Package 'selfisher'

May 11, 2020

Title Selectivity of Fisheries Gear, Modeled using Template Model Builder

Version 0.2.0

Date 2019-4-9

Description Fit selectivity models in R. The models are fit using maximum likelihood estimation via 'TMB' (Template Model Builder). Random effects are assumed to be Gaussian on the scale of the linear predictor and are integrated out using the Laplace approximation. Gradients are calculated using automatic differentiation.

License AGPL-3

Imports methods,
TMB (>= 1.7.13),
lme4 (>= 1.1-18.9000),
Matrix,
nlme

LinkingTo TMB, RcppEigen

Suggests knitr,
rmarkdown,
testthat,
roxygen2,
MASS,
lattice,
ggplot2,
mlmRev,
bbmle,
reshape2

VignetteBuilder knitr

URL https://github.com/mebrooks/selfisher

LazyData TRUE

BugReports https://github.com/mebrooks/selfisher/issues

RoxygenNote 7.0.2

NeedsCompilation yes

Author Mollie Brooks [aut, cre]

Maintainer Mollie Brooks <mollieebrooks@gmail.com>

R topics documented:

`bootSel` *Perform bootstrap*

Description

Perform bootstrap

Usage

```
bootSel(
  x,
  FUN = L50SR,
  nsim = 2,
  seed = NULL,
 type = c("double nonparametric", "double binomial", "parametric", "nonparametric"),
  verbose = FALSE,
  .progress = "none",
  PBargs = list(),
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  cl = NULL
)
```

Arguments

x	a fitted selfisher object
FUN	a function taking a fitted selfisher object as input and returning the *statistic* of interest, which must be a (possibly named) numeric vector. The default (FUN = L50SR) only works with very simple models; use predict.selfisher.
nsim	number of simulations, positive integer
seed	optional argument to set.seed
type	character string specifying the type of bootstrap, from the following: • "double nonparametric" resample hauls, then resample observed fish within hauls. • "double binomial" resample hauls, then in each haul simulate fish into each length class. The total number in each length class is equal to the observed number in that length class in that haul. The probability of ending up in either gear in rbinom is equal to the observed proportion in the original data. • "parametric" simulates from the fitted model. Any random effects are simulated from their estimated normal distribution. • "nonparametric" All resampling is done with replacement. All resampling is done on observed fish only, not the raised numbers. Raising before resamping reduces variability as if more observations were made.

Details

The code structure is based on code from the lme4 package, except that bootstraps of type "double" are specific to fisheries gear selectivity literature (Millar 1993). This code has not been tested on models containing random effects. The double bootstrap procedures account for variability among "hauls" and it should be possible to use this feature to account for any factor that could be treated as a random effect. It is possible to resample hauls from multiple pools while producing the same number of hauls per pool in the bootstrap replicates (Herrmann et al. 2017). See vignette("bootstrap") for an example.

---

confint.selfisher *Calculate confidence intervals*

Description

Calculate confidence intervals

Usage

```
## S3 method for class 'selfisher'
confint(
  object,
  parm,
  level = 0.95,
  method = c("wald", "Wald", "profile", "uniroot"),
  component = c("all", "r", "p", "other"),
```

```
  estimate = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("profile.ncpus", 1L),
  cl = NULL,
  ...
)
```

Arguments

object	selfisher fitted object.
parm	Specification of a parameter subset *after* component subset has been applied.
level	Confidence level.
method	'wald', 'profile', or 'uniroot': see Details function)
component	Which of the three components 'r', 'p' or 'other' to select. Default is to select 'all'.
estimate	(logical) add a third column with estimate ?
parallel	method (if any) for parallel computation
ncpus	number of CPUs/cores to use for parallel computation
cl	cluster to use for parallel computation
...	arguments may be passed to profile.selMod or tmbroot

Details

Available methods are

wald These intervals are based on the standard errors calculated for parameters on the scale of their internal parameterization depending on the family. Derived quantities such as standard deviation parameters and dispersion parameters are backtransformed. It follows that confidence intervals for these derived quantities are asymmetric.

profile This method computes a likelihood profile for the specified parameter(s) using profile.glmmTMB; fits a spline function to each half of the profile; and inverts the function to find the specified confidence interval.

uniroot This method uses the uniroot function to find critical values of one-dimensional profile functions for each specified parameter.

---

findReTrmClasses *list of specials – taken from enum.R*

Description

list of specials – taken from enum.R

Usage

```
findReTrmClasses()
```

fixef *Extract fixed-effects estimates*

Description

Extract the fixed-effects estimates

Usage

```
## S3 method for class 'selfisher'
fixef(object, ...)
```

Arguments

object	any fitted model object from which fixed effects estimates can be extracted.
...	optional additional arguments. Currently none are used in any methods.

