DA 2: COVID 19 Analysis by: Chris Chiang (3101672)
#####Codes only, report in separate file#####

#1a

```
library(dplyr) ##please update it to the latest version
library(stringr)
library(zoo)
library(ggplot2)
library(devtools)
#install urbnmapr
##devtools::install_github("UrbanInstitute/urbnmapr")
library(urbnmapr)
install_github("Stat", force=TRUE)


##some functions we wrote for the analysis
source(file = "data_and_functions/functions_covid19.R")



##get the data from JHU CSSE, which contain the death and confirmed cases at county-level
base =
'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/'
death = 'time_series_covid19_deaths_'
confirm = 'time_series_covid19_confirmed_'

###up-to-date US death and confirmed cases
us_death = read.csv(paste0(base,death,"US.csv"))
us_confirm = read.csv(paste0(base,confirm,"US.csv"))

##dimension
dim(us_death)
dim(us_confirm)
##get the column names of the data
col_names = colnames(us_death)
all_dates = as.Date(paste0(str_sub(col_names[13:dim(us_death)[2]], 2, -1), "20"), tryFormats = c("%m.%d.%Y"))
##these are the dates from the data set
all_dates

##this site provide the confirmed state-level total test from which you can get the positive rate
covid19_project_url = "https://api.covidtracking.com/v1/states/daily.csv"

covid_19_project = read.csv(covid19_project_url)

covid_19_project$date = as.Date(as.character(covid_19_project$date), "%Y %m %d")
##note that this date starts from  the current day
covid_19_project$date


##nation level analysis
##nation level daily confirmed cases, plot and
nation_death = us_death %>%
  dplyr::filter(Country_Region == "US")

###observed confirmed cases
nation_confirmed = us_confirm %>%
  dplyr::filter(Country_Region == "US")


###the first row of the state death data set and you can see the format
nation_death[1,]
###get the cumulative death toll and confirmed cases
nation_death_sum = apply(nation_death[,12:dim(nation_death)[2]], 2, sum)

nation_confirmed_sum = apply(nation_confirmed[,12:dim(nation_confirmed)[2]], 2, sum)
```

```
##get  dates you want to analyze
start_date = as.Date("2020-4-01")

end_date = as.Date("2020-12-01")
##the deaths and confirmed cases for the state on the selected dates
nation_death_selected = nation_death_sum[1 + which(all_dates %in% seq.Date(start_date, end_date, by=1))]
nation_confirmed_selected = nation_confirmed_sum[which(all_dates %in% seq.Date(start_date, end_date, by=1))]

nation_death_selected=as.numeric(nation_death_selected)
nation_confirmed_selected=as.numeric(nation_confirmed_selected)

##plot cumulative confirmed cases and death
date_selected=seq.Date(start_date, end_date, by=1)
par(mfrow=c(1,2))
plot(date_selected,nation_confirmed_selected,xlab='date',ylab='cumulative observed confirmed cases',type='l')
plot(date_selected,nation_death_selected,xlab='date',ylab='cumulative death toll',type='l')
##close it

##daily increase between each date
daily_date_selected=date_selected[2:length(date_selected)]

##let's get the daily confirmed cases
nation_confirmed_selected_daily=nation_confirmed_selected[2:length(nation_confirmed_selected)]-nation_confirmed_selected[1:
(length(nation_confirmed_selected)-1)]
##create a data frame
daily_confirmed_nation_df = data.frame(date = daily_date_selected, value = nation_confirmed_selected_daily)

##let's get the daily death cases
nation_death_selected_daily=nation_death_selected[2:length(nation_death_selected)]-nation_death_selected[1:(length(nation_d
eath_selected)-1)]
##create a data frame
daily_death_nation_df = data.frame(date = daily_date_selected, value = nation_death_selected_daily)

###daily confirmed cases in US
daily_confirmed_nation_df %>%
  ggplot(aes(x=date, y=value)) +
  geom_bar(stat = 'identity', color="white",  fill="#ff8540", width = 1) +
