Homework 1. R and Interactive Visualization (60 points)

Shri Harsha

2023-09-11

```
library(tidyverse)
library(plotly)
library(dplyr)
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. (26 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)

library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

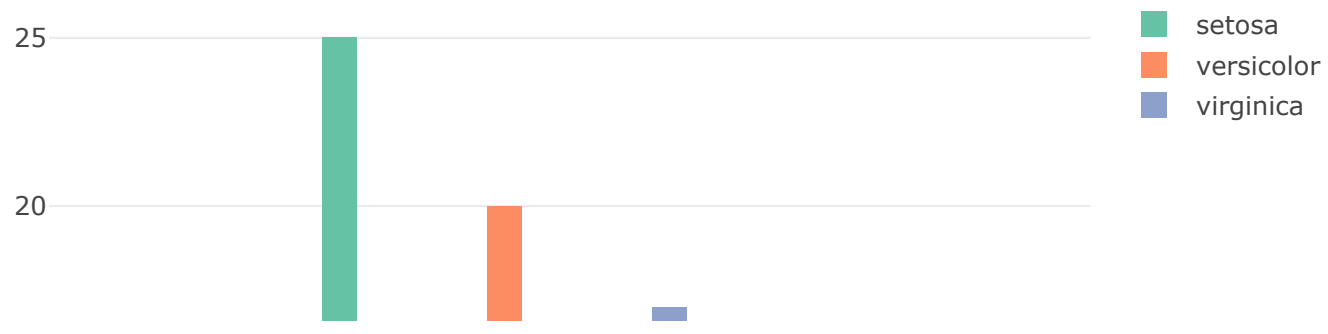
```
# Read iris.csv using fread with stringsAsFactors set to TRUE
iris_data <- fread("iris.csv", stringsAsFactors = TRUE)
```

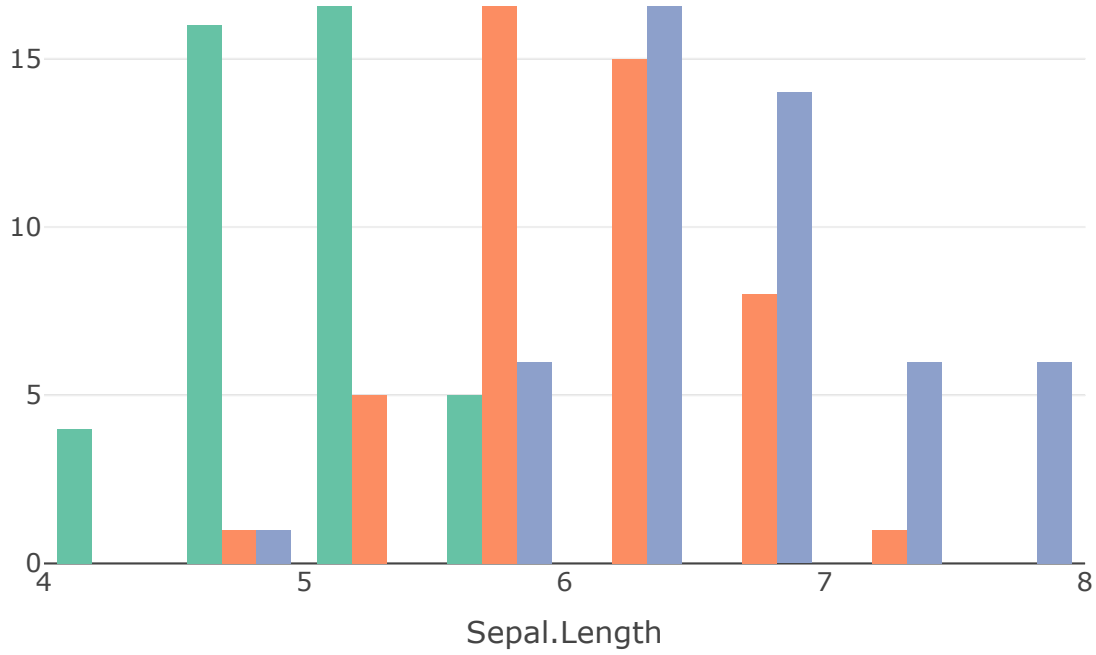
```
# Q1.2. Show some values from data frame (2 points)
head(iris_data)
```

Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

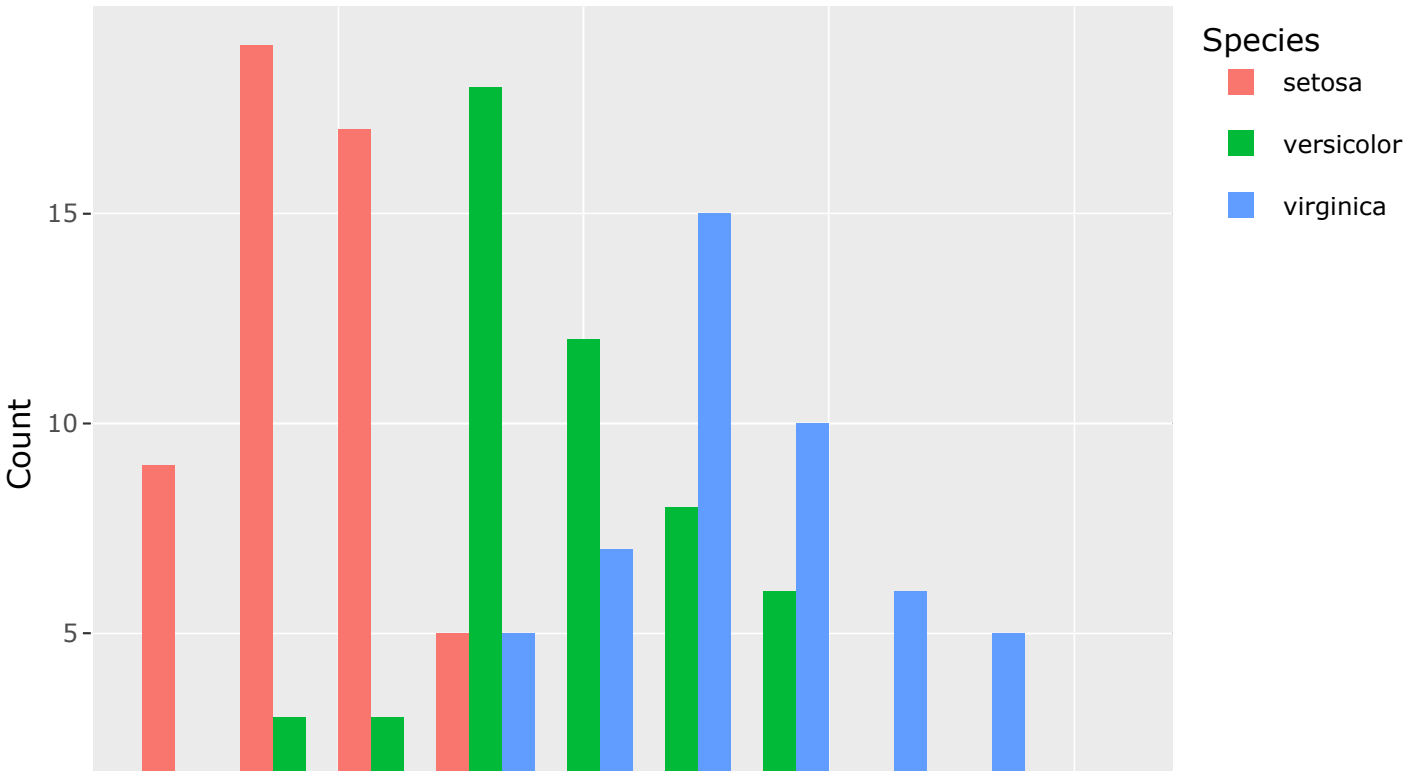
6 rows

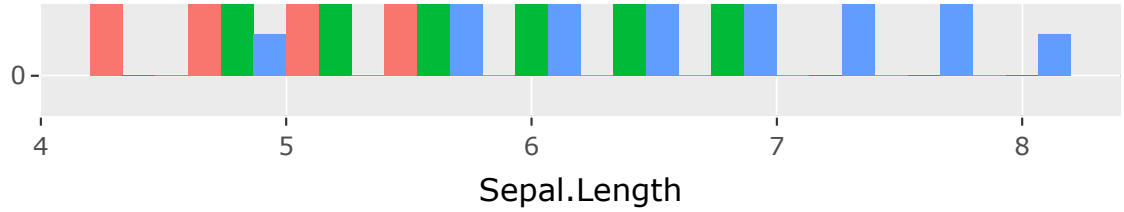
```
# Q1.3. Build histogram plot for Sepal.Length variable for each species using plot_ly
# (use color argument for grouping) (2 points)
# should be one plot
# Create a histogram plot for Sepal.Length by Species

