Interactive Data Visualization – Part 1

Main Code

```
plot_ly(data = dataframe, x = ~name_of_variable1,
                  y = ~name_of_variable2,
          color = ~name_variable3,
          symbol = ~name_variable3,
          type = "", mode = "",
          colors = pallete_vector)
```

Type	Mode	Type of Plot	Add on functions
type = “scatter”	mode = “markers”	Scatter Plot	add_trace() layout()
	mode = "lines"	Line Graph	
	mode = "lines+markers"	Line Graph with dots	
type = "histogram"		Histogram	add_histogram() layout()
type = "bar"		Bar Graph	add_trace() layout() Inside layout: barmode = "group" in layout barmode = "stack" in layout
type = "box"		Box Plot	add_trace

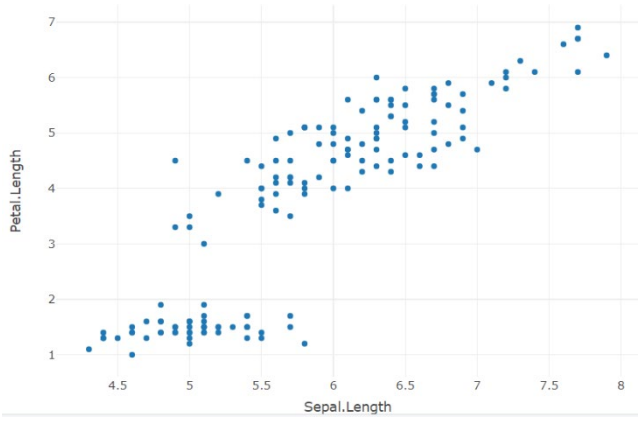
Dataset: Iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
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
11	5.4	3.7	1.5	0.2	setosa

Scatter Plots with Plotly

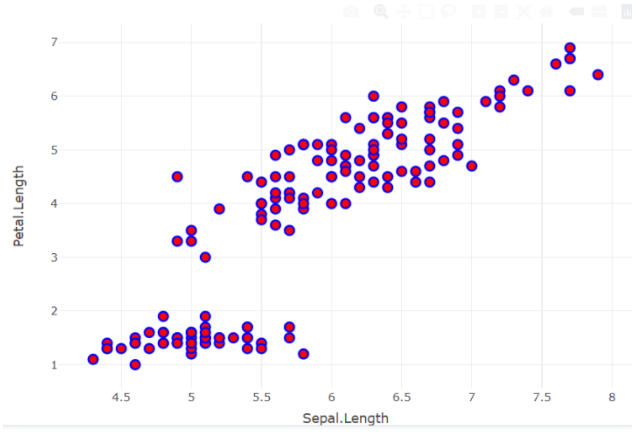
Simple Scatter Plot

```
fig1 <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length,
                type = "scatter", mode = "markers")
fig1
```



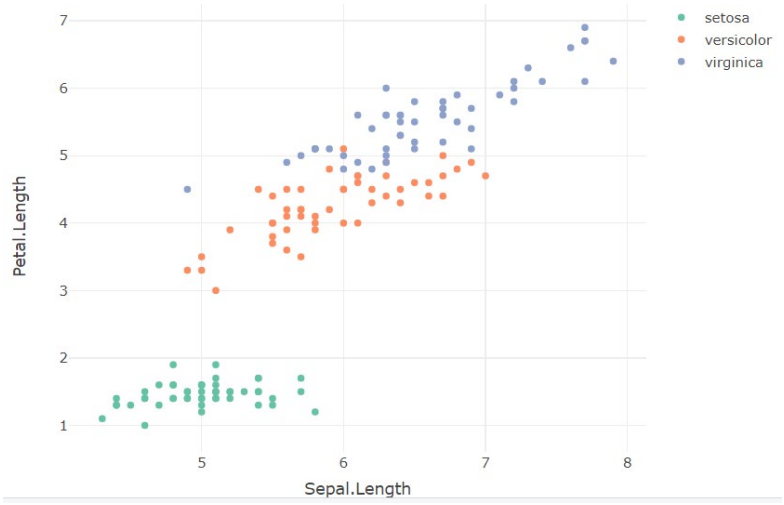
Stylish Dots

```
fig2 <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length,
                type = "scatter", mode = "markers",
                marker = list(size = 10,
                              color = "red",
                              line = list(color = "blue",
                                          width = 2)))
fig2
```



