Лабораторная работа 4: Клиометрика в Python

Немного теории

Клиометрика - это область исторических наук, которая занимается количественным анализом и измерением исторических данных. Основная цель клиометрики заключается в применении статистических методов и моделей для изучения исторических явлений, трендов, паттернов и изменений в общественной, экономической и политической сферах.

Этот подход позволяет исследователям проводить количественный анализ исторических данных, строить статистические модели для анализа временных рядов, выявлять закономерности и тенденции, а также делать выводы о прошлом на основе количественных данных.

Применение методов клиометрики может включать в себя использование статистических методов, анализ временных рядов, множественную регрессию, методы машинного обучения и другие инструменты для изучения и интерпретации исторических событий и процессов.

Важно отметить, что клиометрика не заменяет традиционные исторические исследования, а дополняет их, предоставляя инструменты для более объективного и количественного анализа исторических данных.

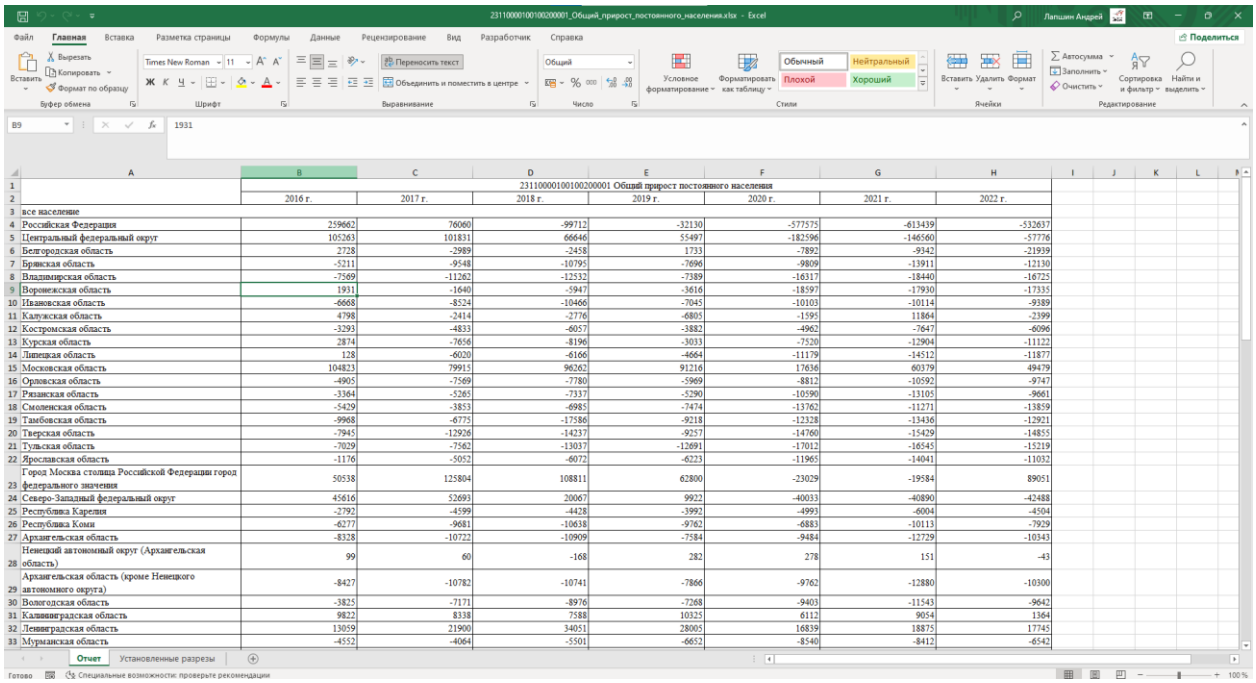
Порядок выполнения лабораторной работы

1 Задача и цель

Целью данной лабораторной работы является применение методов клиометрики для анализа исторических / временных данных с использованием языка программирования Python.

2 Сбор данных

Зайдите на сайт Росстата и скачайте файл с расширением csv или xlsx, после чего откройте его, чтобы убедиться, что данные в таблице находятся в нормальном виде. В качестве примера скачаем данные по общему приросту постоянного населения, как показано на рисунке 1 в расширении xlsx.



