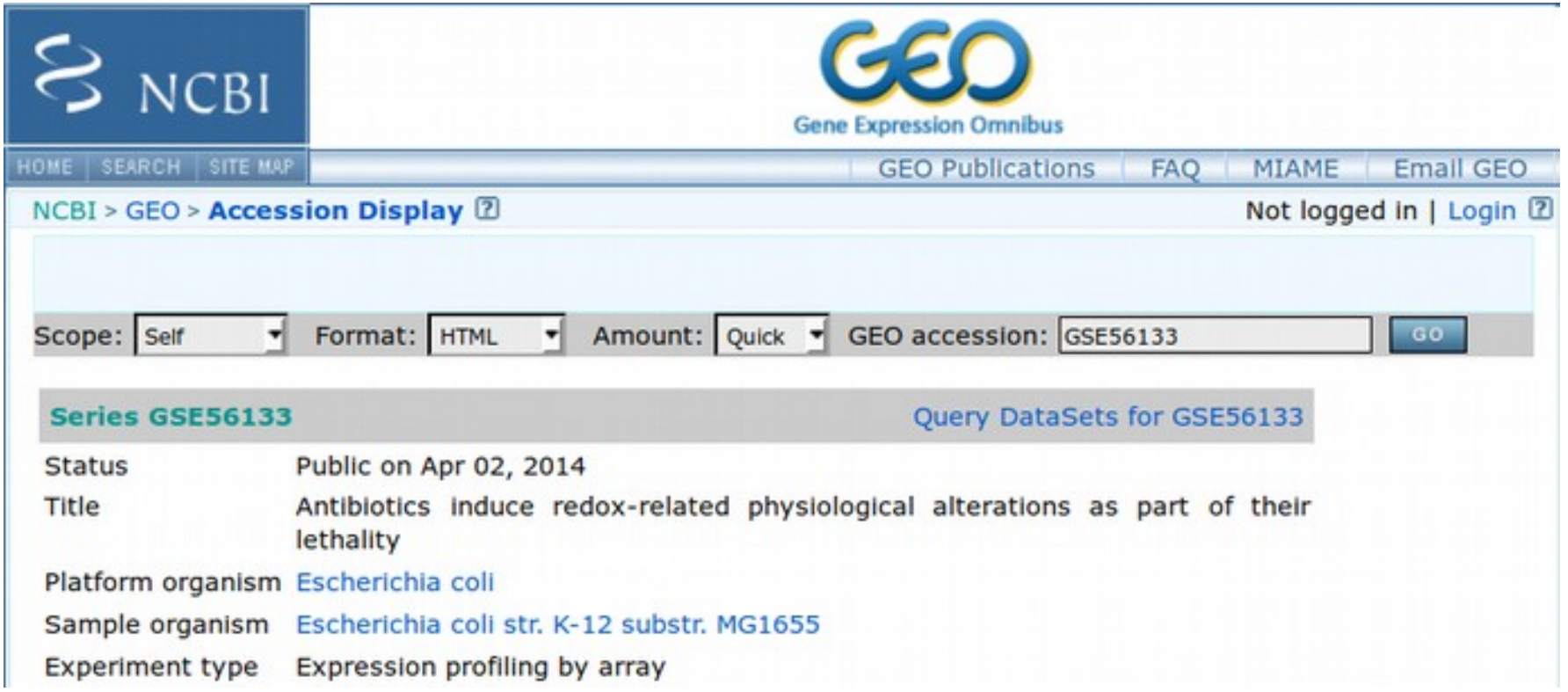
Mikrodizin analizi


NCBI
GEO
Gene Expression Omnibus
HOME
SEARCH
SITE MAP
GEO Publications
FAQ
MIAME
Email GEO
NCBI > GEO > Accession Display
Not logged in | Login
Scope: Self
Format: HTML
Amount: Quick
GEO accession: GSE56133
GO
Series GSE56133
Query DataSets for GSE56133
Status
Public on Apr 02, 2014
Title
Antibiotics induce redox-related physiological alterations as part of their lethality
Platform organism
Escherichia coli
Sample organism
Escherichia coli str. K-12 substr. MG1655
Experiment type
Expression profiling by array
Overall design
WT or mutant E coli cells were grown to OD ~0.3. Untreated cells were harvested at time 0 as controls. Treated cells given the appropriate chemical perturbation and harvested 1 hour post-treatment. All experiments were performed in technical triplicate.

```
%matplotlib inline

import pandas as pd
import numpy as np
from numpy import array
from scipy.stats.stats import ttest_ind
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from numpy import array
import scipy.cluster.hierarchy as sch
import scipy.spatial.distance as dist
```

