



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Компьютерные системы и сети

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01 Информатика и вычислительная техника

МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/05 Современные интеллектуальные программно-аппаратные комплексы

ОТЧЕТ

по домашнему заданию № 1

Название Дескриптивный анализ данных

Дисциплина Методы машинного обучения

Студент гр. ИУ6-21М

(Подпись, дата)

А. А. Куценко

(И.О.Фамилия)

Преподаватель

(Подпись, дата)

С. Ю. Папулин

(И.О.Фамилия)

Москва, 2025

Домашнее задание 1. **Дескриптивный анализ данных**

Куценко А. А (ftruf357ft@gmail.com)

Подключение стилей оформления

```
In [1]: %%html
<link href="css/style.css" rel="stylesheet" type="text/css">
```

Цель работы

Приобрести опыт решения практических задач по анализу данных, таких как загрузка, трансформация, вычисление простых статистик и визуализация данных в виде графиков и диаграмм, посредством языка программирования Python.

Вариант

```
In [2]: surname = "Куценко" # Ваша фамилия

alp = 'абвгдеёжзийклмнопрстуфхцщъыьэюя'
w = [1, 42, 21, 21, 34, 6, 44, 26, 18, 44, 38, 26, 14, 43, 4, 49, 45,
      7, 42, 29, 4, 9, 36, 34, 31, 29, 5, 30, 4, 19, 28, 25, 33]

d = dict(zip(alp, w))
variant = sum([d[el] for el in surname.lower()]) % 40 + 1

print("Задача № 1, шаг 5 - вариант: ", variant % 5 + 1)
print("Задача № 1, шаг 11 - вариант: ", variant % 2 + 1)
print("задача № 2 - вариант: ", variant % 4 + 1)
```

Задача № 1, шаг 5 - вариант: 1
Задача № 1, шаг 11 - вариант: 1
задача № 2 - вариант: 3

Задание 1. Анализ индикаторов качества государственного управления (The Worldwide Government Indicators, WGI) (6 баллов)

```
In [3]: %pwd
```

Loading [MathJax]/extensions/Safe.js

Out[3]: '/home/alexander/Programming/ML/MY/mlassignments/notebooks '




Условие

В качестве индикатора далее необходимо использовать контроль над коррупцией (Control of Corruption) и его показатели `pctrank` и `estimate`.

- Наборы данных:
 - [WGI](#)
 - [Регионы](#)
- [Описание WGI](#)

Замечание. Исходный файл с данными редактировать нельзя.

Порядок работы:

1. Загрузите данные в DataFrame
 2. Отсортируйте данные по убыванию `pctrank`
 3. Отобразите данные по индексу WGI за 2023 год в виде горизонтального столбчатого графика (`pctrank`). Примерный вид графика приведен ниже.
 Сортировка WGI по Control of Corruption
 4. Сформируйте DataFrame из исходного для региона в соответствии с Вашим вариантом
 5. Выведите данные DataFrame'a
 6. Постройте графики индекса WGI за 1996-2023 для стран своего региона (`estimate`). Примерный вид графика приведен ниже
 WGI для стран из региона
 7. Найдите страны с наибольшим и наименьшим значением WGI Вашего варианта региона за 2023 год (estimate)
- Замечание.** У нескольких стран может быть одна и та же позиция в рейтинге из-за одинаковых значений индекса
8. Определите средние значения региона за каждый год в период с 1996 по 2023 (`estimate`)
 9. Постройте графики индекса WGI за 1996-2023 для стран своего региона и выделите страны с наибольшим и наименьшим значением WGI за 2023 год, а также отобразите среднее значение по региону и РФ. Примерный вид графика:  WGI среднее, максимальное, минимальное и по РФ
 10. Определите, как изменилось значение показателя rank с 1996 по 2023 (`rank`)
 11. Выведите таблицу для Вашего варианта (WGI - `rank`)
 12. Отобразите диаграмму размаха (`boxplot`) индекса WGI за 2023 для всех стран и для каждого региона в отдельности (на одном графике) (`estimate`)

Loading [MathJax]/extensions/Safe.js

Выполнение

```
In [4]: # Базовый уровень бытия
import os
import numpy as np
import pandas as pd
```

Проверяем наличие датасета локально, если его нет, то скачиваем

```
In [5]: %%bash

# Messages from utils in stderr is not an error
exec 2>&1

datasets_path="./data/A1_DA_dataset"
wgi_path="wgidataset"
wgi_file="wgidataset.xlsx"
regions_file="regions.xlsx"

echo "Let's check the availability of the dataset locally"

if [ ! -f ${datasets_path}/${wgi_path}/${wgi_file} ] || [ ! -f ${datasets_path}/${regions_file} ]; then
    # TODO: need unzip in docker image?
    apt update && apt install unzip

    echo "Dataset files are missing. Downloading..."
    mkdir -p ${datasets_path}/${wgi_path}

    pushd ${datasets_path}

    echo "Downloading dataset"
    curl -s "https://www.worldbank.org/content/dam/sites/govindicators/doc/wgidataset.zip" -o ${wgi_path}.zip
    unzip ${wgi_path}.zip -d ${wgi_path}
    rm ${wgi_path}.zip

    echo "Downloading regions"
    curl -sL "https://github.com/MLMethods/Assignments/raw/refs/heads/master/regions.xlsx" -o ${regions_file}

    popd
else
    echo "Dataset files exist. I'm skipping the download"
fi
```

Let's check the availability of the dataset locally
Dataset files exist. I'm skipping the download

1. Подгружаем датасет

```
In [6]: dataset_path = './data/A1_DA_dataset/wgidataset/wgidataset.xlsx'

if not os.path.exists(dataset_path):
    raise FileNotFoundError(f'File with dataset {dataset_path} not found!')

# Empty values in the dataset are marked as ".."
name = pd.read_excel(dataset_path, na_values=['..'])
```

Loading [MathJax]/extensions/Safe.js

dataset_frame

Out[6]:

	codeindyr	code	countryname	year	indicator	estimate	stddev	nsource
0	AFGcc1996	AFG	Afghanistan	1996	cc	-1.291705	0.340507	2.0
1	ALBcc1996	ALB	Albania	1996	cc	-0.893903	0.315914	3.0
2	DZAcc1996	DZA	Algeria	1996	cc	-0.566741	0.262077	4.0
3	ASMcc1996	ASM	American Samoa	1996	cc	NaN	NaN	NaN
4	ADOcc1996	ADO	Andorra	1996	cc	1.318143	0.480889	1.0
...
32095	VIRva2023	VIR	Virgin Islands (U.S.)	2023	va	NaN	NaN	NaN
32096	WBGva2023	WBG	West Bank and Gaza	2023	va	-1.118067	0.149837	6.0
32097	YEMva2023	YEM	Yemen, Rep.	2023	va	-1.550217	0.131432	8.0
32098	ZMBva2023	ZMB	Zambia	2023	va	-0.047946	0.118482	12.0
32099	ZWEva2023	ZWE	Zimbabwe	2023	va	-1.092633	0.118235	13.0

32100 rows × 11 columns

2. Отсортируем по убыванию значения `pctrank`

In [7]: dataset_frame.sort_values('pctrank', ascending=0)

Out[7]:

	codeindyr	code	countryname	year	indicator	estimate	stddev	nsource
383	SGPge1996	SGP	Singapore	1996	ge	1.993047	0.168259	5.0
1667	SGPge1998	SGP	Singapore	1998	ge	2.072397	0.205460	5.0
17907	NORva2012	NOR	Norway	2012	va	1.728163	0.131552	13.0
27989	SGPrq2020	SGP	Singapore	2020	rq	2.205299	0.208772	8.0
4877	SGPrq2002	SGP	Singapore	2002	rq	1.888891	0.208489	7.0
...
32008	MTQva2023	MTQ	Martinique	2023	va	NaN	NaN	NaN
32024	ANTva2023	ANT	Netherlands Antilles (former)	2023	va	NaN	NaN	NaN
32029	NIUva2023	NIU	Niue	2023	va	NaN	NaN	NaN
32047	REUva2023	REU	Réunion	2023	va	NaN	NaN	NaN
32095	VIRva2023	VIR	Virgin Islands (U.S.)	2023	va	NaN	NaN	NaN

32100 rows × 11 columns



3. Из всего датасета нас интересует показатель **Control of Corruption**, обозначенный в поле **indicator** как **cc**. График нужно построить за 2023 год

```
In [8]: # Датафрейм с строками для cc и 2023 годом (NaN значение не отрисовываем)
wgi_cc_2023_frame = dataset_frame\
    .query('indicator == "cc" and year == 2023')\
    .filter(items=['countryname', 'year', 'indicator', 'pctr
    .dropna()
wgi_cc_2023_frame
```

Out[8]:

	countryname	year	indicator	pctrank
30816	Afghanistan	2023	cc	13.679245
30817	Albania	2023	cc	43.396225
30818	Algeria	2023	cc	30.188679
30819	American Samoa	2023	cc	87.735847
30820	Andorra	2023	cc	87.735847
...
31025	Virgin Islands (U.S.)	2023	cc	53.301888
31026	West Bank and Gaza	2023	cc	26.415094
31027	Yemen, Rep.	2023	cc	1.886792
31028	Zambia	2023	cc	36.792454
31029	Zimbabwe	2023	cc	9.905661

213 rows × 4 columns

```
In [9]: wgi_cc_2023_pctranks = wgi_cc_2023_frame['pctrank'].unique()
wgi_cc_2023_pctranks = np.flip(np.sort(wgi_cc_2023_pctranks, axis=-1))

# Получили отсортированный по убыванию массив уникальных значений pctrank
print('Top 5 values:')
print(wgi_cc_2023_pctranks[:5])

print('5 lowest values:')
print(wgi_cc_2023_pctranks[-5:])
```

Top 5 values:
 [100. 99.52830505 99.05660248 98.58490753 98.11320496]
 5 lowest values:
 [1.88679242 1.41509438 0.94339621 0.47169811 0.]

```
In [10]: # каждому уникальному значению из wgi_cc_2023_pctranks соответствует положение
position_in_rating = {}
counter = 1
for value in wgi_cc_2023_pctranks:
    position_in_rating[value] = counter
    counter += 1
```

```
In [11]: print('5 top values:')
print(list(position_in_rating.items())[:5])

print('5 lowest values')
print(list(position_in_rating.items())[-5:])
```

