Data Academy 2

Part I

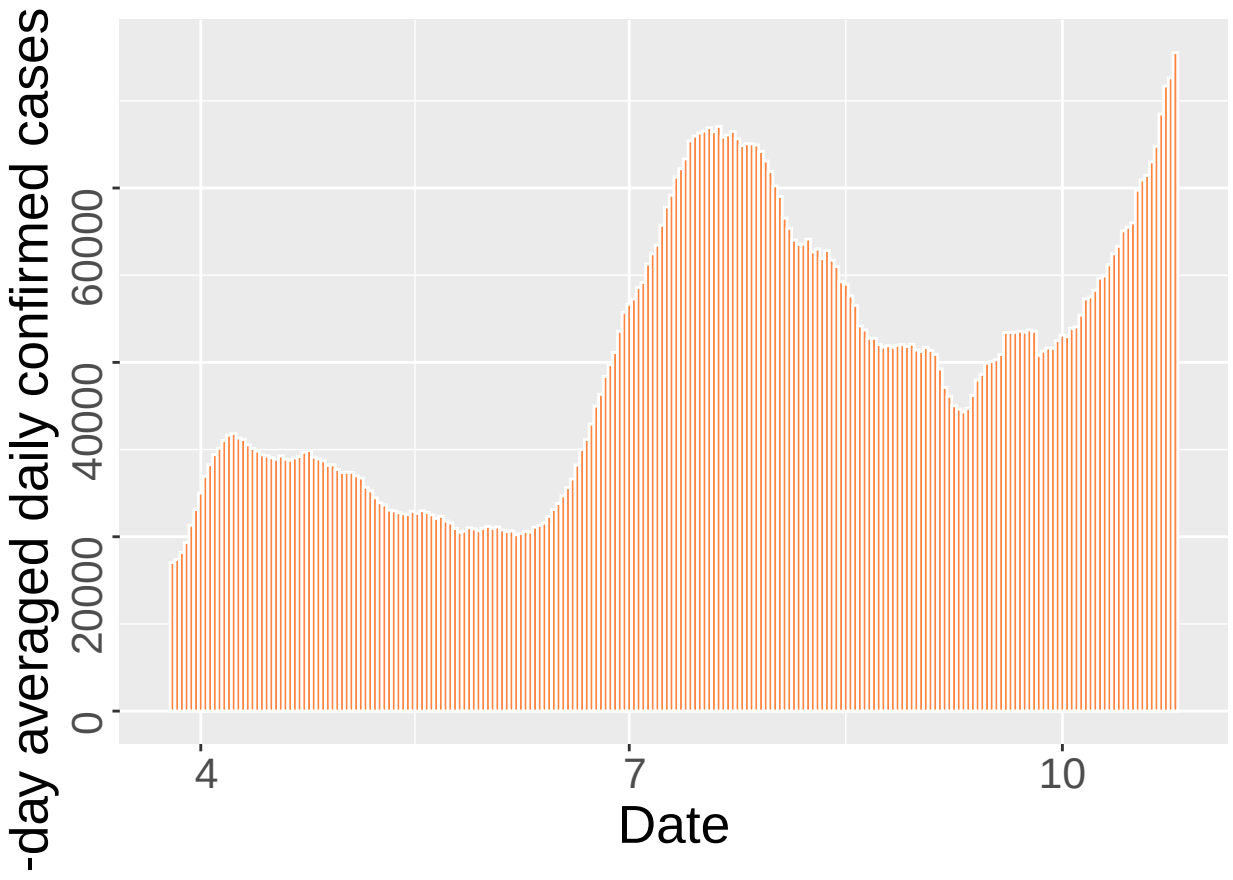
Question 1

Question 1.a

First, draw graphs from March 25 to Oct 25.

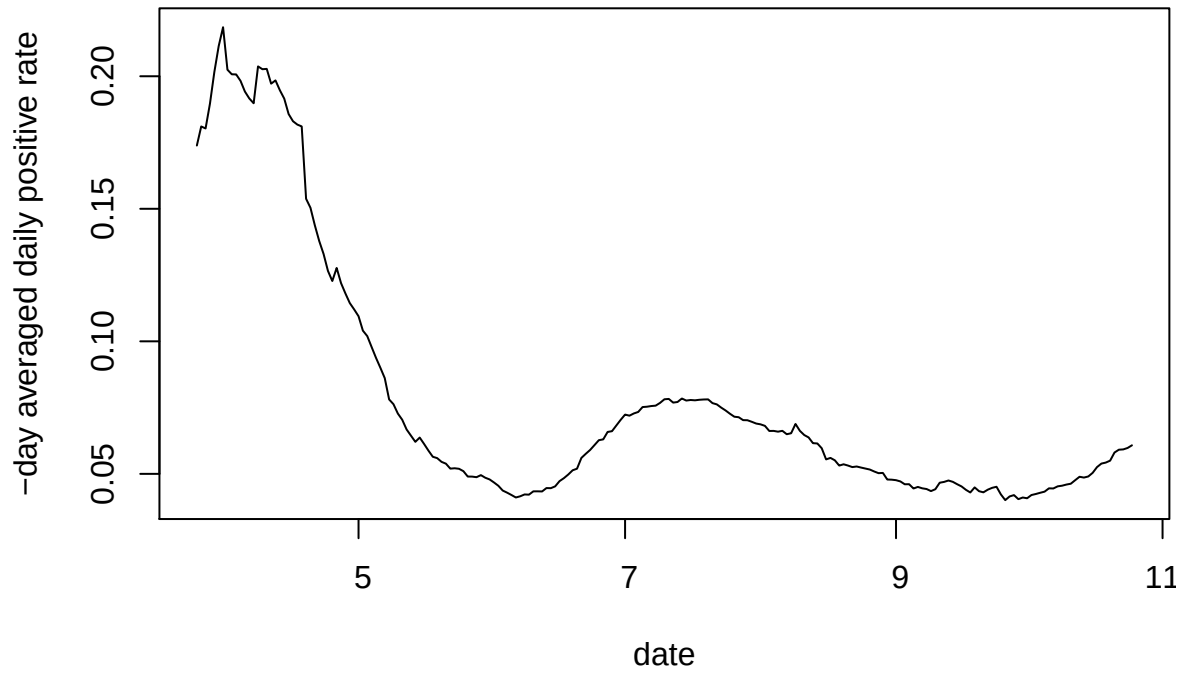
```
death = "time_series_covid19_deaths_"
us_death = read.csv(paste0(death, "US.csv"))
confirm = "time_series_covid19_confirmed_"
us_confirm = read.csv(paste0(confirm, "US.csv"))

## Get the column names of the data
col_names = colnames(us_confirm)
all_dates = as.Date(paste0(str_sub(col_names[13:dim(us_confirm)[2]],
    2, -1), "20"), tryFormats = c("%m.%d.%Y"))
covid19_project_url = "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")

nation_confirmed = us_confirm %>% dplyr::filter(Country_Region ==
    "US")
nation_confirmed_sum = apply(nation_confirmed[, 12:dim(nation_confirmed)[2]],
    2, sum)

start_date = as.Date("2020-3-25")
end_date = as.Date("2020-10-25")


nation_confirmed_selected = nation_confirmed_sum[which(all_dates %in%
    seq.Date(start_date, end_date, by = 1))]

nation_confirmed_selected = as.numeric(nation_confirmed_selected)
date_selected = seq.Date(start_date, end_date, by = 1)
daily_date_selected = date_selected[2:length(date_selected)]

## Daily confirmed cases
nation_confirmed_selected_daily = nation_confirmed_selected[2:length(nation_confirmed_selected)] -
    nation_confirmed_selected[1:(length(nation_confirmed_selected) -
        1)]

daily_confirmed_nation_df = data.frame(date = daily_date_selected,
    value = nation_confirmed_selected_daily)
```

```r
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)


### 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))
```




```r
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$positiveIncrease_7_day_avg = data_seven_day_smoothing(nation_test_aggregated$posi

nation_test_aggregated$totalTestResultsIncrease_7_day_avg = data_seven_day_smoothing(nation_test_aggrega

nation_test_aggregated$positive_rate = nation_test_aggregated$positiveIncrease_7_day_avg/nation_test_agg

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)
```

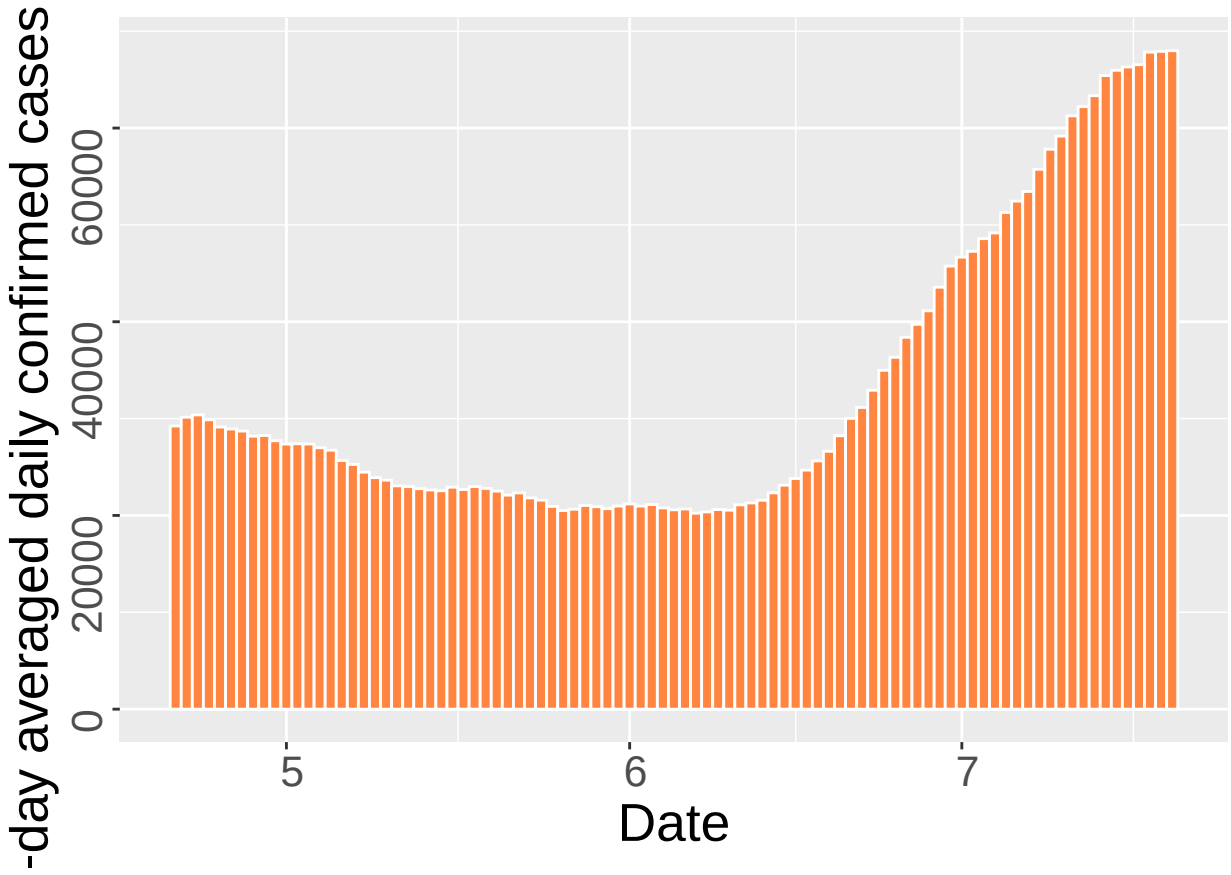
```
us_test_PositiveRate_smoothed = us_test_daily_positive_smoothed/us_test_daily_test_smoothed
us_test_PositiveRate_smoothed_selected = us_test_PositiveRate_smoothed[nation_test_aggregated$date >=
    (start_date) & nation_test_aggregated$date <= end_date]


plot(date_selected, us_test_PositiveRate_smoothed_selected,
    type = "l", xlab = "date", ylab = "-day averaged daily positive rate")
```



Then, draw graphs from Apr 20 and on July 20.