  ylab("Daily confirmed cases in  US")+
  xlab("Date")+
  theme(text = element_text(size = 20),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 15),
        legend.key.width=unit(1,"cm"),
        axis.text.y = element_text(angle=90, hjust=1))

###daily death in US
daily_death_nation_df %>%
  ggplot(aes(x=date, y=value)) +
  geom_bar(stat = 'identity', color="white",  fill="#0000FF", width = 1) +
  ylab("Daily Death cases in US")+
  xlab("Date")+
  theme(text = element_text(size = 20),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 15),
        legend.key.width=unit(1,"cm"),
        axis.text.y = element_text(angle=90, hjust=1))

##let's obtain a seven-day average of the smoothed version of the confirmed cases and deaths

nation_confirmed_selected_daily_avg = data_seven_day_smoothing(nation_confirmed_selected_daily)

daily_confirmed_nation_smoothed_df = data.frame(date = daily_date_selected, value = nation_confirmed_selected_daily_avg)

nation_death_selected_daily_avg = data_seven_day_smoothing(nation_death_selected_daily)

daily_death_nation_smoothed_df = data.frame(date = daily_date_selected, value = nation_death_selected_daily_avg)

##plot the smoothed version
```

```
###daily confirmed cases in US
daily_confirmed_nation_smoothed_df %>%
  ggplot(aes(x=date, y=value)) +
  geom_bar(stat = 'identity', color="white",  fill="#ff8540", width = 1) +
  ylab("7-day averaged daily confirmed cases in  US")+
  xlab("Date")+
  theme(text = element_text(size = 20),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 15),
        legend.key.width=unit(1,"cm"),
        axis.text.y = element_text(angle=90, hjust=1))

###daily death in US
daily_death_nation_smoothed_df %>%
  ggplot(aes(x=date, y=value)) +
  geom_bar(stat = 'identity', color="white",  fill="#0000FF", width = 1) +
  ylab("7-day averaged daily death cases in US")+
  xlab("Date")+
  theme(text = element_text(size = 20),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 15),
        legend.key.width=unit(1,"cm"),
        axis.text.y = element_text(angle=90, hjust=1))


##get the test positive rates

############this is state-level positive rate
###deal with positive rate


nation_test = covid_19_project %>%
  dplyr::select(date, state,totalTestResultsIncrease, positiveIncrease)



nation_test_aggregated = nation_test %>%
  group_by(date) %>%
  summarise_each(funs(sum), positiveIncrease, totalTestResultsIncrease)


nation_test_aggregated$positiveIncrease_7_day_avg = data_seven_day_smoothing(nation_test_aggregated$positiveIncrease)

nation_test_aggregated$totalTestResultsIncrease_7_day_avg =
data_seven_day_smoothing(nation_test_aggregated$totalTestResultsIncrease)

nation_test_aggregated$positive_rate = nation_test_aggregated$positiveIncrease_7_day_avg /
nation_test_aggregated$totalTestResultsIncrease_7_day_avg


# ##reverse the sequence because it start from the current date
# nation_daily_test_selected=rev(us_test_PositiveRateus_test_PositiveRate[nation_test_aggregated$date>=(start_date) &
nation_test_aggregated$date<=end_date])

###let's smooth it and get the seven day average
us_test_daily_test_smoothed = data_seven_day_smoothing(nation_test_aggregated$totalTestResultsIncrease)
us_test_daily_positive_smoothed  = data_seven_day_smoothing(nation_test_aggregated$positiveIncrease)


##note that the following sequences start from the latest day
us_test_PositiveRate_smoothed = us_test_daily_positive_smoothed / us_test_daily_test_smoothed

us_test_PositiveRate_smoothed_selected=us_test_PositiveRate_smoothed[nation_test_aggregated$date>=(start_date) &
nation_test_aggregated$date<=end_date]

###plot  the smoothed positive rates
plot(date_selected, us_test_PositiveRate_smoothed_selected,type='l',xlab='date',ylab='7-day avg daily positive rate')
```

