Package: RAIDS (via r-universe)

November 25, 2026

Type Package

Title Robust Ancestry Inference using Data Synthesis

Description This package implements specialized algorithms that enable genetic ancestry inference from various cancer sequences sources (RNA, Exome and Whole-Genome sequences). This package also implements a simulation algorithm that generates synthetic cancer-derived data. This code and analysis pipeline was designed and developed for the following publication: Belleau, P et al. Genetic Ancestry Inference from Cancer-Derived Molecular Data across Genomic and Transcriptomic Platforms. Cancer Res 1 January 2023; 83 (1): 49–58.

Version 1.12.0

License Apache License (>= 2)

Encoding UTF-8

NeedsCompilation no

VignetteBuilder knitr

Depends R (>= 4.2.0), gdsfmt, SNPRelate, stats, utils, GENESIS, dplyr, Rsamtools, BiocParallel, genio, philentropy

Imports S4Vectors, GenomicRanges, ensembldb, BSgenome, AnnotationDbi, methods, class, BSgenome.Hsapiens.UCSC.hg38, Seqinfo, pROC, IRanges, AnnotationFilter, rlang, VariantAnnotation, MatrixGenerics, ggplot2, stringr

Suggests testthat, knitr, rmarkdown, BiocStyle, withr, EnsDb.Hsapiens.v86, GenomeInfoDb

BugReports https://github.com/KrasnitzLab/RAIDS/issues

URL https://krasnitzlab.github.io/RAIDS/

biocViews Genetics, Software, Sequencing, WholeGenome, PrincipalComponent, GeneticVariability, DimensionReduction, BiocViews

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libglpk-dev make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:51:20 UTC

RemoteUrl https://github.com/bioc/RAIDS

RemoteRef RELEASE_3_24

RemoteSha ad7070c169d53a13e29386c6fd3f25b18e711de5

Contents

add1KG2SampleGDS . . . 8
addBlockFromDetFile . . . 9
addGeneBlockGDSRefAnnot . . . 11
addGeneBlockRefAnnot . . . 13
addGenotypeProfile . . . 15
addRef2GDS1KG . . . 16
addStudy1Kg . . . 18
ancestryInferencePCAKNN . . . 20
anchorId . . . 23
anchorId,RAIDSparam-method . . . 24
anchorId<- . . . 25
anchorId<-,RAIDSparam-method . . . 26
batch . . . 26
batch,RAIDSparam-method . . . 27
batch<- . . . 28
batch<-,RAIDSparam-method . . . 29
blockTypeId . . . 29
blockTypeId,RAIDSparam-method . . . 30
blockTypeId<- . . . 31
blockTypeId<-,RAIDSparam-method . . . 32
CharacterOrNULL-class . . . 32
chrInfo . . . 33
chrInfo,RAIDSparam-method . . . 33
chrInfo<- . . . 34
chrInfo<-,RAIDSparam-method . . . 35
cleanADMIXTUREFiles . . . 36
cleanADMIXTUREFiles,RAIDSparam-method . . . 36
cleanADMIXTUREFiles<- . . . 37
cleanADMIXTUREFiles<-,RAIDSparam-method . . . 38
computeAdmixtureProportions . . . 39
computeAncestryFromSynthetic . . . 40
computeAncestryFromSynthetic2 . . . 45
computeAncestryFromSyntheticFile . . . 48
computeKNNRefSample . . . 53
computeKNNRefSyn . . . 54
computeKNNRefSynthetic . . . 56

computePCAMultiSynthetic . . . 58
computePCARefSample . . . 60
computePoolSyntheticAncestryGr . . . 62
computePoolSyntheticAncestryGr2 . . . 65
computeSyntheticROC . . . 67
createAccuracyGraph . . . 69
createAUROCGraph . . . 70
createStudy2GDS1KG . . . 71
DataFrameOrNULL-class . . . 73
demoKnownSuperPop1KG . . . 74
demoPCA1KG . . . 75
demoPCASyntheticProfiles . . . 77
demoPedigreeEx1 . . . 78
eigenCount . . . 81
eigenCount,RAIDSparam-method . . . 82
eigenCount<- . . . 82
eigenCount<-,RAIDSparam-method . . . 83
eigenCountSyn . . . 84
eigenCountSyn,RAIDSparam-method . . . 85
eigenCountSyn<- . . . 85
eigenCountSyn<-,RAIDSparam-method . . . 86
estimateAllelicFraction . . . 87
fieldPopInfAnc . . . 90
fieldPopInfAnc,RAIDSparam-method . . . 91
fieldPopInfAnc<- . . . 91
fieldPopInfAnc<-,RAIDSparam-method . . . 92
fieldPopInRef . . . 93
fieldPopInRef,RAIDSparam-method . . . 94
fieldPopInRef<- . . . 94
fieldPopInRef<-,RAIDSparam-method . . . 95
fieldSubPop . . . 96
fieldSubPop,RAIDSparam-method . . . 97
fieldSubPop<- . . . 97
fieldSubPop<-,RAIDSparam-method . . . 98
fileReferenceAnnotGDS . . . 99
fileReferenceAnnotGDS,RAIDSparam-method . . . 100
fileReferenceAnnotGDS<- . . . 100
fileReferenceAnnotGDS<-,RAIDSparam-method . . . 101
fileReferenceGDS . . . 102
fileReferenceGDS,RAIDSparam-method . . . 103
fileReferenceGDS<- . . . 103
fileReferenceGDS<-,RAIDSparam-method . . . 104
generateGDS1KG . . . 105
generateMapSnvSel . . . 107
generatePhase1KG2GDS . . . 108
generatePhaseRef . . . 110
generateVCF . . . 112
genome . . . 113

genome,RAIDSparam-method . . . 114
genoSource . . . 115
genoSource,RAIDSparam-method . . . 115
genoSource<- . . . 116
genoSource<-,RAIDSparam-method . . . 117
genoType . . . 118
genoType,RAIDSparam-method . . . 118
genoType<- . . . 119
genoType<-,RAIDSparam-method . . . 120
getMatrixPopSynthetic . . . 121
getRef1KGPop . . . 122
getRefSuperPop . . . 123
getRefSuperPop2 . . . 124
groupChr1KGSNV . . . 125
identifyRelative . . . 126
identifyRelativeRef . . . 128
inferAncestry . . . 129
inferAncestryDNA . . . 134
inferAncestryGeneAware . . . 138
inferenceType . . . 143
inferenceType,RAIDSparam-method . . . 144
inferenceType<- . . . 145
inferenceType<-,RAIDSparam-method . . . 145
kList . . . 146
kList,RAIDSparam-method . . . 147
kList<- . . . 148
kList<-,RAIDSparam-method . . . 148
listPos . . . 149
listPos,RAIDSparam-method . . . 150
listPos<- . . . 151
listPos<-,RAIDSparam-method . . . 151
matKNNSynthetic . . . 152
minCov . . . 154
minCov,RAIDSparam-method . . . 155
minCov<- . . . 155
minCov<-,RAIDSparam-method . . . 156
minProb . . . 157
minProb,RAIDSparam-method . . . 158
minProb<- . . . 159
minProb<-,RAIDSparam-method . . . 159
nbSim . . . 160
nbSim,RAIDSparam-method . . . 161
nbSim<- . . . 162
nbSim<-,RAIDSparam-method . . . 162
np . . . 163
np,RAIDSparam-method . . . 164
np<- . . . 165
np<-,RAIDSparam-method . . . 165

paramAncestry . . . 166
paramAncestry,RAIDSparam-method . . . 167
paramAncestry<- . . . 168
paramAncestry<-,RAIDSparam-method . . . 168
pathADMIXTURE . . . 169
pathADMIXTURE,RAIDSparam-method . . . 170
pathADMIXTURE<- . . . 171
pathADMIXTURE<-,RAIDSparam-method . . . 171
pathProfileGDS . . . 172
pathProfileGDS,RAIDSparam-method . . . 173
pathProfileGDS<- . . . 174
pathProfileGDS<-,RAIDSparam-method . . . 174
PCAalgorithm . . . 175
PCAalgorithm,RAIDSparam-method . . . 176
PCAalgorithm<- . . . 177
PCAalgorithm<-,RAIDSparam-method . . . 177
pcaList . . . 178
pcaList,RAIDSparam-method . . . 179
pcaList<- . . . 180
pcaList<-,RAIDSparam-method . . . 180
PCAmissingRate . . . 181
PCAmissingRate,RAIDSparam-method . . . 182
PCAmissingRate<- . . . 183
PCAmissingRate<-,RAIDSparam-method . . . 183
pedStudy . . . 184
pedStudy,RAIDSparam-method . . . 185
pedStudy<- . . . 186
pedStudy<-,RAIDSparam-method . . . 186
pedSynthetic . . . 187
phase . . . 189
phase,RAIDSparam-method . . . 190
phase<- . . . 191
phase<-,RAIDSparam-method . . . 191
phaseType . . . 192
phaseType,RAIDSparam-method . . . 193
phaseType<- . . . 194
phaseType<-,RAIDSparam-method . . . 194
posOffset . . . 195
posOffset,RAIDSparam-method . . . 196
posOffset<- . . . 197
posOffset<-,RAIDSparam-method . . . 197
pRecomb . . . 198
pRecomb,RAIDSparam-method . . . 199
pRecomb<- . . . 200
pRecomb<-,RAIDSparam-method . . . 200
prefix . . . 201
prefix,RAIDSparam-method . . . 202
prefix<- . . . 203

prefix<-,RAIDSparam-method . . . 204
prepPed1KG . . . 205
prepPedSyntheticRef . . . 206
prepPMatrix . . . 208
prepSynthetic . . . 209
profileFile . . . 211
profileFile,RAIDSparam-method . . . 212
profileFile<- . . . 213
profileFile<-,RAIDSparam-method . . . 214
profileGenoSynthesis . . . 215
pruningMethod . . . 216
pruningMethod,RAIDSparam-method . . . 217
pruningMethod<- . . . 218
pruningMethod<-,RAIDSparam-method . . . 219
pruningProfile2 . . . 220
pruningSample . . . 221
RAIDSparam . . . 224
RAIDSparam-class . . . 230
reference . . . 233
reference,RAIDSparam-method . . . 234
reference<- . . . 235
reference<-,RAIDSparam-method . . . 236
runADMIXTURE . . . 236
runExomeAncestry . . . 238
runRNAAncestry . . . 242
runSynADMIXTURE . . . 246
sampleRef . . . 247
sampleRef,RAIDSparam-method . . . 248
sampleRef<- . . . 249
sampleRef<-,RAIDSparam-method . . . 250
seedADMIXTURE . . . 251
seedADMIXTURE,RAIDSparam-method . . . 251
seedADMIXTURE<- . . . 252
seedADMIXTURE<-,RAIDSparam-method . . . 253
select1KGPop . . . 254
select1KGPopForSynthetic . . . 255
selectForSynthetic . . . 256
seqError . . . 258
seqError,RAIDSparam-method . . . 258
seqError<- . . . 259
seqError<-,RAIDSparam-method . . . 260
seqErrorSyn . . . 261
seqErrorSyn,RAIDSparam-method . . . 261
seqErrorSyn<- . . . 262
seqErrorSyn<-,RAIDSparam-method . . . 263
slideWindowMaxBP . . . 264
slideWindowMaxBP,RAIDSparam-method . . . 264
slideWindowMaxBP<- . . . 265

slideWindowMaxBP<-,RAIDSparam-method . . . 266
snpPositionDemo . . . 267
snvListVCF . . . 269
snvListVCFRef . . . 270
specificSNV . . . 271
specificSNV,RAIDSparam-method . . . 271
specificSNV<- . . . 272
specificSNV<-,RAIDSparam-method . . . 273
splitSelectByPop . . . 274
studyDF . . . 275
studyDF,RAIDSparam-method . . . 275
studyDF<- . . . 276
studyDF<-,RAIDSparam-method . . . 277
studyDFSyn . . . 278
studyDFSyn,RAIDSparam-method . . . 278
studyDFSyn<- . . . 279
studyDFSyn<-,RAIDSparam-method . . . 280
studyType . . . 281
studyType,RAIDSparam-method . . . 282
studyType<- . . . 282
studyType<-,RAIDSparam-method . . . 283
syntheticGeno . . . 284
syntheticGeno2 . . . 286
syntheticRefDF . . . 288
syntheticRefDF,RAIDSparam-method . . . 289
syntheticRefDF<- . . . 289
syntheticRefDF<-,RAIDSparam-method . . . 290
thresholdLD . . . 291
thresholdLD,RAIDSparam-method . . . 292
thresholdLD<- . . . 293
thresholdLD<-,RAIDSparam-method . . . 293
verbose . . . 294
verbose,RAIDSparam-method . . . 295
verbose<- . . . 296
verbose<-,RAIDSparam-method . . . 296
writeBedProfile . . . 297
writeBimPruned . . . 299
writeFamProfile . . . 300
writePMatrix . . . 302

Index **303**

`add1KG2SampleGDS` *Add the genotype information for the list of pruned SNVs into the Profile GDS file*

Description

The function extracts the information about the pruned SNVs from the 1KG GDS file and adds entries related to the pruned SNVs in the Profile GDS file. The nodes are added to the Profile GDS file: 'sample.id', 'snp.id', 'snp.chromosome', 'snp.position', 'snp.index', 'genotype' and 'lap'.

Usage

```
add1KG2SampleGDS(gdsReference, fileProfileGDS, currentProfile, studyID)
```

Arguments

`gdsReference`	an object of class gds.class (a GDS file), the opened 1KG GDS file.
`fileProfileGDS`	a `character` string representing the path and file name of the Profile GDS file. The Profile GDS file must exist.
`currentProfile`	a `character` string corresponding to the sample identifier associated to the current list of pruned SNVs.
`studyID`	a `character` string corresponding to the study identifier associated to the current list of pruned SNVs.

Value

The function returns `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

## The data.frame containing the information about the study
## The 3 mandatory columns: "studyID", "study.desc", "study.platform"
## The entries should be strings, not factors (stringsAsFactors=FALSE)
studyDF <- data.frame(study.id="MYDATA",
                      study.desc="Description",
                      study.platform="PLATFORM",
                      stringsAsFactors=FALSE)
```

```
## Temporary Profile file
fileProfile <- file.path(tempdir(), "ex2.gds")

## Copy required file
file.copy(file.path(dataDir, "ex1_demo_with_pruning.gds"),
        fileProfile)

## Open 1KG file
gds1KG <- snpgdsOpen(fileGDS)

## Compute the list of pruned SNVs for a specific profile 'ex1'
## and save it in the Profile GDS file 'ex2.gds'
add1KG2SampleGDS(gdsReference=gds1KG,
        fileProfileGDS=fileProfile,
        currentProfile=c("ex1"),
        studyID=studyDF$study.id)

## Close the 1KG GDS file (important)
closefn.gds(gds1KG)

## Check content of Profile GDS file
## The 'pruned.study' entry should be present
content <- openfn.gds(fileProfile)
content

## Close the Profile GDS file (important)
closefn.gds(content)

## Remove Profile GDS file (created for demo purpose)
unlink(fileProfile, force=TRUE)
```

`addBlockFromDetFile` *Append information associated to ld blocks, as indexes, into the Population Reference SNV Annotation GDS file*

Description

The function appends the information about the ld blocks into the Population Reference SNV Annotation GDS file. The information is extracted from the Population Reference GDS file and files \'.det\'.

Usage

```
addBlockFromDetFile(
  fileReferenceGDS,
  gdsRefAnnotFile,
  pathBlock,
```

```
  superPop,
  blockName = "ldBlock",
  blockDesc = "Not Define",
  verbose = FALSE
)
```

Arguments

fileReferenceGDS	a character string representing the file name of the Reference GDS file. The file must exist.
gdsRefAnnotFile	a character string representing the file name corresponding the Reference SNV Annotation GDS file. The function will open it in write mode and close it after. The file must exist.
pathBlock	a character string representing the directory where all the output file det from the plink block command are located. The directory must not include other file with the extension \'.det\'. The name of the \'.det\' must include the super-population between \'.\' and the chromosome in the form \'chrNumber.\' \(\'chr1.\'\).
superPop	a character string representing the super population.
blockName	a character string representing the id of the block. The blockName should not exist in \'gdsRefAnnotFile\'. Default: "ldBlock".
blockDesc	a character string representing the description of the block. Default: "Not Define"
verbose	a logical indicating if message information should be printed. Default: FALSE.

Details

More information about GDS file format can be found at the Bioconductor gdsfmt website: https://bioconductor.org/packages/

Value

OL when the function is successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

fileAnnotGDS <- file.path(tempdir(), "ex1_good_small_1KG_Ann_GDS.gds")

## Demo of of output file det from the plink block
## command for chromosome 1
fileLdBlock <- file.path(dirname(fileAnnotGDS), "block.sp.EUR.Ex.chr1.blocks.det")
```

```
file.copy(file.path(dataDir, "tests",
    "ex1_NoBlockGene.1KG_Annot_GDS.gds"), fileAnnotGDS)
file.copy(file.path(dataDir, "block.sp.EUR.Ex.chr1.blocks.det"),
    fileLdBlock)



## GDS Reference file
fileReferenceGDS  <- file.path(dataDir, "tests",
    "ex1_good_small_1KG.gds")




    ## Append information associated to blocks
    addBlockFromDetFile(fileReferenceGDS=fileReferenceGDS,
        gdsRefAnnotFile=fileAnnotGDS,
        pathBlock=dirname(fileAnnotGDS),
        superPop="EUR")

    gdsAnnot1KG <- openfn.gds(fileAnnotGDS)
    print(gdsAnnot1KG)

    closefn.gds(gdsAnnot1KG)


## Remove temporary file
unlink(fileAnnotGDS, force=TRUE)
unlink(fileLdBlock, force=TRUE)
```

`addGeneBlockGDSRefAnnot` *Append information associated to blocks, as indexes, into the Population Reference SNV Annotation GDS file*

Description

The function appends the information about the blocks into the Population Reference SNV Annotation GDS file. The information is extracted from the Population Reference GDS file.

Usage

```
addGeneBlockGDSRefAnnot(
  gdsReference,
  gdsRefAnnotFile,
  winSize = 10000,
```

```
  ensDb,
  suffixBlockName
)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the opened Reference GDS file.
gdsRefAnnotFile	a character string representing the file name corresponding the Reference SNV Annotation GDS file. The function will open it in write mode and close it after. The file must exist.
winSize	a single positive integer representing the size of the window to use to group the SNVs when the SNVs are in a non-coding region. Default: 10000L.
ensDb	An object with the ensembl genome annotation Default: EnsDb.Hsapiens.v86.
suffixBlockName	a character string that identify the source of the block and that will be added to the block description into the Reference SNV Annotation GDS file, as example: Ensembl.Hsapiens.v86.

Value

The integer 0L when the function is successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(SNPRelate)

## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

fileAnnotGDS <- file.path(tempdir(), "ex1_good_small_1KG_Ann_GDS.gds")

## Required library
if (requireNamespace("EnsDb.Hsapiens.v86", quietly=TRUE)) {

    file.copy(file.path(dataDir, "tests",
        "ex1_NoBlockGene.1KG_Annot_GDS.gds"), fileAnnotGDS)

    ## Making a "short cut" on the ensDb object
    edb <- EnsDb.Hsapiens.v86::EnsDb.Hsapiens.v86

    ## GDS Reference file
    fileReferenceGDS  <- file.path(dataDir, "tests",
                "ex1_good_small_1KG.gds")
```

```
## Open the reference GDS file (demo version)
gds1KG <- snpgdsOpen(fileReferenceGDS)

## Append information associated to blocks
addGeneBlockGDSRefAnnot(gdsReference=gds1KG,
    gdsRefAnnotFile=fileAnnotGDS,
    ensDb=edb,
    suffixBlockName="EnsDb.Hsapiens.v86")

gdsAnnot1KG <- openfn.gds(fileAnnotGDS)
print(gdsAnnot1KG)
print(read.gdsn(index.gdsn(gdsAnnot1KG, "block.annot")))

## Close GDS files
closefn.gds(gds1KG)
closefn.gds(gdsAnnot1KG)


## Remove temporary file
unlink(fileAnnotGDS, force=TRUE)

}
```

---

addGeneBlockRefAnnot *Append information associated to blocks, as indexes, into the Population Reference SNV Annotation GDS file*

Description

The function appends the information about the blocks into the Population Reference SNV Annotation GDS file. The information is extracted from the Population Reference GDS file.

Usage

```
addGeneBlockRefAnnot(
  fileReferenceGDS,
  gdsRefAnnotFile,
  winSize = 10000,
  ensDb,
  suffixBlockName
)
```

Arguments

fileReferenceGDS
: a `character` string representing the file name of the Reference GDS file. The file must exist.

gdsRefAnnotFile
: a character string representing the file name corresponding the Reference SNV Annotation GDS file. The function will open it in write mode and close it after. The file must exist.

winSize
: a single positive integer representing the size of the window to use to group the SNVs when the SNVs are in a non-coding region. Default: 10000L.

ensDb
: An object with the ensembl genome annotation Default: EnsDb.Hsapiens.v86.

suffixBlockName
: a character string that identify the source of the block and that will be added to the block description into the Reference SNV Annotation GDS file, as example: Ensembl.Hsapiens.v86.

Value

The integer 0L when the function is successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

fileAnnotGDS <- file.path(tempdir(), "ex1_good_small_1KG_Ann_GDS.gds")

## Required library
if (requireNamespace("EnsDb.Hsapiens.v86", quietly=TRUE)) {

    file.copy(file.path(dataDir, "tests",
        "ex1_NoBlockGene.1KG_Annot_GDS.gds"), fileAnnotGDS)

    ## Making a "short cut" on the ensDb object
    edb <- EnsDb.Hsapiens.v86::EnsDb.Hsapiens.v86

    ## GDS Reference file
    fileReferenceGDS  <- file.path(dataDir, "tests",
                "ex1_good_small_1KG.gds")




        ## Append information associated to blocks
        addGeneBlockRefAnnot(fileReferenceGDS=fileReferenceGDS,
            gdsRefAnnotFile=fileAnnotGDS,
            ensDb=edb,
            suffixBlockName="EnsDb.Hsapiens.v86")

        gdsAnnot1KG <- openfn.gds(fileAnnotGDS)
        print(gdsAnnot1KG)
```

```
        print(read.gdsn(index.gdsn(gdsAnnot1KG, "block.annot")))

      closefn.gds(gdsAnnot1KG)


    ## Remove temporary file
    unlink(fileAnnotGDS, force=TRUE)

}
```

---

addGenotypeProfile *Add the genotype information for the list of pruned SNVs into the Profile GDS file*

Description

The function extracts the information about the pruned SNVs from the 1KG GDS file and adds entries related to the pruned SNVs in the Profile GDS file. The nodes are added to the Profile GDS file: 'sample.id', 'snp.id', 'snp.chromosome', 'snp.position', 'snp.index', 'genotype' and 'lap'.

Usage

```
addGenotypeProfile(pRAIDS)
```

Arguments

pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Value

The function returns 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library for GDS
library(gdsfmt)

## Path to the demo Reference GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

## The data.frame containing the information about the study
## The 3 mandatory columns: "study.id", "study.desc", "study.platform"
## The entries should be strings, not factors (stringsAsFactors=FALSE)
```

```
studyDF <- data.frame(study.id = "MYDATA",
                      study.desc = "Description",
                      study.platform = "PLATFORM",
                      stringsAsFactors = FALSE)

## The data.frame containing the information about the samples
## The entries should be strings, not factors (stringsAsFactors=FALSE)
samplePED <- data.frame(Name.ID = c("ex1", "ex2"),
                    Case.ID = c("Patient_h11", "Patient_h12"),
                    Diagnosis = rep("Cancer", 2),
                    Sample.Type = rep("Primary Tumor", 2),
                    Source = rep("Databank B", 2), stringsAsFactors = FALSE)
rownames(samplePED) <- samplePED$Name.ID

## Temporary Profile GDS file
pathProfileGDS <- tempdir()
profileFile <- file.path(pathProfileGDS, "ex1.gds")

pRAIDS <- RAIDSparam(profileFile=file.path(dataDir, "ex1.txt.gz"),
    studyDF=studyDF,
    genoSource="snp-pileup",
    pedStudy=samplePED,
    pathProfileGDS=pathProfileGDS,
    fileReferenceGDS=fileGDS
    )
## Copy the Profile GDS file demo that has not been pruned yet
file.copy(file.path(dataDir, "ex1_demo.gds"), profileFile)


## Compute the list of pruned SNVs for a specific profile 'ex1'
## and save it in the Profile GDS file 'ex1.gds'
pRAIDS <- pruningProfile2(pRAIDS)
RAIDS::addGenotypeProfile(pRAIDS)

content <- openfn.gds(profileFile)
content

## Close the Profile GDS file (important)
closefn.gds(content)

## Remove Profile GDS file (created for demo purpose)
unlink(profileFile, force=TRUE)
```

---

addRef2GDS1KG *Add the information about the unrelated patients to the Reference GDS file*

Description

This function adds the information about the unrelated patients to the Reference GDS file. More specifically, it creates the field `sample.ref` which as the value `1` when the sample is unrelated and the value `0` otherwise. The `sample.ref` is filled based on the information present in the input RDS file.

Usage

```
addRef2GDS1KG(fileNameGDS, filePart)
```

Arguments

`fileNameGDS`	a `character` string representing the path and file name of the GDS file that contains the Reference information. The Reference GDS file must contain the SNP information, the genotyping information and the pedigree information from Reference dataset. The extension of the file must be '.gds'.
`filePart`	a `character` string representing the path and file name of the RDS file that contains the information about the Reference patients that are unrelated. The extension of the file must be '.rds'. The file must exists.

Value

The integer `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Locate RDS with unrelated/related status for 1KG samples
dataDir <- system.file("extdata", package="RAIDS")
rdsFilePath <- file.path(dataDir, "unrelatedPatientsInfo_Demo.rds")

## Create a temporary GDS file in an test directory
dataDir <- system.file("extdata/tests", package="RAIDS")
gdsFilePath <- file.path(dataDir, "GDS_TEMP_201.gds")

## Create and open the GDS file
tmpGDS  <- createfn.gds(filename=gdsFilePath)
## Create "sample.id" node (the node must be present)
sampleIDs <- c("HG00104", "HG00109", "HG00110")
add.gdsn(node=tmpGDS, name="sample.id", val=sampleIDs)

## Create "snp.id" node (the node must be present)
snpIDs <- c("s1", "s2", "s3", "s4", "s5", "s6")
add.gdsn(node=tmpGDS, name="snp.id", val=snpIDs)

## Create "snp.position" node (the node must be present)
snpPositions <- c(16102, 51478, 51897, 51927, 54489, 54707)
```

```
add.gdsn(node=tmpGDS, name="snp.position", val=snpPositions)

## Create "snp.chromosome" node (the node must be present)
snpPositions <- c(1, 1, 1, 1, 1, 1)
add.gdsn(node=tmpGDS, name="snp.chromosome", val=snpPositions)

## Create "genotype" node (the node must be present)
genotype <- matrix(rep(1, 18), ncol = 3)
add.gdsn(node=tmpGDS, name="genotype", val=genotype)

## Close GDS file
closefn.gds(tmpGDS)

## Create  "sample.ref" node in GDS file using RDS information
addRef2GDS1KG(fileNameGDS=gdsFilePath, filePart=rdsFilePath)

## Read sample reference data.frame
fileGDS <- openfn.gds(gdsFilePath, readonly=TRUE)
read.gdsn(index.gdsn(node=fileGDS, path="sample.ref"))
closefn.gds(gdsfile=fileGDS)

## Delete the temporary GDS file
unlink(x=gdsFilePath, force=TRUE)
```

---

addStudy1Kg *Append information about the 1KG samples into the Profile GDS file*

Description

The information about the samples present in the 1KG GDS file is added into the GDS Sample file. Only the information about the unrelated samples from the 1OOO Genome Study are copied into the GDS Sample file. The information is only added to the GDS Sample file when the 1KG Study is not already present in the GDS Sample file. The sample information for all selected samples is appended to the GDS Sample file "study.annot" node. The study information is appended to the GDS Sample file "study.list" node.

Usage

```
addStudy1Kg(gdsReference, fileProfileGDS, verbose = FALSE)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the opened 1KG GDS file.
fileProfileGDS	a character string representing the path and file name of the GDS Sample file. The GDS Sample file must exist.
verbose	a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Value

The integer `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library for GDS
library(gdsfmt)

## Get the temp folder
tempDir <- tempdir()

## Create a temporary 1KG GDS file and add needed information
fileName1KG <- file.path(tempDir, "GDS_TEMP_addStudy1Kg_1KG.gds")
gds1KG  <- createfn.gds(filename=fileName1KG)
add.gdsn(gds1KG, "sample.id", c("HTT101", "HTT102", "HTT103"))

samples <- data.frame(sex=c(1, 1, 2), pop.group=c("GBR", "GIH", "GBR"),
    superPop=c("EUR", "SAS", "EUR"), batch=rep(0, 3),
    stringsAsFactors = FALSE)

add.gdsn(gds1KG, "sample.annot", samples)
add.gdsn(gds1KG, "sample.ref", c(1,0, 1))
sync.gds(gds1KG)

## Create a temporary Profile GDS file
fileNameProfile <- file.path(tempDir, "GDS_TEMP_addStudy1Kg_Sample.gds")
gdsProfile <- createfn.gds(fileNameProfile)

study.list <- data.frame(study.id=c("HTT Study"),
    study.desc=c("Important Study"),
    study.platform=c("Panel"), stringsAsFactors=FALSE)

add.gdsn(gdsProfile, "study.list", study.list)

study.annot <- data.frame(data.id=c("TOTO1"), case.id=c("TOTO1"),
                sample.type=c("Study"), diagnosis=c("Study"),
                source=rep("IGSR"), study.id=c("Study"),
                stringsAsFactors=FALSE)

add.gdsn(gdsProfile, "study.annot", study.annot)
sync.gds(gdsProfile)
closefn.gds(gdsProfile)

## Append information about the 1KG samples into the Profile GDS file
## The Profile GDS file will contain 'study.list' and 'study.annot' entries
addStudy1Kg(gdsReference=gds1KG, fileProfileGDS=fileNameProfile,
    verbose=TRUE)
```

```
closefn.gds(gds1KG)
unlink(fileNameProfile, recursive=TRUE, force=TRUE)
unlink(fileName1KG, recursive=TRUE, force=TRUE)
unlink(tempDir)
```

`ancestryInferencePCAKNN` *Run most steps leading to the ancestry inference call based on PCA and KNN on a specific profile*

Description

This function runs most steps leading to the ancestry inference call on a specific profile. First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population Reference GDS file.

Usage

```
ancestryInferencePCAKNN(pRAIDS = RAIDSparam())
```

Arguments

`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters

Details

The runExomeAncestry() function generates 3 types of files in the OUTPUT directory.

Ancestry Inference The ancestry inference CSV file (".Ancestry.csv" file)

Inference Informaton The inference information RDS file (".infoCall.rds" file)

Synthetic Information The parameter information RDS files from the synthetic inference ("KNN.synt.*.rds" files in a sub-directory)

In addition, a sub-directory (named using the profile ID) is also created.

Value

a `list` containing 4 entries:

`pcaSample` a `list` containing the information related to the eigenvectors. The `list` contains those 3 entries:

- `sample.id` a `character` string representing the unique identifier of the current profile.
- `eigenvector.ref` a `matrix` of `numeric` containing the eigenvectors for the reference profiles.
- `eigenvector` a `matrix` of `numeric` containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

paraSample a list containing the results with different D and K values that lead to optimal parameter selection. The list contains those entries:

- dfPCA a data.frame containing statistical results on all combined synthetic results done with a fixed value of D (the number of dimensions). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - median a numeric representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - mad a numeric representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - upQuartile a numeric representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - k a numeric representing the optimal K value (the number of neighbors) for a fixed D value.
- dfPop a data.frame containing statistical results on all combined synthetic results done with different values of D (the number of dimensions) and K (the number of neighbors). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - K a numeric representing the value of K (the number of neighbors).
  - AUROC.min a numeric representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of D and K.
  - AUROC a numeric representing the accuracy obtained by grouping all the synthetic results for the specified values of D and K.
  - Accu.CM a numeric representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of D and K.
- dfAUROC a data.frame the summary of the results by super-population. The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - K a numeric representing the value of K (the number of neighbors).
  - Call a character string representing the super-population.
  - L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.
  - AUROC a numeric representing the AUROC obtained for the fixed values of super-population, D and K.
  - H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.
- D a numeric representing the optimal D value (the number of dimensions) for the specific profile.
- K a numeric representing the optimal K value (the number of neighbors) for the specific profile.
- listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

- sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

KNNSynthetic a data.frame containing the inferred ancestry for each synthetic data for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current synthetic data.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

infer.superPop a character string representing the inferred ancestry for the specified D and K values.

ref.superPop a character string representing the known ancestry from the reference

Ancestry a data.frame containing the inferred ancestry for the current profile. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry.

pRAIDS a list of class RAIDSparam a list of the parameters.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")


#################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")
```

```
######################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
######################################################################
demoProfileEx1 <- file.path(dataDir, "example", "snpPileup", "ex1.txt.gz")

######################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
######################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathProfileGDS)){
    dir.create(pathProfileGDS)
}
######################################################################
## Fix seed to ensure reproducible results
######################################################################
set.seed(3043)

pRAIDS <- RAIDS:::RAIDSparam(profileFile=demoProfileEx1,
    genoSource="snp-pileup",
    pathProfileGDS=pathProfileGDS,
    fileReferenceGDS=fileReferenceGDS,
    fileReferenceAnnotGDS=fileAnnotGDS
    )



res <- ancestryInferencePCAKNN(pRAIDS=pRAIDS)

res$Ancestry

unlink(file.path(pathProfileGDS, paste0(pedStudy(res$pRAIDS)$Name.ID[1], ".gds")) , force=TRUE)
if (length(dir(pathProfileGDS)) == 0) {
unlink(pathProfileGDS, recursive = TRUE)
}
```

---

anchorId *Generic function for getting the anchorId slot in a class*

Description

A generic function for getting the anchorId slot in a S4 object.

Usage

```
anchorId(x)
```

Arguments

x	a S4 object.

Value

a value from the anchorId slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(anchorId="character"))

# Create an instance
obj <- new("MyClass", anchorId="123")

# Call the generic (assuming a method is implemented)
# anchorId(obj)
```

---

anchorId,RAIDSparam-method

A getter for the anchorId slot in a RAIDSparam class

Description

A function for getting the anchorId slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
anchorId(x)
```

Arguments

x	a RAIDSparam object.

Value

a character string representing the name of the identifier of the quasi non-admixed profile from the reference.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the anchorId slot for the object
anchorId(paramDemo)
```

---

anchorId<- *Generic function for replacement of anchorId slot in a class*

Description

A generic function for replacement of anchorId slot in a S4 object.

Usage

```
anchorId(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(anchorId="character"))

# Create an instance
obj <- new("MyClass", anchorId="123")

# Call the generic (assuming a method is implemented)
# anchorId(obj) <- "333"
```

anchorId<-,RAIDSparam-method *Setter function for replacement of anchorId slot in a RAIDSparam class*

Description

A function for replacement of anchorId slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
anchorId(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a character string representing the name of the identifier of the quasi non-admixed profile from the reference.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the anchorId slot in the object
anchorId(paramDemo) <-  "sp0.95vT"
```

batch *Generic function for getting the batch slot in a class*

Description

A generic function for getting the batch slot in a S4 object.

Usage

```
batch(x)
```

Arguments

x	a S4 object.

Value

a value from the batch slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(batch="character"))

# Create an instance
obj <- new("MyClass", batch="123")

# Call the generic (assuming a method is implemented)
# batch(obj)
```

---

batch,RAIDSparam-method

A getter for the batch slot in a RAIDSparam class

Description

A function for getting the batch slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
batch(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive integer representing the current identifier for the batch.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the batch slot for the object
batch(paramDemo)
```

---

batch<- *Generic function for replacement of batch slot in a class*

Description

A generic function for replacement of batch slot in a S4 object.

Usage

```
batch(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(batch="character"))

# Create an instance
obj <- new("MyClass", batch="123")

# Call the generic (assuming a method is implemented)
# batch(obj) <- "333"
```

batch<-,RAIDSparam-method *Setter function for replacement of batch slot in a RAIDSparam class*

Description

A function for replacement of batch slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
batch(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single positive integer representing the current identifier for the batch.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the batch slot in the object
batch(paramDemo) <- 34L
```

blockTypeId *Generic function for getting the blockTypeId slot in a class*

Description

A generic function for getting the blockTypeId slot in a S4 object.

Usage

```
blockTypeId(x)
```

Arguments

x	a S4 object.

Value

a value from the blockTypeId slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(blockTypeId="character"))

# Create an instance
obj <- new("MyClass", blockTypeId="123")

# Call the generic (assuming a method is implemented)
# blockTypeId(obj)
```

---

blockTypeId,RAIDSparam-method

A getter for the blockTypeId slot in a RAIDSparam class

Description

A function for getting the blockTypeId slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
blockTypeId(x)
```

Arguments

x	a RAIDSparam object.

Value

a single character string corresponding to the block type used to extract the block identifiers.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the blockTypeId slot for the object
blockTypeId(paramDemo)
```

`blockTypeId<-` *Generic function for replacement of blockTypeId slot in a class*

Description

A generic function for replacement of blockTypeId slot in a S4 object.

