Investigating relationship between covid-19 and standards for happiness and freedom.

In this analysis we will take a look at how happiness and freedom has affected the covid-19 situation in different countries

Processing and Cleaning

Step I: Initial loading and processing of Covid 19 data

We use the data from Johns Hopkins University. This database contains COVID case numbers by country.

The next task is to load the data from URL into MATLAB. Then data preprocessing steps are taken (column name and type, specifying variable property...) to prepare the data for calculation.

```
% Do some housekeeping
clc
clear
close all
```

a) Processing the confirmed time series data

```

fileName = [tempdir 'time_series_covid19_confirmed_global.csv'];
url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse
fileName = websave(fileName, url);
opts = detectImportOptions(fileName); % Detect import parameters

% Fix range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Fix column names and types for first columns
opts.VariableNames(1:4) = [{'ProvinceState'}, {'CountryRegion'}, {'Lat'}, {'Long'}];
opts.VariableTypes(1:4) = [{'string'}, {'string'}, {'double'}, {'double'}];

% Fix file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Fix variable properties
opts = setvaropts(opts, "ProvinceState", "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["ProvinceState", "CountryRegion"], "EmptyFieldRule", "auto");

% Import the data
data_confirmed = readtable(fileName, opts);

% Clear temp variable, opts
clear opts url fileName;
```

b) Processing the death time series data

```
fileName = [tempdir 'time_series_covid19_deaths_global.csv'];
```

```
url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse
fileName = websave(fileName, url);
opts = detectImportOptions(fileName); % Detect import parameters

% Fix range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Fix column names and types for first columns
opts.VariableNames(1:4) = [{'ProvinceState'}, {'CountryRegion'}, {'Lat'}, {'Long'}];
opts.VariableTypes(1:4) = [{'string'}, {'string'}, {'double'}, {'double'}];

% Fix file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Fix variable properties
opts = setvaropts(opts, "ProvinceState", "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["ProvinceState", "CountryRegion"], "EmptyFieldRule", "auto");

% Import the data
data_deaths = readtable(fileName, opts);

% Clear temp variable, opts
clear opts url fileName;
```

c) Processing the recovered time series data

```
fileName = [tempdir 'time_series_covid19_recovered_global.csv'];
url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse
fileName = websave(fileName, url);
opts = detectImportOptions(fileName); % Detect import parameters

% Fix range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Fix column names and types for first columns
opts.VariableNames(1:4) = [{'ProvinceState'}, {'CountryRegion'}, {'Lat'}, {'Long'}];
opts.VariableTypes(1:4) = [{'string'}, {'string'}, {'double'}, {'double'}];

% Fix file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Fix variable properties
opts = setvaropts(opts, "ProvinceState", "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["ProvinceState", "CountryRegion"], "EmptyFieldRule", "auto");

% Import the data
data_recovered = readtable(fileName, opts);

% Clear temp variable, opts
```

```
clear opts url fileName;
```

b) Perform moving average and calculate various parameters

We cumulated the country data and calculated the moving median (7 days).

i) Calculation of average infection rate

```
avg = 7; % Days to average
c = size(data_confirmed, 2); % Number of columns
ds = c-4; % Number of days with data (first four columns give different information)
t0 = datetime('22/1/2020'); % Date of first data
lastDate = t0 + days(ds-1); % Date of last data

% Prepare variables for cumulate country data
cats = unique(data_confirmed.CountryRegion); % Countries
lc = length(cats); % Number of countries
data2_confirmed = data_confirmed; % Copy data to force same structure for both dataset
data2_confirmed(lc+1:end, :) = []; % Remove unnecessary rows

% Cumulate country data
for i = 1:lc
    tmp1 = mean(data_confirmed{data_confirmed.CountryRegion == cats(i), 3:4}, 1); % take mean c
    tmp2 = sum(data_confirmed{data_confirmed.CountryRegion == cats(i), 5:end}, 1); % take sum c
    data2_confirmed(i, 1:2) = table("", cats(i)); % Assign country to first columns
    data2_confirmed(i, 3:end) = array2table([tmp1 tmp2]); % Assign mean of coordiantes and sum
end
clear tmp1 tmp2

data_covid = data2_confirmed(:,2);
%detect the first infection for each country
% Smoothing and daily differences data (7-day moving median)
data  = data2_confirmed{:,5:c};
data = movmedian(data,avg,2,'Endpoints','discard');
data_diff = diff(data,[],2);
data_confirmed_series = [array2table(data) array2table(data_diff)];

[m,n] = find(data_diff);
firstIndex = accumarray(m,n,[size(data_diff,1),1],@min,size(data_diff,2));
for i = 1:length(firstIndex)
    data_covid.Average_Infection_Rate(i) = mean(data_diff(i,firstIndex(i):end));
end
```