#1b

(Change the “end_date” for each graph)

```
end_date = as.Date("2020-10-30")
###Let's see whether we can make a map for CA about the confirmed cases over county popultation
### at a particular date

# set the date for map the end date you selected before
date_for_map = end_date

##cleaning
us_confirm_death_clean = clean_JHU_data_for_map(us_confirm, us_death)

us_confirm_clean = us_confirm_death_clean[[1]]
us_death_clean = us_confirm_death_clean[[2]]

# calculate the daily confirmed cases for all US counties
us_daily_confirm_clean = us_confirm_clean
us_daily_confirm_clean[,12] = NA
us_daily_confirm_clean[,13:dim(us_confirm_clean)[2]] = t(apply(us_confirm_clean[12:dim(us_confirm_clean)[2]], 1, diff))

##you can make the confirmed cases to be zero if it is smaller than zero
#us_daily_confirm_clean[(us_daily_confirm_clean)<0] = 0

# calculate the daily death toll for all US counties
us_daily_death_clean = us_death_clean
us_daily_death_clean[,13] = NA
us_daily_death_clean[,14:dim(us_death_clean)[2]] = t(apply(us_death_clean[13:dim(us_death_clean)[2]], 1, diff))

##you can make the daily death cases to be zero if it is smaller than zero
#us_daily_death_clean[us_daily_death_clean<0] = 0

# select the daily confirm cases for CA counties
state_daily_confirm = us_daily_confirm_clean%>%
  filter(Province_State == state_name)

# This shape file contains the coordinates for county boundaries
state_shape_data = counties %>%
  filter(state_name == state_name)

# get the population for CA counties
state_population = us_daily_death_clean %>%
  filter(Province_State == state_name)%>%
  dplyr::select(Population)
state_population = as.numeric(state_population$Population)

# extract the daily confirmed cases on the selected date date_for_map
state_daily_confirm_selected = state_daily_confirm[, c(1:11, 11+ which(all_dates == date_for_map))]

# the Ratio = daily confirmed cases / population
state_daily_confirm_rate_selected = state_daily_confirm_selected
state_daily_confirm_rate_selected[,12] = state_daily_confirm_selected[,12]/state_population
state_daily_confirm_rate_selected[,12][state_daily_confirm_rate_selected[,12]==0]=NA
colnames(state_daily_confirm_rate_selected)[12] = "Ratio"

##This is  the county and rate
state_daily_confirm_rate_selected[,c(11,12)]

# joint the daily confirmed cases with the shape file
state_daily_confirm_rate_selected_joint <- left_join(state_daily_confirm_rate_selected, state_shape_data, by = "county_fips")

# find the lower and upper limits of the ratio
range(state_daily_confirm_rate_selected$Ratio*100, na.rm = T)

state_daily_confirm_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group, fill = Ratio*100)) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90")+
  geom_polygon(col = "black") +
```

```
  coord_map(projection = "albers", lat0 = 10, lat1 = 45) +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("Daily confirmed cases / population in ", state_name, ", ", date_for_map))+
  xlab("lon") +ylab("lat" )
##close it
#limits

###Let's look at 7-day daily average versus county population

# calculate the 7-day averaged daily confirm cases
state_daily_confirm_avg = state_daily_confirm
state_daily_confirm_avg[, 13:dim(state_daily_confirm)[2]] = t(apply(state_daily_confirm[,13:dim(state_daily_confirm)[2]], 1,
data_seven_day_smoothing))

# extract the averaged daily confirmed cases on the selected date
state_daily_confirm_avg_selected = state_daily_confirm_avg[, c(1:11, 11+ which(all_dates == date_for_map))]

# the Ratio = averaged daily confirmed cases / population
state_daily_confirm_rate_avg_selected = state_daily_confirm_avg_selected
state_daily_confirm_rate_avg_selected[, 12] = state_daily_confirm_avg_selected[, 12]/state_population
state_daily_confirm_rate_avg_selected[,12][state_daily_confirm_rate_avg_selected[,12]==0]=NA
colnames(state_daily_confirm_rate_avg_selected)[12] = "Ratio"

## county and rate
state_daily_confirm_rate_avg_selected[,c(11,12)]

index_largest=which(state_daily_confirm_rate_avg_selected[,c(12)]==max(state_daily_confirm_rate_avg_selected[,c(12)],na.rm=
T))
state_daily_confirm_rate_avg_selected[index_largest,c(11,12)]

# joint the averaged daily confirmed cases with the shape file
state_daily_confirm_rate_avg_selected_joint <- left_join(state_daily_confirm_rate_avg_selected, state_shape_data, by =
"county_fips")


# find the lower and upper limits of the ratio
range(state_daily_confirm_rate_avg_selected$Ratio*100, na.rm = T)

state_daily_confirm_rate_avg_selected_joint %>%
  ggplot(aes(long, lat, group = group, fill = Ratio*100)) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90")+
  geom_polygon(col = "black") +
  coord_map(projection = "albers", lat0 = 10, lat1 = 45) +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("7-day averaged daily confirmed cases / population in ", state_name, ", ", date_for_map))+
  xlab("lon") +ylab("lat" )
##close it
```