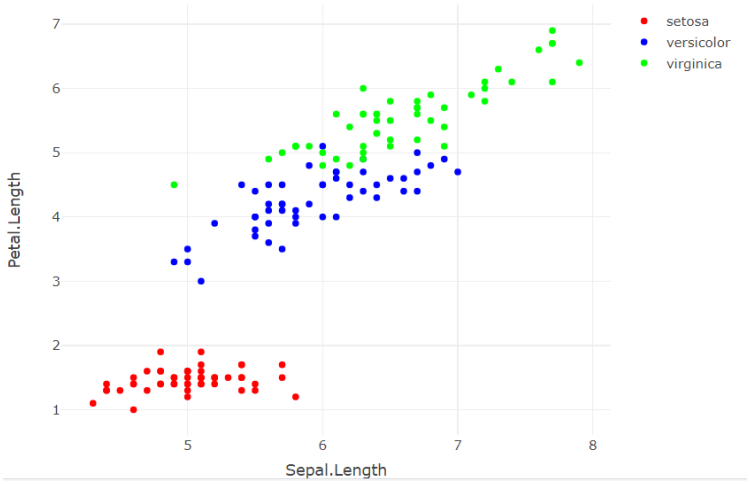
Three Variables

```
fig3 <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length,
                color = ~Species,
                type = "scatter", mode = "markers")
fig3
```



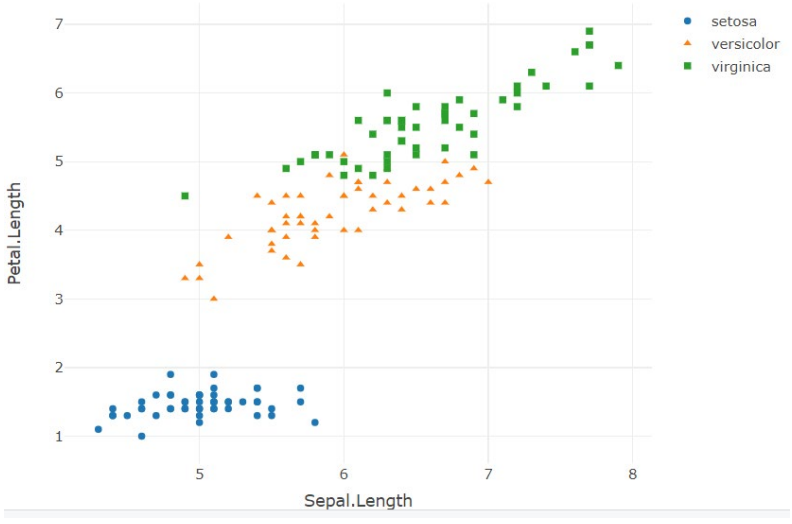
Manually Change the Colors for the Third Variable

```
fig4 <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length,
                color = ~Species,
                type = "scatter", mode = "markers",
                colors = c("red", "blue", "green"))
fig4
```



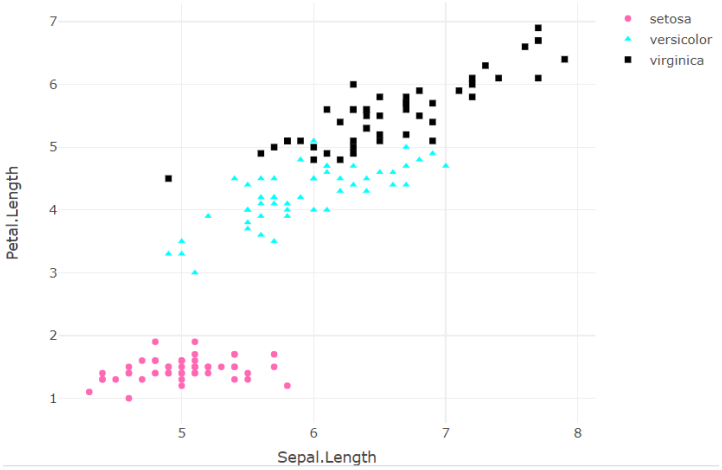
Symbols

```
fig5 <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length,
                symbol = ~Species,
                type = 'scatter', mode = 'markers')
fig5
```



Changing the colors of the symbols.

```
fig6 <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length,
                symbol = ~Species,
                color = ~Species,
                type = 'scatter', mode = 'markers',
                colors = c("hotpink", "cyan", "black"))
fig6
```