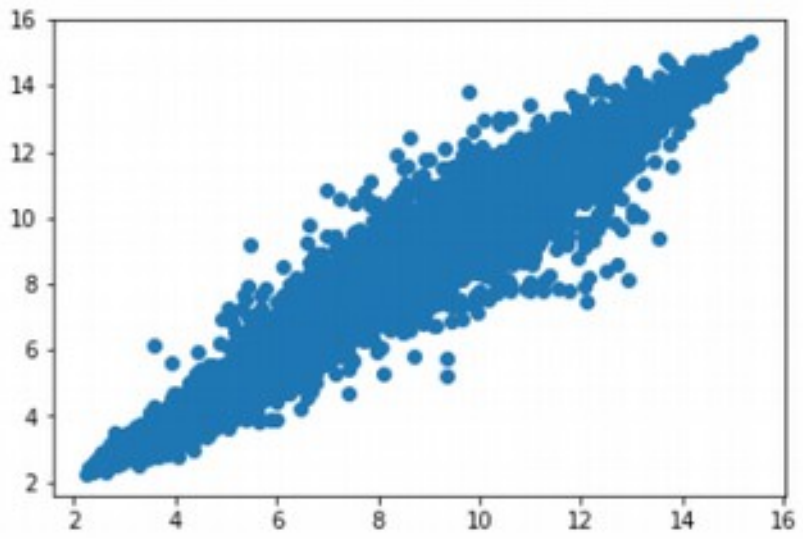
```
class GeoExpressionMatrix(object):
    def __init__(self, filename):
        f = open(filename, 'r')
        l = f.readline()
        self.sampleTitles = []
        self.sampleGeoAccessions = []
        while('!series_matrix_table_begin' not in l):
            l = f.readline()
            if '!Sample_title' in l:
                self.sampleTitles = l.split('\t')[1:]
                for i in range(0, len(self.sampleTitles)):
                    self.sampleTitles[i] = self.sampleTitles[i].replace('"', '').strip()
                #print(self.sampleTitles)
            if '!Sample_geo_accession' in l:
                self.sampleGeoAccessions = l.split('\t')[1:]
                for i in range(0, len(self.sampleGeoAccessions)):
                    self.sampleGeoAccessions[i] = self.sampleGeoAccessions[i].replace('"', '').strip()
                #print(self.sampleGeoAccessions)
        # sample titles - geo accessions dictionary
        self.titles2acc = dict(zip(self.sampleTitles, self.sampleGeoAccessions))
        print(self.titles2acc)
        #compat.PY3 = True
        self.df = pd.read_csv(f, delimiter='\t', comment='!')
```

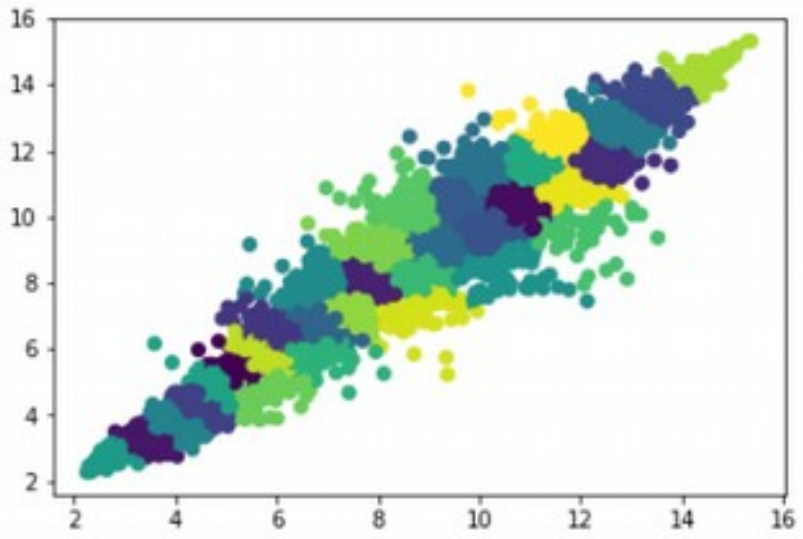
```python
def ttest(self, group1, group2):
    idref = []
    m1 = []
    m2 = []
    std1 = []
    std2 = []
    ts = []
    ps = []
    for index, row in self.df.iterrows():
        g1 = []
        for i1 in group1:
            g1.append(row[self.titles2acc[i1]])
        g2 = []
        for i2 in group2:
            g2.append(row[self.titles2acc[i2]])
        m1.append(np.mean(g1))
        m2.append(np.mean(g2))
        std1.append(np.std(g1))
        std2.append(np.std(g2))
        t, p = ttest_ind(a=g1, b=g2, equal_var=False)
        ts.append(t)
        ps.append(p)
        idref.append(row['ID_REF'])
        #print(np.mean(g1), np.std(g1), np.mean(g2), np.std(g2))
    columns=['ID_REF', 'mean1', 'std1', 'mean2', 'std2', 't', 'p']
    data = {'ID_REF':idref,
            'mean1':m1,
            'std1':std1,
            'mean2':m2,
            'std2':std2,
            't':ts,
            'p':ps}
    print(len(idref))
    dfTtest = pd.DataFrame(data=data, columns=columns)
    dfTtest = self.bonferroni(dfTtest)
    dfTtest.to_csv('ttest.csv', sep='\t')
```

Çok fazla miktarda hipotez testi → muhtemelen bir kısmını yanlışlıkla Önemli kabul ettik...

```
def bonferroni(self, df):
    print('bonferroni correction:')
    geneCount = len(df) + 1
    bonferroniCorr = []
    for index, row in df.iterrows():
        corrP = row['p'] * geneCount
        if corrP > 1:
            corrP = 1
        bonferroniCorr.append(corrP)
    df['bonfCorr'] = array(bonferroniCorr)
    return df
```

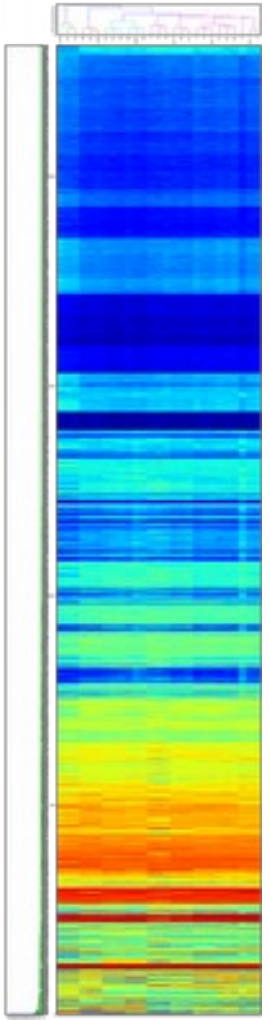
```
def cluster(self, group1, group2):
    m1 = []
    m2 = []
    idref = []
    for index, row in self.df.iterrows():
        g1 = []
        for i1 in group1:
            g1.append(row[self.titles2acc[i1]])
        g2 = []
        for i2 in group2:
            g2.append(row[self.titles2acc[i2]])
        m1.append(np.mean(g1))
        m2.append(np.mean(g2))
        idref.append(row['ID_REF'])
    plt.scatter(m1, m2)
    plt.show()
    mm = []
    for mm1, mm2 in zip(m1, m2):
        mm.append([mm1, mm2])
    #print(mm)
    x = np.matrix(mm)
    #print(x)
    kmeans = KMeans(n_clusters=32).fit(x)
    print(kmeans.labels_)

    plt.scatter(x=m1, y=m2, c=kmeans.labels_)
    plt.show()
```

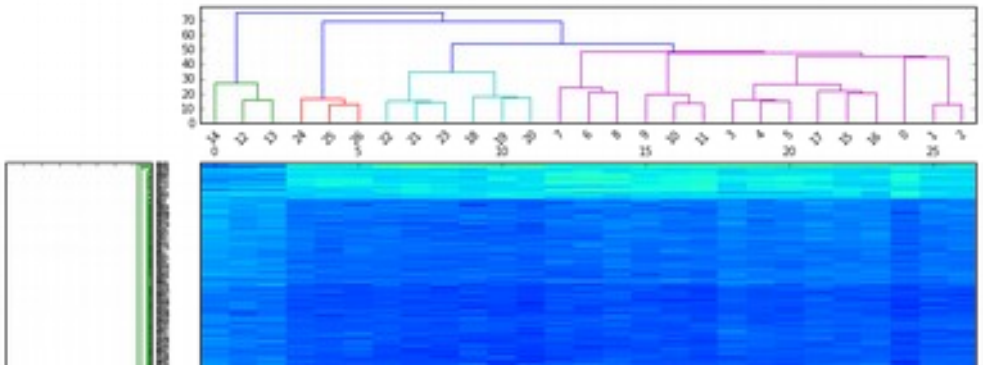




```python
def heatmap(self, df):
    dfNoRowNames = df.drop('ID_REF', axis=1)
    x = dfNoRowNames.values
    #print(x)
    m = len(x) # observations -> örnekler
    n = len(x[0]) # genes

    plotW = 12
    plotH = 48
    fig = plt.figure(figsize=(plotW, plotH))

    # top dendrogram
    distTop = dist.pdist(x.T) # distance matrix
    distSTop = dist.squareform(distTop)
    np.savetxt('distTop.csv', distSTop, delimiter='\t')
    linkageTop = sch.linkage(distSTop)
    dendTopAxis = fig.add_axes([0.2, 0.96, 0.8, 0.03]) #  [left, bottom, width, height]
    dendTop = sch.dendrogram(linkageTop)

    # left dendrogram
    distLeft = dist.pdist(x)
    distSLeft = dist.squareform(distLeft)
    linkageLeft = sch.linkage(distSLeft)
    dendLeftAxis = fig.add_axes([0, 0, 0.15, 0.95])  #  [left, bottom, width, height]
    dendLeft = sch.dendrogram(linkageLeft, orientation='left')

    # heat map
    heatmapAxis = fig.add_axes([0.2,0,0.8,0.95])  #  [left, bottom, width, height]
    xt = x
    dendLeavesTop = dendTop['leaves']
    dendLeavesLeft = dendLeft['leaves']
    xt = xt[:,dendLeavesTop]
    xt = xt[dendLeavesLeft,:]
    heatmap = heatmapAxis.matshow(xt, aspect='auto', origin='lower')

    plt.show()
```

Büyük bir ısı haritası...





Örneklerin birbirine uzaklığını
Gösteren üst dendrogram

Genlerin birbirine uzaklığını
Gösteren sol dendrogram