases wildfires have long durations.
2. All the climatic proponents are dropped for being not significant.
3. No matter which linear model this project chooses, those not including any cluster, or 2 to 3 clusters yield the lowest test error rate. Meanwhile, when the number of clusters increases to 4, an overfitting problem surges.

Limitations

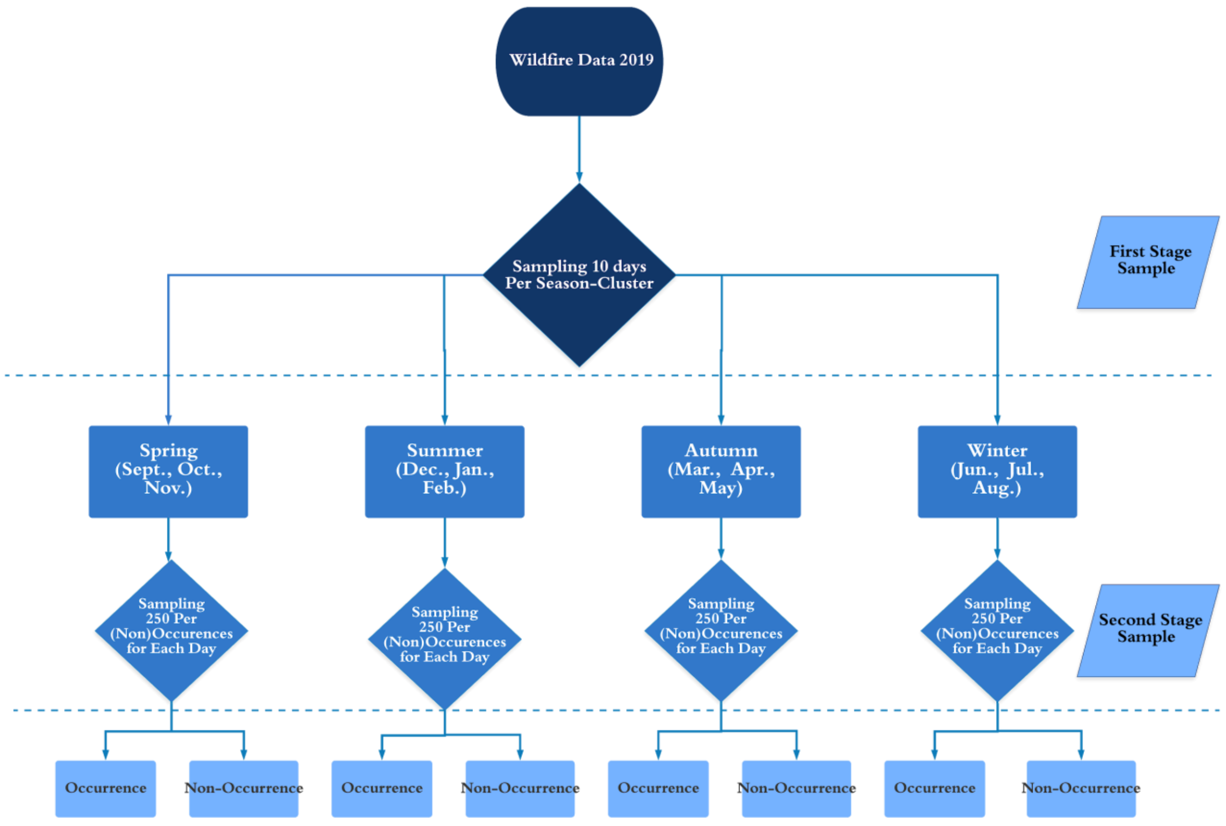
There are two major limitations for this project. Firstly, only one-year data of precedents is collected. According to the LASSO results, precedents that are two years prior to the target day, even three years could influence the prediction (Only those of the same date as the target matter). Future researches should incorporate these data. Secondly, although this project aims to apply as many machine learning techniques as possible, the computation difficulty coerces us to limit the choice to the three methods.

Bibliography

1. *Castelli, M., et al. "Predicting Burned Areas of Forest Fires: An Artificial Intelligence Approach." Fire Ecol. 11, 2015, pp. 106–118. doi:10.4996/fireecology.1101106*
2. *Mhawej, Mario, et al. "Wildfire Likelihood's Elements: A Literature Review." MDPI, Multidisciplinary Digital Publishing Institute, 8 Dec. 2015, www.mdpi.com/2078-1547/6/2/282.*
3. *Rishickesh, R., et al. "Predicting Forest Fires Using Supervised and Ensemble Machine Learning Algorithms." International Journal of Recent Technology and Engineering 2, vol. 8, no. 2, 2019, pp. 3697–3705., doi:10.35940/ijrte.b2878.078219.*
4. *Sayad, Younes Oulad, et al. "Predictive Modeling of Wildfires: A New Dataset and Machine Learning Approach." Fire Safety Journal, vol. 104, 2019, pp. 130–146., doi:10.1016/j.firesaf.2019.01.006.*
5. *Garzón, M. B., et al. "Predicting Habitat Suitability with Machine Learning Models: the Potential Area of Pinus Sylvestris l." In the Iberian Peninsula. Ecol. Model. 197, 2006, pp. 383–393. doi:10.1016/j.ecolmodel.2006.03.015*
6. *Forsell, N., et al. "Reinforcement Learning for Spatial Processes." in 18th World IMACS/MODSIM Congress, Cairns, Australia, 2009, pp. 755–761.*
7. *Dormann, Carsten F., et al. "Methods to Account for Spatial Autocorrelation in the Analysis of Species Distributional Data: a Review." Wiley Online Library, John Wiley & Sons, Ltd, 27 Sept. 2007, onlinelibrary.wiley.com/doi/full/10.1111/j.2007.0906-7590.05171.x.*
8. *"Fire Information for Resource Management System (FIRMS)." NASA, NASA, 24 Jan. 2020, earthdata.nasa.gov/earth-observation-data/near-real-time/firms.*
9. *"Gridded Climate Data" ESRL, NOAA, 24 Jan. 2020 esrl.noaa.gov/psd/data/*

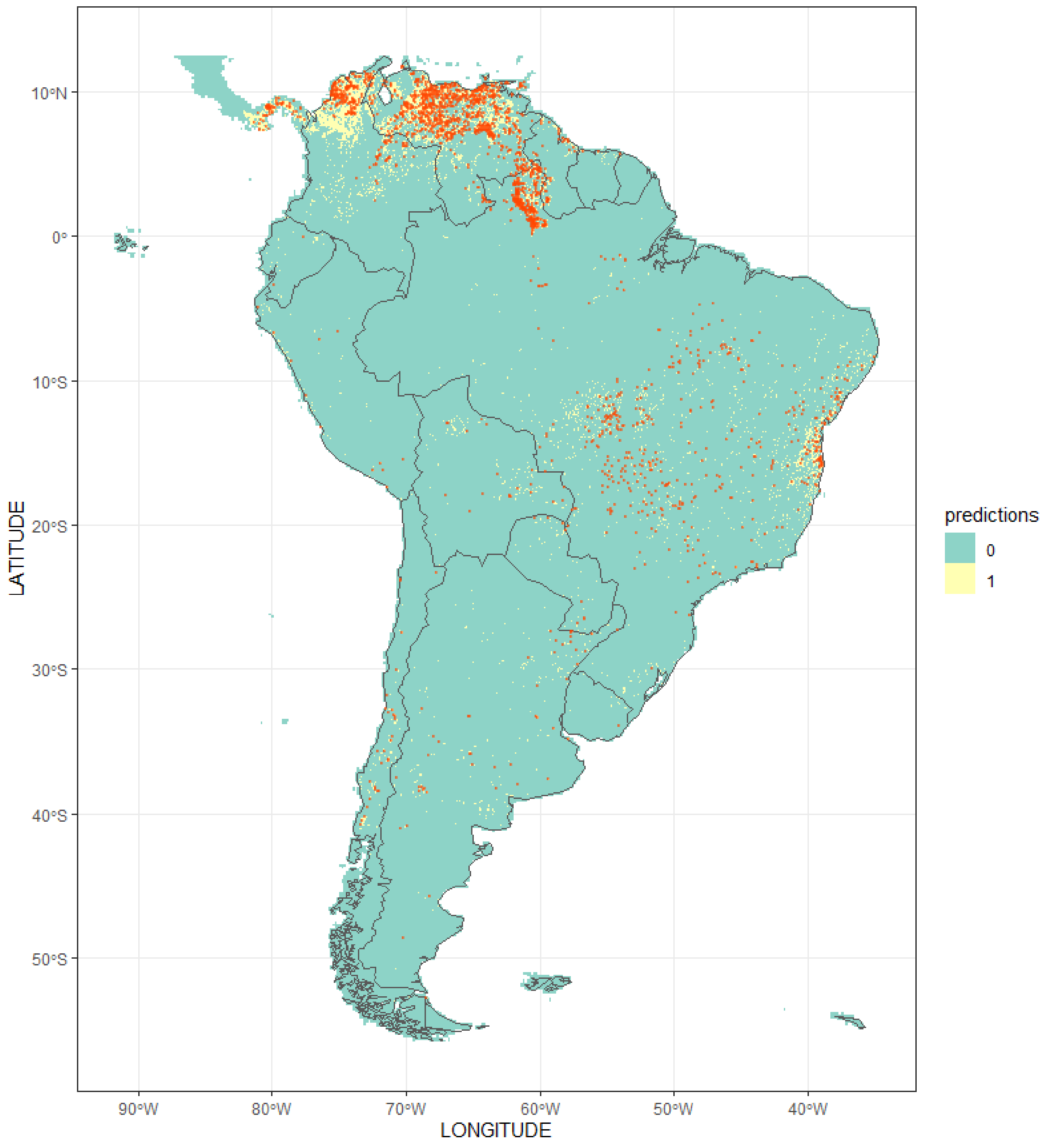
Appendix

Figure 3: Two-Sample Cluster Visulization

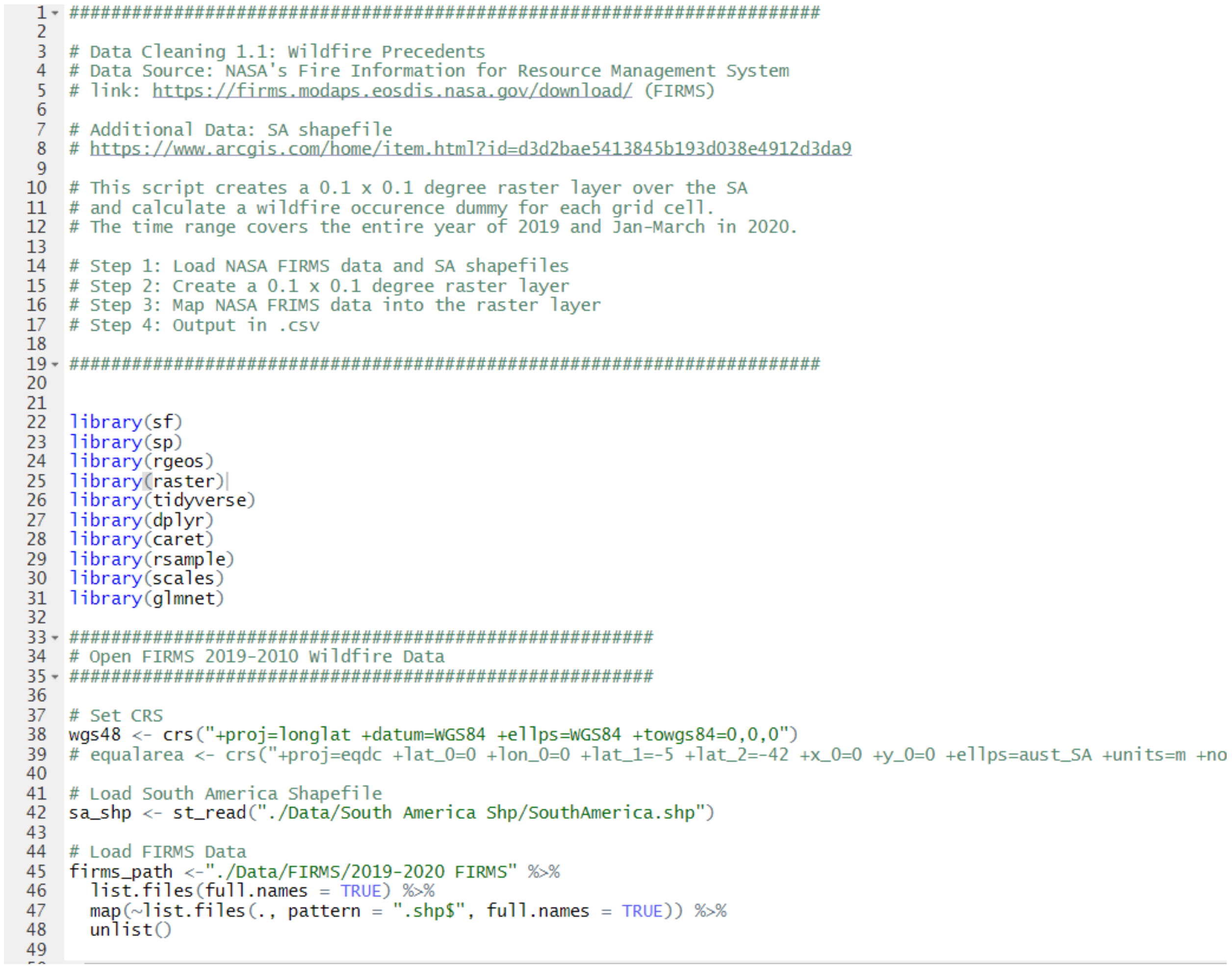


Wildfire Data 2019
Sampling 10 days Per Season-Cluster
First Stage Sample
Spring (Sept., Oct., Nov.)
Summer (Dec., Jan., Feb.)
Autumn (Mar., Apr., May)
Winter (Jun., Jul., Aug.)
Sampling 250 Per (Non)Occurences for Each Day
Sampling 250 Per (Non)Occurences for Each Day
Sampling 250 Per (Non)Occurences for Each Day
Sampling 250 Per (Non)Occurences for Each Day
Second Stage Sample
Occurrence
Non-Occurrence
Occurrence
Non-Occurrence
Occurrence
Non-Occurrence
Occurrence
Non-Occurrence

