Algorithms in GenAMap

Contents

For further questions / comments / suggestions, please contact `genamap-team@gmail.com`

1 Notation

- Let N be the number of samples and let n, $1 \leqslant n \leqslant N$ be an index over the samples.
- Let d be the number of markers and let i, $1 \leqslant i \leqslant d$ be an index over the markers.
- Let t be the number of traits and let j and k, $1 \leqslant j, k \leqslant t$ be indexes of the traits.
- Let X be the marker value matrix. Let $X_{n,i}$ be the marker value of marker i and individual n.
- Let Y be the trait value matrix. Let $Y_{n,j}$ be the trait value of trait j and individual n. Let Y_j be the vector of trait values of trait j for all individuals.
- Let ρ denote a network over traits. Let $\rho_{j,k}$ be the edge value between traits j and k. We always have $\rho_{j,k} = \rho_{k,j}$ for all j, k, i.e. ρ is symmetric, and $\rho_{j,j} = 0$ for all j.
- Let $\hat{\rho}$ be the network that an algorithm outputs.
- Let B be an association value matrix. Let $B_{i,j}$ be the association value of marker i and trait j.
- Let $\hat{B}$ be the association value matrix that an algorithm outputs.
- Let F be a feature value matrix. Features are additional quantitative information on markers. $F_{d,f}$ is the feature value corresponding to feature f and marker d.
- In case we consider multiple populations in an algorithm, let c be an index ranging over these populations from 1 to C, where C is the total number of populations. X^c is the marker value matrix for population c, Y^c is the trait value matrix for population c and B^c are the association strengths between markers and traits for population c. B^c has the same size as B in the single-population case, except associations are considered separately for each population in the multi-population setting. B then becomes a three-dimensional tensor with indexes i, j and c.
- When we distinguish a trait value matrix into gene expression value matrix and phenotype value matrix, we will call the gene expression value matrix Y and the phenotype value matrix Z. Indexes l and m range over phenotypes. The corresponding networks are called ρ^Y and ρ^Z.

2 Currently Implemented

2.1 Structured Association Mapping

Association Algorithms compute an Association Data set, given a Marker Data set and a Trait Data set as input, and possibly side information. An Association Data set corresponds to an association value matrix where each row corresponds to a marker, each column corresponds to a trait and an entry represents the association strength between the corresponding marker and trait. Entries with large absolute values in the association matrix represent strong associations between markers and traits and values close to or equal to zero represent weak or no association.

2.1.1 Lasso

- Algorithm Code: LAS
- Description: Performs $L1$-penalized least-squares regression from all markers to all traits.
- Reference: [?]
- Formula:
$$\hat{B} = \text{argmin}_B \frac{1}{2}||Y - XB||_2^2 + \lambda|B|_1$$
- Implementation Details: Data is normalized; Regularization parameter λ is chosen via cross-validation

2.1.2 Graph-guided fused Lasso

- Algorithm Code: GF2
- Abbreviation: GfLasso
- Description: Performs $L1$-penalized least-squares regression from all markers to all traits with an additional fusion penalty on regression coefficients pertaining to related traits. Relatedness is measured by a trait network. Hence, this algorithm uses a Network as side information.
- Reference: [?]
- Additional Inputs: Network
- Formula:
$$\hat{B} = \text{argmin}_B \frac{1}{2}||Y - XB||_2^2 + \lambda|B|_1 + \gamma \sum_{i,j,k,j<k} \rho_{j,k}|B_{i,j} - \text{sgn}(\rho_{j,k})B_{i,k}|$$

- Implementation Details: Data is normalized; Regularization parameters λ, γ are chosen via cross-validation; Regular Lasso is used for screening prior to running GfLasso to reduce the number of candidate markers; The trait set is partitioned into subsets using graph cuts and each subset is run independently (Network edges between subsets are dropped)

