Package 'modACDC'

May 9, 2026

Title Association of Covariance for Detecting Differential Co-Expression

Version 2.0.1

Maintainer Katelyn Queen <kjqueen@usc.edu>

Description A series of functions to implement association of covariance for detecting differential co-expression (ACDC), a novel approach for detection of differential co-expression that simultaneously accommodates multiple phenotypes or exposures with binary, ordinal, or continuous data types. Users can use the default method which identifies modules by Partition or may supply their own modules. Also included are functions to choose an information loss criterion (ILC) for Partition using OmicS-data-based Complex trait Analysis (OSCA) and Genome-wide Complex trait Analysis (GCTA). The manuscript describing these methods is as follows: Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ``ACDC: a general approach for detecting phenotype or exposure associated co-expression'' (2023) <doi:10.3389/fmed.2023.1118824>.

License MIT + file LICENSE

URL https://github.com/USCbiostats/ACDC

Depends R (>= 4.1.0)

Imports CCP, data.table, doParallel, foreach, genieclust, genio, ggplot2, partition, parallel, tibble, tidyr, tools, utils

Suggests CCA

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Katelyn Queen [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8070-3042>), Joshua Millstein [aut, cph] (ORCID: <https://orcid.org/0000-0001-7961-8943>)

Repository CRAN

Date/Publication 2024-01-30 22:30:02 UTC

Contents

ACDC *ACDC*

Description

ACDC detects differential co-expression between a set of genes, such as a module of co-expressed genes, and a set of external features (exposures or responses) by using canonical correlation analysis (CCA) on the external features and module co-expression values. Modules are detected via Partition.

Usage

```
ACDC(
  fullData,
  ILC = 0.5,
  externalVar,
  identifierList = colnames(fullData),
  numNodes = 1
)
```

Arguments

`fullData`	data frame or matrix with samples as rows, all features as columns; each entry should be numeric gene expression or other molecular data values
`ILC`	information loss criterion for Partition, or the minimum intra-class correlation required for features to be condensed; $0 \leq \text{ILC} \leq 1$; default is 0.50
`externalVar`	data frame, matrix, or vector containing external variable data to be used for CCA, rows are samples; all elements must be numeric
`identifierList`	optional row vector of identifiers, of the same length and order, corresponding to columns in fullData (ex: HUGO symbols for genes); default value is the column names from fullData
`numNodes`	number of available compute nodes for parallelization; default is 1

Details

Modules are identified by Partition, an agglomerative data reduction method which performs both feature condensation and extraction based on a user provided information loss criterion (ILC). Feature condensation into modules are only accepted if the intraclass correlation between the features is at least the ILC. For more information about how the co-expression features are calculated, see the coVar documentation.

Following CCA, which determines linear combinations of the co-expression and external feature vectors that maximize the cross-covariance matrix for each module, a Wilks-Lambda test is performed to determine if the correlation between these linear combinations is significant. If they are significant, that implies there is differential co-expression. If there is only one co-expression value for a module (ie two features in the module) and a single external variable, CCA reduces to a simple correlation test, and the t-distribution is used to test for significant correlation (Widmann, 2005). If the number of co-expression features in a particular module is larger than the number of samples, CCA will return correlation coefficients of 1, and p-values and BH FDR q-values will not be calculated. See ACDChighdim for our solution.

Value

Tibble, sorted by ascending BH FDR value, with columns

moduleNum module identifier

colNames list of column names from fullData of the features in the module

features list of identifiers from input parameter "identifierList" for all features in the module

CCA_corr list of CCA canonical correlation coefficients

CCA_pval Wilks-Lamda F-test p-value or t-test p-value

BHFDR_qval Benjamini-Hochberg false discovery rate q-value

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57** (1995) 289–300.

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824..

Widmann M. One-Dimensional CCA and SVD, and Their Relationship to Regression Maps. *Journal of Climate* **18** (2005) 2785–2792. doi:10.1175/jcli3424.1.

