CSCI E-82

HW 1 Dimensionality Reduction

Due: Sept 17, 2018 11:59pm EST

Note that this is an individual homework to be completed without collaborations except through Piazza.

We encourage you to make progress this weekend since the second homework will likely come out in a week before this one is due.

Your name:

Sharjil Khan

Problem 1 (5 points)

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

Compute XY^T. The answer can be computed by hand and written in Markdown like the above matrices, or computed in python. Either way is acceptable.

In [3]:
```
import numpy as np

X = np.array([[1,2,3],[4,5,6],[7,8,9]])
Y = np.array([[1,2,1],[2,1,2]])

print("Answer = \n", np.dot(X,Y.T))
```

```
Answer =
 [[ 8 10]
 [20 25]
 [32 40]]
```

Problem 2

This problem goes through a combination of python data manipulations as well as the full math projection using PCA. We have divided the problem into multiple parts.

Problem 2a (5 points)

Download and load in the data set from the UCI archive https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/ (https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/). Print the dimensions and the first few rows to demonstrate a successful load.

In [4]:
```
import pandas as pd
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/ecoli.data'
ecoli = pd.read_csv(url, delim_whitespace=True, header= None)
rows, columns = ecoli.shape
print("Rows: %d, Columns: %d" %(rows, columns))
ecoli.head()
```

```
Rows: 336, Columns: 9
```

Out[4]:

	0	1	2	3	4	5	6	7	8
0	AAT_ECOLI	0.49	0.29	0.48	0.5	0.56	0.24	0.35	cp
1	ACEA_ECOLI	0.07	0.40	0.48	0.5	0.54	0.35	0.44	cp
2	ACEK_ECOLI	0.56	0.40	0.48	0.5	0.49	0.37	0.46	cp
3	ACKA_ECOLI	0.59	0.49	0.48	0.5	0.52	0.45	0.36	cp
4	ADI_ECOLI	0.23	0.32	0.48	0.5	0.55	0.25	0.35	cp

Problem 2b (10 points)

Compute and print the covariance matrix for all columns excluding the first and last. Rather than use the built-in function, compute this using python code for practice. The following equation will suffice for this.

Cov(X, Y) = Σ (Xi - X) (Yi - Y) / N

In [5]:
```
from sklearn import decomposition, preprocessing

#DROP FIRST AND LAST COLUMN
cols = [0,8]
ecoli_numeric = ecoli.drop(ecoli.columns[cols],axis=1)

#SUBTRACT MEAN TO CENTER THE DATA AROUND 0 ON ALL AXIS
mu = ecoli_numeric.mean(axis=0)
ecoli_0 = ecoli_numeric - mu

#CALCULATE MANUALLY
means = ecoli_0.mean(axis=0)
ecoli0 = ecoli_0 - means
N, column_num = ecoli0.shape

# FUNCTION TO COMPUTE EACH ROW OF THE COVARIANCE MATRIX
def covariance_row (matrix , column_values ):
    N, column_num = matrix.shape
    row = []
    for i in range(1, column_num+1):
        dot_prod = np.dot(matrix.iloc[:, (i-1): i].T, column_values)
        row.append(dot_prod[0,0]/N)
    return row
# FUNCTION TO COMPUTE THE COVARIANCE MATRIX
def covariance_matrix(dataframe):
    N, column_num = dataframe.shape
    cov = []
    for i in range(0,column_num):
        cov.append(covariance_row(dataframe, dataframe.iloc[:,i:i+1]))
    return cov

#GET THE COVARIANCE MATRIX AND PRINT IT
C = np.array(covariance_matrix(ecoli0))
print(C)
```

```
[[ 3.77696393e-02  1.30758929e-02  2.52169785e-03  3.71935232e-04
   5.24106966e-03  1.66205251e-02  6.78989690e-03]
 [ 1.30758929e-02  2.18851190e-02  5.72619048e-04  7.44047619e-05
   1.26220238e-03  5.52916667e-03 -3.71815476e-03]
 [ 2.52169785e-03  5.72619048e-04  7.80810658e-03  7.50779478e-04
   7.57872732e-04  1.82342687e-03 -1.06371173e-03]
 [ 3.71935232e-04  7.44047619e-05  7.50779478e-04  7.41833192e-04
  -1.48853812e-04 -4.49085884e-05 -2.97220451e-04]
 [ 5.24106966e-03  1.26220238e-03  7.57872732e-04 -1.48853812e-04
   1.49312491e-02  7.35713754e-03  6.45596035e-03]
 [ 1.66205251e-02  5.52916667e-03  1.82342687e-03 -4.49085884e-05
   7.35713754e-03  4.64100872e-02  3.64568931e-02]
 [ 6.78989690e-03 -3.71815476e-03 -1.06371173e-03 -2.97220451e-04
   6.45596035e-03  3.64568931e-02  4.37222497e-02]]
```

Problem 2c (10 points)

Compute the decomposition of the covariance matrix using singular value decomposition. Using a python function is definitely the way to go here.

In [8]:
```
# FIND THE EIGENVALUES AND EIGENVECTORS FROM THE COVARIANCE MATRIX
eigenvalues, eigenvectors = np.linalg.eigh(C)


#SORT THE EIGENVALUES IN DESCENDING ORDER
idx = np.argsort(-eigenvalues)
eigenvalues = eigenvalues[idx]
print('Eigenvalues Sorted:')
print(eigenvalues)
eigenvectors = eigenvectors[:,idx]
print('Eigenvectors Sorted:')
print(eigenvectors)

# COMPUTE U, S, V
U = eigenvectors
S = eigenvalues * np.eye(column_num)
V = U.T

print("\nU:")
print(U)
print("S:")
print(S)
print("V:")
print(V)

# CHECK IF ORIGINAL COVARIANCE MATRIX IS RETURNED
D = np.dot(U, np.dot(S,  V))
np.allclose(C,D)
```

```
Eigenvalues Sorted:
[0.08943556 0.0423127  0.01458897 0.01284528 0.00850822 0.00492059
 0.00065696]
Eigenvectors Sorted:
[[-3.41720629e-01 -7.29824958e-01  4.56914809e-01  3.52555665e-01
  -1.29441525e-01  2.62199804e-02 -8.56193962e-03]
 [-9.17492644e-02 -5.28794379e-01 -7.04480258e-01 -3.42514767e-01
  -1.39018617e-01  2.81111220e-01  2.89703065e-04]
 [-1.99698823e-02 -7.25012267e-02  9.24097367e-02 -1.46705018e-02
   8.54762221e-01  4.93627297e-01 -1.06318528e-01]
 [ 9.74209562e-04 -1.16809378e-02  8.81896510e-03  1.64849996e-02
   8.61437517e-02  6.21081678e-02  9.94100049e-01]
 [-1.47115119e-01 -4.80496808e-02  4.70463079e-01 -8.66343630e-01
  -4.86492931e-02 -3.90505852e-02  1.64278490e-02]
 [-6.89914858e-01  7.21275410e-02 -2.54072739e-01 -1.83906883e-02
   3.50569568e-01 -5.75266798e-01  9.64476070e-03]
 [-6.13827312e-01  4.18124394e-01  2.07514962e-02  8.37356323e-02
  -3.17196689e-01  5.83358230e-01 -5.01770938e-03]]

U:
[[-3.41720629e-01 -7.29824958e-01  4.56914809e-01  3.52555665e-01
  -1.29441525e-01  2.62199804e-02 -8.56193962e-03]
 [-9.17492644e-02 -5.28794379e-01 -7.04480258e-01 -3.42514767e-01
  -1.39018617e-01  2.81111220e-01  2.89703065e-04]
 [-1.99698823e-02 -7.25012267e-02  9.24097367e-02 -1.46705018e-02
   8.54762221e-01  4.93627297e-01 -1.06318528e-01]
 [ 9.74209562e-04 -1.16809378e-02  8.81896510e-03  1.64849996e-02
   8.61437517e-02  6.21081678e-02  9.94100049e-01]
 [-1.47115119e-01 -4.80496808e-02  4.70463079e-01 -8.66343630e-01
  -4.86492931e-02 -3.90505852e-02  1.64278490e-02]
 [-6.89914858e-01  7.21275410e-02 -2.54072739e-01 -1.83906883e-02
   3.50569568e-01 -5.75266798e-01  9.64476070e-03]
 [-6.13827312e-01  4.18124394e-01  2.07514962e-02  8.37356323e-02
  -3.17196689e-01  5.83358230e-01 -5.01770938e-03]]
S:
[[0.08943556 0.         0.         0.         0.         0.
  0.        ]
 [0.         0.0423127  0.         0.         0.         0.
  0.        ]
 [0.         0.         0.01458897 0.         0.         0.
  0.        ]
 [0.         0.         0.         0.01284528 0.         0.
  0.        ]
 [0.         0.         0.         0.         0.00850822 0.
  0.        ]
 [0.         0.         0.         0.         0.         0.00492059
  0.        ]
 [0.         0.         0.         0.         0.         0.
  0.00065696]]
V:
[[-3.41720629e-01 -9.17492644e-02 -1.99698823e-02  9.74209562e-04
  -1.47115119e-01 -6.89914858e-01 -6.13827312e-01]
 [-7.29824958e-01 -5.28794379e-01 -7.25012267e-02 -1.16809378e-02
  -4.80496808e-02  7.21275410e-02  4.18124394e-01]
 [ 4.56914809e-01 -7.04480258e-01  9.24097367e-02  8.81896510e-03
   4.70463079e-01 -2.54072739e-01  2.07514962e-02]
 [ 3.52555665e-01 -3.42514767e-01 -1.46705018e-02  1.64849996e-02
```

```
 -8.66343630e-01 -1.83906883e-02  8.37356323e-02]
[-1.29441525e-01 -1.39018617e-01  8.54762221e-01  8.61437517e-02
 -4.86492931e-02  3.50569568e-01 -3.17196689e-01]
[ 2.62199804e-02  2.81111220e-01  4.93627297e-01  6.21081678e-02
 -3.90505852e-02 -5.75266798e-01  5.83358230e-01]
[-8.56193962e-03  2.89703065e-04 -1.06318528e-01  9.94100049e-01
  1.64278490e-02  9.64476070e-03 -5.01770938e-03]]
```

Out[8]: True

Problem 2d (10 points)

Compute the projection of the raw data onto the appropriate two eigenvectors. Consider which columns should be projected and the normalizations.

