Polar Charts in Python

How to make polar charts in Python with Plotly.

Polar chart with Plotly Express

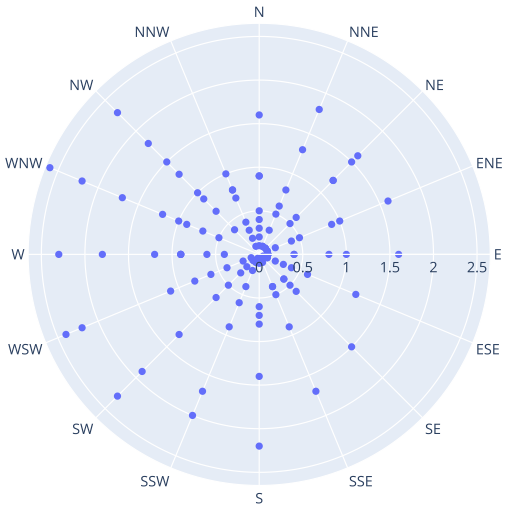
A polar chart represents data along radial and angular axes. With Plotly Express, it is possible to represent polar data as scatter markers with px.scatter_polar, and as lines with px.line_polar.

Plotly Express (/python/plotly-express/) is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data (/python/px-arguments/) and produces easy-to-style figures (/python/styling-plotly-express/).

For other types of arguments, see the section below using go.Scatterpolar.

The radial and angular coordinates are given with the r and theta arguments of px.scatter_polar. In the example below the theta data are categorical, but numerical data are possible too and the most common case.

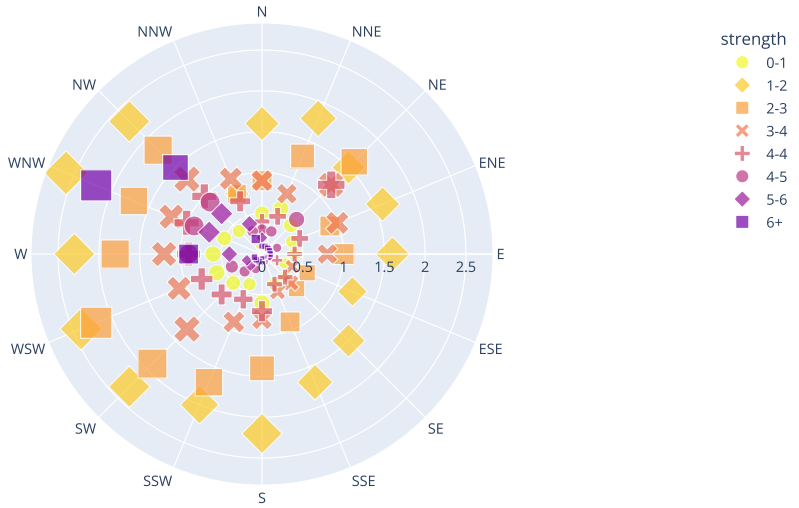
```
import plotly.express as px
df = px.data.wind()
fig = px.scatter_polar(df, r="frequency", theta="direction")
fig.show()
```



The "strength" column corresponds to strength categories of the wind, and there is a frequency value for each direction and strength. Below we use the strength column to encode the color, symbol and size of the markers.

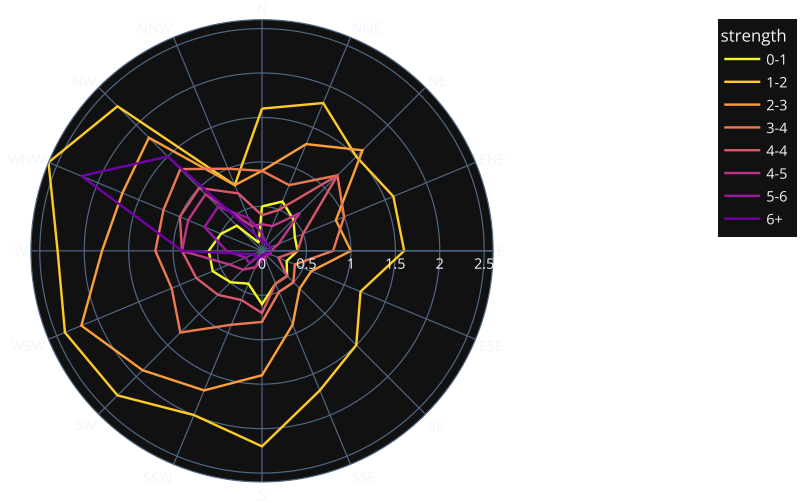
```
import plotly.express as px
df = px.data.wind()
fig = px.scatter_polar(df, r="frequency", theta="direction",
                       color="strength", symbol="strength", size="frequency",
                       color_discrete_sequence=px.colors.sequential.Plasma_r)
fig.show()
```