5 top values:

```
[(100.0, 1), (99.52830505371094, 2), (99.05660247802734, 3), (98.58490753173828, 4), (98.11320495605469, 5)]
```

5 lowest values

```
[(1.8867924213409424, 199), (1.4150943756103516, 200), (0.9433962106704712, 201), (0.4716981053352356, 202), (0.0, 203)]
```

```
In [12]: # Новый столбец в датафрейме с позицией в рейтинге за 2023 год
wgi_cc_2023_position_values = []

for index, row in wgi_cc_2023_frame.iterrows():
    position_value = position_in_rating[row.pctrank]
    wgi_cc_2023_position_values.append(position_value)

wgi_cc_2023_frame['position'] = wgi_cc_2023_position_values

wgi_cc_2023_frame.sort_values('position', ascending=1)
```

```
Out[12]:
```

	countryname	year	indicator	pctrank	position
30868	Denmark	2023	cc	100.000000	1
30881	Finland	2023	cc	99.528305	2
30961	Norway	2023	cc	99.056602	3
30955	New Zealand	2023	cc	98.584908	4
30985	Singapore	2023	cc	98.113205	5
...
31027	Yemen, Rep.	2023	cc	1.886792	199
31023	Venezuela, RB	2023	cc	1.415094	200
30989	Somalia	2023	cc	0.943396	201
31001	Syrian Arab Republic	2023	cc	0.471698	202
30991	South Sudan	2023	cc	0.000000	203

213 rows × 5 columns

```
In [13]: # продвинутый уровень бытия
import matplotlib.pyplot as plt
```

```
In [14]: # Значения и подписи на будущей гистограмме
countries_cc_values = []
countries_names = []

for index, row in wgi_cc_2023_frame\
    .sort_values('position', ascending=0)\
    .iterrows():
    countries_names.append(f'{row.position}.{row.countryname}')
    countries_cc_values.append(row.pctrank)
```

Loading [MathJax]/extensions/Safe.js


```

# print(f'{countries_names[-1]}:{countries_cc_values[-1]}')

for type in ['top', 'lowest']:
    sign = -1 if type == 'top' else 1
    # it is necessary to print the lowest possible rating
    penalty = 0 if type == 'top' else 1

    print(f'5 {type} countries:')
    for i in range(1, 6):
        index = sign * i - penalty
        print(f'{countries_names[index]}:{countries_cc_values[index]}')
    print()

```

5 top countries:
 1.Denmark:100.0
 2.Finland:99.52830505371094
 3.Norway:99.05660247802734
 4.New Zealand:98.58490753173828
 5.Singapore:98.11320495605469

5 lowest countries:
 203.South Sudan:0.0
 202.Syrian Arab Republic:0.4716981053352356
 201.Somalia:0.9433962106704712
 200.Venezuela, RB:1.4150943756103516
 199.Yemen, Rep.:1.8867924213409424

```

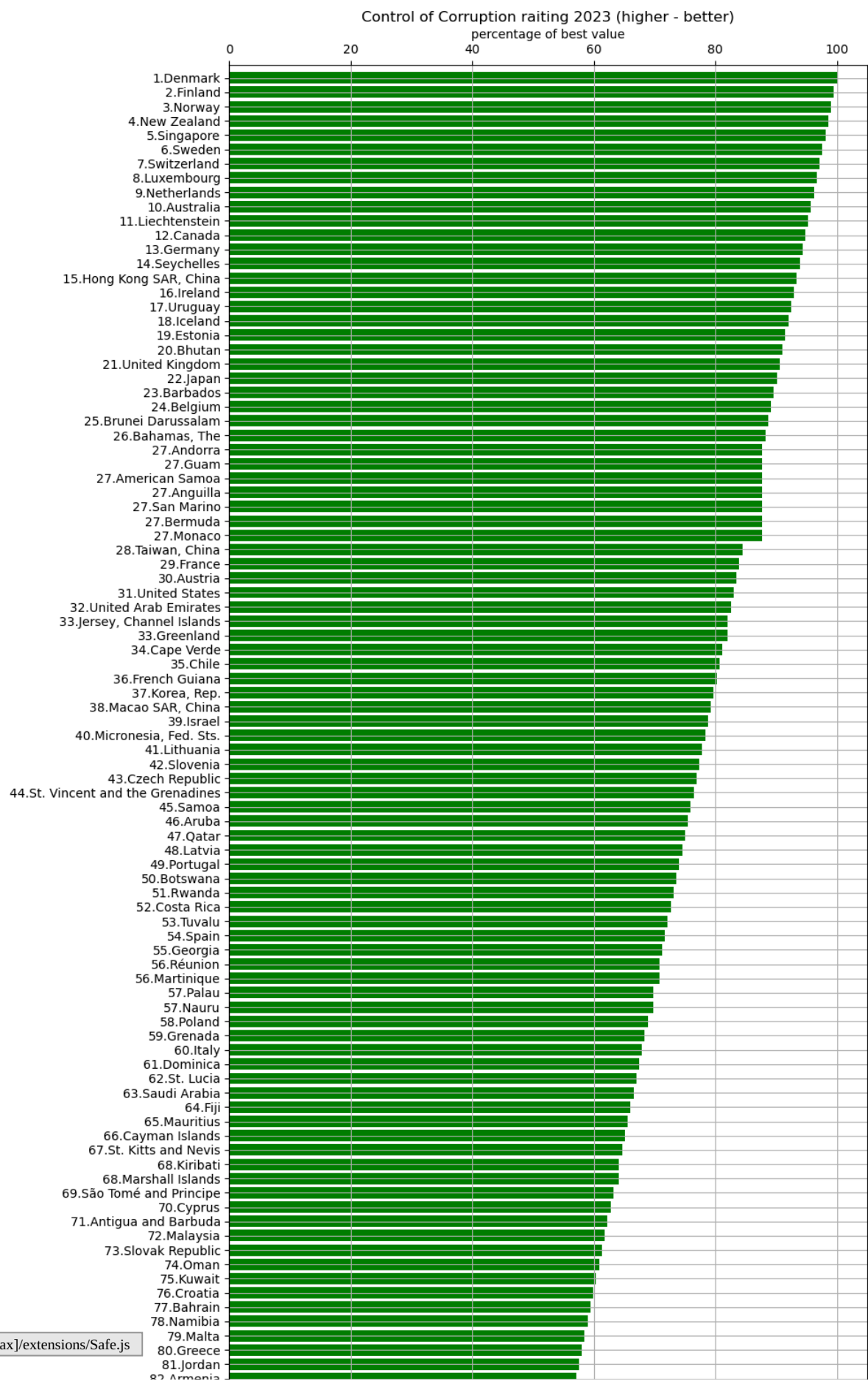
In [15]: plt.figure('pctrank rating', figsize=[10, len(countries_names) / 6]) # чем
plt.title('Control of Corruption rating 2023 (higher - better)')
plt.barh(countries_names, countries_cc_values, color='green')

ax = plt.gca() # текущая ось
ax.margins(y = 0.0025) # уменьшаем вертикальный отступ в %
ax.xaxis.set_label_text('percentage of best value')
ax.xaxis.set_label_position('top')
ax.xaxis.tick_top()

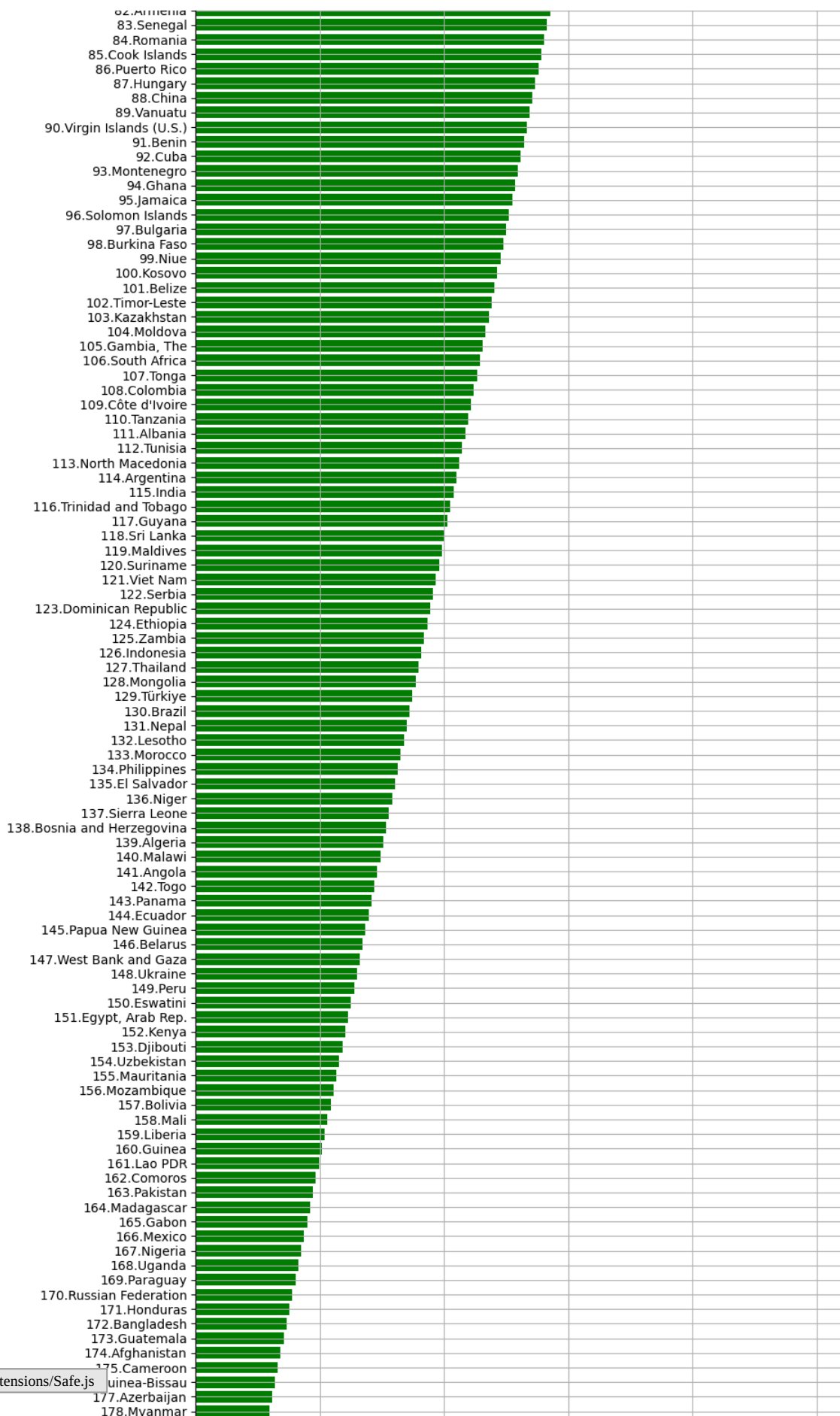
plt.tight_layout()
plt.grid(True)

plt.show()

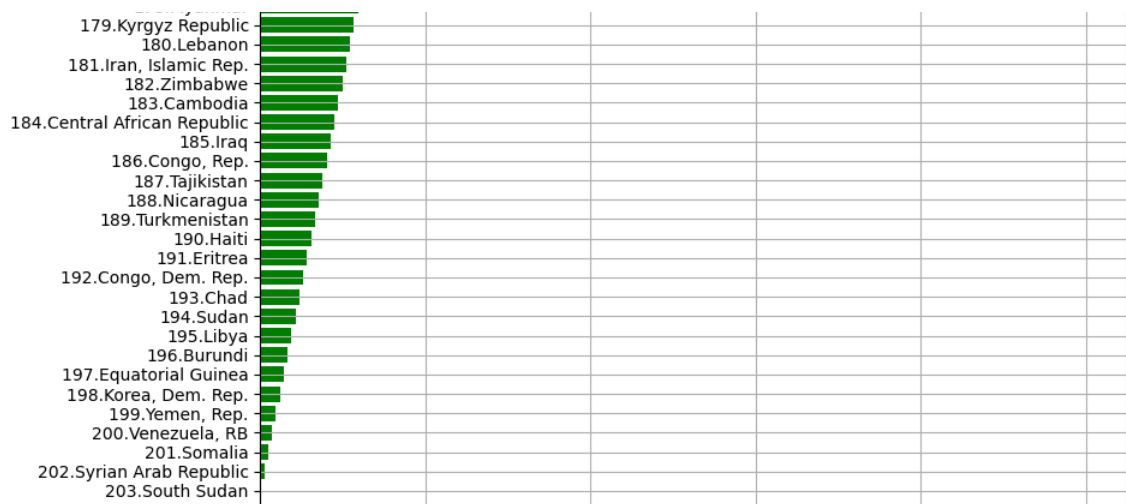
```



