Line Graph in Plotly

In [1]:
```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
```

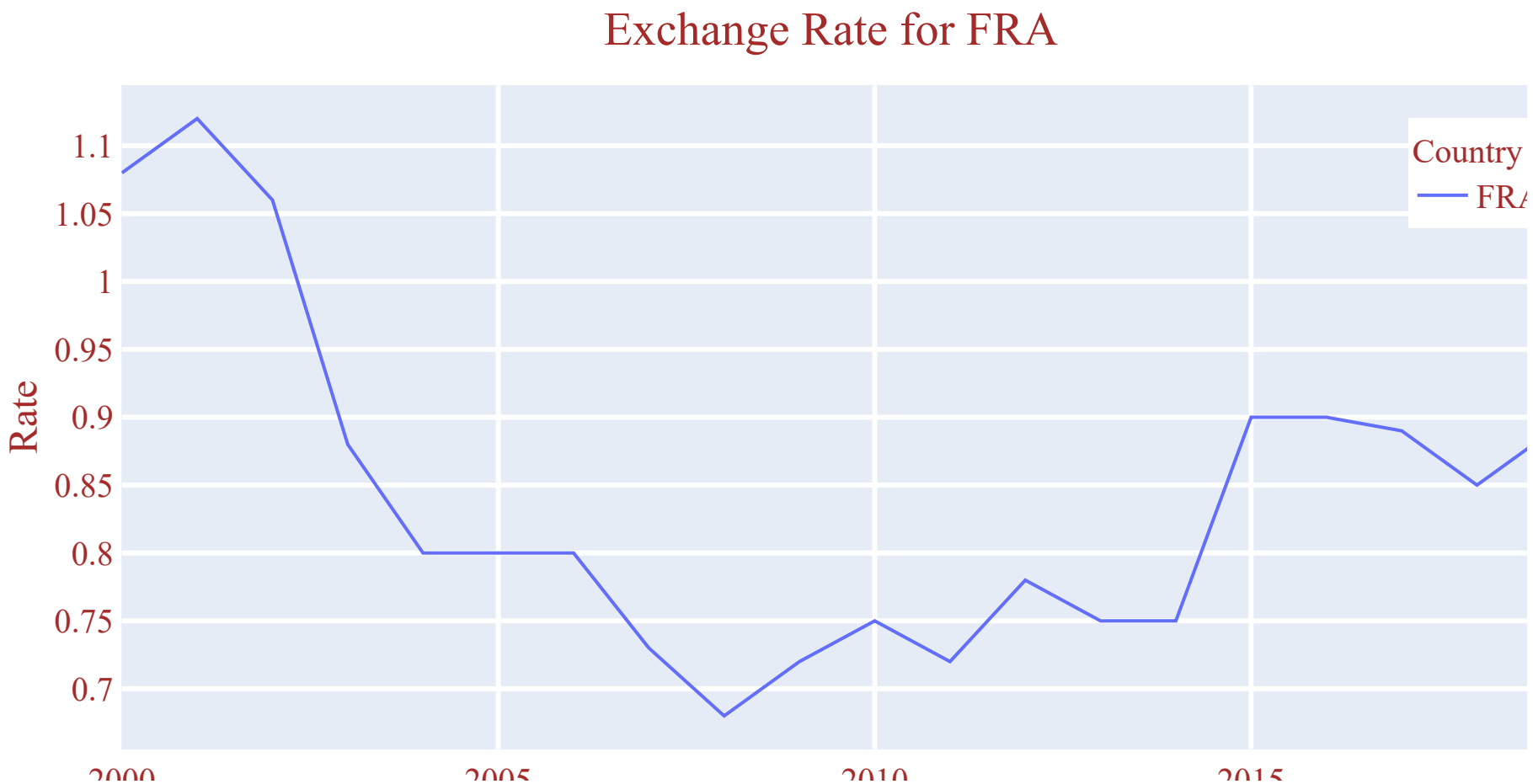
In [2]:
```
ex_rate_df=pd.read_csv('Exchange_Rates.csv')
ex_rate_df['Country']=ex_rate_df['LOCATION']
ex_rate_df['Year']=ex_rate_df['TIME']
ex_rate_df['Rate']=ex_rate_df['Value'].round(2)
ex_rate_df.head()
```

Out[2]:

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value	Flag Codes	Country	Year	Rate
0	AUS	EXCH	TOT	NATUSD	A	2000	1.724827	NaN	AUS	2000	1.72
1	AUS	EXCH	TOT	NATUSD	A	2001	1.933443	NaN	AUS	2001	1.93
2	AUS	EXCH	TOT	NATUSD	A	2002	1.840563	NaN	AUS	2002	1.84
3	AUS	EXCH	TOT	NATUSD	A	2003	1.541914	NaN	AUS	2003	1.54
4	AUS	EXCH	TOT	NATUSD	A	2004	1.359752	NaN	AUS	2004	1.36

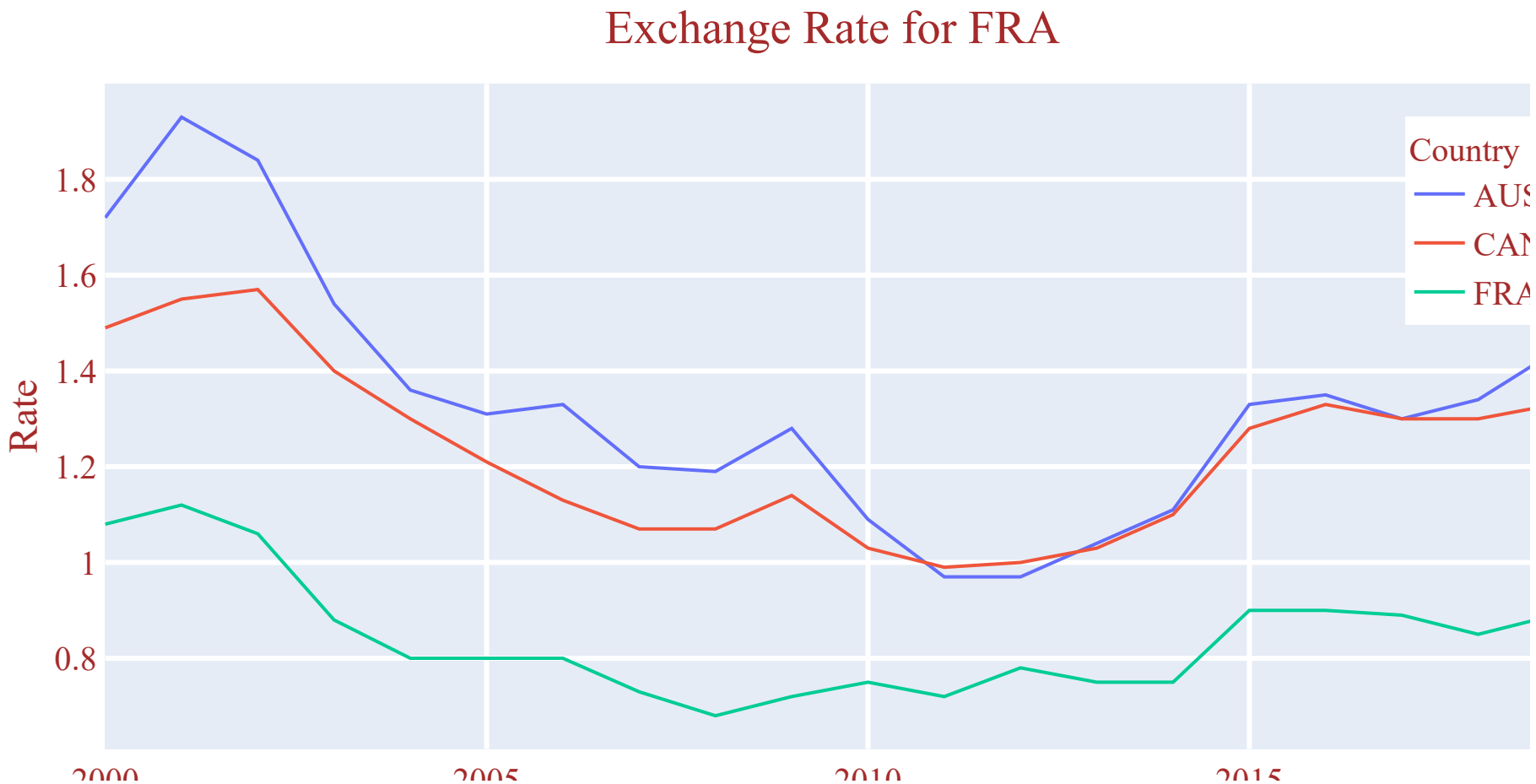
Simple line graph with plotly express

In [3]:
```
fig = px.line(ex_rate_df[ex_rate_df['Country'].isin(['FRA'])], x="Year", y="Rate", color='Country',
              title='Exchange Rate for FRA')

fig.update_layout(title={'text': 'Exchange Rate for FRA','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



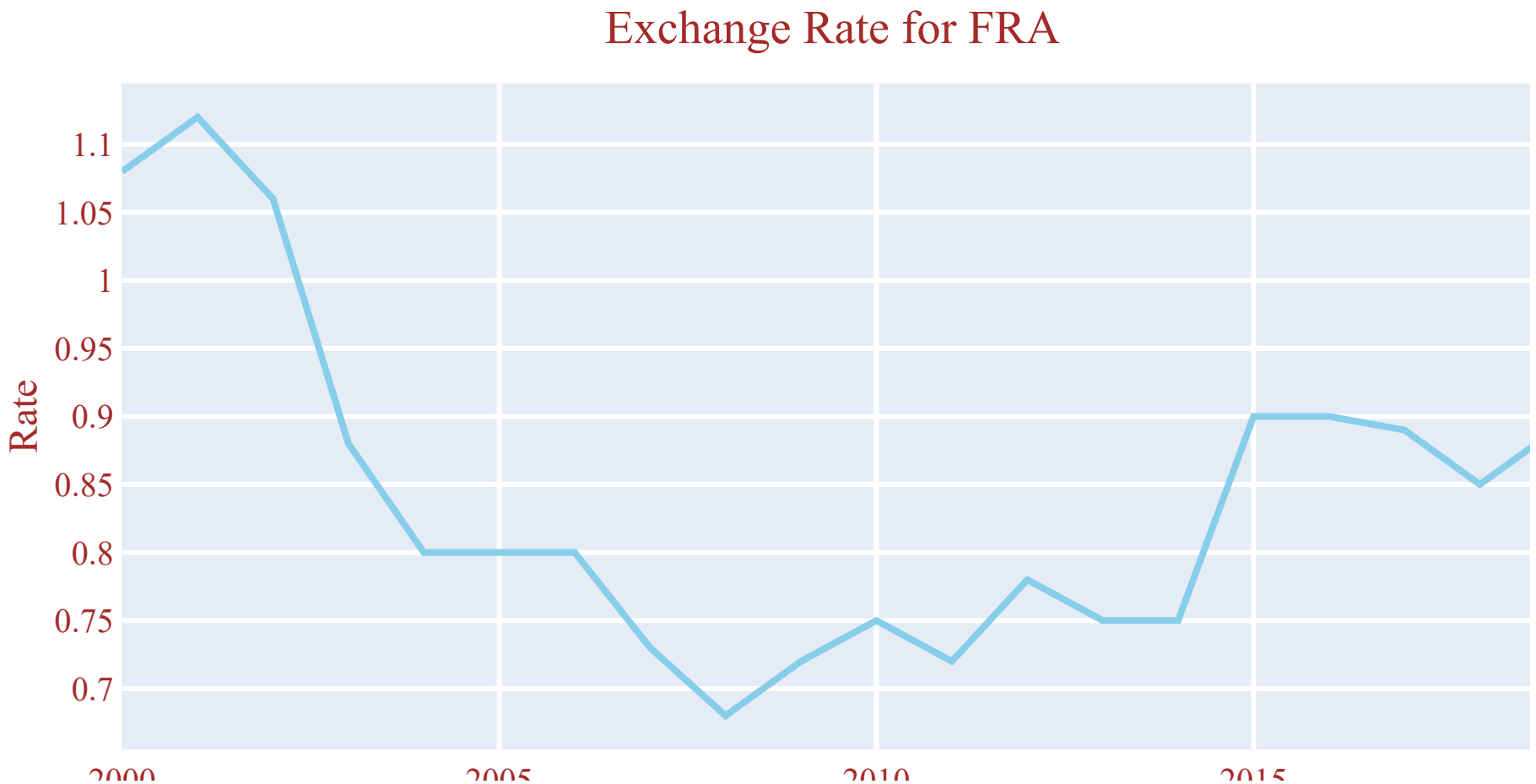
Multiple line graph with plotly express

In [4]:
```
fig = px.line(ex_rate_df[ex_rate_df['Country'].isin(['FRA','CAN','AUS'])], x="Year", y="Rate",
              color='Country',  title='Exchange Rate for AUS, CAN and FRA')

fig.update_layout(title={'text': 'Exchange Rate for FRA','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



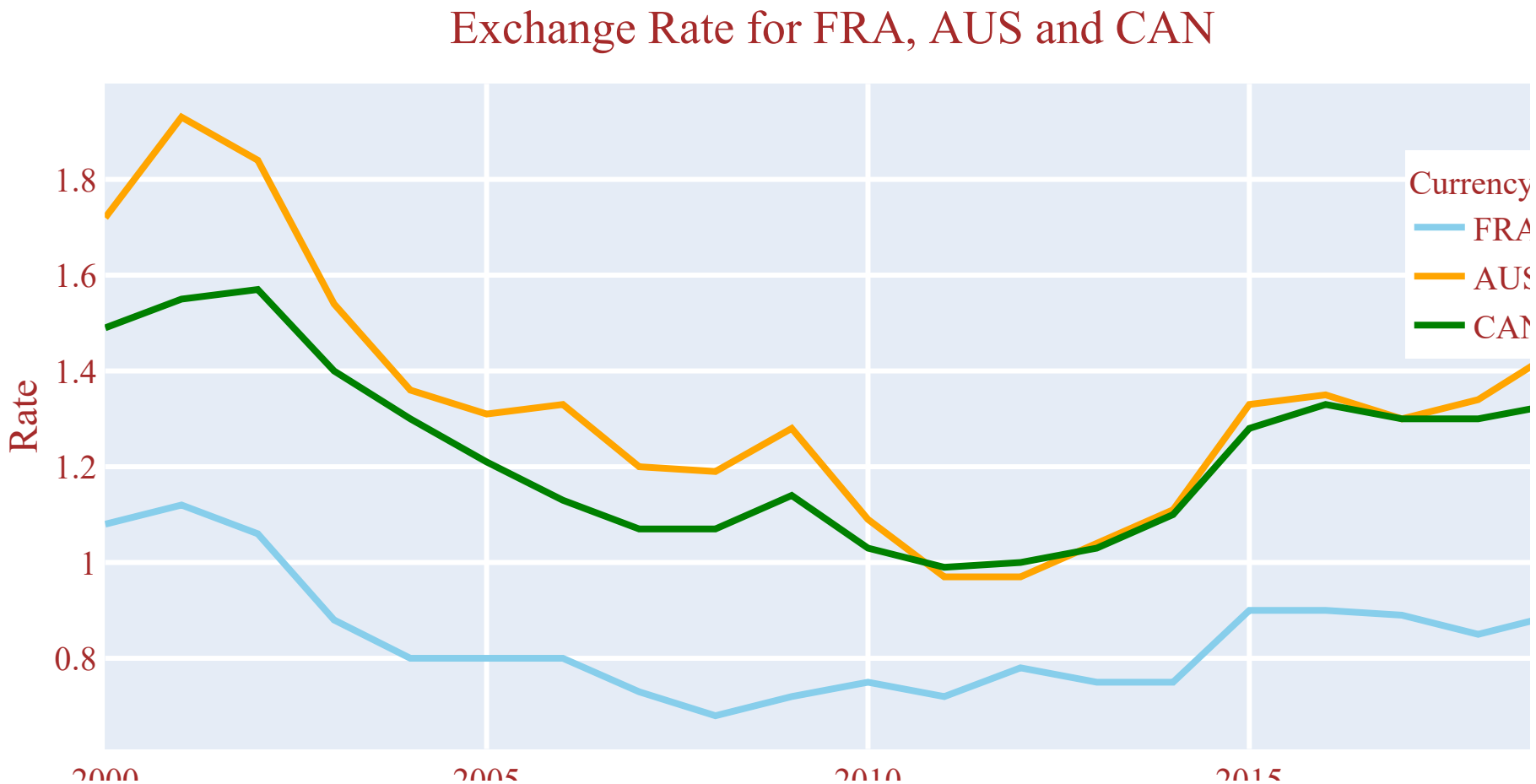
Simple line graph with plotly graph object

In [5]:
```
fig=go.Figure()
fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['FRA'])]['Rate'],
                                 name='FRA',text='Rate',line=dict(color='skyblue', width=4,shape='linear')))

fig.update_layout(title={'text': 'Exchange Rate for FRA','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                           legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                   autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                   font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Multi-line graph with plotly graph object

In [6]:
```
fig=go.Figure()
fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['FRA'])]['Rate'],
                                name='FRA',line=dict(color='skyblue', width=4,shape='linear')))

fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['AUS'])]['Rate'],
                                name='AUS',line=dict(color='orange', width=4,shape='linear')))

fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['CAN'])]['Rate'],
                                name='CAN',line=dict(color='green', width=4,shape='linear')))

fig.update_layout(title={'text': 'Exchange Rate for FRA, AUS and CAN','y':0.95,'x':0.5, 'xanchor': 'center','yanchor':
'top'},
                          legend=dict(yanchor="top",y=0.9,xanchor="right",x=0.95,title='Currency'),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Bar Graph in Plotly

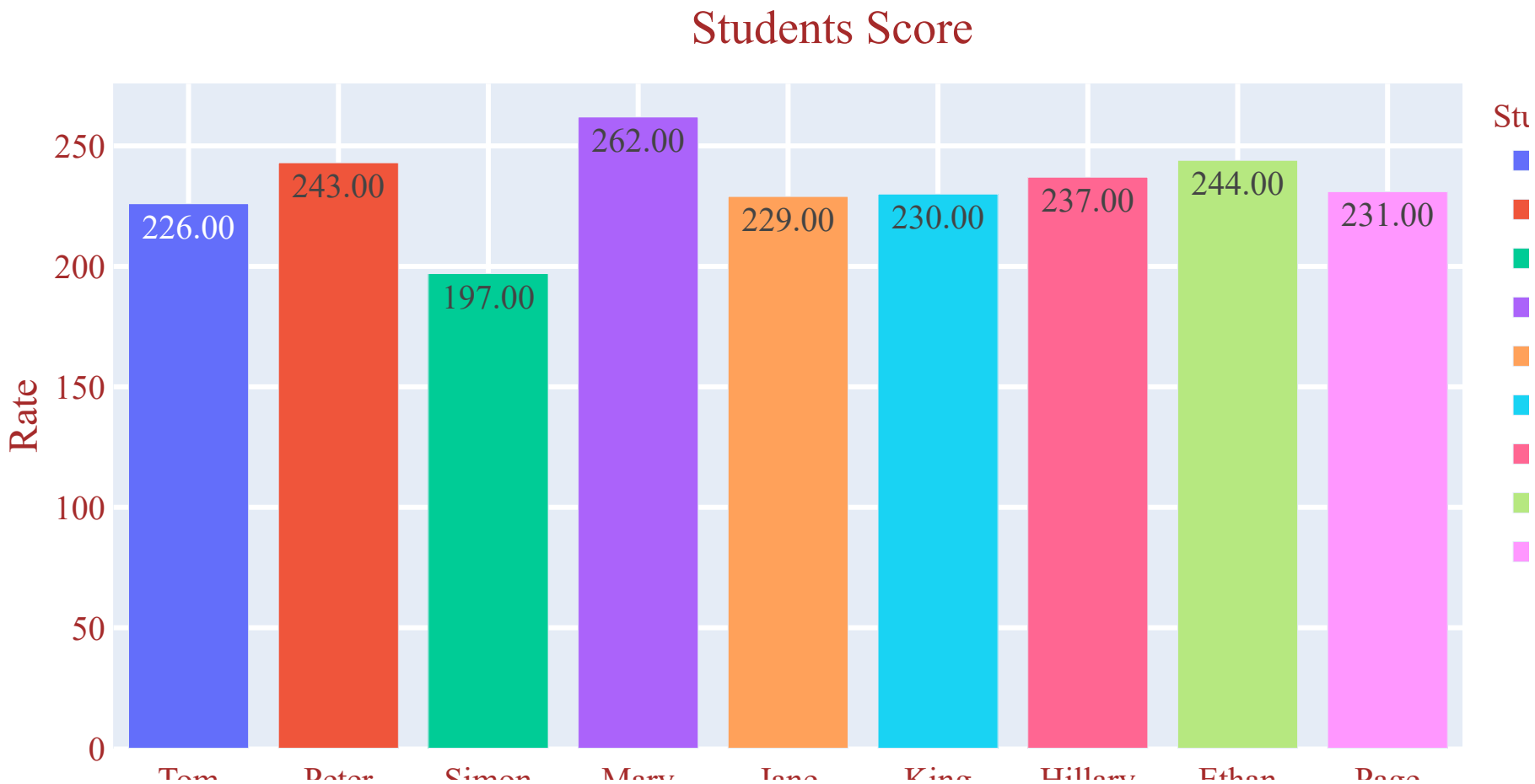
In [7]:
```
score_df = pd.DataFrame(
    {
        "Student": ["Tom", "Peter","Simon", "Mary", "Jane","King","Hillary","Ethan","Page"],
        "Math": [79.00, 67.00,80.00, 84.00, 70.00,60.00,90.00,76.00,75],
        "Physics":[63.00, 98, 60.00, 90,84.00, 77.00,55.00,70,66.00],
        "Computer":[84.00,78.00, 57.00, 88.00, 75.00,93.00,92.00,98.00,90.00],
    }
)

score_df['Total']=score_df[['Math','Physics','Computer']].apply(np.sum,axis=1)
score_df
```

