Plotly Implementation On 3 Data Sets

Plotly

The Plotly Python library is an interactive open-source library. This can be a very helpful tool for data visualization and understanding the data simply and easily. plotly graph objects are a high-level interface to plotly which are easy to use. It can plot various types of graphs and charts like scatter plots, line charts, bar charts, box plots, histograms, pie charts, etc.

Why plotly over other visualization tools or libraries?

Plotly has hover tool capabilities that allow us to detect any outliers or anomalies in a large number of data points. It is visually attractive that can be accepted by a wide range of audiences. It allows us for the endless customization of our graphs that makes our plot more meaningful and understandable for others.

The Plotly Implemetation of following graphs are shown here-:

1. Line Charts
2. Scatter Plot
3. Bar Charts
4. Bubble Charts
5. Histogram
6. Box Plots

In [1]:
```
import numpy as np
import pandas as pd
import plotly as py
from plotly.offline import init_notebook_mode, iplot, plot

init_notebook_mode(connected=True)
import plotly.graph_objs as go
import matplotlib.pyplot as plt
```

In [2]:
```
daily_data = pd.read_csv(r"C:\Users\dell\Downloads\DailyDelhiClimateTrain.csv")

monthly_data = pd.read_csv(r"C:\Users\dell\Downloads\salesmonthly.csv")

annual_data = pd.read_csv(r"C:\Users\dell\Downloads\archive (2)\annual_csv.csv")
```

In [3]:
```
daily_data.head(10)
```

Out[3]:

	date	meantemp	humidity	wind_speed	meanpressure
0	2013-01-01	10.000000	84.500000	0.000000	1015.666667
1	2013-01-02	7.400000	92.000000	2.980000	1017.800000
2	2013-01-03	7.166667	87.000000	4.633333	1018.666667
3	2013-01-04	8.666667	71.333333	1.233333	1017.166667
4	2013-01-05	6.000000	86.833333	3.700000	1016.500000
5	2013-01-06	7.000000	82.800000	1.480000	1018.000000
6	2013-01-07	7.000000	78.600000	6.300000	1020.000000
7	2013-01-08	8.857143	63.714286	7.142857	1018.714286
8	2013-01-09	14.000000	51.250000	12.500000	1017.000000
9	2013-01-10	11.000000	62.000000	7.400000	1015.666667

In [4]:
```
monthly_data.head(10)
```

Out[4]:

	datum	M01AB	M01AE	N02BA	N02BE	N05B	N05C	R03	R06
0	2014-01-31	127.69	99.090	152.100	878.030	354.0	50.0	112.0	48.2
1	2014-02-28	133.32	126.050	177.000	1001.900	347.0	31.0	122.0	36.2
2	2014-03-31	137.44	92.950	147.655	779.275	232.0	20.0	112.0	85.4
3	2014-04-30	113.10	89.475	130.900	698.500	209.0	18.0	97.0	73.7
4	2014-05-31	101.79	119.933	132.100	628.780	270.0	23.0	107.0	123.7
5	2014-06-30	112.07	94.710	122.900	548.225	323.0	23.0	57.0	109.3
6	2014-07-31	117.06	95.010	129.300	491.900	348.0	21.0	61.0	69.1
7	2014-08-31	134.79	99.780	123.800	583.850	420.0	29.0	37.0	70.8
8	2014-09-30	108.78	109.094	122.100	887.820	399.0	14.0	115.0	58.8
9	2014-10-31	154.75	185.241	191.600	1856.815	472.0	30.0	182.0	74.5

1AB - Anti-inflammatory and antirheumatic products, non-steroids, Acetic acid derivatives and related substances

M01AE - Anti-inflammatory and antirheumatic products, non-steroids, Propionic acid derivatives

N02BA - Other analgesics and antipyretics, Salicylic acid and derivatives

N02BE/B - Other analgesics and antipyretics, Pyrazolones and Anilides

N05B - Psycholeptics drugs, Anxiolytic drugs

N05C - Psycholeptics drugs, Hypnotics and sedatives drugs

R03 - Drugs for obstructive airway diseases

R06 - Antihistamines for systemic use

In [5]:
```
annual_data.head(10)
```

Out[5]:

	Date	Price
0	1950-12	34.72
1	1951-12	34.66
2	1952-12	34.79
3	1953-12	34.85
4	1954-12	35.04
5	1955-12	34.97
6	1956-12	34.90
7	1957-12	34.99
8	1958-12	35.09
9	1959-12	35.05

In [6]:
```
daily_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1462 entries, 0 to 1461
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   date          1462 non-null   object 
 1   meantemp      1462 non-null   float64
 2   humidity      1462 non-null   float64
 3   wind_speed    1462 non-null   float64
 4   meanpressure  1462 non-null   float64
dtypes: float64(4), object(1)
memory usage: 57.2+ KB
```

In [7]:
```
daily_data.isnull().sum()
```

Out[7]:
```
date            0
meantemp        0
humidity        0
wind_speed      0
meanpressure    0
dtype: int64
```

In [8]:
```
monthly_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   datum   70 non-null     object 
 1   M01AB   70 non-null     float64
 2   M01AE   70 non-null     float64
 3   N02BA   70 non-null     float64
 4   N02BE   70 non-null     float64
 5   N05B    70 non-null     float64
 6   N05C    70 non-null     float64
 7   R03     70 non-null     float64
 8   R06     70 non-null     float64
dtypes: float64(8), object(1)
memory usage: 5.0+ KB
```

In [9]:
```
monthly_data.isnull().sum()
```

Out[9]:
```
datum    0
M01AB    0
M01AE    0
N02BA    0
N02BE    0
N05B     0
N05C     0
R03      0
R06      0
dtype: int64
```

In [10]:
```
annual_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    70 non-null     object 
 1   Price   70 non-null     float64
dtypes: float64(1), object(1)
memory usage: 1.2+ KB
```

In [11]:
```
annual_data.isnull().sum()
```

Out[11]:
```
Date     0
Price    0
dtype: int64
```

In Daily Data, Data is of daily basis.

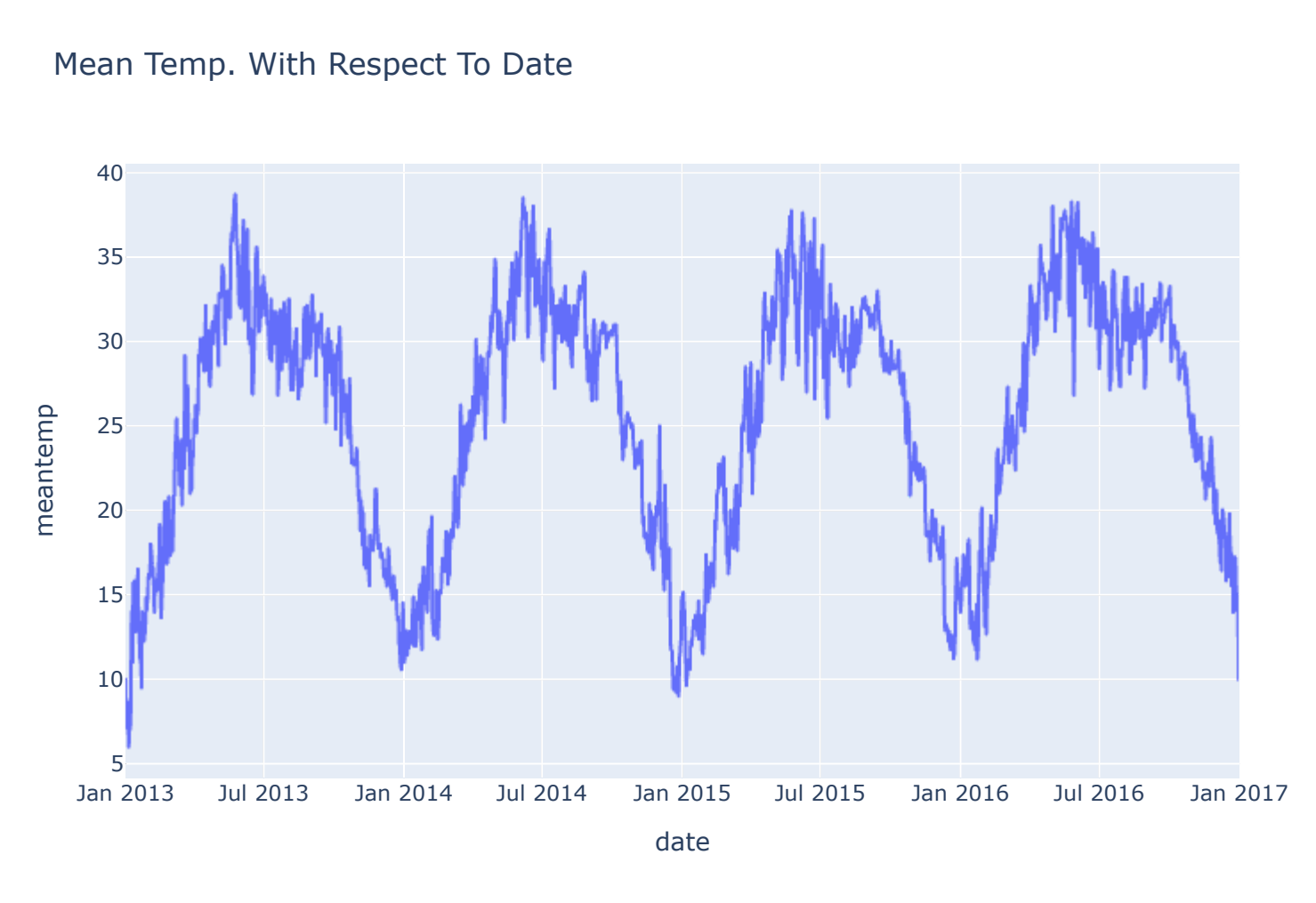
In Monthly Data, Data is of monthly Basis.