plot_ly(iris_data, x = ~Sepal.Length, color = ~Species, type="histogram")
```



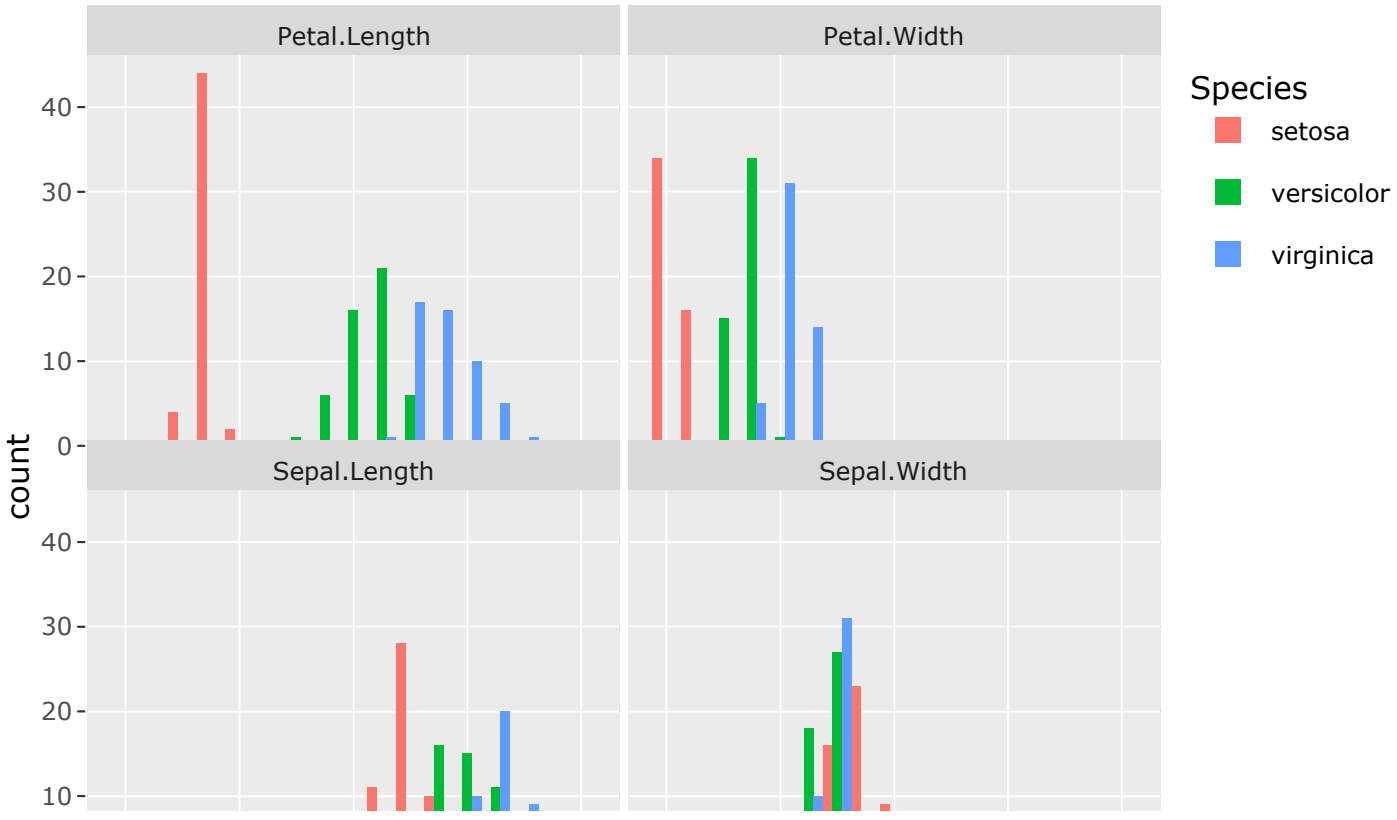


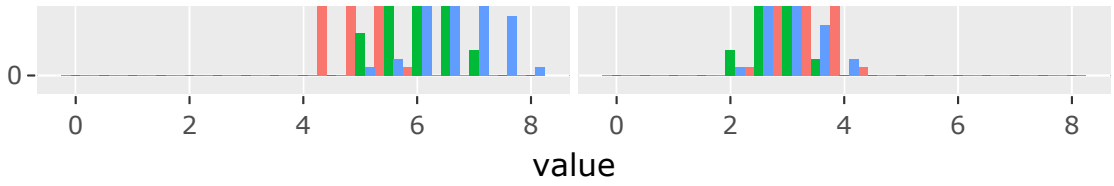