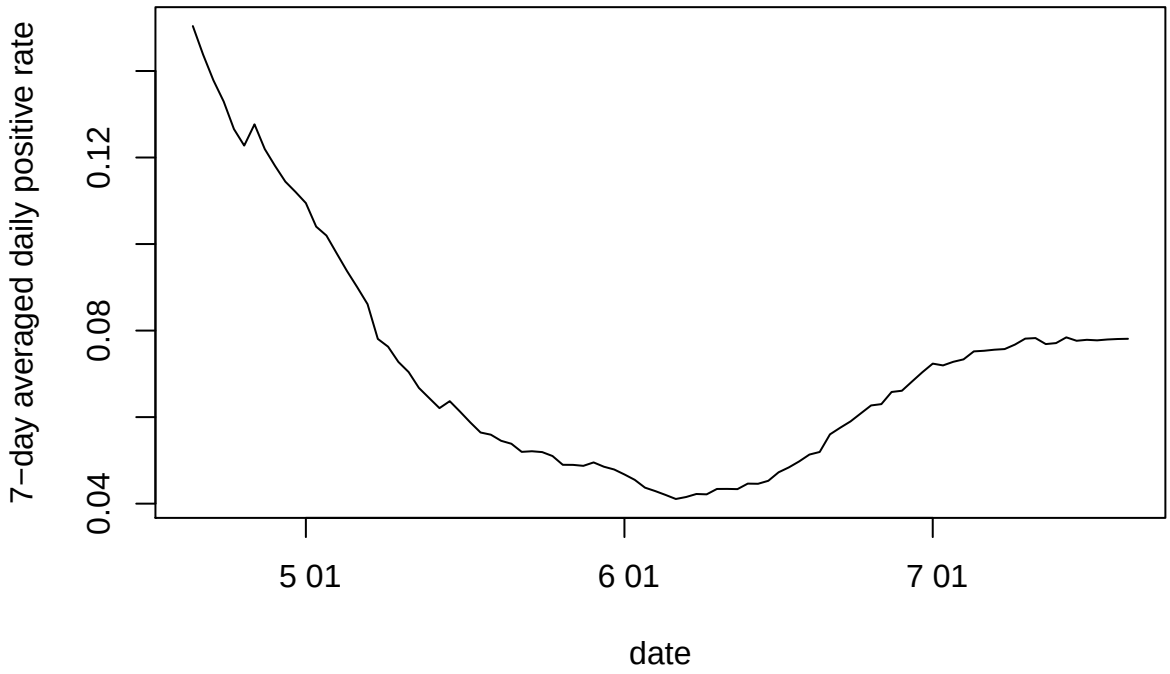
```
start_date = as.Date("2020-4-20")
end_date = as.Date("2020-07-20")


nation_confirmed_selected = nation_confirmed_sum[which(all_dates %in%
    seq.Date(start_date, end_date, by = 1))]
nation_confirmed_selected = as.numeric(nation_confirmed_selected)

date_selected = seq.Date(start_date, end_date, by = 1)
daily_date_selected = date_selected[2:length(date_selected)]

## Daily confirmed cases
nation_confirmed_selected_daily = nation_confirmed_selected[2:length(nation_confirmed_selected)] -
    nation_confirmed_selected[1:(length(nation_confirmed_selected) -
        1)]
daily_confirmed_nation_df = data.frame(date = daily_date_selected,
    value = nation_confirmed_selected_daily)

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)
```

```
### 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))
```




```
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$positiveIncrease_7_day_avg = data_seven_day_smoothing(nation_test_aggregated$posi

nation_test_aggregated$totalTestResultsIncrease_7_day_avg = data_seven_day_smoothing(nation_test_aggrega

nation_test_aggregated$positive_rate = nation_test_aggregated$positiveIncrease_7_day_avg/nation_test_agg

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)
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]
```

```r
plot(date_selected, us_test_PositiveRate_smoothed_selected,
    type = "l", xlab = "date", ylab = "7-day averaged daily positive rate")
```



The trend is similar. The two curves are almost aligned, i.e., when the number of positive cases increased, the positive rate also increased. However, the scale are different. It can be seen that, in October, the number of positive cases reached a new high, but the positive rate did not. Such deviation is not so severe in the period between April 20 and July 20.

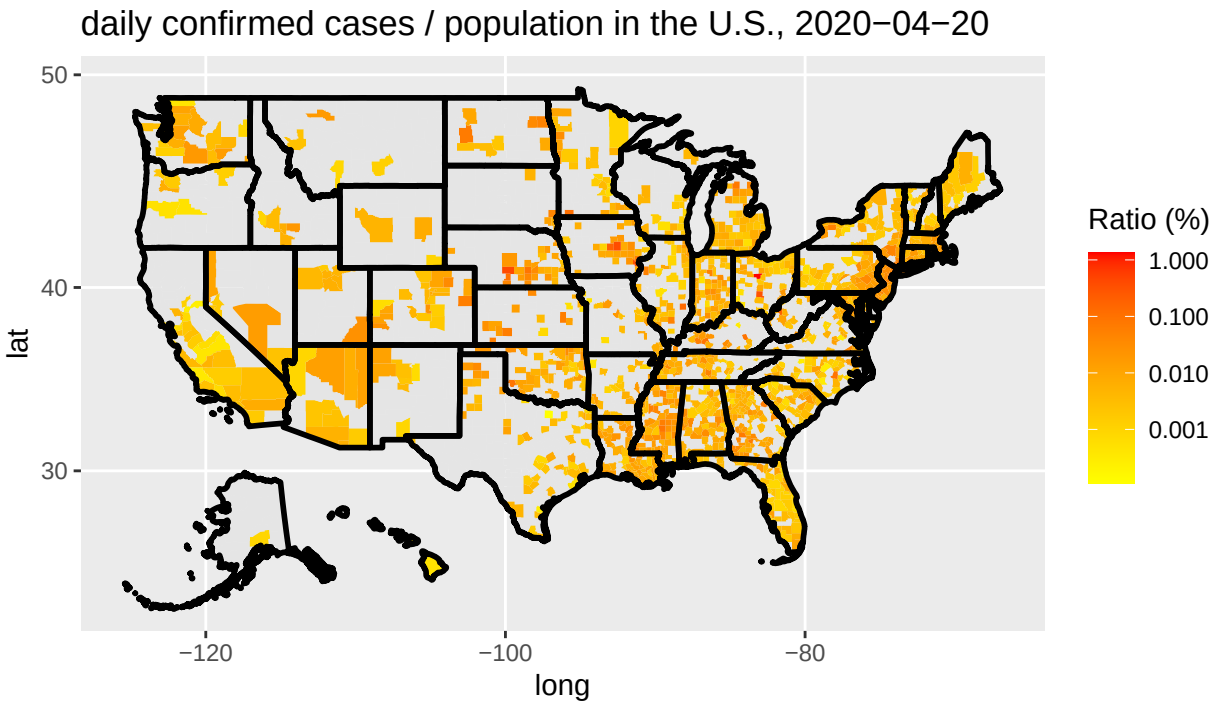
Reasons: 1) The number of tested cases increased

Question 1.b

Map on 2020-04-20:

```r
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]]

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
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
nation_population = us_daily_death_clean$Population

date_for_map = "2020-04-20"
# extract the daily confirmed cases for all US counties on the selected date
nation_daily_confirmed_selected = us_daily_confirm_clean[, c(1:11, 11+which(all_dates==date_for_map))]

# Ratio = daily confirmed cases / population
nation_daily_confirmed_rate_selected = nation_daily_confirmed_selected
nation_daily_confirmed_rate_selected[,12] = nation_daily_confirmed_selected[,12] / nation_population
nation_daily_confirmed_rate_selected[,12][nation_daily_confirmed_rate_selected[,12] == 0] = NA
```

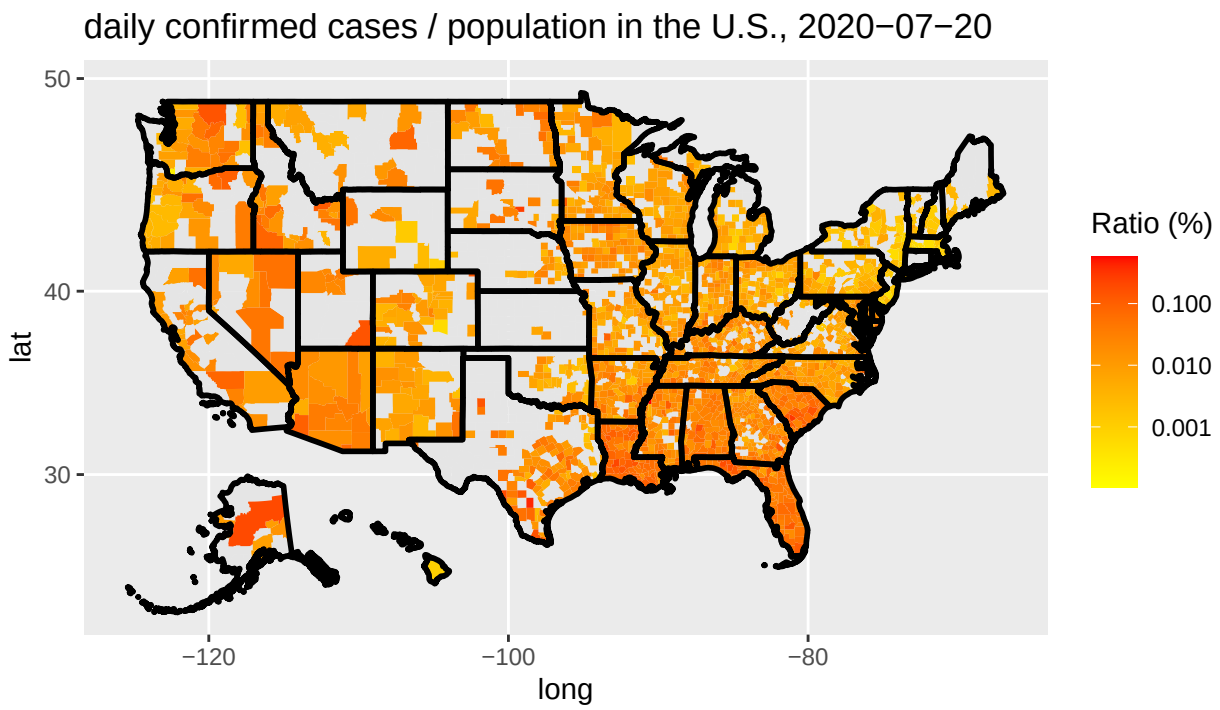
```
colnames(nation_daily_confirmed_rate_selected)[12] = "Ratio"
nation_daily_confirmed_rate_selected_joint <- left_join(nation_daily_confirmed_rate_selected, counties,
nation_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states, mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Map on 2020-07-20:

```
date_for_map = "2020-07-20"
# extract the daily confirmed cases for all US counties on the selected date
nation_daily_confirmed_selected = us_daily_confirm_clean[, c(1:11, 11+which(all_dates==date_for_map))]

