Package 'FlexParamCurve'

July 2, 2014

Type Package

Title Tools to Fit Flexible Parametric Curves

Version 1.4-3

Date 2012-09-23

Author Stephen Oswald

Maintainer Stephen Oswald <ozsao23@hotmail.com>

Description selfStart functions and model selection tools to fit parametric curves in nls, nlsList and nlme frameworks.

License GPL-2

Depends nlme

Suggests

Enhances nlme

LazyLoad yes

Repository CRAN

Date/Publication 2012-10-11 05:45:49

NeedsCompilation no

R topics documented:

FlexParamCurve-package

Tools to Fit Flexible Parametric Curves

Description

selfStart functions and model selection tools to fit parametric curves in

[nls](#), [nlsList](#) and [nlme](#) frameworks.

Details

General approach for using package (also see examples below)

1) Run *modpar* to produce initial parameter estimates and estimates of parameter bounds for your dataset.

These are used to accomodate fixed parameters and are saved in user-specified [list](#) object

All parameters and options in this list can be edited manually or using *change.pnparameters*. The

list could be created manually given that the elements were labelled sufficiently.

2) Determine most appropriate model (number of necessary parameters) for your data

using *pn.mod.compare* or *pn.modselect.step* (these rank competing model and then compare nested models using

extraF). This may take some time as many *nlsList* objects are fitted.

Note that if you perform this step, then you do not need to perform step 1.

If you are sure of your model (e.g. it is a simple logistic) Step 2 may be unnecessary.

3) Fit *nls* or *nlsList* or *nlme* models using *SSposnegRichards* specifying

the appropriate model number and the list of parameters and options. Parameter bounds can be refined to improve fits by altering

this list, either manually or using [change.pnparameters](#).

4) Plot your curves using eqnposnegRichards.eqn specifying the appropriate model number and list of parameters/options.

User level functions include:

pn.mod.compare

all-model selection for positive-negative Richards nlsList models

pn.modselect.step

backward stepwise model selection for positive-negative Richards nlsList models

SSposnegRichards

selfStart function for estimating parameters of 36 possible reductions of the 8-parameter

positive-negative Richards model (double-Richards)

posnegRichards.eqn

function for evaluating 36 possible reductions of the 8-parameter

positive-negative Richards model (double-Richards)

modpar

estimates mean parameters (and parameter bounds) for 8-parameter positive-negative Richards

models or 4-parameter Richards models and saves in objects pnmodelparams

and pnmodelparamsbounds. (required prior to use of the above functions)

change.pnparameters

simple function to update pnmodelparams and pnmodelparamsbounds

with user specified values

extraF

performs extra sum-of-squares F test for two nested nlsList models

extaF.nls

performs extra sum-of-squares F test for two nested nls models

Package:	FlexParamCurve
Type:	Package
Title:	Tools to Fit Flexible Parametric Curves
Version:	1.4-3
Date:	2012-09-23
Author:	Stephen Oswald
Maintainer:	Stephen Oswald <ozsao23@hotmail.com>
License:	GPL-2
Depends:	nlme
Enhances:	nlme
LazyLoad:	yes

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

References

Oswald, S.A. et al. (Early view) FlexParamCurve: R package for flexible

fitting of nonlinear parametric curves. Methods in Ecology and Evolution.

doi: 10.1111/j.2041-210X.2012.00231.x (see also tutorial and introductory videos at:

http://www.methodsinecologyandevolution.org/view/0/podcasts.html

(posted September 2012 - if no longer at this link, check the archived videos at:

http://www.methodsinecologyandevolution.org/view/0/VideoPodcastArchive.html#allcontent

See Also

[nlme](#)
[SSlogis](#)

Examples

```
# run all-model selection for posneg.data object (Step 2) without need to run any previous functions



data(posneg.data)

modseltable <- pn.mod.compare(posneg.data$age, posneg.data$mass,

    posneg.data$id, existing = FALSE, pn.options = "myoptions")



# run backwards stepwise model selection (Step 2) for logist.data object without need to run any previous functions



data(logist.data)

modseltable <- pn.modselect.step(logist.data$age, logist.data$mass,
```

```
      logist.data$id, existing = FALSE, pn.options = "myoptions")
```

```
# estimate fixed parameters use data object posneg.data (Step 1)
```

```
modpar(posneg.data$age,posneg.data$mass, pn.options = "myoptions")
```

```
# change fixed values of M and constrain hatching mass to 45.5 in a growth curve (Step 1)
```

```
change.pnparameters(M=1,RM=0.5,first.y=45.5, pn.options = "myoptions")
```

```
# fit nlsList object using 6 parameter model with values M and RM (Step 3)
```

```
# fixed to value in pnmodelparams and then fit nlme model
```

```
richardsR22.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

      Infl = Infl, RAsym = RAsym, Rk = Rk, Ri = Ri,

      modno = 22, pn.options = "myoptions"), data = posneg.data)

 richardsR22.nlme <- nlme(richardsR22.lis, random = pdDiag(Asym + Infl ~ 1))
```

```
# fit reduced nlsList model and then compare performance with extraF (manual version of Step 2)



richardsR20.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

    Infl = Infl, modno = 20, pn.options = "myoptions"), data = posneg.data)

 extraF(richardsR20.lis,richardsR22.lis)



 # fit and plot a logistic curve (M=1) to data, note that all parameters set to equal 1 here are ignored

 # note code here forces \eqn{modpar} to only estimate 4 curve parameters (simple Richards curve)



data(logist.data)

modpar(logist.data$age,logist.data$mass,force4par=TRUE, pn.options = "myoptions")#create list for fixed paramete

change.pnparameters(M=1) # set M to 1 for subsequent fit

richardsR20.nls <- nls(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

    Infl = Infl, modno = 20, pn.options = "myoptions"), data = logist.data)

plot(logist.data$age , logist.data$mass, xlab = "age", ylab = "mass", pch = ".", cex = 0.7)

par <- coef( richardsR20.nls )

#(Step 4)
```

```
curve(posnegRichards.eqn(x, Asym = par[1], K = par[2], Infl = par[3], modno = 20), add= TRUE, pn.options = "myopti
```

---

change.pnparameters   *Change Fixed Parameter Values*

Description

Function to alter values of parameters to be used by SSposnegRichards

or posnegRichards.eqn

as the fixed values in equations where parameters are fixed

Usage

```
change.pnparameters(Asym = NA,

K = NA,

Infl = NA,

M = NA,

RAsym = NA,

Rk = NA,

Ri = NA,

RM = NA,

first.y = NA,

x.at.first.y = NA,
```

```
    last.y = NA,

    x.at.last.y = NA,

    twocomponent.x = NA,

    verbose = NA,

    force4par = NA,

    pn.options)
```

Arguments

Asym	a numeric value for the asymptote of the positive (increasing) curve
K	a numeric value for the rate parameter of the positive (increasing) curve
Infl	a numeric value for the point of inflection of the positive (increasing) curve
M	a numeric value for the shape parameter of the positive (increasing) curve
RAsym	a numeric value for the asymptote of the negative (decreasing) curve
Rk	a numeric value for the rate parameter of the negative (decreasing) curve
Ri	a numeric value for the point of inflection of the negative (decreasing) curve
RM	a numeric value for the shape parameter of the negative (decreasing) curve
first.y	the value of y at minimum x when it is required to be constrained
x.at.first.y	the final value of x - 0 value is used if not specified when last.y is not NA
last.y	the value of y at maximum x when it is required to be constrained
x.at.last.y	the final value of x - this is option is currently disabled
twocomponent.x	a numerical specifying the x-value (e.g. age) of intersection if a double model of two separate components is to be fitted. Alternatively a logical of value = TRUE if the same type of model is to be fitted but the x of intersection is unknown
verbose	logical indicating whether information on successful optimization and parameters should be returned during when using SSposnegRichards
force4par	logical specifying whether parameters of the negative Richards should be ignored - effectively using simple Richards curve
pn.options	required character string specifying the name of a list object currently populated with starting parameter estimates, fitting options and bounds to be modified

Details

This function provides a simple way for the user to update

the a user-named list that holds fixed values and options

for fitting and solving positive-negative Richards curves with

[SSposnegRichards](#) and [posnegRichards.eqn](#),

respectively. Running this function also concurrently updates the parameterbounds

in the same list which are vthe maximum and minimum values

for parameters to be used by [optim](#) and [nls](#)

during parameter estimation

in [SSposnegRichards](#). The list is written automatically by the function but

it is also output as a return value for assignation in the usual way [myoptions<- change.pnparameters(...)].

The list specified by pn.options must exist before this function is called. Use [modpar](#)

to estimate values for all parameters and easily generate a suitable list. See [modpar](#) for details of bounding.