	A	B	C	D	E	F	G	H
1		2311000010010020000 1 Общий прирост постоянного населения						
2		2016 г.	2017 г.	2018 г.	2019 г.	2020 г.	2021 г.	2022 г.
3	все население							
4	Российская Федерация	259662	76060	-99712	-32130	-577575	-613439	-532637
5	Центральный федеральный округ	105263	101831	66646	55497	-182596	-146560	-57776
6	Белгородская область	2728	-2989	-2458	1733	-7892	-9342	-21939
7	Брянская область	-5211	-9548	-10795	-7696	-9809	-13911	-12130
8	Владимирская область	-7569	-11262	-12532	-7389	-16317	-18440	-16725
9	Воронежская область	1931	-1640	-5947	-3616	-18597	-17930	-17335
10	Ивановская область	-6668	-8524	-10466	-7045	-10103	-10114	-9389
11	Калужская область	4798	-2414	-2776	-6805	-1595	11864	-2399
12	Костромская область	-3293	-4833	-6057	-3882	-4962	-7647	-6096
13	Курская область	2874	-7656	-8196	-3033	-7520	-12904	-11122
14	Липецкая область	128	-6020	-6166	-4664	-11179	-14512	-11877
15	Московская область	104823	79915	96262	91216	17636	60379	49479
16	Орловская область	-4905	-7569	-7780	-5969	-8812	-10592	-9747
17	Рязанская область	-3364	-5265	-7337	-5290	-10590	-13105	-9661
18	Смоленская область	-5429	-3853	-6985	-7474	-13762	-11271	-13859
19	Тамбовская область	-9968	-6775	-17586	-9218	-12328	-13436	-12921
20	Тверская область	-7945	-12926	-14237	-9257	-14760	-15429	-14855
21	Тульская область	-7029	-7562	-13037	-12691	-17012	-16545	-15219
22	Ярославская область	-1176	-5052	-6072	-6223	-11965	-14041	-11032
23	Город Москва столица Российской Федерации город федерального значения	50538	125804	108811	62800	-23029	-19584	89051
24	Северо-Западный федеральный округ	45616	52693	20067	9922	-40033	-40890	-42488
25	Республика Карелия	-2792	-4599	-4428	-3992	-4993	-6004	-4504
26	Республика Коми	-6277	-9681	-10638	-9762	-6883	-10113	-7929
27	Архангельская область	-8328	-10722	-10909	-7584	-9484	-12729	-10343
28	Ненецкий автономный округ (Архангельская область)	99	60	-168	282	278	151	-43
29	Архангельская область (кроме Ненецкого автономного округа)	-8427	-10782	-10741	-7866	-9762	-12880	-10300
30	Вологодская область	-3825	-7171	-8976	-7268	-9403	-11543	-9642
31	Калининградская область	9822	8338	7588	10325	6112	9054	1364
32	Ленинградская область	13059	21900	34051	28005	16839	18875	17745
33	Мурманская область	-4552	-4064	-5501	-6652	-8540	-8412	-6542

