Package ‘crlmm’

December 10, 2023

Type  Package
Title  Genotype Calling (CRLMM) and Copy Number Analysis tool for
       Affymetrix SNP 5.0 and 6.0 and Illumina arrays
Version  1.60.0
Author  Benilton S Carvalho, Robert Scharpf, Matt Ritchie, Ingo
        Ruczinski, Rafael A Irizarry
Maintainer  Benilton S Carvalho <benilton@unicamp.br>,
            Robert Scharpf <rscharpf@jhsph.edu>, Matt Ritchie
            <mritchie@wehi.EDU.AU>
Description  Faster implementation of CRLMM specific to SNP 5.0 and 6.0
             arrays, as well as a copy number tool specific to 5.0, 6.0, and
             Illumina platforms.
License  Artistic-2.0
Depends  R (>= 2.14.0), oligoClasses (>= 1.21.12), preprocessCore (>=
         1.17.7)
LinkingTo  preprocessCore (>= 1.17.7)
Imports  methods, Biobase (>= 2.15.4), BiocGenerics, affyio (>=
         1.23.2), illuminaio, ellipse, mvtnorm, splines, stats, utils,
         lattice, ff, foreach, RcppEigen (>= 0.3.1.2.1), matrixStats,
         VGAM, parallel, graphics, limma, beanplot
Suggests  hapmapsnp6, genomewidesnp6Crlmm (>= 1.0.7), snpStats, RUnit
          methods-CNSetLM.R methods-eSet.R methods-SnpSuperSet.R
          methods-PredictionRegion.R cnrma-functions.R cnset-accessors.R
LazyLoad  yes

## Local Variables
## time-stamp-pattern  ```8/Date: %3a %3b %2d %02H:%02M:%02S %Z %:%y
## End
biocViews Microarray, Preprocessing, SNP, CopyNumberVariation
RoxygenNote 7.1.1
git_url https://git.bioconductor.org/packages/crlmm
git_branch RELEASE_3_18
git_last_commit 1271399
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2023-12-10

R topics documented:

- `crlmm-package` .......................................................... 3
- `ABpanel` ................................................................. 3
- `AssayData-methods` .................................................... 4
- `batchStatisticAccessors` ............................................. 5
- `calculateRBFaff` ......................................................... 6
- `cnrmaAffy` .............................................................. 7
- `CNSet-methods` .......................................................... 8
- `cnSetExample` ........................................................... 9
- `constructAffyCNSet` ................................................... 9
- `constructInf` ............................................................. 10
- `copynumberAccessors` ................................................ 12
- `crlmm` ................................................................. 13
- `crlmmCopynumber` ..................................................... 16
- `genotype` ............................................................ 18
- `genotype.Illumina` ................................................... 21
- `genotypeAffy` .......................................................... 25
- `genotypeInf` ............................................................ 26
- `genotypes` .............................................................. 27
- `ListClassConstructors` ............................................... 28
- `plotSNPs` ............................................................. 29
- `posteriorProbability` ................................................ 30
- `predictionRegion` ...................................................... 31
- `PredictionRegion-class` .............................................. 32
- `preprocessInf` ......................................................... 33
- `readGenCallOutput` ................................................... 35
- `readIdatFiles` .......................................................... 36
- `snprma` ................................................................. 37
- `snprmaAffy` ............................................................ 39
- `validCdfNames` ........................................................ 40
- `validCEL` ............................................................... 40
- `xyplot` ................................................................. 41

Index 43
**crlmm-package**

*Genotype Calling via CRLMM Algorithm*

---

**Description**

Faster implementation of CRLMM specific to SNP 5.0 and 6.0 arrays.

**Details**

Index:

<table>
<thead>
<tr>
<th><strong>crlmm-package</strong></th>
<th>New implementation of the CRLMM Algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>crlmm</strong></td>
<td>Genotype SNP 5.0 or 6.0 samples.</td>
</tr>
<tr>
<td><strong>calls</strong></td>
<td>Accessor for genotype calls.</td>
</tr>
<tr>
<td><strong>confs</strong></td>
<td>Accessor for confidences.</td>
</tr>
</tbody>
</table>

The `crlmm` package reimplements the CRLMM algorithm present in the `oligo` package. This implementation primes for efficient genotyping of samples on SNP 5.0 and SNP 6.0 Affymetrix arrays.

To use this package, the user must have additional data packages: `genomewidesnp5Crlmm` - SNP 5.0 arrays `genomewidesnp6Crlmm` - SNP 6.0 arrays

**Author(s)**

Rafael A Irizarry Maintainer: Benilton S Carvalho <carvalho@bclab.org>

**References**


---

**ABpanel**

* A panel function for plotting prediction regions and log-normalized intensities

---

**Description**

A panel function for plotting prediction regions and log-normalized intensities

**Usage**

`ABpanel(x, y, predictRegion, copyNumber = 0:4, fill, ..., subscripts)`
Arguments

- **x**: log-normalized intensities for the A or B allele
- **y**: log-normalized intensities for the A or B allele
- **predictRegion**: A list. See `predictionRegion`
- **copyNumber**: Integer vector. Indicates which prediction regions are drawn.
- **fill**: Character or integer vector for coloring the points. Only valid for certain point symbols. See `points`.
- **...**: Additional arguments to `panel.xyplot` and `lpolygon`.
- **subscripts**: See `xyplot` in the `lattice` package.

Value

Not applicable

Note

`ABpanel` can be passed as the argument to `panel` in the `xyplot` method for `CNSet` objects. See the examples in `xyplot`.

Author(s)

R. Scharpf

See Also

`xyplot`, `panel.xyplot`, `lpolygon`

Description

The `batchStatistics` slot in a `CNSet` object is an instance of the `AssayData` slot. In general, the accessors for `AssayData` are called indirectly by the corresponding method for the `CNSet` class and not called directly by the user.

Methods

- **Ns** signature(object="AssayData") : Accessor for genotype frequencies
- **corr** signature(object="AssayData") : Accessor for the correlation of the log-transformed normalized intensities within the diallelic genotype clusters
- **mads** signature(x="AssayData") : Accessor for the median absolute deviation of the normalized intensities within the diallelic genotype clusters
- **medians** signature(object="AssayData") : Accessor for the posterior mean of the normalized intensity within the diallelic genotype clusters.
- **tau2** signature(object="AssayData") : Accessor for the median absolute deviation of the log-transformed intensities within the diallelic genotype clusters
**batchStatisticAccessors**

**See Also**

CNSet-class, Ns, tau2, corr, mads, medians

---

**batchStatisticAccessors**

*Accessors for batch-specific summary statistics.*

**Description**

The summary statistics stored here are used by the tools for copy number estimation.

**Usage**

```r
corr(object, ...)
tau2(object, ...)
mads(object, ...)
medians(object, ...)
Ns(object, ...)
```

**Arguments**

- `object`: An object of class `CNSet`.
- `...`: Ignored

**Value**

An array with dimension R x A x G x C, or R x G x C.

- `R`: number of markers
- `A`: number of alleles (2)
- `G`: number of biallelic genotypes (3)
- `C`: number of batches

- `Ns` returns an array of genotype frequencies stratified by batch. Dimension R x G x C.
- `corr` returns an array of within-genotype correlations (log2-scale) stratified by batch. Dimension R x G x C.
- `medians` returns an array of the within-genotype medians (intensity-scale) stratified by batch and allele. Dimension is the same as for `medians`.
- `mads` returns an array of the within-genotype median absolute deviations (intensity-scale) stratified by batch and allele. Only the mads for AA and BB genotypes are stored. Dimension is R x A x G x C, where G is AA or BB. Note that the mad for allele A/B for subjects with genotype BB/AA is a robust estimate of the background variance, whereas the mad for allele A/B for subjects with genotype AA/BB is a robust estimate of the variance for copy number greater than 0 (we assume that on the log-scale the variance is roughly constant for CA, CB > 0).
See Also

`batchStatistics`

Examples

data(cnSetExample)
Ns(cnSetExample)[1:5, , ]
corr(cnSetExample)[1:5, , ]
meds <- medians(cnSetExample)
mads(cnSetExample)[1:5, , , ]
tau2(cnSetExample)[1:5, , , ]

calculateRBaf

*Calculate log R ratios and B allele frequencies.*

**Description**

Calculate log R ratios and B allele frequencies from a `CNSet` object

**Usage**

`calculateRBaf(object, batch.name, chrom)`

**Arguments**

- `object`: A `CNSet` object.
- `batch.name`: A character string indicating the batch. If missing, log R ratios and B allele frequencies are calculated for all batches in the object.
- `chrom`: Integer indicating which chromosome to process. If missing, B allele frequencies and log R ratios are calculated for all autosomal chromosomes and chromosome X that are included in object.