```
# Q1.4. Repeat previous plot with ggplot2 and convert it to plotly with ggplotly (3 p
oints)
# Create a histogram plot for Sepal.Length by Species using ggplot2
histplot <- ggplot(iris_data, aes(x = Sepal.Length, fill = Species)) +
  geom_histogram(binwidth = 0.4, position = "dodge") +
  labs(x = "Sepal.Length", y = "Count") +
  scale_fill_discrete(name = "Species")

interactive_plot <- ggplotly(histplot)
interactive_plot
```






```
# Q1.5. Create facet 2 by 2 plot with histograms similar to previous but for each met
ric
# (3 points)
# hint:
#   following conversion to long format can be useful:
#   iris %>% pivot_longer(...)
#

iris_long <- iris_data %>%
  pivot_longer(
    cols = c(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width),
    names_to = "metric",
    values_to = "value"
  )

facetplot <- ggplot(iris_long, aes(x = value, fill = Species)) +
  geom_histogram(binwidth = 0.5, position = "dodge") +
  facet_wrap(~metric, nrow = 2)

interactive_plot <- ggplotly(facetplot)
interactive_plot
```



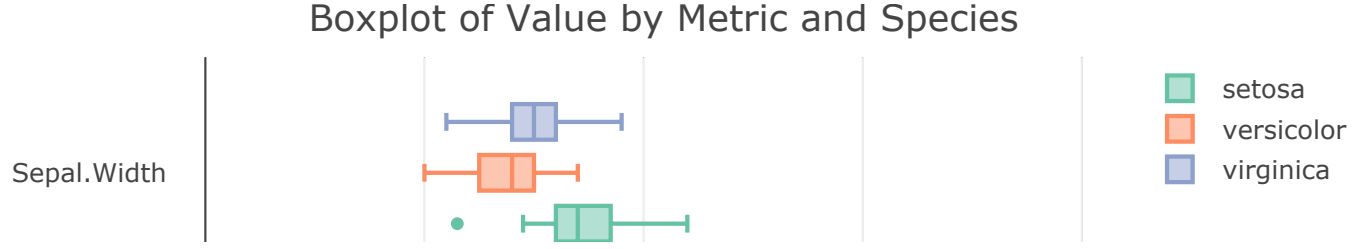


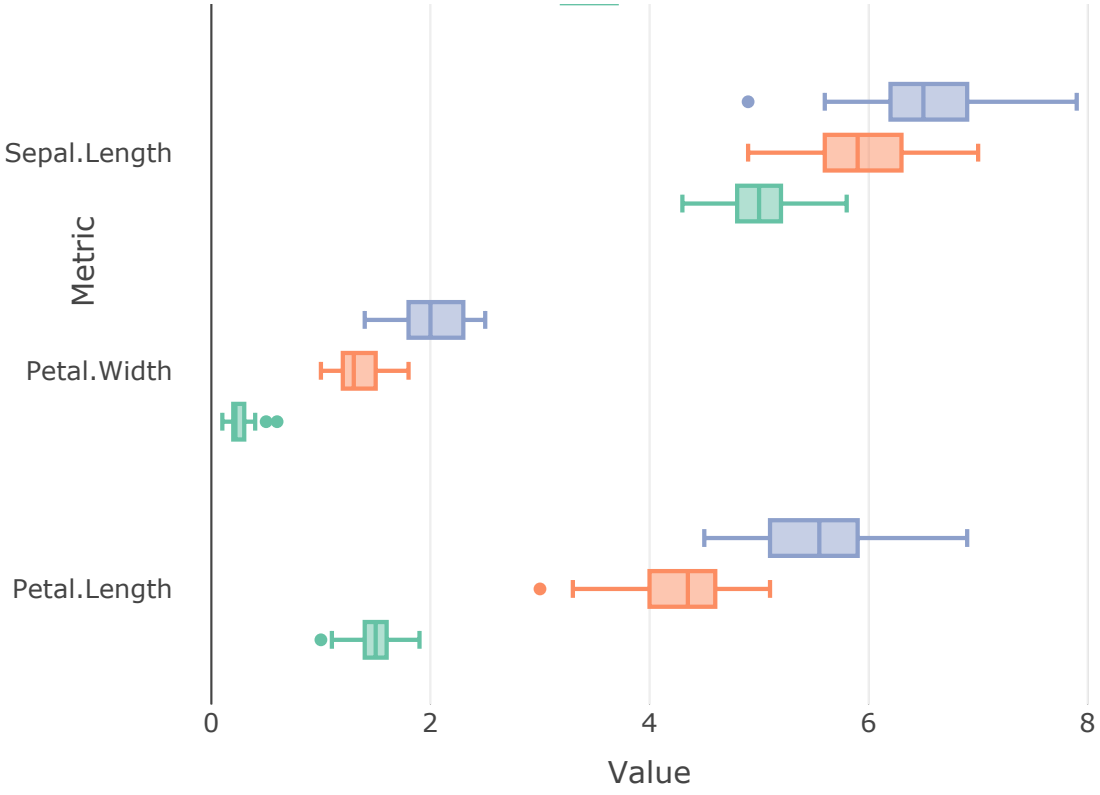
Q1.6. Which metrics has best species separations? (3 points)

Petal Length metric is the best metric to seperate species.

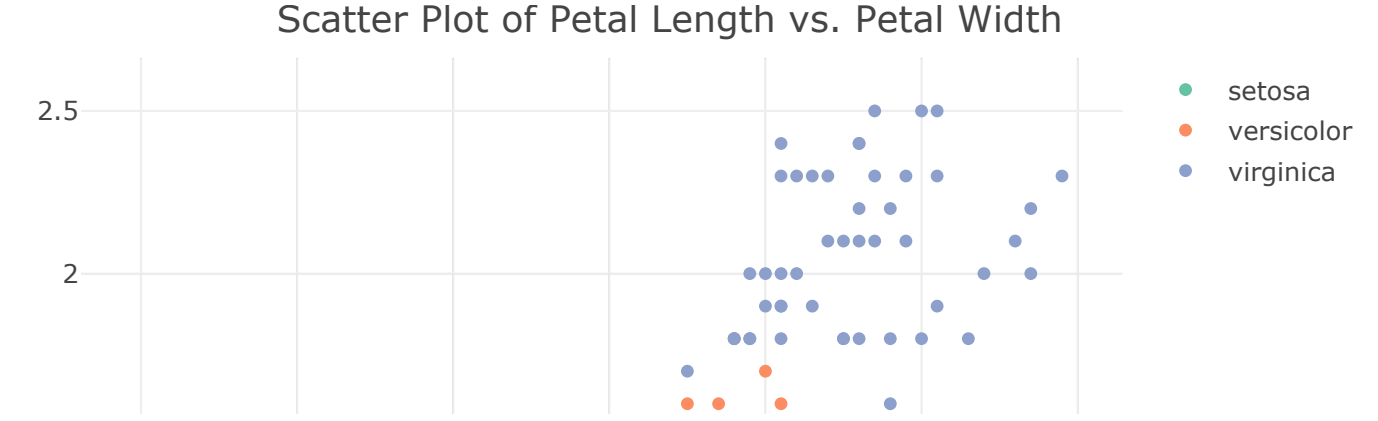
```
# Q1.7. Repeat above plot but using box plot (3 points)

