Homework 1. R and Interactive Visualization (60 points)

Puja Kumari

2023-09-16

```
library(tidyverse)
library(plotly)
```

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
```

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

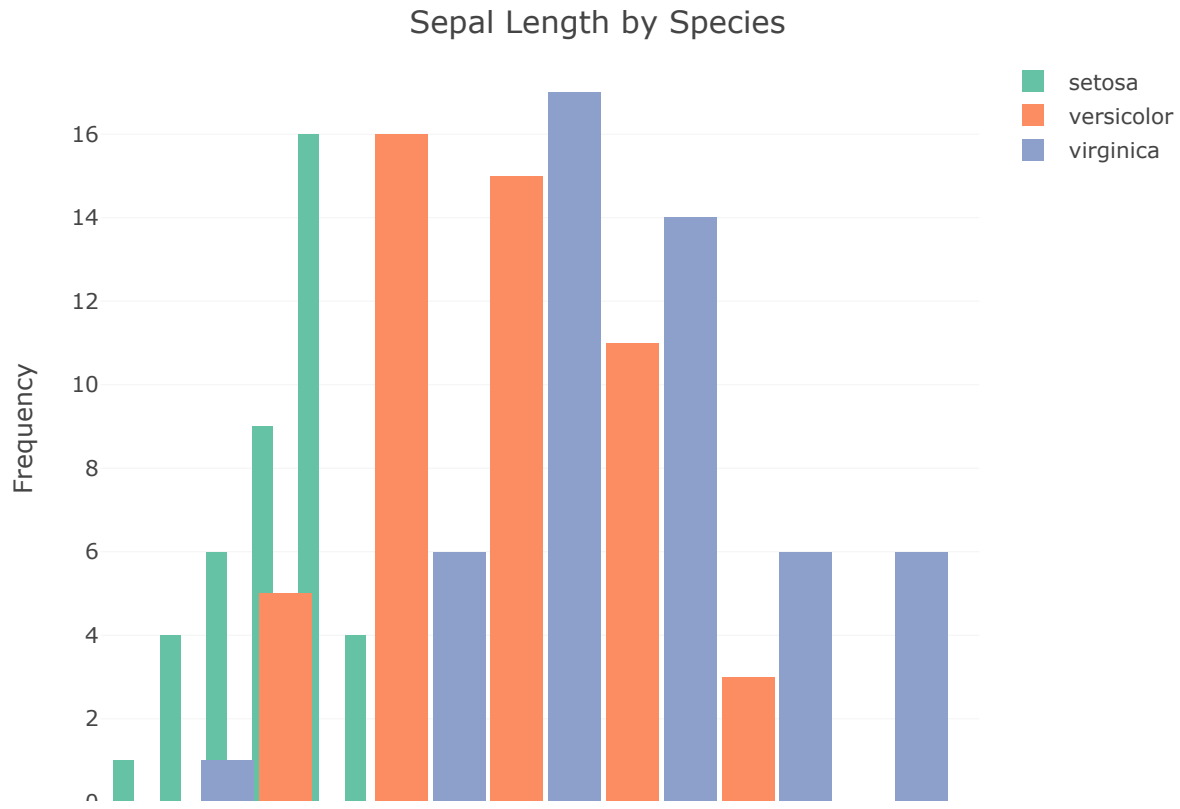
```
## Classes 'data.table' and 'data.frame':   150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

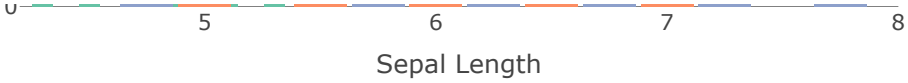
```
# Q1.2. Show some values from data frame (2 points)
head(dataIris)
```

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

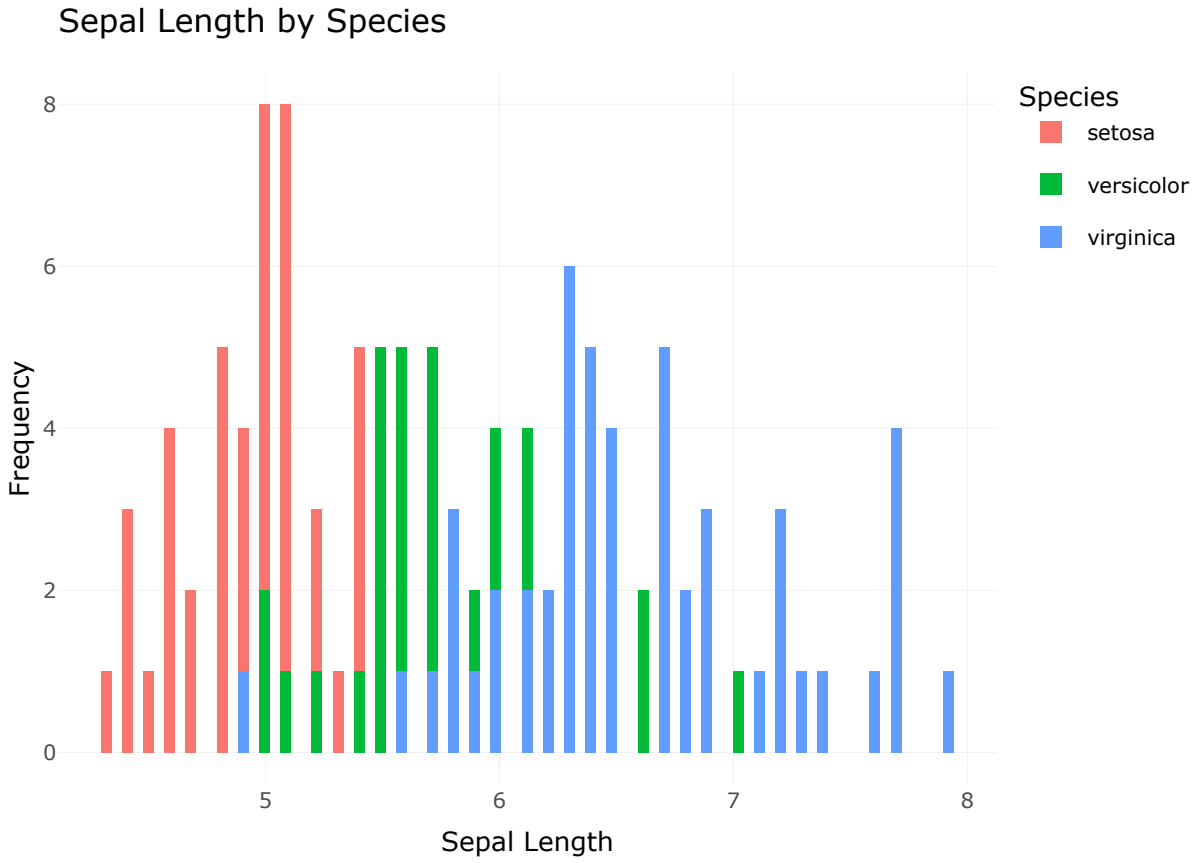
6 rows

```
# 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
library(plotly)
plot_ly(dataIris, x = ~Sepal.Length, color = ~Species, type = "histogram") %>%
  layout(title = "Sepal Length by Species",
         xaxis = list(title = "Sepal Length"),
         yaxis = list(title = "Frequency"),
         barmode = "overlay",
         bargap = 0.55)
```