**Details**

`batch.name` must be a value in `batch(object)`. Currently, one must specify a single `batch.name`. If a character vector for `batch.name` is supplied, only the first is evaluated.

TODO: A description of how these values are calculated.

**Value**

A named list.

- `baf`: Each element in the `baf` list is a matrix of B allele frequencies (one matrix for each chromosome).
- `lrr`: Each element in the `lrr` list is a matrix of log R ratios (one matrix for each chromosome).

The log R ratios were scaled by a factor of 100 and stored as an integer. B allele frequencies were scaled by a factor of 1000 and stored as an integer.
Author(s)

Lynn Mireless

References

Peiffer et al., High-resolution genomic profiling of chromosomal aberrations using Infinium whole-genome genotyping (2006), Genome Research

Examples

data(cnSetExample)
baf.lrr <- suppressWarnings(calculateRBaf(cnSetExample, "SHELF"))
hist(baf.lrr[["baf"]][[1]]/1000, breaks=100)
hist(baf.lrr[["lrr"]][[1]]/100, breaks=100)
## Not run:
library(ff)
baf.lrr <- suppressWarnings(calculateRBaf(cnSetExample, "SHELF"))
class(baf.lrr[["baf"]][[1]]) ## ff_matrix
class(baf.lrr[["lrr"]][[1]]) ## ff_matrix

## End(Not run)

cnrmaAffy quantile normalize nonpolymorphic markers

Description

Quantile normalize nonpolymorphic markers to hapmap reference distribution

Usage

cnrmaAffy(cnSet, seed = 1, verbose = TRUE)

Arguments

<table>
<thead>
<tr>
<th>cnSet</th>
<th>Object of class CNSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>seed</td>
<td>Random number seed</td>
</tr>
<tr>
<td>verbose</td>
<td>Logical.</td>
</tr>
</tbody>
</table>

Value

Returns logical. Normalized intensities are written to the alleleA ff_matrix stored in the CNSet assayData.

Author(s)

R. Scharpf
CNSet-methods

crlmm methods for class "CNSet"

Description

CNSet is a container defined in the oligoClasses package for storing normalized intensities for genotyping platforms, genotype calls, and parameters estimated for copy number. Accessors for data that an object of this class contains are largely defined in the package oligoClasses. CNSet methods that involve more complex calculations that are specific to the crlmm package, such as computing allele-specific copy number, are included in crlmm and described here.

Methods

as(from, "oligoSnpSet") : Method for coercing object from class CNSet) to an object of class oligoSnpSet.

CA signature(object="CNSet") : calculates raw copy number for allele A

CB signature(object="CNSet") : calculates raw copy number for allele B

lines signature(x="CNSet") : plot ellipses (95th percentile) for prediction regions

totalCopynumber signature(object="CNSet") : calculates total raw copy number

rawCopynumber signature(object="CNSet") : same as totalCopynumber

nuA signature(object="CNSet") : estimate of mean background (intensity-scale) for allele A

nuB signature(object="CNSet") : estimate of mean background (intensity-scale) for allele A

phiA signature(object="CNSet") : estimate of slope coefficient (intensity-scale) for allele A

phiB signature(object="CNSet") : estimate of slope coefficient (intensity-scale) for allele B

Ns signature(object="CNSet") : genotype frequencies

corr signature(object="CNSet") : correlation of log-transformed normalized intensities within the genotype clusters

mads signature(x="CNSet") : ...

medians signature(object="CNSet") : ...

tau2 signature(object="CNSet") : ...

OligoSetList(object) : constructs an object of class OligoSetList from object having class CNSet.

BafLrrSetList(object) : constructs an object of class BafLrrSetList from object having class CNSet.

See Also

CNSet-class, CA, CB, totalCopynumber, rawCopynumber
cnSetExample

Object of class 'CNSet'

Description

The data for the first 16 polymorphic markers in the HapMap analysis.

Usage

data(cnSetExample)
data(cnSetExample2)

Format

The data illustrates the CNSet-class, with assayData containing the quantile-normalized intensities for the A and B alleles, genotype calls and confidence scores.

Details

This object was created from the copynumber vignette in inst/scripts. A subset of markers was selected to keep the package size small.

Examples

data(cnSetExample)
data(cnSetExample2)

constructAffyCNSet

Construct an object of class CNSet from Affymetrix cel files

Description

Construct a container for normalized intensities for Affymetrix cel files, referred to as a CNSet

Usage

constructAffyCNSet(filenames, sns, cdfName, batch, verbose = TRUE, genome)

Arguments

filenames Vector of cel file names.
sns Sample identifiers. Defaults to basename(filenames).
cdfName Character string indicating annotation package (e.g., "genomewidesnp6Crlmm")
batch Vector of same length as filenames indicating batch.
verbose Logical.
genome Character string indicating UCSC genome build (hg18 or hg19 supported)
constructInf

Value

An object of class CNSet

Author(s)

R. Scharpf

---

constructInf

Instantiate an object of class CNSet for the Infinium platforms.

Description

Instantiates an object of class CNSet for the Infinium platforms. Elements of assayData and batchStatistics will be ff objects. See details.

Usage

constructInf(sampleSheet = NULL, arrayNames = NULL, path = ".", arrayInfoColNames = list(barcode="SentrixBarcode_A", position="SentrixPosition_A"), highDensity = FALSE, sep = ",", fileExt = list(green = "Grn.idat", red = "Red.idat"), XY, cdfName, anno, genome, verbose = FALSE, batch=NULL, saveDate = TRUE)

Arguments

sampleSheet data.frame containing Illumina sample sheet information (for required columns, refer to BeadStudio Genotyping guide - Appendix A).

arrayNames character vector containing names of arrays to be read in. If NULL, all arrays that can be found in the specified working directory will be read in.

path character string specifying the location of files to be read by the function

arrayInfoColNames (used when sampleSheet is specified) list containing elements 'barcode' which indicates column names in the sampleSheet which contains the arrayNumber/barcode number and 'position' which indicates the strip number. In older style sample sheets, this information is combined (usually in a column named 'SentrixPosition') and this should be specified as list(barcode=NULL, position="SentrixPosition")

highDensity logical (used when sampleSheet is specified). If TRUE, array extensions '_A', '_B' in sampleSheet are replaced with 'R01C01', 'R01C02' etc.

sep character string specifying separator used in .idat file names.

fileExt list containing elements 'Green' and 'Red' which specify the .idat file extension for the Cy3 and Cy5 channels.

XY an NChannelSet containing X and Y intensities.

cdfName annotation package (see also validCdfNames) or 'nopackage' when an anno data.frame and genome supplied

anno data.frame containing SNP annotation information from manifest and additional columns 'isSnp', 'position', 'chromosome' and 'featureNames'. For use when cdfName='nopackage'
constructInf

Genome character string specifying which genome is used in annotation
Verbose 'logical.' Whether to print descriptive messages during processing.
Batch batch variable. See details.
Save date 'logical'. Should the dates from each .idat be saved with sample information?

Details

This function initializes a container for storing the normalized intensities for the A and B alleles at polymorphic loci and the normalized intensities for the 'A' allele at nonpolymorphic loci. CRLMM genotype calls and confidence scores are also stored in the assayData. This function does not do any preprocessing or genotyping – it only creates an object of the appropriate size. The initialized values will all be 'NA'.

The ff package provides infrastructure for accessing and writing data to disk instead of keeping data in memory. Each element of the assayData and batchStatistics slot are ff objects. ff objects in the R workspace contain pointers to several files with the '.ff' extension on disk. The location of where the data is stored on disk can be specified by use of the ldPath function. Users should not move or rename this directory. If only output files are stored in ldPath, one can either remove the entire directory prior to rerunning the analysis or all of the '.ff' files. Otherwise, one would accumulate a large number of '.ff' files on disk that are no longer in use.

We have adopted the ff package in order to reduce crlmm’s memory footprint. The memory usage can be fine-tuned by the utilities ocSamples and ocProbesets provided in the oligoClasses package. In most instances, the user-level interface will be no different than accessing data from ordinary matrices in R. However, the differences in the underlying representation can become more noticeable for very large datasets in which the I/O for accessing data from the disk can be substantial.

Value

A CNSet object

Author(s)

R. Scharpf

See Also

ldPath, ocSamples, ocProbesets, CNSet-class, preprocessInf, genotypeInf

Examples

```r
## See the Illumina vignettes in inst/scripts of the
## source package for an example
```
**Description**

These methods can be applied after an object of class CNSet has been generated by the `crlmmCopynumber` function.

