Week 5:

Visualizing Univariate and Bivariate Data Distributions:

Exploratory Data Analysis (EDA) is a statistical approach used to summarize and visualize a dataset to gain insights and discover patterns, relationships, and anomalies in the data.

It is important to perform this analysis as we can have a closer look at the data before we chose to do further analysis on the data, or to use it on our project. It will also help us to choose a suitable subset of the data if needed.

1. Univariate Analysis:

This involves analyzing each variable to understand its distribution, central tendency, and spread. Visualizations include: histograms, density plots and box plots.

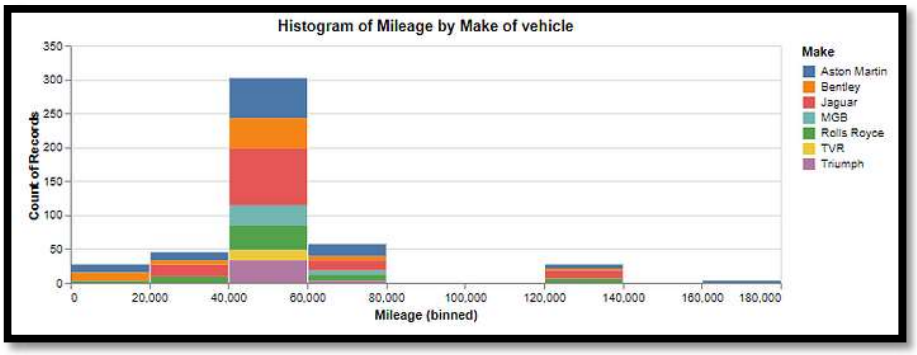
2. Bivariate Analysis:

This involves analyzing the relationship between two variables. Visualizations include: scatter plots and correlation matrices.

Univariate Analysis

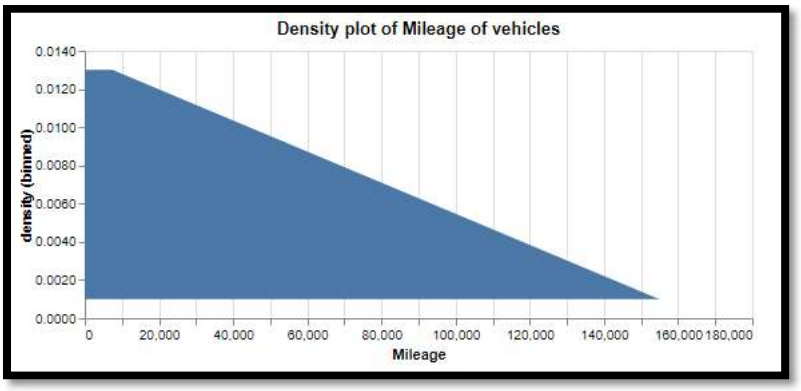
Histogram

```
using VegaLite
data |>
@vlplot(width=600,
    height=200,:bar, x={:Mileage, bin=true}, y="count()",
    title="Histogram of Mileage by Make of vehicle",
    config={
        title={
            color=:black
        }}, color =:Make )
```



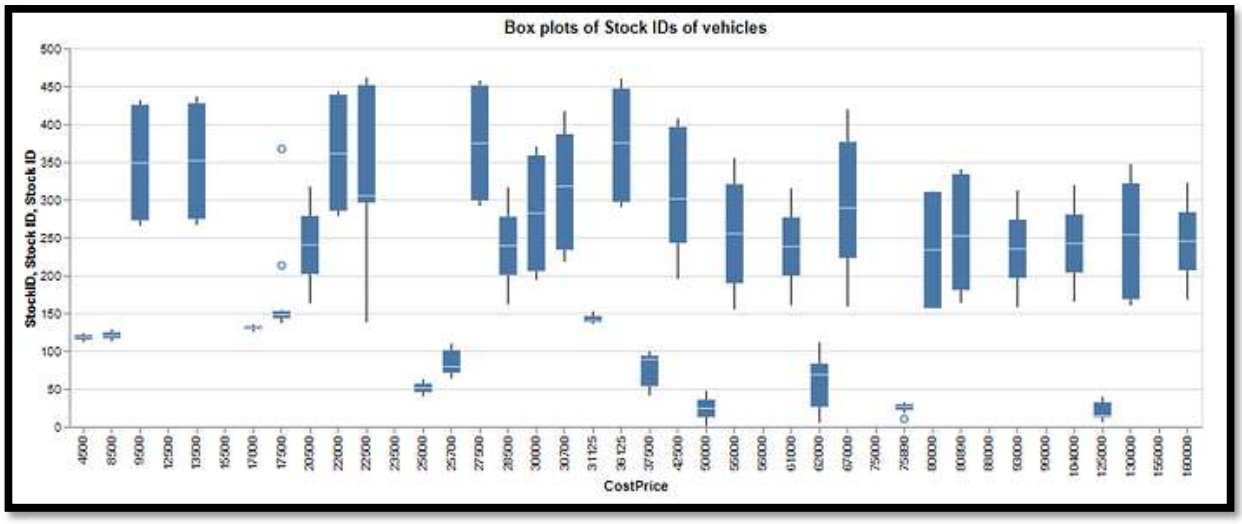
Density plot

```
using VegaLite
data |> @vlplot(
    width=500,
    height=200,
    :area,
    transform=[
        {density="Mileage",bandwidth=0.2}
    ],
   x={"value:q", title="Mileage"},
    y= {"density:q",bin=true}, title="Density plot of Mileage of vehicles",
    config={
        title={
            color=:black
        }}
)
```



Box plot

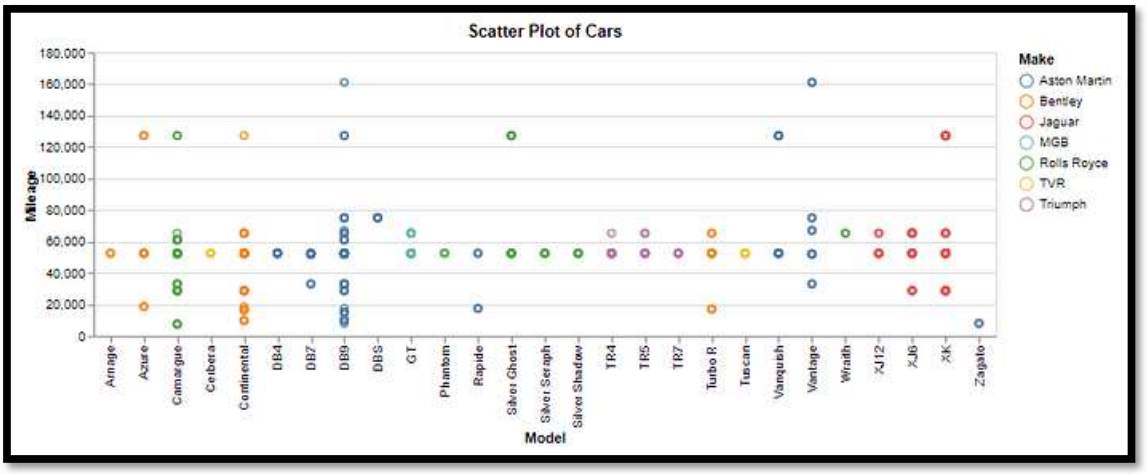
```
using VegaLite
data |> @vlplot(
    width=900,
    height=300,
    mark={:boxplot, extent=1.5},
    x="CostPrice:o",
    y={:StockID, axis={title="Stock ID"}},
    title="Box plots of Stock IDs of vehicles",
    config={
        title={
            color=:black
        }}
)
```



Bivariate Analysis:

Scatter plot

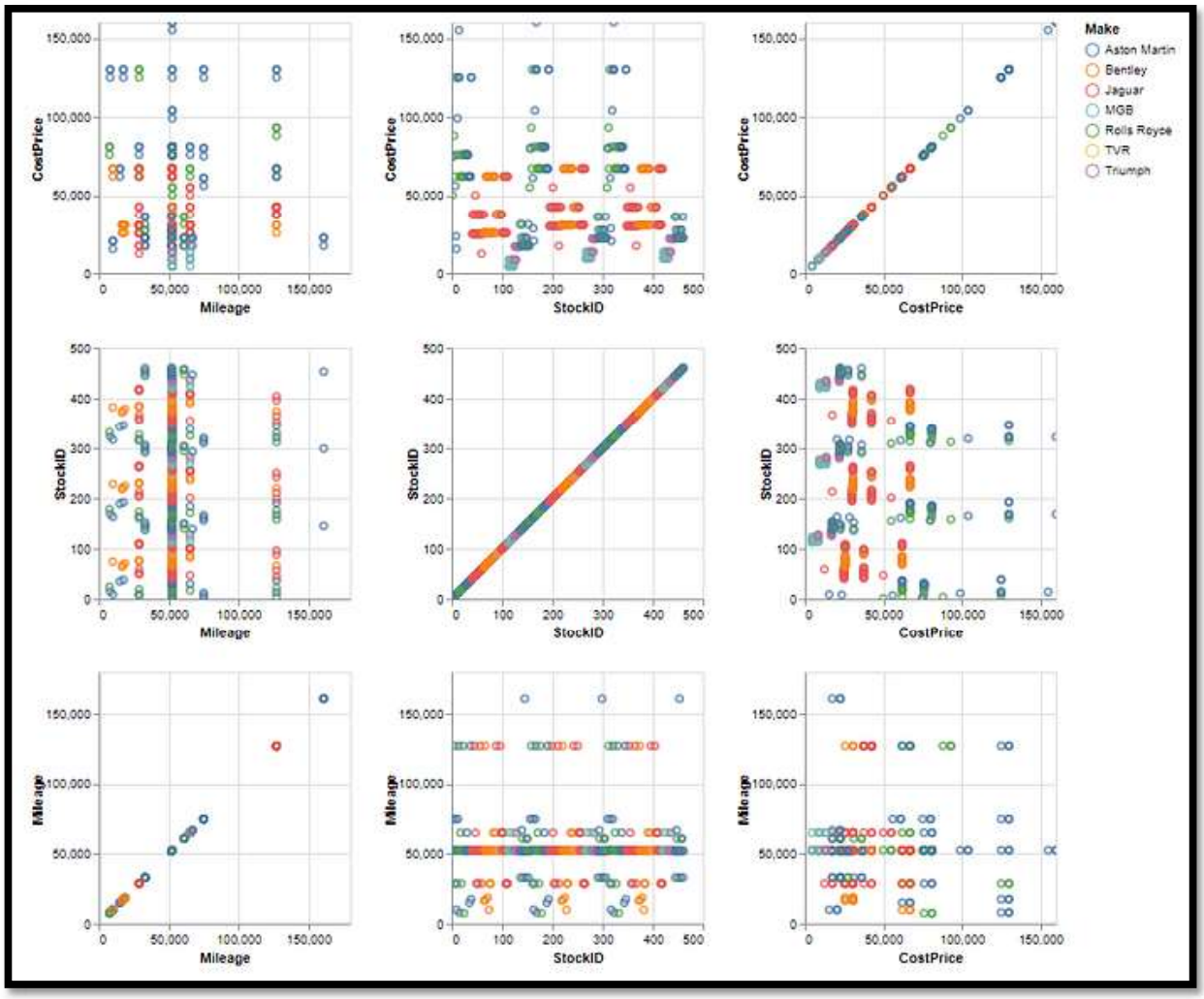
```
using VegaLite
data |>
@vlplot(width=700,
    height=220,:point,
    x=:Model,
    y=:Mileage,
    title="Scatter Plot of Cars",
    config={
        title={
            color=:black
        }},
    color=:Make)
```



