COVID-19 Analysis via Logistic Curve and Clustering

Yimeng Shang, Weijia Xiong, Ruoyuan Qian

April 25, 2020

Abstract

In this report, optimization of logistic curve with a combination of ordinary differential equation(ODE) and gradient descent is conducted, fitted curves are made and the total infection cases, midpoints of infections, the infection rates are estimated for each country in the dataset. Moreover, k-means and Gaussian mixture model with EM algorithm are done to cluster countries into 5 groups. 3D plots and world maps are shown to compare two methods. Using the data before 03/23/2020, the fastest growth country is Trinidad and Tobago, the flattest growth region is Taiwan. The definition of the end phase of the spread are made and they are China, Denmark, etc. For clustering, k-means model is better since it can distinguish countries through their total infection cases, midpoints of infections, the infection rates clearly. China and Europe have larger total cases and speed. The U.S. obtains large total cases and long duration, which are consistent with reality. We also apply the same methods to the updated data from 1/22/2020 to 4/24/2020 for comparisons.

Background:

As for the global pandemic of COVID-19, there is an increasing desire to analyze the total infection cases, midpoints of infections, the infection rates for countries, which can help epidemiologists and the authorities have a better understanding about the fast-changing situation and help them deal with the unprecedented disease.

Dataset:

There are two datasets in the report: data from 1/22/2020 to 3/24/2020, data from 1/22/2020 to 4/24/2020. The result is based on the former one, and we compare the fitting difference between them as well.

Both of the datasets contain 5 variables of interest: Country/Region, Date, CumulativeCases(ConfirmedCases), Lat, Long. We defined cases_change as the difference of cases between that day and the day before.

Since all the countries contain the same start days, however, in reality, not all countries have an outbreak at the begining of record, so too many zeros occurred in cumulative cases for some countries, which will affect the fitted curves.

Objectives:

1. Analyzing the growth rate, maximum number of cases and mid-point for regions through a logistic curve with gradient decent optimization and calculating start values through ordinary differential equation (ODE).
2. Clustering the countries by K-mean and Gaussian mixture model with EM algorithm.
3. Visualization through world maps and 3D plots to compare these two methods.

Method

Methods for optimization

The logistic curve function is a non-linear curve with parameters a,b,c and variable t. To find the best estimated a,b,c, here we use the combination of the following two methods: ODE and Gradient Descent.

Ordinary Differential Equation

The logistic curve function is:

$$f(t) = \frac{a}{1 + e^{-b(t-c)}}$$

We take the derivative of $f(t)$ with respect to t, then we get the following form, which is a bernoulli ordinary differential equation:

$$\frac{df}{dt} = \frac{-a}{(1 + e^{-b(t-c)})^2} e^{-b(t-c)}(-b) = -bf(1 - \frac{f}{a})$$

Then we move the f from right hand side to the left hand side:

$$\frac{df}{dt}\frac{1}{f} = -b(1 - \frac{f}{a}) = \frac{b}{a}f - b$$

where df is the cases-change in a small time interval, dt is the time differenct, f is the cumulative cases at time t.

From the equation above, it's easy to see that we can calculate the value of left hand side from the given data and f. The equation can then be viewed as a linear regression problem with response $\frac{df}{dt}\frac{1}{f}$, predictor f, intercept $-b$ and slope $\frac{b}{a}$. We can then use least square method to estimate a and b. After we get the estimated a and b, we can plug them into the logistic curve function along with the data for one specific date to get the estimated c.

In details, for those countries with only few changes in data, we remove them. For other countries, we use the top 30% of the df to fit the linear regression in order to offset the overfitting of the onset part of disesase. And for estimated c, we calculate all estimated c from all date, and take the average to get to final estimated c.

Gradient Descent

Using gradient descent to optimize the a,b,c is an intuitive way. Here we use a loss function: $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}[(f_\theta(t^{(i)}) - y^{(i)})/y^{(i)}]^2$, which indicates the prediction difference adjusting for cumulative cases. The aim of gradient descent is to minimize the loss function.

We get the gradient:

$$\frac{\partial J}{\partial a} = \sum_{i=1}^{n}(f_\theta(t^{(i)}) - y^{(i)})\frac{1}{(1 + e^{-b(t^{(i)})-c})(y^{(i)})^2}$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{n}(f_\theta(t^{(i)}) - y^{(i)})\frac{a(t^{(i)} - c)}{(1 + e^{-b(t^{(i)})-c})(y^{(i)})^2}$$

$$\frac{\partial J}{\partial c} = \sum_{i=1}^{n}(f_\theta(t^{(i)}) - y^{(i)})\frac{-ab}{(1+e^{-b(t^{(i)})-c})(y^{(i)})^2}$$

$$Gradient = (\frac{\partial J}{\partial a}, \frac{\partial J}{\partial b}, \frac{\partial J}{\partial c})$$

In order to make $\frac{\partial J}{\partial a}$ change faster, we modify the Gradient to $Gradient = (\frac{\partial J}{\partial a} \times (y^{(i)})^2, \frac{\partial J}{\partial b}, \frac{\partial J}{\partial c})$

We repeat the loop until it converges:

$$\theta_j := \theta_j - \alpha \times ModifiedGradient$$

where α is a small number we choose to control the step length called learning rate.

However, as we know, for non-convex function, gradient descent is very sensitive to the initial start value we plug in. The gradient descent can only find local minimal for non-convex function. To solve this issue, we combine these two methods together. We use the ODE method to get the estimated a,b,c and used it as the initial value for gradient descent to get our final estimated a,b,c.

The meaning of a,b,c can be interpreted as the maxmized estimated total cases, the speed of the spread and the days from the onset to the mid-point.

Methods for clustering

Here we use K-means and Gaussion Mixture model for clustering.

K-means

We denote $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\}$ as the centers of the k (unknown) clusters, and denote $\mathbf{r}_i = (r_{i,1}, ..., r_{i,k}) \in \mathbb{R}^k$ as the "hard" cluster assignment of $\mathbf{x}_i$. The cluster assignment $\mathbf{r}_i$ takes form $(0, 0, ..., 0, 1, 0, 0)$ with $r_{i,j} = I\{\mathbf{x}_i$ assigned to cluster $j\}$.

k-means essentially finds cluster centers and cluster assignments that minimize the objective function

$$J(\mathbf{r}, \boldsymbol{\mu}) = \sum_{i=1}^{n}\sum_{j=1}^{k} r_{i,j}\|\mathbf{x}_i - \mu_k\|^2$$

The K-means algorithm follows the following steps:

1. Standardize the data
2. Randomly initialize k cluster centers $\{\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, ..., \boldsymbol{\mu}_k^{(0)}\}$.
3. Repeat the following two steps iteratively until converge.
    - Find optimal cluster assignment fixing cluster centers.Minimizing $J(\mathbf{r}, \boldsymbol{\mu})$ over $\mathbf{r}$ yields

      $$r_{i,j}^{(v+1)} = I\{j = \arg\min_j \|\mathbf{x}_i - \mu_j^{(v)}\|\}$$

      That is, assign $\mathbf{x}_i$ to cluster j with minimal distance $\|\mathbf{x}_i - \mu_j^{(v)}\|$, where $\mu_j^{(v)}$ is the cluster center in the v-th iteration.
    - Calcuate cluster centers using the cluster assignment in the last step. Minimizing $J(\mathbf{r}, \boldsymbol{\mu})$ over $\boldsymbol{\mu}$ yields

$$\mu_j^{(v+1)} = \frac{\sum_{i=1}^n \mathbf{x}_i r_{i,j}^{(v+1)}}{\sum_{i=1}^n r_{i,j}^{(v+1)}}$$

That is the sample mean of $\mathbf{x}_i$ that were assigned to the cluster j in the last step.

