ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

Chương 3: Lập trình Python cho KHDL
- Trực quan hoá dữ liệu với Seaborn

Khoa Công nghệ thông tin
TS. Phạm Công Thắng

Tài liệu tham khảo

- https://seaborn.pydata.org/
- https://pypi.org/project/seaborn/
- https://github.com/mwaskom/seaborn-data
- https://jakevdp.github.io/PythonDataScienceHandbook/04.14-visualization-with-seaborn.html
- Jake VanderPlas, Python Data Science Handbook, O'Reilly Media, Inc., 548 pages, 2016.
- Peter Bruce, Andrew Bruce, Practical Statistics for Data Scientists: 50 Essential Concepts - 1st Edition, O'Reilly Media; 1st edition, 90 pages, 2017
- Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists 1st Edition, 392 pages, 2016
- Lillian Pierson, Data Science For Dummies - 2nd Edition, John Wiley & Sons Inc., 385 pages, 2017.
- David R. Anderson, Dennis J. Sweeney, Thomas A. Williams, Statistics for Business and Economics, South-Western College Pub., 880 pages, 2010.
- Các nguồn internet khác.

Seaborn

- Seaborn là một thư viện để tạo đồ họa thống kê bằng Python.
  - Nó được xây dựng trên matplotlib và tích hợp chặt chẽ với cấu trúc dữ liệu pandas.
    - Một API hướng tập dữ liệu để kiểm tra mối quan hệ giữa nhiều biến
    - Hỗ trợ một cách chuyên biệt cho việc sử dụng các biến phân loại để hiển thị các quan sát hoặc thống kê tổng hợp
    - Các tùy chọn để trực quan hóa các phân phối đơn biến hoặc song biến và so sánh chúng giữa các tập con dữ liệu
    - Tự động ước tính và vẽ biểu đồ của các mô hình hồi quy tuyến tính cho các loại biến phụ thuộc khác nhau
    - Tính trừu tượng cao cho các cấu trúc lưới nhiều ô cho phép dễ dàng xây dựng các hình trực quan phức tạp
    - Kiểm soát ngắn gọn kiểu dáng hình matplotlib với một số chủ đề cài sẵn
    - Các công cụ để chọn bảng màu hiển thị trung thực các mẫu trong dữ liệu

Seaborn

- Trực quan hóa trở thành một phần trung tâm của việc khám phá và hiểu dữ liệu.
- Các chức năng vẽ đồ thị theo hướng tập dữ liệu của nó hoạt động trên khung dữ liệu và mảng chứa toàn bộ tập dữ liệu.
- Thực hiện việc lập bản đồ ngữ nghĩa và tổng hợp thống kê cần thiết để tạo ra các biểu đồ mang thông tin.

Seaborn

- Phân tích thống kê là một quá trình tìm hiểu các biến trong tập dữ liệu liên quan với nhau như thế nào và các mối quan hệ đó phụ thuộc vào các biến khác như thế nào.
- Trực quan có thể là một thành phần cốt lõi của quá trình này bởi vì, khi dữ liệu được trực quan hóa đúng cách, hệ thống trực quan của con người có thể nhìn thấy các xu hướng và mẫu biểu thị mối quan hệ.

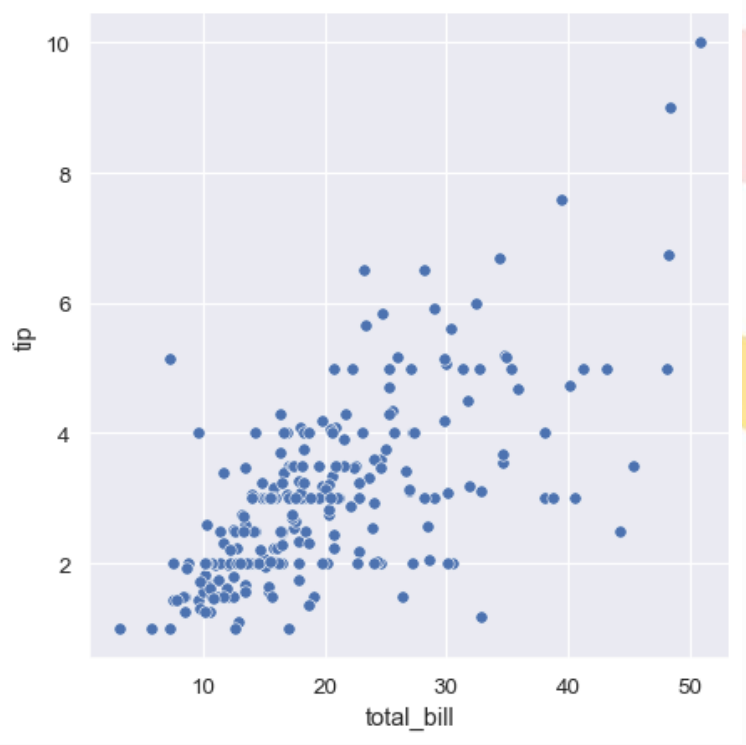
Seaborn

- Các chức năng của seaborn
  - Hàm relplot ().
  - Đây là một hàm figure-level function để trực quan hóa các mối quan hệ thống kê bằng cách sử dụng hai cách tiếp cận phổ biến: biểu đồ phân tán scatter plot và line plots.
    - scatterplot() (with kind="scatter"; the default)
    - lineplot() (with kind="line")
  - Các hàm này có thể khá rõ rang vì chúng sử dụng các biểu diễn dữ liệu đơn giản và dễ hiểu, tuy nhiên cũng có thể biểu diễn các cấu trúc tập dữ liệu phức tạp.
  - Việc vẽ đồ thị hai chiều có thể được nâng cao bằng cách ánh xạ tối đa ba biến bổ sung bằng cách sử dụng ngữ nghĩa của màu sắc, kích thước và kiểu dáng.

Seaborn

- Relating variables with scatter plots:

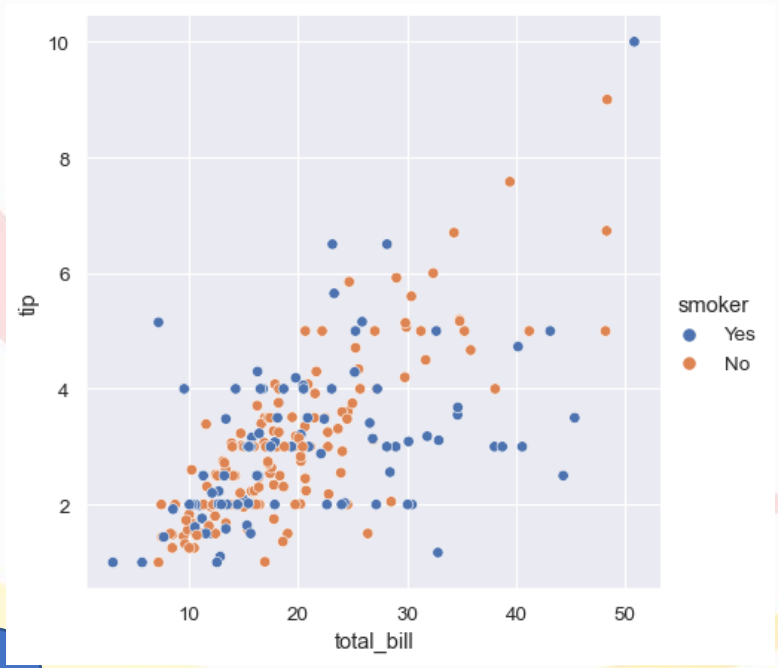
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme(style="darkgrid")