In Annual Data, Data is of Yearly Basis.

1. Line Plot

1. Daily Data With Respect to Mean Temprature by using plotly.express.

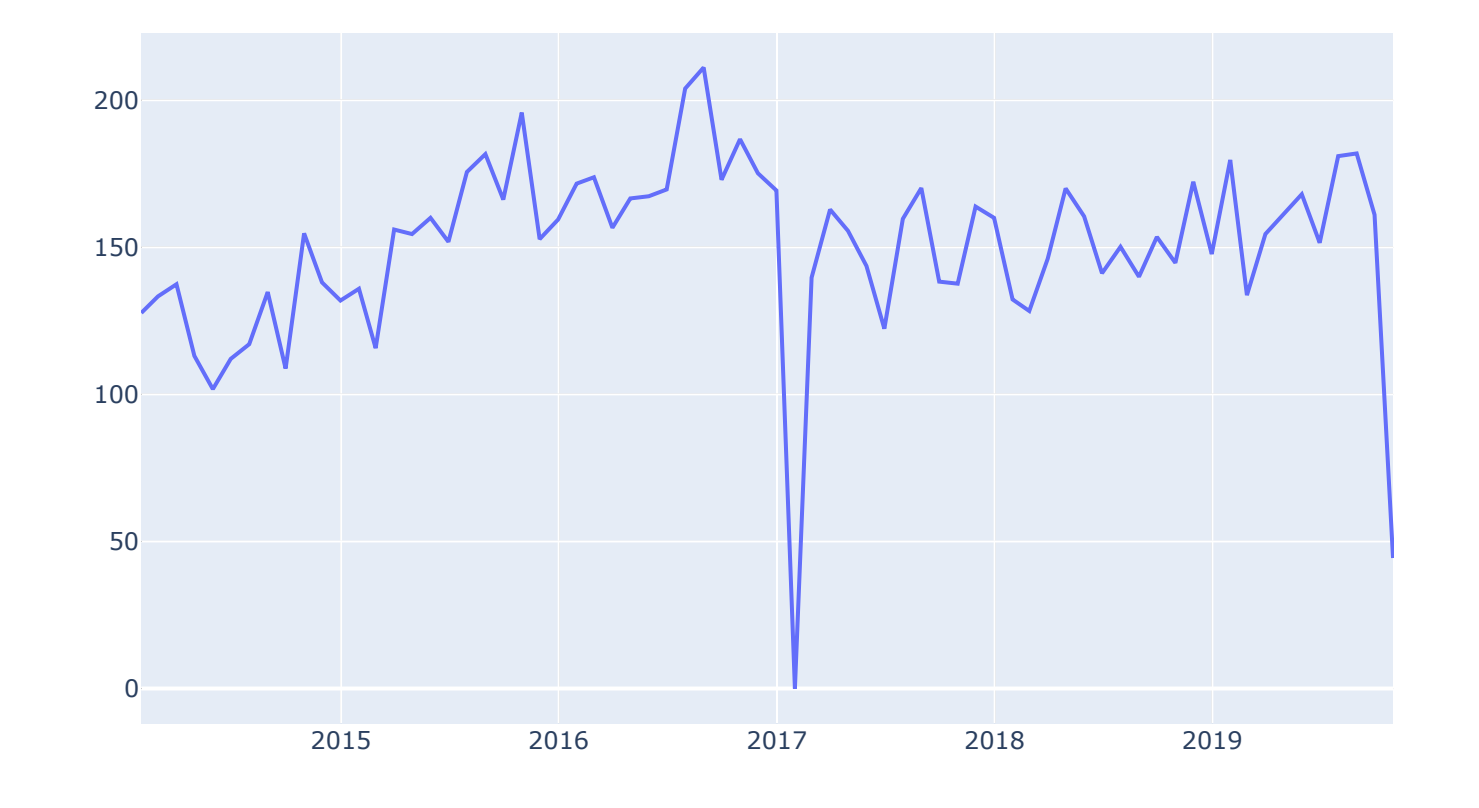
In [12]:
```
import plotly.express as px

fig = px.line(daily_data, x="date", y="meantemp", title='Mean Temp. With Respect To Date')
fig.show()
```



1. Monthty Data of gold prices with respect to months by plotly.graph_objs

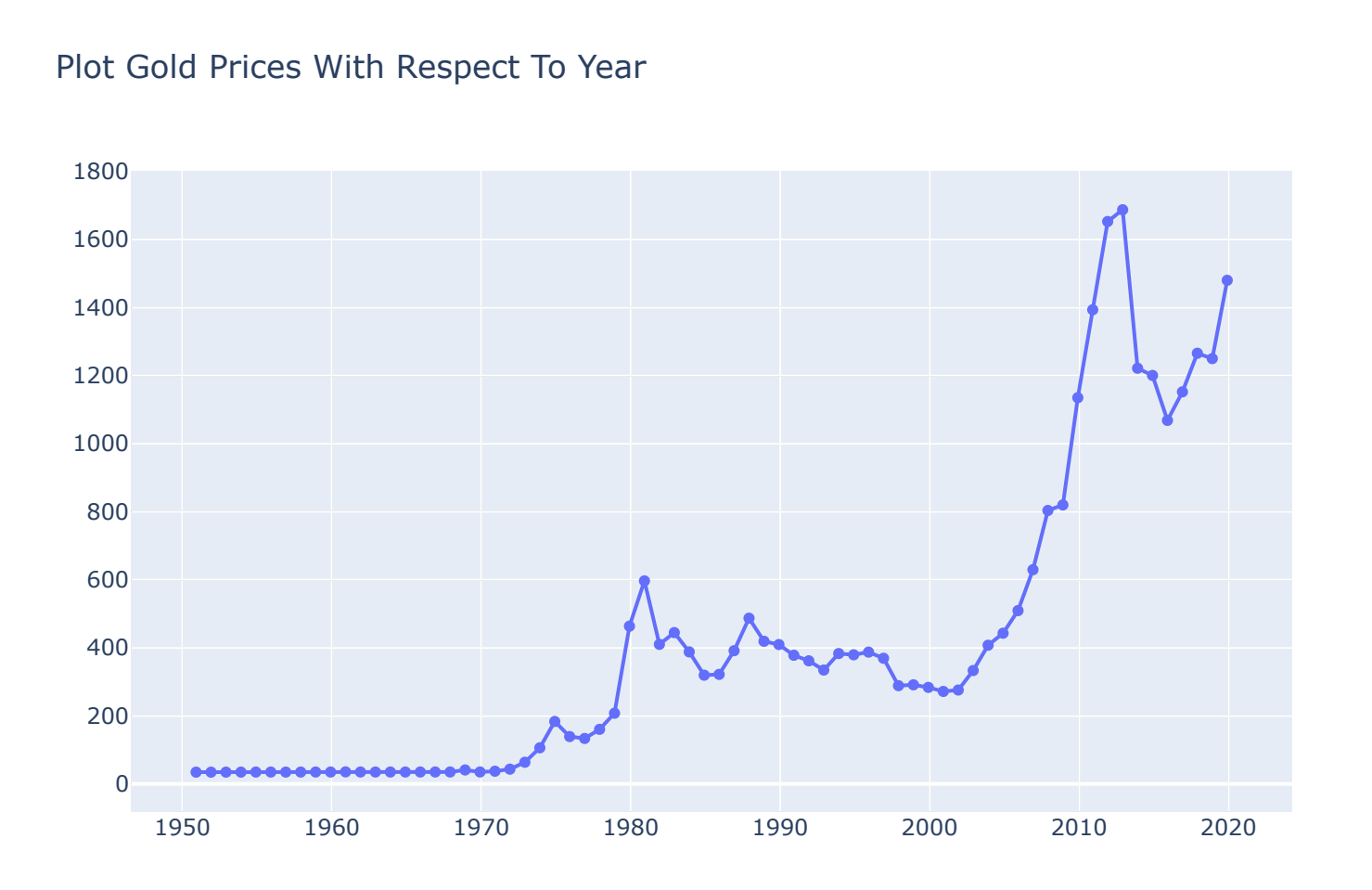
In Graph_objs we have to give a input as dictonary to plot the graph, then only we can plot a graph by plotly.graph_objs.