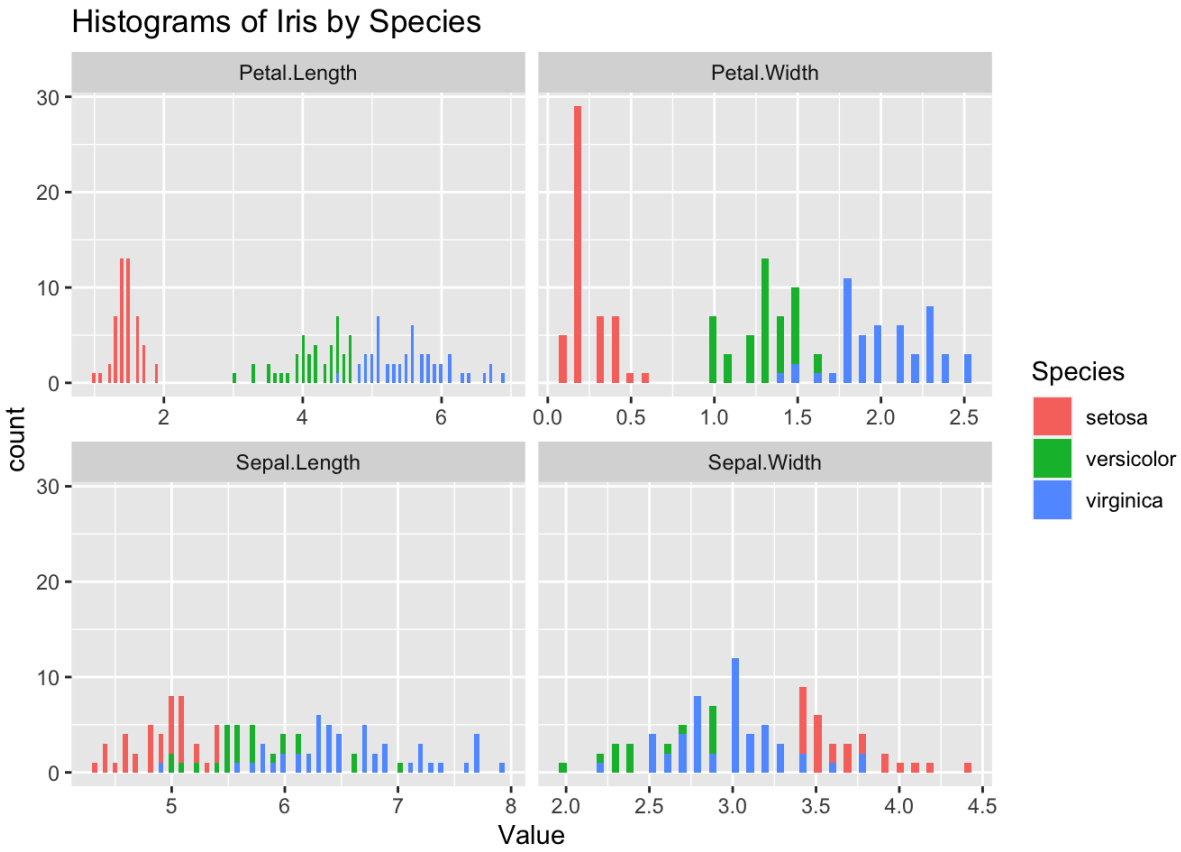





```
# Q1.4. Repeat previous plot with ggplot2 and convert it to plotly with ggplotly (3 points)
library(ggplot2)
library(plotly)

gg_histogram <- ggplot(dataIris, aes(x = Sepal.Length, fill = Species)) +
  geom_histogram(binwidth = 0.045, position = "identity") +
  labs(title = "Sepal Length by Species",
       x = "Sepal Length",
       y = "Frequency") +
  theme_minimal()

plotly_histogram <- ggplotly(gg_histogram)

plotly_histogram
```




```
# Q1.5. Create facet 2 by 2 plot with histograms similar to previous but for each metric
# (3 points)
# hint:
#   following conversion to long format can be useful:
#   iris %>% pivot_longer(...)
#
library(plotly)
iris_long <- iris %>%
  pivot_longer(cols = -Species, names_to = "metric", values_to = "value")

ggplot(data = iris_long, aes(x = value, fill = Species)) +
  geom_histogram(binwidth = 0.045, position = "identity") +
  facet_wrap(~ metric, scales = "free_x", nrow = 2) +
  labs(title = "Histograms of Iris by Species", x = "Value")
```