Gaussian Mixture Model

Let $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^p$ be a collection of p dimensional data points. Assuming that

$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^p$$

are i.i.d. random vectors following a mixture mulitvariate normal distributions with k hidden groups.

$$\mathbf{x}_i \sim \begin{cases} N(\boldsymbol{\mu}_1, \Sigma_1), \text{with probability } p_1 \\ N(\boldsymbol{\mu}_2, \Sigma_2), \text{with probability } p_2 \\ \quad \vdots \quad \quad , \quad \quad \vdots \\ N(\boldsymbol{\mu}_k, \Sigma_k), \text{with probability } p_k \end{cases}$$

$$\sum_{j=1}^k p_j = 1$$

The density of a multivariate normal $\mathbf{x}$ is

$$f(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^p |\Sigma|}}$$

where $\boldsymbol{\mu}$ is p-dimensional mean, and Σ is $p \times p$ variance-covariance matrix;

The observed likelihood of $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ is

$$L(\theta; \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) = \prod_{i=1}^n \sum_{j=1}^k p_j f(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)$$

Then, let $\mathbf{r}_i = (r_{i,1}, ..., r_{i,k}) \in \mathbb{R}^k$ as the cluster indicator of $\mathbf{x}_i$, which takes form $(0, 0, ..., 0, 1, 0, 0)$ with $r_{i,j} = I\{\mathbf{x}_i \text{ belongs to cluster } j\}$. The cluster indicator $\mathbf{r}_i$ is a latent variable that cannot be observed.

The distribution of $\mathbf{r}_i$ is

$$f(\mathbf{r}_i) = \prod_{j=1}^k p_j^{r_i, j}$$

The conditional distribution of $\mathbf{x}_i$ given $\mathbf{r}_i$ is

$$f(\mathbf{x}_i | \mathbf{r}_i) = \prod_{j=1}^k f(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)^{r_i, j}$$

The joint distribution is given by $f(\mathbf{x}_i, \mathbf{r}_i) = f(\mathbf{x}_i | \mathbf{r}_i) f(\mathbf{r}_i)$

Therefore, the complete likelihood function is

$$L(\theta;\mathbf{x},\mathbf{r})=\prod_{i=1}^{n}\prod_{j=1}^{k}[p_j f(\mathbf{x}_i;\boldsymbol{\mu}_j,\Sigma_j)]^{r_{i,j}}$$

The complete log-likelihood is

$$\ell(\theta;\mathbf{x},\mathbf{r})=\sum_{i=1}^{n}\sum_{j=1}^{k}r_{i,j}[\log p_i+\log f(\mathbf{x}_i;\boldsymbol{\mu}_j,\Sigma_j)]=\sum_{i=1}^{n}\sum_{j=1}^{k}r_{i,j}[\log p_i-1/2\log|\Sigma|-1/2(\mathbf{x}_i-\boldsymbol{\mu}_j)^\top\Sigma(\mathbf{x}_i-\boldsymbol{\mu}_j)]$$

We apply EM algorithm to estimate the parameter μ's, $\boldsymbol{\Sigma}$'s and p_j's.

E-step

Evaluate the responsibilities using the current parameter values

$$\gamma_{i,k}^{(t)}=P(r_{i,k}=1|\mathbf{x}_i,\theta^{(t)})=\frac{p_k^{(t)}f(\mathbf{x}_i|\boldsymbol{\mu}_k^{(t)},\Sigma_k^{(t)})}{\sum_{j=1}^{K}f(\mathbf{x}_i|\boldsymbol{\mu}_j^{(t)},\Sigma_j^{(t)})}$$

M-step

$\theta^{(t+1)}=\arg\max\ell(\mathbf{x},\gamma^{(t)},\theta)$.

Let $n_k=\sum_{i=1}^{n}\gamma_{i,k}$, we have

$$\boldsymbol{\mu}_k^{(t+1)}=\frac{1}{n_k}\sum_{i=1}^{n}\gamma_{i,k}\mathbf{x}_i$$

$$\Sigma_k^{(t+1)}=\frac{1}{n_k}\sum_{i=1}^{n}\gamma_{i,k}(\mathbf{x}_i-\boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_i-\boldsymbol{\mu}_k^{(t+1)})^T$$

$$p_k^{(t+1)}=\frac{n_k}{n}$$

Result

Optimization Results

The time period in the given data is from 01/22/2020 to 03/24/2020, 62 days in total. There are 163 countries in total. Among them, 18 countries don't have enough data for us to get a valid estimation and 8 have negative speed, which doesn't make sense, so finally, we get the estimation of 137 countries.

We divide the disease development into 3 phases based on the parameter c (time to mid-point) in logistic regression: initial phase (hasn't passed estimated mid-point), intermediate phase (just passed estimated mid-point), end of the spread phase.

The realistic meaning of b is the speed of spread. According to the estimated b, we can find the top 10 fast/slow growth regions as shown in Table1 and Table2.

country_name	a	b	c
Trinidad and Tobago	58.283	2.519	58.271
Equatorial Guinea	10.959	1.562	58.369
Togo	30.680	1.349	60.256
Andorra	136.841	1.100	57.715
Guyana	21.187	1.099	58.441
Kyrgyzstan	35.017	1.058	59.893
Tanzania	25.282	0.884	60.770
Congo (Brazzaville)	6.149	0.831	58.879
Turkey	2311.972	0.793	60.072
Venezuela	80.038	0.747	55.641

Table 1: top 10 fast growth regions

country_name	a	b	c
Taiwan*	314.655	0.078	62.699
Vietnam	207.725	0.082	62.146
Singapore	5124.963	0.087	86.638
United Arab Emirates	601.296	0.103	68.203
Japan	1447.918	0.108	51.008
Saint Lucia	8.345	0.120	66.699
Oman	156.562	0.126	64.024
Sudan	9.152	0.141	67.895
Iraq	482.618	0.148	59.632
Bahrain	486.651	0.152	54.339

Table 2: top 10 flat growth regions

From the estimated c, there're 83 countries has passed the midpoint using data before 03/24/2020. To measure the distance of the number of cases at 03/24/2020 to the estimated max number of cases, we choose a threshold of 0.05. When $\frac{|MaxCases-NowCases|}{MaxCases} < 0.05$, we can define that the region approaches the end of the spread. Table 3 shows the countries approach the end of the spread.

	country_name	distance
1	Holy See	0.00
2	Uruguay	0.01
3	Philippines	0.02
4	Sweden	0.02
5	Trinidad and Tobago	0.02
6	Pakistan	0.02
7	China	0.02
8	Denmark	0.04
9	Maldives	0.05
10	Estonia	0.05
11	Venezuela	0.05

Table 3: Approach the end of spread region

According to Figure 1, two countries are chosen for each phase, each column represents a phase with the order of initial, intermediate and end of the spread phase. As shown from this plot, our model fits the given data well no matter the phase of disease development.