**Usage**

- `CA(object, ...)`
- `CB(object, ...)`
- `nuA(object)`
- `nuB(object)`
- `phiA(object)`
- `phiB(object)`
- `totalCopynumber(object, ...)`
- `rawCopynumber(object, ...)`

**Arguments**

- `object` An object of class CNSet.
- `...` An additional argument named `i` can be passed to subset the markers and an argument `j` can be passed to subset the samples. Other arguments are ignored.

**Details**

At polymorphic markers, `nuA` and `nuB` provide the intercept coefficient (the estimated background intensity) for the A and B alleles, respectively. `phiA` and `phiB` provide the slope coefficients for the A and B alleles, respectively.

At nonpolymorphic markers, `nuB` and `phiB` are ‘NA’.

These functions can be used to translate the normalized intensities to the copy number scale. Plotting the copy number estimates as a function of physical position can be used to guide downstream algorithms that smooth, as well as to assess possible mosaicism.

**Value**

- `nu[A/B]` and `phi[A/B]` return matrices of the intercept and slope coefficients, respectively.
- `CA` and `CB` return matrices of allele-specific copy number.
- `totalCopynumber` (or `rawCopynumber`) returns a matrix of `CA+CB`.

**Note**

Subsetting the CNSet object before extracting copy number can be very inefficient when the data set is very large, particularly if using ff objects. The `[` method will subset all of the assay data elements and all of the elements in the LinearModelParameter slot.
crlmm

Genotype oligonucleotide arrays with CRLMM

Description

This is a faster and more efficient implementation of the CRLMM algorithm, especially designed for Affymetrix SNP 5 and 6 arrays (to be soon extended to other platforms).
Usage

crlmm(filenames, row.names=TRUE, col.names=TRUE, 
probs=c(1/3, 1/3, 1/3), DF=6, SNRMin=5, 
gender=NULL, save.it=FALSE, load.it=FALSE, 
intensityFile, mixtureSampleSize=10^5, 
eps=0.1, verbose=TRUE, cdfName, sns, recallMin=10, 
recallRegMin=1000, returnParams=FALSE, badSNP=0.7)
crlmm2(filenames, row.names=TRUE, col.names=TRUE, 
probs=c(1/3, 1/3, 1/3), DF=6, SNRMin=5, 
gender=NULL, save.it=FALSE, load.it=FALSE, 
intensityFile, mixtureSampleSize=10^5, 
eps=0.1, verbose=TRUE, cdfName, sns, recallMin=10, 
recallRegMin=1000, returnParams=FALSE, badSNP=0.7)

Arguments

filenames  'character' vector with CEL files to be genotyped.
row.names  'logical'. Use rownames - SNP names?
col.names  'logical'. Use colnames - Sample names?
probs  'numeric' vector with priors for AA, AB and BB.
DF  'integer' with number of degrees of freedom to use with t-distribution.
SNRMin  'numeric' scalar defining the minimum SNR used to filter out samples.
gender  'integer' vector, with same length as 'filenames', defining sex. (1 - male; 2 - female)
save.it  'logical'. Save preprocessed data?
load.it  'logical'. Load preprocessed data to speed up analysis?
intensityFile  'character' with filename to be saved/loaded - preprocessed data.
mixtureSampleSize  Number of SNP's to be used with the mixture model.
eps  Minimum change for mixture model.
verbose  'logical'.
cdfName  'character' defining the CDF name to use ('GenomeWideSnp5', 'GenomeWideSnp6')
sns  'character' vector with sample names to be used.
recallMin  Minimum number of samples for recalibration.
recallRegMin  Minimum number of SNP's for regression.
returnParams  'logical'. Return recalibrated parameters.
badSNP  'numeric'. Threshold to flag as bad SNP (affects batchQC)

Details

'crlmm2' allows one to genotype very large datasets (via ff package) and also permits the use of clusters or multiple cores (via snow package) to speed up genotyping.

As noted above, the call probabilities are stored using an integer representation to reduce file size using the transformation \(\text{round}(-1000*\log2(1-p))\), where \(p\) is the probability. The function \(\text{id2P}\) can be used to convert the integers back to the scale of probabilities.
crlmm

Value

A SnpSet object.

calls            Genotype calls (1 - AA, 2 - AB, 3 - BB)
confs            Confidence scores (round(-1000*log2(1-p)))
SNPQC            SNP Quality Scores
batchQC          Batch Quality Score
params           Recalibrated parameters

References


See Also

i2p, snpCall, snpCallProbability

Examples

## this can be slow
library(oligoClasses)
if (require(genomewidessnp6Crlmm) & require(hapmapsnp6)){
  path <- system.file("celFiles", package="hapmapsnp6")

  ## the filenames with full path...
  ## very useful when genotyping samples not in the working directory
  cels <- list.celfiles(path, full.names=TRUE)
  (crlmmOutput <- crlmm(cels))
  ## If gender is known, one should check that the assigned gender is
  ## correct, or pass the integer coding of gender as an argument to the
  ## crlmm function as done below
}

## Not run:
## HPC Example
library(ff)
library(snow)
library(crlmm)
## genotype 50K SNPs at a time
ocProbesets(50000)
## setup cluster - 8 cores on the machine
library(doSNOW)
c1 <- makeCluster(8, "SOCK")
registerDoSNOW(c1)
##setCluster(8, "SOCK")
path <- system.file("celFiles", package="hapmapsnp6")
cels <- list.celfiles(path, full.names=TRUE)
crlmmOutput <- crlmm2(cels)

## End(Not run)

crlmmCopynumber

Locus- and allele-specific estimation of copy number

Description

Locus- and allele-specific estimation of copy number.

Usage

crlmmCopynumber(object, MIN.SAMPLES=10, SNRMin = 5, MIN.OBS = 1,
DF.PRIOR = 50, bias.adj = FALSE,
prior.prob = rep(1/4, 4), seed = 1, verbose = TRUE,
GT.CONF.THR = 0.80, MIN.NU = 2^3, MIN.PHI = 2^3,
THR.NU.PHI = TRUE, type=c("SNP", "NP", "X.SNP", "X.NP"),
fit.linearModel=TRUE)

Arguments

object object of class CNSet.
MIN.SAMPLES 'Integer'. The minimum number of samples in a batch. Batches with fewer than MIN.SAMPLES are skipped. Therefore, samples in batches with fewer than MIN.SAMPLES have NA's for the allele-specific copy number and NA's for the linear model parameters.
SNRMin Samples with low signal to noise ratios are excluded.
MIN.OBS For a SNP with with fewer than MIN.OBS of a genotype in a given batch, the within-genotype median is imputed. The imputation is based on a regression using SNPs for which all three biallelic genotypes are observed. For example, assume at at a given SNP genotypes AA and AB were observed and BB is an unobserved genotype. For SNPs in which all 3 genotypes were observed, we fit the model E(mean_BB) = beta0 + beta1*mean_AA + beta2*mean_AB, obtaining estimates; of beta0, beta1, and beta2. The imputed mean at the SNP with unobserved BB is then beta0hat + beta1hat * mean_AA of beta2hat * mean_AB.
DF.PRIOR The 2 x 2 covariance matrix of the background and signal variances is estimated from the data at each locus. This matrix is then smoothed towards a common matrix estimated from all of the loci. DF.PRIOR controls the amount of smoothing towards the common matrix, with higher values corresponding to greater smoothing. Currently, DF.PRIOR is not estimated from the data. Future versions may estimate DF.PRIOR empirically.
bias.adj

bias.adj is currently ignored (as well as the prior.prob argument). We plan to add this feature back to the crlmm package in the near future. This feature, when TRUE, updated initial estimates from the linear model after excluding samples with a low posterior probability of normal copy number. Excluding samples that have a low posterior probability can be helpful at loci in which a substantial fraction of the samples have a copy number alteration. For additional information, see Scharpf et al., 2010.

prior.prob

This argument is currently ignored. A numerical vector providing prior probabilities for copy number states corresponding to homozygous deletion, hemizygous deletion, normal copy number, and amplification, respectively.

seed

Seed for random number generation.

verbose

Logical.

GT.CONF.THR

Confidence threshold for genotype calls (0, 1). Calls with confidence scores below this threshold are not used to estimate the within-genotype medians. See Carvalho et al., 2007 for information regarding confidence scores of biallelic genotypes.

MIN.NU

numeric. Minimum value for background intensity. Ignored if THR.NU.PHI is FALSE.

MIN.PHI

numeric. Minimum value for slope. Ignored if THR.NU.PHI is FALSE.

THR.NU.PHI

If THR.NU.PHI is FALSE, MIN.NU and MIN.PHI are ignored. When TRUE, background (nu) and slope (phi) coefficients below MIN.NU and MIN.PHI are set to MIN.NU and MIN.PHI, respectively.

type

Character string vector that must be one or more of "SNP", "NP", "X.SNP", or "X.NP". Type refers to a set of markers. See details below

fit.linearModel

Logical. If TRUE, a linear model is fit to estimate the parameters for computing the absolute copy number. If FALSE, we compute the batch-specific, within-genotype median and MAD at polymorphic loci and the median and MAD at nonpolymorphic loci.

