Package 'RESET'

January 20, 2025

Type Package

Title Reconstruction Set Test

Version 1.0.0

Description Contains logic for sample-
level variable set scoring using randomized reduced rank reconstruction error.
Frost, H. Robert (2023) ``Reconstruction Set Test (RESET): a computationally efficient method for
single sample gene set testing based on randomized reduced rank reconstruction error'' <doi:10.1101/2023.04.03.535366>.

Depends R (>= 3.6.0), Matrix

Imports methods (>= 3.6.0)

Suggests Seurat (>= 4.0.0), SeuratObject (>= 4.0.0), sctransform (>= 0.3.2)

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation no

Author H. Robert Frost [aut, cre]

Maintainer H. Robert Frost <rob.frost@dartmouth.edu>

Repository CRAN

Date/Publication 2024-03-06 23:40:14 UTC

Contents

RESET-package *Reconstruction Set Test*

Description

Implementation of Reconstruction Set Test (RESET), a method for single sample variable set testing based on randomized reduced rank reconstruction error.

Details

Package:	RESET
Type:	Package
Version:	0.2.2
Date:	2023
License:	GPL-2

Note

This work was supported by the National Institutes of Health grants R35GM146586, R21CA253408, P20GM130454 and P30CA023108.

Author(s)

H. Robert Frost

References

- Frost, H. R. (2023). Reconstruction Set Test (RESET): a computationally efficient method for single sample gene set testing based on randomized reduced rank reconstruction error. bioRXiv e-prints. doi: https://doi.org/10.1101/2023.04.03.535366

convertToPerVarScores *Utility function to that converts RESET scores to/from per-variable scores*

Description

Utility function that converts the scores generated by reset or resetViaPCA to or from per-variable scores. This has the same effect as the `per.var` argument to reset or resetViaPCA. Conversion to per-variable scores will divide each overall score and each sample-level score by the scaled size of the associated variable set. The variable set sizes are scaled by dividing by the mean size among all tested sets (this will result in scores not changing in magnitude for sets of mean size). Conversion from per-variable scores will instead multiply scored by the scaled variable set size.

Usage

```
convertToPerVarScores(reset.out, var.sets, to.per.var=TRUE)
```

Arguments

`reset.out`	The list returned from a call to reset or resetViaPCA.
`var.sets`	List of variable sets where each element in the list corresponds to a set and the list element is a vector variable names. List names are variable set names. This must the same variable set structure used to create the `reset.out` input via reset.
`to.per.var`	If true, the scores are converted to a per-variable format by dividing each score by the size of the associated variable set size. If false, scores are converted from a per-variable format by multiplying by the associated variable set size.

Value

Version of the input list where scores have been appropriately converted to or from a per-variable format.

See Also

reset,resetViaPCA

Examples

```
# Create a collection of 3 overlapping variable sets of different sizes
var.sets = list(set1=1:10,
                set2=1:20,
                set3=1:30)

 # Simulate a 100-by-100 matrix of random Poisson data
 X = matrix(rpois(10000, lambda=1), nrow=100)

 # Inflate first 10 rows for first 10 variables, i.e., the first
 # 10 samples should have elevated scores for the first variable set
```

```
X[1:10,1:10] = rpois(100, lambda=5)

# Execute RESET using non-randomized basis computation
reset.out = reset(X, var.sets=var.sets, k=2, random.threshold=20)

# Display the overall scores
reset.out$v

# Convert to per-variable scores
reset.out.2 = convertToPerVarScores(reset.out=reset.out, var.sets=var.sets, to.per.var=TRUE)

# Display the overall scores in per-variable format
reset.out.2$v

# Convert from per-variable scores
reset.out.3 = convertToPerVarScores(reset.out=reset.out.2, var.sets=var.sets, to.per.var=FALSE)

# Display the overall scores
reset.out.3$v
```

`convertToPerVarScoresForSeurat` *Utility function to that converts RESET scores contained in a Seurat object to/from per-variable scores*

Description

Utility function that converts the scores generated by resetForSeurat to or from per-variable scores. This has the same effect as the `per.var` argument to resetForSeurat. Conversion to per-variable scores will divide each overall score and each sample-level score by the scaled size of the associated gene set. The gene set sizes are scaled by dividing by the mean size among all tested sets (this will reset in scores not changing in magnitude for sets of mean size). Conversion from per-variable scores will instead multiply scored by the scaled gene set size.