Usage

```
blockTypeId(x) <- value
```

Arguments

`x`	a S4 object.
`value`	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(blockTypeId="character"))

# Create an instance
obj <- new("MyClass", blockTypeId="123")

# Call the generic (assuming a method is implemented)
# blockTypeId(obj) <- "333"
```

blockTypeId<-,RAIDSparam-method *Setter function for replacement of blockTypeId slot in a RAIDSparam class*

Description

A function for replacement of blockTypeId slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
blockTypeId(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single character string corresponding to the block type used to extract the block identifiers. The block type must be present in the GDS Reference Annotation file.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the new blockTypeId to the blockTypeId slot in the object
blockTypeId(paramDemo) <- "TEST"
```

CharacterOrNULL-class *Class Union for character or NULL*

Description

A virtual class that groups character and NULL together.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

chrInfo *Generic function for getting the chrInfo slot in a class*

Description

A generic function for getting the chrInfo slot in a S4 object.

Usage

```
chrInfo(x)
```

Arguments

x	a S4 object.

Value

a value from the chrInfo slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(chrInfo="character"))

# Create an instance
obj <- new("MyClass", chrInfo="123")

# Call the generic (assuming a method is implemented)
# chrInfo(obj)
```

---

chrInfo,RAIDSparam-method
A getter for the chrInfo slot in a RAIDSparam class

Description

A function for getting the chrInfo slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
chrInfo(x)
```

Arguments

x	a RAIDSparam object.

Value

a vector of positive integer values representing the length of the chromosomes.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the chrInfo slot for the object
chrInfo(paramDemo)
```

---

chrInfo<- *Generic function for replacement of chrInfo slot in a class*

Description

A generic function for replacement of chrInfo slot in a S4 object.

Usage

```
chrInfo(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(chrInfo="character"))

# Create an instance
obj <- new("MyClass", chrInfo="123")

# Call the generic (assuming a method is implemented)
# chrInfo(obj) <- "333"
```

---

chrInfo<-,RAIDSparam-method

Setter function for replacement of chrInfo slot in a RAIDSparam class

---

Description

A function for replacement of chrInfo slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
chrInfo(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a `vector` of positive `integer` values representing the length of the chromosomes.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the new chrInfo to the chrInfo slot in the object
chrInfo(paramDemo) <- c(122222L, 333333L)
```

`cleanADMIXTUREFiles` *Generic function for getting the cleanADMIXTUREFiles slot in a class*

Description

A generic function for getting the cleanADMIXTUREFiles slot in a S4 object.

Usage

```
cleanADMIXTUREFiles(x)
```

Arguments

x	a S4 object.

Value

a value from the cleanADMIXTUREFiles slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(cleanADMIXTUREFiles="character"))

# Create an instance
obj <- new("MyClass", cleanADMIXTUREFiles="123")

# Call the generic (assuming a method is implemented)
# cleanADMIXTUREFiles(obj)
```

`cleanADMIXTUREFiles,RAIDSparam-method`
A getter for the cleanADMIXTUREFiles slot in a RAIDSparam class

Description

A function for getting the cleanADMIXTUREFiles slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
cleanADMIXTUREFiles(x)
```

Arguments

x	a RAIDSparam object.

Value

a single logical indicating if message information should be printed.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the cleanADMIXTUREFiles slot for the object
cleanADMIXTUREFiles(paramDemo)
```

---

cleanADMIXTUREFiles<- *Generic function for replacement of cleanADMIXTUREFiles slot in a class*

Description

A generic function for replacement of cleanADMIXTUREFiles slot in a S4 object.

Usage

```
cleanADMIXTUREFiles(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(cleanADMIXTUREFiles="character"))

# Create an instance
obj <- new("MyClass", cleanADMIXTUREFiles="123")

# Call the generic (assuming a method is implemented)
# cleanADMIXTUREFiles(obj) <- TRUE
```

---

cleanADMIXTUREFiles<-,RAIDSparam-method

Setter function for replacement of cleanADMIXTUREFiles slot in a RAIDSparam class

---

Description

A function for replacement of cleanADMIXTUREFiles slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
cleanADMIXTUREFiles(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `logical` indicating if message information should be printed.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the cleanADMIXTUREFiles slot in the object
cleanADMIXTUREFiles(paramDemo) <-  TRUE
```

computeAdmixtureProportions
compute admixture proportion

Description

TODO

Usage

```
computeAdmixtureProportions(pRAIDS)
```

Arguments

pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Details

TODO

Value

a list containing 2 entries:

profileQ a matrix containing the admixture proportions of the profile

syntheticQ a matrix containing for each synthetic profile the admixture proportions of the synthetic profile, the synthetic proportions of the reference using the same position as the profile, and the admixture proportion of the gold standard

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Not run:
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

pathOut <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathOut)){
    dir.create(pathOut)
}

# The name of the synthetic study
studyID <- "MYDATA"
```

```
studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(pathOut) # , "ex1.gds"
file.copy(file.path(dataDir, "ex1.gds"), file.path(pathOut))
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID
## Path to admixture
## pathADMIXTURE <- "PATH/admixture"

pRAIDS <- RAIDSparam(studyDF=studyDF,
                      pedStudy=pedStudy,
                      pathProfileGDS=pathProfileGDS,
                      fileReferenceGDS=fileReferenceGDS,
                      fieldPopInfAnc="SuperPop",
                      pathADMIXTURE=pathADMIXTURE)
res <- computeAdmixtureProportions(pRAIDS=pRAIDS)



dir(file.path(pathProfileGDS))
head(res$profileQ)
head(res$syntheticQ)

unlink(file.path(pathOut), recursive = TRUE)#'

## End(Not run)
```

`computeAncestryFromSynthetic` *Select the optimal K and D parameters using the synthetic data and infer the ancestry of a specific profile*

Description

The function select the optimal K and D parameters for a specific profile. The results on the synthetic data are used for the parameter selection. Once the optimal parameters are selected, the ancestry is inferred for the specific profile.

Usage

```
computeAncestryFromSynthetic(
```

```
  gdsReference,
  gdsProfile,
  syntheticKNN,
  pedSyn,
  currentProfile,
  spRef,
  studyIDSyn,
  np = 1L,
  listCatPop = c("EAS", "EUR", "AFR", "AMR", "SAS"),
  fieldPopInRef = "superPop",
  fieldPopInfAnc = "SuperPop",
  kList = seq(2, 15, 1),
  pcaList = seq(2, 15, 1),
  algorithm = c("exact", "randomized"),
  eigenCount = 32L,
  missingRate = NaN,
  verbose = FALSE
)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the opened 1KG GDS file.
gdsProfile	an object of class gds.class (a GDS file), the opened Profile GDS file.
syntheticKNN	a vector of character strings representing the name of files that contain the results of ancestry inference done on the synthetic profiles for multiple values of *D* and *K*. The files must exist.
pedSyn	a data.frame containing the columns extracted from the GDS Sample 'study.annot' node with a extra column named as the 'popName' parameter that has been extracted from the 1KG GDS 'sample.annot' node.
currentProfile	a character string representing the profile identifier of the current profile on which ancestry will be inferred.
spRef	a vector of character strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the vector.
studyIDSyn	a character string corresponding to the study identifier. The study identifier must be present in the GDS Sample file.
np	a single positive integer representing the number of threads. Default: 1L.
listCatPop	a vector of character string representing the list of possible ancestry assignations. Default: ("EAS", "EUR", "AFR", "AMR", "SAS").
fieldPopInRef	a character string representing the name of the column that contains the known ancestry for the reference profiles in the Reference GDS file.
fieldPopInfAnc	a character string representing the name of the column that will contain the inferred ancestry for the specified profiles. Default: "SuperPop".
kList	a vector of integer representing the list of values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).

pcaList — a vector of integer representing the list of values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).

algorithm — a character string representing the algorithm used to calculate the PCA. The 2 choices are "exact" (traditional exact calculation) and "randomized" (fast PCA with randomized algorithm introduced in Galinsky et al. 2016). Default: "exact".

eigenCount — a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned. Default: 32L.

missingRate — a numeric value representing the threshold missing rate at with the SNVs are discarded; the SNVs are retained in the snpgdsPCA with "<= missingRate" only; if NaN, no missing threshold. Default: NaN.

verbose — a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Value

a list containing 4 entries:

pcaSample a list containing the information related to the eigenvectors. The list contains those 3 entries:

- sample.id a character string representing the unique identifier of the current profile.
- eigenvector.ref a matrix of numeric containing the eigenvectors for the reference profiles.
- eigenvector a matrix of numeric containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

paraSample a list containing the results with different D and K values that lead to optimal parameter selection. The list contains those entries:

- dfPCA a data.frame containing statistical results on all combined synthetic results done with a fixed value of D (the number of dimensions). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - median a numeric representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - mad a numeric representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - upQuartile a numeric representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - k a numeric representing the optimal K value (the number of neighbors) for a fixed D value.
- dfPop a data.frame containing statistical results on all combined synthetic results done with different values of D (the number of dimensions) and K (the number of neighbors). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - K a numeric representing the value of K (the number of neighbors).

AUROC.min a numeric representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of D and K.

AUROC a numeric representing the accuracy obtained by grouping all the synthetic results for the specified values of D and K.

Accu.CM a numeric representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of D and K.

dfAUROC a data.frame the summary of the results by super-population. The data.frame contains those columns:

D a numeric representing the value of D (the number of dimensions).

K a numeric representing the value of K (the number of neighbors).

Call a character string representing the super-population.

L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

AUROC a numeric representing the AUROC obtained for the fixed values of super-population, D and K.

H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

D a numeric representing the optimal D value (the number of dimensions) for the specific profile.

K a numeric representing the optimal K value (the number of neighbors) for the specific profile.

listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a list containing the inferred ancestry using different D and K values. The list contains those entries:

sample.id a character string representing the unique identifier of the current profile.

matKNN a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

Ancestry a data.frame containing the inferred ancestry for the current profile. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library
library(gdsfmt)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## The Reference GDS file
path1KG <- system.file("extdata/tests", package="RAIDS")

## Open the Reference GDS file
gdsRef <- snpgdsOpen(file.path(path1KG, "ex1_good_small_1KG.gds"))

## Path to the demo synthetic results files
## List of the KNN result files from PCA run on synthetic data
dataDirRes <- system.file("extdata/demoAncestryCall/ex1", package="RAIDS")
listFilesName <- dir(file.path(dataDirRes), ".rds")
listFiles <- file.path(file.path(dataDirRes) , listFilesName)
syntheticKNN <- lapply(listFiles, FUN=function(x){return(readRDS(x))})
syntheticKNN <- do.call(rbind, syntheticKNN)

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))
## Not run:
    pedSyn <- RAIDS:::prepPedSynthetic1KG(gdsReference=gdsRef,
              gdsSample=gdsProfile, studyID=studyID, popName="superPop")

    ## Run the ancestry inference on one profile called 'ex1'
    ## The values of K and D used for the inference are selected using the
    ## synthetic results listFiles=listFiles,
    resCall <- RAIDS:::computeAncestryFromSynthetic(gdsReference=gdsRef,
                                gdsProfile=gdsProfile,
                                syntheticKNN = syntheticKNN,
                                pedSyn = pedSyn,
                                currentProfile=c("ex1"),
                                spRef=demoKnownSuperPop1KG,
                                studyIDSyn=studyID, np=1L)

    ## The ancestry called with the optimal D and K values
```

```
    resCall$Ancestry

## End(Not run)
## Close the GDS files (important)
closefn.gds(gdsProfile)
closefn.gds(gdsRef)
```

`computeAncestryFromSynthetic2` *Select the optimal K and D parameters using the synthetic data and infer the ancestry of a specific profile*

Description

The function select the optimal K and D parameters for a specific profile. The results on the synthetic data are used for the parameter selection. Once the optimal parameters are selected, the ancestry is inferred for the specific profile.

Usage

```
computeAncestryFromSynthetic2(
  syntheticKNN,
  pedSyn,
  spRef,
  listCatPop = c("EAS", "EUR", "AFR", "AMR", "SAS"),
  pRAIDS = pRAIDS
)
```

Arguments

`syntheticKNN`	a `vector` of `character` strings representing the name of files that contain the results of ancestry inference done on the synthetic profiles for multiple values of *D* and *K*. The files must exist.
`pedSyn`	a `data.frame` containing the columns extracted from the GDS Sample 'study.annot' node with a extra column named as the 'popName' parameter that has been extracted from the 1KG GDS 'sample.annot' node.
`spRef`	a `vector` of `character` strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the `vector`.
`listCatPop`	a `vector` of `character` string representing the list of possible ancestry assignations. Default: `("EAS", "EUR", "AFR", "AMR", "SAS")`.
`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters

Value

a list containing 4 entries:

pcaSample a list containing the information related to the eigenvectors. The list contains those 3 entries:

- sample.id a character string representing the unique identifier of the current profile.
- eigenvector.ref a matrix of numeric containing the eigenvectors for the reference profiles.
- eigenvector a matrix of numeric containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

paraSample a list containing the results with different D and K values that lead to optimal parameter selection. The list contains those entries:

- dfPCA a data.frame containing statistical results on all combined synthetic results done with a fixed value of D (the number of dimensions). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - median a numeric representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - mad a numeric representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - upQuartile a numeric representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - k a numeric representing the optimal K value (the number of neighbors) for a fixed D value.
- dfPop a data.frame containing statistical results on all combined synthetic results done with different values of D (the number of dimensions) and K (the number of neighbors). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - K a numeric representing the value of K (the number of neighbors).
  - AUROC.min a numeric representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of D and K.
  - AUROC a numeric representing the accuracy obtained by grouping all the synthetic results for the specified values of D and K.
  - Accu.CM a numeric representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of D and K.
- dfAUROC a data.frame the summary of the results by super-population. The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - K a numeric representing the value of K (the number of neighbors).
  - Call a character string representing the super-population.
  - L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.
  - AUROC a numeric representing the AUROC obtained for the fixed values of super-population, D and K.

H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

D a numeric representing the optimal D value (the number of dimensions) for the specific profile.

K a numeric representing the optimal K value (the number of neighbors) for the specific profile.

listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a list containing the inferred ancestry using different D and K values. The list contains those entries:

sample.id a character string representing the unique identifier of the current profile.

matKNN a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

Ancestry a data.frame containing the inferred ancestry for the current profile. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## The Reference GDS file
path1KG <- system.file("extdata/tests", package="RAIDS")

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")
```

```
# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                      pedStudy=pedStudy,
                      pathProfileGDS=pathProfileGDS,
                      fileReferenceGDS=fileReferenceGDS,
                      fieldPopInfAnc="SuperPop")

pedSyn <- RAIDS:::prepPedSyntheticRef(pRAIDS)
spRef <- getRefSuperPop2(pRAIDS)
syntheticKNN <- readRDS(file.path(dataDir, "ex1", "KNN.synt.ex1.2.rds"))
resCall <- computeAncestryFromSynthetic2(syntheticKNN=syntheticKNN,
                      pedSyn=pedSyn,
                      spRef=spRef,
                      listCatPop=c("EAS", "EUR", "AFR", "AMR", "SAS"),
                      pRAIDS=pRAIDS)

resCall$Ancestry
```

`computeAncestryFromSyntheticFile`

Select the optimal K and D parameters using the synthetic data and infer the ancestry of a specific profile

---

Description

The function select the optimal K and D parameters for a specific profile. The results on the synthetic data are used for the parameter selection. Once the optimal parameters are selected, the ancestry is inferred for the specific profile.

Usage

```
computeAncestryFromSyntheticFile(
  gdsReference,
  gdsProfile,
  listFiles,
  currentProfile,
  spRef,
  studyIDSyn,
  np = 1L,
  listCatPop = c("EAS", "EUR", "AFR", "AMR", "SAS"),
  fieldPopIn1KG = "superPop",
  fieldPopInfAnc = "SuperPop",
  kList = seq(2, 15, 1),
  pcaList = seq(2, 15, 1),
  algorithm = c("exact", "randomized"),
  eigenCount = 32L,
  missingRate = NaN,
  verbose = FALSE
)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the opened 1KG GDS file.
gdsProfile	an object of class gds.class (a GDS file), the opened Profile GDS file.
listFiles	a vector of character strings representing the name of files that contain the results of ancestry inference done on the synthetic profiles for multiple values of *D* and *K*. The files must exist.
currentProfile	a character string representing the profile identifier of the current profile on which ancestry will be inferred.
spRef	a vector of character strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the vector.
studyIDSyn	a character string corresponding to the study identifier. The study identifier must be present in the GDS Sample file.
np	a single positive integer representing the number of threads. Default: 1L.
listCatPop	a vector of character string representing the list of possible ancestry assignations. Default: ("EAS", "EUR", "AFR", "AMR", "SAS").
fieldPopIn1KG	a character string representing the name of the column that contains the known ancestry for the reference profiles in the Reference GDS file.
fieldPopInfAnc	a character string representing the name of the column that will contain the inferred ancestry for the specified profiles. Default: "SuperPop".
kList	a vector of integer representing the list of values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).

pcaList — a vector of integer representing the list of values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).

algorithm — a character string representing the algorithm used to calculate the PCA. The 2 choices are "exact" (traditional exact calculation) and "randomized" (fast PCA with randomized algorithm introduced in Galinsky et al. 2016). Default: "exact".

eigenCount — a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned. Default: 32L.

missingRate — a numeric value representing the threshold missing rate at with the SNVs are discarded; the SNVs are retained in the snpgdsPCA with "<= missingRate" only; if NaN, no missing threshold. Default: NaN.

verbose — a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Value

a list containing 4 entries:

pcaSample a list containing the information related to the eigenvectors. The list contains those 3 entries:

- sample.id a character string representing the unique identifier of the current profile.
- eigenvector.ref a matrix of numeric containing the eigenvectors for the reference profiles.
- eigenvector a matrix of numeric containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

paraSample a list containing the results with different D and K values that lead to optimal parameter selection. The list contains those entries:

- dfPCA a data.frame containing statistical results on all combined synthetic results done with a fixed value of D (the number of dimensions). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - median a numeric representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - mad a numeric representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - upQuartile a numeric representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.
  - k a numeric representing the optimal K value (the number of neighbors) for a fixed D value.
- dfPop a data.frame containing statistical results on all combined synthetic results done with different values of D (the number of dimensions) and K (the number of neighbors). The data.frame contains those columns:
  - D a numeric representing the value of D (the number of dimensions).
  - K a numeric representing the value of K (the number of neighbors).

AUROC.min a numeric representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of D and K.

AUROC a numeric representing the accuracy obtained by grouping all the synthetic results for the specified values of D and K.

Accu.CM a numeric representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of D and K.

dfAUROC a data.frame the summary of the results by super-population. The data.frame contains those columns:

pcaD a numeric representing the value of D (the number of dimensions).

K a numeric representing the value of K (the number of neighbors).

Call a character string representing the super-population.

L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

AUR a numeric representing the AUROC obtained for the fixed values of super-population, D and K.

H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

D a numeric representing the optimal D value (the number of dimensions) for the specific profile.

K a numeric representing the optimal K value (the number of neighbors) for the specific profile.

listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a list containing the inferred ancestry using different D and K values. The list contains those entries:

sample.id a character string representing the unique identifier of the current profile.

matKNN a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

Ancestry a data.frame containing the inferred ancestry for the current profile. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library
library(gdsfmt)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## The Reference GDS file
path1KG <- system.file("extdata/tests", package="RAIDS")

## Open the Reference GDS file
gdsRef <- snpgdsOpen(file.path(path1KG, "ex1_good_small_1KG.gds"))

## Path to the demo synthetic results files
## List of the KNN result files from PCA run on synthetic data
dataDirRes <- system.file("extdata/demoAncestryCall/ex1", package="RAIDS")
listFilesName <- dir(file.path(dataDirRes), ".rds")
listFiles <- file.path(file.path(dataDirRes) , listFilesName)

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

## Run the ancestry inference on one profile called 'ex1'
## The values of K and D used for the inference are selected using the
## synthetic results
resCall <- computeAncestryFromSyntheticFile(gdsReference=gdsRef,
                            gdsProfile=gdsProfile,
                            listFiles=listFiles,
                            currentProfile=c("ex1"),
                            spRef=demoKnownSuperPop1KG,
                            studyIDSyn=studyID, np=1L)

## The ancestry called with the optimal D and K values
resCall$Ancestry

## Close the GDS files (important)
closefn.gds(gdsProfile)
closefn.gds(gdsRef)
```

computeKNNRefSample *Run a k-nearest neighbors analysis on one specific profile*

Description

The function runs k-nearest neighbors analysis on a one specific profile. The function uses the 'knn' package.

Usage

```
computeKNNRefSample(
  listEigenvector,
  listCatPop = c("EAS", "EUR", "AFR", "AMR", "SAS"),
  spRef,
  fieldPopInfAnc = "SuperPop",
  kList = seq(2, 15, 1),
  pcaList = seq(2, 15, 1)
)
```

Arguments

listEigenvector	a list with 3 entries: 'sample.id', 'eigenvector.ref' and 'eigenvector'. The list represents the PCA done on the 1KG reference profiles and one specific profile projected onto it. The 'sample.id' entry must contain only one identifier (one profile).
listCatPop	a vector of character string representing the list of possible ancestry assignations. Default: c("EAS", "EUR", "AFR", "AMR", "SAS").
spRef	vector of character strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the vector.
fieldPopInfAnc	a character string representing the name of the column that will contain the inferred ancestry for the specified profile. Default: "SuperPop".
kList	a vector of integer representing the list of values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).
pcaList	a vector of integer representing the list of values tested for the *D* parameter. The D parameter represents the number of dimensions used in the PCA analysis. If NULL, the value seq(2, 15, 1) is assigned. Default: seq(2, 15, 1).

Value

a `list` containing 4 entries:

`sample.id` a `vector` of `character` strings representing the identifier of the profile analysed.

`matKNN` a `data.frame` containing the super population inference for the profile for different values of PCA dimensions `D` and k-neighbors values `K`. The fourth column title corresponds to the `fieldPopInfAnc` parameter. The `data.frame` contains 4 columns:

`sample.id` a `character` string representing the identifier of the profile analysed.

`D` a `numeric` strings representing the value of the PCA dimension used to infer the ancestry.

`K` a `numeric` strings representing the value of the k-neighbors used to infer the ancestry..

`fieldPopInfAnc` a `character` string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Load the demo PCA on the synthetic profiles projected on the
## demo 1KG reference PCA
data(demoPCASyntheticProfiles)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## The PCA with 1 profile projected on the 1KG reference PCA
## Only one profile is retained
pca <- demoPCASyntheticProfiles
pca$sample.id <- pca$sample.id[1]
pca$eigenvector <- pca$eigenvector[1, , drop=FALSE]

## Projects profile on 1KG PCA
results <- computeKNNRefSample(listEigenvector=pca,
    listCatPop=c("EAS", "EUR", "AFR", "AMR", "SAS"),
    spRef=demoKnownSuperPop1KG, fieldPopInfAnc="SuperPop",
    kList=seq(10, 15, 1), pcaList=seq(10, 15, 1))

## The assigned ancestry to the profile for different values of K and D
head(results$matKNN)
```

`computeKNNRefSyn` *Run a k-nearest neighbors analysis on a subset of the synthetic dataset*

Description

The function runs k-nearest neighbors analysis on a subset of the synthetic data set. The function uses the 'knn' package.

Usage

```
computeKNNRefSyn(listEigenvector, pRAIDS)
```

Arguments

listEigenvector	a list with 3 entries: 'sample.id', 'eigenvector.ref' and 'eigenvector'. The list represents the PCA done on the 1KG reference profiles and the synthetic profiles projected onto it.
pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Value

a list containing 4 entries:

sample.id a vector of character strings representing the identifiers of the synthetic profiles analysed.

sample1Kg a vector of character strings representing the identifiers of the 1KG reference profiles used to generate the synthetic profiles.

sp a vector of character strings representing the known super population ancestry of the 1KG reference profiles used to generate the synthetic profiles.

matKNN a data.frame containing the super population inference for each synthetic profiles for different values of PCA dimensions D and k-neighbors values K. The fourth column title corresponds to the fieldPopInfAnc parameter. The data.frame contains 4 columns:

- sample.id a character string representing the identifier of the synthetic profile analysed.
- D a numeric strings representing the value of the PCA dimension used to infer the super population.
- K a numeric strings representing the value of the k-neighbors used to infer the super population.
- fieldPopInfAnc **value** a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Load the demo PCA on the synthetic profiles projected on the
## demo 1KG reference PCA
data(demoPCASyntheticProfiles)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")
```

```
# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=fileReferenceGDS)
## Projects synthetic profiles on 1KG PCA
results <- computeKNNRefSyn(listEigenvector=demoPCASyntheticProfiles,
    pRAIDS=pRAIDS)

## The inferred ancestry for the synthetic profiles for different values
## of D and K
head(results$matKNN)
```

`computeKNNRefSynthetic` *Run a k-nearest neighbors analysis on a subset of the synthetic dataset*

Description

The function runs k-nearest neighbors analysis on a subset of the synthetic data set. The function uses the 'knn' package.

Usage

```
computeKNNRefSynthetic(
  gdsProfile,
  listEigenvector,
  listCatPop = c("EAS", "EUR", "AFR", "AMR", "SAS"),
  studyIDSyn,
  spRef,
```

```
  fieldPopInfAnc = "SuperPop",
  kList = seq(2, 15, 1),
  pcaList = seq(2, 15, 1)
)
```

Arguments

gdsProfile	an object of class SNPRelate::SNPGDSFileClass, the opened Profile GDS file.
listEigenvector	a list with 3 entries: 'sample.id', 'eigenvector.ref' and 'eigenvector'. The list represents the PCA done on the 1KG reference profiles and the synthetic profiles projected onto it.
listCatPop	a vector of character string representing the list of possible ancestry assignations. Default: c("EAS", "EUR", "AFR", "AMR", "SAS").
studyIDSyn	a character string corresponding to the study identifier. The study identifier must be present in the Profile GDS file.
spRef	vector of character strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the vector.
fieldPopInfAnc	a character string representing the name of the column that will contain the inferred ancestry for the specified data set. Default: "SuperPop".
kList	a vector of integer representing the list of values tested for the K parameter. The K parameter represents the number of neighbors used in the K-nearest neighbors analysis. If NULL, the value seq(2, 15, 1) is assigned. Default: seq(2, 15, 1).
pcaList	a vector of integer representing the list of values tested for the D parameter. The D parameter represents the number of dimensions used in the PCA analysis. If NULL, the value seq(2, 15, 1) is assigned. Default: seq(2, 15, 1).

Value

a list containing 4 entries:

sample.id a vector of character strings representing the identifiers of the synthetic profiles analysed.

sample1Kg a vector of character strings representing the identifiers of the 1KG reference profiles used to generate the synthetic profiles.

sp a vector of character strings representing the known super population ancestry of the 1KG reference profiles used to generate the synthetic profiles.

matKNN a data.frame containing the super population inference for each synthetic profiles for different values of PCA dimensions D and k-neighbors values K. The fourth column title corresponds to the fieldPopInfAnc parameter. The data.frame contains 4 columns:

- sample.id a character string representing the identifier of the synthetic profile analysed.
- D a numeric strings representing the value of the PCA dimension used to infer the super population.

K a numeric strings representing the value of the k-neighbors used to infer the super population.

fieldPopInfAnc **value** a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Load the demo PCA on the synthetic profiles projected on the
## demo 1KG reference PCA
data(demoPCASyntheticProfiles)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

## Projects synthetic profiles on 1KG PCA
results <- computeKNNRefSynthetic(gdsProfile=gdsProfile,
    listEigenvector=demoPCASyntheticProfiles,
    listCatPop=c("EAS", "EUR", "AFR", "AMR", "SAS"), studyIDSyn=studyID,
    spRef=demoKnownSuperPop1KG)

## The inferred ancestry for the synthetic profiles for different values
## of D and K
head(results$matKNN)

## Close Profile GDS file (important)
closefn.gds(gdsProfile)
```

---

computePCAMultiSynthetic

Project synthetic profiles onto existing principal component axes generated using the reference 1KG profiles

---

Description

The function projects the synthetic profiles onto existing principal component axes generated using the reference 1KG profiles. The reference profiles used to generate the synthetic profiles have previously been removed from the set of reference profiles.

Usage

```
computePCAMultiSynthetic(
  gdsProfile,
  listPCA,
  sampleRef,
  studyIDSyn,
  verbose = FALSE
)
```

Arguments

gdsProfile	an object of class gds.class (a GDS file), an opened Profile GDS file.
listPCA	a list containing the PCA object generated with the 1KG reference profiles (excluding the ones used to generate the synthetic data set) in an entry called "pca.unrel".
sampleRef	a vector of character strings representing the identifiers of the 1KG reference profiles that have been used to generate the synthetic profiles that are going to be analysed here. The sub-continental identifiers are used as names for the vector.
studyIDSyn	a character string corresponding to the study identifier. The study identifier must be present in the Profile GDS file.
verbose	a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Value

a list containing 3 entries:

sample.id a vector of character strings representing the identifiers of the synthetic profiles that have been projected onto the 1KG PCA.

eigenvector.ref a matrix of numeric with the eigenvectors of the 1KG reference profiles used to generate the PCA.

eigenvector a matrix of numeric with the eigenvectors of the synthetic profiles projected onto the 1KG PCA.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Loading demo PCA on subset of 1KG reference dataset
data(demoPCA1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

samplesRM <- c("HG00246", "HG00325", "HG00611", "HG01173", "HG02165",
    "HG01112", "HG01615", "HG01968", "HG02658", "HG01850", "HG02013",
    "HG02465", "HG02974", "HG03814", "HG03445", "HG03689", "HG03789",
    "NA12751", "NA19107", "NA18548", "NA19075", "NA19475", "NA19712",
    "NA19731", "NA20528", "NA20908")
names(samplesRM) <- c("GBR", "FIN", "CHS","PUR", "CDX", "CLM", "IBS",
    "PEL", "PJL", "KHV", "ACB", "GWD", "ESN", "BEB", "MSL", "STU", "ITU",
    "CEU", "YRI", "CHB", "JPT", "LWK", "ASW", "MXL", "TSI", "GIH")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

## Projects synthetic profiles on 1KG PCA
results <- computePCAMultiSynthetic(gdsProfile=gdsProfile,
    listPCA=demoPCA1KG,
    sampleRef=samplesRM, studyIDSyn=studyID, verbose=FALSE)

## The eigenvectors for the synthetic profiles
head(results$eigenvector)

## Close Profile GDS file (important)
closefn.gds(gdsProfile)
```

`computePCARefSample` *Project specified profile onto PCA axes generated using known reference profiles*

Description

This function generates a PCA using the know reference profiles. Them, it projects the specified profile onto the PCA axes.

Usage

```
computePCARefSample(
```

```
  gdsProfile,
  currentProfile,
  studyIDRef = "Ref.1KG",
  np = 1L,
  algorithm = c("exact", "randomized"),
  eigenCount = 32L,
  missingRate = NaN,
  verbose = FALSE
)
```

Arguments

gdsProfile	an object of class gds.class, an opened Profile GDS file.
currentProfile	a single character string representing the profile identifier.
studyIDRef	a single character string representing the study identifier.
np	a single positive integer representing the number of CPU that will be used. Default: 1L.
algorithm	a character string representing the algorithm used to calculate the PCA. The 2 choices are "exact" (traditional exact calculation) and "randomized" (fast PCA with randomized algorithm introduced in Galinsky et al. 2016). Default: "exact".
eigenCount	a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigen.cnt' <= 0, then all eigenvectors are returned. Default: 32L.
missingRate	a numeric value representing the threshold missing rate at with the SNVs are discarded; the SNVs are retained in the snpgdsPCA with "<= missingRate" only; if NaN, no missing threshold. Default: NaN.
verbose	a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Value

a list containing 3 entries:

sample.id a character string representing the unique identifier of the analyzed profile.

eigenvector.ref a matrix of numeric representing the eigenvectors of the reference profiles.

eigenvector a matrix of numeric representing the eigenvectors of the analyzed profile.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library
library(gdsfmt)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

## Project a profile onto a PCA generated using reference profiles
## The reference profiles come from 1KG
resPCA <- computePCARefSample(gdsProfile=gdsProfile,
    currentProfile=c("ex1"), studyIDRef="Ref.1KG", np=1L, verbose=FALSE)
resPCA$sample.id
resPCA$eigenvector

## Close the GDS files (important)
closefn.gds(gdsProfile)
```

---

computePoolSyntheticAncestryGr

Run a PCA analysis and a K-nearest neighbors analysis on a small set of synthetic data using all 1KG profiles except the ones used to generate the synthetic profiles

---

Description

The function runs a PCA analysis using 1 synthetic profile from each sub-continental population. The reference profiles used to create those synthetic profiles are first removed from the list of 1KG reference profiles that generates the reference PCA. Then, the retained synthetic profiles are projected on the 1KG PCA space. Finally, a K-nearest neighbors analysis using a range of K and D values is done.

Usage

```
computePoolSyntheticAncestryGr(
  gdsProfile,
  sampleRM,
  spRef,
  studyIDSyn,
  np = 1L,
  listCatPop = c("EAS", "EUR", "AFR", "AMR", "SAS"),
  fieldPopInfAnc = "SuperPop",
  kList = seq(2, 15, 1),
  pcaList = seq(2, 15, 1),
```

```
  algorithm = c("exact", "randomized"),
  eigenCount = 32L,
  missingRate = 0.025,
  verbose = FALSE
)
```

Arguments

gdsProfile	an object of class SNPRelate::SNPGDSFileClass, the opened Profile GDS file.
sampleRM	a vector of character strings representing the identifiers of the 1KG reference profiles that should not be used to create the reference PCA. There should be one per sub-continental population. Those profiles are removed because those have been used to generate the synthetic profiles that are going to be analysed here. The sub-continental identifiers are used as names for the vector.
spRef	vector of character strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the vector.
studyIDSyn	a character string corresponding to the study identifier. The study identifier must be present in the Profile GDS file.
np	a single positive integer representing the number of threads. Default: 1L.
listCatPop	a vector of character string representing the list of possible ancestry assignations. Default: ("EAS", "EUR", "AFR", "AMR", "SAS").
fieldPopInfAnc	a character string representing the name of the column that will contain the inferred ancestry for the specified dataset. Default: "SuperPop".
kList	a vector of integer representing the list of values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).
pcaList	a vector of integer representing the list of values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis. If NULL, the value seq(2,15,1) is assigned. Default: seq(2,15,1).
algorithm	a character string representing the algorithm used to calculate the PCA. The 2 choices are "exact" (traditional exact calculation) and "randomized" (fast PCA with randomized algorithm introduced in Galinsky et al. 2016). Default: "exact".
eigenCount	a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned. Default: 32L.
missingRate	a numeric value representing the threshold missing rate at with the SNVs are discarded; the SNVs are retained in the snpgdsPCA function with "<= missingRate" only; if NaN, no missing threshold. Default: 0.025.
verbose	a logical indicating if message information should be printed. Default: FALSE.

Value

a list containing the following entries:

sample.id a vector of character strings representing the identifiers of the synthetic profiles.

sample1Kg a `vector` of `character` strings representing the identifiers of the reference 1KG profiles used to generate the synthetic profiles.

sp a `vector` of `character` strings representing the known ancestry for the reference 1KG profiles used to generate the synthetic profiles.

matKNN a `data.frame` containing 4 columns. The first column 'sample.id' contains the name of the synthetic profile. The second column 'D' represents the dimension D used to infer the ancestry. The third column 'K' represents the number of neighbors K used to infer the ancestry. The fourth column 'SuperPop' contains the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library
library(gdsfmt)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)


# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

samplesRM <- c("HG00246", "HG00325", "HG00611", "HG01173", "HG02165",
    "HG01112", "HG01615", "HG01968", "HG02658", "HG01850", "HG02013",
    "HG02465", "HG02974", "HG03814", "HG03445", "HG03689", "HG03789",
    "NA12751", "NA19107", "NA18548", "NA19075", "NA19475", "NA19712",
    "NA19731", "NA20528", "NA20908")
names(samplesRM) <- c("GBR", "FIN", "CHS","PUR", "CDX", "CLM", "IBS",
    "PEL", "PJL", "KHV", "ACB", "GWD", "ESN", "BEB", "MSL", "STU", "ITU",
    "CEU", "YRI", "CHB", "JPT", "LWK", "ASW", "MXL", "TSI", "GIH")

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

## Run a PCA analysis and a K-nearest neighbors analysis on a small set
## of synthetic data
results <- computePoolSyntheticAncestryGr(gdsProfile=gdsProfile,
    sampleRM=samplesRM, studyIDSyn=studyID, np=1L,
    spRef=demoKnownSuperPop1KG,
```

```
    kList=seq(10,15,1), pcaList=seq(10,15,1), eigenCount=15L)

## The ancestry inference for the synthetic data using
## different K and D values
head(results$matKNN)

## Close Profile GDS file (important)
closefn.gds(gdsProfile)
```

`computePoolSyntheticAncestryGr2`

Run a PCA analysis and a K-nearest neighbors analysis on a small set of synthetic data using all 1KG profiles except the ones used to generate the synthetic profiles

---

Description

The function runs a PCA analysis using 1 synthetic profile from each sub-continental population. The reference profiles used to create those synthetic profiles are first removed from the list of 1KG reference profiles that generates the reference PCA. Then, the retained synthetic profiles are projected on the 1KG PCA space. Finally, a K-nearest neighbors analysis using a range of K and D values is done.