Correlation plots

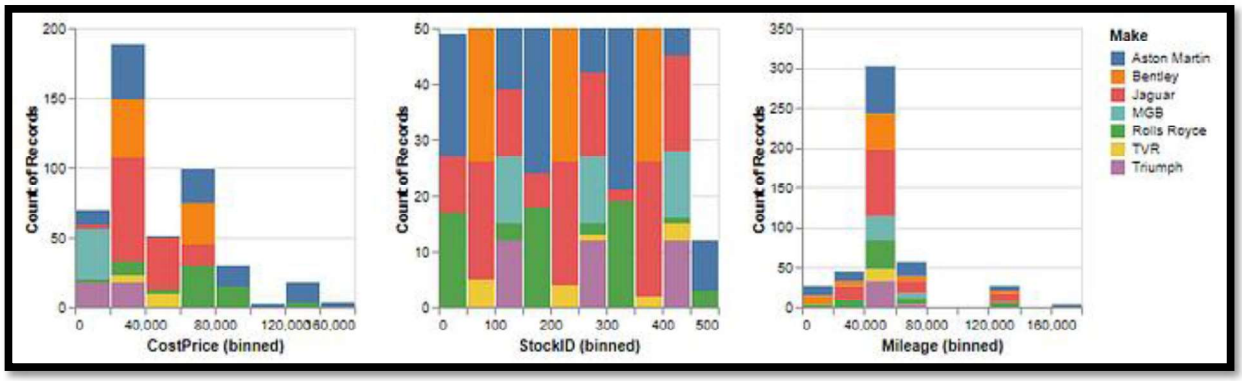
```
using VegaLite
data |>
@vlplot(
    repeat={
        row=[:CostPrice, :StockID, :Mileage],
        column=[:Mileage, :StockID, :CostPrice]
    }
) +
@vlplot(
    :point,
    selection={
        brush={
            type=:interval,
            resolve=:union,
            on="[mousedown[event.shiftKey], window:mouseup] > window:mousemove!",
            translate="[mousedown[event.shiftKey], window:mouseup] > window:mousemove!",
            zoom="wheel![event.shiftKey]"
        },
        grid={
            type=:interval,
            resolve=:global,
            bind=:scales,
```

```
            translate="[mousedown[!event.shiftKey], window:mouseup] > window:mousemove!",
            zoom="wheel![!event.shiftKey]"
        }
    },
    x={field={repeat=:column}, type=:quantitative},
    y={field={repeat=:row}, type=:quantitative},
    color={
        condition={
            selection=:brush,
            field=:Make,
            type=:nominal
        },
        value=:grey
    }
)
```

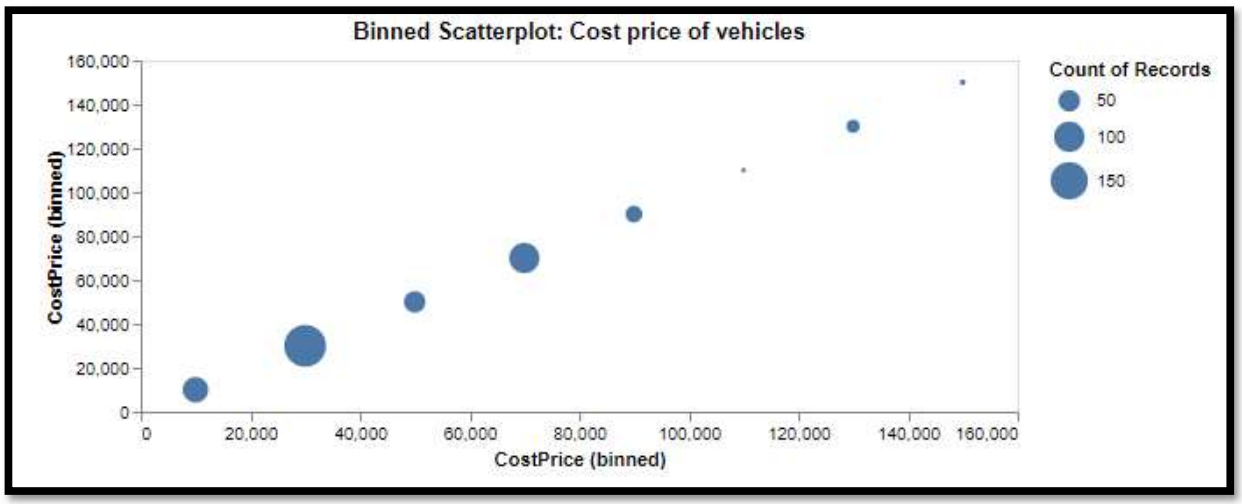


Some extra plots to get an idea of the data...

```
using VegaLite
data |>
@vlplot(repeat={column=[:CostPrice, :StockID, :Mileage]}) +
@vlplot(
    :bar,
    x={field={repeat=:column},bin=true,type=:quantitative},
    y="count()",
    color=:Make
)
```




```
using VegaLite
data |>
@vlplot(width=500,
    height=200,
    :circle,
    x={:CostPrice, bin={maxbins=10}},
    y={:CostPrice, bin={maxbins=10}},
    title="Binned Scatterplot: Cost price of vehicles",
    config={
        title={
            color=:black
        }} ,
    size="count()"
)
```

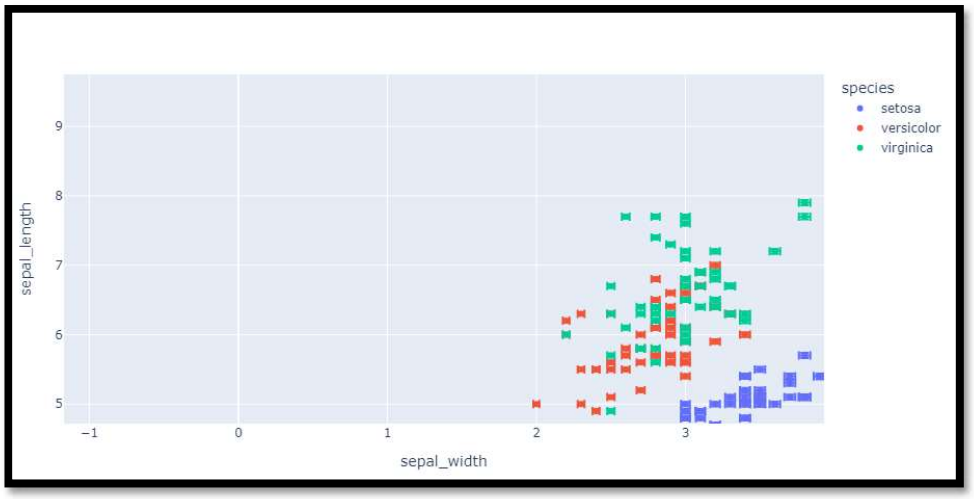


Week 6:

Visualizing Error Bars, Facets and Saving Plots

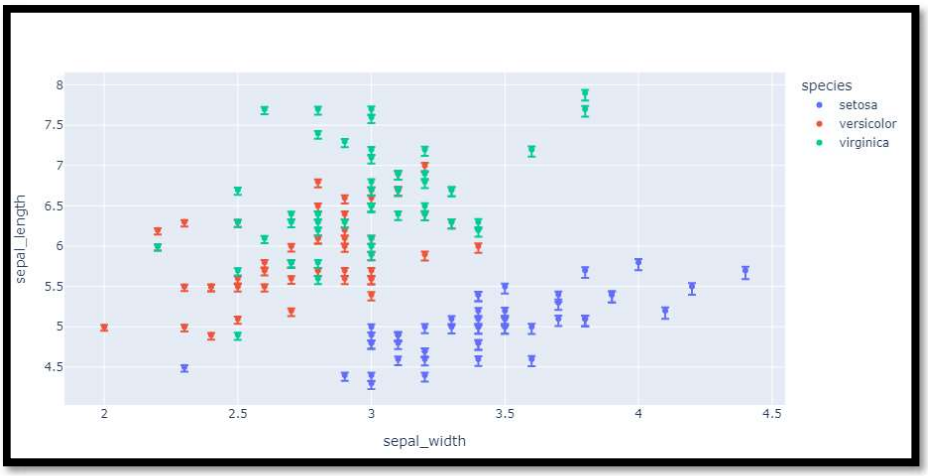
Error Bars

```
using PlotlyJS, CSV, DataFrames



df = dataset(DataFrame, "iris")



df[!, "e"] = df[!, "sepal_width"] ./ 100



plot(

    df, x=:sepal_width, y=:sepal_length, color=:species,

    mode="markers",

    error_x=attr(type="data", array=:e, visible=true),

    error_y=attr(type="data", array=:e, visible=true),

)
```