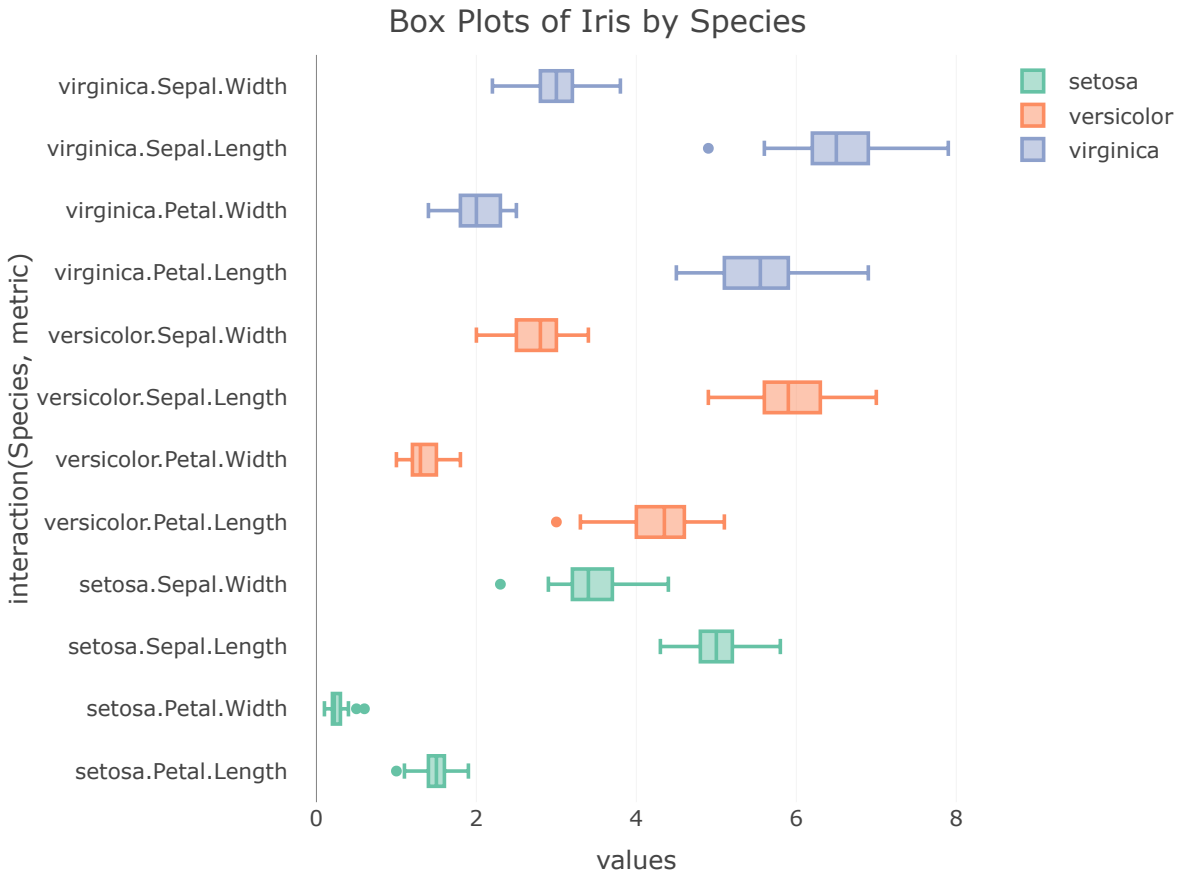
Q1.6. Which metrics has best species separations? (3 points)

From the above plot we can say that petal.length and petal.width have best species separations and we can verify by calculating F-statistic values. Metrics with higher F-statistics have better species separations.We can see that
F_Petal_Length 42.557109
F_Petal_Width 24.957846 have highest F-statistics value.

```
library(dplyr)

iris_grouped <- iris %>%
  group_by(Species)

means_and_vars <- iris_grouped %>%
  summarize(
    mean_Sepal_Length = mean(Sepal.Length),
    var_Sepal_Length = var(Sepal.Length),
    mean_Sepal_Width = mean(Sepal.Width),
    var_Sepal_Width = var(Sepal.Width),
    mean_Petal_Length = mean(Petal.Length),
    var_Petal_Length = var(Petal.Length),
    mean_Petal_Width = mean(Petal.Width),
    var_Petal_Width = var(Petal.Width)
  )

overall_vars <- iris %>%
  summarize(
    overall_var_Sepal_Length = var(Sepal.Length),
    overall_var_Sepal_Width = var(Sepal.Width),
    overall_var_Petal_Length = var(Petal.Length),
    overall_var_Petal_Width = var(Petal.Width)
  )

f_statistics <- means_and_vars %>%
  mutate(
    F_Sepal_Length = overall_vars$overall_var_Sepal_Length / var_Sepal_Length,
    F_Sepal_Width = overall_vars$overall_var_Sepal_Width / var_Sepal_Width,
    F_Petal_Length = overall_vars$overall_var_Petal_Length / var_Petal_Length,
    F_Petal_Width = overall_vars$overall_var_Petal_Width / var_Petal_Width
  )

metric_ranking <- f_statistics %>%
  select(starts_with("F_")) %>%
  summarize_all(mean) %>%
  pivot_longer(everything(), names_to = "Metric", values_to = "F_statistic") %>%
  arrange(desc(F_statistic))

print(metric_ranking)
```

```
## # A tibble: 4 × 2
##   Metric         F_statistic
##   <chr>                <dbl>
## 1 F_Petal_Length       42.6
## 2 F_Petal_Width        25.0
## 3 F_Sepal_Length        3.26
## 4 F_Sepal_Width         1.69
```

