inveRsion

March 24, 2012

`GenoDat-class` *Class "GenoDat"*

Description

Object that handles genotype data

Objects from the Class

Objects can be created by calls of the form `setUpGenoDatFile` or `setUpGenoDatSNPmat` .

Slots

`genoDat`: `"matrix"` . Genotypes

`lociPos`: `"numeric"`. Genomic coordinates

`alleleSum`: `"matrix"`. Total number of variant alleles in the population per SNP

`noMissCount`: `"matrix"`. Total number of subjects with no-missing values

Methods

plot `signature(x = "GenoDat")`: Plots minor allele frequency

show `signature(object = "GenoDat")`: shows data summary

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`setUpGenoDatFile`, `setUpGenoDatSNPmat`

Examples

```
data(gDat)
gDat
```

`GenoDatROI-class` *Class "GenoDatROI"*

Description

`GenoDat` defined within an region of interest

Usage

```
ROIGenoDat(objectGenoDat, ROI)
```

Arguments

`objectGenoDat`	`GenoDat`
`ROI`	`numeric`. Region of interest. 2-component vector that defines the limits of a chromosome segments where an inversions is thought to occur.

Details

`ROIGenoDat` is the constructor of the class.

Objects from the Class

object are created with calls to `ROIGenoDat(objectGenoDat, ROI)`

Slots

`genoDat`: `"matrix"` . Genotypes

`lociPos`: `"numeric"`. Genomic coordinates

`alleleSum`: `"matrix"`. Total number of variant alleles in the population per SNP

`noMissCount`: `"matrix"`. Total number of subjects with no-missing values

`ROI`: `"numeric"`. Region of interest.

Extends

Class `"GenoDat"`, directly.

Methods

initialize `signature(.Object = "GenoDatROI")`

show `signature(object = "GenoDatROI")`

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`GenoDat`

Examples

```
data(gDat)
gDatROI<-ROIGenoDat(gDat,ROI=c(1268,1847))
gDatROI
```

`HaploCode-class` *Class "HaploCode"*

Description

The object stores the haplotype coding for each of the candidate brake-points. It is typically generated by calling the function `codeHaplo` on a `GenoDat` object.

Objects from the Class

Objects can be created by calls of `codeHaplo(objectGenoDat, blockSize, minAllele, saveRes = TRUE, file = NULL, ROI)`

Slots

`haploCode`: Object of class `"matrix"` Haplotype coding into decimal integers

`blockSize`: Object of class `"numeric"` block size of SNP used to identify the haplotype of each candidate brake-point.

`minAllele`: Object of class `"numeric"` minimum allele above which candidate brake-points are considered

Methods

initialize `signature(.Object = "HaploCode")`: Internal, users should use `codeHaplo`

show `signature(object = "HaploCode")`: shows HaploCode

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`codeHaplo`, `GenoDat`

Examples

```
data(gDat)
hapCode <-codeHaplo(gDat,blockSize=3,minAllele=0.3,saveRes=FALSE)
hapCode
```

`ac` *Sample data set of class accuracy*

Description

Illustrative output of function `accBic`. It stores the accuracy of classification of each chromosome into the inverted population and the frequency of the inversion as a for a range of Bic thresholds.

Usage

```
data(hapCode)
```

Format

The format is: Formal class '"accuracy"' [package "inveRsion"] with 1 slots ..@ out: num bicInt prob ac [1,] 0.0000 0.3180 0.9045 [2,] 142.4209 0.3745 0.9610 [3,] 284.8419 0.3945 0.9810 [4,] 427.2628 0.4130 0.9995 [5,] 569.6838 0.4130 0.9995

Examples

```
data(ac)
ac
```

`accBic` *accBic computes "accuracy" from "inversionList"*

Description

`accBic` computes the accuracy of the classification of chromosomes into previously known inverted and non-inverted populations. The classification is obtained from a majority vote of the classifications produced by the trial segment models whose BIC is greater than a given threshold.