Loading [MathJax]/extensions/Safe.js



Loading [MathJax]/extensions/Safe.js



4. Для варианта 1 регион - Asia Pacific

In [16]: `import math`

```
In [17]: regions_path = '../data/A1_DA_dataset/wgidataset/regions.xlsx'

if not os.path.exists(regions_path):
    raise FileNotFoundError(f'File with dataset {regions_path} not found!')

regions_frame = pd.read_excel(regions_path, na_values=['..'])

# Исправляем неправильные названия стран для Азии
names = {
    # wrong      : right
    'Hong Kong' : 'Hong Kong SAR, China',
    'Korea, North' : 'Korea, Dem. Rep.',
    'Korea, South' : 'Korea, Rep.',
    'Laos' : 'Lao PDR',
    'Taiwan' : 'Taiwan, China',
    'Timor-Leste' : 'Timor-Leste',
    'Vietnam' : 'Viet Nam',
}

for wrong, right in names.items():
    regions_frame.loc[regions_frame['Country'] == f'{wrong}', 'Country'] = f'{right}'

regions_frame
```

Out[17]:

	Country	Code	Region
0	Afghanistan	AFG	AP
1	Albania	ALB	ECA
2	Algeria	DZA	MENA
3	Angola	AGO	SSA
4	Argentina	ARG	AME
...
175	Venezuela	VEN	AME
176	Viet Nam	VNM	AP
177	Yemen	YEM	MENA
178	Zambia	ZMB	SSA
179	Zimbabwe	ZWE	SSA

180 rows × 3 columns

Список наименований стран с регионом AP (Asia Pacific):

```
In [18]: asia_pacific_countries = regions_frame.query('Region == "AP"').filter(items=
asia_pacific_countries.sort_values('Country'))
```

Out[18]:

	Country	Region
0	Afghanistan	AP
6	Australia	AP
11	Bangladesh	AP
16	Bhutan	AP
25	Cambodia	AP
31	China	AP
54	Fiji	AP
70	Hong Kong SAR, China	AP
73	India	AP
74	Indonesia	AP
81	Japan	AP
85	Korea, Dem. Rep.	AP
86	Korea, Rep.	AP
90	Lao PDR	AP
100	Malaysia	AP
101	Maldives	AP
108	Mongolia	AP
112	Myanmar	AP
114	Nepal	AP
116	New Zealand	AP
123	Pakistan	AP
125	Papua New Guinea	AP
128	Philippines	AP
143	Singapore	AP
146	Solomon Islands	AP
151	Sri Lanka	AP
157	Taiwan, China	AP
160	Thailand	AP
161	Timor-Leste	AP

Loading [MathJax]/extensions/Safe.js

	Country	Region
174	Vanuatu	AP
176	Viet Nam	AP

5. Вот все данные из исходного датасета для стран из региона Asia Pacific

```
In [19]: asia_pacific_frame = dataset_frame[\
        dataset_frame.countryname.isin(asia_pacific_countries)
        ].query('indicator == "cc").dropna()
asia_pacific_frame
```

```
Out[19]:
```

	codeindyr	code	countryname	year	indicator	estimate	stddev	nsource
0	AFGcc1996	AFG	Afghanistan	1996	cc	-1.291705	0.340507	2.0
11	AUScc1996	AUS	Australia	1996	cc	1.877356	0.210325	6.0
16	BGDcc1996	BGD	Bangladesh	1996	cc	-0.969682	0.262077	4.0
23	BTNcc1996	BTN	Bhutan	1996	cc	0.942838	0.340507	2.0
32	KHMcc1996	KHM	Cambodia	1996	cc	-1.019842	0.275614	3.0
...
31003	TWNcc2023	TWN	Taiwan, China	2023	cc	1.203369	0.178170	9.0
31006	THAcc2023	THA	Thailand	2023	cc	-0.489051	0.160341	10.0
31007	TMPcc2023	TMP	Timor-Leste	2023	cc	-0.226704	0.207288	7.0
31022	VUTcc2023	VUT	Vanuatu	2023	cc	-0.014865	0.257371	6.0
31024	VNMcc2023	VNM	Viet Nam	2023	cc	-0.415814	0.157724	10.0

772 rows × 11 columns

```
In [20]: # Новый датафрейм для будущего графика, первый столбец - наименование стран
asia_pacific_cc = pd.DataFrame(data=asia_pacific_countries.Country)

for year in range(1996, 2024):
    year_data = []
    for country in asia_pacific_countries.Country:
        value = asia_pacific_frame\
            .query(f'countryname == "{country}" and year == {year}')\
            .filter(items=['estimate'])
        if value.empty:
            year_data.append(math.nan)
        else:
            year_data.append(value.iloc[0,0])
    asia_pacific_cc[f'{year}'] = year_data
```

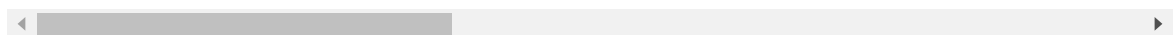
Loading [MathJax]/extensions/Safe.js

```
# Более удобное представление для графика  
asia_pacific_cc = asia_pacific_cc.dropna(axis=1, how='all')  
asia_pacific_cc = asia_pacific_cc.set_index('Country').T  
asia_pacific_cc
```


Out[20]:

Country	Afghanistan	Australia	Bangladesh	Bhutan	Cambodia	China	Fiji
1996	-1.291705	1.877356	-0.969682	0.942838	-1.019842	-0.271190	0.659301
1998	-1.176012	1.798130	-0.773011	0.883641	-0.988312	-0.353955	0.663790
2000	-1.271724	1.862088	-1.212083	0.574340	-0.967183	-0.208549	0.630551
2002	-1.251137	1.761436	-1.449087	0.449922	-0.990784	-0.557898	0.610871
2003	-1.344180	1.895287	-1.541721	1.087011	-0.989836	-0.395265	0.276081
2004	-1.350647	2.005869	-1.597115	0.893403	-1.058346	-0.565062	0.290081
2005	-1.447252	1.942668	-1.406467	0.871917	-1.223740	-0.617888	-0.224621
2006	-1.446292	1.950813	-1.442983	0.854817	-1.260386	-0.518888	0.184131
2007	-1.613251	2.000873	-1.063240	0.928756	-1.151690	-0.599235	0.121461
2008	-1.672096	2.027343	-1.046788	0.912390	-1.241711	-0.526308	0.098341
2009	-1.552299	2.041763	-1.075251	0.937310	-1.181240	-0.519457	-0.154771
2010	-1.645391	2.023611	-1.054080	0.938382	-1.249008	-0.570010	-0.159891
2011	-1.600471	2.037909	-1.097389	0.854587	-1.255650	-0.514728	0.108831
2012	-1.430373	1.977507	-0.856619	0.953947	-1.080682	-0.438276	0.181171
2013	-1.445961	1.777833	-0.893966	0.912138	-1.064034	-0.355888	0.196481
2014	-1.364934	1.849272	-0.892609	1.305882	-1.147174	-0.337720	0.385081
2015	-1.354713	1.841308	-0.844677	0.989430	-1.153562	-0.299922	0.378061
2016	-1.540228	1.772252	-0.887519	1.089876	-1.301628	-0.268094	0.335581
2017	-1.530075	1.752975	-0.858603	1.525725	-1.316545	-0.291626	0.638821
2018	-1.502876	1.767455	-0.926823	1.590201	-1.357102	-0.286848	0.699951
2019	-1.419310	1.788074	-1.017210	1.572299	-1.323500	-0.312614	0.688051
2020	-1.493361	1.632981	-1.003712	1.618372	-1.272042	-0.071028	0.613361
2021	-1.152266	1.707389	-0.986505	1.507003	-1.197976	0.030234	0.443441
2022	-1.183684	1.764367	-1.075752	1.514782	-1.241612	0.016064	0.388871
2023	-1.154932	1.781205	-1.120866	1.531423	-1.299217	-0.005370	0.506271

