Homework_09

Neal Kar

Announcement

Please do not add code folding (code_folding: hide) to your YAML Header or echo = FALSE to your RMD code chunk options. In order to accurately grade your HTML files we need to be able to see all of your code. Thank you!

Question 1

a)

Read in the data “amazon_reviews.csv”. The data contains information on consumer reviews for amazon products like the Kindle and Fire TV Stick from 2016 to 2018.

The original dataset is available at: https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products/data (https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products/data).

```
amazon_df_wide <- read_csv("data/amazon_reviews.csv")
```

b)

We are interested in predicting customer ratings given the information in our data. Use str_remove() to convert `rating` into a numeric variable.

```
amazon_df_wide_cleaned <- amazon_df_wide %>%
  mutate(rating = str_remove(rating, "rating = ")) %>%
  mutate(rating = as.numeric(rating))
```

c)

We want to extract the year the review was submitted from the `date` *variable. Use functions from the* `stringr` *package to obtain the year from the "year-month-day hour:minute:second" date variable.*

```
amazon_df_wide_cleaned <- amazon_df_wide_cleaned %>%
  mutate(year = str_extract(date, "2016|2017|2018")) %>%
  mutate(year = as.numeric(year))
```

d)

We are interested in only analyzing reviews on the Kindle Fire. Use str_detect to filter the data to only include these observations. Hint: makes sure to account for case sensitivity.

```
amazon_df_wide_filtered <- amazon_df_wide_cleaned %>%
  mutate(product_name = str_to_title(product_name)) %>%
  filter(str_detect(product_name, "Kindle Fire"))
```

e)

Create an indicator variable called `title_type` that takes value “good” if the review title contains the word “good” or “great” (regardless of case) and “bad” if it does not. Hint: to detect more than one pattern in a string, you can use pattern = “first pattern|second pattern”. For example str_detect(“apple” , “a|b”) and str_detect(“bird”, “a|b”) both return TRUE.

```
amazon_df_wide_filtered <- amazon_df_wide_filtered %>%
  mutate(title = str_to_lower(title)) %>%
  mutate(title_type = if_else(str_detect(title, "good|great"),
                              "good", "bad")) %>%
  mutate(title = str_to_title(title))
```

f)

Identify a few key words that may indicate a positive review by skimming the review variable. Create another indicator variable called `review_type` *that takes value "good" and "bad" based on the presence of these words in the review variable.*

```
amazon_df_wide_filtered <- amazon_df_wide_filtered %>%
  mutate(review = str_to_lower(review)) %>%
  mutate(review_type = if_else(str_detect(review,
                "good|great|perfect|wonderful|like|love|highly r
ecommend|would recommend|happy|satisfied|enjoy|exceed"),
                "good", "bad")) %>%
  mutate(review = str_to_sentence(review))
```

g)

Perform a multiple regression analysis to quantify the relationship between customer rating and year, type of title, and type of review. Since you created the `review_type` *variable, your results will be unique. Please report the model R-squared and also print a table of intercept and slope parameter estimates using* `kable()`*.*

```
#Construct linear model
rating_lm <- lm(rating ~ year + title_type + review_type,
                data = amazon_df_wide_filtered)

#Put fit stats and parameter stats into tibbles
rating_lm_fit <- glance(rating_lm)
rating_lm_est <- tidy(rating_lm)

#Report the tibble of paramater estimates
kable(rating_lm_est)
```

term	estimate	std.error	statistic	p.value
(Intercept)	60.8521248	158.8009010	0.3831976	0.7017491
year	-0.0280899	0.0787540	-0.3566791	0.7214945
title_typegood	0.3161962	0.0783936	4.0334464	0.0000643

term	estimate	std.error	statistic	p.value
review_typegood	0.2002258	0.0876436	2.2845460	0.0227911

The model R-squared is 0.0475561.

h)

Which variables in your model are significant. If they are significant, interpret the slope estimates for your `review_type` and `title_type` variables. What do you think you could do to improve your `title_type` or `review_type` variables to make them better predictors of rating?

The title_type and review_type variables are significant in the model.

Compared to Kindle Fires with bad review titles, Kindle Fires with good review titles were rated 0.32 units higher (on average), controlling for year of review and review type.

Compared to Kindle Fires with bad reviews, Kindle Fires with good reviews were rated 0.2 units higher (on average), controlling for year of review and title type.

To make title type a better predictor, I could include additional terms to better differentiate the good review titles from the bad review titles. To make both title type and review type better predictors, I could code them based on “polarity” (e.g. a Likert Scale evaluating the titles and reviews from highly negative to highly positive). In that case, certain words would carry certain weights on polarity, and certain punctuation would carry some weight.

Question 2

We will re-visit a similar data set from lab in this question in order to practice using string manipulation functions and setting reference levels for categorical variables.

This dataset includes information on commute time (in minutes), rent (in dollars), and borough. The goal of this problem will to make a model that predicts commute time based on rent and borough. While the variable names are the same, the data values have changed from class!

a)

Load `commute_data.csv` *into your R environment and take a look at the data. How many unique borough entries are there in the raw data? Use* `stringr` *functions to create a clean and corrected borough variable (* `borough_cl` *).*

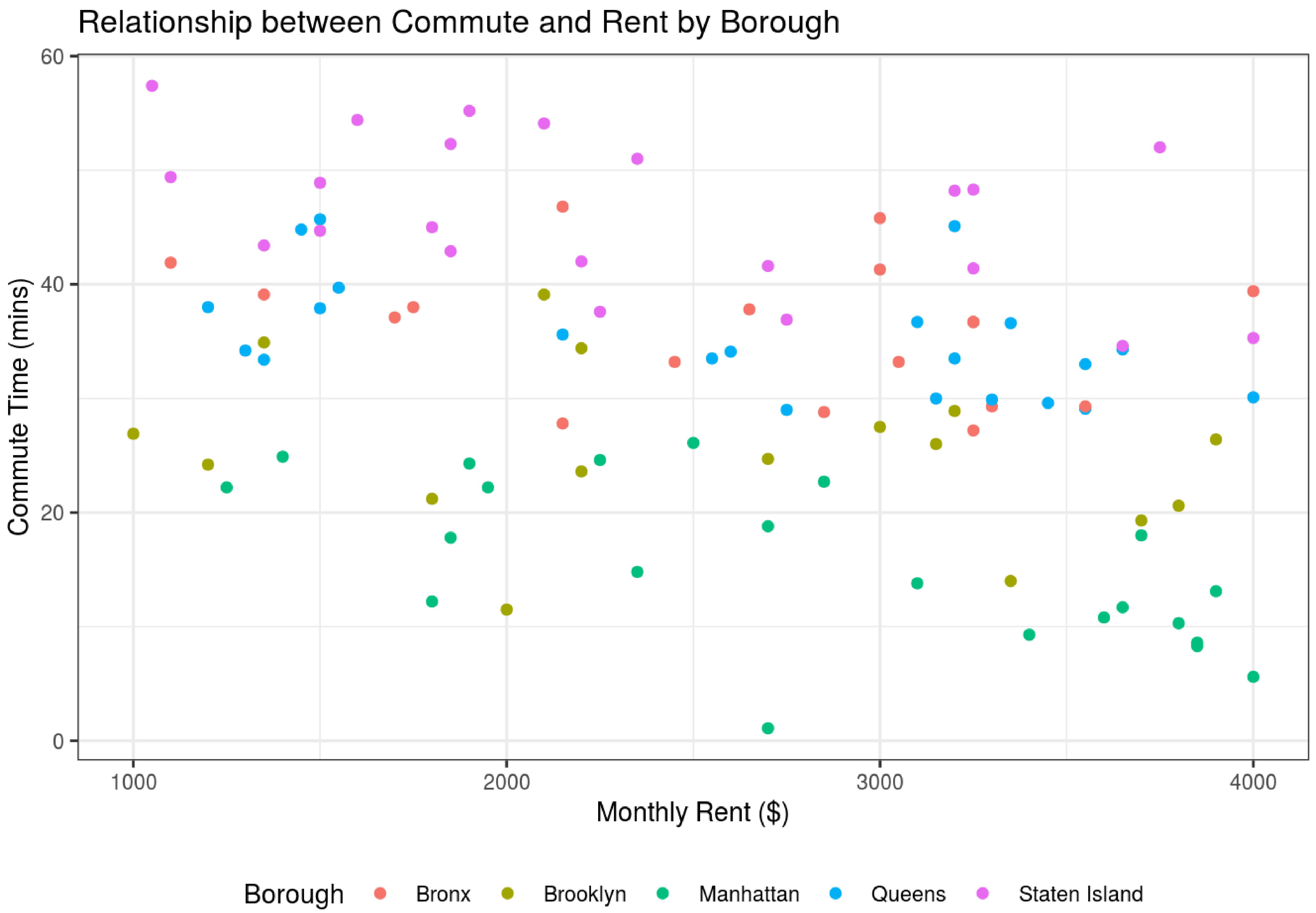
```
#Read in the csv file
commute_df_wide <- read_csv("data/commute_data.csv")

#Create a tibble to view unique borough entries and their freque
ncies
borough_cats <- table(commute_df_wide$borough)
borough_cats <- as.data.frame(borough_cats)
view(borough_cats)

#Create a clean borough variable
commute_df_wide_cleaned <- commute_df_wide %>%
  mutate(borough_cl = str_replace_all(borough, "0", "o")) %>%
  mutate(borough_cl = str_replace_all(borough_cl, "4", "a")) %>%
  mutate(borough_cl = str_replace_all(borough_cl, "xx", "x")) %
>%
  mutate(borough_cl = str_replace_all(borough_cl,"ly n","lyn"))%
>%
  mutate(borough_cl = str_replace_all(borough_cl, "Queens",
                                      "Queen")) %>%
  mutate(borough_cl = str_replace_all(borough_cl, "Queen",
                                      "Queens")) %>%
  mutate(borough_cl = str_replace_all(borough_cl, "Staten  islan
d", "Staten Island")) %>%
  mutate(borough_cl = str_to_title(borough_cl))

#Double check to make sure there are only 5 unique borough entri
es
borough_cats_cleaned <- table(commute_df_wide_cleaned$borough_c
l)
borough_cats_cleaned <- as.data.frame(borough_cats_cleaned)
view(borough_cats_cleaned)
#Confirmed
```