lar



For a line polar plot, use `px.line_polar`:

```
import plotly.express as px
df = px.data.wind()
fig = px.line_polar(df, r="frequency", theta="direction", color="strength", line_close=True,
                    color_discrete_sequence=px.colors.sequential.Plasma_r,
                    template="plotly_dark",)
fig.show()
```

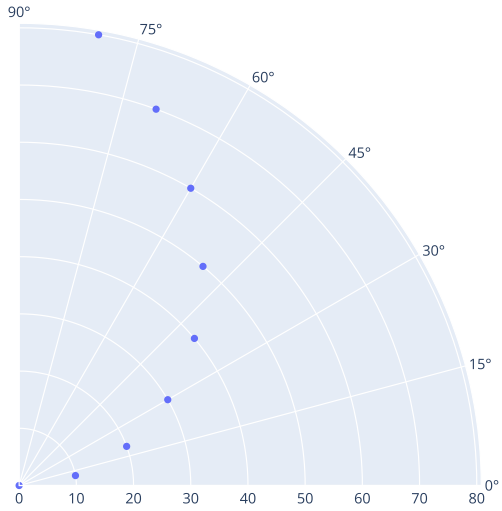


age (https://plotly.com/python/wind-rose-charts/) for more wind rose visualizations in polar coordinates.

You can plot less than a whole circle with the `range_theta` argument, and also control the `start_angle` and `direction`:

```
import plotly.express as px
fig = px.scatter_polar(r=range(0,90,10), theta=range(0,90,10),
                       range_theta=[0,90], start_angle=0, direction="counterclockwise")
fig.show()
```

lar

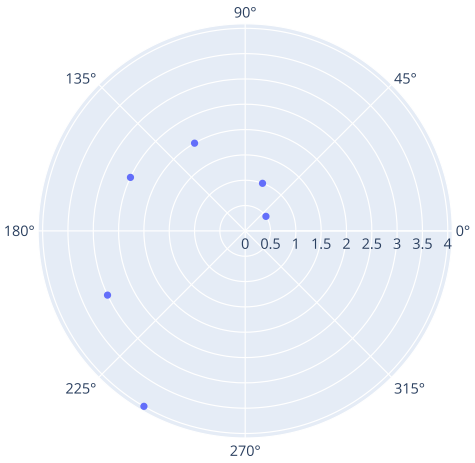


Polar Scatter Plot with go.Scatterpolar

If Plotly Express does not provide a good starting point, you can use the more generic go.Scatterpolar class from plotly.graph_objects (/python/graph-objects/). All the options are documented in the reference page (https://plotly.com/python/reference/scatterpolar/).

Basic Polar Chart

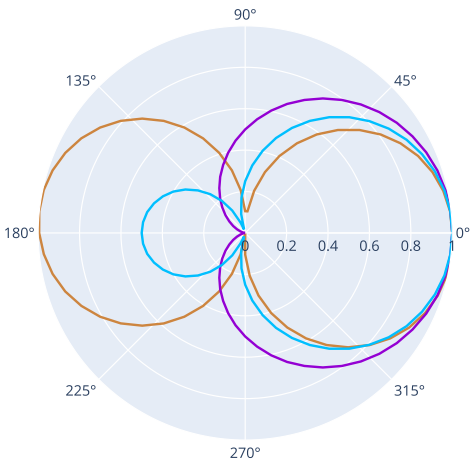
```
import plotly.graph_objects as go

fig = go.Figure(data=
    go.Scatterpolar(
        r = [0.5,1,2,2.5,3,4],
        theta = [35,70,120,155,205,240],
        mode = 'markers',
    ))

fig.update_layout(showlegend=False)
fig.show()
```



