

Package ‘SNPMaP’

April 17, 2009

Type Package

Title SNP Microarrays and Pooling in R

Version 1.0.2

Date 2008-06-24

Author Oliver SP Davis, Leo C Schalkwyk

Maintainer Oliver SP Davis <snpmap@iop.kcl.ac.uk>

Description Pooling DNA on SNP microarrays is a cost-effective way to carry out genome-wide association studies for heritable disorders or traits. The SNPMaP package provides formal SNPMaP objects and methods in R as a base for these analyses using Affymetrix genotyping arrays.

License GPL (>= 3)

LazyLoad yes

LazyData yes

Depends methods, SNPMaP.cdm, affxparser, R.huge, R.utils, R (>= 2.4.0)

Repository CRAN

Date/Publication 2008-07-05 18:25:52

R topics documented:

SNPMaP-package	2
cel2ras	4
cloneSNPMaP	6
disk2memory	7
getSNPMaP	7
minusMismatch	8
norm	9
qcMean	10
snpmap	10
SNPMaP-class	12
writeSNPMaP	16

Description

Pooling DNA on SNP microarrays is a cost-effective way to carry out genome-wide association studies for heritable disorders or traits. The SNPMaP package provides formal SNPMaP objects and methods in R as a base for these analyses using Affymetrix genotyping arrays.

Details

Package: SNPMaP
Type: Package
Version: 1.0.2
Date: 2008-06-24
License: GPL (>= 3)

A SNPMaP study involves selecting individuals based on their disease status (case or control) or their score on a quantitative trait (high or low extremes). DNA from these individuals is pooled in biological replicate pools (20 pools of 50 different individuals each, for example). Each pool is then genotyped according to the standard protocols on genotyping microarrays. The probe intensities from these arrays can be extracted from the CEL files into SNPMaP objects using the `snpmap()` function and Relative Allele Scores (RAS) can be calculated as estimates of allele frequency in each pool. By comparing the allele frequencies in high and low or case and control pools, hundreds of thousands of SNPs across the genome can be screened for association with a trait or disorder. The SNPMaP package also provides methods for visualising the data at each stage of the analysis. Using the `lowMemory` option allows this to be done on a standard desktop computer (albeit slower than if all data is kept in memory). This package is an evolution of the scripts referred to in Meaburn et al (2006) and is described in Davis, Plomin and Schalkwyk (submitted for publication); please cite this paper if you find the package useful. Additional supporting material is available at <http://sgdp.iop.kcl.ac.uk/snpmap/>.

Array types

Details of the arrays supported by the current version of SNPMaP can be found in the `SNPMaP.cdm` package.

Future work

Future releases of the package will build on the range of chips covered as well as introducing pre-packaged methods for carrying out the genome-wide screen for association.

Author(s)

Oliver SP Davis and Leo C Schalkwyk

Maintainer: Oliver SP Davis (snpmap@iop.kcl.ac.uk)

References

Davis, OSP, Plomin, R, and Schalkwyk, LC. (submitted for publication) The SNPMaP package for R: A framework for genome-wide association using DNA pooling on microarrays.

Meaburn E, Butcher LM, Schalkwyk LC, and Plomin R. (2006) Genotyping pooled DNA using 100K SNP microarrays: a step towards genomewide association scans. *Nucleic Acids Research*, 34(4):e28. <http://dx.doi.org/10.1093/nar/gnj027>

See Also

[snpmap\(\)](#) to set up a SNPMaP analysis.
[SNPMaP-class](#) to represent a SNPMaP study.
[SNPMaP.cdm-package](#) for the cdm matrices that interpret the 'raw' format SNPMaP objects.
[affxparser-package](#) that reads the CEL files.
[R.huge-package](#) that provides FileDoubleMatrices for the lowMemory option.
[methods-package](#) for S4 formal classes.

