Homework 1

Nisarg Negi

2022-09-20

```
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)
df <- fread("iris.csv",stringsAsFactors=TRUE)
```

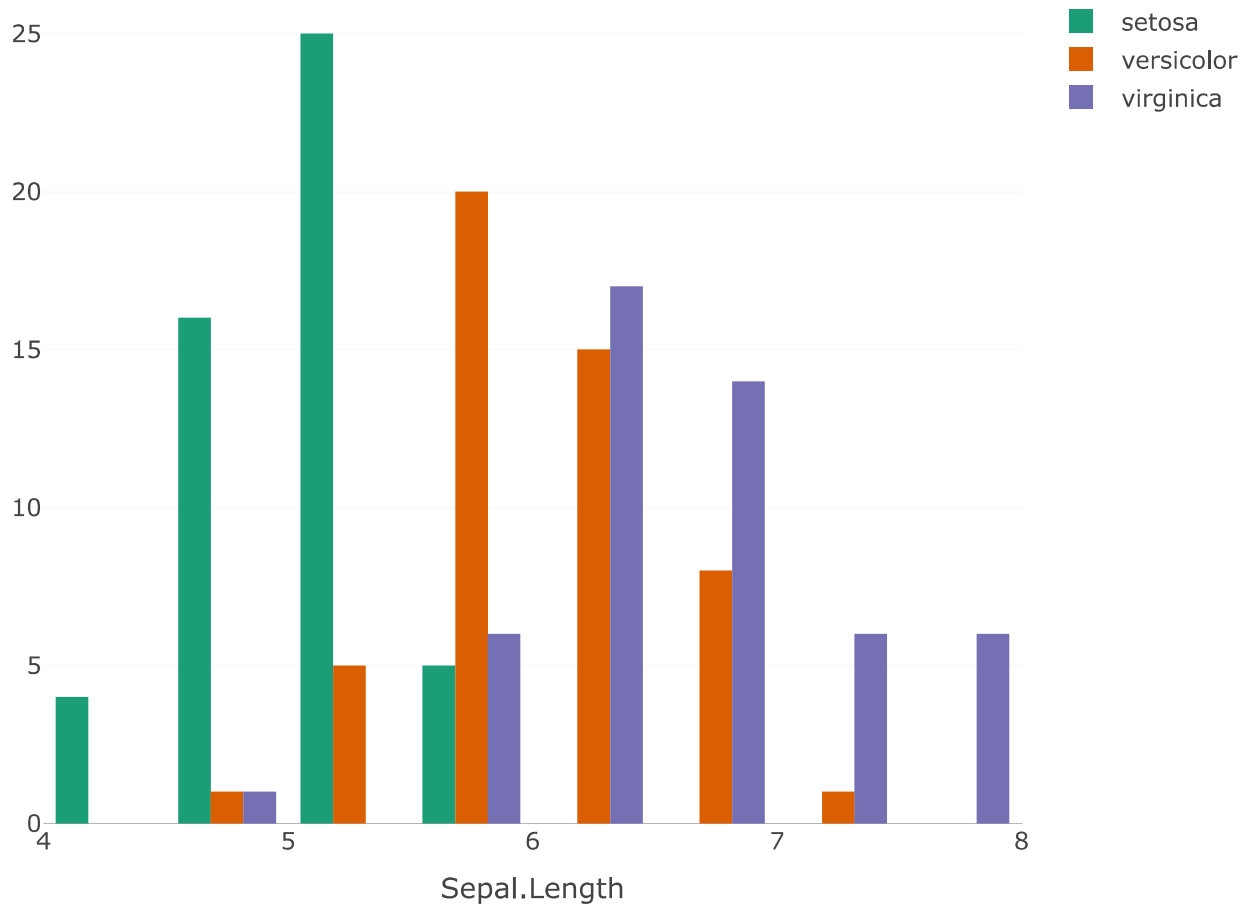
```
# Q1.2. Show some values from data frame (2 points)
head(df,6)
```

