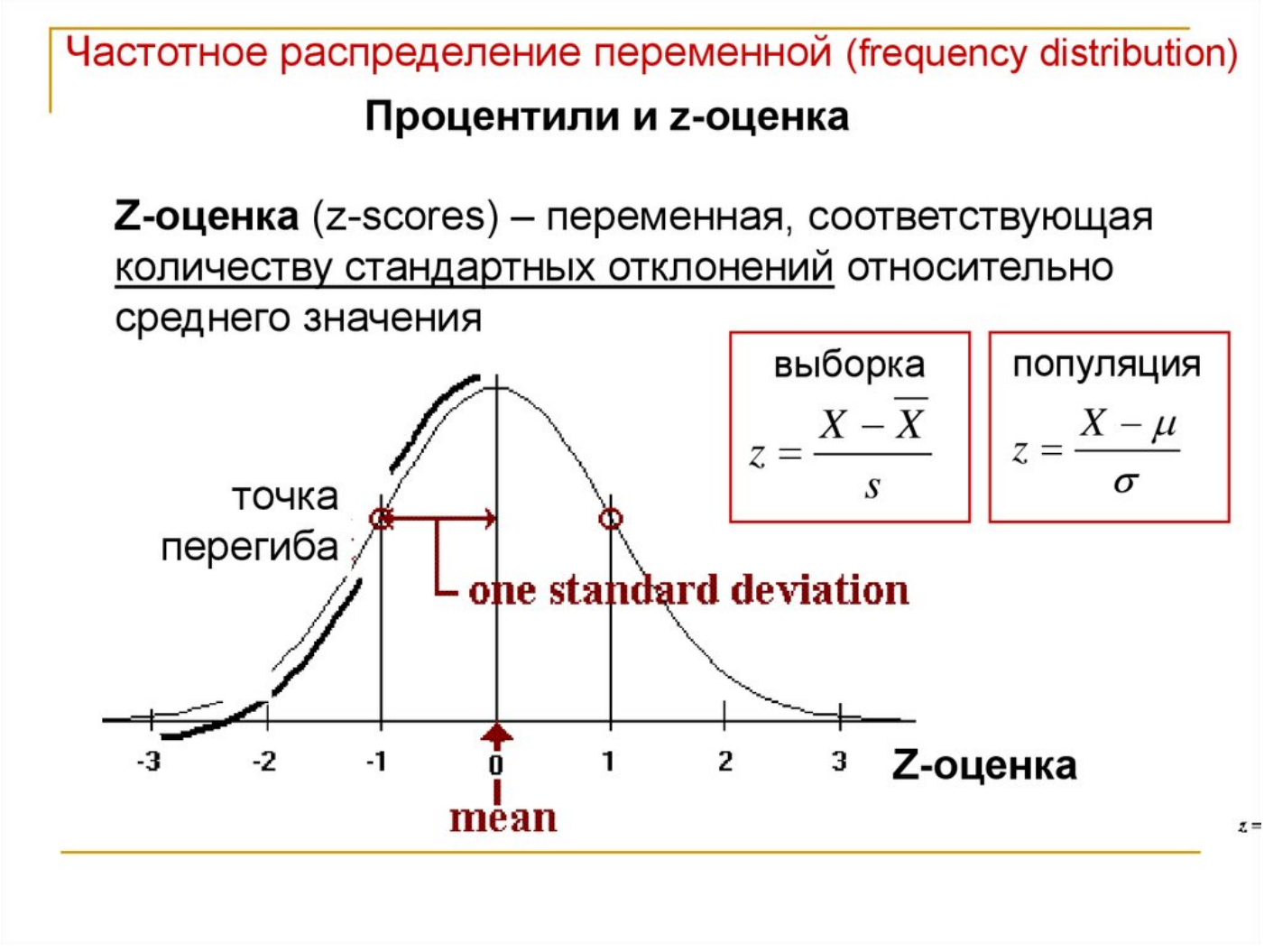
Быковская Арина Александровна

БВТ2002

Стандартизированная оценка (z-оценка) — это относительная мера, которая показывает, на сколько среднеквадратичных отклонений наблюдаемое значение отличается от среднего значения распределения. Знак z-оценки показывает, находится ли значение левее среднего (–) или правее среднего (+).



Выброс — это аномальное значение в данных, которое значительно отличается от значения, выраженного мерой центральной тенденции.

Z-оценка

Ввод [2]:

```
import pandas as pd
import scipy.stats
```

Загрузите датасет eng_test.csv. Обратите внимание на сепаратор (sep=';'). Выведите первые 5 элементов

Ввод [3]:

```
data = pd.read_csv('./eng_test.csv', sep=';')
data.head(5)
```

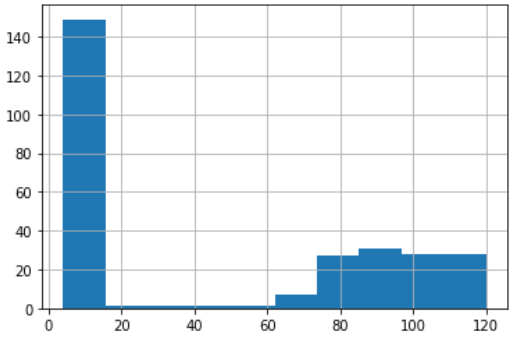
Out[3]:

	Id	Exam	Score	Advanced
0	ID1	TOEFL	77.0	NO
1	ID10	TOEFL	105.0	NO
2	ID100	TOEFL	107.0	YES
3	ID101	TOEFL	72.0	NO
4	ID102	TOEFL	120.0	YES

Постойте гистограмму по оценкам студентов

Ввод [4]:

```
data['Score'].hist();
```



Создайте переменную, содержащую информацию только об оценках TOEFL и выведите гисограмму и основные статистики

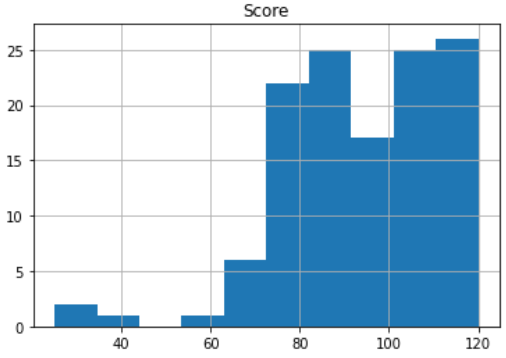
Ввод [5]:

```
toefl = data[data['Exam'] == 'TOEFL']

toefl.hist();
toefl.head()
```

Out[5]:

	Id	Exam	Score	Advanced
0	ID1	TOEFL	77.0	NO
1	ID10	TOEFL	105.0	NO
2	ID100	TOEFL	107.0	YES
3	ID101	TOEFL	72.0	NO
4	ID102	TOEFL	120.0	YES



Создайте переменную, содержащую информацию только об оценках IELTS и выведите гисограмму и основные статистики

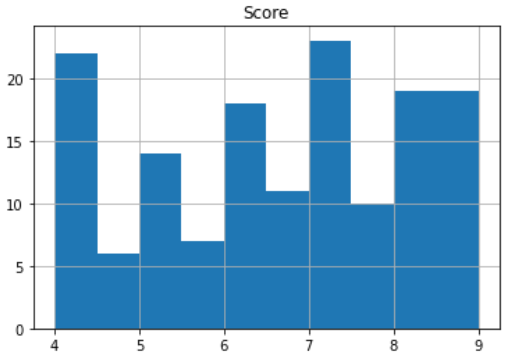
Ввод [6]:

```
ielts = data[data['Exam'] == 'IELTS']

ielts.hist();
ielts.head()
```

Out[6]:

	Id	Exam	Score	Advanced
30	ID126	IELTS	9.0	YES
31	ID127	IELTS	5.0	NO
32	ID128	IELTS	7.5	YES
33	ID129	IELTS	5.0	NO
35	ID130	IELTS	4.0	NO



Загрузите датасет eng_test.csv. Обратите внимание на сепаратор (sep=';')

Ввод [7]:

```
data = pd.read_csv('./eng_test.csv', sep=';')
data.head(5)
```

Out[7]:

	Id	Exam	Score	Advanced
0	ID1	TOEFL	77.0	NO
1	ID10	TOEFL	105.0	NO
2	ID100	TOEFL	107.0	YES
3	ID101	TOEFL	72.0	NO
4	ID102	TOEFL	120.0	YES

Посчитайте z-score для первого студента в списке toefl. Также выведите стандартное отклонение, среднее и само кол-во баллов.

Ввод [8]:

```
print(round(scipy.stats.zscore(toefl['Score'])[0], 3)) # Z-score через библиотеку scipy.stats
print(toefl['Score'].std())
print(toefl['Score'].mean())
print(toefl['Score'].count())
```

```
-0.941
17.90380388488399
93.776
125
```

сохраните в переменные Z-score для ielts и toefl

Ввод [9]:

```
toeflz = scipy.stats.zscore(toefl['Score'])
ieltsz = scipy.stats.zscore(ielts['Score'])
```

Ввод [10]:

```
eng = pd.concat([toeflz, ieltsz]) #собираем результаты обратно в 1 датафрейм
data.insert(data.shape[1], 'z-score', eng)
data
```

Out[10]:

	Id	Exam	Score	Advanced	z-score
0	ID1	TOEFL	77.0	NO	-0.940778
1	ID10	TOEFL	105.0	NO	0.629429
2	ID100	TOEFL	107.0	YES	0.741586
3	ID101	TOEFL	72.0	NO	-1.221172
4	ID102	TOEFL	120.0	YES	1.470611
...	...	...	...	...	...
269	ID95	TOEFL	88.0	YES	-0.323911
270	ID96	TOEFL	90.0	YES	-0.211754
271	ID97	TOEFL	90.0	YES	-0.211754
272	ID98	TOEFL	54.0	NO	-2.230591

Ввод [11]:

```
data[data['z-score'] < -3] # кто написал экзамен хуже, чем 3 стандартных отклонения
```

Out[11]:

	Id	Exam	Score	Advanced	z-score
25	ID121	TOEFL	32.0	NO	-3.464325
27	ID123	TOEFL	25.0	YES	-3.856877

