Psychrophilic amino acid analysis

Building a BLAST protein database

Combine sequences into one file: `cat *.faa > proteins_combined.faa`.

Create the database: `makeblastdb -in proteins_combined.faa -dbtype prot -parse_seqids -out proteins_database -title "Mesophilic species proteins"`

Output:

```
Building a new DB, current time: 11/21/2014 10:14:35
New DB name:   proteins_database
New DB title:  Mesophilic species proteins
Sequence type: Protein
Keep Linkouts: T
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 94501 sequences in 3.75161 seconds.
```

Running BLASTP

The following command was executed:

```
blastp -db ../mesophilic/blast_db/proteins_database -query \
RhodococcusJG3.faa -out RhodococcusJG3.faa.blast.tab  -outfmt \
"6 qseqid sallseqid evalue bitscore length stitle"  -num_threads 4 \
-parse_deflines -evalue 1e-15
```

The total number of protein sequences in Rhodococcus JG3 is 4998, 4326 had a match with an evalue <= 1e-15.

Parse BLAST results

Format BLAST output to separate target definition and species columns using `blast_parser.py`:

```
"""
format blastp output when the following -outfmt option was used:
-outfmt "6 qseqid sallseqid evalue bitscore length stitle"

the final output contains the following items per line:
```

```
Query ID, Target ID, Evalue, Target Accession, Target Definition, Species
E.g.:
2529298360 646584812   3e-40   ROP_27020   hypothetical protein    Rhodococcus opacus B4
"""

from sys import argv
import csv

with open(argv[1], 'r') as csvfile:
    blast_reader = csv.reader(csvfile, delimiter='\t', quotechar='|')
    filename = "{}_parsed.tab".format(argv[1])
    with open(filename, 'w') as csvout:
        csvwriter = csv.writer(csvout, delimiter='\t', quotechar='|')
        for line in blast_reader:
            query_GI = line[0]
            target_GI = line[1]
            evalue = line[2]
            extra = line[5].split('Rhodococcus')
            target_accession = extra[0].split()[0]
            target_definition = ' '.join(extra[0].rstrip('[ ').strip('|')
                                    .split()[1:])
            species = "Rhodococcus{}".format(extra[1].split(':')[0]
                             .split('contig')[0].split('plasmid')[0]).strip()
            csvwriter.writerow([query_GI, target_GI, evalue, target_accession,
                           target_definition, species])
```

Load data into a database

Create an SQlite3 database:

Run `sqlite3 results.db` in shell, within sqlite3 promopt enter the following

```
CREATE TABLE blast_results(
query_id VARCHAR(30) NOT NULL,
target_id VARCHAR(30) NOT NULL,
evalue real,
target_accession VARCHAR(255),
target_definition VARCHAR(255),
species VARCHAR(100));

CREATE TABLE sequences(
id VARCHAR(30) NOT NULL,
definition TEXT NOT NULL,
species TEXT NOT NULL,
peptide_seq TEXT,
coding_seq TEXT);
```

Import parsed BLAST results

Set the mode to CSV: `.mode csv`. Set separator to tab: `.separator \t`. Import the data file into a table: `.import ../rhodococcusJG3/RhodococcusJG3.faa.blast.tab_parsed.tab blast_results`.