#2a

```
start_date_Mi = as.Date("2020-7-1")
end_date_Mi = as.Date("2020-11-1")

date_selected=seq.Date(start_date_Mi, end_date_Mi, by=1)

Mi_confirmed = us_confirm %>%
  filter(Province_State == "Michigan") %>%
  select(starts_with("x"))

Mi_confirmed_sum = apply(Mi_confirmed, 2, sum)

Mi_confirmed_selected = Mi_confirmed_sum[which(all_dates %in% seq.Date(start_date_Mi, end_date_Mi, by=1))]
Mi_confirmed_selected=as.numeric(Mi_confirmed_selected)

Mi_confirmed_selected_daily=Mi_confirmed_selected[2:length(Mi_confirmed_selected)]-Mi_confirmed_selected[1:(length(Mi_con
```

```
firmed_selected)-1)]
daily_date_selected=date_selected[2:length(date_selected)]

daily_date_selected=date_selected[2:length(date_selected)]

# 7-day average of daily cases
Mi_confirmed_selected_daily_avg = data_seven_day_smoothing(Mi_confirmed_selected_daily)
daily_confirmed_Mi_avg_df = data.frame(date = daily_date_selected, value = Mi_confirmed_selected_daily_avg)

# Michigan, 7-day avg confirmed cases
daily_confirmed_Mi_avg_df %>%
  ggplot(aes(x=date, y=value)) +
  geom_bar(stat = 'identity', color="white",  fill="#ff8540", width = 1) +
  ylab("7-day Averaged Daily Confirmed Cases in Michigan")+
  xlab("Date")+
  theme(text = element_text(size = 20),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 15),
        legend.key.width=unit(1,"cm"),
        axis.text.y = element_text(angle=90, hjust=1))

Mi_test = covid_19_project %>%
  dplyr::select(date, state, totalTestResultsIncrease, positiveIncrease) %>%
  filter(state == "MI")

Mi_test_aggregated = Mi_test %>%
  group_by(date) %>%
  summarise_each(funs(sum), positiveIncrease, totalTestResultsIncrease)

Mi_test_aggregated$positiveIncrease_7_day_avg = data_seven_day_smoothing(Mi_test_aggregated$positiveIncrease)

Mi_test_aggregated$totalTestResultsIncrease_7_day_avg =
data_seven_day_smoothing(Mi_test_aggregated$totalTestResultsIncrease)

Mi_test_aggregated$positive_rate = Mi_test_aggregated$positiveIncrease_7_day_avg /
Mi_test_aggregated$totalTestResultsIncrease_7_day_avg


###let's smooth it and get the seven day average
Mi_test_daily_test_smoothed = data_seven_day_smoothing(Mi_test_aggregated$totalTestResultsIncrease)
Mi_test_daily_positive_smoothed  = data_seven_day_smoothing(Mi_test_aggregated$positiveIncrease)

##note that the following sequences start from the latest day
Mi_test_PositiveRate_smoothed = Mi_test_daily_positive_smoothed / Mi_test_daily_test_smoothed

Mi_test_PositiveRate_smoothed_selected=Mi_test_PositiveRate_smoothed[Mi_test_aggregated$date>=(start_date_Mi) &
Mi_test_aggregated$date<=end_date_Mi]

###plot the smoothed positive rates
plot(date_selected, Mi_test_PositiveRate_smoothed_selected,type='l',xlab='date',ylab='7-day averaged daily positive rate for MI')
```