Ввод [12]:

```
data.groupby('Advanced')['z-score'].mean() #кто сдал экзамен лучше? Кто брал продвинутый курс или нет?
```

Out[12]:

```
Advanced
NO    -0.397672
YES    0.440499
Name: z-score, dtype: float64
```

Выбросы

Разберем, как выбросы влияют на меры центральной тенденции

Ввод [13]:

```
import pandas as pd
import numpy as np

bikes = pd.read_pickle('BikesDataVars.pkl')
bikes.head()
```

Out[13]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	257	0	Freezing	0
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True	219	0	Freezing	0
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True	162	0	Freezing	0
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True	148	1	Freezing	0
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True	97	0	Freezing	0

Ввод [14]:

```
bikes['Rental Count'].describe()
```

Out[14]:

```
count    8760.000000
mean      696.582078
std       749.812613
min         0.000000
25%       157.000000
50%       425.500000
75%      1009.000000
max      6012.000000
Name: Rental Count, dtype: float64
```

Найдите интерквартильный размах по атрибуту 'Rental Count'

Ввод [15]:

```
q75, q25 = np.percentile(bikes['Rental Count'], [75 ,25])
iqr = q75 - q25

#display interquartile range
iqr
```

Out[15]:

```
852.0
```

Найдите интерквартильный размах по атрибуту 'Rental Count', а также выведите значения q1 - 1.5 * iqr, q1 + 1.5 * iqr

Ввод [16]:

```
q75, q25 = np.percentile(bikes['Rental Count'], [75 ,25])
iqr = q75 - q25

#display interquartile range
print(iqr)

q1 = np.percentile(bikes['Rental Count'], 1)
iqr_outlier_threshold_up = q1 + 1.5*iqr
iqr_outlier_threshold_bottom = q1 - 1.5*iqr
```

```
852.0
```

Ввод [17]:

```
print(iqr_outlier_threshold_up)
print(iqr_outlier_threshold_bottom)
```

```
1278.0
-1278.0
```

Ввод [18]:

```
# Определили элементы-выбросы. shape Возвращает кортеж, представляющий размерность DataFrame. (строк, колонок)
bikes[bikes['Rental Count'] > iqr_outlier_threshold_up].shape
```

Out[18]:

```
(1556, 14)
```

Определите, в какие часы какое количество выбросов было зафиксировано. (value_counts)

Ввод [19]:

```
bikes[bikes['Rental Count'] > iqr_outlier_threshold_up]['Hour'].value_counts()
```

Out[19]:

```
18    171
17    138
19    137
20    135
21    125
16    114
22    111
8     105
15     81
14     68
12     63
23     60
7      59
9      49
13     46
0      33
11     32
10     18
1       7
6       2
2       2
Name: Hour, dtype: int64
```

Выведите количество выбросов по сезонам

Ввод [20]:

```
bikes[bikes['Rental Count'] > iqr_outlier_threshold_up]['Seasons'].value_counts()
```

Out[20]:

```
Summer    648
Autumn    498
Spring    409
Winter      1
Name: Seasons, dtype: int64
```

Выведите среднее, среднеквадратичное отклонение и пороги для атрибута Rental Count (+- 2.5 стандартных отклонений)

Ввод [21]:

```
#Ваш код
mean = bikes['Rental Count'].mean()
std = bikes['Rental Count'].std()
std_outlier_threshold_bottom = q1 - 2.5*std
std_outlier_threshold_up = q1 + 2.5*std

print(mean)
print(std)
print(std_outlier_threshold_bottom)
print(std_outlier_threshold_up)
```

```
696.5820776255708
749.8126131159588
-1874.531532789897
1874.531532789897
```

Определите количество выбросов по данной метрике (с shape)

Ввод [22]:

```
bikes[bikes['Rental Count'] > std_outlier_threshold_up].shape
```

Out[22]:

```
(702, 14)
```

Выведите количество выбросов по сезонам

Ввод [23]:

```
bikes[bikes['Rental Count'] > std_outlier_threshold_up]['Seasons'].value_counts()
```

Out[23]:

```
Summer    320
Autumn    214
Spring    168
Name: Seasons, dtype: int64
```

Ввод [24]:

```
#Посмотрим "нормальные" значения после фильтрации каждой метрикой
iqr_bikes_no_outliers = bikes[bikes['Rental Count'] <= iqr_outlier_threshold_up]
std_bikes_no_outliers = bikes[bikes['Rental Count'] <= std_outlier_threshold_up]
```

Ввод [25]:

```
#посмотрим меры центральной тенденции
print(bikes['Rental Count'].mean())
print(iqr_bikes_no_outliers['Rental Count'].mean())
print(std_bikes_no_outliers['Rental Count'].mean())
```

```
696.5820776255708
412.01915602443086
531.3824770414495
```

Ввод [26]:

```
print(bikes['Rental Count'].median())
print(iqr_bikes_no_outliers['Rental Count'].median())
print(std_bikes_no_outliers['Rental Count'].median())
```

```
425.5
305.0
367.0
```