2.1.3 Graph-constrained fused Lasso

- Algorithm Code: GC2
- Abbreviation: GcLasso
- Description: Performs $L1$-penalized least-squares regression from all markers to all traits with an additional fusion penalty on regression coefficients pertaining to related traits. Relatedness is measured by a trait network. Hence, this algorithm uses a Network as side information. The difference to GfLasso is that the magnitude of the edge weight in the network is not used for the fusion penalty, only its sign.
- Reference: [?]
- Additional Inputs: Network
- Formula:

$$\hat{B} = \text{argmin}_B \frac{1}{2}||Y - XB||_2^2 + \lambda|B|_1 + \gamma \sum_{i,j,k,j<k} |B_{i,j} - \text{sgn}(\rho_{j,k})B_{i,k}|$$

- Implementation Details: Data is normalized; Regularization parameters λ, γ are chosen via cross-validation; Regular Lasso is used for screening prior to running GcLasso to reduce the number of candidate markers; The trait set is partitioned into subsets using graph cuts and each subset is run independently (Network edges between subsets are dropped)

2.1.4 Tree Lasso

- Algorithm Code: TLS
- Description: Performs least-squares regression from all markers to all traits, with an $L2$-penalty defined by overlapping output groups which follow a tree structure. The traits correspond to the leafs of that tree. The tree is derived from a Network through hierarchical clustering. Hence, this algorithm uses a Network as side information.
- Reference: [?]
- Additional Inputs: Network

- Formula:

$$\hat{B} = \text{argmin}_B \frac{1}{2}||Y - XB||_2^2 + \lambda \sum_i \sum_{v \in V} \omega_v ||B_{i,v}||_2$$

  Here, V is the set of tree nodes and $B_{i,v}$ is the vector of entries of B in row i and in columns corresponding to traits for which node v is an ancestor.

- Implementation Details: Data is normalized; Regularization parameter λ is chosen via cross-validation; Regular Lasso is used for screening prior to running Tree Lasso to reduce the number of candidate markers; The trait set is partitioned into subsets using graph cuts and each subset is run independently (Network edges between subsets are dropped); Tree structure and weights ω_v are generated automatically from the Network through hierarchical clustering

2.1.5 Adaptive Multi-Task Lasso

- Algorithm Code: ADL
- Abbreviation: AMTL
- Description: Performs least-squares regression from all markers to all traits, with an $L2$-penalty defined by non-overlapping output groups. The output groups are generated automatically from a Network, which is used as side information. In addition, AMTL puts an $L1$-penalty on each individual component of B, as in Lasso. Furthermore, each penalty term is weighted according to additional parameters (which are optimized over) that combine with Feature Data.
- Reference: [?]
- Additional Inputs: Network, (any number of) Features
- Formula:

$$\begin{aligned} \hat{B} &= \text{argmin}_B \min_{\omega,\kappa,\sum_f \omega_f = \sum_f \kappa_f = 1} \frac{1}{2}||Y - XB||_2^2 + \lambda \sum_{d,t} (F\omega)_d |B_{d,t}| \\ &\quad + \gamma \sum_{d,h} (F\kappa)_d ||B_{d,h}||_2 + Z(F\omega, F\kappa) \end{aligned}$$

  Here, h ranges over the set of output groups and $B_{i,h}$ is the vector of entries of B in row i and in columns corresponding to traits which belong to group h. Z is a Bayesian normalization term.

- Implementation Details: Data is normalized; Regularization parameters λ, γ are chosen via cross-validation; Regular Lasso is used for screening prior to running AMTL to reduce the number of candidate markers; Output Groups are generated automatically from the network; Very large output groups are further partitioned by using graph cuts on the trait network

2.1.6 Multi-Population Group Lasso

- Algorithm Code: MPL
- Abbreviation: MPGL
- Description: Performs least-squares regression from all markers to all traits. Separate association values are generated for each of a number of populations, which correspond to (disjoint) subsets of the Marker Data / Trait Data sets. The analyses are coupled by a group penalty over matching association values for different populations. This causes all such associations to be selected jointly, while allowing different association magnitudes. A Population Structure, which specifies the partitioning of the data into populations, is required as side information.
- Reference: [?]
- Additional Inputs: Population Structure.
- Formula:

$$\hat{B} = \text{argmin}_B \frac{1}{2} \sum_c ||Y^c - X^c B^c||_2^2 + \lambda \sum_{d,t} ||B_{d,t}||_2$$

  Here, $B_{i,j}$ is the vector of entries of B in row i and column j, extending into the third dimension (which ranges over c).

