

Package: dda (via r-universe)

October 15, 2026

Type Package

Title Direction Dependence Analysis

Version 0.2.0

License MIT + file LICENSE

Maintainer Wolfgang Wiedermann <wiedermannw@missouri.edu>

Description A collection of tests to analyze the causal direction of dependence in linear models (Wiedermann, W., & von Eye, A., 2025, ISBN: 9781009381390). The package includes functions to perform Direction Dependence Analysis for variable distributions, residual distributions, and independence properties of predictors and residuals in competing causal models. In addition, the package contains functions to test the causal direction of dependence in conditional models (i.e., models with interaction terms). For more information see <<https://www.ddaproject.com>>.

Imports boot, dccpp, dHSIC, energy, graphics, interactions, lmtest, mblm, methods, moments, RobustLinearReg, stats

Suggests devtools, doParallel, harmonicmeanp, iterators, spelling, testthat (>= 3.0.0), waldo

Encoding UTF-8

RoxxygenNote 7.3.3

Config/testthat/edition 3

URL <https://github.com/wwiedermann/dda>

NeedsCompilation no

Author Wolfgang Wiedermann [aut, cre], Megan Hirni [aut]

BugReports <https://github.com/wwiedermann/dda/issues>

Config/pak/sysreqs libgsl0-dev libicu-dev

Repository <https://hirnia.r-universe.dev>

Date/Publication 2026-09-15 13:31:22 UTC

RemoteUrl https://github.com/hirnia/dda_archive

RemoteRef HEAD
RemoteSha 550b3921997b443cf274a462bd7634eaacee3ab8

Contents

cdda.indep	2
cdda.vardist	6
dda.bagging	9
dda.decisions	12
dda.indep	14
dda.resdist	16
dda.vardist	18
print.dda_bagging_indep	19
summary.dda_bagging_indep	22
summary_ols	23

Index	25
--------------	-----------

cdda.indep	<i>Conditional Direction Dependence Analysis: Independence Properties</i>
------------	---

Description

cdda.indep evaluates asymmetries of predictor-error independence of competing conditional models ($y \sim x * m$ vs. $x \sim y * m$ with m being a continuous or categorical moderator).

print returns the output of standard linear model coefficients for causally competing target and alternative models.

plot returns graphs of cdda.indep results.

summary returns test statistics from the cdda.indep class object.

Usage

```
cdda.indep(  
  formula = NULL,  
  pred = NULL,  
  mod = NULL,  
  modval = NULL,  
  data = list(),  
  hetero = FALSE,  
  diff = FALSE,  
  nlfun = NULL,  
  hsic.method = "gamma",  
  B = 200,  
  boot.type = "perc",  
  conf.level = 0.95,
```

```

    parallelize = FALSE,
    cores = 1,
    robust = FALSE,
    ...
)

## S3 method for class 'cdda.indep'
print(x, ...)

## S3 method for class 'cdda.indep'
plot(x = NULL, stat = NULL, ylim = NULL, ...)

## S3 method for class 'cdda.indep'
summary(
  object,
  nlfun = FALSE,
  hetero = FALSE,
  hsic = TRUE,
  hsic.diff = FALSE,
  dcor = TRUE,
  dcor.diff = FALSE,
  mi.diff = FALSE,
  ...
)
```

Arguments

formula	Symbolic formula of the model to be tested or an <code>lm</code> object.
pred	A character indicating the variable name of the predictor which serves as the outcome in the alternative model.
mod	A character indicating the variable name of the moderator.
modval	Characters or a numeric sequence specifying the moderator values used in post-hoc probing. Possible characters include <code>c("mean", "median", "JN")</code> . <code>modval = "mean"</code> tests the interaction effect at the moderator values $M - 1SD$, M , and $M + 1SD$; <code>modval = "median"</code> uses $Q1$, $Q2$, and $Q3$. The Johnson-Neyman approach is applied when <code>modval = "JN"</code> with conditional effects being evaluated at the boundary values of the significance regions. When a numeric sequence is specified, the pick-a-point approach is used for the selected numeric values.
data	A required data frame containing the variables in the model.
hetero	A logical value indicating whether separate homoscedasticity tests should be returned when using <code>summary</code> , default is <code>FALSE</code> .
diff	A logical value indicating whether differences in HSIC, dCor, and MI values should be computed. Bootstrap confidence intervals are computed using B bootstrap samples.
nlfun	Determines handling of non-linear correlation tests depending on the function used:

- **For** `cdda.indep`: Either a numeric value or a function of .Primitive type used for non-linear correlation tests. When numeric, the value is used in a power transformation.
- **For** `summary`: A logical value indicating whether non-linear correlation tests should be returned in the output. Default is FALSE.

