AMEND Tutorial

Introduction

A powerful approach for the analysis of omics data is to integrate them with molecular interaction networks. Specifically, the integration of microarray/RNA-seq data with protein-protein interaction (PPI) networks has emerged as an effective way to elucidate important genes involved in a biological process. These approaches are often called active module identification (AMI) methods and have the goal of finding a subset of genes (either connected or disconnected) in the PPI network that are relevant to the biological conditions of the experiment.

AMEND is an AMI method that takes as input a PPI network and gene-wise experimental scores (e.g., log fold change) and returns a connected module. AMEND relies on random walk with restart (RWR) and a heuristic solution to the maximum-weight connected subgraph (MWCS) problem to iteratively filter out genes until an optimal subnetwork is found. At each iteration, the current network is input into RWR, with the gene-wise experimental scores serving as seed values. This produces node weights, which are shifted downwards by a certain quantile (called the filtering rate), resulting in both positive and negative node weights. These weights are used to find a maximum-weight connected subgraph. AMEND uses a heuristic solution first implemented in the BioNet package. This produces a subnetwork, which is the input for the next iteration. Each subnetwork is scored by the product of the mean standardized experimental scores (standardized w.r.t. all genes in original network) and the mean core-clustering coefficient (a measure of node connectivity). The process stops when there is no change in subnetwork between iterations or when only 2 nodes remain in the subnetwork. The subnetwork with the largest score is returned.

A key concept in AMEND is the filtering rate, which determines how the untreated RWR scores are shifted before input into the heuristic MWCS solution. The filtering rate is actually a quantile of the untreated RWR scores. As the quantile decreases, each RWR score is subtracted by a smaller number, resulting in fewer negatively weighted nodes, which results in fewer genes filtered out by the MWCS solution. In this sense, the quantile used to shift the untreated RWR scores is a filtering rate. This filtering rate follows an exponential decay schedule, which has two hyperparameters: the starting filtering rate and the decay value.

The decay parameter determines the rate at which the shifting quantile decreases. This value is determined by simulation. The decay is set to the maximum value that will allow the algorithm to arrive at a subnetwork of size n. If the decay is too large, the filtering rate will approach zero too quickly. This causes the algorithm to stop early since no nodes will be removed with a filtering rate of zero. The parameter n is set by the user and approximates the size of the final module.

The primary function is `run_AMEND()`, which implements the AMEND algorithm and returns a connected subnetwork.

Installation

AMEND is hosted on GitHub and can be installed by running the following code.

```
devtools::install_github("samboyd0/AMEND", build_vignettes = TRUE)
```

Example

This example will focus on a gene expression microarray experiment to illustrate a typical use case for AMEND.

GLUT4 Data Description

The dataset that will be used here is a GLUT4 knockout-overexpression (KO-OX) microarray experiment in mouse adipose tissue and is available on the NCBI's Gene Expression Omnibus under accession GSE35378. GLUT4 is a glucose transporter protein involved in the uptake of glucose into the cell. The experiment involved 4 groups of 3 mice each: GLUT4 KO, KO control, GLUT4 OX, and OX control.