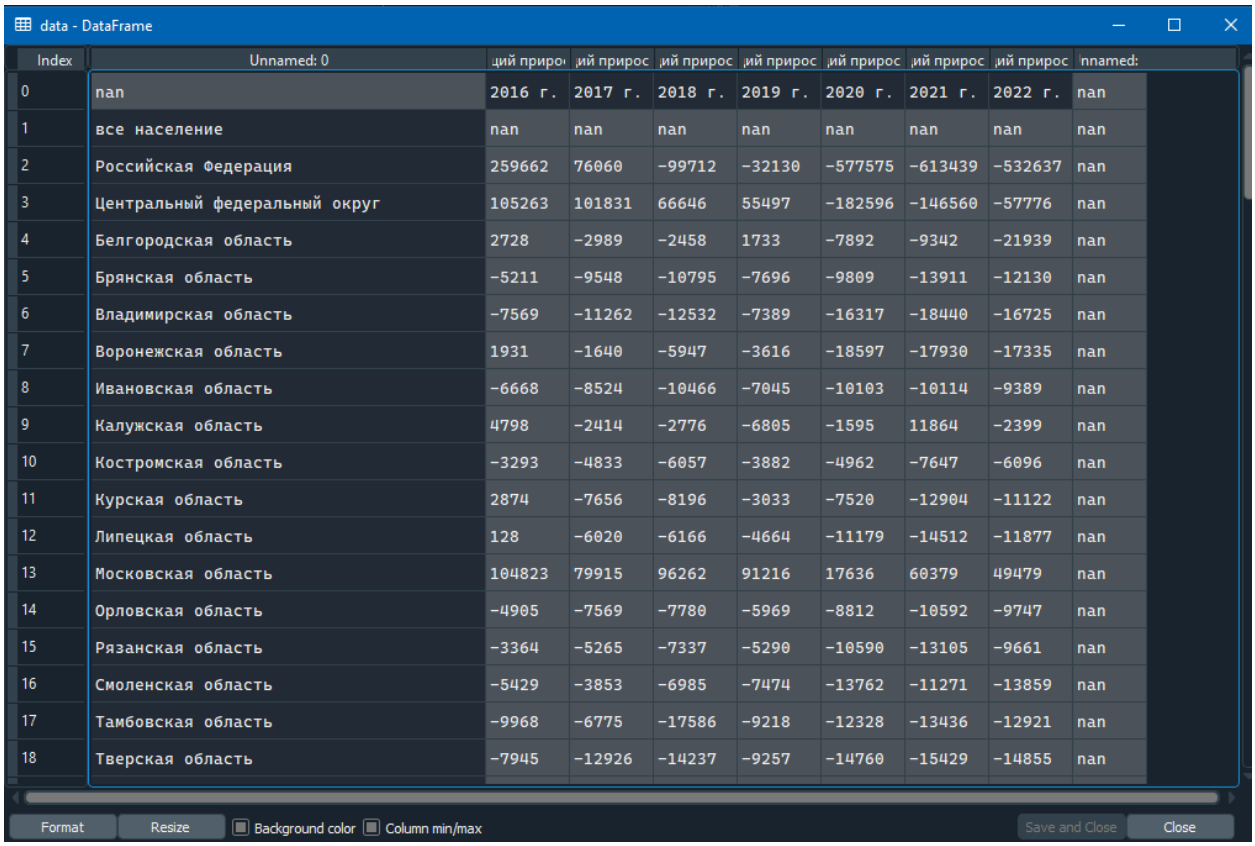
Рисунок 1 — Исходные данные

3 Подготовка данных

Откроем эти данные в Phyton. Предварительно разместив файл в папку, где будет лежать и код.

```
import pandas as pd
import matplotlib.pyplot as plt
# Загрузка данных из файла xlsx
file_path = '23110000100100200001_Общий_прирост_постоянного_населения.xlsx'
data = pd.read_excel(file_path)
```

В результате в переменной data будут лежать наши данные, как показано на рисунке 2.



data - DataFrame

Index	Unnamed: 0	ций приро	ий прирос	ий прирос	ий прирос	ий прирос	ий прирос	ий прирос	nnamed:
0	nan	2016 г.	2017 г.	2018 г.	2019 г.	2020 г.	2021 г.	2022 г.	nan
1	все население	nan	nan	nan	nan	nan	nan	nan	nan
2	Российская Федерация	259662	76060	-99712	-32130	-577575	-613439	-532637	nan
3	Центральный федеральный округ	105263	101831	66646	55497	-182596	-146560	-57776	nan
4	Белгородская область	2728	-2989	-2458	1733	-7892	-9342	-21939	nan
5	Брянская область	-5211	-9548	-10795	-7696	-9809	-13911	-12130	nan
6	Владимирская область	-7569	-11262	-12532	-7389	-16317	-18440	-16725	nan
7	Воронежская область	1931	-1640	-5947	-3616	-18597	-17930	-17335	nan
8	Ивановская область	-6668	-8524	-10466	-7045	-10103	-10114	-9389	nan
9	Калужская область	4798	-2414	-2776	-6805	-1595	11864	-2399	nan
10	Костромская область	-3293	-4833	-6057	-3882	-4962	-7647	-6096	nan
11	Курская область	2874	-7656	-8196	-3033	-7520	-12904	-11122	nan
12	Липецкая область	128	-6020	-6166	-4664	-11179	-14512	-11877	nan
13	Московская область	104823	79915	96262	91216	17636	60379	49479	nan
14	Орловская область	-4905	-7569	-7780	-5969	-8812	-10592	-9747	nan
15	Рязанская область	-3364	-5265	-7337	-5290	-10590	-13105	-9661	nan
16	Смоленская область	-5429	-3853	-6985	-7474	-13762	-11271	-13859	nan
17	Тамбовская область	-9968	-6775	-17586	-9218	-12328	-13436	-12921	nan
18	Тверская область	-7945	-12926	-14237	-9257	-14760	-15429	-14855	nan

