EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Philosophische Fakultät
Seminar für Sprachwissenschaft

Data Science for Linguists

Session 8: Visualisation with Seaborn

Johannes Dellert

15 December, 2023

Table of Contents

Matplotlib and the Role of Seaborn

- **Matplotlib** (standard library for scientific visualisation in Python, you have probably used it), while extremely popular and feature-rich, is taken to have a range of undesirable properties:
  - ▷ API is quite low-level; all kinds of sophisticated visualisations are possible, but simple standard visualisations often require large amounts of boilerplate code
  - ▷ Matplotlib predates Pandas by more than a decade, and is therefore not designed to work with DataFrame objects directly; data series must first be extracted and recombined into the correct input format, which is inelegant and time consuming
  - ▷ color and style defaults look dated (though this has been improved in recent versions)
- **Seaborn** (which we will explore for visualisation) solves these issues:
  - ▷ API on top of Matplotlib which offers more modern default choices for plot styles and colours
  - ▷ simple high-level functions for common statistical plot types (which we will cover today)
  - ▷ integrates with the functionality provided by Pandas (e.g. operating on DataFrame objects)
- Matplotlib proper is adapting, and might regain its status as the tool of choice in the future
- Seaborn is primarily intended for effortless plotting of standard datatypes during data exploration; for full customisability and publication-quality visualisations, dropping down into the underlying Matplotlib functionality for fine-grained tweaking is necessary

Seaborn: Installation Basic Usage

- installation should work in the usual fashion (assuming you work from Jupyter):

```
In [] !pip install seaborn
```

- switch to matplotlib mode for inline visualisations, and conventions for imports:

```
In [] %matplotlib inline
      import matplotlib.pyplot as plt
      import seaborn as sns
      import numpy as np
      import pandas as pd
```

- Seaborn should be initialised with a chart style, this is the default Matplotlib reconfiguration:

```
In [] sns.set()
```

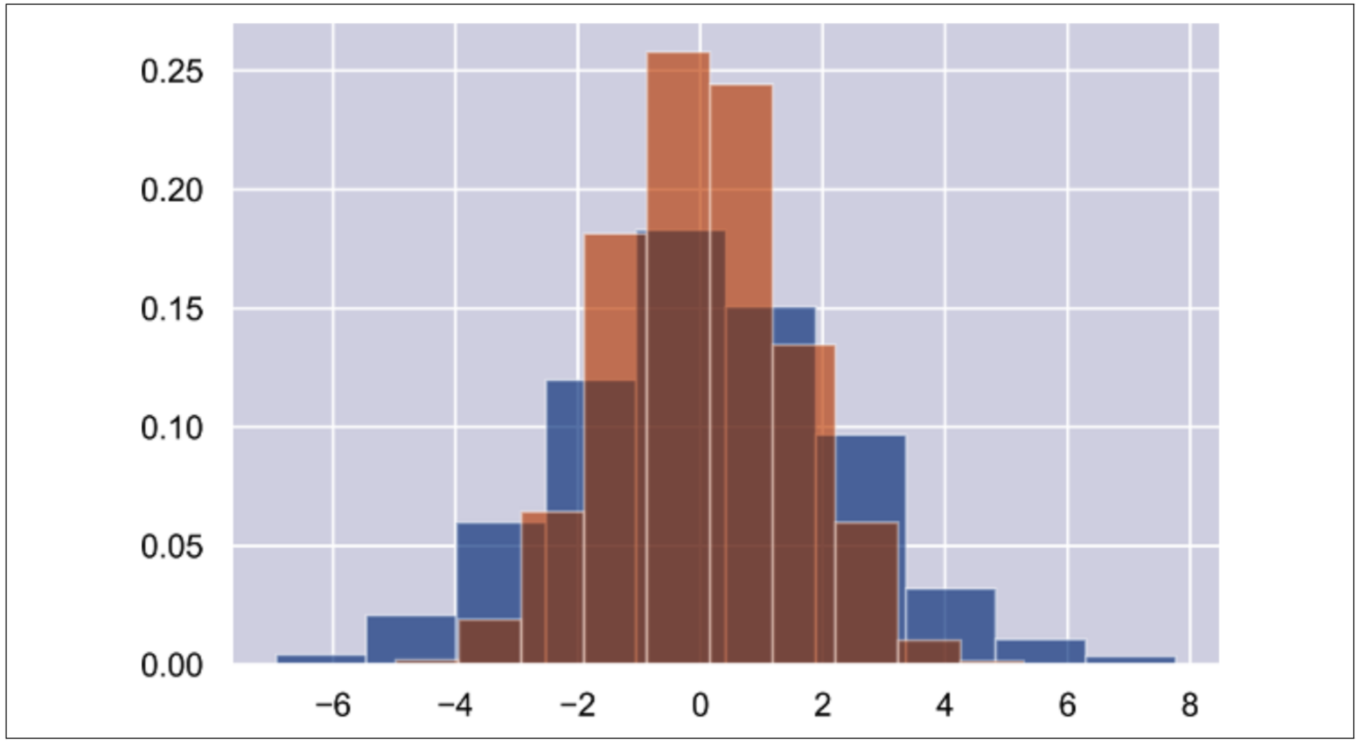
- outside matplotlib mode, visualisations need to be opened explicitly (separate window!):

```
In [] plt.show()
```