<code>hsic.method</code>	A character indicating the inference method for the Hilbert-Schmidt Independence Criterion. Must be one of the four specifications <code>c("gamma", "eigenvalue", "bootstrap", "permutation")</code> . <code>hsic.method = "gamma"</code> is the default.
<code>B</code>	Number of permutations for separate dCor tests and number of resamples when <code>hsic.method = c("bootstrap", "permutation")</code> or <code>diff = TRUE</code> .
<code>boot.type</code>	A character indicating the type of bootstrap confidence intervals. Must be one of the two specifications <code>c("perc", "bca")</code> . <code>boot.type = "perc"</code> is the default.
<code>conf.level</code>	Confidence level for bootstrap confidence intervals.
<code>parallelize</code>	A logical value indicating whether bootstrapping is performed on multiple cores. Only used if <code>diff = TRUE</code> .
<code>cores</code>	A numeric value indicating the number of cores. Only used if <code>parallelize = TRUE</code> .
<code>robust</code>	A logical value indicating whether Siegel's (1982) repeated median estimator should be used for model estimation. If <code>robust = TRUE</code> repeated median estimation is applied for the causally competing models, otherwise ordinary least squares (OLS) estimation is used.
<code>...</code>	Additional arguments to be passed to the function.
<code>x</code>	An object of class <code>cdda.indep</code> when using <code>print</code> or <code>plot</code> .
<code>stat</code>	A character indicating the CDDA statistic to be plotted with the options <code>c("hsic.diff", "dcor.diff", "mi.diff")</code> .
<code>ylim</code>	A numeric vector of length 2 indicating the y-axis limits. If NULL, the function will set the limits automatically.
<code>object</code>	An object of class <code>cdda.indep</code> when using <code>summary</code> .
<code>hsic</code>	A logical value indicating whether separate HSIC tests should be returned when using <code>summary</code> , default is TRUE.
<code>hsic.diff</code>	A logical value indicating whether HSIC difference statistics should be returned when using <code>summary</code> , default is FALSE.
<code>dcor</code>	A logical value indicating whether separate Distance Correlation (dCor) tests should be returned when using <code>summary</code> , default is TRUE.
<code>dcor.diff</code>	A logical value indicating whether dCor difference statistics should be returned when using <code>summary</code> , default is FALSE.
<code>mi.diff</code>	A logical value indicating whether Mutual Information (MI) difference statistics should be returned when using <code>summary</code> , default is FALSE.

Value

A list of class `cdda.indep` containing the results of CDDA independence tests for pre-specified moderator values.

An object of class `cdda.indep` with competing model coefficients.

References

Wiedermann, W., & von Eye, A. (2025). *Direction Dependence Analysis: Foundations and Statistical Methods*. Cambridge, UK: Cambridge University Press.

See Also

[dda.indep](#) for an unconditional version.

Examples

```
set.seed(321)
n <- 700

## --- generate moderator

z <- sort(rnorm(n))
z1 <- z[z <= 0]
z2 <- z[z > 0]

## --- x -> y when z <= 0

x1 <- rchisq(length(z1), df = 4) - 4
e1 <- rchisq(length(z1), df = 3) - 3
y1 <- 0.5 * x1 + e1

## --- y -> x when m z > 0

y2 <- rchisq(length(z2), df = 4) - 4
e2 <- rchisq(length(z2), df = 3) - 3
x2 <- 0.25 * y2 + e2

y <- c(y1, y2); x <- c(x1, x2)

d <- data.frame(x, y, z)

m <- lm(y ~ x * z, data = d)

result <- cdda.indep(m,
  pred = "x",
  mod = "z",
  modval = c(-1, 1),
  data = d,
  hetero = TRUE,
  diff = TRUE,
  parallelize = TRUE,
  cores = 2,
  nlfun = 2,
  B = 2)
# Note: Only 2 bootstrap samples are created here to lower computation time

## Not run:
```

```
## --- Realistic settings; run time is substantial

result <- cdda.indep(m,
  pred = "x",
  mod = "z",
  modval = c(-1, 1),
  data = d,
  hetero = TRUE,
  diff = TRUE,
  parallelize = TRUE,
  cores = 2,
  nlfun = 2,
  B = 500)

print(result)
summary(result, hsic.diff = TRUE, dcor.diff = TRUE, mi.diff = TRUE)
plot(result, stat = "dcor.diff")

## End(Not run)

print(result)

plot(result, stat = "dcor.diff")

summary(result, hetero = FALSE)
```