- Implementation Details: Data is normalized; Optimization problem is decomposed and solved separately for each trait; Regularization parameter λ is chosen via cross-validation, independently for each trait; Final results for all populations are averaged to obtain a final association matrix of size d by t

2.1.7 graph-Graph-constrained fused Lasso

- Algorithm Code: GTA
- Abbreviation: gGfLasso
- Description: Performs $L1$-penalized least-squares regression from a Trait Data set representing gene expressions to a Trait Data set representing phenotypes, using an additional fusion penalty on the regression coefficients induced both by a Network over the gene expressions and a Network over the phenotypes. Hence, it uses two Networks as side information plus an addition Association Data set between markers and gene expressions, which will be used to prepare additional information for the GenAMap 3-way association visualization.
- Reference: [?]

- Inputs: Trait Data set representing gene expressions, Trait Data set representing phenotypes, Networks for both Trait Data sets an Association Data set between a Marker Data set and the gene expressions.
- Outputs: Gene-Trait Association Data
- Formula:

$$\begin{aligned}\hat{B} &= \text{argmin}_B \frac{1}{2}||Z - YB||_2^2 + \lambda|B|_1 + \gamma_1 \sum_{j,l,m,l<m} |B_{j,l} - \text{sgn}(\rho^Z_{l,m})B_{j,m}| \\ &\quad +\gamma_2 \sum_{j,k,l,j<k} |B_{j,l} - \text{sgn}(\rho^Y_{j,k})B_{k,l}|\end{aligned}$$

- Implementation Details: Data is normalized; Regularization parameters $\lambda, \gamma_1, \gamma_2$ are chosen via cross-validation; The trait set is partitioned into subsets using graph cuts and subsets are run independently (Network edges between subsets are dropped)

2.2 Hypothesis Testing

2.2.1 Wald/Chi-Squared Test

- Algorithm Code: PNK
- Description: Performs the Wald test (qualitative traits) or chi-squared test (binary traits) as implemented by the PLINK software
- Reference: [?]
- Implementation Details: Data is normalized; Absolute values of association values are thresholded at a small value to achieve a sparse set of associations

3 Implemented in Desktop Version

3.1 Network Algorithms

Network Algorithms compute a Network Data set (a network over traits) given a single Trait Data set as input. A network is a square edge weight matrix of an undirected graph where vertices correspond to traits and the edges correspond to relatedness between traits. A large positive numerical entry in a network corresponds to two highly related traits, a negative numerical entry with large absolute value corresponds to two highly inversely related traits and a numerical entry close to or equal to zero corresponds to unrelated traits.

3.1.1 Correlation

- Algorithm Code: CR1
- Description: Computes the pair-wise correlation between traits, considering each pair independently of all other traits.
- Formula:
$$\hat{\rho}_{j,k} = \frac{\sum_n (Y_{n,j} - \bar{Y_j})(Y_{n,k} - \bar{Y_k})}{\sqrt{\sum_n (Y_{n,j} - \bar{Y_j})^2 \sum_n (Y_{n,k} - \bar{Y_k})^2}}$$
- Implementation Details: Data is normalized; Absolute values of network entries are thresholded at a small value to achieve a sparse network

3.1.2 Correlation squared

- Algorithm Code: CR2
- Description: Computes the square of the pair-wise correlation between traits, considering each pair independently of all other traits.
- Formula:
$$\hat{\rho}_{j,k} = \frac{(\sum_n (Y_{n,j} - \bar{Y_j})(Y_{n,k} - \bar{Y_k}))^2}{\sum_n (Y_{n,j} - \bar{Y_j})^2 \sum_n (Y_{n,k} - \bar{Y_k})^2}$$
- Implementation Details: Data is normalized; Absolute values of network entries are thresholded at a small value to achieve a sparse network

