Package 'detectRUNS'

October 24, 2019

Type Package

Title Detect Runs of Homozygosity and Runs of Heterozygosity in Diploid Genomes

Version 0.9.6

Date 2019-10-24

Description Detection of runs of homozygosity and of heterozygosity in diploid genomes using two methods: sliding windows (Purcell et al (2007) <doi:10.1086/519795>) and consecutive runs (Marras et al (2015) <doi:10.1111/age.12259>).

Depends R (>= 3.0.0)

License GPL-3

LazyData TRUE

Encoding UTF-8

Imports plyr, iterators, itertools, ggplot2, reshape2, Rcpp, gridExtra, data.table

RoxygenNote 6.1.1

Suggests testthat, knitr, rmarkdown, prettydoc

LinkingTo Rcpp

VignetteBuilder knitr

URL https://github.com/bioinformatics-ptp/detectRUNS/tree/master/detectRUNS

BugReports https://github.com/bioinformatics-ptp/detectRUNS/issues

NeedsCompilation yes

Author Filippo Biscarini [aut, cre],
Paolo Cozzi [aut],
Giustino Gaspa [aut],
Gabriele Marras [aut]

Maintainer Filippo Biscarini <filippo.biscarini@gmail.com>

Repository CRAN

Date/Publication 2019-10-24 13:40:02 UTC

R topics documented:

`consecutiveRUNS.run` *Main function to detect genomic RUNS (ROHom/ROHet) using the consecutive method*

Description

This is the main detectRUNS function to scan the genome for runs (of homozygosity or heterozygosity) using the consecutive method (Marras et al. 2015, Animal Genetics 46(2):110-121). All parameters to detect runs (e.g. minimum n. of SNP, max n. of missing genotypes, max n. of opposite genotypes etc.) are specified here. Input data are in the ped/map Plink format (https://www.cog-genomics.org/plink/1.9/input#ped)

Usage

```
consecutiveRUNS.run(genotypeFile, mapFile, ROHet = FALSE,
  maxOppRun = 0, maxMissRun = 0, minSNP = 15, minLengthBps = 1000,
  maxGap = 10^6)
```

Arguments

genotypeFile	genotype (.ped) file path
mapFile	map file (.map) file path
ROHet	should we look for ROHet or ROHom? (default = FALSE)
maxOppRun	max n. of opposite genotype SNPs in the run (default = 0)
maxMissRun	max n. of missing SNPs in the run (default = 0)
minSNP	minimum n. of SNP in a RUN (default = 15)
minLengthBps	minimum length of run in bps (defaults to 1000 bps = 1 kbps)
maxGap	max distance between consecutive SNP in a window to be still considered a potential run (defaults to 10^6)

Details

This function scans the genome (diploid) for runs using the consecutive method. This is a wrapper function for many component functions that handle the input data (ped/map files), performs internal conversions, accepts parameters specifications, selects the statistical method to detect runs (sliding windows, consecutive loci) and whether runs of homozygosity (RoHom) or of heterozygosity (RoHet) are looked for.

In the ped file, the groups samples belong to can be specified (first column). This is important if comparisons between human ethnic groups or between animal breeds or plant varieties or biological populations are to be performed. Also, if cases and controls are to be compared, this is the place where this information needs to be specified.

This function returns a data frame with all runs detected in the dataset. This data frame can then be written out to a csv file. The data frame is, in turn, the input for other functions of the detectRUNS package that create plots and produce statistics of the results (see plot and statistic functions in this manual, and/or refer to the vignette of detectRUNS).

Value

A dataframe with RUNs of Homozygosity or Heterozygosity in the analysed dataset. The returned dataframe contains the following seven columns: "group", "id", "chrom", "nSNP", "from", "to", "lengthBps" (group: population, breed, case/control etc.; id: individual identifier; chrom: chromosome on which the run is located; nSNP: number of SNPs in the run; from: starting position of the run, in bps; to: end position of the run, in bps; lengthBps: size of the run)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")
```

```
# calculating runs with consecutive run approach
## Not run:
# skipping runs calculation
runs <- consecutiveRUNS.run(genotypeFile, mapFile, minSNP = 15, ROHet = FALSE,
maxOppRun = 0, maxMissRun = 0, maxGap=10^6,
minLengthBps = 100000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.consecutive.csv", package="detectRUNS")
colClasses <- c(rep("character", 3), rep("numeric", 4)  )
runs <- read.csv2(runsFile, header = TRUE, stringsAsFactors = FALSE,
colClasses = colClasses)
```

`consecutiveRunsCpp` *Function to detect consecutive runs in a vector (individual's genotypes)*

Description

This is a core function. It implements the consecutive method for detection of runs in diploid genomes (see Marras et al. 2015)

Usage

