In [201]:

```
import pandas as pd
df=pd.read_excel("ip2countries.xlsx",index_col=0)
df
```

Out[201]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
country										
Algeria	10555.77295	103.84254	35.57219	1.09477	28.55218	18.90770	0.16408	74.34625	14.75335	66.94180
Argentina	21474.46315	120.81504	17.99867	2.00182	14.96615	41.03279	14.70895	75.16305	10.34445	90.68355
Australia	44901.79421	98.17638	20.51751	3.45412	21.40077	73.27461	106.56762	81.42902	5.49395	85.10010
Austria	51798.84394	100.13374	50.61335	2.47289	47.36891	68.53655	28.66159	80.31683	5.06380	58.32815
Bahrain	46856.86522	98.47988	81.96372	4.32123	65.43680	57.79351	77.77578	75.94505	1.05190	88.70820
Bangladesh	2991.50823	104.76766	15.84205	0.90378	21.88604	5.70977	21.00636	69.40680	4.09800	30.22440
Barbados	15592.12452	97.76974	43.37301	8.13230	46.79701	58.80189	58.55669	78.27870	10.05935	32.16005
Belgium	47629.32587	99.32256	76.70332	11.99939	74.46425	68.48175	67.15990	79.93902	7.43000	97.61175
Botswana	14548.50317	96.63532	48.50275	2.73895	45.44328	17.01316	16.88933	59.41175	18.29905	61.47060
Brazil	13833.70048	104.33491	13.00150	3.28786	12.96247	38.49962	49.29575	73.25665	9.15565	84.13355
Bulgaria	16841.76032	90.73149	52.07301	8.27127	57.74115	39.24428	14.89577	73.48122	10.44815	72.11510
Canada	44501.47786	99.48483	34.09799	3.28807	33.32779	77.79967	115.15012	80.89549	6.92770	80.68115
Chile	20349.48488	100.31893	34.81866	6.68223	30.89979	48.52325	100.26290	78.50625	8.40990	87.01780
China	8895.98231	100.38581	25.50290	3.06752	22.10584	29.20289	50.43575	74.25595	4.40130	48.33765
Colombia	11949.50170	98.07028	16.57906	3.76909	20.06917	33.27269	43.04574	75.24760	11.50890	77.68470
Costa Rica	15781.27866	89.77731	36.65046	5.72288	38.63074	39.63250	5.04494	78.76480	7.58605	70.57315
Croatia	23928.31794	96.19452	40.35030	3.60183	44.71456	48.63070	38.00500	76.25430	12.87615	55.16435
Cyprus	36635.98946	95.11678	61.06790	81.22226	62.26724	51.60125	42.18367	79.40770	7.79660	67.68235
Czech Republic	32919.67586	97.60616	66.47427	5.00239	63.15512	56.71221	18.86847	77.22488	6.05335	73.55735
Denmark	52140.02334	97.98371	50.93735	1.89211	44.94532	83.85115	59.09359	79.03061	5.48070	86.61015
Ecuador	10350.13714	96.53075	26.33764	1.11410	27.62234	26.58380	4.07578	74.97390	4.02780	62.44435
Egypt, Arab Rep.	9654.39285	114.77178	20.86568	2.92613	27.11286	23.28638	50.95438	70.27345	10.83420	42.90365
Eswatini	7345.38056	101.94790	53.55133	1.67447	58.69735	14.14583	6.92851	49.05600	25.42495	22.72765
Finland	45295.26712	99.99149	39.10556	3.48922	36.43508	78.45453	103.55751	79.91756	8.58795	83.90895
France	42720.94827	98.57713	28.57364	2.10249	28.89630	62.47824	80.60163	81.25378	8.98700	78.26545
Germany	47801.20053	99.46123	41.40828	2.57246	36.09995	72.58240	46.97452	79.78354	6.86595	76.53935
Ghana	3873.57457	112.50347	34.44416	5.00017	47.27808	13.17222	7.66171	60.58400	6.05695	50.35445
Greece	31962.34835	93.02833	26.00976	0.91478	32.60538	43.73078	38.32220	80.13768	15.58020	76.06385
Hong Kong SAR, China	49589.58807	105.94608	181.31227	28.66604	176.00903	67.60211	869.88774	82.95829	4.45610	100.00000
Hungary	25413.20018	93.76510	76.60953	9.59315	74.10149	54.35430	20.29853	74.11683	7.18480	68.31290
India	4284.96214	104.33095	19.74706	1.62885	22.91012	10.12222	71.70599	66.32010	5.55190	30.85675
Indonesia	8339.33591	97.49869	27.32565	1.26997	24.44618	13.81487	36.37014	68.88000	5.77065	49.25940
Ireland	60116.79090	98.39349	99.81759	20.49490	83.35580	61.63072	48.15344	80.02524	8.04470	61.37360
Israel	34744.83485	96.96515	34.37426	3.84501	34.05125	54.18217	69.72629	81.14805	8.38495	91.81780
Italy	42912.12450	98.36692	27.06483	1.20422	26.11301	47.85542	31.90581	81.75951	9.44660	68.56600
Jamaica	9670.00814	94.66233	34.94371	5.14915	52.83549	27.86422	59.46787	74.13410	12.21475	53.74040
Japan	38070.85497	101.76250	15.02240	0.27683	14.54583	73.09349	82.55889	82.79351	4.10560	88.08510
Jordan	10167.29093	96.37153	44.61357	7.56849	70.14418	31.93834	145.31713	73.24860	13.78015	84.85925
Kazakhstan	19819.59399	104.70469	44.12869	7.46793	34.94117	35.17436	18.34034	68.77698	6.84825	56.79440
Kenya	3395.00723	102.13649	20.41526	1.09682	30.87407	9.13291	26.56016	59.41500	2.78440	23.49480
Korea, Rep.	33333.12866	96.95655	41.19764	0.96456	38.23141	79.28725	73.92151	79.80439	3.60480	81.31850
Kuwait	60623.93756	96.85071	60.02706	0.41493	33.68180	52.62337	102.66606	74.27850	1.81745	99.94540
Lebanon	16020.21213	105.46697	26.92105	8.56679	51.66724	41.01840	21.75351	77.61225	7.17810	87.30400
Luxembourg	107343.23295	98.75192	180.80879	19.54466	150.14630	77.86680	134.35379	80.42476	4.72785	88.15010
Malaysia	21081.19761	99.88125	91.21397	3.18457	76.41910	56.22698	133.59502	74.39320	3.31120	69.97060
Malta	33224.37630	97.78066	137.58204	94.42365	132.66658	55.37244	43.66027	80.52098	6.03520	93.87385
Mauritius	16720.81421	96.10836	52.06777	2.67930	59.85678	30.62768	56.81164	73.15429	7.81720	41.63260
Mexico	18412.74838	99.00911	30.36761	2.75195	31.91886	33.45510	30.70233	75.00430	3.94305	77.63745
Montenegro	16360.30602	97.99434	39.97353	19.01050	64.85083	44.35391	83.04617	74.97575	22.06770	63.72335
Morocco	6102.90485	98.06995	32.80144	2.75574	41.87874	36.63863	64.80749	73.46975	10.05570	57.83720
Namibia	8754.39351	100.80553	42.55438	5.31004	54.31831	13.57242	11.56752	56.51925	20.81545	41.29590
Netherlands	51951.09648	99.81607	72.18789	22.54926	63.56597	81.84495	96.48962	80.32402	4.69270	85.79085

File failed to load: /extensions/MathMenu.js

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
country										
New Zealand	37856.35063	97.18383	29.77545	0.96449	28.66947	74.82313	35.98666	80.49980	5.13720	86.28240
Nigeria	4539.75402	112.32083	21.99824	1.60424	15.46396	10.05308	11.04043	50.47705	4.64150	43.02220
Norway	60693.09250	99.03583	40.65915	2.23321	29.48764	86.83106	55.20694	80.91207	3.63350	79.17670
Oman	32159.29893	94.40321	58.32552	2.64831	39.86930	40.23206	38.46857	75.34010	4.41100	76.54540
Pakistan	3941.11570	99.45464	12.37894	1.19560	18.00158	8.90224	25.18087	65.11595	1.67430	34.90765
Panama	22091.77462	99.66080	60.22187	8.38654	66.40131	33.50675	27.57635	76.73360	3.39875	65.04505
Papua New Guinea	3416.19120	100.33591	70.82334	1.38118	57.81150	4.17368	87.05935	61.85765	2.47225	13.08690
Peru	9722.02297	101.31424	24.45059	3.91049	22.43994	30.52209	40.02295	74.14760	4.05730	76.02095
Philippines	6130.02598	96.31479	33.88348	1.58969	37.28646	23.84051	57.88698	69.83475	3.37915	45.97925
Poland	23516.40608	97.14117	41.25680	3.36783	41.86943	51.75983	29.30771	76.05317	11.20195	60.92800
Portugal	31549.05416	98.62341	33.90843	4.03767	38.51033	49.06291	34.63950	79.25927	8.97575	60.20230
Qatar	96145.88207	93.61611	61.58784	2.26375	30.94415	55.90528	96.98865	78.92940	0.70580	98.08700
Romania	20436.29445	90.15083	31.86744	3.68427	37.89546	36.75650	9.91264	73.35622	6.52185	53.55880
Russian Federation	22734.89822	100.28322	31.03965	2.21054	21.56648	41.04719	49.98071	68.67468	6.69130	73.75705
Rwanda	1487.56928	96.19441	12.73537	2.08152	26.93313	8.21488	31.75331	60.84780	0.99300	16.81935
Saudi Arabia	45022.18167	97.06297	47.03912	2.21751	29.49957	43.15483	80.17610	73.86765	5.57785	81.96785
Serbia	13841.05164	94.47537	33.37592	7.81883	46.25681	43.90288	11.56508	74.07354	17.09405	54.70640
Singapore	76375.62082	100.05456	195.40733	19.77257	170.53138	68.36860	205.81964	81.04439	4.40990	100.00000
Slovak Republic	24865.58294	96.11602	79.62812	4.31179	79.50079	62.74533	4.65656	75.26927	13.29660	54.83590
Slovenia	32871.34276	95.24624	67.07456	2.25116	64.45912	58.28232	20.80310	78.99470	6.76780	52.58360
South Africa	12001.80126	102.07207	29.62289	1.56975	29.30004	27.21603	229.47632	58.14495	27.22165	61.91185
Spain	37659.34480	97.15997	29.24849	3.05490	29.77452	59.13339	76.97588	81.55268	15.86475	78.32765
Sri Lanka	9222.89662	94.88504	26.53651	1.27261	35.19891	12.60057	20.80207	75.09530	5.73425	18.30850
Sweden	48076.45828	98.89887	44.33769	3.71144	39.56083	84.60756	88.17765	81.28659	6.85325	85.37575
Switzerland	63693.94148	97.32003	60.04100	5.12084	50.48810	78.03125	218.18310	82.07841	4.15750	73.59060
Tanzania	2004.92910	109.02283	17.12525	3.22965	22.32128	5.66599	2.87113	58.22105	2.76080	27.96520
Thailand	14121.70167	97.60568	66.29415	2.73688	60.08747	26.34063	74.54546	73.91075	1.11435	42.08575
Tunisia	9450.83646	103.66350	45.53047	2.99353	51.92855	30.92277	16.38934	74.95665	14.68535	66.46745
Turkey	21197.02665	105.49394	23.42068	1.60620	26.25908	36.37218	26.04618	74.18125	9.97945	70.40440
Ukraine	11261.38363	112.51636	48.90944	3.70474	52.22391	25.72610	18.24248	69.69298	8.58480	68.37820
United Arab Emirates	74871.48430	98.37960	79.70390	2.83326	60.83210	65.03714	48.41295	76.20905	2.43215	83.76305
United Kingdom	43018.95237	99.43915	27.43059	4.41162	29.17910	76.13981	88.80302	79.94585	5.64245	81.13395
United States	55544.08838	98.76473	11.52989	1.81037	15.29782	70.95954	130.07946	78.04402	5.88330	80.71515
Vietnam	5119.07850	101.12838	75.70417	5.53746	79.24242	29.63627	25.30281	74.51845	2.00415	30.25160
Zambia	2879.39382	101.31198	33.33517	5.35703	35.92688	9.99825	15.09123	54.49690	11.65595	39.19000

In [202]:

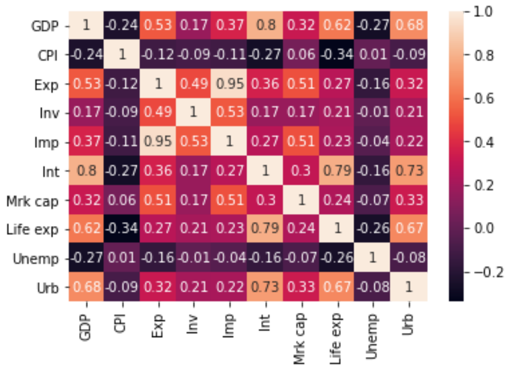
```
import plotly.io as pio
pio.renderers.default='svg'
```

In [203]:

```
#матрица парных корреляций (R)
import seaborn as sns
sns.heatmap(df.corr().round(2),annot=True)
```

Out[203]:

<AxesSubplot:>



File failed to load: /extensions/MathMenu.js

In [204]:

```
import numpy as np
from scipy import stats, linalg

def partial_corr(C):

    C = np.asarray(C)
    p = C.shape[1]
    P_corr = np.zeros((p, p), dtype=np.float)
    for i in range(p):
        P_corr[i, i] = 1
        for j in range(i+1, p):
            idx = np.ones(p, dtype=np.bool)
            idx[i] = False
            idx[j] = False
            beta_i = linalg.lstsq(C[:, idx], C[:, j])[0]
            beta_j = linalg.lstsq(C[:, idx], C[:, i])[0]

            res_j = C[:, j] - C[:, idx].dot( beta_i)
            res_i = C[:, i] - C[:, idx].dot(beta_j)

            corr = stats.pearsonr(res_i, res_j)[0]
            P_corr[i, j] = corr
            P_corr[j, i] = corr

    return P_corr
```

In [205]:

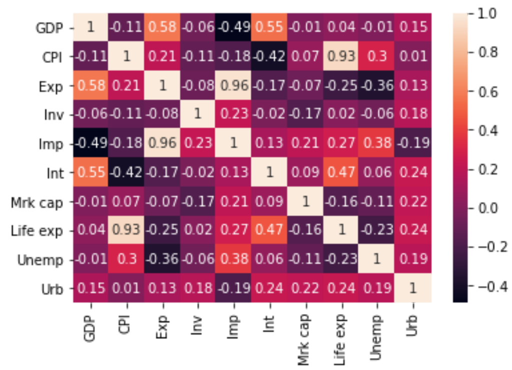
```
df_partial=pd.DataFrame(partial_corr(df), columns=df.columns, index=df.columns)
```

In [206]:

```
sns.heatmap(df_partial.round(2),annot=True)
```

Out[206]:

```
<AxesSubplot:>
```



In [207]:

```
df_corr=df.corr()
```

In [208]:

```
#df_corr.to_excel("корреляции_парные.xlsx", sheet_name="Sheet1")
#df_partial.to_excel("корреляции_частные.xlsx", sheet_name="Sheet2")
#from google.colab import files
#files.download("корреляции_парные.xlsx")
#files.download("корреляции_частные.xlsx")
```

In [209]:

```
alg_dop=np.zeros((df_corr.shape))
```

In [210]:

```
def minor(i,j):
  return np.delete(np.delete(df_corr.values,i,axis=0),j,axis=1)
for i in range(0,10):
  for j in range(0,10):
    alg_dop[i][j]=((-1)**(i+1+j+1))*np.linalg.det(minor(i,j))
```

In [211]:

```
pd.DataFrame(alg_dop)
```

Out[211]:

	0	1	2	3	4	5	6	7	8	9
0	0.00286	0.00014	-0.00364	0.00008	0.00287	-0.00152	0.00001	0.00002	0.00003	-0.00033
1	0.00014	0.00066	0.00002	0.00000	0.00004	-0.00001	-0.00012	0.00032	0.00009	-0.00022
2	-0.00364	0.00002	0.01416	0.00025	-0.01273	0.00084	0.00019	0.00136	0.00124	-0.00073
3	0.00008	0.00000	0.00025	0.00079	-0.00072	0.00004	0.00014	-0.00006	0.00003	-0.00017
4	0.00287	0.00004	-0.01273	-0.00072	0.01254	-0.00064	-0.00063	-0.00124	-0.00119	0.00087
5	-0.00152	-0.00001	0.00084	0.00004	-0.00064	0.00272	-0.00007	-0.00112	-0.00019	-0.00033
6	0.00001	-0.00012	0.00019	0.00014	-0.00063	-0.00007	0.00081	0.00002	0.00005	-0.00019
7	0.00002	0.00032	0.00136	-0.00006	-0.00124	-0.00112	0.00002	0.00192	0.00045	-0.00056
8	0.00003	0.00009	0.00124	0.00003	-0.00119	-0.00019	0.00005	0.00045	0.00075	-0.00027
9	-0.00033	-0.00022	-0.00073	-0.00017	0.00087	-0.00033	-0.00019	-0.00056	-0.00027	0.00146

In [212]:

```
multiple_corr=np.zeros((10,10))
for i in range(0,10):
  for j in range(0,10):
    multiple_corr[i][j]=(1-np.linalg.det(df_corr)/alg_dop[i][j])**0.5
```

```
<ipython-input-212-21e82057b472>:4: RuntimeWarning:

invalid value encountered in double_scalars
```

In [213]:

```
for i in range(0,10):
  for j in range(0,10):
    if i==j:
      print(multiple_corr[i][j])
```

```
0.9046224341583212
0.4539327847348471
0.9814561287970917
0.5821777116359086
0.9790312327688357
0.8994754514876798
0.600238337809388
0.8534474625848688
0.5543247806154716
0.8026976053439938
```

In [214]:

```
#матрица парных корреляций R
R=df.corr()
R
```

Out[214]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP	1.00000	-0.23634	0.53280	0.17432	0.36853	0.80464	0.32245	0.62236	-0.27053	0.67677
CPI	-0.23634	1.00000	-0.11507	-0.08640	-0.10732	-0.26655	0.06466	-0.34092	0.01101	-0.09135
Exp	0.53280	-0.11507	1.00000	0.48726	0.95477	0.35765	0.51154	0.27161	-0.16005	0.31735
Inv	0.17432	-0.08640	0.48726	1.00000	0.53459	0.17169	0.17492	0.20894	-0.01021	0.21121
Imp	0.36853	-0.10732	0.95477	0.53459	1.00000	0.26804	0.51457	0.22629	-0.03794	0.22192
Int	0.80464	-0.26655	0.35765	0.17169	0.26804	1.00000	0.30184	0.79255	-0.16065	0.72748
Mrk cap	0.32245	0.06466	0.51154	0.17492	0.51457	0.30184	1.00000	0.24278	-0.07418	0.33405
Life exp	0.62236	-0.34092	0.27161	0.20894	0.22629	0.79255	0.24278	1.00000	-0.26195	0.66801
Unemp	-0.27053	0.01101	-0.16005	-0.01021	-0.03794	-0.16065	-0.07418	-0.26195	1.00000	-0.08424
Urb	0.67677	-0.09135	0.31735	0.21121	0.22192	0.72748	0.33405	0.66801	-0.08424	1.00000

In [215]:

```
np.set_printoptions(suppress = True)
```

In [216]:

```
#собственные значения матрицы парных корреляций R
np.linalg.eigvals(R)
```

Out[216]:

```
array([4.19669282, 1.82080226, 1.06713464, 0.96557609, 0.74303274,
       0.5115694 , 0.01901481, 0.11817776, 0.25686832, 0.30113114])
```

In [217]:

```
sum(np.linalg.eigvals(R))
```

Out[217]:

```
10.000000000000007
```

In [218]:

```
#таблица собственных значений (eigenvalue)
df_eig_var=pd.DataFrame(pd.DataFrame(np.linalg.eigvals(R),columns=["eigenvalues"]).sort_values(by=["eigenvalues"], ascending=False).\
df_eig_var
```

Out[218]:

	eigenvalues
0	4.19669
1	1.82080
2	1.06713
3	0.96558
4	0.74303
5	0.51157
6	0.30113
7	0.25687
8	0.11818
9	0.01901

In [219]:

```
#percent of variance (доля объясненной дисперсии) и cumulative percentage (процент накопленной дисперсии)
summa=sum(np.linalg.eigvals(R))
df_eig_var["percent of variance"]=df_eig_var["eigenvalues"].apply(lambda x: x/summa*100)
df_eig_var["cumulative percentage"]=df_eig_var["percent of variance"].cumsum()
df_eig_var
```

Out[219]:

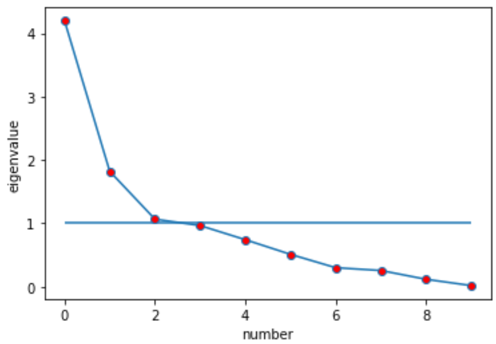
	eigenvalues	percent of variance	cumulative percentage
0	4.19669	41.96693	41.96693
1	1.82080	18.20802	60.17495
2	1.06713	10.67135	70.84630
3	0.96558	9.65576	80.50206
4	0.74303	7.43033	87.93239
5	0.51157	5.11569	93.04808
6	0.30113	3.01131	96.05939
7	0.25687	2.56868	98.62807
8	0.11818	1.18178	99.80985
9	0.01901	0.19015	100.00000

In [220]:

```
import matplotlib.pyplot as plt
plt.plot(df_eig_var["eigenvalues"], marker='o', markerfacecolor="red")
plt.xlabel("number")
plt.ylabel("eigenvalue")
plt.hlines(y=1, xmin=0, xmax=9)
```

Out[220]:

```
<matplotlib.collections.LineCollection at 0x20d66e503a0>
```



In [221]:

```
#метод главных компонент ДЛЯ ВСЕХ ФАКТОРОВ
from sklearn.preprocessing import StandardScaler
standardscaler=StandardScaler()
pd.set_option('display.float_format', lambda x: '%.5f' % x)
from sklearn.decomposition import PCA
pca=PCA(n_components=10)
X=df.values
X=standardscaler.fit_transform(X)
X_pca=pca.fit_transform(X)
X_pca.shape
```

Out[221]:

```
(87, 10)
```

In [222]:

```
#файл с коэффициентами и значениями МГК для всех факторов
#from google.colab import files
#pd.DataFrame(X_pca,index=df.index,columns=["f1","f2","f3","f4","f5","f6","f7","f8","f9","f10"]).to_excel("Главные компоненты на всех
#files.download("Главные компоненты на всех факторах.xlsx")
#pd.DataFrame(pca.components_,columns=["x1","x2","x3","x4","x5","x6","x7","x8","x9","x10"],index=["f1","f2","f3","f4","f5","f6","f7",
#files.download("Коэффициенты МГК на всех факторах.xlsx")
```

In [223]:

```
pd.DataFrame(pca.components_,columns=["x1","x2","x3","x4","x5","x6","x7","x8","x9","x10"],index=["f1","f2","f3","f4","f5","f6","f7",'
```

Out[223]:

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
f1	0.40984	-0.13899	0.36747	0.22218	0.32895	0.40196	0.26955	0.37026	-0.12459	0.36504
f2	-0.19789	0.18897	0.42616	0.35334	0.50258	-0.31699	0.25979	-0.33912	0.13022	-0.25720
f3	0.08193	0.75331	-0.01325	-0.31331	-0.09064	0.00536	0.37848	-0.07476	-0.38893	0.13828
f4	-0.01869	0.25217	-0.12540	-0.04602	-0.05356	0.14143	0.20888	-0.02105	0.86331	0.32553
f5	-0.06147	0.43993	-0.12712	0.70362	-0.11708	0.02588	-0.46044	0.10807	-0.08870	0.21334
f6	-0.43697	-0.19986	-0.33401	0.33827	-0.19431	-0.10011	0.62680	0.29017	-0.13230	0.02785
f7	0.41641	-0.24476	-0.08548	0.24936	-0.30300	-0.14586	0.18966	-0.66561	-0.09277	0.31065
f8	0.35677	0.12676	-0.15673	0.22910	-0.16808	0.43933	0.17205	-0.03823	0.10605	-0.71835
f9	0.50571	0.06670	0.00571	0.02707	-0.11633	-0.70033	0.04554	0.44334	0.14696	-0.12177
f10	0.18618	0.00017	-0.71262	-0.02435	0.66563	-0.04971	-0.02130	-0.07025	-0.06424	0.04321

In [224]:

```
pd.DataFrame(pca.explained_variance_ratio_).cumsum()
```

Out[224]:

	0
0	0.41967
1	0.60175
2	0.70846
3	0.80502
4	0.87932
5	0.93048
6	0.96059
7	0.98628
8	0.99810
9	1.00000

In [225]:

```
df_pca=pd.DataFrame(X_pca,index=df.index,columns=["f1","f2","f3","f4","f5","f6","f7","f8","f9","f10"])
df_pca
```

Out[225]:

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
country										
Algeria	-1.60561	0.04027	-0.01188	1.18863	0.41481	-0.19125	-0.51835	-0.62006	0.67255	-0.30441
Argentina	-1.21126	-0.54723	3.10952	1.90919	2.30034	-0.62819	-0.69672	-0.27858	0.31703	-0.10496
Australia	1.11754	-1.96586	0.40280	0.23404	0.03345	0.56446	0.22437	0.35458	-0.14591	0.08133
Austria	0.98937	-0.95323	0.22199	-0.49502	0.00567	-0.67928	-0.49468	0.87877	0.12595	0.07980
Bahrain	1.84847	-0.21676	0.57012	-0.87661	-0.15049	-0.55820	0.32098	-0.65016	-0.25799	-0.05477
Bangladesh	-2.84744	0.41523	0.80836	-1.00418	0.23879	0.37725	-0.37745	0.22049	0.56767	0.02118
Barbados	-0.36778	0.07697	-0.79662	-0.13913	-0.29774	0.35016	-1.10964	1.20899	-0.24977	-0.18109
Belgium	2.36252	-0.20988	0.00233	0.35717	0.40360	-0.53282	-0.06503	-0.51370	-0.22128	0.12936
Botswana	-1.87609	1.11375	-1.24814	1.30897	-0.65750	-0.73428	0.94737	-0.53867	-0.06689	-0.08401
Brazil	-1.08313	-0.89358	0.80794	0.88232	0.74833	0.44845	0.18632	-0.54372	-0.08103	-0.09625
Bulgaria	0.03310	0.00201	-1.77488	-0.07507	-0.47639	0.11613	0.15605	-0.79958	-0.26874	0.06243
Canada	1.31619	-1.47733	0.46316	0.44565	-0.06811	0.27972	-0.05830	0.53036	-0.28342	0.02658
Chile	0.27517	-0.73445	0.31516	0.63059	0.29581	0.73684	-0.04023	-0.53561	-0.05663	-0.17040
China	-1.44300	-0.29587	0.28988	-0.75172	0.05007	0.67611	-0.30314	0.08057	0.13303	-0.19869
Colombia	-0.99992	-0.83149	-0.42441	0.78296	0.14225	0.65595	0.09309	-0.63823	0.14346	-0.07784
Costa Rica	-0.09769	-0.92246	-1.68591	-0.54927	-0.40209	0.59305	-0.09093	-0.74917	-0.01844	-0.06012
Croatia	-0.30932	-0.27532	-1.00938	0.51507	-0.43801	-0.02999	-0.44433	0.32215	0.05452	-0.02799
Cyprus	2.19903	1.70201	-2.53885	-0.54769	3.56424	1.66237	1.03500	1.15326	0.30764	-0.05123
Czech Republic	0.93716	-0.22919	-0.32222	-0.39220	-0.02675	-0.52947	-0.35441	-0.32856	-0.27829	0.00875
Denmark	1.78248	-1.49066	0.20870	0.06223	-0.07418	-0.49970	0.10278	0.25286	-0.56624	0.04681
Ecuador	-1.16309	-0.69706	-0.31713	-0.91586	-0.04420	0.47724	-0.10868	-0.67321	0.10165	-0.03609
Egypt, Arab Rep.	-2.36252	0.74191	1.94872	0.92213	1.05809	-0.18896	-0.82812	0.66148	0.47930	-0.02391
Eswatini	-3.38450	2.74989	-1.21643	2.11606	-0.91041	-1.63798	0.61119	0.77691	-0.32003	-0.01533
Finland	1.35561	-1.33948	0.38840	0.74939	-0.00156	0.02806	-0.03122	0.44220	-0.32365	-0.00608
France	0.74300	-1.48319	0.05642	0.56659	-0.07488	0.20972	-0.08308	0.27301	0.22441	0.03790
Germany	1.07374	-1.39940	0.20257	0.16541	0.10303	-0.35081	-0.11285	0.44515	-0.13281	-0.00585
Ghana	-2.53902	1.52253	1.80387	-0.07885	1.06418	-0.66743	-0.00419	-0.19646	-0.20763	0.28165
Greece	-0.00817	-1.20468	-1.44954	1.15379	-0.56500	0.21615	-0.08511	-0.29395	0.57336	0.05903
Hong Kong SAR, China	7.17093	5.35238	3.55791	1.30374	-2.58940	3.22372	0.05146	0.15273	0.04778	0.02923
Hungary	0.90041	0.39118	-1.14619	-0.54536	-0.30507	-0.41745	-0.16812	-0.40660	-0.58146	-0.02287
India	-2.70375	0.71111	0.85008	-0.65338	-0.07833	0.49205	-0.01438	0.41705	0.33450	-0.03624
Indonesia	-1.94809	0.05675	-0.20752	-0.77107	-0.35046	0.46203	0.29676	-0.34991	0.25500	-0.11485
Ireland	2.30976	0.79001	-0.67529	-0.35909	0.32245	-0.96682	-0.30679	0.71091	0.47109	-0.12565
Israel	0.84747	-1.35241	-0.18236	0.48849	0.04120	0.38226	0.05582	-0.56619	0.15326	0.01614
Italy	0.14877	-1.40224	-0.23658	0.30274	-0.00904	-0.02668	-0.24682	0.24122	0.75896	0.02359
Jamaica	-0.89814	0.34684	-1.21026	0.24352	-0.61918	0.51649	-0.33051	-0.26159	0.18869	0.20558
Japan	0.78590	-2.22843	1.02237	0.22241	0.43038	0.51699	-0.15213	0.16228	-0.21918	0.00738
Jordan	0.15187	0.75614	-0.64222	1.19150	-0.57813	0.78999	0.08099	-1.08400	-0.10726	0.41166
Kazakhstan	-1.10854	0.36442	0.62961	-0.13832	0.62532	-0.35873	0.06152	0.13837	-0.09310	-0.18498
Kenya	-3.11234	0.98762	0.56037	-1.42563	-0.24164	0.03734	0.45087	0.41745	-0.17685	0.21456
Korea, Rep.	1.21707	-1.53150	0.18436	-0.30920	-0.18066	0.21704	-0.18469	0.05903	-0.82299	-0.04600
Kuwait	1.57224	-1.21151	0.70106	-0.49805	-0.37647	-0.33487	1.34117	-0.66115	0.18176	-0.14269
Lebanon	-0.13100	-0.30217	0.75824	0.46674	1.17132	0.13986	-0.52535	-0.83781	-0.06120	0.47124
Luxembourg	5.84632	2.05806	0.09372	-0.65163	-0.44831	-2.56004	0.15549	0.23278	0.64269	0.10884
Malaysia	1.20557	0.85679	0.61355	-0.63317	-0.60183	-0.06494	-0.44925	-0.41331	-0.72735	-0.28126
Malta	4.48162	3.85232	-2.40951	-0.73305	4.23579	0.86773	0.47370	-0.19595	-0.21721	-0.08202
Mauritius	-0.69265	0.63688	-0.69164	-0.64895	-0.77710	0.07264	-0.45875	0.09894	0.14372	0.10237
Mexico	-0.50719	-0.72307	0.23657	-0.51259	0.23707	0.35465	0.12766	-0.82383	0.02294	0.05156
Montenegro	-0.11842	0.95641	-1.62844	2.23682	0.15953	0.34609	-0.40585	0.29648	0.14205	0.24990
Morocco	-0.98312	0.05300	-0.41271	0.22633	-0.33483	0.51520	-0.38164	-0.20250	-0.20147	0.00193
Namibia	-2.64184	1.92976	-1.04298	1.60249	-0.42066	-0.93675	0.53855	0.15477	-0.05780	0.14674
Netherlands	2.63519	-0.24580	0.12115	-0.10624	0.85472	-0.06361	0.12433	0.47575	-0.43651	-0.02953

Country	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
New Zealand	0.96740	-1.95674	0.03277	-0.02628	0.11188	0.06294	-0.04560	-0.04624	-0.50503	0.02127
Nigeria	-3.68925	1.29076	2.12835	-0.30223	0.82779	-0.75673	1.13962	0.20004	-0.57201	-0.04993
Norway	1.69182	-1.99946	0.49603	-0.23966	0.11513	-0.43253	0.13657	0.81006	-0.28852	-0.02473
Oman	0.39019	-0.60512	-0.44202	-0.75162	-0.38406	-0.04628	0.37836	-0.70014	0.08861	-0.26457
Pakistan	-2.76379	0.15069	0.29752	-1.53753	-0.17145	0.54937	0.42501	0.03107	0.09343	0.08554
Panama	0.21079	0.41408	0.03144	-0.96426	0.25497	-0.07457	-0.48224	-0.58568	0.15354	0.18114
Papua New Guinea	-2.19318	2.24209	0.34961	-1.86643	-1.02741	-0.05548	-0.06464	0.33499	0.07807	-0.28043
Peru	-0.97988	-0.63444	0.57529	-0.35756	0.50357	0.62029	0.06900	-0.78815	-0.05329	-0.10369
Philippines	-1.45726	0.19200	-0.20857	-1.21336	-0.57265	0.59961	0.06025	-0.22473	-0.15043	0.00090
Poland	-0.20933	-0.44660	-0.72644	0.37853	-0.23236	-0.08266	-0.38222	0.16694	-0.12327	-0.08399
Portugal	-0.03972	-0.73263	-0.33018	0.08485	-0.00309	0.03202	-0.46106	0.31946	0.31005	0.06269
Qatar	2.60913	-1.90610	0.32202	-0.91185	-0.55907	-0.71657	1.78510	-0.05267	1.12435	0.02446
Romania	-0.67813	-0.60420	-1.52071	-0.93350	-0.72198	0.32708	0.23756	-0.14586	-0.06691	0.10074
Russian Federation	-0.82926	-0.55077	0.39565	0.06358	0.08836	0.03790	0.68109	-0.27871	-0.31680	-0.15738
Rwanda	-3.09731	0.64539	-0.24213	-2.08166	-0.69756	0.56325	0.60386	0.48231	-0.18120	0.26501
Saudi Arabia	0.47526	-0.84816	0.17129	-0.18444	-0.30866	-0.06769	0.86875	-0.34162	0.30428	-0.15396
Serbia	-0.81047	0.00462	-1.79895	1.05343	-0.30782	0.06504	-0.38450	0.18307	-0.07428	0.04145
Singapore	5.88011	3.05985	0.46516	-0.43281	-0.58393	-1.71561	-0.41490	-0.88053	0.15195	0.01927
Slovak Republic	0.61843	0.57651	-1.30275	0.38567	-0.54405	-0.95951	-0.93884	0.21129	-0.54476	-0.08856
Slovenia	0.72342	-0.18698	-0.81065	-0.69538	-0.58372	-0.49053	-0.78366	0.29382	-0.12619	-0.04146
South Africa	-1.97443	1.49646	-0.19102	3.67520	-1.25523	0.40019	1.12970	0.45328	-0.03939	-0.23287
Spain	0.50292	-1.24883	-0.72461	1.59512	-0.23803	0.20830	-0.23658	0.22744	0.39783	-0.08028
Sri Lanka	-2.03497	0.18794	-0.94228	-1.44984	-0.76442	0.59039	-0.68041	0.49242	0.76688	0.03399
Sweden	1.73103	-1.53020	0.28647	0.40597	0.01815	-0.07738	-0.10339	0.42351	-0.45741	-0.02110
Switzerland	2.52220	-0.84181	0.65405	-0.15559	-0.86691	0.42472	0.24833	0.96560	0.14657	0.01279
Tanzania	-3.52921	1.11360	1.50610	-1.06678	0.65098	-0.24071	0.37194	0.41402	-0.08180	0.11114
Thailand	-0.44969	0.79982	0.07254	-1.71072	-0.65011	0.25707	-0.49722	-0.17741	0.09895	-0.12951
Tunisia	-0.94758	0.46060	-0.09970	1.18510	0.31588	-0.28776	-0.86119	-0.51533	0.23426	-0.04222
Turkey	-1.05533	-0.49776	0.75427	0.74017	0.61994	-0.11637	-0.24533	-0.15677	0.26102	0.01075
Ukraine	-1.27165	0.97829	1.72545	0.62109	1.13452	-0.70072	-0.61806	-0.52683	0.06480	0.04658
United Arab Emirates	2.22152	-0.71535	0.46164	-0.75013	-0.21186	-1.35044	0.66280	0.09574	0.24977	0.09313
United Kingdom	1.08131	-1.62998	0.44545	0.20460	0.16569	0.26703	0.06721	0.46336	-0.35640	0.09356
United States	0.88565	-1.98950	0.65651	0.34629	-0.18488	0.42944	0.73984	0.75834	0.04695	0.24471
Vietnam	-0.62747	1.48359	0.09909	-1.72452	-0.17215	-0.19027	-1.35989	0.04776	-0.12697	-0.01953
Zambia	-3.03614	1.52830	-0.23452	0.15287	-0.16744	-0.44034	0.95001	0.05011	-0.36856	0.01462

In [226]:

```
#вычислим Махаланобиса
import pandas as pd
import scipy as sp
import numpy as np
from scipy.stats import chi2
def mahalanobis(x=None, data=None, cov=None):
    """Compute the Mahalanobis Distance between each row of x and the data
    x    : vector or matrix of data with, say, p columns.
    data : ndarray of the distribution from which Mahalanobis distance of each observation of x is to be computed.
    cov  : covariance matrix (p x p) of the distribution. If None, will be computed from data.
    """
    x_minus_mu = x - np.mean(data)
    if not cov:
        cov = np.cov(data.values.T)
    inv_covmat = sp.linalg.inv(cov)
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()
df_pca['mahala'] = mahalanobis(x=df_pca, data=df_pca)
df_pca['p_value'] = 1 - chi2.cdf(df_pca['mahala'], 2)
countries=df_pca.loc[df_pca.p_value < 0.01].index
df_pca.loc[df_pca.p_value < 0.01]
```

Out[226]:

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	mahala	p_value
country												
Algeria	-1.60561	0.04027	-0.01188	1.18863	0.41481	-0.19125	-0.51835	-0.62006	0.67255	-0.30441	13.31665	0.00128
Argentina	-1.21126	-0.54723	3.10952	1.90919	2.30034	-0.62819	-0.69672	-0.27858	0.31703	-0.10496	24.30410	0.00001
Barbados	-0.36778	0.07697	-0.79662	-0.13913	-0.29774	0.35016	-1.10964	1.20899	-0.24977	-0.18109	12.89098	0.00159
Botswana	-1.87609	1.11375	-1.24814	1.30897	-0.65750	-0.73428	0.94737	-0.53867	-0.06689	-0.08401	10.78377	0.00455
Cyprus	2.19903	1.70201	-2.53885	-0.54769	3.56424	1.66237	1.03500	1.15326	0.30764	-0.05123	40.79303	0.00000
Egypt, Arab Rep.	-2.36252	0.74191	1.94872	0.92213	1.05809	-0.18896	-0.82812	0.66148	0.47930	-0.02391	13.44645	0.00120
Eswatini	-3.38450	2.74989	-1.21643	2.11606	-0.91041	-1.63798	0.61119	0.77691	-0.32003	-0.01533	23.46309	0.00001
Ghana	-2.53902	1.52253	1.80387	-0.07885	1.06418	-0.66743	-0.00419	-0.19646	-0.20763	0.28165	12.79784	0.00166
Hong Kong SAR, China	7.17093	5.35238	3.55791	1.30374	-2.58940	3.22372	0.05146	0.15273	0.04778	0.02923	70.29444	0.00000
Jordan	0.15187	0.75614	-0.64222	1.19150	-0.57813	0.78999	0.08099	-1.08400	-0.10726	0.41166	17.25145	0.00018
Kenya	-3.11234	0.98762	0.56037	-1.42563	-0.24164	0.03734	0.45087	0.41745	-0.17685	0.21456	9.25590	0.00977
Kuwait	1.57224	-1.21151	0.70106	-0.49805	-0.37647	-0.33487	1.34117	-0.66115	0.18176	-0.14269	11.41518	0.00332
Lebanon	-0.13100	-0.30217	0.75824	0.46674	1.17132	0.13986	-0.52535	-0.83781	-0.06120	0.47124	17.85538	0.00013
Luxembourg	5.84632	2.05806	0.09372	-0.65163	-0.44831	-2.56004	0.15549	0.23278	0.64269	0.10884	28.08313	0.00000
Malaysia	1.20557	0.85679	0.61355	-0.63317	-0.60183	-0.06494	-0.44925	-0.41331	-0.72735	-0.28126	11.84751	0.00268
Malta	4.48162	3.85232	-2.40951	-0.73305	4.23579	0.86773	0.47370	-0.19595	-0.21721	-0.08202	45.66874	0.00000
Montenegro	-0.11842	0.95641	-1.62844	2.23682	0.15953	0.34609	-0.40585	0.29648	0.14205	0.24990	12.63818	0.00180
Namibia	-2.64184	1.92976	-1.04298	1.60249	-0.42066	-0.93675	0.53855	0.15477	-0.05780	0.14674	11.42487	0.00330
Nigeria	-3.68925	1.29076	2.12835	-0.30223	0.82779	-0.75673	1.13962	0.20004	-0.57201	-0.04993	17.70186	0.00014
Papua New Guinea	-2.19318	2.24209	0.34961	-1.86643	-1.02741	-0.05548	-0.06464	0.33499	0.07807	-0.28043	13.53661	0.00115
Qatar	2.60913	-1.90610	0.32202	-0.91185	-0.55907	-0.71657	1.78510	-0.05267	1.12435	0.02446	27.00762	0.00000
Rwanda	-3.09731	0.64539	-0.24213	-2.08166	-0.69756	0.56325	0.60386	0.48231	-0.18120	0.26501	14.25446	0.00080
Singapore	5.88011	3.05985	0.46516	-0.43281	-0.58393	-1.71561	-0.41490	-0.88053	0.15195	0.01927	23.52150	0.00001
Slovak Republic	0.61843	0.57651	-1.30275	0.38567	-0.54405	-0.95951	-0.93884	0.21129	-0.54476	-0.08856	10.12288	0.00634
South Africa	-1.97443	1.49646	-0.19102	3.67520	-1.25523	0.40019	1.12970	0.45328	-0.03939	-0.23287	26.21329	0.00000
Sri Lanka	-2.03497	0.18794	-0.94228	-1.44984	-0.76442	0.59039	-0.68041	0.49242	0.76688	0.03399	12.85205	0.00162
Tanzania	-3.52921	1.11360	1.50610	-1.06678	0.65098	-0.24071	0.37194	0.41402	-0.08180	0.11114	9.36087	0.00927
United States	0.88565	-1.98950	0.65651	0.34629	-0.18488	0.42944	0.73984	0.75834	0.04695	0.24471	10.39877	0.00552
Vietnam	-0.62747	1.48359	0.09909	-1.72452	-0.17215	-0.19027	-1.35989	0.04776	-0.12697	-0.01953	10.68474	0.00478

In [227]:

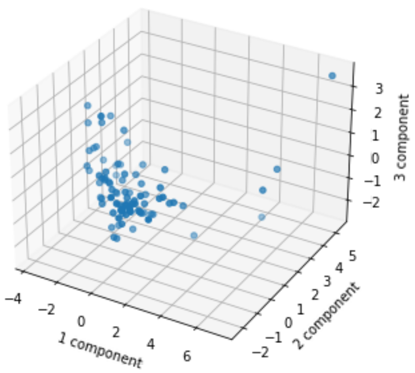
```
df_pca.loc[df_pca.p_value < 0.000005].sort_values(by=["mahala"],ascending=False)
```

Out[227]:

country	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	mahala	p_value
Hong Kong SAR, China	7.17093	5.35238	3.55791	1.30374	-2.58940	3.22372	0.05146	0.15273	0.04778	0.02923	70.29444	0.00000
Malta	4.48162	3.85232	-2.40951	-0.73305	4.23579	0.86773	0.47370	-0.19595	-0.21721	-0.08202	45.66874	0.00000
Cyprus	2.19903	1.70201	-2.53885	-0.54769	3.56424	1.66237	1.03500	1.15326	0.30764	-0.05123	40.79303	0.00000
Luxembourg	5.84632	2.05806	0.09372	-0.65163	-0.44831	-2.56004	0.15549	0.23278	0.64269	0.10884	28.08313	0.00000
Qatar	2.60913	-1.90610	0.32202	-0.91185	-0.55907	-0.71657	1.78510	-0.05267	1.12435	0.02446	27.00762	0.00000
South Africa	-1.97443	1.49646	-0.19102	3.67520	-1.25523	0.40019	1.12970	0.45328	-0.03939	-0.23287	26.21329	0.00000

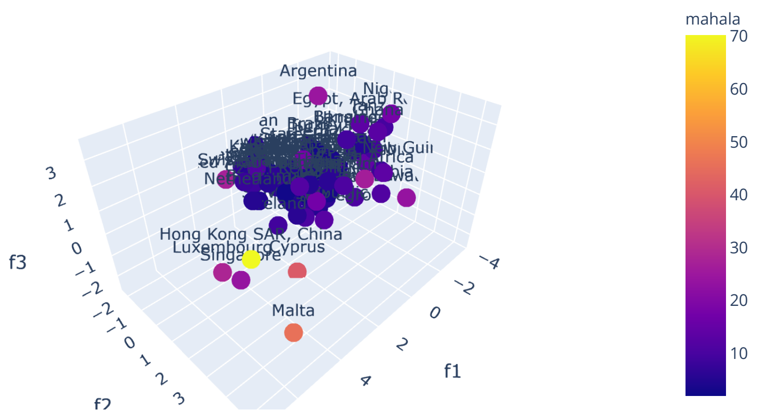
In [228]:

```
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
import random


fig = pyplot.figure()
ax = Axes3D(fig)

ax.scatter(df_pca["f1"],df_pca["f2"],df_pca["f3"])
ax.set_xlabel("1 component")
ax.set_ylabel("2 component")
ax.set_zlabel("3 component")

pyplot.show()
```



In [229]:

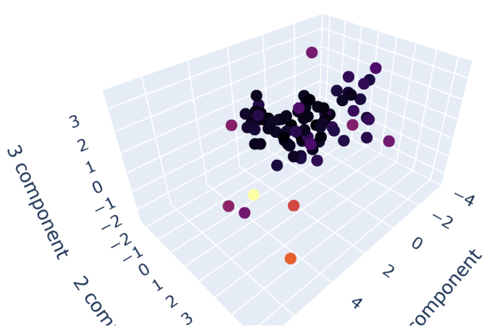
```
import plotly.express as px
fig = px.scatter_3d(df_pca, x='f1', y='f2', z='f3',color='mahala',text=df_pca.index,size_max=10)
fig.show()
```



File failed to load: /extensions/MathMenu.js

In [230]:

```
outlying_observations=df_pca.loc[df_pca.p_value < 0.000005]
import plotly.graph_objects as go

fig = go.Figure()

fig.add_trace(go.Scatter3d(x=df_pca['f1'], y=df_pca['f2'], z=df_pca['f3'], mode='markers',text=df_pca.index,marker=dict(size=5,color=
fig.update_layout(scene = dict(
                    xaxis_title='1 component',
                    yaxis_title='2 component',
                    zaxis_title='3 component')
                    )
fig.show()
```



Сначала построим все 9 компонент, хотя видно, что первые 3 компоненты объясняют более 91% первоначальной изменчивости

In [231]:

```
X=df.drop(columns=["Exp"])
import numpy as np
from sklearn.preprocessing import StandardScaler
standardscaler=StandardScaler()
previous_columns=X.columns
previous_rows=X.index
X=standardscaler.fit_transform(X.values)
X=pd.DataFrame(X,columns=previous_columns,index=previous_rows)
pca=PCA(n_components=9)
X_pca=pca.fit_transform(X)
X_pca=pd.DataFrame(X_pca,columns=["f1","f2","f3","f4","f5","f6","f7","f8","f9"],index=X.index)
weights=pd.DataFrame(pca.components_,columns=["x1","x2","x3","x4","x5","x6","x7","x8","x9"],index=["f1","f2","f3","f4","f5","f6","f7'
weights
```

Out[231]:

	x1	x2	x3	x4	x5	x6	x7	x8	x9
f1	0.44651	-0.16336	0.19864	0.27157	0.46111	0.25756	0.43289	-0.13817	0.41868
f2	-0.12924	0.21748	0.51140	0.59277	-0.21022	0.41617	-0.23769	0.18522	-0.12204
f3	0.08464	0.74360	-0.32754	-0.10043	0.00470	0.36446	-0.07288	-0.40874	0.13150
f4	0.01001	0.22371	-0.22270	-0.14059	0.15879	0.13940	-0.02467	0.85578	0.32076
f5	-0.02158	0.47737	0.58475	-0.17538	0.05347	-0.57252	0.08709	-0.00767	0.24633
f6	0.59954	0.11139	-0.23841	0.41140	0.11857	-0.42284	-0.42307	0.06459	-0.16141
f7	0.28075	-0.26962	0.23030	-0.37036	-0.22352	0.19249	-0.60328	-0.10688	0.44260
f8	0.28417	0.07476	0.29943	-0.44262	0.39908	0.25019	-0.06059	0.10059	-0.62645
f9	0.50712	0.06674	0.02596	-0.10928	-0.70129	0.04448	0.44272	0.14630	-0.12070

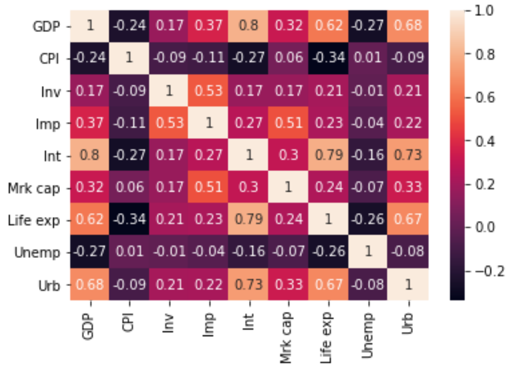
File failed to load: /extensions/MathMenu.js

In [232]:

```
df_dop=df.drop(columns=["Exp"])
sns.heatmap(df_dop.corr().round(2),annot=True)
```

Out[232]:

<AxesSubplot:>



In [233]:

```
#таблица собственных значений (eigenvalue)
R_dop=df_dop.corr()
df_eig_var_dop=pd.DataFrame(pd.DataFrame(np.linalg.eigvals(R_dop),columns=["eigenvalues"]).sort_values(by=["eigenvalues"], ascending=
df_eig_var_dop
```

Out[233]:

	eigenvalues
0	3.72675
1	1.45941
2	1.06686
3	0.94269
4	0.72737
5	0.42086
6	0.29686
7	0.24103
8	0.11817

In [234]:

```
#percent of variance (доля объясненной дисперсии) и cumulative percentage (процент накопленной дисперсии)
summa=sum(np.linalg.eigvals(R_dop))
df_eig_var_dop["percent of variance"]=df_eig_var_dop["eigenvalues"].apply(lambda x: x/summa*100)
df_eig_var_dop["cumulative percentage"]=df_eig_var_dop["percent of variance"].cumsum()
df_eig_var_dop
```

Out[234]:

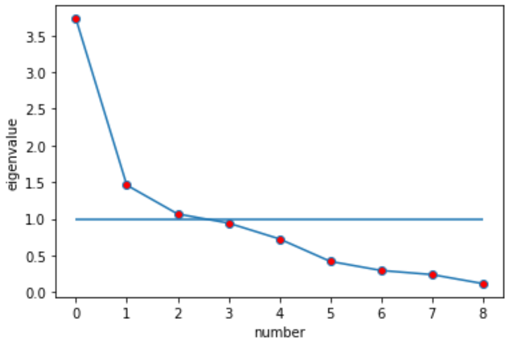
	eigenvalues	percent of variance	cumulative percentage
0	3.72675	41.40838	41.40838
1	1.45941	16.21568	57.62406
2	1.06686	11.85403	69.47809
3	0.94269	10.47432	79.95241
4	0.72737	8.08189	88.03430
5	0.42086	4.67623	92.71053
6	0.29686	3.29839	96.00893
7	0.24103	2.67807	98.68700
8	0.11817	1.31300	100.00000

