— —

September 19, 2024

[1]:
```
import wbdata
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import sympy as sp
import scipy.stats as st
import os
from docx import Document
from docx.shared import Inches, Pt
from docx.table import Table
from openpyxl import load_workbook
from deep_translator import GoogleTranslator
from docx.enum.text import WD_ALIGN_PARAGRAPH
##########################
import locale
locale.setlocale(locale.LC_NUMERIC, 'russian')
plt.rcParams['axes.formatter.use_locale'] = True
#########
#########
plt.rcParams["font.family"] = "Times New Roman"
plt.rcParams['mathtext.fontset'] = 'cm'
###################
sp.init_printing(use_unicode=True,use_latex=True)
###################
def add_excel_table_to_docx(xlsx_file, docx_file,␣
 ↪sheet_name='Sheet1',from_row=1):
    """                        Excel        Word.

    Args:
        xlsx_file (str):          Excel.
        docx_file (str):            Word.
        sheet_name (str, optional):            Excel.          '  1'.
    """

    #             Excel
    workbook = load_workbook(xlsx_file)
```

```
    worksheet = workbook[sheet_name]

    #                 Word
    document = Document(docx_file)

    #                   Word
    table = document.add_table(rows=worksheet.max_row, cols=worksheet.
 ↪max_column)

    #                       Excel
    for row_idx in range(from_row, worksheet.max_row + 1):
        for col_idx in range(1, worksheet.max_column + 1):
            cell = table.cell(row_idx - 1, col_idx - 1)
            cell.text = str(worksheet.cell(row=row_idx, column=col_idx).value)

    #             Word
    document.save(docx_file)
####################
def add_text(docx_file,text:str,font_name='Times New␣
 ↪Roman',font_size=12,par_allign='CENTER'):
    alligments={
        'CENTER': WD_ALIGN_PARAGRAPH.CENTER,
        'LEFT':WD_ALIGN_PARAGRAPH.LEFT,
        'RIGHT':WD_ALIGN_PARAGRAPH.RIGHT,
    }
    document = Document(docx_file)
    paragraph = document.add_paragraph()
    paragraph.alignment = alligments[par_allign]  #               (        )
    run = paragraph.add_run(text)
    run.font.size=Pt(font_size)
    run.font.name=font_name
    document.save(docx_file)
####################
def add_image_to_docx(docx_file, image_path, indent_size=-2, width=10.0):
    indent_size=Inches(indent_size)
    width=Inches(width)
    """             PNG        DOCX                .

    Args:
        docx_file (str):                      DOCX    .
        image_path (str):             PNG.
        indent_size (Inches, optional):                  .         0.5    .
        width (Inches, optional):                .         3    .
    """

    document = Document(docx_file)
    paragraph = document.add_paragraph()
```

```
    paragraph.alignment = WD_ALIGN_PARAGRAPH.CENTER  #                (        )
    run = paragraph.add_run()
    run.add_picture(image_path, width=width)
    paragraph.paragraph_format.first_line_indent = indent_size
    document.save(docx_file)
###################
#            :
country = 'RUS'
indicators = {
    'NY.GDP.MKTP.CD': 'GDP (current USD)',
    'NY.GDP.MKTP.KD.ZG': 'GDP growth (annual %)',
    'SL.UEM.TOTL.ZS': 'Unemployment rate (%)',
}
GDP_norm = 0.025
document = Document()
translator = GoogleTranslator(source='auto', target='ru')
```