In [14]:
```
trace1 = go.Scatter(x= monthly_data.datum, y= monthly_data.M01AB)
fig = dict(data = trace1)
iplot(fig)
```



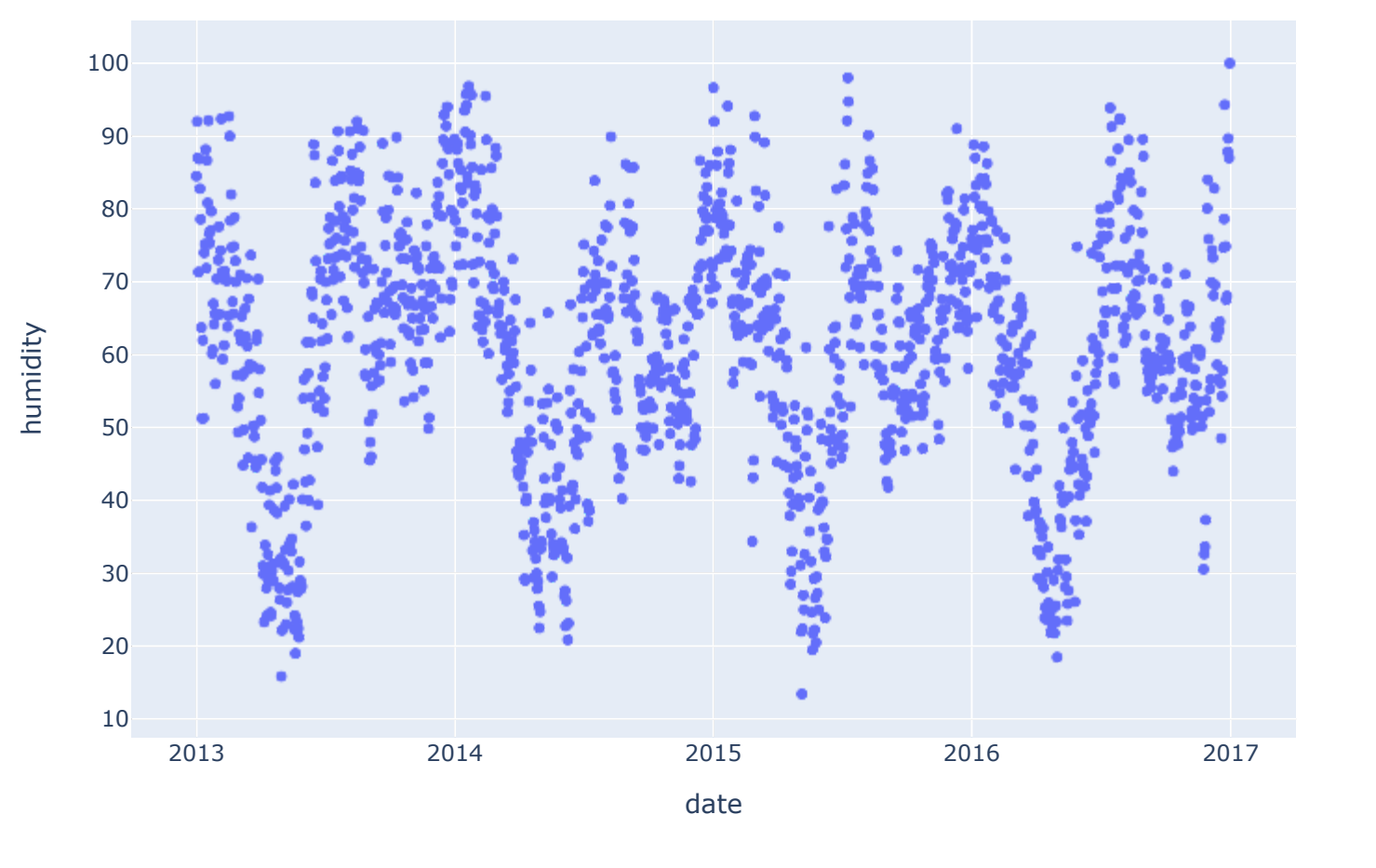
1. Annual Data of gold prices with respect to Year by plotly.graph_objs

In [15]:
```
trace1 = go.Scatter(x= annual_data.Date, y= annual_data.Price,mode = "lines+markers")
layout = dict(title = 'Plot Gold Prices With Respect To Year')
fig = dict(data = trace1, layout = layout)
iplot(fig)
```

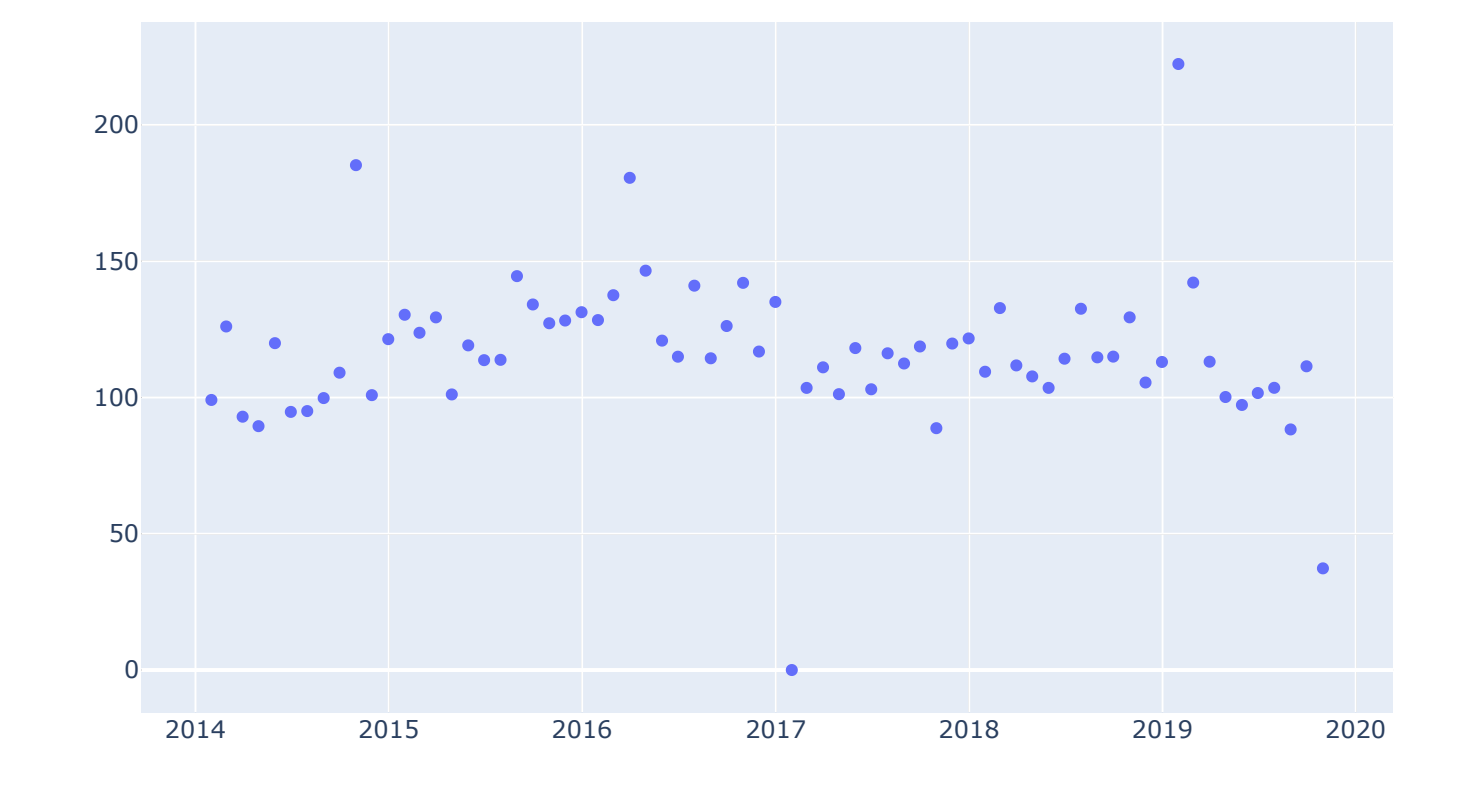


2. Scatter Plot

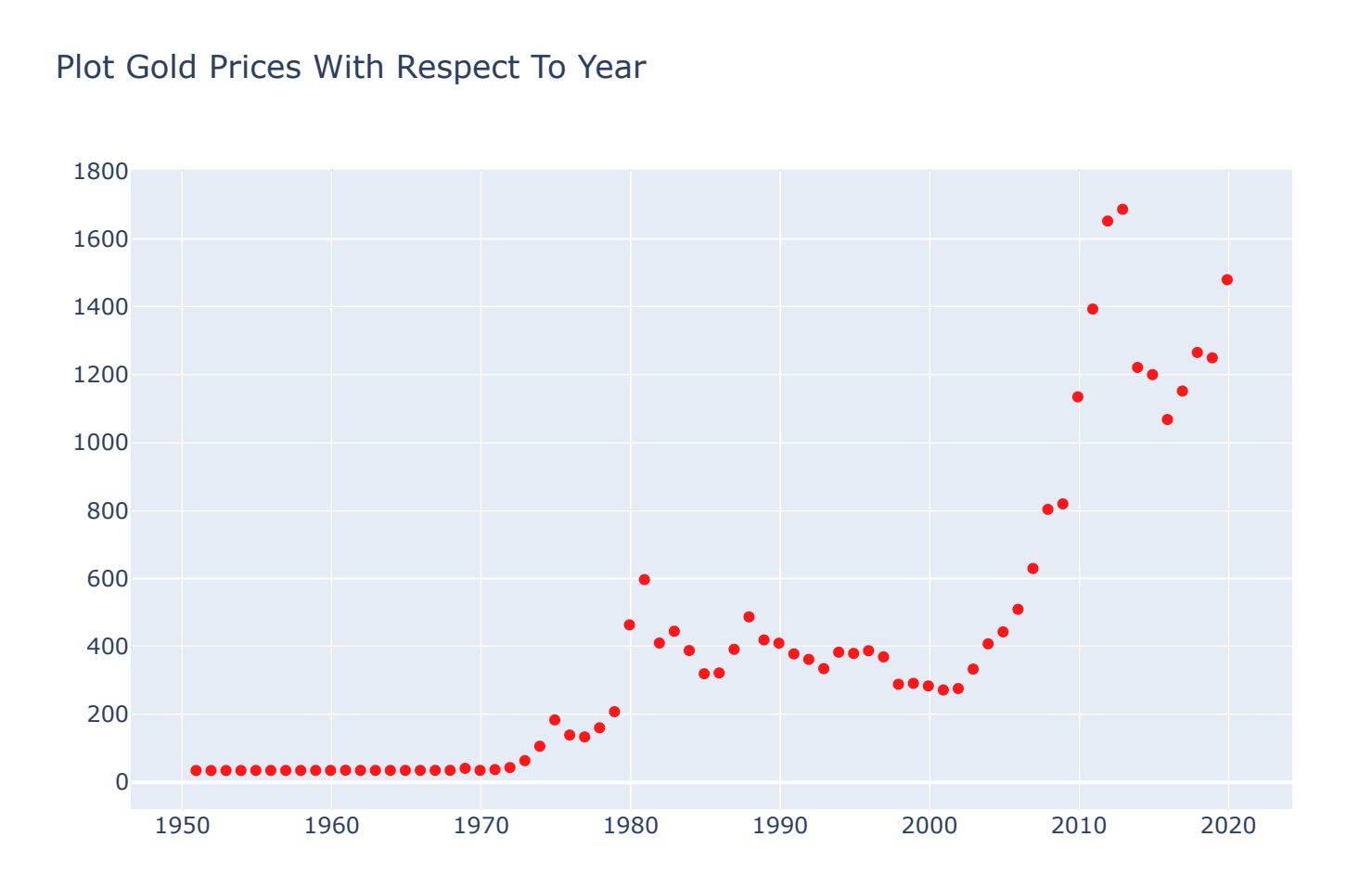
In [16]:
```
#scatter plot for daily data
fig = px.scatter(daily_data,x= "date", y= "humidity")
fig.show()
```