Asymmetric Error Bars

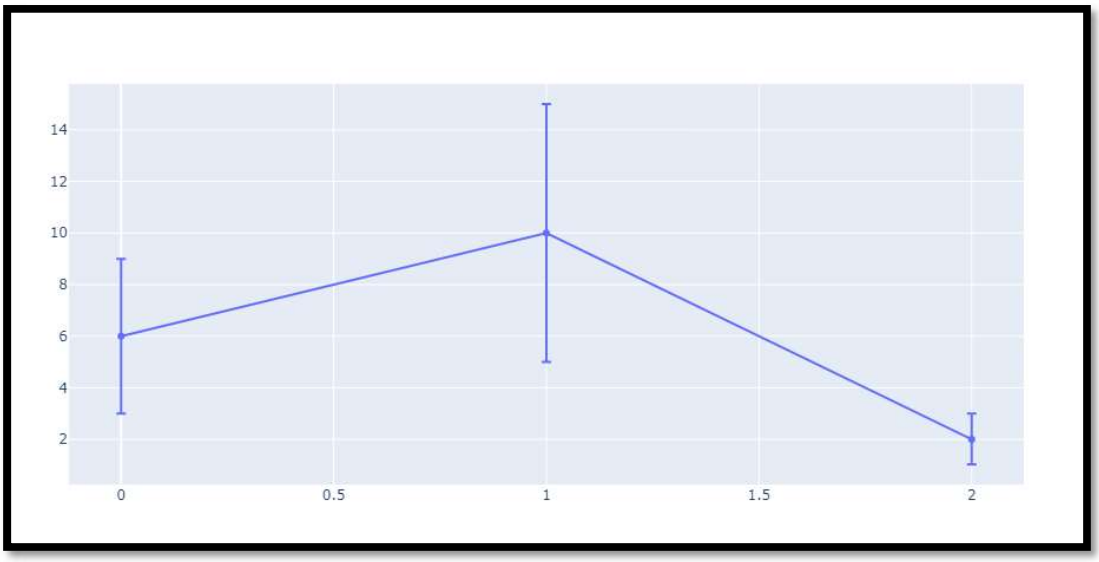
```
using PlotlyJS, CSV, DataFrames


df = dataset(DataFrame, "iris")


df[!, "e_plus"] = df[!, "sepal_width"] ./ 100

df[!, "e_minus"] = df[!, "sepal_width"] ./ 40

plot(

    df, x=:sepal_width, y=:sepal_length, color=:species,

    mode="markers",

    error_y=attr(type="data", array=:e_plus, arrayminus=:e_minus, visible=true),

)
```



Error Bars as a Percentage of the y Value

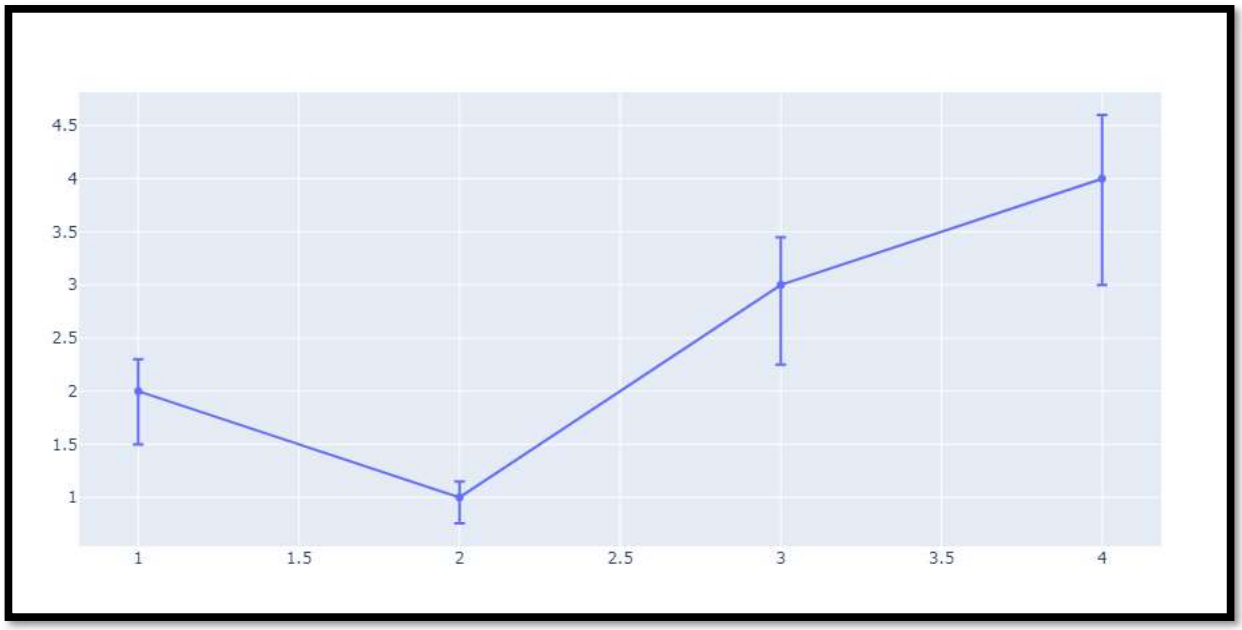
using PlotlyJS

```
plot(scatter(
    x=[0, 1, 2], y=[6, 10, 2], mode="markers+lines",
    error_y=attr(
        type="percent", # value of error bar given as percentage of y value
        value=50,
        visible=true
    )
))
```