Usage

```
convertToPerVarScoresForSeurat(seurat.data, gene.set.collection, to.per.var=TRUE)
```

Arguments

`seurat.data`	The Seurat object returned from a call to resetForSeurat.
`gene.set.collection`	List of m gene sets for which scores are computed. Each element in the list corresponds to a gene set and the list element is a vector of indices for the genes in the set. The index value is defined relative to the order of genes in the Seurat active assay. This needs to be the same gene set collection used in the call to resetForSeurat.
`to.per.var`	If true, the scores are converted to a per-variable format by dividing each score by the size of the associated variable set size. If false, scores are converted from a per-variable format by multiplying by the associated variable set size.

Value

Version of the Seurat object where the RESET scores have been appropriately converted to or from a per-variable format.

See Also

resetForSeurat

Examples

```
 # Only run example code if Seurat package is available
if (requireNamespace("Seurat", quietly=TRUE) & requireNamespace("SeuratObject", quietly=TRUE)) {
    # Define a collection with three overlapping gene sets of different sizes
    collection=list(set1=1:10, set2=1:20, set3=6:10)
    # Execute on the pbmc_small scRNA-seq data set included with SeuratObject
    # See vignettes for more detailed Seurat examples
    reset.out = resetForSeurat(seurat.data=SeuratObject::pbmc_small,
        num.pcs=2, k=2,
        gene.set.collection=collection)
    # Retrieve the overall scores
    reset.out@assays$RESET@meta.features
    # Convert the scores to a per-variable format
    reset.out.2 = convertToPerVarScoresForSeurat(seurat.data=reset.out,
        gene.set.collection=collection)
    # Retrieve the overall scores
    reset.out.2@assays$RESET@meta.features
}
```

---

createVarSetCollection

Utility function to help create a variable set collection list object

---

Description

Utility function that creates a variable set list in the format required by reset (i.e., list of variable indices) given the variable names and an ordered vector of variable names.

Usage

```
createVarSetCollection(var.names, var.sets, min.size=1, max.size)
```

Arguments

var.names	Vector of variable names. This should correspond to the order of variables as represented by the columns of the target matrix X.
var.sets	List of m variable sets where each element in the list corresponds to a set and the list element is a vector variable names. List names are variable set names.

`min.size` Minimum set size after filtering out variable not in the var.names vector. Sets whose post-filtering size is below this are removed from the final collection list. Default is 1 and cannot be set to less than 1.

`max.size` Maximum variable set size after filtering out variables not in the var.names vector. Sets whose post-filtering size is above this are removed from the final collection list. If not specified, no filtering is performed.

Value

Version of the input variable set collection list where variable names have been replaced by position indices, variables not present in the var.names vector have been removed and sets failing the min/max size constraints have been removed.

See Also

`reset`

Examples

```
# Create a collection with two sets defined over 3 variables
createVarSetCollection(var.names=c("A", "B", "C"),
    var.sets = list(set1=c("A", "B"), set2=c("B", "C")),
    min.size=2, max.size=3)
```

---

randomColumnSpace *Implementation of a sparse powered randomized algorithm for computing a basis for the column space of a matrix.*

Description

Computes a rank `k` approximation of the column space of an n-by-p input matrix `X` using a sparse randomized embedding with optional subspace power iterations. Specifically, a p-by-k random text matrix `O` is created where all elements are generated as independent N(0,1) or U(0,1) random variables except for elements designated as sparse via the specified `sparsity.structure`, which are set to 0. If a sparse structure is used, the non-zero elements can alternatively be set to the constant value of 1 for a non-random embedding. The test matrix is used to create an n-by-k sketch matrix `Y` as `Y=XO`. If `q>0`, subspace power iterations are performed on `Y` via algorithm 2 in the paper by Erichson, et al. associated with the `rsvd` R package (https://doi.org/10.18637/jss.v089.i11). The returned rank `k` column space approximation of `X` is then generated via a column-pivoted QR decomposition of `Y`.

