Distplots in Python

How to make interactive Distplots in Python with Plotly.

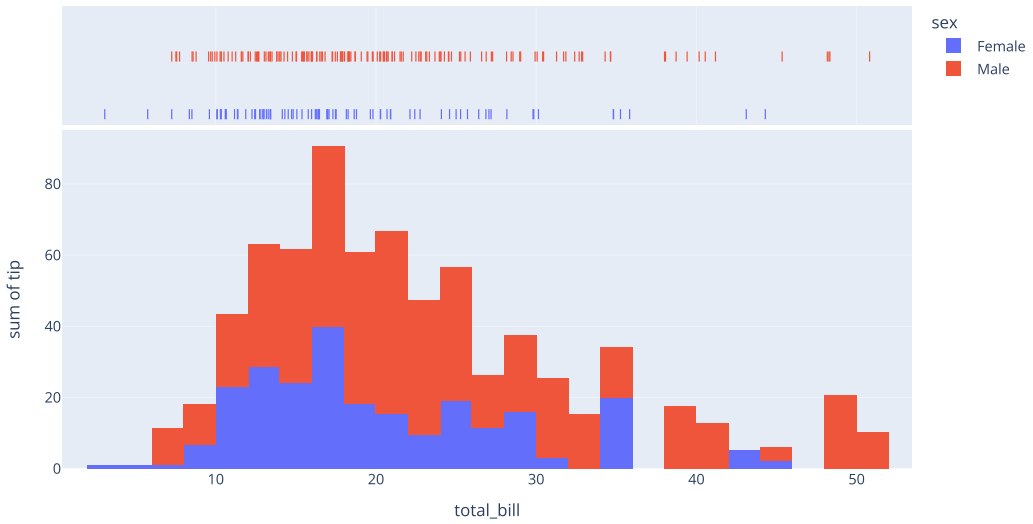
> Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. Sign up for early access now. (https://plotly.com/studio/?utm_medium=graphing_libraries&utm_campaign=studio_early_access&utm_content=sidebar)

Combined statistical representations with px.histogram

Several representations of statistical distributions are available in plotly, such as histograms (https://plotly.com/python/histograms/), violin plots (https://plotly.com/python/violin/), box plots (https://plotly.com/python/box-plots/) (see the complete list here (https://plotly.com/python/statistical-charts/)). It is also possible to combine several representations in the same plot.

For example, the `plotly.express` function `px.histogram` can add a subplot with a different statistical representation than the histogram, given by the parameter `marginal`. 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/).

```
import plotly.express as px
df = px.data.tips()
fig = px.histogram(df, x="total_bill", y="tip", color="sex", marginal="rug",
                   hover_data=df.columns)
fig.show()
```



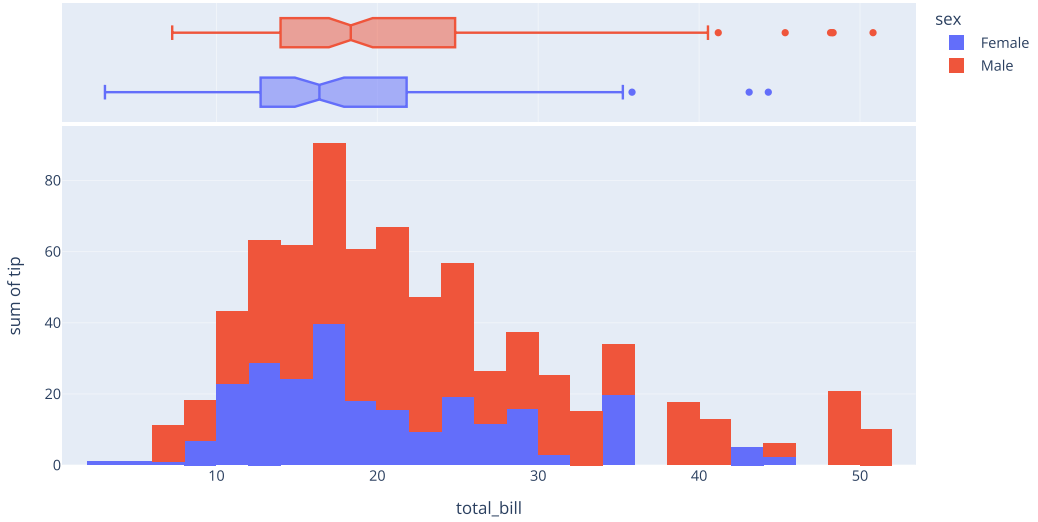
```
import plotly.express as px
df = px.data.tips()
fig = px.histogram(df, x="total_bill", y="tip", color="sex",
                   marginal="box", # or violin, rug
                   hover_data=df.columns)
fig.show()
```

s with

s in

s with

Sizes



Combined statistical representations in Dash

Dash (https://plotly.com/dash/) is the best way to build analytical apps in Python using Plotly figures. To run the app below, run `pip install dash`, click "Download" to get the code and run `python app.py`.

Get started with the official Dash docs (https://dash.plotly.com/installation) and **learn how to effortlessly** style (https://plotly.com/dash/design-kit/) **&** deploy (https://plotly.com/dash/app-manager/) **apps like this with** Dash Enterprise (https://plotly.com/dash/)**.**

s with

s in

s with

Sizes

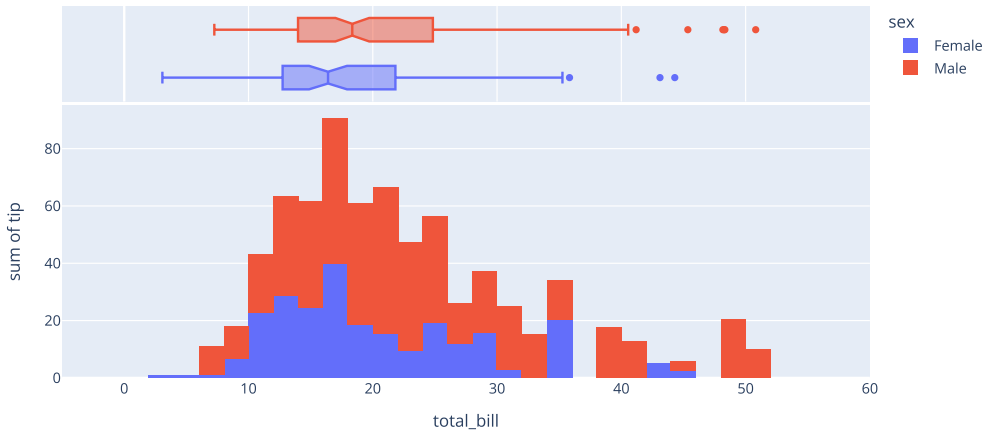
```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px

app = Dash(__name__)

app.layout = html.Div([
    html.H4("Analysis of the restaurant's revenue"),
    html.P("Select Distribution:"),
    dcc.RadioItems(
        id='distribution',
        options=['box', 'violin', 'rug'],
        value='box', inline=True
    ),
    dcc.Graph(id="graph"),
])


@app.callback(
    Output("graph", "figure"),
    Input("distribution", "value"))
def display_graph(distribution):
    df = px.data.tips() # replace with your own data source
    fig = px.histogram(
        df, x="total_bill", y="tip", color="sex",
        marginal=distribution, range_x=[-5, 60],
```

DOWNLOAD

Analysis of the restaurant's revenue

Select Distribution:

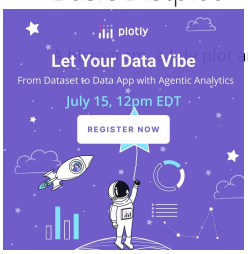
box violin rug



Combined statistical representations with distplot figure factory

The distplot figure factory (/python/figure-factories/) displays a combination of statistical representations of numerical data, such as histogram, kernel density estimation or normal curve, and rug plot.

Basic Distplot



nd a rug plot are displayed.

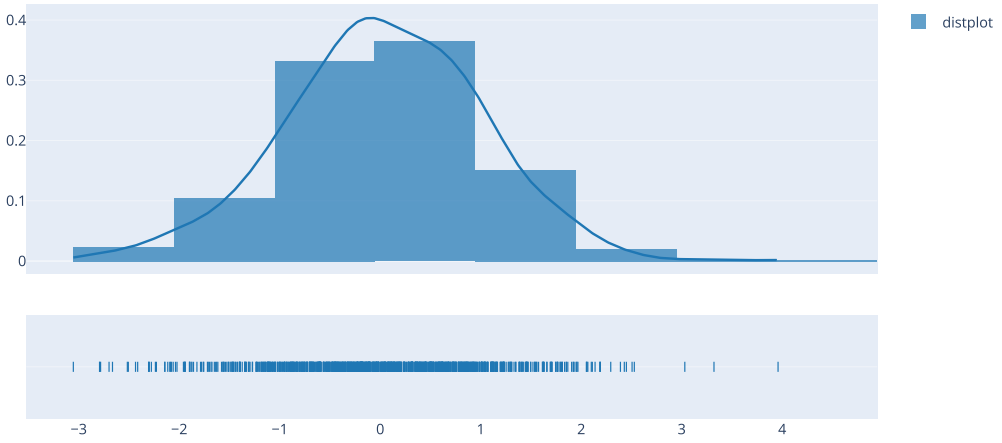
```
import plotly.figure_factory as ff
import numpy as np
np.random.seed(1)

x = np.random.randn(1000)
hist_data = [x]
group_labels = ['distplot'] # name of the dataset

fig = ff.create_distplot(hist_data, group_labels)
fig.show()
```

s with

s in

s with

Sizes



Plot Multiple Datasets

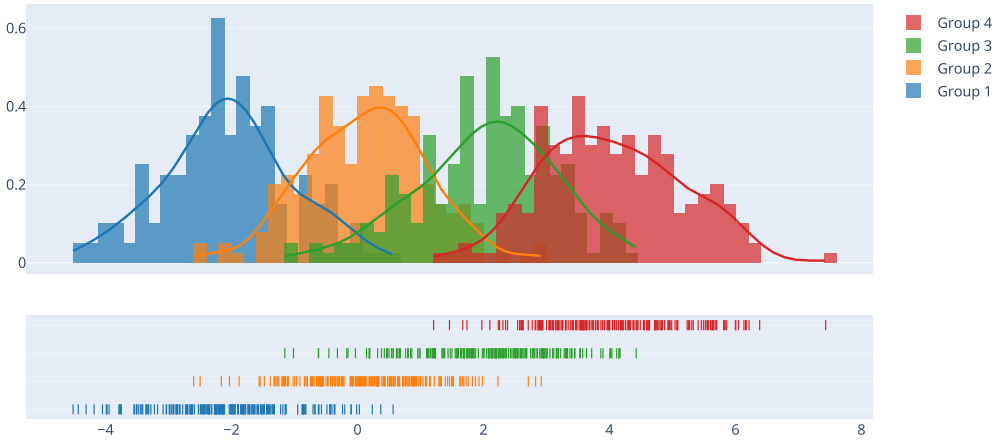
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

# Add histogram data
x1 = np.random.randn(200) - 2
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 2
x4 = np.random.randn(200) + 4

# Group data together
hist_data = [x1, x2, x3, x4]

group_labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4']

# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels, bin_size=.2)
fig.show()
```

Sizes



Use Multiple Bin Sizes

Different bin sizes are used for the different datasets with the `bin_size` argument.

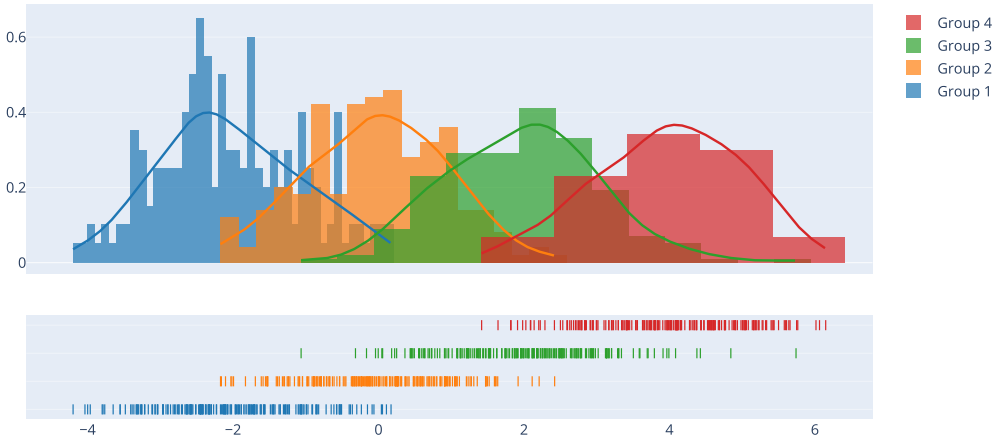
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

# Add histogram data
x1 = np.random.randn(200)-2
x2 = np.random.randn(200)
x3 = np.random.randn(200)+2
x4 = np.random.randn(200)+4

# Group data together
hist_data = [x1, x2, x3, x4]

group_labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4']

# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels, bin_size=[.1, .25, .5, 1])
fig.show()
```

Sizes



Customize Rug Text, Colors & Title

s with

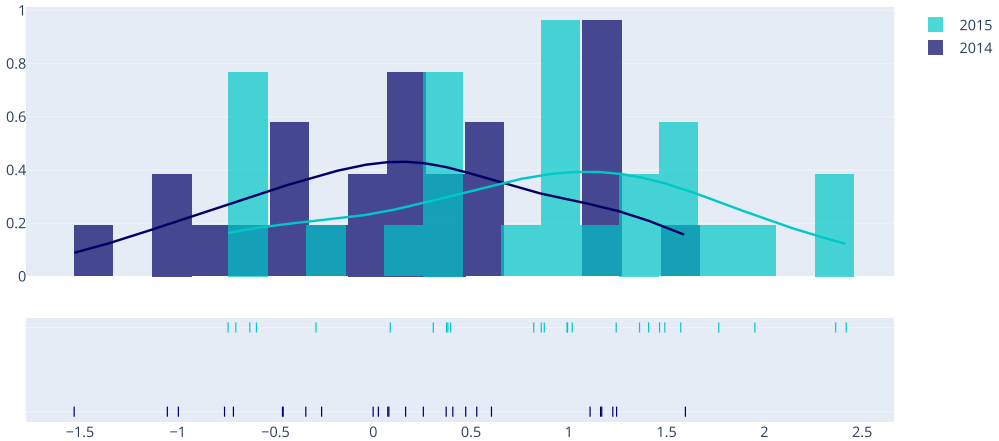
s in

s with

Sizes

```
import plotly.figure_factory as ff
import numpy as np

x1 = np.random.randn(26)
x2 = np.random.randn(26) + .5

group_labels = ['2014', '2015']

rug_text_one = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
                'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
                'u', 'v', 'w', 'x', 'y', 'z']

rug_text_two = ['aa', 'bb', 'cc', 'dd', 'ee', 'ff', 'gg', 'hh', 'ii', 'jj',
                'kk', 'll', 'mm', 'nn', 'oo', 'pp', 'qq', 'rr', 'ss', 'tt',
                'uu', 'vv', 'ww', 'xx', 'yy', 'zz']

rug_text = [rug_text_one, rug_text_two] # for hover in rug plot
colors = ['rgb(0, 0, 100)', 'rgb(0, 200, 200)']

# Create distplot with custom bin_size
fig = ff.create_distplot(
    [x1, x2], group_labels, bin_size=.2,
    rug_text=rug_text, colors=colors)

fig.update_layout(title_text='Customized Distplot')
fig.show()
```

Customized Distplot



Plot Normal Curve

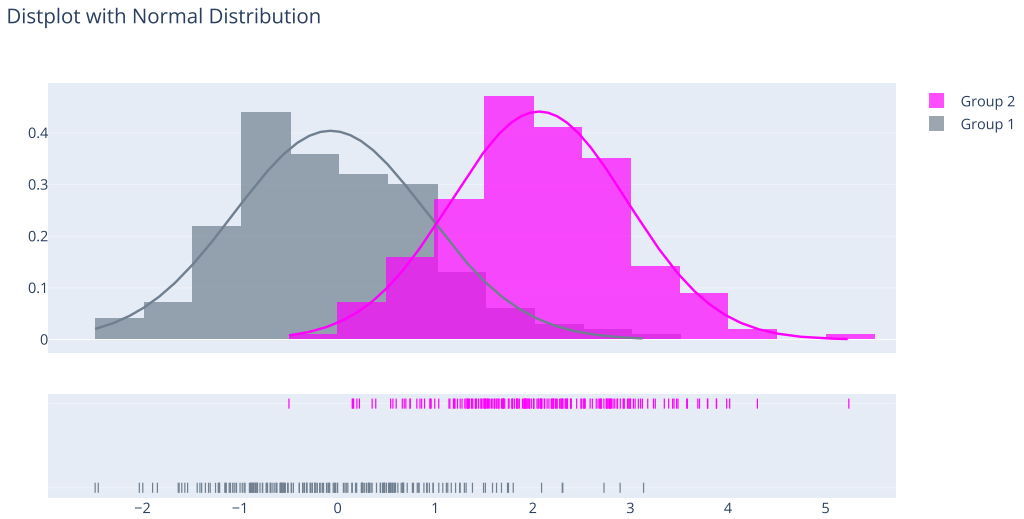
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

x1 = np.random.randn(200)
x2 = np.random.randn(200) + 2

group_labels = ['Group 1', 'Group 2']

colors = ['slategray', 'magenta']

# Create distplot with curve_type set to 'normal'
fig = ff.create_distplot([x1, x2], group_labels, bin_size=.5,
                         curve_type='normal', # override default 'kde'
                         colors=colors)

# Add title
fig.update_layout(title_text='Distplot with Normal Distribution')
fig.show()
```

Sizes



Plot Only Curve and Rug

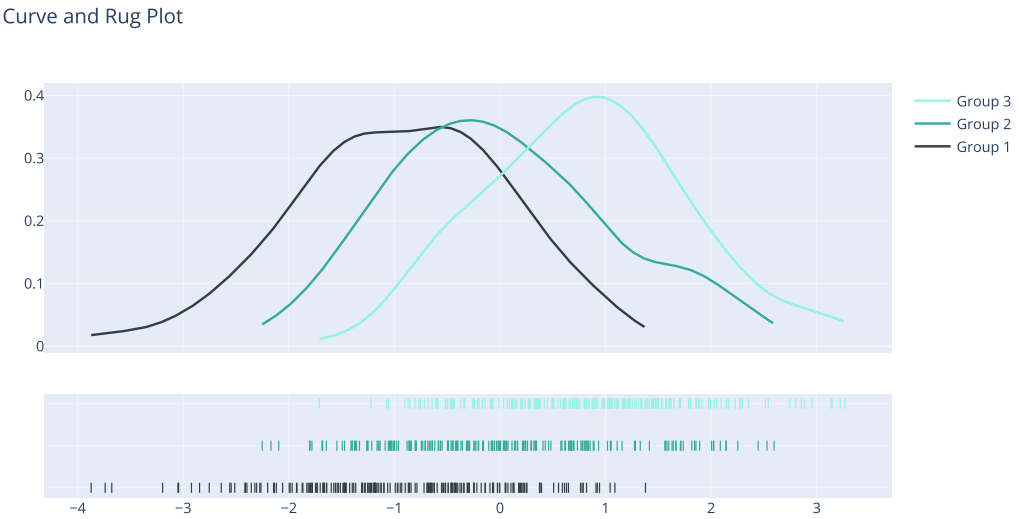
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

x1 = np.random.randn(200) - 1
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 1

hist_data = [x1, x2, x3]

group_labels = ['Group 1', 'Group 2', 'Group 3']
colors = ['#333F44', '#37AA9C', '#94F3E4']

# Create distplot with curve_type set to 'normal'
fig = ff.create_distplot(hist_data, group_labels, show_hist=False, colors=colors)

# Add title
fig.update_layout(title_text='Curve and Rug Plot')
fig.show()
```

Sizes



Plot Only Hist and Rug

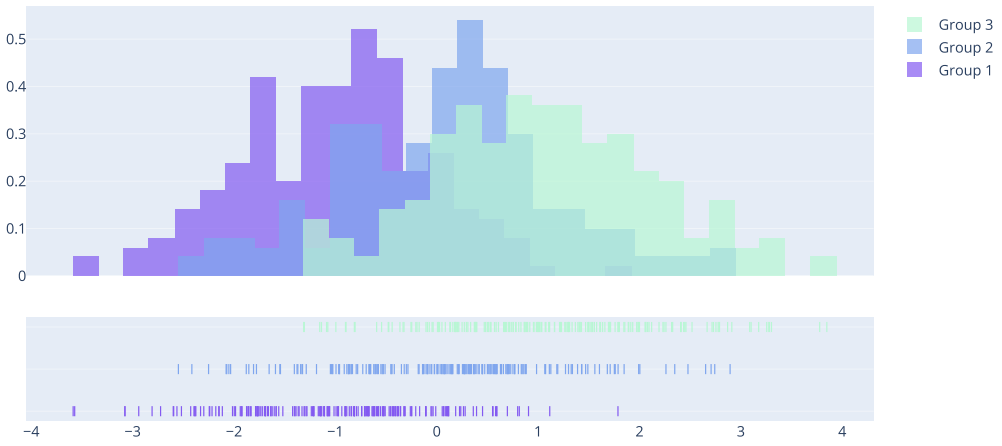
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

x1 = np.random.randn(200) - 1
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 1

hist_data = [x1, x2, x3]

group_labels = ['Group 1', 'Group 2', 'Group 3']
colors = ['#835AF1', '#7FA6EE', '#B8F7D4']

# Create distplot with curve_type set to 'normal'
fig = ff.create_distplot(hist_data, group_labels, colors=colors, bin_size=.25,
                         show_curve=False)

# Add title
fig.update_layout(title_text='Hist and Rug Plot')
fig.show()
```

Sizes



Plot Hist and Rug with Different Bin Sizes

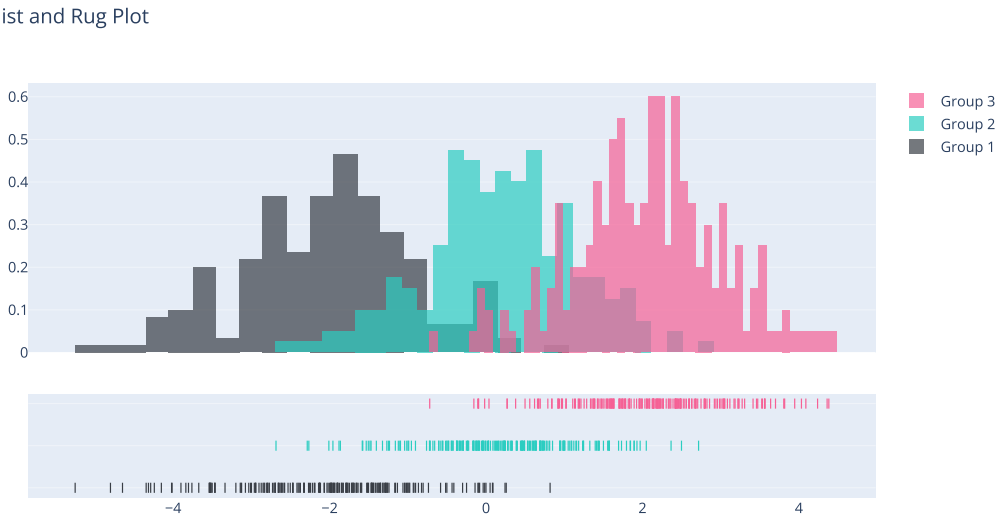
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

x1 = np.random.randn(200) - 2
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 2

hist_data = [x1, x2, x3]

group_labels = ['Group 1', 'Group 2', 'Group 3']
colors = ['#393E46', '#2BCDC1', '#F66095']

fig = ff.create_distplot(hist_data, group_labels, colors=colors,
                         bin_size=[0.3, 0.2, 0.1], show_curve=False)

# Add title
fig.update(layout_title_text='Hist and Rug Plot')
fig.show()
```

Sizes



Plot Only Hist and Curve

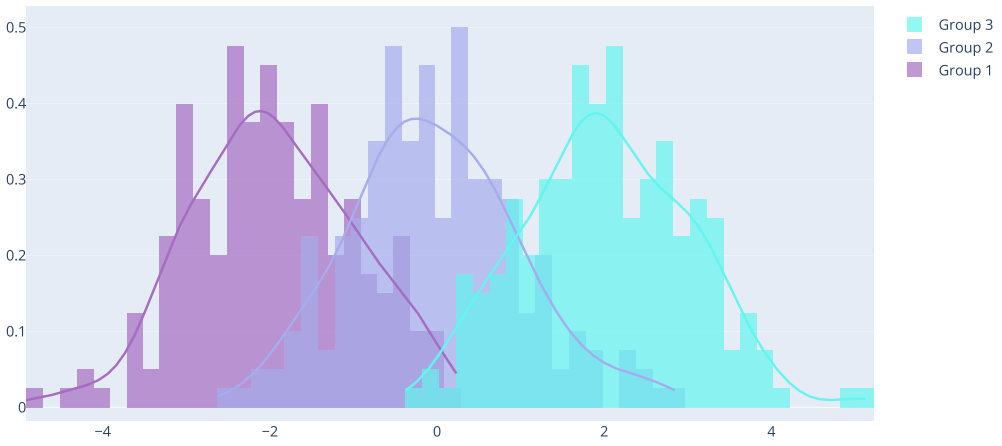
s with

s in

s with

```
import plotly.figure_factory as ff
import numpy as np

x1 = np.random.randn(200) - 2
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 2

hist_data = [x1, x2, x3]

group_labels = ['Group 1', 'Group 2', 'Group 3']
colors = ['#A56CC1', '#A6ACEC', '#63F5EF']

# Create distplot with curve_type set to 'normal'
fig = ff.create_distplot(hist_data, group_labels, colors=colors,
                         bin_size=.2, show_rug=False)

# Add title
fig.update_layout(title_text='Hist and Curve Plot')
fig.show()
```

Sizes



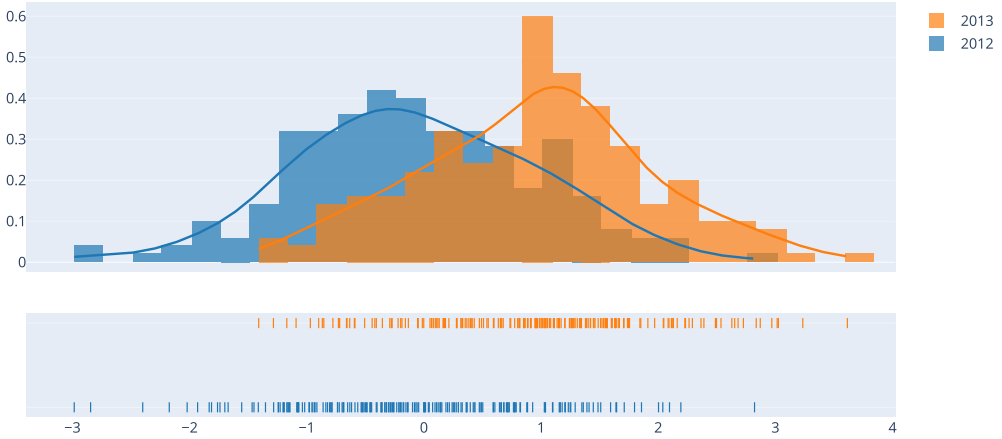
Distplot with Pandas

```
import plotly.figure_factory as ff
import numpy as np
import pandas as pd

df = pd.DataFrame({'2012': np.random.randn(200),
                   '2013': np.random.randn(200)+1})
fig = ff.create_distplot([df[c] for c in df.columns], df.columns, bin_size=.25)
fig.show()
```

s with

s in

s with



Sizes

Reference

For more info on ff.create_distplot(), see the full function reference (https://plotly.com/python-api-reference/generated/plotly.figure_factory.create_distplot.html)

What About Dash?

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
```

s with

s in

s with

JOIN OUR MAILING LIST

Sign up to stay in the loop with all things Plotly — from Dash Club to product updates, webinars, and more!

SUBSCRIBE (HTTPS://GO.PLOT.LY/SUBSCRIPTION)

Products

Dash (https://plotly.com/dash/)
Consulting and Training (https://plotly.com/consulting-and-oem/)

Pricing

Enterprise Pricing (https://plotly.com/get-pricing/)

Sizes

About Us

Careers (https://plotly.com/careers)
Resources (https://plotly.com/resources/)
Blog (https://medium.com/@plotlygraphs)

Support

Community Support (https://community.plot.ly/)
Documentation (https://plotly.com/graphing-libraries)

Terms of Service (https://community.plotly.com/tos) Privacy Policy (https://plotly.com/privacy/)