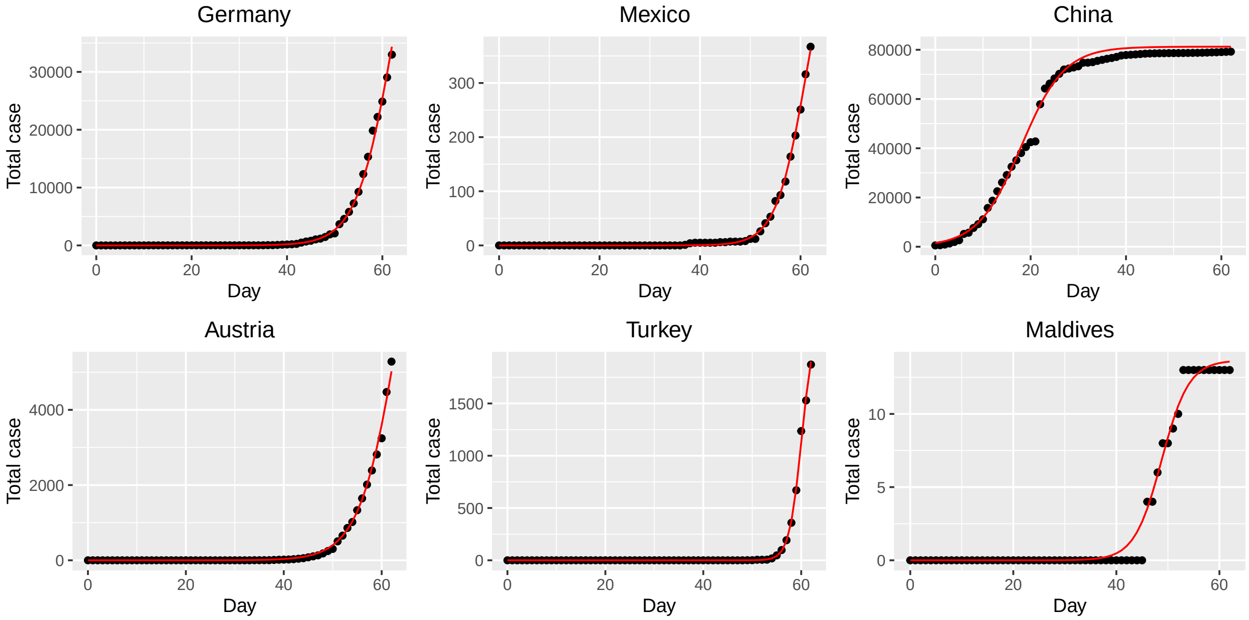

Figure 1: 3 phases of disease development

Clustering Results

We choose k = 5 and apply K-mean and Guassian mixture model (with EM algorithm) to cluster the fitted parameters $\hat{a}, \hat{b}, \hat{c}$ and draw the 3D plots. Then we draw the world maps based on difference cluster centroids (a,b,c). Here we convert the number of (a,b,c) into color Red, Green and Blue to get the rgb Hex Code, suggesting that red groups have larger estimated total cases(a), green groups have larger speed(b) and blue groups have larger time(c).

From the 3D plots and pairs plots(Figure 2, Figure 5 and Figure 6), we can find that comparing with GMM, K-means performs better in clustering. Since we use multivariate normal distribution in GMM, the probability density here takes only very small number and becomes even smaller after taking exponential. So the clusters will become more and more close to each other. Although we set the number of cluster equals to 5, we can only get 3 distinctive clusters.

From the maps(Figure 3), we can find that China and Europe seem to have larger total cases and speed of spread. The U.S. obtains large total cases and long duration. These results are all consistent with the reality.