Figure 4: Wildfire Predictions vs Actual Wildfire Occurrences in 2019-03-19

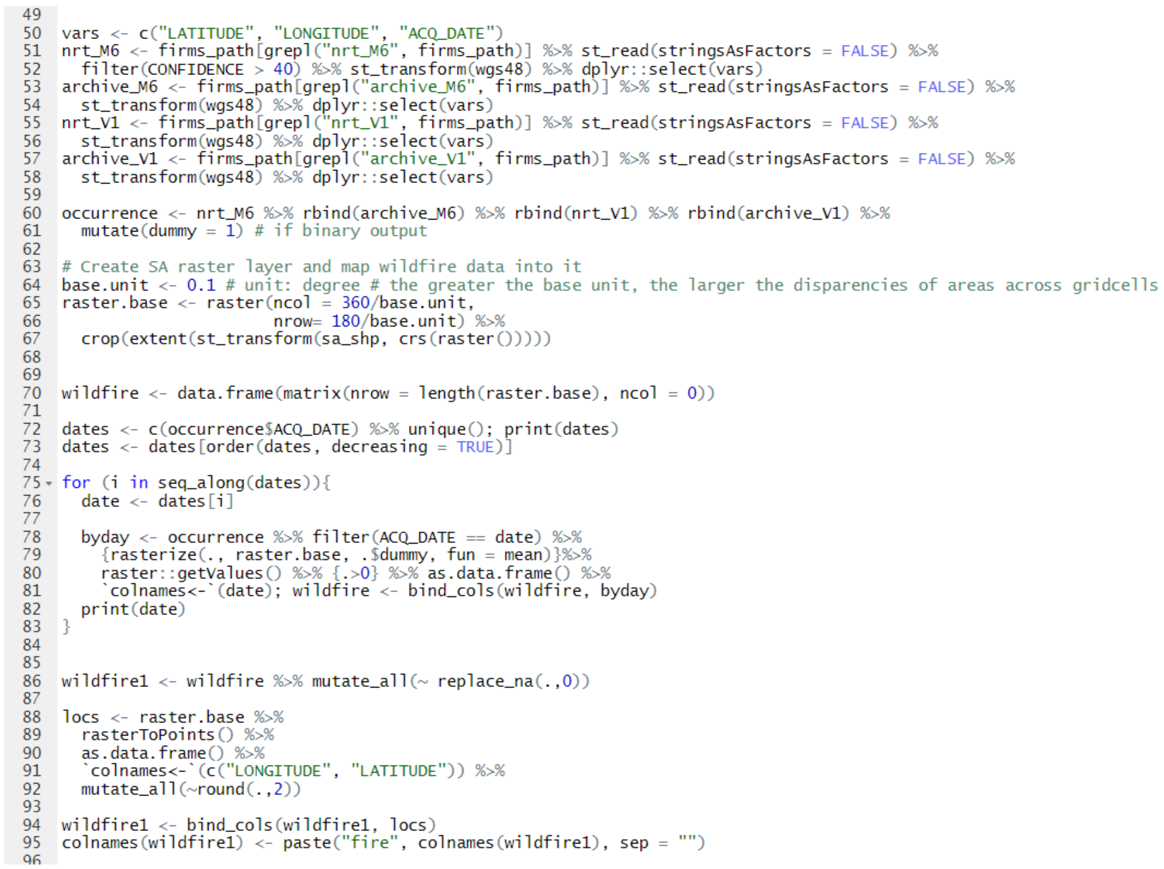

10°N
0°
10°S
20°S
30°S
40°S
50°S
LATITUDE
90°W
80°W
70°W
60°W
50°W
40°W
LONGITUDE
predictions
0
1

R script[12]:

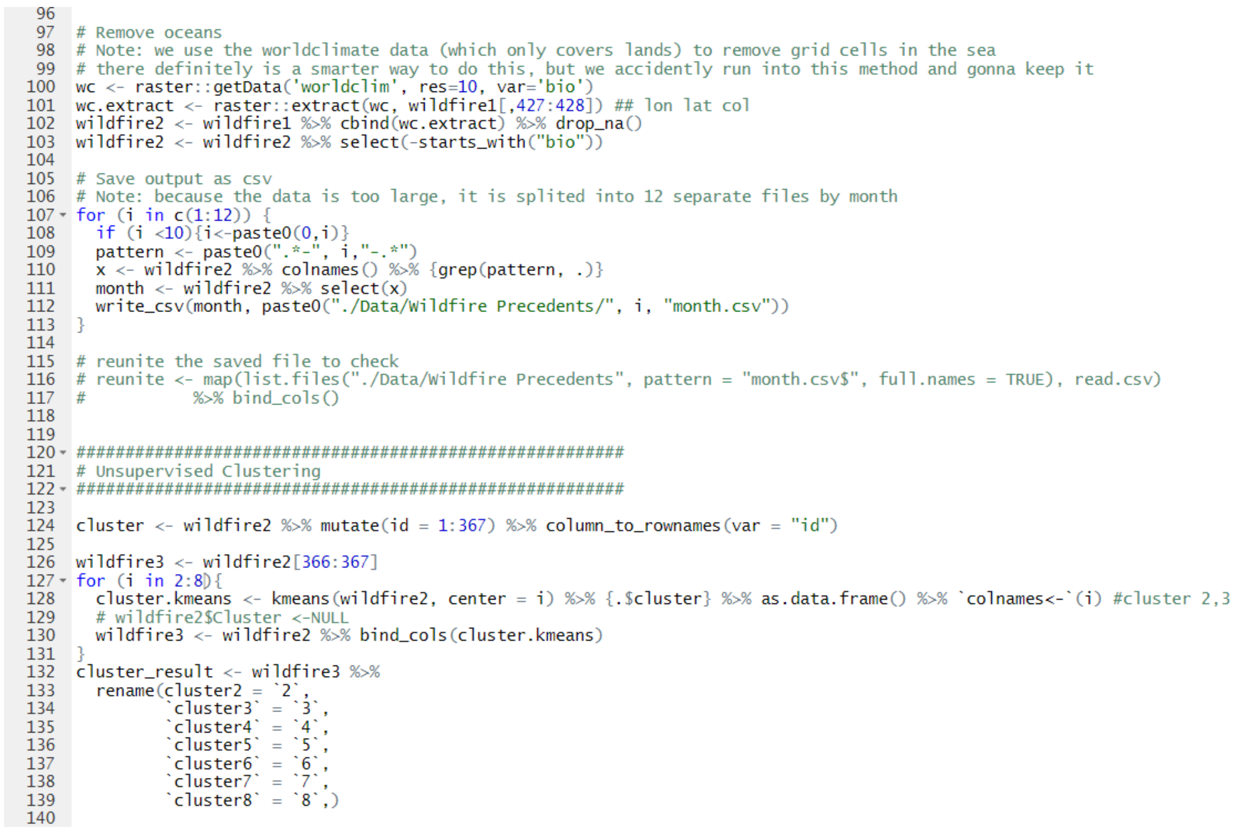


```r
#########################################################################

# Data Cleaning 1.1: Wildfire Precedents
# Data Source: NASA's Fire Information for Resource Management System
# link: https://firms.modaps.eosdis.nasa.gov/download/ (FIRMS)

# Additional Data: SA shapefile
# https://www.arcgis.com/home/item.html?id=d3d2bae5413845b193d038e4912d3da9

# This script creates a 0.1 x 0.1 degree raster layer over the SA
# and calculate a wildfire occurence dummy for each grid cell.
# The time range covers the entire year of 2019 and Jan-March in 2020.

# Step 1: Load NASA FIRMS data and SA shapefiles
# Step 2: Create a 0.1 x 0.1 degree raster layer
# Step 3: Map NASA FRIMS data into the raster layer
# Step 4: Output in .csv

#########################################################################


library(sf)
library(sp)
library(rgeos)
library(raster)
library(tidyverse)
library(dplyr)
library(caret)
library(rsample)
library(scales)
library(glmnet)

########################################################
# Open FIRMS 2019-2010 Wildfire Data
########################################################

# Set CRS
wgs48 <- crs("+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")
# equalarea <- crs("+proj=eqdc +lat_0=0 +lon_0=0 +lat_1=-5 +lat_2=-42 +x_0=0 +y_0=0 +ellps=aust_SA +units=m +no

# Load South America Shapefile
sa_shp <- st_read("./Data/South America Shp/SouthAmerica.shp")

# Load FIRMS Data
firms_path <-"./Data/FIRMS/2019-2020 FIRMS" %>%
  list.files(full.names = TRUE) %>%
  map(~list.files(., pattern = ".shp$", full.names = TRUE)) %>%
  unlist()

```

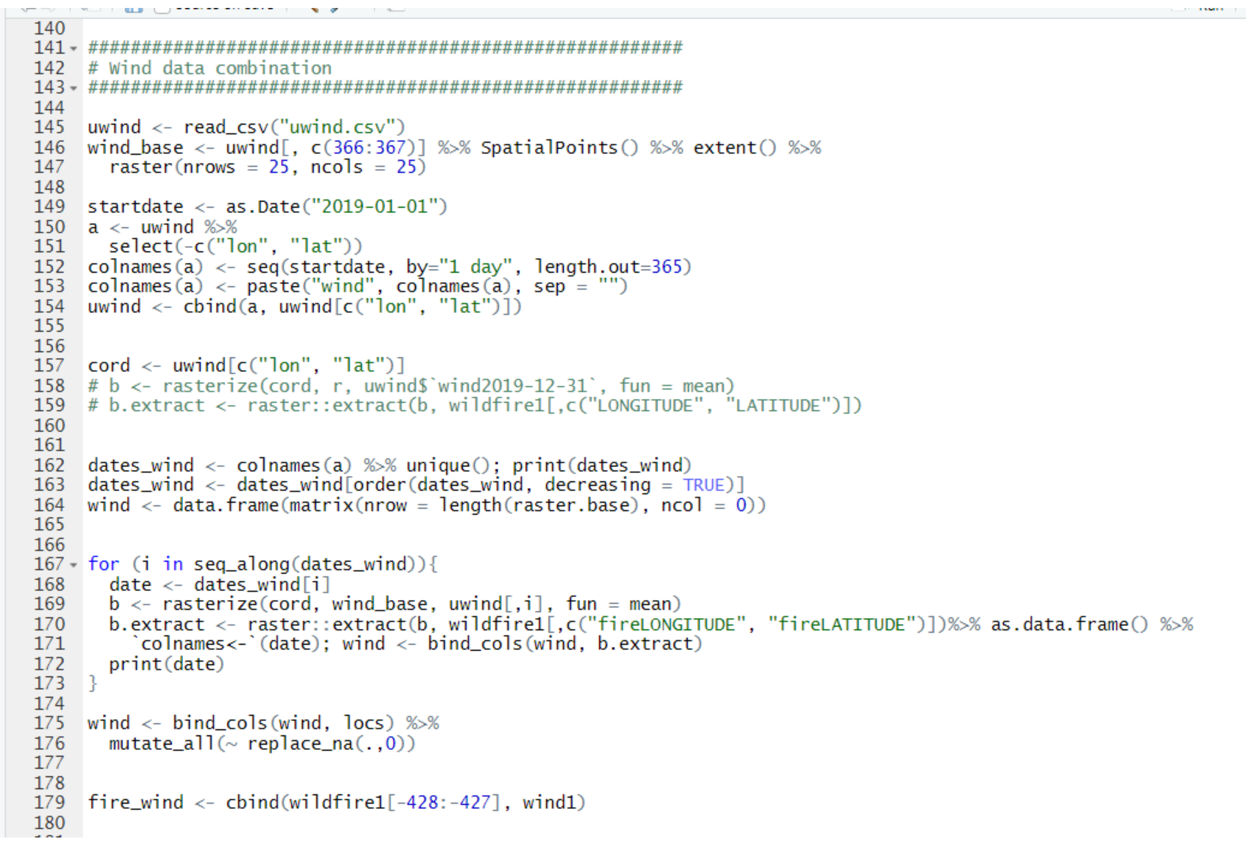
[12] Because it takes approximately more than 1 days to knit the code into a pdf. We simply copy paste all the code here. R scripts are also pushed to Github in case needed.