Details

We suggest a minimum of 10 samples per batch for using crlmmCopynumber. 50 or more samples per batch is preferred and will improve the estimates.

The functions crlmmCopynumberLD and crlmmCopynumber2 have been deprecated.

The argument type can be used to specify a subset of markers for which the copy number estimation algorithm is run. One or more of the following possible entries are valid: 'SNP', 'NP', 'X.SNP', and 'X.NP'.

'SNP' referers to autosomal SNPs.

'NP' refers to autosomal nonpolymorphic markers.

'X.SNP' refers to SNPs on chromosome X.

'X.NP' refers to autosomes on chromosome X.

However, users must run 'SNP' prior to running 'NP' and 'X.NP', or specify type = c('SNP', 'X.NP').
Value

The value returned by the `crlmmCopynumber` function depends on whether the data is stored in RAM or whether the data is stored on disk using the R package `ff` for reading / writing. If uncertain, the first line of the `show` method defined for `CNSet` objects prints whether the `assayData` elements are derived from the `ff` package in the first line. Specifically,

- if the elements of the `batchStatistics` slot in the `CNSet` object have the class "ff_matrix" or "ffdf", then the `crlmmCopynumber` function updates the data stored on disk and returns the value `TRUE`.

- if the elements of the `batchStatistics` slot in the `CNSet` object have the class 'matrix', then the `crlmmCopynumber` function returns an object of class `CNSet` with the elements of `batchStatistics` updated.

Author(s)

R. Scharpf

References


Description

Preprocessing and genotyping of Affymetrix arrays.

Usage

```r
genotype(filenames, cdfName, batch, mixtureSampleSize = 10^5, eps = 0.1, 
        verbose = TRUE, seed = 1, sns, probs = rep(1/3, 3),
        DF = 6, SNRMin = 5, recallMin = 10, recallRegMin = 1000,
        gender = NULL, returnParams = TRUE, badSNP = 0.7, 
        genome=c("hg19", "hg18"))
```

Arguments

- `filenames`: complete path to CEL files
- `cdfName`: annotation package (see also `validCdfNames`)
- `batch`: vector of class `character` denoting the batch for each sample in `filenames`. The batch vector must be the same length as the number of samples. See details.
mixtureSampleSize
Sample size to be use when fitting the mixture model.

eps
Stop criteria.

verbose
Logical. Whether to print descriptive messages during processing.

seed
Seed to be used when sampling. Useful for reproducibility

sns
The sample identifiers. If missing, the default sample names are basename(filenames)

probs
‘numeric’ vector with priors for AA, AB and BB.

DF
‘integer’ with number of degrees of freedom to use with t-distribution.

SNRMin
‘numeric’ scalar defining the minimum SNR used to filter out samples.

recallMin
Minimum number of samples for recalibration.

recallRegMin
Minimum number of SNP’s for regression.

gender
integer vector ( male = 1, female =2 ) or missing, with same length as filenames. If missing, the gender is predicted.

returnParams
‘logical’. Return recalibrated parameters from crlmm.

badSNP
‘numeric’. Threshold to flag as bad SNP (affects batchQC)

genoce
character string indicating the UCSC genome build for the SNP annotation

Details

For large datasets it is important to utilize the large data support by installing and loading the ff package before calling the genotype function. In previous versions of the crlmm package, we used different functions for genotyping depending on whether the ff package is loaded, namely genotype and genotype2. The genotype function now handles both instances.

genotype is essentially a wrapper of the crlmm function for genotyping. Differences include (1) that the copy number probes (if present) are also quantile-normalized and (2) the class of object returned by this function, CNSet, is needed for subsequent copy number estimation. Note that the batch variable that must be passed to this function has no effect on the normalization or genotyping steps. Rather, batch is required in order to initialize a CNSet container with the appropriate dimensions and is used directly when estimating copy number.

Value

A SnpSuperSet instance.

Note

For large datasets, load the ’ff’ package prior to genotyping – this will greatly reduce the RAM required for big jobs. See ldPath and ocSamples.

Author(s)

R. Scharpf
References


See Also

snprma, crlmm, ocSamples, ldOpts, batch, crlmmCopynumber

Examples

if (require(ff) & require(genomewidesnp6Crlmm) & require(hapmapsnp6)){
  ldPath(tempdir())
  path <- system.file("celFiles", package="hapmapsnp6")
  ## the filenames with full path...
  ## very useful when genotyping samples not in the working directory
  cels <- list.celfiles(path, full.names=TRUE)
  ## Note: one would need at least 10 CEL files for copy number estimation
  ## To use less RAM, specify a smaller argument to ocProbesets
  ocProbesets(50e3)
  batch <- rep("A", length(cels))
  (cnSet <- genotype(cels, cdfName="genomewidesnp6", batch=batch))

  ##Segment faults that occur with the above step can often be traced to a
  ##corrupt cel file. To check if any of the files are corrupt, try
  ##reading the files in one at a time:

  ## Not run:
  require(affyio)
  validCEL(cels)

  ## End(Not run)

  ## when gender is not specified (as in the above example), crlmm tries
  ## to predict the gender from SNPs on chromosome X
  cnSet$gender

  ## If gender is known, one should check that the assigned gender is
  ## correct. Alternatively, one can pass gender as an argument to the
  ## genotype function.
  gender <- c("female", "female", "male")
  gender[gender == "female"] <- 2
  gender[gender == "male"] <- 1
  dim(cnSet)
  table(isSnp(cnSet))
}
Preprocessing and genotyping of Illumina Infinium II arrays.

Usage

```r
genotype.Illumina(sampleSheet=NULL, arrayNames=NULL, ids=NULL, path=".",
arrayInfoColNames=list(barcode="SentrixBarcode_A", position="SentrixPosition_A"),
highDensity=FALSE, sep=" ", fileExt=list(green="Grn.idat", red="Red.idat"), XY=NULL, anno, genome,
call.method="crlmm", trueCalls=NULL, cpn=TRUE, batch=NULL, saveDate=FALSE, stripNorm=TRUE,
copynumber=TRUE, batch=NULL, saveDate=FALSE, stripNorm=TRUE, useTarget=TRUE, quantile.method="between", nopackage.norm="quantile", mixtureSampleSize=10^5, fitMixture=FALSE,
eps=0.1, verbose = TRUE, seed = 1, sns, probs = rep(1/3, 3), DF = 6, SNRMin = 5,
recallMin = 10, recallRegMin = 1000, gender = NULL, returnParams = TRUE, badSNP = 0.7)
crlmmIllumina(sampleSheet=NULL, arrayNames=NULL, ids=NULL, path=".",
arrayInfoColNames=list(barcode="SentrixBarcode_A", position="SentrixPosition_A"),
highDensity=FALSE, sep=" ", fileExt=list(green="Grn.idat", red="Red.idat"), XY=NULL, anno, genome,
call.method="crlmm", trueCalls=NULL, cpn=TRUE, batch=NULL, saveDate=FALSE, stripNorm=TRUE,
copynumber=TRUE, batch=NULL, saveDate=FALSE, stripNorm=TRUE, useTarget=TRUE, quantile.method="between", nopackage.norm="quantile", mixtureSampleSize=10^5, fitMixture=FALSE,
eps=0.1, verbose = TRUE, seed = 1, sns, probs = rep(1/3, 3), DF = 6, SNRMin = 5,
recallMin = 10, recallRegMin = 1000, gender = NULL, returnParams = TRUE, badSNP = 0.7)
```

Arguments

- **sampleSheet** data.frame containing Illumina sample sheet information (for required columns, refer to BeadStudio Genotyping guide - Appendix A).
- **arrayNames** character vector containing names of arrays to be read in. If NULL, all arrays that can be found in the specified working directory will be read in.
- **ids** vector containing ids of probes to be read in. If NULL all probes found on the first array are read in.
- **path** character string specifying the location of files to be read by the function.
- **arrayInfoColNames** (used when sampleSheet is specified) list containing elements 'barcode' which indicates column names in the sampleSheet which contains the arrayNumber/barcode number and 'position' which indicates the strip number. In older style sample sheets, this information is combined (usually in a column named 'SentrixPosition') and this should be specified as list(barcode=NULL, position="SentrixPosition")
- **highDensity** logical (used when sampleSheet is specified). If TRUE, array extensions '_A', '_B' in sampleSheet are replaced with 'R01C01', 'R01C02' etc.
- **sep** character string specifying separator used in .idat file names.
fileExt

list containing elements 'Green' and 'Red' which specify the .idat file extension for the Cy3 and Cy5 channels.