Details

Extract the estimates of the fixed-effects parameters from a fitted model.

Value

a named, numeric vector of fixed-effects estimates.

Examples

```
data(haddock)
dat=transform(haddock, tot=nfine+nwide, prop=nwide/(nfine+nwide))
fixef(selfisher(prop~Lengths, p=~1, psplit=TRUE, total=tot, dat))
```

getCapabilities *List model options that selfisher knows about*

Description

List model options that selfisher knows about

Usage

```
getCapabilities(what = "all")
```

Arguments

what	(character) which type of model structure to report on ("all","family","link","covstruct")

Note

these are all the options that are *defined* internally; they have not necessarily all been *implemented* (FIXME!)

getME.selfisher — *Extract or Get Generalize Components from a Fitted Mixed Effects Model*

Description

Extract or Get Generalize Components from a Fitted Mixed Effects Model

Usage

```
## S3 method for class 'selfisher'
getME(
  object,
  name = c("Xr", "Xp", "Zr", "Zp", "Xd", "betar", "betap", "betad", "br", "bp",
    "thetar", "thetap"),
  ...
)
```

Arguments

object	a fitted selfisher object
name	of the component to be retrieved
...	ignored, for method compatibility

See Also

getME Get generic and re-export:

getReStruc — *Calculate random effect structure Calculates number of random effects, number of parameters, blocksize and number of blocks. Mostly for internal use.*

Description

Calculate random effect structure Calculates number of random effects, number of parameters, blocksize and number of blocks. Mostly for internal use.

Usage

```
getReStruc(reTrms, ss = NULL)
```

Arguments

reTrms	random-effects terms list
ss	a character string indicating a valid covariance structure. Must be one of names(selfisher:::.valid default is to use an unstructured variance-covariance matrix ("us") for all blocks).

Value

a list

blockNumTheta	number of variance covariance parameters per term
blockSize	size (dimension) of one block
blockReps	number of times the blocks are repeated (levels)
covCode	structure code

Examples

```
data(sleepstudy, package="lme4")
rt <- lme4::lFormula(Reaction~Days+(1|Subject)+(0+Days|Subject),
                     sleepstudy)$reTrms
rt2 <- lme4::lFormula(Reaction~Days+(Days|Subject),
                     sleepstudy)$reTrms
getReStruc(rt)
```

---

getXReTrms *Create X and random effect terms from formula*

Description

Create X and random effect terms from formula

Usage

```
getXReTrms(formula, mf, fr, ranOK = TRUE, type = "")
```

Arguments

formula	current formula, containing both fixed & random effects
mf	matched call
fr	full model frame
ranOK	random effects allowed here?
type	label for model type

Value

a list composed of

X	design matrix for fixed effects
Z	design matrix for random effects
reTrms	output from mkReTrms from **lme4**
offset	offset vector, or vector of zeros if offset not specified

`interceptinit` *Initialize the intercept based on the link funciton Assuming catchability of length 0 indivs is near 0*

Description

Initialize the intercept based on the link funciton Assuming catchability of length 0 indivs is near 0

Usage

```
interceptinit(link)
```

Arguments

`link`	character

`L50SR` *a function taking a fitted* `selfisher` *object as input and returning the L50 and SR estimates as a named numeric vector.*

Description

a function taking a fitted `selfisher` object as input and returning the L50 and SR estimates as a named numeric vector.

Usage

```
L50SR(x)
```

Arguments

`x`	a fitted `selfisher` object

`numFactor` *Factor with numeric interpretable levels.*

Description

Create a factor with numeric interpretable factor levels.

Usage

```
numFactor(x, ...)

parseNumLevels(levels)
```

Arguments

x	Vector, matrix or data.frame that constitute the coordinates.
...	Additional vectors, matrices or data.frames that constitute the coordinates.
levels	Character vector to parse into numeric values.

Details

Some selfisher covariance structures require extra information, such as temporal or spatial coordinates. numFactor allows to associate such extra information as part of a factor via the factor levels. The original numeric coordinates are recoverable without loss of precision using the function parseNumLevels. Factor levels are sorted coordinate wise from left to right: first coordinate is fastest running.

Value

Factor with specialized coding of levels.