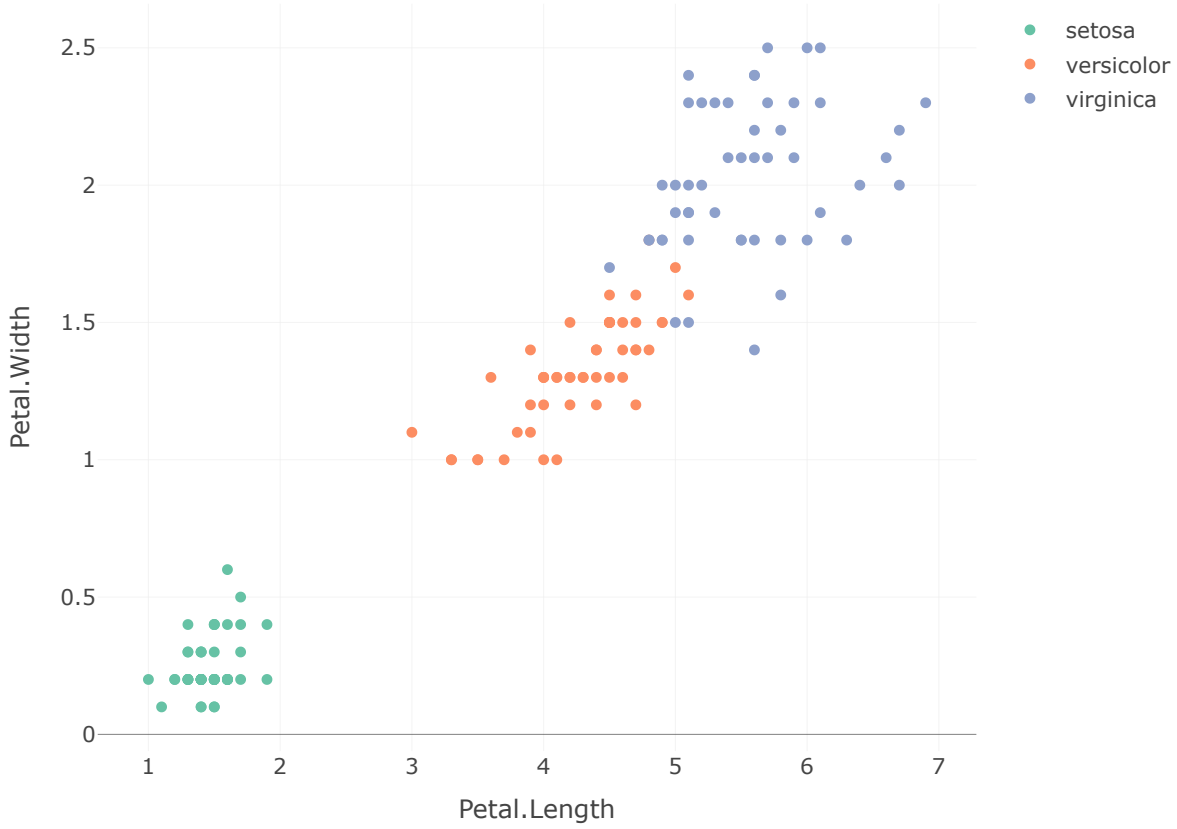
```
# Q1.7. Repeat above plot but using box plot (3 points)
# Load the required libraries

library(plotly)

iris_data <- iris %>%
  select(Species, Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) %>%
  pivot_longer(-Species, names_to = "Metric", values_to = "Value")

metrics_order <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")

iris_data$YLabel <- paste(iris_data$Species, iris_data$Metric, sep = ".")

iris_data$SpeciesFactor <- factor(iris_data$Species, levels = c("setosa", "versicolor", "virgini
ca"))

filtered_data <- iris_data %>%
  filter(Metric %in% metrics_order)

box_plot <- plot_ly(filtered_data,
                    x = ~Value,
                    y = ~YLabel,
                    type = "box",
                    color = ~SpeciesFactor) %>%
  layout(title = "Box Plots of Iris by Species",
         xaxis = list(title = "values"),
         yaxis = list(title = "interaction(Species, metric)"))
box_plot
```



