Homework 3. Interactive Visualization

Zeming Zhang

2023-02-20

```
library(data.table)
library(dplyr)
library(dplyr)
library(tidyr)
library(plotly)
library(lubridate)
```

In this homework you should use plotly unless said otherwise.

To create pdf version of your homework, knit it first to html and then print it to pdf. Interactive plotly plots can be difficult sometimes to convert to static images suitable for insertion to LaTex documents (that is knitting to PDF).

Look for questions in R-chunks as comments and plain text (they are prefixed as Q.).

Part 1. Iris Dataset. (20 points)

> "The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis" https://en.wikipedia.org/wiki/Iris_flower_data_set (https://en.wikipedia.org/wiki/Iris_flower_data_set)

```
# Q1.1. Read the iris.csv file  (2 points)
# hint: use fread from data.table, it is significantly faster than default methods
#       be sure to have strings as factors (see stringsAsFactors argument)

iris <- data.frame(fread("iris.csv", stringsAsFactors = TRUE))

# display summary of data frame
summary(iris)
```

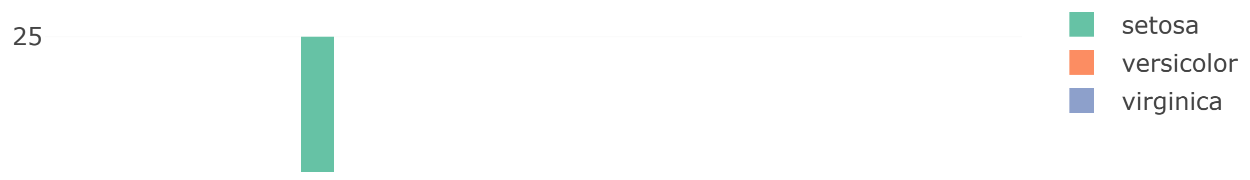
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
## 
```

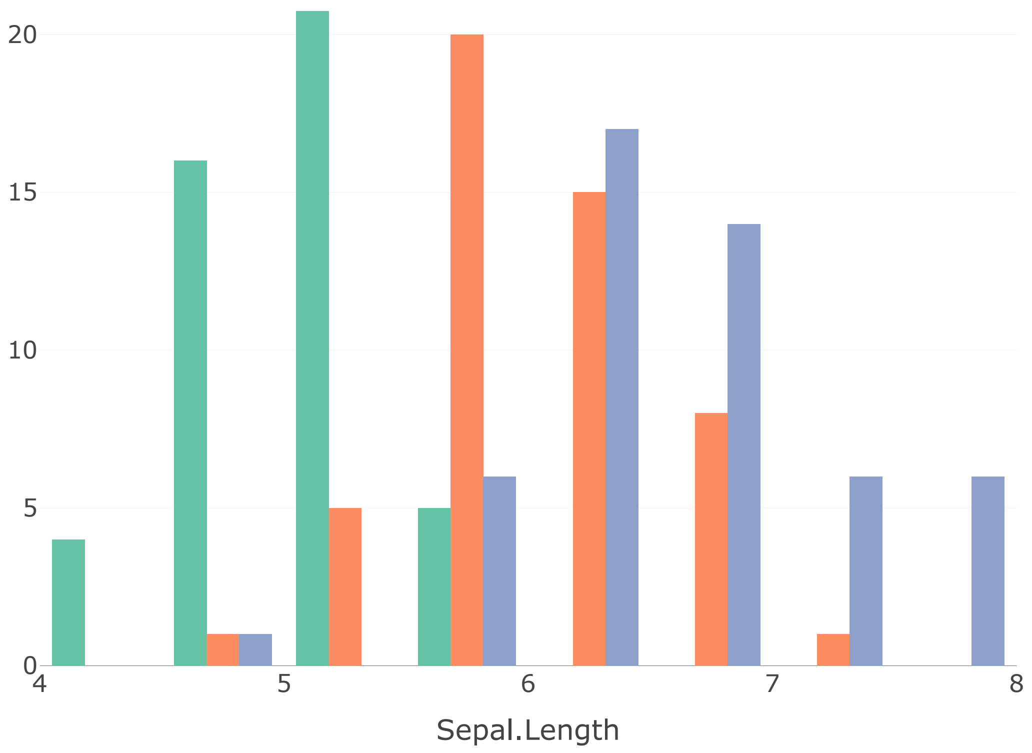
```
# Q1.2. Show some values from data frame (2 points)
head(iris, 100)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

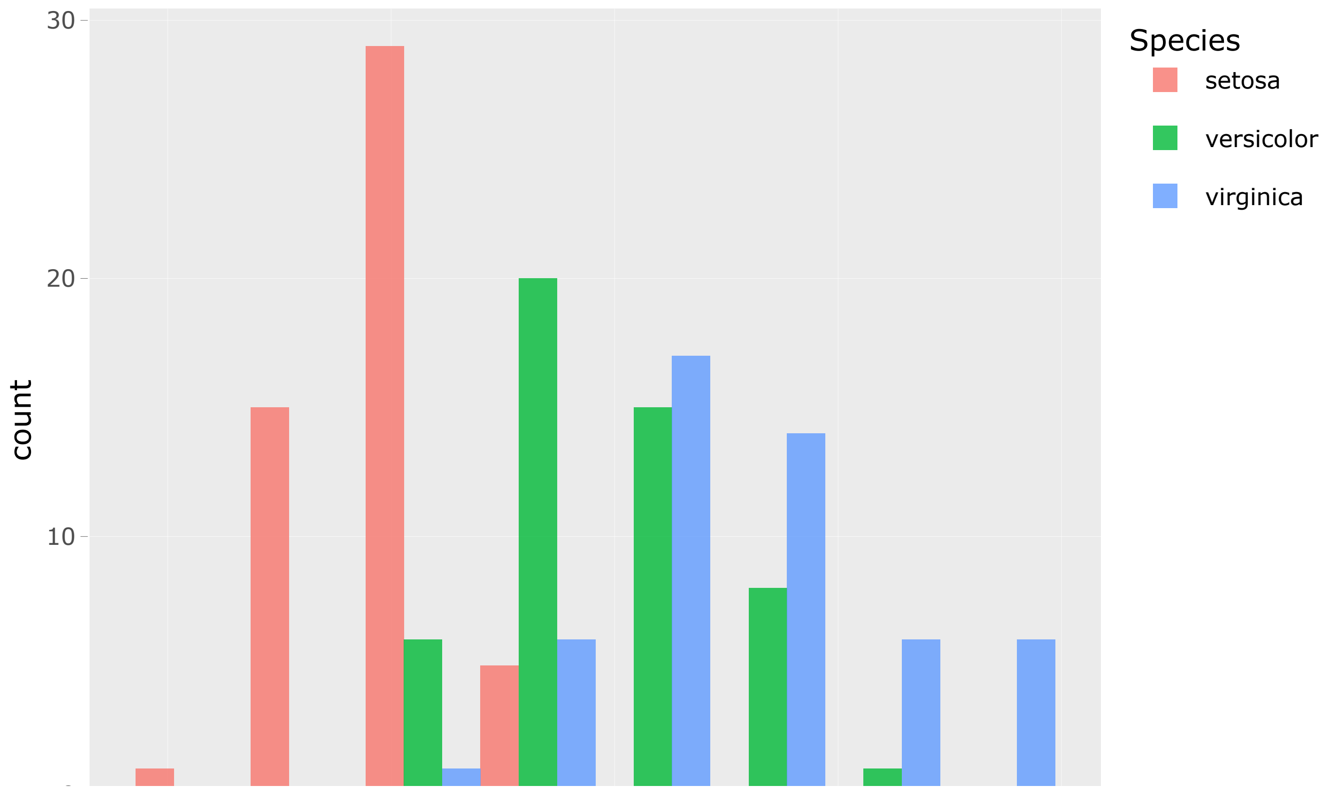
1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