Table of Contents

Matplotlib: Histogram Example

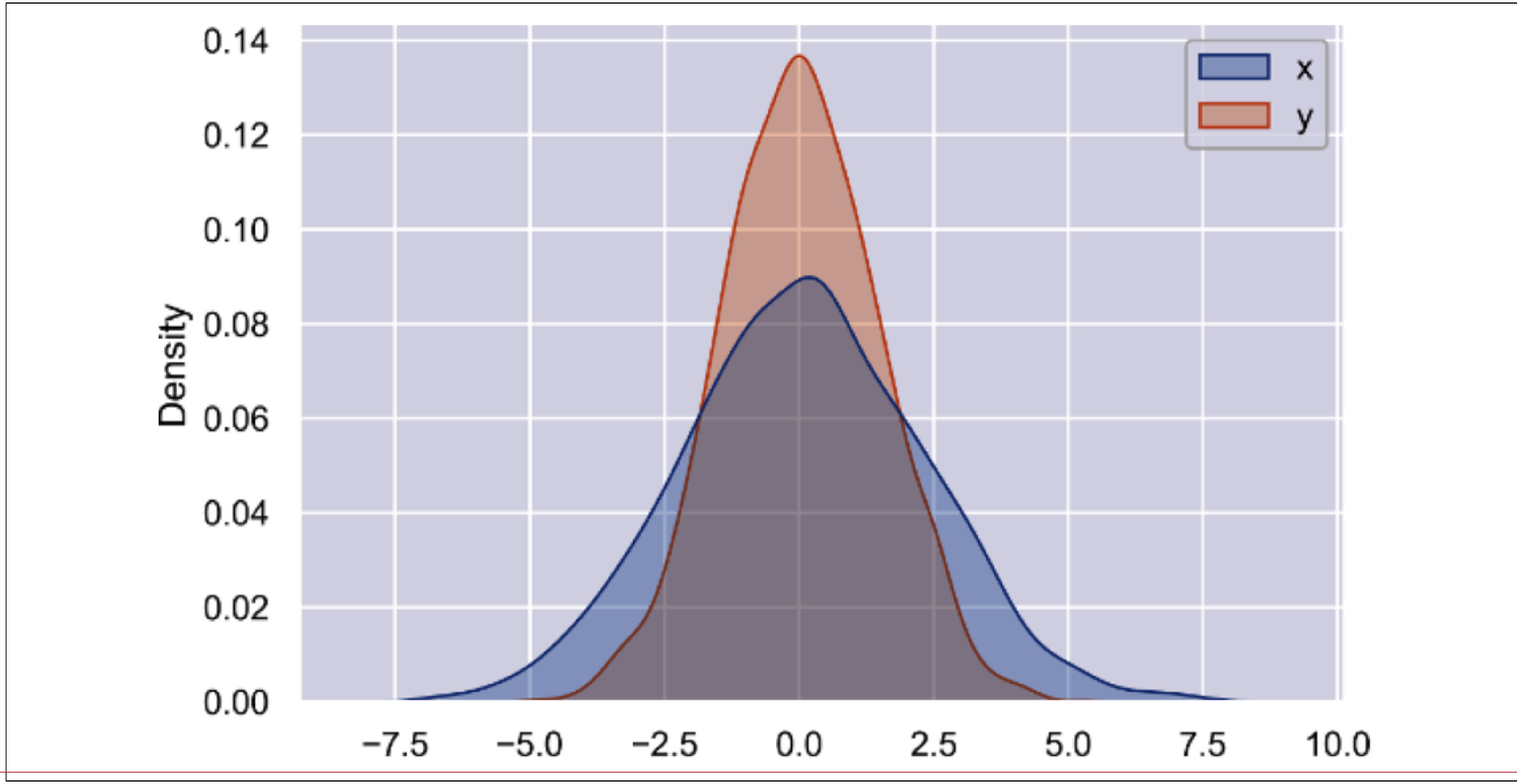
```
In [2]: data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)
        data = pd.DataFrame(data, columns=['x', 'y'])

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



Seaborn: Smooth Density Estimates

- Seaborn one-liner for smooth estimate based on kernel density estimation (Session 12)

```
In [3]: sns.kdeplot(data=data, shade=True);
```



Seaborn: Two-Dimensional Smoothed Joint Density

- by passing column names as dimensions, we get a 2D visualisation of joint density:

```
In [4]: sns.kdeplot(data=data, x='x', y='y');
```

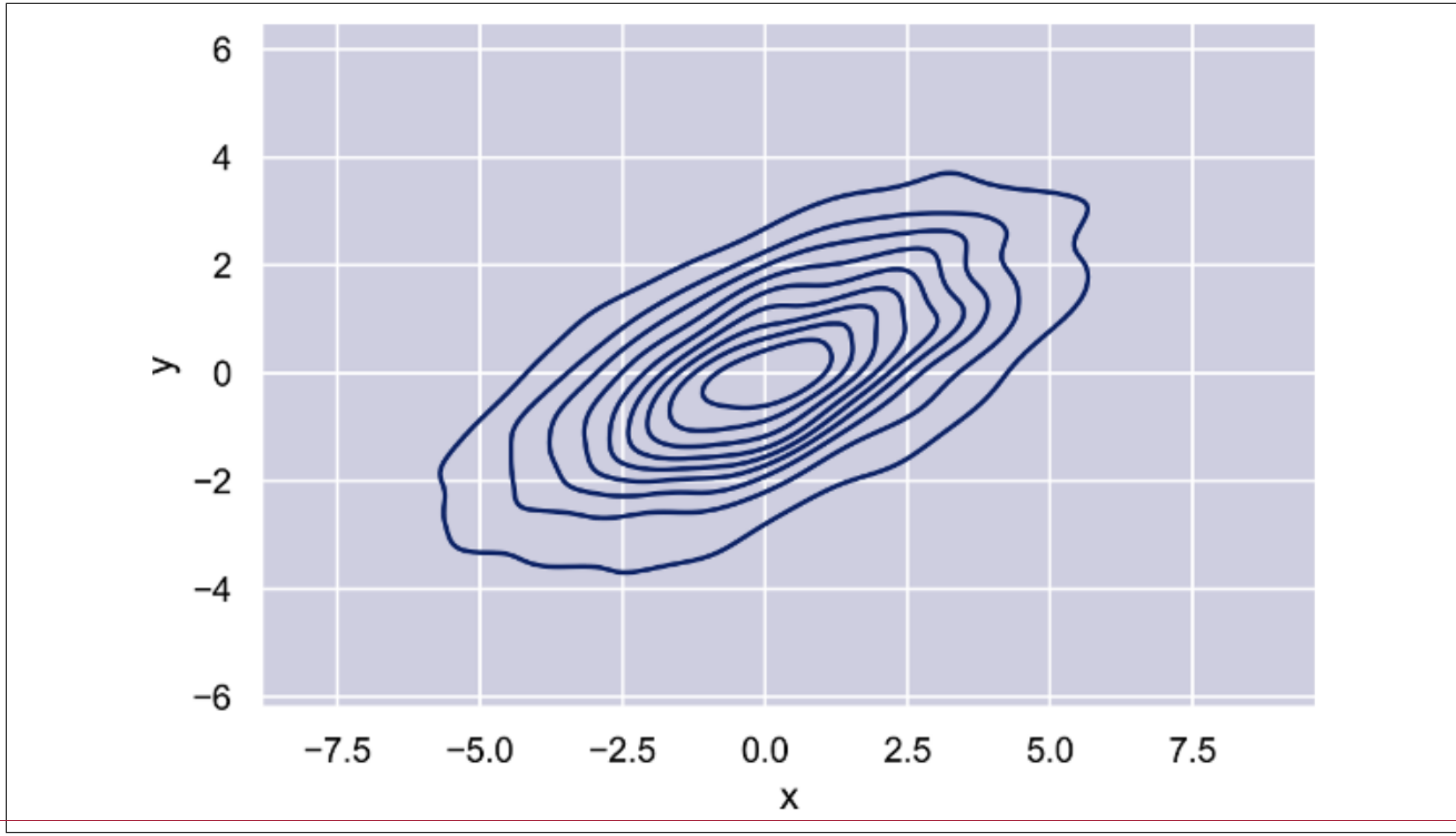


Table of Contents

Seaborn: Pair Plots

- in multidimensional data, it is easiest to spot patterns when we plot all pairs of variables against each other
- standard example: the Iris dataset listing measurements of petals and sepals of three Iris species

  `ir = sns.load_dataset("iris")`

- visualisation is a single function call:

  `sns.pairplot(ir ,hue="species")`

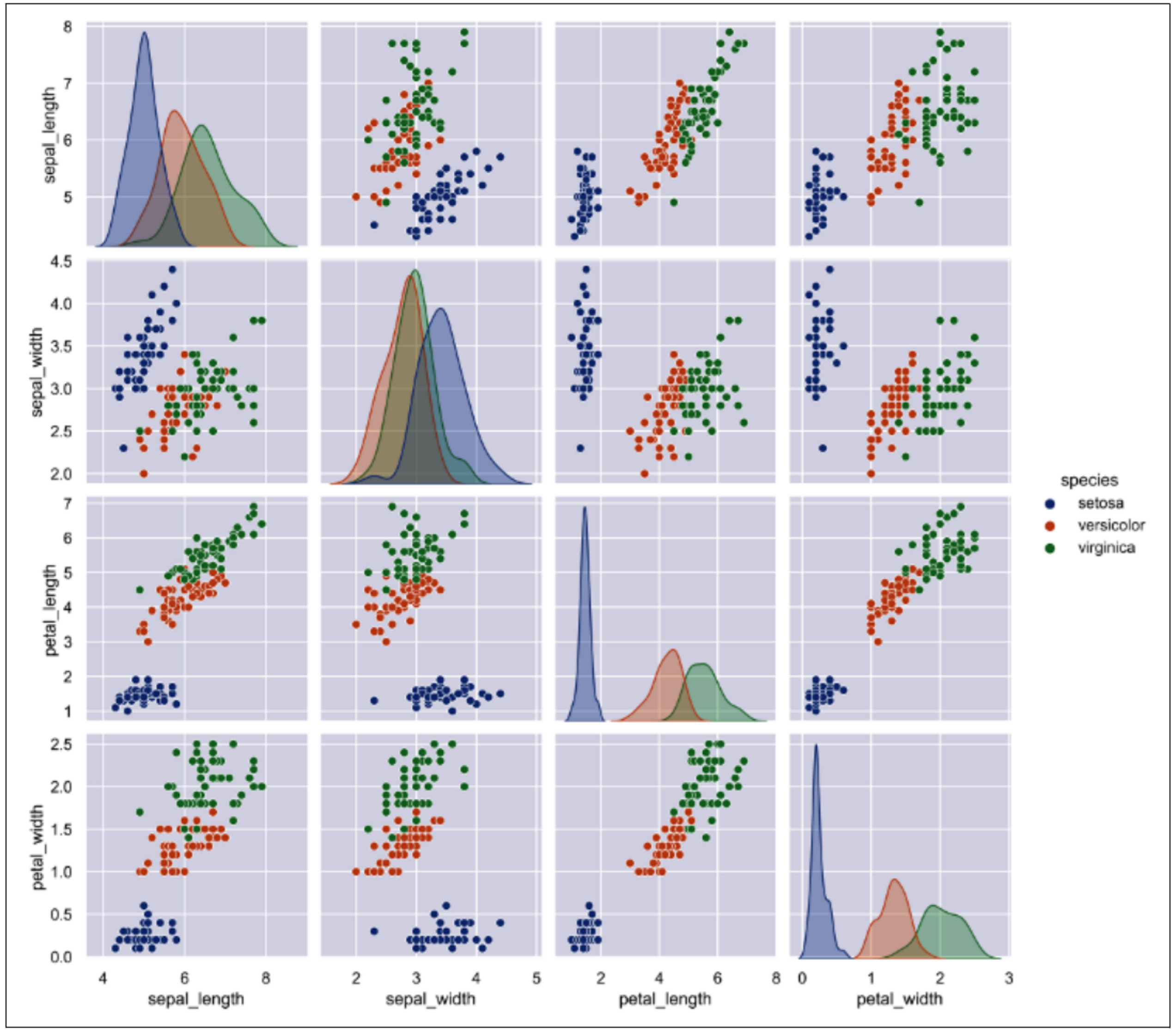


Table of Contents

Seaborn: Faceted Histograms

- sometimes the best way to understand data is via histograms of subsets
- the usefulness of **faceted histograms** can be illustrated using the Tips dataset, wich records the amounts that restaurant staff receive in tips based on various indicator data