```
consecutiveRunsCpp(indGeno, individual, mapFile, ROHet = TRUE,
  minSNP = 3L, maxOppositeGenotype = 1L, maxMiss = 1L,
  minLengthBps = 1000L, maxGap = 1000000L)
```

Arguments

`indGeno`	vector of 0/1/NAs of individual genotypes (0: homozygote; 1: heterozygote)
`individual`	list of group (breed, population, case/control etc.) and ID of individual sample
`mapFile`	Plink map file (for SNP position)
`ROHet`	shall we detect ROHet or ROHom?
`minSNP`	minimum number of SNP in a run
`maxOppositeGenotype`	max n. of homozygous/heterozygous SNP
`maxMiss`	max. n. of missing SNP
`minLengthBps`	min length of a run in bps
`maxGap`	max distance between consecutive SNP in a window to be still considered a potential run

Details

The consecutive method detect runs by consecutively scanning SNP loci along the genome. No sliding windows are used. Checks on minimum n. of SNP, max n. of opposite and missing genotypes, max gap between adjacent loci and minimum length of the run are implemented (as in the sliding window method). Both runs of homozygosity (RoHom) and of heterozygosity (RoHet) can be search for (option ROHet: TRUE/FALSE)

Value

A data frame of runs per individual sample

`createRUNdf` *Function to create a dataframe of RUNS per individual animal Requires a map file (other filename to read or R object) Parameters on maximum number of missing and opposite genotypes in the run (not the window) are implemented here*

Description

Function to create a dataframe of RUNS per individual animal Requires a map file (other filename to read or R object) Parameters on maximum number of missing and opposite genotypes in the run (not the window) are implemented here

Usage

```
createRUNdf(snpRun, mapFile, minSNP = 3, minLengthBps = 1000,
  minDensity = 1/10, oppositeAndMissingSNP, maxOppRun = NULL,
  maxMissRun = NULL)
```

Arguments

`snpRun`	vector of TRUE/FALSE (is the SNP in a RUN?)
`mapFile`	Plink-like map file (data.frame)
`minSNP`	minimum n. of SNP to call a RUN
`minLengthBps`	minimum length of run in bps (defaults to 1000 bps = 1 kbps)
`minDensity`	minimum n. of SNP per kbps (defaults to 0.1 = 1 SNP every 10 kbps)
`oppositeAndMissingSNP`	indexed array of missing and opposite genotypes (SNP order in the genome is the index)
`maxOppRun`	max n. of opposite genotype SNPs in the run (not in the window!)
`maxMissRun`	max n. of missing SNPs in the run (not in the window!)

Value

a data.frame with RUNS per animal

findOppositeAndMissing *Function to calculate oppositeAndMissingGenotypes array*

Description

This is an helper function, this will be called by another function

Usage

```
findOppositeAndMissing(data, ROHet = TRUE)
```

Arguments

data	vector of 0/1/2 genotypes
ROHet	TRUE in ROHet evaluation, FALSE for ROHom

Value

character array; names will be index in which opposite and missing snps are found in data array

Froh_inbreeding *Function to calculated Froh genome-wide or chromosome-wide*

Description

This function calculates the individual inbreeding coefficients based on runs of homozygosity (ROH), either per-chromosome (chromosome-wide) or based on the entire genome (genome-wide). See details of calculations below

Usage

```
Froh_inbreeding(runs, mapFile, genome_wide = TRUE)
```

Arguments

runs	R object (dataframe) with results on runs
mapFile	Plink map file (to retrieve SNP position)
genome_wide	vector of TRUE/FALSE (genome-wide or chromosome-wide; defaults to TRUE/genome-wide)

Details

Froh is calculated as:

$F_{ROH} = \frac{\sum ROH_{length}}{Length_{genome}}$

Depending on whether genome-wide or chromosome-wide calculations are required, the terms in the numerator and denominator will refer to the entire genome or will be restricted to specific chromosomes.

Value

A data frame with the inbreeding coefficients of each individual sample

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1,  minSNP = 15,
ROHet = FALSE,  maxOppositeGenotype = 1, maxMiss = 1,  minLengthBps = 100000,  minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

Froh_inbreeding(runs = runsDF, mapFile = mapFile)
Froh_inbreeding(runs = runsDF, mapFile = mapFile, genome_wide=FALSE)
```

Froh_inbreedingClass *Function to calculated Froh using a ROH-class*

Description

This function calculates the individual inbreeding coefficients based on runs of homozygosity (ROH) using only ROH of specific size classes. The parameter `class` specify the size interval to split up calculations. For example, if `class = 2` Froh based on ROH 0-2, 2-4, 4-8, 80-16, >16 Mbps long will be calculated.