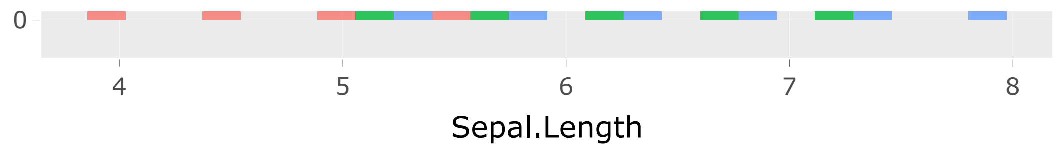
```
# Q1.3. Build histogram plot for Sepal.Length variable for each species using plot_ly
# (use color argument for grouping) (2 points)
# should be one plot
plot_ly(data = iris, x = ~Sepal.Length, color = ~Species, type = "histogram")
```



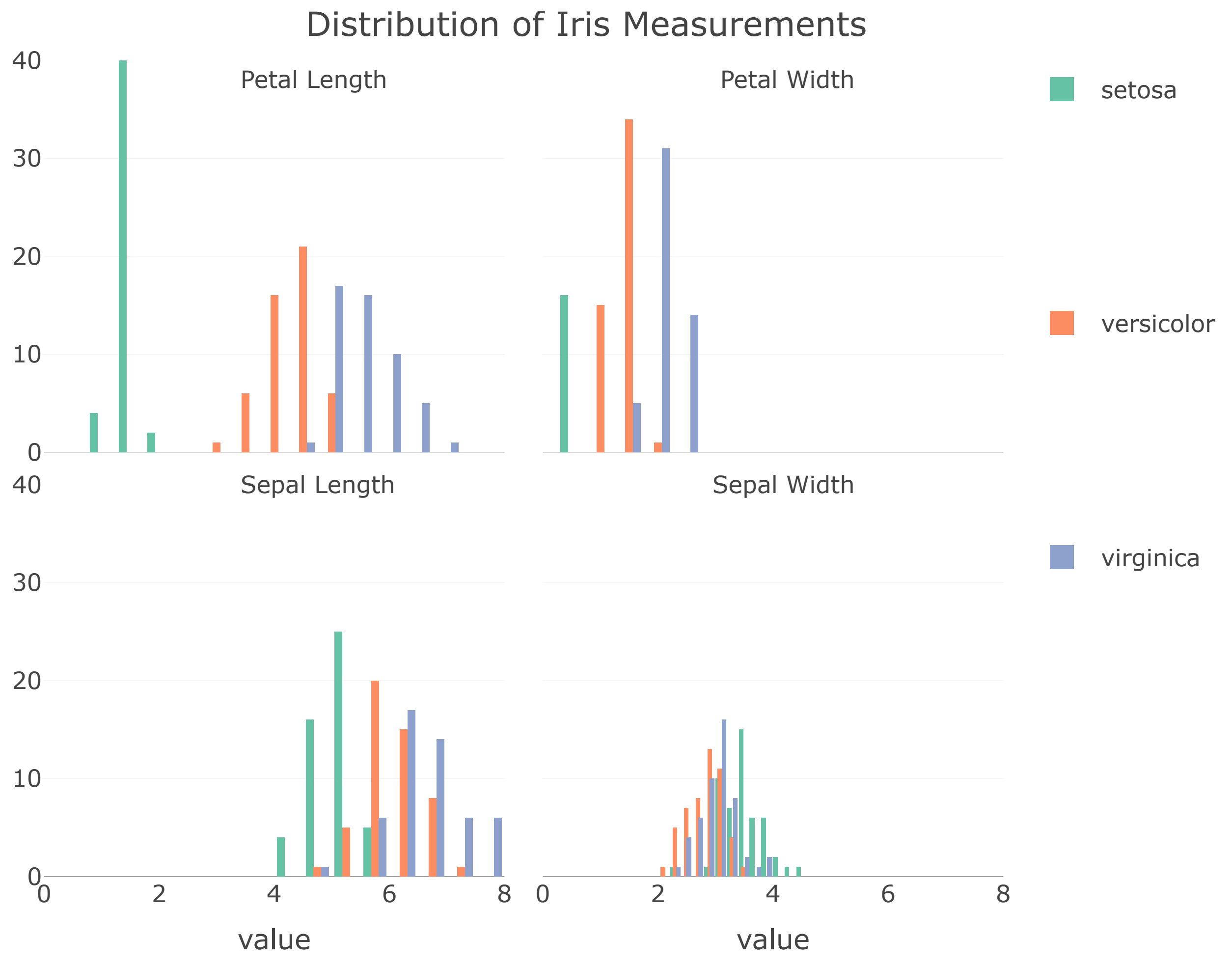


```
# Q1.4. Repeat previous plot with ggplot2 and convert it to plotly with ggplotly (2 poin
ts)
# Create histogram with ggplot2
p <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_histogram(position = "dodge", bins = 8, alpha = 0.8)

# Convert to plotly with ggplotly
ggplotly(p)
```




