Package ‘DMRcompare’

July 14, 2018

Title Compare Different R Tools for Detecting Differentially-
Methylated Regions (DMRs) of a Genome

Version 0.0.0.9000

Description The accompanying package to the method comparison paper ``An evaluation of supervised methods for identifying differentially methylated regions in epigenome-wide association studies'' by Mallik et al. (2018), submitted to Briefings in Bioinformatics as a review article. This package contains the organized and documented R scripts necessary to replicate the multi-design-point comparative simulation study, as well as associated tables and figures.

Depends R (>= 3.3.0),
ChAMPdata,
DMRcatedata

Imports bumphunter,
ChAMP,
ChIPpeakAnno,
data.table,
doParallel,
DMRcate,
foreach,
GenomicRanges,
graphics,
grDevices,
IRanges,
minfi,
parallel,
PRROC,
stats

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests GEOquery,
testthat

R topics documented:

`BuildPRcurve` *Build a List of Precision-Recall Curve Objects*

Description

Given a directory of best-performing results files from one of the simulation functions (`WriteDMRcateResults`, `WriteProbeLassoResults`, `WriteBumphunterResults`, or results from the `Comb-p` method in `Python`), import the raw data files and construct PR-curve objects via the `pr.curve` function.

Usage

```
BuildPRcurve(bestResultsDir, delta = c(0.025, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4),
  seed = c(100, 210, 330, 450, 680), beta_mat = betaVals_mat,
  AclustCPG_df = startEndCPG_df, CPGs_df = cpgLocation_df, min.cpgs = 5)
```

Arguments

`bestResultsDir` The name of the directory where the method results from the best-performing parameter settings are stored. For the full design we have included (`delta = c(0.025, 0.05, 0.1, 0` and `seed = c(100, 210, 330, 450, 680)`), this directory should contain 35 `.RDS` files per method.

`delta` A treatment size corresponding to one of the simulations with completed results files in the `bestResultsDir` directory.

`seed` A seed value corresponding to one of the simulations with completed results files in the `bestResultsDir` directory.

`beta_mat` A matrix of beta values across genome on the array. The default value is given in the `betaVals_mat` data set.

`AclustCPG_df` A data frame of `Aclust` results. The default value is given in the `startEndCPG_df` data set.

`CPGs_df` A data frame matching chromosomes to CPG names and locations. The default value is given in the `cpgLocation_df` data set, passed to the `StandardizeOutput` function. This data set is only necessary if the results directory contains Comb-p results with the specified `delta` and `seed` values.

`min.cpgs` The minimum number of CPGs necessary to consider a result significant. Defaults to 5. This argument is only required if the results directory contains Comb-p results with the specified `delta` and `seed` values.

Value

A list of PR-curve objects, to be plotted via the `PlotPRCurve` function.

Examples

```
## Not run:
   BuildPRcurve(
     bestResultsDir = "best_cases_results/",
     delta = 0.4,
     seed = 100
   )

## End(Not run)
```

`PlotOverlaps` *Plot Venn Diagrams of DMR Overlaps*

Description

Given a directory of best-performing results files from one of the simulation functions (`WriteDMRcateResults`, `WriteProbeLassoResults`, `WriteBumphunterResults`, or results from the `Comb-p` method in `Python`), call the `BuildOverlaps` function to import the raw data files and DMR overlap lists, then plot those Venn diagrams and save the plots to a PDF.

Usage

```
PlotOverlaps(bestResultsDir, figsDir, figFileName, delta_num = c(0.025, 0.05,
  0.1, 0.15, 0.2, 0.3, 0.4), seeds_int = c(100, 210, 330, 450, 680),
  totalTest_int = 3063, CPGs_df = cpgLocation_df, min.cpgs = 5)
```

Arguments

`bestResultsDir` The name of the directory where the method results from the best-performing parameter settings are stored. For the full design we have included (`delta = c(0.025, 0.05, 0.1, 0` and `seed = c(100, 210, 330, 450, 680)`), this directory should contain 35 `.RDS` files per method.

`figsDir` In which directory should the figures be saved?

`figFileName` The name of the figure

`delta_num` A vector of treatment sizes with values corresponding to one of the simulations with completed results files in the `bestResultsDir` directory.