In [17]:
```
#scatter plot for monthly data
trace1 = go.Scatter(x= monthly_data.datum, y= monthly_data.M01AE, mode = "markers")
fig = dict(data = trace1)
iplot(fig)
```



In [18]:
```
#scatter plot for annual data
trace1 = go.Scatter(x= annual_data.Date, y= annual_data.Price,mode = "markers",
    marker_color='rgba(255, 00, 00, .9)')
layout = dict(title = 'Plot Gold Prices With Respect To Year')
fig = dict(data = trace1, layout = layout)
iplot(fig)
```



3. Bar Chart

Creating traces

x = x axis

y = y axis

mode = type of plot like marker, line or line + markers

name = name of the plots

marker = marker is used with dictionary.

color = color of lines. It takes RGB (red, green, blue) and opacity (alpha)

line = It is dictionary. line between bars

color = line color around bars

text = The hover text (hover is curser)

data = is a list that we add traces into it

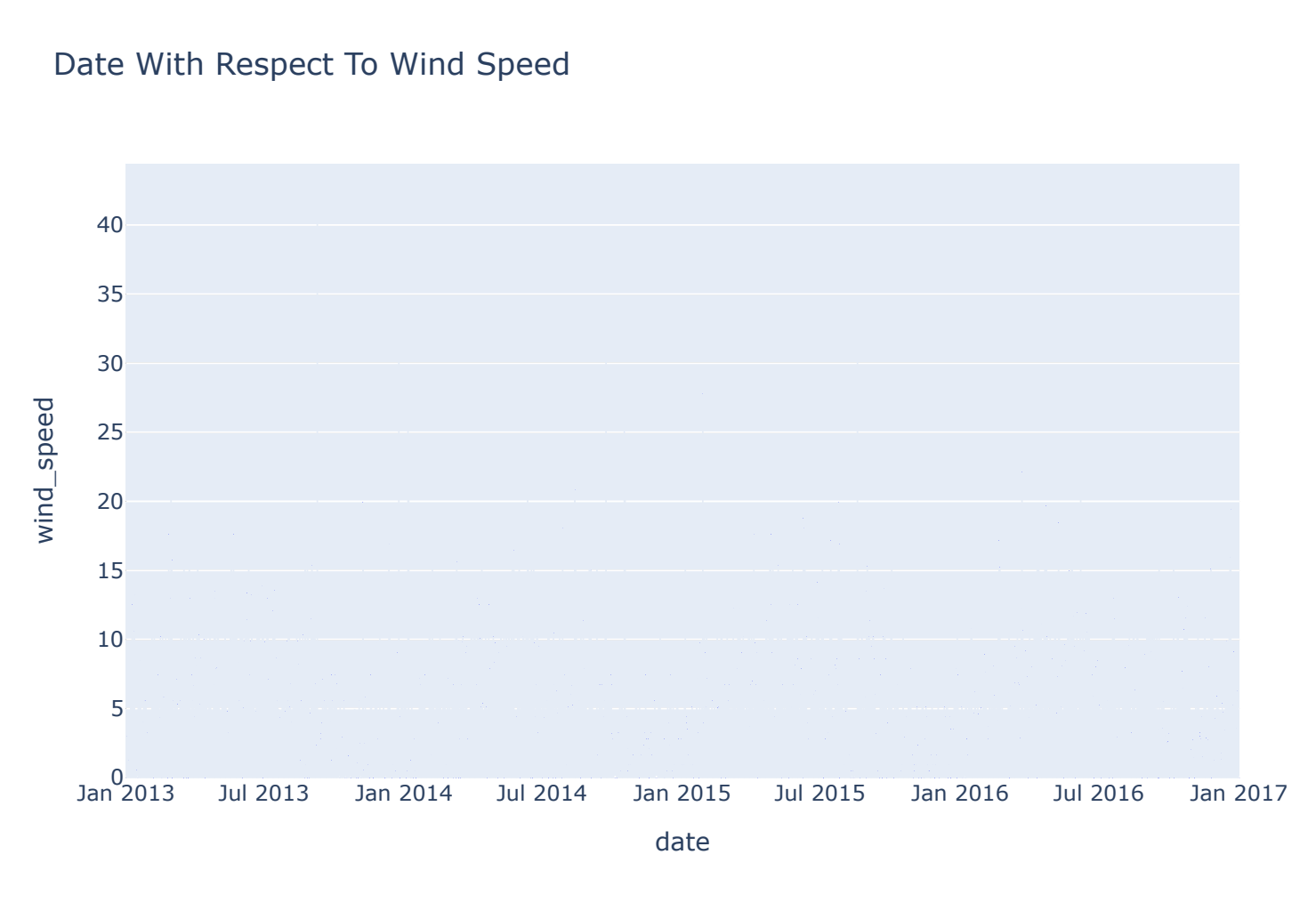
layout = it is dictionary.