#ggplot(iris_long, aes(x = value, y = metric, fill = Species)) +
#  geom_boxplot()



boxplot_plot <- plot_ly(iris_long, x = ~value, y = ~metric, type = 'box', color = ~Sp
ecies)

boxplot_plot <- boxplot_plot %>% layout(
  title = "Boxplot of Value by Metric and Species",
  xaxis = list(title = "Value"),
  yaxis = list(title = "Metric"),
  boxmode = "group"
)



interactive_plot <- ggplotly(boxplot_plot)
interactive_plot
```

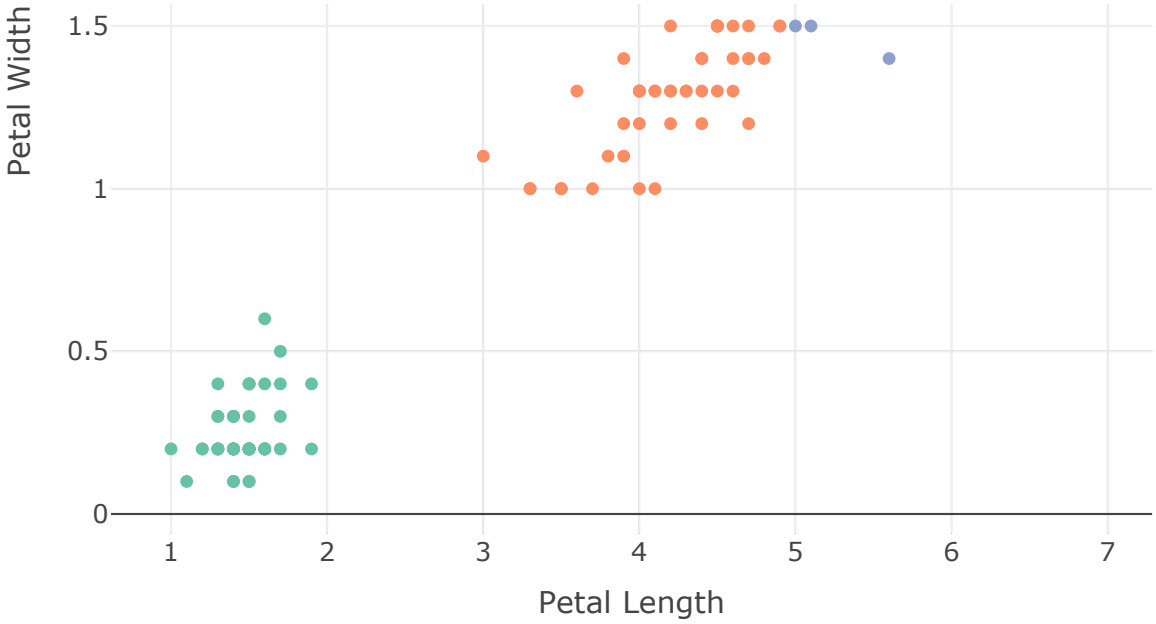
```
## Warning: 'layout' objects don't have these attributes: 'boxmode'
## Valid attributes include:
## '_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calen
dar', 'clickmode', 'coloraxis', 'colorscale', 'colorway', 'computed', 'datarevision',
'dragmode', 'editrevision', 'editType', 'font', 'geo', 'grid', 'height', 'hidesources
', 'hoverdistance', 'hoverlabel', 'hovermode', 'images', 'legend', 'mapbox', 'margin'
, 'meta', 'metasrc', 'modebar', 'newshape', 'paper_bgcolor', 'plot_bgcolor', 'polar',
'scene', 'selectdirection', 'selectionrevision', 'separators', 'shapes', 'showlegend'
, 'sliders', 'smith', 'spikedistance', 'template', 'ternary', 'title', 'transition',
'uirevision', 'uniformtext', 'updatemenus', 'width', 'xaxis', 'yaxis', 'barmode', 'ba
rgap', 'mapType'
```



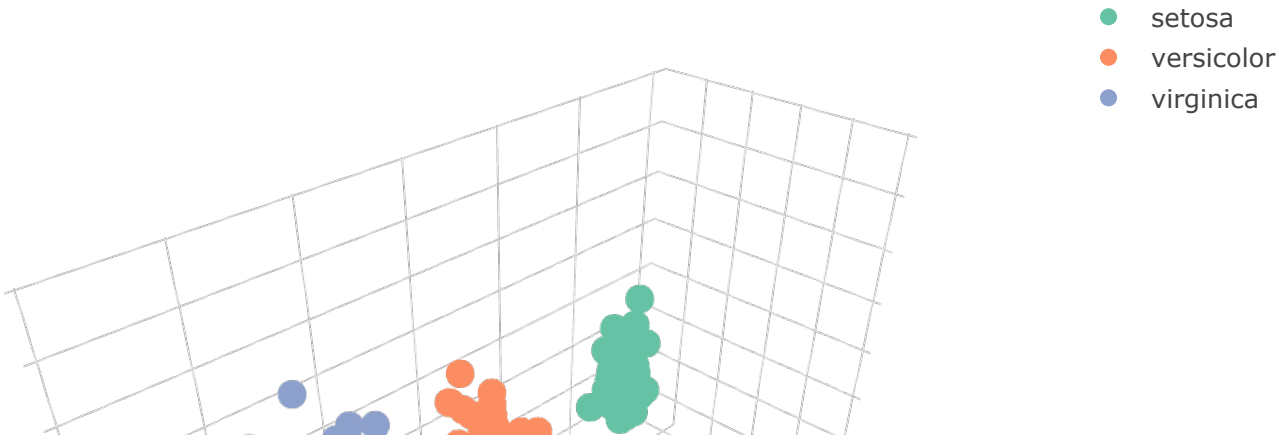


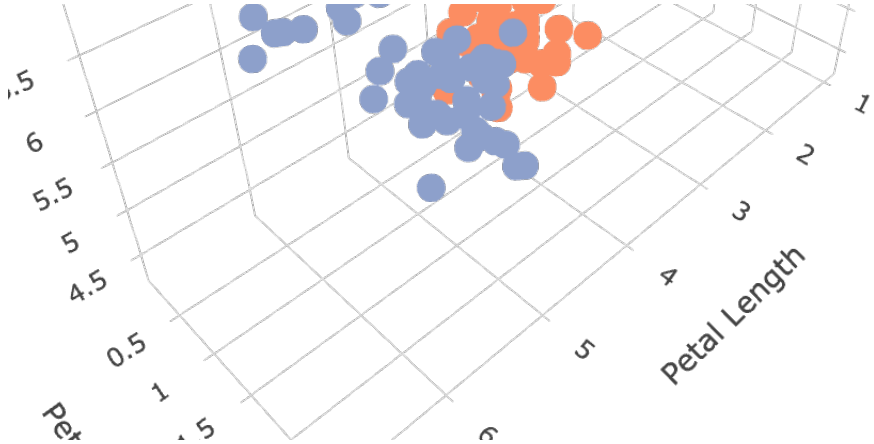

```
# Q1.8. Choose two metrics which separates species the most and use it to make scatte
r plot
# color points by species (3 points)



scatter_plot <- iris_data %>%
  plot_ly(x = ~Petal.Length, y = ~Petal.Width, color = ~Species, type = 'scatter', mo
de = 'markers') %>%
  layout(
    title = "Scatter Plot of Petal Length vs. Petal Width",
    xaxis = list(title = "Petal Length"),
    yaxis = list(title = "Petal Width")
  )


scatter_plot
```






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


scatter3d_plot <- plot_ly(iris, x = ~Petal.Length, y = ~Petal.Width, z = ~Sepal.Lengt
h, color = ~Species, type = 'scatter3d', mode = 'markers')


scatter3d_plot <- scatter3d_plot %>% layout(
  title = "3D Scatter Plot of Petal Length vs. Petal Width vs. Sepal Length",
  scene = list(
    xaxis = list(title = "Petal Length"),
    yaxis = list(title = "Petal Width"),
    zaxis = list(title = "Sepal Length")
  )
)

scatter3d_plot
```





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

Upon careful observation of the 3D plot, it becomes evident that Setosa stands out distinctly in comparison to the other two species. The separation is quite apparent and clearly defined.

While there is some overlap between the Versicolor and Virginica species, a noticeable distinction between them remains. Typically, Versicolor falls within an intermediate range in terms of 'Petal.Length,' 'Petal.Width,' and 'Sepal.Length,' while Virginica tends to exhibit relatively larger values for 'Petal.Length' and 'Petal.Width.' Meanwhile, Setosa is characterized by shorter 'Petal.Length' and 'Petal.Width' compared to the other species.

Part 2. Covid-19 Dataset. (34 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 (3 points)

us_states <- fread("us-states.csv")
```

```
# Q2.2. Show some values from dataframe (3 points)
head(us_states)
```

date	state	fips	cases	deaths
<IDate>	<chr>	<int>	<int>	<int>
2020-01-21	Washington	53	1	0
2020-01-22	Washington	53	1	0
2020-01-23	Washington	53	1	0
2020-01-24	Illinois	17	1	0

	2020-01-24	Washington	53	1	0
	2020-01-25	California	6	1	0

6 rows

```
# Q2.3. Create new dataframe with new cases per month for each state (5 points)
# hint:
#    is cases column cumulative or not cumulative?