0
4
5
6
7
8
Sepal.Length

```r
# Q1.5. Create facet 2 by 2 plot with histograms similar to previous but for each metric
# (2 points)
# hint:
#   following conversion to long format can be useful:
#   iris %>% gather(key = "metric", value = "value",-Species)
#

## convert iris dataset to long format
iris_long <- iris %>% gather(key = "metric", value = "value", -Species)

plot1 <- plot_ly(iris_long %>% filter(metric %in% c("Petal.Length")),
                 x = ~value, color = ~Species, type = "histogram",
                 legendgroup = ~Species, showlegend = TRUE) %>%
  layout(
    xaxis = list(range = c(0, 8)),
    yaxis = list(range = c(0, 40)),
    autosize = TRUE
  )

plot2 <- plot_ly(iris_long %>% filter(metric %in% c("Petal.Width")),
                 x = ~value, color = ~Species, type = "histogram",
                 legendgroup = ~Species, showlegend = FALSE) %>%
  layout(
    xaxis = list(range = c(0, 8)),
    yaxis = list(range = c(0, 40)),
    autosize = TRUE
  )

plot3 <- plot_ly(iris_long %>% filter(metric %in% c("Sepal.Length")),
                 x = ~value, color = ~Species, type = "histogram",
                 legendgroup = ~Species, showlegend = FALSE) %>%
  layout(
    xaxis = list(range = c(0, 8)),
    yaxis = list(range = c(0, 40)),
    autosize = TRUE
  )

plot4 <- plot_ly(iris_long %>% filter(metric %in% c("Sepal.Width")),
                 x = ~value, color = ~Species, type = "histogram",
                 legendgroup = ~Species, showlegend = FALSE) %>%
  layout(
    xaxis = list(range = c(0, 8)),
    yaxis = list(range = c(0, 40)),
    autosize = TRUE
  )

subplot(plot1, plot2, plot3, plot4, nrows = 2, shareX = TRUE, shareY = TRUE) %>%
  layout(
    title = "Distribution of Iris Measurements",
    autosize = TRUE,
    legend = list(tracegroupgap = 100),
    showlegend = TRUE,
    annotations = list(
    list(x = 0.2 , y = 1, text = "Petal Length", showarrow = F,
         xref='paper', yref='paper'),
    list(x = 0.85 , y = 1, text = "Petal Width", showarrow = F,
         xref='paper', yref='paper'),
    list(x = 0.2 , y = 0.48, text = "Sepal Length", showarrow = F,
         xref='paper', yref='paper'),
    list(x = 0.85 , y = 0.48, text = "Sepal Width", showarrow = F,
         xref='paper', yref='paper')
    )
  )
```

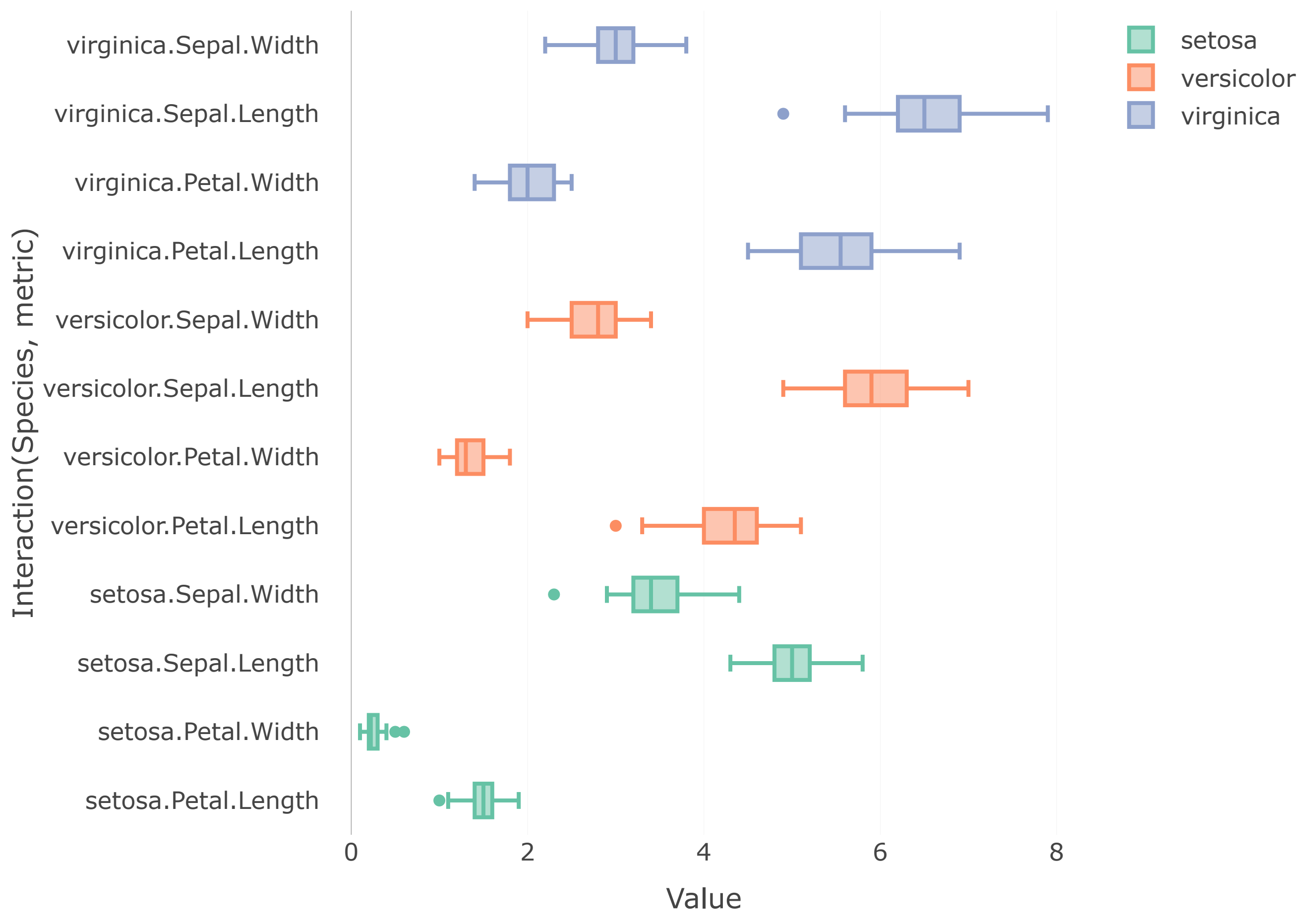


Q1.6. Which metrics has best species separations? (2 points)

