Introduction to the `dna` package

Ryan Gill[*], Somnath Datta[†], Susmita Datta[†]

March 22, 2014

1 Introduction

In this vignette, a description is given for an **R** (R Development Core Team, 2014) package `dna` which implements statistical tests for differential network analysis described in Gill, Datta, and Datta (2010) based on connectivity scores. Built-in options for computing the connectivity scores include partial least squares (PLS), principal components, ridge regression, and correlation; the package also provides the option for the user to specify a custom method for computing the scores. An example of using the package with a custom method based on the least absolute shrinkage and selection operator (LASSO) is presented with code in Section 4.

Once the method for computing the scores has been selected, the following three types of tests are implemented by the `dna` package.

1. Test whether the connectivity scores for a single gene differ between the two networks.

2. Test whether the connectivity scores for a set of important genes differ between the two networks.

3. Test whether the overall modular structure differs between the two networks.

For the last type, a definition of what is meant by a module is required based on the connectivity scores. Therefore, the `dna` package has functions for determining and working with the modular structure of a network based on its corresponding scores.

In Section 2, we describe the three types of statistical tests as well and briefly reviews the mathematical background for the connectivity scores and modules implemented within the `dna` package. Section 3 illustrates each of the tests with several connectivity scores applied to a real data example. Finally, some additional details and features of the package are discussed in Section 4.

[*]Department of Mathematics, University of Louisville
[†]Department of Bioinformatics and Biostatistics, University of Louisville

2 Differential network analysis methodology

The three statistical tests introduced in Gill, Datta, and Datta (2010) are described in this section. Each of these tests are based on one of the methods (PLS, principal components, ridge regression, correlation, or any other) for computing connectivity scores. The tests for differential connectivity in Sections 2.1 and 2.2 also depend on a distance function D which measures the distance between each pair of scores; the test for modular structure in Section 2.3 depends on the modules obtained from the connectivity scores. The methods for computing the connectivity scores that are directly implemented by the `dna` package are described in Section 2.4.

2.1 Tests for an individual gene

This section reviews the test proposed in Gill, Datta, and Datta (2010) to determine whether the difference between the connectivity scores for an individual gene g in two networks is statistically significant. The difference in connectivity in the two networks is measured by the statistic

$$d(g) = \frac{1}{p-1} \sum_{g' \in \mathcal{G}, g \neq g'} D(s^1_{gg'}, s^2_{gg'})$$

where $\mathcal{G}$ is the set of all common genes in the networks, p is the number of genes in $\mathcal{G}$, $s_{gg'}$ is the connectivity score between gene g and g', and D is a distance function specified by the user.

To test the significance of this statistic $d(g)$, Gill, Datta, and Datta (2010) proposed the following permutation test. Let X_1 and X_2 be the $N_1 \times p$ and $N_2 \times p$ matrices of expression values for the two respective networks. Create an $(N_1 + N_2) \times p$ matrix $\mathbb{X}$ by stacking the rows of X_1 and X_2. Then repeat the calculation of the connectivity scores and $d(g)$ for a specified number of random permutations of the rows of $\mathbb{X}$. Specifically, for each random permutation π of the rows of $\mathbb{X}$, we obtain a new $(N_1 + N_2) \times p$ matrix $\mathbb{X}^\pi$, and we let X_1^π denote the first N_1 rows of this matrix and let X_2^π denote the last N_2 rows of this matrix. New connectivity scores $s^\pi_{g,g'}$ and the test statistic

$$d^\pi(g) = \frac{1}{p-1} \sum_{g' \in \mathcal{G}, g \neq g'} D(s^{\pi 1}_{gg'}, s^{\pi 2}_{gg'})$$

are computed for the networks represented by X_1^π and X_2^π. The p-value for this permutation test is approximated by the proportion of times that $d^\pi(g)$ is at least as large as $d(g)$ among the random permutations selected, and the connectivity scores for the gene are statistically significant if the p-value is sufficiently small.

