Package ‘nCal’

March 4, 2015

LazyLoad yes
LazyData yes
Version 2015.3-3
Title Nonlinear Calibration
Author Youyi Fong <yfong@fhcrc.org>, Krisztian Sebestyen <ksebestyen@gmail.com>, Xuesong Yu <xyu@scharp.org>
Maintainer Youyi Fong <yfong@fhcrc.org>
Depends R (>= 3.0.0), drc, gdata, gWidgets, kyotil
Suggests knitr, RUnit, rjags, gWidgetscltk, nlme
VignetteBuilder knitr
Description Performs nonlinear calibration and curve fitting for data from Luminex, RT-PCR, ELISA, RPPA etc. Its precursor is Ruminex.
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2015-03-04 09:36:02

R topics documented:

  bcrm ................................................................. 2
 crm.fit ............................................................ 6
  dat.QIL3 .......................................................... 8
  drm.fit ............................................................. 8
  ED5PL ............................................................ 10
  elisa.R.gh ........................................................ 11
  get.abc ............................................................. 12
  getConc .......................................................... 13
  gnls.fit ........................................................... 13
  hier.model.ex.2 .................................................. 14
  ncal ............................................................... 15
  p.eotaxin ......................................................... 20
  plot5PL ............................................................ 21
  read.luminex.xls ................................................ 22
Description

`bcrm` fit concentration-response curves with a Bayesian random effects model using JAGS

Usage

```r
bcrm(formula, data, 
parameterization=c("gh","classical"),
error.model=c("norm","t4","mixnorm","mix2","replicate_re","tar1","lar1"),
prior=c("cytokine","BAMA","RT-PCR","ELISA","default"),
prior.sensitivity=c("none","1","2","3","4"),
mean.model=c("5PL","4PL"),
n.iter=1e5, jags.seed=1, n.thin=NULL, T.init=NULL, 
keep.jags.samples=FALSE, standards.only=TRUE, n.adapt=1e3, 
t.unk.truth=NULL, params.true=NULL, # for simulation study use 
verbose=FALSE
)
```

## S3 method for class 'bcrm'
```r
plot(x, 
assay_id=NULL, fit.2=NULL, fit.3=NULL, 
points.only=FALSE, all.lines.only=FALSE, 
same ylim=FALSE, lty3=NULL, lcol2=NA, lcol3=NA, 
lcol=1, lwd=.1, lty=1, # for lines 
t=NULL, log="x", col.outliers=TRUE,pch.outliers=TRUE, 
use.dif.pch.for.replicate=FALSE, main=NULL, 
additional.plot.func=NULL, add=FALSE, ... 
)
```

## S3 method for class 'bcrm'
```r
print(x, ...) 
```

## S3 method for class 'bcrm'
```r
coef(object, type="gh", ...) 
```

## S3 method for class 'bcrm'
```r
vcov(object, type="gh", ...) 
```

## S3 method for class 'bcrm'
```r
getVarComponent(object, ...) 
```
Arguments

- **formula**: formula. Gives the response column and concentration column.
- **data**: a data frame. Each row represents the measurement from one well/bead_type.
  - See details
- **parameterization**: string.
- **error.model**: string.
- **prior**: string.
- **mean.model**: mean model
- **pch.outliers**: pch for outliers
- **n.iter**: a number indicating the number of iterations to run.
- **jags.seed**: a number to seed the random number generator used within jags.
- **n.thin**: a number specifying the thinning factor applied to the jags samples.
- **keep.jags.samples**: boolean. If TRUE, the fit object being returned has an element named "jags.samples". coef samples will always be saved in "coef.samples".
- **t.unk.truth**: True unknown concentrations, for simulation study use only.
- **params.true**: True curve parameters, for simulation study use only.
- **T.init**: a integer vector. Initial values for mixture indicators.
- **prior.sensitivity**: integer. A number between 1 and 4. Change priors.
- **standards.only**: boolean. If TRUE, data is subset to standard samples only.
- **n.adapt**: integer. Passed to jags.model. If 0, then no adaptation happens and reproducible results can be obtained from jags.model.
- **verbose**: boolean. If TRUE, debug messages are printed.
- **x**: bcrm fit object.
- **object**: bcrm fit object.
- **type**: string. 5PL parameterization.
- **fit**: bcrm fit object.
- **assay_id**: string. Label for the assay run.
- **add**: Boolean. If TRUE, adding to an existing plot.
- **lcol**: integer. Line color.
- **fit.2**: a bcrm object. A second fit object to be plotted together with x.
- **lwd**: numeric. Line width.
- **points.only**: Boolean. If TRUE, only the data points are plotted and not the fitted curves
- **all.lines.only**: Boolean. If TRUE, only the fitted curves are plotted.
- **t**: a numeric vector. The log concentrations.
- **same.ylim**: Boolean. If TRUE, all fitted curves are plotted with the same ylim.
lty3 integer. lty for plotting fit.3.
fit.3 bcrm object. A third optional bcrm object to be plotted.
lcol2 integer. Line color for plotting fit.2.
lcol3 integer. Line color for plotting fit.3.
col.outliers Boolean. If TRUE, outliers are colored differently.
main string.
lty integer. Line type for plotting x.
log string. If it is "x", then the x axis is labeled on the scale of concentration; otherwise, it is labeled on the scale of log concentration.
additional.plot.func function, to be called after plotting each fit
use.dif.pch.for.replicate boolean

Details
data is expected to contain one to many plates with the same analyte.

- well_role Defines the role of a well. This should be from Standard, Unknown, .... Standard wells are used to generate standard curves, and concentrations of the substance in the Unknown well will be estimated.
- assay_id Identifies an assay, which is defined to be a collection of Standard and non-Standard wells. Measured fi from the Standard wells are used to create a set of standard curves, one of each bead type. Based on the standard curves and the measured fi from the non-Standard wells, concentrations of the substance in the non-Standard wells will be estimated. An assay can be a plate, if every plate has Standard wells; or it can be multiple plates run by one technician at one time, if there are only one set of Standard wells on these plates.
- dilution Standard samples are often prepared by starting with one sample and doing serial dilutions. Unknown samples may be measured at several dilutions so that one of the dilutions may fall into the more reliable region of the standard curve.
- replicate Index of technical replicates for a sample. Typical values are 1 or 2. May be used in plotting. Optional
- expected_conc Standard samples have expected concentrations. If this column is present, the dilution and starting_conc are optional and will not be used. This column does not apply to non-Standard samples.

Main error.model supported: drc, classical_norm, classical_t4, classical_mixnorm, classical_lar1, gh_norm, gh_mixnorm, gh_lar1 Also support: classical_replicate_re, gh_replicate_re, gh_lar1

Only two replicates are supported for now for the correlated noise models.
Sometimes jags.model fails with one .RNG.seed. The function will increase it by 1 and try again. The function tries three times before giving up.

Value
An object of type bcrm.
Author(s)
Youyi Fong <yfong@fhcrc.org>

References

Examples

```r
crm

set.seed(1)
log.conc=log(1e4)-log(3)*9:0
n.replicate=2
fi=simulate1curve(p.eotaxin[1,], rep(log.conc,each=n.replicate), sd.e=0.3)
dat.std=data.frame(fi, expected_conc=exp(rep(log.conc,each=n.replicate)), analyte="test",
    assay_id="assay1", sample_id=NA, well_role="Standard", dilution=rep(3**(9:0), each=n.replicate))
# add unknown
dat.unk=rbind(
    data.frame(fi=exp(6.75), expected_conc=NA, analyte="test", assay_id="assay1",
        sample_id=1, well_role="Unknown", dilution=1)
, data.frame(fi=exp(6.70), expected_conc=NA, analyte="test", assay_id="assay1",
        sample_id=2, well_role="Unknown", dilution=1)
, data.frame(fi=exp(3), expected_conc=NA, analyte="test", assay_id="assay1",
        sample_id=3, well_role="Unknown", dilution=1)
, data.frame(fi=exp(4.4), expected_conc=NA, analyte="test", assay_id="assay1",
        sample_id=4, well_role="Unknown", dilution=10)
)
dat=rbind(dat.std, dat.unk)
# second plate
fi=simulate1curve(p.eotaxin[2,], rep(log.conc,each=n.replicate), sd.e=0.3)
dat.std=data.frame(fi, expected_conc=exp(rep(log.conc,each=n.replicate)), analyte="test",
    assay_id="assay2", sample_id=NA, well_role="Standard", dilution=rep(3**(9:0), each=n.replicate))
# add unknown
dat.unk=rbind(
    data.frame(fi=exp(6.75), expected_conc=NA, analyte="test", assay_id="assay2",
        sample_id=1, well_role="Unknown", dilution=1)
, data.frame(fi=exp(6.70), expected_conc=NA, analyte="test", assay_id="assay2",
        sample_id=2, well_role="Unknown", dilution=1)
, data.frame(fi=exp(3), expected_conc=NA, analyte="test", assay_id="assay2",
        sample_id=3, well_role="Unknown", dilution=1)
, data.frame(fi=exp(4.4), expected_conc=NA, analyte="test", assay_id="assay2",
        sample_id=4, well_role="Unknown", dilution=10)
)
dat=rbind(dat, dat.std, dat.unk)

fits = bcrm(log(fi)-expected_conc, dat, parameterization="gh", error.model="norm",
            prior="cytokine", n.iter=6e3)
par(mfrow=c(1,2))
plot(fits)
```
# Not run:
# takes longer

# Example from Fong et al. (2012)
fits.t4 = bcrm (log(fi)-expected_conc, dat.QIL3, parameterization="gh", error.model="t4", prior="cytokine")
par(mfrow=c(2,3))
plot(fits.t4)

fits.norm = bcrm (log(fi)-expected_conc, dat.QIL3, parameterization="gh", error.model="norm", prior="cytokine")
par(mfrow=c(2,3))
plot(fits.norm)

## End(Not run)

crm.fit  

*Fit Concentration Response Model*

## Description

crm.fit can fit a constant or power variance function or log transform both sides.