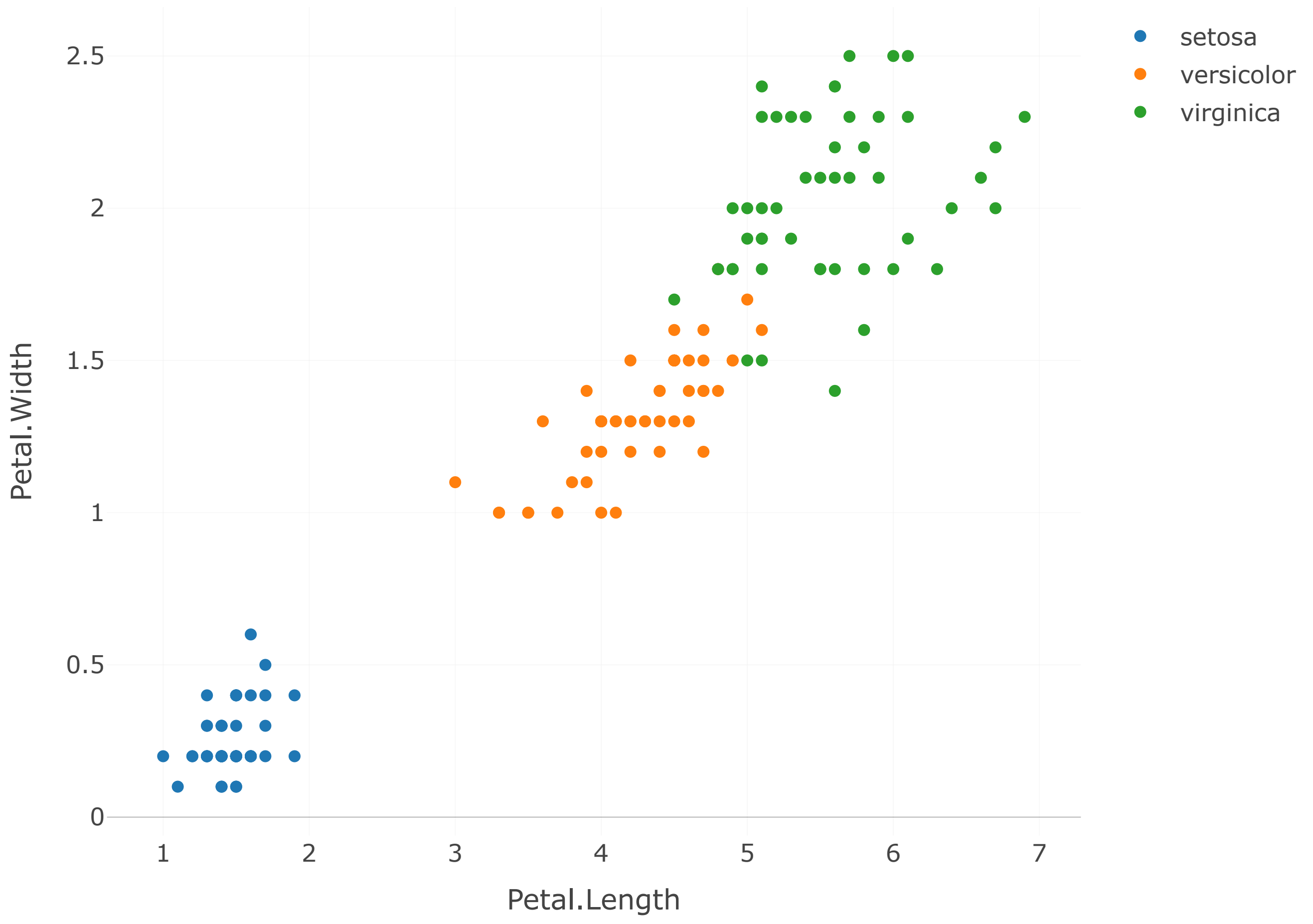
From the 2 by 2 facet plot with histograms for each metric, it looks like the Petal.Width and Petal.Length metrics have the best species separations. This is because the histograms for these metrics show the least overlap between the different species, with each species having a distinct range of values for these metrics.

In contrast, the Sepal.Width and Sepal.Length metrics have more overlap between the different species, with some species having similar ranges of values for these metrics. This makes it harder to distinguish between these species based on these metrics alone.

```
# Q1.7. Repeat above plot but using box plot (2 points)

library(plotly)
library(tidyr)
library(dplyr)

iris_long <- iris %>%
  pivot_longer(cols = -Species, names_to = "metric", values_to = "value")

iris_long <- cbind(iris_long %>% unite("Species_metric", Species, metric,
                                      sep = "."), iris_long %>% select(Species))

plot_ly(data = iris_long, y = ~Species_metric, x = ~value, type = "box", color

        = ~Species, showlegend = TRUE) %>%
  layout(yaxis = list(title = "Interaction(Species, metric)"),
         xaxis = list(title = "Value"))
```