Value

a [list](#) of values for all above arguments,

with new values substituted where specified in the call

Note

Requires [modpar](#) to be have been run prior to execution

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

See Also

[modpar](#) [SSposnegRichards](#) [posnegRichards.eqn](#)

Examples

```
# change all fixed values except K and Rk

    data(posneg.data)


    modpar(posneg.data$age, posneg.data$mass, pn.options = "myoptions")


    change.pnparameters(Asym = 10000, Infl = 80, M = 5, RAsym = 10000,
```

```
        Ri = 240, RM = 5, pn.options = "myoptions")
```

```
# change fixed values of M and constrain hatching mass to 45.5 in a growth curve
```

```
    change.pnparameters(M = 1, RM = 0.5, first.y = 45.5, pn.options = "myoptions")
```

---

extraF   *Compare Two $nlsList$ Models Using Extra Sum-of-Squares F-Tests*

Description

Function to compare two nested models using extra sum-of-squares F-Tests.

Usage

```
extraF(submodel = 1,

genmodel = 1)
```

Arguments

submodel	$nlsList$ model with fewer curve parameters (reduced model)
genmodel	$nlsList$ model with more curve parameters (general model)

Details

Models must be entered in the correct order with the reduced model appearing
first in the call and the more general model appearing later. These must be nested models,
i.e. the general model must contain all of the curve parameters in the reduced model and more.
Entering models with the same number of parameters will produce NAs in the output, but
the function will produce seemingly adequate output with non-nested models. The user must
check that models are nested prior to use.

This function is primarily designed to be called by the model selection functions

`pn.modselect.step` and `pn.mod.compare` but can be used independently.

Extra sum-of-squares is obtained from:

`F = (SS1 - SS2)/(df1 - df2) / (SS2 / df2)`

where SS = sum-of-squares and df = degrees of freedom, for the more reduced model (1) and the

more general model (2), respectively.

If the F value is significant then the more general model provides a significant improvement

over the reduced model, but if the models are not significantly different then the reduced

parameter model is to be preferred.

In extraF (formulated especially for nlsList models), the root mean square error

(and sum of squares) is inflated to the value expected if all groups (levels) were

fitted [i.e. RSE = RSE* (sqrt(n1) / sqrt(n0)), where RSE is root mean square error, n0 is the sample size (total

number of data points used in fit) for the model with missing levels, and n1 is the inflated sample size (total number

of data points in dataset)]. This is based on RSE changing with the square root of sample size, as discussed in the help file for

`pn.mod.compare`. Degrees of freedom are then increased to the vaue if all individuals had been fitted successfully,

i.e. total df - (# curve parameters * # levels). Thus, RSE and df are enlarged for models with missing levels so all models are

compared based on the variability expected if all levels had been fitted . This allows the Fstat from extraF to be independent of missing levels

in either of the two models.

Value

A `data.frame` listing the names of the models compared, F,

numerator degrees of freedom,

demonimator degrees of freedom, P value and the residual sum of squares for both the general

and reduced models

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

References

Ritz, C. and Streibigg, J. C. (2008) $Nonlinear regression with R.$
Springer-Verlag, New York.

See Also

extraF.nls

nlsList

pn.modselect.step

pn.mod.compare

Examples

```
#compare two nested nlsList models (4 vs 8 parameter models)




data(posneg.data)


modpar(posneg.data$age, posneg.data$mass, pn.options.name = "myoptions")


# (only first 4 group levels in data used for example's sake)


subdata<-subset(posneg.data, as.numeric(row.names (posneg.data) ) < 53)


richardsR2.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

 Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri,modno = 2, pn.options = myoptions)

                   , data = subdata)


richardsR12.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

  Infl = Infl, M = M, modno = 12, pn.options = myoptions)

                   , data = subdata)


extraF(richardsR12.lis, richardsR2.lis)
```

---

extraF.nls   *Compare Two nls Models Using Extra Sum-of-Squares F-Tests*

Description

Function to compare two nested [nls](#) models using extra
sum-of-squares F-Tests.

Usage

```
extraF.nls(submodel,

genmodel)
```

Arguments

submodel	nls model with fewer curve parameters (reduced model)
genmodel	nls model with more curve parameters (general model)

Details

Models must be entered in the correct order with the reduced model appearing
first in the call and the more general model appearing later. These must be nested models,
i.e. the general model must contain all of the curve parameters in the reduced model and more.
Entering models with the same number of parameters will produce NAs in the output, but
the function will produce seemingly adequate output with non-nested models. The user must
check that models are nested prior to use.

This function is not promoted for use in model selection as differences in curves of
different grouping levels in the dataset may be obscured when curves are fitted to the
entire dataset, as in [nls](#).

Extra sum-of-squares is obtained from:

```
F = (SS1 - SS2)/(df1 - df2) / (SS2 / df2)
```

where SS = sum-of-squares and df = degrees of freedom, for the more reduced model (1) and the
more general model (2), respectively. To account for missing individuals for different fits
df are scaled in all models to the value they would be if all individuals fit successfully (note
that if all individuals had the same fit, this would not influence extra sum of squares).

If the F value is significant then the more general model provides a significant improvement over the reduced model, but if the models are not significantly different then the reduced parameter model is to be preferred.

Value

A [data.frame](#) listing the names of the models compared, F,

numerator degrees of freedom,

demonimator degrees of freedom, P value and the residual sum of squares for both the general and reduced models

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

References

Ritz, C. and Streibigg, J. C. (2008) $Nonlinear regression with R.$ Springer-Verlag, New York.

See Also

[extraF](#)

[nls](#)

[pn.modselect.step](#)

[pn.mod.compare](#)

Examples

```
#fit and compare two nested nls models (7 vs 8 parameter models)



   data(posneg.data)


  modpar(posneg.data$age, posneg.data$mass, pn.options.name = "myoptions") #create list for fixed parameters


   richardsR1.nls <- nls(mass ~ SSposnegRichards(age, Asym = Asym, K = K,


  Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri, RM = RM, modno = 1, pn.options = myoptions)
```

```
                        , data = posneg.data)


    richardsR2.nls <- nls(mass ~ SSposnegRichards(age, Asym = Asym, K = K,


  Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri, modno = 2, pn.options = myoptions)


                        , data = posneg.data)


    extraF.nls(richardsR2.nls, richardsR1.nls)
```

---

logist.data	*Simulated growth of whiskered terns*

Description

The `logist.data` data frame has 1100 rows and 3 columns of records of the simulated masses for whiskered tern chicks between 0 and 21 days of age.

Usage

```
data(logist.data)
```

Format

This object of class c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame") containing the following columns:

mass a numeric vector of chick masses (g).

age a numeric vector of chick ages (days).

id an ordered factor indicating unique id of each simulated individual, i.e. which data belongs to which individual.

Details

No published parameter estimates with associated variability are available for positive-negative growth curves. These data were simulated using an 3-parameter positive-negative Richards curve (`SSposnegRichards` (model 20)), using parameters drawn from normal distributions with the following means (standard deviations):

```
  Asym=92.35 (15.65)
  K=0.06 (0.138)
  Infl=0.294 (1.72)
```

These values were taken from Pallisson et al. (2008) for 75 chicks reported. Each simulated individual had 11 measurements stratified through the development period, with 1-2 day random differences in timing of each measurement. This data object has methods for nlme grouped-data classes.

Source

Paillisson, J.-M., Latraube, F. & Reeber, S. (2008) Assessing growth and age of Whiskered Tern *Chlidoniashybrida* chicks using biometrics. Ardea, 96, 271-277.

Examples

```
require(stats); require(graphics)
data(logist.data)

modpar(logist.data$age, logist.data$mass, force4par = TRUE, pn.options = "myoptions") #create list for fixed par
plot(mass ~ age, data = logist.data, subset = id == "0.002",
     xlab = "Chick age (day)", las = 1,
     ylab = "Chick mass (g)",
     main = "logist.data and fitted curve (Chick #2 only)")
change.pnparameters(M=1, pn.options = "myoptions") # set curve to logistic (M=1) in subsequent fit
fm1 <- nls(mass ~ SSposnegRichards(age,Asym=Asym,K=K,Infl=Infl,
           modno=20, pn.options = "myoptions"),
           data = logist.data, subset = id == "0.002")
age <- seq(0, 166, length.out = 101)
lines(age, predict(fm1, list(age = age)))
```

---

modpar   *Estimate Values to be Used for Fixed FlexParamCurve Parameters*

Description

This function creates the object pnmodelparams

which holds estimates of values for all 8 FlexParamCurve

parameters used for fitting and solving positive-negative Richards curves with

SSposnegRichards and posnegRichards.eqn,

respectively.

