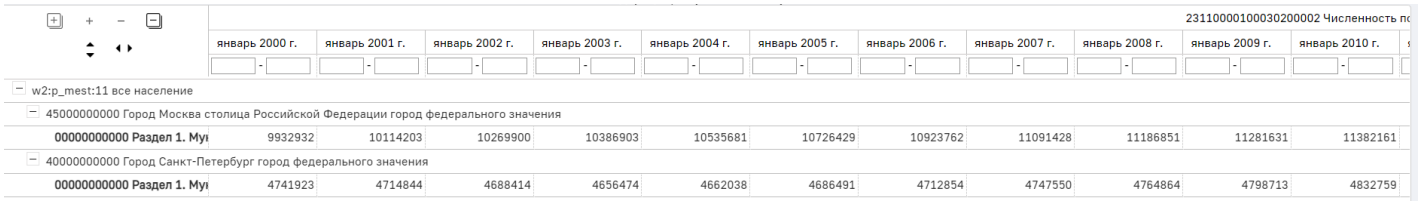
1 Установка

2 Пример использования

Рассмотрим пример для обработки данных по численности постоянного населения Москвы и Санкт-Петербурга за период 2000-2021 годов.

Наши данные выглядят следующим образом:



												23110000100030200002 Численность по
	январь 2000 г.	январь 2001 г.	январь 2002 г.	январь 2003 г.	январь 2004 г.	январь 2005 г.	январь 2006 г.	январь 2007 г.	январь 2008 г.	январь 2009 г.	январь 2010 г.	
w2:p_mest:11 все население												
45000000000 Город Москва столица Российской Федерации город федерального значения												
00000000000 Раздел 1. Мун	9932932	10114203	10269900	10386903	10535681	10726429	10923762	11091428	11186851	11281631	11382161	
40000000000 Город Санкт-Петербург город федерального значения												
00000000000 Раздел 1. Мун	4741923	4714844	4688414	4656474	4662038	4686491	4712854	4747550	4764864	4798713	4832759	

Всего Росстат выделяет нам 6 строчек, из которых нужные нам – 4ая и 6ая.

Открываем Pycharm (Visual Studio Code) и создаем .py-файл. Перво-наперво подключаем нужные нам библиотеки и модули при помощи функции *import*:

```
import pandas as pd
import numpy as np
import scipy.stats as stat
import matplotlib.pyplot as plt
```

Pandas нужен для чтения .xlsx-файла и дальнешей работы с данными; Numpy – для работы с массивами и вычислениями основным статистик; модуль stats из библиотеки Scipy позволяет вычислять некоторые иными статистики, которых нет в Numpy; и, наконец, модуль pyplot для рисования графиков. Приписка *... as ...* в каждой строчке упрощает обращение к библиотекам – теперь нет необходимости писать её длинное название, достаточно лишь написать сокращение, которое мы сами можем выбрать.

Далее следует скачать таблицу и поместить её в одну папку с .py-файлом. Для простоты я переименовал её в "moscow_spb.xlsx". Таким образом, воспользуемся функцией из Pandas для чтения .xlsx-файла:

```
df = pd.read_excel('moscow_spb.xlsx')
```

Теперь мы можем посмотреть, что внутри переменной df.

```
print(df)
```

И мы получим:

```
                                    Unnamed: 0  ... 23110000100030200002 Численность постоянного населения на 1 января.21
0                                          NaN  ...                                    январь 2021 г.
1                                все население  ...                                               NaN
2  Город Москва столица Российской Федерации горо...  ...                                               NaN
3  Раздел 1. Муниципальные образования субъектов ...  ...                                          12655050
4  Город Санкт-Петербург город федерального значения  ...                                               NaN
5  Раздел 1. Муниципальные образования субъектов ...  ...                                           5384342

[6 rows x 24 columns]
```

Да, он не вывел нам всю таблицу, но можно увидеть, что сейчас в датафрейме 6 строк и 24 колонки. Изначально мы уже определили, что нам нужно 2 строчки, а период с 2000 по 2021 составляет 22 значения. Соответственно, нам необходимо "почистить" эти данные.

Со строчками мы определились выше – 4ая и 6ая, а с колонками всё иначе. Первые две колонки содержат ненужные индексы, а их названия слишком громоздки. Программно это выглядит следующим образом:

```
main_df = pd.DataFrame()
main_df = main_df.append(df.iloc[-3])
main_df = main_df.append(df.iloc[-1])

del main_df["Unnamed: 0"]
del main_df["Unnamed: 1"]

main_df = main_df.reset_index(drop=True)
main_df.columns = np.array([year for year in range(2000, 2022)])
```

Сначала мы создаем пустой датафрейм, куда мы положим нужные нам столбцы и строки. Затем мы добавляем нужную строку при помощи метода .append. То, что находится в скобках после .append – это то, что мы добавляем в main_df. Метод .iloc позволяет обращаться к строкам датафрейма по индексу. Этот индекс начинается с нуля, то есть, df.iloc[0] выдаст нам первую строчку, df.iloc[1] – вторую и т.д.. Однако массивы в Python позволяют принимать отрицательные значения, что равносильно "проходу по массиву" в обратную сторону. Это значит, что df.iloc[-1] выдаст последнюю строку, а df.iloc[-3] - третью с конца. Итак, строки добавлены.

Далее мы определили, что первые две колонки содержат незначащие индексы и подписи, поэтому при помощи функции del эти колонки последовательно удаляются. Так как у них не было названия, к ним можно обратиться как "Unnamed: 0" и "Unnamed: 1" соответственно.

Если мы посмотрим на наш датафрейм сейчас, то увидим громоздкие названия столбцов и неверые индексы у строк. К названиям столбцов можно обратиться при помощи метода .columns, а присвоение чего-либо соизмеримого их просто переименует. Так показано, что мы добавляем массив (array), в котором последовательно указаны все года (все целые значения), принадлежащие полуинтервалу $[2000, 2022)$. У любого датафрейма также есть возможность сбросить по умолчанию нумерацию строк – .reset_index(drop=True).

"Очищенные" данные выглядят следующим образом:

```
        2000        2001        2002  ...        2019        2020        2021
0  9932932.0  10114203.0  10269900.0  ...  12615279.0  12678079.0  12655050.0
1  4741923.0   4714844.0   4688414.0  ...   5383890.0   5398064.0   5384342.0

[2 rows x 22 columns]
```

Первый город – Москва, второй – Санкт-Петербург. То есть, наша задача сводится к тому, чтобы пройти датафрейм построчно, описать данные и нанести их на график.

Метод .iterrows() выдает нам 2 значения – индекс строки и саму строку. Таким образом, запустив цикл, мы можем "пройтись" по всем строкам. То есть,

```
for i, row in main_df.iterrows():
```

На каждой из двух итераций в переменную row будет записан массив с 22мя значениями. Этого хватит, чтобы найти разные статистики.

```
print('min: ', np.min(row), main_df.columns[np.argmin(row)])
print('max: ', np.max(row), main_df.columns[np.argmax(row)])
print('mean: ', np.mean(row))
print('median: ', np.median(row))
print('sd: ', np.std(row, ddof=1))
print('interquartile range: ', stat.iqr(row))
print('range: ', np.max(row) - np.min(row))
print('skewness: ', stat.skew(row))
print('kurtosis: ', stat.kurtosis(row))
```

По порядку:

- min - минимум;
- max - максимум;
- mean - среднее;
- median - медиана;
- sd (std) - стандартное отклонение;
- interquartile range (iqr) - интерквартильный размах;
- range - размах;
- skewness (skew) - коэффициент асимметрии;
- kurtosis - эксцесс.