ii) Calculate of average mortality rate

```
c = size(data_deaths, 2); % Number of columns
ds = c-4; % Number of days with data (first four columns give different information)
t0 = datetime('22/1/2020'); % Date of first data
lastDate = t0 + days(ds-1); % Date of last data
```

```
% Prepare variables for cumulate country data
cats = unique(data_deaths.CountryRegion); % Countries
lc = length(cats); % Number of countries
data2_deaths = data_deaths; % Copy data to force same structure for both dataset
data2_deaths(lc+1:end, :) = []; % Remove unnecessary rows

% Cumulate country data
for i = 1:lc
    tmp1 = mean(data_deaths{data_deaths.CountryRegion == cats(i), 3:4}, 1); % take mean of the
    tmp2 = sum(data_deaths{data_deaths.CountryRegion == cats(i), 5:end}, 1); % take sum of the
    data2_deaths(i, 1:2) = table("", cats(i)); % Assign country to first columns
    data2_deaths(i, 3:end) = array2table([tmp1 tmp2]); % Assign mean of coordiantes and sum of
end
clear tmp1 tmp2

%detect the first infection for each country
% Smoothing and daily differences data (7-day moving median)
data  = data2_deaths{:,5:c};
data = movmedian(data,avg,2,'Endpoints','discard');
data_diff = diff(data,[],2);
data_deaths_series = [array2table(data) array2table(data_diff)];
[m,n] = find(data_diff);
firstIndex = accumarray(m,n,[size(data_diff,1),1],@min,size(data_diff,2));
for i = 1:length(firstIndex)
    data_covid.Average_Mortality_Rate(i) = mean(data_diff(i,firstIndex(i):end));
end
```

iii) Calculate of average recovery rate

```
c = size(data_recovered, 2); % Number of columns
ds = c-4; % Number of days with data (first four columns give different information)
t0 = datetime('22/1/2020'); % Date of first data
lastDate = t0 + days(ds-1); % Date of last data

% Prepare variables for cumulate country data
cats = unique(data_recovered.CountryRegion); % Countries
lc = length(cats); % Number of countries
data2_recovered = data_recovered; % Copy data to force same structure for both dataset
data2_recovered(lc+1:end, :) = []; % Remove unnecessary rows

% Cumulate country data
for i = 1:lc
    tmp1 = mean(data_recovered{data_recovered.CountryRegion ...
        == cats(i), 3:4}, 1); % take mean of the coordinates
    tmp2 = sum(data_recovered{data_recovered.CountryRegion ...
        == cats(i), 5:end}, 1); % take sum of the data
    data2_recovered(i, 1:2) = table("", cats(i)); % Assign country to first columns
    data2_recovered(i, 3:end) = array2table([tmp1 tmp2]); % Assign mean of coordiantes and sum
end
clear tmp1 tmp2

%detect the first infection for each country
% Smoothing and daily differences data (7-day moving median)
```

```
data  = data2_recovered{:,5:c};
data = movmedian(data,avg,2,'Endpoints','discard');
data_diff = diff(data,[],2);
data_recovered_series = [array2table(data) array2table(data_diff)];
[m,n] = find(data_diff);
firstIndex = accumarray(m,n,[size(data_diff,1),1] ...
    ,@min,size(data_diff,2));
for i = 1:length(firstIndex)
    data_covid.Average_Recovery_Rate(i) = mean(data_diff ...
        (i,firstIndex(i):end));
end
```

