Econ 434 Project

Yixian Chen, Tong He, Yinuo Song

Corona Virus raw data comes from github user 'Our World in Data[i]'. The complete COVID-19 dataset is a collection of the COVID-19 updated data maintained by Our World in Data. To measure the spread of COVID-19, we calculate the number of positive cases, the number of positive cases per million, average increasing rate (valid data), the number of positive cases per million/Duration from the first occurrence, death ratio.

We get temperature data from National Oceanic and Atmospheric Administration[ii] (NOAA). The dataset includes every country's daily average temperature. One '.dly' file represents one country's part of data. So we write a function converting '.dly' file to '.csv' file. Then we concatenate them all according to country code.

Control variables data includes World Development Indicators[iii] (WDI) from The World Bank and Healthcare Access and Quality Index [iv](HAQ) from Our World in Data. WDI data includes more than 1000 attributes. We select 9 variables covering GDP, population, health expenditure, unemployment, urban ratio, etc. We think these economy-related, population-related and healthcare-related data can affect the spread of COVID-19 much.

Then we link these three dataset via country code and country name. After deleting the country data which includes much invalid data, we get the final dataset.

Most countries across the globe have now implemented a suite of strategies to reduce transmission of the novel coronavirus and mitigate the effects of the COVID-19. At the time when the first case was recognized, Wuhan was experiencing its winter season, where the average temperatures can range from 1 to 11 degrees Celsius. As of March 21, China experienced 81,008 registered cases of novel coronavirus with 3,255 deaths. However, for the first time, China is also reporting no new local cases. We conducted this research to explore if climate conditions may influence the spread of COVID-19.

Our model looks like as follows:

$$\mathbf{Y = D\alpha + Z'\beta + \varepsilon,\ E[\varepsilon|D,Z] = 0,}$$

Where Y is the spread of the coronavirus in the country, D is the temperature in the country, and Z is a vector of controls (including a constant). Specifically, we use the total cases of each country to represent Y; We use the Average Temperature-April, Average Temperature-May and Average Temperature-April & May as D; As for the control variables, we choose Death rate, Population density (people per sq. km of land area), GDP per capita (constant 2010 US$), Current health expenditure(% of GDP), Employment to population ratio, 15+, total (%) (modeled ILO estimate), Unemployment, total (% of total labor force) (modeled ILO estimate), Adjusted savings: education expenditure (% of GNI), Urban population (% of total population), People using at least basic sanitation services (% of population), Physicians (per 1,000 people), Haq index: Healthcare access and quality index (Table 1).

Table 1

Y	Total cases
D	Average Temperature of April
	Average Temperature of May
	Average Temperature of April & May
Z	Death rate
	Population density (people per sq. km of land area)
	GDP per capita (constant 2010 US$)
	Current health expenditure(% of GDP)
	Employment to population ratio, 15+, total (%) (modeled ILO estimate)
	Unemployment, total (% of total labor force) (modeled ILO estimate)
	Adjusted savings: education expenditure (% of GNI)
	Urban population (% of total population)
	People using at least basic sanitation services (% of population)
	Physicians (per 1,000 people)
	Haq index: Healthcare access and quality index

At the beginning, we first pre-process the data by replacing NA value with the value of mean and then scale independent variables and control all the variables (X) and process the dependent variables by: (X-mean)/std; (Y-min)/min

Lasso is a "convex relaxation" of this optimization problem:

$$\widehat{\beta}(\lambda) = \arg\min_{b\in\mathbb{R}^p}\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i'b)^2 + \lambda\|b\|_1\right), \text{ where } \|b\|_1 = \sum_{j=1}^{p}|b_j|$$

where λ is the penalty parameter. Lasso means "Least absolute shrinkage and selection operator". It is introduced in Tibsbirani (1996), "Regression shrinkage and selection via the Lasso," Journal of the Royal Statistical Society. Series B.

Step 1: we apply Lasso as a whole and alpha=50, and we get the coefficients as follows (Table2).