2.2 Tests for a class of genes

Gill, Datta, and Datta (2010) also proposed a test to determine whether the difference between the connectivity scores for a class of "important" genes $\mathcal{F}$ in

two networks is statistically significant. The difference in connectivity in the two networks is measured by the statistic

$$\Delta(\mathcal{F}) = \frac{1}{f(f-1)} \sum_{g,g' \in \mathcal{F}, g \neq g'} D(s^1_{gg'}, s^2_{gg'}).$$

where f is the number of genes in $\mathcal{F}$, and D is a distance function specified by the user. To test the significance of $\Delta(\mathcal{F})$, a permutation test similar to that described in Section 2.1 is used with $d(g)$ and $d^\pi(g)$ replaced by $\Delta(\mathcal{F})$ and

$$\Delta^\pi(\mathcal{F}) = \frac{1}{f(f-1)} \sum_{g,g' \in \mathcal{F}, g \neq g'} D(s^{\pi 1}_{gg'}, s^{\pi 2}_{gg'}).$$

2.3 Tests for overall modular structure

Once the scores have been computed, it is often of interest to find modules of genes which are connected with each other. One approach is to use the definition of a module given in Gill, Datta, and Datta (2010). A module with minimum size parameter m and threshold connectivity parameter ϵ is a set of genes $\mathcal{F}$ such that

1. the cardinality of $\mathcal{F}$ is at least m, and
2. given any two genes f_1 and f_2 in $\mathcal{F}$, they are connected by a path of genes in $\mathcal{F}$, $f_1 = g_1, \ldots, g_k = f_2$, for some $k \geq 2$, such that the connectivity score of each pair on the path is at least ϵ in magnitude.

To define modules in this manner, it must be assumed that the matrix of connectivity scores has been symmetrized.

To determine if the modular structures of two networks differ significantly from each other, the following test proposed by Gill, Datta, and Datta (2010) can be used. Suppose $\mathcal{M}_1 = \{\mathcal{F}_{11}, \ldots, \mathcal{F}_{1J_1}\}$ and $\mathcal{M}_2 = \{\mathcal{F}_{21}, \ldots, \mathcal{F}_{2J_2}\}$ are the collections of all distinct modules in the respective networks. Let $\mathcal{G}_0 = \bigcap_{i=1}^2 \bigcup_{j=1}^{J_i} \mathcal{F}_{ij}$ be the set of all genes present in some module in both networks. Then the test statistic based on these modules is

$$\mathcal{N} = 1 - \sum_{g \in \mathcal{G}_0} \frac{|\mathcal{F}_{1j(g)} \cap \mathcal{F}_{2j(g)}|}{|\mathcal{F}_{1j(g)} \cup \mathcal{F}_{2j(g)}|}$$

where $\mathcal{F}_{ij(g)}$ is the module in network i that contains gene g. To test the significance of $\mathcal{N}$, a permutation test similar to that described in Section 2.1 is used with $d(g)$ and $d^\pi(g)$ replaced by $\mathcal{N}$ and

$$\mathcal{N}(\pi) = 1 - \sum_{g \in \mathcal{G}_0(\pi)} \frac{|\mathcal{F}_{1j(g)}(\pi) \cap \mathcal{F}_{2j(g)}(\pi)|}{|\mathcal{F}_{1j(g)}(\pi) \cup \mathcal{F}_{2j(g)}(\pi)|}$$

where $\mathcal{F}_{ij}(\pi)$ are the modules in the respective networks based on the permuted data and $\mathcal{G}_0(\pi) = \bigcap_{i=1}^2 \bigcup_{j=1}^{J_i} \mathcal{F}_{ij}(\pi)$.

2.4 Scores

Each of the tests are based on connectivity scores $s^k_{gg'}$ between genes g and g' derived from the matrix of expression values X_k for the kth network; the superscript k will be ignored for the remainder of this section. There are many methods available for computing connectivity scores, and herein we describe the four methods implemented directly by the **dna** package. Let $\boldsymbol{x}_i$ denote the ith column of X, giving the expression values for the ith gene, and let $\tilde{X}_g$ denote the deflated design matrix with columns $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{g-1}, \boldsymbol{x}_{g+1}, \ldots, \boldsymbol{x}_p$. Each of the methods described in this section assumes that $\boldsymbol{x}_i$ is centered. Usually, $\boldsymbol{x}_i$ is rescaled so that the standard deviation of $\boldsymbol{x}_i$ equals 1 (i.e., $\|\boldsymbol{x_i}\|^2 = N - 1$), but there are options available when using the **dna** package for not automatically rescaling the expression values that are described in Section 4. After the statistical method is applied to obtain the connectivity scores, the connectivity scores are usually symmetrized ($s_{gg'}$ and $s_{gg'}$ are both re-assigned the value $(s_{gg'} + s_{g'g})/2$) and then often re-scaled so that all scores are divided by the largest score in magnitude. These options for postprocessing the scores using the **dna** package are also discussed in Section 4.

2.4.1 Partial least squares (PLS)

The PLS connectivity score for gene g based on gene g' can be computed using the algorithm described in [6]. Start by letting $X^{(1)} = \tilde{X}_g$. Then iteratively construct the ν latent PLS factors $\boldsymbol{t}_g^{(1)}, \ldots, \boldsymbol{t}_g^{(\nu)}$ where

$$\boldsymbol{t}_g^{(\ell)} = \sum_{j \neq g}^{p} c_{gj}^{(\ell)} X_k^{(\ell)}$$

where

$$\boldsymbol{c}_g^{(\ell)} = [c_{g1}^{(\ell)}, \ldots, c_{g,g-1}^{(\ell)}, c_{g,g+1}^{(\ell)}, \ldots, c_{gp}^{(\ell)}]^\top = \frac{{X^{(\ell)}}^\top \boldsymbol{x}_i}{\|{X^{(\ell)}}^\top \boldsymbol{x}_i\|},$$

$X^{(\ell)}$ is the deflated design matrix with columns $X_1^{(\ell)}, \ldots, X_{g-1}^{(\ell)}, X_{g+1}^{(\ell)}, \ldots X_p^{(\ell)}$, and

$$X^{(\ell+1)} = X^{(\ell)} - \boldsymbol{t}_g^{(\ell)} [\boldsymbol{t}_g^{(\ell)\top} \boldsymbol{t}_g^{(\ell)}]^{-1} {\boldsymbol{t}_g^{(\ell)}}^\top X^{(\ell)}, \ell < \nu.$$

Then

$$\hat{\beta}_{g\ell} = \frac{\boldsymbol{x}_g^\top \boldsymbol{t}_g^{(\ell)}}{\|\boldsymbol{t}_g^{(\ell)}\|^2}$$

is the coefficient obtained when regressing $\boldsymbol{x}_g$ on the latent factor $\boldsymbol{t}_g^{(\ell)}$. Finally, the PLS estimates of the coefficients for regressing $\boldsymbol{x}_g$ on the columns of $\tilde{X}_g$ are

$$s_{gg'} = \sum_{\ell=1}^{\nu} \hat{\beta}_{g\ell} c_{gg'}^{(\ell)}, g' = 1, \ldots, g-1, g+1, \ldots, p.$$

2.4.2 Principal components regression

Principal components regression is a similar method also based on derived inputs that can be used to compute connectivity scores. The ℓth principal component $\boldsymbol{v}_g^{(\ell)}$ is the eigenvector corresponding to the ℓth largest eigenvalue of $\tilde{X}_g^\top \tilde{X}_g$. Then $\hat{\boldsymbol{\beta}}_g = [\hat{\beta}_{g1}, \ldots, \hat{\beta}_{g\nu}]^\top$ where

$$\hat{\beta}_{g\ell} = \frac{\boldsymbol{x}_g^\top \boldsymbol{z}_g^{(\ell)}}{\|\boldsymbol{z}_g^{(\ell)}\|^2}$$

is the coefficient when regressing $\boldsymbol{x}_g$ on the derived input $\boldsymbol{z}_g^{(\ell)} = \tilde{X}_g \boldsymbol{v}_g^{(\ell)}$. Finally, the principal components regression estimates of the coefficients for regressing $\boldsymbol{x}_g$ on the columns of $\tilde{X}_g$ are

$$[s_{g1}, \ldots, s_{g,g-1}, s_{g,g+1}, \ldots, s_{gp}]^\top = V\hat{\boldsymbol{\beta}}_g$$

where V is the matrix with columns $\boldsymbol{v}_g^{(1)}, \ldots, \boldsymbol{v}_g^{(\nu)}$.