Examples

```
## Not run:
## Getting started
## Creates the 'raw' SNPMaP object x on disk with mismatch probes included
x<-snpmap(useMM=TRUE, RUN='cel2raw', lowMemory=TRUE)
## Print a summary of the SNPMaP object
summary(x)
## Add a comment (prints in the summary)
comment(x)<-'High and low extreme pools from January'
## View pseudo image to screen for artefacts
image(x)
## Plot probe intensities
plot(x, FUN=log)
boxplot(x, FUN=log)
## tidy=TRUE removes the FileDoubleMatrix from the old x to keep the disk tidy
x<-raw2ras(x, tidy=TRUE)
## Plot Relative Allele Scores
plot(x)
## Default tidy=FALSE does not remove the original FileDoubleMatrix from disk
## Useful if you want to keep x (no side effects)
y<-ras2rasS(x)
## View the first ten rows
as.matrix(y[1:10,])
## View a set of SNPs
as.matrix(y[c("SNP_A-4192909", "SNP_A-4192918"),])
## Transfer the SNPMaP object from disk to memory
y<-disk2memory(y, tidy=TRUE)
## Run the analysis again from CEL files to RAS summaries without viewing intermediate stages
## This time in memory (may require a lot of RAM)
z<-snpmap(useMM=TRUE, RUN='cel2rasS', lowMemory=FALSE)
plot(z)
```

```
## Get the RAS summary scores as a standard matrix
rasSummaries<-as.matrix(z)
## Read all the sets into a list
allSets<-msnpmap(set=0)
## End(Not run)
```

cel2ras

SNPMaP workflows

Description

Workflow functions to process SNPMaP objects. Called by `snpmap()`.

Usage

```
cel2raw(x, cels = x@chps, lowMemory = x@lowMemory, ...)
cel2long(x, cels = x@chps, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
         log.intensities = x@logInt, ...)
cel2short(x, cels = x@chps, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
          log.intensities = x@logInt, ...)
cel2ras(x, cels = x@chps, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
        log.intensities = x@logInt, subtractMismatch = x@useMM, ...)
cel2rasS(x, cels = x@chps, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
         log.intensities = x@logInt, subtractMismatch = x@useMM, FUN = x@summary, ...)
raw2long(x, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
         log.intensities = x@logInt, tidy = FALSE, ...)
raw2short(x, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
          log.intensities = x@logInt, tidy = FALSE, ...)
raw2ras(x, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
        log.intensities = x@logInt, subtractMismatch = x@useMM, tidy = FALSE, ...)
raw2rasS(x, lowMemory = x@lowMemory, set = x@set, normalize = x@normalize,
         log.intensities = x@logInt, subtractMismatch = x@useMM, FUN = x@summary,
         tidy = FALSE, ...)
long2short(x, lowMemory = x@lowMemory, tidy = FALSE, ...)
long2ras(x, lowMemory = x@lowMemory, subtractMismatch = x@useMM, tidy = FALSE, ...)
long2rasS(x, lowMemory = x@lowMemory, subtractMismatch = x@useMM, FUN = x@summary,
          tidy = FALSE, ...)
short2ras(x, lowMemory = x@lowMemory, subtractMismatch = x@useMM, tidy = FALSE, ...)
short2rasS(x, lowMemory = x@lowMemory, subtractMismatch = x@useMM, FUN = x@summary,
           tidy = FALSE, ...)
ras2rasS(x, lowMemory = x@lowMemory, FUN = x@summary, tidy = FALSE, ...)
```

Arguments

<code>x</code>	An object of class <code>SNPMaP</code> .
<code>cels</code>	character; a vector of the CEL files to be included.
<code>lowMemory</code>	logical; should the SNP data be stored on disk rather than in memory?

tidy	logical; if the data is stored on disk, should the old FileDoubleMatrix be unlinked ?
set	numeric; the set of probesets to include. See SNPMaP.cdm-package .
normalize	logical; quantile normalize probe intensities across arrays.
log.intensities	logical; take natural log of probe intensities.
subtractMismatch	logical; subtract mismatch probe intensities if available from perfect match intensities.
FUN	function; function to compute summary statistic for RAS on each chip. See SNPMaP-class .
...	additional arguments passed to other workflow and internal functions.

Details

The workflow functions work by calling `cel2raw()`, `raw2long()`, `long2short()`, `short2ras()`, `ras2rasS()`.

Value

Object of class [SNPMaP](#).