In [235]:

```
import matplotlib.pyplot as plt
plt.plot(df_eig_var_dop["eigenvalues"], marker='o', markerfacecolor="red")
plt.xlabel("number")
plt.ylabel("eigenvalue")
plt.hlines(y=1, xmin=0, xmax=8)
```

Out[235]:

```
<matplotlib.collections.LineCollection at 0x20d6804b850>
```



In [236]:

```
from sklearn.preprocessing import StandardScaler
standardscaler=StandardScaler()
y=df["Exp"].values.reshape(-1,1)
y_scaled=standardscaler.fit_transform(y)
import statsmodels.api as sm
X = X_pca
X_sm=sm.add_constant(X)
y_sm=y_scaled
model=sm.OLS(y_sm,X_sm)
results = model.fit()
print(results.summary(alpha=0.1))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.963
Model:                            OLS   Adj. R-squared:                  0.959
Method:                 Least Squares   F-statistic:                     224.3
Date:                Wed, 17 Feb 2021   Prob (F-statistic):           1.88e-51
Time:                        22:24:21   Log-Likelihood:                 20.267
No. Observations:                  87   AIC:                            -20.53
Df Residuals:                      77   BIC:                             4.125
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.05      0.95]
------------------------------------------------------------------------------
const       4.163e-17      0.022   1.91e-15      1.000      -0.036      0.036
f1             0.3172      0.011     28.029      0.000       0.298      0.336
f2             0.4975      0.018     27.512      0.000       0.467      0.528
f3            -0.0192      0.021     -0.908      0.367      -0.054      0.016
f4            -0.1877      0.022     -8.342      0.000      -0.225     -0.150
f5            -0.1645      0.026     -6.421      0.000      -0.207     -0.122
f6             0.5534      0.034     16.433      0.000       0.497      0.609
f7            -0.1635      0.040     -4.079      0.000      -0.230     -0.097
f8            -0.3922      0.044     -8.815      0.000      -0.466     -0.318
f9             0.0109      0.064      0.172      0.864      -0.095      0.117
==============================================================================
Omnibus:                        9.114   Durbin-Watson:                   2.102
Prob(Omnibus):                  0.010   Jarque-Bera (JB):               10.846
Skew:                          -0.526   Prob(JB):                      0.00441
Kurtosis:                       4.373   Cond. No.                         5.62
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [237]:

```
X_sm=X_sm.drop(columns=["f3","f9"])
y_sm=y_scaled
model=sm.OLS(y_sm,X_sm)
results = model.fit()
print(results.summary(alpha=0.1))
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.963
Model:                            OLS   Adj. R-squared:                  0.960
Method:                 Least Squares   F-statistic:                     292.5
Date:                Wed, 17 Feb 2021   Prob (F-statistic):           9.79e-54
Time:                        22:24:21   Log-Likelihood:                 19.788
No. Observations:                  87   AIC:                            -23.58
Df Residuals:                      79   BIC:                            -3.848
Df Model:                           7                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|       [0.05       0.95]
------------------------------------------------------------------------------
const       4.163e-17      0.022   1.92e-15      1.000      -0.036       0.036
f1             0.3172      0.011     28.235      0.000       0.298       0.336
f2             0.4975      0.018     27.714      0.000       0.468       0.527
f4            -0.1877      0.022     -8.403      0.000      -0.225      -0.151
f5            -0.1645      0.025     -6.468      0.000      -0.207      -0.122
f6             0.5534      0.033     16.554      0.000       0.498       0.609
f7            -0.1635      0.040     -4.109      0.000      -0.230      -0.097
f8            -0.3922      0.044     -8.879      0.000      -0.466      -0.319
==============================================================================
Omnibus:                        9.429   Durbin-Watson:                   2.065
Prob(Omnibus):                  0.009   Jarque-Bera (JB):               11.195
Skew:                          -0.547   Prob(JB):                      0.00371
Kurtosis:                       4.376   Cond. No.                         3.93
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [238]:

```
outlying_observations
```

Out[238]:

country	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	mahala	p_value
Cyprus	2.19903	1.70201	-2.53885	-0.54769	3.56424	1.66237	1.03500	1.15326	0.30764	-0.05123	40.79303	0.00000
Hong Kong SAR, China	7.17093	5.35238	3.55791	1.30374	-2.58940	3.22372	0.05146	0.15273	0.04778	0.02923	70.29444	0.00000
Luxembourg	5.84632	2.05806	0.09372	-0.65163	-0.44831	-2.56004	0.15549	0.23278	0.64269	0.10884	28.08313	0.00000
Malta	4.48162	3.85232	-2.40951	-0.73305	4.23579	0.86773	0.47370	-0.19595	-0.21721	-0.08202	45.66874	0.00000
Qatar	2.60913	-1.90610	0.32202	-0.91185	-0.55907	-0.71657	1.78510	-0.05267	1.12435	0.02446	27.00762	0.00000
South Africa	-1.97443	1.49646	-0.19102	3.67520	-1.25523	0.40019	1.12970	0.45328	-0.03939	-0.23287	26.21329	0.00000

In [239]:

```
#пересчет МГК без выбросов
dropped_indexes=["Hong Kong SAR, China","South Africa"]
X=df.drop(columns=["Exp"])
X=df.drop(index=dropped_indexes)
remained_indexes=X.index
pca=PCA(n_components=9)
X=standardscaler.fit_transform(X)
X_pca=pca.fit_transform(X)
X_pca=pd.DataFrame(X_pca,columns=["f1","f2","f3","f4","f5","f6","f7","f8","f9"],index=remained_indexes)
X=X_pca
y=df["Exp"].drop(index=dropped_indexes).values.reshape(-1,1)
y_scaled=standardscaler.fit_transform(y)
X_sm=sm.add_constant(X)
y_sm=y_scaled
model=sm.OLS(y_sm,X_sm)
results = model.fit()
print(results.summary(alpha=0.1))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.989
Model:                            OLS   Adj. R-squared:                  0.987
Method:                 Least Squares   F-statistic:                     723.0
Date:                Wed, 17 Feb 2021   Prob (F-statistic):           4.38e-69
Time:                        22:24:21   Log-Likelihood:                 69.560
No. Observations:                  85   AIC:                            -119.1
Df Residuals:                      75   BIC:                            -94.69
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.05       0.95]
------------------------------------------------------------------------------
const       1.249e-16      0.012   1.01e-14      1.000      -0.021       0.021
f1             0.3335      0.006     55.809      0.000       0.324       0.343
f2             0.4832      0.009     52.719      0.000       0.468       0.498
f3            -0.1680      0.012    -14.117      0.000      -0.188      -0.148
f4            -0.0206      0.013     -1.589      0.116      -0.042       0.001
f5            -0.1727      0.014    -12.151      0.000      -0.196      -0.149
f6            -0.2734      0.018    -15.539      0.000      -0.303      -0.244
f7            -0.0155      0.022     -0.695      0.489      -0.053       0.022
f8             0.1088      0.024      4.555      0.000       0.069       0.149
f9             0.0320      0.035      0.908      0.367      -0.027       0.091
==============================================================================
Omnibus:                        9.551   Durbin-Watson:                   2.146
Prob(Omnibus):                  0.008   Jarque-Bera (JB):               11.569
Skew:                          -0.548   Prob(JB):                      0.00307
Kurtosis:                       4.437   Cond. No.                         5.89
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

File failed to load: /extensions/MathMenu.js

In [240]:

```
#выбрасываем f7, f9 и f4
X_sm=X_sm.drop(columns=["f7","f9","f4"])
y_sm=y_scaled
model=sm.OLS(y_sm,X_sm)
results = model.fit()
print(results.summary(alpha=0.1))
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.988
Model:                            OLS   Adj. R-squared:                  0.987
Method:                 Least Squares   F-statistic:                     1072.
Date:                Wed, 17 Feb 2021   Prob (F-statistic):           9.10e-73
Time:                        22:24:22   Log-Likelihood:                 67.443
No. Observations:                  85   AIC:                            -120.9
Df Residuals:                      78   BIC:                            -103.8
Df Model:                           6                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|       [0.05       0.95]
------------------------------------------------------------------------------
const       1.249e-16      0.012   1.01e-14      1.000      -0.021       0.021
f1             0.3335      0.006     55.514      0.000       0.324       0.344
f2             0.4832      0.009     52.440      0.000       0.468       0.499
f3            -0.1680      0.012    -14.042      0.000      -0.188      -0.148
f5            -0.1727      0.014    -12.087      0.000      -0.196      -0.149
f6            -0.2734      0.018    -15.457      0.000      -0.303      -0.244
f8             0.1088      0.024      4.531      0.000       0.069       0.149
==============================================================================
Omnibus:                       11.379   Durbin-Watson:                   2.087
Prob(Omnibus):                  0.003   Jarque-Bera (JB):               16.627
Skew:                          -0.558   Prob(JB):                     0.000245
Kurtosis:                       4.857   Cond. No.                         4.00
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [241]:

```
kendall_coef_list=[]
p_value_kendall_list=[]
from scipy import stats
for i in range(0,10):
  for j in range(0,10):
    if i==j:
      kendall_coef_list.append(np.nan)
      p_value_kendall_list.append(np.nan)
    if i!=j:
      x=df.iloc[:,i]
      y=df.iloc[:,j]
      tau, p_value = stats.kendalltau(x,y)
      kendall_coef_list.append(tau)
      p_value_kendall_list.append(p_value)
```

In [242]:

```
coef_kendall=pd.DataFrame(np.array(kendall_coef_list).reshape([10,10]),index=df.columns,columns=df.columns)
p_value_kendall=pd.DataFrame(np.array(p_value_kendall_list).reshape([10,10]),index=df.columns,columns=df.columns)
```

In [243]:

```
#коэффициенты Кендалла
coef_kendall
```

Out[243]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP	nan	-0.13713	0.31997	0.12911	0.15531	0.73483	0.36541	0.60438	-0.11468	0.56677
CPI	-0.13713	nan	-0.15531	-0.03395	-0.15958	-0.11361	-0.01149	-0.12323	-0.03502	-0.02460
Exp	0.31997	-0.15531	nan	0.34403	0.72841	0.27720	0.12376	0.16921	-0.05694	0.15934
Inv	0.12911	-0.03395	0.34403	nan	0.44988	0.18257	0.05801	0.15370	0.14248	0.13795
Imp	0.15531	-0.15958	0.72841	0.44988	nan	0.17134	0.04571	0.10933	0.04036	0.03689
Int	0.73483	-0.11361	0.27720	0.18257	0.17134	nan	0.35044	0.64608	-0.05266	0.51972
Mrk cap	0.36541	-0.01149	0.12376	0.05801	0.04571	0.35044	nan	0.31836	-0.14408	0.34648
Life exp	0.60438	-0.12323	0.16921	0.15370	0.10933	0.64608	0.31836	nan	-0.04518	0.46678
Unemp	-0.11468	-0.03502	-0.05694	0.14248	0.04036	-0.05266	-0.14408	-0.04518	nan	-0.05347
Urb	0.56677	-0.02460	0.15934	0.13795	0.03689	0.51972	0.34648	0.46678	-0.05347	nan

In [244]:

```
#тау-статистика Кендалла
tau_kendall=coef_kendall.apply(lambda x: abs(x)*np.sqrt(9*87*(87-1)/(2*(2*87+5))))
tau_kendall
```

Out[244]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP	nan	1.88069	4.38829	1.77071	2.12999	10.07803	5.01152	8.28899	1.57274	7.77311
CPI	1.88069	nan	2.12999	0.46559	2.18864	1.55808	0.15764	1.69006	0.48026	0.33732
Exp	4.38829	2.12999	nan	4.71823	9.99004	3.80172	1.69739	2.32062	0.78087	2.18527
Inv	1.77071	0.46559	4.71823	nan	6.17000	2.50393	0.79554	2.10799	1.95402	1.89195
Imp	2.12999	2.18864	9.99004	6.17000	nan	2.34995	0.62690	1.49942	0.55358	0.50599
Int	10.07803	1.55808	3.80172	2.50393	2.34995	nan	4.80622	8.86089	0.72222	7.12779
Mrk cap	5.01152	0.15764	1.69739	0.79554	0.62690	4.80622	nan	4.36629	1.97601	4.75186
Life exp	8.28899	1.69006	2.32062	2.10799	1.49942	8.86089	4.36629	nan	0.61957	6.40182
Unemp	1.57274	0.48026	0.78087	1.95402	0.55358	0.72222	1.97601	0.61957	nan	0.73331
Urb	7.77311	0.33732	2.18527	1.89195	0.50599	7.12779	4.75186	6.40182	0.73331	nan

In [245]:

```
#значимые коэффициенты
tau_kendall[tau_kendall>1.96].fillna("")
```

Out[245]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP			4.38829		2.12999	10.07803	5.01152	8.28899		7.77311
CPI			2.12999		2.18864					
Exp	4.38829	2.12999		4.71823	9.99004	3.80172		2.32062		2.18527
Inv			4.71823		6.17000	2.50393		2.10799		
Imp	2.12999	2.18864	9.99004	6.17000		2.34995				
Int	10.07803		3.80172	2.50393	2.34995		4.80622	8.86089		7.12779
Mrk cap	5.01152					4.80622		4.36629	1.97601	4.75186
Life exp	8.28899		2.32062	2.10799		8.86089	4.36629			6.40182
Unemp							1.97601			
Urb	7.77311		2.18527			7.12779	4.75186	6.40182		