Out[7]:

	Student	Math	Physics	Computer	Total
0	Tom	79.0	63.0	84.0	226.0
1	Peter	67.0	98.0	78.0	243.0
2	Simon	80.0	60.0	57.0	197.0
3	Mary	84.0	90.0	88.0	262.0
4	Jane	70.0	84.0	75.0	229.0
5	King	60.0	77.0	93.0	230.0
6	Hillary	90.0	55.0	92.0	237.0
7	Ethan	76.0	70.0	98.0	244.0
8	Page	75.0	66.0	90.0	231.0

Simple Bar Graph in Plotly Express

In [8]:
```
fig = px.bar(score_df, x='Student', y='Total',text='Total',color='Student')
fig.update_traces(texttemplate='%{text:.2f}', textposition='inside')

fig.update_layout(title={'text': 'Students Score','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



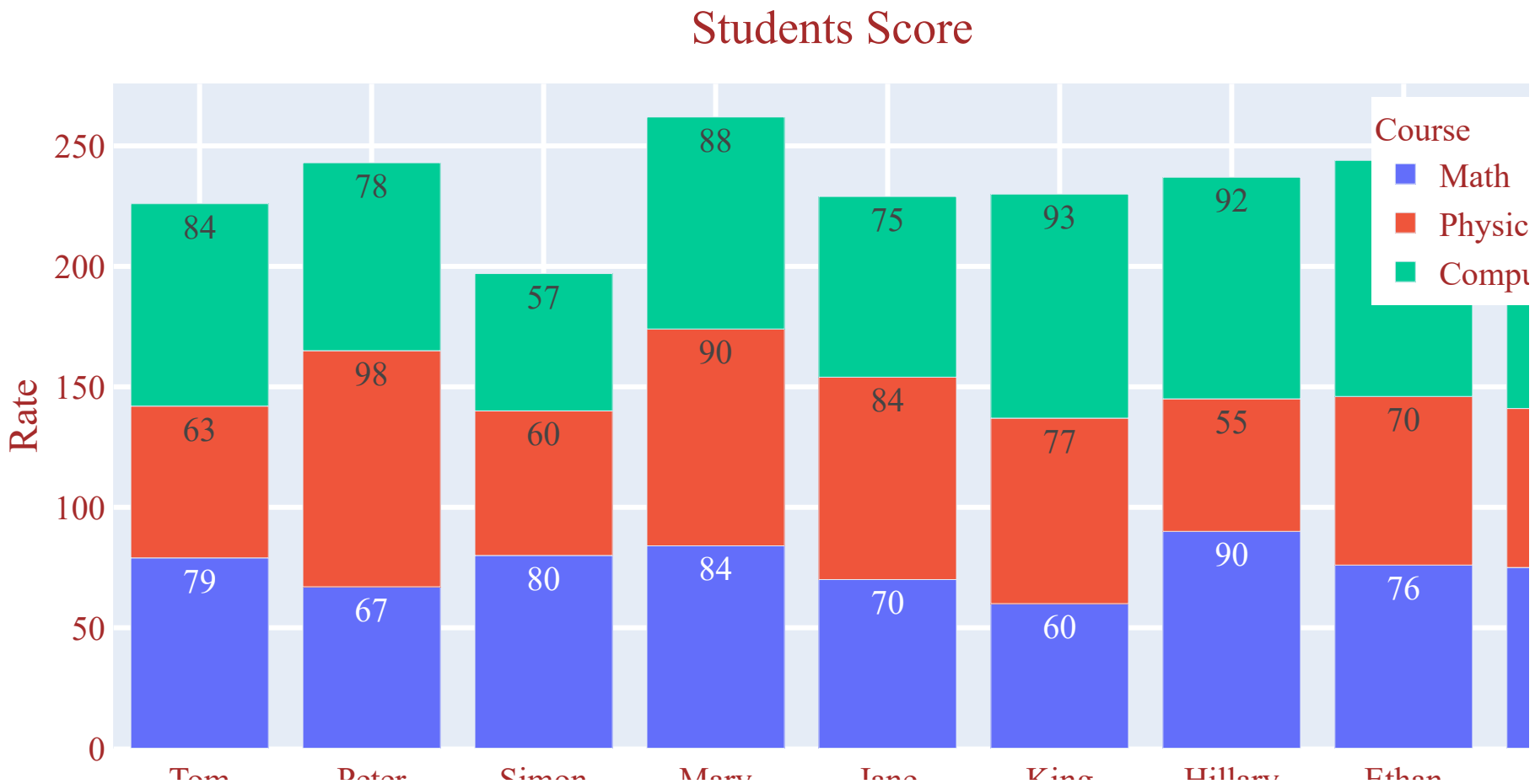
Stacked Bar Graph

In [9]:
```
score_melt_df=pd.melt(score_df.reset_index(),id_vars=['Student'],value_vars=['Math','Physics','Computer'],
        value_name='Score')
score_melt_df['Course']=score_melt_df['variable']
score_melt_df.head()
```

Out[9]:

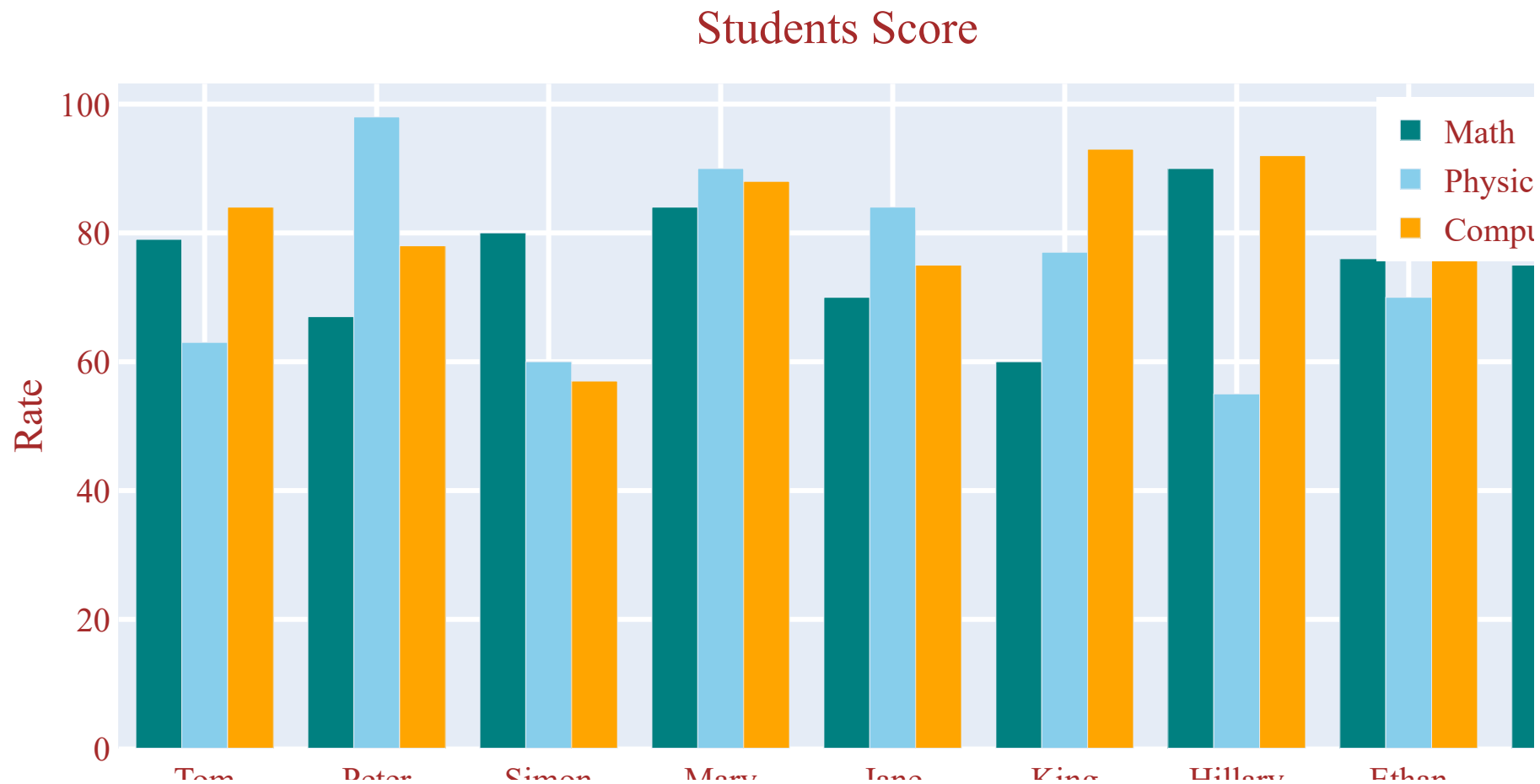
	Student	variable	Score	Course
0	Tom	Math	79.0	Math
1	Peter	Math	67.0	Math
2	Simon	Math	80.0	Math
3	Mary	Math	84.0	Math
4	Jane	Math	70.0	Math

In [10]:
```
fig = px.bar(score_melt_df, x='Student' , y='Score', text='Score',color='Course')
fig.update_traces(texttemplate='%{text:.2s}', textposition='inside')

fig.update_layout(title={'text': 'Students Score','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.98,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Grouped Bar Graph

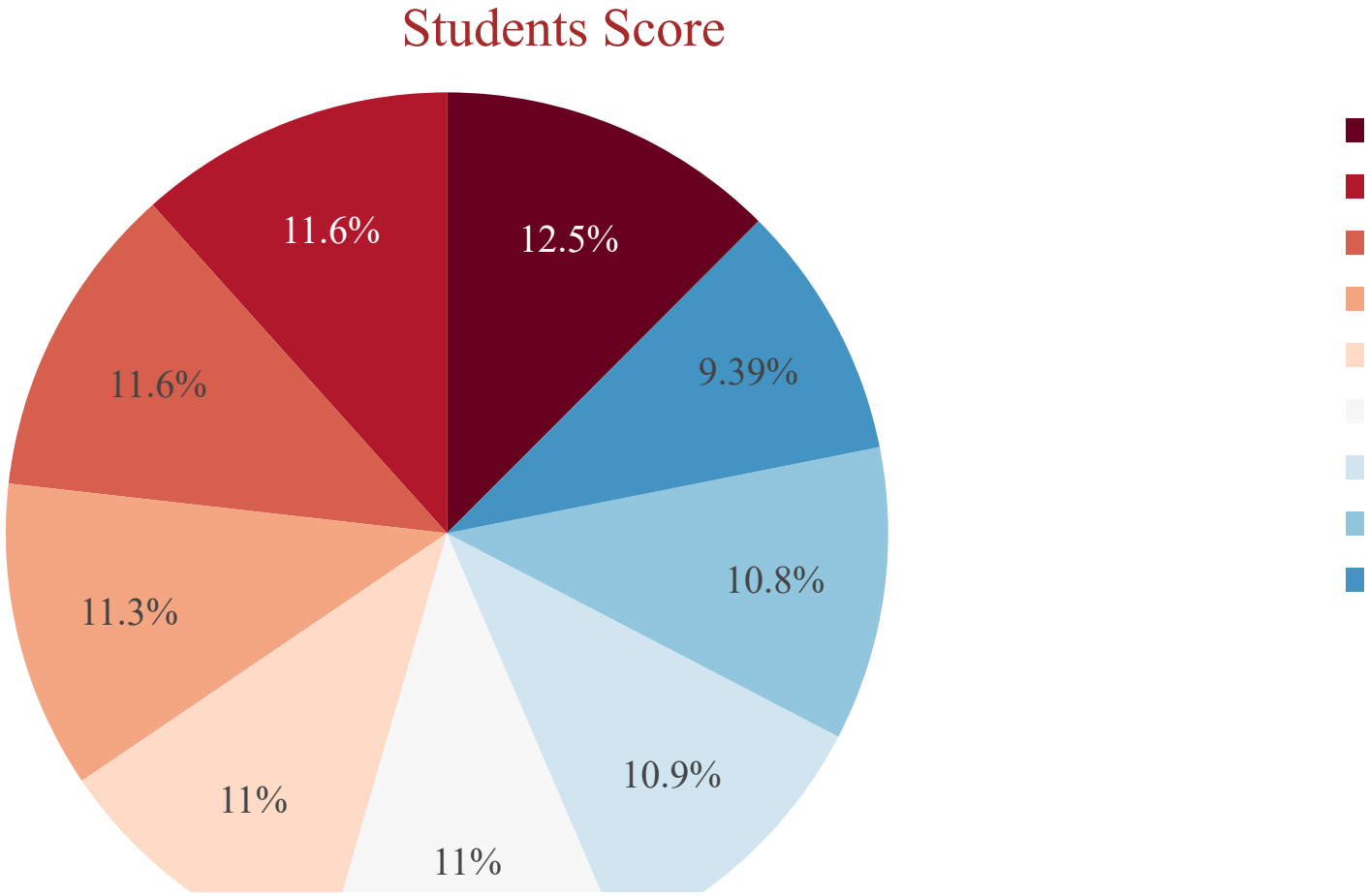
In [11]:
```
fig=go.Figure()
fig.add_trace(go.Bar(x=score_melt_df['Student'], y=score_melt_df[score_melt_df['Course']=='Math']['Score'],
                     name='Math',marker_color='Teal'))
fig.add_trace(go.Bar(x=score_melt_df['Student'], y=score_melt_df[score_melt_df['Course']=='Physics']['Score'],
                     name='Physics',marker_color='skyblue'))
fig.add_trace(go.Bar(x=score_melt_df['Student'], y=score_melt_df[score_melt_df['Course']=='Computer']['Score'],
                     name='Computer',marker_color='orange'))

fig.update_layout(title={'text': 'Students Score','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.98,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Pie Charts in Plotly

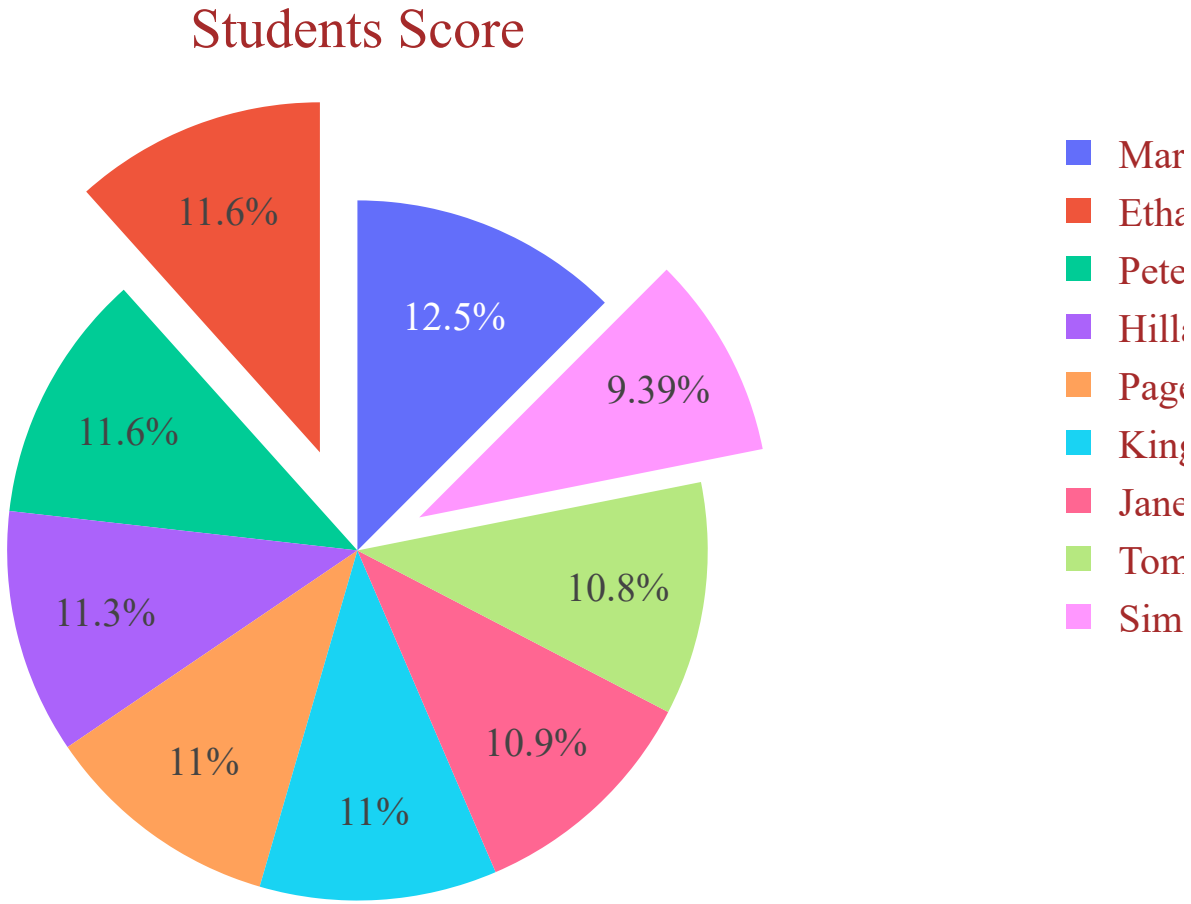
Simple Pie Chart

In [12]:
```
fig = px.pie(score_melt_df, values='Score',names='Student', color_discrete_sequence=px.colors.sequential.RdBu)

fig.update_layout(title={'text': 'Students Score','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0),
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



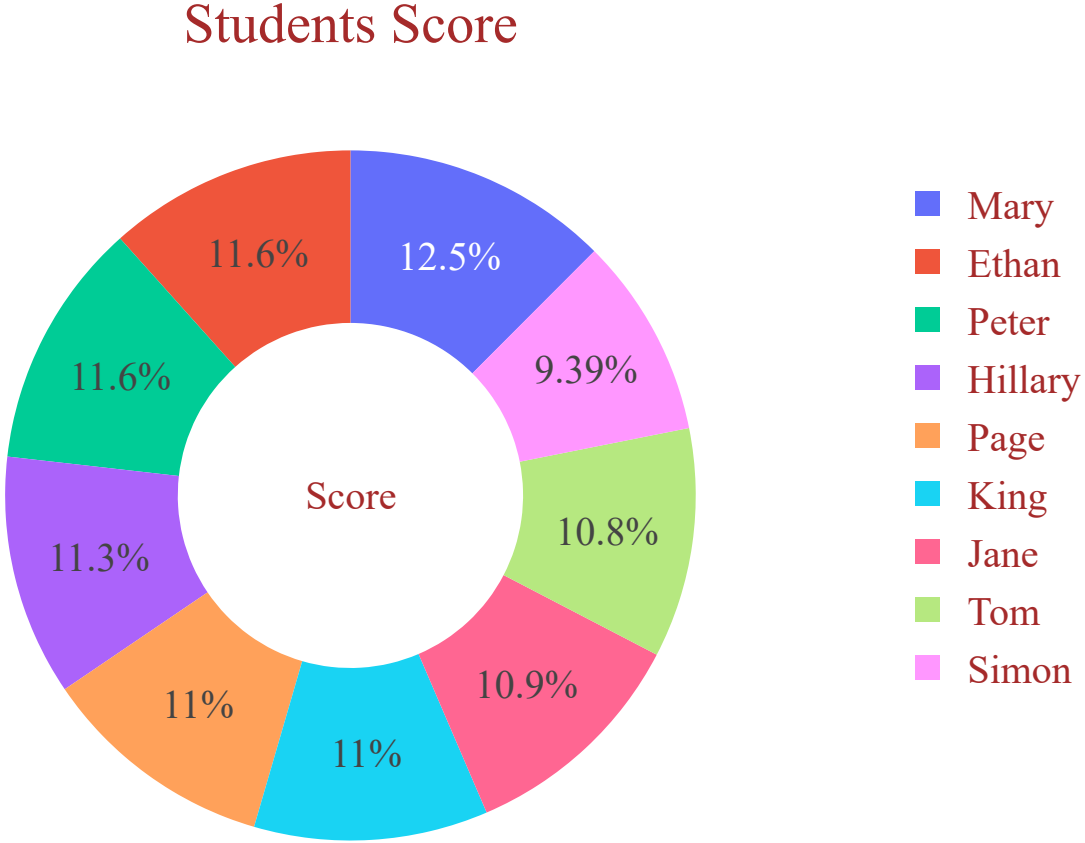
Explode pie

In [13]:
```
fig=go.Figure()
fig.add_trace(go.Pie(labels=score_melt_df['Student'], values=score_melt_df['Score'],
             pull=[0,0,0.2,0,0,0,0,0.3,0]))

fig.update_layout(title={'text': 'Students Score','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.98,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0),
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.show()
```



Donought Pie Graph

In [14]:
```
fig=go.Figure()
fig.add_trace(go.Pie(labels=score_melt_df['Student'], values=score_melt_df['Score']))
fig.update_traces(hole=.5)

fig.update_layout(title={'text': 'Students Score','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.98,xanchor="right",x=0.95),
                  annotations=[dict(text='Score', x=0.50, y=0.5, showarrow=False)],
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.show()
```



