netZooC Documentation

Release 0.2

netZooC

Aug 02, 2020

Contents

netZooC is a catalog of methods for the reconstruction and analysis of network biology methods.

CHAPTER 1

Contents

1.1 Installation guide

To install netZooC on your computer, please check the following requirements:

1.1.1 Requirements

- g++ compiler

1.1.2 Install

- git clone https://github.com/netZoo/netZooC.git
- cd netZooC/zooAnimalFolder
- make
- Upon successful make, please run the program as detailed in the example section of the function description.

PANDA

Written by Kimberly Glass (kglass@jimmy.harvard.edu), available under GPL. As academic code it is provided without warranty. Please contact the author with any comments/questions/concerns. Last updated July, 2014.

PANDA is written in c++ and can be compiled by:

```
g++ PANDA_vc.c -o PANDA
```

To improve runtime speed, it is also highly encouraged to include an optimazation flag:

```
g++ PANDA_v2.c -O3 -o PANDA
```

Running the program without any parameters will return a usage function:

./PANDA

Usage ./PANDA

```
-e (required) file of expression values (can alternately be a list of gene names)

-m (required) pair file of motif edges

-p (optional) pair file of PPI edges

-o (optional) tag for output files

-a (optional) value to be used for update variable, alpha (default=0.1)
```

Additional options (see README):

```
-k (optional) kill the program after it has run k steps (default=1000)

-n (optional) output a "stats" file every n steps (default, no stats file)

-w (optional) file with list of covariate weights

-l (optional) leave out the lth sample when building the network

-j (optional) retain only j samples when building the network

-r (optional) randomization options

-s (optional) value to seed the random number generator (defaults to system time)

-v (optional) verbose output options
```

There are two required parameters, the names for two input files: a tab-delimited text file with the gene expression values and a tab-delimited text file with motif information. You may also choose to specify an string to generate output file names, the default value for this is "PANDA_prediction".

examples:

```
./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -o ToyOutput

./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -p ToyPPIData.txt -a 0.25 -o␣
↪ToyOutput
```

Running these examples will produce a network file: ToyOutput_FinalNetwork.pairs

DATA:

Within the "YeastData" directory, the file "YeastNetwork_allTFxGene" contains the final z-score edge weights for the networks analyzed in "Passing Messages between Biological Networks to Refine Predicted Interactions." This folder also contains the expression, motif and PPI data files used to generate the networks. Please see the supplemental material of the publication for more information on this data.

The "ToyData" directory has some small 'toy' datafiles that can be used to quickly run/understand PANDA.

FILE FORMATS:

Expression data file: In the expression data file the first column must contain gene names, and each subsequent column should contain expression values for those genes across conditions/samples (see "ToyExpressionData.txt"). Note that if fewer than three conditions are contained in the expression data file, PANDA will initialize the co-regulatory network to an identity matrix. This file can also contain header lines, so long as those lines are preceeded by a hashtag (#).

Motif data file: The motif data file contains three columns (see "ToyMotifData.txt"). The first column should contain regulators, the second those regulator's target genes, and the third, an initial weight to give the interactions (recommend 1). Any potential regulator to gene interaction not specified in this file is given a default interaction weight of 0. Any interaction that is multiply defined will be given a weight equal to the sum of the weights given to the instances of that interaction. Note that PANDA will give an warning message if a gene in the motif file is not contained in the expression data file. This file cannot contain header lines.

Protein interaction file: The protein interaction file (optional) contains three columns (see "ToyPPIData.txt"). The first two columns should contain a pair of regulators, and the third, an initial weight to give the interactions (recommend 1). Note that the program will give an warning message if a regulator in this file is not contained in the motif data. This file cannot contain header lines.

Outputted network files (including "stats" files):

The network files contain four columns:

```
TF \t Gene \t Motif-prediction \t PANDA-prediction
```

The values in the fourth, "PANDA-prediction", column, can loosely be interpreted as Z-scores.