Usage

```
computePoolSyntheticAncestryGr2(sampleRM, spRef, pRAIDS)
```

Arguments

`sampleRM`	a `vector` of `character` strings representing the identifiers of the 1KG reference profiles that should not be used to create the reference PCA. There should be one per sub-continental population. Those profiles are removed because those have been used to generate the synthetic profiles that are going to be analysed here. The sub-continental identifiers are used as names for the `vector`.
`spRef`	`vector` of `character` strings representing the known super population ancestry for the 1KG profiles. The 1KG profile identifiers are used as names for the `vector`.
`pRAIDS`	a `RAIDSparam` an object containing all RAIDS parameters.

Value

a `list` containing the following entries:

sample.id a `vector` of `character` strings representing the identifiers of the synthetic profiles.

sample1Kg a `vector` of `character` strings representing the identifiers of the reference 1KG profiles used to generate the synthetic profiles.

sp a vector of character strings representing the known ancestry for the reference 1KG profiles used to generate the synthetic profiles.

matKNN a data.frame containing 4 columns. The first column 'sample.id' contains the name of the synthetic profile. The second column 'D' represents the dimension D used to infer the ancestry. The third column 'K' represents the number of neighbors K used to infer the ancestry. The fourth column 'SuperPop' contains the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

samplesRM <- c("HG00246", "HG00325", "HG00611", "HG01173", "HG02165",
    "HG01112", "HG01615", "HG01968", "HG02658", "HG01850", "HG02013",
    "HG02465", "HG02974", "HG03814", "HG03445", "HG03689", "HG03789",
    "NA12751", "NA19107", "NA18548", "NA19075", "NA19475", "NA19712",
    "NA19731", "NA20528", "NA20908")

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")


# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID
```

```
pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=fileReferenceGDS)

result <- computePoolSyntheticAncestryGr2(sampleRM=samplesRM,
                             spRef=demoKnownSuperPop1KG,
                             pRAIDS=pRAIDS)
## The ancestry inference for the synthetic data using
## different K and D values
head(result$matKNN)
```

`computeSyntheticROC` *Calculate the AUROC of the inferences for specific values of D and K using the inferred ancestry results from the synthetic profiles.*

Description

The function calculates the AUROC of the inferences for specific values of D and K using the inferred ancestry results from the synthetic profiles. The calculations are done on each super-population separately as well as on all the results together.

Usage

```
computeSyntheticROC(
  matKNN,
  matKNNAncestryColumn,
  pedCall,
  pedCallAncestryColumn,
  listCall = c("EAS", "EUR", "AFR", "AMR", "SAS")
)
```

Arguments

`matKNN`	a `data.frame` containing the inferred ancestry results for fixed values of *D* and *K*. On of the column names of the `data.frame` must correspond to the `matKNNAncestryColumn` argument.
`matKNNAncestryColumn`	a `character` string representing the name of the column that contains the inferred ancestry for the specified synthetic profiles. The column must be present in the `matKNN` argument.
`pedCall`	a `data.frame` containing the information about the super-population information from the 1KG GDS file for profiles used to generate the synthetic profiles. The `data.frame` must contained a column named as the `pedCallAncestryColumn` argument. The row names must correspond to the sample identifiers (mandatory).

pedCallAncestryColumn
: a character string representing the name of the column that contains the known ancestry for the reference profiles in the Reference GDS file. The column must be present in the pedCall argument.

listCall
: a vector of character strings representing the list of all possible ancestry assignations. Default: c("EAS", "EUR", "AFR", "AMR", "SAS").

Value

list containing 3 entries:

matAUROC.All a data.frame containing the AUROC for all the ancestry results.

matAUROC.Call a data.frame containing the AUROC information for each super-population.

listROC.Call a list containing the output from the roc function for each super-population.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Loading demo dataset containing pedigree information for synthetic
## profiles and known ancestry of the profiles used to generate the
## synthetic profiles
data(pedSynthetic)

## Loading demo dataset containing the inferred ancestry results
## for the synthetic data
data(matKNNSynthetic)

## The inferred ancestry results for the synthetic data using
## values of D=6 and K=5
matKNN <- matKNNSynthetic[matKNNSynthetic$K == 6 & matKNNSynthetic$D == 5, ]

## Compile statistics from the
## synthetic profiles for fixed values of D and K
results <- RAIDS:::computeSyntheticROC(matKNN=matKNN,
    matKNNAncestryColumn="SuperPop",
    pedCall=pedSynthetic, pedCallAncestryColumn="superPop",
    listCall=c("EAS", "EUR", "AFR", "AMR", "SAS"))

results$matAUROC.All
results$matAUROC.Call
results$listROC.Call
```

createAccuracyGraph *Generate accuracy graph*

Description

This function extracts the required information from an output generated by RAIDS to create a graphic representation of the accuracy for different values of PCA dimensions and K-neighbors through all tested ancestries.

Usage

```
createAccuracyGraph(
  fileRDS,
  title = "",
  selectD = c(3, 7, 11),
  selectColor = c("#5e688a", "#cd5700", "#CC79A7")
)
```

Arguments

`fileRDS`	a `character` string representing the path and file name of the RDS file containing the ancestry information as generated by RAIDS.
`title`	a `character` string representing the title of the graph. Default: `""`.
`selectD`	a `array` of `integer` representing the selected PCA dimensions to plot. The length of the `array` cannot be more than 5 entries. The dimensions must tested by RAIDS (i.e. be present in the RDS file). Default: `c(3,7,11)`.
`selectColor`	a `array` of `character` strings representing the selected colors for the associated PCA dimensions to plot. The length of the `array` must correspond to the length of the `selectD` parameter. In addition, the length of the `array` cannot be more than 5 entries. Default: `c("#5e688a", "#cd5700", "#CC79A7")`.

Value

a `ggplot` object containing the graphic representation of the accuracy for different values of PCA dimensions and K-neighbors through all tested ancestries.

Author(s)

Astrid Deschênes and Pascal Belleau

Examples

```
## Required library
library(ggplot2)

## Path to RDS file with ancestry information generated by RAIDS (demo file)
dataDir <- system.file("extdata", package="RAIDS")
```

```
fileRDS <- file.path(dataDir, "TEST_01.infoCall.RDS")

## Create accuracy graph
accuracyGraph <- createAccuracyGraph(fileRDS=fileRDS, title="Test 01",
    selectD=c(3,6,9,12,15),
    selectColor=c("steelblue", "darkorange", "violet", "pink", "gray80"))

accuracyGraph
```

`createAUROCGraph` *Generate accuracy graph*

Description

This function extracts the required information from an output generated by RAIDS to create a graphic representation of the accuracy for different values of PCA dimensions and K-neighbors through all tested ancestries.

Usage

```
createAUROCGraph(
  dfAUROC,
  title = "",
  selectD = c(3, 7, 11),
  selectColor = c("#5e688a", "#cd5700", "#CC79A7")
)
```

Arguments

`dfAUROC`	a `data.frame` corresponding to res$paraSample$dfAUROC where res is the result of inferAncestry() or inferAncestryGeneAware() functions.
`title`	a `character` string representing the title of the graph. Default: `""`.
`selectD`	a `array` of `integer` representing the selected PCA dimensions to plot. The length of the `array` cannot be more than 5 entries. The dimensions must tested by RAIDS (i.e. be present in the RDS file). Default: `c(3,7,11)`.
`selectColor`	a `array` of `character` strings representing the selected colors for the associated PCA dimensions to plot. The length of the `array` must correspond to the length of the `selectD` parameter. In addition, the length of the `array` cannot be more than 5 entries. Default: `c("#5e688a", "#cd5700", "#CC79A7")`.

Value

a `ggplot` object containing the graphic representation of the accuracy for different values of PCA dimensions and K-neighbors through all tested ancestries.

Author(s)

Astrid Deschênes and Pascal Belleau

Examples

```
## Required library
library(ggplot2)

## Path to RDS file with ancestry information generated by RAIDS (demo file)
dataDir <- system.file("extdata", package="RAIDS")
fileRDS <- file.path(dataDir, "TEST_01.infoCall.RDS")
info <- readRDS(fileRDS)
dfAUROC <- info$paraSample$dfAUROC

## Some of the column names must be updated to fit new standards
colnames(dfAUROC) <- c("D", "K", "Call", "L", "AUROC", "H")

## Create accuracy graph
accuracyGraph <- createAUROCGraph(dfAUROC=dfAUROC, title="Test 01",
    selectD=c(3, 6, 9, 12, 15),
    selectColor=c("steelblue", "darkorange", "violet", "pink", "gray40"))

accuracyGraph
```

---

createStudy2GDS1KG *Create the Profile GDS file(s) for one or multiple specific profiles using the information from a RDS Sample description file and the 1KG GDS file*

Description

The function uses the information for the Reference GDS file and the RDS Sample Description file to create the Profile GDS file. One Profile GDS file is created per profile. One Profile GDS file will be created for each entry present in the `listProfiles` parameter.

Usage

```
createStudy2GDS1KG(
  pathGeno = file.path("data", "sampleGeno"),
  filePedRDS = NULL,
  pedStudy = NULL,
  fileNameGDS,
  batch = 1,
  studyDF,
  listProfiles = NULL,
  pathProfileGDS = NULL,
  genoSource = c("snp-pileup", "generic", "VCF"),
```

```
  verbose = FALSE
)
```

Arguments

pathGeno	a character string representing the path to the directory containing the VCF output of SNP-pileup for each sample. The SNP-pileup files must be compressed (gz files) and have the name identifiers of the samples. A sample with "Name.ID" identifier would have an associated file called if genoSource is "VCF", then "Name.ID.vcf.gz", if genoSource is "generic", then "Name.ID.generic.txt.gz" if genoSource is "snp-pileup", then "Name.ID.txt.gz".
filePedRDS	a character string representing the path to the RDS file that contains the information about the sample to analyse. The RDS file must include a data.frame with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", "Source". All columns must be in character strings. The data.frame must contain the information for all the samples passed in the listSamples parameter. Only filePedRDS or pedStudy can be defined.
pedStudy	a data.frame with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", "Source". All columns must be in character strings (no factor). The data.frame must contain the information for all the samples passed in the listSamples parameter. Only filePedRDS or pedStudy can be defined.
fileNameGDS	a character string representing the file name of the Reference GDS file. The file must exist.
batch	a single positive integer representing the current identifier for the batch. Beware, this field is not stored anymore. Default: 1.
studyDF	a data.frame containing the information about the study associated to the analysed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be in character strings (no factor).
listProfiles	a vector of character string corresponding to the profile identifiers that will have a Profile GDS file created. The profile identifiers must be present in the "Name.ID" column of the Profile RDS file passed to the filePedRDS parameter. If NULL, all profiles present in the filePedRDS are selected. Default: NULL.
pathProfileGDS	a character string representing the path to the directory where the Profile GDS files will be created. Default: NULL.
genoSource	a character string with two possible values: 'snp-pileup', 'generic' or 'VCF'. It specifies if the genotype files are generated by snp-pileup (Facets) or are a generic format CSV file with at least those columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R' and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele and 'File1A' is the depth of the specific alternative allele. Finally the file can be a VCF file with at least those genotype fields: GT, AD, DP.
verbose	a logical indicating if message information should be printed. Default: FALSE.

Value

The function returns 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

## The data.frame containing the information about the study
## The 3 mandatory columns: "study.id", "study.desc", "study.platform"
## The entries should be strings, not factors (stringsAsFactors=FALSE)
studyDF <- data.frame(study.id = "MYDATA",
                        study.desc = "Description",
                        study.platform = "PLATFORM",
                        stringsAsFactors = FALSE)

## The data.frame containing the information about the samples
## The entries should be strings, not factors (stringsAsFactors=FALSE)
samplePED <- data.frame(Name.ID=c("ex1", "ex2"),
                    Case.ID=c("Patient_h11", "Patient_h12"),
                    Diagnosis=rep("Cancer", 2),
                    Sample.Type=rep("Primary Tumor", 2),
                    Source=rep("Databank B", 2), stringsAsFactors=FALSE)
rownames(samplePED) <- samplePED$Name.ID

## Create the Profile GDS File for samples in 'listSamples' vector
## (in this case, samples "ex1")
## The Profile GDS file is created in the pathProfileGDS directory
result <- createStudy2GDS1KG(pathGeno=dataDir,
            pedStudy=samplePED, fileNameGDS=fileGDS,
            studyDF=studyDF, listProfiles=c("ex1"),
            pathProfileGDS=tempdir(),
            genoSource="snp-pileup",
            verbose=FALSE)

## The function returns 0L when successful
result

## The Profile GDS file 'ex1.gds' has been created in the
## specified directory
list.files(tempdir())

## Remove Profile GDS file (created for demo purpose)
unlink(file.path(tempdir(), "ex1.gds"), force=TRUE)
```

`DataFrameOrNULL-class` *Class Union for data.frame or NULL*

Description

A virtual class that groups data.frame and NULL together.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

---

demoKnownSuperPop1KG *The known super population ancestry of the demo 1KG reference profiles.*

Description

The object is a vector.

Usage

```
data(demoKnownSuperPop1KG)
```

Format

The vector containing the know super population ancestry for the demo 1KG reference profiles.

Details

This object can be used to test the computeKNNRefSynthetic and computePoolSyntheticAncestryGr functions.

Value

The vector containing the know super population ancestry for the demo 1KG reference profiles.

See Also

computeKNNRefSynthetic for running a k-nearest neighbors analysis on a subset of the synthetic data set.

computePoolSyntheticAncestryGr for running a PCA analysis using 1 synthetic profile from each sub-continental population.

Examples

```
## Required library
library(gdsfmt)

## Load the demo PCA on the synthetic profiles projected on the
## demo 1KG reference PCA
data(demoPCASyntheticProfiles)
```

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

## Projects synthetic profiles on 1KG PCA
results <- computeKNNRefSynthetic(gdsProfile=gdsProfile,
    listEigenvector=demoPCASyntheticProfiles,
    listCatPop=c("EAS", "EUR", "AFR", "AMR", "SAS"), studyIDSyn=studyID,
    spRef=demoKnownSuperPop1KG)

## The inferred ancestry for the synthetic profiles for different values
## of D and K
head(results$matKNN)

## Close Profile GDS file (important)
closefn.gds(gdsProfile)
```

---

demoPCA1KG — *The PCA results of the demo 1KG reference dataset for demonstration purpose. Beware that the PCA has been run on a very small subset of the 1KG reference dataset and should not be used to call ancestry inference on a real profile.*

Description

The object is a `list`.

Usage

```
data(demoPCA1KG)
```

Format

The `list` containing the PCA results for a small subset of the reference 1KG dataset. The `list` contains 2 entries:

pruned a `vector` of SNV identifiers specifying selected SNVs for the PCA analysis.

pca.unrel a `snpgdsPCAClass` object containing the eigenvalues as generated by snpgdsPCA function.

Details

This object can be used to test the computePCAMultiSynthetic function.

Value

The list containing the PCA results for a small subset of the reference 1KG dataset. The list contains 2 entries:

pruned a vector of SNV identifiers specifying selected SNVs for the PCA analysis.

pca.unrel a snpgdsPCAClass object containing the eigenvalues as generated by snpgdsPCA function.

Examples

```
## Required library
library(gdsfmt)

## Loading demo PCA on subset of 1KG reference dataset
data(demoPCA1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

samplesRM <- c("HG00246", "HG00325", "HG00611", "HG01173", "HG02165",
    "HG01112", "HG01615", "HG01968", "HG02658", "HG01850", "HG02013",
    "HG02465", "HG02974", "HG03814", "HG03445", "HG03689", "HG03789",
    "NA12751", "NA19107", "NA18548", "NA19075", "NA19475", "NA19712",
    "NA19731", "NA20528", "NA20908")
names(samplesRM) <- c("GBR", "FIN", "CHS","PUR", "CDX", "CLM", "IBS",
    "PEL", "PJL", "KHV", "ACB", "GWD", "ESN", "BEB", "MSL", "STU", "ITU",
    "CEU", "YRI", "CHB", "JPT", "LWK", "ASW", "MXL", "TSI", "GIH")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

## Projects synthetic profiles on demo 1KG PCA
results <- computePCAMultiSynthetic(gdsProfile=gdsProfile,
    listPCA=demoPCA1KG, sampleRef=samplesRM, studyIDSyn=studyID,
    verbose=FALSE)

## The eigenvectors for the synthetic profiles
head(results$eigenvector)

## Close Profile GDS file (important)
closefn.gds(gdsProfile)
```

demoPCASyntheticProfiles *The PCA result of demo synthetic profiles projected on the demo subset 1KG reference PCA.*

Description

The object is a `list`.

Usage

```
data(demoPCASyntheticProfiles)
```

Format

The `list` containing the PCA result of demo synthetic profiles projected on the demo subset 1KG reference PCA. The `list` contains 3 entries:

sample.id a `character` string representing the unique identifier of the synthetic profiles.

eigenvector.ref a `matrix` of `numeric` containing the eigenvectors for the reference profiles.

eigenvector a `matrix` of `numeric` containing the eigenvectors for the current synthetic profiles projected on the demo PCA 1KG reference profiles.

Details

This object can be used to test the `computeKNNRefSynthetic` function.

Value

The `list` containing the PCA result of demo synthetic profiles projected on the demo subset 1KG reference PCA. The `list` contains 3 entries:

sample.id a `character` string representing the unique identifier of the synthetic profiles.

eigenvector.ref a `matrix` of `numeric` containing the eigenvectors for the reference profiles.

eigenvector a `matrix` of `numeric` containing the eigenvectors for the current synthetic profiles projected on the demo PCA 1KG reference profiles.

See Also

`computeKNNRefSynthetic` for running a k-nearest neighbors analysis on a subset of the synthetic data set.

Examples

```
## Required library
library(gdsfmt)

## Load the demo PCA on the synthetic profiles projected on the
## demo 1KG reference PCA
data(demoPCASyntheticProfiles)

## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

## Open the Profile GDS file
gdsProfile <- snpgdsOpen(file.path(dataDir, "ex1.gds"))

# The name of the synthetic study
studyID <- "MYDATA.Synthetic"

## Projects synthetic profiles on 1KG PCA
results <- computeKNNRefSynthetic(gdsProfile=gdsProfile,
    listEigenvector=demoPCASyntheticProfiles,
    listCatPop=c("EAS", "EUR", "AFR", "AMR", "SAS"), studyIDSyn=studyID,
    spRef=demoKnownSuperPop1KG)

## The inferred ancestry for the synthetic profiles for different values
## of D and K
head(results$matKNN)

## Close Profile GDS file (important)
closefn.gds(gdsProfile)
```

---

demoPedigreeEx1 *The pedigree information about a demo profile called 'ex1'.*

Description

The object is a `data.frame`.

Usage

```
data(demoPedigreeEx1)
```

Format

The `data.frame` containing the information about a demo profile called 'ex1'. the `data.frame` has 5 columns:

Name.ID a `character` string representing the unique identifier of the profile.

Case.ID a `character` string representing the unique identifier of the case associated to the profile.

Sample.Type a `character` string describing the type of profile.

Diagnosis a `character` string describing the diagnosis of the profile.

Source a `character` string describing the source of the profile.

Details

This object can be used to test the `runExomeAncestry` function.

Value

The `data.frame` containing the information about a demo profile called 'ex1'. the `data.frame` has 5 columns:

Name.ID a `character` string representing the unique identifier of the profile.

Case.ID a `character` string representing the unique identifier of the case associated to the profile.

Sample.Type a `character` string describing the type of profile.

Diagnosis a `character` string describing the diagnosis of the profile.

Source a `character` string describing the source of the profile.

See Also

`runExomeAncestry` for running runs most steps leading to the ancestry inference call on a specific exome profile.

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

#####################################################################
## Load the information about the profile
#####################################################################
data(demoPedigreeEx1)
head(demoPedigreeEx1)

#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
path1KG <- file.path(dataDir, "tests")
```

```
fileReferenceGDS  <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
pathGeno <- file.path(dataDir, "example", "snpPileup")

#####################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")

pathOut <- file.path(tempdir(), "res.out")

#####################################################################
## A data frame containing general information about the study
## is also required. The data frame must have
## those 3 columns: "studyID", "study.desc", "study.platform"
#####################################################################
studyDF <- data.frame(study.id="MYDATA",
                      study.desc="Description",
                      study.platform="PLATFORM",
                      stringsAsFactors=FALSE)

#####################################################################
## Fix seed to ensure reproducible results
#####################################################################
set.seed(2043)

gds1KG <- snpgdsOpen(fileReferenceGDS)
dataRef <- select1KGPop(gds1KG, nbProfiles=2L)
closefn.gds(gds1KG)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]


        runExomeAncestry(pedStudy=demoPedigreeEx1, studyDF=studyDF,
            pathProfileGDS=pathProfileGDS,
            pathGeno=pathGeno, pathOut=pathOut,
            fileReferenceGDS=fileReferenceGDS,
            fileReferenceAnnotGDS=fileAnnotGDS,
            chrInfo=chrInfo, syntheticRefDF=dataRef,
            genoSource="snp-pileup")
```

```
        unlink(pathProfileGDS, recursive=TRUE, force=TRUE)
        unlink(pathOut, recursive=TRUE, force=TRUE)

}
```

---

eigenCount *Generic function for getting the eigenCount slot in a class*

Description

A generic function for getting the eigenCount slot in a S4 object.

Usage

```
eigenCount(x)
```

Arguments

x	a S4 object.

Value

a value from the eigenCount slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(eigenCount="character"))

# Create an instance
obj <- new("MyClass", eigenCount="123")

# Call the generic (assuming a method is implemented)
# eigenCount(obj)
```

eigenCount,RAIDSparam-method

A getter for the eigenCount slot in a RAIDSparam class

Description

A function for getting the eigenCount slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
eigenCount(x)
```

Arguments

x	a RAIDSparam object.

Value

a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the eigenCount slot for the object
eigenCount(paramDemo)
```

eigenCount<-

Generic function for replacement of eigenCount slot in a class

Description

A generic function for replacement of eigenCount slot in a S4 object.

Usage

```
eigenCount(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(eigenCount="character"))

# Create an instance
obj <- new("MyClass", eigenCount="123")

# Call the generic (assuming a method is implemented)
# eigenCount(obj) <- "333"
```

---

eigenCount<-,RAIDSparam-method

Setter function for replacement of eigenCount slot in a RAIDSparam class

---

Description

A function for replacement of eigenCount slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
eigenCount(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the eigenCount slot in the object
eigenCount(paramDemo) <-  33L
```

---

eigenCountSyn *Generic function for getting the eigenCountSyn slot in a class*

Description

A generic function for getting the eigenCountSyn slot in a S4 object.

Usage

```
eigenCountSyn(x)
```

Arguments

x	a S4 object.

Value

a value from the eigenCountSyn slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(eigenCountSyn="character"))

# Create an instance
obj <- new("MyClass", eigenCountSyn="123")

# Call the generic (assuming a method is implemented)
# eigenCountSyn(obj)
```

eigenCountSyn,RAIDSparam-method *A getter for the eigenCountSyn slot in a RAIDSparam class*

Description

A function for getting the eigenCountSyn slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
eigenCountSyn(x)
```

Arguments

x	a RAIDSparam object.

Value

a single integer indicating the number of eigenvectors TODO

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the eigenCountSyn slot for the object
eigenCountSyn(paramDemo)
```

eigenCountSyn<- *Generic function for replacement of eigenCountSyn slot in a class*

Description

A generic function for replacement of eigenCountSyn slot in a S4 object.

Usage

```
eigenCountSyn(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(eigenCountSyn="character"))

# Create an instance
obj <- new("MyClass", eigenCountSyn="123")

# Call the generic (assuming a method is implemented)
# eigenCountSyn(obj) <- "333"
```

---

eigenCountSyn<-,RAIDSparam-method

Setter function for replacement of eigenCountSyn slot in a RAIDSparam class

---

Description

A function for replacement of eigenCountSyn slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
eigenCountSyn(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single integer indicating the number of eigenvectors TODO

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the eigenCountSyn slot in the object
eigenCountSyn(paramDemo) <-  33L
```

---

estimateAllelicFraction

Estimate the allelic fraction of the pruned SNVs for a specific profile

---

Description

The function estimates the allelic fraction of the SNVs for a specific profile and add the information to the associated Profile GDS file. The allelic fraction estimation method is adapted to the type of study (DNA or RNA).

Usage

```
estimateAllelicFraction(
  gdsReference,
  gdsProfile,
  currentProfile,
  studyID,
  chrInfo,
  studyType = c("DNA", "RNA"),
  minCov = 10L,
  minProb = 0.999,
  eProb = 0.001,
  cutOffLOH = -5,
  cutOffHomoScore = -3,
  wAR = 9,
  cutOffAR = 3,
  gdsRefAnnot = NULL,
  blockID = NULL,
  verbose = FALSE
)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the opened Reference GDS file.
gdsProfile	an object of class gds.class (a GDS file), the opened Profile GDS file.

currentProfile — a character string corresponding to the sample identifier as used in pruningSample function.

studyID — a character string corresponding to the name of the study as used in pruningSample function.

chrInfo — a vector of integer values representing the length of the chromosomes. See 'details' section.

studyType — a character string representing the type of study. The possible choices are: "DNA" and "RNA". The type of study affects the way the estimation of the allelic fraction is done. Default: "DNA".

minCov — a single positive integer representing the minimum required coverage. Default: 10L.

minProb — a single numeric between 0 and 1 representing the probability that the calculated genotype call is correct. Default: 0.999.

eProb — a single numeric between 0 and 1 representing the probability of sequencing error. Default: 0.001.

cutOffLOH — a single numeric representing the cutoff, in log, for the homozygote score to assign a region as LOH. Default: -5.

cutOffHomoScore — a single numeric representing the cutoff, in log, that the SNVs in a block are called homozygote by error. Default: -3.

wAR — a single positive integer representing the size-1 of the window used to compute an empty box. Default: 9.

cutOffAR — a single numeric representing the cutoff, in log score, that the SNVs in a gene are allelic fraction different 0.5 Default: 3.

gdsRefAnnot — an object of class gds.class (a GDS file), the opened Reference SNV Annotation GDS file. This parameter is RNA specific. Default: NULL.

blockID — a character string corresponding to the block identifier in gdsRefAnnot. **This parameter is RNA specific.** Default: NULL

verbose — a logicial indicating if the function should print message when running. Default: FALSE.

Details

The chrInfo parameter contains the length of the chromosomes. The length of the chromosomes can be obtain through the seqlengths library.

As example, for hg38 genome:

```
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]
}
```

Value

The integer 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library for GDS
library(gdsfmt)

## Path to the demo 1KG GDS file located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

## Profile GDS file for one profile
fileProfile <- file.path(tempdir(), "ex1.gds")

## Copy the Profile GDS file demo that has been pruned and annotated
## into current directory
file.copy(file.path(dataDir, "ex1_demo_with_pruning_and_1KG_annot.gds"),
                            fileProfile)

## Open the reference GDS file (demo version)
gds1KG <- snpgdsOpen(fileGDS)

## Profile GDS file for one profile
profileGDS <- openfn.gds(fileProfile, readonly=FALSE)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]

    ## Estimate the allelic fraction of the pruned SNVs
    estimateAllelicFraction(gdsReference=gds1KG, gdsProfile=profileGDS,
        currentProfile="ex1", studyID="MYDATA", chrInfo=chrInfo,
        studyType="DNA", minCov=10L, minProb=0.999, eProb=0.001,
        cutOffLOH=-5, cutOffHomoScore=-3, wAR=9, cutOffAR=3,
        gdsRefAnnot=NULL, blockID=NULL)

    ## The allelic fraction is saved in the 'lap' node of Profile GDS file
    ## The 'lap' entry should be present
    profileGDS

    ## Close both GDS files (important)
    closefn.gds(profileGDS)
    closefn.gds(gds1KG)

    ## Remove Profile GDS file (created for demo purpose)
    unlink(fileProfile, force=TRUE)
```

```
}
```

`fieldPopInfAnc` *Generic function for getting the fieldPopInfAnc slot in a class*

Description

A generic function for getting the fieldPopInfAnc slot in a S4 object.

Usage

```
fieldPopInfAnc(x)
```

Arguments

x	a S4 object.

Value

a value from the fieldPopInfAnc slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fieldPopInfAnc="character"))

# Create an instance
obj <- new("MyClass", fieldPopInfAnc="123")

# Call the generic (assuming a method is implemented)
# fieldPopInfAnc(obj)
```

fieldPopInfAnc,RAIDSparam-method *A getter for the fieldPopInfAnc slot in a RAIDSparam class*

Description

A function for getting the fieldPopInfAnc slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
fieldPopInfAnc(x)
```

Arguments

x	a RAIDSparam object.

Value

a character string representing the name of the column in the data frame that contains the inferred super-population ancestry for the samples. The column should be present in the data frame.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the fieldPopInfAnc slot for the object
fieldPopInfAnc(paramDemo)
```

fieldPopInfAnc<- *Generic function for replacement of fieldPopInfAnc slot in a class*

Description

A generic function for replacement of fieldPopInfAnc slot in a S4 object.

Usage

```
fieldPopInfAnc(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fieldPopInfAnc="character"))

# Create an instance
obj <- new("MyClass", fieldPopInfAnc="123")

# Call the generic (assuming a method is implemented)
# fieldPopInfAnc(obj) <- "333"
```

---

fieldPopInfAnc<-,RAIDSparam-method

Setter function for replacement of fieldPopInfAnc slot in a RAIDSparam class

---

Description

A function for replacement of fieldPopInfAnc slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
fieldPopInfAnc(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a character string representing the name of the column in the data frame that contains the inferred super-population ancestry for the samples. The column should be present in the data frame.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the fieldPopInfAnc slot in the object
fieldPopInfAnc(paramDemo) <-  "superPOPULATION"
```

---

fieldPopInRef *Generic function for getting the fieldPopInRef slot in a class*

Description

A generic function for getting the fieldPopInRef slot in a S4 object.

Usage

```
fieldPopInRef(x)
```

Arguments

x	a S4 object.

Value

a value from the fieldPopInRef slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fieldPopInRef="character"))

# Create an instance
obj <- new("MyClass", fieldPopInRef="123")

# Call the generic (assuming a method is implemented)
# fieldPopInRef(obj)
```

fieldPopInRef,RAIDSparam-method *A getter for the fieldPopInRef slot in a RAIDSparam class*

Description

A function for getting the fieldPopInRef slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
fieldPopInRef(x)
```

Arguments

x	a RAIDSparam object.

Value

a character string representing the name of the column that contains the known ancestry for the reference profiles in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter).

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the fieldPopInRef slot for the object
fieldPopInRef(paramDemo)
```

fieldPopInRef<- *Generic function for replacement of fieldPopInRef slot in a class*

Description

A generic function for replacement of fieldPopInRef slot in a S4 object.

Usage

```
fieldPopInRef(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fieldPopInRef="character"))

# Create an instance
obj <- new("MyClass", fieldPopInRef="123")

# Call the generic (assuming a method is implemented)
# fieldPopInRef(obj) <- "333"
```

---

fieldPopInRef<-,RAIDSparam-method

Setter function for replacement of fieldPopInRef slot in a RAIDSparam class

---

Description

A function for replacement of fieldPopInRef slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
fieldPopInRef(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a character string representing the name of the column that contains the known ancestry for the reference profiles in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter).

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the fieldPopInRef slot in the object
fieldPopInRef(paramDemo) <-  "newPop"
```

---

fieldSubPop *Generic function for getting the fieldSubPop slot in a class*

Description

A generic function for getting the fieldSubPop slot in a S4 object.

Usage

```
fieldSubPop(x)
```

Arguments

x	a S4 object.

Value

a value from the fieldSubPop slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fieldSubPop="character"))

# Create an instance
obj <- new("MyClass", fieldSubPop="123")

# Call the generic (assuming a method is implemented)
# fieldSubPop(obj)
```

fieldSubPop,RAIDSparam-method *A getter for the fieldSubPop slot in a RAIDSparam class*

Description

A function for getting the fieldSubPop slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
fieldSubPop(x)
```

Arguments

x	a RAIDSparam object.

Value

a character string representing the name of the column in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter) that contains the sub-population information for the samples. The column should be present in the Population Reference GDS file.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the fieldSubPop slot for the object
fieldSubPop(paramDemo)
```

fieldSubPop<- *Generic function for replacement of fieldSubPop slot in a class*

Description

A generic function for replacement of fieldSubPop slot in a S4 object.

Usage

```
fieldSubPop(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fieldSubPop="character"))

# Create an instance
obj <- new("MyClass", fieldSubPop="123")

# Call the generic (assuming a method is implemented)
# fieldSubPop(obj) <- "333"
```

---

fieldSubPop<-,RAIDSparam-method

Setter function for replacement of fieldSubPop slot in a RAIDSparam class

---

Description

A function for replacement of fieldSubPop slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
fieldSubPop(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a character string representing the name of the column in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter) that contains the sub-population information for the samples. The column should be present in the Population Reference GDS file.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the fieldSubPop slot in the object
fieldSubPop(paramDemo) <-  "super.POPULATION.group"
```

---

fileReferenceAnnotGDS *Generic function for getting the fileReferenceAnnotGDS slot in a class*

Description

A generic function for getting the fileReferenceAnnotGDS slot in a S4 object.

Usage

```
fileReferenceAnnotGDS(x)
```

Arguments

x	a S4 object.

Value

a value from the fileReferenceAnnotGDS slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fileReferenceAnnotGDS="character"))

# Create an instance
obj <- new("MyClass", fileReferenceAnnotGDS="123")

# Call the generic (assuming a method is implemented)
# fileReferenceAnnotGDS(obj)
```

fileReferenceAnnotGDS,RAIDSparam-method

A getter for the fileReferenceAnnotGDS slot in a RAIDSparam class

Description

A function for getting the fileReferenceAnnotGDS slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
fileReferenceAnnotGDS(x)
```

Arguments

x	a RAIDSparam object.

Value

NULL or a character string representing the file name of the Population Reference GDS Annotation file.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the fileReferenceAnnotGDS slot for the object
fileReferenceAnnotGDS(paramDemo)
```

fileReferenceAnnotGDS<-

Generic function for replacement of fileReferenceAnnotGDS slot in a class

Description

A generic function for replacement of fileReferenceAnnotGDS slot in a S4 object.

Usage

```
fileReferenceAnnotGDS(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fileReferenceAnnotGDS="character"))

# Create an instance
obj <- new("MyClass", fileReferenceAnnotGDS="123")

# Call the generic (assuming a method is implemented)
# fileReferenceAnnotGDS(obj) <- "333"
```

---

fileReferenceAnnotGDS<-,RAIDSparam-method

Setter function for replacement of fileReferenceAnnotGDS slot in a RAIDSparam class

---

Description

A function for replacement of fileReferenceAnnotGDS slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
fileReferenceAnnotGDS(x) <- value
```

Arguments

x	a RAIDSparam object.
value	NULL or a character string representing the file name of the Population Reference GDS Annotation file.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Directory where demo GDS files are located
dataDir <- system.file("extdata", package="RAIDS")

## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the existing file to the fileReferenceAnnotGDS slot in the object
fileReferenceAnnotGDS(paramDemo) <- file.path(dataDir,
    "tests/ex1_good_small_1KG_Annot.gds")
```

---

fileReferenceGDS *Generic function for getting the fileReferenceGDS slot in a class*

Description

A generic function for getting the fileReferenceGDS slot in a S4 object.

Usage

```
fileReferenceGDS(x)
```

Arguments

x	a S4 object.

Value

a value from the fileReferenceGDS slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fileReferenceGDS="character"))

# Create an instance
obj <- new("MyClass", fileReferenceGDS="123")

# Call the generic (assuming a method is implemented)
# fileReferenceGDS(obj)
```

fileReferenceGDS,RAIDSparam-method *A getter for the fileReferenceGDS slot in a RAIDSparam class*

Description

A function for getting the fileReferenceGDS slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
fileReferenceGDS(x)
```

Arguments

x	a RAIDSparam object.