There are 14 unique borough entries in the raw data.

b)

Create a scatterplot of commute by rent, with different colors for different boroughs (using your cleaned borough variable). Make sure your plot has an informative title and labels for the x and y axes. This is a good way to check whether you have cleaned your borough variable. FYI: There are only five boroughs in NYC!

```
ggplot(data = commute_df_wide_cleaned) +
  geom_point(aes(x=rent, y=commute, color=borough_cl)) +
  theme_bw() +
  theme(legend.position = "bottom", text = element_text(size=1
0))+
  labs(title="Relationship between Commute and Rent by Borough",
       x="Monthly Rent ($)", y="Commute Time (mins)",
       color="Borough")
```


Relationship between Commute and Rent by Borough
Commute Time (mins)
0
20
40
60
1000
2000
3000
4000
Monthly Rent ($)
Borough
Bronx
Brooklyn
Manhattan
Queens
Staten Island

c)

Fit a linear regression model for commute with rent and borough as predictors. Don't explicitly specify the levels of your cleaned borough factor, let R do it for you. Report the overall model R-squared.

```r
#Run the linear model
commute_lm <- lm(commute ~ rent + borough_cl,
                 data=commute_df_wide_cleaned)

#Store fit stats in tibble
commute_lm_fit <- glance(commute_lm)
```

The overall model R-squared value is 0.8000697.

d)

Check the partial f-test for borough in your model and report whether the categorical borough variable is associated with commute time in minutes. Take a look at the slides or lab code if you are unsure how to obtain this output.

```
commute_lm_anova <- Anova(commute_lm, type = 3)

view(commute_lm_anova)
```

Since the p-value of the partial F-test for borough is less than 0.05, borough is significantly associated with commute time.

e)

What borough did R automatically make the reference level? Report the slope parameter estimates for the model using `tidy()` and `kable()` and indicate which dummy variables are significant. For the significant dummy variables, interpret what *the slope estimates mean. Remember that the outcome is commute time and consider the reference level in your answer.*

```
commute_lm_param <- tidy(commute_lm)
kable(commute_lm_param)
```

term	estimate	std.error	statistic	p.value
(Intercept)	44.8187791	2.2236565	20.1554418	0.0000000
rent	-0.0032916	0.0006628	-4.9660244	0.0000030
borough_clBrooklyn	-11.2560821	1.9823163	-5.6782471	0.0000002
borough_clManhattan	-19.9810216	1.8360948	-10.8823473	0.0000000
borough_clQueens	-1.0579931	1.8325362	-0.5773382	0.5650913
borough_clStaten Island	9.0133361	1.8460558	4.8824831	0.0000043

R automatically made Bronx the reference level for the borough variable. Brooklyn, Manhattan, and Staten Island were the significant dummy variables.

Compared to those living in the Bronx, those living in Brooklyn commuted 11.26 minutes less (on average), controlling for rent.

Compared to those living in the Bronx, those living in Manhattan commuted 19.98 minutes less (on average), controlling for rent.

Compared to those living in the Bronx, those living in Staten Island commuted 9.01 minutes more (on average), controlling for rent.

f)