25 rows × 31 columns



Loading [MathJax]/extensions/Safe.js

6. Строим графики для стран Asia Pacific

In [21]: `asia_pacific_cc.T.columns`

Out[21]: `Index(['1996', '1998', '2000', '2002', '2003', '2004', '2005', '2006', '2007',
'2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016',
'2017', '2018', '2019', '2020', '2021', '2022', '2023'],
dtype='object')`

```
In [22]: # Некоторые года в исходном датасете отсутствуют, поэтому нельзя просто взять
years = [ int(year) for year in asia_pacific_cc.T.columns[:] ]
# years

def plot_asia_cc(params={}):
    # Пример обращения к столбцу для Афганистана - вернется series для Афган
    # asia_pacific_cc['Afghanistan']

    plt.figure('CC in 1996-2023 in Asia Pacific region', figsize=[10, 6])
    plt.title('Control of Corruption rating 1996 - 2023 Asia Pacific (high')

    max_countries = params.get('max', {}).get('countries', [])
    max_color = params.get('max', {}).get('color', [])
    min_countries = params.get('min', {}).get('countries', [])
    min_color = params.get('min', {}).get('color', [])

    for country in asia_pacific_countries['Country']:
        color = 'lightgray'
        label = ''

        if country in max_countries:
            color = f'{max_color}'
            label = 'max'
        elif country in min_countries:
            color = f'{min_color}'
            label = 'min'

        plt.plot(years, asia_pacific_cc[f'{country}'], color=color, marker='o')

    # Обработываем все дополнительные данные
    additional = [key for key in params if key not in ['min', 'max']]
    for country in additional:
        data = params.get(f'{country}', {}).get('data', [])
        color = params.get(f'{country}', {}).get('color', 'red')
        label = f'{country}'

        plt.plot(years, data, color=color, marker='o', markersize=4, label=label)

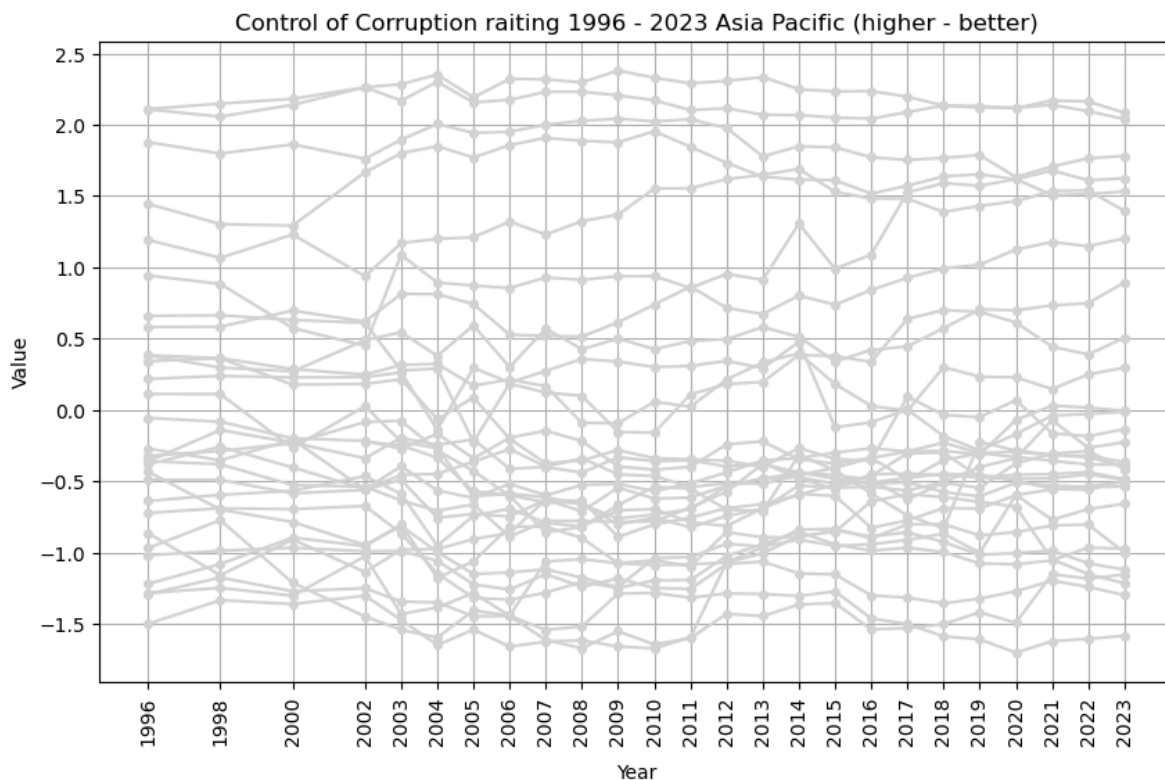
    if params:
        plt.legend(loc=1, fontsize=12)

    plt.grid(True)
    plt.xticks(years, rotation=90)
```

Loading [MathJax]/extensions/Safe.js

```
plt.xlabel('Year', labelpad=10)
plt.ylabel('Value')
plt.show()

plot_asia_cc()
```



7. Теперь найдем страны с максимальным и минимальным значением индекса

```
In [23]: # Удобнее забирать столбцы, а не строки, поэтому транспонируем датафрейм и з
year = years[-1]
asia_pacific_cc_2023_data = asia_pacific_cc.T[f'{year}']

asia_pacific_cc_2023 = {}
asia_pacific_cc_2023['min'] = asia_pacific_cc_2023_data.min()
asia_pacific_cc_2023['max'] = asia_pacific_cc_2023_data.max()
asia_pacific_cc_2023['mean'] = asia_pacific_cc_2023_data.mean()

print(f'All data for Asia Pacific for {year}')
print(f'{asia_pacific_cc_2023_data.sort_values()}\n')
print(f'{"Min value":<15}: {asia_pacific_cc_2023["min"]:>10}')
print(f'{"Max value":<15}: {asia_pacific_cc_2023["max"]:>10}')
print(f'{"Mean value":<15}: {asia_pacific_cc_2023["mean"]:>10}')
```

All data for Asia Pacific for 2023

```
Country
Korea, Dem. Rep.      -1.584476
Cambodia              -1.299217
Myanmar               -1.216499
Afghanistan           -1.154932
Bangladesh            -1.120866
Pakistan              -0.999357
Lao PDR               -0.973836
Papua New Guinea      -0.657154
Philippines           -0.537640
Nepal                 -0.507928
Mongolia              -0.492604
Thailand               -0.489051
Indonesia             -0.486872
Viet Nam              -0.415814
Maldives              -0.396943
Sri Lanka             -0.383977
India                 -0.366015
Timor-Leste           -0.226704
Solomon Islands       -0.134711
Vanuatu               -0.014865
China                 -0.005370
Malaysia              0.295621
Fiji                  0.506272
Korea, Rep.           0.894089
Taiwan, China         1.203369
Japan                 1.395661
Bhutan                1.531423
Hong Kong SAR, China  1.625145
Australia             1.781205
Singapore             2.040184
New Zealand           2.084095
Name: 2023, dtype: float64
```

```
Min value      : -1.584476351737976
Max value      :  2.0840954780578613
Mean value     : -0.0034764338164560257
```

```
In [24]: countries_max = asia_pacific_cc_2023_data[asia_pacific_cc_2023_data == asia_
countries_max
```

```
Out[24]: ['New Zealand']
```

```
In [25]: countries_min = asia_pacific_cc_2023_data[asia_pacific_cc_2023_data == asia_
countries_min
```

```
Out[25]: ['Korea, Dem. Rep.']
```

8. Найдем все средние значения региона Asia Pacific за 1996-2023 годы

```
In [26]: asia_pacific_means_1996_2023 = [asia_pacific_cc.T[f'{year}'].mean() for year
asia_pacific_means_1996_2023
```

Loading [MathJax]/extensions/Safe.js

```
Out[26]: [0.005390121166904767,
0.004409123708804448,
-0.03511409113804499,
-0.04934098553513327,
-0.03201416255004944,
-0.12487848872138609,
-0.15484253269049428,
-0.16874835952635733,
-0.1554863599519576,
-0.17264259486429154,
-0.1757663096631727,
-0.1606008369595774,
-0.15394322898599408,
-0.0850167356191143,
-0.06684924614044928,
-0.003774962117595057,
-0.06374600673875501,
-0.08608150404066808,
-0.041266466723755,
-0.027243216792421955,
-0.0244332232302235,
0.006628758243976101,
0.01564465403076141,
0.005998003206426098,
-0.0034764338164560257]
```

9. Подсветим на графике по `estimate` страну с наибольшим/наименьшим значениями, а также среднее по региону и Россию

```
In [27]: # Нужны данные по России за 1996-2023 год из исходного датасета
russia_frame = dataset_frame[dataset_frame.countryname == 'Russian Federation']
               .query('indicator == "cc"')\
               .dropna()\
               .filter(items=['year', 'estimate', 'pctrank']).set_index('year')
russia_frame['estimate']
```

```
Out[27]: year
1996    -1.053342
1998    -0.954374
2000    -0.943414
2002    -0.954848
2003    -0.783092
2004    -0.825626
2005    -0.847121
2006    -0.940848
2007    -1.017581
2008    -1.125229
2009    -1.141307
2010    -1.099215
2011    -1.074377
2012    -1.051572
2013    -1.020724
2014    -0.918883
2015    -0.974070
2016    -0.838650
2017    -0.914681
2018    -0.869340
2019    -0.822259
2020    -0.930568
2021    -0.921021
2022    -0.979840
2023    -1.104678
Name: estimate, dtype: float64
```

