visualisation

June 29, 2020

1 Introduction to Visualization in Python

1.1 by Shashwat Sridhar

29th June, 2020

1.2 Inspiration

Heavliy inspired by Jake VanderPlas' talk (PyCon 2017)

Links, and further resources at the end of the presentation

2 Outline

- Brief history of visualization in python (aka "what is matplotlib?")
- Why (not) matplotlib?
- Overview of available viz libraries in Python
- Which libraries to use when
- Delving deeper:
  - pandas+seaborn
  - holoviews
  - pyqtgraph

3 Brief history of data visualization in Python

- began with Matplotlib in 2003
  - meant to replicate plotting functionality of Matlab
- actively developed, robust, huge codebase and active community

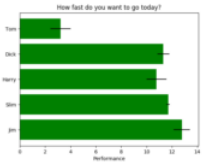
3.1 Matplotlib is great for...

- creating practically any plot/graphic
- embedding into modules/scripts/libraries to generate system agnostic output

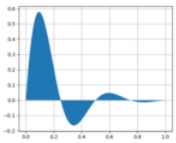
3.2 but not so great for...

- creating practically any plot/graphic quickly/efficiently
- interacting with your visualizations
- working with very large datasets

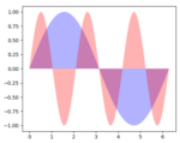
Lines, bars, and markers



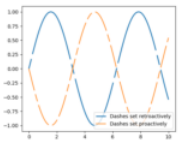
barh_demo



fill_demo



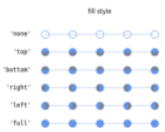
fill_demo_features



line_demo_dash_control

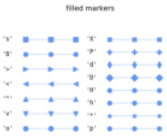
line_styles_reference

linestyles

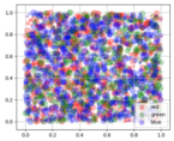


marker_fillstyle_reference

marker_reference

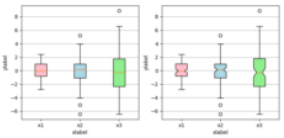


marker_reference

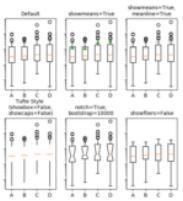


scatter_with_legend

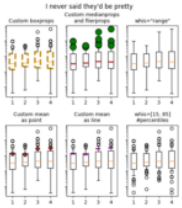
Statistical plots



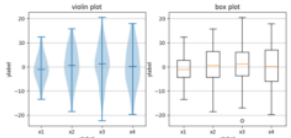
boxplot_color_demo



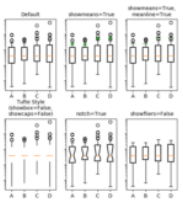
boxplot_demo



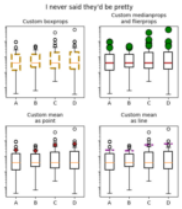
boxplot_demo



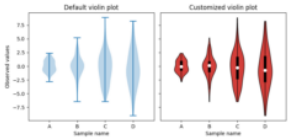
boxplot_vs_violin_demo



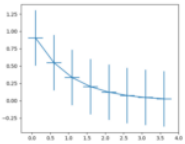
bxp_demo



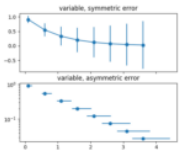
bxp_demo



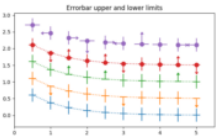
customized_violin_demo



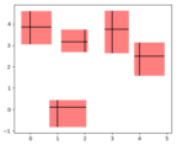
errorbar_demo



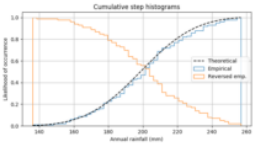
errorbar_demo_features



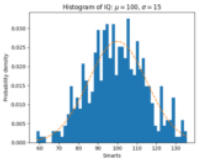
errorbar_limits



errorbars_and_boxes

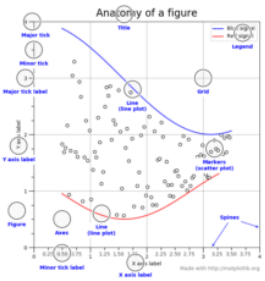


histogram_demo_cumulative

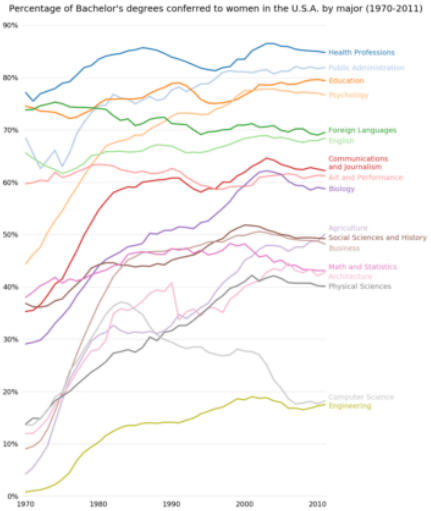


histogram_demo_features

Showcase

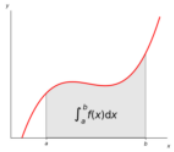


anatomy

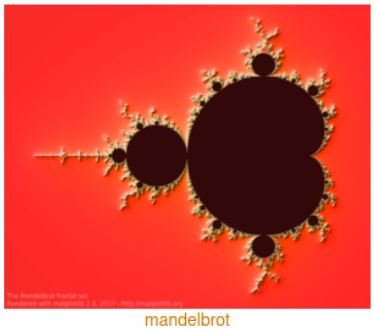


bachelors_degrees_by_gender

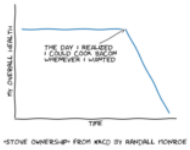
firefox



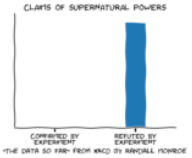
integral_demo



mandelbrot

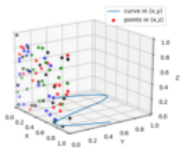


xkcd

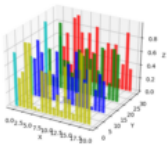


xkcd

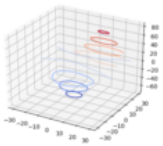
mplot3d toolkit



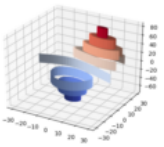
2dcollections3d_demo



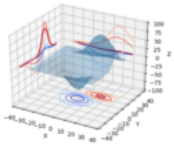
bars3d_demo



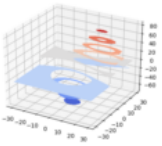
contour3d_demo



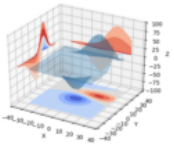