Nobody in NYC ever gives Staten Island enough respect! Use `factor()` *to create a new borough variable whose reference level will be Staten Island. Now re-run the same regression model from part (c) using your new borough variable. Report the slope parameter estimates for the model using* `tidy()` *and* `kable()` *and indicate* *which dummy variables are significant. For the significant dummy variables, interpret what the slope estimates mean. Remember that the outcome is commute time and consider the reference level in your answer.*

```r
#Change factor of borough variable
commute_df_wide_cleaned <- commute_df_wide_cleaned %>%
  mutate(borough_cl = factor(borough_cl,
                             c("Staten Island", "Manhattan",
                               "Brooklyn", "Bronx", "Queens")))

#Double check
commute_df_wide_cleaned %>%
  pull(borough_cl) %>%
  levels()
```

```
## [1] "Staten Island" "Manhattan"     "Brooklyn"      "Bronx"
## [5] "Queens"
```

```
#Run the model again
commute_lm_v2 <- lm(commute ~ rent + borough_cl,
                    data = commute_df_wide_cleaned)

#Store parameter estimates in a tibble and report them
commute_lm_param_v2 <- tidy(commute_lm_v2)
kable(commute_lm_param_v2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	53.8321152	1.9664925	27.374687	0.0e+00
rent	-0.0032916	0.0006628	-4.966024	3.0e-06
borough_clManhattan	-28.9943577	1.7718586	-16.363810	0.0e+00
borough_clBrooklyn	-20.2694182	1.9000621	-10.667766	0.0e+00
borough_clBronx	-9.0133361	1.8460558	-4.882483	4.3e-06

term	estimate	std.error	statistic	p.value
borough_clQueens	-10.0713292	1.7490797	-5.758074	1.0e-07

Manhattan, Brooklyn, Bronx and Queens are the significant dummy variables.

Compared to those living in Staten Island, those living in Manhattan commuted 28.99 minutes less (on average), controlling for rent.

Compared to those living in Staten Island, those living in Brooklyn commuted 20.27 minutes less (on average), controlling for rent.

Compared to those living in Staten Island, those living in the Bronx commuted 9.01 minutes less (on average), controlling for rent.

Compared to those living in Staten Island, those living in Queens commuted 10.07 minutes less (on average), controlling for rent.

Question 3

The New York Times has begun to make daily COVID-19 confirmed case and confirmed death data publicly available on their github page: https://github.com/nytimes/covid-19-data (https://github.com/nytimes/covid-19-data).

a)

Load the US state level data we have provided in `data/us-states.csv` *. This data was downloaded on 3/31/20 and includes information on both confirmed cases and confirmed deaths up to 03/30/20. What is the earliest date recorded in this dataset?*

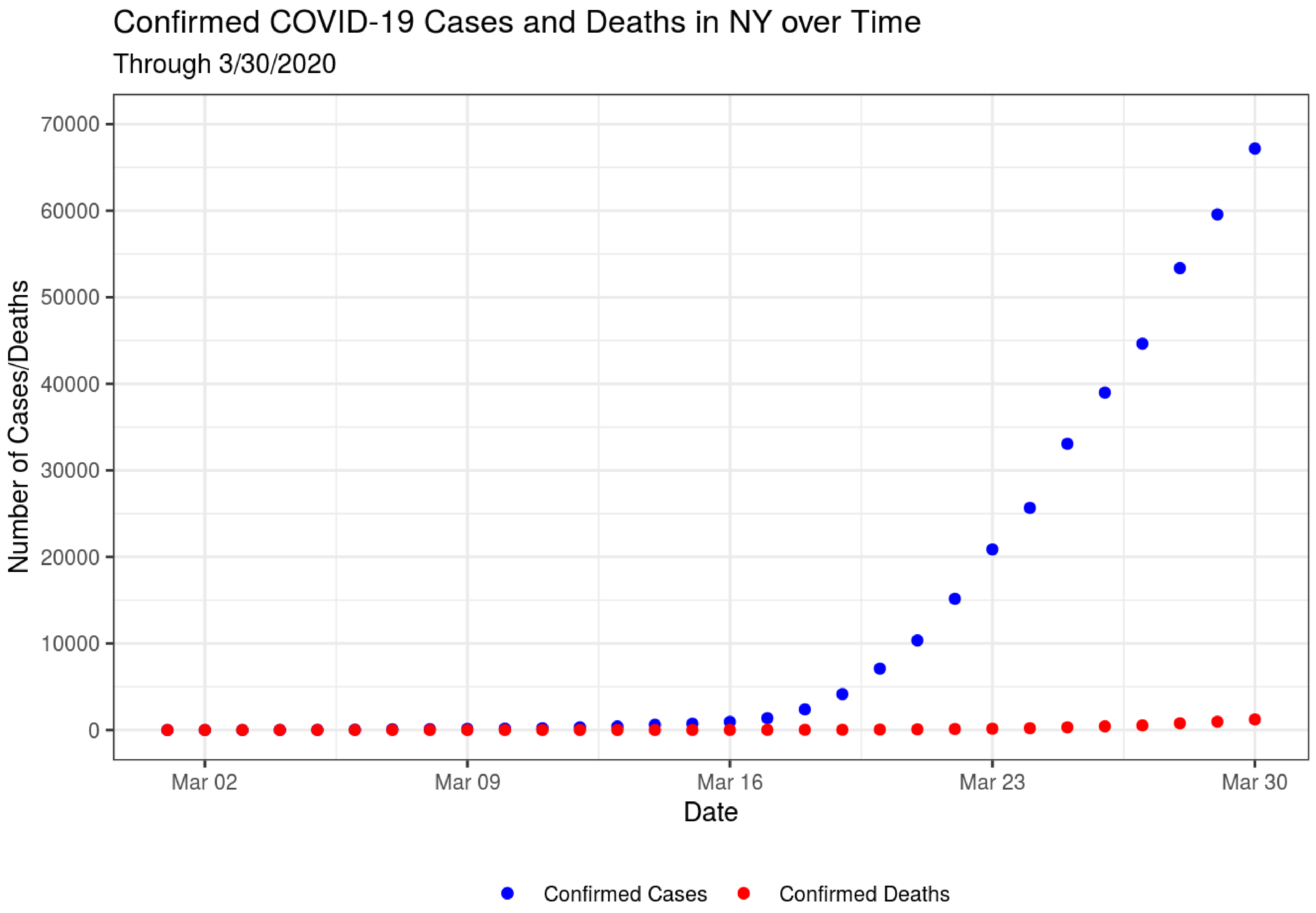
```
covid_df_long <- read_csv("data/us-states.csv")
```

The earliest date recorded in this dataset is January 21, 2020.

b)

Subset the data to only New York State. Create a scatter plot of New York State confirmed COVID-19 cases and confirmed COVID-19 deaths. Make the cases and deaths different colors.

```r
#Filter dataframe by New York obs
covid_df_NY <- covid_df_long %>%
  filter(state == "New York")

#Create scatterplot
ggplot(data = covid_df_NY) +
  geom_point(aes(x=date, y=cases, color="Confirmed Cases")) +
  geom_point(aes(x=date, y=deaths, color="Confirmed Deaths")) +
  theme_bw() +
  theme(legend.position = "bottom", text=element_text(size=10))
+
  scale_y_continuous(limits = c(0,70000),
                     breaks=seq(from=0,to=70000,by=10000)) +
  scale_color_manual(values = c("Confirmed Cases" = "blue",
                                "Confirmed Deaths" = "red")) +
  labs(title="Confirmed COVID-19 Cases and Deaths in NY over Tim
e", subtitle = "Through 3/30/2020", x="Date",
       y="Number of Cases/Deaths", color=NULL)
```

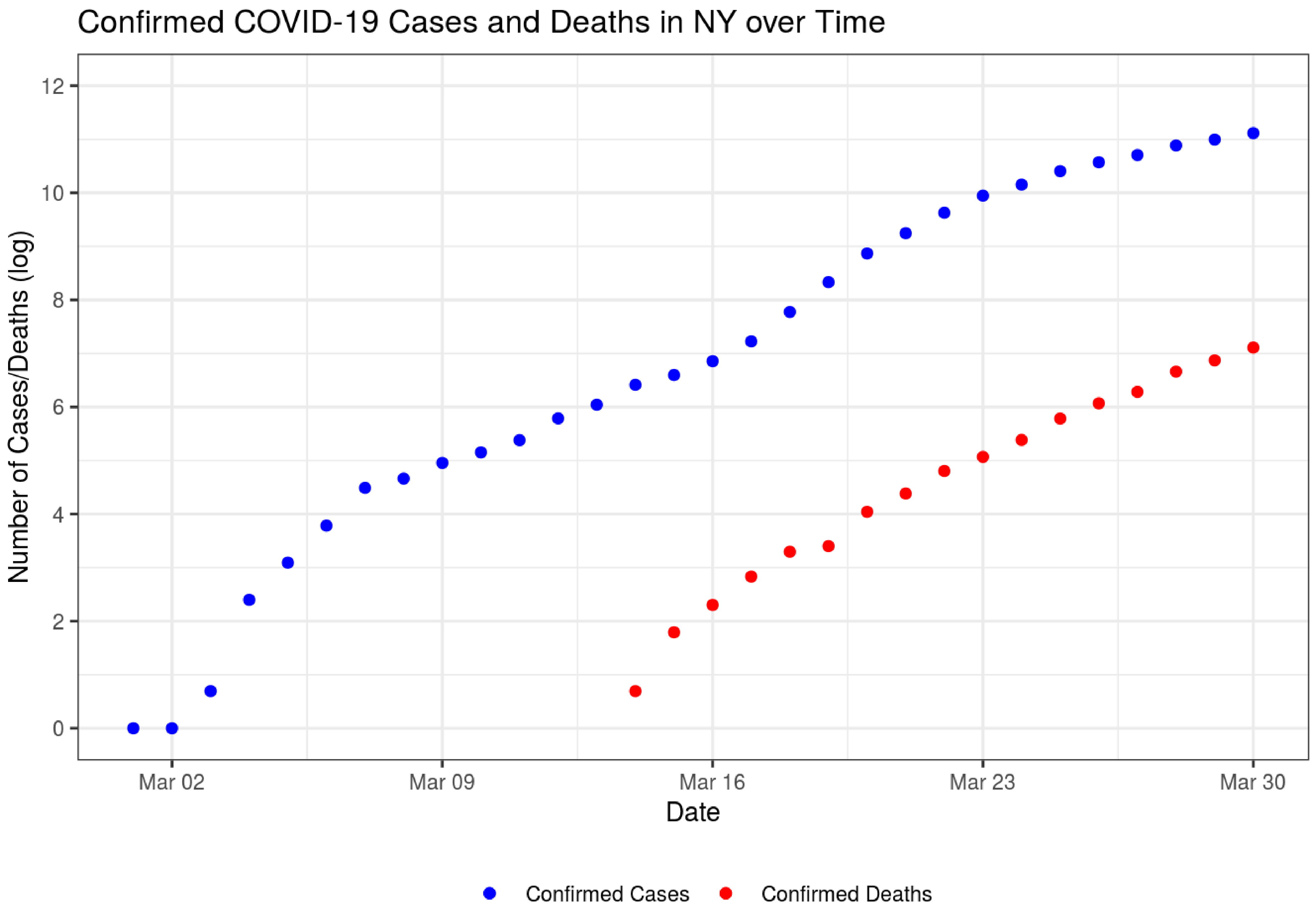

Confirmed COVID-19 Cases and Deaths in NY over Time
Through 3/30/2020
Number of Cases/Deaths
70000
60000
50000
40000
30000
20000
10000
0
Mar 02
Mar 09
Mar 16
Mar 23
Mar 30
Date
Confirmed Cases
Confirmed Deaths

c)

Using your New York data only, create new variables called `log_cases` *and* `log_deaths` *by applying log transformations to both* `cases` *and* `deaths` *variables using the* `log()` *function. When you do this, all of the death counts that were 0 will be set to* `-Inf`*, or negative infinity. You should use mutate to set values of* `-Inf` *to* `NA_real_`*. You can do this by using* `if_else()` *and the* `is.infinite()` *function to check if a value is infinite.*

Once this is done, recreate your graph from (b) for New York but use your cleaned `log_cases` *and* `log_deaths` *variables. Which graph do you think is better, the raw data or the log-transformed data?*