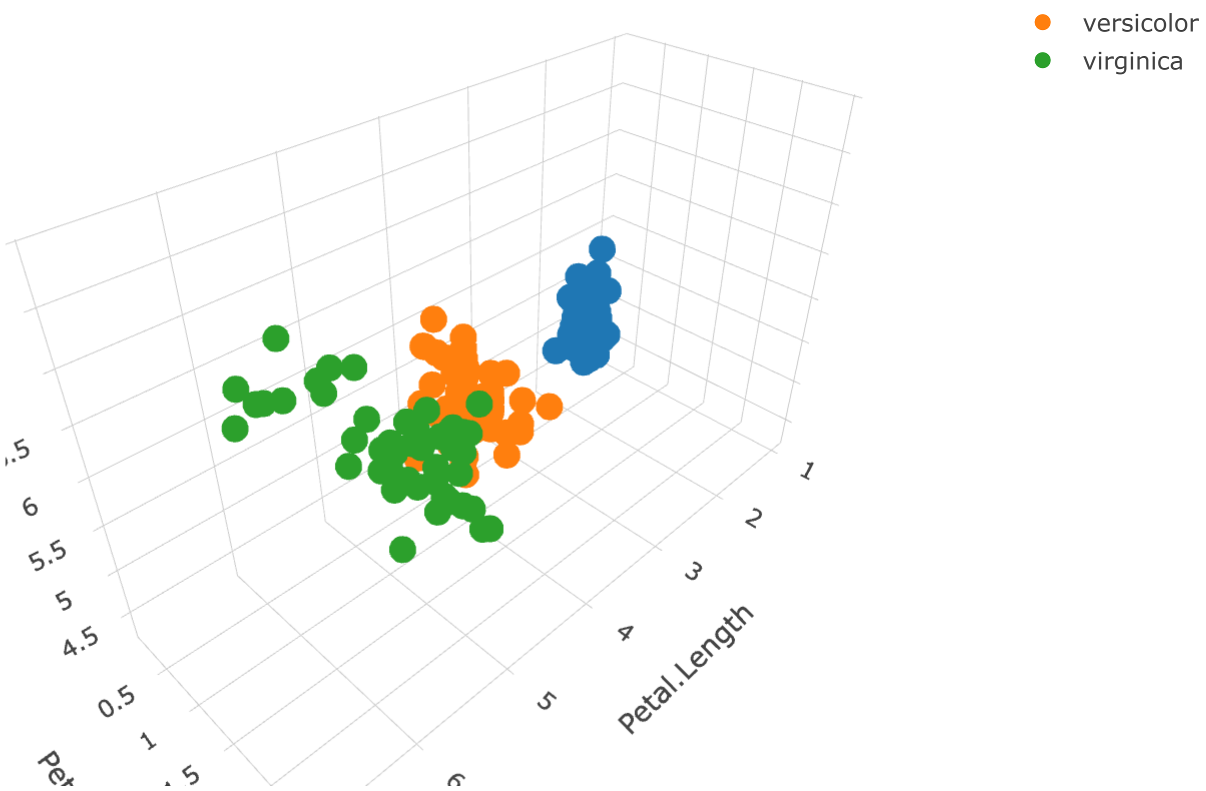


```
# Q1.8. Choose two metrics which separates species the most and use it to make scatter p
lot
# color points by species (2 points)
# create scatter plot with points colored by species
plot <- iris %>%
  plot_ly(x = ~Petal.Length, y = ~Petal.Width, color = ~Species,
          colors = c("#1f77b4", "#ff7f0e", "#2ca02c")) %>%
  add_markers()

plot
```



```
# Q1.9. Choose three metrics which separates species the most and use it to make 3d plot
# color points by species (2 points)

# create 3D scatter plot with points colored by species
plot <- iris %>%
  plot_ly(x = ~Petal.Length, y = ~Petal.Width, z = ~Sepal.Length,
          color = ~Species, colors = c("#1f77b4", "#ff7f0e", "#2ca02c")) %>%
  add_markers()

plot
```

● setosa



Q1.10. Comment on species separation (2 points):

Based on the plots we've created, we can see that the three species of iris (setosa, versicolor, and virginica) can be separated quite well using different combinations of the four measured variables (sepal length, sepal width, petal length, and petal width).

The scatter plot we created using Petal.Length and Petal.Width shows that setosa flowers tend to have smaller petals than the other two species, while virginica and versicolor have similar ranges of petal lengths and widths.

In the 3D scatter plot we created using Petal.Length, Petal.Width, and Sepal.Length, we can see that the three species occupy different regions of the plot, with setosa flowers having the smallest petals and sepal lengths, and virginica flowers having the largest petals and sepal lengths, with versicolor flowers falling somewhere in between.

Overall, these plots suggest that the four measured variables are good indicators for distinguishing between the three iris species.

Part 2. Covid-19 Dataset. (20 points)

Download us-states.csv (https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv) (there is also a copy in homework assignment) from https://github.com/nytimes/covid-19-data/ (https://github.com/nytimes/covid-19-data/). README.md (https://github.com/nytimes/covid-19-data/blob/master/README.md) for details on file content.

```
# Q2.1. Read us-states.csv (2 points)
# read us-states.csv file

covid_data <- data.frame(fread("us-states.csv", stringsAsFactors = TRUE))

# display summary of data frame
summary(covid_data)
```

```
##       date                        state            fips           cases         
##  Min.   :2020-01-21   Washington   :  957   Min.   : 1.00   Min.   :       1  
##  1st Qu.:2020-10-19   Illinois     :  954   1st Qu.:17.00   1st Qu.:   49452  
##  Median :2021-06-06   California   :  953   Median :31.00   Median :  265613  
##  Mean   :2021-06-04   Arizona      :  952   Mean   :32.18   Mean   :  692919  
##  3rd Qu.:2022-01-20   Massachusetts:  946   3rd Qu.:46.00   3rd Qu.:  828389  
##  Max.   :2022-09-03   Wisconsin    :  942   Max.   :78.00   Max.   :11117372  
##                       (Other)      :44982                                     
##      deaths     
##  Min.   :    0  
##  1st Qu.:  826  
##  Median : 4022  
##  Mean   :10068  
##  3rd Qu.:12575  
##  Max.   :95097  
## 
```

```
# Q2.2. Show some values from dataframe

# view first few rows of data
head(covid_data, 50)
```

	date <IDate>	state <fct>	fips <int>	cases <int>	deaths <int>
1	2020-01-21	Washington	53	1	0
2	2020-01-22	Washington	53	1	0
3	2020-01-23	Washington	53	1	0
4	2020-01-24	Illinois	17	1	0
5	2020-01-24	Washington	53	1	0
6	2020-01-25	California	6	1	0
7	2020-01-25	Illinois	17	1	0
8	2020-01-25	Washington	53	1	0
9	2020-01-26	Arizona	4	1	0
10	2020-01-26	California	6	2	0

1-10 of 50 rows   Previous **1** 2 3 4 5 Next

```
# Q2.3. Create new dataframe with new cases per month for each state (2 points)
# hint:
#    is cases column cumulative or not cumulative?

covid_data_monthly <- data.frame(covid_data %>%
  mutate(month = format(as.Date(date), "%Y-%m")) %>%
  group_by(state, fips, month) %>%
  summarize(cases_cumulative = max(cases),
            deaths_cumulative = max(deaths),
            new_cases = max(cases) - min(cases)))
```

```
## `summarise()` has grouped output by 'state', 'fips'. You can override using the
## `.groups` argument.
```

```
covid_data_monthly
```

