KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 3 урок > PYTHON

> Оглавление 3 урока

14. seaborn
15. matplotlib

> Уникальные значения

`unique` — метод, возвращающий уникальные значения в колонке.

```
data.fam_sp.unique()
```

```
array(['PASTA ALIMENTICIA SE'], dtype=object)
```

Уникальные значения возвращаются в форме array — о них будет сказано попозже, для простоты воспринимайте их как списки.

Документация

> Число уникальных значений

`nunique` — метод, который считает число уникальных значений в колонке.

```
data.fam_sp.nunique()
```

```
1
```

Документация

> Медиана

Чтобы посчитать медиану колонки, используйте метод `median`:

```
users_data.orders.median()
```

```
4.0
```

Документация

Среднее

Для среднего — `mean`:

```
users_data.orders.mean()
```

```
5.487290227048371
```

Документация

> split

`split` — метод, разбивающий строку на куски и помещающий фрагменты в список. По умолчанию делит по пустым символам (пробел, табы, перенос строки).

```
brand_info = 'MARAVILLA 500 G Store_Brand'
brand_info.split()
```

```
['MARAVILLA', '500', 'G', 'Store_Brand']
```

Больше информации

> Анонимные функции

Обычно используются, когда нужно куда-то быстро поместить нечасто используемый функционал. Если вы планируете использовать анонимную функцию больше одного раза, напишите обычную функцию :)

```
lambda x: do something
```

- `lambda` — ключевое слово, задающее анонимную функцию (не имеющую имени)

- `x` — то, как мы назвали аргумент, принимаемый функцией
- `:` — разделяет заголовок и тело безымянной функции
- `do something` — тело функции, должно помещаться в одну строчку и будет автоматически возвращаться без `return`

```
# Take 1 argument and add 3 to itlambda x: x + 3
```

Один из примеров использования лямбда-функции — переименование колонок в датафрэйме. Здесь мы делаем их заглавными и заменяем дефисы на нижние подчёркивания:

```
# df is a dataframe as usual
df = df.rename(columns=lambda c: c.upper().replace('-', '_'))
```

Больше информации

> Серии

`pd.Series` — более примитивный тип данных в `pandas`, соответствует колонке датафрейма. В ней хранятся данные одного типа (числа/строки/...). Работая с колонкой, мы работаем именно с серией. Часть методов и атрибутов серии и датафрейма совпадают.

Документация

> Применение функций к датафрейму

`apply` — применяет переданную в него функцию ко всем колонкам вызванного датафрейма. Чтобы применить функцию к одной колонке датафрейма, можно выбрать её перед применением `apply`, например:

```
data.art_sp.apply(lambda c: c.split()[-1])
```

```
0        Store_Brand
1        Store_Brand
2            Brand_1
```

Документация

> Объединение датафреймов

Зачастую называется джойном. Очень частая операция, которую можно сделать с помощью нескольких функций. Одна из них — `merge`. Обязательным аргументом является другой датафрейм, с которым планируется объединение. Объединение идёт по общей колонке, у которой имеется одинаковый смысл и общие значения в обоих датафреймах. Существуют различные типы джойнов, они будут рассмотрены в курсе по SQL. Самый частый, пожалуй, `inner`.

Здесь мы объединяем датафрейм `users_data` с датафреймом `users_lovely_brand_data` по колонке `tc` с помощью `inner` джойна:

```
users_data.merge(users_lovely_brand_data, how='inner', on='tc')
```

	tc	orders	unique_brands	lovely_brand	max_orders
0	1031	6	2	Store_Brand	5
1	4241	5	2	Brand_4	3
2	17311	2	1	Brand_4	2
3	17312	2	2	Brand_1	1

В результате получается один датафрейм, где колонки из двух таблиц, относящиеся к одному наблюдению, объединяются в строку. Звучит сложно, поэтому для практики стоит попробовать сделать несколько простых джойнов :)

Дополнительные аргументы функции merge

- `how` — как объединять датафрэймы, одно из inner, outer, left, right
- `on` — общая колонка, по которой будет идти объединение.

Документация

> Индекс и имена колонок

Индекс — это лейбл строки в таблицы, по умолчанию является её номером. А имена колонок — лейблы, по которым мы можем обращаться к каждому из столбцов.

