Package 'FSSgam'

September 28, 2026

Title Full Subsets Multiple Regression Using GAMs

Version 1.2.0

Description Full-subsets information-theoretic approaches are increasingly used to explore predictive power and variable importance when a wide range of candidate predictors are being considered. This package provides functions that can be used to construct, fit, and compare a complete model set of possible ecological or environmental predictors for a given response variable of interest. Models are based on Generalized Additive Models (GAMs) and build on the 'MuMIn' package. Advantages include the capacity to fit more predictors than there are replicates, automatic removal of models with correlated predictors, and support for model sets that include interactions between factors and smooth predictors, as well as smooth-by-smooth interactions via te(). Methods are described in Fisher et al. (2018) <doi:10.1002/ece3.4134>.

Depends R (>= 4.4.0)

License Apache License (== 2.0)

Encoding UTF-8

LazyData true

Imports doSNOW, foreach, mgcv, MuMIn, nnet, parallel, stats, utils

URL https://github.com/beckyfisher/FSSgam_package, https://beckyfisher.github.io/FSSgam_package/, https://beckyfisher.github.io/FSSgam/

BugReports https://github.com/beckyfisher/FSSgam_package/issues

Suggests covr, gamm4, Matrix, testthat (>= 3.2.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Rebecca Fisher [aut, cre], Australian Institute of Marine Science [cph]

Maintainer Rebecca Fisher <r.fisher@aims.gov.au>

Repository CRAN

Date/Publication 2026-09-28 07:30:08 UTC

Contents

`FSSgam-package` *FSSgam: Full Subsets Multiple Regression Using GAMs*

Description

Full-subsets information-theoretic approaches are increasingly used to explore predictive power and variable importance when a wide range of candidate predictors are being considered. This package provides functions that can be used to construct, fit, and compare a complete model set of possible ecological or environmental predictors for a given response variable of interest. Models are based on Generalized Additive Models (GAMs) and build on the 'MuMIn' package. Advantages include the capacity to fit more predictors than there are replicates, automatic removal of models with correlated predictors, and support for model sets that include interactions between factors and smooth predictors, as well as smooth-by-smooth interactions via te(). Methods are described in Fisher et al. (2018) doi:10.1002/ece3.4134.

Details

Full subsets information theoretic approaches are becoming an increasingly popular tool for exploring predictive power and variable importance where a wide range of candidate predictors are being considered.

This package provides simple function(s) that can be used to construct, fit and compare a complete model set of possible ecological or environmental predictors, given a response variable of interest. The function(s) are based on Generalized Additive Models (GAM) and builds on the MuMIn package.

Advantages include the capacity to fit more predictors than there are replicates, automatic removal of models with correlated predictors, and model sets that include interactions between factors and smooth predictors, as well as smooth interactions with other smooths (via te).

The function(s) takes a range of arguments that allow control over the model set being constructed, including specifying cyclic and linear continuous predictors, specification of the smoothing algorithm used and the maximum complexity allowed for smooth terms.

The full subsets analysis can be carried out via one of two alternative methods allowed in the package.

The first is through a direct call to full_subsets_gam (this is the original function). This function both constructs and fits the complete model set, based on the user supplied input. This function requires that all model fits are saved, and is therefore not suitable for extremely large models sets, as these will cause issues with memory. This method may be superseded in future versions of FSSgam, so for any new project please use the second method.

The second method is via a call to generate_model_set followed by a second call to fit_model_set. This pair of functions splits the process of generating the model set from actually fitting and extracting the relevant model data. This method is useful for large model sets, because it allows the model set to be interrogated before fitting and also optionally allows model fit data to not be saved, thus alleviating memory issues.

The use of the function(s) is demonstrated via case studies that highlight how appropriate model sets can be easily constructed, and the broader utility of the approach for exploratory ecology. Please see the case study files on github for usage examples at https://github.com/beckyfisher/FSSgam

Author(s)

Rebecca Fisher (Australian Institute of Marine Science)

Maintainer: Rebecca Fisher <r.fisher@aims.gov.au>

References

Fisher R, Wilson SK, Sin TM, Lee AC, Langlois TJ (2018) A simple function for full-subsets multiple regression in ecology with R. Ecology and Evolution doi:10.1002/ece3.4134

See Also

Useful links:

- https://github.com/beckyfisher/FSSgam_package
- https://beckyfisher.github.io/FSSgam_package/
- https://beckyfisher.github.io/FSSgam/
- Report bugs at https://github.com/beckyfisher/FSSgam_package/issues

Examples

```
library(FSSgam)
```

build_inclusion_mat *build_inclusion_mat*

Description

Supporting function for functions full_subsets_gam and fit_model_set. Not called directly.

Usage

```
build_inclusion_mat(included.vars, formula.list)
```

Arguments

included.vars	A character vector of variables included in the model set
formula.list	A list of model formula, as obtained through generate_model_set

Details

Builds var.inclusion matrix based on the included variables and set of model formula

Value

A matrix of variables included in the model set

Examples

```
included.vars <- c("depth", "complexity")
formula.list <- list(depth = ~1, "depth+complexity" = ~1)
build_inclusion_mat(included.vars, formula.list)
```

case_study1 *Case study 1 dataset*

Description

This is the dataset used for case study 1.

Format

A data frame with 68 rows and 27 variables

Details

This case study examines how management zoning and habitat structure influence abundance and biomass across multiple reef fish functional groups.

case_study2 *Case study 2 dataset*

Description

This is the dataset used for case study 2.

Format

A data frame with 285 rows and 21 variables

Details

This case study reanalyses data from Langlois et al. (2005) to examine whether large reef-associated predators (rock lobster and snapper) structure adjacent soft-sediment communities.

case_study3 *Case study 3 dataset*

Description

This is the dataset used for case study 3.

Format

A data frame with 1110 rows and 12 variables

Details

This case study examines reproductive patterns over multiple temporal scales in two tropical intertidal gastropods.

check_correlations *check_correlations*

Description

generates a correlation matrix among all columns of a data.frame

Usage

```
check_correlations(dat, parallel = FALSE, n.cores = 4)
```

Arguments

dat	the data.frame containing the columns for which a correlation matrix is sought.
parallel	a logical indicating if calculation of the correlation matrix should be done in parallel. Defaults to FALSE.
n.cores	a numeric value indicating the number of cores to utilise if parallel is TRUE.

Details

The function uses cor to calculate the Pearson correlation coefficient among continuous variables, lm to approximate the correlation coefficient among a continuous variable and a factor variable through the call lm(continuous~factor), and nnet to approximate the correlation among factor variables using a multinomial model fit.

Missing values are handled pairwise: each pair of predictors is evaluated on the rows for which both are present. For a factor-factor pair this applies to the intercept-only model as well as the fitted one, so the two deviances the estimate is a ratio of are always computed on the same rows.