Usage

```
randomColumnSpace(X, k=2, q=0, sparsity.structure=NULL, test.dist="normal")
```

Arguments

X	An n-by-p target matrix.
k	Target rank. Defaults to 2.
q	Number of power iterations. Defaults to 0.
sparsity.structure	Optional sparsity structure. Should be specified as a vector whose elements are the indices (in column-oriented format) of the non-sparse elements in the p x k random test matrix O. If not specified, O will be dense.
test.dist	Type of random variable used to populate non-sparse elements of random test matrix O. Must be either 'normal' for N(0,1) RVs, 'uniform' for U(0,1) RVs or 'constant' for the value of 1. Note that 'constant' should only be used if sparsity.structure is specified.

Value

A n-by-k estimate of the column space of X.

See Also

randomSVD

Examples

```
# Simulate a 100-by-100 matrix of random Poisson data
X = matrix(rpois(10000, lambda=2), nrow=100)
# Create a random sparsity structure for 100-by-5 random test matrix; half elements will be 0
sparsity.structure = sample(1:500, 250, replace=TRUE)
# Compute rank 5 estimate of column space of X using a sparse test matrix
Q = randomColumnSpace(X,k=5,sparsity.structure=sparsity.structure)
# Compute using a dense test matrix with U(0,1) RVs
Q = randomColumnSpace(X,k=5,test.dist="uniform")
```

---

randomSVD *Implementation of a sparse powered randomized singular value decomposition.*

Description

Computes an approximate rank k singular value decomposition (SVD) of an n-by-p input matrix X using a sparse randomized embedding with optional subspace power iterations. The randomColumnSpace method is used to generate an rank k approximation of the column space of X. This n-by-k approximation Y is then used to create a k-by-p projection B of X onto this rank k subspace via B=Y^TX. A non-random SVD is computed for B and this SVD solution is used to generate an approximate rank k SVD of X.

Usage

```
randomSVD(X, k=2, q=0, sparsity.structure=NULL, test.dist="normal")
```

Arguments

X	An n-by-p target matrix.
k	Target rank. Defaults to 2. See description in randomColumnSpace.
q	Number of power iterations. Defaults to 0. See description in randomColumnSpace.
sparsity.structure	Optional sparsity structure. See description in randomColumnSpace.
test.dist	Type of random variable used to populate non-sparse elements of random test matrix. See description in randomColumnSpace.

Value

List with the following elements:

- u a matrix whose columns are the top k approximate left singular vectors of X.
- d a vector containing the top k approximate singular values of X.
- v a matrix whose columns are the top k approximate right singular vectors of X.

See Also

randomColumnSpace

Examples

```
# Simulate a 100-by-100 matrix of random Poisson data
X = matrix(rpois(10000, lambda=2), nrow=100)
# Create a random sparsity structure for 100-by-5 random test matrix; half elements will be 0
sparsity.structure = sample(1:500, 250, replace=TRUE)
# Compute rank 5 SVD of X using a sparse test matrix
svd.out = randomSVD(X,k=5,sparsity.structure=sparsity.structure)
# Compute using a dense test matrix with U(0,1) RVs
svd.out = randomSVD(X,k=5,test.dist="uniform")
```

---

reset *Reconstruction Set Test (RESET)*

Description

Implementation of the Reconstruction Set Test (RESET) method, which transforms an n-by-p input matrix `X` into an n-by-m matrix of sample-level variable set scores and a length m vector of overall variable set scores. Execution of RESET involves the following sequence of steps:

- If `center.X=TRUE`, mean center the columns of `X`. If `X.test` is specified, the centering is instead performed on just the columns of `X` corresponding to each variable set. See documentation for the `X` and `center.X` parameters for more details.
- If `scale.X=TRUE`, scale the columns of `X` to have variance 1. If `X.test` is specified, the scaling is instead performed on just the columns of `X` corresponding to each variable set. See documentation for the `X` and `scale.X` parameters for more details.
- If `center.X.test=TRUE`, mean center the columns of `X.test`. See documentation for the `X.test` and `center.X.test` parameters for more details.
- If `scale.X.test=TRUE`, scale the columns of `X.test`. See documentation for the `X.test` and `scale.X.test` parameters for more details.
- Set the reconstruction target matrix `T` to `X` or, if `X.test` is specified, to `X.test`.
- Compute the norm of `T` and norm of each row of `T`. By default, these are the Frobenius and Euclidean norms respectively.
- For each set in `var.sets`, sample-level and matrix level scores are generated as follows:
  - Create a subset of `X` called `X.var.set` that only includes the columns of `X` correponding to the variables in the set.
  - Compute a rank `k` orthonormal basis `Q` for the column space of `X.var.set`. If the size of the set is less then or equal to `random.threshold`, then this is computed as the top `k` columns of the `Q` matrix from a column-pivoted QR decomposition of `X.var.set`, otherwise, it is approximated using a randomized algorithm implemented by `randomColumnSpace`.
  - The reduced rank reconstruction of `T` is then created as `Q Q^T T`.
  - The original `T` is subtracted from the reconstruction to represent the reconstruction error and the appropriate norm is computed on each row and the entire error matrix.
  - The overall score is the log2 ratio of the norm of the original `T` to the norm of the reconstruction error matrix.
  - The score for each sample is the log2 ratio of the norm of the corresponding row of the original `T` to the norm of the same row of the reconstruction error matrix.
  - If `per.var=TRUE`, then the overall and sample-level scores are divided by the variable set size.

Usage

```
reset(X, X.test, center.X=TRUE, scale.X=FALSE, center.X.test=TRUE, scale.X.test=FALSE,
    var.sets, k=2, random.threshold, k.buff=0, q=0, test.dist="normal", norm.type="2",
      per.var=FALSE)
```

Arguments

`X`	The n-by-p target matrix; columns represent variables and rows represent samples.

X.test — Matrix that will be combined with the var.set variables to compute the reduced rank reconstruction. This is typically a subset or transformation of X, e.g., projection on top PCs. Reconstruction error will be measured on the variables in X.test. If not specified, the entire X matrix will be used for calculating reconstruction error.

center.X — Flag which controls whether the values in X are mean centered during execution of the algorithm. If only X is specified and center.X=TRUE, then all columns in X will be centered. If both X and X.test are specified, then centering is performed on just the columns of X contained in the specified variable sets. Mean centering is especially important for accurate performance when X.test is specified as a reduced rank representation of the X, e.g, as the projection of X onto the top principal components. However, mean centering the entire matrix X can have a dramatic impact on memory requirements if X is a large sparse matrix. In this case, a non-centered X and appropriate X.test (e.g., project onto top PCs of X) can be provided and mean centering performed on just the needed variables during execution of RESET. This "just-in-time" centering is enabled by setting center.X=TRUE and providing both X and X.test. If X has already been mean-centered (and X.test is a subset of this mean-centered matrix or computed using this mean-centered matrix), then center should be specified as FALSE.

scale.X — Flag which controls whether the values in X are are scaled to have variance 1 during execution of the algorithm. Defaults to false. If only X is specified and scale.X=TRUE, then all columns in X will be scaled. If both X and X.test are specified, then scaling is performed on just the columns of X contained in the specified variable sets.

center.X.test — Flag which controls whether the values in X.test, if specified, are mean centered during execution of the algorithm. Centering should be performed consistently for X and X.test, i.e., if center.X is true or X was previously centered, then center.X.test should te true unless X.test previously centered or generated from a centered X.

scale.X.test — Flag which controls whether the values in X.test, if specified, are scaled to have variance 1 during execution of the algorithm. Similar to centering, scaling should be performed consistently for X and X.test, i.e., if scale.X is true or X was previously scaled then scale.X.test should te true unless X.test previously scaled or generated from a scaled X.

var.sets — List of m variable sets, each element is a vector of indices of variables in the set that correspond to columns in X. If variable set information is instead available in terms of variable names, the appropriate format can be generated using createVarSetCollection.

k — Rank of reconstruction. Default to 2. Cannot be larger than the minimum variable set size.