In [10]:
```
# The data was centered so there is not need to normalize the eigenvectors
# The U matrix is sorted from highest eigenvalue to lowest in previous section
 so no need to re-arrange here.

Comp_num = 2

# PROJECT DATA ON TOP TWO EIGENVECTORS
ecoli_PCA =np.dot(ecoli0,U[:,0:Comp_num])
print(ecoli_PCA)
```

```
[[ 0.28560071  0.03527368]
 [ 0.29083817  0.330159  ]
 [ 0.10467597 -0.0152477 ]
 [ 0.08794301 -0.12221767]
 [ 0.3662676   0.21036611]
 [ 0.08023001 -0.08327277]
 [ 0.38555357  0.18739006]
 [ 0.33190109  0.23519751]
 [ 0.0643298   0.28387054]
 [ 0.37151499  0.00296018]
 [ 0.27966348  0.12284697]
 [ 0.4529588   0.08559031]
 [ 0.33173181  0.09719732]
 [ 0.21901728 -0.07979446]
 [ 0.4749507   0.12710027]
 [ 0.12989914  0.24257591]
 [ 0.34865965  0.06878927]
 [-0.02836356  0.19918564]
 [ 0.34335381  0.19045807]
 [ 0.3513704  -0.06094186]
 [ 0.55679426 -0.12278632]
 [ 0.58759337  0.08869104]
 [ 0.15979858 -0.06365125]
 [ 0.52580325  0.18313219]
 [ 0.42007625  0.02949799]
 [ 0.11604049  0.22828575]
 [ 0.26570136  0.19234895]
 [ 0.27790173  0.15703056]
 [ 0.17204237  0.08376635]
 [ 0.22504571  0.06064461]
 [ 0.2613917  -0.03012105]
 [ 0.22853447  0.17780982]
 [ 0.11665082 -0.06620763]
 [ 0.20581659  0.15325011]
 [ 0.46135353  0.17768333]
 [ 0.16504526  0.06125132]
 [ 0.29438979  0.08667428]
 [ 0.13791083 -0.15288892]
 [ 0.31664858  0.11611992]
 [ 0.36998975  0.00297334]
 [ 0.23945791  0.10021823]
 [ 0.13555289  0.20099448]
 [ 0.13612438  0.05619063]
 [ 0.25287127  0.1795886 ]
 [ 0.12896329 -0.10004694]
 [ 0.18661777  0.30221326]
 [-0.07481577  0.17158965]
 [ 0.26887141  0.04867432]
 [ 0.41227242  0.01373134]
 [ 0.27870455  0.25006429]
 [ 0.19577803  0.02512113]
 [ 0.62458105  0.09082702]
 [ 0.2251905   0.26639531]
 [ 0.0283988   0.20207408]
 [ 0.43637704 -0.07237731]
 [ 0.1881318  -0.04718646]
 [ 0.15814802  0.03893181]
```

```
[ 0.22197303  0.11027592]
[ 0.22515069  0.09775956]
[ 0.18610835  0.08039364]
[ 0.01833289  0.02718512]
[ 0.2449825  -0.00100039]
[ 0.05547609  0.16657255]
[ 0.40532813  0.26103644]
[ 0.18859236  0.0922141 ]
[ 0.02063356  0.26520527]
[ 0.27728074  0.21109576]
[-0.00505219 -0.03192357]
[ 0.35946212  0.17999838]
[ 0.12582531  0.10287482]
[ 0.39241511  0.23028946]
[ 0.2758057  -0.03462157]
[ 0.18376345  0.06900056]
[ 0.3965304  -0.05224877]
[ 0.34012643  0.16825698]
[ 0.3004363   0.09054126]
[ 0.28637403 -0.0201646 ]
[ 0.06666642  0.07691664]
[ 0.28975521  0.09382565]
[ 0.19059505  0.17771886]
[ 0.22870848  0.20280805]
[ 0.23115245  0.05505032]
[ 0.37253466  0.08836062]
[ 0.22493802  0.03643629]
[ 0.06900602  0.08403208]
[ 0.00526035  0.11882288]
[ 0.01186153  0.01671049]
[ 0.16378088 -0.01888677]
[ 0.31769947 -0.07932744]
[ 0.2598017  -0.02311136]
[ 0.19601094  0.10233959]
[ 0.36078878  0.01005797]
[ 0.3703031  -0.01930099]
[ 0.40438304  0.0842357 ]
[ 0.31816695  0.10404268]
[ 0.11309767 -0.29697019]
[ 0.24625392  0.14830125]
[ 0.24795757 -0.15165885]
[ 0.19271491 -0.22830605]
[ 0.17510294 -0.04004246]
[ 0.29590036  0.05474484]
[ 0.21694535  0.10696672]
[ 0.42704586  0.09734034]
[ 0.40639913  0.02888906]
[ 0.32083787  0.22517302]
[ 0.2705601   0.12871023]
[ 0.18206702  0.00517858]
[ 0.08795519  0.16047871]
[ 0.21096762 -0.01016018]
[ 0.38233194  0.01046059]
[ 0.55039742  0.06056855]
[ 0.03732716  0.22126026]
[ 0.35127694  0.08669213]
[ 0.28234862  0.1368443 ]
```

```
[ 0.38070224  0.02287165]
[ 0.34196091 -0.04324615]
[ 0.228901    0.12387432]
[ 0.38906584 -0.12848415]
[ 0.15912193  0.17892797]
[ 0.14743047  0.26511122]
[ 0.29721461  0.26950286]
[ 0.38335644  0.15273556]
[ 0.32424186  0.23843235]
[ 0.40055844  0.38065783]
[ 0.11733475 -0.04039222]
[ 0.17711785  0.08954889]
[ 0.39594958  0.16286558]
[ 0.19021922  0.06335507]
[ 0.36291074  0.21463995]
[ 0.15629441  0.11125454]
[ 0.17703748  0.45295886]
[ 0.22632084  0.14699194]
[ 0.37313814  0.08817971]
[ 0.2856448   0.20294986]
[ 0.22693711  0.2059287 ]
[ 0.23048762  0.09370682]
[ 0.43439891  0.11264946]
[ 0.35745222  0.16438102]
[ 0.33289165  0.02817806]
[ 0.3348251   0.23294501]
[-0.02565911  0.1663488 ]
[ 0.23110945  0.17976915]
[ 0.35549193  0.21357201]
[-0.06793977  0.24065691]
[ 0.02540177  0.05242165]
[-0.47232529  0.08703261]
[-0.42041     0.39506455]
[-0.3815037   0.27018134]
[-0.32273309  0.23651334]
[-0.46867564  0.15394197]
[-0.27310884  0.38871233]
[-0.30458056  0.14896224]
[-0.17620367  0.25058282]
[-0.40166486  0.12292555]
[-0.0557508   0.23731631]
[-0.41707814  0.23230157]
[-0.61054576  0.11119793]
[-0.39964732 -0.09867178]
[-0.38032288  0.18314769]
[-0.48627159  0.16775779]
[-0.49530852  0.2507292 ]
[-0.53400861  0.13171157]
[-0.30584045 -0.04344151]
[-0.30415289  0.11257235]
[-0.29943612  0.24742747]
[-0.38630386 -0.02587031]
[-0.44506369  0.09253242]
[-0.35804479  0.11463491]
[-0.51982097  0.07785112]
[-0.38372697  0.19997927]
[-0.39449064  0.30745358]
```

```
[-0.58513294  0.04592758]
[-0.43123346  0.34467284]
[-0.32094354  0.21557283]
[-0.378491    0.00558454]
[-0.22453518  0.15188023]
[-0.27897578 -0.05723657]
[-0.29604724 -0.03172065]
[-0.40893263 -0.06060438]
[-0.49245752  0.30069979]
[-0.46023159  0.02858249]
[-0.51863736  0.06926185]
[-0.4205751   0.03359005]
[ 0.22873538  0.23600789]
[-0.34372453 -0.19705863]
[-0.28728342  0.38783347]
[-0.45041492 -0.18568813]
[-0.4075664  -0.01004846]
[-0.1678591   0.42416141]
[-0.17457982 -0.07087104]
[-0.14391879  0.28628168]
[-0.32762264 -0.08053313]
[-0.31863534  0.19534494]
[-0.32096105  0.16720861]
[-0.46674853  0.18092729]
[-0.44672366  0.05690709]
[-0.19138966  0.12323758]
[-0.21112852  0.02901131]
[-0.43929185  0.11219577]
[-0.29373785  0.25731919]
[-0.32723359  0.02786958]
[-0.21671251  0.22453384]
[-0.26204259  0.24859382]
[-0.25775163  0.21568039]
[-0.32577707  0.32639461]
[-0.53461631  0.02389202]
[-0.31008004  0.12748742]
[-0.14047724  0.37320162]
[-0.50735284  0.42584825]
[-0.43738063 -0.01035674]
[-0.42222923  0.05742981]
[-0.38525487  0.09977841]
[ 0.01265413 -0.10452855]
[ 0.07286098  0.01860053]
[ 0.11184531  0.35528962]
[ 0.21186161  0.19383142]
[ 0.10180869  0.08683856]
[-0.05747366 -0.09273279]
[-0.34455879 -0.14903014]
[-0.73963414  0.07166205]
[-0.04820124 -0.33273675]
[-0.40648747  0.04978136]
[ 0.05855067 -0.33213739]
[-0.49929013  0.04054778]
[-0.45390169  0.01654853]
[-0.53052548 -0.01012169]
[-0.49009677 -0.09618115]
[-0.40276666  0.025512  ]
```

