**Выполнил: Иванников Артём**

**Группа: БИ-2209**

1. **Для вас загружены файлы, с расширениями которых вы прежде не сталкивались. Откройте их любым удобным для вас способом и сохраните датафреймы в формате .xlsx (excel).**
2. **Создайте 4 папки (Kazakhstan, Kyrgyzstan, Uzbekistan, Tajikistan), сгруппировав их по годам.**
3. **Посчитать все формулы, указанные в pdf файле «Задание 5»**

**Начать лучше всего с переменной «children weight»**

**Использовать библиотеки – pandas, numpy, dask**

1. **Согласно статье «Food\_Security\_in\_Latin\_America.pdf» проведите аналитику изменения трендов продовольственной безопасности для каждой страны отдельно, а под конец – объедините все ваши данные по отдельным странам в один датасет и назовите его «Центральная Азия» и проведите анализ изменения продовольственной безопасности с визуализацией.**

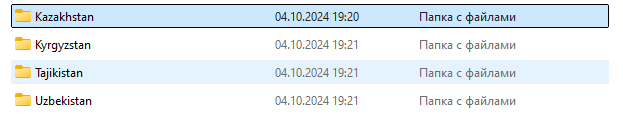
**Заметки: dask.visualize() – обязательно к выполнению на юпитере**

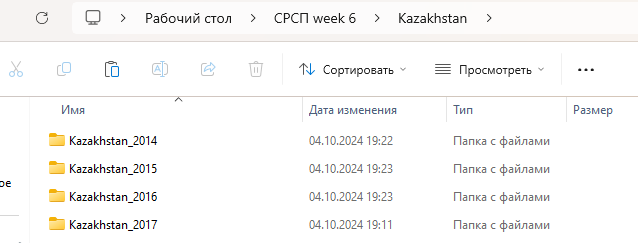
1. **Время обработки - %% time каждой библиотеки, разница, что быстрее и почему.**

**1 шаг:**

1. import os
2. import pandas as pd
3. import pyreadstat
4. import pyreadr
5. # Функция для обработки файлов и сохранения в .xlsx
6. def process\_files(folder\_path):
7. # Ищем файлы с нужными расширениями
8. for root, dirs, files in os.walk(folder\_path):
9. for file in files:
10. file\_path = os.path.join(root, file)
11. file\_name, file\_extension = os.path.splitext(file)
12. # Обработка .dta файлов
13. if file\_extension == ".dta":
14. df, meta = pyreadstat.read\_dta(file\_path)
15. output\_path = os.path.join(root, f"{file\_name}.xlsx")
16. df.to\_excel(output\_path, index=False)
17. print(f"Сохранен файл: {output\_path}")
18. # Обработка .rdata файлов
19. elif file\_extension == ".rdata":
20. result = pyreadr.read\_r(file\_path)
21. df = list(result.values())[0]  # получаем датафрейм
22. output\_path = os.path.join(root, f"{file\_name}.xlsx")
23. df.to\_excel(output\_path, index=False)
24. print(f"Сохранен файл: {output\_path}")
25. # Обработка .sav файлов
26. elif file\_extension == ".sav":
27. df, meta = pyreadstat.read\_sav(file\_path)
28. output\_path = os.path.join(root, f"{file\_name}.xlsx")
29. df.to\_excel(output\_path, index=False)
30. print(f"Сохранен файл: {output\_path}")
31. root\_folder = r'C:\Users\BG\Desktop\СРСП week 6'
32. process\_files(root\_folder)

**2 шаг: Создал 4 папки Kazakhstan, Kyrgyzstan, Uzbekistan, Tajikistan, в каждой из этих папок есть 4 папки по годам ( 2014, 2015 , 2016, 2017 )**

****

****

**3 шаг:**

import os

import pandas as pd

import dask.dataframe as dd

# Функция для обработки каждого Excel файла

def process\_excel\_file(file\_path):

    # Загружаем данные с помощью pandas

    df = pd.read\_excel(file\_path)

    # Преобразуем в Dask DataFrame

    ddf = dd.from\_pandas(df, npartitions=1)