# Ratio = daily confirmed cases / population
nation_daily_confirmed_rate_selected = nation_daily_confirmed_selected
nation_daily_confirmed_rate_selected[,12] = nation_daily_confirmed_selected[,12] / nation_population
nation_daily_confirmed_rate_selected[,12][nation_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(nation_daily_confirmed_rate_selected)[12] = "Ratio"
nation_daily_confirmed_rate_selected_joint <- left_join(nation_daily_confirmed_rate_selected, counties,
nation_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
```

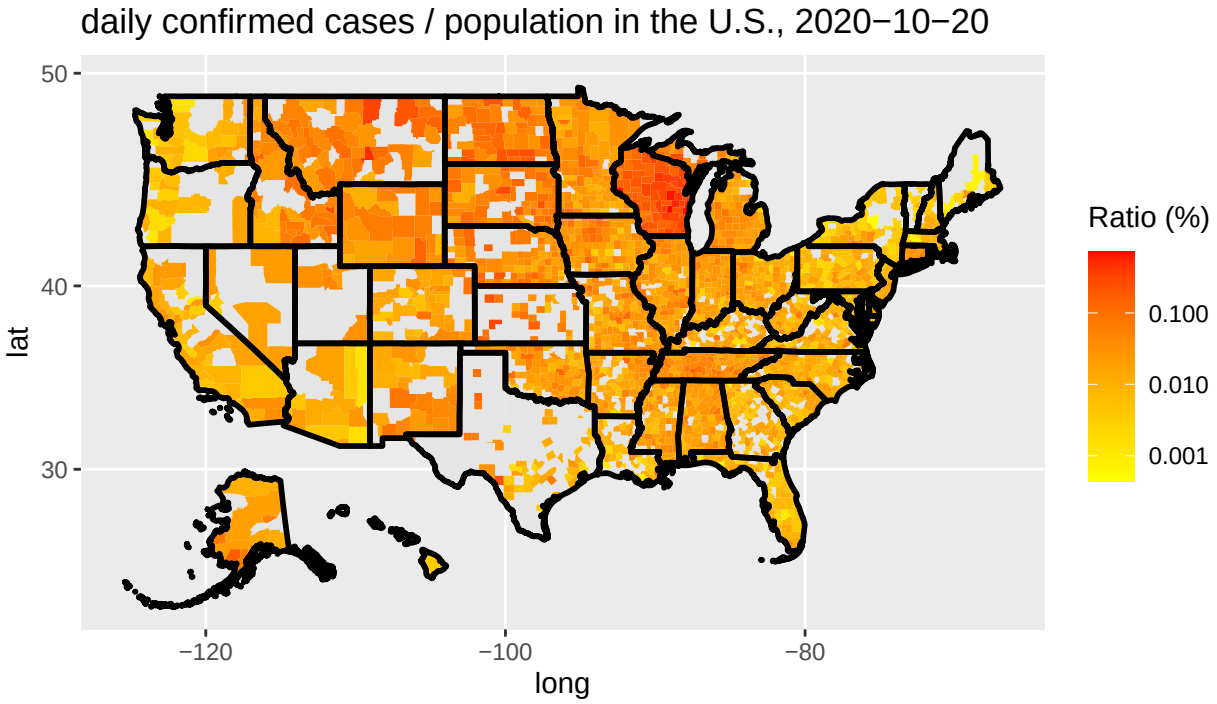
```
    data = urbnmapr::states, mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Map on 2020-10-20:

```
date_for_map = "2020-10-20"
# extract the daily confirmed cases for all US counties on the selected date
nation_daily_confirmed_selected = us_daily_confirm_clean[, c(1:11, 11+which(all_dates==date_for_map))]

# Ratio = daily confirmed cases / population
nation_daily_confirmed_rate_selected = nation_daily_confirmed_selected
nation_daily_confirmed_rate_selected[,12] = nation_daily_confirmed_selected[,12] / nation_population
nation_daily_confirmed_rate_selected[,12][nation_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(nation_daily_confirmed_rate_selected)[12] = "Ratio"
nation_daily_confirmed_rate_selected_joint <- left_join(nation_daily_confirmed_rate_selected, counties,
nation_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states, mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
```

```
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Question 1.c

It can be seen that the outbreak have spreaded to nearly all counties. In April, the disease was mainly located at the coast, but in October some other states showed an emerge of cases (e.g. Wisconsin, Montana).

Question 1.d

On April 20: Protect those states where there are no so many cases. Impose restrictions on states like Calinornia and New York to stop the further spread of virus.

On July 20: Impose more strict restriction on states with severe situation.

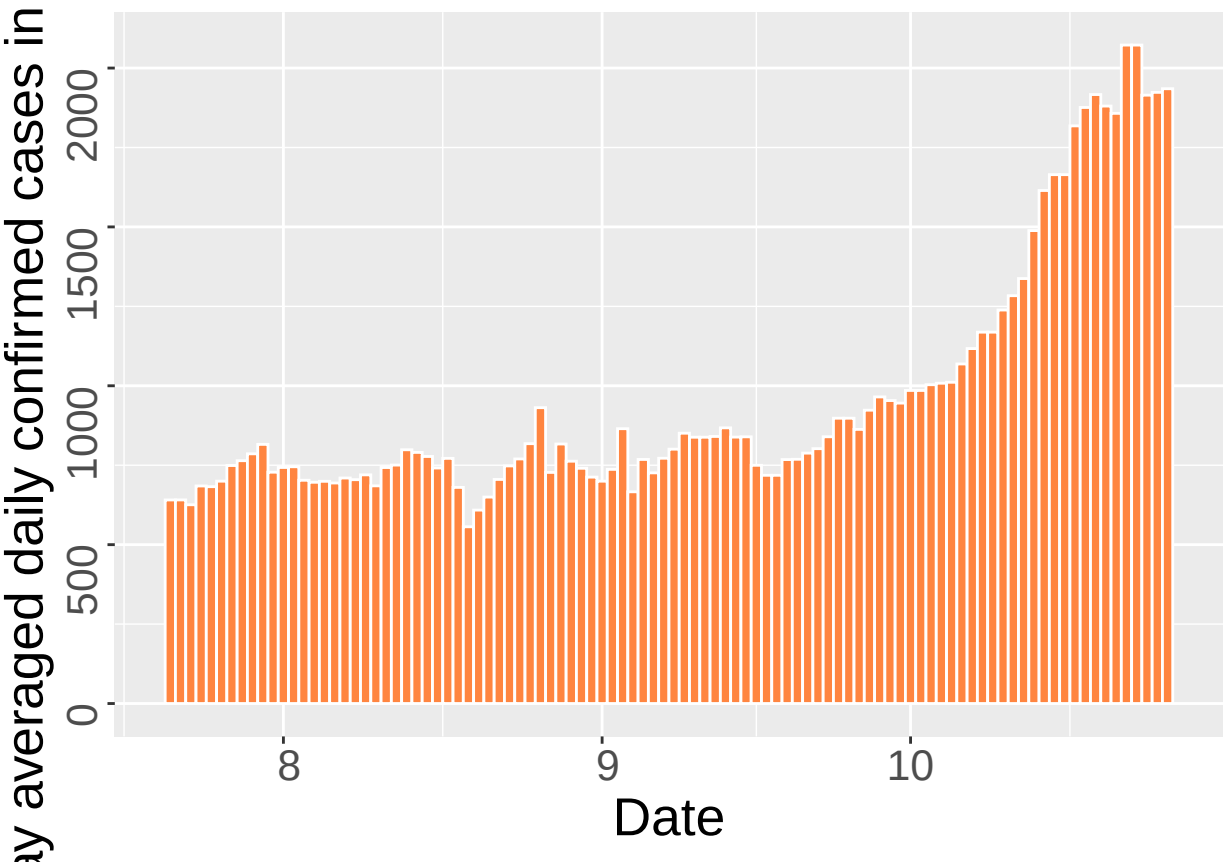
On October 20: Since the disease has spreaded across the nation, the only thing I can do is to ask everyone to protect themselves, try their best to stay at home, to social distance, and wait for vaccines.

Question 2

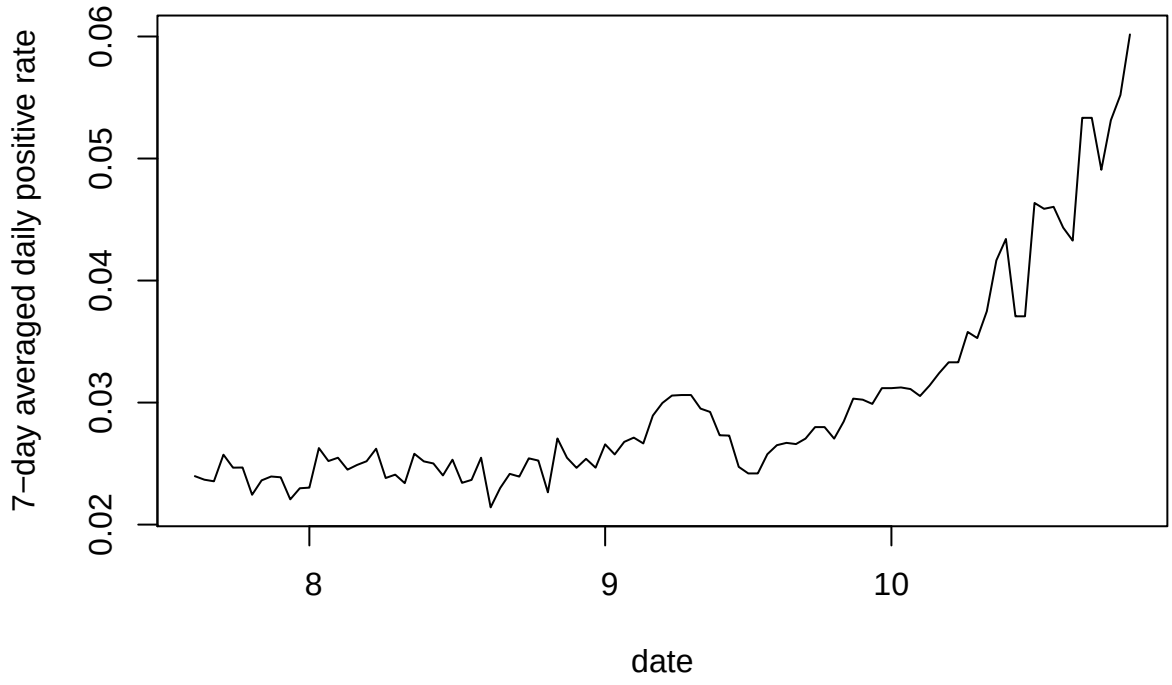
Question 2.a

```
start_date = as.Date("2020-7-20")
end_date = as.Date("2020-10-26")
date_selected = seq.Date(start_date, end_date, by = 1)
state_name = "Michigan"
county_name = "Michigan"
state_name_short = "MI"
state_confirmed = us_confirm %>% filter(Province_State ==
    state_name) %>% select(starts_with("x"))
```

```r
state_confirmed_sum = apply(state_confirmed, 2, sum)
state_confirmed_selected = state_confirmed_sum[which(all_dates %in%
    seq.Date(start_date, end_date, by = 1))]
state_confirmed_selected = as.numeric(state_confirmed_selected)

state_confirmed_selected_daily = state_confirmed_selected[2:length(state_confirmed_selected)] -
    state_confirmed_selected[1:(length(state_confirmed_selected) -
        1)]
daily_date_selected = date_selected[2:length(date_selected)]
daily_confirmed_state_df = data.frame(date = daily_date_selected,
    value = state_confirmed_selected_daily)


state_confirmed_selected_daily_avg = data_seven_day_smoothing(state_confirmed_selected_daily)
daily_confirmed_state_avg_df = data.frame(date = daily_date_selected,
    value = state_confirmed_selected_daily_avg)

## let's plot the 7-day average of daily confirmed
## cases
daily_confirmed_state_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))
```



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

state_test_aggregated = state_test %>% group_by(date) %>%
    summarise_each(funs(sum), positiveIncrease, totalTestResultsIncrease)

state_test_aggregated$positiveIncrease_7_day_avg = data_seven_day_smoothing(state_test_aggregated$posit

state_test_aggregated$totalTestResultsIncrease_7_day_avg = data_seven_day_smoothing(state_test_aggregat

state_test_aggregated$positive_rate = state_test_aggregated$positiveIncrease_7_day_avg/state_test_aggreg

mi_test_daily_test_smoothed = data_seven_day_smoothing(state_test_aggregated$totalTestResultsIncrease)

mi_test_daily_positive_smoothed = data_seven_day_smoothing(state_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[state_test_aggregated$date >=
    (start_date) & state_test_aggregated$date <= end_date]
plot(date_selected, mi_test_PositiveRate_smoothed_selected,
    type = "l", xlab = "date", ylab = "7-day averaged daily positive rate")
```

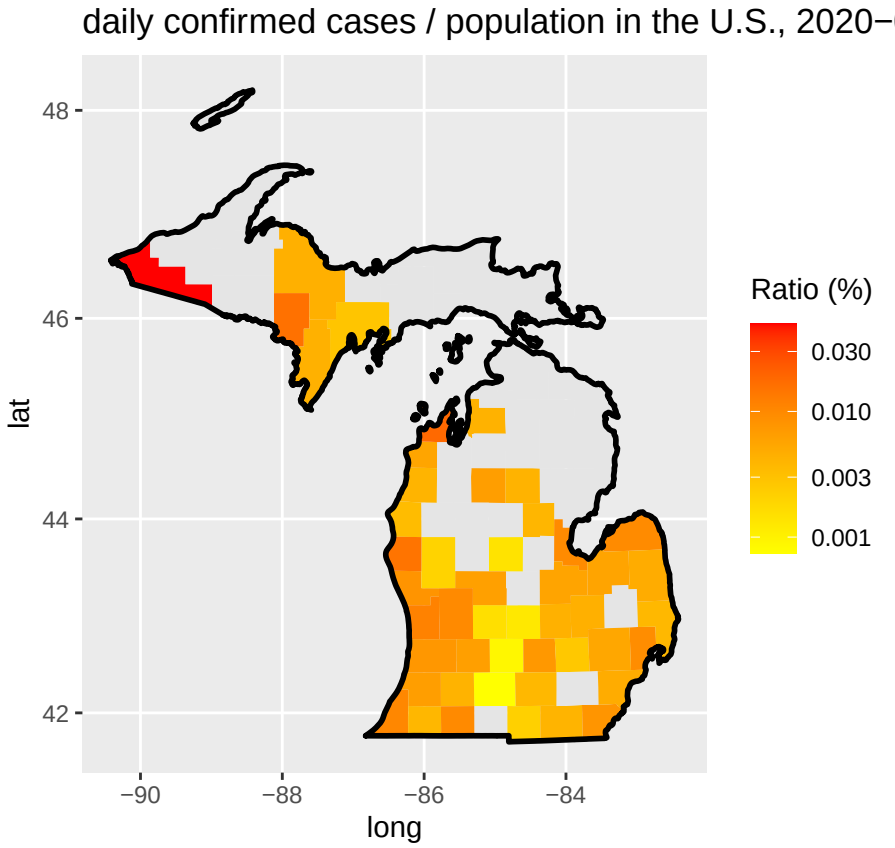


There are more people infected on Oct 20. Although a higher positive rate does not imply a higher number of positive cases because we did not know the number of tested cases, in this case the positive rate agrees with our conclusion that more people are infected on Oct 20.