Usage

```
Froh_inbreedingClass(runs, mapFile, Class = 2)
```

Arguments

runs	R object (dataframe) with ROH results
mapFile	Plink map file (for SNP position)
Class	base ROH-length interval (in Mbps) (default: 0-2, 2-4, 4-8, 8-16, >16)

Value

A data frame with individual inbreeding coefficients based on ROH-length of specific size. The sum of ROH-length of specific size in each individual is reported alongside

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

Froh_inbreedingClass(runs = runsDF, mapFile = mapFile, Class = 2)
```

---

genoConvertCpp *Convert 0/1/2 genotypes to 0/1*

Description

This is a utility function, that convert 0/1/2 genotypes (AA/AB/BB) into 0/1 (either homozygous/heterozygous)

Usage

```
genoConvertCpp(genotype)
```

Arguments

genotype	vector of 0/1/2 genotypes

Value

converted vector of genotypes (0/1)

`heteroZygotTest` *Function to check whether a window is (loosely) heterozygous or not*

Description

This is a core function within the sliding-window workflow. Parameters on how to consider a window heterozygous are here (maxHom, maxMiss)

Usage

```
heteroZygotTest(x, gaps, maxHom, maxMiss, maxGap, i, windowSize)
```

Arguments

`x`	vector of 0/1 genotypes (from genoConvert())
`gaps`	vector of differences between consecutive positions (gaps) in bps
`maxHom`	max n. of homozygous SNP in a heterozygous window
`maxMiss`	max n. of missing in a window
`maxGap`	max distance between consecutive SNP in a window to be still considered a potential run
`i`	index along the genome (genome-vector for each individual)
`windowSize`	size of window (n. of SNP)

Value

a list: i) TRUE/FALSE (whether a window is heterozygous or NOT); ii) indexes of "opposite and missing" genotype

`heteroZygotTestCpp` *Function to check whether a window is (loosely) heterozygous or not*

Description

This is a core function. Parameters on how to consider a window heterozygous are here (maxHom, maxMiss)

Usage

```
heteroZygotTestCpp(x, gaps, maxHom, maxMiss, maxGap)
```

Arguments

x	vector of 0/1 genotypes (from genoConvert())
gaps	vector of differences between consecutive positions (gaps) in bps
maxHom	max n. of homozygous SNP in a heterozygous window
maxMiss	max n. of missing in a window
maxGap	max distance between consecutive SNP in a window to be still considered a potential run

Value

TRUE/FALSE (whether a window is heterozygous or NOT)

---

homoZygotTest *Function to check whether a window is (loosely) homozygous or not*

Description

This is a core function. Parameters on how to consider a window homozygous are here (maxHet, maxMiss)

Usage

```
homoZygotTest(x, gaps, maxHet, maxMiss, maxGap, i, windowSize)
```

Arguments

x	vector of 0/1 genotypes (from genoConvert())
gaps	vector of differences between consecutive positions (gaps) in bps
maxHet	max n. of heterozygous SNP in a homozygous window
maxMiss	max n. of missing in a window
maxGap	max distance between consecutive SNP in a window to be still considered a potential run
i	index along the genome (genome-vector for each individual)
windowSize	size of window (n. of SNP)

Value

a list: i) TRUE/FALSE (whether a window is heterozygous or NOT); ii) indexes of "opposite and missing" genotype

homoZygotTestCpp *Function to check whether a window is (loosely) homozygous or not*

Description

This is a core function. Parameters on how to consider a window homozygous are here (maxHet, maxMiss)

Usage

```
homoZygotTestCpp(x, gaps, maxHet, maxMiss, maxGap)
```

Arguments

x	vector of 0/1 genotypes (from genoConvert())
gaps	vector of differences between consecutive positions (gaps) in bps
maxHet	max n. of heterozygous SNP in a homozygous window
maxMiss	max n. of missing in a window
maxGap	max distance between consecutive SNP in a window to be still considered a potential run

Value

TRUE/FALSE (whether a window is homozygous or NOT)

pedConvertCpp *Convert ped genotypes to 0/1*

Description

This is a utility function, that convert ped genotypes (AA/AB/BB) into 0/1 (either homozygous/heterozygous)

Usage

```
pedConvertCpp(genotype)
```

Arguments

genotype	vector of pair of genotypes (01, AA, AG)

Value

converted vector of genotypes (0/1)

plot_DistributionRuns *Plot Distribution of runs*

Description

This function the distribution of runs per group. The average run length per size-class, the average run length per chromosome (and group), the percent distribution of runs per size-class and group, and the proportion of runs per chromosome are plotted. With `style="All"` all three plots are produced.

Usage

