# Package 'INDperform'

January 9, 2020

**Title** Evaluation of Indicator Performances for Assessing Ecosystem States

**Type** Package

**Version** 0.2.2

**Description** An implementation of the 7-step approach suggested by Otto et al. (2018) <doi:10.1016/j.ecolind.2017.05.045> to validate ecological state indicators and to select a suite of complimentary and well performing indicators. This suite can be then used to assess the current state of the system in comparison to a reference period. However, the tools in this package are very generic and can be used to test any type of indicator (e.g. social or economic indicators).

**URL** https://github.com/saskiaotto/INDperform

**BugReports** https://github.com/SaskiaAOtto/INDperform/issues

**License** GPL

**LazyData** true

**Encoding** UTF-8

**RoxygenNote** 7.0.2

**Depends** R (>= 3.5.0)

**Imports** cowplot (>= 1.0.0), dplyr (>= 0.8.3), grDevices (>= 3.5.0), ggplot2 (>= 3.2.1), htmlwidgets (>= 1.5.1), jsonlite (>= 1.6), magrittr (>= 1.5), mgcv (>= 1.8-26), nlme (>= 3.1-137), purrr (>= 0.3.3), RColorBrewer (>= 1.1-2), rhandsontable (>= 0.3.7), rlang (>= 0.4.2), shiny (>= 1.4.0), stringr (>= 1.4.0), tibble (>= 2.1.3), tidyr (>= 1.0.0), vegan (>= 2.5-6)

**Suggests** doParallel (>= 1.0.15), ggdendro (>= 0.1-20), gridExtra (>= 2.3), knitr (>= 1.26), parallel (>= 3.5.2), pbapply (>= 1.4-2), testthat (>= 2.3.1), tripack (>= 1.3-8)

**NeedsCompilation** no

**Author** Saskia A. Otto [aut, cre] (<https://orcid.org/0000-0001-7780-1322>), Rene Plonus [aut], Steffen Funk [aut], Alexander Keth [aut]

**Maintainer** Saskia A. Otto <saskia.a.otto@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-01-09 12:30:14 UTC

# R topics documented:

---

`INDperform-package` *INDperform: Performance validation of ecological state indicators*

---

## Description

INDperform provides an implementation of the 7-step approach suggested by Otto *et al.* (2018) to validate ecological state indicators and to select a suite of complimentary and well performing indicators. This suite can be then used to assess the current state of the system in comparison to a reference period.

## Details

The package builds upon the tidy data principles and offers functions to

- identify temporal indicator changes,
- model relationships to pressures while taking non-linear responses and temporal autocorrelation into account, and to
- quantify the robustness of these models.

These functions can be executed on any number of indicators and pressures. Based on these analyses and a scoring scheme for selected criteria the individual performances can be quantified, visualized, and compared. The combination of tools provided in this package can significantly help making state indicators operational under given management schemes such as the EU Marine Strategy Framework Directive.

## Usage

INDperform offers function that can be applied individually to some extent but mostly build upon each other to follow the 7-step approach. They can be grouped into 3 broad categories. For demonstration purposes the package provides a dataset of food web indicators and pressure variables in the Central Baltic Sea (modified from Otto *et al.*, 2018).

### 1. Validation of IND performances

The following functions implement the first five steps of the 7-step validation approach and model each IND as a function of time or a single pressure variable using Generalized Additive Models (GAMs) (based on the `mgcv` package):

- `model_trend`
- `ind_init`
- `model_gam`
- `model_gamm`
- `select_model`
- `merge_models`
- `calc_deriv`

- select_interaction
- test_interaction

To show the model diagnostics or complete model results use the functions:

- plot_diagnostics
- plot_trend
- plot_model

**2. Scoring IND performance based on model output**

Among the 16 common indicator selection criteria, five criteria relate to the indicators‘ performances and require time series for their evaluation, i.e.

8. Development reflects ecosystem change caused by variation in manageable pressure(s)

9. Sensitive or responsive to pressures

10. Robust, i.e. responses in a predictive fashion, and statistically sound

11. Links to management measures (responsiveness and specificity)

12. Relates where appropriate to other indicators but is not redundant

In this package, the scoring scheme for these criteria as proposed by Otto *et al.* (2018) serves as basis for the quantification of the IND performance (see the scoring template table crit_scores_tmpl). Sensitivity (criterion 9) and robustness (criterion 10) are specified into more detailed sub-criteria to allow for quantification based on statistical models and rated individually for every potential pressure that might affect the IND directly or indirectly.

However, the scoring template can easily be adapted to any kind of state indicator and management scheme by modifying the scores, the weighting of scores or by removing (sub)criteria.

The following functions relate to the indicator performance scoring (used in this order):

- scoring
- expect_resp
- summary_sc
- plot_spiechart

For examining redundancies and selecting robust indicator suites use (in that order):

- dist_sc
- clust_sc
- plot_clust_sc

**3. Assessment of current state status**

Two approaches based on trajectories in state space to determine the current state of the system in comparison to an earlier period as reference using the selected IND suite (state space = n-dimensional space of possible locations of IND variables)

1. Calculation of the Euclidean distance in state space of any dimensionality between each single year (or any other time step used) and a defined reference year:

- statespace_ed
- plot_statespace_ed

2. Given the identification of a reference domain in state space, more recent observations might lie within or outside this domain. The convex hull is a multivariate measure derived from computational geometry representing the smallest convex set containing all the reference points in Euclidean plane or space. For visualization, only 2 dimensions considered (dimension reduction through e.g. Principal Component Analysis suggested).

- statespace_ch
- plot_statespace_ch

## Author(s)

**Maintainer**: Saskia A. Otto <saskia.a.otto@gmail.com> (ORCID)

Authors:

- Rene Plonus
- Steffen Funk
- Alexander Keth

## References

To learn more about the framework, see

Otto, S.A., Kadin, M., Casini, M., Torres, M.A., Blenckner, T. (2018) A quantitative framework for selecting and validating food web indicators. *Ecological Indicators*, 84: 619-631, doi: https://doi.org/10.1016/j.ecolind.2017.

## See Also

Useful links:

- https://github.com/saskiaotto/INDperform
- Report bugs at https://github.com/SaskiaAOtto/INDperform/issues

---

# all_results_ex *Output tibble after applying all IND~pressure modeling functions*

---

## Description

This is an example output tibble based on the Central Baltic Sea food web indicator demonstration data after applying the calc_deriv and the test_interaction functions on the merge_models_ex tibble.

## Usage

```
all_results_ex
```

## Format

A data frame with 84 rows and 31 variables:

`id` Numerical IDs for the IND~press combinations.

`ind` Indicator names.

`press` Pressure names.

`model_type` Specification of the model type; at this stage containing only "gam" (Generalized Additive Model).

`corrstruc` Specification of the correlation structure; at this stage containing only "none".

`aic` AIC of the fitted models

`edf` Estimated degrees of freedom for the model terms.

`p_val` The p values for the smoothing term (the pressure).

`signif_code` The significance codes for the p-values.

`r_sq` The adjusted r-squared for the models. Defined as the proportion of variance explained, where original variance and residual variance are both estimated using unbiased estimators. This quantity can be negative if your model is worse than a one parameter constant model, and can be higher for the smaller of two nested models.

`expl_dev` The proportion of the null deviance explained by the models.

`nrmse` Absolute values of the root mean square error normalized by the standard deviation (NRMSE) and corrected for the prior (log) transformation.

`ks_test` The p-values from a Kolmogorov-Smirnov Test applied on the model residuals to test for normal distribution. P-values > 0.05 indicate normally distributed residuals.

`tac` logical; indicates whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC.

`pres_outlier` A list-column with outliers identified for each model (i.e. Cook‘s distance > 1). The indices present the position in the training data, including NAs.

`excl_outlier` A list-column listing all outliers per model that have been excluded in the GAM fitting

`model` A list-column of IND~press-specific gam objects.

`prop` The proportion of the observed pressure range where the indicator shows a response (see the last section in *Details*)

`zero_in_conf` A list-column of logical vectors indicating for every pressure value (in press_seq) whether the slope of the indicator response at that pressure value is within the confidence interval, i.e. is zero.

`zic_start_end` A list-column of logical vectors indicating for every pressure value (in press_seq) whether the slope is considered as zero for the proportion calculation (see see the last section in *Details*).

`press_seq` A list-column with sequences of 100 evenly spaced pressure values.

`pred` A list-column with the predicted indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

`pred_ci_up` A list-column with the upper confidence limit of the bootstrapped predictions.

`pred_ci_low` A list-column with the lower confidence limit of the bootstrapped predictions.

`deriv1` A list-column with the first derivatives of the indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

`deriv1_ci_up` A list-column with the upper confidence limit of the bootstrapped first derivatives.

`deriv1_ci_low` A list-column with the lower confidence limit of the bootstrapped first derivatives.

`adj_n_boot` The number of successful bootstrap samples that was actually used for calculating the mean and confidence intervals of the predicted indicator response and the derivative.

`boot_error` A list-column capturing potential error messages that occurred as side effects when refitting the GAM(M)s on each bootstrap sample.

`interaction` logical; if TRUE, at least one thresh_gam performs better than its corresponding gam based on the leave-one-out cross-validation.

`thresh_var` A list-column with the threshold variables of the better performing thresh_models.

`thresh_models` A list-column with nested lists containing the better performing thresh_models.

`thresh_error` A list-column capturing potential error messages that occurred as side effects when fitting each threshold GAMs and performing the LOOCV.

`tac_in_thresh` logical vector; indicates for every listed thresh_model whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC.

---

`approx_deriv` *Alternative method for confidence interval approximations of derivatives*

---

## Description

`approx_deriv` implements a crude approximation for the uncertainty around the first derivatives. It should be used complementary to the conditional bootstrap, if problems with GAMM fittings occur (see `calc_deriv`).

## Usage

```
approx_deriv(init_tbl, mod_tbl, ci_prop_se)
```

## Arguments

| | |
|---|---|
| `init_tbl` | The output tibble of the `ind_init` function. |
| `mod_tbl` | A model output tibble from `model_gam`, `select_model` or `merge_models` representing the best model for each IND~pressure pair. |
| `ci_prop_se` | A conversion factor for approximating derivative CIs in the 'approx_method'; it is multiplied with the ratio between s.e. and mean fitted values of the smoothing curve to represent some level of uncertainty around the slope proportional to the uncertainty in the smoothing curve. Default is 25, which is a compromise representing fairly well the results obtained for the GAMs from the conditional bootstrap. |

## Details

In this approach derivatives are calculated for the original smoother and some level of uncertainty (not exactly the confidence intervals) is estimated based on the standard error (s.e.) of the smoother. The same proportion of error (estimated as the ratio s.e./fitted mean) is adopted for the maximal slope of the derivative and then kept constant across the entire curve. As this results in much smaller uncertainty ranges, a conversion (or multiplication) factor is implemented to allow modifications of the error proportion. The default of 25 is a compromise representing fairly well the results obtained for the GAMs from the conditional bootstrap.

## Value

The function returns the input model tibble with the following 4 columns added

`press_seq` A list-column with sequences of 100 evenly spaced pressure values (with the length of the time series).

`deriv1` A list-column with the first derivatives of the indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

`deriv1_ci_up` A list-column with the upper confidence limit of the bootstrapped first derivatives(for the 100 equally spaced pressure values).

`deriv1_ci_low` A list-column with the lower confidence limit of the bootstrapped first derivatives(for the 100 equally spaced pressure values).

## See Also

the wrapper function `calc_deriv`

## Examples

```
# Using some models of the Baltic Sea demo data
init_tbl <- ind_init_ex[ind_init_ex$id %in% c(5,9,75), ]
mod_tbl <- merge_models_ex[merge_models_ex$id  %in% c(5,9,75), ]
deriv_tbl <- approx_deriv(init_tbl, mod_tbl, ci_prop_se = 25)
```

---

`calc_deriv` *Calculate the derivatives of the IND response functions and the proportion of pressure range in which the response is significant*

---

## Description

In the case of non-linear IND responses to pressures first derivatives of the response curve are calculated to identify whether the IND ceases to respond at the lower or upper end of the pressure range (relevant for sub-crit. 9.2). `calc_deriv` serves as a wrapper function that filters first the model input tibble and applies as default method the 'conditional bootstrap' (see the function `cond_boot`). It calculates from the computed bootstrapped derivatives and confidence intervals the proportion of pressure range in which the IND shows a significant change. The implementation of the 'conditional bootstrap' for Generalized Additive Mixed Models (GAMMs) has some caveats (see *Details*), which is why we implemented additionally a rather quick-and-dirty approach

(through the approx_deriv function). In the 'approx_deriv' method derivatives are calculated directly from the smoothing curve of the original method. The confidence intervals are then just approximated from the standard errors of the smoothing curve. For more informations on this method see approx_deriv.

