Visualization in Bokeh

Neba Nfonsang

[1]:

```
import pandas as pd
import numpy as np

# import functions from bokeh modules
from bokeh.plotting import figure
from bokeh.io import output_notebook, show, curdoc
from bokeh.models import ColumnDataSource, Range1d, LabelSet
from bokeh.models.widgets import Slider, TextInput, Select
from bokeh.layouts import row, widgetbox, gridplot, column
output_notebook()
```

```
BokehJS 2.4.2 successfully loaded.
```

2]:

```
# Load a DSS dataset as a Pandas dataframe
url =
"https://vincentarelbundock.github.io/Rdatasets/csv/carData/Salaries.csv"

salary_df= pd.read_csv(url).iloc[:,1:]
cols = ["rank", "discipline", "yrs_since_phd", "yrs_service",
"gender", "salary"]
salary_df.columns = cols
salary_df.head()
```

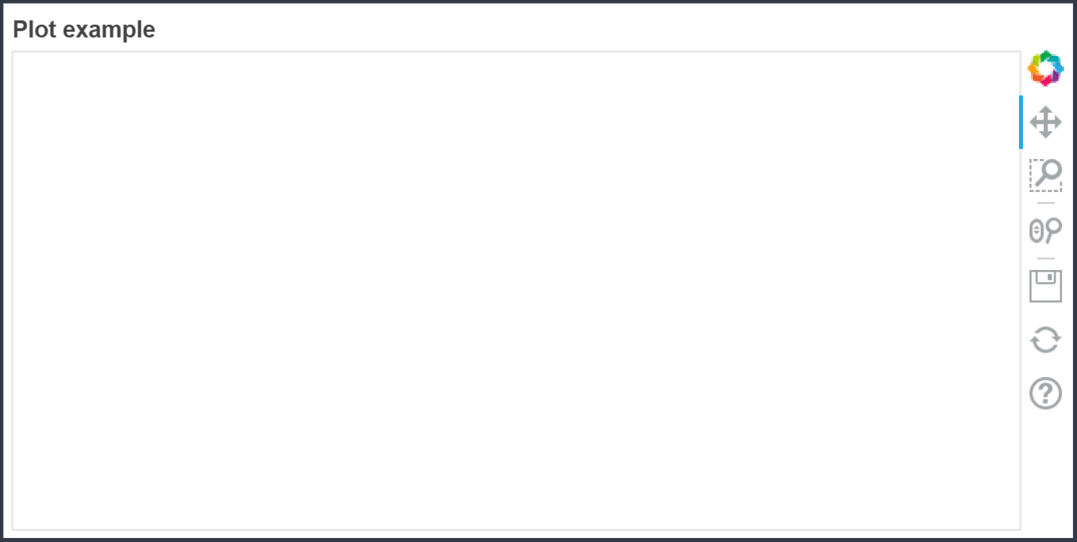
	rank	discipline	yrs_since_phd	yrs_service	gender	salary
0	Prof	B	19	18	Male	139750
1	Prof	B	20	16	Male	173200
2	AsstProf	B	4	3	Male	79750
3	Prof	B	45	39	Male	115000
4	Prof	B	40	41	Male	141500

Setting up an Empty Figure

3]:

```
# initialize an empty figure or plot
fig = figure(title="Plot example",
            width=600,
            height=300,
            x_axis_label="x label",
            y_axis_label="y label")
show(fig)
```

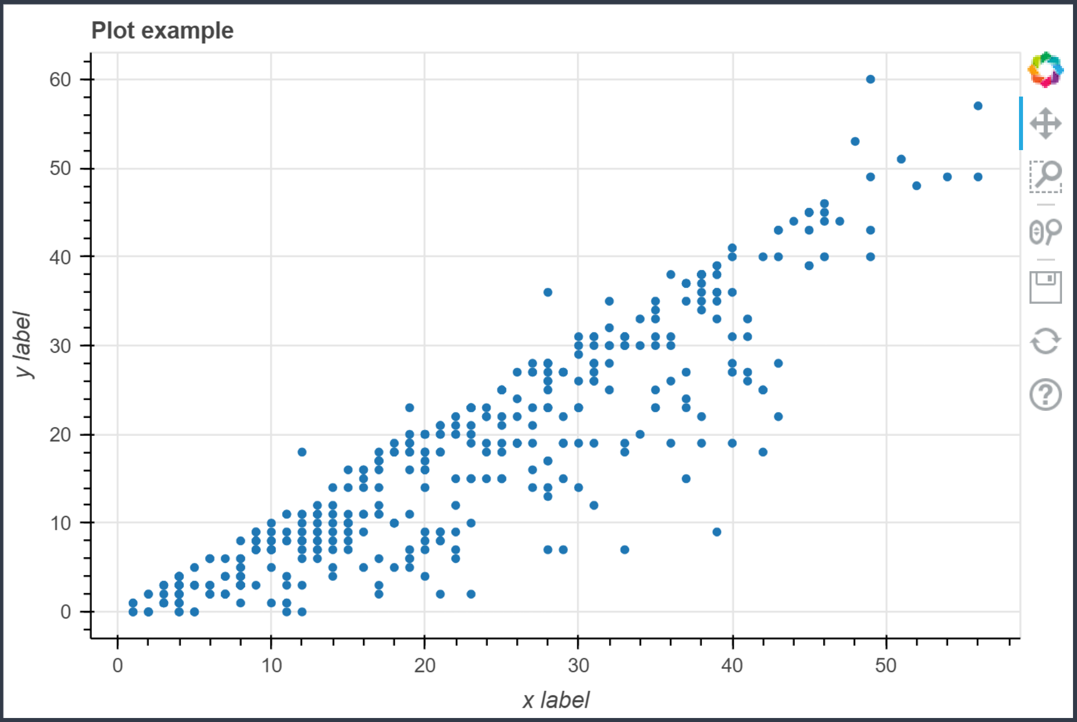
```
WARNING:bokeh.core.validation.check:W-1000
(MISSING_RENDERERS): Plot has no renderers:
Figure(id='1003', ...)
```