Question 2.b

Michigan on Jul 20:

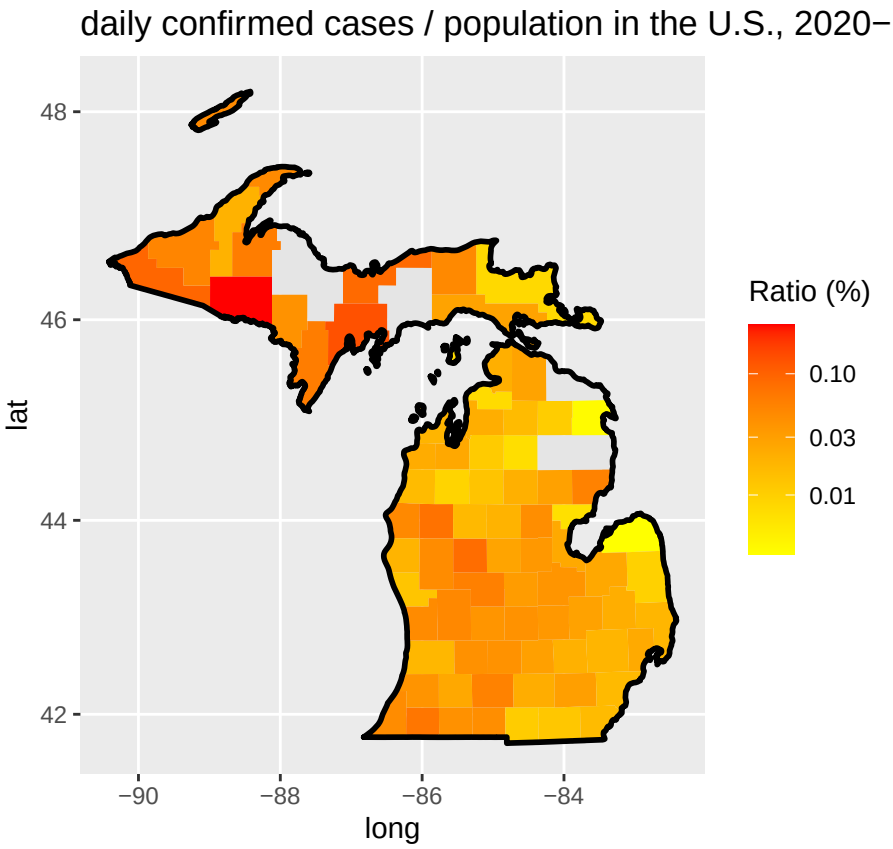
```r
date_for_map = "2020-07-20"
# extract the daily confirmed cases for all US counties on the selected date
mi_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State == 'Michigan
state_population = us_daily_death_clean %>%
  filter(Province_State == "Michigan")%>%
  dplyr::select(Population)

# Ratio = daily confirmed cases / population
mi_daily_confirmed_rate_selected = mi_daily_confirmed_selected
mi_daily_confirmed_rate_selected[,12] = mi_daily_confirmed_selected[,12] / state_population
mi_daily_confirmed_rate_selected[,12][mi_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(mi_daily_confirmed_rate_selected)[12] = "Ratio"
mi_daily_confirmed_rate_selected_joint <- left_join(mi_daily_confirmed_rate_selected, counties, by = "c
mi_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states[urbnmapr::states$state_name=='Michigan'
                            ,], mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Michigan on Oct 20:

```
date_for_map = "2020-10-20"
# extract the daily confirmed cases for all US counties on the selected date
mi_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State == 'Michigan
state_population = us_daily_death_clean %>%
  filter(Province_State == "Michigan")%>%
  dplyr::select(Population)

# Ratio = daily confirmed cases / population
mi_daily_confirmed_rate_selected = mi_daily_confirmed_selected
mi_daily_confirmed_rate_selected[,12] = mi_daily_confirmed_selected[,12] / state_population
mi_daily_confirmed_rate_selected[,12][mi_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(mi_daily_confirmed_rate_selected)[12] = "Ratio"
mi_daily_confirmed_rate_selected_joint <- left_join(mi_daily_confirmed_rate_selected, counties, by = "c
mi_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states[urbnmapr::states$state_name=='Michigan'
                            ,], mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
```

```
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



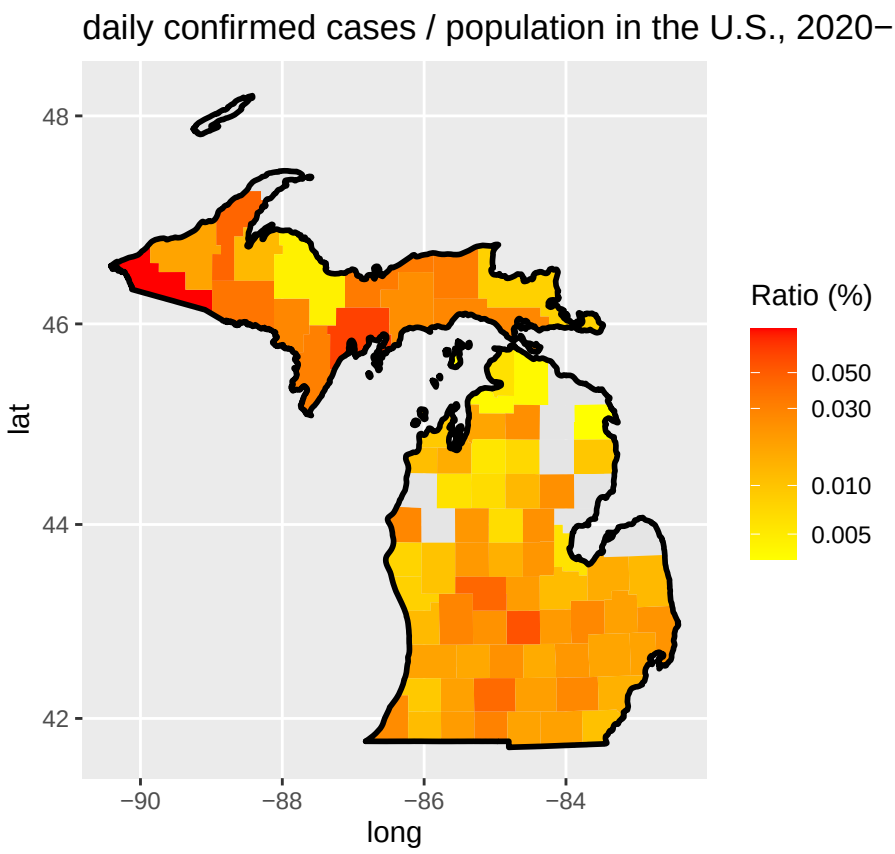
The north part of Michigan showed a rapid increase in positive rate. The south part of the state seems remaining a similar positive rate. Cases also began to show in the middle part (44N to 46N) of the state.

Question 2.c

Oct 22:

```
date_for_map = "2020-10-22"
# extract the daily confirmed cases for all US counties on the selected date
mi_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State == 'Michigan
state_population = us_daily_death_clean %>%
  filter(Province_State == "Michigan")%>%
  dplyr::select(Population)

# Ratio = daily confirmed cases / population
mi_daily_confirmed_rate_selected = mi_daily_confirmed_selected
mi_daily_confirmed_rate_selected[,12] = mi_daily_confirmed_selected[,12] / state_population
mi_daily_confirmed_rate_selected[,12][mi_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(mi_daily_confirmed_rate_selected)[12] = "Ratio"
mi_daily_confirmed_rate_selected_joint <- left_join(mi_daily_confirmed_rate_selected, counties, by = "c
mi_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states[urbnmapr::states$state_name=='Michigan'
```

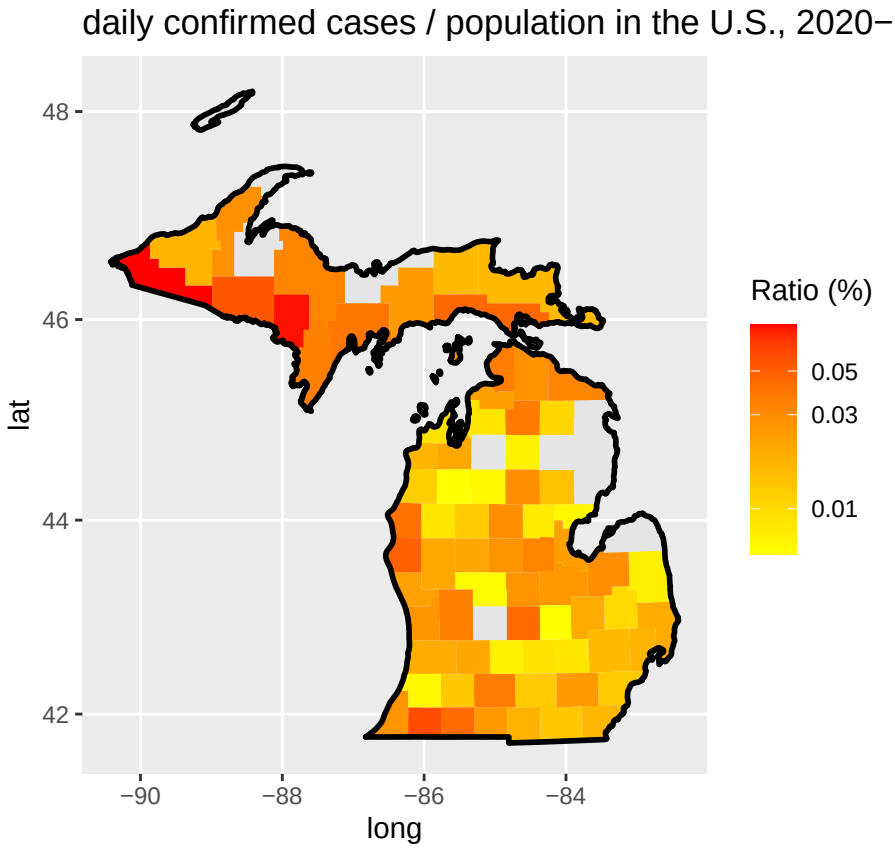
```
                            ,], mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Oct 24:

```
date_for_map = "2020-10-24"
# extract the daily confirmed cases for all US counties on the selected date
mi_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State == 'Michigan
state_population = us_daily_death_clean %>%
  filter(Province_State == "Michigan")%>%
  dplyr::select(Population)