contour3d_demo2



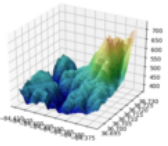
contour3d_demo3



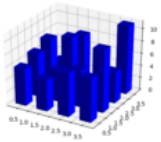
contourf3d_demo



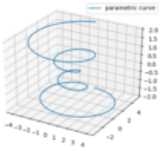
contourf3d_demo2



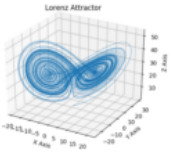
custom_shaded_3d_surface



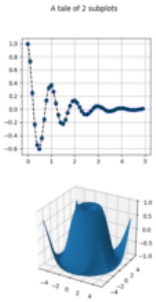
hist3d_demo



lines3d_demo

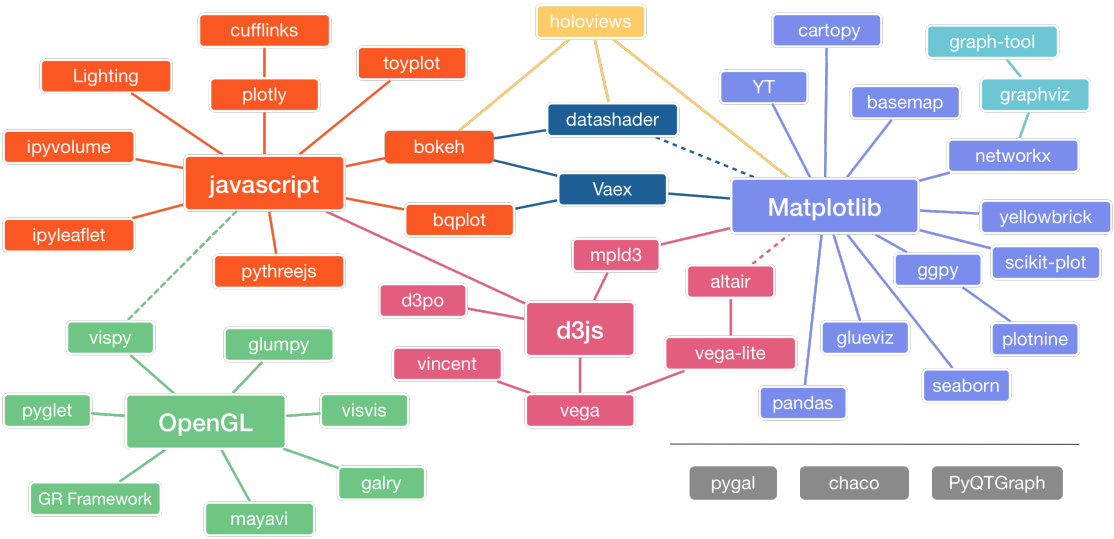


lorenz_attractor



mixed_subplots_demo

4 Plenty of alternatives



4.1 Matplotlib-based

MPL, pandas, seaborn, ggpy, etc.

- use MPL as a stable backend to provide high-level plotting functions
- often designed for specific use-case scenarios
- easier to customize if you already know MPL
- very limited/clunky options for interactivity
- very limited 3d support

4.2 Javascript-based

Bokeh, plotly, pythreejs, etc.

- designed to bring python viz to web-browsers (notebooks)
- interactivity is easier to implement, works smoothly
- interactivity doesn't always scale well with data size
- very limited 3d support

4.3 d3js based

altair, vincent, etc.

- declarative approach to plotting (what to plot, rather than how to plot it)
- alternative (more intuitive?) approach to creating graphs
- relatively new, but being actively developed
- customization options limited compared to MPL

4.4 openGL-based

pyqtgraph, vispy, pyglet, glumpy, ext.

- high-performance packages, meant to integrated into standalone apps
- handle large datasets, can harness CPU/GPU power better
- cannot embed into web-browsers, thus, a bit inconvenient for exploration

4.5 Special mention

- pandas
  - organize data in dataframes
  - supported by most new plotting libraries (also by MPL now)
  - forces you to think about how your data should be organized before you go into plotting it
- holoviews

  - provides high-level plotting functionality to work with multiple backends (MPL, bokeh, datashader) in the browser
  - interactivity enabled by default (zoom, pan, select, etc.)
  - number of customizations possible, depending on backend chosen
  - can handle large datasets and streaming data
- pyqtgraph
  - designed to create or integrate with PyQt applications
  - complete interactivity, with customization only limited by knowledge of Python
  - can handle large datasets and streaming data
- altair
  - declarative grammar is very appealing
  - discussed in depth by Jake VanderPlas (refer to resources)

5 What to use when?

- Quick exploratory data analysis: ~~matplotlib~~, pandas+seaborn, pygal
- sharing results / collaborating:
  - static images: same as above
  - interactive: jupyter lab + (holoviews / altair)
- high performance visualizations:
  - in-browser: holoviews
  - standalone: pyqtgraph
- graphics for publicatins:
  - seaborn + pandas (+ matplotlib)

6 Specific examples

7 Quick setup

[160]:
```
import numpy as np
import pandas as pd
import seaborn as sns; sns.set(style="ticks", color_codes=True)
import matplotlib.pyplot as plt
import holoviews as hv
from holoviews import opts
hv.extension("bokeh")
from sklearn.datasets import load_wine
from IPython.display import Video
```