Usage

```
accBic(object, mem, classFile, nsub, npoints, geno, wROI)
```

Arguments

`object`	of class `inversionList`
`mem`	vector with the numbering of chromosomes known to have the inversion
`classFile`	an alternative to `mem`, it passes the file name containing the numbering of chromosomes known to have the inversion.
`nsub`	total number of subjects (= 2* total number of chromosomes)
`npoints`	number of BIC threshold between 0 and max (BIC) for which the accuracy is to be computed
`geno`	whether the accuracy is assessed for inversion genotype or inversion allele (phased data).
`wROI`	integer indicating the ROI number to be used. The total number of ROIs are the total number of components in the `object` list.

Value

`accuracy`	object of class `accuracy`

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`inversionList`, `accuracy`

Examples

```
data(invList)
memFile <- system.file("extdata", "mem.txt", package = "inveRsion")
ac <- accBic(invList, classFile = memFile, nsub = 1000, npoints = 10)
plot(ac)
```

`accuracy-class` *Class "accuracy"*

Description

These objects hold the accuracy computation for the classification of chromosomes into inverted or non-inverted population for increasing levels of BIC thresholds.

Objects from the Class

`accuracy` is generated by calls to `accBic(inversionList, mem, classFile, nsub, npoints, geno, wROI)`, which is a method for the class `inversionList`

Slots

`out`: Object of class `"matrix"` matrix with three columns storing the range of BIC thresholds, probability of inversion within the population and accuracy of the classification.

Methods

plot `signature(x = "accuracy",w="character")`: plots accuracy Vs BIC for `w="a"`, or frequency of inversion Vs BIC for `w="f"`

Note

version R 2.10.1

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`accBic`, `inversionList`

Examples

```
data(ac)
ac
plot(ac)
```

codeHaplo *Codes haplotypes into decimal integers*

Description

The function labels the haplotypes of size `blockSize` around each candidate brake-point. For labeling genotype data, the function takes objects of class `genoDat` as main argument. For phased data, this argument should be ignored and a file name passed instead.

Usage

```
codeHaplo(objectGenoDat, blockSize, minAllele, saveRes = TRUE, file = NULL, ROI)
```

Arguments

`objectGenoDat`	`Genodat` object; if phased data then provide `file` instead
`blockSize`	`numeric`. number of SNPs flanking each side of each candidate brake-point. Default value 3
`minAllele`	`numeric`. minimum allele frequency for each probe to be considered as a candidate brake-point. Default value 0.1
`saveRes`	`logical`. Whether results should be saved in file `hapCode.RData`
`file`	`character`. File name with phased data
`ROI`	`numeric`. 2-vector specification passes a chromosome segment to be encoded. 4-vector specification passes the region of interest for the left brake-point (ROI[1] and ROI[2]) and the right brake-point (ROI[3] and ROI[4])

Details

When `setUpGenodat` is passed, the coding first computes the local haplotypes for each candidate-brake point form the genotype data and then encodes each haplotype into a decimal integer. The local haplotypes are computed with `haplo.em` form `haplo.stats` and assigns those with highest posterior probability to each chromosome. In the case of phased data, passed through `file`, no local haplotyping is needed and only the labeling is performed.

Value

Object of class `HaploCode`

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

References

http://mayoresearch.mayo.edu/mayo/research/schaid_lab/software.cfm

See Also

GenoDat, HaploCode,

Examples

```
data(gDat)
hapCode <-codeHaplo(gDat,blockSize=3,minAllele=0.3,saveRes=FALSE)
hapCode
```

---

gDat *Sample data of class genoDat*

Description

Data set used to illustrate local haplotype coding performed with `codeHaplo`

Usage

```
data(gDat)
```

Format

The format is: Formal class 'GenoDat' [package "inveRsion"] with 4 slots ..@ genoDat : int [1:9, 1:10] 0 1 0 1 0 0 1 0 0 1 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:9] "V1" "V2" "V3" "V4" ... .$: NULL ..@ lociPos : Named int [1:10] 959 1268 1393 1467 1531 1761 1847 1987 2006 2030 .. ..- attr(*, "names")= chr [1:10] "V1" "V2" "V3" "V4" ... ..@ alleleSum : num [1:10, 1] 3 1 8 3 9 1 1 9 1 1 ..@ noMissCount: num [1:10, 1] 9 9 9 9 9 9 9 9 9 9

Examples

```
data(gDat)
gDat
plot(gDat)
```

---

getClassif-methods *Overall classification*

Description

Classifies into inverted or non-inverted populations each chromosome in the sample.

Usage

```
getClassif(object, thBic, wROI, bin,geno,id)
```

Arguments

`object`	`inversionList`. List of inversions obtained from a chromosome scan.
`thBic`	`numeric`. BIC threshold above which significant segments are chosen for the final classification
`wROI`	`numeric`. ROI number from the list to select classification
`bin`	`logic`. Whether binary or continous classification is retrieved
`geno`	`logic`. Whether inversion genotype is retrieved
`id`	`character`. Vector of subject IDs

Details

The overall classification of chromosomes into inverted and non-inverted populations is given by the majority vote of the classifications obtained for each trial segment in the ROI, with BIC greater than `thBic`.

Value

`numeric`. Vector with values between 0 and 1 representing membership to the non-inverted and inverted population respectively.

Methods

`signature(object = "inversionList")` for each of the inversions of the list, it returns the classification of each chromosome.

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`inversionList`

Examples

```
data(invList)
r<-getClassif(invList)
head(r)
```

`getInv-methods` *gets "scan" into a matrix*

Description

Each row of the matrix represents a trial segment of fixed window size, for which the inversion model has been fit. It lists the left and right brake-points and output of the fitting: Log-likelihood ratio, probability of inversion, entropy, BIC (Bayes Information Criterion) and number of haplotypes.

Usage

```
getInv(object, thBic, rnd, Like)
```

Arguments

`object`	`scan`. Cromosome scanned for inversions with trial segments of fixed window size.
`thBic`	`numeric`. BIC threshold above which data is retrieved.
`rnd`	`logic`. Whether round matrix elements.
`Like`	`numeric`. Log-likelihood ratio threshold above which data is retrieved.

Details

Matrix with output of `scanInv`. Each row corresponds to a trail segments with given brake points and significance measures for the inversion model.

Value

`matrix`.

Methods

`signature(object = "scan")` returns matrix with output of inversion model for each trial segment

Examples

```
data(scanRes)
a<-getInv(scanRes,thBic=2500)
a
```

`getROIs-methods` *Extracts regions of possible inversion events from "scan"*

Description

lists the regions of interest in a matrix, given by the overlapping of significant trial segments (of fixed window seize) that may be part of an inversion event.

Usage

```
getROIs(object, thBic)
```

Arguments

`object`	`scan`. Cromosome scanned for inversions with trial segments of fixed window size.
`thBic`	BIC threshold above which overlapping segments are considered for the definition of each ROI

Details

ROIs are defined as overlapping trial segments with BIC greater than `thBIC`. The output is a matrix for which each row is one ROI. The first two columns give intervals defining the left brake-points and the two subsequent columns are the intervals for the right brake-points. ROIs are given in mega-basis.

Value

`matrix`

Methods

`signature(object = "scan")` list of regions of interest

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`scan`

Examples

```
data(scanRes)
ROI <- getROIs(scanRes, thBic = 0)
ROI
```

`hapCode` *Sample data set of class HaploCode*

Description

Illustrative data set, with local haplotypes encoded, to be used as input of `scanInv`.