XY

NChannelSet containing X and Y intensities.

anno

data.frame containing SNP annotation information from manifest and additional columns 'isSnp', 'position', 'chromosome' and 'featureNames'. For use when cdfName='nopackage'

genome

character string specifying which genome is used in annotation

call.method

character string specifying the genotype calling algorithm to use ('crlmm' or 'krlmm').

trueCalls

matrix specifying known Genotype calls (can contain some NAs) for a subset of samples and features (1 - AA, 2 - AB, 3 - BB).

cdfName

annotation package (see also validCdfNames) or 'nopackage' when combined with 'krlmm', an anno data.frame and genome.

copynumber

'logical'. Whether to store copy number intensities with SNP output.

batch

character vector indicating the batch variable. Must be the same length as the number of samples. See details.

saveDate

'logical'. Should the dates from each .idat be saved with sample information?

stripNorm

'logical'. Should the data be strip-level normalized?

useTarget

'logical' (only used when stripNorm=TRUE). Should the reference HapMap intensities be used in strip-level normalization?

quantile.method

character string specifying the quantile normalization method to use ('within' or 'between' channels).

nopackage.norm

character string specifying normalization to be used when cdfName='nopackage'. Options are 'none', 'quantile' (within channel, between array) and 'loess'.

mixtureSampleSize

Sample size to be use when fitting the mixture model.

fitMixture

'logical'. Whether to fit per-array mixture model.

eps

Stop criteria.

verbose

'logical'. Whether to print descriptive messages during processing.

seed

Seed to be used when sampling. Useful for reproducibility

sns

The sample identifiers. If missing, the default sample names are basename(filenames)

probs

'numeric' vector with priors for AA, AB and BB.

DF

'integer' with number of degrees of freedom to use with t-distribution.

SNRMin

'numeric' scalar defining the minimum SNR used to filter out samples.

recallMin

Minimum number of samples for recalibration.

recallRegMin

Minimum number of SNP’s for regression.

gender

integer vector ( male = 1, female = 2 ) or missing, with same length as filenames. If missing, the gender is predicted.

returnParams

'logical'. Return recalibrated parameters from crlmm.

badSNP

'numeric'. Threshold to flag as bad SNP (affects batchQC)
Details

genotype.Illumina (or equivalently crlmmIllumina) is a wrapper of the crlmm function for genotyping. Differences include (1) that the copy number probes (if present) are also quantile-normalized and (2) the class of object returned by this function, CNSet, is needed for subsequent copy number estimation. Note that the batch variable (a character string) has no effect on the normalization or genotyping steps. Rather, batch is required in order to initialize a CNSet container with the appropriate dimensions.

The new 'krlmm' option is available for certain chip types. Optional argument trueCalls matrix contains known Genotype calls (1 - AA, 2 - AB, 3 - BB) for a subset of samples and features. This will used to compute KRLMM coefficients by calling vglm function from VGAM package.

The 'krlmm' method makes use of functions provided in parallel package to speed up the process. It by default initialises up to 8 clusters. This is configurable by setting up an option named "krlmm.cores", e.g. options("krlmm.cores" = 16).

In general, a chip specific annotation package is required to use the genotype.Illumina function. If this is not available (newer chip types or custom chips often don’t have a chip-specific package available on Bioconductor), consider using cdfName = 'nopackage' and specifying anno and genome, which runs 'krlmm' on the samples available. Here anno is a data.frame read in from the relevant chip-specific manifest, which must have additional columns 'isSnp' which is a logical that indicates whether a probe is polymorphic or not, 'position', 'chromosome' and 'featureNames' that give the location on the chromosome and SNP name.

Value

A SnpSuperSet instance.

Author(s)

Matt Ritchie, Cynthia Liu, Zhiyin Dai

References


See Also

ocSamples, ldOpts
Examples

## Not run:
# example for 'crlmm' option
library(ff)
library(crlmm)
## to enable parallellization, set to TRUE
if(FALSE){
  library(snow)
  library(doSNOW)
  ## with 10 workers
  cl <- makeCluster(10, type="SOCK")
  registerDoSNOW(cl)
}
## path to idat files
datadir <- "/thumper/ctsa/snpmicroarray/illumina/IDATS/370k"
## read in your samplesheet
samplesheet = read.csv(file.path(datadir, "HumanHap370Duo_Sample_Map.csv"), header=TRUE, as.is=TRUE)
samplesheet <- samplesheet[-c(28:46,61:75,78:79), 
arrayNames <- file.path(datadir, unique(samplesheet[, "SentrixPosition"]))
arrayInfo <- list(barcode=NULL, position="SentrixPosition")
cnSet <- genotype.Illumina(sampleSheet=samplesheet,
arrayNames=arrayNames,
arrayInfoColNames=arrayInfo,
cdfName="human370v1c",
batch=rep("1", nrow(samplesheet)))

## End(Not run)
## Not run:
# example for 'krlmm' option
library(crlmm)
library(ff)
# line below is an optional step for krlmm to initialise 16 workers
# options("krlmm.cores" = 16)
# read in raw X and Y intensities output by GenomeStudio's GenCall genotyping module
XY = readGenCallOutput(c("HumanOmni2-5_4v1_FinalReport_83TUSCAN.csv","HumanOmni2-5_4v1_FinalReport_88CHB-JPT.csv"),
cdfName="humanomni25quadv1b",
verbose=TRUE)
krlmmResult = genotype.Illumina(XY=XY,
cdfName=ThiscdfName,
call.method="krlmm",
verbose=TRUE)

# example for 'krlmm' option with known genotype call for some SNPs and samples
library(VGAM)
hapmapCalls = load("hapmapCalls.rda")
# hapmapCalls should have rownames and colnames corresponding to XY featureNames and sampleNames
krlmmResult = genotype.Illumina(XY=XY,
cdfName=ThiscdfName,
call.method="krlmm",
trueCalls=hapmapCalls,
verbose=TRUE)
genotypeAffy

## End(Not run)

---

**genotypeAffy**

*Genotype Affymetrix CEL files*

### Description

Assign diallelic genotypes at polymorphic markers

### Usage

```r
genotypeAffy(cnSet, SNRMin = 5, recallMin = 10, recallRegMin = 1000, gender = NULL, badSNP = 0.7, returnParams = TRUE, verbose = TRUE)
```

#### Arguments

- **cnSet**: An object of class CNSet
- **SNRMin**: See `crlmm`
- **recallMin**: See `crlmm`
- **recallRegMin**: See `crlmm`
- **gender**: See `crlmm`
- **badSNP**: See `crlmm`
- **returnParams**: See `crlmm`
- **verbose**: Logical.

### Details

Wrapper for `crlmm` genotyping.

### Value

Returns logical. SNP genotypes and confidence scores are written to `ff_matrix` objects.

### Author(s)

R.Scharpf

### See Also

- `crlmm.calls.conf`

Genotyping of Illumina Infinium II arrays. This function provides CRLMM/KRLMM genotypes and confidence scores for the polymorphic markers and is a required step prior to copy number estimation.

Usage

```r
genotypeInf(cnSet, mixtureParams, probs = rep(1/3, 3), SNRMin = 5, recallMin = 10, recallRegMin = 1000, verbose = TRUE, returnParams = TRUE, badSNP = 0.7, gender = NULL, DF = 6, cdfName, nopackage.norm="quantile", call.method="crlmm", trueCalls = NULL)
```

Arguments

- `cnSet`: An object of class `CNSet`
- `mixtureParams`: data.frame containing mixture model parameters needed for genotyping. The mixture model parameters are estimated from the `preprocessInf` function.
- `probs`: 'numeric' vector with priors for AA, AB and BB.
- `SNRMin`: 'numeric' scalar defining the minimum SNR used to filter out samples.
- `recallMin`: Minimum number of samples for recalibration.
- `recallRegMin`: Minimum number of SNP’s for regression.
- `verbose`: 'logical.' Whether to print descriptive messages during processing.
- `returnParams`: 'logical'. Return recalibrated parameters from `crlmm`.
- `badSNP`: 'numeric'. Threshold to flag as bad SNP (affects batchQC).
- `gender`: integer vector ( male = 1, female =2 ) or missing, with same length as filenames. If missing, the gender is predicted.
- `DF`: 'integer' with number of degrees of freedom to use with t-distribution.
- `cdfName`: character string indicating which annotation package to load.
- `nopackage.norm`: character string specifying normalization to be used when `cdfName='nopackage'`. Options are 'none', 'quantile' (within channel, between array) and 'quantileloess'.
- `call.method`: character string specifying the genotype calling algorithm to use ('crlmm' or 'krlmm').
- `trueCalls`: matrix specifying known Genotype calls for a subset of samples and features(1 - AA, 2 - AB, 3 - BB).
Details

The genotype calls and confidence scores are written to file using ff protocols for I/O. For the most part, the calls and confidence scores can be accessed as though the data is in memory through the methods snpCall and snpCallProbability, respectively.