7.1 Fetch/load data

[116]:
```
# load dataset
wine_dataset = load_wine()
```

[117]:
```
# examine parts of dataset
wine_dataset.keys()
```

[117]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names'])
```

[118]:
```
data = wine_dataset['data']
target = wine_dataset['target']
frame = wine_dataset['frame']
target_names = wine_dataset['target_names']
description = wine_dataset['DESCR']
feature_names = wine_dataset['feature_names']
```

[119]:
```
# features available
print(feature_names)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
```

7.2 Create Pandas dataframe

[120]:
```
df = pd.DataFrame(data, columns=feature_names)
```

[121]:
```
# select columns
attributes = feature_names[:5]
```

[122]:
```
df
```

[122]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	\
0	14.23	1.71	2.43	15.6	127.0	2.80	
1	13.20	1.78	2.14	11.2	100.0	2.65	
2	13.16	2.36	2.67	18.6	101.0	2.80	
3	14.37	1.95	2.50	16.8	113.0	3.85	
4	13.24	2.59	2.87	21.0	118.0	2.80	
..	…	…	…	…	…	…	
173	13.71	5.65	2.45	20.5	95.0	1.68	
174	13.40	3.91	2.48	23.0	102.0	1.80	
175	13.27	4.28	2.26	20.0	120.0	1.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	
177	14.13	4.10	2.74	24.5	96.0	2.05	

	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	\
0	3.06	0.28	2.29	5.64	1.04	
1	2.76	0.26	1.28	4.38	1.05	
2	3.24	0.30	2.81	5.68	1.03	
3	3.49	0.24	2.18	7.80	0.86	
4	2.69	0.39	1.82	4.32	1.04	
..	…	…	…	…	…	
173	0.61	0.52	1.06	7.70	0.64	
174	0.75	0.43	1.41	7.30	0.70	

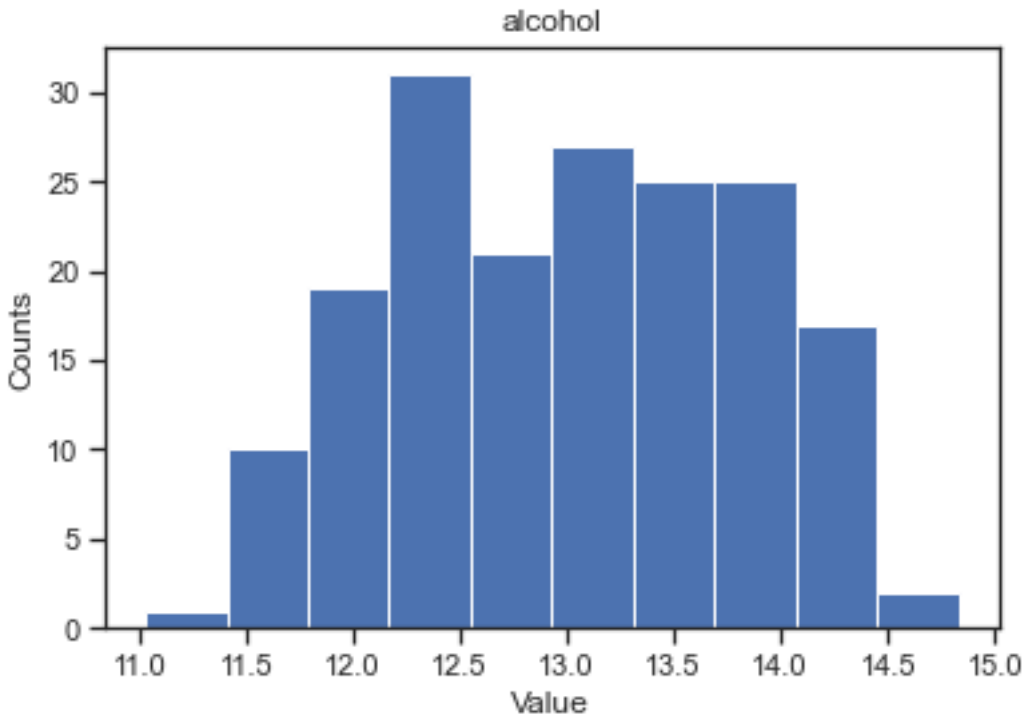
```
175        0.69                  0.43            1.35            10.20  0.59
176        0.68                  0.53            1.46             9.30  0.60
177        0.76                  0.56            1.35             9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            …       …
173                          1.74    740.0
174                          1.56    750.0
175                          1.56    835.0
176                          1.62    840.0
177                          1.60    560.0

[178 rows x 13 columns]
```

[123]:
```
attributes
```

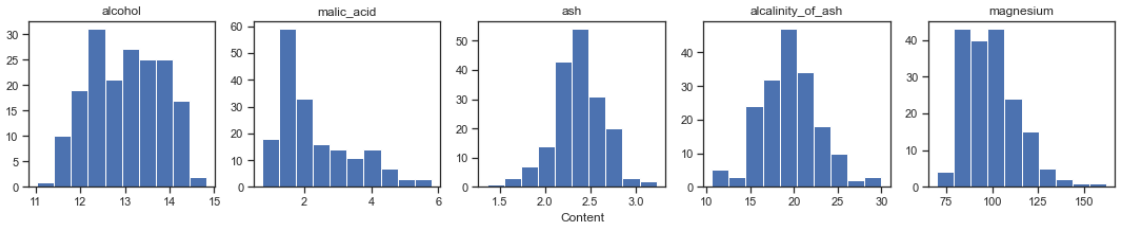
[123]:
```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium']
```

8 Matplotlib

[124]:
```
fig, axis = plt.subplots()
axis.hist(df[feature_names[0]])
axis.set_title(feature_names[0])
axis.set_xlabel("Value")
axis.set_ylabel("Counts")
plt.show()
```

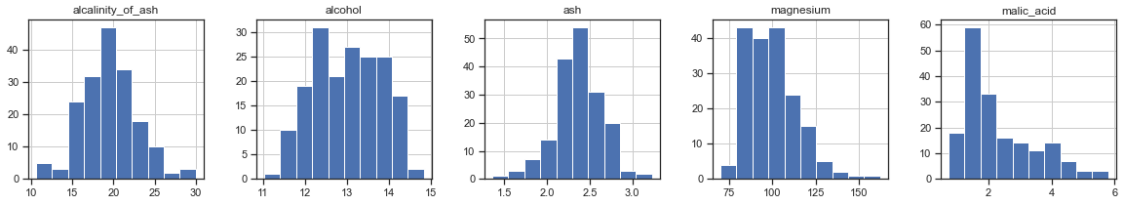


[125]:
```
fig, axes = plt.subplots(nrows=1, ncols=len(attributes),␣
 ↪figsize=(3*len(attributes), 3))
for i, axis in enumerate(axes):
    axis.hist(df[attributes[i]])
    axis.set_title(attributes[i])