Line Polar Chart

lar

```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/polar_dataset.csv")

fig = go.Figure()
fig.add_trace(go.Scatterpolar(
        r = df['x1'],
        theta = df['y'],
        mode = 'lines',
        name = 'Figure 8',
        line_color = 'peru'
    ))
fig.add_trace(go.Scatterpolar(
        r = df['x2'],
        theta = df['y'],
        mode = 'lines',
        name = 'Cardioid',
        line_color = 'darkviolet'
    ))
fig.add_trace(go.Scatterpolar(
        r = df['x3'],
        theta = df['y'],
        mode = 'lines',
        name = 'Hypercardioid',
        line_color = 'deepskyblue'
    ))


fig.update_layout(
    title = 'Mic Patterns',
    showlegend = False
)

fig.show()
```

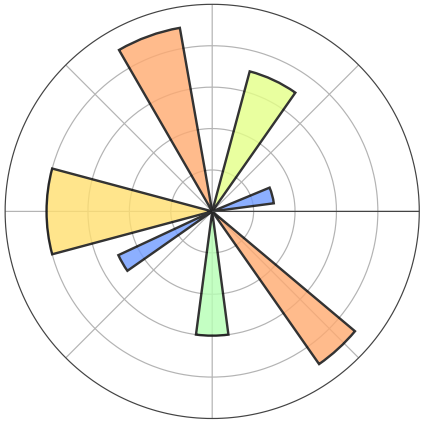


Polar Bar Chart

a.k.a matplotlib logo in a few lines of code

lar

```
import plotly.graph_objects as go

fig = go.Figure(go.Barpolar(
    r=[3.5, 1.5, 2.5, 4.5, 4.5, 4, 3],
    theta=[65, 15, 210, 110, 312.5, 180, 270],
    width=[20,15,10,20,15,30,15,],
    marker_color=["#E4FF87", '#709BFF', '#709BFF', '#FFAA70', '#FFAA70', '#FFDF70', '#B6FFB4'],
    marker_line_color="black",
    marker_line_width=2,
    opacity=0.8
))

fig.update_layout(
    template=None,
    polar = dict(
        radialaxis = dict(range=[0, 5], showticklabels=False, ticks=''),
        angularaxis = dict(showticklabels=False, ticks='')
    )
)

fig.show()
```

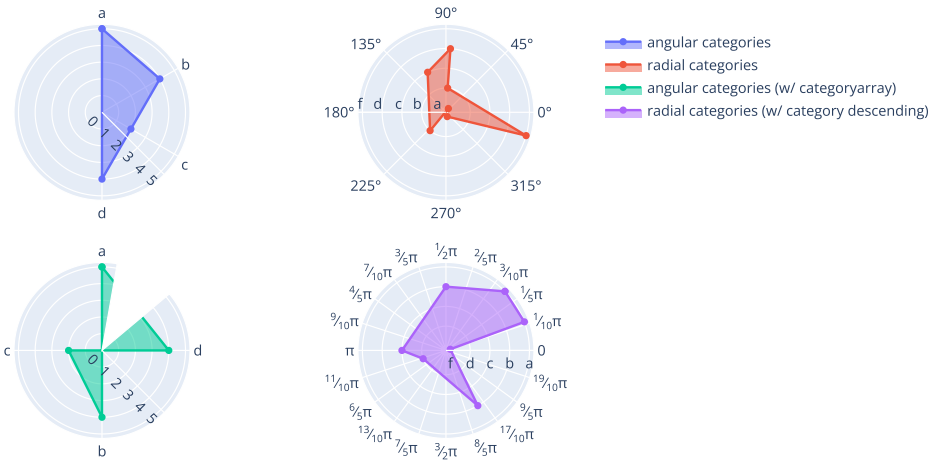


Categorical Polar Chart

lar

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=2, cols=2, specs=[[{'type': 'polar'}]*2]*2)