Examples

```
## 1D example
numFactor(sample(1:5,20,TRUE))
## 2D example
coords <- cbind( sample(1:5,20,TRUE), sample(1:5,20,TRUE) )
(f <- numFactor(coords))
parseNumLevels(levels(f)) ## Sorted
## Used as part of a model.matrix
model.matrix( ~f )
## parseNumLevels( colnames(model.matrix( ~f )) )
## Error: 'Failed to parse numeric levels: (Intercept)'
parseNumLevels( colnames(model.matrix( ~ f-1 )) )
```

---

predict.selfisher *prediction*

Description

prediction

Usage

```
## S3 method for class 'selfisher'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  re.form,
  allow.new.levels = FALSE,
  type = c("response", "selection", "prob", "ratio", "link"),
  na.action = na.pass,
  debug = FALSE,
  ...
)
```

Arguments

object	a selfisher object
newdata	new data for prediction
se.fit	return the standard errors of the predicted values?
re.form	(not yet implemented) specify which random effects to condition on when predicting. For now, all random effects are included.
allow.new.levels	(not yet implemented) allow previously unobserved levels in random-effects grouping variables?
type	• return expected response value ("response": see details below), • predicted selection curve ("selection": r), • relative fishing power of the test gear ("prob"), • catch ratio from catch comparison models ("ratio": r/(1-r)). Some types (ratio, link) might not make sense to use with psplit models.
debug	(logical) return the TMBStruc object that will be used internally for debugging?
...	unused - for method compatibility

Details

Predicting with type="response" returns values comparable to the response variable (the left-hand side of the model's rformula); that is pr/(pr+1-p) in a model with psplit=TRUE or r in a model with psplit=FALSE. This function could work with random effects, but is untested.

Examples

```
data(haddock)
dat <- transform(haddock, tot=nfine+nwide, prop=nwide/(nfine+nwide))
m1 <- selfisher(prop~Lengths, p=~1, psplit=TRUE, total=tot, dat)
nd <- data.frame(Lengths=20:50, tot=100)
predict(m1, newdata=nd, se.fit=TRUE)
```

---

print.VarCorr.selfisher *Printing The Variance and Correlation Parameters of a* selfisher

---

Description

Printing The Variance and Correlation Parameters of a selfisher

Usage

```
## S3 method for class 'VarCorr.selfisher'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  comp = "Std.Dev.",
  formatter = format,
  ...
)
```

Arguments

x	a result of VarCorr(<selfisher>).
digits	number of significant digits to use.
comp	a string specifying the component to format and print.
formatter	a function.
...	optional further arguments, passed the next print method.

---

ranef.selfisher *Extract Random Effects*

Description

Generic function to extract random effects from selfisher models, both for the selectivity (i.e. retention) model and relative fising power model.

Usage

```
## S3 method for class 'selfisher'
ranef(object, ...)
```

Arguments

object	a selfisher model.
...	some methods for this generic function require additional arguments.

Value

Object of class ranef.selfisher with two components:

r	a list of data frames, containing random effects for the selectivity (i.e. retention) model.
p	a list of data frames, containing random effects for the relative fising power model.

Note

When a model has no model of relative fishing power, the default behavior of ranef is to simplify the printed format of the random effects. To show the full list structure, run print(ranef(model),simplify=FALSE). In all cases, the full list structure is used to access the data frames (see example).

See Also

fixef.selfisher. ##'

Examples

```
data(ccmhsdat)
ranef(selfisher(prop~length*type+(1|haul), total=total, ccmhsdat))
print(ranef(selfisher(prop~length*type+(1|haul), total=total, ccmhsdat)), simplify=FALSE)
```

read_in_haul *read in data from a single haul*

Description

read in data from a single haul

Usage

```
read_in_haul(
  x,
  name = "Haul",
  extension = ".txt",
  raising = NULL,
  sampling = NULL
)
```

Arguments

x	possibly a number or other indicator of the unique haul
name	part of the file name that stays the same
extension	what type of file is it
raising	name of raising factor if there is one e.g. "RAISING_FACTOR"
sampling	name of sampled fraction if there is one e.g. "SAMPLING"

Details

the name of the file where the data is stored is paste0(name, x, extension)

refit.selfisher *refit the same model to a new response*

Description

refit the same model to a new response

Usage

```
## S3 method for class 'selfisher'
refit(object, newdata, ...)
```

Arguments

object	a fitted selfisher object
newdata	a data set with the same predictors used in the model

residuals.selfisher *Compute residuals for a selfisher object*

Description

Compute residuals for a selfisher object

Usage

```
## S3 method for class 'selfisher'
residuals(object, type = c("response", "pearson", "deviance"), ...)
```

