Package: rfutils (via r-universe)

September 12, 2024

Title Useful Functions For Model Selection

Version 0.0.3

Description A set of utility functions for modelling.

Imports methods, stats, mgcv, pbapply, gamm4, lme4, Matrix

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://faskally.r-universe.dev

RemoteUrl https://github.com/faskally/rfutils

RemoteRef HEAD

RemoteSha c6bc983c90620d61db1794d8c73602a034198462

Contents

`rfutils-package` *Useful function for model selection*

Description

A set (really fly) of utility functions for modelling.

add1drop1 *Up-down model selection*

Description

At each stage of model selection it was possible to drop interaction terms involving the categorical variables, replace smooth functions of the continuous variables with linear e<U+FB00>ects and drop the main effects of categorical and continuous variables (provided they were not involved in any interactions or expressed as smooth functions). It was also possible to introduce interactions between the continuous variables and Life-stage and Pass, to allow for di<U+FB00>erent relationships for fry and parr and for the <U+FB01>rst and subsequent pass capture probabilities. Two-way interactions were also considered between Lifestage and HA, and up to three-way interactions Lifestage, Year and Organisation

Usage

```
add1drop1(
  model,
  scope,
  smoothTerms,
  notLinear,
  essential,
  criterion,
  cluster = NULL,
  byExpand = FALSE,
  calcLogLik,
  fitModels = TRUE,
  retainModels = TRUE
)
```

Arguments

model	detail
scope	detail
smoothTerms	detail
notLinear	detail
essential	detail
criterion	detail
cluster	detail
byExpand	detail
calcLogLik	detail
fitModels	detail
retainModels	detail

Details

scope is the maximal model which, for simplicity, is expressed only using linear terms. So, if smoothTerms = list(altitude = "s(altitude, k = 3)", dayOfYear = "s(dayOfYear, k = 3)", network = "s(network, k = 4)", hydroArea = "s(hydroArea, k = 4, bs = \"tp\")", longitude = t2(longitude, latitude)) then scope = pass * lifestage * (altitude + dayOfYear + longitude) denotes a maximal model pass + lifestage + pass:lifestage + s(altitude, k = 3, by = pass:lifestage) + s(dayOfYear, k = 3, by = pass:lifestage) + t2(longitude, latitude, by = pass:lifestage) NB Don't use : in scope and don't use 1 unless it is the null model (which is pretty pointless) NB Still need to sort out code when by is used with more that one variable Currently, the term s(x, by = y:z) is actually fitted as s(x, by = y) + s(x, by = z) + s(x, by = y:z) If you get models like this turning up in the model selection, interpret them with great care! smoothTerms gives the form of the smoother for each term in the model NB for a 2 or 3 dimensional smoother, just give the first term - these smoothers must also be identified in the notLinear argument notLinear specifies the smoothers which are not reduced to a linear term before being dropped e.g. notLinear = "hydroArea" would consider dropping the hydroArea term completely, rather than introducing hydroArea as a factor (which doesn't make sense because that is a more complicated model) essential specifies terms that must be in every model these must not be the same as any terms in the scope (or even related to them in any way) leave any offset out of this - dealt with separately criterion (default = AIC) is a function which allows the user to define the model selection criterion cluster is the identity of a cluster to allow parallel processing drop1.efp and other terms (model, the data set used, scope and smoothTerms) must previously be exported to cluster using clusterExport byExpand = FALSE (default) is designed for use when the by variables are factors - fits a separate smoother for each level of the factor, but with no main effect (of the covariate) byExpand = TRUE is for when there is a single by variable and it is continuous - fits both the main effect of the covariate (as a smoother) and the product of the by variable and a different smooth of the covariate calcLogLik (default = logLik) allows for a more general log likelihood calculation, for example, in gam models with random effects, where you would want to use the marginal (integrated) likelihood in calcLogLik = function(model) -model$gcv.ubre is the obvious choice fitModels = TRUE (default) fits all the candidate model fitModels = FALSE just returns the list of candidate models without fitting - useful if you want to just check which terms come in and drop out retainModels = TRUE (default) returns the updated model retainModels = FALSE just returns the updated model formula - saves memory utility functions - throughout id is the identity of a covariate in smoothTerms and rhs is a character vector of model terms converts rhs to single formula-like string

References

Malcom et al. 2019. Development of a large-scale juvenile density model to inform the assessment and management of Atlantic salmon (Salmo salar) populations in Scotland

---

`logLik_gam` *logLik_gam*

Description

Log likelihood function for use in model selection of mixed models using gam or bam Here's a function for extracting the log likelihood and (harder) the degrees of freedom out of a gam or bam model. The degrees of freedom are the number of fixed effects plus the number of variances (not the edf as is usually reported by gam) and assumes the models are fitted by ML.

Usage

```
logLik_gam(model)
```

Arguments

model	detail

Details

The function has been tested with a range of smoothers, but as good check would involve fitting models in gam and also in gamm4 and check the df are the same.

---

my_gamm4 *fixes bug in gamm4 to allow different optimisers*

Description

fixes bug in gamm4 to allow different optimisers

Usage

```
my_gamm4(
  formula,
  random = NULL,
  family = gaussian(),
  data = list(),
  weights = NULL,
  subset = NULL,
  na.action,
  knots = NULL,
  drop.unused.levels = TRUE,
  REML = TRUE,
  control = NULL,
  start = NULL,
  verbose = 0L,
  mer_only = FALSE,
  ...
)
```

Arguments

formula	A GAM formula (see also formula.gam and gam.models). This is like the formula for a glm except that smooth terms (s and t2 but not te) can be added to the right hand side of the formula. Note that ids for smooths and fixed smoothing parameters are not supported.ils
random	An optional formula specifying the random effects structure in lmer style. See example gamm4.

family	A family as used in a call to glm or gam.
data	A data frame or list containing the model response variable and covariates required by the formula. By default the variables are taken from environment(formula), typically the environment from which gamm4 is called.
weights	a vector of prior weights on the observations. NULL is equivalent to a vector of 1s. Used, in particular, to supply the number-of-trials for binomial data, when the response is proportion of successes.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain 'NA's. The default is set by the 'na.action' setting of 'options', and is 'na.fail' if that is unset. The "factory-fresh" default is 'na.omit'.
knots	this is an optional list containing user specified knot values to be used for basis construction. Different terms can use different numbers of knots, unless they share a covariate.
drop.unused.levels	by default unused levels are dropped from factors before fitting. For some smooths involving factor variables you might want to turn this off. Only do so if you know what you are doing.
REML	passed on to lmer fitting routines (but not glmer fitting routines) to control whether REML or ML is used.
control	lmerControl or glmerControl list as appropriate (NULL means defaults are used).
start	starting value list as used by lmer or glmer.
verbose	passed on to fitting lme4 fitting routines.
mer_only	if TRUE returns the mer part of the out and does not go on to fit the gam object. Useful for speeding up model selection and when interest lies in the log-likelihood of the model.
...	additional arguments to gamm4

Details

fixes bug in gamm4 to allow different optimisers based on gamm4 0.2-5 need to explicitly pass optimiser to optimizeLmer or (twice) to optimizeGlmer, twice since the latter optimisation is a two-stage process

Value

object of class gamm4, but with an aditional call object (like lm, glm etc.) for use in model selection

See Also

gamm4

Index