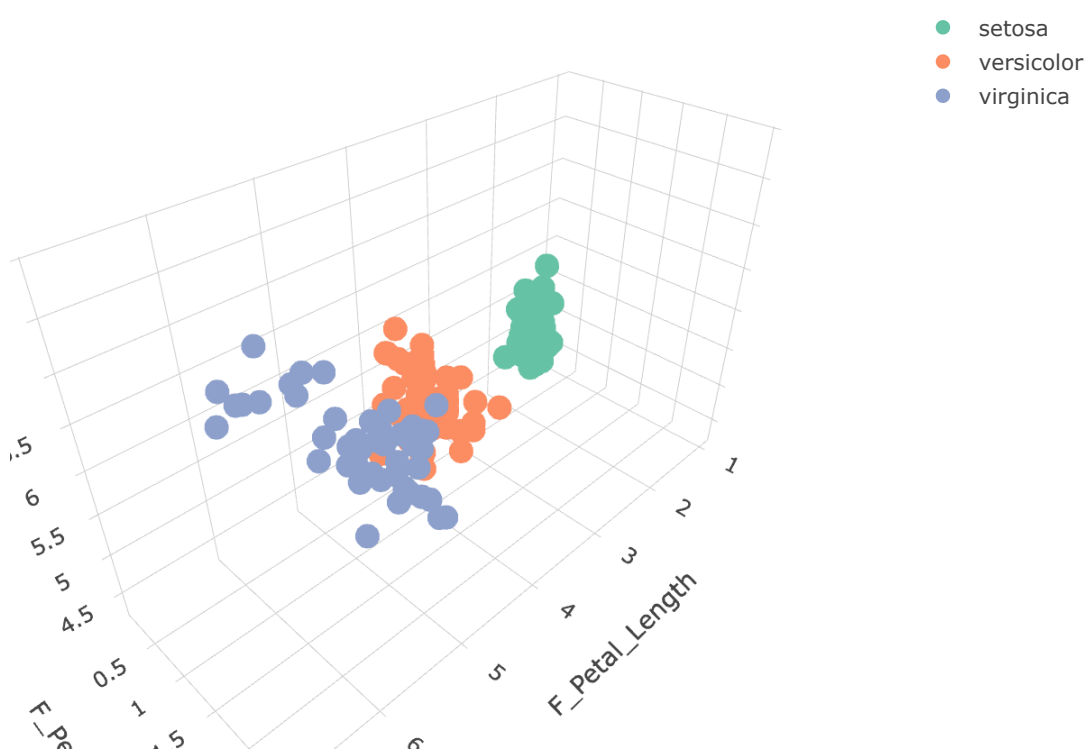
```
# Q1.8. Choose two metrics which separates species the most and use it to make scatter plot
# color points by species (3 points)
# Load the required libraries
library(plotly)

plot_ly(dataIris, x = ~Petal.Length, y = ~Petal.Width, color = ~Species)
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#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 (3 points)
# Load the required libraries
library(plotly)

selected_metrics <- c("F_Petal_Length", "F_Petal_Width", "F_Sepal_Length")

scatter_3d <- plot_ly(dataIris,
                      x = ~Petal.Length,
                      y = ~Petal.Width,
                      z = ~Sepal.Length,
                      color = ~Species
                      ) %>%
  add_markers() %>%
  layout(scene = list(
    xaxis = list(title = selected_metrics[1]),
    yaxis = list(title = selected_metrics[2]),
    zaxis = list(title = selected_metrics[3])
  ))

# Display the 3D scatter plot
scatter_3d
```



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

Based on the analysis and visualization of the data using F-statistics and the creation of 3D scatter plots with selected metrics, we can make the following comments on species separation in the Iris dataset:

The 3D scatter plots vividly illustrate the effectiveness of selected metrics, namely Petal Length, Petal Width, and Sepal Length, in distinguishing between the three Iris species (Setosa, Versicolor, and Virginica). Each species occupies a distinct region in the 3D space, underscoring the robust separation capabilities of these metrics.

Among the chosen metrics, Petal Length and Petal Width emerge as the primary contributors to species differentiation. Notably, Setosa exhibits remarkable isolation from the other two species along these dimensions. Sepal Length also plays a significant role, particularly in discerning Versicolor from Virginica.

The utilization of F-statistics reinforces the quality of species separation achieved by the selected metrics. Higher F-statistic values signify greater dissimilarity between species, indicating a robust discriminative capacity.

In summary, our analysis highlights the significant contribution of Petal Length, Petal Width, and Sepal Length in successfully delineating Iris species. These results are consistent with well-established botanical insights, providing strong support for utilizing these metrics as robust indicators for precise species classification.

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_data <- read.csv("us-states.csv")
```

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

	date <chr>	state <chr>	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

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?

us_states_per_month <- us_states_data %>%
  mutate(date = as.Date(date),
         month = month(date),
         year = year(date),
         yearmonth = floor_date(date, unit = "month")) %>%
  group_by(state, yearmonth) %>%
  summarise(cum_new_cases = max(cases), .groups = "drop") %>%
  mutate(new_cases = cum_new_cases - lag(cum_new_cases, default = 0))
print(us_states_per_month)
```

