Package ‘ramwas’

March 2, 2024

Type  Package
Title  Fast Methylome-Wide Association Study Pipeline for Enrichment Platforms
Version  1.26.0
Date  2019-02-18
Maintainer  Andrey A Shabalin <andrey.shabalin@gmail.com>
Description  A complete toolset for methylome-wide association studies (MWAS). It is specifically designed for data from enrichment based methylation assays, but can be applied to other data as well. The analysis pipeline includes seven steps:
(1) scanning aligned reads from BAM files,
(2) calculation of quality control measures,
(3) creation of methylation score (coverage) matrix,
(4) principal component analysis for capturing batch effects and detection of outliers,
(5) association analysis with respect to phenotypes of interest while correcting for top PCs and known covariates,
(6) annotation of significant findings, and
(7) multi-marker analysis (methylation risk score) using elastic net. Additionally, RaMWAS include tools for joint analysis of methylation and genotype data.
This work is published in Bioinformatics, Shabalin et al. (2018) <doi:10.1093/bioinformatics/bty069>.

URL  https://bioconductor.org/packages/ramwas/

BugReports  https://github.com/andreyshabalin/ramwas/issues
License  LGPL-3
LazyLoad  yes
NeedsCompilation  yes
Depends  R (>= 3.3.0), methods, filematrix
VignetteBuilder  knitr
Suggests  knitr, rmarkdown, pander, BiocStyle, BSgenome.Ecoli.NCBI.20080805
Imports  graphics, stats, utils, digest, glmnet, KernSmooth, grDevices, GenomicAlignments, Rsamtools, parallel, biomaRt, Biostrings, BiocGenerics
biocViews  DNAMethylation, Sequencing, QualityControl, Coverage, Preprocessing, Normalization, BatchEffect, PrincipalComponent, DifferentialMethylation, Visualization

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

git_branch  RELEASE_3_18

git_last_commit  cb4c072

git_last_commit_date  2023-10-24
Repository  Bioconductor 3.18
Date/Publication  2024-03-01

Author  Andrey A Shabalin [aut, cre] (<https://orcid.org/0000-0003-0309-6821>),
Shaunna L Clark [aut],
Mohammad W Hattab [aut],
Karolina A Aberg [aut],
Edwin J C G van den Oord [aut]

R topics documented:

<table>
<thead>
<tr>
<th>R topics documented</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ramwas-package</td>
<td>3</td>
</tr>
<tr>
<td>cachedRDSload</td>
<td>4</td>
</tr>
<tr>
<td>findBestNpvs</td>
<td>5</td>
</tr>
<tr>
<td>get</td>
<td>6</td>
</tr>
<tr>
<td>getCpGset</td>
<td>8</td>
</tr>
<tr>
<td>injectSNPsMAF</td>
<td>9</td>
</tr>
<tr>
<td>insilicoFASTQ</td>
<td>10</td>
</tr>
<tr>
<td>isAbsolutePath</td>
<td>11</td>
</tr>
<tr>
<td>madeBED</td>
<td>12</td>
</tr>
<tr>
<td>makefullpath</td>
<td>13</td>
</tr>
<tr>
<td>manhattan</td>
<td>14</td>
</tr>
<tr>
<td>mat2cols</td>
<td>16</td>
</tr>
<tr>
<td>orthonormalizeCovariates</td>
<td>17</td>
</tr>
<tr>
<td>parameterDump</td>
<td>18</td>
</tr>
<tr>
<td>parameterPreprocess</td>
<td>19</td>
</tr>
<tr>
<td>parametersFromFile</td>
<td>21</td>
</tr>
<tr>
<td>pipeline</td>
<td>22</td>
</tr>
<tr>
<td>plotCV</td>
<td>23</td>
</tr>
<tr>
<td>plotFragmentSizeDistributionEstimate</td>
<td>25</td>
</tr>
<tr>
<td>plotPC</td>
<td>27</td>
</tr>
<tr>
<td>processCommandLine</td>
<td>28</td>
</tr>
<tr>
<td>pvalue2qvalue</td>
<td>29</td>
</tr>
<tr>
<td>QCs</td>
<td>30</td>
</tr>
</tbody>
</table>
Description

RaMWAS provides a complete toolset for methylome-wide association studies (MWAS). It is specifically designed for data from enrichment based methylation assays, but can be applied to other methylomic data as well. The analysis pipeline includes seven steps: (1) scanning aligned reads from BAM files, (2) calculation of quality control measures, (3) creation of methylation score (coverage) matrix, (4) principal component analysis for capturing batch effects and detection of outliers, (5) association analysis with respect to phenotypes of interest while correcting for top PCs and known covariates, (6) annotation of significant findings, and (7) multi-marker analysis (methylation risk score) using elastic net.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>ramwas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>License</td>
<td>LGPL-3</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
<tr>
<td>Depends</td>
<td>methods</td>
</tr>
</tbody>
</table>

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

Maintainer: Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").
## cachedRDSload

### Cached Loading of RDS Files

#### Description

Loads an .rds file `rdsfilename` using `readRDS` and returns the loaded object. The object is also saved in a cache so that repeated calls of the function with the same filename return the same object instantaneously.

#### Usage

```r
cachedRDSload(rdsfilename)
```

#### Arguments

- `rdsfilename` Name of the RDS file.

#### Details

The cached object is stored in a private package environment.

#### Value

Returns the object loaded with `readRDS` from `rdsfilename` at this or a previous call of the function.

#### Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

#### Examples

```r
### Change filename to hg19 CpGset
filename = system.file("extdata", "qc_sample.rds", package = "ramwas")

time1 = system.time( {obj1 = cachedRDSload(filename)} )
time2 = system.time( {obj1 = cachedRDSload(filename)} )

cat("First loading time:",time1[3]," seconds","\n")
cat("Second loading time:",time2[3]," seconds","\n")
```
findBestNpvs

Quickly Find N Smallest P-values in a Long Vector

Description

Finding top, say, 100 p-values out of millions can be slow. This function does it much faster than the usual application of `order(pv)[1:N]`.

Usage

```r
findBestNpvs(pv, n)
```

Arguments

- `pv` Vector of p-values.
- `n` Number of best p-values to select.

Details

The function is a faster analog of `sort(order(pv)[1:N])`

Value

Return a vector of positions of the smallest `N` p-values in `pv`.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See `order`.

Examples

```r
pv = runif(1000)^10
n = 100

# Faster version
topSites1 = findBestNpvs(pv, n)

# Slow alternative
topSites2 = sort(order(pv)[1:n])

# The results must match
stopifnot(all( topSites1 == topSites2 ))
```
Description

Functions for access to data, MWAS results, and location information.

Function `getLocations` obtains the location information for all variables (CpGs).

Function `getMWASandLocations` obtains both MWAS results and location information in a single data frame.

Functions `getDataByLocation` and `getMWASrange` return the data (coverage) and MWAS results for the selected set of variables (CpGs).

Usage

```r
getLocations(x)
getMWAS(x)
getMWASandLocations(x)
getMWASrange(x, chr, start, end)
dataByLocation(x, chr, start, end)
```

Arguments

- `x` Name of directory or list of RaMWAS parameters as described in the "RW6_param.Rmd" vignette.
  
  Try: `vignette("RW6_param","ramwas")`.
  
  If a directory name is provided, it must point to
  
  - Data (coverage) directory (parameter `dircoveragenorm`) for `getDataByLocation` and `getLocations`
  
  - MWAS directory (parameter `dirmwas`) for `getMWAS`, `getMWASandLocations`, and `getMWASrange`

- `chr` Chromosome name or number.

- `start` Start position of the genomic region of interest.

