

# Package ‘rtrim’

October 14, 2022

**Version** 2.1.1

**Date** 2020-04-19

**Title** Trends and Indices for Monitoring Data

**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%2d%2dtrim%2d%2d> for more information about TRIM.

**URL** <https://github.com/markvanderloo/rtrim>

**BugReports** <https://github.com/markvanderloo/rtrim/issues>

**LazyLoad** yes

**LazyData** no

**License** EUPL

**Type** Package

**Imports** methods, utils, stats, graphics, grDevices

**Suggests** testthat, knitr, rmarkdown, R.rsp

**RoxygenNote** 7.1.0

**Encoding** UTF-8

**VignetteBuilder** knitr, R.rsp

**NeedsCompilation** no

**Author** Patrick Bogaart [aut, cre],  
Mark van der Loo [aut],  
Jeroen Pannekoek [aut]

**Maintainer** Patrick Bogaart <rtrim@cbs.nl>

**Repository** CRAN

**Date/Publication** 2020-04-21 12:20:02 UTC

**R topics documented:**

rtrim-package	2
check_observations	3
ci_multipliers	5
coef.trim	5
confint.trim	7
count_summary	7
gof	8
heatmap	9
index	10
now_what	12
overall	13
overdispersion	14
oystercatcher	15
plot.trim.index	15
plot.trim.overall	17
plot.trim.totals	17
read_tcf	19
read_tdf	21
results	22
serial_correlation	23
set_trim_verbose	23
skylark	24
summary.trim	24
totals	25
trim	26
trimcommand	31
vcov.trim	32
wald	33
<b>Index</b>	<b>34</b>

---

 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

Several vignettes have been written to document the 'rtrim' package.

For everybody:

- [rtrim by example](#)
- [rtrim 2 extensions](#)

For users of the original Windows TRIM software:

- [rtrim for TRIM users](#)

For users who would like to have more insight what is going on under the hood:

- [Models and statistical methods in rtrim](#) (PDF),
- [rtrim confidence intervals](#)
- [Taming overdispersion](#).

Enjoy! The rtrim team of Statistics Netherlands

---

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,  
  count_col = "count",  
  year_col = "year",  
  month_col = NULL,  
  covars = character(0),  
  changepoints = numeric(0),  
  eps = 1e-08,  
  ...  
)  
  
## S3 method for class 'trimcommand'  
check_observations(x, ...)  
  
## S3 method for class 'character'  
check_observations(x, ...)
```

**Arguments**

x	A <code>trimcommand</code> object, a <code>data.frame</code> , or the location of a TRIM command file.
...	Parameters passed to other methods.
model	[numeric] Model 1, 2 or 3?
count_col	[character numeric] column index of the counts in x
year_col	[character numeric] column index of years or time points in x
month_col	[character numeric] optional column index of months in x
covars	[character numeric] column index of covariates in x
changepoints	[numeric] Changepoints (model 2 only)
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 insufficient 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 insufficient counts are encountered. Each element is a two-column `data.frame`. The first column 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()`

---

ci_multipliers	<i>Compute Std.err ==&gt; conf.int multipliers.</i>
----------------	---

---

**Description**

Compute Std.err ==> conf.int multipliers.

**Usage**

```
ci_multipliers(lambda, sig2 = NULL, level = 0.95)
```

**Arguments**

lambda	mean
sig2	overdispersion parameter
level	the confidence level required

**Value**

matrix with multipliers. col1=lo; col2=hi

---

coef.trim	<i>Extract TRIM model coefficients.</i>
-----------	---

---

**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,  $\mu_{ij}$  is the estimated number of counts at site  $i$ , time  $j$ . The parameters  $\alpha_i$ ,  $\beta$  and  $\gamma_j$  are referred 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  $\mu_{ij}$  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  $j = 1, 2, \dots, J$  then  $d_j = j - (J + 1)/2$ ). It is not hard to show that

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

The coefficients  $\alpha_i^*$  and  $\gamma_j^*$  are obtained by setting `representation="deviations"`. If `representation="trend"`, the overall trend parameters  $\beta^*$  and  $\alpha^*$  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: `confint.trim()`, `gof()`, `index()`, `now_what()`, `overall()`, `overdispersion()`, `plot.trim.index()`, `plot.trim.overall()`, `results()`, `serial_correlation()`, `summary.trim()`, `totals()`, `trim()`, `vcov.trim()`, `wald()`

## Examples

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

---

confint.trim	<i>Compute time-totals confidence interval</i>
--------------	--

---

**Description**

Computes confidence intervals for the time-totals of a TRIM model. Both imputed and fitted time-totals are supported, and the confidence level can be specified.

**Usage**

```
## S3 method for class 'trim'
confint(object, parm = c("imputed", "fitted"), level = 0.95, ...)
```

**Arguments**

object	a TRIM output object
parm	parameter specification: imputed or fitted time-totals.
level	the confidence level required.
...	not used [included for R compatibility reasons]

**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\(\)](#), [wald\(\)](#)

**Examples**

```
data(skylark2)
z <- trim(count ~ site + year, data=skylark2, model=3)
CI <- confint(z)
```

---

count_summary	<i>Compute a summary of counts</i>
---------------	------------------------------------

---

**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_col = "count",
  site_col = "site",
  year_col = "year",
  eps = 1e-08
)
```

**Arguments**

x	A data.frame with annual counts per site.
count_col	[character numeric] index of the column containing the counts
site_col	[character numeric] index of the column containing the site ID's
year_col	[character numeric] index of the column containing the year
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, Likelihood Ratio and Akaike information content, respectively.

**See Also**

Other analyses: `coef.trim()`, `confint.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
```

---

heatmap

---

*Plot a heatmap representation of observed and/or imputed counts.*


---

**Description**

This function organizes the observed and/or imputed counts into a matrix where rows represent sites and columns represent time points. A bitmap image is constructed in which each pixel corresponds to an element of this matrix. Each pixel is colored according the corresponding count status, and the type of heatmap plot requested ('data', 'imputed' or 'fitted').

**Usage**

```
heatmap(
  z,
  what = c("data", "imputed", "fitted"),
  log = TRUE,
  xlab = "auto",
  ylab = "Site #",
  ...
)
```

**Arguments**

<code>z</code>	output of a call to <code>trim</code> .
<code>what</code>	the type of heatmap to be plotted: 'data' (default), 'imputed' or 'fitted'.
<code>log</code>	flag to indicate whether the count should be log-transformed first.
<code>xlab</code>	x-axis label. The default value "auto" will evaluate to either "Year" or "Time point"
<code>ylab</code>	y-axis label
<code>...</code>	other parameters to be passed to <code>plot</code>

**Details**

The 'imputed' heatmap uses the most elaborate color scheme: Site/time combinations that are observed are colored red, the higher the count, the darker the red. Site/time combinations that are imputed are colored blue, the higher the estimate, the darker the blue.

For the 'data' heatmap, missing site/time combinations are colored gray.

For the 'fitted' heatmap, all site/time combinations are colored blue.

By default, all counts are log-transformed prior to colorization, and observed counts of 0 are indicated as white pixels.

**See Also**

Other graphical post-processing: `plot.trim.index()`, `plot.trim.totals()`

**Examples**

```
data(skylark2)
z <- trim(count ~ site + year, data=skylark2, model=3)
heatmap(z,"imputed")
```

---

index

---

*Extract time-indices from TRIM output.*


---

**Description**

Indices are obtained by dividing the modelled or imputed time totals by a reference value. Most commonly, the time totals for the first time point are used as reference. As a result, the index value for this first time point will be 1.0, with a standard error of 0.0 by definition. Alternatively, a range of time points can be used as reference. In this case, the mean time totals for this range will be used as reference, and the standard errors will be larger than 0.0.

**Usage**

```

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

```

**Arguments**

x	an object of class <code>trim</code>
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] One or more base time point, used as as reference for the index. If just a single number is given, the time total of the correspondong time point will be uses as reference. If a range of numbers is given, the average of the corresponding time totals will be used as reference. The base time points 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.
level	[numeric] the confidence interval required. Must be in the range 0 to 1. A value of 0.95 results in 95% confidence intervals. The default value of NULL results in no confidence interval to be computed.

**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()`, `confint.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")$se_mod
# Include covariates
skylark$Habitat <- factor(s skylark$Habitat) # hack
z <- trim(count ~ site + time + Habitat, data=skylark, model=2)
ind <- index(z, covars=TRUE)
plot(ind)
# Use alternative base year
index(z, base=3)
# Use average of first 5 years as reference for indexing
index(z, base=1:5)
```

---

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()`, `confint.trim()`, `gof()`, `index()`, `overall()`, `overdispersion()`, `plot.trim.index()`, `plot.trim.overall()`, `results()`, `serial_correlation()`, `summary.trim()`, `totals()`, `trim()`, `vcov.trim()`, `wald()`

## Examples

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

---

overall	<i>Compute overall slope</i>
---------	------------------------------

---

### Description

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

### Usage

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

### Arguments

x	an object of class <code>trim</code> .
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)
bc	[logical] Flag to set backwards compatibility with TRIM with respect to trend interpretation. Defaults to FALSE.

### Value

a list of class `trim.overall` containing, a.o., overall slope coefficients (slope), augmented with 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).

Slopes are computed along with associated confidence intervals (CI) for 1% and 5% significance levels, and interpreted using the following table:

Trend meaning	Condition
Strong increase (more than 5% per year)	lower CI limit > 0.05
Moderate increase (less than 5% per year)	lower CI limit > 0
Moderate decrease (less than 5% per year)	upper CI limit < 0
Strong decrease (more than 5% per year)	upper CI limit < -0.05
Stable	-0.05 < lower < 0 < upper < 0.05
Uncertain	any other case

where trend strength takes precedence over significance, i.e., a *strong increase* ( $p < 0.05$ ) takes precedence over a *moderate increase* ( $p < 0.01$ ).

Note that the original TRIM erroneously assumed that the estimated overall trend magnitude is t-distributed, while in fact it is normally distributed, which is being used within rtrim. The option `bc=TRUE` can be set to force backward compatibility, for e.g. comparison purposes.

### See Also

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

---

oystercatcher	<i>Oystercatcher population data</i>
---------------	--------------------------------------

---

**Description**

A sample data set for demonstration of monthly counts.

The oystercatcher data set looks as follows.

Column	Type	Description
site	integer	Site number
year	integer	Year
month	integer	Month
count	integer	Counted oystercatchers

**Usage**

```
data(oystercatcher)
```

**Format**

```
.RData
```

---

plot.trim.index	<i>Plot time-indices from trim output.</i>
-----------------	--

---

**Description**

Uncertainty ranges expressed as standard errors are always plotted. Confidence intervals are plotted when they are present in the `trim.index` object, i.e. when requested for in the call to `index`.

**Usage**

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

```

    ylab = "Index",
    pct = FALSE,
    band = "se"
  )

```

### Arguments

x	an object of class <code>trim.index</code> , as resulting from e.g. a call to <code>index</code> .
...	additional <code>trim.index</code> objects, or parameters that will be passed on to <code>plot</code> .
names	optional character vector with names for the various series.
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.
xlab	a title for the x-axis. The default value is "auto" will be changed to "Time Point" if the time ID's start at 1, and to "Year" otherwise.
ylab	a title for the y-axis. The default value is "Index".
pct	Switch to show the index values as percent instead as fraction (i.e., for the base year it will be 100 instead of 1)
band	Defines if the uncertainty band will be plotted using standard errors ("se") or confidence intervals ("ci").

### See Also

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

Other graphical post-processing: `heatmap()`, `plot.trim.totals()`

### Examples

```

# Simple example
data(skylark2)
z <- trim(count ~ site + year, data=skylark2, model=3)
idx <- index(z)
plot(idx)

# Example with user-modified title, and different y-axis scaling
plot(idx, main="Skylark", pct=TRUE)

# Using covariates:
z <- trim(count ~ site + year + habitat, data=skylark2, model=3)
idx <- index(z, covars=TRUE)
plot(idx)

# Suppressing the plotting of covariate indices:
plot(idx, covar="none")

```



---

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\(\)](#), [confint.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))
```

---

plot.trim.totals      *Plot time-totals from trim output.*

---

### Description

This function plots a time series of one or more trim.totals objects, i.e. the output of totals. Both the time totals themselves, as the associated standard errors will be plotted, the former as a solid line with markers, the latter as a transparent band.

**Usage**

```
## S3 method for class 'trim.totals'
plot(
  x,
  ...,
  names = NULL,
  xlab = "auto",
  ylab = "Time totals",
  leg.pos = "topleft",
  band = "se"
)
```

**Arguments**

x	an object of class <code>trim.totals</code> , as resulting from e.g. a call to <code>totals</code> .
...	optional additional <code>trim.totals</code> objects.
names	optional character vector with names for the various series.
xlab	x-axis label. The default value of "auto" will be changed into "Year" or "Time Point", whichever is more appropriate.
ylab	y-axis label.
leg.pos	legend position, similar as in <a href="#">legend</a> .
band	Defines if the uncertainty band will be plotted using standard errors ("se") or confidence intervals ("ci").

**Details**

Additionally, the observed counts will be plotted (as a line) when this was asked for in the call to `totals`.

Multiple time-total data sets can be compared in a single plot

**See Also**

Other graphical post-processing: [heatmap\(\)](#), [plot.trim.index\(\)](#)

**Examples**

```
# Simple example
data(skylark2)
z <- trim(count ~ site + year, data=skylark2, model=3)
plot(totals(z))

# Extended example
z1 <- trim(count ~ site + year + habitat, data=skylark2, model=3)
z2 <- trim(count ~ site + year, data=skylark2, model=3)
t1 <- totals(z1, obs=TRUE)
t2 <- totals(z2, obs=TRUE)
plot(t1, t2, names=c("with covariates", "without covariates"), main="Skylark", leg.pos="bottom")
```

---

read_tcf	<i>Read a TRIM command file</i>
----------	---------------------------------

---

**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 too little 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, specify 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	<i>Read TRIM data files</i>
----------	-----------------------------

---

**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 <code>trimcommand</code> object
...	(unused)
missing	[integer] missing value indicator. Missing values are translated to <code>NA</code> .
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 <code>cov&lt;i&gt;</code> ( <code>i=1,2,...,ncovars</code> ) 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	required	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, year, month (optionally), observed counts, modelled counts and imputed counts. Missing observations are marked as NA.

**See Also**

Other analyses: [coef.trim\(\)](#), [confint.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()`, `confint.trim()`, `gof()`, `index()`, `now_what()`, `overall()`, `overdispersion()`, `plot.trim.index()`, `plot.trim.overall()`, `results()`, `summary.trim()`, `totals()`, `trim()`, `vcov.trim()`, `wald()`

---

set\_trim\_verbos      *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_verbos(TRUE)`, `trim` will write information about running iterations and convergence to the screen during optimization.

**Usage**

```
set_trim_verbos(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	<i>Skylark population data</i>
---------	--------------------------------

---

### 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	Habitat type (dunes, heath)
Deposition	integer	Deposition type (1, 2, 3, 4)
Weight	numeric	Site weight

### Usage

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

### Format

```
.RData
```

---

summary.trim	<i>Summary information for a TRIM job</i>
--------------	---

---

### 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()`, `confint.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)
```

---

<code>totals</code>	<i>Extract time-totals from TRIM output</i>
---------------------	---

---

**Description**

Extract time-totals from TRIM output

**Usage**

```
totals(x, which = c("imputed", "fitted", "both"), obs = FALSE, level = NULL)
```

**Arguments**

`x`                TRIM output structure (i.e., output of a call to `trim`)  
`which`            select what totals to compute (see `Details` section).  
`obs`                Flag to include total observations (or not).  
`level`            the confidence level required. If `NULL`, no confidence intervals are calculated.

**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()`, `confint.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

Given some count observations, estimate a TRIM model and use these to impute the data set if necessary.

## Usage

```
trim(object, ...)

## S3 method for class 'data.frame'
trim(
  object,
  count_col = "count",
  site_col = "site",
  year_col = "year",
  month_col = NULL,
  weights_col = NULL,
  covar_cols = NULL,
  model = 2,
  changepoints = ifelse(model == 2, 1L, integer(0)),
```

```

    overdisp = FALSE,
    serialcor = FALSE,
    autodelete = TRUE,
    stepwise = FALSE,
    covin = list(),
    ...
)

## S3 method for class 'formula'
trim(object, data = NULL, weights = NULL, ...)

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

```

### Arguments

object	Either a <code>data.frame</code> , a formula or a <code>trimcommand</code> . If object 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, <b>in that specific order</b> . All other variables are treated as covariates.
...	More parameters, see below in the details
count_col	[character] name of the column holding species counts
site_col	[character] name of the column holding the site id
year_col	[character] name of the column holding the time of counting
month_col	[character] optional name of the column holding the season of counting
weights_col	[numeric] Optional vector of site weights. The length of
covar_cols	[character] name(s) of column(s) holding covariates
model	[numeric] TRIM model type 1, 2, or 3.
changepoints	[numeric] Indices for changepoints ('Models').
overdisp	[logical] Take overdispersion into account (See 'Estimation options').
serialcor	[logical] Take serial correlation into account (See 'Estimation details')
autodelete	[logical] Auto-delete changepoints when number of observations is too small. (See 'Demands on data').
stepwise	[logical] Perform stepwise refinement of changepoints.
covin	a list of variance-covariance matrices; one per pseudo-site.
data	[data.frame] Data frame containing at least counts, sites, and times
weights	[character] name of the column in data which represents weights (optional)

### Details

All versions of `trim` support additional 'experts only' arguments:

`verbose` Logical switch to temporarily enable verbose output. (use `option(trim_verbose=TRUE)` for permanent verbosity.)

constrain\_overdisp Numerical value to control overdispersion.

- A value in the range 0..1 uses a Chi-squared outlier detection method.
- A value >1 uses Tukey's Fence.
- A value of 1.0 (which is the default) results in unconstrained overdispersion.

See vignette 'Taming overdispersion' for more information.

conv\_crit Convergence criterion. Used within the iterative emodel estimation algorithm. The default value is 1e-5. May be set to higher values in case models don't converge.

max\_iter Number of iterations. Default value is 200. May be set to higher values in case models don't converge.

remove Probability of removal of changepoints (default value: 0.2). Parameter used in stepwise refinement of models. See the vignette 'Models and statistical methods in rtrim'.

penter Probability of re-entering of changepoints (default value: 0.15). Similar use as remove.

## Models

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

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

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

where  $\alpha_i$  is the site effect. This model implies that the counts vary across 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,  $\alpha_i$  is the log-count of site  $i$  averaged over time and  $\beta$  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

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

where  $\beta_j$  is the deviation of log-counts at time  $j$ , averaged over all sites. To make this model identifiable, the value of  $\beta_1 = 0$  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  $\beta_i$  make the model differ significantly from an overall linear trend (Model 2 without changepoints).

The parameters  $\alpha_i$  and  $\gamma_j$  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 yearly and monthly counts

In many data sets will use only yearly count data, in which case the time  $j$  will reflect the year number. An extension of `trim` is to use monthly (or any other sub-yearly) count data, in combination with index computations on the yearly time scale.

In this case, counts are given as  $f_{i,j,m}$  with  $m = 1, 2, \dots, M$  the month number. As before,  $\mu_{i,j,m}$  will be imputed in case of missing counts.

The contribution of month factors to the model is always similar to the way year factors are used in Model 3, that is,

$\ln \mu_{i,j,m} = \alpha_i + \beta \times (j - 1) + \gamma_m$  for Model 2, and  $\ln \mu_{i,j,m} = \alpha_i + \beta_j + \gamma_m$  for Model 3.

For the same reason why  $\beta_1 = 0$  for Model 3,  $\gamma_1 = 0$  in case of monthly parameters.

### Using covariates

In the basic case of Models 2 and 3, the growth parameter  $\beta$  does not vary across 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  $\beta$  is replaced with a factor

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

Here,  $\beta_0$  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  $\beta$ . With a `wald` test, the significance of contributions of covariates can be tested.

For **Model 3 with covariates** the parameter  $\beta_j$  is replaced by

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

Again, the  $\beta_{j0}$  are referred to as baseline parameters and the  $\beta_{jk}$  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  $\sigma$  is called the *overdispersion*,  $\mu_{ij}$  is the estimated count for site  $i$ , time  $j$  and  $\rho$  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.
- For monthly data, there must be at least one observation for every month.

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 change-points such that at each time piece sufficient counts are available to estimate the model.

### See Also

[rtrim by example](#) for a gentle introduction, [rtrim for TRIM users](#) for users of the classic Delphi-based TRIM implementation, and [rtrim 2 extensions](#) for the major changes from `rtrim` v.1 to `rtrim` v.2

Other analyses: `coef.trim()`, `confint.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_verbos()`, `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
skylark$weights <- w[skylark$site]
# run model
m <- trim(count ~ site + time, data=skylark, weights="weights", model=3)

# 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	<i>Create a trimcommand object</i>
-------------	------------------------------------

---

## 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", "fitted"), ...)
```

### Arguments

<code>object</code>	TRIM output structure (i.e., output of a call to <code>trim</code> )
<code>which</code>	[character] Selector to distinguish between variance-covariance based on the imputed counts (default), or the fitted counts.
<code>...</code>	Arguments to pass to or from other methods (currently unused; included for consistency with <code>vcov</code> ).

### Value

a  $J \times J$  matrix, where  $J$  is the number of years (or time-points).

### See Also

Other analyses: [coef.trim\(\)](#), [confint.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=3);
totals(z)
vcv1 <- vcov(z)           # Use imputed data
vcv2 <- vcov(z,"fitted") # Use fitted 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()`, `confint.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

- \* **analyses**
  - coef.trim, 5
  - confint.trim, 7
  - gof, 8
  - index, 10
  - now\_what, 12
  - overall, 13
  - overdispersion, 14
  - plot.trim.index, 15
  - plot.trim.overall, 17
  - results, 22
  - serial\_correlation, 23
  - summary.trim, 24
  - totals, 25
  - trim, 26
  - vcov.trim, 32
  - wald, 33
- \* **graphical post-processing**
  - heatmap, 9
  - plot.trim.index, 15
  - plot.trim.totals, 17
- \* **modelspec**
  - check\_observations, 3
  - read\_tcf, 19
  - read\_tdf, 21
  - set\_trim\_verbose, 23
  - trim, 26
  - trimcommand, 31
- check\_observations, 3, 20, 22, 23, 30, 32
- ci\_multipliers, 5
- coef.trim, 5, 7, 9, 11, 12, 14–17, 22, 23, 25, 26, 30, 32, 33
- coefficients, 28, 29
- confint.trim, 6, 7, 9, 11, 12, 14–17, 22, 23, 25, 26, 30, 32, 33
- count\_summary, 7
- Encoding, 20
- gof, 6, 7, 8, 11, 12, 14–17, 22, 23, 25, 26, 28, 30, 32, 33
- heatmap, 9, 16, 18
- iconvlist, 20
- index, 6, 7, 9, 10, 12, 14–17, 22, 23, 25, 26, 30, 32, 33
- indices, 28
- Legend, 18
- NA, 21
- now\_what, 6, 7, 9, 11, 12, 14–17, 22, 23, 25, 26, 30, 32, 33
- overall, 6, 7, 9, 11, 12, 13, 15–17, 22, 23, 25, 26, 30, 32, 33
- overdispersion, 6, 7, 9, 11, 12, 14, 14, 16, 17, 22, 23, 25, 26, 30, 32, 33
- oystercatcher, 15
- plot, 10, 16, 17
- plot.trim.index, 6, 7, 9–12, 14, 15, 15, 17, 18, 22, 23, 25, 26, 30, 32, 33
- plot.trim.overall, 6, 7, 9, 11, 12, 14–16, 17, 22, 23, 25, 26, 30, 32, 33
- plot.trim.totals, 10, 16, 17
- read\_tcf, 4, 19, 22, 23, 30, 32
- read\_tdf, 4, 20, 21, 23, 30, 32
- results, 6, 7, 9, 11, 12, 14–17, 22, 23, 25, 26, 30, 32, 33
- rtrim-package, 2
- serial\_correlation, 6, 7, 9, 11, 12, 14–17, 22, 23, 25, 26, 30, 32, 33
- set\_trim\_verbose, 4, 20, 22, 23, 30, 32
- skylark, 24
- skylark2 (skylark), 24
- summary, 28

summary.trim, [6](#), [7](#), [9](#), [11](#), [12](#), [14–17](#), [22](#), [23](#),  
[24](#), [26](#), [30](#), [32](#), [33](#)

summary.trimbatch, [19](#)

time totals, [28](#)

totals, [6](#), [7](#), [9](#), [11](#), [12](#), [14–17](#), [22](#), [23](#), [25](#), [25](#),  
[28](#), [30](#), [32](#), [33](#)

trim, [4](#), [6–17](#), [20](#), [22–26](#), [26](#), [32](#), [33](#)

trimcommand, [4](#), [20–23](#), [27](#), [30](#), [31](#)

vcov, [32](#)

vcov.trim, [6](#), [7](#), [9](#), [11](#), [12](#), [14–17](#), [22](#), [23](#), [25](#),  
[26](#), [30](#), [32](#), [33](#)

wald, [6](#), [7](#), [9](#), [11](#), [12](#), [14–17](#), [22](#), [23](#), [25](#), [26](#),  
[28–30](#), [32](#), [33](#)