#2b and c

(Using Michigan as the state. And change the “end_date” for each graph shown)

```
###Let's see whether we can make a map for CA about the confirmed cases over county population
### at a particular date

# set the date for map the end date you selected before
date_for_map = end_date

##cleaning
us_confirm_death_clean = clean_JHU_data_for_map(us_confirm, us_death)
```

```
us_confirm_clean = us_confirm_death_clean[[1]]
us_death_clean = us_confirm_death_clean[[2]]

# calculate the daily confirmed cases for all US counties
us_daily_confirm_clean = us_confirm_clean
us_daily_confirm_clean[,12] = NA
us_daily_confirm_clean[,13:dim(us_confirm_clean)[2]] = t(apply(us_confirm_clean[12:dim(us_confirm_clean)[2]], 1, diff))

##you can make the confirmed cases to be zero if it is smaller than zero
#us_daily_confirm_clean[(us_daily_confirm_clean)<0] = 0

# calculate the daily death toll for all US counties
us_daily_death_clean = us_death_clean
us_daily_death_clean[,13] = NA
us_daily_death_clean[,14:dim(us_death_clean)[2]] = t(apply(us_death_clean[13:dim(us_death_clean)[2]], 1, diff))

##you can make the daily death cases to be zero if it is smaller than zero
#us_daily_death_clean[us_daily_death_clean<0] = 0

# select the daily confirm cases for CA counties
state_daily_confirm = us_daily_confirm_clean%>%
  filter(Province_State == state_name)

# This shape file contains the coordinates for county boundaries
state_shape_data = counties %>%
  filter(state_name == state_name)

# get the population for CA counties
state_population = us_daily_death_clean %>%
  filter(Province_State == state_name)%>%
  dplyr::select(Population)
state_population = as.numeric(state_population$Population)

# extract the daily confirmed cases on the selected date date_for_map
state_daily_confirm_selected = state_daily_confirm[, c(1:11, 11+ which(all_dates == date_for_map))]

# the Ratio = daily confirmed cases / population
state_daily_confirm_rate_selected = state_daily_confirm_selected
state_daily_confirm_rate_selected[,12] = state_daily_confirm_selected[,12]/state_population
state_daily_confirm_rate_selected[,12][state_daily_confirm_rate_selected[,12]==0]=NA
colnames(state_daily_confirm_rate_selected)[12] = "Ratio"

##This is  the county and rate
state_daily_confirm_rate_selected[,c(11,12)]

# joint the daily confirmed cases with the shape file
state_daily_confirm_rate_selected_joint <- left_join(state_daily_confirm_rate_selected, state_shape_data, by = "county_fips")

# find the lower and upper limits of the ratio
range(state_daily_confirm_rate_selected$Ratio*100, na.rm = T)

state_daily_confirm_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group, fill = Ratio*100)) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90")+
  geom_polygon(col = "black") +
  coord_map(projection = "albers", lat0 = 10, lat1 = 45) +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("Daily confirmed cases / population in ", state_name, ", ", date_for_map))+
  xlab("lon") +ylab("lat" )
##close it
#limits

###Let's look at 7-day daily average versus county population

# calculate the 7-day averaged daily confirm cases
state_daily_confirm_avg = state_daily_confirm
state_daily_confirm_avg[, 13:dim(state_daily_confirm)[2]] = t(apply(state_daily_confirm[,13:dim(state_daily_confirm)[2]], 1,
data_seven_day_smoothing))
```

```
# extract the averaged daily confirmed cases on the selected date
state_daily_confirm_avg_selected = state_daily_confirm_avg[, c(1:11, 11+ which(all_dates == date_for_map))]

# the Ratio = averaged daily confirmed cases / population
state_daily_confirm_rate_avg_selected = state_daily_confirm_avg_selected
state_daily_confirm_rate_avg_selected[, 12] = state_daily_confirm_avg_selected[, 12]/state_population
state_daily_confirm_rate_avg_selected[,12][state_daily_confirm_rate_avg_selected[,12]==0]=NA
colnames(state_daily_confirm_rate_avg_selected)[12] = "Ratio"

## county and rate
state_daily_confirm_rate_avg_selected[,c(11,12)]

index_largest=which(state_daily_confirm_rate_avg_selected[,c(12)]==max(state_daily_confirm_rate_avg_selected[,c(12)],na.rm=
T))
state_daily_confirm_rate_avg_selected[index_largest,c(11,12)]

# joint the averaged daily confirmed cases with the shape file
state_daily_confirm_rate_avg_selected_joint <- left_join(state_daily_confirm_rate_avg_selected, state_shape_data, by =
"county_fips")


# find the lower and upper limits of the ratio
range(state_daily_confirm_rate_avg_selected$Ratio*100, na.rm = T)

state_daily_confirm_rate_avg_selected_joint %>%
  ggplot(aes(long, lat, group = group, fill = Ratio*100)) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90")+
  geom_polygon(col = "black") +
  coord_map(projection = "albers", lat0 = 10, lat1 = 45) +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("7-day averaged daily confirmed cases / population in ", state_name, ", ", date_for_map))+
  xlab("lon") +ylab("lat" )
##close it
```