The data is contained in the `AMEND` package. Let's inspect the PPI network and the vectors of data values, which will be described in the next section.

```
# Mus musculus PPI network.
# This is a reduced version of the full PPIN obtained by
# taking the largest cluster from the Louvain topological clustering algorithm
glut4_graph
#> IGRAPH 082094e UNW- 1033 8052 --
#> + attr: name (v/c), symbol (v/c), ECI (v/n), logFC (v/n), weight (e/n)
#> + edges from 082094e (vertex names):
#>  [1] ENSMUSP00000029445--ENSMUSP00000051619 ENSMUSP00000029445--ENSMUSP00000026572 ENSMUSP0000005161
#>  [5] ENSMUSP00000051619--ENSMUSP00000032399 ENSMUSP00000026572--ENSMUSP00000032399 ENSMUSP0000002944
#>  [9] ENSMUSP00000032399--ENSMUSP00000005671 ENSMUSP00000029445--ENSMUSP00000065983 ENSMUSP0000005161
#> [13] ENSMUSP00000032399--ENSMUSP00000065983 ENSMUSP00000051619--ENSMUSP00000079380 ENSMUSP0000006598
#> [17] ENSMUSP00000079380--ENSMUSP00000007959 ENSMUSP00000029445--ENSMUSP00000066238 ENSMUSP0000005161
#> [21] ENSMUSP00000099759--ENSMUSP00000066238 ENSMUSP00000032399--ENSMUSP00000066238 ENSMUSP0000006598
#> [25] ENSMUSP00000079380--ENSMUSP00000095832 ENSMUSP00000007959--ENSMUSP00000095832 ENSMUSP0000002944
#> [29] ENSMUSP00000032399--ENSMUSP00000056774 ENSMUSP00000005671--ENSMUSP00000056774 ENSMUSP0000007938
#> + ... omitted several edges

# Named vector of ECI scores (Equivalent Change Index)
head(eci_scores)
#> ENSMUSP00000029445 ENSMUSP00000030834 ENSMUSP00000051619 ENSMUSP00000026572 ENSMUSP00000099759 ENSMU
#>       1.310398e-01       -2.548818e-02       -2.540581e-01       -1.702958e-02       -3.653043e-32

# Named vector of log fold changes for GLUT4-KO vs. Control
head(logFC_KO)
#> ENSMUSP00000029445 ENSMUSP00000030834 ENSMUSP00000051619 ENSMUSP00000026572 ENSMUSP00000099759 ENSMU
#>        0.078413720        0.278489588       -0.017334667        0.024667385       -0.566580461
```

Equivalent Change Index

AMEND was developed to accommodate a recently introduced metric, the equivalent change index (ECI). The ECI measures the extent to which a gene is equivalently or inversely expressed between two treatment-control comparisons. It ranges between -1 and 1, with a value of -1 indicating changes in expression in exactly opposing ways (e.g., expression was halved between groups for one experiment but doubled for the other), and a value of 1 indicating changes in expression in exactly equivalent ways (e.g., expression was doubled between groups for both experiments). Formally, the ECI for gene i is

$$\lambda_i = sign(\beta_{i1} * \beta_{i2}) \frac{min(|\beta_{i1}|, |\beta_{i2}|)}{max(|\beta_{i1}|, |\beta_{i2}|)} (1 - max(p_{i1}, p_{i2}))$$

where β_{ij} represents the log fold change and p_{ij} the p-value for gene i from experiment j.

Running AMEND with ECI

Before applying any AMI method, it is important to clarify the biological question of interset that we want to answer. For the GLUT4 KO-OX experiment, it may be of interest to know which genes are affected in opposing ways by the two treatments. A gene that is up-regulated in the KO-control arm and down-regulated in the OX-control arm suggests a close association of that gene with GLUT4. The ECI is well suited to answer this type of question. Since we are interested in inversely regulated genes, we will want to set `data.type = "ECI"` and `DOI = "negative"`. We will set `n = 25` to specify the approximate size of the final module.

The `normalize` argument specifies how to normalize the adjacency matrix for random walk with restart (RWR). `normalize = "core"` will use node coreness to normalize the adjacecy matrix, whereas `normalize = "degree"` will column normalize the adjacency matrix using node degree. The seed.weight argument specifies how to transform the ECI values for use as seed values in RWR. Seed values must be non-negative, but ECI has a range of $[-1, 1]$, necessitating some transformation. The scheme used in AMEND is to take the absolute value of the ECIs, then weight the values *not* in the direction of interest by some constant in $[0, 1]$. For example, when interested in negative ECIs, $seed.weight = 0.5$ translates into weighting a positive ECI gene half that of a negative ECI gene of equal magnitude.

```
# Using the igraph object as input
module = run_AMEND(graph = glut4_graph, n = 25, data.type = "ECI", DOI = "negative",
                   normalize = "degree", seed.weight = 0.5)
#> Starting filtering rate: 0.4978
#> Iteration: 1
#> Iteration: 2
#> Iteration: 3
#> Iteration: 4
#> Iteration: 5
#> Iteration: 6
#> Iteration: 7
#> Iteration: 8
#> Iteration: 9
#> Iteration: 10
#> *** Converged! *** ID=1

# Can also use the adjacency matrix and vector of node scores
if(0){
  module = run_AMEND(adj_matrix = glut4_adjM, node_scores = eci_scores, n = 25, data.type = "ECI",
                     DOI = "negative", normalize = "degree", seed.weight = 0.5)
}
```

