

Introduction to Seaborn

Setup

```
!pip install seaborn
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

# Read the data
fb = pd.read_csv('fb_stock_prices_2018.csv', index_col='date', parse_dates=True)
quakes = pd.read_csv('earthquakes.csv')
```

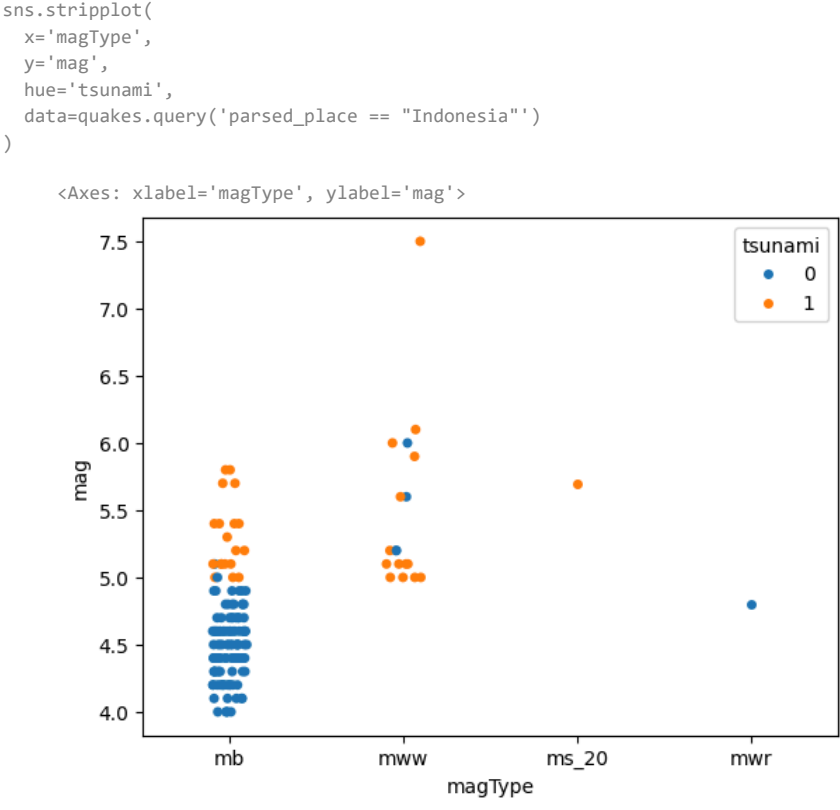
Categorical data

```
quakes.assign(
    time=lambda x: pd.to_datetime(x.time, unit='ns')
).set_index('time').loc['2018-09-28'].query(
    "parsed_place == 'Indonesia' and tsunami == 1 and mag == 7.5"
)
```