Сравните устойчивость мер центральной тенденции к применению фильтров

Type *Markdown* and LaTeX: α^2

Пропущенные значения

Ввод [27]:

```
import pandas as pd
import numpy as np

bikes = pd.read_pickle('BikesData.pkl')
bikes.head()
```

Out[27]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	257	0	Freezing	0
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True	219	0	Freezing	0
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True	162	0	Freezing	0
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True	148	1	Freezing	0
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True	97	0	Freezing	0

Ввод [28]:

```
bikes.info() #обратите внимание на кол-во значений в Temperature. Есть пропущенные значения
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   Date                 8760 non-null   datetime64[ns]
 1   Hour                 8760 non-null   int64         
 2   Temperature          8581 non-null   float64       
 3   Humidity             8760 non-null   int64         
 4   Wind speed           8760 non-null   float64       
 5   Rainfall             8760 non-null   float64       
 6   Snowfall             8760 non-null   float64       
 7   Seasons              8760 non-null   object        
 8   Holiday              8760 non-null   int64         
 9   Functioning Day      8760 non-null   bool          
 10  Rental Count         8760 non-null   int64         
 11  Normal Humidity      8760 non-null   int64         
 12  Temperature Category  8581 non-null   category      
 13  Good Weather         8760 non-null   int64         
dtypes: bool(1), category(1), datetime64[ns](1), float64(4), int64(6), object(1)
memory usage: 838.5+ KB
```

Посчитайте количество пустых ячеек (isna())

Ввод [29]:

```
bikes.isna().sum()
```

Out[29]:

```
Date                     0
Hour                     0
Temperature            179
Humidity                 0
Wind speed               0
Rainfall                 0
Snowfall                 0
Seasons                  0
Holiday                  0
Functioning Day          0
Rental Count             0
Normal Humidity          0
Temperature Category   179
Good Weather             0
dtype: int64
```

Посчитайте количество заполненных ячеек (notna())

Ввод [30]:

```
bikes.notna().sum()
```

Out[30]:

```
Date                   8760
Hour                   8760
Temperature            8581
Humidity               8760
Wind speed             8760
Rainfall               8760
Snowfall               8760
Seasons                8760
Holiday                8760
Functioning Day        8760
Rental Count           8760
Normal Humidity        8760
Temperature Category   8581
Good Weather           8760
dtype: int64
```

Создайте новый датафрэйм bikes1. Скопируйте туда старый. Заполните пустые ячейки числом 42.

Ввод [31]:

```
bikes1 = bikes.copy()
bikes1['Temperature'].fillna(42, inplace=True)
bikes1.isna().sum()
```

Out[31]:

```
Date                       0
Hour                       0
Temperature                0
Humidity                   0
Wind speed                 0
Rainfall                   0
Snowfall                   0
Seasons                    0
Holiday                    0
Functioning Day            0
Rental Count               0
Normal Humidity            0
Temperature Category     179
Good Weather               0
dtype: int64
```

dropna не удаляет значения из датасета, если не добавить параметр inplace=True (но лучше так не надо)

Ввод [32]:

```
bikes.dropna(subset=['Temperature']).shape
```

Out[32]:

```
(8581, 14)
```

Заполните пустые ячейки в bikes медианой. (но созраняем в другую колонку)

Ввод [33]:

```
bikes['Temperature_Median'] = bikes['Temperature'].fillna(bikes['Temperature'].median())
bikes.iloc[38:42] # выводим строку с пропуском + ближайшие
```

Out[33]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather	Tempe
38	2017-12-02	14	7.3	35	1.3	0.0	0.0	Winter	0	True	1054	0	Chilly	0	
39	2017-12-02	15	NaN	41	2.3	0.0	0.0	Winter	0	True	688	1	NaN	0	
40	2017-12-02	16	6.4	48	2.6	0.0	0.0	Winter	0	True	592	1	Chilly	0	
41	2017-12-02	17	6.0	51	2.5	0.0	0.0	Winter	0	True	141	1	Chilly	0	

Ввод [34]:

```
np.random.choice(bikes['Temperature'].dropna()) # генератор случайных объектов из массива
```

Out[34]:

```
15.3
```

В функцию fillna можно передавать вектор значений. Он будет служить для заполнений пустых значений, главное чтобы количество элементов совпала

Ввод [35]:

```
temps = np.random.choice(bikes['Temperature'].dropna(), 8760)
temps[:100]
```

Out[35]:

```
array([ -7.7,   0.8,   2.5,  15.5,  23.5, -15.9,  16.5,  23.6,  21.6,
         3.9,  -2.2,  14.6,  -0.8,  19.9,   9.6,  31. ,   6.6,   9.3,
        11.9,   9.1,  -1.4,  -1.6,  17.6,   0.3,  23.4,  35. ,  35.6,
         9.5,  -8.3,   6.2,  22.7,   5.3, -15.2,   8. , -10.8,   7.3,
        20.6,  24.6,  17.1,   4.9,   3.5,   6.1,   9.1,  20.4,  37.3,
        20.9,  23.5,   9. ,  16.6,  24.1,  22.9,   3.6,   7.1,  26.7,
         1.5,  24.7,   5.7,  14.5,  17.7,  21.3,  16.5,  22.6,  26.4,
        -4.6, -10.6,   8.6,   0.2,  26.6,   5.1,   1.5,   7.1,  14.4,
        23.8,   8.5,  11. ,  13.2,   7.9,  24.8,  19.5,   9. ,  21.6,
        -5.3,  -2.8,  22.5,  -6.3,  30.9,  20.7,   8.1,  16.4,  22. ,
         9.8,  28.8,  23.6,   4.9,  22.9,  20.3, -10.3, -10. ,  23.6,
         2. ])
```

