Package ‘strvalidator’

January 18, 2016

Type Package

Title Process Control and Internal Validation of Forensic STR Kits

Version 1.6.0

Date 2016-01-18

Author Oskar Hansson

Maintainer Oskar Hansson <oskar.hansson@fhi.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), gWidgets, gridExtra, grid, gtable, plyr, scales, gWidgetsRGtk2, RGtk2, data.table, grDevices, graphics, stats, utils

Suggests ResourceSelection, testthat

Description An open source platform for validation and process control.
Tools to analyse 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

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-01-18 11:41:23
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>strvalidator-package</td>
<td>4</td>
</tr>
<tr>
<td>addColor</td>
<td>5</td>
</tr>
<tr>
<td>addData</td>
<td>6</td>
</tr>
<tr>
<td>addData_gui</td>
<td>7</td>
</tr>
<tr>
<td>addDye_gui</td>
<td>8</td>
</tr>
<tr>
<td>addMarker</td>
<td>9</td>
</tr>
<tr>
<td>addMarker_gui</td>
<td>9</td>
</tr>
<tr>
<td>addSize</td>
<td>10</td>
</tr>
<tr>
<td>addSize_gui</td>
<td>11</td>
</tr>
<tr>
<td>blockAT</td>
<td>12</td>
</tr>
<tr>
<td>calculateAllele</td>
<td>13</td>
</tr>
<tr>
<td>calculateAT</td>
<td>13</td>
</tr>
<tr>
<td>calculateAT6</td>
<td>15</td>
</tr>
<tr>
<td>calculateAT6_gui</td>
<td>16</td>
</tr>
<tr>
<td>calculateAT_gui</td>
<td>17</td>
</tr>
<tr>
<td>calculateBalance</td>
<td>17</td>
</tr>
<tr>
<td>calculateBalance_gui</td>
<td>18</td>
</tr>
<tr>
<td>calculateCapillary</td>
<td>19</td>
</tr>
<tr>
<td>calculateCapillary_gui</td>
<td>20</td>
</tr>
<tr>
<td>calculateConcordance</td>
<td>21</td>
</tr>
<tr>
<td>calculateConcordance_gui</td>
<td>22</td>
</tr>
<tr>
<td>calculateDropout</td>
<td>22</td>
</tr>
<tr>
<td>calculateDropout_gui</td>
<td>24</td>
</tr>
<tr>
<td>calculateHeight</td>
<td>25</td>
</tr>
<tr>
<td>calculateHeight_gui</td>
<td>26</td>
</tr>
<tr>
<td>calculateHeterozygous</td>
<td>27</td>
</tr>
<tr>
<td>calculateHeterozygous_gui</td>
<td>27</td>
</tr>
<tr>
<td>calculateLb</td>
<td>28</td>
</tr>
<tr>
<td>calculateLb_gui</td>
<td>29</td>
</tr>
<tr>
<td>calculateMixture</td>
<td>30</td>
</tr>
<tr>
<td>calculateMixture_gui</td>
<td>31</td>
</tr>
<tr>
<td>calculateOL</td>
<td>32</td>
</tr>
<tr>
<td>calculateOL_gui</td>
<td>33</td>
</tr>
<tr>
<td>calculateOverlap</td>
<td>33</td>
</tr>
<tr>
<td>calculateOverlap_gui</td>
<td>34</td>
</tr>
<tr>
<td>calculatePeaks</td>
<td>35</td>
</tr>
<tr>
<td>calculatePeaks_gui</td>
<td>36</td>
</tr>
<tr>
<td>calculatePullup</td>
<td>37</td>
</tr>
<tr>
<td>calculatePullup_gui</td>
<td>38</td>
</tr>
<tr>
<td>calculateRatio</td>
<td>38</td>
</tr>
<tr>
<td>calculateRatio_gui</td>
<td>40</td>
</tr>
<tr>
<td>calculateResultType</td>
<td>40</td>
</tr>
<tr>
<td>calculateResultType_gui</td>
<td>41</td>
</tr>
<tr>
<td>calculateSpike</td>
<td>42</td>
</tr>
<tr>
<td>calculateSpike_gui</td>
<td>43</td>
</tr>
<tr>
<td>calculateStutter</td>
<td>43</td>
</tr>
</tbody>
</table>
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 projects for easy access. Currently, analysis modules for stutter, balance, dropout, mixture, concordance, typing result, precision, pull-up, analytical thresholds, and drop-in are available. In addition there are functions to analyse 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 analyse the data. It allows exploration of the characteristics of DNA typing kits according to ENFSI and SWGDAM recommendations (see references). This facilitates the implementation of probabilistic interpretation of DNA results.

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.
addColor

Created and maintained by:
Oskar Hansson, Department of Forensic Biology (NIPH, Norway)

General user information and tutorials:
https://sites.google.com/site/forensicapps/strvalidator

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

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

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

Author(s)
Oskar Hansson <oskar.hansson@fhi.no>

References
Recommended Minimum Criteria for the Validation of Various Aspects of the DNA Profiling Process

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 labelled with a fluorescent dye. The dyes are represented with single letters (Dye) in exported result files or with strings (Color) in 'panels' files. For visualisation 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 on scheme to another.

Value

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

Examples

# Get marker and colors for SGM 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

# 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

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**

Add Dye Information

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` to add 'Dye' 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=Red, B=Blue). NB! If column 'Color' and 'Dye' is present they will be overwritten.

Value

TRUE
addMarker

See Also

   addColor

addMarker	Add Missing Markers.

Description

Add missing markers to a dataset given a kit.

Usage

   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.

addMarker_gui	Add Missing Markers

Description

GUI wrapper for the addMarker function.

Usage

   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

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.

Description

GUI wrapper for the `addSize` function.

Usage

```r
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`
blockAT

Block And Prepare Data To Analyze Analytical Threshold

Description

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

Usage

```r
blockAT(data, ref = NULL, block.height = TRUE, height = 500,
         block.sample = TRUE, per.dye = TRUE, range.sample = 20,
         block.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'.

block.height: logical to indicate if high peaks should be blocked.

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

block.sample: logical to indicate if sample allelic peaks should be blocked.

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

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

block.ils: logical to indicate if internal lane standard peaks should be blocked.

range.ils: integer to specify the blocking range in (+/-) data points. Active if block.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 blocking. 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'.

**calculateAllele**  

*Calculate Allele*

**Description**

Counts the number of each allele per marker over the entire dataset.

**Usage**

```r
calculateAllele(data, threshold = NULL, debug = FALSE)
```

**Arguments**

- **data**: data.frame including columns 'Marker', 'Allele', 'Height'.
- **threshold**: numeric if not NULL only peak heights above 'threshold' will be considered.
- **debug**: logical indicating printing debug information.

**Details**

Creates a sorted table of the most common alleles in the dataset. The list can be used to calculate allele frequencies or to identify artefacts. NB! Remove NA's and OL's prior to analysis.

**Value**

- data.frame

**See Also**

- *data.table*

---

**calculateAT**  

*Calculate Analytical Threshold*

**Description**

Calculate analytical thresholds estimates.

**Usage**

```r
calculateAT(data, ref = NULL, block.height = TRUE, height = 500,  
block.sample = TRUE, per.dye = TRUE, range.sample = 20,  
block.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'.
block.height  logical to indicate if high peaks should be blocked.
height  integer for global lower peak height threshold for peaks to be excluded from the analysis. Active if 'block.peak=TRUE.
block.sample  logical to indicate if sample allelic peaks should be blocked.
per.dye  logical TRUE if sample peaks should be blocked per dye channel. FALSE if sample peaks should be blocked globally across dye channels.
range.sample  integer to specify the blocking range in (+/-) data points. Active if block.sample=TRUE.
block.ils  logical to indicate if internal lane standard peaks should be blocked.
range.ils  integer to specify the blocking range in (+/-) data points. Active if block.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 analysing the background signal (noise). 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 analysed (i.e. not blocked) minus one as degrees of freedom, and the number of samples. If samples containing DNA are used a range around the allelic peaks can be blocked from the analysis to discard peaks higher than the noise. Blocking can be within each dye or across all dye channels. Similarly a range around the peaks of the internal lane standard (ILS) can be blocked across all dye channels. Which can bleed-through in week 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 two 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.
References


See Also

blockAT, 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'.
weighted  logical to calculate weighted linear regression (weight=1/se^2).
alpha  numeric [0,1] significance level for the t-statistic.
ignore.case  logical to indicate if sample matching should ignore case.
debug  logical to indicate if debug information should be printed.

Details

Calculate the analytical threshold (AT) according to method 6 as outlined in the reference. In short serial dilutions are analysed 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. AT6 = y-intercept + t-statistic * 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.).

Value

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

References


See Also

calculateAT6_gui, calculateAT, calculateAT_gui, lm

calculateAT6_gui Calculate Analytical Threshold

Description

GUI wrapper for the calculateAT6 function.

Usage

calculateAT6_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

calculateAT6, calculateAT, calculateAT_gui, checkSubset
**calculateAT_gui**

_**Calculate Analytical Threshold**_

**Description**

GUI wrapper for the `blockAT` 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

**See Also**

- `calculateAT`, `blockAT`, `checkSubset`

---

**calculateBalance**

_**Calculate Balance**_

**Description**

Calculates the inter and intra locus balance.

**Usage**

```r
calculateBalance(data, ref, lb = "prop", per.dye = TRUE, hb = 1, ignore.case = TRUE, word = FALSE, debug = FALSE)
```
calculateBalance_gui

Arguments

data a data frame containing at least 'Sample.Name', 'Marker', 'Height', 'Allele', and Dye'.
ref a data frame containing at least 'Sample.Name', 'Marker', 'Allele'.
1b string. 'prop' is default and locus balance is calculated proportionally 'norm' locus balance is normalised in relation to the locus with the highest total peakheight.
per.dye logical, default is TRUE and locus balance is calculated within each dye. FALSE locus balance is calculated globally across all dyes.
hb numerical, definition of heterozygous balance. Default is hb=1. hb=1: HMW/LMW, hb=2: LMW/HMW, hb=3: Max2(Ph)/Max1(Ph).
ignore.case logical indicating if sample matching should ignore case.
word logical indicating if word boundaries should be added before sample matching.
debug logical indicating printing debug information.

details

Calculates the inter and intra locus balance for a dataset. Only peaks corresponding to reference alleles will be included in analysis (does not require filtered data). Be careful to not have actual alleles marked as 'OL' in dataset. It will lead to underestimation of the total peak height per locus/sample. Also calculates the allele size difference between heterozygous alleles. NB! Requires at least one row for each marker per sample, even if no data. NB! 'X' and 'Y' will be handled as '1' and '2' respectively.

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

Examples
data(ref2)
data(set2)
# Calculate average balances.
calculateBalance(data=set2, ref=ref2)

calculateBalance_gui Calculate Balance

Description

GUI wrapper for the calculateBalance function.

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

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 \code{calculateBalance} function by providing a graphical user interface.

Value

\code{TRUE}

See Also

\code{link{calculateBalance}}, \code{link{checkSubset}}

\begin{itemize}
  \item \code{calculateCapillary}
  \item \code{Calculate Capillary Balance}
\end{itemize}

Description

Calculates the ILS inter capillary balance.

Usage

\begin{verbatim}
calculateCapillary(samples.table, plot.table, sq = 0, run = "",
  debug = FALSE)
\end{verbatim}

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 columns 'Instrument', 'Instrument.ID', 'Run', 'Mean.Height', 'SQ', 'Injection', 'Capillary', 'Well', 'Comment'.

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

Description

GUI wrapper for the calculateCapillary function.

Usage

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

calculateCapillary

calculateCapillary

calculateCapillary
calculateConcordance  \hspace{1cm} 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", delimiter = ",", 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.
- **delimiter**: character to separate the alleles in a genotype. Default is comma e.g '12,16'.
- **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).
calculateConcordance_gui

*Calculate Concordance*

**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.

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

**Value**
TRUE

**See Also**
*calculateConcordance*

calculateDropout

*Calculate Drop-out Events*

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

**Usage**
```
calculateDropout(data, ref, threshold = NULL, method = c("1", "2", "X",
            "L"), ignore.case = TRUE, 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.
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 Names" in 'ref' must be unique 'core' name of replicate sample names in 'data'. Use checksubset to make sure subsetting works as intended.

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 modelling. 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 modelling by logistic regression in function modelDropout: 'MethodX', 'Method1', 'Method2', 'MethodL' and 'MethodL.Ph'.

References

Peter Gill et.al., DNA commission of the International Society of Forensic Genetics: Recommendations on the evaluation of STR typing results that may include drop-out and/or drop-in using probabilistic methods, Forensic Science International: Genetics, Volume 6, Issue 6, December 2012,
calculateDropout_gui

Example

data(set4)
data(ref4)
drop <- calculateDropout(data=set4, ref=ref4, 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
calculateHeight

Calculate Peak Height.

Description

Calculate peak height metrics for samples.

Usage

```
calculateHeight(data, na = NULL, add = TRUE, exclude = NULL, debug = FALSE)
```

Arguments

- **data**: data.frame with at least columns 'Sample.Name' and 'Height'.
- **na**: replaces NA values.
- **add**: logical default is TRUE which will add/overwrite columns 'H', 'TPH', and 'Peaks' in the provided 'data'.
- **exclude**: character vector (case sensitive) e.g. "OL" excludes rows with "OL" in the 'Allele' column.
- **debug**: logical indicating printing debug information.

Details

Calculates the average peak height (H) and/or the total peak height (TPH) for each sample. To enable calculation of H the sample data must contain a column "Heterozygous", where 1 = heterozygous loci, and 0 = homozygous loci as known from the reference sample. Calculates H according to the formula: \[ H = \frac{\text{sum}(\text{peakheights})}{n[\text{het}] + 2n[\text{hom}]} \]

Value

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

References

calculateHeight_gui  Calculate Peak Height

Description
GUÍ wrapper for the calculateHeight function.

Usage
```
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

See Also
- calculateHeight
calculateHeterozygous  

**Calculate Heterozygous Loci**

**Description**

Calculates the number of alleles in each marker.

**Usage**

```r
calculateHeterozygous(data, debug = FALSE)
```

**Arguments**

- `data`: Data frame containing at least columns 'Sample.Name', 'Marker', and 'Allele*'.
- `debug`: logical indicating printing debug information.

**Details**

Adds a column 'Heterozygous' to 'data'. Calculates the number of unique values in the 'Allele*' columns for each marker. Indicates heterozygous loci as '1' and homozygous as '0'. Sample names must be unique.

**Value**

data.frame the original data frame containing additional columns.

---

**calculateHeterozygous_gui**

**Calculate Heterozygous Loci**

**Description**

GUI wrapper for the `calculateHeterozygous` function.

**Usage**

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

**Arguments**

- `env`: environment in which to search for data frames and save result.
- `debug`: logical indicating printing debug information.
- `parent`: widget to get focus when finished.
calculateLb

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

Value
TRUE

See Also
calculateHeterozygous

calculateLb

Calculate Inter-locus Balance

Description
Calculates the inter-locus balance.

Usage
calculateLb(data, ref = NULL, option = "prop", by.dye = FALSE, ol.rm = TRUE, sex.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 normalised 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.
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 autodetect 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.
The inter-locus balance (Lb), or profile balance, can be calculated as a proportion of the whole, normalised, 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 or sex markers. If not automatic detection is attempted. If missing, dye will be added according to kit. Off-ladder alleles is 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)
Usage

calculateLb_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 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


calculateMixture_gui  Calculate Mixture

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**  
*Analyse Off-ladder Alleles*

**Description**

Analyse the risk for off-ladder alleles.

**Usage**

`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 analysing 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

Analyse Off-ladder Alleles

Description

GUI wrapper for the calculateOL function.

Usage

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

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 analysing 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 neighbouring dye can be counted in full (100 while a non neighbour 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 neighbouring dye can be counted in full (100 while a non neighbour dye get its overlap reduced (to e.g. 10

Value

TRUE

See Also

calculateOverlap

calculatePeaks( data, bins = c(0, 2, 3), labels = c("No contamination", "Drop-in contamination", "Gross contamination"), nool = FALSE, permarker = FALSE, debug = FALSE)
Details

Count the number of peaks in a sample profile based on values in the 'Height' column. Each sample can be labelled according to custom labels defined by the number of peaks. Peaks can be counted per sample or per marker per 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 sample. If 'File' is present this allows for identical sample names in different batches (files) to be identified as different 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**

```r
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*
**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,
                 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.
- `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
numerator character vector with marker names.
deponentominator character vector with marker names.
group character column name to group by.
o. 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 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","TH01"), 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

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

Arguments

env environment in which to search for data frames and save results.
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 `calculateResultType` by providing a graphical user interface.

Value
TRUE

See Also
`calculateResultType`

calculateSpike  Detect Spike

Description
Detect samples with possible spikes in the DNA profile.

Usage
`calculateSpike(data, threshold = NULL, round.to = 1, kit = NULL, 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 an unlabelled peak).
- `round.to` numeric tolerance for Size.
- `kit` string or numeric for the STR-kit used (NULL = auto detect).
- `debug` logical indicating printing debug information.

Details
Creates a list of possible spikes by searching for peaks aligned vertically (i.e. nearly identical size).

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! Labelled 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_gui

checkSubset  
Check Subset

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.
default 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

checkSubset_gui  
Check Subset

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`
### 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.

### Column Actions

**Description**

Perform actions on columns.

**Usage**

```
columns(data, col1 = NA, col2 = NA, operator = "&", fixed = NA, target = NA, 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.
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.
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.

Examples

# 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")

Description

GUI wrapper for the `columns` function.

Usage

columns_gui(env = parent.frame(), savegui = NULL, debug = FALSE,
            parent = NULL)
combineBinsAndPanels
  Combine Bins And Panels Files.

Description
Combines useful information into one dataset.

Usage
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
## combine_gui

**Combine Datasets**

**Description**

GUI for combining two datasets.

**Usage**

```r
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` function. NB! Datasets must have identical column names.

**Value**

TRUE

## cropData_gui

**Crop Or Replace**

**Description**

GUI simplifying cropping and replacing values in data frames.

**Usage**

```r
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 dropdown 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 behaviour. 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**

```r
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 for enable edit.
- `debug`  
  logical indicating printing debug information.
- `parent`  
  widget to get focus when finished.

**Details**

Select a data frame from the dropdown and view/edit. Optionally save as a new dataframe.

**Value**

TRUE

**See Also**

`trim_gui`, `cropData_gui`, `combine_gui`
Description

GUI wrapper for the `export` function.

Usage

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

Arguments

- `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.

Value

`TRUE`

See Also

- `export`

Description

Filter peaks from profiles.

Usage

```r
filterProfile(data, ref, add.missing.loci = FALSE, keep.na = FALSE,
              ignore.case = TRUE, exact = FALSE, invert = FALSE, debug = FALSE)
```
filterProfile_gui

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.
extact logical TRUE use exact matching of sample names.
invert logical TRUE filter peaks NOT matching the reference.
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). 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

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_gui**

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 area expansion (to avoid clipping of labels).

debug logical for printing debug information to the console.

**Details**

Generates an 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 customise 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**

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 `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, debug = 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 '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)
```

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

Guess Profile

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

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

guessProfile_gui  Guess Profile

Description

GUI wrapper for the guessProfile function.

Usage

guessProfile_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 `guessProfile` function by providing a graphical user interface to it.

Value

TRUE

See Also

`guessProfile`, `checkSubset`

---

**import**

**Import Data**

---

**Description**

Import text files and apply post processing.

**Usage**

```
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)
```

**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).
### 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`
listObjects

Details

Simplifies the use of the \texttt{import} function by providing a graphical user interface to it.

Value

\texttt{TRUE}

See Also

\texttt{import}

\begin{Verbatim}
\textbf{listObjects} \hspace{1em} \textit{List Objects}
\end{Verbatim}

Description

Internal helper function to list objects in an environment.

Usage

\texttt{listObjects(env = parent.frame(), obj.class = NULL, debug = FALSE)}

Arguments

\begin{itemize}
\item \texttt{env} \hspace{1em} environment in which to search for objects.
\item \texttt{obj.class} \hspace{1em} character string or vector specifying the object class.
\item \texttt{debug} \hspace{1em} logical indicating printing debug information.
\end{itemize}

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

\begin{verbatim}
## 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)
\end{verbatim}
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

modelDropout_gui  Model And Plot Drop-out Events

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 modelled by logistic regression as described in Appendix B the first reference. 

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

where 'H' is the peak height or log(peak height). 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 modelling, and the peak height is used as explanatory variable. The average peak height 'H' can be used instead of the allele/locus peak height. Optionally, the logarithm of the peak height can be used. To evaluate the goodness of fit for the logistic regression the Hosmer-Lemeshow test is used. A value below 0.05 indicates a poor fit. Alternatives to the logistic regression method are discussed in reference 4 and 5. [13,32].

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 modelling. 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.Ph - peak height of the surviving allele if one allele has dropped out, or the average peak height if no drop-out.

### Value

TRUE
References


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.
**plotBalance_gui**

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.

---

**Description**

GUI simplifying the creation of plots from balance data.

Usage

```r
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 autodetected). 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

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

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

plotDistribution_gui  

Plot Distribution

Description

GUI simplifying the creation of distribution plots.

Usage

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

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>env</code></td>
<td>environment in which to search for data frames and save result.</td>
</tr>
<tr>
<td><code>savegui</code></td>
<td>logical indicating if GUI settings should be saved in the environment.</td>
</tr>
<tr>
<td><code>debug</code></td>
<td>logical indicating printing debug information.</td>
</tr>
<tr>
<td><code>parent</code></td>
<td>widget to get focus when finished.</td>
</tr>
</tbody>
</table>

#### Details

Plot the distribution of data as cumulative distribution function, probability density function, or count. First select a dataset, then select a group (if any), finally select a column to plot the distribution of. It is possible to overlay a boxplot. Various smoothing kernels and bandwidths can be specified. More info on kernels and bandwidth: [http://www.inside-r.org/r-doc/stats/density](http://www.inside-r.org/r-doc/stats/density)

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 dropout data.

### Usage

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

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>env</code></td>
<td>environment in which to search for data frames and save result.</td>
</tr>
<tr>
<td><code>savegui</code></td>
<td>logical indicating if GUI settings should be saved in the environment.</td>
</tr>
<tr>
<td><code>debug</code></td>
<td>logical indicating printing debug information.</td>
</tr>
<tr>
<td><code>parent</code></td>
<td>widget to get focus when finished.</td>
</tr>
</tbody>
</table>

#### 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.
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.
- debug: 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

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 customise titles and font size. Data can be plotted as
frequency or proportion. The values can be printed on the plot with custom number of decimals.
There are several colour palettes to choose 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

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

See Also

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

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 autodetected). 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 as an image.
**plotRatio_gui**

**Value**

TRUE

**See Also**

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

---

**plotRatio_gui**  
*Plot Ratio*

**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  

**Plot Result Type**

**Description**

GUI simplifying the creation of plots from result type data.

**Usage**

```r
plotResultType_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 customise 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 colour 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

plotStutter_gui  

**Plot Stutter**

**Description**

GUI simplifying the creation of plots from stutter data.

**Usage**

```r
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.

Description

A dataset in `GeneMaper` 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 `GeneMaper` 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
**scrambleAlleles**  
*Scramble Alleles*

**Description**
Scrambles alleles in a dataset to anonymise the profile.

**Usage**
```
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 allows 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**
```
data(set1)
```

**Format**
A data frame with 170 rows and 13 variables
**set2**

*SGMPlus 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

slim

Slim Data Frames

Description

Slim data frames with repeated columns.

Usage

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

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

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'.

---

strvalidator                  Graphical User Interface For The STR-validator Package

Description

GUI simplifying the use of the STR-validator package.

Usage

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

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

## End(Not run)

description

Summarize balance analysis data in table format.

Usage

tableBalance(data, scope = "locus", quant = 0.05)

Arguments

data data frame from a balance analysis by calculateBalance.
scope string, summarize 'global' or 'locus'.
quant numeric, quantile to calculate.

Details

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

Value

data.frame with summarized result.
tableBalance_gui  \textit{Table Balance}

\underline{Description}

GUI wrapper for the \textit{tableBalance} function.

\underline{Usage}

\begin{verbatim}
tableBalance_gui(env = parent.frame(), savegui = NULL, debug = FALSE, parent = NULL)
\end{verbatim}

\underline{Arguments}

\begin{itemize}
  \item \texttt{env} \hspace{1cm} environment in which to search for data frames.
  \item \texttt{savegui} \hspace{1cm} logical indicating if GUI settings should be saved in the environment.
  \item \texttt{debug} \hspace{1cm} logical indicating printing debug information.
  \item \texttt{parent} \hspace{1cm} widget to get focus when finished.
\end{itemize}

\underline{Details}

Simplifies the use of the \textit{tableBalance} function by providing a graphical user interface to it.

\underline{Value}

TRUE

\underline{See Also}

\textit{tableBalance}

---

tableCapillary  \textit{Table Capillary}

\underline{Description}

Summarize capillary analysis result in table format.

\underline{Usage}

\begin{verbatim}
tableCapillary(data, scope = "cap", debug = FALSE)
\end{verbatim}
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".

depg 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

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

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

Arguments

env environment in wich to search for data frames.

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

d 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
tablePrecision

*Calculate Precision*

**Description**

Summarize precision analysis result in table format.

**Usage**

```r
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

`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.
### tableau_stutterGUI

**Description**

GUI wrapper for the `tablestutter` function.

**Usage**

```r
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`

---

### trim

**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.

trim_gui  Trim Data

Description

GUI wrapper for the trim function.

Usage

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, 75
  ref11, 76
  ref2, 76
  ref3, 76
  ref4, 77
  ref51, 77
  ref52, 77
  set1, 78
  set2, 79
  set3, 79
  set4, 79
  set5, 80

*Topic **package**
  strvalidator_package, 4

addColor, 5, 8, 9
addData, 6, 7, 8
addData_gui, 7
addDye_gui, 8
addMarker, 9, 9, 10
addMarker_gui, 9
addSize, 10, 11
addSize_gui, 11

blockAT, 12, 15, 17

calculateAllele, 13
calculateAT, 13, 16, 17
calculateAT6, 15, 16
calculateAT6_gui, 16, 16
calculateAT_gui, 16, 17
calculateBalance, 17, 18, 19
calculateBalance_gui, 18
calculateCapillary, 19, 20
calculateCapillary_gui, 20
calculateConcordance, 21, 22
calculateConcordance_gui, 22
calculateDropout, 22, 24, 65, 66
calculateDropout_gui, 24

calculateHeight, 25, 26
calculateHeight_gui, 26
calculateHeterozygous, 27, 28
calculateHeterozygous_gui, 27
calculateLb, 28, 29, 30
calculateLb_gui, 29
calculateMixture, 30, 31, 32
calculateMixture_gui, 31
calculateOL, 32, 33
calculateOL_gui, 33
calculateOverlap, 33, 34, 35
calculateOverlap_gui, 34
calculatePeaks, 35, 36
calculatePeaks_gui, 36
calculatePullup, 37, 38
calculatePullup_gui, 38
calculateRatio, 38, 40
calculateRatio_gui, 40
calculateResultType, 40, 41, 42
calculateResultType_gui, 41
calculateSpike, 42, 43
calculateSpike_gui, 43
calculateStutter, 43, 44, 45
calculateStutter_gui, 44
checkDataset, 45
checkSubset, 15–17, 24, 32, 38, 45, 46, 46, 47, 56, 61, 87
checkSubset_gui, 46
colConvert, 47
colNames, 48
columns, 48, 49, 50
columns_gui, 49
combine_gui, 51, 52, 53
combineBinsAndPanels, 50, 64
cropData_gui, 51, 53
data.table, 13, 42
detectKit, 52
editData_gui, 52, 53
export, 54
export_gui, 54

filterProfile, 28, 39, 54, 55, 56
filterProfile_gui, 55

generateEPG, 56, 57
generateEPG_gui, 57
getKit, 58
ggsave, 59
ggsave_gui, 59
grep, 46
guessProfile, 59, 60, 61
guessProfile_gui, 60

import, 61, 62, 63
import_gui, 62

list.files, 62
listObjects, 63
lm, 16

makeKit_gui, 64
modelDropout_gui, 64

plotAT_gui, 66
plotBalance_gui, 67
plotCapillary_gui, 68
plotDistribution_gui, 68
plotDropout_gui, 66, 69
plotKit_gui, 70
plotPeaks_gui, 71
plotPrecision_gui, 71
plotPullup_gui, 72
plotRatio_gui, 73
plotResultType_gui, 74
plotStutter_gui, 74

rbind, 51
read.table, 61, 62
readBinsFile, 64
readPanelsFile, 64
ref1, 75
ref11, 76
ref2, 76
ref3, 76
ref4, 77
ref51, 77
ref52, 77

sample, 78
scrambleAlleles, 78
set1, 78
set2, 79
set3, 79
set4, 79
set5, 80
slim, 62, 80, 81
slim_gui, 81
sortMarker, 9, 81
strvalidator, 82
strvalidator-package, 4
tableBalance, 83, 84
tableBalance_gui, 84
tableCapillary, 84, 85
tableCapillary_gui, 85
tablePrecision, 86, 86, 87
tablePrecision_gui, 86
tableStutter, 87, 88
tableStutter_gui, 88
trim, 62, 88, 89, 90
trim_gui, 52, 53, 89