Package 'AffiXcan'

November 11, 2026

Title A Functional Approach To Impute Genetically Regulated Expression

Version 1.30.0

biocViews GeneExpression, Transcription, GeneRegulation, DimensionReduction, Regression, PrincipalComponent

Description Impute a GReX (Genetically Regulated Expression) for a set of genes in a sample of individuals, using a method based on the Total Binding Affinity (TBA). Statistical models to impute GReX can be trained with a training dataset where the real total expression values are known.

Depends R (>= 3.6), SummarizedExperiment

License GPL-3

Imports MultiAssayExperiment, BiocParallel, crayon

Suggests BiocStyle, knitr, rmarkdown

Encoding UTF-8

LazyData false

VignetteBuilder knitr

RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/AffiXcan

git_branch RELEASE_3_23

git_last_commit 0be3f9a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Alessandro Lussana [aut, cre]

Maintainer Alessandro Lussana <alessandro.lussana@protonmail.com>

Contents

AffiXcan-package 2
affiXcanBs 3
affiXcanGReX 5

affiXcanImpute 6
affiXcanPca 7
affiXcanPcs 8
affiXcanTrain 9
assoc2list 12
computeBs 13
computeCorrelation 14
computeExpr 15
computePca 16
computePcs 18
computeRSquared 19
exprMatrix 20
overlookRegions 21
regionAssoc 21
subsetKFold 22
trainingCovariates 23

Index **24**

`AffiXcan-package` *A functional approach to impute GReX*

Description

Impute a GReX (Genetically Regulated Expression) for a set of genes in a sample of individuals, using a method based on the Total Binding Affinity (TBA) score. Statistical models to impute GReX can be trained on a training dataset where the real total expression values are known.

Details

For every gene a linear regression model can be fitted on a training set of individuals where the real total expression values, the Total Binding Affinity (TBA) values for a set of genomic regions, and the covariates of the population genetic structure, are known. AffiXcan performs a principal component analysis (PCA) on the TBA values to fit a linear model using the formula: GReX ~ PC1 + PC2 + PC3 + ... + COV1 + COV2 + ... Associations between the expressed genes and the regulatory regions, on which the TBA values have to be computed, are needed. TBA can be computed using "vcf_rider" software (see references)

Author(s)

Alessandro Lussana <alessandro.lussana@protonmail.com> Maintainer: Alessandro Lussana <alessandro.lussana@protonmail.com>

References

https://github.com/vodkatad/vcf_rider

Examples

```
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

testingTbaPaths <- system.file("extdata","testing.tba.toydata.rds",
package="AffiXcan")

exprmatrix <- affiXcanImpute(tbaPaths=testingTbaPaths, affiXcanTraining=training,
scale=TRUE)
```

`affiXcanBs` *Fit linear models and compute ANOVA p-values*

Description

Fit linear models and compute ANOVA p-values

Usage

```
affiXcanBs(
  exprMatrix,
  assay,
  regionAssoc,
  pca,
  cov = NULL,
  BPPARAM = bpparam(),
  trainingSamples
)
```

Arguments

`exprMatrix`	A SummarizedExperiment object containing expression data
`assay`	A string with the name of the object in SummarizedExperiment::assays(exprMatrix) that contains expression values
`regionAssoc`	A data.frame with the associations between regulatory regions and expressed genes, and with colnames = c("REGULATORY_REGION", "EXPRESSED_REGION")
`pca`	A list, which is the returningObject$pca from affiXcanPca()

cov — Optional; a data.frame with covariates values for the population structure where the columns are the PCs and the rows are the individual IDs; default is NULL

BPPARAM — A BiocParallelParam object. Default is bpparam(). For details on BiocParallel-Param virtual base class see browseVignettes("BiocParallel")

trainingSamples — A vector of strings. The identifiers (e.g. row names of MultiAssayExperiment objects from tbaPaths) of the samples that have to be considered in the training phase, and not used for the cross-validation

Value

A list containing lists named as the EXPRESSED_REGIONS found in the param regionAssoc. Each of these lists contain three objects:

- coefficients: An object containing the coefficients of the principal components used in the model, completely similar to the "coefficients" from the results of lm()
- p.val: The uncorrected anova p-value of the model
- r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared
- corrected.p.val: The p-value of the model, corrected for multiple testing with benjamini-hochberg procedure