Создать новую колонку 'Temperature_Random', заполнить пустые значения из Temperature, сохранив новые значчения в Temperature_Random. Значения взять из temps

Ввод [36]:

```
bikes['Temperature_Random'] = bikes['Temperature'].fillna(pd.Series(temps)) # сделал за вас
```

Теперь заполните пробелы в Temperature в датафрейме bikes_1.

Ввод [37]:

```
bikes1['Temperature'].fillna(pd.Series(temps))
bikes1
```

Out[37]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	257	0	Freezing	0
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True	219	0	Freezing	0
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True	162	0	Freezing	0
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True	148	1	Freezing	0
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True	97	0	Freezing	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	2018-11-30	19	4.2	34	2.6	0.0	0.0	Autumn	0	True	644	0	Chilly	0
8756	2018-11-30	20	3.4	37	2.3	0.0	0.0	Autumn	0	True	359	0	Chilly	0
8757	2018-11-30	21	2.6	39	0.3	0.0	0.0	Autumn	0	True	1236	0	Chilly	0
8758	2018-11-30	22	2.1	41	1.0	0.0	0.0	Autumn	0	True	628	1	Chilly	0
8759	2018-11-30	23	1.9	43	1.3	0.0	0.0	Autumn	0	True	843	1	Chilly	0

8760 rows × 14 columns

Добавьте комментарии к коду ниже. Что он делает?

Ввод [38]:

```
# выводит медиану температуры по неделям в каждый час
bikes.groupby([bikes['Date'].dt.isocalendar().week, 'Hour'])['Temperature'].median()
```

Out[38]:

```
week  Hour
1     0      -4.3
      1      -4.8
      2      -5.3
      3      -5.5
      4      -5.1
             ...
52    19     -0.4
      20     -1.0
      21     -1.6
      22     -1.7
      23     -1.0
Name: Temperature, Length: 1248, dtype: float64
```

Type *Markdown* and LaTeX: α^2

Ввод [39]:

```
# находим групповые медианы по часам в неделях
temp_medians = bikes.groupby([bikes['Date'].dt.isocalendar().week, 'Hour'])['Temperature'].transform('median')
temp_medians
```

Out[39]:

```
0       2.75
1       2.50
2       1.35
3       2.15
4       2.15
        ...
8755    5.45
8756    4.70
8757    4.15
8758    3.50
8759    3.40
Name: Temperature, Length: 8760, dtype: float64
```

Type *Markdown* and LaTeX: α^2

Ввод [40]:

```
# создаём колонку Temperature_Median_Group, копируем в неё значения из Temperature и заполняем пустые ячейки temp_medians
bikes['Temperature_Median_Group'] = bikes['Temperature'].fillna(pd.Series(temp_medians))
# выводим строки с пустыми ячейками в Temperature
bikes[bikes['Temperature'].isna()].head()
```

Out[40]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather	Temp
39	2017-12-02	15	NaN	41	2.3	0.0	0.0	Winter	0	True	688	1	NaN	0	
50	2017-12-03	2	NaN	79	1.4	0.0	0.0	Winter	0	True	262	0	NaN	0	
64	2017-12-03	16	NaN	77	1.6	0.0	0.0	Winter	0	True	577	0	NaN	0	
105	2017-12-05	9	NaN	31	1.3	0.0	0.0	Winter	0	True	313	0	NaN	0	
151	2017-12-07	7	NaN	93	0.5	0.0	0.9	Winter	0	True	269	0	NaN	0	

Type *Markdown* and LaTeX: α^2

Ввод [41]:

```
# заполняем пустые ячейки в Temperature значениями temp_medians
bikes = pd.read_pickle('BikesDataVars.pkl')
bikes['Temperature'] = bikes['Temperature'].fillna(pd.Series(temp_medians))
```

Ввод [42]:

```
# определяем категорию температуры с помощью функции get_temp_cat
# применяем её к ячейкам в Temperature и сохраняем значения в Temperature Category
def get_temp_cat(temp):
    if temp < 0:
        return 'Freezing'
    elif temp < 15:
        return 'Chilly'
    elif temp < 26:
        return 'Nice'
    elif temp >= 26:
        return 'Hot'
    else:
        return temp

bikes['Temperature Category'] = pd.Categorical(bikes['Temperature'].apply(get_temp_cat))
```

Ввод [43]:

```
bikes.head()
```

Out[43]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	257	0	Freezing	0
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True	219	0	Freezing	0
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True	162	0	Freezing	0
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True	148	1	Freezing	0
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True	97	0	Freezing	0

Корреляция

Корреляция — это мера линейной взаимосвязи между двумя величинами. в ходе корреляционного анализа мы можем выявить только линейную взаимосвязь.

