Q5_Analysis

June 7, 2021

This is an example of analysis of (anonymous) questionnaire related to motivation system for Generation Z. The full discussion of results can be found in:

- R.A. Kycia, A. Niemczynowicz, J. Niezurawska-Zajac, *Towards the global vision of engagement of Generation Z at the workplace: Mathematical modeling*, Proceedings of IBIMA Conference 2021.

The part of questionnaire is as follows:

Section 5 (S5) Do you agree with the statements below about your engagement at your workplace?

5 – I fully agree 4 – I agree 3 – I'm not sure 2 – I don't agree 1 – I fully disagree

- Q5.1: I'm very satisfied with the work I do
- Q5.2: My job is interesting
- Q5.3: I know exactly what I'm expected to do
- Q5.4: I am prepared to show initiative to do my work well
- Q5.5: My job is challenging (sets new goals, is prospective)
- Q5.6: I have plenty of freedom how to do my work
- Q5.7: I get plenty of opportunities to learn in this job
- Q5.8: The facilities/equipment/tools provided are excellent.
- Q5.9: I have a lot of support from my boss.
- Q5.10: My boss recognizes my work.
- Q5.11: The experience I am getting now will be great help in advancing my future career.
- Q5.12: I find it easy to keep up with the demands of my job.
- Q5.13: I have no problems in achieving balance between my professional and private life.
- Q5.14: I like working with my boss.
- Q5.15: I get on well with my work colleagues.
- Q5.16: I think this organization is a great place to work.
- Q5.17: I believe I have a great future in this organization.
- Q5.18: I intend to go on working for this organization.
- Q5.19: I am happy about the values of this organization – how it conducts its business.
- Q5.20: The products/services provided by this organization are excellent.

Disclaimer:

This is only an example. We tried to write it using high standards, however we are not responsible for all the damages that can be made by this code/notebook. Use at your own risk.

1 Import

[1]:
```python
import numpy as np
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile
import matplotlib.pyplot as plt
import seaborn as sns
```

2 Read data

[2]:
```python
df = pd.read_excel("data.xlsx", sheet_name='Q5', header=0)
```

[3]:
```python
df.head()
```

[3]:
```
   Q5.1  Q5.2  Q5.3  Q5.4  Q5.5  Q5.6  Q5.7  Q5.8  Q5.9  Q5.10  Q5.11  Q5.12  \
0     3     3     4     4     3     3     2     4     4      3      1      3
1     5     4     3     4     3     3     5     4     5      5      5      3
2     3     3     5     4     3     1     5     3     5      5      3      4
3     3     4     5     5     3     5     3     5     4      4      4      5
4     3     4     4     4     4     3     4     4     3      3      3      4

   Q5,13  Q5.14  Q5.15  Q5.16  Q5.17  Q5.18  Q5.19  Q5.20
0      4      4      3      4      2      2      3      4
1      3      5      5      4      4      4      1      4
2      2      5      4      3      1      3      3      4
3      5      5      5      4      3      3      3      2
4      4      4      4      4      4      3      3      3
```

[4]:
```python
df.columns
```

[4]:
```
Index(['Q5.1', 'Q5.2', 'Q5.3', 'Q5.4', 'Q5.5', 'Q5.6', 'Q5.7', 'Q5.8', 'Q5.9',
       'Q5.10', 'Q5.11', 'Q5.12', 'Q5,13', 'Q5.14', 'Q5.15', 'Q5.16', 'Q5.17',
       'Q5.18', 'Q5.19', 'Q5.20'],
      dtype='object')
```

[5]:
```python
#copy of data frame - an example of indexing
df_Q5 = df.loc[:,"Q5.1":"Q5.20"].copy()
```

[6]:
```python
df_Q5.head()
```

[6]:
```
   Q5.1  Q5.2  Q5.3  Q5.4  Q5.5  Q5.6  Q5.7  Q5.8  Q5.9  Q5.10  Q5.11  Q5.12  \
0     3     3     4     4     3     3     2     4     4      3      1      3
1     5     4     3     4     3     3     5     4     5      5      5      3
2     3     3     5     4     3     1     5     3     5      5      3      4
3     3     4     5     5     3     5     3     5     4      4      4      5
```

```
4     3     4     4     4     4     3     4     4     3      3      3      4

   Q5,13  Q5.14  Q5.15  Q5.16  Q5.17  Q5.18  Q5.19  Q5.20
0      4      4      3      4      2      2      3      4
1      3      5      5      4      4      4      1      4
2      2      5      4      3      1      3      3      4
3      5      5      5      4      3      3      3      2
4      4      4      4      4      4      3      3      3
```

3 Cleaning data

[7]:
```
#checking NaN (Not a Number)
df_Q5.isnull().values.any()
```

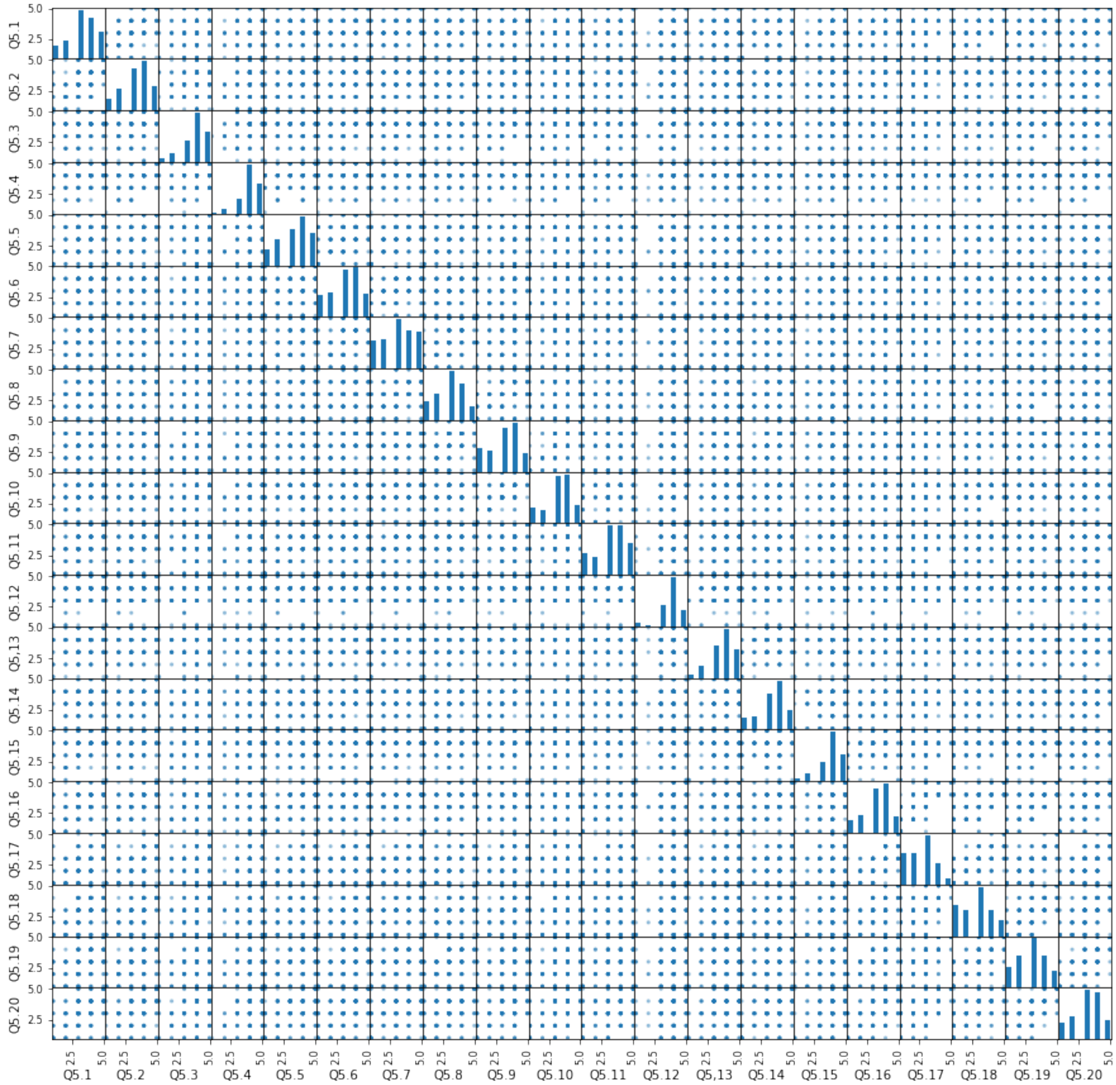
[7]:
```
False
```

The data does not contain NaN values, so data are clean.