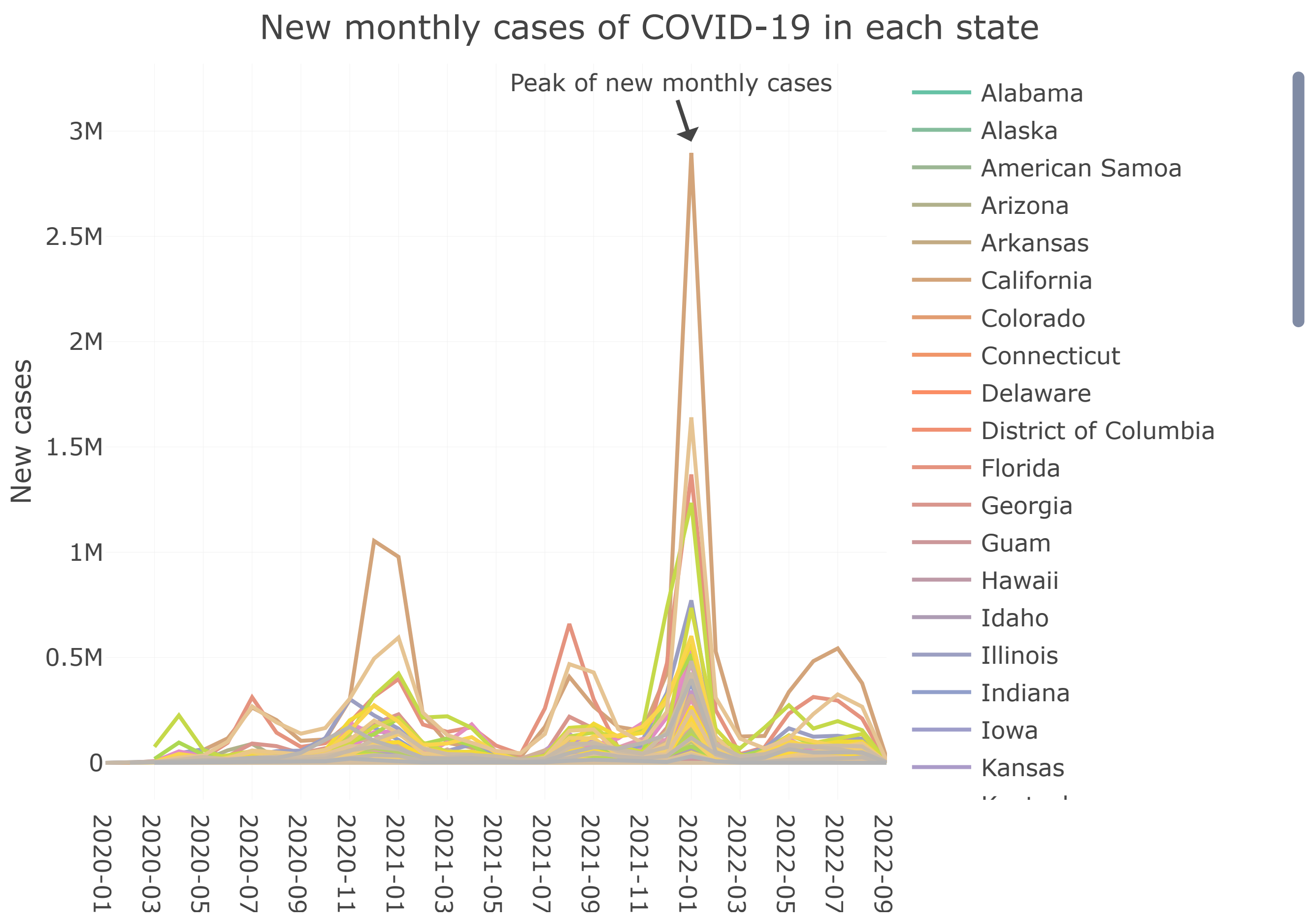
state <fct>	fips <int>	month <chr>	cases_cumulative <int>	deaths_cumulative <int>	new_cases <int>
Alabama	1	2020-03	999	14	993
Alabama	1	2020-04	7068	272	5960
Alabama	1	2020-05	17952	630	10658
Alabama	1	2020-06	38045	950	19511
Alabama	1	2020-07	87723	1580	48761
Alabama	1	2020-08	126058	2182	36709
Alabama	1	2020-09	154701	2540	27085
Alabama	1	2020-10	192285	2967	36541
Alabama	1	2020-11	249524	3578	55539
Alabama	1	2020-12	361226	4827	108326

1-10 of 1,732 rows   Previous **1** 2 3 4 5 6 ... 174 Next

The cases column in the covid_data data frame is cumulative, so we need to compute the number of new cases per month for each state.

```
# Q2.4.Using previous dataframe plot new monthly cases in states, group by states
# The resulting plot is busy, use interactive plotly capabilities to limit number
# of displayed states
# (2 points)

# The resulting plot is busy, use interactive plotly capabilities to limit number of dis
played states
covid_data_monthly_plot <- covid_data_monthly %>%
  plot_ly(x = ~month, y = ~new_cases, color = ~state,
         type = "scatter", mode = "lines") %>%
         layout(title = "New monthly cases of COVID-19 in each state",
         xaxis = list(title = "Month"),
         yaxis = list(title = "New cases"),
         hovermode = "closest")
covid_data_monthly_plot %>%
  config(displayModeBar = F) %>%
  add_annotations(x = "2022-01", y = 2950000, text =
                 "Peak of new monthly cases")
```