The genotype calls are stored using an integer representation: 1 - AA, 2 - AB, 3 - BB. Similarly, the call probabilities are stored using an integer representation to reduce file size using the transformation `round(-1000*log2(1-p))`, where p is the probability. The function i2P can be used to convert the integers back to the scale of probabilities.

An optional trueCalls argument can be provided to KRLMM method which contains known genotype calls (can contain some NAs) for some samples and SNPs. This will used to compute KRLMM parameters by calling vglm function from VGAM package.

The KRLMM method makes use of functions provided in parallel package to speed up the process. It by default initialises up to 8 clusters. This is configurable by setting up an option named "krlmm.cores", e.g. options("krlmm.cores" = 16).

Value

Logical. If the genotyping is completed, the value ‘TRUE’ is returned. Note that assayData elements 'call' and 'callProbability' are updated on disk. Therefore, the genotypes and confidence scores can be retrieved using accessors for the CNSet class.

Author(s)

R. Scharpf

See Also

crlmm, snpCall, snpCallProbability, annotationPackages

Examples

```r
## See the 'illumina_copynumber' vignette in inst/scripts of
## the source package
```

```r
library(crlmm)
library(parallel)

# Example data
data(illumina_copynumber)

# Genotype the data
genotypes <- genotypes(illumina_copynumber, is.snp=TRUE)
```

Description

The possible genotypes for an integer copy number (0-4).

Usage

```r
genotypes(copyNumber, is.snp=TRUE)
```
Arguments

- copyNumber: Integer (0-4 allowed).
- is.snp: Logical. If TRUE, possible genotypes for a polymorphic SNP is returned. If FALSE, only monomorphic genotypes returned.

Value

Character vector.

Author(s)

R. Scharpf

Examples

```r
for(i in 0:4) print(genotypes(i))
for(i in 0:4) print(genotypes(i, FALSE))
```

Description

Constructors for BafLrrSetList and OligoSetList objects.

Usage

```r
BafLrrSetList(object, ...)
OligoSetList(object, ...)
```

Arguments

- object: A CNSet object.
- ...: Additional arguments batch.name and chrom can be used to specify specific batches or chromosomes in the CNSet object.

Details

Constructs a BafLrrSetList object or a OligoSetList object from an object of class CNSet.

Value

A BafLrrSetList or OligoSetList

See Also

BeadStudioSetList
plotSNPs

Examples

data(cnSetExample)
oligoList <- OligoSetList(cnSetExample)
## only contains 1 chromosome, so list only has one element
dims(oligoList)
brList <- BafLrrSetList(cnSetExample)
dims(brList)

plotSNPs

Make M vs S plot for SNPs or samples.

Description

These functions plot the M-values (log-ratios) versus S-values (average intensities) for given SNP(s) or sample(s) or beanplots for M-values from different samples.

Usage

plotSNPs(cnSet, row=1, offset=0, xlim=c(9,16), ylim=c(-5,5), verbose=FALSE)
plotSamples(cnSet, col=1, offset=0, xlim=c(9,16), ylim=c(-5,5), verbose=FALSE, sample=100000, seed=1, type="smoothScatter")

Arguments

- cnSet: An object of class CNSet
- row: scalar/vector of SNP indexes to plot
- col: scalar/vector of sample indexes to plot
- offset: numeric, offset to add to intensities in cnSet before log2-transforming to make log-ratios or average log-intensities
- xlim: the x limits of the plot
- ylim: the y limits of the plot
- verbose: 'logical.' Whether to print descriptive messages during processing
- sample: integer indicating the number of SNPs to sample for the plot
- seed: integer seed for the random number generator to sample the SNPs
- type: character vector specifying the type of sample plot (either 'smoothScatter' or 'beanplot')

Details

The plotSNPs and plotSamples functions plot the M and S values derived from the cnSet object.

Value

One or more M vs S plot for plotSNPs for a given SNP(s) or either a smoothed scatter plot of M vs S or a beanplot of the M-values for a selected sample(s) for plotSamples.
posteriorProbability

Calculate the posterior probability for integer copy numbers.

Description

Calculate the posterior probability for integer copy numbers using the bivariate normal prediction regions.

Usage

posteriorProbability(object, predictRegion, copyNumber = 0:4, w)

Arguments

object A CNSet object.
predictRegion A list containing the bivariate normal prediction region for each of the possible genotypes.
copyNumber Integer vector.
w numeric vector of prior probabilities for each of the copy number states. Must be the same length as copyNumber and sum to 1.

Details

This is currently under development.

Value

An array (features x samples x copy number)
Note
This is under development. Use at your own risk.

Author(s)
R. Scharpf

See Also
predictionRegion, genotypes

Examples
data(cnSetExample)
pr <- predictionRegion(cnSetExample, copyNumber=0:4)
pp <- posteriorProbability(cnSetExample, predictRegion=pr)
dim(pp)

## multiple batches
data(cnSetExample2)
pr <- predictionRegion(cnSetExample2, copyNumber=0:4)
pp <- posteriorProbability(cnSetExample2, predictRegion=pr)

---

**predictionRegion**

Prediction regions for integer copy number

Description
Bivariate normal prediction regions for integer copy number. Copy numbers 0-4 allowed.

Usage
predictionRegion(object, copyNumber)

Arguments
- **object**: A CNSet object.
- **copyNumber**: Integer vector. 0-4 allowed.

Details
We fit a linear regression for each allele to the diallic genotype cluster medians. Denoting the background and slope by nu and phi, respectively, the mean for the bivariate normal prediction region is given by

\[
\mu_A = \nu_A + CA \times \phi_A
\]

and

\[
\mu_B = \nu_B + CB \times \phi_B
\]
The variance and correlation of the normalized intensities is estimated from the diallelic genotype clusters AA, AB, and BB on the log-scale. For copy number not equal to two, we assume that the variance is approximately the same for copy number not equal to 2.

Value

A list named by the genotype. ‘NULL’ refers to copy number zero, ‘A’ is a hemizygous deletion, etc. Each element is a list of the means (mu) and covariance (cov) for each marker stored as an array. For ‘mu’, the dimensions of the array are marker x allele (A or B) x batch. For ‘cov’, the dimensions of the array are marker x 3 (varA, cor, and varB) x batch.

Author(s)

R. Scharpf

References

Scharpf et al., 2011, Biostatistics.

See Also

posteriorProbability, genotypes

Examples

data(cnSetExample)
pr <- predictionRegion(cnSetExample, copyNumber=0:4)
names(pr)
## bivariate normal prediction region for NULL genotype (homozygous deletion)
str(pr["NULL"])
Extends

Class "list", from data part. Class "vector", by class "list", distance 2. Class "AssayData", by class "list", distance 2. Class "list_or_ffdf", by class "list", distance 2. Class vectorORfactor, by class "list", distance 3.

Methods

[ signature(x = "PredictionRegion"): ... Prediction regions can be subset by markers.

Author(s)

R. Scharpf

See Also

predictionRegion

Examples

showClass("PredictionRegion")

Description

This function normalizes the intensities for the 'A' and 'B' alleles for a CNSet object and estimates mixture parameters used for subsequent genotyping. See details for how the normalized intensities are written to file. This step is required for subsequent genotyping and copy number estimation.

Usage

preprocessInf(cnSet, sampleSheet=NULL, arrayNames = NULL, ids = NULL, path = ".", arrayInfoColNames = list(barcode = "SentrixBarcode_A", position = "SentrixPosition_A"), highDensity = TRUE, sep = ",", fileExt = list(green = "Grn.idat", red = "Red.idat"), XY, anno, saveDate = TRUE, stripNorm = TRUE, useTarget = TRUE, mixtureSampleSize = 10^5, fitMixture = TRUE, quantile.method="between", eps = 0.1, verbose = TRUE, seed = 1, cdfName)

Arguments

cnSet object of class CNSet
sampleSheet data.frame containing Illumina sample sheet information (for required columns, refer to BeadStudio Genotyping guide - Appendix A).
arrayNames character vector containing names of arrays to be read in. If NULL, all arrays that can be found in the specified working directory will be read in.
ids vector containing ids of probes to be read in. If NULL all probes found on the first array are read in.
path character string specifying the location of files to be read by the function
arrayInfoColNames (used when sampleSheet is specified) list containing elements 'barcode' which indicates column names in the sampleSheet which contains the arrayNumber/barcode number and 'position' which indicates the strip number. In older style sample sheets, this information is combined (usually in a column named 'SentrixPosition') and this should be specified as list(barcode=NULL, position="SentrixPosition")
highDensity logical (used when sampleSheet is specified). If TRUE, array extensions '_A', '_B' in sampleSheet are replaced with 'R01C01', 'R01C02' etc.
sep character string specifying separator used in .idat file names.
fileExt list containing elements 'Green' and 'Red' which specify the .idat file extension for the Cy3 and Cy5 channels.
XY an NChannelSet object containing X and Y intensities.
anno data.frame containing SNP annotation information from manifest and additional columns 'isSnp', 'position', 'chromosome' and 'featureNames'. For use when cdfName='nopackage'
saveDate 'logical'. Should the dates from each .idat be saved with sample information?
stripNorm 'logical'. Should the data be strip-level normalized?
useTarget 'logical' (only used when stripNorm=TRUE). Should the reference HapMap intensities be used in strip-level normalization?
mixtureSampleSize Sample size to be use when fitting the mixture model.
fitMixture 'logical.' Whether to fit per-array mixture model.
quantile.method character string specifying the quantile normalization method to use ('within' or 'between' channels).
eps Stop criteria.
verbose 'logical.' Whether to print descriptive messages during processing.
seed Seed to be used when sampling. Useful for reproducibility
cdfName character string indicating which annotation package to load.