Usage

```
data(hapCode)
```

Format

The format is: Formal class 'HaploCode' [package "inveRsion"] with 3 slots ..@ haploCode: num [1:2000, 1:583] 8 8 37 37 8 37 8 8 37 37 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: NULL .. .. ..$: chr [1:583] "0.602976-0.604061" "0.604061-0.605972" "0.60602-0.608417" "0.608668-0.608855" ... ..@ blockSize: num 3 ..@ minAllele: num 0.3

Examples

```
data(hapCode)
hapCode
```

`invList` *Sample data of class inversionList*

Description

list of objects of class `inversion`, each of which contains the information of overlapping segments that cover candidate inversions within a chromosome.

Usage

```
data(invList)
```

Format

The format is: Formal class 'inversionList' [package "inveRsion"] with 1 slots ..@ results:List of 1 .. ..$:Formal class 'inversion' [package "inveRsion"] with 8 slots

Details

The object is constructed with the function `listInv`

Examples

```
data(invList)
invList
```

`inveRsion-package` *Detection of Genetic inversions using SNP-array data*

Description

This package scans the whole genome in search of inversion events. Input data can be genotypes or phased haplotypes. It computes regions where inversions are probed by trial segments of fixed length. An inversion model is fit at each trial segment and significance measures, like Bayes Information Criterion, give evidence of segments belonging to the inversion event. Methods are implemented to identify the complete inversion segment and to classify the chromosomes in the sample as inverted or not.

Details

Package:	inveRsion
Type:	Package
Version:	1.0
Date:	2010-11-12
License:	GPL version 2 or newer
LazyLoad:	yes
Depends:	methods, haplo.stats

The package is designed as a stream analysis of a sequence of procedures: `setUpGenoDatFile` loads the genotype data onto R; `HaploCode` performs local haplotyping around the candidate brake points of the inversion; `sanInv` takes trial segments of fixed window size and fits the inversion model, sweeping the whole genome; and `listInv` summarizes the inversion events.

Author(s)

Alejandro Caceres Maintainer: Alejandro Caceres <acaceres@creal.cat>

References

A Caceres el al. Detection of genetic inversions with SNP-array data, manuscript in preparation.

SS Sindi and BJ Raphael, Identification and frequency estimation of inversion polymorphisms from haplotype data, J Comput Biol. 2010 Mar;17(3):517-31.

PF O'Reilly et al., invertFREGENE: software for simulating inversions in population genetic data, Bioinformatics. 2010 Mar 15;26(6):838-40. Epub 2010 Jan 26.

Examples

```
#vignette("inveRsionMan")
```

`inversion-class` *Class "inversion"*

Description

Internal class (not to be called by the user) that retrieves the output on the inversion model run in an ROI. `inversionList` is a list of objects of class `inversion`

Objects from the Class

Objects are created by calls to `listInv(object, hapCode, geno, ROI, saveRes, thBic, all)`

Slots

`classification`: Object of class `"vector"` overall classification (majority vote) of each chromosome for all the trail segments in the ROI

`leftBP`: Object of class `"vector"` left brake-points for each of the trial segments

`rightBP`: Object of class `"vector"` right brake-points for each of the trial segments

`bic`: Object of class `"vector"` BIC for the models on each trail segment

`intLeftBP`: Object of class `"vector"` interval for the left brake-point, in the ROI

`intRightBP`: Object of class `"vector"` interval for the right-brake point in the ROI

`invFreq`: Object of class `"numeric"` overall inversion frequency

`RR`: Object of class `"list"` classification of given by each of the trail segments.

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`listInv`, `inversionList`

`inversionList-class`   *Class "inversionList"*

Description

Lists output of the inversion model for each region of interest specified.

Objects from the Class

Objects can be created by calls of the form `listInv(object, hapCode, geno, ROI, saveRes, thBic, all)` and are lists of `inversion` objects.

Slots

`results`**:** Object of class `"list"` list of `inversion` objects

Methods

accBic `signature(object = "inversionList")`: Computes accuracy for chromosome classification when known

getClassif `signature(object = "inversionList")`: Extracts classification for each chromosome

plot `signature(x = "inversionList")`: Plots BIC values for trial segments

show `signature(object = "inversionList")`: shows object

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`listInv`, `HaploCode`

Examples

```
data(invList)
invList
```

`listInv-methods` *Constructor of class inversionList*

Description

Determines the full inversion sequence from overlapping trial segments of fixed window sized. It computes the limits of the inversion, the population frequency and retrieves the maximum BIC across the trial segments.