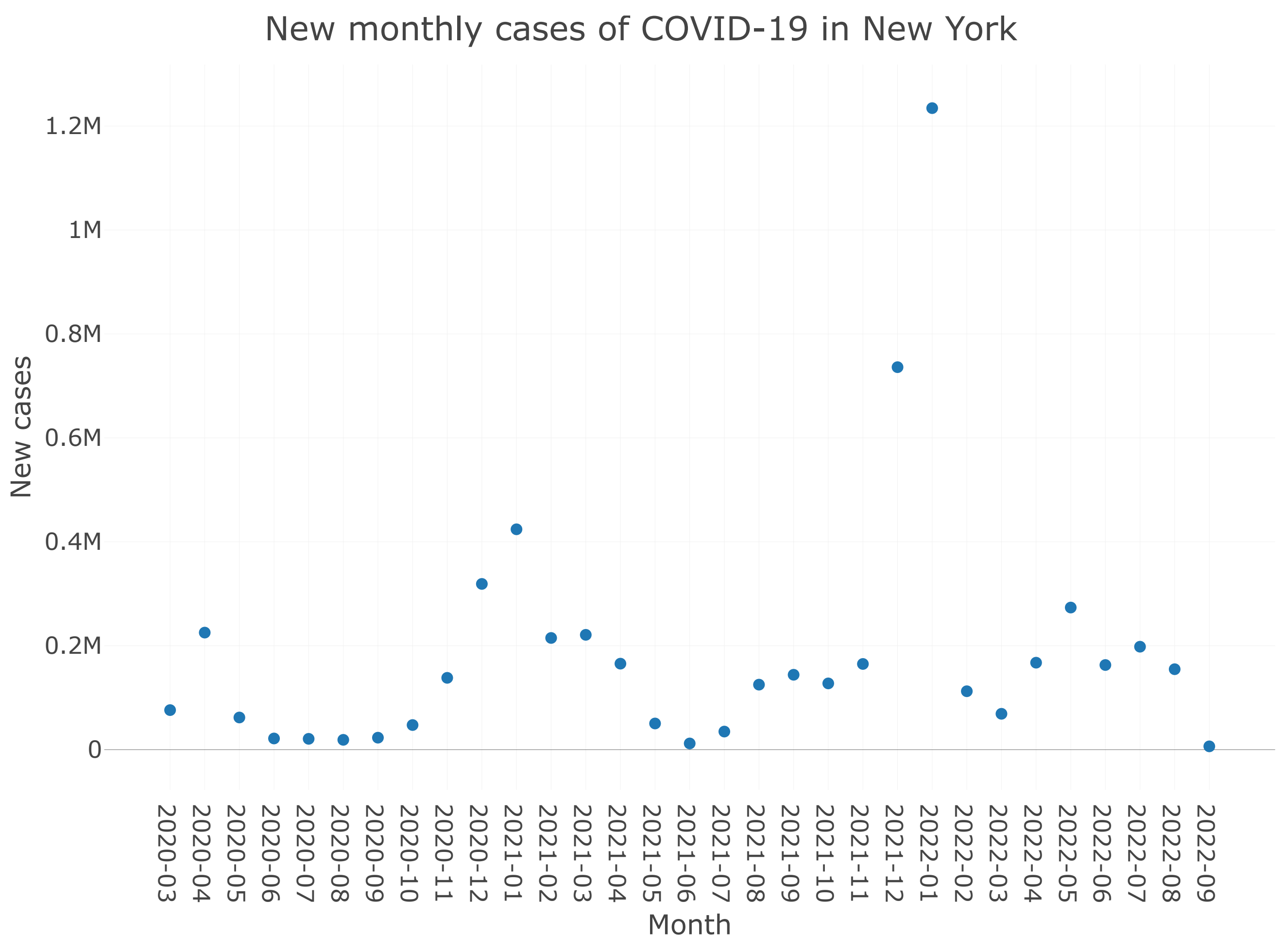
```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for pale
tte Set2 is 8
## Returning the palette you asked for with that many colors

## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for pale
tte Set2 is 8
## Returning the palette you asked for with that many colors
```



Month

```
# annotation to inform user to hover over line to see state name
```

```
# Q2.5.Plot new monthly cases only in NY state
# (2 points)

# Filter the data for the state of New York
ny_covid_data_monthly <- covid_data_monthly %>% filter(state == "New York")

# Create a scatter plot of new monthly cases in NY
ny_covid_data_monthly_plot <- plot_ly(ny_covid_data_monthly, x = ~month,
                              y = ~new_cases, type = "scatter",
                              mode = "markers") %>%
  layout(title = "New monthly cases of COVID-19 in New York",
         xaxis = list(title = "Month"),
         yaxis = list(title = "New cases"),
         hovermode = "closest")

ny_covid_data_monthly_plot
```



```r
# Q2.6. Found the year-month with highest cases in NY state
# (2 points)

# Filter the data for the state of New York
ny_covid_data_monthly <- covid_data_monthly %>% filter(state == "New York")

# Find the row with the highest number of new cases
max_cases_row <- ny_covid_data_monthly %>% slice(which.max(new_cases))

# Display the row with the year-month and number of new cases
max_cases_row
```

state <fct>	fips <int>	month <chr>	cases_cumulative <int>	deaths_cumulative <int>	new_cases <int>
New York	36	2022-01	4789532	64247	1234485

1 row

```r
# Q2.7. Plot new cases in determined above year-month
# using USA state map, color each state by number of cases  (3 points)
# hint:
#   there two build in constants in R: state.abb and state.name
#   to convert full name to abbreviation

# Group the data by state and get the top row for each state with the highest new cases

state_codes <- data.frame(state.name, state.abb)
names(state_codes) <- c("state", "abbr")
covid_data_monthly$abbr <- state_codes[match(covid_data_monthly$state,
                                             state_codes$state), "abbr"]

state_cases <- covid_data_monthly %>%
  group_by(state) %>%
  top_n(1, new_cases) %>%
  ungroup()

state_cases
```

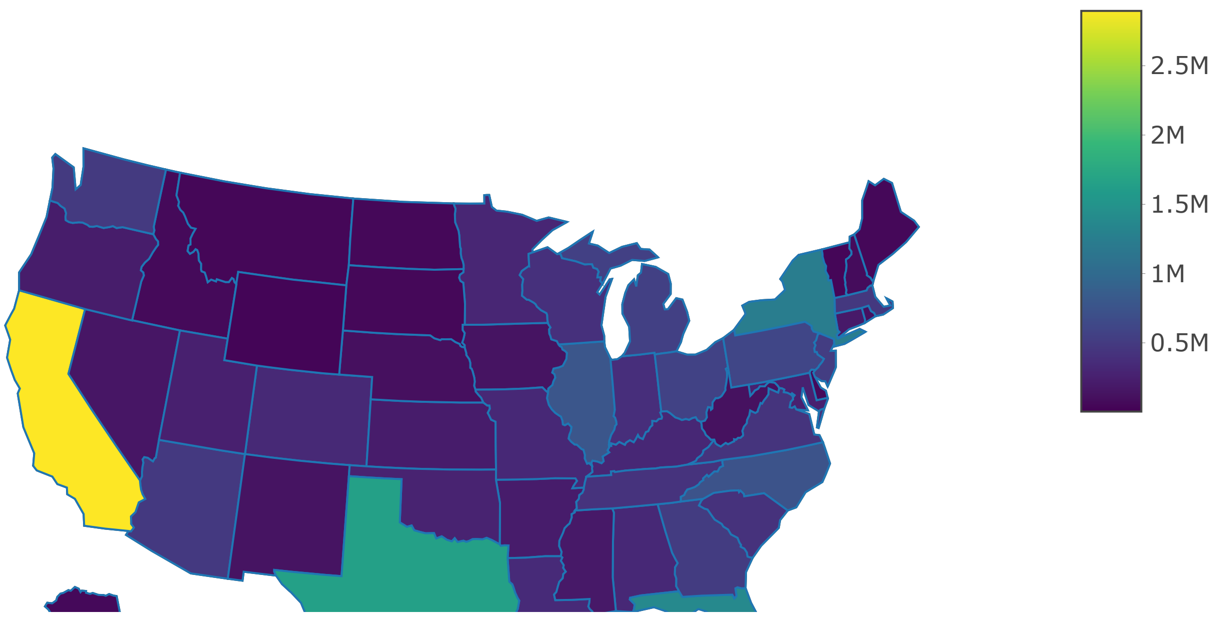
state <fct>	fips <int>	month <chr>	cases_cumulative <int>	deaths_cumulative <int>	new_ca... <int>	a <
Alabama	1	2022-01	1218257	17086	321643	A
Alaska	2	2022-01	215643	1057	59513	A
American Samoa	60	2022-03	3756	0	3662	N
Arizona	4	2022-01	1870644	26206	480936	A
Arkansas	5	2022-01	776759	9634	206118	A