Details
The normalized intensities are written to disk using package ff protocols for writing/reading to disk. Note that the object CNSet containing the ff objects in the assayData slot will be updated after applying this function.

Value
A ff_matrix object containing parameters for fitting the mixture model. Note that while the CNSet object is not returned by this function, the object will be updated as the normalized intensities are written to disk. In particular, after applying this function the normalized intensities in the alleleA and alleleB elements of assayData are now available.
readGenCallOutput

Author(s)
R. Scharpf

See Also
CNSet-class, A, B, constructInf, genotypeInf, annotationPackages

Examples
## See the 'illumina_copynumber' vignette in inst/scripts of
## the source package

readGenCallOutput  Read X and Y intensities from GenCall output

Description
This function reads the raw X and Y intensities output by GenomeStudio’s GenCall genotyping
module in preparation for genotyping with crlmm.

Usage
readGenCallOutput(filenames, path=".", cdfName, colnames=list("SampleID"="Sample ID", "SNPID"="SNP Name", "XRaw"="X Raw", "YRaw"="Y Raw"),
type=list("SampleID"="character", "SNPID"="character", "XRaw"="integer", "YRaw"="integer"), verbose=FALSE)

Arguments
filenames  'character' string, or a vector of character string specifying the name of the
            file(s) to read in
path  'character' string specifying the location of file to be read by the function
cdfName  'character' defining the chip annotation (manifest) to use ('human370v1c', human550v3b', 'human650v3a', 'human1mv1c', 'human370quadv3c', 'human610quadv1b',
            'human660quadv1a', 'human1mduov3b', 'humanomni1quadv1b', 'humanomniexpress12v1b', 'humanomniexpress12v2p1h')
colnames  list containing elements 'SampleID', 'SNPID', 'XRaw' and 'YRaw', which
            specify the column names from in 'file' that pertain to these variables. The
            default should suffice in most situations.
type  list containing data types for the columns to be read in. The default should be
            fine in most situations.
verbose  'logical'. Should processing information be displayed as data is read in?

Details
This function returns an NChannelSet containing raw intensity data (X and Y) from GenCall final
report file. It assumes the GenCall output is formatted to have samples listed one below the other,
and that the columns 'X Raw' and 'Y Raw' are available in the file. The function crlmmilluminia()
can be run on the output of the readGenCallOutput function.
**Value**

NChannelSet containing X and Y bead intensities.

**Author(s)**

Cynthia Liu, Matt Ritchie, Zhiyin Dai

**References**


**Examples**

```r
#XY = readGenCallOutput(file="Hap650Yv3_Final_Report.txt", cdfName="human650v3a")
#crlmmOut = crlmmIllumina(XY=XY)
```

---

**readIdatFiles**  
*Reads Idat Files from Infinium II Illumina BeadChips*

**Description**

Reads intensity information for each bead type from .idat files of Infinium II genotyping BeadChips

**Usage**

```r
readIdatFiles(sampleSheet=NULL, arrayNames=NULL, ids=NULL, path='',
arrayInfoColNames=list(barcode="SentrixBarcode_A",
position="SentrixPosition_A"),
highDensity=FALSE, sep="_",
fileExt=list(green="Grn.idat", red="Red.idat"),
saveDate=FALSE, verbose=FALSE)
```

**Arguments**

- `sampleSheet`  
  *data.frame* containing Illumina sample sheet information (for required columns, refer to BeadStudio Genotyping guide - Appendix A).

- `arrayNames`  
  character vector containing names of arrays to be read in. If NULL, all arrays that can be found in the specified working directory will be read in.

- `ids`  
  vector containing ids of probes to be read in. If NULL all probes found on the first array are read in.

- `path`  
  character string specifying the location of files to be read by the function

- `arrayInfoColNames`  
  (used when `sampleSheet` is specified) list containing elements ‘barcode’ which indicates column names in the `sampleSheet` which contains the arrayNumber/barcode number and ‘position’ which indicates the strip number. In older style sample sheets, this information is combined (usually in a column named ‘SentrixPosition’) and this should be specified as `list(barcode=NULL, position="SentrixPosition")`
SNPRMA will preprocess SNP chips. The preprocessing consists of quantile normalization to a known target distribution and summarization to the SNP-Allele level.

References
Usage

\[
\text{snprma}(\text{filenames, mixtureSampleSize = } 10^5, \text{ fitMixture = FALSE, eps = } 0.1, \text{ verbose = TRUE, seed = 1, cdfName, sns})
\]
\[
\text{snprma2}(\text{filenames, mixtureSampleSize = } 10^5, \text{ fitMixture = FALSE, eps = } 0.1, \text{ verbose = TRUE, seed = 1, cdfName, sns})
\]

Arguments

- **filenames**: 'character' vector with file names.
- **mixtureSampleSize**: Sample size to be use when fitting the mixture model.
- **fitMixture**: 'logical'. Fit the mixture model?
- **eps**: Stop criteria.
- **verbose**: 'logical'.
- **seed**: Seed to be used when sampling.
- **cdfName**: cdfName: 'GenomeWideSnp\_5', 'GenomeWideSnp\_6'
- **sns**: Sample names.

Details

'snprma2' allows one to genotype very large datasets (via ff package) and also permits the use of clusters or multiple cores (via snow package) to speed up preprocessing.

Value

- **A**: Summarized intensities for Allele A
- **B**: Summarized intensities for Allele B
- **sns**: Sample names
- **gns**: SNP names
- **SNR**: Signal-to-noise ratio
- **SKW**: Skewness
- **mixtureParams**: Parameters from mixture model
- **cdfName**: Name of the CDF

Examples

```r
if (require(genomewidesnp6Crlmm) & require(hapmapsnp6) & require(oligoClasses)){
  path <- system.file("celFiles", package="hapmapsnp6")
  ## the filenames with full path...
  ## very useful when genotyping samples not in the working directory
  cels <- list.celfiles(path, full.names=TRUE)
  snprmaOutput <- snprma(cels)
  snprmaOutput[["A"]][1:10,]
  snprmaOutput[["B"]][1:10,]
}
```

## Not run:
## HPC Example
library(ff)
library(snow)
library(crlmm)
## genotype 50K SNPs at a time
ocProbesets(50000)
## setup cluster - 8 cores on the machine
setCluster(8, "SOCK")

path <- system.file("celFiles", package="hapmapsnp6")
cels <- list.celfiles(path, full.names=TRUE)
snprmaOutput <- snprma2(cels)

## End(Not run)

---

**snprmaAffy**

Quantile normalize intensities for SNPs

### Description

Quantile normalize intensities for SNPs to a HapMap target reference distribution

### Usage

```r
snprmaAffy(cnSet, mixtureSampleSize = 10^5, eps = 0.1, seed = 1, verbose = TRUE)
```

### Arguments

- **cnSet**: Object of class CNSet
- **mixtureSampleSize**: Sample size to be use when fitting the mixture model.
- **eps**: Stop criteria.
- **seed**: Seed to be used when sampling.
- **verbose**: Logical.

### Value

Returns nothing. Normalized intensities are written to files.