# Ratio = daily confirmed cases / population
mi_daily_confirmed_rate_selected = mi_daily_confirmed_selected
mi_daily_confirmed_rate_selected[,12] = mi_daily_confirmed_selected[,12] / state_population
mi_daily_confirmed_rate_selected[,12][mi_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(mi_daily_confirmed_rate_selected)[12] = "Ratio"
mi_daily_confirmed_rate_selected_joint <- left_join(mi_daily_confirmed_rate_selected, counties, by = "co
mi_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
```

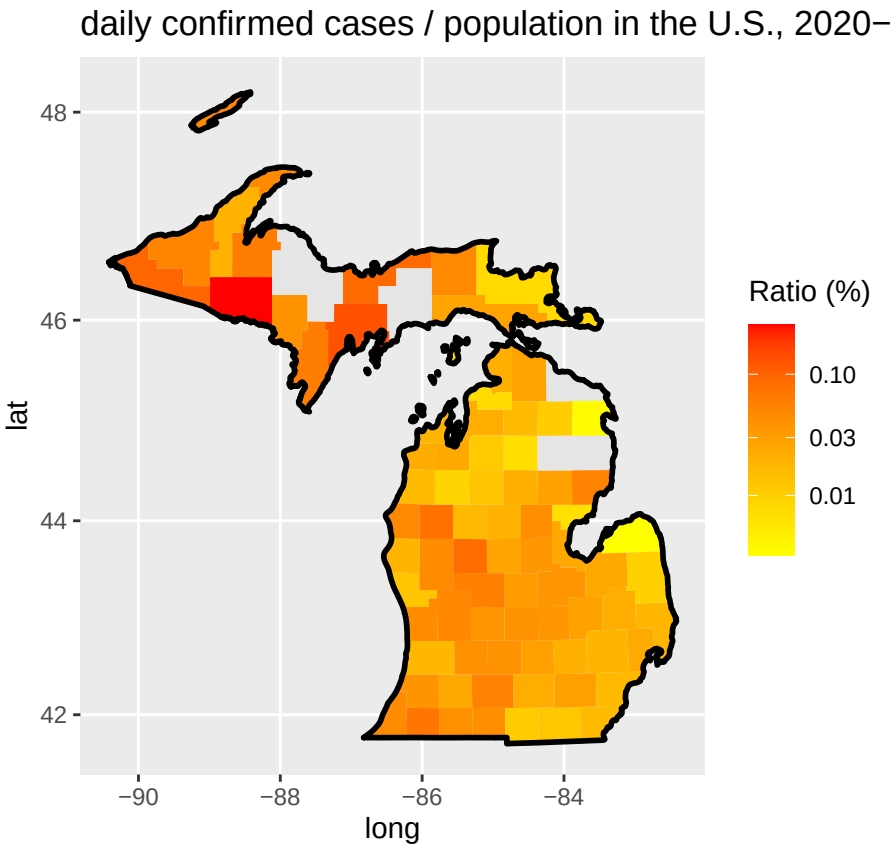
```
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states[urbnmapr::states$state_name=='Michigan'
                            ,], mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Oct 26:

```
date_for_map = "2020-10-20"
# extract the daily confirmed cases for all US counties on the selected date
mi_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State == 'Michigan
state_population = us_daily_death_clean %>%
  filter(Province_State == "Michigan")%>%
  dplyr::select(Population)

# Ratio = daily confirmed cases / population
mi_daily_confirmed_rate_selected = mi_daily_confirmed_selected
mi_daily_confirmed_rate_selected[,12] = mi_daily_confirmed_selected[,12] / state_population
mi_daily_confirmed_rate_selected[,12][mi_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(mi_daily_confirmed_rate_selected)[12] = "Ratio"
```

```r
mi_daily_confirmed_rate_selected_joint <- left_join(mi_daily_confirmed_rate_selected, counties, by = "co
mi_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states[urbnmapr::states$state_name=='Michigan'
                            ,], mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



The northwest part showed a rapid increase of confirmed cases.

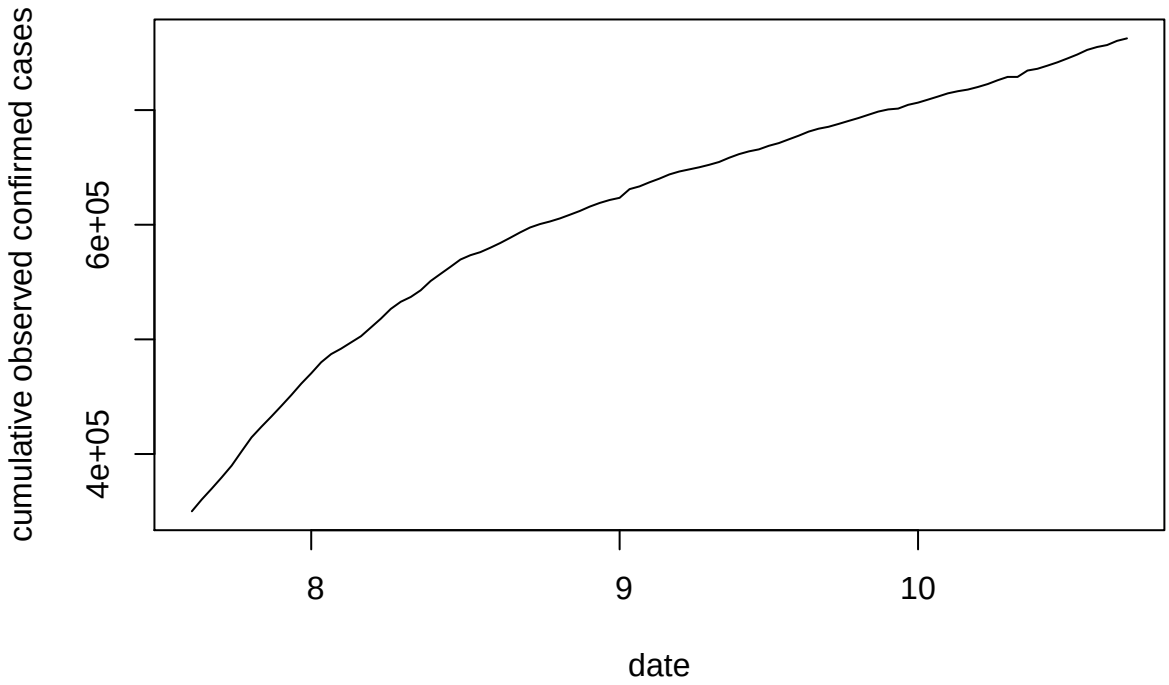
Question 2.d

So firstly, I would recommend people who live in the north part of the state to try their best to stay at home and protect themselves since the situation there is severe. Secondly, for the rest of the state, people should also try their best to keep social distance, in order to prevent the situation from being worse. Keep an eye on important spots like airports, train stations, etc.

Question 3

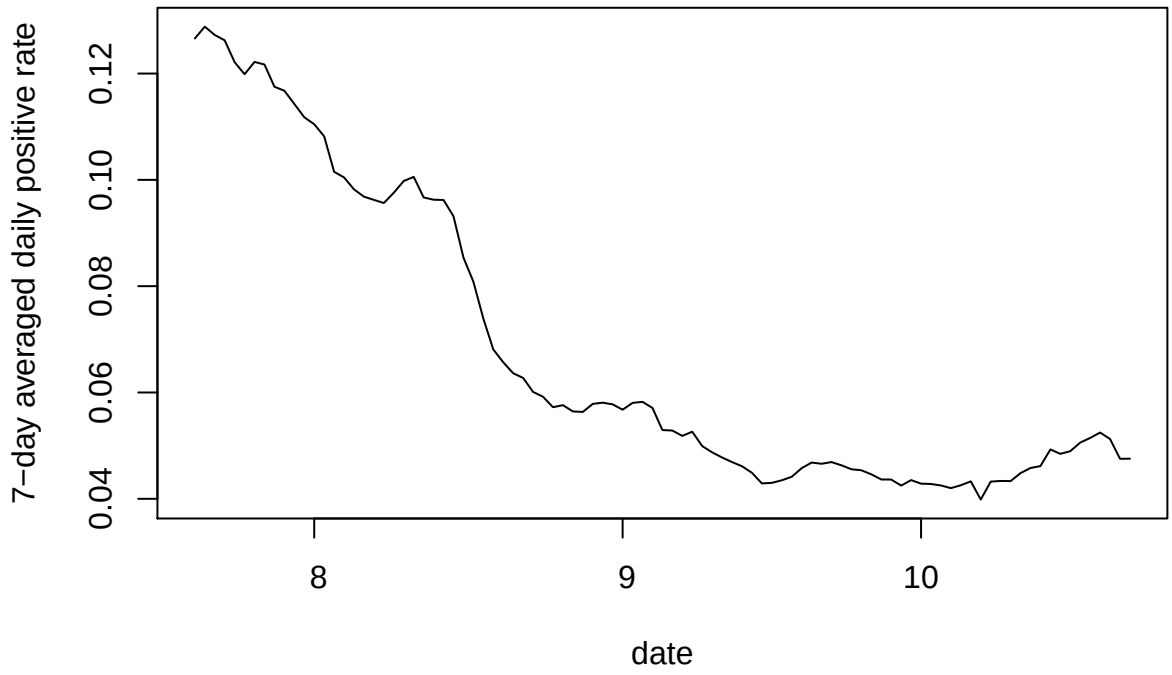
Question 3.a

```
fl_confirmed = us_confirm %>% dplyr::filter(Province_State ==
    "Florida")
start_date = as.Date("2020-7-20")
end_date = as.Date("2020-10-22")
all_dates = as.Date(paste0(str_sub(col_names[13:dim(us_confirm)[2]],
    2, -1), "20"), tryFormats = c("%m.%d.%Y"))
fl_confirmed_sum = apply(fl_confirmed[, 12:dim(fl_confirmed)[2]],
    2, sum)
fl_confirmed_selected = fl_confirmed_sum[which(all_dates %in%
    seq.Date(start_date, end_date, by = 1))]

date_selected = seq.Date(start_date, end_date, by = 1)
plot(date_selected, fl_confirmed_selected, xlab = "date",
    ylab = "cumulative observed confirmed cases", type = "l")
```




```
state_test = covid_19_project %>% filter(state == "FL") %>%
    dplyr::select(date, state, totalTestResultsIncrease,
        positiveIncrease)

state_test_aggregated = state_test %>% group_by(date) %>%
    summarise_each(funs(sum), positiveIncrease, totalTestResultsIncrease)

state_test_aggregated$positiveIncrease_7_day_avg = data_seven_day_smoothing(state_test_aggregated$posit

state_test_aggregated$totalTestResultsIncrease_7_day_avg = data_seven_day_smoothing(state_test_aggregat

state_test_aggregated$positive_rate = state_test_aggregated$positiveIncrease_7_day_avg/state_test_aggreg

mi_test_daily_test_smoothed = data_seven_day_smoothing(state_test_aggregated$totalTestResultsIncrease)
```

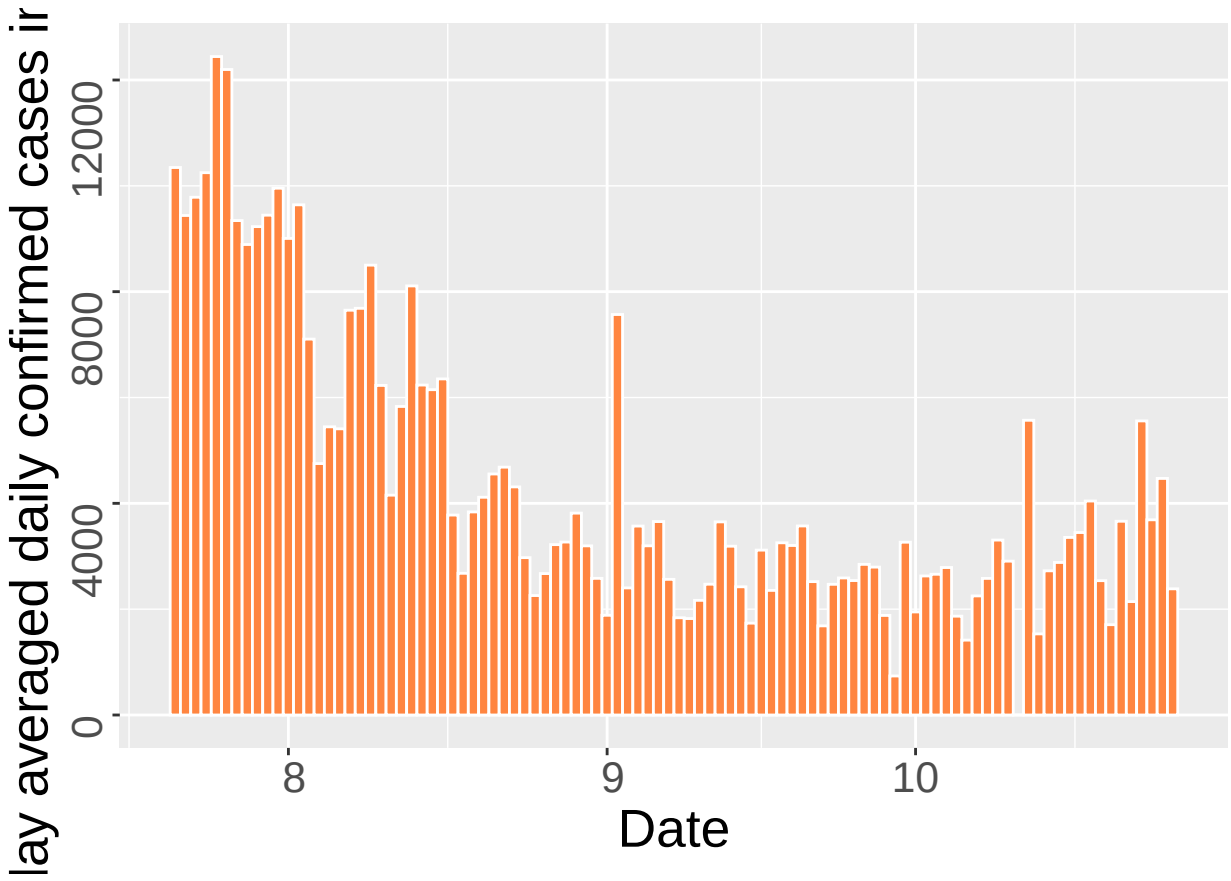
```
mi_test_daily_positive_smoothed = data_seven_day_smoothing(state_test_aggregated$positiveIncrease)

## 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[state_test_aggregated$date >=
    (start_date) & state_test_aggregated$date <= end_date]
plot(date_selected, mi_test_PositiveRate_smoothed_selected,
    type = "l", xlab = "date", ylab = "7-day averaged daily positive rate")
```



Daily positive cases:

```
start_date = as.Date("2020-7-20")
end_date = as.Date("2020-10-26")
date_selected = seq.Date(start_date, end_date, by = 1)
state_name = "Florida"
county_name = "Florida"
state_name_short = "FL"
state_confirmed = us_confirm %>% filter(Province_State ==
    state_name) %>% select(starts_with("x"))

state_confirmed_sum = apply(state_confirmed, 2, sum)
state_confirmed_selected = state_confirmed_sum[which(all_dates %in%
    seq.Date(start_date, end_date, by = 1))]
state_confirmed_selected = as.numeric(state_confirmed_selected)

state_confirmed_selected_daily = state_confirmed_selected[2:length(state_confirmed_selected)] -
    state_confirmed_selected[1:(length(state_confirmed_selected) -
        1)]
daily_date_selected = date_selected[2:length(date_selected)]
daily_confirmed_state_df = data.frame(date = daily_date_selected,
    value = state_confirmed_selected_daily)
```