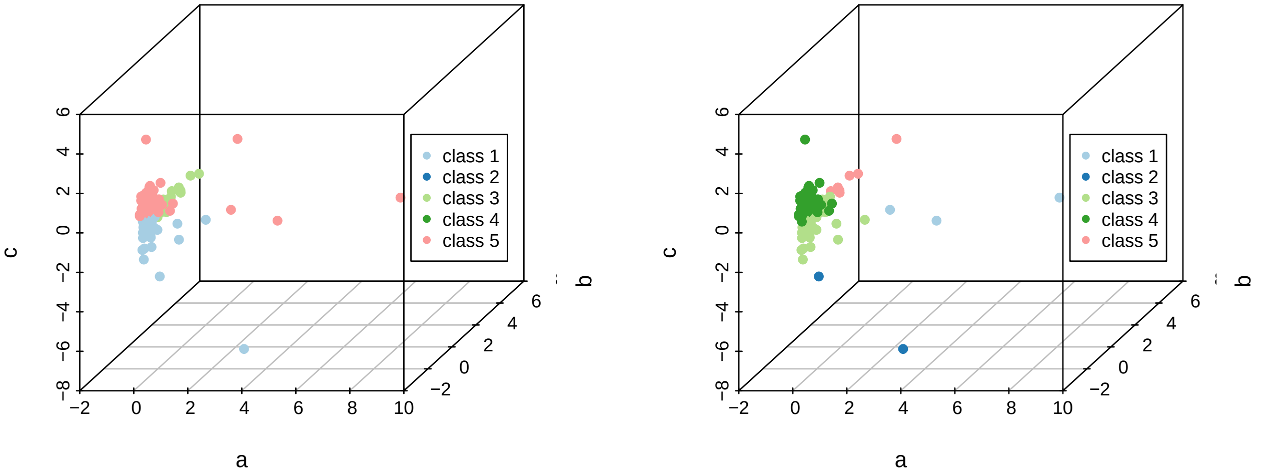

Figure 2: 3 phases of disease development

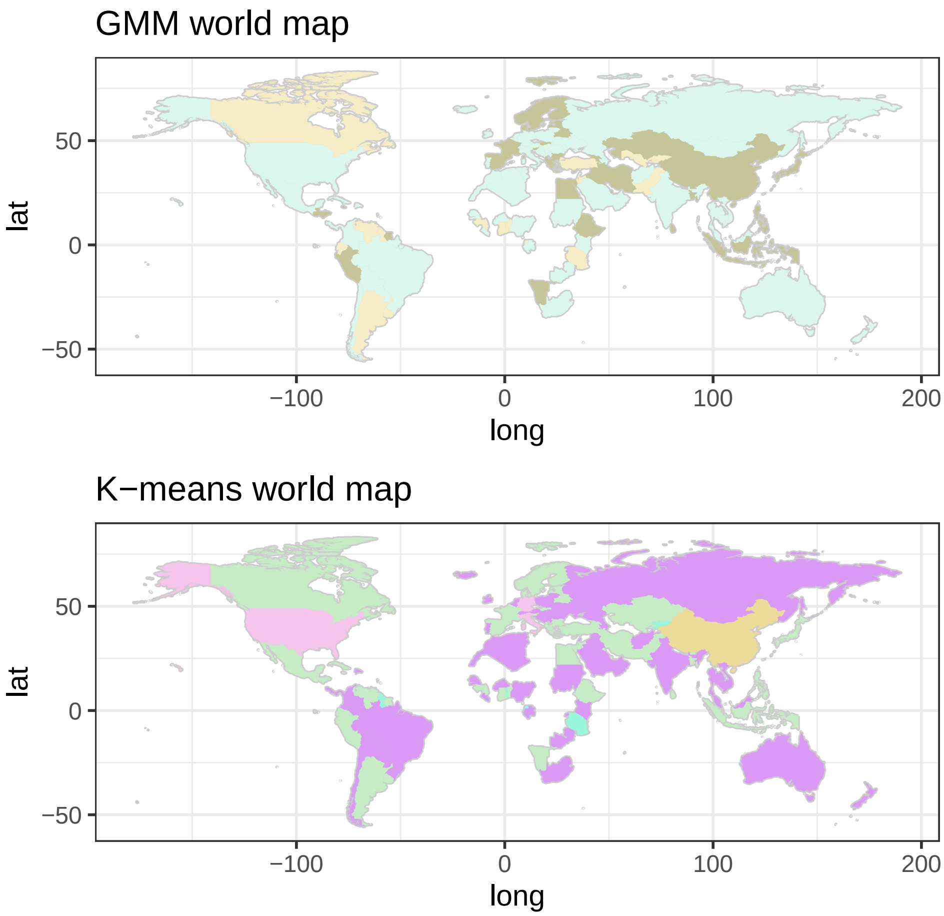

Figure 3: 3 phases of disease development

Comparison to new data

All the results showing above are based on the old data from 1/22/2020 to 3/24/2020, now we apply the same methods to the updated data from 1/22/2020 to 4/24/2020 and do comparisons.

We use the parameters estimated through the old data and check its performance in updated data. According to Figure 4, except for China, the model prediction ability is poor. It turns out that the model performs well only when the country is in end of the spread phase and will not have a second outbreak. There are some possible reasons to explain that. For every country, it takes time to develop a compelte and effective detection system so that the detection ability is poor at the beginning of the disease. As a result, the data at the beginning is very possible to be underestimated. Moreover, the logistic curve model might be better at interpretation and explanation rather than prediction, so the model itself may not be a preferable choice for prediction. Even if we do prediction based on the logistic curve model, it can only do short-term prediction within a few days instead of predicting the next 30 days. The reason is that the real data can be impacted due to a variety of fast-changing effects, such as, the issue of policies in public health, the rate of migration in and out, the climate change, clinical support and etc.

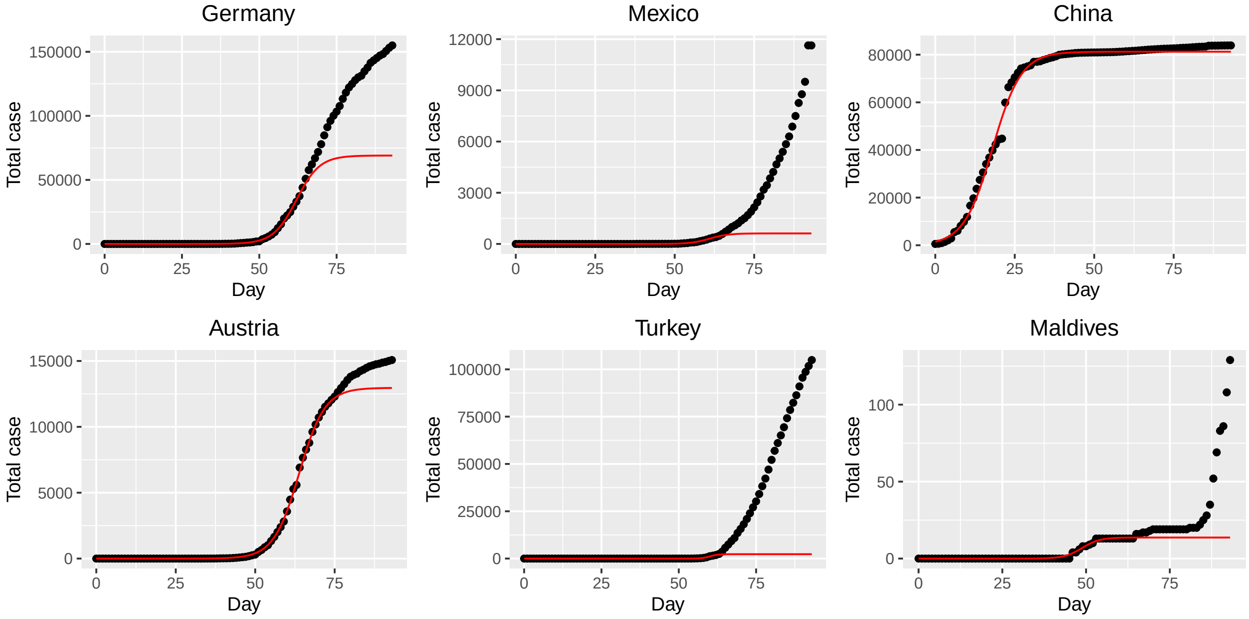

Figure 4: 3 phases of disease development with new data

Prospective

The report focuses on the analysis of country-level data. The province/state-level data for severely infected countries can be analyzed to show where is the most severe region in a country and its relation to other regions.

Based on the logistic curve model, the fitted curves in some countries are not very close to the observed data, such as Vietnam and Taiwan, the reason might be there are too many days with non-zero but small values, so we do not prefix them at the beginning, which makes the model underestimate the result. Therefore, some improvements about how to fit curves for some special countries can be conducted.

For now, our loss function for the logistic curve model is $\sum_{i=1}^{n}(\frac{\hat{y_i}-y_i}{y_i})^2$. We can try some other reasonable loss functions to improve the final result. Moreover, the Bayesian methods can be implemented rather than the logistic curve model for prediction.

Up to now, all we did is based on cumulative infection cases for each country, however, we could standardize the data based on the national population and national land area before the estimation. The result might be more meaningful and informative.

As for clustering, method GMM is not precise enough, so further efforts can be done to improve the efficiency of clustering for GMM.