Scatter Plot in Plotly

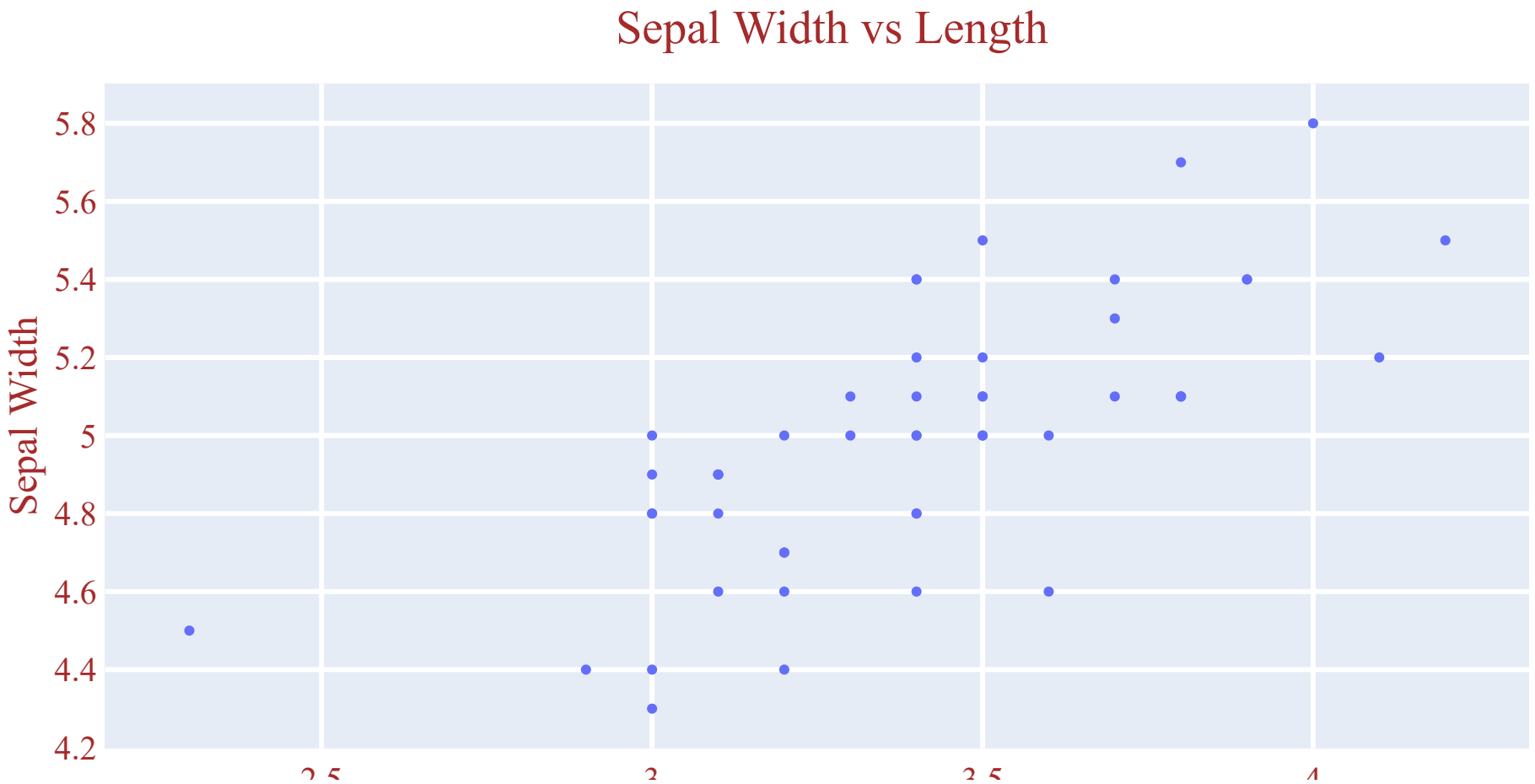
In [15]:
```
iris_df=pd.read_csv('iris.csv')
iris_df.head()
```

Out[15]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

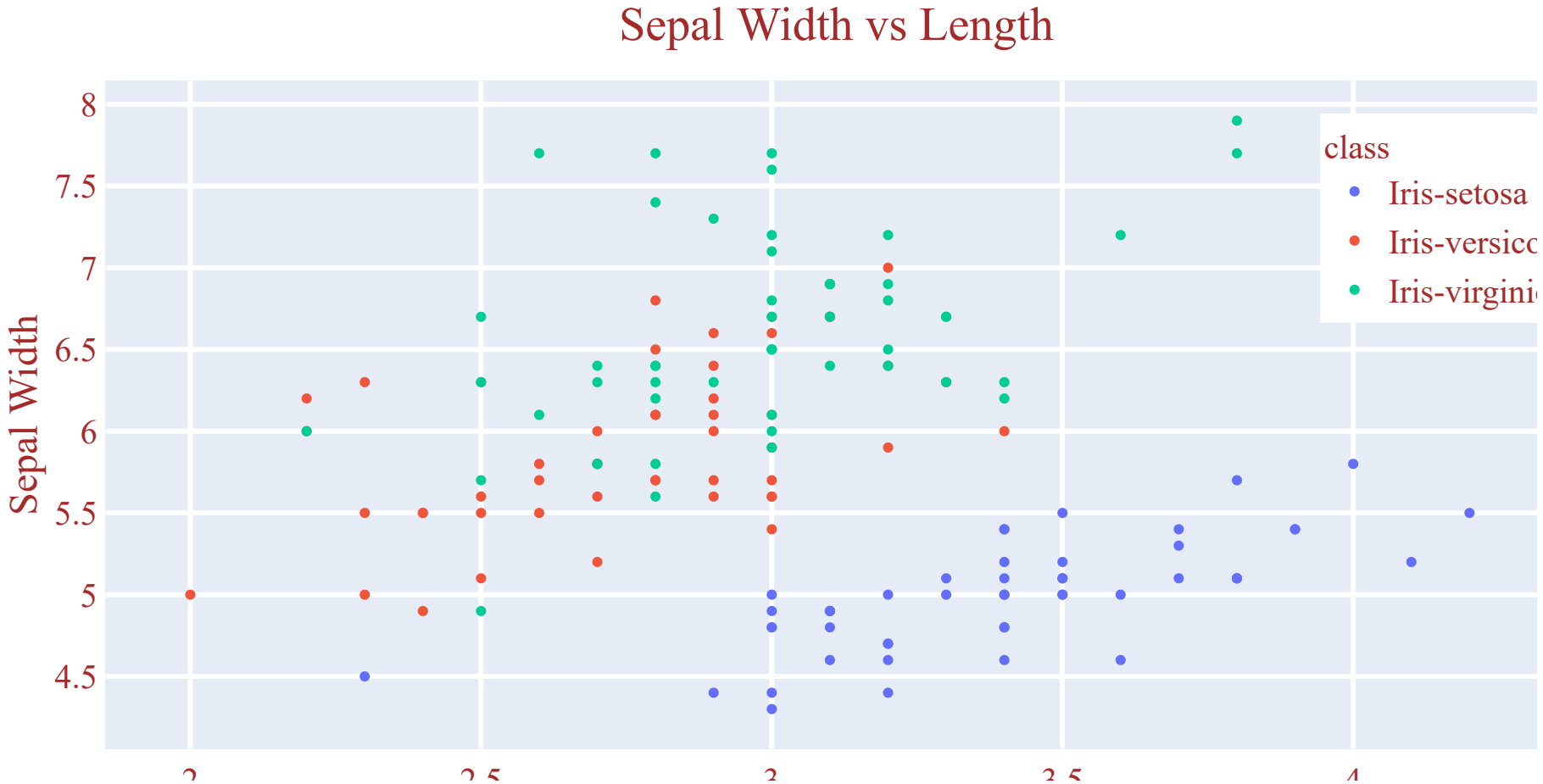
Simple Scatter Plot

In [16]:
```
iris_simple_scatterplot_df=iris_df[iris_df['class'].isin(['Iris-setosa'])]
fig = px.scatter(iris_simple_scatterplot_df, x='sepal_width', y='sepal_length')

fig.update_layout(title={'text': 'Sepal Width vs Length','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Sepal Width',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Scatter plot with Grouped Data

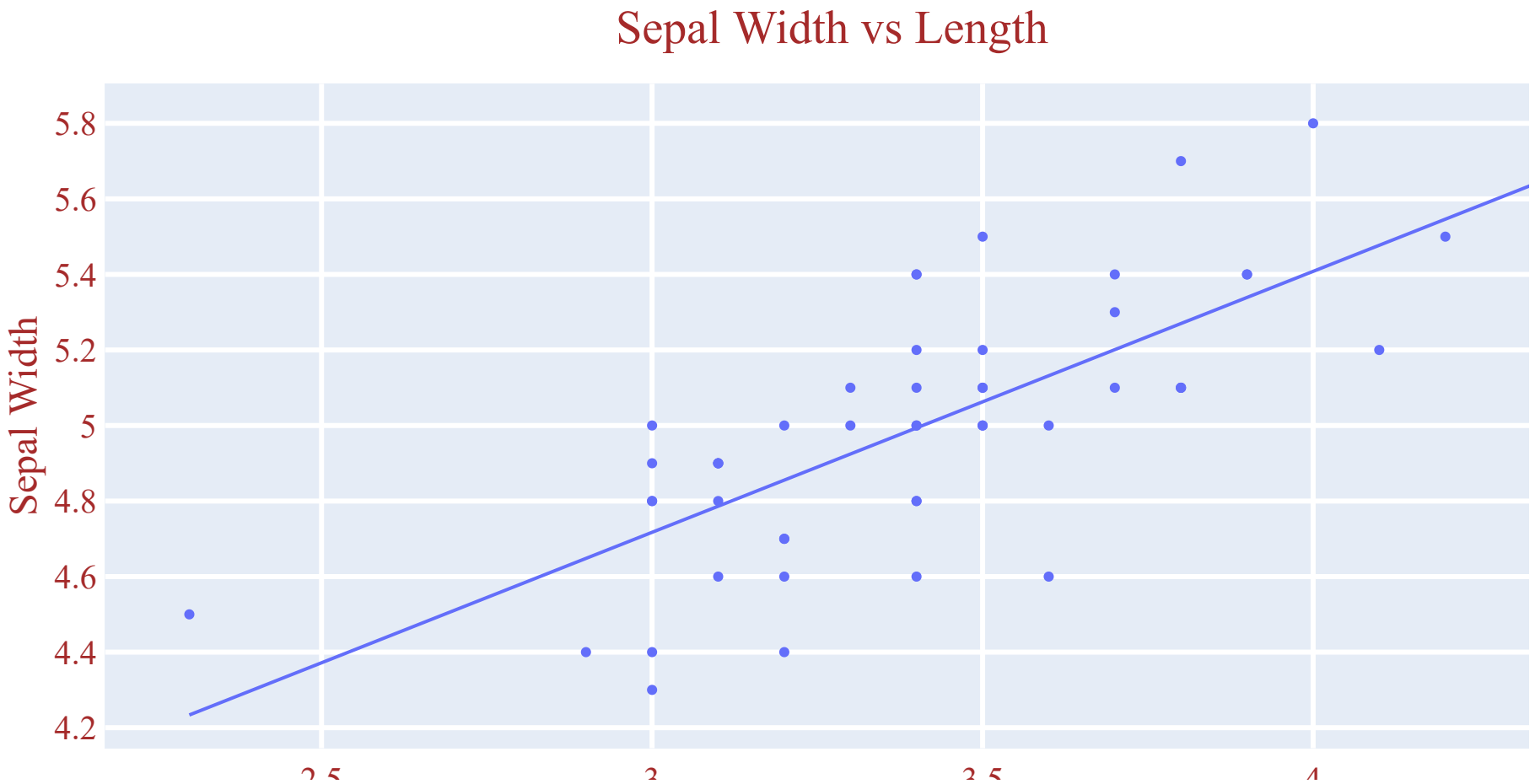
In [17]:
```
fig = px.scatter(iris_df, x='sepal_width', y='sepal_length',color='class')

fig.update_layout(title={'text': 'Sepal Width vs Length','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Sepal Width',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Scatter Plot With Trendline

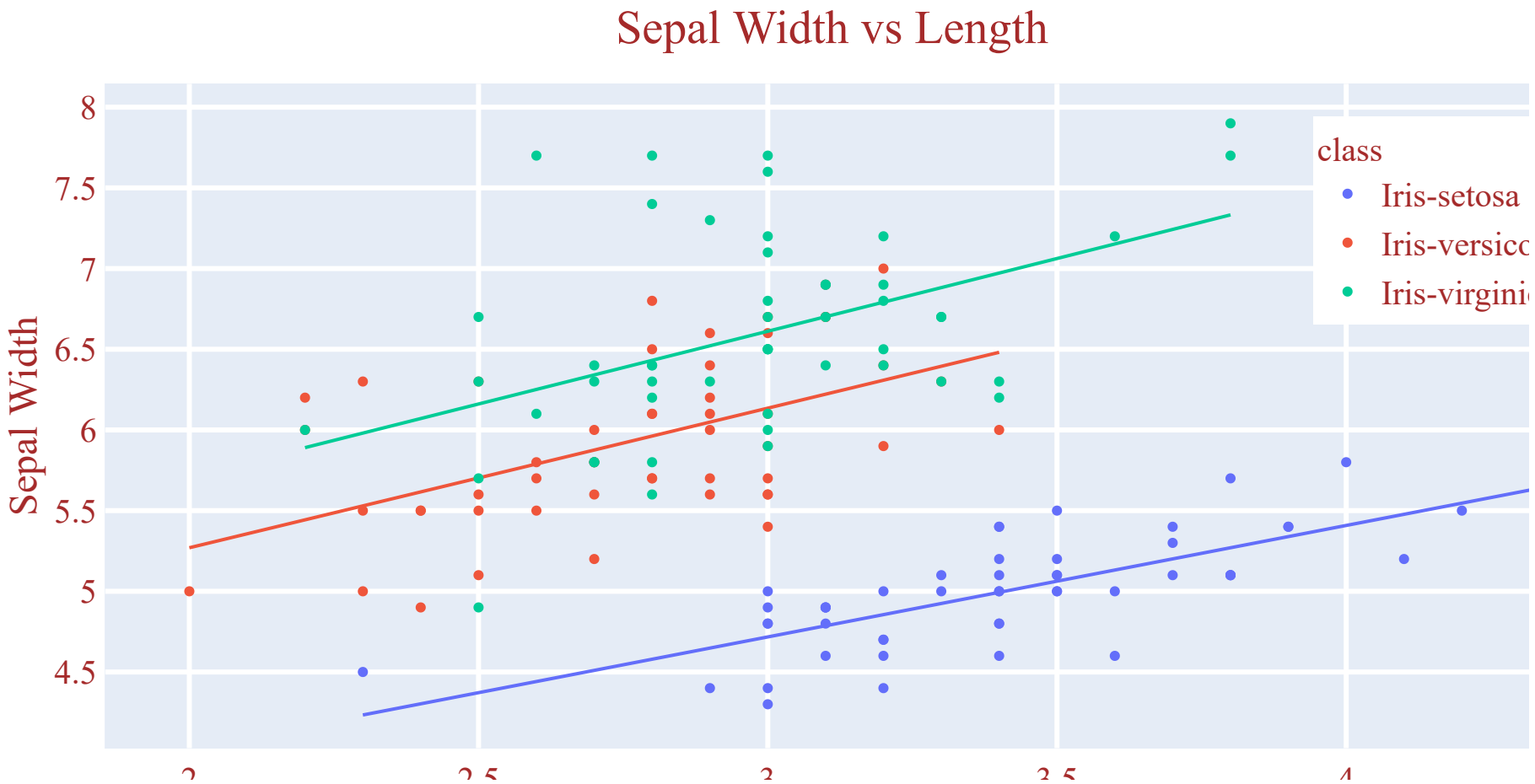
Install statsmodel that comes with ols. Run below command in terminal
pip install statsmodels --upgrade

In [18]:
```
iris_simple_scatterplot_df=iris_df[iris_df['class'].isin(['Iris-setosa'])]
fig = px.scatter(iris_simple_scatterplot_df, x='sepal_width', y='sepal_length', trendline="ols")


fig.update_layout(title={'text': 'Sepal Width vs Length','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Sepal Width',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Scatter plot with Trendline and grouped data

In [19]:
```
fig = px.scatter(iris_df, x='sepal_width', y='sepal_length',color='class',trendline='ols')

fig.update_layout(title={'text': 'Sepal Width vs Length','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Sepal Width',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```


Sepal Width vs Length
Sepal Width
8
7.5
7
6.5
6
5.5
5
4.5
2
2.5
3
3.5
4
class
Iris-setosa
Iris-versicc
Iris-virgini

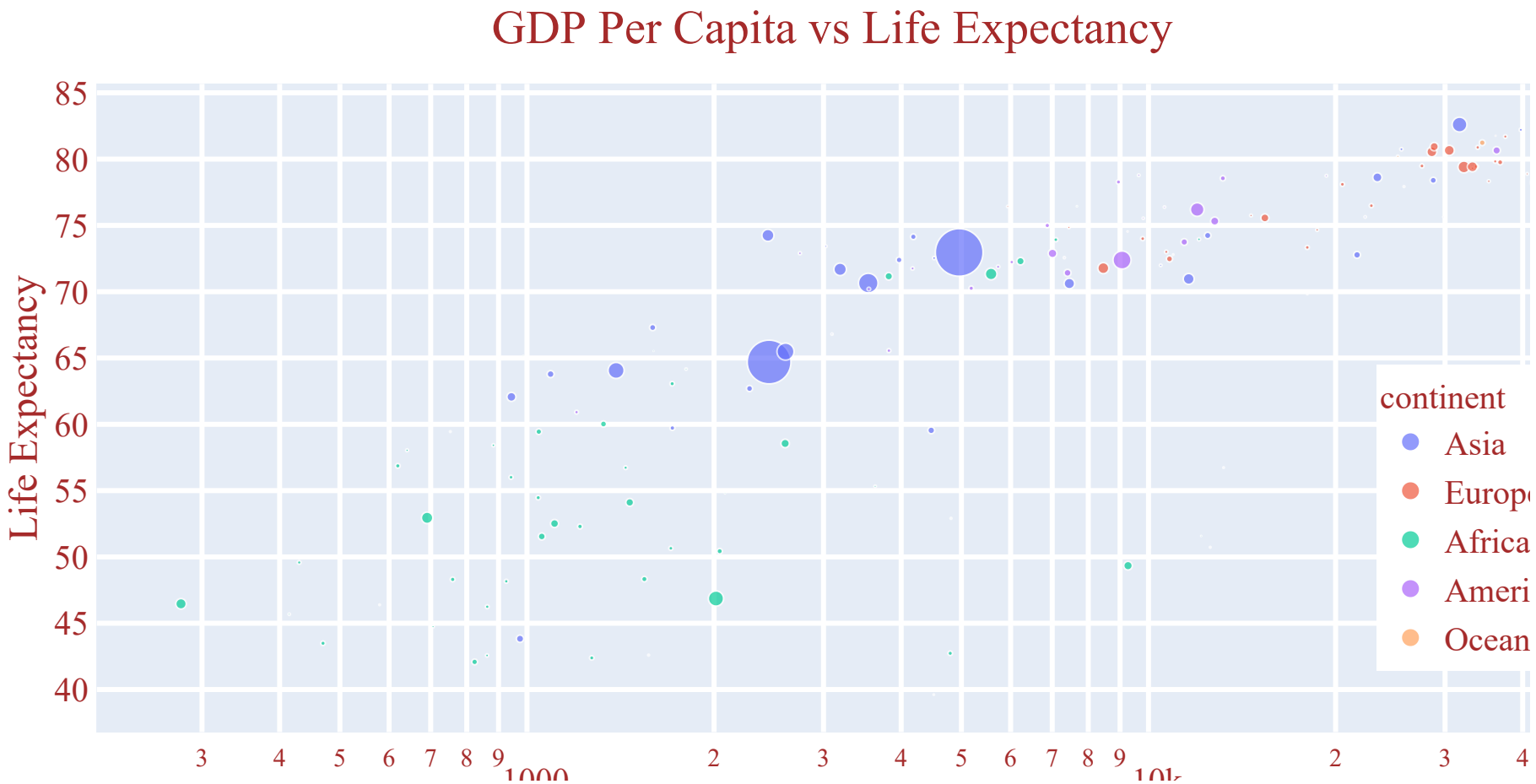
Buble Chart

In [20]:
```
gdpPercap_df=pd.read_csv('gdpPercap.csv')
gdpPercap_df.head()
```

Out[20]:

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

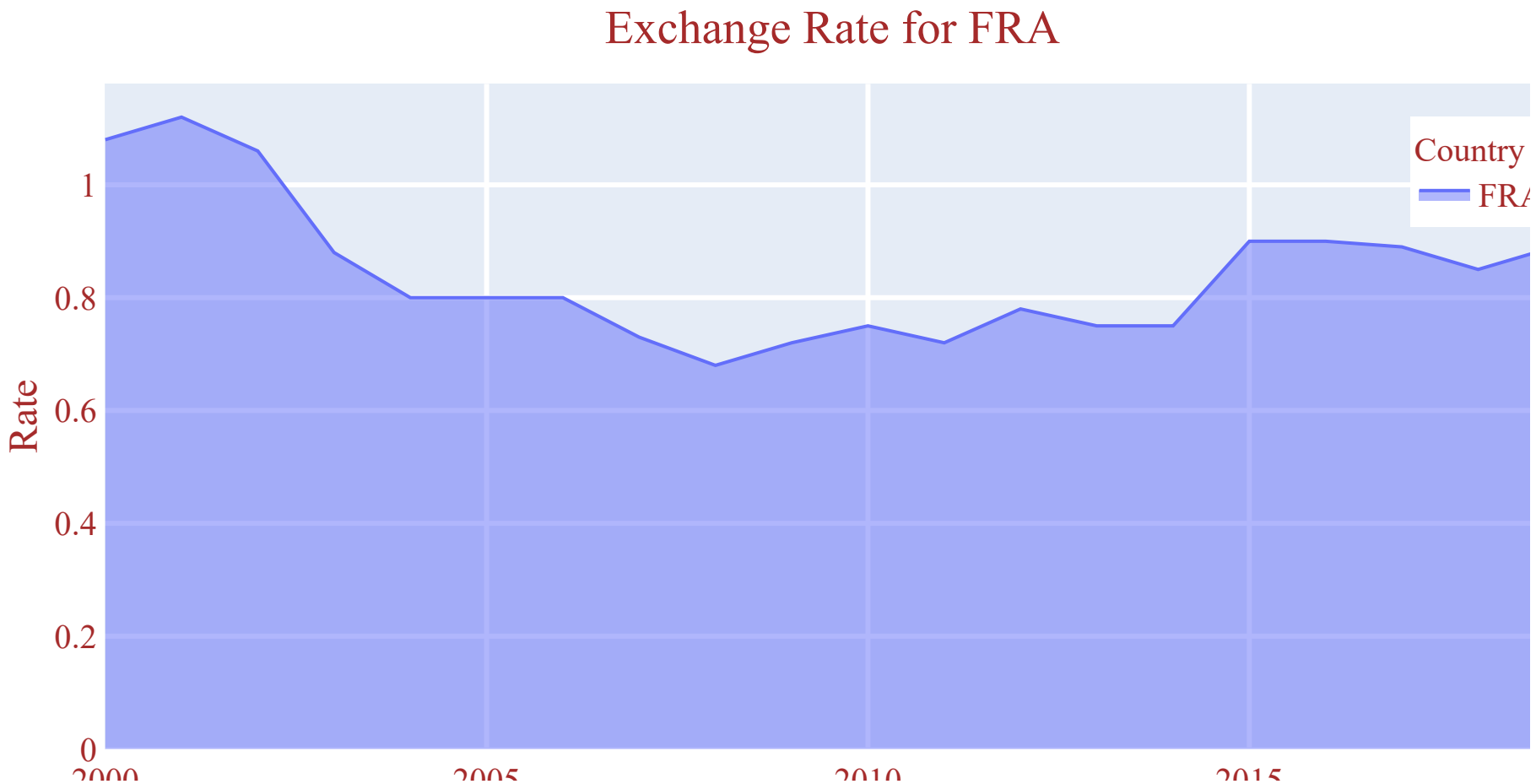
In [21]:
```
fig = px.scatter(gdpPercap_df[gdpPercap_df['year'].isin(['2007'])], x='gdpPercap',
                 y='lifeExp',color='continent',size='pop',log_x=True)

fig.update_layout(title={'text': 'GDP Per Capita vs Life Expectancy','y':0.95,'x':0.5, 'xanchor': 'center','yanchor':
'top'},
                          legend=dict(yanchor="bottom",y=0.095,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='GDP Per Capita', yaxis_title='Life Expecta
ncy',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Area Chart in Plotly

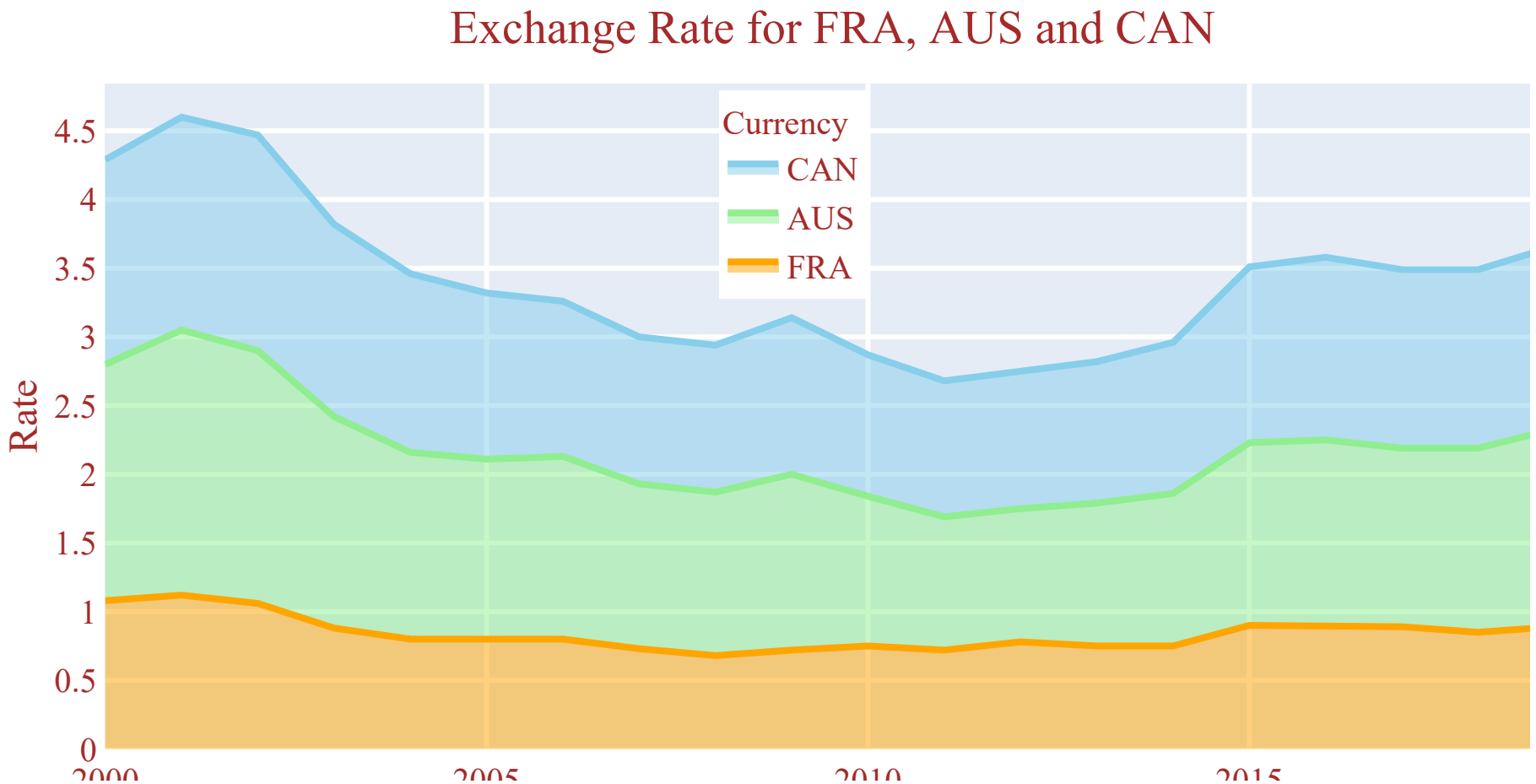
Simple Area Chart

In [22]:
```
fig=go.Figure()
fig = px.area(ex_rate_df[ex_rate_df['Country'].isin(['FRA'])], x="Year", y="Rate", color="Country",line_group="Countr
y")

fig.update_layout(title={'text': 'Exchange Rate for FRA','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



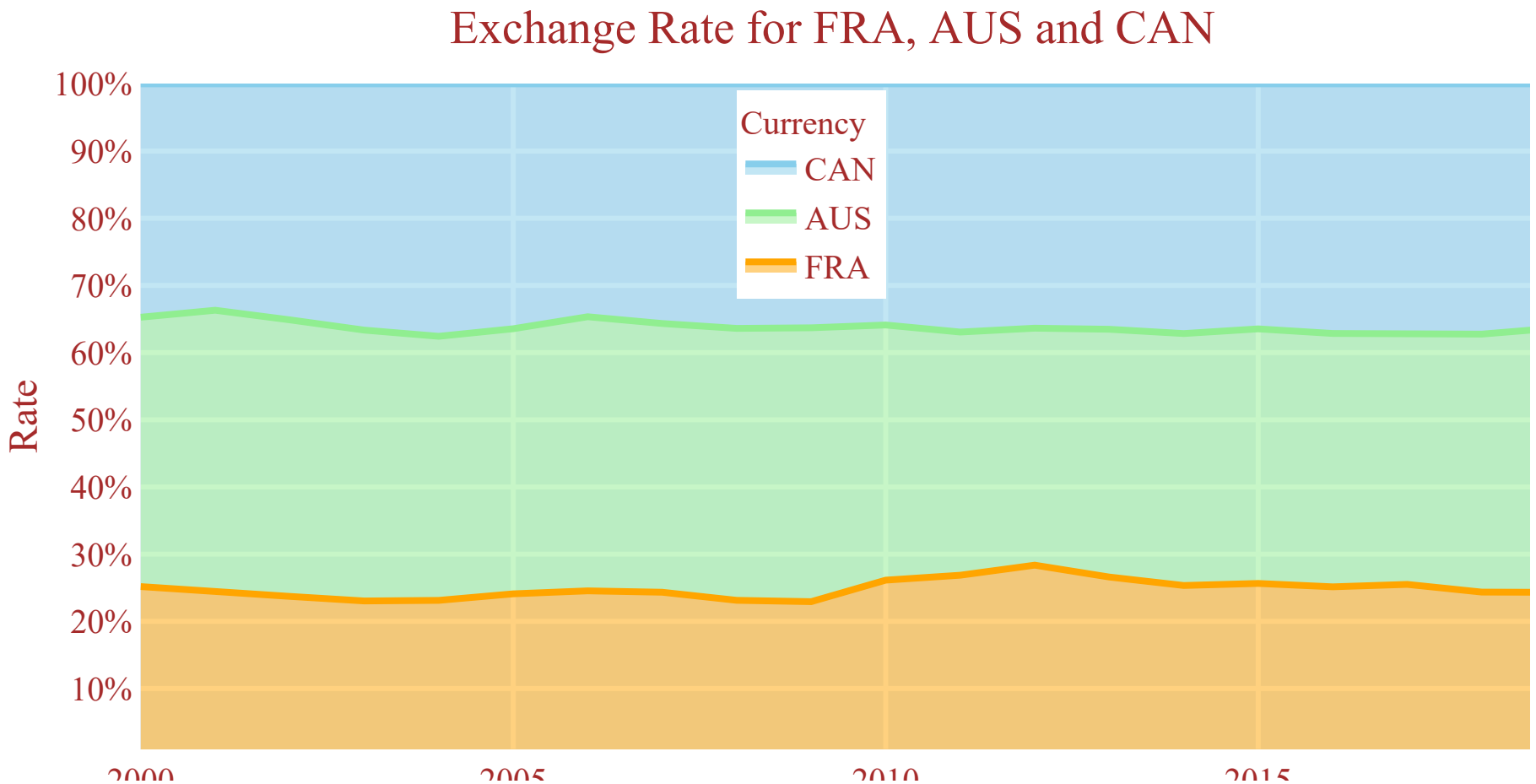
Stacked Area Chart

In [23]:
```
fig=go.Figure()
fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['FRA'])]['Rate'],stackgroup='o
ne',
                                 name='FRA',line=dict(color='orange', width=4,shape='linear')))

fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['AUS'])]['Rate'],stackgroup='o
ne',
                                 name='AUS',line=dict(color='lightgreen', width=4,shape='linear')))

fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['CAN'])]['Rate'],stackgroup='o
ne',
                                 name='CAN',line=dict(color='skyblue', width=4,shape='linear')))

fig.update_layout(title={'text': 'Exchange Rate for FRA, AUS and CAN','y':0.95,'x':0.5, 'xanchor': 'center','yanchor':
'top'},
                          legend=dict(yanchor="top",y=0.99,xanchor="right",x=0.5,title='Currency'),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



100% Stacked Area Chart

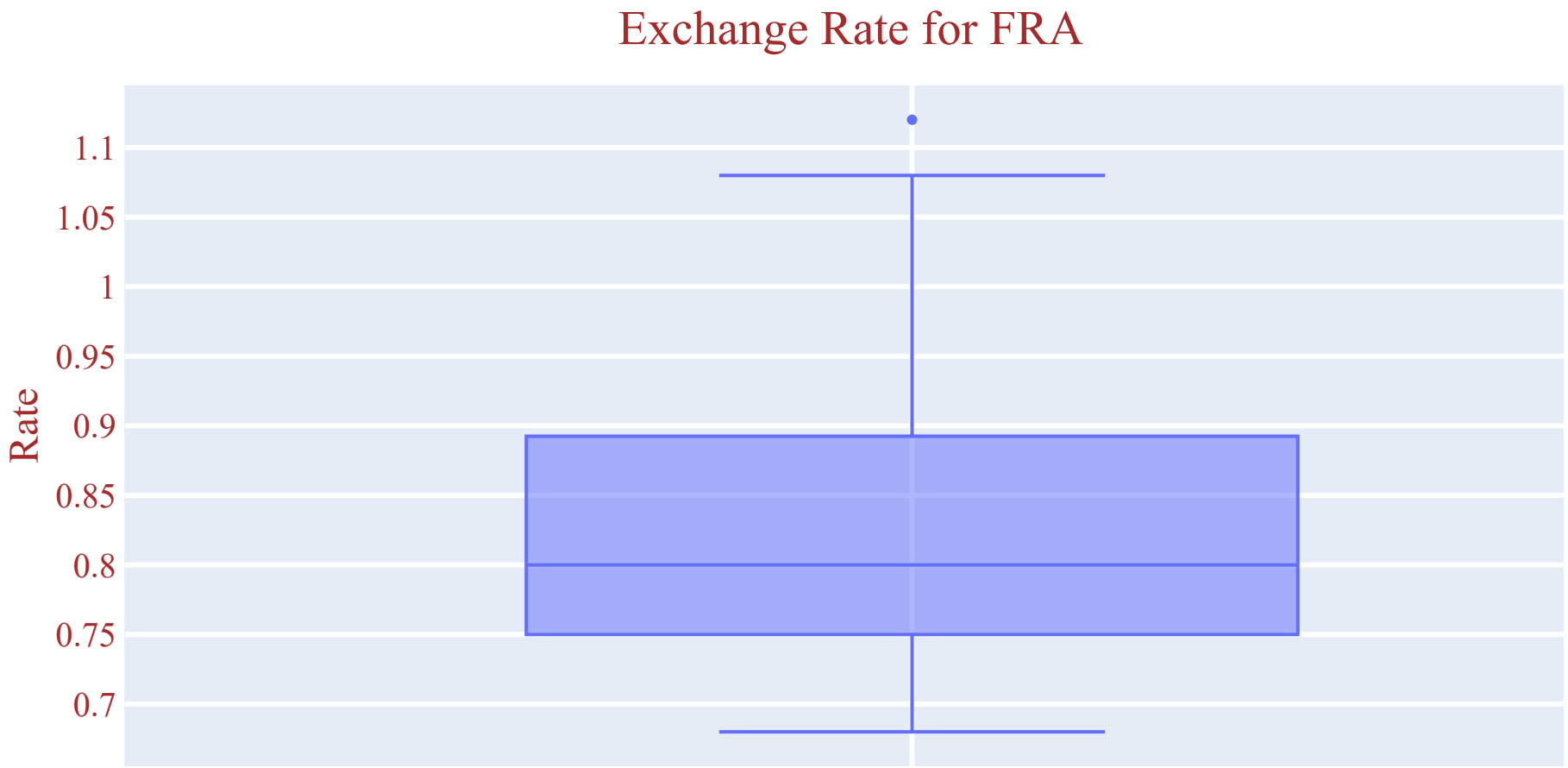
In [24]:
```
fig=go.Figure()
fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['FRA'])]['Rate'],stackgroup='one',
                                groupnorm='percent',  name='FRA',line=dict(color='orange', width=4,shape='linear')))

fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['AUS'])]['Rate'],stackgroup='one',
                                 name='AUS',line=dict(color='lightgreen', width=4,shape='linear')))

fig.add_trace(go.Scatter(x=ex_rate_df['Year'], y=ex_rate_df[ex_rate_df['Country'].isin(['CAN'])]['Rate'],stackgroup='one',
                                  name='CAN',line=dict(color='skyblue', width=4,shape='linear')))

fig.update_layout(title={'text': 'Exchange Rate for FRA, AUS and CAN','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                           legend=dict(yanchor="top",y=0.99,xanchor="right",x=0.5,title='Currency'),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                  yaxis=dict(range=[1, 100],ticksuffix='%'))

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Boxplot in Plotly

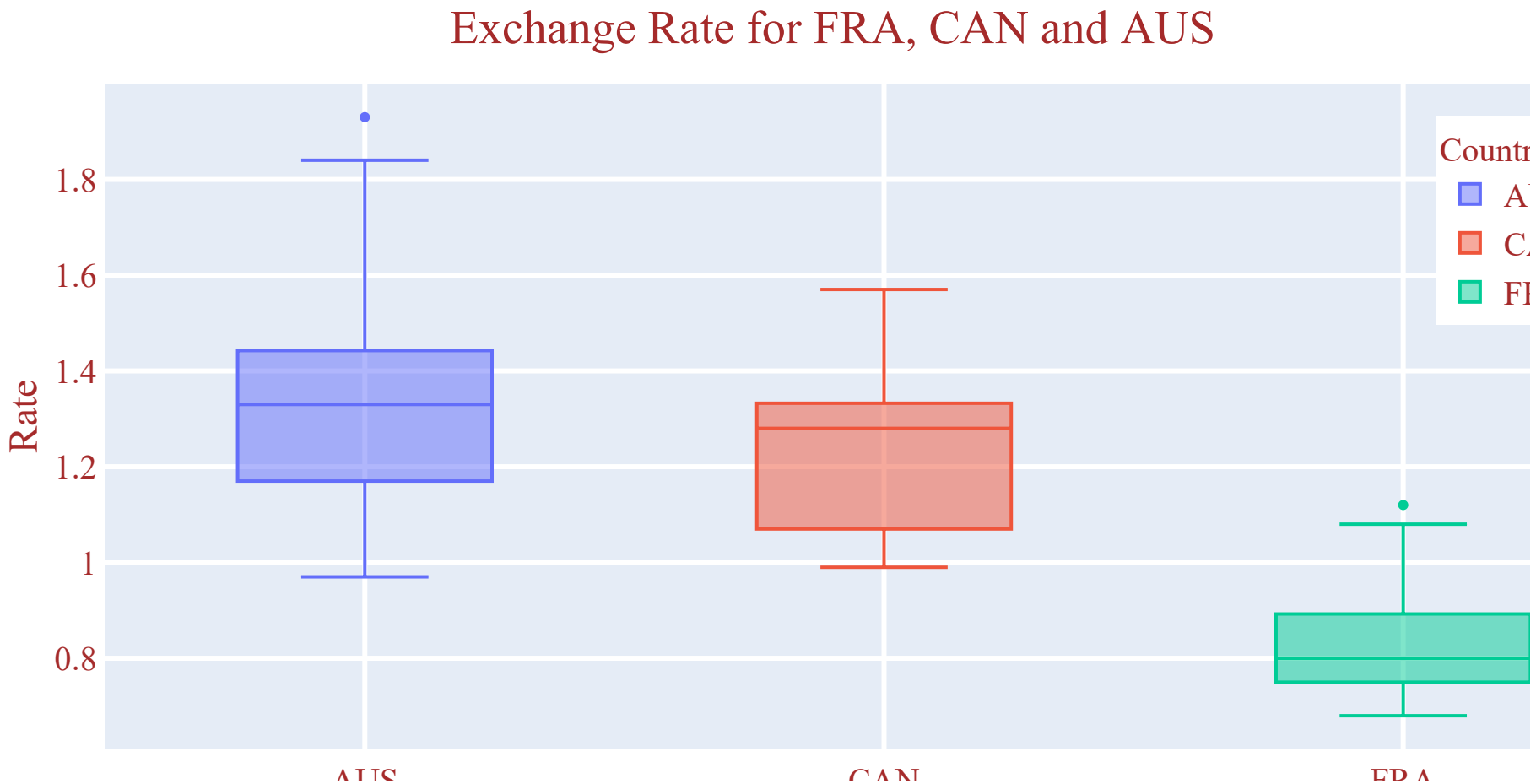
Simple Boxplot

In [25]:
```
fig = px.box(ex_rate_df[ex_rate_df['Country'].isin(['FRA'])], y="Rate")

fig.update_layout(title={'text': 'Exchange Rate for FRA','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Currency', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



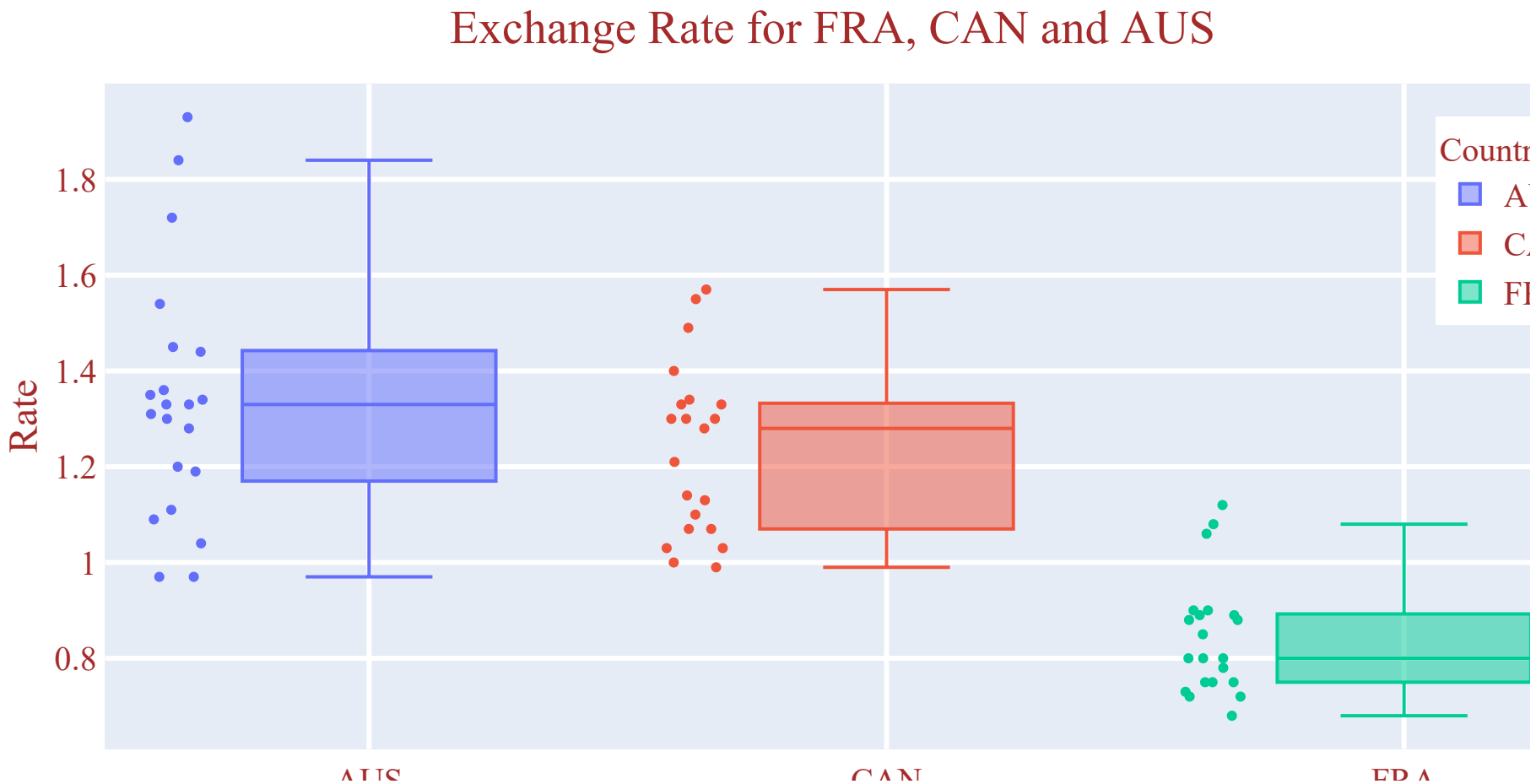
Multi-variable Boxplot

In [26]:
```
fig = px.box(ex_rate_df[ex_rate_df['Country'].isin(['FRA','CAN','AUS'])], y="Rate", x='Country', color='Country')