See Also

[SNPMaP-package](#), [SNPMaP.cdm-package](#), [SNPMaP-class](#), `snpmap()`, `norm()`, `logIntensities()`

Examples

```
## Not run:
## Getting started
## Creates the 'raw' SNPMaP object x on disk with mismatch probes included
x<-snpmap(useMM=TRUE, RUN='cel2raw', lowMemory=TRUE)
## Print a summary of the SNPMaP object
summary(x)
## Add a comment (prints in the summary)
comment(x)<-'High and low extreme pools from January'
## View pseudo image to screen for artefacts
image(x)
## Plot probe intensities
plot(x, FUN=log)
boxplot(x, FUN=log)
## tidy=TRUE removes the FileDoubleMatrix from the old x to keep the disk tidy
x<-raw2ras(x, tidy=TRUE)
## Plot Relative Allele Scores
plot(x)
## Default tidy=FALSE does not remove the original FileDoubleMatrix from disk
## Useful if you want to keep x (no side effects)
y<-ras2rasS(x)
## View the first ten rows
as.matrix(y[1:10,])
```

```

## View a set of SNPs
as.matrix(y[c("SNP_A-4192909", "SNP_A-4192918"),])
## Transfer the SNPMaP object from disk to memory
y<-disk2memory(y, tidy=TRUE)
## Run the analysis again from CEL files to RAS summaries without viewing intermediate stages
## This time in memory (may require a lot of RAM)
z<-snpmap(useMM=TRUE, RUN='cel2rasS', lowMemory=FALSE)
plot(z)
## Get the RAS summary scores as a standard matrix
rasSummaries<-as.matrix(z)
## Read all the sets into a list
allSets<-msnpmap(set=0)
## End(Not run)

```

cloneSNPMaP

Copy SNPMaP objects on disk

Description

If you try to copy a SNPMaP object on disk to another object, it will just copy the `FileDoubleMatrix` filehandles. To copy the data, you need to `cloneSnpmap()`.

Usage

```
cloneSNPMaP(x, ...)
```

Arguments

<code>x</code>	An object of class <code>SNPMaP</code> .
<code>...</code>	Additional arguments.

Value

An object of class `SNPMaP`: a copy of `x`.

See Also

[SNPMaP-class](#)

disk2memory	<i>Transferring SNPMap objects between disk and memory</i>
-------------	--

Description

Transfer the data in a SNPMap object from FileDoubleMatrix to memory or from memory to FileDoubleMatrix.

Usage

```
disk2memory(x, tidy = FALSE, ...)  
memory2disk(x, ...)
```

Arguments

x	An object of class SNPMap .
tidy	logical; should the old FileDoubleMatrix be unlinked ?
...	Additional arguments.

Value

An object of class [SNPMap](#).

See Also

[SNPMap-package](#)

getSNPMap	<i>Get or set slots of a SNPMap object</i>
-----------	--

Description

Get or set the values of the slots in a SNPMap object.

Usage

```
getSNPMap(x, ...)  
setSNPMap(x, namedList)
```

Arguments

x	An object of class SNPMap .
namedList	A list of the form <code>list(useMM=TRUE, normalize=FALSE)</code> ; the exact name of a slot followed by the new value.
...	Slots to retrieve data from (may be partial matches).

Details

Currently, `getSNPMaP()` will retrieve the values of the following slots: `useMM`, `normalize`, `logInt`, `summary`, `lowMemory`, `tempDir`, `table`, `chiptype`, `celDim`, `set`, `snps`, `chps`, `cols`, `width`, `transformation`, `experiment`, `created`, `version`, `majorHistory`. `setSNPMaP()` will set the following slots: `useMM`, `normalize`, `logInt`, `summary`, `lowMemory`, `tempDir`, `set`, `experiment`. Comments can be retrieved and set using `comment()`.

Value

For `getSNPMaP()`, a named list of slot values (suitable as an argument for `setSNPMaP()`). For `setSNPMaP()`, a named list of the old slot values (suitable as an argument for `setSNPMaP()`).

Warning

`setSNPMaP()` does not clone the data in `SNPMaP` objects stored on disk (see `cloneSNPMaP()`.)

See Also

[SNPMaP-class](#)

Examples

```
## Not run:
## Retrieve slot values
getSNPMaP(x, 'experiment', 'snps')
## Set new slot values
x <- setSNPMaP(x, list(useMM=TRUE, normalize=FALSE))
## End(Not run)
```

minusMismatch

Subtract mismatch intensities

Description

Subtract average (within a quartet) mismatch probe (MM) intensities from perfect match probe (PM) intensities in a 'short' format `SNPMaP` object.