Run amino acid analysis script

Run the following script in the `scripts` directory to generate amino acid data (`python aa_analysis.py > aa_analysis.tsv`):

```
from Bio import SeqIO
from sqlalchemy import create_engine
from Bio.SeqUtils.ProtParam import ProteinAnalysis

amino_acids = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N',
               'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']


def aa_count(sequence):
    """ includes the amino acid counts, arg/lys ratio and
        acidic residue counts
    """
    count = ProteinAnalysis(sequence).count_amino_acids()
    count_list = [count[aa] for aa in amino_acids]
    arg_to_lys = float(count['R'])/count['K'] if count['K'] else 'N/A'
    acid_res = count['D'] + count['E']
    all_counts = "{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}\t{7}\t{8}\t{9}\t\
{10}\t{11}\t{12}\t{13}\t{14}\t{15}\t{16}\t{17}\t{18}\t\
{19}\t{20}\t{21}".format(*(count_list+[arg_to_lys]+[acid_res]))
    return all_counts


def aa_percent(sequence):
    """ includes the aliphatic index followed by amino acid percent values
    """
    percent = ProteinAnalysis(sequence).get_amino_acids_percent()
    aliphatic_index = percent['A'] + 2.9 * percent['V'] + 3.9 * (percent['I']
        + percent['L'])
    percent_list = ["{:.5f}".format(percent[aa]) for aa in amino_acids]
    all_percents = "{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}\t{7}\t{8}\t{9}\t\
{10}\t{11}\t{12}\t{13}\t{14}\t{15}\t{16}\t{17}\t{18}\t\
{19}\t{20}".format(*([aliphatic_index] + percent_list))
    return all_percents


def aroma(sequence):
    aromaticity = ProteinAnalysis(sequence).aromaticity()
    return aromaticity


def flex(sequence):
    """ Returns a list of scores for each residue and not a single score for
    the whole protein.
    """
    flexibility = ProteinAnalysis(sequence).flexibility()
    return flexibility


def gravy(sequence):
    if 'X' or '*' in sequence:
        sequence = sequence.replace('X', '')
        sequence = sequence.replace('*', '')
        g = ProteinAnalysis(sequence).gravy()
    else:
        g = ProteinAnalysis(sequence).gravy()
    return g

# load all sequences for quick access
prot_seqs = SeqIO.index("../all_proteins.faa", "fasta")

# connect to the database
engine = create_engine('sqlite:///../results_db/results.db')
conn = engine.connect()

# get the list of Rhodococcus sp. JG3 genes that have BLAST hits
JG3_genes = conn.execute("""SELECT DISTINCT query_id FROM blast_results""")

for gene in JG3_genes:
    gene = str(gene[0]).strip()

    # get the top hit per species, ignore hypothetical proteins
    query = """SELECT query_id, target_id, MIN(evalue),
           target_accession, target_definition, species
           FROM blast_results WHERE query_id={}
           AND target_definition NOT LIKE "%hypothetical%"
```

```
            GROUP BY species""".format(gene)
    results = conn.execute(query)
    results_list = [i for i in results]
    if len(results_list) >= 3:  # get genes with 3 or more BLAST hits
        query_prot_seq = str(prot_seqs["{}".format(gene)].seq)
        print "{}\t".format(gene), "{}\t".format(gene),\
            aa_percent(query_prot_seq), '\t', aa_count(query_prot_seq), '\t',\
            aroma(query_prot_seq), '\t', gravy(query_prot_seq)
        for hit in results_list:
            target_id = hit[1]
            target_prot_seq = str(prot_seqs["{}".format(target_id)].seq)
            print "{}\t".format(gene), "{}\t".format(target_id),\
                aa_percent(target_prot_seq), '\t', aa_count(target_prot_seq),\
                '\t', aroma(target_prot_seq), '\t', gravy(target_prot_seq)
```

Load AA analysis into database

Create the table for aa_analysis data:

```
CREATE TABLE aa_analysis(
query_id VARCHAR(30) NOT NULL,
target_id VARCHAR(30) NOT NULL,
aliphatic_index REAL,
Ap REAL,
Cp REAL,
Dp REAL,
Ep REAL,
Fp REAL,
Gp REAL,
Hp REAL,
Ip REAL,
Kp REAL,
Lp REAL,
Mp REAL,
Np REAL,
Pp REAL,
Qp REAL,
Rp REAL,
Sp REAL,
Tp REAL,
Vp REAL,
Wp REAL,
Yp REAL,
Ac INTEGER,
Cc INTEGER,
Dc INTEGER,
Ec INTEGER,
Fc INTEGER,
Gc INTEGER,
Hc INTEGER,
Ic INTEGER,
Kc INTEGER,
Lc INTEGER,
Mc INTEGER,
Nc INTEGER,
Pc INTEGER,
Qc INTEGER,
Rc INTEGER,
Sc INTEGER,
Tc INTEGER,
Vc INTEGER,
Wc INTEGER,
Yc INTEGER,
RK_ratio TEXT,
acidic_res INTEGER,
aromaticity REAL,
gravy REAL);
```

Set the mode to CSV: `.mode csv`. Set separator to tab: `.separator \t`. Import the data file into a table: `.import ../scripts/aa_analysis.tsv aa_analysis`.