iv) Calculate of average prevalence rate

```
% % Smoothing and daily differences data (7-day moving median)
%
% data  = data2_confirmed{:,5:c} - (data2_deaths{:,5:c} + data2_recovered{:,5:c});
% data = movmedian(data,avg,2,'Endpoints','discard');
% data_diff = diff(data,[],2);
% data_prevalence_series = [array2table(data) array2table(data_diff)];
% [m,n] = find(data_diff);
% firstIndex = accumarray(m,n,[size(data_diff,1),1],@min,size(data_diff,2));
% for i = 1:length(firstIndex)
%     data_covid.Average_Prevalence_Rate(i) = mean(data_diff(i,firstIndex(i):end));
% end
```

Plotting the trajectory

```
total_days = ds-avg+1;
series_type = {data_confirmed_series data_deaths_series data_recovered_series};
display_name = {'Incidence','Deaths','Recoveries'};
display_name1 = {'Average Infection Rate','Average Mortality Rate','Average Recovery Rate'};
color = {'blue','red','green','cyan'};
xlabels = {'Total Incidence','Total Deaths ','Total Recovery'};
India_rate = data_covid{79,{'Average_Infection_Rate','Average_Mortality_Rate','Average_Recovery
India_rate = repmat(India_rate,[total_days,1]);
pak_rate = data_covid{128,{'Average_Infection_Rate','Average_Mortality_Rate','Average_Recovery_
pak_rate = repmat(pak_rate,[total_days,1]);
plot_ind = [128,79];
country_name = {'India','Pakistan'};
for j = 1:2
    figure(j)
    for i = 1:3
        subplot(1,3,i)
        X = series_type{i}{plot_ind(j),1:total_days};
        Y = [0 series_type{i}{plot_ind(j),total_days+1:end}];
        R = pak_rate(:,i);
        h = scatter(X',Y',3,color{i},'filled','DisplayName', ...
            display_name{i}); % Create plot with logarithmic scale
        hold on;
        plot(X',R,'--','Color',color{i},'Marker','.');
        logX = log10(X);
        logX(logX == -inf) = 0;
        logY = log10(Y);
        logY(logY == -inf) = 0;
        P = polyfit(logX' ,logY',1);
        yfit = (X).^P(1).*(10^(P(2)));
        g = plot(X',yfit','Color',color{i}, 'DisplayName', 'Approximation');
        text(2,R(1,1)+0.2*R(1,1),display_name1{i},'FontSize',9)
        set(gca,'Xscale','log')
        set(gca,'Yscale','log')
        xlabel(xlabels{i});
        ylabel('Weekly Differences');
        legend([h g], 'Location', 'southeast');
        grid on
        hold off
    end

formatOut = 'mm/dd';
lastDate = datestr(lastDate, formatOut);
sgtitle( sprintf('%s Covid 19 Trajectory until %s ', ...
    country_name{j},char(lastDate)));
end
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Negative data ignored
```

```
clear data data_diff cats cc ds avg lastDate lc t0 n firstIndex
clear xlabels X logX Y logY color
```

iv) Add total infected, total deaths and total recovered and join population data

```
data_covid.Total_Infections = data2_confirmed{:,c};
data_covid.Total_Deaths = data2_deaths{:,c};
data_covid.Total_Recovery = data2_recovered{:,c};
data_covid.Total_Prevalence = data2_confirmed{:,c} - (data2_deaths{:,c} + data2_recovered{:,c}
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
data_population = readtable(filename);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
 variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers as table variable names.
```

```
data_population.Properties.VariableNames{1} = 'CountryRegion';
data_population.CountryRegion = string(data_population.CountryRegion);
[data_join, ileft, iright] = innerjoin(data_covid,data_population(:,[1:2]),'keys','CountryRegio
indx = find(ismember(1:188,ileft)==0);
data_covid = data_join;

data_covid.percentage_of_infections = data_covid.Total_Infections./data_join.Population;
data_covid.percentage_of_deaths = data_covid.Total_Deaths./data_covid.Population;
data_covid.percentage_of_recovered = data_covid.Total_Recovery./data_covid.Population;
```

```
%data_covid.percentage_of_prevalence = data_covid.Total_Prevalence./data_covid.Population;


%normalize by population to get the average parameter
data_covid{:,{'Average_Infection_Rate','Average_Mortality_Rate','Average_Recovery_Rate'}} = ...
    data_covid{:,{'Average_Infection_Rate','Average_Mortality_Rate','Average_Recovery_Rate'}}./
head(data_covid,5);
clear c data_confirmed data2_confirmed data_deaths data2_deaths data_recovered data2_recovered
```