fig.update_layout(title={'text': 'Exchange Rate for FRA, CAN and AUS','y':0.95,'x':0.5, 'xanchor': 'center','yanchor':
'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Country', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.show()
```



Multi-variate Boxplot with actual points

In [27]:
```
fig = px.box(ex_rate_df[ex_rate_df['Country'].isin(['FRA','CAN','AUS'])], y="Rate", x='Country', color='Country',point
s='all')

fig.update_layout(title={'text': 'Exchange Rate for FRA, CAN and AUS','y':0.95,'x':0.5, 'xanchor': 'center','yanchor':
'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Country', yaxis_title='Rate',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.show()
```

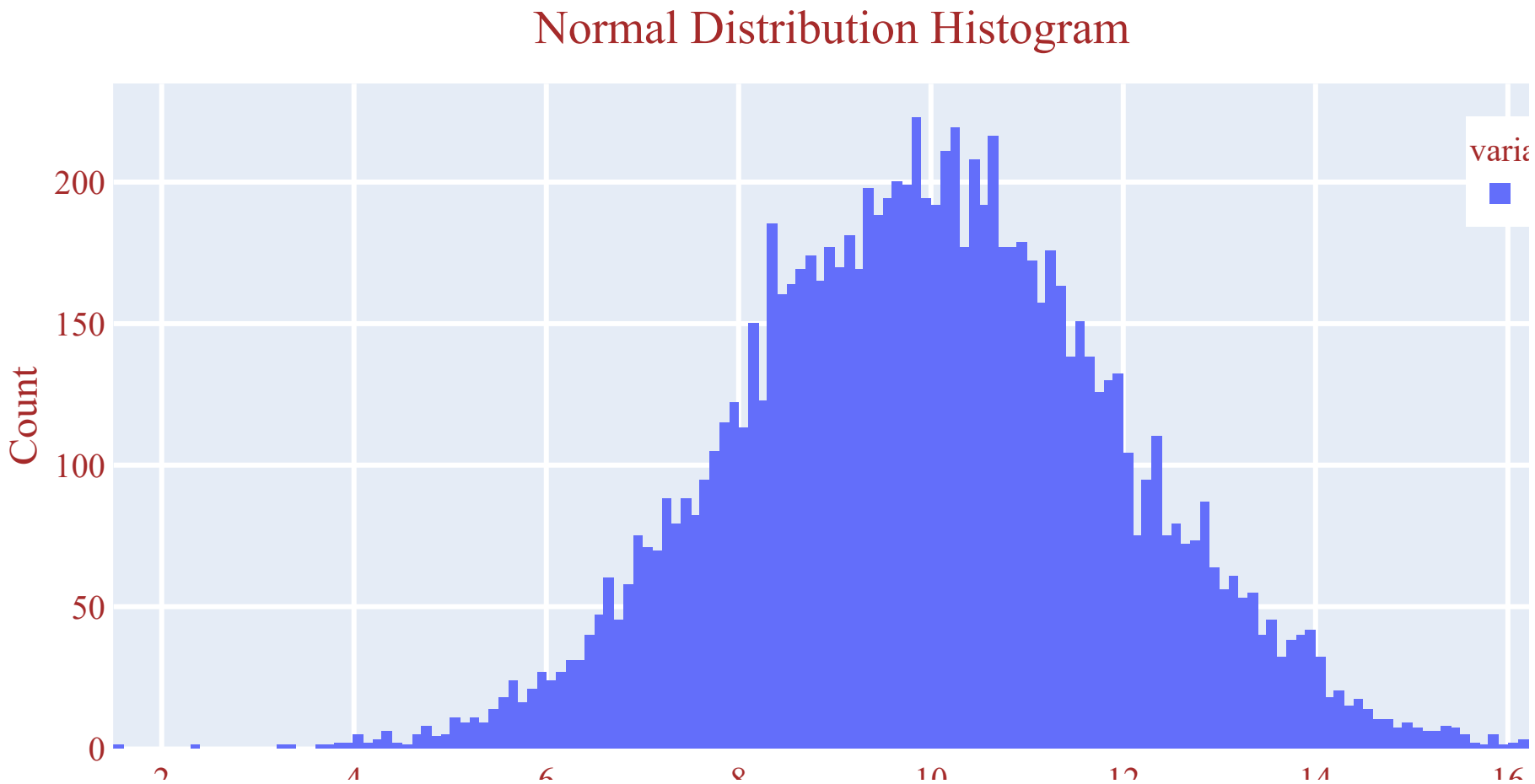


Distribution Plots in Plotly

Histogram

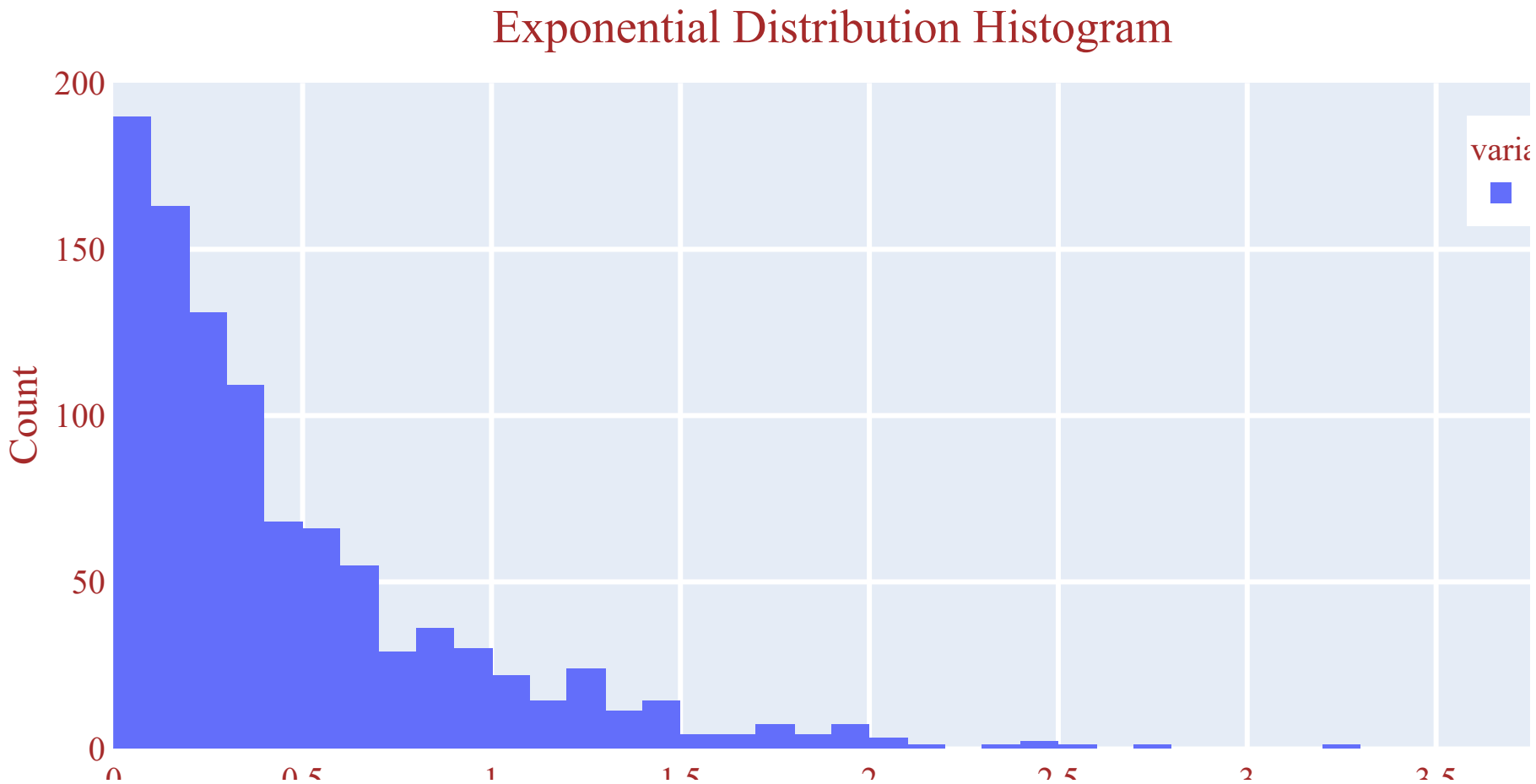
Simple Histogram

In [28]:
```
normal_distributed_variable=np.random.normal(10,2,10000)

fig = px.histogram(normal_distributed_variable)

fig.update_layout(title={'text': 'Normal Distribution Histogram','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Number', yaxis_title='Count',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Exponential Distribution in Plotly

In [29]:
```
exponential_distributed_variable = np.random.exponential(scale=0.5, size=(1000, 1))

fig = px.histogram(exponential_distributed_variable)

fig.update_layout(title={'text': 'Exponential Distribution Histogram','y':0.95,'x':0.5, 'xanchor': 'center','yanchor':
'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Number', yaxis_title='Count',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



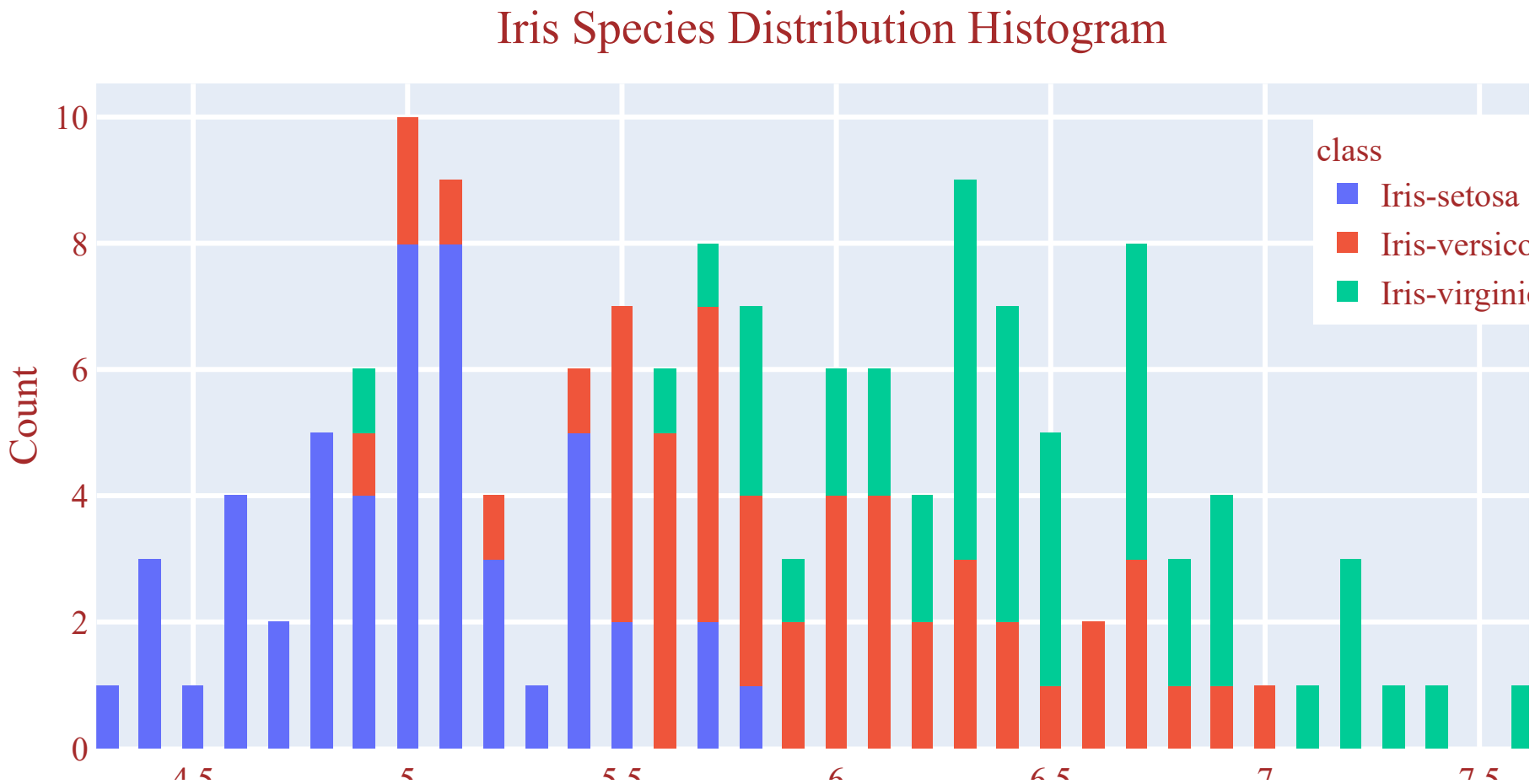
Multi-variate Histogram

In [30]:
```
iris_df=pd.read_csv('iris.csv')
iris_df.head()
```

Out[30]:

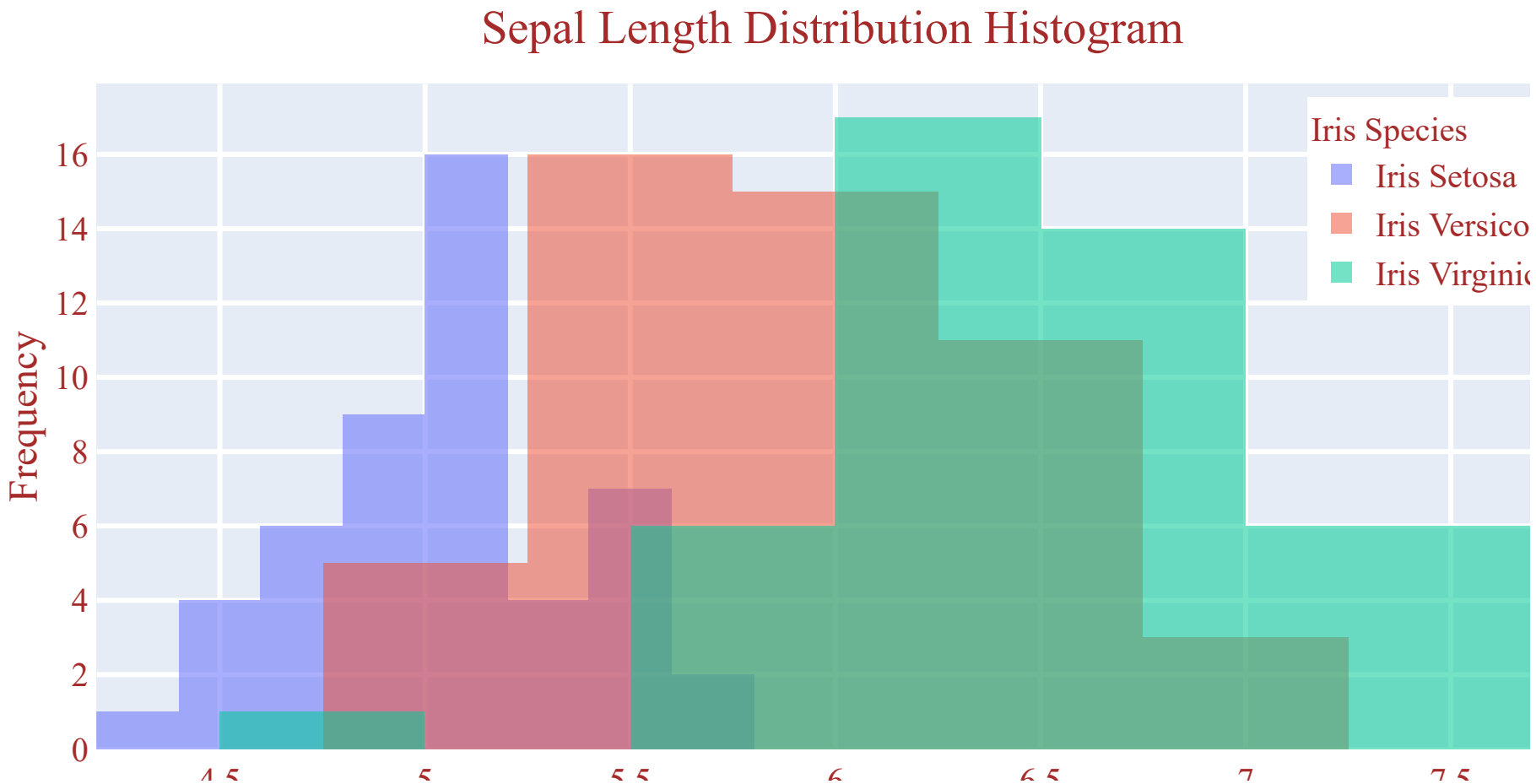
	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [31]:
```
fig = px.histogram(iris_df, x='sepal_length', color='class',nbins=80)

fig.update_layout(title={'text': 'Iris Species Distribution Histogram','y':0.95,'x':0.5, 'xanchor': 'center','yanchor'
: 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Number', yaxis_title='Count',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Multi-variate Histogram with Overlay