Examples

```
if (interactive()) {
data(exprMatrix)
data(trainingCovariates)
data(regionAssoc)

tbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")
regionsCount <- overlookRegions(tbaPaths)
assay <- "values"

sampleNames <- colnames(exprMatrix)
nSamples <- length(sampleNames)
sampGroups <- subsetKFold(k=5, n=nSamples)
for (i in seq(length(sampGroups))) {
     sampGroups[[i]] <- colnames(exprMatrix)[sampGroups[[i]]]
}

testingSamples <- sampGroups[[1]]
trainingSamples <- sampleNames[!sampleNames %in% testingSamples]

pca <- affiXcanPca(tbaPaths=tbaPaths, varExplained=80, scale=TRUE,
regionsCount=regionsCount, trainingSamples=trainingSamples)

bs <- affiXcanBs(exprMatrix=exprMatrix, assay=assay, regionAssoc=regionAssoc,
pca=pca, cov=trainingCovariates, trainingSamples=trainingSamples)
}
```

affiXcanGReX *Compute a GReX from variables and their coefficients for a set of genes*

Description

Compute a GReX from variables and their coefficients for a set of genes

Usage

```
affiXcanGReX(affiXcanTraining, pcs, BPPARAM = bpparam())
```

Arguments

affiXcanTraining	The returning object from affiXcanTrain()
pcs	A list, which is the returning object from affiXcanPcs()
BPPARAM	A BiocParallelParam object. Default is bpparam(). For details on BiocParallelParam virtual base class see browseVignettes("BiocParallel")

Value

A SummarizedExperiment object containing the imputed GReX values

Examples

```
if (interactive()) {
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

testingTbaPaths <- system.file("extdata","testing.tba.toydata.rds",
package="AffiXcan")

pcs <- affiXcanPcs(tbaPaths=testingTbaPaths, affiXcanTraining=training,
scale=TRUE)

exprmatrix <- affiXcanGReX(affiXcanTraining=training, pcs=pcs)
}
```

affiXcanImpute *Impute a GReX for each gene for which a model was generated*

Description

Impute a GReX for each gene for which a model was generated

Usage

```
affiXcanImpute(tbaPaths, affiXcanTraining, scale = TRUE, BPPARAM = bpparam())
```

Arguments

tbaPaths	A vector of strings, which are the paths to MultiAssayExperiment RDS files containing the tba values
affiXcanTraining	The returning object from affiXcanTrain()
scale	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA; default is TRUE
BPPARAM	A BiocParallelParam object. Default is bpparam(). For details on BiocParallel-Param virtual base class see browseVignettes("BiocParallel")

Value

A SummarizedExperiment object containing imputed GReX values

Examples

```
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc,
varExplained=80, scale=TRUE)

testingTbaPaths <- system.file("extdata","testing.tba.toydata.rds",
package="AffiXcan")

exprmatrix <- affiXcanImpute(tbaPaths=testingTbaPaths,
affiXcanTraining=training, scale=TRUE)
```

affiXcanPca *Perform a PCA on each experiment found in MultiAssayExperiment objects*

Description

Perform a PCA on each experiment found in MultiAssayExperiment objects

Usage

```
affiXcanPca(
  tbaPaths,
  varExplained = 80,
  scale = TRUE,
  regionsCount,
  BPPARAM = bpparam(),
  trainingSamples
)
```

Arguments

tbaPaths	A vector of strings, which are the paths to MultiAssayExperiment RDS files containing the tba values
varExplained	An integer between 0 and 100; varExplained=80 means that the principal components selected to fit the models must explain at least 80 percent of variation of TBA values; default is 80
scale	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA; default is TRUE
regionsCount	An integer, that is the summation of length(assays()) of every MultiAssayExperiment RDS object indicated in the param tbaPaths; it is the returning value from overlookRegions()
BPPARAM	A BiocParallelParam object. Default is bpparam(). For details on BiocParallelParam virtual base class see browseVignettes("BiocParallel")
trainingSamples	A vector of strings. The identifiers (e.g. row names of MultiAssayExperiment objects from tbaPaths) of the samples that have to be considered in the training phase, and not used for the cross-validation

