```
In [2]:
import numpy as np
import pandas as pd
```

```
In [3]:
df = pd.read_csv('Dataset Final Exam.csv', sep=',', header=None)
```

```
In [5]:
df.head()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	121	22	74	223	54	254	132	17	77	232	50	249
1	108	30	80	175	40	300	123	32	79	192	64	315
2	122	49	87	266	41	223	129	31	96	250	55	319
3	77	37	66	178	80	209	131	23	67	291	48	310
4	140	35	71	175	38	261	110	24	96	239	42	268

```
In [7]:
df_array = df.values
```

1. Compute the eigen values and corresponding eigenvectors of covariance matrix of the data

Calculate the covariance matrix

```
In [11]:
cov_matrix = np.cov(df_array)
```

Compute the eigen values and corresponding eigenvetcors of covariance matrix

```
In [12]:
w, v = np.linalg.eig(cov_matrix)
```

So, the eigenvalues are `w`:

```
In [13]:
w
```

```
Out[13]: array([ 1.61144702e+05+0.0000000e+00j,  3.02150989e+03+0.0000000e+00j,
         1.70301341e+03+0.0000000e+00j,  1.24683741e+03+0.0000000e+00j,
         6.05961223e+02+0.0000000e+00j,  5.13606549e+02+0.0000000e+00j,
         4.33538836e+02+0.0000000e+00j,  2.79820250e+02+0.0000000e+00j,
         9.99169851e+01+0.0000000e+00j,  6.46542242e+01+0.0000000e+00j,
         4.72197353e+01+0.0000000e+00j,  7.05342510e-12+0.0000000e+00j,
        -3.87211306e-12+0.0000000e+00j, -2.56017107e-12+4.4252166e-13j,
        -2.56017107e-12-4.4252166e-13j,  2.66140327e-12+0.0000000e+00j,
         1.75829854e-12+0.0000000e+00j,  8.66221612e-13+0.0000000e+00j,
        -3.97552641e-13+8.9093833e-14j, -3.97552641e-13-8.9093833e-14j])
```

the eigenvectors are `v`:

```
In [15]:
v
```

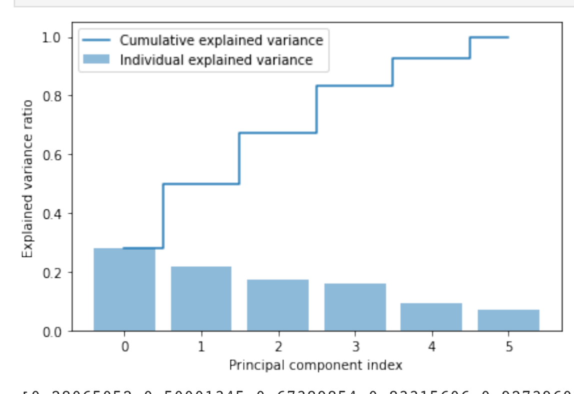
```
Out[15]: array([[-0.22339532+0.j        , -0.19672347+0.j        ,
          0.02897941+0.j        ,  0.01785256+0.j        ,
          0.14341725+0.j        ,  0.23928671+0.j        ,
         -0.29034263+0.j        , -0.32665894+0.j        ,
          0.11154024+0.j        , -0.03818529+0.j        ,
          0.07299793+0.j        ,  0.58479252+0.j        ,
          0.06914614+0.j        ,  0.06562965+0.1794376j ,
          0.06562965-0.1794376j ,  0.07294149+0.j        ,
          0.04674205+0.j        , -0.09394113+0.j        ,
         -0.20001793-0.06927581j, -0.20001793+0.06927581j],
        [-0.2375551 +0.j        ,  0.03676923+0.j        ,
         -0.50511715+0.j        , -0.10069305+0.j        ,
         -0.29072141+0.j        ,  0.14892796+0.j        ,
          0.36782412+0.j        , -0.273773  +0.j        ,
         -0.2399054 +0.j        ,  0.18522274+0.j        ,
         -0.01664072+0.j        , -0.21057783+0.j        ,
          0.13630588+0.j        ,  0.02988464+0.12937947j,
          0.02988464-0.12937947j,  0.04147694+0.j        ,
         -0.11957747+0.j        , -0.17608583+0.j        ,
         -0.23811405-0.09635452j, -0.23811405+0.09635452j],
        [-0.24074403+0.j        , -0.14505034+0.j        ,
          0.18805723+0.j        , -0.51311768+0.j        ,
          0.45997858+0.j        , -0.00309595+0.j        ,
          0.10443368+0.j        ,  0.13381827+0.j        ,
         -0.1922277 +0.j        ,  0.05934263+0.j        ,
         -0.07855447+0.j        , -0.22598327+0.j        ,
          0.14096269+0.j        ,  0.10555996+0.09177507j,
          0.10555996-0.09177507j,  0.23884943+0.j        ,
          0.19581873+0.j        ,  0.26810166+0.j        ,
          0.22285071+0.07913985j,  0.22285071-0.07913985j],
        [-0.23513568+0.j        ,  0.24870562+0.j        ,
          0.50028877+0.j        , -0.20216978+0.j        ,
         -0.23740992+0.j        ,  0.40409945+0.j        ,
          0.22308116+0.j        , -0.06828526+0.j        ,
          0.17644174+0.j        , -0.0208232 +0.j        ,
         -0.16933988+0.j        ,  0.0797985 +0.j        ,
         -0.18475975+0.j        , -0.09965848-0.07728204j,
         -0.09965848+0.07728204j,  0.11177164+0.j        ,
         -0.23022086+0.j        , -0.11373889+0.j        ,
         -0.0096744 +0.01658992j, -0.0096744 -0.01658992j],
        [-0.22525712+0.j        , -0.04807457+0.j        ,
         -0.0388007 +0.j        ,  0.21389391+0.j        ,
         -0.0518221 +0.j        , -0.31200064+0.j        ,
          0.14944391+0.j        , -0.07667185+0.j        ,
         -0.13611852+0.j        , -0.10834483+0.j        ,
          0.14264432+0.j        ,  0.25078092+0.j        ,
         -0.19644096+0.j        ,  0.5195753 +0.j        ,
          0.5195753 -0.j        ,  0.26267487+0.j        ,
         -0.01964951+0.j        ,  0.116842  +0.j        ,
         -0.01949124-0.05276126j, -0.01949124+0.05276126j],
        [-0.20869806+0.j        , -0.54645275+0.j        ,
          0.23956697+0.j        , -0.03871384+0.j        ,
         -0.29163671+0.j        , -0.04596679+0.j        ,
          0.14331209+0.j        ,  0.08377293+0.j        ,
          0.01917624+0.j        ,  0.08841875+0.j        ,
         -0.14569303+0.j        ,  0.1379338 +0.j        ,
          0.18588715+0.j        ,  0.21936388+0.06420816j,
          0.21936388-0.06420816j, -0.53377686+0.j        ,
         -0.29286599+0.j        , -0.31462166+0.j        ,
         -0.06593982-0.0391898j , -0.06593982+0.0391898j ],
        [-0.23768554+0.j        ,  0.09843809+0.j        ,
         -0.01813065+0.j        , -0.28775226+0.j        ,
         -0.26482874+0.j        , -0.1274894 +0.j        ,
         -0.08902571+0.j        ,  0.38056816+0.j        ,
          0.22303625+0.j        , -0.43011331+0.j        ,
          0.39601049+0.j        ,  0.00715969+0.j        ,
          0.20794094+0.j        , -0.08515635+0.05363623j,
         -0.08515635-0.05363623j, -0.03109924+0.j        ,
          0.01497451+0.j        ,  0.06546839+0.j        ,
         -0.14719211-0.14464789j, -0.14719211+0.14464789j],
        [-0.21821454+0.j        ,  0.02662209+0.j        ,
          0.04745986+0.j        , -0.18324273+0.j        ,
         -0.19891062+0.j        , -0.03408796+0.j        ,
         -0.17071882+0.j        , -0.13838422+0.j        ,
         -0.55322722+0.j        , -0.28022853+0.j        ,
          0.22681792+0.j        , -0.06391512+0.j        ,
         -0.23126718+0.j        , -0.07451254-0.12392564j,
         -0.07451254+0.12392564j, -0.00429446+0.j        ,
          0.18551371+0.j        , -0.00618431+0.j        ,
          0.05483229+0.16639751j,  0.05483229-0.16639751j],
        [-0.22493075+0.j        , -0.09712674+0.j        ,
         -0.01086398+0.j        ,  0.24960573+0.j        ,
          0.06507747+0.j        , -0.15928497+0.j        ,
          0.11977803+0.j        , -0.19779652+0.j        ,
         -0.11515304+0.j        , -0.3822026 +0.j        ,
         -0.33791127+0.j        ,  0.07554234+0.j        ,
          0.43728503+0.j        , -0.4532368 -0.12398867j,
         -0.4532368 +0.12398867j,  0.16990495+0.j        ,
          0.13704872+0.j        ,  0.14074627+0.j        ,
          0.35798605+0.08185784j,  0.35798605-0.08185784j],
        [-0.24240293+0.j        ,  0.24217039+0.j        ,
         -0.16647353+0.j        ,  0.04877833+0.j        ,
         -0.11109116+0.j        , -0.08799234+0.j        ,
          0.09028438+0.j        ,  0.26944387+0.j        ,
          0.2136273 +0.j        ,  0.3511583 +0.j        ,
          0.03819922+0.j        ,  0.27399684+0.j        ,
          0.39640653+0.j        , -0.09760668-0.07838063j,
         -0.09760668+0.07838063j,  0.07932584+0.j        ,
          0.20304901+0.j        ,  0.22375083+0.j        ,
          0.10657685+0.20750648j,  0.10657685-0.20750648j],