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

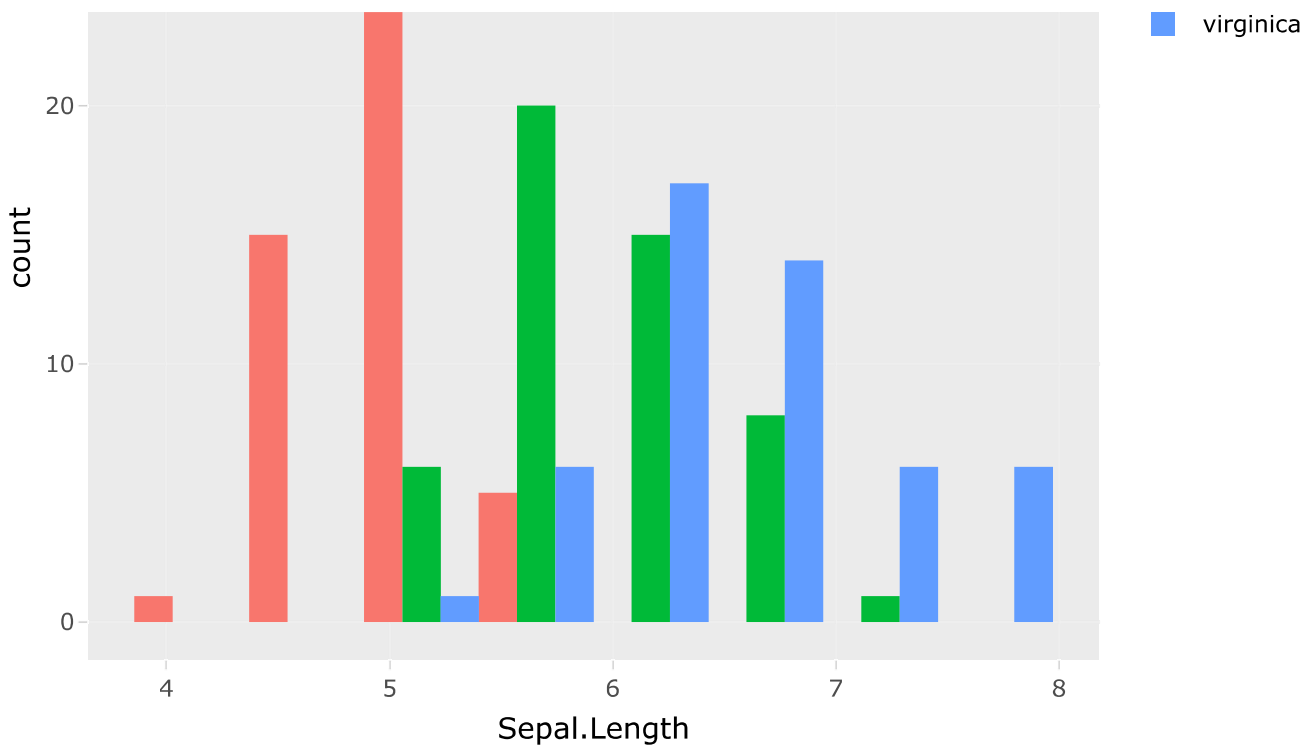
Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

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
plot_ly(df, x = ~Sepal.Length, color = ~Species, colors = "Dark2", type="histogram")
```




```
# Q1.4. Repeat previous plot with ggplot2 and convert it to plotly with ggplotly (2 points)
p <- ggplot(df, aes(x = Sepal.Length)) + geom_histogram(bins = 8,position = "dodge",aes(fill = S
pecies))
ggplotly(p)
```

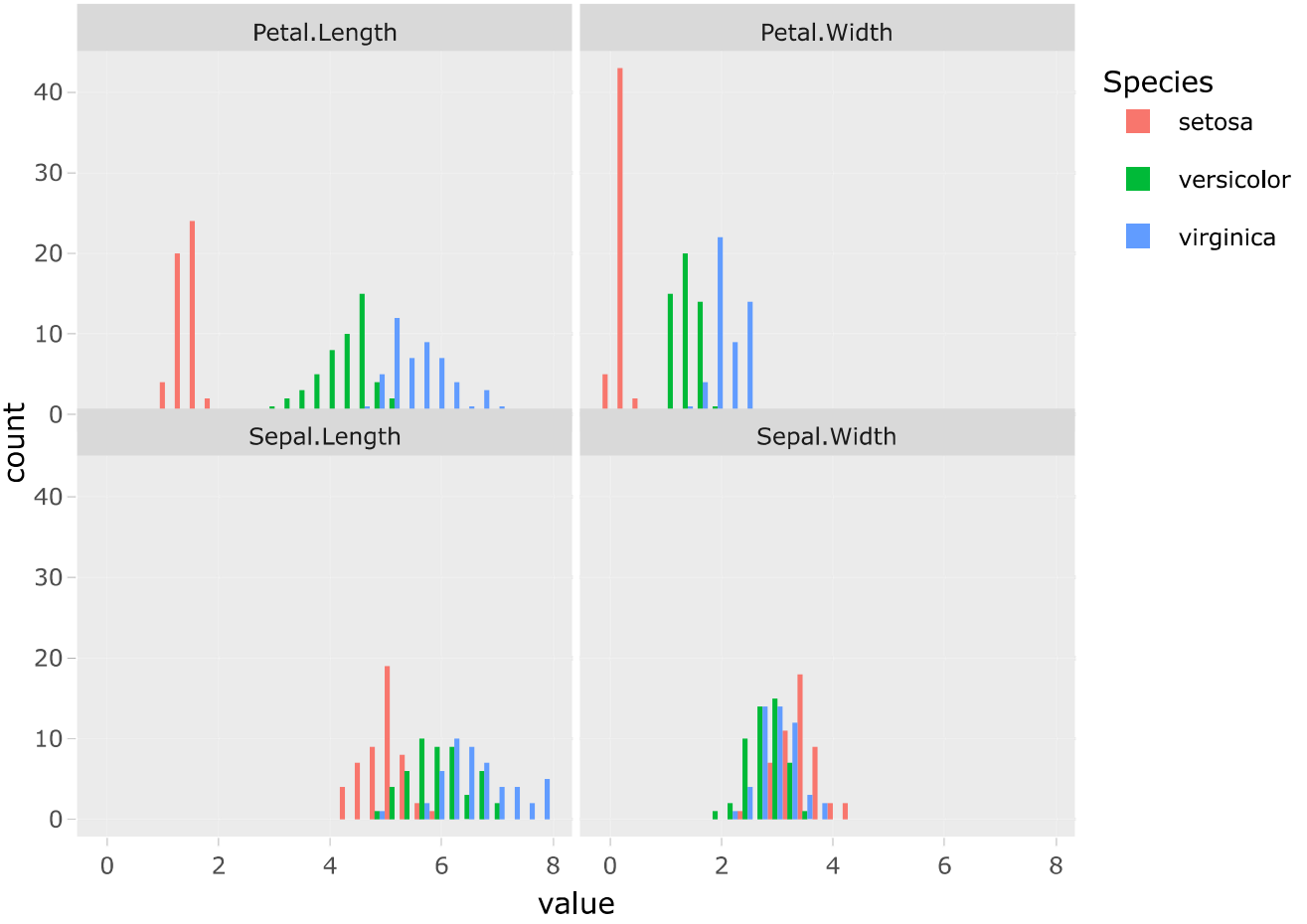