Value

pca: A list containing lists named as the MultiAssayExperiment::experiments() found in the MultiAssayExperiment objects listed in the param tbaPaths. Each of these lists contain two objects:

- eigenvectors: A matrix containing eigenvectors for those principal components selected according to the param varExplained

- pcs: A matrix containing the principal components values selected according to the param varExplained
- eigenvalues: A vector containing eigenvalues for those principal components selected according to the param varExplained

Examples

```
if (interactive()) {
data(exprMatrix)

tbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")
regionsCount <- overlookRegions(tbaPaths)

sampleNames <- colnames(exprMatrix)
nSamples <- length(sampleNames)
sampGroups <- subsetKFold(k=3, n=nSamples)
for (i in seq(length(sampGroups))) {
     sampGroups[[i]] <- colnames(exprMatrix)[sampGroups[[i]]]
}

testingSamples <- sampGroups[[1]]
trainingSamples <- sampleNames[!sampleNames %in% testingSamples]

pca <- affiXcanPca(tbaPaths=tbaPaths, varExplained=80, scale=TRUE,
regionsCount=regionsCount, trainingSamples=trainingSamples)
}
```

`affiXcanPcs` *Compute PCs in MultiAssayExperiment objects using eigenvectors given by user*

Description

Compute PCs in MultiAssayExperiment objects using eigenvectors given by user

Usage

```
affiXcanPcs(
  tbaPaths,
  affiXcanTraining,
  scale,
  BPPARAM = bpparam(),
  testingSamples = NULL
)
```

Arguments

tbaPaths	A vector of strings, which are the paths to MultiAssayExperiment RDS files containing the tba values
affiXcanTraining	The returning object from affiXcanTrain()
scale	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA
BPPARAM	A BiocParallelParam object. Default is bpparam(). For details on BiocParallelParam virtual base class see browseVignettes("BiocParallel")
testingSamples	A vector of strings. The identifiers (e.g. row names of MultiAssayExperiment objects from tbaPaths) of the samples that have not been considered in the training phase, to be used in the cross-validation; default is NULL; if is.null(testingSamples)==TRUE then no filtering is performed

Value

A list of matrices containing the principal components values of TBA for each region; each object of the list is named after the MultiAssayExperiment object from which it derives

Examples

```
if (interactive()) {
data(exprMatrix)
data(trainingCovariates)
data(regionAssoc)

trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")
testingTbaPaths <- system.file("extdata","testing.tba.toydata.rds",
package="AffiXcan")

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

pcs <- affiXcanPcs(tbaPaths=testingTbaPaths, affiXcanTraining=training,
scale=TRUE)
}
```

---

affiXcanTrain *Train the model needed to impute a GReX for each gene*

Description

Train the model needed to impute a GReX for each gene

Usage

```
affiXcanTrain(
  exprMatrix,
  assay,
  tbaPaths,
  regionAssoc,
  cov = NULL,
  varExplained = 80,
  scale = TRUE,
  BPPARAM = bpparam(),
  kfold = 1
)
```

Arguments

exprMatrix	A SummarizedExperiment object containing expression data
assay	A string with the name of the object in SummarizedExperiment::assays(exprMatrix) that contains expression values
tbaPaths	A vector of strings, which are the paths to MultiAssayExperiment RDS files containing the tba values
regionAssoc	A data.frame with the association between regulatory regions and expressed genes and with colnames = c("REGULATORY_REGION", "EXPRESSED_REGION")
cov	Optional. A data.frame with covariates values for the population structure where the columns are the PCs and the rows are the individual IIDs Default is NULL
varExplained	An integer between 0 and 100; varExplained=80 means that the principal components selected to fit the models must explain at least 80 percent of variation of TBA values; default is 80
scale	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA; default is TRUE
BPPARAM	A BiocParallelParam object. Default is bpparam(). For details on BiocParallelParam virtual base class see browseVignettes("BiocParallel")
kfold	An integer. The k definition of k-fold cross-validation on the training dataset. Default is 1. This argument controls the behavior of the function in the following way: • kfold<2: train the models on the whole dataset, not performing the cross-validation • kfold>=2: perform the k-fold cross-validation