Appendix

Figures

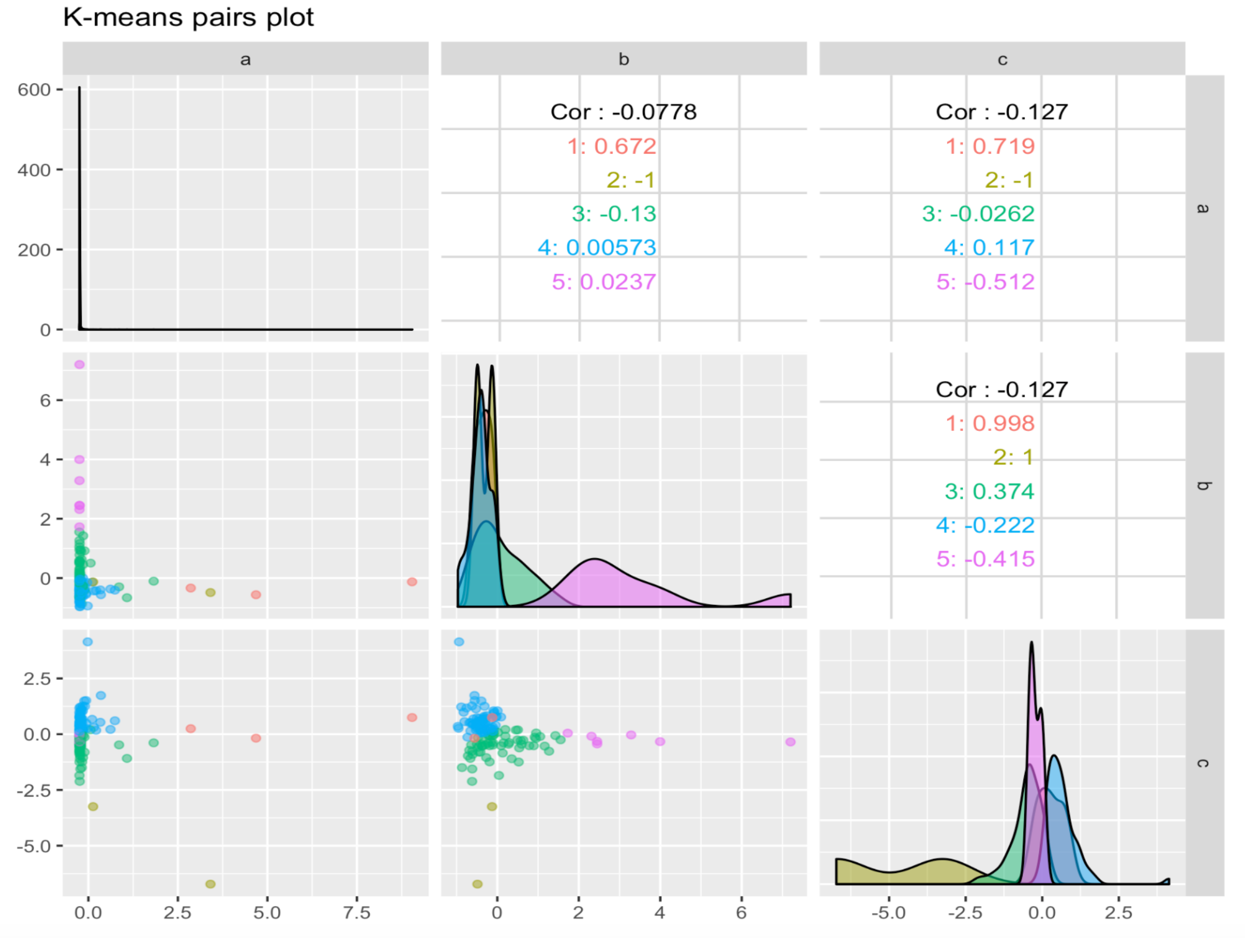

Figure 5: K-means pairs plot

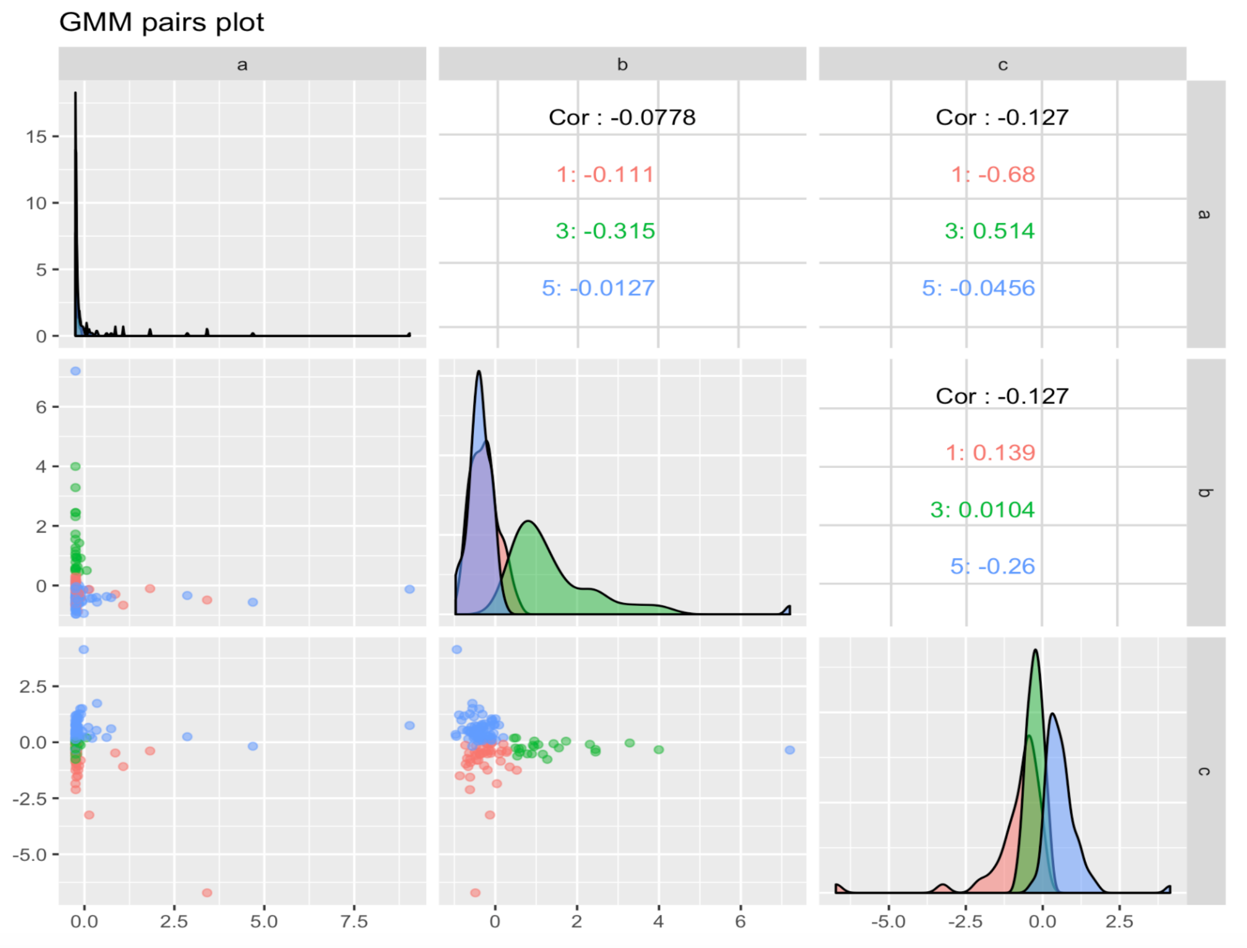

Figure 6: Gmm pairs plot

Code

```
library(tidyverse)
library(magrittr)
library(tidyverse)
library(InspectChangepoint)
library(mvtnorm)
library(ClusterR)
library(patchwork)
library(GGally)
library("rgl")
library("car")
library("RColorBrewer")
library("scatterplot3d")
```

Task1 Part 1: functions preparation

```
# select data for linear regression

data_truncated = function(country, percentile = 0.3) {
  ft = country$total_case
  df = tail(ft, -1) - head(ft, -1)
  df = c(0,df)

  df_dataframe = data.frame(cases_change = df)
  cutpoint_ind = as.integer(length(df)*(1 - percentile))
  new_dataframe = bind_cols(country, df_dataframe) %>% arrange(cases_change) %>%
    filter(cases_change >= cases_change[cutpoint_ind])

  return(new_dataframe)
}

cost = function(para, X, Y) {
  # X : n * 1  (time1, time2, time3 ... timeN)
  # y : n * 1  (res1, res2 ... resN)
  # para: 3 * 1 (a, b, c)
  # return: 1 float number
  a = para[1]
  b = para[2]
  c = para[3]

  prediction = predict(para, X)
  return(sum((prediction - Y) / Y)^2)
}

gradient = function(para, X, Y) {
  # X : n * 1  (time1, time2, time3 ... timeN)
  # y : n * 1  (res1, res2 ... resN)
  # para: 3 * 1 (a, b, c)
  # return: 3 * 1 (fa, fb, fc)
  a = para[1]
  b = para[2]
  c = para[3]

  e = exp(-b * (X - c))
  prediction = predict(para, X)

  fa = (prediction - Y) * (1 / (1 + e))
  fb = (prediction - Y) / Y^2 * (a * (X - c) / (1 / e + 2 + e))
  fc = (prediction - Y) / Y^2 * (-a * b  / (1 / e + 2 + e))

  s_fa = mean(fa)
  s_fb = mean(fb)
  s_fc = mean(fc)
  return(c(s_fa, s_fb, s_fc))
}

predict = function(para, X) {
  # X is n * 1   (time1, time2, time3 ... timeN)
  # return n * 1 (predict1, predict2 ... predictN)
  a = para[1]
  b = para[2]
  c = para[3]

  e = exp(-b * (X - c))
  return(a / (1 + e))
}

ode_method = function(country_df) {
  t = country_df$date_diff
  ft = country_df$total_case
  df = country_df$cases_change

  lhs = df/ft
  rhs = ft
  ## Y = f'/f = b(1-f/a) = -b/a * f + b

  # do regression
  lm_res = lm(lhs~rhs)
  beta0 = lm_res$coefficients[1]
  beta1 = lm_res$coefficients[2]

  b = beta0
  a = -b/beta1

  c = t + log((a/ft) - 1)/b
  c = c[!is.nan(c)]
  c = mean(c)

  return(c(a,b,c))
}

# 1.Clean Data
gradient_descent = function(country, alpha = 1e-3, tol = 1e-1) {

  Y = country$total_case
  X = country$date_diff

  ite_count = 0
  max_ite = 1000000

  # 2. Call ode method to get init value
  # para_new = ode_method(xxx)

  para = c(0, 0, 0)