## Usage

```r
crm.fit (formula, data, fit.4pl=FALSE, var.model=c("constant","power"),
          robust="mean", method=c("gls-pl","gnls","mle"), max.iter=50, reltol=1e-3,
          gof.threshold=0.2, log.both.sides=FALSE, verbose=FALSE)
```

## Arguments

- `formula`
- `data`
- `fit.4pl`  
  Boolean
**Details**

`crm.fit` implements an iterative method for estimating a model with power variance. `method`: gls-pl means GLS-PL (see reference) `log.both.sides`: transform both sides (see reference)

**Value**

An object of crm and drm type.

- `var.power` estimated power parameter in the power variance function

**References**


**Examples**

```r
dat.std<-dat.QIL3[dat.QIL3$assay_id=="LMX001",]

# run 3 iter to save time for examples
fit.1<-crm.fit(fi~expected_conc, dat.std, var.model="power", verbose=TRUE, max.iter=2)
fit.2<-crm.fit(log(fi)~expected_conc, dat.std, verbose=TRUE)
fit.3<-crm.fit(log(fi)~expected_conc, dat.std, var.model="power", verbose=TRUE, max.iter=2)

sapply(list(fit.1, fit.2, fit.3), coef)

fit.1$var.power
fit.2$var.power
fit.3$var.power

plot(fit.1, log="xy", type="all", lwd=3, pch="*")
lines(fit.2, expy=TRUE, col=2, lwd=3)
lines(fit.3, expy=TRUE, col=4, lty=2, lwd=3)
```
**dat.QIL3**

*An example for hierarchical modeling used in Fong, Wakefield, De Rosa, Frahm (2012)*

**Description**

An example for hierarchical modeling used in Fong, Wakefield, De Rosa, Frahm (2012)

**Format**

A data frame with 120 observations on the following 13 variables.

- **well**: a character vector. Well identifier on a microplate.
- **assay_id**: a character vector. Assay identifier.
- **analyte**: a character vector. Substance to be measured.
- **well_role**: a character vector. Defines the role of a well. See `?ncal` for more information.
- **beadct**: a numeric vector. Number of beads in the well, specific to multiplex bead array assay.
- **dilution**: a numeric vector. The dilution factor of a sample.
- **expected_conc**: a numeric vector. The expected concentrations of a sample.
- **fi**: a numeric vector. Fluorescence intensity readout.
- **replicate**: a numeric vector. Technical replicate identifier.

**References**


---

**drm.fit**

*Fit drm*

**Description**

`drm.fit` fit concentration-response curves using `drm` function from `drc` package.

**Usage**

```r
drm.fit (formula, data, robust="mean", fit.4pl=FALSE, w=NULL, gof.threshold=.2, verbose=FALSE, bcVal = NULL, bcAdd = 0)
```

```r
## S3 method for class 'drc'
getVarComponent(object, ...)
```
Arguments

- **formula**: a formula object.
- **data**: a data frame object.
- **robust**: a string. Passed to drm. See ?drm for more details.
- **fit.4pl**: boolean. If TRUE, 4PL model is fitted. If FALSE, 5PL model is fitted.
- **gof.threshold**: a threshold to determine when to try more self start functions.
- **w**: weights.
- **bcVal**: numeric, passed to drm.
- **bcAdd**: numeric, passed to drm.
- **...**: ...
- **verbose**: Boolean. If TRUE, verbose messages are printed.
- **object**: a drm object.

Details

drm.fit differs from drc:: drm in several aspects.
(1) It tries several self start functions in order to get better fits.
(2) It uses gof.threshold to report lack of fit.
(3) It tried to determine whether the standard deviation of the parameter estimates can be estimated.

Value

An object of type drm.

Examples

```R
# simulate a dataset
set.seed(1)
log.conc=log(1e4)-log(3)*9:0
n.replicate=2
fi=simulate1curve(p.eotoxin[1,], rep(log.conc,each=n.replicate), sd.e=0.2)
dat.std=data.frame(fi, expected_conc=exp(rep(log.conc,each=n.replicate)), analyte="Test",
assay_id="Run 1", sample_id=NA, well_role="Standard", dilution=rep(3**(9:0), each=n.replicate),
replicate=rep(1:n.replicate, 10))

fit = drm.fit(log(fi) - expected_conc, dat = dat.std)
plot(fit, log="xy")
fit
```
Description

5PL functions.