ADDITIONAL OPTIONS:

PANDA allows the user to specify a number of options. Some of these have been implemented as we apply the PANDA algorithm to other systems. Any comments/questions/suggestions are welcomes (email: kglass@jimmy.harvard.edu).

"-a" allows a user to specify a different value for the update parameter, alpha. Larger values of alpha causes quicker convergence, and often greater "mixing" of motif and non-motif edges. Recommend values between 0.05 and 0.25. (Author note: The default of 0.1 was chosen since as ChIP-seq validation of human networks appears to show this gives the most accurate results. However for quicker analysis I strongly suggest starting with 0.25 as there is only a small loss in accuracy but an immense speed-up in time).

example: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -a 0.25 -o ToyOutput`

"-k" sets the maximum number of iterations PANDA can run before exiting.

example: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -k 2 -o ToyOutput`

"-n" tells PANDA to print the network every N steps, instead of just the final predicted network.

example: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -n 5 -o ToyOutput`

"-w" triggers PANDA to calculate a weighted Pearson (in lieu of the standard Pearson correlation), using weights specified in a "covariate weight file." The file should contain a single column of numbers. Each row in the file corresponds directly to a column in the expression data file.

example usage: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -w ToyWeights.txt -o ToyOutput`

"-l" can be used to leave out a specified expression sample when building the network. 1 leaves out the first sample, 2 leaves out the second sample, etc.

example usage: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -l 10 -o ToyOutput (leaves out the 10th column of the expression data when building the network)`

"-j" can be used to jack-knife (random sampling without replacement) the samples in the original expression data. The value indicates the number of samples to use building the network, so this value should be less than the total number of samples and greater than two (since one needs at least three data points to calculate a correlation).

example usage: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -j 40 -o ToyOutput (chooses 40 random expression samples/conditions to build network)`

"-r" allows the user to specify several different randomization options. "-r 1" swaps gene labels, such that each row in the expression data is assigned to a random gene. "-r 2" swaps condition labels, such that each column in the expression data is assigned to a random weight value in the covariate weight file. "-r 3" assigns each condition a random weight value and prints out these values. If used in conjuction with the "-w" option, only conditions that have an original weight value greater than zero are assigned random weight values.

examples:

```
./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -r 1 -o ToyOutput

./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -w ToyWeights.txt -r 2 -o␣
↪ToyOutput
```

"-s" allows the user to seed the random number generator (which otherwise defaults to the current system time). Must be an integer greater than zero.

example: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -r 1 -s 1000 -o ToyOutput`

"-v" triggers several verbose output options. "-v 1" tells PANDA to print out the mean and standard-deviation of the availability and responsibility at each iteration. "-v 2" tells PANDA to write files containing all three networks (gene-gene co-regulatory, TF-gene regulatory, and TF-TF co-regulating) at both the initial and final states (producing six files). The gene-gene co-regulatory networks can be very large, so use this option with caution.

example usage: `./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -v 1 -o ToyOutput`

PUMA

PUMA, or **P**ANDA **U**sing **M**icroRNA **A**ssociations, is a regulatory network reconstruction algorithm that uses message passing to model gene regulatory interactions. PUMA can reconstruct gene regulatory networks using both transcription factors and microRNAs as regulators of mRNA expression levels. This Github repository contains both a MATLAB and a C++ version of PUMA.

The code can be compiled using:

```
g++ PUMA.c -O3 -o PUMA
```

To run PUMA with the assumption that all regulators can form complexes (estimate *responsibility* for all regulators, *eg* a gene regulatory prior with transcription factors only):

```
./PUMA -e ToyExpressionData.txt -m ToyMotifData.txt -o ToyOutput
```

To tell PUMA to discriminate between regulators that can, and regulators that cannot cannot form complexes (for example a list of microRNAs in `miRlist.txt`), run:

```
./PUMA -e ToyExpressionData.txt -m ToyMotifData.txt -u miRlist.txt -o ToyOutput
```

The PUMA C++ code was based on the PANDA C++ version 2 sourceforge project: https://sourceforge.net/projects/panda-net/.