`seeds_int` A vector of random seeds with values corresponding to one of the simulations with completed results files in the `bestResultsDir` directory.

`totalTest_int` Parameter passed to the `makeVennDiagram` function. This is an interger value specifying the total number of tests performed to obtain the list of peaks. It should be much larger than the number of peaks in the largest peak set.

`CPGs_df` A data frame matching chromosomes to CPG names and locations. The default value is given in the `cpgLocation_df` data set. This data set is only necessary if the results directory contains Comb-p results with the specified `delta` and `seed` values. This is passed to the `BuildOverlaps` function.

min.cpgs	The minimum number of CPGs before we consider a result significant. Defaults to 5. This argument is only required if the results directory contains Comb-p results with the specified delta and seed values. This is passed to the BuildOverlaps function.

Value

Nothing. A PDF file of plots is created as a side effect.

Examples

```
## Not run:
  PlotOverlaps(
    bestResultsDir = "best_cases_results/",
    figsDir = "best_cases_results/resultsFigures/",
    figFileName = "testVenn_allDesigns2"
  )

## End(Not run)
```

PlotPRCurve *Plot Precision-Recall Curves*

Description

Given a list of PR-curve objects as returned by the BuildPRcurve function, plot the precision-recall curve for each method in a shared figure.

Usage

```
PlotPRCurve(prCurves_ls, new = TRUE, lineWidth = 1, colours = NULL)
```

Arguments

prCurves_ls	A list of PR-curve objects
new	Should the PR curves from this list form their own graph (TRUE) or be added onto a previous PR-curve figure (FALSE). Defaults to TRUE.
lineWidth	The line width of each PR curve in the plot. Defaults to 1.
colours	Optionally add your own colours for each line. Otherwise, the colours are created with the hcl function.

Value

Nothing. A plot is created as a side effect.

Examples

```
## Not run:
   prCurves_0.4_100_ls <-
     BuildPRcurve(
       bestResultsDir = "best_cases_results/",
       delta = 0.4,
       seed = 100
     )

   PlotPRCurve(prCurves_0.4_100_ls)

## End(Not run)
```

ProcessBumphunterResults
Process Bumphunter Results Files

Description

Given a directory of saved Bumphunter results, as written by the WriteBumphunterResults function, import and summarize these data files.

Usage

```
ProcessBumphunterResults(resultsDir, beta_mat, AclustCPG_df, verbose = TRUE)
```

Arguments

resultsDir	The name of the directory where the Bumphunter method results are stored. This should match the directory name supplied to the resultsDir argument of the WriteBumphunterResults function.
beta_mat	A matrix of beta values across genome on the array. This is given in the betaVals_mat data set.
AclustCPG_df	A data frame of Aclust results. This is given in the startEndCPG_df data set.
verbose	Should the function print progress messages? Defaults to TRUE.

Value

A data frame of model fit statistics for the Bumphunter method under each of the given parameter combinations to the data generated for each design configuration

Examples

```
## Not run:
  data("betaVals_mat")
  data("startEndCPG_df")

  bumphunterRes_df <- ProcessBumphunterResults(
    resultsDir = "DMRcate_results/",
    beta_mat = betaVals_mat,
    AclustCPG_df = startEndCPG_df
  )
```

```
## End(Not run)
```

ProcessCombpResults *Extract and Process Comb-p Results Files*

Description

Given a directory of saved Comb-p results, as .RDS files, import, standardize, and summarize these data files.

Usage

```
ProcessCombpResults(resultsDir, beta_mat, AclustCPG_df, cpgLocation_df,
  dmr.sig.threshold = 0.05, min.cpgs = 5, verbose = TRUE)
```

Arguments

resultsDir The name of the directory where the Comb-p method results are stored.

beta_mat A matrix of beta values across genome on the array. This is given in the betaVals_mat data set.

AclustCPG_df A data frame of Aclust results. This is given in the startEndCPG_df data set.

cpgLocation_df A data frame matching chromosomes to CPG names and locations. This is given in the cpgLocation_df data set.

dmr.sig.threshold
Significance level to select regions (with dmr.pval less than the specified value) passed to the internal MergeDMRsWithCPGs function.

min.cpgs The minimum number of CPGs necessary to consider a result significant. Defaults to 5.

verbose Should the function print progress messages? Defaults to TRUE.