tips = sns.load_dataset("tips")
sns.relplot(x="total_bill", y="tip", data=tips);
```



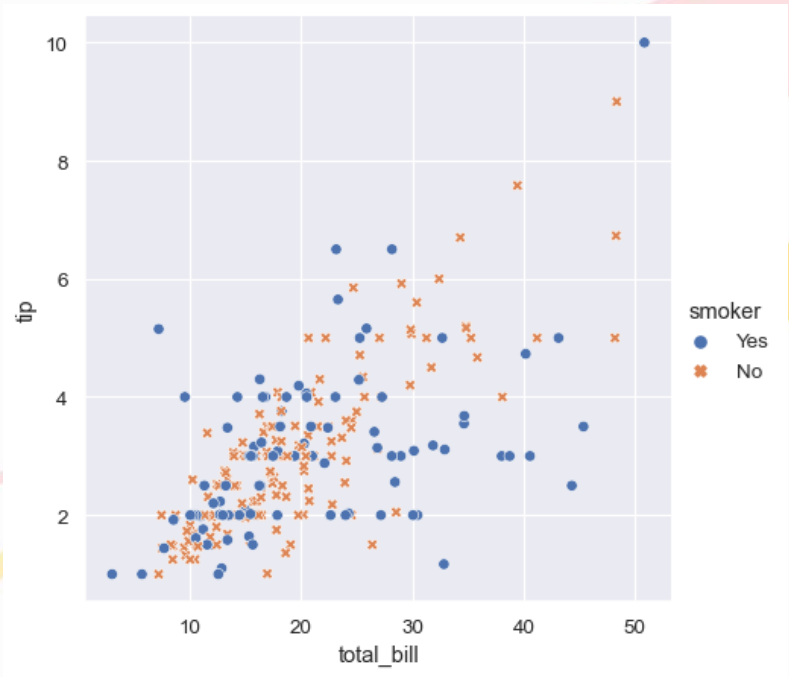
Seaborn

- Khi các điểm được vẽ theo hai chiều, một chiều khác có thể được thêm vào biểu đồ bằng cách tô màu các điểm theo một biến thứ ba.
  - Trong seaborn, điều này được gọi là sử dụng "ngữ nghĩa màu", bởi vì màu sắc của điểm mang ý nghĩa:

sns.relplot(x="total_bill", y="tip", hue="smoker", data=tips);



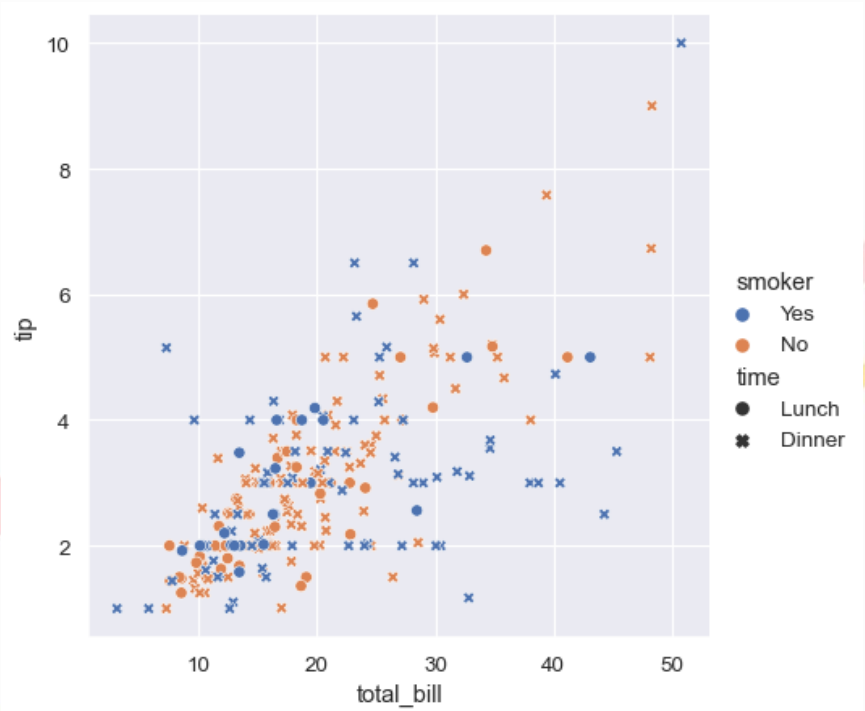
sns.relplot(x="total_bill", y="tip", hue="smoker", style="smoker", data=tips);



Seaborn

- Cũng có thể biểu diễn bốn biến bằng cách thay đổi màu sắc và kiểu dáng của từng điểm một cách độc lập.
  - Điều này nên được thực hiện cẩn thận, vì mắt ít nhạy cảm với hình dạng hơn nhiều so với màu sắc:

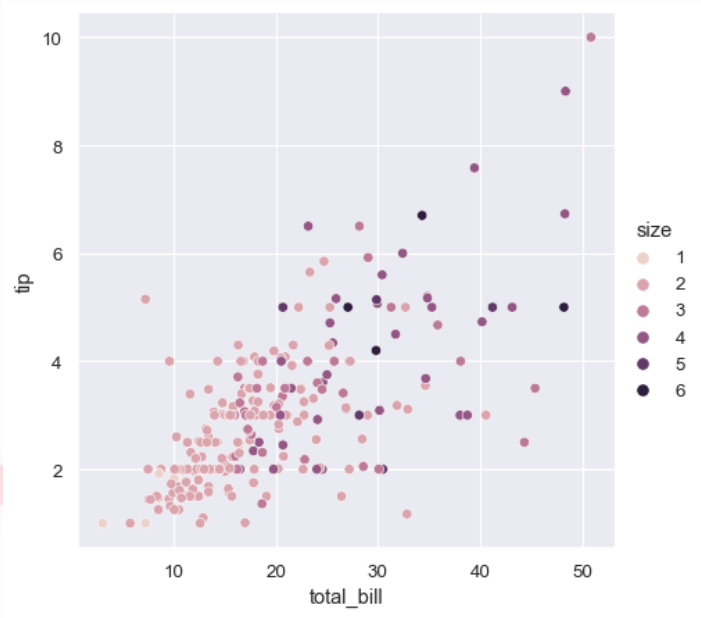
```
sns.relplot(x="total_bill", y="tip", hue="smoker", style="time", data=tips);
```



Seaborn

- Ngữ nghĩa màu sắc được phân loại, vì vậy bảng màu định tính mặc định được áp dụng.
  - Nếu ngữ nghĩa của màu sắc là số (cụ thể là nếu nó có thể được ép để nổi), màu mặc định sẽ chuyển sang bảng tuần tự:

```
sns.relplot(x="total_bill", y="tip", hue="size", data=tips);
```



Seaborn

- Có nhiều lựa chọn để thực hiện
  - Sử dụng giao diện chuỗi thành cubehelix_palette ():

sns.palplot(sns.cubehelix_palette())

sns.palplot(sns.cubehelix_palette(rot=-.4))

sns.palplot(sns.cubehelix_palette(start=2.8, rot=.1))

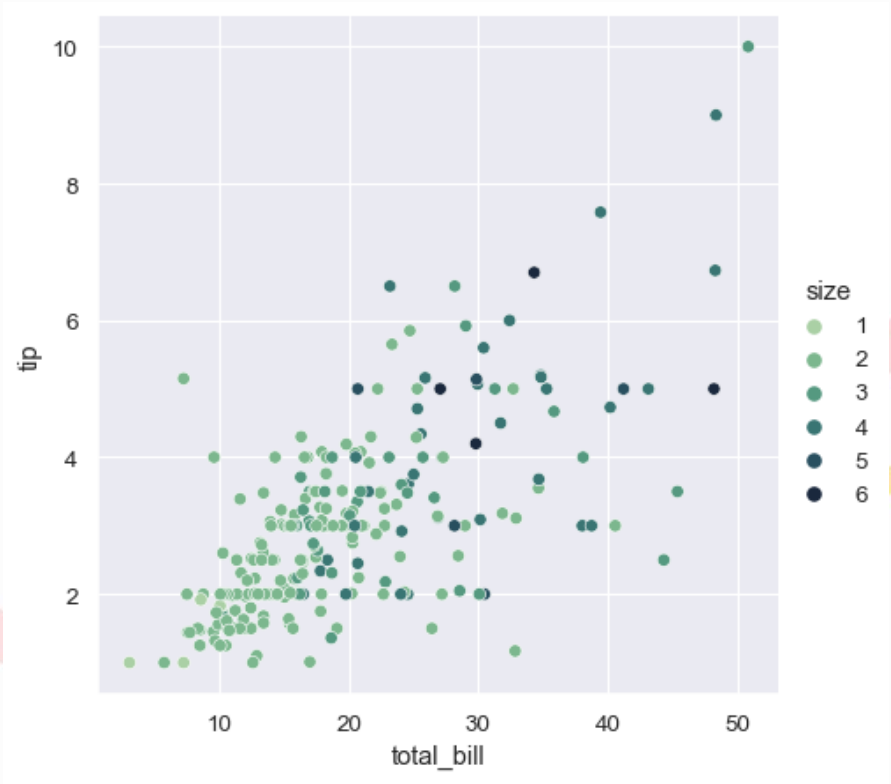
sns.palplot(sns.cubehelix_palette(reverse=True))

sns.palplot(sns.color_palette("ch:2,r=.2,l=.6"))

Seaborn

- Sử dụng giao diện chuỗi thành cubehelix_palette ():

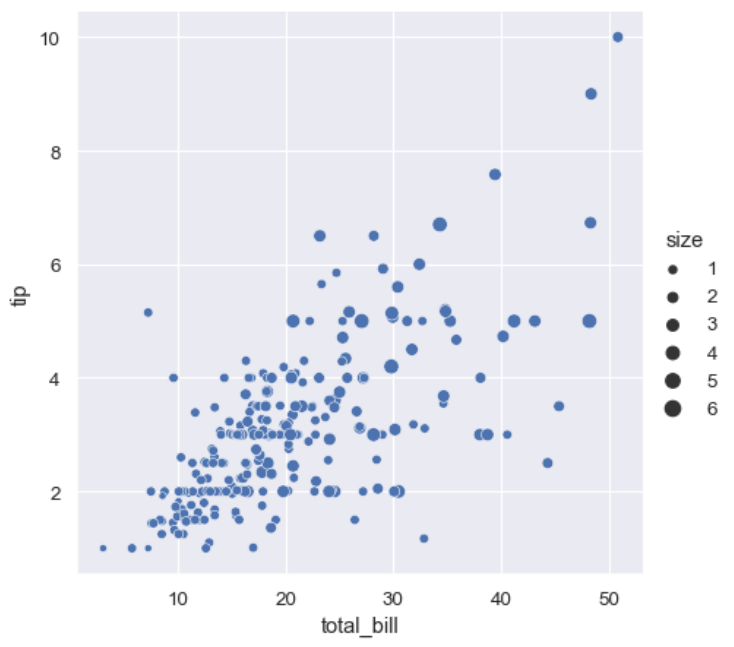
```
sns.relplot(x="total_bill", y="tip", hue="size", palette="ch:r=-.5,l=.75",
data=tips);
```



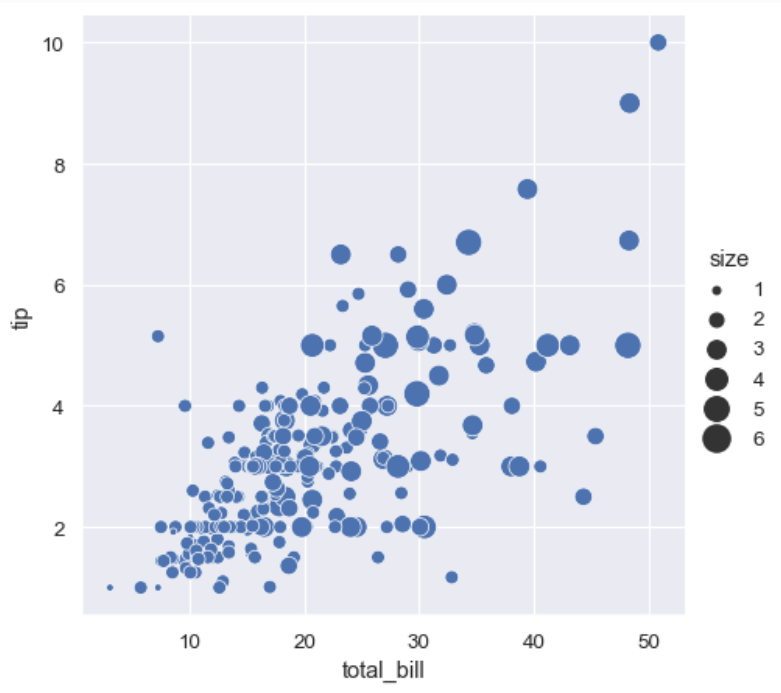
Seaborn

- Loại biến ngữ nghĩa tiếp theo là thay đổi kích thước của mỗi điểm:

```
sns.relplot(x="total_bill", y="tip", size="size", data=tips);
```




```
sns.relplot(x="total_bill", y="tip", size="size", sizes=(15, 200), data=tips);
```



Seaborn

- Biểu đồ phân tán có hiệu quả cao, nhưng không có loại hình trực quan nào tối ưu nhất.
  - Thay vào đó, phần trình bày trực quan nên được điều chỉnh cho phù hợp với các chi tiết cụ thể của tập dữ liệu và cho yêu cầu đang cố gắng trả lời với sơ đồ.
  - Với một số bộ dữ liệu, chúng ta có thể muốn hiểu những thay đổi trong một biến dưới dạng hàm thời gian hoặc một biến liên tục tương tự.
    - Trong tình huống này, một lựa chọn tốt là vẽ một biểu đồ đường thẳng.

      thực hiện bằng hàm lineplot () trực tiếp

      hoặc với relplot () bằng cách thiết lập kind = "line":

Seaborn

- Emphasizing continuity with line plots

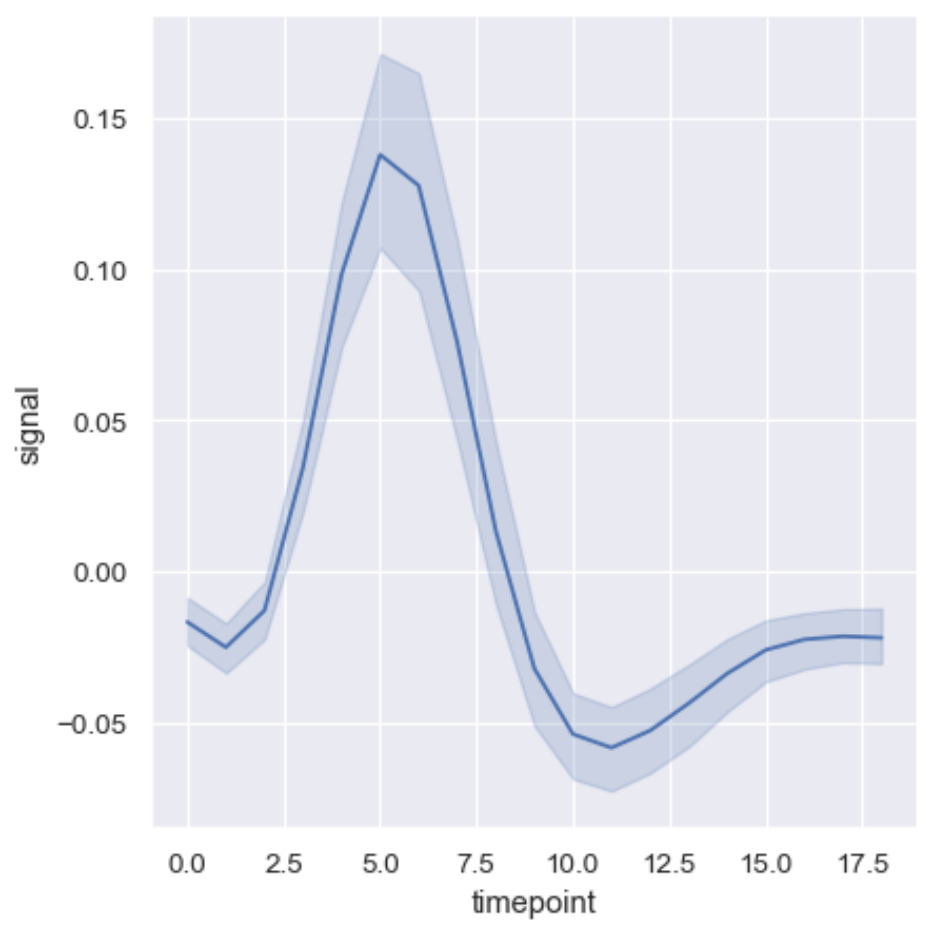
```
df = pd.DataFrame(dict(time=np.arange(500), value=np.random.randn(500).cumsum()))
g = sns.relplot(x="time", y="value", kind="line", data=df) g.fig.autofmt_xdate()
```

Seaborn

- Các bộ dữ liệu phức tạp hơn sẽ có nhiều phép đo cho cùng một giá trị của biến x.
    - Hành vi mặc định trong seaborn là tổng hợp nhiều phép đo ở mỗi giá trị x bằng cách vẽ biểu đồ giá trị trung bình và khoảng tin cậy 95% xung quanh giá trị trung bình:

Seaborn

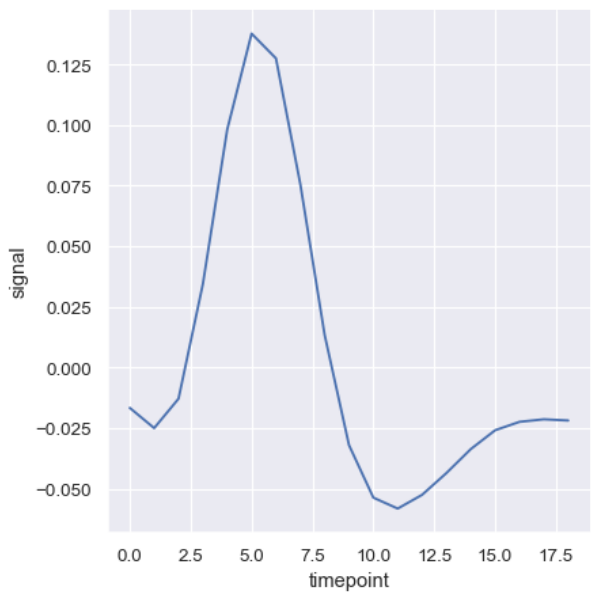
```
fmri = sns.load_dataset("fmri")
sns.relplot(x="timepoint", y="signal", kind="line", data=fmri);
```



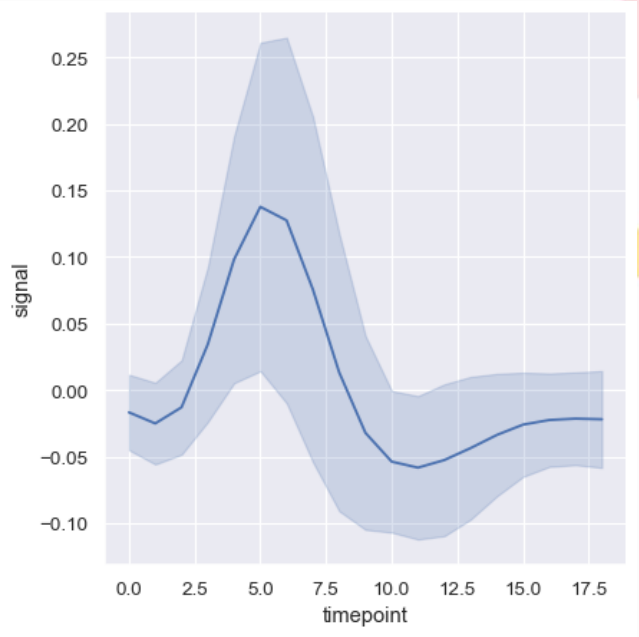
Seaborn

- Khoảng tin cậy được tính bằng cách sử dụng bootstrapping, có thể tốn nhiều thời gian cho các tập dữ liệu lớn hơn.
  - Do đó, có thể vô hiệu hóa chúng:

```
sns.relplot (x = "timepoint", y = "signal", ci = None, kind = "line", data = fmri);
```




```
sns.relplot(x="timepoint", y="signal", kind="line", ci="sd", data=fmri);
```



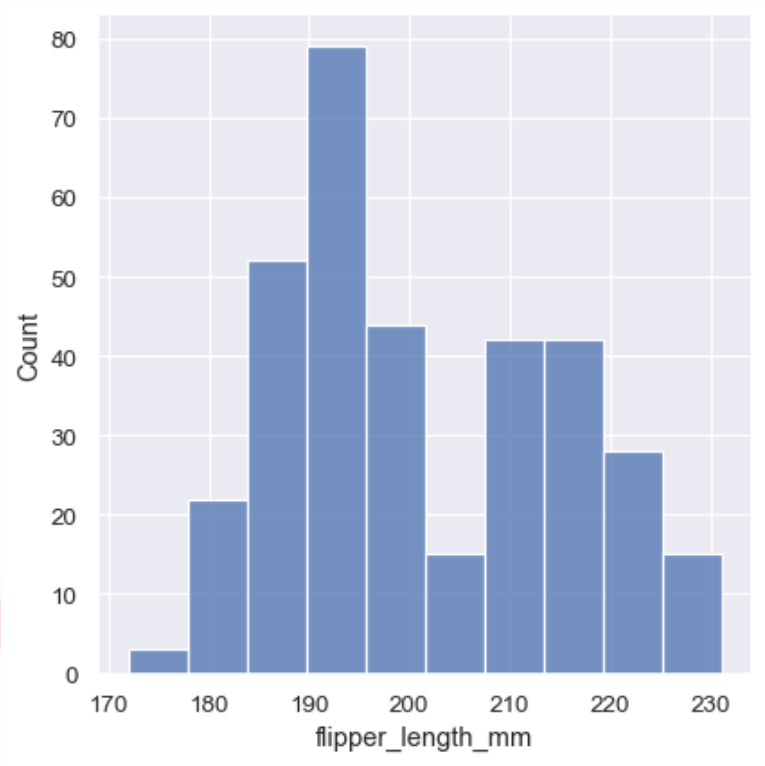
Seaborn

- Một số nội dung về Visualizing statistical relationships
  - https://seaborn.pydata.org/tutorial/relational.html

Seaborn

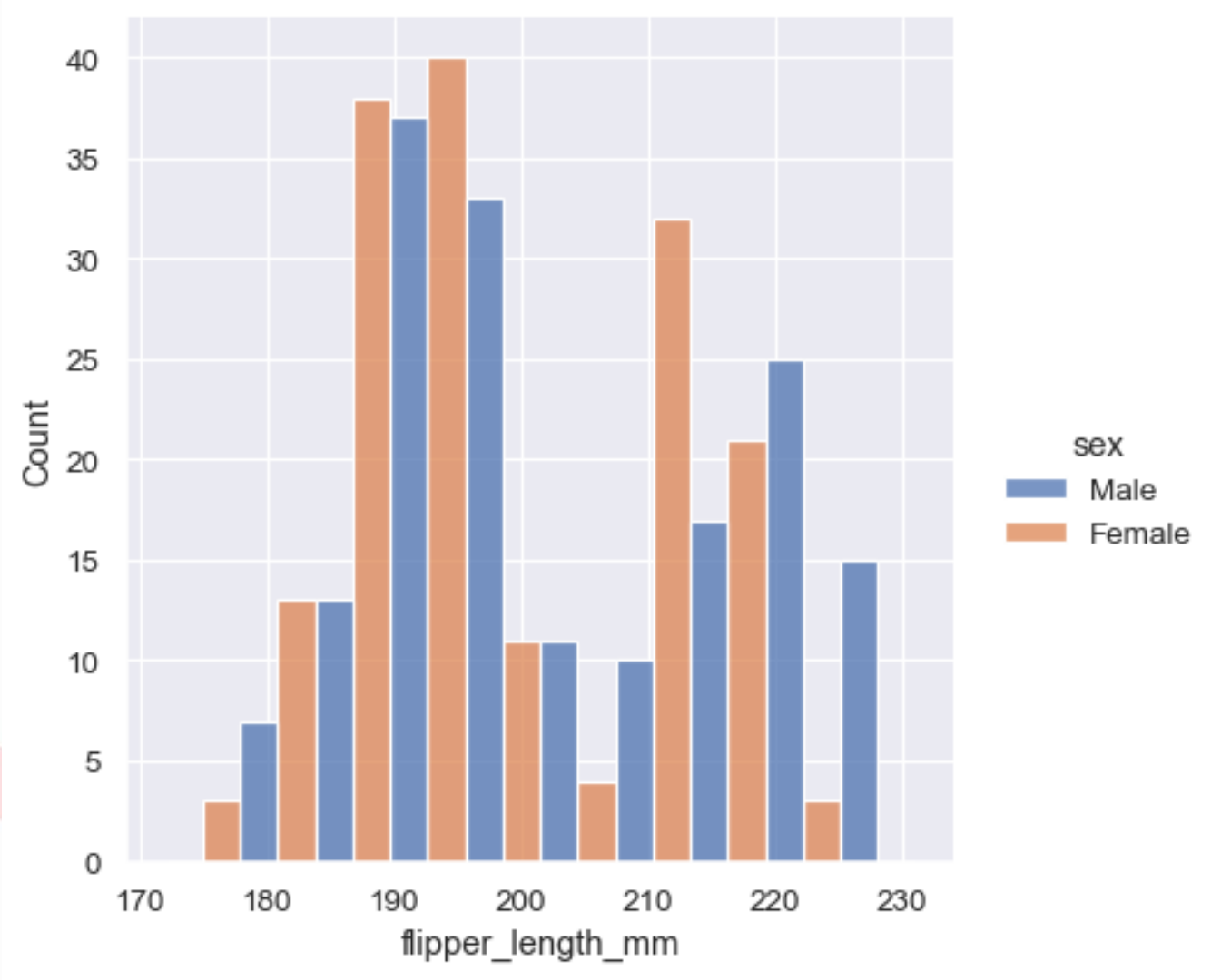
- Plotting univariate histograms