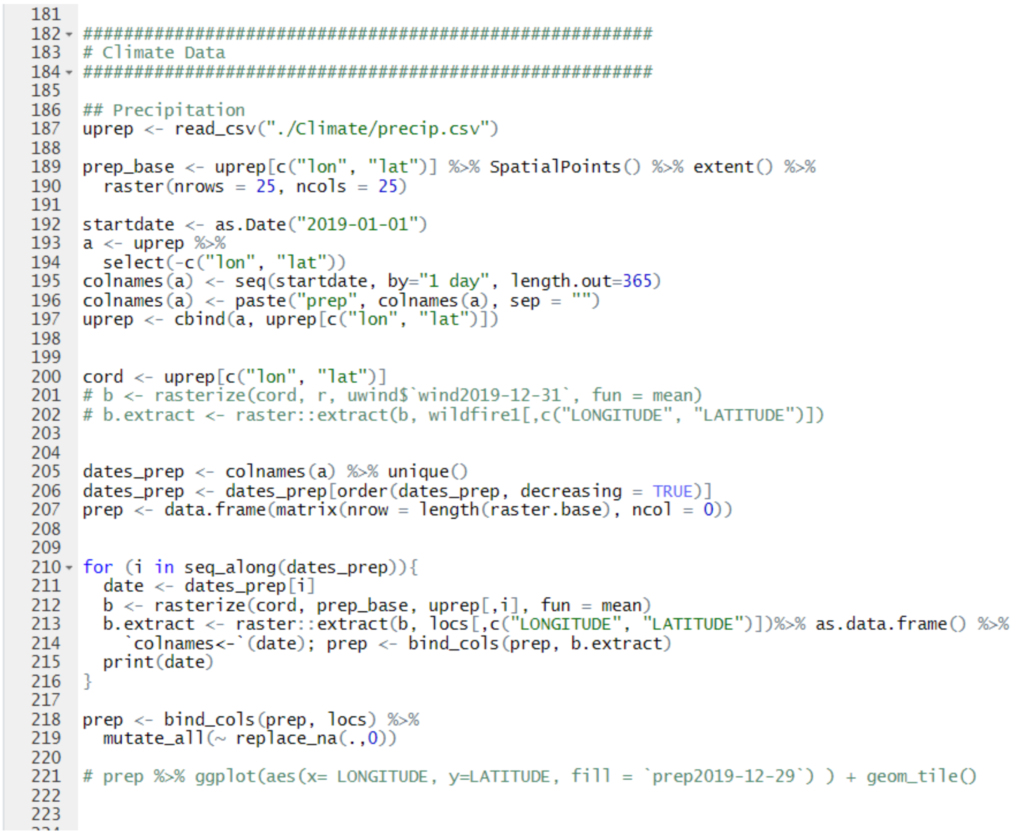
```r

vars <- c("LATITUDE", "LONGITUDE", "ACQ_DATE")
nrt_M6 <- firms_path[grepl("nrt_M6", firms_path)] %>% st_read(stringsAsFactors = FALSE) %>%
  filter(CONFIDENCE > 40) %>% st_transform(wgs48) %>% dplyr::select(vars)
archive_M6 <- firms_path[grepl("archive_M6", firms_path)] %>% st_read(stringsAsFactors = FALSE) %>%
  st_transform(wgs48) %>% dplyr::select(vars)
nrt_V1 <- firms_path[grepl("nrt_V1", firms_path)] %>% st_read(stringsAsFactors = FALSE) %>%
  st_transform(wgs48) %>% dplyr::select(vars)
archive_V1 <- firms_path[grepl("archive_V1", firms_path)] %>% st_read(stringsAsFactors = FALSE) %>%
  st_transform(wgs48) %>% dplyr::select(vars)

occurrence <- nrt_M6 %>% rbind(archive_M6) %>% rbind(nrt_V1) %>% rbind(archive_V1) %>%
  mutate(dummy = 1) # if binary output

# Create SA raster layer and map wildfire data into it
base.unit <- 0.1 # unit: degree # the greater the base unit, the larger the disparencies of areas across gridcells
raster.base <- raster(ncol = 360/base.unit,
                      nrow= 180/base.unit) %>%
  crop(extent(st_transform(sa_shp, crs(raster()))))


wildfire <- data.frame(matrix(nrow = length(raster.base), ncol = 0))

dates <- c(occurrence$ACQ_DATE) %>% unique(); print(dates)
dates <- dates[order(dates, decreasing = TRUE)]

for (i in seq_along(dates)){
  date <- dates[i]

  byday <- occurrence %>% filter(ACQ_DATE == date) %>%
    {rasterize(., raster.base, .$dummy, fun = mean)}%>%
    raster::getValues() %>% {.>0} %>% as.data.frame() %>%
    `colnames<-`(date); wildfire <- bind_cols(wildfire, byday)
  print(date)
}


wildfire1 <- wildfire %>% mutate_all(~ replace_na(.,0))

locs <- raster.base %>%
  rasterToPoints() %>%
  as.data.frame() %>%
  `colnames<-`(c("LONGITUDE", "LATITUDE")) %>%
  mutate_all(~round(.,2))

wildfire1 <- bind_cols(wildfire1, locs)
colnames(wildfire1) <- paste("fire", colnames(wildfire1), sep = "")

```



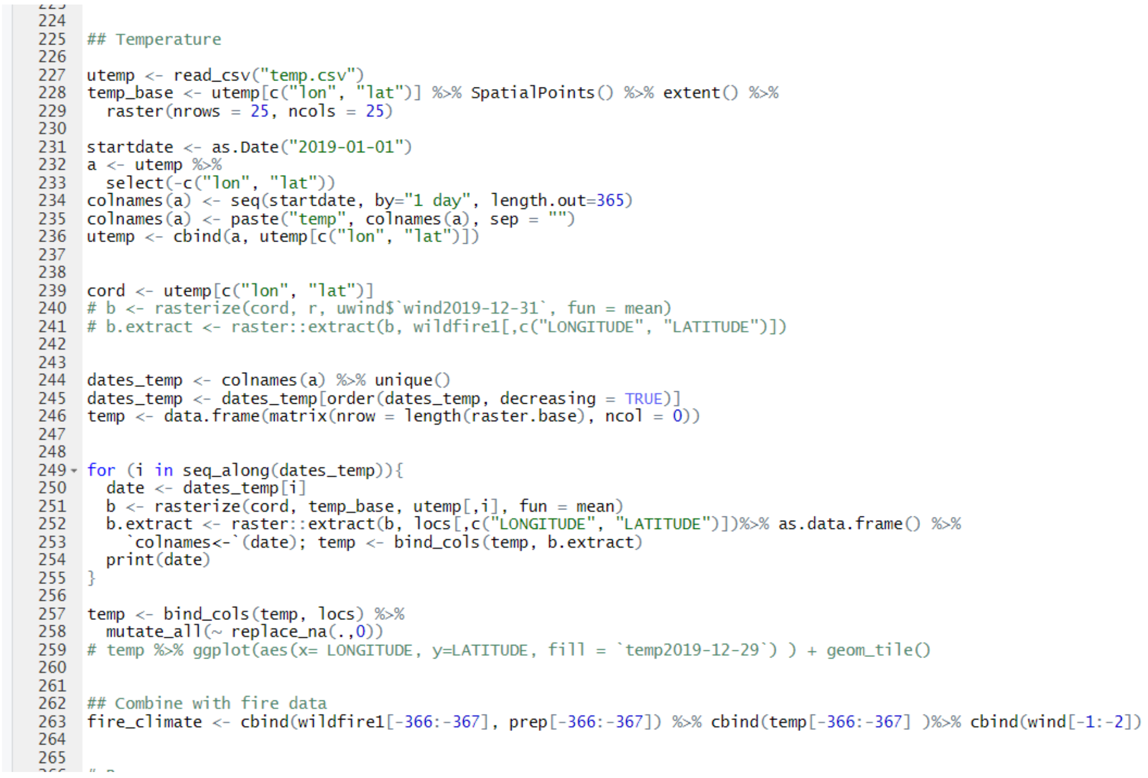
```r

# Remove oceans
# Note: we use the worldclimate data (which only covers lands) to remove grid cells in the sea
# there definitely is a smarter way to do this, but we accidently run into this method and gonna keep it
wc <- raster::getData('worldclim', res=10, var='bio')
wc.extract <- raster::extract(wc, wildfire1[,427:428]) ## lon lat col
wildfire2 <- wildfire1 %>% cbind(wc.extract) %>% drop_na()
wildfire2 <- wildfire2 %>% select(-starts_with("bio"))

# Save output as csv
# Note: because the data is too large, it is splited into 12 separate files by month
for (i in c(1:12)) {
  if (i <10){i<-paste0(0,i)}
  pattern <- paste0(".*-", i,"-.*")
  x <- wildfire2 %>% colnames() %>% {grep(pattern, .)}
  month <- wildfire2 %>% select(x)
  write_csv(month, paste0("./Data/Wildfire Precedents/", i, "month.csv"))
}

# reunite the saved file to check
# reunite <- map(list.files("./Data/Wildfire Precedents", pattern = "month.csv$", full.names = TRUE), read.csv)
#            %>% bind_cols()


###########################################################
# Unsupervised Clustering
###########################################################

cluster <- wildfire2 %>% mutate(id = 1:367) %>% column_to_rownames(var = "id")

wildfire3 <- wildfire2[366:367]
for (i in 2:8){
  cluster.kmeans <- kmeans(wildfire2, center = i) %>% {.$cluster} %>% as.data.frame() %>% `colnames<-`(i) #cluster 2,3
  # wildfire2$Cluster <-NULL
  wildfire3 <- wildfire2 %>% bind_cols(cluster.kmeans)
}
cluster_result <- wildfire3 %>%
  rename(cluster2 = `2`,
         `cluster3` = `3`,
         `cluster4` = `4`,
         `cluster5` = `5`,
         `cluster6` = `6`,
         `cluster7` = `7`,
         `cluster8` = `8`,)

```



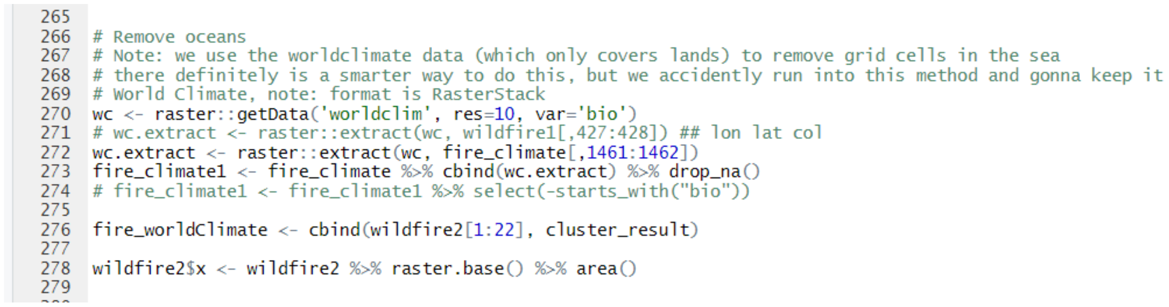
```r

###########################################################
# Wind data combination
###########################################################

uwind <- read_csv("uwind.csv")
wind_base <- uwind[, c(366:367)] %>% SpatialPoints() %>% extent() %>%
  raster(nrows = 25, ncols = 25)

startdate <- as.Date("2019-01-01")
a <- uwind %>%
  select(-c("lon", "lat"))
colnames(a) <- seq(startdate, by="1 day", length.out=365)
colnames(a) <- paste("wind", colnames(a), sep = "")
uwind <- cbind(a, uwind[c("lon", "lat")])


cord <- uwind[c("lon", "lat")]
# b <- rasterize(cord, r, uwind$`wind2019-12-31`, fun = mean)
# b.extract <- raster::extract(b, wildfire1[,c("LONGITUDE", "LATITUDE")])


dates_wind <- colnames(a) %>% unique(); print(dates_wind)
dates_wind <- dates_wind[order(dates_wind, decreasing = TRUE)]
wind <- data.frame(matrix(nrow = length(raster.base), ncol = 0))


for (i in seq_along(dates_wind)){
  date <- dates_wind[i]
  b <- rasterize(cord, wind_base, uwind[,i], fun = mean)
  b.extract <- raster::extract(b, wildfire1[,c("fireLONGITUDE", "fireLATITUDE")])%>% as.data.frame() %>%
    `colnames<-`(date); wind <- bind_cols(wind, b.extract)
  print(date)
}

wind <- bind_cols(wind, locs) %>%
  mutate_all(~ replace_na(.,0))


fire_wind <- cbind(wildfire1[-428:-427], wind1)

```



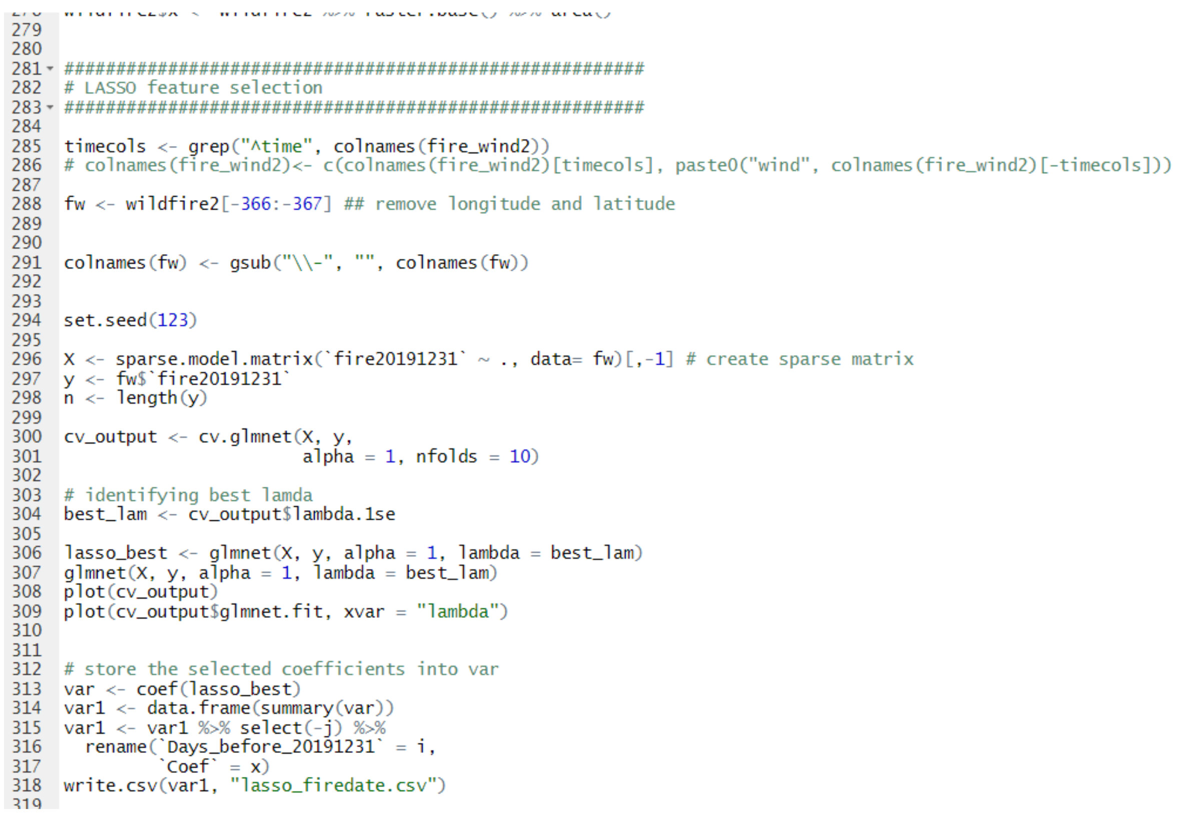
```r

#########################################################
# Climate Data
#########################################################

## Precipitation
uprep <- read_csv("./Climate/precip.csv")

prep_base <- uprep[c("lon", "lat")] %>% SpatialPoints() %>% extent() %>%
  raster(nrows = 25, ncols = 25)

startdate <- as.Date("2019-01-01")
a <- uprep %>%
  select(-c("lon", "lat"))
colnames(a) <- seq(startdate, by="1 day", length.out=365)
colnames(a) <- paste("prep", colnames(a), sep = "")
uprep <- cbind(a, uprep[c("lon", "lat")])


cord <- uprep[c("lon", "lat")]
# b <- rasterize(cord, r, uwind$`wind2019-12-31`, fun = mean)
# b.extract <- raster::extract(b, wildfire1[,c("LONGITUDE", "LATITUDE")])


dates_prep <- colnames(a) %>% unique()
dates_prep <- dates_prep[order(dates_prep, decreasing = TRUE)]
prep <- data.frame(matrix(nrow = length(raster.base), ncol = 0))


for (i in seq_along(dates_prep)){
  date <- dates_prep[i]
  b <- rasterize(cord, prep_base, uprep[,i], fun = mean)
  b.extract <- raster::extract(b, locs[,c("LONGITUDE", "LATITUDE")])%>% as.data.frame() %>%
    `colnames<-`(date); prep <- bind_cols(prep, b.extract)
  print(date)
}

prep <- bind_cols(prep, locs) %>%
  mutate_all(~ replace_na(.,0))

# prep %>% ggplot(aes(x= LONGITUDE, y=LATITUDE, fill = `prep2019-12-29`) ) + geom_tile()