        [-0.2231001 +0.j        ,  0.25145408+0.j        ,
         -0.15190797+0.j        , -0.0987627 +0.j        ,
          0.05408229+0.j        , -0.25641513+0.j        ,
         -0.19938903+0.j        , -0.26858055+0.j        ,
          0.10890998+0.j        , -0.01094961+0.j        ,
         -0.06210712+0.j        ,  0.06477428+0.j        ,
         -0.11965591+0.j        , -0.1268194 +0.05205708j,
         -0.1268194 -0.05205708j, -0.56351886+0.j        ,
         -0.54587489+0.j        ,  0.06933604+0.j        ,
          0.03568305-0.01164776j,  0.03568305+0.01164776j],
        [-0.22238945+0.j        ,  0.04255912+0.j        ,
         -0.07624655+0.j        , -0.28418979+0.j        ,
          0.19259532+0.j        , -0.26947791+0.j        ,
         -0.26293062+0.j        ,  0.10578673+0.j        ,
         -0.15032239+0.j        ,  0.40712331+0.j        ,
         -0.15553634+0.j        ,  0.17824276+0.j        ,
         -0.08874527+0.j        , -0.10217923-0.1118537j ,
         -0.10217923+0.1118537j ,  0.00354229+0.j        ,
         -0.11145567+0.j        , -0.31261837+0.j        ,
         -0.14413041-0.12166388j, -0.14413041+0.12166388j],
        [-0.21246133+0.j        , -0.40204153+0.j        ,
         -0.32607725+0.j        , -0.07612021+0.j        ,
          0.06173788+0.j        ,  0.16406661+0.j        ,
          0.18151422+0.j        , -0.03905083+0.j        ,
          0.35486806+0.j        ,  0.08206663+0.j        ,
          0.25378588+0.j        , -0.10052913+0.j        ,
         -0.39912937+0.j        , -0.20694131-0.16710106j,
         -0.20694131+0.16710106j,  0.1394013 +0.j        ,
          0.20237432+0.j        ,  0.1886881 +0.j        ,
          0.35217406+0.0959033j ,  0.35217406-0.0959033j ],
        [-0.19855991+0.j        ,  0.08953525+0.j        ,
         -0.01679749+0.j        ,  0.29363384+0.j        ,
          0.2461571 +0.j        ,  0.07183505+0.j        ,
         -0.08637812+0.j        ,  0.18707022+0.j        ,
          0.01014215+0.j        , -0.05175944+0.j        ,
          0.33156786+0.j        , -0.11331321+0.j        ,
          0.04405722+0.j        ,  0.07668238+0.08350062j,
          0.07668238-0.08350062j, -0.01371466+0.j        ,
         -0.37377934+0.j        , -0.63307498+0.j        ,
          0.17672185+0.0799223j ,  0.17672185-0.0799223j ],
        [-0.21019657+0.j        ,  0.23640311+0.j        ,
          0.1857049 +0.j        ,  0.17391113+0.j        ,
         -0.25775689+0.j        ,  0.18190264+0.j        ,
         -0.32290426+0.j        , -0.18563975+0.j        ,
         -0.06406031+0.j        ,  0.33102973+0.j        ,
          0.11876119+0.j        , -0.21487283+0.j        ,
          0.10233005+0.j        ,  0.08806148+0.00027843j,
          0.08806148-0.00027843j,  0.13059066+0.j        ,
          0.35499276+0.j        ,  0.24316595+0.j        ,
          0.36935314+0.j        ,  0.36935314-0.j        ],
        [-0.23572075+0.j        ,  0.09678348+0.j        ,
          0.20759731+0.j        ,  0.10061708+0.j        ,
          0.36742988+0.j        , -0.10424664+0.j        ,
          0.15090711+0.j        , -0.39150611+0.j        ,
          0.29127743+0.j        , -0.03537984+0.j        ,
          0.18566305+0.j        , -0.3027663 +0.j        ,
          0.10135356+0.j        ,  0.08194975-0.05878449j,
          0.08194975+0.05878449j, -0.12998844+0.j        ,
          0.11259598+0.j        , -0.00507008+0.j        ,
         -0.33774521-0.0568601j , -0.33774521+0.0568601j ],
        [-0.23720374+0.j        ,  0.25374981+0.j        ,
          0.01444574+0.j        ,  0.12297294+0.j        ,
         -0.03825101+0.j        , -0.27906567+0.j        ,
          0.21357162+0.j        ,  0.16756334+0.j        ,
          0.15171473+0.j        , -0.10264319+0.j        ,
         -0.38525035+0.j        , -0.07269736+0.j        ,
         -0.41846505+0.j        ,  0.16359153+0.1150344j ,
          0.16359153-0.1150344j ,  0.09505162+0.j        ,
          0.26301805+0.j        , -0.13334922+0.j        ,
          0.01746735-0.01125317j,  0.01746735+0.01125317j],
        [-0.21319548+0.j        , -0.36000346+0.j        ,
          0.15744169+0.j        ,  0.29289024+0.j        ,
         -0.18683328+0.j        , -0.18688669+0.j        ,
         -0.33563653+0.j        ,  0.13291381+0.j        ,
          0.04995255+0.j        ,  0.15116531+0.j        ,
         -0.0961445 +0.j        , -0.38468796+0.j        ,
         -0.12894916+0.j        , -0.18506151+0.00625298j,
         -0.18506151-0.00625298j,  0.23232297+0.j        ,
          0.0075483 +0.j        ,  0.17934068+0.j        ,
         -0.18482894+0.01152633j, -0.18482894-0.01152633j],
        [-0.20815042+0.j        ,  0.05636968+0.j        ,
          0.08790932+0.j        ,  0.36358754+0.j        ,
          0.24441433+0.j        ,  0.27374537+0.j        ,
          0.25382303+0.j        ,  0.34302462+0.j        ,
         -0.38475879+0.j        ,  0.09598762+0.j        ,
          0.12537585+0.j        ,  0.15043851+0.j        ,
         -0.09051312+0.j        , -0.17865328-0.00371411j,
         -0.17865328+0.00371411j, -0.30835922+0.j        ,
         -0.11073022+0.j        ,  0.1992339 +0.j        ,
         -0.19355774-0.12139977j, -0.19355774+0.12139977j],
        [-0.20992135+0.j        ,  0.02315776+0.j        ,
         -0.35106461+0.j        ,  0.03261913+0.j        ,
          0.10479045+0.j        ,  0.45290972+0.j        ,
         -0.34389905+0.j        ,  0.19019633+0.j        ,
          0.0730208 +0.j        , -0.28123025+0.j        ,
         -0.41195968+0.j        , -0.12943298+0.j        ,
         -0.01713095+0.j        ,  0.26350384-0.04204039j,
          0.26350384+0.04204039j, -0.0642151 +0.j        ,
          0.00999399+0.j        ,  0.0282345 +0.j        ,
         -0.11244688-0.01367986j, -0.11244688+0.01367986j]])
```

2. Find the proportion of the total variance explained by the componets