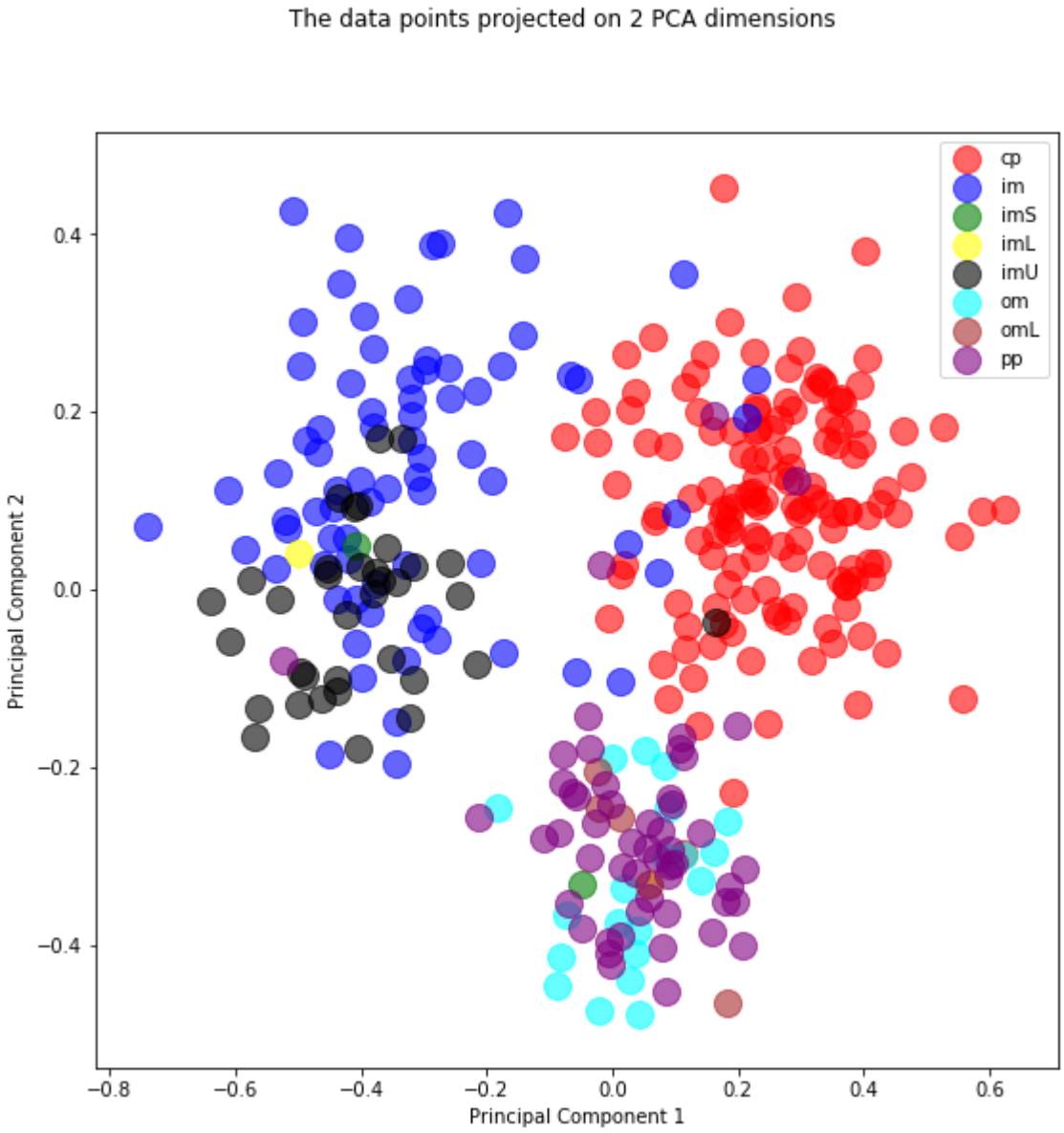
```
[-0.42361659 -0.02770693]
[-0.45437828  0.0241945 ]
[-0.37910929 -0.00438115]
[-0.37000461  0.1708445 ]
[-0.2588356   0.0293947 ]
[-0.34373534  0.00794068]
[-0.35921303  0.04652237]
[-0.40616255 -0.17984001]
[-0.46282688 -0.12342605]
[-0.43852679 -0.10214397]
[-0.31614256 -0.10186661]
[-0.36773735  0.01099519]
[-0.40427554  0.09458247]
[-0.43550641  0.10269954]
[-0.64047398 -0.01223116]
[-0.6072157  -0.05861517]
[-0.49879848 -0.12874011]
[-0.33475406  0.16876817]
[-0.21605044 -0.08421317]
[-0.57571397  0.0105166 ]
[-0.24207489 -0.00778687]
[-0.49509968 -0.09326866]
[-0.41093323  0.09207756]
[-0.31779603  0.0253273 ]
[-0.43730234 -0.11358708]
[-0.56887236 -0.16610282]
[-0.37702255  0.01897167]
[-0.56287737 -0.13379916]
[-0.32335314 -0.1447148 ]
[ 0.16336102 -0.03700957]
[-0.35169929 -0.0770857 ]
[ 0.0374417  -0.40912583]
[ 0.11350214 -0.29975106]
[ 0.02695423 -0.43931122]
[ 0.15992954 -0.29501925]
[-0.07210282 -0.36557991]
[-0.02051332 -0.4743981 ]
[-0.08875028 -0.44570022]
[-0.08307043 -0.41309171]
[ 0.00915109 -0.3743173 ]
[ 0.04305255 -0.47740786]
[ 0.04058347 -0.3836835 ]
[ 0.05083137 -0.18166656]
[ 0.00133115 -0.18936227]
[ 0.08909816 -0.2466405 ]
[ 0.18120883 -0.26012916]
[-0.18221007 -0.2450057 ]
[ 0.14128623 -0.32713834]
[ 0.08365929 -0.19783404]
[ 0.01701157 -0.33526193]
[ 0.08477446 -0.30071741]
[ 0.1835235  -0.46623496]
[-0.02519471 -0.20403302]
[-0.02045377 -0.24662308]
[ 0.05719648 -0.33100722]
[ 0.012207   -0.25664472]
[-0.01069591 -0.22039497]
```

```
[-0.05780273 -0.2335581 ]
[ 0.08415265 -0.3631955 ]
[-0.0060356  -0.39720509]
[-0.03841681 -0.14174703]
[ 0.084145   -0.45143735]
[ 0.06857974 -0.30131124]
[ 0.08050425 -0.40224993]
[ 0.09398324 -0.24098621]
[-0.06870081 -0.35424354]
[ 0.10934398 -0.16535533]
[-0.03610165 -0.30133026]
[ 0.07730555 -0.2711938 ]
[ 0.15918513 -0.38500197]
[ 0.17921619 -0.35213511]
[-0.21324142 -0.25762672]
[ 0.29292033  0.12333481]
[-0.06496879 -0.22909376]
[-0.00291865 -0.24114075]
[-0.04746685 -0.38096075]
[ 0.05793017 -0.34773807]
[ 0.20647878 -0.4008436 ]
[ 0.05840106 -0.26265205]
[-0.00689613 -0.40991812]
[ 0.01389074 -0.31256163]
[ 0.03625891 -0.31985453]
[-0.07815722 -0.18641177]
[-0.00371419 -0.42339406]
[ 0.09175669 -0.23467383]
[ 0.18467557 -0.33431481]
[-0.52496176 -0.08101752]
[ 0.01274853 -0.38940058]
[ 0.21104953 -0.31454606]
[ 0.19421481 -0.35182051]
[ 0.03166292 -0.28492959]
[-0.0378903  -0.179018  ]
[ 0.09185903 -0.29361228]
[ 0.09700778 -0.30922729]
[-0.07964565 -0.21691282]
[ 0.04257996 -0.36113851]
[ 0.05585319 -0.29159337]
[ 0.16047059  0.1961778 ]
[ 0.09286279 -0.31078603]
[ 0.19887035 -0.15329584]
[ 0.08738149 -0.32219706]
[-0.02848041 -0.26202269]
[-0.01691867  0.02818486]
[-0.08423256 -0.27480267]
[ 0.13902577 -0.27411596]
[ 0.11190373 -0.18710274]
[ 0.10620378 -0.17885061]
[-0.10876379 -0.28112947]]
```

Problem 2e (10 points)

Plot the projected points such that the 8 different classes can be visually identified. Be sure to label the classes and axes. Commont on the quality of the separation of the different classes using PCA.

In [13]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
import matplotlib.cm as cm
import matplotlib as mpl
import matplotlib.style

from matplotlib.colors import ListedColormap
# Remember to use inline to get your plots in the notebook
%matplotlib inline

# FUNCTION TO DRAW THE PLOT
def draw_plot(df_PCA, df_orig, title):
    cdict = {'cp':'red', 'im':'blue', 'imS':'green', 'imL':'yellow', 'imU':'bl
ack', 'om':'cyan', 'omL':'brown', 'pp':'purple'}
    plt.figure(figsize=(9,9))
    plt.suptitle(title)
    for c in df_orig[8].unique():
        idx = np.where(df_orig[8]== c)
        plt.scatter(df_PCA.T[0][idx], df_PCA.T[1][idx], s=200, color = cdict[c
], alpha = 0.6, label= c)
    plt.legend()
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.show()

draw_plot(ecoli_PCA, ecoli, 'The data points projected on 2 PCA dimensions')
```



The two PCA components are able to clearly seperate between 'cp', 'im' and 'pp'. But these two PCA components are not able to capture the difference between 'im' and 'imU'. It is also not able to capture the difference between 'om' and 'pp'.

So we can conclude these 2 PCA dimensions are able to seperate between some classes very well. But we might need to look at some of the other dimensions to see if we can find the difference between the classes that are not very well seperated on these 2 dimensions.

Problem 2f (10 points)

The PCA that you have just completed takes each data point and projects it using a weighted sum of features. One could also do the opposite to map the features as a weighted sum of the data entries. How could this be done? What is a potential issue? Describe these in a few sentences (do not code it).

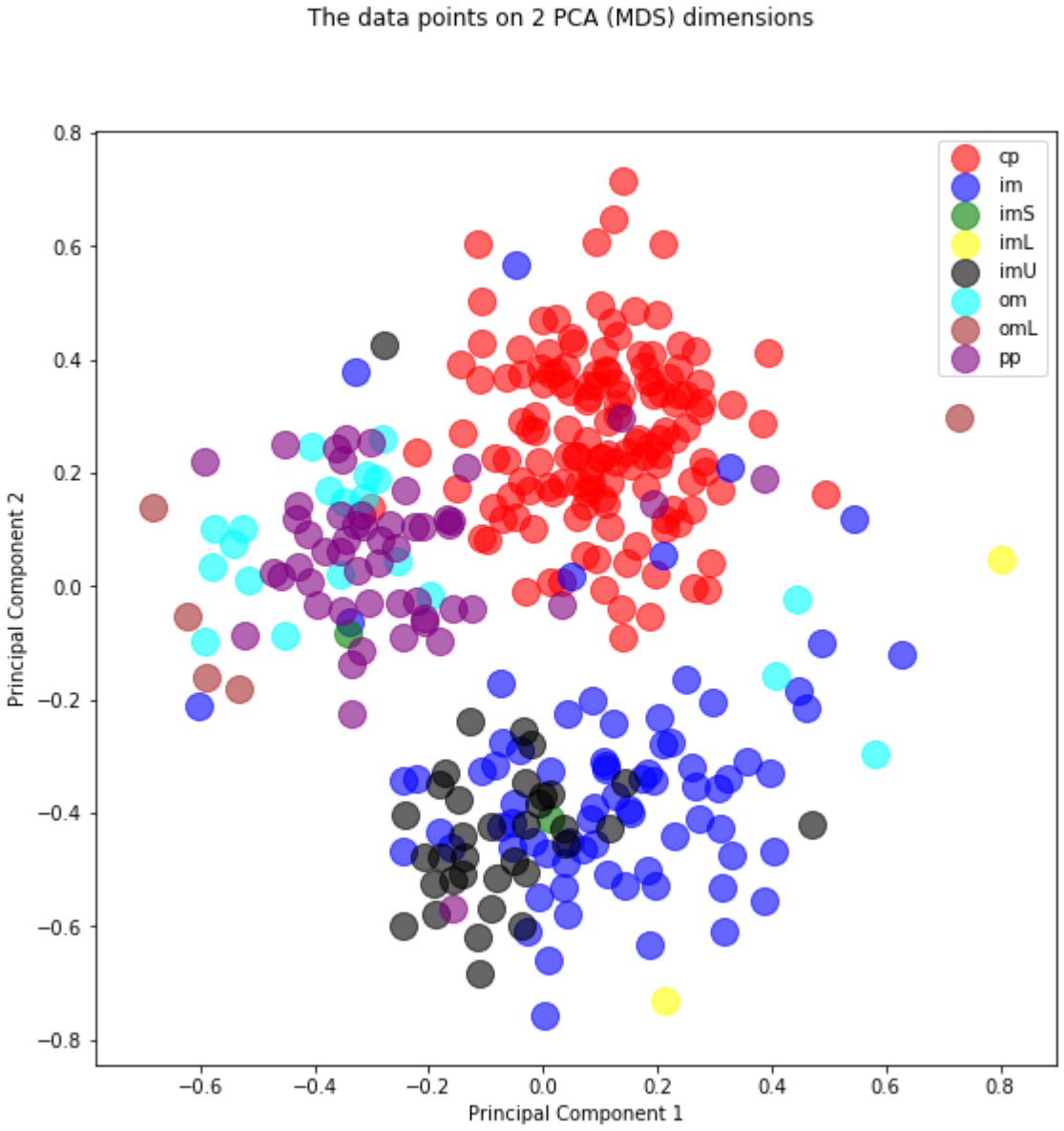
We could take the transpose of the original data matrix such that each data entry would become seperate columns(features), and the features will become rows.Then we can apply the same PCA dimensionality reduction techniques to map the features into a weighted sum of the data entries.

Since, we have a lot of data entries, the covariance matrix will be very large. If the covariance matrix is very large, each eigenvector will likely capture a small percentage of the variance. So, we will have to include many components to capture enough variance of the data to be able to do meaningful analysis.

Problem 3 MDS (10 points)

For the same data set, repeat 2e using sklearn's Multidimensional scaling algorithm.

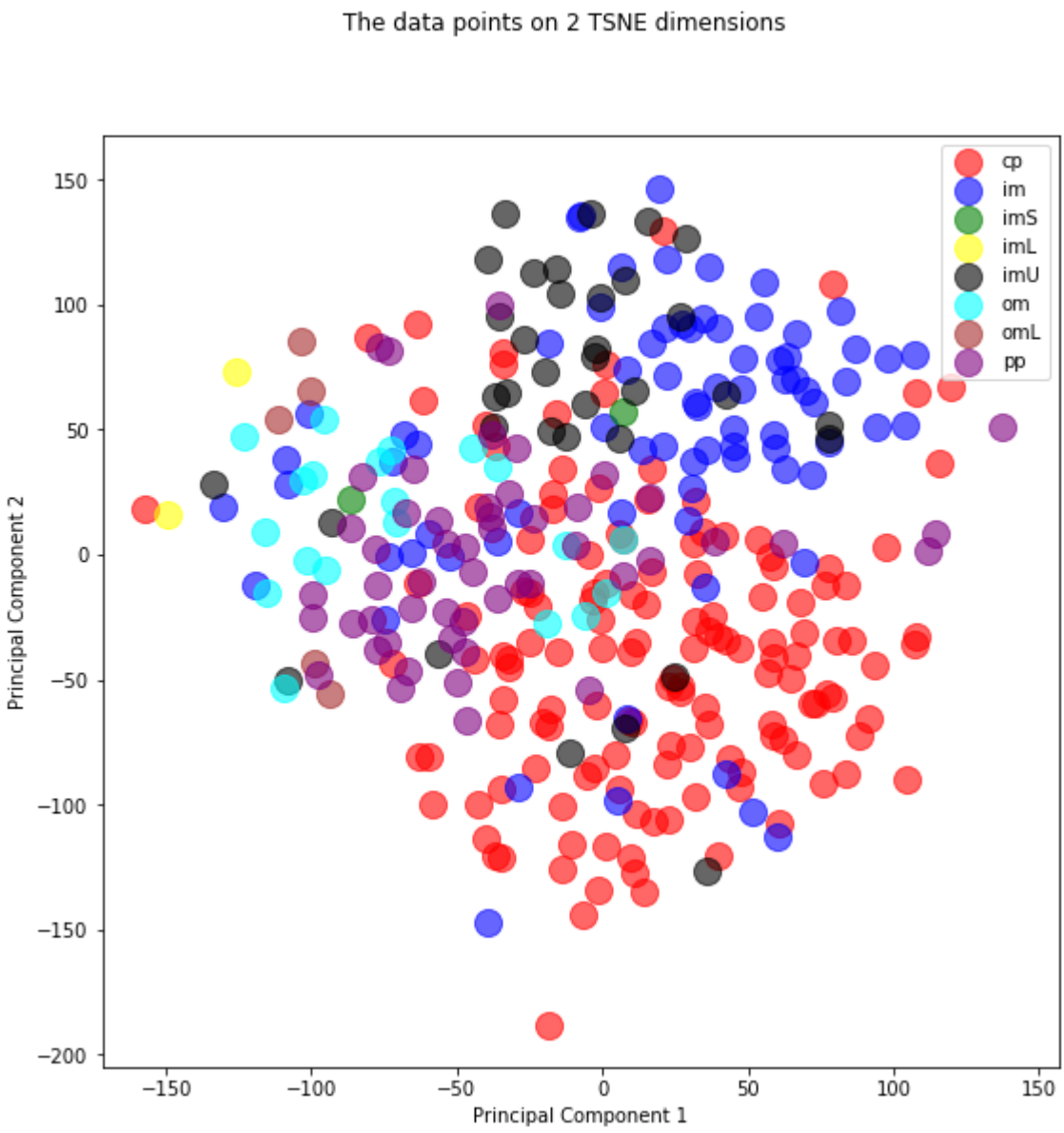
In [14]:
```
from sklearn.manifold import MDS
ecoli_PCA_MDS = MDS(n_components = 2).fit_transform(ecoli0)