# Extract date and state columns and create a month column
us_states_per_month <- us_states %>%
  mutate(date = as.Date(date),
         month = month(date),
         year = year(date),
         yearmonth=floor_date(date,unit="month"))

# Calculate new cases per month for each state
us_states_per_month <- us_states_per_month %>%
  group_by(state,yearmonth) %>%
  summarise(cum_new_cases = max(cases)) %>%
  mutate(new_cases=cum_new_cases-lag(cum_new_cases,default=0))
```

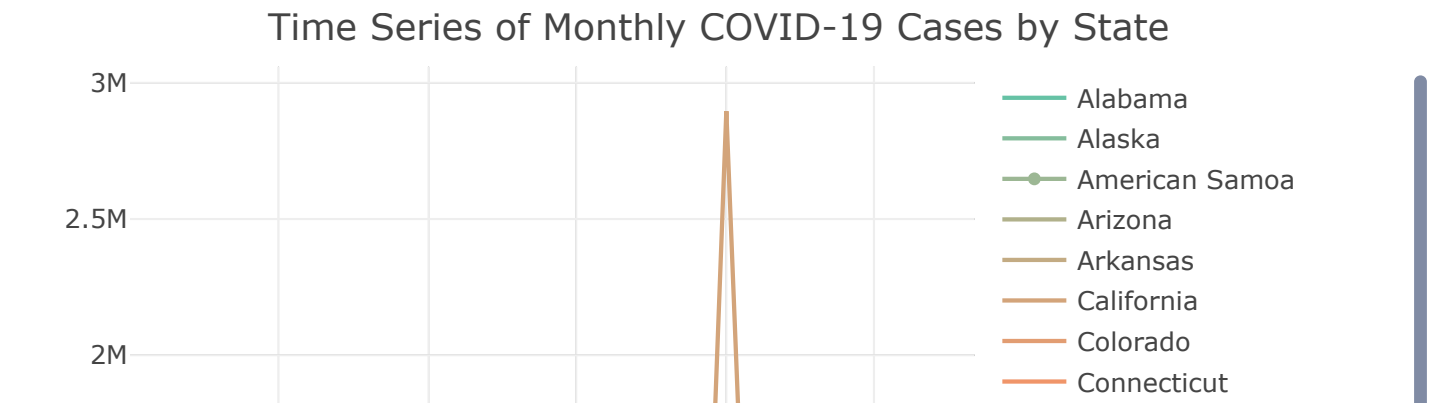
```
## `summarise()` has grouped output by 'state'. You can override using the
## `.groups` argument.
```

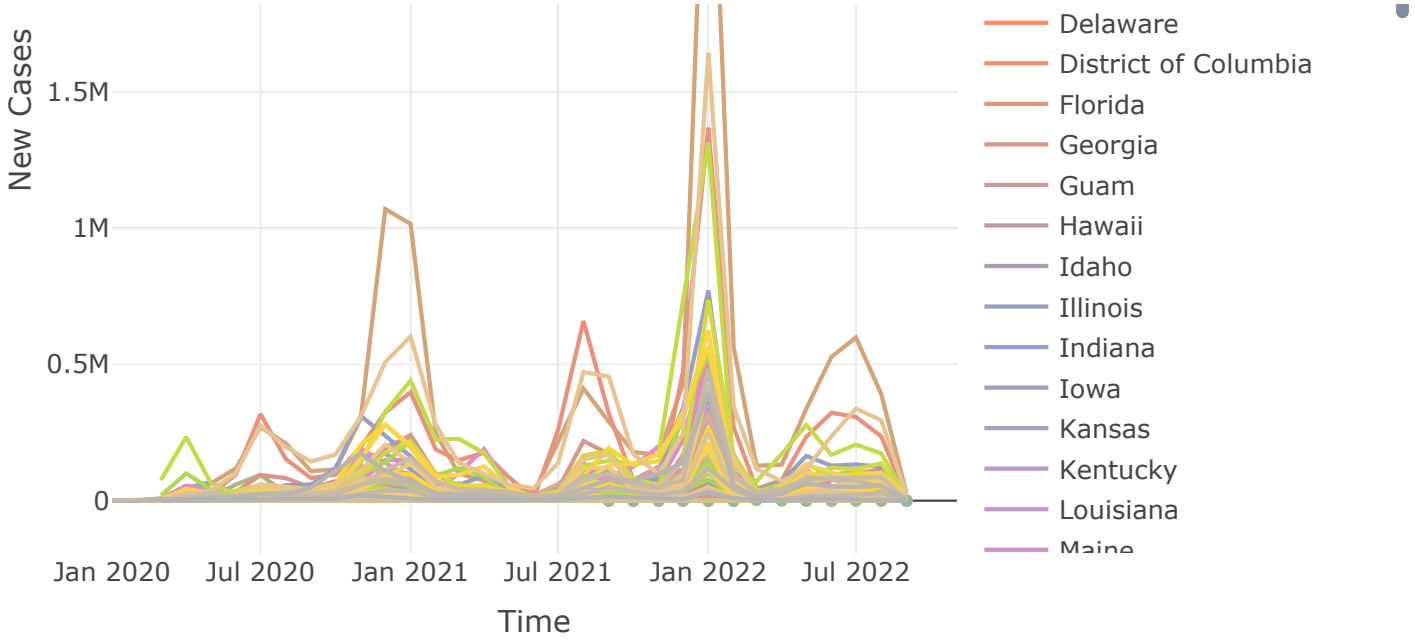