Dataset: dummy_data_1

	x	col_1	col_2	col_3	col_4
1	1	5.181458	-0.160951045	-0.951807493	-6.273066
2	2	5.768219	0.479718521	-0.610394550	-5.378716
3	3	6.520355	-0.846518233	-0.575741088	-4.148412
4	4	5.488036	0.355705894	2.464083378	-6.305281
5	5	5.088665	0.512848807	-0.499907128	-5.396206
6	6	4.392657	0.401019388	1.399282445	-4.572251
7	7	4.336057	-0.125980801	-0.746640188	-6.134401
8	8	4.651283	-0.245551624	0.423829716	-6.329959
9	9	6.017906	-0.160596048	0.821952838	-4.140733
10	10	4.213647	0.132291868	1.528079629	-4.230242

Layout (titles, background...etc)

Figure without layout

```
fig_7 <- plot_ly(data = dummy_data_1, x = ~col_2, y = ~col_3,
                         type = "scatter", mode = "markers")
fig_7
```

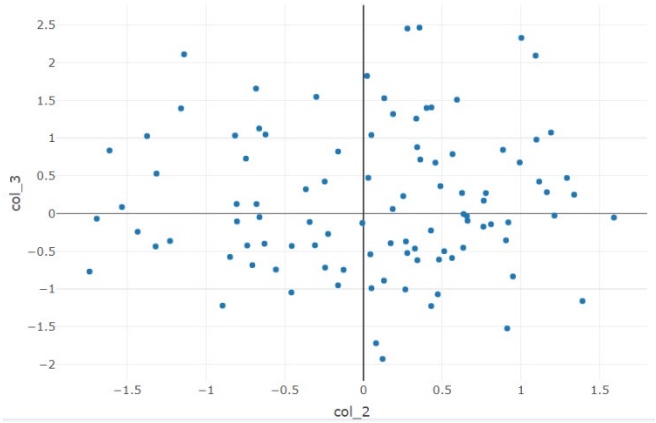


Figure with layout

```
fig8 <- plot_ly(data = dummy_data_1, x = ~col_2, y = ~col_3,
                type = "scatter", mode = "markers") %>%
  layout(title = "This is the Title of the Plot",
         legend = list(title=list(text="Legend Title")),
         plot_bgcolor="yellow",
         xaxis = list(
           title = "This is my x axis",
           zerolinecolor = "blue",
           zerolinewidth = 10,
           gridcolor = "red"),
         yaxis = list(
           title = "This is my y axis",
           zerolinecolor = "pink",
           zerolinewidth = 10,
           gridcolor = "black"))
fig8
```

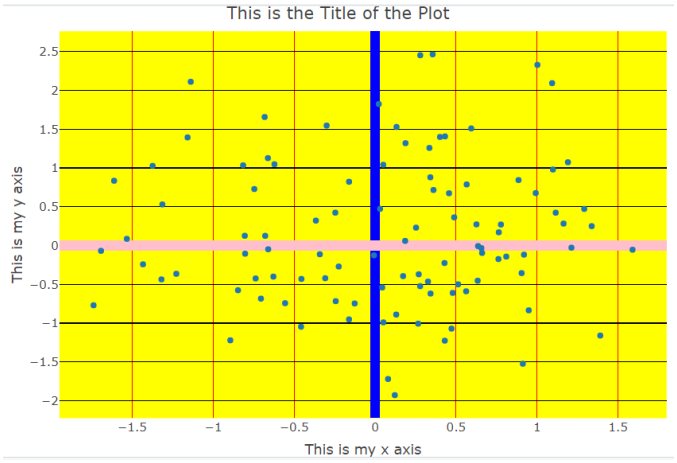
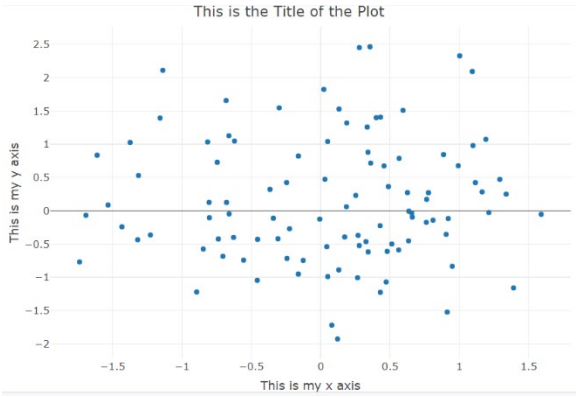