```
In [34]:
def show_proportion_of_total_variance(data):
    from sklearn.decomposition import PCA
    from sklearn.preprocessing import StandardScaler
    import matplotlib.pyplot as plt
    sc = StandardScaler()
    sc.fit(data)
    X_train_std = sc.transform(data)
    #
    # Instantiate PCA
    #
    pca = PCA()
    #
    # Determine transformed features
    #
    X_train_pca = pca.fit_transform(X_train_std)
    #
    # Determine explained variance using explained_variance_ration_ attribute
    #
    exp_var_pca = pca.explained_variance_ratio_
    #
    # Cumulative sum of eigenvalues; This will be used to create step plot
    # for visualizing the variance explained by each principal component.
    #
    cum_sum_eigenvalues = np.cumsum(exp_var_pca)
    #
    # Create the visualization plot
    #
    plt.bar(range(0,len(exp_var_pca)), exp_var_pca, alpha=0.5, align='center', label='Individual explained variance')
    plt.step(range(0,len(cum_sum_eigenvalues)), cum_sum_eigenvalues, where='mid',label='Cumulative explained variance')
    plt.ylabel('Explained variance ratio')
    plt.xlabel('Principal component index')
    plt.legend(loc='best')
    plt.tight_layout()
    plt.show()
    print(cum_sum_eigenvalues)
```

Class One

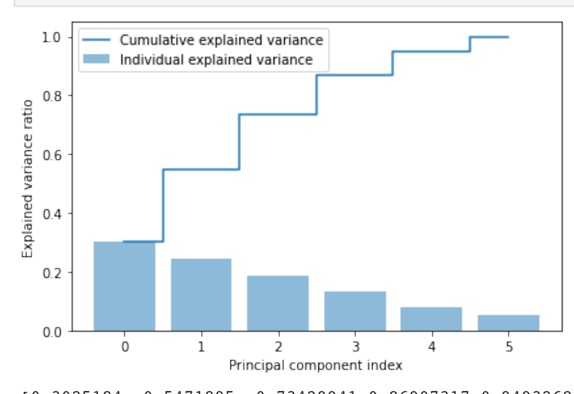
```
In [35]:
show_proportion_of_total_variance(df_array[:,:6])
```