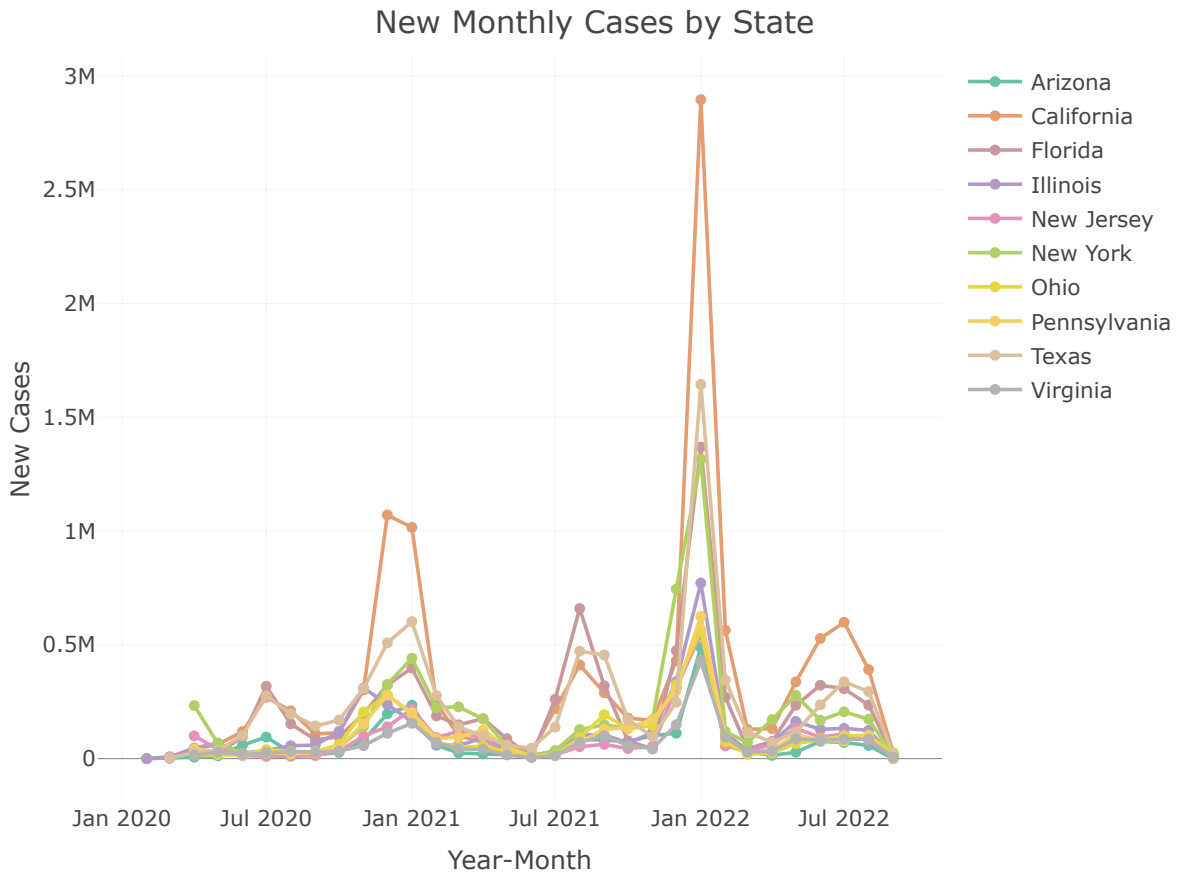
```
## # A tibble: 1,732 × 4
##    state   yearmonth  cum_new_cases new_cases
##    <chr>   <date>             <int>     <int>
##  1 Alabama 2020-03-01           999       999
##  2 Alabama 2020-04-01          7068      6069
##  3 Alabama 2020-05-01         17952     10884
##  4 Alabama 2020-06-01         38045     20093
##  5 Alabama 2020-07-01         87723     49678
##  6 Alabama 2020-08-01        126058     38335
##  7 Alabama 2020-09-01        154701     28643
##  8 Alabama 2020-10-01        192285     37584
##  9 Alabama 2020-11-01        249524     57239
## 10 Alabama 2020-12-01        361226    111702
## # ℹ 1,722 more rows
```

The cases column in the us_states_data dataframe is not cumulative. It represents the number of new cases reported for each day.

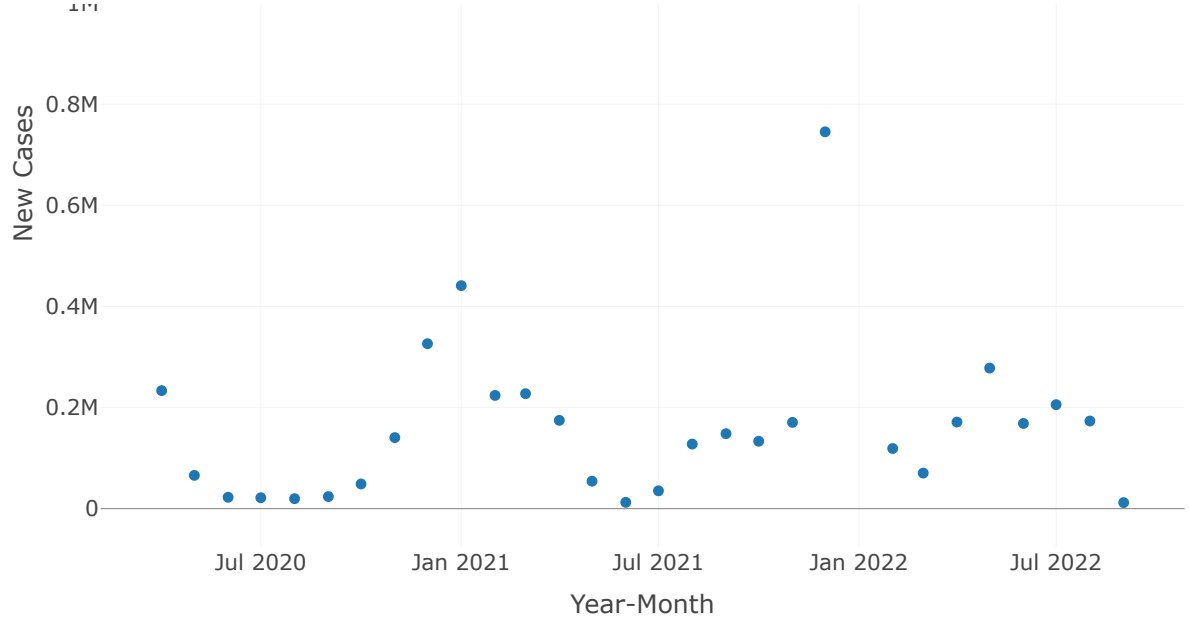
```
# Q2.4.Using previous dataframe plot new monthly cases in states, group by states
# The resulting plot is busy, use interactive plotly capabilities to limit number
# of displayed states
# (4 points)
library(plotly)
top_states_data <- us_states_per_month %>%
  group_by(state) %>%
  summarise(total_new_cases = sum(new_cases)) %>%
  arrange(desc(total_new_cases)) %>%
  slice(1:10)

filtered_data <- us_states_per_month %>%
  filter(state %in% top_states_data$state)

filtered_data <- filtered_data %>%
  filter(new_cases >= 0)

plot <- plot_ly(filtered_data, x = ~yearmonth, y = ~new_cases, color = ~state, type = "scatter",
mode = "lines+markers") %>%
  layout(title = "New Monthly Cases by State",
         xaxis = list(title = "Year-Month"),
         yaxis = list(title = "New Cases"),
         showlegend = TRUE,
         colors = RColorBrewer::brewer.pal(length(unique(filtered_data$state)), "Set3"))

plot
```

```
## 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
```

```
## Warning: 'layout' objects don't have these attributes: 'colors'
## Valid attributes include:
## '_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calendar', 'clic
kmode', 'coloraxis', 'colorscale', 'colorway', 'computed', 'datarevision', 'dragmode', 'editrevi
sion', 'editType', 'font', 'geo', 'grid', 'height', 'hidesources', 'hoverdistance', 'hoverlabe
l', 'hovermode', 'images', 'legend', 'mapbox', 'margin', 'meta', 'metasrc', 'modebar', 'newshap
e', 'paper_bgcolor', 'plot_bgcolor', 'polar', 'scene', 'selectdirection', 'selectionrevision',
'separators', 'shapes', 'showlegend', 'sliders', 'smith', 'spikedistance', 'template', 'ternar
y', 'title', 'transition', 'uirevision', 'uniformtext', 'updatemenus', 'width', 'xaxis', 'yaxi
s', 'barmode', 'bargap', 'mapType'
```