У датафрейма есть 2 атрибута `index` и `columns`, позволяющие получить доступ к соответствующей информации в виде array (на самом деле, не совсем array)

	journey_id	driver_id
easy	135	99
executive	22737	19484
group	239	143

```
df.index
Index(['easy', 'executive', 'group'], dtype='object')
```

```
df.columns
Index(['journey_id', 'driver_id'], dtype='object')
```

Документация

> Сброс индекса

Иногда вам может захотеться перевести индекс датафрэйма в колонку. Для этого существует метод `reset_index`. Индексом становится дефолтная последовательность чисел от 0 до числа строк - 1.

```
df
```

event date_day	click	view
2019-04-01	881	41857
2019-04-02	1612	165174
2019-04-03	1733	224843
2019-04-04	1447	107098
2019-04-05	581790	2050279

```
df.reset_index()
```

	event	date_day	click	view
0		2019-04-01	881	41857
1		2019-04-02	1612	165174
2		2019-04-03	1733	224843
3		2019-04-04	1447	107098
4		2019-04-05	581790	2050279

Удаление индекса

Аргумент `drop` отвечает за то, нужно ли переводить индекс в колонку, или убрать его из таблицы:

```
df.reset_index(drop=True)
```

event	click	view
0	881	41857
1	1612	165174
2	1733	224843
3	1447	107098
4	581790	2050279

Документация

> Поиск пустых значений

`isna` — это чудо-метод, с помощью которого можно быстро найти пропущенные значения в датафрейме:

```
df.head()
```

	Id	sum
0	1	150.0
1	2	230.0
2	3	NaN
3	4	143.0
4	5	NaN

Применив его, на выходе мы получаем датафрэйм той же размерности, где в каждой ячейке `True` или `False` — в зависимости от того, было ли значение пропущено:

```
df.isna()
```

	id	sum
0	False	False
1	False	False
2	False	True
3	False	False
4	False	True

В связке с ним можно использовать, например, `sum`, чтобы посмотреть на число `NA` в разных колонках:

```
df.isna().sum()
```

```
id      0
sum     4
dtype: int64
```

Документация

> Графики

Графики — важная часть анализа данных, так как они наглядно — если тип графика хорошо подобран — представляют данные и позволяют быстро разобраться в сути.

Чтобы в юпитер ноутбуке отображались графики, выполните строчку:

```
%matplotlib inline
```

Существуют разные способы создания графиков в python, несколько популярных библиотек:

- `pandas`
- `seaborn`
- `matplotlib`

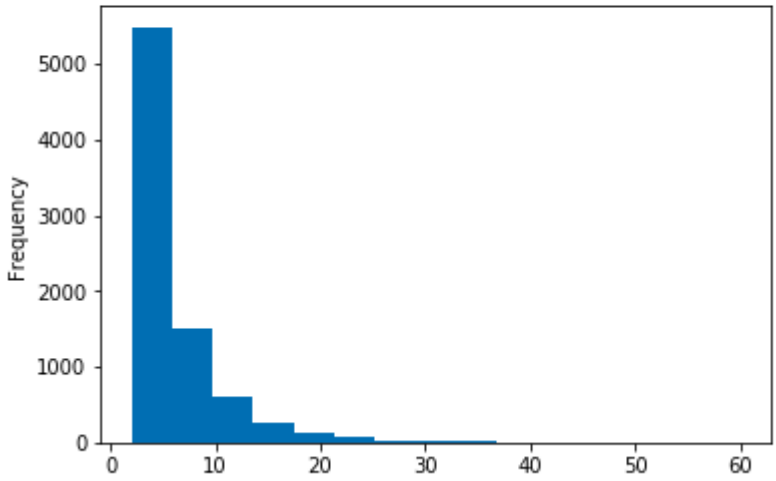
Давайте начнём постепенно в них разбираться!

> pandas

Самый простой способ визуализировать данные — вызвать метод `plot` у датафрейма (или его колонки). Например, гистограмма значений в колонке `orders` :

```
users_data.orders.plot('hist', bins=15)
```