- `end` End position of the genomic region of interest.

Details

The functions return the MWAS results and/or locations.

Value

Function `getLocations` returns a data frame with

```r
chr Chromosome
```
Function `getMWAS` returns a data frame with

- `cor`: coverage - phenotype correlation
- `t.test`: t-statistic
- `p.value`: p-value
- `q.value`: q-value (FDR)

If the outcome variable was categorical, columns `cor` and `t.test` are replaced with `R.squared` and `F-test`.

Functions `getMWASandLocations` and `getMWASrange` return a data frame with elements of output of both `getLocations` and `getMWAS`.

Function `getDataByLocation` returns a list with

- `locations`: Chromosomal location information for located variables
- `matrix`: Data (coverage) matrix for the selected locations

**Author(s)**

Andrey A Shabalin <andrey.shabalin@gmail.com>

**See Also**

See vignettes: `browseVignettes("ramwas")`.

**Examples**

```r
## Not run:
# Extract locations using parameter vector
getLocations(param)

# Extract locations using directory name
getLocations("/data/myMWAS")

# Extract MWAS using parameter vector
getMWAS(param)

# Extract MWAS using directory name
getMWAS("/data/myMWAS")

# Extract MWAS using parameter vector
getMWASandLocations(param)

# Extract MWAS using directory name
getMWASandLocations("/data/myMWAS")
```
# Extract MWAS for a region
getMWASrange(param, 1, 123321, 223321)

# Chromosome can be character
getMWASrange(param, "chr1", 123321, 223321)

# Extract data for a region
getDataByLocation(param, 1, 123321, 223321)

# Chromosome can be character
getDataByLocation(param, "chr1", 123321, 223321)

## End(Not run)

---

**getCpGset**  
*Construct CpG set for a Reference Genome*

**Description**
Finds all CpGs in a reference genome.

**Usage**
```
getCpGsetCG(genome)
getCpGsetALL(genome)
```

**Arguments**
- **genome**  
  A BSgenome object or a character vector with genome sequences.

**Details**
The getCpGsetCG function searches for all CG pairs in the genome.  
The getCpGsetALL function also works for genomes with injected SNPs.

**Value**
Returns a list with CpG coordinates for each genome sequence.

**Author(s)**
Andrey A Shabalin <andrey.shabalin@gmail.com>
Examples

### Using a BSGenome input

```r
library(BSgenome.Ecoli.NCBI.20080805)
cpgset = getCpGsetCG(BSgenome.Ecoli.NCBI.20080805)

print("First 10 CpGs in NC_008253:")
print(cpgset$NC_008253[1:10])
```

### Using a character vector input

```r
genome = list(
  chr1 = "AGCGTTTTCATTCTGACTGCAACGGGCYR",
  chr2 = "AAAAACGCCTTAGTAAGTGATTTTCGYR"
)
cpgset1 = getCpGsetCG(genome)
cpgset2 = getCpGsetALL(genome)

print("Pure CG coordinates in the toy genome:")
print(cpgset1)

print("CG coordinates in the toy genome possible with SNPs:")
print(cpgset2)
```

---

**injectSNPsMAF**

Inject SNPs from VCF Count File into a DNA Sequence

### Description

Injects SNPs from a VCF count file into a DNA sequence.

### Usage

```r
injectSNPsMAF(gensequence, frqcount, MAF = 0.01)
```

### Arguments

- **gensequence**: A string or DNAString of the DNA sequence.
- **frqcount**: File name of the allele count file produced by vcf tools with --counts parameter. Alternatively, the file content can be provided as a character vector (see readLines).
- **MAF**: SNPs with minor allele frequency at or above MAF are injected.

### Value

Returns a string with the genome sequence with SNPs injected.
**insilicoFASTQ**

Construct FASTQ File for In-silico Alignment Experiment

---

**Description**

Creates a FASTQ file with all fragments of `fraglength` bp long.

**Usage**

```r
insilicoFASTQ(con, gensequence, fraglength)
```

**Arguments**

- `con` A connection object or a character string naming the output file. If the name ends with ".gz", a compressed file is created. An empty string can be used to output to the console.
- `gensequence` A string or DNAString of the DNA sequence.
- `fraglength` Fragment length.

---

### Examples

```r
gensequence1 = "AAAACAAAA"
frqcount = c(
  "CHROM\tPOS\nALLELES\nCHR\tALLELE:COUNT",
  "1\t6\t1000\tA:400\tG:600",
  "1\t7\t1000\tA:800\tC:200",
  "1\t9\t1000\tA:900\tG:100")
MAF = 0.01

gensequence2 = injectSNPsMAF(gensequence1, frqcount, MAF)

### No CpGs without SNPs
show(gensequence1)
getCpGsetCG(gensequence1)

### SNPs create 1 CpG
show(gensequence2)
getCpGsetALL(gensequence2)
```
Details

The function a FASTQ file with all fragments of *fraglength* bp long from the forward strand of the DNA sequence.

Value

Returns a list with CpG coordinates for each genome sequence.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

Examples

```r
## There are four 4 bp fragments in a 7 basepair sequence:
insilicoFASTQ(con="", gensequence = "ABCDEFG", fraglength=4)
```

---

**isAbsolutePath**  
*Check if Path is Absolute.*

Description

Check whether a path is relative or absolute.

Usage

```r
isAbsolutePath(path)
```

Arguments

- `path`  
  Path to be tested.

Details

The function is designed to work with both Windows and Unix paths.

Value

TRUE if the path is absolute, FALSE otherwise.

Note

This function improves upon the analog function in *R.utils* package. For instance, "~hi" is not an absolute path.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>
See Also

See also makefullpath.

Examples

```r
isAbsolutePath("C:/123") # TRUE
isAbsolutePath("-123")  # FALSE
isAbsolutePath("~/123") # TRUE
isAbsolutePath("/123")  # TRUE
isAbsolutePath("\123")  # TRUE
isAbsolutePath("asd\123") # FALSE
isAbsolutePath("a\123")  # FALSE
```

Description

Functions for exporting MWAS results in BED format files.

Function `madeBED` saves MWAS findings in BED format for all variables (CpGs), while `madeBEDrange` selects only variables on a given chromosome between given locations.

Functions `madeBEDgraph` and `madeBEDgraphRange` do the same, but create a file in BedGraph format.

Usage

```r
madeBED(x, filename)
madeBEDrange(x, filename, chr, start, end)
madeBEDgraph(x, filename)
madeBEDgraphRange(x, filename, chr, start, end)
```

Arguments

- `x` Name of MWAS directory (parameter `dirmwas`) or list of RaMWAS parameters as described in the "RW6_param.Rmd" vignette.
  Try: `vignette("RW6_param","ramwas").`
- `filename` Name of the BED file to create. If file exists, it’s overwritten.
- `chr` Chromosome name or number.
- `start` Start position of the genomic region of interest.
- `end` End position of the genomic region of interest.

Details

The function returns the MWAS results with locations.
Value

Returns a data.frame with BED file content:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrom</td>
<td>Chromosome</td>
</tr>
<tr>
<td>chromStart</td>
<td>Start position</td>
</tr>
<tr>
<td>chromEnd</td>
<td>End position</td>
</tr>
<tr>
<td>name</td>
<td>Empty name column. BED format only</td>
</tr>
<tr>
<td>score</td>
<td>p-value</td>
</tr>
</tbody>
</table>

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

```r
## Not run:
# Extract BED file using parameter vector
madeBED(param, "file.bed")
madeBEDrange(param, "file.bed", 1, 123321, 223321)

# Extract BED file using directory name
madeBED("/data/myMWAS", "file.bed")
madeBEDrange("/data/myMWAS", "file.bed", 1, 123321, 223321)
```

makefullpath

Combine Path and Filename into Filename with Path

Description

Combine a path with a filename into filename with path.