```
#Create new log-transformed variables
covid_df_NY <- covid_df_NY %>%
  mutate(log_cases = log(cases),
         log_deaths = if_else(is.infinite(log(deaths)),
                              NA_real_, log(deaths)))

#Create new scatterplot
ggplot(data = covid_df_NY) +
  geom_point(aes(x=date, y=log_cases, color="Confirmed Cases"))
+
  geom_point(aes(x=date, y=log_deaths, color="Confirmed Death
s"))+
  theme_bw() +
  theme(legend.position = "bottom", text=element_text(size=10))
+
  scale_y_continuous(limits=c(0,12), breaks=seq(0,12,by=2)) +
  scale_color_manual(values = c("Confirmed Cases" = "blue",
                       "Confirmed Deaths" = "red")) +
  labs(title="Confirmed COVID-19 Cases and Deaths in NY over Tim
e", x="Date", y="Number of Cases/Deaths (log)", color=NULL)
```



Confirmed COVID-19 Cases and Deaths in NY over Time
Number of Cases/Deaths (log)
0
2
4
6
8
10
12
Mar 02
Mar 09
Mar 16
Mar 23
Mar 30
Date
● Confirmed Cases
● Confirmed Deaths

I think the graph with log-transformed data looks better because the trend in confirmed deaths is clearer.

d)

Using your New York data only, create a new variable called `days` *by subtracting* `as.Date("2020-03-01")` *from the* `date` *variable.*

Then, fit a simple linear regression with `log_cases` *as the outcome and* `days` *as your predictor. R will automatically treat* `days` *as a continuous numeric, so each unit increase in* `days` *will represent 1 day since the first day in this New York State dataset (03/01/20). Report the overall R-squared of this model along with the table of slope parameter estimates using* `kable()`*. Does this model seem to explain a lot of the variance in* `log_cases`*?*

```
#Create new variable
covid_df_NY <- covid_df_NY %>%
  mutate(days = date - as.Date("2020-03-01"))

#Linear regression
covid_NY_lm <- lm(log_cases ~ days, data = covid_df_NY)

#Store fit stats and parameters estimates in tibbles
covid_NY_lm_fit <- glance(covid_NY_lm)
covid_NY_lm_param <- tidy(covid_NY_lm)

#Report parameter estimate tibble
kable(covid_NY_lm_param)
```

term	estimate	std.error	statistic	p.value
(Intercept)	1.2873979	0.2365217	5.443044	8.3e-06
days	0.3759335	0.0140062	26.840431	0.0e+00

The model R-squared value is 0.9625873. Since the R-squared is close to 1.0, the model does explain a lot of the variance in the log of confirmed cases.

e)

Your model from (d) has a log-transformed outcome `log_cases` *, so an interpretation of the date slope is not straightforward. However, we can use the parameter estimates to make predictions about future dates. We usually do not want to estimate outside of our data range, but in this case we will use information from past data to predict cases a few days in the future.*

Before we look to the future, first use the intercept and parameter estimates to predict the log number of cases in New York state for 03/25/20. In order to do this you can add the intercept estimate to the product of the slope estimate for `days` *and the number of days 03/25/20 is from 03/01/20 (this should be 24). Once you have predicted the log number of cases for 03/25/20, you can get the number of confirmed cases by using the* `exp()` *function on your result.*

Report the number of cases in New York State your model predicts for 03/25/20 and compare that to the number of actual cases on 03/25/20. Are the numbers close?

```
#Predict number of cases in NY on 3/25/2020
NY_pred_cases_3_25_2020 = exp(covid_NY_lm_param$estimate[1] +
                              (covid_NY_lm_param$estimate[2]*2
4))

print(paste("Predicted number of COVID-19 cases in NY on 3/25/20
20:", round(NY_pred_cases_3_25_2020, 0)))
```

```
## [1] "Predicted number of COVID-19 cases in NY on 3/25/2020: 3
0025"
```

```
print(paste("Reported number of COVID-19 cases in NY on 3/25/202
0: ", covid_df_NY$cases[25]))
```

```
## [1] "Reported number of COVID-19 cases in NY on 3/25/2020:  3
3066"
```

Yes, the numbers are fairly close.

f)

Write a function that uses the model intercept and parameter estimates you obtained in (e) to predict the number of confirmed cases in New York state on a given date. Your function should take the number of days from the first data point in New York State (03/01/20) and it should return the number of cases (not the log number of cases). Using this function, make predictions for confirmed cases in New York state for 03/30/20, 03/31/20, 04/01/20 and 04/02/20 and report them.

```r
#Create function
predict_NY_cases <- function(date) {

  mdy_date = lubridate::mdy(date)

  days = mdy_date - as.Date("2020-03-01")

  days = as.numeric(days)

  #Predict number of cases in NY on given date
  NY_pred_cases = covid_NY_lm_param$estimate[1] +
    (covid_NY_lm_param$estimate[2] * days)

  NY_pred_cases = exp(NY_pred_cases)

  #Return the predicted number of cases
  return(paste("Number of predicted COVID-19 cases in NY on",
               date, ":", round(NY_pred_cases, 0)))

}

#Plug in dates into function
predict_NY_cases("03/30/20")
```

```
## [1] "Number of predicted COVID-19 cases in NY on 03/30/20 : 1
96707"
```

```
predict_NY_cases("03/31/20")
```

```
## [1] "Number of predicted COVID-19 cases in NY on 03/31/20 : 2
86474"
```

```
predict_NY_cases("04/01/20")
```

```
## [1] "Number of predicted COVID-19 cases in NY on 04/01/20 : 4
17207"
```

```
predict_NY_cases("04/02/20")
```

```
## [1] "Number of predicted COVID-19 cases in NY on 04/02/20 : 6
07599"
```

g)

Compare your model prediction for 03/30/20 against the actual recorded cases in New York state for 03/30/20. Does this prediction seem reasonable or unreasonable? What do you think is happening with the model?