Value

NULL or a single character string representing the file name of the Reference GDS file.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the fileReferenceGDS slot for the object
fileReferenceGDS(paramDemo)
```

fileReferenceGDS<- *Generic function for replacement of fileReferenceGDS slot in a class*

Description

A generic function for replacement of fileReferenceGDS slot in a S4 object.

Usage

```
fileReferenceGDS(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(fileReferenceGDS="character"))

# Create an instance
obj <- new("MyClass", fileReferenceGDS="123")

# Call the generic (assuming a method is implemented)
# fileReferenceGDS(obj) <- "333"
```

---

fileReferenceGDS<-,RAIDSparam-method

Setter function for replacement of fileReferenceGDS slot in a RAIDSparam class

---

Description

A function for replacement of fileReferenceGDS slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
fileReferenceGDS(x) <- value
```

Arguments

x	a RAIDSparam object.
value	NULL or a single character string representing the file name of the Reference GDS file. If specified, the file must exist.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Directory where demo GDS files are located
dataDir <- system.file("extdata", package="RAIDS")

## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the existing file to the fileReferenceGDS slot in the object
fileReferenceGDS(paramDemo) <- file.path(dataDir,
    "tests/ex1_good_small_1KG.gds")
```

---

generateGDS1KG *Generate the GDS file that will contain the information from Reference data set (reference data set)*

Description

This function generates the GDS file that will contain the information from Reference. The function also add the samples information, the SNP information and the genotyping information into the GDS file.

Usage

```
generateGDS1KG(
  pathGeno = file.path("data", "sampleGeno"),
  filePedRDS,
  fileSNVIndex,
  fileSNVSelected,
  fileNameGDS,
  listSamples = NULL,
  verbose = FALSE
)
```

Arguments

pathGeno	a character string representing the path where the 1K genotyping files for each sample are located. The name of the genotyping files must correspond to the individual identification (Individual.ID) in the pedigree file. Default: "./data/sampleGeno".
filePedRDS	a character string representing the path and file name of the RDS file that contains the pedigree information. The file must exist. The file must be a RDS file.

`fileSNVIndex` a `character` string representing the path and file name of the RDS file that contains the indexes of the retained SNPs. The file must exist. The file must be a RDS file.

`fileSNVSelected` a `character` string representing the path and file name of the RDS file that contains the filtered SNP information. The file must exist. The file must be a RDS file.

`fileNameGDS` a `character` string representing the path and file name of the GDS file that will be created. The GDS file will contain the SNP information, the genotyping information and the pedigree information from 1000 Genomes. The extension of the file must be '.gds'.

`listSamples` a `vector` of `character` string corresponding to samples (must be the sample.ids) that will be retained and added to the GDS file. When `NULL`, all the samples are retained. Default: `NULL`.

`verbose` a `logical` indicating if the funciton must print messages when running. Default: `FALSE`.

Details

More information about GDS file format can be found at the Bioconductor gdsfmt website: https://bioconductor.org/packages/

Value

The integer `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Path to the CSV genoytype files
pathGeno <- file.path(dataDir, "demoProfileGenotypes")

## The RDS file containing the pedigree information
pedigreeFile <- file.path(dataDir, "PedigreeReferenceDemo.rds")

## The RDS file containing the indexes of the retained SNPs
snpIndexFile <- file.path(dataDir, "listSNPIndexes_Demo.rds")

## The RDS file containing the filtered SNP information
filterSNVFile <- file.path(dataDir, "mapSNVSelected_Demo.rds")

## Temporary Reference GDS file
tempRefGDS <- file.path(tempdir(), "1KG_TEMP.gds")

## Create a temporary Reference GDS file
```

```
generateGDS1KG(pathGeno=pathGeno, filePedRDS=pedigreeFile,
            fileSNVIndex=snpIndexFile, fileSNVSelected=filterSNVFile,
            fileNameGDS=tempRefGDS, listSamples=NULL)

## Remove temporary files
unlink(tempRefGDS, force=TRUE)
```

`generateMapSnvSel` *Generate the filter SNP information file in RDS format*

Description

The function applies a cut-off filter to the SNP information file to retain only the SNP that have a frequency superior or equal to the specified cut-off in at least one super population. The information about the retained SNPs is saved in a RDS format file. A RDS file containing the indexes of the retained SNP is also created.

Usage

```
generateMapSnvSel(cutOff = 0.01, fileSNV, fileSNPsRDS, fileFREQ)
```

Arguments

`cutOff`	a single `numeric` value, the cut-off for the frequency in at least one super population. Default: `0.01`.
`fileSNV`	a `character` string representing the path and file name of the bulk SNP information file from Reference. The file must be in text format. The file must exist.
`fileSNPsRDS`	a `character` string representing the path and file name of the RDS file that will contain the indexes of the retained SNPs. The file extension must be '.rds'.
`fileFREQ`	a `character` string representing the path and file name of the RDS file that will contain the filtered SNP information. The file extension must be '.rds'.

Details

The filtered SNP information RDS file (parameter `fileFREQ`), contains a `data.frame` with those columns:

CHROM a `character` string representing the chromosome where the SNV is located.

POS a `character` string representing the SNV position on the chromosome.

REF a `character` string representing the reference DNA base for the SNV.

ALT a `character` string representing the alternative DNA base for the SNV.\

EAS_AF a `character` string representing the allele frequency of the EAS super population.

AFR_AF a `character` string representing the allele frequency of the AFR super population.

AMR_AF a `character` string representing the allele frequency of the AMR super population.

SAS_AF a `character` string representing the allele frequency of the SAS super population.

Value

The integer `0` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Demo SNV information file used as input
snvFile <- file.path(dataDir, "matFreqSNV_Demo.txt.bz2")

## Temporary output files
## The first file contains the indexes of the retained SNPs
## The second file contains the filtered SNP information
snpIndexFile <- file.path(tempdir(), "listSNP_TEMP.rds")
filterSNVFile <- file.path(tempdir(), "mapSNVSel_TEMP.rds")

## Create a data.frame containing the information of the retained
## samples (samples with existing genotyping files)
generateMapSnvSel(cutOff=0.01, fileSNV=snvFile,
        fileSNPsRDS=snpIndexFile, fileFREQ=filterSNVFile)

## Remove temporary files
unlink(snpIndexFile, force=TRUE)
unlink(filterSNVFile, force=TRUE)
```

`generatePhase1KG2GDS` *Adding the phase information into the Reference GDS file*

Description

The function is adding the phase information into the Reference Phase GDS file. The phase information is extracted from a Reference GDS file and is added into a Reference Phase GDS file. An entry called 'phase' is added to the Reference Phase GDS file.

Usage

```
generatePhase1KG2GDS(
  gdsReference,
  gdsReferencePhase,
  pathGeno,
  fileSNVIndex,
  verbose = FALSE
)
```

Arguments

gdsReference	an object of class gds.class (GDS file), an opened Reference GDS file.
gdsReferencePhase	an object of class gds.class (GDS file), an opened Reference Phase GDS file.
pathGeno	a character string representing the path where the 1K genotyping files for each sample are located. The name of the genotyping files must correspond to the individual identification (Individual.ID) in the pedigree file. Default: "./data/sampleGeno".
fileSNVIndex	a character string representing the path and file name of the RDS file that contains the indexes of the retained SNPs. The file must exist. The file must be a RDS file.
verbose	a logicial indicating if the function should print messages when running. Default: FALSE.

Value

The function returns 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required package
library(gdsfmt)

## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Path where the demo genotype CSV files are located
pathGeno <- file.path(dataDir, "demoProfileGenotypes")

## The RDS file containing the pedigree information
pedigreeFile <- file.path(dataDir, "PedigreeReferenceDemo.rds")

## The RDS file containing the indexes of the retained SNPs
snpIndexFile <- file.path(dataDir, "listSNPIndexes_Demo.rds")

## The RDS file containing the filtered SNP information
filterSNVFile <- file.path(dataDir, "mapSNVSelected_Demo.rds")

## Temporary Reference GDS file containing reference information
fileReferenceGDS <- file.path(tempdir(), "1KG_TEMP_02.gds")

## Create a temporary Reference GDS file containing information from 1KG
generateGDS1KG(pathGeno=pathGeno, filePedRDS=pedigreeFile,
        fileSNVIndex=snpIndexFile, fileSNVSelected=filterSNVFile,
        fileNameGDS=fileReferenceGDS, listSamples=NULL)
```

```
## Temporary Phase GDS file that will contain the 1KG Phase information
fileRefPhaseGDS <- file.path(tempdir(), "1KG_TEMP_Phase_02.gds")

## Create Reference Phase GDS file
gdsPhase <- createfn.gds(fileRefPhaseGDS)

## Open Reference GDS file
gdsRef <- openfn.gds(fileReferenceGDS)

## Fill temporary Reference Phase GDS file
if (FALSE) {
    generatePhase1KG2GDS(gdsReference=gdsRef,
            gdsReferencePhase=gdsPhase,
            pathGeno=pathGeno, fileSNVIndex=snpIndexFile,
            verbose=FALSE)
}

## Close Reference Phase information file
closefn.gds(gdsPhase)

## Close Reference information file
closefn.gds(gdsRef)

## Remove temporary files
unlink(fileReferenceGDS, force=TRUE)
unlink(fileRefPhaseGDS, force=TRUE)
```

`generatePhaseRef` *Adding the phase information into the Reference GDS file*

Description

The function is adding the phase information into the Reference Phase GDS file. The phase information is extracted from a Reference GDS file and is added into a Reference Phase GDS file. An entry called 'phase' is added to the Reference Phase GDS file.

Usage

```
generatePhaseRef(
  fileReferenceGDS,
  fileReferenceAnnotGDS,
  pathGeno,
  fileSNVIndex,
  verbose = FALSE
)
```

Arguments

fileReferenceGDS	a character string representing the file name of the Reference GDS file. The file must exist.
fileReferenceAnnotGDS	a character string representing the file name of the Population Reference GDS Annotation file. The file must exist.
pathGeno	a character string representing the path where the 1K genotyping files for each sample are located. The name of the genotyping files must correspond to the individual identification (Individual.ID) in the pedigree file. Default: "./data/sampleGeno".
fileSNVIndex	a character string representing the path and file name of the RDS file that contains the indexes of the retained SNPs. The file must exist. The file must be a RDS file.
verbose	a logicial indicating if the function should print messages when running. Default: FALSE.

Value

The function returns 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Path where the demo genotype CSV files are located
pathGeno <- file.path(dataDir, "demoProfileGenotypes")

## The RDS file containing the pedigree information
pedigreeFile <- file.path(dataDir, "PedigreeReferenceDemo.rds")

## The RDS file containing the indexes of the retained SNPs
snpIndexFile <- file.path(dataDir, "listSNPIndexes_Demo.rds")

## The RDS file containing the filtered SNP information
filterSNVFile <- file.path(dataDir, "mapSNVSelected_Demo.rds")

## Temporary Reference GDS file containing reference information
fileReferenceGDS <- file.path(tempdir(), "1KG_TEMP_02.gds")

## Create a temporary Reference GDS file containing information from 1KG
generateGDS1KG(pathGeno=pathGeno, filePedRDS=pedigreeFile,
        fileSNVIndex=snpIndexFile, fileSNVSelected=filterSNVFile,
        fileNameGDS=fileReferenceGDS, listSamples=NULL)
```

```
## Temporary Phase GDS file that will contain the 1KG Phase information
fileRefPhaseGDS <- file.path(tempdir(), "1KG_TEMP_Phase_02.gds")


## Fill temporary Reference Phase GDS file
if (FALSE) {
    generatePhaseRef(fileReferenceGDS=fileReferenceGDS,
            fileReferenceAnnotGDS=fileRefPhaseGDS,
            pathGeno=pathGeno, fileSNVIndex=snpIndexFile,
            verbose=FALSE)
}


## Remove temporary files
unlink(fileReferenceGDS, force=TRUE)
unlink(fileRefPhaseGDS, force=TRUE)
```

`generateVCF` *Generate a VCF with the information from the SNPs that pass a cut-off threshold*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
generateVCF(freqCutoff = NULL, chr = NULL, fileOut, pRAIDS)
```

Arguments

`freqCutoff`	a single positive `numeric` specifying the cut-off to keep a SNP. If `NULL`, all SNPs are retained. Default: `NULL`.
`chr`	a `integer` between 1 to 22 chr1:22 specify SNP retained are on chr. If `NULL`, all SNPs are retained. Default: `NULL`.
`fileOut`	a `character` string representing the path and the file name of the new VCF.
`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters

Value

The integer `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Demo 1KG Reference GDS file
fileGDS <- openfn.gds(file.path(dataDir,
                    "PopulationReferenceDemo.gds"))

## Output VCF file that will be created (temporary)
vcfFile <- file.path(tempdir(), "Demo_TMP_01.vcf")

## Create a VCF file with the SNV dataset present in the GDS file
## No cutoff on frequency, so all SNVs are saved
snvListVCF(gdsReference=fileGDS, fileOut=vcfFile, offset=0L,
                    freqCutoff=NULL)

## Close GDS file (IMPORTANT)
closefn.gds(fileGDS)

## Remove temporary VCF file
unlink(vcfFile, force=TRUE)
```

---

genome *Generic function for getting the genome slot in a class*

Description

A generic function for getting the genome slot in a S4 object.

Usage

```
genome(x)
```

Arguments

x	a S4 object.

Value

a value from the genome slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(genome="character"))

# Create an instance
obj <- new("MyClass", genome="123")

# Call the generic (assuming a method is implemented)
# genome(obj)
```

---

genome,RAIDSparam-method

A getter for the genome slot in a RAIDSparam class

Description

A function for getting the genome slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
genome(x)
```

Arguments

x	a RAIDSparam object.

Value

character string with one possible value: 'HG38'.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the genome slot for the object
genome(paramDemo)
```

genoSource *Generic function for getting the genoSource slot in a class*

Description

A generic function for getting the genoSource slot in a S4 object.

Usage

```
genoSource(x)
```

Arguments

x	a S4 object.

Value

a value from the genoSource slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(genoSource="character"))

# Create an instance
obj <- new("MyClass", genoSource="123")

# Call the generic (assuming a method is implemented)
# genoSource(obj)
```

genoSource,RAIDSparam-method
A getter for the genoSource slot in a RAIDSparam class

Description

A function for getting the genoSource slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
genoSource(x)
```

Arguments

x	a RAIDSparam object.

Value

a single character string corresponding to the type of file with the genotype and the allele information of the profile that will be provided in the 'profileFile' slot. The valid options are: "VCF", "generic", "snp-pileup", and "bam".

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the genoSource slot for the object
genoSource(paramDemo)
```

---

genoSource<- *Generic function for replacement of genoSource slot in a class*

Description

A generic function for replacement of genoSource slot in a S4 object.

Usage

```
genoSource(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(genoSource="character"))

# Create an instance
obj <- new("MyClass", genoSource="123")

# Call the generic (assuming a method is implemented)
# genoSource(obj) <- "333"
```

`genoSource<-,RAIDSparam-method`

Setter function for replacement of genoSource slot in a RAIDSparam class

---

Description

A function for replacement of genoSource slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
genoSource(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `character` string corresponding to the type of file with the genotype and the allele information of the profile that will be provided in the 'profileFile' slot. The valid options are: "VCF", "generic", "snp-pileup", and "bam".

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the new study type to the genoSource slot in the object
genoSource(paramDemo) <- "bam"
```

genoType *Generic function for getting the genoType slot in a class*

Description

A generic function for getting the genoType slot in a S4 object.

Usage

```
genoType(x)
```

Arguments

x	a S4 object.

Value

a value from the genoType slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(genoType="character"))

# Create an instance
obj <- new("MyClass", genoType="123")

# Call the generic (assuming a method is implemented)
# genoType(obj)
```

genoType,RAIDSparam-method
A getter for the genoType slot in a RAIDSparam class

Description

A function for getting the genoType slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
genoType(x)
```

Arguments

x	a RAIDSparam object.

Value

a single character string representing the field in the GDS profile file that contains all the genotype information.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the genoType slot for the object
genoType(paramDemo)
```

---

genoType<- *Generic function for replacement of genoType slot in a class*

Description

A generic function for replacement of genoType slot in a S4 object.

Usage

```
genoType(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(genoType="character"))

# Create an instance
obj <- new("MyClass", genoType="123")

# Call the generic (assuming a method is implemented)
# genoType(obj) <- "333"
```

---

genoType<-,RAIDSparam-method *Setter function for replacement of genoType slot in a RAIDSparam class*

Description

A function for replacement of genoType slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
genoType(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single character string representing the field in the GDS profile file that contains all the genotype information.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the genoType slot in the object
genoType(paramDemo) <- "geno.REF"
```

getMatrixPopSynthetic *Generate a matrix of synthetic group per pop*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
getMatrixPopSynthetic(pRAIDS)
```

Arguments

pRAIDS	a RAIDSparam object with all the RAIDS parameters

Value

a matrix containing the sample identifiers and where each column is the name of a subcontinental population. The number of row corresponds to the number of samples for each subcontinental population.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
dataDir <- system.file("extdata/demoKNNSynthetic", package="RAIDS")

# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                      study.desc=studyID,
                      study.platform="NotDef",
                      stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                       Case.ID=c("ex1"),
                       Sample.Type=c("type"),
                       Diagnosis="NotDef",
                       Source=c("NotDef"),
                       stringsAsFactors=FALSE)
row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
```

```
                    pathProfileGDS=pathProfileGDS,
                    fileReferenceGDS=fileReferenceGDS)
matGr <- getMatrixPopSynthetic(pRAIDS)
dim(matGr)
```

---

getRef1KGPop *Extract the specified column from the 1KG GDS 'sample.ref' node for the reference profiles (real ancestry assignation)*

Description

The function extract the specified column for the 'sample.ref' node present in the Reference GDS file. The column must be present in the `data.frame` saved in the 'sample.ref' node. Only the information for the reference profiles is returned. The values represent the known ancestry assignation.

Usage

```
getRef1KGPop(gdsReference, popName = "superPop")
```

Arguments

`gdsReference`	an object of class gds.class (a GDS file), the opened Reference GDS file.
`popName`	a `character` string representing the name of the column that will be fetched in the `data.frame` present in the Reference GDS "sample.ref" node. The column must be present in the `data.frame`. Default: `"superPop"`.

Value

`vector` of `character` strings representing the content of the extracted column for the 1KG GDS 'sample.ref' node. The values represent the known ancestry assignation. The profile identifiers are used as names for the `vector`.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Open existing demo 1K GDS file with "sample.ref" node
nameFileGDS <- file.path(dataDir, "PopulationReferenceDemo.gds")
fileGDS <- snpgdsOpen(nameFileGDS)
```

```
## Extract super population information for the 1KG profiles
getRef1KGPop(gdsReference=fileGDS, popName="superPop")

## Close 1K GDS file
closefn.gds(fileGDS)
```

---

getRefSuperPop *Extract the from the 1KG GDS 'sample.ref' node for the reference profiles (real ancestry assignation)*

Description

The function extract the specified column for the 'sample.ref' node present in the Reference GDS file. The column must be present in the `data.frame` saved in the 'sample.ref' node. Only the information for the reference profiles is returned. The values represent the known ancestry assignation.

Usage

```
getRefSuperPop(fileReferenceGDS)
```

Arguments

`fileReferenceGDS`
: a `character` string representing the file name of the Reference GDS file. The file must exist.

Value

`vector` of `character` strings representing the content of the extracted column for the 1KG GDS 'sample.ref' node. The values represent the known ancestry assignation. The profile identifiers are used as names for the `vector`.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Open existing demo 1K GDS file with "sample.ref" node
nameFileGDS <- file.path(dataDir, "PopulationReferenceDemo.gds")

## Extract super population information for the 1KG profiles
getRefSuperPop(fileReferenceGDS=nameFileGDS)
```

getRefSuperPop2 *Extract the specified column from the 1KG GDS 'sample.ref' node for the reference profiles (real ancestry assignation)*

Description

The function extract the specified column for the 'sample.ref' node present in the Reference GDS file. The column must be present in the `data.frame` saved in the 'sample.ref' node. Only the information for the reference profiles is returned. The values represent the known ancestry assignation.

Usage

```
getRefSuperPop2(pRAIDS)
```

Arguments

pRAIDS	a `RAIDSparam` an object with all the RAIDS parameters

Value

`vector` of `character` strings representing the content of the extracted column for the 1KG GDS 'sample.ref' node. The values represent the known ancestry assignation. The profile identifiers are used as names for the `vector`.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
dataDir <- system.file("extdata/tests", package="RAIDS")

## Path for gds reference files
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

pRAIDS <- RAIDSparam(fileReferenceGDS=fileGDS)

spRef <- getRefSuperPop2(pRAIDS)

head(spRef)
```

groupChr1KGSNV *Merge the genotyping files per chromosome into one file*

Description

This function merge all the genotyping files associated to one specific sample into one file. That merged VCF file will be saved in a specified directory and will have the name of the sample. It will also be compressed (bzip). The function will merge the files for all samples present in the input directory.

Usage

```
groupChr1KGSNV(pathGenoChr, pathOut)
```

Arguments

pathGenoChr	a character string representing the path where the genotyping files for each sample and chromosome are located. The path must contains sub-directories (one per chromosome) and the genotyping files must be present in those sub-directories. The path must exists.
pathOut	a character string representing the path where the merged genotyping files for each sample will be created. The path must exists.

Value

The integer 0L when successful or FALSE if not.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo vcf files in this package
dataDir <- system.file("extdata", package="RAIDS")
pathGenoTar <- file.path(dataDir, "demoGenoChr", "demoGenoChr.tar")

## Path where the chromosomes files will be located
pathGeno <- file.path(tempdir(), "tempGeno")
dir.create(pathGeno, showWarnings=FALSE)

## Untar the file that contains the VCF files for 3 samples split by
## chromosome (one directory per chromosome)
untar(tarfile=pathGenoTar, exdir=pathGeno)

## Path where the output VCF file will be created is
## the same where the split VCF are (pathGeno)

## The files must not exist
```

```
if (!file.exists(file.path(pathGeno, "NA12003.csv.bz2")) &&
        !file.exists(file.path(pathGeno, "NA12004.csv.bz2")) &&
        !file.exists(file.path(pathGeno, "NA12005.csv.bz2"))) {

        ## Return 0 when successful
        ## The files "NA12003.csv.bz2", "NA12004.csv.bz2" and
        ## "NA12005.csv.bz2" should not be present in the current directory
        groupChr1KGSNV(pathGenoChr=pathGeno, pathOut=pathGeno)

        ## Validate that files have been created
        file.exists(file.path(pathGeno, "NA12003.csv.bz2"))
        file.exists(file.path(pathGeno, "NA12004.csv.bz2"))
        file.exists(file.path(pathGeno, "NA12005.csv.bz2"))

}

## Remove temporary directory
unlink(pathGeno, recursive=TRUE, force=TRUE)
```

`identifyRelative` *Identify genetically unrelated patients in GDS Reference file*

Description

The function identify patients that are genetically related in the Reference file. It generates a first RDS file with the list of unrelated patient. It also generates a second RDS file with the kinship coefficient between the patients.

Usage

```
identifyRelative(gds, maf = 0.05, thresh = 2^(-11/2), fileIBD, filePart)
```

Arguments

`gds`	an object of class `SNPRelate::SNPGDSFileClass`, the Reference GDS file.
`maf`	a single `numeric` representing the threshold for the minor allele frequency. Only the SNPs with ">= maf" will be used. Default: `0.05`.
`thresh`	a single `numeric` representing the threshold value used to decide if a pair of individuals is ancestrally divergent. Default: `2^(-11/2)`.
`fileIBD`	a `character` string representing the path and file name of the RDS file that will be created. The RDS file will contain the kinship coefficient between the patients. The extension of the file must be '.rds'.
`filePart`	a `character` string representing the path and file name of the RDS file that will be created. The RDS file will contain the information about the Reference patients that are unrelated. The file will contains two lists: the `list` of related samples, called `rels` and the list of unrelated samples, called `unrels`. The extension of the file must be '.rds'.

Value

NULL invisibly.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required package
library(gdsfmt)

## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Open existing demo Reference GDS file
fileGDS <- file.path(dataDir, "PopulationReferenceDemo.gds")
tmpGDS <- snpgdsOpen(fileGDS)

## Temporary output files
## The first RDS file will contain the list of unrelated patients
## The second RDS file will contain the kinship information between patients
patientTmpFile <-  "unrelatedPatients_TEMP.rds"
ibdTmpFile <- "ibd_TEMP.rds"

## Different code depending of the withr package availability
if (requireNamespace("withr", quietly=TRUE)) {

    ## Temporary output files
    ## The first RDS file will contain the list of unrelated patients
    ## The second RDS file will contain the kinship information
    ## between patients
    patientTmpFileLocal <- withr::local_file(patientTmpFile)
    ibdTmpFileLocal <- withr::local_file(ibdTmpFile)

    ## Identify unrelated patients in demo Reference GDS file
    identifyRelative(gds=tmpGDS, maf=0.05, thresh=2^(-11/2),
        fileIBD=ibdTmpFileLocal, filePart=patientTmpFileLocal)

    ## Close demo Reference GDS file
    closefn.gds(tmpGDS)

    ## Remove temporary files
    withr::deferred_run()

} else {

    ## Identify unrelated patients in demo Reference GDS file
    identifyRelative(gds=tmpGDS, maf=0.05, thresh=2^(-11/2),
        fileIBD=ibdTmpFile, filePart=patientTmpFile)

    ## Close demo Reference GDS file
```

```
    closefn.gds(tmpGDS)

    ## Remove temporary files
    unlink(patientTmpFile, force=TRUE)
    unlink(ibdTmpFile, force=TRUE)
}
```

`identifyRelativeRef` *Identify genetically unrelated patients in GDS Reference file*

Description

The function identify patients that are genetically related in the Reference file. It generates a first RDS file with the list of unrelated patient. It also generates a second RDS file with the kinship coefficient between the patients.

Usage

```
identifyRelativeRef(
  fileReferenceGDS,
  maf = 0.05,
  thresh = 2^(-11/2),
  fileIBD,
  filePart
)
```

Arguments

`fileReferenceGDS`	a `character` string representing the file name of the Reference GDS file. The file must exist.
`maf`	a single `numeric` representing the threshold for the minor allele frequency. Only the SNPs with ">= maf" will be used. Default: `0.05`.
`thresh`	a single `numeric` representing the threshold value used to decide if a pair of individuals is ancestrally divergent. Default: `2^(-11/2)`.
`fileIBD`	a `character` string representing the path and file name of the RDS file that will be created. The RDS file will contain the kinship coefficient between the patients. The extension of the file must be '.rds'.
`filePart`	a `character` string representing the path and file name of the RDS file that will be created. The RDS file will contain the information about the Reference patients that are unrelated. The file will contains two lists: the `list` of related samples, called `rels` and the list of unrelated samples, called `unrels`. The extension of the file must be '.rds'.

Value

`NULL` invisibly.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Open existing demo Reference GDS file
fileGDS <- file.path(dataDir, "PopulationReferenceDemo.gds")

## Temporary output files
## The first RDS file will contain the list of unrelated patients
## The second RDS file will contain the kinship information between patients
patientTmpFile <-  "unrelatedPatients_TEMP.rds"
ibdTmpFile <- "ibd_TEMP.rds"

## Different code depending of the withr package availability
if (requireNamespace("withr", quietly=TRUE)) {

    ## Temporary output files
    ## The first RDS file will contain the list of unrelated patients
    ## The second RDS file will contain the kinship information
    ## between patients
    patientTmpFileLocal <- withr::local_file(patientTmpFile)
    ibdTmpFileLocal <- withr::local_file(ibdTmpFile)

    ## Identify unrelated patients in demo Reference GDS file
    identifyRelativeRef(fileReferenceGDS=fileGDS, maf=0.05, thresh=2^(-11/2),
        fileIBD=ibdTmpFileLocal, filePart=patientTmpFileLocal)

    ## Remove temporary files
    withr::deferred_run()

} else {

    ## Identify unrelated patients in demo Reference GDS file
    identifyRelativeRef(fileReferenceGDS=fileGDS, maf=0.05, thresh=2^(-11/2),
        fileIBD=ibdTmpFile, filePart=patientTmpFile)

    ## Remove temporary files
    unlink(patientTmpFile, force=TRUE)
    unlink(ibdTmpFile, force=TRUE)
}
```

---

inferAncestry *Run most steps leading to the ancestry inference call on a specific DNA profile*

Description

This function runs most steps leading to the ancestry inference call on a specific RNA profile. First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population Reference GDS file.

Usage

```
inferAncestry(
  profileFile,
  pathProfileGDS,
  fileReferenceGDS,
  fileReferenceAnnotGDS,
  chrInfo,
  syntheticRefDF,
  genoSource = c("snp-pileup", "generic", "VCF", "bam"),
  np = 1L,
  verbose = FALSE
)
```

Arguments

profileFile	a character string representing the path and the file name of the genotype file or the bam if genoSource is snp-pileup the fine extension must be .txt.gz, if VCF the extension must be .vcf.gz
pathProfileGDS	a character string representing the path to the directory where the GDS Profile files will be created. Default: NULL.
fileReferenceGDS	a character string representing the file name of the Population Reference GDS file. The file must exist.
fileReferenceAnnotGDS	a character string representing the file name of the Population Reference GDS Annotation file. The file must exist.
chrInfo	a vector of positive integer values representing the length of the chromosomes. See 'details' section.
syntheticRefDF	a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns: **sample.id** a character string representing the sample identifier. **pop.group** a character string representing the subcontinental population assigned to the sample. **superPop** a character string representing the super-population assigned to the sample.
genoSource	a character string with four possible values: 'snp-pileup', 'generic', 'VCF' or 'bam'. It specifies if the genotype files are generated by snp-pileup (Facets) or are a generic format CSV file with at least those columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R' and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele and 'File1A'

is the depth of the specific alternative allele. Finally the file can be a VCF file with at least those genotype fields: GT, AD, DP.

`np` a single positive `integer` specifying the number of threads to be used. Default: `1L`.

`verbose` a `logical` indicating if messages should be printed to show how the different steps in the function. Default: `FALSE`.

Value

a `list` containing 4 entries:

`pcaSample` a `list` containing the information related to the eigenvectors. The `list` contains those 3 entries:

- `sample.id` a `character` string representing the unique identifier of the current profile.
- `eigenvector.ref` a `matrix` of `numeric` containing the eigenvectors for the reference profiles.
- `eigenvector` a `matrix` of `numeric` containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

`paraSample` a `list` containing the results with different `D` and `K` values that lead to optimal parameter selection. The `list` contains those entries:

- `dfPCA` a `data.frame` containing statistical results on all combined synthetic results done with a fixed value of `D` (the number of dimensions). The `data.frame` contains those columns:
  - `D` a `numeric` representing the value of `D` (the number of dimensions).
  - `median` a `numeric` representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed `D` value and all tested `K` values.
  - `mad` a `numeric` representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed `D` value and all tested `K` values.
  - `upQuartile` a `numeric` representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed `D` value and all tested `K` values.
  - `k` a `numeric` representing the optimal `K` value (the number of neighbors) for a fixed `D` value.
- `dfPop` a `data.frame` containing statistical results on all combined synthetic results done with different values of `D` (the number of dimensions) and `K` (the number of neighbors). The `data.frame` contains those columns:
  - `D` a `numeric` representing the value of `D` (the number of dimensions).
  - `K` a `numeric` representing the value of `K` (the number of neighbors).
  - `AUROC.min` a `numeric` representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of `D` and `K`.
  - `AUROC` a `numeric` representing the accuracy obtained by grouping all the synthetic results for the specified values of `D` and `K`.
  - `Accu.CM` a `numeric` representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of `D` and `K`.
- `dfAUROC` a `data.frame` the summary of the results by super-population. The `data.frame` contains those columns:

D a numeric representing the value of D (the number of dimensions).

K a numeric representing the value of K (the number of neighbors).

Call a character string representing the super-population.

L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

AUROC a numeric representing the AUROC obtained for the fixed values of super-population, D and K.

H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

D a numeric representing the optimal D value (the number of dimensions) for the specific profile.

K a numeric representing the optimal K value (the number of neighbors) for the specific profile.

listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

KNNSynthetic a data.frame containing the inferred ancestry for each synthetic data for different values of K and D. The data.frame contains those columns: "sample.id", "D", "K", "infer.superPop", "ref.superPop"

sample.id a character string representing the unique identifier of the current synthetic data.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

infer.superPop a character string representing the inferred ancestry for the specified D and K values.

ref.superPop a character string representing the known ancestry from the reference

Ancestry a data.frame containing the inferred ancestry for the current profile. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
demoProfileEx1 <- file.path(dataDir, "example", "snpPileup", "ex1.txt.gz")

#####################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")

#####################################################################
## Fix seed to ensure reproducible results
#####################################################################
set.seed(3043)

gds1KG <- snpgdsOpen(fileReferenceGDS)
dataRef <- select1KGPop(gds1KG, nbProfiles=2L)
closefn.gds(gds1KG)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]
```

```
res <- inferAncestry(profileFile=demoProfileEx1,
    pathProfileGDS=pathProfileGDS,
    fileReferenceGDS=fileReferenceGDS,
    fileReferenceAnnotGDS=fileAnnotGDS,
    chrInfo=chrInfo,
    syntheticRefDF=dataRef,
    genoSource="snp-pileup")

unlink(pathProfileGDS, recursive=TRUE, force=TRUE)


}
```

`inferAncestryDNA` *Run most steps leading to the ancestry inference call on a specific DNA profile (alias for inferAncestry)*

Description

This function runs most steps leading to the ancestry inference call on a specific RNA profile. First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population Reference GDS file.

Usage

```
inferAncestryDNA(
  profileFile,
  pathProfileGDS,
  fileReferenceGDS,
  fileReferenceAnnotGDS,
  chrInfo,
  syntheticRefDF,
  genoSource = c("snp-pileup", "generic", "VCF", "bam"),
  np = 1L,
  verbose = FALSE
)
```

Arguments

`profileFile`	a `character` string representing the path and the file name of the genotype file or the bam if genoSource is snp-pileup the fine extension must be .txt.gz, if VCF the extension must be .vcf.gz
`pathProfileGDS`	a `character` string representing the path to the directory where the GDS Profile files will be created. Default: `NULL`.

fileReferenceGDS
: a character string representing the file name of the Population Reference GDS file. The file must exist.

fileReferenceAnnotGDS
: a character string representing the file name of the Population Reference GDS Annotation file. The file must exist.

chrInfo
: a vector of positive integer values representing the length of the chromosomes. See 'details' section.

syntheticRefDF
: a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns:

  sample.id a character string representing the sample identifier.

  pop.group a character string representing the subcontinental population assigned to the sample.

  superPop a character string representing the super-population assigned to the sample.

genoSource
: a character string with four possible values: 'snp-pileup', 'generic', 'VCF' or 'bam'. It specifies if the genotype files are generated by snp-pileup (Facets) or are a generic format CSV file with at least those columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R' and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele and 'File1A' is the depth of the specific alternative allele. Finally the file can be a VCF file with at least those genotype fields: GT, AD, DP.

np
: a single positive integer specifying the number of threads to be used. Default: 1L.

verbose
: a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Value

a list containing 4 entries:

pcaSample a list containing the information related to the eigenvectors. The list contains those 3 entries:

  sample.id a character string representing the unique identifier of the current profile.

  eigenvector.ref a matrix of numeric containing the eigenvectors for the reference profiles.

  eigenvector a matrix of numeric containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

paraSample a list containing the results with different D and K values that lead to optimal parameter selection. The list contains those entries:

  dfPCA a data.frame containing statistical results on all combined synthetic results done with a fixed value of D (the number of dimensions). The data.frame contains those columns:

  D a numeric representing the value of D (the number of dimensions).

  median a numeric representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.

mad a numeric representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.

upQuartile a numeric representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed D value and all tested K values.

k a numeric representing the optimal K value (the number of neighbors) for a fixed D value.

dfPop a data.frame containing statistical results on all combined synthetic results done with different values of D (the number of dimensions) and K (the number of neighbors). The data.frame contains those columns:

D a numeric representing the value of D (the number of dimensions).

K a numeric representing the value of K (the number of neighbors).

AUROC.min a numeric representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of D and K.

AUROC a numeric representing the accuracy obtained by grouping all the synthetic results for the specified values of D and K.

Accu.CM a numeric representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of D and K.

dfAUROC a data.frame the summary of the results by super-population. The data.frame contains those columns:

D a numeric representing the value of D (the number of dimensions).

K a numeric representing the value of K (the number of neighbors).

Call a character string representing the super-population.

L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

AUROC a numeric representing the AUROC obtained for the fixed values of super-population, D and K.

H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

D a numeric representing the optimal D value (the number of dimensions) for the specific profile.

K a numeric representing the optimal K value (the number of neighbors) for the specific profile.

listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

KNNSynthetic a data.frame containing the inferred ancestry for each synthetic data for different values of K and D. The data.frame contains those columns: "sample.id", "D", "K", "infer.superPop", "ref.superPop"

`sample.id` a `character` string representing the unique identifier of the current synthetic data.

`D` a `numeric` representing the value of `D` (the number of dimensions) used to infer the ancestry.

`K` a `numeric` representing the value of `K` (the number of neighbors) used to infer the ancestry.

`infer.superPop` a `character` string representing the inferred ancestry for the specified `D` and `K` values.

`ref.superPop` a `character` string representing the known ancestry from the reference

`Ancestry` a `data.frame` containing the inferred ancestry for the current profile. The `data.frame` contains those columns:

`sample.id` a `character` string representing the unique identifier of the current profile.

`D` a `numeric` representing the value of `D` (the number of dimensions) used to infer the ancestry.

`K` a `numeric` representing the value of `K` (the number of neighbors) used to infer the ancestry.

`SuperPop` a `character` string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
demoProfileEx1 <- file.path(dataDir, "example", "snpPileup", "ex1.txt.gz")

#####################################################################
## The path where the Profile GDS Files (one per sample)
```

```
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")

#####################################################################
## Fix seed to ensure reproducible results
#####################################################################
set.seed(3043)

gds1KG <- snpgdsOpen(fileReferenceGDS)
dataRef <- select1KGPop(gds1KG, nbProfiles=2L)
closefn.gds(gds1KG)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]



        res <- inferAncestryDNA(profileFile=demoProfileEx1,
            pathProfileGDS=pathProfileGDS,
            fileReferenceGDS=fileReferenceGDS,
            fileReferenceAnnotGDS=fileAnnotGDS,
            chrInfo=chrInfo,
            syntheticRefDF=dataRef,
            genoSource="snp-pileup")

        unlink(pathProfileGDS, recursive=TRUE, force=TRUE)


}
```

`inferAncestryGeneAware` *Run most steps leading to the ancestry inference call on a specific RNA profile*

Description

This function runs most steps leading to the ancestry inference call on a specific RNA profile. First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population Reference GDS file.

Usage

```
inferAncestryGeneAware(
  profileFile,
  pathProfileGDS,
  fileReferenceGDS,
  fileReferenceAnnotGDS,
  chrInfo,
  syntheticRefDF,
  genoSource = c("snp-pileup", "generic", "VCF", "bam"),
  np = 1L,
  blockTypeID,
  verbose = FALSE
)
```

Arguments

profileFile	a character string representing the path and the file name of the genotype file or the bam if genoSource is snp-pileup the fine extension must be .txt.gz, if VCF the extension must be .vcf.gz
pathProfileGDS	a character string representing the path to the directory where the GDS Profile files will be created. Default: NULL.
fileReferenceGDS	a character string representing the file name of the Population Reference GDS file. The file must exist.
fileReferenceAnnotGDS	a character string representing the file name of the Population Reference GDS Annotation file. The file must exist.
chrInfo	a vector of positive integer values representing the length of the chromosomes. See 'details' section.
syntheticRefDF	a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns: **sample.id** a character string representing the sample identifier. **pop.group** a character string representing the subcontinental population assigned to the sample. **superPop** a character string representing the super-population assigned to the sample.
genoSource	a character string with four possible values: 'snp-pileup', 'generic', 'VCF' or 'bam'. It specifies if the genotype files are generated by snp-pileup (Facets) or are a generic format CSV file with at least those columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R' and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele and 'File1A' is the depth of the specific alternative allele. Finally the file can be a VCF file with at least those genotype fields: GT, AD, DP.
np	a single positive integer specifying the number of threads to be used. Default: 1L.

`blockTypeID` a `character` string corresponding to the block type used to extract the block identifiers. The block type must be present in the GDS Reference Annotation file.

`verbose` a `logical` indicating if messages should be printed to show how the different steps in the function. Default: `FALSE`.

Details

The runExomeAncestry() function generates 3 types of files in the OUTPUT directory.

Ancestry Inference The ancestry inference CSV file (".Ancestry.csv" file)

Inference Informaton The inference information RDS file (".infoCall.rds" file)

Synthetic Information The parameter information RDS files from the synthetic inference ("KNN.synt.*.rds" files in a sub-directory)

In addition, a sub-directory (named using the profile ID) is also created.

Value

a `list` containing 4 entries:

`pcaSample` a `list` containing the information related to the eigenvectors. The `list` contains those 3 entries:

- `sample.id` a `character` string representing the unique identifier of the current profile.
- `eigenvector.ref` a `matrix` of `numeric` containing the eigenvectors for the reference profiles.
- `eigenvector` a `matrix` of `numeric` containing the eigenvectors for the current profile projected on the PCA from the reference profiles.

`paraSample` a `list` containing the results with different `D` and `K` values that lead to optimal parameter selection. The `list` contains those entries:

- `dfPCA` a `data.frame` containing statistical results on all combined synthetic results done with a fixed value of `D` (the number of dimensions). The `data.frame` contains those columns:
  - `D` a `numeric` representing the value of `D` (the number of dimensions).
  - `median` a `numeric` representing the median of the minimum AUROC obtained (within super populations) for all combination of the fixed `D` value and all tested `K` values.
  - `mad` a `numeric` representing the MAD of the minimum AUROC obtained (within super populations) for all combination of the fixed `D` value and all tested `K` values.
  - `upQuartile` a `numeric` representing the upper quartile of the minimum AUROC obtained (within super populations) for all combination of the fixed `D` value and all tested `K` values.
  - `k` a `numeric` representing the optimal `K` value (the number of neighbors) for a fixed `D` value.
- `dfPop` a `data.frame` containing statistical results on all combined synthetic results done with different values of `D` (the number of dimensions) and `K` (the number of neighbors). The `data.frame` contains those columns:
  - `D` a `numeric` representing the value of `D` (the number of dimensions).
  - `K` a `numeric` representing the value of `K` (the number of neighbors).

AUROC.min a numeric representing the minimum accuracy obtained by grouping all the synthetic results by super-populations, for the specified values of D and K.

AUROC a numeric representing the accuracy obtained by grouping all the synthetic results for the specified values of D and K.

Accu.CM a numeric representing the value of accuracy of the confusion matrix obtained by grouping all the synthetic results for the specified values of D and K.

dfAUROC a data.frame the summary of the results by super-population. The data.frame contains those columns:

D a numeric representing the value of D (the number of dimensions).

K a numeric representing the value of K (the number of neighbors).

Call a character string representing the super-population.

L a numeric representing the lower value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

AUROC a numeric representing the AUROC obtained for the fixed values of super-population, D and K.

H a numeric representing the higher value of the 95% confidence interval for the AUROC obtained for the fixed values of super-population, D and K.

D a numeric representing the optimal D value (the number of dimensions) for the specific profile.

K a numeric representing the optimal K value (the number of neighbors) for the specific profile.

listD a numeric representing the optimal D values (the number of dimensions) for the specific profile. More than one D is possible.

KNNSample a data.frame containing the inferred ancestry for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry for the specified D and K values.

KNNSynthetic a data.frame containing the inferred ancestry for each synthetic data for different values of K and D. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current synthetic data.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

infer.superPop a character string representing the inferred ancestry for the specified D and K values.

ref.superPop a character string representing the known ancestry from the reference

Ancestry a data.frame containing the inferred ancestry for the current profile. The data.frame contains those columns:

sample.id a character string representing the unique identifier of the current profile.

D a numeric representing the value of D (the number of dimensions) used to infer the ancestry.

K a numeric representing the value of K (the number of neighbors) used to infer the ancestry.

SuperPop a character string representing the inferred ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")


#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
demoProfileEx1 <- file.path(dataDir, "example", "snpPileup", "ex1.txt.gz")

#####################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")

#########################################################################
## Fix seed to ensure reproducible results
#########################################################################
set.seed(3043)

gds1KG <- snpgdsOpen(fileReferenceGDS)
dataRef <- select1KGPop(gds1KG, nbProfiles=2L)
closefn.gds(gds1KG)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
```

```
 requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

## Chromosome length information
## chr23 is chrX, chr24 is chrY and chrM is 25
chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]



    res <- inferAncestryGeneAware(profileFile=demoProfileEx1,
        pathProfileGDS=pathProfileGDS,
        fileReferenceGDS=fileReferenceGDS,
        fileReferenceAnnotGDS=fileAnnotGDS,
        chrInfo=chrInfo,
        syntheticRefDF=dataRef,
        blockTypeID="GeneS.Ensembl.Hsapiens.v86",
        genoSource="snp-pileup")

    unlink(pathProfileGDS, recursive=TRUE, force=TRUE)


}
```

`inferenceType` *Generic function for getting the inferenceType slot in a class*

Description

A generic function for getting the inferenceType slot in a S4 object.

Usage

```
inferenceType(x)
```

Arguments

x	a S4 object.

Value

a value from the inferenceType slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(inferenceType="character"))

# Create an instance
obj <- new("MyClass", inferenceType="123")

# Call the generic (assuming a method is implemented)
# inferenceType(obj)
```

`inferenceType,RAIDSparam-method`
A getter for the inferenceType slot in a RAIDSparam class

Description

A function for getting the inferenceType slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
inferenceType(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

a single `character` string representing the genotype ancestry inference method.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the inferenceType slot for the object
inferenceType(paramDemo)
```

inferenceType<- *Generic function for replacement of inferenceType slot in a class*

Description

A generic function for replacement of inferenceType slot in a S4 object.

Usage

```
inferenceType(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(inferenceType="character"))

# Create an instance
obj <- new("MyClass", inferenceType="123")

# Call the generic (assuming a method is implemented)
# inferenceType(obj) <- "333"
```

inferenceType<-,RAIDSparam-method *Setter function for replacement of inferenceType slot in a RAIDSparam class*

Description

A function for replacement of inferenceType slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
inferenceType(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single character string representing the genotype ancestry inference method. The two possible values are 'PCAknn' and 'haploAdmixture'.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Directory where demo GDS files are located
dataDir <- system.file("extdata", package="RAIDS")

## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the inferenceType slot in the object
inferenceType(paramDemo) <- "haploAdmixture"
```

---

kList *Generic function for getting the kList slot in a class*

Description

A generic function for getting the kList slot in a S4 object.

Usage

```
kList(x)
```

Arguments

x	a S4 object.

Value

a value from the kList slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(kList="character"))

# Create an instance
obj <- new("MyClass", kList="123")

# Call the generic (assuming a method is implemented)
# kList(obj)
```

---

kList,RAIDSparam-method

A getter for the kList slot in a RAIDSparam class

Description

A function for getting the kList slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
kList(x)
```

Arguments

x	a RAIDSparam object.

Value

a vector of positive integer representing the values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the kList slot for the object
kList(paramDemo)
```

kList<- *Generic function for replacement of kList slot in a class*

Description

A generic function for replacement of kList slot in a S4 object.

Usage

```
kList(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(kList="character"))

# Create an instance
obj <- new("MyClass", kList="123")

# Call the generic (assuming a method is implemented)
# kList(obj) <- "333"
```

kList<-,RAIDSparam-method *Setter function for replacement of kList slot in a RAIDSparam class*

Description

A function for replacement of kList slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
kList(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a vector of positive integer representing the values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the kList slot in the object
kList(paramDemo) <-  c(2L, 3L, 4L, 5L)
```

---

listPos *Generic function for getting the listPos slot in a class*

Description

A generic function for getting the listPos slot in a S4 object.

Usage

```
listPos(x)
```

Arguments

x	a S4 object.

Value

a value from the listPos slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(listPos="character"))

# Create an instance
obj <- new("MyClass", listPos="123")

# Call the generic (assuming a method is implemented)
# listPos(obj)
```

---

listPos,RAIDSparam-method

A getter for the listPos slot in a RAIDSparam class

Description

A function for getting the listPos slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
listPos(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive integer specifying the number of threads to be used.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the listPos slot for the object
listPos(paramDemo)
```

listPos<- *Generic function for replacement of listPos slot in a class*

Description

A generic function for replacement of listPos slot in a S4 object.

Usage

```
listPos(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(listPos="character"))

# Create an instance
obj <- new("MyClass", listPos="123")

# Call the generic (assuming a method is implemented)
# listPos(obj) <- "333"
```

listPos<-,RAIDSparam-method *Setter function for replacement of listPos slot in a RAIDSparam class*

Description

A function for replacement of listPos slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
listPos(x) <- value
```

Arguments

x	a RAIDSparam object.
value	NULL or a data.frame containing 2 columns named: "snp.chromosome" and "snp.position". The first column, called "snp.chromosome", contains the name of the chromosome where the SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new values to the listPos slot in the object
listPos(paramDemo) <-data.frame(snp.chromosome=c("1", "1"),
    snp.position=c(12232, 433333))
```

---

matKNNSynthetic *A small* data.frame *containing the inferred ancestry on the synthetic profiles.*

Description

The object is a data.frame with 4 columns.

Usage

```
data(matKNNSynthetic)
```

Format

The data.frame containing the information about the synthetic profiles. The data.frame contains 4 columns:

sample.id a character string representing the unique synthetic profile identifier.

D a numeric representing the number of dimensions used to infer the ancestry of the synthetic profile.

K a numeric representing the number of neighbors used to infer the ancestry of the synthetic profile.

SuperPop a character string representing the inferred ancestry of the synthetic profile for the specific D and K values.

Details

This dataset can be used to test the computeSyntheticROC function.

Value

The data.frame containing the information about the synthetic profiles. The data.frame contains 4 columns:

sample.id a character string representing the unique synthetic profile identifier.

D a numeric representing the number of dimensions used to infer the ancestry of the synthetic profile.

K a numeric representing the number of neighbors used to infer the ancestry of the synthetic profile.

SuperPop a character string representing the inferred ancestry of the synthetic profile for the specific D and K values.

See Also

computeSyntheticROC for calculating the AUROC of the inferences for specific values of D and K using the inferred ancestry results from the synthetic profiles

Examples

```
## Loading demo dataset containing pedigree information for synthetic
## profiles
data(pedSynthetic)

## Loading demo dataset containing the inferred ancestry results
## for the synthetic data
data(matKNNSynthetic)

## Retain one K and one D value
matKNN <- matKNNSynthetic[matKNNSynthetic$D == 5 & matKNNSynthetic$K == 4, ]

## Compile statistics from the
## synthetic profiles for fixed values of D and K
results <- RAIDS:::computeSyntheticROC(matKNN=matKNN,
    matKNNAncestryColumn="SuperPop",
```

```
    pedCall=pedSynthetic, pedCallAncestryColumn="superPop",
    listCall=c("EAS", "EUR", "AFR", "AMR", "SAS"))

results$matAUROC.All
results$matAUROC.Call
results$listROC.Call
```

`minCov` *Generic function for getting the minCov slot in a class*

Description

A generic function for getting the minCov slot in a S4 object.

Usage

```
minCov(x)
```

Arguments

`x`	a S4 object.

Value

a value from the minCov slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(minCov="character"))

# Create an instance
obj <- new("MyClass", minCov="123")

# Call the generic (assuming a method is implemented)
# minCov(obj)
```

minCov,RAIDSparam-method *A getter for the minCov slot in a RAIDSparam class*

Description

A function for getting the minCov slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
minCov(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive integer representing the minimum required coverage.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the minCov slot for the object
minCov(paramDemo)
```

minCov<- *Generic function for replacement of minCov slot in a class*

Description

A generic function for replacement of minCov slot in a S4 object.

Usage

```
minCov(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(minCov="character"))

# Create an instance
obj <- new("MyClass", minCov="123")

# Call the generic (assuming a method is implemented)
# minCov(obj) <- "333"
```

---

minCov<-,RAIDSparam-method

Setter function for replacement of minCov slot in a RAIDSparam class

---

Description

A function for replacement of minCov slot in a RAIDSparam class.

A function for replacement of minCov slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
minCov(x) <- value

## S4 replacement method for signature 'RAIDSparam'
minCov(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single positive integer representing the minimum required coverage.

Value

the modified RAIDSparam object when the new value is valid.

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the minCov slot in the object
minCov(paramDemo) <- 3L


## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the minCov slot in the object
minCov(paramDemo) <- 3L
```

---

minProb *Generic function for getting the minProb slot in a class*

Description

A generic function for getting the minProb slot in a S4 object.

Usage

```
minProb(x)
```

Arguments

x	a S4 object.

Value

a value from the minProb slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(minProb="character"))

# Create an instance
obj <- new("MyClass", minProb="123")

# Call the generic (assuming a method is implemented)
# minProb(obj)
```

`minProb,RAIDSparam-method`
A getter for the minProb slot in a RAIDSparam class

Description

A function for getting the minProb slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
minProb(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

a single `numeric` between `0` and `1` representing the probability that the calculated genotype call is correct.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the minProb slot for the object
minProb(paramDemo)
```

minProb<- *Generic function for replacement of minProb slot in a class*

Description

A generic function for replacement of minProb slot in a S4 object.

Usage

```
minProb(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(minProb="character"))

# Create an instance
obj <- new("MyClass", minProb="123")

# Call the generic (assuming a method is implemented)
# minProb(obj) <- "333"
```

minProb<-,RAIDSparam-method *Setter function for replacement of minProb slot in a RAIDSparam class*

Description

A function for replacement of minProb slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
minProb(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single numeric between 0 and 1 representing the probability that the calculated genotype call is correct.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the minProb slot in the object
minProb(paramDemo) <- 0.03
```

---

nbSim *Generic function for getting the nbSim slot in a class*

Description

A generic function for getting the nbSim slot in a S4 object.

Usage

```
nbSim(x)
```

Arguments

x	a S4 object.

Value

a value from the nbSim slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(nbSim="character"))

# Create an instance
obj <- new("MyClass", nbSim="123")

# Call the generic (assuming a method is implemented)
# nbSim(obj)
```

---

nbSim,RAIDSparam-method

A getter for the nbSim slot in a RAIDSparam class

Description

A function for getting the nbSim slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
nbSim(x)
```

Arguments

x	a RAIDSparam object.

Value

a positive integer representing TODO.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the nbSim slot for the object
nbSim(paramDemo)
```

nbSim<- *Generic function for replacement of nbSim slot in a class*

Description

A generic function for replacement of nbSim slot in a S4 object.

Usage

```
nbSim(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(nbSim="character"))

# Create an instance
obj <- new("MyClass", nbSim="123")

# Call the generic (assuming a method is implemented)
# nbSim(obj) <- "333"
```

nbSim<-,RAIDSparam-method
Setter function for replacement of nbSim slot in a RAIDSparam class

Description

A function for replacement of nbSim slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
nbSim(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a positive integer representing TODO.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the nbSim slot in the object
nbSim(paramDemo) <- 1L
```

---

np *Generic function for getting the np slot in a class*

Description

A generic function for getting the np slot in a S4 object.

Usage

```
np(x)
```

Arguments

x	a S4 object.

Value

a value from the np slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(np="character"))

# Create an instance
obj <- new("MyClass", np="123")

# Call the generic (assuming a method is implemented)
# np(obj)
```

---

np,RAIDSparam-method *A getter for the np slot in a RAIDSparam class*

Description

A function for getting the np slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
np(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive integer specifying the number of threads to be used.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the np slot for the object
np(paramDemo)
```

np<- *Generic function for replacement of np slot in a class*

Description

A generic function for replacement of np slot in a S4 object.

Usage

```
np(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(np="character"))

# Create an instance
obj <- new("MyClass", np="123")

# Call the generic (assuming a method is implemented)
# np(obj) <- "333"
```

np<-,RAIDSparam-method *Setter function for replacement of np slot in a RAIDSparam class*

Description

A function for replacement of np slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
np(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single positive integer specifying the number of threads to be used.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the np slot in the object
np(paramDemo) <-3L
```

---

paramAncestry *Generic function for getting the paramAncestry slot in a class*

Description

A generic function for getting the paramAncestry slot in a S4 object.

Usage

```
paramAncestry(x)
```

Arguments

x	a S4 object.

Value

a value from the paramAncestry slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(paramAncestry="character"))

# Create an instance
obj <- new("MyClass", paramAncestry="123")

# Call the generic (assuming a method is implemented)
# paramAncestry(obj)
```

---

paramAncestry,RAIDSparam-method

A getter for the paramAncestry slot in a RAIDSparam class

Description

A function for getting the paramAncestry slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
paramAncestry(x)
```

Arguments

x	a RAIDSparam object.

Value

a list of parameters related to ancestry.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the paramAncestry slot for the object
paramAncestry(paramDemo)
```

paramAncestry<- *Generic function for replacement of paramAncestry slot in a class*

Description

A generic function for replacement of paramAncestry slot in a S4 object.

Usage

```
paramAncestry(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(paramAncestry="character"))

# Create an instance
obj <- new("MyClass", paramAncestry="123")

# Call the generic (assuming a method is implemented)
# paramAncestry(obj) <- "333"
```

paramAncestry<-,RAIDSparam-method *Setter function for replacement of paramAncestry slot in a RAIDSparam class*

Description

A function for replacement of paramAncestry slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
paramAncestry(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a vector of positive integer values representing the length of the chromosomes.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the new chrInfo to the paramAncestry slot in the object
paramAncestry(paramDemo) <- list(ScanBamParam=NULL, PileupParam=NULL,
    yieldSize=1000023)
```

---

pathADMIXTURE *Generic function for getting the pathADMIXTURE slot in a class*

Description

A generic function for getting the pathADMIXTURE slot in a S4 object.

Usage

```
pathADMIXTURE(x)
```

Arguments

x	a S4 object.

Value

a value from the pathADMIXTURE slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pathADMIXTURE="character"))

# Create an instance
obj <- new("MyClass", pathADMIXTURE="123")

# Call the generic (assuming a method is implemented)
# pathADMIXTURE(obj)
```

---

pathADMIXTURE,RAIDSparam-method

A getter for the pathADMIXTURE slot in a RAIDSparam class

Description

A function for getting the pathADMIXTURE slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
pathADMIXTURE(x)
```

Arguments

x	a RAIDSparam object.

Value

a character string representing the path pf the software ADMIXTURE and the binary.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pathADMIXTURE slot for the object
pathADMIXTURE(paramDemo)
```

pathADMIXTURE<- *Generic function for replacement of pathADMIXTURE slot in a class*

Description

A generic function for replacement of pathADMIXTURE slot in a S4 object.

Usage

```
pathADMIXTURE(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pathADMIXTURE="character"))

# Create an instance
obj <- new("MyClass", pathADMIXTURE="123")

# Call the generic (assuming a method is implemented)
# pathADMIXTURE(obj) <- "333"
```

pathADMIXTURE<-,RAIDSparam-method *Setter function for replacement of pathADMIXTURE slot in a RAIDSparam class*

Description

A function for replacement of pathADMIXTURE slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
pathADMIXTURE(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a character string representing the path pf the software ADMIXTURE and the binary. If specified, the file must exist.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Directory where demo GDS files are located
dataDir <- system.file("extdata", package="RAIDS")

## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the existing path to the pathADMIXTURE slot in the object
pathADMIXTURE(paramDemo) <- file.path(dataDir)
```

---

pathProfileGDS *Generic function for getting the pathProfileGDS slot in a class*

Description

A generic function for getting the pathProfileGDS slot in a S4 object.

Usage

```
pathProfileGDS(x)
```

Arguments

x	a S4 object.

Value

a value from the pathProfileGDS slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pathProfileGDS="character"))

# Create an instance
obj <- new("MyClass", pathProfileGDS="123")

# Call the generic (assuming a method is implemented)
# pathProfileGDS(obj)
```

---

pathProfileGDS,RAIDSparam-method

A getter for the pathProfileGDS slot in a RAIDSparam class

Description

A function for getting the pathProfileGDS slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
pathProfileGDS(x)
```

Arguments

x	a RAIDSparam object.

Value

NULL or a single character string representing the path to the directory where the GDS Profile files will be created.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pathProfileGDS slot for the object
pathProfileGDS(paramDemo)
```

pathProfileGDS<- *Generic function for replacement of pathProfileGDS slot in a class*

Description

A generic function for replacement of pathProfileGDS slot in a S4 object.

Usage

```
pathProfileGDS(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pathProfileGDS="character"))

# Create an instance
obj <- new("MyClass", pathProfileGDS="123")

# Call the generic (assuming a method is implemented)
# pathProfileGDS(obj) <- "333"
```

pathProfileGDS<-,RAIDSparam-method *Setter function for replacement of pathProfileGDS slot in a RAIDSparam class*

Description

A function for replacement of pathProfileGDS slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
pathProfileGDS(x) <- value
```

Arguments

x	a RAIDSparam object.
value	NULL or a single character string representing the path to the directory where the GDS Profile files will be created. If specified, the directory must exist.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Directory where demo GDS files are located
dataDir <- system.file("extdata", package="RAIDS")

## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the existing path to the pathProfileGDS slot in the object
pathProfileGDS(paramDemo) <- file.path(dataDir)
```

PCAalgorithm *Generic function for getting the PCAalgorithm slot in a class*

Description

A generic function for getting the PCAalgorithm slot in a S4 object.

Usage

```
PCAalgorithm(x)
```

Arguments

x	a S4 object.

Value

a value from the PCAalgorithm slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(PCAalgorithm="character"))

# Create an instance
obj <- new("MyClass", PCAalgorithm="123")

# Call the generic (assuming a method is implemented)
# PCAalgorithm(obj)
```

PCAalgorithm,RAIDSparam-method

A getter for the PCAalgorithm slot in a RAIDSparam class

Description

A function for getting the PCAalgorithm slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
PCAalgorithm(x)
```

Arguments

x	a RAIDSparam object.

Value

a character string representing the algorithm to use with the snpgdsPCA function.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the PCAalgorithm slot for the object
PCAalgorithm(paramDemo)
```

PCAalgorithm<- *Generic function for replacement of PCAalgorithm slot in a class*

Description

A generic function for replacement of PCAalgorithm slot in a S4 object.

Usage

```
PCAalgorithm(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(PCAalgorithm="character"))

# Create an instance
obj <- new("MyClass", PCAalgorithm="123")

# Call the generic (assuming a method is implemented)
# PCAalgorithm(obj) <- "333"
```

PCAalgorithm<-,RAIDSparam-method *Setter function for replacement of PCAalgorithm slot in a RAIDSparam class*

Description

A function for replacement of PCAalgorithm slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
PCAalgorithm(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a character string representing the algorithm to use with the snpgdsPCA function. The algorithm must be implemented and available to the snpgdsPCA function.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the PCAalgorithm slot in the object
PCAalgorithm(paramDemo) <-  "randomized"
```

---

pcaList *Generic function for getting the pcaList slot in a class*

Description

A generic function for getting the pcaList slot in a S4 object.

Usage

```
pcaList(x)
```

Arguments

x	a S4 object.

Value

a value from the pcaList slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pcaList="character"))

# Create an instance
obj <- new("MyClass", pcaList="123")

# Call the generic (assuming a method is implemented)
# pcaList(obj)
```

---

pcaList,RAIDSparam-method

A getter for the pcaList slot in a RAIDSparam class

Description

A function for getting the pcaList slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
pcaList(x)
```

Arguments

x	a RAIDSparam object.

Value

a vector of positive integer representing the values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pcaList slot for the object
pcaList(paramDemo)
```

pcaList<- *Generic function for replacement of pcaList slot in a class*

Description

A generic function for replacement of pcaList slot in a S4 object.

Usage

```
pcaList(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pcaList="character"))

# Create an instance
obj <- new("MyClass", pcaList="123")

# Call the generic (assuming a method is implemented)
# pcaList(obj) <- "333"
```

pcaList<-,RAIDSparam-method
Setter function for replacement of pcaList slot in a RAIDSparam class

Description

A function for replacement of pcaList slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
pcaList(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a vector of positive integer representing the values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the pcaList slot in the object
pcaList(paramDemo) <-  c(2L, 3L, 4L, 5L)
```

PCAmissingRate *Generic function for getting the PCAmissingRate slot in a class*

Description

A generic function for getting the PCAmissingRate slot in a S4 object.

Usage

```
PCAmissingRate(x)
```

Arguments

x	a S4 object.

Value

a value from the PCAmissingRate slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(PCAmissingRate="character"))

# Create an instance
obj <- new("MyClass", PCAmissingRate="123")

# Call the generic (assuming a method is implemented)
# PCAmissingRate(obj)
```

PCAmissingRate,RAIDSparam-method

A getter for the PCAmissingRate slot in a RAIDSparam class

Description

A function for getting the PCAmissingRate slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
PCAmissingRate(x)
```

Arguments

x	a RAIDSparam object.

Value

a positive numeric representing the maximum missing rate accepted to use SNPs in the PCA analysis done with the snpgdsPCA function. If NaN, no missing threshold.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the PCAmissingRate slot for the object
PCAmissingRate(paramDemo)
```

PCAmissingRate<- *Generic function for replacement of PCAmissingRate slot in a class*

Description

A generic function for replacement of PCAmissingRate slot in a S4 object.

Usage

```
PCAmissingRate(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(PCAmissingRate="character"))

# Create an instance
obj <- new("MyClass", PCAmissingRate="123")

# Call the generic (assuming a method is implemented)
# PCAmissingRate(obj) <- "333"
```

PCAmissingRate<-,RAIDSparam-method *Setter function for replacement of PCAmissingRate slot in a RAIDSparam class*

Description

A function for replacement of PCAmissingRate slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
PCAmissingRate(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a positive numeric representing the maximum missing rate accepted to use SNPs in the PCA analysis done with the snpgdsPCA function. If NaN, no missing threshold.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the PCAmissingRate slot in the object
PCAmissingRate(paramDemo) <-  0.03322
```

---

pedStudy *Generic function for getting the pedStudy slot in a class*

Description

A generic function for getting the pedStudy slot in a S4 object.

Usage

```
pedStudy(x)
```

Arguments

x	a S4 object.

Value

a value from the pedStudy slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(studyDF="character"))

# Create an instance
obj <- new("MyClass", studyDF="123")

# Call the generic (assuming a method is implemented)
# studyDF(obj)
```

---

pedStudy,RAIDSparam-method

A getter for the pedStudy slot in a RAIDSparam class

Description

A function for getting the pedStudy slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
pedStudy(x)
```

Arguments

x	a RAIDSparam object.

Value

a data.frame.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pedStudy slot for the object
pedStudy(paramDemo)
```

pedStudy<- *Generic function for replacement of pedStudy slot in a class*

Description

A generic function for replacement of pedStudy slot in a S4 object.

Usage

```
pedStudy(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pedStudy="character"))

# Create an instance
obj <- new("MyClass", pedStudy="123")

# Call the generic (assuming a method is implemented)
# pedStudy(obj) <- "333"
```

pedStudy<-,RAIDSparam-method *Setter function for replacement of pedStudy slot in a RAIDSparam class*

Description

A function for replacement of pedStudy slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
pedStudy(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a data.frame containing the information TODO with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", and "Source". All columns must be character strings (no factors). All row names should correspond to the values in the "Name.ID" column.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Demo PED study
pedStudy <- data.frame(Name.ID=c("TEST11", "Test12"),
    Case.ID=c("Case1", "Case2"),
    Sample.Type=c("Cancer", "Cancer"),
    Diagnosis=c("sarcoma", "sarcoma"),
    Source=c("CSHL", "CSHL"), stringsAsFactors=FALSE,
    row.names=c("TEST11", "Test12"))

## Assign the new PED study to the pedStudy slot in the object
pedStudy(paramDemo) <- pedStudy
```

---

pedSynthetic — *A small* data.frame *containing the information related to synthetic profiles. The ancestry of the profiles used to generate the synthetic profiles must be present.*

Description

The object is a data.frame with 7 columns. The row names of the data.frame must be the profile unique identifiers.

Usage

```
data(pedSynthetic)
```

Format

The `data.frame` containing the information about the synthetic profiles. The row names of the `data.frame` correspond to the profile unique identifiers. The `data.frame` contains 7 columns:

`data.id` a `character` string representing the unique synthetic profile identifier.

`case.id` a `character` string representing the unique profile identifier that was used to generate the synthetic profile.

`sample.type` a `character` string representing the type of profile.

`diagnosis` a `character` string representing the diagnosis of profile that was used to generate the synthetic profile.

`source` a `character` string representing the source of the synthetic profile.

`study.id` a `character` string representing the name of the study to which the synthetic profile is associated.

`superPop` a `character` string representing the super population of the profile that was used to generate the synthetic profile.

Details

This dataset can be used to test the `computeSyntheticROC` function.

Value

The `data.frame` containing the information about the synthetic profiles. The row names of the `data.frame` correspond to the profile unique identifiers. The `data.frame` contains 7 columns:

`data.id` a `character` string representing the unique synthetic profile identifier.

`case.id` a `character` string representing the unique profile identifier that was used to generate the synthetic profile.

`sample.type` a `character` string representing the type of profile.

`diagnosis` a `character` string representing the diagnosis of profile that was used to generate the synthetic profile.

`source` a `character` string representing the source of the synthetic profile.

`study.id` a `character` string representing the name of the study to which the synthetic profile is associated.

`superPop` a `character` string representing the super population of the profile that was used to generate the synthetic profile.

See Also

`computeSyntheticROC` for calculating the AUROC of the inferences for specific values of D and K using the inferred ancestry results from the synthetic profiles

Examples

```
## Loading demo dataset containing pedigree information for synthetic
## profiles
data(pedSynthetic)

## Loading demo dataset containing the inferred ancestry results
## for the synthetic data
data(matKNNSynthetic)

## Retain one K and one D value
matKNN <- matKNNSynthetic[matKNNSynthetic$D == 5 & matKNNSynthetic$K == 4, ]

## Compile statistics from the
## synthetic profiles for fixed values of D and K
results <- RAIDS:::computeSyntheticROC(matKNN=matKNN,
    matKNNAncestryColumn="SuperPop",
    pedCall=pedSynthetic, pedCallAncestryColumn="superPop",
    listCall=c("EAS", "EUR", "AFR", "AMR", "SAS"))

results$matAUROC.All
results$matAUROC.Call
results$listROC.Call
```

---

phase *Generic function for getting the phase slot in a class*

Description

A generic function for getting the phase slot in a S4 object.

Usage

```
phase(x)
```

Arguments

x	a S4 object.

Value

a value from the phase slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(phase="character"))

# Create an instance
obj <- new("MyClass", phase="123")

# Call the generic (assuming a method is implemented)
# phase(obj)
```

---

phase,RAIDSparam-method

A getter for the phase slot in a RAIDSparam class

Description

A function for getting the phase slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
phase(x)
```

Arguments

x	a RAIDSparam object.

Value

a single logical TODO

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the phase slot for the object
phase(paramDemo)
```

phase<- *Generic function for replacement of phase slot in a class*

Description

A generic function for replacement of phase slot in a S4 object.

Usage

```
phase(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(phase="character"))

# Create an instance
obj <- new("MyClass", phase="123")

# Call the generic (assuming a method is implemented)
# phase(obj) <- "333"
```

phase<-,RAIDSparam-method *Setter function for replacement of phase slot in a RAIDSparam class*

Description

A function for replacement of phase slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
phase(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single logical TODO

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the phase slot in the object
phase(paramDemo) <-  TRUE
```

---

phaseType *Generic function for getting the phaseType slot in a class*

Description

A generic function for getting the phaseType slot in a S4 object.

Usage

```
phaseType(x)
```

Arguments

x	a S4 object.

Value

a value from the phaseType slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(phaseType="character"))

# Create an instance
obj <- new("MyClass", phaseType="123")

# Call the generic (assuming a method is implemented)
# phaseType(obj)
```

---

phaseType,RAIDSparam-method

A getter for the phaseType slot in a RAIDSparam class

Description

A function for getting the phaseType slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
phaseType(x)
```

Arguments

x	a RAIDSparam object.

Value

a single character string TODO

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the phaseType slot for the object
phaseType(paramDemo)
```

phaseType<- *Generic function for replacement of phaseType slot in a class*

Description

A generic function for replacement of phaseType slot in a S4 object.

Usage

```
phaseType(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(phaseType="character"))

# Create an instance
obj <- new("MyClass", phaseType="123")

# Call the generic (assuming a method is implemented)
# phaseType(obj) <- "333"
```

phaseType<-,RAIDSparam-method *Setter function for replacement of phaseType slot in a RAIDSparam class*

Description

A function for replacement of phaseType slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
phaseType(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single character string TODO

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the phaseType slot in the object
phaseType(paramDemo) <- "test"
```

---

posOffset *Generic function for getting the posOffset slot in a class*

Description

A generic function for getting the offposOffsetset slot in a S4 object.

Usage

```
posOffset(x)
```

Arguments

x	a S4 object.

Value

a value from the posOffset slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(posOffset="character"))

# Create an instance
obj <- new("MyClass", posOffset="123")

# Call the generic (assuming a method is implemented)
# posOffset(obj)
```

---

posOffset,RAIDSparam-method

A getter for the posOffset slot in a RAIDSparam class

Description

A function for getting the posOffset slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
posOffset(x)
```

Arguments

x	a RAIDSparam object.

Value

a single integer that is added to the SNP position to switch from 0-based to 1-based coordinate when needed (or reverse).