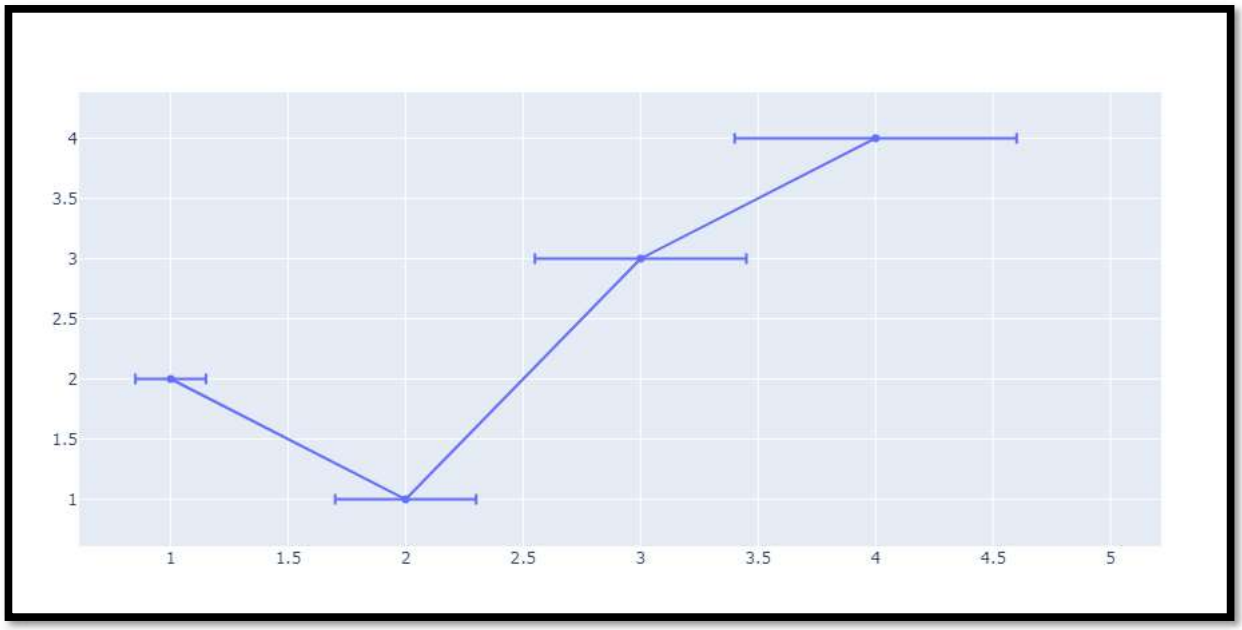
Asymmetric Error Bars with a Constant Offset

```
using PlotlyJS


plot(scatter(
    x=[1, 2, 3, 4],
    y=[2, 1, 3, 4],
    mode="markers+lines",
    error_y=attr(
        type="percent",
        symmetric=false,
        value=15,
        valueminus=25
    )
))
```



Horizontal Error Bars

```
using PlotlyJS


plot(scatter(
    x=[1, 2, 3, 4],
    y=[2, 1, 3, 4],
    mode="markers+lines",
    error_x=attr(
        type="percent",
        value=15
    )
))
```