4 Correlation analysis

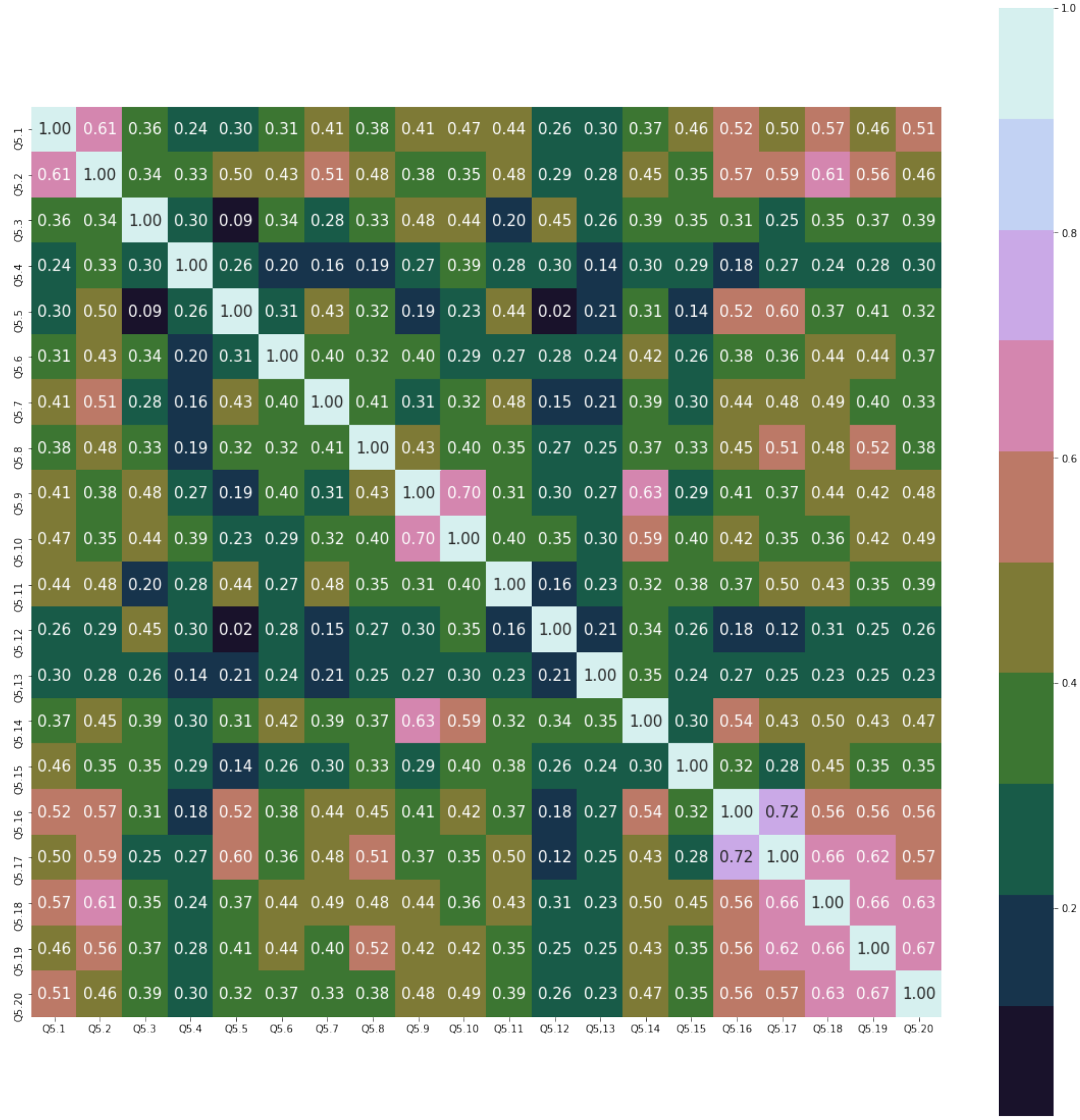
[8]:
```
def cov_matrix(m, method = 'spearman'):
    """Plot covariance matrix and returns it
    method = {'pearson', 'kendall', 'spearman'}
    """
    print(m.columns)
    cov_data = m.corr(method)
    plt.figure(figsize=(20,20))
    sns.heatmap(cov_data, cbar=True, cmap= sns.color_palette("cubehelix", 10),␣
 ↪annot=True, square=True, annot_kws={'size':15}, fmt='.2f', xticklabels= m.
 ↪columns, yticklabels=m.columns)
    return(cov_data)
```

[9]:
```
from pandas.plotting import scatter_matrix
paverageScatterPlot=scatter_matrix(df_Q5, alpha =0.5, figsize=(15,15),␣
 ↪grid=True)
```



[10]:
```
#Correlation matrix
CorMatrix = cov_matrix(df_Q5)
plt.savefig("Q5CorrelationMatrix.png")
```

```
Index(['Q5.1', 'Q5.2', 'Q5.3', 'Q5.4', 'Q5.5', 'Q5.6', 'Q5.7', 'Q5.8', 'Q5.9',
       'Q5.10', 'Q5.11', 'Q5.12', 'Q5,13', 'Q5.14', 'Q5.15', 'Q5.16', 'Q5.17',
       'Q5.18', 'Q5.19', 'Q5.20'],
      dtype='object')
```



Below we present noticable correlations between answers for specific questions which are also suggestions for designing motivaion systems.

- The first group - clusters expression of general satsifaction from the work.
  - Q5.1 (I'm very satisfied with the work I do)
  - Q5.2 (My job is interesting)
  - Q5.16 (I think this organization is a great place to work)
  - Q5.17 (I believe I have a great future in this organization)
  - Q5.18 (I intend to go on working for this organization)
  - Q5.19 (I am happy about the values of this organization – how it conducts its business)
  - Q5.20 (The products/services provided by this organization are excellent)
- The second group - connects learning oppurtunities and the level of interests from the work:

  - Q5.2 (My job is interesting)
  - Q5.7 (I get plenty of opportunities to learn in this job)
- The third group - connets chellenging work and satisfaction with great prospects for future
  - Q5.5 (My job is challenging (sets new goals, is prospective)
  - Q5.16 (I think this organization is a great place to work)
  - Q5.17 (I believe I have a great future in this organization)
- The fourth group - connects the quality of tools and facilietes with the values and with prospects for future
  - Q5.8 (The facilities/equipment/tools provided are excellent)
  - Q5.17 (I believe I have a great future in this organization)
  - Q5.19 (I am happy about the values of this organization – how it conducts its business)
- The fith group - clusters the relations with the boss
  - Q5.9 (I have a lot of support from my boss)
  - Q5.10 (My boss recognizes my work)
  - Q5.14 (I like working with my boss)
- The sixth group - indicates that the relations with the boss is the main ingredient of satisfaction from work
  - Q5.14 (I like working with my boss)
  - Q5.16 (I think this organization is a great place to work)