Let's inspect the module returned by `run_AMEND()`. A named list is returned containing the final module, module score, a list of node names contained in the intermediate subnetworks, network statistics for all iterations, the runtime, and a list of the input parameters.

```
# The final module
module$module
#> IGRAPH 676c7f2 UNW- 45 72 --
#> + attr: name (v/c), symbol (v/c), ECI (v/n), logFC (v/n), seeds (v/n), Z (v/n), weight (e/n)
#> + edges from 676c7f2 (vertex names):
#>  [1] ENSMUSP00000051619--ENSMUSP00000103981 ENSMUSP00000056720--ENSMUSP00000028259 ENSMUSP0000005161
#>  [5] ENSMUSP00000028259--ENSMUSP00000099890 ENSMUSP00000032198--ENSMUSP00000099890 ENSMUSP0000005161
#>  [9] ENSMUSP00000005164--ENSMUSP00000097547 ENSMUSP00000051619--ENSMUSP00000067786 ENSMUSP0000003414
#> [13] ENSMUSP00000051619--ENSMUSP00000021090 ENSMUSP00000034148--ENSMUSP00000021090 ENSMUSP0000006778
#> [17] ENSMUSP00000051619--ENSMUSP00000121111 ENSMUSP00000005164--ENSMUSP00000121111 ENSMUSP0000005161
#> [21] ENSMUSP00000070019--ENSMUSP00000131010 ENSMUSP00000070019--ENSMUSP00000105179 ENSMUSP0000013101
#> [25] ENSMUSP00000067786--ENSMUSP00000112765 ENSMUSP00000105179--ENSMUSP00000031606 ENSMUSP0000010517
```

```
#> [29] ENSMUSP00000051619--ENSMUSP00000040307 ENSMUSP00000107576--ENSMUSP00000040307 ENSMUSP0000003219
#> + ... omitted several edges

# data.frame of network statistics for all iterations
module$stats
#>   Decay Restart parameter Network score Avg Z Avg CCC Nodes Edges Density Filtering rate Observed fi
#> 1  0.19              0.85         0.139 0.164   0.851   470  2704   0.025          0.498
#> 2  0.18              0.60         0.440 0.520   0.847   269  1635   0.045          0.416
#> 3  0.19              0.95         0.720 0.867   0.830   157   421   0.034          0.344
#> 4  0.19              0.90         0.957 1.167   0.820    98   238   0.050          0.284
#> 5  0.20              0.75         1.169 1.359   0.860    76   163   0.057          0.233
#> 6  0.21              0.95         1.226 1.458   0.841    57    89   0.056          0.189
#> 7  0.21              0.55         1.308 1.512   0.866    48    77   0.068          0.153
#> 8  0.21              0.95         1.359 1.588   0.856    45    72   0.073          0.124
#> 9  0.21              0.95         1.354 1.575   0.860    43    70   0.078          0.101

# Runtime
module$time
#> Time difference of 13.9617 secs
```