state	fips	month	cases_cumulative	deaths_cumulative	new_ca...
<fct>	<int>	<chr>	<int>	<int>	<int>
California	6	2022-01	8411819	80267	2896206
Colorado	8	2022-01	1258871	11339	325614
Connecticut	9	2022-01	699400	10010	189212
Delaware	10	2022-01	248111	2531	64231
District of Columbia	11	2022-01	130888	1289	36602

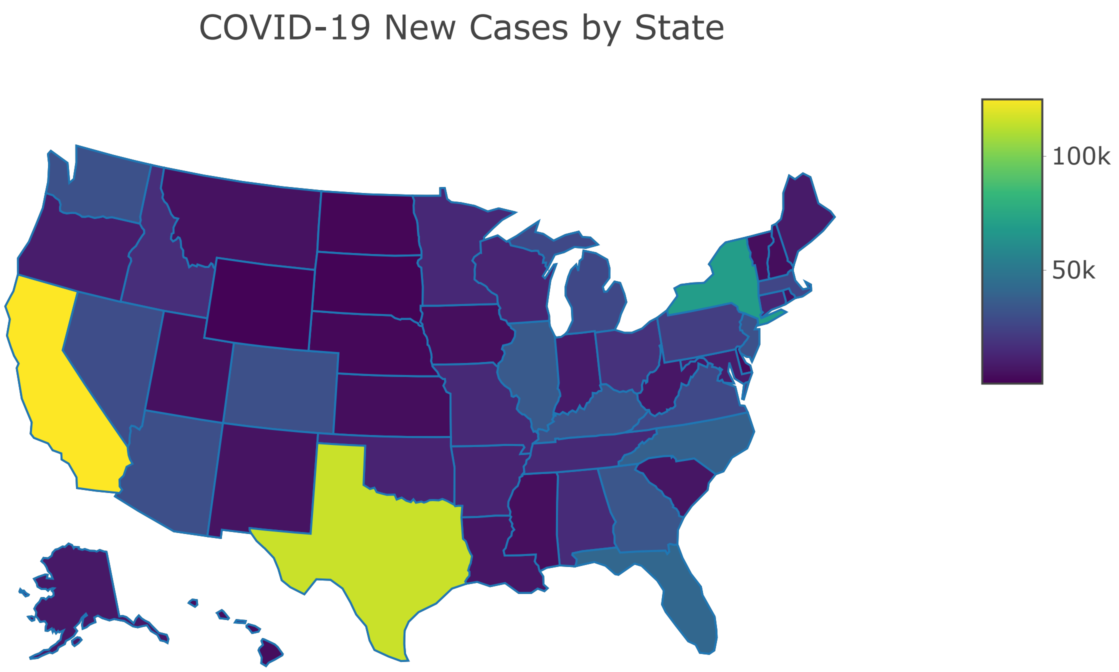
1-10 of 56 rows   Previous **1** 2 3 4 5 6 Next

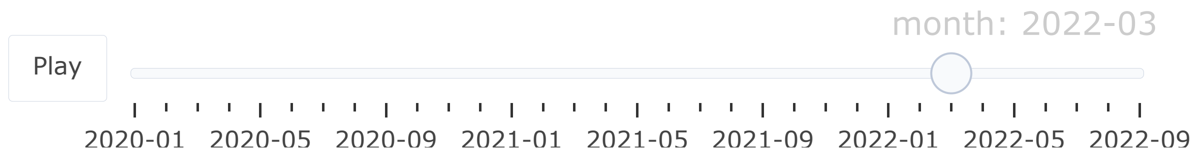
```r
# Create the plotly USA state map
fig <- plot_geo(state_cases, locationmode = "USA-states") %>%
  add_trace(
    z = state_cases$new_cases,
    locations = state_cases$abbr,
    text = paste("State: ", state_cases$state, "<br>",
                 "New cases: ", state_cases$new_cases),
    hoverinfo = "text",
    type = "choropleth",
    showscale = TRUE
  ) %>%
  layout(
    title = "COVID-19 New Cases by State",
    geo = list(scope = "usa",
               projection = list(type = "albers usa")),
    margin = list(l = 0, r = 0, b = 0, t = 40))

# Display the plotly map
fig
```

COVID-19 New Cases by State



```
# Q2.8. Add animation capability (3 points)
# hint:
#     for variable frame you need either integer or character/factorial so
#     convert date to character or factorial

# Create the plotly USA state map
fig <- plot_geo(covid_data_monthly, locationmode = "USA-states" , frame = ~month) %>%
  add_trace(
    z = covid_data_monthly$new_cases,
    locations = covid_data_monthly$abbr,
    text = paste("State: ", covid_data_monthly$state, "<br>",
                 "New cases: ", covid_data_monthly$new_cases),
    hoverinfo = "text",
    type = "choropleth",
    showscale = TRUE
  ) %>%
  layout(
    title = "COVID-19 New Cases by State",
    geo = list(scope = "usa",
               projection = list(type = "albers usa")),
    margin = list(l = 0, r = 0, b = 0, t = 40))

# Display the plotly map
fig
```





Q2.9. Compare animated plot from Q2.8 to plots from Q2.4/Q2.5

The animated plot from Q2.8 is useful for visualizing the temporal evolution of COVID-19 new cases by state over time. It allows us to see how the new cases have changed over the months and how they compare across different states. This type of plot is particularly useful for detecting trends and patterns in the data.On the other hand, the plots from Q2.4/Q2.5 are useful for comparing the COVID-19 new cases across states at a specific point in time (i.e., September 2021). They provide a snapshot of the data and allow us to see how the new cases are distributed geographically. This type of plot is particularly useful for comparing the relative new case counts across different states.

Therefore, the choice between the two types of plots depends on the specific question being asked and the goal of the analysis. If the goal is to analyze the temporal evolution of new cases over time, the animated plot from Q2.8 would be preferred. If the goal is to compare new cases across states at a specific point in time, the plots from Q2.4/Q2.5 would be preferred.