Usage

```
modpar(x,

y,
```

```
pn.options = NA,

first.y = NA,

x.at.first.y = NA,

last.y = NA,

x.at.last.y = NA,

twocomponent.x = NA,

verbose = FALSE,

force8par = FALSE,

force4par = FALSE,

suppress.text = FALSE,

taper.ends = 0.45,

width.bounds = 1,

bounds.error = FALSE,

...)
```

Arguments

x	a numeric vector of primary predictor variable

y	a numeric vector of response variable
first.y	the value of y at minimum x when it is required to be constrained
x.at.first.y	the final value of x - 0 value is used if not specified when last.y is not NA
last.y	the value of y at maximum x when it is required to be constrained
x.at.last.y	the final value of x - must be specified if last.y is not NA
twocomponent.x	a numerical specifying the x-value (e.g. age) of intersection if a double model of
	two separate components is to be fitted. Alternatively a logical of value
	= TRUE if the same type of model is to be fitted but the x of
	intersection is unknown
verbose	logical indicating whether information on successful optimization and
	parameters should be returned during when using SSposnegRichards
force8par	logical specifying whether parameters of the negative Richards
	curve should be set to defaults if they cannot be estimated
force4par	logical specifying whether parameters of the negative Richards
	should be ignored - effectively using simple Richards curve
pn.options	character string specifying name of [list](#) object populated with starting
	parameter estimates, fitting options and bounds or the destination for modpar to write a new list
suppress.text	logical specifying whether modpar should return descriptive text to the screen during execution
taper.ends	numeric representing the proportion of the range of the x variable for which data are extended at
	the two ends of the data set. This is used in initial estimation (prior to optim and nls optimizations) and can
	speed up subsequent optimizations by imposing a more pronounced S-shape to both first and second curves. Defaults to 0.45.
width.bounds	a numeric indicating the proportion of the usual width of parameter bounds to be imposed during optimizations.
	Large values may slow or terminate computations, however they could better accomodate data in which different levels exhibit very different
	parameter values.
bounds.error	a logical. If true parameter estimation will terminate if initial estimation of parameters leads to
	values outside specified bounds in pn.options. If false, more appropriate bounds will be determined automatically.
...	additional optional arguments to be passed to nlsList

Details

This function creates a formatted [list](#) object as named by the argument pn.options. This list

holds estimates of values for all 8 FlexParamCurve parameters, fitting options and parameter bounds used for

fitting and solving double-Richards curves with `SSposnegRichards` and `posnegRichards.eqn`,

respectively. Parameter bounds are the maximum and minimum parameters values that can be used by `optim`

and `nls` during parameter estimation. For definitions of parameters see either `SSposnegRichards`

or `posnegRichards.eqn`. The list is written automatically by the function but

it is also output as a return value for assignation in the usual way [myoptions<- change.pnparameters(...)].

Estimates are produced by fitting positive-negative or double Richards curves in

`nls` using

`SSposnegRichards` for the full 8 parameter model (R1).

If this fails, the function `getInitial` is called to

attempt to produce initial estimates using the same 8 parameter model.

If this also fails, estimates are attempted in the same way using the

4 parameter (positive only) model (R12). In this case, only the positive

parameters are returned (NAs are substituted for negative parameters)

unless argument force8par=TRUE, in which case negative parameters are

defaulted to: RAsym = 0.05*Asym, Rk = K, Ri = Infl, RM = M.

This function can now fit biphasic (and more generally

double-Richards) curves, where the final curve is effectively either two positive curves

or two negative curves, as well as negative-positive curves. This functionality is default

and does not need to be specified.

Parameter bounds estimated here for use in `optim` and `nls`

fits within `SSposnegRichards` are

applicable to a wide range of curves, although user may

change these manually in `list` object specified by pn.options.

Bounds are estimated by `modpar` by adding or subtracting multiples

of fixed parameter values to estimated mean parameter values:

-Asym*0.5 and +Asym*2.5,

-K*0.5 and +K*0.5,

-Infl*2.5 and +Infl*10

-M*2 and +M*2

-RAsym*0.5 and +RAsym*2.5,

-Rk*0.5 and +Rk*0.5,

-Ri*2.5 and +Ri*5

-RM*2 and +RM*2.

Use force8par = TRUE if initial call to `modpar` produces estimates for

only 4 parameters and yet an 8 parameter model is desired for `SSposnegRichards`

or `posnegRichards.eqn`.

Use force4par = TRUE if you desire to produces estimates only for the four parameters of

a single Richards curves. This should also be used if you wish to fit simple logistic

Gompertz or von Bertalanffy curves: see [SSposnegRichards](#) for more details.

When specified, first.y and last.y are saved in list object specified by pn.options to instruct

[SSposnegRichards](#) to add this as the first or last value of the response, respectively,

during estimation.

To fit two-component double-curves, in which one curve equation is used up to (and including)

the x of intersection and a separate equation is used for x-values greater than the x of intersection

the argument twocomponent.x should be set to the value for the x of intersection. If this argument

is anything other than NA then a two-component model will be fitted when [SSposnegRichards](#)

is called. This option will be saved in list object specified by pn.options and can be changed at will.

taper.ends can be used to speed up optimization as it extends the dataset at maximum and minimum extremes

of x by repeatedly pasting the y values at these extremes for a specified proportion of the range of x.

taper.ends is a numeric value representing the proportion of the range of x values are extended for and

defaults to 0.45 (45

tend towards a zero slope this is a suitable values. If tapered ends are not desirable then choose taper.ends = 0.

If the argument verbose = TRUE then details concerning the optimization processes within

[SSposnegRichards](#) are printed on screen whenever [SSposnegRichards](#) is called.

These include whether optimization of the first or second parts of the curve or simultaneous optimizations

are successful, if these have been further refined by nls, whether default parameters were used or the

parameterization was aborted and what parameter values were finally exported by [SSposnegRichards](#).

This option will be saved in the list object specified by pn.options and can be changed at will.

Value

a [list](#) of estimated fixed values for all

above arguments

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

Examples

```
# estimate fixed parameters use data object posneg.data

    data(posneg.data)

    modpar(posneg.data$age, posneg.data$mass, pn.options = "myoptions")




# estimate fixed parameters use data object posneg.data (only first

# 4 group levels for example's sake) and specify a fixed hatching

# mass for curve optimization using \code{\link{SSposnegRichards}}



 modpar(posneg.data$age, posneg.data$mass, pn.options = "myoptions")

    subdata <- subset(posneg.data,posneg.data$id == as.character(36)

    | posneg.data$id == as.character(9)

    | posneg.data$id == as.character(32)

    | posneg.data$id == as.character(43))

 richardsR22.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym,

      K = K, Infl = Infl, RAsym = RAsym, Rk = Rk, Ri = Ri,

       modno = 22, pn.options = "myoptions"), data = subdata)
```

```
# fit a two component model - enter your own data in place of "mydata"
```

```
## Not run:
```

```
 modpar(mydata$x,mydata$y,twocomponent.x=TRUE, pn.options = "myoptions") # if x of intersection unknown
```

```
 modpar(mydata$x,mydata$y,twocomponent.x=75, pn.options = "myoptions") # if x of intersection = 75
```

```
 richardsR1.nls <- nls(y~ SSposnegRichards(x, Asym = Asym, K = K,
```

```
  Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri, RM = RM, modno = 1, pn.options = "myoptions")
```

```
                     , data = mydata)
```

```
## End(Not run)
```

```
# force an 8 parameter estimate on logistic data
```

```
data(logist.data)
```

```
modpar(logist.data$age,logist.data$mass,force8par=TRUE, pn.options = "myoptions")
```

```
# force an 4 parameter model on logistic data
```

```
data(logist.data)

modpar(logist.data$age,logist.data$mass,force4par=TRUE, pn.options = "myoptions")
```

```
# troubleshoot the fit of a model

data(posneg.data)

modpar(posneg.data$age,posneg.data$mass,verbose=TRUE, pn.options = "myoptions")
```

---

penguin.data   *Field data on growth of little penguins Eudyptulaminor*

Description

The `penguin.data` data frame has 2244 rows and 11 columns of records of the measured masses
for little penguin chicks between 13 and 74 days of age collected at Philip Island, Victoria, Australia
in 2000 and 2002 (see Chiaradia & Nisbet 2006).

Usage

```
data(penguin.data)
```

Format

This object of class `c("nfnGroupedData", "nfGroupedData",    "groupedData", "data.frame")`
containing the following columns:

site Three character factor for the site (only one site in dataset).

year A factor specifying the year of measurement.

bandid an ordered factor indicating unique id of each individual: the union of the laying date of the nest relative to the colony and the band combination

siteyear A factor specifying levels of year for different sites (only one site in dataset).

weight a numeric vector of chick masses (g).

ckage a numeric vector of chick ages (days).