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# run function for diet and genotype
ACDC(fullData = nutrimouse$lipid,
     ILC = 0.50,
     externalVar = data.frame(diet=as.numeric(nutrimouse$diet),
                              genotype=as.numeric(nutrimouse$genotype)))
```

ACDChighdim *ACDChighdim*

Description

ACDC detects differential co-expression between a set of genes, such as a module of co-expressed genes, and a set of external features (exposures or responses) by using canonical correlation analysis (CCA) on the external features and module co-expression values. A high-dimensional module is supplied by the user.

Usage

```
ACDChighdim(
  moduleIdentifier = 1,
  moduleCols,
  fullData,
  externalVar,
  identifierList = colnames(fullData),
  corrThreshold = 0.75
)
```

Arguments

moduleIdentifier	the module identifier given by Partition or other dimension reduction/clustering algorithm; default is 1
moduleCols	list containing indices of column locations in fullData that specify features in the module
fullData	data frame or matrix with samples as rows, all features as columns; each entry should be numeric gene expression or other molecular data values
externalVar	data frame, matrix, or vector containing external variable data to be used for CCA, rows are samples; all elements must be numeric

`identifierList` optional row vector of identifiers, of the same length and order, corresponding to columns in fullData (ex: HUGO symbols for genes); default value is the column names from fullData

`corrThreshold` minimum correlation required between two features to be kept in the dataset; $0 \leq$ corrThreshold ≤ 1; default value is 0.75

Details

If the number of co-expression features in a particular module is larger than the number of samples, CCA will return correlation coefficients of 1, and p-values and BH FDR q-values will not be calculated. This function accepts one of these high dimension modules and reduces the dimensionality by calculating the pairwise correlations for all features and only keeping feature pairs with |correlation| > the user defined corrThreshold with a maximum number of features pairs of $\lfloor \frac{N}{2} \rfloor$. We posit that these highly correlated pairs are the skeleton structure of the full module and therefore an appropriate approximation. Once this structure is identified, co-expression values are calculated and CCA is performed as in ACDC.

For more information about how the co-expression features are calculated, see the coVar documentation.

Following CCA, which determines linear combinations of the co-expression and external feature vectors that maximize the cross-covariance matrix for each module, a Wilks-Lambda test is performed to determine if the correlation between these linear combinations is significant. If they are significant, that implies there is differential co-expression. If there is only one co-expression value for a module (ie two features in the module) and a single external variable, CCA reduces to a simple correlation test, and the t-distribution is used to test for significant correlation (Widmann, 2005).

Value

Tibble, designed to be row binded with output from other ACDC functions after removing the final column, with columns

moduleNum module identifier

colNames list of column names from fullData of the features in the module

features list of identifiers from input parameter "identifierList" for all features in the module

CCA_corr list of CCA canonical correlation coefficients

CCA_pval Wilks-Lamda F-test p-value or t-test p-value

numPairsUsed number of feature pairs with correlation above corrThreshold

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57** (1995) 289–300.

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824..

Widmann M. One-Dimensional CCA and SVD, and Their Relationship to Regression Maps. *Journal of Climate* **18** (2005) 2785–2792. doi:10.1175/jcli3424.1.

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# run function for diet and genotype
ACDChighdim(moduleIdentifier = 1,
            moduleCols = list(1:ncol(nutrimouse$lipid)),
            fullData = nutrimouse$lipid,
            externalVar = data.frame(diet=as.numeric(nutrimouse$diet),
                                     genotype=as.numeric(nutrimouse$genotype)))
```

ACDCmod *ACDCmod*

Description

ACDCmod detects differential co-expression between a set of genes, such as a module of co-expressed genes, and a set of external features (exposures or responses) by using canonical correlation analysis (CCA) on the external features and module co-expression values. Modules are provided by the user.

