COMP 364: Computer Tools for Life Sciences
Python libraries; How to read and use an API

Christopher J.F. Cameron and Carlos G. Oliver
Problem solving

Today’s lecture will be slightly different than most
▶ we’re going to define a problem
▶ then try to solve it using an unknown toolset
▶ this will help you to learn Python on your own

We’re going to learn
▶ how to use Google to search for Python modules
▶ reading module documentation/API
   ▶ application programming interface (API)
Genes and their role in a cell

Remembering the **central dogma:**

1. genes are made up of DNA
   - DNA ∈ \{A, C, G, T\}
2. genes are transcribed into RNA
   - RNA ∈ \{A, C, G, U\}
3. RNA is then translated into protein(s)

Proteins play a vital role in our survival

- the ‘building blocks’ of cells
- mutations in genes can lead to a malfunctioning protein
  - genes contain the instructions to build proteins
- many diseases have been linked to malfunctioning proteins
  - cystic fibrosis, Huntington’s disease, etc.
Problem

To better understand the role(s) some genes play in cells

- we will group them by a similarity measure
- in our case, using gene expression

Gene expression can be measured by the amount of RNA found within a cell

- where each RNA is related to a gene
- the more RNA attributed to a gene, the more it was expressed

Problem:
Given a dataset containing a set of genes and their expression over a time course, group genes based on their expression between two time points.
Gene expression dataset

The gene expression dataset can be downloaded from:
http://www.exploreddata.net/Downloads/
Gene-Expression-Data-Set

This dataset includes:
▶ rows - 4381 observed genes
▶ columns - across 25 time points (in mins)
▶ each floating point value represents a gene’s expression for a specific time point

Let’s start by reading the file into memory
▶ and storing it in a useful data structure
   ▶ what would be an appropriate data structure?
```python
data_dict = {}
with open("./Spellman.csv","r") as f:
    header = f.readline().rstrip().split("","\n")
    time_points = [int(val) for val in header[1:]]
for line in f:
    gene_name, *exp_counts = line.rstrip().split("","\n")
    exp_counts = [float(val) for val in exp_counts]
    try:
        data_dict[gene_name]
        print("Warning - multiple entries for the" "same gene '"+gene_name+"'")
    except:
        data_dict[gene_name] = exp_counts
print(len(data_dict.keys())) # prints: 4381
```

line 6 - '*' is extended iterable unpacking in Python 3
Okay, now let’s now select 10 genes randomly to analyze
  ▶ gene names are equivalent to dictionary keys

Steps:
  1. obtain a list of the dictionary’s keys
  2. randomly choose keys from the list

Wait, how can we figure out the Python implementation of the second step?
Okay, now let’s now select 10 genes randomly to analyze

- gene names are equivalent to dictionary keys

Steps:

1. obtain a list of the dictionary’s keys
2. randomly choose keys from the list

Wait, how can we figure out the Python implementation of the second step?

**Answer:** let’s try Google

http://lmgtfy.com/?q=how+to+randomly+select+keys+from+a+Python+dictionary
import random

rand_genes = random.sample(list(data_dict.keys()),k=10)
print(rand_genes)

# prints: ['YNR040W', 'YLR078C', 'YLL065W',
# 'YMR102C', 'YLR237W', 'YBR195C',
# 'YDR459C', 'YIL144W', 'YOR310C',
# 'YOR015W']

* source: https://docs.python.org/3/library/random.html
Choosing time points

Let’s start by randomly selecting a pair of time points

- the early time point will be start
- the later time point will be end
- how can we do this with Python’s random module?
Choosing time points #2

Let’s start by randomly selecting a pair of time points
  ▶ the early time point will be `start`
  ▶ the later time point will be `end`
  ▶ how can we do this with Python’s random module?

```python
import random

start_tp = random.choice(time_points)
end_tp = start_tp
while end_tp == start_tp:
    end_tp = random.choice(time_points)
print(start_tp, end_tp)  # prints: 240 220
# ensure proper ordering of time points
start_tp, end_tp = sorted([start_tp, end_tp])
print(start_tp, end_tp)  # prints: 220 240
```
Extracting expression data

Now, let’s extract the gene expression data for our genes

- at the randomly chosen time points

In other words,

- For each gene that was randomly selected
- find the expression value for said gene
- at the start and end time points
- and store the expression values in a useful data structure
  - perhaps a list of tuples?
  - or can someone think of a better implementation?
obs = []

# obtain list indices of time points
start_index = time_points.index(start_tp)
end_index = time_points.index(end_tp)

# iterate over genes and extract expression data
for gene_name in rand_genes:
    pair = []
    pair.append(data_dict[gene_name][start_index])
    pair.append(data_dict[gene_name][end_index])
    obs.append(tuple(pair))

print(obs)

# prints:
#   [(-0.48, 0.49), (0.0, -0.05), (0.06, -0.24),
#    (0.41, -0.4), (0.09, 0.43), (0.01, 0.36),
#    (-0.06, 0.29), (-0.24, 0.53), (0.19, -0.24),
#    (0.52, -0.32)]
Putting it together

Okay, now that we have a list that contains

- expression data for
- 10 randomly selected genes at
- two randomly chosen time points

How can we group these genes together based on their expression?
Okay, now that we have a list that contains

- expression data for
- 10 randomly selected genes at
- two randomly chosen time points

How can we group these genes together based on their expression?

**Answer:** Google

http://lmgtfy.com/?q=how+to+group+genes+expression
Clustering (or sometimes called ‘cluster analysis’)

- is the task of grouping a set of objects
- in such a way that objects in the same group (cluster)
- are more similar to each other than to those in other groups

How can we possibly learn to cluster gene expression data in Python?

**Answer:** Google! (hmmm.... a trend is forming here)

http://lmgtfy.com/?q=python+clustering+genes+expression
SciPy clustering

SciPy pronounced (‘Sigh Pie’) is a popular Python module

▶ provides many user-friendly and efficient functions
▶ useful for mathematics, science and engineering

API may be accessed from:
https://docs.scipy.org/doc/scipy/reference/

Let's navigate the API documentation

▶ to find possible clustering algorithms
▶ and implement one clustering algorithm in our Python script
from scipy.cluster.vq import kmeans

k = 3
code_book, distortion = kmeans(obs,3)
print(code_book,distortion)

# prints:
# [[ 0.55333333  0.16333333]
# [-0.77    -0.19   ]
# [ 0.03166667  0.095   ]] 0.157753028754

Well, that’s not entirely helpful
  ▶ kmeans() returns a list of centroid coordinates
    ▶ a centroid is the centre of a cluster
  ▶ and some measure called ‘distortion’
What's this `kmeans2()`?

```python
from scipy.cluster.vq import kmeans2
k = 3
centroid, label = kmeans2(obs, 3)
print(centroid, label)
# prints:
# 
# 
# 
# [ [-0.23 0.388 ]
#  [ 0.105 -0.485 ]
#  [ 0.05 -0.12666667]] [0 2 0 2 1 0 2 0 0 1]
```

That's better

- now we have centroid coordinates
- and a list of group/cluster labels
Next week - Matplotlib