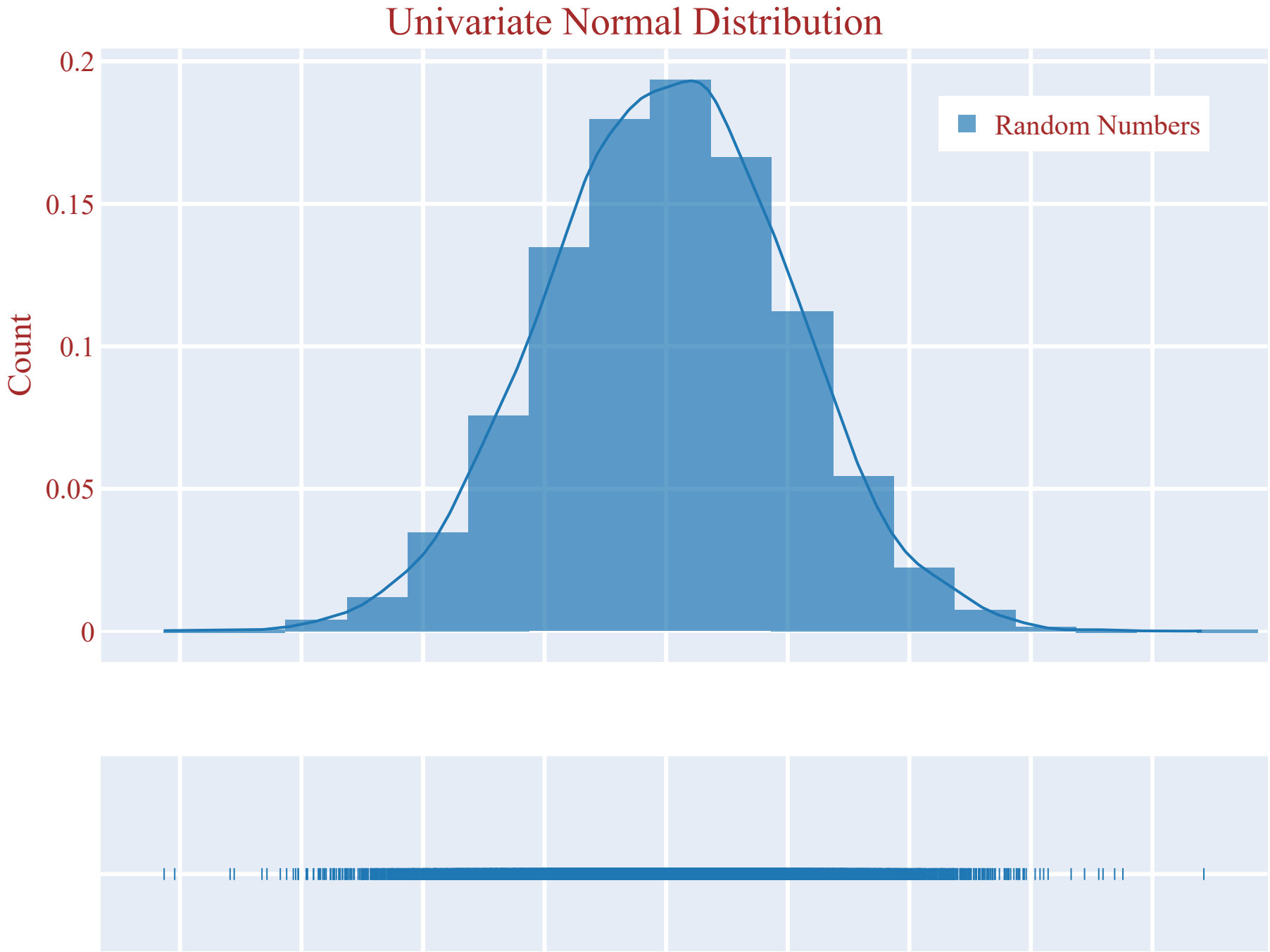
In [32]:
```
fig=go.Figure()
fig.add_trace(go.Histogram(x=iris_df[iris_df['class']=='Iris-setosa']['sepal_length'], name='Iris Setosa'))
fig.add_trace(go.Histogram(x=iris_df[iris_df['class']=='Iris-versicolor']['sepal_length'], name='Iris Versicolor'))
fig.add_trace(go.Histogram(x=iris_df[iris_df['class']=='Iris-virginica']['sepal_length'],name='Iris Virginica'))

fig.update_traces(opacity=0.55)
fig.update_layout(title={'text': 'Sepal Length Distribution Histogram','y':0.95,'x':0.5, 'xanchor': 'center','yanchor'
: 'top'},
                            legend=dict(title="Iris Species", yanchor="top",y=0.98,xanchor="right",x=0.95),
                   autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Frequency',
                   font=dict(size=20, family='Times New Romans', color='brown'),
                  barmode='overlay')

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```

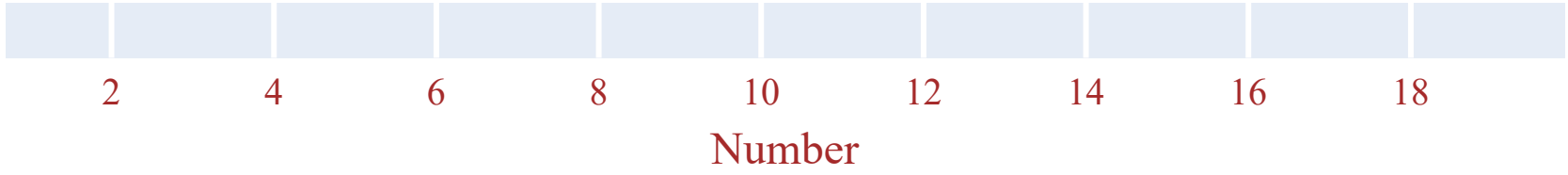


Distribution Plot

Univariate Normal Distribution Plot

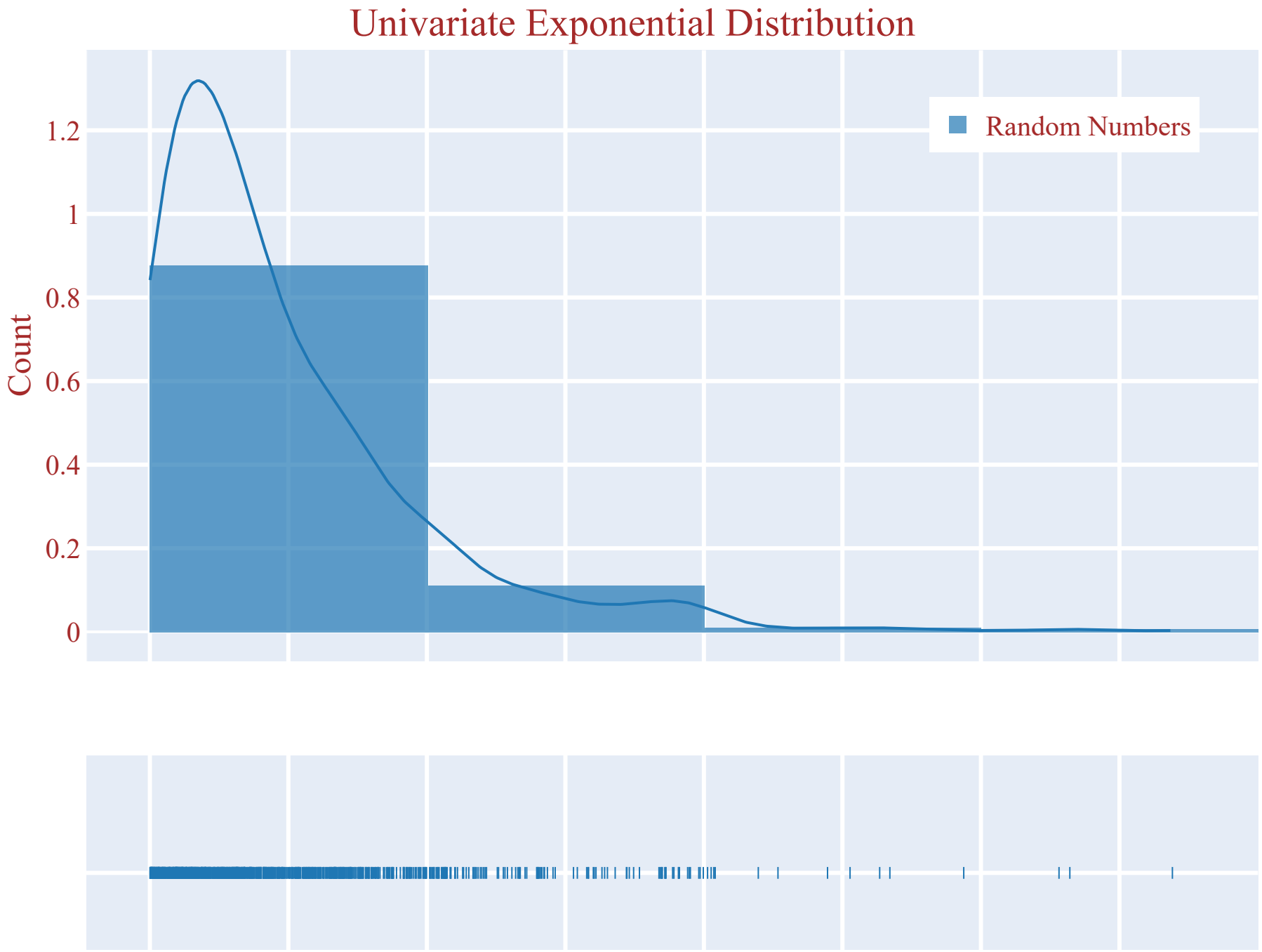
In [33]:
```
import plotly.figure_factory as ff

normal_distributed_variable=np.random.normal(10,2,10000)

fig = ff.create_distplot([normal_distributed_variable],group_labels=['Random Numbers'])

fig.update_layout(title={'text': 'Univariate Normal Distribution','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Number', yaxis_title='Count',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                 width=900, height=800)

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```

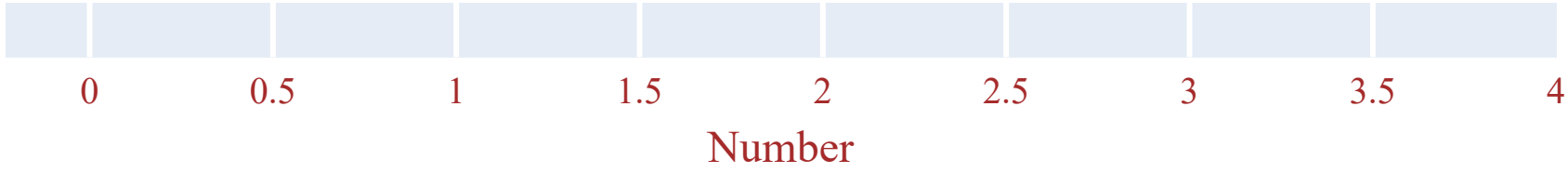

Univariate Normal Distribution
Random Numbers
Count
0.2
0.15
0.1
0.05
0



Univariate Exponential Distribution Plot

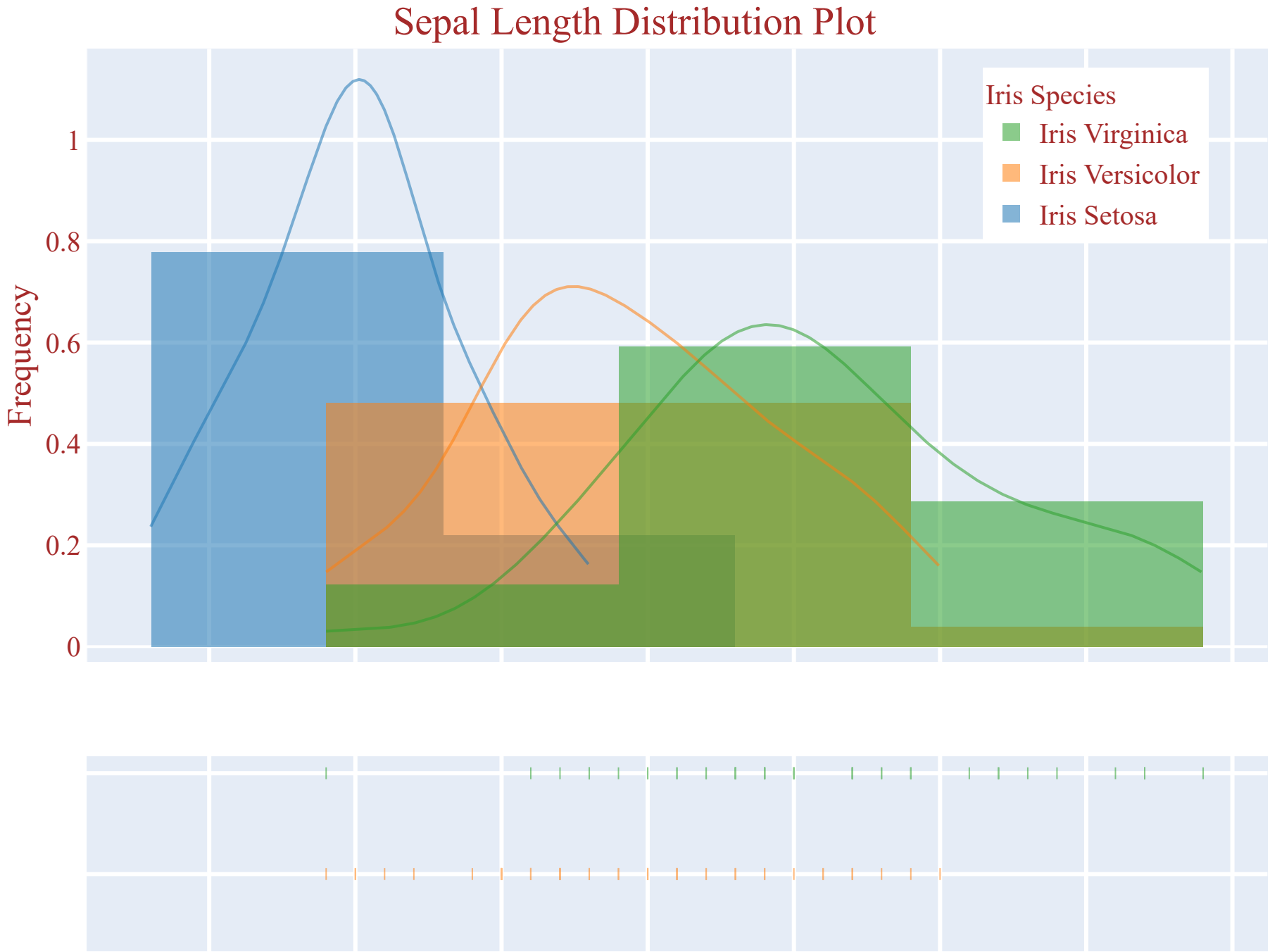
In [34]:
```
exponential_distributed_variable = np.random.exponential(scale=0.5, size=(1000, 1))
exponential_distributed_variable=exponential_distributed_variable[~np.isnan(exponential_distributed_variable)]

fig = ff.create_distplot([exponential_distributed_variable], group_labels=['Random Numbers'])

fig.update_layout(title={'text': 'Univariate Exponential Distribution','y':0.95,'x':0.5, 'xanchor': 'center','yanchor'
: 'top'},
                            legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Number', yaxis_title='Count',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                 width=900, height=800)

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```


Univariate Exponential Distribution
Random Numbers
Count
1.2
1
0.8
0.6
0.4
0.2
0



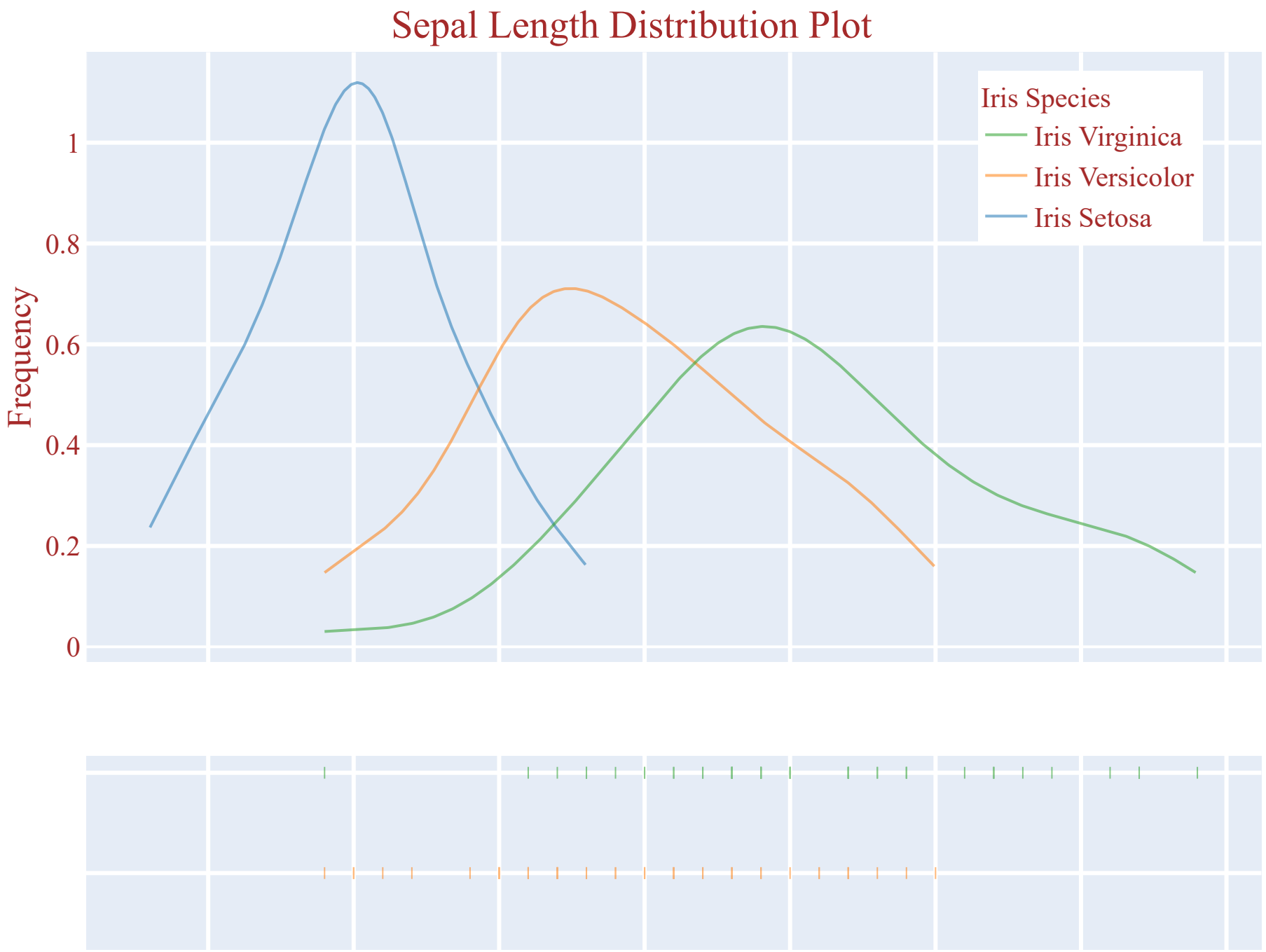
Multi-Variate Distribution Plot

In [35]:
```
fig=go.Figure()

dist_data=[iris_df[iris_df['class']=='Iris-setosa']['sepal_length'], iris_df[iris_df['class']=='Iris-versicolor']['sep
al_length'],
          iris_df[iris_df['class']=='Iris-virginica']['sepal_length']]
grop_labels=['Iris Setosa','Iris Versicolor','Iris Virginica']

fig = ff.create_distplot(dist_data,grop_labels)

fig.update_traces(opacity=0.55)
fig.update_layout(title={'text': 'Sepal Length Distribution Plot','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
                          legend=dict(title="Iris Species", yanchor="top",y=0.98,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Frequency',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                  width=900, height=800
                 )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```


Sepal Length Distribution Plot
Iris Species
Iris Virginica
Iris Versicolor
Iris Setosa
Frequency
1
0.8
0.6
0.4
0.2
0

Multi-Variate Distribution Plot without Histogram

In [36]:
```
fig=go.Figure()

dist_data=[iris_df[iris_df['class']=='Iris-setosa']['sepal_length'], iris_df[iris_df['class']=='Iris-versicolor']['sep
al_length'],
          iris_df[iris_df['class']=='Iris-virginica']['sepal_length']]
grop_labels=['Iris Setosa','Iris Versicolor','Iris Virginica']

fig = ff.create_distplot(dist_data,grop_labels,show_hist=False)

fig.update_traces(opacity=0.55)
fig.update_layout(title={'text': 'Sepal Length Distribution Plot','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
                          legend=dict(title="Iris Species", yanchor="top",y=0.98,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Sepal Length', yaxis_title='Frequency',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                  width=900, height=800
                 )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```


Sepal Length Distribution Plot
Frequency
1
0.8
0.6
0.4
0.2
0
Iris Species
Iris Virginica
Iris Versicolor
Iris Setosa

Heatmap in Plotly

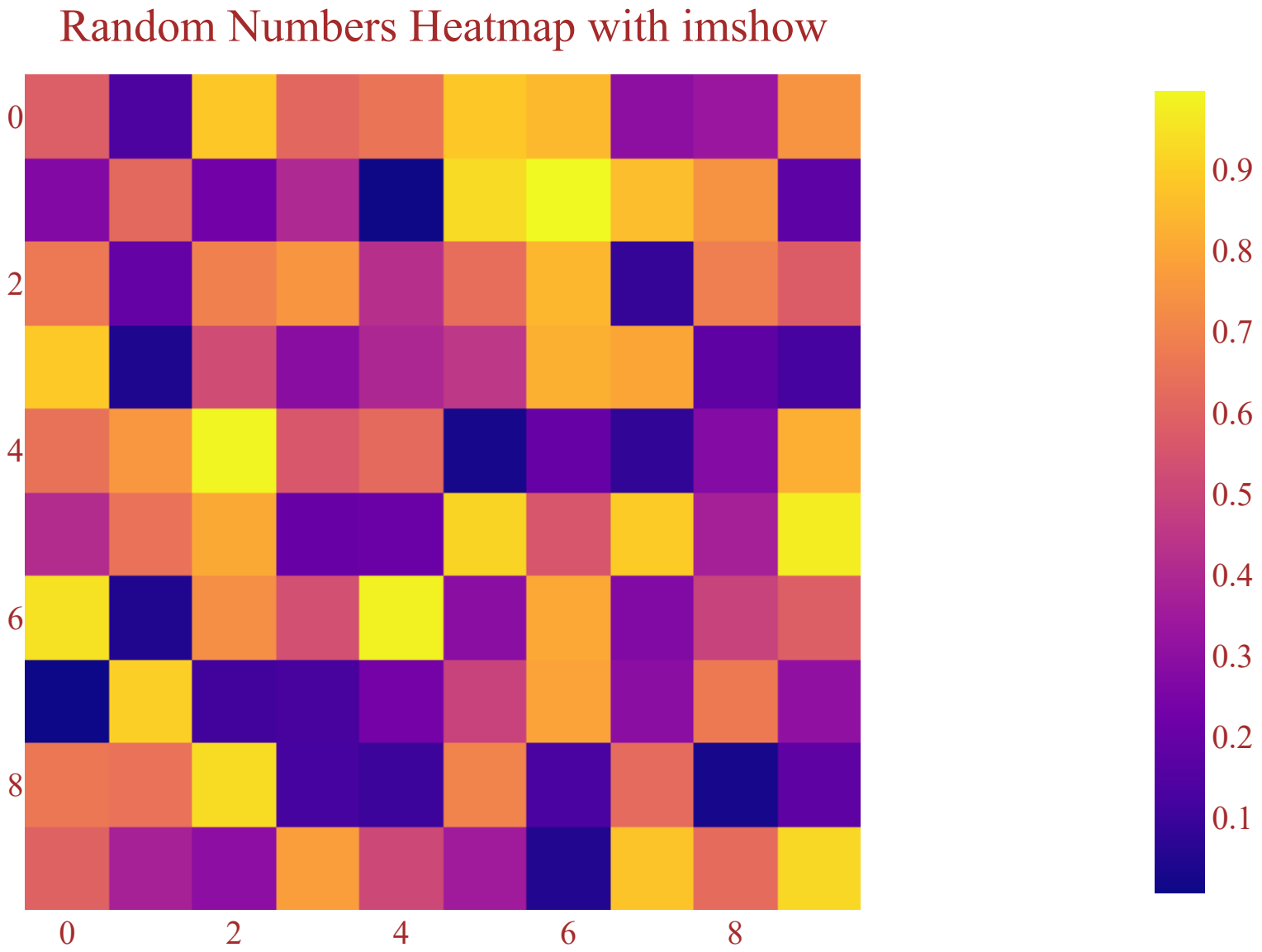
Heatmap with imshow

In [37]:
```
ex_rate_df=pd.read_csv('Exchange_Rates.csv')
heatmap_ex_rate_df= pd.crosstab(ex_rate_df['TIME'],ex_rate_df['LOCATION'],values=ex_rate_df['Value'],aggfunc='mean')

