Package 'drLumi'

September 24, 2015

Type Package

Title Multiplex Immunoassays Data Analysis

Description Contains quality control routines for multiplex immunoassay data, including several approaches for: treating the background noise of the assay, fitting the dose-response curves and estimating the limits of quantification.

Version 0.1.2

Depends R (>= 3.2.2)

Imports reshape, Hmisc, chron, gdata, plyr, stringr, ggplot2, minpack.lm, stats, irr, rootSolve, msm

Date 2015-09-20

VignetteBuilder knitr

Suggests knitr, testthat, xtable

License GPL (>= 2)

NeedsCompilation no

Author Hector Sanz [aut, cre],
John Aponte [aut],
Jaroslaw Harezlak [aut],
Yan Dong [aut],
Magdalena Murawska [aut],
Clarissa Valim [aut],
Aintzane Ayestaran [ctb],
Ruth Aguilar [ctb],
Gemma Moncunill [ctb]

Maintainer Hector Sanz <hector.sanz@isglobal.org>

Repository CRAN

Date/Publication 2015-09-24 01:27:27

R topics documented:

conf_bands 2
data_selection 3

ecdata 5
est_conc 6
get_outliers 7
intra_icc 8
invest 10
loq_cv 11
loq_derivatives 12
loq_interval 13
lum_export 15
lum_import 16
mfidata 17
plot.scluminex 17
scluminex 19
SSexp 21
SSexpcons 22
SSl4 23
SSl4cons 24
SSl5 25
SSl5cons 27
summary.scluminex 28

Index **30**

`conf_bands` *Confidence interval for a scluminex object*

Description

Computes confidence or prediction interval for the response variable given a concentration value.

Usage

```
conf_bands(x, analyte = NULL, xvalue, level = 0.95,
  interval = "confidence")
```

Arguments

`x`	a `scluminex` object.
`analyte`	character vector specifying the analytes to estimate the interval. Default `NULL` (all analytes).
`xvalue`	vector of numeric values of the concentration.
`level`	numeric value for interval confidence or prediction level. Default 0.95.
`interval`	character defining type of interval, either 'confidence' or 'prediction'. Default 'confidence'.

Details

Two types of interval can be estimated 'prediction' interval and 'confidence' interval. If the did not converge the function returns NA for all xvalue. If the function cannot estimate the value NaN is returned.

Value

A data.frame with predicted response, lower and upper confidence (prediction) limits, standard error, concentration value, analyte, interval method and background method.

References

Ruckstuhl, A. (2010). Introduction to Nonlinear Regression. http://stat.ethz.ch/~stahel/courses/cheming/nlreg10E.pdf

See Also

predict.nls

Examples

```
# Load data and fit models
data(ecdata)
data(mfidata)

plate1 <- mfidata[mfidata$plate=="plate_1",]
datasets <- data_selection(plate1, ecfile = ecdata)

background <- datasets[[1]]$background
standard <- datasets[[1]]$standard
mod <- scluminex(plateid = "plate_1", standard = standard,
background = background, bkg = "ignore",lfct="SSl4",
fmfi = "mfi", verbose = FALSE)

# Confidence-prediction intervals for FGF analyte
conf_bands(mod, "FGF", xvalue = c(1,3,4), interval = "confidence")
conf_bands(mod, "FGF", xvalue = c(1,3,4), interval = "prediction")

# For all analytes the prediction interval
conf_bands(mod, xvalue = 0.5, interval = "prediction")
```

---

data_selection *Extracts samples information from a lum_export object or data.frame*

Description

Extracts from a lum_export or data.frame object controls, dilutions points, background and samples to be calibrated. These files can be merged to an expected concentration and flag dataset.

Usage

```
data_selection(x, ecfile = NULL, flagsfile = NULL, backname = "Back",
    stanname = "Stan", posname = "Con", unkname = NULL,
    byvar.ecfile = c("analyte", "sample"), byvar.flagsfile = c("well",
    "analyte"), fsample = "sample", fanalyte = "analyte", fbatch = "plate",
    ...)
```

Arguments

x	a lum_export object or data.frame with samples information.
ecfile	a data.frame or CSV path file with the expected concentration in order to merge to x.
flagsfile	a data.frame or CSV path with the flags information in order to merge to x.
backname	character vector or list of the background samples identification of x.
stanname	character vector or list of the standard points identification of x.
posname	character vector or list of the positive controls identification of x.
unkname	character vector or list of the samples identification of x. Default NULL.
byvar.ecfile	character vector of the merging variable(s) of x and ecfile.
byvar.flagsfile	character vector of the merging variable(s) of x and flagsfile.
fsample	character vector of the name of the sample variable.
fanalyte	character vector of the name of the analyte variable.
fbatch	character vector of the name of the Batch variable.
...	other options. Ignored.