barmode = bar mode of bars like grouped

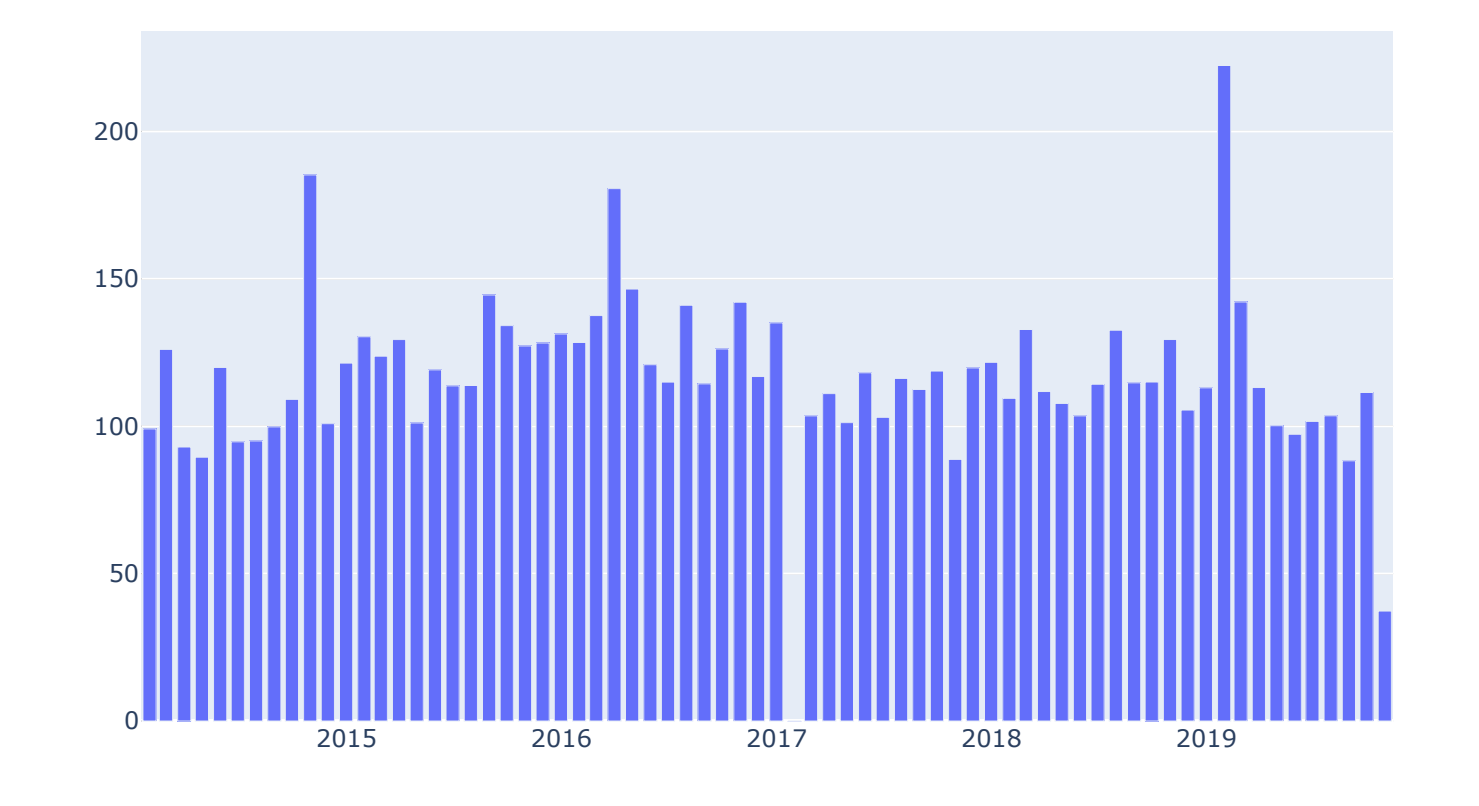
fig = it includes data and layout

iplot() = plots the figure(fig) that is created by data and layout

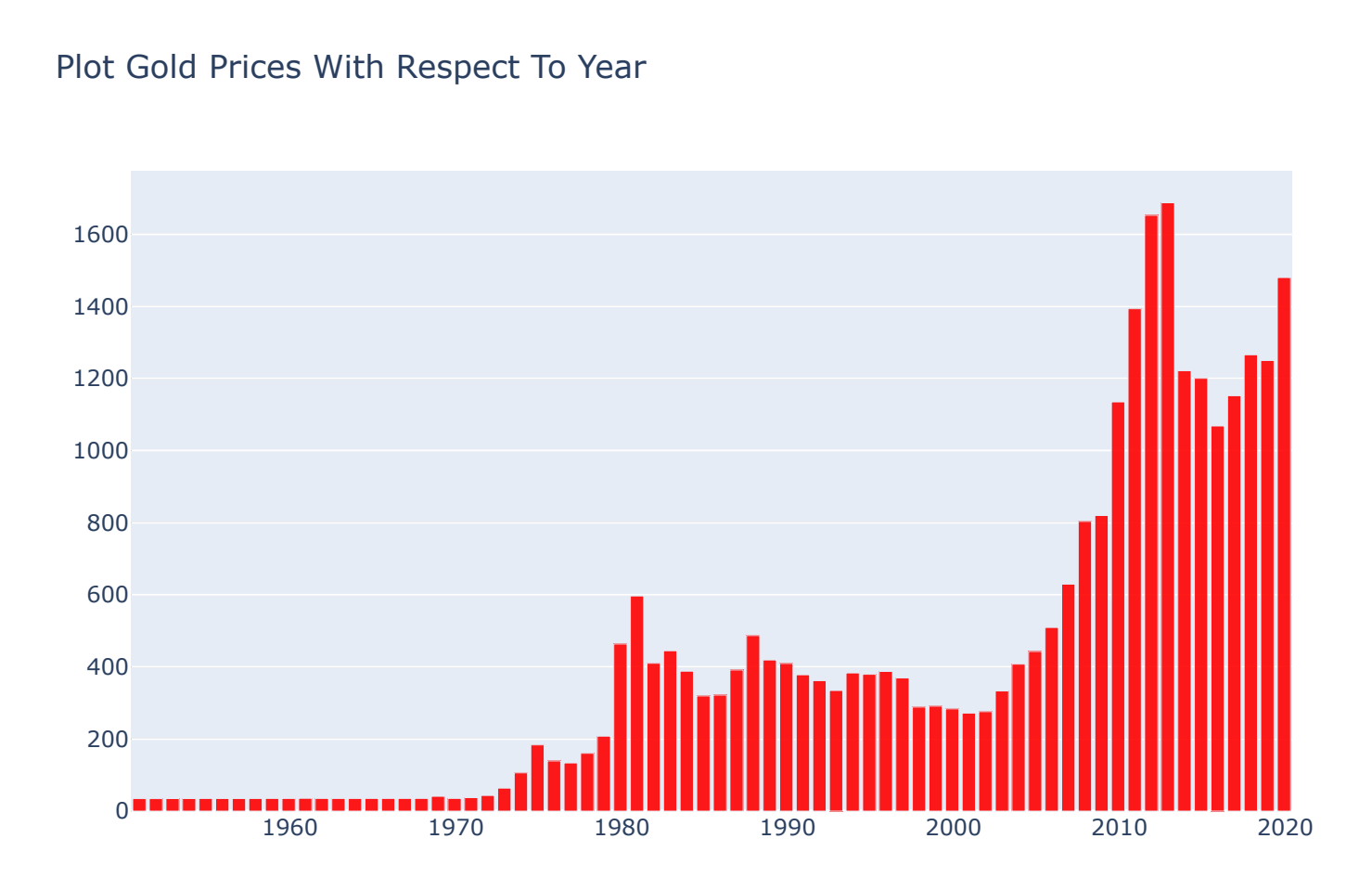
In [19]:
```
#bar chart for daily data
fig = px.bar(daily_data, x="date", y="wind_speed", title="Date With Respect To Wind Speed")
fig.show()
```



In [20]:
```
#scatter plot for monthly data
trace1 = go.Bar(x= monthly_data.datum, y= monthly_data.M01AE)
fig = dict(data = trace1)
iplot(fig)
```



In [21]:
```
#bar chart for annual data
trace1 = go.Bar(x= annual_data.Date, y= annual_data.Price,
    marker_color='rgba(255, 00, 00, .9)')
layout = dict(title = 'Plot Gold Prices With Respect To Year')
fig = dict(data = trace1, layout = layout)
iplot(fig)
```



4. Bubble Charts

To Add Dimesionality in visualization we can use Bubble charts.

x = x axis

y = y axis

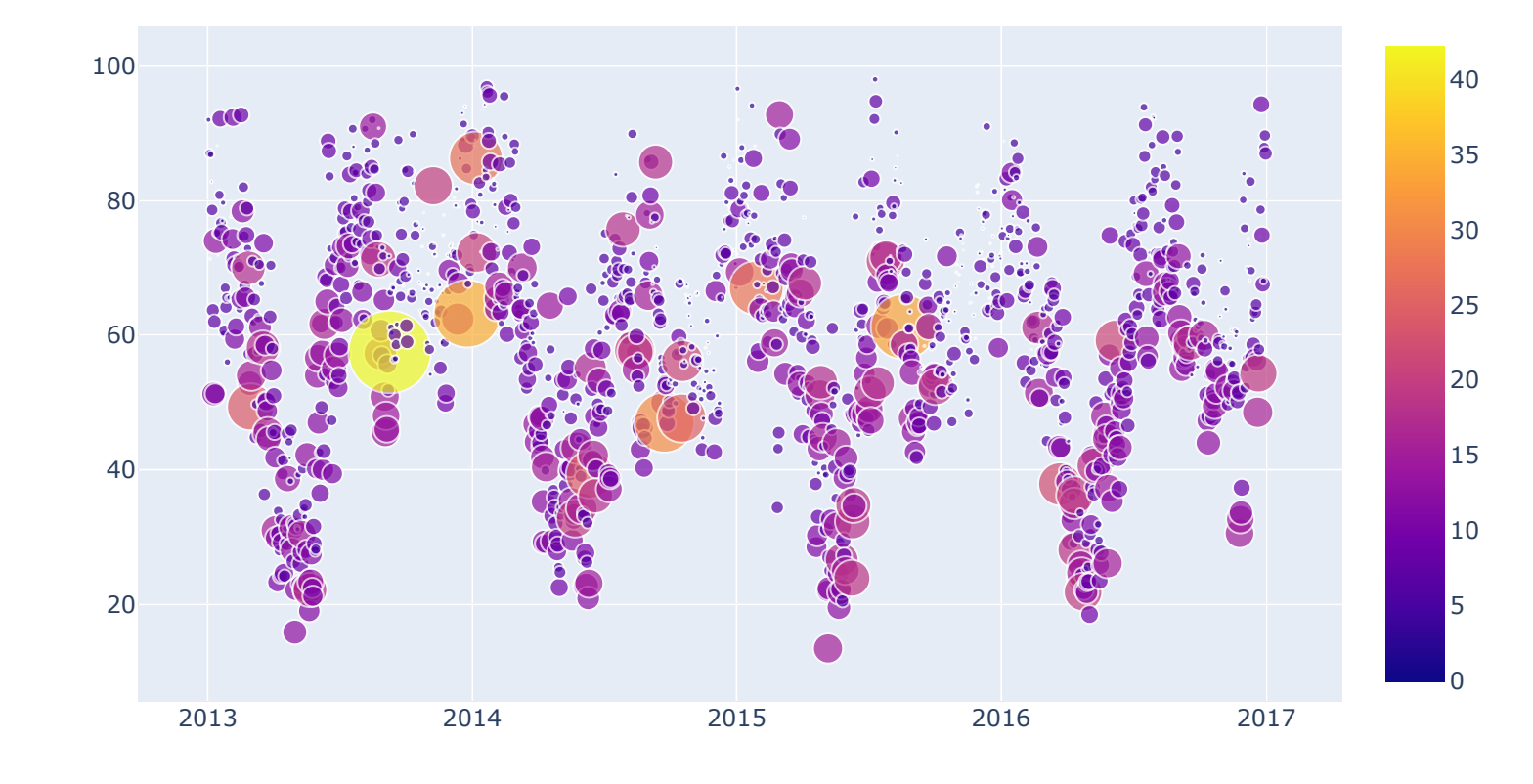
mode = markers(scatter)