A Scatter Plot

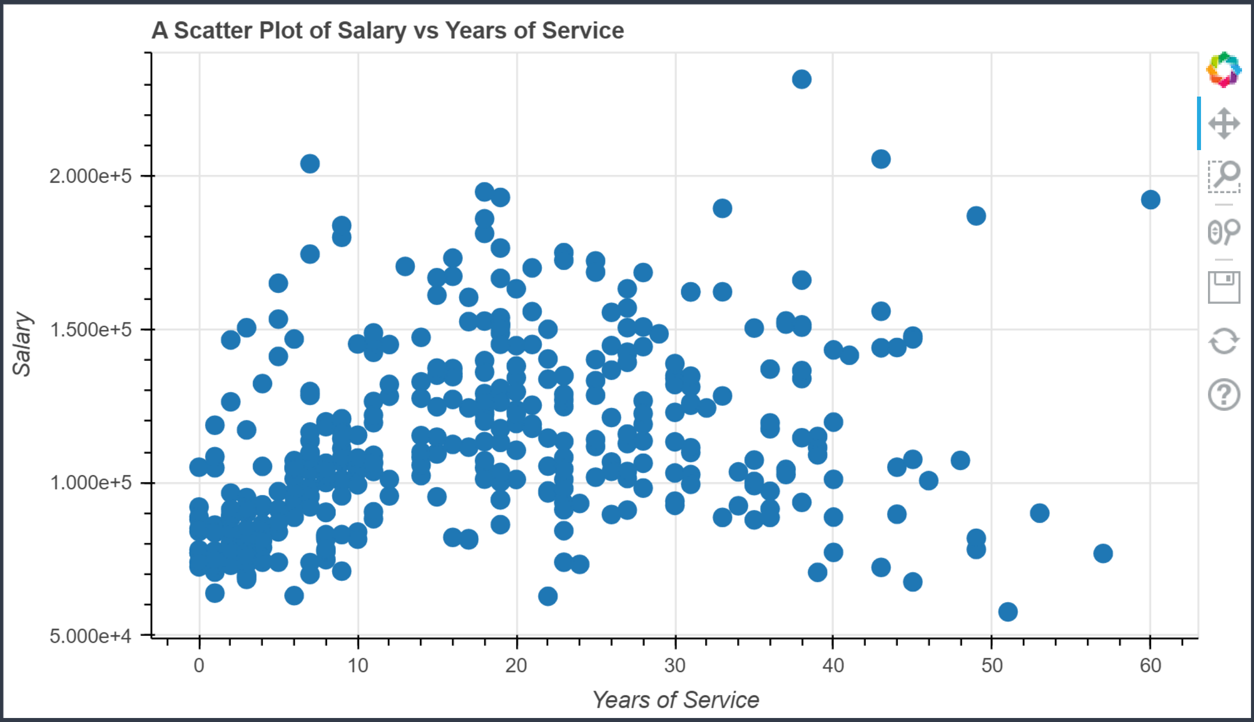
4]:

```
# initialize the figure or plot
fig = figure(title="Plot example",
             width=600,
             height=400,
             x_axis_label="x label",
             y_axis_label="y label")

fig.scatter(x=salary_df["yrs_since_phd"],
y=salary_df["yrs_service"])

show(fig)
```



5]:

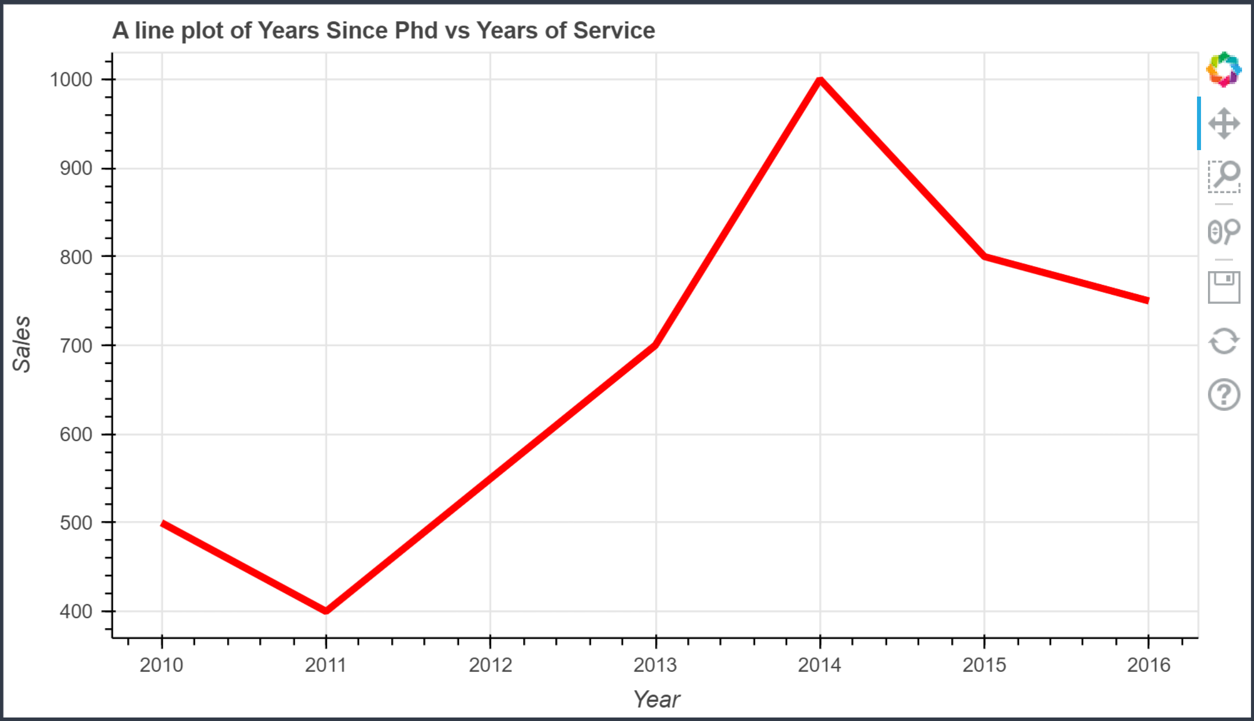
```python
# initialize the figure or plot
fig = figure(title="A Scatter Plot of Salary vs Years of Service",
            width=700,
            height=400,
            x_axis_label="Years of Service",
            y_axis_label="Salary")

fig.circle(x=salary_df["yrs_service"], y=salary_df["salary"],
size=10)

show(fig)
```



