Database Homework

- Enter Sudent Name: Fernando Ramirez: Kevin McShane
- Enter Date: 21Oct2020

In [1]:
```
# When you load this file in Jupyter, you need to use Cell->RunAll to m
ake the instructor-provided code to take effect.

%load_ext autoreload
%autoreload 2

# import ahmet's bmes module that contains useful functions for downloa
ding files from web.
import sys, pathlib
#sys.path.append('D:/ahmet/doc/Dropbox/share/bmes.ahmet') #this is only
for ahmet's computer.
sys.path.append('C:/Users/Fernando A. Ramirez/Dropbox/bmes.ahmet')
#if bmes.ahmet is not in your PYTHONPATH and none of the following can
locate where you have your bmes.ahmet folder,
#  you will need to hard-code it.
sys.path.append('../bmes.ahmet'); sys.path.append('../../bmes.ahmet');
sys.path.append(str(pathlib.Path.home())+'/Dropbox/bmes.ahmet');

import bmes
import sqlite3
```

Comments

The above path was hardcorded to the file location of my dropbox. This was performed becuase I had some issues running the following commands in the command-line (as administrator).

setx /M BMESAHMETDIR C:/Users/Fernando A. Ramirez/Dropbox/bmes.ahmet

setx /M BMESAHMETDIR C:/Users/nando/Dropbox/bmes.ahmet

setx does create an environment variable under the Users 'nando' however, I was able to succesfully make the connection on the executable routing to the folder name under 'nando'. I believe the issue has to do with spaces on 'Fernando A. Ramirez'

Thus, this is why I choose to hard-code the sys.path directory. This is not ideal because having it route to bmes.ahmet will update and synchronize the code below (if changes were made to the database).

Comments

The section above is incomplete. I think I have it mapped to the dropbox, because that is my directory location. However, not sure what is going on with the hard-code.. e.g (sys.path.append('../bmes.ahmet).....

```
In [2]: print(bmes.tempdir())
        #shows the temporary directory where files are being stored
```

```
C:/Users/FERNAN~1.RAM/AppData/Local/Temp/bmes
```

===== Yeast apoptosis genes (20pt)

Write a GO query to find the names of yeast genes that are associated with "execution phase of apoptosis". Here, we define "yeast" as any organism under the genus Saccharomyces.

- Fetch the results of your GO query from the web and display them as output from your python/Matlab code.

In [3]:
```
#database to be provided ---
#find any yeast genes, that are associated with apoptosis
import gzip
import sqlite3
from pandas import DataFrame

url = 'http://sacan.biomed.drexel.edu/ftp/binf/godb.sqlite'
godbfile = bmes.downloadurl(url)
db = sqlite3.connect(godbfile)
cur = db.cursor();

def myselect(sql):
    cur.execute(sql);
    rows=cur.fetchall();
    if len(rows)==0:
        print('No results returned for SQL query.');
    else:
        df = DataFrame(rows)
        df.columns = [x[0] for x in cur.description]
        display(df)

#first pulling the ALL type of genus where column = Saccharomyces from
the table species
#yeast is as

#name of the yeast are under the gene_product table as symbols

#species id under the gene_product

#* FROM term WHERE name LIKE "%execution phase of apoptosis%"

cur.execute(''' SELECT DISTINCT(gene_product.symbol)
          FROM term AS t1, graph_path, term AS t2, association, gene_pro
duct, species
          WHERE t1.name LIKE "%apoptosis%"
          AND t1.id = graph_path.term1_id
          AND graph_path.term2_id = t2.id
          AND t2.id = association.term_id
          AND association.gene_product_id = gene_product.id
          AND species.genus="Saccharomyces"
          AND  species.id = gene_product.species_id ''')

#intact table connections from lecture, however wild-card to associate
the term name
#with any genus containing apoptosis and genus Saccharomyces, containin
g the associated
#yeast genes.

rs=cur.fetchall();
print('Number of genes found: [%d].' %(len(rs)))
print(', '.join([x[0] for x in rs[0:10]]))
```

```
Number of genes found: [2].
NUC1, YBL055C
```

===== mirdb (80pt)

Download file & Set up db connection

This section is sufficient for downloading the data file and setting up the database connection. You may make changes/improvements or keep it as is.