#3a

```
##FL level analysis
##FL level daily confirmed cases, plot and
FL_death = us_death %>%
  filter(Province_State == "Florida") %>%
  select(starts_with("x"))

###observed confirmed cases
FL_confirmed = us_confirm %>%
  filter(Province_State == "Florida") %>%
  select(starts_with("x"))


###the first row of the state death data set and you can see the format
FL_death[1,]
###get the cumulative death toll and confirmed cases
FL_death_sum = apply(FL_death[,12:dim(FL_death)[2]], 2, sum)

FL_confirmed_sum = apply(FL_confirmed[,12:dim(FL_confirmed)[2]], 2, sum)

##get  dates you want to analyze
start_date = as.Date("2020-7-01")

end_date = as.Date("2020-11-01")



##the deaths and confirmed cases for the state on the selected dates
FL_death_selected = FL_death_sum[1 + which(all_dates %in% seq.Date(start_date, end_date, by=1))]
```

```
FL_confirmed_selected = FL_confirmed_sum[which(all_dates %in% seq.Date(start_date, end_date, by=1))]

FL_death_selected=as.numeric(FL_death_selected)
FL_confirmed_selected=as.numeric(FL_confirmed_selected)

##plot cumulative confirmed cases and death
date_selected=seq.Date(start_date, end_date, by=1)
par(mfrow=c(1,2))
plot(date_selected,FL_confirmed_selected,xlab='date',ylab='cumulative observed confirmed cases',type='l')
plot(date_selected,FL_death_selected,xlab='date',ylab='cumulative death toll',type='l')
##close it
```

FOR THE POSITIVE RATE PLOT: SAME AS CODE FOR 2A EXCEPT CHANGE STATE TO FLORIDA

#3b (same as 2bc but changing the state to Florida)

#3c (looking at previous graphs already shown)

#4a (refer to #3a but changing the state to CA)

#4c(same as 2bc but changing the state to CA and the end_date)

#4d

```
date_seq = seq.Date(start_date, end_date, by=1)

plot(N-param_record_approx_for_beta[1,]~date_seq, type="l", col="blue", xlab = "Date", ylab = "Infective Cases", main =
paste0(county_names[each_index], ", population=", round(N/10^6,2),"M", ", Ratio = ", round(ratio_real,3)))
lines(confirm_selected~date_seq, type="l", col="red")
legend("topleft", legend = c("Estimated Confirmed Cases", "Observed Confirmed Cases"), lty = c(1,1), col = c("red", "blue"))
```