marker = marker properties

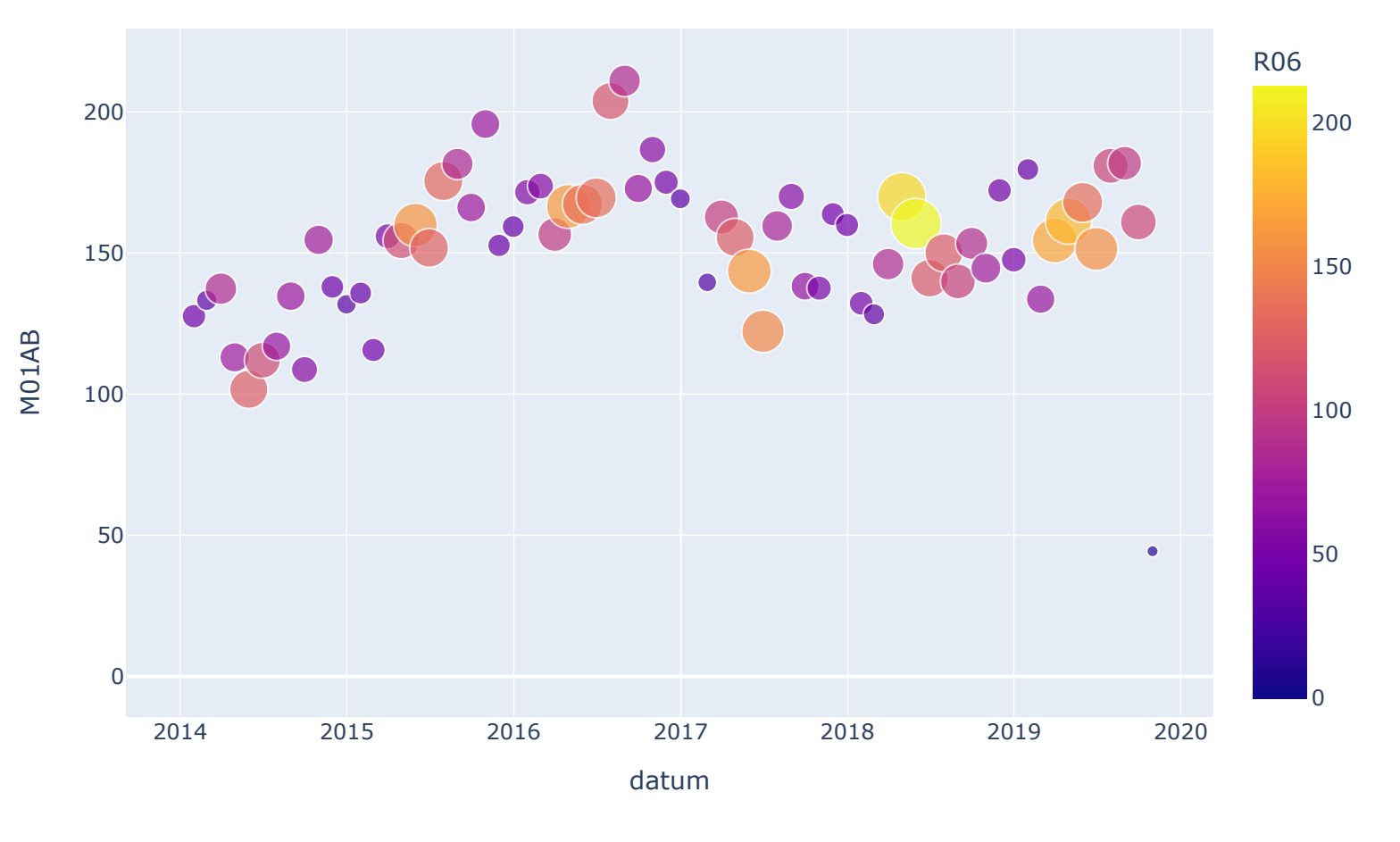
color = third dimension of plot

size = fourth dimension of plot

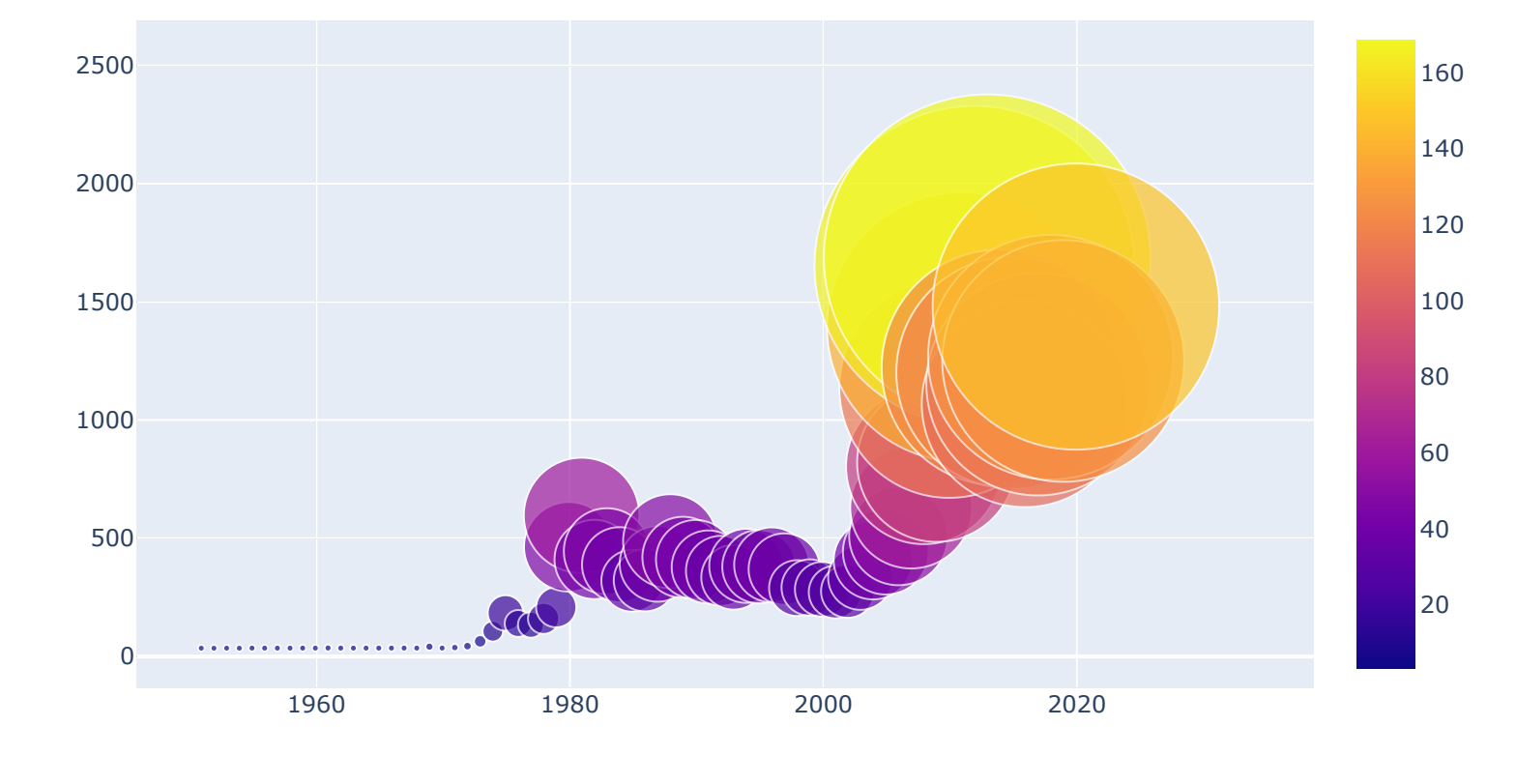
In [28]:
```
data = [
    {
        'x': daily_data.date,
        'y': daily_data.humidity,
        'mode': 'markers',
        'marker': {
            'color': daily_data.wind_speed,
            'size': daily_data.wind_speed,
            'showscale': True
        }

    }
]
iplot(data)
```



In [31]:
```
fig = px.scatter(monthly_data, x="datum", y="M01AB",size="R06", color="R06",
                 hover_name="datum")
fig.show()
```



In [37]:
```
a = annual_data.Price/10
data = [
    {
        'x': annual_data.Date,
        'y': annual_data.Price,
        'mode': 'markers',
        'marker': {
            'color': a,
            'size': a,
            'showscale': True
        }

    }
]
iplot(data)
```