[2]:
```
#country_dict=dict([[i.split('   ')[0],translator.translate(i.split('   ')[1])]␣
 ↪for i in str(wbdata.get_countries()).split('\n')[2:]])
country_dict={
    'ABW': '   ',
 'AFE': '               ',
 'AFG': '      ',
 'AFR': '    ',
 'AFW': '                 ',
 'AGO': '    ',
 'ALB': '    ',
 'AND': '    ',
 'ARB': '       ',
 'ARE': '                  ',
 'ARG': '     ',
 'ARM': '    ',
 'ASM': '           ',
 'ATG': '           ',
 'AUS': '     ',
 'AUT': '    ',
 'AZE': '      ',
 'BDI': '    ',
 'BEA': '                       (          )',
 'BEC': '                 (          )',
 'BEL': '    ',
 'BEN': '   ',
 'BFA': '    -  ',
 'BGD': '     ',
 'BGR': '     ',
 'BHI': '                                    ',
 'BHR': '    ',
```

```
'BHS': '          ,',
'BIH': '            ',
'BLA': '                          (          )',
'BLR': '    ',
'BLZ': '   ',
'BMN': '                   (          )',
'BMU': '          ',
'BOL': '    ',
'BRA': '    ',
'BRB': '    ',
'BRN': '   -     ',
'BSS': '               (          )',
'BTN': '   ',
'BWA': '    ',
'CAA': '               (        IFC)',
'CAF': '                  ',
'CAN': '   ',
'CEA': '                      (        IFC)',
'CEB': '                  ',
'CEU': '                (        IFC)',
'CHE': '     ',
'CHI': '           ',
'CHL': '  ',
'CHN': '   ',
'CIV': "  - '   ",
'CLA': '                      (        IFC)',
'CME': '                   (        IFC)',
'CMR': '    ',
'COD': '   ,               .',
'COG': '         ',
'COL': '    ',
'COM': '          ',
'CPV': '  -   ',
'CRI': '   -  ',
'CSA': '       (        IFC)',
'CSS': '                    ',
'CUB': '  ',
'CUW': '    ',
'CYM': '          ',
'CYP': '  ',
'CZE': '   ',
'DEA': '                      (   ,                    )',
'DEC': '                (   ,                  )',
'DEU': '    ',
'DJI': '    ',
'DLA': '                          (   ,                    )',
'DMA': '    ',
```

```
'DMN': '                    (   ,                    )',
'DNK': '   ',
'DNS': '                                          ',
'DOM': '             ',
'DSA': '       (   ,                      )',
'DSF': '                                       ',
'DSS': '                   (   ,                    )',
'DZA': '   ',
'EAP': '                      (                       )',
'EAR': '                  ',
'EAS': '                     ',
'ECA': '                (                        )',
'ECS': '               ',
'ECU': '    ',
'EGY': '   ,           ',
'EMU': '    ',
'ERI': '    ',
'ESP': '    ',
'EST': '    ',
'ETH': '    ',
'EUU': '    ',
'FCS': '                    ',
'FIN': '     ',
'FJI': '   ',
'FRA': '    ',
'FRO': '          ',
'FSM': '     ,     .  .',
'FXS': '      ,                    ,                          ',
'GAB': '   ',
'GBR': '        ',
'GEO': '    ',
'GHA': '   ',
'GIB': '     ',
'GIN': '    ',
'GMB': '   ,',
'GNB': '   -   ',
'GNQ': '            ',
'GRC': '    ',
'GRD': '    ',
'GRL': '     ',
'GTM': '     ',
'GUM': '   ',
'GUY': '    ',
'HIC': '        ',
'HKG': '    ,   ,    ',
'HND': '    ',
'HPC': '                      (HIPC)',
```

```
'HRV': '     ',
'HTI': '   ',
'HUN': '    ',
'IBB': '   ,           ',
'IBD': '       ',
'IBT': '          ',
'IDA': '    ADI',
'IDB': '      ',
'IDN': '     ',
'IDX': '      ',
'IMN': '      ',
'IND': '   ',
'INX': '           ',
'IRL': '     ',
'IRN': '   ,             ',
'IRQ': '   ',
'ISL': '     ',
'ISR': '    ',
'ITA': '    ',
'JAM': '    ',
'JOR': '     ',
'JPN': '    ',
'KAZ': '     ',
'KEN': '   ',
'KGZ': '            ',
'KHM': '     ',
'KIR': '     ',
'KNA': '   -        ',
'KOR': '          .',
'KWT': '    ',
'LAC': '                        (                          )',
'LAO': '       ',
'LBN': '   ',
'LBR': '    ',
'LBY': '   ',
'LCA': '   -   ',
'LCN': '                      ',
'LDC': '              :           ',
'LIC': '       ',
'LIE': '      ',
'LKA': '  -   ',
'LMC': '                 ',
'LMY': '              ',
'LSO': '    ',
'LTE': '                   ',
'LTU': '   ',
'LUX': '      ',
```

```
'LVA': '    ',
'MAC': '   ,   ,    ',
'MAF': '  -    (          )',
'MAR': '    ',
'MCO': '    ',
'MDA': '    ',
'MDE': '         (           )',
'MDG': '      ',
'MDV': '     ',
'MEA': '                    ',
'MEX': '    ',
'MHL': '           ',
'MIC': '        ',
'MKD': '           ',
'MLI': '   ',
'MLT': '    ',
'MMR': '    ',
'MNA': '                    (                          )',
'MNE': '      ',
'MNG': '     ',
'MNP': '                ',
'MOZ': '     ',
'MRT': '      ',
'MUS': '     ',
'MWI': '    ',
'MYS': '     ',
'NAC': '          ',
'NAF': '         ',
'NAM': '    ',
'NCL': '         ',
'NER': '   ',
'NGA': '    ',
'NIC': '     ',
'NLD': '      ',
'NOR': '     ',
'NPL': '   ',
'NRS': '                  ,           ',
'NRU': '   ',
'NXS': '      ,                                  ,                              ',
'NZL': '        ',
'OED': '      ',
'OMN': '   ',
'OSS': '              ',
'PAK': '     ',
'PAN': '    ',
'PER': '   ',
'PHL': '     ',
```

```
'PLW': '   ',
'PNG': '   -       ',
'POL': '    ',
'PRE': '                ',
'PRI': '    -   ',
'PRK': '   ,     -              ',
'PRT': '      ',
'PRY': '     ',
'PSE': '                  ',
'PSS': '                       ',
'PST': '                ',
'PYF': '             ',
'QAT': '   ',
'ROU': '    ',
'RRS': '                            ',
'RUS': '            ',
'RWA': '    ',
'SAS': '      ',
'SAU': '          ',
'SDN': '   ',
'SEN': '    ',
'SGP': '     ',
'SLB': '           ',
'SLE': '    -   ',
'SLV': '      ',
'SMR': '  -    ',
'SOM': '    ',
'SRB': '    ',
'SSA': '                (                          )',
'SSD': '       ',
'SSF': '               ',
'SST': '          ',
'STP': '  -          ',
'SUR': '    ',
'SVK': '            ',
'SVN': '     ',
'SWE': '    ',
'SWZ': '     ',
'SXM': '   -    (          )',
'SXZ': '              ,                ',
'SYC': '           ',
'SYR': '                 ',
'TCA': '              ',
'TCD': '  ',
'TEA': '                       (           )',
'TEC': '                (           )',
'TGO': '   ',
```