#print(ecoli_PCA_MDS)
draw_plot(ecoli_PCA_MDS, ecoli, 'The data points on 2 PCA (MDS) dimensions')
```



Problem 4a t-SNE (5 points)

Repeat 2e using a t-SNE plot with the default settings.

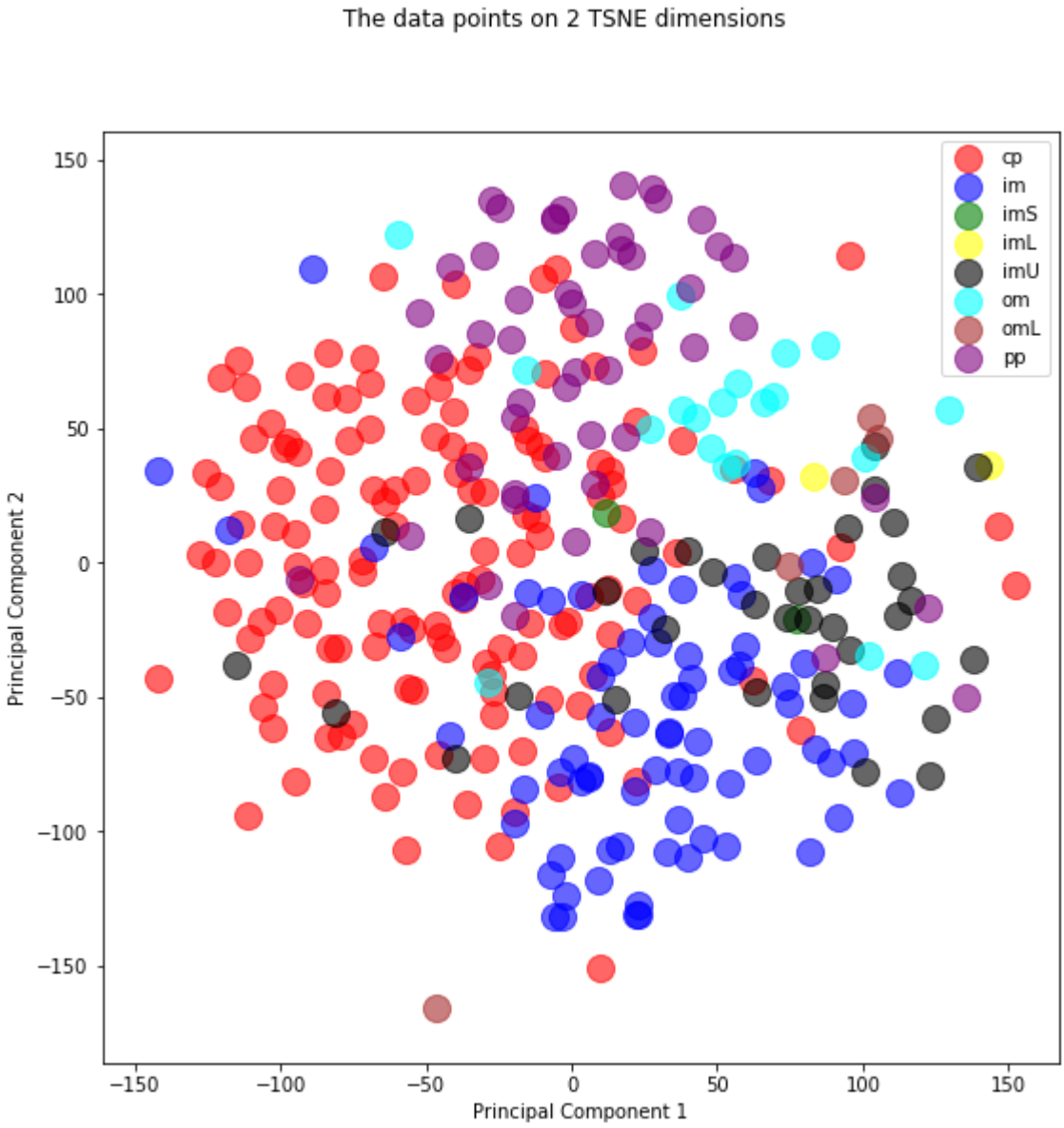
In [15]:
```
from sklearn.manifold import TSNE