    # Проверка наличия необходимых столбцов

    required\_columns = ['wt', 'N\_adults', 'N\_child', 'Prob\_Mod\_Sev', 'Prob\_sev']

    for col in required\_columns:

        if col not in ddf.columns:

            raise ValueError(f"Столбец {col} отсутствует в данных файла {file\_path}")

    # Преобразование данных в числовые значения

    for col in ['wt', 'N\_adults', 'N\_child']:

        ddf[col] = dd.to\_numeric(ddf[col], errors='coerce')  # Преобразуем с заменой ошибок на NaN

    # Удаляем строки с NaN значениями в критически важных столбцах

    ddf = ddf.dropna(subset=['wt', 'N\_adults', 'N\_child', 'Prob\_Mod\_Sev', 'Prob\_sev'])

    # 1. Рассчитываем переменную children weight

    ddf['children\_weight'] = ddf['wt'] / ddf['N\_adults'] \* ddf['N\_child']

    # 2. Рассчитываем F\_mod+sev\_ad и F\_sev\_ad (для взрослых)

    F\_mod\_sev\_ad = (ddf['Prob\_Mod\_Sev'] \* ddf['wt']).sum() / ddf['wt'].sum()

    F\_sev\_ad = (ddf['Prob\_sev'] \* ddf['wt']).sum() / ddf['wt'].sum()

    # 3. Рассчитываем F\_mod+sev\_child и F\_sev\_child (для детей)

    F\_mod\_sev\_child = (ddf['Prob\_Mod\_Sev'] \* ddf['children\_weight']).sum() / ddf['children\_weight'].sum()

    F\_sev\_child = (ddf['Prob\_sev'] \* ddf['children\_weight']).sum() / ddf['children\_weight'].sum()

    # 4. Рассчитываем общие значения для всех возрастов

    Pop\_ad = ddf['wt'].sum()

    Pop\_child = ddf['children\_weight'].sum()

    Pop\_mod\_sev\_ad = F\_mod\_sev\_ad.compute() \* Pop\_ad.compute()

    Pop\_mod\_sev\_child = F\_mod\_sev\_child.compute() \* Pop\_child.compute()

    F\_mod\_sev\_tot = (Pop\_mod\_sev\_ad + Pop\_mod\_sev\_child) / (Pop\_ad.compute() + Pop\_child.compute())

    Pop\_sev\_ad = F\_sev\_ad.compute() \* Pop\_ad.compute()

    Pop\_sev\_child = F\_sev\_child.compute() \* Pop\_child.compute()

    F\_sev\_tot = (Pop\_sev\_ad + Pop\_sev\_child) / (Pop\_ad.compute() + Pop\_child.compute())

    # Возвращаем результаты для текущего файла

    return {

        'F\_mod\_sev\_ad': F\_mod\_sev\_ad.compute(),

        'F\_sev\_ad': F\_sev\_ad.compute(),

        'F\_mod\_sev\_child': F\_mod\_sev\_child.compute(),

        'F\_sev\_child': F\_sev\_child.compute(),

        'F\_mod\_sev\_tot': F\_mod\_sev\_tot,

        'F\_sev\_tot': F\_sev\_tot

    }

# Основная функция для обхода всех папок и обработки всех файлов

def process\_all\_files(root\_folder):

    results = []

    # Проходим по странам и годам

    for country in os.listdir(root\_folder):

        country\_path = os.path.join(root\_folder, country)

        # Проходим по годам в каждой стране

        if os.path.isdir(country\_path):

            for year in os.listdir(country\_path):

                year\_path = os.path.join(country\_path, year)

                if os.path.isdir(year\_path):

                    # Ищем Excel файл в папке года

                    for file in os.listdir(year\_path):

                        if file.endswith('.xlsx'):

                            file\_path = os.path.join(year\_path, file)

                            print(f"Обработка файла: {file\_path}")

                            # Обрабатываем файл и сохраняем результаты

                            result = process\_excel\_file(file\_path)

                            result['country'] = country

                            result['year'] = year

                            results.append(result)

    # Преобразуем результаты в DataFrame и сохраняем в Excel

    df\_results = pd.DataFrame(results)

