43

DA_Homework_06_r09546042_TerryYang

April 11, 2021

1 Q1

import package

```
[1]: import numpy as np
     import pandas as pd
```

read AutoMPG data

```
[2]: data = pd.read_csv(r"C:\Users\TerryYang\pythonwork\pythonwork\Data Analytics␣
      ↪Homework\DA_Demo.csv")
     X=data.drop(" car name",axis = 1).to_numpy()
```

define Factor Analysis function

```
[3]: def FA(dataMat, factor_number):
         meanVals = np.mean(dataMat, axis=0)
         meanRemoved = dataMat - meanVals
         covMat = np.cov(meanRemoved, rowvar=0)
         eigVal, eigVect = np.linalg.eig(np.mat(covMat))
         selected_eigenvalue = eigVal[:factor_number]
         eigenvalues_diagonal = np.zeros((eigVal.shape[0], eigVal.shape[0]), float)
         np.fill_diagonal(eigenvalues_diagonal, eigVal)
         eigenvalues_diagonal_total_sqrt = np.sqrt(eigenvalues_diagonal)
         All_T =   eigenvalues_diagonal_total_sqrt @ eigVect
         All = All_T.T
         X_variance = np.diag(np.diag(All_T @ All))
         eigenvalues_diagonal = np.zeros((factor_number, factor_number), float)
         np.fill_diagonal(eigenvalues_diagonal, eigVal[:factor_number])
         eigenvalues_diagonal_sqrt = np.sqrt(eigenvalues_diagonal)
         A_T = eigVect[:, :factor_number] @ eigenvalues_diagonal_sqrt
         A = A_T.T
         Psi = X_variance - A_T @ A
         Psi_inverse = np.linalg.inv(Psi)
         inner = np.linalg.inv(A @ (Psi_inverse) @ (A_T))
         F = dataMat @ Psi_inverse @ (A_T) @ (inner)
         communality_vector = A_T @ A
         return F, A,communality_vector, Psi,eigVal, eigVect, selected_eigenvalue
```

should be a vector, not matrix
Just get the diagonal elements

calling function

[4]:
```
F, A,communality_vector, Psi,eigVal, eigVect, selected_eigenvalue = FA(X, 2)
```

find factor contribution percentage

[5]:
```
Total_eigenvalues = eigVal.sum()
for i in range(0,len(selected_eigenvalue)):
    print("Factor",i+1,"contribute",format(selected_eigenvalue[i]*100/
 ↪Total_eigenvalues,'2.2f'),"%")
```

```
Factor 1 contribute 99.75 %
Factor 2 contribute 0.21 %
```

compare to the PCA model, the explanation is almost alike, as the first factor(or component) has 99.75% explanation on variance

print out Factor matrix, Loading matrix, Communality Vector and Uniqueness Vector