Usage

```
ACDCmod(
  fullData,
  modules,
  externalVar,
  identifierList = colnames(fullData),
  numNodes = 1
)
```

Arguments

fullData	data frame or matrix with samples as rows, all probes as columns; each entry should be numeric gene expression or other molecular data values
modules	vector of lists where each list contains indices of column locations in fullData that specify features in each module
externalVar	data frame, matrix, or vector containing external variable data to be used for CCA, rows are samples; all elements must be numeric
identifierList	optional row vector of identifiers, of the same length and order, corresponding to columns in fullData (ex: HUGO symbols for genes); default value is the column names from fullData
numNodes	number of available compute nodes for parallelization; default is 1

Details

For more information about how the co-expression features are calculated, see the coVar documentation.

Following CCA, which determines linear combinations of the co-expression and external feature vectors that maximize the cross-covariance matrix for each module, a Wilks-Lambda test is performed to determine if the correlation between these linear combinations is significant. If they are significant, that implies there is differential co-expression. If there is only one co-expression value for a module (ie two features in the module) and a single external variable, CCA reduces to a simple correlation test, and the t-distribution is used to test for significant correlation (Widmann, 2005). If the number of co-expression features in a particular module is larger than the number of samples, CCA will return correlation coefficients of 1, and p-values and BH FDR q-values will not be calculated. See ACDChighdim for our solution.

Value

Tibble, sorted by ascending BH FDR value, with columns

moduleNum module identifier

colNames list of column names from fullData of the features in the module

features list of identifiers from input parameter "identifierList" for all features in the module

CCA_corr list of CCA canonical correlation coefficients

CCA_pval Wilks-Lamda F-test p-value; t-test p-value if there are only 2 features in the module and a single external variable

BHFDR_qval Benjamini-Hochberg false discovery rate q-value

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57** (1995) 289–300.

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824.

Widmann M. One-Dimensional CCA and SVD, and Their Relationship to Regression Maps. *Journal of Climate* **18** (2005) 2785–2792. doi:10.1175/jcli3424.1.

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# partition dataset and save modules
library(partition)
part <- partition(nutrimouse$lipid, threshold = 0.50)
mods <- part$mapping_key[which(grepl("reduced_var_", part$mapping_key$variable)), ]$mapping

# run function for diet and genotype
ACDCmod(fullData = nutrimouse$lipid,
        modules = mods,
        externalVar = data.frame(diet=as.numeric(nutrimouse$diet),
                                 genotype=as.numeric(nutrimouse$genotype)))
```

`coVar` *coVar*

Description

Function to calculate ACDC covariances within a data pair for all samples

Usage

```
coVar(dataPair, fullData)
```

Arguments

dataPair	column indices of two genes to calculate covariance between
fullData	dataframe or matrix with samples as rows, all probes as columns; each entry should be numeric gene expression or other molecular data values

Details

Co-expression for a single sample, s, is defined as

$$c_{s,j,k} \equiv (g_{s,j} - \bar{g_j})\,(g_{s,k} - \bar{g_k})$$

where $g_{s,j}$ denotes the expression of gene j in sample s and $\bar{g_j}$ denotes the mean expression of gene j in all samples.

Denoting the sample size as N, coVar returns the co-expression profile across all samples:

$$c_{j,k} = (c_{1,j,k}, c_{2,j,k}, ..., c_{N,j,k})$$

Value

Co-expression profile, or pairwise covariances for all samples, vector for given features

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824.

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# run function with first two samples
coVar(dataPair = c(1, 2),
      fullData = nutrimouse$lipid)
```

GCTA_par *GCTA_par*

Description

GCTA_par determines the average heritability of the first principal component of either the co-expression or covariance of gene expression modules for a range of increasingly reduced datasets. Dimension reduction is done with Partition, where features are only condensed into modules if the intraclass correlation between the features is at least the user-supplied information loss criterion (ILC), 0 <= ILC <= 1. An ILC of one returns the full dataset with no reduction, and an ILC of zero returns one module of all input features, reducing the dataset to the mean value. For each ILC value, with the number of ILCs tested determined by input parameter ILCincrement, the function returns the point estimate and standard error of the average heritability of the first principal component of the co-expression or covariance of the gene expression modules in the reduced dataset. If input parameter permute is true, the function also returns the same values for a random permutation of the first principle component of the appropriate matrix.