Value

A data frame of model fit statistics for the Comb-p method under each of the given parameter combinations to the data generated for each design configuration

Examples

```
## Not run:
  data("betaVals_mat")
  data("startEndCPG_df")
  data("cpgLocation_df")

  combpRes_df <- ProcessCombpResults(
    resultsDir = "DMRcate_results/",
    beta_mat = betaVals_mat,
    AclustCPG_df = startEndCPG_df,
    cpgLocation_df = cpgLocation_df
  )

## End(Not run)
```

ProcessDMRcateResults *Process DMRcate Results Files*

Description

Given a directory of saved DMRcate results, as written by the WriteDMRcateResults function, import and summarize these data files.

Usage

```
ProcessDMRcateResults(resultsDir, beta_mat, AclustCPG_df, verbose = TRUE)
```

Arguments

resultsDir	The name of the directory where the DMRcate method results are stored. This should match the directory name supplied to the resultsDir argument of the WriteDMRcateResults function.
beta_mat	A matrix of beta values across genome on the array. This is given in the betaVals_mat data set.
AclustCPG_df	A data frame of Aclust results. This is given in the startEndCPG_df data set.
verbose	Should the function print progress messages? Defaults to TRUE.

Value

A data frame of model fit statistics for the DMRcate method under each of the given parameter combinations to the data generated for each design configuration

Examples

```
## Not run:
  data("betaVals_mat")
  data("startEndCPG_df")

  dmrcateRes_df <- ProcessDMRcateResults(
    resultsDir = "DMRcate_results/",
    beta_mat = betaVals_mat,
    AclustCPG_df = startEndCPG_df
  )

## End(Not run)
```

ProcessProbeLassoResults *Process ProbeLasso Results Files*

Description

Given a directory of saved ProbeLasso results, as written by the WriteProbeLassoResults function, import and summarize these data files.

Usage

```
ProcessProbeLassoResults(resultsDir, beta_mat, AclustCPG_df, verbose = TRUE)
```

Arguments

`resultsDir` The name of the directory where the ProbeLasso method results are stored. This should match the directory name supplied to the `resultsDir` argument of the `WriteProbeLassoResults` function.

`beta_mat` A matrix of beta values across genome on the array. This is given in the `betaVals_mat` data set.

`AclustCPG_df` A data frame of `Aclust` results. This is given in the `startEndCPG_df` data set.

`verbose` Should the function print progress messages? Defaults to `TRUE`.

Value

A data frame of model fit statistics for the ProbeLasso method under each of the given parameter combinations to the data generated for each design configuration

Examples

```
## Not run:
  data("betaVals_mat")
  data("startEndCPG_df")

  probeLassoRes_df <- ProcessProbeLassoResults(
    resultsDir = "DMRcate_results/",
    beta_mat = betaVals_mat,
    AclustCPG_df = startEndCPG_df
  )

## End(Not run)
```

`RunBumphunter` *Return Results from the* `bumphunter` *Function*

Description

A wrapper function for the Bumphunter method as implemented in the `bumphunter` package, called internally by the `WriteBumphunterResults` function.

Usage

```
RunBumphunter(betaVals_mat, labels_fct = factor(c(rep("Tumor", 7),
  rep("Normal", 7))), chromos_char, chromPosit_num, cpgLocation_df,
  pickCutoffQ_num, maxGap_int, B_int = 10, numCores = detectCores() - 1,
  dmr.sig.threshold = 0.05, min.cpgs = 5)
```

Arguments

`betaVals_mat` A matrix of beta values returned in the second entry of the output from the `SimulateData` function, ordered by the CPGs.

`labels_fct` A factor vector of subject class labels. These should match the observations contained in the columns of the `betaVals_mat` matrix. Defaults to `factor(c(rep("Tumor", 7), rep("`

`chromos_char` A character vector with the chromosomes of each location

`chromPosit_num` A numeric vector representing the chromosomal position

`cpgLocation_df` A data frame matching chromosomes to CPG names and locations. This is given in the `cpgLocation_df` data set.

`pickCutoffQ_num`
The quantile used for picking the cutoff using the permutation distribution, passed to the `bumphunter` function.