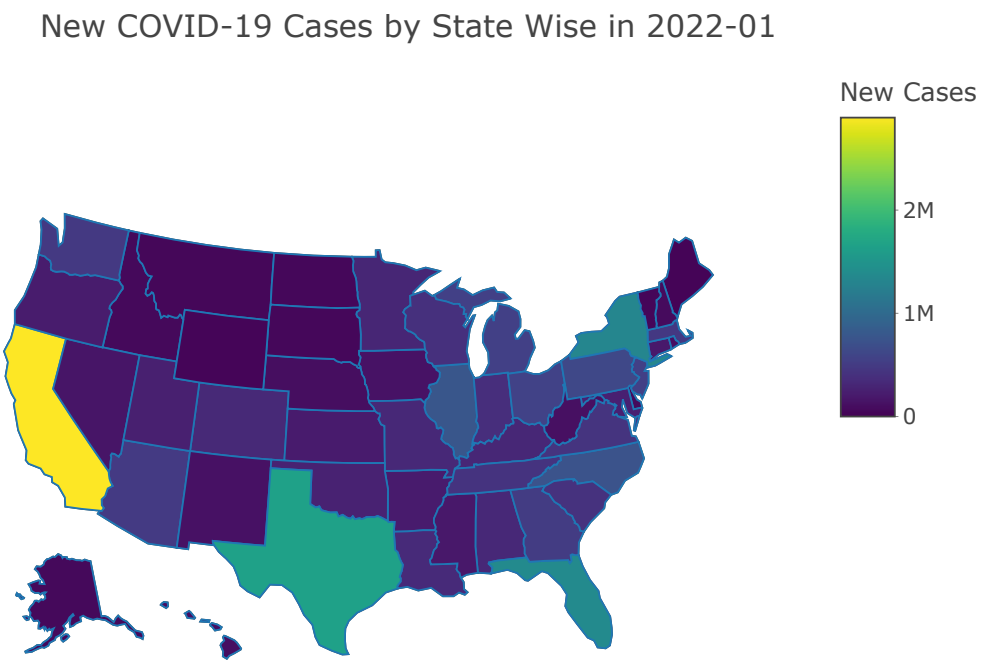

```
# Q2.5.Plot new monthly cases only in NY state
# (3 points)
# Filter the data for NY state
library(plotly)
ny_data <- us_states_per_month %>%
  filter(state == "New York", new_cases >= 0)

plot_NewYork <- plot_ly(ny_data, x = ~yearmonth, y = ~new_cases, type = "scatter", mode = "marke
rs") %>%
  layout(title = "New Monthly COVID-19 Cases in New York",
         xaxis = list(title = "Year-Month"),
         yaxis = list(title = "New Cases"))

# Show the interactive plot
plot_NewYork
```