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the posOffset slot for the object
posOffset(paramDemo)
```

posOffset<- *Generic function for replacement of posOffset slot in a class*

Description

A generic function for replacement of posOffset slot in a S4 object.

Usage

```
posOffset(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(posOffset="character"))

# Create an instance
obj <- new("MyClass", posOffset="123")

# Call the generic (assuming a method is implemented)
# posOffset(obj) <- "333"
```

posOffset<-,RAIDSparam-method *Setter function for replacement of posOffset slot in a RAIDSparam class*

Description

A function for replacement of posOffset slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
posOffset(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single integer that is added to the SNP position to switch from 0-based to 1-based coordinate when needed (or reverse).

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the posOffset slot in the object
posOffset(paramDemo) <- 0L
```

---

pRecomb *Generic function for getting the pRecomb slot in a class*

Description

A generic function for getting the pRecomb slot in a S4 object.

Usage

```
pRecomb(x)
```

Arguments

x	a S4 object.

Value

a value from the pRecomb slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pRecomb="character"))

# Create an instance
obj <- new("MyClass", pRecomb="123")

# Call the generic (assuming a method is implemented)
# pRecomb(obj)
```

---

pRecomb,RAIDSparam-method

A getter for the pRecomb slot in a RAIDSparam class

Description

A function for getting the pRecomb slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
pRecomb(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive numeric between 0 and 1 that represents the frequency of phase switching in the synthetic profiles.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pRecomb slot for the object
pRecomb(paramDemo)
```

pRecomb<- *Generic function for replacement of pRecomb slot in a class*

Description

A generic function for replacement of pRecomb slot in a S4 object.

Usage

```
pRecomb(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pRecomb="character"))

# Create an instance
obj <- new("MyClass", pRecomb="123")

# Call the generic (assuming a method is implemented)
# pRecomb(obj) <- "333"
```

pRecomb<-,RAIDSparam-method *Setter function for replacement of pRecomb slot in a RAIDSparam class*

Description

A function for replacement of pRecomb slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
pRecomb(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single positive numeric between 0 and 1 that represents the frequency of phase switching in the synthetic profiles.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the pRecomb slot in the object
pRecomb(paramDemo) <- 0.03
```

---

prefix *Generic function for getting the prefix slot in a class*

Description

A generic function for getting the prefix slot in a S4 object.

Usage

```
prefix(x)
```

Arguments

x	a S4 object.

Value

a value from the prefix slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(prefix="character"))

# Create an instance
obj <- new("MyClass", prefix="123")

# Call the generic (assuming a method is implemented)
# prefix(obj)
```

`prefix,RAIDSparam-method`

A getter for the prefix slot in a RAIDSparam class

Description

A function for getting the prefix slot in a `RAIDSparam` class.

A function for getting the prefix slot in a `RAIDSparam` class.

A function for getting the prefix slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
prefix(x)

## S4 method for signature 'RAIDSparam'
prefix(x)

## S4 method for signature 'RAIDSparam'
prefix(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

a single `character` string representing TODO.

a single `character` string representing TODO.

a single `character` string representing TODO.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the prefix slot for the object
prefix(paramDemo)


## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the prefix slot for the object
prefix(paramDemo)


## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the prefix slot for the object
prefix(paramDemo)
```

`prefix<-` *Generic function for replacement of prefix slot in a class*

Description

A generic function for replacement of prefix slot in a S4 object.

Usage

```
prefix(x) <- value
```

Arguments

`x`	a S4 object.
`value`	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(prefix="character"))

# Create an instance
obj <- new("MyClass", prefix="123")

# Call the generic (assuming a method is implemented)
# prefix(obj) <- "333"
```

`prefix<-,RAIDSparam-method`

Setter function for replacement of prefix slot in a RAIDSparam class

---

Description

A function for replacement of prefix slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
prefix(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `character` string representing TODO.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the prefix slot in the object
prefix(paramDemo) <- "test"
```

prepPed1KG *Prepare the pedigree file using pedigree information from Reference*

Description

Using the pedigree file from Reference, this function extracts needed information and formats it into a data.frame so in can be used in following steps of the ancestry inference process. The function also requires that the genotyping files associated to each sample be available in a specified directory.

Usage

```
prepPed1KG(filePed, pathGeno = file.path("data", "sampleGeno"), batch = 0L)
```

Arguments

filePed	a character string representing the path and file name of the pedigree file (PED file) that contains the information related to the profiles present in the Reference GDS file. The PED file must exist.
pathGeno	a character string representing the path where the Reference genotyping files for each profile are located. Only the profiles with associated genotyping files are retained in the creation of the final data.frame. The name of the genotyping files must correspond to the individual identification (Individual.ID) in the pedigree file (PED file). Default: "./data/sampleGeno".
batch	ainteger that uniquely identifies the source of the pedigree information. The Reference is usually 0L. Default: 0L.

Value

a data.frame containing the needed pedigree information from Reference. The data.frame contains those columns:

sample.id a character string representing the profile unique ID.

Name.ID a character string representing the profile name.

sex a character string representing the sex of the profile.

pop.group a character string representing the sub-continental ancestry of the profile.

superPop a character string representing the continental ancestry of the profile.

superPop a integer representing the batch of the profile.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Path where the demo genotype CSV files are located
pathGeno <- file.path(dataDir, "demoProfileGenotypes")

## Demo pedigree file
pedDemoFile <- file.path(dataDir, "PedigreeDemo.ped")

## Create a data.frame containing the information of the retained
## samples (samples with existing genotyping files)
prepPed1KG(filePed=pedDemoFile, pathGeno=pathGeno, batch=0L)
```

---

prepPedSyntheticRef *Extract the sample information from the 1KG GDS file for a list of profiles associated to a specific study in the Profile GDS file*

Description

The function extracts the information for the profiles associated to a specific study in the GDS Sample file. The information is extracted from the 'study.annot' node as a 'data.frame'.

Then, the function used the 1KG GDS file to extract specific information about each sample and add it, as an extra column, to the 'data.frame'.

As example, this function can extract the synthetic profiles for a GDS Sample and the super-population of the 1KG samples used to generate each synthetic profile would be added as an extra column to the final 'data.frame'.

Usage

```
prepPedSyntheticRef(pRAIDS)
```

Arguments

pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Details

As example, this function can extract the synthetic profiles for a Profile GDS and the super-population of the 1KG samples used to generate each synthetic profile would be added as an extra column to the final 'data.frame'. In that situation, the 'popName' parameter would correspond to the super-population column and the 'studyID' parameter would be the name given to the synthetic dataset.

Value

data.frame containing the columns extracted from the GDS Sample 'study.annot' node with a extra column named as the 'popName' parameter that has been extracted from the 1KG GDS 'sample.annot' node. Only the rows corresponding to the specified study ('studyID' parameter) are returned.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## The Reference GDS file
path1KG <- system.file("extdata/tests", package="RAIDS")

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=fileReferenceGDS,
                     fieldPopInfAnc="SuperPop")

pedSyn <- RAIDS:::prepPedSyntheticRef(pRAIDS)
head(pedSyn)
```

prepPMatrix *Generate the row for the ADMIXTURE P-matrix*

Description

This function write with P-matrix

Usage

```
prepPMatrix(anchor, snpId, matGr = NULL, pRAIDS)
```

Arguments

anchor	a vector of character representing the ancestral group for each profile from the reference where sample.ref == 1. If the admixture proportion is higher than the threshold for one ancestral group, the value is the name of the ancestral group; else "-".
snpId	a vector of character representing the list of snp.id from gdsReference to keep.
matGr	a matrix containing the sample identifiers and where each column is the name of a subcontinental population. The number of row corresponds to the number of samples for each subcontinental population.
pRAIDS	a RAIDSparam an object with all the RAIDS parameters.

Value

a list containing one entry per ancestral group:

ancestral group a list contenains 2 entries:

- f a list contenains number of rows in matGr + 1 entries:
  - vector the first is frequency for all the anchor of each snpId
  - vector the other is frequency for all the anchor minus one row of matGr of each snpId
- index a vector contenains 2 entriescontenains number of rows in matGr entries. Each value is the index of the row for which the f of this index corresponds.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(SNPRelate)
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")
```

```
# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                      pedStudy=pedStudy,
                      pathProfileGDS=pathProfileGDS,
                      fileReferenceGDS=fileReferenceGDS,
                      fieldPopInfAnc="SuperPop")


matGr <- getMatrixPopSynthetic(pRAIDS=pRAIDS)
fileGDSProfile <- file.path(pathProfileGDS(pRAIDS),
                            paste0(pedStudy(pRAIDS)$Name.ID[1], ".gds"))


gdsProfile <- snpgdsOpen(fileGDSProfile)
pruned <- read.gdsn(index.gdsn(gdsProfile, "pruned.study"))
snpgdsClose(gdsProfile)
gdsReference <- snpgdsOpen(fileReferenceGDS(pRAIDS))

admixDesc <- read.gdsn(index.gdsn(gdsReference, "admixture.desc"))
pos <- admixDesc[admixDesc$id == anchorId(pRAIDS), "id.seq"]

anchor <- read.gdsn(index.gdsn(gdsReference, "anchor.ref"),
                    start=c(1, pos), count=c(-1,1))
snpgdsClose(gdsReference)

rowF <- prepPMatrix( anchor=anchor, snpId = pruned, matGr=matGr, pRAIDS=pRAIDS)
names(rowF)
```

`prepSynthetic` *Add information related to the synthetic profiles (study and synthetic reference profiles information) into a Profile GDS file*

Description

This function add entries related to synthetic profiles into a Profile GDS file. The entries are related to two types of information: the synthetic study and the synthetic profiles.

The study information is appended to the Profile GDS file "study.list" node. The "study.platform" entry is always set to 'Synthetic'.

The profile information, for all selected synthetic profiles, is appended to the Profile GDS file "study.annot" node. Both the "Source" and the "Sample.Type" entries are always set to 'Synthetic'.

The synthetic profiles are assigned unique names by combining: `prefix.data.id.profile.listSampleRef.simulation number(1 to nbSim)`

Usage

```
prepSynthetic(
  fileProfileGDS,
  listSampleRef,
  profileID,
  studyDF,
  nbSim = 1L,
  prefix = "",
  verbose = FALSE
)
```

Arguments

`fileProfileGDS`	a `character` string representing the file name of the Profile GDS file containing the information about the reference profiles used to generate the synthetic profiles.
`listSampleRef`	a `vector` of `character` string representing the identifiers of the selected 1KG profiles that will be used as reference to generate the synthetic profiles.
`profileID`	a `character` string representing the profile identifier present in the `fileProfileGDS` that will be used to generate synthetic profiles.
`studyDF`	a `data.frame` containing the information about the study associated to the analysed sample(s). The `data.frame` must have those 2 columns: "study.id" and "study.desc". Those 2 columns must be in `character` strings (no factor). Other columns can be present, such as "study.platform", but won't be used.
`nbSim`	a single positive `integer` representing the number of simulations per combination of sample and 1KG reference. Default: `1L`.
`prefix`	a single `character` string representing the prefix that is going to be added to the name of the synthetic profile. The prefix enables the creation of multiple synthetic profile using the same combination of sample and 1KG reference. Default: `""`.
`verbose`	a `logical` indicating if messages should be printed to show how the different steps in the function. Default: `FALSE`.

Value

`0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")

## Temporary Profile GDS file
fileNameGDS <- file.path(tempdir(), "ex1.gds")

## Copy the Profile GDS file demo that has been pruned and annotated
file.copy(file.path(dataDir, "ex1_demo_with_pruning_and_1KG_annot.gds"),
                 fileNameGDS)

## Information about the synthetic data set
syntheticStudyDF <- data.frame(study.id="MYDATA.Synthetic",
        study.desc="MYDATA synthetic data", study.platform="PLATFORM",
        stringsAsFactors=FALSE)

## Add information related to the synthetic profiles into the Profile GDS
prepSynthetic(fileProfileGDS=fileNameGDS,
        listSampleRef=c("HG00243", "HG00150"), profileID="ex1",
        studyDF=syntheticStudyDF, nbSim=1L, prefix="synthetic",
        verbose=FALSE)

## Open Profile GDS file
profileGDS <- openfn.gds(fileNameGDS)

## The synthetic profiles should be added in the 'study.annot' entry
tail(read.gdsn(index.gdsn(profileGDS, "study.annot")))

## The synthetic study information should be added to
## the 'study.list' entry
tail(read.gdsn(index.gdsn(profileGDS, "study.list")))

## Close GDS file (important)
closefn.gds(profileGDS)

## Remove Profile GDS file (created for demo purpose)
unlink(fileNameGDS, force=TRUE)
```

`profileFile` *Generic function for getting the profileFile slot in a class*

Description

A generic function for getting the profileFile slot in a S4 object.

Usage

```
profileFile(x)
```

Arguments

x	a S4 object.

Value

a value from the profileFile slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(profileFile="character"))

# Create an instance
obj <- new("MyClass", profileFile="123")

# Call the generic (assuming a method is implemented)
# profileFile(obj)
```

---

profileFile,RAIDSparam-method

A getter for the profileFile slot in a RAIDSparam class

Description

A function for getting the profileFile slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
profileFile(x)
```

Arguments

x	a `RAIDSparam` object.

Value

NULL or a character string representing the path to the file with genotype and the allele information of the profile.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the profileFile slot for the object
profileFile(paramDemo)
```

---

profileFile<- *Generic function for replacement of profileFile slot in a class*

Description

A generic function for replacement of profileFile slot in a S4 object.

Usage

```
profileFile(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(profileFile="character"))

# Create an instance
obj <- new("MyClass", profileFile="123")

# Call the generic (assuming a method is implemented)
# profileFile(obj) <- "333"
```

`profileFile<-,RAIDSparam-method`

Setter function for replacement of profileFile slot in a RAIDSparam class

---

Description

A function for replacement of profileFile slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
profileFile(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	`NULL` or a `character` string representing the path to the file with genotype and the allele information of the profile. The format of the accepted file is determined by the value in the 'genoSource' slot. A profile would have a file with extension "vcf.gz" when 'genoSource' slot is "VCF". If 'genoSource' is "generic" or "snp-pileup", then ".txt.gz". If 'genoSource' is "bam", then ".bam" (the file needs to be indexed with an existing corresponding ".bai" file).

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Directory where demo GDS files are located
dataDir <- system.file("extdata", package="RAIDS")

## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Adjust the genoSource slot to the updated format file
genoSource(paramDemo) <- "generic"

## Assign the existing profile file to the profileFile slot in the object
profileFile(paramDemo) <- file.path(dataDir, "tests/ex1.txt.gz")
```

`profileGenoSynthesis` *Create the gds file for the profile with the genotype and synthetic data*

Description

First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population Reference GDS file.

Usage

```
profileGenoSynthesis(pRAIDS)
```

Arguments

`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters

Value

The integer `0L` when the function is successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")


#################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
```

```
#####################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
demoProfileEx1 <- file.path(dataDir, "example", "snpPileup", "ex1.txt.gz")

#####################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathProfileGDS)){
    dir.create(pathProfileGDS)
}

#####################################################################
## Fix seed to ensure reproducible results
#####################################################################
set.seed(3043)

pRAIDS <- RAIDSparam(profileFile=demoProfileEx1,
    genoSource="snp-pileup",
    pathProfileGDS=pathProfileGDS,
    fileReferenceGDS=fileReferenceGDS,
    fileReferenceAnnotGDS=fileAnnotGDS
    )



pRAIDS <- profileGenoSynthesis(pRAIDS=pRAIDS)

unlink(file.path(pathProfileGDS, paste0(pedStudy(pRAIDS)$Name.ID[1], ".gds")) , force=TRUE)
if (length(dir(pathProfileGDS)) == 0) {
unlink(pathProfileGDS, recursive = TRUE)
}
```

---

pruningMethod *Generic function for getting the pruningMethod slot in a class*

Description

A generic function for getting the pruningMethod slot in a S4 object.

Usage

```
pruningMethod(x)
```

Arguments

x	a S4 object.

Value

a value from the pruningMethod slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pruningMethod="character"))

# Create an instance
obj <- new("MyClass", pruningMethod="123")

# Call the generic (assuming a method is implemented)
# pruningMethod(obj)
```

---

pruningMethod,RAIDSparam-method

A getter for the pruningMethod slot in a RAIDSparam class

Description

A function for getting the pruningMethod slot in a RAIDSparam class.

A function for getting the pruningMethod slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
pruningMethod(x)

## S4 method for signature 'RAIDSparam'
pruningMethod(x)
```

Arguments

x	a RAIDSparam object.

Value

a single character string representing the method that will be used to calculate the linkage disequilibrium in the snpgdsLDpruning() function.

a single character string representing the method that will be used to calculate the linkage disequilibrium in the snpgdsLDpruning() function.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pruningMethod slot for the object
pruningMethod(paramDemo)


## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the pruningMethod slot for the object
pruningMethod(paramDemo)
```

---

pruningMethod<- *Generic function for replacement of pruningMethod slot in a class*

Description

A generic function for replacement of pruningMethod slot in a S4 object.

Usage

```
pruningMethod(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(pruningMethod="character"))

# Create an instance
obj <- new("MyClass", pruningMethod="123")

# Call the generic (assuming a method is implemented)
# pruningMethod(obj) <- "333"
```

`pruningMethod<-,RAIDSparam-method`

Setter function for replacement of pruningMethod slot in a RAIDSparam class

---

Description

A function for replacement of pruningMethod slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
pruningMethod(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `character` string representing the method that will be used to calculate the linkage disequilibrium in the `snpgdsLDpruning`() function. The 4 possible values are: "corr", "r", "dprime", and "composite".

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the pruningMethod slot in the object
pruningMethod(paramDemo) <- "r"
```

pruningProfile2 *Compute the list of pruned SNVs for a specific profile using the information from the Reference GDS file and a linkage disequilibrium analysis*

Description

This function computes the list of pruned SNVs for a specific profile. When a group of SNVs are in linkage disequilibrium, only one SNV from that group is retained. The linkage disequilibrium is calculated with the snpgdsLDpruning() function. The initial list of SNVs that are passed to the snpgdsLDpruning() function can be specified by the user.

Usage

```
pruningProfile2(pRAIDS)
```

Arguments

pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Value

The function returns a RAIDSparam an object with all the RAIDS parameters updated.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo Reference GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

## The data.frame containing the information about the study
## The 3 mandatory columns: "study.id", "study.desc", "study.platform"
## The entries should be strings, not factors (stringsAsFactors=FALSE)
studyDF <- data.frame(study.id = "MYDATA",
                      study.desc = "Description",
                      study.platform = "PLATFORM",
                      stringsAsFactors = FALSE)

## The data.frame containing the information about the samples
## The entries should be strings, not factors (stringsAsFactors=FALSE)
samplePED <- data.frame(Name.ID = c("ex1", "ex2"),
                    Case.ID = c("Patient_h11", "Patient_h12"),
                    Diagnosis = rep("Cancer", 2),
                    Sample.Type = rep("Primary Tumor", 2),
                    Source = rep("Databank B", 2), stringsAsFactors = FALSE)
```

```
rownames(samplePED) <- samplePED$Name.ID

## Temporary Profile GDS file
pathProfileGDS <- tempdir()
profileFile <- file.path(pathProfileGDS, "ex1.gds")

pRAIDS <- RAIDSparam( studyDF=studyDF,
                      profileFile=file.path(dataDir, "ex1.txt.gz"),
                      genoSource = "snp-pileup",
                      pedStudy=samplePED,
                      pathProfileGDS=pathProfileGDS,
                      fileReferenceGDS=fileGDS)

## Copy the Profile GDS file demo that has not been pruned yet
file.copy(file.path(dataDir, "ex1_demo.gds"), profileFile)


## Compute the list of pruned SNVs for a specific profile 'ex1'
## and save it in the Profile GDS file 'ex1.gds'
pRAIDS <- pruningProfile2(pRAIDS)


## Check content of Profile GDS file
## The 'pruned.study' entry should be present
content <- openfn.gds(profileFile)
content

## Close the Profile GDS file (important)
closefn.gds(content)

## Remove Profile GDS file (created for demo purpose)
unlink(profileFile, force=TRUE)
```

`pruningSample` *Compute the list of pruned SNVs for a specific profile using the information from the Reference GDS file and a linkage disequilibrium analysis*

Description

This function computes the list of pruned SNVs for a specific profile. When a group of SNVs are in linkage disequilibrium, only one SNV from that group is retained. The linkage disequilibrium is calculated with the `snpgdsLDpruning`() function. The initial list of SNVs that are passed to the `snpgdsLDpruning`() function can be specified by the user.

Usage

```
pruningSample(
  gdsReference,
```

```
  method = c("corr", "r", "dprime", "composite"),
  currentProfile,
  studyID,
  listSNP = NULL,
  slideWindowMaxBP = 500000L,
  thresholdLD = sqrt(0.1),
  np = 1L,
  verbose = FALSE,
  chr = NULL,
  superPopMinAF = NULL,
  keepPrunedGDS = TRUE,
  pathProfileGDS = NULL,
  keepFile = FALSE,
  pathPrunedGDS = ".",
  outPrefix = "pruned"
)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the 1 KG GDS file (reference data set).
method	a character string that represents the method that will be used to calculate the linkage disequilibrium in the snpgdsLDpruning() function. The 4 possible values are: "corr", "r", "dprime" and "composite". Default: "corr".
currentProfile	a character string corresponding to the profile identifier used in LD pruning done by the snpgdsLDpruning() function. A Profile GDS file corresponding to the profile identifier must exist and be located in the pathProfileGDS directory.
studyID	a character string corresponding to the study identifier used in the snpgdsLDpruning function. The study identifier must be present in the Profile GDS file.
listSNP	a vector of SNVs identifiers specifying selected to be passed the the pruning function; if NULL, all SNVs are used in the snpgdsLDpruning function. Default: NULL.
slideWindowMaxBP	a single positive integer that represents the maximum basepairs (bp) in the sliding window. This parameter is used for the LD pruning done in the snpgdsLDpruning function. Default: 500000L.
thresholdLD	a single numeric value that represents the LD threshold used in the snpgdsLDpruning function. Default: sqrt(0.1).
np	a single positive integer specifying the number of threads to be used. Default: 1L.
verbose	a logicial indicating if information is shown during the process in the snpgdsLDpruning function. Default: FALSE.
chr	a character string representing the chromosome where the selected SNVs should belong. Only one chromosome can be handled. If NULL, the chromosome is not used as a filtering criterion. Default: NULL.
superPopMinAF	a single positive numeric representing the minimum allelic frequency used to select the SNVs. If NULL, the allelic frequency is not used as a filtering criterion. Default: NULL.

keepPrunedGDS a logicial indicating if the information about the pruned SNVs should be added to the GDS Sample file. Default: TRUE.

pathProfileGDS a character string representing the directory where the Profile GDS files will be created. The directory must exist.

keepFile a logical indicating if RDS files containing the information about the pruned SNVs must be created. Default: FALSE.

pathPrunedGDS a character string representing an existing directory. The directory must exist. Default: ".".

outPrefix a character string that represents the prefix of the RDS files that will be generated. The RDS files are only generated when the parameter keepFile=TRUE. Default: "pruned".

Value

The function returns 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library for GDS
library(gdsfmt)

## Path to the demo Reference GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")
fileGDS <- file.path(dataDir, "ex1_good_small_1KG.gds")

## The data.frame containing the information about the study
## The 3 mandatory columns: "study.id", "study.desc", "study.platform"
## The entries should be strings, not factors (stringsAsFactors=FALSE)
studyDF <- data.frame(study.id = "MYDATA",
                      study.desc = "Description",
                      study.platform = "PLATFORM",
                      stringsAsFactors = FALSE)

## The data.frame containing the information about the samples
## The entries should be strings, not factors (stringsAsFactors=FALSE)
samplePED <- data.frame(Name.ID = c("ex1", "ex2"),
                    Case.ID = c("Patient_h11", "Patient_h12"),
                    Diagnosis = rep("Cancer", 2),
                    Sample.Type = rep("Primary Tumor", 2),
                    Source = rep("Databank B", 2), stringsAsFactors = FALSE)
rownames(samplePED) <- samplePED$Name.ID

## Temporary Profile GDS file
profileFile <- file.path(tempdir(), "ex1.gds")

## Copy the Profile GDS file demo that has not been pruned yet
```

```
file.copy(file.path(dataDir, "ex1_demo.gds"), profileFile)

## Open 1KG file
gds1KG <- snpgdsOpen(fileGDS)

## Compute the list of pruned SNVs for a specific profile 'ex1'
## and save it in the Profile GDS file 'ex1.gds'
pruningSample(gdsReference=gds1KG, currentProfile=c("ex1"),
              studyID = studyDF$study.id, pathProfileGDS=tempdir())

## Close the Reference GDS file (important)
closefn.gds(gds1KG)

## Check content of Profile GDS file
## The 'pruned.study' entry should be present
content <- openfn.gds(profileFile)
content

## Close the Profile GDS file (important)
closefn.gds(content)

## Remove Profile GDS file (created for demo purpose)
unlink(profileFile, force=TRUE)
```

`RAIDSparam` *Create a RAIDSparam object*

Description

The function uses all parameters to create a `RAIDSparam` class object that contains all the parameters used in the RAIDS workflow.

Usage

```
RAIDSparam(
  studyDF = NULL,
  studyDFSyn = NULL,
  pedStudy = NULL,
  studyType = c("LD", "GeneAware"),
  genoSource = NULL,
  blockTypeId = "GeneS.Ensembl.Hsapiens.v86",
  reference = "1KGv1.0",
  genome = "HG38",
  chrInfo = NULL,
  paramAncestry = NULL,
  profileFile = NULL,
  profileFileGeno = NULL,
```

```
  pathProfileGDS = NULL,
  fileReferenceGDS = NULL,
  fileReferenceAnnotGDS = NULL,
  inferenceType = "PCAknn",
  sampleRef = NULL,
  batch = 1L,
  prefix = "1",
  nbSim = 1L,
  posOffset = -1L,
  minCov = 10L,
  minProb = 0.999,
  seqError = 0.001,
  seqErrorSyn = 0.001,
  pRecomb = 0.01,
  np = 1L,
  listPos = NULL,
  syntheticRefDF = NULL,
  pruningMethod = c("corr", "r", "dprime", "composite"),
  slideWindowMaxBP = 500000L,
  thresholdLD = sqrt(0.1),
  specificSNV = NULL,
  genoType = "geno.ref",
  phaseType = "phase.ref",
  phase = FALSE,
  PCAmissingRate = 0.025,
  PCAalgorithm = "exact",
  eigenCount = 32L,
  eigenCountSyn = 15L,
  kList = seq(2L, 15L, 1L),
  pcaList = seq(2L, 15L, 1L),
  fieldPopInRef = "superPop",
  fieldPopInfAnc = "superPop",
  fieldSubPop = "pop.group",
  anchorId = "sp0.95v0.2",
  pathADMIXTURE = NULL,
  seedADMIXTURE = 432L,
  cleanADMIXTUREFiles = TRUE,
  verbose = FALSE
)
```

Arguments

studyDF	a `data.frame` containing the information about the study associated to the analysed sample(s). The `data.frame` must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be `character` strings (no factors). If `NULL`, the following will be assigned: `data.frame(study.id="NotDef", study.desc="NotDef",study.plat stringsAsFactors=FALSE)`. Default: `NULL`.
studyDFSyn	a `data.frame` containing the information about the synthetic data to the anal-

ysed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be character strings (no factors). If NULL, the following will be assigned: data.frame(study.id="NotDef.Synthetic",study.desc="NotDef synthetic data", study.platform="Synthetic",stringsAsFactors=FALSE). Default: NULL.

pedStudy — a data.frame containing the information TODO with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", and "Source". All columns must be in character strings (no factors). All row names should correspond to the values in the "Name.ID" column. If NULL, the following will be assigned: data.frame(Name.ID=c("ProfileId"),Case.ID=c("ProfileId"), Sample.Type=c("type"), Diagnosis="NotDef",Source=c("NotDef"), stringsAsFactors=FALSE, row.names = c("ProfileId")). Default: NULL.

studyType — a single character string representing the type of study. The possible choices are: "LD" and "GeneAware". The study type affects how the allelic fraction is estimated. Default: "LD".

genoSource — a single character string corresponding to the type of file with the genotype and the allele information of the profile that will be provided in the 'profileFile' slot. The valid options are: "VCF", "generic", "snp-pileup", and "bam". The "generic" format CSV file must have at least these columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R', and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele, and 'File1A' is the depth of the specific alternative allele. Finally, in the case of a "VCF" file, the file must have at least those genotype fields: GT, AD, and DP. Default: NULL.

blockTypeId — a single character string corresponding to the block type used to extract the block identifiers. The block type must be present in the GDS Reference Annotation file. Default: "GeneS.Ensembl.Hsapiens.v86".

reference — a character string with two possible values: '1KGv1.0', '1k_hgdpV0.1'. It specifies the type of inference. Default: "1KGv1.0".

genome — a character string with one possible value: 'HG38'. It specifies the genome used. Default: "HG38".

chrInfo — a vector of positive integer values representing the length of the chromosomes. See 'details' section. If NULL, the following value is assigned: seqlengths(Hsapiens)[seq_len(Default: NULL.

paramAncestry — a list of parameters related to ancestry. The list should contain those three entries: ScanBamParam, PileupParam, and yieldSize. If NULL, the default value list(ScanBamParam=NULL, PileupParam=NULL, yieldSize=10000000) will be assigned to the parameter. Default: NULL.

profileFile — a character string representing the path to the file with genotype and the allele information of the profile. The format of the accepted file is determined by the value in the 'genoSource' slot. A profile would have a file with extension "*vcf.gz" when 'genoSource' slot is "VCF". If 'genoSource' is "generic" or "snp-pileup", then "*.txt.gz". If 'genoSource' is "bam", then "*.bam" (the file need to be indexed with a existing corresponding "*.bai" file). Default: NULL.

profileFileGeno — TODO

pathProfileGDS a single character string representing the path to the directory where the GDS Profile files will be created. If specified, the directory must exist. Default: NULL.

fileReferenceGDS
: a single character string representing the file name of the Reference GDS file. If specified, the file must exist. Default: NULL.

fileReferenceAnnotGDS
: a character string representing the file name of the Population Reference GDS Annotation file. If specified, the file must exist. Default: NULL.

inferenceType a single character string representing the genotype ancestry inference method. The two possible values: 'PCAknn' and 'haploAdmixture'. Default: "PCAknn".

sampleRef a vector of character strings that represent a subset of sample identifiers from the GDS Reference File. Those samples will be used for the ancestry inference. If NULL, all the samples are retained. This is usefull if you want to removed some specific samples (ex: samples with admixed ancestry). Default: NULL.

batch a single positive integer representing the current identifier for the batch. Beware, this field is not stored anymore. Default: 1L.

prefix a single character string representing TODO. Default: "1".

nbSim a positive integer representing the number of replicated simulation (Only 1 is supported for the moment). Default: 1L.

posOffset a single integer that is added to the SNP position to switch from 0-based to 1-based coordinate when needed (or reverse). Default: -1L.

minCov a single positive integer representing the minimum required coverage. Default: 10L.

minProb a single numeric between 0 and 1 representing the probability that the calculated genotype call is correct. Default: 0.999.

seqError a single numeric between 0 and 1 representing the probability of sequencing error. Default: 0.001.

seqErrorSyn a single numeric between 0 and 1 representing the probability of sequencing error for synthetic. Default: 0.001.

pRecomb a single positive numeric between 0 and 1 that represents the frequency of phase switching in the synthetic profiles. Default: 0.01.

np a single positive integer specifying the number of threads to be used. Default: 1L.

listPos a data.frame containing 2 columns named: "snp.chromosome" and "snp.position". The first column, called "snp.chromosome", contains the name of the chromosome where the SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome. Default: NULL.

syntheticRefDF a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns:

sample.id a character string representing the sample identifier.

pop.group a character string representing the subcontinental population assigned to the sample.

superPop a character string representing the super-population assigned to the sample.

Default: NULL.

pruningMethod a character string representing the method that will be used to calculate the linkage disequilibrium in the snpgdsLDpruning() function. The 4 possible values are: "corr", "r", "dprime", and "composite". Default: "corr".

slideWindowMaxBP
a single positive integer that represents the maximum basepairs (bp) in the sliding window. This parameter is used for the LD pruning done by the snpgdsLDpruning function. Default: 500000L.

thresholdLD a single positive numeric value that represents the LD threshold used in the snpgdsLDpruning function. Default: sqrt(0.1).

specificSNV a data.frame containing 2 columns. The first column, called "snp.chromosome" contains the name of the chromosome where the SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome. Optionally, the column "snvKeep" contains the SNV index in the reference or -1 if not in the reference. It is used during the pruning step. Default: NULL.

genoType a single character string representing the field in the GDS profile file that contains all the genotype information. Default: "geno.ref".

phaseType a TODO. Default: "phase.ref".

phase a logical indicating TODO. Default: FALSE.

PCAmissingRate a positive numeric representing the maximum missing rate retained accepted to use SNPs in the PCA analysis done with the snpgdsPCA function. If NaN, no missing threshold. Default: 0.025.

PCAalgorithm a character string representing the algorithm to use with the snpgdsPCA function. The algorithm must be implemented and available to the snpgdsPCA function. Default: "exact".

eigenCount a single integer indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned. Default: 32L.

eigenCountSyn a TODO. Default: 15L.

kList a vector of positive integer representing the values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis. Default: seq(2L, 15L, 1L).

pcaList a vector of positive integer representing the values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis. Default: seq(2L, 15L, 1L).

fieldPopInRef a character string representing the name of the column that contains the known ancestry for the reference profiles in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter). Default: "superPop".

fieldPopInfAnc a character string representing the name of the column in the data frame that contains the inferred super-population ancestry for the samples. The column should be present in the data frame. Default: "superPop".

`fieldSubPop` a `character` string representing the name of the column in the Population Reference GDS file (corresponding to the `fileReferenceGDS` parameter) that contains the sub-population information for the samples. The column should be present in the Population Reference GDS file. Default: `"pop.group"`.

`anchorId` a `character` string representing the name of the identifier of the quasi non-admixed profile from the reference. Default: `"sp0.95v0.2"`

`pathADMIXTURE` a `character` string representing the path pf the software ADMIXTURE and the binary. If specified, the file must exist. Default: `NULL`.

`seedADMIXTURE` a single `integer` seed for the ADMIXTURE program. Default: `432L`.

`cleanADMIXTUREFiles` a single `logical` indicating if RAIDS removes ADMIXTURE files after processing. Default: `TRUE`.

`verbose` a `logical` indicating if message information should be printed. Default: `FALSE`

Details

The `chrInfo` parameter contains the length of the chromosomes. The length of the chromosomes can be obtain through the `seqlengths` library.

As example, for hg38 genome:

```
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]
}
```

Value

an object of class `RAIDSparam` that contains all the required parameters needed by the RAIDS workflow.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## New object of class "RAIDSparam" with default parameters
newParam1 <- RAIDSparam()

## New object of class "RAIDSparam" with non-default parameters
newParam2 <- RAIDSparam(studyDF=data.frame(study.id="Test1",
    study.desc="Test Desc", study.platform="Test", stringsAsFactors=FALSE),
    nbSim=1L, minCov=8L)
```

RAIDSparam-class *An S4 class to represent the RAIDS parameters*

Description

An S4 class to represent the RAIDS parameters

Details

The chrInfo slot contains the length of the chromosomes. The length of the chromosomes can be obtain through the seqlengths library.

As example, for hg38 genome:

```
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]
}
```

Slots

studyDF a data.frame containing the information about the study associated to the analysed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be in character strings (no factors). Default: data.frame(study.id="NotDef", study.desc="NotDef",study.platform="NotDef", stringsAsFactors=FALSE).

studyDFSyn a data.frame containing the information about the synthetic data to the analysed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be in character strings (no factors). Default: data.frame(study.id="NotDef.Synthetic",study synthetic data", study.platform="Synthetic",stringsAsFactors=FALSE).

pedStudy a data.frame containing the information TODO with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", and "Source". All columns must be in character strings (no factors). All row names should correspond to the values in the "Name.ID" column. Default: data.frame(Name.ID=c("ProfileId"), Case.ID=c("ProfileId"),Sample.Type=c("type"), Diagnosis="NotDef", Source=c("NotDef"),stringsAsFactors=FALSE, row.names = c("ProfileId")).

studyType a single character string representing the type of study. The possible choices are: "LD" and "GeneAware". The type of study affects the way the estimation of the allelic fraction is done. Default: "LD".

genoSource a single character string corresponding to the type of file with the genotype and the allele information of the profile that will be provided in the 'profileFile' slot. The valid options are: "VCF", "generic", "snp-pileup", and "bam". The "generic" format CSV file must have at least these columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R', and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele, and 'File1A' is the depth of the specific alternative allele. Finally, in the case of a "VCF" file, the file must have at least those genotype fields: GT, AD, and DP. Default: NULL.

`blockTypeId` a single `character` string corresponding to the block type used to extract the block identifiers. The block type must be present in the GDS Reference Annotation file. Default: `"GeneS.Ensembl.Hsapiens.v86"`.