[6]:
```
print("\nFactor matrix:\n",F)
print("\nLoading matrix:\n",A)
print("\nCommunality Vector:\n",communality_vector)
print("\nUniqueness Vector:\n",Psi)
```

```
Factor matrix:
 [[ 4.11130246  2.56228537]
 [ 4.33721447  1.9724159 ]
 [ 4.03424091  2.04065671]
 [ 4.02873985  2.40165545]
 [ 4.046891    2.51429333]
 [ 5.10082817  1.84986084]
 [ 5.11978803  1.20602863]
 [ 5.06897722  1.45175908]
 [ 5.20236803  1.39365311]
 [ 4.5254432   1.37339716]
 [ 4.19121337  0.69306769]
 [ 4.23825085  1.98262172]
 [ 4.42314157  0.87559924]
 [ 3.64886693 -2.74096123]
 [ 2.76949913  4.2357598 ]
 [ 3.31663549  3.410327  ]
 [ 3.24835279  3.19962533]
 [ 3.03139445  2.6082012 ]
 [ 2.48633615  3.9188881 ]
 [ 2.14356974  3.05161488]
 [ 3.11702854  5.24978455]
 [ 2.83586279  4.57870528]
 [ 2.77168215  4.48308066]
 [ 2.61076533  3.57936837]
```

```
[ 3.10208235  2.8178054 ]
[ 5.40897057  4.50315445]
[ 5.1238498   5.18236971]
[ 5.13252015  4.89959836]
[ 5.53636305  6.36829384]
[ 2.48633619  3.91888912]
[ 2.64803537  3.19342145]
[ 2.60243038  3.79109518]
[ 3.09072463  1.8914559 ]
[ 4.02373385  4.55714732]
[ 3.89966057  3.56151597]
[ 3.8681887   3.4905834 ]
[ 3.84949391  3.91094419]
[ 4.93587547  3.56574956]
[ 5.23906552  3.02046941]
[ 4.87206357  3.38193701]
[ 4.79997105  4.07866452]
[ 5.80632805  4.98099814]
[ 5.56617977  3.89643102]
[ 6.02335769  5.10785605]
[ 3.47514473  2.206311  ]
[ 2.814886    3.65678822]
[ 3.84513105  3.41637909]
[ 3.67907621  2.98724885]
[ 2.5943382   3.53770694]
[ 2.48098194  3.3927169 ]
[ 2.41855022  4.2407435 ]
[ 2.40947758  3.96863347]
[ 2.06811513  3.52804755]
[ 1.88267646  3.00338408]
[ 2.14257956  3.03399029]
[ 2.28222016  3.55595482]
[ 2.66044122  3.94551363]
[ 2.48167165  3.90165098]
[ 2.62979135  4.33717923]
[ 2.81510629  3.6381392 ]
[ 2.60130117  3.55627758]
[ 5.01128818  3.76646001]
[ 5.1474105   2.77653979]
[ 4.84521903  4.19909972]
[ 4.84305838  3.30473357]
[ 4.30602616  3.13964211]
[ 5.43972805  2.74116259]
[ 5.27569152  4.48087837]
[ 5.22238364  4.33365492]
[ 5.19052132  2.8752495 ]
[ 2.71457915  5.24166414]
[ 4.56126977  3.81898339]
```

```
[ 4.80045976  4.39649693]
[ 5.02725971  5.12361171]
[ 4.77792808  4.02001367]
[ 3.42173089  5.73883944]
[ 2.93169125  4.47314877]
[ 3.47465358  5.9333259 ]
[ 2.55441369  4.14727338]
[ 2.79737377  4.07811359]
[ 2.66969765  4.4026647 ]
[ 2.926       4.46232775]
[ 2.52582984  4.00573169]
[ 2.45153178  3.82628909]
[ 4.80953716  3.21882849]
[ 4.3060262   3.13964313]
[ 4.67922881  2.90408828]
[ 4.73485289  4.3485619 ]
[ 4.42986856  3.09363789]
[ 5.80970847  3.73655813]
[ 5.23876051  3.04639462]
[ 5.11460512  4.02210562]
[ 4.96355966  4.5140772 ]
[ 5.55974156  2.75795997]
[ 5.8126314   3.0178757 ]
[ 4.48728505  2.0928269 ]
[ 3.65479261  3.57523408]
[ 3.84049245  3.40408089]
[ 3.45154745  2.8518265 ]
[ 3.54217468  2.62292014]
[ 3.39900979  3.62957503]
[ 2.27699261  3.40672775]
[ 5.85714585  4.69224605]
[ 5.75177474  4.39360532]
[ 5.4536678   4.6702125 ]
[ 5.27251569  4.44568418]
[ 3.27055633  2.37010682]
[ 2.65920847  4.3790556 ]
[ 2.80676505  3.63518391]
[ 2.77688759  4.39067028]
[ 2.47549349  4.61282674]
[ 2.69874677  3.8167276 ]
[ 2.89174283  3.42033786]
[ 2.64313208  4.30724696]
[ 4.78828714  3.194345  ]
[ 5.02393985  2.38908736]
[ 2.17654529  3.91423844]
[ 2.52140663  3.5163676 ]
[ 3.01323498  4.86189191]
[ 3.34631895  5.53814806]
```

```
[ 3.99131379  1.92640978]
[ 3.1049729   4.89792549]
[ 3.28073543  4.41276005]
[ 4.30375182  1.86731435]
[ 3.62872878  4.24097638]
[ 3.4004989   2.71596267]
[ 3.90778309  3.58315944]
[ 2.27464913  3.86095596]
[ 2.8622718   4.25726255]
[ 2.14120892  3.72261843]
[ 2.97038747  4.06741766]
[ 4.42406986  4.95724711]
[ 4.25247864  4.27524181]
[ 4.22560858  5.09445112]
[ 4.84974853  4.65114054]
[ 5.50418962  5.09438435]
[ 5.21880189  5.19338541]
[ 5.42636685  6.18582983]
[ 4.98474312  4.94609721]
[ 2.58967833  4.1724806 ]
[ 2.28972921  3.9010382 ]
[ 2.68344727  4.45419038]
[ 1.92481232  3.02636135]
[ 2.33664365  3.92504652]
[ 2.4793659   4.10205117]
[ 2.45964329  4.04957499]
[ 2.62350276  3.78807275]
[ 2.9062771   4.40984507]
[ 2.79079704  4.4287449 ]
[ 2.33265677  4.0152967 ]
[ 3.82057804  4.02715421]
[ 4.05054749  3.9577645 ]
[ 4.01882258  3.90868542]
[ 3.7009281   3.0625924 ]
[ 5.47568434  3.65557525]
[ 5.20363772  4.29980988]
[ 5.26637062  5.3200048 ]
[ 5.4555807   4.94030624]
[ 4.56763642  5.83849357]
[ 4.55871465  5.31027989]
[ 4.36617849  4.57786794]
[ 4.42504174  5.63596864]
[ 3.56058294  3.15814844]
[ 3.7762275   2.89926363]
[ 3.72190324  1.66113193]
[ 2.53374655  4.0589957 ]
[ 3.08302416  4.35864622]
[ 3.41558147  2.75610557]
```

```
[ 3.02843496  4.21872634]
[ 3.15540658  4.69832128]
[ 2.59301758  4.40884069]
[ 2.97110509  4.60924514]
[ 3.48795084  4.58854966]
[ 2.26118784  3.52671317]
[ 3.76003792  3.68357619]
[ 3.14337083  5.17715635]
[ 3.44913985  5.86431734]
[ 3.43548229  5.79040875]
[ 3.11779585  4.92670229]
[ 2.09637929  3.07947579]
[ 2.87526145  4.68786124]
[ 2.5934131   3.7016209 ]
[ 3.0053996   4.14239707]
[ 2.6313974   4.28781155]
[ 2.57038735  4.04063069]
[ 4.93603543  4.80028725]
[ 4.90902954  4.36892633]
[ 4.64211802  4.06625752]
[ 4.94282292  3.57132483]
[ 3.78467269  3.92623912]
[ 3.9275658   3.63042919]
[ 3.52442977  3.92471182]
[ 3.6138538   3.29451991]
[ 2.37395099  3.98035093]
[ 2.52559062  4.02658845]
[ 2.26118801  3.5267176 ]
[ 2.09637925  3.0794751 ]
[ 4.26963582  5.21699436]
[ 4.1836436   4.34093984]