2.4.3 Ridge regression

The ridge regression connectivity scores are the coefficients when regressing $\boldsymbol{x}_g$ on the columns of $\tilde{X}_g$ with penalty parameter is the $p-1$ dimensional vector $\boldsymbol{\beta}$ which minimizes

$$\|\boldsymbol{x}_g - \tilde{X}_g\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2$$

and can be computed by

$$[s_{g1}, \ldots, s_{g,g-1}, s_{g,g+1}, \ldots, s_{gp}]^\top = (\tilde{X}_g^\top \tilde{X}_g + \lambda I)^{-1}\tilde{X}_g^\top \boldsymbol{x}_g.$$

2.4.4 Correlation

The Pearson correlation coefficient

$$s_{gg'} = \frac{\boldsymbol{x}_g^\top \boldsymbol{x}_{g'}}{\|\boldsymbol{x}_g\|\|\boldsymbol{x}_{g'}\|}$$

is a simple and quick method of measuring the association between the expression values for genes g and g' where $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^\top \boldsymbol{x}}$ is the Euclidean norm.

3 Usage

In this section, we illustrate the functions provided by the `dna` package which implement the test procedures described in Sections 2.1, 2.2, and 2.3 on real genomic data from the liver tissue of female mice that was analyzed previously in Ghazalpour *et* al. (2006), Fuller *et* al. (2007), and Gill, Datta, and Datta (2010). The matrix `HeavyMice` includes expression values of 314 genes for 50 mice with weights greater than 40.5 grams, while the matrix `LeanMice` includes the expression values of the same 314 genes for 50 mice with weights below 36.9 grams. The library and data can be loaded into `R` as follows.

```
> library("dna")
> data("HeavyMice")
> data("LeanMice")
```

First, we perform the tests for individual genes described in Section 2.1 to determine whether there is a difference between the connectivity scores for each respective individual gene in the Heavy and Lean networks. The `dna` package includes a function `test.individual.genes` which performs the test for all genes with options for various methods of computing the connectivity scores and distance function as well as several preprocessing and postprocessing options. The arguments for the function are displayed below.

```
R> args(test.individual.genes)
function (X1, X2, scores = "PLS", distance = "abs", num.permutations = 1000,
    check.networks = TRUE, ...)
```

The mandatory inputs `X1` and `X2` are two matrices containing the expression levels for the networks with experimental units in the rows and genes in the columns. The user does not necessarily need to place the genes in the same order in the columns of each matrix since the function automatically preprocesses the networks as long as the optional argument `check.networks` is set to `TRUE`. This and other technical details concerning the functions are discussed in Section 4.

The type of connectivity score to be used is specified by the argument `scores`. Either a built-in method (`"PLS"`, `"PC"`, `"RR"`, or `"cor"`) or a properly-defined function specified by the user is accepted. Any arguments accepted by the method or function for the connectivity score can be passed as additional arguments to `test.individual.genes`. The distance function D to be used is specified by the optional argument `distance`, and it also can either be one of the built-in distance functions `"abs"` or `"sqr"`) or a user-defined function; `distance="abs"` indicates that the L_1-distance $D(s_1, s_2) = |s_1 - s_2|$ should be used and `distance="sqr"` indicates that the L_2-distance $D(s_1, s_2) = (s_1 - s_2)^2$ should be used. Examples of user-defined scores and distance functions will be discussed in Section refsec:additional. The number of permutations for the test can be controlled by the optional argument `num.permutations`; if it is not specified, then it is set to 1000 by default.

To illustrate this function in use, we perform the test for differential connectivity for individual genes based on PLS connectivity scores (rescaled) with L_1 distances using the following command.

```
R> tig.results=test.individual.genes(LeanMice,HeavyMice,scores="PLS",
+ distance="abs",rescale.scores=TRUE,num.permutations=1000)
```

The function outputs an object of class `resultsIndTest`. A summary method is available for the object which outputs the number of genes which have P-values less than various levels.

```
R> summary(tig.results)
Tests for differential connectivity of individual genes
```

```
9 genes are significant at level 0.001
20 genes are significant at level 0.005
29 genes are significant at level 0.01
57 genes are significant at level 0.05
```

The full results can be extracted and inserted into a data frame with columns containing the gene names, values of the test statistic, and P-values with the method `get.results`.

```
R> get.results(tig.results)
                        d p.value
Anxa2          0.11804280   0.000
F7             0.12197477   0.000
Anxa5          0.11906887   0.000
Map4k4         0.14530306   0.000
Kng2           0.16757005   0.000
Scnn1a         0.14848855   0.000
Slc43a1        0.16212205   0.000
Apom           0.18553058   0.000
Spp1           0.23177885   0.000
Slc22a7        0.15434813   0.001
.
.
.
```

Next, we perform the test for a class of genes described in Section 2.2. Most of the arguments for the function `test.class.genes` are similar to those for `test.individual.genes`. There is one additional optional argument `genelist` which specifies the class of genes to be considered; `genelist` can either be a vector which contains the names of the genes or it can be a numeric vector with the indices of the genes. Usually, the user will want to choose a subset of genes to be tested, but if `genelist` is not specified by the user, then the default choice for the class is to use all genes.

The following example demonstrates the use of this function to test a class of four genes *Anxa2*, *Anxa5*, *F7*, and *Proz*. This set of genes was identified to be related to blood coagulation (Gill, Datta, and Datta, 2010). The following code uses PLS scores with L_1 distances.

```
R> ourgenelist=c("Anxa2","Anxa5","F7","Proz")
R> tcg.results=test.class.genes(LeanMice,HeavyMice,genelist=ourgenelist,
+ scores="PLS",distance="abs",rescale.scores=TRUE,num.permutations=1000)
R> tcg.results
Tests for differential connectivity of a class of genes

Class of genes:
Proz,Anxa2,F7,Anxa5
```

```
Test statistic: delta= 0.1511344
P-value= 0
```

The function outputs an object of class `resultsClassTest`. There is a `get.results` method that accepts this object as an argument and outputs a list including the P-value, test statistic, and class of genes.

```
R> get.results(tcg.results)
$p.value
[1] 0

$delta
[1] 0.1511344

$class.genes
[1] "Proz"  "Anxa2" "F7"    "Anxa5"
```

Finally, we demonstrate the function `test.modular.structure` which implements the test for overall modular structure described in Section 2.3. The arguments for the function are displayed below.

```
R> args(test.modular.structure)
function (X1, X2, scores = "PLS", min.module.size = 5, epsilon = 0.5,
    num.permutations = 1000, check.networks = TRUE, ...)
```

The following example shows the function being applied to the mice networks with (rescaled) PLS connectivity scores, minimum size parameter $m = 5$, and threshold connectivity parameter $\epsilon = 0.5$.

```
R> tms.results=test.modular.structure(LeanMice,HeavyMice,min.module.size=5,
+ epsilon=.5,scores="PLS",rescale.scores=TRUE,num.permutations=1000)
```

The function outputs an object of class `resultsModTest`. A summary method is available which accepts objects of this class and outputs information about each network, the test statistic, and the P-value for the test as shown below.

```
R> summary(tms.results)
Tests for differential modular structure in two networks of genes

Network 1:
Class: modules
10 genes in Module 1
44 genes in Module 2
6 genes in Module 3
5 genes in Module 4
10 genes in Module 5

Network 2:
```

```
Class: modules
300 genes in Module 1

Test statistic: N= 0.9755889
P-value= 0.033
```

There is also a method `get.results` (not shown here) which extracts the networks, test statistic, and P-value.

4 Additional features and details

The functions available in the `dna` package for performing the tests of significance have been described in the previous section. This section discusses some additional options available in performing these tests as well as additional functions and methods provided with the package which may be useful to users.

4.1 Preprocessing tools

The `dna` package uses custom S4 classes and methods to preprocess the pair of networks specified by a user. Using the package does not require users to work directly with these classes and methods since the functions which perform the statistical tests will create the classes and employ their methods when needed. However, it is important that users understand the implications of how they input their networks of genes.

It is assumed that each network specified by the user is a matrix with rows which represent the experimental units and columns which represent the genes to be considered. To illustrate the objects, we create the following small example.

```
R> X1=cbind(Gz=c(.4,.5,-.8),Gy=c(.8,-.8,-.3),
+ Gb=c(1.1,.3,.8),Ga=c(1.5,-.6,-1.5))
R> X2=cbind(Gc=c(-1.6,1.8,-.5,.6),Ga=c(.6,-.2,.8,2.2),Gb=c(2,1.6,.3,.5))
R> networks=new("pairOfNetworks",network1=X1,network2=X2)
R> networks
Class: pairOfNetworks
Network 1: 3 subjects and 4 genes.
Network 2: 4 subjects and 3 genes.
The networks have 2 genes in common.
```

If labels for the columns are provided as they are above, then the class can handle networks with different numbers of columns. When networks are used in the statistical tests shown in Section 3, a method `get.common.networks` is invoked to extract the genes common to both networks. Thus, in this example, only the two genes Gb and Ga will be used.

```
R> get.common.networks(networks)
$network1
      Gb   Ga
```

```
[1,] 1.1  1.5
[2,] 0.3 -0.6
[3,] 0.8 -1.5

$network2
      Gb   Ga
[1,] 2.0  0.6
[2,] 1.6 -0.2
[3,] 0.3  0.8
[4,] 0.5  2.2
```

When no gene names are supplied, the method automatically assumes that the genes are specified in the same order in each network. In this case, it gives the columns the generic names `Gene 1`, `Gene 2`, ... that will be used by other functions.

4.2 Connectivity scores

The tests discussed in Section 3 are based on connectivity scores discussed in Section 2.4. While some users may only wish to directly use the statistical tests, others may want to access the connectivity scores and possibly use them for other purposes. There are built-in functions available for each of the methods described in Section 2.4 which return a matrix of connectivity scores.

The default connectivity scores for the test procedures are PLS scores. To directly compute these scores, the function `PLSnet` is available. By default, this function uses three PLS components, but this can be changed by specifying a different value for the `ncom` argument. No intercept is included in the computation of the PLS regression, so each of the columns are automatically centered. There is an optional argument `rescale.data` which indicates whether the values for each gene should be rescaled prior to computing the scores; typically this is done for PLS regression so the default is `rescale.data=TRUE`. A few options for transforming the PLS scores are also available after PLS regression has been performed; `symmetrize.scores` indicates whether scores should be symmetrized and `rescale.scores` indicates whether scores should be rescaled by dividing each score by the largest score in magnitude. By default, the function `PLSnet` sets `symmetrize.scores=TRUE` and `rescale.scores=FALSE`; if both are set to true, then symmetrization occurs before rescaling. The following code illustrates using this function with two PLS components and all optional arguments set to `TRUE` on the four genes in the network `X1` defined in Section 4.1.

```
R> PLSnet(X1,ncom=2,rescale.data=TRUE,symmetrize.scores=TRUE,
+ rescale.scores=TRUE)
           Gz         Gy         Gb        Ga
Gz  1.0000000 -0.1609580 -0.8778705 1.0000000
Gy -0.1609580  1.0000000  0.9692391 0.6735266
Gb -0.8778705  0.9692391  1.0000000 0.3125738
Ga  1.0000000  0.6735266  0.3125738 1.0000000
```

Each of the other methods of computing connectivity scores that are described in Section 2.4 are also implemented by the `dna` package. Connectivity scores based on principal components regression can be computed using the function `PCnet`. The arguments and default values for `PCnet` are the same as those for `PLSnet`. The function `RRnet` computes the scores based on ridge regression; `rescale.data`, `symmetrize.scores`, and `rescale.scores` are also optional arguments of `RRnet` with the same defaults, but it has a different complexity parameter `lambda` set to 1 by default. Finally, the function `cornet` computes the connectivity scores based on correlation. No centering or scaling is necessary since the correlation coefficient for two variables is not affected by linear transformations of the variables. Also, correlation is symmetric is no symmetrization option is necessary. The only optional argument available for `cornet` is `rescale.scores` and it is `FALSE` by default.