Usage

```
minusMismatch(x, tidy = FALSE, ...)
```

Arguments

<code>x</code>	An object of class <code>SNPMaP</code> in 'short' format.
<code>tidy</code>	logical; should the old <code>FileDoubleMatrix</code> be <code>unlinked</code> ?
<code>...</code>	Additional arguments.

Value

An object of class [SNPMaP](#).

See Also

[SNPMaP-package](#)

norm

Transform a SNPMaP object

Description

Quantile normalize or (natural) log the intensities in a 'raw' format SNPMaP object.

Usage

```
norm(x, Force = FALSE, lowMemory = x@lowMemory, tidy = FALSE, ...)  
logIntensities(x, lowMemory = x@lowMemory, tidy = FALSE, ...)
```

Arguments

x	An object of class SNPMaP in 'raw' format.
Force	logical; Force=TRUE will allow SNPMaP objects in formats other than 'raw' to be quantile normalized.
lowMemory	logical; should the data be stored on disk rather than in memory?
tidy	logical; should the old FileDoubleMatrix be unlinked ?
...	Additional arguments.

Value

An object of class [SNPMaP](#).

See Also

[SNPMaP-package](#)

 qcMean

Summarise Relative Allele Scores

Description

The default function for summarising the RAS from the quartets in a probeset into one value per probeset per array.

Usage

```
qcMean(x)
```

Arguments

`x` numeric; a vector of RAS from the quartets comprising a single probeset on a single array.

Value

Returns the mean of the RAS scores (na.rm=T) unless more than a third of the quartets are NA, in which case returns NA.

See Also

[ras2rasS](#) for the workflow function that calls the RAS summary function. [SNPMaP-package](#).

 snpmap

SNP Microarrays and Pooling

Description

Functions to process SNPMaP data from CEL files to Relative Allele Score summaries (rasS).

Usage

```
snpmap(cels = dir(pattern = ".[cC][eE][lL]$"), lowMemory = TRUE, set = 1,
       useMM = FALSE, normalize = FALSE, log.intensities = FALSE, ras.summary = qcMean,
       tempDir = ".", RUN = "cel2rasS", interactive=FALSE, ...)
msnpmap(cels = dir(pattern = ".[cC][eE][lL]$"), lowMemory = TRUE, set = 0,
        useMM = FALSE, normalize = FALSE, log.intensities = FALSE, ras.summary = qcMean,
        tempDir = ".", RUN = "cel2rasS", interactive=FALSE, ...)
```

Arguments

<code>cels</code>	character; a vector of CEL files to use in the analysis
<code>lowMemory</code>	logical; should the SNP data be stored on disk rather than in memory?
<code>set</code>	numeric, a single number for <code>snpmap</code> or a vector for <code>msnpmap</code> ; the probesets to include see SNPMaP.cdm-package . The special <code>set=0</code> corresponds to all the <code>sets</code> on the array.
<code>useMM</code>	logical; should mismatch probes be used in the analysis (if they are present on the array)?
<code>normalize</code>	logical; should the probe intensities be quantile normalised across arrays?
<code>log.intensities</code>	logical; use the natural log of the probe intensities?
<code>ras.summary</code>	function to compute summary statistic for RAS scores on each chip. See SNPMaP-class .
<code>tempDir</code>	character; writable directory to store the probe data if <code>lowMemory=TRUE</code> .
<code>RUN</code>	character, name of a workflow function <code>cel2*</code> ; see <code>cel2ras()</code> .
<code>interactive</code>	logical; should <code>snpmap()</code> prompt for CEL files (Windows only)?
<code>...</code>	additional arguments passed on to workflow function.

Details

`snpmap()` sets up the SNPMaP analysis and calls a workflow function to extract the data from the CEL files to 'raw' intensity data for all probes on the array (at any `set`), to a 'long' format matrix (one column per array), to a 'short' format matrix (one row per probeset), to 'ras' (a matrix of Relative Allele Scores, one per quartet, one row per probeset), or to 'rasS' (a matrix of RAS, one per probeset, one column per array). Specifying `lowMemory=TRUE` stores the data as a [FileDoubleMatrix](#) on disk rather than in memory.