Details

Default method for identifying background, standard and positives samples is to define a character vector of length one and apply agrep functions in order to extract databases. Samples to be calibrated (unknowns) are identified as the remaining ones (default NULL).

If the arguments are defined as a list the function will subset exactly that information from x object.

The expected concentration file is merged based on byvar.ecfile. Only applies when ecfile is not NULL. Same applies for flags file. A variable named 'flag' must be in flags data in order to perform the merge.

Value

The name of the batch in a list format with the following components: background, standard, positive and unknowns datasets.

Examples

```
# Load data
data(ecdata)
data(mfidata)

dat <- subset(mfidata,plate=="plate_1" & analyte=="FGF")

# Example 1
sdf <- data_selection(dat)

lapply(sdf$plate_1, function(x) head(x))

# Example 2 (merge ecdata)
sdf <- data_selection(dat, ecfile = ecdata,
             byvar.ecfile=c("analyte","sample"))

lapply(sdf$plate_1, function(x) head(x))

# Example 3 (extract specific samples names with list)
sdf <- data_selection(dat,
             stanname=list("Standard10"),
             backname = list("Background0"),
             posname = list("Control1","Control2"),
             unkname = list("B_sid_13_CSP"))

lapply(sdf$plate_1, function(x) head(x))

# Example 4 (extract aproximate names samples)
sdf <- data_selection(dat,
             stanname="Standard1",
             backname = "Background0",
             posname = "Control1",
             unkname = "B_sid_13_CSP")

lapply(sdf$plate_1, function(x) (x))
```

---

ecdata *Expected concentration data*

Description

Expected concentration data based on multiplex immunoassays experiments.

Format

A `data.frame` with the expected concentration of 30 analytes of one plate. It consists in 1 background sample, 3 positives controls and 16 dilution points (first dilution is duplicated).

- `sample`: type of sample (background, control or standard).

- analyte: analyte tested.
- ec: expected concentration value.

---

est_conc *Estimates concentration given an scluminex object*

Description

Given a scluminex object with standard curve information and a data.frame with response values add to the original dataset the concentration data.

Usage

```
est_conc(x, df, fanalyte = "analyte", fmfi = "median", dilution = 1,
    one.curve = FALSE, level = 0.95)
```

Arguments

x	a scluminex object.
df	input data.frame with the analyte and median fluorescence intensity variables.
fanalyte	name of the field with the analyte information. Default 'analyte'.
fmfi	name of the field with the mfi (response) information. Default 'median'.
dilution	numeric value of the dilution that must be used for the estimation of the concentration.
one.curve	logical according if only one curve must be used for estimation.
level	numeric value, confidence level, required. Default 0.95.

Details

Given a scluminex object and a data.frame with analyte and MFI information adds the concentration information to the dataset (concentration, standard error of the concentration and a warning variable). The MFI data will be transformed into log10(MFI). The method utilized is the Delta method of invest function.

Merging variables are defined in the fanalyte and fmi arguments of the function.

If only one standard curve is fitted for several analytes one.curve argument must be specified to TRUE and scluminex must have only one analyte information. The same scluminex information will be used for all analytes of the df data.frame.

If one standard curve is fitted by each analyte one.curve must be FALSE, so the function will merge each model of the scluminex object with the corresponding analyte of the df argument.

Value

Input `data.frame` with the following merged variables:

- `log10.fitted.conc`, log10 fitted concentration
- `log10.fitted.conc.se`, log10 standard error of the log10 fitted concentration
- `dilution`, dilution to be applied to the samples
- `dil.fitted.conc`, diluted fitted concentration in original scale
- `dil.lb.conc`, diluted fitted lower bound concentration in original scale
- `dil.ub.conc`, diluted fitted upper bound concentration in original scale
- `warning`, warning message (if necessary)

Examples

```
# Load data and fit models
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]
sdf <- data_selection(dat, ecdata)$plate_1

igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
            lfct="SSl4", bkg="ignore", fmfi="mfi", verbose=FALSE)

# Data to estimate concentration
concdf <- sdf$positive

# Dilution factor 1
est_conc(igmodels, concdf, fmfi="mfi", dilution = 1)

# Dilution factor 2
est_conc(igmodels, concdf, fmfi="mfi", dilution = 2)
```

---

get_outliers *Extract outliers from a scluminex object*

Description

This function extracts outliers based on a residuals cutoff from `scluminex` object.

Usage

```
get_outliers(x, out.limit = 2.5)
```

Arguments

`x`	a `scluminex` object
`out.limit`	cutoff point for outliers

Details

The residuals are the standardized ones for `nls` models.

Value

A `data.frame` with samples whose standardized residual is greater than `out.limit`.