  para_new = ode_method(country)

  while ((ite_count < max_ite)) {
    para = para_new
    grad = gradient(para, X, Y)
    # if(sum(abs(grad)) < tol) {
```

```
    #   print("converge!")
    #   break;
    # }
    para_new = para - alpha * grad
    ite_count = ite_count + 1
    # if(ite_count %% 10000 == 1) {
    #   print(cost(para, X, Y))
    #   print(para)
    # }
  }
  return(para_new)
}
```

Task1 Part 2: Data manipulation (data untill 03/23)

```
path = "../covid19-1.csv"
data = read_csv(path)

data = data %>% janitor::clean_names() %>%
  mutate(date = as.Date(date, format = '%m/%d/%y')) %>%
  mutate(date_diff = date - date[1])

task1 = data %>%
  group_by(country_region, date_diff) %>%
  summarise(total_case = sum(confirmed_cases)) %>%
  mutate(date_diff = as.numeric(date_diff))

country_list = unique(task1$country_region)
```

Task1 Part3: get the estimated a,b,c

```
fitted_data_all = data.frame()
para_all = data.frame()
errorlist = list()

for (country_name in country_list) {

  skip_to_next <- FALSE
  few_data <- FALSE

  country_raw = task1 %>% filter(country_region == country_name) %>% filter(total_case != 0)
  data_selected = tryCatch(data_truncated(country_raw, 0.5),
                           error = function(e) {
                             few_data <<- TRUE
                             print(paste("Too many zero in", country_name))},
                           finally = print(paste(country_name, "Selection Done")))

  para = tryCatch(gradient_descent(data_selected),
                  error = function(e) {
                    print(paste("Error in optimization for", country_name))
                    skip_to_next <<- TRUE},
                  finally = print(paste(country_name, "Optimization Done")))

  if (few_data) {errorlist = c(errorlist, country_name)}
  if (skip_to_next) {next}

  para_res = data.frame(country_name, para[1], para[2], para[3])
  col_name <- c("country_name","a", "b", "c")
  colnames(para_res) <- col_name
  para_all = bind_rows(para_all, para_res)

  spe_country = task1 %>% filter(country_region == country_name)
  fitted_Y = predict(as.numeric(para), spe_country$date_diff)
  fitted_data = data.frame(spe_country$date_diff,
                           spe_country$total_case,
                           fitted_Y)

  fitted_data = data.frame(country_name, fitted_data)
  fitted_data_all = bind_rows(fitted_data_all, fitted_data)
}

write_csv(para_all,"para_all_old.csv")
write_csv(fitted_data_all,"fitted_all_old.csv")
write_delim(errorlist, "errorlist.csv")
```

