Visualization

* [Matplotlib](#_Matplotlib)
* [Seaborn](#_Seaborn)
* [Plotly Cufflinks](#_PLOTLY_AND_CUFFLINKS)
* [Plotly](#_PLOTLY_INDEPENDENT_CHARTS)
* [Dash](#_DASH)
* [Graphic Continuum](#_GRAPIC_CONTINUUM)
* Altair – Vega
* Panel

# Matplotlib

|  |  |
| --- | --- |
| Basic plot | plt.plot(x, y, 'r') # 'r' is the color red  plt.xlabel('X Axis Title Here')  plt.ylabel('Y Axis Title Here')  plt.title('String Title Here')  plt.show() |
| Multiplot (separate) | # plt.subplot(nrows, ncols, plot\_number)  plt.subplot(1,2,1)  plt.plot(x, y, 'r--') # More on color options later  plt.subplot(1,2,2)  plt.plot(y, x, 'g\*-'); |
| Obj oriented method | # Create Figure (empty canvas)  fig = plt.figure()  # Add set of axes to figure  axes = fig.add\_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height (range 0 to 1)  # Plot on that set of axes  axes.plot(x, y, 'b')  axes.set\_xlabel('Set X Label') # Notice the use of set\_ to begin methods  axes.set\_ylabel('Set y Label')  axes.set\_title('Set Title') |
| Obj oriented method – plot within a plot | # Creates blank canvas  fig = plt.figure()  axes1 = fig.add\_axes([0.1, 0.1, 0.8, 0.8]) # main axes  axes2 = fig.add\_axes([0.2, 0.5, 0.4, 0.3]) # inset axes  # Larger Figure Axes 1  axes1.plot(x, y, 'b')  axes1.set\_xlabel('X\_label\_axes2')  axes1.set\_ylabel('Y\_label\_axes2')  axes1.set\_title('Axes 2 Title')  # Insert Figure Axes 2  axes2.plot(y, x, 'r')  axes2.set\_xlabel('X\_label\_axes2')  axes2.set\_ylabel('Y\_label\_axes2')  axes2.set\_title('Axes 2 Title'); |
| Subplots() | # Use similar to plt.figure() except use tuple unpacking to grab fig and axes  fig, axes = plt.subplots()  # Now use the axes object to add stuff to plot  axes.plot(x, y, 'r')  axes.set\_xlabel('x')  axes.set\_ylabel('y')  axes.set\_title('title'); |
| Subplots – Multipls | # Empty canvas of 1 by 2 subplots  fig, axes = plt.subplots(nrows=1, ncols=2)  fig, axes = plt.subplots(nrows=1, ncols=2)  for ax in axes:  ax.plot(x, y, 'g')  ax.set\_xlabel('x')  ax.set\_ylabel('y')  ax.set\_title('title')  fig  plt.tight\_layout() |
| Size and aspect ratio | fig = plt.figure(figsize=(8,4), dpi=100) |
| Saving figures | fig.savefig("filename.png", dpi=200) |
| Legends (and labels) | fig = plt.figure()  ax = fig.add\_axes([0,0,1,1])  ax.plot(x, x\*\*2, label="x\*\*2")  ax.plot(x, x\*\*3, label="x\*\*3")  ax.legend()  Location  ax.legend(loc=1) # upper right corner  ax.legend(loc=2) # upper left corner  ax.legend(loc=3) # lower left corner  ax.legend(loc=4) # lower right corner  # .. many more options are available  # Most common to choose  ax.legend(loc=0) # let matplotlib decide the optimal location |
| Formatting | # marker size and color  ax.plot(x, x+13, color="purple", lw=1, ls='-', marker='o', markersize=2)  ax.plot(x, x+14, color="purple", lw=1, ls='-', marker='o', markersize=4)  ax.plot(x, x+15, color="purple", lw=1, ls='-', marker='o', markersize=8, markerfacecolor="red")  ax.plot(x, x+16, color="purple", lw=1, ls='-', marker='s', markersize=8,  markerfacecolor="yellow", markeredgewidth=3, markeredgecolor="green"); |