```r
## let's plot the 7-day average of daily confirmed
## cases
daily_confirmed_state_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 Florida") +
    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))
```



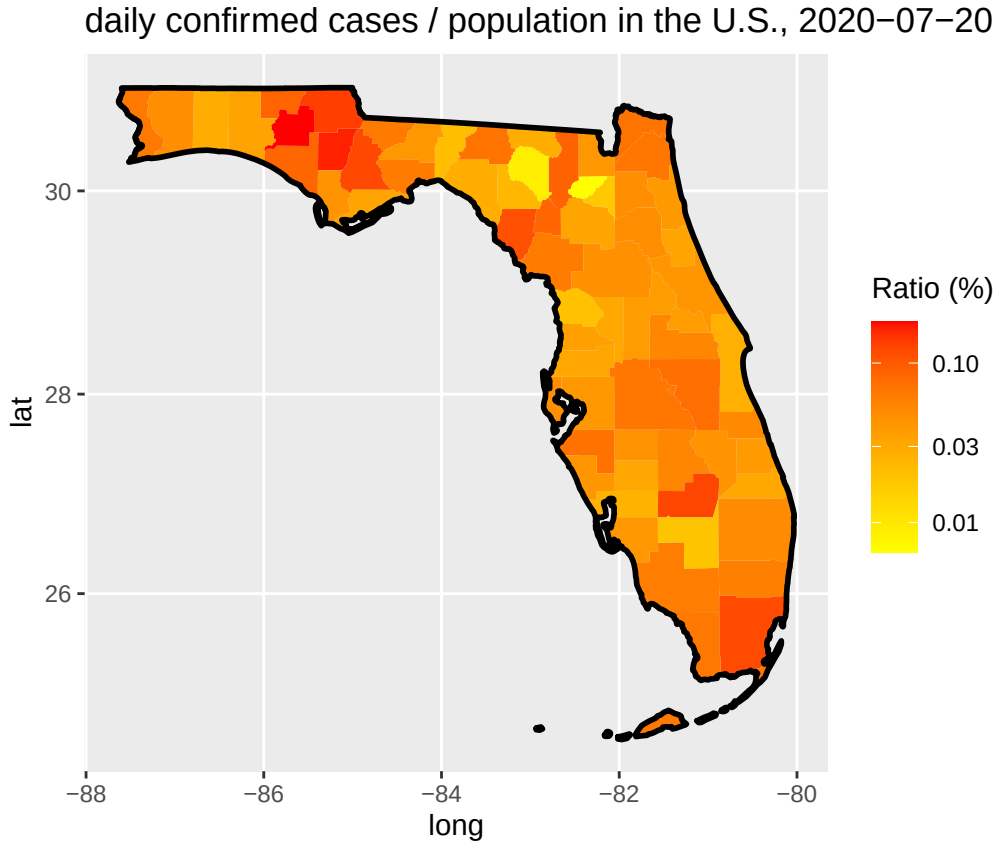
It is clear that on there are more people infected on July 20. The positive rate agrees too.

Question 3.b

Florida on Jul 20:

```r
date_for_map = "2020-07-20"
# extract the daily confirmed cases for all US counties on the selected date
fl_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State == 'Florida'
state_population = us_daily_death_clean %>%
  filter(Province_State == "Florida")%>%
  dplyr::select(Population)

# Ratio = daily confirmed cases / population
fl_daily_confirmed_rate_selected = fl_daily_confirmed_selected
fl_daily_confirmed_rate_selected[,12] = fl_daily_confirmed_selected[,12] / state_population
```

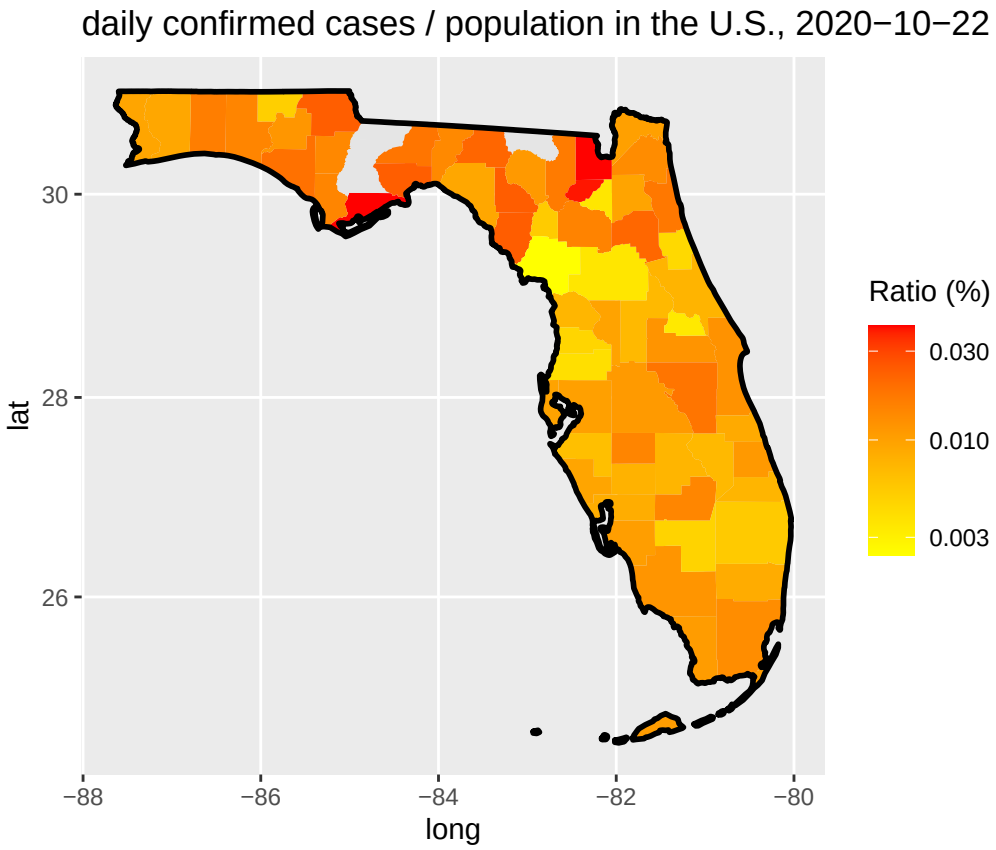
```
fl_daily_confirmed_rate_selected[,12][fl_daily_confirmed_rate_selected[,12] == 0] = NA
colnames(fl_daily_confirmed_rate_selected)[12] = "Ratio"
fl_daily_confirmed_rate_selected_joint <- left_join(fl_daily_confirmed_rate_selected, counties, by = "c
fl_daily_confirmed_rate_selected_joint %>%
  ggplot(aes(long, lat, group = group)) +
  geom_polygon(aes(fill = (Ratio*100)), show.legend = T) +
  geom_polygon(
    data = urbnmapr::states[urbnmapr::states$state_name=='Florida'
                            ,], mapping = aes(x = long, y = lat, group = group),
    fill = NA, color = 'black', size = 1
  ) +
  scale_fill_gradient(low = "yellow", high = "red", na.value = "grey90", trans = "log10"
                      ###, limits = c(0.00038, 1.158)
                      )+
  coord_map() +
  labs(fill = expression("Ratio (%)")) +
  ggtitle(paste0("daily confirmed cases / population in the U.S.", ", ", date_for_map))
```



Florida on Oct 22:

```
date_for_map = "2020-10-22"
# extract the daily confirmed cases for all US
# counties on the selected date
fl_daily_confirmed_selected = us_daily_confirm_clean[us_daily_confirm_clean$Province_State ==
    "Florida", c(1:11, 11 + which(all_dates == date_for_map))]
state_population = us_daily_death_clean %>% filter(Province_State ==
    "Florida") %>% dplyr::select(Population)
```

```
# Ratio = daily confirmed cases / population
fl_daily_confirmed_rate_selected = fl_daily_confirmed_selected
fl_daily_confirmed_rate_selected[, 12] = fl_daily_confirmed_selected[,
    12]/state_population
fl_daily_confirmed_rate_selected[, 12][fl_daily_confirmed_rate_selected[,
    12] == 0] = NA
colnames(fl_daily_confirmed_rate_selected)[12] = "Ratio"
fl_daily_confirmed_rate_selected_joint <- left_join(fl_daily_confirmed_rate_selected,
    counties, by = "county_fips")
fl_daily_confirmed_rate_selected_joint %>% ggplot(aes(long,
    lat, group = group)) + geom_polygon(aes(fill = (Ratio *
    100)), show.legend = T) + geom_polygon(data = urbnmapr::states[urbnmapr::states$state_name ==
    "Florida", ], mapping = aes(x = long, y = lat,
    group = group), fill = NA, color = "black", size = 1) +
    scale_fill_gradient(low = "yellow", high = "red",
        na.value = "grey90", trans = "log10") + coord_map() +
    labs(fill = expression("Ratio (%)")) + ggtitle(paste0("daily confirmed cases / population in the U.S
    ", ", date_for_map))
```



Overall, the maps of positive rate are similar. The northwest part of the state has the highest rate of positive, while the rest of the state shares a similar level. On both Jul 20 and Oct 20, the northest part has larger confirmed cases over the county population.

Question 3.c

On Jul 22, Florida had a more severe situation than Michigan, since nearly all counties across Florida had shown cases, however there were still counties in Michigan that had no case at that time.

On Oct 22, Michigan is worse because there is obvious more counties that has a positive rate greater than 0.1%, and now nearly all counties had people infected.

Question 3.d

As the situation is not so bad for most counties in Florida, we can try make more tests for those counties with positive rate greater than 0.1% when people trying to leave these counties, so it can help control the transmission. As for other counties, the situation is not so bad, therefore basically people can have a normal life if eqiupped with some kind of protection, like masks. They can have less restrictions.

Question II

Question 4

Question 4.a

```
set.seed(1)
# select the state name, its abbreviation and the
# county name
state_name = "California"
state_name_short = "CA"
county_name_selected = "Santa Barbara"

# define the start and end date
start_date_ini = as.Date("2020-03-21")
end_date = as.Date("2020-10-31")
file_pow_param_path = "data_and_functions/"
file_sudden_death_path = "data_and_functions/"

death = "time_series_covid19_deaths_"
us_death = read.csv(paste0(death, "US.csv"))
confirm = "time_series_covid19_confirmed_"
us_confirm = read.csv(paste0(confirm, "US.csv"))

n = as.numeric(end_date - start_date) + 1
prediction_length = 21

# define the length of the training period
fitted_days_beta = 224
gamma = 0.2
theta = 0.1
delta = 0.0066
county_sudden_death = read.csv(paste0(file_sudden_death_path,
    "counties with sudden death increase.csv"))
county_sudden_death$state = as.character(county_sudden_death$state)
us_pow_param = read.csv(paste0(file_pow_param_path,
    "power parameter in US states.csv"))
us_pow_param$state_name = as.character(us_pow_param$state_name)

us_death = clean_us_death(us_death)
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"))
```

```r
death_rate = function(x) {
    x/us_death$Population * 10^6
}
us_death_rate = us_death %>% dplyr::mutate_at(vars(starts_with("X")),
    death_rate)

n_ini = as.numeric(end_date - start_date_ini) + 1
power_parameter = as.numeric(us_pow_param$pow_param[us_pow_param$state_name ==
    state_name])

state_test = covid_19_project %>% dplyr::filter(state ==
    state_name_short) %>% dplyr::select(date, totalTestResultsIncrease,
    positiveIncrease)
state_test$totalTestResultsIncrease[state_test$totalTestResultsIncrease <
    0] = abs(state_test$totalTestResultsIncrease[state_test$totalTestResultsIncrease <
    0])
state_test$positiveIncrease[state_test$positiveIncrease <
    0] = abs(state_test$positiveIncrease[state_test$positiveIncrease <
    0])
state_test$daily_test_avg = rep(0, dim(state_test)[1])
state_test$daily_positive_avg = rep(0, dim(state_test)[1])
state_test$daily_test_avg = data_seven_day_smoothing(state_test$totalTestResultsIncrease)
state_test$daily_positive_avg = data_seven_day_smoothing(state_test$positiveIncrease)
state_test$PositiveRate = state_test$daily_positive_avg/state_test$daily_test_avg
for (i in 2:(dim(state_test)[1])) {
    if (is.na(state_test$PositiveRate[i]) | is.infinite(state_test$PositiveRate[i])) {
        state_test$PositiveRate[i] = state_test$PositiveRate[i -
            1]
    }
}

state_test$PositiveRate[is.na(state_test$PositiveRate)] = state_test$PositiveRate[which(!is.na(state_tes

# change negative rate to positive
state_test$PositiveRate = abs(state_test$PositiveRate)
# change positive rate larger tha 1 to 1
state_test$PositiveRate[state_test$PositiveRate > 1] = 1/state_test$PositiveRate[state_test$PositiveRat
    1]

state_death = us_death %>% dplyr::filter(Province_State ==
    state_name)

state_confirmed = us_confirm %>% dplyr::filter(Province_State ==
    state_name)

# Part 1: Data Cleaning

# Select the county-level death and confirmed cases
county_death_raw = state_death %>% dplyr::filter(Admin2 ==
    county_name_selected)

county_confirm_raw = state_confirmed %>% dplyr::filter(Admin2 ==
    county_name_selected)
```

