1 Basic Statistics

January 31, 2018

1 Table of Contents

2 Introduction

In this notebook, I will go over the basic results from the RNA-seq in what is essentially a top-level view of the results. Nothing specific, mainly numbers, some histograms and that's it. First, I will load a number of useful libraries. Notable libraries to load are `genpy`, a module that contains useful graphing functions tailored specifically for this project and developed by us; `morgan` a module that specifies what a Morgan object and a McClintock object are, and `gvars`, which contains globally defined variables that we used in this project.

In [1]:
```
# important stuff:
import os
import pandas as pd
import numpy as np

import morgan as morgan
import genpy
import gvars
import pretty_table as pretty
import epistasis as epi

# Graphics
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rc

rc('text', usetex=True)
rc('text.latex', preamble=r'\usepackage{cmbright}')
```

```
rc('font', **{'family': 'sans-serif', 'sans-serif': ['Helvetica']})

# Magic function to make matplotlib inline;
%matplotlib inline

# This enables SVG graphics inline.
# There is a bug, so uncomment if it works.
%config InlineBackend.figure_formats = {'png', 'retina'}

# JB's favorite Seaborn settings for notebooks
rc = {'lines.linewidth': 2,
      'axes.labelsize': 18,
      'axes.titlesize': 18,
      'axes.facecolor': 'DFDFE5'}
sns.set_context('notebook', rc=rc)
sns.set_style("dark")

mpl.rcParams['xtick.labelsize'] = 16
mpl.rcParams['ytick.labelsize'] = 16
mpl.rcParams['legend.fontsize'] = 14
```

Next, I will specify my q-value cutoff. A typical value for RNA-seq datasets is q=0.1 for statistical significance. I will also initialize a `genvar.genvars` object, which contains all of the global variables used for this project.

```
In [2]: q = 0.1
        # this loads all the labels we need
        genvar = gvars.genvars()
```

2.1 Data initialization

Now, I will prepare to initialize a Morgan project. Morgan objects have a large number of attributes. I wrote the Morgan library, but over the past year it has become deprecated and less useful. We will load it here, but it's a bit messy. I am in the process of cleaning it up. When you initialize a Morgan object, you must pass at least a set of 4 strings. These strings are, in order, the column where the isoform names (unique) reside, the name of the column that holds the regression coefficient from sleuth; the name of the column that holds the TPM values passed by Kallisto and the name of the column that holds the q-values.

We can also add what I call a genmap. A genmap is a file that maps read files to genotypes. A genmap file has three columns: *'project_name'*, *'genotype'* and *'batch'* in that exact order. For this project, the genotypes are coded. In other words, they are letters, *'a'*, *'b'*, *'d'*,... and not specific genotypes. The reason for this is that we wanted to make sure that, at least during the initial phase of the project, I could not unduly bias the results by searching the literature and what not. Because the genotypes are coded, we need to specify which of the letters represent single mutants, and which letters represent double mutants. I also need to be able to figure out what the individual components of a double mutant are. Finally, we need to set the q-value threshold. If no q-value is specified, the threshold defaults to 0.1.

I will now initialize the object. I call it thomas. Then I will load in all the variables we will use; I will load in the genmap, and at last I will load in the datasets that contain the TPM and the Sleuth

β coefficients. After everything has been loaded, I will call `thomas.filter_data`, which drops all the rows that have a β coefficient equal to NaN

```
In [15]: # Specify the genotypes to refer to:
         single_mutants = ['b', 'c', 'd', 'e', 'g']

         # Specify which letters are double mutants and their genotype
         double_mutants = {'a' : 'bd', 'f':'bc'}

         # initialize the morgan.hunt object:
         thomas = morgan.hunt('target_id', 'b', 'tpm', 'qval')
         # input the genmap file:
         thomas.add_genmap('../input/library_genotype_mapping.txt', comment='#')
         # add the names of the single mutants
         thomas.add_single_mutant(single_mutants)
         # add the names of the double mutants
         thomas.add_double_mutants(['a', 'f'], ['bd', 'bc'])
         # set the q-value threshold for significance to its default value, 0.1
         thomas.set_qval()

         # Add the tpm files:
         kallisto_loc = '../input/kallisto_all/'
         sleuth_loc = '../sleuth/kallisto/'
         thomas.add_tpm(kallisto_loc, '/kallisto/abundance.tsv', '')
         # load all the beta dataframes:
         for file in os.listdir("../sleuth/kallisto"):
             if file[:4] == 'beta':
                 letter = file[-5:-4].lower()
                 thomas.add_beta(sleuth_loc + file, letter)
                 thomas.beta[letter].sort_values('target_id', inplace=True)
                 thomas.beta[letter].reset_index(inplace=True)
                 thomas.filter_data()

         # thomas.filter_data()
```

Finally, we will place all the data in a tidy dataframe, where each row is an observation.

```
In [17]: frames = []
         for key, df in thomas.beta.items():
             df['genotype'] = genvar.mapping[key]
             df['code'] = key
             df['sorter'] = genvar.sort_muts[key]
             df.sort_values('target_id', inplace=True)
             frames += [df]

         tidy = pd.concat(frames)
         tidy.dropna(subset=['ens_gene'], inplace=True)
```

```
        # Save to table
        tidy[['ens_gene', 'ext_gene', 'target_id', 'b', 'se_b', 'qval', 'genotype', 'sorter',

        tidy.sort_values('sorter', inplace=True)
```

3 Isoforms Identified in all Genotypes

```
In [5]: total_genes_id = tidy.target_id.unique().shape[0]
        print("Total isoforms identified in total: {0}".format(total_genes_id))
```

```
Total isoforms identified in total: 19676
```

We identified 19,676 isoforms using 7 million reads. Not bad considering there are ~25,000 protein-coding isoforms in *C. elegans*. Each gene has just slightly over 1 isoform on average, so what this means is that we sampled almost 80% of the genome.

