Project Design

Genome 559: Introduction to Statistical and Computational Genomics

Elhanan Borenstein

Hypothesis:

The average degree in the metabolic networks of Prokaryotes is higher than the average degree in the metabolic networks of Eukaryotes

KEGG Home
Introduction
Overview
Release notes
Current statistics

KEGG Identifiers
Pathway maps
Brite hierarchies

KEGG XML

KEGG API

KEGG FTP

KegTools

GenomeNet

DBGET/LinkDB

Feedback

KEGG FTP

KEGG FTP Site for Academic Users

The KEGG data may be downloaded by academic users from the KEGG FTP site:

ftp://ftp.genome.jp/pub/kegg/

Non-academic users are required to obtain a license agreement for downloading KEGG.

- Terms of use
- Licensing from Pathway Solutions

Announcement:
A new directory, "module", is created.

Posted on December 22, 2010 » Past announcements

Directories and Files

pathway/	KEGG PATHWAY (daily updated)
map/	Reference pathway maps
ko/	Reference pathway maps (KO)
ec/	Reference pathway maps (EC)
rn/	Reference pathway maps (reaction)
organisms/	Organism-specific pathway maps
pathway	Pathway entries (text data)
map_title.tab	List of pathways available
module/	KEGG MODULE (daily updated) *New!*
ko/	Reference module maps (KO) - to be added
organisms/	Organism-specific module maps - to be added
module	Module entries (text data)

ko.txt

```
ENTRY       K00001                      KO
NAME        E1.1.1.1, adh
DEFINITION  alcohol dehydrogenase [EC:1.1.1.1]
PATHWAY     ko00010  Glycolysis / Gluconeogenesis
            ko00071  Fatty acid metabolism
MODULE      M00236  Retinol biosynthesis, beta-cacrotene => retinol
CLASS       Metabolism; Carbohydrate Metabolism; Glycolysis / Gluconeogenesis
            [PATH:ko00010]
            Metabolism; Lipid Metabolism; Fatty acid metabolism [PATH:ko00071]
            Metabolism; Amino Acid Metabolism; Tyrosine metabolism
            [PATH:ko00350]
            Metabolism; Metabolism of Cofactors and Vitamins; Retinol metabolism
DBLINKS     RN: R00623 R00754 R02124 R04805 R04880 R05233 R05234 R06917 R06927
                R07105 R08281 R08306 R08310
            COG: COG1012 COG1062 COG1064 COG1454
            GO: 0004022 0004023 0004024 0004025
GENES       HSA: 124(ADH1A) 125(ADH1B) 126(ADH1C) 127(ADH4) 130(ADH6) 131(ADH7)
            PTR: 461394(ADH4) 461395(ADH6) 461396(ADH1B) 471257(ADH7)
                 744064(ADH1A) 744176(ADH1C)
            MCC: 707367 707682(ADH1A) 708520 711061(ADH1C)
...
            PAS: Pars_0396 Pars_0534 Pars_0547 Pars_1545 Pars_2114
            TPE: Tpen_1006 Tpen_1516
///
ENTRY       K00002                      KO
NAME        E1.1.1.2, adh
DEFINITION  alcohol dehydrogenase (NADP+) [EC:1.1.1.2]
PATHWAY     ko00010  Glycolysis / Gluconeogenesis
            ko00561  Glycerolipid metabolism

...
```

reaction.txt

```
R00005: 00330: C01010 => C00011
R00005: 00791: C01010 => C00011
R00005: 01100: C01010 <=> C00011
R00006: 00770: C00022 => C00900
R00008: 00362: C06033 => C00022
R00008: 00660: C00022 => C06033
R00010: 00500: C01083 => C00031
R00013: 00630: C00048 => C01146
R00013: 01100: C00048 <=> C01146
R00014: 00010: C00022 + C00068 => C05125
R00014: 00020: C00068 + C00022 => C05125
R00014: 00290: C00022 => C05125
R00014: 00620: C00068 + C00022 => C05125
R00014: 00650: C00068 + C00022 => C05125
R00014: 01100: C00022 <=> C05125
R00018: 00960: C00134 => C06366
R00019: 00630: C00080 => C00282
R00019: 00680: C00080 => C00282
R00021: 00910: C00025 <= C00064
R00022: 00520: C01674 => C00140
...
```

