Package: RAIDS (via r-universe)

September 28, 2024

Type Package

Title Accurate Inference of Genetic Ancestry from Cancer Sequences

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 1.3.0

License Apache License (>= 2)

Encoding UTF-8

NeedsCompilation no

VignetteBuilder knitr

Depends R (>= 4.2.0), gdsfmt, SNPRelate, stats, utils, GENESIS

Imports S4Vectors, GenomicRanges, ensembldb, BSgenome, AnnotationDbi, methods, class, pROC, IRanges, AnnotationFilter, rlang, VariantAnnotation, MatrixGenerics,

Suggests testthat, knitr, rmarkdown, BiocStyle, withr, GenomeInfoDb, BSgenome.Hsapiens.UCSC.hg38, EnsDb.Hsapiens.v86

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

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

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

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://bioc.r-universe.dev

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

RemoteRef HEAD

RemoteSha 8c31b43b8656bfbf888c0c9627083eaca70cf7eb

Contents

`RAIDS-package` *RAIDS: Accurate Inference of Genetic Ancestry from Cancer Sequences*

Description

The RAIDS package implements specialized algorithms that enable ancestry inference from various cancer data sources (RNA, Exome and Whole-Genome sequencing).

Details

The RAIDS package also implements simulation algorithm that generates synthetic cancer-derived data.

This code and analysis pipeline was designed and developed for the following publication:

Pascal Belleau, Astrid Deschênes, Nyasha Chambwe, David A. Tuveson, Alexander Krasnitz; Genetic Ancestry Inference from Cancer-Derived Molecular Data across Genomic and Transcriptomic Platforms. Cancer Res 1 January 2023; 83 (1): 49–58. https://doi.org/10.1158/0008-5472.CAN-22-0682

Value

RAIDS

Author(s)

Pascal Belleau, Astrid Deschênes and Alexander Krasnitz

Maintainer: Pascal Belleau pascal_belleau@hotmail.com

References

Pascal Belleau, Astrid Deschênes, Nyasha Chambwe, David A. Tuveson, Alexander Krasnitz; Genetic Ancestry Inference from Cancer-Derived Molecular Data across Genomic and Transcriptomic Platforms. Cancer Res 1 January 2023; 83 (1): 49–58. https://doi.org/10.1158/0008-5472.CAN-22-0682

See Also

- `runExomeAncestry` This function runs most steps leading to the ancestry inference call on a specific exome profile.
- `runExomeAncestry` This function runs most steps leading to the ancestry inference call on a specific RNA profile.
- `estimateAllelicFraction` This function estimates the allelic fraction of the pruned SNVs for a specific sample and add the information to the associated GDS Sample file. The allelic fraction estimation method is adapted to the type of study (DNA or RNA).
- `computeSyntheticROC` This function calculate the AUROC of the inferences for specific values of D and K using the inferred ancestry results from the synthetic profiles.

- generateMapSnvSel The function applies a cut-off filter to the SNV information file to retain only the SNV that have a frequency superior or equal to the specified cut-off in at least one super population.

---

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)
```

`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)

}
```

---

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)
```

`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)
```

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.ida vector of character strings representing the identifiers of the synthetic profiles.
- sample1Kga vector of character strings representing the identifiers of the reference 1KG profiles used to generate the synthetic profiles.
- spa vector of character strings representing the known ancestry for the reference 1KG profiles used to generate the synthetic profiles.
- matKNNa 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)
```

`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 superpopulation 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
```

`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 OL 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)
```

`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("GenomeInfoDb", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- GenomeInfoDb::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)

}
```

---

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("GenomeInfoDb", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {
    chrInfo <- GenomeInfoDb::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("GenomeInfoDb", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- GenomeInfoDb::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)

}
```

`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,
  fileSNPsRDS,
  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".
fileSNPsRDS	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, fileSNPsRDS=filterSNVFile,
            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)
```

`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)
```

---

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)
}
```

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
```

`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
```

`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.ida `character` string representing the profile unique ID.
- Name.IDa `character` string representing the profile name.
- sexa `character` string representing the sex of the profile.
- pop.groupa `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)
```

---

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)
```

---

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)
```

`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 InferenceThe ancestry inference CSV file (".Ancestry.csv" file)
- Inference InformatonThe inference information RDS file (".infoCall.rds" file)
- Synthetic InformationThe 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("GenomeInfoDb", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- GenomeInfoDb::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 InferenceThe ancestry inference CSV file (".Ancestry.csv" file)
- Inference InformatonThe inference information RDS file (".infoCall.rds" file)
- Synthetic InformationThe 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("GenomeInfoDb", quietly=TRUE) &&
     requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly=TRUE)) {

    ## Chromosome length information
    ## chr23 is chrX, chr24 is chrY and chrM is 25
    chrInfo <- GenomeInfoDb::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)


}
```

`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)
```

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)
```

---

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)
```

---

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)
```

Index