random.threshold — If specified, indicates the variable set size above which a randomized reduced-rank reconstruction is used. If the variable set size is less or equal to random.threshold, then a non-random reconstruction is computed. Defaults to k and cannot be less than k.

k.buff — Additional dimensions used in randomized reduced-rank construction algorithm. Defaults to 0. Values above 0 can improve the accuracy of the randomized recon-

struction at the expense of additional computational complexity. If k.buff=0, then the reduced rank reconstruction can be generated directly from the output of randomColumnSpace, otherwise, a reduced rank SVD must also be computed with the reconstruction based on the top k components.

q — Number of power iterations for randomized SVD (see randomSVD). Defaults to 0. Although power iterations can improve randomized SVD performance in general, it can decrease the sensitivity of the RESET method to detect mean or covariance differences.

test.dist — Distribution for non-zero elements of random test matrix used in randomized SVD algorithm. See description for test.dist parameter of randomSVD method.

norm.type — The type of norm to use for computing reconstruction error. Defaults to "2" for Euclidean/Frobenius norm. Other supported option is "1" for L1 norm.

per.var — If true, the computed scores for each variable set are divided by the scaled variable set size to generate per-variable scores. Variable set size scaling is performed by dividing all sizes by the mean size (this will generate per-variable scores of approximately the same magnitude as the non-per-variable scores).

Value

A list with the following elements:

- S an n-by-m matrix of sample-level variable set scores.
- v a length m vector of overall variable set scores.

See Also

createVarSetCollection,randomColumnSpace

Examples

```
# Create a collection of 5 variable sets each of size 10
var.sets = list(set1=1:10,
                set2=11:20,
                set3=21:30,
                set4=31:40,
                set5=41:50)

# Simulate a 100-by-100 matrix of random Poisson data
X = matrix(rpois(10000, lambda=1), nrow=100)

# Inflate first 10 rows for first 10 variables, i.e., the first
# 10 samples should have elevated scores for the first variable set
X[1:10,1:10] = rpois(100, lambda=5)

# Execute RESET using non-randomized basis computation
reset(X, var.sets=var.sets, k=2, random.threshold=10)

# Execute RESET with randomized basis computation
# (random.threshold will default to k value which is less
# than the size of all variable sets)
```

```
reset(X, var.sets=var.sets, k=2, k.buff=2)

# Execute RESET with non-zero k.buff
reset(X, var.sets=var.sets, k=2, k.buff=2)

# Execute RESET with non-zero q
reset(X, var.sets=var.sets, k=2, q=1)

# Execute RESET with L1 vs L2 norm
reset(X, var.sets=var.sets, k=2, norm.type="1")

# Project the X matrix onto the first 5 PCs and use that as X.test
# Scale X before calling prcomp() so that no centering or scaling
# is needed within reset()
X = scale(X)
X.test = prcomp(X,center=FALSE,scale=FALSE,retx=TRUE)$x[,1:5]
reset(X, X.test=X.test, center.X=FALSE, scale.X=FALSE,
  center.X.test=FALSE, scale.X.test=FALSE, var.sets=var.sets, k=2)
```

`resetForSeurat` *RESET wrapper for scRNA-seq data processed using the Seurat framework*

Description

Executes the Reconstruction Set Test (RESET) method (`reset`) on normalized scRNA-seq data stored in a Seurat object. Will analyze the normalized counts in the active assay (i.e., either the log-normalized or SCTransformed counts).

Usage

```
resetForSeurat(seurat.data, num.pcs, gene.set.collection, k=10, random.threshold,
               k.buff=0, q=0, test.dist="normal", norm.type="2", per.var=FALSE)
```

Arguments

`seurat.data`	The Seurat object that holds the scRNA-seq data. Assumes PCA has already been performed on a centered and scaled version of the normalized counts.
`num.pcs`	Number of PCs to use in the RESET method. If not specified, will use all computed PCs. The projection of the scRNA-seq data onto these PCs will be used as the `X.test` matrix for the `reset` call.
`gene.set.collection`	List of m gene sets for which scores are computed. Each element in the list corresponds to a gene set and the list element is a vector of indices for the genes in the set. The index value is defined relative to the order of genes in the Seurat active assay. Gene set names should be specified as list names. See `createVarSetCollection` for utility function that can be used to help generate this list of indices.

k	See description in reset
random.threshold	See description in reset
k.buff	See description in reset
q	See description in reset
test.dist	See description in reset
norm.type	See description in reset
per.var	See description in reset

Value

An updated Seurat object with the following modifications:

- RESET sample-level gene set score matrix S stored in the "data" slot of a new "RESET" Assay object.
- RESET overall gene set scores stored in the feature metadata column "RESET".