```
penguins = sns.load_dataset("penguins")
sns.displot(penguins, x="flipper_length_mm")
```



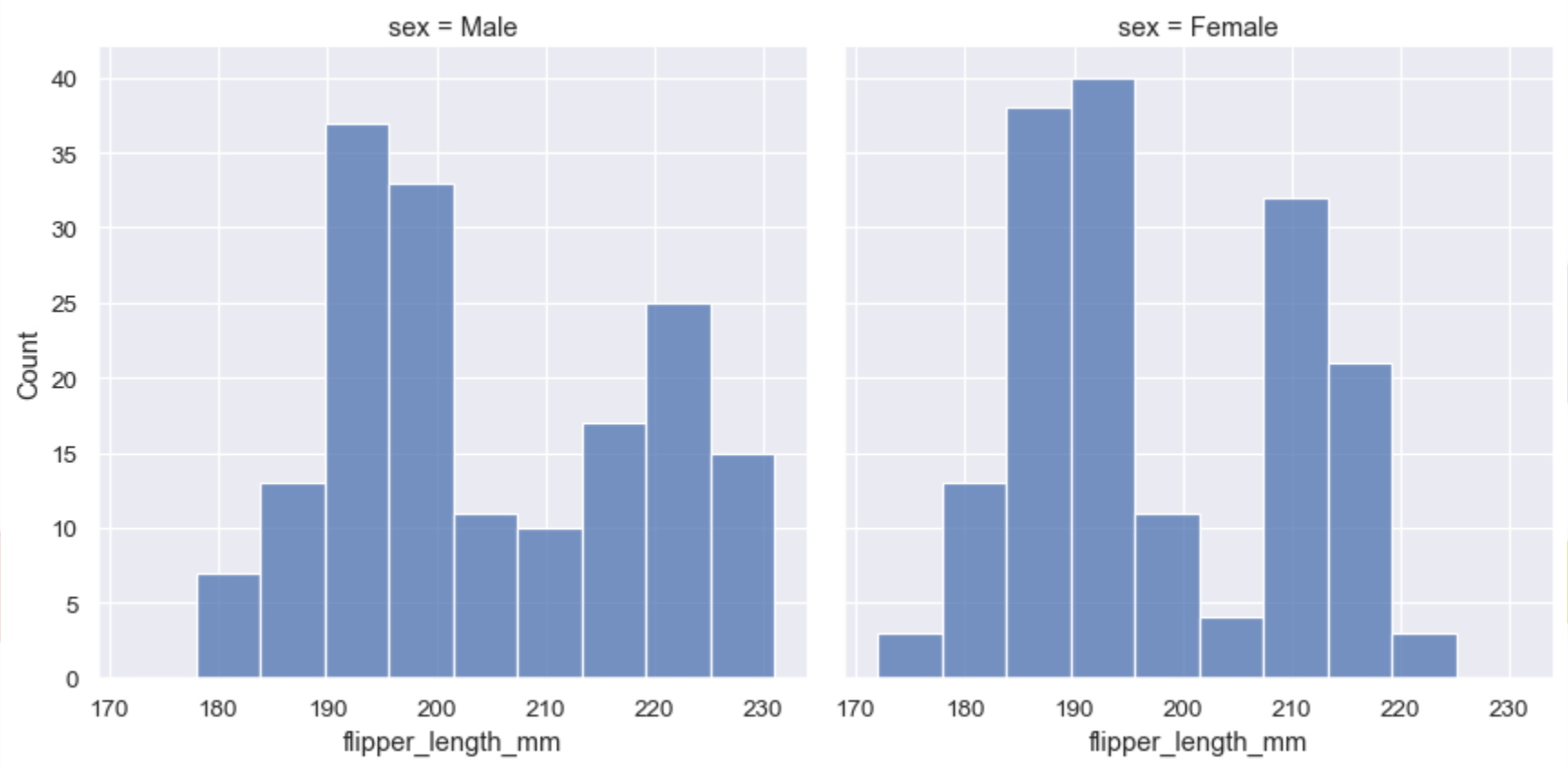
Seaborn

sns.displot(penguins, x="flipper_length_mm", hue="sex", multiple="dodge")



Seaborn

sns.displot(penguins, x="flipper_length_mm", col="sex", multiple="dodge")



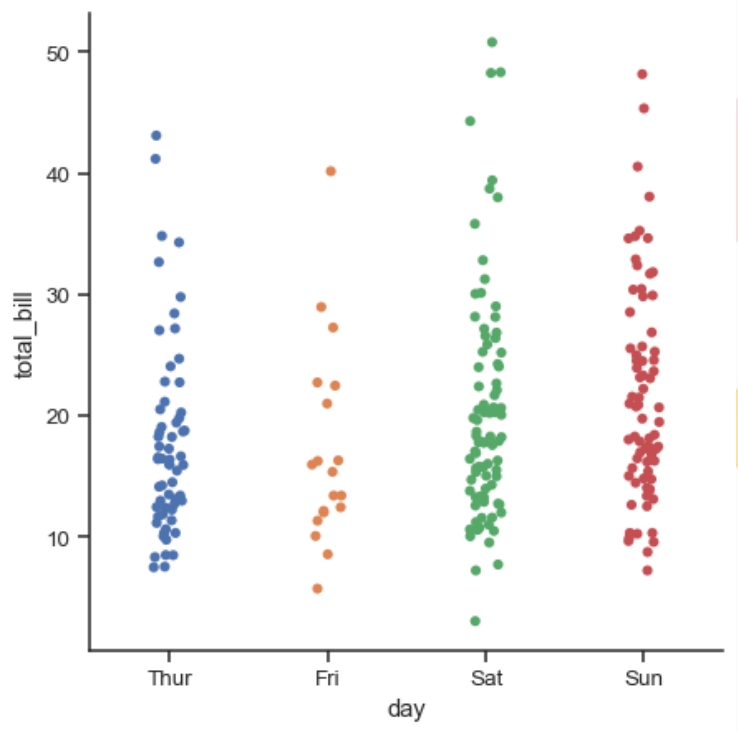
Seaborn

- Tham khảo thêm:
  - https://seaborn.pydata.org/tutorial/distributions.html

Seaborn

- Categorical scatterplots

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks", color_codes=True)

tips = sns.load_dataset("tips")
sns.catplot(x="day", y="total_bill", data=tips)
```



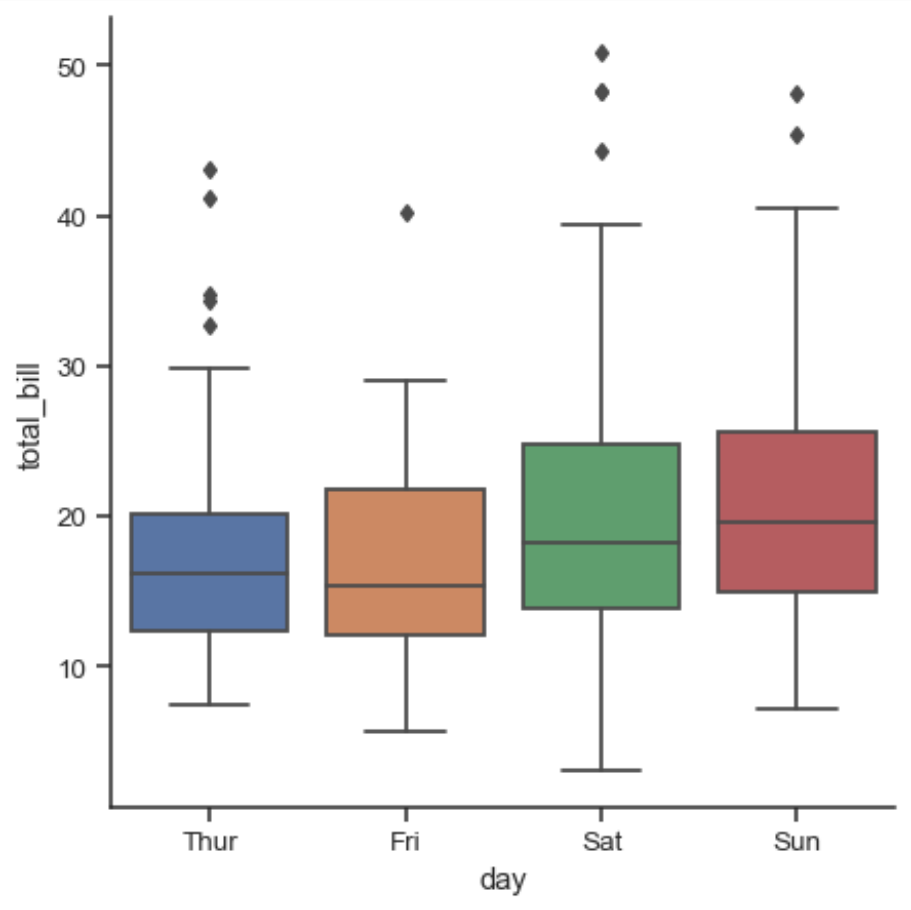
Seaborn

- Boxplots
  - Khi kích thước của tập dữ liệu tăng lên, các biểu đồ phân tán phân loại trở nên hạn chế về thông tin mà chúng có thể cung cấp về việc phân phối các giá trị trong mỗi danh mục.
    - Khi điều này xảy ra, có một số cách tiếp cận để tóm tắt thông tin phân phối theo những cách tạo điều kiện dễ dàng so sánh giữa các cấp độ danh mục.
  - Loại biểu đồ này hiển thị ba giá trị phần tư của phân phối cùng với các giá trị cực trị.

Seaborn

- Boxplots

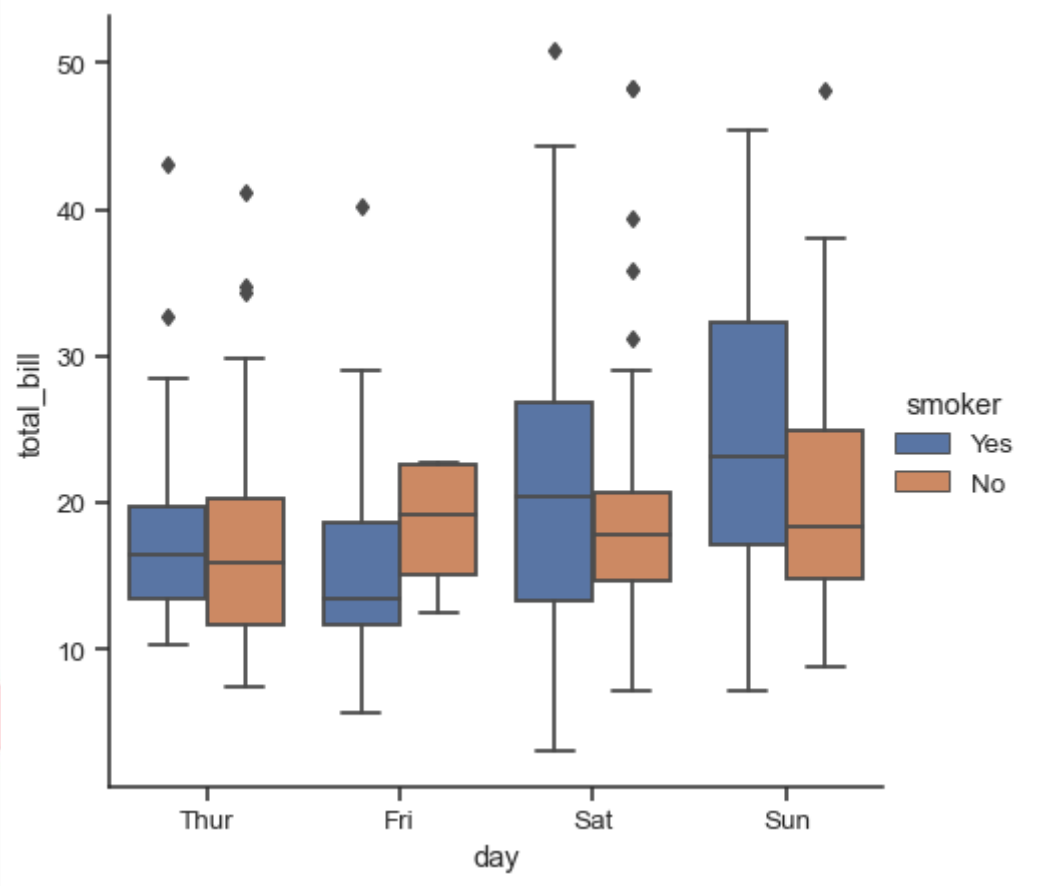
```
sns.catplot(x="day", y="total_bill", kind="box", data=tips)
```



Seaborn

- Boxplots

```
sns.catplot(x="day", y="total_bill", hue="smoker", kind="box", data=tips)
```

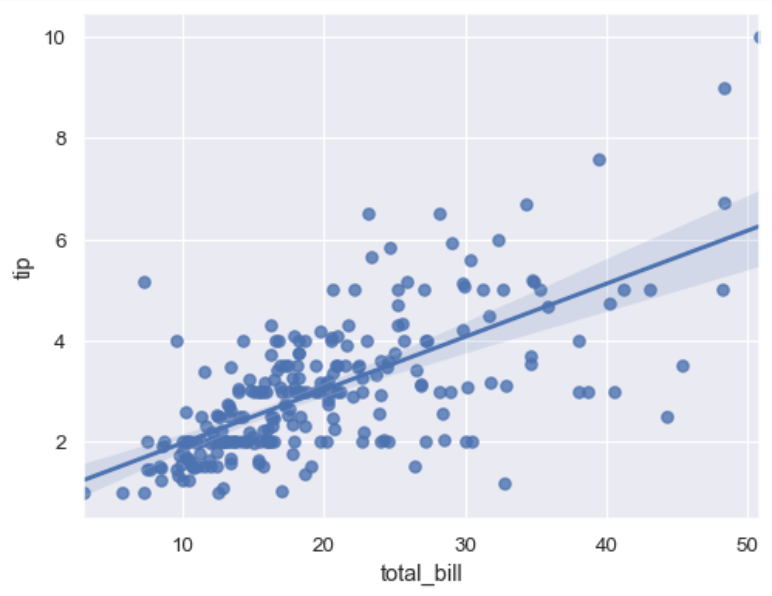


Seaborn

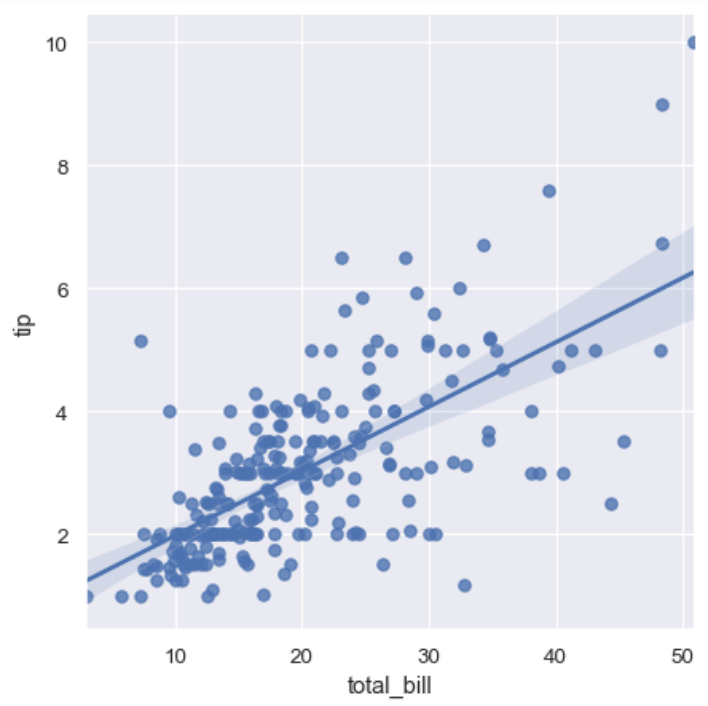
- Tham khảo thêm
  - https://seaborn.pydata.org/tutorial/categorical.html

Seaborn

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(color_codes=True)
tips = sns.load_dataset("tips")
sns.regplot(x="total_bill", y="tip", data=tips);
```



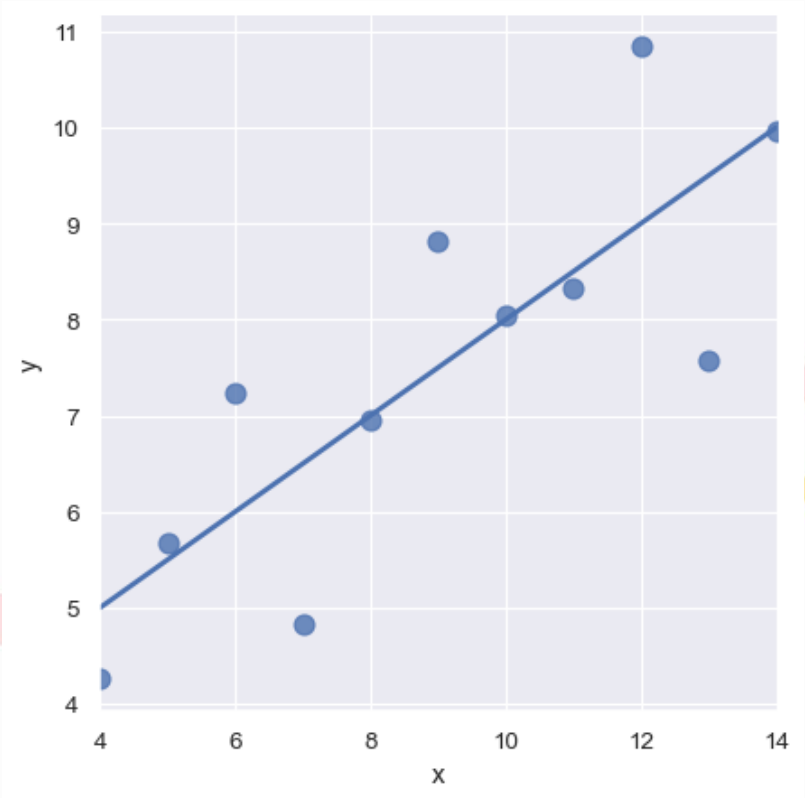

```
sns.lmplot(x="total_bill", y="tip", data=tips);
```



Seaborn

- Fitting different kinds of models

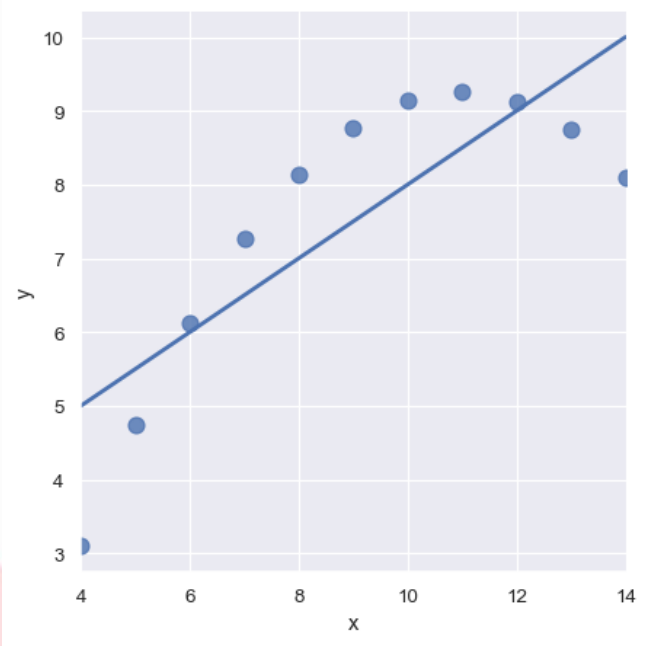
```
anscombe = sns.load_dataset("anscombe")
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'I'"),
           ci=None, scatter_kws={"s": 80});
```



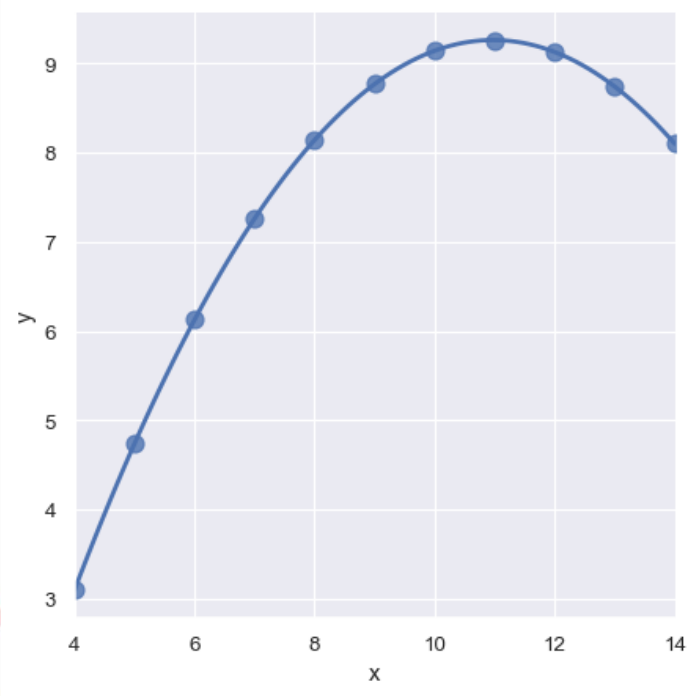
Seaborn

- Fitting different kinds of models

```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
        ci=None, scatter_kws={"s": 80});
```




```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
        order=2, ci=None, scatter_kws={"s": 80});
```



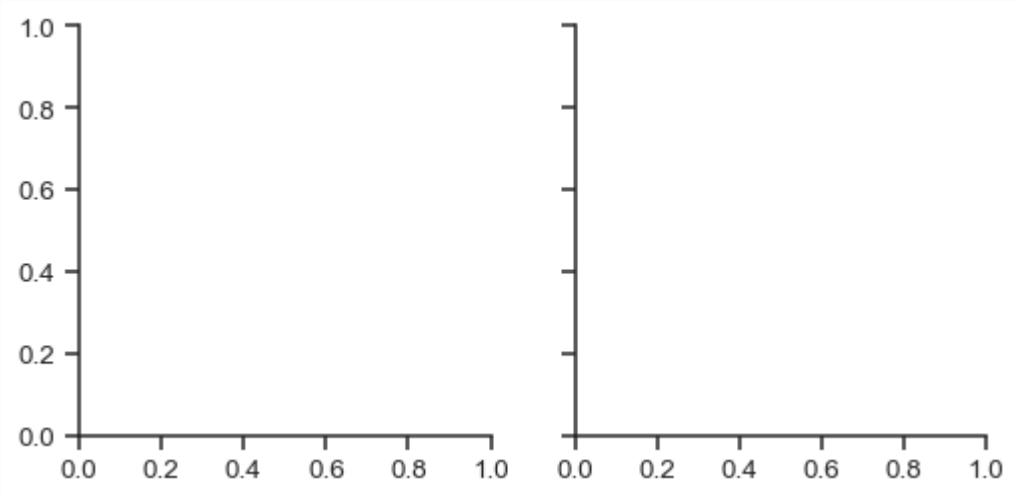
Seaborn

- Tham khảo thêm:
  - https://seaborn.pydata.org/tutorial/regression.html

Seaborn

- Multi-plot grids

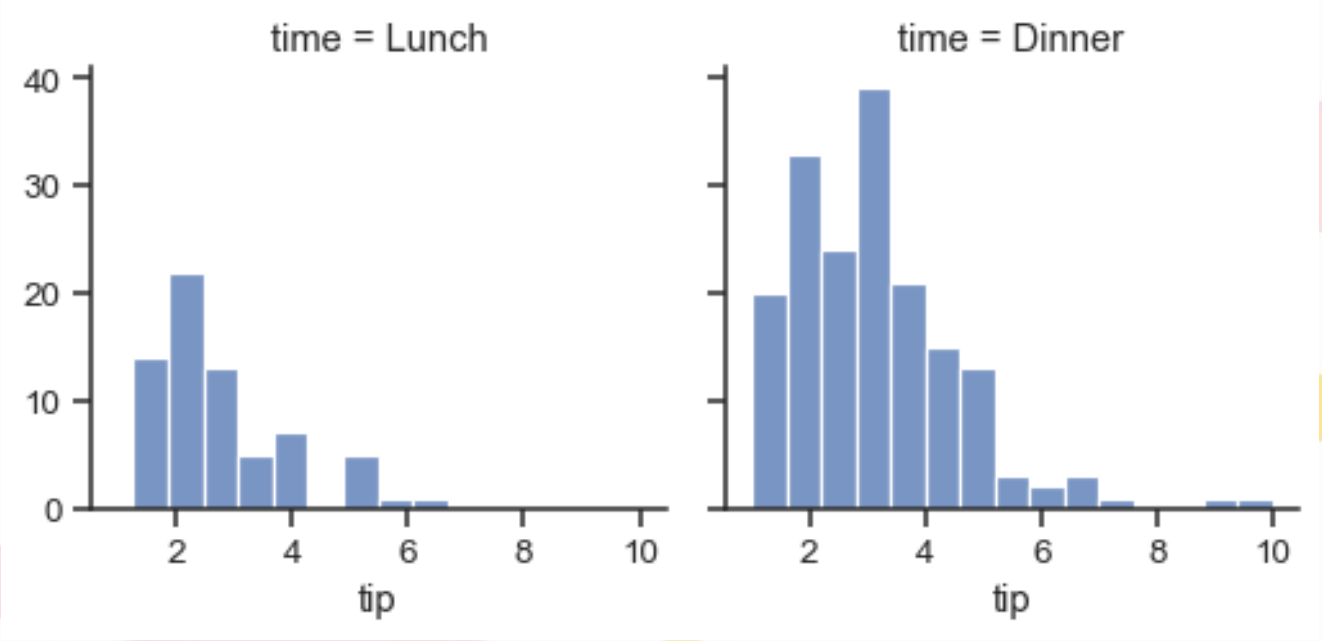
```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time")
```



Seaborn

- Multi-plot grids

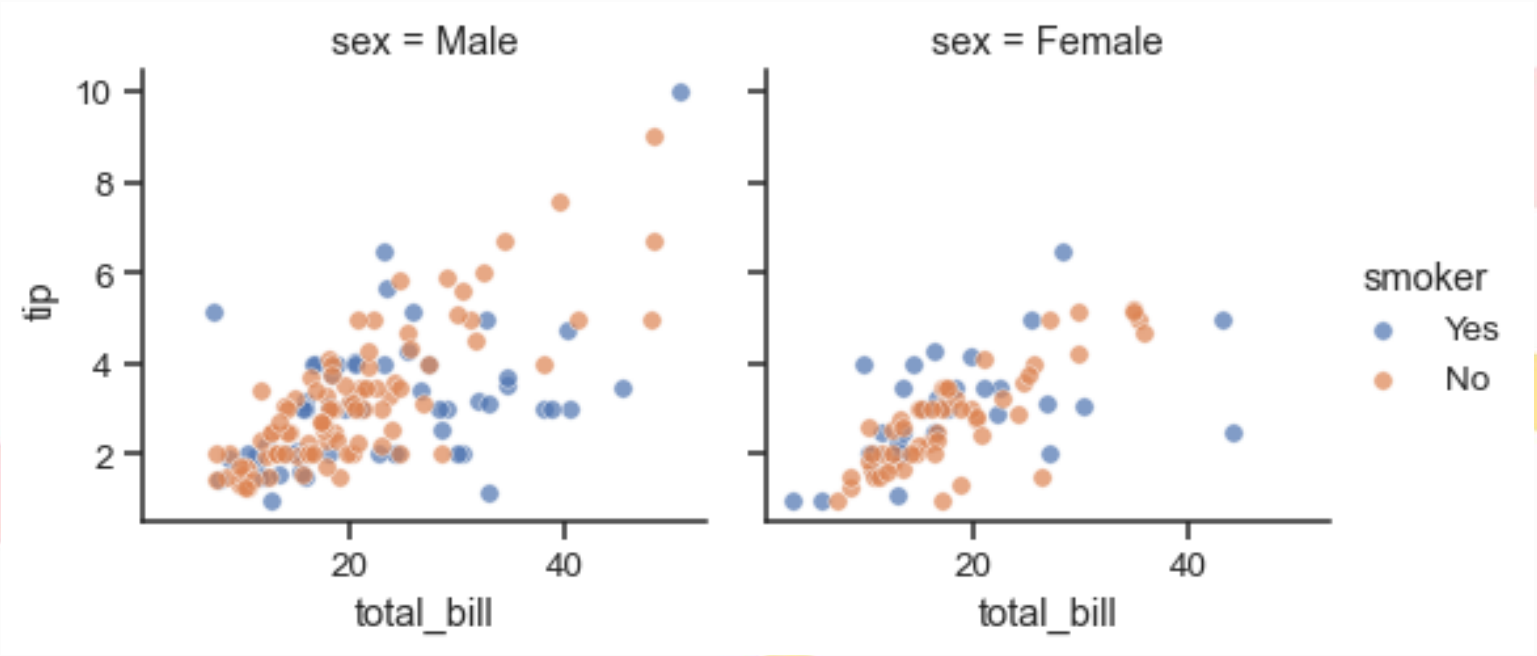
```
g = sns.FacetGrid(tips, col="time")
g.map(sns.histplot, "tip")
```



Seaborn

- Multi-plot grids

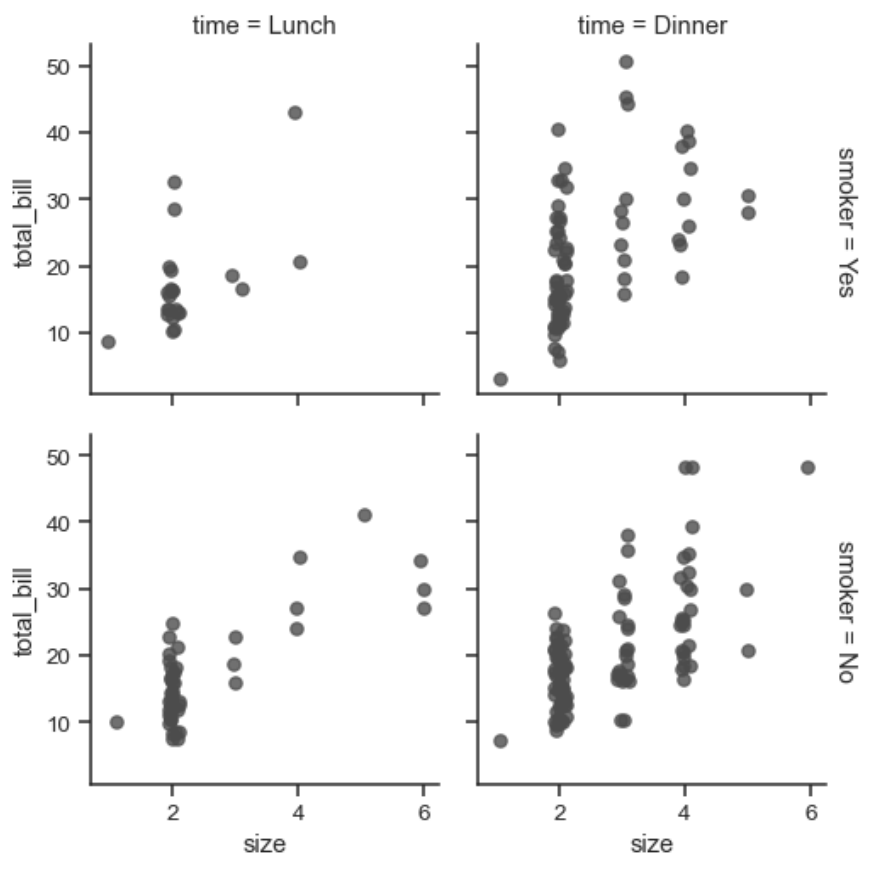
```
g = sns.FacetGrid(tips, col="sex", hue="smoker")
g.map(sns.scatterplot, "total_bill", "tip", alpha=.7)
g.add_legend()
```



Seaborn

- Multi-plot grids

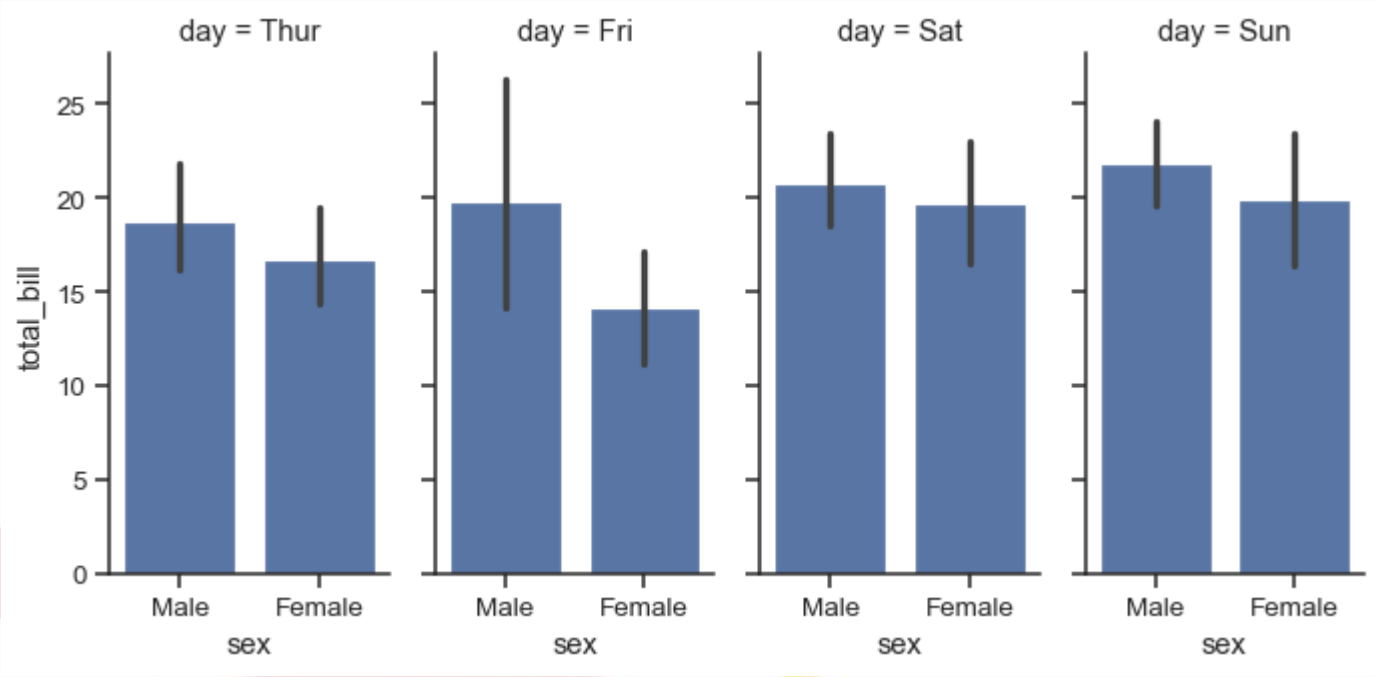
```
g = sns.FacetGrid(tips, row="smoker", col="time", margin_titles=True)
g.map(sns.regplot, "size", "total_bill", color=".3", fit_reg=False, x_jitter=.1)
```



Seaborn

- Multi-plot grids

```
g = sns.FacetGrid(tips, col="day", height=4, aspect=.5)
g.map(sns.barplot, "sex", "total_bill", order=["Male", "Female"])
```

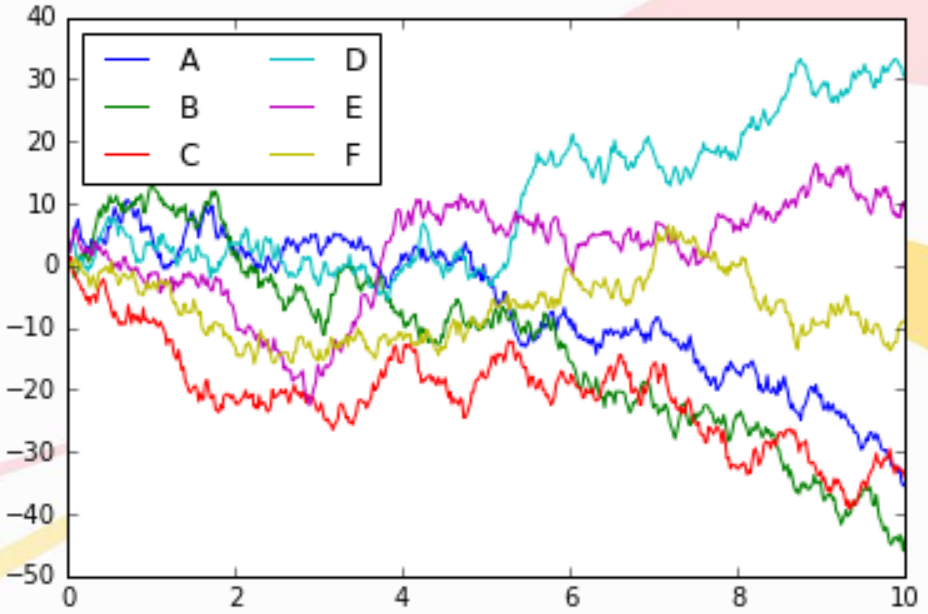


Seaborn

- Tham khảo them:
  - https://seaborn.pydata.org/tutorial/axis_grids.html

Seaborn Versus Matplotlib and other examples

```
import matplotlib.pyplot as plt
plt.style.use('classic')
%matplotlib inline
import numpy as np
import pandas as pd
%Now we create some random walk data:
# Create some data
rng = np.random.RandomState(0)
x = np.linspace(0, 10, 500)
y = np.cumsum(rng.randn(500, 6), 0)
%And do a simple plot:
# Plot the data with Matplotlib defaults
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
```



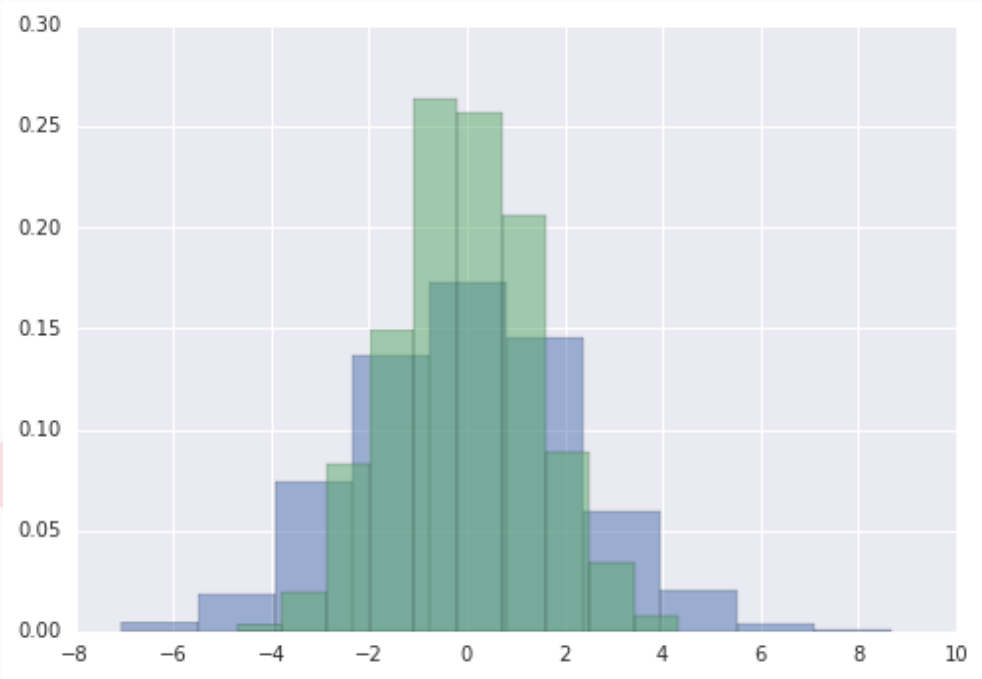
Seaborn Versus Matplotlib and other examples

```
import seaborn as sns
sns.set()
# same plotting code as above!
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
```

Seaborn Versus Matplotlib and other examples

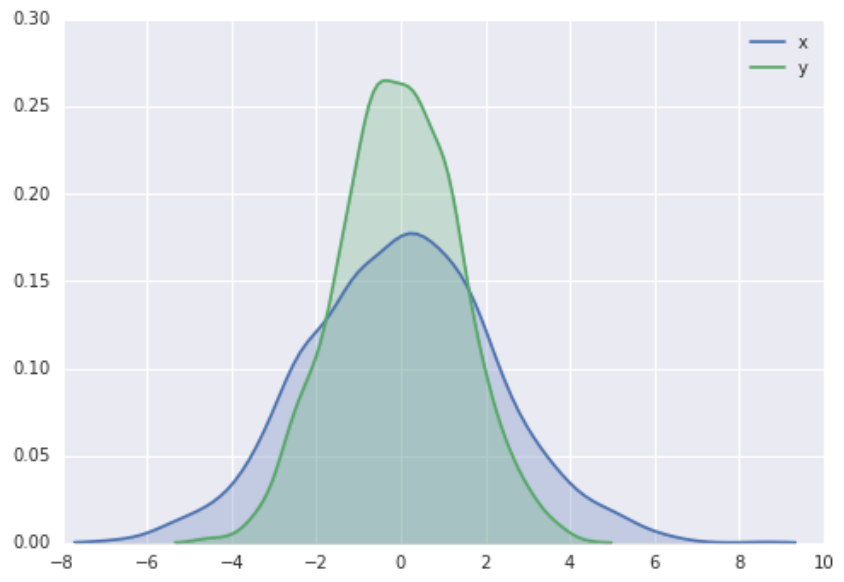
- Histograms, KDE, and densities

```
data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])

for col in 'xy':
    plt.hist(data[col], normed=True, alpha=0.5)
```

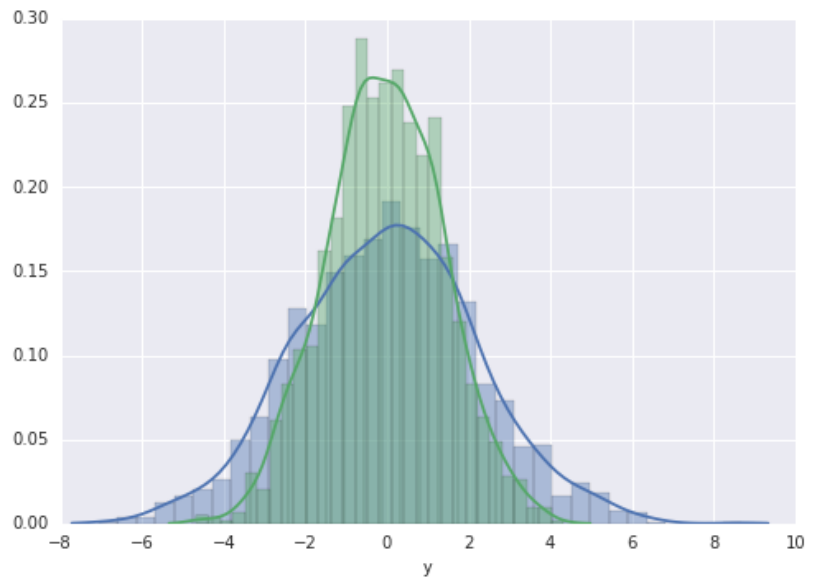


Seaborn Versus Matplotlib and other examples

```
for col in 'xy':
    sns.kdeplot(data[col], shade=True)
```

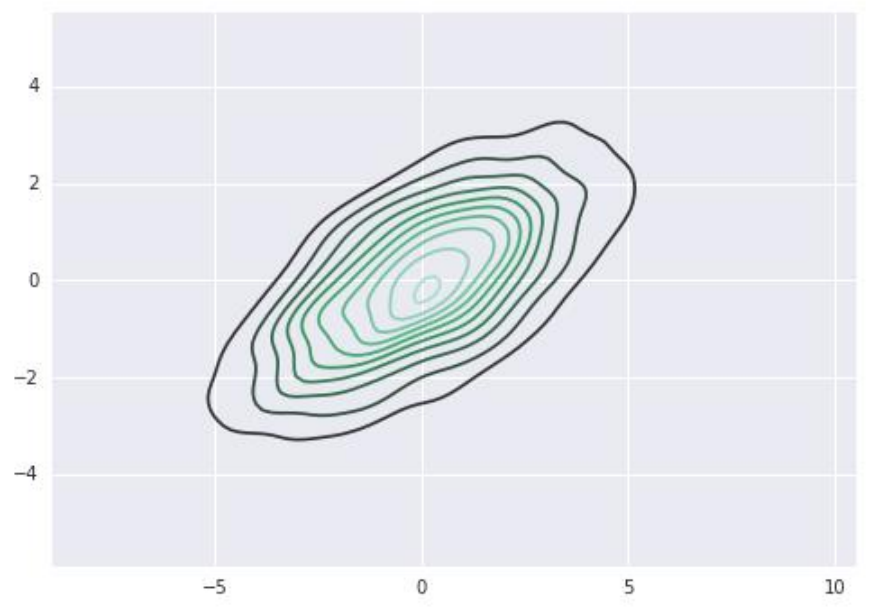



```
sns.distplot(data['x'])
sns.distplot(data['y']);
```

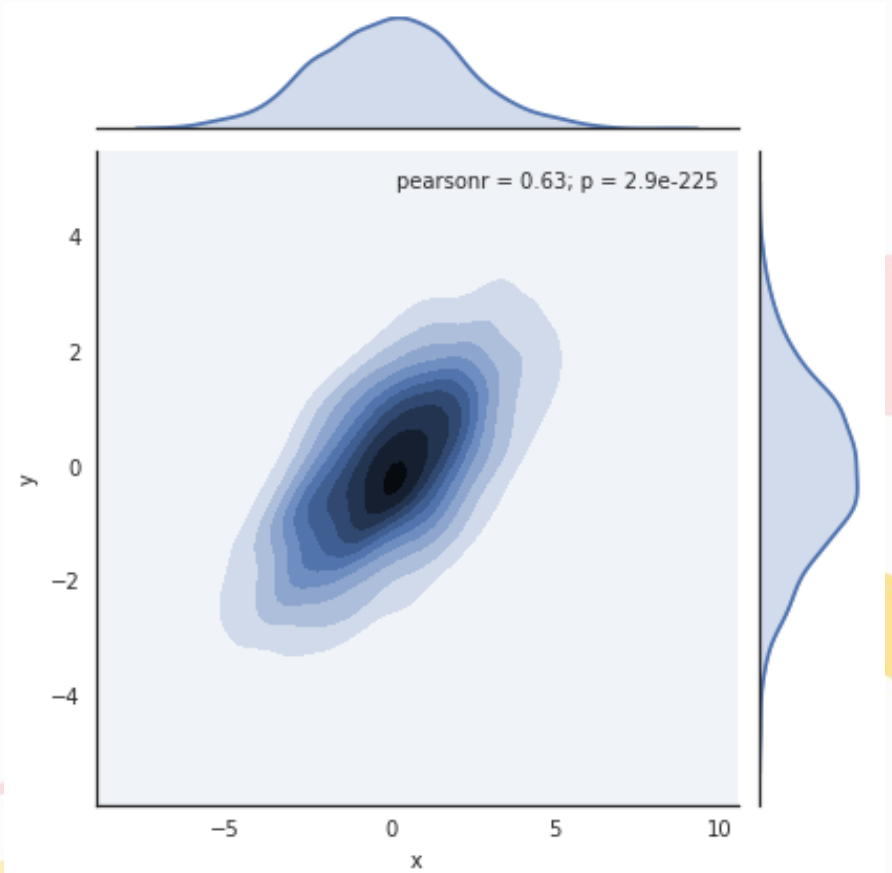


Seaborn Versus Matplotlib and other examples

sns.kdeplot(data);

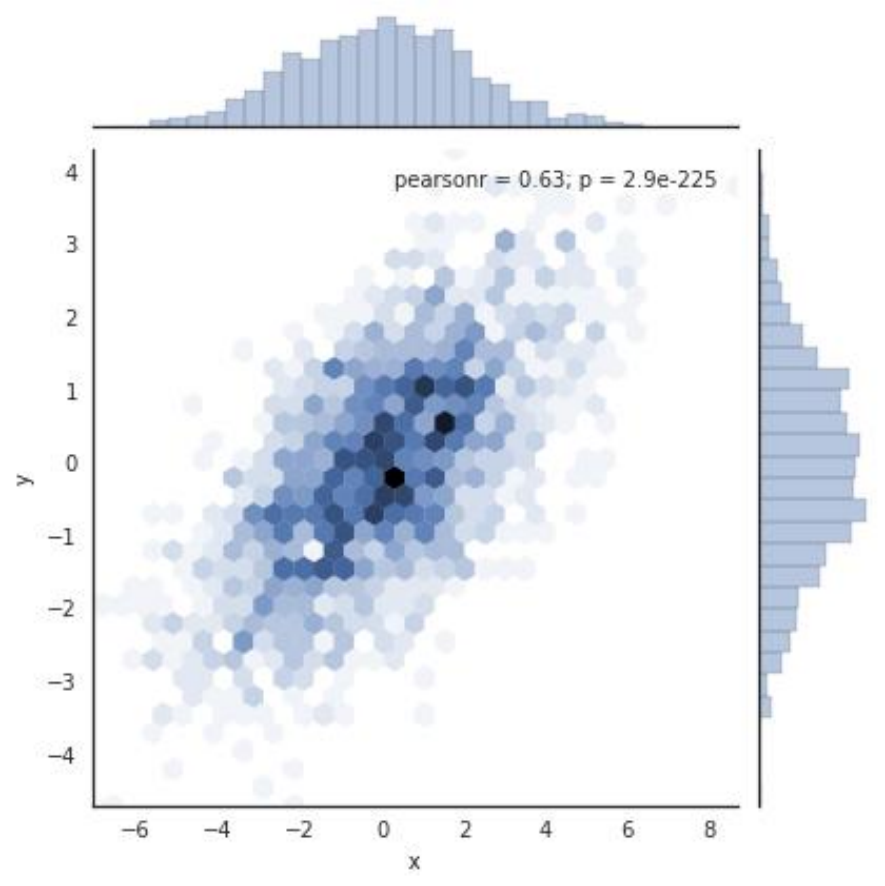


with sns.axes_style('white'):
sns.jointplot("x", "y", data, kind='kde');



Seaborn Versus Matplotlib and other examples

```
with sns.axes_style('white'):
    sns.jointplot("x", "y", data, kind='hex')
```



Seaborn Versus Matplotlib and other examples

- Pair plots

```
iris = sns.load_dataset("iris")
iris.head()
sns.pairplot(iris, hue='species', size=2.5);
```