# Seaborn

Distribution plots

|  |  |
| --- | --- |
| \* distplot | sns.distplot(tips['total\_bill'],kde=False,bins=30) |
| \* jointplot | Kind = \* “scatter” \* “reg” \* “resid” \* “kde” \* “hex”  sns.jointplot(x='total\_bill',y='tip',data=tips,kind='scatter') |
| \* pairplot | sns.pairplot(tips,hue='sex',palette='coolwarm') |
| \* rugplot | sns.rugplot(tips['total\_bill']) |
| \* kdeplot | sns.kdeplot(tips['tip']) |

Regression plot

|  |  |
| --- | --- |
| \* lmplot | sns.lmplot(x='total\_bill',y='tip',data=tips,col='day',hue='sex',palette='coolwarm',  aspect=0.6,size=8) |

Categorical plots

|  |  |
| --- | --- |
| \* barplot | sns.barplot(x='sex',y='total\_bill',data=tips) – default is mean  sns.barplot(x='sex',y='total\_bill',data=tips,estimator=np.std) |
| \* countplot | sns.countplot(x='sex',data=tips) |
| \* boxplot | sns.boxplot(x="day", y="total\_bill", hue="smoker",data=tips, palette="coolwarm") |
| \* violinplot | sns.violinplot(x="day", y="total\_bill", data=tips,hue='sex',palette='Set1')  sns.violinplot(x="day", y="total\_bill", data=tips,hue='sex',split=True,palette='Set1') – split violin plot |
| \* stripplot | sns.stripplot(x="day", y="total\_bill", data=tips,jitter=True,hue='sex',palette='Set1',split=True) – with jitter and split |
| \* swarmplot | sns.swarmplot(x="day", y="total\_bill",hue='sex',data=tips, palette="Set1", split=True) |
| Combining plots | sns.violinplot(x="tip", y="day", data=tips,palette='rainbow')  sns.swarmplot(x="tip", y="day", data=tips,color='black',size=3) |
| \* factorplot | sns.factorplot(x='sex',y='total\_bill',data=tips,kind='bar')  factorplot is the most general form of a categorical plot. It can take in a **kind** parameter to adjust the plot type |

Matrix plots – both row and column should have headings like Pivot table

|  |  |
| --- | --- |
| \* Heat map | sns.heatmap(tips.corr(),cmap='coolwarm',annot=True) |
| \* Cluster maps | sns.clustermap(pvflights,cmap='coolwarm',standard\_scale=1) – Heat map with the cluster tree diagrams on the axis |

Grids

|  |  |
| --- | --- |
| Pair grid | # Map to upper,lower, and diagonal  g = sns.PairGrid(iris)  g.map\_diag(plt.hist)  g.map\_upper(plt.scatter)  g.map\_lower(sns.kdeplot) |
| Pair plot | sns.pairplot(iris,hue='species',palette='rainbow') |
| Facet grid | g = sns.FacetGrid(tips, col="time", row="smoker")  g = g.map(plt.hist, "total\_bill")  2 argument example  g = sns.FacetGrid(tips, col="time", row="smoker",hue='sex')  # Notice how the arguments come after plt.scatter call  g = g.map(plt.scatter, "total\_bill", "tip").add\_legend() |
| Joint Grid | General version of joint plot  g = sns.JointGrid(x="total\_bill", y="tip", data=tips)  g = g.plot(sns.regplot, sns.distplot)  (reg plot is a scatter with the regression lines and confidence intervals as a shaded area) |
|  |  |

|  |  |
| --- | --- |
|  | Style and color |
| Style and palette | sns.set\_style('white')  sns.set\_style('ticks')  sns.countplot(x='sex',data=tips,palette='deep') |
| Spine removal | sns.countplot(x='sex',data=tips)  sns.despine() |
| Size and aspect | # Non Grid Plot  plt.figure(figsize=(12,3))  sns.countplot(x='sex',data=tips)  # Grid Type Plot  sns.lmplot(x='total\_bill',y='tip',size=2,aspect=4,data=tips) |
| Scale and context | # Large font in the axis  sns.set\_context('poster',font\_scale=4)  sns.countplot(x='sex',data=tips,palette='coolwarm') |
| Subplots | fig,ax = plt.subplots(1,3,figsize=(14,6))  df\_out.transpose().plot(kind='bar',stacked=True, ax=ax[0])  df\_out.plot(kind='bar',stacked=True, ax=ax[1], cmap='coolwarm') #Pandas plot  ax[2]=sns.heatmap(df\_out, cmap='coolwarm', annot=True, fmt='g') #seaborn plot |