```



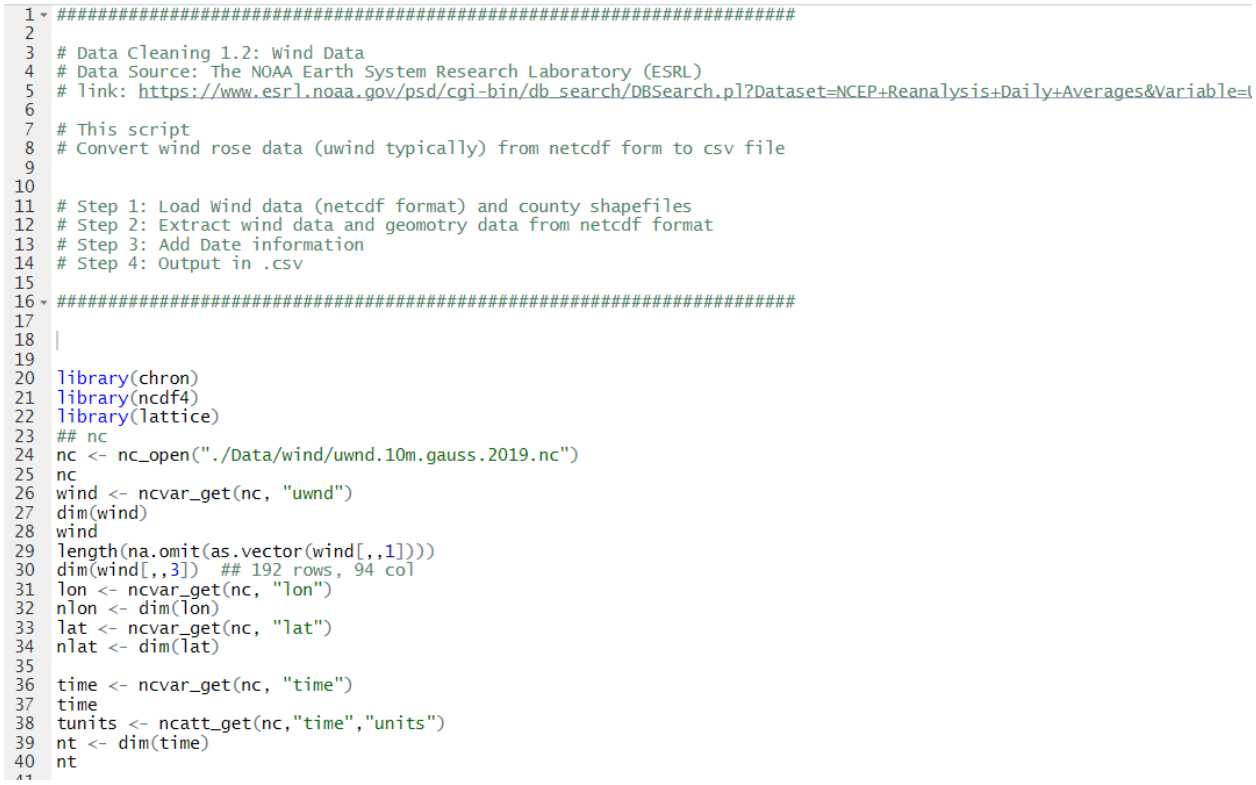
```r

## Temperature

utemp <- read_csv("temp.csv")
temp_base <- utemp[c("lon", "lat")] %>% SpatialPoints() %>% extent() %>%
  raster(nrows = 25, ncols = 25)

startdate <- as.Date("2019-01-01")
a <- utemp %>%
  select(-c("lon", "lat"))
colnames(a) <- seq(startdate, by="1 day", length.out=365)
colnames(a) <- paste("temp", colnames(a), sep = "")
utemp <- cbind(a, utemp[c("lon", "lat")])


cord <- utemp[c("lon", "lat")]
# b <- rasterize(cord, r, uwind$`wind2019-12-31`, fun = mean)
# b.extract <- raster::extract(b, wildfire1[,c("LONGITUDE", "LATITUDE")])


dates_temp <- colnames(a) %>% unique()
dates_temp <- dates_temp[order(dates_temp, decreasing = TRUE)]
temp <- data.frame(matrix(nrow = length(raster.base), ncol = 0))


for (i in seq_along(dates_temp)){
  date <- dates_temp[i]
  b <- rasterize(cord, temp_base, utemp[,i], fun = mean)
  b.extract <- raster::extract(b, locs[,c("LONGITUDE", "LATITUDE")])%>% as.data.frame() %>%
    `colnames<-`(date); temp <- bind_cols(temp, b.extract)
  print(date)
}

temp <- bind_cols(temp, locs) %>%
  mutate_all(~ replace_na(.,0))
# temp %>% ggplot(aes(x= LONGITUDE, y=LATITUDE, fill = `temp2019-12-29`) ) + geom_tile()


## Combine with fire data
fire_climate <- cbind(wildfire1[-366:-367], prep[-366:-367]) %>% cbind(temp[-366:-367] )%>% cbind(wind[-1:-2])


```



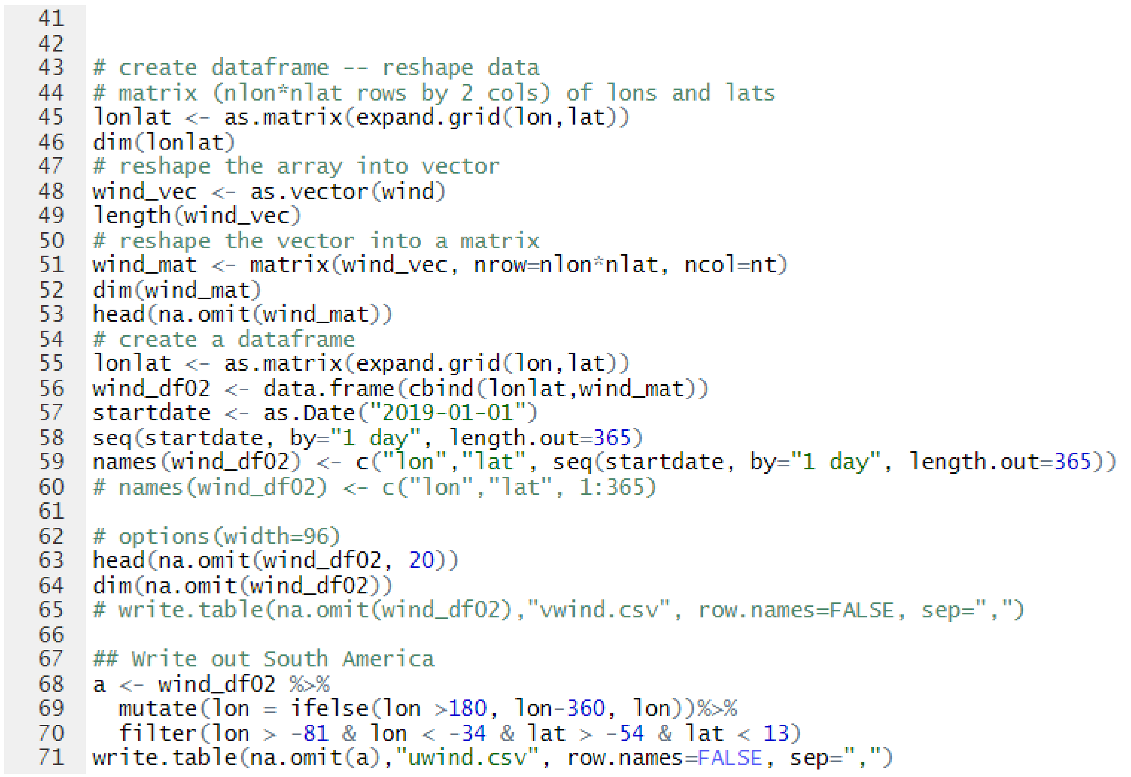
```r

# Remove oceans
# Note: we use the worldclimate data (which only covers lands) to remove grid cells in the sea
# there definitely is a smarter way to do this, but we accidently run into this method and gonna keep it
# World Climate, note: format is RasterStack
wc <- raster::getData('worldclim', res=10, var='bio')
# wc.extract <- raster::extract(wc, wildfire1[,427:428]) ## lon lat col
wc.extract <- raster::extract(wc, fire_climate[,1461:1462])
fire_climate1 <- fire_climate %>% cbind(wc.extract) %>% drop_na()
# fire_climate1 <- fire_climate1 %>% select(-starts_with("bio"))

fire_worldClimate <- cbind(wildfire2[1:22], cluster_result)

wildfire2$x <- wildfire2 %>% raster.base() %>% area()

```



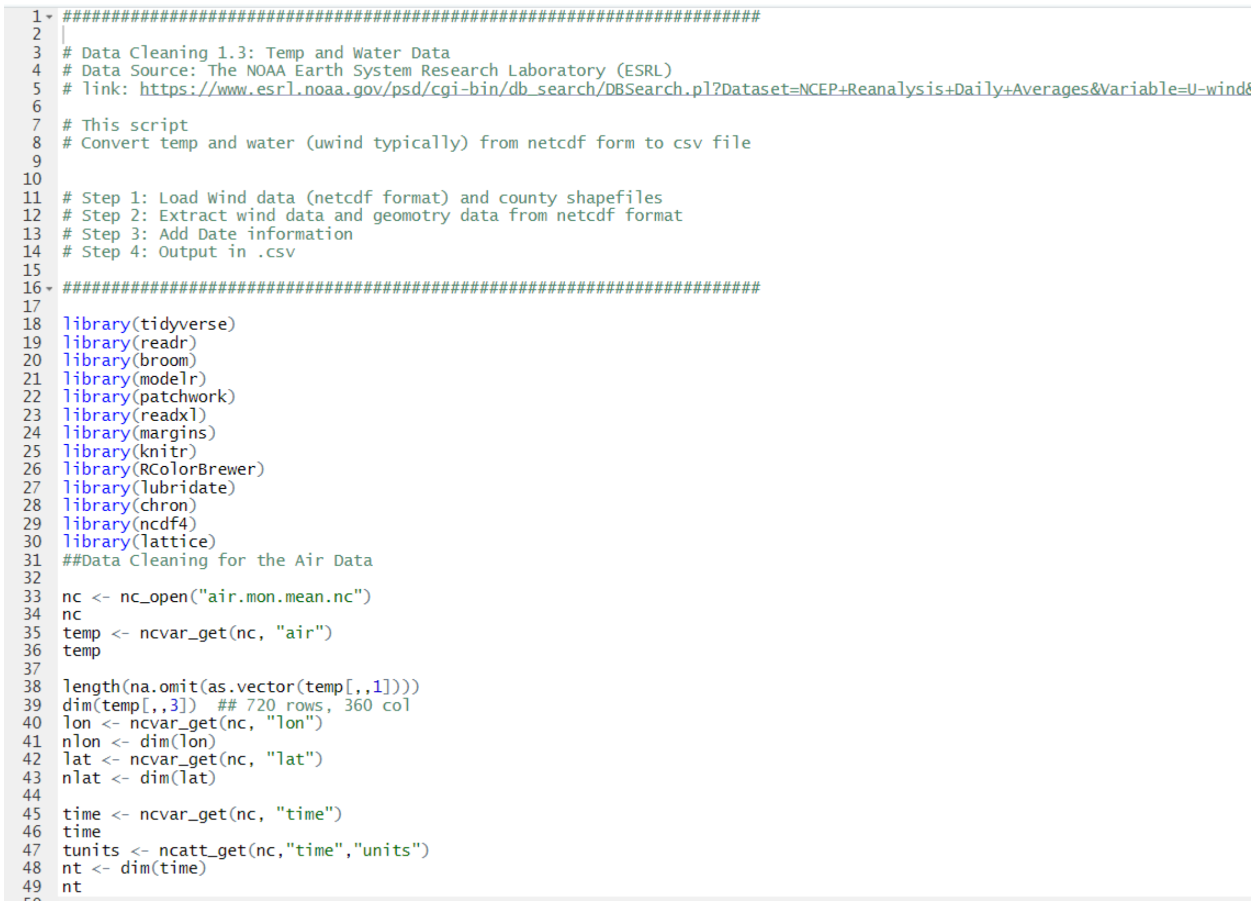
```r


##########################################################
# LASSO feature selection
##########################################################

timecols <- grep("^time", colnames(fire_wind2))
# colnames(fire_wind2)<- c(colnames(fire_wind2)[timecols], paste0("wind", colnames(fire_wind2)[-timecols]))

fw <- wildfire2[-366:-367] ## remove longitude and latitude


colnames(fw) <- gsub("\\-", "", colnames(fw))


set.seed(123)

X <- sparse.model.matrix(`fire20191231` ~ ., data= fw)[,-1] # create sparse matrix
y <- fw$`fire20191231`
n <- length(y)

cv_output <- cv.glmnet(X, y,
                       alpha = 1, nfolds = 10)

# identifying best lamda
best_lam <- cv_output$lambda.1se

lasso_best <- glmnet(X, y, alpha = 1, lambda = best_lam)
glmnet(X, y, alpha = 1, lambda = best_lam)
plot(cv_output)
plot(cv_output$glmnet.fit, xvar = "lambda")


# store the selected coefficients into var
var <- coef(lasso_best)
var1 <- data.frame(summary(var))
var1 <- var1 %>% select(-j) %>%
  rename(`Days_before_20191231` = i,
         `Coef` = x)
write.csv(var1, "lasso_firedate.csv")
```