5 PCA

[11]:
```
#transpose dataframe
df_Q5T = df_Q5.T.copy(); df_Q5T.head()
```

[11]:
```
      0  1  2  3  4  5  6  7  8  9  …  190  191  192  \
Q5.1  3  5  3  3  3  2  4  4  1  3  …    4    3    3
Q5.2  3  4  3  4  4  1  4  3  1  4  …    2    4    5
Q5.3  4  3  5  5  4  4  5  5  3  4  …    2    4    5
Q5.4  4  4  4  5  4  4  3  3  1  4  …    3    4    5
Q5.5  3  3  3  3  4  1  3  4  1  3  …    4    4    5

      193  194  195  196  197  198  199
Q5.1    3    4    2    3    2    1    1
Q5.2    3    2    3    4    1    5    4
Q5.3    3    2    3    4    2    5    5
Q5.4    4    2    2    4    3    5    4
Q5.5    4    4    4    5    4    5    5

[5 rows x 200 columns]
```

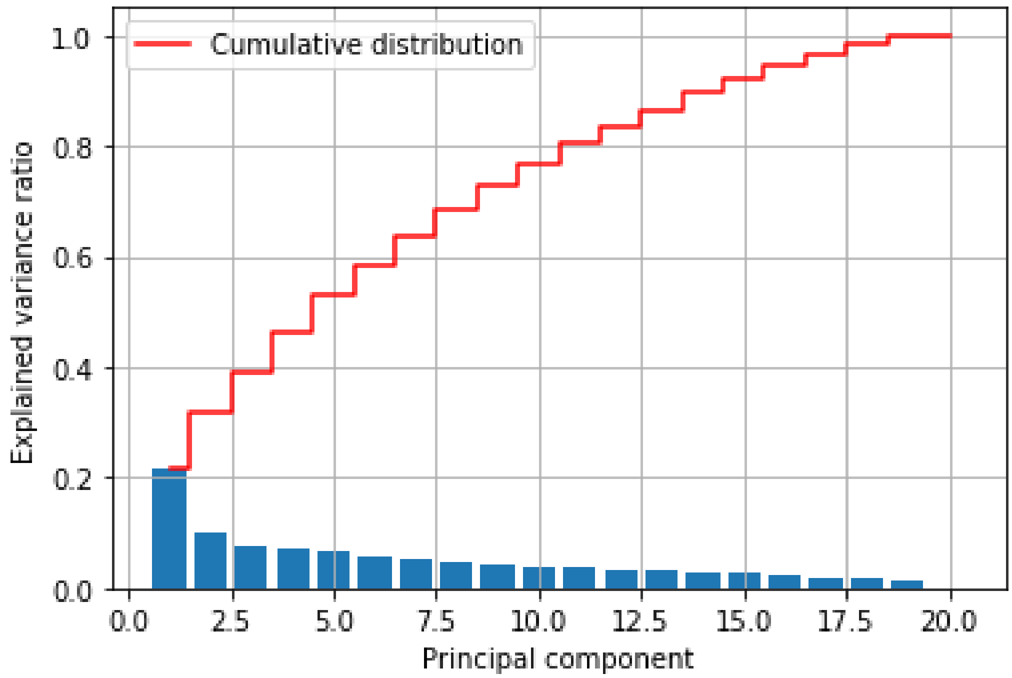
[12]:
```
# import
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

[13]:
```
#standarization of data
df_st =  StandardScaler().fit_transform(df_Q5T)
```

[14]:
```
#PCA
pca_out = PCA().fit(df_st)
```

[15]:
```
print("Explained variance ratio = ", pca_out.explained_variance_ratio_)
print("Explained variance (eigenvalues) = ", pca_out.explained_variance_)
print("Cumulative sum = ", np.cumsum(pca_out.explained_variance_ratio_))

plt.grid(True)
plt.step(range(1,len(pca_out.explained_variance_ratio_)+1), np.cumsum(pca_out.
 ↪explained_variance_ratio_), where='mid', color='red', label='Cumulative␣
 ↪distribution')

plt.xlabel("Principal component")
plt.ylabel("Explained variance ratio")
plt.bar(range(1,len(pca_out.explained_variance_ratio_)+1),pca_out.
 ↪explained_variance_ratio_)
plt.legend()
plt.savefig("pca.png")
```

```
Explained variance ratio =  [2.17797771e-01 9.92819512e-02 7.46577769e-02
6.94343593e-02
 6.68968814e-02 5.47357588e-02 5.26694959e-02 4.96076113e-02
 4.36607429e-02 3.92880197e-02 3.52773558e-02 3.19774077e-02
 3.04258514e-02 2.96470197e-02 2.66644542e-02 2.39750826e-02
 2.02341390e-02 1.82762794e-02 1.54920422e-02 2.17690131e-32]
Explained variance (eigenvalues) =  [4.53936406e+01 2.06924488e+01
1.55602525e+01 1.44715823e+01
 1.39427184e+01 1.14080845e+01 1.09774318e+01 1.03392706e+01
 9.09981799e+00 8.18845042e+00 7.35254363e+00 6.66476497e+00
 6.34138798e+00 6.17906306e+00 5.55743362e+00 4.99691196e+00
 4.21722054e+00 3.80916139e+00 3.22886774e+00 4.53712063e-30]
Cumulative sum =  [0.21779777 0.31707972 0.3917375  0.46117186 0.52806874
0.5828045
 0.63547399 0.68508161 0.72874235 0.76803037 0.80330772 0.83528513
 0.86571098 0.895358   0.92202246 0.94599754 0.96623168 0.98450796
 1.         1.        ]
```



It results that we should take at least 9 components for explainign at least 70% of variance.

6 PCA with 9 components

[16]:
```python
#get standarized scores
scores=PCA(n_components=9).fit_transform(df_st)
num_pc = pca_out.n_features_
cols =["PC"+str(i) for i in list(range(1, 10))]
df_scores=pd.DataFrame(scores, columns = cols, index=df_Q5T.index)
df_scores
```

[16]:

	PC1	PC2	PC3	PC4	PC5	PC6 \
Q5.1	-1.072508	-0.205395	-3.744323	-2.548834	-5.214509	-3.230973
Q5.2	0.539856	4.381662	-1.296618	1.461805	-1.195015	-1.427036
Q5.3	-9.434172	-2.959422	0.283085	2.301236	-0.203073	0.023097
Q5.4	-12.190105	3.851061	-3.845692	-1.400164	1.896829	-1.532640
Q5.5	2.420341	11.040246	0.282513	-0.075163	7.920347	1.162162
Q5.6	2.222431	-0.270571	4.192161	10.400137	2.130560	-7.234127
Q5.7	4.662478	4.817740	10.536602	-0.510792	-6.061730	-1.149040
Q5.8	4.775332	-2.571513	3.034487	2.171818	-3.331634	8.735359
Q5.9	1.241159	-9.359327	4.635144	-3.275167	3.854318	-1.677873
Q5.10	-1.802763	-6.516394	1.977409	-5.679006	2.834638	-0.776718
Q5.11	1.078349	6.273068	2.013666	-8.478055	-1.397613	-1.649695
Q5.12	-8.484684	-0.101190	-0.430150	3.047781	-1.756650	2.107388

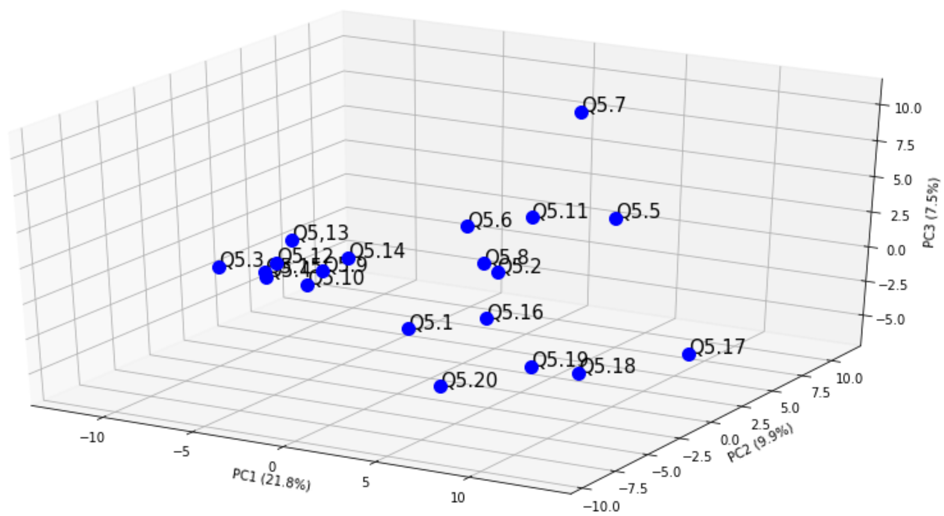
```
Q5,13  -7.674070   0.073352   1.282243   2.253755  0.557193  6.423755
Q5.14  -1.813430  -3.547096   2.606910  -2.032879  5.057454 -0.810438
Q5.15  -9.673821   0.536800  -1.654180  -0.621523 -5.121664 -0.364441
Q5.16   2.761785   0.474692  -2.540860   1.234092  3.046170  1.992303
Q5.17  13.473749   0.753643  -3.061074  -0.826653  2.218792  2.873957
Q5.18   9.665149  -2.196495  -3.912901  -0.705799 -4.693951 -2.010078
Q5.19   6.715757  -1.682706  -4.286014   3.071195 -1.014041 -0.155071
Q5.20   2.589166  -2.792156  -6.072409   0.212216  0.473578 -1.299892