1.1.3 Troubleshooting

- To report any installation issue or function bug, please report through opening an issue on github.

1.2 Functions

1.2.1 PANDA

Description:

```
PANDA is an algorithm that creates gene regulatory network through passing messages␣
↪between gene expression networks,
TF-Gene motif priors, and TF-TF PPI networks.
```

Inputs:

```
-e (required) file of expression values (can alternately be a list of gene names)
-m (required) pair file of motif edges
-p (optional) pair file of PPI edges
-o (optional) tag for output files
-a (optional) value to be used for update variable, alpha (default=0.1)
Additional options (see README):
-k (optional) kill the program after it has run k steps (default=1000)
-n (optional) output a "stats" file every n steps (default, no stats file)
-w (optional) file with list of covariate weights
-l (optional) leave out the lth sample when building the network
-j (optional) retain only j samples when building the network
-r (optional) randomization options
-s (optional) value to seed the random number generator (defaults to system time)
-v (optional) verbose output options
```

Outputs:

```
(tag_for_output_file)_FinalNetwork.pairs: Bipartite gene-TF regulatory network where
each line represents an edge in the graph (a TF-GENE pair).
```

Examples:

```
./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -o ToyOutput
./PANDA -e ToyExpressionData.txt -m ToyMotifData.txt -p ToyPPIData.txt -a 0.25 -o␣
↪ToyOutput
```

Publications:

```
https://doi.org/10.1371/journal.pone.0064832
```

Authors:

```
Kimberley Glass
```

Changelog:

```
Version 1 Modifications (May 2013):
1) added "randomseed" variable which allows the user to specific the random number␣
↪generator seed
("srand(randomseed)").  This is useful when doing paired randomizations (e.g. if one␣
↪wants the gene labels to
be swapped the same way for two different sets of input data).
2) added in a second method of randweights (randweight=2 set by specifying -r 3 at␣
↪the command prompt).  This
generates a random-weight value for any covariate weight that was initially greater␣
↪than zero, but leaves
zero weights unchanged.
3) Removed criteria that a protein in the PPI must be a member of the regulatory␣
↪prior.  Instead add in any
"new" proteins into the regulatory prior assuming no known regulatory interactions.
4) Removed criteria that the TF/motif in the regulatory prior must also be a gene in␣
↪the expression data.
This allows regulators to take names that aren't gene names (e.g. a regulator could␣
↪be TAL1::GATA1, but the
genes are in RefSeq annotation).  One limitation is that correlation in expresion␣
↪levels between "TFs" and
genes is no longer calculated.  This information, however, was never used by PANDA,␣
↪so removing the
calculation had the added benefit of freeing up memory.
5) added in another verboseoutput option.  Now setting "-v 2" at the command prompt␣
↪will cause PANDA to print
out additional files recording the initial and final protein-interaction and co-
↪regulatory networks.  Changed
the behavior of the code such that an initial regulatory network is only printed out␣
↪when using this option.
6) Increased the number of Regulators allowed by Program to 1000.
7) Modified function that reads in regulatory and co-regulating information such that␣
↪it can handle
multiple instances.  Single instances in the input files will result in the initial␣
↪value being set equal to
the value in the "weight" column.  If an interaction is multiply defined, the initial␣
↪value will be set
equal to the sum of values associated with these instances in the "weight" column. ␣
↪Undefined instances
are given a default value of 0.
Version 2 Modifications (July 2014):
1) added in "LeaveOutSample" to leave out a single sample from the network␣
↪reconstruction.
2) fixed missing string termination that could cause the terminal window to become␣
↪bold.
3) added in the "JackKnife" option to designate number of samples to use in a jack-
↪knife network.
4) modified the ReadInExpression function to allow users to add additional rows to␣
↪their expression file (likely
header rows), so long as the first character in these rows is a hashtag (#).
5) Changed length of "TF" in regulation struct to be 64 characters, in preparation to␣
↪longer miRNA names.
6) Created default value for outtag so that the -o command-line option is now␣
↪optional instead of required.
```