Line Plot

6]:

```
# initialize the figure or plot
years = [2010, 2011, 2013, 2014, 2015, 2016]
sales = [500, 400, 700, 1000, 800, 750]


fig = figure(title="A line plot of Years Since Phd vs Years of
Service",
            width=700,
            height=400,
            x_axis_label="Year",
            y_axis_label="Sales")

fig.line(x=years, y=sales, line_width=4, color="red")
show(fig)
```



A Plot of Categorical Data

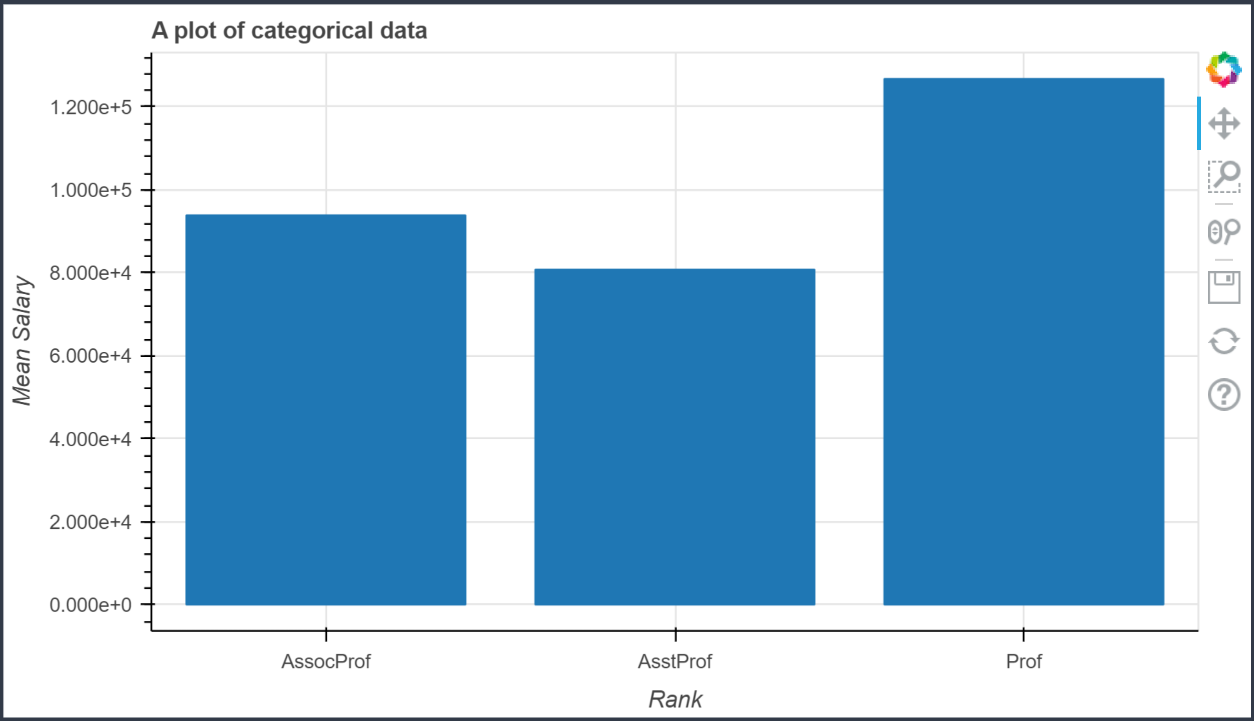
7]:

```
sal_by_rank = salary_df.groupby(by="rank", as_index=False)
["salary"].mean()
sal_by_rank
```

	rank	salary
0	AssocProf	93876.437500
1	AsstProf	80775.985075
2	Prof	126772.109023

8]:

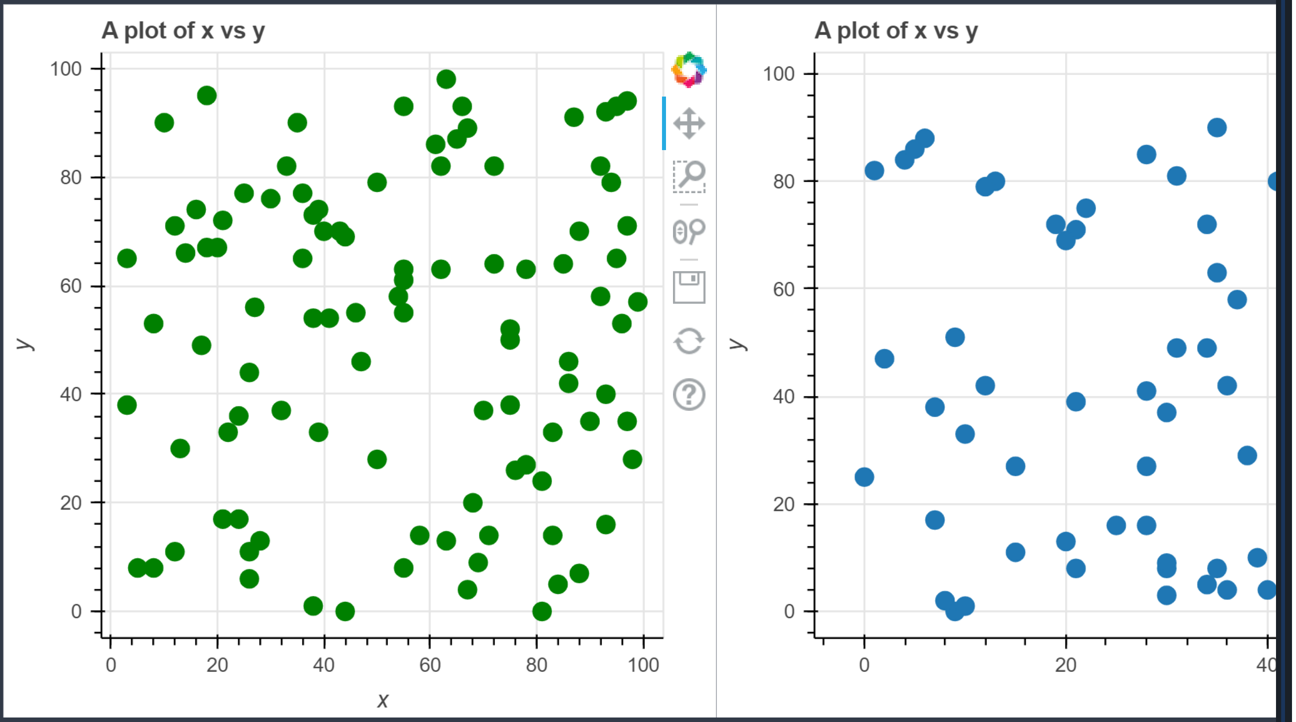
```python
fig = figure(title="A plot of categorical data",
            width=700,
            height=400,
            x_axis_label="Rank",
            y_axis_label="Mean Salary",
            x_range=sal_by_rank["rank"].values)

fig.vbar(x=sal_by_rank["rank"], width=0.8,
top=sal_by_rank["salary"])
show(fig)
```