Format Resize Background color Column min/max Save and Close Close

Рисунок 2 — Данные

Как видно из рисунка 2 в data есть целая строка и столбец, которые содержат в себе nan (пустые значения). Это будет мешать в дальнейшем, поэтому уберём их методами drop и dropna.

Пример:

```
data.dropna(axis=1, how='all', inplace=True)  # Удаление столбцов, где все значения пропущены
data.drop(1, inplace=True) # Удаление строки с пустыми значениями
```

Немного о методах:

dropna()

Что делает: Метод dropna() используется для удаления строк или столбцов, содержащих пропущенные значения (NaN).

Параметры:

- axis: Указывает, удалять строки (0) или столбцы (1). По умолчанию удаляет строки (0).

- how: Определяет, когда удалять. 'any' удаляет строки/столбцы, если есть хотя бы одно пропущенное значение; 'all' удаляет, если все значения пропущены.
- subset: Опциональный параметр, позволяющий указать определенные столбцы или строки для проверки на наличие пропущенных значений.

drop()

Что делает: Метод drop() используется для удаления строк или столбцов по их метке (индексу) или меткам.

Параметры:

- labels: Индекс или список индексов строк/столбцов, которые необходимо удалить.
- axis: Указывает, удалять строки (0) или столбцы (1). По умолчанию удаляет строки (0).
- inplace: Булево значение, определяющее, изменять исходный DataFrame или возвращать новый. Если True, изменяет исходный DataFrame, иначе возвращает новый.

Краткое сравнение методов:

- dropna() - удаляет строки или столбцы с пропущенными значениями NaN.
- drop() - удаляет строки или столбцы по их индексу (метке).

Итак, после применения данных методов наш DataFrame выглядит так, как показано на рисунке 3.

Index	Unnamed: 0	ий приро	ий прирос	ий прирос	ий прирос	ий прирос	ий прирос	ий прирос
0	nan	2016 г.	2017 г.	2018 г.	2019 г.	2020 г.	2021 г.	2022 г.
2	Российская Федерация	259662	76060	-99712	-32130	-577575	-613439	-532637
3	Центральный федеральный округ	105263	101831	66646	55497	-182596	-146560	-57776
4	Белгородская область	2728	-2989	-2458	1733	-7892	-9342	-21939
5	Брянская область	-5211	-9548	-10795	-7696	-9809	-13911	-12130
6	Владимирская область	-7569	-11262	-12532	-7389	-16317	-18440	-16725
7	Воронежская область	1931	-1640	-5947	-3616	-18597	-17930	-17335
8	Ивановская область	-6668	-8524	-10466	-7045	-10103	-10114	-9389
9	Калужская область	4798	-2414	-2776	-6805	-1595	11864	-2399
10	Костромская область	-3293	-4833	-6057	-3882	-4962	-7647	-6096
11	Курская область	2874	-7656	-8196	-3033	-7520	-12904	-11122
12	Липецкая область	128	-6020	-6166	-4664	-11179	-14512	-11877
13	Московская область	104823	79915	96262	91216	17636	60379	49479
14	Орловская область	-4905	-7569	-7780	-5969	-8812	-10592	-9747
15	Рязанская область	-3364	-5265	-7337	-5290	-10590	-13105	-9661
16	Смоленская область	-5429	-3853	-6985	-7474	-13762	-11271	-13859
17	Тамбовская область	-9968	-6775	-17586	-9218	-12328	-13436	-12921
18	Тверская область	-7945	-12926	-14237	-9257	-14760	-15429	-14855
19	Тульская область	-7029	-7562	-13037	-12691	-17012	-16545	-15219
20	Ярославская область	-1176	-5052	-6072	-6223	-11965	-14041	-11032
21	Город Москва столица Российской Федера…	50538	125804	108811	62800	-23029	-19584	89051
22	Северо-Западный федеральный округ	45616	52693	20067	9922	-40033	-40890	-42488
23	Республика Карелия	-2792	-4599	-4428	-3992	-4993	-6004	-4504
24	Республика Коми	-6277	-9681	-10638	-9762	-6883	-10113	-7929
25	Архангельская область	-8328	-10722	-10909	-7584	-9484	-12729	-10343
26	Ненецкий автономный округ (Архангельск…	99	60	-168	282	278	151	-43
27	Архангельская область (кроме Ненецкого…	-8427	-10782	-10741	-7866	-9762	-12880	-10300
28	Вологодская область	-3825	-7171	-8976	-7268	-9403	-11543	-9642
29	Калининградская область	9822	8338	7588	10325	6112	9054	1364
30	Ленинградская область	13059	21900	34051	28005	16839	18875	17745
31	Мурманская область	-4552	-4064	-5501	-6652	-8540	-8412	-6542

Рисунок 3 — Очищенные данные

Как видно теперь, нам остается лишь изменить значение столбца Unnamed:0 с индексом 0, чтобы убрать последний nan. Воспользуемся методом loc.

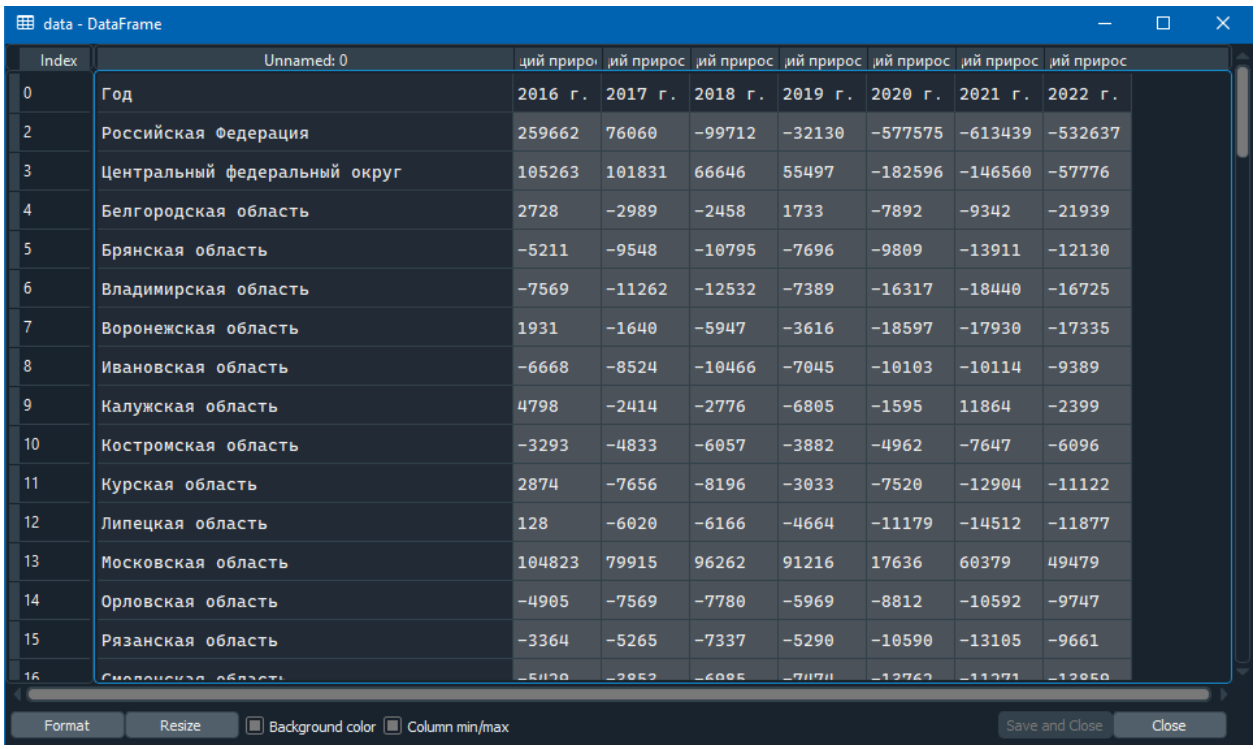
```
data.loc[0, 'Unnamed: 0'] = 'Год'
```

Немного о методе:

loc в Pandas - это метод, который позволяет обращаться к данным в DataFrame по меткам индекса строк и названиям столбцов. Он используется

для чтения, изменения и добавления данных в DataFrame, а также для работы с срезами данных по меткам. Этот метод обеспечивает гибкость доступа к данным, используя понятный и удобный синтаксис для манипулирования информацией в DataFrame.