```
# Q1.5. Create facet 2 by 2 plot with histograms similar to previous but for each metric
# (2 points)
# hint:
#   following conversion to long format can be useful:
#   iris %>% gather(key = "metric", value = "value",-Species)
#

iris_gathered <- gather(df, metric, value, -Species)
iris_gathered
```

Species <fct>	**metric** <chr>	**value** <dbl>
setosa	Sepal.Length	5.1
setosa	Sepal.Length	4.9
setosa	Sepal.Length	4.7
setosa	Sepal.Length	4.6
setosa	Sepal.Length	5.0
setosa	Sepal.Length	5.4
setosa	Sepal.Length	4.6
setosa	Sepal.Length	5.0
setosa	Sepal.Length	4.4
setosa	Sepal.Length	4.9

```
iris_facet <- ggplot(iris_gathered, aes(value, fill = Species))+ geom_histogram(position = "dodg
e") +facet_wrap(~ metric)
ggplotly(iris_facet)
```

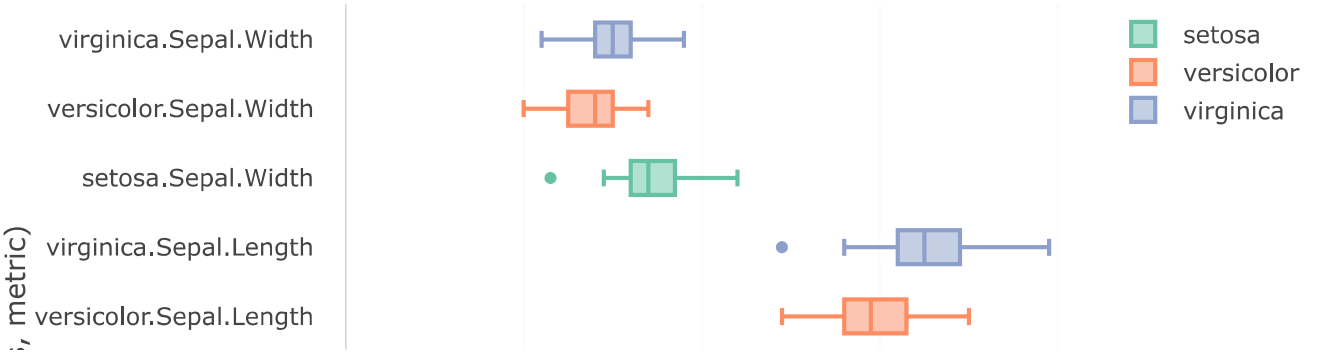
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

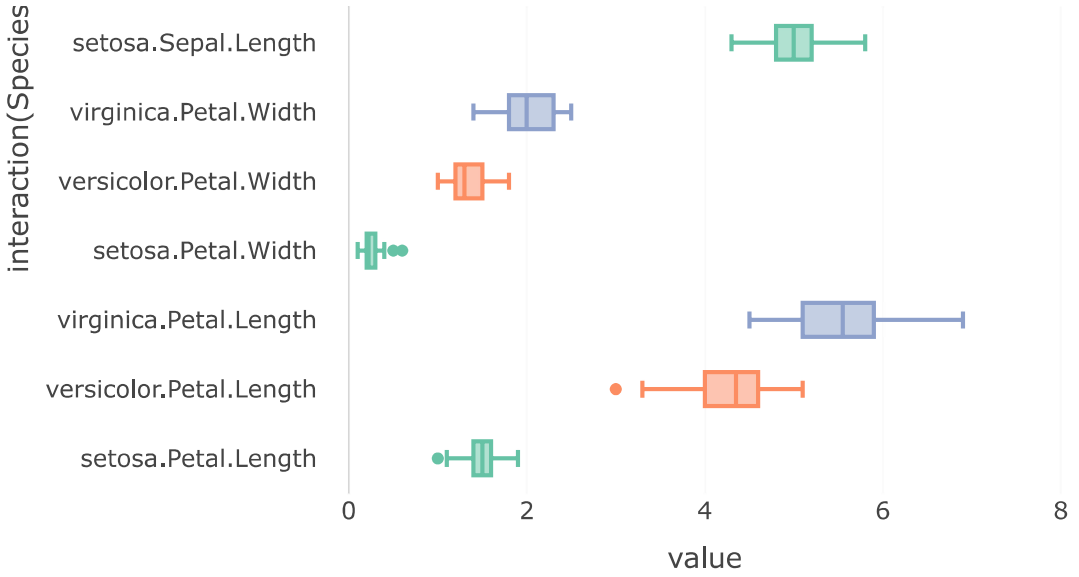


Q1.6. Which metrics has best species separations? (2 points)

Petal Length and Petal Width are the best for species separation as they segregate the data best.

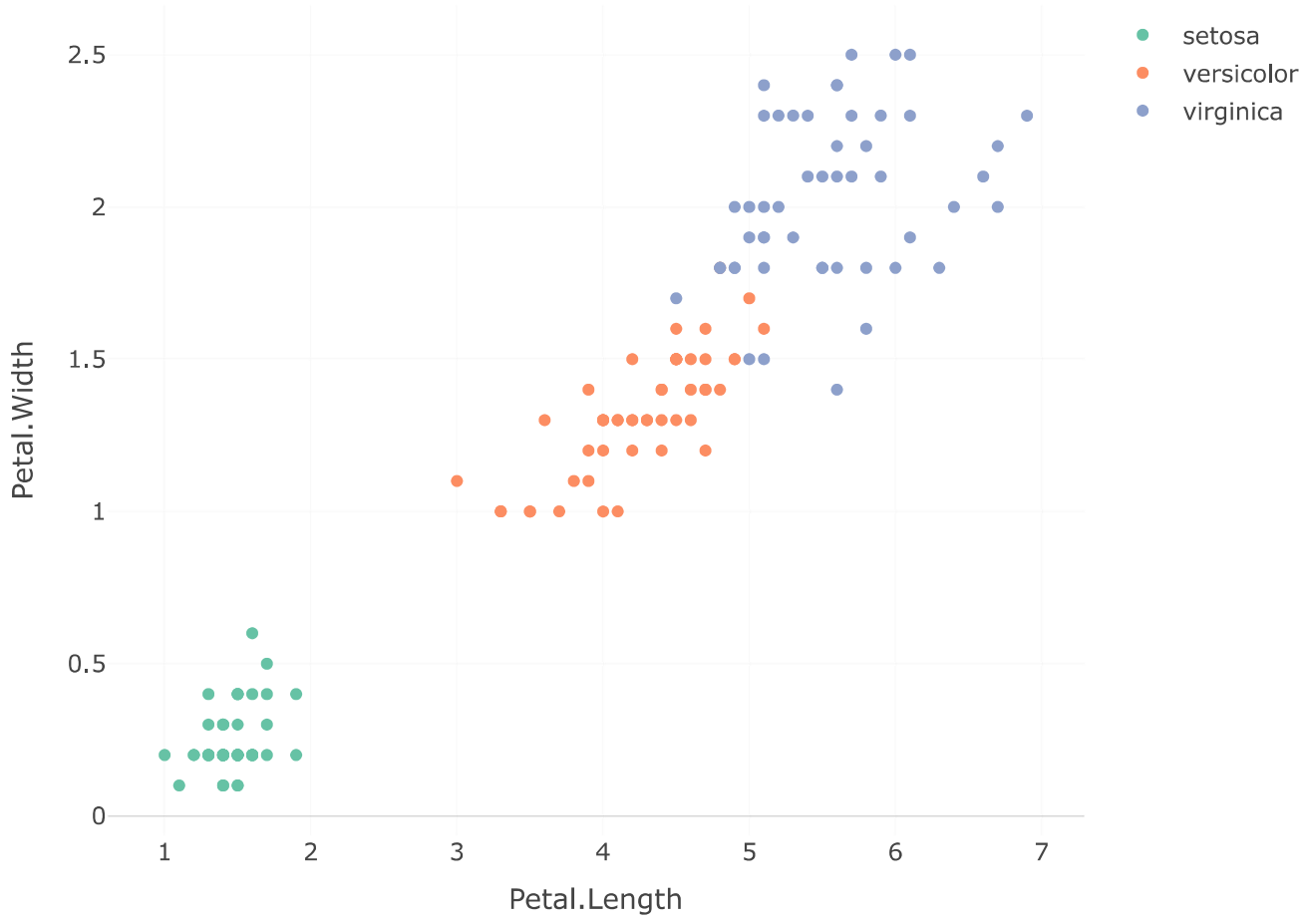
```
# Q1.7. Repeat above plot but using box plot (2 points)
plot_ly(data = iris_gathered, x = ~value, y = ~interaction(Species,metric), color = ~Species, ty
pe = "box")
```