In the remainder of this problem, you need to use the mirtxtfile and db variables created here.

In [4]:
```
from pprint import pprint

mirurl='http://sacan.biomed.drexel.edu/lib/exe/fetch.php?rev=&media=course:bcomp2:db:homework_mirdb_dog75.txt';
mirtxtfile=bmes.downloadurl(mirurl,'mirdb_dog75.txt')

mirdbfile=bmes.datadir()+'/mirdb_dog.sqlite'
db = sqlite3.connect(mirdbfile)

print(db)

## code provided by prof. sacan
```

```
<sqlite3.Connection object at 0x000001D805803990>
```

%% (30pt) Create a database from mirdb data.

- Any downloaded files should be stored elsewhere on your computer (i.e., in a "Temporary" directory).
- Store the database elsewhere (in "Temporary" directory) on your computer; not within the same folder as your assignment.

In [5]:
```
#The code in the section above already accomplishes the file download &
location requirements.
#Just make use of mirtxtfile and db variables here.

import gzip
import sqlite3
#modules we need to import
gzfile = bmes.downloadurl('http://mirdb.org/download/miRDB_v6.0_predict
ion_result.txt.gz')
i = 0;
#http://mirdb.org/download/miRDB_v6.0_prediction_result.txt.gz

conn = sqlite3.connect('mirdb.sqlite')
#cur = conn.cursor();
#not sure if we need an additional cursor here
conn.execute("""CREATE TABLE IF NOT EXISTS mirdb (
id INTEGER PRIMARY KEY,
miRNA VARCHAR(30),
gene VARCHAR(30),
score FLOAT); """);

#structure of database is (miRNA, target which genes, and the type of s
core) for the information type

conn.commit();

with gzip.open(gzfile,'rt') as file:
    for line in file:
        items = line.strip().split("\t")
        conn.execute("INSERT INTO mirdb (miRNA, gene, score) VALUES
('"+items[0]+"', \
        '"+items[1]+"', '"+items[2]+"');");

conn.commit();
cur = conn.cursor();
conn.close();

#example from in-class
# print('INSERT INTO myfruits(name,weight,color) VALUES (' + items[0] +
', ' + items[1] + ', ' + items[2] + ') ')
```

In [ ]:
```
#% If your database creation code does not work, you may use a database
#% created by the instructor. Uncomment the following lines to use the
#% instructor's database. If you are using the instructor's database, w
e
#% will assume that your database creation code does not work.

#mirdbfile='http://sacan.biomed.drexel.edu/lib/exe/fetch.php?rev=&media
=course:bcomp2:db:homework_mirdb_dog.sqlite';
#mirdbfile=bmes.downloadurl(mirdbfile);
#db = sqlite3.connect(mirdbfile)


#then possible to use the selection queries to answer these values.
```

%% Find miRNAs for a target

Comments

ideally the way I would check the execution queries would be using bash, including nano and pipe commands to sort the the target and counts to confirm the correct output was achieved.

In [6]:
```
#% * (20pt) How many miRNAs are predicted to target XM_532324  ?
# print only the value of how many miRNA there are

cur.execute("SELECT count(*) FROM mirdb WHERE gene = 'XM_532324';");
rows = cur.fetchall()[0][0] # if remove [0][0] output is (,245)
print(rows)

# rows = cur.fetchall()
# #cur.execute("SELECT * FROM students;");
# # rows = cur.fetchall()
```

```
294
```