Jdate a numeric vector of first egg-laying date of the nest(days), relative to the mean laying date for all nests in that year.

nest A factor of unique codes that identify each nest.

ck A factor of hatching order for each chick (A = first hatched, B = second hatched).

outcome A factor of codes for fate of each chick (F = fledged; only fledged chicks included).

clutch A factor of size of clutch/brood that each chick comes from (either 1- or 2-chick brood).

Details

Data were collected as outlined in Chiaradia & Nisbet (2006). Penguin chicks are generally considered to exhibit a double-Gompertz growth form.Please contact Andre Chiaradia (a.chiaradia@penguins.org.au) for use in collaborations.

Source

Chiaradia, A. & Nisbet, I.C.T. (2006) Plasticity in parental provisioning and chick growth in Little Penguins $Eudyptula minor$ in years of high and low breeding success. Ardea, 94, 257-270.

Examples

```
require(stats); require(graphics)
data(penguin.data)

modpar(penguin.data$ckage, penguin.data$weight, pn.options = "myoptions")
plot(weight ~ ckage, data = penguin.data, subset = bandid == penguin.data$bandid[1],
     xlab = "Chick age (day)", las = 1,
     ylab = "Chick mass (g)",
     main = "penguin.data and fitted curve (Chick #307 only)")
fm1 <- nls(weight ~ SSposnegRichards(ckage,Asym=Asym,K=K,Infl=Infl, RAsym=RAsym,modno=31, pn.options= "myoptions
           data = penguin.data, subset = bandid == penguin.data$bandid[1])
ckage <- seq(0, 74, length.out = 101)
lines(ckage, predict(fm1, list(ckage = ckage)))
```

---

pn.mod.compare   *Compare All Possible Positive-Negative Richards* $nlslist$ *Models*

Description

This function performs model selection for nlsList models fitted using SSposnegRichards.

Usage

```
pn.mod.compare(x,

y,

grp,

pn.options,

forcemod = 0,

existing = FALSE,

penaliz = "1/sqrt(n)",

taper.ends = 0.45,

mod.subset = c(NA),

...)
```

Arguments

x	a numeric vector of the primary predictor
y	a numeric vector of the response variable
grp	a factor of same length as x and y that distinguishes groups within the dataset
pn.options	required character string specifying name of [list](#) object populated with starting parameter estimates, fitting options and bounds
forcemod	optional numeric value to constrain model selection (see Details)
existing	optional logical value specifying whether some of the relevant models have already been fitted
penaliz	optional character value to determine how models are ranked (see Details)

taper.ends	numeric representing the proportion of the range of the x variable for which data are extended at
	the two ends of the data set. This is used in initial estimation (prior to optim and nls optimizations) and can
	speed up subsequent optimizations by imposing a more pronounced S-shape to both first and second curves. Defaults to 0.45.
mod.subset	optional vector containing modno of models that the user desires to be estimated. If not NA, only
	[nlsList](#) models in mod.subset will be fitted and ranked
...	additional optional arguments to be passed to nlsList

Details

First, whether parameter M should be fixed

(see [SSposnegRichards](#)) is determined by fitting models 12 and 20 and comparing

their perfomance using [extraF](#). Note that model 20 is identical to model 32.

If model 12 provides superior performance (variable values of M) then 16 models that estimate M are run

(models 1 through 16), otherwise the models with fixed M are fitted (models 21 through 36).

Fitting these [nlsList](#) models can be time-consuming (2-4 hours using the dataset

[posneg.data](#) that encompasses 100 individuals) and if several of the relevant

models are already fitted the option existing=TRUE can be used to avoid refitting models that

already exist globally (note that a model object in which no grouping levels were successfully

parameterized will be refitted, as will objects that are not of class nlsList).

Specifying forcemod=3 will force model selection to only consider fixed M models and setting

forcemod=4 will force model selection to consider models with varying values of M only.

If fitting both models

12 and 20 fails, fixed M models will be used by default.

taper.ends can be used to speed up optimization as it extends the dataset at maximum and minimum extremes

of x by repeatedly pasting the y values at these extremes for a specified proportion of the range of x.

taper.ends is a numeric value representing the proportion of the range of x values are extended for and

defaults to 0.45 (45

tend towards a zero slope this is a suitable values. If tapered ends are not desirable then choose taper.ends = 0.

Models are ranked by modified pooled residual square error. By default residual standard error

is divided by the square root of sample size. This exponentially penalizes models for which very few

grouping levels (individuals) are successfully parameterized (the few individuals that are

parameterized in these models are fit unsuprisingly well) using a function based on the relationship

between standard error and sample size. However, different users may have different preferences

and these can be specified in the argument penaliz (which residual

standard error is multiplied by). This argument must be a character value

that contains the character n (sample size) and must be a valid right hand side (RHS) of a formula:

e.g. 1*(n), (n)^2. It cannot contain more than one n but could be a custom function, e.g. FUN(n).

Value

A list object with two components: $'Model rank table' contains the

statistics from [extraF](#) ranked by the modified residual standard error,

and $'P values from pairwise extraF comparison' is a matrix of P values from

[extraF](#) for legitimate comparisons (nested and successfully fitted models).

The naming convention for models is a concatenation of 'richardsR', the modno and '.lis'

which is shortened in the matrix output, where the number of parameters has been

pasted in parentheses to allow users to easily distinguish the more general model from

the more reduced model

(see [extraF](#) and [SSposnegRichards](#)).

For extra flexibility, mod.subset can specify a vector of modno values (a number of different models) that

can be fitted in [nlsList](#) and then evaluated by model selection. This prevents fitting of unwanted models or

attempts to fit models that are known to fail. If the [nlsList](#) model already exists it will not be refitted

and thus existing models can be included in the ranking table without adding noticeably to processing time.

Note

If appropriate bounds (or starting parameters) are not available in the list specified by the variable supplied

to pn.options, [modpar](#) will be called automatically prior to model selection.

During selection, text is output to the screen to inform the user of the progress of model selection

(which model is being fitted, which were fit successfully)

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

See Also

[extraF](#)
[SSposnegRichards](#)
[nlsList](#)

Examples

```
#these examples will take a long while to run as they have to complete the 32 model comparison

#run model selection for posneg.data object (only first 3 group levels for example's sake)



   data(posneg.data)

   subdata <- subset(posneg.data, as.numeric(row.names (posneg.data) ) < 40)

   modseltable <- pn.mod.compare(subdata$age, subdata$mass,

      subdata$id, existing = FALSE, pn.options = "myoptions")



#fit nlsList model initially and then run model selection

#for posneg.data object when at least one model is already fit

# note forcemod is set to 3 so that models 21-36 are evaluated



    richardsR22.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

      Infl = Infl, RAsym = RAsym, Rk = Rk, Ri = Ri , modno = 22)

                      ,data = posneg.data, pn.options = "myoptions")

   modseltable <- pn.mod.compare(subdata$age, subdata$mass,

      subdata$id, forcemod = 3, existing = TRUE, pn.options = "myoptions")
```

```
#run model selection ranked by residual standard error*sample size
```

```
modseltable <- pn.mod.compare(subdata$age, subdata$mass,

  subdata$id, penaliz='1*(n)', existing = TRUE, pn.options = "myoptions")
```

---

pn.modselect.step	*Backwards Stepwise Selection of Positive-Negative Richards* $nlslist$ *Models*

Description

This function performs backawards stepwise model selection for [nlsList](#)
models fitted using
[SSposnegRichards](#).

Usage

```
pn.modselect.step(x,

y,

grp,

pn.options,

forcemod = 0,

existing = FALSE,
```

```
penaliz = "1/sqrt(n)",

taper.ends = 0.45,

...)
```

Arguments

x	a numeric vector of the primary predictor
y	a numeric vector of the response variable
grp	a factor of same length as x and y that distinguishes groups within the dataset
pn.options	required character string specifying name of [list](#) object populated with starting parameter estimates, fitting options and bounds
forcemod	optional numeric value to constrain model selection (see Details)
existing	optional logical value specifying whether some of the relevant models have already been fitted
penaliz	optional character value to determine how models are ranked (see Details)
taper.ends	numeric representing the proportion of the range of the x variable for which data are extended at
	the two ends of the data set. This is used in initial estimation (prior to optim and nls optimizations) and can
	speed up subsequent optimizations by imposing a more pronounced S-shape to both first and second curves. Defaults to 0.45.
...	additional optional arguments to be passed to nlsList

Details

First, whether parameter M should be fixed

(see [SSposnegRichards](#)) is determined by fitting models 12 and 32 and comparing

their perfomance using [extraF](#).