heatmap_ex_rate_df.iloc[0:,0:13]
```

Out[37]:

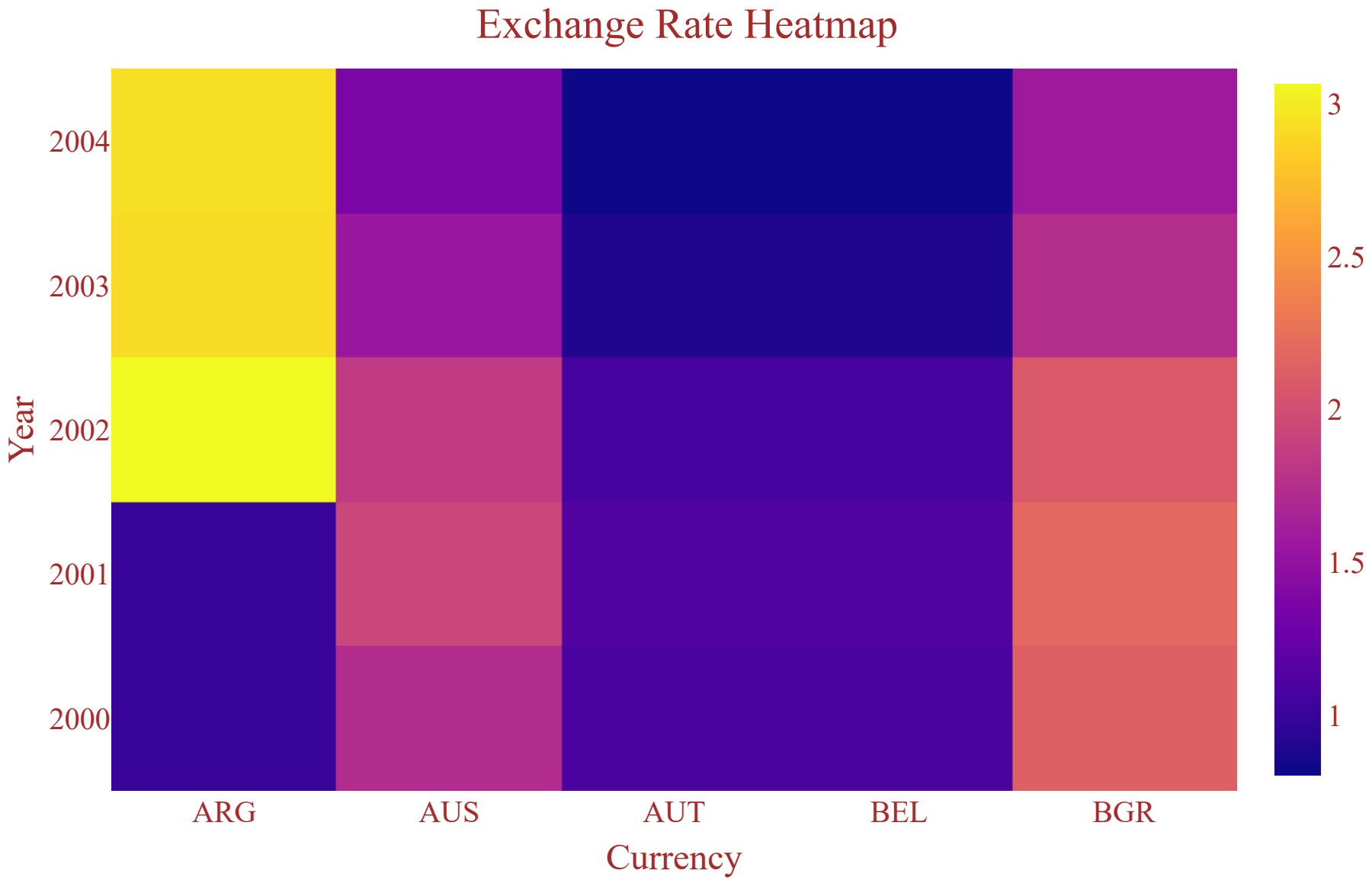
LOCATION	ARG	AUS	AUT	BEL	BGR	BRA	CAN	CHE	CHL	CHN	COL	
TIME												
2000	0.999500	1.724827	1.082705	1.082705	2.123275	1.829423	1.485394	1.688843	539.587500	8.278504	2087.903842	308.
2001	0.999500	1.933443	1.116533	1.116533	2.184708	2.349632	1.548840	1.687615	634.938333	8.277068	2299.633156	328.
2002	3.063257	1.840563	1.057559	1.057559	2.076975	2.920363	1.570343	1.558607	688.936667	8.276958	2504.241331	359.
2003	2.900629	1.541914	0.884048	0.884048	1.732702	3.077475	1.401015	1.346651	691.397500	8.277037	2877.652458	398.
2004	2.923301	1.359752	0.803922	0.803922	1.575109	2.925119	1.301282	1.243496	609.529167	8.276801	2628.612903	437.
2005	2.903658	1.309473	0.803800	0.803800	1.574133	2.434390	1.211405	1.245177	559.767500	8.194317	2320.834177	477.
2006	3.054313	1.327973	0.796433	0.796433	1.559267	2.175327	1.134345	1.253843	530.275000	7.973438	2361.139407	511.
2007	3.095649	1.195073	0.729672	0.729672	1.429050	1.947058	1.074046	1.200366	522.464167	7.607532	2078.291837	516.
2008	3.144165	1.192178	0.679923	0.679923	1.337117	1.833767	1.067087	1.083090	522.461036	6.948655	1967.711309	526.
2009	3.710107	1.282189	0.716958	0.716958	1.406692	1.999428	1.141535	1.088142	560.859895	6.831416	2158.255903	573.
2010	3.896295	1.090159	0.754309	0.754309	1.477392	1.759227	1.030113	1.042906	510.249167	6.770269	1898.569636	525.
2011	4.110140	0.969463	0.718414	0.718414	1.406458	1.672829	0.989258	0.888042	483.667500	6.461461	1848.139470	505.
2012	4.536934	0.965801	0.778338	0.778338	1.522050	1.953069	0.999365	0.937684	486.471303	6.312333	1796.895912	502.
2013	5.459353	1.035843	0.752945	0.752945	1.473567	2.156089	1.030137	0.926904	495.272878	6.195758	1868.785327	499.
2014	8.075276	1.109363	0.752728	0.752728	1.474183	2.352952	1.104747	0.916151	570.348216	6.143434	2001.781048	538.
2015	9.233186	1.331090	0.901296	0.901296	1.764400	3.326904	1.278786	0.962381	654.124084	6.227489	2741.880855	534.
2016	14.758175	1.345214	0.903421	0.903421	1.768042	3.491313	1.325615	0.985394	676.957736	6.644478	3054.121673	544.
2017	16.562707	1.304758	0.885206	0.885206	1.735458	3.191389	1.297936	0.984692	648.833793	6.758755	2951.327402	567.
2018	28.094992	1.338412	0.846773	0.846773	1.657042	3.653825	1.295818	0.977883	641.276813	6.615957	2955.703970	576.
2019	48.147892	1.438507	0.893276	0.893276	1.747042	3.944471	1.326793	0.993775	702.897423	6.908385	3280.831631	587.
2020	70.539167	1.453085	0.875506	0.875506	1.716333	5.155179	1.341153	0.938895	792.727206	6.900767	3694.854072	584.

In [38]:
```
heatmap_data = np.random.random(( 10 , 10 ))

fig = px.imshow(heatmap_data)

fig.update_layout(title={'text': 'Random Numbers Heatmap with imshow','y':0.95,'x':0.45, 'xanchor': 'center','yanchor'
: 'top'},
                            legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                   autosize=True,margin=dict(t=70,b=0,l=0,r=0),
                   font=dict(size=20, family='Times New Romans', color='brown'),
                  width=900, height=600)

fig.show()
```



Heatmap with Plotly Graph Object

In [39]:
```
fig=go.Figure()
fig.add_trace(go.Heatmap(z=heatmap_ex_rate_df.iloc[0:5,0:5],
                         x=heatmap_ex_rate_df.iloc[0:5,0:5].columns,
                         y=heatmap_ex_rate_df.iloc[0:5,0:5].index))

fig.update_layout(title={'text': 'Exchange Rate Heatmap','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                           legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                   autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Currency', yaxis_title='Year',
                   font=dict(size=20, family='Times New Romans', color='brown'),
                  width=900, height=600)

fig.show()
```



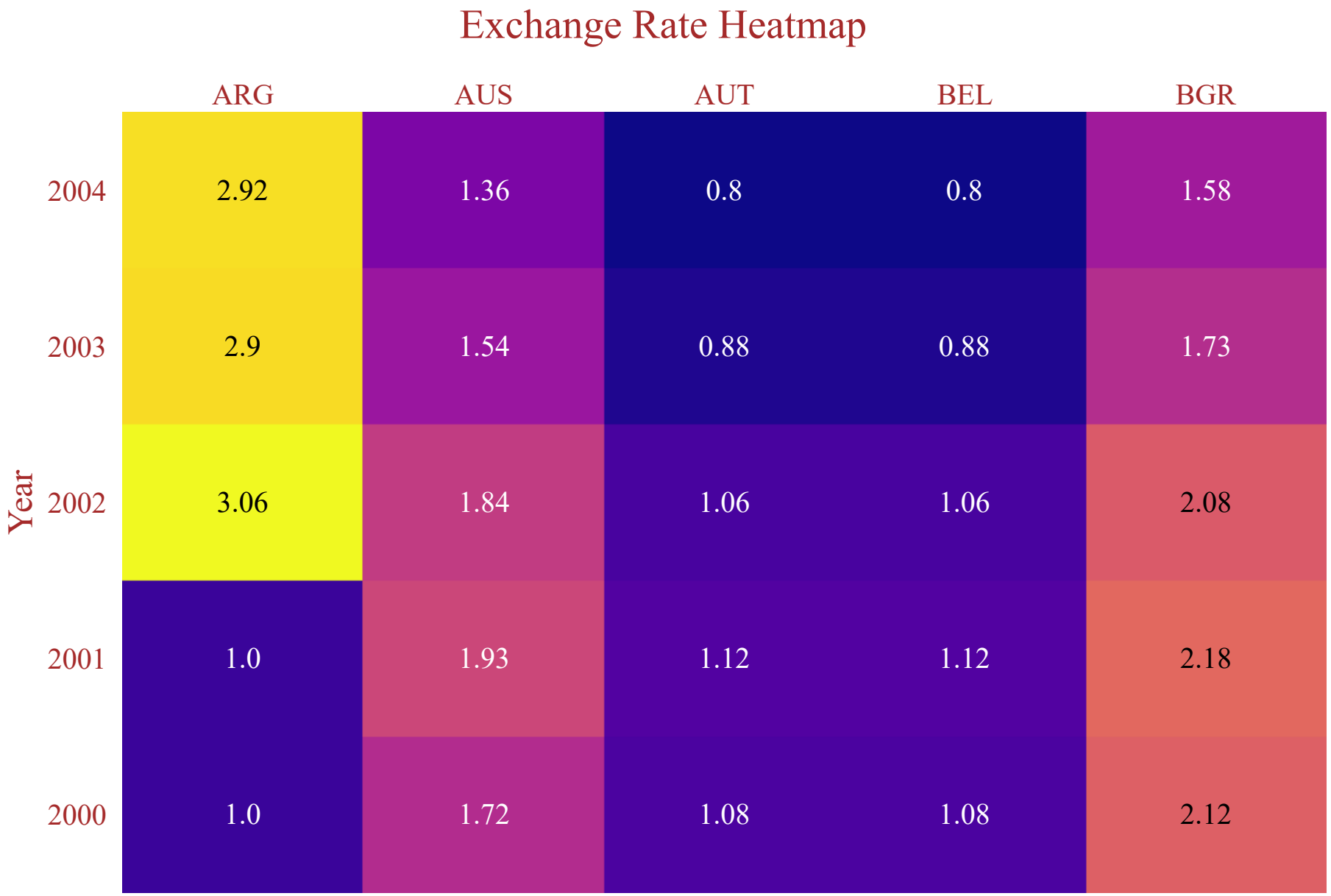
Annotated Heatmap

In [40]:
```
import plotly.figure_factory as ff
heatmap_data = np.random.random(( 10 , 10 ))

fig=ff.create_annotated_heatmap(heatmap_data.round(2))

fig.update_layout(title={'text': 'Random Numbers Heatmap with Figure Factory','y':1.0,'x':0.5,
                         'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                 width=900, height=600)

fig.show()
```

Random Numbers Heatmap with Figure Factory

Annotated Heatmap with Pandas DataFrame

In [41]:
```
z = heatmap_ex_rate_df.iloc[0:5,0:5].round(2).values.tolist()
x = heatmap_ex_rate_df.iloc[0:5,0:5].columns.tolist()
y = heatmap_ex_rate_df.iloc[0:5,0:5].index.tolist()

fig=ff.create_annotated_heatmap(z,x=x,y=y, annotation_text=z)
fig.update_layout(title={'text': 'Exchange Rate Heatmap','y':1.0,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                          legend=dict(yanchor="top",y=0.95,xanchor="right",x=0.95),
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='', yaxis_title='Year',
                  font=dict(size=20, family='Times New Romans', color='brown'),
                 width=900, height=600)

fig.show()
```



Treemap in Plotly

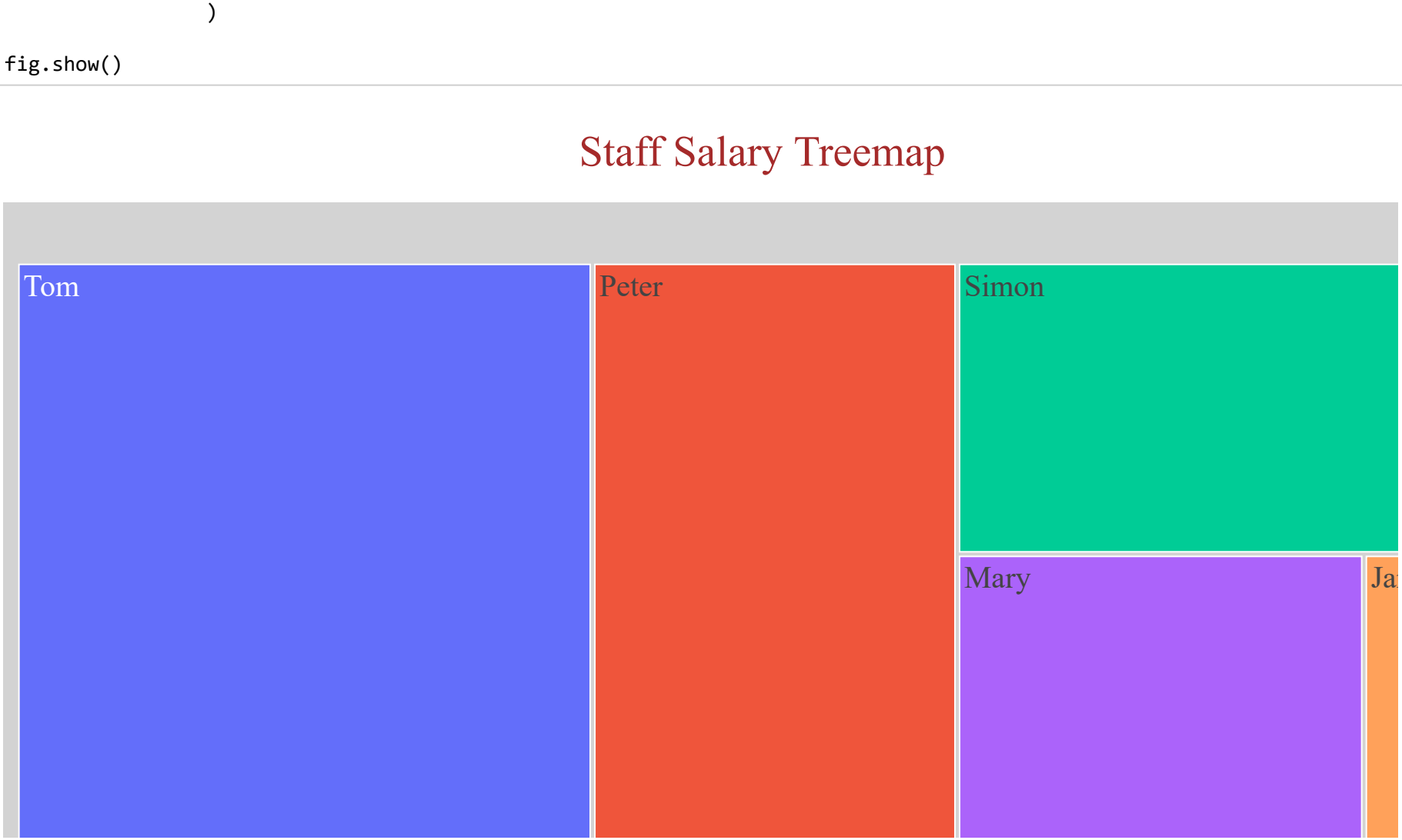
In [42]:
```
salary_df = pd.DataFrame(
    {
        "Department": ["Finance","Technology","Finance","Technology","Technology"],
        "Staff": ["Tom", "Peter","Simon", "Mary", "Jane"],
        "Salary": [90000.00, 57000.00,40000.00, 34000.00, 12000.00]
    },
)

salary_df
```

Out[42]:

	Department	Staff	Salary
0	Finance	Tom	90000.0
1	Technology	Peter	57000.0
2	Finance	Simon	40000.0
3	Technology	Mary	34000.0
4	Technology	Jane	12000.0

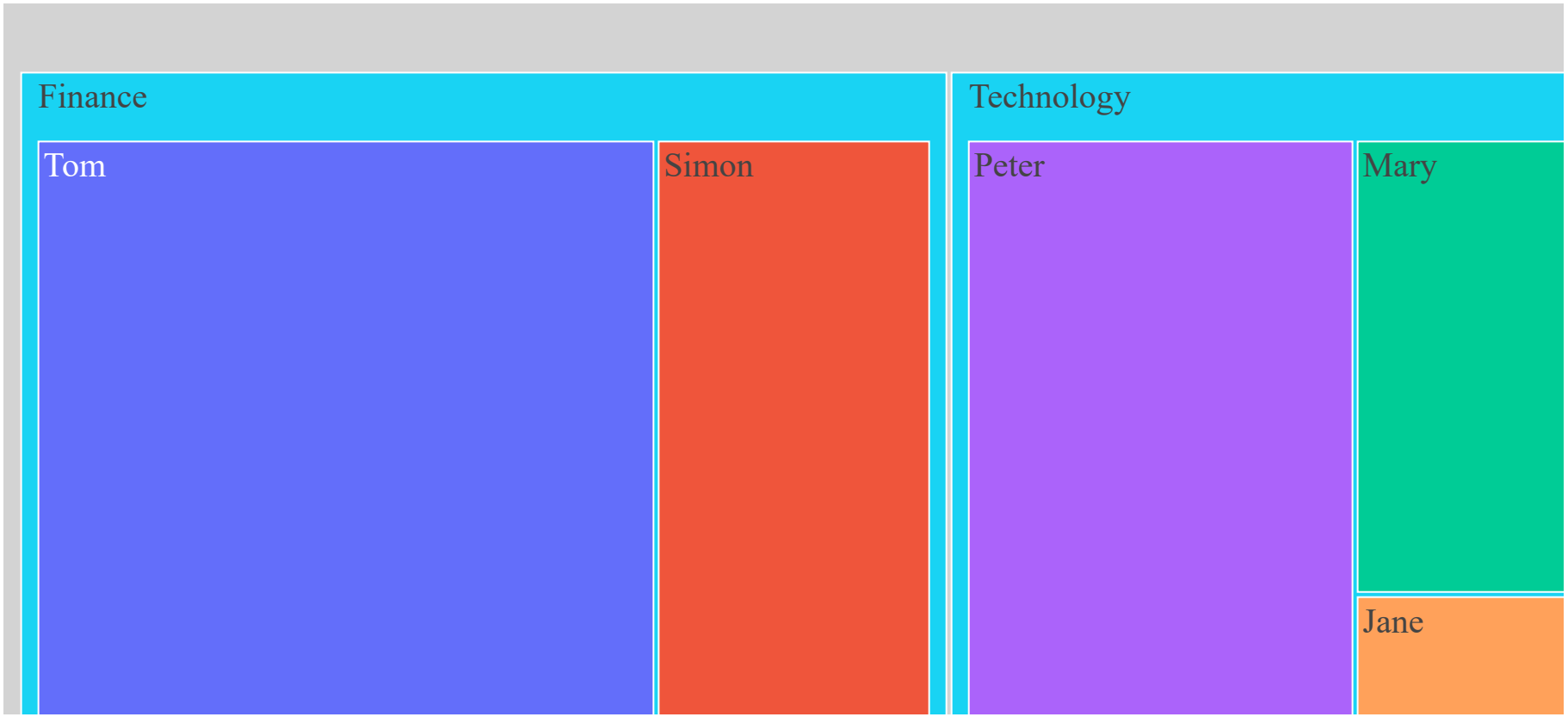
Univariate Treemap with Plotly Express

In [43]:
```
fig = px.treemap(salary_df, path=['Staff'], values='Salary')
fig.update_traces(root_color="lightgrey")

fig.update_layout(title={'text': 'Staff Salary Treemap','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0),
                  font=dict(size=20, family='Times New Romans', color='brown'),
                 )

fig.show()
```



Multi-variate Treemap with Plotly Express

In [44]:
```
fig = px.treemap(salary_df, path=['Department','Staff'], values='Salary',
                 color='Staff', color_continuous_scale='RdBu',)
fig.update_traces(root_color="lightgrey")

fig.update_layout(title={'text': 'Department Staff Salary Treemap','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'top'},
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0),
                  font=dict(size=20, family='Times New Romans', color='brown'),
                 )

fig.update_xaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)
fig.update_yaxes(showline=True, linewidth=1, linecolor='white', gridwidth=3, gridcolor='white', mirror=True)