### Author(s)

R.Scharpf

### See Also

*snprma*
validCdfNames

**Description**

Supported annotation packages for crlmm genotyping

**Usage**

validCdfNames()

**Details**

List of available annotation packages

**Value**

character vector

**Author(s)**

R.Scharpf

**Examples**

validCdfNames()

validCEL

**Description**

Reads cel files and return an error if a file is not read

**Usage**

validCEL(celfiles)
celDates(celfiles)

**Arguments**

celfiles vector of cel file names to read

**Value**

Returns a message that cel files were successfully read, or an error if there were problems reading the cel files.
Author(s)
R. Scharpf

See Also
read.celfile.header, POSIXt, read.celfile

Examples

library(oligoClasses)
if(require(hapmapsnp6)){
  path <- system.file("celFiles", package="hapmapsnp6")
  cels <- list.celfiles(path, full.names=TRUE)
  validCEL(cels)
  celDates(cels)
}

xyplot(x, data, ...)

Description
Plot prediction regions for integer copy number and normalized intensities.

Usage
xyplot(x, data, ...)

Arguments

x A formula.
data A CNSet object.
... Additional arguments passed to xyplot function in lattice.

Value
A trellis object.

Author(s)
R. Scharpf

See Also
xyplot, ABpanel
Examples

library(oligoClasses)
data(cnSetExample2)
table(batch(cnSetExample2))
sample.index <- which(batch(cnSetExample2) == “CUPID”)
## A single SNP
pr <- predictionRegion(cnSetExample2[1:4, sample.index], copyNumber=0:4)
gt <- calls(cnSetExample2[1:4, sample.index])
lim <- c(6,13)
xyplot(B~A|snpid, data=cnSetExample2[1:4, sample.index],
       predictRegion=pr,
       panel=ABpanel,
       pch=21,
       fill=c(“red”, “blue”, “green3”)[gt],
       xlim=lim, ylim=lim)

## multiple SNPs, prediction regions for 3 batches
## Not run:
tab <- table(batch(cnSetExample2))
bns <- names(tab)[tab > 50]
sample.index <- which(batch(cnSetExample2)
pr <- predictionRegion(cnSetExample2[7:10, sample.index], copyNumber=0:4)
gt <- as.integer(calls(cnSetExample2[7:10, sample.index]))
xyplot(B~A|snpid, data=cnSetExample2[7:10, sample.index],
       predictRegion=pr,
       panel=ABpanel,
       pch=21,
       fill=c(“red”, “blue”, “green3”)[gt],
       xlim=lim, ylim=lim)

## nonpolymorphic markers
data(cnSetExample2)
tab <- table(batch(cnSetExample2))
bns <- names(tab)[tab > 50]
sample.index <- which(batch(cnSetExample2)
np.index <- which(!isSnp(cnSetExample2))[1:10]
taus <- tau2(cnSetExample)[np.index, , , ]
pr <- predictionRegion(cnSetExample2[np.index, sample.index],
       copyNumber=0:4)
pp <- posteriorProbability(cnSetExample2[np.index, sample.index],
       predictRegion=pr,
       copyNumber=0:4)

## End(Not run)
# Index

* **IO**
  - `readGenCallOutput`, 35
  - `readIdatFiles`, 36
  - `validCEL`, 40
* **aplot**
  - `ABpanel`, 3
* **array**
  - `posteriorProbability`, 30
* **classes**
  - `PredictionRegion-class`, 32
* **classif**
  - `crlmm`, 13
  - `genotype`, 18
  - `genotype.Illumina`, 21
  - `genotypeAffy`, 25
  - `genotypeInf`, 26
  - `snprma`, 37
* **datasets**
  - `cnSetExample`, 9
* **distribution**
  - `predictionRegion`, 31
* **dplot**
  - `xyplot`, 41
* **hplot**
  - `plotSNPs`, 29
  - `xyplot`, 41
* **list**
  - `calculateRBaf`, 6
  - `predictionRegion`, 31
* **manip**
  - `AssayData-methods`, 4
  - `batchStatisticAccessors`, 5
  - `constructAffyCNSet`, 9
  - `constructInf`, 10
  - `copynumberAccessors`, 12
  - `crlmmCopynumber`, 16
  - `genotypes`, 27
  - `ListClassConstructors`, 28
  - `preprocessInf`, 33
  - `snprma`, 37
  - `validCdfNames`, 40
* **methods**
  - `calculateRBaf`, 6
  - `CNSet-methods`, 8
* **package**
  - `crlmm-package`, 3
* **robust**
  - `cnrmaAffy`, 7
  - `snprmaAffy`, 39
  - `[,PredictionRegion,ANY,ANY,ANY-method (PredictionRegion-class), 32

A, 35
  - `ABpanel`, 3, 41
  - `annotationPackages`, 27, 35
  - `AssayData`, 33
  - `AssayData-methods`, 4
B, 35
  - `BafLrrSetList (ListClassConstructors)`, 28
  - `BafLrrSetList,CNSet-method (CNSet-methods)`, 8
  - `batch`, 20
  - `batchStatisticAccessors`, 5
  - `batchStatistics`, 6
  - `BeadStudioSetList`, 28
CA, 8
  - `CA (copynumberAccessors)`, 12
  - `CA,CNSet-method (CNSet-methods)`, 8
  - `calculateRBaf`, 6
  - `calculateRBaf,CNSet-method (calculateRBaf)`, 6
  - `calls`, 25
CB, 8
  - `CB (copynumberAccessors)`, 12
  - `CB,CNSet-method (CNSet-methods)`, 8
  - `celDates (validCEL)`, 40
Index

Affy, 7
CNRSet-methods, 8
cnSetExample, 9
cnSetExample2 (cnSetExample), 9
cnSetExample2 (cnSetExample), 9
coerce, CNSet, oligoSnspSet-method (CNSet-methods), 8
corrs, 25
constructAffyCNSet, 9
constructInf, 10, 35
copynumberAccessors, 12
corr, 5
corr (batchStatisticAccessors), 5
corr, AssayData-method (AssayData-methods), 4
corr, CNSet-method (CNSet-methods), 8
crlmm, 13, 20, 25, 27
crlmm-package, 3
crlmm2 (crlmm), 13
crlmmCopynumber, 13, 16, 20
crlmmCopynumber2 (crlmmCopynumber), 16
crlmmCopynumberLD (crlmmCopynumber), 16
crlmmIllumina (genotype.Illumina), 21
genotype, 18
genotype.Illumina, 21, 30
genotypeInf (genotype), 18
genotypeAffy, 25
genotypeInf, 11, 26, 35
genotypeLD (genotype), 18
ngenotypes, 27, 31, 32
i2p, 15
ldOpt, 20, 23
ldPath, 11
lines, CNSet-method (CNSet-methods), 8
list, 33
list_or_ffdf, 33
ListClassConstructors, 28
lpolygon, 4
mads, 5
mads (batchStatisticAccessors), 5
mads, AssayData-method (AssayData-methods), 4
mads, CNSet-method (CNSet-methods), 8
medians, 5
medians (batchStatisticAccessors), 5
medians, AssayData-method (AssayData-methods), 4
medians, CNSet-method (CNSet-methods), 8
Ns, 5
Ns (batchStatisticAccessors), 5
Ns, AssayData-method (AssayData-methods), 4
Ns, CNSet-method (CNSet-methods), 8
nuA (copynumberAccessors), 12
nuA, CNSet-method (CNSet-methods), 8
nuB (copynumberAccessors), 12
nuB, CNSet-method (CNSet-methods), 8
ocProbesets, 11
ocSamples, 11, 20, 23
OligoSetList (ListClassConstructors), 28
OligoSetList, CNSet-method (CNSet-methods), 8
panel.xplot, 4
phiA (copynumberAccessors), 12
phiA, CNSet-method (CNSet-methods), 8
phiB (copynumberAccessors), 12
phiB, CNSet-method (CNSet-methods), 8
plotSamples (plotSNPs), 29
plotSNPs, 29
POSIXt, 41
posteriorProbability, 30, 32
posteriorProbability, CNSet-method (posteriorProbability), 30
predictionRegion, 31, 31, 33
predictionRegion, CNSet, integer-method (predictionRegion), 31
PredictionRegion-class, 32
preprocessInf, 11, 33
rawCopynumber, 8
rawCopynumber (copynumberAccessors), 12
rawCopynumber, CNSet-method (CNSet-methods), 8
read.celfile, 41
read.celfile.header, 41
readGenCallOutput, 35
readIdatFiles, 36
readIdatFiles2 (readIdatFiles), 36
snpCall, 15, 27
snpCallProbability, 15, 27
snprma, 20, 37, 39
snprma2 (snprma), 37
INDEX

snprmaAffy, 8, 39

tau2, 5
tau2(batchStatisticAccessors), 5
tau2, AssayData-method
  (AssayData-methods), 4
tau2, CNSet-method (CNSet-methods), 8
totalCopynumber, 8
totalCopynumber (copynumberAccessors), 12
totalCopynumber, CNSet-method
  (CNSet-methods), 8

validCdfNames, 40
validCEL, 40
vector, 33

xyplot, 4, 41, 41
xyplot, formula, CNSet-method (xyplot), 41