Table 2

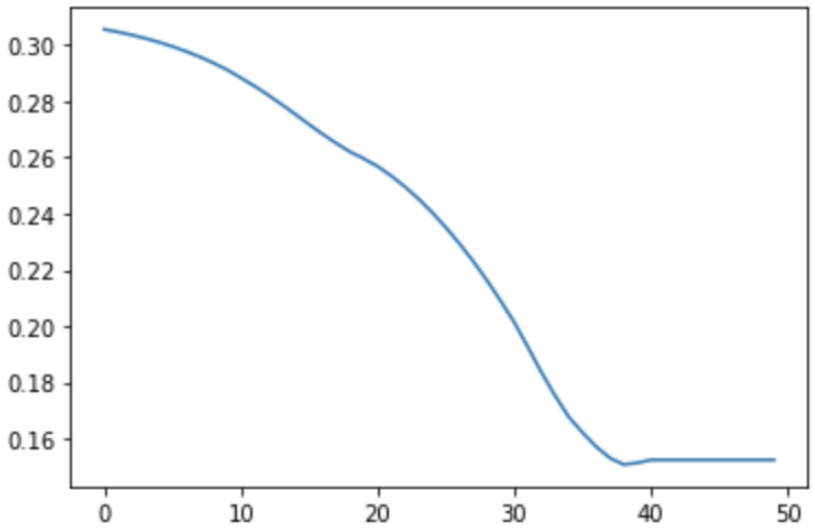
Variables name	Coefficient
Average Temperature-April	-26.81
Average Temperature-May	-0.
Average Temperature-April & May	-0.
Death rate	147.63
Population density	-0.
GDP per capita	170.02
Current health expenditure	-0.
Employment to population ratio	-0.

Unemployment	-0.
Adjusted savings: education expenditure	-0.
Urban population	44.32
People using at least basic sanitation services	-0.
Physicians	-89.99
Haq index	-0.

An older method is sampling splitting: use part of the data for model selection and part for inference. There is an inference-prediction tradeoff: splitting increases the accuracy and robustness of inference but can decrease the accuracy of the predictions.

Step 2: we do the sample splitting to decide alpha and calculate the fit for each value of lambda. We get the plot of residuals as follows (Figure 1):

Figure 1



Step 3: we add 2-degree interactions and plot the residuals again. This time they are the same, and there is no need to add high order interaction anymore (Figure 2).

Figure 2

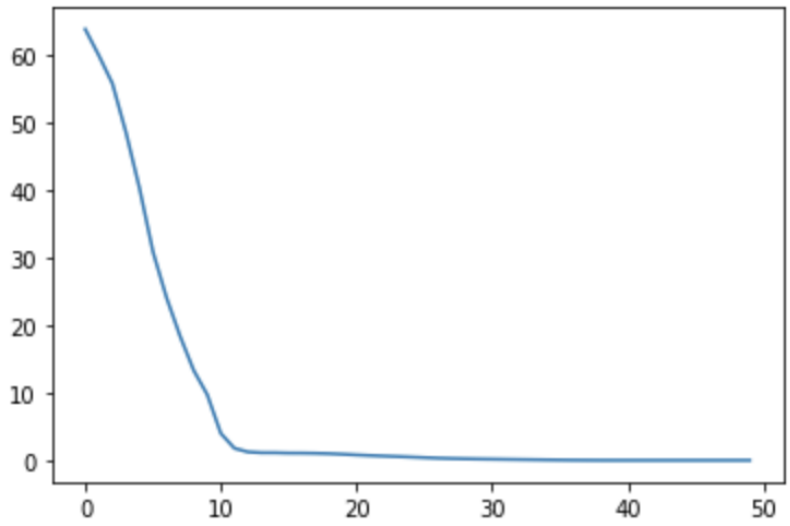


Table 3 shows the coefficient after we add 2-degree interaction. We can see that it kills the variable 'Average Temperature-April & May'. Therefore, sample splitting selects too many variables.