# Pandas inbuilt visualization

|  |  |
| --- | --- |
|  | There are several plot types built-in to pandas, most of them statistical plots by nature:  \* df.plot.area  \* df.plot.barh  \* df.plot.density  \* df.plot.hist  \* df.plot.line  \* df.plot.scatter  \* df.plot.bar  \* df.plot.box  \* df.plot.hexbin  \* df.plot.kde  \* df.plot.pie  You can also just call df.plot(kind='hist') or replace that kind argument with any of the key terms shown in the list above (e.g. 'box','barh', etc..)  \_\_\_ |
| Styles | plt.style.use('dark\_background')  df1['A'].hist()  df2.plot.area(alpha=0.4)  df1.plot.scatter(x='A',y='B',c='C',cmap='coolwarm')  df.plot.hexbin(x='a',y='b',gridsize=25,cmap='Oranges') |

# PLOTLY AND CUFFLINKS IN JUYTER NOTEBOOK

|  |  |
| --- | --- |
|  |  |
| Imports | # Standard plotly imports,  import plotly.plotly as py,  import plotly.graph\_objs as go,  from plotly.offline import iplot, init\_notebook\_mode,  # Using plotly + cufflinks in offline mode,  import cufflinks,  cufflinks.go\_offline(connected=True),  init\_notebook\_mode(connected=True) |
| Plots | df['BillQ'].iplot() |
|  | df['BillQ'].iplot(kind='hist', xTitle='Sales per day', yTitle='count', title='Sales Distribution') |
|  | df\_pivot\_normalized.iplot(kind='heatmap', colorscale = 'greys', title=”Fraction of Annual Sales”, xTitle=”Months”, yTitle = “Years”) |