`reference` a single `character` string with two possible values: '1KGv1.0', '1k_hgdpV0.1'. It specifies the type of inference. Default: `"1KGv1.0"`.

`genome` a `character` string with one possible value: 'HG38'. It specifies the genome used. Default: `"HG38"`.

`chrInfo` a `vector` of positive `integer` values representing the length of the chromosomes. See 'details' section. Default: `seqlengths(Hsapiens)[seq_len(25)]`.

`paramAncestry` a `list` of parameters related to ancestry. The `list` should contain those three entries: `ScanBamParam`, `PileupParam`, and `yieldSize`. Default: `list(ScanBamParam=NULL, PileupParam=NULL,yieldSize=10000000)`.

`profileFile` a `character` string representing the path to the file with genotype and the allele information of the profile. The format of the accepted file is determined by the value in the 'genoSource' slot. A profile would have a file with extension "*vcf.gz" when 'genoSource' slot is "VCF". If 'genoSource' is "generic" or "snp-pileup", then "*.txt.gz". If 'genoSource' is "bam", then "*.bam" (the file needs to be indexed with a existing corresponding "*.bai" file). Default: `NULL`.

`profileFileGeno` not used in the current version

`pathProfileGDS` a single `character` string representing the path to the directory where the GDS Profile files will be created. If specified, the directory must exist. Default: `NULL`.

`fileReferenceGDS` a single `character` string representing the file name of the Reference GDS file. If specified, the file must exist. Default: `NULL`.

`fileReferenceAnnotGDS` a `character` string representing the file name of the Population Reference GDS Annotation file. If specified, the file must exist. Default: `NULL`.

`inferenceType` a single `character` string representing the genotype ancestry inference method. The two possible values: 'PCAknn' and 'haploAdmixture'. Default: `"PCAknn"`.

`sampleRef` a `vector` of `character` strings that represent a subset of sample identifiers from the GDS Reference File. Those samples will be used for the ancestry inference. If `NULL`, all the samples are retained. This is usefull if you want to removed some specific samples (ex: samples with admixed ancestry). Default: `NULL`.

`batch` a single positive `integer` representing the current identifier for the batch. Beware, this field is not stored anymore. Default: `1L`.

`prefix` a single `character` string representing at the beginning of the synthetic name id. Default: `"1"`.

`nbSim` a positive `integer` representing number of replicated synthitic data (restricted to 1 for the moment). Default: `1L`.

`posOffset` a single `integer` that is added to the SNP position to switch from 0-based to 1-based coordinate when needed (or reverse). Default: `-1L`.

`minCov` a single positive `integer` representing the minimum required coverage. Default: `10L`.

`minProb` a single `numeric` between `0` and `1` representing the probability that the calculated genotype call is correct. Default: `0.999`.

`seqError` a single `numeric` between `0` and `1` representing the probability of sequencing error. Default: `0.001`.

`seqErrorSyn` a single `numeric` between `0` and `1` representing the probability of sequencing error for synthetic. Default: `0.001`.

`pRecomb` a single positive `numeric` between `0` and `1` that represents the frequency of phase switching in the synthetic profiles. Default: `0.01`.

`np` a single positive `integer` specifying the number of threads to be used. Default: `1L`.

`listPos` a `data.frame` containing 2 columns named: "snp.chromosome" and "snp.position". The first column, called "snp.chromosome", contains the name of the chromosome where the SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome. Default: `NULL`.

`syntheticRefDF` a `data.frame` containing a subset of reference profiles for each sub-population present in the Reference GDS file. The `data.frame` must have those columns:

sample.id a `character` string representing the sample identifier.

pop.group a `character` string representing the subcontinental population assigned to the sample.

superPop a `character` string representing the super-population assigned to the sample.

Default: `NULL`.

`pruningMethod` a single `character` string representing the method that will be used to calculate the linkage disequilibrium in the `snpgdsLDpruning`() function. The 4 possible values are: "corr", "r", "dprime", and "composite". Default: `"corr"`.

`slideWindowMaxBP` a single positive `integer` that represents the maximum basepairs (bp) in the sliding window. This parameter is used for the LD pruning done by the `snpgdsLDpruning` function. Default: `500000L`.

`thresholdLD` a single positive `numeric` value that represents the LD threshold used in the `snpgdsLDpruning` function. Default: `sqrt(0.1)`.

`specificSNV` a `data.frame` containing 2 columns. The first column, called "snp.chromosome" contains the name of the chromosome where the SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome. Optionally, the column "snvKeep" contains the SNV index in the reference or -1 if not in the reference. It is used during the pruning step. Default: `NULL`.

`genoType` a single `character` string representing the field in the GDS profile file that contains all the genotype information. Default: `"geno.ref"`.

`phaseType` a TODO. Default: `"phase.ref"`.

`phase` a `logical` indicating TODO. Default: `FALSE`.

`PCAmissingRate` a positive `numeric` representing the maximum missing rate retained accepted to use SNPs in the PCA analysis done with the snpgdsPCA function. If `NaN`, no missing threshold. Default: `0.025`.

`PCAalgorithm` a single `character` string representing the algorithm to use with the snpgdsPCA function. The algorithm must be implemented and available to the snpgdsPCA function. The current options are: 'exact' or 'randomized'. Default: `"exact"`.

`eigenCount` a single `integer` indicating the number of eigenvectors that will be in the output of the snpgdsPCA function; if 'eigenCount' <= 0, then all eigenvectors are returned. Default: `32L`.

`eigenCountSyn` a TODO. Default: `15L`.

kList a vector of positive integer representing the values tested for the *K* parameter. The *K* parameter represents the number of neighbors used in the K-nearest neighbor analysis. Default: seq(2L, 15L, 1L).

pcaList a vector of positive integer representing the values tested for the *D* parameter. The *D* parameter represents the number of dimensions used in the PCA analysis. Default: seq(2L, 15L, 1L).

fieldPopInRef a single character string representing the name of the column that contains the known ancestry for the reference profiles in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter). Default: "superPop".

fieldPopInfAnc a character string representing the name of the column in the data frame that contains the inferred super-population ancestry for the samples. The column should be present in the data frame. Default: "superPop".

fieldSubPop a character string representing the name of the column in the Population Reference GDS file (corresponding to the fileReferenceGDS parameter) that contains the sub-population information for the samples. The column should be present in the Population Reference GDS file. Default: "pop.group".

anchorId a character string representing the name of the identifier of the quasi non-admixed profile from the reference. Default: "sp0.95v0.2".

pathADMIXTURE a character string representing the path pf the software ADMIXTURE and the binary. If it use a function using ADMIXTURE, then if it is null, it will try to find it with Sys.which("admixture"). Default: NULL.

seedADMIXTURE a single integer seed for ADMIXTURE program. Default: 432L.

cleanADMIXTUREFiles a single logical indicating if RAIDS removes ADMIXTURE files after processing. Default: TRUE.

verbose a logical indicating if message information should be printed. Default: FALSE.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

---

reference *Generic function for getting the reference slot in a class*

Description

A generic function for getting the reference slot in a S4 object.

Usage

```
reference(x)
```

Arguments

x	a S4 object.

Value

a value from the reference slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(reference="character"))

# Create an instance
obj <- new("MyClass", reference="123")

# Call the generic (assuming a method is implemented)
# reference(obj)
```

`reference,RAIDSparam-method`

A getter for the reference slot in a RAIDSparam class

Description

A function for getting the reference slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
reference(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

a single `character` string with two possible values: '1KGv1.0', '1k_hgdpV0.1'. It specifies the type of inference.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the reference slot for the object
reference(paramDemo)
```

`reference<-` *Generic function for replacement of reference slot in a class*

Description

A generic function for replacement of reference slot in a S4 object.

Usage

```
reference(x) <- value
```

Arguments

`x`	a S4 object.
`value`	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(reference="character"))

# Create an instance
obj <- new("MyClass", reference="123")

# Call the generic (assuming a method is implemented)
# reference(obj) <- "333"
```

`reference<-,RAIDSparam-method` *Setter function for replacement of reference slot in a RAIDSparam class*

Description

A function for replacement of reference slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
reference(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `character` string with two possible values: '1KGv1.0', '1k_hgdpV0.1'. It specifies the type of inference.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the new reference to the reference slot in the object
reference(paramDemo) <- "1k_hgdpV0.1"
```

`runADMIXTURE` *Call ADMIXTURE*

Description

TODO

Usage

```
runADMIXTURE(pathOut, fileBed, seedAdmix = 3435L, pRAIDS)
```

Arguments

pathOut	a character representing path where the file for ADMIXTURE (BED,BIM, FAM, and P.in) are located and where the output of ADMIXTURE will be generated.
fileBed	a character represented the BED file name (without path) for a
seedAdmix	a integer representing the seed pass to ADMIXTURE. Default: 3435L.
pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Details

TODO

Value

0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Not run:
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

pathOut <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathOut)){
    dir.create(pathOut)
}

# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID
```

```
## Path to admixture
## pathADMIXTURE <- "PATH/admixture"

pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=fileReferenceGDS,
                     fieldPopInfAnc="SuperPop",
                     pathADMIXTURE=pathADMIXTURE)
matPIndex <- RAIDS:::computePmatrix(pathOut=pathOut, pRAIDS=pRAIDS)
RAIDS:::generateProfilePlinkFiles(pathOut, pRAIDS)

profileQ <- runADMIXTURE(pathOut = file.path(pathOut, "profileAdmixture"),
    fileBed = paste0(pedStudy(pRAIDS)$Name.ID[1], ".bed"),
    seedAdmix=seedADMIXTURE(pRAIDS), pRAIDS)


dir(file.path(pathOut,"profileAdmixture"))
print(profileQ)
unlink(file.path(pathOut), recursive = TRUE)

## End(Not run)
```

`runExomeAncestry` *Run most steps leading to the ancestry inference call on a specific exome profile*

Description

This function runs most steps leading to the ancestry inference call on a specific exome profile. First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population reference GDS file.

Usage

```
runExomeAncestry(
  pedStudy,
  studyDF,
  pathProfileGDS,
  pathGeno,
  pathOut,
  fileReferenceGDS,
  fileReferenceAnnotGDS,
  chrInfo,
  syntheticRefDF,
  genoSource = c("snp-pileup", "generic", "VCF"),
  np = 1L,
```

```
  verbose = FALSE
)
```

Arguments

pedStudy	a data.frame with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", "Source". All columns must be in character strings (no factor). The data.frame must contain the information for all the samples passed in the listSamples parameter. Only filePedRDS or pedStudy can be defined.
studyDF	a data.frame containing the information about the study associated to the analysed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be in character strings (no factor).
pathProfileGDS	a character string representing the path to the directory where the GDS Profile files will be created. Default: NULL.
pathGeno	a character string representing the path to the directory containing the VCF output of SNP-pileup for each sample. The SNP-pileup files must be compressed (gz files) and have the name identifiers of the samples. A sample with "Name.ID" identifier would have an associated file called if genoSource is "VCF", then "Name.ID.vcf.gz", if genoSource is "generic", then "Name.ID.generic.txt.gz" if genoSource is "snp-pileup", then "Name.ID.txt.gz".
pathOut	a character string representing the path to the directory where the output files are created.
fileReferenceGDS	a character string representing the file name of the Reference GDS file. The file must exist.
fileReferenceAnnotGDS	a character string representing the file name of the Population Reference GDS Annotation file. The file must exist.
chrInfo	a vector of positive integer values representing the length of the chromosomes. See 'details' section.
syntheticRefDF	a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns: **sample.id** a character string representing the sample identifier. **pop.group** a character string representing the subcontinental population assigned to the sample. **superPop** a character string representing the super-population assigned to the sample.
genoSource	a character string with two possible values: 'snp-pileup', 'generic' or 'VCF'. It specifies if the genotype files are generated by snp-pileup (Facets) or are a generic format CSV file with at least those columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R' and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele and 'File1A' is the depth of the specific alternative allele. Finally the file can be a VCF file with at least those genotype fields: GT, AD, DP.

np	a single positive integer specifying the number of threads to be used. Default: 1L.
verbose	a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Details

The runExomeAncestry() function generates 3 types of files in the OUTPUT directory.

Ancestry Inference The ancestry inference CSV file (".Ancestry.csv" file)

Inference Informaton The inference information RDS file (".infoCall.rds" file)

Synthetic Information The parameter information RDS files from the synthetic inference ("KNN.synt.*.rds" files in a sub-directory)

In addition, a sub-directory (named using the profile ID) is also created.

Value

The integer 0L when successful. See details section for more information about the generated output files.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

#####################################################################
## Load the information about the profile
#####################################################################
data(demoPedigreeEx1)
head(demoPedigreeEx1)

#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
```

```
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS  <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
pathGeno <- file.path(dataDir, "example", "snpPileup")

#####################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")

pathOut <- file.path(tempdir(), "res.out")

#####################################################################
## A data frame containing general information about the study
## is also required. The data frame must have
## those 3 columns: "studyID", "study.desc", "study.platform"
#####################################################################
studyDF <- data.frame(study.id="MYDATA",
                      study.desc="Description",
                      study.platform="PLATFORM",
                      stringsAsFactors=FALSE)

#####################################################################
## Fix seed to ensure reproducible results
#####################################################################
set.seed(3043)

gds1KG <- snpgdsOpen(fileReferenceGDS)
dataRef <- select1KGPop(gds1KG, nbProfiles=2L)
closefn.gds(gds1KG)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]



        runExomeAncestry(pedStudy=demoPedigreeEx1, studyDF=studyDF,
            pathProfileGDS=pathProfileGDS,
            pathGeno=pathGeno,
            pathOut=pathOut,
            fileReferenceGDS=fileReferenceGDS,
```

```
            fileReferenceAnnotGDS=fileAnnotGDS,
            chrInfo=chrInfo,
            syntheticRefDF=dataRef,
            genoSource="snp-pileup")

        unlink(pathProfileGDS, recursive=TRUE, force=TRUE)
        unlink(pathOut, recursive=TRUE, force=TRUE)

}
```

`runRNAAncestry` *Run most steps leading to the ancestry inference call on a specific RNA profile*

Description

This function runs most steps leading to the ancestry inference call on a specific RNA profile. First, the function creates the Profile GDS file for the specific profile using the information from a RDS Sample description file and the Population Reference GDS file.

Usage

```
runRNAAncestry(
  pedStudy,
  studyDF,
  pathProfileGDS,
  pathGeno,
  pathOut,
  fileReferenceGDS,
  fileReferenceAnnotGDS,
  chrInfo,
  syntheticRefDF,
  genoSource = c("snp-pileup", "generic", "VCF"),
  np = 1L,
  blockTypeID,
  verbose = FALSE
)
```

Arguments

`pedStudy`	a `data.frame` with those mandatory columns: "Name.ID", "Case.ID", "Sample.Type", "Diagnosis", "Source". All columns must be in `character` strings (no factor). The `data.frame` must contain the information for all the samples passed in the `listSamples` parameter. Only `filePedRDS` or `pedStudy` can be defined.

studyDF — a data.frame containing the information about the study associated to the analysed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be in character strings (no factor).

pathProfileGDS — a character string representing the path to the directory where the GDS Profile files will be created. Default: NULL.

pathGeno — a character string representing the path to the directory containing the VCF output of SNP-pileup for each sample. The SNP-pileup files must be compressed (gz files) and have the name identifiers of the samples. A sample with "Name.ID" identifier would have an associated file called if genoSource is "VCF", then "Name.ID.vcf.gz", if genoSource is "generic", then "Name.ID.generic.txt.gz" if genoSource is "snp-pileup", then "Name.ID.txt.gz".

pathOut — a character string representing the path to the directory where the output files are created.

fileReferenceGDS — a character string representing the file name of the Population Reference GDS file. The file must exist.

fileReferenceAnnotGDS — a character string representing the file name of the Population Reference GDS Annotation file. The file must exist.

chrInfo — a vector of positive integer values representing the length of the chromosomes. See 'details' section.

syntheticRefDF — a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns:

sample.id a character string representing the sample identifier.

pop.group a character string representing the subcontinental population assigned to the sample.

superPop a character string representing the super-population assigned to the sample.

genoSource — a character string with two possible values: 'snp-pileup', 'generic' or 'VCF'. It specifies if the genotype files are generated by snp-pileup (Facets) or are a generic format CSV file with at least those columns: 'Chromosome', 'Position', 'Ref', 'Alt', 'Count', 'File1R' and 'File1A'. The 'Count' is the depth at the specified position; 'FileR' is the depth of the reference allele and 'File1A' is the depth of the specific alternative allele. Finally the file can be a VCF file with at least those genotype fields: GT, AD, DP.

np — a single positive integer specifying the number of threads to be used. Default: 1L.

blockTypeID — a character string corresponding to the block type used to extract the block identifiers. The block type must be present in the GDS Reference Annotation file.

verbose — a logical indicating if messages should be printed to show how the different steps in the function. Default: FALSE.

Details

The runExomeAncestry() function generates 3 types of files in the OUTPUT directory.

Ancestry Inference The ancestry inference CSV file (".Ancestry.csv" file)

Inference Informaton The inference information RDS file (".infoCall.rds" file)

Synthetic Information The parameter information RDS files from the synthetic inference ("KNN.synt.*.rds" files in a sub-directory)

In addition, a sub-directory (named using the profile ID) is also created.

Value

The integer `0L` when successful. See details section for more information about the generated output files.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

References

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

Examples

```
## Required library for GDS
library(SNPRelate)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

#####################################################################
## Load the information about the profile
#####################################################################
data(demoPedigreeEx1)
head(demoPedigreeEx1)

#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#####################################################################
## The Sample SNP pileup files (one per sample) need
## to be located in the same directory.
#####################################################################
```

```
pathGeno <- file.path(dataDir, "example", "snpPileup")

#####################################################################
## The path where the Profile GDS Files (one per sample)
## will be created need to be specified.
#####################################################################
pathProfileGDS <- file.path(tempdir(), "out.tmp")

pathOut <- file.path(tempdir(), "res.out")

#####################################################################
## A data frame containing general information about the study
## is also required. The data frame must have
## those 3 columns: "studyID", "study.desc", "study.platform"
#####################################################################
studyDF <- data.frame(study.id="MYDATA",
                      study.desc="Description",
                      study.platform="PLATFORM",
                      stringsAsFactors=FALSE)

#####################################################################
## Fix seed to ensure reproducible results
#####################################################################
set.seed(3043)

gds1KG <- snpgdsOpen(fileReferenceGDS)
dataRef <- select1KGPop(gds1KG, nbProfiles=2L)
closefn.gds(gds1KG)

## Required library for this example to run correctly
if (requireNamespace("Seqinfo", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- Seqinfo::seqlengths(BSgenome.Hsapiens.UCSC.hg38::Hsapiens)[1:25]



        runRNAAncestry(pedStudy=demoPedigreeEx1, studyDF=studyDF,
            pathProfileGDS=pathProfileGDS,
            pathGeno=pathGeno,
            pathOut=pathOut,
            fileReferenceGDS=fileReferenceGDS,
            fileReferenceAnnotGDS=fileAnnotGDS,
            chrInfo=chrInfo,
            syntheticRefDF=dataRef,
            blockTypeID="GeneS.Ensembl.Hsapiens.v86",
            genoSource="snp-pileup")

        unlink(pathProfileGDS, recursive=TRUE, force=TRUE)
        unlink(pathOut, recursive=TRUE, force=TRUE)
```

```
}
```

---

runSynADMIXTURE *run ADMIXTURE for the synthetic data*

Description

TODO

Usage

```
runSynADMIXTURE(pathOut, matPIndex, pRAIDS)
```

Arguments

pathOut	a character representing path where the directory for the synthetic admixture will be processed
matPIndex	a array of the frequency matrix.
pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Details

TODO

Value

0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Not run:
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

pathOut <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathOut)){
    dir.create(pathOut)
}

# The name of the synthetic study
```

```
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID
## Path to admixture
## pathADMIXTURE <- "PATH/admixture"
pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=fileReferenceGDS,
                     fieldPopInfAnc="SuperPop",
                     pathADMIXTURE=pathADMIXTURE)
matPIndex <- RAIDS:::computePmatrix(pathOut=pathOut, pRAIDS=pRAIDS)
RAIDS:::generateProfilePlinkFiles(pathOut, pRAIDS)

syntheticQ <- runSynADMIXTURE(pathOut=pathOut, matPIndex=matPIndex, pRAIDS=pRAIDS)

dir(file.path(pathOut))
head(syntheticQ)
unlink(file.path(pathOut), recursive = TRUE)

## End(Not run)
```

`sampleRef` *Generic function for getting the sampleRef slot in a class*

Description

A generic function for getting the sampleRef slot in a S4 object.

Usage

```
sampleRef(x)
```

Arguments

x	a S4 object.

Value

a value from the sampleRef slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(sampleRef="character"))

# Create an instance
obj <- new("MyClass", sampleRef="123")

# Call the generic (assuming a method is implemented)
# sampleRef(obj)
```

---

sampleRef,RAIDSparam-method

A getter for the inferenceType slot in a RAIDSparam class

Description

A function for getting the sampleRef slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
sampleRef(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

`NULL` or a `vector` of `character` strings that represent a subset of sample identifiers from the GDS Reference File. Those samples will be used for the ancestry inference. If `NULL`, all the samples are retained.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the sampleRef slot for the object
sampleRef(paramDemo)
```

`sampleRef<-` *Generic function for replacement of sampleRef slot in a class*

Description

A generic function for replacement of sampleRef slot in a S4 object.

Usage

```
sampleRef(x) <- value
```

Arguments

`x`	a S4 object.
`value`	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(sampleRef="character"))

# Create an instance
obj <- new("MyClass", sampleRef="123")

# Call the generic (assuming a method is implemented)
# sampleRef(obj) <- "333"
```

sampleRef<-,RAIDSparam-method

Setter function for replacement of sampleRef slot in a RAIDSparam class

Description

A function for replacement of sampleRef slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
sampleRef(x) <- value
```

Arguments

x	a RAIDSparam object.
value	NULL or a vector of character strings that represent a subset of sample identifiers from the GDS Reference File. Those samples will be used for the ancestry inference. If NULL, all the samples are retained.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new values to the sampleRef slot in the object
sampleRef(paramDemo) <- c("Sample1", "Sample2")
```

seedADMIXTURE *Generic function for getting the seedADMIXTURE slot in a class*

Description

A generic function for getting the seedADMIXTURE slot in a S4 object.

Usage

```
seedADMIXTURE(x)
```

Arguments

x	a S4 object.

Value

a value from the seedADMIXTURE slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(seedADMIXTURE="character"))

# Create an instance
obj <- new("MyClass", seedADMIXTURE="123")

# Call the generic (assuming a method is implemented)
# seedADMIXTURE(obj)
```

seedADMIXTURE,RAIDSparam-method *A getter for the seedADMIXTURE slot in a RAIDSparam class*

Description

A function for getting the seedADMIXTURE slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
seedADMIXTURE(x)
```

Arguments

x	a RAIDSparam object.

Value

a single integer seed for ADMIXTURE program.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the seedADMIXTURE slot for the object
seedADMIXTURE(paramDemo)
```

---

seedADMIXTURE<- *Generic function for replacement of seedADMIXTURE slot in a class*

Description

A generic function for replacement of seedADMIXTURE slot in a S4 object.

Usage

```
seedADMIXTURE(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(seedADMIXTURE="character"))

# Create an instance
obj <- new("MyClass", seedADMIXTURE="123")

# Call the generic (assuming a method is implemented)
# seedADMIXTURE(obj) <- "333"
```

`seedADMIXTURE<-,RAIDSparam-method` *Setter function for replacement of seedADMIXTURE slot in a RAIDSparam class*

Description

A function for replacement of seedADMIXTURE slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
seedADMIXTURE(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `integer` seed for the ADMIXTURE program.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the seedADMIXTURE slot in the object
seedADMIXTURE(paramDemo) <-  33L
```

`select1KGPop` *Random selection of a specific number of reference profiles in each subcontinental population present in the 1KG GDS file*

Description

The function randomly selects a fixed number of reference for each subcontinental population present in the 1KG GDS file. When a subcontinental population has less samples than the fixed number, all samples from the subcontinental population are selected.

Usage

```
select1KGPop(gdsReference, nbProfiles)
```

Arguments

`gdsReference`	an object of class gds.class (a GDS file), the opened 1KG GDS file.
`nbProfiles`	a single positive `integer` representing the number of samples that will be selected for each subcontinental population present in the 1KG GDS file. If the number of samples in a specific subcontinental population is smaller than the `nbProfiles`, the number of samples selected in this subcontinental population will correspond to the size of this population.

Value

a `data.frame` containing those columns:

sample.id a `character` string representing the sample identifier.

pop.group a `character` string representing the subcontinental population assigned to the sample.

superPop a `character` string representing the super-population assigned to the sample.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## The number of samples needed by subcontinental population
## The number is small for demonstration purpose
nbProfiles <- 5L

## Open 1KG GDS Demo file
## This file only one superpopulation (for demonstration purpose)
dataDir <- system.file("extdata", package="RAIDS")
fileGDS <- file.path(dataDir, "PopulationReferenceDemo.gds")
```

```
gdsFileOpen <- openfn.gds(fileGDS, readonly=TRUE)

## Extract a selected number of random samples
## for each subcontinental population
## In the 1KG GDS Demo file, there is one subcontinental population
dataR <- select1KGPop(gdsReference=gdsFileOpen, nbProfiles=nbProfiles)

## Close the 1KG GDS Demo file (important)
closefn.gds(gdsFileOpen)
```

`select1KGPopForSynthetic`

Random selection of a specific number of reference profiles in each subcontinental population present in the 1KG GDS file (same as select1KGPop but the function doesn't need gds object as parameters but the file name of the referenceGDS)

Description

The function randomly selects a fixed number of reference for each subcontinental population present in the 1KG GDS file. When a subcontinental population has less samples than the fixed number, all samples from the subcontinental population are selected.

Usage

```
select1KGPopForSynthetic(fileReferenceGDS, nbProfiles)
```

Arguments

`fileReferenceGDS`	a `character` string representing the file name of the Reference GDS file. The file must exist.
`nbProfiles`	a single positive `integer` representing the number of samples that will be selected for each subcontinental population present in the 1KG GDS file. If the number of samples in a specific subcontinental population is smaller than the `nbProfiles`, the number of samples selected in this subcontinental population will correspond to the size of this population.

Value

a `data.frame` containing those columns:

sample.id a `character` string representing the sample identifier.

pop.group a `character` string representing the subcontinental population assigned to the sample.

superPop a `character` string representing the super-population assigned to the sample.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## The number of samples needed by subcontinental population
## The number is small for demonstration purpose
nbProfiles <- 5L

## 1KG GDS Demo file
## This file only one superpopulation (for demonstration purpose)
dataDir <- system.file("extdata", package="RAIDS")
fileGDS <- file.path(dataDir, "PopulationReferenceDemo.gds")

## Extract a selected number of random samples
## for each subcontinental population
## In the 1KG GDS Demo file, there is one subcontinental population
dataR <- select1KGPopForSynthetic(fileReferenceGDS=fileGDS, nbProfiles=nbProfiles)
```

`selectForSynthetic` *Random selection of a specific number of reference profiles in each subcontinental population present in the refrence GDS file. (call select1KGPopForSynthetic or selectHGDP1kgForSynthetic)*

Description

The function randomly selects a fixed number of reference for each subcontinental population present in the 1KG GDS file. When a subcontinental population has less samples than the fixed number, all samples from the subcontinental population are selected.

Usage

```
selectForSynthetic(pRAIDS, nbProfiles = 30L)
```

Arguments

`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters
`nbProfiles`	a single positive `integer` representing the number of samples that will be selected for each subcontinental population present in the 1KG GDS file. If the number of samples in a specific subcontinental population is smaller than the `nbProfiles`, the number of samples selected in this subcontinental population will correspond to the size of this population.

Value

a data.frame containing those columns:

sample.id a character string representing the sample identifier.

pop.group a character string representing the subcontinental population assigned to the sample.

superPop a character string representing the super-population assigned to the sample.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata", package="RAIDS")


#####################################################################
## The 1KG GDS file and the 1KG SNV Annotation GDS file
## need to be located in the same directory
## Note that the 1KG GDS file used for this example is a
## simplified version and CANNOT be used for any real analysis
#####################################################################
path1KG <- file.path(dataDir, "tests")

fileReferenceGDS <- file.path(path1KG, "ex1_good_small_1KG.gds")
fileAnnotGDS <- file.path(path1KG, "ex1_good_small_1KG_Annot.gds")

#########################################################################
## Fix seed to ensure reproducible results
#########################################################################
set.seed(3043)

pRAIDS <- RAIDSparam(
    fileReferenceGDS=fileReferenceGDS,
    fileReferenceAnnotGDS=fileAnnotGDS
    )

## The number of samples needed by subcontinental population
## The number is small for demonstration purpose
nbProfiles <- 5L

## Extract a selected number of random samples
## for each subcontinental population
## In the 1KG GDS Demo file, there is one subcontinental population
dataR <- selectForSynthetic(pRAIDS=pRAIDS, nbProfiles=nbProfiles)
```

seqError *Generic function for getting the seqError slot in a class*

Description

A generic function for getting the seqError slot in a S4 object.

Usage

```
seqError(x)
```

Arguments

x	a S4 object.

Value

a value from the seqError slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(seqError="character"))

# Create an instance
obj <- new("MyClass", seqError="123")

# Call the generic (assuming a method is implemented)
# seqError(obj)
```

seqError,RAIDSparam-method
A getter for the seqError slot in a RAIDSparam class

Description

A function for getting the seqError slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
seqError(x)
```

Arguments

x	a RAIDSparam object.

Value

a single numeric between 0 and 1 representing the probability of sequencing error.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the seqError slot for the object
seqError(paramDemo)
```

---

seqError<- *Generic function for replacement of seqError slot in a class*

Description

A generic function for replacement of seqError slot in a S4 object.

Usage

```
seqError(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(seqError="character"))

# Create an instance
obj <- new("MyClass", seqError="123")

# Call the generic (assuming a method is implemented)
# seqError(obj) <- "333"
```

---

seqError<-,RAIDSparam-method

Setter function for replacement of seqError slot in a RAIDSparam class

---

Description

A function for replacement of seqError slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
seqError(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single numeric between 0 and 1 representing the probability of sequencing error.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the seqError slot in the object
seqError(paramDemo) <- 0.03
```

seqErrorSyn *Generic function for getting the seqErrorSyn slot in a class*

Description

A generic function for getting the seqErrorSyn slot in a S4 object.

Usage

```
seqErrorSyn(x)
```

Arguments

x	a S4 object.

Value

a value from the seqErrorSyn slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(seqErrorSyn="character"))

# Create an instance
obj <- new("MyClass", seqErrorSyn="123")

# Call the generic (assuming a method is implemented)
# seqErrorSyn(obj)
```

---

seqErrorSyn,RAIDSparam-method
A getter for the seqErrorSyn slot in a RAIDSparam class

Description

A function for getting the seqErrorSyn slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
seqErrorSyn(x)
```

Arguments

x	a RAIDSparam object.

Value

a single numeric between 0 and 1 representing the probability of sequencing error for synthetic.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the seqErrorSyn slot for the object
seqErrorSyn(paramDemo)
```

---

seqErrorSyn<- *Generic function for replacement of seqErrorSyn slot in a class*

Description

A generic function for replacement of seqErrorSyn slot in a S4 object.

Usage

```
seqErrorSyn(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(seqErrorSyn="character"))

# Create an instance
obj <- new("MyClass", seqErrorSyn="123")

# Call the generic (assuming a method is implemented)
# seqErrorSyn(obj) <- "333"
```

`seqErrorSyn<-,RAIDSparam-method`

Setter function for replacement of seqErrorSyn slot in a RAIDSparam class

---

Description

A function for replacement of seqErrorSyn slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
seqErrorSyn(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single `numeric` between `0` and `1` representing the probability of sequencing error for synthetic.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign a new value to the seqErrorSyn slot in the object
seqErrorSyn(paramDemo) <- 0.03
```

slideWindowMaxBP *Generic function for getting the slideWindowMaxBP slot in a class*

Description

A generic function for getting the slideWindowMaxBP slot in a S4 object.

Usage

```
slideWindowMaxBP(x)
```

Arguments

x	a S4 object.

Value

a value from the slideWindowMaxBP slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(slideWindowMaxBP="character"))

# Create an instance
obj <- new("MyClass", slideWindowMaxBP="123")

# Call the generic (assuming a method is implemented)
# slideWindowMaxBP(obj)
```

slideWindowMaxBP,RAIDSparam-method
A getter for the slideWindowMaxBP slot in a RAIDSparam class

Description

A function for getting the slideWindowMaxBP slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
slideWindowMaxBP(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive integer that represents the maximum basepairs (bp) in the sliding window. This parameter is used for the LD pruning done by the snpgdsLDpruning function.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the slideWindowMaxBP slot for the object
slideWindowMaxBP(paramDemo)
```

---

slideWindowMaxBP<- *Generic function for replacement of slideWindowMaxBP slot in a class*

Description

A generic function for replacement of slideWindowMaxBP slot in a S4 object.

Usage

```
slideWindowMaxBP(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(slideWindowMaxBP="character"))

# Create an instance
obj <- new("MyClass", slideWindowMaxBP="123")

# Call the generic (assuming a method is implemented)
# slideWindowMaxBP(obj) <- "333"
```

`slideWindowMaxBP<-,RAIDSparam-method`

Setter function for replacement of slideWindowMaxBP slot in a RAIDSparam class

---

Description

A function for replacement of slideWindowMaxBP slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
slideWindowMaxBP(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a single positive `integer` that represents the maximum basepairs (bp) in the sliding window. This parameter is used for the LD pruning done by the `snpgdsLDpruning` function.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the slideWindowMaxBP slot in the object
slideWindowMaxBP(paramDemo) <- 1000L
```

snpPositionDemo *A small* data.frame *containing the SNV information.*

Description

The object is a data.frame with 17 columns.

Usage

```
data(snpPositionDemo)
```

Format

The data.frame containing the information about the synthetic profiles. The data.frame contains 4 columns:

cnt.tot a integer representing the number of reads at the SNV position.

cnt.ref a integer representing the number of reads corresponding to the reference at the SNV position.

cnt.alt a integer representing the number of reads different than the reference at the SNV position.

snp.pos a integer representing the position of the SNV on the chromosome.

snp.chr a integer representing the chromosome on which the SNV is located.

normal.geno a integer representing the genotype (0=wild-type reference; 1=heterozygote; 2=homozygote alternative; 3=unkown).

pruned a logical indicated if the SNV is pruned.

snp.index a integer representing the index of the SNV in the reference SNV GDS file.

keep a logical indicated if the genotype exists for the SNV.

hetero a logical indicated if the SNV is heterozygote.

homo a logical indicated if the SNV is homozygote.

block.id a integer representing the block identifier associated to the current SNV.

phase a integer representing the block identifier associated to the current SNV.

lap a numeric representing the lower allelic fraction.

LOH a integer indicating if the SNV is in an LOH region (0=not LOH, 1=in LOH).

imbAR a integer indicating if the SNV is in an imbalanced region (-1=not classified as imbalanced or LOH, 0=in LOH; 1=tested positive for imbalance in at least 1 window).

freq a numeric representing the frequency of the variant in the the reference.

Details

This dataset can be used to test the calcAFMLRNA and tableBlockAF internal functions.

Value

The `data.frame` containing the information about the synthetic profiles. The `data.frame` contains 4 columns:

`cnt.tot` a `integer` representing the number of reads at the SNV position.

`cnt.ref` a `integer` representing the number of reads corresponding to the reference at the SNV position.

`cnt.alt` a `integer` representing the number of reads different than the reference at the SNV position.

`snp.pos` a `integer` representing the position of the SNV on the chromosome.

`snp.chr` a `integer` representing the chromosome on which the SNV is located.

`normal.geno` a `integer` representing the genotype (0=wild-type reference; 1=heterozygote; 2=homozygote alternative; 3=unkown).

`pruned` a `logical` indicated if the SNV is pruned.

`snp.index` a `integer` representing the index of the SNV in the reference SNV GDS file.

`keep` a `logical` indicated if the genotype exists for the SNV.

`hetero` a `logical` indicated if the SNV is heterozygote.

`homo` a `logical` indicated if the SNV is homozygote.

`block.id` a `integer` representing the block identifier associated to the current SNV.

`phase` a `integer` representing the block identifier associated to the current SNV.

`lap` a `numeric` representing the lower allelic fraction.

`LOH` a `integer` indicating if the SNV is in an LOH region (0=not LOH, 1=in LOH).

`imbAR` a `integer` indicating if the SNV is in an imbalanced region (-1=not classified as imbalanced or LOH, 0=in LOH; 1=tested positive for imbalance in at least 1 window).

`freq` a `numeric` representing the frequency of the variant in the the reference.