```
predict_NY_cases("03/30/20")
```

```
## [1] "Number of predicted COVID-19 cases in NY on 03/30/20 : 1
96707"
```

```
print(paste("Reported number of COVID-19 cases in NY on 3/30/2
0:",
            covid_df_NY$cases[30]))
```

```
## [1] "Reported number of COVID-19 cases in NY on 3/30/20: 6717
4"
```

The predicted number of cases in NY on that day is much higher than the reported number of cases, so the prediction seems unreasonable. The reason could be that the model is estimating linear growth in the log of the number of cases. It seems the predicted number of cases is grows faster than the reported number of cases, suggesting the actual growth in the log of the number of cases may actually level off at some point.

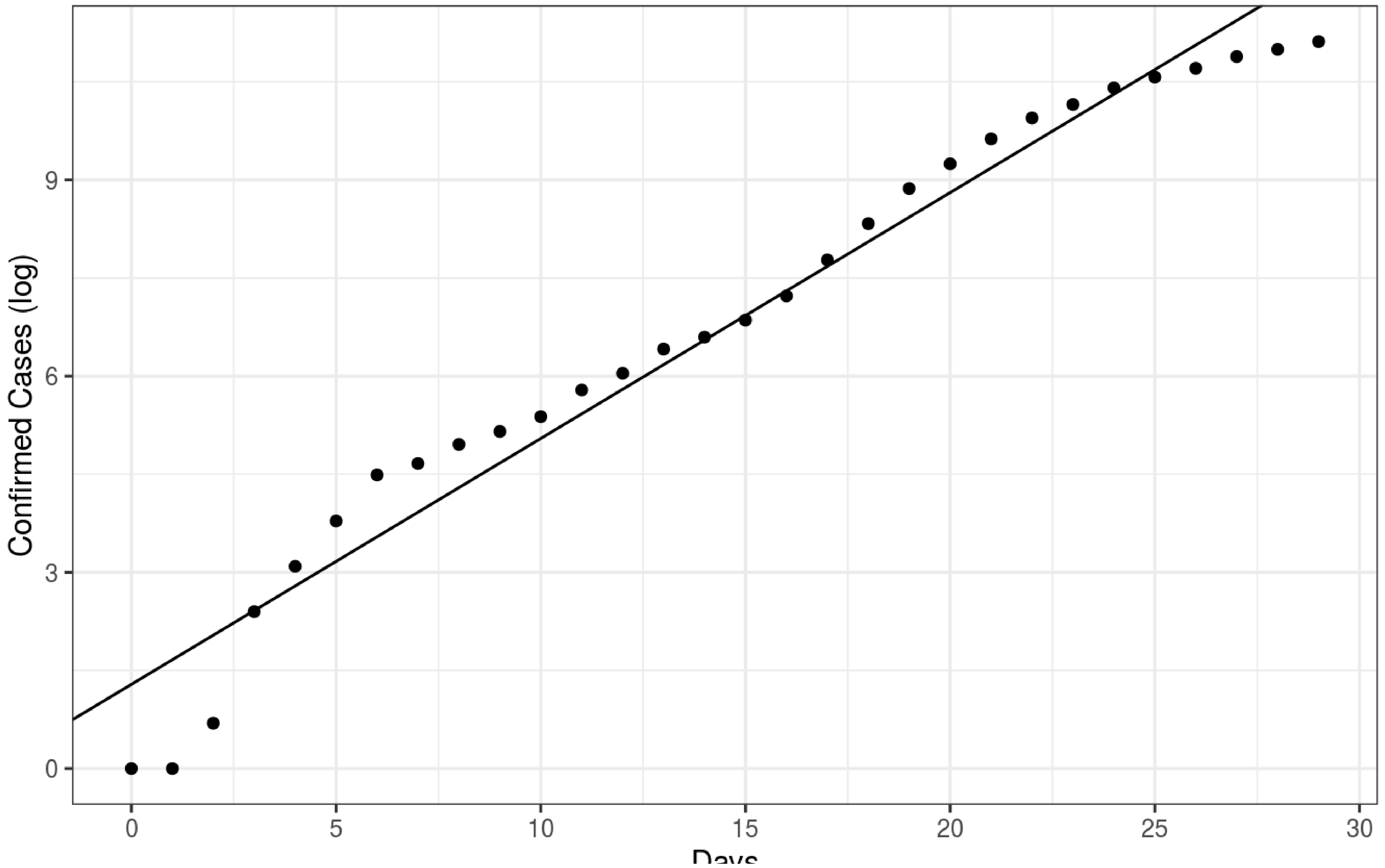
h)

Create a scatter plot of `log_cases` by `days` using the dataset you used to model New York state log cases in part (d). Add your modeled regression line to your plot by using a `geom_abline()` statement. This geom does not need mapping aesthetics (no `aes()`) but will take `intercept` and `slope` arguments. Set the intercept equal to your intercept estimate from your model in (d) and slope equal to the parameter estimate for `days` and you should see your modeled regression line. What do you think about how the line fits the data? Does this shed any light on part (g)?

```r
ggplot(data = covid_df_NY) +
  geom_point(aes(x=days, y=log_cases)) +
  geom_abline(intercept = covid_NY_lm_param$estimate[1],
              slope = covid_NY_lm_param$estimate[2]) +
  theme_bw() +
  scale_x_continuous(breaks = seq(0, 30, by = 5)) +
  labs(title="Confirmed COVID-19 Cases in NY Over Time",
       subtitle="Through 3/30/2020", x="Days",
       y="Confirmed Cases (log)")
```

Confirmed COVID-19 Cases in NY Over Time

Through 3/30/2020



The trendline doesn’t fit the scatterplot that well. It indicates that after 25 days, the predicted number of cases would exceed the reported number of cases. The earlier estimate predicted the number of cases after 29 days, and it overestimated the reported number of cases. This provides some support to the conclusion drawn from the graph.

i)

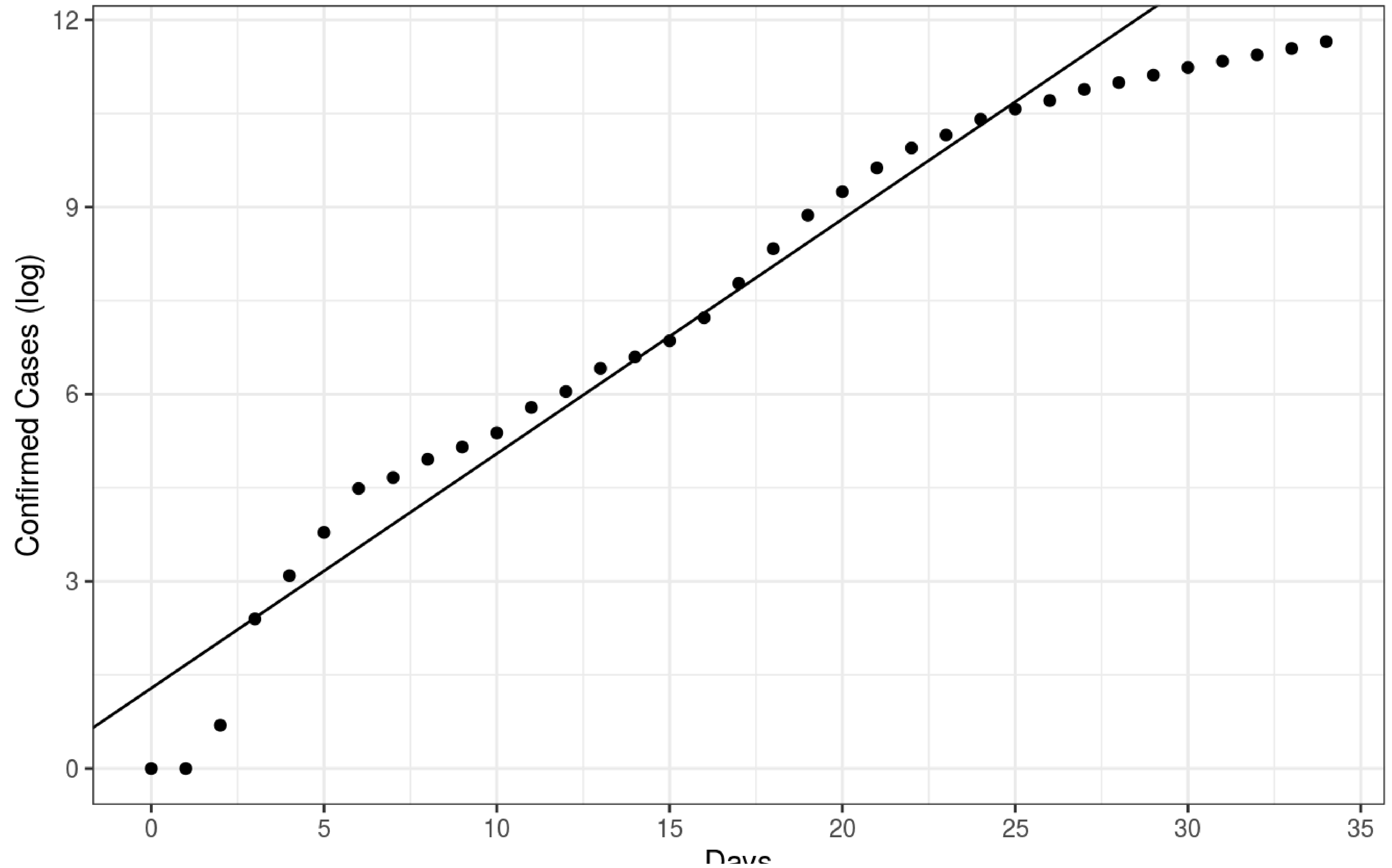
Go to https://github.com/nytimes/covid-19-data (https://github.com/nytimes/covid-19-data) and download the latest US States data. You should be able to download it by right-clicking on the links on the page that say “Raw CSV” and selecting “Save link as”.

Unless you finished this homework extremely early, your updated data should contain cases for at least 03/31/20. Please recreate the graph of `log_cases` *by* `days` *using your updated dataset (including the regression line from your model in* *part (e)) – don’t run a new regression using the updated data. How do you think the regression line you modeled using older data is fitting the new updated log case count?*