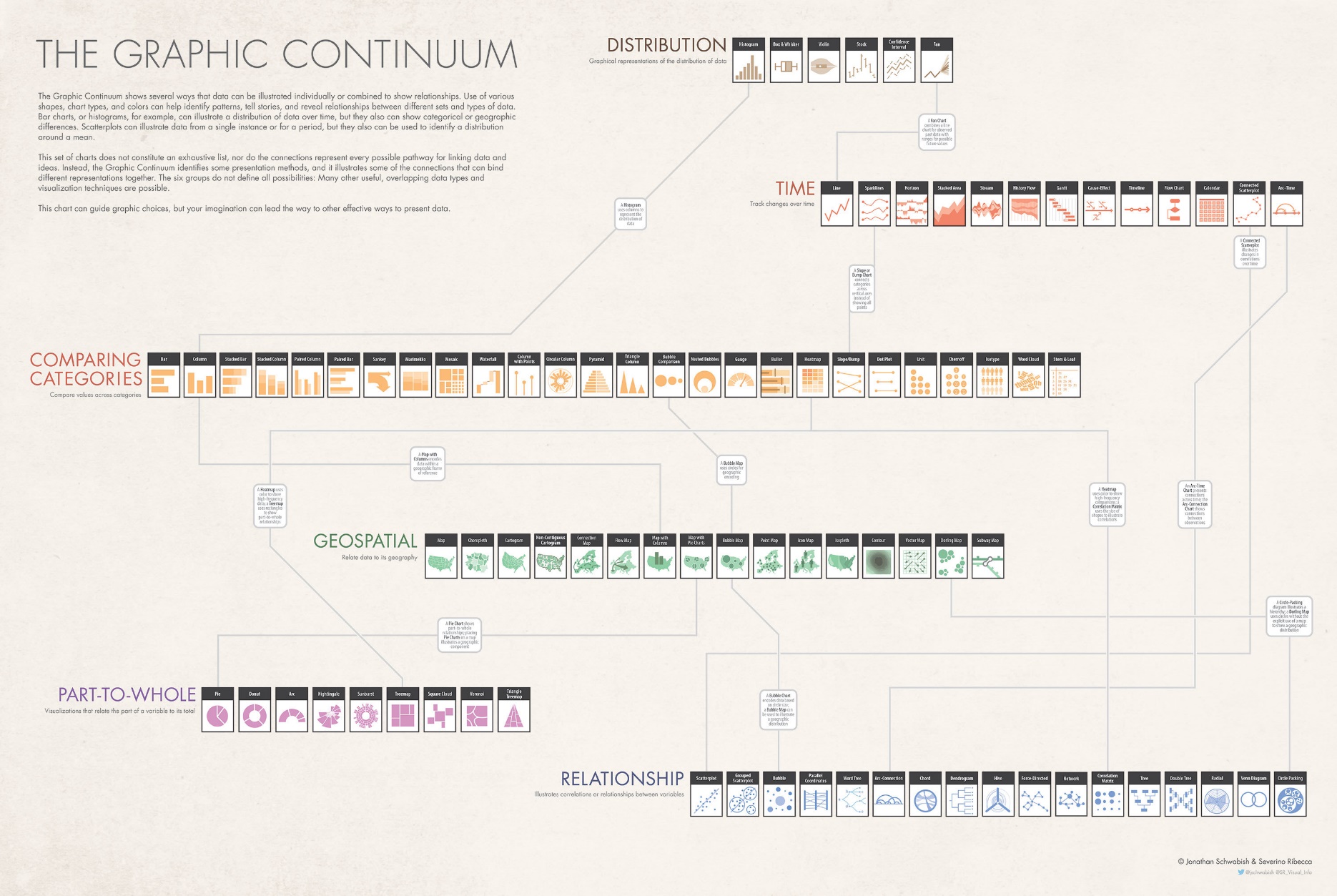
# PLOTLY INDEPENDENT CHARTS

|  |  |
| --- | --- |
| Imports | import plotly.offline as pyo  import plotly.graph\_objs as go |
| Overview | Define the process steps   1. Data – graph objects (go) 2. Layout 3. Figure – combine Data , Layout 4. Plot figure with plotly offline |
|  | **SCATTER PLOTS** |
| Step 1 | data = [go.Scatter(  x = random\_x,  y = random\_y,  mode = 'markers',  marker = dict( # change the marker style  size = 12,  color = 'rgb(51,204,153)',  symbol = 'pentagon',  line = dict(  width = 2,  )  )  )] |
| Step 2 | layout = go.Layout(  title = 'Random Data Scatterplot', # Graph title  xaxis = dict(title = 'Some random x-values'), # x-axis label  yaxis = dict(title = 'Some random y-values'), # y-axis label  hovermode ='closest' # handles multiple points landing on the same vertical  ) |
| Step 3 | fig = go.Figure(data=data, layout=layout) |
| Step 4 | pyo.plot(fig, filename='scatter3.html') |
|  | **LINE PLOTS** |
|  | ***# Multiple lines in a plot – see the traces is a list of data plots***  traces=[go.Scatter(x = df2.columns, y = df2.loc[name], mode = 'markers+lines', name = name ) for name in df2.index] layout = go.Layout(title = 'Population Estimates of the Six New England States' ) fig = go.Figure(data=traces, layout=layout)  pyo.plot(fig, filename='line3.html') |
|  | **BAR CHARTS (including stacked bar/multiple bars)** |
|  | trace1 = go.Bar(x=df['NOC'], # NOC stands for National Olympic Committee y=df['Gold'], name = 'Gold', marker=dict(color='#FFD700') # set the marker color to gold ) trace2 = go.Bar(x=df['NOC'], y=df['Silver'], name='Silver', marker=dict(color='#9EA0A1') # set the marker color to silver )  trace3 = go.Bar(x=df['NOC'],  y=df['Bronze'],  name='Bronze', marker=dict(color='#CD7F32') # set the marker color to bronze ) data = [trace1, trace2, trace3]  layout = go.Layout(title='2018 Winter Olympic Medals by Country',  barmode='stack')  fig = go.Figure(data=data, layout=layout) pyo.plot(fig, filename='bar3.html') |
|  | BUBBLE CHARTS |