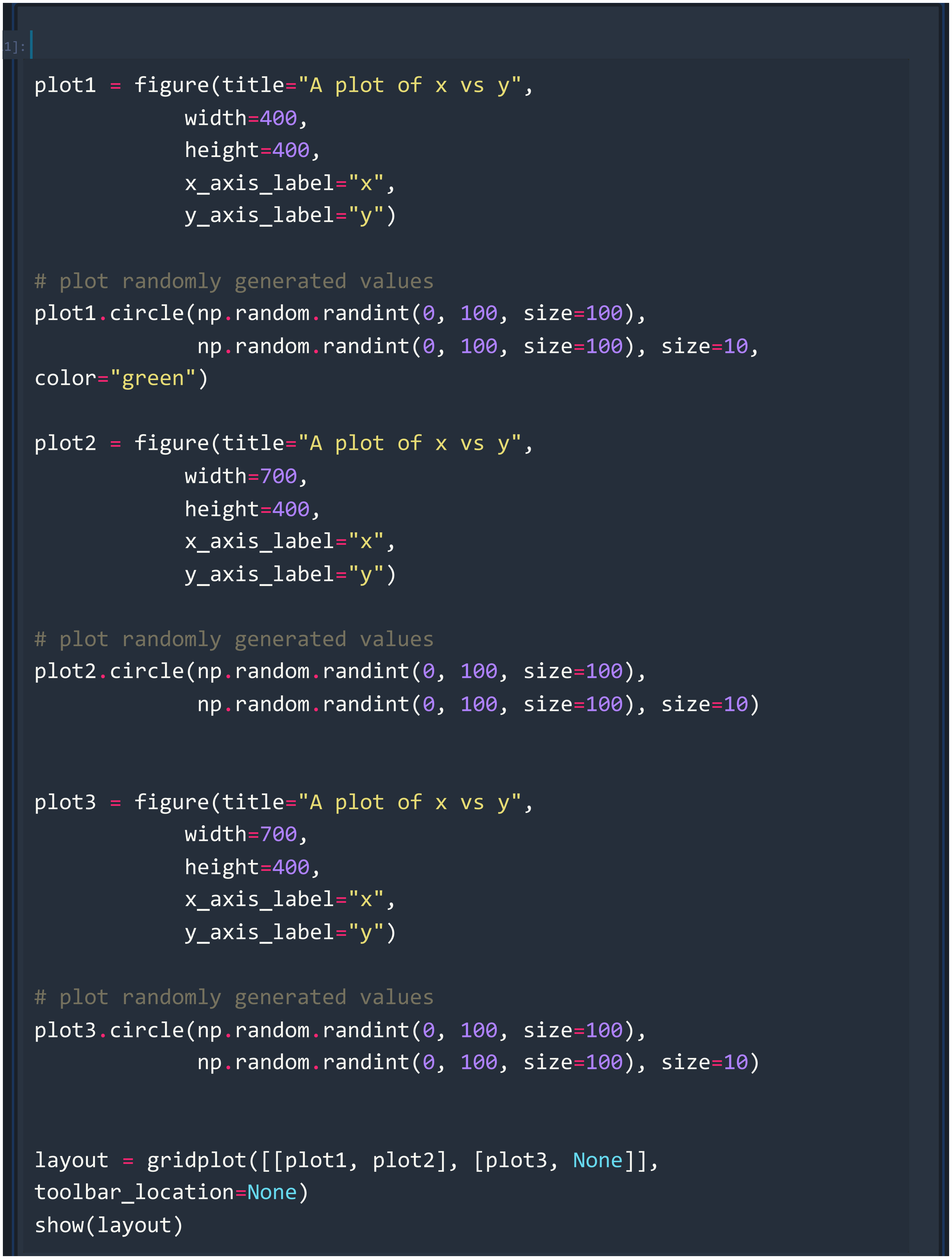
Layout

0]:

```
plot1 = figure(title="A plot of x vs y",
            width=400,
            height=400,
            x_axis_label="x",
            y_axis_label="y")

# plot randomly generated values
plot1.circle(np.random.randint(0, 100, size=100),
             np.random.randint(0, 100, size=100), size=10,
color="green")

plot2 = figure(title="A plot of x vs y",
            width=700,
            height=400,
            x_axis_label="x",
            y_axis_label="y")

# plot randomly generated values
plot2.circle(np.random.randint(0, 100, size=100),
             np.random.randint(0, 100, size=100), size=10)


layout = row(plot1, plot2)
show(layout)
```

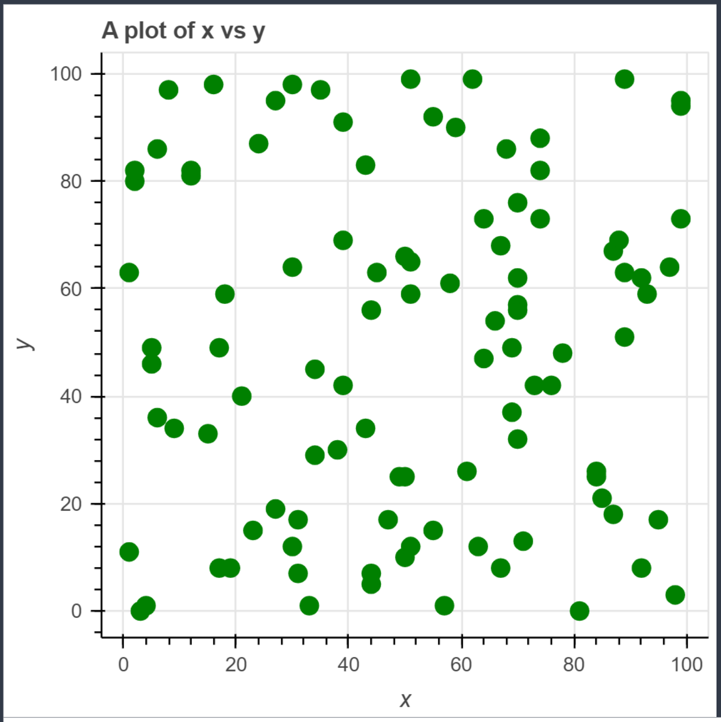

A plot of x vs y
100
80
60
40
20
0
y
0
20
40
60
80
100
x
A plot of x vs y
100
80
60
40
20
0
y
0
20
40