Examples

```
# Load data and estimate models
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

# Select data and fit models
sdf <- data_selection(dat, ecdata)[[1]]
igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
                      lfct=c("SSl4", "SSl5"),
                      bkg="ignore",
                      fmfi="mfi",
                      verbose=FALSE)

# Extract outliers > abs(2.5)
get_outliers(igmodels, out.limit=2.5)

# Extract outliers > abs(1.5)
get_outliers(igmodels, out.limit=1.5)

# Extract outliers > abs(1)
get_outliers(igmodels, out.limit=1)
```

`intra_icc` *Estimate ICC of a data.frame*

Description

Given a `data.frame` in long format estimate ICC from the `irr` package

Usage

```
intra_icc(x, id.var = c("sample", "analyte"), value.var = "dil.fitted.conc",
    by = NULL, ...)
```

Arguments

x	a data.frame in long format with analyte, control number and concentration variables
id.var	one or more variables that identify each one of replicates samples
value.var	character vector with the name of the variable to estimate icc
by	character vector of the variable to stratify the icc results
...	arguments for the icc function from the irr package

Details

The icc function is the one from the irr package.

Value

A list with three objects is returned:

- icc.df, trasnformed data.frame in order to perform the icc analysis
- icc.mod, the model output of the icc
- icc.value, the icc value of the icc model

See Also

irr

Examples

```
# Generate data.frame
set.seed(123)
controls <- sort(paste("Control", rep(1:3,4),sep=""))
values <-  sort(unlist(lapply(1:12, function(x)runif(1,10+x,13+x))))
plateno <- rep(c("plate1","plate2"),6)
df <- data.frame(controls,values, plateno)
df <- df[order(df$plateno),]

# Estimate ICC
intra_icc(df, id.var = c("controls","plateno"),
value.var = "values", type="agreement",model="twoway", unit="single")
intra_icc(df, id.var = c("controls","plateno"),
value.var = "values", by = "plateno", type="agreement",model="twoway",
unit="single")
```

`invest` *Estimate the concentration given a response value*

Description

Estimates the inverse of the funtion. Given a response value, estimates the corresponding concentration value and the standard error.

Usage

```
invest(x, analyte=NULL, yvalue, ci.method = c("delta", "bootstrap"),
        level = 0.95, seed.boot = 123, nboot = 100)
```

Arguments

`x`	a `scluminex` object.
`analyte`	the specific analyte to estimate the invert values. Default `NULL` (all analytes).
`yvalue`	value of the response model to estimate the inverse in log10 scale.
`ci.method`	character defining the method to be applied for estimating standard error ('delta' or 'bootstrap'). Default 'delta'.
`level`	confidence level. Default 0.95.
`seed.boot`	numeric for the seed of the bootstrap method. Only applies for bootstrap method. Default 123.
`nboot`	number of bootstrap replicates. Only applies for bootstrap method. Default 100.

Details

Delta method function used is `deltamethod` from the `msm` package. Bootstrap method generates `nboot` response vectors (assuming normality) and fit the same model with original concentration data. The confidence interval is calculated by the percentile method specified in the `level` argument (`1-level/2`, `1-(1-level)/2`).

Value

A `data.frame` with the following components:

- MFI variable, the `yvalue` response vector
- Fit of the concentration, concentration estimation of the `yvalue` vector
- Fit of the concentration.lci, lower confidence bounds for the concentration estimation
- Fit of the concentration.uci, upper confidence bounds for the concentration estimation
- Fit of the concentration.se, estimation of the Standard Error of the concentration. If `ci.method` 'bootstrap' is `NA`

Examples

```
# Load data
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

# Estimate models
sdf <- data_selection(dat, ecdata)[[1]]
igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
lfct="SSl4", bkg="ignore", fmfi="mfi", verbose=FALSE)

# Delta
invest(igmodels, "FGF", c(2, 2.5, 3),  "delta")

# Bootstrap
invest(igmodels, "FGF" ,c(2, 2.5, 3), "bootstrap", nboot=10)
```

`loq_cv` *Limits of quantifications estimation using coefficient of variation*

Description

Estimates the limits of quantification based on an approximation of the coefficient of variation.

Usage

```
loq_cv(x, subset.list = NULL, max.cv = 0.2, n.cuts = 100)
```

Arguments

`x`	a `scluminex` object
`subset.list`	list of analytes to estimate. Default `NULL` (all analytes of the `scluminex` object).
`max.cv`	is the target coeficient of variation by default 0.2
`n.cuts`	is the number of cuts to search the coefficent of variation. Default 100.

Details

For each value of the response, the estimated concentration value and the approximated standard deviation is estimated. The function `invest` with the delta method approach is used. The coefficient of variation of the log10 concentration is calculated as the

$$\sqrt{e^{(SE \times log(10))^2} - 1}$$

Value

Object of class `loq`.