```
In [7]: tips = sns.load_dataset('tips')
        tips.head()
Out[7]:    total_bill   tip     sex smoker  day    time  size
        0       16.99  1.01  Female     No  Sun  Dinner     2
        1       10.34  1.66    Male     No  Sun  Dinner     3
        2       21.01  3.50    Male     No  Sun  Dinner     3
        3       23.68  3.31    Male     No  Sun  Dinner     2
        4       24.59  3.61  Female     No  Sun  Dinner     4

In [8]: tips['tip_pct'] = 100 * tips['tip'] / tips['total_bill']

        grid = sns.FacetGrid(tips, row="sex", col="time", margin_titles=True)
        grid.map(plt.hist, "tip_pct", bins=np.linspace(0, 40, 15));
```

Seaborn: Faceted Histograms

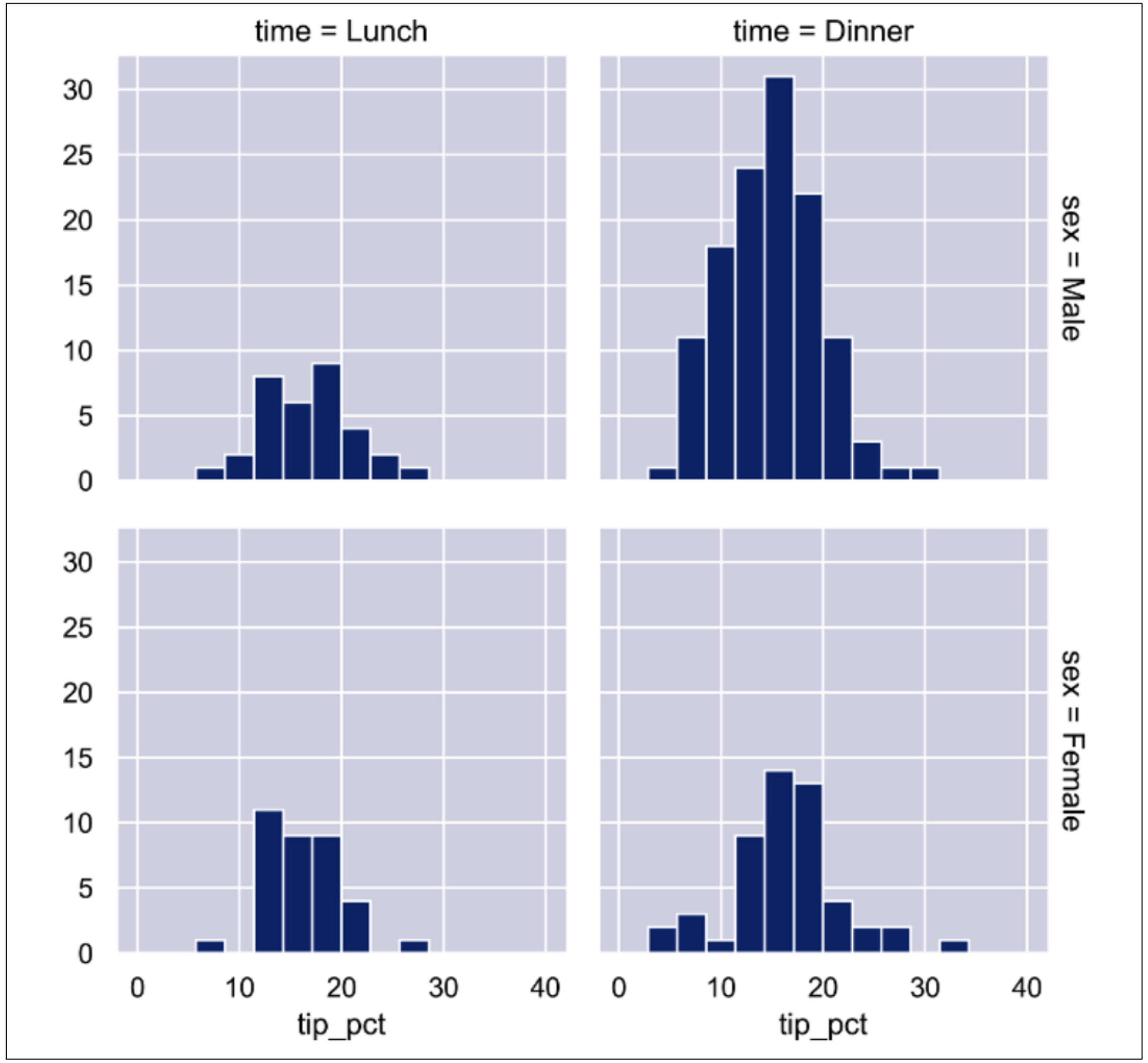
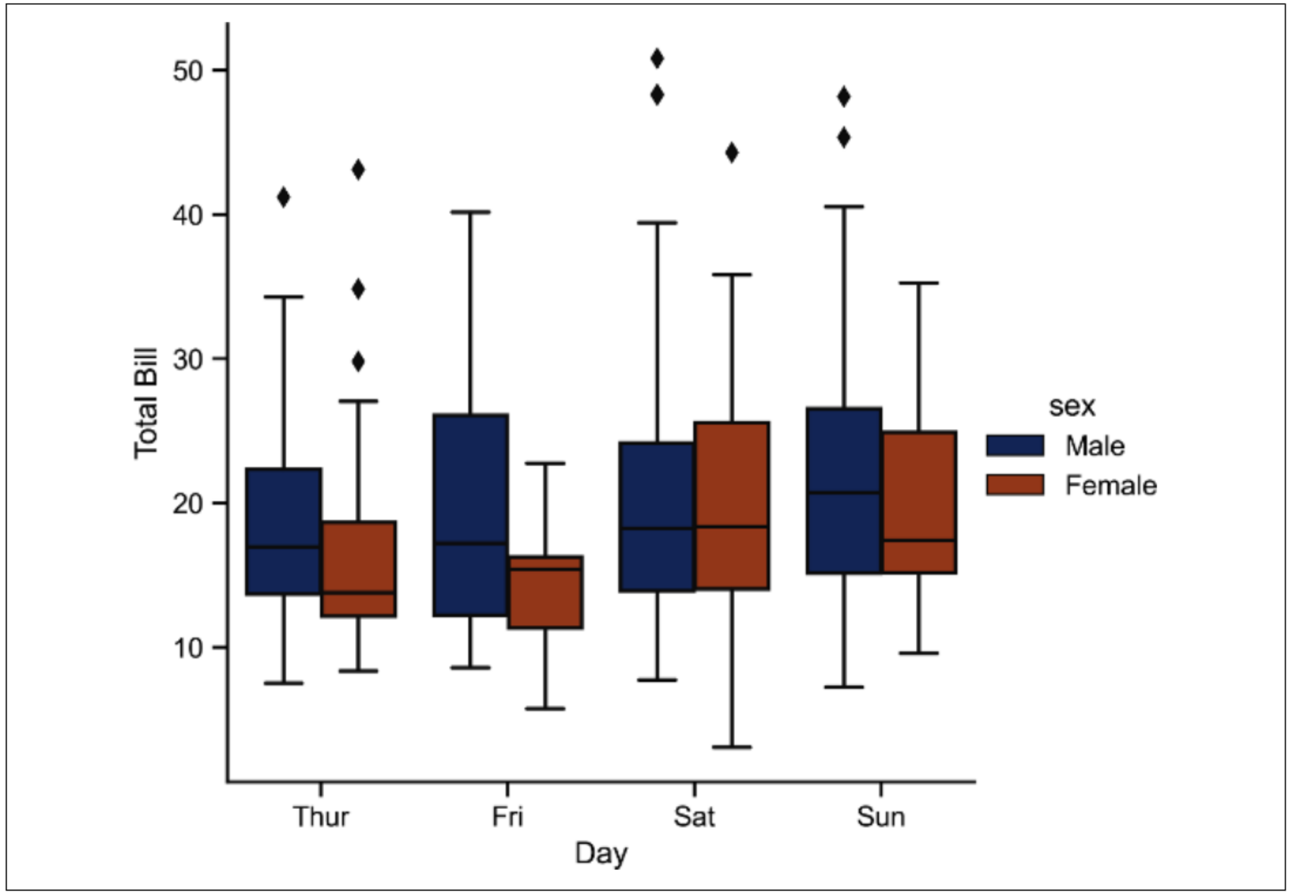


Table of Contents

Categorical Plots: Factor Plot