Ввод [44]:

```
import pandas as pd
import warnings
import numpy as np
warnings.filterwarnings('ignore')
bikes = pd.read_pickle('BikesDataImputed.pkl')
bikes.head()
```

Out[44]:

	Date	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Seasons	Holiday	Functioning Day	Rental Count	Normal Humidity	Temperature Category	Good Weather
0	2017-12-01	0	-5.2	37	2.2	0.0	0.0	Winter	0	True	257	0	Freezing	0
1	2017-12-01	1	-5.5	38	0.8	0.0	0.0	Winter	0	True	219	0	Freezing	0
2	2017-12-01	2	-6.0	39	1.0	0.0	0.0	Winter	0	True	162	0	Freezing	0
3	2017-12-01	3	-6.2	40	0.9	0.0	0.0	Winter	0	True	148	1	Freezing	0
4	2017-12-01	4	-6.0	36	2.3	0.0	0.0	Winter	0	True	97	0	Freezing	0

Ввод [45]:

```
#подготовим данные, возьмём аггрегированные по неделям значения температуры и количества аренд велосипедов.
# Посмотрим, насколько спрос зависит от погоды
temp_mean = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Temperature'].mean()
bikes_sum = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Rental Count'].sum()
```

Ввод [46]:

```
#Сведём переменные в одну таблицу с помощью функции concat. axis=1 служит для разделение на колонки
bikes_week = pd.concat([temp_mean, bikes_sum], axis=1)
bikes_week.head()
```

Out[46]:

week	Temperature	Rental Count
1	-2.694940	39441
2	-5.079762	30871
3	2.662500	42193
4	-10.038690	23079
5	-5.650595	28415

Ввод [47]:

```
#Возьмём 5 первых значений датафрейма с помощью функции iloc
first_five = bikes_week.iloc[0:4]
first_five
```

Out[47]:

week	Temperature	Rental Count
1	-2.694940	39441
2	-5.079762	30871
3	2.662500	42193
4	-10.038690	23079

Ввод [48]:

```
#вычислим X-Mx и Y - My. Температура - X, аренды - Y
first_five['X - Mx'] = bikes_week['Temperature'] - bikes_week['Temperature'].mean()
first_five['Y - My'] = bikes_week['Rental Count'] - bikes_week['Rental Count'].mean()

first_five
# должна получиться табличка вида Temperature   Rental Count    X - Mx  Y - My . И 5 недель строк
```

Out[48]:

week	Temperature	Rental Count	X - Mx	Y - My
1	-2.694940	39441	-15.598503	-77906.288462
2	-5.079762	30871	-17.983325	-86476.288462
3	2.662500	42193	-10.241063	-75154.288462
4	-10.038690	23079	-22.942253	-94268.288462

Ввод [49]:

```
# Найдём сумму квадратов для X и Y, а также сумму произведений
SSx = (bikes_week['Temperature']**2).sum()
SSy = (bikes_week['Rental Count']**2).sum()
SP = np.prod(bikes_week['Temperature']) + np.prod(bikes_week['Rental Count'])
print(SSx, SSy, SP)
```

```
15311.512057225234 888770645077 -1.3829934757025703e+52
```

Ввод [50]:

```
#найдём коэффициент корреляции
r = bikes['Temperature'].corr(bikes['Rental Count'])
r
```

Out[50]:

```
0.4547490734555483
```

Ввод [51]:

```
#проверим значение с помощью функции corr для first_five. Всё так же, корреляция Temperature и Rental Count
first_five['Temperature'].corr(first_five['Rental Count'])
```

Out[51]:

```
0.9546494504553402
```

Ввод [52]:

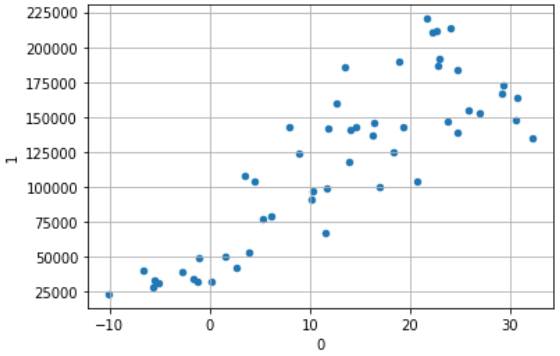
```
# Теперь найдём корреляцию не для 5 элементов, а ля всей генеральной совокупности (bikes_week)
bikes_week['Temperature'].corr(bikes_week['Rental Count'])
```

Out[52]:

```
0.8458075200534891
```

Ввод [53]:

```
# Построим график вида scatter (график рассеивания) по Temperature и Rental Count в bikes_week
pd.DataFrame(np.array([bikes_week['Temperature'], bikes_week['Rental Count']]).T).plot.scatter(0, 1, grid=True);
```



Объясните код ниже

Ввод [54]:

```
# исследуем корреляционные связи между числовыми столбцами в датафрейме 'bikes' и определить взаимосвязь
bikes.corr()
```

Out[54]:

	Hour	Temperature	Humidity	Wind speed	Rainfall	Snowfall	Holiday	Functioning Day	Rental Count	Normal Humidity	Good Weather
Hour	1.000000e+00	0.123610	-0.241644	0.285197	0.008715	-0.021516	-1.391486e-16	0.005439	0.345622	0.107503	0.073698
Temperature	1.236105e-01	1.000000	0.159793	-0.036418	0.050758	-0.217846	-5.570102e-02	-0.049849	0.454749	0.025467	0.206979
Humidity	-2.416438e-01	0.159793	1.000000	-0.336683	0.236397	0.108183	-5.027765e-02	-0.020800	-0.169085	-0.285947	-0.115874
Wind speed	2.851967e-01	-0.036418	-0.336683	1.000000	-0.019674	-0.003554	2.301677e-02	0.005037	0.097583	0.074964	0.032127
Rainfall	8.714642e-03	0.050758	0.236397	-0.019674	1.000000	0.008500	-1.426911e-02	0.002055	-0.103519	-0.095339	-0.042127
Snowfall	-2.151645e-02	-0.217846	0.108183	-0.003554	0.008500	1.000000	-1.259072e-02	0.032089	-0.120869	-0.067939	-0.054942
Holiday	-1.391486e-16	-0.055701	-0.050278	0.023017	-0.014269	-0.012591	1.000000e+00	-0.027624	-0.068822	-0.020156	0.029008
Functioning Day	5.439377e-03	-0.049849	-0.020800	0.005037	0.002055	0.032089	-2.762445e-02	1.000000	0.173437	0.002488	-0.101806
Rental Count	3.456218e-01	0.454749	-0.169085	0.097583	-0.103519	-0.120869	-6.882160e-02	0.173437	1.000000	0.128521	0.194224
Normal Humidity	1.075026e-01	0.025467	-0.285947	0.074964	-0.095339	-0.067939	-2.015629e-02	0.002488	0.128521	1.000000	0.440102
Good Weather	7.369784e-02	0.206979	-0.115874	0.032127	-0.042127	-0.054942	2.900771e-02	-0.101806	0.194224	0.440102	1.000000

Type *Markdown* and LaTeX: α^2

Ввод [55]:

```
# переменные содержащие значения влажности и скорости ветра для каждой недели в датафрейме 'bikes'
humidity_mean = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Humidity'].mean()
wind_mean = bikes.groupby(bikes['Date'].dt.isocalendar().week)['Wind speed'].mean()
```

Type *Markdown* and LaTeX: α^2

Ввод [56]:

```
# 'bikes_week' - это новый датафрем, в который bikes_week, humidity_mean, wind_mean объединяем в одну таблицу
bikes_week = pd.concat([bikes_week, humidity_mean, wind_mean], axis=1)
bikes_week
```

Out[56]:

week	Temperature	Rental Count	Humidity	Wind speed
1	-2.694940	39441	43.660714	1.524405
2	-5.079762	30871	53.958333	1.995833
3	2.662500	42193	55.178571	1.385119
4	-10.038690	23079	38.410714	2.575000
5	-5.650595	28415	47.815476	2.256548
6	-5.486310	33259	41.571429	2.275000
7	-1.225298	32139	38.422619	2.332738
8	1.631548	50136	50.041667	2.055357
9	4.004167	52958	55.166667	1.770238
10	5.269940	77316	60.988095	1.802381
11	10.133036	90547	58.119048	1.723810
12	6.162202	79109	59.136905	2.245833
13	13.890774	118031	62.321429	1.817262
14	10.283631	97000	67.178571	2.385714
15	11.713988	98468	51.166667	2.233333
16	14.669940	142918	41.059524	1.642262
17	14.061310	141147	56.172619	1.798810
18	16.365476	146127	66.392857	2.025595
19	16.220238	136607	66.142857	1.623214
20	19.307738	142346	69.363095	1.380952
21	18.988988	189749	48.666667	1.793452
22	22.223214	210326	53.220238	1.817262
23	22.628869	211869	59.916667	1.763095
24	21.751190	220392	64.244048	1.422024
25	24.036012	213553	57.553571	1.929167
26	23.757143	146455	80.690476	1.381548
27	24.758333	183652	67.958333	1.339881
28	25.858333	154848	78.303571	1.416667
29	29.297619	172636	61.255952	1.554167
30	30.740774	163447	63.547619	1.644048
31	32.193750	135086	56.791667	1.819048
32	30.518452	147911	64.642857	1.578571
33	29.238690	166679	53.708333	1.427976
34	26.901488	152282	66.357143	1.975000
35	24.691071	138529	76.297619	1.490476
36	22.962798	191800	63.065476	1.858333
37	22.782143	186208	62.607143	1.375000
38	20.743750	103925	66.869048	1.528571
39	18.413690	124820	51.065476	1.510714
40	17.019643	99622	64.946429	1.801786
41	12.672024	159527	53.934524	1.501190
42	13.542857	185695	56.696429	1.219643
43	11.788393	141509	63.410714	1.552976
44	8.882440	123557	54.797619	1.476786
45	11.590774	66682	74.005952	1.480357
46	7.953571	142787	54.488095	1.228571
47	4.422321	103454	51.904762	1.550000
48	3.505208	107727	54.859375	1.447917
49	-0.992560	48680	54.958333	1.697024
50	-6.645536	40147	42.898810	2.022619
51	0.210714	31938	70.178571	1.380952
52	-1.651786	34460	52.136905	1.900595