```
# Q1.8. Choose two metrics which separates species the most and use it to make scatter plot
# color points by species (2 points)
plot_ly(data = df, x = ~Petal.Length, y = ~Petal.Width, color = ~Species, type = "scatter")
```

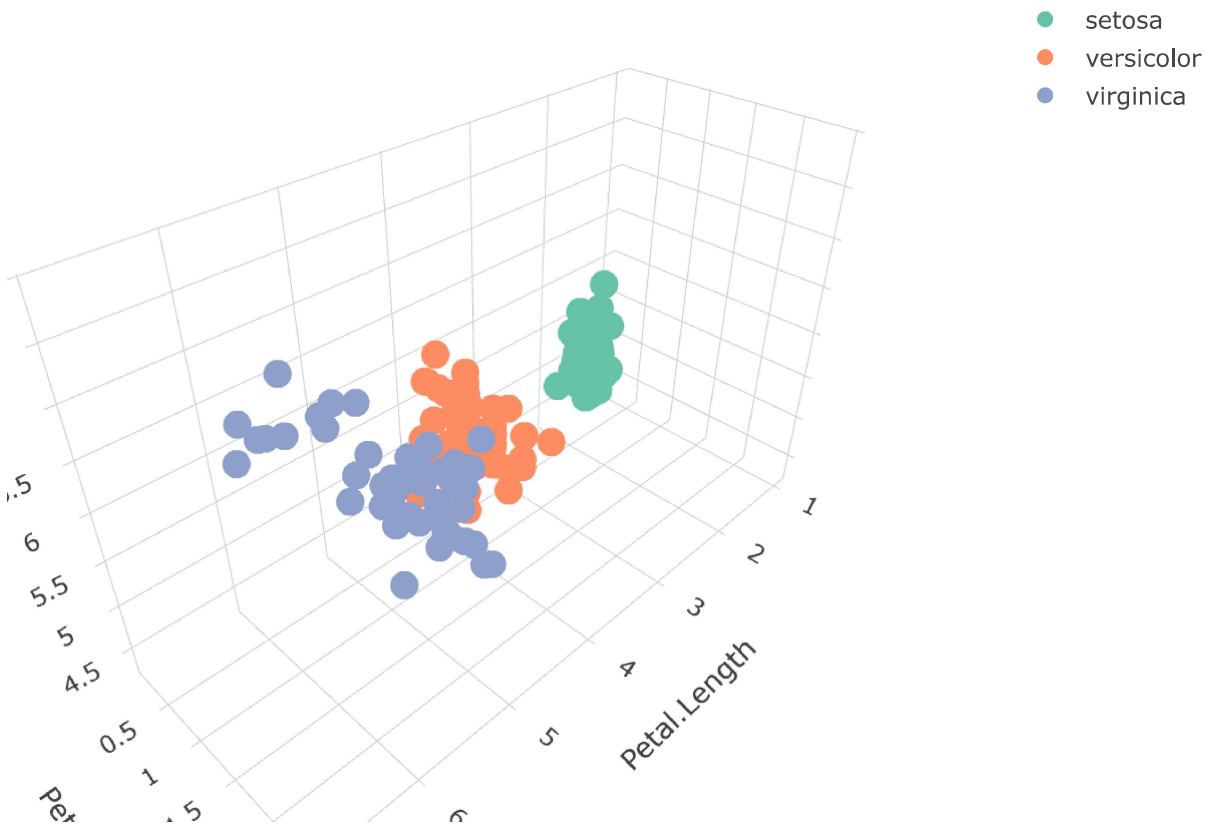
```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode
```