fig.add_trace(go.Scatterpolar(
      name = "angular categories",
      r = [5, 4, 2, 4, 5],
      theta = ["a", "b", "c", "d", "a"],
    ), 1, 1)
fig.add_trace(go.Scatterpolar(
      name = "radial categories",
      r = ["a", "b", "c", "d", "b", "f", "a"],
      theta = [1, 4, 2, 1.5, 1.5, 6, 5],
      thetaunit = "radians",
    ), 1, 2)
fig.add_trace(go.Scatterpolar(
      name = "angular categories (w/ categoryarray)",
      r = [5, 4, 2, 4, 5],
      theta = ["a", "b", "c", "d", "a"],
    ), 2, 1)
fig.add_trace(go.Scatterpolar(
      name = "radial categories (w/ category descending)",
      r = ["a", "b", "c", "d", "b", "f", "a", "a"],
      theta = [45, 90, 180, 200, 300, 15, 20, 45],
    ), 2, 2)

fig.update_traces(fill='toself')
fig.update_layout(
    polar = dict(
      radialaxis_angle = -45,
      angularaxis = dict(
        direction = "clockwise",
        period = 6)
    ),
    polar2 = dict(
      radialaxis = dict(
        angle = 180,
        tickangle = -180 # so that tick labels are not upside down
      )
    ),
    polar3 = dict(
      sector = [80, 400],
      radialaxis_angle = -45,
      angularaxis_categoryarray = ["d", "a", "c", "b"]
    ),
    polar4 = dict(
      radialaxis_categoryorder = "category descending",
      angularaxis = dict(
        thetaunit = "radians",
        dtick = 0.3141592653589793
      ))
)

fig.show()
```

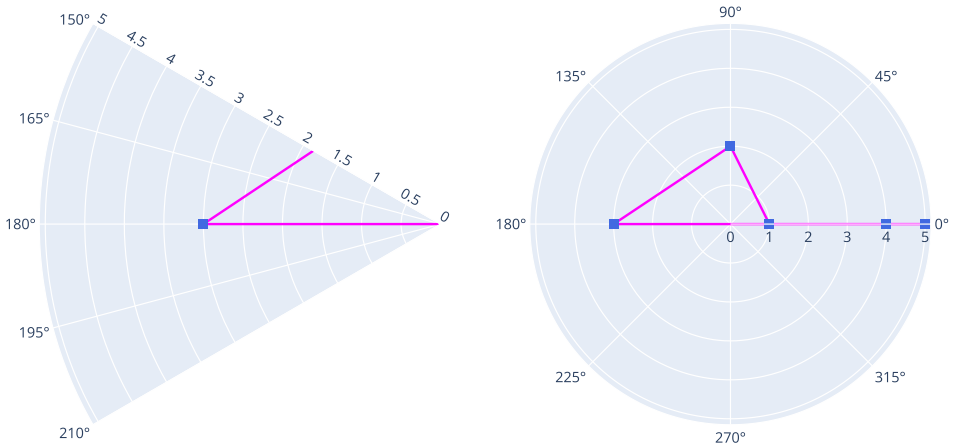
lar



Polar Chart Sector

lar

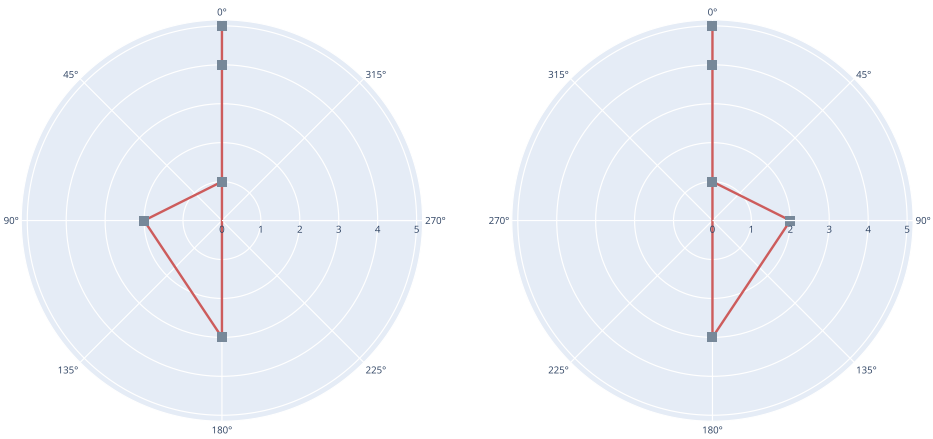
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'polar'}]*2])

fig.add_trace(go.Scatterpolar(), 1, 1)
fig.add_trace(go.Scatterpolar(), 1, 2)

# Same data for the two Scatterpolar plots, we will only change the sector in the layout
fig.update_traces(mode = "lines+markers",
      r = [1,2,3,4,5],
      theta = [0,90,180,360,0],
      line_color = "magenta",
      marker = dict(
        color = "royalblue",
        symbol = "square",
        size = 8
      ))

# The sector is [0, 360] by default, we update it for the first plot only
fig.update_layout(
    showlegend = False,
    polar = dict(# setting parameters for the second plot would be polar2=dict(...)
      sector = [150,210],
    ))


fig.show()
```



Polar Chart Directions

lar

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'polar'},    {'type': 'polar'}]])

r = [1,2,3,4,5]
theta = [0,90,180,360,0]

fig.add_trace(go.Scatterpolar(), 1, 1)
fig.add_trace(go.Scatterpolar(), 1, 2)

# Same data for the two Scatterpolar plots, we will only change the direction in the layout
fig.update_traces(r= r, theta=theta,
                  mode="lines+markers", line_color='indianred',
                  marker=dict(color='lightslategray', size=8, symbol='square'))
fig.update_layout(
    showlegend = False,
    polar = dict(
      radialaxis_tickfont_size = 8,
      angularaxis = dict(
        tickfont_size=8,
        rotation=90, # start position of angular axis
        direction="counterclockwise"
      )
    ),
    polar2 = dict(
      radialaxis_tickfont_size = 8,
      angularaxis = dict(
        tickfont_size = 8,
        rotation = 90,
        direction = "clockwise"
      ),
    ))

fig.show()
```



Webgl Polar Chart

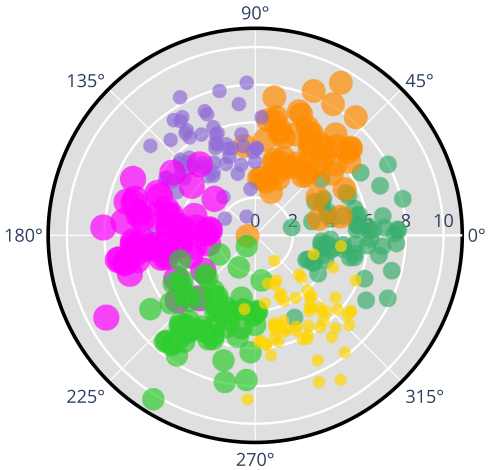
The go.Scatterpolargl trace uses the WebGL (https://en.wikipedia.org/wiki/WebGL) plotting engine for GPU-accelerated rendering.

lar

```
import plotly.graph_objects as go
import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/hobbs-pearson-trials.csv")

fig = go.Figure()

fig.add_trace(go.Scatterpolargl(
      r = df.trial_1_r,
      theta = df.trial_1_theta,
      name = "Trial 1",
      marker=dict(size=15, color="mediumseagreen")
    ))
fig.add_trace(go.Scatterpolargl(
      r = df.trial_2_r,
      theta = df.trial_2_theta,
      name = "Trial 2",
      marker=dict(size=20, color="darkorange")
    ))
fig.add_trace(go.Scatterpolargl(
      r = df.trial_3_r,
      theta = df.trial_3_theta,
      name = "Trial 3",
      marker=dict(size=12, color="mediumpurple")
    ))
fig.add_trace(go.Scatterpolargl(
      r = df.trial_4_r,
      theta = df.trial_4_theta,
      name = "Trial 4",
      marker=dict(size=22, color = "magenta")
    ))
fig.add_trace(go.Scatterpolargl(
      r = df.trial_5_r,
      theta = df.trial_5_theta,
      name = "Trial 5",
      marker=dict(size=19, color = "limegreen")
      ))
fig.add_trace(go.Scatterpolargl(
      r = df.trial_6_r,
      theta = df.trial_6_theta,
      name = "Trial 6",
      marker=dict(size=10, color = "gold")
      ))

# Common parameters for all traces
fig.update_traces(mode="markers", marker=dict(line_color='white', opacity=0.7))

fig.update_layout(
    title = "Hobbs-Pearson Trials",
    font_size = 15,
    showlegend = False,
    polar = dict(
      bgcolor = "rgb(223, 223, 223)",
      angularaxis = dict(
        linewidth = 3,
        showline=True,
        linecolor='black'
      ),
      radialaxis = dict(
        side = "counterclockwise",
        showline = True,
        linewidth = 2,
        gridcolor = "white",
        gridwidth = 2,
      )
    ),
    paper_bgcolor = "rgb(223, 223, 223)"
)
```

lar

Hobbs-Pearson Trials



Polar Chart Subplots

lar

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=2, cols=2, specs=[[{'type': 'polar'}]*2]*2)

fig.add_trace(go.Scatterpolar(
        r = [1, 2, 3],
        theta = [50, 100, 200],
        marker_symbol = "square"
    ), 1, 1)
fig.add_trace(go.Scatterpolar(
        r = [1, 2, 3],
        theta = [1, 2, 3],
        thetaunit = "radians"
    ), 1, 1)
fig.add_trace(go.Scatterpolar(
        r = ["a", "b", "c", "b"],
        theta = ["D", "C", "B", "A"],
        subplot = "polar2"
    ), 1, 2)
fig.add_trace(go.Scatterpolar(
        r = [50, 300, 900],
        theta = [0, 90, 180],
        subplot = "polar3"
    ), 2, 1)
fig.add_trace(go.Scatterpolar(
        mode = "lines",
        r = [3, 3, 4, 3],
        theta = [0, 45, 90, 270],
        fill = "toself",
        subplot = "polar4"
    ), 2, 2)


fig.update_layout(
    polar = dict(
      radialaxis_range = [1, 4],
      angularaxis_thetaunit = "radians"
    ),
    polar3 = dict(
      radialaxis = dict(type = "log", tickangle = 45),
      sector = [0, 180]
    ),
    polar4 = dict(
      radialaxis = dict(visible = False, range = [0, 6])),
    showlegend = False
)

fig.show()
```

Reference

See function reference for px.(scatter_polar) (https://plotly.com/python-api-reference/generated/plotly.express.scatter_polar) or function reference for px.(line_polar) (https://plotly.com/python-api-reference/generated/plotly.express.line_polar) or https://plotly.com/python/reference/scatterpolar/ (https://plotly.com/python/reference/scatterpolar/) for more information and chart attribute options!

What About Dash?

lar

Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

Learn about how to install Dash at https://dash.plot.ly/installation (https://dash.plot.ly/installation).

Everywhere in this page that you see fig.show(), you can display the same figure in a Dash application by passing it to the figure argument of the Graph component (https://dash.plot.ly/dash-core-components/graph) from the built-in dash_core_components package like this:

```
import plotly.graph_objects as go # or plotly.express as px
fig = go.Figure() # or any Plotly Express function e.g. px.bar(...)
# fig.add_trace( ... )
# fig.update_layout( ... )

from dash import Dash, dcc, html

app = Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run(debug=True, use_reloader=False)  # Turn off reloader if inside Jupyter
```