|  | data = [go.Scatter( x=df['horsepower'],  y=df['mpg'],  text=df['text2'], # use the new column for the hover text mode='markers', marker=dict(size=1.5\*df['cylinders']))]  layout = go.Layout(title='Vehicle mpg vs. horsepower', hovermode='closest')  fig = go.Figure(data=data, layout=layout) pyo.plot(fig, filename='bubble2.html') |
|  | BOX PLOTS |
|  | data = [ go.Box(y=snodgrass, name='QCS'),  go.Box(y=twain, name='MT')] layout = go.Layout(  title = 'Comparison of three-letter-word frequencies between Quintus Curtius Snodgrass and Mark Twain')  fig = go.Figure(data=data, layout=layout) pyo.plot(fig, filename='box3.html') |
| **FIGURE FACTORY** | HISTOGRAM and RUG PLOTS |
|  | import plotly.offline as pyo  import plotly.figure\_factory as ff snodgrass = [.209,.205,.196,.210,.202,.207,.224,.223,.220,.201] twain = [.225,.262,.217,.240,.230,.229,.235,.217] hist\_data = [snodgrass,twain]  group\_labels = ['Snodgrass','Twain']  fig = ff.create\_distplot(hist\_data, group\_labels, bin\_size=[.005,.005]) pyo.plot(fig, filename='SnodgrassTwainDistplot.html') |
| **SUBPLOTS** | HEATMAPS |
| **3 CHARTS IN ONE** | import plotly.offline as pyo  import plotly.graph\_objs as go  from plotly import tools  import pandas as pd  df1 = pd.read\_csv('../data/2010SitkaAK.csv')  df2 = pd.read\_csv('../data/2010SantaBarbaraCA.csv')  df3 = pd.read\_csv('../data/2010YumaAZ.csv')  trace1 = go.Heatmap(  x=df1['DAY'],  y=df1['LST\_TIME'],  z=df1['T\_HR\_AVG'],  colorscale='Jet',  zmin = 5, zmax = 40 # add max/min color values to make each plot consistent  )  trace2 = go.Heatmap(  x=df2['DAY'],  y=df2['LST\_TIME'],  z=df2['T\_HR\_AVG'],  colorscale='Jet',  zmin = 5, zmax = 40  )  trace3 = go.Heatmap(  x=df3['DAY'],  y=df3['LST\_TIME'],  z=df3['T\_HR\_AVG'],  colorscale='Jet',  zmin = 5, zmax = 40  )  fig = tools.make\_subplots(rows=1, cols=3,  subplot\_titles=('Sitka, AK','Santa Barbara, CA', 'Yuma, AZ'),  shared\_yaxes = True, # this makes the hours appear only on the left  )  fig.append\_trace(trace1, 1, 1)  fig.append\_trace(trace2, 1, 2)  fig.append\_trace(trace3, 1, 3)  fig['layout'].update( # access the layout directly!  title='Hourly Temperatures, June 1-7, 2010'  )  pyo.plot(fig, filename='AllThree.html') |