Value

Object of class [SNPMaP](#) for `snpmap()`; list of [SNPMaP](#) objects for `msnpmap()`.

Warning

`snpmap()` returns the SNPs from a single `set` on the array. For some arrays you will need to call `snpmap()` several times to access all the SNPs, or `msnpmap(set=0)`.

See Also

[SNPMaP-package](#).
[SNPMaP.cdm-package](#).
[SNPMaP-class](#) for [SNPMaP](#) objects and methods.
`cel2ras()` for workflow functions.
`disk2memory()` to transfer [SNPMaP](#) objects between disk and memory.
`cloneSNPMaP()` to copy a [SNPMaP](#) object on disk.
`writeSNPMaP()` to write [SNPMaP](#) objects to text files.
`norm()`, the quantile normalization function.
`logIntensities()`, the function that logs the raw intensities.

Examples

```
## Not run:
## Getting started
## Creates the 'raw' SNPMaP object x on disk with mismatch probes included
x<-snpmap(useMM=TRUE, RUN='cel2raw', lowMemory=TRUE)
## Print a summary of the SNPMaP object
summary(x)
## Add a comment (prints in the summary)
comment(x)<-'High and low extreme pools from January'
## View pseudo image to screen for artefacts
image(x)
## Plot probe intensities
plot(x, FUN=log)
boxplot(x, FUN=log)
## tidy=TRUE removes the FileDoubleMatrix from the old x to keep the disk tidy
x<-raw2ras(x, tidy=TRUE)
## Plot Relative Allele Scores
plot(x)
## Default tidy=FALSE does not remove the original FileDoubleMatrix from disk
## Useful if you want to keep x (no side effects)
y<-ras2rasS(x)
## View the first ten rows
as.matrix(y[1:10,])
## View a set of SNPs
as.matrix(y[c("SNP_A-4192909", "SNP_A-4192918"),])
## Transfer the SNPMaP object from disk to memory
y<-disk2memory(y, tidy=TRUE)
## Run the analysis again from CEL files to RAS summaries without viewing intermediate stages
## This time in memory (may require a lot of RAM)
z<-snpmap(useMM=TRUE, RUN='cel2rasS', lowMemory=FALSE)
plot(z)
## Get the RAS summary scores as a standard matrix
rasSummaries<-as.matrix(z)
## Read all the sets into a list
allSets<-msnpmap(set=0)
## End(Not run)
```

SNPMaP-class

Class SNPMaP

Description

A class of objects for SNP Microarrays and Pooling using Affymetrix arrays.

Usage

```
## S4 method for signature 'SNPMaP':
as.matrix(x, mm=FALSE, ...)
## S4 method for signature 'SNPMaP, missing':
```

```

plot(x, FUN=function(x){x}, xlim=c(loX, hiX), ylim=c(0, hiY), xlab="guess",
      ylab="Density", main=x@table, col=rainbow(length(x@chps)),
      legend.position="left", legend.bty="n", lty=1:length(x@chps),
      zero.line=TRUE, ...)
## S4 method for signature 'SNPMaP':
boxplot(x, FUN=function(x){x}, ylab="guess", main=x@table, ...)
## S4 method for signature 'SNPMaP':
image(x, chips=x@chps, prompt=FALSE, FUN=log, col=grey(seq(0,1,0.01)), fastRender=4,
      rows=x@celDim[1], cols=x@celDim[2], ...)

```

Arguments

<code>x</code>	object of class <code>SNPMaP</code> .
<code>mm</code>	logical indicating whether the data in the <code>mismatch</code> slot should be returned instead.
<code>FUN</code>	function specifying how each data point should be transformed before plotting. Try <code>log</code> for probe intensity data.
<code>xlim</code>	the x limits (<code>x1</code> , <code>x2</code>) of the plot. Note that <code>x1 > x2</code> is allowed and leads to a 'reversed axis'.
<code>ylim</code>	the y limits of the plot.
<code>xlab</code>	character; a label for the x axis, defaults to a description of x.
<code>ylab</code>	character; a label for the y axis, defaults to a description of y.
<code>main</code>	character; a main title for the plot.
<code>col</code>	the colours for the lines.
<code>legend.position</code>	position of the legend in the pane to the right of the graph.
<code>legend.bty</code>	the type of box to be drawn around the legend. The allowed values are "o" and "n".
<code>lty</code>	the line type.
<code>zero.line</code>	logical; if TRUE, add a base line at <code>y = 0</code>
<code>chips</code>	character or numeric; the arrays to be imaged from the <code>chps</code> slot.
<code>prompt</code>	logical indicating whether the user should be asked before moving on to the next image.
<code>fastRender</code>	numeric <i>n</i> indicating $1/n$ rows and columns should be imaged.
<code>rows</code>	numeric; the number of rows of probes on the array.
<code>cols</code>	numeric; the number of columns of probes on the array.
<code>...</code>	additional arguments passed to methods.