```
# the real start date is defined as the date that
# the state confirmed cases is no less than 5 for
# the first time
start_date = all_dates[which(as.numeric(county_confirm_raw[12:length(county_confirm_raw)]) >=
    5)[1]]

if (start_date < start_date_ini) {
    start_date = start_date_ini
}

# the length of real training period
n = as.numeric(end_date - start_date) + 1

# get county data from real start_date to end_date

death_with_county_names = get_output_same_time_zone(data_type = "death",
    state_name = state_name, start_date = start_date,
    state_name_short = state_name_short, duration = n,
    training_length = n, criterion_death = 2, smoothness = F)

death_with_county_names_all = get_output_same_time_zone(data_type = "death",
    state_name = state_name, start_date = start_date,
    state_name_short = state_name_short, duration = n +
        prediction_length, training_length = n, criterion_death = 2,
    smoothness = F)


county_names = death_with_county_names[[2]]
each_index = which(county_names == county_name_selected)

state_confirmed_selected = state_confirmed %>% dplyr::filter(Admin2 %in%
    county_names)

state_confirmed_selected = state_confirmed_selected[,
    12:dim(state_confirmed_selected)[2]]

state_confirmed_selected = state_confirmed_selected[,
    which(all_dates %in% seq.Date(start_date, end_date,
        by = 1))]
SIRDC_county_population = us_death %>% filter(Admin2 ==
    county_name_selected, Province_State == state_name) %>%
    dplyr::select(Population)

N = SIRDC_county_population$Population

death_selected = death_with_county_names[[1]][each_index,
    ]
confirm_selected = as.numeric(state_confirmed_selected[each_index,
    ])

for (i in 1:(n - 1)) {
    if (death_selected[i + 1] < death_selected[i]) {
        death_selected[i + 1] = death_selected[i]
```

```
    }
}

for (i in 1:(n - 1)) {
    if (confirm_selected[i + 1] < confirm_selected[i]) {
        confirm_selected[i + 1] = confirm_selected[i]
    }
}

if (paste0(state_name_short, county_name_selected) %in%
    paste0(county_sudden_death$state, county_sudden_death$county)) {
    daily_death = diff(death_selected)
    sudden_increase_index = which(daily_death == max(daily_death))[1]
    daily_death[(sudden_increase_index - 13):(sudden_increase_index -
        1)] = daily_death[(sudden_increase_index -
        13):(sudden_increase_index - 1)] + daily_death[sudden_increase_index]/14
    daily_death[sudden_increase_index] = daily_death[sudden_increase_index]/14
    death_selected = c(death_selected[1], cumsum(daily_death) +
        death_selected[1])
}

daily_confirm_selected = diff(confirm_selected)
daily_confirm_selected_avg = data_seven_day_smoothing(daily_confirm_selected)
unadjusted_confirm_selected_smoothed = c(confirm_selected[1],
    cumsum(daily_confirm_selected_avg) + confirm_selected[1])

daily_positive_rate_selected = rep(0, n)

# extract the positive rate on selected date
daily_positive_rate_selected = rev(state_test$PositiveRate[state_test$date >=
    (start_date) & state_test$date <= end_date])

# calculate the weights
c_t_seq = daily_positive_rate_selected^(power_parameter)

daily_confirm_selected = diff(confirm_selected)
daily_adjusted_confirm_smoothed = data_seven_day_smoothing(daily_confirm_selected *
    c_t_seq[2:n])
confirm_selected_smoothed = c(c_t_seq[1] * confirm_selected[1],
    cumsum(daily_adjusted_confirm_smoothed) + c_t_seq[1] *
        confirm_selected[1])

# eliminate decreasing trend in the smoothed county
# confirmed cases
for (i in 2:n) {
    if (confirm_selected_smoothed[i] < confirm_selected_smoothed[i -
        1]) {
        confirm_selected_smoothed[i] = confirm_selected_smoothed[i -
            1]
    }
}
```

Step 2:

```
# try constrained optimization
ui = matrix(c(1, 0, -1, 0, 1, -1), 3, 2)
ci = matrix(c(0, 0, -((confirm_selected_smoothed[1] *
    N)/confirm_selected_smoothed[n] - death_selected[1] -
    0.1)))

I_0_ini = 1000
R_0_ini = 1000

if ((I_0_ini + R_0_ini) > abs(ci[3])) {
    I_0_ini = abs(ci[3])/4
    R_0_ini = abs(ci[3])/4
}

# do optimization
m_approx = constrOptim(c(I_0_ini, R_0_ini), loss_approx_beta,
    NULL, ui = ui, ci = ci, death_cases = death_selected,
    confirmed_cases = confirm_selected_smoothed, unadjusted_confirm_selected_smoothed = unadjusted_confi
    N_population = N, trianing_length = n, fixed_global_params = c(gamma,
        theta, delta), penalty = F, fitted_days = fitted_days_beta,
    weight_loss = T)
I_0 = m_approx$par[1]
R_0 = m_approx$par[2]
```

Step 3:

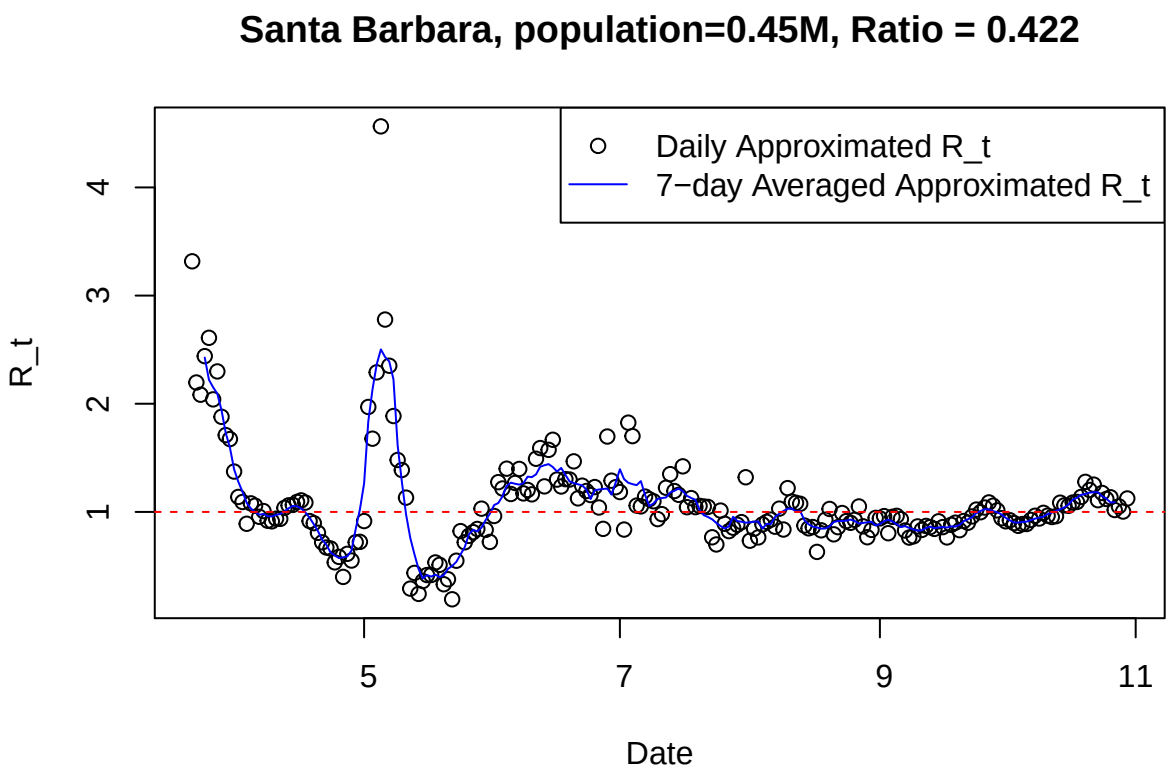
```
ratio = confirm_selected_smoothed[1]/(I_0 + R_0 + death_selected[1])

ratio_real = confirm_selected[2]/(I_0 + R_0 + death_selected[1])

# use ratio to adjust smoothed confirmed cases and
# get susceptible cases S_t
estimated_confirm = confirm_selected_smoothed/ratio

# make sure the adjusted confirmed cases is at
# least larger than the observed confirm cases
for (i in 1:length(estimated_confirm)) {
    if (estimated_confirm[i] < unadjusted_confirm_selected_smoothed[i]) {
        estimated_confirm[i] = unadjusted_confirm_selected_smoothed[i]
    }
}

S_t_seq = N - estimated_confirm

# initial values of each compartment
init_for_beta = c(S_t_seq[1], I_0, R_0, death_selected[1],
    0)

param_record_approx_for_beta = matrix(0, 5, n)  # 5 rows: S_t, I_t, R_t, D_t, C_t
param_record_approx_for_beta[, 1] = init_for_beta
param_record_approx_for_beta[1, ] = S_t_seq

# record the value of transmission rate
approx_beta_seq = rep(0, n - 1)
```

```
# iterative approach for calculating the seq of
# compartments in SIRDC
for (i in 1:(n - 1)) {
    S_t_1 = param_record_approx_for_beta[1, i]
    S_t_2 = param_record_approx_for_beta[1, i + 1]
    I_t_1 = param_record_approx_for_beta[2, i]
    R_t_1 = param_record_approx_for_beta[3, i]
    D_t_1 = param_record_approx_for_beta[4, i]
    C_t_1 = param_record_approx_for_beta[5, i]

    if (I_t_1 < 1) {
        I_t_1 = 1
    }

    beta_t_1_2 = uniroot(find_root_beta, c(0, 10^6),
        tol = 1e-04, param = c(S_t_1, S_t_2, I_t_1),
        N = N, gamma = gamma)

    I_t_2 = I_t_1 * exp(beta_t_1_2$root * (S_t_1 +
        S_t_2)/(2 * N) - gamma)
    R_t_2 = (2 - theta)/(2 + theta) * R_t_1 + gamma/(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[2:5, i + 1] = c(I_t_2,
        R_t_2, D_t_2, C_t_2)
    approx_beta_seq[i] = beta_t_1_2$root
}

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

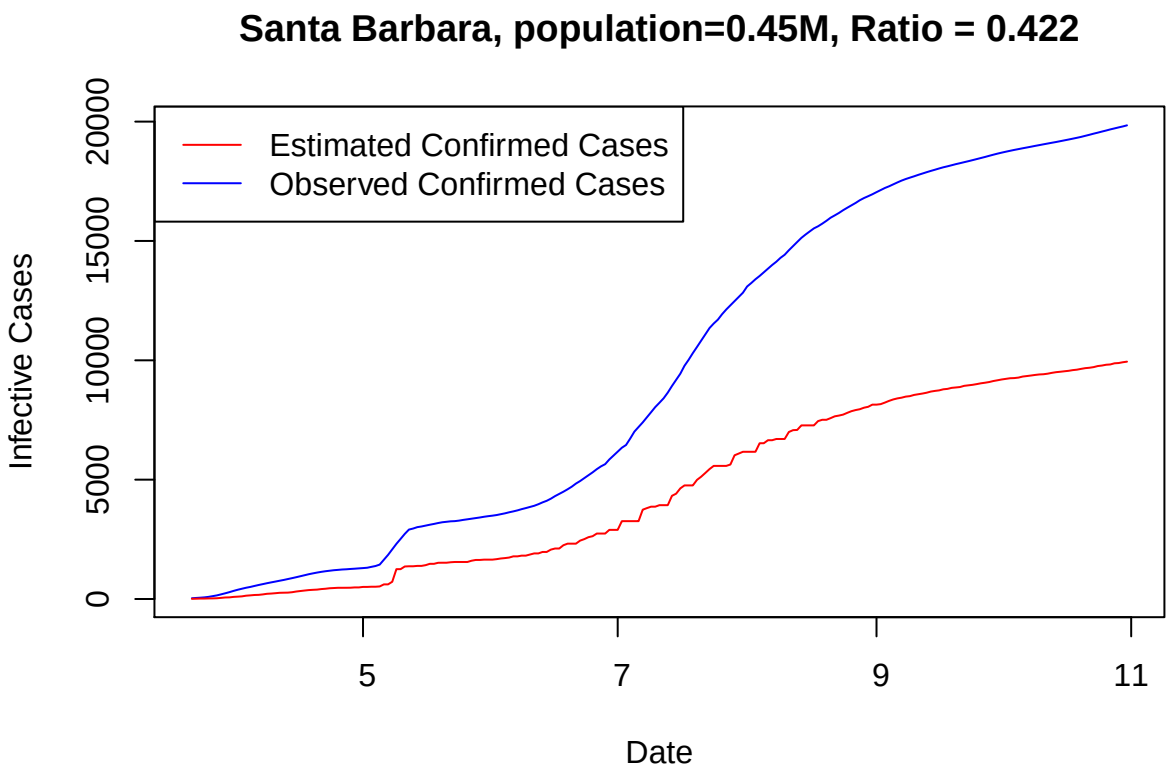
Plots:

```
date_seq_beta = seq.Date(start_date, end_date - 1,
    by = 1)
date_seq_beta_smoothed = seq.Date(start_date + 3, end_date -
    1 - 3, by = 1)

y_limit_R_t = c(min(approx_beta_seq/0.2), max(approx_beta_seq/0.2))

## plot the effective reproduction number 7-day
## average
plot(approx_beta_seq/0.2 ~ date_seq_beta, type = "p",
    ylim = y_limit_R_t, ylab = "R_t", xlab = "Date",
    main = paste0(county_names[each_index], ", population=",
        round(N/10^6, 2), "M", ", Ratio = ", round(ratio_real,
            3)))
lines(approx_beta_seq_smoothed/0.2 ~ date_seq_beta_smoothed,
    col = "blue")
```

```
abline(h = 1, lty = 2, col = "red")
abline(h = 0, lty = 1, col = "black")
legend("topright", legend = c("Daily Approximated R_t",
    "7-day Averaged Approximated R_t"), pch = c(1,
    NA), lty = c(NA, 1), col = c("black", "blue"))
```




```
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"))
```




```
print(paste("observed:", confirm_selected[n]))
```

```
## [1] "observed: 9944"
```

```
print(paste("estimated:", N - param_record_approx_for_beta[1,
    ][n]))
```