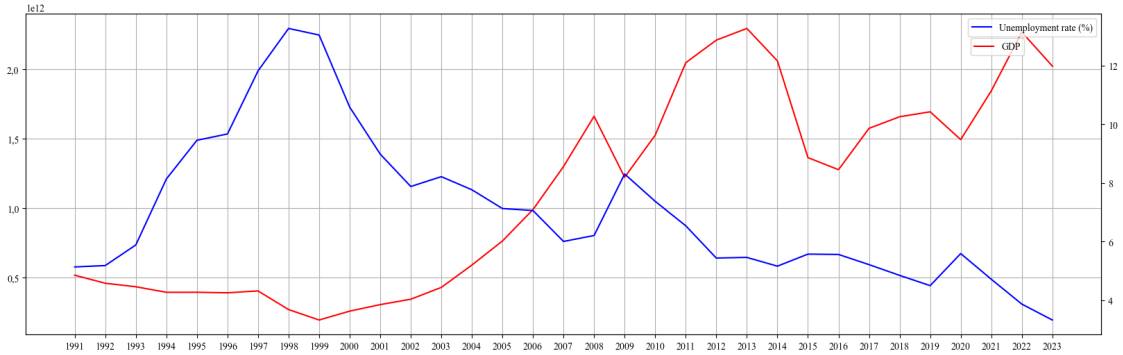
```
 'THA': '    ',
 'TJK': '      ',
 'TKM': '       ',
 'TLA': '                              (           )',
 'TLS': '   -   ',
 'TMN': '                      (           )',
 'TON': '   ',
 'TSA': '       (       )',
 'TSS': '                (           )',
 'TTO': '           ',
 'TUN': '   ',
 'TUR': '    ',
 'TUV': '    ',
 'TZA': '     ',
 'UGA': '    ',
 'UKR': '    ',
 'UMC': '             ',
 'URY': '    ',
 'USA': '          ',
 'UZB': '      ',
 'VCT': '  -            ',
 'VEN': '     ,  ',
 'VGB': '                  ',
 'VIR': '            ( )',
 'VNM': '    ',
 'VUT': '    ',
 'WLD': '  ',
 'WSM': '   ',
 'XKX': '    ',
 'XZN': '               ,                        ',
 'YEM': '   ,       ',
 'ZAF': '  ',
 'ZMB': '    ',
 'ZWE': '     '}
```