             PC7       PC8       PC9
Q5.1   6.002138 -1.732583 -4.654623
Q5.2   2.067605 -1.795215 -4.610541
Q5.3  -2.963912 -4.448509 -0.086511
Q5.4  -4.523083  0.570390 -0.615326
Q5.5  -0.918652 -1.546126 -0.590189
Q5.6   1.477259  5.545999 -0.323676
Q5.7  -1.717454 -6.025343  2.884440
Q5.8  -3.869955  3.083605 -4.133780
Q5.9   0.231792  0.072244 -1.511283
Q5.10 -1.003776  0.942674 -0.423754
Q5.11 -0.430706  6.964334  2.091437
Q5.12 -2.987103  3.080188 -1.960978
Q5,13  8.883074  1.466386  5.885487
Q5.14  1.548461 -2.244098  0.244141
Q5.15  0.320384  0.684649  0.482115
Q5.16  2.170933 -4.438071 -2.362218
Q5.17  0.916433  1.429212 -1.096276
Q5.18  0.877754  0.615608  0.804627
Q5.19 -3.957770 -1.033931  4.312489
Q5.20 -2.123422 -1.191411  5.664420
```

We now plot clusters projecting to the subspace spanned by first three PC.

[17]:
```python
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
from numpy.random import rand
from pylab import figure


fig = figure()

fig = plt.figure(figsize=plt.figaspect(0.5)*1.5)
ax = fig.gca(projection='3d')
ax = Axes3D(fig)


for i in range(len(scores[:,0])):
```

```
    ax.scatter(scores[i,0],scores[i,1],scores[i,2],color='b', s=100)
    ax.text(scores[i,0],scores[i,1],scores[i,2],  '%s' % (df_scores.index[i]),␣
 ↪size=15,zorder=1,  color='k')

plt.grid(True)
ax.set_xlabel("PC1 ({:.1f}%)".format(pca_out.explained_variance_ratio_[0]*100))
ax.set_ylabel("PC2 ({:.1f}%)".format(pca_out.explained_variance_ratio_[1]*100))
ax.set_zlabel("PC3 ({:.1f}%)".format(pca_out.explained_variance_ratio_[2]*100))

plt.savefig("pca9-3dim.png")
```

```
<Figure size 432x288 with 0 Axes>
```

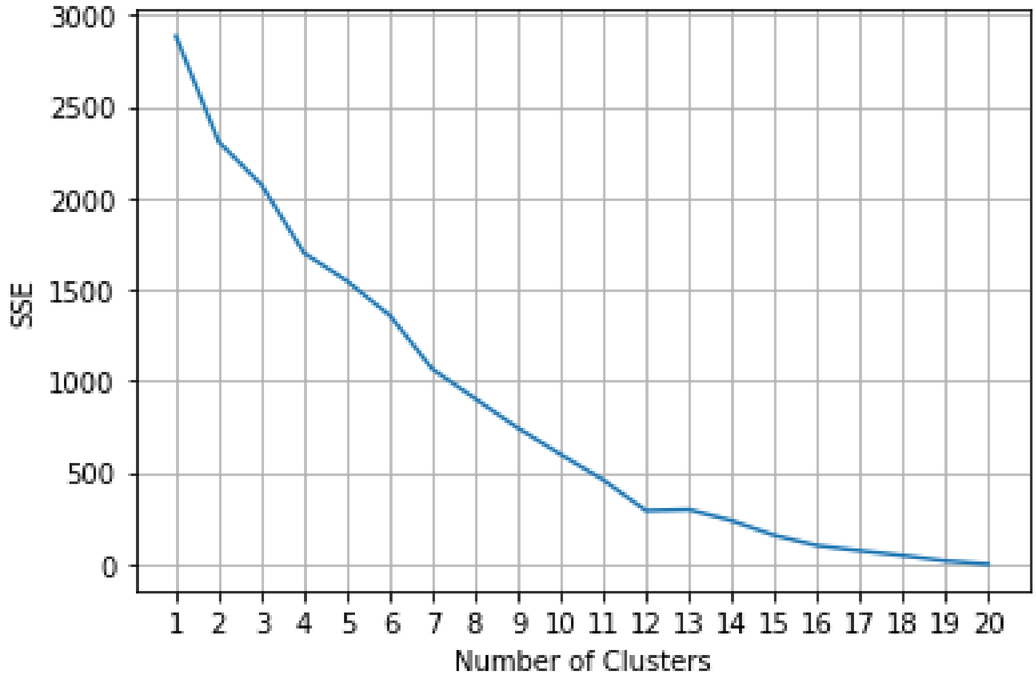


6.1 K-Means for 9 components

K-means algorith for checking proper number of clusters.

[18]:
```
# scree plot
from sklearn.cluster import KMeans


kmeans_kwargs = {"init": "random", "n_init": 10,  "max_iter": 300,␣
 ↪"random_state": 42}
sse = []
for k in range(1, 21):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(scores)
    sse.append(kmeans.inertia_)
```