References

Gottschalk PG, and Dunn JR. (2005). Determining the error of dose estimates and minimum and maximum acceptable concentrations from assays with nonlinear dose-response curves. *Comput Methods Programs Biomed* **80**, 204-215.

Defawe OD, Fong Y, Vasilyeva E, Pickett M, Carter DK, Gabriel E, Rerks-Ngarm S, Nityaphan S, Frahm N, McElrath MJ and De Rosa SC.(2012). Optimization and qualification of a multiplex bead array to assess cytokine and chemokine production by vaccine-specific cells. *J Immunol Methods* **382**, 117-128.

Examples

```
# Load data and estimate models
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)$plate_1

igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
lfct=c("SSl4", "SSl5"), bkg="ignore", fmfi="mfi", verbose=FALSE)

loq_cv(igmodels, max.cv=0.25, n.cuts=100)
```

---

loq_derivatives *Limits of quantifications estimation using derivatives*

Description

Estimates the limits of quantification based on the second order derivative of the functions.

Usage

```
loq_derivatives(x, subset.list = NULL, ...)
```

Arguments

x	a scluminex class object
subset.list	list of analytes to estimate. Default NULL (all analytes of the scluminex object)
...	further arguments to be passed to uniroot.all function.

Details

The limits of quantification are based on the maximum and minimum points of the second order derivative of the function. The solution is based on the specific expressions of the derivatives. The uniroot.all function is used in order to find the global maximum and minimum.

Value

Object of class loq.

References

Ritz C and Spiess AN (2008). qpcR: an R package for sigmoidal model selection in q uantitative real-time polymerase chain reaction analysis. *Bioinformatics* **24**, 1549-51.

Examples

```
# Load data and estimate models
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)[[1]]

igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
                lfct=c("SSl4", "SSl5"),
                bkg="ignore",
                fmfi="mfi",
                verbose=FALSE)

loq_derivatives(igmodels)
```

---

loq_interval *Limits of quantifications estimation using interval method*

Description

Estimates the limits of quantification based on the asymptotes coefficients.

Usage

```
loq_interval(x, subset.list = NULL, low.asymp = NULL, high.asymp = NULL,
    lowci = -Inf, highci = Inf, inter.method = "prediction", level = 0.95)
```

Arguments

x	a scluminex object
subset.list	list of analytes to estimate. Default NULL (all analytes of the scluminex object)
low.asymp	a number or character specifying the low asymptote coefficient in the model. Default NULL assumes LLOQ is the minimum value of the concentration variable.
high.asymp	a number or character specifying the high asymptote coefficient. Default NULL assumes HLOQ is the maximum value of the concentration variable

lowci	specify the lowest limit if exists for asymptote, only applies if low.asymp equals !NULL.
highci	specify the highest limit if exists for asymptote, only applies if high.asymp equals !NULL.
inter.method	interval method for estimating interval LOQ ('prediction' or 'confidence')
level	0 to 1 value specifying level of confidence. Default 0.95.

Details

If low.asymp (high.asymp) is specified lowci (highci) must be -Inf (Inf). When lowci (highci) is specified asymptote argument must be NULL. When low.asymp and lowci arguments are the default values, the funtion assumes that LOQs are maximum and minimum values of data. If low.asymp (high.asymp) or lowci (highci) arguments are specified but it is not possible to estimate the LOQ (e.g., coefficient position is not well specified or estimated values are beyond observed data) the function estimates the LOQs based on maximum and minimum values. When the background method is the constraint one, the LLOQ is the concentration value of the log10(Background MFI mean) and low.asymp and lowci doesn't apply.

Value

Object of class loq.

References

Quinn CP, Semenova VA, Elie CM et al. (2002). Specific, Sensitive, and Quantitative Enzyme-Linked Immunosorbent Assay for Human Immunoglobulin G Antibodies to Anthrax Toxin Protective Antigen. *Emerg Infect Dis* **8 (10)**,1103-10

Examples

```
# Load data and estimate models
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)$plate_1

igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
                lfct="SSl4",
                bkg="ignore",
                fmfi="mfi",
                verbose=FALSE)

# All default arguments
loq_interval(igmodels)

# Low asymptote coefficient of the model is 2
loq_interval(igmodels, low.asymp = 2)
```

```
# Low asymptote coefficient of the model is 2 and high is 3
loq_interval(igmodels, low.asymp = 2, high.asymp = 3)

# Fix to 2 and 3 the lower and upper asymptote
loq_interval(igmodels, lowci=2, highci=3)
```