- a factor plot shows the distribution of a parameter within bins defined by some other parameter

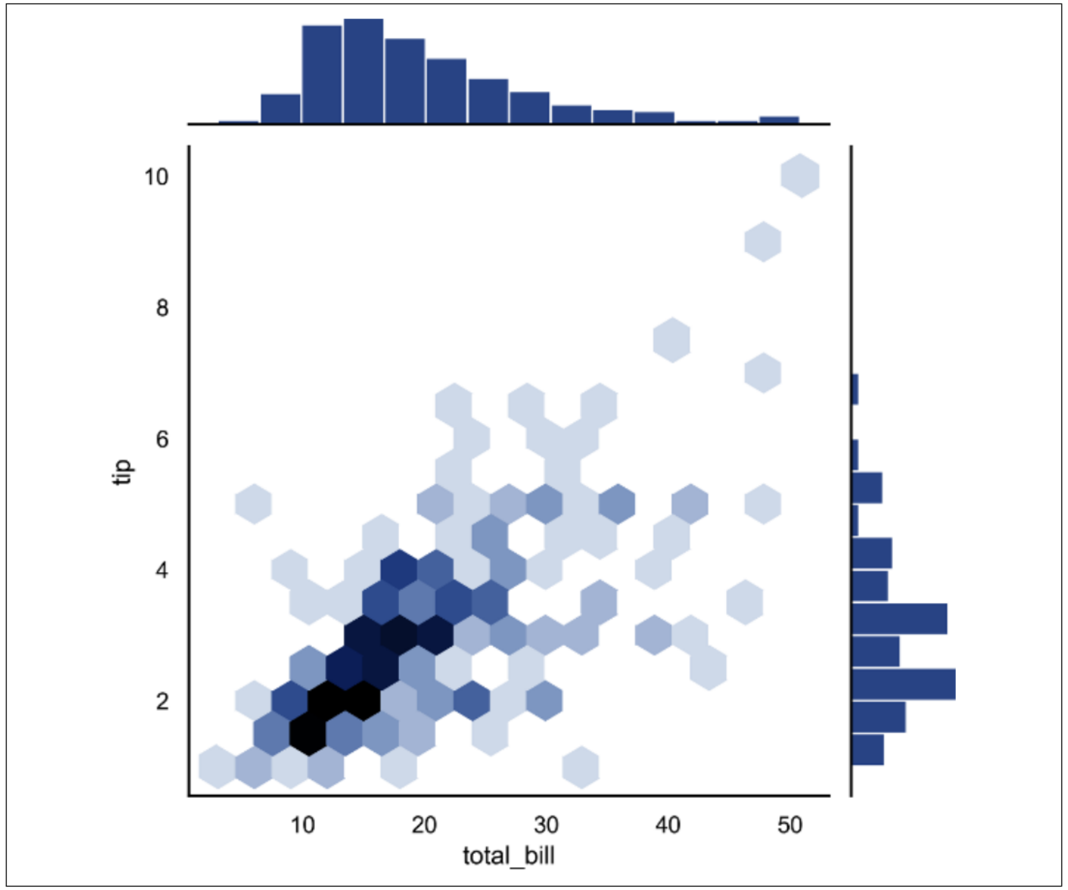
```
In [9]: with sns.axes_style(style='ticks'):
            g = sns.catplot(x="day", y="total_bill", hue="sex",
                            data=tips, kind="box")
            g.set_axis_labels("Day", "Total Bill");
```



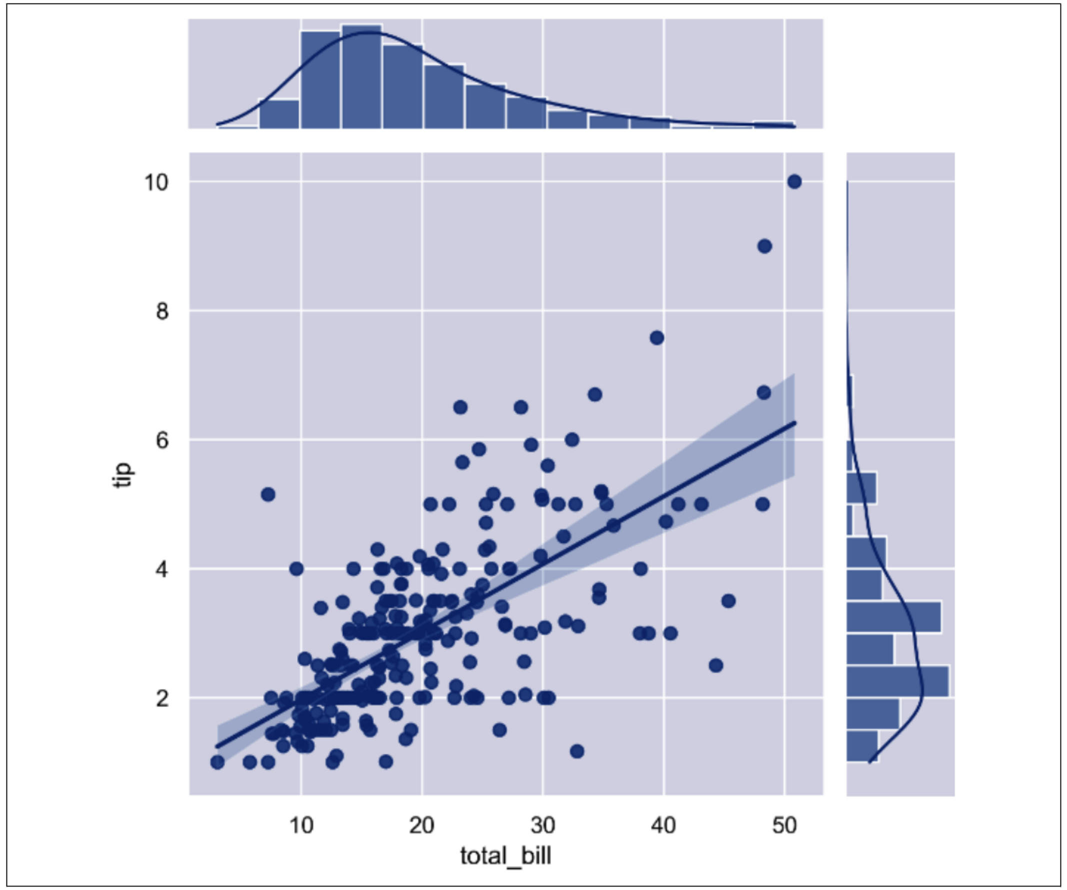
Categorical Plots: Joint Distributions

- joint distribution along with associated marginal distributions are shown by `jointplot`:
  `sns.jointplot(x="total_bill", y="tip", data=tips, kind=kind)`

`kind='hex'` yields hexes for joint density:



`kind='reg'` yields KDEs and regression:



Categorical Plots: Bar Plots

- to illustrate more general bar plots, we use the Planets dataset, which contains data about known exoplanets along with the year and the method of their discovery

