Package ‘iSubGen’

October 13, 2022

Title Integrative Subtype Generation

Version 1.0.1

Date 2021-04-15

Author Natalie Fox

Maintainer Paul C Boutros <pboutros@mednet.ucla.edu>

Description
Multi-data type subtyping, which is data type agnostic and accepts missing data. Subtyping is performed using intermediary assessments created with autoencoders and similarity calculations.

Depends R (>= 3.2.3)

Imports ConsensusClusterPlus, cluster (>= 1.14.4), keras, tensorflow, philentropy

Suggests knitr, rmarkdown

VignetteBuilder knitr

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2021-04-22 14:50:10 UTC

R topics documented:

apply.scaling *Apply scaling factors*

Description

Apply scaling factors prior to autoencoder

Usage

```
apply.scaling(data.matrices, scaling.factors);
```

Arguments

`data.matrices`	list, where each element is a matrix. The list has one matrix for each data type to be scaled
`scaling.factors`	list with two elements named: \"center\" and \"scale\", and each element is a named numerical vector or a list of named numerical vectors. If scaling.factors$center or scaling.factors$scale are a list then each element needs to correspond to a one of the data matrices. Finally, the named numerical vectors should match the row and rownames from the corresponding data matrix.

Details

The names for the data matrices and the center and scale lists all must match.

Value

A list of matrices of the same format as the data.matrices

Author(s)

Natalie Fox

Examples

```
# Load molecular profiles for three data types and calculate scaling for each
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');
molecular.data <- list();
scaling.factors <- list();
for(i in c('cna','snv','methy')) {
  # Load molecular profiles from example files saved
  # in the package as <data type>_profiles.txt
  molecular.data[[i]] <- load.molecular.aberration.data(
    paste0(example.molecular.data.dir,i,'_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    );
  scaling.factors[[i]] <- list();

  scaling.factors[[i]]$center <- apply(molecular.data[[i]], 1, mean);
  scaling.factors[[i]]$scale <- apply(molecular.data[[i]], 1, sd);
  }

# Example 1: Transform the molecular profiles by the scaling factors
scaled.molecular.data <- apply.scaling(molecular.data, scaling.factors);

# Example 2: Transform one of the data types based on the scaling factors
scaled.molecular.data2 <- apply.scaling(
  molecular.data[[1]],
  scaling.factors[[1]]
  );
```

`calculate.cis.matrix` *Calculate consensus integrative correlation matrix*

Description

Calculate consensus pairwise correlations between patient distances

Usage

```
calculate.cis.matrix(data.types, data.matrices, dist.metrics,
correlation.method = "spearman", filter.to.common.patients = FALSE,
patients.to.return = NULL, patients.for.correlations = NULL,
patient.proportion = 0.8, feature.proportion = 1, num.iterations = 10,
print.intermediary.similarity.matrices.to.file = TRUE, print.dir = '.',
patient.proportion.seeds = seq(1,num.iterations),
feature.proportion.seeds = seq(1,num.iterations))
```

Arguments

`data.types`	vector of the IDs for the different data types that are the names of the lists for the data.matrices and dist.metrics
`data.matrices`	list of the matrices with features (rows) by patients (columns)
`dist.metrics`	list of the distance metrics for comparing patient profiles. ex. euclidean. Options are from philentropy::distance
`correlation.method`	specifies the type of correlation for similarity comparison. Options are pearson, spearman or kendall.
`filter.to.common.patients`	logical, where TRUE indicates to filter out patients that don't have all data types
`patients.to.return`	vector of patients to calculate CIS for. For example, this is the testing cohort patients when calculating CIS for the testing cohort using the training cohort patients. If NULL all patients/columns will be used.

`patients.for.correlations`	vector of patients to use to calculate the similarities. For example, this would be the training cohort patients when calculating CIS for the testing cohort. If NULL all patients/columns will be used.
`patient.proportion`	proportion of patients.for.correlations to sample for each iteration (sampled without replacement).
`feature.proportion`	proportion of the features to sample for each iteration (sampled without replacement).
`num.iterations`	number of iterations to take the median from
`print.intermediary.similarity.matrices.to.file`	logical, where TRUE indicates that created intermediary integrative similarity matrix from each iteration should be printed to file
`print.dir`	directory for where to print the intermediary similarity matrices to file
`patient.proportion.seeds`	vector of scalars of the length num.iterations specifying the seeds used for random sampling for selecting the patient subsets at each iteration
`feature.proportion.seeds`	vector of scalars of the length num.iterations specifying the seeds used for random sampling for selecting the feature subsets at each iteration

Value

CIS matrix where rows are patients and columns are pairs of data types

Author(s)

Natalie Fox

Examples

```
# Load molecular profiles for three data types from example files saved
# in the package as <data type>_profiles.txt
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');
molecular.data <- list();
for(i in c('cna','snv','methy')) {
  molecular.data[[i]] <- load.molecular.aberration.data(
    paste0(example.molecular.data.dir,i,'_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    );
  }

# Example 1: calculate the consensus integrative similarity (CIS) matrix
corr.matrix <- calculate.cis.matrix(
  data.types = names(molecular.data),
  data.matrices = molecular.data,
  dist.metrics = list(
    cna = 'euclidean',
    snv = 'euclidean',
    methy = 'euclidean'
    ),
  print.intermediary.similarity.matrices.to.file = FALSE
  );

# Example 2: calculate the CIS matrix for patients EP001 through EP009 in relation
# to patients EP010 through EP030 meaning the profile of EP001 is correlated to
# the profiles of EP010 through EP030 so when assessing new patients, they can be
# compared to the training profiles
corr.matrix2 <- calculate.cis.matrix(
  data.types = names(molecular.data),
  data.matrices = molecular.data,
  dist.metrics = list(
    cna = 'euclidean',
    snv = 'euclidean',
    methy = 'euclidean'
    ),
  patients.to.return = paste0('EP00',1:9),
  patients.for.correlations = paste0('EP0',10:30),
  print.intermediary.similarity.matrices.to.file = FALSE
  );

# Example 3: Adjusting the proportion of the features that will be used to correlate
# the patient profiles
corr.matrix3 <- calculate.cis.matrix(
  data.types = names(molecular.data),
  data.matrices = molecular.data,
  dist.metrics = list(
    cna = 'euclidean',
    snv = 'euclidean',
    methy = 'euclidean'
    ),
  patients.to.return = paste0('EP00',1:9),
  patients.for.correlations = paste0('EP0',10:30),
  feature.proportion = 0.6,
  print.intermediary.similarity.matrices.to.file = FALSE
  );
```

calculate.integrative.similarity.matrix

Calculate integrative similarity matrix

Description

Calculate pairwise correlations between patient distances

Usage

```
calculate.integrative.similarity.matrix(data.types, data.matrices, dist.metrics,
correlation.method = "spearman", filter.to.common.patients = FALSE,
patients.to.return = NULL, patients.for.correlations = NULL)
```

Arguments

`data.types`	vector, where each element is a data type ID matching the names in data.matrices and dist.metrics
`data.matrices`	list, where each element is a matrix with features as rows and patients as columns
`dist.metrics`	list, where each element is the distance metric to use for comparing patient profiles. ex. euclidean. Options are from philentropy::distance
`correlation.method`	specifies the type of correlation. Options are pearson, spearman or kendall.
`filter.to.common.patients`	logical, where TRUE indicates to filter out patients that don't have all data types
`patients.to.return`	vector, where each element a patient ID specifying the patients to calculate integrative similarity for. For example, this is the testing cohort patients when calculating integrative similarity for the testing cohort using the training cohort patients. If NULL all patients/columns will be used.
`patients.for.correlations`	vector, where each element a patient ID specifying the patients to use to calculate the similarities. For example, this would be the training cohort patients when calculating integrative similarity for the testing cohort. If NULL all patients/columns will be used.

Value

matrix where rows are patients and columns are pairs of data types

Author(s)

Natalie Fox

Examples

```
# Load molecular profiles for three data types from example files saved
# in the package as <data type>_profiles.txt
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');
molecular.data <- list();
for(i in c('cna','snv','methy')) {
  molecular.data[[i]] <- load.molecular.aberration.data(
    paste0(example.molecular.data.dir,i,'_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    );
  }
```

```
# Example 1: calculate integrative similarity between pairs of CNA, coding SNVs, methylation data
corr.matrix <- calculate.integrative.similarity.matrix(
  data.types = names(molecular.data),
  data.matrices = molecular.data,
  dist.metrics = list(
    cna = 'euclidean',
    snv = 'euclidean',
    methy = 'euclidean'
    )
  );

# Example 2: calculate the integrative similarity for patients EP001 through EP009
# in relation to patients EP010 through EP030 meaning the profile of EP001 is
# correlated to the profiles of EP010 through EP030 so when assessing new patients,
# they can be compared to the training profiles
corr.matrix2 <- calculate.integrative.similarity.matrix(
  data.types = names(molecular.data),
  data.matrices = molecular.data,
  dist.metrics = list(
    cna = 'euclidean',
    snv = 'euclidean',
    methy = 'euclidean'
    ),
  patients.to.return = paste0('EP00',1:9),
  patients.for.correlations = paste0('EP0',10:30)
  );

# Example 3: Calculate integrative similarity between CNA and methylation data
corr.matrix3 <- calculate.integrative.similarity.matrix(
  data.types=names(molecular.data)[c(1,3)],
  data.matrices=molecular.data[c(1,3)],
  dist.metrics=list(
    cna='euclidean',
    snv='euclidean',
    methy='euclidean'
    )[c(1,3)],
  patients.to.return=paste0('EP00',1:9),
  patients.for.correlations=paste0('EP0',10:30)
  );
```

`calculate.scaling` *Calculate scaling factors*

Description

Calculate scaling factors

Usage

```
calculate.scaling(data.matrices);
```

Arguments

`data.matrices`	list, where each element is a matrix. The list has one matrix for each data type to be scaled

Details

The names for the data matrices and the center and scale lists all must match.

Value

a list with two elements named: \"center\" and \"scale\", and each of these element is a named numerical vector or a list of named numerical vectors. If scaling.factors$center or scaling.factors$scale are a list then each element will correspond to a one of the data matrices. Finally, the named numerical vectors will match the row and rownames from the data matrices.

Author(s)

Natalie Fox

Examples

```
# Load molecular profiles for three data types from example files saved
# in the package as <data type>_profiles.txt
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');
molecular.data <- list();
for(i in c('cna','snv','methy')) {
  molecular.data[[i]] <- load.molecular.aberration.data(
    paste0(example.molecular.data.dir,i,'_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    );
  }

# Example 1: Calculate scaling factors for all three data types
scaling.factors <- calculate.scaling(molecular.data);

# Example 2: Calculate scaling factors for only the methylation data
scaling.factors2 <- calculate.scaling(molecular.data[['methy']]);
```

`cluster.patients` *Clustering to find patient subtypes*

Description

A wrapper function for using consensus clustering to subtype patients

Usage

```
cluster.patients(data.matrix, distance.metric, parent.output.dir,
new.result.dir, subtype.table.file = NULL, max.num.subtypes = 12,
clustering.reps = 1000, proportion.features = 0.8, proportion.patients = 0.8,
verbose = FALSE, consensus.cluster.write.table = TRUE);
```

Arguments

`data.matrix`	matrix with patients as rows and features as columns
`distance.metric`	distance metric for comparing patient profiles. ex. euclidean
`parent.output.dir`	directory where the consensus clustering function will create a directory of results
`new.result.dir`	directory name for consensus clustering results
`subtype.table.file`	filename for subtype assignment table for different number of clusters
`max.num.subtypes`	maximum number of clusters to separate patients into
`clustering.reps`	number of subsamples for consensus clustering function
`proportion.features`	proportion of features to sample for each clustering iteration
`proportion.patients`	proportion of patients to sample for each clustering iteration
`verbose`	logical, where TRUE indicates to print messages to the screen to indicate progress
`consensus.cluster.write.table`	logical, where TRUE indicates for the ConsensusClusterPlus function to writeTable