Of the 4326 proteins that had a match with an evalue <= 1e-15, 3829 had 3 or more significant hits.

Run one sample t-test comparison script

The following script was run (`python significance_analysis.py > sig_analysis.tsv`):

```
from Bio import SeqIO
from sqlalchemy import create_engine
import numpy as np
from scipy import stats

# load all sequences for quick access
prot_seqs = SeqIO.index("../all_proteins.faa", "fasta")


# connect to the database
```

```
engine = create_engine('sqlite:///../results_db/results.db')
conn = engine.connect()

# get the list of Rhodococcus sp. JG3 genes that have BLAST hits
JG3_genes = conn.execute("""SELECT DISTINCT query_id FROM aa_analysis""")


def aliphatic_comp(gene_list, hit_list):
    """ returns a letter for the one sample t-test of aliphatic index:
    Z - all hit aliphatic indeces are the same, therefore can't runt the test
    S - no significant difference
    A - psychrophilic query is significantly different and higher than targets
    B - psychrophilic query is significantly different and lower than targets
    """
    aliphatic_index_q = gene_list[2]
    aliphatic_index_t = [i[2] for i in hit_list]

    # check if all the indeces are the same, in that case the standard
    # deviation would be 0 and would result in division by 0.
    if len(set(aliphatic_index_t)) == 1:
        return 'Z'
    with np.errstate(divide='ignore'):
        p_value = stats.ttest_1samp(aliphatic_index_t, aliphatic_index_q)[1]
    if p_value <= 0.05:
        if aliphatic_index_q > np.mean(aliphatic_index_t):
            return 'A'
        else:
            return 'B'
    else:
        return 'S'


def aa_percent_comp(gene_list, hit_list, aa):
    """ returns a letter for the one sample t-test of amino acid percent:
    Z - all hit amino acid percents are the same, can't runt the test
    S - no significant difference
    A - psychrophilic query is significantly different and higher than targets
    B - psychrophilic query is significantly different and lower than targets
    """
    aa_dict = {"A": 3, "C": 4, "D": 5, "E": 6, "F": 7, "G": 8, "H": 9,
               "I": 10, "K": 11, "L": 12, "M": 13, "N": 14, "P": 15, "Q": 16,
               "R": 17, "S": 18, "T": 19, "V": 20, "W": 21, "Y": 22}
    aa_q = gene_list[3]
    aa_t = [i[3] for i in hit_list]
    if len(set(aa_t)) == 1:
        return 'Z'
    with np.errstate(divide='ignore'):
        p_value = stats.ttest_1samp(aa_t, aa_q)[1]
    if p_value <= 0.05:
        if aa_q > np.mean(aa_t):
            return 'A'
        else:
            return 'B'
    else:
        return 'S'


def arg_lys_ratio_comp(gene_list, hit_list):
    """ returns a letter for the one sample t-test of RK ratio:
    Z - all hit RK ratios are the same, can't runt the test
    S - no significant difference
    A - psychrophilic query is significantly different and higher than targets
    B - psychrophilic query is significantly different and lower than targets
    N - RK ratio for query couldn't be calculated due to K = 0 for query
    """
    rk_q = gene_list[43]
    if rk_q != 'N/A':  # check if RK ratio was not calculated due to K = 0
        rk_q = float(rk_q)
        # remove all the RK ratios where there were no lysines (i.e. N/A)
        rk_t = [float(i[43]) for i in hit_list if i[43] != 'N/A']
        # if rest of the numbers are the same (1), or there are less than 3
        # numbers (2) or all values are N/A and an empty list is returned (0)
        # then return Z
        if len(set(rk_t)) <= 2:
            return 'Z'
        with np.errstate(divide='ignore'):
            p_value = stats.ttest_1samp(rk_t, rk_q)[1]
        if p_value <= 0.05:
            if rk_q > np.mean(rk_t):
                return 'A'
            else:
                return 'B'
        else:
            return 'S'
    else:
        return 'N'


def acidic_res_comp(gene_list, hit_list):
    ac_q = int(gene_list[44])
    ac_t = [float(i[44]) for i in hit_list]
    if len(set(ac_t)) == 1:
        return 'Z'
    with np.errstate(divide='ignore'):
        p_value = stats.ttest_1samp(ac_t, ac_q)[1]
    if p_value <= 0.05:
        if ac_q > np.mean(ac_t):
            return 'A'
        else:
            return 'B'
    else:
        return 'S'


def aromaticity_comp(gene_list, hit_list):
    ar_q = float(gene_list[45])
    ar_t = [float(i[45]) for i in hit_list]
    if len(set(ar_t)) == 1:
        return 'Z'
    with np.errstate(divide='ignore'):
        p_value = stats.ttest_1samp(ar_t, ar_q)[1]
    if p_value <= 0.05:
        if ar_q > np.mean(ar_t):
            return 'A'
        else:
            return 'B'
    else:
        return 'S'


def gravy_comp(gene_list, hit_list):
    gr_q = float(gene_list[46])
    gr_t = [float(i[46]) for i in hit_list]
    if len(set(gr_t)) == 1:
        return 'Z'
    with np.errstate(divide='ignore'):
        p_value = stats.ttest_1samp(gr_t, gr_q)[1]
    if p_value <= 0.05:
        if gr_q > np.mean(gr_t):
            return 'A'
        else:
            return 'B'
    else:
        return 'S'

for gene in JG3_genes:
    gene = gene[0]
    gene_data = conn.execute("""SELECT * FROM aa_analysis WHERE query_id={0}
                                AND target_id={0}""".format(gene)).fetchall()
    hit_data = conn.execute("""SELECT * FROM aa_analysis WHERE query_id={0}
                                AND target_id!={0}""".format(gene)).fetchall()
    for result in gene_data:
        print gene,'\t', aliphatic_comp(result, hit_data),'\t',\
        arg_lys_ratio_comp(result, hit_data),'\t',\
        acidic_res_comp(result, hit_data),'\t',\
        aromaticity_comp(result, hit_data),'\t',\
        gravy_comp(result, hit_data)
```