In [246]:

```
spearman_coef_list=[]
p_value_spearman_list=[]
from scipy import stats
for i in range(0,10):
  for j in range(0,10):
    if i==j:
      spearman_coef_list.append(np.nan)
      p_value_spearman_list.append(np.nan)
    if i!=j:
      x=df.iloc[:,i]
      y=df.iloc[:,j]
      sp, p_value = stats.spearmanr(x,y)
      spearman_coef_list.append(sp)
      p_value_spearman_list.append(p_value)
```

In [247]:

```
coef_spearman=pd.DataFrame(np.array(spearman_coef_list).reshape([10,10]),index=df.columns,columns=df.columns)
p_value_spearman=pd.DataFrame(np.array(p_value_spearman_list).reshape([10,10]),index=df.columns,columns=df.columns)
```

In [248]:

```
#коэффициенты Спирмена
coef_spearman
```

Out[248]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP	nan	-0.24196	0.44793	0.18577	0.23157	0.90989	0.51043	0.80675	-0.12069	0.75491
CPI	-0.24196	nan	-0.23815	-0.04904	-0.24331	-0.21965	-0.03683	-0.21300	-0.05182	-0.04837
Exp	0.44793	-0.23815	nan	0.49293	0.88671	0.37568	0.18096	0.24249	-0.09104	0.22355
Inv	0.18577	-0.04904	0.49293	nan	0.62889	0.25365	0.08406	0.21694	0.21896	0.20386
Imp	0.23157	-0.24331	0.88671	0.62889	nan	0.23859	0.07345	0.16048	0.05940	0.05359
Int	0.90989	-0.21965	0.37568	0.25365	0.23859	nan	0.51638	0.85301	-0.04403	0.71083
Mrk cap	0.51043	-0.03683	0.18096	0.08406	0.07345	0.51638	nan	0.44689	-0.20382	0.50909
Life exp	0.80675	-0.21300	0.24249	0.21694	0.16048	0.85301	0.44689	nan	-0.06200	0.66408
Unemp	-0.12069	-0.05182	-0.09104	0.21896	0.05940	-0.04403	-0.20382	-0.06200	nan	-0.06290
Urb	0.75491	-0.04837	0.22355	0.20386	0.05359	0.71083	0.50909	0.66408	-0.06290	nan

In [249]:

```
#тау-статистика Спирмена
tau_spearman=coef_spearman.apply(lambda x: abs(x)/np.sqrt(1-x**2)*np.sqrt(87-2))
tau_spearman
```

Out[249]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP	nan	2.29910	4.61900	1.74309	2.19466	20.22160	5.47246	12.58770	1.12090	10.61229
CPI	2.29910	nan	2.26071	0.45272	2.31272	2.07581	0.33982	2.00991	0.47836	0.44648
Exp	4.61900	2.26071	nan	5.22323	17.68245	3.73740	1.69639	2.30444	0.84282	2.11451
Inv	1.74309	0.45272	5.22323	nan	7.45741	2.41755	0.77771	2.04887	2.06893	1.91984
Imp	2.19466	2.31272	17.68245	7.45741	nan	2.26511	0.67900	1.49894	0.54858	0.49481
Int	20.22160	2.07581	3.73740	2.41755	2.26511	nan	5.55940	15.06893	0.40636	9.31734
Mrk cap	5.47246	0.33982	1.69639	0.77771	0.67900	5.55940	nan	4.60561	1.91938	5.45309
Life exp	12.58770	2.00991	2.30444	2.04887	1.49894	15.06893	4.60561	nan	0.57274	8.18886
Unemp	1.12090	0.47836	0.84282	2.06893	0.54858	0.40636	1.91938	0.57274	nan	0.58103
Urb	10.61229	0.44648	2.11451	1.91984	0.49481	9.31734	5.45309	8.18886	0.58103	nan

In [250]:

```
#значимые коэффициенты
tau_spearman[tau_spearman>2.23].fillna("")
```

Out[250]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP		2.29910	4.61900			20.22160	5.47246	12.58770		10.61229
CPI	2.29910		2.26071		2.31272					
Exp	4.61900	2.26071		5.22323	17.68245	3.73740		2.30444		
Inv			5.22323		7.45741	2.41755				
Imp		2.31272	17.68245	7.45741		2.26511				
Int	20.22160		3.73740	2.41755	2.26511		5.55940	15.06893		9.31734
Mrk cap	5.47246					5.55940		4.60561		5.45309
Life exp	12.58770		2.30444			15.06893	4.60561			8.18886
Unemp										
Urb	10.61229					9.31734	5.45309	8.18886		

In [251]:

```
combo=tau_spearman[tau_spearman<=2.23].fillna(0)*tau_kendall[tau_kendall<=1.96].fillna(0)
combo=combo.replace(0,"")
combo
```

Out[251]:

	GDP	CPI	Exp	Inv	Imp	Int	Mrk cap	Life exp	Unemp	Urb
GDP				3.08650					1.76288	
CPI				0.21078		3.23428	0.05357	3.39686	0.22973	0.15061
Exp							2.87944		0.65813	
Inv	3.08650	0.21078					0.61870		4.04273	3.63223
Imp							0.42566	2.24755	0.30368	0.25037
Int		3.23428							0.29348	
Mrk cap		0.05357	2.87944	0.61870	0.42566					
Life exp		3.39686			2.24755				0.35485	
Unemp	1.76288	0.22973	0.65813	4.04273	0.30368	0.29348		0.35485		0.42608
Urb		0.15061		3.63223	0.25037				0.42608	

In [252]:

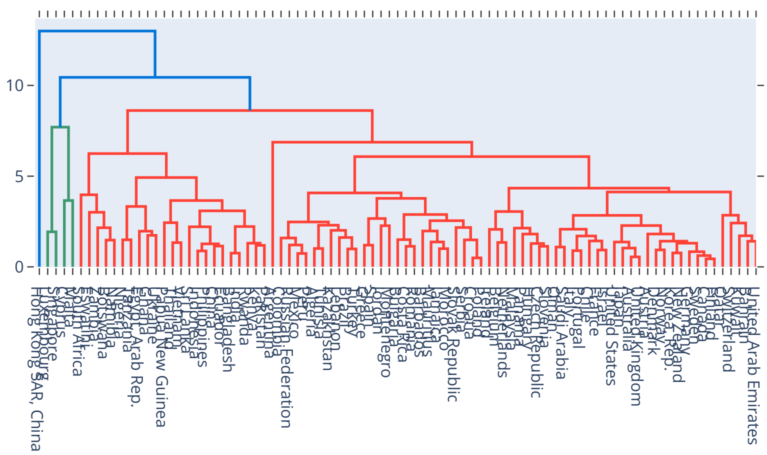
```
combo.iloc[[1,3,8,9],[1,3,8,9]]
```

Out[252]:

	CPI	Inv	Unemp	Urb
CPI		0.21078	0.22973	0.15061
Inv	0.21078		4.04273	3.63223
Unemp	0.22973	4.04273		0.42608
Urb	0.15061	3.63223	0.42608	

In [253]:

```
#группировка в полном признаковом пространстве
import plotly.figure_factory as ff
import numpy as np
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import StandardScaler
standardscaler=StandardScaler()
df_for_poln=df
df_scaled=pd.DataFrame(standardscaler.fit_transform(df_for_poln.values),columns=df_for_poln.columns,index=df_for_poln.index)
X = df_scaled.values
names = df_scaled.index
fig = ff.create_dendrogram(X, labels=names,linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [254]:

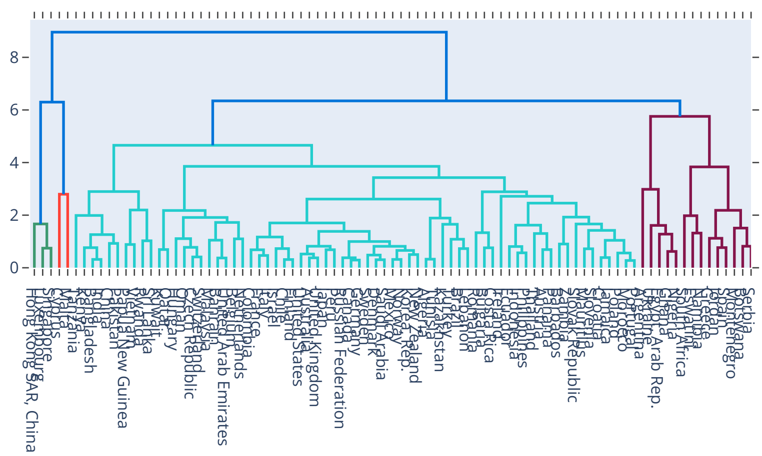
```
df_for_poln=df.drop(index=["Hong Kong SAR, China"])
df_scaled=pd.DataFrame(standardscaler.fit_transform(df_for_poln.values),columns=df_for_poln.columns,index=df_for_poln.index)
X = df_scaled.values
```

In [255]:

```
from scipy.cluster.hierarchy import fclusterdata
pd.set_option('display.max_rows',100)
ip4_poln=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df_scaled.index,columns=["полное признаковое г
```

In [256]:

```
#группировка в сокращенном признаковом пространстве по Кендаллу и Спирмену методом дальнего соседа
import plotly.figure_factory as ff
import numpy as np
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import StandardScaler
standardscaler=StandardScaler()
df_for_sokr=df[["CPI","Exp","Inv","Unemp","Urb"]]
df_scaled=pd.DataFrame(standardscaler.fit_transform(df_for_sokr.values),columns=df_for_sokr.columns,index=df_for_sokr.index)
X = df_scaled.values
names = df_scaled.index
fig = ff.create_dendrogram(X, labels=names,linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [257]:

```
ip4_sokr=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df_scaled.index,columns=["сокращенное признакс
countries_cl=pd.concat([ip4_poln,ip4_sokr],axis=1)
countries_cl.to_excel("группировка ИП4.xlsx")
```

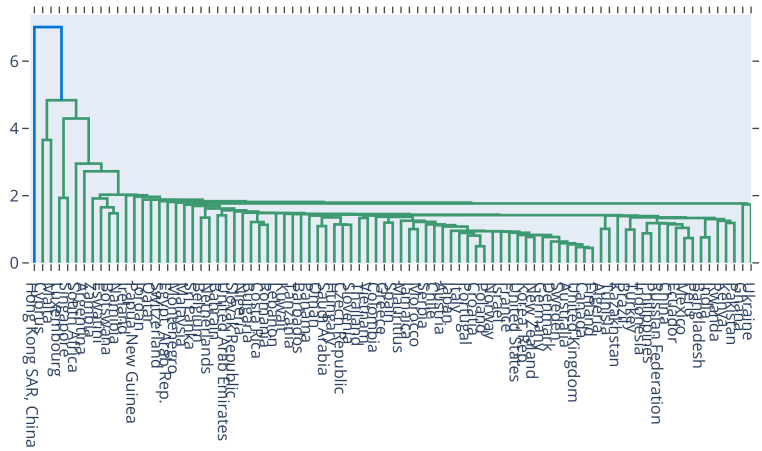
Кластерный анализ

In [258]:

```
#стандартизируем данные
from sklearn.preprocessing import StandardScaler
standardscaler=StandardScaler()
df_scaled=pd.DataFrame(standardscaler.fit_transform(df.values),columns=df.columns,index=df.index)
```

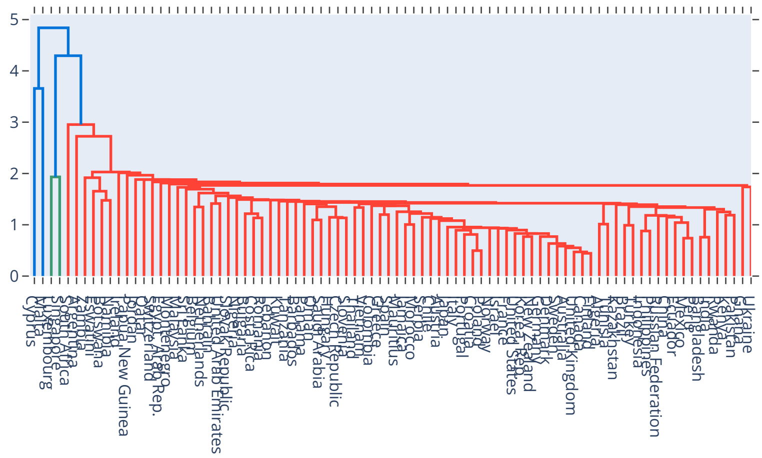
In [259]:

```
#метод ближайшего соседа на первоначальных данных
import plotly.figure_factory as ff
import numpy as np
import scipy.cluster.hierarchy as sch
X = df_scaled.values
names = df_scaled.index
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "single"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



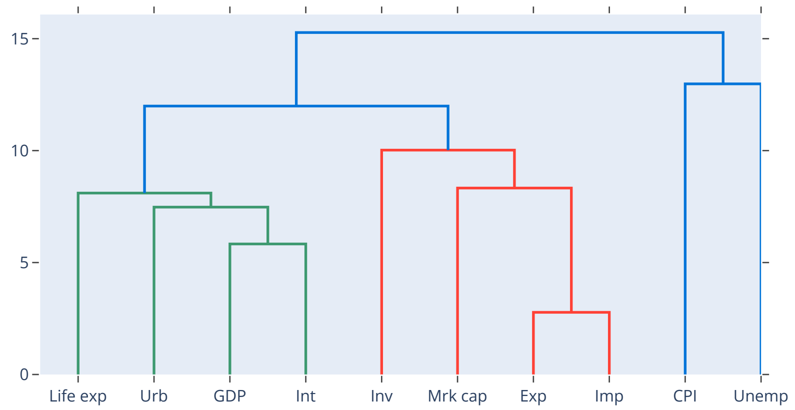
In [260]:

```
#убираем 1 выброс - Гонконг
df_dropped=df_scaled.drop(index=["Hong Kong SAR, China"])
X = df_dropped.values
names = df_dropped.index
fig = ff.create_dendrogram(X, labels=names,linkagefun=lambda x: sch.linkage(x, "single"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



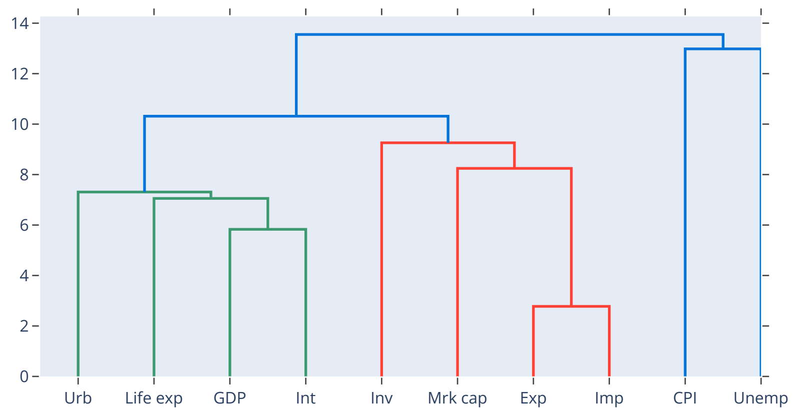
In [261]:

```
#метод дальнего соседа
X = df_dropped.values.T
names = df_dropped.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [262]:

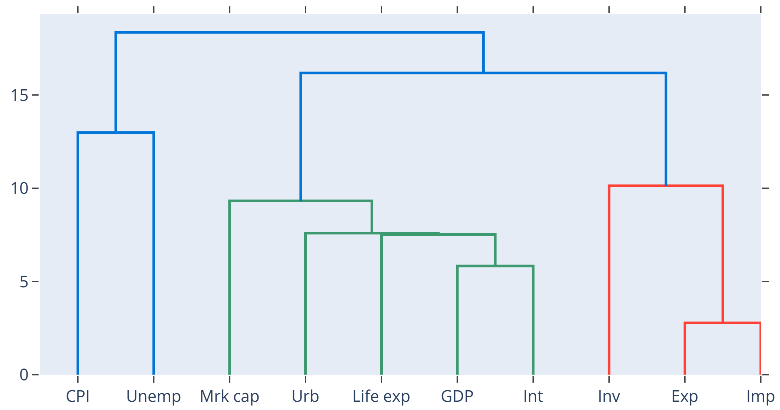
```
#group average method
X = df_dropped.values.T
names = df_dropped.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "average"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



File failed to load: /extensions/MathMenu.js

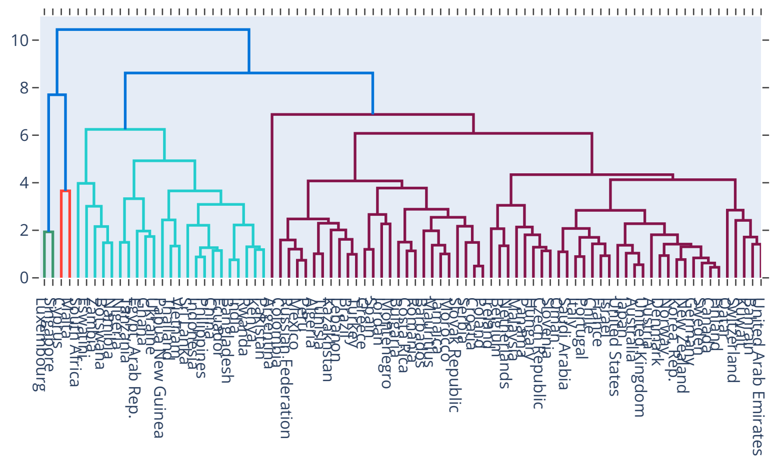
In [263]:

```
#метод Варда
X = df_dropped.values.T
names = df_dropped.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "ward"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [264]:

```
#Метод дальнего соседа в ПОЛНОМ признаковом пространстве (без выбросов)
X = df_dropped.values
names = df_dropped.index
fig = ff.create_dendrogram(X, labels=names,linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```

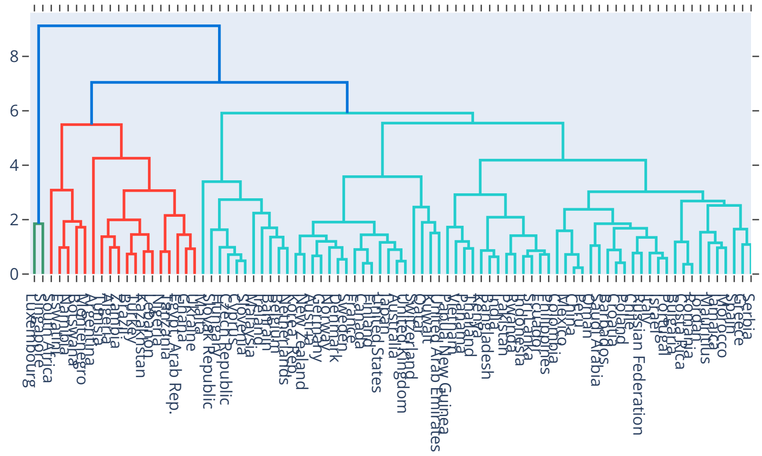


In [266]:

```
from scipy.cluster.hierarchy import fclusterdata
pd.set_option('display.max_rows',100)
countries_clusters_poln=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df_sokr.index,columns=["полное
```

In [265]:

```
#Метод дальнего соседа в СОКРАЩЕННОМ признаковом пространстве (без выбросов)
df_sokr=df_dropped.drop(columns=["Life exp","Urb","Inv"])
X = df_sokr.values
names = df_sokr.index
fig = ff.create_dendrogram(X, labels=names,linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [267]:

```
countries_clusters_sokr=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df_sokr.index,columns=["сокраще
countries_clusters=pd.concat([countries_clusters_poln,countries_clusters_sokr],axis=1)
```

In [268]:

```
#countries_clusters.to_excel("кластеры по странам.xlsx")
#files.download("кластеры по странам.xlsx")
```

In [269]:

```
import pandas as pd
df=pd.read_excel("ИП1 опрос данные измененные.xlsx",index_col=0)
```

In [270]:

```
df
```

Out[270]:

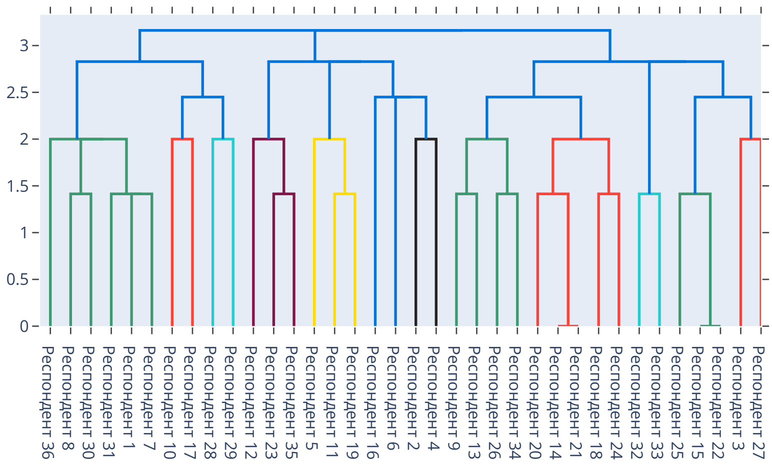
	Женский	Мужской	18 - 20	21-23	24-30	31-40	41 и старше	Высшее (бакалавриат или специалитет)	Высшее (магистратура и выше)	Основное общее	...	Среднее профессиональное	Гуманитарны науки социологи
Респондент 1	0	1	1	0	0	0	0	0	0	0	...	0	
Респондент 2	0	1	0	0	1	0	0	0	0	0	...	1	
Респондент 3	1	0	1	0	0	0	0	0	0	0	...	0	
Респондент 4	0	1	0	0	0	0	1	0	0	0	...	1	
Респондент 5	1	0	0	0	0	0	1	0	1	0	...	0	
Респондент 6	0	1	0	0	0	1	0	0	1	0	...	0	
Респондент 7	0	1	1	0	0	0	0	0	0	0	...	0	
Респондент 8	0	1	1	0	0	0	0	0	0	0	...	0	
Респондент 9	1	0	0	1	0	0	0	0	0	0	...	0	
Респондент 10	1	0	1	0	0	0	0	0	0	1	...	0	
Респондент 11	1	0	0	0	1	0	0	0	1	0	...	0	
Респондент 12	1	0	0	1	0	0	0	1	0	0	...	0	
Респондент 13	1	0	0	1	0	0	0	0	0	0	...	0	
Респондент 14	0	1	0	1	0	0	0	0	0	0	...	0	
Респондент 15	1	0	0	1	0	0	0	0	0	0	...	0	
Респондент 16	0	1	0	0	1	0	0	1	0	0	...	0	
Респондент 17	1	0	1	0	0	0	0	0	0	1	...	0	
Респондент 18	0	1	0	1	0	0	0	0	0	0	...	1	
Респондент 19	1	0	0	0	1	0	0	0	1	0	...	0	
Респондент 20	0	1	0	1	0	0	0	0	0	0	...	0	
Респондент 21	0	1	0	1	0	0	0	0	0	0	...	0	
Респондент 22	1	0	0	1	0	0	0	0	0	0	...	0	
Респондент 23	1	0	0	1	0	0	0	1	0	0	...	0	
Респондент 24	0	1	0	1	0	0	0	1	0	0	...	0	
Респондент 25	0	1	0	1	0	0	0	0	0	0	...	0	
Респондент 26	0	1	0	1	0	0	0	0	0	0	...	0	
Респондент 27	1	0	0	1	0	0	0	0	0	0	...	0	
Респондент 28	1	0	1	0	0	0	0	1	0	0	...	0	
Респондент 29	1	0	1	0	0	0	0	1	0	0	...	0	
Респондент 30	0	1	1	0	0	0	0	0	0	0	...	0	
Респондент 31	0	1	1	0	0	0	0	0	0	0	...	0	
Респондент 32	1	0	0	1	0	0	0	0	0	0	...	1	
Респондент 33	1	0	0	1	0	0	0	0	0	0	...	1	
Респондент 34	0	1	0	1	0	0	0	0	0	0	...	0	

	Женский	Мужской	18 - 20	21-23	24-30	31-40	41 и старше	Высшее (бакалавриат или специалитет)	Высшее (магистратура и выше)	Основное общее	...	Среднее профессиональное	Гуманитарнь науки социологи
Респондент 35	0	1	0	1	0	0	0	1	0	0	...	0	
Респондент 36	0	1	1	0	0	0	0	0	0	0	...	0	

36 rows × 21 columns

In [271]:

```
#ПОЛНОЕ признаковое пространство (метод дальнего соседа)
import plotly.figure_factory as ff
import numpy as np
import scipy.cluster.hierarchy as sch
X = df.values
names = df.index
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



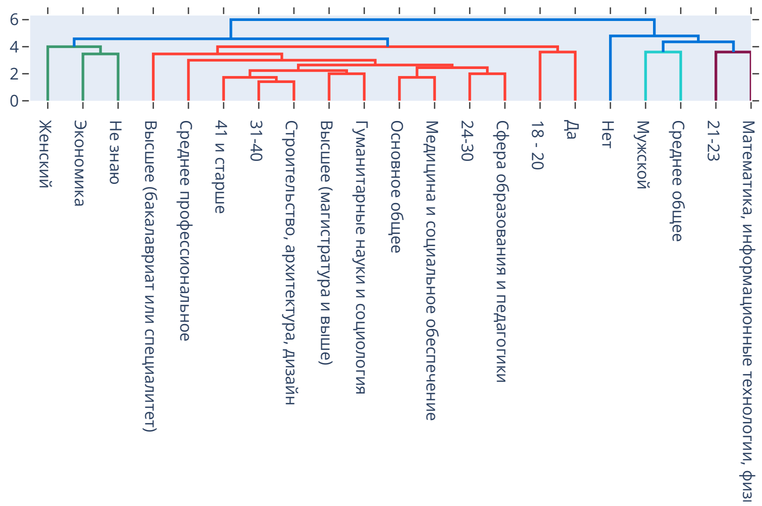
In [272]:

```
from scipy.cluster.hierarchy import fclusterdata
ip5_poln=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df.index,columns=["Полное признаковое простран
```