ecoli_TSNE = TSNE(n_components=3, perplexity=35, verbose=0).fit_transform(ecol
i0)
draw_plot(ecoli_TSNE, ecoli, 'The data points on 2 TSNE dimensions')
```



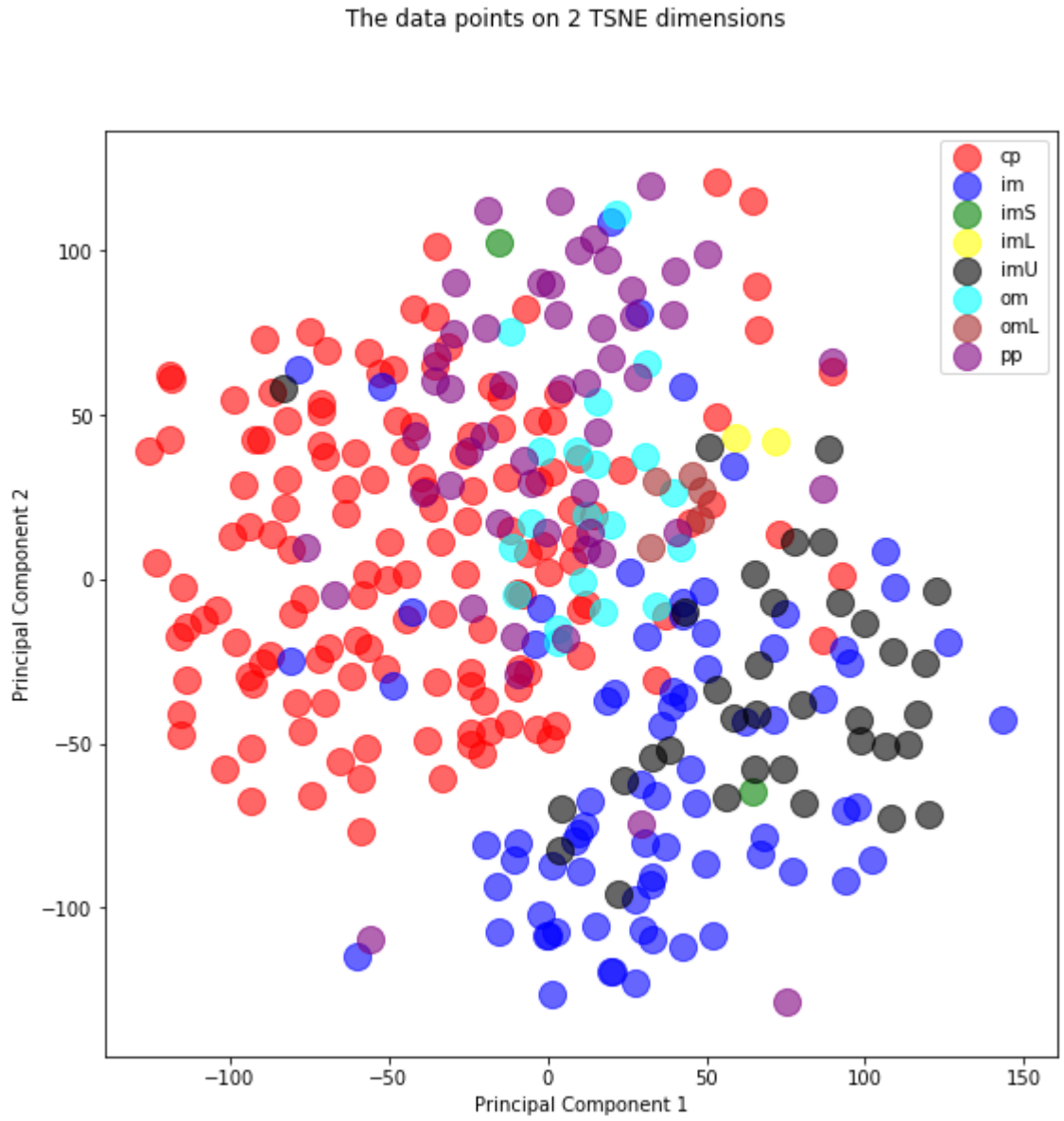
Problem 4b t-SNE perplexity (5 points)

Try out a few t-SNE plots by varying the perplexity. State the best perplexity for separating the 8 different classes and describe your rationale in a sentence or two. Report the average calculation time for the t-SNE projection over a number of iterations.

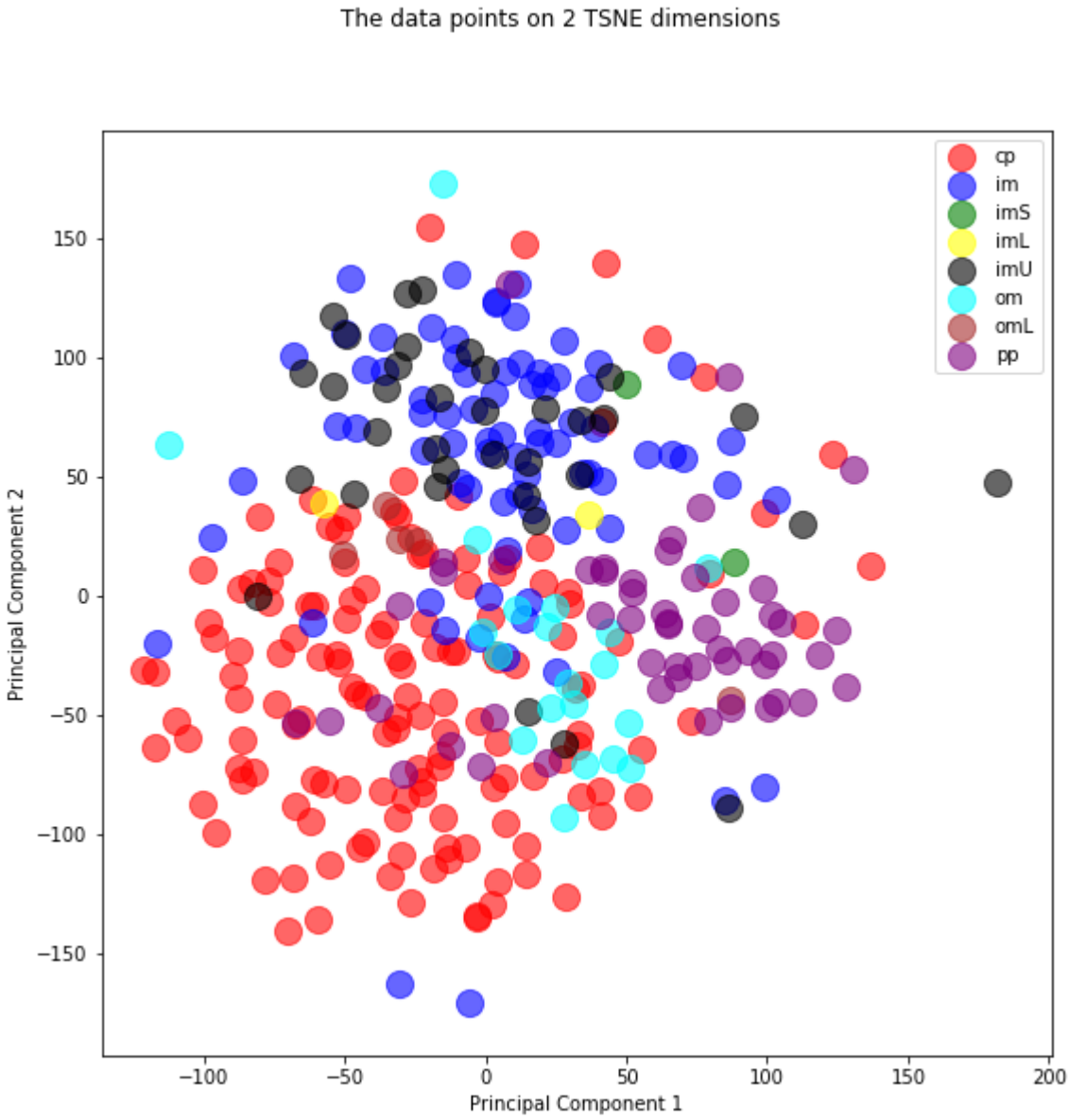
In [16]:
```
import time

starttime = time.time()
ecoli_TSNE = TSNE(n_components=3, perplexity=35, verbose=0).fit_transform(ecol
i0)
draw_plot(ecoli_TSNE, ecoli, 'The data points on 2 TSNE dimensions')
endtime = time.time()
print("{:03.9f} seconds".format(endtime-starttime))

# AVERAGE TIME TO COMPUTE TSNE PROJECTION
times = []
n = 0
for i in range(5):
    perplexity = (25+(n*5))
    starttime = time.time()
    print('Perplexity = %d'%(perplexity))
    ecoli_TSNE = TSNE(n_components=3, perplexity= perplexity, verbose=0).fit_t
ransform(ecoli0)
    endtime = time.time()
    draw_plot(ecoli_TSNE, ecoli, 'The data points on 2 TSNE dimensions')
    times.append(endtime-starttime)
    n = n + 1

print('Average Time for TSNE projection: {:03.9f} seconds'.format(np.mean(time
s)))
```



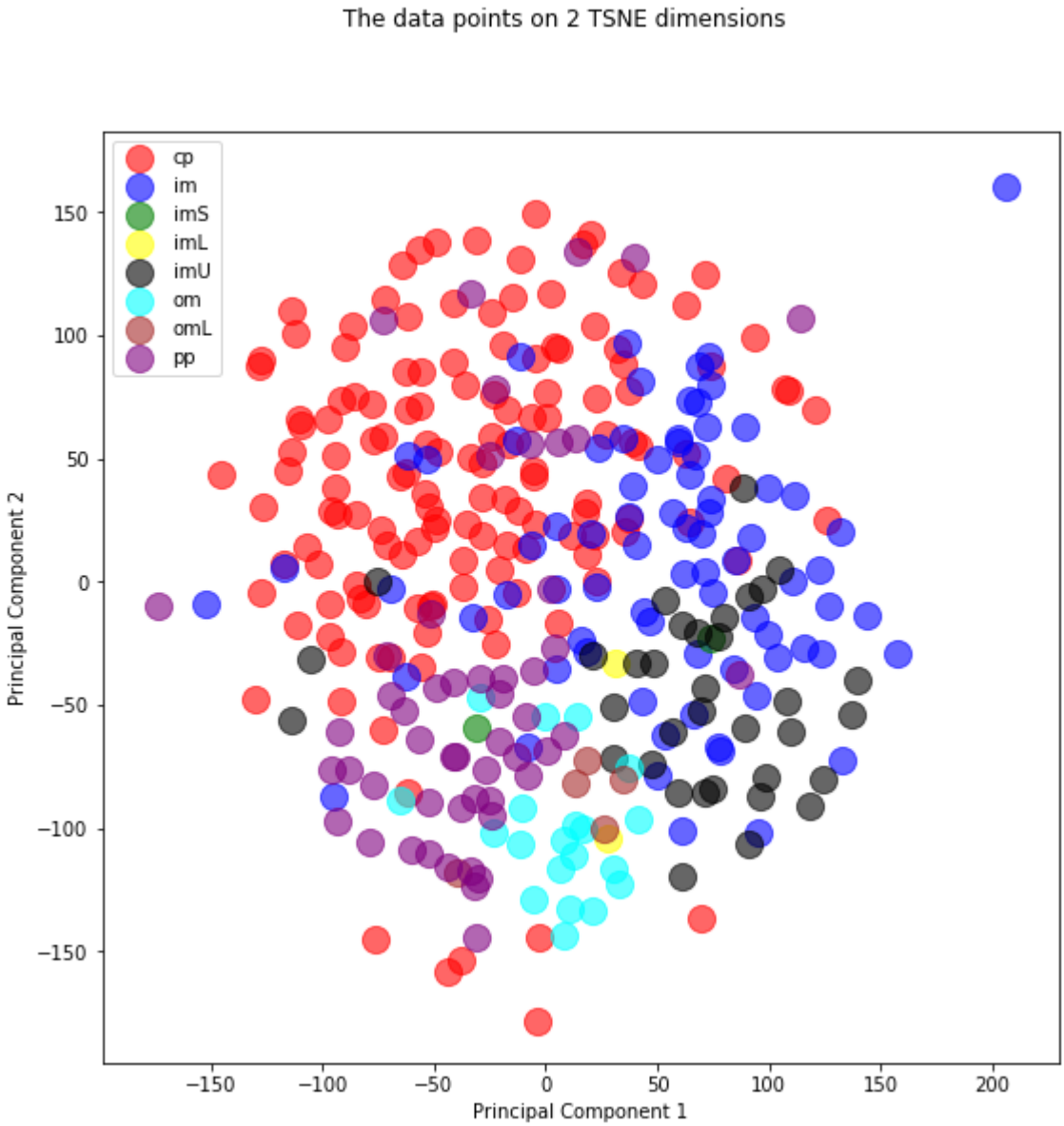

```
12.971605778 seconds
Perplexity = 25
```



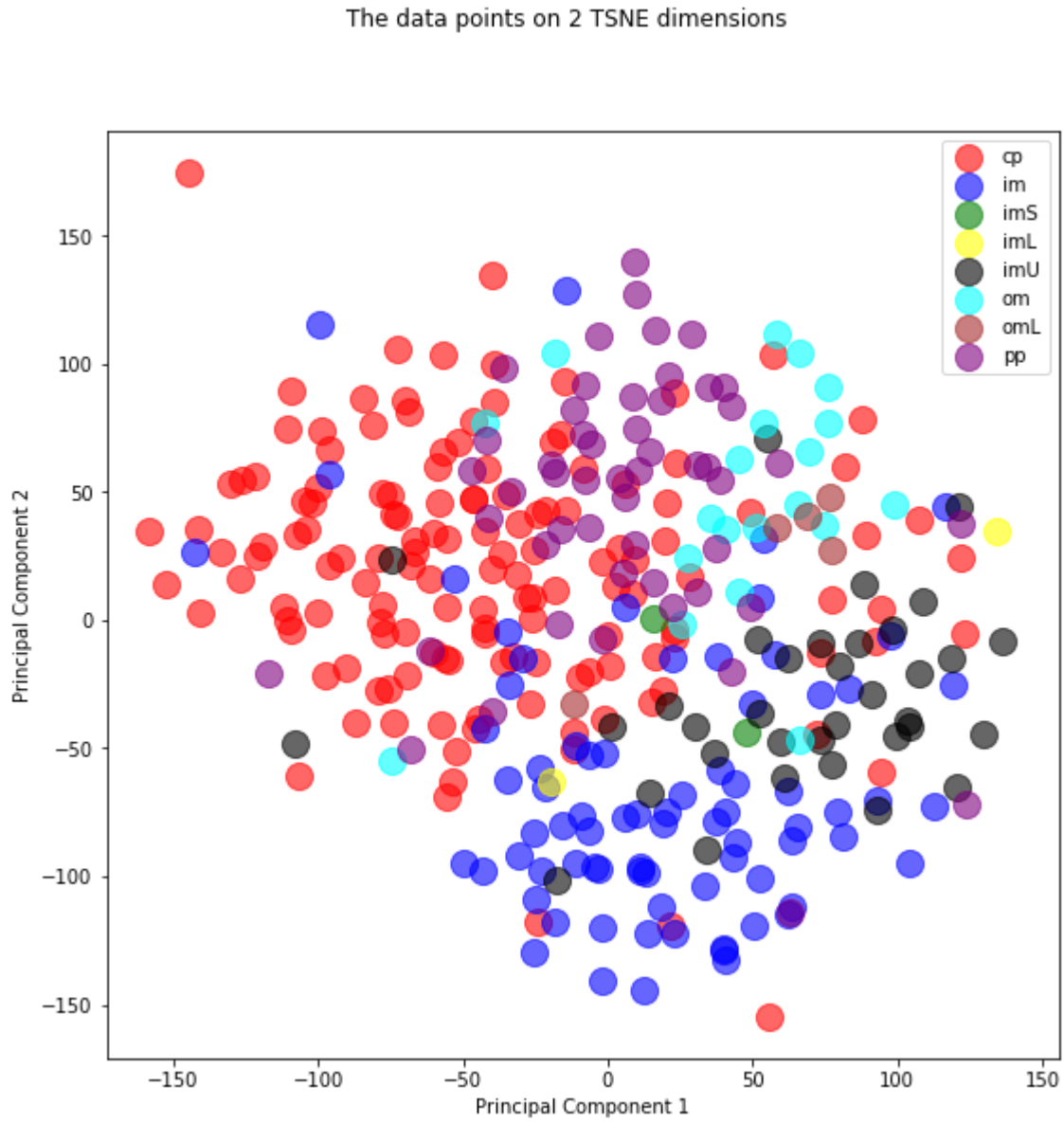
Perplexity = 30



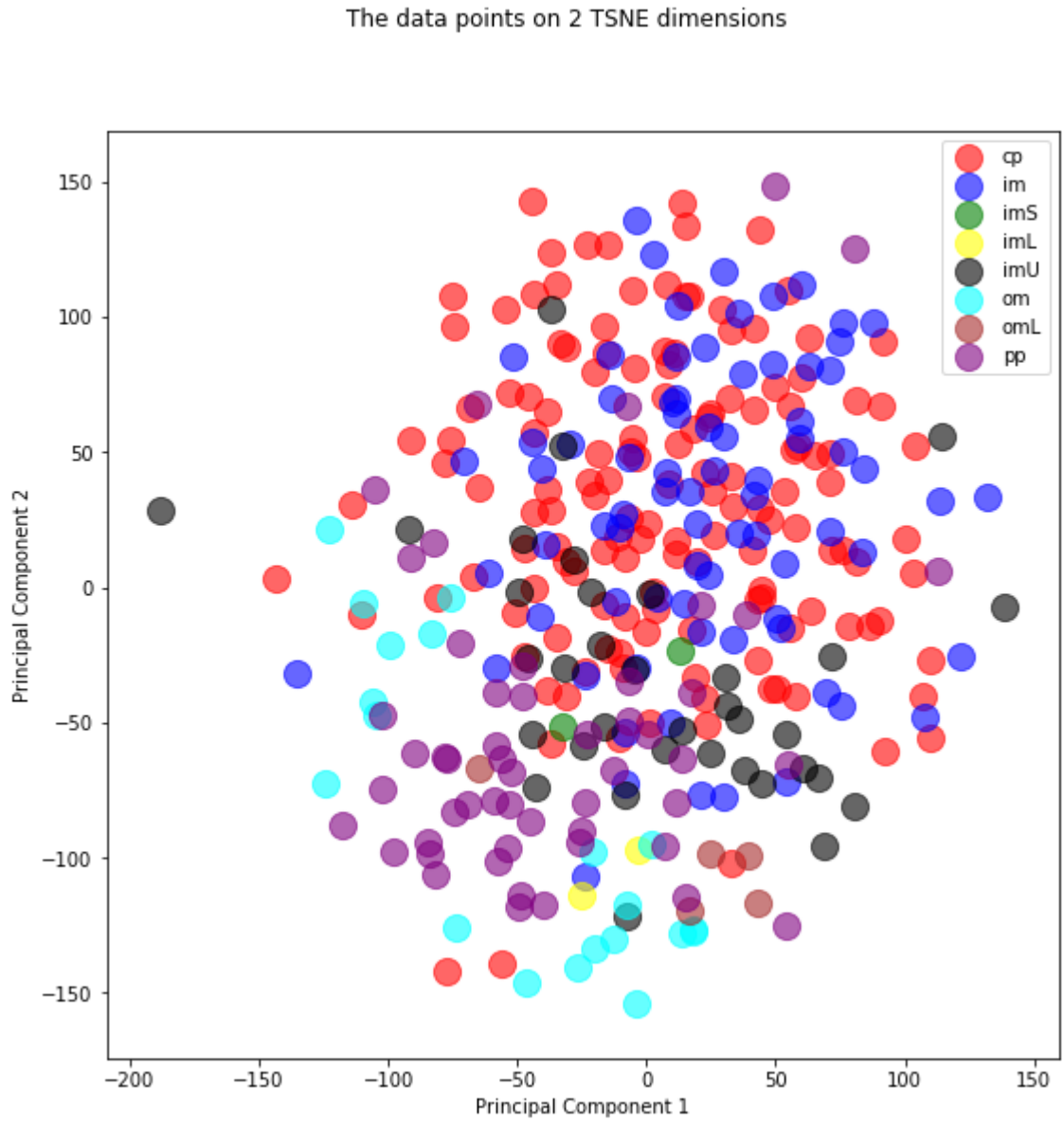
Perplexity = 35



Perplexity = 40



Perplexity = 45




```
Average Time for TSNE projection: 13.229911995 seconds
```

Looking at the distribution of the different Perplexities, it looks like 40 is the best value for perplexity because it groups the different classes with the least amount of overlap for this value.

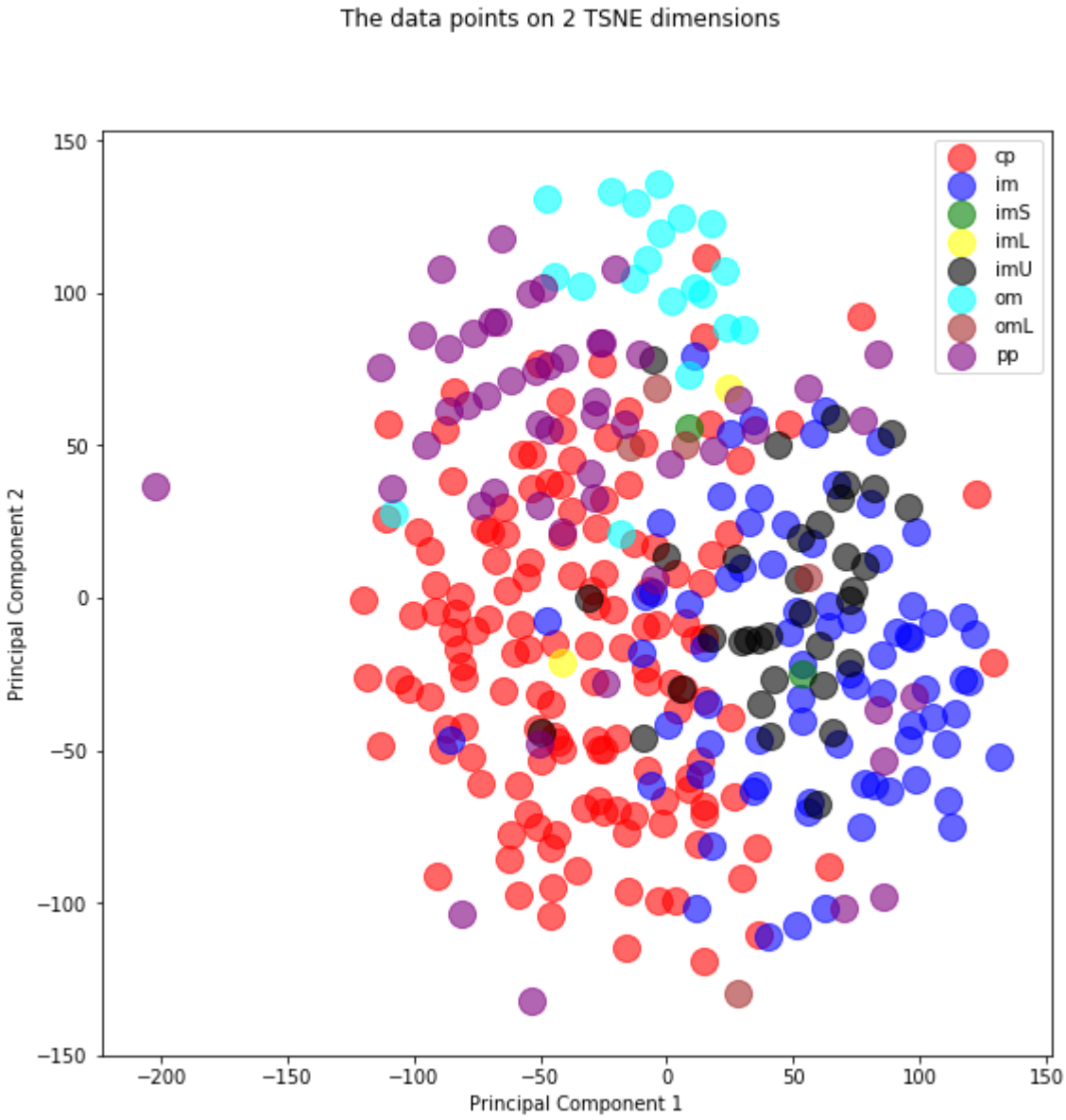
The average Time for TSNE projections is about 13.22 seconds.

Problem 4c t-SNE randomization (10 points)

The S of t-SNE means stochastic or random, usually as a function of time. Explore whether you can reproduce the result in 4b through a second projection and plot.

In [17]:
```
ecoli_TSNE_second = TSNE(n_components=3, perplexity=35, verbose=0).fit_transfo
rm(ecoli0)
print(np.allclose(ecoli_TSNE,ecoli_TSNE_second))
draw_plot(ecoli_TSNE_second, ecoli, 'The data points on 2 TSNE dimensions')
```

```
False
```

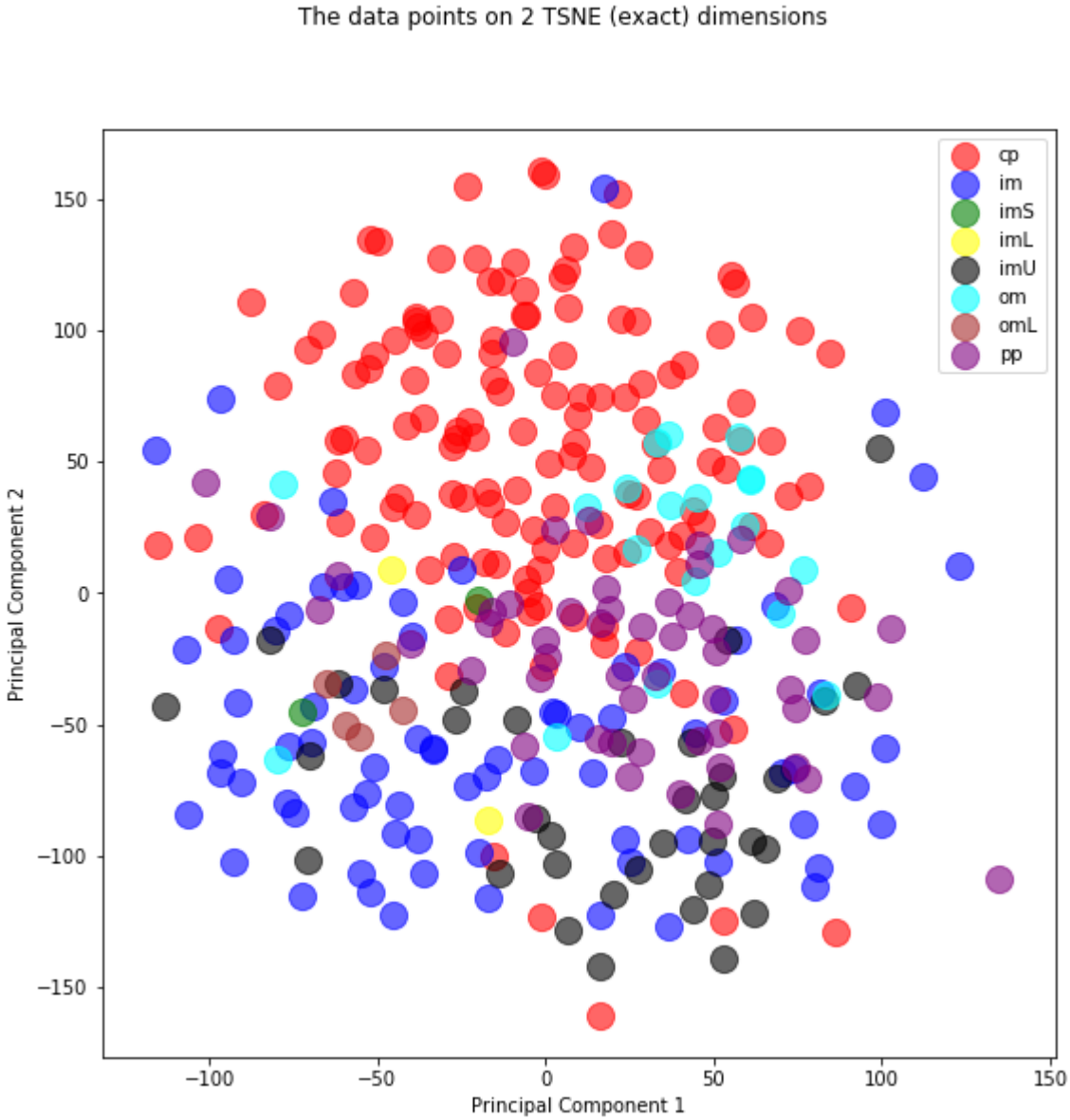


Running np.allclose returns false on the second TSNE showing there is variation between the two calculations of TSNE in 4b and 4c.
Looking at the plot it is also clear that there is some variation between the two runs which shows unlike PCA, TSNE has randomness in it.

Problem 4d t-SNE Barnes-Hut (5 points)

The default t-SNE method of 4b uses the Barnes-Hut approximation. Keeping the other parameters the same as 4b, plot the t-SNE result using the exact method. Which method do you prefer? Compare the average calculation time for the exact method over a number of iterations.

In [18]:
```
starttime = time.time()
ecoli_TSNE = TSNE(n_components=3, method='exact', perplexity=35, verbose=0).fi
t_transform(ecoli0)
draw_plot(ecoli_TSNE, ecoli, 'The data points on 2 TSNE (exact) dimensions')
endtime = time.time()
print("{:03.9f} seconds".format(endtime-starttime))

# AVERAGE TIME TO COMPUTE TSNE PROJECTION
times = []
for i in range(2):
    starttime = time.time()
    ecoli_TSNE = TSNE(n_components=3, method='exact', perplexity=35, verbose=0
).fit_transform(ecoli0)
    endtime = time.time()
    times.append(endtime-starttime)
print('Average Time for TSNE(exact) projection: {:03.9f} seconds'.format(np.me
an(times)))
```




```
13.064965248 seconds
Average Time for TSNE(exact) projection: 13.893085599 seconds
```

The data classes were better seperated using the Barnes-Hut approximation.
The Average time for computations is also around ~13 seconds using both methods.
So, I would prefer the default mehtod over the 'exact' method for this computation.

How many hours did this homework take?

This will not affect your grade. We will be monitoring time spent on homework to be sure that we are not over-burdening students.

In [ ]:
```
16
```

Last step (5 points)

Save this notebook as LastnameFirstnameHW1.ipynb such as MuskElonHW1.ipynb. Create a pdf of this notebook named similarly. Submit both the python notebook and the pdf version to the Canvas dropbox. We require both versions.