# imports

In [15]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
from sympy import *
import time as dt
import warnings
from IPython.display import Math,Latex
from pandas.errors import SettingWithCopyWarning

# sklearn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures,StandardScaler,MinMaxScaler
from sklearn.decomposition import PCA

from statsmodels.api import *

from scipy.stats import jarque_bera, kstest, kstwobign
```

# Задача 3.1.

Для оценки удовлетворенности населения жизнью проведен социологический опрос в 80 странах мира. В таблице представлены агрегированные результаты проведенного опроса в разрезе стран, единицей измерения показателей является доля (%) респондентов, положительно оценивающих параметр. В качестве основных параметров удовлетворенности жизнью выбраны следующие:

- x1 – работа, %;
- x2 – здоровье, %;
- x3 - материальное благополучие, %;
- x4 - достижение поставленных целей, %;
- x5 - социальный статус, %;
- x6 - социальные контакты, %.

Необходимо с помощью метода главных компонент выявить факторы удовлетворенности населения жизнью, ранжировать страны по уровню удовлетворенности населения жизнью.

## Возьмем данные из `Исходные данные для задачи 3.1.pdf`

In [16]:
```python
def get_countries_back(text:str)->str:
    repladict = {
        'Буркина-фасо':'Буркина-Фасо',
        'республика': ' Республика',
        'Тринидадитобаго':'Тринидад и Тобаго',
        'Боснияигерцеговина':'Босния и Герцеговина',
        'Сьерралеон':'Сьерра Леон',
        '-африканская': '-Африканская',

    }
    text = text.capitalize()
    for key,value in repladict.items():
        text = text.replace(key,value)

    return text

# Скопируем таблицу из Исходные данные для задачи 3.1.pdf
text3_1 ='''№ Страна x1 x2 x3 x4 x5 x6 1 Израиль 80 80 71 88 81 85 2 Греция 80 8

# Преобразцем полученную строку в массив так, чтобы страна была одним элементом
strs3_1 = np.array(text3_1.lower().replace(' и ','и').replace('  ',' ').replace(

# Изменим размерность массива и запишем данные в pandas.DataFrame
df3_1 = pd.DataFrame(strs3_1.reshape(strs3_1.size//8,8))

# Обозначим первую строку как заголовок, первый столбец за индекс и выполним обр
df3_1.columns = df3_1.iloc[0]
df3_1 = df3_1.drop(0).reset_index(drop=True)
df3_1.set_index('№', inplace=True)
df3_1['страна'] = df3_1['страна'].apply(get_countries_back)

# Переименуем столбцы в удобный вид
df3_1.columns = ['Страна']+[f'x{i}' for i in range(1,7)]

# Трансформируем значения числовых столбцов в числовой тип данных
for i in range(1,7):
    df3_1[f'x{i}']=df3_1[f'x{i}'].transform(int)


df3_1
```

Out[16]:

| № | Страна | x1 | x2 | x3 | x4 | x5 | x6 |
|---|---|---|---|---|---|---|---|
| **1** | Израиль | 80 | 80 | 71 | 88 | 81 | 85 |
| **2** | Греция | 80 | 82 | 57 | 90 | 92 | 79 |
| **3** | Словакия | 76 | 72 | 47 | 85 | 78 | 93 |
| **4** | Эстония | 79 | 64 | 46 | 72 | 79 | 85 |
| **5** | Венгрия | 83 | 69 | 43 | 88 | 88 | 90 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **76** | Чад | 78 | 69 | 52 | 93 | 79 | 57 |
| **77** | Мозамбик | 74 | 82 | 46 | 93 | 89 | 75 |
| **78** | Нигер | 54 | 82 | 52 | 99 | 93 | 77 |
| **79** | Конго | 60 | 74 | 40 | 98 | 79 | 67 |
| **80** | Зимбабве | 49 | 72 | 27 | 91 | 81 | 81 |

80 rows × 7 columns

# Перейдем к выполнению алгоритма главных компонент

## 1. Центрирование данных

In [17]:
```
data_centered = df3_1.iloc[:,1:] - np.mean(df3_1.iloc[:,1:], axis=0)
data_centered
```

Out[17]:

| № | x1 | x2 | x3 | x4 | x5 | x6 |
|---|---|---|---|---|---|---|
| **1** | 10.8125 | 6.5375 | 24.425 | -2.7 | -1.4 | 8.65 |
| **2** | 10.8125 | 8.5375 | 10.425 | -0.7 | 9.6 | 2.65 |
| **3** | 6.8125 | -1.4625 | 0.425 | -5.7 | -4.4 | 16.65 |
| **4** | 9.8125 | -9.4625 | -0.575 | -18.7 | -3.4 | 8.65 |
| **5** | 13.8125 | -4.4625 | -3.575 | -2.7 | 5.6 | 13.65 |
| **...** | ... | ... | ... | ... | ... | ... |
| **76** | 8.8125 | -4.4625 | 5.425 | 2.3 | -3.4 | -19.35 |
| **77** | 4.8125 | 8.5375 | -0.575 | 2.3 | 6.6 | -1.35 |
| **78** | -15.1875 | 8.5375 | 5.425 | 8.3 | 10.6 | 0.65 |
| **79** | -9.1875 | 0.5375 | -6.575 | 7.3 | -3.4 | -9.35 |
| **80** | -20.1875 | -1.4625 | -19.575 | 0.3 | -1.4 | 4.65 |

80 rows × 6 columns

## 2. Ковариационная матрица

In [18]:
```
cov_matrix = np.cov(data_centered, rowvar=False)
cov_matrix
```

Out[18]:
```
array([[177.14161392,  44.40585443, 118.89082278, -15.90506329,
         15.27848101,  46.90822785],
       [ 44.40585443,  83.41629747,  78.65474684,  27.45696203,
         16.52151899,  20.83607595],
       [118.89082278,  78.65474684, 205.36139241,  19.7443038 ,
         37.65316456,  34.08734177],
       [-15.90506329,  27.45696203,  19.7443038 ,  47.55443038,
         25.99493671, -10.61518987],
       [ 15.27848101,  16.52151899,  37.65316456,  25.99493671,
         55.07848101,  18.37721519],
       [ 46.90822785,  20.83607595,  34.08734177, -10.61518987,
         18.37721519, 109.39493671]])
```

## 3. Собственные значения и собственные векторы

In [19]:
```
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
print('Собственные значения = ' ,eigenvalues,'\n')
print('Собственные векторы = ' ,eigenvectors)
```

```
Собственные значения =  [ 13.04851148  43.29783624  47.54565899  91.62374906 119.
19843693
 363.23295919]