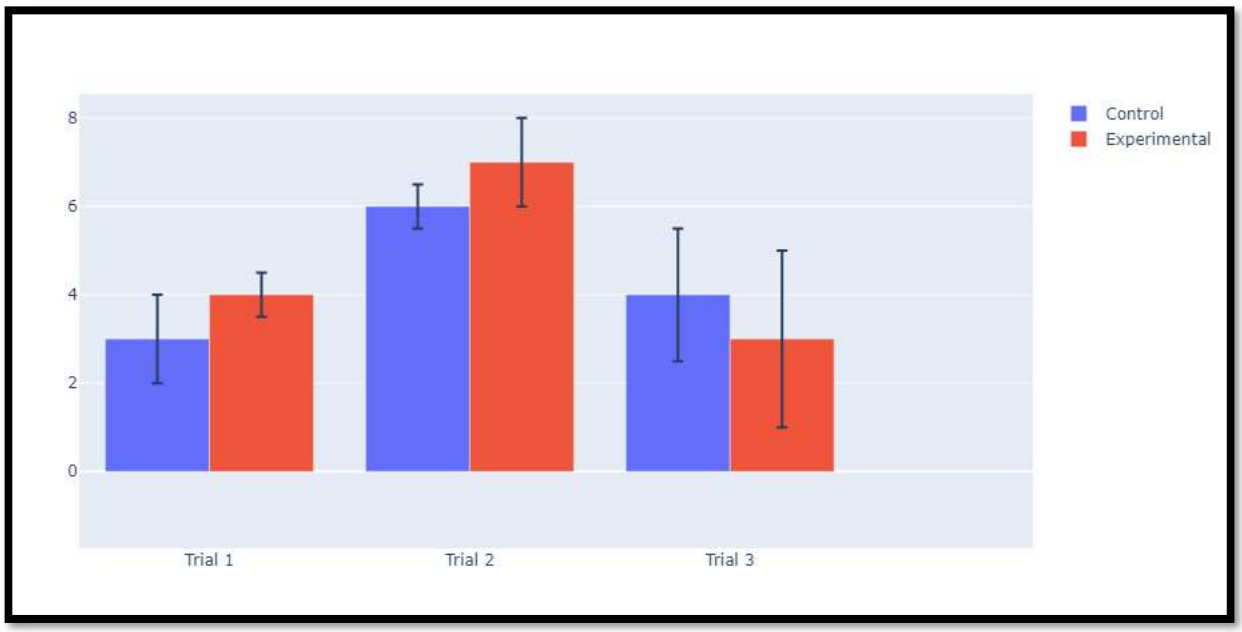
Bar Chart with Error Bars

using PlotlyJS

```
plot([
    bar(
        name="Control",
        x=["Trial 1", "Trial 2", "Trial 3"], y=[3, 6, 4],
        error_y=attr(type="data", array=[1, 0.5, 1.5])
    ),
    bar(
        name="Experimental",
        x=["Trial 1", "Trial 2", "Trial 3"], y=[4, 7, 3],
        error_y=attr(type="data", array=[0.5, 1, 2])
```

)

])



Colored and Styled Error Bars

using PlotlyJS

```
x_theo = range(-4, stop=4, length=100)

sincx = sinc.(x_theo)

x = [-3.8, -3.03, -1.91, -1.46, -0.89, -0.24, -0.0, 0.41, 0.89, 1.01, 1.91, 2.28, 2.79, 3.56]

y = [-0.02, 0.04, -0.01, -0.27, 0.36, 0.75, 1.03, 0.65, 0.28, 0.02, -0.11, 0.16, 0.04, -0.15]


trace1 = scatter(

    x=x_theo, y=sincx,

    name="sinc(x)"
```

```
)


trace2 = scatter(
    x=x, y=y,
    mode="markers",
    name="measured",
    error_y=attr(
        type="constant",
        value=0.1,
        color="purple",
        thickness=1.5,
        width=3,
    ),
    error_x=attr(
        type="constant",
        value=0.2,
        color="purple",
        thickness=1.5,
        width=3,
    ),
    marker=attr(color="purple", size=8)
)
```

plot([trace1, trace2])