If model 12 provides superior performance (variable values of M) then 16 models that estimate M

are run (models 1 through 16), otherwise the models with fixed M are fitted (models 21 through 36).

Model selection then proceeds by fitting the most general model (8-parameter, model 1 for variable M;

7-parameter, model 21 for fixed M). At each subsequent step reduced models are evaluated

by creating `nlsList` models through removal of a single parameter from the decreasing

section of the curve (i.e. RAsym, Rk, Ri or RM). This is repeated until all possible models with

one less parameter have been fitted and then these models are then ranked by modified pooled residual

standard error (see below) to determine which reduced parameter model provides the best fit. This ranking

esnures that in all cases subsequent extra sum-of-squares F-tests are only made between fully nested models.

The best ranked reduced parameter model is then compared with the more general model retained from the

the previous step using the function `extraF` to determine whether the more general

model provides significant improvement over the best reduced model. The most appropriate model

is then retained to be used as the general model in the next step. This process continues

for up to six steps (all steps will be attempted even if the general model provides better

performance to allow for much more reduced models to also be evaluated). The most reduced model

possible to evaluate in this function contains only parameters for the positive section of the curve

(4-parameters for variable M, 3-parameters for fixed M).

Fitting these `nlsList` models can be time-consuming (2-4 hours using the dataset

`posneg.data` that encompasses 100 individuals) and if several of the relevant

models are already fitted the option existing=TRUE can be used to avoid refitting models that

already exist globally (note that a model object in which no grouping levels were successfully

parameterized will be refitted, as will objects that are not of class `nlsList`).

Specifying forcemod=3 will force model selection to only consider fixed M models and setting

forcemod=4 will force model selection to consider models with varying values of M only.

If fitting both models 12 and 32 fails, fixed M models will be used by default.

taper.ends can be used to speed up optimization as it extends the dataset at maximum and minimum extremes

of x by repeatedly pasting the y values at these extremes for a specified proportion of the range of x.

taper.ends is a numeric value representing the proportion of the range of x values are extended for and

defaults to 0.45 (45

tend towards a zero slope this is a suitable values. If tapered ends are not desirable then choose taper.ends = 0.

Competing non-nested models are ranked by modified pooled residual square error. By default this is residual

standard error divided by the square root of sample size. This exponentially penalizes models for which very few

grouping levels (individuals) are successfully parameterized (the few individuals that are

parameterized in these models are fit unsuprisingly well) using a function based on the relationship
between standard error and sample size. However, different users may have different preferences
and these can be specified in the argument penaliz (which residual
standard error is multiplied by). This argument must be a character value
that contains the character n (sample size) and must be a valid right hand side (RHS) of a formula:
e.g. 1*(n), (n)^2. It cannot contain more than one n but could be a custom function, e.g. FUN(n).

Value

A `data.frame` containing statistics produced by `extraF`
evaluations at each step, detailing the name of the general and best reduced model at each
step. The overall best model evaluated by the end of the function is saved globally as
pn.bestmodel.lis
The naming convention for models is a concatenation of 'richardsR', the modno and '.lis'
(see `SSposnegRichards`).

Note

If appropriate bounds (or starting parameters) are not available in the list specified by the variable
supplied
to `pn.options`, `modpar` will be called automatically prior to model selection.
During selection, text is output to the screen to inform the user of the progress of model selection
(which model is being fitted)

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

See Also

`pn.mod.compare`
`extraF`
`SSposnegRichards`
`nlsList`

Examples

```
#these examples will take a long while to run as they have to complete the 32 model comparison


#run model selection for posneg.data object (only first 3 group levels for example's sake)
```

```
data(posneg.data)

   subdata<-subset(posneg.data, as.numeric(row.names (posneg.data) ) < 40)

   modseltable <- pn.modselect.step(subdata$age, subdata$mass,

      subdata$id, existing = FALSE, pn.options = "myoptions")



#fit nlsList model initially and then run model selection

#for posneg.data object when at least one model is already fit

#note forcemod is set to 3 so that models 21-36 are evaluated

#(only first 4 group levels for example's sake)



subdata<-subset(posneg.data, as.numeric(row.names (posneg.data) ) < 40)

   richardsR22.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym, K = K,

      Infl = Infl, RAsym = RAsym, Rk = Rk, Ri = Ri , modno = 22, pn.options = "myoptions")

                     ,data = subdata)

   modseltable <- pn.modselect.step(subdata$age, subdata$mass,

      subdata$id, forcemod = 3, existing = TRUE, pn.options = "myoptions")
```

```
#run model selection ranked by residual standard error*sample size

#(only first 4 group levels for example's sake)



modseltable <- pn.modselect.step(subdata$age, subdata$mass,

    subdata$id, penaliz='1*(n)', existing = TRUE, pn.options = "myoptions")
```

---

posneg.data   *Simulated growth of black-browed albatrosses*

Description

The posneg.data data frame has 1300 rows and 3 columns of records of the simulated masses for black-browed albatross chicks between 0 and 166 days of age.

Usage

```
data(posneg.data)
```

Format

This object of class c("nfnGroupedData", "nfGroupedData",   "groupedData", "data.frame") containing the following columns:

mass a numeric vector of chick masses (g).

age a numeric vector of chick ages (days).

id an ordered factor indicating unique id of each simulated individual, i.e. which data belongs to which individual.

Details

No published parameter estimates with associated variability are available for positive-negative growth curves. These data were simulated using an 8-parameter positive-negative Richards curve (SSposnegRichards (model 1)), using parameters drawn from normal distributions with the following means (standard deviations):

```
Asym=4300 (180)
K=0.06 (0.01)
Infl=23 (0.4)
M=0.1 (0.05)
RAsym=1433.3 (540) #1/3 of Asym, more variable
Rk=0.108 (0.03) #1.8 times faster recession, more variable
Ri=Infl+87.259 (1.7) # more variable but linked to Infl
RM=M (0.15) #more variable
```

These values were chosen through comparison of growth curves with Huin and Prince (2000) Fig 2 and variability observed between individual chicks of little penguins in a 10 year dataset (Chiaradia and Nisbet unpublished data). Each simulated individual had 13 measurements stratified through the development period, with 1-13 day random differences in timing of each measurement. This data object has methods for [nlme](#) grouped-data classes.

Source

Huin, N. & Prince, P.A. (2000) Chick growth in albatrosses: curve fitting with a twist. Journal of Avian Biology, 31, 418-425.

Examples

```
require(stats); require(graphics)
data(posneg.data)

modpar(posneg.data$age, posneg.data$mass, pn.options = "myoptions") #create list for fixed parameters
plot(mass ~ age, data = posneg.data, subset = id == "1",
     xlab = "Chick age (day)", las = 1,
     ylab = "Chick mass (g)",
     main = "posneg.data data and fitted curve (Chick #1 only)")
fm1 <- nls(mass ~ SSposnegRichards(age,Asym=Asym,K=K,Infl=Infl, RAsym=RAsym,
           Rk=Rk,Ri=Ri,modno=22, pn.options= "myoptions"),
           data = posneg.data, subset = id == "1")
age <- seq(0, 166, length.out = 101)
lines(age, predict(fm1, list(age = age)))
```

---

posnegRichards.calls   *List of calls for fitting 33 SSposnegRichards models in* [nlsList](#)

Description

The posnegRichards.calls list has two components of 17 and 16 rows and 1 column, respectively, called 'Examples of calls for FlexParamCurve models that estimate parameter m' (models with 4 estimable first curve parameters) and "Examples of calls for FlexParamCurve models that fix parameter m" (models with 3 estimable second curve parameters, i.e. M is fixed to value in $pnmodelparams$. Individual calls can be accessed by indexing first the component number and then the model number - see examples below. Note that model 17 is formulated differently (see [SSposnegRichards](#))

Usage

```
data(posnegRichards.calls)
```

Format

This object of class list containing the components:

Examples of calls for FlexParamCurve models that estimate parameter m a list of 16 possible reductions (nos. 1-16) of the FlexParamCurve double-Richards model that estimate parameter m. Also includes a custom model (17; see SSposnegRichards).

Examples of calls for FlexParamCurve models that fix parameter m a list of 16 possible reductions (nos. 21-36) of the FlexParamCurve double-Richards model that do not estimate parameter m but instead fix it to a mean across the dataset or user-specified value.

Details

A list object to provide users with examples of how to fit 33 different nlsList models using the selfStart function SSposnegRichards.