3.1.3 Scale-free Network

- Algorithm Code: SFN
- Reference: [?]
- Implementation Details: Data is normalized; Absolute values of network entries are thresholded at a small value to achieve a sparse network

3.1.4 SFN with Topological Overlap Matrix

- Algorithm Code: TOM
- Reference: [?].
- Implementation Details: Data is normalized; Absolute values of network entries are thresholded at a small value to achieve a sparse network

3.1.5 Graphical Lasso

- Algorithm Code: GLO
- Description: Fits a Gaussian graphical model to the trait data by maximizing the log-likelihood penalized the $L1$-norm of the precision matrix (= inverse covariance matrix), except that diagonal elements are set to zero. The network itself is equal to the precision matrix.
- Reference: [?]
- Formula:

$$\hat{\rho} = \text{argmax}_{\rho,\ \rho \text{ non-negative definite}} \log\det\rho - tr(Y^T Y\rho) - \lambda|\rho|_1$$

- Implementation Details: Data is normalized; Regularization parameter λ is chosen via cross-validation

3.2 Population Analysis

3.2.1 Population Analysis

- Algorithm Code: PAA
- Description: This algorithm is a conglomerate of four different tests. Each test is conducted independently on marker-trait pairs and each population and the resulting association values are then averaged across populations. The four tests are: a cross-validation based test, the Wald Test (continuous trait) / chi-squared test (binary trait) as implemented in PLINK, a likelihood test and a two-sided t-test on the trait distribution by marker value.
- References: [?] [?] [?]
- Additional Inputs: Population Structure
- Further explanation: The cross-validation based test is conducted in the following manner. We split each population into training and test set (10-fold). We then regress the trait on the marker using the training set, obtaining a scalar regression coefficient. The prediction error obtained on the test set is our test statistic.
- Implementation Details: Data is normalized; Absolute values of association values are thresholded at a small value to achieve a sparse set of associations

3.2.2 Structure

- Algorithm Code: STR
- Description: Computes a Population Structure for a Marker Data set. For details, please see the references.
- References: [?] [?]
- Input: Marker Data set
- Output: Population Structure

3.2.3 Agglomerate Hierarchical Clustering

- Algorithm Code: HCL
- Description: Computes a tree structure over traits using a Network through agglomerate hierarchical clustering and then uses the ordering of the leaf nodes in that tree to define a permutation of the traits that pulls related traits together. Hierarchical clustering is performed by successively merging sets of traits. At each step, we choose to merge the two sets of traits for which the average of the absolute values of edges from one set to the other in the Network is maximized.
- Input: Network
- Output: Clustering

3.2.4 Agglomerate Hierarchical Clustering (for trees)

- Algorithm Code: TRE
- Description: Computes a Tree (tree structure over traits) using a Network through agglomerate hierarchical clustering, as above.
- Input: Network
- Output: Tree

3.3 Trait Analysis

3.3.1 Module Analysis

- Algorithm Code: GMD
- Description: Computes the top 20 tightly-connected modules (sets of traits) in a Network which are contiguous with respect to a specified Clustering, i.e. the indices of the module are adjacent under the permutation induced by the Clustering. The algorithms runs in a greedy fashion by selecting the trait subset with the largest average internal node degree,

then repeating the procedure on the remaining nodes. The internal degree of a node in a subset is defined as the sum of absolute weights of edges originating from that node and ending inside the subset. Furthermore, for each module, we compute the eQTL enrichment induced by an Association Data set, which is an additional input to this algorithm.

- Reference: [**?**]
- Input: Network, Clustering, Association Data
- Output: Modules
- Implementation Details: Modules must have at least size 20.

3.3.2 Top k connected traits

- Algorithm Code: N / A
- Description: Computes, for each trait, the sum of absolute edge weights to all other traits, then selects the top k traits under this ranking.
- Input: Network
- Output: Subset

3.3.3 Top connected trait and top k neighbors

- Algorithm Code: N / A
- Description: Computes, for each trait, the sum of absolute edge weights to all other traits, then selects the top trait under this ranking. After this, it selects the traits with the k largest absolute edge weights to this trait.
- Input: Network
- Output: Subset