4.3 User-defined scores and distances

The `dna` package also provides a flexible way of incorporating other methods to create new user-defined connectivity scores through the function `gennet`. The arguments for `gennet` are listed below.

```
R> args(gennet)
function (data, f, recenter.data = FALSE, rescale.data = FALSE,
    symmetrize.scores = FALSE, rescale.scores = FALSE, ...)
```

The mandatory argument `f` is a regression method for modeling each gene based on the remaining genes, and it is used to compute each row of the matrix of connectivity scores. For example, user-defined connectivity scores based on the LASSO implemented by the **R** package `lars` (Hastie and Efron, 2012) can be computed for network `X1` from Section 4.1 as follows.

```
R> require(lars)
R> our.LASSO=function(X,y,s=s,mode=mode){
+  coef(lars(X,y,type="lasso",normalize=FALSE,intercept=FALSE),
+  s=s,mode=mode)
+ }
R> gennet(X1,our.LASSO,recenter.data=TRUE,rescale.data=TRUE,
+ symmetrize.scores=TRUE,rescale.scores=TRUE,s=1,mode="lambda")
          Gz        Gy Gb        Ga
Gz 1.0000000 0.0000000  0 0.3953771
Gy 0.0000000 1.0000000  1 0.5017072
Gb 0.0000000 1.0000000  1 0.0000000
Ga 0.3953771 0.5017072  0 1.0000000
```

The argument `distance` for the tests described in Section 3 can also be customized by the user. If `distance` is a user-defined function, then the two arguments to the function should be corresponding scores or matrices of scores from the two networks. For instance, the piecewise distance function $D(s_1, s_2) = \min\{|s_1 - s_2|, 1\}$ can be created using the command

```
R> our.dist=function(score1,score2){pmin(abs(score1-score2),1)}
```

Then, suppose we want to perform a test for the class of genes discussed in Section 3 with LASSO connectivity scores and the user-defined distance function. The following code shows how this can be accomplished.

```
R> our.LASSOnet=function(data,s=1,mode="lambda"){
+   gennet(data,our.LASSO,recenter.data=TRUE,rescale.data=TRUE,
+   symmetrize.scores=TRUE,rescale.scores=TRUE,s=s,mode=mode)
+  }
R> ourgenelist=c("Anxa2","Anxa5","F7","Proz")
R> results=test.class.genes(LeanMice,HeavyMice,genelist=ourgenelist,
+ scores=our.LASSOnet,distance=our.dist,num.permutations=1000)
R> results
Tests for differential connectivity of a class of genes

Class of genes:
Proz,Anxa2,F7,Anxa5

Test statistic: delta= 0.01925669
P-value= 0.353
```

4.4 Modules

The `dna` package provides tools for working with modules given a matrix of connectivity scores `s` and user-supplied parameters `m` and `epsilon`. To illustrate these tools, consider the following matrix of connectivity scores for a network.

```
> set.seed(26)
> s=matrix(runif(100,-1,1),10,10);diag(s)=1;s=round((s+t(s))/2,1)
> s

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]  1.0 -0.2  0.8  0.6 -0.4 -0.3  0.0  0.6 -0.5   0.1
 [2,] -0.2  1.0  0.2 -0.5  0.7 -0.1 -0.4  0.0  0.3  -0.2
 [3,]  0.8  0.2  1.0  0.0 -0.1 -0.1  0.8  0.3  0.0  -0.3
 [4,]  0.6 -0.5  0.0  1.0  0.0  0.2 -0.3 -0.7  0.2   0.8
 [5,] -0.4  0.7 -0.1  0.0  1.0 -0.2  0.1 -0.1  0.5  -0.3
 [6,] -0.3 -0.1 -0.1  0.2 -0.2  1.0 -0.6  0.4  0.2   0.4
 [7,]  0.0 -0.4  0.8 -0.3  0.1 -0.6  1.0  0.6  0.0  -0.1
 [8,]  0.6  0.0  0.3 -0.7 -0.1  0.4  0.6  1.0  0.7  -0.2
 [9,] -0.5  0.3  0.0  0.2  0.5  0.2  0.0  0.7  1.0   0.4
[10,]  0.1 -0.2 -0.3  0.8 -0.3  0.4 -0.1 -0.2  0.4   1.0
```