Figure with layout (zeroline)

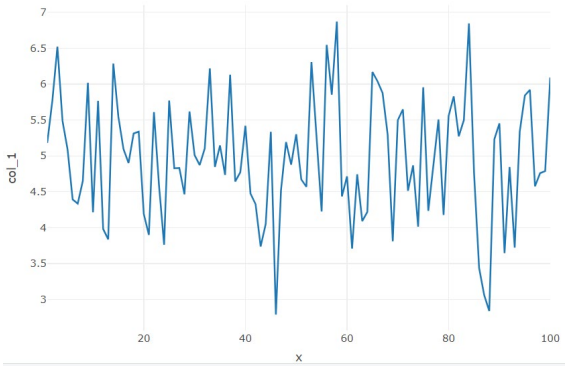
```
fig9 <- plot_ly(data = dummy_data_1, x = ~col_2, y = ~col_3,
                type = "scatter", mode = "markers") %>%
  layout(title = "This is the Title of the Plot",
         xaxis = list(
           title = "This is my x axis",
           zeroline = FALSE),
         yaxis = list(
           title = "This is my y axis"))
fig9
```



Line Plots

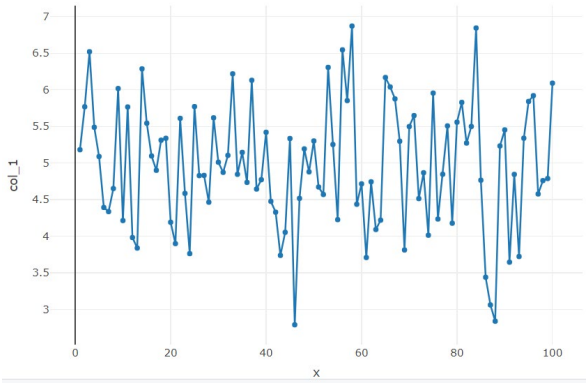
Simple Line Plot

```
fig10 <- plot_ly(data = dummy_data_1, x = ~x, y = ~col_1,
                 type = "scatter", mode = "lines")
fig10
```



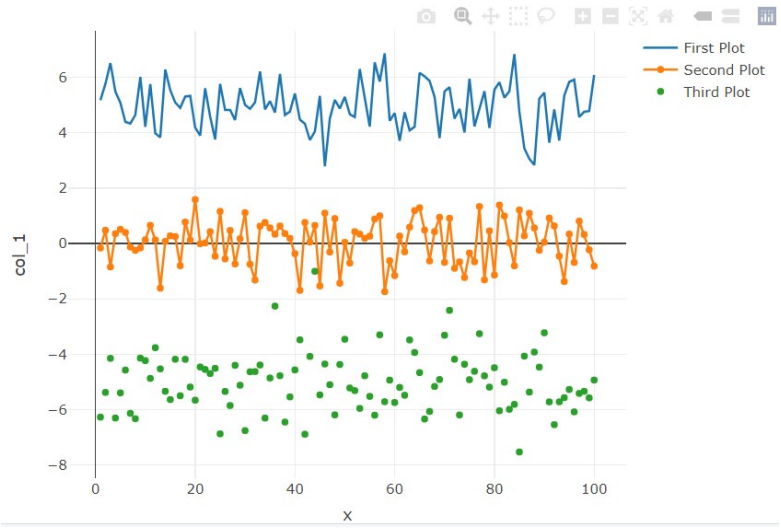
Line Plot with Markers

```
fig11 <- plot_ly(data = dummy_data_1, x = ~x, y = ~col_1,
                 type = "scatter", mode = "lines+markers")
fig11
```



Combining Multiple Plots

```
fig12 <- plot_ly(data = dummy_data_1, x = ~x, y = ~col_1,
                 name = "First Plot",
                 type = "scatter", mode = "lines") %>%
  add_trace(y = ~col_2, name = "Second Plot",
            type = "scatter", mode = "lines+markers") %>%
  add_trace(y = ~col_4, name = "Third Plot",
            type = "scatter", mode = "markers")
fig12
```