```
# Q1.9. Choose three metrics which separates species the most and use it to make 3d plot
# color points by species (2 points)
plot_ly(data = df, x = ~Petal.Length, y = ~Petal.Width, z= ~Sepal.Length, color = ~Species)
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter3d' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter3d
```

```
## No scatter3d mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode
```



Q1.10. Comment on species separation (2 points): The feature that seperates the species best are Petal Length and Petal Width. Among the three species, Setosa is the most seperated based, while Virginica & versicolor are pretty similar in some edge cases. Virginica has the Longest Sepal & Petal & thickest Petals meanwhile Setosa is short and all the three parameters. Versicolor lies in the middle and all these three parameters.

Part 2. Covid-19 Dataset. (18 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)

us <- fread("us-states.csv",stringsAsFactors=TRUE)
```

```
# Q2.2. Show some values from dataframe
head(us,6)
```

date <date>	state <fct>	fips <int>	cases <int>	deaths <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 (2 points)
# hint:
#    is cases column cumulative or not cumulative?

us$year <- year(ymd(us$date))
us$month <- month(ymd(us$date))
us$day <- day(ymd(us$date))
df$date <- as.POSIXct( df$date, format="%Y/%m/%d" )
us_grp_state <- us %>% group_by(state,year,month)  %>%
                    summarise(date = max(date),fips = max(fips),cases = max(cases), deaths = max
(deaths))
```

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

```
final_us_state <- us_grp_state %>% group_by(state) %>% mutate(lag = shift(cases, 1, 0, type="la
g"), cases, total_cases_month = (cases - lag) )