[ 4.26640502  4.52692375]
[ 3.74296971  2.93522859]
[ 2.13226757  2.99468654]
[ 2.32195846  3.82265114]
[ 2.51518374  4.00959813]
[ 2.99703298  4.14149178]
[ 3.67467134  6.18127577]
[ 4.61898093  3.59697037]
[ 3.81228436  6.83089497]
[ 3.42326988  4.80711338]
[ 4.45772332  7.22551707]
[ 5.13445231  4.07822776]
[ 4.75696127  3.1109594 ]
[ 4.53521429  3.82472153]
[ 4.40434542  3.02573407]
[ 2.38762265  3.65078282]
[ 2.51726492  3.6341437 ]
```

```
[ 2.12951462  3.48430177]
[ 2.68727723  3.77439973]
[ 2.26974958  3.68369606]
[ 4.54743088  3.76067127]
[ 4.7493463   5.54294716]
[ 4.85095923  4.21973033]
[ 5.02829712  5.13705176]
[ 4.12138082  4.1409485 ]
[ 4.00835853  4.35527103]
[ 4.24527176  5.15215079]
[ 4.12703702  4.16887102]
[ 4.95603894  2.26185343]
[ 4.88488776  3.42469766]
[ 5.07798236  2.57572924]
[ 5.0820109   3.94500507]
[ 2.26577717  3.34257194]
[ 3.20185237  4.37441291]
[ 2.64280649  4.34929193]
[ 3.2176793   4.71059743]
[ 2.39452168  3.67446718]
[ 2.42260993  3.72782482]
[ 2.31785289  3.49295947]
[ 2.55582563  4.11452449]
[ 3.2882687   4.72785937]
[ 3.03536049  4.71263886]
[ 3.16865788  6.16798934]
[ 2.3166138   3.6978651 ]
[ 2.1033476   2.89626694]
[ 2.31492875  4.01103603]
[ 2.41477525  4.06970743]
[ 2.10226511  3.08764318]
[ 3.94300835  3.39684152]
[ 4.38101885  2.97433159]
[ 4.18726426  2.88900557]
[ 4.13598163  4.69497335]
[ 3.69050918  4.35177263]
[ 3.46994828  3.77540848]
[ 3.17706001  4.60355391]
[ 4.01323244  4.5345758 ]
[ 3.75887796  3.68049481]
[ 3.95614901  4.21630062]
[ 3.59176938  4.09964675]
[ 4.23379271  5.11092461]
[ 3.99503562  3.57933069]
[ 4.01954268  2.35571817]
[ 4.03229302  4.35477399]
[ 3.76379196  1.76191764]
[ 4.78128648  4.03964403]
```

```
[ 2.51524273  3.99041184]
[ 2.99064727  4.26089993]
[ 2.6868558   3.85268883]
[ 2.60335206  4.02958687]
[ 2.93843891  4.12195746]
[ 3.20858341  4.24001265]
[ 3.3352393   4.73265226]
[ 2.80867642  4.17691517]
[ 3.30357307  5.16518205]
[ 3.66812784  5.25278432]
[ 3.26166167  5.30963466]
[ 3.98147977  6.07824523]
[ 2.32254675  3.71579644]
[ 2.4920389   3.92865684]
[ 3.79964461  3.7890709 ]
[ 3.49895487  3.85264632]
[ 3.37429453  5.12851966]
[ 3.82268942  3.85034393]
[ 3.93214057  4.30805211]
[ 4.50084193  3.65275801]
[ 4.36697277  3.37799747]
[ 4.64100156  2.78300917]
[ 4.49117773  3.27289417]
[ 5.11094505  4.04243642]
[ 4.75591051  3.08457747]
[ 4.22264868  3.93724361]
[ 4.62504443  2.48623439]
[ 2.24713348  3.51507269]
[ 2.30463766  3.75504281]
[ 2.23694109  3.23685706]
[ 3.11620925  4.95992457]
[ 4.12290215  5.9784956 ]
[ 4.5768906   2.65317137]
[ 3.72229498  6.04617458]
[ 4.00657948  3.58751947]
[ 2.56848424  3.94211354]
[ 2.51047569  3.78775064]
[ 2.35670487  3.92050973]
[ 2.48523825  4.0972723 ]
[ 3.12066322  4.15619315]
[ 3.03714152  3.31577972]
[ 3.15896312  3.64002991]
[ 2.98839902  3.80413198]
[ 2.50257712  3.94812433]
[ 2.29689088  3.65933497]
[ 2.4746597   3.88024936]
[ 2.35568711  3.89092912]
[ 3.12994517  4.18090539]
```

```
[ 3.35109123  5.06677756]
[ 3.50701107  5.1845139 ]
[ 3.95626184  4.39366521]
[ 2.55350654  4.10838083]
[ 3.1657766   4.7323705 ]
[ 2.96750062  4.59635404]
[ 2.84226112  4.27164913]
[ 2.64439255  4.05833692]
[ 2.46126591  4.17194788]
[ 3.27239258  4.40980655]
[ 2.46112246  4.19841612]
[ 2.4326335   4.00665429]
[ 2.72268366  4.77864773]
[ 3.44091583  5.8375707 ]
[ 3.79257759  6.10331065]
[ 2.16035861  3.23473668]
[ 2.50348531  3.98704377]
[ 2.15420887  3.27739814]
[ 3.40208888  4.40305974]
[ 2.81903291  5.51645085]
[ 2.91921831  4.40025617]
[ 2.67321767  4.16511937]
[ 2.90944475  4.02973623]
[ 3.07998344  4.05434721]
[ 3.0633987   3.86747245]
[ 3.18790676  3.7224057 ]
[ 2.78762231  3.7054656 ]
[ 2.04829991  3.26812827]
[ 2.18860685  3.44731809]
[ 2.0544135   3.22858848]
[ 2.41066961  3.74001438]
[ 2.30449631  3.78156248]
[ 2.39205028  3.91043993]
[ 2.31699874  3.65059665]
[ 2.58580291  3.99571821]
[ 2.38758452  3.65385742]
[ 2.77625301  4.68835964]
[ 2.55693193  3.90710792]
[ 2.58043693  3.91493933]
[ 2.74301028  4.32083138]
[ 3.05235405  4.82225262]
[ 3.07538713  4.88457416]
[ 3.76880971  6.16028587]
[ 3.68813876  5.84146863]
[ 3.39029233  4.38878821]
[ 3.42197175  5.05911133]
[ 3.99681702  4.31919505]
[ 4.37361333  2.13356084]
```

```
 [ 3.58020447  4.06566848]
 [ 4.05365569  4.65819173]
 [ 3.03959831  4.98901865]
 [ 3.08020443  5.09707592]
 [ 2.79595624  4.34054552]
 [ 3.00475342  4.89943535]
 [ 2.95005182  4.13781667]
 [ 3.1960774   4.35693971]
 [ 3.34533799  5.0471651 ]
 [ 2.31329179  3.25868181]
 [ 2.36340812  3.77414865]
 [ 2.29959697  3.60430737]
 [ 2.48138525  3.71781045]
 [ 2.48046201  3.89572125]
 [ 2.52446291  3.40327544]
 [ 2.57463605  3.89950287]
 [ 2.62112883  4.0017941 ]
 [ 2.29378195  3.58986047]
 [ 2.29378244  3.5898724 ]
 [ 2.32858864  3.68251604]
 [ 3.44430732  4.19022701]
 [ 3.53690995  2.29016751]
 [ 3.02279198  3.75940086]
 [ 3.32407199  2.49971235]
 [ 3.11392358  4.31962436]
 [ 2.77021949  3.65915075]
 [ 3.44551883  5.02081284]
 [ 3.2582495   4.82178318]
 [ 2.48590384  3.95638947]
 [ 2.68320388  3.42753909]
 [ 3.06384652  4.84851894]
 [ 3.17395788  5.16521346]]

Loading matrix:
 [[-6.49968979e+00 -4.71932579e-01 -1.14396203e+00 -1.15757873e+00
   8.49389535e+02  3.33426322e+01  9.78372906e+01  1.53387616e+00]
 [ 6.84080607e-01  1.26349839e-01  9.32021530e-01  1.35543475e+00
   4.70363212e+00 -1.16096732e+01 -3.67976506e+01 -5.18450498e-01]]

Communality Vector:
```

