Configuration and instructions

Configuration files

Six configuration/data files need to be added/edited when adding a new data set:

the data file (the evolutionary signature), the group_config file (to define molecular feature groups in data set), the data set_config file (configuration for all data sets), the dataset_config2.csv file. Annotation target file and tstats file.

One data file to be generated by python script: similar IDR data file.

All configuration/data files use csv format. The delimiters for column in evolutionary signature data file should be tab (“,” could be included in data file by accident). Other csv file use “,” as delimiter.

All configuration/data files should be put in “data” directory. Data file for download put in “templates” directory.

The data set_config.csv file

This file list all the data sets, add one row when new data set added.
First column: data set name;
Second column: the data file name of new data set to be read by program;
Third column: the group_config file name of new data set to be read;
Fourth column: the similar IDR file of new data set to be read.
Fifth to seventh column: symbols for the example IDR, for gene, protein, IDR name respectively.

The data file

The evolutionary signature zscore data file for the new data set. One file for every data set.
First row is header.

First column: unique name for every IDR, can’t be missing;
Second column: systematic protein name for yeast/Uniprot ID for human, missing value will be filled with “N/A” by program
Third column: Protein common name for yeast/gene name for human, missing value will be filled with “N/A” by program.
All other columns: the Z-Scores for every feature, missing value will be filled with “0” (and 0 will be to used calculate similar IDRs).
Z-Score columns order will be the order to be displayed on the webpage, features in same group should be in adjacent columns in order to properly define groups, groups ordered similarly for mean and log_variance z-scores.

Z-Score columns names should not decorate heavily otherwise the figures and webpage will be messy and hard to understand. But if the mean and log_var z-scores columns named exactly, they will be automatically decorated (".1" added).

The group_config file
The file to define molecular feature groups, one file for every data set. It defines groups using the column position of the molecular features in data file.
First row is header (ignored by program).
First column: specify the columns in the data is z-scores for mean or log variance.
Second column: short name for the group, 1-7 characters. Longer names will cause figure labels overlap. No blank space in short name.
Third column: long name for webpage, more detailed description for features. Blank space allowed. Don't specify mean/variance in long name.
Fourth column: the start column in the data file for this group.
Fifth column: the end column in the data file for this group.
Six column: group id, naturally ordered for all groups.

dataset_config2.csv file
Config file for annotation and tstats file.
First two columns the same as dataset_config.csv.
Third column: annotation file name.
Fourth column: Tstats file name.

Annotation target file and tstats file
Be careful the molecular feature symbol, IDR symbol, annotation term(Go term) symbol should match to all files each other.
The annotation term symbol format: GO id + "," + GO name. If it's not GO term, still separate the symbols in the term by one comma.
Number of IDRs in the two files also should match.

The similar IDR file(sim file)
generated by findsim.py.

Procedures for adding new data set to the website

1. Prepare configuration/data file, put in "data" directory.
2. Run findsim.py with data file in "templates" directory, name the result file and add to data set.config.csv. Running could take hours.

Operation on server

1. Python version: currently 3.7. package needed: Flask (v 1.1.1), pandas, numpy, matplotlib, waitress server. Recommend anaconda to manage package and create virtual environment.
2. Start application in server:
   - Activate virtual environment: conda activate myenv
   - use command: nohup python waitress_server.py.

3. For restarting the app, stop the existing process first. `ps -ef|grep waitress` to view the process.

Trouble shooting

Common problems come from the data files: unexpected symbols, missing values, unexpected data types.

Since the user inputs are limited, easy to do error checking and usually error input only generate no result. Data files on the other hand have a lot of chance to break the code.

For example, just one non-numeric value in zscore cells result in the read_csv function infer the whole column data type incorrectly, from number to string.

Another example: the annotation symbol in annotation file and tstats file should be exactly the same (though you can switch order), missing comma or extra space produce “key error”. The R code for FAIDR will change the symbol of annotation, need be treated (change code or output).