Usage

```
GCTA_par(
  df,
  ILCincrement = 0.05,
  fileLoc,
  gctaPath,
  remlAlg = 0,
  maxRemlIt = 100,
  numCovars = NULL,
  catCovars = NULL,
  summaryType,
  permute = TRUE,
  numNodes = 1,
  verbose = TRUE
)
```

Arguments

df	n x p data frame or matrix of numeric -omics values with no ID column
ILCincrement	float between zero and one determining interval between tested ILC values; default is 0.05
fileLoc	absolute file path to bed, bim, and fam files, including prefix
gctaPath	absolute path to GCTA software
remlAlg	algorithm to run REML iterations in GCTA; 0 = average information (AI), 1 = Fisher-scoring, 2 = EM; default is 0 (AI)
maxRemlIt	the maximum number of REML iterations; default is 100
numCovars	n x c_n matrix of numerical covariates to adjust heritability model for; must be in same person order as fam file; default is NULL

`catCovars`	n x c_c matrix of categorical covariates to adjust heritability model for; must be in same person order as fam file; default is NULL
`summaryType`	one of "coexpression" or "covariance"; determines how to summarize each module
`permute`	boolean value for whether or not to calculate values for a random permutation module summary; default is true
`numNodes`	number of available compute nodes for parallelization; default is 1
`verbose`	logical for whether or not to display progress updates; default is TRUE

Details

Genome-wide Complex Trait Analysis (GCTA) is a suite of C++ functions. In order to use the GCTA functions, the user must specify the absolute path to the GCTA software, which can be downloaded from the Yang Lab website here.

Here, we use GCTA's Genomics REstricted Maximum Likelihood (GREML) method to estimate the heritability of an external phenotype. GREML is called 2*number of modules for each ILC tested if permutations are included.

Dimension reduction is done with Partition, an agglomerative data reduction method which performs both feature condensation and extraction based on a user provided information loss criterion (ILC). Feature condensation into modules are only accepted if the intraclass correlation between the features is at least the ILC. The superPartition function is called if the gene expression dataset contains more than 4,000 features.

Value

Data frame with columns

ILC the information loss criterion used for that iteration

InformationLost percent information lost due to data reduction

PercentReduction percent of variables condensed compared to unreduced data

AveVarianceExplained_Observed average heritability estimate for PC1 of observed summary data

OverallSD_Observed standard deviation of the heritability estimates for PC1 of observed summary data

VarianceExplained_Observed list of heritability estimates for PC1 of observed summary for all modules

SE_Observed list of standard errors of the heritability estimates for PC1 of observed summary data for all modules

AveVarianceExplained_Permuted average heritability for PC1 of permuted summary data

OverallSD_Permuted standard deviation of the heritability estimates for PC1 of permuted summary data

VarianceExplained_Permuted list of heritability estimates for PC1 of permuted summary data for all modules

SE_Permuted list of standard errors of the heritability estimates for PC1 of permuted summary data for all modules

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Yang J, Lee SH, Goddard ME, Visscher PM. GCTA: a tool for genome-wide complex trait analysis. Am J Hum Genet. 2011 Jan 7;88(1):76-82. doi: 10.1016/j.ajhg.2010.11.011. Epub 2010 Dec 17. PMID: 21167468; PMCID: PMC3014363.

See Also

GCTA software - https://yanglab.westlake.edu.cn/software/gcta/

Examples

```
# run function; input absolute path to OSCA software before running
## Not run: GCTA_par(df = geneExpressionData,
          ILCincrement = 0.25,
          fileLoc = "pathHere",
          gctaPath = "pathHere",
          summaryType = "coexpression",
          permute = TRUE,
          numNodes = 1)
## End(Not run)
```

`GCTA_parPlot` *GCTA_parPlot*

Description

GCTA_parPlot creates a graph of the output from the GCTA_par function, plotting average heritability of the first principal component of either co-expression or covariance of gene modules against information lost/percent reduction for both observed and permuted data.

Usage

```
GCTA_parPlot(df, dataName = "", summaryType)
```

Arguments

`df`	output from GCTA_par function with permutations
`dataName`	string of name of data for graph labels; default is blank
`summaryType`	one of "coexpression" or "covariance"; how modules were summarized for GCTA calculations

Details

Genome-wide Complex Trait Analysis (GCTA) is a suite of C++ functions. In order to use the GCTA functions, the user must specify the absolute path to the GCTA software, which can be downloaded from the Yang Lab website here.

In GCTA_par, we use GCTA's Genomics REstricted Maximum Likelihood (GREML) method to estimate the average heritability of the first principal component of either co-expression or covariance of gene modules. The produced plot shows these heritability estimates at varying levels of dataset reduction, calculated for observed data in blue and permuted data in red. An information loss value of 0 represents the unreduced dataset, and an information loss level of 100 represents the data being reduced to the average expression of all features.

Value

ggplot object

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Yang J, Lee SH, Goddard ME, Visscher PM. GCTA: a tool for genome-wide complex trait analysis. Am J Hum Genet. 2011 Jan 7;88(1):76-82. doi: 10.1016/j.ajhg.2010.11.011. Epub 2010 Dec 17. PMID: 21167468; PMCID: PMC3014363.

See Also

GCTA software - https://yanglab.westlake.edu.cn/software/gcta/#Overview

Examples

```
# run OSCA_par and save output; input absolute path to OSCA software before running
## Not run: par <- GCTA_par(df = geneExpressionData,
          ILCincrement = 0.25,
          fileLoc = "pathHere",
          gctaPath = "pathHere",
          summaryType = "coexpression",
          permute = TRUE,
          numNodes = 1)
## End(Not run)

# run function
## Not run: GCTA_parPlot(df=par, dataName = "Example Data", summaryType = "coexpression")
```

GCTA_singleValue *GCTA_singleValue*

Description

Function to return the heritability of an external phenotype for a single dataset

Usage

```
GCTA_singleValue(
  fileLoc,
  externalVar,
  gctaPath,
  remlAlg = 0,
  maxRemlIt = 100,
  numCovars = NULL,
  catCovars = NULL
)
```

Arguments

fileLoc	absolute file path to bed, bim, and fam files, including prefix
externalVar	vector of length n of external variable values with no ID column; must be in the same sample order as bed, bim, fam files
gctaPath	absolute path to GCTA software
remlAlg	algorithm to run REML iterations in GCTA; 0 = average information (AI), 1 = Fisher-scoring, 2 = EM; default is 0 (AI)
maxRemlIt	the maximum number of REML iterations; default is 100
numCovars	n x c_n matrix of numerical covariates to adjust heritability model for; must be in same person order as fam file; default is NULL
catCovars	n x c_c matrix of categorical covariates to adjust heritability model for; must be in same person order as fam file; default is NULL

Details

Genome-wide Complex Trait Analysis (GCTA) is a suite of C++ functions. In order to use the GCTA functions, the user must specify the absolute path to the GCTA software, which can be downloaded from the Yang Lab website here.

Here, we use GCTA's Genomics REstricted Maximum Likelihood (GREML) method to estimate the heritability of an external phenotype.

Value

Row of GREML output containing heritability point estimate of external data and standard error

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Yang J, Lee SH, Goddard ME, Visscher PM. GCTA: a tool for genome-wide complex trait analysis. Am J Hum Genet. 2011 Jan 7;88(1):76-82. doi: 10.1016/j.ajhg.2010.11.011. Epub 2010 Dec 17. PMID: 21167468; PMCID: PMC3014363.