Task1 Part4: results analysis

```
para_all_3 = read_csv("para_all_old3.csv")
errorlist_3 = read_csv("errorlist_3.csv")

## NA country: limited cases
na_countrylist_3 = errorlist_3 %>% as.character()

## NaN country: negative predicted a value, perhaps give an initial by guessing a/b/c
nan_countrylist_3 = para_all_3 %>% filter(a == "NaN") %>% select(country_name) %>% as.list()
nan_countrylist_3 = nan_countrylist_3$country_name # 29

## Negative b country:
para_negativeB_3 = para_all_3 %>% filter(b < 0)
negative_b_countrylist_3 = para_negativeB_3$country_name # 6


# When threshold = 0.3
# Total country: 163

# parameter all = 145
# NA country: 18
# 145 + 18 = 163

# NaN country: 29
# Negative b country: 6

# Analysis of old data result (threshold = 0.3)
# 18 NA countries
# 145 parameters

para_final = read_csv("para_final.csv") %>% filter(b > 0)

# dim(para_final)[1] # 137 countries in total

# task1 %>% group_by(country_region) %>% summarise(max(date_diff))

pass_midpoint = para_final %>% filter(c < 62)

# dim(pass_midpoint)[1] # 83 pass the midpoint

fast = para_final %>% arrange(-b)
fast10_speed = fast[1:10,]
fast10_speed %>% knitr::kable(caption = "top 10 fase growth region")

slow = para_final  %>% arrange(b)
slow10_speed = slow[1:10,]
slow10_speed %>% knitr::kable(caption = "top 10 flat growth region")

path = "../covid19-1.csv"
data = read_csv(path)

data = data %>% janitor::clean_names() %>%
  mutate(date = as.Date(date, format = '%m/%d/%y')) %>%
  mutate(date_diff = date - date[1])

max_cases = data %>%
  group_by(country_region, date_diff) %>%
  summarise(total_case = sum(confirmed_cases)) %>%
  mutate(date_diff = as.numeric(date_diff)) %>%
  group_by(country_region) %>% summarise(max_cases = max(total_case)) %>%
  rename("country_name" = "country_region")

end = left_join(para_final, max_cases, by = "country_name") %>%
  mutate(distance = (a - max_cases)/a) %>% arrange(-distance) %>%
  select(country_name, distance)

end %>% filter(distance > 0.8 | distance < 0) %>% knitr::kable(caption = "Approch the end of spread reg
```

Task2 Part1: Kmean

```
kmeans_EM <- function(X, k){
  # set.seed(10241024)
  set.seed(8)
  print("KNN begin")
  # X = apply(X,2,function(xc){scale(xc, center = TRUE, scale = TRUE)})
  p <- ncol(X)  # number of parameters
  n <- nrow(X)  # number of observations
  Delta <- 1
  iter <- 0
  itermax <- 30


  #initial
  centroid <- X[sample(nrow(X), k),]
  centroid_new <- centroid

  while (Delta > 1e-4 && iter <= itermax) {
    print(iter)
    # equivalent to E-step
    d <- sapply(1:k, function(c) {
      sapply(1:n, function(i)
      {sum((centroid[c,] - X[i,])^2) })
        }
      )
    cluster <- apply(d, 1, which.min)

    # equivalent to M-step
    centroid <- t(sapply(1:k, function(c)
      apply(matrix(X[cluster == c,], ncol = p),2, mean)))
      # apply(X[cluster == c,], 2, mean)))

    Delta <- sum((centroid - centroid_new)^2)
    iter <- iter + 1; centroid_new <- centroid
  }
  print("KNN end!!!")
  return(list(centroid = centroid, cluster = cluster))
}

# run K-means
path = "task2/para_final.csv"
para_all = read_csv(path) %>%
  filter(a != "NaN") %>%
  filter(b > 0) %>% as.data.frame()

X = para_all[,2:4]
X = apply(X,2,function(xc){scale(xc, center = TRUE, scale = TRUE)})
km <- kmeans_EM(X, 10)
pairs(X, lower.panel = NULL, col = km$cluster)
```

Task2 Part2: GMM

```
# pdf of multivariate normal distribution
multi_pdf = function(X, sigma) {

  X =  t(as.matrix(as.numeric(X)))
  l = vector.norm(X,2)
  nom = exp((-1 * (X %*% solve(sigma) %*% t(X))/2))
  demo = ((2*pi)^dim(X)[2] * det(sigma))^0.5

  # return((nom/demo))
  return((nom/demo)/l^3)
}

# mixture_gaussian model

mixture_gaussian = function(X, k=10) {
  # Given X: n * 3
  # k: num of classifications
  # Return classification n * 1
  # W: distribution of latent variable we want to maximize

  max_iter = 100
  iter_count = 0

  num_train = dim(X)[1] # n
  dms = dim(X)[2]       # 3
  mu = matrix(runif(dms * k, 0, 1), ncol = k) # dms * k

  sigma = array(rep(diag(rep(1, dms)),k), dim = c(dms, dms, k)) # 3*3*k matrix
  phi = matrix(rep(1 / k, k))
  W = matrix(rep(1 / k, num_train * k), ncol = k)
  # W = matrix(runif(num_train * k, 0,1), ncol = k)   # n*k

  while (iter_count < max_iter) {
    if (iter_count %% 10 == 0) {
      print(iter_count)
      # print(W)
      # print(mu)
    }
    iter_count = iter_count + 1

    # E step
    for (i in 1:num_train) {
      nom = rep(0,k)
      for (j in 1:k) {
        # multi_norm = 1 / (sum((X[i,] - mu[,j])^2) + 0.001)
         multi_norm = multi_pdf(X[i,] - mu[,j], sigma[,,j])
        # multi_norm = dmvnorm(X[i,] , mean = mu[,j], sigma = sigma[,,j], log = FALSE)

        nom[j] = multi_norm * phi[j]+1e-5
      }
      denom = sum(nom)
      W[i,] = nom / denom
    }

    # avoid W = 0
    W[W == 0] = 1e-5
    for (i in 1:num_train) {
      W[i,] = W[i,]/sum(W[i,])
    }

    # M step
    for (j in 1:k) {
      phi[j] = sum(W[,j])/num_train

      X_center = as.matrix(X - mu[,j])
      sigma[,,j] = t(W[,j] * X_center) %*% X_center
      # Update Mu
      for (t in 1:dms) {
        mu[t, j] = sum(W[,j] * X[,t]) / sum(W[,j])
      }
    }
  }

  return(list(pdf = phi,
              mu = mu,
              sigma = sigma))
}


group_assignment = function(test_df, k = 3) {
  res = mixture_gaussian(test_df, k)
  mu = res$mu
  sigma = res$sigma
  phi = res$pdf
  num_train = dim(test_df)[1]
  X = test_df

  W = matrix(rep(1 / k, num_train * k), ncol = k)
  for (i in 1:num_train) {
    nom = rep(0,k)
    for (j in 1:k) {
      # multi_norm = 1 / (sum((test_df[i,] - mu[,j])^2) + 0.001)
       multi_norm =  multi_pdf(X[i,] - mu[,j], sigma[,,j])
      # multi_norm = dmvnorm(X[i,] , mean = mu[,j], sigma = sigma[,,j], log = FALSE)
      nom[j] = multi_norm * phi[j]
    }
    denom = sum(nom)
    W[i,] = nom / denom
  }
  id = as.factor(c(1:num_train))

  result =
    W %>% data.frame(id) %>% pivot_longer(
      -id,
      values_to = "probability",
      names_to = "result")

  subject_list = split(result, result$id)

  max_index = lapply(subject_list, function(x) {return(x[which.max(x$probability),])})
  group_assignment = data.frame()
  for (i in 1:num_train) {
    group_assignment = bind_rows(group_assignment, max_index[[i]])
  }

  return(list(res$mu, group_assignment))
}
```