Slots

snpdata: Object of class `SNPMaPdata`, either a matrix or a `FileDoubleMatrix` with probe intensities or Relative Allele Score (RAS) estimates from the SNPMaP experiment.

mismatch: As `snpdata`. Contains mismatch probe intensities if they are present on the microarray and required for analysis.

- useMM:** Object of class `logical`, if `TRUE` then the mismatch probe intensities will be subtracted from the perfect match probe intensities during the analysis. Defaults to `FALSE`.
- normalize:** Object of class `logical`, if `TRUE` then the probe intensities will be quantile normalised across arrays during the analysis. Defaults to `FALSE`.
- logInt:** Object of class `logical`, if `TRUE` then the raw probe intensities from the array will be replaced with the natural log of the intensities during the analysis. Defaults to `FALSE`.
- summary:** Object of class `function`, the function that will be used to summarise the vector of RAS scores from all the quartets corresponding to a single SNP within an array into a single statistic per SNP per array. Should accept a `numeric` vector comprising doubles between 0 and 1 and NA and return a single value. Defaults to `qcMean`.
- lowMemory:** Object of class `logical`, if `TRUE` then the analysis will attempt to minimise the amount of RAM used by storing the `snpdata` and `mismatch` matrices as objects of class `FileDoubleMatrix` (see the link `[R.huge:R.huge-package]{R.huge}` package) on disk at the location named in the `tempDir` slot. Defaults to `TRUE`.
- tempDir:** Object of class `character`. Names the location of a writable directory where `lowMemory` versions of the `snpdata` and `mismatch` matrices may be stored. Defaults to `'.'` (ie the working directory).
- table:** Object of class `character`. Gives the current format of the `snpdata`. May be (in order) `'empty'`, `'raw'`, `'long'`, `'short'`, `'ras'` or `'rasS'`.
- chiptype:** Object of class `character` naming the type of array involved in this analysis (`'Mapping250K_Sty'`, for example). The arrays in a `SNPMaP` object must be of a single `chiptype`.
- celDim:** Object of class `numeric` vector of two giving the number of rows and columns of probes on the chip. Read from the CEL header by `cel2raw()`, used for pseudo images of arrays.
- set:** Object of class `numeric`. Several of the Affymetrix chips have groups of probesets with different numbers of quartets. For example, the `Mapping250K_Sty` array has probesets comprising 6 or 10 quartets. `set` specifies which group of probes are to be included in the current `SNPMaP` object. It defaults to 1, which corresponds to the largest group of probes on the array (ie those with 6 quartets on the `Mapping250K_Sty`). Further groups can be accessed by setting `set` to 2 or higher. For more information, see `SNPMaP.cdm-package`.
- snp:** Object of class `character`. A vector of the SNPs being analysed in the current `SNPMaP` object.
- chps:** Object of class `character`. A vector of the CEL files in the current analysis. Defaults to all the files ending `'.CEL'` in the working directory.
- cols:** Object of class `character`. A vector identifying the columns of the `snpdata` and `mismatch` slots of the current `SNPMaP` object.
- width:** Object of class `numeric`. The number of quartets per SNP in the current group of probesets.
- transformation:** Object of class `character`. A vector to record transformations carried out on the data.
- experiment:** Object of class `factor`. A factor identifying the experimental group each array belongs to. Not used at present.
- annotation:** Object of class `list`. A slot to hold annotation to be used in the analysis. Not used at present.

- created:** Object of class `character`; the date and time when this SNPMaP object was created.
- version:** Object of class `character`; the version number of the SNPMaP package that created this object.
- majorHistory:** Object of class `list`. A record of the major functions carried out on this SNPMaP object, along with the date and time they were called.