```
plot_DistributionRuns(runs, mapFile, groupSplit = TRUE,
  style = c("MeanClass", "MeanChr", "RunsPCT", "RunsPCT_Chr", "All"),
  savePlots = FALSE, outputName = NULL, plotTitle = NULL,
  Class = 2)
```

Arguments

runs	R object (dataframe) with results on detected runs
mapFile	Plink map file (for SNP position)
groupSplit	plots split by group (defaults to TRUE)
style	type of plot: MeanClass, MeanChr, RunsPCT, RunsPCT_Chr, All (all plots)
savePlots	should plots be saved out to files or plotted in the graphical terminal (default)?
outputName	title prefix (the base name of graph, if savePlots is TRUE)#'
plotTitle	title in plot (default NULL)
Class	group of length (in Mbps) by class (default: 0-2, 2-4, 4-8, 8-16, >16)

Value

plot Distribution Runs

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1,  minSNP = 15,
ROHet = FALSE,  maxOppositeGenotype = 1, maxMiss = 1,  minLengthBps = 100000,  minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
```

```
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

plot_InbreedingChr(runs = runsDF, mapFile = mapFile, style='All')
```

`plot_InbreedingChr` *Plot Froh-based inbreeding coefficients by group*

Description

The function plots the distribution of inbreeding/consanguinity coefficients per chromosome and/or group. Three types of plots can be produces: barplots, boxplots, violin plots. With `style="All"` all three plots are produced.

Usage

```
plot_InbreedingChr(runs, mapFile, groupSplit = TRUE,
  style = c("ChrBarPlot", "ChrBoxPlot", "FrohBoxPlot", "All"),
  outputName = NULL, plotTitle = NULL, savePlots = FALSE)
```

Arguments

`runs`	R object (dataframe) with results on detected runs
`mapFile`	Plink map file (for SNP position)
`groupSplit`	plots split by group (defaults to TRUE)
`style`	type of plot: ChrBarPlot, ChrBoxPlot, FrohBoxPlot, All (all plots)
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)
`plotTitle`	title in plot (default NULL)
`savePlots`	should plots be saved out to files or plotted in the graphical terminal (default)?

Value

plots of the distribution of inbreeding by chromosome and group

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)
```

```
## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

plot_InbreedingChr(runs = runsDF, mapFile = mapFile, style='All')
```

`plot_manhattanRuns` *Plot the proportion of times SNPs are inside runs - MANHATTAN PLOT*

Description

Function to plot the proportion of times/percentage each SNP in inside a run (population-specific signals) against SNP position in all chromosomes together Proportions on the y-axis, bps on the x-axis for all analysed chromosomes This is similar to the familiar GWAS Manhattan plot

Usage

```
plot_manhattanRuns(runs, genotypeFile, mapFile, savePlots = FALSE,
  outputName = NULL, plotTitle = NULL)
```

Arguments

`runs`	a data.frame with runs per individual (group, id, chrom, nSNP, start, end, length)
`genotypeFile`	genotype (.ped) file path
`mapFile`	map file (.map) file path
`savePlots`	should plots be saved out in files (default) or plotted in the graphical terminal?
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)
`plotTitle`	title in plot (default)

Value

Manhattan plots of proportion of times SNPs are inside runs, per population (pdf files)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)
```

```
## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

# plot runs per animal (interactive)
plot_manhattanRuns(runs = runsDF, genotypeFile = genotypeFile, mapFile = mapFile,
savePlots = FALSE, plotTitle = "ROHom")
```

`plot_PatternRuns` *Plot sum of run-lengths (or average run-lengths) against the number of runs per individual*

Description

Function to plot the sum of run lengths (or the average run length) per individual against the average number of runs per individual. Points can be differentially coloured by group/population. This plot can be useful to identify patterns in the distribution of runs in different groups (e.g. few long runs vs many short runs)

Usage

```
plot_PatternRuns(runs, mapFile, method = c("sum", "mean"),
  outputName = NULL, savePlots = FALSE, plotTitle = NULL)
```

Arguments

`runs`	a data.frame with runs per individual (group, id, chrom, nSNP, start, end, length)
`mapFile`	map file (.map) file path
`method`	"sum" or "mean" of run lengths per individual sample
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)#'
`savePlots`	should plots be saved out to files or plotted in the graphical terminal (default)?
`plotTitle`	title in plot (default NULL)

Value

plot of number of runs vs run-length sum/mean per individual sample

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
```

```
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

plot_PatternRuns(runs = runsDF, mapFile = mapFile, method = 'sum')
plot_PatternRuns(runs = runsDF, mapFile = mapFile, method = 'mean')
```

`plot_Runs` *Function to plot runs per individual*

Description

Function to plot runs per individual (see Williams et al. 2016, Animal Genetics, for an example with animal data) Individual IDs on the y-axis, bps on the x-axis (position along the chromosome)

Usage

```
plot_Runs(runs, suppressInds = FALSE, savePlots = FALSE,
  separatePlots = FALSE, outputName = NULL)
```

Arguments

`runs`	a data.frame with runs per individual (group, id, chrom, nSNP, start, end, length)
`suppressInds`	shall we suppress individual IDs on the y-axis? (defaults to FALSE)
`savePlots`	should plots be saved out to files (one pdf file for all chromosomes) or plotted in the graphical terminal (default)?
`separatePlots`	should plots for each chromosome be saved out to separate files?
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)

Value

plot of runs by chromosome

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
```

```
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

# plot runs per animal (interactive)
plot_Runs(runs = runsDF, suppressInds = FALSE, savePlots = FALSE, outputName = "ROHom")
```

`plot_SnpsInRuns` *Plot the number of times each SNP falls inside runs*

Description

Function to plot the number of times/percentage each SNP is inside a run (population-specific signals) against the SNP positions in the genome. Proportions on the y-axis, bps on the x-axis

Usage

```
plot_SnpsInRuns(runs, genotypeFile, mapFile, savePlots = FALSE,
  separatePlots = FALSE, outputName = NULL)
```

Arguments

`runs`	a data.frame with runs per individual (group, id, chrom, nSNP, start, end, length)
`genotypeFile`	genotype (.ped) file path
`mapFile`	map file (.map) file path
`savePlots`	should plots be saved out in files (default) or plotted in the graphical terminal?
`separatePlots`	should plots for each chromosome be saved out to separate files?
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)

Value

plot number of times a SNP is in a run by chromosome and population (pdf files)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
# skipping runs calculation
## Not run:
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
```

```
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

# plot runs per animal (interactive)
plot_SnpsInRuns(runs = runsDF, genotypeFile = genotypeFile, mapFile = mapFile,
savePlots = FALSE, outputName = "ROHom")
```

`plot_StackedRuns` *Plot stacked runs*

Description

Function to plot stacked runs along the chromosome (signaling presence of large numbers of runs in specific regions of a chromosome) Counts on the y-axis, bps on the x-axis (position along the chromosome)

Usage

```
plot_StackedRuns(runs, savePlots = FALSE, separatePlots = FALSE,
  outputName = NULL)
```

Arguments

`runs`	a data.frame with runs per individual (group, id, chrom, nSNP, start, end, length)
`savePlots`	should plots be saved out in files (default) or plotted in the graphical terminal?
`separatePlots`	should plots for chromosomes be saved out to separate files?
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)

Value

plot of stacked runs by population and by chromosome (pdf files)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)
```

```
## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

# plot runs per animal (interactive)
plot_StackedRuns(runs = runsDF, savePlots = FALSE, outputName = "ROHom")
```

`plot_ViolinRuns` *Violin plot of run length per individual (either sum or mean)*

Description

Function to produce violin plots of the distribution of runs lengths per group The sum of run lengths, or its average, per individual sample is used to characterize the distribution of runs

Usage

```
plot_ViolinRuns(runs, method = c("sum", "mean"), outputName = NULL,
  plotTitle = NULL, savePlots = FALSE)
```

Arguments

`runs`	a data.frame with runs per individual (group, id, chrom, nSNP, start, end, length)
`method`	"sum" or "mean" of run lengths per individual samples
`outputName`	title prefix (the base name of graph, if savePlots is TRUE)
`plotTitle`	title in plot (default NULL)
`savePlots`	should plots be saved out to files or plotted in the graphical terminal (default)?

Value

Violin plot of the distribution of runs-lengths (sum or mean)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1,  minSNP = 15,
ROHet = FALSE,  maxOppositeGenotype = 1, maxMiss = 1,  minLengthBps = 100000,  minDensity = 1/10000)

## End(Not run)
```

```
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

plot_ViolinRuns(runs = runsDF, method = "sum" , savePlots = FALSE)
plot_ViolinRuns(runs = runsDF, method = "mean" , savePlots = FALSE)
```

`readExternalRuns` *Read runs from external files*

Description

Function to read in, from external files, the output of software for ROH:

1. `detectRUNS`: output saved out to a file (e.g. write.table)
2. `Plink`: output from the `--homozyg` option (`.hom` files)
3. `BCFtools`: output from the `roh` option

Usage

```
readExternalRuns(inputFile = NULL, program = c("plink", "BCFtools",
  "detectRUNS"))
```

Arguments

`inputFile`	name of (path to) external file
`program`	source program that produced the ROH file (one of `detectRUNS`, `Plink`, `BCFtools`)

Value

dataframe in the correct format to be used with plots and statistics functions from `detectRUNS`

Examples

```
# getting map and ped paths
## Not run:
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1,  minSNP = 15,
ROHet = FALSE,  maxMissRun = 1, maxMissWindow = 1,  minLengthBps = 100000,  minDensity = 1/10000)

write.table(x= runs,file = 'RunsFileTest.txt', quote=F, row.names = F)
newData=readRunsFromFile(runsFile = 'RunsFileTest.txt', program = 'detectRUNS')

## End(Not run)
```

readPOPCpp *Function to return a dataframe of population (POP, ID)*

Description

This is a core function. Read PED file and returns a data.frame with the first two columns

Usage

```
readPOPCpp(genotypeFile)
```

Arguments

genotypeFile	genotype (.ped) file location

Value

a dataframe of POP, ID

reorderDF *Function to reorder data frames by CHROMOSOME*

Description

The data frame will be reordered according to chromosome: from 1 to n, then X, Y, XY, MT The data frame needs to have a column with name "CHROMOSOME"

Usage

```
reorderDF(dfx)
```

Arguments

dfx	data frame to be reordered (with column "CHROMOSOME")

Details

Reorder results based on chromosome

Value

A reordered data frame by chromosome

`slidingRUNS.run` *Main function to detect RUNS (ROHom/ROHet) using sliding windows (a la Plink)*

Description

This is one of the main function of detectRUNS and is used to detect runs (of homozygosity or heterozygosity) in the genome (diploid) with the sliding-window method. All parameters to detect runs (e.g. minimum n. of SNP, max n. of missing genotypes, max n. of opposite genotypes etc.) are specified here. Input data are in the ped/map Plink format (https://www.cog-genomics.org/plink/1.9/input#ped)

Usage

```
slidingRUNS.run(genotypeFile, mapFile, windowSize = 15,
  threshold = 0.05, minSNP = 3, ROHet = FALSE, maxOppWindow = 1,
  maxMissWindow = 1, maxGap = 10^6, minLengthBps = 1000,
  minDensity = 1/1000, maxOppRun = NULL, maxMissRun = NULL)
```

Arguments

`genotypeFile`	genotype (.ped) file path
`mapFile`	map file (.map) file path
`windowSize`	the size of sliding window (number of SNP loci) (default = 15)
`threshold`	the threshold of overlapping windows of the same state (homozygous/heterozygous) to call a SNP in a RUN (default = 0.05)
`minSNP`	minimum n. of SNP in a RUN (default = 3)
`ROHet`	should we look for ROHet or ROHom? (default = FALSE)
`maxOppWindow`	max n. of homozygous/heterozygous SNP in the sliding window (default = 1)
`maxMissWindow`	max. n. of missing SNP in the sliding window (default = 1)
`maxGap`	max distance between consecutive SNP to be still considered a potential run (default = 10^6 bps)
`minLengthBps`	minimum length of run in bps (defaults to 1000 bps = 1 kbps)
`minDensity`	minimum n. of SNP per kbps (defaults to 0.1 = 1 SNP every 10 kbps)
`maxOppRun`	max n. of opposite genotype SNPs in the run (optional)
`maxMissRun`	max n. of missing SNPs in the run (optional)

Details

This function scans the genome (diploid) for runs using the sliding-window method. This is a wrapper function for many component functions that handle the input data (ped/map files), perform internal conversions, accept parameters specifications, select whether runs of homozygosity (RoHom) or of heterozygosity (RoHet) are looked for.

In the ped file, the groups samples belong to can be specified (first column). This is important if comparisons between human ethnic groups or between animal breeds or plant varieties or biological populations are to be performed. Also, if cases and controls are to be compared, this is the place where this information needs to be specified.

This function returns a data frame with all runs detected in the dataset. This data frame can then be written out to a csv file. The data frame is, in turn, the input for other functions of the detectRUNS package that create plots and produce statistics from the results (see plots and statistics functions in this manual, and/or refer to the detectRUNS vignette).

Value

A dataframe with RUNs of Homozygosity or Heterozygosity in the analysed dataset. The returned dataframe contains the following seven columns: "group", "id", "chrom", "nSNP", "from", "to", "lengthBps" (group: population, breed, case/control etc.; id: individual identifier; chrom: chromosome on which the run is located; nSNP: number of SNPs in the run; from: starting position of the run, in bps; to: end position of the run, in bps; lengthBps: size of the run)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")
# calculating runs with sliding window approach
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1,
minSNP = 15, ROHet = FALSE,  maxOppWindow = 1, maxMissWindow = 1, maxGap=10^6,
minLengthBps = 100000,  minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
colClasses <- c(rep("character", 3), rep("numeric", 4)  )
runs <- read.csv2(runsFile, header = TRUE, stringsAsFactors = FALSE,
colClasses = colClasses)
```

---

slidingWindow *Function to slide a window over a vector (individual's genotypes)*

Description

This is a core function. The functions to detect RUNS are slid over the genome

Usage

```
slidingWindow(data, gaps, windowSize, step, maxGap, ROHet = TRUE,
  maxOppositeGenotype = 1, maxMiss = 1)
```

Arguments

data	vector of 0/1/2 genotypes
gaps	vector of differences between consecutive positions (gaps) in bps
windowSize	size of window (n. of SNP)
step	by which (how many SNP) is the window slid
maxGap	max distance between consecutive SNP in a window to be still considered a potential run
ROHet	shall we detect ROHet or ROHom?
maxOppositeGenotype	max n. of homozygous/heterozygous SNP
maxMiss	max. n. of missing SNP

Value

vector of TRUE/FALSE (whether a window is homozygous or NOT)

---

slidingWindowCpp *Function to slide a window over a vector (individual's genotypes)*

Description

This is a core function. The functions to detect RUNS are slid over the genome

Usage

```
slidingWindowCpp(data, gaps, windowSize, step, maxGap, ROHet = TRUE,
  maxOppositeGenotype = 1L, maxMiss = 1L)
```

Arguments

data	vector of 0/1/2 genotypes
gaps	vector of differences between consecutive positions (gaps) in bps
windowSize	size of window (n. of SNP)
step	by which (how many SNP) is the window slid
maxGap	max distance between consecutive SNP in a window to be still considered a potential run
ROHet	shall we detect ROHet or ROHom?
maxOppositeGenotype	max n. of homozygous/heterozygous SNP
maxMiss	max. n. of missing SNP

Value

vector of TRUE/FALSE (whether a window is homozygous or NOT)

snpInRun *Function to return a vector of T/F for whether a SNP is or not in a RUN*

Description

This is a core function. The function to determine whether a SNP is or not in a RUN. The ratio between homozygous/heterozygous windows and total n. of windows is computed here

Usage

```
snpInRun(RunVector, windowSize, threshold)
```

Arguments

RunVector	vector of TRUE/FALSE (is a window homozygous/heterozygous?)
windowSize	size of window (n. of SNP)
threshold	threshold to call a SNP in a RUN

Value

vector of TRUE/FALSE (whether a SNP is in a RUN or NOT)

snpInRunCpp *Function to return a vector of T/F for whether a SNP is or not in a RUN*

Description

This is a core function. The function to determine whether a SNP is or not in a RUN. The ratio between homozygous/heterozygous windows and total n. of windows is computed here

Usage

```
snpInRunCpp(RunVector, windowSize, threshold)
```

Arguments

RunVector	vector of TRUE/FALSE (is a window homozygous/heterozygous?)
windowSize	size of window (n. of SNP)
threshold	threshold to call a SNP in a RUN

Value

vector of TRUE/FALSE (whether a SNP is in a RUN or NOT)

snpInsideRuns *Function to count number of times a SNP is in a RUN*

Description

Function to count number of times a SNP is in a RUN

Usage

```
snpInsideRuns(runsChrom, mapChrom, genotypeFile)
```

Arguments

runsChrom	R object (dataframe) with results per chromosome (column names:"POPULATION","IND","CHROMOS
mapChrom	R object (dataframe) with SNP name and position per chromosome (map file) (column names: "CHR","SNP_NAME","x","POSITION")
genotypeFile	genotype (.ped) file location

Value

dataframe with counts per SNP in runs (per population)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# defining mapChrom
mappa <- data.table::fread(mapFile, header = FALSE)
names(mappa) <- c("CHR","SNP_NAME","x","POSITION")
mappa$x <- NULL
chrom <- "24"
mapChrom <- mappa[mappa$CHR==chrom, ]

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
colClasses <- c(rep("character", 3), rep("numeric", 4)  )
runs <- read.csv2(runsFile, header = TRUE, stringsAsFactors = FALSE,
colClasses = colClasses)

# fix column names and define runsChrom
```

```
names(runs) <- c("POPULATION","IND","CHROMOSOME","COUNT","START","END","LENGTH")
runsChrom <- runs[runs$CHROMOSOME==chrom, ]

snpInsideRuns(runsChrom, mapChrom, genotypeFile)
```

---

snpInsideRunsCpp *Function to count number of times a SNP is in a RUN*

Description

Function to count number of times a SNP is in a RUN

Usage