```
plt.grid()
plt.plot(range(1, 21), sse)
plt.xticks(range(1, 21))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")

plt.savefig("kMenasPCA9-3dimScree.png")
```



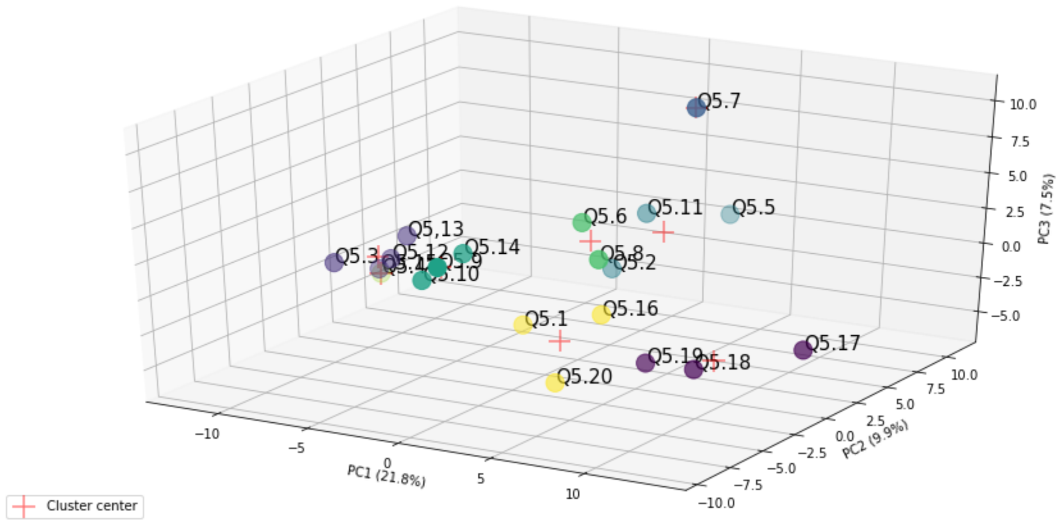
From the Elbow method: It seems that 7-8 clusters is enough.

[19]:
```
# Plot the data with K Means Labels
from sklearn.cluster import KMeans
kmeans = KMeans(8, random_state=0)
labels = kmeans.fit_predict(scores.T[0:3].T)
print("Labels = ", labels)
print("etiquettes = ", df_scores.index)
print("clusters = ", list(zip(labels,df_scores.index)))

from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
from numpy.random import rand
from pylab import figure
```

```python
fig = figure()

fig = plt.figure(figsize=plt.figaspect(0.5)*1.5)
ax = fig.gca(projection='3d')
ax = Axes3D(fig)

for i in range(len(scores[:,0])):
    ax.text(scores[i,0],scores[i,1],scores[i,2],  '%s' % (df_scores.index[i]),␣
 ↪size=15,zorder=1,  color='k')

ax.scatter(scores[:,0],scores[:,1],scores[:,2], c=labels, s=200, cmap='viridis')

plt.grid(True)
ax.set_xlabel("PC1 ({:.1f}%)".format(pca_out.explained_variance_ratio_[0]*100))
ax.set_ylabel("PC2 ({:.1f}%)".format(pca_out.explained_variance_ratio_[1]*100))
ax.set_zlabel("PC3 ({:.1f}%)".format(pca_out.explained_variance_ratio_[2]*100))

centers = kmeans.cluster_centers_
ax.scatter(centers[:, 0], centers[:, 1], centers[:, 2], c='red', marker='+',␣
 ↪s=300, alpha=0.5, label="Cluster center");
ax.legend(loc="lower left")


plt.savefig("kMansPCA9-3dim.png")
```

```
Labels =  [7 3 1 6 3 5 2 5 4 4 3 1 1 4 1 7 0 0 0 7]
etiquettes =  Index(['Q5.1', 'Q5.2', 'Q5.3', 'Q5.4', 'Q5.5', 'Q5.6', 'Q5.7',
'Q5.8', 'Q5.9',
       'Q5.10', 'Q5.11', 'Q5.12', 'Q5,13', 'Q5.14', 'Q5.15', 'Q5.16', 'Q5.17',
       'Q5.18', 'Q5.19', 'Q5.20'],
      dtype='object')
clusters =  [(7, 'Q5.1'), (3, 'Q5.2'), (1, 'Q5.3'), (6, 'Q5.4'), (3, 'Q5.5'),
(5, 'Q5.6'), (2, 'Q5.7'), (5, 'Q5.8'), (4, 'Q5.9'), (4, 'Q5.10'), (3, 'Q5.11'),
(1, 'Q5.12'), (1, 'Q5,13'), (4, 'Q5.14'), (1, 'Q5.15'), (7, 'Q5.16'), (0,
'Q5.17'), (0, 'Q5.18'), (0, 'Q5.19'), (7, 'Q5.20')]
```

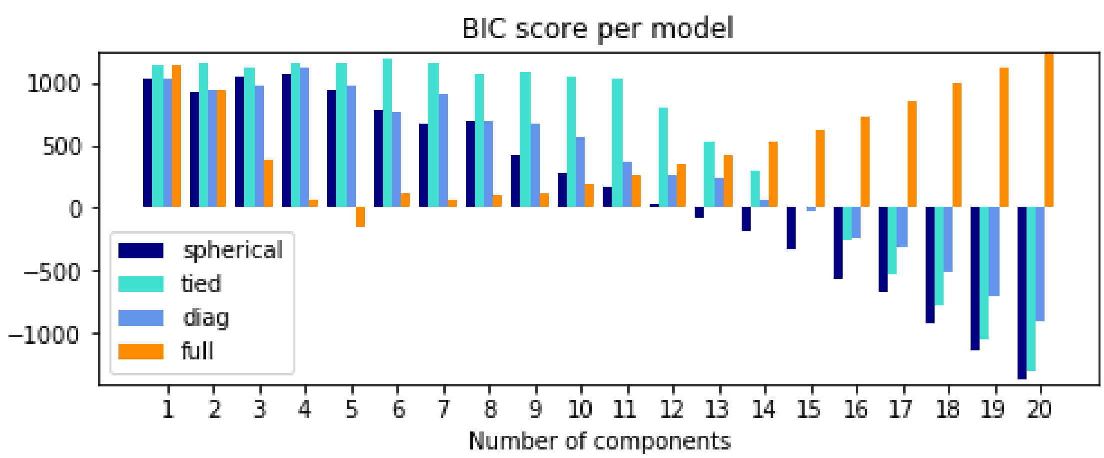
```
<Figure size 432x288 with 0 Axes>
```



6.2 GMM for 8 clusters and 9 PCA components

[20]:
```
# select best GMM based on BIC metric
# adapted from: https://scikit-learn.org/stable/auto_examples/mixture/
↪plot_gmm_selection.html#sphx-glr-auto-examples-mixture-plot-gmm-selection-py

from sklearn import mixture
import itertools
from scipy import linalg

X = scores

lowest_bic = np.infty
bic = []
n_components_range = range(1, 21)
cv_types = ['spherical', 'tied', 'diag', 'full']
for cv_type in cv_types:
    for n_components in n_components_range:
        # Fit a Gaussian mixture with EM
        gmm = mixture.GaussianMixture(n_components=n_components, ␣
↪covariance_type=cv_type)
        gmm.fit(X)
        bic.append(gmm.bic(X))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm

bic = np.array(bic)
```

```
color_iter = itertools.cycle(['navy', 'turquoise', 'cornflowerblue',
                              'darkorange'])
clf = best_gmm
bars = []

# Plot the BIC scores
plt.figure(figsize=(8, 6))
spl = plt.subplot(2, 1, 1)
for i, (cv_type, color) in enumerate(zip(cv_types, color_iter)):
    xpos = np.array(n_components_range) + .2 * (i - 2)
    bars.append(plt.bar(xpos, bic[i * len(n_components_range):
                                  (i + 1) * len(n_components_range)],
                        width=.2, color=color))
plt.xticks(n_components_range)
plt.ylim([bic.min() * 1.01 - .01 * bic.max(), bic.max()])
plt.title('BIC score per model')
xpos = np.mod(bic.argmin(), len(n_components_range)) + .65 +\
    .2 * np.floor(bic.argmin() / len(n_components_range))
spl.set_xlabel('Number of components')
spl.legend([b[0] for b in bars], cv_types)

plt.savefig("BICScorePCA9 - 3dim.png")
```



For 8 clusters the lowest BIC is obtained for 'full' gaussian model.

[21]:
```
from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=8,covariance_type='full', random_state=0).
 ↪fit(X)
#print(gm.means_[:,0])
print("means of gaussian model = \n", gm.means_)
print("weights of each mixture = \n", gm.weights_)
```