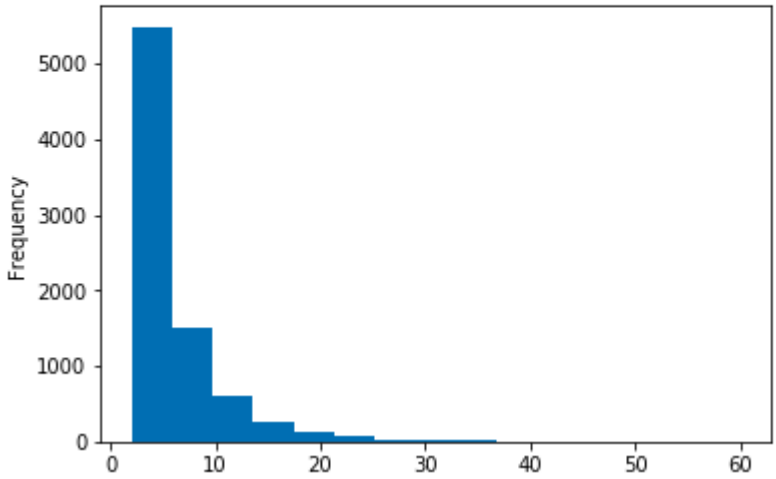
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d35869e8>
```



Другой вариант записи:

```
users_data.orders.plot.hist(bins=15)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0f0dd1ef0>
```



Функции рисования имеют весьма большое количество параметров, используйте их при необходимости. `bins` здесь — число диапазонов (корзин/бакетов), на которые мы разделяем значения.

Документация

> seaborn

Продвинутая библиотека, позволяющая сделать очень красивые графики. Конвенционально загружается как:

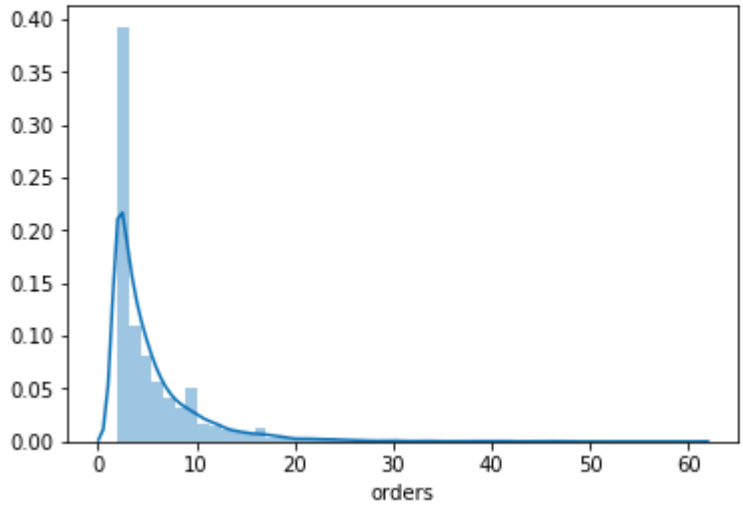
```
import seaborn as sns
```

Далее пример создания нескольких графиков с её помощью.

Гистограмма

```
sns.distplot(users_data.orders)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d9315d68>
```

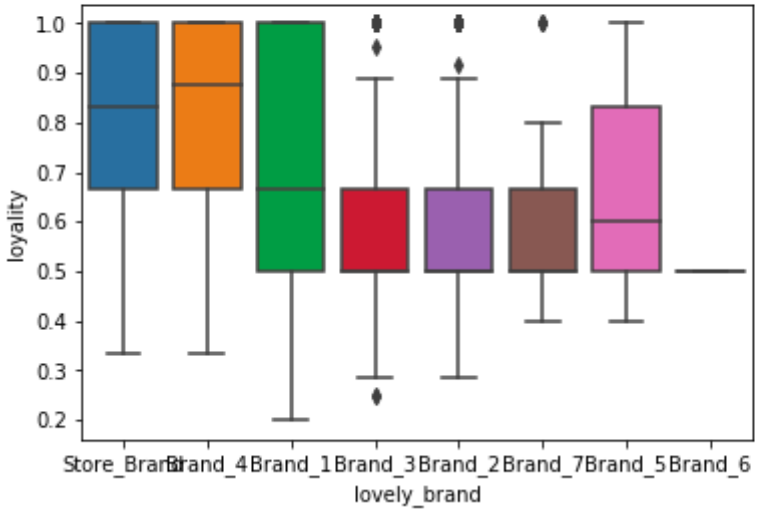


Документация

Боксплот