File failed to load: /extensions/MathMenu.js

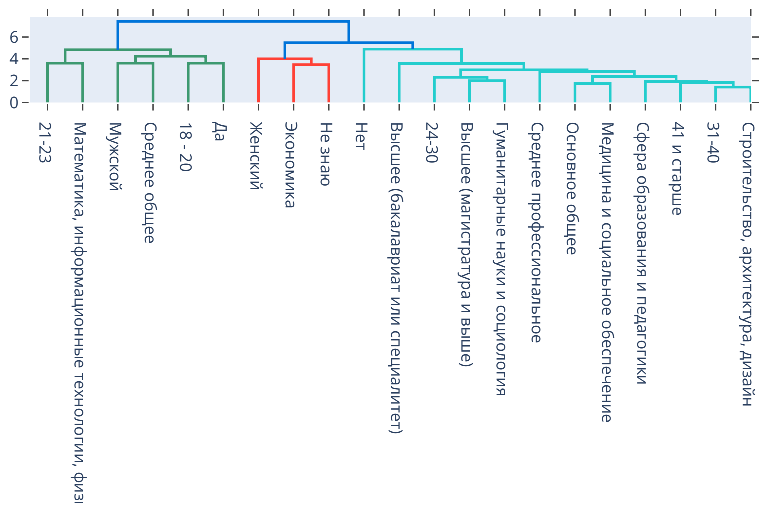
In [273]:

```
#дендрограмма признаков (метод дальнего соседа)
X = df.values.T
names = df.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



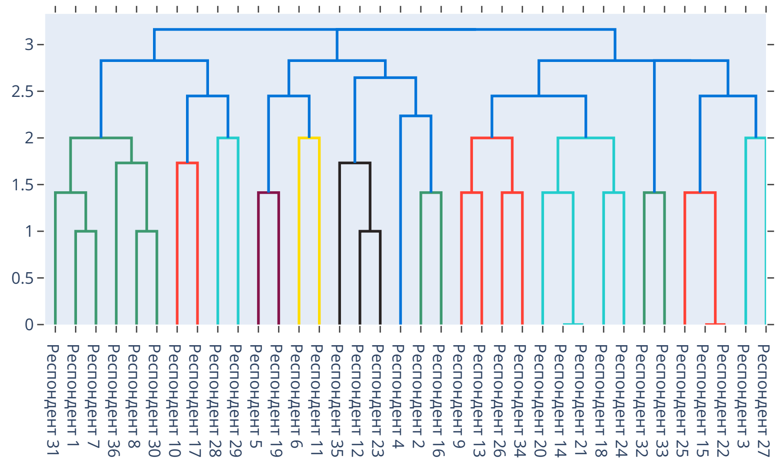
In [274]:

```
#дендрограмма признаков (метод Уорда)
X = df.values.T
names = df.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "ward"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [275]:

```
#СОКРАЩЕННОЕ признаковое пространство - убрали ... (метод дальнего соседа)
X = df.drop(columns=["Строительство, архитектура, дизайн", "Медицина и социальное обеспечение","Сфера образования и педагогики"]).va]
names = df.index
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [276]:

```
ip5_sokr=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df.index,columns=["Сокращенное признаковое прс
ip5_2_cl=pd.concat([ip5_poln,ip5_sokr],axis=1)
ip5_2_cl.to_excel("группировка ИП5 часть 2.xlsx")
```

In [277]:

```
import pandas as pd
df=pd.read_excel("подсчет дубликатов.xlsx",index_col=0)
df
```

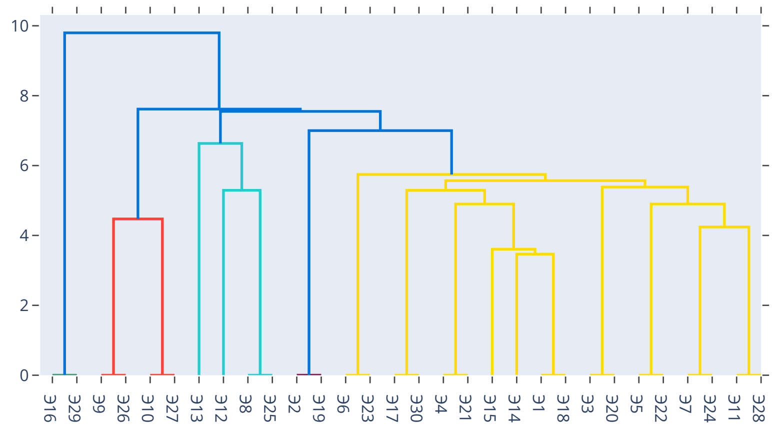
Out[277]:

	Э1	Э2	Э3	Э4	Э5	Э6	Э7	Э8	Э9	Э10	...	Э21	Э22	Э23	Э24	Э25	Э26	Э27	Э28	Э29	Э30
x1	3	1	8	5	5	4	5	1	1	1	...	5	5	4	5	1	1	1	7	6	8
x2	2	1	1	5	2	4	1	1	8	7	...	5	2	4	1	1	8	7	1	8	7
x3	9	9	9	9	9	9	9	9	9	9	...	9	9	9	9	9	9	9	9	2	9
x4	6	1	6	3	5	4	5	5	4	5	...	3	5	4	5	5	4	5	3	4	5
x5	3	1	1	3	2	1	1	8	1	1	...	3	2	1	1	8	1	1	1	8	2
x6	8	7	3	5	2	4	4	5	6	8	...	5	2	4	4	5	6	8	3	3	2
x7	1	6	6	1	1	1	1	1	6	5	...	1	1	1	1	1	6	5	3	6	2
x8	3	1	4	1	5	4	8	7	4	1	...	1	5	4	8	7	4	1	7	1	1
x9	6	8	4	5	8	1	5	1	3	1	...	5	8	1	5	1	3	1	3	4	5

9 rows × 30 columns

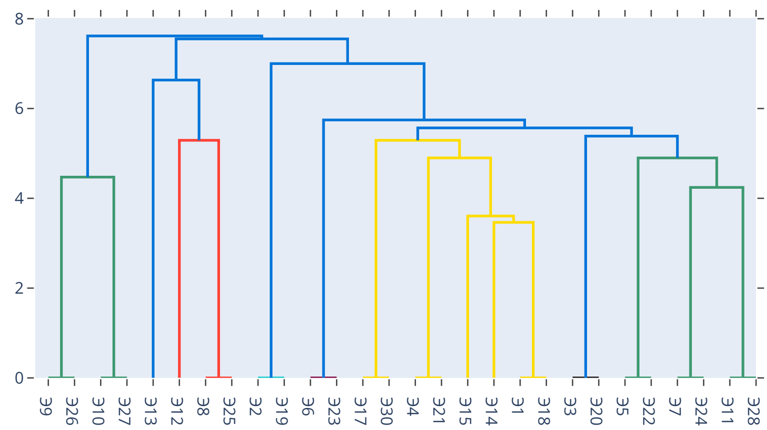
In [278]:

```
#дендрограмма всех экспертов (метод ближнего соседа)
import plotly.figure_factory as ff
import numpy as np
import scipy.cluster.hierarchy as sch
from scipy.cluster.hierarchy import fclusterdata
X = df.values.T
names = df.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "single"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



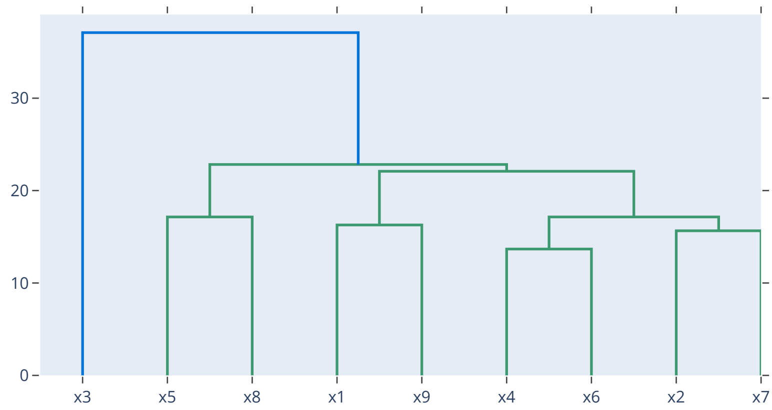
In [279]:

```
#убираем выбросы - Э16 и Э29
df_dropped=df.drop(columns=["Э16","Э29"])
X = df_dropped.values.T
names = df_dropped.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "single"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



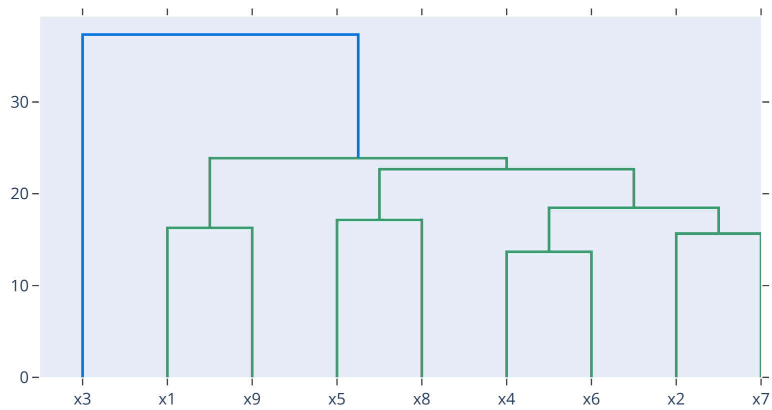
In [280]:

```
#оцениваем признаки (без выбросов) - метод дальнего соседа
X = df_dropped.values
names = df_dropped.index
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



In [281]:

```
#метод Уорда
X = df_dropped.values
names = df_dropped.index
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "ward"))
fig.update_layout(width=1300, height=1000)
fig.show()
```

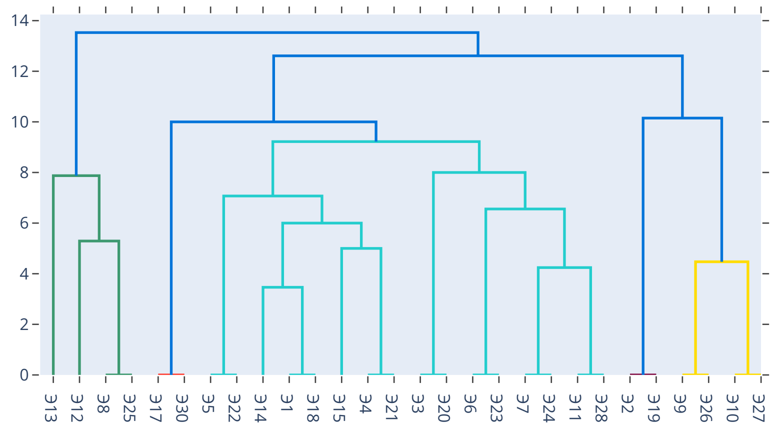


Отбираем x3, x4, x8 (и x1?)

File failed to load: /extensions/MathMenu.js

In [282]:

```
#Группировка в полном признаковом пространстве методом дальнего соседа (без выбросов)
X = df_dropped.values.T
names = df_dropped.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



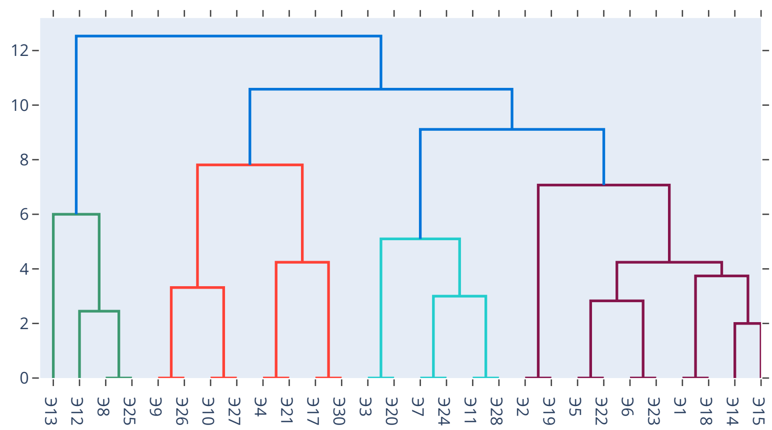
In [283]:

```
ip5_3_poln=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df_dropped.T.index,columns=["Полное признако
```

x6 x7

In [284]:

```
#Группировка в СОКРАЩЕННОМ признаковом пространстве методом дальнего соседа (без выбросов)
df_sokr=df_dropped.drop(index=["x6","x7","x9"])
X = df_sokr.values.T
names = df_sokr.columns
fig = ff.create_dendrogram(X, labels=names, linkagefun=lambda x: sch.linkage(x, "complete"))
fig.update_layout(width=1300, height=1000)
fig.show()
```



File failed to load: /extensions/MathMenu.js

In [285]:

```
ip5_3_sokr=pd.DataFrame(fclusterdata(X,criterion='maxclust',method='complete',t=3),index=df_dropped.T.index,columns=["Сокращенное при
ip5_3_cl=pd.concat([ip5_3_poln,ip5_3_sokr],axis=1)
ip5_3_cl["Сокращенное признаковое пространство"]=ip5_3_cl["Сокращенное признаковое пространство"].apply(lambda x: 2 if x==3 else 3)
ip5_3_cl["разница"]=ip5_3_cl["Полное признаковое пространство"]-ip5_3_cl["Сокращенное признаковое пространство"]
ip5_3_cl
```

Out[285]:

	Полное признаковое пространство	Сокращенное признаковое пространство	разница
Э1	2	2	0
Э2	3	2	1
Э3	2	2	0
Э4	2	3	-1
Э5	2	2	0
Э6	2	2	0
Э7	2	2	0
Э8	1	3	-2
Э9	3	3	0
Э10	3	3	0
Э11	2	2	0
Э12	1	3	-2
Э13	1	3	-2
Э14	2	2	0
Э15	2	2	0
Э17	2	3	-1
Э18	2	2	0
Э19	3	2	1
Э20	2	2	0
Э21	2	3	-1
Э22	2	2	0
Э23	2	2	0
Э24	2	2	0
Э25	1	3	-2
Э26	3	3	0
Э27	3	3	0
Э28	2	2	0
Э30	2	3	-1

In [286]:

```
ip5_3_cl.to_excel("группировка ИП5 часть 3.xlsx")
```