`maxGap_int` The maximum location gap, passed to the `bumphunter` function. This will be used to define the clusters of locations that are to be analyzed together via the `clusterMaker` function.

`B_int` An integer denoting the number of resamples to use when computing null distributions, passed to the `bumphunter` function.

`numCores` The number of computing cores for parallel execution, passed to the `registerDoParallel` function. Defaults to one less than the number of cores available on your machine, as detected via the `detectCores` function.

`dmr.sig.threshold`
Significance level to select regions (with `dmr.pval` less than the specified value) passed to the internal `StandardizeOutput` function.

`min.cpgs` The minimum number of CPGs before we consider a result significant, passed to the internal `StandardizeOutput` function. Defaults to 5.

Value

A list of two elements: a data frame of `bumphunter` results that have been standardized by the `StandardizeOutput` function and the computing time for the bumphunter method.

Examples

```
# Called internally by the WriteBumphunterResults() function.
## Not run:
   data("betaVals_mat")
   data("cpgLocation_df")
   data("startEndCPG_df")

   treat_ls <- SimulateData(beta_mat = betaVals_mat,
                            AclustCPG_df = startEndCPG_df,
                            delta_num = 0.4,
                            seed_int = 100)
   class_fct <- factor(c(rep("Tumor", 7), rep("Normal", 7)))

   RunBumphunter(
     betaVals_mat = treat_ls$simBetaVals_df,
     labels_fct = class_fct,
     cpgLocation_df = cpgLocation_df,
     pickCutoffQ_num = 0.95,
     maxGap_int = 250
```

```
  )

## End(Not run)
```

RunDMRcate — *Return Results from the* `dmrcate` *Function*

Description

A wrapper function for the DMRcate method from the `DMRcate` package, called internally by the `WriteDMRcateResults` function.

Usage

```
RunDMRcate(betaVals_mat, labels_fct = factor(c(rep("Tumor", 7), rep("Normal",
  7))), cpgLocation_df, lambda_int, C_int, nCores = 1,
  dmr.sig.threshold = 0.05, min.cpgs = 5, genome = "hg19")
```

Arguments

`betaVals_mat` A matrix of beta values returned in the second entry of the output from the `SimulateData` function.

`labels_fct` A factor vector of subject class labels. These should match the observations contained in the columns of the `betaVals_mat` matrix. Defaults to `factor(c(rep("Tumor", 7), rep("`

`cpgLocation_df` A data frame matching chromosomes to CPG names and locations. This is given in the `cpgLocation_df` data set.

`lambda_int` Gaussian kernel bandwidth for smoothed-function estimation in the called `dmrcate` function.

`C_int` Scaling factor for bandwidth in the internal call to the `dmrcate` function

`nCores` How many cores should be used to perform calculations? Defaults to 1. Note that this function should be called from within the `WriteDMRcateResults` function, which is already written in parallel. Further note that the `DMRcate` package (as of version 1.16.0), does not support parallelization in Windows environments.

`dmr.sig.threshold`
Significance level to select regions (with `dmr.pval` less than the specified value) passed to the internal `StandardizeOutput` function.

`min.cpgs` The minimum number of CPGs necessary to consider a result significant, passed to the internal `StandardizeOutput` function. Defaults to 5.

`genome` Reference genome for annotating DMRs with promoter overlaps, passed to the `extractRanges` function. Can be one of `"hg19"`, `"hg38"`, or `"mm10"`. Defaults to `"hg19"`.

Value

A list of two elements: a data frame of `dmrcate` results that have been standardized by the `StandardizeOutput` function and the computing time for the DMRcate method.

Examples

```
# Called internally by the WriteDMRcateResults() function.
## Not run:
   data("betaVals_mat")
   data("cpgLocation_df")
   data("startEndCPG_df")

   treat_ls <- SimulateData(beta_mat = betaVals_mat,
                            AclustCPG_df = startEndCPG_df,
                            delta_num = 0.4,
                            seed_int = 100)
   class_fct <- factor(c(rep("Tumor", 7), rep("Normal", 7)))

   RunDMRcate(
     betaVals_mat = treat_ls$simBetaVals_df,
     labels_fct = class_fct,
     cpgLocation_df = cpgLocation_df,
     lambda_int = 500, C_int = 5
   )

## End(Not run)
```

RunProbeLasso *Return Results from the* champ.DMR *Function*

Description

A wrapper function for the ProbeLasso method, called internally by the WriteProbeLassoResults function. This function calls the champ.DMR function to perform the ProbeLasso method calculations.

Usage

```
RunProbeLasso(betaVals_mat, labels_fct = factor(c(rep("Tumor", 7),
  rep("Normal", 7))), cpgLocation_df, adjPvalProbe_num, meanLassoRadius_int,
  minDmrSep_int, nCores = 1, dmr.sig.threshold = 0.05, min.cpgs = 5)
```

Arguments

betaVals_mat A matrix of beta values returned in the second entry of the output from the SimulateData function

labels_fct A factor vector of subject class labels. These should match the observations contained in the columns of the betaVals_mat matrix. Defaults to factor(c(rep("Tumor", 7), rep("

cpgLocation_df A data frame matching chromosomes to CPG names and locations. This is given in the cpgLocation_df data set.

adjPvalProbe_num The minimum threshold of significance for probes to be includede in DMRs, passed to the champ.DMR function.

meanLassoRadius_int Radius around each DMP to detect DMR, passed to the champ.DMR function.

`minDmrSep_int` The minimum seperation (bp) between neighbouring DMRs, passed to the `champ.DMR` function.

`nCores` How many cores should be used to perform calculations? Defaults to 1. Note that this function should be called from within the `WriteProbeLassoResults` function, which is already written in parallel. If this function is executed directly (not from within this function), then this argument is passed to the `cores` argument of the `champ.DMR` function.

`dmr.sig.threshold` Significance level to select regions (with `dmr.pval` less than the specified value) passed to the internal `StandardizeOutput` function

`min.cpgs` The minimum number of CPGs necessary to consider a result significant, passed to the internal `StandardizeOutput` function. Defaults to 5.

Value

A list of two elements: a data frame of `champ.DMR` results that have been standardized by the `StandardizeOutput` function and the computing time for the ProbeLasso method.

Examples

```
# Called internally by the WriteProbeLassoResults() function.
## Not run:
   data("betaVals_mat")
   data("cpgLocation_df")
   data("startEndCPG_df")

   treat_ls <- SimulateData(beta_mat = betaVals_mat,
                            AclustCPG_df = startEndCPG_df,
                            delta_num = 0.4,
                            seed_int = 100)
   class_fct <- factor(c(rep("Tumor", 7), rep("Normal", 7)))

   RunProbeLasso(
     betaVals_mat = treat_ls$simBetaVals_df,
     labels_fct = class_fct,
     cpgLocation_df = cpgLocation_df,
     adjPvalProbe_num = 0.05,
     meanLassoRadius_int = 1000,
     minDmrSep_int = 1000
   )

## End(Not run)
```

SimulateData *Simulate Differences in Methylation Data*

Description

Given a randomly selected subset of clusters, add some constant value to each beta value in one observation class

Usage

```
SimulateData(beta_mat, AclustCPG_df, delta_num, seed_int, betaCols_idx = 9:22,
  numEx_int = 7, numClusters_int = 500)
```

Arguments

beta_mat	A matrix of beta values across genome on the array. This is given in the betaVals_mat data set.
AclustCPG_df	A data frame of Aclust results. This is given in the startEndCPG_df data set.
delta_num	The treatment size: a non-negative real number to add to the beta values within randomly-selected clusters for a single class of subjects. This artifically creates differentially-methylated regions (DMRs).
seed_int	The seed value passed to the Random function to enable reproducible results
betaCols_idx	The column numbers of the AclustCPG_df data frame in which beta values for each subject are stored. This function assumes that the subject columns are grouped by their class.
numEx_int	The number of samples in the first group. Once again, this function assumes that these samples are contiguous columns of the AclustCPG_df data frame.
numClusters_int	The total number of clusters to randomly select to be inflated by the treatment amount, delta_num

Value

A list with two elements:

- simBetaVals_dfA data frame of beta values after treatment effects were added, used for input for different DMR-finding methods. Note this is whole-genome data.
- simAclusters_dfA data frame of the methylation values only for Aclust and annotation for whether treatment effects were added. Note this has only CPGs mapped to all the clusters found by the Aclust method.