---

lum_export *Extract and generates data.frame from lum_import object*

Description

Extract, generates and export different information from a lum_import object

Usage

```
lum_export(x, y = NULL, path = NULL, backname = "ackgro",
    stanname = "tandar", samplename = "Sample",
    dilutionname = "Dilution Factor", batchname = "batch", ...)
```

Arguments

x	a lum_import object with all information of the xPONENT software imported
y	an optional lum_import object different from x
path	the directory name where all files are exported. The directory is created according to batch name and system date. Default is NULL
backname	character vector that defines background
stanname	character vector that defines standard
samplename	character vector that defines the name of the samples variable
dilutionname	character vector that defines the name of the dilution factor variable
batchname	character vector that defines the name of the batch field in CSV
...	other parameters of the function

Details

This function returns a list with all datasets from an object of class lum_import. Also is possible to export into several csv files (for Fluorescence-type data) or zip file (for Bead-type data).

Examples

```
# Read all data
imp_path <-  system.file(c("inst","extdata"),"plate1.csv",
package="drLumi")
imp <- lum_import(imp_path)
exp <- lum_export(imp)
names(exp)
```

```
# See variables
imp$well_vars

# Select only 2
imp$well_vars <- c("Median", "Net MFI")
exp <- lum_export(imp)
head(exp$well)
```

`lum_import` *Import Luminex data*

Description

Import xPONENT software raw data from Luminex assays and converts in an easy-to-work file

Usage

```
lum_import(x, ...)
```

Arguments

`x`	a file or directory of files for Fluorescence and Metadata-type or Bead files respectively
`...`	other parameters of the function

Details

The list of variables that return is based on the main structure of luminex information.

Value

If import data is Fluorescence-type data several objects are returned in a list format:

- `dtblock`, blocks of information from raw data
- `raw_metadata`, raw metadata
- `well_vars`, list of variables for the well data
- `scurve_vars`, list of variables for the standard curve data
- `average_vars`, list of variables for the average data
- `batch_vars`, list of variables for the batch data
- `name_batch`, name of the batch
- `type_raw_data`, Fluorescence
- `region_vars`, list of variables for the region data
- `sample_vars`, list of variables for the sample data

If import data is Bead-type data two objects are returned in a list format:

- `bead_files`, all information from bead csv files
- `name_batch`, name of the batch
- `type_raw_data`, Bead

See Also

lum_export

Examples

```
imp_path <-  system.file(c("inst","extdata"),"plate1.csv",
package="drLumi")
imp <- lum_import(imp_path)
imp
```

---

mfidata *Median Fluorescence Intensity data*

Description

Dataset with the median fluorescence intensity values for three 96-wells plates (from the same experiment), with 30 analytes information per plate (multiplex experiment).

Format

A `data.frame` with the median fluorescence intensity values of three differents plates based on 30 analytes perplate. The variables are:

- `plate`: plate/batch information
- `well`: well position in plate based on 96 wells plate
- `analyte`: analyte (30 per well)
- `sample`: type of sample (blank, standard, positive control or sample to be analyzed)
- `mfi`: median fluorescence intensity

---

plot.scluminex *Plot method for the scluminex class*

Description

This function takes a `scluminex` object and creates a standard curve, residuals or QQ-plot using `ggplot2` package.

Usage

```
## S3 method for class 'scluminex'
plot(x, type = "scurve", subset.list = NULL,
    psize = 1.8, ncol = NULL, nrow = NULL, out.limit = 2.5,
    size.text = 1.5, size.legend = 2.5, interval = "confidence",
    level = 0.95, color.bkg = "green", ...)
```

Arguments

x	an object of class scluminex
type	character describing the type of plot ('scurve','residuals' or 'qqplot'). Default 'scurve'.
subset.list	list of analytes to be plotted. Default all analytes.
psize	numeric point size
ncol	number of columns to plot the analytes.
nrow	number of rows to plot the analytes.
out.limit	value that defines an outlier. Must be positive value. Only applies for type 'residuals'.
size.text	value that defines the size of the well into the residuals plot. Only applies for type 'residuals'.
size.legend	size of the legend. NA for not showing. Only applies for type 'scurve'.
interval	'confidence' or 'prediction' character in order to plot the fit and the corresponding bands. If NULL only observed points are plotted. Only applies for type 'scurve'.
level	confidence level for the interval. Default 0.95, only applies for type 'scurve'.
color.bkg	character specifying the color of the background line. NA for not showing background. Only applies for type 'scurve'.
...	other arguments to be passed to ggplot function

Details

All information in order to generate the plots is extracted from the scluminex object.

