Package ‘strvalidator’

August 12, 2017

Type Package
Title Process Control and Internal Validation of Forensic STR Kits
Version 2.0.0
Date 2017-08-11
Author Oskar Hansson
Maintainer Oskar Hansson <oskhan@ous-hf.no>

URL https://sites.google.com/site/forensicapps/strvalidator

BugReports https://github.com/OskarHansson/strvalidator/issues

Depends R (>= 3.1.3)

Imports ggplot2 (>= 2.0.0), gWidgets2, gridExtra, grid, gtable, plyr, scales, gWidgets2RGtk2, RGtk2, data.table, grDevices, graphics, stats, utils, MASS

Suggests ResourceSelection, testthat

Description An open source platform for validation and process control.
Tools to analyze data from internal validation of forensic short tandem repeat (STR) kits are provided. The tools are developed to provide the necessary data to conform with guidelines for internal validation issued by the European Network of Forensic Science Institutes (ENFSI) DNA Working Group, and the Scientific Working Group on DNA Analysis Methods (SWGDAM). A front-end graphical user interface is provided. More information about each function can be found in the respective help documentation.

License GPL-2

RooxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-11 22:14:43 UTC
R topics documented:

strvalidator-package ........................................ 4
addColor ....................................................... 6
addData ......................................................... 7
addData_gui .................................................... 8
addDye_gui ..................................................... 9
addMarker ...................................................... 9
addMarker_gui .................................................. 10
addOrder ....................................................... 11
addSize ......................................................... 12
addSize_gui ..................................................... 12
auditTrail ....................................................... 13
calculateAllele ................................................ 14
calculateAllele_gui .......................................... 15
calculateAT ..................................................... 16
calculateAT6 .................................................... 17
calculateAT6_gui .............................................. 18
calculateAT_gui ............................................... 19
calculateCapillary ............................................ 20
calculateCapillary_gui ....................................... 21
calculateConcordance ........................................ 21
calculateConcordance_gui ................................... 22
calculateCopies ............................................... 23
calculateCopies_gui ......................................... 24
calculateDropout .............................................. 24
calculateDropout_gui ........................................ 26
calculateHb ..................................................... 27
calculateHb_gui ............................................... 28
calculateHeight ............................................... 28
calculateHeight_gui .......................................... 30
calculateLb ..................................................... 31
calculateLb_gui ............................................... 32
calculateMixture .............................................. 33
calculateMixture_gui ......................................... 34
calculateOL ..................................................... 35
calculateOL_gui ............................................... 36
calculateOverlap ............................................. 36
calculateOverlap_gui ........................................ 37
calculatePeaks ............................................... 38
calculatePeaks_gui .......................................... 39
calculatePullup ............................................... 40
calculatePullup_gui .......................................... 41
calculateRatio ............................................... 41
calculateRatio_gui ........................................... 43
calculateResultType ......................................... 43
calculateResultType_gui .................................... 44
calculateSlope ............................................... 45
R topics documented:

- calculateSlope_gui ........................................... 46
- calculateSpike .............................................. 47
- calculateSpike_gui .......................................... 48
- calculateStutter ............................................... 48
- calculateStutter_gui ......................................... 49
- checkDataset .................................................. 50
- checkSubset .................................................... 51
- checkSubset_gui ............................................... 51
- colConvert ..................................................... 52
- colNames ....................................................... 53
- columns ........................................................ 53
- columns_gui ..................................................... 55
- combineBinsAndPanels ........................................ 55
- combine_gui ..................................................... 56
- cropData_gui .................................................... 57
- detectKit ....................................................... 57
- editData_gui .................................................... 58
- export .......................................................... 59
- export_gui ....................................................... 60
- filterProfile ................................................... 61
- filterProfile_gui ............................................... 62
- generateEPG .................................................... 62
- generateEPG_gui ............................................... 64
- getKit .......................................................... 64
- ggsave_gui ...................................................... 65
- guessProfile .................................................... 66
- guessProfile_gui ............................................... 67
- heightToPeak ................................................... 68
- import .......................................................... 68
- import_gui ....................................................... 70
- listObjects ...................................................... 70
- makeKit_gui .................................................... 71
- maskAT .......................................................... 72
- modelDropout_gui .............................................. 73
- plotAT_gui ....................................................... 75
- plotBalance_gui ............................................... 76
- plotCapillary_gui .............................................. 77
- plotContamination_gui ....................................... 77
- plotDistribution_gui ......................................... 78
- plotDropout_gui ................................................. 79
- plotKit_gui ...................................................... 80
- plotPeaks_gui ................................................... 80
- plotPrecision_gui .............................................. 81
- plotPullup_gui .................................................. 82
- plotRatio_gui .................................................... 83
- plotResultType_gui ............................................ 83
- plotSlope_gui ................................................... 84
- plotStutter_gui ................................................ 85
Description

STR-validator is a free and open source R-package intended for process control and internal validation of forensic STR DNA typing kit. Its graphical user interface simplifies the analysis of data exported from e.g. GeneMapper software, without extensive knowledge about R. It provides functions to import, view, edit, and export data. After analysis the results, generated plots, heat-maps, and data can be saved in a project for easy access. Currently, analysis modules for stutter, balance, dropout, mixture, concordance, typing result, precision, pull-up, and analytical thresholds are available. In addition there are functions to analyze the GeneMapper bins- and panels files. EPG like plots can be generate from data. STR-validator can greatly increase the speed of validation by reducing the time and effort needed to analyze the validation data. It allows exploration of the characteristics of DNA typing kits according to ENFSI and SWGDAM recommendations. This facilitates the implementation of probabilistic interpretation of DNA results.

STR-validator was written and is maintained by Oskar Hansson, senior forensic scientist at Oslo University Hospital (OUS), Section for Forensic Biology. The work initially received external funding from the European Union seventh Framework Programme (FP7/2007-2013) under grant agreement no 285487 (EUROFORGEN-NoE) but development and maintenance is now performed as a part of my position at OUS, and on personal spare time.

Effort has been made to assure correct results. Refer to the main website for a list of functions specifically tested at build time.

Click Index at the bottom of the page to see a complete list of functions.

Created and maintained by:
Oskar Hansson, Section for Forensic Biology (OUS, Norway)

More information can be found at:
https://sites.google.com/site/forensicapps/strvalidator

Info and user community at Facebook:
https://www.facebook.com/groups/strvalidator/

The source code is hosted at GitHub:
https://github.com/OskarHansson/strvalidator

Please report bugs to:
https://github.com/OskarHansson/strvalidator/issues
addColor

Author(s)

Oskar Hansson <oskhan@ous-hf.no>

References


addColor

Add Color Information.

Description

Add color information 'Color', 'Dye' or 'R Color'.

Usage

addColor(data, kit = NA, have = NA, need = NA, overwrite = FALSE, ignore.case = FALSE, debug = FALSE)

Arguments

data: data frame or vector.
kit: string representing the forensic STR kit used. Default is NA, in which case 'have' must contain a valid column.
have: character string to specify color column to be matched. Default is NA, in which case color information is derived from 'kit' and added to a column named 'Color'. If 'data' is a vector 'have' must be a single string.
need: character string or string vector to specify color columns to be added. Default is NA, in which case all columns will be added. If 'data' is a vector 'need' must be a single string.
overwrite: logical if TRUE and column exist it will be overwritten.
ignore.case: logical if TRUE case in marker names will be ignored.
debug: logical indicating printing debug information.

Details

Primers in forensic STR typing kits are labeled with a fluorescent dye. The dyes are represented with single letters (Dye) in exported result files or with strings (Color) in 'panels' files. For visualization in R the R color names are used (R.Color). The function can add new color schemes matched to the existing, or it can convert a vector containing one scheme to another.
Value

data.frame with additional columns for added colors, or vector with converted values.

Examples

# Get marker and colors for SG1 Plus.
df <- getKit("SGMPlus", what="Color")
# Add dye color.
dfDye <- addColor(data=df, need="Dye")
# Add all color alternatives.
dfAll <- addColor(data=df)
# Convert a dye vector to R colors
addColor(data=c("R","G","Y","B"), have="dye", need="r.color")

addData

Adds New Data Columns to a Data Frame

Description

Adds values from columns in 'new.data' to 'data' by keys.

Usage

addData(data, new.data, by.col, then.by.col = NULL, exact = TRUE,
         ignore.case = TRUE, what = NULL, debug = FALSE)

Arguments

data
  Data frame containing your main data.
new.data
  Data frame containing information you want to add to 'data'.
by.col
  character, primary key column.
then.by.col
  character, secondary key column.
exact
  logical, TRUE matches keys exact.
ignore.case
  logical, TRUE ignore case.
what
  character vector defining columns to add. Default is all new columns.
debug
  logical indicating printing debug information.

Details

Information in columns in data frame 'new.data' is added to data frame 'data' based on primary key value in column 'by.col', and optionally on secondary key values in column 'then.by.col'.

Value

data.frame the original data frame containing additional columns.
Examples

```r
# Get marker names and alleles for Promega PowerPlex ESX 17.
x <- getKit("ESX17", what="Allele")
# Get marker names and colors for Promega PowerPlex ESX 17.
y <- getKit("ESX17", what="Color")
# Add color information to allele information.
z <- addData(data=x, new.data=y, by.col="Marker")
print(x)
print(y)
print(z)
```

Description

GUI wrapper for `addData`.

Usage

```r
addData_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

Arguments

- `env` environment in which to search for data frames.
- `savegui` logical indicating if GUI settings should be saved in the environment.
- `debug` logical indicating printing debug information.
- `parent` widget to get focus when finished.

Details

Simplifies the use of the `addData` function by providing a graphical user interface to it.

Value

`TRUE`

See Also

`addData`
addDye_gui

**Description**

GUI wrapper to the addColor function.

**Usage**

```r
addDye_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env` environment in which to search for data frames and save result.
- `savegui` logical indicating if GUI settings should be saved in the environment.
- `debug` logical indicating printing debug information.
- `parent` widget to get focus when finished.

**Details**

Convenience GUI for the use of `addColor` and `addOrder` to add 'Dye', 'Color', 'R.Color', and marker 'Order' to a dataset. 'Dye' is the one letter abbreviations for the fluorophores commonly used to label primers in forensic STR typing kits (e.g. R and Y), 'Color' is the corresponding color name (e.g. red and yellow), 'R.Color' is the plot color used in R (e.g. red and black). 'Order' is the marker order in the selected kit. NB! Existing columns will be overwritten.

**Value**

TRUE

**See Also**

- `addColor`

---

addMarker

**Description**

Add missing markers to a dataset given a set of markers.

**Usage**

```r
addMarker(data, marker, ignore.case = FALSE, debug = FALSE)
```
Arguments

- **data**: data.frame or vector with sample names.
- **marker**: vector with marker names.
- **ignore.case**: logical. TRUE ignores case in marker names.
- **debug**: logical indicating printing debug information.

Details

Given a dataset or a vector with sample names the function loops through each sample and add any missing markers. Returns a dataframe where each sample have at least one row per marker in the specified marker vector. Use `sortMarker` to sort the markers according to a specified kit. Required columns are: 'Sample.Name'.

Value

data.frame.

Description

GUI wrapper for the `addMarker` function.

Usage

```r
addMarker_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

Arguments

- **env**: environment in which to search for data frames and save result.
- **savegui**: logical indicating if GUI settings should be saved in the environment.
- **debug**: logical indicating printing debug information.
- **parent**: widget to get focus when finished.

Details

Simplifies the use of the `addMarker` function by providing a graphical user interface to it.

Value

TRUE

See Also

- `addMarker`
addOrder

Add Marker Order.

Description

Add marker order to data frame containing a column 'Marker'.

Usage

addOrder(data, kit = NULL, overwrite = FALSE, ignore.case = FALSE, debug = FALSE)

Arguments

data data frame or vector.

kit string representing the forensic STR kit used. Default is NULL and automatic
detection of kit will be attempted.

overwrite logical if TRUE and column exist it will be overwritten.

ignore.case logical if TRUE case in marker names will be ignored.

default logical indicating printing debug information.

Details

Markers in a kit appear in a certain order. Not all STR-validator functions keep the original marker
order in the result. A column indicating the marker order is added to the dataset. This is especially
useful when exporting the data to an external spread-sheet software and allow to quickly sort the
data in the correct order.

Value

data.frame with additional numeric column 'Order'.

Examples

# Load a dataset containing two samples.
data("set2")

# Add marker order when kit is known.
addOrder(data = set2, kit = "SGMPlus")
addSize

Add Size Information.

Description
Add size information to alleles.

Usage
addSize(data, kit = NA, bins = TRUE, ignore.case = FALSE, debug = FALSE)

Arguments
data data.frame with at least columns 'Marker' and 'Allele'.
kit data.frame with columns 'Marker', 'Allele', and 'Size' (for bins=TRUE) or 'Marker', 'Allele', 'Offset' and 'Repeat' (for bins=FALSE).
bins logical TRUE alleles get size from corresponding bin. If FALSE the size is calculated from the locus offset and repeat unit.
ignore.case logical TRUE case in marker names are ignored.
debug logical indicating printing debug information.

details
Adds a column 'Size' with the fragment size in base pair (bp) for each allele as estimated from kit bins OR calculated from offset and repeat. The bins option return NA for alleles not in bin. The calculate option handles all named alleles including micro variants (e.g. '9.3'). Handles 'X' and 'Y' by replacing them with '1' and '2'.

Value
data.frame with additional columns for added size.

addSize_gui

Add Size Information

Description
GUI wrapper for the addSize function.

Usage
addSize_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
Arguments

- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

Details

Simplifies the use of the `addsize` function by providing a graphical user interface to it.

Value

TRUE

See Also

`addsize`

---

### Description

Adds an audit trail to a dataset.

### Usage