[3]:
```
data = wbdata.get_dataframe(indicators, country=country)
data = data.dropna().sort_values('date')
data['GDP (current USD)'] = data['GDP (current USD)']
data.rename(columns={'GDP (current USD)': 'GDP'}, inplace=True)
try:
    os.mkdir(f'{country}')
    os.chdir(f'{country}')
except:
    os.chdir(f'{country}')
try:
    data.to_excel(f'{country}_1.xlsx')
except:
```

```
    pass
display(data)
document.save(f'{country}.docx')
add_text(f'{country}.docx',f'                                    ␣
 ↪{country_dict[country]}',font_size=15)
add_text(f'{country}.docx',f'\t                                                (GDP),      ␣
 ↪                (GDP growth (annual %))              (Unemployment rate (%))␣
 ↪                      {country_dict[country]}.',par_allign='LEFT')
add_excel_table_to_docx(f'{country}_1.xlsx',f'{country}.docx')
```

	GDP	GDP growth (annual %)	Unemployment rate (%)
date			
1991	5.179630e+11	-5.046939	5.133
1992	4.602906e+11	-14.531074	5.181
1993	4.350837e+11	-8.668540	5.883
1994	3.950773e+11	-12.569756	8.131
1995	3.955372e+11	-4.143528	9.449
1996	3.917249e+11	-3.755069	9.665
1997	4.049290e+11	1.399916	11.813
1998	2.709555e+11	-5.299962	13.261
1999	1.959071e+11	6.399915	13.036
2000	2.597101e+11	10.000067	10.581
2001	3.066021e+11	5.100051	8.978
2002	3.454705e+11	4.699992	7.875
2003	4.303474e+11	7.299952	8.210
2004	5.910167e+11	7.199948	7.763
2005	7.640160e+11	6.399965	7.124
2006	9.899321e+11	8.200068	7.055
2007	1.299703e+12	8.499978	6.002
2008	1.660848e+12	5.199969	6.205
2009	1.222646e+12	-7.799994	8.301
2010	1.524917e+12	4.500000	7.369
2011	2.045923e+12	4.300029	6.536
2012	2.208294e+12	4.024086	5.436
2013	2.292470e+12	1.755422	5.458
2014	2.059242e+12	0.736267	5.160
2015	1.363482e+12	-1.972719	5.571
2016	1.276786e+12	0.193690	5.559
2017	1.574199e+12	1.825790	5.212
2018	1.657329e+12	2.807245	4.846
2019	1.693115e+12	2.198076	4.496
2020	1.493076e+12	-2.653655	5.589
2021	1.843392e+12	5.614290	4.715
2022	2.266029e+12	-2.069712	3.867
2023	2.021421e+12	3.600000	3.325

[4]:
```
fig, ax = plt.subplots(figsize=(20, 6))

gdp, = ax.plot(data['GDP'], 'r', label='GDP')

ax1 = ax.twinx()
unemplt, = ax1.plot(data['Unemployment rate (%)'], 'b', label='Unemployment␣
 ↪rate (%)')

ax.legend(loc='upper right', bbox_to_anchor=(0.933, 0.94))
ax1.legend()

ax.grid(True)

try:
    plt.savefig(f'{country}_2.png')
except:
    pass
add_text(f'{country}.docx','\n\n                                        ')
add_image_to_docx(f'{country}.docx',f'{country}_2.png')
maxgdp_year=int(*data[data['GDP']==max(data['GDP'])].index)
maxunemp_year=int(*data[data['Unemployment rate (%)']==max(data['Unemployment␣
 ↪rate (%)'])].index)
mingdp_year=int(*data[data['GDP']==min(data['GDP'])].index)
minunemp_year=int(*data[data['Unemployment rate (%)']==min(data['Unemployment␣
 ↪rate (%)'])].index)
add_text(f'{country}.docx',f'\t                ,                                ␣
 ↪       {maxgdp_year}    ,           {maxunemp_year}    ,           {mingdp_year}␣
 ↪ {minunemp_year}                          .',par_allign='LEFT')
plt.show()
```