## Usage

```
calc_deriv(
  init_tbl,
  mod_tbl,
  edf_filter = 1.5,
  sign_level = 0.05,
  excl_outlier = FALSE,
  method = "cond_boot",
  n_boot = 200,
  ci_boot = 0.95,
  ci_prop_se = 25,
  par_comp = FALSE,
  no_clust = NULL,
  seed = NULL
)
```

## Arguments

| | |
|---|---|
| init_tbl | The output tibble of the ind_init function. |
| mod_tbl | A model output tibble from model_gam, select_model or merge_models representing the best model for each IND~pressure pair. |
| edf_filter | The minimum edf value at which derivatives are calculated for the respective model. The default is set to 1.5. |
| sign_level | Significance level for selecting models on which to calculate the derivatives. Only models with a p value smaller than the sign_level will be selected; the default is 0.05. |
| excl_outlier | logical; if TRUE, the outliers excluded in the original models will be also excluded in the bootstrapped models. |
| method | Method for calculating derivatives and CI; can be either 'conditional bootstrap' (default) or 'approx_deriv'. |
| n_boot | Number of bootstraps. Select n_boot so that (n_boot - (n_boot *ci)) / 2 will be an integer. Otherwise, the function will increase n_boot automatically. The default is set to 200. |
| ci_boot | Confidence interval of the bootstrapped smoothing functions and their derivatives. Must be between 0 and 1, default is 0.95. |
| ci_prop_se | A conversion factor for approximating derivative CIs in the 'approx_method'; it is multiplied with the ratio between s.e. and mean fitted values of the smoothing curve to represent some level of uncertainty around the slope proportional to the uncertainty in the smoothing curve. Default is 25, which is a compromise representing fairly well the results obtained for the GAMs from the conditional bootstrap. |

| | |
|---|---|
| par_comp | logical; if TRUE, the conditional bootstrap will be processed in parallel using several clusters, which can speed up the iteration process depending on the number of n_boot, models to bootstrap and number of processor cores. |
| no_clust | Number of clusters ("workers") for the parallel computation, with one cluster per core. If no_clust is set to NULL default, the number of clusters is set as the numbers of available cores – 1. |
| seed | A single value, interpreted as an integer, which specifies the seed of the random number generator (RNG) state for reproducibility. Due to the work splitting in the parallel computation, RNG streams are not comparable with the stream under serial computation. To reproduce results use the same type of computation with the same seed and number of clusters. |

## Details

In the case of non-linear IND responses to pressures first derivatives of the response curve are calculated to identify whether the IND ceases to respond at the lower or upper end of the pressure range (relevant for sub-crit. 9.2).

**First Derivative:**

The first derivative of a smoothing function i.e. s‘(x), represents the instantaneous rate of change of a dependent variable y to that of the independent variable x. It tells us whether a function is increasing or decreasing, and by how much. This information is reflected by the slope of the tangent line to a point x on the graph of a function. A positive slope tells us that, as x increases, s(x) also increases. Derivatives can be applied to any function, even one as potentially complex as a fitted spline function, by using the method of finite differences (Trenkel and Rochet, 2009) as done in the cond_boot function: The local first derivatives of the fitted GAM(M) time series is estimated as the difference between fitted values at time step t and t + d divided by d, where d is a small value, e.g. 1E-7.

**Confidence intervals (CI) based on a conditional bootstrap:**

By calculating approximate confidence intervals for s‘(x), it may be determined whether or not apparent IND changes are statistically significant, i.e. whether the non-zero estimate obtained for the slope is distinguishable from zero given the uncertainty in the estimate (Fewster et al., 2000). To estimate the CI of the estimated first derivative, a conditional bootstrap is carried out by resampling from the GAM residuals (in the cond_boot function) (Large *et al.*, 2013). This approach has the advantage to retain the information in the pressure variable and to be distribution-free.

In specific, new IND time series are created in cond_boot by resampling from the residuals of the original IND-Pressure GAM(M) and adding these to the original IND time series repeatedly (the number of repetitions is defined by n_boot). A separate GAM(M) is then fitted to each bootstrap IND time series in cond_boot, using the same effective degrees of freedom (edf) as found optimal for the original IND time series. For each of the bootstrapped GAMs, predicted IND time series given an evenly spaced pressure sequence are produced for which the first derivatives are calculated. Confidence intervals are computed by sorting the bootstrapped derivatives into ascending order and calculating the upper and lower percentiles defined by the ci argument (the default is the 2.5% and 97.5% percentiles representing the 95% CI).

A problem with GAMMs is that the smoothing parameters cannot be fixed, because the gamm function will treat the wiggly components of the smooth terms as random effects, the variance of which to be estimated by lme. Hence, the GAMMs refitted on the bootstrapped IND time series are allowed to have every shape from linear to the max. edfs set originally. This leads to often positive as

well as negative linear smoothers, which increases the confidence intervals of both fitted smoothers as well as derivative. When the IND response is weak and/or the error around the smoother high, the implemented routine to estimate the proportion of pressure range with significant slope can lead to 0%. For those models, the method 'approx_deriv' can be applied for comparison.

**Calculating the proportion of pressure range:**

The lack of any further IND response to pressure changes at its minimum or maximum measured is noted when zero is contained within the CI of the first derivative and quantified by calculating the proportion of points (evenly distributed along the pressure axis) inside the CI for scoring criteria 10.2. In specific, `calc_deriv` implements a routine, which identifies first which pressure values has a slope of zero (see the returned list-column `zero_in_conf`). It then checks whether the first value in `zero_in_conf` is TRUE (i.e., the lowest value of the evenly spaced pressure sequence has zero slope). If so, the first value in the vector zic_start_end (see the returned list-column `zic_start_end`) is also set to TRUE, meaning it goes into the proportion calculation as no IND response. The routine proceeds to the next values `zero_in_conf` and stops at the first FALSE value. It then starts the same procedure from end of the `zero_in_conf` vector. The proportion of pressure range (returned as `prop` variable in the output tibble) reflects the proportion of FALSE values in `zero_in_conf`.

## Value

The function returns the input model tibble with the following 12 columns added

`prop` The proportion of the observed pressure range where the IND indicator shows a response (see the last section in *Details*). Significant models with edf < edf_filter get as default a value of 1.0 (i.e., the IND responses to the entire observed pressure range).

`zero_in_conf` A list-column of logical vectors indicating for every pressure value (in press_seq) whether the slope of the IND response at that pressure value is within the confidence interval, i.e. is zero.

`zic_start_end` A list-column of logical vectors indicating for every pressure value (in press_seq) whether the slope is considered as zero for the proportion calculation (see the last section in *Details*).

`press_seq` A list-column with sequences of 100 evenly spaced pressure values.

`pred` A list-column with the predicted indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

`pred_ci_up` A list-column with the upper confidence limit of the bootstrapped predictions.

`pred_ci_low` A list-column with the lower confidence limit of the bootstrapped predictions.

`deriv1` A list-column with the first derivatives of the indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

`deriv1_ci_up` A list-column with the upper confidence limit of the bootstrapped first derivatives.

`deriv1_ci_low` A list-column with the lower confidence limit of the bootstrapped first derivatives.

`adj_n_boot` The number of successful bootstrap samples that was actually used for calculating the mean and confidence intervals of the predicted indicator response and the derivative.

`boot_error` A list-column capturing potential error messages that occurred as side effects when refitting the GAM(M)s on each bootstrap sample.

If none of the significant models has edf > edf_filter, only the variable `prop` will be added. If the 'approx_deriv' method was used, the output tibble will not contain the `pred`, `pred_ci_up`, and `pred_ci_low` variables.

## References

Fewster, R.M., Buckland, S.T., Siriwardena, G.M., Baillie, S.R., Wilson, J.D. (2000) Analysis of population trends for farmland birds using generalized additive models. *Ecology* 81, 1970-1984.

Large, S.I., Fay, G., Friedland, K.D., Link, J.S. (2013) Defining trends and thresholds in responses of ecological indicators to fishing and environmental pressures. *ICES Journal of Marine Science* 70, 755-767.

Trenkel, V.M., Rochet, M.J. (2009) Intersection-union tests for characterizing recent changes in smoothed indicator time series. *Ecological Indicators* 9, 732-739.

## See Also

`cond_boot` and `approx_deriv`

## Examples

```
# Using some models of the Baltic Sea demo data
init_tbl <- ind_init_ex[ind_init_ex$id %in% c(5,9,48,75), ]
mod_tbl <- merge_models_ex[merge_models_ex$id  %in% c(5,9,48,75), ]
deriv_tbl <- calc_deriv(init_tbl=init_tbl, mod_tbl=mod_tbl,
  n_boot = 40, par_comp = FALSE, seed=1)
```

---

`calc_nrmse` *Calculates the Normalized Root Mean Square Error (NRMSE) for a list of models*

---

## Description

`calc_nrmse` is a wrapper function that applies the `nrmse` function to a list of models given the input indicator and pressure observations. The function calculates first the predicted values for each model, which are then used for the NRMSE computation. The normalization method and transformation types required for `nrmse` can be set for all models the same or individually.

## Usage

```
calc_nrmse(
  press,
  ind,
  model,
  method = "sd",
  transformation = "none",
  trans_function = "none"
)
```

## Arguments

| | |
|---|---|
| press | A list of vectors containing the pressure values. |
| ind | A list of with vectors containing the indicator values. |
| model | A list containing the models. |
| method | A character string or vector of the same length as the model list indicating the value(s) to be used for the normalization of the RMSE. The default is the standard deviation, alternative methods are the "mean", "maxmin" (difference between the maximum and minimum observed values) or "iq" (interquartile) (see also nrmse). |
| transformation | A character string or vector of the same length as the model list indicating the type of transformation applied to the observations prior to the analysis. Choose one of the following: "none" (default), "sqrt", "4thrt" (fourth root), "log" (natural logarithm), "log10" (common, i.e. base 10, logarithm), "log2" (binary logarithm), "log1p" (i.e. log(1+x)), "arcsine" (if data is proportional, NOT percentage) or "other". |
| trans_function | If transformation is set to ""other" for some or all models, the function for the back-transformation needs to be defined here as single character string (applied to all models ) or as character vector (with one string per model). If no special transformation was applied use default setting "none". |

## Details

This wrapper function is used within the model_gam and model_gamm functions with the default "sd" method and no transformation. If another normalization is required or indicators where standardized prior to the analysis, this wrapper function should be applied to the final model output tibble to compute NRMSE that are based on the original indicator scale (advised for cross-indicator comparisons).

The more common transformation types applied to the indicator can be simply specified in the transformation argument, which will invoke the respective back-transformation of the observed and predicted indicator values before the NRMSE calculation. Any other transformation applied should be indicated with transformation = "other" and the respective back-transformation (simply the inverse of the original transformation) specified under trans_function, e.g. e.g. "5^x" if observations log(x, base = 5) transformed or "exp(x) - 0.001" if observations log(x + 0.001) transformed.

Missing values in obs and pred are removed before the computation proceeds, and only those positions with non-missing values in both pred and obs are considered in the computation.

## Value

The function returns a numerical vector of the same length then the input lists, with one NRMSE value for each model.

NOTE: If NA is returned for some models it means that either no model is available or that not enough observations with both indicator and pressure values are available (minimum of 2 required).

## See Also

nrmse, model_gam and model_gamm

## Examples

```
# Calculate NRMSE for the indicators TZA (~Fcod), which was let's say
# log(x+ 0.001)-transformed, and MS (~TSum), which was not transformed:
calc_nrmse(press = ind_init_ex$press_test[7:8], ind = ind_init_ex$ind_test[7:8],
model = model_gam_ex$model[7:8], method = "sd", transformation = c("other", "none"),
trans_function = c("exp(x) - 0.001", "none") )
```

---

`clust_sc` *Score-based cluster analysis*

---

## Description

`clust_sc` computes a hierarchical cluster analysis for the identification of indicator redundancies.

## Usage

```
clust_sc(dist_mat, method_clust = "average", ...)
```

## Arguments

| | |
|---|---|
| `dist_mat` | The distance matrix computed by the `dist_sc` function. |
| `method_clust` | The agglomeration method to be used in the `hclust` function. Default is "average", for alternatives see `hclust`. |
| `...` | Further arguments to be passed to the method `hclust`. |

## Value

An object of class `hclust` is returned, which describes the tree produced by the clustering process. See for more details `hclust`. Additionally, the cophenetic correlation coefficient and the Gower distance are printed in the console as guidance for selecting the best agglomeration method.

## See Also

`hclust`

Other score-based IND performance functions: `dist_sc_group()`, `dist_sc()`, `expect_resp()`, `plot_clust_sc()`, `plot_spiechart()`, `scoring()`, `summary_sc()`

## Examples

```
# Using the Baltic Sea demo data
scores_tbl <- scoring(trend_tbl = model_trend_ex,
  mod_tbl = all_results_ex, press_type = press_type_ex)
scores_mat <- summary_sc(scores_tbl)$scores_matrix
dist_matrix <- dist_sc(scores_mat, method_dist = "euclidean")
clust_analysis <- clust_sc(dist_matrix, method_clust = "complete")
```

cond_boot *Conditional bootstraps*

## Description

cond_boot creates n_boot predicted IND time series based on a conditional bootstrap for calculating the derivatives of the resulting smoothing curves.