Table 3

Variables name	Coefficient
Average Temperature-April	-979.4213
Average Temperature-May	911.8256
Average Temperature-April & May	-0.
Death rate	200.8928
Population density	17.9588
GDP per capita	209.2183
Current health expenditure	30.2909
Employment to population ratio	986.4219
Unemployment	-1011.7987
Adjusted savings: education expenditure	-23.3993
Urban population	123.6952
People using at least basic sanitation services	-53.7595
Physicians	-181.5253
Haq index	-19.6233

Boosted Regression Tree (BRT) models are a combination of two techniques: decision tree algorithms and boosting methods. Like Random Forest models, BRTs repeatedly fit many decision trees to improve the accuracy of the model. Boosted Regression Trees have two important parameters that need to be specified by the user:

- Tree complexity (tc): this controls the number of splits in each tree. A tc value of 1 results in trees with only 1 split, and means that the model does not take into account interactions between environmental variables. A tc value of 2 results in two splits and so on.

- Learning rate (lr): this determines the contribution of each tree to the growing model. As small value of lr results in many trees to be built.

Step 4: we then use BRT to find lambda, finding that lambda3 equals 656.773266. We can see from Table 4 that BRT kills all the variables.

Table 4

Variables name	Coefficient
Average Temperature-April	-0.
Average Temperature-May	-0.
Average Temperature-April & May	-0.
Death rate	0.
Population density	-0.
GDP per capita	0.
Current health expenditure	-0.
Employment to population ratio	-0.
Unemployment	-0.
Adjusted savings: education expenditure	-0.
Urban population	0.
People using at least basic sanitation services	-0.
Physicians	-0.
Haq index	0.

A better estimator is Double Lasso. This estimator is introduced in Belloni, Chernozhukov, and Hansen (2014). Double Lasso consists of 3 steps:

- run Lasso of Y on D and Z to get $\alpha^{\wedge}$ and $\gamma^{\wedge}$
- run Lasso of D and Z to get $\psi^{\wedge}$
- use analogy principle to estimate α:

$$\tilde{\alpha} = \frac{\sum_{i=1}^{n}(Y_i - Z_i'\hat{\gamma})(D_i - Z_i'\hat{\psi})}{\sum_{i=1}^{n} D_i(D_i - Z_i'\hat{\psi})}$$

We then try to apply double lasso, and first we choose lambda0 = 50. Table 5 shows the related coefficients of lambda at this value.

Table 5

Variables name	Coefficient
Average Temperature-April	-26.8126
Average Temperature-May	-0.
Average Temperature-April & May	-0.
Death rate	147.6259
Population density	-0.
GDP per capita	170.0174
Current health expenditure	-0.
Employment to population ratio	-0.
Unemployment	-0.

Adjusted savings: education expenditure	-0.
Urban population	44.316
People using at least basic sanitation services	-0.
Physicians	-89.9882
Haq index	0.

Then we use BTR to choose lambda which equals: lambda5=639.613192. We can see from Table 6 that this method kills all the control variables.

Table 6

Variables name	Coefficient
Death rate	-0.
Population density	0.
GDP per capita	-0.
Current health expenditure	0.
Employment to population ratio	-0.
Unemployment	-0.
Adjusted savings: education expenditure	0.
Urban population	-0.
People using at least basic sanitation services	-0.
Physicians	-0.
Haq index	-0.

Cross-validation is a data-efficient version of sample splitting K-fold cross-validation consists of three steps:

- split the sample into K subsamples (typically, K = 5 or 10) of similar size;
- for each subsample k = 1,...,K:
  - use subsample k as a validation sample and all other subsamples as a training sample;
  - calculate the fit as in sample-splitting and denote it by Fk (λ)

Next we use cross validation to choose lambda. We can see from Table 7 that this method kills all the control variables too.

Table 7

Variables name	Coefficient
Death rate	-0.
Population density	0.
GDP per capita	-0.
Current health expenditure	0.
Employment to population ratio	-0.
Unemployment	-0.
Adjusted savings: education expenditure	0.
Urban population	-0.
People using at least basic sanitation services	-0.
Physicians	-0.

Haq index	-0.