[5]:
```
X = data['GDP'].values
Y = data['Unemployment rate (%)'].values
n = X.shape[0]
```

[6]:
```
sumX, sumY = sum(X), sum(Y)
meanX, meanY = X.mean(), Y.mean()

sumX2, sumY2 = sum(X**2), sum(Y**2)
meanX2, meanY2 = (X**2).mean(), (Y**2).mean()

sumDX, sumDY = sum((X-meanX)**2), sum((Y-meanY)**2)
meanDX, meanDY = ((X-meanX)**2).mean(), ((Y-meanY)**2).mean()

sumXY = sum(X*Y)
meanXY = (X*Y).mean()

b0, b1 = sp.symbols('b1 b2', real=True)

b0, b1 = sp.solve((-sumY + n*b0 + b1*sumX, -sumXY + b0*sumX + b1*sumX2), (b0,␣
 ↪b1)).values()

b0, b1
```

[6]:
$$(9.87633181714691,\ -2.54065643359145 \cdot 10^{-12})$$

[7]:
```
text = '          ,                                    ,                  ␣
 ↪   ,              .' if str(b1/abs(b1))[0]=='-' else '          ,       ␣
 ↪                          ,                          ,                  ␣
 ↪      .'
add_text(f'{country}.docx', '\t'+text,par_allign='LEFT')
```

[8]:
```
rXY = (meanXY - meanX*meanY) / np.sqrt((meanX2-meanX**2)*(meanY2-meanY**2))
dXY = rXY**2
rXY, dXY
```

[8]:
$$(-0.723472587641463,\ 0.523412585068634)$$

[9]:
```
text = f'                              b0, b1,                          .'
add_text(f'{country}.docx', '\t'+text,par_allign='LEFT')
```

[10]:
```
S = np.sqrt(sumDY/(n-2))

Sb0 = S * np.sqrt(sumX2 / (n * sumDX))
Sb1 = S * np.sqrt(1 / sumDX)

tb0 = b0/Sb0
tb1 = b1/Sb1


t_table = abs(st.t.ppf(1 - (1-0.95)/2, n-2))

text1=f'b0 {"           " if abs(tb0)>t_table else "             "}      {b0}'
```

```
text2=f'b1 {"            " if abs(tb1)>t_table else "              "}      {b1}'
add_text(f'{country}.docx', '\t'+text1,par_allign='LEFT')
add_text(f'{country}.docx', '\t'+text2,par_allign='LEFT')
```

[11]:
```
Fr = rXY**2 / (1-rXY**2) * (n-2)
F_table = st.f.ppf(0.95, 1, n-2)
display(Fr)
h0 = f"{Fr}>{F_table}" +"\n\n\tH0       ,                          " if␣
 ↪Fr>F_table else f"{Fr}>{F_table}"+"\n\tH0 is valid =>                    "
text = '                              : \n\n\t '+f'F_    = {Fr}' '\n\t F_    =␣
 ↪'f'{F_table}' + '\n\n\t' + h0
add_text(f'{country}.docx', '\t'+text,par_allign='LEFT')
```

34.045779701221

[12]:
```
st.f.ppf(0.95, 1, n-2)
```

[12]: 4.15961509803176

[13]:
```
#
dY = S*t_table*np.sqrt( 1 + 1/n + (1.1*meanX-meanX)**2/sumDX )
dY
```

[13]: 5.27765713269743

[14]:
```
equation = f'y = {b0} - {str(b1)[1:]}x'
equation
```

[14]: 'y = 9.87633181714691 - 2.54065643359145e-12x'

[15]:
```
df1 = pd.DataFrame({'                    ': (b0, b1, rXY, dXY, equation)},␣
 ↪index=('     b0', '     b1', '             (r)', '              (r^2)',␣
 ↪'           '))
df2 = pd.DataFrame({'          ': (0.95, n-2, t_table, tb0, tb1)}, index=('    ␣
 ↪      ', '             ', '                 ', 't-     (b0)', 't-     (b1)'))
df3 = pd.DataFrame({'         ': (0.95, n-2, F_table, Fr)}, index=('    ␣
 ↪      ', '             ', '               ', '              '))
display(df1)
display(df2)
display(df3)
try:
    text='\n\t                                                                ␣
 ↪          '
    add_text(f'{country}.docx',text,par_allign="LEFT")
    add_text(f'{country}.docx','\n                    ')
    df1.to_excel(f'{country}_3.xlsx')
    add_excel_table_to_docx(f'{country}_3.xlsx',f'{country}.docx',from_row=2)
    add_text(f'{country}.docx','\n          ')
```

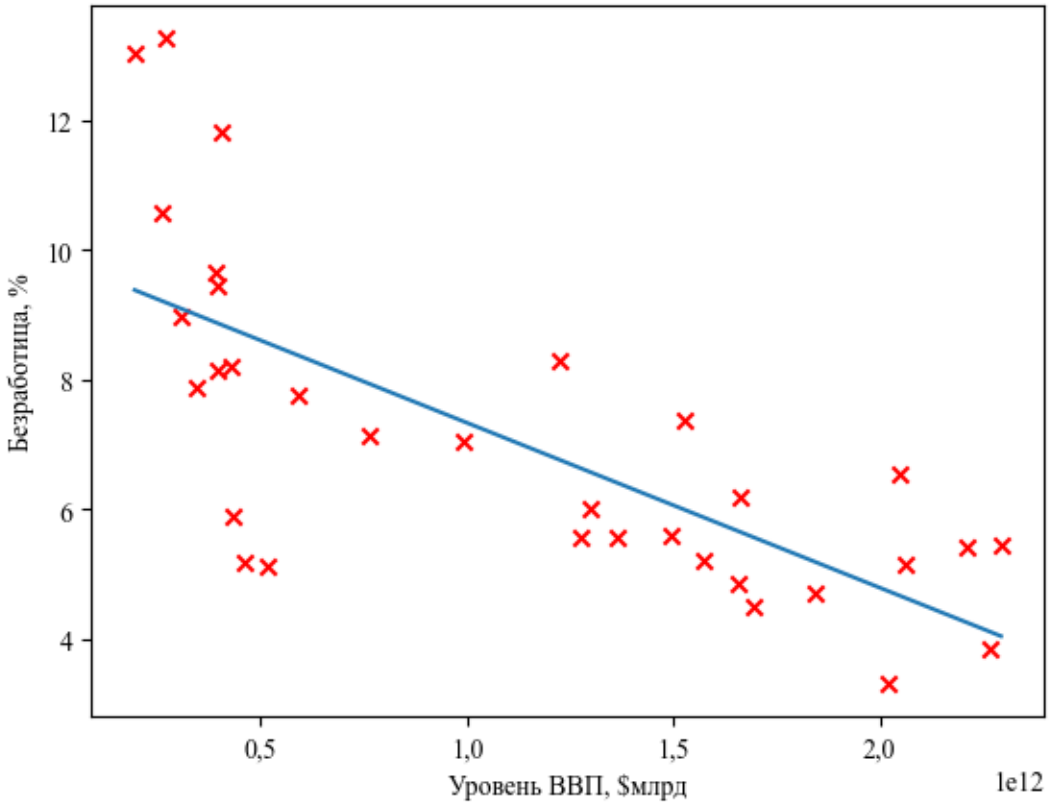
```
    df2.to_excel(f'{country}_4.xlsx')
    add_excel_table_to_docx(f'{country}_4.xlsx',f'{country}.docx',from_row=2)
    add_text(f'{country}.docx','\n         ')
    df3.to_excel(f'{country}_5.xlsx')
    add_excel_table_to_docx(f'{country}_5.xlsx',f'{country}.docx',from_row=2)
except:
    pass
```

```
     b0                                                  9.87633181714691
     b1                                                -2.54065643359145e-12
             (r)                                             -0.723473
              (r^2)                                          0.523413
                       y = 9.87633181714691 - 2.54065643359145e-12x


                                        0.95
                                         31
                                    2.039513
t-      (b0)                         11.9096508242394
t-      (b1)                        -4.02812489095455


                                    0.950000
                                   31.000000
                                  4.159615
                                  34.045780
```

[16]:
```
text=f'''
    \t                                          ,                                 ␣
 ↪        {dY}.
    \n\t                                  ,     95 %                               ␣
 ↪           .
    \n\t                  ,                   , «b0»                  ,     ␣
 ↪t-           «b1»                          ,                              .
    \n\t{'       ' if str(rXY/abs(rXY))[0]=='-' else '       '}                   ␣
 ↪      {rXY}                                 .
    \n\t                                                 ,                      , ␣
 ↪                  :
    \n\t          :
    \n\t{equation}
'''
add_text(f'{country}.docx',text,par_allign='LEFT')
```

[17]:
```
plt.scatter(X, Y, color='r', marker='x')
plt.plot(np.linspace(min(X),max(X),10000),b0-abs(b1)*np.
 ↪linspace(min(X),max(X),10000))
```

```
plt.xlabel('       , $   ')
plt.ylabel('       , %')
try:
    plt.savefig(f'{country}_6.png')
except:
    pass
add_image_to_docx(f'{country}.docx',f'{country}_6.png',0,6)

plt.show()
```



[18]:
```
text=f'''
    \n\t                        {'    ' if str(rXY/abs(rXY))[0]=='-' else '   '}␣
 ↪                              .
    \n\n\t .                        :
    \n\tUt - Ut-1= -k (T   -        ) (1)
    \n\t• Ut -                                    (   . Unemployment -       );
    \n\t• Ut-1 -                                   ;
    \n\t• T   -                          ;
    \n\t•          -          ,                                ;
    \n\n\t                                .                                  :
    \n\n\tk = (-1) * (Ut - Ut-1) / (T   -        )
```

```
    \n\n\t                                              .                  ,␣
↪                                          ,                          .␣
↪                       2,5 %
'''
add_text(f'{country}.docx',text,par_allign='LEFT')
```

[19]:
```
def OKUN(Ut, Ut_1, T):
    return (-1)*(Ut - Ut_1) / (T - 100*GDP_norm)
```

[20]:
```
data_p = data.copy()
data_p['         , %'] = GDP_norm*100

O = []
base_year = data_p.iloc[0,:]['Unemployment rate (%)']
for i in range(0,X.shape[0]):
  j = data_p.iloc[i,:]
  O.append(OKUN( j['Unemployment rate (%)'], base_year, j['GDP growth (annual␣
↪%)']))
  base_year = j['Unemployment rate (%)']
data_p['Oyken'] = O
data_p.to_excel(f'{country}_7.xlsx')
add_excel_table_to_docx(f'{country}_7.xlsx',f'{country}.docx')
data_p
```

[20]:

date	GDP	GDP growth (annual %)	Unemployment rate (%) \
1991	5.179630e+11	-5.046939	5.133
1992	4.602906e+11	-14.531074	5.181
1993	4.350837e+11	-8.668540	5.883
1994	3.950773e+11	-12.569756	8.131
1995	3.955372e+11	-4.143528	9.449
1996	3.917249e+11	-3.755069	9.665
1997	4.049290e+11	1.399916	11.813
1998	2.709555e+11	-5.299962	13.261
1999	1.959071e+11	6.399915	13.036
2000	2.597101e+11	10.000067	10.581
2001	3.066021e+11	5.100051	8.978
2002	3.454705e+11	4.699992	7.875
2003	4.303474e+11	7.299952	8.210
2004	5.910167e+11	7.199948	7.763
2005	7.640160e+11	6.399965	7.124
2006	9.899321e+11	8.200068	7.055
2007	1.299703e+12	8.499978	6.002
2008	1.660848e+12	5.199969	6.205
2009	1.222646e+12	-7.799994	8.301
2010	1.524917e+12	4.500000	7.369
2011	2.045923e+12	4.300029	6.536