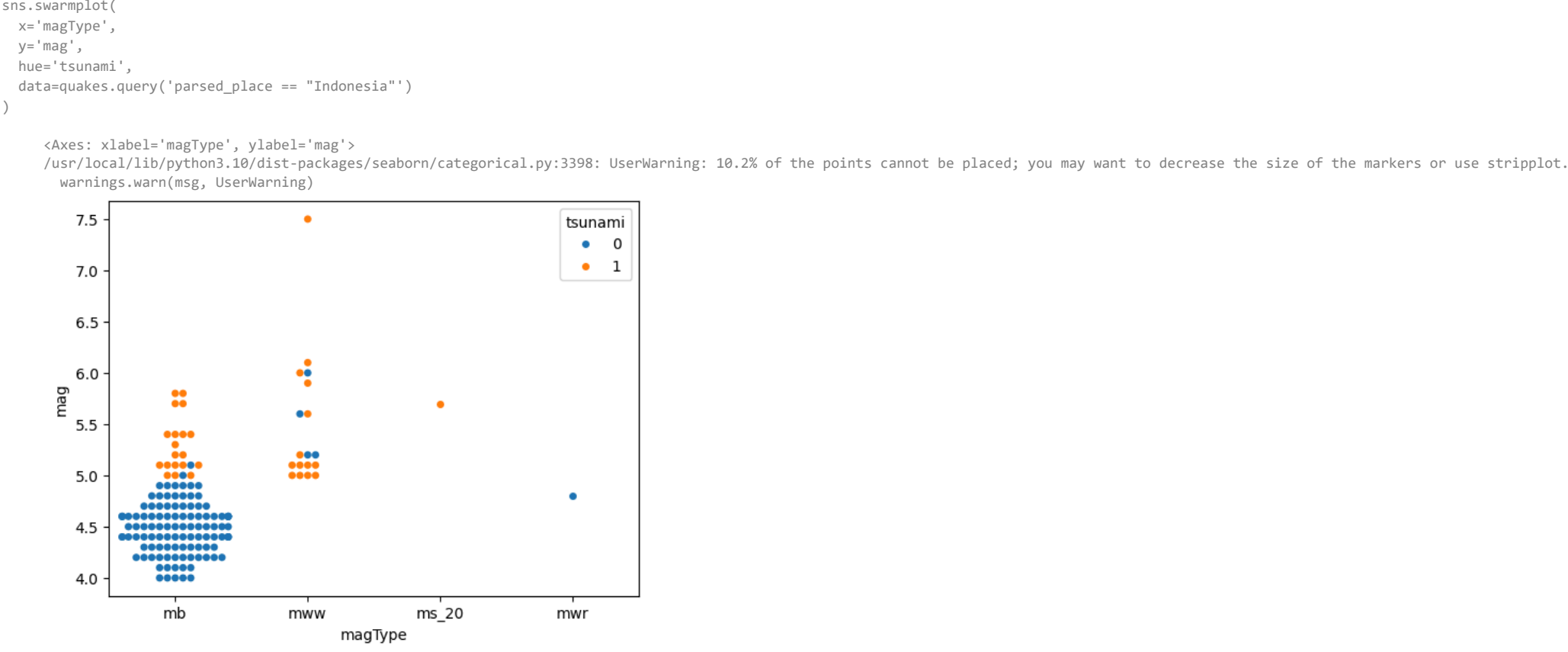
stripplot()

The stripplot() function helps us visualize categorical data on one axis and numerical data on the other. We also now have the option of coloring our points using a column of our data (with the hue parameter). Using a strip plot, we can see points for each earthquake that was measured with a given mag, however, it isn't too easy to see density of the points due to overlap.



swarmplot()

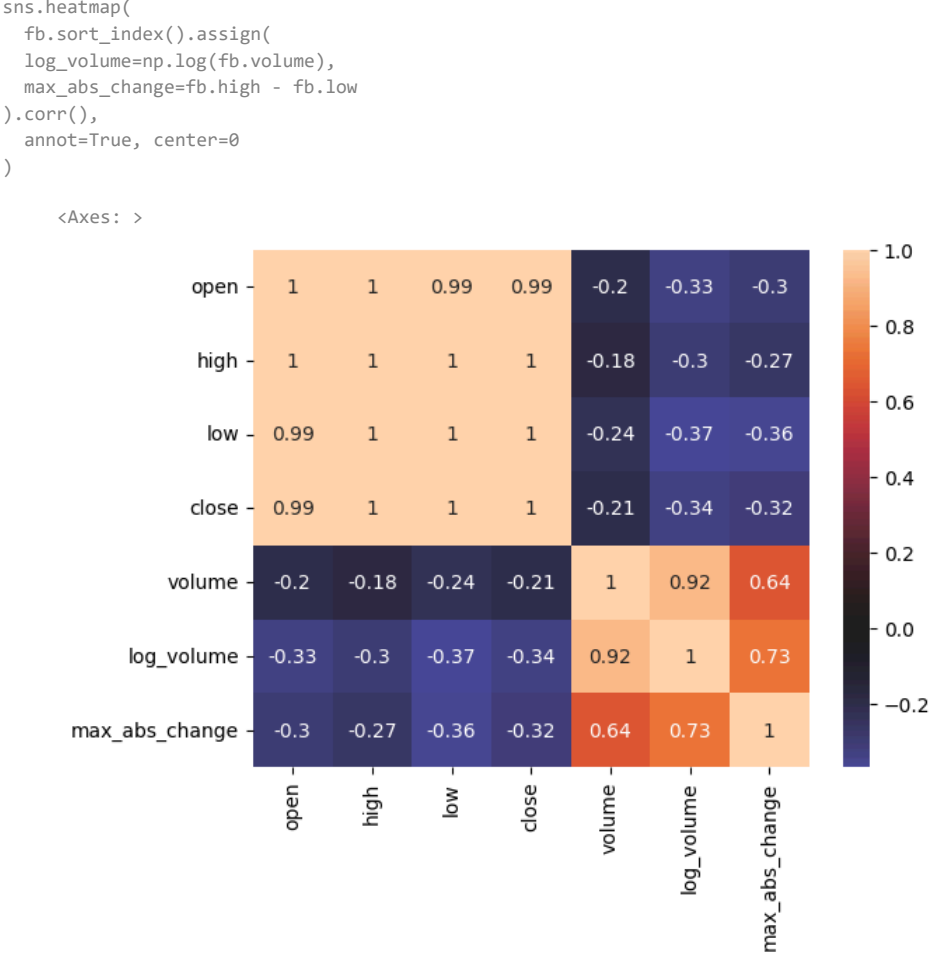
The bee swarm plot helps address this issue by keeping the points from overlapping. Notice how many more points we can see for the blue section of the mb magType.