Usage

makefullpath(path, filename)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path, relative to which the filename is expected to be. Can be absolute, relative, or NULL.</td>
</tr>
<tr>
<td>filename</td>
<td>Can be just filename, include relative path, or include absolute path.</td>
</tr>
</tbody>
</table>
Details
Function returns filename if it includes absolute path or if path is NULL.

Value
Filename with the path included.

Author(s)
Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also
See also `isAbsolutePath`.

Examples
```r
makefullpath("dir1/dir2", "file.txt")
# "dir1/dir2/file.txt"

makefullpath("dir1/dir2", "dir3/file.txt")
# "dir1/dir2/dir3/file.txt"

# Path is ignored if the filename already includes absolute path
makefullpath("dir1/dir2", "/file.txt")
# "/file.txt"

makefullpath("dir1/dir2", "C:/file.txt")
# "C:/file.txt"
```

manhattan  

Fast Manhattan plot for Large Number of P-values

Description
The function `manPlotFast` creates a Manhattan plot. The function `manPlotPrepare` extracts necessary information from a vector of p-values sufficient for creating a Manhattan plot. It optimized to work quickly even for tens of millions of p-values.

Usage
```r
manPlotPrepare(
  pvalues,  
  chr,      
  pos,      
  ismlog10 = FALSE,
)```
chrmargins = 5e6)

manPlotFast(
  man,
  ylim = NULL,
  colorSet = c('steelblue4','#2C82D1','#4CB2D1'),
  yaxmax = NULL,
  lwd = 3,
  axistep = 2,
  cex = 1)

Arguments

pvalues   Vector of p-values. As is (if ismlog10 = FALSE) or minus log10 transformed (if ismlog10 = TRUE).
chr, pos  Vectors indicating the chromosomes and genomic positions (in basepairs) for each p-value in pvalues.
ismlog10  Specifies whether the provides p-values (pvalues parameter) are minus log10 transformed (~ log10(pv))
chrmargins The plot margins at the ends of chromosomes (in basepairs).
man       Object returned by manPlotPrepare.
ylim      Numeric vectors of length 2, giving the y coordinate range. Exactly as in Plotting Parameters.
colorSet  Colors of points, rotating over chromosomes. Points for first chromosome have color colorSet[1], next colorSet[2], etc. Once the colors are exhausted, the colors are reused from the beginning.
yaxmax    Maximum reach of the y axis.
lwd       The line width. As in Graphics Parameters.
axistep   Distance between axis label ticks for y axis.
cex       The size of Manhattan plot points. As in Graphics Parameters.

Details

The function manPlotFast creates Manhattan plot. It requires the use of the function manPlotPrepare which extracts the necessary information from a vector of p-values sufficient for creating Manhattan plot. The resulting object is many times smaller than the vector of p-values.

Value

This function manPlotPrepare returns an object with information for creating Manhattan plot.

Note

The plot has no title. To add a title use title.
**Author(s)**

Andrey A Shabalin <andrey.shabalin@gmail.com>

**See Also**

See vignettes: browseVignettes("ramwas").

**Examples**

```r
# Simulate data (9 chromosomes, million tests each)
chr = rep(paste0('chr', 1:9), each = 1e6)
pos = rep(1:1e6, 9)
pv = runif(9e6)^1.1

# Extract the Manhattan plot info
man = manPlotPrepare(pv, chr, pos, chrmargins = 1000)

# Create Manhattan plot
manPlotFast(man)
title("Manhattan plot")

# Size of p-values before extraction of Manhattan plot info
object.size(list(pv, chr, pos))

# Size of the Manhattan plot info object
object.size(man)
```

---

**mat2cols**

*Split a Matrix into Column Vectors*

**Description**

Internal function for splitting a matrix into column vectors.

**Usage**

```r
mat2cols(x)
```

**Arguments**

- `x` A matrix.

**Value**

List of matrix columns.

**Author(s)**

Andrey A Shabalin <andrey.shabalin@gmail.com>
orthonormalizeCovariates

See Also

See vignettes: browseVignettes("ramwas").

Examples

```r
# Sample data
data = matrix(1:12, nrow = 3)

# Split it
mat2cols(data)
```

orthonormalizeCovariates

Orthonormalize Covariates

Description

Takes a matrix of data frame with covariates, adds a constant covariate (optional), and orthonormalizes the set.

Usage

```r
orthonormalizeCovariates(cvrt, modelhasconstant)
```

Arguments

- `cvrt`: A matrix or data frame with covariates (one column per covariate).
- `modelhasconstant`: Set to TRUE to add a constant covariate into the set before normalization.

Details

Factor variables are split into dummy variables before orthonormalization. The operation is performed via QR decomposition (`qr`).

Value

Returns a matrix with orthogonal columns with unit length, whose columns spans the same space as the covariates plus a constant (if `modelhasconstant` is TRUE).

Note

This function is used in several parts of the pipeline.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>
Examples

# Sample matrix of covariates
covariates = data.frame(a = 1:12, b = 12:1)

# Orthonormalizing Covariates
cvrtqr = orthonormalizeCovariates(covariates, modelhasconstant = TRUE)

# Checking the results (round to ignore rounding errors)
print( round(crossprod(cvrtqr),15) )

# Stop if not orthonormal
stopifnot(all.equal( crossprod(cvrtqr), diag(ncol(cvrtqr)) ))

# Example with a factor variable
groups = data.frame(gr = c("a","a","a","b","b","b","c","c","c"))
orthonormalizeCovariates(groups)

parameterDump

Save Parameters in a Text File

Description

Saves parameters in a text file, prioritizing those listed in toplines.

Usage

parameterDump(dir, param, toplines = NULL)

Arguments

dir
  Directory to save the parameters to. The file is named "UsedSettings.txt".

param
  A list with RaMWAS parameters. Or any list in general.
  For detailed description of all available parameters run:
  browseVignettes("ramwas").

toplines
  Names of the elements in param to save first (top of the file).

Details

This function is used internally by multiple RaMWAS functions to record parameters used to run
the analysis.

Value

The function creates a file and returns nothing.

Note

This function is not intended to be run by the user.
Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas")

Examples

```r
param = ramwasParameters(
    number = 123123,
    integer = 312L,
    textline = "Hi there",
    characterVector = c("Hi","Hi again","Bye"),
    dataframe = data.frame(a = 1:12, b = 12:1)
)

thedir = tempdir()

parameterDump(thedir, param, c("integer","characterVector"))

cat( readLines( paste0(thedir,"/UsedSettings.txt") ), sep = "\n")

file.remove( paste0(thedir,"/UsedSettings.txt") )
```

Description

Fill in missing parameters with default values, read supporting data files, make relative directory path parameters absolute.

Usage

```r
parameterPreprocess(param)
```

Arguments

- **param** List with RaMWAS parameters.
  For detailed description of all available parameters run: browseVignettes("ramwas").
parameterPreprocess

Details

A number of common preprocessing steps necessary for parameters of multiple pipeline parts are combined in this function. The actions include

- Fill in default values for all missing parameters.
- Set `bamnames` parameter to the content of file `filebamlist` file (if `bamnames` was not set).
- Set `bam2sample` parameter to processed content of file `filebam2sample` file (if `bam2sample` was not set).
- Set `covariates` parameter to the data frame from file `filecovariates` file (if `covariates` was not set).
- Check parameters for consistency, i.e. that `modelcovariates` include only names of columns in `covariates`.
- Check that files `filecpgset` and `filenoncpgset` exist if the parameters are set.

Value

Returns preprocessed list of parameters.

Note

This function is not intended to be run by the user.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: `browseVignettes("ramwas")`.

Examples

```r
param = ramwasParameters(
    dirproject = "."
)

param2 = parameterPreprocess(param)

print(param2)
```
parametersFromFile

Scan Parameters From a R Code File

Description

The pipeline parameters can be stored in a simple file, formatted as R code. The parametersFromFile function transforms them into a parameter list used by RaMWAS steps.

Usage

parametersFromFile(.parameterfile)

Arguments

絮.var parameterfile  Name of the file with the parameters set as R variables. See the example below.

Details

Variables with names starting with period (.) are ignored.

Value

Returns the list with all the variables set in the file.

Note

The file .parameterfile is executed as R code, so use only trusted parameter files.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

filename = tempfile()

# Create a file with lines
# dirproject = "."
# modelcovariates = c("Age","Sex")
writeLines(
    con = filename,
    text = c(
        "dirproject = ".",
        "modelcovariates = c("Age","Sex")")
)

pipeline

RaMWAS: High Level Pipeline Functions

Description

These functions provide a simple way to run all steps of RaMWAS pipeline.

Usage

ramwas1scanBams(param)
pipelineProcessBam(bamname, param)
ramwas2collectqc(param)
ramwas3normalizedCoverage(param)
ramwas4PCA(param)
ramwas5MWAS(param)
ramwas6annotateTopFindings(param)
ramwas7ArunMWASes(param)
ramwas7BrunElasticNet(param)
ramwas7CplotByNCpGs(param)
ramwas7riskScoreCV(param)
ramwasSNPs(param)

Arguments

param List with RaMWAS parameters.
For detailed description of all available parameters run:
browseVignettes("ramwas").

bamname Name of the BAM file to process. Can be absolute or relative to dirbam parameter (in param list).

Details

See vignettes for details: browseVignettes("ramwas").

Value

Function pipelineProcessBam returns "OK. <bamname>" if no error occurred. Otherwise, returns text with error. Other functions return nothing.
Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

```r
param = ramwasParameters(
    dirbam = "/project/bams",
    dirproject = "/project",
    filebamlist = "000_list_of_files.txt",
    scoretag = "AS",
    minscore = 100,
    cputhreads = 4,
    filecpgset = "/RaMWAS/hg19_1kG_MAF_0.01_chr1-22_bowtie2_75bp.rds",
    filenoncpgset = "/RaMWAS/hg19_1kG_MAF_0.01_chr1-22_bowtie2_75bp_nonCpG.rds",
    maxrepeats = 3,
    maxfragmentsize = 250,
    minfragmentsize = 75,
    filebam2sample = "000_list_of_files.txt",
    filecovariates = "Covariates.txt",
    modelcovariates = c("Age","Sex"),
    modeloutcome = "CellType",
    modelPCs = 1,
    cvnfolds = 10,
    mmncpgs = 1000,
    mmalpha = 0
)

## Not run:
ramwas1scanBams(param)
ramwas2collectqc(param)
ramwas3normalizedCoverage(param)
ramwas4PCA(param)
ramwas5MWAS(param)
ramwas6annotateTopFindings(param)
ramwas7riskScoreCV(param)
## End(Not run)
```

Description

The function `plotPrediction` plots cross validation predictions of a phenotype against true values of the phenotype with multiple summary stats in the title.
The function `plotCVcors` plots the predictive power (correlations) across predictions using various numbers of markers.

The function `plotROC` plots an ROC (Receiver operating characteristic) curve for predictions of a binary outcome.

Usage

```r
plotPrediction(
  param,
  outcome,
  forecast,
  cpgs2use,
  main,
  dfFull = NULL)
```

```r
plotCVcors(cl, param)
```

```r
plotROC(outcome, forecast)
```

Arguments

- `param` List of parameters as described in the "RW6_param.Rmd" vignette. Try: `vignette("RW6_param","ramwas")`. Only `modeloutcome`, `cvnfolds` and `mmalpha` elements are used.
- `outcome` Values of a phenotype. Must be binary for `plotROC`.
- `forecast` Predictions for the phenotype.
- `cpgs2use` Number of variables used for prediction (for the legend).
- `main` Part of the title (summary stats are added beneath).
- `dfFull` Number of degrees of freedom for the significance testing. Default is: `length(forecast) - 2`
- `cl` List with three elements:
  - `x` - vector with the number of variables used for prediction
  - `corp` - Pearson correlations between the predictions and the true value of the phenotype.
  - `cors` - Spearman correlations between the predictions and the true value of the phenotype.

Details

The `plotROC` and plot has no title.
To add a title use `title`.

Value

The `plotROC` returns the area under the curve (AUC) for the ROC.  
The `plotPrediction` function returns the list of calculated statistics printed in the title.  
The `plotCVcors` returns nothing (`NULL`).
Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

```
# Sample data
n = 300
param = list(modeloutcome = "Age", mmalpha = 0, cvnfolds = 5)
outcome = rnorm(n, mean = 50, sd = 20)
forecast = outcome + rnorm(n, mean = 0, sd = 20)
cpgs2use = 1000
main = "Prediction success (simulated data)"

# Plot phenotype-prediction plot
plotPrediction(
  param,
  outcome,
  forecast,
  cpgs2use,
  main)

# Artificial data for plotCVcors()
cl = list(
  x = c(50, 100, 200, 500, 1000),
  corp = c(0.1, 0.6, 0.7, 0.85, 0.8),
  cors = c(0.1, 0.6, 0.7, 0.85, 0.8) + rnorm(5)*0.1)

# Plot prediction performance by the number of markers
plotCVcors(cl, param)

# Make the outcome binary for ROC plot
outcome = (outcome > 50)

# Plot ROC curve and calculate the AUC
plotROC(outcome, forecast)
```

Description

RaMWAS functions for estimation and plotting of the fragment size distribution.
Usage

\[
\text{estimateFragmentSizeDistribution}(\text{frdata}, \text{seqLength})
\]

\[
\text{plotFragmentSizeDistributionEstimate(}
\begin{array}{l}
\text{frdata,} \\
\text{estimate,} \\
\text{col1 = "blue",} \\
\text{col2 = "red"}
\end{array}
\]

Arguments

- \text{frdata} \quad \text{Distribution of distances from the starts of isolated reads to the respective CpGs.}
- \text{seqLength} \quad \text{The length of sequenced part of the fragments.}
  \text{The fragments are assumed to not be smaller than seqLength.}
- \text{estimate} \quad \text{Fragment size distribution estimate.}
- \text{col1} \quad \text{Color of frdata points.}
- \text{col2} \quad \text{Color of estimate curve.}

Value

The function \text{estimateFragmentSizeDistribution} returns the estimate of the fragment size distribution.

Note

If the length of \text{frdata} is equal to \text{seqLength}, the fragments are assumed to all be of length \text{seqLength}.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

```
# Simulate data
x = 0:250
truemean = 1 - pnorm(x, mean = 150, sd = 50)
frdata = rpois(n = length(x), lambda = truemean*300)