Value

a correlation matrix

Examples

```
data(case_study1)
check_correlations(case_study1[, c("depth", "complexity", "ZONE")])
```

check_non_linear_correlations *check_non_linear_correlations*

Description

generates a correlation matrix among all columns of a data.frame

Usage

```
check_non_linear_correlations(dat)
```

Arguments

dat	the data.frame containing the columns for which a correlation matrix is sought.

Details

The function uses gam to estimate a correlation coefficient among continuous variables (continuous~s(continuous), lm to approximate the correlation coefficient between a continuous variable (as response) and a factor variable (as a predictor) through the call lm(continuous~factor), and nnet to approximate the correlation for factor variables as responses using a multinomial model fit through a call to multinom(factor~factor) or (factor~continuous).

Missing values are handled pairwise: each pair of predictors is evaluated on the rows for which both are present, including the intercept-only model the factor-response estimates are scaled by.

Value

an approximate correlation matrix

Note

The resulting "correlation" matrix is asymmetric as the row variable is used as the "response" and the column variable is used as the "predictor". The use of gam may be slightly oversensitive for continuous-continuous correlations and users may wish to increase cov.cutoff. Inspect individual relationships manually. Values are only approximate "correlations" and are in fact the sqrt of the R-square values reported for each of the fitted relationships. Note that the function assumes a gaussian distribution for continuous response variables. Substantial deviations from this assumption will yield spurious "correlation" estimates.

Examples

```
data(case_study1)
check_non_linear_correlations(case_study1[, c("depth", "complexity", "ZONE")])
```

coral_data *Coral_data*

Description

This is the dataset used for the extra examples vignette.

Format

A data frame with 350 rows and 73 variables

extract_mod_dat *extract_mod_dat*

Description

Supporting function for functions full_subsets_gam and fit_model_set. Not called directly.

Usage

```
extract_mod_dat(mod.fit, r2.type. = "r2.lm.est", logLik.fn = NULL)
```

Arguments

mod.fit	A dsm, gam or uGamm fitted model object
r2.type.	The type of r2 to extract. Passed through arguments supplied to fit_model_set
logLik.fn	A function of one argument, a fitted model, returning a single log-likelihood value, or NULL (the default) to read AICc from MuMIn::AICc and BIC from stats::BIC as before. When supplied, AICc and BIC are built from the value it returns, at the degrees of freedom and sample size the default route uses, so only the log-likelihood changes. fit_model_set passes this through, and supplies it itself for a test.fit fitted with one of mgcv's censored families.

Details

Extracts model fit parameters from a dsm, gam or uGamm fitted model object. Called directly, this function reads AICc and BIC from MuMIn::AICc and stats::BIC whatever the fitted family is, so a censored fit gives the value mgcv reports rather than one built from a censored log-likelihood. It is fit_model_set that resolves which log-likelihood a model set is ranked on and passes it here as logLik.fn.

Value

A list of model fit parameters

Examples

```
library(mgcv)
library(MuMIn)
data(case_study1)
fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr"),
           family = tw(), data = case_study1)
extract_mod_dat(fit, r2.type. = "r2")
```

`fit.model.set` *fit.model.set*

Description

Deprecated alias for `fit_model_set`. Retained because `fit.model.set` is the function name cited in Fisher et al. (2018, Ecology and Evolution) and is used by existing downstream code. New code should call `fit_model_set` directly.

Usage

```
fit.model.set(...)
```

Arguments

`...`	Arguments passed on to `fit_model_set`.

Value

See `fit_model_set`.

Examples

```
library(mgcv)
data(case_study1)
use.dat <- case_study1
use.dat$site <- as.factor(use.dat$site)
test.fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr") + s(site, bs = "re"),
                family = tw(), data = use.dat)
model.set <- generate_model_set(
  use.dat = use.dat,
  test.fit = test.fit,
  pred.vars.cont = c("complexity", "depth"),
  pred.vars.fact = "ZONE",
  null.terms = "s(site,bs='re')",
  max.predictors = 2,
  k = 3
)
fit.model.set(model.set, parallel = FALSE)
```

fit_model_set *fit_model_set*

Description

Conducts a full subsets analysis based on gam(m4) using the list generated by a call to generate_model_set

Usage

```
fit_model_set(
  model.set.list,
  max.models = 200,
  save.model.fits = TRUE,
  parallel = FALSE,
  n.cores = 4,
  r2.type = "r2.lm.est",
  report.unique.r2 = FALSE,
  VI.mods = "min.n",
  progress = interactive(),
  logLik.fn = NULL
)
```

Arguments

model.set.list	A list as returned by generate_model_set()
max.models	The total number of models allowed to be fit and the model fits to be saved during fitting. Defaults to 200. If the candidate set is bigger than this value, a warning message will be returned.
save.model.fits	Should all successfully fitted models be saved to list success.models. Defaults to TRUE. Value is overwritten if the number of models n the set is bigger than max.models.
parallel	A logical value indicating if parallel processing should be used. The default is FALSE.
n.cores	An integer indicating the number of cores to use if parallel is TRUE. Defaults to 4.
r2.type	The value to extract from the gam model fit to use as the R squared value. Defaults to r2.lm.est which returns and estimated R squared value based on a linear regression between the observed and predicted values. r2 will return the adjusted R.sq as reported by gam, gamm or gamm4.dev will return the deviance explained as reported by gam or gamm. Note that a gamm fit reaches this only through MuMIn::uGamm: a bare mgcv::gamm fit cannot be used as a test.fit, records no call, and is rejected by generate_model_set. Note gamm4 does not currently return a deviance.

`report.unique.r2`
: Should the r2.vals.unique column of mod.data.out be populated. Defaults to FALSE, which leaves it NA. When TRUE, the null model R2 is subtracted from each model R2 to give the variance explained beyond the terms supplied in null.terms. See the description of mod.data.out under Value for what the column is and is not.

`VI.mods`
: The set of models used to calculate summed variable importance scores. Defaults to 'min.n', which uses only the best n models for each variable (n being the minimum number of models any one predictor is present in, counted over the candidates that were given a criterion). Set to 'all' to use all models in the candidate set instead.

`progress`
: Should a text progress bar be written to the console while models are fitted. Defaults to interactive(), so the bar appears at the console but not in scripts, reports or checks.

`logLik.fn`
: A function of one argument, a fitted model, returning a single log-likelihood value, or NULL (the default). AICc and BIC are ordinarily read from MuMIn::AICc and stats::BIC, which both resolve to the fitted family's own aic slot. Where that slot does not give a log-likelihood the whole ranking is unusable, so this argument allows one to be supplied. When it is, AICc and BIC are built from the value it returns, at the degrees of freedom and sample size the default route uses, so only the log-likelihood changes and every model in the set is scored the same way. Passing function(fit) as.numeric(stats::logLik(fit)) reproduces the criterion of FSSgam 1.1.0 and earlier for any family.