```r
#Read in new file
covid_df_updated <- read_csv("data/us-states_04042020.csv")

#Filter NY data and create new variables
covid_df_updated_NY <- covid_df_updated %>%
  filter(state == "New York") %>%
  mutate(date = lubridate::mdy(date)) %>%
  mutate(log_cases = log(cases),
         log_deaths = if_else(is.infinite(log(deaths)),
                              NA_real_, log(deaths))) %>%
  mutate(days = date - as.Date("2020-03-01"))

#Create a new graph
ggplot(data = covid_df_updated_NY) +
  geom_point(aes(x=days, y=log_cases)) +
  geom_abline(intercept = covid_NY_lm_param$estimate[1],
             slope = covid_NY_lm_param$estimate[2]) +
  theme_bw() +
  scale_x_continuous(breaks = seq(0, 35, by = 5)) +
  labs(title="Confirmed COVID-19 Cases in NY Over Time",
       subtitle="Through 4/4/2020", x="Days",
       y="Confirmed Cases (log)")
```

Confirmed COVID-19 Cases in NY Over Time

Through 4/4/2020



Days

I don’t think the regression line is fitting the data very well. It further supports the conclusion that after 25 days, the model will overestimate the reported number of cases.

j)

Would you want to use this linear model to predict what will happen far into the future? For example, would you want to use it to predict confirmed cases on 05/15/20? Why or why not? Share your thoughts!

No, I wouldn’t want to use this linear model to predict the number of cases far into the future because it seems to overestimate the number of cases beyond 25 days after the first date of data. The model predicts that the log of the number of cases will grow linearly indefinitely, but that’s likely not what will happen in reality. In reality, the growth in the log of the number of cases will likely level off and then decline. So, the optimal model would be a non-linear model; perhaps a quadratic function would be better.

Question 4

The New York Times has also compiled County-level data and made it freely available at https://github.com/nytimes/covid-19-data (https://github.com/nytimes/covid-19-data). Please download the latest county-level dataset for the following questions.

a)

How many counties are included in the county-level dataset? Which county has the most observations? How many counties only have 1 observation?

```r
#Read in the file
covid_df_counties_long <- read_csv("data/us-counties_04042020.cs
v")

#Update date variables and add log-transformed variables
covid_df_counties_long <- covid_df_counties_long %>%
  mutate(date = lubridate::mdy(date)) %>%
  mutate(log_cases = log(cases),
         log_deaths = if_else(is.infinite(log(deaths)),
                              NA_real_, log(deaths)))

#Find out how many counties there are
covid_df_by_county <- covid_df_counties_long %>%
  group_by(county) %>%
  summarize(n = n())

DT::datatable(covid_df_by_county)
```

Show 10 entries Search:

	county	n
1	Abbeville	17
2	Acadia	14
3	Accomack	16
4	Ada	23
5	Adair	43
6	Adams	140
7	Addison	16
8	Aiken	16

	county	n
9	Alachua	26
10	Alamance	13

Showing 1 to 10 of 1,497 entries

Previous 1 2 3 4 5 … 150 Next

```
#Re-order df by number of obs (descending order)
#Find out which county has the most obs
attach(covid_df_by_county)
covid_df_by_county <- covid_df_by_county[order(-n),]

DT::datatable(covid_df_by_county)
```

Show 10 entries Search:

	county	n
1	Washington	445
2	Unknown	436
3	Jefferson	331
4	Franklin	293
5	Jackson	269
6	Montgomery	268
7	Lincoln	237
8	Madison	218

	county	n
9	Douglas	202
10	Monroe	190

Showing 1 to 10 of 1,497 entries

Previous 1 2 3 4 5 … 150 Next

```
#Find out how many counties have only 1 obs
covid_df_county_one <- covid_df_by_county %>%
  filter(n == 1)

DT::datatable(covid_df_county_one)
```

Show 10 **entries** **Search:**

	county	n
1	Ashe	1
2	Beaverhead	1
3	Bronx	1
4	Caribou	1
5	Cherry	1
6	Dixie	1
7	Hemphill	1
8	Jewell	1

	county	n
9	Los Alamos	1
10	Manitowoc	1

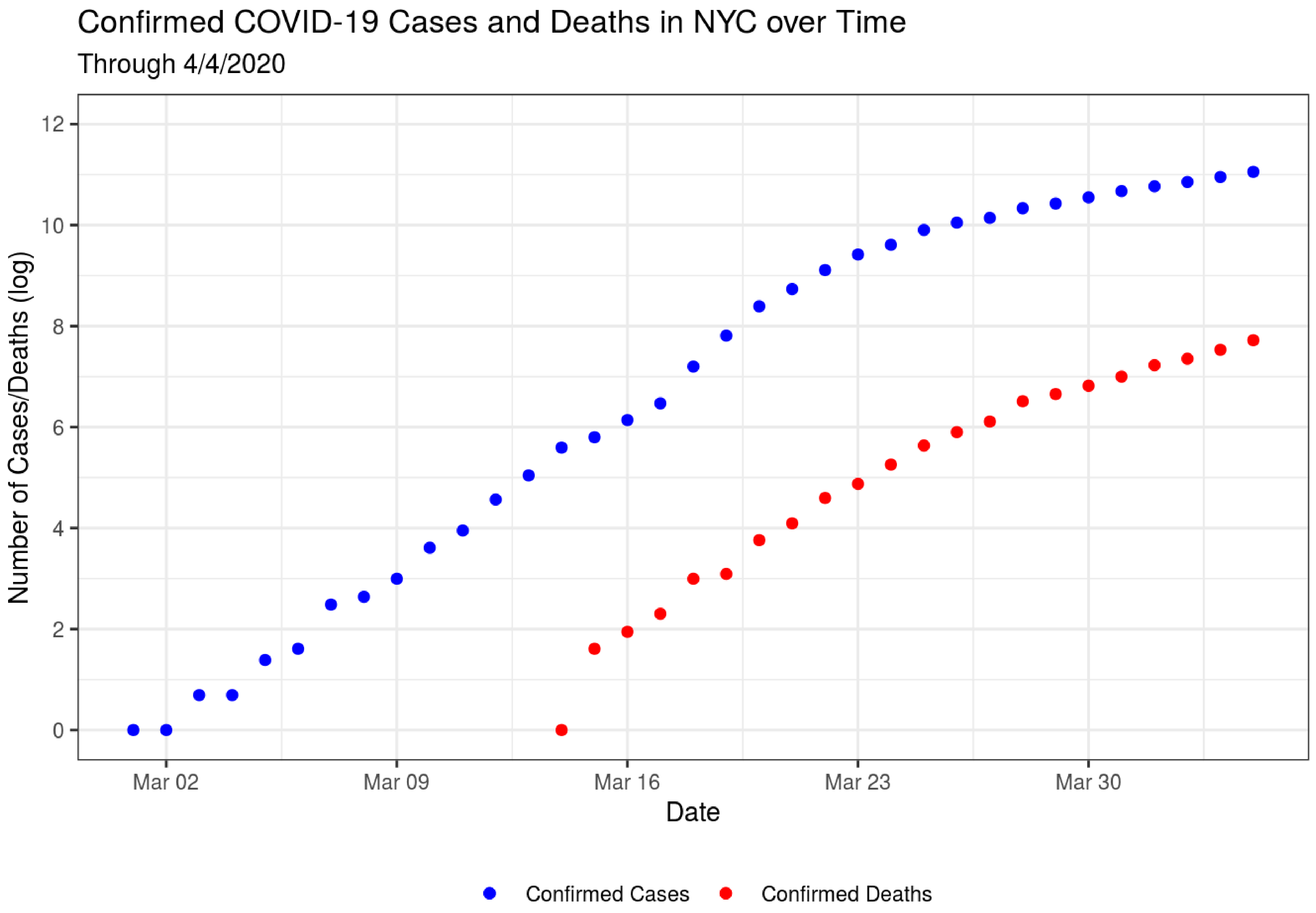
Showing 1 to 10 of 24 entries Previous 1 2 3 Next

There are 1,497 counties in the county-level dataset. Washington county has the most observations (445). 24 counties only have one observation.

b)

Recreate the plot from question 3(c) for New York City. Make sure it includes both log cases and log deaths in different colors, and that it has informative titles. Please add the date you downloaded the data as a subtitle in your graph.

```
#Create new log-transformed variables
covid_df_NYC <- covid_df_counties_long %>%
  filter(county == "New York City")

#Create new scatterplot
ggplot(data = covid_df_NYC) +
  geom_point(aes(x=date, y=log_cases, color="Confirmed Cases"))
+
  geom_point(aes(x=date, y=log_deaths, color="Confirmed Death
s"))+
  theme_bw() +
  theme(legend.position = "bottom", text=element_text(size=10))
+
  scale_y_continuous(limits=c(0,12), breaks=seq(0,12,by=2)) +
  scale_color_manual(values = c("Confirmed Cases" = "blue",
                     "Confirmed Deaths" = "red")) +
  labs(title="Confirmed COVID-19 Cases and Deaths in NYC over Ti
me",
        subtitle="Through 4/4/2020", x="Date",
        y="Number of Cases/Deaths (log)", color=NULL)
```

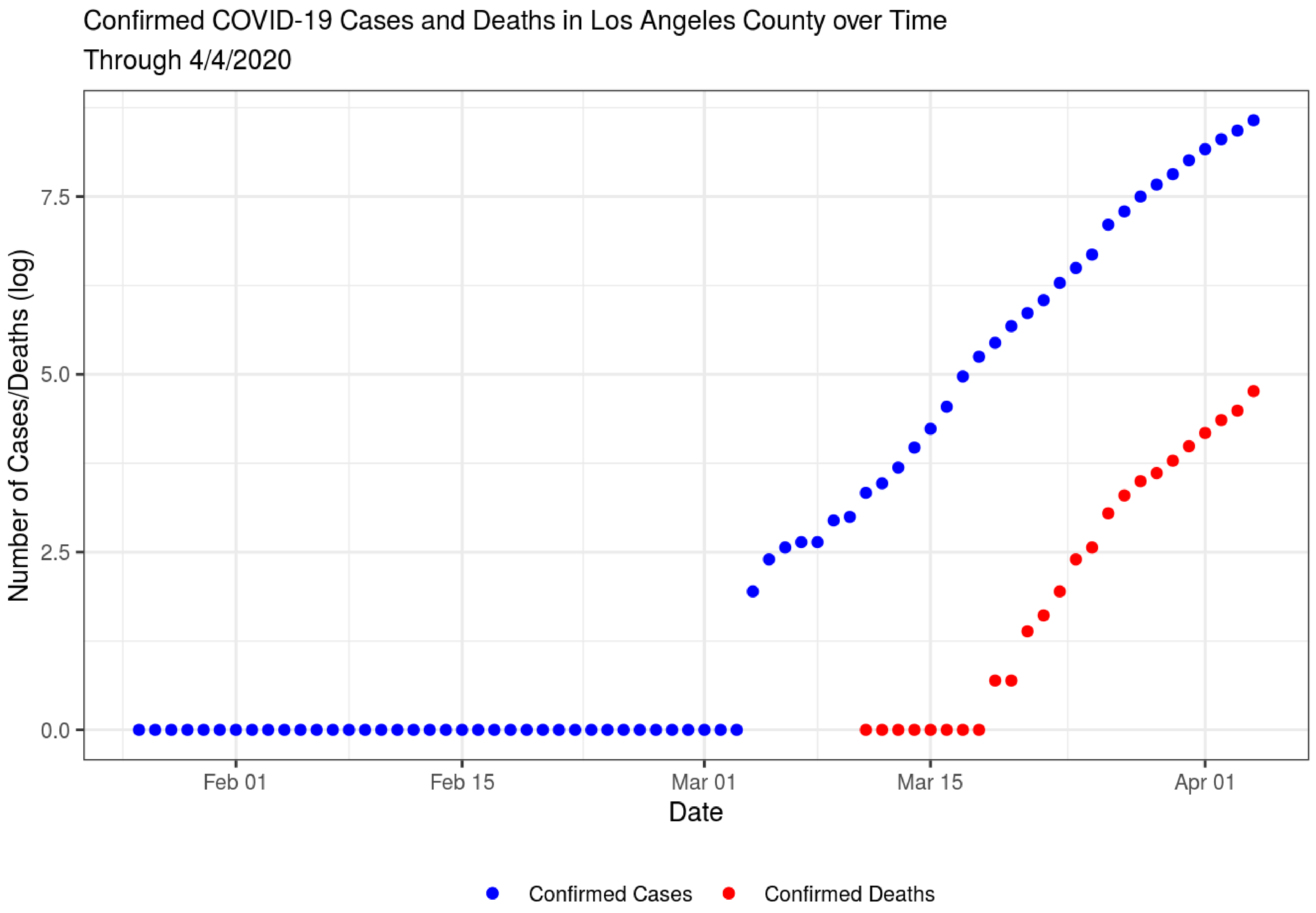

Confirmed COVID-19 Cases and Deaths in NYC over Time
Through 4/4/2020
Number of Cases/Deaths (log)
12
10
8
6
4
2
0
Mar 02
Mar 09
Mar 16
Mar 23
Mar 30
Date
Confirmed Cases
Confirmed Deaths

c)

Write a function that will create the graph in part (b) for any county in the dataset. It is probably easiest if you create the `log_cases` and `log_deaths` variables (and deal with `-Inf` issues) outside of your function and refer to that prepared data frame in your function. Your function should take the county to graph in as a string variable and it should output a nice scatter plot with a custom title that uses `str_c()` to incorporate the name of the county.

```
plot_cases_deaths <- function(cty) {

  #create filtered dataframe
  covid_df_county <- covid_df_counties_long %>%
    filter(county == cty)