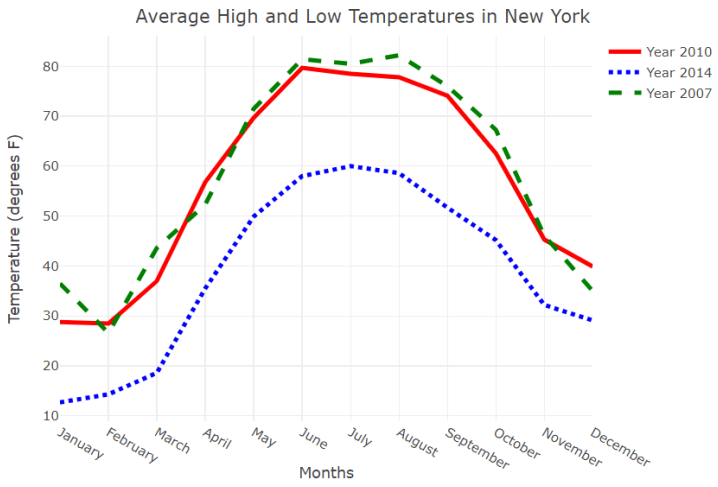
Dataset: dummy_data_2

	month	year_2007	year_2010	year_2014
1	January	36.5	28.8	12.7
2	February	26.6	28.5	14.3
3	March	43.6	37.0	18.6
4	April	52.3	56.8	35.5
5	May	71.5	69.7	49.9
6	June	81.4	79.7	58.0
7	July	80.5	78.5	60.0
8	August	82.2	77.8	58.6
9	September	76.0	74.1	51.7
10	October	67.3	62.6	45.2
11	November	46.1	45.3	32.2
12	December	35.0	39.9	29.1

Style- Multiple Line Plot

```
fig13 <- plot_ly(dummy_data_2, x = ~month, y = ~year_2010, name = "Year 2010",
                 type = "scatter", mode = "lines",
                 line = list(color = "red", width = 4))%>%
  add_trace(y = ~year_2014, name = "Year 2014",
            type = "scatter", mode = "lines",
            line = list(color = "blue", width = 4, dash = "dot")) %>%
  add_trace(y = ~year_2007, name = "Year 2007",
            type = "scatter", mode = "lines",
            line = list(color = "green", width = 4, dash = "dash")) %>%
  layout(title = "Average High and Low Temperatures in New York",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Temperature (degrees F)"))
```

```
fig13
```



Combining ggplot2 with Plotly

You can enhance static ggplot2 plots with interactivity by using ggplotly(). The ggplotly() function is used to convert the ggplot plot to a Plotly interactive plot.

```
fig14 <- dummy_data_1 %>%
  ggplot(aes(x = col_1, y = col_2))+
  geom_point()
fig14

ggplotly(fig14)
```

Note: See this plot on R

Interactive Data Visualization - Part 2

Main Code

```
plot_ly(data = dataframe, x = ~name_of_variable1,
                  y = ~name_of_variable2,
          color = ~name_variable3,
          symbol = ~name_variable3,
          type = "", mode = "",
          colors = pallete_vector)
```

Type	Mode	Type of Plot	Add on functions
type = “scatter”	mode = “markers”	Scatter Plot	add_trace() layout()
	mode = "lines"	Line Graph	
	mode = "lines+markers"	Line Graph with dots	
type = "histogram"		Histogram	add_histogram() layout()
type = "bar"		Bar Graph	add_trace() layout() Inside layout: barmode = "group" in layout barmode = "stack" in layout
type = "box"		Box Plot	add_trace

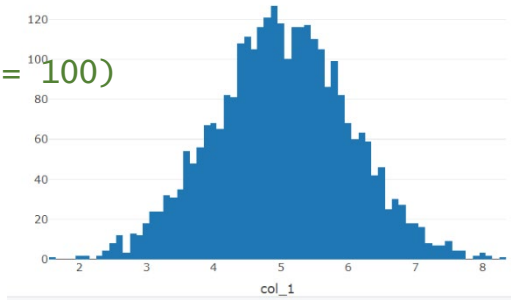
Histograms

Dataframe: data_1

	x	col_1	col_2	col_3	col_4
1	1	4.452418	3.6876016	0.052225403	-4.756294
2	2	2.955263	2.8928292	1.227396980	-3.771439
3	3	5.779916	3.6968157	2.321996332	-3.720551
4	4	4.724015	3.6440147	1.123346227	-3.463984
5	5	4.289810	2.4371923	0.136189320	-5.512061
6	6	3.477498	4.7305132	-0.492429713	-4.876484
7	7	4.626856	3.1279934	-0.239440013	-4.561630
8	8	2.612686	4.4751611	0.612236758	-4.645969
9	9	3.982926	1.9682513	-0.531182509	-5.214656
10	10	3.691051	2.6501433	0.786965740	-4.992421