## Usage

```
cond_boot(
  init_tbl,
  mod_tbl,
  excl_outlier,
  n_boot,
  ci,
  par_comp,
  no_clust,
  seed
)
```

## Arguments

| | |
|---|---|
| init_tbl | The output tibble of the ind_init function. |
| mod_tbl | A model output tibble from model_gam, select_model or merge_models representing the best model for each IND~pressure pair. |
| excl_outlier | logical; if TRUE, the outliers excluded in the original models will be also excluded in the bootstrapped models. |
| n_boot | Number of bootstraps. Select n_boot so that (n_boot - (n_boot *ci)) / 2 will be an integer. Otherwise, the function will increase n_boot automatically. The default is set to 200. |
| ci | Confidence interval of the bootstrapped smoothing functions and their derivatives. Must be between 0 and 1, default is 0.95. |
| par_comp | logical; if TRUE, the conditional bootstrap will be processed in parallel using several clusters, which can speed up the iteration process depending on the number of n_boot, models to bootstrap and number of processor cores. |
| no_clust | Number of clusters ("workers") for the parallel computation, with one cluster per core. If no_clust is set to NULL default, the number of clusters is set as the numbers of available cores – 1. |
| seed | A single value, interpreted as an integer, which specifies the seed of the random number generator (RNG) state for reproducibility. Due to the work splitting in the parallel computation, RNG streams are not comparable with the stream under serial computation. To reproduce results use the same type of computation with the same seed and number of clusters. |

## Details

cond_boot produces first n_boot new IND time series by resampling from the residuals of the original IND-Pressure GAM(M) and adding these to the original IND time series repeatedly. For GAMMs the correlation structure in the bootstrapped residuals is kept constant by using the arima.sim function with the bootstrapped residuals as times series of innovations and the correlation parameters from the original model. A separate GAM(M) is then fitted to each bootstrapped IND time series. If errors occur during the n_boot iterations of resampling and model fitting (e.g., convergence errors for GAMMs), the process is repeated until n_boot models have been fitted successfully.

The function calculates then the first derivatives of each bootstrapped IND time series prediction and computes a mean and confidence intervals (CI) of both IND predictions and derivatives. The CIs are computed by sorting the n_boot bootstrapped derivatives into ascending order and calculating the upper and lower percentiles defined by the ci argument (the default is the 2.5% and 97.5% percentiles representing the 95% CI).

The parallel computation in this function builds on the packages parallel and pbapply with its function pblapply. This allows the vectorized computations similar to lapply and adds further a progress bar.

## Value

The function returns the input model tibble with the following 9 columns added

press_seq A list-column with sequences of 100 evenly spaced pressure values.

pred A list-column with the predicted indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

pred_ci_up A list-column with the upper confidence limit of the bootstrapped predictions.

pred_ci_low A list-column with the lower confidence limit of the bootstrapped predictions.

deriv1 A list-column with the first derivatives of the indicator responses averaged across all bootstraps (for the 100 equally spaced pressure values).

deriv1_ci_up A list-column with the upper confidence limit of the bootstrapped first derivatives.

deriv1_ci_low A list-column with the lower confidence limit of the bootstrapped first derivatives.

adj_n_boot The number of successful bootstrap samples that was actually used for calculating the mean and confidence intervals of the predicted indicator response and the derivative.

boot_error A list-column capturing potential error messages that occurred as side effects when refitting the GAM(M)s on each bootstrap sample.

## See Also

the wrapper function calc_deriv

## Examples

```
# Using some models of the Baltic Sea demo data
init_tbl <- ind_init_ex[ind_init_ex$id %in% c(5,9,75), ]
mod_tbl <- merge_models_ex[merge_models_ex$id  %in% c(5,9,75), ]
deriv_tbl <- cond_boot(mod_tbl = mod_tbl, init_tbl = init_tbl,
```

```
excl_outlier = TRUE, n_boot = 200,ci = 0.95,
par_comp = FALSE, no_clust = NULL, seed = NULL)
```

---

`crit_scores_tmpl` *Template of (sub-)criteria and corresponding scoring information*

---

## Description

This table serves as basis for the `scoring` function and builds on the criterion-scoring scheme described in the underlying framework (Otto *et al.*, 2018). The user can modify the weights, scores, conditions or remove specific (sub-)crits.

## Usage

```
crit_scores_tmpl
```

## Format

A data frame with 27 rows and 12 variables:

**crit_id** ID of main criteria.

**crit** A vector of the main criteria.

**subcrit_id** ID of sub-criteria.

**subcrit** A vector of the sub-criteria.

**definition** A short description of the (sub)criteria.

**score_explanation** A short explanation on the required condition for a specific score.

**score** A vector of scores for each (sub-)criterion and the respective condition.

**weight** Weights assigned to each score to allow easy user adjustments. Default is 1 for all scores.

**score_pressure_specific** Additional information whether the scoring is pressure-specific or not.

**condition** A list-column single elements or vectors with various elements containing score-specific conditions put in R Syntax against which (sub-)criterion-specific variables (see condition_var) are checked.

**condition_var** The variables used as basis for scoring the specific (sub-)criterion.

**func_name** The names of the function that generate output tibbles containing the required variables.

dist_sc *Score-based distance matrix*

## Description

The function computes a distance matrix based on the scoring output tibble (or the output tibble from the expect_resp function).

## Usage

```
dist_sc(scores_mat, scores_tbl, method_dist = "euclidean", ...)
```

## Arguments

| | |
|---|---|
| scores_mat | A data frame or matrix containing the scores. This could be the $scores_matrix output of the summary_sc function. |
| scores_tbl | Deprecated argument from earlier version (0.1.0); scores_tbl represented the output tibble from the scoring function. |
| method_dist | Dissimilarity index used in the vegdist function to calculate the dissimilarity matrix based on the scores. Default is 'euclidean', for alternatives see vegdist. |
| ... | Further arguments to be passed to the method vegdist. |

## Value

The function returns a dist object.

## See Also

vegdist for the computation of the dissimilarity index

Other score-based IND performance functions: clust_sc(), dist_sc_group(), expect_resp(), plot_clust_sc(), plot_spiechart(), scoring(), summary_sc()

## Examples

```
# Using the Baltic Sea demo data
scores_tbl <- scoring(trend_tbl = model_trend_ex,
  mod_tbl = all_results_ex, press_type = press_type_ex)
scores_mat <- summary_sc(scores_tbl)$scores_matrix
dist_matrix <- dist_sc(scores_mat, method_dist = "euclidean")
```

dist_sc_group *Distance matrix averaged across group scores*

## Description

The dist_sc_group function computes a distance matrix for each group of scoring criteria and then averages the pair-wise distances across groups. This allows the computation of a more weighted distance measure and a closer clustering of indicators that e.g. respond to the same pressure types.

## Usage

```
dist_sc_group(x, method_dist = "euclidean", ...)
```

## Arguments

| | |
|---|---|
| x | A list of data frames or matrices that contain the indicator scores per group (see details). This could be the $scores_matrix output of the summary_sc, split into the different criteria groups. |
| method_dist | Dissimilarity index used in the vegdist function to to calculate the dissimilarity matrix based on the scores. Default is 'euclidean', for alternatives see vegdist. |
| ... | Further arguments to be passed to the method vegdist. |

## Details

Ordinary distance measures such as the Euclidean, Bray Curtis or Canberra distance treat all variables, i.e. here the criteria, the same, which might be not always desirable. For instance, two indicators that show no trend but respond each to a specific type of fishing pressure (with a score of 1) and a third indicator that only shows a trend (score of 1 here) would have all the same distance to each other. So to add more weight to the similarity of the first two indicators responding to the same pressure type, this function computes separate distance matrices that are then averaged.

## Value

The function returns a dist object.

## See Also

summary_sc and vegdist for the computation of the dissimilarity index

Other score-based IND performance functions: clust_sc(), dist_sc(), expect_resp(), plot_clust_sc(), plot_spiechart(), scoring(), summary_sc()

## Examples

```
# Using the Baltic Sea demo data
scores_tbl <- scoring(trend_tbl = model_trend_ex,
  mod_tbl = all_results_ex, press_type = press_type_ex)
scores_mat <- summary_sc(scores_tbl)$scores_matrix
# Split the scores by pressure-independent criteria and pressure types
dist_matrix <- dist_sc_group(x = list(
 scores_mat[,1:2], scores_mat[,3:8],
 scores_mat[,9:12], scores_mat[,13:16]) )
```

---

# expect_resp *Score adjustments for sub-criterion 10.1*

---

## Description

`expect_resp` runs a shiny app in which the expectation of the IND response to a pressure (sub-criterion 10.1) can be manually changed to 'yes' or 'no' based on visual inspection of the IND response curve.

## Usage

```
expect_resp(mod_tbl, scores_tbl, crit_scores = INDperform::crit_scores_tmpl)
```

## Arguments

| | |
|---|---|
| `mod_tbl` | Output tibble from the IND~pressure modeling functions. |
| `scores_tbl` | The output tibble from the `scoring` function. |
| `crit_scores` | The(un)modified criterion-scoring template `crit_scores_tmpl`; has to be the same than used in `scoring`. Default is the unmodified template `crit_scores_tmpl`. |

## Details

The sub-criterion 10.1 (i.e. the IND response to a pressure, which has been found significant, is in line with expectations based on ecological knowledge) has been set to a default score of 1 (no expectation / unclear as response is highly non-linear) in the `scoring` function. Determining whether the IND response modeled in the GAM/GAMM meets specific expectations can only be done based on visual model inspections. `expect_resp` provides only a very simple graphical representation of this smoothing function.

For a more comprehensive figure use the `plot_model` function and then go back to this function for modifications of the expectation scores.

## Value

The function returns the input scoring tibble, but with modified scores in the variable `C10_1`, once the "Press Me!" button is activated.

## See Also

plot_model for visualization of the IND responses to pressures

Other score-based IND performance functions: clust_sc(), dist_sc_group(), dist_sc(), plot_clust_sc(), plot_spiechart(), scoring(), summary_sc()

## Examples

```
## Not run:
 # Using the Baltic Sea demo data:
 # Apply first the scoring on the model outputs
 scores_tbl <- scoring(trend_tbl = model_trend_ex, mod_tbl = all_results_ex,
   press_type = press_type_ex)
 # Then run the expect_resp() shiny function to correct one criterion
 scores_tbl <- expect_resp(all_results_ex, scores_tbl)
 # Check if it worked:
 expect_resp(all_results_ex, scores_tbl)

## End(Not run)
```

---

# find_id *Extracts the IND~pressure IDs in the tibble.*

---

## Description

find_id is a helper function for the user to extract the ID for a specific indicator(IND), pressure or IND~pressure combination. The 'id' in the returned tibble can then be used for filtering tibbles when using the other IND~pressure modeling functions.

## Usage

```
find_id(mod_tbl, ind_name = NULL, press_name = NULL)
```

## Arguments

| | |
|---|---|
| mod_tbl | A tibble containing the columns ind and press.. |
| ind_name | One or more character string naming the indicators of interest. |
| press_name | One or more character string naming the pressures of interest. |

## Value

The function returns a tibble including the id for the respective 'ind' and/or 'press'.

## See Also

Other IND~pressure modeling functions: ind_init(), model_gamm(), model_gam(), plot_diagnostics(), plot_model(), scoring(), select_model(), test_interaction()

## Examples

```
# Using the Baltic Sea demo data:
# Look for specific INDs in combination with every pressure
ind_name <- c("TZA","MS")
find_id(model_gam_ex, ind_name)$id
# Look for specific IND~pressure combinations
press_name <- c("Tsum", "Swin")
find_id(model_gam_ex, ind_name, press_name)
# Look for specific pressures in combination with every IND
find_id(model_gam_ex, press_name = press_name)
```

---

# ind_ex *Food web indicators of the Central Baltic Sea (Bornholm Basin).*

---

## Description

Food web indicators of the Central Baltic Sea (Bornholm Basin).

## Usage

```
ind_ex
```

## Format

A data frame with 30 rows and 13 variables. These indicator time series represent a slight modification of the original time series represented in Otto et al. (2018).

**Year** year given as integer

**TZA** ln-transformed Total Zooplankton Abundance (in N m^-3^)

**MS** Mean Size of zooplankton (in wet weight micro grams)

**rCC** ln-transformed ratio of Cladoceran to Copepod Abundance

**Cops** ln-transformed Copepod biomass (in mg)

**Micro** ln-transformed biomass of Microphageous zooplankton (in mg)

**rZPPP** ratio of Zooplankton to Phytoplankton biomass

**Sprat** ln-transformed Sprat abundance (in million N)

**Herring** ln-transformed Herring abundance (in millions N)

**Stickle** ln-transformed Stickleback CPUE (in kg h^-1^)

**Cod** ln-transformed Cod CPUE (in kg h^-1^)

**SPF** ln-transformed Small Predatory Fish (i.e. sprat and herring <10cm) (in million g)

**LPF** ln-transformed Large Predatory Fish (i.e. Cod >38cm) (in kg h^-1^)

`ind_init` *Initialization of indicator-pressure models*

## Description

`ind_init` combines the time vector and the indicator (IND) and pressure data into one tibble with defined training and test observations. All INDs are combined with all pressures provided as input.

## Usage

```
ind_init(ind_tbl, press_tbl, time, train = 0.9, random = FALSE)
```

## Arguments

| | |
|---|---|
| `ind_tbl` | A data frame, matrix or tibble containing only the (numeric) IND variables. Single indicators should be coerced into a data frame to keep the indicator name. If kept as vector, default name will be 'ind'. |
| `press_tbl` | A data frame, matrix or tibble containing only the (numeric) pressure variables. Single pressures should be coerced into a data frame to keep the pressure name. If kept as vector, default name will be 'press'. |
| `time` | A vector containing the actual time steps (e.g. years; should be the same as in the IND and pressure data). |
| `train` | The proportion of observations that should go into the training data on which the GAMs are later fitted. Has to be a numeric value between 0 and 1; the default is 0.9. |
| `random` | logical; should the observations for the training data be randomly chosen? Default is FALSE, so that the last time units (years) are chosen as test data. |

## Details

`ind_init` will combine every column in ind_tbl with every column in press_tbl so that each row will represent one IND~press combination. The input data will be split into a training and a test data set. The returned tibble is the basis for all IND~pressure modeling functions.