# Estimate fragment size distribution
estimate = estimateFragmentSizeDistribution(frdata, seqLength = 50)

# Plot fragment size distribution estimate
plotFragmentSizeDistributionEstimate(frdata, estimate)
```
Description
The function plotPCvalues plots PC values (variation explained).
The function plotPCvectors plots PC vectors (loadings).

Usage
plotPCvalues(values, n = 40, ylim = NULL, col = "blue")
plotPCvectors(eigenvector, i, col = "blue1")

Arguments
- values: Vector of PC values.
- n: Number of top PCs to plot.
- ylim: Numeric vectors of length 2, giving the y coordinate range. Exactly as in Plotting Parameters.
- col: Color of the plotted points.
- eigenvector: The i-th eigenvector. See eigen.
- i: Indicates loadings of which PC to plot.

Value
This function creates a PC plot and returns nothing (NULL).

Author(s)
Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also
See vignettes: browseVignettes("ramwas").

Examples
# Sample data
# for 1000 observations and 10 samples
m = 1000
n = 10
data = matrix(rnorm(n*m), nrow = m)

covmat = crossprod(data)
e = eigen(covmat)
# Plot PC values
plotPCvalues(e$values)

# Plot PC vectors
plotPCvectors(e$vectors[,1], 1)
plotPCvectors(e$vectors[,2], 2)

## processCommandLine

### Scan Parameters From Command Line

**Description**

The pipeline parameters can be provided via command line.

For example:

```
R pipeline.r dirproject="/project" maxrepeats=0 modeloutcome="Age"
```

Each command line argument is treated as an R statement.

All variables defined this way are collected in a list which is returned.

### Usage

```
processCommandLine(.arg = NULL)
```

### Arguments

<table>
<thead>
<tr>
<th>.arg</th>
<th>Vector of command line parameters. Obtained from <code>commandArgs</code> if omitted.</th>
</tr>
</thead>
</table>

### Details

If a command line argument defines variable "fileparam", it is assumed to be a filename, and the file with this name is scanned for extra pipeline parameters, as by `parametersFromFile`.

### Value

Returns the list with all the variables set by the statement in the command line.

### Note

Variables with names starting with period (.) are ignored.

### Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

### See Also

See vignettes: `browseVignettes("ramwas")`. 
pvalue2qvalue

Examples

```r
filename = tempfile()

# Assume command line with two components:
# dirproject="."
# modelcovariates=c("Age","Sex")

arg = c(
  "dirproject = ".",
  "modelcovariates = c("Age","Sex")"
)

# Process the command line
param = processCommandLine(arg)

# Show the list
print(param)
```

---

Calculate Benjamini-Hochberg q-values

Description

Calculate Benjamini-Hochberg q-values for a vector of p-values.

Usage

```r
pvalue2qvalue(pv, n = length(pv))
```

Arguments

- `pv` Vector of p-values.
- `n` If `pv` has only top p-values from a bigger set, `n` should indicate the number of tests performed.

Details

The q-values can be slightly conservative compared to other popular q-value calculation methods.

Value

Return a vector of q-values matching p-values in `pv`.

Note

The function runs faster if the vector `pv` is sorted.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>
Examples

```r
pv = runif(20)^2
qv = pvalue2qvalue(pv)
```

### Quality Control Measures

RaMWAS calculates a number of QC measures for each BAM and saves them in R .rds files.

For full description of the QC measures and the plotting options run `vignette("RW3_BAM_QCs")`

#### Usage

```r
qcmean(x)
## S3 method for class 'NULL'
qcmean(x)
## S3 method for class 'qcChrX'
qcmean(x)
## S3 method for class 'qcChrY'
qcmean(x)
## S3 method for class 'qcCoverageByDensity'
qcmean(x)
## S3 method for class 'qcEditDist'
qcmean(x)
## S3 method for class 'qcEditDistBF'
qcmean(x)
## S3 method for class 'qcFrwrev'
qcmean(x)
## S3 method for class 'qcHistScore'
qcmean(x)
## S3 method for class 'qcHistScoreBF'
qcmean(x)
## S3 method for class 'qcIsoDist'
qcmean(x)
## S3 method for class 'qcLengthMatched'
qcmean(x)
## S3 method for class 'qcLengthMatchedBF'
qcmean(x)
## S3 method for class 'qcNonCpGreads'
qcmean(x)
```

```r
plot(x, samplename="", xstep = 25, ...)
```
## S3 method for class 'qcHistScoreBF'
plot(x, samplename="", xstep = 25, ...)

## S3 method for class 'qcEditDist'
plot(x, samplename="", xstep = 5, ...)

## S3 method for class 'qcEditDistBF'
plot(x, samplename="", xstep = 5, ...)

## S3 method for class 'qcLengthMatched'
plot(x, samplename="", xstep = 25, ...)

## S3 method for class 'qcLengthMatchedBF'
plot(x, samplename="", xstep = 25, ...)

## S3 method for class 'qcIsoDist'
plot(x, samplename="", xstep = 25, ...)

## S3 method for class 'qcCoverageByDensity'
plot(x, samplename="", ...)

### Arguments

- **x**: The QC object. See the examples below.
- **samplename**: Name of the sample for plot title.
- **xstep**: The distance between x axis ticks.
- **...**: Parameters passed to the underlying `plot` or `barplot` function.

### Value

Function `qcmean` returns one value summary of most QC measures. Run `vignette("RW3_BAM_QCs")` for description of values returned by it.

### Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

### See Also

See vignettes: `browseVignettes("ramwas").`

### Examples

```r
# Load QC data from a sample project
filename = system.file("extdata", "bigQC.rds", package = "ramwas")
qc = readRDS(filename)$qc

## The number of BAM files
cat("N BAMs: ", qc$nbams)

## Total number of reads in the BAM file(s)
cat("Reads total: ", qc$reads.total)

## Number of reads aligned to the reference genome
cat("Reads aligned: ", qc$reads aligned, "\n")
```
cat("This is ", qc$reads.aligned / qc$reads.total * 100, 
"% of all reads", sep="")

## Number of reads that passed minimum score filter and are recorded
## Number of recorded reads aligned to
## the forward and reverse strands respectively
## Distribution of the read scores
## Distribution of the length of the aligned part of the reads
## Distribution of edit distance between
## the aligned part of the read and the reference genome
## Number of reads after removal of duplicate reads
## Number of reads away from CpGs
## Average coverage of CpGs and non-CpGs

par(mfrow = c(1,2))
plot(qc$hist.score1)
plot(qc$bf.hist.score1)

par(mfrow = c(1,2))
plot(qc$hist.length.matched)
plot(qc$bf.hist.length.matched)

par(mfrow = c(1,2))
plot(qc$hist.edit.dist1)
plot(qc$bf.hist.edit.dist1)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)

par(mfrow = c(1,2))
plot(qc$hist.no.repeats)
plot(qc$bf.hist.no.repeats)
cat("Average CpG coverage:", qc$avg.cpg.coverage, "\n")
cat("Average non-CpG coverage:", qc$avg.noncpg.coverage,"\n")
cat("Enrichment ratio:", qc$avg.cpg.coverage / qc$avg.noncpg.coverage)

## Coverage around isolated CpGs
plot(qc$hist.isolated.dist1)

## Fraction of reads from chrX and chrY
cat("ChrX reads: ", qc$chrX.count[1], " which is ",
     qcmean(qc$chrX.count)*100, "% of total", sep="", "\n")
cat("ChrX reads: ", qc$chrY.count[1], " which is ",
     qcmean(qc$chrY.count)*100, "% of total", sep="", "\n")

## Coverage vs. CpG density
cat("Highest coverage is observed at CpG density of",
     qcmean(qc$avg.coverage.by.density)^2)
plot(qc$avg.coverage.by.density)

---

**qqPlotFast**

*Fast QQ-plot for Large Number of P-values*

**Description**

Function `qqPlotFast` creates a QQ-plot with a confidence band and an estimate of inflation factor lambda. It optimized to work quickly even for tens of millions of p-values.

**Usage**

