Конспект по теме "Библиотека seaborn"

Почему не хватает matplotlib?

Matplotlib — низкоуровневая библиотека для визуализации данных в *Python*, основа более продвинутых библиотек: например, *seaborn*. *Matplotlib* содержит в себе множество настроек, которые пригодятся при построении уникальных графиков. Однако аналитику чаще нужны стандартные. Строить их следует быстро, а значит написание длинного кода для каждого графика тут не подойдёт.

Ещё *matplotlib* ругают за то, что она «некрасивая». Улучшить визуальное восприятие графиков помогут два подхода:

- вызовите встроенные стили в *matplotlib*;
- импортируйте библиотеку, например, *seaborn*.

Чтобы узнать, какие стили доступны, примените метод **available** к пакету **style** :

```
import matplotlib.pyplot as plt
print(plt.style.available) # вызовем разные наборы цветов
```

Иногда к определённому стилю приводят только часть графиков проекта. Передайте контекстному менеджеру **with** название стиля в методе **context()**, а затем укажите область ограничения изменений:

```
with plt.style.context('seaborn-pastel'):
    plt.bar([10, 20, 30, 40],[3, 9, 18, 7])
```

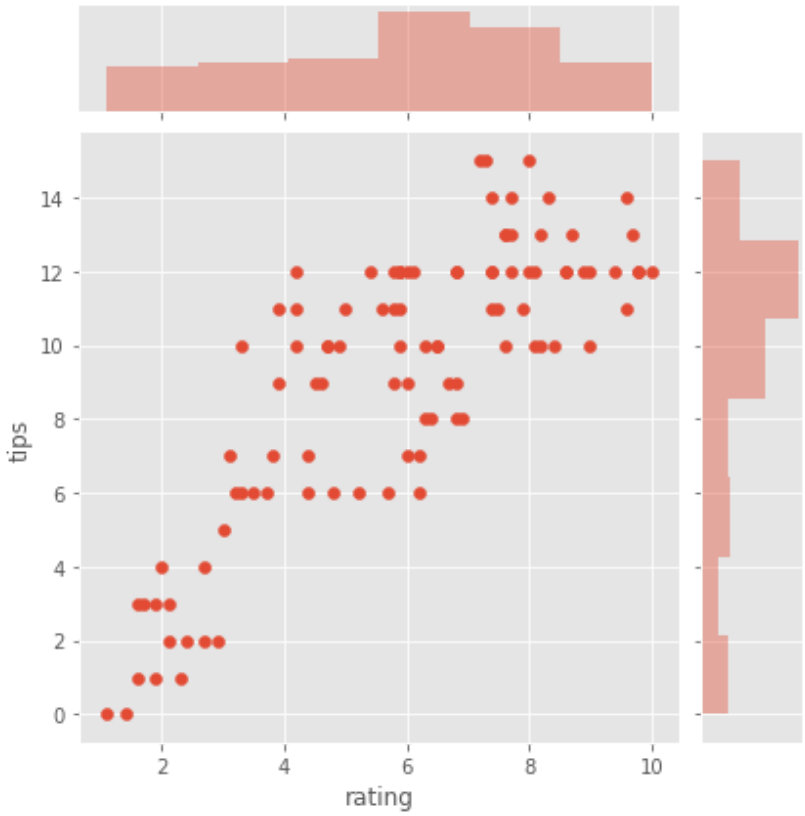
Если все графики должны быть оформлены в едином стиле, в начале проекта вызовите метод **plt.style.use()**:

```
plt.style.use('ggplot') # здесь выбран стиль ggplot
```

Метод jointplot()

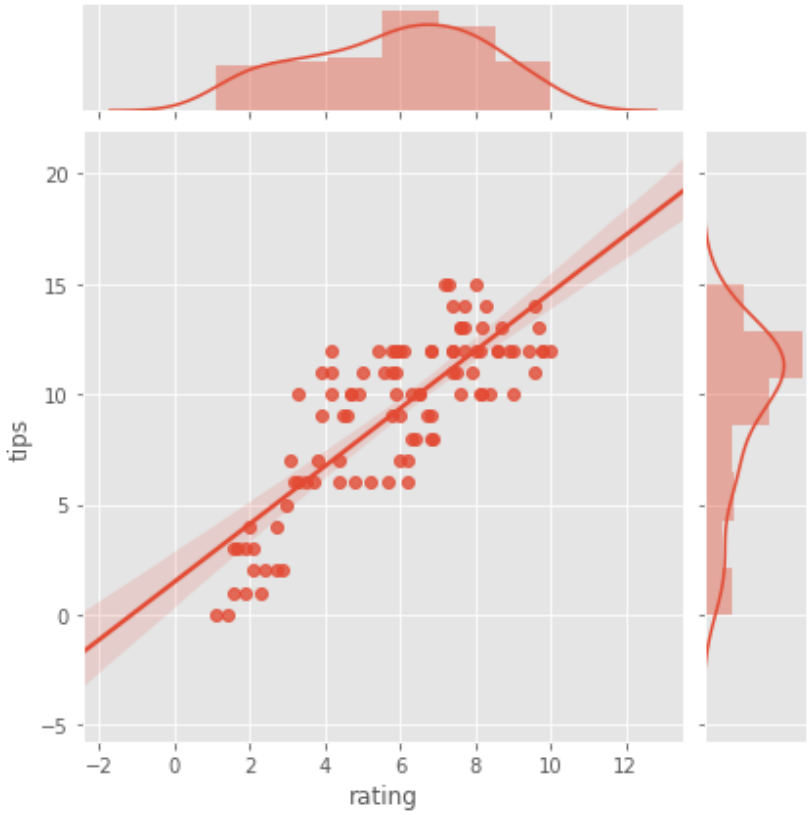
Joint plot из *seaborn* позволяет построить два распределения на одном графике. Наличие этого метода — одно из преимуществ *seaborn* перед *matplotlib*: всего одна строка кода, и визуализация совместного распределения готова!

```
import seaborn as sns
import pandas as pd

taxi = pd.read_csv('/datasets/taxi.csv')

sns.jointplot(x="rating", y="tips", data=taxi)
```



Добавим дополнительную информацию — плотность распределения и регрессию. Присвоим аргументу `kind` значение `"reg"`:

```
sns.jointplot(x="rating", y="tips", data=taxi, kind='reg')
```



Такой график стоит добавить в отчёт для коллег, руководству без дополнительных пояснений он будет не понятен.