See Also

reset,createVarSetCollection

Examples

```
 # Only run example code if Seurat package is available
if (requireNamespace("Seurat", quietly=TRUE) & requireNamespace("SeuratObject", quietly=TRUE)) {
     # Define a collection with three gene sets
     collection=list(set1=1:10, set2=11:20, set3=21:30)
     # Execute on the pbmc_small scRNA-seq data set included with SeuratObject
     # See vignettes for more detailed Seurat examples
     reset.out = resetForSeurat(seurat.data=SeuratObject::pbmc_small,
         num.pcs=5,
         gene.set.collection=collection)
     # Retrieve the scores for the first 10 cells
     reset.out@assays$RESET[,1:10]
     # Retrieve the overall scores
     reset.out@assays$RESET@meta.features
 }
```

---

resetViaPCA *Reconstruction Set Test (RESET) via PCA*

Description

Wrapper around the reset method that uses the projection of X onto the top num.pcs principal components as X.test. This PC projection is computed using a randomized reduced rank SVD as implemented by randomSVD.

Usage

```
resetViaPCA(X, center=TRUE, scale=FALSE, num.pcs=2, pca.buff=2, pca.q=1, var.sets, k=2,
    random.threshold, k.buff=0, q=0, test.dist="normal", norm.type="2", per.var=FALSE)
```

Arguments

X	See description in reset
center	Flag which controls whether the values in X are mean centered. Note that if center is set to true, centering is performed on the entire X matrix prior to calling randomSVD, which may have significant performance and memory implications of X is large and/or sparse. If center is false, then X will be projected onto the uncentered PCs and center.X and center.X.test will be set to TRUE in the call to reset.
scale	Flag which controls whether the values in X are are scaled to have variance 1. Note that if scale is set to true, scaling is performed on the entire X matrix prior to calling randomSVD, which may have significant performance and memory implications of X is large and/or sparse.
num.pcs	Number of principal components used for computing the projection of X.
pca.buff	Number of extra dimensions used when calling randomSVD to compute the PCs. See k.buff parameter for randomSVD function.
pca.q	Number of power iterations used when calling randomSVD to compute the PCs. See q parameter for randomSVD function.
var.sets	See description in reset
k	See description in reset
random.threshold	See description in reset
k.buff	See description in reset
q	See description in reset
test.dist	See description in reset
norm.type	See description in reset
per.var	See description in reset

Value

A list with the following elements:

- S an n-by-m matrix of sample-level variable set scores.
- v a length m vector of overall variable set scores.

See Also

reset,createVarSetCollection,randomColumnSpace

Examples

```
# Create a collection of 5 variable sets each of size 10
var.sets = list(set1=1:10,
                set2=11:20,
                set3=21:30,
                set4=31:40,
                set5=41:50)

# Simulate a 100-by-100 matrix of random Poisson data
X = matrix(rpois(10000, lambda=1), nrow=100)

# Inflate first 10 rows for first 10 variables, i.e., the first
# 10 samples should have elevated scores for the first variable set
X[1:10,1:10] = rpois(100, lambda=5)

# Execute RESET when reconstruction measured on top 10 PCs
# with mean centering performed before computing PCs
resetViaPCA(X, num.pcs=10, var.sets=var.sets, k=2, random.threshold=10)

# Execute RESET when reconstruction measured on top 10
# uncentered PCs with centering performed as needed inside reset()
resetViaPCA(X, center=FALSE, num.pcs=10, var.sets=var.sets, k=2, random.threshold=10)
```

Index