c) Visualize corona virus data.

```
%sort_coronavirus_data
covid_confirmed = sortrows(data_covid(:,{'CountryRegion','Total_Infections'}),'Total_Infections
covid_confirmed_rate = sortrows(data_covid(:,{'CountryRegion','Average_Infection_Rate'}),'Avera
covid_deaths = sortrows(data_covid(:,{'CountryRegion','Total_Deaths'}),'Total_Deaths','Ascend')
covid_deaths_rate = sortrows(data_covid(:,{'CountryRegion','Average_Mortality_Rate'}),'Average_
covid_recovery = sortrows(data_covid(:,{'CountryRegion','Total_Recovery'}),'Total_Recovery','As
covid_recovery_rate = sortrows(data_covid(:,{'CountryRegion','Average_Recovery_Rate'}),'Average
%covid_prevalence = sortrows(data_covid(:,{'CountryRegion','Total_Prevalence'}),'Total_Prevaler
%covid_prevalence_rate = sortrows(data_covid(:,{'CountryRegion','Average_Prevalence_Rate'}),'Av
covid_rate = data_covid(:,{'CountryRegion','Average_Infection_Rate','Average_Mortality_Rate','A
total_data = {covid_confirmed covid_confirmed_rate covid_deaths covid_deaths_rate covid_recover
titles_for_plots = {'Incidence','Average infection Parameter','Deaths', 'Average Mortality Para
j = 1;
color = {'b','r','g','c'};
%get top 20 countries
figure(3)
for i = 1:3
    subplot(3,2,j)
    barh(total_data{j}{end-4:end,2},color{i})
    set(gca,'YTick',[1:5],'yticklabel',total_data{j}(end-4:end,:).CountryRegion);
    title(titles_for_plots{j})

    subplot(3,2,j+1)
    barh(total_data{j+1}{end-4:end,2},color{i})
    set(gca,'YTick',[1:5],'yticklabel',total_data{j+1}(end-4:end,:).CountryRegion);
    title(titles_for_plots{j+1})
    j = j+2;
end
    sgtitle('Top 5 countries')
```

```
clear color total_data titles_for_plots covid_confirmed_rate covid_deaths_rate covid_recovery_r
```

Step 2: Processing happiness index data

```
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
```

```
Warning: Negative data ignored
filename =
'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\happiness_index_2020.csv'
```

```
data_happiness = readtable(filename);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
 variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers as table variable names.
```

```
data_happiness.Properties.VariableNames{1} = 'CountryRegion';
data_happiness.Properties.VariableNames{2} = 'HappinessScore';
data_happiness.CountryRegion = string(data_happiness.CountryRegion);
%joining covid 19 and happiness data
[data_join, ileft, iright] = innerjoin(data_covid,data_happiness(:,1:end),'keys','CountryRegior
indx = find(ismember(1:153,iright)==0);
head(data_happiness,5);
```