Цветовые гаммы

Важное качество аналитика — чувство прекрасного. Даже правильные графики не будут иметь смысла, если никто не сможет их рассмотреть.

Онлайн-сервисы с подобранными палитрами помогут избежать таких проблем. Вам остаётся только задать нужные цвета для графика:

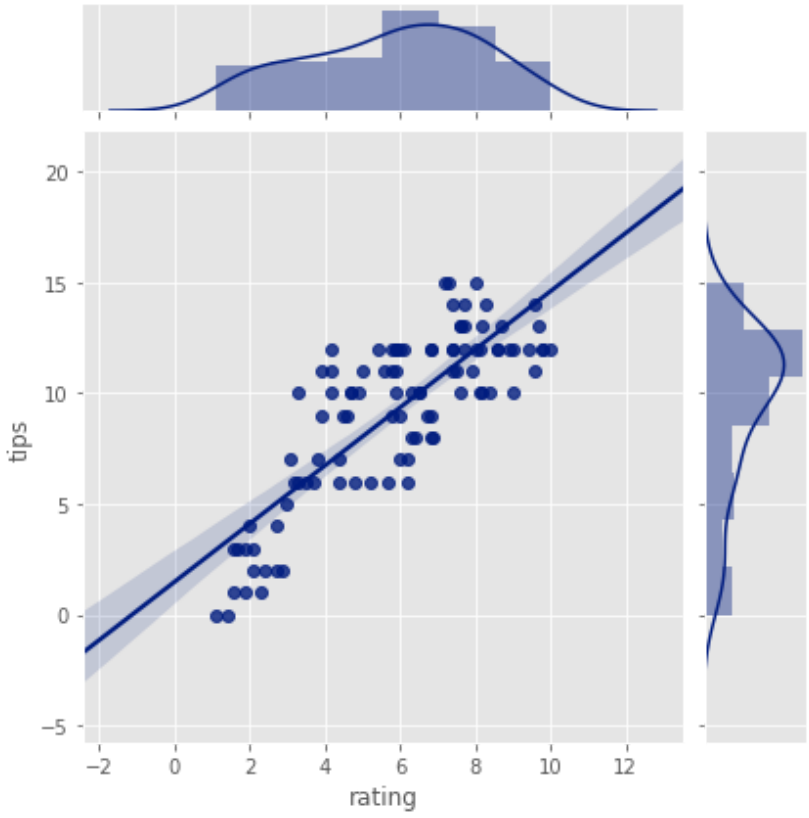
- https://colorhunt.co/
- http://fabianburghardt.de/swisscolors/
- https://flatuicolors.com/
- https://uxpro.cc/toolbox/visual-design/colors/

В seaborn реализована поддержка цветовых палитр. Доступ к ним получим методом **color_pallete()**:

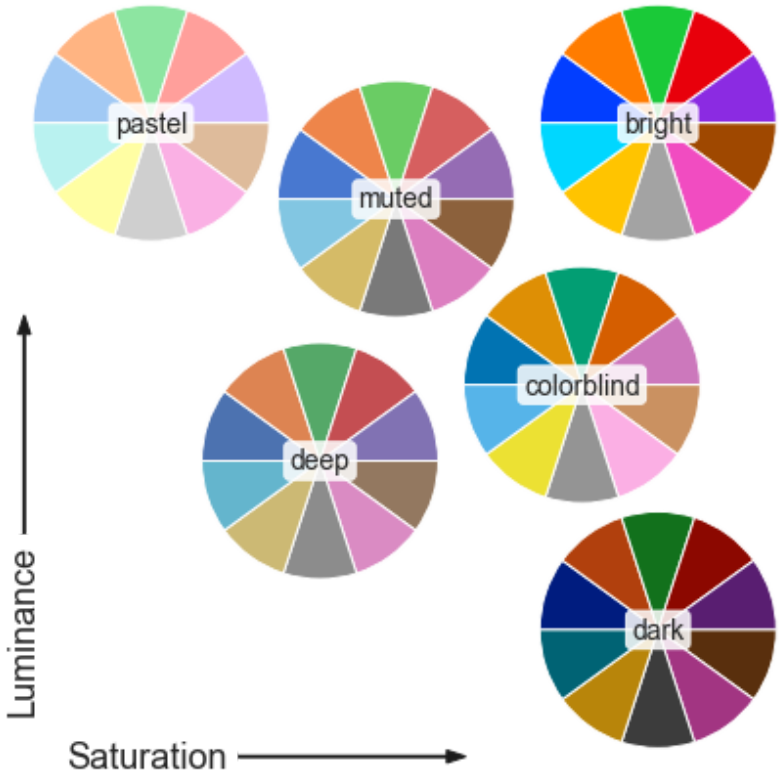
```
current_palette = sns.color_palette("coolwarm", 20)
print(sns.palplot(current_palette))
```

Стандартную палитру для всех графиков задают методом **set_palette()** :

```
sns.set_palette('dark')
```