Methods

- as.matrix** signature (`x="SNPMaP"`): returns the contents of the `snpdata` slot as a `matrix`, or, if `mm=TRUE`, returns the contents of the `mismatch` slot.
- plot** signature (`x="SNPMaP"`, `y="missing"`): plots the density of the contents of the `snpdata` slot, with each array represented by a single line identified in a legend. Uses `layout`.
- boxplot** signature (`x="SNPMaP"`): box plot for each array in the `snpdata` slot.
- image** signature (`x="SNPMaP"`): if the object is in 'raw' format, draws a pseudo image of the array as it was scanned to check for artefacts.
- open** signature (`con="SNPMaP"`): if the data in the SNPMaP object is stored on disk, open a connection.
- close** signature (`con="SNPMaP"`): if the data is on disk, close the connection.
- summary** signature (`object="SNPMaP"`): summary of the SNPMaP object.
- [signature (`x="SNPMaP"`, `i="missing"`, `j="missing"`): Extract subsets of the data contained in the `snpdata` slot of the SNPMaP object. Using a single index (`[]` rather than `[,]`) returns `rows`.
- [signature (`x="SNPMaP"`, `i="ANY"`, `j="missing"`): see above.
- [signature (`x="SNPMaP"`, `i="missing"`, `j="ANY"`): see above.
- [signature (`x="SNPMaP"`, `i="ANY"`, `j="ANY"`): see above.
- initialize** signature (`.Object="SNPMaP"`): default initialize method.

See Also

- [SNPMaP-package](#).
- [SNPMaP.cdm-package](#).
- `snpmap()` to set up a SNPMaP analysis.
- `disk2memory()` to transfer SNPMaP objects between disk and memory.
- `cel2ras()` for workflow functions.
- `cloneSNPMaP()` to copy a SNPMaP object on disk.
- `writeSNPMaP()` to write SNPMaP objects to text files.

Examples

```
## Not run:
## Getting started
## Creates the 'raw' SNPMaP object x on disk with mismatch probes included
x<-snpmap(useMM=TRUE, RUN='cel2raw', lowMemory=TRUE)
## Print a summary of the SNPMaP object
summary(x)
```

```

## Add a comment (prints in the summary)
comment(x)<-'High and low extreme pools from January'
## View pseudo image to screen for artefacts
image(x)
## Plot probe intensities
plot(x, FUN=log)
boxplot(x, FUN=log)
## tidy=TRUE removes the FileDoubleMatrix from the old x to keep the disk tidy
x<-raw2ras(x, tidy=TRUE)
## Plot Relative Allele Scores
plot(x)
## Default tidy=FALSE does not remove the original FileDoubleMatrix from disk
## Useful if you want to keep x (no side effects)
y<-ras2rasS(x)
## View the first ten rows
as.matrix(y[1:10,])
## View a set of SNPs
as.matrix(y[c("SNP_A-4192909", "SNP_A-4192918"),])
## Transfer the SNPMaP object from disk to memory
y<-disk2memory(y, tidy=TRUE)
## Run the analysis again from CEL files to RAS summaries without viewing intermediate stages
## This time in memory (may require a lot of RAM)
z<-snpmap(useMM=TRUE, RUN='cel2rasS', lowMemory=FALSE)
plot(z)
## Get the RAS summary scores as a standard matrix
rasSummaries<-as.matrix(z)
## Read all the sets into a list
allSets<-msnpmap(set=0)
## End(Not run)

```

writeSNPMaP

Write a SNPMaP object to a text file

Description

Transfer the data in a SNPMaP object to a text file (and read the file back into R efficiently).

Usage

```

writeSNPMaP(x, file="", mm=FALSE, sep="\t", dec=".", transpose=TRUE, ...)
readSNPMaP(file="", sep="\t", dec=".", transpose=TRUE, ...)

```

Arguments

x	object of class SNPMaP.
file	character; name of the file that should be created or appended to. The default, "", causes the output to be written to the console.
mm	logical indicating whether the data in the mismatch slot should be returned instead.

sep	character; the separator to be used between columns in the file.
dec	character; the decimal point to be used.
transpose	logical; should the data frame be transposed for writing to file (or transposed when reading from file)?
...	additional arguments passed to methods.