#4e

```
###fitted death and forecast death
date_seq_all = seq.Date(start_date, end_date+prediction_length, by=1)

ylimit_death_all = c(min(param_record_approx_all[4,], death_selected), max(param_record_approx_all[4,], death_selected))

plot(param_record_approx_all[4,]~date_seq_all,ylim = ylimit_death_all, type="l", col="blue", xlab = "Date", ylab = "Death Cases",
main = paste0(county_names[each_index], ", population=", round(N/10^6,2),"M", ", Ratio = ", round(ratio_real,3)))
lines(death_selected~date_seq, col = "red")
lines(date_seq_all[(n+1):(n+prediction_length-1) ],death_with_county_names_all[[1]][each_index,(n+1):(n+prediction_length-1) ],
      col = "red",lty=2)
abline(v = end_date+1, lty=2)
legend("topleft", legend = c("Observed death toll", "Estimated death toll","Held-out death toll"), lty = c(1,1,2), col = c("red",
"blue",'red'))
```

#4g,h

(For part g, gamma_new is 1/ 4.75 for 5% reduction. For part h, it's 1/ 4.5 for 10% reduction)

```
###work on some simulation if the infectious period decreases
gamma_new = 1/ 4.75 ###suppose it changes from 5 day to 4.75 days

param_record_approx_for_beta_new = matrix(0, 5, n) # 5 rows: S_t, I_t, R_t, D_t, C_t
```

```
param_record_approx_for_beta_new[,1] = init_for_beta
param_record_approx_for_beta_new[1,] = S_t_seq

# record the value of transmission rate
# approx_beta_seq_new = rep(0, n-1)
# we should fix the beta when we change the gamma parameter

# iterative approach for calculating the seq of compartments in SIRDC
for (i in 1:(n-1)){
  S_t_1 = param_record_approx_for_beta_new[1,i]
  I_t_1 = param_record_approx_for_beta_new[2,i]
  R_t_1 = param_record_approx_for_beta_new[3,i]
  D_t_1 = param_record_approx_for_beta_new[4,i]
  C_t_1 = param_record_approx_for_beta_new[5,i]

  beta_t_1_2 = approx_beta_seq[i]

  if(I_t_1<1){
    I_t_1 = 1
  }

  S_t_2 = uniroot(find_root_S_t_2, c(0, N), tol = 0.0001, param = c(S_t_1, beta_t_1_2, I_t_1), N = N, gamma=gamma_new)
  I_t_2 = I_t_1 * exp(beta_t_1_2*(S_t_1 + S_t_2$root)/(2*N) - gamma_new)
  R_t_2 = (2-theta)/(2+theta)*R_t_1 + gamma_new/(2+theta)*(I_t_1+I_t_2)
  D_t_2 = D_t_1 + delta*theta*(R_t_1+R_t_2)/2
  C_t_2 = C_t_1 + (1-delta)*theta*(R_t_1+R_t_2)/2

  param_record_approx_for_beta_new[, i+1] = c(S_t_2$root, I_t_2, R_t_2, D_t_2, C_t_2)
}

# calculate the smoothed transmission rate using 7 day average
# approx_beta_seq_smoothed_new = rollapply(approx_beta_seq_new, width = 7, by = 1, FUN = mean, align = "left")


###plot the simulated death
plot(param_record_approx_for_beta_new[4,]~date_seq,ylim = ylimit_death, type="l", col="blue", xlab = "Date", ylab = "Death
Cases", main = paste0(county_names[each_index], ", population=", round(N/10^6,2),"M", ", Ratio = ", round(ratio_real,3)))
lines(death_selected~date_seq, col = "red")
legend("topleft", legend = c("Observed death toll", "Simulated death toll"), lty = c(1,1), col = c("red", "blue"))
```