Собственные векторы =  [[ 0.11985032 -0.48057759 -0.18402864  0.38369624  0.48113
766 -0.58489168]
 [-0.38267687 -0.44473463  0.63460139 -0.19194274 -0.33942831 -0.31780026]
 [ 0.07171259  0.62146814 -0.01168077  0.12416326 -0.34038398 -0.69081144]
 [ 0.74643291 -0.36258469 -0.15389928 -0.22000562 -0.48702893 -0.04566848]
 [-0.49594227 -0.17519795 -0.72452153 -0.35322081 -0.22724366 -0.14836063]
 [ 0.17587151  0.15123381  0.12108531 -0.79205508  0.49859489 -0.2357706 ]]
```

## 4. Сортировка собственных значений и векторов

In [20]:
```
# Отсортируем по убыванию собственные значения
sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[sorted_indices]

# И векторы
eigenvectors = eigenvectors[:, sorted_indices]

print('Собственные значения = ' ,eigenvalues,'\n')
print('Собственные векторы = ' ,eigenvectors)
```

```
Собственные значения =  [363.23295919 119.19843693  91.62374906  47.54565899  43.
29783624
  13.04851148]

Собственные векторы =  [[-0.58489168  0.48113766  0.38369624 -0.18402864 -0.48057
759  0.11985032]
 [-0.31780026 -0.33942831 -0.19194274  0.63460139 -0.44473463 -0.38267687]
 [-0.69081144 -0.34038398  0.12416326 -0.01168077  0.62146814  0.07171259]
 [-0.04566848 -0.48702893 -0.22000562 -0.15389928 -0.36258469  0.74643291]
 [-0.14836063 -0.22724366 -0.35322081 -0.72452153 -0.17519795 -0.49594227]
 [-0.2357706   0.49859489 -0.79205508  0.12108531  0.15123381  0.17587151]]
```

## 5. Проекция данных

In [21]:
```
projected_data = np.dot(data_centered, eigenvectors)
projected_data
```

Out[21]:
```
array([[-2.69832360e+01,  6.15374714e-01,  1.63825443e-01,
         4.35084026e+00,  9.60808957e+00,  7.45950418e-01],
       [-1.82561566e+01, -1.76341356e+00, -1.53145527e+00,
        -3.22047344e+00, -3.54168322e+00, -6.04110777e+00],
       [-6.82587003e+00,  1.57070429e+01, -7.43209725e+00,
         3.89442707e+00,  2.99626006e+00,  2.26236217e+00],
       [-3.01583714e+00,  2.23216396e+01,  3.97366303e+00,
        -1.41530264e+00,  7.81946878e+00, -5.99492674e+00],
       [-8.11676515e+00,  1.62255194e+01, -6.48310804e+00,
        -7.32102359e+00, -4.81288672e+00,  7.14756170e-01],
       [-1.86872208e+01,  9.91811710e+00, -5.68191267e+00,
        -6.27679253e+00, -1.33631682e+01, -2.39048103e+00],
       [-2.52283917e+01,  5.86358434e+00, -4.51031597e+00,
        -7.65427989e+00,  8.93447929e+00, -1.24215390e+00],
       [-2.46712148e+01, -2.04509038e-01, -1.57591966e+00,
        -9.48458339e+00,  7.24623759e+00, -1.48078338e+00],
       [ 7.46486617e+00,  1.99594493e+01,  6.20260882e+00,
        -4.54746766e+00, -3.58680979e+00, -3.04652598e+00],
       [-8.88337487e+00,  1.50190405e+01, -3.27119549e+00,
         9.53032474e+00,  1.40515963e+00,  6.23679767e-01],
       [ 1.22953680e+01,  2.02280195e+01,  1.75951578e+00,
         1.98645191e+00, -3.26369063e+00, -5.06061021e+00],
       [-6.05545889e+00,  1.45368665e+00, -1.18226063e+01,
        -3.36109040e+00, -1.39902191e+01, -1.95530585e+00],
       [ 5.68824783e+00,  1.32385382e+01, -9.79255508e-01,
         4.76776602e+00, -4.59014863e+00, -1.52549400e+00],
       [ 1.63086046e+01,  2.74941043e+01,  1.12749818e-01,
         1.87305313e+00,  1.31924259e+01, -1.96715327e+00],
       [-9.32529848e+00, -2.47524714e+00, -2.54706644e+00,
        -7.17710430e+00,  2.74809887e-01,  2.81784552e+00],
       [ 2.42761577e+00,  1.28270539e+01,  6.12165506e+00,
         1.32083848e+01,  3.27129306e+00, -2.37972895e+00],
       [ 7.73870415e+00,  2.32128251e+01, -2.97999669e+00,
        -9.06728802e+00,  4.78036918e+00, -4.81013423e-01],
       [-1.12304991e+01,  1.39542540e+01, -1.62489966e+00,
        -3.03556241e+00,  2.00808860e+00,  4.67113629e+00],
       [ 4.36554840e+00,  5.65146887e+00,  7.40369336e+00,
        -1.60860034e+00, -9.66686568e-01,  3.78577525e-01],
       [ 4.08717153e+00,  1.28733518e+01,  1.10332720e+01,
         1.23302898e+01, -2.44303032e+00, -1.07772748e+00],
       [ 2.14122680e+01,  2.66150309e+01,  2.85725264e+00,
        -5.45344661e+00,  2.18027395e-01, -3.87370496e+00],
       [-5.83343591e+00, -9.92823069e+00,  1.27773341e+01,
         4.83213026e+00, -1.54643318e-02, -7.03693533e+00],
       [ 4.62726388e+00,  2.27517757e+00, -3.82300936e+00,
         6.09139574e+00, -1.28220625e+01, -1.25035798e+00],
       [ 3.34471957e+01,  4.36082767e+00,  1.76025489e+01,
        -1.58812461e+01, -3.64546564e+00, -1.26188302e+00],
       [ 2.31714143e+01,  1.02588072e+00,  3.42071729e+00,
        -1.75648266e+01, -4.85421829e-02,  2.53150545e+00],
       [-1.66272928e+01, -4.34891369e+00, -1.70100952e+00,
        -9.10488287e+00, -4.10038476e+00,  1.55460296e+00],
       [-2.46997542e+01, -8.11065604e-01,  2.21939822e+00,
         1.02591946e+01,  7.53507769e+00,  3.45142074e+00],
       [-1.79113289e+01,  4.35881091e+00, -9.81091832e+00,
         9.21693092e+00, -1.05050196e+01,  5.43377225e+00],
       [ 5.22077549e+00,  7.77978096e-01,  1.60109662e+01,
         1.11217595e+01, -8.77995762e-01, -2.23585852e-01],
       [-1.26396114e+01, -6.80627038e+00, -1.06485376e+01,
        -2.78333030e+00,  1.21480044e+00, -1.23615253e+00],
```