```r
auditTrail(obj, f.call = NULL, key = NULL, value = NULL, label = NULL,
            arguments = TRUE, exact = TRUE, remove = FALSE, package = NULL)
```

### Arguments

- `obj`: object to add or update the audit trail.
- `f.call`: the function call i.e. `match.call()`.
- `key`: list or vector of additional keys to log.
- `value`: list or vector of additional values to log.
- `label`: optional label used if `f.call=NULL`.
- `arguments`: logical. TRUE log function arguments.
- `exact`: logical for exact matching of attribute name.
- `remove`: logical. If TRUE the 'audit trail' attribute is removed.
- `package`: character to log the package version.
calculate Allele

Details

Automatically add or updates an attribute 'audit trail' with arguments and parameters extracted from the function call. To list the arguments with the default set but not overridden arguments=TRUE must be set (default). Additional custom key-value pairs can be added. The label is extracted from the function name from f.call. Specify package to include the version number of a package.

Value

object with added or updated attribute 'audit trail'.

Examples

# A simple function with audit trail logging.
myFunction <- function(x, a, b=5){
  x <- x + a + b
  x <- auditTrail(obj = x, f.call = match.call(), package = "strvalidator")
  return(x)
}
# Run the function.
myData <- myFunction(x=10, a=2)
# Check the audit trail.
cat(attr(myData, "audit trail"))

# Remove the audit trail.
myData <- auditTrail(myData, remove=TRUE)
# Confirm that the audit trail is removed.
cat(attr(myData, "audit trail"))

calculateAllele Calculate Allele

Description

Calculates summary statistics for alleles per marker over the entire dataset.

Usage

calculateAllele(data, threshold = NULL, sex.rm = FALSE, kit = NULL,
  debug = FALSE)

Arguments

data data.frame including columns 'Marker' and 'Allele', and optionally 'Height' and 'Size'.
threshold numeric if not NULL only peak heights above 'threshold' will be considered.
sex.rm logical TRUE removes all sex markers. Requires 'kit'.
kit character for the DNA typing kit defining the sex markers.
debug logical indicating printing debug information.
Details

Creates a table of the alleles in the dataset sorted by number of observations. For each allele the proportion of total observations is calculated. Using a threshold this can be used to separate likely artefacts from likely drop-in peaks. In addition the observed allele frequency is calculated. If columns 'Height' and/or 'Size' are available summary statistics is calculated. NB! The function removes NA's and OL's prior to analysis.

Value


See Also

data.table


seeAlso

data.table

Description

GUI wrapper for the calculateAllele function.

Usage

calculateAllele_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments
/env
environment in which to search for data frames.
/savegui
logical indicating if GUI settings should be saved in the environment.
/debug
logical indicating printing debug information.
/parent
widget to get focus when finished.

Details

Simplifies the use of the calculateAllele function by providing a graphical user interface to it.

Value

TRUE
**calculateAT**

**Calculate Analytical Threshold**

**Description**

Calculate analytical thresholds estimates.

**Usage**

```r
calculateAT(data, ref = NULL, mask.height = TRUE, height = 500,
mask.sample = TRUE, per.dye = TRUE, range.sample = 20,
mask.ils = TRUE, range.ils = 10, k = 3, rank.t = 0.99, alpha = 0.01,
ignore.case = TRUE, word = FALSE, debug = FALSE)
```

**Arguments**

- **data**: a data frame containing at least 'Dye.Sample.Peak', 'Sample.File.Name', 'Marker', 'Allele', 'Height', and 'Data.Point'.
- **ref**: a data frame containing at least 'Sample.Name', 'Marker', 'Allele'.
- **mask.height**: logical to indicate if high peaks should be masked.
- **height**: integer for global lower peak height threshold for peaks to be excluded from the analysis. Active if `mask.peak=TRUE`.
- **mask.sample**: logical to indicate if sample allelic peaks should be masked.
- **per.dye**: logical TRUE if sample peaks should be masked per dye channel. FALSE if sample peaks should be masked globally across dye channels.
- **range.sample**: integer to specify the masking range in (+/-) data points. Active if mask.sample=TRUE.
- **mask.ils**: logical to indicate if internal lane standard peaks should be masked.
- **range.ils**: integer to specify the masking range in (+/-) data points. Active if mask.ils=TRUE.
- **k**: numeric factor for the desired confidence level (method AT1).
- **rank.t**: numeric percentile rank threshold (method AT2).
- **alpha**: numeric one-sided confidence interval to obtain the critical value from the t-distribution (method AT4).
- **ignore.case**: logical to indicate if sample matching should ignore case.
- **word**: logical to indicate if word boundaries should be added before sample matching.
- **debug**: logical to indicate if debug information should be printed.

**Details**

Calculate the analytical threshold (AT) according to method 1, 2, and 4 as recommended in the reference by analyzing the background signal (noise). In addition method 7, a log-normal version of method 1 has been implemented. Method 1: The average signal + 'k' * the standard deviation. Method 2: The percentile rank method. The percentage of noise peaks below 'rank.t'. Method 4: Utilize the mean and standard deviation and the critical value obtained from the t-distribution for
confidence interval 'alpha' (one-sided) and observed peaks analyzed (i.e. not masked) minus one as degrees of freedom, and the number of samples. Method 7: The average natural logarithm of the signal + \( k \) * the standard deviation.

If samples containing DNA are used, a range around the allelic peaks can be masked from the analysis to discard peaks higher than the noise. Masking can be within each dye or across all dye channels. Similarly a range around the peaks of the internal lane standard (ILS) can be masked across all dye channels. Which can bleed-through in weak samples (i.e. negative controls) The mean, standard deviation, and number of peaks are calculated per dye per sample, per sample, globally across all samples, and globally across all samples per dye, for each method to estimate AT. Also the complete percentile rank list is calculated.

Value

list of three data frames. The first with result per dye per sample, per sample, globally across all samples, and globally across all samples per dye, for each method. The second is the complete percentile rank list. The third is the masked raw data used for calculation to enable manual check of the result.

References


See Also

calculateAT6, maskAT, checkSubset

calculateAT6

Calculate Analytical Threshold

Description

Calculate analytical thresholds estimate using linear regression.

Usage

calculateAT6(data, ref, amount = NULL, weighted = TRUE, alpha = 0.05, ignore.case = TRUE, debug = FALSE)

Arguments

- **data**: data.frame containing at least columns 'Sample.Name', 'Marker', 'Allele', and 'Height'.
- **ref**: data.frame containing at least columns 'Sample.Name', 'Marker', and 'Allele'.
- **amount**: data.frame containing at least columns 'Sample.Name' and 'Amount'. If NULL 'data' must contain a column 'Amount'.
Calculate the analytical threshold (AT) according to method 6 as outlined in the reference. In short serial dilutions are analyzed and the average peak height is calculated. Linear regression or Weighted linear regression with amount of DNA as the predictor for the peak height is performed. Method 6: A simplified version of the upper limit approach. \( \text{AT6} = y\text{-intercept} + t\text{-statistic} \times \text{standard error of the regression} \). Assumes the y-intercept is not different from the mean blank signal. The mean blank signal should be included in the confidence range ('Lower' to 'AT6' in the resulting data frame). NB! This is an indirect method to estimate AT and should be verified by other methods. From the reference: A way to determine the validity of this approach is based on whether the y-intercept \(+ (1-a)100\) contains the mean blank signal. If the mean blank signal is included in the y-intercept band, the following relationship [i.e. AT6] can be used to determine the AT. However, it should be noted that the ATs derived in this manner need to be calculated for each color and for all preparations (i.e., different injections, sample preparation volumes, post-PCR cleanup, etc.). NB! Quality sensors must be removed prior to analysis.

Value

data.frame with columns 'Amount', 'Height', 'Sd', 'Weight', 'N', 'Alpha', 'Lower', 'Intercept', and 'AT6'.

References


See Also

calculateAT6_gui, calculateAT, calculateAT_gui, lm

Description

GUI wrapper for the calculateAT6 function.

Usage

calculateAT6_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
**calculateAT_gui**

**Arguments**

- **env**
  environment in which to search for data frames and save result.
- **savegui**
  logical indicating if GUI settings should be saved in the environment.
- **debug**
  logical indicating printing debug information.
- **parent**
  widget to get focus when finished.

**Details**

Scores dropouts for a dataset.

**Value**

TRUE

**See Also**

`calculateAT6`, `calculateAT`, `calculateAT_gui`, `checkSubset`

---

**calculateAT_gui**  
*Calculate Analytical Threshold*

**Description**

GUI wrapper for the `maskAT` and `calculateAT` function.

**Usage**

```r
calculateAT_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- **env**
  environment in which to search for data frames and save result.
- **savegui**
  logical indicating if GUI settings should be saved in the environment.
- **debug**
  logical indicating printing debug information.
- **parent**
  widget to get focus when finished.

**Details**

Simplifies the use of the `calculateAT` and `calculateAT` function by providing a graphical user interface. In addition there are integrated control functions.

**Value**

TRUE
calculateCapillary  

Calculate Capillary Balance

Description

Calculates the ILS inter capillary balance.

Usage

calculateCapillary(samples.table, plot.table, sq = 0, run = "", debug = FALSE)

Arguments

samples.table  data frame containing at least the columns 'Sample.File', 'Sample.Name', 'Size.Standard', 'Instrument.Type', 'Instrument.ID', 'Cap', 'Well', and 'SQ'.
plot.table   data frame containing at least the columns 'Sample.File.Name', 'Size', and 'Height'.
sq   numeric threshold for 'Sizing Quality' (SQ).
run   character string for run name.
debug   logical indicating printing debug information.

Details

Calculates the inter capillary balance for the internal lane standard (ILS). Require information from both the 'samples.table' and the 'plot.table'.

Value

data.frame with with columns 'Instrument', 'Instrument.ID', 'Run', 'Mean.Height', 'SQ', 'Injec-
tion', 'Capillary', 'Well', 'Comment'.

See Also

calculateAT, maskAT, checkSubset
**calculateCapillary_gui**

*Calculate Capillary Balance*

**Description**

GUI wrapper for the `calculateCapillary` function.

**Usage**

```r
calculateCapillary_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

**Details**

Simplifies the use of the `calculateCapillary` function by providing a graphical user interface.

**Value**

`TRUE`  

**See Also**

- `calculateCapillary`

---

**calculateConcordance**

*Calculate Concordance.*

**Description**

Calculates concordance and discordance for profiles in multiple datasets.

**Usage**

```r
calculateConcordance(data, kit.name = NA, no.marker = "NO MARKER", no.sample = "NO SAMPLE", delimeter = ",", list.all = FALSE, debug = FALSE)
```
Arguments

data 

list of data frames in 'slim' format with at least columns 'Sample.Name', 'Marker', and 'Allele'.

kit.name 

character vector for DNA typing kit names in same order and of same lengths as data sets in 'data' list. Default is NA in which case they will be numbered.

no.marker 

character vector for string when marker is missing.

no.sample 

character vector for string when sample is missing.

delimeter 

character to separate the alleles in a genotype. Default is comma e.g '12,16'.

list.all 

logical TRUE to return missing samples.

debug 

logical indicating printing debug information.

Details

Takes a list of datasets as input. It is assumed that each unique sample name represent a result originating from the same source DNA and thus is expected to give identical DNA profiles. The function first compare the profiles for each sample across datasets and lists discordant results. Then it performs a pair-wise comparison and compiles a concordance table. The tables are returned as two data frames in a list. NB! Typing and PCR artefacts (spikes, off-ladder peaks, stutters etc.) must be removed before analysis. NB! It is expected that the unique set of marker names across a dataset is present in each sample for that dataset (a missing marker is a discordance).

Value

list of data.frames (discordance table, and pair-wise comparison).

Description

GUI wrapper for the calculateConcordance function.

Usage

calculateConcordance_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env 

environment in which to search for data frames and save result.

savegui 

logical indicating if GUI settings should be saved in the environment.

debug 

logical indicating printing debug information.

parent 

widget to get focus when finished.
calculateCopies

Details
Simplifies the use of the calculateConcordance function by providing a graphical user interface.

Value
TRUE

See Also
calculateConcordance

---

calculateCopies Calculate Allele Copies

Description
Calculates the number of alleles in each marker.

Usage
calculateCopies(data, observed = FALSE, copies = TRUE,
    heterozygous = FALSE, debug = FALSE)

Arguments
data Data frame containing at least columns 'Sample.Name', 'Marker', and 'Allele*'.
observed logical indicating if a column 'Observed' should be used to count the number of
    unique alleles.
copies logical indicating if a column 'Copies' should be used to indicate the number of
    allele copies, '1' for heterozygotes and '2' for homozygotes.
heterozygous logical indicating if a column 'Heterozygous' should be used to indicate het-
    erozygotes with '1' and homozygotes with '0'.
debug logical indicating printing debug information.

Details
Calculates the number of unique values in the 'Allele*' columns for each marker, the number of
allele copies, or indicate heterozygous loci. Observed - number of unique alleles. Copies - number
of allele copies, '1' for heterozygotes and '2' for homozygotes. Heterozygous - '1' for heterozygous
and '0' for homozygous loci. NB! The 'copies' and 'heterozygous' option are intended for known
complete profiles, while 'observed' can be used for any samples to count the number of peaks.
Sample names must be unique. The result is per marker but repeated for each row of that marker.
Data in 'fat' format is auto slimmed.

Value
data.frame the original data frame with optional columns 'Observed', 'Copies', and 'Heterozygous'.
calculateCopies_gui  

*Calculate Allele Copies*

**Description**

GUI wrapper for the `calculateCopies` function.

**Usage**

```r
calculateCopies_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env`  
  environment in which to search for data frames and save result.
- `savegui`  
  logical indicating if GUI settings should be saved in the environment.
- `debug`  
  logical indicating printing debug information.
- `parent`  
  widget to get focus when finished.

**Details**

Simplifies the use of the `calculateCopies` function by providing a graphical user interface to it.

**Value**

`TRUE`

**See Also**

`calculateCopies`

---

calculateDropout  

*Calculate Drop-out Events*

**Description**

Calculate drop-out events (allele and locus) and records the surviving peak height.

**Usage**

```r
calculateDropout(data, ref, threshold = NULL, method = c("1", "2", "X", "L"), ignore.case = TRUE, sex.rm = FALSE, qs.rm = TRUE, kit = NULL, debug = FALSE)
```

---
calculateDropout

Arguments

data: data frame in GeneMapper format containing at least a column 'Allele'.
ref: data frame in GeneMapper format.
threshold: numeric, threshold in RFU defining a dropout event. Default is 'NULL' and dropout is scored purely on the absence of a peak.
method: character vector, specifying which scoring method(s) to use. Method 'X' for random allele, '1' or '2' for the low/high molecular weight allele, and 'L' for the locus method (the option is case insensitive).
ignore.case: logical, default TRUE for case insensitive.
sex.rm: logical, default FALSE to include sex markers in the analysis.
qs.rm: logical, default TRUE to exclude quality sensors from the analysis.
kit: character, required if sex.rm=TRUE or qs.rm=TRUE to define the kit.
debug: logical indicating printing debug information.

Details

Calculates drop-out events. In case of allele dropout the peak height of the surviving allele is given. Homozygous alleles in the reference set can be either single or double notation (X or X X). Markers present in the reference set but not in the data set will be added to the result. NB! 'Sample.Name' in 'ref' must be unique core name of replicate sample names in 'data'. Use checkSubset to make sure subsetting works as intended. There are options to remove sex markers and quality sensors from analysis.

NB! There are several methods of scoring drop-out events for regression. Currently the 'MethodX', 'Method1', and 'Method2' are endorsed by the DNA commission (see Appendix B in ref 1). However, an alternative method is to consider the whole locus and score drop-out if any allele is missing.