Usage

FivePL.t (t,param)
FivePL.t.func (param)
FivePL.x (x,param)
FivePL.x.inv (y,param)
FivePL.x.inv.func (param)
FivePL.t.inv (y,param)
FivePL.t.inv.func (param)
FourPL.x.inv (y, param)
FourPL.x (x, param)
FourPL.t.func (param)
cla2gh (param)
gh2cla (param)
cla2ed50 (param)
cia2ed50b (param)
ed502cla (param)
ed50b2cla (param)
get.curve.param.list (param)
simulate1curve (param, t, sd.e=0.1, expy=TRUE, gamma=0)
vpl1.deriv (x,param)
vpl1.deriv.func (param)
vpl2.deriv (x,param)
vpl2.deriv.func (param)
vpl3.deriv (x,param)
vpl3.deriv.func (param)

ED5PL (param, tao)

Arguments

param vector of numbers. Parameters of the 5PL curve.
t numeric vector. Log concentrations.
x numeric vector. Concentrations.
y numeric vector. Response.
.sd.e sd.e
tao vector of numbers or single number. Effective doses. Vectorized for this argument.
elisa.R.gh

exp y  Boolean. Controls whether to exponentiate y
gamma  power variance function parameter

Details

FivePL.t and other functions are vectorized for the x and the y arguments.

Four parameterizations are implemented. Classical: b,c,d,e,f gh: c,d,f,g,h ED50: c,d,f,b,tao(ED50)
ED50b: c,d,f,h,tao(ED50). This is called tao-h in Cumberland et al. (2014)

Author(s)

Youyi Fong <yfong@fhcrc.org>, Xuesong Yu, William N. Cumberland

Examples

FivePL.t(5:6, p.eotaxin[1,])
FivePL.t.func(p.eotaxin[1,])
FivePL.x.inv(c(4,5,11), p.eotaxin[1,])
FivePL.t.inv.func(p.eotaxin[1,])

elisa.R.gh  ELISA prior for gh Parameterization

Description

Priors.

Format

elisa.R.gh is a 5x5 diagonal matrix. rowNames/colNames: "c" "d" "g" "logh" "logf" elisa.mean.distr.gh
is a 2x5 matrix. colNames: "c" "d" "g" "logh" "logf". rowNames: mean, prec
get.abc

Compute a measure of distance between two curves.

Description

abc criterion is area between curves divided by the width of t.range. S1 criterion between two
curves is defined as the integrated squared distance divided by the width of t.range. S2 is relative
bias divided by the width of t.range.

Usage

get.abc(p1, p2, t.range)
get.S1(p1, p2, t.range)
get.S2(p1, p2, t.range)
get.abs.dev(p1, p2, t.range, y.range)

Arguments

p1 a vector of coefficients specify the first curve
p2 a vector of coefficients specify the second curve
t.range a vector two real numbers. The range of log standard concentrations. The first
one is the minimum t and the second one is the maximum t

y.range

Details

p1 and p2 can be in either classical or gh parameterization.

Value

a real number

Examples

get.abc(p.eotaxin[1,], p.eotaxin[2,], t.range=log(c(0.51,le4)))
get.S1(p.eotaxin[1,], p.eotaxin[2,], t.range=log(c(0.51,le4)))
get.S2(p.eotaxin[1,], p.eotaxin[2,], t.range=log(c(0.51,le4)))
get.abs.dev(p.eotaxin[1,], p.eotaxin[2,], t.range=log(c(0.51,le4)), y.range=c(5,6))
**getConc**

*Concentration Estimation*

**Description**

Estimate analyte concentrations based on observed outcome and a fitted curve.

**Usage**

```r
getConc(fit, y, n.replicate = 1, check.out.of.range = 1, x.range = NULL, y.range = NULL, verbose = FALSE)
```

**Arguments**

- `fit`: drc object or bcrm object
- `y`: numeric vector. Observed outcome in the samples of interest. Each element corresponds to one sample
- `n.replicate`: integer. Number of replicates that are averaged to generate y
- `check.out.of.range`: integer.
- `x.range`: vector of 2 numbers. The minimal and maximal concentration of the standard samples
- `y.range`: vector of 2 numbers. The minimal and maximal observed response of the standard samples
- `verbose`: Boolean.

**Details**

Vectorized for `y`.

---

**gnls.fit**

*Fit with gnlS function*

**Description**

Fit with gnlS function

**Usage**

```r
gnls.fit(formula, data, fit.4pl = FALSE, startVal = NULL, varFun = nlme::varPower(), verbose = FALSE)
```
**Arguments**

- `formula`
- `data`
- `fit.4pl`
- `startVal`
- `varFun`
- `verbose`

**Details**

When `startVal` is given, `varPower(value)` does not seem to change numerical values of the fit.

**hier.model.ex.2**

An example for hierarchical modeling used in Fong, Yu et al.

**Description**

An example for hierarchical modeling used in Fong, Yu et al.