Suppose we define our modules using parameter values $m = 3$ and $\epsilon = .7$. The modules are obtained using the function `network.modules` with the appropriate arguments.

```
R> the.modules=network.modules(s,m=3,epsilon=.7)
R> the.modules
Module 1:
Gene 1,Gene 3,Gene 7

Module 2:
Gene 4,Gene 8,Gene 9,Gene 10
```

Note that, although the score for genes 2 and 5 is 0.7, they do not form a module because a module requires at least m genes.

The function `network.modules` returns an object of class "modules". In addition to the default output, there are other ways to view and access the modules. For large networks, it may be helpful to first view a summary of the modular structure. This can be accomplished using the `summary` method for this class.

```
R> summary(the.modules)
Class: modules
3 genes in Module 1
4 genes in Module 2
```

In other cases, users may wish to extract the module number for each gene using the method `get.modules`.

```
R> get.modules(the.modules)
 [1] 1 0 1 2 0 0 1 2 2 2
```

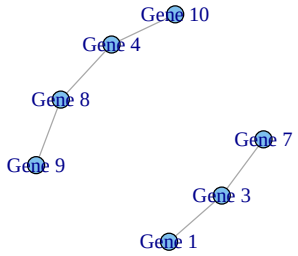
Here, genes which do not belong to any module are listed in module 0.

By default, `network.modules` does not display a graph of the networks, but there is an optional argument `plot` which can be set to `TRUE` to output a graph.

```
> network.modules(s,m=3,epsilon=.7,plot=TRUE,interactive=FALSE)

Module 1:
Gene 1,Gene 3,Gene 7

Module 2:
Gene 4,Gene 8,Gene 9,Gene 10
```



The figure above ia created using the `plot` function from the `igraph` package (Csardi and Nepusz, 2006). If the optional argument is changed to `interactive=TRUE`, then an interactive graphing device is opened within R using the `tkplot` function from the `igraph` package; the function gives the user many options for modifying the graph both before and after the function call. Additional arguments can be passed to `tkplot` through `network.modules`.

References

[1] G. Csardi and T. Nepusz. (2005). The **igraph** software package for complex network research. *I*nterjournal, Complex Systems, 1695.

[2] T. Fuller, A. Ghazalpour, J. Aten, T. Drake, A. Lusis, and S. Horvath. (2007). Weighted gene coexpression network analysis strategies applied to mouse weight. *M*ammalian Genome, **18**, 463–472.

[3] A. Ghazalpour, S. Doss, B. Zhang, S. Wang, C. Plaisier, R. Castellanos, A. Brozell, E.E. Schadt, T.A. Drake, A. Lusis, and S. Horvath. (2006). Integrating genetic and network analysis to characterize genes related to mouse weight. *P*LoS Genet, **2**(8), e130.

[4] R. Gill, S. Datta, and S. Datta. (2010). A statistical framework for differential network analysis from microarray data. *B*MC Bioinformatics, **11**, 95.

[5] T. Hastie and B. Efron. (2012). lars: Least Angle Regression, Lasso and Forward Stagewise. R package version 1.1. `http://CRAN.R-project.org/package=lars`

[6] V. Pihur, S. Datta, and S. Datta. (2008). Reconstruction of genetic association networks from microarray data: a partial least squares approach. *B*ioinformatics, **24**(4), 561–568.

[7] R Development Core Team. (2014). R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org/