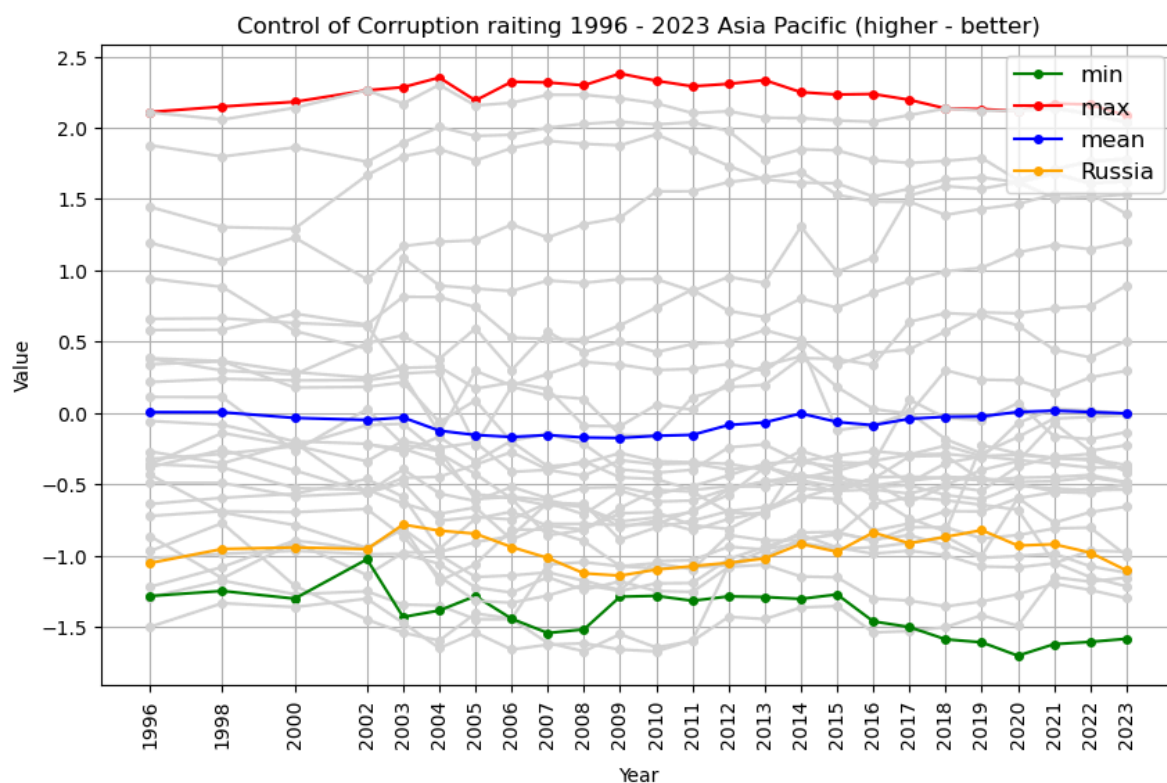
```
In [28]: params = {}
params['max'] = {}
params['max']['countries'] = countries_max
params['max']['color'] = 'red'

params['min'] = {}
params['min']['countries'] = countries_min
params['min']['color'] = 'green'

params['mean'] = {}
params['mean']['data'] = asia_pacific_means_1996_2023
params['mean']['color'] = 'blue'

params['Russia'] = {}
params['Russia']['data'] = russia_frame['estimate']
params['Russia']['color'] = 'orange'

plot_asia_cc(params)
```



10. Определим, как изменилось значение `pctrank` для стран региона Asia Pacific за 1996-2023 годы

```
In [29]: asia_pacific_pctrank = pd.DataFrame(data=asia_pacific_countries.Country)
first_year = years[0]
last_year = years[-1]

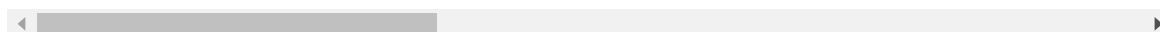
for year in years:
    year_data = []
    for country in asia_pacific_countries.Country:
        value = asia_pacific_frame\
            .query(f'countryname == "{country}" and year == {year}')\
            .filter(items=['pctrank'])
        if value.empty:
            year_data.append(math.nan)
        else:
            year_data.append(value.iloc[0,0])
    asia_pacific_pctrank[f'{year}'] = year_data

asia_pacific_pctrank = asia_pacific_pctrank.dropna(axis=1, how='all')
asia_pacific_pctrank = asia_pacific_pctrank.set_index('Country').T
asia_pacific_pctrank
```

Out[29]:

Country	Afghanistan	Australia	Bangladesh	Bhutan	Cambodia	China	
1996	4.301075	93.548386	17.741936	81.182793	16.129032	48.387096	73.655
1998	8.021390	92.513367	28.877005	81.283424	19.251337	45.989304	74.331
2000	4.787234	93.617020	6.914894	71.276596	18.085106	49.468086	72.340
2002	4.761905	92.063492	1.587302	70.899467	17.460318	35.978836	73.544
2003	4.761905	93.650795	0.529101	82.010582	14.285714	43.915344	65.079
2004	6.403941	96.551727	0.985222	80.788177	14.285714	33.497536	63.546
2005	1.463415	95.609756	2.926829	79.024391	10.243902	32.195122	49.756
2006	1.951220	95.121948	2.926829	77.560974	7.317073	37.073170	60.000
2007	0.970874	95.631065	13.592233	78.155342	10.679611	32.524273	59.708
2008	0.485437	96.116508	15.048544	78.640778	5.339806	36.407768	59.708
2009	0.956938	96.172249	14.832536	79.425835	9.569378	36.842106	53.110
2010	0.952381	95.238098	14.761905	79.523811	6.190476	33.809525	53.333
2011	0.473934	96.208534	13.744076	77.251183	5.213270	36.966824	60.189
2012	1.421801	95.734596	20.853081	79.146919	13.270143	41.232227	63.033
2013	0.947867	93.838860	21.327015	78.672989	12.796208	45.023697	63.033
2014	5.288462	95.192307	18.750000	88.461540	11.538462	45.673077	65.865
2015	5.714286	94.761902	21.904762	80.952377	11.428572	47.142857	64.761
2016	3.809524	93.333336	18.571428	81.428574	9.047619	47.142857	64.285
2017	3.809524	92.857140	18.571428	90.952377	9.047619	46.190475	72.380
2018	4.761905	92.380951	17.142857	91.428574	8.571428	45.238094	75.714
2019	5.714286	94.761902	16.190475	91.428574	9.523809	43.809525	74.761
2020	4.761905	93.809525	17.142857	93.333336	10.476191	52.857143	71.904
2021	12.380953	94.761902	18.095238	90.476189	11.428572	56.190475	65.714
2022	12.264151	95.283020	15.566038	90.094337	9.905661	55.188679	64.622
2023	13.679245	95.754715	14.622642	91.037735	9.433962	54.245281	66.037

25 rows × 31 columns



Loading [MathJax]/extensions/Safe.js

11. Сформируем данные для сравнительной таблицы и выведем ее

In [30]: `%pip install prettytable`

Requirement already satisfied: prettytable in /opt/conda/lib/python3.11/site-packages (3.16.0)
 Requirement already satisfied: wcwidth in /opt/conda/lib/python3.11/site-packages (from prettytable) (0.2.5)
 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: <https://pip.pypa.io/warnings/venv>
 Note: you may need to restart the kernel to use updated packages.

In [31]: `from prettytable import PrettyTable`

In [32]: `table = PrettyTable()
 table.field_names = ['', 'Регион', 'Страна', 'WGI 1996', 'WGI 2023', 'Изменение']
 table.align = 'c'

 mean_1996 = asia_pacific_means_1996_2023[0]
 mean_2023 = asia_pacific_means_1996_2023[-1]
 mean_delta = mean_2023 - mean_1996
 table.add_row([f'mean {last_year}', 'AP', '-', f'{mean_1996}', f'{mean_2023}', f'{mean_delta}'])

 for country in countries_max:
 region = regions_frame.set_index('Country').T[f'{country}']['Region']
 wgi_1996 = asia_pacific_pctrank[f'{country}'][f'{first_year}']
 wgi_2023 = asia_pacific_pctrank[f'{country}'][f'{last_year}']
 delta = wgi_2023 - wgi_1996

 table.add_row([f'max {last_year}', region, country, wgi_1996, wgi_2023, f'{delta}'])

 for country in countries_min:
 region = regions_frame.set_index('Country').T[f'{country}']['Region']
 wgi_1996 = asia_pacific_pctrank[f'{country}'][f'{first_year}']
 wgi_2023 = asia_pacific_pctrank[f'{country}'][f'{last_year}']
 delta = wgi_2023 - wgi_1996

 table.add_row([f'min {last_year}', region, country, wgi_1996, wgi_2023, f'{delta}'])

 russia_region = regions_frame.set_index('Country').T[f'{"Russia"}']['Region']
 russia_1996 = russia_frame.T[first_year]['pctrank']
 russia_2023 = russia_frame.T[last_year]['pctrank']
 russia_delta = russia_2023 - russia_1996

 table.add_row(['Russia_2023', russia_region, 'Russia', f'{russia_1996}', f'{russia_2023}', f'{russia_delta}'])
 table`

Out[32]:

	Регион	Страна	WGI 1996	WGI 2023	
mean 2023	AP	—	0.005390121166904767	-0.0034764338164560257	-0.008860
max 2023	AP	New Zealand	97.8494644165039	98.58490753173828	0.739
min 2023	AP	Korea, Dem. Rep.	4.838709831237793	2.358490467071533	-2.480
Russia_2023	ECA	Russia	15.053763389587402	15.566038131713867	0.512