Explanation of the methods: Dropout - all alleles are scored according to LDT. This is pure observations and is not used for modeling. MethodX - a random reference allele is selected and drop-out is scored in relation to the the partner allele. Method1 - the low molecular weight allele is selected and drop-out is scored in relation to the partner allele. Method2 - the high molecular weight allele is selected and drop-out is scored in relation to the partner allele. MethodL - drop-out is scored per locus i.e. drop-out if any allele has dropped out.

Method X/1/2 records the peak height of the partner allele to be used as the explanatory variable in the logistic regression. The locus method L also do this when there has been a drop-out, if not the the mean peak height for the locus is used. Peak heights for the locus method are stored in a separate column.

Value

data.frame with columns 'Sample.Name', 'Marker', 'Allele', 'Height', 'Dropout', 'Rfu', 'Heterozygous', and 'Model'. Dropout: 0 indicate no dropout, 1 indicate allele dropout, and 2 indicate locus dropout. Rfu: height of surviving allele. Heterozygous: 1 for heterozygous and 0 for homozygous. And any of the following containing the response (or explanatory) variable used for modeling by logistic regression in function modeldropout: 'MethodX', 'Method1', 'Method2', 'MethodL' and 'MethodL.Ph'.

functions:
calculatedropout......
References


Examples

data(set4)
data(ref4)
drop <- calculateDropout(data=set4, ref=ref4, kit="ESX17", ignore.case=TRUE)

calculateDropout_gui  Calculate Dropout Events

Description

GUI wrapper for the calculateDropout function.

Usage

calculateDropout_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env environment in which to search for data frames and save result.
savegui logical indicating if GUI settings should be saved in the environment.
debug logical indicating printing debug information.
parent widget to get focus when finished.

Details

Scores dropouts for a dataset.

Value

TRUE

See Also

calculateDropout, checkSubset
**calculateHb**  
*Calculate Heterozygote Balance*

**Description**
Calculates the heterozygote (intra-locus) peak balance.

**Usage**
```
calculateHb(data, ref, hb = 1, kit = NULL, sex.rm = FALSE,  
qs.rm = FALSE, ignore.case = TRUE, exact = FALSE, word = FALSE,  
debug = FALSE)
```

**Arguments**
- `data`: a data frame containing at least 'Sample.Name', 'Marker', 'Height', and 'Allele'.
- `ref`: a data frame containing at least 'Sample.Name', 'Marker', 'Allele'.
- `hb`: numerical, definition of heterozygote balance. Default is hb=1. hb=1: HMW/LMW,  
hb=2: LMW/HMW, hb=3: min(Ph)/max(Ph).
- `kit`: character defining the kit used. If NULL automatic detection is attempted.
- `sex.rm`: logical TRUE removes sex markers defined by 'kit'.
- `qs.rm`: logical TRUE removes quality sensors defined by 'kit'.
- `ignore.case`: logical indicating if sample matching should ignore case.
- `exact`: logical indicating if exact sample matching should be used.
- `word`: logical indicating if word boundaries should be added before sample matching.
- `debug`: logical indicating printing debug information.

**Details**
Calculates the heterozygote (intra-locus) peak balance for a dataset. Known allele peaks will be 
extracted using the reference prior to analysis. Calculates the heterozygote balance (Hb), size dif-
ference between heterozygous alleles (Delta), and mean peak height (MPH). NB! 'X' and 'Y' will 
be handled as '1' and '2' respectively.

**Value**
`data.frame` with with columns 'Sample.Name', 'Marker', 'Delta', 'Hb', 'MPH'.

**Examples**
```r
data(ref2)
data(set2)
# Calculate average balances.
calculateHb(data=set2, ref=ref2)
```
### calculateHb_gui

**Calculate Heterozygote Balance**

#### Description

GUI wrapper for the `calculateHb` function.

#### Usage

```r
calculateHb_gui(env = parent.frame(), savegui = NULL, debug = FALSE, 
    parent = NULL)
```

#### Arguments

- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

#### Details

Simplifies the use of the `calculateHb` function by providing a graphical user interface.

#### Value

TRUE

#### See Also

`link{calculateHb}`, `link{checkSubset}`

### calculateHeight

**Calculate Peak Height.**

#### Description

Calculate peak height metrics for samples.

#### Usage

```r
calculateHeight(data, ref = NULL, na.replace = NULL, add = TRUE, 
    exclude = NULL, sex.rm = FALSE, qs.rm = FALSE, kit = NULL, 
    ignore.case = TRUE, exact = FALSE, word = FALSE, debug = FALSE)
```
calculateHeight

Arguments

data  data.frame with at least columns 'Sample.Name' and 'Height'.
ref  data.frame with at least columns 'Sample.Name' and 'Allele'.
na.replace  replaces NA values in the final result.
add  logical default is TRUE which will add or overwrite columns 'TPH', 'Peaks', 'H', and 'Proportion' in the provided 'data'.
exclude  character vector (case sensitive) e.g. "OL" excludes rows with "OL" in the 'Allele' column.
sex.rm  logical, default FALSE to include sex markers in the analysis.
qs.rm  logical, default TRUE to exclude quality sensors from the analysis.
k Kit  character, required if sex.rm=TRUE or qs.rm=TRUE to define the kit.
ignore.case  logical TRUE ignores case in sample name matching.
exact  logical TRUE for exact sample name matching.
word  logical TRUE to add word boundaries to sample name matching.
debug  logical indicating printing debug information.

Details

Calculates the total peak height (TPH), and number of observed peaks (Peaks), for each sample by default. If a reference dataset is provided average peak height (H), and profile proportion (Proportion) are calculated.

H is calculated according to the formula: $H = \frac{\text{sum(peakheights)}}{n[\text{het}]} + 2n[\text{hom}]$ Where:
- $n[\text{het}]$ = number of observed heterozygous alleles
- $n[\text{hom}]$ = number of observed homozygous alleles

Tip: If it is known that all expected peaks are observed and no unexpected peaks are present, the dataset can be used as a reference for itself.

Note: If a reference dataset is provided the known alleles will be extracted from the dataset.

Value

data.frame with with at least columns 'Sample.Name', 'TPH', and 'Peaks'.

References

**calculateHeight_gui**  
*Calculate Peak Height*

**Description**

GUI wrapper for the `calculateHeight` function.

**Usage**

```r
calculateHeight_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env`  
  environment in which to search for data frames and save result.

- `savegui`  
  logical indicating if GUI settings should be saved in the environment.

- `debug`  
  logical indicating printing debug information.

- `parent`  
  widget to get focus when finished.

**Details**

Simplifies the use of the `calculateHeight` function by providing a graphical user interface to it.

**Value**

`TRUE`

**References**

http://dx.doi.org/10.1111/j.1467-9876.2010.00722.x

**See Also**

`calculateHeight`
**calculateLb**  

**Calculate Inter-locus Balance**

**Description**

Calculates the inter-locus balance.

**Usage**

```r
calculateLb(data, ref = NULL, option = "prop", by.dye = FALSE,  
ol.rm = TRUE, sex.rm = FALSE, qs.rm = FALSE, na = NULL, kit = NULL,  
ignore.case = TRUE, word = FALSE, exact = FALSE, debug = FALSE)
```

**Arguments**

- **data**: data.frame containing at least 'Sample.Name', 'Marker', and 'Height'.
- **ref**: data.frame containing at least 'Sample.Name', 'Marker', 'Allele'. If provided alleles matching 'ref' will be extracted from 'data' (see `filterProfile`).
- **option**: character: 'prop' for proportional Lb, 'norm' for normalized LB, and 'cent' for centred Lb.
- **by.dye**: logical. Default is FALSE for global Lb, if TRUE Lb is calculated within each dye channel.
- **ol.rm**: logical. Default is TRUE indicating that off-ladder 'OL' alleles will be removed.
- **sex.rm**: logical. Default is FALSE indicating that all markers will be considered. If TRUE sex markers will be removed.
- **qs.rm**: logical. Default is TRUE indicating that all quality sensors will be removed.
- **na**: numeric. Numeric to replace NA values e.g. locus dropout can be given a peak height equal to the limit of detection threshold, or zero. Default is NULL indicating that NA will be treated as missing values.
- **kit**: character providing the kit name. Attempt to auto detect if NULL.
- **ignore.case**: logical indicating if sample matching should ignore case. Only used if 'ref' is provided and 'data' is filtered.
- **word**: logical indicating if word boundaries should be added before sample matching. Only used if 'ref' is provided and 'data' is filtered.
- **exact**: logical indicating if exact sample matching should be used. Only used if 'ref' is provided and 'data' is filtered.
- **debug**: logical indicating printing debug information.
The inter-locus balance (Lb), or profile balance, can be calculated as a proportion of the whole, normalized, or as centred quantities (as in the reference but using the mean total marker peak height instead of H). Lb can be calculated globally across the complete profile or within each dye channel. All markers must be present in each sample. Data can be unfiltered or filtered since the sum of peak heights by marker is used. A reference dataset is required to filter the dataset, which also adds any missing markers. A kit should be provided for filtering of known profile, sex markers, or quality sensors. If not automatic detection will be attempted. If missing, dye will be added according to kit. Off-ladder alleles and quality sensors are by default removed from the dataset. Sex markers are optionally removed. Some columns in the result may vary: TPH: Total (marker) Peak Height. TPPH: Total Profile Peak Height. MTPH: Maximum (sample) Total Peak Height. MPH: Mean (marker) Peak Height.

Value
data.frame with at least columns 'Sample.Name', 'Marker', 'TPH', 'Peaks', and 'Lb'. See description for additional columns.

References

Examples
# Load data.
data(set2)

# Calculate inter-locus balance.
res <- calculateLb(data = set2)
print(res)
calculateMixture

Arguments

env  environment in which to search for data frames and save result.
savegui  logical indicating if GUI settings should be saved in the environment.
debug  logical indicating printing debug information.
parent  widget to get focus when finished.

Details

Simplifies the use of the `calculateLB` function by providing a graphical user interface.

Value

TRUE

See Also

`link{calculateLB}`, `link{checkSubset}`

calculateMixture  

Calculate Mixture.

Description

Calculate Mx, drop-in, and

Usage

calculateMixture(data, ref1, ref2, ol.rm = TRUE, ignore.dropout = TRUE, debug = FALSE)

Arguments

data  list of data frames in 'slim' format with at least columns 'Sample.Name', 'Marker', and 'Allele'.
ref1  data.frame with known genotypes for the major contributor.
ref2  data.frame with known genotypes for the minor contributor.
ol.rm  logical TRUE removes off-ladder alleles (OL), FALSE count OL as drop-in.
ignore.dropout  logical TRUE calculate Mx also if there are missing alleles.
debug  logical indicating printing debug information.
Details

Given a set of mixture results, reference profiles for the major component, and reference profile for the minor component the function calculates the mixture proportion (Mx), the average Mx, the absolute difference D=|Mx-AvgMx| for each marker, the percentage profile for the minor component, number of drop-ins. The observed and expected number of free alleles for the minor component (used to calculate the profile percentage) is also given.

NB! All sample names must be unique within and between each reference dataset. NB! Samples in ref1 and ref2 must be in 'sync'. The first sample in ref1 is combined with the first sample in ref2 to make a mixture sample. For example: ref1 "A" and ref2 "B" match mixture samples "A_B_1", "A_B_2" and so on. NB! If reference datasets have unequal number of unique samples the smaller dataset will limit the calculation.

Mixture proportion is calculated in accordance with:
Locus style (minor:MAJOR) | Mx
AA:AB | (A-B)/(A+B)
AB:AA | (2*B)/(A+B)
AB:AC | B/(B+C)
AA:BB | A/(A+B)
AB:CC | (A+B)/(A+B+C)
AB:CD | (A+B)/(A+B+C+D)
AB:AB | NA - cannot be calculated
AA:AA | NA - cannot be calculated

Value
data.frame with columns 'Sample.Name', 'Marker', 'Style', 'Mx', 'Average', 'Difference', 'Observed', 'Expected', 'Profile', and 'Dropin'.

References

j.fsigen.2009.07.002

Calculate Mixture

calculateMixture_gui

Description

GUI wrapper for the calculateMixture function.

Usage

calculateMixture_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
Arguments

- env: environment in which to search for data frames and save result.
- savegui: logical indicating if GUI settings should be saved in the environment.
- debug: logical indicating printing debug information.
- parent: widget to get focus when finished.

Details

Simplifies the use of the `calculateMixture` function by providing a graphical user interface.

Value

TRUE

See Also

`calculateMixture`, `checkSubset`

calculateOL  Analyze Off-ladder Alleles

Description

Analyze the risk for off-ladder alleles.

Usage

