Package 'clasp'

December 25, 2014

Title Cell Line Authentication by SNP Profiling

Version 0.1

Description CLASP uses SNP array data to determine whether a
cell line matches its expected genetic background and predicts
whether it is contaminated and/or aneuploid. If you use this
software in a publication, please cite: Didion JP et al, 'SNP
array profiling of mouse cell lines identifies their strains
of origin and reveals cross-contamination and widespread
aneuploidy,' BMC Genomics 2014, 15:847, doi:10.1186/1471-2164-15-847.
A vignette showing how to use CLASP to reproduce the analyses in
this paper is available in the CLASP GitHub repository: https://github.com/jdidion/clasp.

Depends R (>= 3.1.1)

License file LICENSE

URL https://github.com/jdidion/clasp

LazyData true

Suggests knitr, rfigshare

VignetteBuilder knitr

Imports RSQLite, tools, bitops, limma, genoCN, reshape2, RColorBrewer,Rmpfr

Author John Didion [aut, cre]

Maintainer John Didion <john.didion@nih.gov>

R topics documented:

add.samples 2
compute.dilution.series 3
compute.metrics 4
compute.metrics.Illumina 4
connect.database 5
copy.number.genoCN 5
copy.number.genoCN.plot 6
develop.assay 7
estimate.contamination 8

estimate.copy.number 9
execute.assay 10
get.all.arrays 10
get.all.genomes 11
get.all.marker.sets 11
get.all.samples 12
get.array.id 12
get.assay 13
get.genome.id 13
get.genotypes 14
get.marker.set 14
get.markers 15
get.platform.data 15
get.samples 16
import.array 16
import.genome 17
import.genotypes 17
import.marker.set 18
import.sample.set 19
import.samples 19
init.database 20
insert.assay 21
intersect.marker.sets 21
load.objects 22
metrics.plot 22
print.summary.CLASPAssay 23
rgb.alpha 24
summary.CLASPAssay 24

Index **25**

`add.samples` *Import sample data and add the samples to an existing sample set.*

Description

Import sample data and add the samples to an existing sample set.

Usage

```
add.samples(sample.set.id, samples, array.id, db, pedigree = NULL)
```

Arguments

`sample.set.id`	The sample set to add to.
`samples`	a data.frame as specified in `import.samples`.
`array.id`	ID of the array.
`db`	The database connection.
`pedigree`	Pedigree information as specified in `import.samples`.

Value

a list with two elements: 1. ids: a vector of database IDs, one for each sample. 2. ped: the pedigree matrix with sample names replaced by ids (or NULL if no pedigree was specified).

`compute.dilution.series`
Create a DilutionSeries object from a set of samples in the database.

Description

A DilutionSeries is used to predict sample contamination. To create a dilution series, make in vitro mixtures of two geneticially divergent samples in a series of ratios (the more the better). Then genotype these samples and import them into the database.

Usage

```
compute.dilution.series(marker.set.id, sample.set.id, ratios, params, db,
  platform = "Illumina")
```

Arguments

`marker.set.id`	ID of marker set for which to compute dilution series metrics.
`sample.set.id`	ID of sample set containing dilution series samples.
`ratios`	A data frame with three columns (see note).
`params`	Reference parameters.
`db`	Database connection
`platform`	array platform.

Details

For MUGA and MegaMUGA arrays, you can use the pre-computed DilutionSeries `default.dilution.series`.

Ratios table columns: 1. sample.id 2. control The proportion of the mixture containing the control sample 3. contaminant The proportion of the mixture containing the contaminant sample control.frac and contam.frac must be numeric values summing to 1.0 for each row.

Value

a DilutionSeries object.

`compute.metrics` *Compute copy-number metrics.*

Description

Compute copy-number metrics (BAF and LRR) for cell line samples based on intensity values.