Usage

```
listInv(object, hapCode, geno, ROI, saveRes, thBic, all, saveROI)
```

Arguments

`object`	`scan`. Results of scanning the genome for an inversion with a fixed window size; output of `scanInv`.
`hapCode`	`HaploCode`. Object with the result of coding haplotypes for each candidate brake-point; output of `codeHaplo`
`geno`	`logical`. Whether original data is genotypes of phased haplotypes.
`ROI`	`numeric`. 2-vector specification passes the chromosome segment to be analyzed. 4-vector specification passes the region of interest for the left brake-point (ROI[1] and ROI[2]) and the right brake-point (ROI[3] and ROI[4]). ROI should be specified in mega-basis units.
`saveRes`	`logical`.Whether results should be saved into file `list Inv.RData`
`thBic`	`numeric`. BIC threshold above which trial segments are selected.
`all`	`logical`. Whether recomputing within the ROI should be done for all possible segment sizes or the window size in `scan`.
`saveROI`	`logical`. Whether saving the blocks for the candidate break-points for all the ROIs.

Details

`listInv` is both a method for class `scan` and constructor of `inversionList`. It re-runs the inversion model within the ROIs found in the previous scan. However, it is also possible to explicitly pass the ROIs defined by the user. The re-run is done with the same window size of the scan, which is convenient if enough significant trial segments where found within the inversion segment. If a more detailed re-run is needed set `all=TRUE`. This computes the model for all possible trial segments of any length within the ROI. Note that for high SNP density this can be computational intensive.

Value

`inversionList`

Methods

`signature(object = "scan")`

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`HaploCode`,`scan`,`inversionList`

Examples

```
data(scanRes)
data(hapCode)
invList<-listInv(scanRes,hapCode=hapCode,geno=FALSE,saveRes=FALSE,all=FALSE,thBic=0, save
invList
```

`scan-class` *Class "scan"*

Description

Results from scanning the genome with the inversion model for trial segments of fixed window size.

Objects from the Class

`scan` objects are typically generated by callings to the constructor function `scanInv`

Slots

`leftBP`: Object of class `"matrix"` Left brake-point coordinates (right SNP)

`rightBP`: Object of class `"matrix"` Right brake-point coordinates (right SNP)

`leftBP2`: Object of class `"matrix"` Left brake-point coordinates (left SNP)

`rightBP2`: Object of class `"matrix"` Right brake-point coordinates (right SNP)

`LogLike`: Object of class `"matrix"` Log-likelihood ratio for each trial segment

`prob`: Object of class `"matrix"` probability of no-inversion for each trial segment

`ent`: Object of class `"matrix"` entropy for each trial segment

`entTh`: Object of class `"matrix"` entropy threshold for each trial segment

`bic`: Object of class `"matrix"` BIC for each trial segment

`window`: Object of class `"numeric"` window size

Methods

getInv `signature(object = "scan")`: gets scan results into a matrix

getROIs `signature(object = "scan")`: get regions of interest, overlapping trial segments with significant BIC

listInv `signature(object = "scan")`: determines the inversion sequence for each ROI

plot `signature(x = "scan")`: plots `scan` results, set option `which=c("bic","log","prob","ent")` to plot BIC, log-likelihood ratio, probability of no inversion or entropy; and `thBic=0` to plot segments with BIC greater than 0

show `signature(object = "scan")`: shows scan results for each ROI

Author(s)

Alejandro Caceres <acaceres@creal.cat>

See Also

`listInv`, `HaploCode`, `getInv`, `getROIs`

Examples

```
data(scanRes)
scanRes
plot(scanRes,which="bic",thBic=0)
```

---

scanInv *Inversion scan*

Description

This function scans a whole chromosome in search for inversion events. The scan is done by fitting an inversion model to all segments in the chromosome with fixed length size.