fig.text(x=0.5, y=0.0, s="Content")
plt.tight_layout()
plt.show()
```

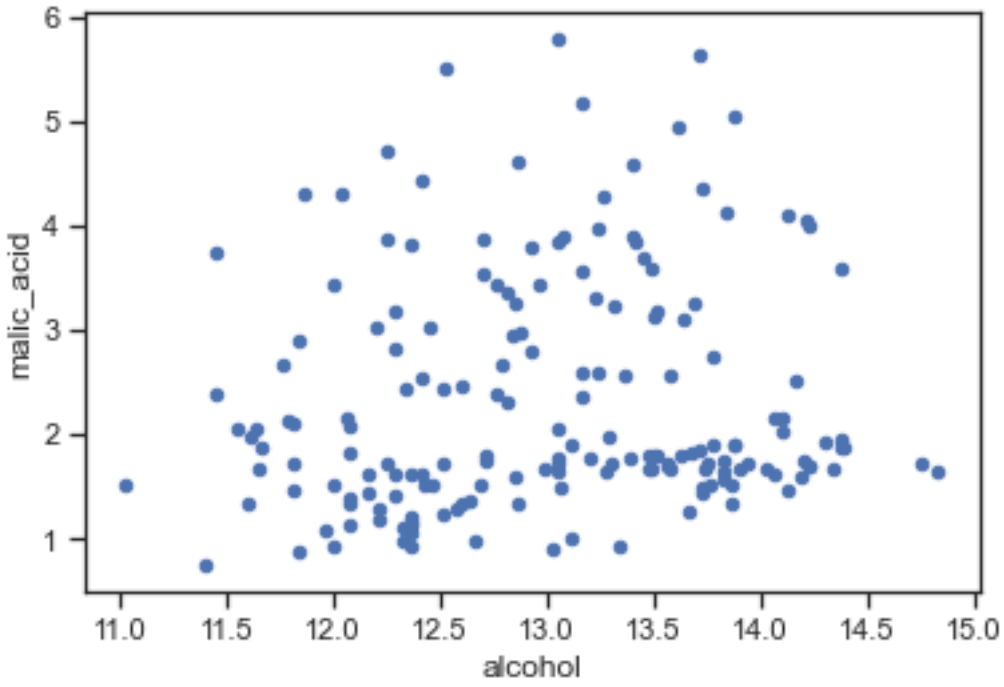


9 Pandas only

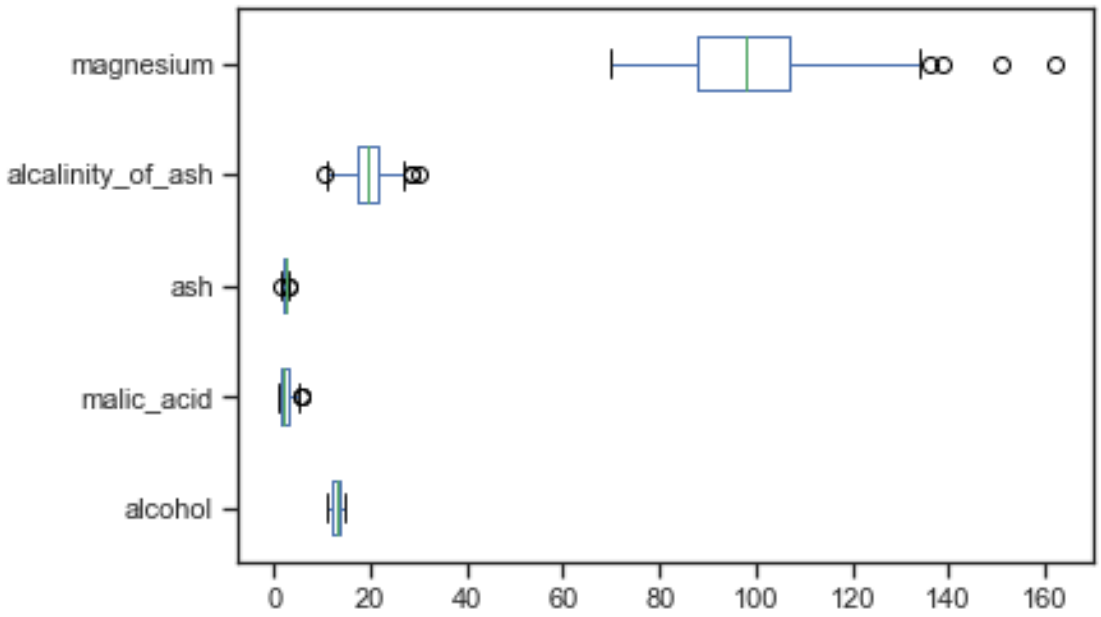
[126]:
```
# histograms
df[attributes].hist(figsize=(20, 3), layout=(1,5))
plt.show()
```



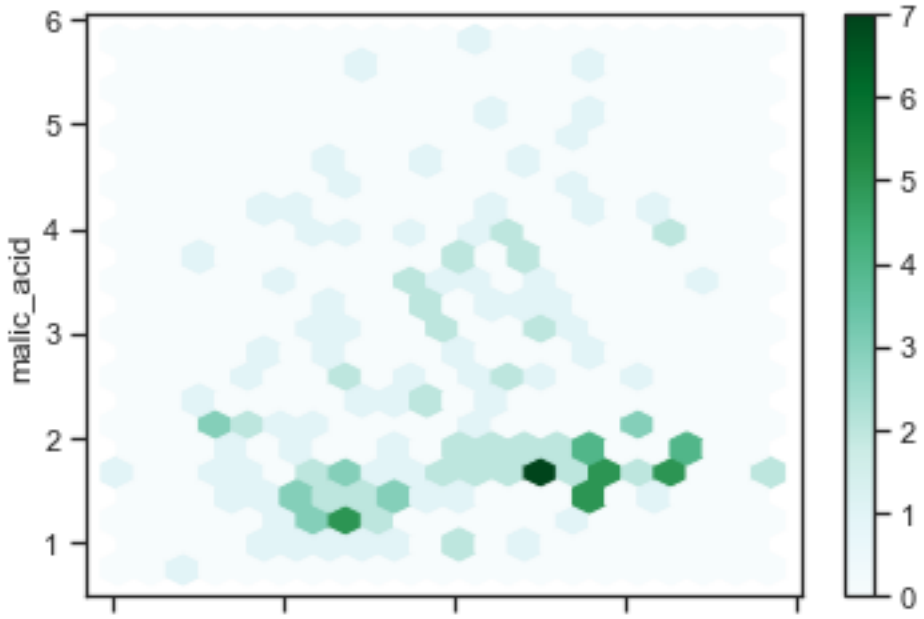
[127]:
```
# scatter plot
df[attributes].plot.scatter(x='alcohol', y='malic_acid', c='b')
plt.show()
```



[128]:
```
# line plot
df[attributes].plot.box(vert=False)
plt.show()
```



[129]:
```
# hexbin plot
df[attributes].plot.hexbin(x='alcohol', y='malic_acid', gridsize=20)
plt.show()
```



10 Pandas+Seaborn

10.0.1 Restructure dataframe

[130]:
```
df['sample'] = np.array(['sample_' + str(i) for i in range(len(df.index))])
restructured_df = df.melt(id_vars='sample', value_vars=feature_names,␣
↪var_name="features", value_name="values")
```

[131]:
```
df
```

[131]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	\
0	14.23	1.71	2.43	15.6	127.0	2.80	
1	13.20	1.78	2.14	11.2	100.0	2.65	