Usage

```
compute.metrics(marker.set.id, params, db, cell.lines = NULL,
  platform = "Illumina", normalize = TRUE)
```

Arguments

`marker.set.id`	ID of marker set containing markers that match params
`params`	reference parameters (see note)
`db`	Database connection.
`cell.lines`	Sample set ID for cell line samples, or NULL to select all cell lines.
`platform`	Platform name.
`normalize`	whether to normalize the intensity values.

Details

Computing LRR requires a large number of reference samples. We provide reference parameters for the MUGA and MegaMUGA array. A future version of the software will provide tools to derive new reference parameters. In the meantime, please contact the author for help adapting new arrays to work with this software.

Value

list of metrics (one per sample)

`compute.metrics.Illumina` *Compute metrics using Illumina array data.*

Description

Compute metrics using Illumina array data.

Usage

```
compute.metrics.Illumina(data, params, normalize = TRUE)
```

Arguments

`data`	list of matricies of platform-specific data (one per variable)
`params`	reference parameters
`normalize`	whether to normalize intensity data

Value

list of metrics (baf and lrr), one per sample.

`connect.database` *Open the SQLite database specified by* `dbfile`.

Description

Open the SQLite database specified by `dbfile`.

Usage

```
connect.database(dbfile)
```

Arguments

`dbfile`	the database file

Value

a datbase connection

`copy.number.genoCN` *Copy number analysis using the genoCN package. genoCN was designed for copy number analysis of human tumor genotypes. Therefore, it assumes that the default state is copy number 2 and all three genotypes possible (AA/AB/BB). This is state 1 in both the genoCNV and genoCNA HMM. For inbred strains, the default state should actually be one of homozygosity (state 2).*

Description

Short segments can be avoided by setting seg.nSNP to a larger number (default=3).

Usage

```
copy.number.genoCN(samples, metrics, pfb, ...)
```

Arguments

`samples`	data.frame of sample info. At a minimum, name and type are required.
`metrics`	list of data.frames (one per sample), each with copy number metrics (baf and lrr) for each marker.
`pfb`	data.frame with data from PennCNV PFB file (see notes).
`...`	additional arguments passed to .run.genoCNA.

Details

The PFB format originated with PennCNV and is also used in genoCNA. A file is derived for each array. We provide PFBs for MUGA and MegaMUGA. A future version of the software will include a function to derive your own PFB, but for the time being please contact the author for help adpating this software to a new array. A PFB has three columns: Chr, Position, PFB, with rownames being marker IDs.

Value

A list of genoCNAResult objects,

See Also

`estimate.copy.number`

`copy.number.genoCN.plot`

Plot the BAF, LRR and copy number states of a sample.

Description

Plot the BAF, LRR and copy number states of a sample.

Usage

```
copy.number.genoCN.plot(sample.id, marker.set.id, metrics, result, norm.params,
  db, cn.range = NULL, ...)
```

Arguments

`sample.id`	ID of sample to plot
`marker.set.id`	ID of marker set
`metrics`	copy number metrics
`result`	from calling copy.number.genoCN
`norm.params`	normalization parameters
`db`	database connection
`cn.range`	the range of copy numbers to display
`...`	additional arguments to pass to `plot.metrics`.

`develop.assay` *Develop a new assay.*

Description

Develop a new assay by selecting the most informative subset of `markers` for the specified samples.

Usage

```
develop.assay(name, marker.set, db, sample.set = NULL, insert = TRUE,
  description = NULL, ob.name = NULL, ob.description = NULL,
  exclude.cell.lines = TRUE, min.call.rate = 0.8, min.maf = 0,
  max.hwe.pvalue = NULL, consistency.check = TRUE, max.ld.r2 = NULL,
  max.ld.pvalue = NULL, keep.informative.linked.markers = TRUE,
  max.sdp.diff = 0.05, diag.coeffs = list(N = 1, AA = 1, BB = 1, AB = 1),
  reduce = FALSE, min.diff = NULL, min.markers = NULL,
  diag.threshold.iters = seq(0.25, 0.7, 0.05))
```