Correlations and Heatmaps

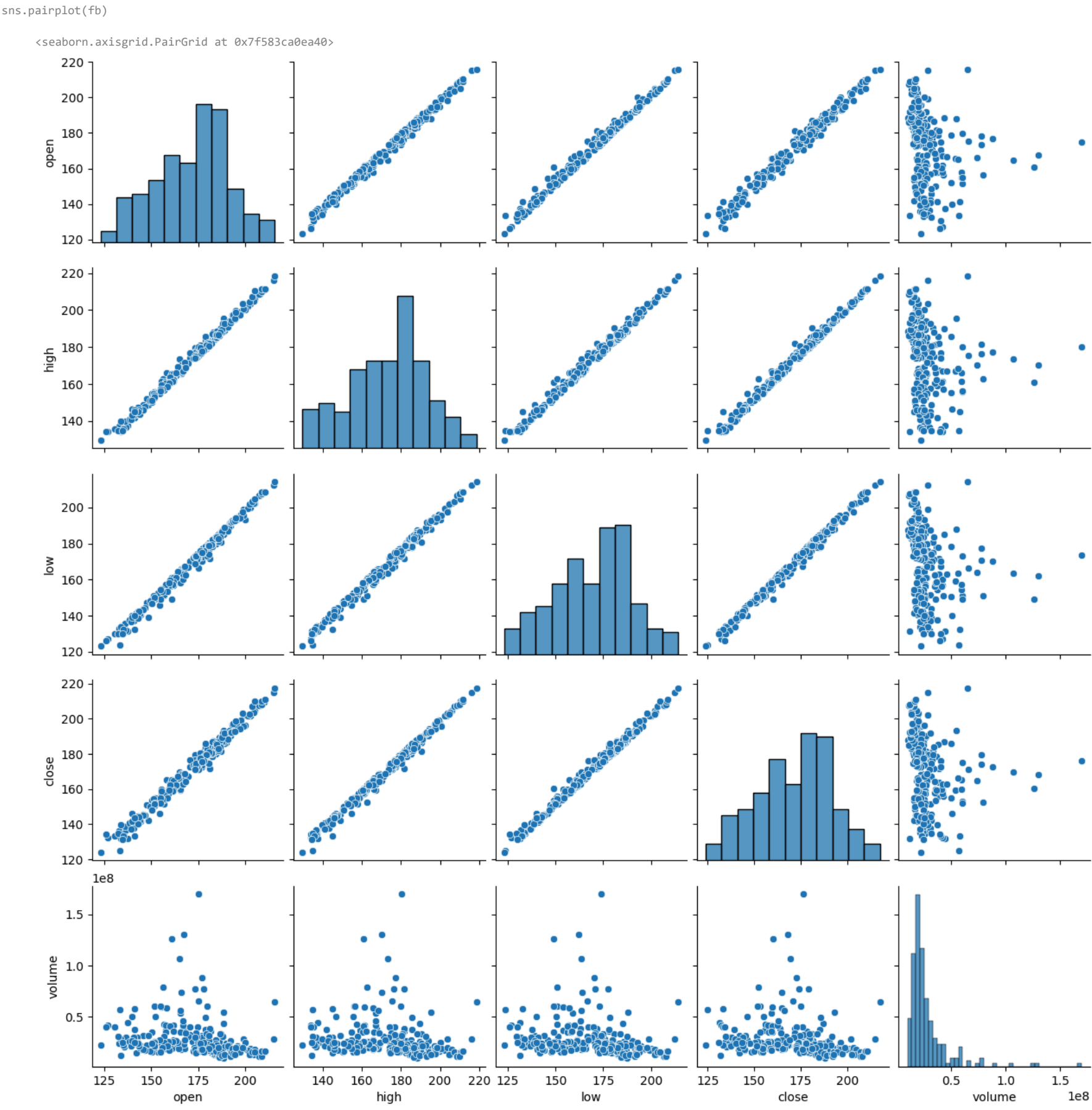
heatmap()