Value

The output depends on the parameter kfold

If kfold<2: a list containing three objects: pca, bs, regionsCount

- pca: A list containing lists named as the MultiAssayExperiment::experiments() found in the MultiAssayExperiment objects listed in the param tbaPaths. Each of these lists contain two objects:

  - eigenvectors: A matrix containing eigenvectors for those principal components of the TBA selected according to the param varExplained
  - pcs: A matrix containing the principal components values of the TBA selected according to the param varExplained
  - eigenvalues: A vector containing eigenvalues for those principal components of the TBA selected according to the param varExplained
- bs: A list containing lists named as the EXPRESSED_REGIONS found in the param regionAssoc that have a correspondent rowname in the expression values stored SummarizedExperiment::assays(exprMatrix)$assay. Each of the lists in bs contains three objects:
  - coefficients: The coefficients of the principal components used in the model, completely similar to the "coefficients" from the results of lm()
  - p.val: The uncorrected anova pvalue of the model
  - r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared
  - corrected.p.val: The p-value of the model, corrected for multiple testing with benjamini-hochberg procedure
- regionsCount: An integer, that is the number of genomic regions taken into account during the training phase

If kfold>=2: a list containing k-fold objects, named from 1 to kfold and corresponding to the different cross-validations [i]; each one of these objects is a list containing lists named as the expressed gene IDs [y] (i.e. the rownames() of the object in SummarizedExperiment::assays(exprMatrix) containing the expression values), for which a GReX could be imputed. Each of these inner lists contain two objects:

- rho: the pearson's correlation coefficient (R) between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y, computed with cor()
- rho.sq: the coefficient of determination (R^2) between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y, computed as pearson^2
- cor.test.p.val: the p-value of the cor.test() between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y
- model.p.val: The uncorrected anova pvalue of the model
- model.corrected.p.val: The p-value of the model, corrected for multiple testing with benjamini-hochberg procedure
- model.r.sq: the model's coefficient of determination (R^2) on the training data

Examples

```
if(interactive()) {
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"
```

```
training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE, kfold=3)
}
```

`assoc2list` *Reorganize associations table in a list*

Description

Reorganize associations table in a list

Usage

```
assoc2list(gene, regionAssoc)
```

Arguments

`gene`	A string; the name of an expressed gene
`regionAssoc`	A data.frame with the associations between regulatory regions and expressed genes, and with colnames = c("REGULATORY_REGION", "EXPRESSED_REGION")

Value

A list of data frames, each of which has the same structure of the param regionAssoc, except that contains the information relative to one expressed gene

Examples

```
if (interactive()) {
data(regionAssoc)
expressedRegions <- as.list(as.vector(unique(regionAssoc$EXPRESSED_REGION)))
gene <- expressedRegions[[1]]
assocList <- assoc2list(gene, regionAssoc)
}
```

computeBs *Fit a linear model to impute a GReX for a certain gene*

Description

Fit a linear model to impute a GReX for a certain gene

Usage

```
computeBs(assocRegions, pca, expr, covariates = NULL)
```

Arguments

assocRegions	A data.frame with the associations between regulatory regions and one expressed gene, and with colnames = c("REGULATORY_REGION", "EXPRESSED_REGION")
pca	The returningObject$pca from affiXcanTrain()
expr	A matrix containing the real total expression values, where the columns are genes and the rows are individual IIDs
covariates	Optrional; a data.frame with covariates values for the population structure where the columns are the PCs and the rows are the individual IIDs; default is NULL

Value

A list containing three objects:

- coefficients: An object containing the coefficients of the principal components used in the model, completely similar to the "coefficients" from the results of lm()
- p.val: The uncorrected anova pvalue of the model
- r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared

Examples

```
if (interactive()) {
data(exprMatrix)
data(trainingCovariates)
data(regionAssoc)

tbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")
regionsCount <- overlookRegions(tbaPaths)
assay <- "values"

sampleNames <- colnames(exprMatrix)
nSamples <- length(sampleNames)
sampGroups <- subsetKFold(k=5, n=nSamples)
for (i in seq(length(sampGroups))) {
```

```
    sampGroups[[i]] <- colnames(exprMatrix)[sampGroups[[i]]]
}

testingSamples <- sampGroups[[1]]
trainingSamples <- sampleNames[!sampleNames %in% testingSamples]

pca <- affiXcanPca(tbaPaths=tbaPaths, varExplained=80, scale=TRUE,
regionsCount=regionsCount, trainingSamples=trainingSamples)

cov <- trainingCovariates
cov <- cov[rownames(cov) %in% trainingSamples,]

expr <- SummarizedExperiment::assays(exprMatrix)[[assay]]
expr <- expr[,colnames(expr) %in% trainingSamples]
expr <- t(as.data.frame(expr))

expressedRegions <- as.vector(unique(regionAssoc$EXPRESSED_REGION))
assocList <- BiocParallel::bplapply(X=expressedRegions, FUN=assoc2list,
regionAssoc)
names(assocList) <- expressedRegions
assocRegions <- assocList[[1]]

bs <- computeBs(assocRegions=assocRegions, pca=pca, expr=expr,
covariates=cov)
}
```

`computeCorrelation` *Compute R and R^2 on a particular row of two SummarizedExperiment assays*

Description

Compute R and R^2 on a particular row of two SummarizedExperiment assays

Usage

```
computeCorrelation(geneName, realExpr, imputedExpr)
```

Arguments

`geneName`	A string. The row name in realExpr and imputedExpr objects that identifies the vectors between which R and R^2 have to be computed
`realExpr`	A SummarizedExperiment object containing expression data
`imputedExpr`	The returning object of affiXcanImpute()

Value

A list of two objects:

- rho: the pearson's correlation coefficient (R) between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y, computed with cor()
- rho.sq: the coefficient of determination (R^2) between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y, computed as pearson^2
- cor.test.p.val: the p-value of the cor.test() between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y

Examples

```
if (interactive()) {
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

imputedExpr <- affiXcanImpute(tbaPaths=trainingTbaPaths,
affiXcanTraining=training, scale=TRUE)
realExpr <- exprMatrix

geneName <- "ENSG00000256377.1"
imputedExpr <- SummarizedExperiment::assays(imputedExpr)$GReX
realExpr <- SummarizedExperiment::assays(realExpr)[[assay]]

correlation <- computeCorrelation(geneName, realExpr, imputedExpr)
}
```

---

computeExpr *Compute the imputed GReX for a certain gene on a set of individuals*

Description

Compute the imputed GReX for a certain gene on a set of individuals

Usage

```
computeExpr(bs, pcs)
```

Arguments

bs	A list containing three objects: • coefficients: An object containing the coefficients of the principal components used in the model, completely similar to the "coefficients" object from the results of lm() • p.val: The uncorrected anova pvalue of the model • r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared
pcs	A list, which is the returning object of affiXcanPcs()

Value

A vector of imputed GReX values

Examples

```
if (interactive()) {
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

testingTbaPaths <- system.file("extdata","testing.tba.toydata.rds",
package="AffiXcan")

pcs <- affiXcanPcs(tbaPaths=testingTbaPaths, affiXcanTraining=training,
scale=TRUE)

region <- "ENSG00000256377.1"
bs <- training$bs[[region]]

exprmatrix <- computeExpr(bs=bs, pcs=pcs)
}
```

---

computePca *Perform a PCA on a matrix where columns are variables*

Description

Perform a PCA on a matrix where columns are variables

Usage

```
computePca(data, varExplained = 80, scale = TRUE)
```

Arguments

data	A matrix containing the TBA values for a certain genomic region; columns are PWMs, rows are individuals IIDs
varExplained	An integer between 0 and 100; varExplained=80 means that the principal components selected to fit the models must explain at least 80 percent of variation of TBA values; default is 80
scale	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA; default is TRUE

Value

A list containing two objects:

- eigenvectors: A matrix containing eigenvectors for those principal components selected according to the param varExplained
- pcs: A matrix containing the principal components values selected according to the param varExplained
- eigenvalues: A vector containing eigenvalues for those principal components selected according to the param varExplained