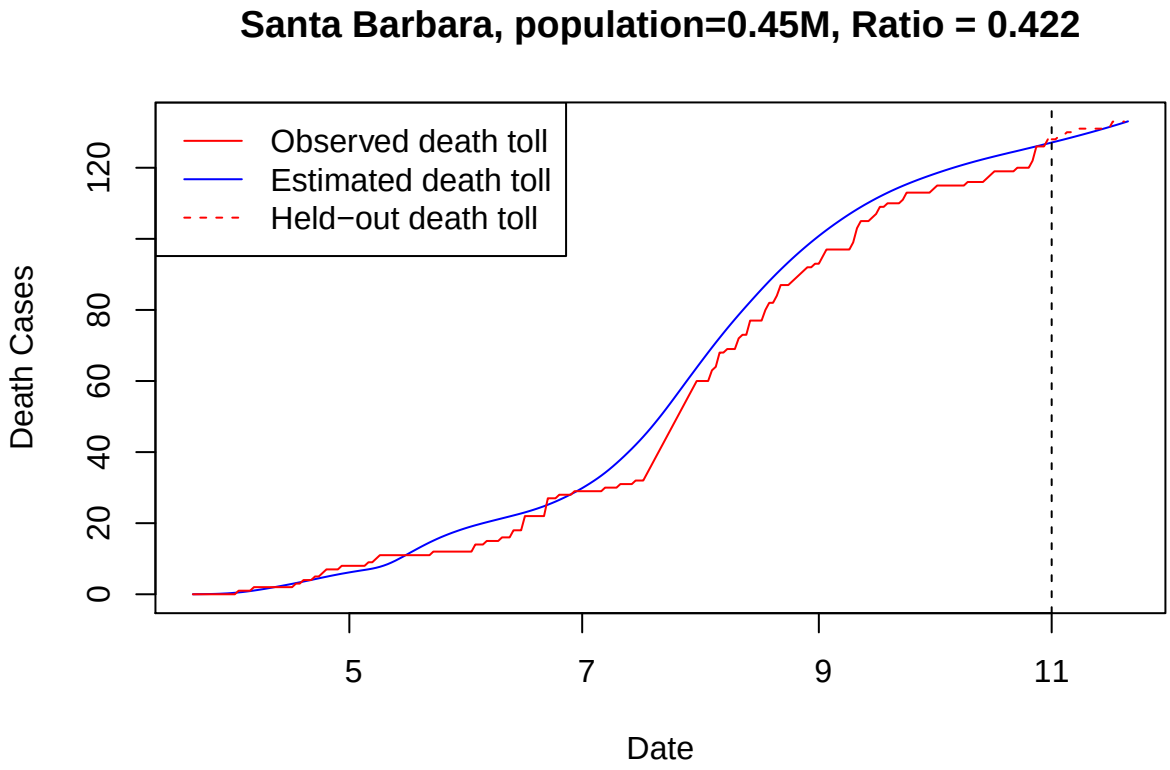
```
## [1] "estimated: 19843.3713946382"
```

Based on the model, it is believed that about 19844 people have contracted the virus as of Oct 31. However, the observed number is 9944. So there is 9900 people were not tested positively.

Question 4.b

```
param_record_approx_all = matrix(0, 5, n + prediction_length)  # 5 rows: S_t, I_t, R_t, D_t, C_t
param_record_approx_all[, 1] = init_for_beta

approx_beta_seq_all = c(approx_beta_seq, rep(approx_beta_seq[n -
    1], prediction_length))

for (i in 1:(n + prediction_length - 1)) {
    S_t_1 = param_record_approx_all[1, i]
    I_t_1 = param_record_approx_all[2, i]
    R_t_1 = param_record_approx_all[3, i]
    D_t_1 = param_record_approx_all[4, i]
    C_t_1 = param_record_approx_all[5, i]

    beta_t_1_2 = approx_beta_seq_all[i]

    if (I_t_1 < 1) {
        I_t_1 = 1
```

```
    }

    S_t_2 = uniroot(find_root_S_t_2, c(0, N), tol = 1e-04,
        param = c(S_t_1, beta_t_1_2, I_t_1), N = N,
        gamma = gamma)
    I_t_2 = I_t_1 * exp(beta_t_1_2 * (S_t_1 + S_t_2$root)/(2 *
        N) - gamma)
    R_t_2 = (2 - theta)/(2 + theta) * R_t_1 + gamma/(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_all[, i + 1] = c(S_t_2$root,
        I_t_2, R_t_2, D_t_2, C_t_2)

}

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"))
```

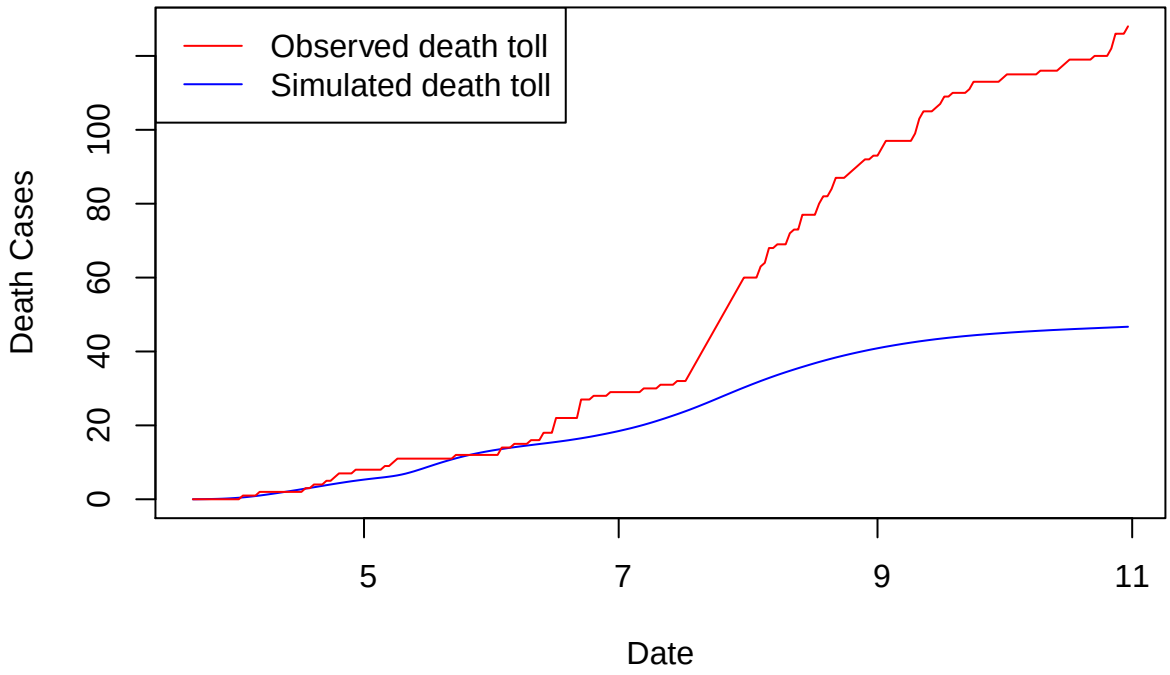


As we can see from the graph, the estimated death toll and the observed death toll are very similar. There is some fluctuation in the observed trend, but the overall tendency is identical to the estimated data.

Question 4.c

```
date_seq_all = seq.Date(start_date, end_date + prediction_length,
    by = 1)
ylimit_death = c(min(param_record_approx_for_beta[4,
    ], death_selected), max(param_record_approx_for_beta[4,
    ], death_selected))
ylimit_death_all = c(min(param_record_approx_all[4,
    ], death_selected), max(param_record_approx_all[4,
    ], death_selected))

gamma_new = 1/(5 * 0.95)

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

# 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 = 1e-04,
        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"))
```



The simulated deaths toll is:

```
param_record_approx_for_beta_new[4, 225]
```

```
## [1] 46.69298
```

Calculate the number of saved death:

```
param_record_approx_for_beta[4, 225] - param_record_approx_for_beta_new[4,
    225]
```

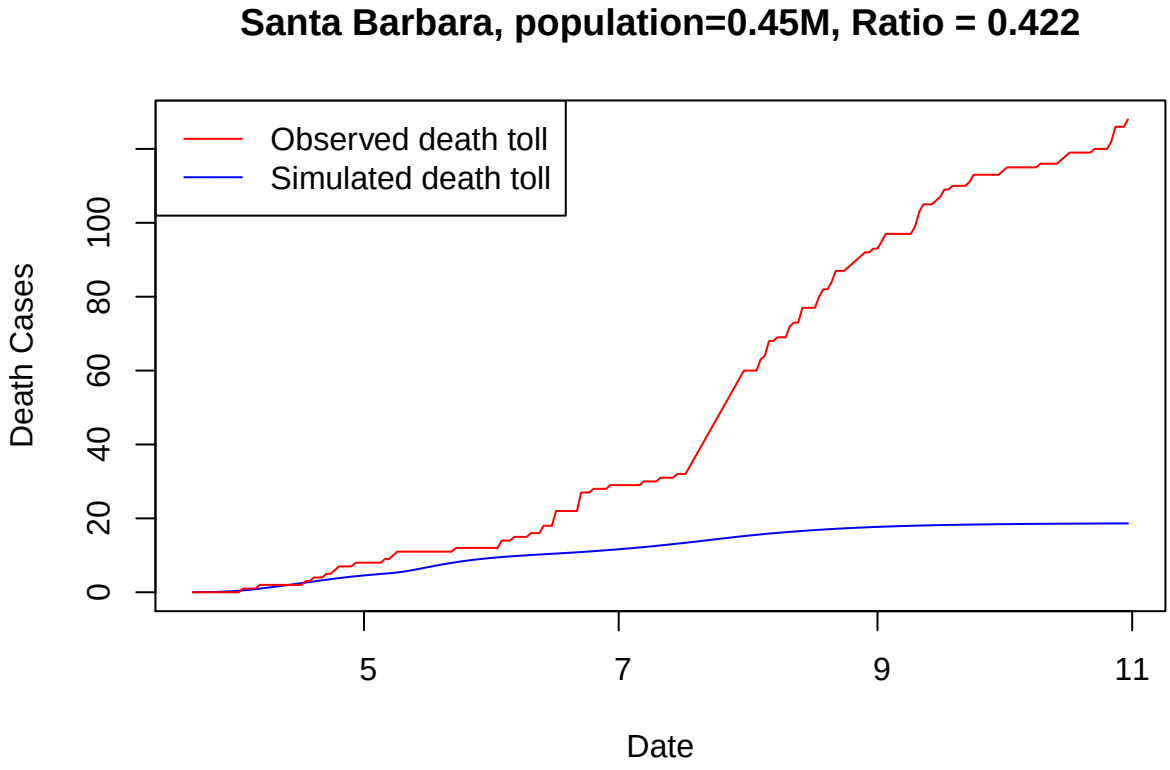
```
## [1] 80.13213
```

About 80 lives can be saved.

Question 4.d

```
gamma_new = 1/(5 * 0.9)

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

# 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 = 1e-04,
        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"))
```



The simulated deaths toll is:

```
param_record_approx_for_beta_new[4, 225]
```

```
## [1] 18.64699
```

Calculate the number of saved death:

```
param_record_approx_for_beta[4, 225] - param_record_approx_for_beta_new[4,
    225]
```

```
## [1] 108.1781
```

About 108 lives can be saved.

Question 4.e

1. Find newly infected people as soon as possible.
2. Quarantine
3. Reduce social activities.

Question 5

Question 5.a

Since the vaccine is designed for healthy people to forestall the disease, it can reduce the amount of S.