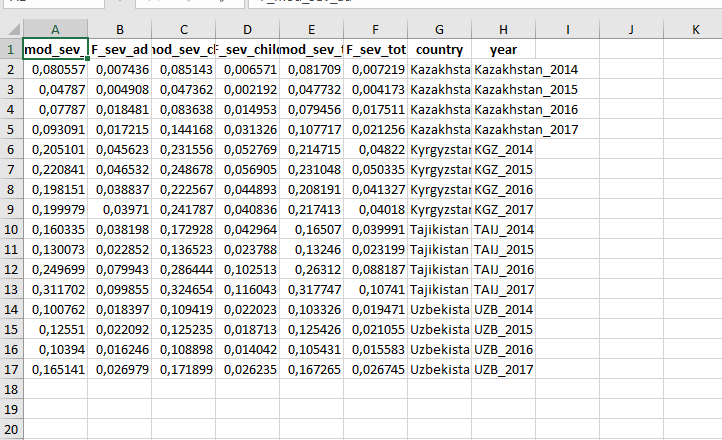
    df\_results.to\_excel('aggregated\_results.xlsx', index=False)

    print("Все результаты сохранены в aggregated\_results.xlsx")

# Укажите путь к корневой папке, где находятся страны

root\_folder = r'C:\Users\BG\Desktop\СРСП week 6'

process\_all\_files(root\_folder)

****

**4 шаг :**

# Импорт необходимых библиотек

import pandas as pd

import dask.dataframe as dd

import matplotlib.pyplot as plt

import seaborn as sns

# Путь к вашему файлу

file\_path = r"C:\Users\BG\Desktop\СРСП week 6\aggregated\_results.xlsx"

# Используем pandas для загрузки данных

df\_pandas = pd.read\_excel(file\_path)

# Используем pandas для загрузки данных, а затем преобразуем в dask

df\_dask = dd.from\_pandas(df\_pandas, npartitions=4)

# Анализ для каждой страны отдельно (pandas)

def analyze\_pandas(df):

    countries = df['country'].unique()

    trends = {}

    for country in countries:

        country\_data = df[df['country'] == country]

        # Вычисляем средние значения только для числовых столбцов

        trends[country] = country\_data.mean(numeric\_only=True)

    return trends

# Анализ для каждой страны отдельно (dask)

def analyze\_dask(df):

    countries = df['country'].unique().compute()

    trends = {}

    for country in countries:

        country\_data = df[df['country'] == country].compute()

        # Вычисляем средние значения только для числовых столбцов

        trends[country] = country\_data.mean(numeric\_only=True)

    return trends

# Объединение данных по годам, учитывая только числовые столбцы

def combine\_data(df):

    df\_combined = df.groupby('year').mean(numeric\_only=True)

    return df\_combined

# Выполним анализ с помощью pandas

%time trends\_pandas = analyze\_pandas(df\_pandas)

# Выполним анализ с помощью dask

%time trends\_dask = analyze\_dask(df\_dask)

# Объединение с pandas

df\_combined\_pandas = combine\_data(df\_pandas)

# Проверка имен столбцов

print("Columns in combined dataframe:", df\_combined\_pandas.columns)

# Проверка первых нескольких строк датафрейма

print(df\_combined\_pandas.head())

# Объединение с dask

df\_combined\_dask = combine\_data(df\_dask.compute())

# Визуализация трендов с использованием matplotlib

plt.figure(figsize=(12, 6))  # Увеличиваем размер графика

# Используем правильное имя столбца для y

sns.lineplot(data=df\_combined\_pandas, x='year', y='F\_mod\_sev\_tot')  # Замените на правильное имя столбца

plt.title('Изменение продовольственной безопасности (модерированная тяжесть)')

plt.xlabel('Год')