Usage

```
scanInv(objectHaploCode, window, maxSteps = 30, geno = FALSE, saveRes = TRUE, sa
```

Arguments

`objectHaploCode`	Object of class `HaploCode` produced by the `codeHaplo` function.
`window`	`numeric`, size of the window in mega-basis.
`maxSteps`	`numeric`, maximum number of iteration in the EM algorithm for the inversion model
`geno`	`logical`. Whether the original data is genotypes or phases haplotypes.
`saveRes`	`logical`. Whether results should be saves into file `invRes.RData`
`saveBlocks`	`logical`. Whether save blocks for each candidate break point.

Details

The function processes the haplotypes coded in `objectHaploCode`. If subsequent re-runs are requires for different window sizes, this object can be omitted. The function will thus search the local directory for previous results to speed up further scans.

Value

object of class `scan`

Author(s)

Alejandro Caceres <acaceres@creal.cat>

See Also

`HaploCode`, `scan`

Examples

```
data(hapCode)
window<-0.5
scanRes<-scanInv(hapCode,window=window,saveRes=FALSE,geno=FALSE,saveBlocks=FALSE)
scanRes
plot(scanRes)
```

`scanRes` *Sample data set of class scan*

Description

Sample set that illustrates the result of scanning a chromosome with segments of fix window size. It fits the inversion model for each segment and stores its results.

Usage

```
data(scanRes)
```

Format

The format is: Formal class 'scan' [package "inveRsion"] with 10 slots ..@ leftBP : num [1:189, 1] 0.603 0.604 0.606 0.609 0.612 ... ..@ rightBP : num [1:189, 1] 1.15 1.15 1.15 1.15 1.15 ... ..@ leftBP2 : num [1:189, 1] 0.604 0.606 0.608 0.609 0.612 ... ..@ rightBP2: num [1:189, 1] 1.15 1.15 1.15 1.15 1.15 ... ..@ LogLike : num [1:189, 1] 27.2 45.4 89.8 108.2 113.3 ... ..@ prob : num [1:189, 1] 0.996 0.987 0.947 0.936 0.946 ... ..@ ent : num [1:189, 1] 2.41 2.33 2.78 2.87 2.92 ... ..@ entTh : num [1:189, 1] 29 23 26 30 24 28 20 21 24 20 ... ..@ bic : num [1:189, 1] -193.2 -129.4 -107.8 -119.8 -69.1 ... ..@ window : num 0.5

Details

The object is constructed with the function `scanInv` and used as an input for `listInv`.

Examples

```
data(scanRes)
plot(scanRes)
```

`setUpGenoDatFile` *Loads genotype data onto R*

Description

Loads onto an R session genotype data from text files or PLINK files.

Usage

```
setUpGenoDatFile(file = "GenoDat.txt", saveRes = FALSE, sortMinor = TRUE)
setUpGenoDatSNPmat(Chr, Geno, Annot, saveRes = FALSE, saveGeno = FALSE)
```

Arguments

`file`	`character`. File path with genotype information
`saveRes`	`logical`. Wether results should be saved into file `gDat.RData`
`sortMinor`	`logical`. Whether genotypes should be sorted by minor allele frequency.
`Chr`	`numeric`. Chromosome number
`Geno`	`snpMatrix`. Matrix of `raw` with genotype data
`Annot`	`numeric`. Annotation information read from an `.bim` file
`saveGeno`	`logical`. Wether `.txt` file should be saved with genotype information

Value

`GenoDat` object

Author(s)

Alejandro Caceres `<acaceres@creal.cat>`

See Also

`GenoDat`

Examples

```
gen <- system.file("extdata", "genotypes.txt", package = "inveRsion")
gDat <-setUpGenoDatFile(file=gen,sortMinor=TRUE,saveRes=FALSE)
gDat
```

Index