Value

`consensus_cluster_result`	consensus clustering function return value
`subtype_table`	the table written to subtype.table.file

Author(s)

Natalie Fox

Examples

```
## Not run:

# For this example instead of clustering CIS and IRF matrices,
# create a data matrix to see how the function works without
# running through the whole iSubGen process.
# This example is created with to have 4 distinct clusters
```

```
set.seed(5);
ex.matrix <- matrix(
  c(
    sample(c(0,1), 30, replace = TRUE), rep(1,75), rep(0,25),
    sample(c(0,1), 30, replace = TRUE), rep(1,75), rep(0,25),
    sample(c(0,1), 30, replace = TRUE), rep(1,75), rep(0,25),
    sample(c(0,1), 30, replace = TRUE), rep(1,100),
    sample(c(0,1), 30, replace = TRUE), rep(1,100),
    sample(c(0,1), 30, replace = TRUE), rep(1,100),
    sample(c(0,1), 30, replace = TRUE), rep(0,100),
    sample(c(0,1), 30, replace = TRUE), rep(0,100),
    sample(c(0,1), 30, replace = TRUE), rep(0,100),
    sample(c(0,1), 30, replace = TRUE), rep(0,75), rep(1,25),
    sample(c(0,1), 30, replace = TRUE), rep(0,75), rep(1,25),
    sample(c(0,1), 30, replace = TRUE), rep(0,75), rep(1,25)
    ),
  nrow=130);
rownames(ex.matrix) <- paste0('gene',1:130);
colnames(ex.matrix) <- paste0('patient',LETTERS[1:12]);

# Use Consensus clustering to subtype the patient profiles
subtyping.results <- cluster.patients(
  data.matrix = ex.matrix,
  distance.metric = 'euclidean',
  parent.output.dir = './',
  new.result.dir = 'example_subtyping',
  max.num.subtypes = 6,
  clustering.reps = 50,
  consensus.cluster.write.table = FALSE
  );

## End(Not run)
```

combine.integrative.features

Combine iSubGen integrative features

Description

Combine a independent reduced features matrix (ex. from autoencoders) and pairwise integrative similarity matrices into one integrative feature matrix.

Usage

```
combine.integrative.features(irf.matrix, cis.matrix,
irf.rescale.recenter = NA, cis.rescale.recenter = NA,
irf.rescale.denominator = NA, cis.rescale.denominator = NA,
irf.weights = rep(1, ncol(irf.matrix)),
cis.weights = rep(1, ncol(cis.matrix)))
```

Arguments

`irf.matrix`	matrix of independent reduced features with patients as rows and features as columns
`cis.matrix`	matrix of consensus integrative similarity or integrative similarity features with patients as rows and features as columns
`irf.rescale.recenter`	either NA, "mean", a single number or a vector of numbers of length equal to the number of columns of irf
`cis.rescale.recenter`	either NA, "mean", a single number or a vector of numbers of length equal to the number of columns of cis
`irf.rescale.denominator`	either NA, "sd", a single number or a vector of numbers of length equal to the number of columns of irf
`cis.rescale.denominator`	either NA, "sd", a single number or a vector of numbers of length equal to the number of columns of cis
`irf.weights`	single number or vector of numbers of length equal to the number of columns of irf
`cis.weights`	single number or vector of numbers of length equal to the number of columns of cis

Details

The recenter values determine the how column centering is performed. If NA, no recentering is done. If the values equal "mean", then the mean of each column will be used. Otherwise, the numeric values specified will be used. The denominator values determine how column scaling is performed. If NA, no recentering is done. If the denominator values equal "sd", then the standard deviation of each column will be used. Otherwise, the numeric values specified will be used. The values used are returned by the function along with the compressed feature matrix to be recorded for reproducibility purposes.