  Two cases are handled without it. A test.fit fitted with one of mgcv's censored families, cnorm or clog, is given a censored log-likelihood computed by the package, with a message saying so, because the value mgcv reports for those families is not built from one; wrap the call in suppressMessages to silence it. A test.fit fitted with a quasi-likelihood such as quasipoisson or quasibinomial stops the call before any candidate is fitted, naming the family. Such a family has no log-likelihood: through gam the criterion is NA, and through uGamm or gamm it is a number read from the PQL working model, which ranks nothing. Supplying this argument is what allows such a set to be fitted and ranked. Note that generate_model_set fits the null model, so on the full_subsets_gam route that one fit precedes the refusal.

  One restriction applies under parallel = TRUE with save.model.fits = FALSE, which is the only combination that evaluates this function on a worker process: a function written at the top level of a script has its environment replaced before it is sent, so any object it refers to and does not define is not found there and every candidate is given no criterion, which stops the call. Write it so that it refers to nothing outside itself, or build it with a constructor – make_ll <- function(k) function(fit) ... k ...; logLik.fn = make_ll(2) – whose environment is sent with it. This is the restriction GitHub issue beckyfisher/FSSgam#10 reports for the family argument, and it has the same cause.

Details

The function constructs and fits a complete model set based on the supplied arguments. for more information see Fisher R, Wilson SK, Sin TM, Lee AC, Langlois TJ (2018) A simple function for

full-subsets multiple regression in ecology with R. Ecology and Evolution doi:10.1002/ece3.4134

Value

A list of the following output files:

mod.data.out - A data.frame that contains the statistics associated with each model fit. This includes AICc and BIC, delta values (e.g. AICc-(min(AICc)), corresponding weight values (Burnham and Anderson 2003), an estimate of the model R2, and a column for each of the included predictor variables containing either 0 (variable not included in the model) or 1 (variable is present in the model). Use of BIC in information theoretic approaches has been heavily criticised because of the inherent assumption of BIC that there is a true model that is represented in the candidate set (Anderson & Burnham 2002). Rather than decide a-priori which model selection tool users should adopt, we supply both as part of the function outputs. To simplify output, only AICc and AICc based model weights, rather than AIC, are included as these are asymptotically equivalent at large sample sizes, and for small sample sizes AICc should be used in any case. AICc and BIC are read from MuMIn::AICc and stats::BIC unless logLik.fn is supplied, or the test.fit was fitted with one of mgcv's censored families, in which cases both are built from that log-likelihood at the degrees of freedom and sample size the default route uses. The delta values, the weights and variable importance follow whichever was used. Calculating R2 values is non-trivial for mixed models, especially non-gaussian cases (and some argue should not be done at all). We have supplied a range of methods for estimating R2 (r2.type), as in our experience a single method rarely performs adequately across all scenarios. A column r2.vals.unique is also present. It is NA unless report.unique.r2 is TRUE, in which case it is the model R2 minus the R2 of the null model, that is, the variance explained beyond the terms supplied in null.terms. It is NA with report.unique.r2 TRUE wherever r2.vals is itself NA, which happens for a gamm test.fit under the default r2.type. Where null.terms is empty the null model's formula is the intercept alone and the column equals r2.vals. This drops every term of the test.fit's formula, a random effect written as s(site, bs = 're') included; that is what null.terms exists to put back. A random structure supplied through a separate argument rather than through the formula, as in uGamm(random = ~(1|site)), is not part of the formula and is retained by the null model either way. The column is a per-model quantity, not a variance partition among terms: with max.predictors greater than one it is the joint contribution of every predictor in that model, and it is a per-predictor contribution only at max.predictors = 1. It is on whatever scale r2.type produced, so a candidate fitting worse than the null on the chosen measure gives a negative value. Values of r2.vals.unique are comparable only within a model set sharing the same null.terms and the same r2.type. Under the default r2.type the R squared is estimated by regressing the response on the fitted values, and for a censored response that response is the recorded bound rather than the latent value, so the estimate is a fit to the censored data. Measured on a simulated set with 20 per cent left censoring, 0.663 against 0.667 computed on the latent response, and the difference grows with the censored fraction.

failed.models - A list of model formula that failed to fit. Ideally the list of failed models should be empty, but when this is not the case interrogating failed.models provides a useful means of troubleshooting. Users can examine which models are not fitting and explore the reasons for this by fitting the failed models outside the full_subsets_gam call based on the listed formula. When a large number of models fail to fit properly it usually indicates poor specification of the initial test.fit or other arguments in the call to full_subsets_gam (such as the inclusion of factor interactions when there are few data within each level of the factor), or that inappropriate variables are being included in the model set.

success.models - A complete list of all successfully fitted model formula. If models were saved, this

can be used for multimodel inference and creating model averaged predictions. otherwise the formula can be used to refit the top model set via a call to update of the test fit: update(test.fit,formula=mod.formula[[l]])

variable.importance - A list containing importance scores for each included predictor. To determine the relative importance of each predictor across the whole model set we summed the wi values for all models containing each variable. The higher the combined weights for an explanatory parameter, the more important it is in the analysis (Burnham & Anderson, 2002). An assumption of the use of summed model weights to infer variable importance is that the number of models in which the different predictors are present is uniform. As our function removes models with correlated predictors, this is not always the case. To overcome this issue, the summed variable.importance scores are the summed weights for the best n models, where n is equal to the minimum number of models any one predictor is present in. If you would like the variable importance scores to be based on all models in the set instead, set VI.mods="all".

Examples

```
library(mgcv)
data(case_study1)
use.dat <- case_study1
use.dat$site <- as.factor(use.dat$site)
test.fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr") + s(site, bs = "re"),
                family = tw(), data = use.dat)
model.set <- generate_model_set(
  use.dat = use.dat,
  test.fit = test.fit,
  pred.vars.cont = c("complexity", "depth"),
  pred.vars.fact = "ZONE",
  null.terms = "s(site,bs='re')",
  max.predictors = 2,
  k = 3
)
fit_model_set(model.set, parallel = FALSE)
```

`full.subsets.gam` *full.subsets.gam*

Description

Deprecated alias for `full_subsets_gam`. Retained because `full.subsets.gam` is used directly in the companion docs repository's published case studies (https://github.com/beckyfisher/FSSgam), e.g. case-study-2.Rmd. New code should call `full_subsets_gam` directly.

Usage

```
full.subsets.gam(...)
```

Arguments

`...`	Arguments passed on to `full_subsets_gam`.

Value

See full_subsets_gam.

Examples

```
library(mgcv)
data(case_study1)
use.dat <- case_study1
use.dat$site <- as.factor(use.dat$site)
test.fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr") + s(site, bs = "re"),
                family = tw(), data = use.dat)
full.subsets.gam(
  use.dat = use.dat,
  test.fit = test.fit,
  pred.vars.cont = c("complexity", "depth"),
  pred.vars.fact = "ZONE",
  null.terms = "s(site,bs='re')",
  max.predictors = 2,
  k = 3
)
```

---

full_subsets_gam *full_subsets_gam*

Description

Conducts a full subsets analysis based on gam(m4). In the most recent version of FSSgam this function is now a wrapper for the two input functions, generate_model_set and fit_model_set. Input arguments are the same as these two underlying functions. calling these underlying functions explicitly is the recommended method for running a full subsets analysis with the FSSgam package because this enables the user to interrogate the candidate model set and the predictor correlation matrix before actually running the analysis.