```
[-1.90869769e+01, -7.36106931e+00,  5.05632112e+00,
 -4.64410334e+00, -4.82010039e+00,  6.60840984e-01],
[-1.92465270e+01,  7.12482420e+00,  4.90996115e+00,
  4.29236815e+00, -6.85723646e-01,  5.87221488e+00],
[-2.62949564e+01,  5.42886395e+00,  6.53857152e+00,
  4.69194831e+00, -2.14588102e+00,  9.54634960e+00],
[-2.65588097e+01, -2.70339920e+00, -1.11265185e+00,
 -4.59646936e+00,  1.91922777e+00,  6.88830407e-01],
[-2.90492382e+01,  5.37001744e-01, -9.24533088e+00,
 -5.09042933e+00, -3.38143944e+00, -3.76268959e+00],
[-2.61199471e+01, -6.74093349e+00,  1.50277784e+00,
 -9.68866416e+00,  1.24803496e+00,  1.55635912e-01],
[ 1.07322738e+01,  1.25464071e+00, -9.50948804e+00,
 -1.80673547e+00,  5.21510044e+00,  2.57476911e+00],
[ 1.05668782e+01,  1.77265961e+01,  1.81502814e+00,
  1.18003585e+00,  8.74511109e+00,  1.56438007e+00],
[-2.62021906e+01, -4.52207662e-01,  1.50767398e-01,
  9.83840672e+00, -1.76617502e+00,  5.07755347e+00],
[-2.53662368e+01, -3.43593029e+00, -3.26867140e+00,
  2.76997210e+00, -6.44634883e+00,  2.46236250e+00],
[-2.69915381e+01, -4.11207426e+00, -1.52622442e+00,
 -3.21823701e+00, -2.27224105e+00, -7.90832795e-01],
[-1.20774273e+01, -1.49178416e+01, -7.93339086e+00,
 -4.06370723e-01,  5.32658466e+00, -4.54638412e+00],
[-8.89679642e+00,  6.92119573e+00, -4.73378648e+00,
 -2.90435860e+00, -3.01985842e+00,  9.12508867e-01],
[ 1.41530307e-01,  7.48048182e-01, -1.36793712e+01,
  4.16050153e+00, -4.40162214e+00,  3.62468396e+00],
[-1.75226186e+01, -8.26575740e+00,  1.70550089e+01,
  2.30845573e+00,  8.31354117e+00,  3.47778878e+00],
[-1.43705930e+01, -9.17122934e+00, -5.53691544e+00,
 -4.90660427e+00, -2.05998595e-02,  2.63008739e-01],
[-2.43700516e+01, -1.01845731e+01, -8.85815281e+00,
  5.34799563e+00,  9.20981313e+00, -6.10849042e+00],
[-2.22346828e+01, -4.46108507e+00, -5.10179798e+00,
 -4.57041153e+00, -1.66541767e+00,  2.25370053e+00],
[-1.24768910e+01, -9.45394976e+00,  1.32283186e+01,
  1.25032018e+01,  1.66878579e+00,  7.57558695e-01],
[-9.53437686e+00,  1.17067216e+01,  1.58881011e+00,
 -6.02924985e+00,  3.10401150e+00, -5.20715432e+00],
[ 2.28276225e+01,  1.42102699e+00, -8.84133402e+00,
  2.68286043e+00, -7.48646967e+00,  6.41430956e+00],
[-1.05120912e+01, -1.64507779e+01,  2.08857376e+01,
 -8.40706357e+00,  1.60561964e+00, -1.75335804e+00],
[ 2.19249189e+01, -1.19780834e+01,  1.03349081e+00,
 -8.59116645e+00, -2.84430550e+00,  4.57512442e-01],
[ 9.87834258e+00, -2.60560872e+00, -1.10495196e+00,
 -4.25979056e+00, -9.24043560e-01,  3.32787630e-01],
[-9.12665577e+00, -1.81221946e+00,  2.48759771e+00,
  2.28084826e+00, -7.32803201e-01, -4.51052003e+00],
[ 2.87566796e+01, -5.36356144e+00, -1.19604702e+01,
  9.31661407e+00, -4.89209230e+00,  1.37951563e+00],
[ 3.48784508e+00, -7.28064652e+00, -1.07938546e+01,
  4.07731231e+00,  3.07222313e+00, -2.30412278e+00],
[ 6.08763793e+00, -4.47963610e+00, -2.39856707e-02,
  5.28366759e+00, -5.86513697e+00, -2.57549264e+00],
[ 1.86942290e+01,  1.86758989e+00, -1.26483700e+01,
 -1.95652186e-01,  4.77630410e+00,  8.01451634e+00],
[ 3.15106010e+01, -3.45161778e+00, -1.71922596e+01,
  1.25842502e+00,  5.63567678e+00, -4.67196641e+00],
```