Value

`integrative.feature.matrix`	a matrix of compressed features with patients as rows and features as columns
`irf.rescale.recenter`	a numeric vector with length equal to the number of columns of irf
`cis.rescale.recenter`	a numeric vector with length equal to the number of columns of cis
`irf.rescale.denominator`	a numeric vector with length equal to the number of columns of irf
`cis.rescale.denominator`	a numeric vector with length equal to the number of columns of cis
`irf.weights`	a numeric vector with length equal to the number of columns of irf
`cis.weights`	a numeric vector with length equal to the number of columns of cis

Author(s)

Natalie Fox

Examples

```
# Create matrices for combining
irf.matrix <- matrix(runif(25*4), ncol = 4);
rownames(irf.matrix) <- c(paste0('EP00',1:9), paste0('EP0',10:25));
cis.matrix <- matrix(runif(25*6), ncol=6);
rownames(cis.matrix) <- c(paste0('EP00',1:9), paste0('EP0',10:25));

# Example 1: Join the matrices without any weighting adjustments
isubgen.feature.matrix <- combine.integrative.features(
  irf.matrix,
  cis.matrix
  )$integrative.feature.matrix;

# Example 2: Combine matrices after scaling each column by subtracting the mean
# and dividing by the standard devation of the column
isubgen.feature.matrix.rescaled.result <- combine.integrative.features(
  irf.matrix,
  cis.matrix,
  irf.rescale.recenter = 'mean',
  cis.rescale.recenter = 'mean',
  irf.rescale.denominator = 'sd',
  cis.rescale.denominator = 'sd'
  );
isubgen.feature.matrix.2 <- isubgen.feature.matrix.rescaled.result$integrative.feature.matrix;

# Example 3: Combine matrices
isubgen.feature.matrix.reweighted.result <- combine.integrative.features(
  irf.matrix,
  cis.matrix,
  irf.weights = 1/4,
  cis.weights = 1/6
  );
isubgen.feature.matrix.3 <- isubgen.feature.matrix.reweighted.result$integrative.feature.matrix;
```

`create.autoencoder` *Create an autoencoder for dimensionality reduction*

Description

Create an autoencoder for dimensionality reduction using keras and tensorflow packages

Usage

```
create.autoencoder(data.type, data.matrix, encoder.layers.node.nums = c(15,2),
autoencoder.activation = 'tanh', optimization.loss.function = 'mean_squared_error',
model.file.output.dir = '.')
```

Arguments

`data.type`	data type ID. The ID will be used for naming the output file
`data.matrix`	matrix with data features as rows and patients as columns
`encoder.layers.node.nums`	vector with the number of nodes for each layer when the reducing the feature dimensions within the autoencoder. The autoencoder will be made symmetrically so the number of nodes in each layer will be used in reverse, not repeating the last layer to re encode the features in the autoencoder
`autoencoder.activation`	activation function to use in the autoencoder
`optimization.loss.function`	loss function used for optimization while fitting the autoencoder
`model.file.output.dir`	file location for the autoencoder file

Value

`autoencoder`	the autoencoder created by the keras package
`autoencoder.file`	the hdf5 file that the model was saved in and can be loaded from

Author(s)

Natalie Fox

Examples

```
## Not run:

example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');

ae.result <- create.autoencoder(
  data.type = 'cna',
  data.matrix = load.molecular.aberration.data(
    paste0(example.molecular.data.dir,'cna_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    ),
  encoder.layers.node.nums = c(15,5,2)
  );

## End(Not run)
```

create.autoencoder.irf.matrix
Create matrix of independent reduced features

Description

Create matrix of independent reduced features using autoencoders

Usage

```
create.autoencoder.irf.matrix(data.types, data.matrices,
autoencoders, filter.to.common.patients = FALSE,
patients.to.return = NULL)
```

Arguments

`data.types`	vector, where each element is a data type ID matching the names in data.matrices and dist.metrics
`data.matrices`	list, where each element is a matrix with features as rows and patients as columns
`autoencoders`	list, where each element is an autoencoder corresponding to each data type. Can be either an keras autoencoder object or the file where the autoencoder was saved.
`filter.to.common.patients`	logical, where TRUE indicates to filter out patients that don't have all data types.
`patients.to.return`	vector of patients to return correlations for. If NULL all patients/columns will be used.