Конечный вид данных представлен на рисунке 4.



Index	Unnamed: 0	ий приро	ий прирос	ий прирос	ий прирос	ий прирос	ий прирос	ий прирос
0	Год	2016 г.	2017 г.	2018 г.	2019 г.	2020 г.	2021 г.	2022 г.
2	Российская Федерация	259662	76060	-99712	-32130	-577575	-613439	-532637
3	Центральный федеральный округ	105263	101831	66646	55497	-182596	-146560	-57776
4	Белгородская область	2728	-2989	-2458	1733	-7892	-9342	-21939
5	Брянская область	-5211	-9548	-10795	-7696	-9809	-13911	-12130
6	Владимирская область	-7569	-11262	-12532	-7389	-16317	-18440	-16725
7	Воронежская область	1931	-1640	-5947	-3616	-18597	-17930	-17335
8	Ивановская область	-6668	-8524	-10466	-7045	-10103	-10114	-9389
9	Калужская область	4798	-2414	-2776	-6805	-1595	11864	-2399
10	Костромская область	-3293	-4833	-6057	-3882	-4962	-7647	-6096
11	Курская область	2874	-7656	-8196	-3033	-7520	-12904	-11122
12	Липецкая область	128	-6020	-6166	-4664	-11179	-14512	-11877
13	Московская область	104823	79915	96262	91216	17636	60379	49479
14	Орловская область	-4905	-7569	-7780	-5969	-8812	-10592	-9747
15	Рязанская область	-3364	-5265	-7337	-5290	-10590	-13105	-9661
16	Смоленская область	-5429	-3853	-6985	-7474	-13762	-11271	-13859

Рисунок 4 — Конечный вид данных

4 Анализ данных и визуализация

Допишем код до конца и разберем его остальные части.

Пример:

```
import pandas as pd
import matplotlib.pyplot as plt

# Загрузка данных из файла xlsx
file_path = '23110000100100200001_Общий_прирост_постоянного_населения.xlsx'
data = pd.read_excel(file_path)
data.dropna(axis=1, how='all', inplace=True)  # Удаление столбцов, где
```