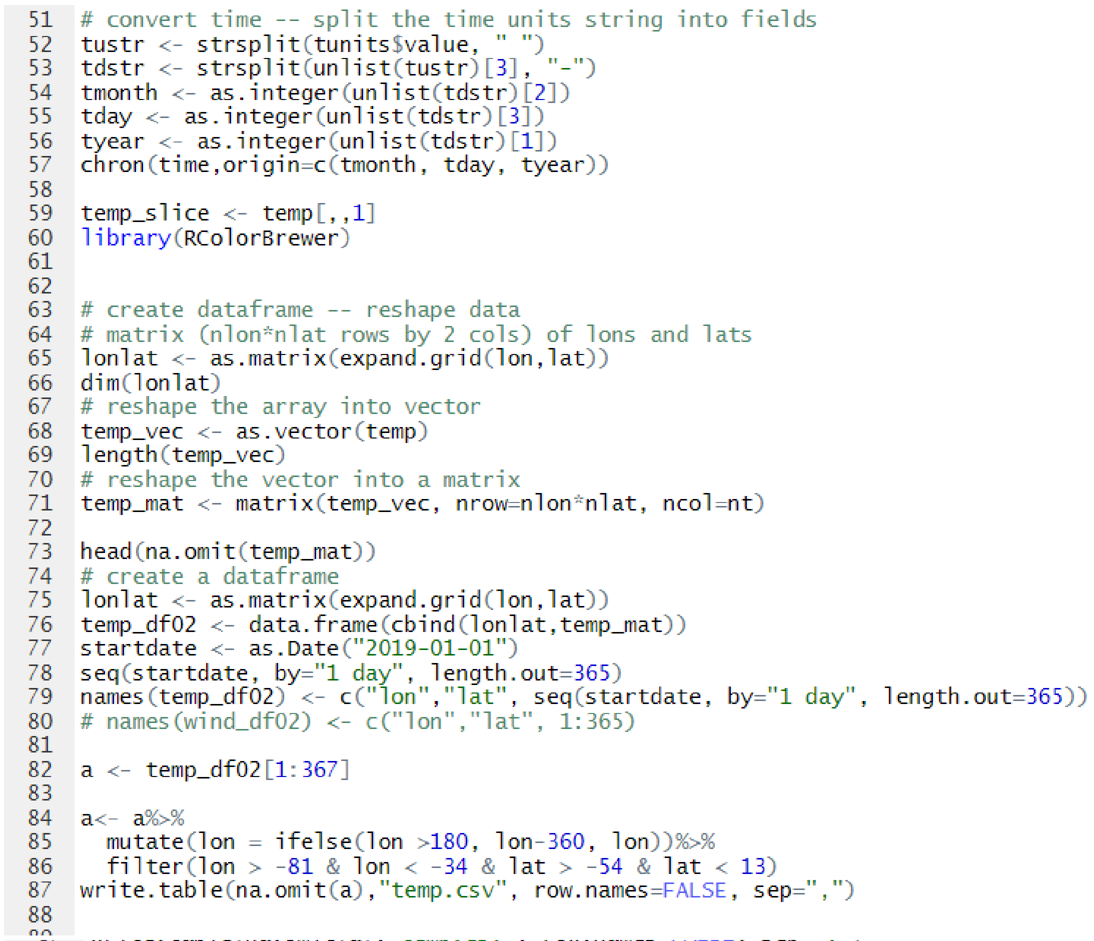
```r
########################################################################

# Data Cleaning 1.2: Wind Data
# Data Source: The NOAA Earth System Research Laboratory (ESRL)
# link: https://www.esrl.noaa.gov/psd/cgi-bin/db_search/DBSearch.pl?Dataset=NCEP+Reanalysis+Daily+Averages&Variable=

# This script
# Convert wind rose data (uwind typically) from netcdf form to csv file


# Step 1: Load Wind data (netcdf format) and county shapefiles
# Step 2: Extract wind data and geomotry data from netcdf format
# Step 3: Add Date information
# Step 4: Output in .csv

########################################################################



library(chron)
library(ncdf4)
library(lattice)
## nc
nc <- nc_open("./Data/wind/uwnd.10m.gauss.2019.nc")
nc
wind <- ncvar_get(nc, "uwnd")
dim(wind)
wind
length(na.omit(as.vector(wind[,,1])))
dim(wind[,,3])  ## 192 rows, 94 col
lon <- ncvar_get(nc, "lon")
nlon <- dim(lon)
lat <- ncvar_get(nc, "lat")
nlat <- dim(lat)

time <- ncvar_get(nc, "time")
time
tunits <- ncatt_get(nc,"time","units")
nt <- dim(time)
nt
```



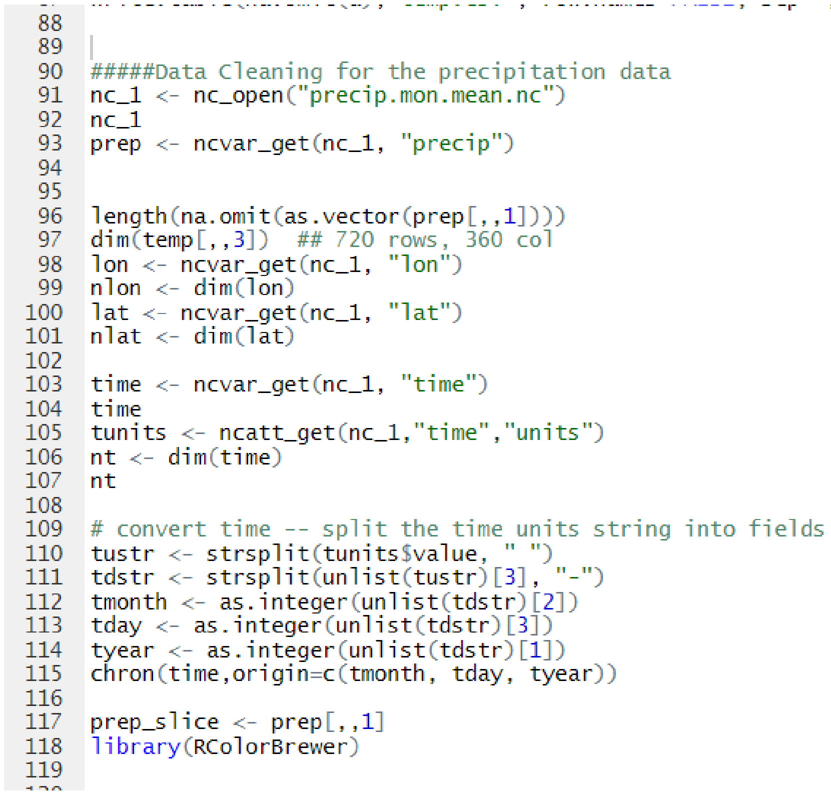
```r


# create dataframe -- reshape data
# matrix (nlon*nlat rows by 2 cols) of lons and lats
lonlat <- as.matrix(expand.grid(lon,lat))
dim(lonlat)
# reshape the array into vector
wind_vec <- as.vector(wind)
length(wind_vec)
# reshape the vector into a matrix
wind_mat <- matrix(wind_vec, nrow=nlon*nlat, ncol=nt)
dim(wind_mat)
head(na.omit(wind_mat))
# create a dataframe
lonlat <- as.matrix(expand.grid(lon,lat))
wind_df02 <- data.frame(cbind(lonlat,wind_mat))
startdate <- as.Date("2019-01-01")
seq(startdate, by="1 day", length.out=365)
names(wind_df02) <- c("lon","lat", seq(startdate, by="1 day", length.out=365))
# names(wind_df02) <- c("lon","lat", 1:365)

# options(width=96)
head(na.omit(wind_df02, 20))
dim(na.omit(wind_df02))
# write.table(na.omit(wind_df02),"vwind.csv", row.names=FALSE, sep=",")

## Write out South America
a <- wind_df02 %>%
  mutate(lon = ifelse(lon >180, lon-360, lon))%>%
  filter(lon > -81 & lon < -34 & lat > -54 & lat < 13)
write.table(na.omit(a),"uwind.csv", row.names=FALSE, sep=",")
```



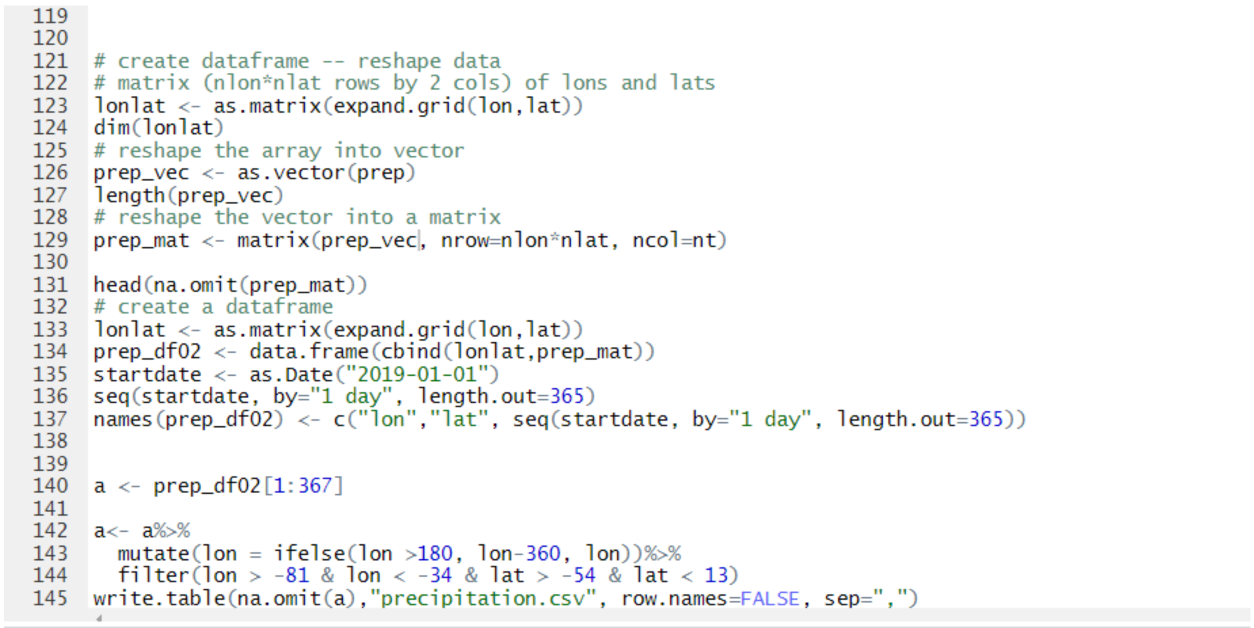
```r
##########################################################################

# Data Cleaning 1.3: Temp and Water Data
# Data Source: The NOAA Earth System Research Laboratory (ESRL)
# link: https://www.esrl.noaa.gov/psd/cgi-bin/db_search/DBSearch.pl?Dataset=NCEP+Reanalysis+Daily+Averages&Variable=U-wind&

# This script
# Convert temp and water (uwind typically) from netcdf form to csv file


# Step 1: Load Wind data (netcdf format) and county shapefiles
# Step 2: Extract wind data and geomotry data from netcdf format
# Step 3: Add Date information
# Step 4: Output in .csv

##########################################################################

library(tidyverse)
library(readr)
library(broom)
library(modelr)
library(patchwork)
library(readxl)
library(margins)
library(knitr)
library(RColorBrewer)
library(lubridate)
library(chron)
library(ncdf4)
library(lattice)
##Data Cleaning for the Air Data

nc <- nc_open("air.mon.mean.nc")
nc
temp <- ncvar_get(nc, "air")
temp

length(na.omit(as.vector(temp[,,1])))
dim(temp[,,3])  ## 720 rows, 360 col
lon <- ncvar_get(nc, "lon")
nlon <- dim(lon)
lat <- ncvar_get(nc, "lat")
nlat <- dim(lat)

time <- ncvar_get(nc, "time")
time
tunits <- ncatt_get(nc,"time","units")
nt <- dim(time)
nt
```