Usage

```
full_subsets_gam(
  use.dat,
  test.fit,
  pred.vars.cont = NA,
  pred.vars.fact = NA,
  cyclic.vars = NA,
  linear.vars = NA,
  factor.smooth.interactions = pred.vars.fact,
  factor.factor.interactions = FALSE,
  smooth.smooth.interactions = FALSE,
  cov.cutoff = 0.28,
  cor.matrix = NA,
  non.linear.correlations = FALSE,
```

```
  max.predictors = 3,
  k = 5,
  bs.arg = "'cr'",
  null.terms = "",
  null.cov.cutoff = 0.8,
  max.models = 200,
  save.model.fits = TRUE,
  parallel = FALSE,
  n.cores = 4,
  r2.type = "r2.lm.est",
  report.unique.r2 = FALSE,
  VI.mods = "min.n",
  progress = interactive(),
  logLik.fn = NULL,
  factor.interactions,
  smooth.interactions,
  size
)
```

Arguments

use.dat	A data.frame, with columns matching those included in pred.vars.cont and pred.vars.fact, the response variable to be analysed and any other fields required for the analysis (such as random effects, see test.fit). Note that any variables in use.dat that are used in model fits must not contain missing values, as this invalidates comparison via AICc/ BIC. If missing values occur among the predictor variables the function will return an error warning indicating that these rows need to be removed or interpolated.
test.fit	A gam model fitted via a call to gam (mgcv) or uGamm (MuMIn). This can use any of the (preferably continuous) predictors in the call and will be used as a model to update in the fitting of the model set. The test fit must contain the appropriate random effects and call to family (if not gaussian) and if gamm4 should be used, or gamm in the case of a uGamm call (see ?uGamm). Both gamm from mgcv and gamm4 have slightly different features, as well as advantages and disadvantages, thus it is important that the full subsets function is able to deal with test.fit models based on either package. For example gamm4 is based on the lme4 package [Bates, D.M. (2010) lme4: Mixed-Effects Modeling with R. Springer, New York] which allows crossed random effects and avoids issues with PQL for non-gaussian model fits. On the other hand gamm (mgcv), reached through MuMIn::uGamm since a bare mgcv::gamm fit cannot be refitted, is based on nlme which allows correlation structures [Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day], variance structures [Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296], and a broader range of families that are not yet available in lmer (see ?family.mgcv). Models that have no random effects and are based only on gam (mgcv) are best fit via a direct call to gam, rather than using the uGamm wrapper.

pred.vars.cont A character vector indicating the continuous predictors to use. By default all continuous predictors will be fitted using a smoother (but see argument linear.vars). These must match column names in use.dat exactly. If NA is used the function can be run without any smooth predictors.

pred.vars.fact A character vector indicating the factor predictors to use. These must match column names in use.dat exactly. If NA is used the function can be run without any factor predictors.

cyclic.vars NA if there are no cyclic predictors, or if there are cyclic predictors, a character vector containing the names of any of the continuous predictors that should be modelled as cyclic variables. Note that these must also be contained in the pred.vars.cont character vector. Please also note there are issues with bs='cc' and model selection as this uses by default shrinkage. With shrinkage, variables are retained in models but with zero edf, which makes interpretation of AICc and BIC confusing. To account for this always select only the most parsimonious model (that with the fewest parameters), not just that with the lowest AICc. Reported estimated degrees of freedom (edf) in the model output table represent the sum of the edf of the smooth terms plus the number of parametric coefficients. When cyclic variables are included and shrinkage is used, any estimated edf of the smooth terms that are less than 1 are reset to 1 before summing to ensure the the total number of predictors in the model is captured properly.

linear.vars NA if there are no continuous predictors to be treated as linear (not fitted as smooths). Only use this where variables are clearly continuous in nature, but you are confident a linear relationship is valid. It may also be useful for continuous predictors that are not well distributed along the x-axis (ie, sampling was conducted in clumped distances from a feature of interest). Where this is necessary, transformations should be considered where they can be used to theoretically linearize response relationships. Does not need to be contained in vector pred.vars.cont

factor.smooth.interactions

Default is the character vector pred.vars.fact, meaning that all factor predictors will be included as by arguments with all the continuous predictors. If factor.factor.interactions is TRUE, factor interactions variables will also be included as by arguments, yielding higher order interactions up to the specified model max.predictors. If a character vector is supplied, this must specify which factor variables should be included as "by" argument interaction terms with the continuous smooth predictors. If a list is supplied, this must be a named list containing the elements fact.vars, linear.vars, cont.vars, each a character vector indicating what predictors should be used to construct the factor smooth interactions. Note that specified factors, linear predictors and continuous predictors must also be included in their respective character vectors (pred.vars.fact, linear.vars, pred.vars.cont). If specified as NA no factor-continuous predictor interactions will be included.

factor.factor.interactions

A logical value indicating if interactions between factors should be included, or only their main effects. Defaults to FALSE. Note that this can substantially increase the number of models in the candidate set. Not recommended when there are factors with many levels. Alternatively character vector specifying which factor predictors to include as interactions. These must be contained within

pred.vars.fact. If factor.factor.interactions is set to TRUE the function automatically generates hard coded interaction variables up to the maximum number of predictors (see max.predictors below) using combn. New factors are generated by pasting the resulting unique combinations together. This method of generating interaction terms is necessary because smooth-factor interactions are specified as by arguments in calls to gam(m4). Because the full subsets function automatically checks for collinearity, there is no issue with constructing model sets with multiple factor arguments that are higher order factors of each other, as these are invariably collinear and subsequently removed (see cov.cutoff below). A user defined cor.matrix (see cor.matrix) need not include these hard coded interactions: rows and columns for any it does not name are computed from use.dat.

`smooth.smooth.interactions`
: A logical value indicating if the function should include te smooths of second order continuous predictor interactions. If set to TRUE, all continuous predictors will be combined as bivariate calls to te. Alternatively character vector specifying which continuous predictors to include. These must be contained within pred.vars.cont.

`cov.cutoff`
: A numeric value between 0 and 1 indicating the correlation cutoff value to use for excluding collinear models, based on the cor.matrix (see below). The default value is 0.28 (see Graham MH (2003). It is highly recommended to keep this value low, as correlation among predictors can yield spurious results. Note that predictors with a correlation greater than the specified value will still appear in the model set but will never appear in the same model. Including highly correlated predictors can make interpreting variable importance values difficult.

`cor.matrix`
: A user-supplied pairwise correlation matrix, or NA (the default) to compute one from use.dat. When supplied it governs every stage that screens on correlation: which factor-factor interaction columns are built, which te smooth-smooth interaction terms are built, and which assembled candidate models are excluded. It must therefore carry a row and a column for every predictor named in pred.vars.cont, pred.vars.fact and linear.vars; any that are missing are reported by name. The hard coded factor interaction columns that setting factor.factor.interactions causes to be created are the exception, because which of them exist depends on the supplied matrix itself and so cannot be known in advance. Rows and columns for any of those the matrix does not carry are computed from use.dat and appended, leaving every supplied value as supplied. Each dimension is treated separately, so a name supplied as a column and not as a row keeps the column given for it and has only its row computed; because collinearity is screened in both directions, a value supplied in one dimension alone can tighten a screen but never loosen it. Computing them reads the data of every predictor, so this is the one case in which a predictor of a class check_correlations cannot classify has to be named in the supplied matrix along with the interaction columns. When supplied it replaces the automatic estimate rather than overriding it: except in the interaction case above, check_correlations is not called at all, and a predictor of a class it does not accept can be used. By default predictor correlations are evaluated via a call to check_correlations, a function taking a data.frame (containing all predictors) as argument and generating a correlation matrix comprised of: 1) correlation coefficients between all

continuous predictors via a call to cor; 2) approximate correlation values between continuous predictors and factors, as the square root of the R2 value obtained via a call to lm, where the continuous predictor is modelled as a response and the factor variable as a single fixed factor; and 3) approximate correlations values between factor predictors, as the square root of the R2 value obtained via a call multinom (from package nnet, Venables & Ripley 2002). Note that any user constructed pairwise matrix can be passed to the function and used for pairwise exclusion of variables from individual models, subject to two rules. It must be two-dimensional: a matrix, a data.frame, or any other object with two dimensions, so the value Matrix::Matrix returns is accepted while a length-one value of any other class is not, and NA is reserved for the default. And it must not contain NA between two terms that are actually screened against cov.cutoff, which is reported by naming the pairs; an NA on a pair no screen compares is accepted, so which cells matter depends on max.predictors and on the interaction arguments. An NA between a variable named in null.terms and a predictor is accepted, that screen being null.cov.cutoff's rather than cov.cutoff's: where the reverse direction gives a value that value is used, and where neither direction does the pair is computed from use.dat.

`non.linear.correlations`

Set this argument to TRUE if you would like to exclude continuous predictor combinations that are potentially "correlated" through non-linear relationships. See ?check_non_linear_correlations for more details.

`max.predictors` An integer indicating the maximum number of predictors to include in any one model.

`k` An integer indicating the dimension of the basis used to represent the smooth term (see ?s). The default value is 5. Higher values are not recommended unless a complex trend between the response variable and the continuous predictor variables is expected, and the data are sufficient to support this. k can be reduced to as low as 3 where there is trouble obtaining convergence, or sample size is low. Note that this must be set to override the default value, regardless of what k is used in the test.fit

`bs.arg` Specification of the smoother to use, see ?s for more information on smoother provided in gam (mgcv). Note that all continuous predictors specified in pred.vars.cont will be fitted using the same smooth, unless they are also specified as linear.terms or cyclic.vars. Note that any specification of bs in test.fit is discarded.

`null.terms` A character vector indicating the form of any re smooths to be included in gam [e.g. s(site,bs=re)] or any other fixed terms or smooths that the user wants to include in the null model. Use of bs=re is an alternative way of fitting simple random structures that avoids use of PQL and allows a the greater range of families available in gam.mgcv to be used. see ?s and links therein. Note: make sure you use gam instead of uGamm to make sure PQL is not used. To fit a correlation structure, pass MuMIn::uGamm(..., correlation = ...) as the test.fit. A fit produced by mgcv::gamm() directly cannot be used: it records no call, so the candidate models cannot be refitted from it, and generate_model_set stops saying so.

`null.cov.cutoff`

The correlation above which a predictor is dropped for being correlated with a variable named in null.terms. Defaults to 0.8. Terms supplied through null.terms

are forced into every candidate model and are outside the cov.cutoff screen, which covers pred.vars.cont, pred.vars.fact and linear.vars only, so without this a candidate could be arbitrarily strongly correlated with a forced term and still appear in every model in the set. That inflates the variance of the forced term's estimate, which is frequently the term the analysis exists to estimate, and nothing in the output would indicate it. A separate cutoff, with a much looser default than cov.cutoff, because the two screens answer different questions: cov.cutoff decides which predictors may appear together, and this decides which predictor is so nearly a restatement of a forced term that fitting both is not informative. Set it to 1 to admit every predictor whatever its correlation with a forced term. A variable inside a bs='re' smooth is exempt from this screen, however the argument is written – quoted either way, or wrapped in c(). A bs given as a variable rather than a literal is exempt too, since it cannot be read here and screening a grouping factor in error drops a legitimate predictor, while not screening a fixed term in error leaves the behaviour of earlier versions. A random-effect grouping factor is correlated with the predictors measured within it by construction – with null.terms = "s(Location,Site,bs='re')" and Status nested in Location their correlation is 1 – and that is the design of the study rather than collinearity to screen out. The screen applies to fixed forced terms, which compete with a candidate for the same variation. Correlations among the null.terms variables themselves are neither computed nor screened: those terms are forced in by your decision, and dropping one is what must not happen. Where the correlation estimate is asymmetric, as check_non_linear_correlations returns it, both directions are read and the larger is used, which is what the cov.cutoff screen already does. The correlations this screens on are returned by generate_model_set as null.term.correlations, each cell being the value screened on rather than one direction of it, whether or not anything is dropped, so they can be inspected even where no warning is raised; full_subsets_gam does not return them, its output being that of fit_model_set. A supplied cor.matrix is used for any pair it gives a value for, in either dimension, and every other pair is computed from use.dat, which is the ordinary case since a supplied matrix is indexed by predictor and a forced term is not one. A pair the matrix gives no value for, NA in each direction it has, is one of those others, computed rather than read as a correlation of zero, so a cell left empty does not decide that a predictor may be fitted alongside a forced term; a forced term can therefore be part supplied and part computed. Where that computation fails, which is what a predictor of a class check_correlations cannot classify causes, the screen is skipped with a warning rather than the call stopping. A variable named in null.terms that is not a column of use.dat, such as a function or a term written over several columns, is skipped, a correlation not being defined for it.

`max.models` The total number of models allowed to be fit and still save the model fits. Defaults to 200. If the candidate set is bigger than this value, a warning will be returned indicating that model fits will not be saved.

`save.model.fits`

Are the model fits to be saved in the output list? If TRUE this will be overwritten if the model candidate set is bigger than max.models. If FALSE only model output data are saved.

`parallel` A logical value indicating if parallel processing should be used. The default is

FALSE.

`n.cores` An integer indicating the number of cores to use if parallel is TRUE. Defaults to 4.

`r2.type` The value to extract from the gam model fit to use as the R squared value. Defaults to r2.lm.est which returns and estimated R squared value based on a linear regression between the observed and predicted values. r2 will return the adjusted R.sq as reported by gam, gamm or gamm4.dev will return the deviance explained as reported by gam or gamm. Note gamm4 does not currently return a deviance.

`report.unique.r2`
Should the r2.vals.unique column of mod.data.out be populated. Defaults to FALSE, which leaves it NA. When TRUE, the null model R2 is subtracted from each model R2 to give the variance explained beyond the terms supplied in null.terms. See `fit_model_set()` for what the column is and is not.

`VI.mods` The set of models used to calculate summed variable importance scores. Defaults to 'min.n', which uses only the best n models for each variable (n being the minimum number of models any one predictor is present in, counted over the candidates that were given a criterion). Set to 'all' to use all models in the candidate set instead.

`progress` Should a text progress bar be written to the console while models are fitted. Defaults to interactive(), so the bar appears at the console but not in scripts, reports or checks.

`logLik.fn` A function of one argument, a fitted model, returning a single log-likelihood value, or NULL (the default). Passed to fit_model_set; see ?fit_model_set for what it does and for the two cases it is supplied for automatically.

`factor.interactions`
Deprecated. Superseded by factor.factor.interactions; retained only so older code does not break, and will warn if used.

`smooth.interactions`
Deprecated. Superseded by factor.smooth.interactions; retained only so older code does not break, and will warn if used.

`size` Deprecated. Superseded by max.predictors; retained only so older code does not break, and will warn if used.

Details

The function constructs and fits a complete model set based on the supplied arguments. for more information see Fisher R, Wilson SK, Sin TM, Lee AC, Langlois TJ (2018) A simple function for full-subsets multiple regression in ecology with R. Ecology and Evolution doi:10.1002/ece3.4134

Value

A list of the following output files:

mod.data.out - A data.frame that contains the statistics associated with each model fit. This includes AICc and BIC, delta values (e.g. AICc-(min(AICc)), corresponding weight values (Burnham and Anderson 2003), an estimate of the model R2, and a column for each of the included predictor

variables containing either 0 (variable not included in the model) or 1 (variable is present in the model). A column r2.vals.unique is also present, and is NA unless report.unique.r2 is TRUE. This data.frame is the one fit_model_set() produced, passed through unaltered, so see `fit_model_set()` for what the column is and is not. Use of BIC in information theoretic approaches has been heavily criticised because of the inherent assumption of BIC that there is a true model that is represented in the candidate set (Anderson & Burnham 2002). Rather than decide a-priori which model selection tool users should adopt, we supply both as part of the function outputs. To simplify output, only AICc and AICc based model weights, rather than AIC, are included as these are asymptotically equivalent at large sample sizes, and for small sample sizes AICc should be used in any case. Calculating R2 values is non-trivial for mixed models, especially non-gaussian cases (and some argue should not be done at all). We have supplied a range of methods for estimating R2 (r2.type), as in our experience a single method rarely performs adequately across all scenarios.

used.data - A data.frame which is identical to the data.frame initially supplied by the user, but with any hard coded interaction terms appended via cbind.

predictor.correlations - The matrix of estimated predictor correlations returned by the function check_correlations and used for model exclusion based on cov.cutoff

failed.models - A list containing the try-error catch associated with models that failed to fit. Ideally the list of failed models should be empty, but when this is not the case interrogating failed.models provides a useful means of troubleshooting. Users can examine which models are not fitting and explore the reasons for this by fitting the failed models outside the full_subsets_gam call based on the listed formula. When a large number of models fail to fit properly it usually indicates poor specification of the initial test.fit or other arguments in the call to full_subsets_gam (such as the inclusion of factor interactions when there are few data within each level of the factor), or that inappropriate variables are being included in the model set.

success.models - A complete list of all successfully fitted models. This can be used for multimodel inference and creating model averaged predictions.

variable.importance - A list containing importance scores for each included predictor. To determine the relative importance of each predictor across the whole model set we summed the wi values for all models containing each variable. The higher the combined weights for an explanatory parameter, the more important it is in the analysis (Burnham & Anderson, 2002). An assumption of the use of summed model weights to infer variable importance is that the number of models in which the different predictors are present is uniform. As our function removes models with correlated predictors, this is not always the case. To overcome this issue, the summed variable.importance scores are the summed weights for the best n models, where n is equal to the minimum number of models any one predictor is present in.

Examples

```
library(mgcv)
data(case_study1)
use.dat <- case_study1
use.dat$site <- as.factor(use.dat$site)
test.fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr") + s(site, bs = "re"),
                family = tw(), data = use.dat)
full_subsets_gam(
  use.dat = use.dat,
  test.fit = test.fit,
  pred.vars.cont = c("complexity", "depth"),
```

```
  pred.vars.fact = "ZONE",
  null.terms = "s(site,bs='re')",
  max.predictors = 2,
  k = 3
)
```

`generate.model.set` *generate.model.set*

Description

Deprecated alias for `generate_model_set`. Retained because `generate.model.set` is the function name cited in Fisher et al. (2018, Ecology and Evolution) and is used by existing downstream code. New code should call `generate_model_set` directly.

Usage

```
generate.model.set(...)
```

Arguments

`...`	Arguments passed on to `generate_model_set`.

Value

See `generate_model_set`.

Examples

```
library(mgcv)
data(case_study1)
use.dat <- case_study1
use.dat$site <- as.factor(use.dat$site)
test.fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr") + s(site, bs = "re"),
                family = tw(), data = use.dat)
model.set <- generate.model.set(
  use.dat = use.dat,
  test.fit = test.fit,
  pred.vars.cont = c("complexity", "depth"),
  pred.vars.fact = "ZONE",
  null.terms = "s(site,bs='re')",
  max.predictors = 2,
  k = 3
)
```

generate_model_set *generate_model_set*

Description

Generate a complete full subsets model set for analysis based on gam(m4) via a call to fit_model_set.

Usage

```
generate_model_set(
  use.dat,
  test.fit,
  pred.vars.cont = NA,
  pred.vars.fact = NA,
  cyclic.vars = NA,
  linear.vars = NA,
  factor.smooth.interactions = pred.vars.fact,
  factor.factor.interactions = FALSE,
  smooth.smooth.interactions = FALSE,
  cov.cutoff = 0.28,
  cor.matrix = NA,
  non.linear.correlations = FALSE,
  max.predictors = 3,
  k = 5,
  bs.arg = "'cr'",
  null.terms = "",
  null.cov.cutoff = 0.8
)
```

Arguments

use.dat	A data.frame, with columns matching those included in pred.vars.cont and pred.vars.fact, the response variable to be analysed and any other fields required for the analysis (such as random effects, see test.fit). Note that any variables in use.dat that are used in model fits must not contain missing values, as this invalidates comparison via AICc/ BIC. If missing values occur among the predictor variables the function will return an error warning indicating that these rows need to be removed or interpolated.
test.fit	A gam model fitted via a call to gam (mgcv) or uGamm (MuMIn). This can use any of the (preferably continuous) predictors in the call and will be used as a model to update in the fitting of the model set. The test fit must contain the appropriate random effects and call to family (if not gaussian) and if gamm4 should be used, or gamm in the case of a uGamm call (see ?uGamm). Both gamm from mgcv and gamm4 have slightly different features, as well as advantages and disadvantages, thus it is important that the full subsets function is able to deal with test.fit models based on either package. For example gamm4 is based on the lme4 package [Bates, D.M. (2010) lme4: Mixed-Effects Modeling

with R. Springer, New York] which allows crossed random effects and avoids issues with PQL for non-gaussian model fits. On the other hand gamm (mgcv), reached through MuMIn::uGamm since a bare mgcv::gamm fit cannot be refitted, is based on nlme which allows correlation structures [Box, G.E.P., Jenkins, G.M., and Reinsel G.C. (1994) "Time Series Analysis: Forecasting and Control", 3rd Edition, Holden-Day], variance structures [Pinheiro, J.C. and Bates., D.M. (1996) "Unconstrained Parametrizations for Variance-Covariance Matrices", Statistics and Computing, 6, 289-296], and a broader range of families that are not yet available in lmer (see ?family.mgcv). Models that have no random effects and are based only on gam (mgcv) are best fit via a direct call to gam, rather than using the uGamm wrapper.

`pred.vars.cont` A character vector indicating the continuous predictors to use. By default all continuous predictors will be fitted using a smoother (but see argument linear.vars). These must match column names in use.dat exactly. If NA is used the function can be run without any smooth predictors.

`pred.vars.fact` A character vector indicating the factor predictors to use. These must match column names in use.dat exactly. If NA is used the function can be run without any factor predictors.

`cyclic.vars` NA if there are no cyclic predictors, or if there are cyclic predictors, a character vector containing the names of any of the continuous predictors that should be modelled as cyclic variables. Note that these must also be contained in the pred.vars.cont character vector. Please also note there are issues with bs='cc' and model selection as this uses by default shrinkage. With shrinkage, variables are retained in models but with zero edf, which makes interpretation of AICc and BIC confusing. To account for this always select only the most parsimonious model (that with the fewest parameters), not just that with the lowest AICc. Reported estimated degrees of freedom (edf) in the model output table represent the sum of the edf of the smooth terms plus the number of parametric coefficients. When cyclic variables are included and shrinkage is used, any estimated edf of the smooth terms that are less than 1 are reset to 1 before summing to ensure the the total number of predictors in the model is captured properly.

`linear.vars` NA if there are no continuous predictors to be treated as linear (not fitted as smooths). Only use this where variables are clearly continuous in nature, but you are confident a linear relationship is valid. It may also be useful for continuous predictors that are not well distributed along the x-axis (ie, sampling was conducted in clumped distances from a feature of interest). Where this is necessary, transformations should be considered where they can be used to theoretically linearize response relationships. Does not need to be contained in vector pred.vars.cont

`factor.smooth.interactions`

Default is the character vector pred.vars.fact, meaning that all factor predictors will be included as by arguments with all the continuous predictors. If factor.factor.interactions is TRUE, factor interactions variables will also be included as by arguments, yielding higher order interactions up to the specified model max.predictors. If a character vector is supplied, this must specify which factor variables should be included as "by" argument interaction terms with the continuous smooth predictors. If a list is supplied, this must be a named list

containing the elements fact.vars, linear.vars, cont.vars, each a character vector indicating what predictors should be used to construct the factor smooth interactions. Note that specified factors, linear predictors and continuous predictors must also be included in their respective character vectors (pred.vars.fact, linear.vars, pred.vars.cont). If specified as NA no factor-continuous predictor interactions will be included.

`factor.factor.interactions`

A logical value indicating if interactions between factors should be included, or only their main effects. Defaults to FALSE. Note that this can substantially increase the number of models in the candidate set. Not recommended when there are factors with many levels. Alternatively character vector specifying which factor predictors to include as interactions. These must be contained within pred.vars.fact. If factor.factor.interactions is set to TRUE the function automatically generates hard coded interaction variables up to the maximum number of predictors (see max.predictors below) using combn. New factors are generated by pasting the resulting unique combinations together. This method of generating interaction terms is necessary because smooth-factor interactions are specified as by arguments in calls to gam(m4). Because the full subsets function automatically checks for collinearity, there is no issue with constructing model sets with multiple factor arguments that are higher order factors of each other, as these are invariably collinear and subsequently removed (see cov.cutoff below). A user defined cor.matrix (see cor.matrix) need not include these hard coded interactions: rows and columns for any it does not name are computed from use.dat.

`smooth.smooth.interactions`

A logical value indicating if the function should include te smooths of second order continuous predictor interactions. If set to TRUE, all continuous predictors will be combined as bivariate calls to te. Alternatively character vector specifying which continuous predictors to include. These must be contained within pred.vars.cont.

`cov.cutoff` A numeric value between 0 and 1 indicating the correlation cutoff value to use for excluding collinear models, based on the cor.matrix (see below). The default value is 0.28 (see Graham MH (2003). It is highly recommended to keep this value low, as correlation among predictors can yield spurious results. Note that predictors with a correlation greater than the specified value will still appear in the model set but will never appear in the same model. Including highly correlated predictors can make interpreting variable importance values difficult.

`cor.matrix` A user-supplied pairwise correlation matrix, or NA (the default) to compute one from use.dat. When supplied it governs every stage that screens on correlation: which factor-factor interaction columns are built, which te smooth-smooth interaction terms are built, and which assembled candidate models are excluded. It must therefore carry a row and a column for every predictor named in pred.vars.cont, pred.vars.fact and linear.vars; any that are missing are reported by name. The hard coded factor interaction columns that setting factor.factor.interactions causes to be created are the exception, because which of them exist depends on the supplied matrix itself and so cannot be known in advance. Rows and columns for any of those the matrix does not carry are computed from use.dat and appended, leaving every supplied value as supplied.

Each dimension is treated separately, so a name supplied as a column and not as a row keeps the column given for it and has only its row computed; because collinearity is screened in both directions, a value supplied in one dimension alone can tighten a screen but never loosen it. Computing them reads the data of every predictor, so this is the one case in which a predictor of a class check_correlations cannot classify has to be named in the supplied matrix along with the interaction columns. When supplied it replaces the automatic estimate rather than overriding it: except in the interaction case above, check_correlations is not called at all, and a predictor of a class it does not accept can be used. By default predictor correlations are evaluated via a call to check_correlations, a function taking a data.frame (containing all predictors) as argument and generating a correlation matrix comprised of: 1) correlation coefficients between all continuous predictors via a call to cor; 2) approximate correlation values between continuous predictors and factors, as the square root of the R2 value obtained via a call to lm, where the continuous predictor is modelled as a response and the factor variable as a single fixed factor; and 3) approximate correlations values between factor predictors, as the square root of the R2 value obtained via a call multinom (from package nnet, Venables & Ripley 2002). Note that any user constructed pairwise matrix can be passed to the function and used for pairwise exclusion of variables from individual models, subject to two rules. It must be two-dimensional: a matrix, a data.frame, or any other object with two dimensions, so the value Matrix::Matrix returns is accepted while a length-one value of any other class is not, and NA is reserved for the default. And it must not contain NA between two terms that are actually screened against cov.cutoff, which is reported by naming the pairs; an NA on a pair no screen compares is accepted, so which cells matter depends on max.predictors and on the interaction arguments. An NA between a variable named in null.terms and a predictor is accepted, that screen being null.cov.cutoff's rather than cov.cutoff's: where the reverse direction gives a value that value is used, and where neither direction does the pair is computed from use.dat.