Should be vector

```
 [[ 4.27139337e+01  3.15384884e+00  8.07297620e+00  8.45112931e+00
  -5.51755083e+03 -2.24658718e+02 -6.61084598e+02 -1.03243811e+01]
 [ 3.15384884e+00  2.38684641e-01  6.57633723e-01  7.17558081e-01
  -4.00260291e+02 -1.72023548e+01 -5.08219821e+01 -7.89392267e-01]
 [ 8.07297620e+00  6.57633723e-01  2.17731327e+00  2.58752049e+00
  -9.67285493e+02 -4.89631707e+01 -1.46218348e+02 -2.23790311e+00]
 [ 8.45112931e+00  7.17558081e-01  2.58752049e+00  3.17719189e+00
  -9.76859796e+02 -5.43328765e+01 -1.63131181e+02 -2.47830824e+00]
```

```
[-5.51755083e+03 -4.00260291e+02 -9.67285493e+02 -9.76859796e+02
  7.21484707e+05  2.82662752e+04  8.29288882e+04  1.30041975e+03]
[-2.24658718e+02 -1.72023548e+01 -4.89631707e+01 -5.43328765e+01
  2.82662752e+04  1.24651563e+03  3.68936149e+03  5.71625094e+01]
[-6.61084598e+02 -5.08219821e+01 -1.46218348e+02 -1.63131181e+02
  8.29288882e+04  3.68936149e+03  1.09262025e+04  1.69148047e+02]
[-1.03243811e+01 -7.89392267e-01 -2.23790311e+00 -2.47830824e+00
  1.30041975e+03  5.71625094e+01  1.69148047e+02  2.62156698e+00]]

Uniqueness Vector:
```