If not all IND~pressure combinations should be modeled, the respective rows can simply be removed from the output tibble or `ind_init` is applied multiple times on data subsets and their output tibbles merged later using e.g. `bind_rows`.

## Value

The function returns a `tibble`, which is a trimmed down version of the data.frame(), including the following elements:

`id` Numerical IDs for the IND~press combinations.

`ind` Indicator names.These might be modified to exclude any character, which is not in the model formula (e.g. hyphens, brackets, etc. are replaced by an underscore, variables starting with a number will get an x before the number.

press Pressure names.These might be modified to exclude any character, which is not in the model formula (e.g. hyphens, brackets, etc. are replaced by an underscore, variables starting with a number will get an x before the number.

ind_train A list-column with indicator values of the training data.

press_train A list-column with pressure values of the training data.

time_train A list-column with the time steps of the training data.

ind_test A list-column with indicator values of the test data.

press_test A list-column with pressure values of the test data.

time_test A list-column with the time steps of the test data.

train_na logical; indicates the joint missing values in the training IND and pressure data. That includes the original NAs as well as randomly selected test observations that are within the training period. This vector is needed later for the determination of temporal autocorrelation.

## See Also

tibble and the vignette("tibble") for more informations on tibbles

Other IND~pressure modeling functions: find_id(), model_gamm(), model_gam(), plot_diagnostics(), plot_model(), scoring(), select_model(), test_interaction()

## Examples

```
# Using the Baltic Sea demo data in this package
press_tbl <- press_ex[ ,-1] # excl. Year
ind_tbl <- ind_ex[ ,-1] # excl. Year
time <- ind_ex[ ,1]
# Assign randomly 50% of the observations as training data and
# the other 50% as test data
ind_init(ind_tbl, press_tbl, time, train = 0.5, random = TRUE)
# To keep the name when testing only one indicator and pressure, coerce both vectors
# data frames
ind_init(ind_tbl = data.frame(MS = ind_tbl$MS), press_tbl = data.frame(Tsum = press_tbl$Tsum),
 time, train = .5, random = TRUE)
```

---

# ind_init_ex *Output tibble from the ind_init function*

---

## Description

This is an example output tibble from the ind_init function applied on the Central Baltic Sea food web indicator demonstration data.

## Usage

```
ind_init_ex
```

## Format

A data frame with 84 rows and 10 variables:

`id` Numerical IDs for the IND~press combinations.

`ind` Indicator names.

`press` Pressure names.

`ind_train` A list-column with indicator values of the training data.

`press_train` A list-column with pressure values of the training data.

`time_train` train data from year

`ind_test` A list-column with indicator values of the test data.

`press_test` A list-column with pressure values of the test data.

`time_test` test data from year

`train_na` logical; indicates the joint missing values in the training IND and pressure data. That includes the original NAs as well as randomly selected test observations that are within the training period

---

# `merge_models` *Merging two model output tibbles.*

---

## Description

The function appends the second model output tibble to the first while keeping all variables from both tibbles.

## Usage

```
merge_models(mod_tbl1, mod_tbl2)
```

## Arguments

| | |
|---|---|
| `mod_tbl1` | Model output tibble created e.g. with `model_gam`. |
| `mod_tbl2` | Model output tibble created e.g. with `select_model`. |

## Details

`merge_models` function applies internally the dplyr::bind_rows function so that columns are matched by name, and any missing columns will be filled with NA. The function has also some data validation incorporated to check for double entries.

## Value

`merge_models` returns the same type as the input including all columns of both tibbles.

## Examples

```
# Using some models of the Baltic Sea demo data:
# Merging GAM and GAMM tibbles
test_ids <- 47:50 # choose subset
gam_tbl <- model_gam_ex[test_ids,]
gamm_tbl <- model_gamm(ind_init_ex[test_ids,], filter= gam_tbl$tac)
best_gamm <- select_model(gam_tbl, gamm_tbl)
merge_models(gam_tbl[gam_tbl$tac == FALSE,], best_gamm)

# Merge 2 IND-specific GAM tibbles (where)
dat_init <- ind_init(
  ind_tbl = ind_ex[, c("TZA", "Cod")],
  press_tbl = press_ex[, c("Tsum", "Swin")],
  time = ind_ex[,1])
gam_tbl1 <- model_gam(dat_init[1:2, ])
# treat a subset differently, e.g. when setting k
gam_tbl2 <- model_gam(dat_init[3:4, ], k = 3)
merge_models(gam_tbl1, gam_tbl2)
```

---

# merge_models_ex *Output tibble from the* merge_models *function*

---

## Description

This is an example output tibble from the merge_models function applied on the Central Baltic Sea food web indicator demonstration data. More specifically, the function is applied on a subset of the model_gam_ex tibble (including only GAMs with no temporal autocorrelation) and the model_gamm_ex after selecting the best GAMMs using select_model.

## Usage

```
merge_models_ex
```

## Format

A data frame with 84 rows and 17 variables:

`id` Numerical IDs for the IND~press combinations.

`ind` Indicator names.

`press` Pressure names.

`model_type` Specification of the model type; at this stage containing only "gam" (Generalized Additive Model).

`corrstruc` Specification of the correlation structure; at this stage containing only "none".

`aic` AIC of the fitted models

`edf` Estimated degrees of freedom for the model terms.

`p_val` The p values for the smoothing term (the pressure).

`signif_code` The significance codes for the p-values.

`r_sq` The adjusted r-squared for the models. Defined as the proportion of variance explained, where original variance and residual variance are both estimated using unbiased estimators. This quantity can be negative if your model is worse than a one parameter constant model, and can be higher for the smaller of two nested models.

`expl_dev` The proportion of the null deviance explained by the models.

`nrmse` Absolute values of the root mean square error normalized by the standard deviation (NRMSE) using no back-transformation.

`ks_test` The p-values from a Kolmogorov-Smirnov Test applied on the model residuals to test for normal distribution. P-values > 0.05 indicate normally distributed residuals.

`tac` logical; indicates whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC.

`pres_outlier` A list-column with outliers identified for each model (i.e. Cook's distance > 1). The indices present the position in the training data, including NAs.

`excl_outlier` A list-column listing all outliers per model that have been excluded in the GAM fitting

`model` A list-column of IND~press-specific gam objects.

---

`model_gam` *Modeling of indicator responses to single pressures with GAMs*

---

## Description

`model_gam` applies Generalized Additive Models (GAMs) to each IND~pressure combination created in `ind_init` and returns a tibble with IND~pressure-specific GAM outputs.

## Usage

```
model_gam(init_tbl, k = 5, family = stats::gaussian(), excl_outlier = NULL)
```

## Arguments

| | |
|---|---|
| `init_tbl` | The output tibble of the `ind_init` function. |
| `k` | Choice of knots (for the smoothing function `s`); the default is 5. |
| `family` | A description of the error distribution and link to be used in the GAM. This needs to be defined as a family function (see also `family`). All standard family functions can be used as well some of the distribution families in the mgcv package (see `family.mgcv`; e.g.`negbin` or `nb`). |
| `excl_outlier` | A list of values identified as outliers in specific IND~pressure GAMs, which should be excluded in this modeling step (the output tibble of this function includes the variable 'pres_outlier', which is a column-list containing all indices of values with cook's distance > 1 (see below). The function can be re-run again, then excluding all these outliers provided in `$pres_outlier` from the the first run (see example)). |

## Details

To evaluate the IND's sensitivity and robustness time series of the IND are modeled as a smoothing function of one single pressure variable (using a subset of the data as training dataset, e.g. excluding the years of the annual time series). The GAMs are build using the default settings in the `gam` function and the smooth term function `s`). However, the user can adjust the distribution and link by modifying the family argument as well as the maximum level of non-linearity by setting the number of knots:

```
gam(ind ~ s(press,k = k),family = family,data = training_data)
```

In the presence of significant temporal auto-correlation, GAMs should be extended to Generalized Additive Mixed Models (GAMMs) by including auto-regressive error structures to correct for the auto-correlation (Pinheiro and Bates, 2000). This is implemented in the function `model_gamm`.

The returned tibble contains various model outputs needed for scoring the sensitivity and robustness subcriteria:

- `p_val` to identify whether an IND responds to a specific pressure
- `r_sq` for the strength of the IND response
- `edf` for the non-linearity of the IND response
- `nrmse` for the robustness of the established IND~pressure relationship

The robustness of the modeled pressure relationship based on the training data is evaluated by measuring how well the model prediction matches the test dataset, e.g. the last years. This is quantified by computing the absolute value of the normalized root mean square error (NRMSE) on the test dataset. The normalization to the mean of the observed test data allows for comparisons and a general scoring of the model robustness across INDs with different scales or units.

## Value

The function returns a `tibble`, which is a trimmed down version of the data.frame(), including the following elements:

`id` Numerical IDs for the IND~press combinations.

`ind` Indicator names.

`press` Pressure names.

`model_type` Specification of the model type; at this stage containing only "gam" (Generalized Additive Model).

`corrstruc` Specification of the correlation structure; at this stage containing only "none".

`aic` AIC of the fitted models

`edf` Estimated degrees of freedom for the model terms.

`p_val` The p values for the smoothing term (the pressure).

`signif_code` The significance codes for the p-values.

`r_sq` The adjusted r-squared for the models. Defined as the proportion of variance explained, where original variance and residual variance are both estimated using unbiased estimators. This quantity can be negative if your model is worse than a one parameter constant model, and can be higher for the smaller of two nested models.

`expl_dev` The proportion of the null deviance explained by the models.

`nrmse` Absolute values of the root mean square error normalized by the standard deviation (NRMSE).

`ks_test` The p-values from a Kolmogorov-Smirnov Test applied on the model residuals to test for normal distribution. P-values > 0.05 indicate normally distributed residuals.

`tac` logical; indicates whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC. NAs in the time series due to real missing values, test data extraction or exclusion of outliers are explicitly considered. The test is based on the following condition: if any of the acf **and** pacf values of lag 1 - 5 are greater than 0.4 or lower than -0.4, a TRUE is returned.

`pres_outlier` A list-column with all indices of values identified as outliers in each model (i.e.cook's distance > 1). The indices present the position in the training data, including NAs.

`excl_outlier` A list-column listing all outliers per model that have been excluded in the GAM fitting

`model` A list-column of IND~press-specific gam objects that contain additionally the logical vector indicating missing values (`$train_na`).

## References

Pinheiro, J.C., Bates, D.M. (2000) Mixed-Effects Models in S and S-Plus. Springer, New York, 548pp.

## See Also

`tibble` and the `vignette("tibble")` for more informations on tibbles, `gam` for more information on GAMs, and `plot_diagnostics` for assessing the model diagnostics

Other IND~pressure modeling functions: `find_id()`, `ind_init()`, `model_gamm()`, `plot_diagnostics()`, `plot_model()`, `scoring()`, `select_model()`, `test_interaction()`

## Examples

```
# Using the Baltic Sea demo data in this package
dat_init <- ind_init(
  ind_tbl = ind_ex[, c("Sprat", "Cod")],
  press_tbl = press_ex[, c("Tsum", "Swin", "Fcod", "Fher")],
  time = ind_ex[ ,1])
gam_tbl <- model_gam(dat_init)
# Any outlier?
gam_tbl$pres_outlier
# Exclude outliers by passing this list as input:
gam_tbl_out <- model_gam(dat_init, excl_outlier = gam_tbl$pres_outlier)


 # Using another error distribution
 ind_sub <- round(exp(ind_ex[ ,c(2,8,9)]),0) # to unlog data and convert to integers
 ind_tbl2 <- ind_init(ind_sub, press_ex, time = ind_ex$Year)
 model_gam(ind_tbl2, family = poisson(link="log"))
```

model_gamm *Modeling of indicator responses to single pressures with GAMMs*

## Description

model_gamm accounts for temporal autocorrelation (TAC) in the time series by fitting Generalized Additive Mixed Models (GAMMs) that include AR or ARMA correlation structures using the gamm function. The GAMMs are applied to all IND~pressure combinations provided as input or only those with significant TAC in the GAM residuals (using filter argument).