**Format**

A data frame with 120 observations on the following 8 variables.

- `assay_id` a character vector. Assay identifier.
- `analyte` a character vector. Substance to be measured.
- `expected_conc` a numeric vector. The expected concentrations of a sample.
- `dilution` a numeric vector. The dilution factor of a sample.
- `fi` a numeric vector. Fluorescence intensity readout.
- `well_role` a character vector. Defines the role of a well. See ?ncal for more information.
- `sample_id` a character vector. Sample identifier.
- `replicate` a numeric vector. Technical replicate identifier.

**References**

Description

tcal fits standard curves and estimates unknown sample concentrations. rumi exists for backwards compatibility.

Usage

## S3 method for class 'formula'
ncal(formula, data, 
    bcrm.fit=FALSE, bcrm.model="t4", robust="mean", bcrm.prior="default", 
    force.fit=TRUE, fit.4pl=FALSE, return.fits=FALSE, 
    plot=TRUE, auto.layout=TRUE, plot.se.profile=TRUE, plot.log="x", plot.unknown=TRUE, 
    test.LOD=FALSE, find.LOD=FALSE, find.LQ=FALSE, grid.len=50, lod.ci=95, 
    unk.replicate=NULL, find.best.dilution=FALSE, unk.median=FALSE, 
    control.jags=list(n.iter=1e5, jags.seed=1, n.thin=NULL, 
    keep.jags.samples=FALSE, n.adapt=1e3), 
    cex=.5, additional.plot.func=NULL, check.out.of.range=1, xlab=NULL, ylab=NULL, 
    main=NULL, 
    var.model=c("constant","power"), log.both.sides=FALSE, 
    control.crm.fit=list(max.iter=20), 
    verbose=FALSE, 
    ...) 

## S3 method for class 'character'
ncal(file, is.luminex.xls, formula, bcrm.fit, verbose=FALSE, ...) 
rumi(data, ...) 
ncalGUI (verbose=FALSE) 

Arguments

formula  a formula. If the first argument is formula, bcrm.formula is called.
data  a data frame. If the first argument is dat, bcrm.data.frame is called. Each row of the data frame represents the measurement from one well/bead_type
bcrm.fit  Boolean. If TRUE, a Bayesian random effects model is fitted, else a drm model is fitted.
bcrm.model  string. Noise model used in bcrm. See error.model in the help file for bcrm for more information.
bcrm.prior  string.
var.model constant: constant variance. power: power variance function
robust string. Passed to drm.
plot a Boolean, default FALSE. This controls whether or not to make plots of the fitted curves and the standard error profiles
plot.unknown a Boolean. This controls whether or not to make plots of the fitted curves with unknown sample concentrations layered on top
auto.layout a Boolean, default TRUE. This controls whether or not to let bcrm controls the layout of the plots
test.LOD a Boolean, default TRUE. This controls whether or not to test for limits of detection
plot.se.profile a Boolean, default FALSE. This controls whether or not to make plots of standard error profile, this doubles as find.LOQ
force.fit a Boolean, default FALSE. If FALSE, return NULL when the goodness of fit is bad; otherwise, always return the fit
find.LOD Boolean. Controls whether or not to compute limits of detection.
find.LOQ Boolean. Controls whether or not to compute limits of quantification.
find.best.dilution a Boolean, default FALSE. When there are more than one dilutions for a given non-standard sample, this controls whether or not to find the best dilution
return.fits a Boolean, default FALSE. This controls whether or not to return as an attribute the curve fits
grid.len an integer, default 500. This determines the resolution of the standard error profile plot and limits of quantification
unk.replicate an integer, default NULL. This is the number of replicates for an unknown sample (at a single dilution if multiple dilutions are available)
unk.median a Boolean, default FALSE. If TRUE, median of unknown replicates are used to estimate conc, instead of mean
fit.4pl a Boolean, default FALSE
lod.ci an integer, default 95. If default, one sided 50 percent CI is used to determine LODi.
plot.log a string, default "x". Used to populate the log argument of the plot function.
control.jags list. Parameters controlling the behavior of posterior sampling by JAGS.
is.luminex.xls Boolean. Indicates whether the file is in the Luminex Excel file format outputted by the Bio-Plex software.
cex numeric. Passed to plot functions.
main plotting parameter
xlab plotting parameter
ylab plotting parameter
additional.plot.func function. Will be called after the first panel is drawn.
check.out.of.range
integer. If 1, an estimated concentration is set to half of the smallest standard concentration, or the largest standard if it is outside the range of standard concentration. If 2, do this only when the estimated concentration is 0 or Inf

verbose
a Boolean, default FALSE. This controls whether or not to print detailed messages

log.both.sides
Boolean, whether to log transform both sides of the formula

control.crm.fit
list. Parameters controlling the behavior of crm.fit

file
string. Name of the data file.