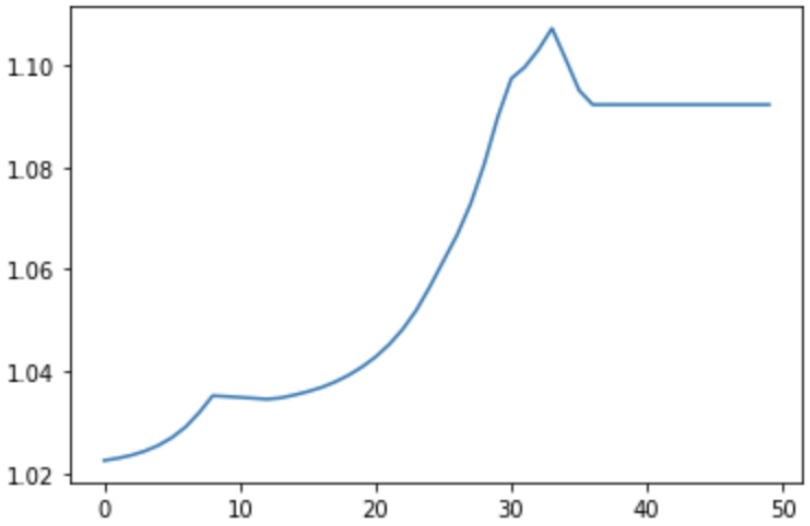
Then we use sample splitting to choose lambda, finding that this method selects all the control variables (Table 8).

Table 8

Variables name	Coefficient
Death rate	0.056
Population density	0.1492
GDP per capita	-0.123
Current health expenditure	0.217
Employment to population ratio	2.1479
Unemployment	-2.1315
Adjusted savings: education expenditure	0.0434
Urban population	0.0197
People using at least basic sanitation services	-0.1411
Physicians	-0.0813
Haq index	-0.0695

Figure 3 shows the plot of the residuals at this time.

Figure 3



We then try to plug in to estimate alpha (coef of 'Temperature in April'), finding that this estimator equals [[-76.53492495]], which is negative. This estimator means that there is negative relationship between the temperature and the spread of COVID-19, indicating that the higher the temperature gets, the less the spread of the COVID-19 will get.

$$\left[\bar{X}_n - \frac{\sigma z_{1-\alpha/2}}{\sqrt{n}}, \bar{X}_n + \frac{\sigma z_{1-\alpha/2}}{\sqrt{n}}\right]$$

is called the $(1 - \alpha) \times 100\%$ asymptotic confidence interval for μ. We also calculate the confidence interval for alpha is (-188.22457602640566, 35.154726126316305) which covers zero.

[i] https://github.com/owid/covid-19-data/tree/master/public/data
[ii] https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/
[iii] https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/
[iv] https://ourworldindata.org/grapher/healthcare-access-and-quality-index

Appendix(Code)

Data Preparation

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import re
import os
```

```
def convert_dly_to_dataframe(dly_filename):
    def build_dly_value_dict():
        dly_value_dict_base={
        "VALUE":[22,26],
        "MFLAG":[27,27],
        "QFLAG":[28,28],
        "SFLAG":[29,29]}
        dly_value_step=8
        dly_value_dict={}
        for day in range(31):
            for k,v in dly_value_dict_base.items():
                dly_value_dict[k+str(day+1)]=
[v[0]+dly_value_step*day,v[1]+dly_value_step*day]
        return dly_value_dict

    dly_dict={
        "ID": [1,11],
        "YEAR":[12,15],
        "MONTH": [16,17],
        "ELEMENT":[18,21],}
    dly_dict.update(build_dly_value_dict())

    columns_value=["VALUE"+str(day+1) for day in range(31)]
    columns=["YEAR","MONTH","ELEMENT"]+columns_value

    def extract_data_from_line(string,loc,astype=str):
        return astype(string[loc[0]-1:loc[1]])

    data=[]
    for line in open(dly_filename):
        data_item=[]
        for k in columns:
            data_item.append(extract_data_from_line(line,dly_dict[k]))
        data.append(data_item)
    df=(pd.DataFrame(data,columns=columns)
         .apply(pd.to_numeric, errors='ignore')
         .replace(-9999,np.nan))
    return df
```

```
# get all file name
filePath = 'C:\\Users\\chenyixian\\Desktop\\434\\project1\\data-meta-temp\\'
file_dir = os.listdir(filePath)