genome.txt

```
ENTRY        T00001            Complete  Genome
NAME         hin, H.influenzae, HAEIN, 71421
DEFINITION   Haemophilus influenzae Rd KW20 (serotype d)
ANNOTATION   manual
TAXONOMY     TAX:71421
  LINEAGE    Bacteria; Proteobacteria; Gammaproteobacteria; Pasteurellales;
             Pasteurellaceae; Haemophilus
DATA_SOURCE  RefSeq
ORIGINAL_DB  JCVI-CMR
DISEASE      Meningitis, septicemia, otitis media, sinusitis and chronic
             bronchitis
CHROMOSOME   Circular
  SEQUENCE   RS:NC_000907
  LENGTH     1830138
STATISTICS   Number of nucleotides:       1830138
             Number of protein genes:        1657
             Number of RNA genes:              81
REFERENCE    PMID:7542800
  AUTHORS    Fleischmann RD, et al.
  TITLE      Whole-genome random sequencing and assembly of Haemophilus
             influenzae Rd.
  JOURNAL    Science 269:496-512 (1995)
///
ENTRY        T00002            Complete  Genome
NAME         mge, M.genitalium, MYCGE, 243273
DEFINITION   Mycoplasma genitalium G-37
ANNOTATION   manual
TAXONOMY     TAX:243273
  LINEAGE    Bacteria; Tenericutes; Mollicutes; Mycoplasmataceae; Mycoplasma
...
```

hin_ko.txt

```
ace:Acel_0001       ko:K02313
ace:Acel_0002       ko:K02338
ace:Acel_0003       ko:K03629
ace:Acel_0005       ko:K02470
ace:Acel_0006       ko:K02469
ace:Acel_0012       ko:K03767
ace:Acel_0018       ko:K01664
ace:Acel_0019       ko:K08884
ace:Acel_0020       ko:K05364
ace:Acel_0026       ko:K01552
ace:Acel_0029       ko:K00111
ace:Acel_0031       ko:K00627
ace:Acel_0032       ko:K00162
ace:Acel_0033       ko:K00161
ace:Acel_0035       ko:K00817
ace:Acel_0036       ko:K07448
ace:Acel_0039       ko:K04750
ace:Acel_0041       ko:K03281
ace:Acel_0048       ko:K08323
ace:Acel_0051       ko:K03734
ace:Acel_0052       ko:K03147
ace:Acel_0057       ko:K03088
ace:Acel_0059       ko:K01010
ace:Acel_0061       ko:K03711
ace:Acel_0062       ko:K06980
ace:Acel_0063       ko:K07560
ace:Acel_0072       ko:K12373
ace:Acel_0075       ko:K01834
ace:Acel_0076       ko:K09796

...
```

Designing with Pseudo-Code Comments

Top down approach

```
# Preprocessing
# =============
```

```
# Build networks and calc degree
# ==============================

# Print output
# ============
```

Add details

```
# Preprocessing
# =============

# Read and store mapping from KO to RN



# Read and store mapping from RN to edges



# Read and store species list and lineages
```

```
# Build networks and calc degree
# ==============================

# Loop over species

    # Read KO list of current species

    # Map KO to RN and RN to edges



    # Calculate degree

    # Store: species, degree, phyla


# Print output
# ============

# Calculated average degree per P and per E

# Print
```

Add notes to self

```
# Preprocessing
# =============

# Read and store mapping from KO to RN




# Read and store mapping from RN to edges




# Read and store species list and lineages
```

```
# Build networks and calc degree
# ==============================

# Loop over species


    # Read KO list of current species


    # Map KO to RN and RN to edges


    # -> Here I should have a full network
    # -> TBD: What data structure should I use?

    # Calculate degree


    # Store: species, degree, phyla
    # -> TBD: How do I store results?


# Print output
# ============

# Calculated average degree per P and per E


# Print
```

Add variables, loops, if-s, function calls

```
# Preprocessing
# =============


# Read and store mapping from KO to RN
KO_file = 'ko.txt'
KO_to_RN = {}



# Read and store mapping from RN to edges
RN_file = 'reaction.txt'
RN_to_EDGES = {}



# Read and store species list and lineages
Genomes_file = 'genome.txt'
species_list = []
species_lineage = {}
```

```
# Build networks and calc degree
# ==============================

# Loop over species
for species in species_list:

    # Read KO list of current species


    # Map KO to RN and RN to edges


    # -> Here I should have a full network
    # -> TBD: What data structure should I use?

    # Calculate degree
    degree = CalcDegree(network)

    # Store: species, degree, phyla
    # -> TBD: How do I store results?


# Print output
# ============

# Calculated average degree per P and per E


# Print
```

Start coding small chunks

```
# Preprocessing
# =============


# Read and store mapping from KO to RN
KO_file = 'ko.txt'
KO_to_RN = {}



# Read and store mapping from RN to edges
RN_file = 'reaction.txt'
RN_to_EDGES = {}



# Read and store species list and lineages
Genomes_file = 'genome.txt'
species_list = []
species_lineage = {}
< LET'S WRITE THIS PART >
```