... passed to the function plotting fitted curves

Details

Certain columns are expected of the input data frame:

• bead_type Identifies a type of bead. Beads can be named after the antibody that coats the bead, or they can be named by the protein that coats the bead. This is also known as analyte or antigen in different contexts. To be retro-compatible, this column can also be named analyte

• well_role Defines the role of a well. This should be from Standard, Unknown, .... Standard wells are used to generate standard curves, and concentrations of the substance in the Unknown well will be estimated

• assay_id Identifies an assay, which is defined to be a collection of Standard and non-Standard wells. Measured fI from the Standard wells are used to create a set of standard curves, one of each bead type. Based on the standard curves and the measured fI from the non-Standard wells, concentrations of the substance in the non-Standard wells will be estimated. An assay can be a plate, if every plate has Standard wells; or it can be multiple plates run by one technician at one time, if there are only one set of Standard wells on these plates

• dilution Standard samples are often prepared by starting with one sample and doing serial dilutions. Unknown samples may be measured at several dilutions so that one of the dilutions may fall into the more reliable region of the standard curve

• replicate Index of technical replicates for a sample. Typical values are 1 or 2. May be used in plotting. Optional

• sample_id Identifies a non-Standard sample. One sample may be measured in replicates and/or in several dilutions. Should be defined meaningfully for Unknowns

If no formula is specified, we also expect

• fI Fluorescence intensity

• starting_conc Standard samples are often prepared by starting with one sample and doing serial dilutions. This column does not apply to non-Standard samples

• expected_conc Standard samples have expected concentrations. If this column is present, the dilution and starting_conc are optional and will not be used. This column does not apply to non-Standard samples
The program processes each assay separately. For each assay, each bead_type is processed separately. First fit.drc() is called to fit dose-response curves using information from the Standard wells. A plot of the fitted curve is generated. Then for each non-Standard sample, the number of dilutions is determined. For each dilution, log(f_i) from replicate wells are averaged and used to compute the estimated concentration and the standard error of the estimate.

Bad fits can happen for three reasons. 1) An error occurs in drm. 2) Estimated variance for some curve parameter is negative. 3) A goodness of fit statistics is above a pre-set threshold. When a bad fit happens, by default we plot the observed f_i, not the fitted curve, and do not proceed to estimation of concentration. However, when force.fit is set to TRUE, in the latter two cases of bad fits, we will proceed to plot the fitted curve and use it to estimate concentrations. In the second case, the standard errors of the estimated concentrations are set to NA. In the third case, standard errors will be computed. The default value for force.fit is set to FALSE to promote caution.

An important factor affecting the success of the fitting procedure is the choice of self start function. In addition to trying the default self start function in the most current drc package, we also try a self start function that is based on four parameter log-logistic model, and the self start function in version 1.5.2 of the drc package.

Standard errors convey the uncertainty we have about estimated concentrations. Standard error profiles show the relationship between standard errors and estimated concentrations. A sequence of hypothetical f_i between the expected f_i for the smallest and largest concentrations on the Standard curve are generated. The number of hypothetical f_i is controlled by the variable grid.len. For each hypothetical f_i, the estimated concentration and associated standard error are computed. There are two sources of uncertainty in an estimated concentration. One part of the uncertainty, we call replication-sensitive, comes from the fact that the measured f_i has measurement error in it. It can be reduced by doing replicates of the non-Standard samples. The number of replicates used in computing standard error profile is controlled by the variable rep.se.profile. The other part, we call replication-insensitive, comes from the fact that there are uncertainty about the Standard curve as well. The total uncertainty is plotted in black, the replication-sensitive in red, and the replication-insensitive in blue. The two parts of uncertainty add up to the total not on the standard error scale, but on the variance scale, which is the square of standard errors. We choose to plot on the standard error scale because this scale is more comparable to the estimated concentration.

When unk.replicate is NULL (default), the program sets it to the number of replicates of the first non-Standard sample it encounters; if there is no non-Standard sample, it is set to 1. This value is only used in the computation of LOD. It does not affect se of unknown sample conc estimate.

When find.best.dilution is false, the estimated concentrations for all dilutions (after adjusting for dilutions) are returned; otherwise, the best dilution, determined as the dilution having the smallest standard error, is returned.

When plot is FALSE, no plot is made. When plot is TRUE, but plot.rep.profile is false, only fitted curves are plotted.

When auto.layout is TRUE, bcrm will choose a layout that makes sense for showing one analyte per page. For example, if plot.se.profile is TRUE, it will be 2x2.

When test.LOD is true, estimated concentration will be tested against the null hypothesis that it is not different from the extremem data points of the standard samples.

The difference between Inf and NA for se: Inf is if the f_i is outside certain ranges, NA is if the s.e. is bad for some reason.

find.LOD, return an attribute "LOD", that is the lowest and highest concentration detectable, defined in the sense that ... plot.se.profile also leads to the return of an attribute "LOQ", that is the lowest
and highest concentration at which percent cv is at 20
drm requires that weights are evaluated in the global env. drm.weights (drm.fit.R) is our answer to
that.

Value
A data frame, each row contains one estimated concentration. All columns of the input data frame
are preserved, in addition two new columns are added: est.log.conc and se. est.log.conc contains
estimated log concentration, while se is the standard error of the est.log.conc.

When return.fits is TRUE, the returned data frame has an attribute "fits", that is a list of the fitted
curves.
Standard curves are plotted. When plot.se.profile is TRUE, error profiles are also plotted. Two
error profile are plotted, one is the standard error of the estimated log concentration versus esti-
mated log concentration. The other is 100 x (standard error of estimated concentration / estimated
centration).

Author(s)
Youyi Fong <yfong@fhcrc.org>, Xuesong Yu <xyu@scharp.org>.

References
Regression Analysis of Serial Dilution Assays, submitted

Examples

# begin = Sys.time()
# basic example

# simulate a dataset
set.seed(1)
log.conc=log(1e4)-log(3)*9:0
n.replicate=2
fi=simulate1curve (p.eotaxin[1,], rep(log.conc,each=n. replicate), sd.e=0.2)
dat.std=data.frame(fi, expected_conc=exp(rep(log.conc,each=n. replicate)), analyte="Test",
   assay_id="Run 1", sample_id=NA, well_role="Standard", dilution=rep(3**(9:0), each=n. replicate),
   replicate=rep(1:n. replicate, 10))
# add unknown
dat.unk=rbind(
   data.frame(fi=exp(6.75), expected_conc=NA, analyte="Test", assay_id="Run 1", sample_id=1,
      well_role="Unknown", dilution=1, replicate=1)
 , data.frame(fi=exp(6.70), expected_conc=NA, analyte="Test", assay_id="Run 1", sample_id=2,
      well_role="Unknown", dilution=1, replicate=1)
 , data.frame(fi=exp(3), expected_conc=NA, analyte="Test", assay_id="Run 1", sample_id=3,
      well_role="Unknown", dilution=1, replicate=1)
 , data.frame(fi=exp(4.4), expected_conc=NA, analyte="Test", assay_id="Run 1", sample_id=4,
      well_role="Unknown", dilution=1, replicate=1)
 , data.frame(fi=exp(4.8), expected_conc=NA, analyte="Test", assay_id="Run 1", sample_id=5,
well_role="Unknown", dilution=10, replicate=1)
)
dat=rbind(dat.std, dat.unk)

## Not run:
# to save some time

# does drm fit
out = ncal(log(fi)-expected_conc, dat, return.fits = TRUE, plot.se.profile=TRUE)
fit=attr(out, "fits")[[1]]

# does jags fit and collect 1e5 posterior samples, it may be better to set n.iter higher
out.norm = ncal(log(fi)-expected_conc, dat, bcrm.fit=TRUE, bcrm.model="norm",
    return.fits = TRUE, plot.se.profile=TRUE,
    control.jags=list(n.iter=1e4), verbose=FALSE)
fit.norm=attr(out.norm, "fits")

# compare drm fit with bcrm fit
rbind(out, out.norm)
rbind(cla2gh(coef(fit)), coef(fit.norm))
rbind(sqrt(diag(vcov(fit))), sqrt(diag(vcov(fit.norm, type="classical")) )

sd.est = c(summary(fit)$rseMat[1], getVarComponent.bcrm(fit.norm)^,.5)

# do ncal with imported data from a raw Luminex output file
# the importing step can step a litte time
out = ncal(paste(system.file(package="ncal")[[1]], '/misc/02-14A22-IgA-Biotin-tiny.xls', sep='"'),
    is.luminex.xls=TRUE, formula=log(fi)-expected_conc, bcrm.fit=FALSE, return.fits=TRUE)
fit=attr(out, "fits")[[1]]
getConc(fit, c(5,6))

## End(Not run)

#end=Sys.time();print(end-begin)

---

**p.eotaxin**

**SPL parameters from an Eotaxin dataset.**

**Description**

SPL parameters from an Eotaxin dataset.
Format

A matrix with 6 rows and 12 columns: c, d, loge, logmb, logf, b, e, f, g, h, logh, and logtao. Each row represents one curve.

plot5PL  
Plot a 5PL function

Description

plot5PL plots a 5PL function

Usage

plot5PL (param=NULL, xlim=NULL, ylim=NULL, col=NULL, lty=NULL, lwd=1, plot.legend=FALSE, add=FALSE, legend=NULL, main=NULL, xlab=NULL, ylab=NULL, xaxt="s", yaxis.log.scale=FALSE, expy=FALSE, logy=FALSE)
lines5PL (param, xlim, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>param</td>
<td>Can be a vector or a matrix. If matrix, each row represents one curve. First check for classical parameterization, then gh, then ED50 parameterization. e/b/f can be provided in log form.</td>
</tr>
<tr>
<td>xlim</td>
<td>range of t, i.e. log (concentration)</td>
</tr>
<tr>
<td>ylim</td>
<td>ylim</td>
</tr>
<tr>
<td>col</td>
<td>col</td>
</tr>
<tr>
<td>lty</td>
<td>lty</td>
</tr>
<tr>
<td>lwd</td>
<td>lwd</td>
</tr>
<tr>
<td>plot.legend</td>
<td>Boolean. Indicates whether to plot legend.</td>
</tr>
<tr>
<td>add</td>
<td>Boolean. Indicates whether to add to an existing plot or create a new plot.</td>
</tr>
<tr>
<td>legend</td>
<td>vector. Legend.</td>
</tr>
<tr>
<td>main</td>
<td>string. Title.</td>
</tr>
<tr>
<td>xlab</td>
<td>xlab</td>
</tr>
<tr>
<td>ylab</td>
<td>ylab</td>
</tr>
<tr>
<td>xaxt</td>
<td>xaxt</td>
</tr>
<tr>
<td>yaxis.log.scale</td>
<td>Boolean. Controls whether to draw y axis on the log scale</td>
</tr>
<tr>
<td>expy</td>
<td>Boolean. Controls whether to exponentiate 5PL function values</td>
</tr>
<tr>
<td>logy</td>
<td>Boolean. Controls whether to log 5PL function values</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments</td>
</tr>
</tbody>
</table>
Details

x axis is always drawn in the log scale.

Author(s)

Youyi Fong <yfong@fhcrc.org>

---

read.luminex.xls  Read a Luminex File

Description

Read a Luminex raw output .xls file

Usage

read.luminex.xls (file, verbose = FALSE, sheets = NULL, assay_id=NULL, na.strings = c("NA", "DIV/0!")...) perl = "perl")

Arguments

file  string. The file name.
verbose Boolean. If TRUE, some debug messages are printed.
sheets vector of strings. The names of the sheets to be read. If NULL, all sheets are read.
assay_id string. Used to name the assay that generated the data file. If not provided, a random number is generated as the name.
na.strings vector of strings. Strings to be mapped to NA.
... perl string. Command used to execute perl. If perl.exe is not in the path, the full path can be specified, e.g. "C:/Perl64/bin/perl.exe"

Examples

#begin=Sys.time()

# example from https://www.labkey.org/wiki/home/Documentation/page.view?name=luminexFileFormats
dat = read.luminex.xls(paste(system.file(package="nCal")[[1], '/misc/02-14A22-IgA-Biotin-tiny.xls', sep=""), verbose=TRUE)
out = ncal(log(fi)~expected_conc, dat, return.fits = TRUE, plot.se.profile=FALSE)
out

#end=Sys.time();print(end-begin)
Index

*Topic \textasciitilde kwd1
  get.abc, 12
  read.luminex.xls, 22

*Topic \textasciitilde kwd2
  get.abc, 12
  read.luminex.xls, 22

*Topic distribution
  bcrm, 2
  drm.fit, 8
  ncal, 15
  plot5PL, 21

bcrm, 2
cla2ed50 (ED5PL), 10
cal2ed50b (ED5PL), 10
cal2gh (ED5PL), 10
coef.bcrm (bcrm), 2
coef.crm (crm.fit), 6
CRM.fit, 6
dat.QL3, 8
deviance.crm (crm.fit), 6
drm.fit, 8
ed502cla (ED5PL), 10
ed50b2cla (ED5PL), 10
ED5PL, 10
elisa.mean.distr.gh (elisa.R.gh), 11
elisa.R.gh, 11
FivePL.t (ED5PL), 10
FivePL.x (ED5PL), 10
FourPL.t.func (ED5PL), 10
FourPL.x (ED5PL), 10

get.abc, 12
get.abs.dev (get.abc), 12
get.curve.param.list (ED5PL), 10
get.S1 (get.abc), 12
get.S2 (get.abc), 12
get.single.fit (bcrm), 2
getConc, 13
getVarComponent.bcrm (bcrm), 2
getVarComponent.drc (drm.fit), 8
gh2cla (ED5PL), 10
gnls.fit, 13
hier.model.ex.2, 14
lines.crm (crm.fit), 6
lines5PL (plot5PL), 21

nCal (ncal), 15
ncal, 15
ncalGUI (ncal), 15

p.eotaxin, 20
plot.bcrm (bcrm), 2
plot5PL, 21
print.bcrm (bcrm), 2

read.luminex.xls, 22
rumi (ncal), 15

simulateQcurve (ED5PL), 10

vcoev.bcrm (bcrm), 2
vpl1 deriv (ED5PL), 10
vpl2 deriv (ED5PL), 10
vpl3 deriv (ED5PL), 10