5. Histogram

trace1 = first histogram

x = x axis

y = y axis

opacity = opacity of histogram

name = name of legend

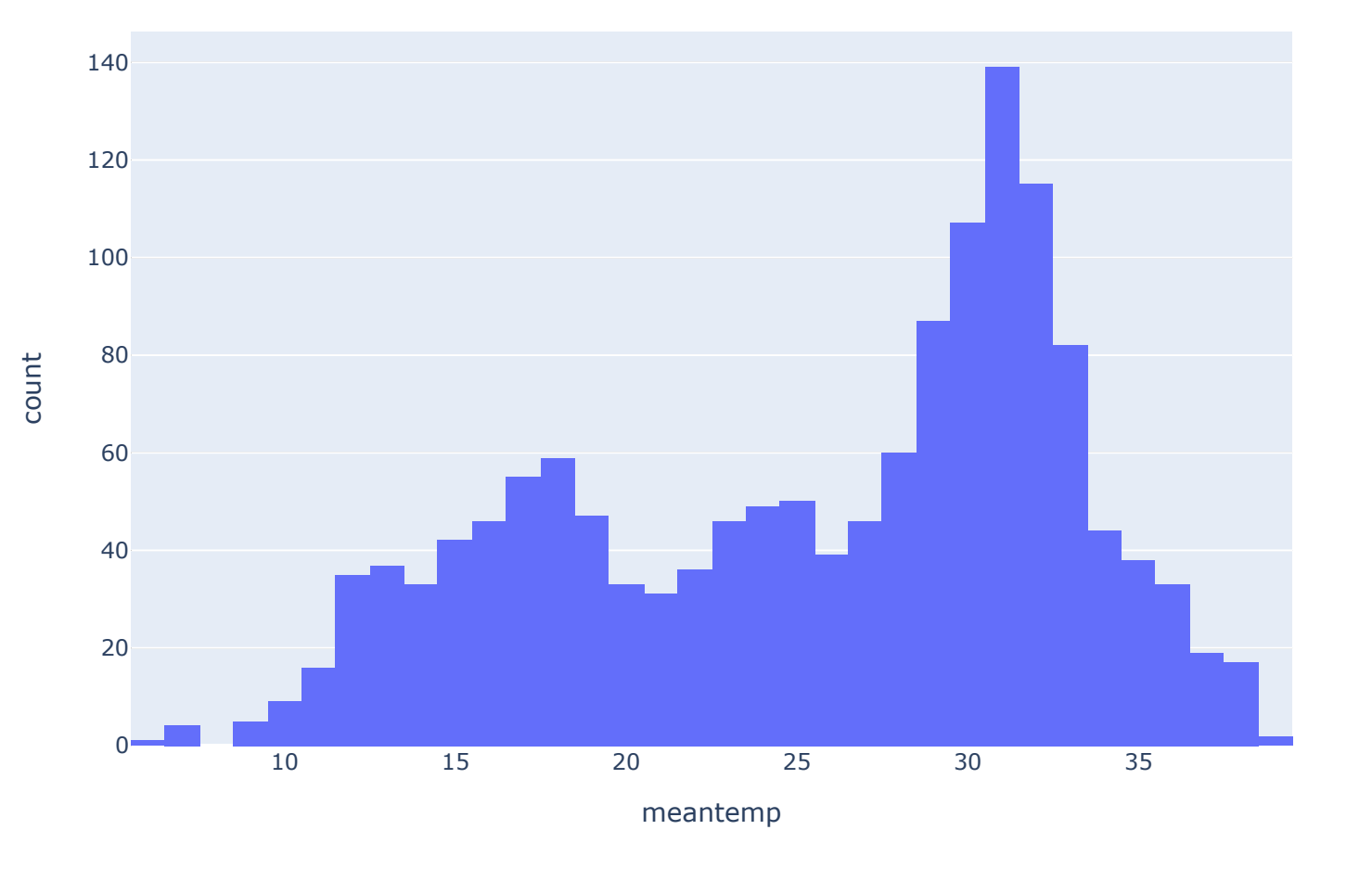
marker = color of histogram

trace2 = second histogram

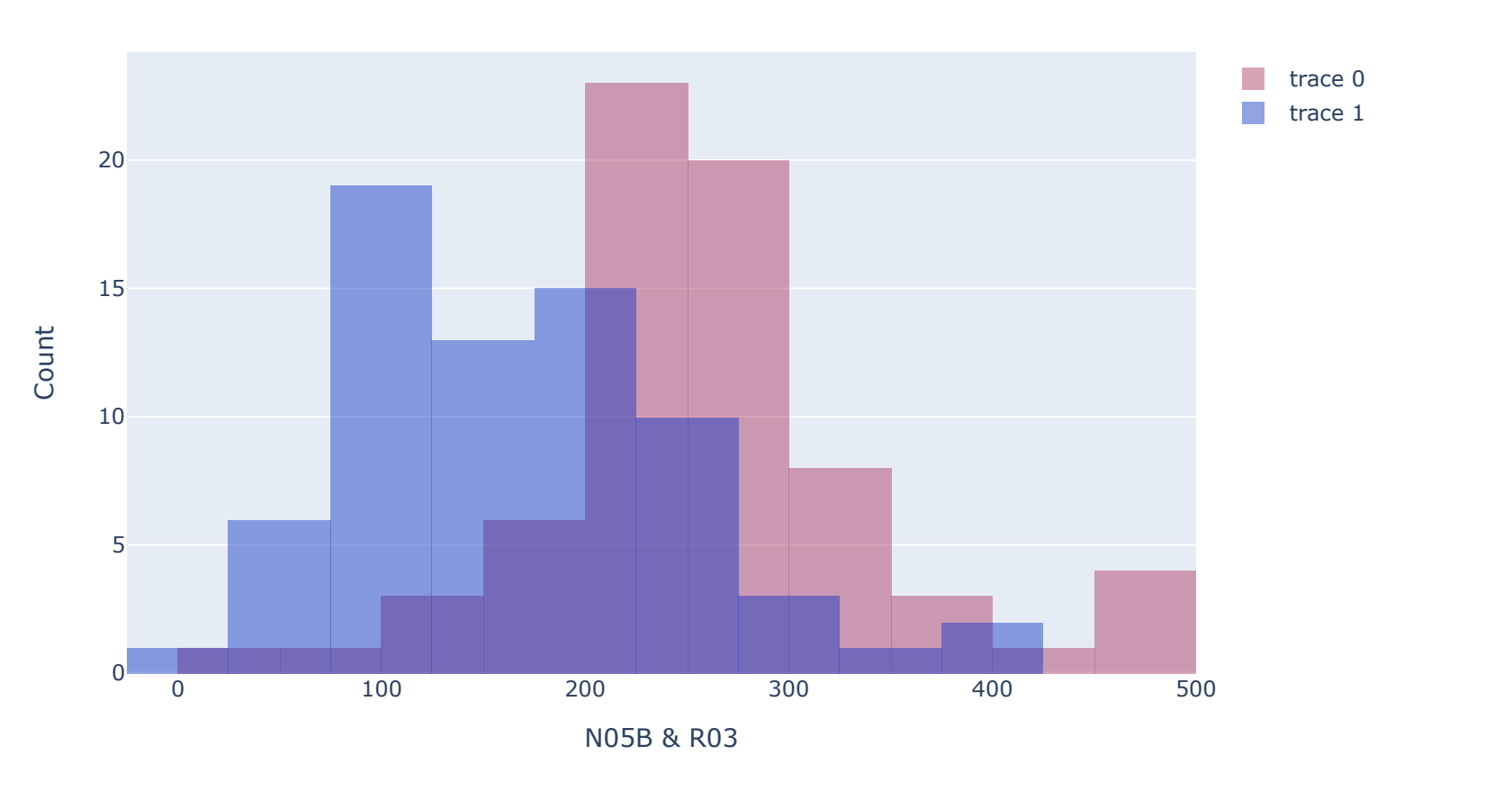
layout = layout

barmode = mode of histogram like overlay. Also you can change it with stack

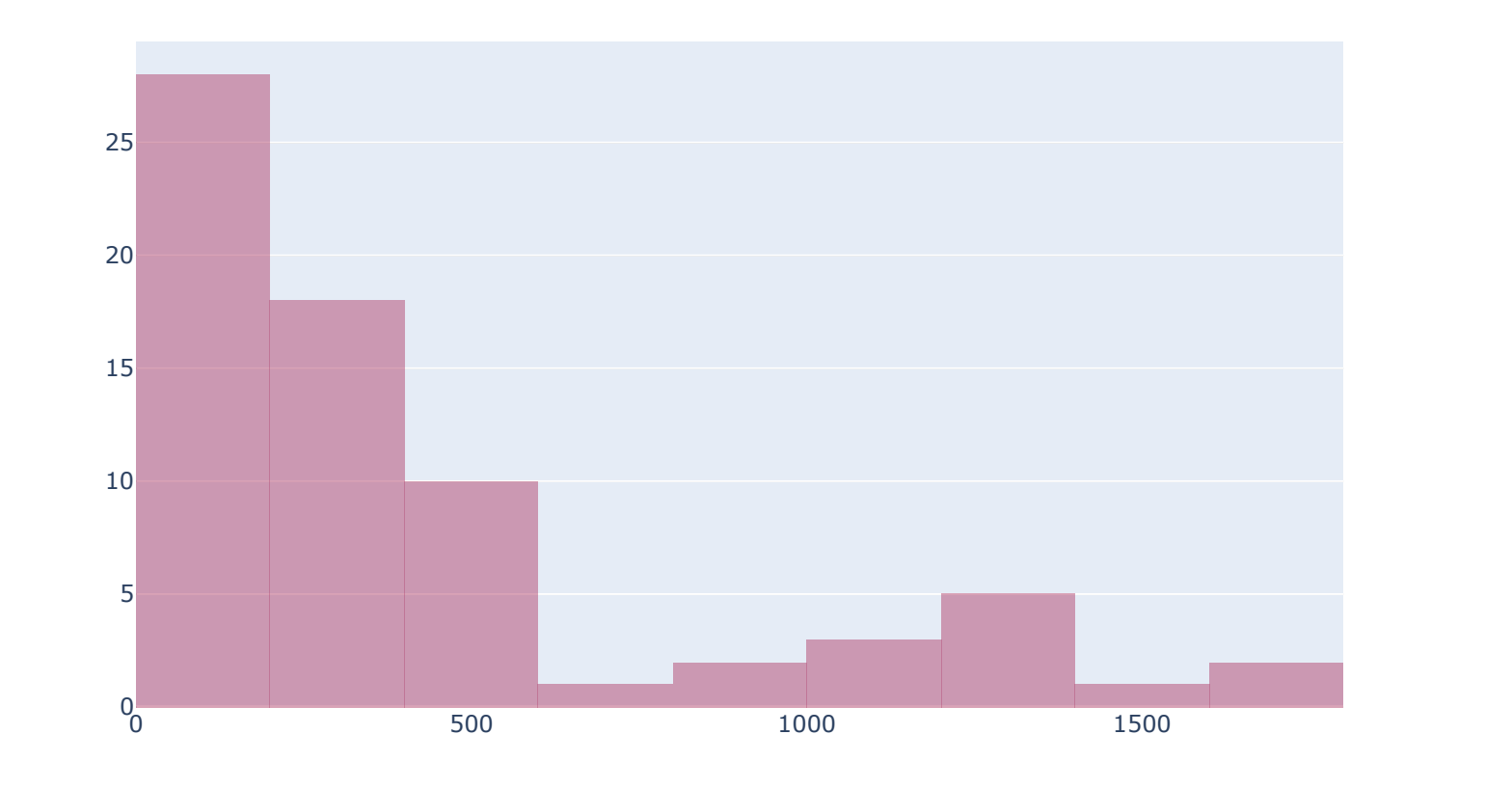
In [45]:
```
fig = px.histogram(daily_data, x="meantemp")
fig.show()
```



```
In [55]: trace1 = go.Histogram(
             x= monthly_data.N05B,
             opacity=0.75,
             marker=dict(color='rgba(171, 50, 96, 0.6)'))

         trace2 = go.Histogram(
             x= monthly_data.R03,
             opacity=0.75,
             marker=dict(color='rgba(12, 50, 196, 0.6)'))

         layout = go.Layout(barmode='overlay',
                            xaxis=dict(title='N05B & R03'),
                            yaxis=dict( title='Count'),
         )

         data = [trace1,trace2]

         fig = dict(data = data, layout = layout)
         iplot(fig)
```




```
In [51]: trace1 = go.Histogram(
             x= annual_data.Price,
             opacity=0.75,
             marker=dict(color='rgba(171, 50, 96, 0.6)'))

         fig = dict(data = trace1)
         iplot(fig)
```