# DASH

|  |
| --- |
| PLOTLY DASHBOARD  Step 1: Standard Imports  import dash  import dash\_core\_components as dcc  import dash\_html\_components as html  from dash.dependencies import Input, Output  Step 2: Create the app  app = dash.Dash()  Step 3: Create layout and put all the objects with ids   * Put the html Divs * Put the dcc.Graph or dcc.Dropdown/Sliders   Step 4: Create the call back  Link the inputs and Outputs  (Output is single, Input is a list)  Step 5: Create the function for the update with inputs in same order and send back the inputs for Graph object |
| #######  # This is Dash's tutorial script for multiple inputs  # using Chris Parmer's indicators.csv dataset  ######  import dash  import dash\_core\_components as dcc  import dash\_html\_components as html  from dash.dependencies import Input, Output  import plotly.graph\_objs as go  import pandas as pd  app = dash.Dash()  df = pd.read\_csv(  'https://gist.githubusercontent.com/chriddyp/'  'cb5392c35661370d95f300086accea51/raw/'  '8e0768211f6b747c0db42a9ce9a0937dafcbd8b2/'  'indicators.csv')  available\_indicators = df['Indicator Name'].unique()  app.layout = html.Div([  html.Div([  html.Div([  dcc.Dropdown(  id='xaxis-column',  options=[{'label': i, 'value': i} for i in available\_indicators],  value='Fertility rate, total (births per woman)'  ),  dcc.RadioItems(  id='xaxis-type',  options=[{'label': i, 'value': i} for i in ['Linear', 'Log']],  value='Linear',  labelStyle={'display': 'inline-block'}  )  ],  style={'width': '48%', 'display': 'inline-block'}),  html.Div([  dcc.Dropdown(  id='yaxis-column',  options=[{'label': i, 'value': i} for i in available\_indicators],  value='Life expectancy at birth, total (years)'  ),  dcc.RadioItems(  id='yaxis-type',  options=[{'label': i, 'value': i} for i in ['Linear', 'Log']],  value='Linear',  labelStyle={'display': 'inline-block'}  )  ],style={'width': '48%', 'float': 'right', 'display': 'inline-block'})  ]),  dcc.Graph(id='indicator-graphic'),  dcc.Slider(  id='year--slider',  min=df['Year'].min(),  max=df['Year'].max(),  value=df['Year'].max(),  step=None,  marks={str(year): str(year) for year in df['Year'].unique()}  )  ], style={'padding':10})  @app.callback(  Output('indicator-graphic', 'figure'),  [Input('xaxis-column', 'value'),  Input('yaxis-column', 'value'),  Input('xaxis-type', 'value'),  Input('yaxis-type', 'value'),  Input('year--slider', 'value')])  def update\_graph(xaxis\_column\_name, yaxis\_column\_name,  xaxis\_type, yaxis\_type,  year\_value):  dff = df[df['Year'] == year\_value]  return {  'data': [go.Scatter(  x=dff[dff['Indicator Name'] == xaxis\_column\_name]['Value'],  y=dff[dff['Indicator Name'] == yaxis\_column\_name]['Value'],  text=dff[dff['Indicator Name'] == yaxis\_column\_name]['Country Name'],  mode='markers',  marker={  'size': 15,  'opacity': 0.5,  'line': {'width': 0.5, 'color': 'white'}  }  )],  'layout': go.Layout(  xaxis={  'title': xaxis\_column\_name,  'type': 'linear' if xaxis\_type == 'Linear' else 'log'  },  yaxis={  'title': yaxis\_column\_name,  'type': 'linear' if yaxis\_type == 'Linear' else 'log'  },  margin={'l': 40, 'b': 40, 't': 10, 'r': 0},  hovermode='closest'  )  }  if \_\_name\_\_ == '\_\_main\_\_':  app.run\_server() |

# GRAPIC CONTINUUM



# ALTAIR-VEGA

|  |  |
| --- | --- |
| Grammer of visualization | 1. **Data – whats to be plotted** 2. **Transform – grouping, binning, means etv** 3. **Marks – representation of the data** 4. **Encoding – linking the data such as columns to marks** 5. **Scale – actual to visual scaling** 6. **Guides – axis – legends etc**  * **Data**, **Marks**, and **Encodings**: the three core pieces of an Altair chart * **Encoding Types**: Q (quantitative), N (nominal), O (ordinal), T (temporal), which drive the visual representation of the encodings * **Binning and Aggregation**: which let you control aspects of the data representation within Altair. |
| Imports | **import** **altair** **as** **alt**  *# Altair plots render by default in JupyterLab and nteract*  *# Uncomment/run this line to enable Altair in the classic notebook (not in JupyterLab)*  *# alt.renderers.enable('notebook')*  *# Uncomment/run this line to enable Altair in Colab*  *# alt.renderers.enable('colab')*  #For large data sets, reoves limit of 5000 and saves to json  alt.data\_transformers.disable\_max\_rows()  alt.data\_transformers.enable('json') |
|  | | **Mark Name** | **Method** | **Description** | **Example** | | --- | --- | --- | --- | | area | [mark\_area()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_area) | A filled area plot. | [Simple Stacked Area Chart](https://altair-viz.github.io/gallery/simple_stacked_area_chart.html#gallery-simple-stacked-area-chart) | | bar | [mark\_bar()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_bar) | A bar plot. | [Simple Bar Chart](https://altair-viz.github.io/gallery/simple_bar_chart.html#gallery-simple-bar-chart) | | circle | [mark\_circle()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_circle) | A scatter plot with filled circles. | [One Dot Per Zipcode](https://altair-viz.github.io/gallery/one_dot_per_zipcode.html#gallery-one-dot-per-zipcode) | | geoshape | [mark\_geoshape()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_geoshape) | A geographic shape | [Choropleth Map](https://altair-viz.github.io/gallery/choropleth.html#gallery-choropleth) | | line | [mark\_line()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_line) | A line plot. | [Simple Line Chart](https://altair-viz.github.io/gallery/simple_line_chart.html#gallery-simple-line-chart) | | point | [mark\_point()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_point) | A scatter plot with configurable point shapes. | [Multi-panel Scatter Plot with Linked Brushing](https://altair-viz.github.io/gallery/scatter_linked_brush.html#gallery-scatter-linked-brush) | | rect | [mark\_rect()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_rect) | A filled rectangle, used for heatmaps | [Simple Heatmap](https://altair-viz.github.io/gallery/simple_heatmap.html#gallery-simple-heatmap) | | rule | [mark\_rule()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_rule) | A vertical or horizontal line spanning the axis. | [Candlestick Chart](https://altair-viz.github.io/gallery/candlestick_chart.html#gallery-candlestick-chart) | | square | [mark\_square()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_square) | A scatter plot with filled squares. | N/A | | text | [mark\_text()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_text) | A scatter plot with points represented by text. | [Bar Chart with Labels](https://altair-viz.github.io/gallery/bar_chart_with_labels.html#gallery-bar-chart-with-labels) | | tick | [mark\_tick()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_tick) | A vertical or horizontal tick mark. | [Simple Strip Plot](https://altair-viz.github.io/gallery/strip_plot.html#gallery-strip-plot) | |
|  | | **Mark Name** | **Method** | **Description** | **Example** | | --- | --- | --- | --- | | box plot | [mark\_boxplot()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_boxplot) | A box plot. | [Box Plot with Min/Max Whiskers](https://altair-viz.github.io/gallery/boxplot.html#gallery-boxplot) | | error band | [mark\_errorband()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_errorband) | A continuous band around a line. | N/A | | error bar | [mark\_errorbar()](https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html#altair.Chart.mark_errorbar) | An errorbar around a point. | N/A | |
|  | | **Data Type** | **Shorthand Code** | **Description** | | --- | --- | --- | | quantitative | Q | a continuous real-valued quantity | | ordinal | O | a discrete ordered quantity | | nominal | N | a discrete unordered category | | temporal | T | a time or date value | | geojson | G | a geographic shape | |
|  | Position Channels:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | x | [X](https://altair-viz.github.io/user_guide/generated/channels/altair.X.html#altair.X) | The x-axis value | [Simple Scatter Plot with Tooltips](https://altair-viz.github.io/gallery/scatter_tooltips.html#gallery-scatter-tooltips) | | y | [Y](https://altair-viz.github.io/user_guide/generated/channels/altair.Y.html#altair.Y) | The y-axis value | [Simple Scatter Plot with Tooltips](https://altair-viz.github.io/gallery/scatter_tooltips.html#gallery-scatter-tooltips) | | x2 | [X2](https://altair-viz.github.io/user_guide/generated/channels/altair.X2.html#altair.X2) | Second x value for ranges | [Error Bars showing Confidence Interval](https://altair-viz.github.io/gallery/errorbars_with_ci.html#gallery-errorbars-with-ci) | | y2 | [Y2](https://altair-viz.github.io/user_guide/generated/channels/altair.Y2.html#altair.Y2) | Second y value for ranges | [Line Chart with