See Also

GCTA software - https://yanglab.westlake.edu.cn/software/gcta/

Examples

```
externalVar <- c()

# run function; input data before running
## Not run: OSCA_singleValue(fileLoc = "pathHere",
                  externalVar = externalVar,
                  gctaPath = "pathHere")
## End(Not run)
```

OSCA_par *OSCA_par*

Description

OSCA_par determines the percent variance explained in an external variable (exposures or responses) for a range of increasingly reduced datasets. Dimension reduction is done with Partition, where features are only condensed into modules if the intraclass correlation between the features is at least the user-supplied information loss criterion (ILC), 0 <= ILC <= 1. An ILC of one returns the full dataset with no reduction, and an ILC of zero returns one module of all input features, reducing the dataset to the mean value. For each ILC value, with the number of ILCs tested determined by input parameter ILCincrement, the function returns the point estimate and standard error of the percent variance explained in the observed external variable by the reduced dataset. If input parameter permute is true, the function also returns the same values for a random permutation of the external variable.

Usage

```
OSCA_par(
  df,
  externalVar,
  ILCincrement = 0.05,
  oscaPath,
  remlAlg = 0,
```

```
  maxRemlIt = 100,
  numCovars = NULL,
  catCovars = NULL,
  permute = TRUE,
  numNodes = 1,
  verbose = TRUE
)
```

Arguments

df	n x p data frame or matrix of numeric -omics values with no ID column
externalVar	vector of length n of external variable values with no ID column
ILCincrement	float between zero and one determining interval between tested ILC values; default is 0.05
oscaPath	absolute path to OSCA software
remlAlg	which algorithm to run REML iterations in GCTA; 0 = average information (AI), 1 = Fisher-scoring, 2 = EM; default is 0 (AI)
maxRemlIt	the maximum number of REML iterations; default is 100
numCovars	n x c_n matrix of numerical covariates to adjust heritability model for; must be in same person order as externalVar; default is NULL
catCovars	n x c_c matrix of categorical covariates to adjust heritability model for; must be in same person order as externalVar; default is NULL
permute	boolean value for whether or not to calculate values for a random permutation of the external variable; default is true
numNodes	number of available compute nodes for parallelization; default is 1
verbose	logical for whether or not to display progress updates; default is TRUE

Details

OmicS-data-based Complex trait Analysis (OSCA) is a suite of C++ functions. In order to use the OSCA functions, the user must specify the absolute path to the OSCA software, which can be downloaded from the Yang Lab website here.

Here, we use OSCA's Omics Restricted Maximum Likelihood (OREML) method to estimate the percent of variance in an external phenotype that can be explained by an omics profile, akin to heritability estimates in GWAS. OREML is called twice for each ILC tested if permutations are included.

Dimension reduction is done with Partition, an agglomerative data reduction method which performs both feature condensation and extraction based on a user provided information loss criterion (ILC). Feature condensation into modules are only accepted if the intraclass correlation between the features is at least the ILC. The superPartition function is called if the gene expression dataset contains more than 4,000 features.

Value

Data frame with columns

ILC the information loss criterion used for that iteration

InformationLost percent information lost due to data reduction

PercentReduction percent of variables condensed compared to unreduced data

VarianceExplained_Observed percent variance explained in observed external variable by the data

SE_Observed standard error of the percent variance estimate for observed external variable

VarianceExplained_Permuted percent variance explained in permuted external variable by the data; only if input parameter "permute" is true

SE_Permuted standard error of the percent variance estimate for permuted external variable; only if input parameter "permute" is true

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57** (1995) 289–300.

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824.

See Also

OSCA software - https://yanglab.westlake.edu.cn/software/osca/

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# run function; input absolute path to OSCA software before running
## Not run: OSCA_par(df = nutrimouse$gene,
          externalVar = as.numeric(nutrimouse$diet),
```

```
          ILCincrement = 0.25,
          oscaPath = "pathHere")
## End(Not run)
```

OSCA_parPlot *OSCA_parPlot*

Description

OSCA_parPlot creates a graph of the output from the OSCA_par function, plotting percent variance explained in an external variable (exposure or response) against information lost/percent reduction for both observed and permuted data.

Usage