```
       [ 3.15603138e+01,  5.10699131e+00,  1.36046046e+01,
         1.10706454e+00,  1.60594432e+01,  4.30706905e+00],
       [-4.51156464e-01, -7.92523811e+00,  1.77177700e+01,
        -7.25432065e+00,  3.51042414e+00, -7.04716402e-01],
       [ 3.50007290e+01, -1.09852740e+00, -8.91744843e+00,
         6.03065104e+00, -1.53636275e+00,  5.05414176e+00],
       [ 2.28265820e+01, -1.58652810e+01,  8.61877048e-02,
         4.39924085e+00, -2.75997606e+00, -6.28203665e+00],
       [ 3.18307229e+01, -1.46953631e+01,  8.43168553e+00,
         1.15809323e+00,  1.06514646e+01, -3.36011498e+00],
       [-9.14196464e+00, -1.80738878e+01, -1.63191292e+00,
        -2.49733483e+00,  8.06155647e+00,  1.68752360e+00],
       [-1.49617636e+01, -7.40754750e+00, -1.38589702e+01,
        -1.40769345e+00,  9.74076981e+00,  3.04808219e+00],
       [ 1.09255723e+00, -6.40667906e+00,  2.63260527e+01,
         1.49154683e+01,  3.29463670e+00, -1.99401505e+00],
       [ 1.72788244e+01, -6.97875225e+00,  8.91533902e+00,
        -4.22297440e+00, -1.76058842e+01, -3.19100184e+00],
       [ 2.02469409e+01, -3.92841296e+00, -6.49228724e+00,
         1.35089510e+01,  3.57059551e-01, -2.55677550e+00],
       [ 4.32313608e+01, -3.23254815e+00,  6.54023868e+00,
        -1.49659287e+01, -6.92031255e-01,  8.82154239e+00],
       [ 9.01172901e+00, -3.83574728e+00,  1.79200579e+01,
         6.14971224e+00, -2.07942876e+01,  1.59785473e+00],
       [ 3.45583692e+01, -1.79103959e+01, -1.86498108e+01,
         1.79341847e+00,  5.78262794e+00, -7.70354369e-01],
       [ 2.87353024e+01, -6.73192718e+00, -8.94741722e+00,
         9.50291516e-01, -1.29023365e+00, -1.28129089e+00],
       [ 1.84359022e+01, -2.28919384e+01,  4.70939428e+00,
        -1.47082555e+00,  5.76826605e+00,  2.53756565e+00],
       [-2.52227648e+00, -6.08720782e+00,  2.09326570e+01,
        -4.75062517e+00, -2.04364304e+00,  3.15280292e+00],
       [-5.89672172e+00, -3.67975112e+00, -1.63156271e+00,
        -7.60287610e-01, -8.66146260e+00, -4.52540867e+00],
       [ 3.17248632e-01, -1.81787667e+01, -1.28775351e+01,
        -7.29110607e-01,  2.10506576e+00, -3.64556816e+00],
       [ 1.21204613e+01, -9.80941528e+00,  2.55587285e+00,
         2.31642327e+00, -3.37512253e+00,  3.71244143e+00],
       [ 2.48925887e+01, -6.30375935e-02, -1.31501962e+01,
         4.54683176e+00, -9.73415271e-01, -1.52753576e+00]])
```

## 6. Результаты

In [22]:
```
results = {
        "explained_variance": eigenvalues,
        "explained_variance_ratio": eigenvalues / np.sum(eigenvalues),
        "components": eigenvectors,
        "projected_data": projected_data
    }

print("Объяснённая дисперсия:", results['explained_variance'],'\n' )
print("Доля объяснённой дисперсии:", results['explained_variance_ratio'],'\n')
print("Главные компоненты (векторы):\n", results['components'],'\n')
print("Проекция данных (первые строки):\n", results['projected_data'][:5],'\n')
```

```
Объяснённая дисперсия: [363.23295919 119.19843693  91.62374906  47.54565899  43.2
9783624
  13.04851148]

Доля объяснённой дисперсии: [0.53578359 0.17582261 0.13514881 0.07013181 0.063866
09 0.01924709]

Главные компоненты (векторы):
 [[-0.58489168  0.48113766  0.38369624 -0.18402864 -0.48057759  0.11985032]
 [-0.31780026 -0.33942831 -0.19194274  0.63460139 -0.44473463 -0.38267687]
 [-0.69081144 -0.34038398  0.12416326 -0.01168077  0.62146814  0.07171259]
 [-0.04566848 -0.48702893 -0.22000562 -0.15389928 -0.36258469  0.74643291]
 [-0.14836063 -0.22724366 -0.35322081 -0.72452153 -0.17519795 -0.49594227]
 [-0.2357706   0.49859489 -0.79205508  0.12108531  0.15123381  0.17587151]]

Проекция данных (первые строки):
 [[-26.98323599   0.61537471   0.16382544   4.35084026   9.60808957
    0.74595042]
 [-18.25615656  -1.76341356  -1.53145527  -3.22047344  -3.54168322
   -6.04110777]
 [ -6.82587003  15.70704291  -7.43209725   3.89442707   2.99626006
    2.26236217]
 [ -3.01583714  22.32163963   3.97366303  -1.41530264   7.81946878
   -5.99492674]
 [ -8.11676515  16.2255194   -6.48310804  -7.32102359  -4.81288672
    0.71475617]]
```

# Выведем полный алгоритм функцией

In [23]:

```python
def pca_manual(data, n_components=None, print_results=False):
    """
    Реализация PCA вручную.

    :param data: DataFrame или массив данных (каждая строка — наблюдение, столбе
    :param n_components: Число главных компонент (если None, используются все).
    :return: Словарь с результатами PCA.
    """
    import numpy as np
    import pandas as pd

    def fix_signs(vectors):
        for i in range(vectors.shape[1]):
            # Если первый элемент отрицательный, инвертировать знак всего вектор
            if vectors[0, i] < 0:
                vectors[:, i] *= -1
        return vectors
    # 1. Центрирование данных
    data_centered = data - np.mean(data, axis=0)

    # 2. Ковариационная матрица
    cov_matrix = np.cov(data_centered, rowvar=False)

    # 3. Собственные значения и собственные векторы
    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
    eigenvectors=fix_signs(eigenvectors)
    # 4. Сортировка собственных значений и векторов
    sorted_indices = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[sorted_indices]
```

```
    eigenvectors = eigenvectors[:, sorted_indices]

    # 5. Выбор числа компонент
    if n_components is not None:
        eigenvectors = eigenvectors[:, :n_components]
        eigenvalues = eigenvalues[:n_components]

    # 6. Проекция данных
    projected_data = np.dot(data_centered, eigenvectors)

    loadings = pd.DataFrame(
        eigenvectors.transpose(),
        columns=[f'x{i+1}' for i in range(data.shape[1]-1)] + ['PCA_score'],
        index=[f'PC{i+1}' for i in range(len(eigenvectors.transpose()))]
    )

    # 7. Результаты
    results = {
        "explained_variance": eigenvalues,
        "explained_variance_ratio": eigenvalues / np.sum(eigenvalues),
        "components": eigenvectors,
        "projected_data": projected_data,
        'loadings':loadings
    }
    if print_results:
        # Вывод результатов
        print("Объяснённая дисперсия:", results['explained_variance'],'\n' )
        print("Доля объяснённой дисперсии:", results['explained_variance_ratio']
        print("Главные компоненты (векторы):\n", results['components'],'\n')
        print("Проекция данных (первые строки):\n", results['projected_data'],'\