cdda.vardist

Conditional Directional Dependence Analysis: Variable Distributions

Description

cdda.vardist evaluates variable distributions of competing conditional models ($y \sim x * m$ vs. $x \sim y * m$ with m being a continuous or categorical moderator).

print returns the output of standard linear model coefficients for causally competing target and alternative models.

plot returns graphs of cdda.vardist results.

summary returns test statistics from the cdda.vardist class object.

Usage

```
cdda.vardist(
  formula,
  pred = NULL,
  mod = NULL,
  data = list(),
  modval = NULL,
```

```

    B = 200,
    boot.type = "perc",
    conf.level = 0.95
)

## S3 method for class 'cdda.vardist'
print(x, ...)

## S3 method for class 'cdda.vardist'
plot(x, stat = NULL, ylim = NULL, ...)

## S3 method for class 'cdda.vardist'
summary(object, skew = TRUE, coskew = FALSE, kurt = TRUE, cokurt = FALSE, ...)
```

Arguments

formula	Symbolic formula of the model to be tested or an lm object
pred	A character indicating the variable name of the predictor which serves as the outcome in the alternative model.
mod	A character indicating the variable name of the moderator.
data	A required data frame containing the variables in the model.
modval	Characters or a numeric sequence specifying the moderator values used in post-hoc probing. Possible characters include c("mean", "median", "JN"). modval = "mean" tests the interaction effect at the moderator values $M - 1SD$, M , and $M + 1SD$; modval = "median" uses Q1, Md, and Q3. The Johnson-Neyman approach is applied when modval = "JN" with conditional effects being evaluated at the boundary values of the significance regions. When a numeric sequence is specified, the pick-a-point approach is used for the selected numeric values.
B	Number of bootstrap samples.
boot.type	A character indicating the type of bootstrap confidence intervals. Must be one of the two values c("perc", "bca"). boot.type = "perc" is the default.
conf.level	Confidence level for bootstrap confidence intervals.
x	An object of class cdda.vardist when using print or plot.
...	Additional arguments to be passed to the function.
stat	A character indicating the statistic to be plotted, default is "rhs", with options c("coskew", "cokurt", "rhs", "rcc", "rtanh").
ylim	A numeric vector of length 2 indicating the y-axis limits. If NULL, the function will set the limits automatically.
object	An object of class cdda.vardist when using summary.
skew	A logical value indicating whether skewness differences and separate D'Agostino skewness tests should be returned when using summary, default is TRUE.
coskew	A logical value indicating whether co-skewness differences should be returned when using summary, default is FALSE.
kurt	A logical value indicating whether excess kurtosis differences and Anscombe-Glynn kurtosis tests should be returned when using summary, default is TRUE.

cokurt A logical value indicating whether co-kurtosis differences should be returned when using summary, default is FALSE.

Value

An object of class `cdda.vardist` containing the results of conditional direction dependence tests of variable distributions.

References

Wiedermann, W., & von Eye, A. (2025). *Direction Dependence Analysis: Foundations and Statistical Methods*. Cambridge, UK: Cambridge University Press.

See Also

[dda.vardist](#) for an unconditional version.