```
OSCA_parPlot(df, externalVarName = "", dataName = "")
```

Arguments

df	output from OSCA_par function with permutations
externalVarName	string of name of external variable for graph labels; default is blank
dataName	string of name of data for graph labels; default is blank

Details

OmicS-data-based Complex trait Analysis (OSCA) is a suite of C++ functions. In order to use the OSCA functions, the user must specify the absolute path to the OSCA software, which can be downloaded from the Yang Lab website here.

In OSCA_par, we use OSCA's Omics Restricted Maximum Likelihood (OREML) method to estimate the percent of variance in an external phenotype that can be explained by an omics profile, akin to heritability estimates in GWAS. The produced plot shows the percent variance explained in an external variable at varying levels of dataset reduction, calculated for observed external variables in blue and permuted external variables in red. An information loss value of 0 represents the unreduced dataset, and an information loss level of 100 represents the data being reduced to the average expression of all features.

Value

ggplot object

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57** (1995) 289–300.

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824.

See Also

OSCA software - https://yanglab.westlake.edu.cn/software/osca/

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# run OSCA_par and save output; input absolute path to OSCA software before running
## Not run: par <- OSCA_par(df = nutrimouse$gene,
                 externalVar = as.numeric(nutrimouse$diet),
                  ILCincrement = 0.25,
                  oscaPath = "pathHere")
## End(Not run)

# run function
## Not run: OSCA_parPlot(df=par, externalVarName = "Diet", dataName = "Nutritional Issue Genes")
```

`OSCA_singleValue` *OSCA_singleValue*

Description

Function to return the percent variance explained in an external phenotype for a single dataset

Usage

```
OSCA_singleValue(
  df,
  externalVar,
  oscaPath,
  remlAlg = 0,
  maxRemlIt = 100,
  numCovars = NULL,
  catCovars = NULL
)
```

Arguments

df	n x p dataframe or matrix of numeric -omics values with no ID column
externalVar	vector of length n of external variable values with no ID column
oscaPath	absolute path to OSCA software
remlAlg	which algorithm to run REML iterations in GCTA; 0 = average information (AI), 1 = Fisher-scoring, 2 = EM; default is 0 (AI)
maxRemlIt	the maximum number of REML iterations; default is 100
numCovars	n x c_n matrix of numerical covariates to adjust heritability model for; must be in same person order as fam file; default is NULL
catCovars	n x c_c matrix of categorical covariates to adjust heritability model for; must be in same person order as fam file; default is NULL

Details

OmicS-data-based Complex trait Analysis (OSCA) is a suite of C++ functions. In order to use the OSCA functions, the user must specify the absolute path to the OSCA software, which can be downloaded from the Yang Lab website here.

Here, we use OSCA's Omics Restricted Maximum Likelihood (OREML) method to estimate the percent of variance in an external phenotype that can be explained by an omics profile, akin to heritability estimates in GWAS.

Value

Row of OREML output containing percent variance explained in external data and standard error

Author(s)

Katelyn Queen, <kjqueen@usc.edu>

References

Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* **57** (1995) 289–300.

Martin P, et al. Novel aspects of PPARalpha-mediated regulation of lipid and xenobiotic metabolism revealed through a nutrigenomic study. *Hepatology*, in press, 2007.

Millstein J, Battaglin F, Barrett M, Cao S, Zhang W, Stintzing S, et al. Partition: a surjective mapping approach for dimensionality reduction. *Bioinformatics* **36** (2019) 676–681. doi:10.1093/bioinformatics/btz661.

Queen K, Nguyen MN, Gilliland F, Chun S, Raby BA, Millstein J. ACDC: a general approach for detecting phenotype or exposure associated co-expression. *Frontiers in Medicine* (2023) 10. doi:10.3389/fmed.2023.1118824.

See Also

OSCA software - https://yanglab.westlake.edu.cn/software/osca/

Examples

```
#load CCA package for example dataset
library(CCA)

# load dataset
data("nutrimouse")

# run function; input absolute path to OSCA software before running
## Not run: OSCA_singleValue(df = nutrimouse$gene,
                  externalVar = as.numeric(nutrimouse$diet),
                  oscaPath = "pathHere")
## End(Not run)
```

Index