```
snpInsideRunsCpp(runsChrom, mapChrom, genotypeFile)
```

Arguments

runsChrom	R object (dataframe) with results per chromosome
mapChrom	R map object with SNP per chromosome
genotypeFile	genotype (.ped) file location

Value

dataframe with counts per SNP in runs (per population)

---

summaryRuns *Summary statistics on detected runs*

Description

This function processes the results from slidingRUNS.run and consecutiveRUNS.run and produces a number of interesting descriptives statistics on results.

Usage

```
summaryRuns(runs, mapFile, genotypeFile, Class = 2, snpInRuns = FALSE)
```

Arguments

runs	R object (dataframe) with results on detected runs
mapFile	Plink map file (for SNP position)
genotypeFile	Plink ped file (for SNP position)
Class	group of length (in Mbps) by class (default: 0-2, 2-4, 4-8, 8-16, >16)
snpInRuns	TRUE/FALSE (default): should the function snpInsideRuns be called to compute the proportion of times each SNP falls inside a run in the group/population?

Details

summaryRuns calculates: i) the number of runs per chromosome and group/population; ii) the percent distribution of runs per chromosome and group; iii) the number of runs per size-class and group; iv) the percent distribution of runs per size-class and group; v) the mean length of runs per chromosome and group; vi) the mean length of runs per size-class and group; vii) individual inbreeding coefficient estimated from ROH; viii) individual inbreeding coefficient estimated from ROH per chromosome; ix) individual inbreeding coefficient estimated from ROH per size-class

Value

A list of dataframes containing the most relevant descriptives statistics on detected runs. The list conveniently contains 9 dataframes that can be used for further processing and visualization, or can be written out to text files

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1, minSNP = 15,
ROHet = FALSE, maxOppositeGenotype = 1, maxMiss = 1, minLengthBps = 100000, minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF <- readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

summaryRuns(runs = runsDF, mapFile = mapFile, genotypeFile = genotypeFile, Class = 2,
snpInRuns = FALSE)
```

---

tableRuns *Function to retrieve most common runs in the population*

Description

This function takes in input either the run results or the output from the function snpInsideRuns (proportion of times a SNP is inside a run) in the population/group, and returns a subset of the runs most commonly found in the group/population. The parameter threshold controls the definition of most common (e.g. in at least 50%, 70% etc. of the sampled individuals)

Usage

```
tableRuns(runs = NULL, SnpInRuns = NULL, genotypeFile, mapFile,
  threshold = 0.5)
```

Arguments

runs	R object (dataframe) with results on detected runs
SnpInRuns	dataframe with the proportion of times each SNP falls inside a run in the population (output from snpInsideRuns)
genotypeFile	Plink ped file (for SNP position)
mapFile	Plink map file (for SNP position)
threshold	value from 0 to 1 (default 0.7) that controls the desired proportion of individuals carrying that run (e.g. 70%)

Value

A dataframe with the most common runs detected in the sampled individuals (the group/population, start and end position of the run, chromosome and number of SNP included in the run are reported in the output dataframe)

Examples

```
# getting map and ped paths
genotypeFile <- system.file("extdata", "Kijas2016_Sheep_subset.ped", package = "detectRUNS")
mapFile <- system.file("extdata", "Kijas2016_Sheep_subset.map", package = "detectRUNS")

# calculating runs of Homozygosity
## Not run:
# skipping runs calculation
runs <- slidingRUNS.run(genotypeFile, mapFile, windowSize = 15, threshold = 0.1,  minSNP = 15,
ROHet = FALSE,  maxOppositeGenotype = 1, maxMiss = 1,  minLengthBps = 100000,  minDensity = 1/10000)

## End(Not run)
# loading pre-calculated data
runsFile <- system.file("extdata", "Kijas2016_Sheep_subset.sliding.csv", package="detectRUNS")
runsDF = readExternalRuns(inputFile = runsFile, program = 'detectRUNS')

tableRuns(runs = runsDF, genotypeFile = genotypeFile, mapFile = mapFile, threshold = 0.5)
```

---

writeRUN *Function to write out RUNS per individual animal*

Description

Function to write out RUNS per individual animal

Usage

```
writeRUN(ind, dRUN, ROHet = TRUE, group, outputName)
```

Arguments

ind	ID of animals
dRUN	data.frame with RUNS per animal
ROHet	shall we detect ROHet or ROHom?
group	group (factor): population, breed, ethnicity, case/control etc.
outputName	output filename

Value

TRUE/FALSE if RUNS are written out or not

Index