Arguments

`name`	Assay name
`marker.set`	A vector of marker set IDs (see details) or a marker data.frame.
`db`	The database connection.
`sample.set`	A vector of sample set IDs, a sample data.frame or NULL if all samples should be used.
`insert`	Whether or not to insert the new assay into the database.
`description`	If insert=TRUE, the text description of the new marker set.
`ob.name`	If insert=TRUE, the name of the new outbred-specific marker set.
`ob.description`	If insert=TRUE, the text description of the new outbred-specific marker set.
`exclude.cell.lines`	If TRUE, cell line samples are not used to develop the assay.
`min.call.rate`	Minimum fraction of samples that have to have a non-missing (i.e. non-zero) genotype (inclusive).
`min.maf`	Minimum fraction of samples with non-missing genotype information that must have a minor allele. The value is not inclusive, so a value of 0 means the MAF must be > 0, whereas a value of NULL means the check is not performed.
`max.hwe.pvalue`	The maximum p-value (inclusive) for a test of deviation from the expected Hardy-Weinberg allele frequency.
`consistency.check`	Whether a consistency check should be performed for replicates.
`max.ld.r2`	Maximum `R^2` value (inclusive) for pairwise linkage between consecutive markers.
`max.ld.pvalue`	Maximum p-value (inclusive) for a test of pairwise linkage between consecutive markers.
`keep.informative.linked.markers`	If a linkage test is performed, this parameter specifies that markers that exceed the `R^2` threshold should not be filtered if their SDPs differ (see Details).

`max.sdp.diff`	Maximum fraction by which two SDPs can differ and still be considered equal for the purposes of `keep.informative.linked.markers`.
`diag.coeffs`	genotype-speific coefficients for computing diagnostic values.
`reduce`	Logical indicating whether the maximal marker set should be pruned using the the following parameters.
`min.diff`	When reduce=TRUE, the minimum allowable distance between any two markers.
`min.markers`	When reduce=TRUE, the minimum number of markers in the final marker set. The actual size of the final marker set may be smaller than this if the other filtering parameters remove too many markers.
`diag.threshold.iters`	A vector of thresholds to iterate through when pruining markers.

Details

An assay may be developed from a single marker set or from multiple intersecting marker sets. In the latter case (i.e. if `markers` is a vector of two or more IDs), the intersection of all the marker sets will be found using `intersect.marker.sets`.

For the purposes of the consistency check and removing replicate samples, two samples are considered replicates if they meet any of the following criteria: 1. Identical names 2. Identical non-NULL cell line names AND identical sources 3. Null cell line names AND identical non-NULL backgrounds

An SDP (sample distribution pattern) is the partitioning of a set of samples based on genotypes for a single marker. It can be thought of as the phylogenetic tree with N braches, where N is the number of possible genotypes. Consecutive markers with different SDPs are considered informative even if their degree of linkage exceeds the specified `R^2` threshold.

It is expected that all samples in the database have passed QC checks; this function does not perform any sample-level QC.

Value

A list with the following elements: set.id: The ID of the new marker set created with the markers for this assay, or NULL if insert=FALSE. markers: Full information for the set of markers included in the assay. samples: Full information for the set of all samples considered for this assay. sample.ixs: Indices of samples that were actually used to develop the assay (i.e. after removing replicates). replicates: A list of replicate groups. geno: MxN matrix of genotypes, where M is the assay markers and N is the assay samples. diagnostic.values: MxN matrix of the diagnostic value for each SNP and for each sample. distmat: NxN matrix of the pairwise number of unequal markers.

`estimate.contamination` *Estimate contamination using platform-specific (i.e. intensity) information.*

Description

Estimate contamination using platform-specific (i.e. intensity) information.