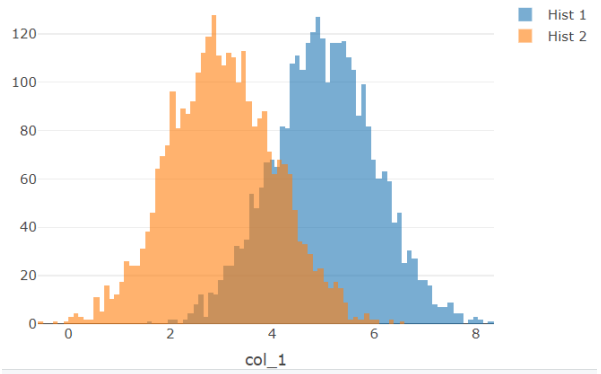
Simple Histogram

```
fig1 <- plot_ly(data = data_1,  x = ~col_1,
                type = "histogram", nbinsx = 100)
fig1
```



Overlaid Histogram (alpha controls the level of transparency)

```
fig2 <-  plot_ly(data = data_1, x = ~col_1,
                 type = "histogram", alpha = 0.6, name = "Hist 1") %>%
        add_histogram(x = ~col_2, name = "Hist 2") %>%
        layout(barmode = "overlay")
fig2
```



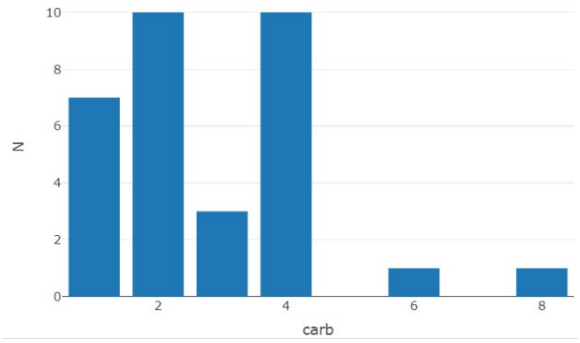
Bar Graphs

Dataframe: data_2

	carb	N
1	1	7
2	2	10
3	3	3
4	4	10
5	6	1
6	8	1

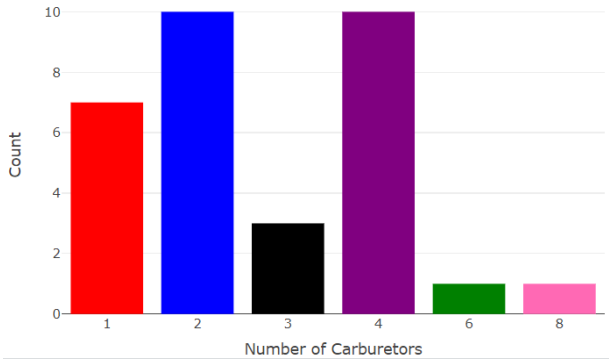
Simple Bar Graph

```
fig3 <- plot_ly(data = data_2,
                x = ~carb, y = ~N,
                type = "bar")
fig3
```



Changing colors in a Bar Graph

```
fig4 <- plot_ly(data = data_2,
                x = ~factor(carb), y = ~N,
                type = "bar",
                marker = list(color = c("red", "blue", "black",
                                        "purple", "green", "hotpink"))) %>%
  layout(xaxis = list(title = "Number of Carburetors"),
         yaxis = list(title = "Count"))
fig4
```