Examples

```
# extract the call for fitting a nls model with 8-parameter double-Richards curve (model 1)
data(posnegRichards.calls)
data(posneg.data)
#for an example just fit a subset of the data, 3 group levels (individuals)
   data <- subset(posneg.data, as.numeric(row.names (posneg.data) ) < 40)

modtofit <- as.character( posnegRichards.calls [[2]] [row.names(posnegRichards.calls [[2]]) == "22",] )
#change the data source
modtofit <- sub("posneg.data","data",modtofit)
modtofit <- parse(text = modtofit)

modpar(posneg.data$age, posneg.data$mass, pn.options = "myoptions") #create list for fixed parameters
eval(modtofit)
richardsR22.lis


# view call for model 1
posnegRichards.calls [[1]] [row.names(posnegRichards.calls [[1]]) == "1",]

# view call for model 21
posnegRichards.calls [[2]] [row.names(posnegRichards.calls [[2]]) == "21",]
```

---

posnegRichards.eqn   *Equations of the FlexParamCurve Family*

Description

Function to solve any of the equations in the FlexParamCurve family,
depending on user-specified parameters and model choice

Usage

```
posnegRichards.eqn(x,

Asym = NA,

K = NA,

Infl = NA,

M = NA,

RAsym = NA,

Rk = NA,

Ri = NA,

RM = NA,

modno,

pn.options)
```

Arguments

x	a numeric vector of the primary predictor variable
Asym	a numeric value for the asymptote of the positive (increasing) curve
K	a numeric value for the rate parameter of the positive (increasing) curve
Infl	a numeric value for the point of inflection of the positive (increasing) curve
M	a numeric value for the shape parameter of the positive (increasing) curve
RAsym	a numeric value for the asymptote of the negative (decreasing) curve
Rk	a numeric value for the rate parameter of the negative (decreasing) curve
Ri	a numeric value for the point of inflection of the negative (decreasing) curve
RM	a numeric value for the shape parameter of the negative (decreasing) curve
modno	a numeric value (currently integer only) between 1 and 36 specifying the identi-fication
	number of the equation to be fitted
pn.options	a character vector specifying a list of parameters and options for plotting

Details

This function fits 1 of 32 possible FlexParamCurve equations (plus custon model #17). Equations
can fit both monotonic and non-monotonic curves (two different trajectories).

These equations have also been described as double-Richards curves, or positive-negative Richards
curves.

From version 1.2 onwards this function can fit curves that exhibit negative followed by positive

trajectories or double-positive or double-negative trajectories. This function can now also fit two

component (biphasic) models, where the first curve is used up to the x-value (e.g. age) of intersec-
tion and the

second curve is used afterwards, thus the curves are not joined as in standard models (see `SSposnegRichards`

for details.

The 32 possible equations are all based on the subtraction of one Richards curve from another,
producing:

$y = A/([1 + mexp(-k(t - i))]1/m) - A'/([1 + m'exp(-k'(t - i'))]1/m')$, where A=Asym, k=K,
i=Infl, m=M,

A'=RAsym, k'=Rk, i'=Ri, m'=RM; as described in the Arguments section above.

All 32 possible equations are simply reformulations of this equation, in each case fixing a parameter
or

multiple parameters to (by default) the mean parameter across all individuals in the dataset (such as
produced by a `nls`

model). All models are detailed in the `SSposnegRichards` help file. Any models that require
parameter fixing

(i.e. all except model #1) extract appropriate values from the specified list passed by name to
pn.options for the fixed parameters.

This object is most easily created by running [modpar](#) and can be adjusted manually or by using [change.pnparameters](#) to user required specification.

If parameters are omitted in the call but required by the $modno$ specified in the call, then they will be automatically extracted

from the pn.options object supplied, with the appropriate warning. Thus, it is not necessary to list out parameters and modno but is

a useful exercise if you are unfamiliar or in doubt of exactly which model is being specified by $modno$, see [SSposnegRichards](#)

for a list. If a parameter is supplied separately with the call then this value will override those stored in for the same parameter in $modno$:

see examples below.

Value

the solution of the equation specified (by modno), given the user-entered parameters

Note

Any models that require parameter fixing (i.e. all except model #1) extract appropriate values from the specified

list passed to pn.options for the fixed parameters. This object is most easily created by running [modpar](#)

and can be adjusted manually or by using [change.pnparameters](#) to user required specification.

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

See Also

[SSposnegRichards](#)

[modpar](#)

Examples

```
require(graphics)


# calculate y (dependent variable) for a given x for an 8-parameter double-Richards model


    data(posneg.data)


   modpar(posneg.data$age, posneg.data$mass, pn.options = "myoptions") #create pnmodelparams for fixed paramete
```

```
      x = 10

      y <- posnegRichards.eqn(x, 1000, 0.5, 25, 1, 100, 0.5, 125,

      1, modno = 1, pn.options = "myoptions")

      print( c(x = x, y = y) )
```

```
# plot 8-parameter model using saved parameter values from modpar

 plot(posneg.data$age, posneg.data$mass, pch = ".")

 curve(posnegRichards.eqn(x,modno = 1, pn.options = "myoptions"), add = TRUE, lwd = 3)
```

```
# plot 3-parameter model using saved parameter values from modpar

      curve(posnegRichards.eqn(x,modno = 32, pn.options = "myoptions"), add = TRUE, col =2 , lwd = 3)
```

```
# tweak the plot of a 3-parameter model by user specifying a lower asymptote: ie give some parameter values

# directly and others through pn.options by default

      curve(posnegRichards.eqn(x,modno = 32, Asym = 3200, pn.options = "myoptions"), add = TRUE, col = 5, lwd =
```

```
# calculate y (dependent variable) for a given x for a 4-parameter Richards model, note that second curve paramete
```

```
# and replaced with value from pn.options. User-supplied variables over-ride those stored in pn.options object

 x = 10

    y <- posnegRichards.eqn(x, 1000, 0.5, 25, 1,

    1, modno = 12, pn.options = "myoptions")

    print( c(x = x, y = y) )



# plot a logistic curve (M=1),  note that second curve parameters are unneeded

    plot(1:200, posnegRichards.eqn(1:200, Asym = 1000, K = 0.5, Infl = 25, M = 1,

     modno = 12, pn.options = "myoptions"), xlim = c(1, 200), xlab = "x",

    ylab = "y", pch = 1, cex = 0.7)



# plot a double-logistic curve (M=1, RM=1),  note that second curve parameters are unneeded

    plot(1:200, posnegRichards.eqn(1:200, Asym = 1000, K = 0.5, Infl = 25, M = 1, RAsym = -100, Rk = 0.5, Ri = 125,

     modno = 1, pn.options = "myoptions"), xlim = c(1, 200), xlab = "x",

    ylab = "y", pch = 1, cex = 0.7)
```

---

posnegRichardsInit   *posnegRichardsInit*

Description

Not a user accessible function - integral to the workings of selfStart
SSposnegRichards

Usage

```
posnegRichardsInit(mCall,

LHS,

data,

modno,

pn.options)
```

Arguments

mCall	Call from SSposnegRichards
LHS	Left hand side of the formula from SSposnegRichards
data	Data object specified from SSposnegRichards
modno	Identification number of the model to be fitted specified from SSposnegRichards
pn.options	character string specifying name of list object populated with starting parameter estimates, fitting options and bounds

Details

Not a user accessible function - integral to the workings of selfStart
SSposnegRichards

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

See Also

SSposnegRichards

SSposnegRichards | *Self-Starting Positive-Negative Richards Model (double-Richards)*

Description

This selfStart function evaluates a range of flexible logistic
functions. It also has an initial attribute that creates
initial estimates of the parameters
for the model specified.

Usage

```
SSposnegRichards(x,

Asym = NA,

K = NA,

Infl = NA,

M = NA,

RAsym = NA,

Rk = NA,

Ri = NA,

RM = NA,

modno,

pn.options)
```

Arguments

x	a numeric vector of the primary predictor variable at which to evaluate the model
Asym	a numeric value for the asymptote of the positive (increasing) curve
K	a numeric value for the rate parameter of the positive (increasing) curve
Infl	a numeric value for the point of inflection of the positive (increasing) curve
M	a numeric value for the shape parameter of the positive (increasing) curve
RAsym	a numeric value for the asymptote of the negative (decreasing) curve
Rk	a numeric value for the rate parameter of the negative (decreasing) curve
Ri	a numeric value for the point of inflection of the negative (decreasing) curve
RM	a numeric value for the shape parameter of the negative (decreasing) curve
modno	a numeric value (currently integer only) between 1 and 36 specifying the identification number of the equation to be fitted
pn.options	character string specifying name of [list](#) object populated with starting parameter estimates, fitting options and bounds

Details

This selfStart function evaluates a range of flexible logistic
functions. It also has an initial attribute that creates
initial estimates of the parameters
for the model specified. Equations can fit both monotonic and non-monotonic
curves (two different trajectories). These equations have also been described as
double-Richards curves, or positive-negative Richards curves. **From version 1.2
onwards this function can fit curves that exhibit negative followed by positive
trajectories or double positive or double negative trajectories.***
The 32 possible equations (plus custom model #17) are all based on the subtraction of one Richards
curve from another, producing:

$$y = A/([1 + mexp(-k(t-i))]1/m) + A'/([1 + m'exp(-k'(t-i'))]1/m'),$$

where A=Asym, k=K, i=Infl, m=M,

A'=RAsym, k'=Rk, i'=Ri, m'=RM; as described in the Arguments section above.