Running AMEND with log fold change

AMEND can also accommodate log fold changes. We have log fold changes for the GLUT4-KO vs. control DE analysis. Suppose we are interested in genes with large log fold changes, regardless of direction. Then we would set `data.type = "logFC"` and `DOI = "both"`.

```
# Using the igraph object as input
module2 = run_AMEND(graph = glut4_graph, n = 25, data.type = "logFC", DOI = "both",
                    normalize = "degree", seed.weight = 0.5)
#> Starting filtering rate: 0.4978
#> Iteration: 1
#> Iteration: 2
#> Iteration: 3
#> Iteration: 4
#> Iteration: 5
#> Iteration: 6
#> Iteration: 7
#> Iteration: 8
#> Iteration: 9
#> Iteration: 10
#> Iteration: 11
#> Iteration: 12
#> Iteration: 13
#> Iteration: 14
#> *** Converged! *** ID=1

# Can also use the adjacency matrix and vector of node scores
if(0){
  module2 = run_AMEND(adj_matrix = glut4_adjM, node_scores = logFC_KO, n = 25, data.type = "logFC",
                      DOI = "both", normalize = "degree", seed.weight = 0.5)
}
```

Here are the results.

```
# The final module
module2$module
#> IGRAPH a8cde46 UNW- 36 76 --
#> + attr: name (v/c), symbol (v/c), ECI (v/n), logFC (v/n), seeds (v/n), Z (v/n), weight (e/n)
#> + edges from a8cde46 (vertex names):
#>  [1] ENSMUSP00000079380--ENSMUSP00000007959 ENSMUSP00000079380--ENSMUSP00000056774 ENSMUSP0000000795
#>  [5] ENSMUSP00000056774--ENSMUSP00000001780 ENSMUSP00000001780--ENSMUSP00000016673 ENSMUSP0000005677
#>  [9] ENSMUSP00000016673--ENSMUSP00000066743 ENSMUSP00000064394--ENSMUSP00000066743 ENSMUSP0000006674
#> [13] ENSMUSP00000079380--ENSMUSP00000097547 ENSMUSP00000007959--ENSMUSP00000097547 ENSMUSP0000005677
#> [17] ENSMUSP00000064394--ENSMUSP00000097547 ENSMUSP00000066743--ENSMUSP00000097547 ENSMUSP0000005677
#> [21] ENSMUSP00000007959--ENSMUSP00000112765 ENSMUSP00000056774--ENSMUSP00000112765 ENSMUSP0000009754
#> [25] ENSMUSP00000079380--ENSMUSP00000004986 ENSMUSP00000097547--ENSMUSP00000004986 ENSMUSP0000005677
#> [29] ENSMUSP00000091238--ENSMUSP00000030747 ENSMUSP00000091238--ENSMUSP00000107214 ENSMUSP0000003074
#> + ... omitted several edges

# data.frame of network statistics for all iterations
module2$stats
#>    Decay Restart parameter Network score Avg Z Avg CCC Nodes Edges Density Filtering rate Observed f
#> 1   0.19              0.95         0.349 0.410   0.850   503  3588   0.028          0.498
#> 2   0.18              0.95         0.673 0.818   0.823   267  1325   0.037          0.416
#> 3   0.18              0.95         1.008 1.225   0.823   164   559   0.042          0.347
#> 4   0.19              0.95         1.282 1.553   0.826   112   354   0.057          0.287
#> 5   0.19              0.95         1.565 1.863   0.840    85   214   0.060          0.238
#> 6   0.20              0.95         1.848 2.085   0.886    69   165   0.070          0.194
#> 7   0.20              0.90         1.982 2.272   0.872    59   133   0.078          0.159
#> 8   0.18              0.90         2.115 2.425   0.872    53   113   0.082          0.133
#> 9   0.15              0.90         2.213 2.525   0.876    49    98   0.083          0.114
#> 10  0.12              0.50         2.301 2.611   0.881    44    93   0.098          0.102
#> 11  0.12              0.75         2.400 2.717   0.883    41    88   0.107          0.090
#> 12  0.14              0.70         2.514 2.830   0.888    37    81   0.122          0.078
#> 13  0.12              0.80         2.577 2.859   0.901    36    76   0.121          0.069

# Runtime
module2$time
#> Time difference of 13.85992 secs
```

Suppose that we want to investigate an intermediate subnetwork that was generated during `run_AMEND()`, either because the final module was too small or because we want to see which genes were filtered out. From the output of `module2$stats`, let's choose the subnetwork from iteration 6. We can retrieve this subnetwork by using the function `get_subnetwork()`. We can also look at the nodes in this subnetwork directly from the `module2` object.

```
# Using get_subnetwork()
subnet6 = get_subnetwork(amend_object = module2, k = 6)
subnet6
#> IGRAPH 78113dc UNW- 69 165 --
#> + attr: name (v/c), symbol (v/c), ECI (v/n), logFC (v/n), seeds (v/n), Z (v/n), weight (e/n)
#> + edges from 78113dc (vertex names):
#>  [1] ENSMUSP00000079380--ENSMUSP00000007959 ENSMUSP00000005671--ENSMUSP00000056774 ENSMUSP0000007938
#>  [5] ENSMUSP00000007959--ENSMUSP00000001780 ENSMUSP00000056774--ENSMUSP00000001780 ENSMUSP0000005677
#>  [9] ENSMUSP00000001780--ENSMUSP00000016673 ENSMUSP00000005671--ENSMUSP00000064394 ENSMUSP0000005677
#> [13] ENSMUSP00000079380--ENSMUSP00000115883 ENSMUSP00000007959--ENSMUSP00000115883 ENSMUSP0000005677
#> [17] ENSMUSP00000016673--ENSMUSP00000115883 ENSMUSP00000001780--ENSMUSP00000091238 ENSMUSP0000000795
#> [21] ENSMUSP00000001780--ENSMUSP00000102538 ENSMUSP00000091238--ENSMUSP00000102538 ENSMUSP0000001667
```

```
#> [25] ENSMUSP00000066743--ENSMUSP00000020970 ENSMUSP00000066743--ENSMUSP00000099889 ENSMUSP0000002097
#> [29] ENSMUSP00000005671--ENSMUSP00000005164 ENSMUSP00000007959--ENSMUSP00000005164 ENSMUSP0000011588
#> + ... omitted several edges

# Directly from module2
module2$subnetworks[[6]]
#>  [1] "ENSMUSP00000099759" "ENSMUSP00000005671" "ENSMUSP00000079380" "ENSMUSP00000007959" "ENSMUSP000
#>  [8] "ENSMUSP00000016673" "ENSMUSP00000064394" "ENSMUSP00000115883" "ENSMUSP00000091238" "ENSMUSP000
#> [15] "ENSMUSP00000020970" "ENSMUSP00000099889" "ENSMUSP00000025691" "ENSMUSP00000005164" "ENSMUSP000
#> [22] "ENSMUSP00000066209" "ENSMUSP00000032201" "ENSMUSP00000112765" "ENSMUSP00000124963" "ENSMUSP000
#> [29] "ENSMUSP00000088837" "ENSMUSP00000030747" "ENSMUSP00000107214" "ENSMUSP00000022599" "ENSMUSP000
#> [36] "ENSMUSP00000003154" "ENSMUSP00000029674" "ENSMUSP00000030714" "ENSMUSP00000004326" "ENSMUSP000
#> [43] "ENSMUSP00000020157" "ENSMUSP00000035092" "ENSMUSP00000027189" "ENSMUSP00000110746" "ENSMUSP000
#> [50] "ENSMUSP00000140404" "ENSMUSP00000032705" "ENSMUSP00000129001" "ENSMUSP00000023687" "ENSMUSP000
#> [57] "ENSMUSP00000127921" "ENSMUSP00000047680" "ENSMUSP00000018066" "ENSMUSP00000029964" "ENSMUSP000
#> [64] "ENSMUSP00000048978" "ENSMUSP00000104825" "ENSMUSP00000022638" "ENSMUSP00000073612" "ENSMUSP000
```