```
sns.boxplot(data=users_data, x='lovely_brand', y='loyality')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d8ead048>
```

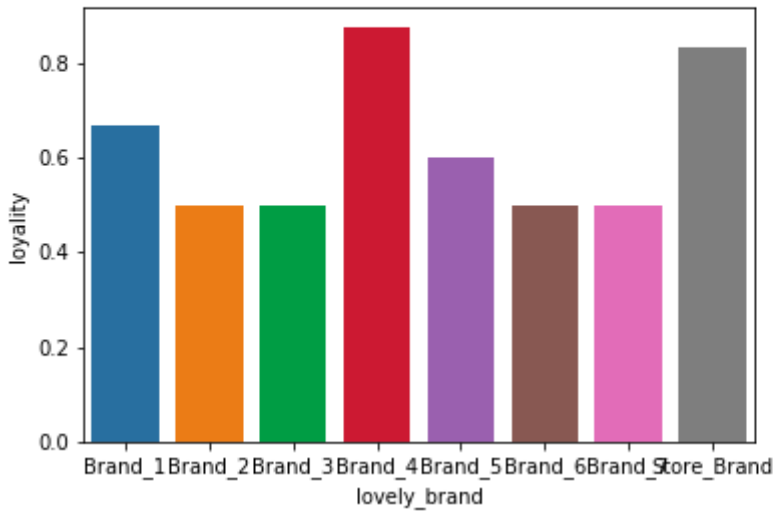


Документация

Барплот

```
sns.barplot(x='lovely_brand', y='loyality', data=loyality_stat)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d8a28f28>
```



На графиках перекрываются подписи — скоро мы разберём, как это поправить.

Документация

> matplotlib

Базовая библиотека для рисования в питоне. На ней построены более продвинутые и простые в использовании типа `seaborn`.
Через `matplotlib` можно нарисовать что угодно, но часто на это уходит слишком много строк кода, и её в основном используют для тонкой настройки графиков и их сохранения.

Традиционно `matplotlib` импортируется следующим образом:

```
import matplotlib.pyplot as plt
```

Настройка графиков

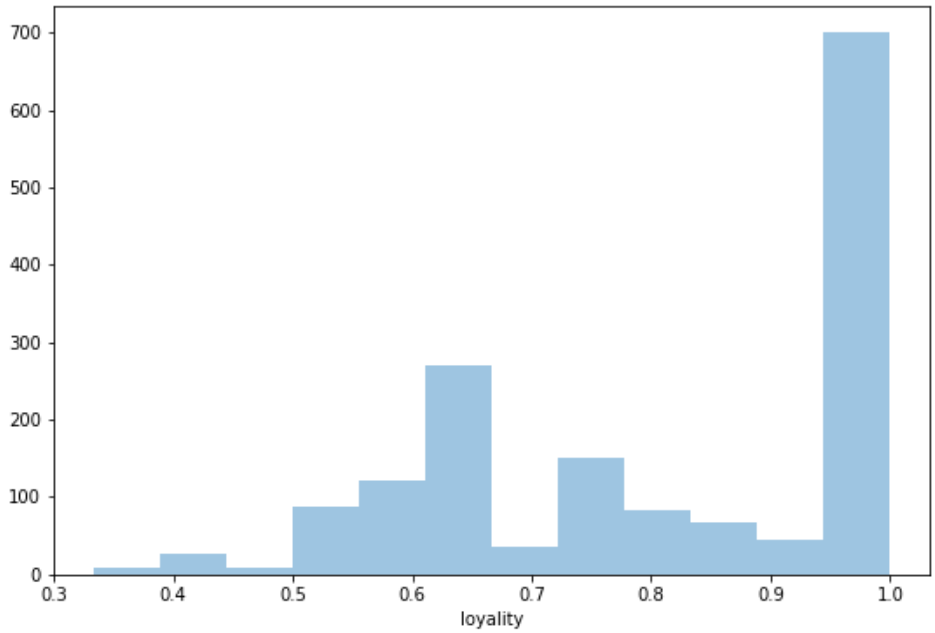
Важный момент: большинство настроек должны быть написаны к каждому графику отдельно. Иными словами, настройки, написанные в ячейке с одним графиком, не будут применены к другому.

Изменить размер

В `figure` в `figsize` подаётся кортеж (как список, только в круглых скобках) с масштабом графика формата `(ширина, высота)`

```
plt.figure(figsize=(9, 6))
sns.distplot(users_data.query('lovely_brand == "Store_Brand"').loyality, kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d3514240>
```



Больше информации

Сохранение картинки

Сохранить график можно с помощью `savefig` , где аргумент — путь к сохраняемой картинке (желаемое название и формат):

```
sns.distplot(users_data.query('lovely_brand == "Store_Brand"').loyality, kde=False)
plt.savefig('1.jpg')
```

Документация