```
us_states_per_month
```

state <chr>	**yearmonth** <date>	**cum_new_cases** <int>	**new_cases** <int>
Alabama	2020-03-01	999	999
Alabama	2020-04-01	7068	6069
Alabama	2020-05-01	17952	10884
Alabama	2020-06-01	38045	20093
Alabama	2020-07-01	87723	49678
Alabama	2020-08-01	126058	38335
Alabama	2020-09-01	154701	28643
Alabama	2020-10-01	192285	37584
Alabama	2020-11-01	249524	57239
Alabama	2020-12-01	361226	111702

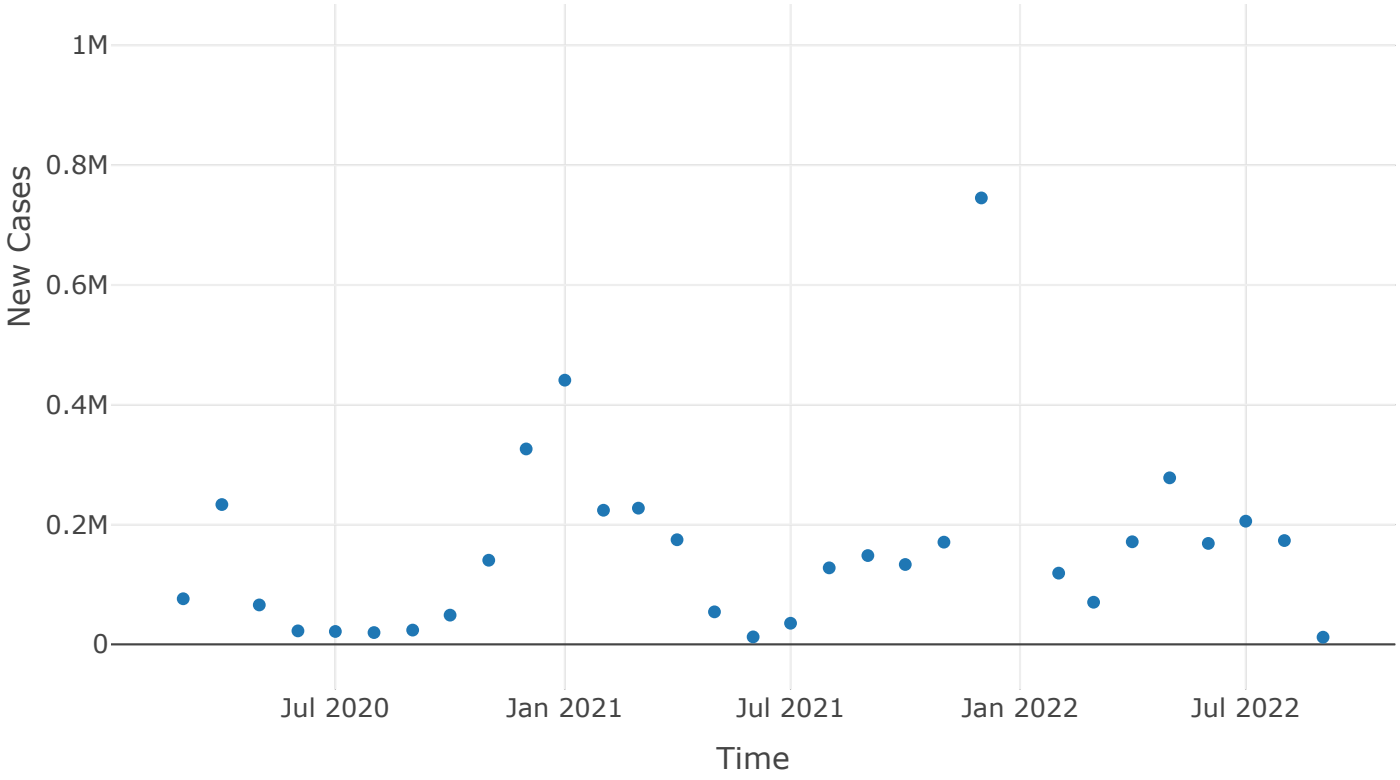
```
# Q2.4.Using previous dataframe plot new monthly cases in states, group by states
# The resulting plot is busy, use interactive plotly capabilities to limit number
# of displayed states
# (4 points)