    return results

# Применение PCA
results = pca_manual(df3_1.iloc[:,1:].values, n_components=6)
results
```

Out[23]:

```
{'explained_variance': array([363.23295919, 119.19843693,  91.62374906,  47.545
65899,
         43.29783624,  13.04851148]),
 'explained_variance_ratio': array([0.53578359, 0.17582261, 0.13514881, 0.07013
181, 0.06386609,
        0.01924709]),
 'components': array([[ 0.58489168,  0.48113766,  0.38369624,  0.18402864,  0.4
8057759,
          0.11985032],
        [ 0.31780026, -0.33942831, -0.19194274, -0.63460139,  0.44473463,
         -0.38267687],
        [ 0.69081144, -0.34038398,  0.12416326,  0.01168077, -0.62146814,
          0.07171259],
        [ 0.04566848, -0.48702893, -0.22000562,  0.15389928,  0.36258469,
          0.74643291],
        [ 0.14836063, -0.22724366, -0.35322081,  0.72452153,  0.17519795,
         -0.49594227],
        [ 0.2357706 ,  0.49859489, -0.79205508, -0.12108531, -0.15123381,
          0.17587151]]),
 'projected_data': array([[ 2.69832360e+01,  6.15374714e-01,  1.63825443e-01,
         -4.35084026e+00, -9.60808957e+00,  7.45950418e-01],
        [ 1.82561566e+01, -1.76341356e+00, -1.53145527e+00,
          3.22047344e+00,  3.54168322e+00, -6.04110777e+00],
        [ 6.82587003e+00,  1.57070429e+01, -7.43209725e+00,
         -3.89442707e+00, -2.99626006e+00,  2.26236217e+00],
        [ 3.01583714e+00,  2.23216396e+01,  3.97366303e+00,
          1.41530264e+00, -7.81946878e+00, -5.99492674e+00],
        [ 8.11676515e+00,  1.62255194e+01, -6.48310804e+00,
          7.32102359e+00,  4.81288672e+00,  7.14756170e-01],
        [ 1.86872208e+01,  9.91811710e+00, -5.68191267e+00,
          6.27679253e+00,  1.33631682e+01, -2.39048103e+00],
        [ 2.52283917e+01,  5.86358434e+00, -4.51031597e+00,
          7.65427989e+00, -8.93447929e+00, -1.24215390e+00],
        [ 2.46712148e+01, -2.04509038e-01, -1.57591966e+00,
          9.48458339e+00, -7.24623759e+00, -1.48078338e+00],
        [-7.46486617e+00,  1.99594493e+01,  6.20260882e+00,
          4.54746766e+00,  3.58680979e+00, -3.04652598e+00],
        [ 8.88337487e+00,  1.50190405e+01, -3.27119549e+00,
         -9.53032474e+00, -1.40515963e+00,  6.23679767e-01],
        [-1.22953680e+01,  2.02280195e+01,  1.75951578e+00,
         -1.98645191e+00,  3.26369063e+00, -5.06061021e+00],
        [ 6.05545889e+00,  1.45368665e+00, -1.18226063e+01,
          3.36109040e+00,  1.39902191e+01, -1.95530585e+00],
        [-5.68824783e+00,  1.32385382e+01, -9.79255508e-01,
         -4.76776602e+00,  4.59014863e+00, -1.52549400e+00],
        [-1.63086046e+01,  2.74941043e+01,  1.12749818e-01,
         -1.87305313e+00, -1.31924259e+01, -1.96715327e+00],
        [ 9.32529848e+00, -2.47524714e+00, -2.54706644e+00,
          7.17710430e+00, -2.74809887e-01,  2.81784552e+00],
        [-2.42761577e+00,  1.28270539e+01,  6.12165506e+00,
         -1.32083848e+01, -3.27129306e+00, -2.37972895e+00],
        [-7.73870415e+00,  2.32128251e+01, -2.97999669e+00,
          9.06728802e+00, -4.78036918e+00, -4.81013423e-01],
        [ 1.12304991e+01,  1.39542540e+01, -1.62489966e+00,
          3.03556241e+00, -2.00808860e+00,  4.67113629e+00],
        [-4.36554840e+00,  5.65146887e+00,  7.40369336e+00,
          1.60860034e+00,  9.66686568e-01,  3.78577525e-01],
        [-4.08717153e+00,  1.28733518e+01,  1.10332720e+01,
         -1.23302898e+01,  2.44303032e+00, -1.07772748e+00],
        [-2.14122680e+01,  2.66150309e+01,  2.85725264e+00,
```

```
  5.45344661e+00, -2.18027395e-01, -3.87370496e+00],
[ 5.83343591e+00, -9.92823069e+00,  1.27773341e+01,
 -4.83213026e+00,  1.54643318e-02, -7.03693533e+00],
[-4.62726388e+00,  2.27517757e+00, -3.82300936e+00,
 -6.09139574e+00,  1.28220625e+01, -1.25035798e+00],
[-3.34471957e+01,  4.36082767e+00,  1.76025489e+01,
  1.58812461e+01,  3.64546564e+00, -1.26188302e+00],
[-2.31714143e+01,  1.02588072e+00,  3.42071729e+00,
  1.75648266e+01,  4.85421829e-02,  2.53150545e+00],
[ 1.66272928e+01, -4.34891369e+00, -1.70100952e+00,
  9.10488287e+00,  4.10038476e+00,  1.55460296e+00],
[ 2.46997542e+01, -8.11065604e-01,  2.21939822e+00,
 -1.02591946e+01, -7.53507769e+00,  3.45142074e+00],
[ 1.79113289e+01,  4.35881091e+00, -9.81091832e+00,
 -9.21693092e+00,  1.05050196e+01,  5.43377225e+00],
[-5.22077549e+00,  7.77978096e-01,  1.60109662e+01,
 -1.11217595e+01,  8.77995762e-01, -2.23585852e-01],
[ 1.26396114e+01, -6.80627038e+00, -1.06485376e+01,
  2.78333030e+00, -1.21480044e+00, -1.23615253e+00],
[ 1.90869769e+01, -7.36106931e+00,  5.05632112e+00,
  4.64410334e+00,  4.82010039e+00,  6.60840984e-01],
[ 1.92465270e+01,  7.12482420e+00,  4.90996115e+00,
 -4.29236815e+00,  6.85723646e-01,  5.87221488e+00],
[ 2.62949564e+01,  5.42886395e+00,  6.53857152e+00,
 -4.69194831e+00,  2.14588102e+00,  9.54634960e+00],
[ 2.65588097e+01, -2.70339920e+00, -1.11265185e+00,
  4.59646936e+00, -1.91922777e+00,  6.88830407e-01],
[ 2.90492382e+01,  5.37001744e-01, -9.24533088e+00,
  5.09042933e+00,  3.38143944e+00, -3.76268959e+00],
[ 2.61199471e+01, -6.74093349e+00,  1.50277784e+00,
  9.68866416e+00, -1.24803496e+00,  1.55635912e-01],
[-1.07322738e+01,  1.25464071e+00, -9.50948804e+00,
  1.80673547e+00, -5.21510044e+00,  2.57476911e+00],
[-1.05668782e+01,  1.77265961e+01,  1.81502814e+00,
 -1.18003585e+00, -8.74511109e+00,  1.56438007e+00],
[ 2.62021906e+01, -4.52207662e-01,  1.50767398e-01,
 -9.83840672e+00,  1.76617502e+00,  5.07755347e+00],
[ 2.53662368e+01, -3.43593029e+00, -3.26867140e+00,
 -2.76997210e+00,  6.44634883e+00,  2.46236250e+00],
[ 2.69915381e+01, -4.11207426e+00, -1.52622442e+00,
  3.21823701e+00,  2.27224105e+00, -7.90832795e-01],
[ 1.20774273e+01, -1.49178416e+01, -7.93339086e+00,
  4.06370723e-01, -5.32658466e+00, -4.54638412e+00],
[ 8.89679642e+00,  6.92119573e+00, -4.73378648e+00,
  2.90435860e+00,  3.01985842e+00,  9.12508867e-01],
[-1.41530307e-01,  7.48048182e-01, -1.36793712e+01,
 -4.16050153e+00,  4.40162214e+00,  3.62468396e+00],
[ 1.75226186e+01, -8.26575740e+00,  1.70550089e+01,
 -2.30845573e+00, -8.31354117e+00,  3.47778878e+00],
[ 1.43705930e+01, -9.17122934e+00, -5.53691544e+00,
  4.90660427e+00,  2.05998595e-02,  2.63008739e-01],
[ 2.43700516e+01, -1.01845731e+01, -8.85815281e+00,
 -5.34799563e+00, -9.20981313e+00, -6.10849042e+00],
[ 2.22346828e+01, -4.46108507e+00, -5.10179798e+00,
  4.57041153e+00,  1.66541767e+00,  2.25370053e+00],
[ 1.24768910e+01, -9.45394976e+00,  1.32283186e+01,
 -1.25032018e+01, -1.66878579e+00,  7.57558695e-01],
[ 9.53437686e+00,  1.17067216e+01,  1.58881011e+00,
  6.02924985e+00, -3.10401150e+00, -5.20715432e+00],
[-2.28276225e+01,  1.42102699e+00, -8.84133402e+00,
```