final_us_state
```

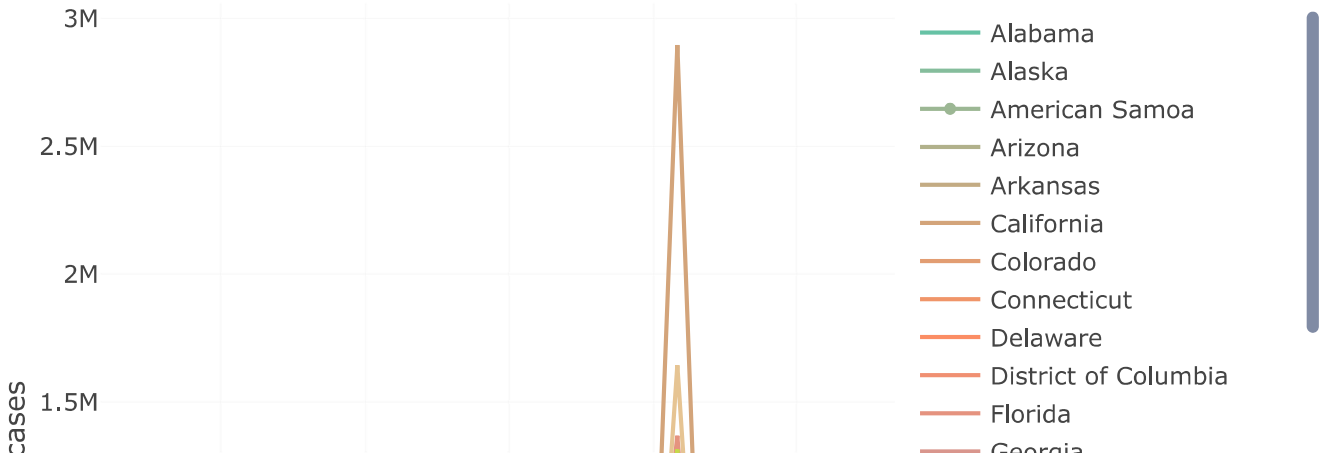
state <fct>	y… <dbl>	mo… <dbl>	date <date>	fips <int>	cases <int>	deaths <int>	lag <int>	total_cases_month <int>
Alabama	2020	3	2020-03-31	1	999	14	0	999

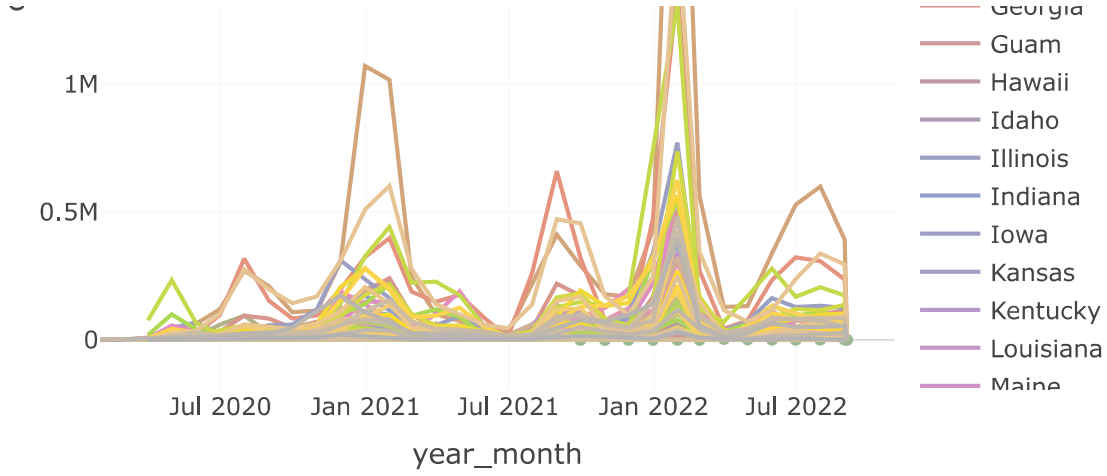
state <fct>	y… <dbl>	mo… <dbl>	date <date>	fips <int>	cases <int>	deaths <int>	lag <int>	total_cases_month <int>
Alabama	2020	4	2020-04-30	1	7068	272	999	6069
Alabama	2020	5	2020-05-31	1	17952	630	7068	10884
Alabama	2020	6	2020-06-30	1	38045	950	17952	20093
Alabama	2020	7	2020-07-31	1	87723	1580	38045	49678
Alabama	2020	8	2020-08-31	1	126058	2182	87723	38335
Alabama	2020	9	2020-09-30	1	154701	2540	126058	28643
Alabama	2020	10	2020-10-31	1	192285	2967	154701	37584
Alabama	2020	11	2020-11-30	1	249524	3578	192285	57239
Alabama	2020	12	2020-12-31	1	361226	4827	249524	111702

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

```
# 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)
plot_ly(data = final_us_state, x = ~date , y = ~total_cases_month , color = ~state, type = 'scat
ter', mode = "line")%>%
  layout(xaxis = list(title = 'year_month'),
         yaxis = list(title = 'cases')
         )
```

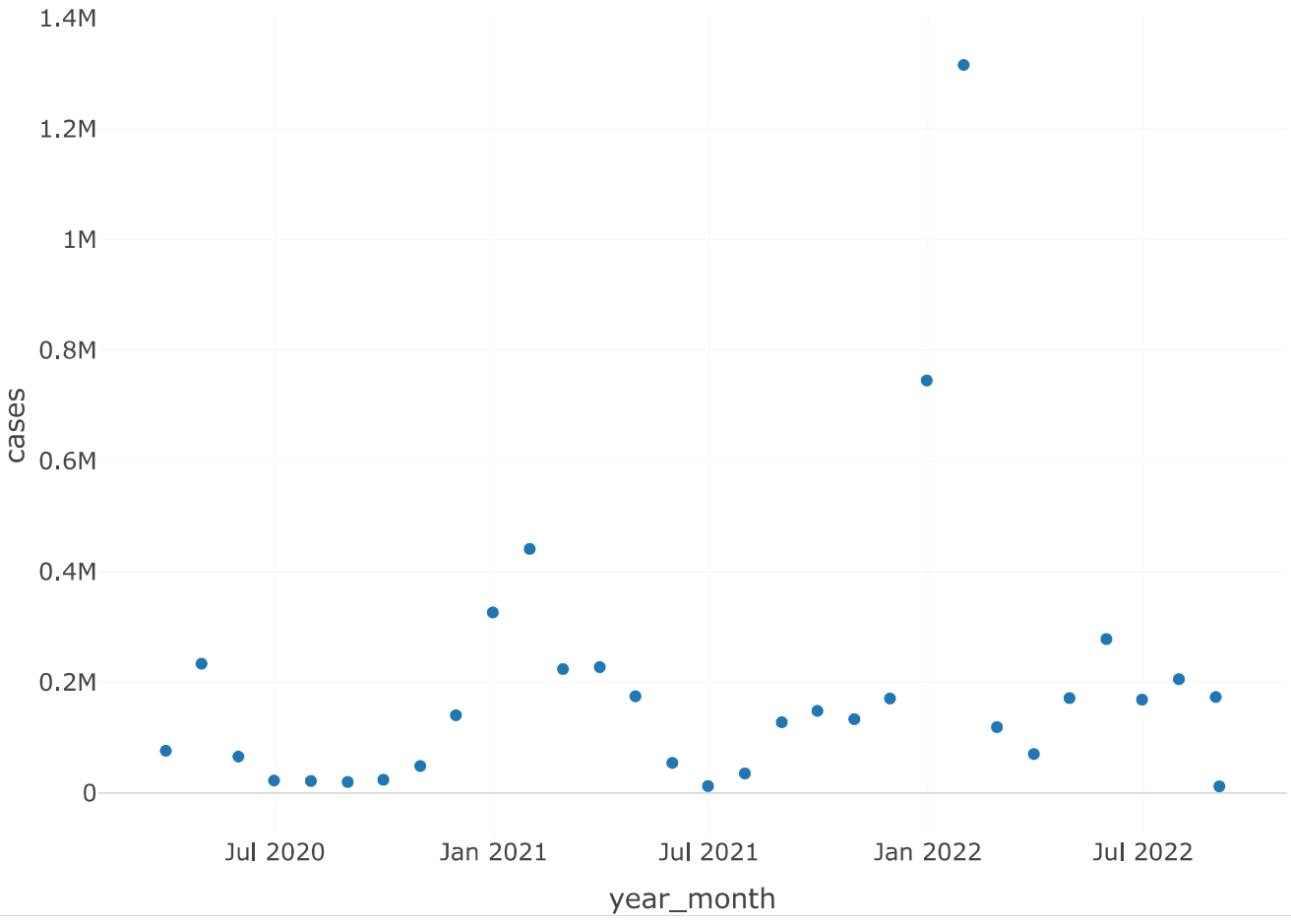
```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): n too large, allowed maximum for palette Set2
is 8
## Returning the palette you asked for with that many colors