6. Box Plots

Box Plots

Median (50th percentile) = middle value of the data set. Sort and take the data in the middle. It is also called 50% percentile that is 50% of data are less that median(50th quartile)(quartile)

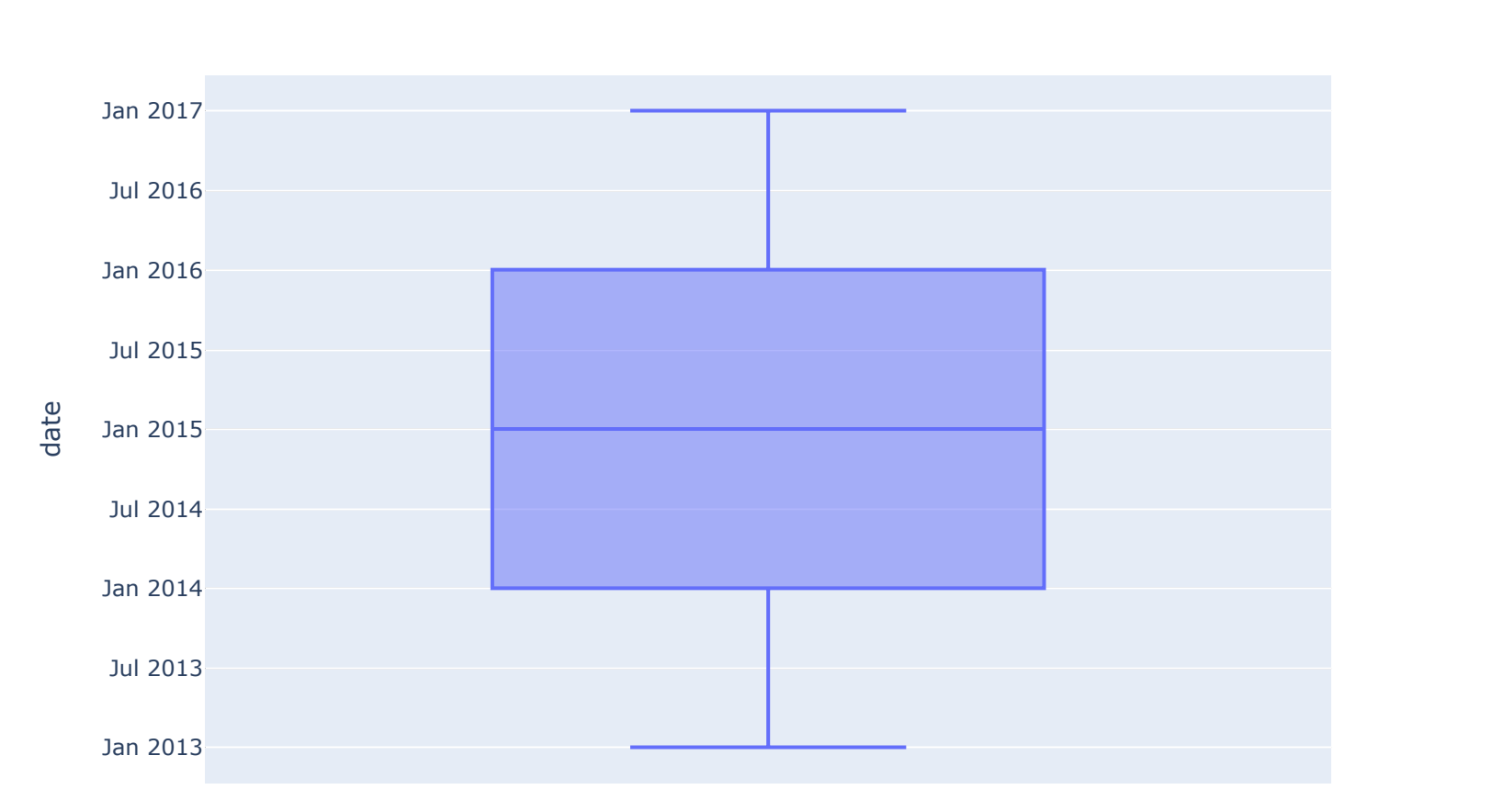
1. 25th percentile = quartile 1 (Q1) that is lower quartile
2. 75th percentile = quartile 3 (Q3) that is higher quartile
3. height of box = IQR = interquartile range = Q3-Q1
4. Whiskers = 1.5 * IQR from the Q1 and Q3
5. Outliers = being more than 1.5*IQR away from median commonly.

trace = box

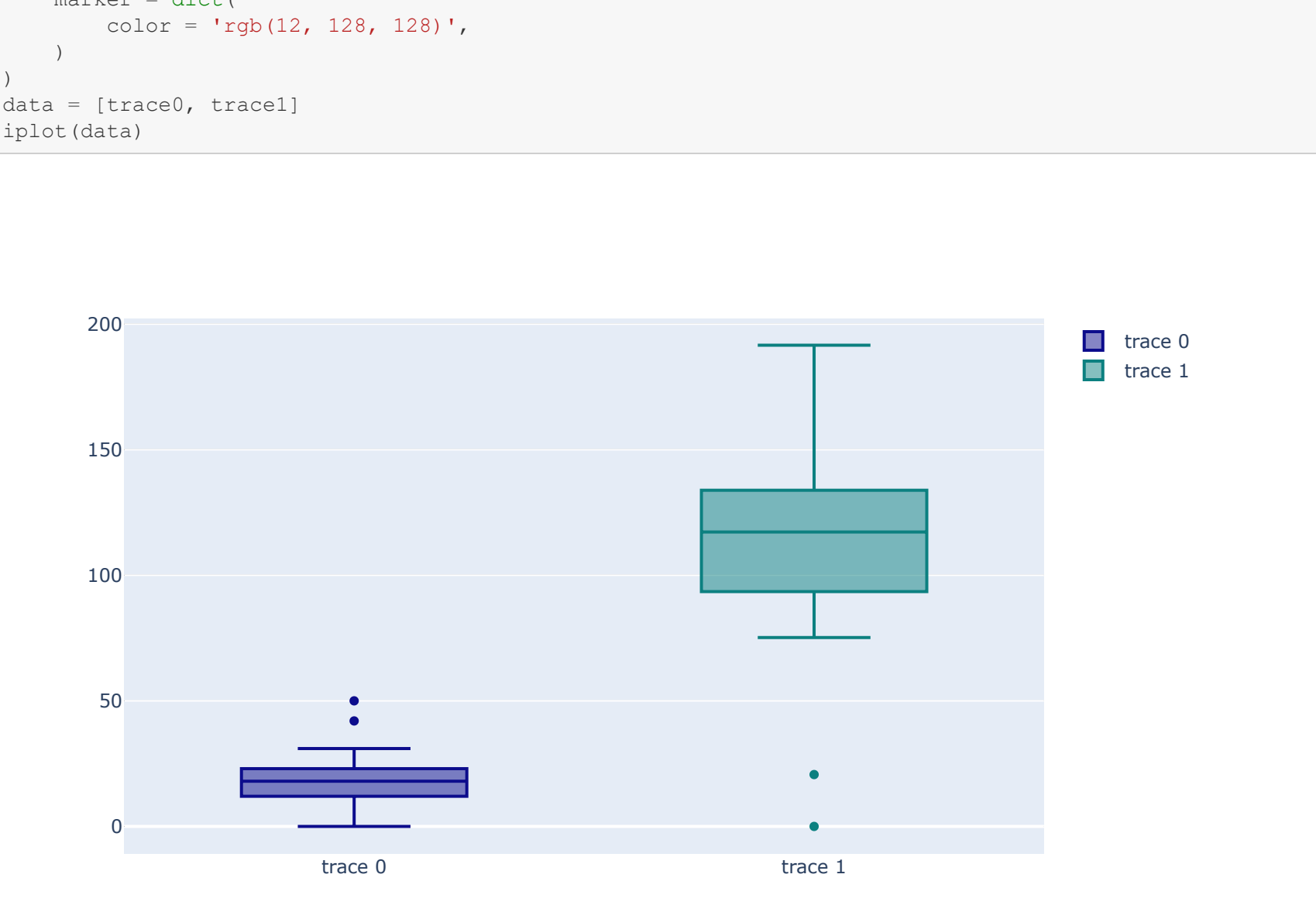
y = data we want to visualize with box plot

marker = color

```
In [56]: fig = px.box(daily_data, y="date")
         fig.show()
```




```
In [57]: trace0 = go.Box(
             y=monthly_data.N05C,
             marker = dict(
                 color = 'rgb(12, 12, 140)',
             )
         )
         trace1 = go.Box(
             y=monthly_data.N02BA,
             marker = dict(
                 color = 'rgb(12, 128, 128)',
             )
         )
         data = [trace0, trace1]
         iplot(data)
```




```
In [63]: fig = go.Figure()

         fig.add_trace(go.Box(
             y=annual_data.Price,
             marker = dict(
                 color = 'rgb(12, 12, 140)',
             )))

         fig.show()
```