Step 3: Process press freedom index data

```
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
data_pfi = readtable(filename);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
 variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers as table variable names.
```

```
data_pfi = data_pfi(:,2:end);
data_pfi.Properties.VariableNames{1} = 'PFRank';
data_pfi.Properties.VariableNames{2} = 'CountryRegion';
data_pfi.Properties.VariableNames{5} = 'PressScore';
data_pfi{:,3:end} = 100 - data_pfi{:,3:end}; %reversing the trend
data_pfi.CountryRegion = string(data_pfi.CountryRegion);
%joining covid 19 and happiness data
[data_join2, ileft, iright] = innerjoin(data_join,data_pfi(:,2:end),'keys','CountryRegion');
indx = find(ismember(1:149,ileft)==0);
%data_join.CountryRegion{indx};
head(data_pfi,5);
```

Step 4: Process democracy index data

```
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
data_di = readtable(filename);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
 variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers as table variable names.
```

```
data_di.Properties.VariableNames{1} = 'CountryRegion';
data_di.Properties.VariableNames{2} = 'DemocracyScore';
data_di.CountryRegion = string(data_di.CountryRegion);
[data_join3, ileft, iright] = innerjoin(data_join2,data_di,'keys','CountryRegion');
indx = find(ismember(1:149,ileft)==0);
head(data_di,5);
```

Step 5: Process economic freedom index data

```
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
data_ei = readtable(filename);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
 variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers as table variable names.
```

```
data_ei.Properties.VariableNames{1} = 'CountryRegion';
data_ei.WorldRank = [];
data_ei.Properties.VariableNames{2} = 'EconomicScore';
data_ei.CountryRegion = string(data_ei.CountryRegion);
[data_join4, ileft, iright] = innerjoin(data_join3,data_ei,'keys','CountryRegion');
indx = find(ismember(1:147,ileft)==0);
head(data_ei,5);
```

Step 6: Human Freedom Index data

```
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
data_hfi = readtable(filename);
data_hfi.Properties.VariableNames{1} = 'CountryRegion';
data_hfi.Properties.VariableNames{3} ='HumanFreedomScore';
data_hfi(:,[2 4]) = [];
data_hfi.CountryRegion = string(data_hfi.CountryRegion);
head(data_hfi,5);
[data_join5, ileft, iright] = innerjoin(data_join4,data_hfi,'keys','CountryRegion');
indx = find(ismember(1:146,ileft)==0);
```

Step 7: World Population data

```
filename = 'C:\Users\User\Desktop\Summer 2020\Data Analysis\ECE 579A\covid19_analysis\datasets\
data_pop = readtable(filename);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
 variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers as table variable names.
```

```
data_pop.Properties.VariableNames{1} = 'CountryRegion';
data_pop.CountryRegion = string(data_pop.CountryRegion);
head(data_pop,5);
[data_set, ileft, iright] = innerjoin(data_join5,data_pop(:,[1 3:end]),'keys','CountryRegion');
indx = find(ismember(1:143,ileft)==0);
data_set.Migrants_net_ = data_set.Migrants_net_ ./data_set.Population;
data_set.Migrants_net_ = data_set.Migrants_net_ + repmat(abs(min(data_set.Migrants_net_)) + 0.0
```

Step 8: Create final data, address missing values

```
data_set = rmmissing(data_set); %remove the missing values
data_set.Properties.VariableNames{1} = 'Country';

data_set.Properties.RowNames = data_set.Country;
data_set = movevars(data_set,{'PressScore','DemocracyScore','EconomicScore','HumanFreedomScore'
covid_rate.Properties.RowNames = covid_rate.CountryRegion;
covid_rate = covid_rate(data_set.Country,:);

%normalize rates by Population and density
```