Usage

```
estimate.contamination(cl.metrics, dilution.series, baf.thresholds = c(0.02,
  0.46))
```

Arguments

`cl.metrics`	Metrics for cell lines.
`dilution.series`	A DilutionSeries, or a path to an RData file containing a pre-computed DilutionSeries.
`baf.thresholds`	vector of two BAF thresholds (homozygous and heterozygous). Estimating BAF thresholds requires a large number of heterozygous control samples. For our paper, we used the F1 samples in the database to compute the mean mirrored BAF for homozygous and heterozygous calls and rounded up and down, respectively, to two decimal places. A future version of this software will provide a more robust method of determining the thresholds, as well as tools for generating reference parameters for new arrays. Until then, please contact the author for help adapting new arrays to work with this software.

Value

Numeric vector with names as sample IDs and values as predictions as the fraction of contaminant cells.

`estimate.copy.number` *Estimate copy number for a set of cell lines.*

Description

Estimate copy number for a set of cell lines.

Usage

```
estimate.copy.number(samples, metrics, cn.args, method = "genoCN")
```

Arguments

`samples`	data.frame of sample info. At a minimum, name and type are required.
`metrics`	Metrics for the cell line samples.
`cn.args`	List of rguments for the copy number estimation function.
`method`	Copy number estimation method.

Value

list of copy number predictions. Each list item is a list with three elements: 1. mean.cn Vector of numeric values, the weighted mean copy number for each chromosome. 2. summary Table with one or more entries for each chromosome. Each entry gives the fraction of the chromosome that has a given copy number. 3. details Original results from the specific copy number prediction method that was used.

`execute.assay` *Execute an assay on a set of cell lines.*

Description

Execute an assay on a set of cell lines.

Usage

```
execute.assay(assay, db, error.rate, cell.lines = NULL, impute.bgs = T,
  results.per.cl = 10, secondary.threshold = 0.96)
```

Arguments

`assay`	CLASPAssay object
`db`	database connection
`error.rate`	The error rate to use for PIA calculations.
`cell.lines`	sample set IDs or vector of sample IDs of cell lines to test. If null, all cell lines in the database will be tested.
`impute.bgs`	boolean or vector of background (e.g. inbred strain) names. If TRUE, synthetic F1s will be imputed for pairwise combinations of all backgrounds with their impute field set to TRUE. Only samples with impute column set to TRUE are used. If there is more than one sample of a given background, one is selected randomly.
`results.per.cl`	Number of matches to report for each cell line.
`secondary.threshold`	match score below which a secondary background is searched.

Value

The return value is a list with the following elements: 1. samples: Data frame of cell line sample data 2. pairwise.matrix: Pairwise similarity matrix between all pairs of cell lines 3. table: a summary data frame of the results with the following columns: id: Cell line sample ID primary.match: Name of best matching sample primary.match.score: Alignment score of primary match secondary.match: Name of best secondary match (if primary.match.score < secondary.threshold) secondary.match.score: Alignment score of secondary match pia: Probabilty of Incorrect Assignment 4. details: a list the same length as the number of cell lines with the following elements: primary.matches secondary.matches

`get.all.arrays` *Get information for all genotyping arrays in the database.*

Description

Get information for all genotyping arrays in the database.

Usage

```
get.all.arrays(db)
```

Arguments

`db`	the database connection.

Value

table of array information.

`get.all.genomes` *Get information for all genome builds in the database.*

Description

Get information for all genome builds in the database.

Usage

```
get.all.genomes(db)
```

Arguments

`db`	The database connection.

Value

table of genome information.

`get.all.marker.sets` *Get a list of all marker sets.*

Description

Get a list of all marker sets.

Usage

```
get.all.marker.sets(db)
```

Arguments

`db`	the datbase connection.

Value

a table of marker sets.

get.all.samples *Get a list of all samples.*

Description

Get a list of all samples.

Usage

```
get.all.samples(db, where = NULL, include.pedigree = TRUE)
```

Arguments

`db`	database connection
`where`	list of key/value pairs to specify conditions for sample table columns.
`include.pedigree`	whether to return pedigree information in the sample table.

Value

table of sample information

get.array.id *Get array ID by name.*

Description

Get array ID by name.

Usage

```
get.array.id(name, db)
```

Arguments

`name`	the array name.
`db`	database connection

Value

the ID.

`get.assay` *Fetch assay from the database.*

Description

Fetch assay from the database.

Usage

```
get.assay(id, db, load = TRUE)
```

Arguments

`id`	assay ID.
`db`	database connection.
`load`	logical; if true, sample, marker and genotype data are loaded from the database.

Value

a list containing fields from the Assay table. If load=TRUE, the list also contains loaded sample, marker and genotype data AND the object is designated as a CLASPAssay object.

`get.genome.id` *Get genome ID by name.*

Description

Get genome ID by name.

Usage

```
get.genome.id(name, db)
```

Arguments

`name`	genome name
`db`	database connection

Value

the ID

get.genotypes — *Fetch genotype data for arbitrary sets of markers and samples.*

Description

Fetch genotype data for arbitrary sets of markers and samples.

Usage

```
get.genotypes(markers, db, samples = NULL, as.matrix = TRUE)
```

Arguments

`markers`	A marker set ID or a vector of marker IDs.
`db`	The database connection.
`samples`	A sample set ID, a list of sample IDs, or NULL for all samples.
`as.matrix`	Whether the result should be converted from a three-column data frame (marker ID, sample ID, call) to a MxN matrix.

Value

A MxN data frame of (recoded) genotypes, where M = markers and N = samples. Row names are marker IDs and colnames are sample IDs.

get.marker.set — *Get a specific marker set.*

Description

Get a specific marker set.

Usage

```
get.marker.set(set.id, db)
```

Arguments

`set.id`	The marker set ID.
`db`	the datbase connection.

Value

table with marker information.

`get.markers` *Load the markers from a marker set or list of marker IDs.*

Description

Load the markers from a marker set or list of marker IDs.

Usage

```
get.markers(ids, db)
```

Arguments

`ids`	The marker set ID, or a list of marker IDs.
`db`	the datbase connection. name (the marker set-specific name for the marker, if any).

`get.platform.data` *Fetch platform-specific data.*

Description

Fetch platform-specific data.

Usage

```
get.platform.data(marker.set.id, platform, db, samples = NULL,
  as.matrix = TRUE, format = 1)
```

Arguments

`marker.set.id`	The assay ID.
`platform`	The name of the platform-specific data table.
`db`	The database connection.
`samples`	The set of samples, or all samples if NULL.
`as.matrix`	Whether the result should be converted from a four column data frame to a list of two MxN matricies.
`format`	If as.matrix=TRUE, controls whether rows are samples and columns are markers (format=1) or vice-versa (format=2).

Value

A list of data frames. Each is a MxN data frame of a platform-specific variable, where M = markers and N = samples.

`get.samples` *Get a list of the samples in the set with the specified ID, or by a vector of sample IDs.*

Description

Get a list of the samples in the set with the specified ID, or by a vector of sample IDs.

Usage

```
get.samples(ids, db, where = NULL, include.pedigree = TRUE)
```

Arguments

`ids`	The sample set ID, or a list of sample IDs.
`db`	the datbase connection.
`where`	list of key/value pairs to specify conditions for sample table columns.
`include.pedigree`	whether to return pedigree information in the sample table.

Value

data frame with sample information.

`import.array` *Define a genotyping array in the database (if it doesn't already exist) and import the default marker set for that array for a given genome.*

Description

Define a genotyping array in the database (if it doesn't already exist) and import the default marker set for that array for a given genome.

Usage

```
import.array(name, platform, genome.id, markers, db, ...)
```

Arguments

`name`	array name
`platform`	array platform (e.g. Illumina)
`genome.id`	ID of the genome on which the specified marker annotations are based.
`markers`	marker data, as described in import.marker.set
`db`	the database
`...`	additional parameters to import.marker.set

Value

a vector of two IDs: (array.id, marker.set.id)

`import.genome` *Define a genome build in the database.*

Description

For organisms that have multiple maintainers of the same build, the most official name should be used. E.g. for mouse, 'GRCm38' should be used rather than 'mm10'.

Usage

```
import.genome(taxon, version, db)
```

Arguments

`taxon`	Taxon of the genome (i.e. Mus musculus)
`version`	Genome build version.
`db`	database connection

Value

the ID of the genome.

`import.genotypes` *Import genotype data.*

Description

`genotypes` must be a data.frame with at least three columns. The first two columns specify the sample and marker. Each may be either numeric, in which case it is interpreted as an ID, or character, in which case it is interpreted as a name. The third column is the genotype call. Subsequent columns are interpreted as platform-specific data.

Usage

```
import.genotypes(genotypes, db, N.call = "N", platform = NULL,
  marker.set.id = NULL, sample.set.id = NULL)
```

Arguments

`genotypes`	Genotype data. Either a path to a tab-delimited file (supports gzipped files) or a data.frame. See Details.
`db`	The database connection.
`N.call`	The genotype used for missing information (no-call). Any other genotype that is not the two possible alleles or the N.call value is considered a heterozygous call.
`platform`	The name of the table in which to store platform-specific data, if applicable.
`marker.set.id`	ID of a marker set. If NULL, marker IDs will be selected by name.
`sample.set.id`	ID of a sample set. If NULL, sample IDs will be selected by name.

Details

Genotypes are .recoded into a bit vector (stored as an integer), where each bit represents one of the possible alleles. Missing information is coded as 0.

`import.marker.set` *Import marker data.*

Description

Currently, CLASP only supports bi-allelic markers and does not support sex chromosome markers. The later is because cell lines are often of indeterminate sex or exhibit sex chromosome mosaicism.

Usage

```
import.marker.set(name, markers, db, description = NA, genome.id = NULL,
  start.transaction = TRUE)
```

Arguments

`name`	Name for the marker set.
`markers`	vector of marker IDs or path to marker data file or data.frame containing marker information. See Details.
`db`	the datbase connection.
`description`	Text description of the marker set.
`genome.id`	ID of the genome build from which the markers were sourced.
`start.transaction`	set to false if calling from another method that starts a transaction.

Details

If `markers` is a vector, the elements are marker IDs and the names can optionally be marker set-specific names. If it is a data.frame, it must have five columns: 1. name: Name of the marker. 2. chromosome: Name of the chromosome. Must be an integer. 3. bp: Physical position of the marker. 4. cm: Genetic position of the marker (in centimorgans). 5. alleles: String listing the two possible genotypes for this marker. The alleles should be listed with the reference allele first, followed by the variant allele. If this marker is to be associated with intensity data, the intensity values should be reordered to match the allele order.

Value

If markers is a path, returns a list with two elements, set.id and markers. Otherwise returns the integer ID of the new marker set.

`import.sample.set` *Create a sample set from a list of samples.*

Description

A sample set is created. If samples is a vector of sample IDs, those samples are added to the set. If it is a data frame, `import.samples` is called.

Usage

```
import.sample.set(name, db, description = NULL, samples = NULL,
  array.id = NULL, pedigree = NULL, fetch = FALSE)
```

Arguments

`name`	A unique name for the sample set.
`db`	the datbase connection.
`description`	A text description of the sample set.
`samples`	a vector of sample IDs, a data.frame as specified in `import.samples`, or a path to a file containing sample data.
`array.id`	array ID to use for import.samples. Ignored unless samples is a data frame or path.
`pedigree`	Pedigree information as specified in `import.samples`.
`fetch`	if `samples` is a vector of sample IDs, whether to fetch the sample information from the database rather than the default, which is to just return the new sample set ID.

Value

sample set ID, or, if samples was a data frame, a list with the following elements: 1. set.id: The ID of the new sample set. 2. ids: A vector of ids for the samples that were imported. 3. ped: Pedigree data.frame, with sample names replaced with IDs.

See Also

`import.samples`

`import.samples` *Import sample data.*

Description

The samples and pedigree parameters can be specified as a single or two separate data.frames or files. If they are specfied as files, than blank fields in logical columns will be given the default value of FALSE.

Usage

```
import.samples(samples, array.id, db, pedigree = NULL, fetch = FALSE,
  start.transaction = TRUE)
```

Arguments

`samples`	path to file, data.frame containing sample information, or vector of sample IDs.
`array.id`	ID of the array.
`db`	the database connection.
`pedigree`	pedigree information for samples with known parentage, or NULL.
`fetch`	if `samples` us a list of IDs, whether to fetch sample information.
`start.transaction`	whether to start a database transaction. Should only be set to FALSE when called from another function that itself starts a transaction.

Details

`samples` must be a data.frame with eight columns: 1. name: Sample name. 2. cell_line: Name of the cell line, or blank if this is a non-cell line sample. 3. background: Genetic background information, e.g. name of mouse inbred strain. 4. type: sample type (inbred, outbred, F1, wild); see note. 5. taxon: The sample taxon, as precise as possible. 6. source: Person/lab/institution that provided the sample. 7. description: Text description of the sample. 8. impute: boolean, whether the sample should be used for imputation of synthetic F1s.

`pedigree` must be a data.frame with three columns: name, mother, father. These must all uniquely refer to a sample within the `samples` data frame.

If `samples` is a path, then pedigree can be specified as an additional three columns.

Samples must be coded as being of one of four types: 1. inbred: fully inbred mouse strain with little to no variablility between the animals of that strain. 2. outbred: outbred stock; individuals share a common background but are randomly mated such that there is a large degree of variability between individuals. 3. F1: a cross between two distinct parental strains/individuals. Pedigree information should be provided if the parental strains are also among the samples. 4. wild: wild-caught individual; generally does not exhibit high identity with any other strain in the database.

Value

a list with two elements: 1. ids: a vector of database IDs, one for each sample. 2. ped: the pedigree matrix with sample names replaced by ids (or NULL if no pedigree was specified).

`init.database` *Initialize a new database with the default schema.*

Description

Using an alternate schema definition will probably break the program unless you make appropriate changes to R the code as well.

Usage

```
init.database(dbfile, schema.file = NULL)
```

Arguments

`dbfile`	the database file to create
`schema.file`	the file from which to load the DDL.

Value

The open database connection.

`insert.assay` *Insert assay into database.*

Description

Insert assay into database.

Usage

```
insert.assay(assay, db)
```

Arguments

`assay`	CLASPAssay object.
`db`	database connection.

Value

the assay updated with an ID column.

`intersect.marker.sets` *Find the intersection of two or more marker sets.*

Description

Find the intersection of two or more marker sets.

Usage

```
intersect.marker.sets(marker.set.ids, db, insert = FALSE, name = NULL,
  description = NULL)
```

Arguments

`marker.set.ids`	A vector of marker.set.ids.
`db`	The database connection.
`insert`	Logical, whether the intersected marker set should be inserted into the database.
`name`	Name for the new marker set.
`description`	Text description of the new marker set.

Value

A data frame with 4+N columns, where N is the number of marker sets. The first four columns are: chromosome, position_bp, position_cm, alleles. The remaining columns are the marker set-specific names for each marker. If insert=TRUE, the result is actually a two-element list: set.id=new marker set id, markers=marker data frame.

`load.objects` *Load an R object file and return a specific object (or a list of all objects if name == NA) without affecting the current environment.*

Description

Load an R object file and return a specific object (or a list of all objects if name == NA) without affecting the current environment.

Usage

```
load.objects(f, name = NA)
```

Arguments

`f`	file name
`name`	object name to retrieve

`metrics.plot` *Plot BAF and LRR values for a single sample.*

Description

`data` must be a data.frame with one row per marker and with at least two columns: chrm and pos (marker chromosome and position). If, `plot.baf=TRUE`, there must be a `baf` column, and if `plot.lrr=TRUE`, there must be a `lrr` column. If there is also a `call` column, points in the BAF plot will be colored according to genotype call.

Usage

```
metrics.plot(name, sample.id, marker.set.id, metrics, norm.params, db,
  chromosomes = 1:19, zoom = NULL, tick.interval = NULL,
  plot.baf = TRUE, plot.lrr = TRUE, lrr.ylim = c(-2, 2),
  plot.trend = TRUE, density.band.window = NULL, density.band.frac = 0.75,
  other.plots = NULL, other.data = NULL, ...)
```

Arguments

`name`	sample name.
`sample.id`	sampleID.
`marker.set.id`	marker set ID.
`metrics`	data.frame with metrics (BAF and LRR).
`norm.params`	normalization params.
`db`	database connection.
`chromosomes`	the chromosomes to plot.
`zoom`	if plotting a single chromosome, the region to zoom in on.
`tick.interval`	distance between x-axis ticks (in Mb). Defaults to a single tick per chromosome.
`plot.baf`	whether to plot B allele frequencies.
`plot.lrr`	whether to plot Log R Ratios.
`lrr.ylim`	y-axis range within which to plot LRRs.
`plot.trend`	whether to plot a trend line for LRR values.
`density.band.window`	if non-null, overlays a plot of the region that contains the specified fraction of values (`density.band.frac`) within windows the specified size.
`density.band.frac`	the fraction of values to include in the density band.
`other.plots`	plotting function(s) to add additional panels to the BAF and LRR plots.
`other.data`	additional data required by the additional plotting functions.
`...`	additional arguments to pass to panel plotting functions.

`print.summary.CLASPAssay` *Generate a string representation of a summary.CLASPAssay object.*

Description

Generate a string representation of a summary.CLASPAssay object.

Usage

```
## S3 method for class 'summary.CLASPAssay'
print(x, ...)
```

Arguments

`x`	an object of class summary.CLASPAssay
`...`	unused.

`rgb.alpha` *Apply an alpha adjustment to a color*

Description

Apply an alpha adjustment to a color

Usage

```
rgb.alpha(r, alpha = 1)
```

Arguments

`r`	color name (convertible by col2rgb), or vector or matrics of RGB values.
`alpha`	alpha value to apply.

Value

vector of hex colors.

`summary.CLASPAssay` *An S3 summary function for a CLASPAssay object.*

Description

An S3 summary function for a CLASPAssay object.

Usage

```
## S3 method for class 'CLASPAssay'
summary(object, ...)
```

Arguments

`object`	object
`...`	unused

Index

add.samples, 2

compute.dilution.series, 3
compute.metrics, 4
compute.metrics.Illumina, 4
connect.database, 5
copy.number.genoCN, 5
copy.number.genoCN.plot, 6

develop.assay, 7

estimate.contamination, 8
estimate.copy.number, *6*, 9
execute.assay, 10

get.all.arrays, 10
get.all.genomes, 11
get.all.marker.sets, 11
get.all.samples, 12
get.array.id, 12
get.assay, 13
get.genome.id, 13
get.genotypes, 14
get.marker.set, 14
get.markers, 15
get.platform.data, 15
get.samples, 16

import.array, 16
import.genome, 17
import.genotypes, 17
import.marker.set, 18
import.sample.set, 19
import.samples, *19*, 19
init.database, 20
insert.assay, 21
intersect.marker.sets, 21

load.objects, 22

metrics.plot, 22

print.summary.CLASPAssay, 23

rgb.alpha, 24

summary.CLASPAssay, 24