```python
print("covariance matrices = \n", gm.covariances_)
print("convergent = ", gm.converged_)
```

```
means of gaussian model =
 [[-7.67407002  0.07335233  1.28224272  2.25375459  0.55719339  6.42375474
   8.88307436  1.46638564  5.88548653]
 [13.47374927  0.7536434  -3.06107357 -0.82665287  2.21879172  2.87395672
   0.91643269  1.4292119  -1.0962763 ]
 [ 5.93635072 -2.31071747 -2.8092091   1.1873574  -2.14151189  1.31757969
  -2.26834818  0.36846761  1.66193902]
 [-0.79167798 -6.4742725   3.0731543  -3.66235051  3.91546991 -1.08834286
   0.25882574 -0.40972652 -0.5636319 ]
 [-9.94569525  0.33181217 -1.41173416  0.83183263 -1.29613961  0.05835093
  -2.53842834 -0.02832066 -0.54517508]
 [ 2.22243122 -0.27057115  4.19216114 10.40013743  2.13056049 -7.23412728
   1.47725931  5.54599871 -0.32367626]
 [ 1.8623905   4.10178915  0.64746276 -0.08777856 -0.30094732 -0.53051669
   1.52091376 -3.10746776 -1.86662613]
 [ 1.07834907  6.27306838  2.01366612 -8.47805494 -1.3976127  -1.64969461
  -0.43070624  6.96433429  2.09143663]]
weights of each mixture =
 [0.05 0.05 0.2  0.15 0.2  0.05 0.25 0.05]
covariance matrices =
 [[[ 1.00000000e-06 -2.81031697e-30 -4.87121609e-29 -8.24359646e-29
   -2.06089911e-29 -2.39813715e-28 -3.29743858e-28 -5.62063395e-29
   -2.24825358e-28]
  [-2.81031697e-30  1.00000000e-06  4.80712114e-31  8.13512809e-31
    2.03378202e-31  2.36658272e-30  3.25405123e-30  5.54667824e-31
    2.21867130e-30]
  [-4.87121609e-29  4.80712114e-31  1.00000000e-06  1.41008887e-29
    3.52522217e-30  4.10207671e-29  5.64035547e-29  9.61424228e-30
    3.84569691e-29]
  [-8.24359646e-29  8.13512809e-31  1.41008887e-29  1.00000000e-06
    5.96576060e-30  6.94197597e-29  9.54521695e-29  1.62702562e-29
    6.50810247e-29]
  [-2.06089911e-29  2.03378202e-31  3.52522217e-30  5.96576060e-30
    1.00000000e-06  1.73549399e-29  2.38630424e-29  4.06756404e-30
    1.62702562e-29]
  [-2.39813715e-28  2.36658272e-30  4.10207671e-29  6.94197597e-29
    1.73549399e-29  1.00000000e-06  2.77679039e-28  4.73316543e-29
    1.89326617e-28]
  [-3.29743858e-28  3.25405123e-30  5.64035547e-29  9.54521695e-29
    2.38630424e-29  2.77679039e-28  1.00000000e-06  6.50810247e-29
    2.60324099e-28]
  [-5.62063395e-29  5.54667824e-31  9.61424228e-30  1.62702562e-29
    4.06756404e-30  4.73316543e-29  6.50810247e-29  1.00000000e-06
    4.43734259e-29]
```

```
 [-2.24825358e-28  2.21867130e-30  3.84569691e-29  6.50810247e-29
   1.62702562e-29  1.89326617e-28  2.60324099e-28  4.43734259e-29
   1.00000000e-06]]

[[ 1.00000000e-06  5.02898827e-29 -2.01159531e-28 -5.69952004e-29
   1.47516989e-28  1.87748895e-28  6.03478592e-29  9.38744477e-29
  -7.37584946e-29]
 [ 5.02898827e-29  1.00000000e-06 -1.10933565e-29 -3.14311767e-30
   8.13512809e-30  1.03537994e-29  3.32800694e-30  5.17689969e-30
  -4.06756404e-30]
 [-2.01159531e-28 -1.10933565e-29  1.00000000e-06  1.25724707e-29
  -3.25405123e-29 -4.14151975e-29 -1.33120278e-29 -2.07075988e-29
   1.62702562e-29]
 [-5.69952004e-29 -3.14311767e-30  1.25724707e-29  1.00000000e-06
  -9.21981183e-30 -1.17343060e-29 -3.77174120e-30 -5.86715298e-30
   4.60990591e-30]
 [ 1.47516989e-28  8.13512809e-30 -3.25405123e-29 -9.21981183e-30
   1.00000000e-06  3.03711449e-29  9.76215370e-30  1.51855724e-29
  -1.19315212e-29]
 [ 1.87748895e-28  1.03537994e-29 -4.14151975e-29 -1.17343060e-29
   3.03711449e-29  1.00000000e-06  1.24245593e-29  1.93270922e-29
  -1.51855724e-29]
 [ 6.03478592e-29  3.32800694e-30 -1.33120278e-29 -3.77174120e-30
   9.76215370e-30  1.24245593e-29  1.00000000e-06  6.21227963e-30
  -4.88107685e-30]
 [ 9.38744477e-29  5.17689969e-30 -2.07075988e-29 -5.86715298e-30
   1.51855724e-29  1.93270922e-29  6.21227963e-30  1.00000000e-06
  -7.59278621e-30]
 [-7.37584946e-29 -4.06756404e-30  1.62702562e-29  4.60990591e-30
  -1.19315212e-29 -1.51855724e-29 -4.88107685e-30 -7.59278621e-30
   1.00000000e-06]]

[[ 6.76575587e+00  7.07410700e-01 -2.82144537e-01 -8.67481080e-01
  -4.00255158e+00 -3.35174362e+00  2.94720880e+00  4.74343735e-01
  -1.94974869e+00]
 [ 7.07410700e-01  1.76811900e-01 -2.51624075e-01  2.94889774e-01
  -1.33027029e-01 -4.94826825e-01 -8.84255934e-02 -2.02400957e-01
   2.87800151e-01]
 [-2.82144537e-01 -2.51624075e-01  1.20490880e+01  2.06059250e+00
  -3.58405617e+00  1.44340273e+01 -2.70240242e+00  5.68873417e+00
  -1.24743627e+01]
 [-8.67481080e-01  2.94889774e-01  2.06059250e+00  2.26323820e+00
   8.08607245e-01  3.34511497e+00 -2.71417606e+00  2.71070321e-01
  -7.48102290e-01]
 [-4.00255158e+00 -1.33027029e-01 -3.58405617e+00  8.08607245e-01
   4.01030625e+00 -2.20992903e+00 -1.91247612e+00 -2.38063556e+00
   5.63527870e+00]
 [-3.35174362e+00 -4.94826825e-01  1.44340273e+01  3.34511497e+00
```