```
In [12]: planets = sns.load_dataset('planets')
         planets.head()
Out[12]:
```

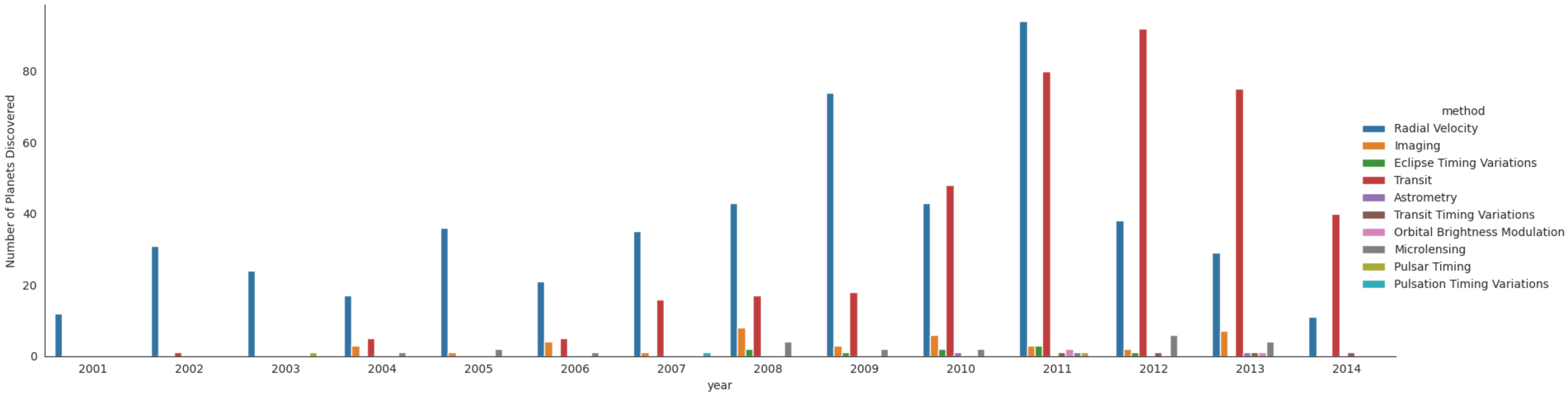
	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

```
In [13]: with sns.axes_style('white'):
             g = sns.catplot(x="year", data=planets, aspect=2,
                             kind="count", color='steelblue')
             g.set_xticklabels(step=5)
```

Categorical Plots: Bar Plots

- we create a bar plot of the number of planets discovered each year, classified by the methods of discovery (keyword hue with column ID, because bars are distinguished by their colour)

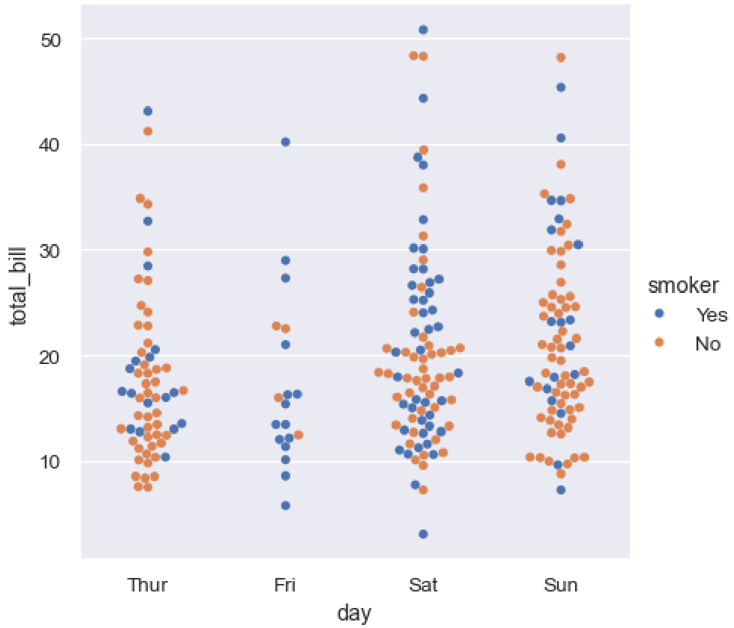
```
In [14]: with sns.axes_style('white'):
             g = sns.catplot(x="year", data=planets, aspect=4.0, kind='count',
                             hue='method', order=range(2001, 2015))
             g.set_ylabels('Number of Planets Discovered')
```



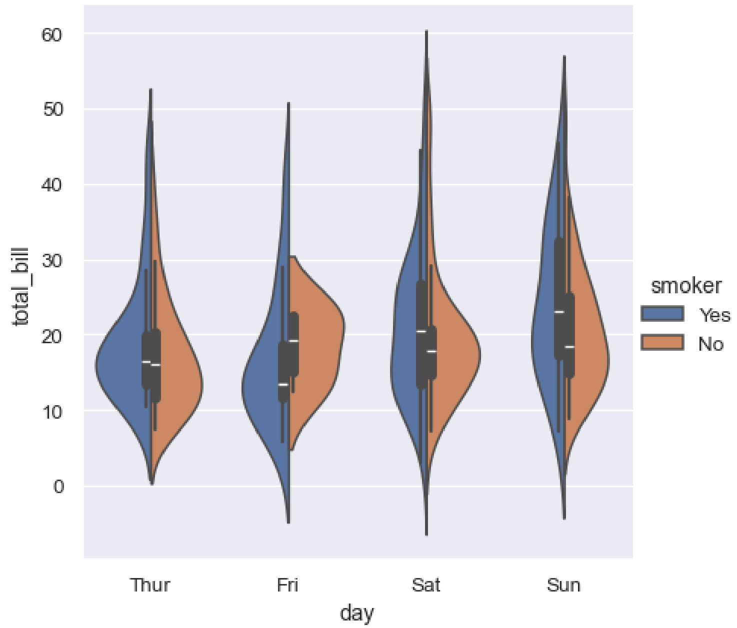
Categorical Plots: Swarm and Violin Plots

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

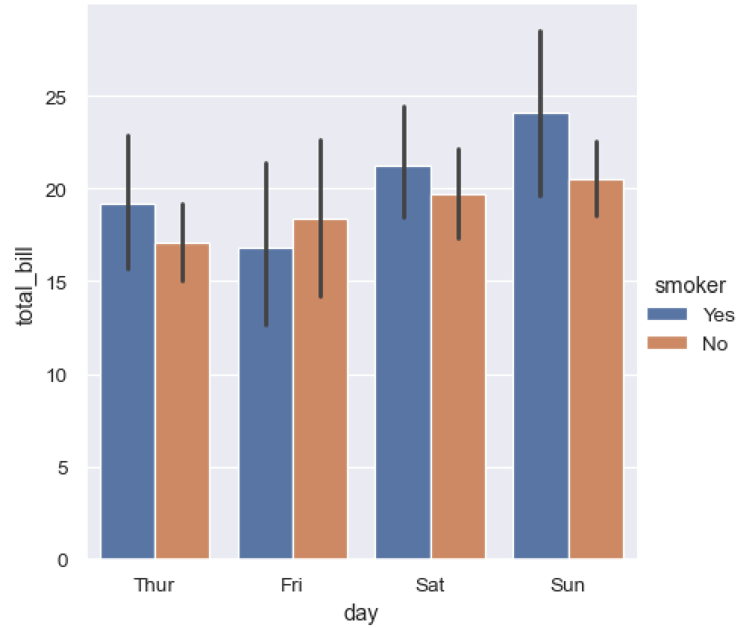
with kind="swarm":



with kind="violin":



with kind="bar":



Categorical Plots: Scatter Plots