Examples

```
if (interactive()) {
data(exprMatrix)
sampleNames <- colnames(exprMatrix)
nSamples <- length(sampleNames)
sampGroups <- subsetKFold(k=3, n=nSamples)
for (i in seq(length(sampGroups))) {
     sampGroups[[i]] <- colnames(exprMatrix)[sampGroups[[i]]]
}
testingSamples <- sampGroups[[i]]
trainingSamples <- sampleNames[!sampleNames %in% testingSamples]

tbaMatrixMAE <- readRDS(system.file("extdata","training.tba.toydata.rds",
package="AffiXcan"))
tbaMatrixMAE <- MultiAssayExperiment::subsetByRow(tbaMatrixMAE,
                                                  trainingSamples)
tbaMatrix <- MultiAssayExperiment::experiments(tbaMatrixMAE)
tba <- tbaMatrix$ENSG00000256377.1

pca <- computePca(data=tba, varExplained=80, scale=TRUE)

}
```

computePcs *Compute a matrix product between variables and eigenvectors*

Description

Compute a matrix product between variables and eigenvectors

Usage

```
computePcs(region, tbaMatrix, scale, pca)
```

Arguments

region	A string, which is the name of the object in the list MultiAssayExperiment::experiments(tbaMatrix) that contains the TBA values of the genomic region of interest (see the param tbaMatrix)
tbaMatrix	A MultiAssayExperiment object containing the TBA values
scale	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA
pca	The returningObject$pca from affiXcanTrain()

Value

A data.frame containing the principal components values of the TBA in a certain genomic region, as the result of the matrix product between the TBA values and the eigenvectors

Examples

```
if (interactive()) {
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

region <- "ENSG00000256377.1"

tbaMatrixMAE <- readRDS(system.file("extdata","training.tba.toydata.rds",
package="AffiXcan"))

pca <- training$pca
```

```
pcs <- computePcs(region=region, tbaMatrix=tbaMatrixMAE, scale=TRUE, pca=pca)
}
```

`computeRSquared` *Compute R and R^2 between rows of two SummarizedExperiment assays*

Description

Compute R and R^2 between rows of two SummarizedExperiment assays

Usage

```
computeRSquared(
  realExpr,
  imputedExpr,
  assay,
  testingSamples = NULL,
  BPPARAM = bpparam()
)
```

Arguments

`realExpr`	A SummarizedExperiment object containing expression data
`imputedExpr`	The returning object of affiXcanImpute()
`assay`	A string with the name of the object in SummarizedExperiment::assays(realExpr) that contains expression values
`testingSamples`	A vector of strings. The identifiers of the samples that have to be considered by the function; default is NULL; if is.null(testingSamples)==TRUE then no filtering is performed
`BPPARAM`	A BiocParallelParam object. Default is bpparam(). For details on BiocParallelParam virtual base class see browseVignettes("BiocParallel")

Value

A list of lists; inner lists are named after the rows for which the correlation between realExpr and imputedExpr have been computed; inner lists contain two objects:

- rho: the pearson's correlation coefficient (R) between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y, computed with cor()
- rho.sq: the coefficient of determination (R^2) between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y, computed as pearson^2
- cor.test.p.val: the p-value of the cor.test() between the real expression values and the imputed GReX for the cross-validation i on the expressed gene y

Examples

```
if (interactive()) {
trainingTbaPaths <- system.file("extdata","training.tba.toydata.rds",
package="AffiXcan")

data(exprMatrix)
data(regionAssoc)
data(trainingCovariates)

assay <- "values"

training <- affiXcanTrain(exprMatrix=exprMatrix, assay=assay,
tbaPaths=trainingTbaPaths, regionAssoc=regionAssoc, cov=trainingCovariates,
varExplained=80, scale=TRUE)

imputedExpr <- affiXcanImpute(tbaPaths=trainingTbaPaths,
affiXcanTraining=training, scale=TRUE)
realExpr <- exprMatrix

correlation <- computeRSquared(realExpr, imputedExpr, assay)
}
```

---

exprMatrix *Expression data of two genes for 229 individuals*

Description

Toy dataset used in examples to describe affiXcanTrain() function.

Usage

```
data(exprMatrix)
```

Format

An object of class `SummarizedExperiment`.