Step 9: Normalization

```
data_set_log  = data_set;
replicate = repmat(min(data_set{:,2:8},[],1),[size(data_set{:,2:8},1) 1]);
data_set_log{:,2:8} = data_set_log{:,2:8} + abs(replicate);
data_set_log(:,2:8) = varfun(@log10,data_set_log(:,2:8));
%data_set_log{:,51} = log(data_set{:,51})/log(8);
data = data_set_log{:,2:8};
for i  = 1:size(data,2)
    data_column = data(:,i);
    sorted =  sort(data_column);
    sorted(sorted==-inf) = [];
    data_column(data_column == -inf) = sorted(1) ;
    data(:,i) = data_column;
end
data_set_log{:,2:8} = data;
data_set_normalized = data_set;
data_set_normalized{:,2:8} = (1 - exp(-normalize(data_set_log{:,2:8}))))./(1 + exp(-normalize(da


clear data_happiness data_di data_ei data_hfi data_pfi data_join data_join1 data_join2 data_joi
head(data_set,5);
head(data_set_normalized,5);
titles_for_plots = {'average infection parameter', 'average mortality parameter' ,'average reco
figure(4)
j = 1;
for i = 1:6
    subplot(6,2,j)
    histogram(data_set{:,i+1})
    xlabel('Samples')
    ylabel('Frequency')
    title(sprintf('Original values of %s',titles_for_plots{i}))
    subplot(6,2,j+1)
    histogram(data_set_normalized{:,1+i})
    xlabel('Samples')
    ylabel('Frequency')
    title(sprintf('Normalized values of %s',titles_for_plots{i}))
    j = j+2;
end
```

Correlation between Average Infection Parameter, Average Mortality Parameter and Average Recovery Parameter.

```
figure(5)
varname = {'AIP','AMP', 'ARP'};
corrplot(data_set_normalized(:,["Average_Infection_Rate","Average_Mortality_Rate", "Average_Rec
title('Linear correlation between average infection parameter, average mortality parameter and
```

```
clear indx ileft iright filename
```

Visualization

a) Perform PCA for dimension reduction

```
%Create a scatter plot to find the relationship between infection rate and
%how to people recoover in happier counters and counteries with good gdp
%per capita?
%group the data according to infection rate, recovery rate and death rate
%find threshold for grouping
%Perform K means clustering and use percentage of recovered, percentage of
%deaths as the predictors

%create a group for clustering


%"AIR_population","ADR_population","ARR_population",
rand('state',47);
Xc = data_set_normalized{:,["Average_Infection_Rate","Average_Mortality_Rate","Average_Recovery
%Xr = data_set_by_region_normalized{:,["AIR_population","ADR_population","ARR_population","Tota

% Apply PCA for dimension reduction
[Uqc,score_c,latent,tsquared,explainedc,mu] = pca(Xc);
projc = Xc*Uqc;
% [Uqr,score_r,latent,tsquared,explainedr,mu] = pca(Xr);
%
% projr = Xr*Uqr;
value = abs(normalize(Xc'*projc));
figure(5)
heatmap(value.*explainedc','XLabel','Principal Components','YLabel','Variables');
title('Heatmap to illustrate the significant parameters and significant principal components')
```

```
rank_variables = sum(value.*explainedc',2);
rank_variables = rank_variables./sum(rank_variables)
```

```
rank_variables = 6×1
    0.1434
    0.1954
    0.3169
    0.1198
    0.0751
    0.1493
```

b) Perform cluster analysis using K-means algorithm

```


%group 142 countries into 3 clusters using K means clustering

K  = 2;
[idxc,Cc] = kmeans(projc(:,1:3),K,'MaxIter',10);
% [idxr, Cr] = kmeans(projr,K,'MaxIter',100);
% subplot(1,2,1)
figure(6)
```

```
grpstats(data_set{:,["Average_Infection_Rate","Average_Mortality_Rate", "Average_Recovery_Rate"
xlabel('Covid19 groups')
ylabel('Mean per group');
set(gca,'xtick',1:length(unique(idxc)),'xticklabel',{'Cluster 1', 'Cluster 2'});
title('Mean and 95% confidence intervals (Country)')
legend("Average Infection Parameter","Average Mortality Parameter", "Average Recovery Parameter
```

```
Warning: Ignoring extra legend entries.
```

```

figure(7)
%number of dimensions to reduce
% proj = (proj-min(proj)./(max(proj)-min(proj))); % 0 1 range
proj1 = projc(idxc==1,:);
proj2 = projc(idxc==2,:);
proj3 = projc(idxc==3,:);
plot3(proj1(:,1),proj1(:,2), proj1(:,3),'.b','linew',1.5,'MarkerSize',10)
hold on
plot3(proj2(:,1),proj2(:,2), proj2(:,3),'.r', 'linew',1.5,'MarkerSize',10)
plot3(proj3(:,1),proj3(:,2), proj3(:,3),'.g', 'linew',1.5,'MarkerSize',10)
grid on
grid minor
xlabel('PC 1')
ylabel('PC 2')
zlabel('PC 3')
title('Reduced dimensions to visualize clusters')
legend('Cluster 1', 'Cluster 2','location','northeast');
cluster_1 = data_set{:,1}(idxc == 1)
```

```
cluster_1 = 52×1 string
"Algeria"
"Australia"
"Benin"
"Burkina Faso"
"Burma"
"Burundi"
"Cambodia"
"Cameroon"
"Central African Republic"
"Chad"
      ⋮
```