All 32 possible equations are simply reformulations of this equation, in each

case fixing a parameter or multiple parameters to (by default) the mean parameter across

all individuals in the dataset (such as produced by a nls

model). Thus, a model in which one parameter is fixed has a 7-parameter equation,

and one in which four are fixed has a 4-parameter equation, thus reducing

complexity and computation and compensatory parameter changes when a parameter does not

vary across group levels (e.g individuals)

[the most appropriate equation can be determined using model selection in

pn.modselect.step or pn.mod.compare].

Any models that require parameter fixing (i.e. all except #1)

extract appropriate values from the list object specified by pn.options for the fixed

parameters. This object is most easily created by running

modpar and can be adjusted manually or by using

change.pnparameters to user-required specification.

Each of the 32 equations is identified by an integer value for modno (1 to 36).Models

21-36 are the same as 1-16 except that in the former the first curve parameter m

is fixed and not estimated. All equations (except 17 - see below) contain parameters Asym, K, and Infl.

The list below summarizes which of the other 5 parameters are contained in

which of the models (Y indicates that the parameter is estimated, blank indicates

it is fixed).

modno	M	RAsym	Rk	Ri	RM	NOTES
1	Y	Y	Y	Y	Y	8 parameter model
2	Y	Y	Y	Y		
3	Y			Y	Y	
4	Y	Y			Y	

```
5      Y                    Y

6      Y    Y          Y    Y

7      Y         Y    Y    Y

8      Y    Y    Y          Y

9      Y         Y          Y

10     Y              Y

11     Y    Y

12     Y  4 parameter, standard Richards model

13     Y    Y          Y

14     Y         Y    Y

15     Y    Y    Y

16     Y         Y

17     see below

18     see below

19     see below

20     see below
```

modno						
21		Y	Y	Y	Y	7 parameter model, 4 recession params
22	Y	Y		Y		6 parameters (used for double-logistic, double-Gompertz or double-Von Bert
23				Y	Y	
24		Y			Y	
25					Y	
26		Y		Y	Y	
27			Y	Y	Y	
28		Y	Y		Y	
29			Y		Y	
30				Y		
31		Y				
32	only 3 parameters (used for logistic, Gompertz or Von Bertalannfy - see below)					
33		Y		Y		
34			Y	Y		
35		Y	Y			
36			Y			

modno 17 represents a different parameterization for a custom model:

(Asym/ 1 + exp(Infl - x)/ M) - (RAsym / 1 + exp(Ri - x)/ RM), in

which M and RM actually represent scale parameters not shape parameters. This model and a suite of reductions:

modno 17.1:

modno 17.2:

modno 17.3:

are designed for use in modeling migration, *sensu*. Bunnefeld et al. 2011, Singh et al. In press.

modnos 18, 19 and 20 are reserved for internal use by modpar.

To access common 3 parameter sigmoidal models use modno = 32, fixing

parameters (using change.pnparameters) to M = 1 for logistic,

M = 0.1 for Gompertz, and M= -0.3 for von Bertalanffy. The same settings can be

used with modno = 2 to fit the double-logistic, double-Gompertz or double-Von Bertalannfy.

Note that to fit only 3 or 4 parameter curves, the option force4par = TRUE should be specified

when running modpar.

The call for SSposnegRichards only differs from

conventional selfStart models in that it requires a value for modno and a list of fitting options

and values from modpar to which to fix parameters in the reduced models.

Depending on the model chosen, different combinations of the 8 possible

parameters are required: if one is missing the routine will stop with an

appropriate error, if an extra one is added, it will be ignored (provided

that it is labelled, e.g. M = 1; this is good practice to prevent accidental

misassignments).

Here are two examples (7 parameter and 3 parameter):

```
richardsR2.lis <- nlsList(mass ~ SSposnegRichards(age,

        Asym = Asym, K = K, Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri,

        , modno = 2), data = posneg.data)


        #correct call includes all necessary parameters

richardsR20.lis <- nlsList(mass ~ SSposnegRichards(age,

        Asym = Asym, K = K, Infl = Infl, modno = 2), data = logist.data)


        #incorrect call missing required parameters, function terminates and generates an error messag
```

Examples for all models can be found in the list object

posnegRichards.calls.

If specified using modpar optional constraints

may be placed to specify response values at the minimum value and/or

maximum values of the predictor. Such constraint allows realistic fits for

datasets which are missing data

at either end of the curve (e.g. hatching weight for some growth curves).

Estimates are produced by splitting the two curves into separate positive

and negative curves and estimating parameters for each curve separately

in a similar manner to SSlogis. Each curve is fit first by

optim using the parameter bounds in pnmodelparamsbounds

(see modpar) and a subsequent refinement is attempted using

nls with more restrictive parameter bounds. Finally, both curves

are annealed

and parameters are again estimated using restrictive bounds and starting values

already determined during separate estimates. Equations for which the positive

curve was inestimable are not estimated further, but if negative curve estimation

or overall curve estimation fail, partial estimates are used: either default

negative parameters (RAsym = 0.05*Asym, Rk = K, Ri = Infl, RM = M) annealed to positive

curves or separate estimates annealed; both with compensation for interation

between asymptotes.

From version 1.2 onwards, this function can now fit two component models, where

the first curve is used up to the x-value (e.g. age) of intersection and the second curve is used

afterwards. Confusingly, these are also called "double Richards", "double Gompertz"

or "double logistic": see Murphy et al. (2009) or Ross et al. (1995) for examples.

To specify such models set twocomponent.x = TRUE (this will estimate the x of

intersection) when running modpar. Alternatively, if known, the

x of intersection can be set directly by setting

twocomponent.x = # (where # is the x of intersection). Whne {codemodpar

is run this option will be saved in pnmodelparas and can be changed at will,

either manually or using change.pnparameters.

From version 1.2 onwards, this function can now fit bilogistic (and more generally

biRichards) curves, where the final curve is effectively either two positive curves

or two negative curves. See Meyer (1994) for examples. This functionality is default

and does not need to be specified.

Value

a numeric vector of the same length as x containing parameter

estimates for equation specified (fixed parameters

are not return but are substituted in calls to nls

nlsList and nlme with the fixed parameters

stored in pnmodelparams; see modpar

Note

Any models that require parameter fixing (i.e. all except #1)

extract appropriate values from the object pnmodelparams for the fixed

parameters. This object is created by running

modpar and can be adjusted manually or by using

change.pnparameters to user required specification.

Output may show errors and warnings especially during a nlsList

fit, in which the function is called repeatedly: once for each group level in the

dataset. Warnings indicate conditions for which default parameters or incomplete estimates

are used - see Details section - and errors occur from insufficient data or singularities.

As a result of possible interaction and correlation between the parameters in some models,

singularities may be common, but do not be alarmed by repeated error messages, as

examination of a fitted nlsList model may releave a large number of

well estimated group levels, thus the elimation of unsuitable outlying groups only. Also,

because very few of the 32 equations are likely to be suitable for the majority

of datasets, consideration of the model being fitted is crucial when examining the output. Functions

pn.modselect.step and pn.mod.compare provide the ability

for model selection of these equations through stepwise backward deletion or all

model comparison, respectively. These offer powerful ways to determine the

best equation for your dataset.

To increase the ability of optimization routines to deal with

a wide variety of values, particularly negative values for M or RM,

only real component of complex numbers are modelled and integer versions

of M and RM are used during estimation if floating values cause conversion

issues.

Speed of the function depends on the complexity of the data being fit.