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






```
# Q2.5.Plot new monthly cases only in NY state
# (2 points)
new_york_state <- final_us_state %>% filter(state == "New York")
plot_ly(data = new_york_state , x = ~date , y = ~total_cases_month, type = "scatter")%>%
  layout(xaxis = list(title = 'year_month'),
         yaxis = list(title = 'cases')
         )
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode
```

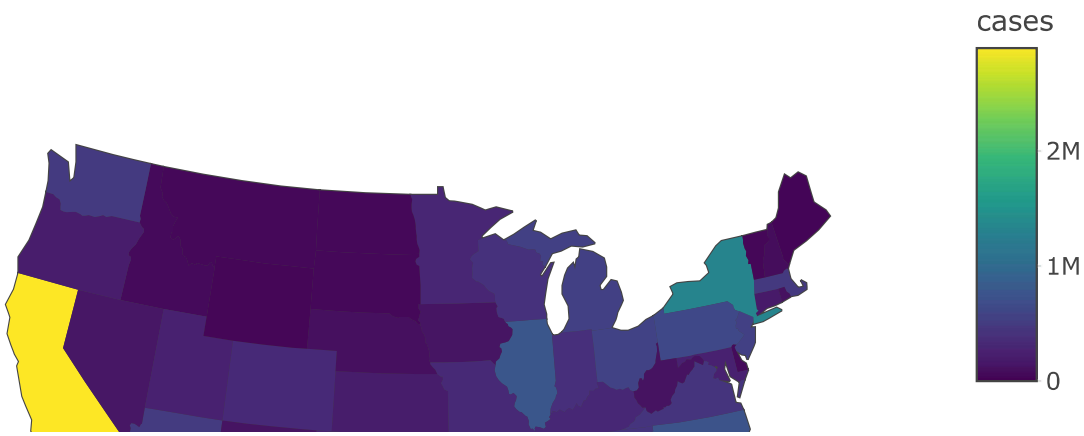


```
# Q2.6. Found the year-month with highest cases in NY state
# (2 points)
max_cases_new_york <- new_york_state %>% group_by(state) %>% slice(which.max(total_cases_month))
max_cases_new_york_final <- max_cases_new_york %>% select(-year,-month,-lag)

max_cases_new_york_final
```

state <fct>	date <date>	fips <int>	cases <int>	deaths <int>	total_cases_month <int>
New York	2022-01-31	36	4789532	64247	1315562