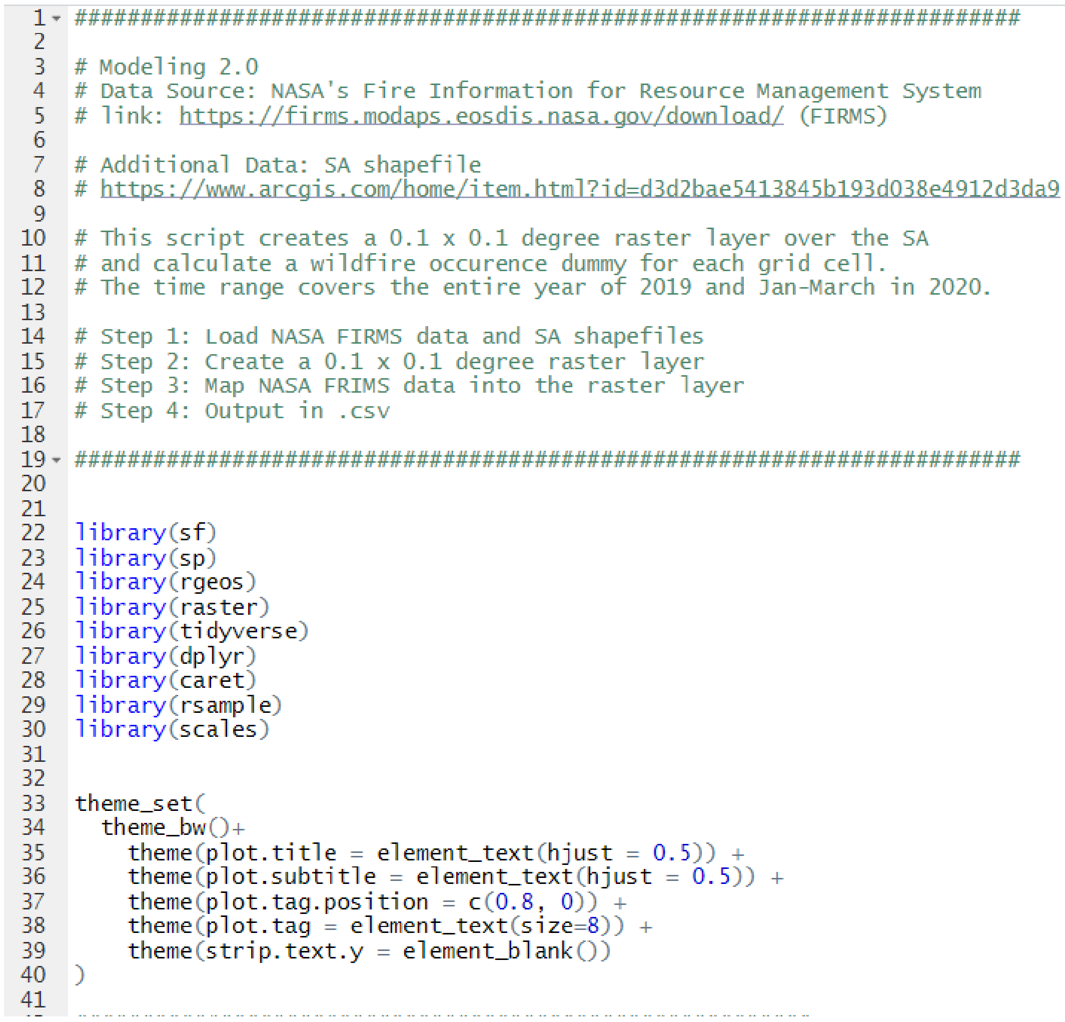
```r
# convert time -- split the time units string into fields
tustr <- strsplit(tunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth <- as.integer(unlist(tdstr)[2])
tday <- as.integer(unlist(tdstr)[3])
tyear <- as.integer(unlist(tdstr)[1])
chron(time,origin=c(tmonth, tday, tyear))

temp_slice <- temp[,,1]
library(RColorBrewer)


# create dataframe -- reshape data
# matrix (nlon*nlat rows by 2 cols) of lons and lats
lonlat <- as.matrix(expand.grid(lon,lat))
dim(lonlat)
# reshape the array into vector
temp_vec <- as.vector(temp)
length(temp_vec)
# reshape the vector into a matrix
temp_mat <- matrix(temp_vec, nrow=nlon*nlat, ncol=nt)

head(na.omit(temp_mat))
# create a dataframe
lonlat <- as.matrix(expand.grid(lon,lat))
temp_df02 <- data.frame(cbind(lonlat,temp_mat))
startdate <- as.Date("2019-01-01")
seq(startdate, by="1 day", length.out=365)
names(temp_df02) <- c("lon","lat", seq(startdate, by="1 day", length.out=365))
# names(wind_df02) <- c("lon","lat", 1:365)

a <- temp_df02[1:367]

a<- a%>%
  mutate(lon = ifelse(lon >180, lon-360, lon))%>%
  filter(lon > -81 & lon < -34 & lat > -54 & lat < 13)
write.table(na.omit(a),"temp.csv", row.names=FALSE, sep=",")

```



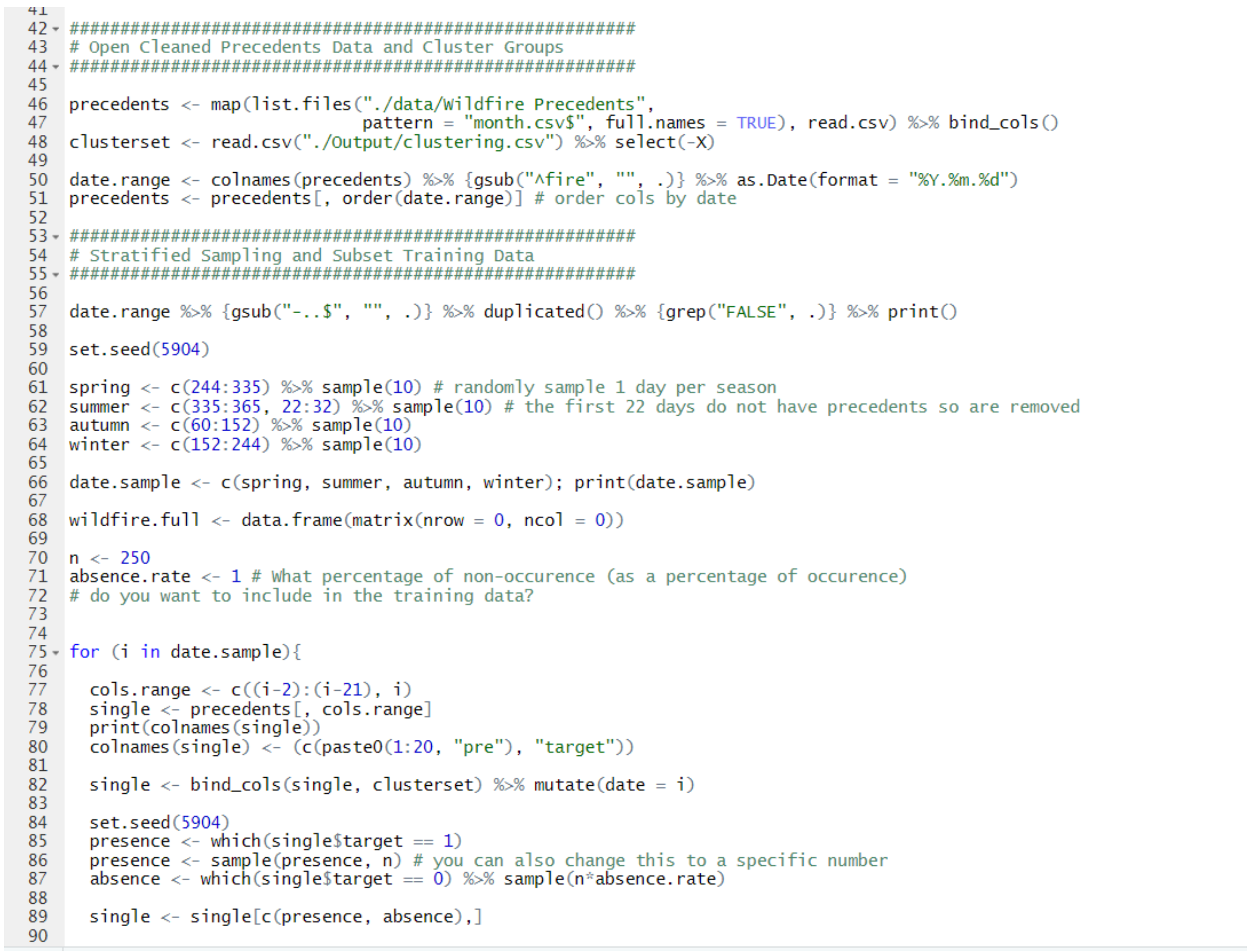
```r

#####Data Cleaning for the precipitation data
nc_1 <- nc_open("precip.mon.mean.nc")
nc_1
prep <- ncvar_get(nc_1, "precip")


length(na.omit(as.vector(prep[,,1])))
dim(temp[,,3])  ## 720 rows, 360 col
lon <- ncvar_get(nc_1, "lon")
nlon <- dim(lon)
lat <- ncvar_get(nc_1, "lat")
nlat <- dim(lat)

time <- ncvar_get(nc_1, "time")
time
tunits <- ncatt_get(nc_1,"time","units")
nt <- dim(time)
nt

# convert time -- split the time units string into fields
tustr <- strsplit(tunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth <- as.integer(unlist(tdstr)[2])
tday <- as.integer(unlist(tdstr)[3])
tyear <- as.integer(unlist(tdstr)[1])
chron(time,origin=c(tmonth, tday, tyear))

prep_slice <- prep[,,1]
library(RColorBrewer)

```



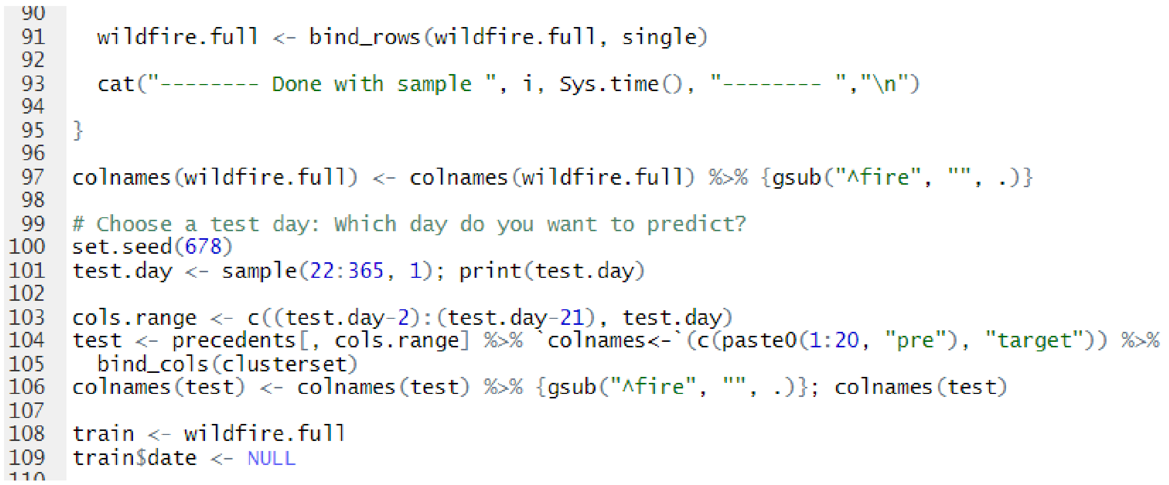
```r


# create dataframe -- reshape data
# matrix (nlon*nlat rows by 2 cols) of lons and lats
lonlat <- as.matrix(expand.grid(lon,lat))
dim(lonlat)
# reshape the array into vector
prep_vec <- as.vector(prep)
length(prep_vec)
# reshape the vector into a matrix
prep_mat <- matrix(prep_vec, nrow=nlon*nlat, ncol=nt)

head(na.omit(prep_mat))
# create a dataframe
lonlat <- as.matrix(expand.grid(lon,lat))
prep_df02 <- data.frame(cbind(lonlat,prep_mat))
startdate <- as.Date("2019-01-01")
seq(startdate, by="1 day", length.out=365)
names(prep_df02) <- c("lon","lat", seq(startdate, by="1 day", length.out=365))


a <- prep_df02[1:367]

a<- a%>%
  mutate(lon = ifelse(lon >180, lon-360, lon))%>%
  filter(lon > -81 & lon < -34 & lat > -54 & lat < 13)
write.table(na.omit(a),"precipitation.csv", row.names=FALSE, sep=",")
```



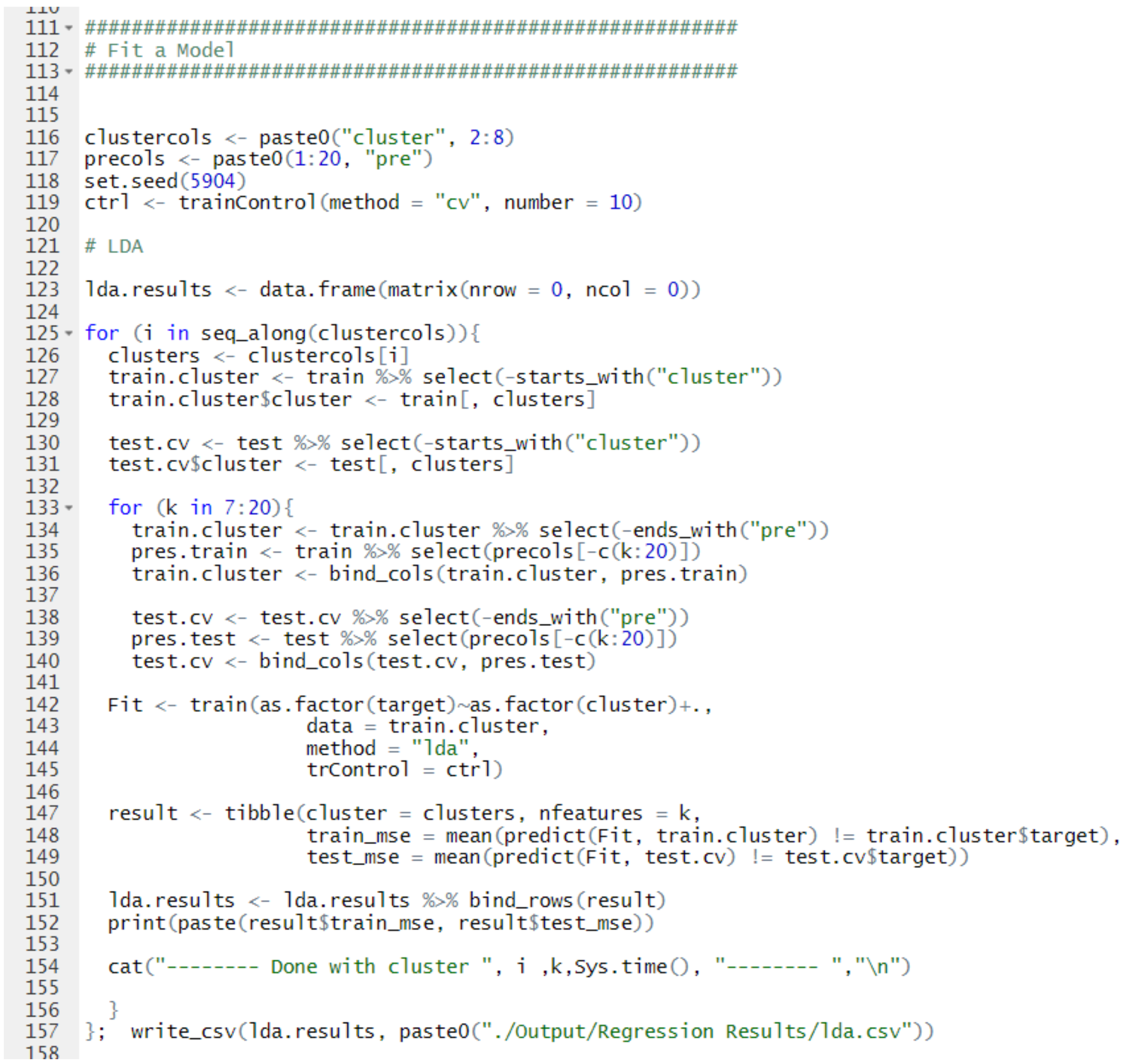
```r
######################################################################

# Modeling 2.0
# Data Source: NASA's Fire Information for Resource Management System
# link: https://firms.modaps.eosdis.nasa.gov/download/ (FIRMS)

# Additional Data: SA shapefile
# https://www.arcgis.com/home/item.html?id=d3d2bae5413845b193d038e4912d3da9

# This script creates a 0.1 x 0.1 degree raster layer over the SA
# and calculate a wildfire occurence dummy for each grid cell.
# The time range covers the entire year of 2019 and Jan-March in 2020.

# Step 1: Load NASA FIRMS data and SA shapefiles
# Step 2: Create a 0.1 x 0.1 degree raster layer
# Step 3: Map NASA FRIMS data into the raster layer
# Step 4: Output in .csv

######################################################################


library(sf)
library(sp)
library(rgeos)
library(raster)
library(tidyverse)
library(dplyr)
library(caret)
library(rsample)
library(scales)


theme_set(
  theme_bw()+
    theme(plot.title = element_text(hjust = 0.5)) +
    theme(plot.subtitle = element_text(hjust = 0.5)) +
    theme(plot.tag.position = c(0.8, 0)) +
    theme(plot.tag = element_text(size=8)) +
    theme(strip.text.y = element_blank())
)

```



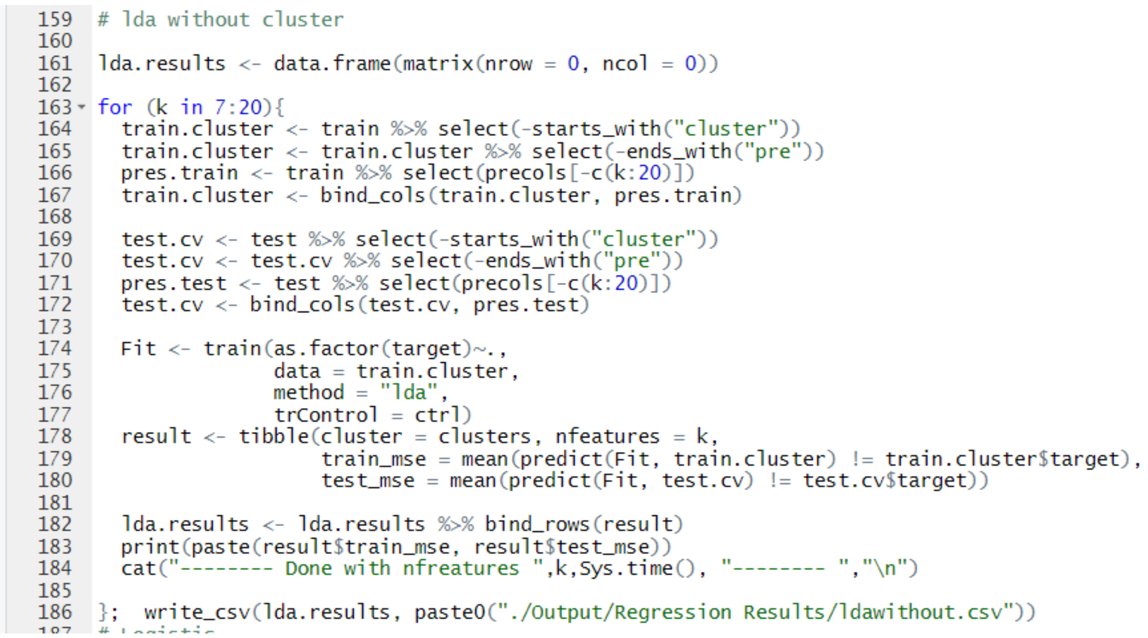
```r
############################################################
# Open Cleaned Precedents Data and Cluster Groups
############################################################

precedents <- map(list.files("./data/Wildfire Precedents",
                             pattern = "month.csv$", full.names = TRUE), read.csv) %>% bind_cols()
clusterset <- read.csv("./Output/clustering.csv") %>% select(-X)

date.range <- colnames(precedents) %>% {gsub("^fire", "", .)} %>% as.Date(format = "%Y.%m.%d")
precedents <- precedents[, order(date.range)] # order cols by date

############################################################
# Stratified Sampling and Subset Training Data
############################################################

date.range %>% {gsub("-..$", "", .)} %>% duplicated() %>% {grep("FALSE", .)} %>% print()

set.seed(5904)

spring <- c(244:335) %>% sample(10) # randomly sample 1 day per season
summer <- c(335:365, 22:32) %>% sample(10) # the first 22 days do not have precedents so are removed
autumn <- c(60:152) %>% sample(10)
winter <- c(152:244) %>% sample(10)

date.sample <- c(spring, summer, autumn, winter); print(date.sample)

wildfire.full <- data.frame(matrix(nrow = 0, ncol = 0))

n <- 250
absence.rate <- 1 # What percentage of non-occurence (as a percentage of occurence)
# do you want to include in the training data?


for (i in date.sample){

  cols.range <- c((i-2):(i-21), i)
  single <- precedents[, cols.range]
  print(colnames(single))
  colnames(single) <- (c(paste0(1:20, "pre"), "target"))

  single <- bind_cols(single, clusterset) %>% mutate(date = i)

  set.seed(5904)
  presence <- which(single$target == 1)
  presence <- sample(presence, n) # you can also change this to a specific number
  absence <- which(single$target == 0) %>% sample(n*absence.rate)

  single <- single[c(presence, absence),]

```



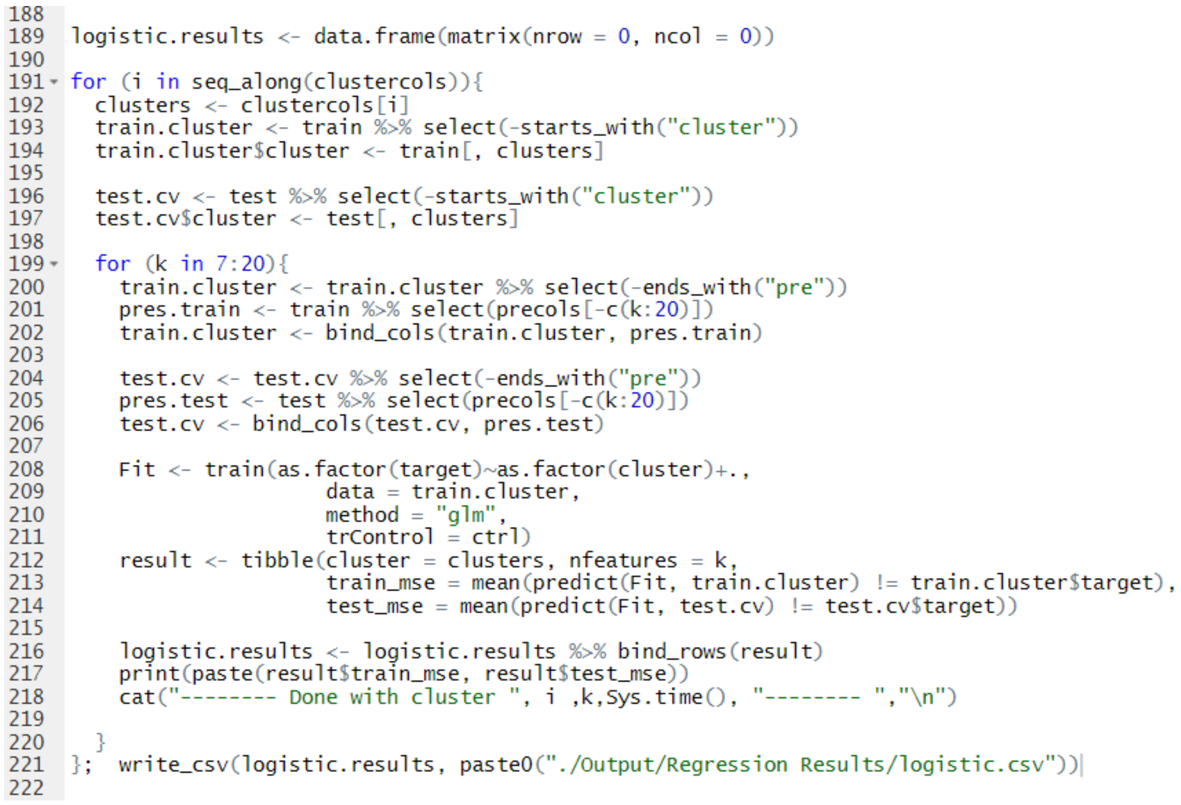
```r

  wildfire.full <- bind_rows(wildfire.full, single)

  cat("-------- Done with sample ", i, Sys.time(), "-------- ","\n")

}

colnames(wildfire.full) <- colnames(wildfire.full) %>% {gsub("^fire", "", .)}

# Choose a test day: Which day do you want to predict?
set.seed(678)
test.day <- sample(22:365, 1); print(test.day)

cols.range <- c((test.day-2):(test.day-21), test.day)
test <- precedents[, cols.range] %>% `colnames<-`(c(paste0(1:20, "pre"), "target")) %>%
  bind_cols(clusterset)
colnames(test) <- colnames(test) %>% {gsub("^fire", "", .)}; colnames(test)

train <- wildfire.full
train$date <- NULL
```



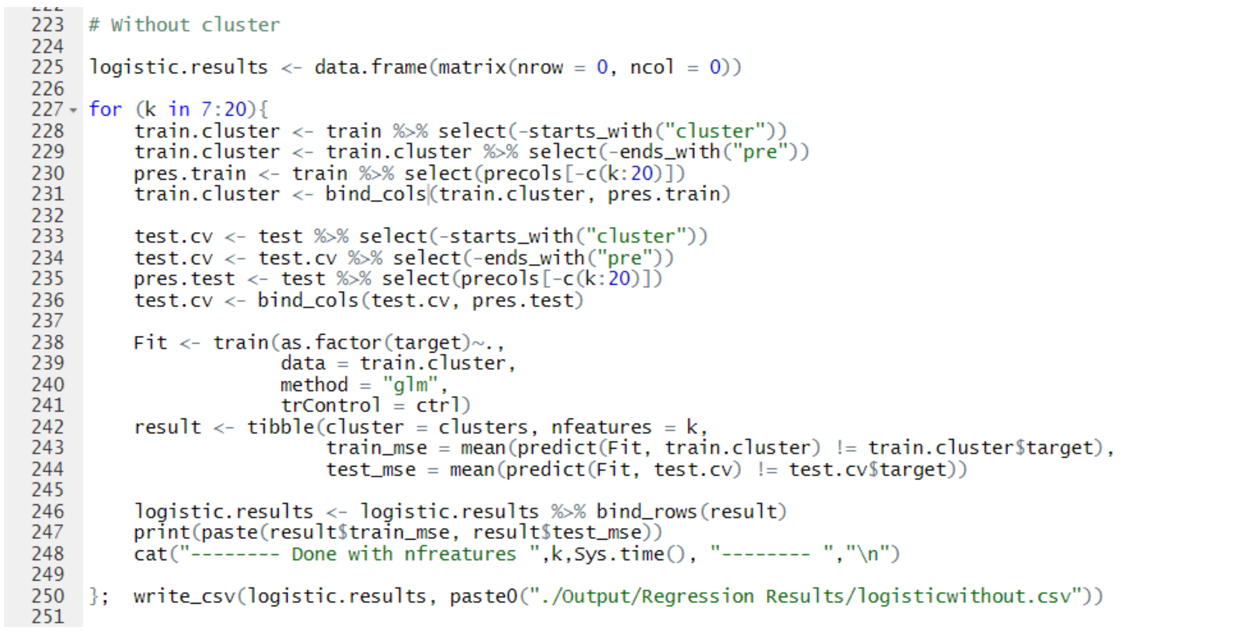
```r
##########################################################
# Fit a Model
##########################################################


clustercols <- paste0("cluster", 2:8)
precols <- paste0(1:20, "pre")
set.seed(5904)
ctrl <- trainControl(method = "cv", number = 10)

# LDA

lda.results <- data.frame(matrix(nrow = 0, ncol = 0))

for (i in seq_along(clustercols)){
  clusters <- clustercols[i]
  train.cluster <- train %>% select(-starts_with("cluster"))
  train.cluster$cluster <- train[, clusters]

  test.cv <- test %>% select(-starts_with("cluster"))
  test.cv$cluster <- test[, clusters]

  for (k in 7:20){
    train.cluster <- train.cluster %>% select(-ends_with("pre"))
    pres.train <- train %>% select(precols[-c(k:20)])
    train.cluster <- bind_cols(train.cluster, pres.train)

    test.cv <- test.cv %>% select(-ends_with("pre"))
    pres.test <- test %>% select(precols[-c(k:20)])
    test.cv <- bind_cols(test.cv, pres.test)

  Fit <- train(as.factor(target)~as.factor(cluster)+.,
                 data = train.cluster,
                 method = "lda",
                 trControl = ctrl)

  result <- tibble(cluster = clusters, nfeatures = k,
                 train_mse = mean(predict(Fit, train.cluster) != train.cluster$target),
                 test_mse = mean(predict(Fit, test.cv) != test.cv$target))

  lda.results <- lda.results %>% bind_rows(result)
  print(paste(result$train_mse, result$test_mse))

  cat("-------- Done with cluster ", i ,k,Sys.time(), "-------- ","\n")

  }
};  write_csv(lda.results, paste0("./Output/Regression Results/lda.csv"))
```



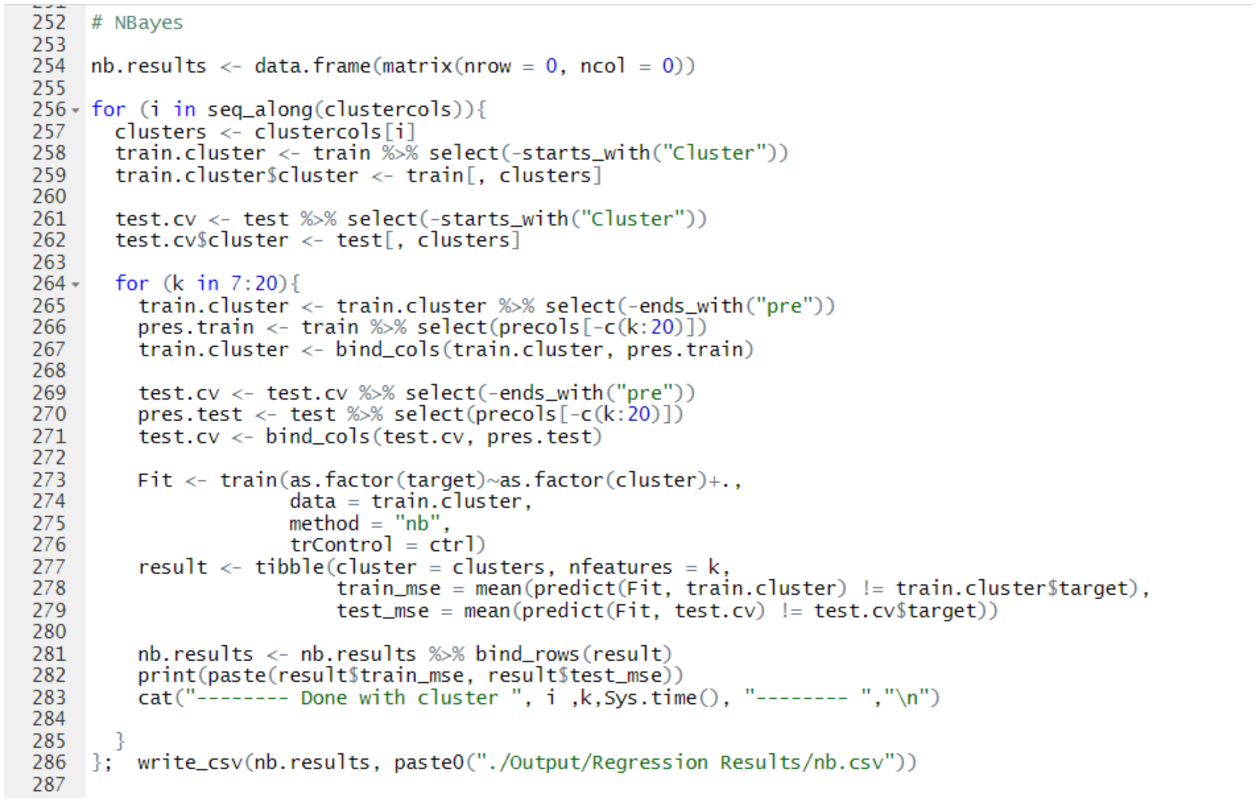
```r
# lda without cluster

lda.results <- data.frame(matrix(nrow = 0, ncol = 0))

for (k in 7:20){
  train.cluster <- train %>% select(-starts_with("cluster"))
  train.cluster <- train.cluster %>% select(-ends_with("pre"))
  pres.train <- train %>% select(precols[-c(k:20)])
  train.cluster <- bind_cols(train.cluster, pres.train)

  test.cv <- test %>% select(-starts_with("cluster"))
  test.cv <- test.cv %>% select(-ends_with("pre"))
  pres.test <- test %>% select(precols[-c(k:20)])
  test.cv <- bind_cols(test.cv, pres.test)

  Fit <- train(as.factor(target)~.,
               data = train.cluster,
               method = "lda",
               trControl = ctrl)
  result <- tibble(cluster = clusters, nfeatures = k,
                   train_mse = mean(predict(Fit, train.cluster) != train.cluster$target),
                   test_mse = mean(predict(Fit, test.cv) != test.cv$target))

  lda.results <- lda.results %>% bind_rows(result)
  print(paste(result$train_mse, result$test_mse))
  cat("-------- Done with nfeatures ",k,Sys.time(), "-------- ","\n")

};  write_csv(lda.results, paste0("./Output/Regression Results/ldawithout.csv"))
```



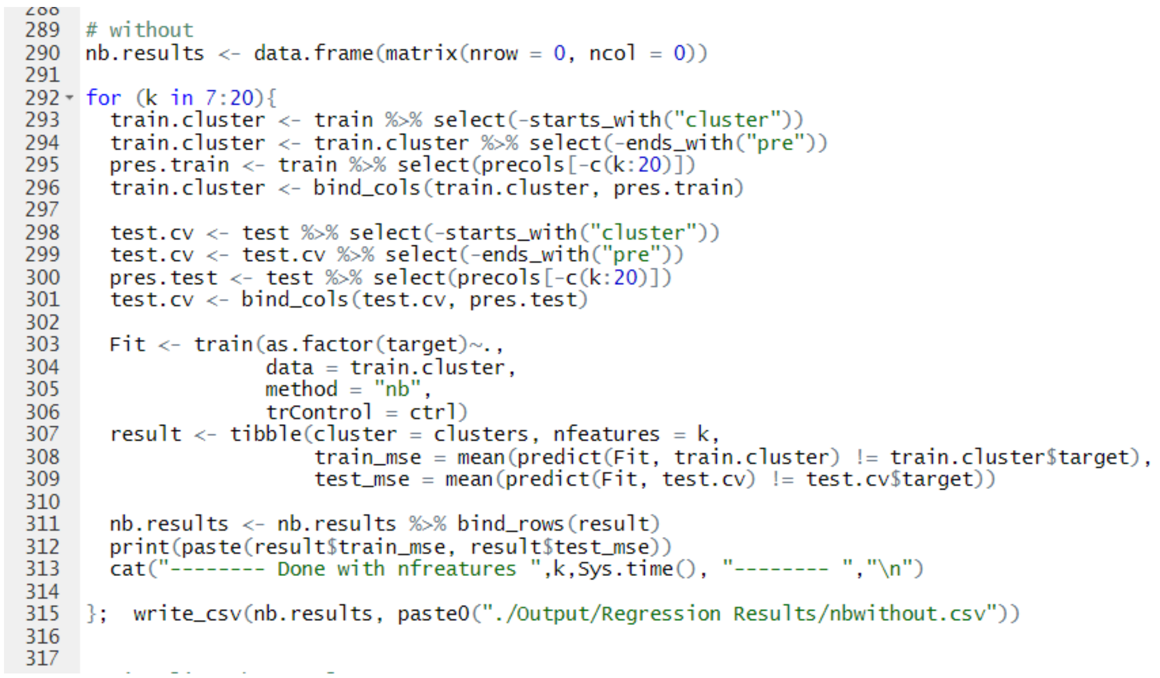
```r

logistic.results <- data.frame(matrix(nrow = 0, ncol = 0))

for (i in seq_along(clustercols)){
  clusters <- clustercols[i]
  train.cluster <- train %>% select(-starts_with("cluster"))
  train.cluster$cluster <- train[, clusters]

  test.cv <- test %>% select(-starts_with("cluster"))
  test.cv$cluster <- test[, clusters]

  for (k in 7:20){
    train.cluster <- train.cluster %>% select(-ends_with("pre"))
    pres.train <- train %>% select(precols[-c(k:20)])
    train.cluster <- bind_cols(train.cluster, pres.train)

    test.cv <- test.cv %>% select(-ends_with("pre"))
    pres.test <- test %>% select(precols[-c(k:20)])
    test.cv <- bind_cols(test.cv, pres.test)

    Fit <- train(as.factor(target)~as.factor(cluster)+.,
                 data = train.cluster,
                 method = "glm",
                 trControl = ctrl)
    result <- tibble(cluster = clusters, nfeatures = k,
                     train_mse = mean(predict(Fit, train.cluster) != train.cluster$target),
                     test_mse = mean(predict(Fit, test.cv) != test.cv$target))

    logistic.results <- logistic.results %>% bind_rows(result)
    print(paste(result$train_mse, result$test_mse))
    cat("-------- Done with cluster ", i ,k,Sys.time(), "-------- ","\n")

  }
};  write_csv(logistic.results, paste0("./Output/Regression Results/logistic.csv"))

```



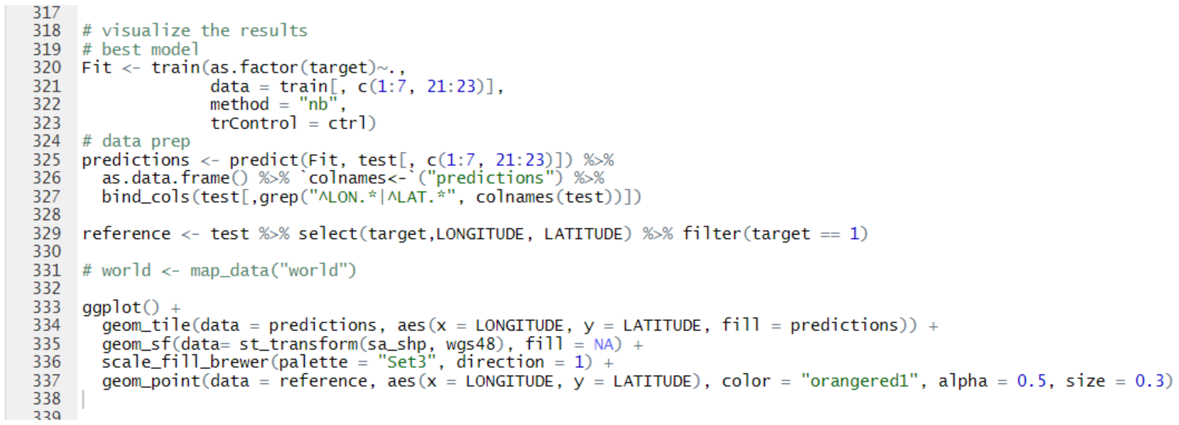
```
# Without cluster

logistic.results <- data.frame(matrix(nrow = 0, ncol = 0))

for (k in 7:20){
    train.cluster <- train %>% select(-starts_with("cluster"))
    train.cluster <- train.cluster %>% select(-ends_with("pre"))
    pres.train <- train %>% select(precols[-c(k:20)])
    train.cluster <- bind_cols(train.cluster, pres.train)

    test.cv <- test %>% select(-starts_with("cluster"))
    test.cv <- test.cv %>% select(-ends_with("pre"))
    pres.test <- test %>% select(precols[-c(k:20)])
    test.cv <- bind_cols(test.cv, pres.test)

    Fit <- train(as.factor(target)~.,
                 data = train.cluster,
                 method = "glm",
                 trControl = ctrl)
    result <- tibble(cluster = clusters, nfeatures = k,
                     train_mse = mean(predict(Fit, train.cluster) != train.cluster$target),
                     test_mse = mean(predict(Fit, test.cv) != test.cv$target))

    logistic.results <- logistic.results %>% bind_rows(result)
    print(paste(result$train_mse, result$test_mse))
    cat("-------- Done with nfreatures ",k,Sys.time(), "-------- ","\n")

};  write_csv(logistic.results, paste0("./Output/Regression Results/logisticwithout.csv"))

```



```
# NBayes

nb.results <- data.frame(matrix(nrow = 0, ncol = 0))

for (i in seq_along(clustercols)){
  clusters <- clustercols[i]
  train.cluster <- train %>% select(-starts_with("Cluster"))
  train.cluster$cluster <- train[, clusters]

  test.cv <- test %>% select(-starts_with("Cluster"))
  test.cv$cluster <- test[, clusters]

  for (k in 7:20){
    train.cluster <- train.cluster %>% select(-ends_with("pre"))
    pres.train <- train %>% select(precols[-c(k:20)])
    train.cluster <- bind_cols(train.cluster, pres.train)

    test.cv <- test.cv %>% select(-ends_with("pre"))
    pres.test <- test %>% select(precols[-c(k:20)])
    test.cv <- bind_cols(test.cv, pres.test)

    Fit <- train(as.factor(target)~as.factor(cluster)+.,
                 data = train.cluster,
                 method = "nb",
                 trControl = ctrl)
    result <- tibble(cluster = clusters, nfeatures = k,
                     train_mse = mean(predict(Fit, train.cluster) != train.cluster$target),
                     test_mse = mean(predict(Fit, test.cv) != test.cv$target))

    nb.results <- nb.results %>% bind_rows(result)
    print(paste(result$train_mse, result$test_mse))
    cat("-------- Done with cluster ", i ,k,Sys.time(), "-------- ","\n")

  }
};  write_csv(nb.results, paste0("./Output/Regression Results/nb.csv"))

```



```r
# without
nb.results <- data.frame(matrix(nrow = 0, ncol = 0))

for (k in 7:20){
  train.cluster <- train %>% select(-starts_with("cluster"))
  train.cluster <- train.cluster %>% select(-ends_with("pre"))
  pres.train <- train %>% select(precols[-c(k:20)])
  train.cluster <- bind_cols(train.cluster, pres.train)

  test.cv <- test %>% select(-starts_with("cluster"))
  test.cv <- test.cv %>% select(-ends_with("pre"))
  pres.test <- test %>% select(precols[-c(k:20)])
  test.cv <- bind_cols(test.cv, pres.test)

  Fit <- train(as.factor(target)~.,
               data = train.cluster,
               method = "nb",
               trControl = ctrl)
  result <- tibble(cluster = clusters, nfeatures = k,
                   train_mse = mean(predict(Fit, train.cluster) != train.cluster$target),
                   test_mse = mean(predict(Fit, test.cv) != test.cv$target))

  nb.results <- nb.results %>% bind_rows(result)
  print(paste(result$train_mse, result$test_mse))
  cat("-------- Done with nfreatures ",k,Sys.time(), "-------- ","\n")

};  write_csv(nb.results, paste0("./Output/Regression Results/nbwithout.csv"))
```



```r
# visualize the results
# best model
Fit <- train(as.factor(target)~.,
             data = train[, c(1:7, 21:23)],
             method = "nb",
             trControl = ctrl)
# data prep
predictions <- predict(Fit, test[, c(1:7, 21:23)]) %>%
  as.data.frame() %>% `colnames<-`("predictions") %>%
  bind_cols(test[,grep("^LON.*|^LAT.*", colnames(test))])

reference <- test %>% select(target,LONGITUDE, LATITUDE) %>% filter(target == 1)

# world <- map_data("world")

ggplot() +
  geom_tile(data = predictions, aes(x = LONGITUDE, y = LATITUDE, fill = predictions)) +
  geom_sf(data= st_transform(sa_shp, wgs48), fill = NA) +
  scale_fill_brewer(palette = "Set3", direction = 1) +
  geom_point(data = reference, aes(x = LONGITUDE, y = LATITUDE), color = "orangered1", alpha = 0.5, size = 0.3)
```