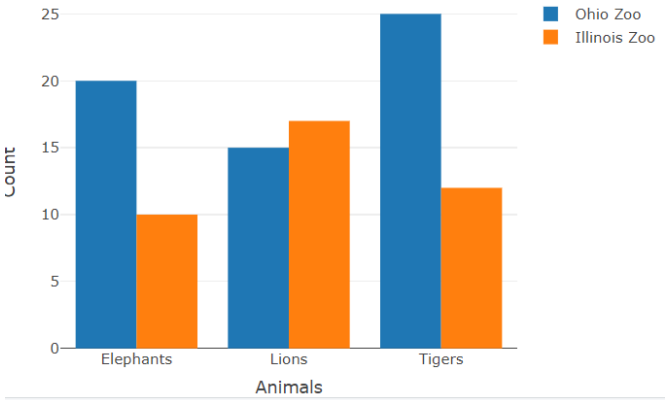


Dataframe: data_3

	Animals	OH_Zoo	IL_Zoo
1	Elephants	20	10
2	Lions	15	17
3	Tigers	25	12

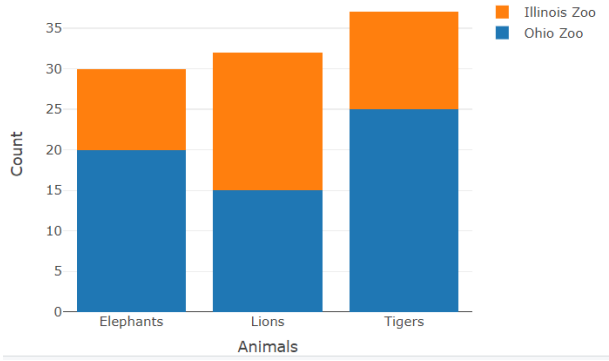
Grouped Bar Graph

```
fig5 <- plot_ly(data = data_3, x = ~Animals, y = ~OH_Zoo,
                type = "bar", name = "Ohio Zoo") %>%
        add_trace(y = ~IL_Zoo, name = "Illinois Zoo") %>%
        layout(yaxis = list(title = "Count"),
               barmode = "group")
fig5
```



Stacked Bar Graph

```
fig6 <- plot_ly(data = data_3, x = ~Animals, y = ~OH_Zoo,
                type = "bar", name = "Ohio Zoo") %>%
        add_trace(y = ~IL_Zoo, name = "Illinois Zoo") %>%
        layout(yaxis = list(title = "Count"),
               barmode = "stack")
fig6
```



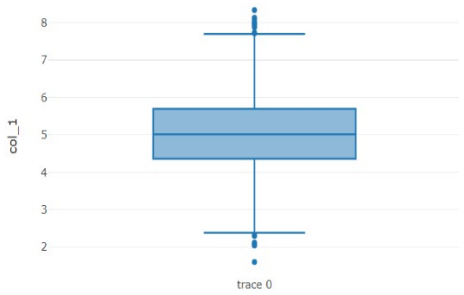
Boxplots

Dataframe: data_1

	x	col_1	col_2	col_3	col_4
1	1	4.452418	3.6876016	0.052225403	-4.756294
2	2	2.955263	2.8928292	1.227396980	-3.771439
3	3	5.779916	3.6968157	2.321996332	-3.720551
4	4	4.724015	3.6440147	1.123346227	-3.463984
5	5	4.289810	2.4371923	0.136189320	-5.512061
6	6	3.477498	4.7305132	-0.492429713	-4.876484
7	7	4.626856	3.1279934	-0.239440013	-4.561630
8	8	2.612686	4.4751611	0.612236758	-4.645969
9	9	3.982926	1.9682513	-0.531182509	-5.214656
10	10	3.691051	2.6501433	0.786965740	-4.992421

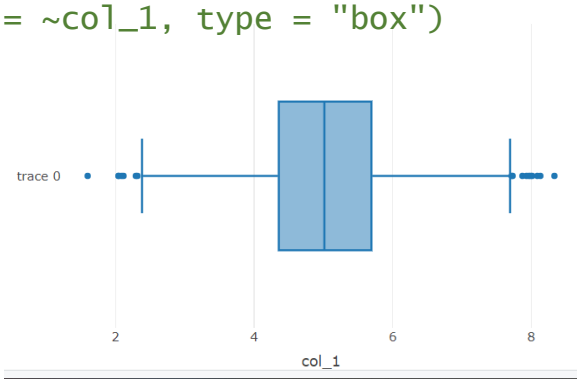
Vertical Boxplot

```
fig7 <- plot_ly(data = data_1, y = ~col_1, type = "box")
fig7
```



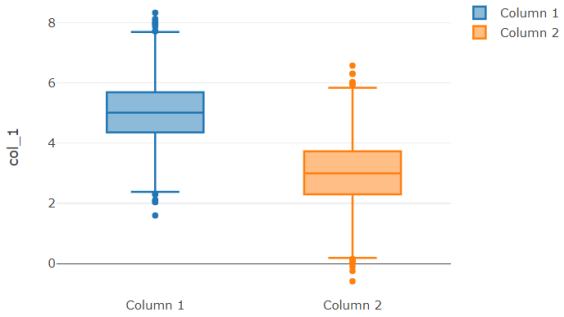
Horizontal Boxplot

```
fig8 <- plot_ly(data = data_1, x = ~col_1, type = "box")
fig8
```