4 Differentially Expressed Genes per genotype

Next, let's figure out how many *genes* were differentially expressed in each mutant relative to the wild-type control.

```
In [6]: print('Genotype: DEG')
        for x in tidy.genotype.unique():
            # select the DE isoforms in the current genotype:
            sel = (tidy.qval < q) & (tidy.genotype == x)
            # extract the number of unique genes:
            s = tidy[sel].ens_gene.unique().shape[0]
            print("{0}: {1}".format(x, s))
```

```
Genotype: DEG
rhy-1: 3005
egl-9: 2549
vhl-1: 1275
hif-1: 1075
fog-2: 2840
egl-9;vhl-1: 3654
egl-9 hif-1: 744
```

From the above exploration, we can already conclude that: * *hif-1(lf)* has a transcriptomic phenotype * *hif-1;egl-9(lf)* has a transcriptomic phenotype * The *egl-9* phenotype is stronger than the *vhl-1* or the *hif-1* phenotypes.

We should be careful is saying whether *rhy-1*, *egl-9* and *egl-9;vhl-1(lf)* are different from each other, and the same goes for *hif-1(lf)*, *vhl-1(lf)* and *egl-9;hif-1(lf)* because we set our FDR threshold at 10%. Notice that *egl-9(lf)* and *rhy-1(lf)* are barely 300 genes separated from each other. A bit of wiggle from both, and they might be identical.

5 Pairwise shared transcriptomic phenotypes

5.1 SI Table 1

In order to be able to assess whether two genes are interacting, we must first determine that the mutants we are studying act upon the same phenotype. What defines a phenotype in transcriptomic space? We use an operational definition -- two genotypes share the same phenotype if they regulate more than a pre-specified(and admittedly subjective) number of genes in common between the two of them, agnostic of direction. In our paper, we call this the Shared Transcriptomic Phenotype (STP). Let's figure out to what extent the genes we have studied share the same phenotype.

We will measure the size of the STP using two distinct definitions. The first, percent shared isoforms, is defined as the number of isoforms in the STP divided by the number of differentially expressed isoforms in EITHER of the two mutants being compared. The second measurement, percent internalization, is defined as the number of isoforms in the STP divided by the number of differentially expressed isoforms in the mutant that has the smallest number of differentially expressed isoforms out of the two being compared.

```
In [7]: sig = (tidy.qval < q)
        string = 'pair,STP,% shared,% internalization'

        # print table header
        l = string.split(',')
        pretty.table_print(l, space=30)

        # print rest:
        for i, g1 in enumerate(tidy.genotype.unique()):
            for j, g2 in enumerate(tidy.genotype.unique()[i+1:]):
                tmp = tidy[sig]  # define a temporary dataframe with only DE genes in it

                # find DE genes in either genotype
                DE1 = tmp[tmp.genotype == g1]
                DE2 = tmp[tmp.genotype == g2]

                # find the overlap between the two genotypes:
                overlap = epi.find_overlap([g1, g2], df=tidy, col='genotype')
                n = len(overlap)  # number of DE isoforms in both genotypes
                genes_in_stp = tidy[tidy.target_id.isin(overlap)].ens_gene.unique()
                n_genes_stp = len(genes_in_stp)  # number of DE genes in both genotypes

                # find total number of DE transcripts in either genotype
                OR = ((tmp.genotype == g1) | (tmp.genotype == g2))
                ntot = tmp[OR].target_id.shape[0]

                # find which genotype has fewer DE transcripts
                n_intern = np.min([DE1.shape[0], DE2.shape[0]])

                # print
                string = "{0} & {1},{2},{3:.2g}%,{4:.2g}%".format(g1, g2, n_genes_stp, 100*n/nt
```

```
l = string.split(',')
pretty.table_print(l, space=30)
```

pair	STP	% shared	% int
rhy-1 & egl-9	1808	32%	70%
rhy-1 & vhl-1	879	20%	69%
rhy-1 & hif-1	456	11%	42%
rhy-1 & fog-2	839	14%	29%
rhy-1 & egl-9;vhl-1	1730	26%	57%
rhy-1 & egl-9 hif-1	484	13%	64%
egl-9 & vhl-1	872	23%	68%
egl-9 & hif-1	387	10%	36%
egl-9 & fog-2	782	14%	30%
egl-9 & egl-9;vhl-1	1872	30%	73%
egl-9 & egl-9 hif-1	415	12%	54%
vhl-1 & hif-1	296	12%	27%
vhl-1 & fog-2	450	11%	35%
vhl-1 & egl-9;vhl-1	971	19%	76%
vhl-1 & egl-9 hif-1	323	16%	43%
hif-1 & fog-2	361	8.8%	33%
hif-1 & egl-9;vhl-1	494	10%	46%
hif-1 & egl-9 hif-1	161	8.9%	22%
fog-2 & egl-9;vhl-1	1069	16%	37%
fog-2 & egl-9 hif-1	247	6.6%	32%
egl-9;vhl-1 & egl-9 hif-1	535	12%	70%

The number of genes that is shared between mutants of the same pathway ranges from ~100 genes all the way to ~1,300. However, the hypoxia mutants share between ~140 and ~700 genes in common with another mutant, the *fog-2(lf)* mutant that has never been reported to act in the hypoxia pathway. What are we to make of this? My own conclusion is that *fog-2* probably interacts with effectors downstream of the hypoxia pathway.