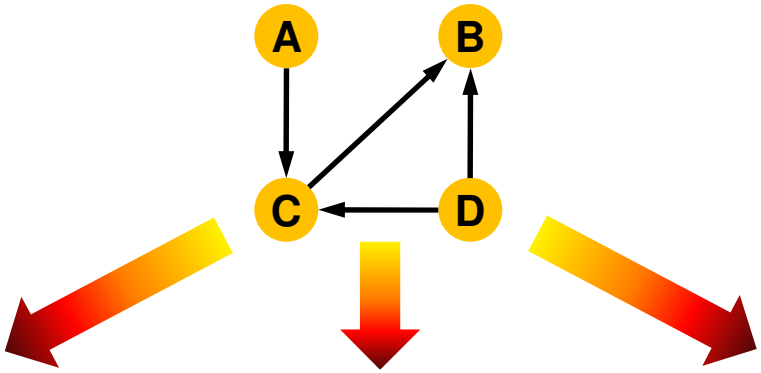
```
# Build networks and calc degree
# ==============================

# Loop over species
for species in species_list:

    # Read KO list of current species


    # Map KO to RN and RN to edges


    # -> Here I should have a full network
    # -> TBD: What data structure should I use?

    # Calculate degree
    degree = CalcDegree(network)

    # Store: species, degree, phyla
    # -> TBD: How do I store results?


# Print output
# ============

# Calculated average degree per P and per E


# Print
```

Define interfaces

```
# Preprocessing
# =============


# Read and store mapping from KO to RN
KO_file = 'ko.txt'
KO_to_RN = {}



# Read and store mapping from RN to edges
RN_file = 'reaction.txt'
RN_to_EDGES = {}



# Read and store species list and lineages
Genomes_file = 'genome.txt'
species_list = []
species_lineage = {}
< LET'S WRITE THIS PART >
```

```
# Build networks and calc degree
# ==============================

# Loop over species
for species in species_list:

    # Read KO list of current species


    # Map KO to RN and RN to edges


    # -> Here I should have a full network
    # -> TBD: What data structure should I use?

    # Calculate degree
    degree = CalcDegree(network)

    # Store: species, degree, phyla
    # -> TBD: How do I store results?


# Print output
# ============

# Calculated average degree per P and per E


# Print
```

Computational Representation of Networks



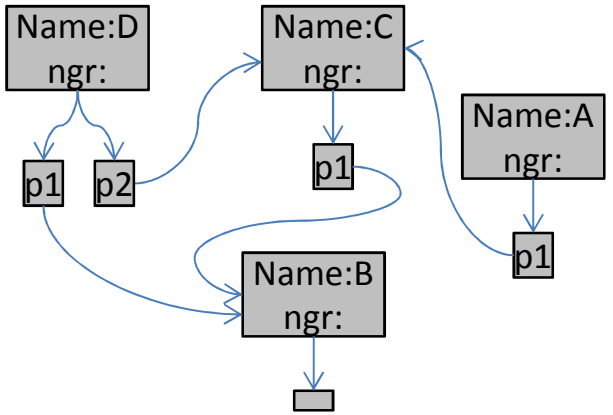
List of edges: (ordered) pairs of nodes

[(A,C) , (C,B) , (D,B) , (D,C)]

Connectivity Matrix

	A	B	C	D
A	0	0	1	0
B	0	0	0	0
C	0	1	0	0
D	0	1	1	0

Object Oriented

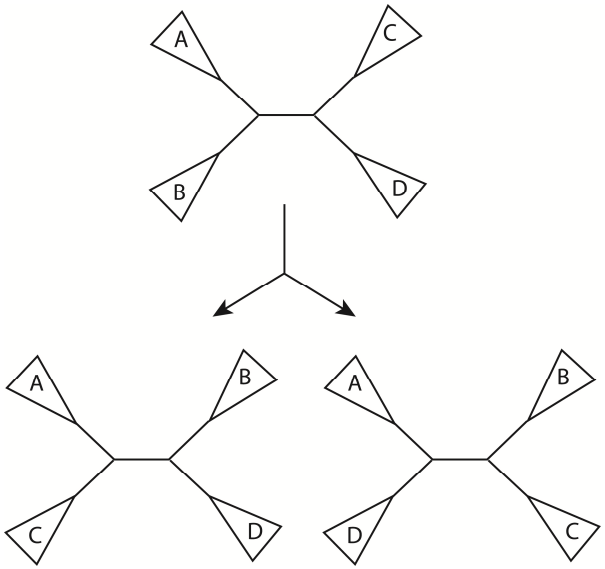


Final Exam

- **Two parts:**
  - *The first will focus on the bioinformatics topics covered in class.*
  - *The second on programming.*
- Both parts will comprise very simple and brief questions to account for the short time allowed for the exam.
- Open books (basically, any static resource you want is ok).

Common Mistakes: Parsimony

- Figure out how many possible Nearest-Neighbor Interchanges there are on a specific unrooted tree with 8 leaves (that is, the number of competing trees that would be considered in one step of the hill-climbing method using NNIs). Hint: a subtree can be any part of the tree, including a single leaf. Justify your answer.



Common Mistakes: Programming

- Comments !!!
- continue, elif, if …

```
for items in list:
      if (…):
            do_something
      else:
            continue
```

```
if (a > 10):
      do_nothing
else:
      print …
```

- Lists vs. Dictionaries

… it’s a wrap …
Hope you enjoyed!