# get country name & code
name_code = pd.read_csv('country-name.csv')
```

```
import datetime
starttime = datetime.datetime.now()
df_final = pd.DataFrame()
for file in file_dir:
    df_temp = convert_dly_to_dataframe(file)
# get country name
    file_code = file[0:2]
    file_name = name_code[name_code['CODE'] == file_code].iat[0, 1]
    df_temp_2020 = df_temp[df_temp['YEAR']==2020]
    df_temp_2020['COUNTRY'] = file_name
#     df_temp_2020['COUNTRY'] = file_name
    df_final = pd.concat([df_final,df_temp_2020])
endtime = datetime.datetime.now()
print(endtime - starttime)
```

```
F:\Anaconda\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

```
0:03:30.145297
```

```
df_final.to_csv("2020temperature_country.csv")
```

Model

```
# %load "code-434-proj1.py"
"""
Created on Sat May 23 10:02:55 2020

@author: heton
"""
import numpy as np
import pandas as pd
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import Imputer
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
data = pd.read_csv("D:/spring/434/proj/data final version.csv")
data = data.set_index(['Country'])
D = data[['Average Temperature-April','Average Temperature-May','Average
Temperature-April&May']]
Y = data[['total_cases']]
Z =data[['Death rate','Population density (people per sq. km of land area)','GDP
per capita (constant 2010 US$)','Current health expenditure(% of
GDP)','Employment to population ratio, 15+, total (%) (modeled ILO
estimate)','Unemployment, total (% of total labor force) (modeled ILO
estimate)','Adjusted savings: education expenditure (% of GNI)','Urban
population (% of total population)','People using at least basic sanitation
services (% of population)','Physicians (per 1,000 people)','Haq index']]
X = np.concatenate((D,Z),axis=1)

#replace NA with mean
nan_model=Imputer(missing_values='NaN',strategy='mean',axis=1)
X0=nan_model.fit_transform(X)
#scale X : (X-mean)/std
muhat = np.mean(X0, axis = 0)
stdhat = np.std(X0, axis = 0)
Xtilde = (X0-muhat)/stdhat
#(Y-min)/min
minhat = np.min(Y, axis = 0)
Y0 = (Y-minhat)/minhat


D = Xtilde[:,0:3]
Z =Xtilde[:,3:15]


#1 Lasso as a whole, alpha=50
lambda0=50
lasso0=Lasso(alpha = lambda0)
lasso0.fit(Xtilde,Y0)
coef0 = lasso0.coef_
print(np.around(coef0,2))

#2 sample splitting to decide alpha

#(Y-min)/std
minhat = np.min(Y, axis = 0)
stdhatY = np.std(Y, axis = 0)
Y1 = (Y-minhat)/stdhatY

(Xt,Xv,Yt,Yv) = train_test_split(Xtilde,Y1,test_size = 0.25,random_state = 0)
 #calculate the fit for each value of lambda

Yv=Yv.to_numpy()
alpha_grid = np.geomspace(0.001,1)
res = np.zeros(alpha_grid.size)
for i in range(alpha_grid.size):
     lasso1 = Lasso(alpha=alpha_grid[i])
     lasso1.fit(Xt,Yt)
     Y_pred= lasso1.predict(Xv)
     res[i]=np.mean((Yv-Y_pred)**2)
plt.plot(res)
lambda1 = alpha_grid[38]
```

```
#3 add 2-degree interations
plo_int=PolynomialFeatures(degree=2,include_bias=False)
Xnew=plo_int.fit_transform(Xtilde)
muhat = np.mean(Xnew, axis = 0)