Examples

```
## Not run:
   data("startEndCPG_df")
   data("betaVals_mat")

   SimulateData(beta_mat = betaVals_mat,
                AclustCPG_df = startEndCPG_df,
                delta_num = 0.4,
                seed_int = 12345)

## End(Not run)
```

WriteBumphunterResults

Calculate and Save Bumphunter Method Results for Specified Design Points

Description

Given a set of design points, simulate appropriate DMR data and apply the bumphunter method to them (with parameters also within the design). Write the results to a file.

Usage

```
WriteBumphunterResults(beta_mat, CPGs_df, Aclusters_df, parallel = TRUE,
  numCores = detectCores() - 2, deltas_num = c(0, 0.025, 0.05, 0.1, 0.15,
  0.2, 0.3, 0.4), seeds_int = c(100, 210, 330, 450, 680),
  cutoffQ_num = c(0.9, 0.95, 0.99), maxGap_int = c(200, 250, 500, 750,
  1000), resultsDir = "DMRcate_compare/", verbose = TRUE)
```

Arguments

beta_mat — A matrix of beta values across genome on the array. This is given in the betaVals_mat data set.

CPGs_df — A data frame matching chromosomes to CPG names and locations. This data frame contains the variables ILMNID, MAPINFO, and chr (e.g. chr1) and is given in the cpgLocation_df data set.

Aclusters_df — A data frame of Aclust results. This is given in the startEndCPG_df data set.

parallel — Should computing be completed over multiple computing cores? Defaults to TRUE.

numCores — If parallel, how many cores should be used? Defaults to two less than the number of available cores (as calculated by the detectCores function). These cores are used internally by the bumphunter function.

deltas_num — A vector of treatment sizes: non-negative real numbers to add to the beta values within randomly-selected clusters for a single class of subjects. This artifically creates differentially-methylated regions (DMRs).

seeds_int — A vector of seed values passed to the Random function to enable reproducible results

cutoffQ_num — A vector of quantiles used for picking the cutoff using the permutation distribution, passed through the call to the internal RunBumphunter call to bumphunter.

maxGap_int — A vector of maximum location gaps, passed to the bumphunter function. These will be used to define the clusters of locations that are to be analyzed together via the clusterMaker function.

resultsDir — Where should the results be saved? Defaults to DMRcate_compare/.

verbose — Should the function print progress messages? Defaults to TRUE.

Details

This function creates matrices of all combinations of design points and all combinations of parameters. For each combination, this function executes the internal RunBumphunter function and saves the results as a compressed .RDS file.

Value

Nothing. Saves output to a file in the specified results directory.

Examples

```
## Not run:
   data("betaVals_mat")
   data("cpgLocation_df")
   data("startEndCPG_df")

   WriteBumphunterResults(
     beta_mat = betaVals_mat,
     CPGs_df = cpgLocation_df,
     Aclusters_df = startEndCPG_df
   )

## End(Not run)
```

`WriteDMRcateResults` *Calculate and Save DMRcate Method Results for Specified Design Points*

Description

Given a set of design points, simulate appropriate DMR data and apply the `dmrcate` method to them (with parameters also within the design). Write the results to a file.

Usage

```
WriteDMRcateResults(beta_mat, CPGs_df, Aclusters_df, parallel = TRUE,
  numCores = detectCores() - 2, deltas_num = c(0, 0.025, 0.05, 0.1, 0.15,
  0.2, 0.3, 0.4), seeds_int = c(100, 210, 330, 450, 680),
  lambdas_num = c(200, 250, 500, 750, 1000), Cs_int = 1:5,
  resultsDir = "DMRcate_compare/", verbose = !parallel)
```

Arguments

`beta_mat` A matrix of beta values across genome on the array. This is given in the `betaVals_mat` data set.

`CPGs_df` A data frame matching chromosomes to CPG names and locations. This data frame contains the variables `ILMNID`, `MAPINFO`, and `chr` (e.g. `chr1`) and is given in the `cpgLocation_df` data set.

`Aclusters_df` A data frame of `Aclust` results. This is given in the `startEndCPG_df` data set.

`parallel` Should computing be completed over multiple computing cores? Defaults to `TRUE`.