```
        -2.68286043e+00,  7.48646967e+00,  6.41430956e+00],
       [ 1.05120912e+01, -1.64507779e+01,  2.08857376e+01,
         8.40706357e+00, -1.60561964e+00, -1.75335804e+00],
       [-2.19249189e+01, -1.19780834e+01,  1.03349081e+00,
         8.59116645e+00,  2.84430550e+00,  4.57512442e-01],
       [-9.87834258e+00, -2.60560872e+00, -1.10495196e+00,
         4.25979056e+00,  9.24043560e-01,  3.32787630e-01],
       [ 9.12665577e+00, -1.81221946e+00,  2.48759771e+00,
        -2.28084826e+00,  7.32803201e-01, -4.51052003e+00],
       [-2.87566796e+01, -5.36356144e+00, -1.19604702e+01,
        -9.31661407e+00,  4.89209230e+00,  1.37951563e+00],
       [-3.48784508e+00, -7.28064652e+00, -1.07938546e+01,
        -4.07731231e+00, -3.07222313e+00, -2.30412278e+00],
       [-6.08763793e+00, -4.47963610e+00, -2.39856707e-02,
        -5.28366759e+00,  5.86513697e+00, -2.57549264e+00],
       [-1.86942290e+01,  1.86758989e+00, -1.26483700e+01,
         1.95652186e-01, -4.77630410e+00,  8.01451634e+00],
       [-3.15106010e+01, -3.45161778e+00, -1.71922596e+01,
        -1.25842502e+00, -5.63567678e+00, -4.67196641e+00],
       [-3.15603138e+01,  5.10699131e+00,  1.36046046e+01,
        -1.10706454e+00, -1.60594432e+01,  4.30706905e+00],
       [ 4.51156464e-01, -7.92523811e+00,  1.77177700e+01,
         7.25432065e+00, -3.51042414e+00, -7.04716402e-01],
       [-3.50007290e+01, -1.09852740e+00, -8.91744843e+00,
        -6.03065104e+00,  1.53636275e+00,  5.05414176e+00],
       [-2.28265820e+01, -1.58652810e+01,  8.61877048e-02,
        -4.39924085e+00,  2.75997606e+00, -6.28203665e+00],
       [-3.18307229e+01, -1.46953631e+01,  8.43168553e+00,
        -1.15809323e+00, -1.06514646e+01, -3.36011498e+00],
       [ 9.14196464e+00, -1.80738878e+01, -1.63191292e+00,
         2.49733483e+00, -8.06155647e+00,  1.68752360e+00],
       [ 1.49617636e+01, -7.40754750e+00, -1.38589702e+01,
         1.40769345e+00, -9.74076981e+00,  3.04808219e+00],
       [-1.09255723e+00, -6.40667906e+00,  2.63260527e+01,
        -1.49154683e+01, -3.29463670e+00, -1.99401505e+00],
       [-1.72788244e+01, -6.97875225e+00,  8.91533902e+00,
         4.22297440e+00,  1.76058842e+01, -3.19100184e+00],
       [-2.02469409e+01, -3.92841296e+00, -6.49228724e+00,
        -1.35089510e+01, -3.57059551e-01, -2.55677550e+00],
       [-4.32313608e+01, -3.23254815e+00,  6.54023868e+00,
         1.49659287e+01,  6.92031255e-01,  8.82154239e+00],
       [-9.01172901e+00, -3.83574728e+00,  1.79200579e+01,
        -6.14971224e+00,  2.07942876e+01,  1.59785473e+00],
       [-3.45583692e+01, -1.79103959e+01, -1.86498108e+01,
        -1.79341847e+00, -5.78262794e+00, -7.70354369e-01],
       [-2.87353024e+01, -6.73192718e+00, -8.94741722e+00,
        -9.50291516e-01,  1.29023365e+00, -1.28129089e+00],
       [-1.84359022e+01, -2.28919384e+01,  4.70939428e+00,
         1.47082555e+00, -5.76826605e+00,  2.53756565e+00],
       [ 2.52227648e+00, -6.08720782e+00,  2.09326570e+01,
         4.75062517e+00,  2.04364304e+00,  3.15280292e+00],
       [ 5.89672172e+00, -3.67975112e+00, -1.63156271e+00,
         7.60287610e-01,  8.66146260e+00, -4.52540867e+00],
       [-3.17248632e-01, -1.81787667e+01, -1.28775351e+01,
         7.29110607e-01, -2.10506576e+00, -3.64556816e+00],
       [-1.21204613e+01, -9.80941528e+00,  2.55587285e+00,
        -2.31642327e+00,  3.37512253e+00,  3.71244143e+00],
       [-2.48925887e+01, -6.30375935e-02, -1.31501962e+01,
        -4.54683176e+00,  9.73415271e-01, -1.52753576e+00]]),
'loadings':            x1        x2        x3        x4        x5  PCA_score
```