`non.linear.correlations`
: Set this argument to TRUE if you would like to exclude continuous predictor combinations that are potentially "correlated" through non-linear relationships. See ?check_non_linear_correlations for more details.

`max.predictors` An integer indicating the maximum number of predictors to include in any one model.

`k` An integer indicating the dimension of the basis used to represent the smooth term (see ?s). The default value is 5. Higher values are not recommended unless a complex trend between the response variable and the continuous predictor variables is expected, and the data are sufficient to support this. k can be reduced to as low as 3 where there is trouble obtaining convergence, or sample size is low. Note that this must be set to override the default value, regardless of what k is used in the test.fit

`bs.arg` Specification of the smoother to use, see ?s for more information on smoother provided in gam (mgcv). Note that all continuous predictors specified in pred.vars.cont will be fitted using the same smooth, unless they are also specified as linear.terms or cyclic.vars. Note that any specification of bs in test.fit is discarded.

`null.terms` A character vector indicating the form of any re smooths to be included in gam [e.g. s(site,bs=re)] or any other fixed terms or smooths that the user wants to include in the null model. Use of bs=re is an alternative way of fitting simple random structures that avoids use of PQL and allows a the greater range of families available in gam.mgcv to be used. see ?s and links therein. Note: make sure you use gam instead of uGamm to make sure PQL is not used. To fit a correlation structure, pass MuMIn::uGamm(..., correlation = ...) as the test.fit. A fit produced by mgcv::gamm() directly cannot be used: it records no call, so the candidate models cannot be refitted from it, and generate_model_set stops saying so.

`null.cov.cutoff`

The correlation above which a predictor is dropped for being correlated with a variable named in null.terms. Defaults to 0.8. Terms supplied through null.terms are forced into every candidate model and are outside the cov.cutoff screen, which covers pred.vars.cont, pred.vars.fact and linear.vars only, so without this a candidate could be arbitrarily strongly correlated with a forced term and still appear in every model in the set. That inflates the variance of the forced term's estimate, which is frequently the term the analysis exists to estimate, and nothing in the output would indicate it. A separate cutoff, with a much looser default than cov.cutoff, because the two screens answer different questions: cov.cutoff decides which predictors may appear together, and this decides which predictor is so nearly a restatement of a forced term that fitting both is not informative. Set it to 1 to admit every predictor whatever its correlation with a forced term. A variable inside a bs='re' smooth is exempt from this screen, however the argument is written – quoted either way, or wrapped in c(). A bs given as a variable rather than a literal is exempt too, since it cannot be read here and screening a grouping factor in error drops a legitimate predictor, while not screening a fixed term in error leaves the behaviour of earlier versions. A random-effect grouping factor is correlated with the predictors measured within it by construction – with null.terms = "s(Location,Site,bs='re')" and Status nested in Location their correlation is 1 – and that is the design of the study rather than collinearity to screen out. The screen applies to fixed forced terms, which compete with a candidate for the same variation. Correlations among the null.terms variables themselves are neither computed nor screened: those terms are forced in by your decision, and dropping one is what must not happen. Where the correlation estimate is asymmetric, as check_non_linear_correlations returns it, both directions are read and the larger is used, which is what the cov.cutoff screen already does. The correlations this screens on are returned by generate_model_set as null.term.correlations, each cell being the value screened on rather than one direction of it, whether or not anything is dropped, so they can be inspected even where no warning is raised; full_subsets_gam does not return them, its output being that of fit_model_set. A supplied cor.matrix is used for any pair it gives a value for, in either dimension, and every other pair is computed from use.dat, which is the ordinary case since a supplied matrix is indexed by predictor and a forced term is not one. A pair the matrix gives no value for, NA in each direction it has, is one of those others, computed rather than read as a correlation of zero, so a cell left empty does not decide that a predictor may be fitted alongside a forced term; a forced term can therefore be part supplied and part

computed. Where that computation fails, which is what a predictor of a class check_correlations cannot classify causes, the screen is skipped with a warning rather than the call stopping. A variable named in null.terms that is not a column of use.dat, such as a function or a term written over several columns, is skipped, a correlation not being defined for it.

Details

The function constructs a complete model set based on the supplied arguments. for more information see Fisher R, Wilson SK, Sin TM, Lee AC, Langlois TJ (2018) A simple function for full-subsets multiple regression in ecology with R. Ecology and Evolution doi:10.1002/ece3.4134

Value

A list of the following output files:

n.mods - The number of candidate models generated, equal to length(mod.formula).

predictor.correlations - The matrix of estimated predictor correlations returned by the function check_correlations and used for model exclusion based on cov.cutoff

null.term.correlations - A matrix of the correlations between each variable named in null.terms and each candidate predictor, used for model exclusion based on null.cov.cutoff. Rows are the null.terms variables and columns the predictors; a variable whose correlations could not be computed and which the supplied cor.matrix gave no value for has no row, the warning naming it instead. NULL where null.terms is empty, where it names only random-effect grouping factors or no column of use.dat, where every predictor is itself a null.terms variable, or where the computation failed and the supplied cor.matrix gave no value for any pair, so that there is nothing to report. The element is always present in the returned list; it is its value that is NULL. Correlations among the null.terms variables are not included, being neither computed nor screened.

mod.formula - A named list containing the model formula that were generated (and will be fitted by fit_model_set). The names are the candidate model names used in the modname column of fit_model_set's output.

used.data - A data.frame which is identical to the data.frame initially supplied by the user, but with any hard coded interaction terms appended via cbind.

test.fit - The test.fit supplied by the user, passed through so that fit_model_set can update it.

included.vars - A character vector of the predictors included in the model set, used to build the variable inclusion columns of fit_model_set's output.

Examples

```
library(mgcv)
data(case_study1)
use.dat <- case_study1
use.dat$site <- as.factor(use.dat$site)
test.fit <- gam(Herbivore.abundance ~ s(depth, k = 3, bs = "cr") + s(site, bs = "re"),
                family = tw(), data = use.dat)
model.set <- generate_model_set(
  use.dat = use.dat,
  test.fit = test.fit,
  pred.vars.cont = c("complexity", "depth"),
```

```
  pred.vars.fact = "ZONE",
  null.terms = "s(site,bs='re')",
  max.predictors = 2,
  k = 3
)
```

---

wi *wi*

Description

Supporting function for functions full_subsets_gam and fit_model_set. Not called directly.

Usage

```
wi(AIC.vals)
```

Arguments

AIC.vals	vector of AICc, AIC or BIC values

Details

Calculates Akaike weight values from a vector of AICc, AIC or BIC values

Value

A vector of Akaike weights

Examples

```
wi(c(100, 102, 105, 110))
```

Index