1 row

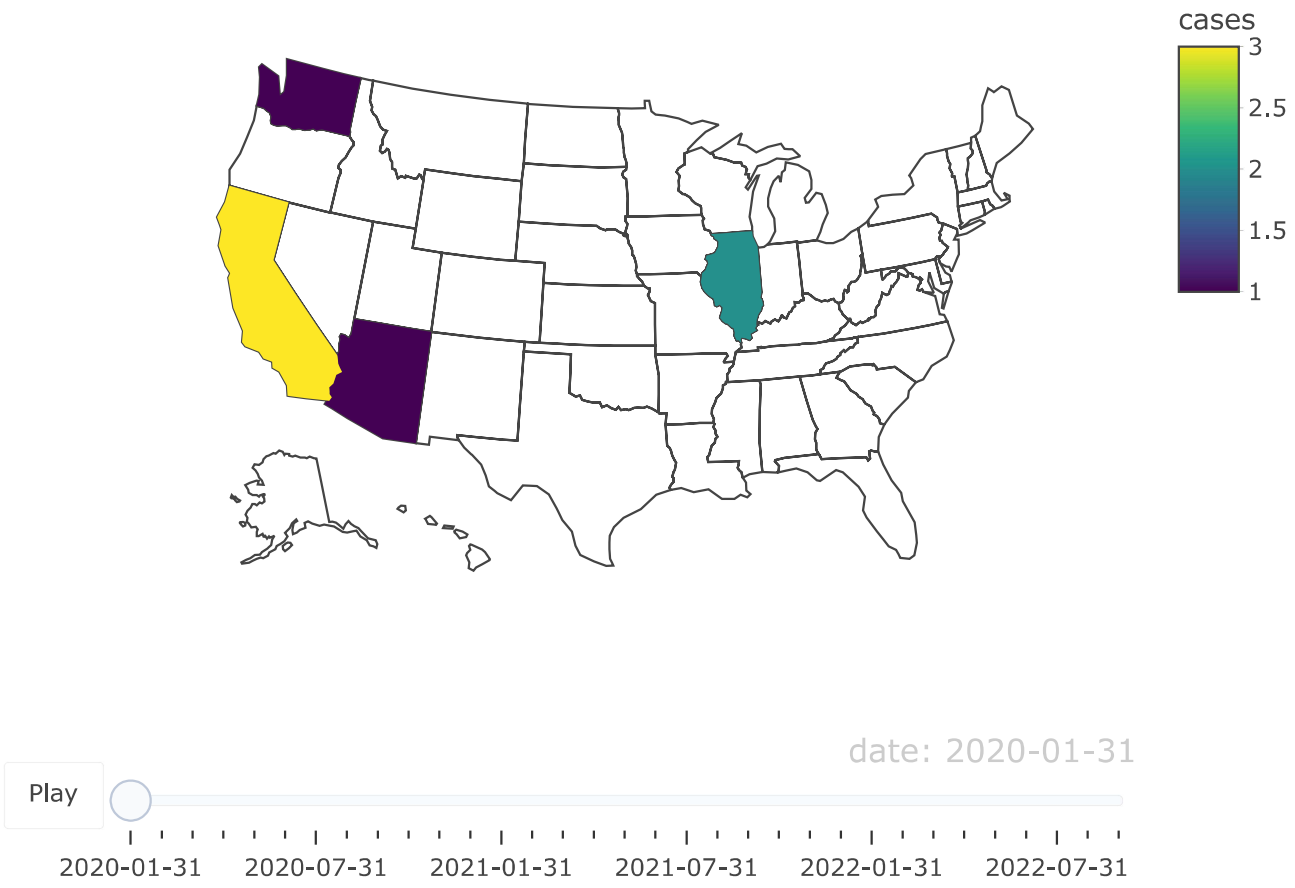
```
# Q2.7. Plot new cases in determined above year-month
# using USA state map, color each state by number of cases  (2 points)
# hint:
#   there two build in constants in R: state.abb and state.name
#   to convert full name to abbreviation
us_state_jan_2022 <- final_us_state %>% filter( year == 2022,month == 1)

us_state_jan_2022 <- us_state_jan_2022 %>% mutate(code = state.abb[match(state,state.name)])

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

plot_geo() %>%
  add_trace(
    z = ~us_state_jan_2022$total_cases_month, text=us_state_jan_2022$state, span = I(0),
    locations=us_state_jan_2022$code, locationmode = 'USA-states'
  ) %>%
  layout(geo = g) %>%
  colorbar(title = "cases")
```



```
# Q2.8. Add animation capability (2 points)
# hint:
#     for variable frame you need either integer or character/factorial so
#     convert date to character or factorial
#final_us_state <- final_us_state %>% mutate(date = format(date, "%Y-%m"))

anim <- final_us_state %>% mutate(date = format(as.character((date))), code = state.abb[match(st
ate,state.name)])


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

state	y…	mo…	date	fips	cases	deat…	lag	total_cases_month	c…
<fct>	<dbl>	<dbl>	<chr>	<int>	<int>	<int>	<int>	<int>	<chr>
Alabama	2020	3	2020-03-31	1	999	14	0	999	AL
Alabama	2020	4	2020-04-30	1	7068	272	999	6069	AL
Alabama	2020	5	2020-05-31	1	17952	630	7068	10884	AL
Alabama	2020	6	2020-06-30	1	38045	950	17952	20093	AL
Alabama	2020	7	2020-07-31	1	87723	1580	38045	49678	AL
Alabama	2020	8	2020-08-31	1	126058	2182	87723	38335	AL
Alabama	2020	9	2020-09-30	1	154701	2540	126058	28643	AL
Alabama	2020	10	2020-10-31	1	192285	2967	154701	37584	AL
Alabama	2020	11	2020-11-30	1	249524	3578	192285	57239	AL
Alabama	2020	12	2020-12-31	1	361226	4827	249524	111702	AL

```
plot_geo() %>%
  add_trace(
    z = ~anim$total_cases_month, text=anim$state, span = I(0),
    locations=anim$code, locationmode = 'USA-states',frame = ~anim$date
  ) %>%
  layout(geo = g) %>%
  colorbar(title = "cases")%>%
  animation_slider(
    currentvalue = list(prefix = "date: ")
  )
```



Q2.9. Compare animated plot from Q2.8 to plots from Q2.4/Q2.5 (When you would prefer one or another?) (2 points)

The animated plot is more user friendly and provides better access to the time series data hence I would prefer it. The plot in Q2.4 is useful for studying the pattern across USA. It will be useful if we want to study a pattern in USA but otherwise in a general scenario I would prefer the animated graph. The plot in Q2.5 is limited and shows only one state, New York. It will be useful if we want to study a pattern only in New York but otherwise in a general scenario I would prefer the animated graph.