# Convert 'month' and 'year' columns to a Date object
#us_states_per_month$time <- as.Date(paste(us_states_per_month$year, us_states_per_mo
nth$month, "1", sep = "-"))

# Create an interactive Plotly time series plot
plot <- us_states_per_month %>%
  plot_ly(
    x = ~yearmonth,
    y = ~new_cases,
    color = ~state,
    type = 'scatter',
    mode = 'path'
  ) %>%
  layout(
    title = "Time Series of Monthly COVID-19 Cases by State",
    xaxis = list(title = "Time"),
    yaxis = list(title = "New Cases"),
    showlegend = TRUE
  )

# Display the interactive plot
plot
```

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






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


ny_data <- us_states_per_month %>%
  filter(state == "New York")


# Create an interactive Plotly time series scatter plot for New York
plot <- ny_data %>%
  plot_ly(
    x = ~yearmonth,
    y = ~new_cases,
    type = 'scatter',
    mode = 'markers'
  ) %>%
  layout(
    title = "Time Series of Monthly COVID-19 Cases in New York",
    xaxis = list(title = "Time"),
    yaxis = list(title = "New Cases"),
    showlegend = FALSE
  )

# Display the interactive plot
plot
```

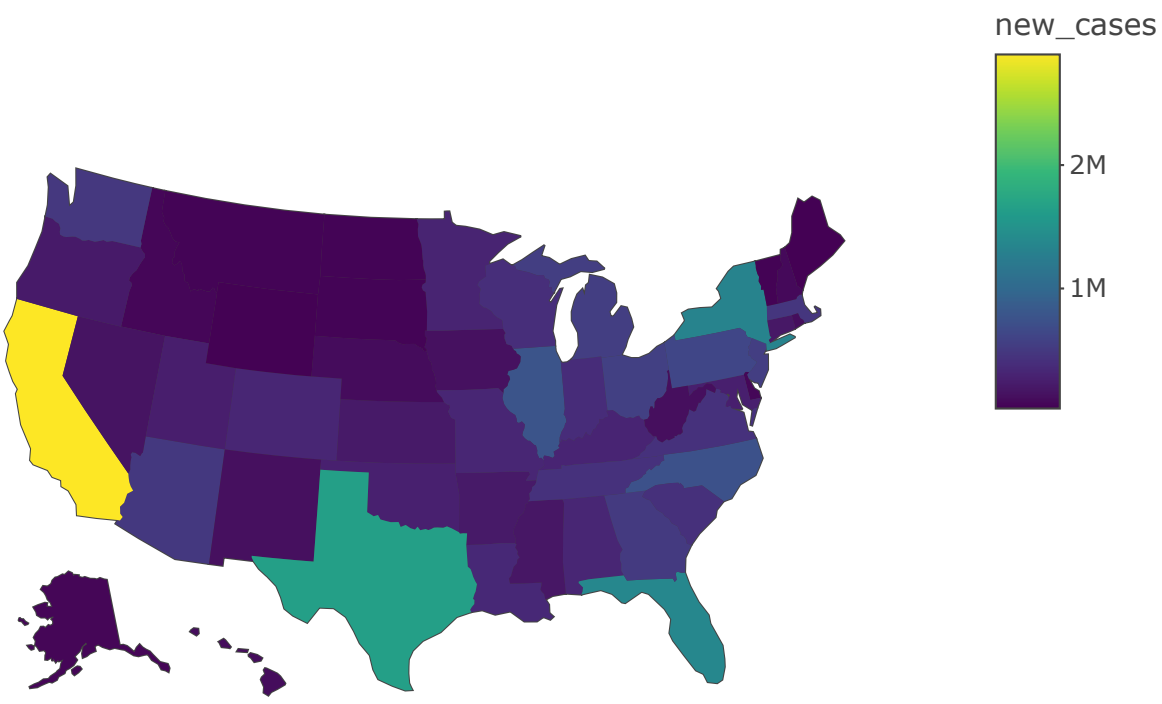



```
# Q2.6. Found the year-month with highest cases in NY state
# (3 points)

ny_data[which.max(ny_data$new_cases), ]
```