```r
qqPlotPrepare(
pvalues,
ntests = NULL,
ismlog10 = FALSE)
qqPlotFast(
x,
ntests = NULL,
ismlog10 = FALSE,
ci.level = 0.05,
ylim = NULL,
newplot = TRUE,
col = "#D94D4C",
cex = 0.5,
yaxmax = NULL,
lwd = 3,
axistep = 2,
col.band = "#ECA538",
makelegend = TRUE,
xlab = expression(paste("\u2013", " log"[10]*"(" , italic("P") , "), null")),
ylab = expression(paste("\u2013", " log"[10]*"(" , italic("P") , "), observed")))
```
Arguments

- **pvalues**: Vector of p-values. As is (if ismlog10 = FALSE) or minus log10 transformed (if ismlog10 = TRUE).
- **ntests**: If only significant p-values are provided, the total number of tests performed. By default ntests is equal to the length of pvalues.
- **ismlog10**: Specifies whether the provided p-values (pvalues parameter) are minus log10 transformed (−log10(pv))
- **x**: Either a vector of p-values, as in qqPlotPrepare, or the object returned by qqPlotPrepare.
- **ci.level**: Significance level of the confidence band. Set to NULL avoid plotting the confidence band.
- **ylim**: Numeric vectors of length 2, giving the y coordinate range. Exactly as in Plotting Parameters.
- **newplot**: If TRUE, the function creates a new plot window.
- **col**: The QQ-plot curve color.
- **col.band**: Confidence band curve color.
- **cex**: The size of QQ-plot points. As in Graphics Parameters.
- **lwd**: The line width. As in Graphics Parameters.
- **axistep**: Distance between axis label ticks for both axis.
- **yaxmax**: Maximum reach of the y axis.
- **makelegend**: If true, add legend to the plot.
- **xlab, ylab**: Axis labels. As in plot function.

Details

The function qqPlotFast creates a QQ-plot.

The function qqPlotPrepare extracts the necessary information from a vector of p-values sufficient for creating QQ-plot.

The resulting object is many times smaller than the vector of p-values.

Value

The function qqPlotPrepare returns an object with the necessary information from a vector of p-values sufficient for creating QQ-plot.

Note

The plot has no title. To add a title use title.

Note

The function works faster if the p-values are sorted.
Description

Creates a set of artificial BAM files and supplementary files which can be used to test run the pipeline. The BAMs contain reads aligned only to one human chromosome, with methylation effects embedded for simulated age and case-control status.

Usage

\[
\text{ramwas0createArtificialData}(\text{dir}, \\
\quad \text{nsamples = 20,} \\
\quad \text{nreads = 1e6,})
\]
ncpgs = 500e3,
randseed = 18090212,
threads = 1)

Arguments

- **dir**: Directory for generated RaMWAS project files and BAMs.
- **nsamples**: Number of samples/BAMs to create.
- **nreads**: Number of reads in each BAM file.
- **ncpgs**: Number of CpGs in the generated genome (with a single chromosome).
- **randseed**: Random number generator seed for consistency of the output.
- **threads**: Number of CPU cores to use for data generation.

Details

The function generates a number of files within `dir` directory.

1. `bam_list.txt` - list of created BAM files. To be used in `filebamlist` and `filebam2sample` parameters in the pipeline.
2. `covariates.txt` - table with age and sex status covariates. For use in `filecovariates` parameter in the pipeline.
3. `Single_chromosome.rds` - CpG location file with the selected chromosome only.
4. `bams` - directory with all the BAM files.

The generated BAMs have 600 CpGs affected by `sex`, namely fully methylated or not methylated at all, depending on `sex`. The methylation level of 1% of all CpGs is affected by age. The methylation of those CpGs is equal to `age/100` or `1-age/100`. The age is generated randomly in the range from 20 to 80.

Value

The function creates multiple files but returns no value.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: `browseVignettes("ramwas")`.

Examples

```r
### Location for the artificial project
dr = paste0(tempdir(), "/simulated_project")

ramwas0createArtificialData(
  dr,
  nsamples = 4,
```
ramwasAnnotateLocations

Extract Biomart Annotation for a Vector of Locations.

Description

Calls biomart annotation database for a vector of locations and assignes the tracks to the locations.

Usage

ramwasAnnotateLocations(param, chr, pos)

Arguments

param List of parameters as described in the "RW6_param.Rmd" vignette. Try: vignette("RW6_param","ramwas").
chr A vector of chromosome names or numbers.
pos A vector of genomic locations on the chromosomes.

Details

This function is used internally by RaMWAS annotation step.

Value

An annotation table, on line per supplied location.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas")
Examples

```r
bihost = "grch37.ensembl.org"
bimart = "ENSEMBL_MART_ENSEMBL"
bidataset = "hsapiens_gene_ensembl"
biatributes = c("hgnc_symbol", "entrezgene", "strand")
bifilters = list(with_hgnc_trans_name = TRUE)
biflank = 0

param = ramwasParameters(
  bihost = bihost,
  bimart = bimart,
  bidataset = bidataset,
  biatributes = biatributes,
  bifilters = bifilters,
  biflank = biflank)

# Test a location
chr = "chr1"
pos = 15975530
## Not run:
ramwasAnnotateLocations(param, chr, pos)
## End(Not run)
```

---

**Description**

RaMWAS parameter vector which is used by major functions of the pipeline is a regular R list and setting it does not require a special function. However, using this function makes it much simpler in RStudio as the names and role of every parameter is showed in the RStudio IDE.

**Usage**