Task2 Part3: Run GMM

```
path = "para_final.csv"
para_all = read_csv(path) %>%
  filter(a != "NaN") %>%
  filter(b > 0) %>% as.data.frame()

name = as.list(para_all %>% select(country_name))$country_name
rownames(para_all) = name

para_data = para_all %>% select(a,b,c)

para_data = apply(para_data,2,function(xc) {scale(xc, center = TRUE, scale = TRUE)})

num_cluster = 5

res = group_assignment(para_data, k = num_cluster)
centroid = res[[1]]
cluster = res[[2]]

cluster_res = cbind(para_data, cluster) %>% data.frame() %>%
  select(a,b,c,result) %>% mutate(region = name)
```

Visulization

```
set.seed(2)
num_cluster = 5
res = group_assignment(para_data, k = num_cluster)
centroid = res[[1]]
cluster = res[[2]]

cluster_res = cbind(para_data, cluster) %>% data.frame() %>%
  select(a,b,c,result) %>% mutate(region = name)

centroid
```

```
colors <- brewer.pal(n = num_cluster, name = "Dark2")
scatter3d(x = cluster_res[,1], y = cluster_res[,2], z = cluster_res[,3],
          groups = factor(cluster_res[,4]),
          surface = FALSE, ellipsoid = TRUE, grid = FALSE,
          surface.col = colors,
          xlab = "a", ylab = "b",
          zlab = "c")


group = data.frame(res[[2]]$result) %>%
  mutate(res..2...result = str_replace(res..2...result, "X", "")) %>%
  mutate(res..2...result = as.numeric(res..2...result))
group = group$res..2...result


#setEPS()
#postscript("./plot/gmm_pairs.eps")
pairs(para_data, lower.panel = NULL, col = group)
#dev.off

colors<- brewer.pal(n = num_cluster, name = "Set3")

#setEPS()
#postscript("./plot/gmm_ggplot.eps")
s3d = scatterplot3d(cluster_res[,1:3], angle = 50, pch = 16, color=colors[as.numeric(group)])
legend("right",
       legend = c("class 1","class 2","class 3","class 4","class 5"
                  ), #"class 6","class 7","class 8","class 9","class 10"),
       inset = 0.1,
       xpd = TRUE,
       col =  colors, pch = 16)
#dev.off()
```

Mixed

```
world = map_data("world") %>%
  mutate(
    region = dplyr::recode(region, "USA" = "US")
  )
```

Kmean

```
X = para_all[,3:5]
X = apply(X,2,function(xc){scale(xc, center = TRUE, scale = TRUE)})

# run K-means

km <- kmeans_EM(X, num_cluster)
pairs(X, lower.panel = NULL, col = km$cluster)

colors <- brewer.pal(num_cluster,name = "Paired")
s3d = scatterplot3d(X, angle = 50, pch = 16,
                    color = colors[as.numeric(km$cluster)])
legend("right",
       legend = c("class 1","class 2","class 3","class 4","class 5"),
                  #, "class 6","class 7","class 8","class 9","class 10"),
       inset = 0.1,
       xpd = TRUE,
       col =  colors, pch = 16)
```

```
layout(matrix(c(2,1),1))
data.frame(para_data) %>%
  ggpairs(aes(color = as.factor(km$cluster),alpha = 0.4)) +
  labs(
    title = "K-means pairs plot"
  )

data.frame(para_data) %>%
  ggpairs(aes(color = as.factor(group),alpha = 0.4)) +
   labs(
    title = "GMM pairs plot"
  )
```

```
layout(matrix(c(2,1),1))
colors <- brewer.pal(num_cluster,name = "Paired")
s3d = scatterplot3d(X, angle = 50, pch = 16, color=colors[as.numeric(km$cluster)])
legend("right",
       legend = c("class 1","class 2","class 3","class 4","class 5"),
                  #, "class 6","class 7","class 8","class 9","class 10"),
       inset = 0.1,
       xpd = TRUE,
       cex = 0.8,
       col =  colors, pch = 16)

colors <- brewer.pal(num_cluster,name = "Paired")
s3d = scatterplot3d(X, angle = 50, pch = 16, color=colors[as.numeric(group)])
legend("right",
       legend = c("class 1","class 2","class 3","class 4","class 5"),
                  #, "class 6","class 7","class 8","class 9","class 10"),
       inset = 0.1,
       cex = 0.8,
       xpd = TRUE,
       col =  colors, pch = 16)
```

Map plots with new levels

```
gmm_centroid = t(centroid)
km_centroid = km$centroid

order_gmm = apply(gmm_centroid,2,rank)
order_km = apply(km_centroid,2,rank)
```

```
rgb_1 = c("88","AA","BB","DD","FF")
color_rgb_gmm = c(0)
for (i in 1:5){
  color_rgb_gmm[i] = str_c("#",rgb_1[order_gmm[,1]][i],rgb_1[order_gmm[,2]][i],rgb_1[order_gmm[,3]][i])
}
```

```
rgb_2 = c("88","A1","C5","DD","FF")
color_rgb_km = c(0)
for (i in 1:5){
  color_rgb_km[i] = str_c("#",rgb_2[order_km[,1]][i],rgb_2[order_km[,2]][i],rgb_2[order_km[,3]][i])
}
```

```
map_data = left_join(cluster_res, world,by = "region")
norm_plot =
  ggplot(data = map_data, aes(x = long, y = lat, group = group)) +
  geom_path(colour="grey80") +
  geom_polygon(aes(fill = factor(result))) +
  scale_fill_manual(name = "result", values = color_rgb_gmm, guide=FALSE) +
  theme_bw() +
  coord_fixed() +
  labs(
    title = "GMM world map"
  )

norm_plot
```

```
km_all = data.frame(X,
                    region = name,
                    result = sapply(km$cluster,function(num)str_c("X",num)))

map_data_km = left_join(km_all, world,by = "region")

norm_plot_km =
  ggplot(data = map_data_km, aes(x = long, y = lat, group = group)) +
  geom_path(colour="grey80") +
  geom_polygon(aes(fill = factor(result))) +
  scale_fill_manual(name = "result", values = color_rgb_km, guide = FALSE) +
  theme_bw() +
  coord_fixed() +
  labs(
    title = "K-means world map"
  )

norm_plot_km
```

```
norm_plot/norm_plot_km
```