```
[0.28065052 0.50001345 0.67389854 0.83315606 0.92739607 1.        ]
```

Class Two

```
In [36]:
show_proportion_of_total_variance(df_array[:,6:12])
```




```
[0.3025184  0.5471895  0.73429941 0.86907317 0.94932685 1.        ]
```

3. How much of the total variance account of first two principal components

From the graph above:

For group one the total cumulative variance of the first two principal components is 0.50001345, and for group two, it is 0.5471895

4. Plot eigenvalues to visualize the proportion of variance explained by each subsequential eigenvalue

See the plot above...

5. Use PCA to transforms the data into a new set of variables (PCs).

```
In [42]:
from sklearn.decomposition import PCA
```

```
In [45]:
pca = PCA()
PCs1 = pca.fit_transform(df_array[:,:6])
```

```
In [51]:
PCs1.shape
```

```
Out[51]: (20, 6)
```

```
In [62]:
# Show the first ten transformed dataset
PCs1[:10]
```

```
Out[62]: array([[ 2.95855406e+01,  1.04753398e+00, -4.43359660e+00,
         -8.45079931e+00,  4.75320892e-02,  1.54174710e+01],
        [-1.81949155e+01, -3.93707582e+01, -3.00355646e+01,
         -1.91765084e+01, -1.11702615e+01,  6.92929231e+00],
        [ 7.33590225e+01,  1.80561182e+01,  2.26865371e+00,
          2.55280398e+01, -1.18482170e+01, -2.79241192e+00],
        [-1.73078743e+01,  6.36302576e+01, -2.36981746e+01,
         -5.71272977e+00, -1.19577840e+00, -5.91896560e+00],
        [-1.63538209e+01, -2.15793445e+01,  3.80172637e+00,
          7.73820269e+00,  1.03899916e+01,  5.99788322e+00],
        [ 4.79477477e+01,  1.61066481e+01, -1.65991050e+01,
         -8.16642089e+00,  1.11677116e+01, -1.23526371e+00],
        [ 1.45888192e+00,  1.65109788e+01,  1.52143518e+00,
         -1.28063812e+01,  2.24243943e+01, -7.06842434e+00],
        [ 5.57689169e+00,  1.80258311e+01,  2.15045621e+01,
         -1.89776666e+01, -1.15200007e+01, -1.95812627e+00],
        [ 6.33592677e+00, -3.47624299e+01,  1.68524134e+01,
         -2.07153944e+00, -1.02093509e+01, -1.47505438e+01],
        [-3.00515242e+01, -1.95890840e+01, -3.79466049e+00,
         -3.06080860e+00,  5.50127482e+00, -3.41556633e-01]])
```

```
In [63]:
pca = PCA()
PCs2 = pca.fit_transform(df_array[:,6:12])
```

```
In [64]:
PCs2.shape
```

```
Out[64]: (20, 6)
```

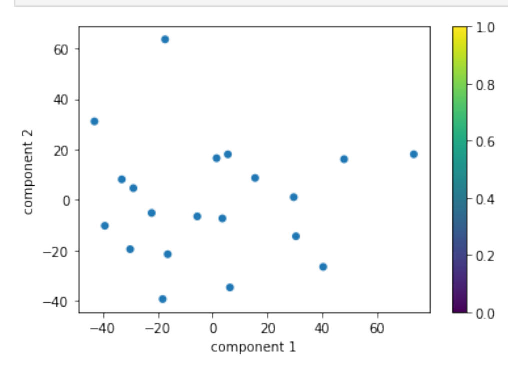
```
In [65]:
# Show the first ten transformed dataset
PCs2[:10]
```

```
Out[65]: array([[ 2.57987568e+01, -4.44996417e-01, -1.36587030e+01,
          1.46937098e+01,  1.46387053e+01,  8.75623533e+00],
        [-2.05691979e+01,  4.97833007e+01,  2.50045904e+01,
          1.30672508e+01,  9.72814533e-01, -1.48479781e+00],
        [-3.57185310e+01, -7.59731771e+00,  1.99682674e+01,
          3.89541223e+00, -1.12608034e+01,  9.26376967e-01],
        [-3.80795840e+01, -4.68440883e+01,  1.44431891e+01,
          7.47341690e-01,  2.24085124e+01,  6.88124030e-01],
        [ 2.01706192e+01, -7.22063755e+00,  1.12826372e+01,
         -1.62821365e+00, -5.32677627e+00,  6.93451440e+00],
        [ 9.42216922e+01, -1.15579932e+01,  4.24446586e+01,
         -5.40734636e+00,  8.31219725e+00, -8.33845688e-02],
        [-3.27328458e+01,  1.59619735e+01,  2.06471902e+01,
         -2.06716231e+01, -2.60369028e+00,  6.76587287e+00],
        [-1.60668186e+01,  4.25459049e+00,  1.19255363e+01,
          1.03632796e+01, -1.64871753e+01,  3.27343404e+00],
        [ 5.31503747e+00, -2.24090026e+01, -2.13695812e+00,
          1.54438195e+01, -1.40463416e+01,  2.68544536e+00],
        [-4.47924520e+01,  6.28367625e+00, -5.64217643e-01,
         -2.07886207e+01,  8.90666627e-02, -7.74991201e+00]])
```

6. Find and plot top first 2 PCs

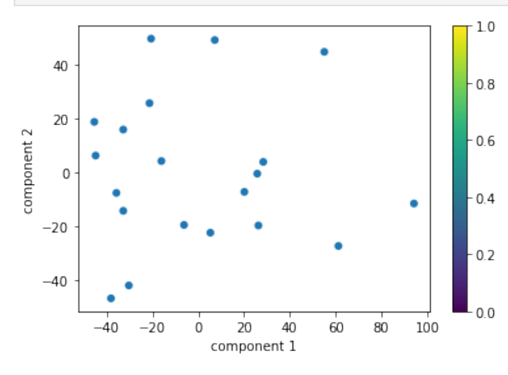
Group One

```
In [69]:
import matplotlib.pyplot as plt

plt.scatter(PCs1[:, 0], PCs1[:, 1], edgecolor='none')
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```



Group Two

```
In [68]:
import matplotlib.pyplot as plt

plt.scatter(PCs2[:, 0], PCs2[:, 1],edgecolor='none')
plt.xlabel('component 1')
plt.ylabel('component 2')
plt.colorbar();
```



7. Calculate the correlation between the original data and the component. Interpreted based which variable they are most correlated in either a positive or negative direction

```
In [77]:
from scipy.stats.stats import pearsonr
```

Group One, the first column is the correlation, while the second column is the p-value of it

```
In [78]:
for i in range(0,6):
    print(pearsonr(df_array[:,i],PCs1[:,i]))
```

```
(0.06443976579649642, 0.7872312944499033)
(-0.23561348797700016, 0.3173008531423038)
(0.48627097053316576, 0.029705481882213156)
(-0.0199265233478159, 0.9335462703764398)
(-0.029205019669966647, 0.9027212706909494)
(-0.023247303429626366, 0.9225012869007734)
```

Group Two, the first column is the correlation, while the second column is the p-value of it

```
In [80]:
for i in range(0,6):
    print(pearsonr(df_array[:,6+i],PCs2[:,i]))
```

```
(-0.6967817622764083, 0.0006410022205784092)
(0.3171408130009202, 0.1730547124932433)
(-0.04261720113097531, 0.8584101937933257)
(-0.008883966504801565, 0.9703473814209748)
(-0.05857956480843332, 0.8062106147959324)
(0.009756211602223001, 0.9674374413498783)
```