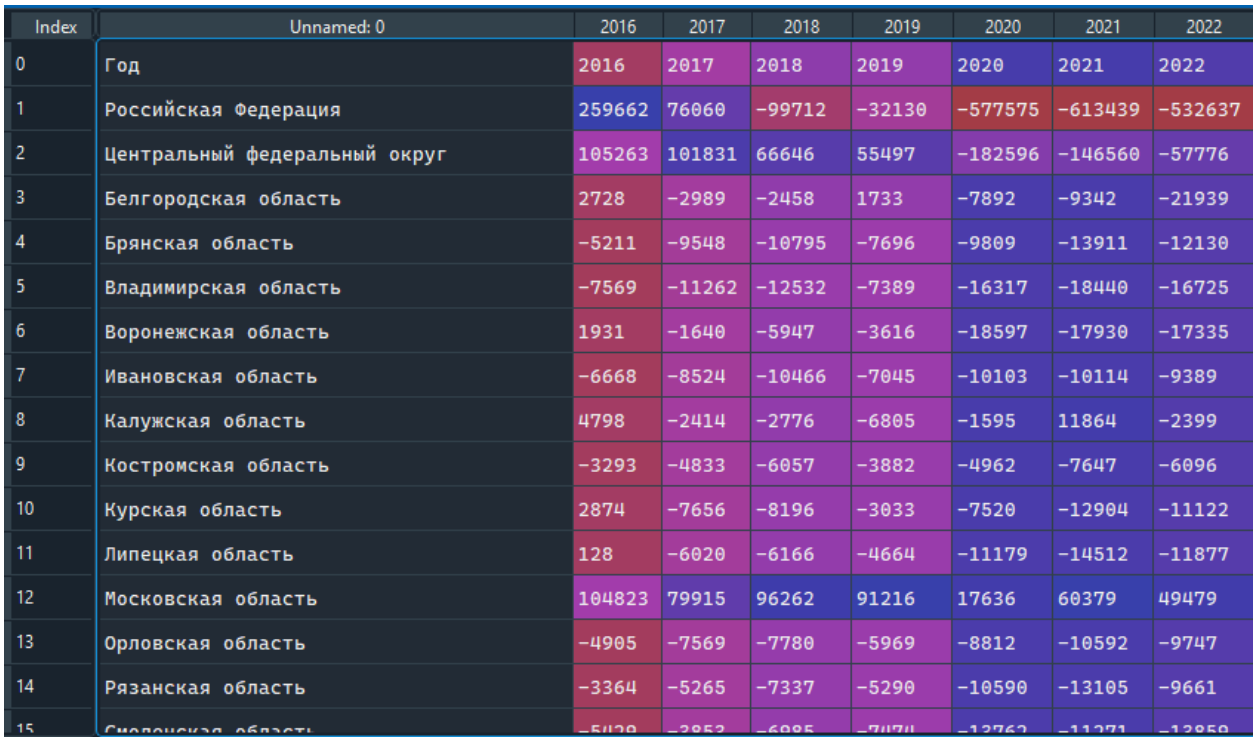
```
все значения пропущены
        data.drop(1, inplace=True) # Удаление строки с пустыми значениями
        data.loc[0, 'Unnamed: 0'] = 'Год'
        data = data.reset_index(drop=True)
        data.rename(columns={
           'Общий прирост постоянного населения': '2016',
           '23110000100100200001 Общий прирост постоянного населения':
'2017',
           '23110000100100200001 Общий прирост постоянного населения.1':
'2018',
           '23110000100100200001 Общий прирост постоянного населения.2':
'2019',
           '23110000100100200001 Общий прирост постоянного населения.3':
'2020',
           '23110000100100200001 Общий прирост постоянного населения.4':
'2021',
           '23110000100100200001 Общий прирост постоянного населения.5':
'2022'
        }, inplace=True)

        row_to_plot = data.iloc[1]  # Выбираем строку с индексом 1
        row_to_plot.drop('Unnamed: 0', axis=0, inplace=True)




        row_to_plot = row_to_plot.astype(float)  # Преобразуем значения в
числовой формат, если они не числа

        plt.figure(figsize=(10, 6))
        plt.plot(row_to_plot.index,     row_to_plot.values,     marker='o')          #
```

```
Построение графика с маркерами для каждой точки
    plt.xlabel('Года')
    plt.ylabel('Значения')
    plt.title('График временного ряда')
    plt.grid(True)
    plt.show()
```

Метод reset_index() в pandas используется для сброса индекса DataFrame и создания нового целочисленного индекса. При использовании параметра drop=True, старый индекс будет удален, а новый индекс будет создан с нуля, начиная с 0. Данный метод мы используем так как ранее удаляли строки с пустыми значениями, вследствие чего в индексах строк произошел пробел.

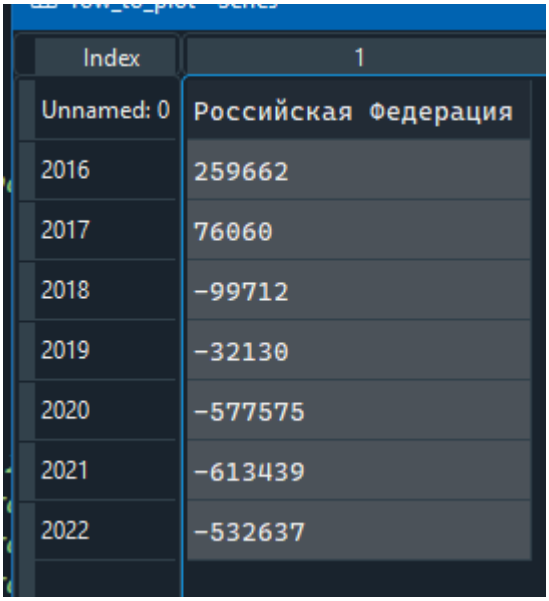
Метод data.rename дает возможность переименовать название наших столбцов. В данном случае мы убираем большое кол-во цифр, которые содержатся в названии как файла, так и столбцов.