should be vector

```
[[ 7.32151205e+05 -3.15384884e+00 -8.07297620e+00 -8.45112931e+00
  5.51755083e+03  2.24658718e+02  6.61084598e+02  1.03243811e+01]
[-3.15384884e+00  1.51419566e+03 -6.57633723e-01 -7.17558081e-01
  4.00260291e+02  1.72023548e+01  5.08219821e+01  7.89392267e-01]
[-8.07297620e+00 -6.57633723e-01  2.59495868e+02 -2.58752049e+00
  9.67285493e+02  4.89631707e+01  1.46218348e+02  2.23790311e+00]
[-8.45112931e+00 -7.17558081e-01 -2.58752049e+00  2.00797674e+01
  9.76859796e+02  5.43328765e+01  1.63131181e+02  2.47830824e+00]
[ 5.51755083e+03  4.00260291e+02  9.67285493e+02  9.76859796e+02
 -7.21479163e+05 -2.82662752e+04 -8.29288882e+04 -1.30041975e+03]
[ 2.24658718e+02  1.72023548e+01  4.89631707e+01  5.43328765e+01
 -2.82662752e+04 -1.24365843e+03 -3.68936149e+03 -5.71625094e+01]
[ 6.61084598e+02  5.08219821e+01  1.46218348e+02  1.63131181e+02
 -8.29288882e+04 -3.68936149e+03 -1.09258421e+04 -1.69148047e+02]
[ 1.03243811e+01  7.89392267e-01  2.23790311e+00  2.47830824e+00
 -1.30041975e+03 -5.71625094e+01 -1.69148047e+02 -2.36309207e+00]]
```