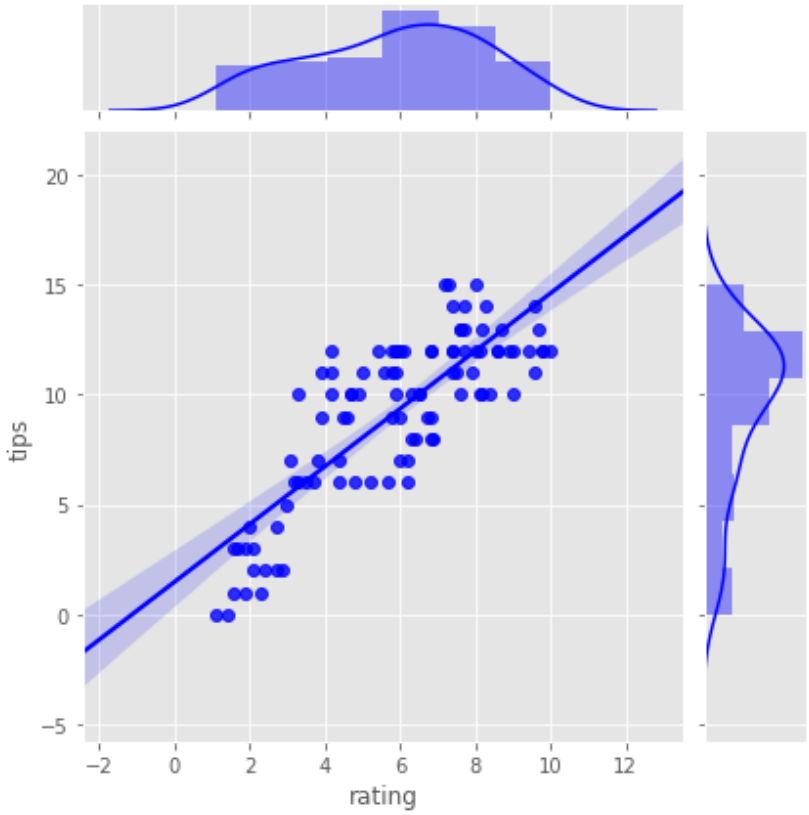


Познакомьтесь с видами палитр в документации. Стандартные изображены здесь:



Чтобы выбрать цвет для определённого графика, добавьте аргумент `color` и укажите название цвета. Изменим график из прошлого урока:

```
sns.jointplot(x="rating", y="tips", data=taxi, kind='reg', color='blue')
```



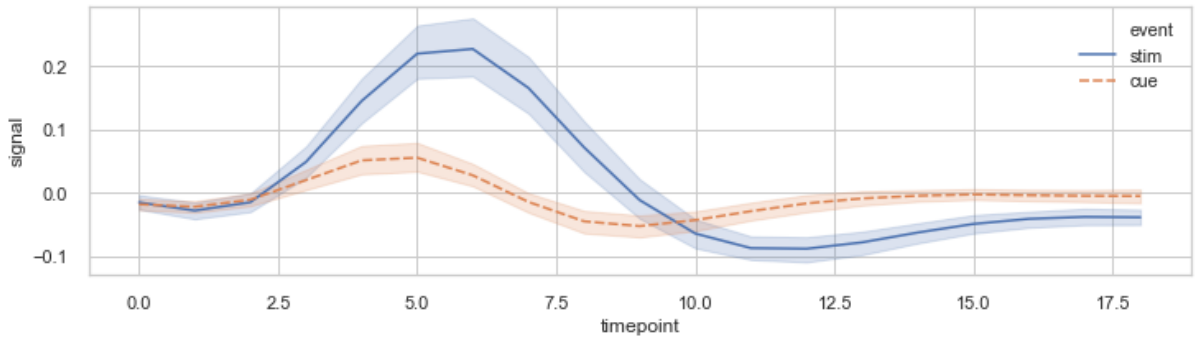
Стили графиков

Вот методы *matplotlib*, которые работают и для *seaborn*:

- **set_title()**;
- **set_xlabel()** и **set_ylabel()**.

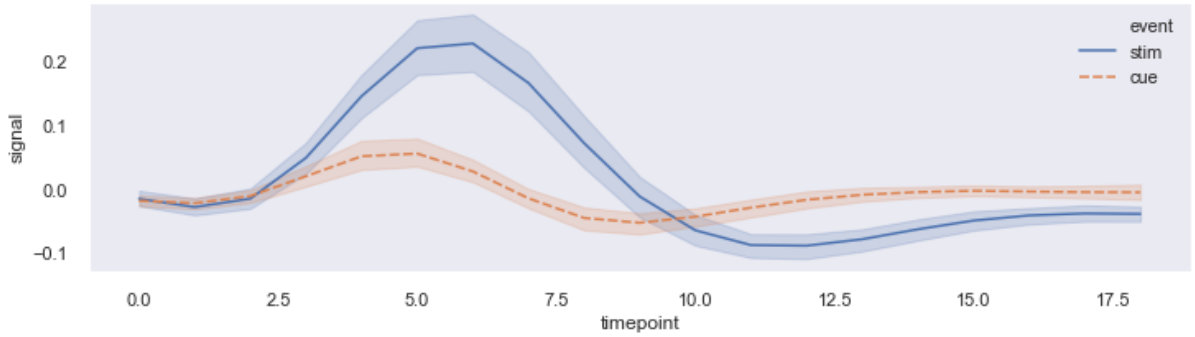
Размер графика изменяют методом `figure()` с аргументом `figuresize` :

```
plt.figure(figsize=(12, 3)) # Важно! Этот код нужно писать до момента создания графика
ax = sns.lineplot(x="timepoint", y="signal", hue="event", style="event", data=fmri)
```



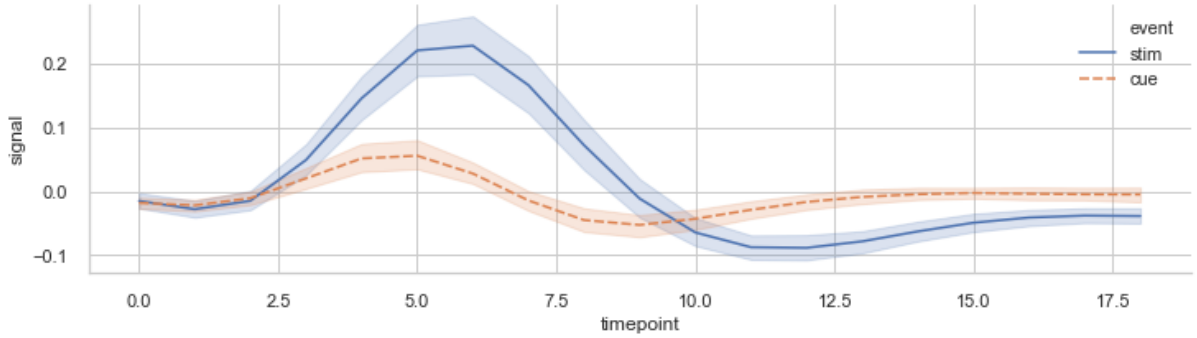
Стиль графика задают методом `set_style()` . Аргументом здесь будет одна из пяти тем: `'darkgrid'`, `'whitegrid'`, `'dark'`, `'white'` или `'ticks'`. По умолчанию установлена тема `'darkgrid'`. Если вам не нужна сетка на графике, подключите темы `'dark'`, `'white'` или `'ticks'`. Тема `'whitegrid'` подойдёт для сложных графиков.

```
sns.set_style("dark")
```



Графики обычно строят на двух осях: X и Y. Чтобы оставить только оси видимыми, применяют метод `despine()`. Такое отображение наглядное и хорошо подходит для презентации.

```
sns.despine()
```



Категориальные данные

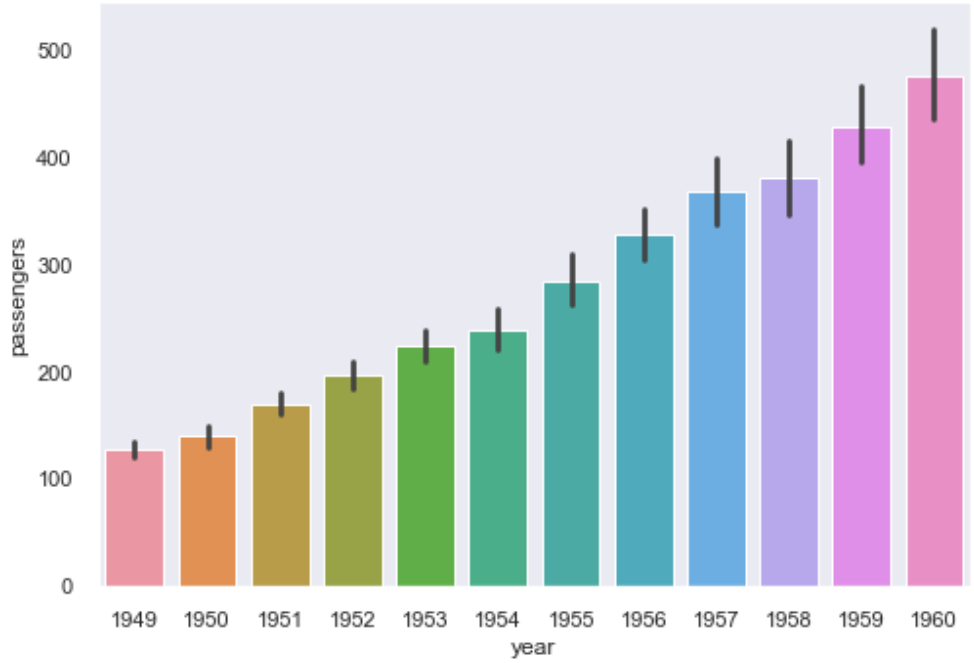
В *seaborn* есть возможность протестировать графики на встроенных наборах данных. Подробнее познакомиться с ними можно здесь. Получим доступ к встроенным наборам данных методом **load_dataset()**:

```
import seaborn as sns
iris = sns.load_dataset("iris")
print(iris.head())
```

Вы знакомы с *barplot* — это столбчатая гистограмма. График строят методом `barplot()` с аргументами:

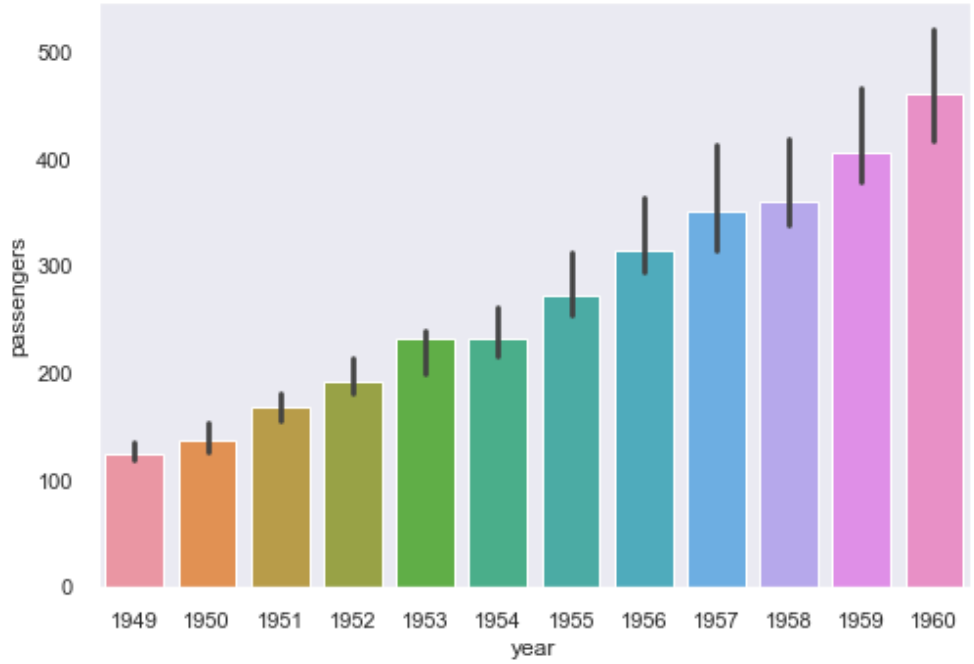
- `x` — данные по оси X;
- `y` — данные по оси Y;
- `data` — набор данных, по которому строят график;
- `color` или `palette` — цвет или палитра.

```
import seaborn as sns
ax = sns.barplot(x="year", y="passengers", data=flights)
```



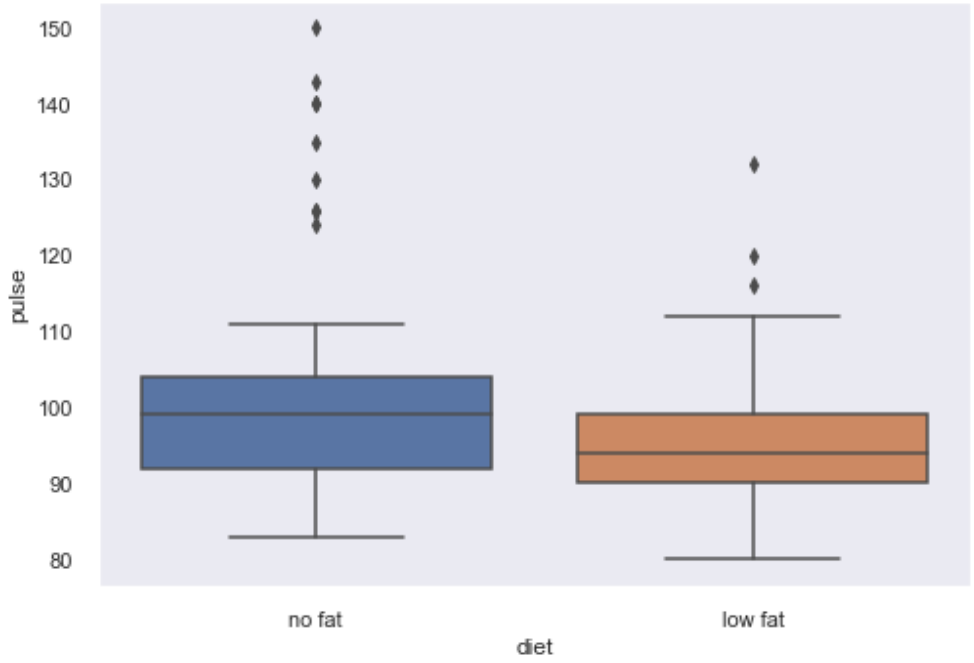
Barplot() самостоятельно агрегирует данные: по умолчанию считает среднее. Эту опцию изменяют в аргументе **estimator**:

```
import seaborn as sns
from numpy import median
flights = sns.load_dataset("flights")
ax = sns.barplot(x="year", y="passengers", data=flights, estimator=median)
```



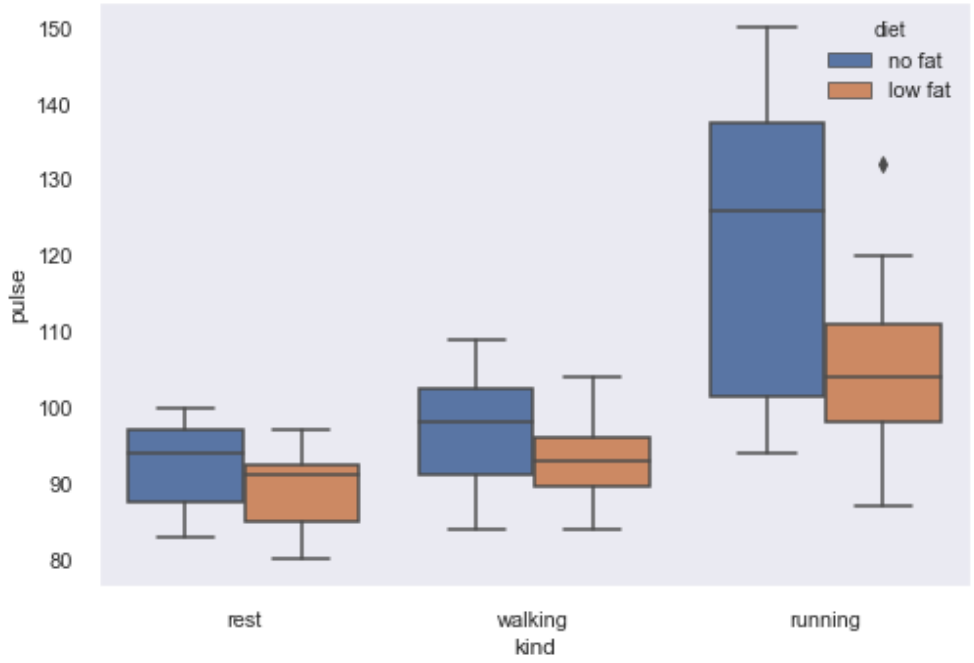
В курсе «Исследовательский анализ данных» вы познакомились с графиком *boxplot*, или «ящик с усами». В *seaborn* его строят методом метод `boxplot()`:

```
import seaborn as sns

sport = sns.load_dataset("exercise")
ax = sns.boxplot(x="diet", y="pulse", data=sport)
```



Помимо осей X и Y, можно добавить графику третье измерение. Например, рассмотреть приверженцев разных диет не только с точки зрения пульса, но и физической активности. Параметру **hue** метода `boxplot()` передают столбец, из которого следует взять информацию:

```
import seaborn as sns
from numpy import median
sport = sns.load_dataset("exercise")
ax = sns.boxplot(x="kind", y="pulse", hue="diet", data=sport)
```

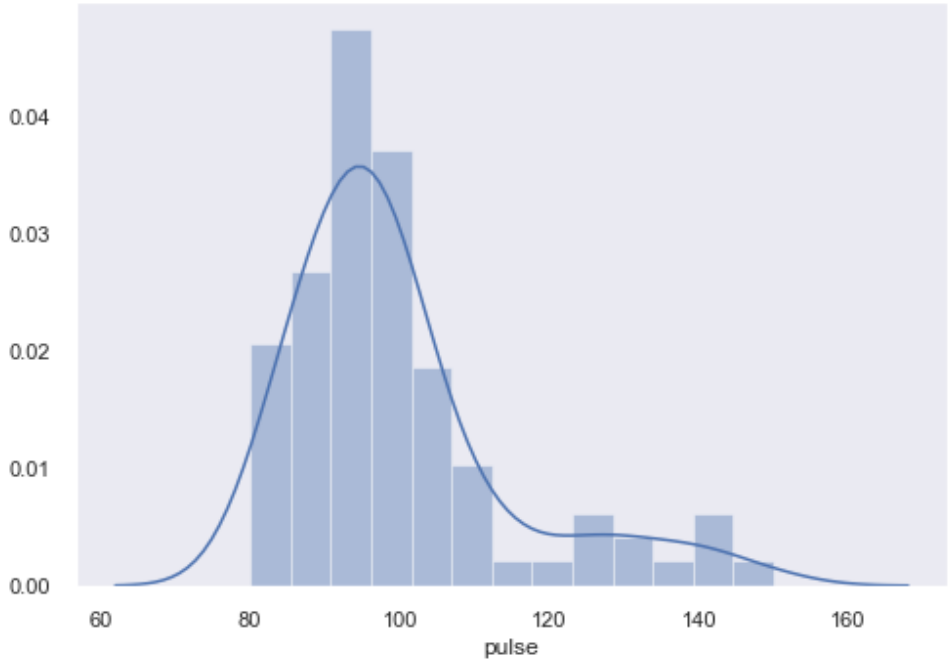


Визуализация распределения

Раньше вы строили распределения методом `bar()`. В *seaborn* есть готовые графики для визуализации распределения одной переменной и совместного распределения.

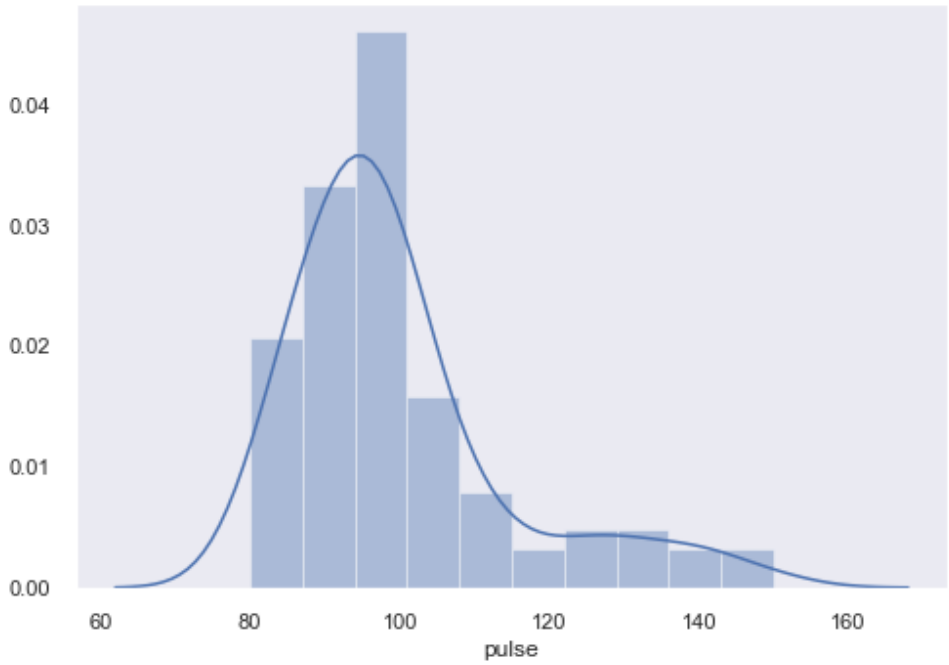
Метод **distplot()** показывает распределение величины, его плотность и сочетает гистограмму с линейным графиком:

```
import seaborn as sns
sport = sns.load_dataset("exercise")
sns.distplot(sport['pulse'])
```



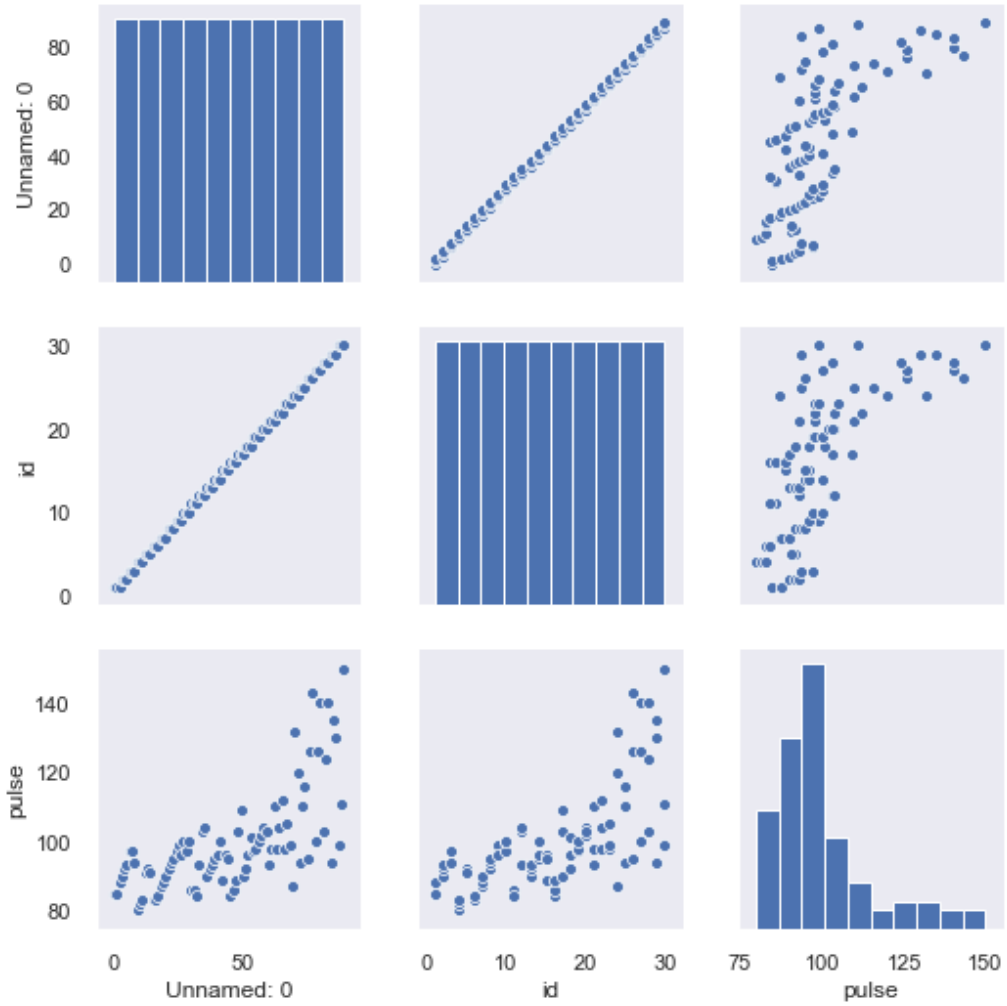
Как и в графике `bar()` корзины задают в аргументе `bins` :

```
sns.distplot(sport['pulse'], bins=10)
```



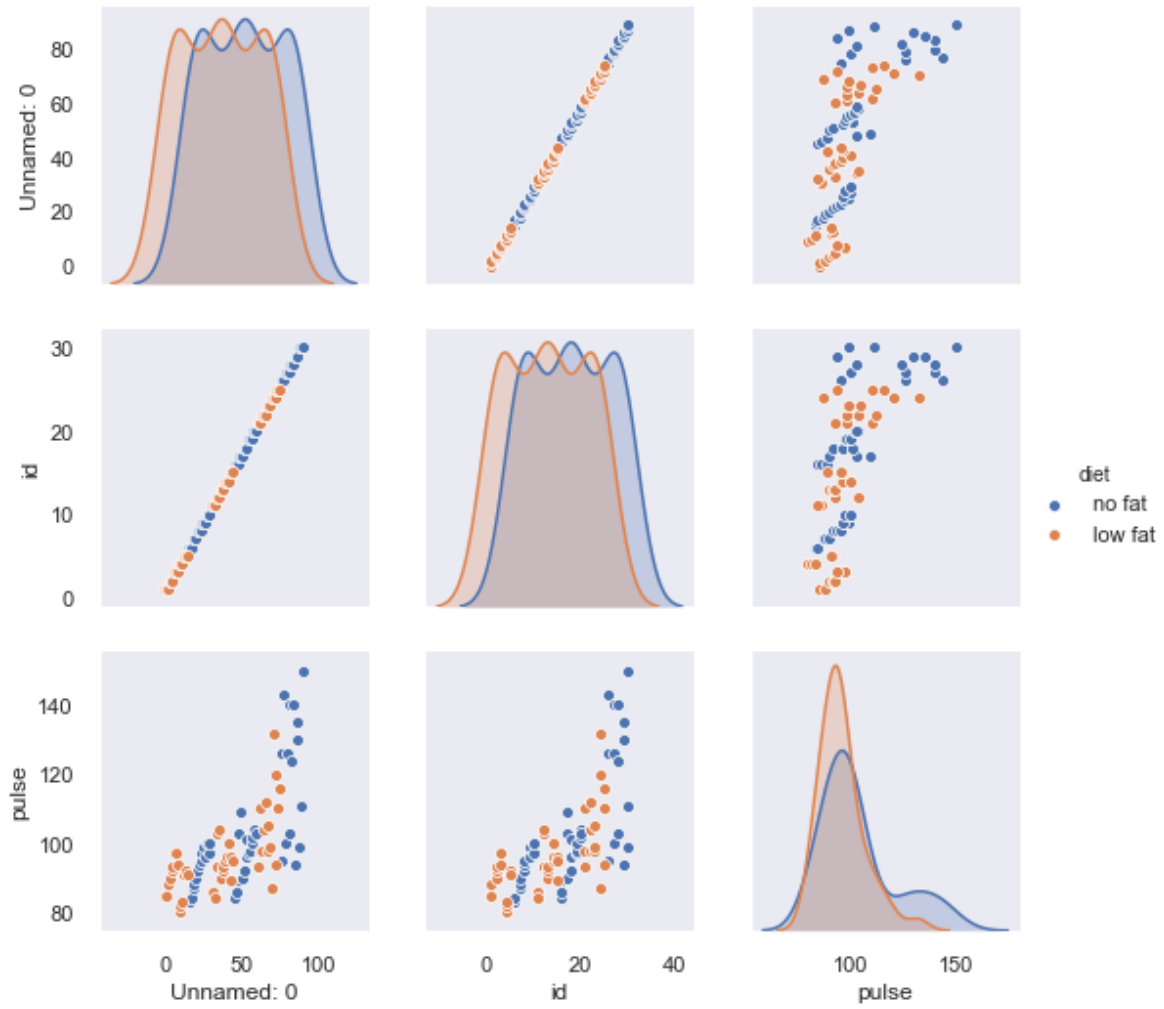
Методом `pairplot()` строят график совместного распределения:

```
sns.pairplot(sport)
```



Доступно и третье измерение. Его объявляют в аргументе `hue`:

```
sns.pairplot(sport, hue='diet')
```



Нестандартные графики в seaborn

violinplot()

Как и *boxplot()*, этот график характеризует форму распределения. Необычный внешний вид получается из-за сложения двух графиков плотности распределения. Основное преимущество перед *boxplot()* — возможность изучить распределение и определить его тип. Так *violinplot()* соответствует *boxplot()*:

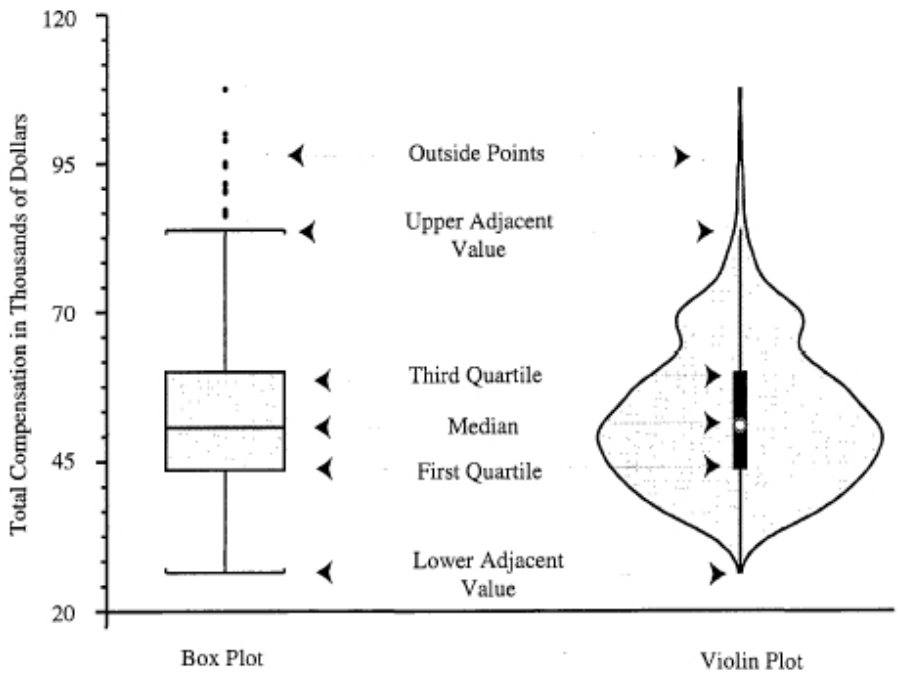
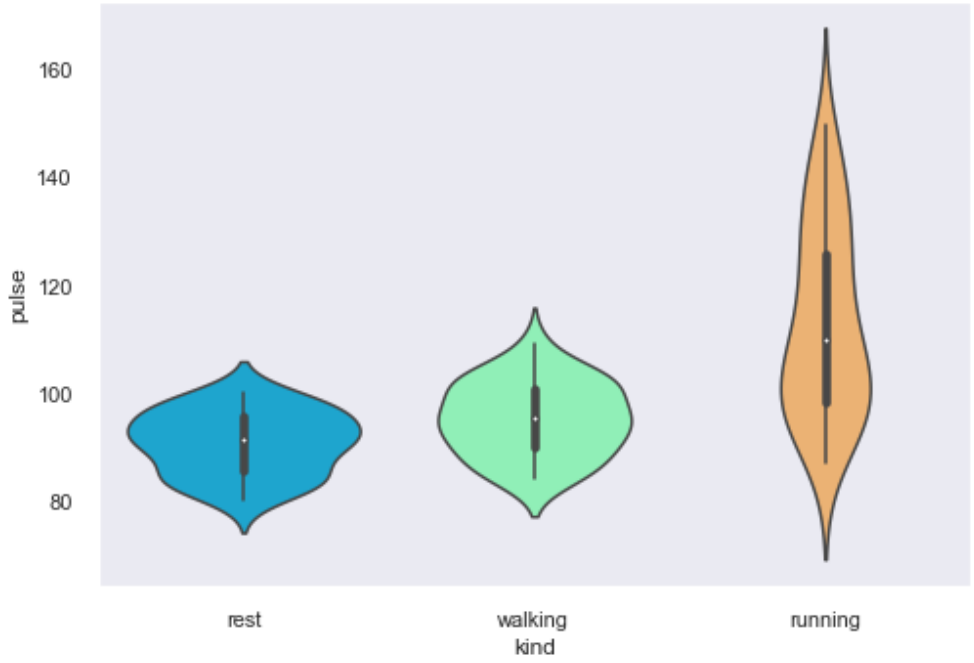


Figure 1. Common Components of Box Plot and Violin Plot. Total compensation for all academic ranks.

В *seaborn* такой график строят методом `violinplot()`:

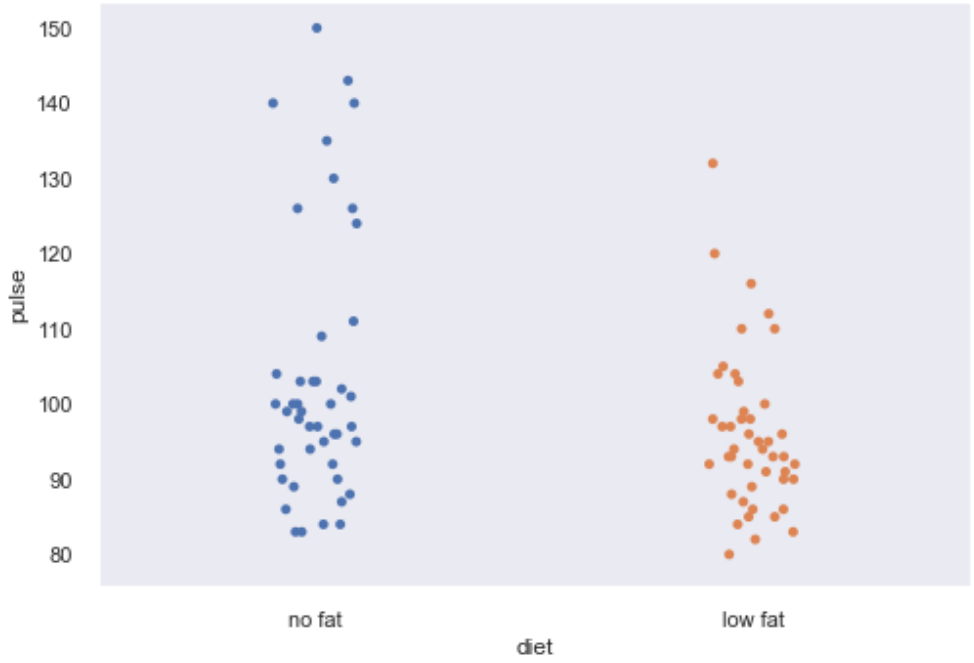
```
sns.violinplot(x="kind", y="pulse", data=sport, palette='rainbow')
```



stripplot()

stripplot() — ещё один способ отображения категориальных данных. Для каждой категории получаем диаграмму рассеяния. График рекомендуют строить в сочетании с другими, например, с *violinplot()*. В *seaborn* такой график строят методом `stripplot()`:

```
sns.stripplot(x="diet", y="pulse", data=sport)
```


150
140
130
120
110
100
90
80
pulse
no fat
low fat
diet