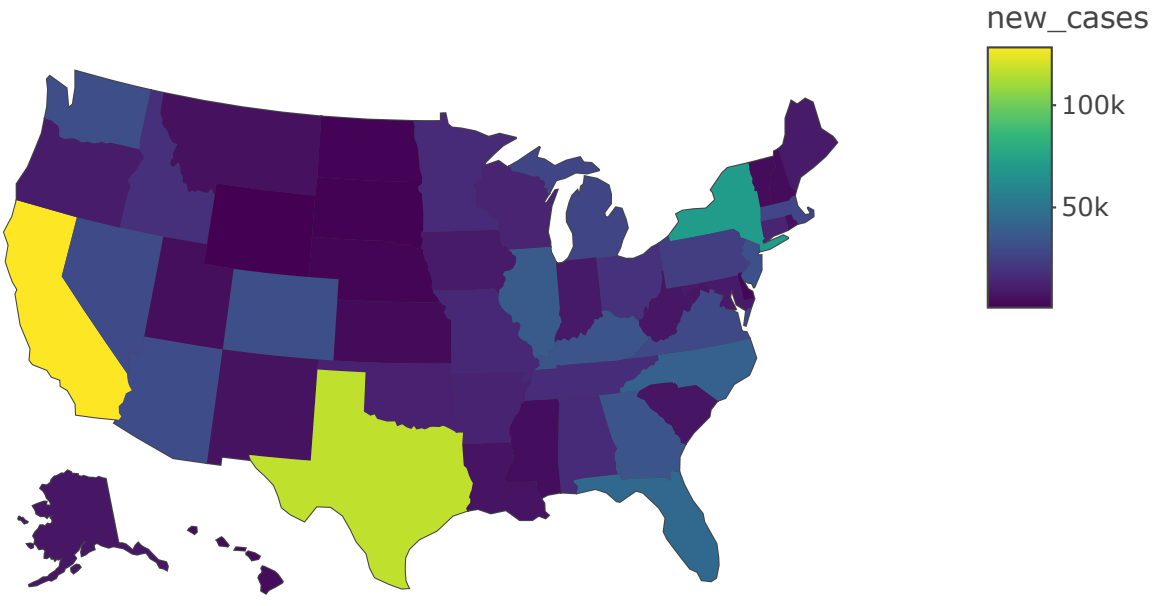
state	yearmonth	cum_new_cases	new_cases
<chr>	<date>	<int>	<int>
New York	2022-01-01	4789532	1315562

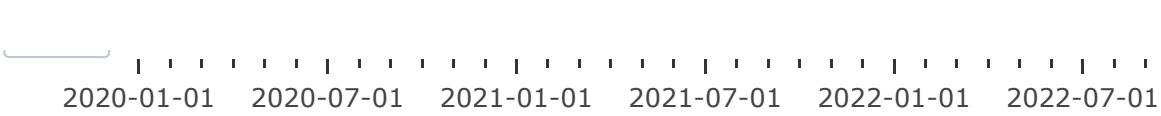
1 row

```r
# Q2.7. Plot new cases in determined above year-month
# using USA state map, color each state by number of cases  (5 points)
# hint:
#   there two build in constants in R: state.abb and state.name
#   to convert full name to abbreviation

# Create a map using plot_geo

states_df <- tibble(state=state.name, abb=state.abb)
us_states_for_a_day <- us_states_per_month %>%
  filter(yearmonth=="2022-01-01" & state %in% state.name) %>%
  left_join(states_df,by="state")

g <- list(
  scope = 'usa',
  projection = list(type = 'albers usa'),
  lakecolor = toRGB('white')
)

plot_geo(us_states_for_a_day) %>%
  add_trace(
    z = ~new_cases, text = ~state, span = I(0),
    locations = ~abb, locationmode = 'USA-states'
  ) %>%
  layout(geo = g)
```



```
# Q2.8. Add animation capability (5 points)
# hint:
#     for variable frame you need either integer or character/factorial so
#     convert date to character or factorial

us_states_per_month_filtered <- us_states_per_month %>%
  filter(state %in% state.name) %>%
  left_join(states_df,by="state")

g <- list(
  scope = 'usa',
  projection = list(type = 'albers usa'),
  lakecolor = toRGB('white')
)

plot_geo(us_states_per_month_filtered) %>%
  add_trace(
    z = ~new_cases, text = ~state, span = I(0),frame=~factor(yearmonth),
    locations = ~abb, locationmode = 'USA-states'
  ) %>%
  layout(geo = g)
```





Q2.9. Compare animated plot from Q2.8 to plots from Q2.4/Q2.5 (When you would prefer one or another?) (3 points)

I prefer Maps because they are useful for COVID-19 cases because they show where the virus is spreading. They help us see which areas have more infections and where the problem is worst. Maps can also show how the virus changes over time. This helps us make smart choices about safety rules, travel, and resources. In short, maps are a great tool for understanding and dealing with COVID-19 because they show us where it is and how it’s spreading.