2	13.16	2.36	2.67	18.6	101.0	2.80	
3	14.37	1.95	2.50	16.8	113.0	3.85	
4	13.24	2.59	2.87	21.0	118.0	2.80	
..	…	…	…	…	…	…	
173	13.71	5.65	2.45	20.5	95.0	1.68	
174	13.40	3.91	2.48	23.0	102.0	1.80	
175	13.27	4.28	2.26	20.0	120.0	1.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	
177	14.13	4.10	2.74	24.5	96.0	2.05	

	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	\
0	3.06	0.28	2.29	5.64	1.04	
1	2.76	0.26	1.28	4.38	1.05	
2	3.24	0.30	2.81	5.68	1.03	
3	3.49	0.24	2.18	7.80	0.86	
4	2.69	0.39	1.82	4.32	1.04	
..	…	…	…	…	…	
173	0.61	0.52	1.06	7.70	0.64	
174	0.75	0.43	1.41	7.30	0.70	
175	0.69	0.43	1.35	10.20	0.59	
176	0.68	0.53	1.46	9.30	0.60	
177	0.76	0.56	1.35	9.20	0.61	

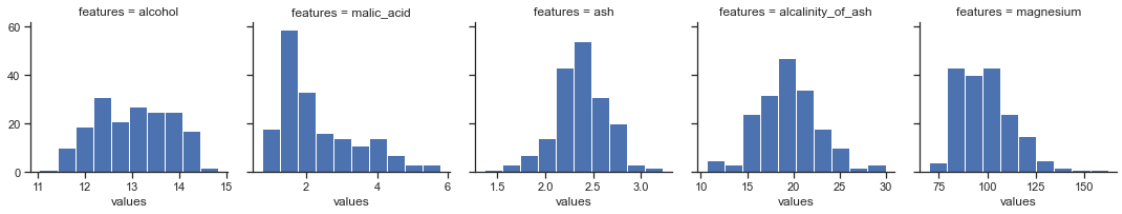
	od280/od315_of_diluted_wines	proline	sample
0	3.92	1065.0	sample_0
1	3.40	1050.0	sample_1
2	3.17	1185.0	sample_2
3	3.45	1480.0	sample_3
4	2.93	735.0	sample_4
..	…	…	…
173	1.74	740.0	sample_173
174	1.56	750.0	sample_174
175	1.56	835.0	sample_175
176	1.62	840.0	sample_176
177	1.60	560.0	sample_177

```
[178 rows x 14 columns]
```

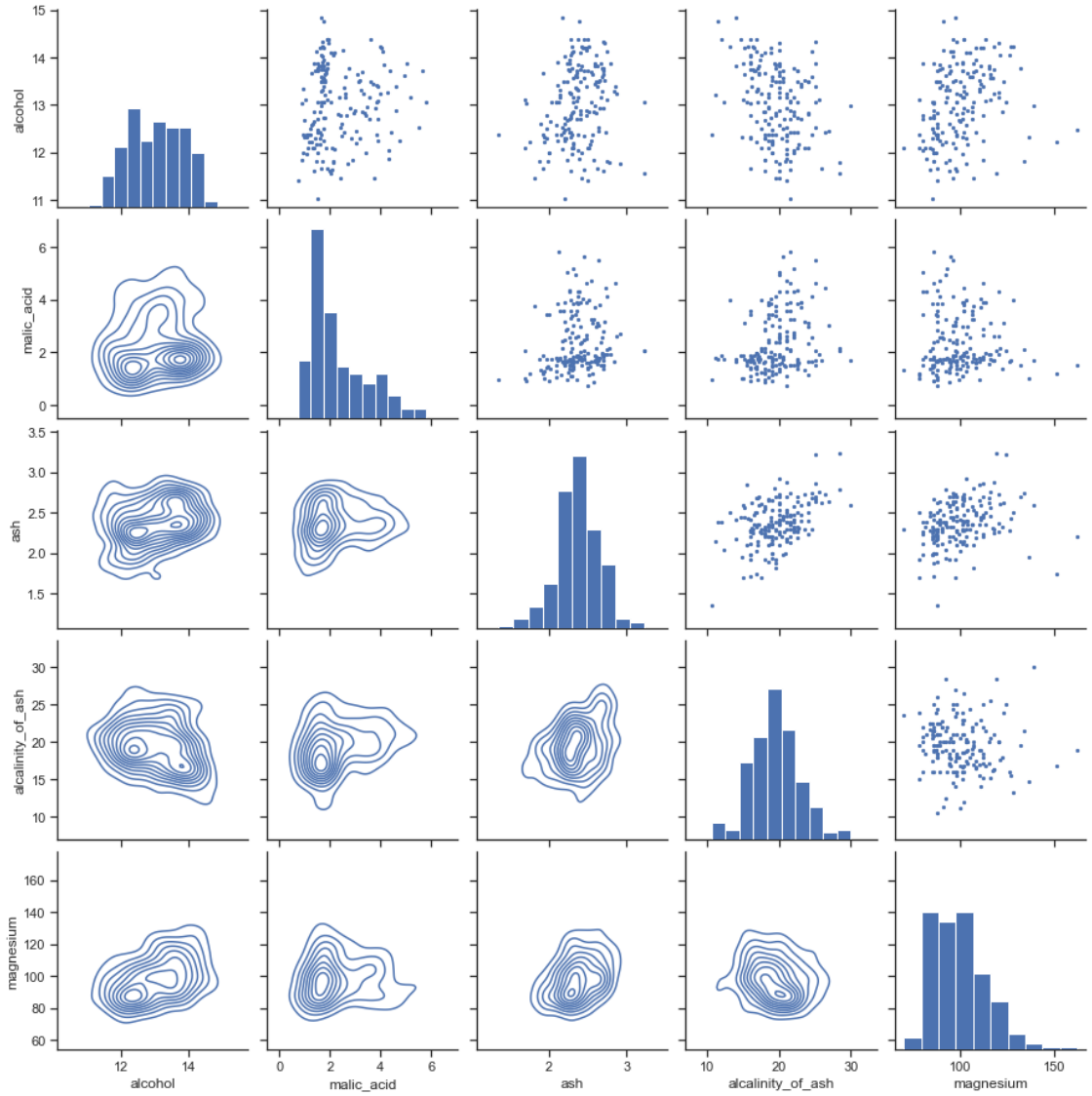
[132]:
```
restructured_df
```

[132]:
```
          sample features  values
0       sample_0  alcohol   14.23
1       sample_1  alcohol   13.20
2       sample_2  alcohol   13.16
3       sample_3  alcohol   14.37
4       sample_4  alcohol   13.24
…            …        …      …
2309  sample_173  proline  740.00
2310  sample_174  proline  750.00
2311  sample_175  proline  835.00
2312  sample_176  proline  840.00
2313  sample_177  proline  560.00

[2314 rows x 3 columns]
```

[133]:
```
grid = sns.FacetGrid(restructured_df[np.isin(restructured_df['features'],␣
↪attributes)], col='features', col_wrap=6, sharex=False)
grid = grid.map(plt.hist, 'values')
```

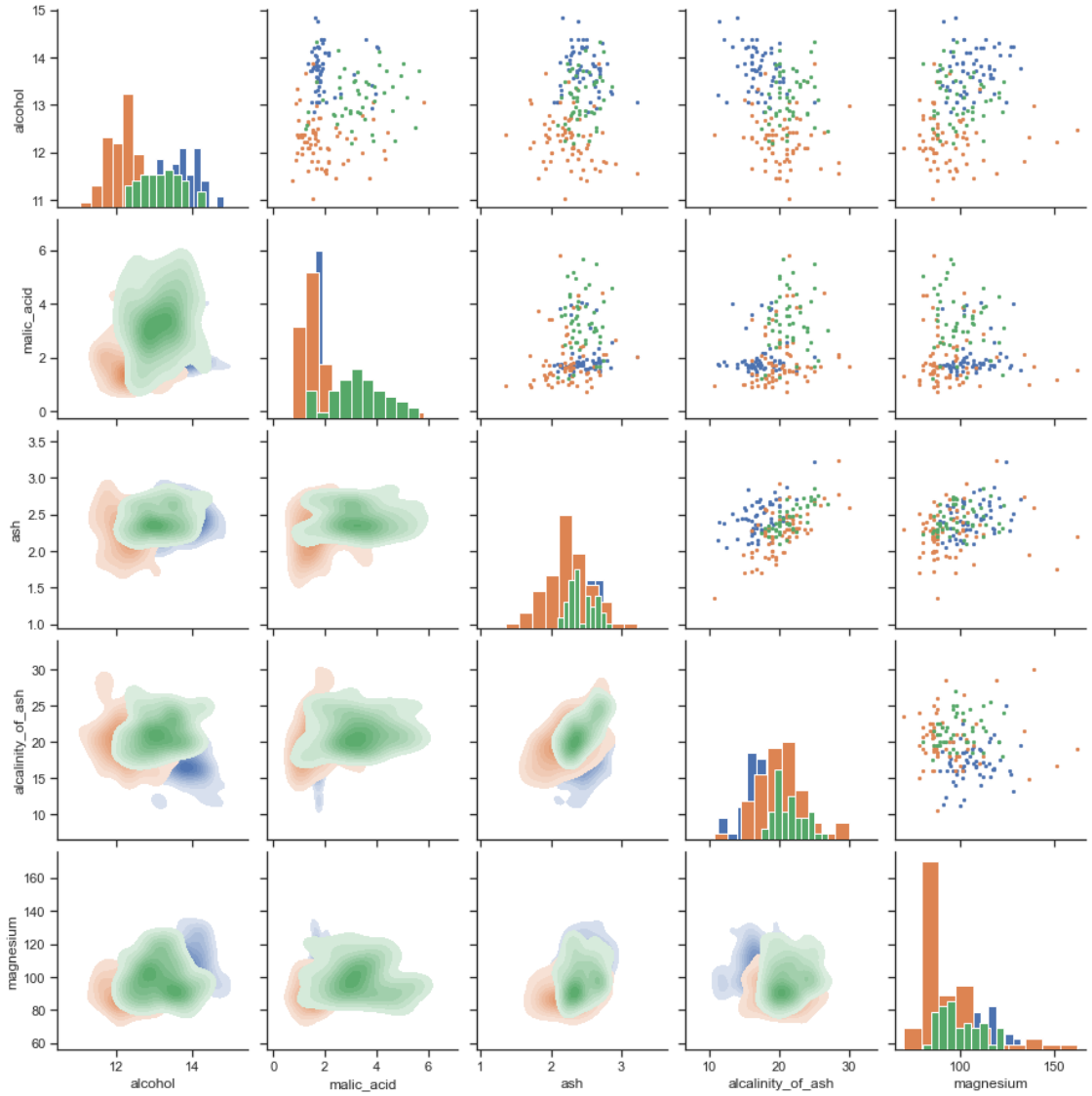


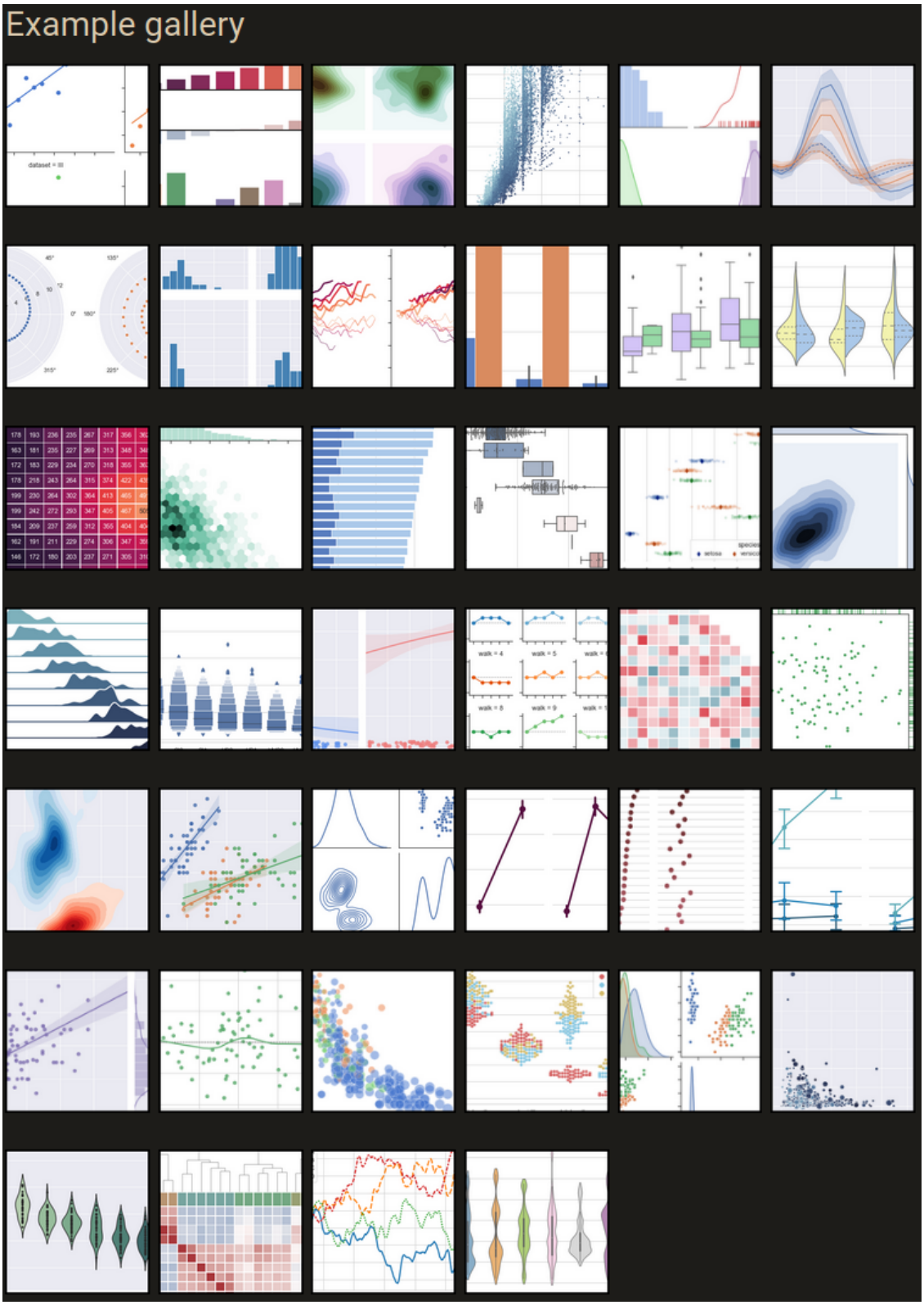
[134]:
```
g = sns.PairGrid(df[attributes])
g = g.map_diag(plt.hist)
g = g.map_upper(plt.scatter, s=5)
g = g.map_lower(sns.kdeplot, colors="C0")
```



[135]:
```
result_df = df.copy()
result_df['target'] = pd.Series(target)
```

[136]:
```
g = sns.PairGrid(result_df[attributes + ['target']], hue='target')
g = g.map_diag(plt.hist)
g = g.map_upper(plt.scatter, s=5)
g = g.map_lower(sns.kdeplot, cumulative=False, shade=True, shade_lowest=False)
```


alcohol
malic_acid
ash
alcalinity_of_ash
magnesium


Example gallery

11 Holoviews

[137]:
```
y, x = np.histogram(df[attributes[0]])
hv.Histogram((x, y))
```

[137]:
```
:Histogram   [x]   (Frequency)
```

11.0.1 Create elements

[138]:
```
curve = hv.Curve((x, y)).opts(color='red')
hist = hv.Histogram(curve)
```

11.0.2 Overlay elements

[139]:
```
hist*curve
```

[139]:
```
:Overlay
   .Histogram.I :Histogram   [x]   (y)
   .Curve.I     :Curve   [x]   (y)
```

11.0.3 Combine elements

[140]:
```
hist+curve
```

[140]:
```
:Layout
   .Histogram.I :Histogram   [x]   (y)
   .Curve.I     :Curve   [x]   (y)
```

[143]:
```
%%opts Layout [shared_axes=False]