Value

A ggplot object

Examples

```
# Load data and estimate models
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)[[1]]
sdf_luminex <- scluminex("plate_1",sdf$standard, sdf$background,
"SSl4", bkg="ignore", fmfi="mfi", verbose=FALSE)

# Plot standard curves
plot(sdf_luminex, "sc")

# Plots residuals
plot(sdf_luminex, "res")
```

```
# Plot QQplot
plot(sdf_luminex, "qq")
```

---

scluminex *Estimates different models for each analyte.*

Description

Given a dilutions points and background `data.frame` estimates a model (in a recursive way is possible) for a background method.

Usage

```
scluminex(plateid, standard, background, lfct,
    bkg = c("ignore","subtract", "include", "constraint"),
    neill.method = "finest", fmfi = "median", fec = "ec",
    fanalyte = "analyte", fwell = "well", fflag = "flag",
    verbose = TRUE, ...)
```

Arguments

`plateid`	character to identify the plate
`standard`	a `data.frame` with standard values information
`background`	a `data.frame` with the value of the blank controls.
`lfct`	a character vector of SelfStarting models for background method. They will be used in order if no convergence is achieved, ie: the first `lfct` first, if no convergence the second function, etc. Options are `SSl5`, `SSl4` and `SSexp`.
`bkg`	character vector specifying how the background values are treated. Options are 'ignore', 'subtract', 'include' or 'constraint'.
`neill.method`	character specifying the grouping method for the Neill test. Default 'finest'. Other options 'c-finest', 'percentiles' or the name of the grouping variable.
`fmfi`	name of the column with MFI values
`fec`	name of the column with the concentration
`fanalyte`	name of the column with the name of the analyte
`fwell`	name of the variable with the well information
`fflag`	name of the variable with the flag to not include a record in the standard curve estimation
`verbose`	logical whether show the process of estimation.
`...`	other parameters for the model

Details

The models are fitted by the `nlsLM` function from the `minpack.lm` package. The background data can be ignore, or use to subtract the values of all MFI or be included as a point in the standard curve with a value half of the lower value of the standard points. If two or more blank controls are specified the geometric mean of the MFI is used. The names on the two datasets need to be the same and are specified by the fmfi, fec and fanalyte arguments of the function. The routine should receive the values of the MFI from the luminex fluorescence data. Analysis is performed in logarithm scale (base 10) both for the MFI and the concentrations.

The grouping variable for the `neill.method` can specified if there are replicates of doses in the assay. If there are no replicates one of the three 'grouping' methods can be selected.

Value

A list with the following components model, convergence, coef, data, rsquare

- `model`, the nls model
- `convergence`, convergence of the model
- `coef`, coefficients values for the `nls` model
- `data`, data of the model
- `rsquare`, R^2 values for the performed models

Examples

```
# Load data
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)$plate_1

# Fit model and summary object
igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
                lfct=c("SSl4", "SSl5"),
                bkg="ignore",
                fmfi="mfi",
                verbose=FALSE)
ss <- summary(igmodels)

# Information
names(igmodels)
names(igmodels$FGF)

# Summary data
ss
as.data.frame(ss)
as.data.frame(igmodels)

# Plot the standard curve
plot(igmodels,"sc")
```

SSexp *Self-Starting Nls exponential regression model*

Description

This selfStart model evaluates the exponential growth regression model and its gradient. It has an `initial` attribute that will evaluate initial estimates of the parameters `y0`, and `b` for a given set of data. Instead of the standard `exp` function this implementation use the `10^` function.

$$f(x) = y_0 \times 10^b$$

Usage

```
SSexp(x, b, y0)
```

Arguments

`x`	a numeric vector of values at which to evaluate the model
`b`	a numeric parameter representing the growth rate
`y0`	a numeric parameter representing the value of the response when x is 0

Format

A selfStart model

Value

The value returned is a list containing the nonlinear function, the self starter function and the parameter names.

Author(s)

John Aponte <john.aponte@cresib.cat>

Examples

```
# Load data
data(ecdata)
data(mfidata)

# Select analyte FGF for plate 1
dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)$plate_1

ig <- scluminex("plate_1",sdf$standard, sdf$background,
            lfct="SSexp",
            bkg="ignore",
```

```
            fmfi="mfi",
            verbose=FALSE)

summary(ig)
```

SSexpcons *Self-Starting Nls exponential constraint regression model*

Description

This selfStart model evaluates the exponential growth regression model and its gradient. It has an `initial` attribute that will evaluate initial estimates of the parameters `y0`, and `b` for a given set of data. Instead of the standard `exp` function this implementation use the `10^` function.

$$f(x) = y_0 \times 10^b$$

Usage

```
SSexpcons(..constraint.value, x, b)
```

Arguments

`..constraint.value`	a numeric value representing the value of the response when `x` is 0. In this function this value is not a parameter is just a numeric value to constraint `y_0` parameter.
`x`	a numeric vector of values at which to evaluate the model
`b`	a numeric parameter representing the growth rate

Format

A selfStart model

Value

The value returned is a list containing the nonlinear function, the self starter function and the parameter names.

Examples

```
# Load data
data(ecdata)
data(mfidata)

# Select analyte FGF for plate 1
dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)$plate_1
```

```
cons <- scluminex("plate_1",sdf$standard, sdf$background,
            lfct="SSexp",
            bkg="constraint",
            fmfi="mfi",
            verbose=FALSE)

summary(cons)

# Comparison constraint vs no constraint (same returning value but estimate
# one parameter).
b <- 3
y0 <- 1
concentration <- 2
SSexp(concentration, b, y0)
SSexpcons(y0, concentration, b)
```

SSl4 *Self-Starting Nls 4 parameters logistic regression model*

Description

This selfStart model evaluates the 4 parameters logistic regression model and its gradient. It has an `initial` attribute that will evaluate initial estimates of the parameters `hAsym`, `lAsym`, `Slope` and `xMid` for a given set of data Instead of the standard `exp` function this implementation use the `10^` function.

$$f(x) = lAsym + \frac{hAsym - lAsym}{1 + 10^{Slope(x - xMid)}}$$

Usage

```
SSl4(x, Slope, lAsym, hAsym, xMid)
```

Arguments

`x`	a numeric vector of values at which to evaluate the model
`Slope`	a numeric parameter representing the `-slope` of the function at the inflection point
`lAsym`	a numeric parameter representing the lower asymptote when `x` trend to `-Inf`
`hAsym`	a numeric parameter representing the higher asymptote when `x` trend to `Inf`
`xMid`	is the x value corresponding to the inflection point

Format

A selfStart model

Value

The value returned is a list containing the nonlinear function, the self starter function and the parameter names.

Author(s)

John Aponte <john.aponte@cresib.cat>

Examples

```
# Load data
data(ecdata)
data(mfidata)

# Select analyte FGF for plate 1
dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)[[1]]

ig <- scluminex("plate_1",sdf$standard, sdf$background,
           lfct="SSl4",
           bkg="ignore",
           fmfi="mfi",
           verbose=FALSE)

summary(ig)
```

`SSl4cons` *Self-Starting Nls 4 parameters logistic constraint regression model*

Description

This selfStart model evaluates the 4 parameters logistic regression model and its gradient. It has an `initial` attribute that will evaluate initial estimates of the parameters `hAsym`, `Slope` and `xMid` for a given set of data. Instead of the standard `exp` function this implementation use the `10^` function.

$$f(x) = lAsym + \frac{hAsym - lAsym}{1 + 10^{Slope(x - xMid)}}$$

Usage

```
SSl4cons(..constraint.value, x, Slope, hAsym, xMid)
```

Arguments

`..constraint.value`	a numeric value representing the lower asymptote when `x` trend to `-Inf`. In this function this value is not a parameter is just a numeric value to constraint `lAsym` parameter.
`x`	a numeric vector of values at which to evaluate the model
`Slope`	a numeric parameter representing the `-slope` of the function at the inflection point
`hAsym`	a numeric parameter representing the higher asymptote when `x` trend to `Inf`
`xMid`	is the x value corresponding to the inflection point

Format

A selfStart model

Value

The value returned is a list containing the nonlinear function, the self starter function and the parameter names.

Examples

```
# Load data
data(ecdata)
data(mfidata)

# Select analyte FGF for plate 1
dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)[[1]]

cons <- scluminex("plate_1",sdf$standard, sdf$background,
           lfct="SSl4",
           bkg="constraint",
           fmfi="mfi",
           verbose=FALSE)

summary(cons)

# Comparison constraint vs no constraint (same returning value but
# estimate 3 parameters).
lAsym <- 1
Slope <- 2
hAsym   <- 2
xMid <- 3
concentration <- 2
SSl4(concentration, Slope, lAsym, hAsym, xMid)
SSl4cons(lAsym, concentration, Slope, hAsym, xMid)
```

SSl5 *Self-Starting Nls 5 parameters logistic regression model*

Description

This selfStart model evaluates the 5 parameters logistic regression model and its gradient. It has an `initial` attribute that will evaluate initial estimates of the parameters `hAsym`, `lAsym`, `Slope`, `xMid` and `Asymetry` for a given set of data Instead of the standard `exp` function this implementation use the `10^` function.

$$f(x) = lAsym + \frac{hAsym - lAsym}{(1 + 10^{Slope(x - xMid)})^{Asymetry}}$$

Usage

```
SSl5(x, Slope, lAsym, hAsym, xMid, Asymetry)
```

Arguments

x	a numeric vector of values at which to evaluate the model
Slope	is a numeric parameter representing the -slope of the function at the inflection point
lAsym	a numeric parameter representing the lower asymptote when x trend to -Inf
hAsym	a numeric parameter representing the higher asymptote when x trend to Inf
xMid	is the x value corresponding to the inflection point
Asymetry	is a numeric parameter representing the asymetry around the inflection point

Format

A selfStart model

Value

The value returned is a list containing the nonlinear function, the self starter function and the parameter names.