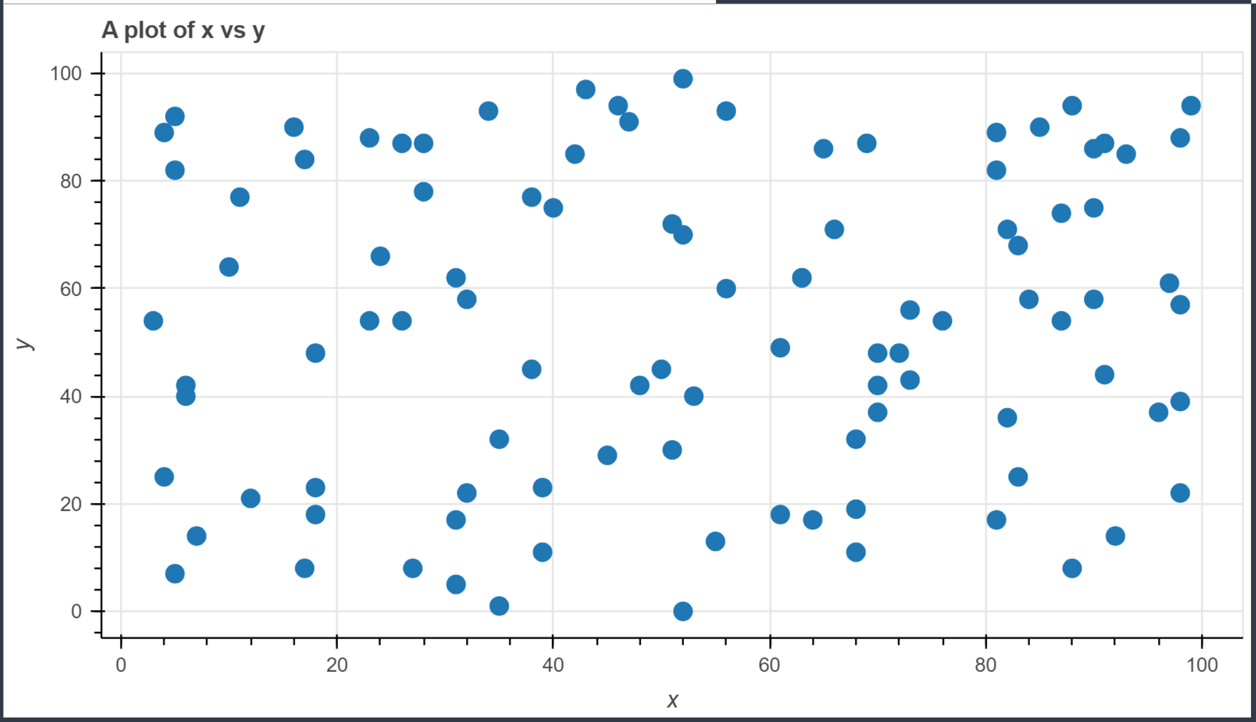
Gridplot



```
1]:
```

```python
plot1 = figure(title="A plot of x vs y",
            width=400,
            height=400,
            x_axis_label="x",
            y_axis_label="y")

# plot randomly generated values
plot1.circle(np.random.randint(0, 100, size=100),
             np.random.randint(0, 100, size=100), size=10,
color="green")

plot2 = figure(title="A plot of x vs y",
            width=700,
            height=400,
            x_axis_label="x",
            y_axis_label="y")

# plot randomly generated values
plot2.circle(np.random.randint(0, 100, size=100),
             np.random.randint(0, 100, size=100), size=10)


plot3 = figure(title="A plot of x vs y",
            width=700,
            height=400,
            x_axis_label="x",
            y_axis_label="y")

# plot randomly generated values
plot3.circle(np.random.randint(0, 100, size=100),
             np.random.randint(0, 100, size=100), size=10)


layout = gridplot([[plot1, plot2], [plot3, None]],
toolbar_location=None)
show(layout)
```


A plot of x vs y
y
x
0
20
40
60
80
100


A plot of x vs y
y
x
0
20
40
60
80
100

Configuration Tools

Here are five main tools as seen on the right side of the plot with the following names

- PanTool: pan ==> used to drag plot around
- BoxZoomTool: box_zoom ==> allows you to select a portion of the plot, then zoom into that
- WheelZoomTool: wheel_zoom ==> used to zoom the plot through scrolling
- Save: save ==> allows you to save the plot
- Reset: reset ==> reset to clear any action you have taken to go back to the original plot

Tools could be grouped as follows: Pan/drag tools

- pan
- boox_select
- box_zoom
- lasso_select

Click/tap tools

- poly_select
- tap

Scroll/pinch tools

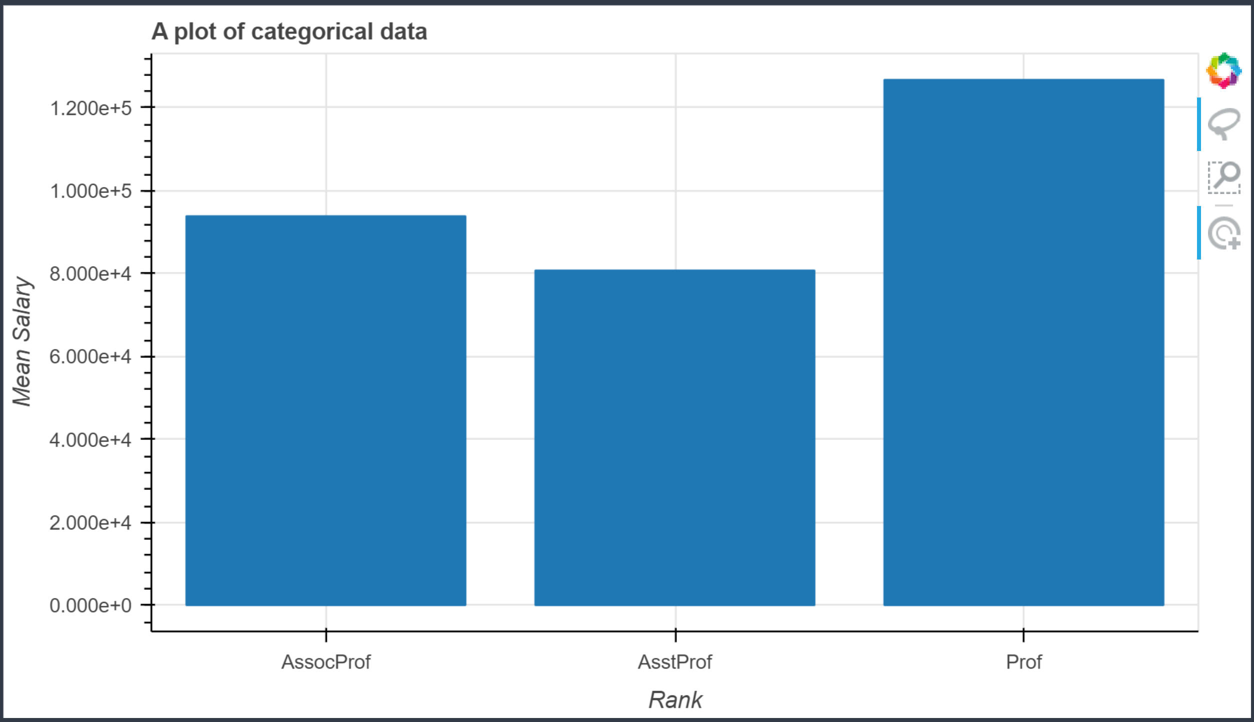
- wheel_zoom
- xwheel_pan
- ywheel_pan

Inspectors

- croshair
- hover

Customize the Tools

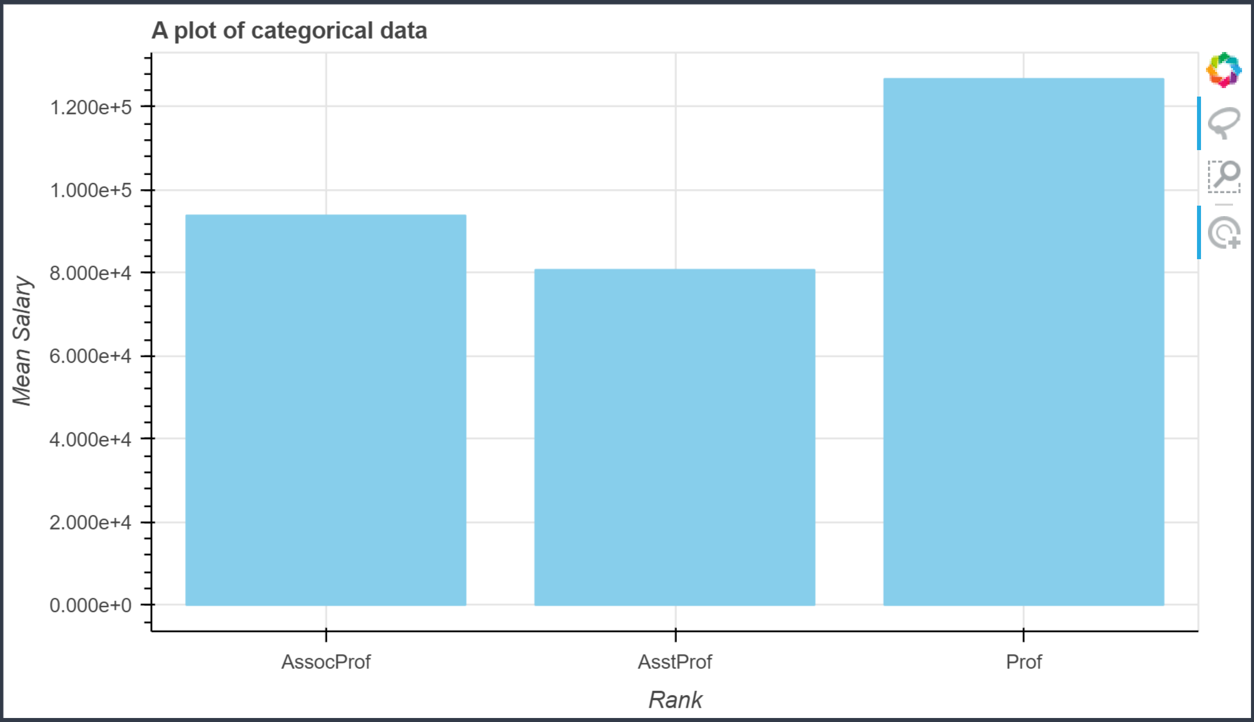
```
tools = ["lasso_select", "box_zoom", "tap"]
fig = figure(title="A plot of categorical data",
            width=700,
            height=400,
            x_axis_label="Rank",
            y_axis_label="Mean Salary",
            x_range=sal_by_rank["rank"].values,
            tools=tools)

fig.vbar(x=sal_by_rank["rank"], width=0.8,
top=sal_by_rank["salary"])
show(fig)
```



Plot using Data Source

3]:

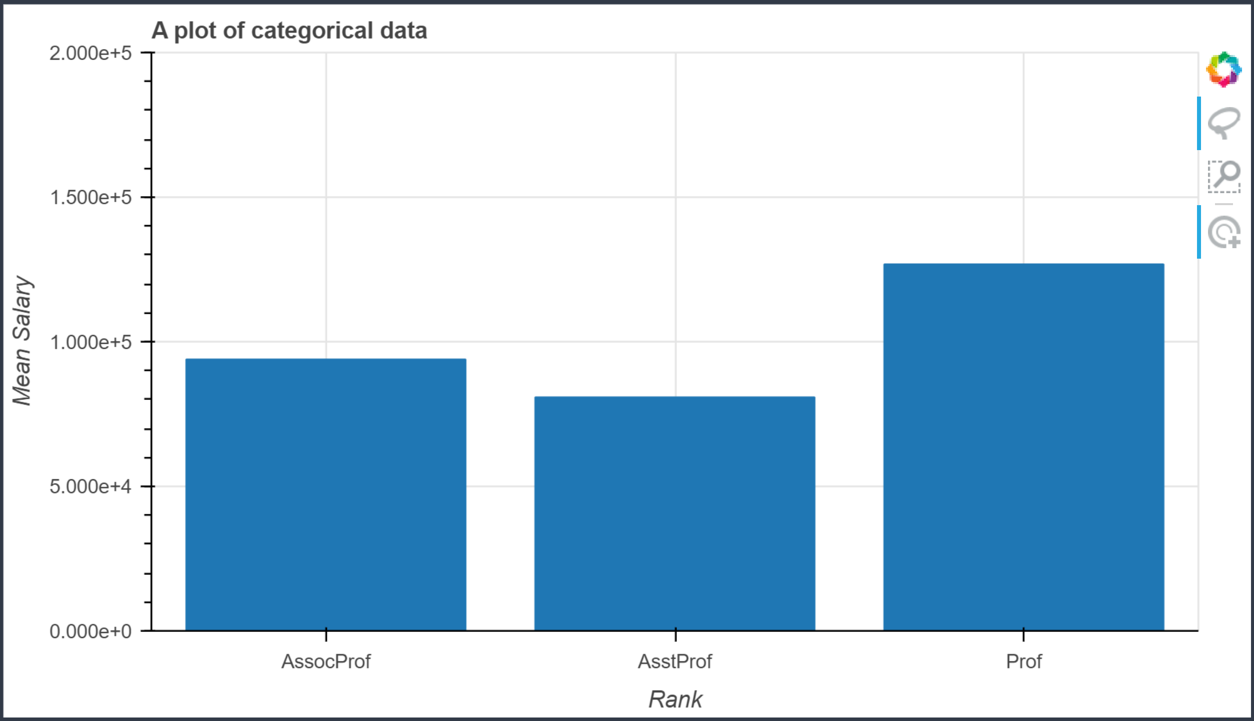
```python
fig = figure(title="A plot of categorical data",
            width=700,
            height=400,
            x_axis_label="Rank",
            y_axis_label="Mean Salary",
            x_range=sal_by_rank["rank"].values,
            tools=tools)

source = ColumnDataSource(sal_by_rank)

fig.vbar(x="rank", width=0.8, top="salary", source=source,
color="skyblue")
show(fig)
```



]:

4]:

```
fig = figure(title="A plot of categorical data",
             width=700,
             height=400,
             x_axis_label="Rank",
             y_axis_label="Mean Salary",
             x_range=sal_by_rank["rank"].values,
             tools=tools)

source = ColumnDataSource(sal_by_rank)

# set the range of the y axis
fig.y_range = Range1d(0, 200000)
fig.vbar(x="rank", width=0.8, top="salary", source=source)
show(fig)
```



Interactive Visualization

Note: bokeh server must be used when working with interactive widgets. The primary purpose of the bokeh server is to synchronize data between the underlying Python environment and the BokehJS library running in the browser.

]:

```
# # parameters/inputs
# input_dataset = "salary_data_prepared"
# x_column = "rank"
# y_column = "salary"
# discipline_column = "discipline"
# gender_column = "gender"
# service_column = "yrs_service"

# # create the plot
# tools = ["lasso_select", "box_zoom", "tap", "crosshair",
#          "pan", "reset", "save", "wheel_zoom"]

# plot = figure(title="Mean " + y_column + " by " + x_column,
#               width=500,
#               height=400,
#               x_axis_label=x_column.title(),
#               y_axis_label="Mean " + y_column.title(),
#               x_range=salary_df[x_column].values,
#               tools=tools)

# plot.y_range = Range1d(0, 200000) ## set y range
# ## label bar plots with y data
# labels = LabelSet(x=x_column, y=y_column, text=y_column,
level='glyph',
#          x_offset=-13.5, y_offset=0, source=source,
render_mode='canvas')

# plot.vbar(x=x_column, width=0.8, top=y_column, source=source)

# plot.add_layout(labels)

# # set up the widget
# title_text = TextInput(title="Title", value="Mean " + y_column + "
by " + x_column)

# service_data = salary_df[service_column].values
# min_yrs_service = Slider(title="Min Years in Service",
#                            value=min(service_data),
start=min(service_data),
#                            end=max(service_data), step=1)

# max_yrs_service = Slider(title="Max Years in Service",
#                            value=max(service_data),
start=min(service_data),
#                            end=max(service_data), step=1)

# gender_categories = salary_df[gender_column].unique().tolist()
# gender_categories.insert(0, "All")
# gender_cat = Select(title="Gender Category",
value=gender_categories[0],
#                       options=gender_categories)


# discipline_categories =
salary_df[discipline_column].unique().tolist()
# discipline_categories.insert(0, "All")
# discipline_cat = Select(title="Discipline Category",
value=discipline_categories[0],
#                          options=discipline_categories)

# #show(title_text)

# # set up update functions and callbacks

# def update_title(attrname, old, new):
#     plot.title.text = title_text.value # get current title text
value
# title_text.on_change("value", update_title) # trigger the function
upon changes in title_text


# def update_data(attrname, old, new):
#     selected = salary_df[(service_data>=min_yrs_service.value) &
#                          (service_data<=max_yrs_service.value)]
#     if (gender_cat.value!=gender_categories[0]):
#         selected = 
selected[selected[gender_column]==gender_cat.value]

#     if (discipline_cat.value!=discipline_categories[0]):
#         selected = 
selected[selected[discipline_column]==discipline_cat.value]

#     df = selected.groupby(by=x_column, as_index=False)
[y_column].mean()
#     df = df.round(2)
#     source.data = ColumnDataSource.from_df(df)



# for w in [min_yrs_service, max_yrs_service, gender_cat, 
discipline_cat]:
#     w.on_change("value", update_data)

# # set up layout and add to document
# menu = column(title_text, min_yrs_service, max_yrs_service, 
gender_cat, discipline_cat)
# layout = column(plot, menu)
# curdoc().add_root(layout)
# show(layout)
```