## Usage

```
model_gamm(
  init_tbl,
  k = 5,
  family = stats::gaussian(),
  excl_outlier = NULL,
  filter = NULL
)
```

## Arguments

| | |
|---|---|
| init_tbl | The output tibble of the ind_init function. |
| k | Choice of knots (for the smoothing function s); the default is 5. |
| family | A description of the error distribution and link to be used in the GAM. This needs to be defined as a family function (see also family). All standard family functions can be used as well some of the distribution families in the mgcv package (see family.mgcv; e.g.negbin). Note that nb, which estimates theta parameter, cannot be used for gamm. |
| excl_outlier | A list of values identified as outliers in specific IND~pressure GAMMs, which should be excluded in this modeling step (the output tibble of this function includes the variable 'pres_outlier', which is a column-list containing all indices of values with cook's distance > 1 (see below). The function can be re-run again, then excluding all these outliers provided in $pres_outlier from the the first run (see example)). |
| filter | logical; a filter used to select specific rows in init_tbl (row gets selected if value TRUE). That could be the tac column in the model_gam output tibble which indicates whether the model residuals show TAC. |

## Details

Modeling first-differenced indicator time series can be an alternative solution to avoid temporal dependence between observations. However, this approach does often not help reducing the significant auto-correlation while GAMMs do as found in Otto *et al.* (2018). Such an extension implies

that the single elements of the response variable are not independent anymore and that the correlation between the residuals at time t1 and t2 only depends on their time difference t1 – t2 (Wood, 2006).

In `model_gamm` six GAMMs are computed for each filtered IND~pressure pair, i.e.

1. no correlation structure (for AIC comparison)
2. auto-regressive error structure of order p=1 (AR1)
3. auto-regressive error structure of order p=2 (AR2)
4. auto-regressive moving average of order p=1 and q=1 (ARMA11)
5. auto-regressive moving average of order p=1 and q=2 (ARMA12)
6. auto-regressive moving average of order p=2 and q=1 (ARMA21)

## Value

Returns a model output tibble that contains for each filtered IND~pressure pair 6 rows with the individual GAMM outputs. The structure remains the same as in `model_gam` except for the explained deviance, which is not computed by the gamm function. The selection of the final correlation structure for each IND~pressure model can be done manually on this tibble or with an automatized routine using `select_model`.

## References

Otto, S.A., Kadin, M., Casini, M., Torres, M.A., Blenckner, T. (2018) A quantitative framework for selecting and validating food web indicators. *Ecological Indicators*, 84: 619-631, doi: https://doi.org/10.1016/j.ecolind.2017.

Wood, S.N. (2006) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press

## See Also

`gamm` for more information on GAMMs and `plot_diagnostics` for assessing the model diagnostics

Other IND~pressure modeling functions: `find_id()`, `ind_init()`, `model_gam()`, `plot_diagnostics()`, `plot_model()`, `scoring()`, `select_model()`, `test_interaction()`

## Examples

```
# Using the Baltic Sea demo data in this package
dat_init <- ind_init(
  ind_tbl = data.frame(Cod = ind_ex$Sprat),
  press_tbl = press_ex[, c("Fsprat", "Fher")],
  time = ind_ex[ ,1])
gam_tbl <- model_gam(dat_init)
# Any temporal autocorrelation
gam_tbl$tac
# Applying model_gamm function and passing the $tac variable as filter
gamm_tbl <- model_gamm(dat_init, filter = gam_tbl$tac)
```

model_gamm_ex *Model output tibble from the* model_gamm *function*

## Description

This is an example output tibble from the model_gamm function applied on the Central Baltic Sea food web indicator demonstration data.

## Usage

```
model_gamm_ex
```

## Format

A data frame with 234 rows and 16 variables:

id Numerical IDs for the IND~press combinations.

ind Indicator names.

press Pressure names.

model_type Specification of the model type; at this stage containing only "gam" (Generalized Additive Model).

corrstruc Specification of the correlation structure; at this stage containing only "none".

aic AIC of the fitted models

edf Estimated degrees of freedom for the model terms.

p_val The p values for the smoothing term (the pressure).

signif_code The significance codes for the p-values.

r_sq The adjusted r-squared for the models. Defined as the proportion of variance explained, where original variance and residual variance are both estimated using unbiased estimators. This quantity can be negative if your model is worse than a one parameter constant model, and can be higher for the smaller of two nested models.

nrmse Absolute values of the root mean square error normalized by the standard deviation (NRMSE) using no back-transformation.

ks_test The p-values from a Kolmogorov-Smirnov Test applied on the model residuals to test for normal distribution. P-values > 0.05 indicate normally distributed residuals.

tac logical; indicates whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC.

pres_outlier A list-column with outliers identified for each model (i.e. Cook‘s distance > 1). The indices present the position in the training data, including NAs.

excl_outlier A list-column listing all outliers per model that have been excluded in the GAM fitting

model A list-column of IND~press-specific gam objects.

model_gam_ex *Model output tibble from the* model_gam *function*

## Description

This is an example output tibble from the model_gam function applied on the Central Baltic Sea food web indicator demonstration data.

## Usage

```
model_gam_ex
```

## Format

A data frame with 84 rows and 17 variables:

id Numerical IDs for the IND~press combinations.

ind Indicator names.

press Pressure names.

model_type Specification of the model type; at this stage containing only "gam" (Generalized Additive Model).

corrstruc Specification of the correlation structure; at this stage containing only "none".

aic AIC of the fitted models

edf Estimated degrees of freedom for the model terms.

p_val The p values for the smoothing term (the pressure).

signif_code The significance codes for the p-values.

r_sq The adjusted r-squared for the models. Defined as the proportion of variance explained, where original variance and residual variance are both estimated using unbiased estimators. This quantity can be negative if your model is worse than a one parameter constant model, and can be higher for the smaller of two nested models.

expl_dev The proportion of the null deviance explained by the models.

nrmse Absolute values of the root mean square error normalized by the standard deviation (NRMSE) using no back-transformation.

ks_test The p-values from a Kolmogorov-Smirnov Test applied on the model residuals to test for normal distribution. P-values > 0.05 indicate normally distributed residuals.

tac logical; indicates whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC.

pres_outlier A list-column with outliers identified for each model (i.e. Cook‘s distance > 1). The indices present the position in the training data, including NAs.

excl_outlier A list-column listing all outliers per model that have been excluded in the GAM fitting

model A list-column of IND~press-specific gam objects.

model_trend *Modeling of indicator trends*

## Description

The function models the long-term trend of each indicator (IND) based on Generalized Additive Models (GAM) and returns a tibble with IND-specific GAM outputs.

## Usage

```
model_trend(
  ind_tbl,
  time,
  train = 1,
  random = FALSE,
  k = 4,
  family = stats::gaussian()
)
```

## Arguments

| | |
|---|---|
| ind_tbl | A data frame, matrix or tibble containing only the (numeric) IND variables. Single indicators should be coerced into a data frame to keep the indicator name. If kept as vector, default name will be 'ind'. |
| time | A vector containing the actual time steps (e.g. years; should be the same for the IND data). |
| train | The proportion of observations that should go into the training data on which the GAMs are fitted. Has to be a numeric value between 0 and 1; the default is 1 (i.e. the full time series is fitted). |
| random | logical; should the observations for the training data be randomly chosen? Default is FALSE. |
| k | Choice of knots (for the smoothing function s); the default is 4. |
| family | A description of the error distribution and link to be used in the GAM. This needs to be defined as a family function (see also family). All standard family functions can be used as well some of the distribution families in the mgcv package (see family.mgcv; e.g.negbin or nb). |

## Details

To test for linear or non-linear long-term changes, each indicator (IND) in the ind_tbl is modeled as a smoothing function of the time vector (usually years) using the gam function. The trend can be tested for the full time series (i.e. all observations are used as training data) or for a random or selected subset.

The GAMs are build using the default settings in the gam function and the smooth term function s). However, the user can adjust the distribution and link by modifying the family argument as well as the maximum level of non-linearity by setting the number of knots:

```
gam(ind ~ s(time,k = k),family = family,data = training_data)
```

## Value

The function returns a tibble, which is a trimmed down version of the data.frame(), including the following elements:

`ind_id` Indicator IDs.

`ind` Indicator names. These might be modified to exclude any character, which is not in the model formula (e.g. hyphens, brackets, etc. are replaced by an underscore, variables starting with a number will get an x before the number.

`p_val` The p values for the smoothing term (here time).

`model` A list-column of indicator-specific gam objects.

`ind_train` A list-column with indicator values of the training data.

`time_train` A list-column with the time values (e.g. years) of the training data.

`pred` A list-column with indicator values predicted from the GAM for the training period.

`ci_up` A list-column with the upper 95% confidence interval of predicted indicator values.

`ci_low` A list-column with the lower 95% confidence interval of predicted indicator values.

## See Also

`plot_diagnostics` for assessing model diagnostics, `plot_trend` for trend visualization, `tibble` and the `vignette("tibble")` for more information on tibbles, `gam` for more information on GAMs

## Examples

```
# Using the Baltic Sea demo data in this package
ind_tbl <- ind_ex[ ,-1] # excluding the year
time <- ind_ex$Year
# Using the default settings
trend_tbl <- model_trend(ind_tbl, time)
# Change the training and test data assignment
model_trend(ind_tbl, time, train = .5, random = TRUE)
# To keep the name when testing only one indicator, coerce vector to data frame
model_trend(data.frame(MS = ind_tbl$MS), time, train = .5, random = TRUE)
```

---

# model_trend_ex *Model output tibble from the* `model_trend` *function*

---

## Description

This is an example output tibble from the `model_trend` function applied on the Central Baltic Sea food web indicator demonstration data.

## Usage

```
model_trend_ex
```

## Format

A data frame with 12 rows and 9 variables:

`ind_id` Indicator IDs.

`ind` Indicator names.

`p_val` The p values for the smoothing term (here time).

`model` A list-column of indicator-specific gam objects.

`ind_train` A list-column with indicator values of the training data.

`time_train` A list-column with the time values (e.g. years) of the training data.

`pred` A list-column with indicator values predicted from the GAM for the training period.

`ci_up` A list-column with the upper 95% confidence interval of predicted indicator values.

`ci_low` A list-column with the lower 95% confidence interval of predicted indicator values.

---

# nrmse *Normalized Root Mean Square Error*

---

## Description

`nrmse` is a function that allows the user to calculate the normalized root mean square error (NRMSE) as absolute value between predicted and observed values using different type of normalization methods. It further allows the NRMSE calculation on the scale of the untransformed indicator, which is advisable for a comparison across indicators.

## Usage

```
nrmse(
  pred,
  obs,
  method = "sd",
  transformation = "none",
  trans_function = "none"
)
```

## Arguments

| | |
|---|---|
| `pred` | A vector of predicted values. |
| `obs` | A vector of observed values. |
| `method` | A character string indicating the value to be used for the normalization of the RMSE. The default is the standard deviation. Alternatively, you can choose the `"mean"`, `"maxmin"` (difference between the maximum and minimum observed values) or `"iq"` (interquartile) |

`transformation` The type of transformation applied to the observations prior to the analysis. Choose one of the following: `"none"` (default), `"sqrt"`, `"4thrt"` (fourth root), `"log"` (natural logarithm), `"log10"` (common, i.e. base 10, logarithm), `"log2"` (binary logarithm), `"log1p"` (i.e. log(1+x)), `"arcsine"` (if data is proportional, NOT percentage) or `"other"`.

`trans_function` If `transformation` is set to `""other"`, the function for the back-transformation needs to be defined here as character string (simply the inverse of the original transformation), e.g. "5^x" if observations log(x, base = 5) transformed or "exp(x) - 0.001" if observations log(x + 0.001) transformed. Default is `"none"`.

## Details

The for most common normalization methods are implemented here:

- the **mean**: NRMSE = RMSE / mean(obs) - the **standard deviation**: NRMSE = RMSE / sd(obs) - the **difference between maximum and minimum**: NRMSE = RMSE / (max(obs)-min(obs) - the **interquartile range**; NRMSE = RMSE / (Q1-Q3), i.e. the difference between the 25th and 75th percentile of observations

Missing values in obs and pred are removed before the computation proceeds, and only those positions with non-missing values in both pred and obs are considered in the computation.

## Value

The function returns a single NRMSE value (expressed as absolute value). In case the number of positions with non-missing values in both pred and obs is less then 2, NA is returned with a message.

## See Also

`calc_nrmse`

## Examples

```
obs <- c(10, 14, 20)
pred <- c(9, 12, 13)
# Calculating the sd-based NRMSE for untransformed data
nrmse(pred, obs)
# Calculating the iq-based NRMSE for log(x+0.001) transformed data
nrmse(pred, obs, method = "iq", transformation = "other", trans_function = "exp(x)-0.001")
```

---

# plot_clust_sc *Dendrogram of the cluster analysis.*

---

## Description

`plot_clust_sc` generates from the cluster analysis a dendrogram based on the `ggplot2` and `ggdendro` packages.

### Usage

```
plot_clust_sc(x, rotate = FALSE, text_size = 15)
```

### Arguments

| | |
|---|---|
| x | Output from the cluster analysis (object of class hclust). |
| rotate | If TRUE, rotates plot by 90 degrees. |
| text_size | Size of the title and axes labels. |

### Value

The function returns a ggplot object.

### See Also

ggplot, dendro_data for the data extraction from the hclust object to produce the dendrogram

Other score-based IND performance functions: clust_sc(), dist_sc_group(), dist_sc(), expect_resp(), plot_spiechart(), scoring(), summary_sc()

### Examples

```
# Using the Baltic Sea demo data
scores_tbl <- scoring(trend_tbl = model_trend_ex,
  mod_tbl = all_results_ex, press_type = press_type_ex)
scores_mat <- summary_sc(scores_tbl)$scores_matrix
dist_matrix <- dist_sc(scores_mat, method_dist = "euclidean")
clust_analysis <- clust_sc(dist_matrix, method_clust = "complete")
plot_clust_sc(clust_analysis)

# To modify the plot:
plot_clust_sc(clust_analysis, rotate = TRUE, text_size = 20) +
 ggplot2::theme(title = ggplot2::element_text(colour = "blue", size = 10))
```

---

## plot_diagnostics *Diagnostic plots for a fitted GAM, GAMM or threshold-GAM(M)*

---

### Description

plot_diagnostics takes a list of models of class 'gam', 'gamm' or 'thresh_gam' or a mix of those and produces some diagnostic information of the fitting procedure and results. The function returns a tibble with 6 list-columns containing individual plots (ggplot2 objects) and one list-column containing a plot that shows all diagnostic plots together.

