Package 'AffiXcan'

October 15, 2019

Title A Functional Approach To Impute Genetically Regulated Expression

Version 1.2.0

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

Suggests BiocStyle, knitr, rmarkdown

Encoding UTF-8

LazyData false

VignetteBuilder knitr

RoxygenNote 6.1.1

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

git_branch RELEASE_3_9

git_last_commit 6062c39

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

Author Alessandro Lussana [aut, cre]

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

R topics documented:

`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 + COV3. 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 a linear model and compute ANOVA p value*

Description

Fit a linear model and compute ANOVA p value

Usage

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

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`	A data.frame with covariates values for the population structure
`BPPARAM`	A BiocParallelParam object. Default is bpparam(). For details on BiocParallel-Param virtual base class see browseVignettes("BiocParallel")

Value

A list containing lists named as the REGULATORY_REGIONS found in the param regionAssoc that have a correspondent name in the param pca. Each of these lists contains 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()

pval: The uncorrected anova pvalue of the model, retrieved from anova(model, modelReduced, test="F")$'Pr(>F)'[2]

r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared

correctedP: The p value after the benjamini-hochberg correction for multiple testing, retrived using p.adjust(pvalues, method="BH")

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)

pca <- training$pca

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

`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 BiocParallel-Param 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)

bs <- training$bs

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, 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
`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

```
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")

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, scale, regionsCount,
  BPPARAM = bpparam())
```

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
`scale`	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA
`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")

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 contains 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

Examples

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

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

`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())
```

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 BiocParallel-Param virtual base class see browseVignettes("BiocParallel")

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()) {
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")

regionsCount <- overlookRegions(testingTbaPaths)

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, varExplained,
  scale, BPPARAM = bpparam())
```

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`	A data.frame with covariates values for the population structure where the columns are the PCs and the rows are the individual 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
`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")

Value

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 contains 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

bs: A list containing lists named as the REGULATORY_REGIONS found in the param regionAssoc that have a correspondent colname in the experiments stored in MultiAssayExperiment objects listed in the param tbaPaths. Each of the lists in bs contains four objects:

coefficients: The coefficients of the principal components used in the model, completely similar to the "coefficients" from the results of lm()

pval: The uncorrected anova pvalue of the model, retrieved from anova(model, modelReduced, test="F")$'Pr(>F)'[2]

r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared

correctedP: The p value after the benjamini-hochberg correction for multiple testing, retrived using p.adjust(pvalues, method="BH")

regionsCount: An integer, that is the number of genomic regions taken into account during the training phase

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)
}
```

`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)
```

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`	A data.frame with covariates values for the population structure where the columns are the PCs and the rows are the individual IIDs

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()

pval: The uncorrected anova pvalue of the model, retrieved from anova(model, modelReduced, test="F")$'Pr(>F)'[2]

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

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

assay <- "values"

assocRegions <- regionAssoc[regionAssoc$EXPRESSED_REGION==
"ENSG00000256377.1",]

regionsCount <- overlookRegions(trainingTbaPaths)

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

expr <- SummarizedExperiment::assays(exprMatrix)[[assay]]
expr <- t(as.data.frame(expr))

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

`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() pval: The uncorrected anova pvalue of the model, retrieved from anova(model, modelReduced, test="F")$'Pr(>F)'[2] r.sq: The coefficient of determination between the real total expression values and the imputed GReX, retrived from summary(model)$r.squared
`pcs`	A list, which is the returning object of affiXcanPcs()

Value

A summarizedExperiment object; SummarizedExperiment::assays(returningObject)$GReX is a matrix 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)

bs <- training$bs

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, scale)
```

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
`scale`	A logical; if scale=FALSE the TBA values will be only centered, not scaled before performing PCA

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

Examples

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

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)
}
```

`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)
```

---

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