Arguments

object	a "selfisher" object
type	(character) residual type
...	ignored, for method compatibility

Richardsdelta *Extract Richards exponent parameter*

Description

Extract Richards exponent parameter

Usage

```
Richardsdelta(object, ...)
```

Arguments

object	a "selfisher" fitted object
...	(ignored; for method compatibility)

`selfisher` *Fit gear selectivity models with TMB*

Description

Fit gear selectivity models with TMB

Usage

```
selfisher(
  rformula,
  data = NULL,
  pformula = ~1,
  dformula = ~1,
  psplit = FALSE,
  start = NULL,
  link = "logit",
  total = NULL,
  haul = NULL,
  pool = NULL,
  offset = NULL,
  Lp = "basic",
  se = TRUE,
  verbose = FALSE,
  debug = FALSE,
  optControl = list(iter.max = 300, eval.max = 400)
)
```

Arguments

`rformula`	combined fixed and random effects formula for the selectivity model, following lme4 syntax. The left-hand side of the formula should be the proportion of fish entering the test gear.
`data`	data frame
`pformula`	a *one-sided* (i.e., no response variable) formula for the ralaive fishing power of the test versus the control gear combining fixed and random effects: `~0` can be used to specify equal fishing power (p=0.5). The relative fishing power model uses a logit link.
`dformula`	a formula for the delta parameter in Richards selection curve. Ignored unless `link="richards"`.
`psplit`	(logical) Does the model contain psplit as in eqn 3 of Wileman et al. 1996? For covered codend and catch comparison, use psplit=FALSE.
`start`	starting values, expressed as a list with possible components `betar`, `betap`, `betad` (fixed-effect parameters for retention, psplit, Richards delta models); `br`, `bp` (conditional modes for retention and psplit models); `thetar`, `thetap` (random-effect parameters, on the standard deviation/Cholesky scale, for retention and psplit models);

link	A character indicating the link function for the selectivity model. "logit"(logistic) is the default, but other options are "probit" (i.e. normal probability ogiv), "cloglog" (i.e. negative extreme value), "loglog" (i.e. extreme value/Gompert), or "Richards"
total	The number of total fish caught in the test and control gear.
haul	Name of column representing different hauls. Needed for double bootstrap.
pool	(Optional) name of column representing different pools of hauls. Used in double bootstrap to produce same number of hauls by pool.
Lp	controls calculation of length (L) at retention prob (p), see details
se	whether to return standard errors
verbose	logical indicating if some progress indication should be printed to the console.
debug	whether to return the preprocessed data and parameter objects, without fitting the model
optControl	control parameters passed to nlminb

Details

- in all cases selfisher returns maximum likelihood estimates.
- You only need to specify haul in models that are going to be bootstraped with type="double".
- Lp="basic" will return values for L50 and SR.
- Lp="none" supresses calculation of L50 and SR to save time.
- Lp="full" will return values of Lp for p=5 to 95 as well as SR
- Lp="100" will return values of Lp for p=1 to 100 as well as SR
- Use getCapabilities() to see options for links and RE

Examples

```
dat <- transform(haddock, tot=nfine+nwide, prop=nwide/(nfine+nwide))
m0 <- selfisher(prop~Lengths, pformula=~0, psplit=TRUE, total=tot, dat)
m1 <- selfisher(prop~Lengths, pformula=~1, psplit=TRUE, total=tot, dat)
```

---

simulate.selfisher *Simulate from a selfisher fitted model*

Description

Simulate from a selfisher fitted model

Usage

```
## S3 method for class 'selfisher'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	selfisher fitted model
nsim	number of response lists to simulate. Defaults to 1.
seed	random number seed
...	extra arguments

Details

Random effects are also simulated from their estimated distribution. Currently, it is not possible to condition on estimated random effects.

Value

returns a list of vectors. The list has length `nsim`. Each simulated vector of observations is the same size as the vector of response variables in the original data set.

`vcov.selfisher` *Calculate Variance-Covariance Matrix for a Fitted selfisher model*

Description

Calculate Variance-Covariance Matrix for a Fitted selfisher model

Usage

```
## S3 method for class 'selfisher'
vcov(object, full = FALSE, ...)
```

Arguments

`object`	a "selfisher" fit
`full`	return a full variance-covariance matrix?
`...`	ignored, for method compatibility

Value

By default (`full==FALSE`), a list of separate variance-covariance matrices for each model component (conditional, zero-inflation, dispersion). If `full==TRUE`, a single square variance-covariance matrix for *all* top-level model parameters (conditional, dispersion, and variance-covariance parameters)

Index