Value

matrix where rows are patients and columns are pairs of data types

Author(s)

Natalie Fox

Examples

```
## Not run:

# Load three data types and create an autoencder for each
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');
molecular.data <- list();
ae.result <- list();
for(i in c('cna','snv','methy')) {
  molecular.data[[i]] <- load.molecular.aberration.data(
    paste0(example.molecular.data.dir,i,'_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    );
  ae.result[[i]] <- create.autoencoder(
    data.type = i,
    data.matrix = molecular.data[[i]],
    encoder.layers.node.nums = c(10,2)
    )$autoencoder;
  }

# Create a matrix of the bottleneck layers
irf.matrix <- create.autoencoder.irf.matrix(
  data.types = names(molecular.data),
  data.matrices = molecular.data,
  autoencoders = ae.result
  );

## End(Not run)
```

---

load.molecular.aberration.data

Load molecular aberration data

Description

Load the molecular aberration profiles/feature annotation

Usage

```
load.molecular.aberration.data(file, patients = NULL, annotation.fields = NULL);
```

Arguments

`file`	file name of the matrix containing molecular and annotation data. If it does not contain an _absolute_ path, the file name is _relative_ to the current working directory, 'getwd()' as in read.table.
`patients`	vector of patients IDs. Must match colnames from aberration file
`annotation.fields`	vector referencing the column names for the feature annotation columns

Details

The annotation.fields argument will look for any colnames which contain the values specified in annotation.fields and then the column will be renamed to the value that matched from annotation.fields.

Value

If the patients argument is specified then the patient molecular aberration profiles are returned. If the annotation.fields argument is specified then the feature annotation is returned. If both are specified then the two matrices are returned in a list. If neither is specified then the entire matrix with the mix of patients and annotation is returned.

Author(s)

Natalie Fox

Examples

```
example.aberration.data <- paste0(
  path.package('iSubGen'),
  '/exdata/cna_profiles.txt'
  );

# Load the CNA profiles for patients EP001 through EP030
cna.profiles <- load.molecular.aberration.data(
  example.aberration.data,
  patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
  );

# Load feature annotation for the CNA data
cna.annotation <- load.molecular.aberration.data(
  example.aberration.data,
  annotation.fields = c('gene','start','end')
  );
```

`read.scaling.factors` *Read scaling factors from file*

Description

Read scaling factors from file

Usage

```
read.scaling.factors(scaling.factor.files.dir,data.types);
```

Arguments

`scaling.factor.files.dir`	the directory where the files were saved
`data.types`	a vector of the data types with saved scaling factors

Details

One scale and one center file is saved per data type

Value

a list with a key \"center\" list and a key \"scale\" list. The center and scale list keys match the data.matrices list keys

Author(s)

Natalie Fox

Examples

```
# Get the path for the scaling provided in this R package
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');

# Example #1: reading scaling factors for a single data type
scaling.factors <- read.scaling.factors(example.molecular.data.dir, 'cna');

# Example #2: reading scaling factors for multiple data types
scaling.factors <- read.scaling.factors(example.molecular.data.dir, c('cna','snv','methy'));
```

`write.scaling.factors` *Write scaling factors to file*

Description

Write scaling factors to file

Usage

```
write.scaling.factors(scaling.factors, scaling.factor.files.dir=NULL)
```

Arguments

`scaling.factors`	list with the scaling factors created by calculate.scaling
`scaling.factor.files.dir`	directory to output scaling factor files

Details

Creates two files for each data type key. One file for the recentering values and one file for the rescaling values. Files have the names <data type>_gene_recenter.txt or <data type>_gene_rescale.txt

Value

No return value, called for side effects

Author(s)

Natalie Fox

Examples

```
## Not run:

# load the aberration profiles for three data types
example.molecular.data.dir <- paste0(path.package('iSubGen'),'/exdata/');
molecular.data <- list();
for(i in c('cna','snv','methy')) {
  molecular.data[[i]] <- load.molecular.aberration.data(
    paste0(example.molecular.data.dir,i,'_profiles.txt'),
    patients = c(paste0('EP00',1:9), paste0('EP0',10:30))
    );
  }

# calculate scaling factors for all three data types
scaling.factors <- calculate.scaling(molecular.data);

# save the scaling factors to file
write.scaling.factors(scaling.factors);

## End(Not run)
```

Index