```
 -2.20992903e+00  1.87791555e+01 -5.06023365e+00  6.36651843e+00
 -1.36295587e+01]
 [ 2.94720880e+00 -8.84255934e-02 -2.70240242e+00 -2.71417606e+00
 -1.91247612e+00 -5.06023365e+00  3.83456510e+00 -3.56969957e-01
  6.71860394e-01]
 [ 4.74343735e-01 -2.02400957e-01  5.68873417e+00  2.71070321e-01
 -2.38063556e+00  6.36651843e+00 -3.56969957e-01  2.95824912e+00
 -6.47713988e+00]
 [-1.94974869e+00  2.87800151e-01 -1.24743627e+01 -7.48102290e-01
  5.63527870e+00 -1.36295587e+01  6.71860394e-01 -6.47713988e+00
  1.43426516e+01]]

[[ 2.06623383e+00 -2.93770219e+00  1.58651635e+00  3.87058467e-01
 -6.61082107e-02 -5.99149307e-01 -3.20147265e-02  4.95549783e-01
 -9.64397548e-01]
 [-2.93770219e+00  5.63122523e+00 -1.94168292e+00  1.24588317e+00
  1.18824751e+00  8.33725205e-01  1.30205460e+00 -2.27233490e+00
  1.69754176e+00]
 [ 1.58651635e+00 -1.94168292e+00  1.28595235e+00  6.84929150e-01
  1.85451470e-01 -4.63957733e-01  2.46660010e-01  4.20703033e-02
 -6.70037067e-01]
 [ 3.87058467e-01  1.24588317e+00  6.84929150e-01  2.29066409e+00
  1.33893969e+00 -1.34619858e-01  1.54572973e+00 -1.84325687e+00
  2.22413757e-01]
 [-6.61082107e-02  1.18824751e+00  1.85451470e-01  1.33893969e+00
  8.25355845e-01  5.53307976e-03  9.46351824e-01 -1.19533786e+00
  2.76409855e-01]
 [-5.99149307e-01  8.33725205e-01 -4.63957733e-01 -1.34619858e-01
  5.53307976e-03  1.73963298e-01 -6.37537605e-03 -1.24158096e-01
  2.75580668e-01]
 [-3.20147265e-02  1.30205460e+00  2.46660010e-01  1.54572973e+00
  9.46351824e-01 -6.37537605e-03  1.08601817e+00 -1.36208037e+00
  2.96913108e-01]
 [ 4.95549783e-01 -2.27233490e+00  4.20703033e-02 -1.84325687e+00
 -1.19533786e+00 -1.24158096e-01 -1.36208037e+00  1.80873443e+00
 -5.83107804e-01]
 [-9.64397548e-01  1.69754176e+00 -6.70037067e-01  2.22413757e-01
  2.76409855e-01  2.75580668e-01  2.96913108e-01 -5.83107804e-01
  5.23369289e-01]]

[[ 1.87687635e+00 -2.53976713e+00  1.92448227e+00  2.15088656e+00
 -2.08001865e+00  1.60788037e+00  1.08961298e+00  2.82655937e-01
 -3.49286476e-01]
 [-2.53976713e+00  5.86171158e+00 -3.65461845e+00 -3.48713332e+00
  1.76363254e+00 -1.61424130e+00 -1.20095682e+00  3.86376122e+00
 -2.33205559e-01]
 [ 1.92448227e+00 -3.65461845e+00  2.45471319e+00  2.61261486e+00
 -1.36088755e+00  1.48161518e+00  7.43982228e-01 -1.51756176e+00
```

```
 -1.72673815e-01]
[ 2.15088656e+00 -3.48713332e+00  2.61261486e+00  3.54090844e+00
 -2.45289833e-01  2.16357768e+00 -3.36144074e-01 -4.94815694e-01
 -9.49956696e-01]
[-2.08001865e+00  1.76363254e+00 -1.36088755e+00 -2.45289833e-01
  6.55913945e+00 -1.11118039e+00 -4.38296481e+00 -1.76971924e+00
 -7.50143197e-01]
[ 1.60788037e+00 -1.61424130e+00  1.48161518e+00  2.16357768e+00
 -1.11118039e+00  1.72745064e+00  2.61133516e-01  1.31782473e+00
 -8.09980786e-01]
[ 1.08961298e+00 -1.20095682e+00  7.43982228e-01 -3.36144074e-01
 -4.38296481e+00  2.61133516e-01  3.12350239e+00  3.34005287e-01
  8.79034217e-01]
[ 2.82655937e-01  3.86376122e+00 -1.51756176e+00 -4.94815694e-01
 -1.76971924e+00  1.31782473e+00  3.34005287e-01  7.51691843e+00
 -1.43449801e+00]
[-3.49286476e-01 -2.33205559e-01 -1.72673815e-01 -9.49956696e-01
 -7.50143197e-01 -8.09980786e-01  8.79034217e-01 -1.43449801e+00
  8.18780649e-01]]

[[ 1.00000000e-06 -2.98288030e-30  4.33873498e-29  1.12807109e-28
   2.38630424e-29 -7.80972296e-29  1.62702562e-29  6.07422897e-29
  -3.52522217e-30]
 [-2.98288030e-30  1.00000000e-06 -5.42341872e-30 -1.41008887e-29
  -2.98288030e-30  9.76215370e-30 -2.03378202e-30 -7.59278621e-30
   4.40652771e-31]
 [ 4.33873498e-29 -5.42341872e-30  1.00000000e-06  2.05103835e-28
   4.33873498e-29 -1.41994963e-28  2.95822839e-29  1.10440527e-28
  -6.40949485e-30]
 [ 1.12807109e-28 -1.41008887e-29  2.05103835e-28  1.00000000e-06
   1.12807109e-28 -3.69186904e-28  7.69139383e-29  2.87145370e-28
  -1.66646866e-29]
 [ 2.38630424e-29 -2.98288030e-30  4.33873498e-29  1.12807109e-28
   1.00000000e-06 -7.80972296e-29  1.62702562e-29  6.07422897e-29
  -3.52522217e-30]
 [-7.80972296e-29  9.76215370e-30 -1.41994963e-28 -3.69186904e-28
  -7.80972296e-29  1.00000000e-06 -5.32481111e-29 -1.98792948e-28
   1.15370907e-29]
 [ 1.62702562e-29 -2.03378202e-30  2.95822839e-29  7.69139383e-29
   1.62702562e-29 -5.32481111e-29  1.00000000e-06  4.14151975e-29
  -2.40356057e-30]
 [ 6.07422897e-29 -7.59278621e-30  1.10440527e-28  2.87145370e-28
   6.07422897e-29 -1.98792948e-28  4.14151975e-29  1.00000000e-06
  -8.97329280e-30]
 [-3.52522217e-30  4.40652771e-31 -6.40949485e-30 -1.66646866e-29
  -3.52522217e-30  1.15370907e-29 -2.40356057e-30 -8.97329280e-30
   1.00000000e-06]]
```

