Package 'rtrim'

November 28, 2016

Author Patrick Bogaart, Mark van der Loo, and Jeroen Pannekoek

Maintainer Adriaan Gmelig Meyling <rtrim@cbs.nl>

License EUPL

Title Trends and Indices for Monitoring Data

LazyData no

Type Package

LazyLoad yes

Description The TRIM model is widely used for estimating growth and decline of animal populations based on (possibly sparsely available) count data. The current package is a reimplementation of the original TRIM software developed at Statistics Netherlands by Jeroen Pannekoek. See <https://www.cbs.nl/en-gb/society/nature-and-environment/indices-and-trends--trim--> for more information about TRIM.

Version 1.0.1

Imports methods, utils, stats, graphics, grDevices

URL https://github.com/markvanderloo/rtrim

BugReports https://github.com/markvanderloo/rtrim/issues

Date 2016-11-21

Suggests testthat, knitr, rmarkdown

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-28 16:54:05

R topics documented:

`rtrim-package` *Trend and Indices for Monitoring Data*

Description

The TRIM model is used to estimate species populations based on frequent (annual) counts at a varying collection of sites. The model is able to take account of missing data by imputing prior to estimation of population totals. The current package is a complete re-implementation of the `Delphi` based TRIM software developed at Statistics Netherlands by Jeroen Pannekoek.

Getting started

Users of the original TRIM software can get started by following the rtrim for trim users or following the TRIM by example manuals.

check_observations *Check whether there are sufficient observations to run a model*

Description

Check whether there are sufficient observations to run a model

Usage

```
check_observations(x, ...)

## S3 method for class 'data.frame'
check_observations(x, model, covars = list(),
  changepoints = numeric(0), time.id = "time", count.id = "count",
  eps = 1e-08, ...)

## S3 method for class 'trimcommand'
check_observations(x, ...)

## S3 method for class 'character'
check_observations(x, ...)
```

Arguments

x	A trimcommand object, a data.frame, or the location of a TRIM command file.
...	Parameters passed to other methods.
model	[numeric] Model 1, 2 or 3?
covars	[character|numeric] column index of covariates in x
changepoints	[numeric] Changepoints (model 2 only)
time.id	[character|numeric] column index of time points in x
count.id	[character|numeric] column index of the counts in x
eps	[numeric] Numbers whose absolute magnitude are lesser than eps are considered zero.

Value

A list with two components. The component sufficient takes the value TRUE or FALSE depending on whether sufficient counts have been found. The component errors is a list, of which the structure depends on the chosen model, that indicates under what conditions insufficient data is present to estimate the model.

- For model 3 without covariates, $errors is a list whose single element is a vector of time points with insufficient counts.

- For model 3 with covariates, `$errors` is a named list with an element for each covariate for which insufficients counts are encountered. Each element is a two-column `data.frame`. The first column indicates the time point, the second column indicates for which covariate value insufficient counts are found.
- For Model 2, without covariates `$errors` is a list with a single element `changepoints`. It points out what changepoints lead to a time slice with zero observations.
- For Model 2, with covariates `$errors` is a named list with an element for each covariate for which inssufficients counts are encountered. Each element is a two-column `data.frame`, The first colum indicates the changepoint, the second column indicates for which covariate value insufficient counts are found.

See Also

Other modelspec: `read_tcf`, `read_tdf`, `set_trim_verbose`, `trimcommand`, `trim`

`coef.trim` *Extract TRIM model coefficients.*

Description

Extract TRIM model coefficients.

Usage

```
## S3 method for class 'trim'
coef(object, representation = c("standard", "trend",
  "deviations"), ...)
```

Arguments

`object`	TRIM output structure (i.e., output of a call to `trim`)
`representation`	`[character]` Choose the coefficient representation. Options `"trend"` and `"deviations"` are for model 3 only.
`...`	currently unused

Value

A `data.frame` containing coefficients and their standard errors, both in additive and multiplicative form.

Details

Extract the site, growth or time effect parameters computed with `trim`.

Additive versus multiplicative representation

In the simplest cases (no covariates, no change points), the trim Model 2 and Model 3 can be summarized as follows:

- Model 2: $\ln \mu_{ij} = \alpha_i + \beta \times (j-1)$
- Model 3: $\ln \mu_{ij} = \alpha_i + \gamma_j$.

Here, μ_{ij} is the estimated number of counts at site i, time j. The parameters α_i, β and γ_j are refererred to as coefficients in the additive representation. By exponentiating both sides of the above equations, alternative representations can be written down. Explicitly, one can show that

- Model 2: $\mu_{ij} = a_i b^{(j-1)} = b\mu_{ij-1}$, where $a_i = e^{\alpha_i}$ and $b = e^{\beta}$.
- Model 3: $\mu_{ij} = a_i c_j$, where $a_i = e^{\alpha_i}$, $c_1 = 1$ and $c_j = e^{\gamma_j}$ for $j > 1$.

The parameters a_i, b and c_j are referred to as coefficients in the *multiplicative form*.

Trend and deviation (Model 3 only)

The equation for Model 3

$\ln \mu_{ij} = \alpha_i + \gamma_j$,

can also be written as an overall slope resulting from a linear regression of the μ_{ij} over time, plus site- and time effects that record deviations from this overall slope. In such a reparametrisation the previous equation can be written as

$\ln \mu_{ij} = \alpha_i^* + \beta^* d_j + \gamma_j^*$,

where d_j equals j minus the mean over all j (i.e. if $j = 1, 2, \ldots, J$ then $d_j = j - (J+1)/2$). It is not hard to show that

- The α_i^* are the mean $\ln \mu_{ij}$ per site
- The γ_j^* must sum to zero.

The coefficients α_i^* and γ_j^* are obtained by setting `representation="deviations"`. If `representation="trend"`, the overall trend parameters β^* and α^* from the overall slope defined by $\alpha^* + \beta^* d_j$ is returned.

Finally, note that both the overall slope and the deviations can be written in multiplicative form as well.

See Also

Other analyses: `gof`, `index`, `now_what`, `overall`, `overdispersion`, `plot.trim.index`, `plot.trim.overall`, `results`, `serial_correlation`, `summary.trim`, `totals`, `trim`, `vcov.trim`, `wald`

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2, overdisp=TRUE)
coefficients(z)
```

count_summary *Compute a summary of counts*

Description

Summarize counts over a trim input dataset. Sites without counts are removed before any counting takes place (since these will not be used when calling `trim`). For the remaining records, the total number of zero-counts, positive counts, total number of observed counts and the total number of missings are reported.

Usage

```
count_summary(x, count.id = "count", site.id = "site", time.id = "time",
  eps = 1e-08)
```

Arguments

`x`	A `data.frame` with annual counts per site.
`count.id`	`[character|numeric]` index of the column containing the counts
`site.id`	`[character|numeric]` index of the column containing the site id's
`time.id`	`[character|numeric]` index of the column containing the time codes
`eps`	`[numeric]` Numbers smaller then `eps` are treated a zero.

Value

A `list` of class `count.summary` containing individual names.

Examples

```
data(skylark)
count_summary(skylark)

s <- count_summary(skylark)
s$zero_counts # obtain number of zero counts
```

gof *Extract TRIM goodness-of-fit information.*

Description

`trim` computes three goodness-of-fit measures:

- Chi-squared
- Likelihood ratio
- Akaike Information content

Usage

```
gof(x)

## S3 method for class 'trim'
gof(x)
```

Arguments

x	an object of class trim (as returned by trim)

Value

a list of type "trim.gof", containing elements chi2, LR and AIC, for Chi-squared, Likelihoof Ratio and Akaike informatiuon content, respectively.

See Also

Other analyses: coef.trim, index, now_what, overall, overdispersion, plot.trim.index, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2)
# prettyprint GOF information
gof(z)

# get individual elements, e.g. p-value
L <- gof(z)
LR_p <- L$LR$p # get p-value for likelihood ratio
```

---

index *Extract time-indices from trim output*

Description

Extract time-indices from trim output

Usage

```
index(x, which = c("imputed", "fitted", "both"), covars = FALSE, base = 1)
```

Arguments

x	an object of class trim
which	[character] Selector to distinguish between time indices based on the imputed data (default), the fitted model, or both.
covars	[logical] Switch to compute indices for covariate categories as well.
base	[integer|numeric] Base time point, for which the index is 1. If the data contains J time points, the base time point can be given in the interval 1...J, or, if the time points are proper years, say year1...yearn, the base year can be given. So, if the data range 2000...2016, base=2 and base=2001 are equivalent.

Value

A data frame containing indices and their uncertainty expressed as standard error. Depending on the chosen output, columns fitted and se_fit, and/or imputed and se_imp are present. If covars is TRUE, additional indices are computed for the individual covariate categories. In this case additional columns covariate and category are present. The overall indices are marked as covariate 'Overall' and category 0.

See Also

Other analyses: coef.trim, gof, now_what, overall, overdispersion, plot.trim.index, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2)
index(z)
# mimic classic TRIM:
index(z, "both")
# Extract standard errors for the imputed data
SE <- index(z,"imputed")$se_mod
# Include covariates
z <- trim(count ~ site + time + Habitat, data=skylark, model=2)
ind <- index(z, covars=TRUE)
plot(ind)
```

---

now_what *Give advice on further refinement of TRIM models*

Description

Give advice on further refinement of TRIM models

Usage

```
now_what(z)
```

Arguments

z	an object of class trim.

See Also

trim

Other analyses: coef.trim, gof, index, overall, overdispersion, plot.trim.index, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2, overdisp=TRUE)
now_what(z)
```

---

overall *Compute overall slope*

Description

The overal slope represents the total growth over the piecewise linear model.

Usage

```
overall(x, which = c("imputed", "fitted"), changepoints = numeric(0))
```

Arguments

x	an object of class trim.
which	[character] Choose between "imputed" or "fitted" counts.
changepoints	[numeric] Change points for which to compute the overall slope, or "model", in which case the changepoints from the model are used (if any)

Value

a list of class trim.overall containing, a.o., overall slope coefficients (slope), augmented wih p-values and an interpretation).

Details

The overall slope represents the mean growth or decline over a period of time. This can be determined over the whole time period for which the model is fitted (this is the default) or may be computed over time slices that can be defined with the cp parameter. The values for changepoints do not depend on changepoints that were used when specifying the trim model (See also the example below).

See Also

Other analyses: coef.trim, gof, index, now_what, overdispersion, plot.trim.index, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
# obtain the overall slope accross all change points.
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2)
overall(z)
plot(overall(z))

# Overall is a list, you can get information out if it using the $ syntax,
# for example
L <- overall(z)
L$slope

# Obtain the slope from changepoint to changepoint
z <- trim(count ~ site + time, data=skylark, model=2,changepoints=c(1,4,6))
# slope from time point 1 to 5
overall(z,changepoints=c(1,5,7))
```

---

overdispersion *Extract overdispersion from trim object*

Description

Extract overdispersion from trim object

Usage

```
overdispersion(x)
```

Arguments

x	An object of class trim

Value

The overdispersion value if computed, otherwise NULL.

See Also

Other analyses: coef.trim, gof, index, now_what, overall, plot.trim.index, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

plot.trim.index *Plot time-indices from trim output*

Description

Plot time-indices from trim output

Usage

```
## S3 method for class 'trim.index'
plot(x, covar = "auto", ...)
```

Arguments

x	an object of class trim.index, as resulting from e.g. a call to index.
covar	[character] the name of a covariate to include in the plot. If set to "auto" (the default), the first (or only) covariate is used. If set to "none" plotting of covariates is suppressed and only the overall index is shown.
...	Further options passed to plot

See Also

Other analyses: coef.trim, gof, index, now_what, overall, overdispersion, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
z <- trim(count ~ site + time + Habitat, data=skylark, model=2)
idx <- index(z, covars=TRUE)
plot(idx, covar="Habitat", main="Skylark")
```

plot.trim.overall *Plot overall slope*

Description

Creates a plot of the overall slope, its 95% confidence band, the total population per time and their 95% confidence intervals.

Usage

```
## S3 method for class 'trim.overall'
plot(x, imputed = TRUE, ...)
```

Arguments

x	An object of class trim.overall (returned by overall)
imputed	[logical] Toggle to show imputed counts
...	Further options passed to plot

See Also

Other analyses: coef.trim, gof, index, now_what, overall, overdispersion, plot.trim.index, results, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
m <- trim(count ~ site + time, data=skylark, model=2)
plot(overall(m))
```

---

read_tcf *Read a TRIM command file*

Description

Read TRIM Command Files, compatible with the Windows TRIM programme.

Usage

```
read_tcf(file, encoding = getOption("encoding"), simplify = TRUE)
```

Arguments

file	Location of TRIM command file.
encoding	The encoding in which the file is stored.
simplify	Return a single trimcommand object if only one model is specified in the TRIM command file.

Value

A trimcommand object, or in the case of multiple models in a single TRIM command file, a list of trimcommand objects. In the latter case, a useful summary can be printed with summary.trimbatch.

TRIM Command file format

TRIM command files are text files that specify a TRIM job, where a job consists of one or more models to be computed on a single data input file. TRIM command files are commonly stored with the extension `.tcf`, but this is not a strict requirement.

A TRIM command file consists of two parts. The first part describes the data file to be read, the second part describes the model(s) to be run. A TRIM command file can only contain a single data specification part, but multiple models may be specified.

Each command starts on a new line with a keyword, followed by at least one space and at least one option value, where multiple option values are separated by spaces. All commands must be written on a single line, except the `LABELS` command (to set labels for covariates). The latter command starts with `LABELS` on a single line, followed by a newline, followed by a new label on each following line. The keyword `END` (at the beginning of a line) signals the end of the labels command.

The keyword `RUN` (at the beginning of a single line) ends the specification of a single model. After this a new model can be specified. Parameters not specified in the current model will be copied from the previous one.

TRIM commands

The commands are identical to those in the original TRIM software. Commands that represent a simple toggle (on/off, present/absent) are translated to a `logical` upon reading. Below we give commands in upper case, but the commands are parsed case insensitively.

Data	
`FILE`	data filename and path.
`TITLE`	A title (appears in output when exported).
`NTIMES`	[positive integer] Number of time points in data file.
`NCOVARS`	[nonnegative integer] Number of covariates in data file.
`LABELS`	Covariate labels (multiline command).
`END`	Signals end of `LABELS` command.
`MISSING`	missing value indicator.
`WEIGHT`	[`present`, `absent`] Indicates whether weights are present in the data file [translated to `logical`].
Model	
`COMMENT`	A comment for the current model.
`WEIGHTING`	[`on`,`off`] Switch use of weights for current model [translated to `logical`].
`SERIALCOR`	[`on`,`off`] Switch use of serial correlation for current model [translated to `logical`].
`OVERDISP`	[`on`,`off`] Switch use of overdispersion for current model [translated to `logical`].
`BASETIME`	[integer] Index of base time-point.
`MODEL`	[`1`, `2`, `3`] Choose the current model
`COVARIATES`	[integers] indices of covariates to use (1st covariate has index 1)
`CHANGEPOINTS`	[integers] indices of changepoints
`STEPWISE`	[`on`,`off`] Switch stepwise selection of changepoints [translated to `logical`].
`AUTODELETE`	[`on`, `off`] Delete changepoints when the corresponding time segment has to litte observations.
`OVERALLCHANGEPOINTS`	[integers] indices of overall changepoints
`RUN`	Signals end of current model specification.
Output	
`IMPCOVOUT`	[`on`, `off`] Switch to save variance-covariance matrix
`COVIN`	[`on`, `off`] Switch to read variance-covariance matrix

Encoding issues

To read files containing non-ASCII characters encoded in a format that is not native to your system, specifiy the `encoding` option. This causes R to re-encode to native encoding upon reading. Input encodings supported for your system can be listed by calling `iconvlist()`. For more information on Encoding in R, see `Encoding`.

Note on filenames

If the `file` is specified using backslashes to separate directories (Windows style), this will be converted to a filename using forward slashes (POSIX style, as used by R).

See Also

Working with TRIM command files and TRIM data files.

Other modelspec: `check_observations`, `read_tdf`, `set_trim_verbose`, `trimcommand`, `trim`

---

read_tdf *Read TRIM data files*

Description

Read data files intended for the original TRIM programme.

Usage

```
read_tdf(x, ...)

## S3 method for class 'character'
read_tdf(x, missing = -1, weight = FALSE, ncovars = 0,
  labels = character(0), ...)

## S3 method for class 'trimcommand'
read_tdf(x, ...)
```

Arguments

`x`	a filename or a `trimcommand` object
`...`	(unused)
`missing`	`[integer]` missing value indicator. Missing values are translated to `NA`.
`weight`	`[logical]` indicate presence of a weight column
`ncovars`	`[logical]` The number of covariates in the file
`labels`	`[character]` (optional) specify labels for the covariates. Defaults to `cov<i>` (`i=1,2,...,ncovars`) if none are specified.

Value

A data.frame.

The TRIM data file format

TRIM input data is stored in a ASCII encoded file where headerless columns are separated by one or more spaces. Below are the columns as read_tdf expects them.

Variable	status	R type
site	requiered	integer
time	required	integer
count	required	numeric
weight	optional	numeric
<covariate1>	optional	integer
...		
<covariateN>	optional	integer

See Also

Other modelspec: check_observations, read_tcf, set_trim_verbose, trimcommand, trim

---

results *collect observed, modelled, and imputed counts from TRIM output*

Description

collect observed, modelled, and imputed counts from TRIM output

Usage

```
results(z)
```

Arguments

z	TRIM output structure (i.e., output of a call to trim)

Value

A data.frame, one row per site-time combination, with columns for site, time, observed counts, modelled counts and imputed counts. Missing observations are marked as NA.

See Also

Other analyses: coef.trim, gof, index, now_what, overall, overdispersion, plot.trim.index, plot.trim.overall, serial_correlation, summary.trim, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2);
out <- results(z)
```

`serial_correlation` *Extract serial correlation from TRIM object*

Description

Extract serial correlation from TRIM object

Usage

```
serial_correlation(x)
```

Arguments

x	An object of class `trim`

Value

The serial correlation coefficient if computed, otherwise `NULL`.

See Also

Other analyses: `coef.trim`, `gof`, `index`, `now_what`, `overall`, `overdispersion`, `plot.trim.index`, `plot.trim.overall`, `results`, `summary.trim`, `totals`, `trim`, `vcov.trim`, `wald`

`set_trim_verbose` *Set verbosity of trim model functions*

Description

Control how much output `trim` writes to the screen while fitting the model. By default, `trim` only returns the output and does not write any progress to the screen. After calling `set_trim_verbose(TRUE)`, `trim` will write information about running iterations and convergence to the screen during optmization.

Usage

```
set_trim_verbose(verbose = FALSE)
```

Arguments

verbose	`[logical]` toggle verbosity. `TRUE` means: be verbose, `FALSE` means be quiet (this is the default).

See Also

Other modelspec: check_observations, read_tcf, read_tdf, trimcommand, trim

---

skylark *Skylark population data*

Description

The Skylark dataset that was included with the original TRIM software.

The dataset can be loaded in two forms. The skylark dataset is exactly equal to the data set in the original TRIM software:

Column	Type	Description
site	integer	Site number
time	integer	Time point coded as integer sequence
count	numeric	Counted skylarks
Habitat	integer	Habitat type (1, 2)
Deposition	integer	Deposition type (1, 2, 3, 4)

The current implementation is more flexible and allows time points to be coded as years and covariates as factors. The skylark2 data set looks as follows.

Column	Type	Description
site	factor	Site number
year	integer	Time point coded as year
count	integer	Counted skylarks
Habitat	factor	Habithat type (dunes, heath)
Deposition	integer	Deposition type (1, 2, 3, 4)
Weight	numeric	Site weight

Usage

```
data(skylark); data(skylark2)
```

Format

.RData

---

summary.trim *Summary information for a TRIM job*

Description

Print a summary of a trim object.

Usage

```
## S3 method for class 'trim'
summary(object, ...)
```

Arguments

object	an object of class trim.
...	Currently unused

Value

A list of class trim.summary containing the call that created the object, the model code, the coefficients (in additive and multiplicative form) , the goodness of fit parameters,the overdispersion and the serial correlation parameters (if computed).

See Also

trim

Other analyses: coef.trim, gof, index, now_what, overall, overdispersion, plot.trim.index, plot.trim.overall, results, serial_correlation, totals, trim, vcov.trim, wald

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2, overdisp=TRUE)

summary(z)
```

---

totals *Extract time-totals from TRIM output*

Description

Extract time-totals from TRIM output

Usage

```
totals(x, which = c("imputed", "fitted", "both"))
```

Arguments

x	TRIM output structure (i.e., output of a call to trim)
which	select what totals to compute (see Details section).

Value

A `data.frame` with subclass `trim.totals` (for pretty-printing). The columns are `time`, `fitted` and `se_fit` (for standard error), and/or `imputed` and `se_imp`, depending on the selection.

Details

The idea of `TRIM` is to impute those site-time combinations where no counts are available. Time-totals (i.e. summed over sites) can be obtained for two cases:

- `"imputed"`: Time totals are computed after replacing missing values with values predicted by the model.
- `"fitted"`: Time totals are computed after replacing both missing values and observed values with values predicted by the model.

See Also

Other analyses: `coef.trim`, `gof`, `index`, `now_what`, `overall`, `overdispersion`, `plot.trim.index`, `plot.trim.overall`, `results`, `serial_correlation`, `summary.trim`, `trim`, `vcov.trim`, `wald`

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2, changepoints=c(3,5))
totals(z)

totals(z, "both") # mimics classic TRIM
```

---

trim *Estimate TRIM model parameters*

Description

Compute TRIM model parameters.

Usage

```
trim(x, ...)

## S3 method for class 'trimcommand'
trim(x, ...)

## S3 method for class 'data.frame'
trim(x, count.id = "count", site.id = "site",
  time.id = "time", covars = character(0), model = 2,
  weights = numeric(0), serialcor = FALSE, overdisp = FALSE,
  changepoints = integer(0), stepwise = FALSE, autodelete = FALSE, ...)
```

```
## S3 method for class 'formula'
trim(x, data, model = 2, weights = numeric(0),
  serialcor = FALSE, overdisp = FALSE, changepoints = integer(0),
  stepwise = FALSE, autodelete = FALSE, ...)
```

Arguments

x	a trimcommand, a data.frame, or a formula If x is a formula, the dependent variable (left-hand-side) is treated as the 'counts' variable. The first and second independent variable are treated as the 'site' and 'time' variable, **in that specific order**. All other variables are treated as covariates.
...	Currently unused
count.id	[character] name of the column holding species counts
site.id	[character] name of the column holding the site id
time.id	[character] name of the column holding the time of counting
covars	[character] name(s) of column(s) holding covariates
model	[numeric] TRIM model type 1, 2, or 3.
weights	[numeric] Optional vector of site weights. The length of weights must be equal to the number of rows in the data.
serialcor	[logical] Take serial correlation into account (See 'Estimation details')
overdisp	[logical] Take overdispersion into account (See 'Estimation options').
changepoints	[numeric] Indices for changepoints ('Models').
stepwise	[logical] Perform stepwise refinement of changepoints.
autodelete	[logical] Auto-delete changepoints when number of observations is too small. (See 'Demands on data').
data	[data.frame] Data containing at least counts, sites, and times

Models

The purpose of trim is to estimate population totals over time, based on a set of counts f_{ij} at sites $i = 1, 2, \ldots, I$ and times $j = 1, 2, \ldots, J$. If no count data is available at site and time (i, j), a value μ_{ij} will be imputed.

In **Model 1**, the imputed values are modeled as

$$\ln \mu_{ij} = \alpha_i,$$

where α_i is the site effect. This model implies that the counts vary accross sites, not over time. The model-based time totals are equal to each time point and the model-based indices are all equal to one.

In **Model 2**, the imputed values are modeled as

$$\ln \mu_{ij} = \alpha_i + \beta \times (j - 1).$$

Here, α_i is the log-count of site i averaged over time and β is the mean growth factor that is shared by all sites over all of time. The assumption of a constant growth rate may be relaxed by passing a number of changepoints that indicate at what times the growth rate is allowed to change. Using

a `wald` test one can investigate whether the changes in slope at the changepoints are significant. Setting `stepwise=TRUE` makes `trim` automatically remove changepoints where the slope does not change significantly.

In **Model 3**, the imputed values are modeled as

$\ln \mu_{ij} = \alpha_i + \gamma_j,$

where γ_j is the deviatiation of log-counts at time j, averaged over all sites. To make this model identifiable, the value of $\gamma_1 = 0$ by definition. Model 3 can be shown to be equivalent to Model 2 with a changepoint at every time point. Using a `wald` test, one can estimate whether the collection of deviations γ_i make the model differ significantly from an overall linear trend (Model 2 without changepoints).

The parameters α_i, β and γ_j are referred to as the *additive representation* of the coefficients. Once computed, they can be represented and extracted in several representations, using the `coefficients` function. (See also the examples below).

Other model parameters can be extracted using functions such as `gof` (for goodness of fit), `summary` or `totals`. Refer to the 'See also' section for an overview.

Using covariates

In the basic case of Models 2 and 3, the growth parameter β does not vary accross sites. If auxiliary information is available (for instance a classification of the type of soil or vegetation), the effect of these variables on the per-site growth rate can be taken into account.

For **Model 2 with covariates** the growth factor β is replaced with a factor

$\beta_0 + \sum_{k=1}^{K} z_{ijk}\beta_k.$

Here, β_0 is referred to as the *baseline* and z_{ijk} is a dummy variable that combines dummy variables for all covariates. Since a covariate with L classes is modeled by $L - 1$ dummy variables, the value of K is equal to the sum of the numbers of categories for all covariates minus the number of covariates. Observe that this model allows for a covariate to change over time at a certain sites. It is therefore possible to include situations for example where a site turns from farmland to rural area. The `coefficients` function will report every individual value of β. With a `wald` test, the significance of contributions of covariates can be tested.

For **Model 3 with covariates** the parameter γ_j is replaced by

$\gamma_{j0} + \sum_{k=1}^{K} z_{ijk}\gamma_{jk}.$

Again, the γ_{j0} are referred to as baseline parameters and the γ_{jk} record mean differences in log-counts within a set of sites with equal values for the covariates. All coefficients can be extracted with `coefficients` and the significance of covariates can be investigated with the `wald` test.

Estimation options

In the simplest case, the counts at different times and sites are considered independently Poisson distributed. The (often too strict) assumption that counts are independent over time may be dropped, so correlation between time points at a certain site can be taken into account. The assumption of being Poisson distributed can be relaxed as well. In general, the variance-covariance structure of counts f_{ij} at site i for time j is modeled as

- $\text{var}(f_{ij}) = \sigma^2 \mu_{ij}$

- $\text{cor}(f_{ij}, f_{i,j+1}) = \rho$,

where σ is called the *overdispersion*, μ_{ij} is the estimated count for site i, time j and ρ is called the *serial correlation*.

If $\sigma = 1$, a pure Poisson distribution is assumed to model the counts. Setting `overdispersion = TRUE` makes `trim` relax this condition. Setting `serialcor=TRUE` allows `trim` to assume a non-zero correlation between adjacent time points, thus relaxing the assumption of independence over time.

Demands on data

The data set must contain sufficient counts to be able to estimate the model. In particular

- For model 2 without covariates there must be at least one observation for each time segment defined by the change points.
- For model 2 with covariates there must be at least one observation for every value of each covariate, at each time segment defined by the change points.
- For model 3 without covariates there must be at least one observation for each time point.
- For model 3 with covariates there must be at least one observation for every value of each covariate, at each time point.

The function `check_observations` identifies cases where too few observations are present to compute a model. Setting the option `autodelete=TRUE` (Model 2 only) makes `trim` remove changepoints such that at each time piece sufficient counts are available to estimate the model.

See Also

rtrim for TRIM users, TRIM by example

Other analyses: `coef.trim`, `gof`, `index`, `now_what`, `overall`, `overdispersion`, `plot.trim.index`, `plot.trim.overall`, `results`, `serial_correlation`, `summary.trim`, `totals`, `vcov.trim`, `wald`

Other modelspec: `check_observations`, `read_tcf`, `read_tdf`, `set_trim_verbose`, `trimcommand`

Examples

```
data(skylark)
m <- trim(count ~ site + time, data=skylark, model=2)
summary(m)
coefficients(m)

# An example using weights
# set up some random weights (one for each site)
w <- runif(55, 0.1, 0.9)
# match weights to sites
weights <- w[skylark$site]
# run model
m <- trim(count ~ site + time, data=skylark, model=3, weights=weights)


# An example using change points, a covariate, and overdispersion
# 1 is added as cp automatically
cp <- c(2,6)
```

```
m <- trim(count ~ site + time + Habitat, data=skylark, model=2, changepoints=cp, overdisp=TRUE)
coefficients(m)
# check significance of changes in slope
wald(m)
plot(overall(m))
```

`trimcommand` *Create a trimcommand object*

Description

Create a trimcommand object

Usage

```
trimcommand(...)
```

Arguments

`...`	Options in the form of `key=value`. See below for all options.

Description

A `trimcommand` object stores a single TRIM model, including the specification of the data file. Normally, such an object is defined by reading a legacy TRIM command file.

Options

- `file` [character] name of file containing training data.
- `title` [character] A string to be printed in the output file.
- `ntimes` [character] Number of time points.
- `ncovars` [character] Number of covariates.
- `labels` [character] Covariate label.
- `missing` [integer] Missing value indicator.
- `weight` [logical] Whether a weight column is present in the `file`.
- `comment` [character] A string to be printed in the output file.
- `weighting` [logical] Whether weights are to be used in the model.
- `serialcor` [logical] Whether serial correlation is assumed in the model.
- `overdist` [logical] Whether overdispersion is taken into account by the model.
- `basetime` [integer] Position of the base time point (must be positive).
- `model` [integer] What model to use (1, 2 or 3).
- `covariates` [integer] Number of covariates to include.
- `changepoints` [integer] Positions of the change points to include.

- `stepwise [logical]` Whether stepwise selection of the changepoints is to be used.
- `autodelete [logical]` Whether to autodelete change points when number of observations is to low in a time segment.
- `outputfiles [character]` Type of outputfile to generate ('F' and/or 'S')
- `overallchangepoints [integer]` Positions of the overall change points.
- `impcovout [logical]` Whether the covariance matrix of the imputed counts is saved.
- `covin [logical]` Whether the covariance matrix is read in.

See Also

Working with TRIM command files and TRIM data files.

Other modelspec: `check_observations`, `read_tcf`, `read_tdf`, `set_trim_verbose`, `trim`

`vcov.trim` *Extract variance-covariance matrix from TRIM output*

Description

Extract variance-covariance matrix from TRIM output

Usage

```
## S3 method for class 'trim'
vcov(object, which = c("imputed", "model"), ...)
```

Arguments

`object`	TRIM output structure (i.e., output of a call to `trim`)
`which`	`[character]` Selector to distinguish between variance-covariance based on the imputed data (default), or the modelled data.
`...`	Arguments to pass to or from other methods (currently unused)

Value

a JxJ matrix, where J is the number of time points.

See Also

Other analyses: `coef.trim`, `gof`, `index`, `now_what`, `overall`, `overdispersion`, `plot.trim.index`, `plot.trim.overall`, `results`, `serial_correlation`, `summary.trim`, `totals`, `trim`, `wald`

Examples

```
data(skylark)
z <- trim(count ~ site + time, data=skylark, model=2);
totals(z)
vcv1 <- vcov(z)       # Use imputed data
vcv2 <- vcov(z,"model") # Use modelled data
```

wald *Test significance of TRIM coefficients with the Wald test*

Description

Test significance of TRIM coefficients with the Wald test

Usage

```
wald(x)

## S3 method for class 'trim'
wald(x)
```

Arguments

x	TRIM output structure (i.e., output of a call to trim)

Value

A model-dependent list of Wald statistics

See Also

Other analyses: coef.trim, gof, index, now_what, overall, overdispersion, plot.trim.index, plot.trim.overall, results, serial_correlation, summary.trim, totals, trim, vcov.trim

Examples

```
data(skylark)
z2 <- trim(count ~ site + time, data=skylark, model=2)
# print info on significance of slope parameters
print(z2)
z3 <- trim(count ~ site + time, data=skylark, model=3)
# print info on significance of deviations from linear trend
wald(z3)
```

Index