Author(s)

Stephen Oswald <ozsao23@hotmail.com>

References

Oswald, S.A. et al. (Early view) FlexParamCurve: R package for flexible
fitting of nonlinear parametric curves. Methods in Ecology and Evolution.
doi: 10.1111/j.2041-210X.2012.00231.x (see also tutorial and introductory videos at:
http://www.methodsinecologyandevolution.org/view/0/podcasts.html
(posted September 2012 - if no longer at this link, check the archived videos at:
http://www.methodsinecologyandevolution.org/view/0/VideoPodcastArchive.html#allcontent
#1# Nelder, J.A. (1962) Note: an alternative form of a generalized
logistic equation. Biometrics, 18, 614-616.
#2#
Huin, N. & Prince, P.A. (2000) Chick growth in albatrosses: curve fitting
with a twist. Journal of Avian Biology, 31, 418-425.
#3#
Meyer, P. (1994) Bi-logistic growth. Technological Forecasting and Social
Change. 47: 89-102
#4#
Murphy, S. et al. (2009) Importance of biological parameters in assessing
the status of `Delphinus delphis`. Marine Ecology Progress Series 388: 273-291.
#5#
Pinheiro, J. & Bates, D. (2000) `Mixed-Effects Models in S and S-Plus`.
Springer Verlag, Berlin.
#6#
Ross, J.L. et al. (1994) Age, growth, mortality, and reproductive biology
of red drums in North Carolina waters. Transactions of the American Fisheries
Society 124: 37-54.
#7#
Bunnefeld et al. (2011) A model-driven approach to quantify migration patterns:
individual, regional and yearly differences. Journal of Animal Ecology 80: 466-476.
#8#
Singh et al. (In press) From migration to nomadism: movement variability in a northern
ungulate across its latitudinal range. Ecological Applications.

See Also

SSlogis
SSgompertz
posnegRichards.eqn

Examples

```
set.seed(3) #for compatability issues

require(graphics)

   # retrieve mean estimates of 8 parameters using getInitial

   # and posneg.data object

   data(posneg.data)

 modpar(posneg.data$age, posneg.data$mass,verbose=TRUE, pn.options = "myoptions", wide.bounds=TRUE)

   getInitial(mass ~ SSposnegRichards(age, Asym, K, Infl, M,

       RAsym, Rk, Ri, RM, modno = 1, pn.options = "myoptions"), data = posneg.data)



   # retrieve mean estimates and produce plot to illustrate fit for

   # curve with M, Ri and Rk fixed

   pars <- coef(nls(mass ~ SSposnegRichards(age,

       Asym = Asym, K = K, Infl = Infl, RAsym = RAsym,

        RM = RM, modno = 24, pn.options = "myoptions"), data = posneg.data,

        control=list(tolerance = 10)))

   plot(posneg.data$age, posneg.data$mass, pch=".")

   curve(posnegRichards.eqn(x, Asym = pars[1], K = pars[2],
```

```
         Infl = pars[3], RAsym = pars[4],

         RM = pars[5],  modno = 24, pn.options = "myoptions"), xlim = c(0,

         200), add = TRUE)



    # retrieve mean estimates and produce plot to illustrate fit for custom model 17



   ## Not run:    pars<-as.numeric( getInitial(mass ~ SSposnegRichards(age, Asym, K, Infl,

         M, RAsym, Rk, Ri, RM, modno = 17, pn.options = "myoptions"), data = datansd) )

    plot(datansd$jday21March, datansd$moosensd)

    curve( posnegRichards.eqn(x, Asym = pars[1], K = 1, Infl = pars[2],

          M = pars[3], RAsym = pars[4], Rk = 1, Ri = pars[5], RM = pars[6],

          modno = 17, pn.options = "myoptions"), lty = 3, xlim = c(0, 200) , add = TRUE)
## End(Not run)



    # fit nls object using 8 parameter model

    # note: ensure data object is a groupedData object

    richardsR1.nls <- nls(mass ~ SSposnegRichards(age, Asym = Asym,

       K = K, Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri,
```

```
        RM = RM, modno = 1, pn.options = "myoptions"), data = posneg.data)



    # fit nlsList object using 8 parameter model

    # note: ensure data object is a groupedData object

    # also note: not many datasets require all 8 parameters



    ## Not run:

    richardsR1.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym,

        K = K, Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri,

        RM = RM, modno = 1, pn.options = "myoptions"), data = posneg.data)

    summary(richardsR1.lis)
## End(Not run)



    # fit nlsList object using 6 parameter model with value M and RM

    # fixed to value in pnmodelparams and then fit nlme model

    # note data is subset to provide estimates for a few individuals

    # as an example

    subdata <- subset(posneg.data,posneg.data$id == as.character(36)
```

```
| posneg.data$id == as.character(9)

| posneg.data$id == as.character(32)

| posneg.data$id == as.character(43))

richardsR22.lis <- nlsList(mass ~ SSposnegRichards(age, Asym = Asym,

    K = K, Infl = Infl, RAsym = RAsym, Rk = Rk, Ri = Ri,

    modno = 22, pn.options = "myoptions"), data = subdata)

summary(richardsR22.lis )

richardsR22.nlme <- nlme(richardsR22.lis, random = pdDiag(Asym + Infl ~ 1) )

summary(richardsR22.nlme)



# fit nls object using simple logistic model, with

# M, RAsym, Rk, Ri, and RM fixed to values in pnmodelparams

 data(logist.data)

modpar(logist.data$age, logist.data$mass ,force4par = TRUE, pn.options = "myoptions")

change.pnparameters(M = 1, pn.options = "myoptions") #set to logistic (M =1) prior to fit

 richardsR32.nls <- nls(mass ~ SSposnegRichards(age, Asym = Asym,

    K = K, Infl = Infl, modno = 32, pn.options = "myoptions"), data = logist.data)

coef(richardsR32.nls)
```

```
# fit a two component model - enter your own data in place of "mydata"



## Not run:

modpar(mydata$x,mydata$y,twocomponent.x=TRUE, pn.options = "myoptions") # if x of intersection unknown

modpar(mydata$x,mydata$y,twocomponent.x=75, pn.options = "myoptions") # if x of intersection = 75

richardsR1.nls <- nls(y~ SSposnegRichards(x, Asym = Asym, K = K,

 Infl = Infl, M = M, RAsym = RAsym, Rk = Rk, Ri = Ri, RM = RM, modno = 1, pn.options = "myoptions")

                     , data = mydata)


coef(richardsR1.nls)
## End(Not run)
```

---

tern.data   *Field data on growth of common terns $Sterna hirundo$*

Description

The tern.data data frame has 1164 rows and 12 columns of records of the measured masses for common tern chicks between 1 and 30 days of age collected at at Grays Beach, MA, in 1973 (Nisbet 1975) and Monomoy, MA, in 1975 (Nisbet et al. 1978)

Usage

```
data(tern.data)
```

Format

This object of class c("nfnGroupedData", "nfGroupedData",   "groupedData", "data.frame")
containing the following columns:

site Four character factor for the two sites (MYMA: Monomoy Island, MA; GBCT: Grays Beach,
  CT).

year A factor specifying the year of measurement: 1973 or 1976.

bandid an ordered factor indicating unique id of each individual: the union of the laying date of
  the nest relative to the colony and the band combination

siteyear A factor specifying levels of year for different sites (different years at each site).

weight a numeric vector of chick masses (g).

ckage a numeric vector of chick ages (days).

Jdate a numeric vector of first egg-laying date of the nest(days), relative to the mean laying date
  for all nests in that year.

nest A factor of unique codes that identify each nest.

ck A factor of hatching order for each chick (A = first hatched, B = second hatched C = third
  hatched).

outcome A factor of codes for fate of each chick (F = fledged; only fledged chicks included).

eggmass A numeric vector of the mass of the egg (from which the chick hatched) at laying.

clutch A factor of size of clutch/brood that each chick comes from (either 1- or 2-chick brood).

Details

Data were collected as outlined in Nisbet (1975)[Grays Beach, MA, 1973] and Nisbet et al.(1978)
[Monomoy, MA, 1975]. Please contact Ian Nisbet <icnisbet@verizon.net> for use in collaborations.

Source

Nisbet, I.C.T. (1975) Selective effects of predation in a tern colony. Condor, 77, 221-226. Nisbet,
I.C.T., Wilson, K.J. & Broad, W.A. (1978) Common Terns raise young after death of their mates.
Condor, 80, 106-109.

Examples

```
require(stats); require(graphics)
data(tern.data)

modpar(tern.data$ckage, tern.data$weight, force4par = TRUE, pn.options = "myoptions") #create pnmodelparams for
plot(weight ~ ckage, data = tern.data, subset = bandid == tern.data$bandid[1],
     xlab = "Chick age (day)", las = 1,
     ylab = "Chick mass (g)",
     main = "tern.data and fitted curve (Chick #156 only)")
fm1 <- nls(weight ~ SSposnegRichards(ckage,Asym=Asym,K=K,Infl=Infl,modno=32, pn.options= "myoptions"),
           data = tern.data, subset = bandid == tern.data$bandid[1])
ckage <- seq(0, 30, length.out = 101)
lines(ckage, predict(fm1, list(ckage = ckage)))
```

Index