Examples

```
## Loading demo dataset containing SNV information
data(snpPositionDemo)

## Only use a subset of heterozygote SNVs related to one block
subset <- snpPositionDemo[which(snpPositionDemo$block.id == 2750 &
                  snpPositionDemo$hetero), c("cnt.ref", "cnt.alt", "phase")]

## Compute the log likelihood ratio based on the coverage of
## each allele in a specific block
result <- RAIDS:::calcAFMLRNA(subset)
head(result)
```

snvListVCF *Generate a VCF with the information from the SNPs that pass a cut-off threshold*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
snvListVCF(gdsReference, fileOut, offset = 0L, freqCutoff = NULL)
```

Arguments

gdsReference	an object of class gds.class (a GDS file), the 1KG GDS file.
fileOut	a character string representing the path and file name of the VCF file that will be created wit the retained SNP information. The file should have the ".vcf" extension.
offset	a single integer that is added to the SNP position to switch from 0-based to 1-based coordinate when needed (or reverse). Default: 0L.
freqCutoff	a single positive numeric specifying the cut-off to keep a SNP. If NULL, all SNPs are retained. Default: NULL.

Value

The integer 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Path to the demo pedigree file is located in this package
dataDir <- system.file("extdata", package="RAIDS")

## Demo 1KG Reference GDS file
fileGDS <- openfn.gds(file.path(dataDir,
                   "PopulationReferenceDemo.gds"))

## Output VCF file that will be created (temporary)
vcfFile <- file.path(tempdir(), "Demo_TMP_01.vcf")

## Create a VCF file with the SNV dataset present in the GDS file
```

```
## No cutoff on frequency, so all SNVs are saved
snvListVCF(gdsReference=fileGDS, fileOut=vcfFile, offset=0L,
                    freqCutoff=NULL)

## Close GDS file (IMPORTANT)
closefn.gds(fileGDS)

## Remove temporary VCF file
unlink(vcfFile, force=TRUE)
```

`snvListVCFRef` *Generate a VCF with the information from the SNPs that pass a cut-off threshold*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
snvListVCFRef(pRAIDS, fileOut, freqCutoff = NULL)
```

Arguments

`pRAIDS`	a TODO.
`fileOut`	a `character` string representing the path and file name of the VCF file that will be created wit the retained SNP information. The file should have the ".vcf" extension.
`freqCutoff`	a single positive `numeric` specifying the cut-off to keep a SNP. If `NULL`, all SNPs are retained. Default: `NULL`.

Value

The integer `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## TODO
```

specificSNV *Generic function for getting the specificSNV slot in a class*

Description

A generic function for getting the specificSNV slot in a S4 object.

Usage

```
specificSNV(x)
```

Arguments

x	a S4 object.

Value

a value from the specificSNV slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(specificSNV="character"))

# Create an instance
obj <- new("MyClass", specificSNV="123")

# Call the generic (assuming a method is implemented)
# specificSNV(obj)
```

specificSNV,RAIDSparam-method
A getter for the specificSNV slot in a RAIDSparam class

Description

A function for getting the specificSNV slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
specificSNV(x)
```

Arguments

x	a RAIDSparam object.

Value

NULL or a data.frame containing 2 columns. The first column, called "snp.chromosome" contains the name of the chromosome where the SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome. Optionally, the column "snvKeep" contains the SNV index in the reference or -1 if not in the reference. It is used during the pruning step.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the specificSNV slot for the object
specificSNV(paramDemo)
```

---

specificSNV<- *Generic function for replacement of specificSNV slot in a class*

Description

A generic function for replacement of specificSNV slot in a S4 object.

Usage

```
specificSNV(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(specificSNV="character"))

# Create an instance
obj <- new("MyClass", specificSNV="123")

# Call the generic (assuming a method is implemented)
# specificSNV(obj) <- "333"
```

`specificSNV<-,RAIDSparam-method`
Setter function for replacement of specificSNV slot in a RAIDSparam class

---

Description

A function for replacement of specificSNV slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
specificSNV(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	`NULL` or a `data.frame` containing 2 columns. The first column, called "snp.chromosome" contains the name of the chromosome wherethe SNV is located. The second column, called "snp.position", contains the position of the SNV on the chromosome. Optionally, the column "snvKeep" contains the SNV index in the reference or -1 if not in the reference.

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the specificSNV slot in the object
specificSNV(paramDemo) <- data.frame(snp.chromosome=c("1", "1"),
    snp.position=c(13333, 2222213))
```

---

splitSelectByPop *Group samples per subcontinental population*

Description

The function groups the samples per subcontinental population and generates a matrix containing the sample identifiers and where each column is a subcontinental population.

Usage

```
splitSelectByPop(dataRef)
```

Arguments

dataRef	a data.frame containing those columns:
	sample.id a character string representing the sample identifier.
	pop.group a character string representing the subcontinental population assigned to the sample.
	superPop a character string representing the super-population assigned to the sample.

Value

a matrix containing the sample identifiers and where each column is the name of a subcontinental population. The number of row corresponds to the number of samples for each subcontinental population.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## A data.frame containing samples from 2 subcontinental populations
demo <- data.frame(sample.id=c("SampleA", "SampleB", "SampleC", "SampleD"),
    pop.group=c("TSI", "TSI", "YRI", "YRI"),
    superPop=c("EUR", "EUR", "AFR", "AFR"))

## Generate a matrix populated with the sample identifiers and where
```

```
## each row is a subcontinental population
splitSelectByPop(dataRef=demo)
```

`studyDF` *Generic function for getting the studyDF slot in a class*

Description

A generic function for getting the studyDF slot in a class. The function is implemented for the `RAIDSparam` class.

Usage

```
studyDF(x)
```

Arguments

`x`	an object.

Value

an object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

`studyDF,RAIDSparam-method`
Getter for the studyDF slot in a RAIDSparam class

Description

A function for getting the studyDF slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
studyDF(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

a `data.frame`.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the studyDF slot for the object
studyDF(paramDemo)
```

---

studyDF<- *Generic function for replacement of studyDF slot in a class*

Description

A generic function for replacement of studyDF slot in a S4 object.

Usage

```
studyDF(x) <- value
```

Arguments

`x`	a S4 object.
`value`	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(studyDF="character"))

# Create an instance
obj <- new("MyClass", studyDF="123")

# Call the generic (assuming a method is implemented)
# studyDF(obj) <- "333"
```

`studyDF<-,RAIDSparam-method`
Replacement of studyDF slot in a `RAIDSparam` *object*

Description

A function for replacement of the studyDF slot in a `RAIDSparam` class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
studyDF(x) <- value
```

Arguments

`x`	a `RAIDSparam` object.
`value`	a `data.frame` containing the information about the study associated with the analyzed sample(s). The `data.frame` must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be character strings (no factors).

Value

the modified `RAIDSparam` object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Demo study data frame
studyDF <- data.frame(study.id="TEST1", study.desc="Study1",
    study.platform="CSHL", stringsAsFactors=FALSE)
```

```
## Assign the new table to the studyDF slot in the object
studyDF(paramDemo) <- studyDF
```

`studyDFSyn` *Generic function for getting the studyDFSyn slot in a class*

Description

A generic function for getting the studyDFSyn slot in a class. The function is implemented for the `RAIDSparam` class.

Usage

```
studyDFSyn(x)
```

Arguments

`x`	an object.

Value

an object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

`studyDFSyn,RAIDSparam-method`
A getter for the studyDFSyn slot in a RAIDSparam class

Description

A function for getting the studyDFSyn slot in a `RAIDSparam` class.

Usage

```
## S4 method for signature 'RAIDSparam'
studyDFSyn(x)
```

Arguments

`x`	a `RAIDSparam` object.

Value

a data.frame.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the studyDFSyn slot for the object
studyDFSyn(paramDemo)
```

---

studyDFSyn<- *Generic function for replacement of studyDFSyn slot in a class*

Description

A generic function for replacement of studyDFSyn slot in a S4 object.

Usage

```
studyDFSyn(x) <- value
```

Arguments

x	an S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(studyDFSyn="character"))

# Create an instance
obj <- new("MyClass", studyDFSyn="123")

# Call the generic (assuming a method is implemented)
# studyDFSyn(obj) <- "333"
```

---

studyDFSyn<-,RAIDSparam-method

Replacement of studyDFSyn slot in a RAIDSparam *object*

Description

A function for replacement of the studyDFSyn slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
studyDFSyn(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a data.frame containing the information about the synthetic data for the analyzed sample(s). The data.frame must have those 3 columns: "study.id", "study.desc", "study.platform". All columns must be character strings (no factors).

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Demo synthetic study data frame
studyDFSyn <- data.frame(study.id="TEST1", study.desc="Synthetic",
    study.platform="CSHL", stringsAsFactors=FALSE)
```

```
## Assign the new table to the studyDFSyn slot in the object
studyDFSyn(paramDemo) <- studyDFSyn
```

`studyType` *Generic function for getting the studyType slot in a class*

Description

A generic function for getting the studyType slot in a S4 object.

Usage

```
studyType(x)
```

Arguments

`x`	a S4 object.

Value

a value from the studyType slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(studyType="character"))

# Create an instance
obj <- new("MyClass", studyType="123")

# Call the generic (assuming a method is implemented)
# studyType(obj)
```

studyType,RAIDSparam-method

A getter for the studyType slot in a RAIDSparam class

Description

A function for getting the studyType slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
studyType(x)
```

Arguments

x	a RAIDSparam object.

Value

a single character string representing the type of study.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the studyType slot for the object
studyType(paramDemo)
```

studyType<-

Generic function for replacement of studyType slot in a class

Description

A generic function for replacement of studyType slot in a S4 object.

Usage

```
studyType(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(studyType="character"))

# Create an instance
obj <- new("MyClass", studyType="123")

# Call the generic (assuming a method is implemented)
# studyType(obj) <- "333"
```

---

studyType<-,RAIDSparam-method

Setter function for replacement of studyType slot in a RAIDSparam class

---

Description

A function for replacement of studyType slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
studyType(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single character string representing the type of study. The possible choices are: "LD" and "GeneAware". The study type affects how the allelic fraction is estimated.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign the new study type to the studyType slot in the object
studyType(paramDemo) <- "GeneAware"
```

`syntheticGeno` *Generate synthetic profiles for each cancer profile and 1KG reference profile combination and add them to the Profile GDS file*

Description

The functions uses one cancer profile in combination with one 1KG reference profile to generate an synthetic profile that is saved in the Profile GDS file.

When more than one 1KG reference profiles are specified, the function recursively generates synthetic profiles for each cancer profile + 1KG reference profile combination.

The number of synthetic profiles generated by combination is specified by the number of simulation requested.

Usage

```
syntheticGeno(
  gdsReference,
  gdsRefAnnot,
  fileProfileGDS,
  profileID,
  listSampleRef,
  nbSim = 1L,
  prefix = "",
  pRecomb = 0.01,
  minProb = 0.999,
  seqError = 0.001
)
```

Arguments

`gdsReference`	an object of class `gds.class` (a GDS file), the opened 1KG GDS file.
`gdsRefAnnot`	an object of class `gds.class` (a GDS file), the opened 1KG SNV Annotation GDS file.

fileProfileGDS	a character string representing the file name of Profile GDS file containing the information about the sample. The file must exist.
profileID	a character string representing the unique identifier of the cancer profile.
listSampleRef	a vector of character strings representing the sample identifiers of the 1KG selected reference samples.
nbSim	a single positive integer representing the number of simulations that will be generated per sample + 1KG reference combination. Default: 1L.
prefix	a character string that represent the prefix that will be added to the name of the synthetic profiles generated by the function. Default: "".
pRecomb	a single positive numeric between 0 and 1 that represents the frequency of phase switching in the synthetic profiles, Default: 0.01.
minProb	a single positive numeric between 0 and 1 that represents the probability that the genotype is correct. Default: 0.999.
seqError	a single positive numeric between 0 and 1 representing the sequencing error rate. Default: 0.001.

Value

The integer 0L when the function is successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")

## Profile GDS file (temporary)
fileNameGDS <- file.path(tempdir(), "ex1.gds")

## Copy the Profile GDS file demo that has been pruned and annotated
file.copy(file.path(dataDir, "ex1_demo_with_pruning_and_1KG_annot.gds"),
                fileNameGDS)

## Information about the synthetic data set
syntheticStudyDF <- data.frame(study.id="MYDATA.Synthetic",
        study.desc="MYDATA synthetic data", study.platform="PLATFORM",
        stringsAsFactors=FALSE)

## Add information related to the synthetic profiles into the Profile GDS
prepSynthetic(fileProfileGDS=fileNameGDS,
        listSampleRef=c("HG00243", "HG00150"), profileID="ex1",
        studyDF=syntheticStudyDF, nbSim=1L, prefix="synthTest",
        verbose=FALSE)
```

```
## The 1KG files
gds1KG <- snpgdsOpen(file.path(dataDir,
                               "ex1_good_small_1KG.gds"))
gds1KGAnnot <- openfn.gds(file.path(dataDir,
                               "ex1_good_small_1KG_Annot.gds"))

## Generate the synthetic profiles and add them into the Profile GDS
syntheticGeno(gdsReference=gds1KG, gdsRefAnnot=gds1KGAnnot,
        fileProfileGDS=fileNameGDS, profileID="ex1",
        listSampleRef=c("HG00243", "HG00150"), nbSim=1,
        prefix="synthTest",
        pRecomb=0.01, minProb=0.999, seqError=0.001)

## Open Profile GDS file
profileGDS <- openfn.gds(fileNameGDS)

tail(read.gdsn(index.gdsn(profileGDS, "sample.id")))

## Close GDS files (important)
closefn.gds(profileGDS)
closefn.gds(gds1KG)
closefn.gds(gds1KGAnnot)

## Remove Profile GDS file (created for demo purpose)
unlink(fileNameGDS, force=TRUE)
```

`syntheticGeno2` *Generate synthetic profiles for each cancer profile and 1KG reference profile combination and add them to the Profile GDS file*

Description

The functions uses one cancer profile in combination with one 1KG reference profile to generate an synthetic profile that is saved in the Profile GDS file.

When more than one 1KG reference profiles are specified, the function recursively generates synthetic profiles for each cancer profile + 1KG reference profile combination.

The number of synthetic profiles generated by combination is specified by the number of simulation requested.

Usage

```
syntheticGeno2(pRAIDS)
```

Arguments

`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters

Value

The integer `0L` when the function is successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Required library
library(gdsfmt)

## Path to the demo 1KG GDS file is located in this package
dataDir <- system.file("extdata/tests", package="RAIDS")

## Profile GDS file (temporary)
pathProfileGDS <- tempdir()

fileNameGDS <- file.path(pathProfileGDS, "ex1.gds")

## Copy the Profile GDS file demo that has been pruned and annotated
file.copy(file.path(dataDir, "ex1_demo_with_pruning_and_1KG_annot.gds"),
                fileNameGDS)

studyDF <- data.frame(study.id = "MYDATA",
                      study.desc = "Description",
                      study.platform = "PLATFORM",
                      stringsAsFactors = FALSE)
samplePED <- data.frame(Name.ID = c("ex1", "ex2"),
                        Case.ID = c("Patient_h11", "Patient_h12"),
                        Diagnosis = rep("Cancer", 2),
                        Sample.Type = rep("Primary Tumor", 2),
                        Source = rep("Databank B", 2), stringsAsFactors = FALSE)
rownames(samplePED) <- samplePED$Name.ID

pRAIDS <- RAIDSparam(profileFile=file.path(dataDir, "ex1.txt.gz"),
                     genoSource="snp-pileup",
                     studyDF=studyDF,
                     pedStudy=samplePED,
                     prefix="synthTest",
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=file.path(dataDir,
                                                "ex1_good_small_1KG.gds"),
                     fileReferenceAnnotGDS=file.path(dataDir,
                                                     "ex1_good_small_1KG_Annot.gds")
)

syntheticRefDF(pRAIDS) <- selectForSynthetic(pRAIDS = pRAIDS,
                           nbProfiles=30L)

## Generate the synthetic profiles and add them into the Profile GDS
syntheticGeno2(pRAIDS)
```

```
profileGDS <- openfn.gds(fileNameGDS)

tail(read.gdsn(index.gdsn(profileGDS, "sample.id")))

## Close GDS files (important)
closefn.gds(profileGDS)

## Remove Profile GDS file (created for demo purpose)
unlink(fileNameGDS, force=TRUE)
```

`syntheticRefDF` *Generic function for getting the syntheticRefDF slot in a class*

Description

A generic function for getting the syntheticRefDF slot in a S4 object.

Usage

```
syntheticRefDF(x)
```

Arguments

x	a S4 object.

Value

a value from the syntheticRefDF slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(syntheticRefDF="character"))

# Create an instance
obj <- new("MyClass", syntheticRefDF="123")

# Call the generic (assuming a method is implemented)
# syntheticRefDF(obj)
```

syntheticRefDF,RAIDSparam-method *A getter for the syntheticRefDF slot in a RAIDSparam class*

Description

A function for getting the syntheticRefDF slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
syntheticRefDF(x)
```

Arguments

x	a RAIDSparam object.

Value

NULL or a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the syntheticRefDF slot for the object
syntheticRefDF(paramDemo)
```

syntheticRefDF<- *Generic function for replacement of syntheticRefDF slot in a class*

Description

A generic function for replacement of syntheticRefDF slot in a S4 object.

Usage

```
syntheticRefDF(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(syntheticRefDF="character"))

# Create an instance
obj <- new("MyClass", syntheticRefDF="123")

# Call the generic (assuming a method is implemented)
# syntheticRefDF(obj) <- "333"
```

---

syntheticRefDF<-,RAIDSparam-method

Setter function for replacement of syntheticRefDF slot in a RAIDSparam class

---

Description

A function for replacement of syntheticRefDF slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
syntheticRefDF(x) <- value
```

Arguments

x	a RAIDSparam object.
value	NULL or a data.frame containing a subset of reference profiles for each sub-population present in the Reference GDS file. The data.frame must have those columns: **sample.id** a character string representing the sample identifier. **pop.group** a character string representing the subcontinental population assigned to the sample. **superPop** a character string representing the super-population assigned to the sample.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new values to the syntheticRefDF slot in the object
syntheticRefDF(paramDemo) <- data.frame(sample.id=c("A", "B"),
    pop.group=c("CEU", "CEU"), superPop=c("EUR", "EUR"))
```

---

thresholdLD *Generic function for getting the thresholdLD slot in a class*

Description

A generic function for getting the thresholdLD slot in a S4 object.

Usage

```
thresholdLD(x)
```

Arguments

x	a S4 object.

Value

a value from the thresholdLD slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(thresholdLD="character"))

# Create an instance
obj <- new("MyClass", thresholdLD="123")

# Call the generic (assuming a method is implemented)
```

```
# thresholdLD(obj)
```

---

thresholdLD,RAIDSparam-method
A getter for the thresholdLD slot in a RAIDSparam class

Description

A function for getting the thresholdLD slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
thresholdLD(x)
```

Arguments

x	a RAIDSparam object.

Value

a single positive numeric value that represents the LD threshold used in the snpgdsLDpruning function.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the thresholdLD slot for the object
thresholdLD(paramDemo)
```

thresholdLD<- *Generic function for replacement of thresholdLD slot in a class*

Description

A generic function for replacement of thresholdLD slot in a S4 object.

Usage

```
thresholdLD(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(thresholdLD="character"))

# Create an instance
obj <- new("MyClass", thresholdLD="123")

# Call the generic (assuming a method is implemented)
# thresholdLD(obj) <- "333"
```

thresholdLD<-,RAIDSparam-method *Setter function for replacement of thresholdLD slot in a RAIDSparam class*

Description

A function for replacement of thresholdLD slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
thresholdLD(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single positive numeric value that represents the LD threshold used in the snpgdsLDpruning function.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the thresholdLD slot in the object
thresholdLD(paramDemo) <- 1000L
```

---

verbose *Generic function for getting the verbose slot in a class*

Description

A generic function for getting the verbose slot in a S4 object.

Usage

```
verbose(x)
```

Arguments

x	a S4 object.

Value

a value from the verbose slot in the S4 object.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(verbose="character"))

# Create an instance
obj <- new("MyClass", verbose="123")

# Call the generic (assuming a method is implemented)
# verbose(obj)
```

---

verbose,RAIDSparam-method

A getter for the verbose slot in a RAIDSparam class

Description

A function for getting the verbose slot in a RAIDSparam class.

Usage

```
## S4 method for signature 'RAIDSparam'
verbose(x)
```

Arguments

x	a RAIDSparam object.

Value

a single logical indicating if message information should be printed.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Extract the verbose slot for the object
verbose(paramDemo)
```

verbose<- *Generic function for replacement of verbose slot in a class*

Description

A generic function for replacement of verbose slot in a S4 object.

Usage

```
verbose(x) <- value
```

Arguments

x	a S4 object.
value	the new value to assign or update.

Value

the modified S4 object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
# Define a dummy class to show usage
setClass("MyClass", slots = list(verbose="character"))

# Create an instance
obj <- new("MyClass", verbose="123")

# Call the generic (assuming a method is implemented)
# verbose(obj) <- "333"
```

verbose<-,RAIDSparam-method
Setter function for replacement of verbose slot in a RAIDSparam class

Description

A function for replacement of verbose slot in a RAIDSparam class.

Usage

```
## S4 replacement method for signature 'RAIDSparam'
verbose(x) <- value
```

Arguments

x	a RAIDSparam object.
value	a single logical indicating if message information should be printed.

Value

the modified RAIDSparam object when the new value is valid.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Create a RAIDSparam object
paramDemo <- RAIDSparam()

## Assign new value to the verbose slot in the object
verbose(paramDemo) <-  TRUE
```

---

writeBedProfile *Generate a plink bed file TODO*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
writeBedProfile(
  fileOut,
  listProfile,
  listRM = NULL,
  profileOnly = FALSE,
  pRAIDS
)
```

Arguments

fileOut	a character string representing the path and the file name of the new bim file.
listProfile	a vector of character representing the list of profile from gdsProfile to keep
listRM	a list TODO. Default: NULL.
profileOnly	a boolean TODO. Default: FALSE.
pRAIDS	a RAIDSparam an object with all the RAIDS parameters

Value

The integer 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

pathOut <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathOut)){
    dir.create(pathOut)
}

# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                      pedStudy=pedStudy,
                      pathProfileGDS=pathProfileGDS,
                      fileReferenceGDS=fileReferenceGDS,
                      fieldPopInfAnc="SuperPop")
```

```
writeBedProfile(fileOut=file.path(pathOut,
    paste0(pedStudy(pRAIDS)$Name.ID[1], ".bed")),
    listProfile=pedStudy(pRAIDS)$Name.ID[1],
    profileOnly = TRUE, pRAIDS=pRAIDS)

dir(file.path(pathOut))

unlink(file.path(pathOut, paste0(pedStudy(pRAIDS)$Name.ID[1], ".bed")))
unlink(file.path(pathOut), recursive=TRUE)
```

`writeBimPruned` *Generate a plink bim file*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
writeBimPruned(fileOut, pRAIDS)
```

Arguments

`fileOut`	a `character` string representing the path and the file name of the new bim file.
`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters

Value

The integer `0L` when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

pathOut <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathOut)){
```

```
    dir.create(pathOut)
}

# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
                     pedStudy=pedStudy,
                     pathProfileGDS=pathProfileGDS,
                     fileReferenceGDS=fileReferenceGDS,
                     fieldPopInfAnc="SuperPop")


writeBimPruned(file.path(pathOut, "test.bim"), pRAIDS)

dir(file.path(pathOut))

unlink(file.path(pathOut, "test.bim"))
unlink(file.path(pathOut), recursive=TRUE)
```

`writeFamProfile` *Generate a plink fam file*

Description

This function extract the SNPs that pass a frequency cut-off in at least one super population from a GDS SNP information file and save the retained SNP information into a VCF file.

Usage

```
writeFamProfile(
  fileOut,
  listProfile,
  listRM = NULL,
```

```
  profileOnly = FALSE,
  pRAIDS
)
```

Arguments

fileOut	a character string representing the path and the file name of the new fam file.
listProfile	a vector of character representing the list of profile from gdsProfile to keep.
listRM	a TODO. Default: NULL.
profileOnly	a integer TODO. Default: FALSE.
pRAIDS	a RAIDSparam an object with all the RAIDS parameters.

Value

The integer 0L when successful.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## Load the known ancestry for the demo 1KG reference profiles
data(demoKnownSuperPop1KG)

## Path to the demo Profile GDS file is located in this package
dataDir <- system.file("extdata/demoAncestryCall", package="RAIDS")

pathOut <- file.path(tempdir(), "out.tmp")
if(!dir.exists(pathOut)){
    dir.create(pathOut)
}

# The name of the synthetic study
studyID <- "MYDATA"

studyDF <- data.frame(study.id=studyID,
                              study.desc=studyID,
                              study.platform="NotDef",
                              stringsAsFactors=FALSE)
pathProfileGDS <- file.path(dataDir) # , "ex1.gds"
fileReferenceGDS <- system.file("extdata/tests/ex1_good_small_1KG.gds", package="RAIDS")
pedStudy <- data.frame(Name.ID=c("ex1"),
                              Case.ID=c("ex1"),
                              Sample.Type=c("type"),
                              Diagnosis="NotDef",
                              Source=c("NotDef"),
                              stringsAsFactors=FALSE)
     row.names(pedStudy) <- pedStudy$Name.ID

pRAIDS <- RAIDSparam(studyDF=studyDF,
```

```
                        pedStudy=pedStudy,
                        pathProfileGDS=pathProfileGDS,
                        fileReferenceGDS=fileReferenceGDS,
                        fieldPopInfAnc="SuperPop")


    writeFamProfile(fileOut=file.path(pathOut,
        paste0(pedStudy(pRAIDS)$Name.ID[1], ".fam")),
        listProfile=pedStudy(pRAIDS)$Name.ID[1],
        profileOnly = TRUE, pRAIDS=pRAIDS)

    dir(file.path(pathOut))

    unlink(file.path(pathOut, paste0(pedStudy(pRAIDS)$Name.ID[1], ".fam")))
    unlink(file.path(pathOut), recursive=TRUE)
```

`writePMatrix` *Generate the row for the ADMIXTURE P-matrax*

Description

This function write with P-matrix

Usage

```
writePMatrix(pathOut, rowF, pRAIDS)
```

Arguments

`pathOut`	a `character` representing the output path
`rowF`	a `list` containing allele frequencies for each population and and for a subset if intersect quasi-no admix subset intersect the synthetic
`pRAIDS`	a `RAIDSparam` an object with all the RAIDS parameters.

Value

a `array` of the frequency matrix.

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Examples

```
## see computePmatrix
```

Index

∗ **classes**
RAIDSparam-class, 230
∗ **datasets**
demoKnownSuperPop1KG, 74
demoPCA1KG, 75
demoPCASyntheticProfiles, 77
demoPedigreeEx1, 78
matKNNSynthetic, 152
pedSynthetic, 187
snpPositionDemo, 267

add1KG2SampleGDS, 8
addBlockFromDetFile, 9
addGeneBlockGDSRefAnnot, 11
addGeneBlockRefAnnot, 13
addGenotypeProfile, 15
addRef2GDS1KG, 16
addStudy1Kg, 18
ancestryInferencePCAKNN, 20
anchorId, 23
anchorId,RAIDSparam-method, 24
anchorId<-, 25
anchorId<-,RAIDSparam-method, 26

batch, 26
batch,RAIDSparam-method, 27
batch<-, 28
batch<-,RAIDSparam-method, 29
blockTypeId, 29
blockTypeId,RAIDSparam-method, 30
blockTypeId<-, 31
blockTypeId<-,RAIDSparam-method, 32

calcAFMLRNA, *267*
CharacterOrNULL-class, 32
chrInfo, 33
chrInfo,RAIDSparam-method, 33
chrInfo<-, 34
chrInfo<-,RAIDSparam-method, 35
cleanADMIXTUREFiles, 36
cleanADMIXTUREFiles,RAIDSparam-method, 36
cleanADMIXTUREFiles<-, 37
cleanADMIXTUREFiles<-,RAIDSparam-method, 38
computeAdmixtureProportions, 39
computeAncestryFromSynthetic, 40
computeAncestryFromSynthetic2, 45
computeAncestryFromSyntheticFile, 48
computeKNNRefSample, 53
computeKNNRefSyn, 54
computeKNNRefSynthetic, 56, *74*, *77*
computePCAMultiSynthetic, 58, *76*
computePCARefSample, 60
computePoolSyntheticAncestryGr, 62, *74*
computePoolSyntheticAncestryGr2, 65
computeSyntheticROC, 67, *153*, *188*
createAccuracyGraph, 69
createAUROCGraph, 70
createStudy2GDS1KG, 71

DataFrameOrNULL-class, 73
demoKnownSuperPop1KG, 74
demoPCA1KG, 75
demoPCASyntheticProfiles, 77
demoPedigreeEx1, 78

eigenCount, 81
eigenCount,RAIDSparam-method, 82
eigenCount<-, 82
eigenCount<-,RAIDSparam-method, 83
eigenCountSyn, 84
eigenCountSyn,RAIDSparam-method, 85
eigenCountSyn<-, 85
eigenCountSyn<-,RAIDSparam-method, 86
estimateAllelicFraction, 87

fieldPopInfAnc, 90
fieldPopInfAnc,RAIDSparam-method, 91
fieldPopInfAnc<-, 91

fieldPopInfAnc<-,RAIDSparam-method, 92
fieldPopInRef, 93
fieldPopInRef,RAIDSparam-method, 94
fieldPopInRef<-, 94
fieldPopInRef<-,RAIDSparam-method, 95
fieldSubPop, 96
fieldSubPop,RAIDSparam-method, 97
fieldSubPop<-, 97
fieldSubPop<-,RAIDSparam-method, 98
fileReferenceAnnotGDS, 99
fileReferenceAnnotGDS,RAIDSparam-method, 100
fileReferenceAnnotGDS<-, 100
fileReferenceAnnotGDS<-,RAIDSparam-method, 101
fileReferenceGDS, 102
fileReferenceGDS,RAIDSparam-method, 103
fileReferenceGDS<-, 103
fileReferenceGDS<-,RAIDSparam-method, 104

gds.class, *8*, *12*, *18*, *41*, *49*, *59*, *61*, *87, 88*, *109*, *122*, *222*, *254*, *269*, *284*
generateGDS1KG, 105
generateMapSnvSel, 107
generatePhase1KG2GDS, 108
generatePhaseRef, 110
generateVCF, 112
genome, 113
genome,RAIDSparam-method, 114
genoSource, 115
genoSource,RAIDSparam-method, 115
genoSource<-, 116
genoSource<-,RAIDSparam-method, 117
genoType, 118
genoType,RAIDSparam-method, 118
genoType<-, 119
genoType<-,RAIDSparam-method, 120
getMatrixPopSynthetic, 121
getRef1KGPop, 122
getRefSuperPop, 123
getRefSuperPop2, 124
groupChr1KGSNV, 125

identifyRelative, 126
identifyRelativeRef, 128
inferAncestry, 129
inferAncestryDNA, 134
inferAncestryGeneAware, 138
inferenceType, 143
inferenceType,RAIDSparam-method, 144
inferenceType<-, 145
inferenceType<-,RAIDSparam-method, 145

kList, 146
kList,RAIDSparam-method, 147
kList<-, 148
kList<-,RAIDSparam-method, 148

listPos, 149
listPos,RAIDSparam-method, 150
listPos<-, 151
listPos<-,RAIDSparam-method, 151

matKNNSynthetic, 152
minCov, 154
minCov,RAIDSparam-method, 155
minCov<-, 155
minCov<-,RAIDSparam-method, 156
minProb, 157
minProb,RAIDSparam-method, 158
minProb<-, 159
minProb<-,RAIDSparam-method, 159

nbSim, 160
nbSim,RAIDSparam-method, 161
nbSim<-, 162
nbSim<-,RAIDSparam-method, 162
np, 163
np,RAIDSparam-method, 164
np<-, 165
np<-,RAIDSparam-method, 165

paramAncestry, 166
paramAncestry,RAIDSparam-method, 167
paramAncestry<-, 168
paramAncestry<-,RAIDSparam-method, 168
pathADMIXTURE, 169
pathADMIXTURE,RAIDSparam-method, 170
pathADMIXTURE<-, 171
pathADMIXTURE<-,RAIDSparam-method, 171
pathProfileGDS, 172
pathProfileGDS,RAIDSparam-method, 173
pathProfileGDS<-, 174
pathProfileGDS<-,RAIDSparam-method, 174
PCAalgorithm, 175

PCAalgorithm,RAIDSparam-method, 176
PCAalgorithm<-, 177
PCAalgorithm<-,RAIDSparam-method, 177
pcaList, 178
pcaList,RAIDSparam-method, 179
pcaList<-, 180
pcaList<-,RAIDSparam-method, 180
PCAmissingRate, 181
PCAmissingRate,RAIDSparam-method, 182
PCAmissingRate<-, 183
PCAmissingRate<-,RAIDSparam-method, 183
pedStudy, 184
pedStudy,RAIDSparam-method, 185
pedStudy<-, 186
pedStudy<-,RAIDSparam-method, 186
pedSynthetic, 187
phase, 189
phase,RAIDSparam-method, 190
phase<-, 191
phase<-,RAIDSparam-method, 191
phaseType, 192
phaseType,RAIDSparam-method, 193
phaseType<-, 194
phaseType<-,RAIDSparam-method, 194
posOffset, 195
posOffset,RAIDSparam-method, 196
posOffset<-, 197
posOffset<-,RAIDSparam-method, 197
pRecomb, 198
pRecomb,RAIDSparam-method, 199
pRecomb<-, 200
pRecomb<-,RAIDSparam-method, 200
prefix, 201
prefix,RAIDSparam-method, 202
prefix<-, 203
prefix<-,RAIDSparam-method, 204
prepPed1KG, 205
prepPedSyntheticRef, 206
prepPMatrix, 208
prepSynthetic, 209
profileFile, 211
profileFile,RAIDSparam-method, 212
profileFile<-, 213
profileFile<-,RAIDSparam-method, 214
profileGenoSynthesis, 215
pruningMethod, 216
pruningMethod,RAIDSparam-method, 217
pruningMethod<-, 218
pruningMethod<-,RAIDSparam-method, 219
pruningProfile2, 220
pruningSample, *88*, 221

RAIDSparam, 224
RAIDSparam-class, 230
reference, 233
reference,RAIDSparam-method, 234
reference<-, 235
reference<-,RAIDSparam-method, 236
runADMIXTURE, 236
runExomeAncestry, *79*, 238
runRNAAncestry, 242
runSynADMIXTURE, 246

sampleRef, 247
sampleRef,RAIDSparam-method, 248
sampleRef<-, 249
sampleRef<-,RAIDSparam-method, 250
seedADMIXTURE, 251
seedADMIXTURE,RAIDSparam-method, 251
seedADMIXTURE<-, 252
seedADMIXTURE<-,RAIDSparam-method, 253
select1KGPop, 254
select1KGPopForSynthetic, 255
selectForSynthetic, 256
seqError, 258
seqError,RAIDSparam-method, 258
seqError<-, 259
seqError<-,RAIDSparam-method, 260
seqErrorSyn, 261
seqErrorSyn,RAIDSparam-method, 261
seqErrorSyn<-, 262
seqErrorSyn<-,RAIDSparam-method, 263
seqlengths, *88*, *229*, *230*
slideWindowMaxBP, 264
slideWindowMaxBP,RAIDSparam-method, 264
slideWindowMaxBP<-, 265
slideWindowMaxBP<-,RAIDSparam-method, 266
snpgdsLDpruning, *218–222*, *228*, *232*, *265*, *266*, *292*, *294*
snpgdsPCA, *42*, *50*, *61*, *63*, *75*, *76*, *82*, *83*, *176*, *178*, *182*, *184*, *228*, *232*
snpPositionDemo, 267
SNPRelate::SNPGDSFileClass, *57*, *63*, *126*
snvListVCF, 269

snvListVCFRef, 270
specificSNV, 271
specificSNV,RAIDSparam-method, 271
specificSNV<-, 272
specificSNV<-,RAIDSparam-method, 273
splitSelectByPop, 274
studyDF, 275
studyDF,RAIDSparam-method, 275
studyDF<-, 276
studyDF<-,RAIDSparam-method, 277
studyDFSyn, 278
studyDFSyn,RAIDSparam-method, 278
studyDFSyn<-, 279
studyDFSyn<-,RAIDSparam-method, 280
studyType, 281
studyType,RAIDSparam-method, 282
studyType<-, 282
studyType<-,RAIDSparam-method, 283
syntheticGeno, 284
syntheticGeno2, 286
syntheticRefDF, 288
syntheticRefDF,RAIDSparam-method, 289
syntheticRefDF<-, 289
syntheticRefDF<-,RAIDSparam-method, 290

tableBlockAF, *267*
thresholdLD, 291
thresholdLD,RAIDSparam-method, 292
thresholdLD<-, 293
thresholdLD<-,RAIDSparam-method, 293

verbose, 294
verbose,RAIDSparam-method, 295
verbose<-, 296
verbose<-,RAIDSparam-method, 296

writeBedProfile, 297
writeBimPruned, 299
writeFamProfile, 300
writePMatrix, 302