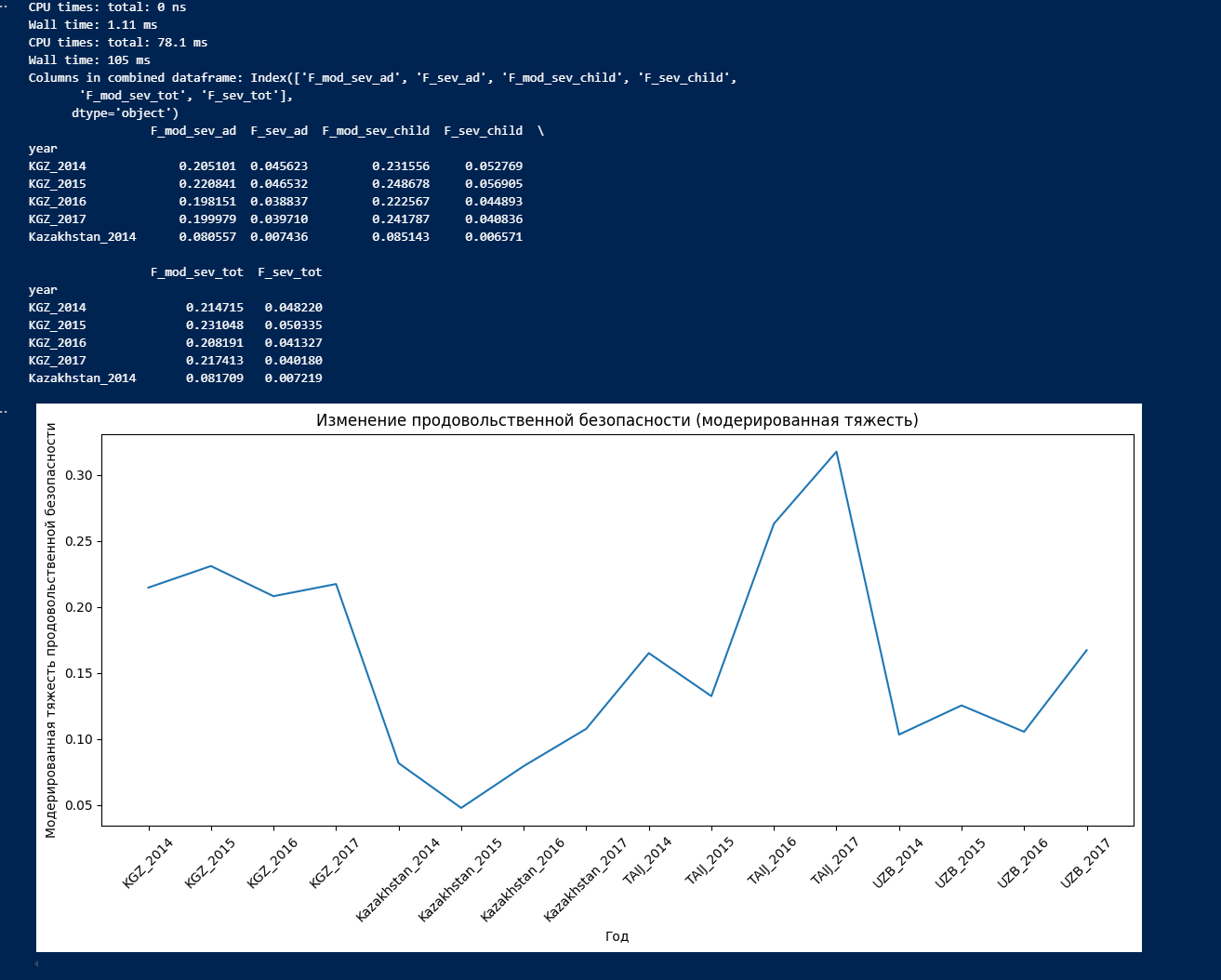
plt.ylabel('Модерированная тяжесть продовольственной безопасности')

# Поворачиваем метки на оси X

plt.xticks(rotation=45, fontsize=10)  # Повернуть метки на 45 градусов и изменить размер шрифта

plt.tight\_layout()  # Автоматическая подгонка графика

plt.show()

****

**5 шаг :**

**Разница в производительности:**

**pandas**: Хорошо подходит для небольших и средних наборов данных. Он загружает данные в память и выполняет операции непосредственно на всех данных. Если данные помещаются в память, это обычно быстрее.

**dask**: Проектирует работу с большими наборами данных, которые могут не поместиться в память. Он разбивает данные на более мелкие части и обрабатывает их параллельно. В некоторых случаях это может быть медленнее, особенно для небольших наборов данных из-за накладных расходов на распределение.