Cairo University

Faculty of Engineering

Systems and Biomedical Department

Biostatistics Final Project

Contributors:

Ashar Seif Al-Naser Saleh

Alaa Alaah Esaam Mera

Aya Ehab Saleh

Mayar Tarek Hassan

SBE304_Biostatistics

Dr.Ibrahim Mohamed Ibrahim

19 January, 2021

Statistical Analysis on genes in healthy and cancerous tissues

1. Introduction

In this report, we concern studying two sets of data of the same genes in which samples are taken from healthy tissues and cancerous tissues. Our purpose is to apply data analysis on the gene expression (GE) data using statistics in order to find out the effect of cancer type Lung Squamous Cell Carcinoma (LUSC) on such genes, and define which genes their expression levels changed due cancer.

2. Methodology

In this statistical study we aim to:

- Compute the **correlation** between the normal samples and the diseased samples for each gene.
- **Hypothesis Testing.** Infer the differentially expressed genes (DEGs); the genes whose expression level differ from one condition (healthy) to another (diseased).
    - The test is applied for two cases: samples are paired and Samples are independent.
    - Apply the FDR multiple tests correction method.

2.1 Correlation

Correlation is a measure of the strength of a linear relationship between two quantitative variables. Positive correlation is a relationship between two variables in which both variables move in the same direction. Negative correlation is a relationship where one variable increases as the other decreases, and vice versa.

We'd apply Pearson's Correlation which is the correlation coefficient r measures the strength and direction of a linear relationship. Values between -1 and 1 denote the strength of the correlation.

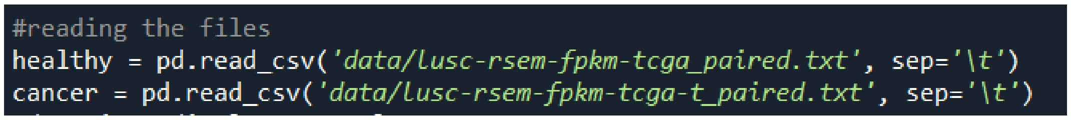
But before the correlation calculations we need to apply some filtration of data given.

The data is given as tab-separated files in which rows represent genes expressions in samples and the 1st 2 columns have the gene name by HUGO gene nomenclature committee and the gene ID, and the rest of columns are the samples.

In this analysis we've used python software.

1. First step is to **open** our files in our used software to get access to the data. In this step we used pandas package using read method

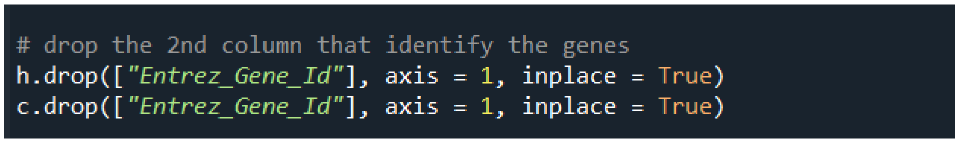
(Read a tab-separated values (tsv) file into DataFrame.)



```
#reading the files
healthy = pd.read_csv('data/lusc-rsem-fpkm-tcga_paired.txt', sep='\t')
cancer = pd.read_csv('data/lusc-rsem-fpkm-tcga-t_paired.txt', sep='\t')
```

After this step using shape method (healthy.shape) we found out that the file is (19648,25)

Drop the column that identify the genes (gene_id) in order to apply our test only on the required data. The drop () function is used to drop specified labels from rows or columns. Removes rows or columns by specifying label names and corresponding axis



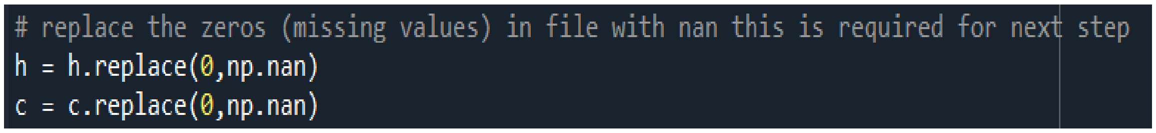
```
# drop the 2nd column that identify the genes
h.drop(["Entrez_Gene_Id"], axis = 1, inplace = True)
c.drop(["Entrez_Gene_Id"], axis = 1, inplace = True)
```

We can also drop the column having gene names but we needed the names in further steps. Now our data is (19648, 51)

Filtration

In our data files there are some missing data (zeros) that we can't accept and may be misleading concerning our results so, we removed the rows having more than 50% of missing data in both files. After applying this step we found out that 11.76% of data rows are removed and we choose to accept it In order to achieve that we performed 3 operations:

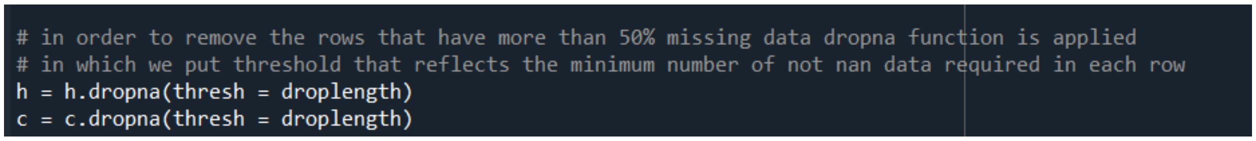
1. **Replacing** all missing data with NAN values using numpy package(replace method)



```
# replace the zeros (missing values) in file with nan this is required for next step
h = h.replace(0,np.nan)
c = c.replace(0,np.nan)
```