hv_plots = []
for attribute in attributes:
    y, x = np.histogram(df[attribute])
    hist = hv.Histogram((x, y), label=attribute)
    hv_plots.append(hist)

hv.Layout(hv_plots).cols(len(attributes))
```

[143]:
```
:Layout
   .Histogram.Alcohol           :Histogram   [x]   (Frequency)
   .Histogram.Malic_acid        :Histogram   [x]   (Frequency)
   .Histogram.Ash               :Histogram   [x]   (Frequency)
   .Histogram.Alcalinity_of_ash :Histogram   [x]   (Frequency)
   .Histogram.Magnesium         :Histogram   [x]   (Frequency)
```

[145]:
```
scatter_points = hv.Points(df[attributes[:2]].to_numpy())
scatter_points.hist(dimension=['x', 'y']).redim.label(x=attributes[0],␣
↪y=attributes[1])
```

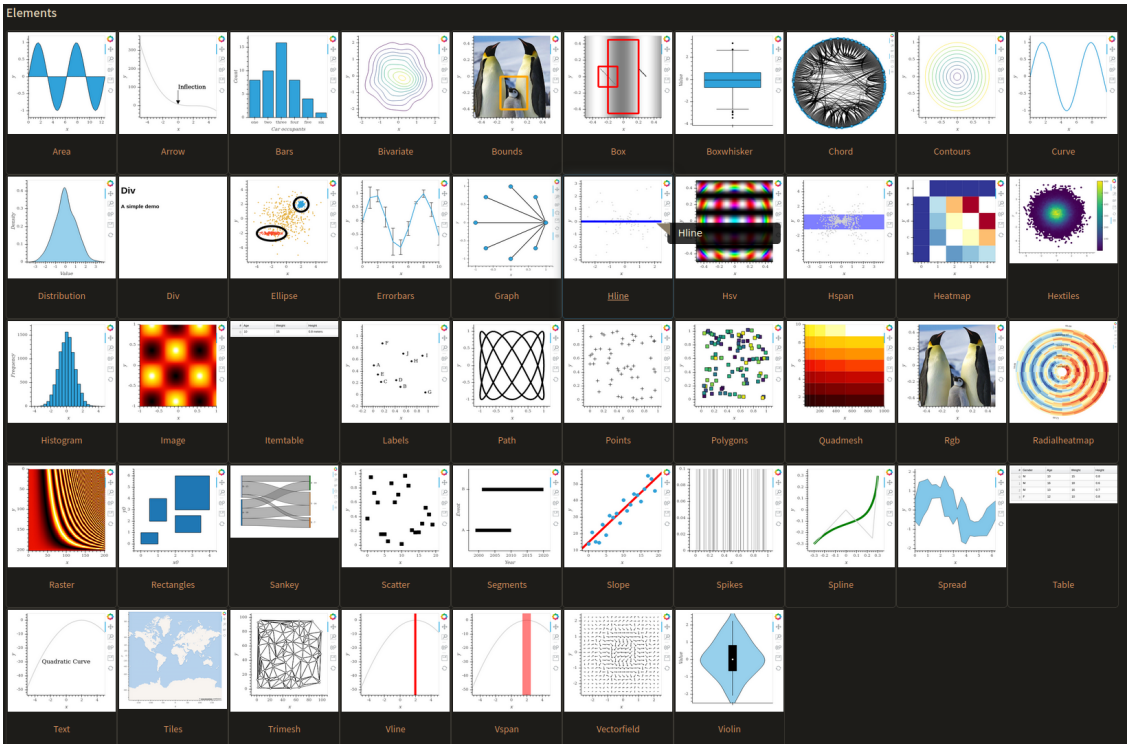
[145]:
```
:AdjointLayout
   :Points    [x,y]
   :Histogram    [y]    (y_frequency)
   :Histogram    [x]    (x_frequency)
```

[151]:
```
from holoviews.operation import gridmatrix

ds = hv.Dataset(df[attributes])
```

[153]:
```
gridmatrix(ds)
```

[153]:
```
:GridMatrix    [X,Y]
   :Histogram    [malic_acid]    (malic_acid_frequency)
```

11.1 Developing library, many more possibilities



12 PyQtGraph

12.0.1 (or pyqtgraph)

```
import sys
import numpy as np
```

```
import pandas as pd
import pyqtgraph as pg
from pyqtgraph.Qt import QtGui, QtCore, QtWidgets
from sklearn.datasets import load_wine


if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    pg_win = pg.GraphicsLayoutWidget()

    dataset = load_wine()
    data = dataset['data']
    features = dataset["feature_names"]

    attributes = features[:5]

    df = pd.DataFrame(data, columns=features)

    plots = []

    for a, attribute in enumerate(attributes):
        y, x = np.histogram(df[attribute])
        plot = pg_win.addPlot(name=attribute, title=attribute)
        plot.plot(x, y, stepMode=True, fillLevel=0, fillOutline=True, brush=(0, 0, 255, 128))
        plot.setLabel("bottom", "Values")
        if a == 0:
            plot.setLabel("left", "Counts")
        plots.append(plot)

    for plot in plots[1:]:
        plot.setYLink(plots[0])

    pg_win.show()
    sys.exit(app.exec_())
```

12.1 Fast and lightweight

```
from pyqtgraph import examples
examples.run()
```

12.2 Embedding into complete applications

[161]:
```
Video("./graphics/swan_1.mp4")
```

[161]:
```
<IPython.core.display.Video object>
```

13 External links and resources

- Jake VanderPlas' talk + slides: https://www.youtube.com/watch?v=FytuB8nFHPQ https://speakerdeck.com/jakevdp/pythons-visualization-landscape-pycon-2017
- Pyviz website: www.pyviz.org
- Python Data Visualization 2018: Why So Many Libraries? https://www.anaconda.com/blog/python-data-visualization-2018-why-so-many-libraries
- A Dramatic Tour through Python's Data Visualization Landscape (including ggplot and Altair) (**this is hilarious**) https://dsaber.com/2016/10/02/a-dramatic-tour-through-pythons-data-visualization-landscape-including-ggplot-and-altair/
- A reddit thread https://www.reddit.com/r/Python/comments/4hdqb6/difference_between_plotting_librar
- Recent Medium blogpost https://medium.com/@lulunana/python-visualization-landscape-3b95ede3d030

14 Thanks for listening!