(continues on next page)

(continued from previous page)

```
7) Defined MAXGENES, MAXTFS, MAXCONDITIONS, BUFSIZE, and MAXPATHLENGTH to allow␣
↪easier manipulation of these
values should they need to be altered.
8) Added some additional outputs that tell the user what the code is doing.
9) Put in catch when normalizing the prior for the case where a TF and its potential␣
↪target gene both have no
targets/inputs (variance of 0 in the prior).
10) removed index, indegree and outdegree parts of the genes and regulation␣
↪structures as they were not being used.
11) Restructured code so that there are much fewer global variables and most are␣
↪declared locally.
12) Modified ReadInExpression function to sum over multiple entries of a gene in the␣
↪expression file.
13) Added in program exits triggers for if input files have more than MAXGENES MAXTFS␣
↪or MAXCONDITIONS.
Potential Future improvements:
* change the Genes.corr and Regulation.P from symmetrix matrices to vectors to save␣
↪on memory space (especially
for the former)
* remove the exp and stderr portions of the Genes and Regulation structs and make␣
↪them local vectors (currently
they are only used to normalize the initial PPI and corr matrices and nothing else).
* parralize the code by adding options for multi-threading for-loops using the openmp␣
↪library, shoud enhance speed.
```

1.2.2 PUMA

Description:

```
    PUMA, or PANDA Using MicroRNA Associations, is an extension of the gene␣
↪regulatory network reconstruction algorithm PANDA.
    PUMA can reconstruct gene regulatory networks using both transcription factors␣
↪and microRNAs as regulators of mRNA expression levels.
```

Inputs:

```
    -e (required) file of expression values (can alternately be a list of gene names)
    -m (required) pair file of motif edges
    -p (optional) pair file of PPI edges
    -o (optional) tag for output files
    -a (optional) value to be used for update variable, alpha (default=0.1)
    Additional options (see README):
    -k (optional) kill the program after it has run k steps (default=1000)
    -n (optional) output a "stats" file every n steps (default, no stats file)
    -w (optional) file with list of covariate weights
    -l (optional) leave out the lth sample when building the network
    -j (optional) retain only j samples when building the network
    -r (optional) randomization options
    -s (optional) value to seed the random number generator (defaults to system time)
    -v (optional) verbose output options
```

Outputs:

```
    (tag_for_output_file)_FinalNetwork.pairs: Bipartite gene-TF regulatory network␣
↪where
    each line represents an edge in the graph (a TF-GENE pair).
```

Examples:

```
    To run PUMA with the assumption that all regulators can form complexes (estimate␣
↪*responsibility* for all regulators, *eg* a gene regulatory prior with␣
↪transcription factors only):
    ./PUMA -e ToyExpressionData.txt -m ToyMotifData.txt -o ToyOutput

    To tell PUMA to discriminate between regulators that can, and regulators that␣
↪cannot cannot form complexes (for example a list of microRNAs in `miRlist.txt`),␣
↪run:
    ./PUMA -e ToyExpressionData.txt -m ToyMotifData.txt -u ToyMiRList.txt -o ToyOutput
```

Publications:

```
    https://www.ncbi.nlm.nih.gov/pubmed/28506242
```

Authors:

```
    Marieke Kuijjer
```

1.3 Tutorials

Tutorials in netZooC are under construction

1.4 Changelog

1.4.1 0.2.0 (2019-6-31)

- PUMA

1.4.2 0.1.0 (2019-5-28)

- PANDA

CHAPTER 2

Indices and tables

- genindex
- modindex
- search