Type *Markdown* and LaTeX: α^2

Ввод [57]:

```
bikes_week.corr()
#обратите внимение на единицы по диагонали. Почему они появились? Оцените степень корреляции между атрибутами
#Единицы на диагонали матрицы корреляции возникают, так как на диагонали отображаются коэффициенты корреляции между каждым атрибутом
#Оцените степень корреляции между атрибутами
# степень корреляции в основном не доходит до 0.6 или -0.6, это означает, что не можем предсказать увеличение одного значения вместе
```

Out[57]:

	Temperature	Rental Count	Humidity	Wind speed
Temperature	1.000000	0.845808	0.584642	-0.420474
Rental Count	0.845808	1.000000	0.389963	-0.434142
Humidity	0.584642	0.389963	0.000000	-0.456225
Wind speed	-0.420474	-0.434142	-0.456225	1.000000

Type *Markdown* and LaTeX: α^2

Ввод [58]:

```
дет расчет коэф корреляции между столбцом rental count и bikes_week для оценки степени взаимосвязи между атрибутами и кол-вом аренд
r()['Rental Count']
```

Out[58]:

```
Temperature     0.845808
Rental Count    1.000000
Humidity        0.389963
Wind speed     -0.434142
Name: Rental Count, dtype: float64
```

Type *Markdown* and LaTeX: α^2

Ввод [59]:

```
#Составьте список убывания коэффициента корреляции между всеми столбцами по убыванию (по модулю)
bikes_corr = bikes.corr()
corrs = [(bikes_corr.iloc[row,col], bikes.columns[row], bikes.columns[col]) for row in range(1, 10) for col in range(row)]
sorted_corrs = sorted(corrs, key=lambda x: abs(x[0]), reverse=True)
sorted_corrs
```

Out[59]:

```
[(0.4547490734555487, 'Holiday', 'Hour'),
 (0.34562175315919386, 'Holiday', 'Date'),
 (-0.33668304169134267, 'Humidity', 'Temperature'),
 (-0.28594695010526905, 'Functioning Day', 'Temperature'),
 (0.28519666008802735, 'Humidity', 'Date'),
 (-0.24164378742345013, 'Temperature', 'Date'),
 (0.23639667038954237, 'Wind speed', 'Temperature'),
 (-0.2178457098353822, 'Rainfall', 'Hour'),
 (0.1734370473148034, 'Holiday', 'Seasons'),
 (-0.16908520336621008, 'Holiday', 'Temperature'),
 (0.15979286879517288, 'Temperature', 'Hour'),
 (0.12852056916519566, 'Functioning Day', 'Holiday'),
 (0.1236104739437595, 'Hour', 'Date'),
 (-0.12086903229450317, 'Holiday', 'Rainfall'),
 (0.10818345255675188, 'Rainfall', 'Temperature'),
 (0.1075025932578112, 'Functioning Day', 'Date'),
 (-0.10351877421692345, 'Holiday', 'Wind speed'),
 (0.09758344676299194, 'Holiday', 'Humidity'),
 (-0.09533875715092673, 'Functioning Day', 'Wind speed'),
 (0.07496403997364812, 'Functioning Day', 'Humidity'),
 (-0.06882159717340262, 'Holiday', 'Snowfall'),
 (-0.067939471848175, 'Functioning Day', 'Rainfall'),
 (-0.055701017228717856, 'Snowfall', 'Hour'),
 (0.050757583884366936, 'Wind speed', 'Hour'),
 (-0.050277646340957605, 'Snowfall', 'Temperature'),
 (-0.04984945881895369, 'Seasons', 'Hour'),
 (-0.03641779329385089, 'Humidity', 'Hour'),
 (0.032088609547791656, 'Seasons', 'Rainfall'),
 (-0.027624448466706388, 'Seasons', 'Snowfall'),
 (0.02546715042359334, 'Functioning Day', 'Hour'),
 (0.023016771111451454, 'Snowfall', 'Humidity'),
 (-0.021516454679546427, 'Rainfall', 'Date'),
 (-0.02079995500527071, 'Seasons', 'Temperature'),
 (-0.02015628714430786, 'Functioning Day', 'Snowfall'),
 (-0.019674088872894423, 'Wind speed', 'Humidity'),
 (-0.01426911025264864, 'Snowfall', 'Wind speed'),
 (-0.012590722095085303, 'Snowfall', 'Rainfall'),
 (0.008714641861776019, 'Wind speed', 'Date'),
 (0.008499653476506157, 'Rainfall', 'Wind speed'),
 (0.0054393772084616885, 'Seasons', 'Date'),
 (0.005036937745485467, 'Seasons', 'Humidity'),
 (-0.003554186117569801, 'Rainfall', 'Humidity'),
 (0.0024878684856583766, 'Functioning Day', 'Seasons'),
 (0.002054573192442602, 'Seasons', 'Wind speed'),
 (-1.3914864368407556e-16, 'Snowfall', 'Date')]
```