From inspection, cluster 1 represents countries that are highly affected by covid, cluster 2 represents moderately affected countries and cluster 3 represents low affected countries.

```
cluster_2 = data_set{:,1}(idxc == 2)
```

```
cluster_2 = 88×1 string
"Albania"
"Argentina"
"Armenia"
"Austria"
"Azerbaijan"
"Bahrain"
"Bangladesh"
"Belarus"
```

```
"Belgium"
"Bolivia"
    ⋮
```

```
cluster_3 = data_set{:,1}(idxc == 3)
```

```
cluster_3 = 

  0×1 empty string array
```

```

%text(proj1(:,1),proj1(:,2),poorcovid,'FontSize',7,'HorizontalAlignment', 'left','VerticalAlign
% set(gca,'xscale','log');
hold off;
```

c) How is the performance of Covid 19 with respect to Happiness in various countries?

```
%clf([6 7 8])
visualize_plots(data_set,covid_rate, 'HappinessScore',idxc,K,0,'HappinessScore',1,1,7);
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

c) How is the performance of Covid 19 with respect to Life Expectancy in various countries?

```
%clf([9 10 11])
visualize_plots(data_set, covid_rate, 'HealthyLifeExpectancy',idxc,K,0,'Life Expectancy',1,1,10
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

d) How is the performance of Covid 19 with respect to gdp per capita in various countries?

```
%clf([9 10 11])
visualize_plots(data_set, covid_rate, 'GDPPerCapita',idxc,K,0,'GDP per Capita',1,1,13);
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

```
visualize_plots(data_set, covid_rate, 'PressScore',idxc,K,0,'Press Freedom',1,1,16);
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

```
%clf([9 10 11])
visualize_plots(data_set, covid_rate, 'HumanFreedomScore',idxc,K,0,'Human Freedom Index',1,1,19
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

```
visualize_plots(data_set, covid_rate, 'EconomicScore',idxc,K,0,'Index of Economic Freedom',1,1,
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

```
visualize_plots(data_set, covid_rate, 'UrbanPop_',idxc,K,0,'Urban Population',1,1,25);
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

```
visualize_plots(data_set, covid_rate, 'Med_Age',idxc,K,0,'Median Age',1,1,28);
```

```
Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.

Warning: Ignoring extra legend entries.
```

```
% figure(6)
% plot(data_set_normalized{:,'GDPPerCapita'})
% hold on
% plot(data_set_normalized{:,'Total_Infections'})
% plot(data_set_normalized{:,'Total_Deaths'})
% plot(data_set_normalized{:,'Total_Recovery'})
% hold off
% legend('total infections','total deaths','total recovery','location','northeast')
% legend boxoff
```

Visualizations

```
% %how does GDP per capita affect the covid spread?
% data_set_new = sortrows(data_set_normalized,'FreedomToMakeLifeChoices','descend');
% head(data_set_new,15);
% figure(7)
% stackedplot([data_set_new(1:20,4:6) data_set_new(1:20, ...
% 'FreedomToMakeLifeChoices') data_set_new(1:20,'pf_score') data_set_new(1:20,'ef_score') data_
% cell2table(data_set_new(1:20,:).Properties.RowNames,'VariableNames',{'Country'})
```