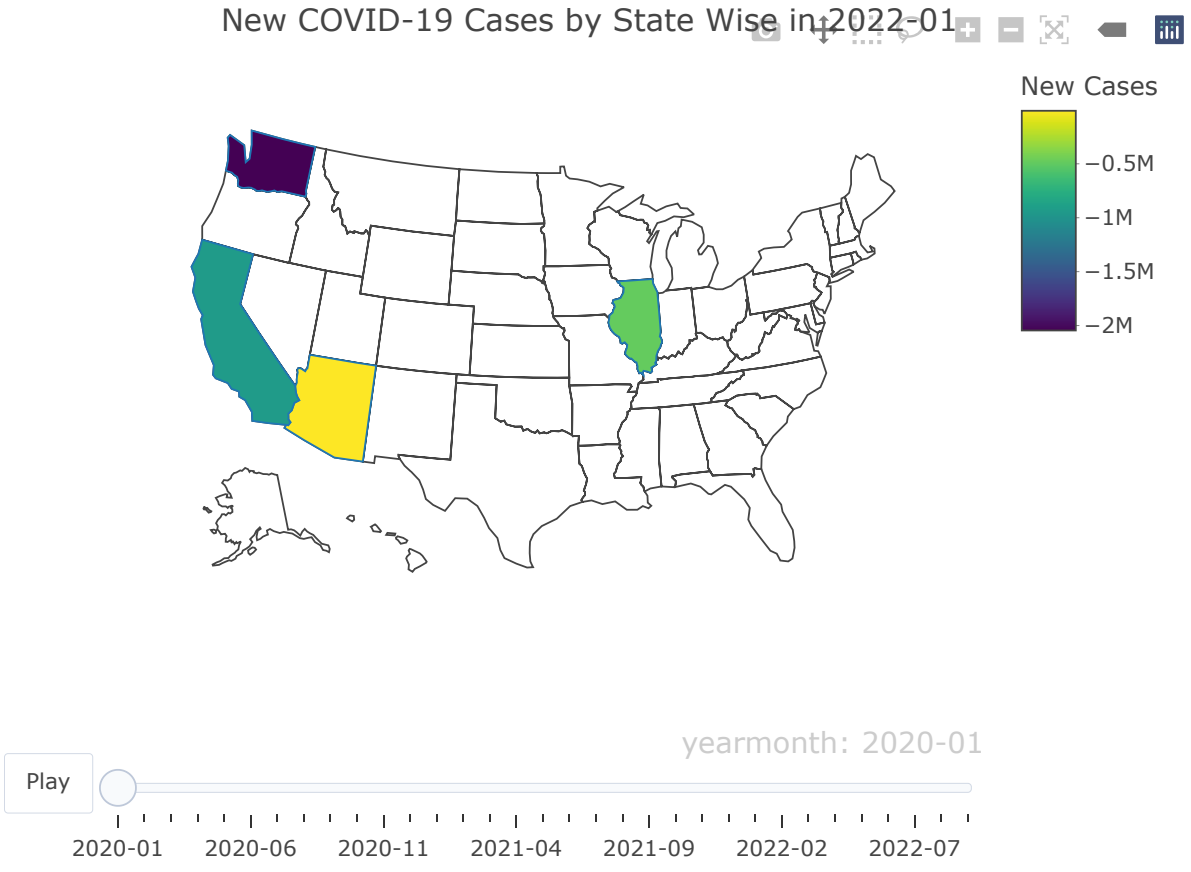

```
# Q2.6. Found the year-month with highest cases in NY state
# (3 points)
ny_data <- us_states_per_month %>%
  filter(state == "New York")
highest_cases_month <- ny_data %>%
  filter(new_cases == max(new_cases))
print(highest_cases_month)
```

```
## # A tibble: 1 × 4
##   state    yearmonth  cum_new_cases new_cases
##   <chr>    <date>             <int>     <int>
## 1 New York 2022-01-01       4789532   1315562
```

```
# 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

library(plotly)
library(lubridate)

highest_cases_yearmonth <- "2022-01"

us_states_per_month <- us_states_per_month %>%
  mutate(yearmonth = ymd(yearmonth))

selected_month_data <- us_states_per_month %>%
  filter(format(yearmonth, "%Y-%m") == highest_cases_yearmonth)

state_names_data <- data.frame(abbreviation = state.abb, state_name = state.name)

selected_month_data <- merge(selected_month_data, state_names_data, by.x = "state", by.y = "stat
e_name", all.x = TRUE)

map_plot <- plot_ly(selected_month_data, type = "choropleth", locations = ~abbreviation, locatio
nmode = "USA-states", z = ~new_cases,
  text = ~paste("State: ", state, "<br> New Cases: ", new_cases, "<br> Year-Month: ", format(yea
rmonth, "%b %Y")),
  colorscale = "Viridis",
  colorbar = list(title = "New Cases", zmin = 1, zmax = 2))

map_plot <- map_plot %>% layout(title = paste("New COVID-19 Cases by State Wise in", highest_cas
es_yearmonth),
  geo = list(scope = "usa"))


map_plot
```



```
# 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
library(plotly)
library(lubridate)

us_states_per_month <- us_states_per_month %>%
  mutate(yearmonth = format(ymd(yearmonth), "%Y-%m"))

state_names_data <- data.frame(abbreviation = state.abb, state_name = state.name)

us_states_per_month <- merge(us_states_per_month, state_names_data, by.x = "state", by.y = "stat
e_name", all.x = TRUE)

if (!"abbreviation" %in% colnames(us_states_per_month)) {
  stop("Column 'abbreviation' not found in the merged data frame.")
}

map_plot <- plot_ly(us_states_per_month, type = "choropleth", locations = ~abbreviation, locatio
nmode = "USA-states", z = ~new_cases,
  text = ~paste("State: ", state, "<br> New Cases: ", new_cases, "<br> Year-Month: ", yearmont
h),
  colorscale = "Viridis",
  colorbar = list(title = "New Cases", zmin = 1, zmax = 2),
  frame = ~yearmonth)  # Specify the frame variable

map_plot <- map_plot %>% layout(title = paste("New COVID-19 Cases by State Wise in", highest_cas
es_yearmonth),
  geo = list(scope = "usa"))

map_plot
```



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

The preference for using an animated plot (Q2.8) versus static plots (Q2.4 and Q2.5) depends on the specific goals and audience for our data visualization. Here are some considerations for when we might prefer one over the other:

Animated Plot (Q2.8):

Temporal Analysis: Animated plots are excellent for visualizing changes over time. If our primary goal is to show how COVID-19 cases evolve over different months or periods, an animated plot is a great choice.

Engagement: Animated plots can be more engaging and visually appealing, especially when presenting to a general audience or in interactive dashboards.

Storytelling: They are useful for storytelling and conveying the progression of events or trends. Static Plots (Q2.4/Q2.5):

Cross-Sectional Comparison: When the goal is to compare COVID-19 cases among states or regions at a specific moment or between different states, static plots take the lead. They offer a snapshot view of the data.

Comparative Analysis: If we want to compare COVID-19 cases across states or regions for a specific point in time or between different states, static plots are more suitable. They provide a snapshot view of the data.

Publication: Static plots are commonly used in academic papers, reports, or publications where clarity and precise data representation are crucial.

Limited Interactivity: If we prefer a simple, less interactive visualization or our audience may not benefit from the interactivity of animated plots, static plots are more appropriate.

In essence, our decision hinges on the nature of our data analysis, the audience we serve, and the narrative we wish to craft. Additionally, we can harmoniously employ both animated and static plots within a larger data visualization framework to offer a comprehensive data perspective.