2 Q2

import package

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from numpy import array
from tkinter import _flatten
```

read data

[2]:
```
X = np.zeros((400, 2576))
for j in range(0, 40):
    for i in range(0, 10):
        image = Image.open(r"C:\Users\TerryYang\pythonwork\pythonwork\Data␣
↪Analytics Homework\ORL Faces\%s_%s.png" %(j+1, i+1))
        image_array = array(image)
        X[i+j*10] = image_array.flatten()
```

define Factor Analysis function

[3]:
```python
def FA(dataMat, factor_number):
    meanVals = np.mean(dataMat, axis=0)
    meanRemoved = dataMat - meanVals
    covMat = np.cov(meanRemoved, rowvar=0)
    eigVal, eigVect = np.linalg.eig(np.mat(covMat))
    selected_eigenvalue = eigVal[:factor_number]
    eigenvalues_diagonal = np.zeros((eigVal.shape[0], eigVal.shape[0]), float)
    np.fill_diagonal(eigenvalues_diagonal, eigVal)
    eigenvalues_diagonal_total_sqrt = np.sqrt(eigenvalues_diagonal)
    All_T =   eigenvalues_diagonal_total_sqrt @ eigVect
    All = All_T.T
    X_variance = np.diag(np.diag(All_T @ All))
    eigenvalues_diagonal = np.zeros((factor_number, factor_number), float)
    np.fill_diagonal(eigenvalues_diagonal, eigVal[:factor_number])
    eigenvalues_diagonal_sqrt = np.sqrt(eigenvalues_diagonal)
    A_T = eigVect[:, :factor_number] @ eigenvalues_diagonal_sqrt
    A = A_T.T
    Psi = X_variance - A_T @ A
    Psi_inverse = np.linalg.inv(Psi)
    inner = np.linalg.inv(A @ (Psi_inverse) @ (A_T))
    F = dataMat @ Psi_inverse @ (A_T) @ (inner)
    communality_vector = A_T @ A
    return F, A,communality_vector, Psi,eigVal, eigVect, selected_eigenvalue
```

define Factor Analysis function

[8]:
```python
def analyse_data(eigenvalues_selected, Total_eigenvalues):
    Printed = 0
    cumulated_values = 0
    for i in range(0, len(eigenvalues_selected)):
        cumulated_values += eigenvalues_selected[i]
        if  60 > (cumulated_values/Total_eigenvalues*100).real > 50 and Printed␣
↪== 0:
            print('Principal components %s, Variance percentage %s%%, Cumulated␣
↪percentage %s%%' % (format(i+1, '2.0f'), format(eigenvalues_selected[i]/
↪Total_eigenvalues*100, '4.2f'), format(cumulated_values/
↪Total_eigenvalues*100, '4.1f')))
            Printed = 1
        elif 70 > (cumulated_values/Total_eigenvalues*100).real > 60 and␣
↪Printed == 1:
            print('Principal components %s, Variance percentage %s%%, Cumulated␣
↪percentage %s%%' % (format(i+1, '2.0f'), format(eigenvalues_selected[i]/
↪Total_eigenvalues*100, '4.2f'), format(cumulated_values/
↪Total_eigenvalues*100, '4.1f')))
            Printed = 2
        elif 80 > (cumulated_values/Total_eigenvalues*100).real > 70 and␣
↪Printed == 2:
```