`numCores` If `parallel`, how many cores should be used? Defaults to two less than the number of available cores (as calculated by the `detectCores` function).

`deltas_num` A vector of treatment sizes: non-negative real numbers to add to the beta values within randomly-selected clusters for a single class of subjects. This artifically creates differentially-methylated regions (DMRs).

seeds_int
A vector of seed values passed to the Random function to enable reproducible results

lambdas_num
A vector of Gaussian kernel bandwidths for smoothed- function estimation in the called dmrcate function

Cs_int
A vector of scaling factors for bandwidth in the internal call to the dmrcate function

resultsDir
Where should the results be saved? Defaults to DMRcate_compare/.

verbose
Should the function print progress messages? Defaults to TRUE only if parallel = FALSE. See the internal RunDMRcate function for more details about parallel computing with DMRcate.

Details

This function creates matrices of all combinations of design points and all combinations of parameters. For each combination, this function executes the internal RunDMRcate function and saves the results as a compressed .RDS file.

Value

Nothing. Saves output to a file in the specified results directory.

Examples

```
## Not run:
   data("betaVals_mat")
   data("cpgLocation_df")
   data("startEndCPG_df")

   WriteDMRcateResults(
     beta_mat = betaVals_mat,
     CPGs_df = cpgLocation_df,
     Aclusters_df = startEndCPG_df
   )

## End(Not run)
```

WriteProbeLassoResults

Calculate and Save ProbeLasso Method Results for Specified Design Points

---

Description

Given a set of design points, simulate appropriate DMR data and apply the ProbeLasso method (via the champ.DMR function) to them (with parameters also within the design). Write the results to a file.

Usage

```
WriteProbeLassoResults(beta_mat, CPGs_df, Aclusters_df, parallel = TRUE,
  numCores = detectCores() - 2, deltas_num = c(0, 0.025, 0.05, 0.1, 0.15,
  0.2, 0.3, 0.4), seeds_int = c(100, 210, 330, 450, 680),
  pVals_num = c(0.001, 0.01, 0.05, 0.1), aveLassoRad_int = c(375, 700,
  1000), minDmrSep_int = c(200, 250, 500, 750, 1000),
  resultsDir = "DMRcate_compare/", verbose = !parallel)
```

Arguments

beta_mat	A matrix of beta values across genome on the array. This is given in the betaVals_mat data set.
CPGs_df	A data frame matching chromosomes to CPG names and locations. This data frame contains the variables ILMNID, MAPINFO, and chr (e.g. chr1) and is given in the cpgLocation_df data set.
Aclusters_df	A data frame of Aclust results. This is given in the startEndCPG_df data set.
parallel	Should computing be completed over multiple computing cores? Defaults to TRUE.
numCores	If parallel, how many cores should be used? Defaults to two less than the number of available cores (as calculated by the detectCores function).
deltas_num	A vector of treatment sizes: non-negative real numbers to add to the beta values within randomly-selected clusters for a single class of subjects. This artifically creates differentially-methylated regions (DMRs).
seeds_int	A vector of seed values passed to the Random function to enable reproducible results
pVals_num	A vector of the minimum thresholds of significance for probes to be includede in DMRs, passed through the RunProbeLasso function to the champ.DMR function.
aveLassoRad_int	A vector of radii around each differential methylation position to detect DMR, passed to the champ.DMR function.
minDmrSep_int	A vector of the minimum seperation (bp) values between neighbouring DMRs, passed to the champ.DMR function.
resultsDir	Where should the results be saved? Defaults to DMRcate_compare/.
verbose	Should the function print progress messages? Defaults to TRUE only if parallel = FALSE.

Details

This function creates matrices of all combinations of design points and all combinations of parameters. For each combination, this function executes the internal RunProbeLasso function and saves the results as a compressed .RDS file.

Value

Nothing. Saves output to a file in the specified results directory.

Examples

```
## Not run:
   data("betaVals_mat")
   data("cpgLocation_df")
   data("startEndCPG_df")

   WriteProbeLassoResults(
     beta_mat = betaVals_mat,
     CPGs_df = cpgLocation_df,
     Aclusters_df = startEndCPG_df
   )

## End(Not run)
```

Index