fig.show()
```



Maps in Plotly

choropleth Map with Plotly Express

In [45]:
```
sentiment_polarity_df=pd.read_csv('datasets/sentiment_polarity.csv')
sentiment_polarity_df.head()
```

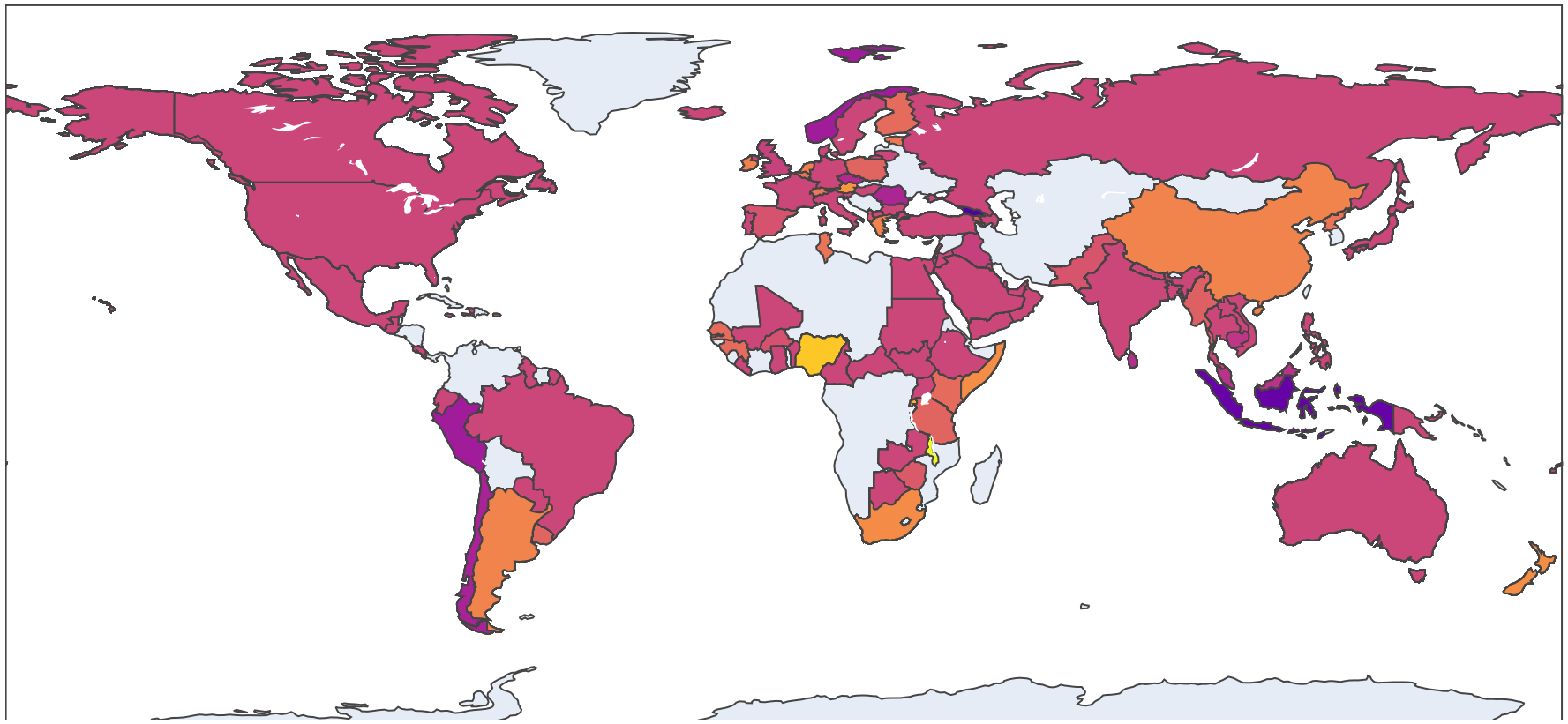
Out[45]:

	Unnamed: 0	Unnamed: 0.1	screen_name	name	user_verification	followers_count	friends_count	listed_count	retweet_coun
0	0	0	nat___price	natalie price	False	62	732	0	2
1	1	1	libertad717	Punto	False	1084	4760	468	11
2	2	2	marylouisepearc	marylouise lady of leisure	False	124	244	3	7
3	3	3	JhSalford	JHSalford	False	177	177	0	336
4	4	4	SarahEdmondsPhD	Fear is the MindKiller	False	817	1861	9	387

5 rows × 24 columns

The 'scope' property is an enumeration that may be specified as - One of the following enumeration values: ['world', 'usa', 'europe', 'asia', 'africa', 'north america', 'south america']

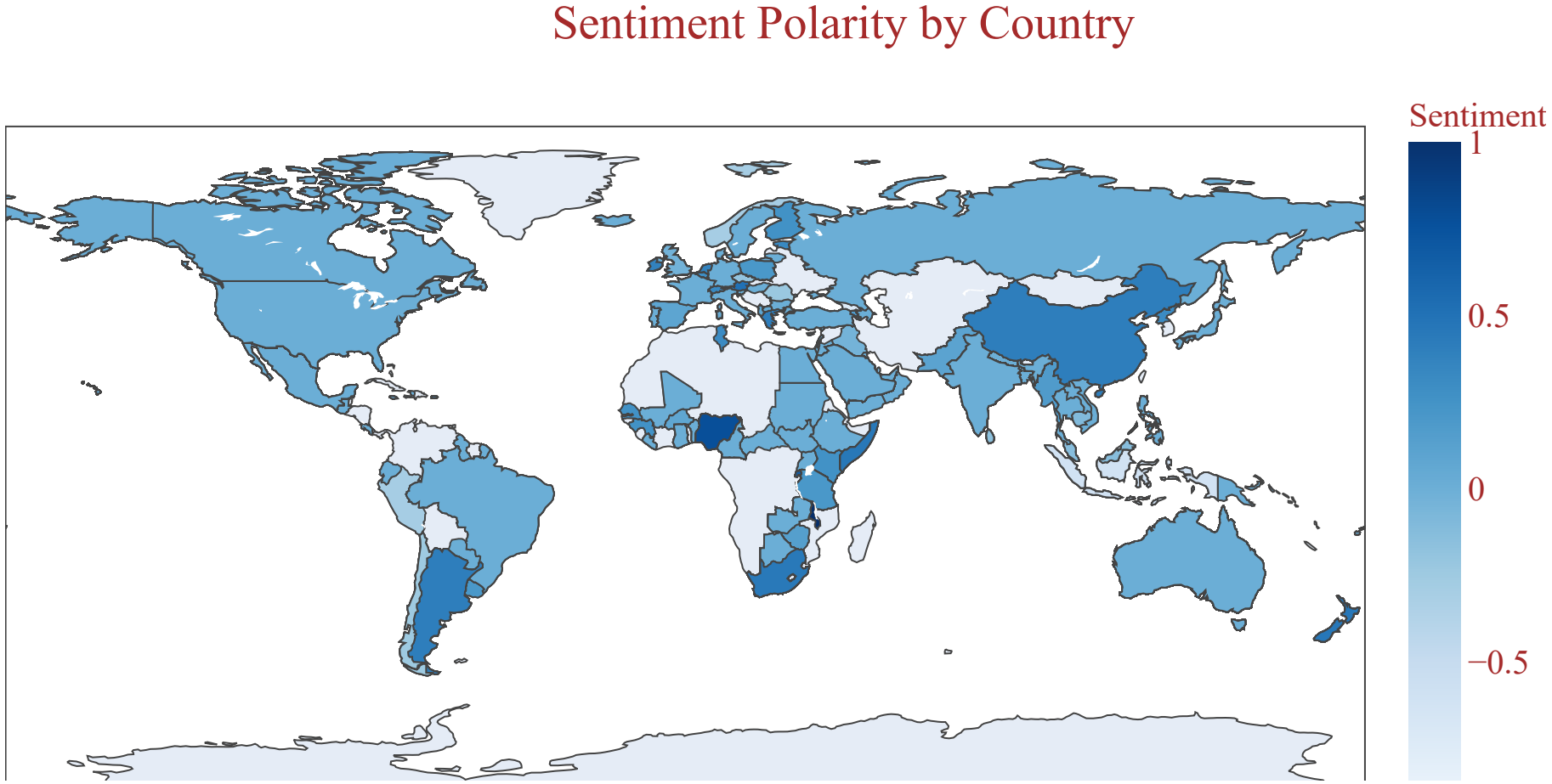
In [46]:
```
fig = px.choropleth(locations=sentiment_polarity_df['country'], color=sentiment_polarity_df['sentiment_polarity'],
                    locationmode="country names", scope="world",
                    hover_name=sentiment_polarity_df['country'])

fig.update_layout(title={'text': 'Sentiment Polarity by Country','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Year',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.show()
```



choropleth Map with Plotly Graph Object

The 'locationmode' property is an enumeration that may be specified as - One of the following enumeration values: ['ISO-3', 'USA-states', 'country names', 'geojson-id']

In [47]:
```
fig = go.Figure(data=go.Choropleth(
    locations=sentiment_polarity_df['country'],
    z = sentiment_polarity_df['sentiment_polarity'].astype(float),
    locationmode = 'country names',
    colorscale = 'Blues',
    colorbar_title = "Sentiment Polarity",
    #     text=df['country']
    ))

fig.update_layout(title={'text': 'Sentiment Polarity by Country','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
                  autosize=True,margin=dict(t=70,b=0,l=0,r=0), xaxis_title='Year', yaxis_title='Year',
                  font=dict(size=20, family='Times New Romans', color='brown') )

fig.show()
```



Mapbox map with Plotly Express

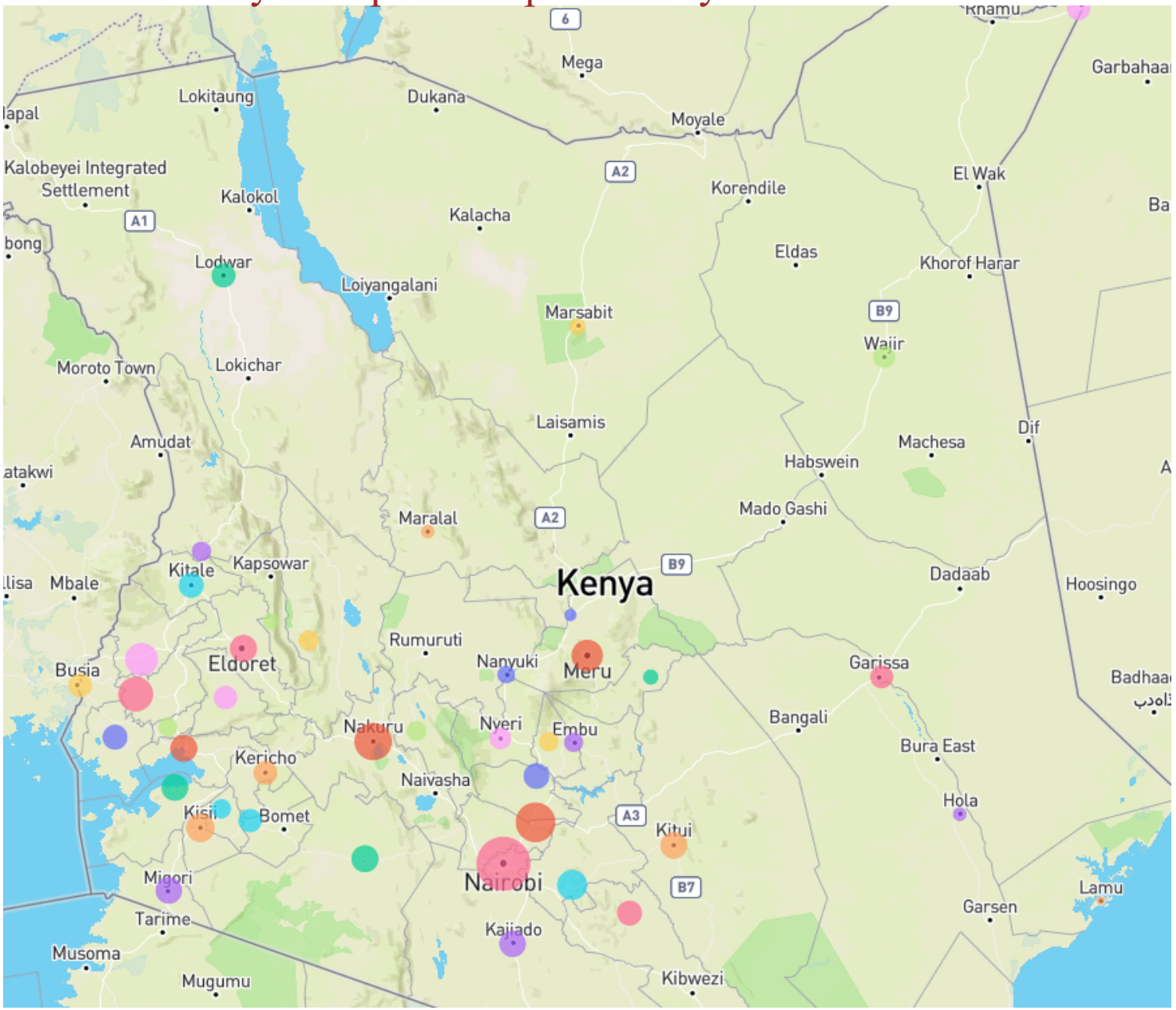
In [48]:
```
kenya_county_population = pd.read_csv('datasets/kenyan_population_census_2019.csv')
kenya_county_population.head()
```

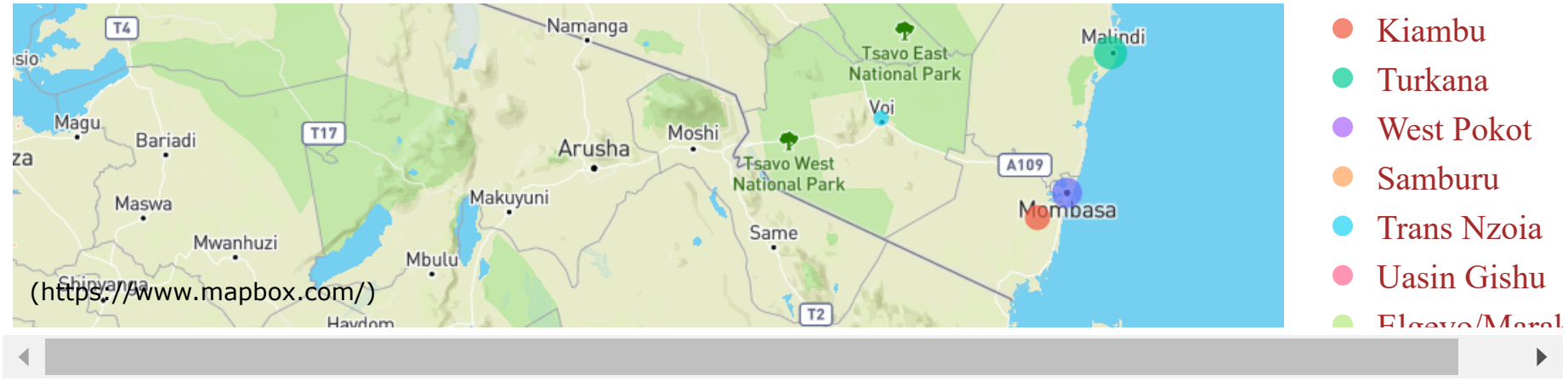
Out[48]:

	County	Male	Female	Intersex	Total Population	Town	lat	lng
0	Mombasa	610257	598046	30	1208333	Mombasa	-4.0500	39.6667
1	Kwale	425121	441681	18	866820	Kwale	-4.1737	39.4521
2	Kilifi	704089	749673	25	1453787	Malindi	-3.2100	40.1000
3	Tana River	158550	157391	2	315943	Tana River	-1.5000	40.0300
4	Lamu	76103	67813	4	143920	Lamu	-2.2686	40.9003

In [49]:
```
mapbox_access_token = open("mapboxtoken.mapbox_token").read()

fig = px.scatter_mapbox(kenya_county_population, lat="lat", lon="lng", color="County",
                        size='Total Population',
                   color_continuous_scale=px.colors.cyclical.IceFire,
                        size_max=25, zoom=5,
                       hover_name="County")

fig.update_layout( mapbox=dict( accesstoken=mapbox_access_token,
                                bearing=10, pitch=0, zoom=6, style='outdoors' ,
                                center=go.layout.mapbox.Center( lat=0.05, lon=37.65 )),
                   width=950, height=900,
    title={'text': 'Kenyan Population per County for 2019 Census','y':0.95,'x':0.4, 'xanchor': 'center','yanchor': 'to
p'},
    autosize=True,margin=dict(t=70,b=0,l=0,r=0),
    font=dict(size=20, family='Times New Romans', color='brown') )

fig.show()
```

Kenyan Population per County for 2019 Census

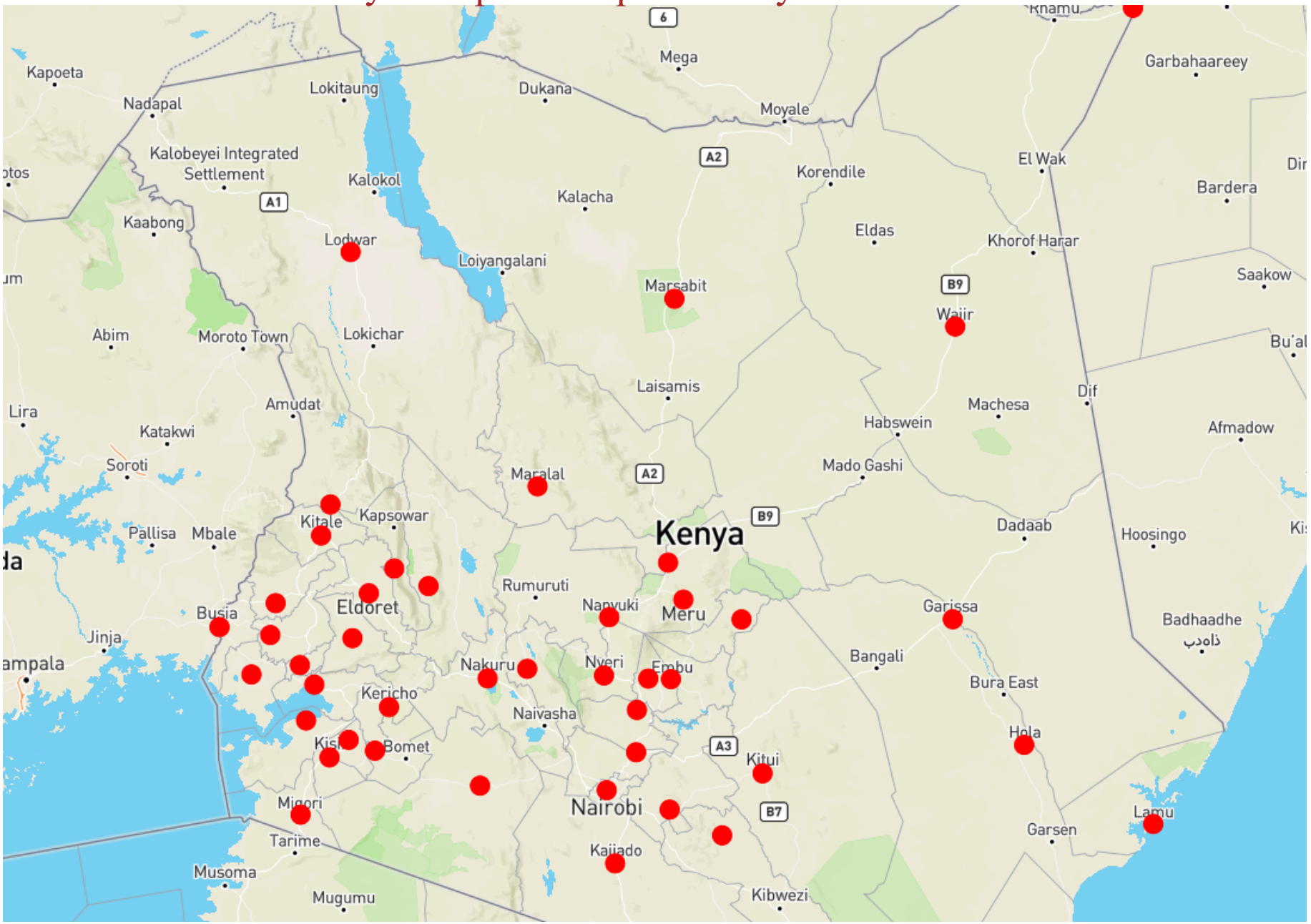




Mapbox map with Plotly Graph Object

In [50]:
```
mapbox_access_token = open("mapboxtoken.mapbox_token").read()

fig = go.Figure(go.Scattermapbox(lat=kenya_county_population["lat"], lon=kenya_county_population["lng"],
        mode="markers", hovertext=kenya_county_population['Total Population'],
        marker=go.scattermapbox.Marker(size=15, color = 'red'),

    ))

fig.update_layout( mapbox=dict( accesstoken=mapbox_access_token,
                                bearing=10, pitch=0, zoom=6, style='streets' ,
                                center=go.layout.mapbox.Center( lat=0.05, lon=37.65 )),
                  width=950, height=900,
    title={'text': 'Kenyan Population per County for 2019 Census','y':0.95,'x':0.5, 'xanchor': 'center','yanchor': 'to
p'},
    autosize=True,margin=dict(t=70,b=0,l=0,r=0),
    font=dict(size=20, family='Times New Romans', color='brown') )

fig.show()
```

Kenyan Population per County for 2019 Census




T4
Nansio
Magu
Bariadi
Mwanza
T17
Namanga
Arusha
Moshi
Tsavo East
National Park
Malindi
Voi
Tsavo West
National Park
A109
Mombasa
Geita
Maswa
Makuyuni
Same
T4
Mwanhuzi
Mbulu
(https://www.mapbox.com/)
T2