In [7]:
```
#%  * Show at most 10 miRNAs that are predicted to target XM_532324.
from pandas import DataFrame
cur.execute("SELECT * FROM mirdb WHERE gene = 'XM_532324';");
rows=cur.fetchall();
df = DataFrame(rows)

df.columns = [x[0] for x in cur.description] #iterate over columns for
the headers using
#list comprehension
display(df.head(10))
#display only the begining top beginning rows
```

	id	miRNA	gene	score
0	57	cfa-miR-1185	XM_532324	76.632935
1	3103	cfa-miR-544	XM_532324	63.267940
2	20806	cfa-miR-342	XM_532324	84.569807
3	40353	cfa-let-7b	XM_532324	69.058900
4	73181	cfa-miR-345	XM_532324	54.234400
5	79009	cfa-miR-1836	XM_532324	59.643486
6	83638	cfa-miR-8881	XM_532324	81.215490
7	94396	cfa-miR-144	XM_532324	68.878200
8	99485	cfa-miR-98	XM_532324	56.143000
9	109804	cfa-miR-8797	XM_532324	62.892762

%% Find targets for a miRNA

In [8]:
```
#% * (20pt) How many predicted targets of cfa-let-7a have a prediction
score of at least 80?
# find at least how many of the targets gene > 80.
cur.execute("SELECT count(*) FROM mirdb WHERE miRNA = 'cfa-let-7a' AND
score >= 80").fetchall()[0][0]
```

Out[8]:
```
1818
```

In [9]:
```
#% * Show at most 10 predicted targets of cfa-let-7a that have a predic
tion score of at least 80.
from pandas import DataFrame
cur.execute("SELECT * FROM mirdb WHERE miRNA = 'cfa-let-7a' AND score >
= 80")
rows = cur.fetchall();
df = DataFrame(rows)

df.columns = [x[0] for x in cur.description]
display(df.head(10))
```

	id	miRNA	gene	score
0	351845	cfa-let-7a	XM_014119515	92.568227
1	351846	cfa-let-7a	XM_847837	97.108376
2	351847	cfa-let-7a	XM_014111346	89.706800
3	351852	cfa-let-7a	XM_541808	98.033650
4	351854	cfa-let-7a	XM_005621935	89.702500
5	351857	cfa-let-7a	XM_014118125	92.237420
6	351859	cfa-let-7a	XM_847579	91.793000
7	351863	cfa-let-7a	XM_005630512	83.195500
8	351864	cfa-let-7a	XM_005618982	89.675100
9	351866	cfa-let-7a	XM_014114613	83.496800

%% Summarize miRNAs and target counts

In [10]:
```
#% * (10pt) List the miRNAs and the number of their targets.
#(Each row of the result should contain a distinct miRNA).
#(Use count() and GROUP BY). Show only top 10 rows of the result.

cur.execute("SELECT miRNA, count(*) FROM mirdb GROUP BY miRNA");

rows = cur.fetchall() #cursor needed to fetch all results
i = 0
for row in rows:
    print(row)
    i = i + 1
    if i >= 10: break

# i=i+1;
# if i>=10: break

#find-out how many targets are found in total, but only show the start
of the frist ten.
#for the selection statement.
```

```
('cfa-let-7a', 8292)
('cfa-let-7b', 8304)
('cfa-let-7c', 8292)
('cfa-let-7d', 582)
('cfa-let-7e', 8292)
('cfa-let-7f', 8238)
('cfa-let-7g', 8334)
('cfa-let-7j', 8244)
('cfa-miR-1', 9168)
('cfa-miR-101', 13320)
```

In [11]:
```
#Same way as above, just wanted to try out the for loop as well.
cur.execute("SELECT miRNA, count(*) FROM mirdb GROUP BY miRNA");

rows=cur.fetchall();

df = DataFrame(rows)
df.columns = [x[0] for x in cur.description] #iterate over columns for
the headers using list
#comprehension
display(df.head(10))
```

	miRNA	count(*)
0	cfa-let-7a	8292
1	cfa-let-7b	8304
2	cfa-let-7c	8292
3	cfa-let-7d	582
4	cfa-let-7e	8292
5	cfa-let-7f	8238
6	cfa-let-7g	8334
7	cfa-let-7j	8244
8	cfa-miR-1	9168
9	cfa-miR-101	13320

In [ ]: