gainLoss project manual

A probabilistic framework for analyzing phyletic patterns

A Brief tutorial for the command-line executable.

gainLoss is used in webservers:

GLOOME: Gain Loss Mapping Engine

CoPAP: Co-evolution of Presence-Absence Patterns

Ofir Cohen [1], Haim Ashkenazy [1], Eli Levy Karin [1], David Burstein [1], and Tal Pupko [1§]

1) Department of Cell Research and Immunology, George S. Wise Faculty of Life Sciences, Tel Aviv University, Ramat Aviv 69978, Israel;

May 2015,

DRAFT

For internal usage only

Contents

Overview and recommended “starting point” for users

The gainLoss is the stand-alone program at the core of the CoPAP webserver used for analyzing co-evolution interaction for binary data.

This draft manual consists of several sections for reference. The “Quick guide”, “Parameters file + brief description” in this manual is a recommended starting point for expert users.

Alternative starting point for expert and novice users alike is to first use the webservers either GLOOME (for gain-loss mapping) or CoPAP (co-evolution computation based on gain-loss mapping).

In this webserver the user utilize GUI to choose the parameters (with brief explanations when using toy example data (in each one can simply “load” such example input) and then download the generated parameters file derived from these setting, modify the path to the input files (0/1 pattern and tree) and use this parameters file to run locally.

Abstracts of the GLOOME and CoPAP methods

GLOOME: Gain Loss Mapping Engine

The evolutionary analysis of presence and absence profiles (phyletic patterns) is provided in this server. It is assumed that the observed phyletic pattern is the result of gain and loss dynamics along a phylogenetic tree. Examples of characters were represented by phyletic patterns include restriction sites, gene families, introns, and indels, to name a few. This main purpose of GLOOME server is to accurately infers branch specific and site specific gain and loss events. The novel inference methodology is based on a stochastic mapping approach utilizing models that reliably capture the underlying evolutionary processes. A variety of features are available including the ability to analyze the data with various evolutionary models, to infer gain and loss events using either stochastic mapping or maximum parsimony, and to estimate gain and loss rates for each character analyzed.

Additional details available at: http://gloome.tau.ac.il/overview.php

CoPAP: Co-evolution of Presence-Absence Patterns

Evolutionary analysis of phyletic patterns (phylogenetic profiles) is widely used in biology, representing presence or absence of characters such as genes, restriction sites, introns, indels, and methylation sites. The phyletic pattern observed in extant genomes is the result of ancestral gain and loss events along the phylogenetic tree. Here we present CoPAP (Co-evolution of Presence-Absence Patterns), a user-friendly web server, which allows for accurate inference of co-evolving characters as manifested by co-occurring gains and losses. CoPAP utilizes state-of-the-art probabilistic methodologies to infer co-evolution and allows for advanced network analysis and visualization. We developed a platform for comparing different softwares for detecting co-evolution, which includes simulated data with pairs of co-evolving sites and independent sites. We use these simulated data to demonstrate that CoPAP performance is better than alternative methods. We exemplify CoPAP utility by analyzing co-evolution among thousands of genes across bacterial 681 genomes. Clusters of co-evolving genes that were detected using our method largely coincide with known biosynthesis pathways and cellular modules, thus exhibiting the capability of CoPAP to infer biologically meaningful interactions.
Additional details available at: http://copap.tau.ac.il/overview.php

Building and running gainLoss

Download and build

The gainLoss package, including C++ source and some binaries, is available from

http://copap.tau.ac.il/source/gainLoss.tar.zip

To build gainLoss, you will need to first obtain the phylogeny library from this site. Once the source code is in place, just follow the instructions on any UNIX/Linux system with the GNU Compiler (gcc/g++) installed.

Windows

Download the [win32] executable file and save it to your computer . This is a simple command line application that can be run from MS-DOS. Do not double click on the program from Windows Explorer since this will not work. See the section below on "Usage" for a list of the program's arguments.

Unix

A.

Use the [unix] executable.

B.

To build gainLoss, you will need to first obtain the phylogeny library from this site. Once the source code is in place, just follow the instructions on any UNIX/Linux system with the GNU Compiler (gcc/g++) installed.

1. Download the [ProjectCode].

2. In order to unzip and untar the files, use:

```
unzip gainLoss.tar.zip
tar -xvf gainLoss.tar
```

This will create the following directories:

```
libs/phylogeny
```

```
programs/gainLoss
```

3. You may use the makefiles to compile the program. You first make the phylogeny library and than the program gainLoss. If this does not work, skip to item 4.

Make sure you are in the directory where you unzipped the files, and type:

```
cd libs/phylogeny
make
```

In order to run the second Makefile, first, get to the gainLoss directory.

```
cd ../../programs/gainLoss
make
```

This will result in an executable file called gainLoss which will reside in the programs/gainLoss directory.

4. In some systems, the makefiles will not be operable. Thus, follow steps 1-2 and compile directly using g++:

1. cd to the library where you unzipped the files.
2. Type:

   ```
   mv libs/phylogeny/* programs/gainLoss/
   ```

3. cd to the gainLoss library

   ```
   cd programs/gainLoss
   ```

4. To compile, type

   ```
   g++ -O3 -o gainLoss *.cpp
   ```

This will result in an executable file called gainLoss which will reside in the src/gainLoss directory.

Program execution

The command line for gainLoss is as follows:
gainLoss *paramFileName*
The needed arguments and various options of the program are depicted in the *paramFileName*. The only necessary argument (and parameter) is the phyletic patterns file which contains the analyzed sequences and the run type for co-evolutionary analysis. In this case, all other parameters are set to their default values.

Quick guide

The minimum input:

```
_seqFile <sequncesFileName> # essential 0/1 file as fasta format
_treeFile <treeFileName> # optional, Newick format

_performParametricBootstapCorrelation 1 # run gainLoss for CoPAP
```

Parameters file + brief description (download).

```
######## gainLoss Parameters file ########
_seqFile  <sequncesFileName> # Minimal input
_treeFile  <treeFileName> # recommended, Newick format

################################################# Model parameters
## if all-zeros positions are missing from the data
_accountForMissingData 1
_minNumOfOnes 1 # e.g., for COG and EggNOG only patterns with 3 or more are observable
_minNumOfZeros 0       # e.g., for indels, there is no position with only 1s => minNumOfZeros=1



#### Mixture model (gain and loss rates distributions are independent)
_gainLossDist 1 # if 1, independent distribution for gain rate and loss rate

_isRootFreqEQstationary 1 # if 0 it will Allow root frequencies to differ from stationary ones


#### distribution type for gain rate, loss rate
# Options = {UNIFORM, GAMMA, GENERAL_GAMMA, GAMMA_PLUS_INV, GENERAL_GAMMA_PLUS_INV};
_gainDistributionType  GENERAL_GAMMA_PLUS_INV
_lossDistributionType  GENERAL_GAMMA_PLUS_INV

#### distribution type for overall rate, relevant only for non-mixture model. i.e., overall rate for gain and loss
# Options = {UNIFORM, GAMMA, GENERAL_GAMMA, GAMMA_PLUS_INV, GENERAL_GAMMA_PLUS_INV};
_rateDistributionType GENERAL_GAMMA_PLUS_INV # general distribution, PLUS_INV = invariant category
```

```
################################################## optimizations
_performOptimizations 1
_performOptimizationsBBL 1     # MLE optimize all branch lengths
_isMultipleAllBranchesByFactorAtStart 1 # MLE-based branch-proportion
_performOptimizationsManyStarts 0 # multiple starting points for optimization

## Generic "knob" to set the optimization level - tradeoff for running time and parameter estimation accuracy
# Options = {VVVlow,VVlow, Vlow, low, mid, high, Vhigh}
_optimizationLevel mid

######################### number of categories in Gamma distribution
## Categories for the gain/loss mixture model.
# Model Running time is proportional to _numberOfLossCategories x _numberOfGainCategories
## If GENERAL_GAMMA_PLUS_INV is selected, an additional invariant rate category is used
_numberOfLossCategories 3
_numberOfGainCategories 3

## Categories for the overall rate, relevant only for non-mixture model
_numberOfRateCategories 4


###############################################################relevant when no estimation/optimization is done
#_userGainLossRatio 1
#_userAlphaGain 1.0
#_userBetaGain 1.0
#_userProbInvariantGain 0.05
#_userAlphaLoss 1.0
#_userBetaLoss 1.0
#_userProbInvariantLoss 0.05
#_userProbInvariantRate  0.1
#_userAlphaRate 0.6
#_userBetaRate 1.0
#_userGain 0.7
#_userLoss 4.5
#_userTheta 0.88
```

Example:

Input – sequences - 0/1 “MSA” file (download)

```
>A
11111111111111101111
>B11
00111111010100000000
>B12
00000111011111101111
>B21
11111111011111101111
>B22
00000111010100000000
>C1
00000111010111100000
>C2
11111111010111100000
>D11
00000111010100000000
>D12
11111111010100000000
>D21
00000111010100000000
>D22
11111111010100000000
>E11
11111011010011100000
>E12
00000011000111100000
>E21
00000011010111100000
>E22
11111011010111100000
>F11
00000111010111101111
>F12
11111011011100000000
>F21
00000111010100000000
>F22
11111011011000001100
```

Input – tree file (download)

```
(A:0.7,((((E11:0.1,E12:0.1):0.5,(E21:0.1,E22:0.1):0.25):0.5,((F11:0.1,F12:0.1):0.5,(F21:0.1,F22:0.1):0.25):0.5):0.5,(((D11:0.1,D
12:0.1):0.5,(D21:0.1,D22:0.1):0.25):0.5,(((B11:0.1,B12:0.1):0.5,(B21:0.1,B22:0.1):0.25):0.5,(C1:0.1,C2:0.01):0.5):0.1):0.3):0.4;
```

Program output

The program outputs several files to the output directory (default "RESULTS"):

Co-evolution related files

- significantCorrelations.isNormForBr.0.txt
  The main output with all significant correlations between pairs of positions. Significance is determined by the confidence level α (defined by the user, defaults to XXX) and a BH FDR procedure to correct for multiple testing. For the p-values of the tests in which the null hypothesis was rejected, please refer the column named "0_pVal".
- significantCorrelations.isNormForBr.0.pairsAboveBH.txt
  This file contains the results for all pairs for which the calculated p-value was below the confidence level α, without correcting for multiple testing

General files

- log.txt
  The output of the program while running including the values of the estimated parameters of the evolutionary model.
- EstimatedParameters.txt
  The values of the evolutionary model parameters used for the co-evolutionary computation.
- TheTree.ph
  The phylogenetic tree with branch lengths (Newick format).
- TheTree.INodes.ph
  The phylogenetic tree with branch lengths with the annotation of the inner nodes.
- seq.fa
  The phyletic pattern sequence analyzed (re-printing the input in fasta format).

- gainLossProbExpPerPosPerBranch.txt

  The most comprehensive output file contains both the expectation and the probability of events at each position and each branch.

  Note: The default is to print only possible events with probability > 0.05 (definable parameter, see below)

- ExpectationPerBranch.txt

  Summing the expectation for each branch over all positions.

- PosteriorExpectationOfChange.txt

  Summing the expectation for each position over **all** branches.

- PosteriorProbabilityOfChange.txt

  Summing the probability for each position over **all** branches.

- gainLossProbExpCountPerPos.txt

  Summing the expectation, the probability and "counts" (see below for details) for each position over all branches, where only events with probability > 0.05 (definable parameter, see below).

- ProbabilityPerPosPerBranch.txt

  A comprehensive output file containing the probability of events at each position and each branch.

  Note: By default, only events with probability > 0.5 (definable parameter, see below) are written.

List of main parameters in gainLoss

The minimum input:

```
_seqFile <sequncesFileName> # essential 0/1 file as fasta format

_treeFile <treeFileName> # optional, newick format


_performParametricBootstapCorrelation 1 # run gainLoss for CoPAP
```

GLOOME options

Evolutionary model

Gain vs. Loss rates:

The simple evolutionary model assumes that gain and loss probabilities may be different but the gain/loss ratio is identical across all sites. The more complex: "Variable gain/loss ratio (mixture)" allows for gain/loss ratio varies among sites as a mixture model.

This mixture options is controlled by:

```
_gainLossDist 1
```

Accepted values: 0 for non-mixture (default) or 1 for mixture.

gain rate defines 0 => 1 dynamics, while loss rate defines 1 => 0 dynamics.

Rate distribution:

The evolutionary model determines how variability among positions in evolutionary rates is modeled "Equal rate for all characters": assuming that a single evolutionary rate characterizes all sites. "Gamma": among site rate variation, assuming that the rate is gamma distributed "Gamma plus invariant category": gamma distributed with an additional invariant rate category (rate~zero).

The “Rate distribution” is controlled by:

```
_rateDistributionType  <rate type>
```

Accepted values: String =
{UNIFORM, GENERAL_GAMMA, GENERAL_GAMMA_PLUS_INV}.

If the "mixture model" option is selected, use:

```
_gainDistributionType  <rate type>
_lossDistributionType  <rate type>
```

Accepted values: String =
{GENERAL_GAMMA, GENERAL_GAMMA_PLUS_INV}.

The continuos gamma distribution is approximated with a discrete number of categories. To control the Number of rate categories use:

```
_numberOfRateCategories <num of categories>
```

If the "mixture model" option is selected,use:

```
_numberOfGainCategories <num of categories>
_numberOfLossCategories <num of categories>
```

Accepted values: Integers. Typically between 2 and 5.

Correction for un-observable data

Likelihood correction is used when specific patterns are not included in the data, such as position absent from all species.
If zero is selected the model allows sites with only zeros to appear (does not account for un-observable data). Select more than one if singletons are also un-observable.
“Minimum number of ones” controls:

```
_minNumOfOnes <min number of ones>
```

Accepted values: Integers between 0 and 3.

Similarly, “Minimum number of zeros” controls:

```
_minNumOfZeros <min number of zeros>
```

Accepted values: *i*ntegers between 0 and 3.

Estimation of model parameters

Likelihood estimation of parameters can be avoided by setting their values based on character counts directly from the phyletic pattern.

This is controlled by:

```
_performOptimizations 0
```

Accepted values: 0 for non-optimization or 1 optimization (default).

Even when the model's free parameters are optimized, likelihood estimation of branch lengths is optional. This is controlled by:

```
_performOptimizationsBBL 1
```

Accepted values: 0 for non-optimization (default) or 1 optimization (default).

Note: branch length optimization depend on selection of "_performOptimizations 0".

Running times can be modified by changing the optimization level. The optimization level is controlled by:

```
_optimizationLevel <optimization level>
```

Accepted values: String =

{Vlow, low, mid, high}.

Co-evolution computation (CoPAP-specific option)

Controls the Minimal value for reported co-evolutionary interactions after correction for multiple tests (i.e., the False Discovery Rate cutoff is a lower value derived from the number of tested interactions). If '1' is selected, all possible co-evolutionary interactions are computed and reported.
The “Co-evolutionary interactions significance level (alpha)” in the webserver control the:

```
_pValueCutOffForBootStrap <significance level>
```

Accepted values: Real within (0,1] range. Typically between 0.001 and 1.

Higher accuracy level allows better estimation of the statistical significance for reported co-evolutionary interactions, but results in a longer running duration. Where the "Accuracy level of co-evolution inference" which controls the:

```
_numberOfIterations2simulate <num of simulations iteration>
```

Accepted values: integers. Typically between 1 and 100.
Note: (1) ‘Fast and approximate’ =2; (2) ‘Normal speed and accuracy’ =10; and (3) ‘Slow and accurate’ =100.

If characters (e.g., genes) have too few evolutionary events, significant co-evolutionary interactions can’t be inferred and thus are omitted). This is set automatically by the program. It is possible to manually set the minimal number of events required in characters to look for co-evolution. The “Minimal number of events required in characters to look for co-evolution” in the webserver controls the:

```
_minNumOfMPEvent2RemoveSimulatedPositions <min required events>
```

Accepted values: integers. Typically between 0 and 5.

And to over-ride the automatic value for minimum required events set:

```
_isUpdateminNumOfMPEvent2RemoveSimulatedPositions 0
```

Accepted values: 0 or 1.