An easier way to create correlation matrix is to use seaborn:

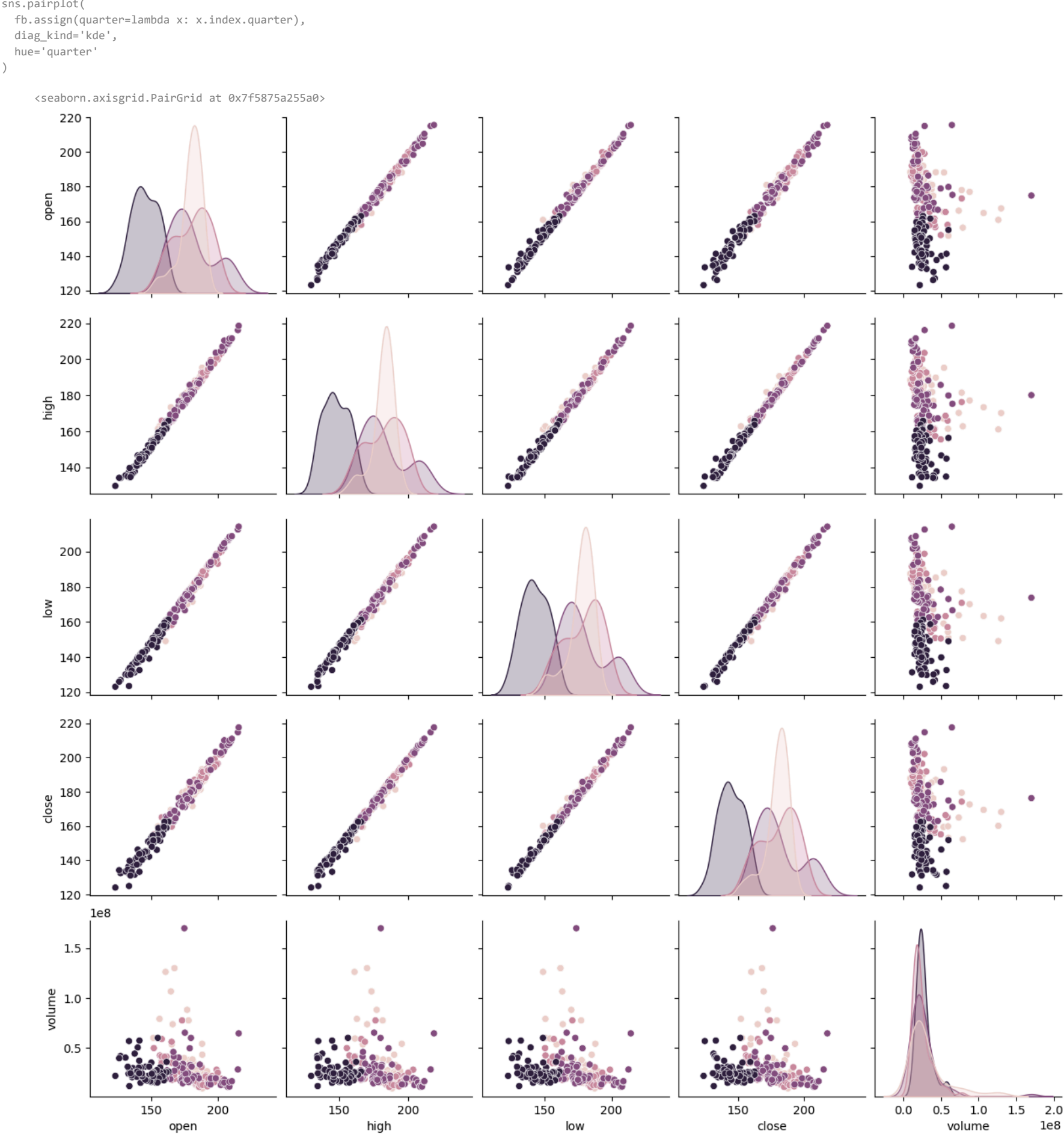


pairplot()

The pair plot is seaborn's answer to the scatter matrix we saw in the pandas subplotting notebook