Details

transpose defaults to TRUE because it is *much* faster to write one (or a few) row(s) per chip than it is to write one row per SNP. It also produces smaller text files. Set transpose=FALSE to preserve the orientation of the matrix. Likewise, readSNPMaP assumes the file was written using the default transpose=TRUE; change this to FALSE to read in an untransposed file.

Value

writeSNPMaP returns TRUE invisibly on success. readSNPMaP returns a named matrix.

See Also

[SNPMaP-package](#)

Examples

```
## Not run:
## Write to a tab-delimited file
writeSNPMaP(x, file='mySNPMaP.dat')
## Write to a csv file
writeSNPMaP(x, file='mySNPMaP.csv', sep=',')
## Write a semicolon delimited file with commas for decimal points
## (Standard in some parts of Western Europe)
writeSNPMaP(x, file='mySNPMaP.csv', sep=';', dec=',')
## Read a tab-delimited file written using writeSNPMaP()
y<-readSNPMaP(file='mySNPMaP.dat')
## End(Not run)
```

Index

*Topic **classes**

SNPMaP-class, 12

*Topic **manip**

cel2ras, 4

cloneSNPMaP, 6

disk2memory, 7

getSNPMaP, 7

minusMismatch, 8

norm, 9

qcMean, 10

snpmap, 10

writeSNPMaP, 16

*Topic **package**

SNPMaP-package, 2

[, SNPMaP, ANY, ANY, missing-method
(SNPMaP-class), 12

[, SNPMaP, ANY, missing, missing-method
(SNPMaP-class), 12

[, SNPMaP, missing, ANY, missing-method
(SNPMaP-class), 12

[, SNPMaP, missing, missing, missing-method
(SNPMaP-class), 12

affxparser-package, 3

as.matrix, SNPMaP-method
(SNPMaP-class), 12

boxplot, SNPMaP-method
(SNPMaP-class), 12

cel2long (cel2ras), 4

cel2ras, 4, 11, 15

cel2rasS (cel2ras), 4

cel2raw, 14

cel2raw (cel2ras), 4

cel2short (cel2ras), 4

cloneSNPMaP, 6, 8, 11, 15

close, SNPMaP-method
(SNPMaP-class), 12

disk2memory, 7, 11, 15

FileDoubleMatrix, 11

FileDoubleMatrix-class
(SNPMaP-class), 12

getSNPMaP, 7

image, SNPMaP-method
(SNPMaP-class), 12

initialize, SNPMaP-method
(SNPMaP-class), 12

logIntensities, 5, 11

logIntensities (norm), 9

long2ras (cel2ras), 4

long2rasS (cel2ras), 4

long2short (cel2ras), 4

memory2disk (disk2memory), 7

methods-package, 3

minusMismatch, 8

msnpmap (snpmap), 10

norm, 5, 9, 11

open, SNPMaP-method
(SNPMaP-class), 12

plot, SNPMaP, missing-method
(SNPMaP-class), 12

plot, SNPMaP-method
(SNPMaP-class), 12

print.summary.SNPMaP
(SNPMaP-class), 12

qcMean, 10, 14

R.huge-package, 3

ras2rasS, 10

ras2rasS (cel2ras), 4

raw2long (cel2ras), 4

raw2ras (cel2ras), 4

raw2rasS (*cel2ras*), 4
raw2short (*cel2ras*), 4
readSNPMaP (*writeSNPMaP*), 16

set, 11
setSNPMaP (*getSNPMaP*), 7
short2ras (*cel2ras*), 4
short2rasS (*cel2ras*), 4
SNPMaP, 4, 5, 7–9, 11
snpmap, 3–5, 10, 15
SNPMaP-class, 3, 5, 6, 8, 11
SNPMaP-package, 5, 7, 9–11, 15, 17
SNPMaP-class, 12
SNPMaP-package, 2
SNPMaP.cdm, 2
SNPMaP.cdm-package, 3, 5, 11, 15
SNPMaP.cdm-package, 14
SNPMaPdata-class (*SNPMaP-class*),
12

summary, SNPMaP-method
(*SNPMaP-class*), 12

unlink, 4, 7–9

writeSNPMaP, 11, 15, 16