2012	2.208294e+12	4.024086	5.436
2013	2.292470e+12	1.755422	5.458
2014	2.059242e+12	0.736267	5.160
2015	1.363482e+12	-1.972719	5.571
2016	1.276786e+12	0.193690	5.559
2017	1.574199e+12	1.825790	5.212
2018	1.657329e+12	2.807245	4.846
2019	1.693115e+12	2.198076	4.496
2020	1.493076e+12	-2.653655	5.589
2021	1.843392e+12	5.614290	4.715
2022	2.266029e+12	-2.069712	3.867
2023	2.021421e+12	3.600000	3.325

, %	Oyken
date	
1991	2.5 0.000000
1992	2.5 0.002818
1993	2.5 0.062855
1994	2.5 0.149173
1995	2.5 0.198389
1996	2.5 0.034532
1997	2.5 1.952578
1998	2.5 0.185642
1999	2.5 0.057694
2000	2.5 0.327330
2001	2.5 0.616526
2002	2.5 0.501365
2003	2.5 -0.069792
2004	2.5 0.095107
2005	2.5 0.163848
2006	2.5 0.012105
2007	2.5 0.175501
2008	2.5 -0.075186
2009	2.5 0.203495
2010	2.5 0.466000
2011	2.5 0.462770
2012	2.5 0.721744
2013	2.5 0.029547
2014	2.5 -0.168960
2015	2.5 0.091890
2016	2.5 -0.005203
2017	2.5 -0.514676
2018	2.5 1.191230
2019	2.5 -1.159231
2020	2.5 0.212083
2021	2.5 0.280642
2022	2.5 -0.185570

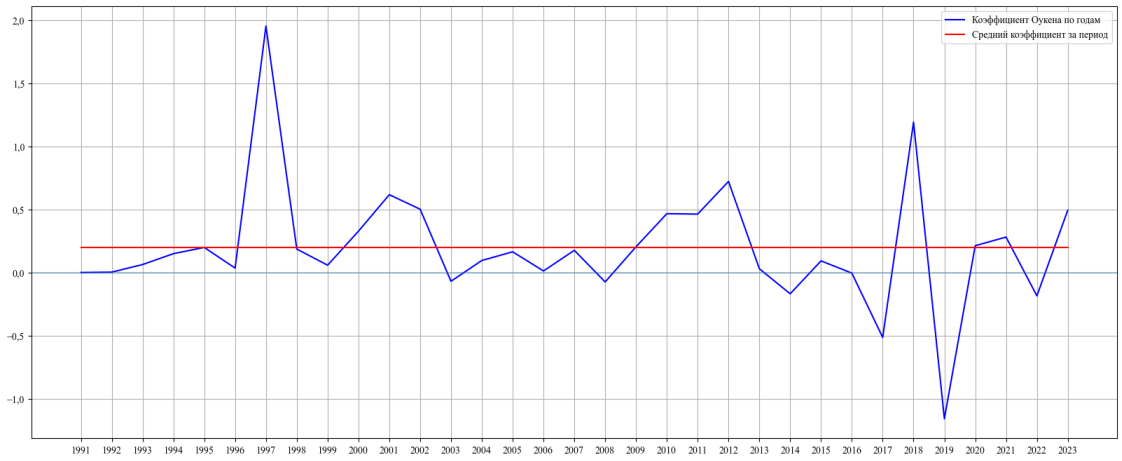
2023 2.5 0.492727

[21]:
```
k = data_p['Oyken'].mean()
print(f'                : {k}')
```

: 0.19724161785452324

[22]:
```
text=f'''
    \n\n\t                                  (      )      .
 ↪  1991-2023          (    {k},        ),                  (
 ↪           )
'''
add_text(f'{country}.docx',text,par_allign='LEFT')
```

[23]:
```
fig, ax = plt.subplots(figsize=(20, 8))

ax.plot(data_p['Oyken'], color='b', label='                ')
ax.plot([k]*n, color='r',label='                ')
loc = matplotlib.ticker.MultipleLocator(base=1)
plt.axhline(alpha=0.35)
ax.grid(True)
ax.legend()

try:
    plt.savefig(f'{country}_8.png')
except:
    pass

add_image_to_docx(f'{country}.docx',f'{country}_8.png')

plt.show()
```



[24]:
```
text=f'\n\t                    ,      {k},       ,                                  {k}%␣
 ↪                                .\n\n\t        ,              ,                      ␣
 ↪                                              {country_dict[country]}.    '
add_text(f'{country}.docx',text,par_allign="LEFT")
```