stdhat = np.std(Xnew, axis = 0)
Xn = (Xnew-muhat)/stdhat
(Xt,Xv,Yt,Yv) = train_test_split(Xn,Y1,test_size = 0.25,random_state = 0)
Yv=Yv.to_numpy()
alpha_grid = np.geomspace(0.001,1)
res = np.zeros(alpha_grid.size)
for i in range(alpha_grid.size):
     lasso2 = Lasso(alpha=alpha_grid[i],max_iter=1000)
     lasso2.fit(Xt,Yt)
     Y_pred= lasso2.predict(Xv)
     res[i]=np.mean((Yv-Y_pred)**2)
plt.plot(res)
lambda2 = alpha_grid[38]
#they are the same, no need to add high order interaction

lasso2 = Lasso(lambda2,max_iter=10000)
lasso2.fit(Xtilde,Y0)
coef2 = lasso2.coef_
print(np.around(coef2,4))

#killed 'Average Temperature-April&May',sample splitting select too many
variables


#4. use BRT to find lambda
sigma = np.std(Y0)
(n,p)=Xtilde.shape
c=1.1
a=0.05
lambda3 = 2*c*sigma*norm.ppf(1-a/(2*p))/np.sqrt(n)
print(lambda3)
lasso3 = Lasso(lambda3.item(),max_iter=10000)
lasso3.fit(Xtilde,Y0)
coef3 = lasso3.coef_
print(np.around(coef3,4))

#BRT killed all variables

#Double lasso
 #1.Y~D,Z
# choose lambda0 = 50

lasso4 = Lasso(alpha=lambda0)
lasso4 =lasso4.fit(Xtilde,Y0)
coef4 = lasso4.coef_
print(np.around(coef4,4))
alphahat=coef4[0:3]
gamahat=coef4[3:15]

 #2.D~Z
 #(1)use BTR to choose lambda
sigma5 = np.std(D[:,0])
```

```
(n,p)=Z.shape
c=1.1
a=0.05
lambda5 = 2*c*sigma*norm.ppf(1-a/(2*p))/np.sqrt(n)
print(lambda5)
lasso5 = Lasso(alpha=lambda5.item())
lasso5 =lasso5.fit(Z,D[:,0])
coef5 = lasso5.coef_
print(np.around(coef5,4))
#killed all Z

 #(2)use cross validation to choose lambda
lasso5=LassoCV(cv=5)
lasso5 =lasso5.fit(Z,D[:,0])
coef5 = lasso5.coef_
print(np.around(coef5,4))
#killed all Z

 #(3)use sample splitting to choose lambda
(Zt,Zv,Dt,Dv) = train_test_split(Z,D,test_size = 0.25,random_state = 0)
#Dv=Dv.to_numpy()
alpha_grid = np.geomspace(0.001,1)
res5 = np.zeros(alpha_grid.size)
for i in range(alpha_grid.size):
     lasso5 = Lasso(alpha=alpha_grid[i],max_iter=1000)
     lasso5.fit(Z,D[:,0])
     D_pred= lasso5.predict(Zv)
     res5[i]=np.mean((Dv[:,0]-D_pred)**2)
plt.plot(res5)
lambda5 = alpha_grid[0]
lasso5 = Lasso(alpha=lambda5.item())
lasso5 =lasso5.fit(Z,D[:,0])
coef5 = lasso5.coef_
print(np.around(coef5,4))
#SELECT ALL Z

phihat= coef5
#plug in to estimate alpha(coef of 'temperature in April')

Y0=Y0.to_numpy()
a=np.dot(np.transpose(Y0-np.dot(Z,gamahat).reshape(n,1)),(D[:,0]-
np.dot(Z,phihat)).reshape(n,1))
b=np.dot(D[:,0],(D[:,0]-np.dot(Z,phihat)).reshape(n,1))
alphahat=a/b
print(alphahat)
#which is negative, make sense

#CI of alphahat

sigma_sqaure=np.array((Y0-np.dot(Z,gamahat).reshape(n,1)-
np.dot(D,alphahat).reshape(n,1))**2)
v_sqaure=(D[:,0]-np.dot(Z,phihat)).reshape(n,1)**2
Vhat=np.mean(sigma_sqaure*v_sqaure)/(np.mean(v_sqaure)**2)

#CI:
CI_L=alphahat[0]-np.sqrt(Vhat)*1.96/np.sqrt(n).astype("float64")
CI_U=alphahat[0]+np.sqrt(Vhat)*1.96/np.sqrt(n).astype("float64")
print("confidence interval for alpha is (",CI_L,",",CI_U,")")
```

```
#zero is in the interval, the effect is not clear
```