```
PC1  0.584892  0.317800  0.690811  0.045668  0.148361   0.235771
PC2  0.481138 -0.339428 -0.340384 -0.487029 -0.227244   0.498595
PC3  0.383696 -0.191943  0.124163 -0.220006 -0.353221  -0.792055
PC4  0.184029 -0.634601  0.011681  0.153899  0.724522  -0.121085
PC5  0.480578  0.444735 -0.621468  0.362585  0.175198  -0.151234
PC6  0.119850 -0.382677  0.071713  0.746433 -0.495942   0.175872}
```

In [24]:
```python
# Применение PCA
results = pca_manual(df3_1.iloc[:,1:].values)
# Используем первую компоненту (или взвешенную сумму нескольких, если требуется)
df3_1['PCA_Score'] = results['projected_data'][:, 0]  # Первая главная компонент

# Ранжирование стран
data_sorted = df3_1.sort_values(by='PCA_Score', ascending=False)

# Вывод результатов
print("Доля объясненной дисперсии (вклад каждой компоненты):")
print(results['explained_variance_ratio'])

print("\nФакторные нагрузки (вклад признаков в компоненты):")
print(results['loadings'])

print("\nРанжированные страны по удовлетворённости:")
data_sorted[['Страна', 'PCA_Score']]
```

```
Доля объясненной дисперсии (вклад каждой компоненты):
[0.53578359 0.17582261 0.13514881 0.07013181 0.06386609 0.01924709]

Факторные нагрузки (вклад признаков в компоненты):
           x1        x2        x3        x4        x5  PCA_score
PC1  0.584892  0.317800  0.690811  0.045668  0.148361   0.235771
PC2  0.481138 -0.339428 -0.340384 -0.487029 -0.227244   0.498595
PC3  0.383696 -0.191943  0.124163 -0.220006 -0.353221  -0.792055
PC4  0.184029 -0.634601  0.011681  0.153899  0.724522  -0.121085
PC5  0.480578  0.444735 -0.621468  0.362585  0.175198  -0.151234
PC6  0.119850 -0.382677  0.071713  0.746433 -0.495942   0.175872

Ранжированные страны по удовлетворённости:
```

Out[24]:

| № | Страна | PCA_Score |
|---|---|---|
| **35** | Парагвай | 29.049238 |
| **41** | Гондурас | 26.991538 |
| **1** | Израиль | 26.983236 |
| **34** | Боливия | 26.558810 |
| **33** | Таиланд | 26.294956 |
| **...** | ... | ... |
| **65** | Руанда | -31.830723 |
| **24** | Грузия | -33.447196 |
| **73** | Мали | -34.558369 |
| **63** | Танзания | -35.000729 |
| **71** | Сьерра Леон | -43.231361 |

80 rows × 2 columns

In [25]:
```
data_sorted[['Страна', 'PCA_Score']].to_excel('ranged_countries.xlsx')
```

# Задача 3.2

Провести факторный анализ социальноэкономического развития регионов Российской Федерации по следующим показателям:

- x1 – коэффициент Джини (индекс концентрации доходов), коэффициент;
- x2 - коэффициент фондов (соотношение доходов 10% наиболее и 10% наименее обеспеченного населения), коэффициент;
- x3 - доля населения с денежными доходами ниже величины прожиточного минимума, %;
- x4 - среднедушевые денежные доходы, тыс. руб.;
- x5 - дефицит дохода, %;
- x6 - число родившихся в расчете на 1000 населения за год, %0;
- x7 - число умерших в расчете на 1000 населения за год, %0;
- x8 - демографическая нагрузка на население трудоспособного возраста, на 1000 человек трудоспособного возраста приходится лиц нетрудоспособных возрастов на начало года, %0;
- x9 - уровень безработицы по методологии МОТ, %;
- x10 - валовый региональный продукт на душу населения, тыс. руб.;
- x11 - инвестиции в основной капитал на душу населения, тыс. руб.

Данные для анализа представлены в следующей таблице. Распределение показателей по вариантам выполнения задания:

- Вариант 1 - x1; x3; x4; x6; x8; x9; x10;
- Вариант 2 - x1; x3; x4; x7; x8; x9; x11;
- Вариант 3 - x2; x4; x5; x6; x8; x9; x10;
- Вариант 4 - x2; x4; x5; x7; x8; x9; x11.

В моем случае:

- Вариант 3 - x2; x4; x5; x6; x8; x9; x10;

## Возьмем данные из `Исходные данные для задачи 3.2.pdf`

In [26]:
```
text = '''Регион Показатели x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 Белгородская обла

# Уберем номера страниц, заменим запятые на точки, а из строки сделаем список
text_split = text.replace(',','.').replace(' 46 ','').replace(' 44 ','').replace

# В полученном списке преобразуем значения с плавающей точкой
table = []
for i in range(len(text_split)):
    try:
        table.append(float(text_split[i]))
    except:
        table.append(text_split[i])

# Зададим временные переменные
data = []
temp = []
k = 0
word = ''

# Пройдемся по всему списку
for i in range(len(table)):
    # Отсекаем слово показатели
    if table[i]=='Показатели':
        continue

    # Добавляем заголовки
    if i in list(range(13)):
        temp.append(table[i])
        continue
    if i == 13:
        data.append(temp)
        temp = []

    # Проверяем, строка это или нет
    if isinstance(table[i],str):
        # Если строка, то добавляем к слову
        word+= ' ' + table[i]
        continue

    elif isinstance(table[i],float):
        # Если не строка, то добавляем слово во временный список, а потом убирае
        if isinstance(table[i-1],str):
            temp.append(word)
```

```
            word = ''
            k+=1
        temp.append(table[i])
        k+=1

    # Так как в каждой строке 12 значений
    # в двумерный список будем добавлять каждые 12 значений
    if k!=0 and k%12==0:
        data.append(temp)
        temp = []


df3_2 = pd.DataFrame(np.array(data))
# Обозначим первую строку как заголовок, первый столбец за индекс и выполним обр
df3_2.columns = df3_2.iloc[0]
df3_2 = df3_2.drop(0).reset_index(drop=True)
df3_2.columns = ['Регион']+[f'x{i+1}' for i in range(11)]
df3_2
```

Out[26]:

| | Регион | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Белгородская область | 0.4 | 14.3 | 10.2 | 14.0 | 0.9 | 11.0 | 14.4 | 605.33 | 4.8 | 209.62 | 67.85 |
| **1** | Брянская область | 0.39 | 12.8 | 15.3 | 11.39 | 1.7 | 11.1 | 16.9 | 628.6 | 10.7 | 97.4 | 18.8 |
| **2** | Владимирская область | 0.35 | 10.1 | 20.4 | 10.47 | 2.8 | 10.8 | 18.4 | 624.83 | 8.8 | 122.01 | 32.41 |
| **3** | Воронежская область | 0.4 | 14.6 | 21.2 | 11.78 | 2.8 | 10.4 | 17.0 | 640.78 | 8.6 | 127.16 | 40.5 |
| **4** | Ивановская область | 0.35 | 10.0 | 23.4 | 9.15 | 3.5 | 10.6 | 18.6 | 630.14 | 10.8 | 79.98 | 22.73 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **74** | Амурская область | 0.37 | 11.5 | 23.1 | 13.12 | 3.4 | 13.2 | 14.6 | 539.97 | 8.7 | 157.76 | 74.74 |
| **75** | Магаданская область | 0.4 | 14.6 | 16.3 | 23.69 | 1.8 | 12.1 | 13.3 | 455.89 | 6.6 | 255.17 | 72.88 |
| **76** | Сахалинская область | 0.42 | 15.9 | 12.0 | 27.55 | 1.1 | 12.1 | 14.5 | 506.55 | 10.0 | 650.26 | 291.14 |
| **77** | Еврейская Автономная область | 0.38 | 12.0 | 22.5 | 12.97 | 3.2 | 13.2 | 14.8 | 542.15 | 8.5 | 143.93 | 43.06 |
| **78** | Чукотский Автономный округ | 0.41 | 15.3 | 9.3 | 39.18 | 0.8 | 14.2 | 13.0 | 434.73 | 4.4 | 615.31 | 176.28 |

79 rows × 12 columns

- Вариант 3 - x2; x4; x5; x6; x8; x9; x10;

In [27]:
```
df3_2_var_3 = df3_2[['Регион', 'x2','x4','x5','x6','x8','x9','x10']].copy()

for i in df3_2_var_3.columns[1:]:
    df3_2_var_3[i]=df3_2_var_3[i].transform(float)

df3_2_var_3
```

Out[27]:

| | Регион | x2 | x4 | x5 | x6 | x8 | x9 | x10 |
|---|---|---|---|---|---|---|---|---|
| **0** | Белгородская область | 14.3 | 14.00 | 0.9 | 11.0 | 605.33 | 4.8 | 209.62 |
| **1** | Брянская область | 12.8 | 11.39 | 1.7 | 11.1 | 628.60 | 10.7 | 97.40 |
| **2** | Владимирская область | 10.1 | 10.47 | 2.8 | 10.8 | 624.83 | 8.8 | 122.01 |
| **3** | Воронежская область | 14.6 | 11.78 | 2.8 | 10.4 | 640.78 | 8.6 | 127.16 |
| **4** | Ивановская область | 10.0 | 9.15 | 3.5 | 10.6 | 630.14 | 10.8 | 79.98 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **74** | Амурская область | 11.5 | 13.12 | 3.4 | 13.2 | 539.97 | 8.7 | 157.76 |
| **75** | Магаданская область | 14.6 | 23.69 | 1.8 | 12.1 | 455.89 | 6.6 | 255.17 |
| **76** | Сахалинская область | 15.9 | 27.55 | 1.1 | 12.1 | 506.55 | 10.0 | 650.26 |
| **77** | Еврейская Автономная область | 12.0 | 12.97 | 3.2 | 13.2 | 542.15 | 8.5 | 143.93 |
| **78** | Чукотский Автономный округ | 15.3 | 39.18 | 0.8 | 14.2 | 434.73 | 4.4 | 615.31 |

79 rows × 8 columns

In [28]:
```
# Применение PCA
results = pca_manual(df3_2_var_3.iloc[:,1:].values,7)

# Используем первую компоненту (или взвешенную сумму нескольких, если требуется)
df3_2_var_3['PCA_Score'] = results['projected_data'][:, 0]  # Первая главная ком

# Ранжирование стран
data_sorted = df3_2_var_3.sort_values(by='PCA_Score', ascending=False)

# Вывод результатов
print("Доля объясненной дисперсии (вклад каждой компоненты):")
print(results['explained_variance_ratio'])

print("\nФакторные нагрузки (вклад признаков в компоненты):")
print(results['loadings'])
data_sorted[['Регион', 'PCA_Score']].to_excel('ranged_regions.xlsx')
data_sorted[['Регион', 'PCA_Score']]
```

```
Доля объясненной дисперсии (вклад каждой компоненты):
[9.26088181e-01 7.18164959e-02 1.33235626e-03 4.08136189e-04
 1.80205529e-04 1.43967129e-04 3.06579631e-05]

Факторные нагрузки (вклад признаков в компоненты):
           x1        x2        x3        x4        x5        x6  PCA_score
PC1  0.015968  0.035834 -0.003201 -0.000293 -0.210680 -0.011864   0.976690
PC2  0.016506 -0.007660 -0.008392 -0.016211  0.977118 -0.015468   0.210563
PC3  0.073979  0.158361 -0.137684 -0.285273 -0.015870 -0.931859  -0.022299
PC4  0.592966  0.779150 -0.042090  0.129238  0.007726  0.146844  -0.034930
PC5  0.350346 -0.344860  0.053958  0.819433  0.000025 -0.289712   0.003834
PC6  0.720007 -0.493762 -0.025452 -0.471008 -0.022723  0.121525   0.002694
PC7  0.035186  0.061443  0.987741 -0.091298  0.005150 -0.104849   0.000217
```

Out[28]:

| | Регион | PCA_Score |
|---|---|---|
| **56** | Тюменская область | 750.018358 |
| **17** | г.Москва | 608.757108 |
| **76** | Сахалинская область | 465.659411 |
| **78** | Чукотский Автономный округ | 447.129065 |
| **70** | Республика Саха | 137.767004 |
| **...** | ... | ... |
| **60** | Республика Тыва | -116.367351 |
| **28** | Республика Адыгея | -117.202040 |
| **4** | Ивановская область | -118.135724 |
| **38** | Республика Северная Осетия - Алания | -118.824398 |
| **35** | Республика Ингушетия | -157.611623 |

79 rows × 2 columns