```r
ramwasParameters(
  dirproject,
  dirfilter,
  dirrbam,
  dirrqc,
  dirqc,
  dircoveragenorm,
  dirtemp,
  dirpca,
  dirmwas,
  dircv,
  dirbam,
  filebamlist,
```
ramwasParameters

bamnames, filebam2sample, bam2sample, filecpgset, filenoncpgset, filecovariates, covariates, cputhreads, diskthreads, usefilelock, scoretag, minscore, maxrepeats, minavgcpgcoverage, minnonzerosamples, buffersize, doublesize, modelcovariates, modeloutcome, modelPCs, modelhasconstant, qqplottitle, toppvthreshold, mmncpgs, mmalpha, cvnfolds, bihost, bimart, bidataset, biattributes, bifilters, biflank, fileSNPs, dirSNPs, ...

Arguments

**dirproject**  
The project directory. Default is current directory. Files specified by "file*" parameters are looked for here, unless they have full path specified.

**dirfilter**  
By default, the same as "dirproject". All files created by RaMWAS are created within this directory. If the user wants to test different read filtering rules, they can dirfilter to TRUE. This will set it to something like "Filter_MAPQ_4", where "MAPQ" is the BAM field used for filtering and "4" is the threshold.

**dirrbam**  
Directory where RaMWAS saves RaMWAS raw data files (read start locations) after scanning BAMs.
ramwasParameters

It is "rds_rbam" by default and located in "dirfilter".

**dirrqc**
Directory where RaMWAS saves QC files in R format after scanning BAMs.
It is "rds_qc" by default and located in "dirfilter".

**dirqc**
Directory where RaMWAS saves QC plots and text files (BAM QC info) after
scanning BAMs.
It is "qc" by default and located in "dirfilter".

**dircoveragenorm**
Directory where RaMWAS saves coverage matrix at Step 3 of the pipeline.
It is "coverage_norm_123" by default (123 is the number of samples) and lo-
cated in "dirfilter".

**dirtemp**
Directory where RaMWAS stores temporary files during construction of cover-
age matrix at Step 3 of the pipeline.
It is "temp" by default and located in "dircoveragenorm".
For better performance it can be set to a location on a different hard drive than
"dircoveragenorm".

**dirpca**
Directory where RaMWAS saves results of PCA analysis at Step 4 of the pipeline.
It is "PCA_12_cvrts_0b0a0c" by default and located in "dircoveragenorm", where
12 is the number of covariates regressed out and "0b0a0c" is a unique code to
differenciate different sets of 12 covariates.

**dirmwas**
Directory where RaMWAS saves results of MWAS analysis at Step 5 of the
pipeline.
It is "Testing_age_7_PCs" by default and located in "dirpca", where "age" is the
phenotype being tested and "7" is number of top PCs included in the model.

**dircv**
Directory where RaMWAS saves results of Methylation Risk Score analysis at
Step 7 of the pipeline.
It is "CV_10_folds" by default and located in "dirmwas", where 10 is number of
folds in N-fold cross validation.

**dirbam**
Location of BAM files.
If not absolute, it is considered to be relative to "dirproject".

**filebamlist**
If defined, must point to a text file with one BAM file name per line.
BAM file names may include path, relative to "dirbam" or absolute.

**bamnames**
A character vector with BAM file names.
Not required if "filebamlist" is specified.
BAM file names may include path, relative to "dirbam" or absolute.

**filebam2sample**
Allowes multiple BAMs contain information about common sample.
Must point to a file with lines like "sample1=bam1,bam2,bam3".

**bam2sample**
Allowes multiple BAMs contain information about common sample.
Not required if "filebam2sample" is specified.
Must be a list like list(sample1 = c("bam1","bam2","bam3"), sample2 =
"bam2")

**filecpgset**
Name of the file storing a set of CpGs.

**filenoncpgset**
If defined, must point to a file storing vetted locations away from any CpGs.
filecovariates
Name of the file containing phenotype and covariates for the available samples.
If the file has extension ".*csv", it is assumed to be comma separated, otherwise - tab separated.

covariates
Data frame with phenotype and covariates for the available samples.
Not required if "filecovariates" is specified.

cputhreads
Maximum number of CPU intensive tasks running in parallel.
Set to the number of CPU cores by default.

diskthreads
Maximum number of disk intensive tasks running in parallel.
Set to 2 by default.

usefilelock
If TRUE, parallel jobs are prevented from simultaneous access to file matrices.
Can improve performance on some systems.

scoretag
Reads from BAM files are filtered by this tag.
The "minscore" parameter defines the minimum admissible score.

minscore
Reads from BAM files with score "scoretag" below this are excluded.

maxrepeats
Duplicate reads (reads with the same start position and direction) in excess of this limit are removed.

minavgcpgcoverage
CpGs with average coverage below this threshold are removed.

minnonzerosamples
CpGs with fraction of samples with non-zero coverage below this threshold are removed.

buffersize
Coverage matrix transposition is performed using buffers of this size.
Larger "buffersize" improves speed of Step 3 of the pipeline, but requires more memory.
Default is 1e9, i.e. 1 GB.

doublesize
The coverage matrix is stored with this number of bytes per value.
Set to 8 for full (double) precision.
Set to 4 to use single precision and create 50% smaller coverage filematrix.

modelcovariates
Names of covariates included in PCA and MWAS.

modeloutcome
Name of the outcome variable for MWAS.

modelPCs
Number of principal components accounted for in MWAS.

modelhasconstant
By default, the tested linear model includes a constant.
To exclude it, set "modelhasconstant" parameter to FALSE.

qqplottitle
The title of the QQ-plot produced by MWAS (step 4 of the pipeline).

toppvthreshold
Determines the number of top MWAS results saved in text file.
If it is 1 or smaller, it defines the p-value threshold.
If larger than 1, it defines the exact number of top results.

mmncpgs
Parameter for multi-marker elastic net cross validation (MRS).
Defines the number of top CpGs on which to train the elastic net.
Can be set of a vector of multiple values, each is tested separately.
ramwasParameters

**mmalpha** Parameter for multi-marker elastic net validation (MRS). Elastic net mixing parameter alpha. Set to 0 by default.

**cvnfolds** Parameter for multi-marker elastic net validation (MRS). The number of folds in the N-fold cross validation.

**bihost** Parameter for BiomaRt annotation (Step 6 of the pipeline). BioMart host site. Set to "grch37.ensembl.org" by default.

**bimart** Parameter for BiomaRt annotation (Step 6 of the pipeline). BioMart database name, see listMarts. Set to "ENSEMBL_MART_ENSEMBL" by default.

**bidataset** Parameter for BiomaRt annotation (Step 6 of the pipeline). BioMart data set, see listDatasets. Set to "hsapiens_gene_ensembl" by default.

**biattributes** Parameter for BiomaRt annotation (Step 6 of the pipeline). BioMart attributes of interest, see listAttributes. Set to c("hgnc_symbol","entrezgene","strand") by default.

**bifilters** Parameter for BiomaRt annotation (Step 6 of the pipeline). BioMart filters (if any), see listFilters. Set to list(with_hgnc_transcript_name=TRUE) by default ignore genes without names.

**biflank** Parameter for BiomaRt annotation (Step 6 of the pipeline). Allowed distance between CpGs and genes or other annotation track elements. Set to 0 by default, requiring direct overlap.

**fileSNPs** Name of the filematrix with genotype (SNP) data. The filematrix dimensions must match the coverage matrix.

**dirSNPs** Directory where RaMWAS saves the results of joint methylation-genotype analysis.

... Any other named parameters can be added here.

**Details**

The function simply collects all the parameters in a list. The main benefit of the function is that the user does not need to memorize the names of RaMWAS parameters.

![Image of RStudio interface](image.png)

Here is how it helps in RStudio:

**Value**

List with provided parameters.
Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

ramwasParameters(dirproject = ".", cputhreads = 4)

rowcolSumSq  Form Row and Column Sums of Squares

Description

Form row and column sums of squares for numeric matrices. The functions are introduced as faster analogs of rowSums(x^2) and colSums(x^2) calls.

Usage

rowSumsSq(x)
colSumsSq(x)

Arguments

x  Numeric matrix.

Details

The function is implemented in C for better performance.

Value

Return a vector of sums of values in each row/column for matrix x (rowSumsSq/colSumsSq).

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See rowSums and colSums for simple (not squared) row/column sums.
**Examples**

```r
x = matrix(1:99, 9, 11)