  #Plot the graph
  plot <- ggplot(data = covid_df_county) +
    geom_point(aes(x=date, y=log_cases, color="Confirmed Case
s"))+
    geom_point(aes(x=date,y=log_deaths,color="Confirmed Death
s"))+
    theme_bw() +
    theme(legend.position = "bottom", text=element_text(size=1
0),
          plot.title = element_text(size=10)) +
    scale_color_manual(values = c("Confirmed Cases" = "blue",
                              "Confirmed Deaths" = "red")) +
    labs(title=str_c("Confirmed COVID-19 Cases and Deaths in", c
```

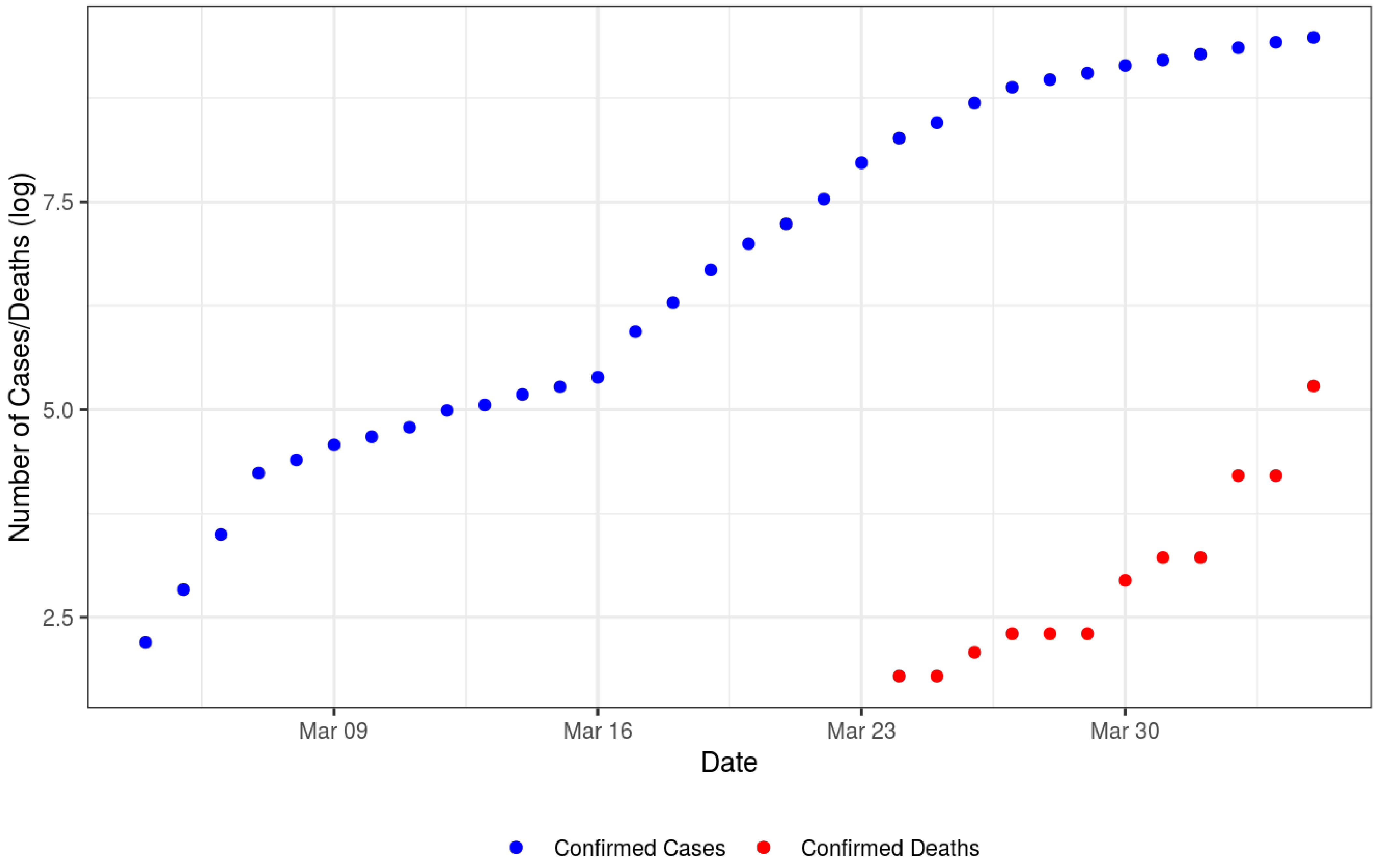
```r
ty, "County over Time", sep = " "),
         subtitle="Through 4/4/2020", x="Date",
         y="Number of Cases/Deaths (log)", color=NULL)

  #Return the graph
  return(plot)

}

#Test the function
plot_cases_deaths("Los Angeles")
```



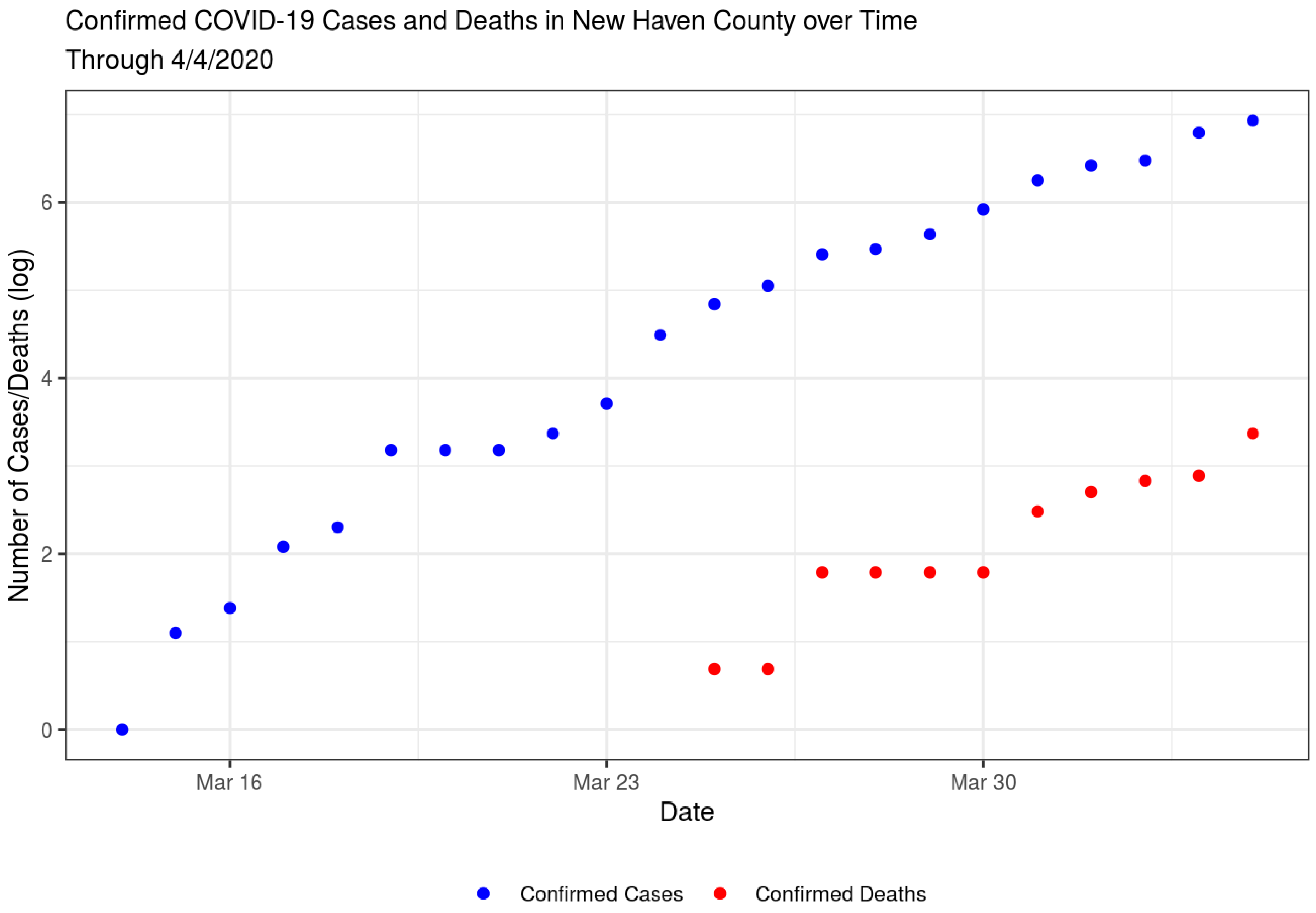
Confirmed COVID-19 Cases and Deaths in Los Angeles County over Time
Through 4/4/2020
Number of Cases/Deaths (log)
7.5
5.0
2.5
0.0
Feb 01
Feb 15
Mar 01
Mar 15
Apr 01
Date
● Confirmed Cases
● Confirmed Deaths

```
plot_cases_deaths("Westchester")
```

Confirmed COVID-19 Cases and Deaths in Westchester County over Time
Through 4/4/2020



```
plot_cases_deaths("New Haven")
```


Confirmed COVID-19 Cases and Deaths in New Haven County over Time
Through 4/4/2020
Number of Cases/Deaths (log)
0
2
4
6
Mar 16
Mar 23
Mar 30
Date
● Confirmed Cases
● Confirmed Deaths