Author(s)

John Aponte <john.aponte@cresib.cat>

Examples

```
# Load data
data(ecdata)
data(mfidata)

# Select analyte FGF for plate 1
dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)[[1]]

# SSl5
ig <- scluminex("plate_1",sdf$standard, sdf$background,
            lfct="SSl5",
            bkg="ignore",
            fmfi="mfi",
            verbose=FALSE)

summary(ig)
```

SSl5cons *Self-Starting Nls 5 parameters logistic constraint regression model*

Description

This selfStart model evaluates the 5 parameters logistic regression model and its gradient for the lower asymptote constraint method. It has an `initial` attribute that will evaluate initial estimates of the parameters `hAsym`, `Slope`, `xMid` and `Asymetry` for a given set of data Instead of the standard `exp` function this implementation use the `10^` function.

$$f(x) = lAsym + \frac{hAsym - lAsym}{(1 + 10^{Slope(x - xMid)})^{Asymetry}}$$

Usage

```
SSl5cons(..constraint.value,x, Slope, hAsym, xMid, Asymetry)
```

Arguments

`..constraint.value`	a numeric value representing the lower asymptote when `x` trend to `-Inf`. In this function this value is not a parameter is just a numeric value to constraint `lAsym` parameter.
`x`	a numeric vector of values at which to evaluate the model
`Slope`	is a numeric parameter representing the `-slope` of the function at the inflection point
`hAsym`	a numeric parameter representing the higher asymptote when `x` trend to `Inf`
`xMid`	is the x value corresponding to the inflection point
`Asymetry`	is a numeric parameter representing the asymetry around the inflection point

Format

A selfStart model

Value

The value returned is a list containing the nonlinear function, the self starter function and the parameter names.

Examples

```
# Load data
data(ecdata)
data(mfidata)

# Select analyte FGF for plate 1
dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]
```

```
sdf <- data_selection(dat, ecdata)[[1]]

# SSl5
cons <- scluminex("plate_1",sdf$standard, sdf$background,
            lfct="SSl5",
            bkg="constraint",
            fmfi="mfi",
            verbose=FALSE)

summary(cons)

# Comparison constraint vs no constraint (same returning value but estimate
# 4 parameters).
lAsym <- 1
Slope <- 2
hAsym   <- 2
xMid <- 3
Asymetry <- 1.5

concentration <- 2
SSl5(concentration, Slope, lAsym, hAsym, xMid, Asymetry)
SSl5cons(lAsym, concentration, Slope, hAsym, xMid, Asymetry)
```

`summary.scluminex` *Summary of a scluminex object*

Description

Summary of a scluminex object

Usage

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

Arguments

`object`	object of `scluminex` class.
`...`	other summary arguments. Ignored.

Value

An object of class summary.scluminex with the following fields:

- parameters of the model, coefficients, standard errors, t and p values
- obs, number of observations
- rsquare, R squared of the model

- modelfit, p value for goodnes of fit
- convergence, convergence code for the model (1 = converged, 0 = otherwise)
- plateid, plate identification name
- fct, function used to fit the model

Examples

```
# Load data
data(ecdata)
data(mfidata)

dat <- mfidata[mfidata$plate=="plate_1" & mfidata$analyte=="FGF",]

sdf <- data_selection(dat, ecdata)$plate_1

# Fit model and summary object
igmodels <- scluminex("plate_1",sdf$standard, sdf$background,
                lfct=c("SSl4", "SSl5"),
                bkg="ignore",
                fmfi="mfi",
                verbose=FALSE)
summary(igmodels)
```

Index

∗Topic **datasets**
SSexp, 21
SSexpcons, 22
SSl4, 23
SSl4cons, 24
SSl5, 25
SSl5cons, 27
∗Topic **data**
ecdata, 5
mfidata, 17

agrep, *4*

conf_bands, 2

data_selection, 3
deltamethod, *10*

ecdata, 5
est_conc, 6

get_outliers, 7

icc, *9*
intra_icc, 8
invest, *6*, 10, *11*

loq_cv, 11
loq_derivatives, 12
loq_interval, 13
lum_export, 15, *17*
lum_import, 16

mfidata, 17

nls, *8*

plot.scluminex, 17
predict.nls, *3*

scluminex, *6*, 19
SSexp, 21
SSexpcons, 22
SSl4, 23
SSl4cons, 24
SSl5, 25
SSl5cons, 27
summary.scluminex, 28

uniroot.all, *12*