# Calculate sums of squared elements in each row
rsum2 = rowSumsSq(x)

# Compare with alternative calculation
stopifnot(all.equal(rsum2, rowSums(x^2)))

# Calculate sums of squared elements in each column
csum2 = colSumsSq(x)

# Compare with alternative calculation
stopifnot(all.equal(csum2, colSums(x^2)))
```

---

**rwDataClass-class** *Class for Accessing Data (Coverage) Matrix*

**Description**

This class is a wrapper for accessing the data (coverage) matrix. It automatically subsets the samples to those listed in the covariates. Data access function imputes missing values and can residualize the variables.

**Extends**

rwDataClass is a reference classes (see envRefClass).

**Fields**

- `fmdata`: Filematrix object for the data matrix. Not intended to be accessed directly.
- `samplenames`: Vector of sample names.
- `nsamples`: Number of samples
- `ncpgs`: Number of variables (CpG sites) in the data matrix.
- `ndatarows`: Number of variables in the data matrix (may be bigger than the number of samples).
- `rowsubset`: Indices of samples in the data matrix.
- `cvrtqr`: Matrix of orthonormalized covariates.
subsetData

Methods

initialize(param = NULL, getPCs = TRUE, lockfile = NULL): Create the data access class. 'param' should contain the RaMWAS parameter vector. 'getPCs' indicates if the covariate set should include Principal components. 'lockfile' is the 'lockfile' parameter used in accessing the data filematrix.

open(param = NULL, getPCs = TRUE, lockfile = NULL): The same as 'initialize' method, but for already created object.

close(): Clears the object. Closes the filematrix.

dataGetRez(colset, resid = TRUE): Extracts data for variables indexed by 'colset'. The data is residualized unless resid = FALSE.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas")

Examples

# Create an empty rwDataClass
data = new("rwDataClass")

## Not run:
# Connect to the data
data$open(param)

# Create a rwDataClass and connect to the data
data = new("rwDataClass", param = param)

# close the object
data$close()

## End(Not run)

subsetData               Subset a data matrix and locations

Description

Subset a data (coverage) matrix and corresponding matrix of locations to a specified set of locations.

Usage

subsetCoverageDirByLocation(x, chr, start, targetdir)
testPhenotype

Argument

x Name of data (coverage) directory or list of RaMWAS parameters as described in the "RW6_param.Rmd" vignette. Try: vignette("RW6_param","ramwas").

chr Vector of chromosome names or numbers.

start Start positions of the CpGs of interest.

targetdir Directory name for the new (subset) data matrix and locations.

Value

The function returns nothing.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: browseVignettes("ramwas").

Examples

x = "/data/myCoverageMatrix"
chr = c("chr1", "chr2", "chr3")
start = c(12345, 123, 12)
targetdir = "/data/subsetCoverageMatrix"

## Not run:
subsetCoverageDirByLocation(x, chr, start, targetdir)

## End(Not run)

testPhenotype Test the Phenotype of Interest for Association with Methylation Coverage.

Description

An internal, function for fast association testing. It tests the phenotype of interest for association with methylation coverage (columns of the data parameter).

Usage

testPhenotype(phenotype, data1, cvrtqr)
testPhenotype

Arguments

- **phenotype**: Vector with phenotype. Can be numerical, character, or factor vector.
- **data1**: Matrix with data (normalized coverage), one variable (CpG) per column.
- **cvrtqr**: Orthonormalized covariates, one covariate per column. See `orthonormalizeCovariates`.

Details

The testing is performed using matrix operations and C/C++ code, employing an approach similar to that in `MatrixEQTL`.

Value

If the phenotype is numerical, the output is a list with

- **correlation**: Correlations between residualized phenotype and data columns.
- **tstat**: Corresponding T-statistics
- **pvalue**: Corresponding P-values
- **nVarTested**: Always 1
- **dfFull**: Number of degrees of freedom of the T-test

If the phenotype is a factor (or character)

- **Rsquared**: R-squared for the residualized ANOVA F-test.
- **Fstat**: Corresponding F-test
- **pvalue**: Corresponding P-values
- **nVarTested**: First number of degrees of freedom for the F-test. Equal to the number of factor levels reduced by 1
- **dfFull**: Second number of degrees of freedom for the F-test.

Note

This function is used in several parts of the pipeline.

Author(s)

Andrey A Shabalin <andrey.shabalin@gmail.com>

See Also

See vignettes: `browseVignettes("ramwas")`.
Also check `orthonormalizeCovariates`. 
Examples

### Generate data inputs

# Random data matrix with signal in the first column
 data = matrix(runif(30*5), nrow = 30, ncol = 5)
data[,1] = data[,1] + rep(0:2, each = 10)

# Two random covariates
cvrt = matrix(runif(2*30), nrow = 30, ncol = 2)
cvrtqr = orthonormalizeCovariates(cvrt)

### First, illustrate with numerical phenotype

# Numerical, 3 value phenotype
 phenotyp = rep(1:3, each = 10)

# Test for association
output = testPhenotype(phenotyp, data, cvrtqr)

# Show the results
print(output)

# Comparing with standard R code for the first variable
summary(lm( data[,1] ~ phenotype + cvrt ))

### First, illustrate with numerical phenotype

# Categorical, 3 group phenotype
 phenotyp = rep(c("Normal", "Sick", "Dead"), each = 10)

# Test for association
output = testPhenotype(phenotyp, data, cvrtqr)

# Show the results
print(output)

# Comparing with standard R code for the first variable
anova(lm( data[,1] ~ cvrt + phenotype ))
Index

* RaMWAS
  ramwas-package, 3
* Rbam
  cachedRDSload, 4
* bam
  cachedRDSload, 4
* classes
  rwDataClass-class, 44
* package
  ramwas-package, 3
* ramwas
  ramwas-package, 3

barplot, 31
BSgenome, 8
cachedRDSload, 4
colSums, 43
colSumsSq (rowcolSumSq), 43
commandArgs, 28
connection, 10
DNAString, 9, 10
eigen, 27
envRefClass, 44
estimateFragmentSizeDistribution
  (plotFragmentSizeDistributionEstimate), 25
findBestNpvs, 5
get, 6
getCpGset, 8
getCpGsetALL (getCpGset), 8
getCpGsetCG (getCpGset), 8
getDataByLocation (get), 6
getLocations (get), 6
getMWAS (get), 6
getMWASandLocations (get), 6
getMWASrange (get), 6

Graphics Parameters, 15, 34
injectSNPs, 10
injectSNPsMAF, 9
insilicoFASTQ, 10
isAbsolutePath, 11, 14
listAttributes, 42
listDatasets, 42
listFilters, 42
listMarts, 42

madeBED, 12
madeBEDgraph (madeBED), 12
madeBEDgraphRange (madeBED), 12
madeBEDrange (madeBED), 12
makefullpath, 12, 13
manhattan, 14
manPlotFast (manhattan), 14
manPlotPrepare (manhattan), 14
mat2cols, 16

order, 5
orthonormalizeCovariates, 17, 47

parameterDump, 18
parameterPreprocess, 19
parametersFromFile, 21, 28
pipeline, 22
pipelineProcessBam (pipeline), 22
plot, 31
plot function, 34
plot.qcCoverageByDensity (QCs), 30
plot.qcEditDist (QCs), 30
plot.qcEditDistBF (QCs), 30
plot.qcHistScore (QCs), 30
plot.qcHistScoreBF (QCs), 30
plot.qcIsoDist (QCs), 30
plot.qcLengthMatched (QCs), 30
plot.qcLengthMatchedBF (QCs), 30
plotCV, 23
plotCVcors (plotCV), 23
plotFragmentSizeDistributionEstimate,
   25
plotPC, 27
plotPCvalues (plotPC), 27
plotPCvectors (plotPC), 27
plotPrediction (plotCV), 23
plotROC (plotCV), 23
Plotting Parameters, 15, 27, 34
processCommandLine, 28
pvalue2qvalue, 29
qcmean (QC), 30
QC., 30
qqPlotFast, 33
qqPlotPrepare (qqPlotFast), 33
qr, 17
ramwas (ramwas-package), 3
ramwas-package, 3
ramwas0createArtificialData, 35
ramwas1scanBams (pipeline), 22
ramwas2collectqc (pipeline), 22
ramwas3normalizedCoverage (pipeline), 22
ramwas4PCA (pipeline), 22
ramwas5MWAS (pipeline), 22
ramwas6annotateTopFindings (pipeline), 22
ramwas7ArunMWASes (pipeline), 22
ramwas7BrunElasticNet (pipeline), 22
ramwas7CplotByNCpGs (pipeline), 22
ramwas7riskScoreCV (pipeline), 22
ramwasAnnotateLocations, 37
ramwasParameters, 38
ramwasSNPs (pipeline), 22
readLines, 9
readRDS, 4
rowcolSumSq, 43
rowSums, 43
rowSumsSq (rowcolSumSq), 43
rwDataClass (rwDataClass-class), 44
rwDataClass-class, 44
subsetCoverageDirByLocation
   (subsetData), 45
subsetData, 45
testPhenotype, 46
title, 15, 24, 34