Строка row_to_plot = data.iloc[1] дает возможность выбирать строку с индексом 1 в нашем DataFrame, таким образом при визуализации мы сможем выбрать данные по годам для определенного города (для данного файла). Важно отметить, что после того, как вы используете данный метод, выбрав одну строку, ваш DataFrame транспонируется, вследствие чего в нем будет не массив данных с двумя строчками, где верхняя будет содержать годы, а нижняя название области и значения, а будет массив данных с двумя столбцами, где левый будет отображать года, а правый числовые значения с данными по области. Для более хорошего понимания взгляните на рисунки 5 и 6.



Index	Unnamed: 0	2016	2017	2018	2019	2020	2021	2022
0	Год	2016	2017	2018	2019	2020	2021	2022
1	Российская Федерация	259662	76060	-99712	-32130	-577575	-613439	-532637
2	Центральный федеральный округ	105263	101831	66646	55497	-182596	-146560	-57776
3	Белгородская область	2728	-2989	-2458	1733	-7892	-9342	-21939
4	Брянская область	-5211	-9548	-10795	-7696	-9809	-13911	-12130
5	Владимирская область	-7569	-11262	-12532	-7389	-16317	-18440	-16725
6	Воронежская область	1931	-1640	-5947	-3616	-18597	-17930	-17335
7	Ивановская область	-6668	-8524	-10466	-7045	-10103	-10114	-9389
8	Калужская область	4798	-2414	-2776	-6805	-1595	11864	-2399
9	Костромская область	-3293	-4833	-6057	-3882	-4962	-7647	-6096
10	Курская область	2874	-7656	-8196	-3033	-7520	-12904	-11122
11	Липецкая область	128	-6020	-6166	-4664	-11179	-14512	-11877
12	Московская область	104823	79915	96262	91216	17636	60379	49479
13	Орловская область	-4905	-7569	-7780	-5969	-8812	-10592	-9747
14	Рязанская область	-3364	-5265	-7337	-5290	-10590	-13105	-9661
15	Смоленская область	-5429	-3853	-6985	-7474	-13762	-11271	-13859

Рисунок 5 — DataFrame до использования строки row_to_plot = data.iloc[1]



Index	1
Unnamed: 0	Российская Федерация
2016	259662
2017	76060
2018	-99712
2019	-32130
2020	-577575
2021	-613439
2022	-532637

Рисунок 6 — DataFrame после использования строки row_to_plot = data.iloc[1]

Далее используем строку row_to_plot.drop('Unnamed: 0', axis=0, inplace=True), чтобы убрать верхнюю стоку, т.к. она не даст визуализировать результаты.

Строка row_to_plot = row_to_plot.astype(float) дает возможность преобразовать значения в числовой формат, если они не числа.

И данная часть кода отвечает за построение самого графика:

```
    plt.figure(figsize=(10, 6))
    plt.plot(row_to_plot.index, row_to_plot.values, marker='o') # Построение графика с маркерами для каждой точки
    plt.xlabel('Года')
    plt.ylabel('Значения')
    plt.title('График временного ряда')
    plt.grid(True)
    plt.show()
```

Результат визуализации представлен на рисунке 7.

Рисунок 7 — Результат визуализации данных для всей РФ.

Таким образом, меняя цифру в строке row_to_plot = data.iloc[1] мы можем далее визуализировать данные для других регионов.

Сделайте выводы на основе данных, которые вы получите на своих файлах, скачанных с Росстата, и покажите их преподавателю.

Задание

1) Посчитать ВВП за 100-150лет лет во Франции, Великобритании, США, Австралии, Португалию, Испанию, Австрию, Бельгию, Канада , РФ, Китай любая страна

2) Курсы основных индексов этой страны, построить график,

3) Курсы основных индексов этой страны выраженный в золоте, построить график ,

4) Цены на железо (нефть , другой актив)за это время построить график

5) график развития технологий за этот период (любой)

6) графики солнечной активности (или другой активности за пределами Земли)

7) Графики построить сначала свои, потом совместить и посмотреть есть ли циклы