```r
calculateOL(kit, db, virtual = TRUE, limit = TRUE, debug = FALSE)
```

Arguments

- kit: data.frame, providing kit information.
- db: data.frame, allele frequency database.
- virtual: logical default is TRUE, calculation includes virtual alleles.
- limit: logical default is TRUE, limit small frequencies to 5/2N.
- debug: logical indicating printing debug information.

Details

By analyzing the allelic ladders the risk for getting off-ladder (OL) alleles are calculated. The frequencies from a provided population database is used to calculate the risk per marker and in total for the given kit(s). Virtual alleles can be excluded from the calculation. Small frequencies can be limited to the estimate 5/2N.

Value

data.frame with columns 'Kit', 'Marker', 'Database', 'Risk', and 'Total'.
**calculateOL_gui**  
*Analyze Off-ladder Alleles*

**Description**
GUI wrapper for the `calculateOL` function.

**Usage**
```r
calculateOL_gui(env = parent.frame(), savegui = NULL, debug = TRUE, parent = NULL)
```

**Arguments**
- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

**Details**
By analysis of the allelic ladder the risk for getting off-ladder (OL) alleles are calculated. The frequencies from a provided population database is used to calculate the risk per marker and in total for the given kit(s). Virtual alleles can be excluded from the calculation. Small frequencies can be limited to the estimate $5/2N$.

**Value**
TRUE

**See Also**
- `calculateOL`

---

**calculateOverlap**  
*Calculate Bins Overlap*

**Description**
Analyses the bins overlap between colors.

**Usage**
```r
calculateOverlap(data, db = NULL, penalty = NULL, virtual = TRUE, debug = FALSE)
```
Arguments

- **data**: data frame providing kit information.
- **db**: data frame allele frequency database.
- **penalty**: vector with factors for reducing the impact from distant dye channels. NB! Length must equal number of dyes in kit minus one.
- **virtual**: logical default is TRUE meaning that overlap calculation includes virtual bins.
- **debug**: logical indicating printing debug information.

Details

By analyzing the bins overlap between dye channels a measure of the risk for spectral pull-up artefacts can be obtain. The default result is a matrix with the total bins overlap in number of base pairs. If an allele frequency database is provided the overlap at each bin is multiplied with the frequency of the corresponding allele. If no frequency exist for that allele a frequency of $5/2N$ will be used. X and Y alleles is given the frequency 1. A penalty matrix can be supplied to reduce the effect by spectral distance, meaning that overlap with the neighboring dye can be counted in full (100 while a non neighbor dye get its overlap reduced (to e.g. 10

Value

Data frame with columns 'Kit', 'Color', [dyes], 'Sum', and 'Score'.
Details

By analysis of the bins overlap between dye channels a measure of the risk for spectral pull-up artefacts can be obtained. The default result is a matrix with the total bins overlap in number of base pairs. If an allele frequency database is provided the overlap at each bin is multiplied with the frequency of the corresponding allele. If no frequency exist for that allele a frequency of 5/2N will be used. X and Y alleles is given the frequency 1. A scoring matrix can be supplied to reduce the effect by spectral distance, meaning that overlap with the neighboring dye can be counted in full (100 while a non neighbor dye get its overlap reduced (to e.g. 10

Value

TRUE

See Also

calculateOverlap

calculatePeaks

Calculate Peaks

Description

Calculates the number of peaks in samples.

Usage

calculatePeaks(data, bins = c(0, 2, 3), labels = NULL, ol.rm = FALSE, 
by.marker = FALSE, debug = FALSE)

Arguments

data data frame containing at least the columns 'Sample.Name' and 'Height'.
bins numeric vector containing the cut-off points defined as maximum number of 
peaks for all but the last label, which is anything above final cut-off. Must be 
sorted in ascending order.
labels character vector defining the group labels. Length must be equal to number of 
bins + one label for anything above the final cut-off.
ol.rm logical if TRUE, off-ladder alleles 'OL' peaks will be discarded. if FALSE, all 
peaks will be included in the calculations.
by.marker logical if TRUE, peaks will counted per marker. if FALSE, peaks will counted 
per sample.
debug logical indicating printing debug information.
count the number of peaks in a sample profile based on values in the 'Height' column. Each sample is labeled according to custom labels defined by the number of peaks. Peaks can be counted by sample or by marker within a sample. There is an option to discard off-ladder peaks ("OL"). The default purpose for this function is to categorize contamination in negative controls, but it can be used to simply calculating the number of peaks in any sample. NB! A column 'Peaks' for the number of peaks will be created. If present it will be overwritten. NB! A column 'Group' for the sample group will be created. If present it will be overwritten. NB! A column 'Id' will be created by combining the content in the 'Sample.Name' and 'File' column (if available). The unique entries in the 'Id' column will be the definition of a unique sample. If 'File' is present this allows for identical sample names in different batches (files) to be identified as unique samples. If 'Id' is present it will be overwritten.

Value
data.frame with with additional columns 'Peaks', 'Group', and 'Id'.

**calculatePeaks_gui**

**Calculate Peaks**

**Description**

GUI wrapper for the calculatePeaks function.

**Usage**

```
calculatePeaks_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env` environment in which to search for data frames.
- `savegui` logical indicating if GUI settings should be saved in the environment.
- `debug` logical indicating printing debug information.
- `parent` widget to get focus when finished.

**Details**

Counts the number of peaks in samples and markers with option to discard off-ladder peaks and to label groups according to maximum number of peaks.

**Value**

TRUE

**See Also**

calculatePeaks
calculatePullup  

*Calculate Spectral Pull-up*

**Description**

Calculates possible pull-up peaks.

**Usage**

```r
calculatePullup(data, ref, pullup.range = 6, block.range = 12,
ol.rm = FALSE, ignore.case = TRUE, word = FALSE, discard = FALSE,
limit = 1, debug = FALSE)
```

**Arguments**

- `data` a data frame containing at least 'Sample.Name', 'Marker', 'Height', 'Allele', 'Dye', 'Data.Point' and 'Size'.
- `ref` a data frame containing at least 'Sample.Name', 'Marker', 'Allele'.
- `pullup.range` numeric to set the analysis window to look for pull-up peaks (known allele data point +- pullup.range/2)
- `block.range` numeric to set blocking range to check for known allele overlap (known allele data point +- block.range/2).
- `ol.rm` logical TRUE if off-ladder peaks should be excluded from analysis. Default is FALSE to include off-ladder peaks.
- `ignore.case` logical indicating if sample matching should ignore case.
- `word` logical indicating if word boundaries should be added before sample matching.
- `discard` logical TRUE if known alleles with no detected pull-up should be discarded from the result. Default is FALSE to include alleles not causing pull-up.
- `limit` numeric remove ratios > limit from the result. Default is 1 to remove pull-up peaks that are higher than the source peak and hence likely not a real pull-up.
- `debug` logical indicating printing debug information.

**Details**

Calculates possible pull-up (aka. bleed-through) peaks in a dataset. Known alleles are identified and the analysis window range is marked. If the blocking range of known alleles overlap, they are excluded from the analysis. Pull-up peaks within the data point analysis window, around known alleles, are identified, the data point difference, and the ratio is calculated. Off-ladder ('OL') alleles are included by default but can be excluded. All known peaks included in the analysis are by default written to the result even if they did not cause any pull-up. These rows can be discarded from the result.

**Value**

**calculatePullup_gui**  
*Calculate Spectral Pull-up*

**Description**

GUI wrapper for the `calculatePullup` function.

**Usage**

```r
calculatePullup_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

**Details**

Simplifies the use of the `calculatePullup` function by providing a graphical user interface.

**Value**

TRUE

**See Also**

`calculatePullup`, `checkSubset`

---

**calculateRatio**  
*Calculate Ratio*

**Description**

Calculates the peak height ratio between specified loci.

**Usage**

```r
calculateRatio(data, ref = NULL, numerator = NULL, denominator = NULL,  
               group = NULL, ol.rm = TRUE, ignore.case = TRUE, word = FALSE,  
               exact = FALSE, debug = FALSE)
```
calculateRatio

Arguments

data  a data frame containing at least 'Sample.Name', 'Marker', 'Height', 'Allele'.
ref  a data frame containing at least 'Sample.Name', 'Marker', 'Allele'. If provided alleles matching 'ref' will be extracted from 'data' (see filterProfile).
numerator  character vector with marker names.
denominator  character vector with marker names.
group  character column name to group by.
ol.rm  logical indicating if off-ladder 'OL' alleles should be removed.
ignore.case  logical indicating if sample matching should ignore case.
word  logical indicating if word boundaries should be added before sample matching.
exact  logical indicating if exact sample matching should be used.
debug  logical indicating printing debug information.

Details

Default is to calculate the ratio between all unique pairwise combinations of markers/loci. If equal number of markers are provided in the numerator and the denominator the provided pairwise ratios will be calculated. If markers are provided in only the numerator or only the denominator the ratio of all possible combinations of the provided markers and the markers not provided will be calculated. If the number of markers provided are different in the numerator and in the denominator the shorter vector will be repeated to equal the longer vector in length. Data can be unfiltered or filtered since the sum of peak heights per marker is used. Off-ladder alleles is by default removed from the dataset before calculations.

Value

data.frame with columns 'Sample.Name', 'Marker', 'Delta', 'Hb', 'Lb', 'MPH', 'TPH'.

Examples

data(set2)
  # Calculate ratio between the shortest and longest marker in each dye.
  numerator <- c("D3S1358", "AMEL","D19S433")
  denominator <- c("D2S1338", "D18S51", "FGA")
  calculateRatio(data=set2, numerator=numerator, denominator=denominator)
  calculateRatio(data=set2, numerator=NULL, denominator=AMEL)
  calculateRatio(data=set2, numerator=c("AMEL","THO1"), denominator=NULL)
  calculateRatio(data=set2, numerator=NULL, denominator=NULL)
**calculateRatio_gui**  
*Calculate Ratio*

**Description**

GUI wrapper for the `calculateRatio` function.

**Usage**

```r
calculateRatio_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

**Details**

Simplifies the use of the `calculateRatio` function by providing a graphical user interface.

**Value**

`TRUE`

**See Also**

- `link{calculateRatio}`, `link{checkSubset}`

---

**calculateResultType**  
*Calculate Result Type*

**Description**

Calculate the result type for samples.

**Usage**

```r
calculateResultType(data, kit = NULL, add.missing.marker = TRUE, threshold = NULL, mixture.limits = NULL, partial.limits = NULL, subset.name = NA, marker.subset = NULL, debug = FALSE)
```
Arguments

- **data**: a data frame containing at least the column 'Sample.Name'.
- **kit**: character string or integer defining the kit.
- **add.missing.marker**: logical, default is TRUE which adds missing markers.
- **threshold**: integer indicating the dropout threshold.
- **mixture.limits**: integer or vector indicating subtypes of 'Mixture'.
- **partial.limits**: integer or vector indicating subtypes of 'Partial'.
- **subset.name**: string naming the subset of 'Complete'.
- **marker.subset**: string with marker names defining the subset of 'Complete'.
- **debug**: logical indicating printing debug information.

Details

Calculates result types for samples in 'data'. Defined types are: 'No result', 'Mixture', 'Partial', and 'Complete'. Subtypes can be defined by parameters. An integer passed to 'threshold' defines a subtype of 'Complete' "Complete profile all peaks >threshold". An integer or vector passed to 'mixture.limits' define subtypes of 'Mixture' "> [mixture.limits] markers". An integer or vector passed to 'partial.limits' define subtypes of 'Partial' "> [partial.limits] peaks". A string with marker names separated by pipe (|) passed to 'marker.subset' and a string 'subset.name' defines a subtype of 'Partial' "Complete [subset.name]".

Value

data.frame with columns 'Sample.Name', 'Type', and 'Subtype'.

---

**calculateResultType_gui**

*Calculate Result Type*

Description

GUI wrapper for the `calculateResultType` function.

Usage

```r
calculateResultType_gui(env = parent.frame(), savegui = NULL,
                       debug = FALSE, parent = NULL)
```

Arguments

- **env**: environment in which to search for data frames and save result.
- **savegui**: logical indicating if GUI settings should be saved in the environment.
- **debug**: logical indicating printing debug information.
- **parent**: widget to get focus when finished.
**calculateSlope**

**Details**
Simplifies the use of `calculateResultType` by providing a graphical user interface.

**Value**
TRUE

**See Also**
`calculateResultType`

---

**calculateSlope**  
*Calculate Profile Slope.*

**Description**
Calculate profile slope for samples.

**Usage**
calculateSlope(data, ref, conf = 0.975, kit = NULL, debug = FALSE, ...)

**Arguments**
- **data**  
data.frame with at least columns 'Sample.Name', 'Marker', and 'Height'.
- **ref**  
data.frame with at least columns 'Sample.Name', 'Marker', and 'Allele'
- **conf**  
umERIC confidence limit to calculate a confidence interval from (Student t Distribution with 'Peaks'-2 degree of freedom). Default is 0.975 corresponding to a 95% confidence interval.
- **kit**  
character string or vector specifying the analysis kits used to produce the data. If length(kit) != number of groups, kit[1] will be used for all groups.
- **debug**  
logical indicating printing debug information.
- **...**  
additional arguments to the `filterProfile` function

**Details**
Calculates the profile slope for each sample. The slope is calculated as a linear model specified by the response (natural logarithm of peak height) by the term size (in base pair). If 'Size' is not present in the dataset, one or multiple kit names can be given as argument 'kit'. The specified kits will be used to estimate the size of each allele. If 'kit' is NULL the kit(s) will be automatically detected, and the 'Size' will be calculated.

The column 'Group' can be used to separate datasets to be compared, and if so 'kit' must be a vector of equal length as the number of groups, and in the same order. If not the first 'kit' will be recycled for all groups.

Data will be filtered using the reference profiles.
Value

data.frame with with columns 'Sample.Name', 'Kit', 'Group', 'Slope', 'Error', 'Peaks', 'Lower', and 'Upper'.


Description

GUI wrapper for the `calculateSlope` function.

Usage

calculateSlope_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env                    environment in which to search for data frames.
savegui                logical indicating if GUI settings should be saved in the environment.
debug                  logical indicating printing debug information.
parent                  widget to get focus when finished.

Details

Simplifies the use of the `calculateSlope` function by providing a graphical user interface.

Value

TRUE

See Also

`calculateSlope`
**calculateSpike**

**Detect Spike**

**Description**
Detect samples with possible spikes in the DNA profile.

**Usage**
calculateSpike(data, threshold = NULL, tolerance = 2, kit = NULL, quick = FALSE, debug = FALSE)

**Arguments**
- **data**: data.frame with including columns 'Sample.Name', 'Marker', 'Size'.
- **threshold**: numeric number of peaks of similar size in different dye channels to pass as a possible spike (NULL = number of dye channels minus one to allow for one unlabeled peak).
- **tolerance**: numeric tolerance for Size. For the quick and dirty rounding method e.g. 1.5 rounds Size to +/- 0.75 bp. For the slower but more accurate method the value is the maximum allowed difference between peaks in a spike.
- **kit**: string or numeric for the STR-kit used (NULL = auto detect).
- **quick**: logical TRUE for the quick and dirty method. Default is FALSE which use a slower but more accurate method.
- **debug**: logical indicating printing debug information.

**Details**
Creates a list of possible spikes by searching for peaks aligned vertically (i.e. nearly identical size). There are two methods to search. The default method (quick=FALSE) method that calculates the distance between each peak in a sample, and the quick and dirty method (quick=TRUE) that rounds the size and then group peaks with identical size. The rounding method is faster because it uses the data.table package. The accurate method is slower because it uses nested loops - the first through each sample to calculate the distance between all peaks, and the second loops through the distance matrix to identify which peaks lies within the tolerance. NB! The quick method may not catch all spikes since two peaks can be separated by rounding e.g. 200.5 and 200.6 becomes 200 and 201 respectively.

**Value**
data.frame

**See Also**
data.table
**calculateSpike_gui**  
*Detect Spike*

**Description**

GUI wrapper for the `calculateSpike` function.

**Usage**

```r
calculateSpike_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
                    parent = NULL)
```

**Arguments**

- `env`  
  environment in which to search for data frames.
- `savegui`  
  logical indicating if GUI settings should be saved in the environment.
- `debug`  
  logical indicating printing debug information.
- `parent`  
  widget to get focus when finished.

**Details**

Simplifies the use of the `calculateSpike` function by providing a graphical user interface.

**Value**

TRUE

**See Also**

`calculateSpike`

---

**calculateStutter**  
*Calculate Stutter*

**Description**

Calculate statistics for stutters.

**Usage**

```r
calculateStutter(data, ref, back = 2, forward = 1, interference = 0,
                  replace.val = NULL, by.val = NULL, debug = FALSE)
```
Arguments

data data frame with genotype data. Requires columns 'Sample.Name', 'Marker', 'Allele', 'Height'.

ref data frame with the known profiles. Requires columns 'Sample.Name', 'Marker', 'Allele'.

back integer for the maximal number of backward stutters (max size difference 2 = \( n-2 \) repeats).

forward integer for the maximal number of forward stutters (max size difference 1 = \( n+1 \) repeats).

interference integer specifying accepted level of allowed overlap.

replace.val numeric vector with 'false' stutters to replace.

by.val numeric vector with correct stutters.

debug logical indicating printing debug information.

Details

Calculates stutter ratios based on the 'reference' data set and a defined analysis range around the true allele.

NB! Off-ladder alleles ('OL') is NOT included in the analysis. NB! Labeled pull-ups or artefacts within stutter range IS included in the analysis.

There are three levels of allowed overlap (interference). 0 = no interference (default): calculate the ratio for a stutter only if there are no overlap between the stutter or its allele with the analysis range of another allele. 1 = stutter-stutter interference: calculate the ratio for a stutter even if the stutter or its allele overlap with a stutter within the analysis range of another allele. 2 = stutter-allele interference: calculate the ratio for a stutter even if the stutter and its allele overlap with the analysis range of another allele.

Value

data.frame with extracted result.

calculateStutter_gui Calculate Stutter

Description

GUI wrapper for the calculateStutter function.

Usage

calculateStutter_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
**Arguments**

- `env` environment in which to search for data frames and save result.
- `savegui` logical indicating if GUI settings should be saved in the environment.
- `debug` logical indicating printing debug information.
- `parent` widget to get focus when finished.

**Details**

Simplifies the use of the `calculateStutter` function by providing a graphical user interface to it.

**Value**

`TRUE`

**See Also**

`calculateStutter`, `checkSubset`

---

**checkDataset**

**Check Dataset**

**Description**

Internal function to check a data.frame before analysis.

**Usage**

```r
checkDataset(name, reqcol = NULL, slim = FALSE, slimcol = NULL, string = NULL, stringcol = NULL, env = parent.frame(), parent = NULL, debug = FALSE)
```

**Arguments**

- `name` character name of data.frame.
- `reqcol` character vector with required column names.
- `slim` logical TRUE to check if ’slim’ data.
- `slimcol` character vector with column names to check if ’slim’ data.
- `string` character vector with invalid strings in ’stringcol’, return FALSE if found.
- `stringcol` character vector with column names to check for ’string’.
- `env` environment where to look for the data frame.
- `parent` parent gWidget.
- `debug` logical indicating printing debug information.

**Details**

Check that the object exist, there are rows, the required columns exist, if data.frame is ’fat’, and if invalid strings exist. Show error message if not.
checkSubset

Description

Check the result of subsetting

Usage

checkSubset(data, ref, console = TRUE, ignore.case = TRUE, word = FALSE,
exact = FALSE, debug = FALSE)

Arguments

data a data frame in GeneMapper format containing column 'Sample.Name'.
ref a data frame in GeneMapper format containing column 'Sample.Name', OR an
atomic vector e.g. a single sample name string.
console logical, if TRUE result is printed to R console, if FALSE a string is returned.
ignore.case logical, if TRUE case insensitive matching is used.
word logical, if TRUE only word matching (regex).
exact logical, if TRUE only exact match.
debug logical indicating printing debug information.

Details

Check if ref and sample names are unique for subsetting. Prints the result to the R-prompt.

See Also

grep

checkSubset_gui

Description

GUI wrapper for the checkSubset function.

Usage

checkSubset_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
parent = NULL)
Arguments

env environment in which to search for data frames.
savegui logical indicating if GUI settings should be saved in the environment.
debug logical indicating printing debug information.
parent widget to get focus when finished.

Details

Simplifies the use of the `checkSubset` function by providing a graphical user interface to it.

Value

TRUE

See Also

`checkSubset`

---

**colConvert**  
Convert Columns

Description

Internal helper function.

Usage

```r
colConvert(data, columns = "Height\|Size\|Data.Point", ignore.case = TRUE,
fixed = FALSE, debug = FALSE)
```

Arguments

- **data**: data.frame.
- **columns**: character string containing a regular expression (or character string for fixed = TRUE) to be matched in the given character vector (separate multiple column names by | in reg.exp).
- **ignore.case**: logical TRUE to ignore case in matching.
- **fixed**: logical TRUE if columns is a string to be matched as is.
- **debug**: logical indicating printing debug information.

Details

Takes a data frame as input and return it after converting known numeric columns to numeric.

Value

- **data.frame**.
**colNames**  

**Column Names**

**Description**

Internal helper function.

**Usage**

```
colNames(data, slim = TRUE, concatenate = NULL, numbered = TRUE, debug = FALSE)
```

**Arguments**

- `data` : data.frame.
- `slim` : logical, TRUE returns column names occurring once, FALSE returns column names occurring multiple times.
- `concatenate` : string, if not NULL returns a single string with column names concatenated by the provided string instead of a vector.
- `numbered` : logical indicating if repeated column names must have a number suffix.
- `debug` : logical indicating printing debug information.

**Details**

Takes a data frame as input and return either column names occurring once or multiple times. Matching is done by the 'base name' (the substring to the left of the last period, if any). The return type is a string vector by default, or a single string of column names separated by a string 'concatenate' (see 'collapse' in `paste` for details). There is an option to limit multiple names to those with a number suffix.

**Value**

character, vector or string.

---

**columns**  

**Column Actions**

**Description**

Perform actions on columns.

**Usage**

```
columns(data, col1 = NA, col2 = NA, operator = "&", fixed = NA, target = NA, start = 1, stop = 1, debug = FALSE)
```
Arguments

- **data**: a data frame.
- **col1**: character column name to perform action on.
- **col2**: character optional second column name to perform action on.
- **operator**: character to indicate operator: '& ' concatenate, '+' add, '*' multiply, '-' subtract, '/' divide, 'substr' extract a substring.
- **fixed**: character or numeric providing the second operand if 'col2' is not used.
- **target**: character to specify column name for result. Default is to overwrite 'col1'. If not present it will be added.
- **start**: integer, the first position to be extracted.
- **stop**: integer, the last position to be extracted.
- **debug**: logical to indicate if debug information should be printed.

Details

Perform actions on columns in a data frame. There are five actions: concatenate, add, multiply, subtract, divide. The selected action can be performed on two columns, or one column and a fixed value, or a new column can be added. A target column for the result is specified. NB! if the target column already exist it will be overwritten, else it will be created. A common use is to create a unique Sample.Name from the existing Sample.Name column and e.g. the File.Name or File.Time columns. It can also be used to calculate the Amount from the Concentration.

Value

data frame.

See Also

- `substr`

Examples

```r
# Get a sample dataset.
data(set2)
# Add concatenate Sample.Name and Dye.
set2 <- columns(data=set2, col1="Sample.Name", col2="Dye")
# Multiply Height by 4.
set2 <- columns(data=set2, col1="Height", operator="*", fixed=4)
# Add a new column.
set2 <- columns(data=set2, operator="&", fixed="1234", target="Batch")
```
### columns_gui

**Column Actions**

**Description**

GUI wrapper for the `columns` function.

**Usage**

```r
columns_gui(env = parent.frame(), savegui = NULL, debug = FALSE, 
parent = NULL)
```

**Arguments**

- `env`: environment in which to search for data frames.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

**Details**

Simplifies the use of the `columns` function by providing a graphical user interface to it.

**Value**

`TRUE`

### combineBinsAndPanels

**Combine Bins And Panels Files.**

**Description**

Combines useful information into one dataset.

**Usage**

```r
combineBinsAndPanels(bin, panel)
```

**Arguments**

- `bin`: data frame created from the 'bins' file.
- `panel`: data frame created from the 'panels' file.
Details

Combines information from two sources ('Bins' and 'Panels' file) to create a dataset containing information about panel name, marker name, alleles in the allelic ladder, their size and size range, a flag indicating virtual alleles, fluorophore color, repeat size, marker range. The short name, full name, and sex marker flag is populated through the makeKit_gui user interface. In addition the function calculates an estimated offset for each marker, which can be used for creating EPG like plots. Note: offset is estimated by taking the smallest physical ladder fragment e.g. 98.28 for D3 in ESX17. Round this to an integer (98) and finally subtract the number of base pair for that repeat i.e. 4*9=36, which gives an offset of 98-36 = 62 bp. Microvariants are handled by taking the decimal part multiplied with 10 and adding this to the number of base pair e.g. 9.3 = 4*9 + 0.3*10 = 39 bp.

Value


Description

GUI for combining two datasets.

Usage

combine_gui(env = parent.frame(), debug = FALSE, parent = NULL)

Arguments

env environment in which to search for data frames.
debug logical indicating printing debug information.
parent widget to get focus when finished.

Details

Simple GUI to combine two datasets using the rbind.fill function. NB! Datasets must have identical column names but not necessarily in the same order.

Value

TRUE
cropData_gui  

**Crop Or Replace**  

**Description**  
GUI simplifying cropping and replacing values in data frames.

**Usage**  
cropData_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

**Arguments**  
- **env**: environment in which to search for data frames.  
- **savegui**: logical indicating if GUI settings should be saved in the environment.  
- **debug**: logical indicating printing debug information.  
- **parent**: widget to get focus when finished.

**Details**  
Select a data frame from the drop-down and a target column. To remove rows with 'NA' check the appropriate box. Select to discard or replace values and additional options. Click button to 'Apply' changes. Multiple actions can be performed on one dataset before saving as a new dataframe. NB! Check that data type is correct before click apply to avoid strange behavior. If data type is numeric any string will become a numeric 'NA'.

**Value**  
TRUE

**See Also**  
trim_gui, editData_gui, combine_gui

detectKit  

**Detect Kit**  

**Description**  
Finds the most likely STR kit for a dataset.

**Usage**  
detectKit(data, index = FALSE, debug = FALSE)
Arguments

data  data frame with column ‘Marker’ or vector with marker names.
index  logical, returns kit index if TRUE or short name if FALSE.
debug  logical, prints debug information if TRUE.

Details

The function first check if there is a ‘kit’ attribute for the dataset. If there was a ‘kit’ attribute, and a match is found in getKit the corresponding kit or index is returned. If an attribute does not exist the function looks at the markers in the dataset and returns the most likely kit(s).

Value

integer or string indicating the detected kit.

editData_gui  Edit or View Data Frames

Description

GUI to edit and view data frames.

Usage

editData_gui(env = parent.frame(), savegui = NULL, data = NULL,
               name = NULL, edit = TRUE, debug = FALSE, parent = NULL)

Arguments

env  environment in which to search for data frames.
savegui  logical indicating if GUI settings should be saved in the environment.
data  data.frame for instant viewing.
name  character string with the name of the provided dataset.
edit  logical TRUE to enable edit (uses gdf), FALSE to view and enable sorting by clicking a column header (uses gtable).
debug  logical indicating printing debug information.
parent  widget to get focus when finished.

Details

Select a data frame from the drop-down to view or edit a dataset. It is possible to save as a new dataframe. To enable sorting by clicking the column headers the view mode must be used (i.e. edit = FALSE). There is an option to limit the number of rows shown that can be used to preview large datasets that may otherwise cause performance problems. Attributes of the dataset can be views in a separate window.
export

Value
TRUE

See Also
trim_gui, cropData_gui, combine_gui

export  Export

Description
Exports or saves various objects.

Usage
export(object, name = NA, use.object.name = is.na(name),
env = parent.frame(), path = NA, ext = "auto", delim = "\t",
width = 3000, height = 2000, res = 250, overwrite = FALSE,
debug = FALSE)

Arguments
object   string, list or vector containing object names to be exported.
name     string, list or vector containing file names. Multiple names as string must be
          separated by pipe '|'. If not equal number of names as objects, first name will be
          used to construct names.
use.object.name  logical, if TRUE file name will be the same as object name.
env       environment where the objects exists.
path      string specifying the destination folder exported objects.
ext       string specifying file extension. Default is 'auto' for automatic .txt or .png based
          on object class. If .RData all objects will be exported as .RData files.
delim     string specifying the delimiter used as separator.
width     integer specifying the width of the image.
height    integer specifying the height of the image.
res       integer specifying the resolution of the image.
overwrite logical, TRUE if existing files should be overwritten.
debug     logical indicating printing debug information.

Details
Export objects to a directory on the file system. Currently only objects of class data.frames or
ggplot are supported. data.frame objects will be exported as '.txt' and ggplot objects as '.png'.
.RData applies to all supported object types.
Value

NA if all objects were exported OR, data.frame with columns 'Object', 'Name', and 'New.Name' with objects that were not exported.

Description

GUI wrapper for the export function.

Usage

```r
export_gui(obj = listObjects(env = env, obj.class = c("data.frame", "ggplot")), env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

Arguments

- `obj`: character vector with object names.
- `env`: environment where the objects exist. Default is the current environment.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

Details

Simplifies the use of the `export` function by providing a graphical user interface to it. Currently all available objects provided are selected by default.

Value

TRUE

See Also

`export`
filterProfile

Description

Filter peaks from profiles.

Usage

filterProfile(data, ref = NULL, add.missing.loci = FALSE, keep.na = FALSE,
ignore.case = TRUE, exact = FALSE, word = FALSE, invert = FALSE,
sex.rm = FALSE, qs.rm = FALSE, kit = NULL, filter.allele = TRUE,
debug = FALSE)

Arguments

data data frame with genotype data in 'slim' format.
ref data frame with reference profile in 'slim' format.
add.missing.loci logical. TRUE add loci present in ref but not in data. Overrides keep.na=FALSE.
keep.na logical. FALSE discards NA alleles. TRUE keep loci/sample even if no matching allele.
ignore.case logical TRUE ignore case.
exact logical TRUE use exact matching of sample names.
word logical TRUE adds word boundaries when matching sample names.
invert logical TRUE filter peaks NOT matching the reference.
sex.rm logical TRUE removes sex markers defined by 'kit'.
qs.rm logical TRUE removes quality sensors defined by 'kit'.
kit character string defining the kit used. If NULL automatic detection will be attempted.
filter.allele logical TRUE filter known alleles. FALSE increase the performance if only sex markers or quality sensors should be removed.
debug logical indicating printing debug information.

Details

Filters out the peaks matching (or not matching) specified known profiles from typing data containing 'noise' such as stutters. If 'ref' does not contain a 'Sample.Name' column it will be used as reference for all samples in 'data'. The 'invert' option filters out peaks NOT matching the reference (e.g. drop-in peaks). Sex markers and quality sensors can be removed. NB! add.missing.loci overrides keep.na. Returns data where allele names match/not match 'ref' allele names. Required columns are: 'Sample.Name', 'Marker', and 'Allele'.

Value

data.frame with extracted result.
**filterProfile_gui**  
*Filter Profile*

**Description**

GUI wrapper for the `filterProfile` function.

**Usage**

```r
filterProfile_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- **env**: environment in which to search for data frames.
- **savegui**: logical indicating if GUI settings should be saved in the environment.
- **debug**: logical indicating printing debug information.
- **parent**: widget to get focus when finished.

**Details**

Simplifies the use of the `filterProfile` function by providing a graphical user interface to it. All data not matching/matching the reference will be discarded. Useful for filtering stutters and artifacts from raw typing data or to identify drop-ins.

**Value**

`TRUE`

**See Also**

- `filterProfile`
- `checkSubset`

---

**generateEPG**  
*Generate EPG*

**Description**

Visualizes an EPG from DNA profiling data.

**Usage**

```r
generateEPG(data, kit, title = NULL, wrap = TRUE, boxplot = FALSE, peaks = TRUE, collapse = TRUE, silent = FALSE, ignore.case = TRUE, at = 0, scale = "free", limit.x = TRUE, label.size = 3, label.angle = 0, label.vjust = 1, label.hjust = 0.5, expand = 0.1, debug = FALSE)
```
Arguments

data  data frame containing at least columns 'Sample.Name', 'Allele', and 'Marker'.
kit  string or integer representing the STR typing kit.
title  string providing the title for the EPG.
wrap  logical TRUE to wrap by dye.
boxplot  logical TRUE to plot distributions of peak heights as boxplots.
peaks  logical TRUE to plot peaks for distributions using mean peak height.
collapse  logical TRUE to add the peak heights of identical alleles peaks within each marker. NB! Removes off-ladder alleles.
silent  logical FALSE to show plot.
ignore.case  logical FALSE for case sensitive marker names.
at  numeric analytical threshold (Height <= at will not be plotted).
scale  character "free" free x and y scale, alternatively "free_y" or "free_x".
limit.x  logical TRUE to fix x-axis to size range. To get a common x scale set scale="free_y" and limit.x=TRUE.
label.size  numeric for allele label text size.
label.angle  numeric for allele label print angle.
label.vjust  numeric for vertical justification of allele labels.
label.hjust  numeric for horizontal justification of allele labels.
expand  numeric for plot are expansion (to avoid clipping of labels).
debug  logical for printing debug information to the console.

Details

Generates a electropherogram like plot from 'data' and 'kit'. If 'Size' is not present it is estimated from kit information and allele values. If 'Height' is not present a default of 1000 RFU is used. Off-ladder alleles can be plotted if 'Size' is provided. There are various options to customize the plot scale and labels. It is also possible to plot 'distributions' of peak heights as boxplots.

Value

ggplot object.
generateEPG_gui  Generate EPG

Description
GUI wrapper for the generateEPG function.

Usage
generateEPG_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
parent = NULL)

Arguments
denv environment in which to search for data frames and save result.
savegui logical indicating if GUI settings should be saved in the environment.
ddebug logical indicating printing debug information.
dparent widget to get focus when finished.

Details
Simplifies the use of the generateEPG function by providing a graphical user interface to it.

Value
TRUE

See Also
generateEPG

getKit  Get Kit

Description
Provides information about STR kits.

Usage
getKit(kit = NULL, what = NA, show.messages = FALSE, .kit.info = NULL,
ddebug = FALSE)
Arguments

kit  string or integer to specify the kit.
what  string to specify which information to return. Default is 'NA' which return all info. Not case sensitive.
show.messages  logical, default TRUE for printing messages to the R prompt.
.kit.info  data frame, run function on a data frame instead of the kits.txt file.
debug  logical indicating printing debug information.

Details

The function returns the following information for a kit specified in kits.txt: Panel name, short kit name (unique, user defined), full kit name (user defined), marker names, allele names, allele sizes (bp), minimum allele size, maximum allele size (bp), flag for virtual alleles, marker color, marker repeat unit size (bp), minimum marker size, maximum marker, marker offset (bp), flag for sex markers (TRUE/FALSE).

If no matching kit or kit index is found NA is returned. If kit='NULL' or '0' a vector of available kits is printed and NA returned.

Value

data.frame with kit information.

Examples

# Show all information stored for kit with short name 'ESX17'.
getKit("ESX17")

ggsave_gui  Save Image

Description

A simple GUI wrapper for ggsave.

Usage

ggsave_gui(ggplot = NULL, name = "", env = parent.frame(),
savegui = NULL, debug = FALSE, parent = NULL)
guessProfile

Arguments

ggplot plot object.
name optional string providing a file name.
env environment where the objects exist. Default is the current environment.
savegui logical indicating if GUI settings should be saved in the environment.
debug logical indicating printing debug information.
parent object specifying the parent widget to center the message box, and to get focus when finished.

Details

Simple GUI wrapper for ggsave.

Value

TRUE

See Also

ggsave

guessProfile HdataL ratio = PN6L height = UPL naNrm = falseL olNrm = trueL debug = falseI

Arguments

data a data frame containing at least 'Sample.Name', 'Marker', 'Allele', Height'.
ratio numeric giving the peak height ratio threshold.
height numeric giving the minimum peak height.
na.rm logical indicating if rows with no peak should be discarded.
ol.rm logical indicating if off-ladder alleles should be discarded.
debug logical indicating printing debug information.

Description

Guesses the correct profile based on peak height.

Usage

guessProfile(data, ratio = 0.6, height = 50, na.rm = FALSE, ol.rm = TRUE, debug = FALSE)

Arguments

data a data frame containing at least 'Sample.Name', 'Marker', 'Allele', Height'.
ratio numeric giving the peak height ratio threshold.
height numeric giving the minimum peak height.
na.rm logical indicating if rows with no peak should be discarded.
ol.rm logical indicating if off-ladder alleles should be discarded.
debug logical indicating printing debug information.
Details
Takes typing data from single source samples and filters out the presumed profile based on peak height and a ratio. Keeps the two highest peaks if their ratio is above the threshold, or the single highest peak if below the threshold.

Value
data.frame 'data' with genotype rows only.

Examples

```r
# Load an example dataset.
data(set2)
# Filter out probable profile with criteria at least 70% Hb.
guessProfileGui(data=set2, ratio=0.7)
```

Description
GUI wrapper for the `guessProfile` function.

Usage

```r
guessProfileGui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

Arguments

- `env` environment in which to search for data frames.
- `savegui` logical indicating if GUI settings should be saved in the environment.
- `debug` logical indicating printing debug information.
- `parent` widget to get focus when finished.

Details
Simplifies the use of the `guessProfile` function by providing a graphical user interface to it.

Value
TRUE

See Also
`guessProfile`, `checkSubset`
**heightToPeak**  
*Height To Peak.*

**Description**

Helper function to convert a peak into a plotable polygon.

**Usage**

```r
eightToPeak(data, width = 1L, keep.na = TRUE, debug = FALSE)
```

**Arguments**

- `data`: data frame containing at least columns 'Height' and 'Size'.
- `width`: numeric specifying the width of the peak in bp.
- `keep.na`: logical. TRUE to keep empty markers.
- `debug`: logical. TRUE prints debug information.

**Details**

Converts a single height and size value to a plotable 0-height-0 triangle/peak value. Makes 3 data points from each peak size for plotting a polygon representing a peak. Factors in other columns might get converted to factor level.

**Value**

data.frame with new values.

---

**import**  
*Import Data*

**Description**

Import text files and apply post processing.

**Usage**

```r
import(folder = TRUE, extension = "txt", suffix = NA, prefix = NA,
       import.file = NA, folder.name = NA, file.name = TRUE,
       time.stamp = TRUE, separator = "\t", ignore.case = TRUE,
       auto.trim = FALSE, trim.samples = NULL, trim.invert = FALSE,
       auto.slim = FALSE, slim.na = TRUE, na.strings = c("NA", ""),
       debug = FALSE)
```
import

Arguments

folder logical, TRUE all files in folder will be imported, FALSE only selected file will be imported.
extension string providing the file extension.
suffix string, only files with specified suffix will be imported.
prefix string, only files with specified prefix will be imported.
import.file string if file name is provided file will be imported without showing the file open dialogue.
folder.name string if folder name is provided files in folder will be imported without showing the select folder dialogue.
file.name logical if TRUE the file name is written in a column 'File.Name'. NB! Any existing 'File.Name' column is overwritten.
time.stamp logical if TRUE the file modified time stamp is written in a column 'Time'. NB! Any existing 'Time' column is overwritten.
separator character for the delimiter used to separate columns (see 'sep' in read.table for details).
ignore.case logical indicating if case should be ignored. Only applies to multiple file import option.
auto.trim logical indicating if dataset should be trimmed.
trim.samples character vector with sample names to trim.
trim.invert logical to keep (TRUE) or remove (FALSE) samples.
auto.slim logical indicating if dataset should be slimmed.
slim.na logical indicating if rows without data should remain.
na.strings character vector with strings to be replaced by NA.
debug logical indicating printing debug information.

Details

Imports text files (e.g. GeneMapper results exported as text files) as data frames. Options to import one or multiple files. For multiple files it is possible to specify prefix, suffix, and file extension to create a file name filter. The file name and/or file time stamp can be imported. NB! Empty strings ("") and NA strings ("NA") are converted to NA. See list.files and read.table for additional details.

Value
data.frame with imported result.

See Also

trim, slim, list.files, read.table
import_gui  Import Data

Description

GUI wrapper for the import function.

Usage

import_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env  environment into which the object will be saved. Default is the current environment.
savegui  logical indicating if GUI settings should be saved in the environment.
debug  logical indicating printing debug information.
parent  widget to get focus when finished.

Details

Simplifies the use of the import function by providing a graphical user interface to it.

Value

TRUE

See Also

import

listObjects  List Objects

Description

Internal helper function to list objects in an environment.

Usage

listObjects(env = parent.frame(), obj.class = NULL, debug = FALSE)
Arguments

env  environment in which to search for objects.
obj.class character string or vector specifying the object class.
debug logical indicating printing debug information.

Details

Internal helper function to retrieve a list of objects from a workspace. Take an environment as argument and optionally an object class. Returns a list of objects of the specified class in the environment.

Value

character vector with the object names.

Examples

## Not run:
# List data frames in the workspace.
listObjects(obj.class="data.frame")
# List functions in the workspace.
listObjects(obj.class="function")

## End(Not run)

makeKit_gui

Make Kit

Description

Add new kits or edit the kit file.

Usage

makeKit_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
parent = NULL)

Arguments

env  environment in which to search for data frames.
savegui logical indicating if GUI settings should be saved in the environment. [Not currently used]
debug logical indicating printing debug information.
parent widget to get focus when finished.
Details

A graphical user interface for reading information from 'bins' and 'panels' file for the creation of additional kits. It is also possible to edit the short and full name of existing kits or removing kits. The gender marker of each kits is auto detected but can be changed manually. NB! Short name must be unique.

Value

TRUE

See Also

readBinsFile, readPanelsFile, combineBinsAndPanels

maskAT

Mask And Prepare Data To Analyze Analytical Threshold

Description

Break-out function to prepare data for the function calculateAT.

Usage

```r
maskAT(data, ref = NULL, mask.height = TRUE, height = 500,
mask.sample = TRUE, per.dye = TRUE, range.sample = 20,
mask.ils = TRUE, range.ils = 10, ignore.case = TRUE, word = FALSE,
debug = FALSE)
```

Arguments

data: a data frame containing at least 'Dye.Sample.Peak', 'Sample.File.Name', 'Marker', 'Allele', 'Height', and 'Data.Point'.

ref: a data frame containing at least 'Sample.Name', 'Marker', 'Allele'.

mask.height: logical to indicate if high peaks should be masked.

height: integer for global lower peak height threshold for peaks to be excluded from the analysis. Active if 'mask.peak=TRUE.

mask.sample: logical to indicate if sample allelic peaks should be masked.

per.dye: logical TRUE if sample peaks should be masked per dye channel. FALSE if sample peaks should be masked globally across dye channels.

range.sample: integer to specify the masking range in (+/-) data points. Active if mask.sample=TRUE.

mask.ils: logical to indicate if internal lane standard peaks should be masked.

range.ils: integer to specify the masking range in (+/-) data points. Active if mask.ils=TRUE.

ignore.case: logical to indicate if sample matching should ignore case.

word: logical to indicate if word boundaries should be added before sample matching.

debug: logical to indicate if debug information should be printed.
Details

Prepares the 'SamplePlotSizingTable' for analysis of analytical threshold. It is needed by the plot functions for control of masking. The preparation consist of converting the 'Height' and 'Data.Point' column to numeric (if needed), then dye channel information is extracted from the 'Dye.Sample.Peak' column and added to its own 'Dye' column, known fragments of the internal lane standard (marked with an asterisk '*') is flagged as 'TRUE' in a new column 'ILS'.

Value

data.frame with added columns 'Dye' and 'ILS'.

See Also

calculateAT

Description

Model the probability of drop-out and plot graphs.

Usage

modeldropout_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env environment in which to search for data frames and save result.
savegui logical indicating if GUI settings should be saved in the environment.
debug logical indicating printing debug information.
parent widget to get focus when finished.

Details

calculateDropout score drop-out events relative to a user defined LDT in four different ways: (1) by reference to the low molecular weight allele (Method1), (2) by reference to the high molecular weight allele (Method2), (3) by reference to a random allele (MethodX), and (4) by reference to the locus (MethodL). Options 1-3 are recommended by the DNA commission (see reference), while option 4 is included for experimental purposes. Options 1-3 may discard many dropout events while option 4 catches all drop-out events. On the other hand options 1-3 can score events below the LDT, while option 4 cannot, making accurate predictions possible below the LDT.

Method X/1/2 records the peak height of the partner allele to be used as the explanatory variable in the logistic regression. The locus method L also do this when there has been a drop-out, if not
the the mean peak height for the locus is used. Peak heights for the locus method are stored in a separate column.

Using the scored drop-out events and the peak heights of the surviving alleles the probability of drop-out can be modeled by logistic regression as described in Appendix B in reference [1].

\[
P(\text{dropout}|H) = B_0 + B_1 \times \log(H)\]

This produces a plot with the predicted probabilities for a range of peak heights. There are options to print the model parameters, mark the stochastic threshold at a specified probability of drop-out, include the underlying observations, and to calculate a specified prediction interval. A conservative estimate of the stochastic threshold can be calculated from the prediction interval: the risk of observing a drop-out probability greater than the specified threshold limit, at the conservative peak height, is less than a specified value (e.g. 1-0.95=0.05). By default the gender marker is excluded from the dataset used for modeling, and the peak height is used as explanatory variable. The logarithm of the average peak height 'H' can be used instead of the allele/locus peak height [3]. To evaluate the goodness of fit for the logistic regression the Hosmer-Lemeshow test is used [4]. A value below 0.05 indicates a poor fit. Alternatives to the logistic regression method are discussed in reference [5] and [6].

Explanation of the result: Dropout - all alleles are scored according to the limit of detection threshold (LDT). This is the observations and is not used for modeling. Rfu - peak height of the surviving allele. MethodX - a random reference allele is selected and drop-out is scored in relation to the partner allele. Method1 - the low molecular weight allele is selected and drop-out is scored if the high molecular weight allele is missing. Method2 - the high molecular weight allele is selected and drop-out is scored if the low molecular weight allele is missing. MethodL - drop-out is scored per locus i.e. drop-out if any allele is missing. MethodL_P - peak height of the surviving allele if one allele has dropped out, or the average peak height if no drop-out.

Value
TRUE

References


plotAT_gui


See Also

calculateDropout, plotDropout_gui

plotAT_gui

Plot Analytical Threshold

Description

GUI simplifying the creation of plots from analytical threshold data.

Usage

plotAT_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env environment in which to search for data frames.
savegui logical indicating if GUI settings should be saved in the environment.
debug logical indicating printing debug information.
parent widget to get focus when finished.

Details

Select data to plot in the drop-down menu. Plot regression data Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE

See Also

http://docs.ggplot2.org/current/ for details on plot settings.
plotBalance_gui

Description

GUI simplifying the creation of plots from balance data.

Usage

plotBalance_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env             environment in which to search for data frames and save result.
savegui         logical indicating if GUI settings should be saved in the environment.
debug           logical indicating printing debug information.
parent          widget to get focus when finished.

Details

Select a dataset to plot and the typing kit used (if not automatically detected). Plot heterozygote peak balance versus the average locus peak height, the average profile peak height 'H', or by the difference in repeat units (delta). Plot inter-locus balance versus the average locus peak height, or the average profile peak height 'H'. Automatic plot titles can be replaced by custom titles. Sex markers can be excluded. It is possible to plot logarithmic ratios. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE

See Also

http://docs.ggplot2.org/current/ for details on plot settings.
**plotCapillary_gui**  
*Plot Capillary Balance*

**Description**
GUI simplifying the creation of plots from capillary balance data.

**Usage**

```r
plotCapillary_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**
- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

**Details**
Select a dataset to plot from the drop-down menu. Plot capillary balance as a dotplot, boxplot or as a distribution. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

**Value**
TRUE

**See Also**


---

**plotContamination_gui**  
*Plot Contamination*

**Description**
GUI simplifying the creation of plots from negative control data.

**Usage**

```r
plotContamination_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```
Arguments

env  environment in which to search for data frames.
savegui  logical indicating if GUI settings should be saved in the environment.
debug  logical indicating printing debug information.
parent  widget to get focus when finished.

Details

Select data to plot in the drop-down menu. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE

References


plotDistribution_gui  Plot Distribution

Description

GUI simplifying the creation of distribution plots.

Usage

plotDistribution_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env  environment in which to search for data frames and save result.
savegui  logical indicating if GUI settings should be saved in the environment.
debug  logical indicating printing debug information.
parent  widget to get focus when finished.
**plotDropout_gui**

**Details**

Plot the distribution of data as cumulative distribution function, probability density function, or count. First select a dataset, then select a group (in column ‘Group’ if any), finally select a column to plot the distribution of. It is possible to overlay a boxplot and to plot logarithms. Various smoothing kernels and bandwidths can be specified. The bandwidth or the number of bins can be specified for the histogram. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

**Value**

TRUE

**See Also**

log, geom_density

---

**plotDropout_gui**  
*Plot Drop-out Events*

**Description**

GUI simplifying the creation of plots from dropout data.

**Usage**

plotDropout_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

**Arguments**

- **env**  
  environment in which to search for data frames and save result.
- **savegui**  
  logical indicating if GUI settings should be saved in the environment.
- **debug**  
  logical indicating printing debug information.
- **parent**  
  widget to get focus when finished.

**Details**

Plot dropout data as heatmap arranged by, average peak height, amount, concentration, or sample name. It is also possible to plot the empirical cumulative distribution (ecdp) of the peak heights of surviving heterozygote alleles (with dropout of the parter allele), or a dotplot of all dropout events. The peak height of homozygote alleles can be included in the ecdp. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

**Value**

TRUE
plotKit_gui  

Plot Kit Marker Ranges

Description

GUI for plotting marker ranges for kits.

Usage

plotKit_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env  
environment in which to search for data frames and save result.

savegui  
logical indicating if GUI settings should be saved in the environment.

default  
logical indicating printing debug information.

parent  
widget to get focus when finished.

Details

Create an overview of the size range for markers in different kits. It is possible to select multiple kits, specify titles, font size, distance between two kits, distance between dye channels, and the transparency of dyes.

Value

TRUE

plotPeaks_gui  

Plot Peaks

Description

GUI simplifying the creation of plots from result type data.

Usage

plotPeaks_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
Arguments

env         environment in which to search for data frames and save result.
savegui    logical indicating if GUI settings should be saved in the environment.
debug      logical indicating printing debug information.
parent     widget to get focus when finished.

Details

Plot result type data. It is possible to customize titles and font size. Data can be plotted as as frequency or proportion. The values can be printed on the plot with custom number of decimals. There are several color palettes to chose from. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE

plotPrecision_gui   Plot Precision

Description

GUI simplifying the creation of plots from precision data.

Usage

plotPrecision_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
                  parent = NULL)

Arguments

env         environment in which to search for data frames.
savegui    logical indicating if GUI settings should be saved in the environment.
debug      logical indicating printing debug information.
parent     widget to get focus when finished.

Details

Plot precision data for size, height, or data point as dotplot or boxplot. Plot per marker or all in one. Use the mean value or the allele designation as x-axis labels. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE
plotPullup_gui

See Also

http://docs.ggplot2.org/current/ for details on plot settings.

plotPullup_gui | Plot Pull-up

Description

GUI simplifying the creation of plots from pull-up data.

Usage

plotPullup Gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env | environment in which to search for data frames and save result.
savegui | logical indicating if GUI settings should be saved in the environment.
debug | logical indicating printing debug information.
parent | widget to get focus when finished.

Details

Select a dataset to plot and the typing kit used (if not automatically detected). Plot pull-up peak ratio versus the peak height of the known allele. Automatic plot titles can be replaced by custom titles. Sex markers can be excluded. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or an image.

Value

TRUE

See Also

http://docs.ggplot2.org/current/ for details on plot settings.
plotRatio_gui

Description
GUI simplifying the creation of plots from marker ratio data.

Usage
plotRatio_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments
env environment in which to search for data frames.
savegui logical indicating if GUI settings should be saved in the environment.
debug logical indicating printing debug information.
parent widget to get focus when finished.

Details
Select data to plot in the drop-down menu. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value
TRUE

See Also
http://docs.ggplot2.org/current/ for details on plot settings.

plotResultType_gui

Description
GUI simplifying the creation of plots from result type data.

Usage
plotResultType_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
plotSlope_gui

Arguments

env: environment in which to search for data frames and save result.
savegui: logical indicating if GUI settings should be saved in the environment.
debug: logical indicating printing debug information.
parent: widget to get focus when finished.

Details

Plot result type data. It is possible to customize titles and font size. Data can be plotted as a frequency or proportion. The values can be printed on the plot with custom number of decimals. There are several color palettes to choose from. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE

Description

GUI simplifying the creation of plots from slope data.

Usage

plotSlope_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env: environment in which to search for data frames and save result.
savegui: logical indicating if GUI settings should be saved in the environment.
debug: logical indicating printing debug information.
parent: widget to get focus when finished.

Details

Select a dataset to plot. Plot slope by sample. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE
plotStutter_gui

See Also

http://docs.ggplot2.org/current/ for details on plot settings.

---

plotStutter_gui  

Plot Stutter

Description

GUI simplifying the creation of plots from stutter data.

Usage

plotStutter_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments

env  

environment in which to search for data frames.

savegui  

logical indicating if GUI settings should be saved in the environment.

debug  

logical indicating printing debug information.

parent  

widget to get focus when finished.

Details

Select data to plot in the drop-down menu. Check that the correct kit has been detected. Plot stutter data by parent allele or by peak height. Automatic plot titles can be replaced by custom titles. A name for the result is automatically suggested. The resulting plot can be saved as either a plot object or as an image.

Value

TRUE

See Also

http://docs.ggplot2.org/current/ for details on plot settings.
ESX17 Positive Control Profile

Description
A dataset in 'GeneMapper' format containing the DNA profile of the ESX17 positive control sample with homozygotes as one entry.

Usage
data(ref1)

Format
A data frame with 17 rows and 4 variables

ESX17 Positive Control Profile

Description
A dataset in 'GeneMapper' format containing the DNA profile of the ESX17 positive control sample with homozygotes as two entries.

Usage
data(ref11)

Format
A data frame with 17 rows and 4 variables

SGMPlus example data

Description
A slimmed reference dataset containing an arbitrary SGMPlus DNA profile.

Usage
data(ref2)

Format
A data frame with 16 rows and 3 variables
ESX17 example data for dropout analysis.

Description
Reference profiles for source samples. Text file in GeneMapper format.

Format
ASCII text file

ESX17 example data for dropout analysis.

Description
A slimmed dataset containing reference profiles for source samples in set4. Reference 'A2' has double entries for homozygotes. Reference 'F2' has single entries for homozygotes. Reference 'bc' has double entries for homozygotes, and lower case sample name.

Usage
data(ref4)

Format
A data frame with 98 rows and 3 variables

ESX17 example data for mixture analysis.

Description
A slimmed dataset containing the reference profile for the major component in set5.

Usage
data(ref51)

Format
A data frame with 34 rows and 3 variables
ESX17 example data for mixture analysis.

Description
A slimmed dataset containing the reference profile for the minor component in set5.

Usage
data(ref52)

Format
A data frame with 34 rows and 3 variables

Fusion example data for dropout analysis.

Description
A slimmed dataset containing the reference profile for the samples in set6. NB! Marker order is different from set6. NB! Reference R has a Y marker with NA.

Usage
data(ref61)

Format
A data frame with 89 rows and 3 variables

Fusion example data for dropout analysis.

Description
A slimmed dataset containing the reference profile for the samples in set6. NB! Marker order is same as set6. NB! Reference R has a Y marker with NA.

Usage
data(ref62)

Format
A data frame with 89 rows and 3 variables
**Description**

A slimmed dataset containing the reference profile for the samples in set7.

**Usage**

```r
data(ref7)
```

**Format**

A data frame with 35 rows and 4 variables

---

**removeArtefact**  
*Remove Artefacts*

**Description**

Remove artefact peaks from data.

**Usage**

```r
removeArtefact(data, artefact = NULL, marker = NULL, allele = NULL, threshold = NULL, na.rm = FALSE, debug = FALSE)
```

**Arguments**

- `data`  
data.frame with data to remove spikes from.
- `artefact`  
data.frame that lists artefacts in columns 'Marker', 'Allele', optionally with 'Allele.Proportion'. Alternatively artefacts can be provided using 'marker' and 'allele'.
- `marker`  
character vector with marker names paired with values in 'allele'.
- `allele`  
character vector with allele names paired with values in 'marker'.
- `threshold`  
numeric value defining a minimum proportion for artefacts. Requires 'artefacts' including the column 'Allele.Proportion'.
- `na.rm`  
logical TRUE to preserve Allele=NA in 'data'.
- `debug`  
logical indicating printing debug information.

**Details**

Removes identified artefacts from the dataset. Likely artefacts can be identified using the function `calculateAllele`. The output should then be provided to the 'artefact'. Alternatively known artefacts can be provided using the 'marker' and 'allele' arguments.
removeSpike

Value
data.frame with spikes removed.

removeArtefact_gui  Remove Artefact

Description
GUI wrapper for the removeArtefact function.

Usage
removeArtefact_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)

Arguments
- env: environment in which to search for data frames.
- savegui: logical indicating if GUI settings should be saved in the environment.
- debug: logical indicating printing debug information.
- parent: widget to get focus when finished.

Details
Simplifies the use of the removeArtefact function by providing a graphical user interface to it.

Value
TRUE

removeSpike  Remove Spikes

Description
Remove spikes from data.

Usage
removeSpike(data, spike, invert = FALSE, debug = FALSE)
Arguments

- **data**: data.frame with data to remove spikes from.
- **spike**: data.frame with list of spikes.
- **invert**: logical, FALSE to remove spikes, TRUE to keep spikes.
- **debug**: logical indicating printing debug information.

Details

Removes identified spikes from the dataset. Spikes are identified using the function `calculateSpike` and provided as a separate dataset. NB! Samples must have unique identifiers. Some laboratories use non-unique names for e.g. negative controls. To allow identification of specific samples when multiple batches are imported into one dataset an id is automatically created by combining the sample name and the file name. This works well as long as there is at most 1 identically named sample in each file (batch). To enable multiple identically named samples in one file, the sample names can be prefixed with the lane or well number before importing them to STR-validator.

Value

data.frame with spikes removed.

---

### removeSpike_gui

**Description**

GUI wrapper for the `removeSpike` function.

**Usage**

```r
removeSpike_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

**Arguments**

- **env**: environment in which to search for data frames.
- **savegui**: logical indicating if GUI settings should be saved in the environment.
- **debug**: logical indicating printing debug information.
- **parent**: widget to get focus when finished.

**Details**

Simplifies the use of the `removeSpike` function by providing a graphical user interface to it.

**Value**

TRUE
scrambleAlleles  *Scramble Alleles*

**Description**

Scrambles alleles in a dataset to anonymize the profile.

**Usage**

```r
scrambleAlleles(data, db = "ESX 17 Hill")
```

**Arguments**

- `data`: data.frame with columns 'Sample.Name', 'Marker', and 'Allele'.
- `db`: character defining the allele frequency database to be used.

**Details**

Internal helper function to create example data. Assumes data with unique alleles per marker i.e. no duplications. This allow for sampling without replacement see `sample`. Sex markers are currently not scrambled i.e. they are kept intact. Alleles in the dataset is replaced with random alleles sampled from the allele database. If 'Size' is in the dataset it will be replaced by an estimated size. If 'Data.Point' is present it will be removed.

**Value**

data.frame with changes in 'Allele' column.

---

**set1  *Typing data in 'GeneMapper' format***

**Description**

A dataset containing ESX17 genotyping result for 8 replicates of the positive control sample, a negative control and ladder.

**Usage**

```r
data(set1)
```

**Format**

A data frame with 170 rows and 13 variables
**set2  SGMPPlus example data**

**Description**
A slimmed dataset containing SGM Plus genotyping result for 2 replicates of 'sampleA'.

**Usage**
```r
data(set2)
```

**Format**
A data frame with 32 rows and 5 variables

---

**set3  ESX17 example data for dropout analysis.**

**Description**
Data from dilution experiment for dropout analysis. Text file with exported GeneMapper genotypes table.

**Format**
ASCII text file

---

**set4  ESX17 example data for dropout analysis.**

**Description**
A slimmed dataset containing data from dilution experiment for dropout analysis (from set3). One sample replicate has lower case sample name (bc9).

**Usage**
```r
data(set4)
```

**Format**
A data frame with 1609 rows and 5 variables
**set5**

*ESX17 example data for mixture analysis.*

**Description**

A slimmed dataset containing data from mixture experiment for Mx analysis.

**Usage**

data(set5)

**Format**

A data frame with 1663 rows and 7 variables

---

**set6**

*Fusion example data for dropout analysis.*

**Description**

A slimmed dataset containing data from sensitivity experiment for dropout analysis.

**Usage**

data(set6)

**Format**

A data frame with 1848 rows and 7 variables

---

**set7**

*ESSplex SE QS example data for inhibition analysis.*

**Description**

A slimmed dataset containing data from inhibition experiment.

**Usage**

data(set7)

**Format**

A data frame with 883 rows and 7 variables
**slim**

*Slim Data Frames*

**Description**

Slim data frames with repeated columns.

**Usage**

```r
slim(data, fix = NULL, stack = NULL, keep.na = TRUE, debug = FALSE)
```

**Arguments**

- **data**: data.frame.
- **fix**: vector of strings with column names to keep fixed.
- **stack**: vector of strings with column names to slim.
- **keep.na**: logical, keep a row even if no data.
- **debug**: logical indicating printing debug information.

**Details**

Stack repeated columns into single columns. For example, the following data frame: Sample.Name|Marker|Allele.1|Allele.2|Size.1|Size.2|Data.Point.

using this command: slim(data, fix=c("Sample.Name","Marker"), stack=c("Allele","Size")) would result in this data frame (NB! 'Data.Point' is dropped): Sample.Name|Marker|Allele|Size

**Value**

data.frame

---

**slim_gui**

*Slim Data Frames*

**Description**

GUI wrapper for the *slim* function.

**Usage**

```r
slim_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```
Arguments

env          environment in which to search for data frames and save result.
savegui     logical indicating if GUI settings should be saved in the environment.
debug       logical indicating printing debug information.
parent      widget to get focus when finished.

Details

Simplifies the use of the `slim` function by providing a graphical user interface to it.

Value

TRUE

See Also

`slim`

---

**sortMarker**  

**Sort Markers**

Description

Sort markers and dye as they appear in the EPG.

Usage

```r
sortMarker(data, kit, add.missing.levels = FALSE, debug = FALSE)
```

Arguments

- `data`  
  data.frame containing a column 'Marker' and optionally 'Dye'.
- `kit`    
  string or integer indicating kit.
- `add.missing.levels`  
  logical, TRUE missing markers are added, FALSE missing markers are not added.
- `debug`  
  logical indicating printing debug information.

Details

Change the order of factor levels for 'Marker' and 'Dye' according to 'kit'. Levels in data must be identical with kit information.

Value

data.frame with factor levels sorted according to 'kit'.
**Description**

GUI simplifying the use of the STR-validator package.

**Usage**

```r
strvalidator(debug = FALSE)
```

**Arguments**

- `debug`: logical indicating printing debug information.

**Details**

The graphical user interface give easy access to all graphical versions of the functions available in the strvalidator package. It connects functions ‘under the hood’ to allow a degree of automation not available using the command based functions. In addition it provides a project based workflow.

Click Index at the bottom of the help page to see a complete list of functions.

**Value**

TRUE

**Examples**

```r
# To start the graphical user interface.
## Not run:
strvalidator()
## End(Not run)
```

---

**Description**

Summarize Hb or Lb analysis data in table format.

**Usage**

```r
tableBalance(data, scope = "locus", quant = 0.05)
```
tableBalance_gui

Arguments

- **data**: data frame from a balance analysis by `calculateHb` and `calculateLb`.
- **scope**: string, summarize 'global' or 'locus'.
- **quant**: numeric, quantile to calculate.

Details

Summarize the Hb or Lb analysis in table format with different scope. (locus, or global). Returns a dataframe with columns for marker name 'Marker', number of allele ratios 'Xb.n', the minimum observed allele ratio 'Xb.Min', the mean allele ratio 'Xb.Mean', its standard deviation 'Xb.Stdv', the XXth percentile 'Xb.Perc.XX' For more details see `min`, `mean`, `sd`, `quantile`.

Value

data.frame with summarized result.

description

GUI wrapper for the `tableBalance` function.

Usage

```r
tableBalance_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```

Arguments

- **env**: environment in which to search for data frames.
- **savegui**: logical indicating if GUI settings should be saved in the environment.
- **debug**: logical indicating printing debug information.
- **parent**: widget to get focus when finished.

Details

Simplifies the use of the `tableBalance` function by providing a graphical user interface to it.

Value

TRUE

See Also

`tableBalance`
tableCapillary  

Table Capillary

Description
Summarize capillary analysis result in table format.

Usage
```r
tableCapillary(data, scope = "cap", debug = FALSE)
```

Arguments
- `data` : data frame from a capillary analysis by `calculateCapillary`.
- `scope` : character string. Make table by capillary, injection, plate row, run, or instrument. Values "cap", "inj", "row", "run", "instr".
- `debug` : logical indicating printing debug information.

Details
Summarize the capillary analysis result in table format by capillary, injection, plate row, or instrument. Returns a dataframe with number of observations, min, max, median, mean, standard deviation, and the 25th and 75th percentile.

Value
```r
data.frame with columns 'Instrument', 'Capillary/Injection/Row/Run/Instrument', 'N', 'Min', 'Q1', 'Median', 'Mean', 'Q3', 'Max', 'Std.Dev'.
```

tableCapillary_gui  

Table Capillary

Description
GUI wrapper for the `tableCapillary` function.

Usage
```r
tableCapillary_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
```
Arguments

env  
    environment in which to search for data frames.

savegui  
    logical indicating if GUI settings should be saved in the environment.

debug  
    logical indicating printing debug information.

parent  
    widget to get focus when finished.

Details

Simplifies the use of the `tableCapillary` function by providing a graphical user interface to it.

Value

TRUE

See Also

tableCapillary

description

Summarize precision analysis result in table format.

Usage

`tablePrecision(data, key = c("Marker", "Allele"), target = c("Size"), debug = FALSE)`

Arguments

data  
    Data frame containing at least columns defined in 'key' and 'target'.

key  
    vector containing column names to create keys from.

target  
    vector containing column <base> names to calculate precision for.

debug  
    logical indicating printing debug information.

Details

Calculates summary statistics for 'target' columns for each unique 'key' combination. For example the precision of determined size for alleles in multiple allelic ladders. Requires a 'slimmed' and 'filtered' data frame. For more details see `min`, `max`, `mean`, `sd`, `quantile`.

Value

data.frame with results.
**tablePrecision_gui**  
**Table Precision**

**Description**

GUI wrapper for the `tablePrecision` function.

**Usage**

```r
tablePrecision_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
                   parent = NULL)
```

**Arguments**

- `env` : environment in which to search for data frames and save result.
- `savegui` : logical indicating if GUI settings should be saved in the environment.
- `debug` : logical indicating printing debug information.
- `parent` : widget to get focus when finished.

**Details**

Simplifies the use of the `tablePrecision` function by providing a graphical user interface.

**Value**

`TRUE`

**See Also**

- `tablePrecision`, `checkSubset`

---

**tableStutter**  
**Table Stutter**

**Description**

Summarizes stutter analysis result in table format.

**Usage**

```r
tableStutter(data, scope = "stutter", quant = 0.95)
```
Arguments

- **data**: data frame from a stutter analysis by `calculateStutter`.
- **scope**: string, summarize 'global', by 'locus', or by 'stutter'.
- **quant**: numeric, quantile to calculate.

Details

Summarize the stutter analysis in table format with different scope (stutter, locus, or global). Returns a dataframe with columns for marker name 'Marker', stutter type 'Type', number of alleles 'n.alleles', number of stutters 'n.stutters', mean ratio 'Mean', standard deviation 'Stdv', the XXth percentile 'Perc.XX', and the maximum observed ratio 'Max'. For more details see `mean`, `sd`, `quantile`, `max`.

Value

data.frame with summarized result.

description

GUI wrapper for the `tableStutter` function.

Usage

```
tableStutter_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
                 parent = NULL)
```

Arguments

- **env**: environment in which to search for data frames.
- **savegui**: logical indicating if GUI settings should be saved in the environment.
- **debug**: logical indicating printing debug information.
- **parent**: widget to get focus when finished.

Details

Simplifies the use of the `tableStutter` function by providing a graphical user interface to it.

Value

TRUE

See Also

`tableStutter`
Description

Extract data from a dataset.

Usage

```r
trim(data, samples = NULL, columns = NULL, word = FALSE, ignore.case = TRUE, invert.s = FALSE, invert.c = FALSE, rm.na.col = TRUE, rm.empty.col = TRUE, missing = NA, debug = FALSE)
```

Arguments

- `data`: data.frame with genotype data.
- `samples`: string giving sample names separated by pipe (|).
- `columns`: string giving column names separated by pipe (|).
- `word`: logical indicating if a word boundary should be added to `samples` and `columns`.
- `ignore.case`: logical, TRUE ignore case in sample names.
- `invert.s`: logical, TRUE to remove matching samples from 'data', FALSE to remove samples NOT matching (i.e. keep matching samples).
- `invert.c`: logical, TRUE to remove matching columns from 'data', FALSE to remove columns NOT matching (i.e. keep matching columns). while TRUE will remove columns NOT given.
- `rm.na.col`: logical, TRUE columns with only NA are removed from 'data' while FALSE will preserve the columns.
- `rm.empty.col`: logical, TRUE columns with no values are removed from 'data' while FALSE will preserve the columns.
- `missing`: value to replace missing values with.
- `debug`: logical indicating printing debug information.

Details

Simplifies extraction of specific data from a larger dataset. Look for samples in column named 'Sample.Name', 'Sample.File.Name', or the first column containing the string 'Sample' in mentioned order (not case sensitive). Remove unwanted columns.

Value

data.frame with extracted result.
Description

GUI wrapper for the `trim` function.

Usage

```r
trim_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
         parent = NULL)
```

Arguments

- `env`: environment in which to search for data frames and save result.
- `savegui`: logical indicating if GUI settings should be saved in the environment.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.

Details

Simplifies the use of the `trim` function by providing a graphical user interface to it.

Value

`TRUE`

See Also

`trim`
Index

*Topic datasets
  ref1, 86
  ref11, 86
  ref2, 86
  ref3, 87
  ref4, 87
  ref5, 87
  ref6, 88
  ref61, 88
  ref6R, 88
  ref7, 89
  set1, 92
  set2, 93
  set3, 93
  set4, 93
  set5, 94
  set6, 94
  set7, 94

*Topic package
  strvalidator-package, 4

addColor, 6, 9
addData, 7, 8
addData_gui, 8
addDye_gui, 9
addMarker, 9, 10
addMarker_gui, 10
addOrder, 9, 11
addSize, 12, 12, 13
addSize_gui, 12
auditTrail, 13

calculateAllele, 14, 15, 89
calculateAllele_gui, 15
calculateAT, 16, 18–20, 73
calculateAT6, 17, 18, 19
calculateAT6_gui, 18, 18
calculateAT_gui, 18, 19, 19
calculateCapillary, 20, 21
calculateCapillary_gui, 21
calculateConcordance, 21, 22, 23
calculateConcordance_gui, 22
calculateCopies, 23, 24
calculateCopies_gui, 24
calculateDropout, 24, 26, 73, 75
calculateDropout_gui, 26
calculateHb, 27, 28
calculateHb_gui, 28
calculateHeight, 28, 30
calculateHeight_gui, 30
calculateLb, 31, 32, 33
calculateLb_gui, 32
calculateMixture, 33, 34, 35
calculateMixture_gui, 34
calculateOL, 35, 36
calculateOL_gui, 36
calculateOverlap, 36, 37, 38
calculateOverlap_gui, 37
calculatePeaks, 38, 39
calculatePeaks_gui, 39
calculatePullup, 40, 41
calculatePullup_gui, 41
calculateRatio, 41, 43
calculateRatio_gui, 43
calculateResultType, 43, 44, 45
calculateResultType_gui, 44
calculateSlope, 45, 46
calculateSlope_gui, 46
calculateSpike, 47, 48, 91
calculateSpike_gui, 48
calculateStutter, 48, 49, 50
calculateStutter_gui, 49
checkDataset, 50
checkSubset, 17, 19, 20, 26, 35, 41, 50, 51, 51, 52, 62, 67, 101
checkSubset_gui, 51
colConvert, 52
colNames, 53
columns, 53, 55
columns_gui, 55
combine_gui, 56, 57, 59
combineBinsAndPanels, 55, 72
cropData_gui, 57, 59
data.table, 15, 47
detectKit, 57editData_gui, 57, 58
export, 59, 60
export_gui, 60
filterProfile, 31, 42, 45, 61, 62
filterProfile_gui, 62
gdf, 58
generateEPG, 62, 64
generateEPG_gui, 64
geom_density, 79
getKit, 64
ggsave, 65, 66
ggsave_gui, 65
grep, 51
gtable, 58
guessProfile, 66, 67
guessProfile_gui, 67
heightToPeak, 68
import, 68, 70
import_gui, 70
list.files, 69
listObjects, 70
lm, 18
log, 79
makeKit_gui, 71
maskAT, 17, 19, 20, 72
model Dropout_gui, 73
plotAT_gui, 75
plotBalance_gui, 76
plotCapillary_gui, 77
plotContamination_gui, 77
plotDistribution_gui, 78
plot Dropout_gui, 75, 79
plotKit_gui, 80
plotPeaks_gui, 80
plotPrecision_gui, 81
plotPullup_gui, 82
plotRatio_gui, 83
plotResultType_gui, 83
plotSlope_gui, 84
plotStutter_gui, 85
rbind.fill, 56
read.table, 69
readBinsFile, 72
readPanelsFile, 72
ref1, 86
ref2, 86
ref3, 87
ref4, 87
ref51, 87
ref52, 88
ref61, 88
ref62, 88
ref7, 89
removeArtefact, 89, 90
removeArtefact_gui, 90
remove Spike, 90, 91
remove Spike_gui, 91
sample, 92
scrambleAlleles, 92
set1, 92
set2, 93
set3, 93
set4, 93
set5, 94
set6, 94
set7, 94
slim, 69, 95, 95, 96
slim_gui, 95
sortMarker, 10, 96
strvalidator, 97
strvalidator-package, 4
substr, 54
tableBalance, 97, 98
tableBalance_gui, 98
tableCapillary, 99, 99, 100
tableCapillary_gui, 99
tablePrecision, 100, 101
tablePrecision_gui, 101
tableStutter, 101, 102
tableStutter_gui, 102
INDEX

trim, 69, 103, 104
trim_gui, 57, 59, 104