Multiple Boxplots

```
fig9 <- plot_ly(data = data_1, y = ~col_1,
                type = "box", name = "Column 1") %>%
        add_trace(y = ~col_2, name = "Column 2")
fig9
```

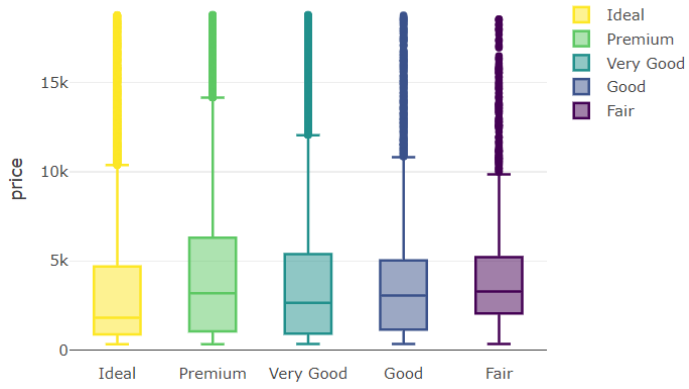


Multiple Boxplots by Categorical Variable

Dataframe: diamonds

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

```
fig10 <- plot_ly(data = diamonds,
                 y = ~price, color = ~cut, type = "box")
fig10
```



Interactive Tables with Datatable

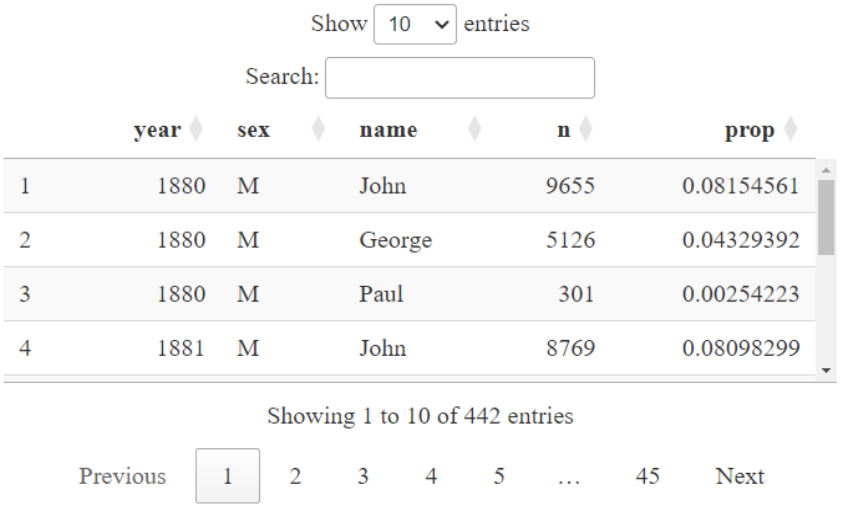
Dataframe: Beatles

	year	sex	name	n	prop
1	1880	M	John	9655	0.08154561
2	1880	M	George	5126	0.04329392
3	1880	M	Paul	301	0.00254223
4	1881	M	John	8769	0.08098299
5	1881	M	George	4664	0.04307272
6	1881	M	Paul	291	0.00268743

```
data("babynames")
Beatles <- babynames %>%
  filter(name %in% c("John", "Paul", "George", "Ringo") & sex == "M")
```

In this section, an interactive table is generated using the DT package. With the function `datatable()` we will turn a dataframe into an interactive table.

```
interactive_table <- datatable(Beatles, options = list(pageLength = 10))
interactive_table
```



Dygraph

This code generates an interactive time series plot showing the popularity of Beatles names over time. You can adjust the date range and explore the data dynamically.

Dygraphs is a powerful tool for visualizing time-dependent information.

- We use the Beatles dataset.
- We select the relevant columns, such as year, name, and prop (popularity).
- We use pivot_wider to reformat the data.
- dygraph(main = "Popularity of Beatles names over time") initializes a Dygraph plot and sets the title.
- dyRangeSelector(dateWindow = c("1900", "1980")) provides a range selector to focus on specific years.

```
fig11 <- Beatles %>%
  select(year, name, prop) %>%
  pivot_wider(names_from = name, values_from = prop) %>%
  dygraph(main = "Popularity of Beatles names over time") %>%
  dyRangeSelector(dateWindow = c("1900", "1980"))
fig11
```