```
% figure(8)
% ax = axes; % create axes
% data_set_new = [data_set_new(:,1:3) data_set_new(:,'GDPPerCapita')];
% plot(ax,table2array(data_set_new(1:20,1:4))); % plot data
% ax.XTick = 1:20; % limit X-axis ticks no. to columns
% ax.XTickLabel = data_set_new(1:20,1:3).Properties.RowNames; % get columns names
% xtickangle(ax,45);
% legend('Average Infection Rate', 'Average Mortality Rate', 'Average Recovery Rate', 'GDPPerCa
```

Step 3: Visualize the results

Let's answer our questions about case numbers vs specific countries.

In the figure below, you can see the most significant worldwide cases, or you can filter to the territory of Gamax Laboratory Solutions and Europe.

It was easy to add country-specific filtering with Live Editor. You can add many interactive tasks to your Live Script.% % Filter by region

```
%functions
```

```
function  visualize_plots(dcountry,covid_rate, param,idxc,K,xscale,xlabelv,yscale,textc,lastfig
    color = {'b','r','g'};
    C = {'Total_Infections','Total_Deaths','Total_Recovery'};
    D = {"Average_Infection_Rate","Average_Mortality_Rate", "Average_Recovery_Rate"};
    E = string({sprintf('%s and Covid-19 infections',xlabelv), sprintf('%s and Covid-19 deaths'
    covid_confirmed_rate = sortrows(covid_rate(:,{'CountryRegion','Average_Infection_Rate'}),'A
    covid_deaths_rate = sortrows(covid_rate(:,{'CountryRegion','Average_Mortality_Rate'}),'Aver
    covid_recovery_rate = sortrows(covid_rate(:,{'CountryRegion','Average_Recovery_Rate'}),'Ave
    %covid_prevalence_rate = sortrows(covid_rate(:,{'CountryRegion','Average_Prevalence_Rate'})
    F = {covid_confirmed_rate, covid_deaths_rate, covid_recovery_rate};
    G = {'Average Infection Parameter', 'Average Mortality Parameter', 'Average Recovery Parame
    countries = sortrows(dcountry(:,{'Country',param}),param,'descend');
    poorcovid = dcountry{:,1}(idxc == 2);
    best_worse = countries{:,1}([1:5 end-4:end]);

    for i = 1:length(C)
        figure(lastfigno +i);
        markerSizes = normalize(dcountry{:,C{i}},'range',[10 1000]);
%         markerSizes = markerSizes + abs(min(markerSizes));
%         markerSizes = round(markerSizes + 1);
        for j = 1:K
            scatter(dcountry{:,param}(idxc == j),dcountry{:,D{i}}(idxc == j),markerSizes(idxc =
            hold on;
            grid on;

        end
        countries_to_plot = F{i}{:,'CountryRegion'}(end-19:end);
        countries_to_plot = unique([countries_to_plot ; best_worse ; poorcovid]);
        legend('Cluster 1', 'Cluster 2', 'Cluster 3','location','southeast');
        xminvalc = min(dcountry{:,param});
        xmaxvalc = max(dcountry{:,param});
        yminvalc = min(dcountry{:,D{i}});
        ymaxvalc = max(dcountry{:,D{i}});

        if textc
            text(dcountry{countries_to_plot,param},(dcountry{countries_to_plot,D{i}}),countries
                'FontSize',7,'HorizontalAlignment', 'left','VerticalAlignment', 'bottom');
        end
        hold off;
        if xscale
           set(gca,'xscale','log');
        end
        if yscale
           set(gca,'yscale','log');
           axis([xminvalc - 0.2*xminvalc xmaxvalc + 0.2*xmaxvalc  0 ymaxvalc + 0.2*ymaxvalc])
        else
            axis([xminvalc - 0.2*xminvalc xmaxvalc + 0.2*xmaxvalc  yminvalc - 0.2*yminvalc ymax
        end

        title(E{i});
        ylabel(G{i});
        xlabel(xlabelv);
    end
end
```