```
[[ 3.86468807e+00  2.97697096e+00  8.01596553e+00  1.03700362e+00
   1.41399977e+00  2.11855316e+00 -4.74384042e+00 -2.85330754e+00
   5.07625343e+00]
 [ 2.97697096e+00  1.60881552e+01  6.89688768e+00  1.20480495e+00
   1.23383456e+01  2.70633734e+00 -8.15029451e+00  1.60317153e+00
   5.05921292e+00]
 [ 8.01596553e+00  6.89688768e+00  2.62321796e+01 -1.21296843e-01
  -9.46475705e+00 -2.35047587e-01 -1.07900420e+01 -6.75440247e+00
   1.31353812e+01]
 [ 1.03700362e+00  1.20480495e+00 -1.21296843e-01  2.07688989e+00
   3.53443527e+00  1.77491587e+00 -1.59661154e+00 -3.71021903e-01
  -7.85628242e-03]
 [ 1.41399977e+00  1.23383456e+01 -9.46475705e+00  3.53443527e+00
   2.73843882e+01  7.99875448e+00 -4.34654265e+00  3.45259626e+00
  -4.76491906e-01]
 [ 2.11855316e+00  2.70633734e+00 -2.35047587e-01  1.77491587e+00
   7.99875448e+00  3.54171394e+00 -2.61559670e+00 -7.59705187e-01
   1.59210052e+00]
 [-4.74384042e+00 -8.15029451e+00 -1.07900420e+01 -1.59661154e+00
  -4.34654265e+00 -2.61559670e+00  7.44825598e+00  2.33076035e+00
  -6.56310243e+00]
 [-2.85330754e+00  1.60317153e+00 -6.75440247e+00 -3.71021903e-01
   3.45259626e+00 -7.59705187e-01  2.33076035e+00  3.26692151e+00
  -3.72890233e+00]
 [ 5.07625343e+00  5.05921292e+00  1.31353812e+01 -7.85628242e-03
  -4.76491906e-01  1.59210052e+00 -6.56310243e+00 -3.72890233e+00
   7.94990745e+00]]

[[ 1.00000000e-06  3.47098798e-29  1.08468374e-29 -4.77260848e-29
  -7.59278621e-30 -8.67746996e-30 -2.30495296e-30  3.68792473e-29
   1.08468374e-29]
 [ 3.47098798e-29  1.00000000e-06  6.31088724e-29 -2.77679039e-28
  -4.41762107e-29 -5.04870979e-29 -1.34106354e-29  2.14570166e-28
   6.31088724e-29]
 [ 1.08468374e-29  6.31088724e-29  1.00000000e-06 -8.67746996e-29
  -1.38050658e-29 -1.57772181e-29 -4.19082356e-30  6.70531769e-29
   1.97215226e-29]
 [-4.77260848e-29 -2.77679039e-28 -8.67746996e-29  1.00000000e-06
   6.07422897e-29  6.94197597e-29  1.84396237e-29 -2.95033979e-28
  -8.67746996e-29]
 [-7.59278621e-30 -4.41762107e-29 -1.38050658e-29  6.07422897e-29
   1.00000000e-06  1.10440527e-29  2.93357649e-30 -4.69372239e-29
  -1.38050658e-29]
 [-8.67746996e-30 -5.04870979e-29 -1.57772181e-29  6.94197597e-29
   1.10440527e-29  1.00000000e-06  3.35265885e-30 -5.36425416e-29
  -1.57772181e-29]
 [-2.30495296e-30 -1.34106354e-29 -4.19082356e-30  1.84396237e-29
   2.93357649e-30  3.35265885e-30  1.00000000e-06 -1.42488001e-29
```

```
 -4.19082356e-30]
 [ 3.68792473e-29  2.14570166e-28  6.70531769e-29 -2.95033979e-28
  -4.69372239e-29 -5.36425416e-29 -1.42488001e-29  1.00000000e-06
   6.70531769e-29]
 [ 1.08468374e-29  6.31088724e-29  1.97215226e-29 -8.67746996e-29
  -1.38050658e-29 -1.57772181e-29 -4.19082356e-30  6.70531769e-29
   1.00000000e-06]]]
convergent =  True
```

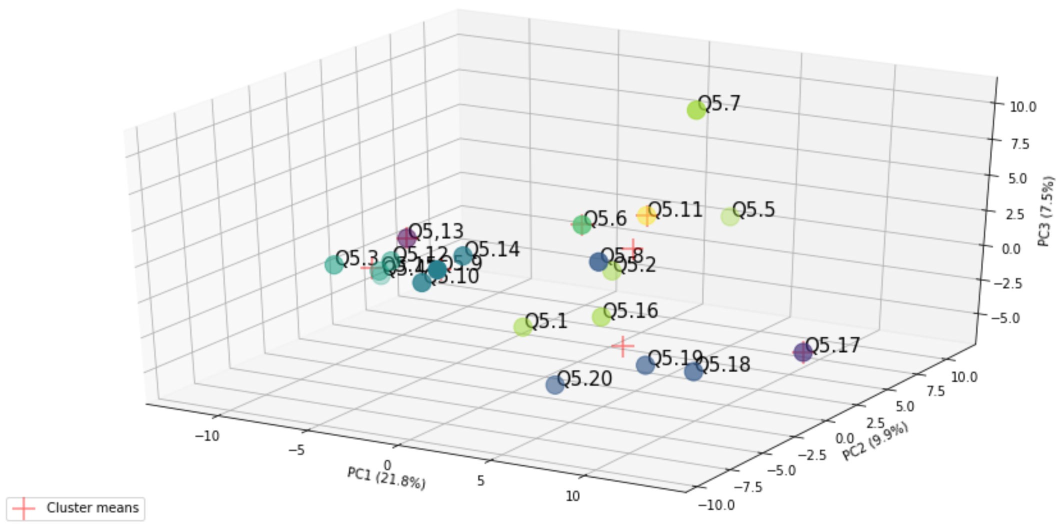
[22]:

```python
labels = gm.fit_predict(scores)
print("Labels = ", labels)
print("etiquettes = ", df_scores.index)
print("clusters = ", list(zip(labels, df_scores.index)))

from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
from numpy.random import rand
from pylab import figure

fig = figure()

fig = plt.figure(figsize=plt.figaspect(0.5)*1.5)
ax = fig.gca(projection='3d')
ax = Axes3D(fig)

ax.scatter(scores[:,0],scores[:,1],scores[:,2], c=labels, s=200, cmap='viridis')

for i in range(len(scores[:,0])):
    ax.text(scores[i,0],scores[i,1],scores[i,2],  '%s' % (df_scores.index[i]),␣
 ↪size=15,zorder=1,  color='k')

centers = gm.means_
ax.scatter(centers[:, 0], centers[:, 1], centers[:, 2], c='red', marker='+',␣
 ↪s=300, alpha=0.5, label="Cluster means");
ax.legend(loc="lower left")


plt.grid(True)
ax.set_xlabel("PC1 ({:.1f}%)".format(pca_out.explained_variance_ratio_[0]*100))
ax.set_ylabel("PC2 ({:.1f}%)".format(pca_out.explained_variance_ratio_[1]*100))
ax.set_zlabel("PC3 ({:.1f}%)".format(pca_out.explained_variance_ratio_[2]*100))


plt.savefig("GaussianPCA9-3dim.png")
```

```
Labels =  [6 6 4 4 6 5 6 2 3 3 7 4 0 3 4 6 1 2 2 2]
etiquettes =  Index(['Q5.1', 'Q5.2', 'Q5.3', 'Q5.4', 'Q5.5', 'Q5.6', 'Q5.7',
'Q5.8', 'Q5.9',
```

```
       'Q5.10', 'Q5.11', 'Q5.12', 'Q5,13', 'Q5.14', 'Q5.15', 'Q5.16', 'Q5.17',
       'Q5.18', 'Q5.19', 'Q5.20'],
      dtype='object')
clusters =  [(6, 'Q5.1'), (6, 'Q5.2'), (4, 'Q5.3'), (4, 'Q5.4'), (6, 'Q5.5'),
(5, 'Q5.6'), (6, 'Q5.7'), (2, 'Q5.8'), (3, 'Q5.9'), (3, 'Q5.10'), (7, 'Q5.11'),
(4, 'Q5.12'), (0, 'Q5,13'), (3, 'Q5.14'), (4, 'Q5.15'), (6, 'Q5.16'), (1,
'Q5.17'), (2, 'Q5.18'), (2, 'Q5.19'), (2, 'Q5.20')]
```

```
<Figure size 432x288 with 0 Axes>
```



[ ]: