PyMicroArray Ver 1.0 Documentation

This code base contains various programs for data retireval, cleaning filtering and analysis.

Find the microRNAs regulating the target genes

> PyMicroArray/to find miRNAs of target genes from mirtarbase/

To find the microRNAs regulating the target genes from the MIRTARBASE local database. Given the MIRTARBASE local database and a gene list, this program can findout all down regulated genes from the MIRTARBASE which is in the given gene list.

```
import pandas as pd
data       =     pd.read_csv('hsa_MTI_MIRTARBASE_DB_7.0.csv').iloc[:,[0,1,3,4]]
genes      =     pd.read_csv('RR_total_downregulated_gene_list.txt').values.flatten()
result     = data[data['Target Gene'].isin(genes)]
result.to_csv('DownRegulatedGene_MiRNA.csv', sep=',')
```

The MIRTARBASE local database is available in the file **hsa_MTI_MIRTARBASE_DB_7.0.csv**. We can select the only required fields by specifying the columns needed in the *iloc* field. The result will be saved as a CSV file as given in the last line.

Find Common genes between two datasets

> PyMicroArray/to extract TSG & oncogenes from database

This code snippet finds common genes between two data sets and saves the same to a text file. The name of column in which gene name come can be specified in the program. If no common element is found then that is displayed as a message.

```
import pandas as pd

data1 = pd.read_csv("Human_TSGs_TSG2.0_database.csv")["GeneSymbol"]
data2 = pd.read_csv("DEGs_SRR926257.csv")["gene_id"]

common = set(data1).intersection(data2)
outfile = open('TSGs_in_DEGs_SRR926257.txt', 'w')
if bool(common):
    outfile.write("\n".join(common))
else:
    outfile.write("No Common Genes Found")
outfile.close()
```

To filter dataset with a key matching

> PyMicroArray/toFilterDFwithKey

This code can filter a dataset with a coloumn matching a key term. Here we want to extract the rows which match/contain the term *hsa-miR-103a-*. The extracted data is saved into a file in CSV format.

```
import pandas as pd
data = pd.read_csv("Mirtarbase_human_interactions_database.csv")
result = pd.DataFrame()
match  = data.miRNA.str.contains('^hsa-miR-103a-')
result = data[match]
result.to_csv('mirtarbase_miR-103a.csv', sep=',')
```

To retrieve data from RegNetwork dataset online.

> PyMicroArray/RegNetworkParser

Given a list of items in an excel file (*miRNA_to_find_TF.xlsx*) whose data is to be fetched from Reg Network online. This program does the same and saves each result as a CSV file.

```
import pandas as pd
import urllib
import time

testfile = urllib.URLopener()
searchterms = pd.read_excel(open('miRNA_to_find_TF.xlsx','rb'), sheetname=0)['miRNA']

for item in searchterms:
    time.sleep(5)
    #URL = 'http://www.regnetworkweb.org/export.jsp?format=csv&sql=SELECT+*+FROM+human+WHERE+%28%28UPPER
    URL  = 'http://www.regnetworkweb.org/export.jsp?format=csv&sql=SELECT+*+FROM+human+WHERE+%28%28UPPER
    testfile.retrieve(URL, item+".csv")
```

To find average expression levels

> PyMicroArray/proteome_data

This finds multiple occurances of a gene and finds the average value of control and treated expression for each gene. The result is exported into a spreadsheet.

```
import pandas as pd
import numpy as np
from pandas import ExcelWriter

data        =         pd.read_excel(open('Proteome_data_cell_fractions_combined.xlsx','rb'), sheetname=0,
columns     =         ['UNIPROT', 'SYMBOL', 'GENENAME', 'UNTREATED', 'VEGF', 'Ratio', 'FC']
result         =         pd.DataFrame(columns = columns)
genes         =         list(set(data['SYMBOL'].values))

for gene in genes:
    row = data[data['SYMBOL']==gene]
    repeat = row.shape[0]
    if repeat>1:
        Vc = np.mean(row['VEGF'])
        V6 = np.mean(row['UNTREATED'])
        if V6==0:
            Ratio ='Inf'
            Fc    ='Inf'
        else:
            Ratio = Vc/V6
            Fc = np.log2(Ratio)
        value = [row['UNIPROT'].values[0], gene, row['GENENAME'].values[0], V6, Vc, Ratio, Fc]
        result.loc[len(result)] =  value
    else:
        row = row.values[0].tolist()
        if row[-2]==0:
            Ratio ='Inf'
            Fc    ='Inf'
        else:
            Ratio = row[-1]/row[-2]
            Fc = np.log2(Ratio)
        row = row+[Ratio, Fc]
        result.loc[len(result)] = row

writer = ExcelWriter('Proteome_Profile_MeanCalculated_FC.xlsx')
result.to_excel(writer,'Sheet1')
writer.save()
```

Normalize and classify into UP/Down regulated

> PyMicroArray/proteome_data

This code adds a flag 'Regulated' to the output to classify genes based on their regulation level into Up/Down. A Z score cutoff of 1.65 equivalent to 95% CI is used for this purpose.

```
import pandas as pd

cutoff                  =   1.65
table                   =    pd.read_csv('plosone.csv')
data                    =     table['HUVEC']
table['zscore']   =   (data - data.mean())/data.std(ddof=0)

table.loc[table['zscore'] > cutoff, 'Regulation'] = 'UP'
table.loc[table['zscore'] < -1*cutoff, 'Regulation'] = 'Down'
table.to_csv('Regulation.csv', sep=',', index = False)
```

Filter specific gene details from NCBI dataset

> PyMicroArray/NCBIOfflineSelector

Given a set of genes check for its presece in local NCBI data set and retreive only necessary details. Here *Homo_sapiens.csv* is the local NCBI data. *GSE837_DEGs_GSEA.xlsx* contains the list of genes to be checked.

```
import pandas as pd

data = pd.read_csv('Homo_sapiens.csv', low_memory=False)
data = data[['Symbol', 'Synonyms', 'type_of_gene']]
genes = pd.read_excel(open('GSE837_DEGs_GSEA.xlsx','rb'), sheetname=0)['Total_DEG_GSE837_GSEA'].values
col = ['Symbol', 'Synonyms', 'type_of_gene']
result = pd.DataFrame(columns = col)

for index, row in data.iterrows():
    if row.values[1]!='-':
        line = row.values[0]+'|'+row.values[1]
    else:
        line = row.values[0]
    line = line.split('|')

    for gene in genes:
        if gene in line:
            result.loc[len(result)] = row.values

writer = pd.ExcelWriter('GSE837_DEGs_GSEA_Genetype_Out.xlsx')
result.to_excel(writer,'Sheet1')
writer.save()
```