Package 'coat'

June 27, 2025

Title Conditional Method Agreement Trees (COAT)

Version 0.2.1

Date 2025-06-27

Description Agreement of continuously scaled measurements made by two techniques, devices or methods is usually
evaluated by the well-established Bland-
Altman analysis or plot. Conditional method agreement trees (COAT),
proposed by Karapetyan, Zeileis, Henriksen, and Hapfelmeier (2023) <doi:10.48550/arXiv.2306.04456>,
embed the Bland-
Altman analysis in the framework of recursive partitioning to explore heterogeneous method agreement in dependence of covariates. COAT can also be used to perform a Bland-
Altman test for differences
in method agreement.

License GPL-2 | GPL-3

Depends R (>= 3.5.0)

Imports partykit

Suggests disttree, MethComp

Additional_repositories https://R-Forge.R-project.org

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Alexander Hapfelmeier [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6765-6352>),
Siranush Karapetyan [aut] (ORCID:
<https://orcid.org/0000-0003-1831-9741>),
Achim Zeileis [aut] (ORCID: <https://orcid.org/0000-0003-0918-3766>)

Maintainer Alexander Hapfelmeier <Alexander.Hapfelmeier@mri.tum.de>

Repository CRAN

Date/Publication 2025-06-27 15:50:02 UTC

Contents

---

batest *Bland-Altman Test of Method Agreement*

Description

Function to perform a Bland-Altman test of differences in method agreement. Additional functions are given for printing and plotting.

Usage

```
batest(formula, data, subset, na.action, weights, ...)

## S3 method for class 'batest'
print(x, digits = 2, type = c("test", "model", "both"), ...)

## S3 method for class 'batest'
plot(x, ...)
```

Arguments

`formula`	symbolic description of the model used to perform the Bland-Altman test of type `y1 + y2 ~ x`. The left-hand side should specify a pair of measurements (`y1` and `y2`) to assess the agreement. The right-hand side should specify a factor with two levels indicating two independent groups or samples to be compared. Alternatively, multilevel factors or continuously scaled variables can be specified to perform a Bland-Altman test of association, followed by binary splitting into two subgroups.
`data`, `subset`, `na.action`	arguments controlling the formula processing via `model.frame`.
`weights`	optional numeric vector of weights (case/frequency weights, by default).
`...`	further control arguments, passed to `ctree_control`
`x`	an object as returned by `batest`.
`digits`	a numeric specifying the number of digits to display.
`type`	character string specifying whether `"test"` statistics (default), the `"model"` or `"both"` should be printed.

Value

Object of class `batest` with elements

`test`	result of the Bland-Altman test.
`model`	tree model used to perform the Bland-Altman test.

Methods (by generic)

- `print(batest)`: function to print the result of the Bland-Altman test.
- `plot(batest)`: function to plot the result of the Bland-Altman test.

Examples

```
## package and data (reshaped to wide format)
library("coat")
data("VitCap", package = "MethComp")
VitCap_wide <- reshape(VitCap, v.names = "y", timevar = "instrument",
                       idvar = c("item", "user"), drop = "meth", direction = "wide")

## two-sample BA-test
testresult <- batest(y.St + y.Exp ~ user, data = VitCap_wide)

## display
testresult
print(testresult, digits = 1, type = "both")
plot(testresult)
```

---

coat *Conditional Method Agreement Trees (COAT)*

Description

Tree models capturing the dependence of method agreement on covariates. The classic Bland-Altman analysis is used for modeling method agreement while the covariate dependency can be learned either nonparametrically via conditional inference trees (CTree) or using model-based recursive partitioning (MOB).

Usage

```
coat(
  formula,
  data,
  subset,
  na.action,
  weights,
  means = FALSE,
  type = c("ctree", "mob"),
```

```
  minsize = 10L,
  minbucket = minsize,
  minsplit = NULL,
  ...
)
```

Arguments

`formula`	symbolic description of the model of type `y1 + y2 ~ x1 + ... + xk`. The left-hand side should specify a pair of measurements (`y1` and `y2`) for the Bland-Altman analysis. The right-hand side can specify any number of potential split variables for the tree.
`data, subset, na.action`	arguments controlling the formula processing via `model.frame`.
`weights`	optional numeric vector of weights (case/frequency weights, by default).
`means`	logical. Should the intra-individual mean values of measurements be included as potential split variable?
`type`	character string specifying the type of tree to be fit. Either `"ctree"` (default) or `"mob"`.
`minsize, minbucket`	integer. The minimum number of observations in a subgroup. Only one of the two arguments should be used (see also below).
`minsplit`	integer. The minimum number of observations to consider splitting. Must be at least twice the minimal subgroup size (`minsplit` or `minbucket`). If set to `NULL` (the default) it is set to be at least 2.5 times the minimal subgroup size.
`...`	further control arguments, either passed to `ctree_control` or `mob_control`, respectively.

Details

Conditional method agreement trees (COAT) employ unbiased recursive partitioning in order to detect and model dependency on covariates in the classic Bland-Altman analysis. One of two recursive partitioning techniques can be used to find subgroups defined by splits in covariates to a pair of measurements, either nonparametric conditional inference trees (CTree) or parametric model-based trees (MOB). In both cases, each subgroup is associated with two parameter estimates: the mean of the measurement difference ("Bias") and the corresponding sample standard deviation ("SD") which can be used to construct the limits of agreement (i.e., the corresponding confidence intervals).

The minimum number of observations in a subgroup defaults to 10, so that the mean and variance of the measurement differences can be estimated reasonably for the Bland-Altman analysis. The default can be changed with with the argument `minsize` or, equivalently, `minbucket`. (The different names stem from slightly different conventions in the underlying tree functions.) Consequently, the minimum number of observations to consider splitting (`minsplit`) must be, at the very least, twice the minimum number of observations per subgroup (which would allow only one possible split, though). By default, `minsplit` is 2.5 times `minsize`. Users are encouraged to consider whether for their application it is sensible to increase or decrease these defaults. Finally, further control

parameters can be specified through the ... argument, see ctree_control and mob_control, respectively, for details.

In addition to the standard specification of the two response measurements in the formula via y1 + y2 ~ ..., it is also possible to use y1 - y2 ~ .... The latter may be more intuitive for users that think of it as a model for the difference of two measurements. Finally cbind(y1, y2) ~ ... also works. Internally, all of these are processed in the same way, namely as a bivariate dependent variable that can then be modeled and plotted appropriately.

To add the means of the measurement pair as a potential splitting variable, there are also different equivalent strategies. The standard specification would be via the means argument: y1 + y2 ~ x1 + ..., means = TRUE. Alternatively, the user can also extend the formula argument via y1 + y2 ~ x1 + ... + means(y1, y2).

The SD is estimated by the usual sample standard deviation in each subgroup, i.e., divided by the sample size $n - 1$. Note that the inference in the MOB algorithm internally uses the maximum likelihood estimate (divided by n) instead so the the fluctuation tests for parameter instability can be applied.

Value

Object of class coat, inheriting either from constparty (if ctree is used) or modelparty (if mob is used).

References

Karapetyan S, Zeileis A, Henriksen A, Hapfelmeier A (2023). "Tree Models for Assessing Covariate-Dependent Method Agreement." arXiv 2306.04456, *arXiv.org E-Print Archive*. doi: 10.48550/arXiv.2306.04456

Examples

```
## package and data (reshaped to wide format)
library("coat")
data("scint", package = "MethComp")
scint_wide <- reshape(scint, v.names = "y", timevar = "meth", idvar = "item", direction = "wide")

## coat based on ctree() without and with mean values of paired measurements as predictor
tr1 <- coat(y.DTPA + y.DMSA ~ age + sex, data = scint_wide)
tr2 <- coat(y.DTPA + y.DMSA ~ age + sex, data = scint_wide, means = TRUE)

## display
print(tr1)
plot(tr1)

print(tr2)
plot(tr2)

## tweak various graphical arguments of the panel function (just for illustration):
## different colors, nonparametric bootstrap percentile confidence intervals, ...
plot(tr1, tp_args = list(
  xscale = c(0, 150), linecol = "deeppink",
  confint = TRUE, B = 250, cilevel = 0.5, cicol = "gold"
```

```
))
```

`diffs` *Convenience Functions for Bland-Altman Analysis*

Description

Auxiliary functions for obtain the differences and means of a measurement pair, as used in the classic Bland-Altman analysis.

Usage

```
diffs(y1, y2)

means(y1, y2)
```

Arguments

`y1`, `y2`	numeric. Vectors of numeric measurements of the same length.

Value

Numeric vector with the differences or means of `y1` and `y2`, respectively.

Examples

```
## pair of measurements
y1 <- 1:4
y2 <- c(2, 2, 1, 3)

## differences and means
diffs(y1, y2)
means(y1, y2)
```

`print.coat` *Methods for Conditional Method Agreement Trees (COAT)*

Description

Extracting information from or visualization of conditional method agreement trees. Visualizations use trees with Bland-Altman plots in terminal nodes, drawn via grid graphics.

Usage

```
## S3 method for class 'coat'
print(
  x,
  digits = 2L,
  header = TRUE,
  footer = TRUE,
  title = "Conditional method agreement tree (COAT)",
  ...
)

## S3 method for class 'coat'
coef(object, node = NULL, drop = TRUE, ...)

## S3 method for class 'coat'
plot(x, terminal_panel = node_baplot, tnex = 2, drop_terminal = TRUE, ...)

node_baplot(
  obj,
  level = 0.95,
  digits = 2,
  pch = 1,
  cex = 0.5,
  col = 1,
  linecol = 4,
  lty = c(1, 2),
  bg = "white",
  confint = FALSE,
  B = 500,
  cilevel = 0.95,
  cicol = "lightgray",
  xscale = NULL,
  yscale = NULL,
  ylines = 3,
  id = TRUE,
  mainlab = NULL,
  gp = gpar()
)
```

Arguments

x, object, obj	a coat object as returned by coat.
digits	numeric. Number of digits used for rounding the displayed coefficients or limits of agreement.
header, footer	logical. Should a header/footer be printed for the tree?
title	character with the title for the tree.
...	further arguments passed to methods.

node	integer. ID of the node for which the Bland-Altman parameters (coefficients) should be extracted.
drop	logical. Should the matrix attribute be dropped if the parameters from only a single node are extracted?
terminal_panel	a panel function or panel-generating function passed to plot.party. By default, node_baplot is used to generate a suitable panel function for drawing Bland-Altman plots based on the the provided coat object. It can be customized using the tp_args argument (passed through ...).
tnex	numeric specification of the terminal node extension relative to the inner nodes (default is twice the size).
drop_terminal	logical. Should all terminal nodes be "dropped" to the bottom row?
level	numeric level for the limits of agreement.
pch, cex, col, linecol, lty, bg	graphical parameters for the scatter plot and limits of agreement in the Bland-Altman plot (scatter plot character, character extension, plot color, line color, line types, and background color).
confint	logical. Should nonparametric bootstrap percentile confidence intervals be plotted?
B	numeric. Number of bootstrap samples to be used if confint = TRUE.
cilevel	numeric. Level of the confidence intervals if confint = TRUE.
cicol	color specification for the confidence intervals if confint = TRUE.
xscale, yscale	numeric specification of scale of x-axis and y-axis, respectively. By default the range of all scatter plots and limits of agreement across all nodes are used.
ylines	numeric. Number of lines for spaces in y-direction.
id	logical. Should node IDs be plotted?
mainlab	character or function. An optional title for the plots. Either a character or a function(id, nobs).
gp	grid graphical parameters.

Details

Various methods are provided for trees fitted by coat, in particular print, plot (via **grid/partykit**) and coef. The plot method draws Bland-Altman plots in the terminal panels by default, using the function node_baplot.

In addition to these dedicated coat methods, further methods are inherited from ctree or mob, respectively, depending on which type of coat was fitted.

Value

The print() method returns the printed object invisibly. The coef() method returns the vector (for a single node) or matrix (for multiple nodes) of estimated parameters (bias and standard deviation). The plot() method returns NULL. The node_baplot() panel-generating function returns a function that can be plugged into the plot() method.

Examples

```
## package and data (reshaped to wide format)
library("coat")
data("scint", package = "MethComp")
scint_wide <- reshape(scint, v.names = "y", timevar = "meth", idvar = "item", direction = "wide")

## conditional method agreement tree
tr <- coat(y.DTPA + y.DMSA ~ age + sex, data = scint_wide)

## illustration of methods (including some customization)

## printing
print(tr)
print(tr, header = FALSE, footer = FALSE)

## extracting Bland-Altman parameters
coef(tr)
coef(tr, node = 1)

## visualization (via grid with node_baplot)
plot(tr)
plot(tr, ip_args = list(id = FALSE),
  tp_args = list(col = "slategray", id = FALSE, digits = 3, pch = 19))
```

Index