A header was appended to the results file: `cat sig_analysis_header.tsv sig_analysis.tsv > sig_analysis_w_header.tsv`.

Chi square analysis of hot and cold adapted genes

Numbers of hot and cold adapted genes were obtained manually for the 5 categories:

`csvcut -t -c 2 sig_analysis_w_header.tsv|grep -c "A"` or use `awk` `awk {'print $2'} sig_analysis.tsv|grep -c "A"`.

The following script `python chi2.py` was used to obtain the p-value:

```
"""
Chi squared test for 2 categories with Yates correction as desctibed here:
http://archive.bio.ed.ac.uk/jdeacon/statistics/tress9.html#Chi-squared test
"""

from scipy.stats import chisqprob
from math import fabs

A = 1440
B = 1023

expected = (A + B)/2.0  # the null hypothesis is that the numbers are the same

A_stat = ((fabs(A-expected)-0.5)**2)/expected
B_stat = ((fabs(B-expected)-0.5)**2)/expected

chi = A_stat + B_stat

probability = chisqprob(chi, 1)

print A_stat, B_stat, probability
```

Analysis of disordered proteins

IUPred (http://iupred.enzim.hu/Help.php) was used to calculate the proportion of residues with a disorder score of => 0.5. The query proportion was compared to hit proportions in the same manner as other parameters. Number of genes with a significantly lower disorder proporation was 1282 and those with significantly higher 1033. According to (http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0012069) psychrophilic prokaryotes have lower protein disorder parameters than mesophiles.