12. Построим диаграмму размаха для всех стран и по регионам

```
In [33]: regions_frame['Region'].unique()
```

```
Out[33]: array(['AP', 'ECA', 'MENA', 'SSA', 'AME', 'WE/EU'], dtype=object)
```

```
In [34]: # Подготовим данные для графика
regions_data = {}
for region_code in regions_frame['Region'].unique():
    region_countries = regions_frame.query(f'Region == "{region_code}"').filter(
        region_frame = dataset_frame[dataset_frame.countryname.isin(region_countries)
        .query(f'indicator == "cc" and year == {last_year}')]
        .dropna()
        .filter(items=['countryname', 'estimate'])
        .set_index('countryname')
    regions_data[region_code] = region_frame['estimate'].to_numpy()
regions_data
```

```

Out[34]: {'AP': array([-1.15493178,  1.78120494, -1.12086642,  1.53142273, -1.299216
75,
        -0.0053701 ,  0.50627202,  1.62514544, -0.36601475, -0.4868722 ,
        1.39566064, -1.58447635,  0.89408857, -0.97383636,  0.29562137,
        -0.39694333, -0.4926044 , -1.21649933, -0.50792795,  2.08409548,
        -0.99935746, -0.65715402, -0.53763998,  2.04018402, -0.13471074,
        -0.383977 ,  1.20336866, -0.48905078, -0.22670417, -0.01486515,
        -0.41581428]),
'ECA': array([-0.33221853,  0.05839965, -1.19547141, -0.66502804, -0.58156
002,
        0.6176948 , -0.26659307, -0.18528606, -0.28033847, -0.07930987,
        -0.34861547, -0.44720426, -1.3787334 , -1.42362428, -0.68614632,
        -0.80752933]),
'MENA': array([-0.58930832,  0.17778482, -1.32103753,  0.82935941,  0.0886
5869,
        0.20661183, -1.22706246, -1.52501881, -0.53634858,  0.21515796,
        0.70220983,  0.53604299, -0.34189978,  1.07366383]),
'SSA': array([-0.6099202 , -0.04957945,  0.68579209, -0.16961551, -1.55602
598,
        -1.16255975, -1.31139302, -1.48240697, -0.99010926, -0.78092742,
        -1.5703727 , -1.47526491, -0.73498547, -0.46506163, -1.01552844,
        -0.09902111, -0.89767343, -0.770913 , -0.50947851, -0.88834929,
        -1.00018215, -0.60281152, -0.86183155, -0.82265264,  0.45319736,
        -0.82843775,  0.11007203, -0.56384397, -1.04097438,  0.67165309,
        0.05783223,  1.62884772, -0.57455784, -1.73283362, -0.28467044,
        -1.96955538, -1.50206709, -0.31989732, -0.61917496, -1.04257476,
        -0.47770688, -1.26121604]),
'AME': array([-0.36088395,  1.3445859 , -0.83830106, -0.50364316,  1.67150
104,
        0.96860379, -0.30919129,  0.65374941, -0.05509086,  0.54451454,
        -0.44947493, -0.64733744, -0.56131506,  0.55816859, -1.12161291,
        -0.38374501, -1.44387412, -1.11195815, -0.11208299, -1.02057874,
        -1.38844013, -0.6280942 , -1.05772293, -0.7210409 , -0.40435159,
        -0.37414855,  1.57394731]),
'WE/EU': array([ 1.13365328e+00,  1.33684444e+00, -1.37721658e-01,  1.8192
8679e-01,
        3.32707405e-01,  2.37605309e+00,  1.53812230e+00,  2.22151661e+00,
        1.18482184e+00,  1.66416621e+00,  9.78469253e-02, -4.98130627e-04,
        1.55130613e+00,  1.57528889e+00,  5.50129175e-01,  6.98113978e-01,
        7.81545520e-01,  1.92665911e+00,  1.09746583e-01,  1.86710048e+00,
        2.11352158e+00,  5.65399945e-01,  6.96312070e-01,  4.04953435e-02,
        7.76647091e-01,  6.29368722e-01,  2.02824616e+00,  2.02199912e+00,
        1.48091388e+00])})

```

In [35]: *# Рисуем диаграмму размаха*

```
world = np.concatenate(list(regions_data.values()))
```

```
data = [world]
```

```
labels = ['WORLD']
```

```
for key in regions_data:
```

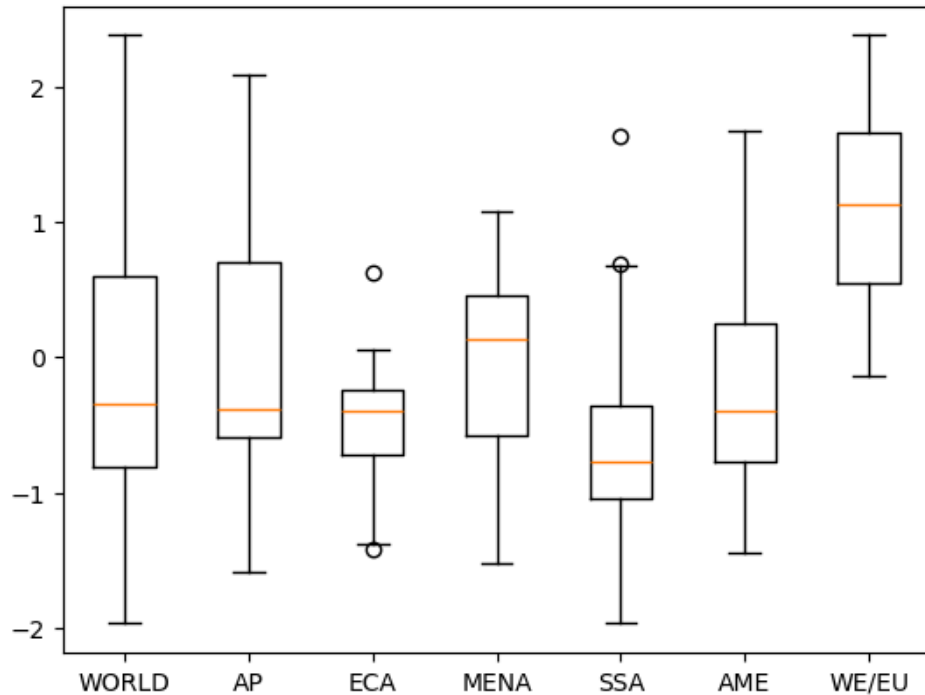
```
    data.append(regions_data[f'{key}'])
```

```
    labels.append(f'{key}')
```

```
(data, labels=labels)
```

```
plt.title('Chart of the Control of Corruption Index by World Region 2023 (hi  
plt.show()
```

Chart of the Control of Corruption Index by World Region 2023 (higher - better)



```
In [36]: # Данные со всего мира  
all_values = np.concatenate(list(regions_data.values()))  
all_values.sort()  
all_values
```

```
Out[36]: array([-1.96955538e+00, -1.73283362e+00, -1.58447635e+00, -1.57037270e+00,
-1.55602598e+00, -1.52501881e+00, -1.50206709e+00, -1.48240697e+00,
-1.47526491e+00, -1.44387412e+00, -1.42362428e+00, -1.38844013e+00,
-1.37873340e+00, -1.32103753e+00, -1.31139302e+00, -1.29921675e+00,
-1.26121604e+00, -1.22706246e+00, -1.21649933e+00, -1.19547141e+00,
-1.16255975e+00, -1.15493178e+00, -1.12161291e+00, -1.12086642e+00,
-1.11195815e+00, -1.05772293e+00, -1.04257476e+00, -1.04097438e+00,
-1.02057874e+00, -1.01552844e+00, -1.00018215e+00, -9.99357462e-01,
-9.90109265e-01, -9.73836362e-01, -8.97673428e-01, -8.88349295e-01,
-8.61831546e-01, -8.38301063e-01, -8.28437746e-01, -8.22652638e-01,
-8.07529330e-01, -7.80927420e-01, -7.70913005e-01, -7.34985471e-01,
-7.21040905e-01, -6.86146319e-01, -6.65028036e-01, -6.57154024e-01,
-6.47337437e-01, -6.28094196e-01, -6.19174957e-01, -6.09920204e-01,
-6.02811515e-01, -5.89308321e-01, -5.81560016e-01, -5.74557841e-01,
-5.63843966e-01, -5.61315060e-01, -5.37639976e-01, -5.36348581e-01,
-5.09478509e-01, -5.07927954e-01, -5.03643155e-01, -4.92604405e-01,
-4.89050776e-01, -4.86872196e-01, -4.77706879e-01, -4.65061635e-01,
-4.49474931e-01, -4.47204262e-01, -4.15814281e-01, -4.04351592e-01,
-3.96943331e-01, -3.83976996e-01, -3.83745015e-01, -3.74148548e-01,
-3.66014749e-01, -3.60883951e-01, -3.48615468e-01, -3.41899782e-01,
-3.32218528e-01, -3.19897324e-01, -3.09191287e-01, -2.84670442e-01,
-2.80338466e-01, -2.66593069e-01, -2.26704165e-01, -1.85286060e-01,
-1.69615507e-01, -1.37721658e-01, -1.34710744e-01, -1.12082988e-01,
-9.90211144e-02, -7.93098733e-02, -5.50908595e-02, -4.95794490e-02,
-1.48651544e-02, -5.37010469e-03, -4.98130627e-04, 4.04953435e-02,
5.78322262e-02, 5.83996475e-02, 8.86586905e-02, 9.78469253e-02,
1.09746583e-01, 1.10072032e-01, 1.77784815e-01, 1.81928679e-01,
2.06611827e-01, 2.15157956e-01, 2.95621365e-01, 3.32707405e-01,
4.53197360e-01, 5.06272018e-01, 5.36042988e-01, 5.44514537e-01,
5.50129175e-01, 5.58168590e-01, 5.65399945e-01, 6.17694795e-01,
6.29368722e-01, 6.53749406e-01, 6.71653092e-01, 6.85792089e-01,
6.96312070e-01, 6.98113978e-01, 7.02209830e-01, 7.76647091e-01,
7.81545520e-01, 8.29359412e-01, 8.94088566e-01, 9.68603790e-01,
1.07366383e+00, 1.13365328e+00, 1.18482184e+00, 1.20336866e+00,
1.33684444e+00, 1.34458590e+00, 1.39566064e+00, 1.48091388e+00,
1.53142273e+00, 1.53812230e+00, 1.55130613e+00, 1.57394731e+00,
1.57528889e+00, 1.62514544e+00, 1.62884772e+00, 1.66416621e+00,
1.67150104e+00, 1.78120494e+00, 1.86710048e+00, 1.92665911e+00,
2.02199912e+00, 2.02824616e+00, 2.04018402e+00, 2.08409548e+00,
2.11352158e+00, 2.22151661e+00, 2.37605309e+00])
```

Задание 2. Анализ рынка акций (4 балла)

Условие

1. Загрузить [набор данных](#), каждый файл содержит данные по одной акции (компании). Все данные загрузить в один датафрейм, в качестве индекса для него взять поле "Date", наименование столбцов - название акции (название файла без .csv), содержание столбца - значение столбца "Close" для каждой акции;

Loading [MathJax]/extensions/Safe.js ать корреляционную матрицу для всех акций

3. Отобразить корреляционную матрицу в виде диаграммы. Примерный вид графика ниже:  корреляция
4. Согласно варианту определить:
 - акцию с максимальной положительной корреляцией (max);
 - акцию с максимальной отрицательной корреляцией (min);
 - акцию с минимальной корреляцией (которая больше всего соответствует отсутствию какой-либо корреляции (none);
5. Построить диаграммы разброса (Ваша компания - Компания с min), (Ваша компания - Компания с max), (Ваша компания - Компания с none);
6. Рассчитать среднюю цену акций для каждого месяца (исходные данные взяты с интервалом в месяц);
7. Построить графики для акций из пункта 4 и средней из пункта 6. Примерный вид графика приведен ниже:  график стоимости акций

1. Загружаем набор данных в один датафрейм

Подгружаем данные, если их нет локально

```
In [37]: %%bash

# Messages from utils in stderr is not an error
exec 2>&1

dataset_path="../data/A1_StockMarket_dataset/stock"

mkdir -p ${dataset_path}
count_files=$(find ${dataset_path} -maxdepth 1 -type f | wc -l)

echo "Let's check the availability of the dataset locally"

if (( count_files == 0 )); then
  # TODO: need unzip in docker image?
  apt update && apt install unzip
  curl -L "https://github.com/MLMethods/Assignments/archive/refs/heads/master.zip" -o archive.zip
  mkdir -p tmp
  unzip archive.zip -d tmp
  cp -r tmp/Assignments-master/data/A1_Descriptive_Analysis/stock/* "${dataset_path}"
  rm -rf tmp
  rm -rf archive.zip
else
  echo "Files are available locally"
fi
```

Let's check the availability of the dataset locally
Files are available locally

```
In [38]: dataset_path = "../data/A1_StockMarket_dataset/stock"
files = files = os.listdir(dataset_path)
```

Loading [MathJax]/extensions/Safe.js

```
data = {}
columns = []
is_first_file = True

# print(files)

columns = ['Date'] + [os.path.splitext(file)[0] for file in files]

for file in files:
    basename = os.path.splitext(f'{file}')[0]
    tmp = pd.read_csv(dataset_path + '/' + f'{file}')

    if is_first_file:
        data['Date'] = tmp['Date']
        is_first_file = False

    data[f'{basename}'] = tmp['Close']

stock_dataset = pd.DataFrame(data).set_index('Date')
stock_dataset
```

Out[38]:

	GOOGL	ABNB	META	INTC	ADBE	EBAY	NVD
Date							
2022-01-01	135.303497	153.970001	313.260010	48.820000	534.299988	60.070000	244.860000
2022-02-01	135.057007	151.490005	211.029999	47.700001	467.679993	54.590000	243.850000
2022-03-01	139.067505	171.759995	222.360001	49.560001	455.619995	57.259998	272.859998
2022-04-01	114.109497	153.210007	200.470001	43.590000	395.950012	51.919998	185.470000
2022-05-01	113.762001	120.870003	193.639999	44.419998	416.480011	48.669998	186.720000
2022-06-01	108.962997	89.080002	161.250000	37.410000	366.059998	41.669998	151.589998
2022-07-01	116.320000	110.980003	159.100006	36.310001	410.119995	48.630001	181.630000
2022-08-01	108.220001	113.120003	162.929993	31.920000	373.440002	44.130001	150.940000
2022-09-01	95.650002	105.040001	135.679993	25.770000	275.200012	36.810001	121.389998
2022-10-01	94.510002	106.910004	93.160004	28.430000	318.500000	39.840000	134.970000
2022-11-01	100.989998	102.139999	118.099998	30.070000	344.929993	45.439999	169.229998
2022-12-01	88.230003	85.500000	120.339996	26.430000	336.529999	41.470001	146.139998
2023-01-01	98.839996	111.110001	148.970001	28.260000	370.339996	49.500000	195.369998
2023-02-01	90.059998	123.279999	174.940002	24.930000	323.950012	45.900002	232.160000
2023-03-01	103.730003	124.400002	211.940002	32.669998	385.369995	44.369999	277.769998
2023-04-01	107.339996	119.669998	240.320007	31.059999	377.559998	46.430000	277.489998
2023-05-01	122.870003	109.769997	264.720001	31.440001	417.790009	42.540001	378.339998
2023-06-01	119.699997	128.160004	286.980011	33.439999	488.989990	44.689999	423.019998

Loading [MathJax]/extensions/Safe.js

	GOOGL	ABNB	META	INTC	ADBE	EBAY	NVD
Date							
2023-07-01	132.720001	152.190002	318.600006	35.770000	546.169983	44.509998	467.29000
2023-08-01	136.169998	131.550003	295.890015	35.139999	559.340027	44.779999	493.54998
2023-09-01	130.860001	137.210007	300.209991	35.549999	509.899994	44.090000	434.98999
2023-10-01	124.080002	118.290001	301.269989	36.500000	532.059998	39.230000	407.79998
2023-11-01	132.529999	126.339996	327.149994	44.700001	611.010010	41.009998	467.70001
2023-12-01	139.690002	136.139999	353.959991	50.250000	596.599976	43.619999	495.22000
2024-01-01	140.100006	144.139999	390.140015	43.080002	617.780029	41.070000	615.27002
2024-02-01	138.460007	157.470001	490.130005	43.049999	560.280029	47.279999	791.11999
2024-03-01	138.500000	166.669998	499.750000	45.240002	579.140015	50.910000	919.13000
2024-03-12	138.500000	166.669998	499.750000	45.240002	579.140015	50.910000	919.13000

28 rows × 25 columns

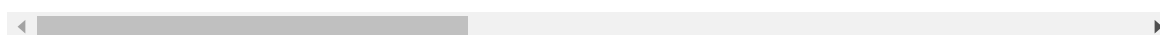
2. Корреляционная матрица для датасета

```
In [39]: stock_corr = stock_dataset.corr()
stock_corr
```

Out[39]:

	GOOGL	ABNB	META	INTC	ADBE	EBAY	NVDA	
GOOGL	1.000000	0.780440	0.808784	0.826042	0.915440	0.375794	0.715287	0.6
ABNB	0.780440	1.000000	0.723419	0.738241	0.670509	0.644140	0.649664	0.5
META	0.808784	0.723419	1.000000	0.594611	0.873388	0.190361	0.961389	0.8
INTC	0.826042	0.738241	0.594611	1.000000	0.713875	0.580047	0.458281	0.4
ADBE	0.915440	0.670509	0.873388	0.713875	1.000000	0.180354	0.802739	0.8
EBAY	0.375794	0.644140	0.190361	0.580047	0.180354	1.000000	0.087027	-0.0
NVDA	0.715287	0.649664	0.961389	0.458281	0.802739	0.087027	1.000000	0.8
PINS	0.640675	0.554616	0.822643	0.452144	0.804657	-0.002757	0.815629	1.0
MU	0.867191	0.842928	0.858401	0.839860	0.817961	0.512637	0.796707	0.7
AAPL	0.806847	0.617430	0.705358	0.507251	0.833129	0.115591	0.633114	0.6
TSLA	0.326662	0.353807	-0.144519	0.425236	0.071508	0.434899	-0.277600	-0.2
TCOM	0.322718	0.294269	0.707029	-0.014994	0.533298	-0.149330	0.787859	0.7
HPQ	0.263251	0.390153	-0.035611	0.591406	0.081518	0.744560	-0.160502	-0.2
CSCO	0.600025	0.594365	0.374998	0.420854	0.554172	0.494938	0.320159	0.3
TWLO	0.315410	0.429915	-0.072886	0.585988	0.067604	0.753732	-0.244797	-0.1
GTLB	0.535473	0.460602	0.467641	0.535441	0.496556	0.251066	0.404702	0.5
XIACY	0.680658	0.564475	0.573429	0.791377	0.697612	0.535223	0.445645	0.5
SHOP	-0.725121	-0.378079	-0.687123	-0.691324	-0.750258	-0.587906	-0.534172	0.1
ORCL	0.618983	0.471504	0.821696	0.239485	0.785432	-0.070414	0.875089	0.7
UBER	0.737311	0.680764	0.954444	0.512572	0.834611	0.085736	0.969790	0.9
SPOT	0.821587	0.753797	0.973401	0.645555	0.863827	0.296858	0.925270	0.8
NFLX	0.717756	0.646901	0.897908	0.447049	0.821314	0.138580	0.910910	0.9
MSFT	0.845993	0.679204	0.966868	0.627531	0.913842	0.127010	0.935386	0.8
DBX	0.669228	0.332740	0.552874	0.390625	0.816359	-0.157363	0.519374	0.7
AMZN	0.912332	0.830690	0.830910	0.816519	0.819614	0.434078	0.765294	0.6

25 rows × 25 columns



3. Корреляционная матрица в виде диаграммы

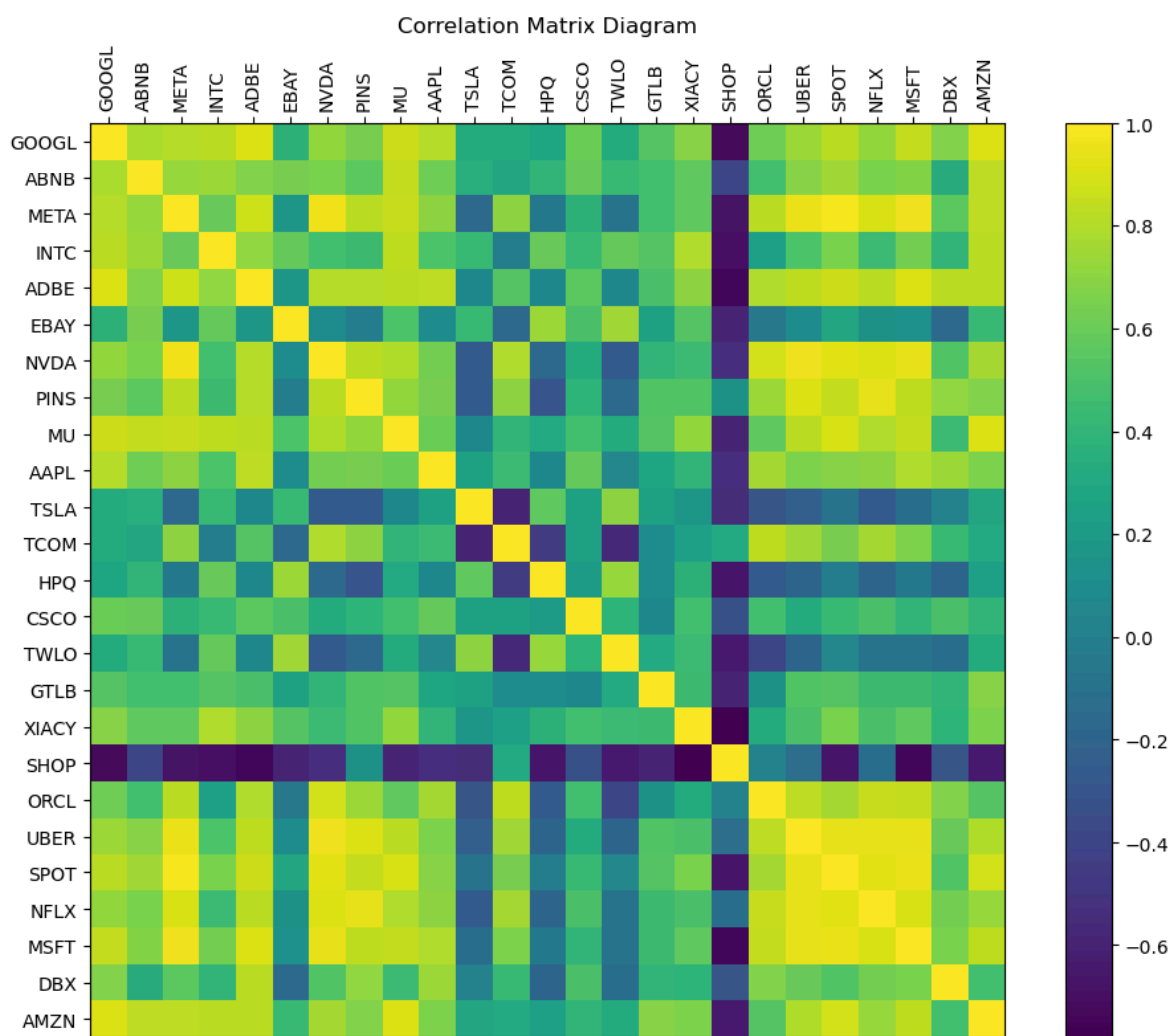
Loading [MathJax]/extensions/Safe.js

```
In [40]: labels = stock_corr.columns.to_list()

fig = plt.figure(figsize=(12,9))
ax = fig.add_subplot(1,1,1)
cax = ax.matshow(stock_corr)
fig.colorbar(cax)

plt.xticks(range(len(stock_corr.columns)), stock_corr.columns, rotation=90)
plt.yticks(range(len(stock_corr.columns)), stock_corr.columns)
plt.title("Correlation Matrix Diagram")

plt.show()
```



4. Для варианта 3 надо найти максимально, минимально коррелирующие акции с акцией компании Oracle (ORCL), а также наиболее некоррелирующую акцию (none)

```
In [41]: COMPANY = 'ORCL'
```

Loading [MathJax]/extensions/Safe.js

```
ORCL_CORR = {}
```

```

orcl_corr['min'] = stock_corr.T[COMPANY].sort_values().index[0]
orcl_corr['max'] = stock_corr.T[COMPANY].sort_values(ascending=0).index[1]
orcl_corr['none'] = abs(stock_corr.T[COMPANY]).sort_values().index[0]

print(f"Min correlation: {orcl_corr['min']} ({stock_corr[COMPANY][orcl_corr['min']]}")
print(f"Max correlation: {orcl_corr['max']} ({stock_corr[COMPANY][orcl_corr['max']]}")
print(f"None correlation: {orcl_corr['none']} ({stock_corr[COMPANY][orcl_corr['none']]}")

```

Min correlation: TWLO (-0.393535537911863)

Max correlation: NVDA (0.875088713879195)

None correlation: SHOP (0.023692039836557897)

5. Строим диаграммы разброса акций компании из варианта (ORCL) с акциями, наденными выше

```

In [42]: # Рисуем диаграммы разброса

keys = ['max', 'min', 'none']
x = [stock_dataset[COMPANY] for _ in range(len(keys))]
y = [stock_dataset[orcl_corr[j]] for j in keys]
plot_settings = [
    {"color": "blue", "marker": "o", "title": "Max correlation"},
    {"color": "green", "marker": "o", "title": "Min correlation"},
    {"color": "red", "marker": "o", "title": "None correlation"}
]

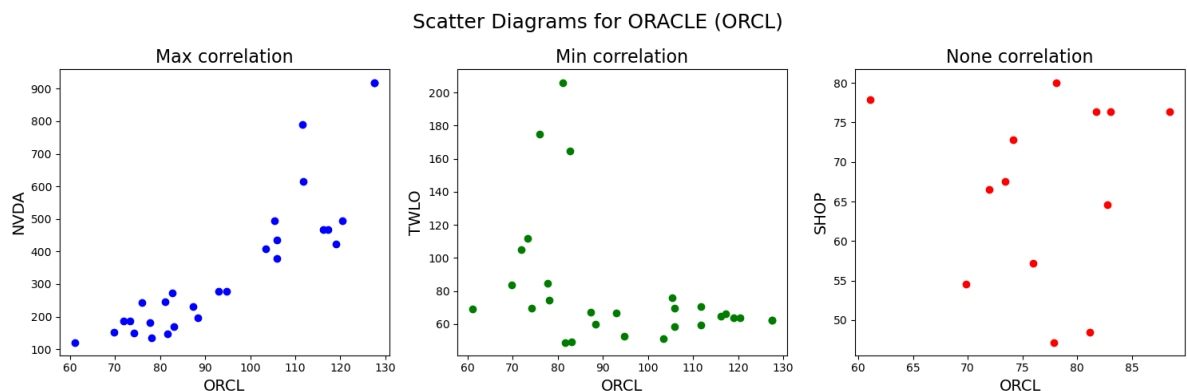
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15,5))

for i, ax in enumerate(axes):
    ax.scatter(x[i], y[i],
               color=plot_settings[i]['color'],
               marker=plot_settings[i]['marker'])

    ax.set_title(plot_settings[i]['title'], fontsize=16)
    ax.set_xlabel(COMPANY, fontsize=14)
    ax.set_ylabel(orcl_corr[keys[i]], fontsize=14)

plt.suptitle('Scatter Diagrams for ORACLE (ORCL)', fontsize=18)
plt.tight_layout()
plt.show()

```



Loading [MathJax]/extensions/Safe.js

6. Срдение по месяцам

```
In [43]: stock_means = stock_dataset.mean(axis=1)
stock_means
```

```
Out[43]: Date
2022-01-01    150.600880
2022-02-01    137.431334
2022-03-01    142.045395
2022-04-01    113.836173
2022-05-01    110.482993
2022-06-01     97.469452
2022-07-01    111.341999
2022-08-01    105.998399
2022-09-01     93.775600
2022-10-01     96.541201
2022-11-01     99.699599
2022-12-01     91.402200
2023-01-01    107.002758
2023-02-01    108.613126
2023-03-01    120.210832
2023-04-01    118.584166
2023-05-01    134.344584
2023-06-01    148.794582
2023-07-01    156.775000
2023-08-01    155.579584
2023-09-01    145.393333
2023-10-01    144.340623
2023-11-01    162.973751
2023-12-01    168.482916
2024-01-01    178.837501
2024-02-01    194.328293
2024-03-01    201.071667
2024-03-12    201.071667
dtype: float64
```

7. Строим график по имеющимся данным

```
In [44]: import matplotlib.dates as mdates
```

```
In [45]: plt.figure('ORACLE comparision', figsize=[10,6])

companies = [f'{COMPANY}'] + [orcl_corr[key] for key in keys]
colors = { f'{COMPANY}' : 'blue', 'max' : 'orange', 'min': 'green', 'none':

plot_settings = []

# Настройки для ORACLE
plot_settings.append({"color": f"{colors[COMPANY]}", "marker": "o", "label":

for key in keys:
    plot_settings.append({"color": f"{colors[key]}", "marker": "o", "label":
```

Loading [MathJax]/extensions/Safe.js

```

dates = pd.to_datetime( stock_dataset[COMPANY].index.to_list() )

for i in range(len(companies)):
    company = companies[i]
    plt.plot(dates, stock_dataset[company],
             marker='o',
             markersize=5,
             color=plot_settings[i]['color'],
             label=plot_settings[i]['label'])

# Отдельно добавим средние на график
plt.plot(dates, stock_means, marker='o', markersize=5, label='Average', color='gray')

plt.legend(loc=0, fontsize=12)

plt.grid(True)
plt.title('Oracle comparision')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()

```