Details

The data consist in a SummarizedExperiment object that contains a matrix of expression values (RPKM) of two randomly chosen genes ("ENSG00000139269.2" and "ENSG00000256377.1") for 229 individuals of european descent. The data was retrieved subsetting the RNA-sequencing data of EBV-transformed lymphocytes from the GEUVADIS public dataset (see 'source')

Source

https://www.ebi.ac.uk/arrayexpress/files/E-GEUV-3/

Examples

```
library(SummarizedExperiment)
data(exprMatrix)
toyExpressionMatrix <- assays(exprMatrix)$values
```

`overlookRegions` *Count the number of genomic regions on which the TBA was computed*

Description

Count the number of genomic regions on which the TBA was computed

Usage

```
overlookRegions(tbaPaths)
```

Arguments

`tbaPaths`	A vector of strings, which are the paths to MultiAssayExperiment RDS files containing the tba values

Value

An integer, that is the summation of length(assays()) of every MultiAssayExperiment RDS object indicated in the param tbaPaths

Examples

```
if (interactive()) {
testingTbaPaths <- system.file("extdata","testing.tba.toydata.rds",
package="AffiXcan")

regionsCount <- overlookRegions(tbaPaths=testingTbaPaths)
}
```

`regionAssoc` *Associations between regulatory regions and expressed genes*

Description

Toy data used in examples to describe affiXcanTrain() and affiXcanImpute() functions.

Usage

```
data(regionAssoc)
```

Format

An object of class `data.frame`

Details

The object consists in a data.frame with two columns: for every "EXPRESSED_REGION" are listed the associated "REGULATORY_REGION"(s). For the illustrative purpose there are only two expressed genes: "ENSG00000139269.2" and "ENSG00000256377.1" (the same names are used in the SummarizedExperiment object containing the expression matrix, see help(exprMatrix)).

For every gene, the associated regulatory regions were retrieved among the enhancers predicted by preSTIGE, a tool developed by O. Corradin et al. (https://genome.cshlp.org/content/24/1/1.full), in EBV-transformed lymphocytes cell lines. The name of the regulatory region being identical to the name of the expressed gene means that the regulatory region refers to a genomic window that spans at least for 2 Kbp and includes the most upstream TSS of the gene.

Examples

```
data(regionAssoc)
head(regionAssoc)
```

---

subsetKFold *Split the numbers from 1 to n in k equally-sized lists*

Description

Split the numbers from 1 to n in k equally-sized lists

Usage

```
subsetKFold(k, n)
```

Arguments

k	An integer. The number of groups in which the first n natural numbers are to be splitted
n	An integer. Defines the interval 1..n

Value

A list of lists; the first natural n numbers equally distributed across k lists

Examples

```
sampGroups <- subsetKFold(k=5, n=23)
```

`trainingCovariates` *Covariates of the population structure for 229 individuals*

Description

Toy data used in examples to describe affiXcanTrain() function.

Usage

```
data(trainingCovariates)
```

Format

An object of class `data.frame`

Details

This object consists in a data.frame where columns are the first three principal components of the population genetic structure and rows are individuals' IDs. These individuals are the same whom expression values are stored in the expression matrix (see help(exprMatrix))

Genotypes of the individuals were downloaded from the GEUVADIS public dataset (https://www.ebi.ac.uk/arrayexpress/files/E-GEUV-1/) in vcf format. Following L. Price et al. (https://www.sciencedirect.com/science/article/pii/S0002929708003534), long range linkage disequilibrium (LRLD) regions were first filtered out with vcf-tools. Then, following J. Novembre et al. (www.nature.com/articles/nature07331), non-common alleles (MAF < 0.05) were filtered out with vcftools and LD pruning was performed with plink. Finally, principal components were computed with eigenstrat.

Examples

```
data(trainingCovariates)
head(trainingCovariates)
```

Index

∗ **datasets**
exprMatrix, 20
regionAssoc, 21
trainingCovariates, 23
∗ **package**
AffiXcan-package, 2

AffiXcan *(*AffiXcan-package*)*, 2
AffiXcan-package, 2
affiXcanBs, 3
affiXcanGReX, 5
affiXcanImpute, 6
affiXcanPca, 7
affiXcanPcs, 8
affiXcanTrain, 9
assoc2list, 12

computeBs, 13
computeCorrelation, 14
computeExpr, 15
computePca, 16
computePcs, 18
computeRSquared, 19

exprMatrix, 20

overlookRegions, 21

regionAssoc, 21

subsetKFold, 22

trainingCovariates, 23