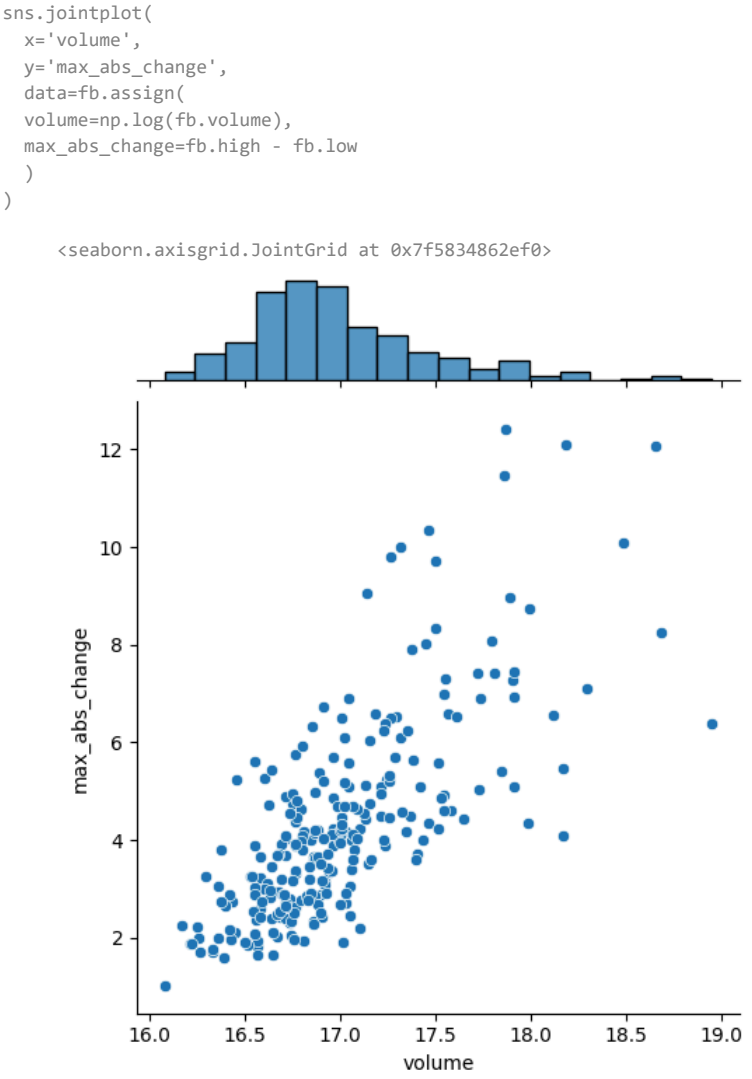


seaborn allows us to change the color or other data with the same shape

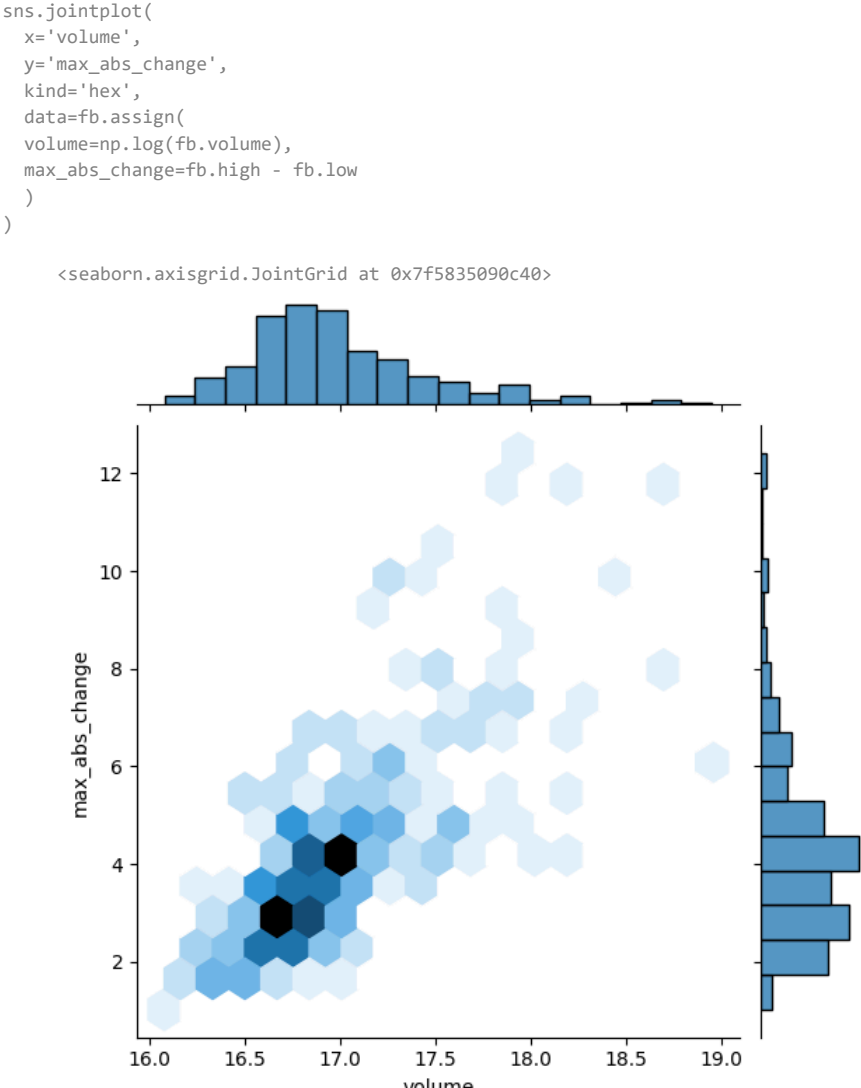


jointplot()

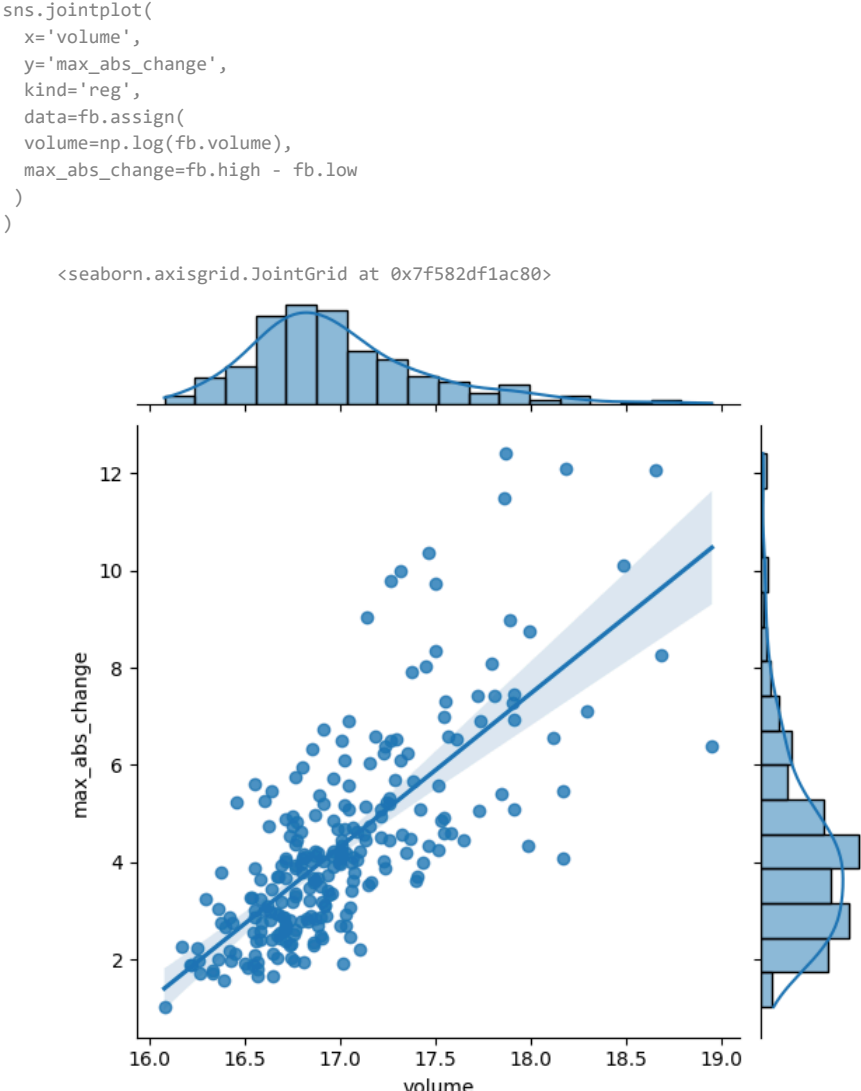
The joint plot allows us to visualize the relationship between two variables, like a scatter plot. However, we get the added benefit of being able to visualize their distributions at the same time (as a histogram or KDE). The default options give us a scatter plot in the center and histograms on the sides:



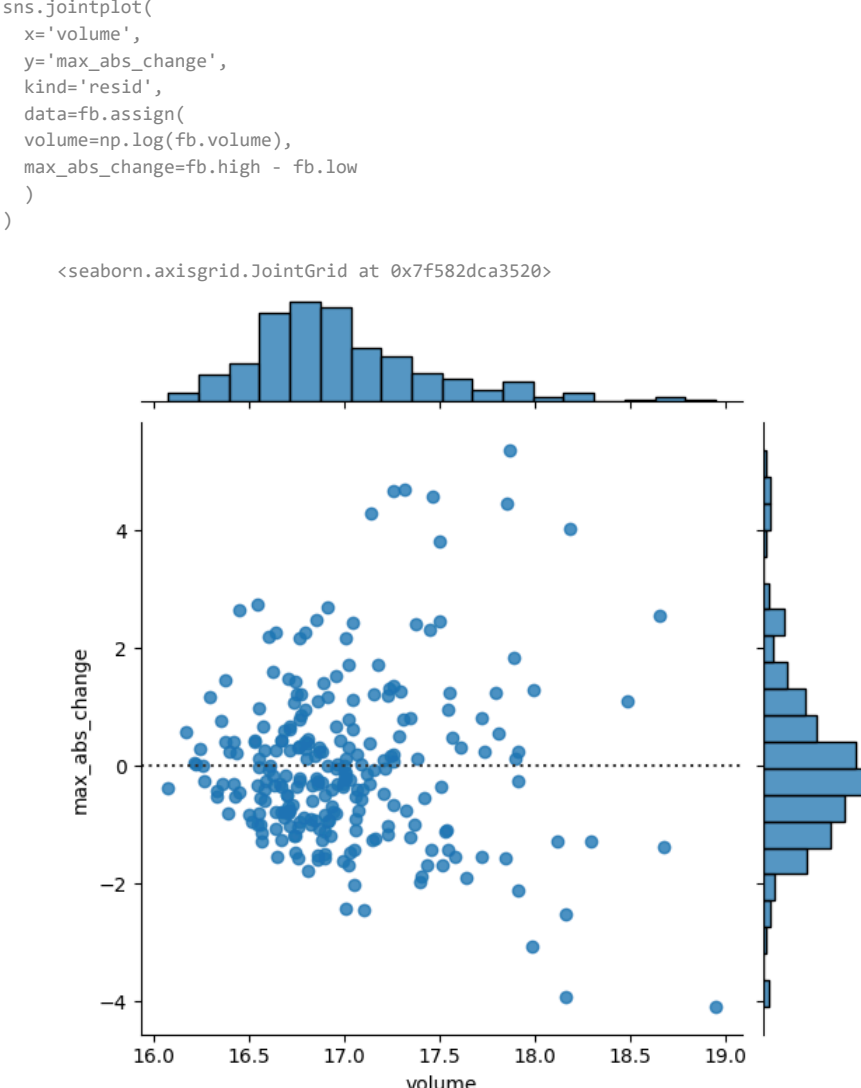
By changing the kind argument, we can change how the center of the plot is displayed. For example, we can pass kind='hex' for hexbins:



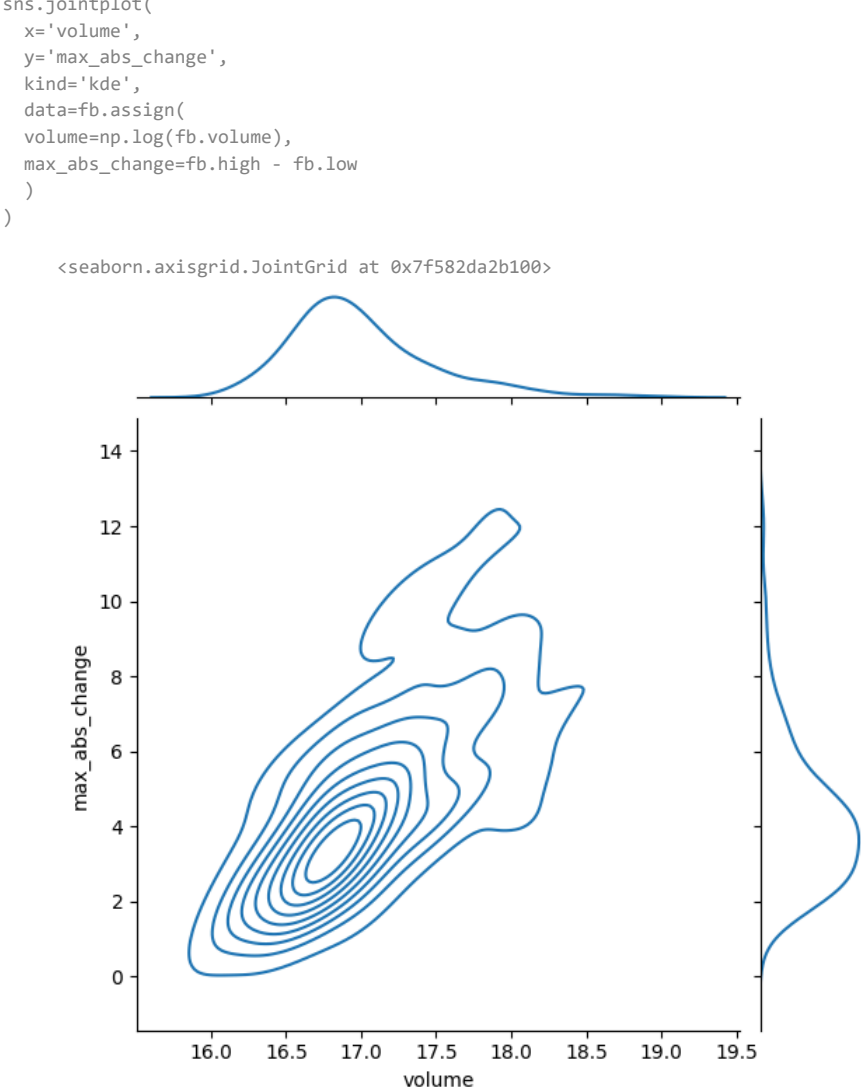
If we specify kind='reg' instead, we get a regression line in the center and KDEs on the sides:



If we pass kind='resid', we get the residuals from the aforementioned regression:



Finally, if we pass kind='kde', we get a contour plot of the joint density estimate with KDEs along the sides:



Regression plots

We are going to use seaborn to visualize a linear regression between the log of the volume traded in Facebook stock and the maximum absolute daily change (daily high stock price - daily low stock price). To do so, we first need to isolate this data:

```
fb_reg_data = fb.assign(
    volume=np.log(fb.volume),
    max_abs_change=fb.high - fb.low
).iloc[:, 2:]
```

Import Itertools & to use permutation and combination

Itertools gives us efficient iterators. Iterators are objects that we loop over, exhausting them. This is an iterator from itertools; notice how the second loop doesn't do anything.

```
iterator = itertools.repeat("I'm an iterator", 1)
for i in iterator:
    print(f"-->{i}")
print("This printed once because the iterator has been exhausted")
for i in iterator:
    print(f"-->{i}")
-->I'm on iterator
This printed once because the iterator has been exhausted
```

```
iterable = list(itertools.repeat("I'm an iterable", 1))
for i in iterable:
    print(f"-->{i}")
print("This prints again because it's an iterable:")
for i in iterable:
    print(f"-->{i}")
-->I'm an iterable
This prints again because it's an iterable:
-->I'm an iterable
```

The reg_resid_plots() function from the reg_resid_plot.py module in this folder uses regplot() and residplot() from seaborn along with itertools to plot the regression and residuals side-by-side

```
from reg_resid_plot import reg_resid_plots
reg_resid_plots(fb_reg_data)

ModuleNotFoundError: Traceback (most recent call last):
  line 1: from reg_resid_plot import reg_resid_plots
    2 reg_resid_plots(fb_reg_data)

ModuleNotFoundError: No module named 'reg_resid_plot'

NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either 'pip' or 'apt'.

To view examples of installing some common dependencies, click the
'Open Examples' button below.
```

We can use Inplot() to split our regression across subsets of our data. For example, we can perform a regression per quarter on the Facebook stock data

```
sns.lmplot(
    x="volume",
    y="max_abs_change",
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low,
        quarter=lambda x: x.index.quarter
    ),
    col="quarter"
)
```