2. We choosed a threshold to the accepted rows using **dropna** method.

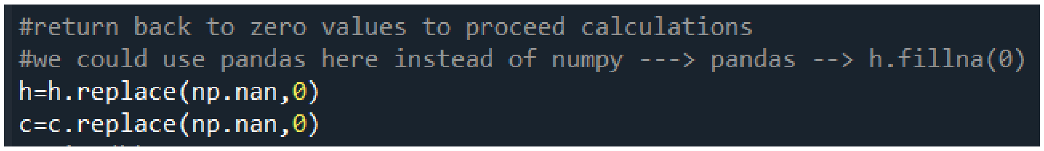
Pandas dropna() method allows the user to analyze and drop Rows/Columns with Null values in different ways. thresh: thresh takes integer value which tells minimum amount of not nan values we need.



```
# in order to remove the rows that have more than 50% missing data dropna function is applied
# in which we put threshold that reflects the minimum number of not nan data required in each row
h = h.dropna(thresh = droplength)
c = c.dropna(thresh = droplength)
```

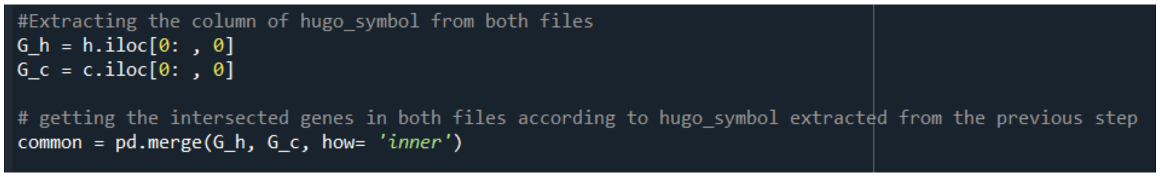
Now our healthy data is (17675, 51) and cancerous data is(17782,51)

3. To proceed our calculations we need to get back the zeros using replace method again or by fillna method by pandas package.



```
#return back to zero values to proceed calculations
#we could use pandas here instead of numpy ---> pandas --> h.fillna(0)
h=h.replace(np.nan,0)
c=c.replace(np.nan,0)
```

4. Now what happened may result in dropping rows of certain gene from one file and not the other and this is not accepted so we merge the column of gene names from both files and take only the intersection between them and those intersected rows will be our data

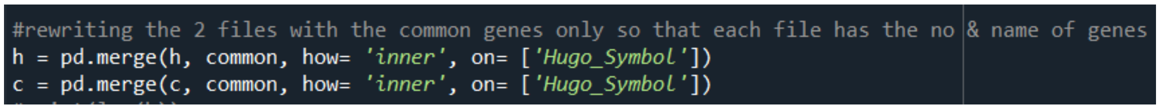


```
#Extracting the column of hugo_symbol from both files
G_h = h.iloc[0: , 0]
G_c = c.iloc[0: , 0]

# getting the intersected genes in both files according to hugo_symbol extracted from the previous step
common = pd.merge(G_h, G_c, how= 'inner')
```

Here we used iloc method from pandas package to extract gene names then using merge method

Inner: Use intersection of keys

Our new files are ready for our tests



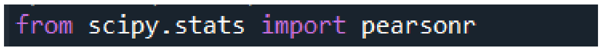
```
#rewriting the 2 files with the common genes only so that each file has the no & name of genes
h = pd.merge(h, common, how= 'inner', on= ['Hugo_Symbol'])
c = pd.merge(c, common, how= 'inner', on= ['Hugo_Symbol'])
```

Now our data (17337, 51)

Those were basic steps made our data accepted to apply statistical test.

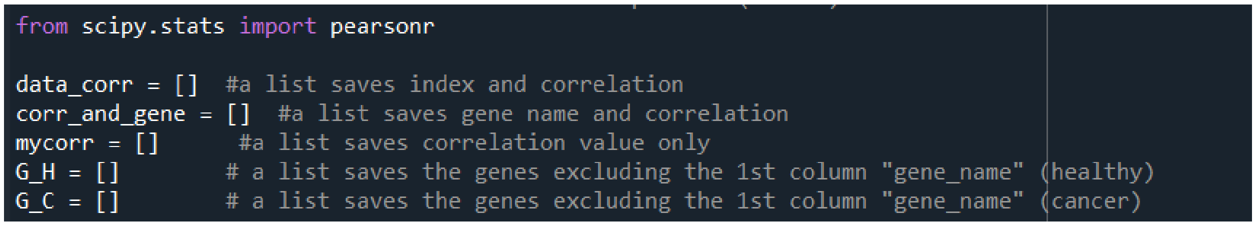
A. Correlation analysis

Using scipy.stats package and pearson method



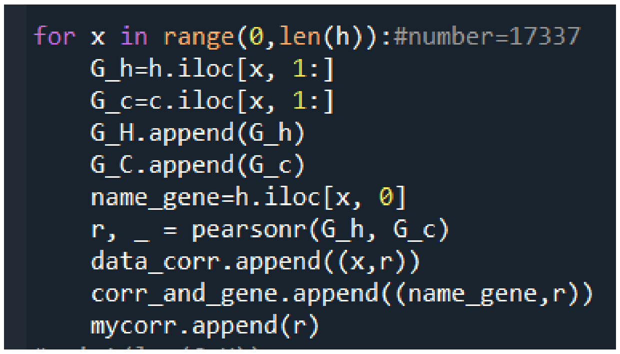
```
from scipy.stats import pearsonr
```

At first we prepared needed lists



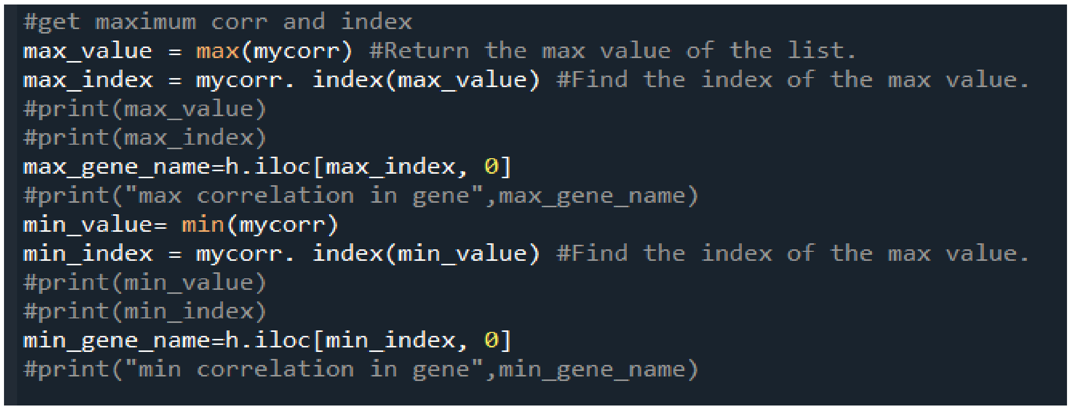
```
from scipy.stats import pearsonr

data_corr = []  #a list saves index and correlation
corr_and_gene = []  #a list saves gene name and correlation
mycorr = []      #a list saves correlation value only
G_H = []        # a list saves the genes excluding the 1st column "gene_name" (healthy)
G_C = []        # a list saves the genes excluding the 1st column "gene_name" (cancer)
```

1. saving expression levels for each gene, gene name ,correlation and corr_and_gene in lists in order to use it in further steps.



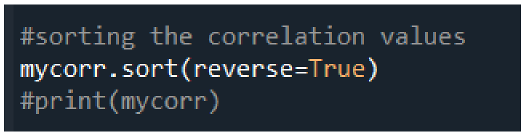
```
for x in range(0,len(h)):#number=17337
    G_h=h.iloc[x, 1:]
    G_c=c.iloc[x, 1:]
    G_H.append(G_h)
    G_C.append(G_c)
    name_gene=h.iloc[x, 0]
    r, _ = pearsonr(G_h, G_c)
    data_corr.append((x,r))
    corr_and_gene.append((name_gene,r))
    mycorr.append(r)
```

2. Getting maximum and minimum correlations and their corresponding indices to retrieve their names



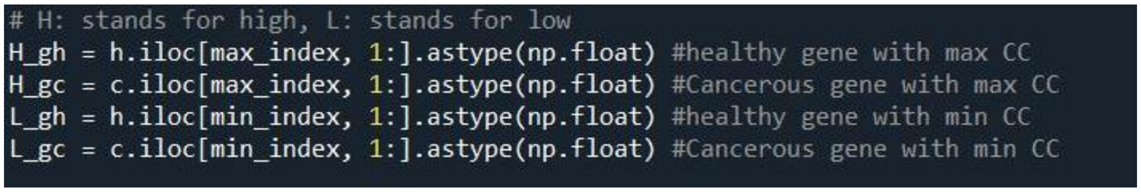
```
#get maximum corr and index
max_value = max(mycorr) #Return the max value of the list.
max_index = mycorr. index(max_value) #Find the index of the max value.
#print(max_value)
#print(max_index)
max_gene_name=h.iloc[max_index, 0]
#print("max correlation in gene",max_gene_name)
min_value= min(mycorr)
min_index = mycorr. index(min_value) #Find the index of the max value.
#print(min_value)
#print(min_index)
min_gene_name=h.iloc[min_index, 0]
#print("min correlation in gene",min_gene_name)
```

3. Ranking correlations



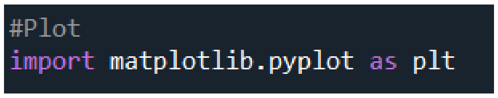
```
#sorting the correlation values
mycorr.sort(reverse=True)
#print(mycorr)
```

4. Preparing the data that we are going to use in plotting, we used astype(np.float) method to make sure that the data will be suitable to be used in np.polyfiy() method in the curve fitting step.



```
# H: stands for high, L: stands for low
H_gh = h.iloc[max_index, 1:].astype(np.float) #healthy gene with max CC
H_gc = c.iloc[max_index, 1:].astype(np.float) #Cancerous gene with max CC
L_gh = h.iloc[min_index, 1:].astype(np.float) #healthy gene with min CC
L_gc = c.iloc[min_index, 1:].astype(np.float) #Cancerous gene with min CC
```

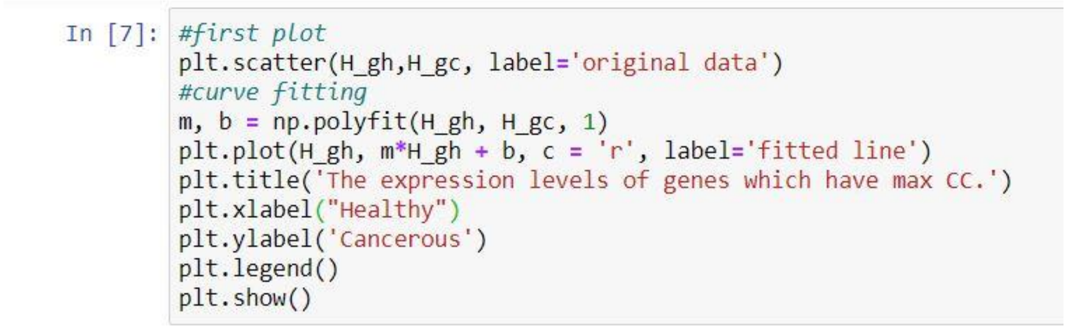
5. Plotting the expression levels of the above two genes.
Using matplotlib package ,pyplot and scatter method



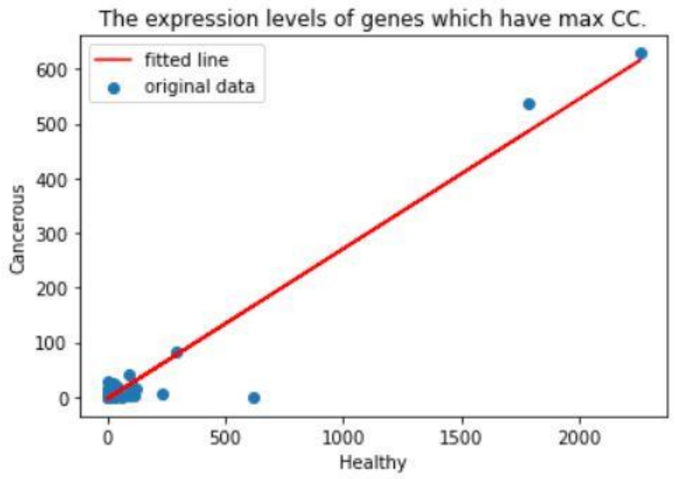
```
#Plot
import matplotlib.pyplot as plt
```

- Performing two plots

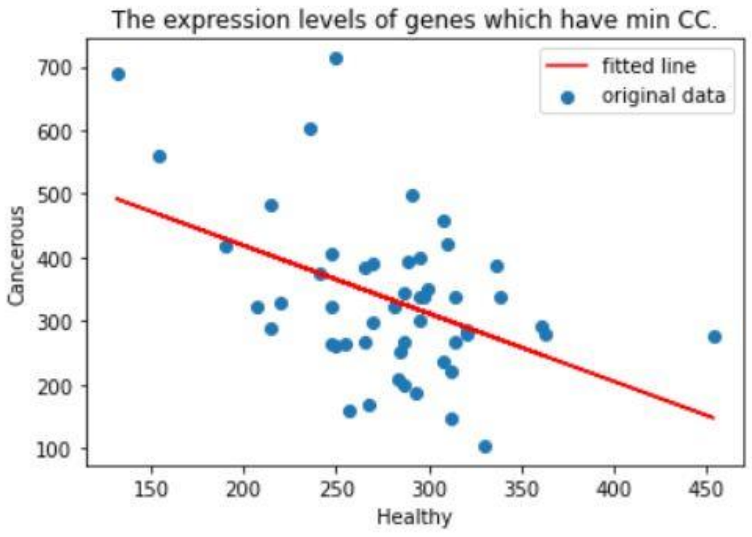
First one between healthy and cancerous genes that gives maximum correlation and the second one for those who gave minimum negative correlation.



```
In [7]: #first plot
        plt.scatter(H_gh,H_gc, label='original data')
        #curve fitting
        m, b = np.polyfit(H_gh, H_gc, 1)
        plt.plot(H_gh, m*H_gh + b, c = 'r', label='fitted line')
        plt.title('The expression levels of genes which have max CC.')
        plt.xlabel("Healthy")
        plt.ylabel('Cancerous')
        plt.legend()
        plt.show()
```



```
In [8]: #second plot
        plt.scatter(L_gh,L_gc, label='original data')
        #curve fitting
        m, b = np.polyfit(L_gh, L_gc, 1)
        plt.plot(L_gh, m*L_gh + b, c = 'r', label='fitted line')
        plt.title('The expression levels of genes which have min CC.')
        plt.xlabel("Healthy")
        plt.ylabel('Cancerous')
        plt.legend()
        plt.show()
```



```
In [4]: print('max CC: ',max_value)
        print('index of max CC: ',max_index)
        print("max correlation in gene",max_gene_name)
        print('min CC: ',min_value)
        print('index of min CC: ', min_index)
        print("min correlation in gene",min_gene_name)
```

```
max CC:  0.9690441442970706
index of max CC:  10829
max correlation in gene AREGB
min CC:  -0.4528072785247083
index of min CC:  12974
min correlation in gene FAM222B
```

From the analysis above we get that:
The correlation results are:

- highest positive correlation coefficient in gene AREGB and its value equals to 0.9690441442970706
- lowest negative correlation coefficient in gene FAM222B and its value equals to -0.4528072785247083
- the results and plots indicates strong positive correlation and relatively weak negative correlation

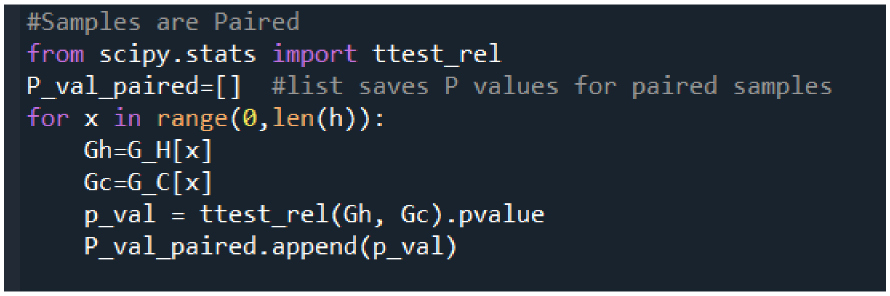
2.2 Hypothesis test

In this paired test we put null hypothesis represents that the gene expression doesn’t change from healthy to diseased tissue and the alternative hypothesis is the gene expression changes from healthy to diseased tissue. While in the independent test the null hypothesis represents the expression values of the samples of gene doesn’t differ from population and alternative hypothesis represents that the gene expression values in samples differ from that of population and this test applied on the healthy and cancerous data independently.

As mentioned before we apply filtration the same way we did before correlation. Regarding this test we will perform it for both Paired and independent samples.

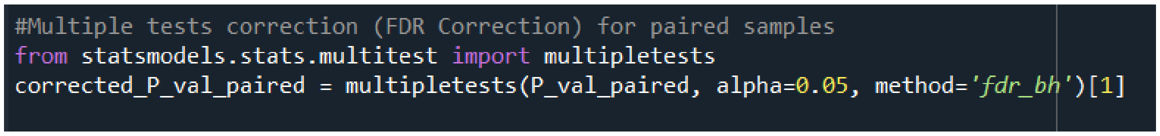
B.1 for paired samples

1. Using (ttest_rel) method from scipy package we apply test over our samples from both files and save resulted p-values in a list called (p_val_paired).



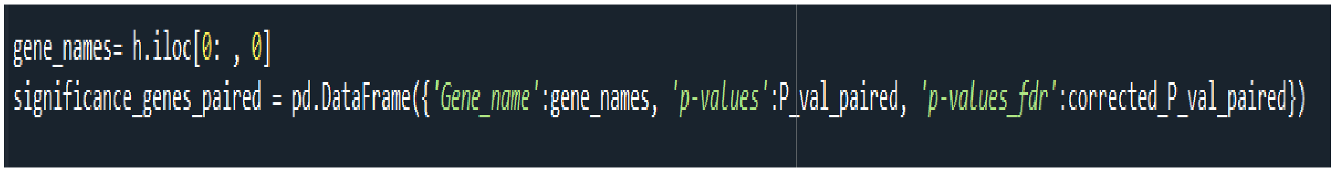
```python
#Samples are Paired
from scipy.stats import ttest_rel
P_val_paired=[]  #list saves P values for paired samples
for x in range(0,len(h)):
    Gh=G_H[x]
    Gc=G_C[x]
    p_val = ttest_rel(Gh, Gc).pvalue
    P_val_paired.append(p_val)
```

2. using (statsmodels.stats.multitest.multipletest) method we did a FDR correction taking alpha=0.05 and Applying the FDR multiple tests correction method



```python
#Multiple tests correction (FDR Correction) for paired samples
from statsmodels.stats.multitest import multipletests
corrected_P_val_paired = multipletests(P_val_paired, alpha=0.05, method='fdr_bh')[1]
```

3. getting gene names again by iloc and making dataframe of paired samples with gene names,p_val and corrected_p_val after applying FDR correction.



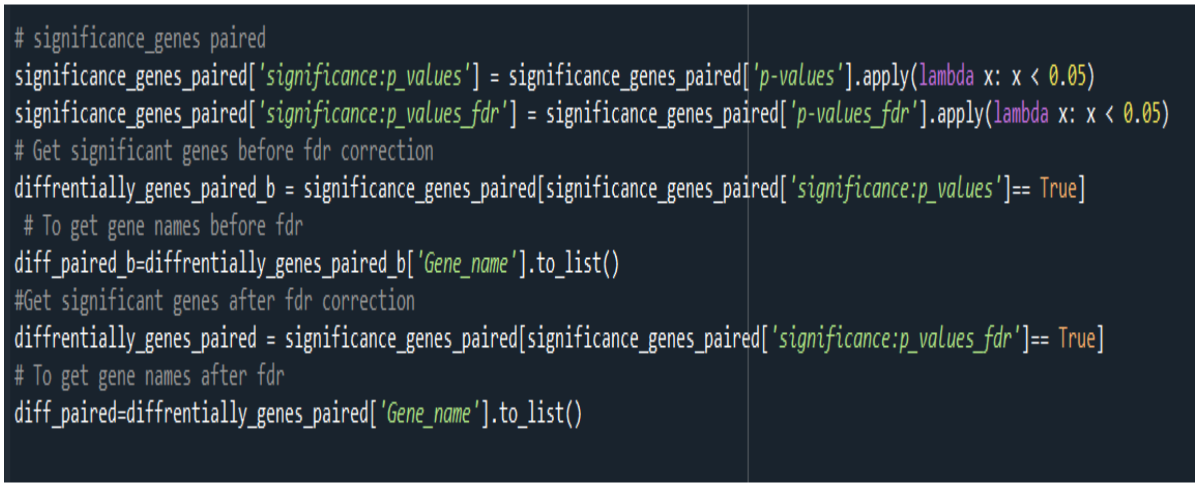
```python
gene_names= h.iloc[0: , 0]
significance_genes_paired = pd.DataFrame({'Gene_name':gene_names, 'p-values':P_val_paired, 'p-values_fdr':corrected_P_val_paired})
```

4. then we get the significant genes from the above dataset in which pvalues before and after FDR correction are given true status when they are less than alpha value setted 0.05 indicating rejection of null hypothesis and the others are false indecating fail to reject null hypothesis.

Then we saved the names of the differentialy expressed genes (DEGS) in a list and we found out that:

The number of DEGS in paired case before FDR correction is 12724

The number of DEGS in paired case after FDR correction is 12410

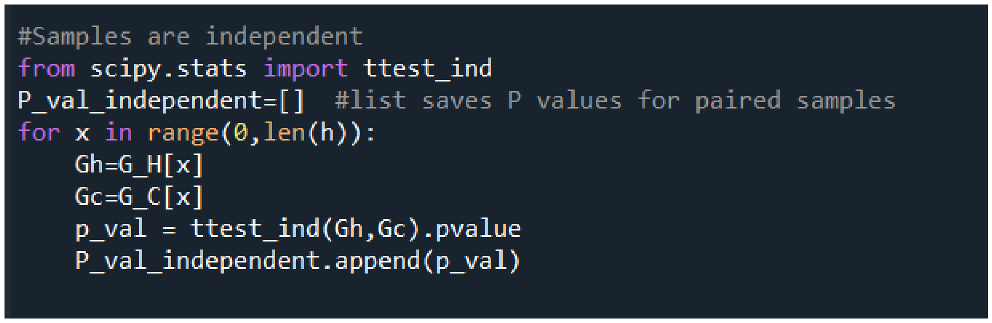


```python
# significance_genes paired
significance_genes_paired['significance:p_values'] = significance_genes_paired['p-values'].apply(lambda x: x < 0.05)
significance_genes_paired['significance:p_values_fdr'] = significance_genes_paired['p-values_fdr'].apply(lambda x: x < 0.05)
# Get significant genes before fdr correction
diffrentially_genes_paired_b = significance_genes_paired[significance_genes_paired['significance:p_values']== True]
 # To get gene names before fdr
diff_paired_b=diffrentially_genes_paired_b['Gene_name'].to_list()
#Get significant genes after fdr correction
diffrentially_genes_paired = significance_genes_paired[significance_genes_paired['significance:p_values_fdr']== True]
# To get gene names after fdr
diff_paired=diffrentially_genes_paired['Gene_name'].to_list()
```

B.2 #Samples are independent

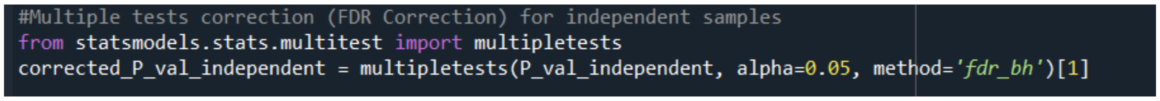
Same previous steps as paired samples

1.



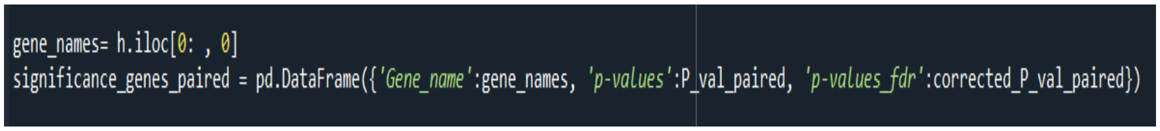
```python
#Samples are independent
from scipy.stats import ttest_ind
P_val_independent=[]  #list saves P values for paired samples
for x in range(0,len(h)):
    Gh=G_H[x]
    Gc=G_C[x]
    p_val = ttest_ind(Gh,Gc).pvalue
    P_val_independent.append(p_val)
```

2.



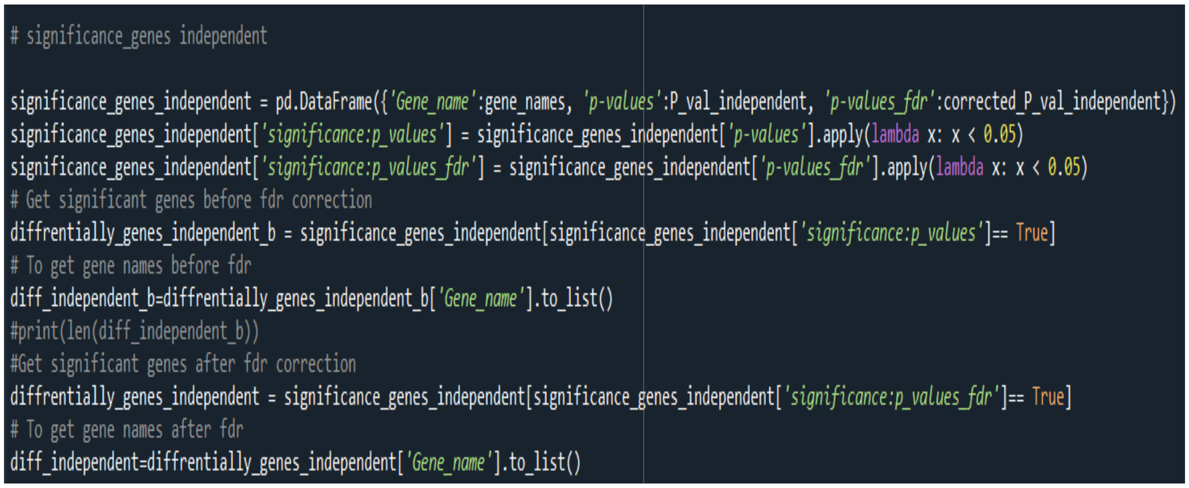
```python
#Multiple tests correction (FDR Correction) for independent samples
from statsmodels.stats.multitest import multipletests
corrected_P_val_independent = multipletests(P_val_independent, alpha=0.05, method='fdr_bh')[1]
```

3.



```python
gene_names= h.iloc[0: , 0]
significance_genes_paired = pd.DataFrame({'Gene_name':gene_names, 'p-values':P_val_paired, 'p-values_fdr':corrected_P_val_paired})
```

4.


```
# significance_genes independent

significance_genes_independent = pd.DataFrame({'Gene_name':gene_names, 'p-values':P_val_independent, 'p-values_fdr':corrected_P_val_independent})
significance_genes_independent['significance:p_values'] = significance_genes_independent['p-values'].apply(lambda x: x < 0.05)
significance_genes_independent['significance:p_values_fdr'] = significance_genes_independent['p-values_fdr'].apply(lambda x: x < 0.05)
# Get significant genes before fdr correction
diffrentially_genes_independent_b = significance_genes_independent[significance_genes_independent['significance:p_values']== True]
# To get gene names before fdr
diff_independent_b=diffrentially_genes_independent_b['Gene_name'].to_list()
#print(len(diff_independent_b))
#Get significant genes after fdr correction
diffrentially_genes_independent = significance_genes_independent[significance_genes_independent['significance:p_values_fdr']== True]
# To get gene names after fdr
diff_independent=diffrentially_genes_independent['Gene_name'].to_list()
```

then we get the significant genes from the above dataset in which pvalues before and after FDR correction are given true status when they are less than alpha value setted 0.05 indicating rejection of null hypothesis and the others are false indecating fail to reject null hypothesis.

Then we saved the names of the differentialy expressed genes (DEGS) in a list and we found out that:

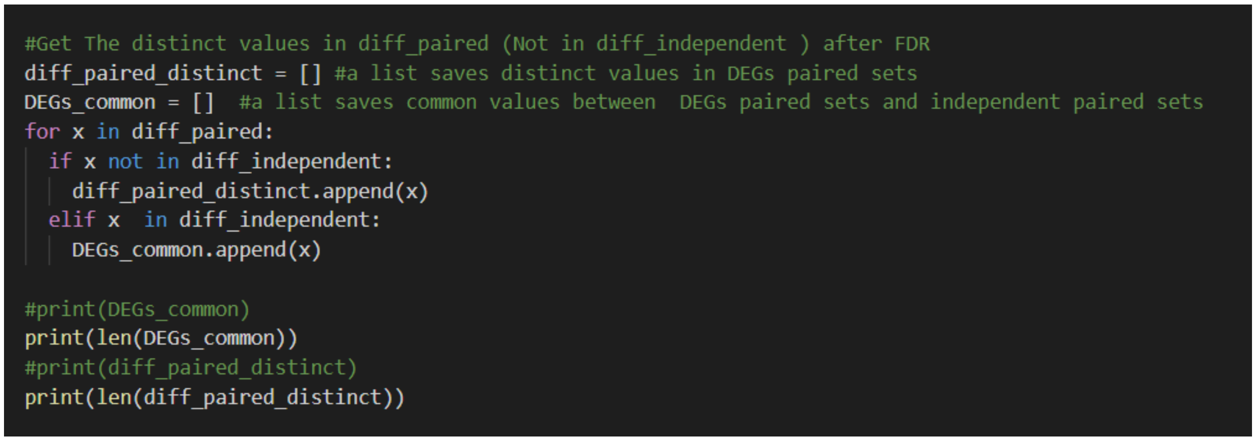
The number of DEGS in independent case before FDR correction is 12631

The number of DEGS in independent case after FDR correction is 12320

Comparing between paired and independent DEGS sets:

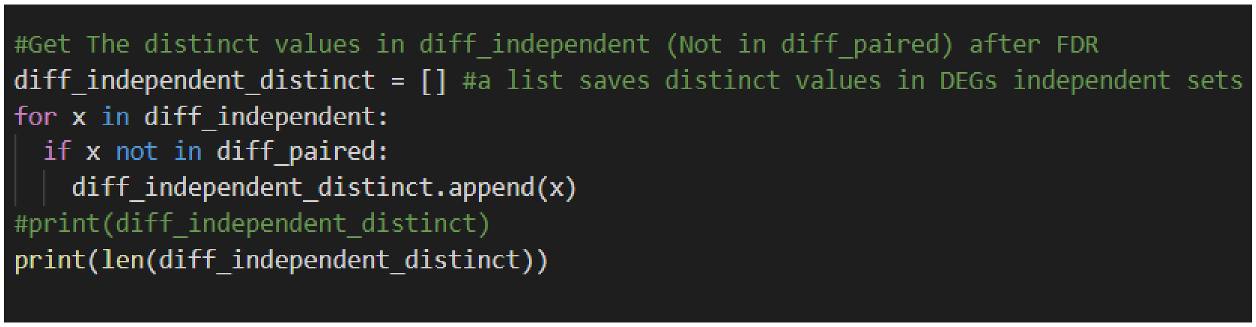
In this step we compared between the paired DEGs set saved in diff_paired list and the independent DEGs set saved in diff_independent list **after the FDR correction** to identify the common differentially genes between the two sets and also distinguish the distinct differentialy genes in both of them .

- Frist , We get the distict values in the paired DEGs set which do not exist in the independent DEGs set and saved these values in the diff_paired_distinct list, we also get the common values between the two sets and saved them on the DEGs_common list



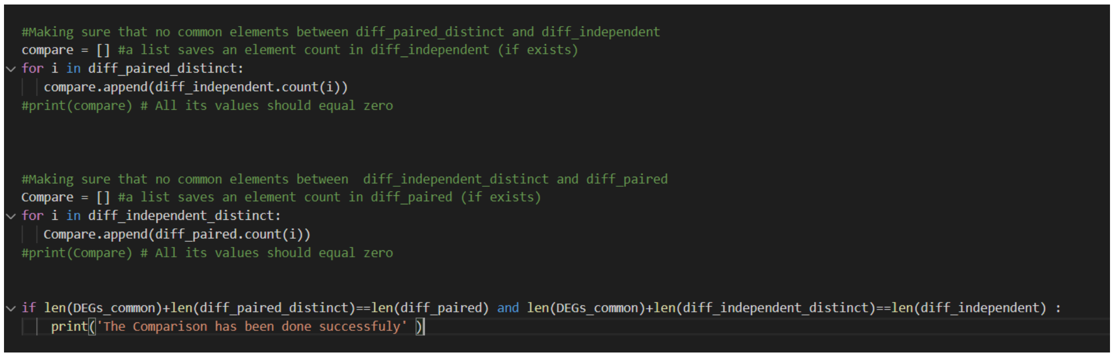
```python
#Get The distinct values in diff_paired (Not in diff_independent ) after FDR
diff_paired_distinct = [] #a list saves distinct values in DEGs paired sets
DEGs_common = []  #a list saves common values between  DEGs paired sets and independent paired sets
for x in diff_paired:
  if x not in diff_independent:
    diff_paired_distinct.append(x)
  elif x  in diff_independent:
    DEGs_common.append(x)

#print(DEGs_common)
print(len(DEGs_common))
#print(diff_paired_distinct)
print(len(diff_paired_distinct))
```

- Second , We get the distict values in the independent DEGs set which do not exist in the paired DEGs set and saved these values in the diff_paired_independent list



```python
#Get The distinct values in diff_independent (Not in diff_paired) after FDR
diff_independent_distinct = [] #a list saves distinct values in DEGs independent sets
for x in diff_independent:
  if x not in diff_paired:
    diff_independent_distinct.append(x)
#print(diff_independent_distinct)
print(len(diff_independent_distinct))
```

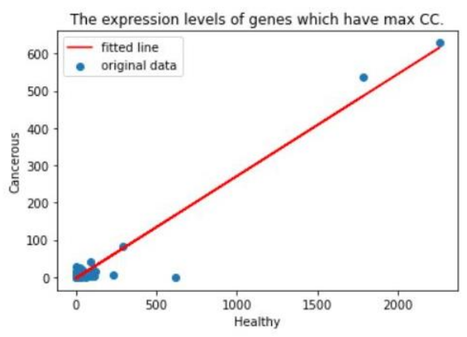
- And last , we make small tests on the results we got to make sure that no common differentially expressed genes between the two lists (diff_paired_distinct and diff_paired_independent) and that both of them added to the common list will equal to the original DEGs set .

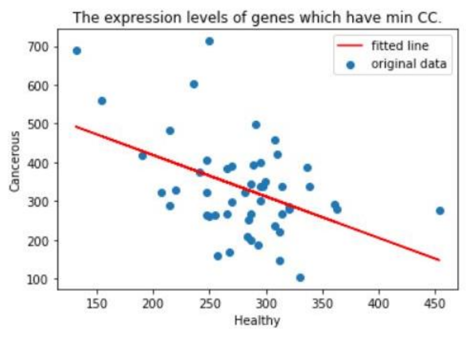


```python
#Making sure that no common elements between diff_paired_distinct and diff_independent
compare = [] #a list saves an element count in diff_independent (if exists)
for i in diff_paired_distinct:
  compare.append(diff_independent.count(i))
#print(compare) # All its values should equal zero



#Making sure that no common elements between  diff_independent_distinct and diff_paired
Compare = [] #a list saves an element count in diff_paired (if exists)
for i in diff_independent_distinct:
  Compare.append(diff_paired.count(i))
#print(Compare) # All its values should equal zero


if len(DEGs_common)+len(diff_paired_distinct)==len(diff_paired) and len(DEGs_common)+len(diff_independent_distinct)==len(diff_independent) :
   print('The Comparison has been done successfuly' )
```

Results:

1. Highest positive CC and the lowest negative CC and the names of these two genes and plots





- highest positive correlation coefficient in gene AREGB and its value equals to 0.9690441442970706
- lowest negative correlation coefficient in gene FAM222B and its value equals to -0.4528072785247083
- the results and plots indicates strong positive correlation and relatively weak negative correlation

2. Paired case hypothesis test before and after FDR correction

- The number of DEGS in paired case before FDR correction is 12724 saved in list called (diff_paired_b)
- The number of DEGS in paired case after FDR correction is 12410 saved in list called (diff_paired)

3. Independent case hypothesis test before and after FDR correction

- The number of DEGS in independent case before FDR correction is 12631 saved in list called (diff_independent_b)
- The number of DEGS in independent case after FDR correction is 12320 saved in list called (diff_independent)

4. Compare the two DEGs sets.

- The number of distinct DEGs in the paired samples is 169
- The number of distinct DEGs in the independent samples is 79
- The number of common DEGs between the paired and the independent samples is 12241

Conclusion

- Correlation: The correlation of gene AREGB (Max correlation) and its value equals to 0.9690441442970706 represents strong positive correlation and that indicate that expression level of gene increases in healthy sample as in cancerous sample and the correlation of gene FAM222B (Min correlation) and its value equals to -0.4528072785247083 represents relatively weak negative correlation and that indicate that expression level of gene inversely changes between healthy and cancerous sample (if increases in healthy ,it decreases in cancerous)

- Hypothesis test: The results of the number of DEGs in the paired test is more than that in independent test and that indicates more null hypothesis rejection in that case (paired), The results of the number of DEGs after the FDR correction in case of paired or independent samples less than the number of DEGs before the FDR correction due to that FDR decreases type I error then number of true null hypothesis rejected decreases so the overall rejected null hypothesis values decreases.

- Null hypothesis is rejected and the alternative hypothesis is accepted

Contribution

Filtration(mayar&aya)
Correlation calculation(alaa&mayar)
Plotting(Alaa&aya)
Hypothesis calculation (ashar&alaa),FDR correction (mayar)
Comparison(ashar)