```
            print('Principal components %s, Variance percentage %s%%, Cumulated␣
↪percentage %s%%' % (format(i+1, '2.0f'), format(eigenvalues_selected[i]/
↪Total_eigenvalues*100, '4.2f'), format(cumulated_values/
↪Total_eigenvalues*100, '4.1f')))
            Printed = 3
        elif 90 > (cumulated_values/Total_eigenvalues*100).real > 80 and␣
↪Printed == 3:
            print('Principal components %s, Variance percentage %s%%, Cumulated␣
↪percentage %s%%' % (format(i+1, '2.0f'), format(eigenvalues_selected[i]/
↪Total_eigenvalues*100, '4.2f'), format(cumulated_values/
↪Total_eigenvalues*100, '4.1f')))
            Printed = 4
        elif (cumulated_values/Total_eigenvalues*100).real > 90 and Printed ==␣
↪4:
            print('Principal components %s, Variance percentage %s%%, Cumulated␣
↪percentage %s%%' % (format(i+1, '2.0f'), format(eigenvalues_selected[i]/
↪Total_eigenvalues*100, '4.2f'), format(cumulated_values/
↪Total_eigenvalues*100, '4.1f')))
            Printed = 5
```

call function

[9]:
```
F, A,communality_vector, Psi,eigVal, eigVect, selected_eigenvalue = FA(X.T, 100)
analyse_data(selected_eigenvalue.real,eigVal.real.sum())
```

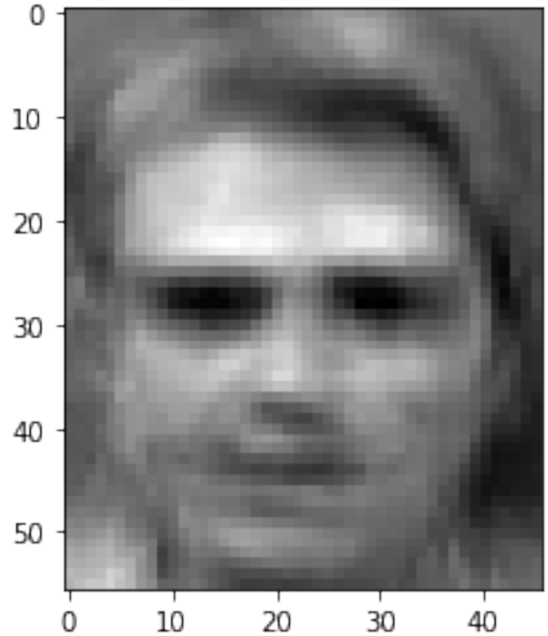
```
Principal components  2, Variance percentage 10.55%, Cumulated percentage 57.0%
Principal components  3, Variance percentage 4.57%, Cumulated percentage 61.5%
Principal components  6, Variance percentage 2.07%, Cumulated percentage 70.5%
Principal components 15, Variance percentage 0.66%, Cumulated percentage 80.2%
Principal components 47, Variance percentage 0.17%, Cumulated percentage 90.1%
```

plot image when selecting 15 factors

[12]:
```
F, A,communality_vector, Psi,eigVal, eigVect, eigenvalues_selected = FA(X.T, 15)

first_PC = F@A
first_PC=first_PC.T[0]
first_PC_array=first_PC.reshape(56,46).real
min_first_PC_array = np.min(first_PC_array)
range_first_PC_array = np.max(first_PC_array) - np.min(first_PC_array)
for i, j in enumerate(first_PC_array):
    first_PC_array[i] = 255 * ((j - min_first_PC_array) / range_first_PC_array)

imgplot = plt.imshow(first_PC_array, cmap='gray', vmin=0, vmax=255)
plt.show()
```



```
[ ]:
```