CENG 465

Introduction to Bioinformatics

Spring '2019-2020

Assignment 4 Report

Deadline: June 28, 23:59

Submission: via ODTUCLASS

Student Information

Full Name : Yasin Fatih ALPUL
Id Number : 2098739

Goal 1

By using the MEV tool, we upload and import the given dataset to the website. To find the values m and n, we need to cluster the dataset. Using the techniques we have learned in the lectures and course slides, we apply *K-Means clustering*.

We use the MEV tool to achieve that. Using the web interface, we select *Clustering* from the left menu. Then, we select *K-Means clustering*. From there, we select *column* as clustering dimension because we are clustering the samples. We then select 2 as the number of clusters because we are clustering our samples as *healthy* and *diseased*. We then select 1000 as the number of iterations because it would converge more than 100.

When we click *Analyze*, the MEV tool processes the data and gives us two clusters. From there, we can see that $m = 17$ and $n = 13$. In *Table 1*, we can see the clusters and the corresponding cluster for each sample.

Goal 2

For each gene, values have been calculated to find whether the gene is expressed in the healthy and diseased genes or not. Statistical methods have been utilized for this task, namely, the *t-test*.

Each gene has been split their signal intensity values according the whether the tissue is deemed to be healthy or not using the values found in the first part. Thus, we get two sets of values. These values are then *log2* scaled in order to eliminate bias of high intensity valued genes.

Cluster 1	Cluster 2
sample29	sample19
sample28	sample18
sample1	sample17
sample5	sample21
sample4	sample3
sample8	sample10
sample7	sample2
sample30	sample15
sample11	sample9
sample20	sample16
sample12	sample13
sample22	sample14
sample23	sample6
sample24	
sample25	
sample26	
sample27	

Table 1: Clusters

We use the *scipy* module from *Python* programming language to help us calculate the t-test. We used two tailed t test and 0.05 as threshold value. Then, we sort the values according the their p-value of each edge to find the genes. By using the sign of the t statistics and property of symmetry one tailed values are obtained to find top genes for both conditions.

Then, we find the following genes that are expressed in the diseased tissue.

1. `210646_x_at`
2. `213477_x_at`
3. `200884_at`
4. `211710_x_at`
5. `219690_at`
6. `207419_s_at`
7. `218907_s_at`
8. `217740_x_at`
9. `217865_at`
10. `211927_x_at`

We also find the following genes that are expressed in the healthy tissue.

1. 207023_x_at
2. 209154_at
3. 203804_s_at
4. 217758_s_at
5. 201588_at
6. 210065_s_at
7. 211185_s_at
8. 201938_at
9. 203227_s_at
10. 204635_at

We have also implemented the suggested technique of computing the differences of average expression values of healthy and diseased samples for every gene then sort based on the differences.

Below are the genes according to this approach.

1. 205725_at
2. 220542_s_at
3. 204892_x_at
4. 203021_at
5. 210646_x_at
6. 206559_x_at
7. 213477_x_at
8. 201257_x_at
9. 212869_x_at
10. 201492_s_at

1. 211628_x_at
2. 204351_at
3. 201884_at
4. 214385_s_at
5. 209699_x_at
6. 214303_x_at
7. 204151_x_at
8. 210297_s_at
9. 205623_at
10. 207430_s_at

It can be observed that some genes are different. We attribute this phenomena to the fact that simply using the averages have a bias in the genes with high intensity values. For example, if some gene has a consistent signal intensities for diseased samples as around 70 each and around $120 - 150$ healthy samples. This gene can be said to be highly expressed consistently. However, if some other gene has its intensity values in the range $30000 - 40000$, the difference of means would be in the order of $100s$ even if the expressiveness would be inconclusive. Thus, this gene would be included in the latter approach.

Extra Notes

Below is a portion of the code I have written to find the genes using the suggested average difference approach.

```
    labels = [1, 0, 0, 1, ...]
    for k in range(len(data)):
    L1, L2 = [], []
    for i in range(len(data[0])):
      if labels[i] == 1:
           L1.append(data[k, i])
      else:
           L2.append(data[k, i])
    pvalues[genid[k]] = (L1, L2)

l = [(np.mean(pvalues[a][0]) - np.mean(pvalues[a][1]), a) for a in pvalues]

l.sort()
```

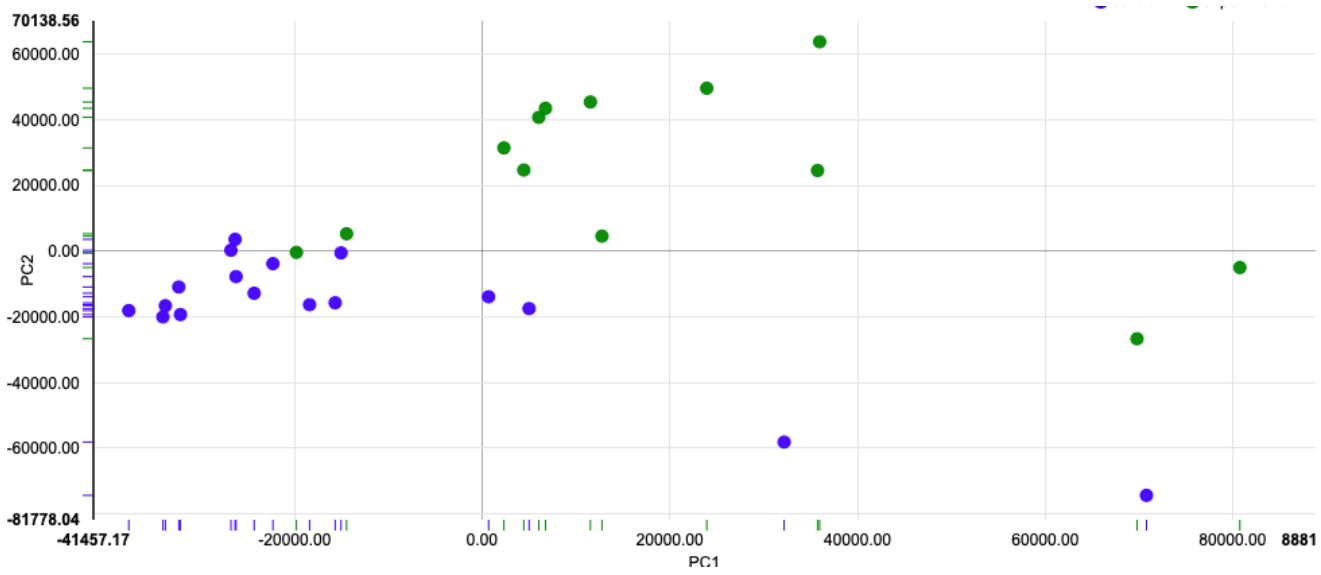
Below is the portion of the code I have written to find the genes using the t-test.

```
    labels = [1, 0, 0, 1, ...]
    pvals1, pvals2 = {}, {}
    for k in range(len(data)):
    L1, L2 = [], []
    for i in range(len(data[0])):
      if labels[i] == 1:
           L1.append(data[k, i])
      else:
           L2.append(data[k, i])
    t, p = st.ttest_ind(np.log2(L1), np.log2(L2), equal_var = False)
    # One paired test results are obtained in here
    if t < 0 and p/2 < .05:
        pvals1[genid[k]] = (p/2, L1, L2)
    elif t > 0 and p/2 < .05:
        pvals2[genid[k]] = (p/2, L1, L2)

l = [(pvals1[a][0], a, np.mean(pvals1[a][1]), np.mean(pvals1[a][2])) for a in pvals1]
m = [(pvals2[a][0], a, np.mean(pvals2[a][1]), np.mean(pvals2[a][2])) for a in pvals2]

l.sort()
m.sort()
```

Below is the PCA (principal component analysis) plot from MEV.



At the first glance, we see a $26-4$ clustering from the plot. This is also reflected in my initial code which included the code `sklearn.cluster.KMeans(n_clusters=2).fit(data)`. This code also clusters the samples as $26-4$ similar to the clusters in upper-left and lower-right clusters in the plot. However, I later decided that this clustering lacked the scaling factor and had a bias for high signal intensity values. Then, I added to my code:
`sklearn.preprocessing.StandardScaler().fit_transform(data)`. Then used `KMeans` and it was $17-13$. I used this clustering since it was also the same as the clustering I get from the MEV tool. The $26-4$ clustering had the samples $sample10, sample19, sample25, sample27$ for 4 and the rest for 26.