Confidence Interval Band](https://altair-viz.github.io/gallery/line_with_ci.html#gallery-line-with-ci) | | longitude | [Longitude](https://altair-viz.github.io/user_guide/generated/channels/altair.Longitude.html#altair.Longitude) | Longitude for geo charts | [Locations of US Airports](https://altair-viz.github.io/gallery/airports.html#gallery-airports) | | latitude | [Latitude](https://altair-viz.github.io/user_guide/generated/channels/altair.Latitude.html#altair.Latitude) | Latitude for geo charts | [Locations of US Airports](https://altair-viz.github.io/gallery/airports.html#gallery-airports) | | longitude2 | [Longitude2](https://altair-viz.github.io/user_guide/generated/channels/altair.Longitude2.html#altair.Longitude2) | Second longitude value for ranges | N/A | | latitude2 | [Latitude2](https://altair-viz.github.io/user_guide/generated/channels/altair.Latitude2.html#altair.Latitude2) | Second latitude value for ranges | N/A | | xerror | [XError](https://altair-viz.github.io/user_guide/generated/channels/altair.XError.html#altair.XError) | The x-axis error value | N/A | | yerror | [YError](https://altair-viz.github.io/user_guide/generated/channels/altair.YError.html#altair.YError) | The y-axis error value | N/A | | xerror2 | [XError2](https://altair-viz.github.io/user_guide/generated/channels/altair.XError2.html#altair.XError2) | The second x-axis error value | N/A | | yerror2 | [YError2](https://altair-viz.github.io/user_guide/generated/channels/altair.YError2.html#altair.YError2) | The second y-axis error value | N/A |   Mark Property Channels:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | color | [Color](https://altair-viz.github.io/user_guide/generated/core/altair.Color.html#altair.Color) | The color of the mark | [Simple Heatmap](https://altair-viz.github.io/gallery/simple_heatmap.html#gallery-simple-heatmap) | | fill | [Fill](https://altair-viz.github.io/user_guide/generated/channels/altair.Fill.html#altair.Fill) | The fill for the mark | N/A | | fillopacity | [FillOpacity](https://altair-viz.github.io/user_guide/generated/channels/altair.FillOpacity.html#altair.FillOpacity) | The opacity of the mark’s fill | N/A | | opacity | [Opacity](https://altair-viz.github.io/user_guide/generated/channels/altair.Opacity.html#altair.Opacity) | The opacity of the mark | [Horizon Graph](https://altair-viz.github.io/gallery/horizon_graph.html#gallery-horizon-graph) | | shape | [Shape](https://altair-viz.github.io/user_guide/generated/channels/altair.Shape.html#altair.Shape) | The shape of the mark | N/A | | size | [Size](https://altair-viz.github.io/user_guide/generated/channels/altair.Size.html#altair.Size) | The size of the mark | [Table Bubble Plot (Github Punch Card)](https://altair-viz.github.io/gallery/table_bubble_plot_github.html#gallery-table-bubble-plot-github) | | stroke | [Stroke](https://altair-viz.github.io/user_guide/generated/channels/altair.Stroke.html#altair.Stroke) | The stroke of the mark | N/A | | strokeopacity | [StrokeOpacity](https://altair-viz.github.io/user_guide/generated/channels/altair.StrokeOpacity.html#altair.StrokeOpacity) | The opacity of the line | N/A | | strokewidth | [StrokeWidth](https://altair-viz.github.io/user_guide/generated/channels/altair.StrokeWidth.html#altair.StrokeWidth) | The width of the line | N/A |   Text and Tooltip Channels:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | text | [Text](https://altair-viz.github.io/user_guide/generated/channels/altair.Text.html#altair.Text) | Text to use for the mark | [Simple Scatter Plot with Labels](https://altair-viz.github.io/gallery/scatter_with_labels.html#gallery-scatter-with-labels) | | key | [Key](https://altair-viz.github.io/user_guide/generated/channels/altair.Key.html#altair.Key) | – | N/A | | tooltip | [Tooltip](https://altair-viz.github.io/user_guide/generated/channels/altair.Tooltip.html#altair.Tooltip) | The tooltip value | [Simple Scatter Plot with Tooltips](https://altair-viz.github.io/gallery/scatter_tooltips.html#gallery-scatter-tooltips) |   Hyperlink Channel:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | href | [Href](https://altair-viz.github.io/user_guide/generated/channels/altair.Href.html#altair.Href) | Hyperlink for points | N/A |   Level of Detail Channel:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | detail | [Detail](https://altair-viz.github.io/user_guide/generated/channels/altair.Detail.html#altair.Detail) | Additional property to group by | [Ranged Dot Plot](https://altair-viz.github.io/gallery/ranged_dot_plot.html#gallery-ranged-dot-plot) |   Order Channel:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | order | [Order](https://altair-viz.github.io/user_guide/generated/channels/altair.Order.html#altair.Order) | Sets the order of the marks | [Connected Scatterplot (Lines with Custom Paths)](https://altair-viz.github.io/gallery/connected_scatterplot.html#gallery-connected-scatterplot) |   Facet Channels:   | **Channel** | **Altair Class** | **Description** | **Example** | | --- | --- | --- | --- | | column | [Column](https://altair-viz.github.io/user_guide/generated/channels/altair.Column.html#altair.Column) | The column of a faceted plot | [Trellis Scatter Plot](https://altair-viz.github.io/gallery/trellis_scatter_plot.html#gallery-trellis-scatter-plot) | | row | [Row](https://altair-viz.github.io/user_guide/generated/channels/altair.Row.html#altair.Row) | The row of a faceted plot | [Becker’s Barley Trellis Plot](https://altair-viz.github.io/gallery/beckers_barley_trellis_plot.html#gallery-beckers-barley-trellis-plot) | |