### Usage

```
plot_diagnostics(model_list)
```

## Arguments

| | |
|---|---|
| model_list | A list with models of class gam(m) and/or thresh_gam, e.g. the list-column model from the model_gam output tibble. |

## Details

The function can deal with any model of the classes 'gam', 'gamm' or 'thresh_gam' as long as the input is a flat list. That means:

- If only one model is provided as input coerce the model explicitly to class 'list'. An input such as model_gam_ex[1, "model"] will not work as the class is a tibble. Use instead model_gam_ex$model[1].
- If the input are one or more threshold-GAMs selected from the test_interaction output (variable thresh_models the model list features a nested structure: each IND~pressure pair (row) might have more than one threshold-GAM. To remove the nested structure use e.g. the flatten function (see examples).

## Value

The function returns a tibble, which is a trimmed down version of the data.frame(), including the following elements:

ind Indicator names.

press Pressure names.

cooks_dist A list-column of ggplot2 objects that show the Cook's distance of all observations, which is a leave-one-out deletion diagnostics to measure the influence of each observation. Data points with a large Cook's distance (> 1) are considered to merit closer examination in the analysis.

acf_plot A list-column of ggplot2 objects that show the autocorrelation function for the residuals. NAs in the time series due to real missing values, test data extraction or exclusion of outliers are explicitly considered.

pacf_plot A list-column of ggplot2 objects that show the partial autocorrelation function for the residuals. NAs are explicitly considered.

resid_plot A list-column of ggplot2 objects that show residuals vs. fitted values.

qq_plot A list-column of ggplot2 objects that show the quantile-quantile plot for normality.

gcvv_plot A list-column of ggplot2 objects that show for a threshold-GAM the development of the generalized cross-validation value at different thresholds level of the modifying pressure variable. The GCV value of the final chosen threshold should be distinctly lower than for all other potential thresholds, i.e., the line should show a pointy negative peak at this threshold. If this is not the case, e.g. the trough is very wide with similar GCV values for nearby thresholds, the threshold-GAM is not optimal and should not be favored over a GAM despite the better LOOCV (leave-one-out cross-validation value).

all_plots A list-column of ggplot2 objects that show all five (six if threshold-GAM) plots together. For this plot, drawing canvas from the cowplot package were added on top of ggplot2.

## See Also

cooks.distance, acf, pacf, qqnorm, and flatten for removing a level hierarchy from a list

Other IND~pressure modeling functions: find_id(), ind_init(), model_gamm(), model_gam(), plot_model(), scoring(), select_model(), test_interaction()

## Examples

```
# Using some models of the Baltic Sea demo data:
# Apply function to a list of various model types
model_list <- c(all_results_ex$thresh_models[[5]],
  model_gam_ex$model[39], all_results_ex$model[76])
plots <- plot_diagnostics(model_list)
plots$cooks_dist[[1]]
plots$acf_plot[[2]]
plots$pacf_plot[[3]]
plots$resid_plot[[1]]
plots$qq_plot[[1]]
plots$gcvv_plot[[1]] # for threshold models
plots$all_plots[[1]] # shows all 5-6 plots

# Make sure that thresh_models have not a nested list structure:
model_list <- all_results_ex$thresh_models[5:6] %>% purrr::flatten(.)
plots <- plot_diagnostics(model_list)
```

---

# plot_model *Visualization of all IND~pressure-model results relevant for the scoring*

---

## Description

plot_model creates a tibble with up to 4 individual plots and one combined plot (all ggplot2 objects) for each IND~pressure pair in the input tibble. The number of plots generated depends on the information provided in the input tibble. If all model IND~pressure modeling functions have been applied to create the final input tibble all five plots will be produced.

## Usage

```
plot_model(
  init_tbl,
  mod_tbl,
  choose_thresh_gam = NULL,
  pos_label = "topleft",
  header = TRUE
)
```

## Arguments

| | |
|---|---|
| init_tbl | The output tibble of the ind_init function. |
| mod_tbl | Any output tibble from the IND~pressure modeling functions. |
| choose_thresh_gam | Selects the threshold_GAM for the thresh_plot, which is relevant if several models are listed in 'thresh_models'.<br>The default is NULL, which shows the best performing threshold_GAM (based on the GCV as selection criterion). |
| pos_label | Specifies the position of the annotation in the plot. Should be one of "topleft" (default), "topright", "bottomleft" or "bottomright". For more details see place_text. |
| header | logical; if TRUE, each plot will have a header including the IND name, pressure name(s) and the model type. |

## Value

The function returns a tibble, including the following elements:

id Numerical IDs of the IND~press combinations.

ind Indicator names.

press Pressure names.

response_plot A list-column of ggplot2 objects that show the observed (black points) and predicted IND response to the single pressure (based on the training data). The solid blue line represents the predicted mean and the transparent polygon the 95% confidence interval. The effective degrees of freedom (edf), R_sq, and p-value from the fitted model are additionally provided. The input needed for this plot is generated from the model_gam or model_gamm functions.

predict_plot A list-column of ggplot2 objects that show the robustness of the modeled relationship expressed as the predictive performance (the NRMSE) on a test dataset, e.g the last years of the time series. The solid green line represents the predicted IND value given the observed pressure value for that particular year (both in the training and test data, the latter displayed as green triangles). The transparent polygon represents the 95% confidence interval. Observed IND values of the test data are shown as black triangle, the trainings observations are presented as black circles. The input needed for this plot is generated from the model_gam or model_gamm functions.

deriv_plot A list-column of ggplot2 objects that show the first derivatives (S') of non-linear IND~pressure response curves (edf > 1.5) and the proportion of the pressure range where the IND shows no further significant change (i.e., slope approximates zero). Black triangles represent values at the pressure's boundary where the zero line falls into the confidence interval, which indicates no further significant IND change. Circle represent values that were considered positive for the calculation of the pressure range (see for more details calc_deriv). The input needed for this plot is generated from the calc_deriv function.

thresh_plot A list-column of ggplot2 objects that show the observed IND response curve for a specific pressure under a low (left panel, in black) and high (right panel, in red) regime of an interacting 2nd pressure variable. The solid lines represent the predicted mean and the transparent polygons the 95% confidence intervals. Filled circles represent the observed

training observations in each regime. If no thresh_plot is created for that IND~pressure pair, no interaction was found. If more than one interacting pressure variable has been detected, i.e. more than one threshold-GAM performed better than its corresponding GAM, the threshold-GAM with the best GCV will be displayed. The input needed for this plot is generated from the test_interaction function. If the plot shows strange patterns such as smoothers hardly differ in both regimes with wide confidence intervals at the edge or few data points in one regime check the model diagnostics of this threshold model! Outliers can cause such patterns or if threshold is at the edge of the pressure range or other thresholds are similarly likely (see also plot_diagnostics.)

all_plots A list-column of ggplot2 objects that show all plots together using additional drawing canvas from the cowplot package on top of ggplot2.

## See Also

Other IND~pressure modeling functions: find_id(), ind_init(), model_gamm(), model_gam(), plot_diagnostics(), scoring(), select_model(), test_interaction()

## Examples

```
# Using some models of the Baltic Sea demo data in this package
mod_tbl <- all_results_ex[4:5, ]
init_tbl <- ind_init_ex[4:5, ]
dat <- plot_model(init_tbl, mod_tbl, pos_label = "topleft")
dat$response_plot[[1]]
dat$predict_plot[[1]]
dat$deriv_plot[[2]]
dat$thresh_plot[[2]]
dat$all_plots[[2]]


 # Apply function to all sign. models and save specific plots
 id <- which(all_results_ex$p_val <= 0.05)
 init_tbl <- ind_init_ex[id, ]
 mod_tbl <- all_results_ex[id, ]
 dat <- plot_model(init_tbl, mod_tbl, pos_label = "bottomright")
 pdf(file.path(tempdir(), "Plot.pdf"), height=10, width=10)
 dat$all_plots
 dev.off()
```

---

# plot_spiechart *Create score-based spie chart*

---

## Description

plot_spiechart generates for each indicator in the scoring tibble a ggplot2-based spie chart to visualize the scores of each criterion.

## Usage

```
plot_spiechart(
  summary_tbl,
  col_press_type = NULL,
  col_crit8_11 = NULL,
  lab_size = 6,
  title_size = 8
)
```

## Arguments

| | |
|---|---|
| summary_tbl | The output tibble from the summary_sc function. |
| col_press_type | Colors for the spie chart slices representing criteria 9 (sensitivity; opaque) and 9 (robustness; transparent).<br>The colors distinguish the different pressure types. The default is set to the RColourBrewer palette "Set1". |
| col_crit8_11 | Colors for the spie chart slices representing criteria 8 (trend) and 11 (management application). The default is set to cyan1 and yellow2. |
| lab_size | Size for the labels naming the significant pressures. The default is 6. |
| title_size | Size for the title naming the indicator. The default is 8. |

## Details

The overall performance of each tested IND is illustrated using a spie chart, which has been shown to be a well-suited graphical tool for displaying multivariate data in comparative indicator evaluations (Stafoggia *et al.*, 2011). A spie chart superimposes a normal pie chart with a modified polar area chart to permit the comparison of two sets of related data, e.g. the maximum achievable scores and each IND's realized scores. In this function, the slice width is kept constant, while the length of the slices represents the percentage of scores achieved, with the boundary line (i.e. the inner gray circle) indicating the full 100

The two unlabeled slices at the top represent the trend (right) and management (left) criteria. The sensitivity and robustness scores are shown individually for each pressure where a significant relationship was found. These are the labeled slices grouped by their pressure type represented by dotted division lines and the segmented outer gray circle as well as pressure type-specific colors (sensitivity scores are displayed in opaque color, robustness scores in transparent color).

The plot slices adjust to the number of criteria used for the scoring function (that are present in crit_scores_tmpl).

## Value

The function returns a list of ggplot objects.

## References

Stafoggia, M., Lallo, A., Fusco, D., Barone, A.P., D'Ovidio, M., Sorge, C., Perucci, C.A. (2011) Spie charts, target plots, and radar plots for displaying comparative outcomes of health care. *Journal of Clinical Epidemiology* 64, 770-778.

## See Also

Other score-based IND performance functions: clust_sc(), dist_sc_group(), dist_sc(), expect_resp(), plot_clust_sc(), scoring(), summary_sc()

## Examples

```
# Using the Baltic Sea demo data in this package
scores_tbl <- scoring(trend_tbl = model_trend_ex,
  mod_tbl = all_results_ex, press_type = press_type_ex)
summary_tbl <- summary_sc(scores_tbl)
p <- plot_spiechart(summary_tbl)
p$TZA


 # Show all spiecharts together
 gridExtra::grid.arrange(grobs = p)

# To modify the plot
p <- plot_spiechart(summary_tbl, col_crit8_11 = c("black",
  "thistle4"), col_press_type = RColorBrewer::brewer.pal(3,
  name = "Accent"), lab_size = 4, title_size = 4)
gridExtra::grid.arrange(grobs = p)

# Remove pressure-independent criteria for the plot (e.g.
# management) (easiest in the score tibble)
scores_tbl$C11 <- NULL
summary_tbl <- summary_sc(scores_tbl)
p <- plot_spiechart(summary_tbl)
gridExtra::grid.arrange(grobs = p)

# Exclude additionally one pressure-specific criterion
 # (e.g. sensitivity C9) (easiest by removing columns
 # in the first list of the summary)
summary_tbl[[1]] <- summary_tbl[[1]][ ,
  ! names(summary_tbl[[1]]) %in% c("C9", "C9_in%")]
p <- plot_spiechart(summary_tbl)
gridExtra::grid.arrange(grobs = p)
```

---

# plot_statespace_ch *Convex hull plot*

---

## Description

plot_statespace_ch generates a scatter plot of all observed combinations in the 2-dimensional indicator space including the convex hull from defined reference conditions and the current period.

## Usage

```
plot_statespace_ch(x, col_ch_ref = "red", col_ch_cur = "blue", size_time = 4)
```

### Arguments

| | |
|---|---|
| x | An object from the statespace_ch function. |
| col_ch_ref | Color of reference period (for points, path, and labels). |
| col_ch_cur | Color of current period (for points, path, and labels). |
| size_time | Text size of the time labels (both periods). |

### Value

The function returns a ggplot object.

### See Also

Other state assessment functions: plot_statespace_ed(), statespace_ch(), statespace_ed()

### Examples

```
# Using the Baltic Sea demo data in the package
x <- statespace_ch(x = ind_ex$TZA, y = ind_ex$MS,
  time = ind_ex$Year, period_ref = 1979:1983,
  period_current = 2004:2008)
plot_statespace_ch(x)

# To modify the plot:
p <- plot_statespace_ch(x, col_ch_ref = "green4", col_ch_cur = "orange3",
  size_time = 6)
p + ggplot2::xlab("TZA") + ggplot2::ylab("MS") +
  ggplot2::ggtitle("Deviation of current state from reference period") +
  ggplot2::theme(plot.title = ggplot2::element_text(size = 10)) +
  ggplot2::theme(axis.text = ggplot2::element_text(size = 6),
    axis.title = ggplot2::element_text(size = 20))
```

---

## plot_statespace_ed *Time series plot of Euclidean distance*

---

### Description

plot_statespace_ed generates a time series plot of the Euclidean distance in indicator state space from a defined reference conditions.

### Usage

```
plot_statespace_ed(x)
```

### Arguments

| | |
|---|---|
| x | The output tibble from the statespace_ed function. |

## Value

The function returns a ggplot object.

## See Also

Other state assessment functions: plot_statespace_ch(), statespace_ch(), statespace_ed()

## Examples

```
# Using the Baltic Sea demo data in the package
ind_sel <- ind_ex[,c(2,3,4,8,10,11)]
  # --> selection of complementary and well performing indicators
ed <- statespace_ed(x = ind_sel, time = ind_ex$Year, ref_time = ind_ex$Year[1])
plot_statespace_ed(x = ed)

# To modify the plot:
p <- plot_statespace_ed(x = ed)
p + ggplot2::geom_point(col = "red") +
  ggplot2::ylab("Eucl. Distance") +
  ggplot2::geom_smooth(col="blue") +
  ggplot2::theme(axis.text = ggplot2::element_text(size = 16),
    axis.title=ggplot2::element_text(size = 18))
```

---

# plot_trend *Create indicator trend plot*

---

## Description

plot_trend creates for each indicator (IND) in the input tibble a time series plot including the smoothed trend with 95% confidence interval and the corresponding p- value based on the IND ~ time GAM.

## Usage

```
plot_trend(trend_tbl, pos_label = "topleft")
```

## Arguments

| | |
|---|---|
| trend_tbl | Output tibble from the model_trend function. |
| pos_label | Specifies the position of the annotation in the plot. Should be one of "topleft" (default), "topright", "bottomleft" or "bottomright". For more details see place_text. |

## Value

The function returns a list of ggplot objects; one for each indicator.

## See Also

model_trend that generates the model tibble for this function

## Examples

```
# Using the example data
trend_tbl <- model_trend_ex
pt <- plot_trend(trend_tbl)
# Show single plots using indicator names or indices
pt[[2]]
pt$Sprat
# Show all plots together
gridExtra::grid.arrange(grobs = pt)
```

---

`press_ex` *Environmental variables representing pressures for pelagic food webs in the Central Baltic Sea (Bornholm Basin).*

---

## Description

Environmental variables representing pressures for pelagic food webs in the Central Baltic Sea (Bornholm Basin).

## Usage

```
press_ex
```

## Format

A data frame with 30 rows and 8 variables. These indicator time series represent a slight modification of the original time series represented in Otto et al. (2018)

**Year** year given as integer

**Tsum** temperature summer (in degree C)

**Swin** salinity winter

**Pwin** phosphate in winter (in mg m^-3^)

**Nwin** nitrogen winter (in mg m^-3^)

**Fsprat** fishing mortality of sprat

**Fher** fishing mortality of herring

**Fcod** fishing mortality of cod

## press_type_ex — *Pressure variables and their associated pressure types from our example data*

### Description

Pressure variables and their associated pressure types from our example data

### Usage

```
press_type_ex
```

### Format

A data frame with 7 rows and 2 variables:

**press** pressure name

**press_type** corresponding pressure type

## scoring — *Scoring of indicator performance*

### Description

Scoring of each indicator based on an internal criterion-scoring scheme applied on the output of the trend and pressure model functions.

### Usage

```
scoring(
  trend_tbl = NULL,
  mod_tbl,
  press_type = NULL,
  crit_scores = INDperform::crit_scores_tmpl,
  sign_level = 0.05
)
```

### Arguments

| | |
|---|---|
| trend_tbl | Output tibble from the model_trend function. |
| mod_tbl | Output tibble from the IND~pressure modeling functions. |
| press_type | Data frame or tibble with pressure names (named 'press') in first column and corresponding pressure types in second column (named 'press_type'). Needed for the spie chart! (see for an example press_type_ex) |

`crit_scores` Internal tibble of (sub)criteria and respective scores named `crit_scores_tmpl`; can be modified by saving this data frame as new object and removing single (sub)criteria or assigning weights (default is 1). The variable 'condition' represents a list of single elements or vectors with various elements to base the scoring on. This can be modified but needs to follow the same syntax.

`sign_level` Significance level on which scoring is built; default is 0.05.

## Details

Among the 16 common indicator selection criteria summarized in Otto *et al.* (2018) five criteria relate to the indicators' performances and require time series for their evaluation, i.e.

- Crit. 8: Development reflects ecosystem change caused by variation in manageable pressure(s)
- Crit. 9: Sensitive or responsive to pressures
- Crit. 10: Robust, i.e. responses in a predictive fashion, and statistically sound
- Crit. 11: Links to management measures (responsiveness and specificity)
- Crit. 12: Relates where appropriate to other indicators but is not redundant (not scored)

In this function, the scoring scheme for these criteria as proposed by Otto *et al.* (2018) serves as basis for the quantification of the IND performance. Sensitivity (criterion 9) and robustness (criterion 10) are specified into more detailed sub-criteria to allow for quantification based on statistical models and rated individually for every potential pressure that might affect the IND directly or indirectly. In the case of non-significant relationships between a IND and a specific pressures, sub-crit. 9.1 and all following pressure-specific sub-crit. in criteria 9 and 10 are scored zero for this pressure.

The template tibble `crit_scores_tmpl` contains all relevant informations and serves as basis for the scoring in this function. See for more details `crit_scores_tmpl` or `View(crit_scores_tmpl)`. The scoring scheme can easily be adapted to any kind of state indicator and management scheme by modifying the scores, the weighting of scores or by removing or adding (sub)criteria in the `crit_scores_tmpl` template. The `condition` variable can also be modified but needs to follow the same syntax.

## Value

The function returns a nested tibble with the following elements depending on the criteria rated

`ind` A vector of the indicator names.

`C8` **and/or** `C11` A vector of IND-specific scores for criterion 8 (trend indication) and/or C11 (management application).

`press_spec_sc` A list-column of IND-specific data frames containing pressure-specific scores for the sub-criteria 9.1-9.2, 10.1-10.4.

The tibble can easily be unnested by using the `unnest` function. That is, each element of the data frame in the list-column `press_spec_sc` becomes its own row in the tibble.

## References

Otto, S.A., Kadin, M., Casini, M., Torres, M.A., Blenckner, T. (2018) A quantitative framework for selecting and validating food web indicators. *Ecological Indicators*, 84: 619-631, doi: https://doi.org/10.1016/j.ecolind.2017.

## See Also

unnest to make each element of the data frame in the list-column press_spec_sc its own row and nest for the inverse operation

Other score-based IND performance functions: clust_sc(), dist_sc_group(), dist_sc(), expect_resp(), plot_clust_sc(), plot_spiechart(), summary_sc()

Other IND~pressure modeling functions: find_id(), ind_init(), model_gamm(), model_gam(), plot_diagnostics(), plot_model(), select_model(), test_interaction()

## Examples

```
# Using the Baltic Sea demo data in this package
scores_tbl <- scoring(trend_tbl = model_trend_ex, mod_tbl = all_results_ex,
  press_type = press_type_ex)
## Not run:
 # To see the criterion template and change it potentially
 View(crit_scores_tmpl)
 # E.g. exclude the trend criterion
 crit_scores_tmpl_new <- crit_scores_tmpl[crit_scores_tmpl$crit_id > 1, ]
 # Now the trend tibble is not needed anymore
 scores_tbl <- scoring(mod_tbl = all_results_ex, press_type = press_type_ex,
  crit_scores = crit_scores_tmpl_new)

## End(Not run)
```

---

# select_interaction *Create tibble of all potential pressure combinations to test for interactions.*

---

## Description

select_interaction is a helper function that creates a tibble with all indicator-specific combinations of pressures pairs as input for the test_interaction function. The pressures in the IND~pressure GAM(M)s are combined with all other pressures in the model tibble. If specific combinations should not be modeled simply delete them from this data frame.

## Usage

```
select_interaction(mod_tbl)
```

## Arguments

| | |
|---|---|
| mod_tbl | A model output tibble from model_gam, select_model, merge_models or calc_deriv representing the best model for each IND~pressure pair. |

### Details

For each IND~pressure pair specific pressures to test for interactions can be selected by creating a tibble containing the IND (termed 'ind'), the pressure 1 (termed 'press') and the pressure 2 (termed 't_var'). The easiest is to use the helper function `select_interaction`: it creates all combinations of IND~press pairs and the threshold variables based on the input model tibble. If specific combinations should not be modeled simply delete them from this data frame.

### Value

The functions returns a tibble with three columns:

`ind` The name of the indicator

`press` The pressure name for the smooth term

`t_var` The pressure name for the threshold variable

### Examples

```
# Using some models of the Baltic Sea demo data
test <- select_interaction(mod_tbl = merge_models_ex[1:5,])
```

---

## select_model *Select the best correlation structure in the GAMM*

---

### Description

The function selects and returns the best GAMM out of the six GAMMs computed in `model_gamm`. In the case that the GAMM without any correlation structure performs best, the output tibble contains the information from the original `model_gam` output tibble (therefore needed as input).

### Usage

```
select_model(gam_tbl, gamm_tbl)
```

### Arguments

| | |
|---|---|
| `gam_tbl` | Output tibble from the `model_gam` function. |
| `gamm_tbl` | Output tibble from the `model_gamm` function. |

### Details

The best error structure is chosen here based on the Akaike's Information Criterion (AIC). The GAMM with the lowest AIC value is selected, but only if the AIC difference to the GAMMs with a less complex error structure is greater than 2 (or respectively 4 or 6 depending on the level of nested complexity) (Burnham and Anderson, 2002). Otherwise the less complex GAMM is chosen. The following hierarchy of complexity is considered:

- no structure < AR1 < AR2 and ARMA1,1 < ARMA2,1 and ARMA1,2

## Value

`select_model` returns the same model output tibble as `model_gamm` but with only **one** final GAMM for each filtered IND~pressure pair.

## References

Burnham, K.P., Anderson, D.R. (2002) Model Selection and Multimodel Inference - A Practical Information-Theoretic Approach. Springer, New York.

## See Also

Other IND~pressure modeling functions: `find_id()`, `ind_init()`, `model_gamm()`, `model_gam()`, `plot_diagnostics()`, `plot_model()`, `scoring()`, `test_interaction()`

## Examples

```
# Using some models of the Baltic Sea demo data
test_ids <- c(67:70)
gam_tbl <- model_gam_ex[model_gam_ex$id %in% test_ids,]
gamm_tbl <- model_gamm(ind_init_ex[test_ids,], filter = gam_tbl$tac)
best_gamm <- select_model(gam_tbl, gamm_tbl)
```

---

# statespace_ch *Convex hull in 2-dimensional space*

---

## Description

`statespace_ch` calculates the convex hull in 2-dimensional space, e.g. for two selected indicators, using the `tri.mesh` function.

## Usage

```
statespace_ch(x, y, time, period_ref, period_current)
```

## Arguments

| | |
|---|---|
| `x` | The coordinates of points in the first dimension (e.g. indicator 1 or PC1 scores from a PCA). |
| `y` | The coordinates of points in the second dimension (e.g. indicator 2 or the PC2 scores from a PCA). |
| `time` | A vector containing the actual time series. |
| `period_ref` | Vector of time units (e.g. years) used as reference period (minimum of 3 time units required). |
| `period_current` | Vector of time units (e.g. years) used as current period to compare with the reference period (minimum of 3 time units required). |

## Details

statespace_ch implements a second state space approach to assess the development of a suite of ecological state indicators (Otto *et al.* 2018, Tett *et al.* 2008). While unidimensional approaches such as the Euclidean distance (see statespace_ed) feature the disadvantage of defining one particular year or time step as reference condition, this approach accounts for inter-annual variation by defining a reference domain in state space based on several years: more recent observations might be characterized as either within or outside this domain.

The reference domain can be described by a convex hull, which is a multivariate measure derived from computational geometry representing the smallest convex set containing all the points in Euclidean plane or in Euclidean space (de Berg *et al.*, 2008). While the convex hull can be calculated for high-dimensional data, reducing the space to two dimensions allows for an easier visualization and interpretation. Therefore, the statespace_ch function only calculates the convex hull for two dimensions, i.e. for two indicators or principal axes obtained by multivariate analysis such as a Principal Component Analysis (PCA).

## Value

The function returns a list with the following elements

ch_ref A vector of the position of the convex hull of the reference period.

ch_cur A vector of the position of the convex hull of the current period.

inside_ch_ref A logical vector indicating whether each year (time step) of the current period lies inside (TRUE) or outside (FALSE) the state space domain of the reference period.

xy A data frame of the x and y coordinates.

time A vector of the full time series.

period_ref A vector of years (time steps) defined as the reference period.

period_current A vector of years (time steps) defined as the current period.