```
import seaborn as sns
sns.set_theme(style="whitegrid")

# Load the example planets dataset
planets = sns.load_dataset("planets")

cmap = sns.cubehelix_palette(rot=-.2, as_cmap=True)
g = sns.relplot(
    data=planets,
    x="distance", y="orbital_period",
    hue="year", size="mass",
    palette=cmap, sizes=(10, 200),
)
g.set(xscale="log", yscale="log")
g.ax.xaxis.grid(True, "minor", linewidth=.25)
g.ax.yaxis.grid(True, "minor", linewidth=.25)
g.despine(left=True, bottom=True)
```

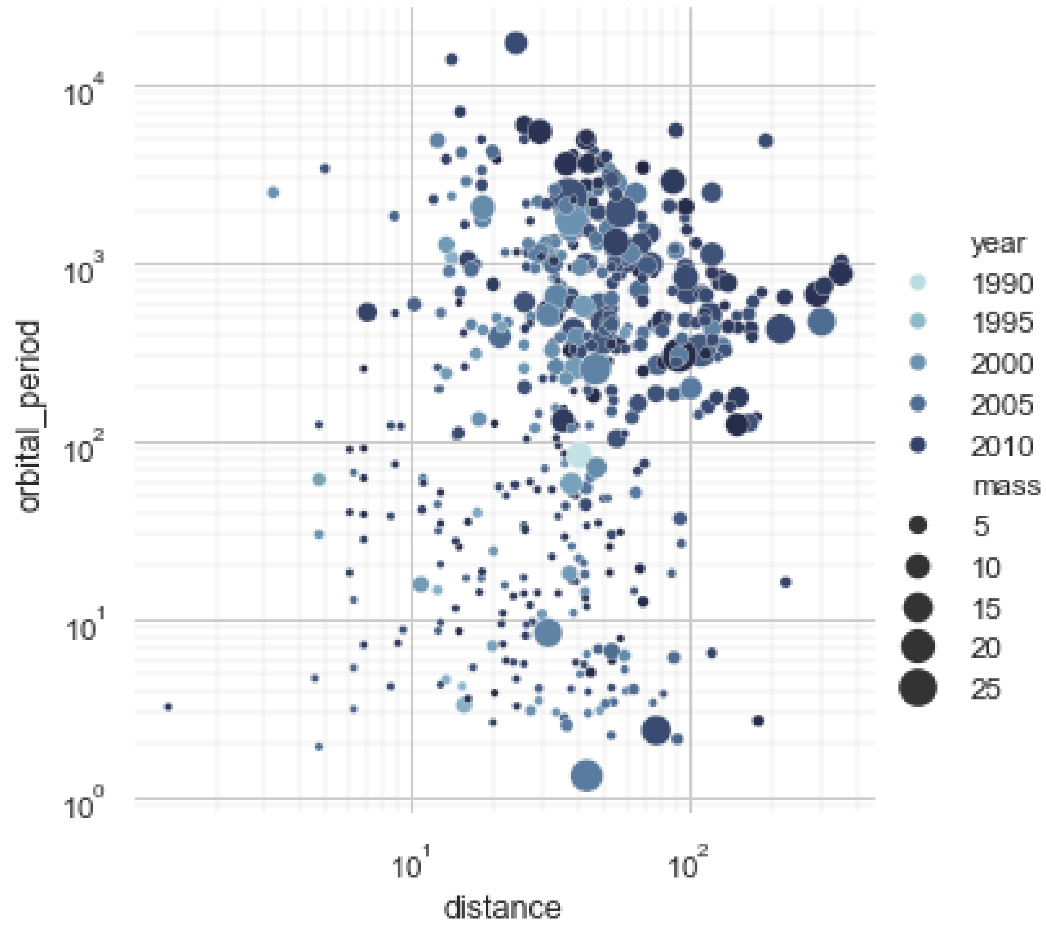


Table of Contents

Assignment 6: Tasks

This assignment is designed to jointly practice data aggregation and visualisation.
1) Load the contents of the Chinese lexical decision dataset by Tsang (2018), and of the affective ratings dataset by Mohammad (2018) into Pandas DataFrame objects.
2) Merge the information about Concepticon concepts for which there is data in both datasets into a single DataFrame object, discard all concepts for which either affective ratings data or lexical decision data is missing.
3) Group the data by the number of strokes in the Chinese characters (column `CHINESE_STROKE`), and compute the average arousal values (ENGLISH_AROUSAL_MEAN) and reaction times (column `CHINESE_RT_MEAN`) for each group of characters.
4) Use two-dimensional KDE plots to inspect the joint distributions of the following pairs of variables: strokes and reaction time, arousal and reaction time, strokes and arousal. Are there any interesting patterns.
5) Plot the join distribution of arousal values and reaction times for different numbers of strokes in the characters, e.g. by using hues in a scatter plot.
6) Come up with one additional visualisation which you suspect might give an interesting result, implement the visualisation, and comment whether your suspicion has been confirmed.

Sources

- most of this presentation was a summary of Chapter 36 in VanderPlas (2023): "Python Data Science Handbook, 2nd edition“, which in turn is mostly extracted from the documentation
- further examples were extracted directly from the Seaborn documentation

Preliminary Course Plan

1	**27/10**	**IPython and Jupyter**
2	**03/11**	**Introduction to NumPy**
3	**10/11**	**Pandas and Data Frames**
4	**17/11**	**Data Cleaning and Preparation**
5	**24/11**	**Linguistic Preprocessing**
6	**01/12**	**Data Wrangling**
7	**08/12**	**Data Aggregation and Grouping**
8	**15/12**	**Visualisation with Seaborn**
9	22/12	Modeling and Prediction
10	12/01	Classification
11	19/01	Clustering
12	26/01	Pattern Extraction and Density Estimation
13	02/02	Statistical Inference
14	09/02	Data Science Projects

Questions

Questions?

Comments?

Suggestions?