## References

de Berg, M., Cheong, O., van Kreveld, M., Overmars, M. (2008) Computational Geometry - Algorithms and Applications. Springer Berlin Heidelberg, 386pp.

Otto, S.A., Kadin, M., Casini, M., Torres, M.A., Blenckner, T. (2018) A quantitative framework for selecting and validating food web indicators. *Ecological Indicators*, 84: 619-631, doi: https://doi.org/10.1016/j.ecolind.2017.

Tett, P., Carreira, C., Mills, D.K., van Leeuwen, S., Foden, J., Bresnan, E., Gowen, R.J. (2008) Use of a Phytoplankton Community Index to assess the health of coastal waters. *ICES Journal of Marine Science* 65, 1475-1482.

## See Also

tri.mesh for the computation of the convex hull.

Other state assessment functions: plot_statespace_ch(), plot_statespace_ed(), statespace_ed()

## Examples

```
# Using the Baltic Sea demo data in the package
time <- ind_ex$Year
period_ref <- 1979:1983
period_current <- 2004:2008

# Apply function on 2 indicators
ch <- statespace_ch(x = ind_ex$TZA, y = ind_ex$MS,
 time, period_ref, period_current)

# Conduct PCA on selected indicators using the correlation matrix (scale=T)
ind_sel <- ind_ex[,c(2,3,4,8,10,11)]
pca_out <- vegan::rda(ind_sel, scale=TRUE)
pca_sum <- summary(pca_out)
prop_expl <- as.vector(pca_sum$cont$importance[2,])
scores_unsc <- vegan::scores(pca_out, scaling = 0)
scores_sites <- as.data.frame(scores_unsc$sites)
x <- scores_sites$PC1
y <- scores_sites$PC2
# Apply function
ch <- statespace_ch(x, y, time, period_ref, period_current)
```

---

# statespace_ed *Euclidean distance in state space*

---

## Description

`statespace_ed` generates a time series of Euclidean distances from defined reference conditions to assess the development of a suite of ecological state indicators.

## Usage

```
statespace_ed(x, time, ref_time = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A data frame, tibble, matrix or vector of selected indicator(s). |
| `time` | A vector containing the actual time series. |
| `ref_time` | The reference time (single point in time, e.g. specific year) on which to base the Euclidean distance. Default is set to the first time point. |

## Details

This function implements an approach adopted from Tett *et al.* (2013) to assess changes in the ecosystem state by studying trajectories in state space. State space is defined here as the n-dimensional space of possible locations of state variables or indicators. For a robust suite of indicators the uni-dimensional Euclidean distance between each year (or any other time step) and a reference year in state space is calculated. That means, the function calculates the square root of the sum of squared distances between each standardized indicator value in a specific year and its reference value, which is defined by the user.

## Value

`statespace_ed` returns a tibble with the time vector `time`, the Euclidean distance `ed`, and a logical vector `ref_time` indicating the time step defined as reference.

## References

Tett, P., Gowen, R.J., Painting, S.J., Elliott, M., Forster, R., Mills, D.K., Bresnan, E., Capuzzo, E., Fernandes, T.F., Foden, J., Geider, R.J., Gilpin, L.C., Huxham, M., McQuatters-Gollop, A.L., Malcolm, S.J., Saux-Picart, S., Platt, T., Racault, M.F., Sathyendranath, S., van der Molen, J., Wilkinson, M. (2013) Framework for understanding marine ecosystem health. *Marine Ecology Progress Series.* 494, 1-27.

## See Also

Other state assessment functions: `plot_statespace_ch()`, `plot_statespace_ed()`, `statespace_ch()`

## Examples

```
# Using the Baltic Sea demo data in the package
ind_sel <- ind_ex[,c(2,3,4,8,10,11)]
# --> selection of complementary and well performing indicators
# There are different ways to define the reference time step:
ed <- statespace_ed(x = ind_sel, time = ind_ex$Year, ref_time = ind_ex$Year[1])
ed <- statespace_ed(x = ind_sel, time = ind_ex$Year, ref_time = 1987)
ed <- statespace_ed(x = ind_sel, time = ind_ex$Year, ref_time = "1987")
```

---

# `summary_sc` *Summary of indicator performance scores*

---

## Description

Summarizes the scoring output tibble so that IND-specific scores for each criterion as well as the pressure-specific sub-criteria scores (in crit. 9 and 10) can be easily compared and used for further score-based IND performance functions.

## Usage

```
summary_sc(scores_tbl, crit_scores = INDperform::crit_scores_tmpl)
```

## Arguments

| | |
|---|---|
| `scores_tbl` | The output tibble from the `scoring` function. |
| `crit_scores` | The (un)modified criterion-scoring template `crit_scores_tmpl`; required to calculate the scores in percentage. Has to be the same than used in `scoring`. Default is the unmodified template `crit_scores_tmpl`. |

## Value

The function returns a list of 3 data frames

`overview` IND-specific scores and percentages from max. score for all criteria (crit 9 and 10 averaged across all sign. pressures and the number of significant pressures).

`subcriteria_per_press` IND- and pressure-specific scores for all (sub-)criteria and the percentages from max. criterion score.

`scores_matrix` TEXT

## See Also

Other score-based IND performance functions: `clust_sc()`, `dist_sc_group()`, `dist_sc()`, `expect_resp()`, `plot_clust_sc()`, `plot_spiechart()`, `scoring()`

## Examples

```
# Using the Baltic Sea demo data in this package
scores_tbl <- scoring(trend_tbl = model_trend_ex, mod_tbl = all_results_ex,
  press_type = press_type_ex)
summary_sc(scores_tbl)
```

---

# test_interaction *Test for interactions between 2 pressure variables*

---

## Description

`test_interaction` tests for each significant GAM(M) whether any other pressure modifies the IND response to the original pressure using a threshold formulation of the GAM.

## Usage

```
test_interaction(
  init_tbl,
  mod_tbl,
  interactions,
  sign_level = 0.05,
  k = 4,
  a = 0.2,
  b = 0.8,
  excl_outlier = FALSE
)
```

## Arguments

| | |
|---|---|
| init_tbl | The (full) output tibble of the ind_init function. |
| mod_tbl | A model output tibble from model_gam, select_model, merge_models or calc_deriv representing the best model for each IND~pressure pair. |
| interactions | A tibble of all potential pressure combinations to test for interactions for specific INDs. |
| sign_level | Significance level for selecting models on which to test for interactions. Only models with a p value <= sign_level will be selected; the default is 0.05. |
| k | Choice of knots (for the smoothing function s); the default is here 4 to avoid over-parameterization. |
| a | The lower quantile value of the selected threshold variable, which the estimated threshold is not allowed to exceed; the default is 0.2. |
| b | The upper quantile value of the selected threshold variable, which the estimated threshold is not allowed to exceed; the default is 0.8. |
| excl_outlier | logical; if TRUE, the outliers excluded in the original models will be also excluded in the interaction models. |

## Details

### Threshold-GAMs

To identify potential interactions between pressures (relevant for sub-crit. 10.4), threshold formulations are applied to every significant IND-pressure GAM. Threshold-GAMs or TGAMs represent a special type of varying-coefficient models and were first introduced by Ciannelli *et al.* (2004). In varying-coefficient models coefficients are allowed to vary as a smooth functions of other variables (Hastie and Tibshirani, 1993), which allows to detect interactions between two external pressures. Threshold-GAMs are particularly useful if the response to the interacting pressure variables is not continuous but rather step-wise. They have been applied to data from real ecosystems to model population dynamics (e.g. Otto *et al.*, 2014) up to food web dynamics in response to climate, nutrient, and fishing interactions (e.g. Llope *et al.*, 2011). The following threshold formulation is applied to every sign. IND-Pressure GAM:

$$IND_t = \alpha_1 + s_1(press1_t) + \epsilon_t if press2 <= r$$

$$IND_t = \alpha_2 + s_2(press1_t) + \epsilon_t if press2 > r$$

where the thin-plate regression spline function $s$ can differ depending on whether pressure 2 is below or above a threshold $r$ with possible changes in the intercept (from *a_1* to *a_2*). The index $t$ represents each observation in the training data. The threshold formulation can be implemented in the GAM by including two smoothing functions for the same pressure and using the by argument in the smoothing function s to select specific observations. The threshold itself is estimated from the data and chosen by minimizing the GCV score (termed "gcvv" in the threshold-GAM object) over an interval defined by the lower and upper quantiles (see the a and b arguments respectively) of Pressure 2.

### Selection of threshold-GAMs

To compare the performance of a GAM without interactions with the threshold-GAM we implemented a leave-one-out cross-validation (LOOCV) approach as suggested by Ciannelli *et al.* (2004).

LOOCV involves splitting the full set of observations n (i.e. the training data defined in ind_init) into two parts: a new training set containing n-1 observations and a test set containing 1 observations. The threshold-GAM and corresponding GAM are then fit on the new training data and a prediction is made for the excluded test observation using the corresponding pressure 1 and pressure 2 values for that time step. A square prediction error is then calculated for each predicted test observation. Repeating this approach n times produces n squared errors, MSE_1, . . . , MSE_n. The LOOCV estimate for the test MSE, also termed (genuine) cross-validatory squared prediction error, is the root of the average of these n test error estimates. This approach properly accounts for the estimation of the threshold value as well as the effective degrees of freedom of all model terms.

**Implementation of threshold modeling**

For each IND~pressure pair, specific pressures to test for interactions can be selected by creating a tibble containing the IND (termed 'ind'), the pressure 1 (termed 'press') and the pressure 2 (termed 't_var'). The easiest is to use the helper function select_interaction: it creates all combinations of IND~press pairs and the threshold variables based on the input model tibble. If specific combinations should not be modeled simply delete them from this data frame.

test_interaction takes this data frame and the ind_init and model tibble as input and applies the following procedure:

- Filters significant GAMs and the corresponding IND ~ pressure | threshold pressure combination.
- Extracts all data needed for the model, excluding outliers if set to TRUE.
- Computes the LOOCV for each IND ~ pressure | threshold pressure model (threshold-GAM and GAM).
- If the LOOCV estimate of the threshold-GAM is better than its corresponding GAM the threshold-GAM and its model output are saved in the returned model tibble. Note, however, that it is crucial to inspect the model diagnostic plots (i.e. the $thresh_plot from the plot_diagnostics function! The performance of the final model (in terms of its Generalized Cross-Validation value) might be not much lower than for models with other thresholds indicating a lack of robustness. In this case, the interaction might need to be ignored but that needs to be decided on a case-by-case basis.

There is no threshold-GAMM implemented in this package yet. Until then, threshold-GAMs are also applied to GAMMs when GAM residuals show temporal autocorrelation (TAC). However, the residuals of threshold GAMs often show less TAC due to the splitting of observations into a low and high regime of the threshold variable. In the case of significant TAC (indicated by the output variable tac_in_thresh) the user can decide whether that interaction should be neglected and modify the interaction) output variable accordingly.

## Value

The function returns the input model tibble 'mod_tbl' with the following 5 columns added:

interaction logical; if TRUE, at least one thresh_gam performs better than its corresponding gam based on LOOCV value.

thresh_var A list-column with the threshold variables of the better performing thresh_models.

thresh_models A list-column with nested lists containing the better performing thresh_models.

`thresh_error` A list-column capturing potential error messages that occurred as side effects when fitting each threshold GAMs and performing the LOOCV.

`tac_in_thresh` logical vector; indicates for every listed thresh_model whether temporal autocorrelation (TAC) was detected in the residuals. TRUE if model residuals show TAC.

## References

Ciannelli, L., Chan, K.-S., Bailey, K.M., Stenseth, N.C. (2004) Nonadditive effects of the environment on the survival of a large marine fish population. *Ecology* 85, 3418-3427.

Hastie, T., Tibshirani, R. (1993) Varying-Coefficient Models. *Journal of the Royal Statistical Society. Series B (Methodological)* 55, 757-796.

Llope, M., Daskalov, G.M., Rouyer, T.A., Mihneva, V., Chan, K.-S., Grishin, A.N., Stenseth, N.C. (2011) Overfishing of top predators eroded the resilience of the Black Sea system regardless of the climate and anthropogenic conditions. *Global Change Biology* 17, 1251-1265.

Otto, S.A., Kornilovs, G., Llope, M., Möllmann, C. (2014) Interactions among density, climate, and food web effects determine long-term life cycle dynamics of a key copepod. *Marine Ecology Progress Series* 498, 73-84.

## See Also

`plot_diagnostics` for assessing the model diagnostics

Other IND~pressure modeling functions: `find_id()`, `ind_init()`, `model_gamm()`, `model_gam()`, `plot_diagnostics()`, `plot_model()`, `scoring()`, `select_model()`

## Examples

```
# Using some models of the Baltic Sea demo data in this package
init_tbl <- ind_init_ex
mod_tbl <- merge_models_ex[c(5:7),]
interactions <- select_interaction(mod_tbl)
test <- test_interaction(init_tbl, mod_tbl, interactions)

# if only one combination should be tested
interactions <- select_interaction(mod_tbl)[1:2, ]
test <- test_interaction(init_tbl, mod_tbl, interactions)

# Determine manually what to test for (e.g. only TZA ~ Fsprat | Pwin)
interactions <- tibble::tibble(ind = "TZA",
                               press = "Fsprat",
                               t_var = "Pwin" )
test <- test_interaction(init_tbl, mod_tbl, interactions)
```

# Index