Examples

```
set.seed(321)
n <- 700

## --- generate moderator

z <- sort(rnorm(n))
z1 <- z[z <= 0]
z2 <- z[z > 0]

## --- x -> y when z <= 0

x1 <- rchisq(length(z1), df = 4) - 4
e1 <- rchisq(length(z1), df = 3) - 3
y1 <- 0.5 * x1 + e1

## --- y -> x when m z > 0

y2 <- rchisq(length(z2), df = 4) - 4
e2 <- rchisq(length(z2), df = 3) - 3
x2 <- 0.25 * y2 + e2

y <- c(y1, y2); x <- c(x1, x2)

d <- data.frame(x, y, z)

m <- lm(y ~ x * z, data = d)

result <- cdda.vardist(m, pred = "x", mod = "z", B = 50,
  modval = c(-1, 1), data = d)
# Note: Only 50 bootstrap samples are created here to lower computation time

## Not run:
## --- Realistic settings; run time is substantial
```

```

result <- cdda.vardist(m, pred = "x", mod = "z", B = 500,
                     modval = c(-1, 1), data = d)

print(result)
plot(result, stat = "rtanh", ylim = c(-0.05, 0.05))
summary(result, skew = FALSE, kurt = FALSE, coskew = TRUE)

## End(Not run)

print(result)

plot(result, stat = "rtanh", ylim = c(-0.05, 0.05))

summary(result, skew = FALSE, kurt = FALSE, coskew = TRUE)

```

dda.bagging

*Bootstrap Aggregated Direction Dependence Analysis (DDA)***Description**

dda.bagging performs bootstrap aggregation (bagging) on an existing Direction Dependence Analysis (DDA) object to test the stability and robustness of direction dependence decisions.

Usage

```

dda.bagging(
  dda_result,
  iter = 100,
  progress = TRUE,
  save_file = NULL,
  alpha = 0.05,
  data = NULL,
  agg_stat = c("mean", "median", "trimmed", "winsorized", "midhinge", "tukey"),
  trim_prob = 0.1,
  win_prob = 0.1,
  inner_B = NULL,
  nlcor.adjust = c("none", "bonferroni")
)

```

Arguments

dda_result	An output object from any base DDA function (e.g., dda.indep, dda.vardist, or dda.resdist).
iter	Number of bootstrap samples (default: 100).
progress	Logical. Whether to display a progress bar during resampling (default: TRUE).

save_file	Character. Optional file path used to save the resulting object as an R serialized data file (e.g., "results.rds").
alpha	Numeric. Significance level used for causal model selection (default: 0.05).
data	A data.frame containing ALL raw variables used in the original DDA model (outcome, predictor, and any covariates).
agg_stat	Character. Specifies the method used for aggregating test statistics and coefficients across bootstrap samples. Must be one of the following specifications <code>c("mean", "median", "trimmed", "winsorized", "midhinge", "tukey")</code> .
trim_prob	Numeric. Proportion of observations to be trimmed from each side of the sampling distribution when <code>agg_stat = "trimmed"</code> (default: 0.10).
win_prob	Numeric. Proportion of observations to be winsorized from each side of the sampling distribution when <code>agg_stat = "winsorized"</code> (default: 0.10).
inner_B	Optional positive integer. Caps the number of inner bootstrap resamples (B) passed to each per-iteration DDA call. NULL (default) keeps whatever B was used in the original DDA call. Every base DDA function runs its own resampling on each of the <code>iter</code> outer iterations, so total cost grows as $\text{iter} \times B$; capping <code>inner_B</code> is the main lever for reducing run time without changing how the outer bagging loop aggregates results. The effect is largest for <code>dda.indep</code> called with <code>diff = TRUE</code> , where each outer iteration runs a full difference-statistic bootstrap, and for <code>dda.resdist</code> and <code>dda.vardist</code> , which bootstrap their confidence intervals internally.
nlcor.adjust	Character. Multiplicity adjustment passed to dda.decisions for the non-linear correlation decision. One of "none" (default) or "bonferroni".

Details

This function uses a fitted DDA output object (obtained from `dda.indep`, `dda.vardist`, or `dda.resdist`) and performs bootstrap aggregation of DDA test statistics. The function computes DDA statistics across `iter` bootstrap samples and aggregates the results to evaluate the stability and robustness of DDA model selection. p-values obtained from significance tests are aggregated using the harmonic mean p-value approach (Wilson, 2019). Model selection decisions within each bootstrap sample are obtained with [dda.decisions](#) (p-value based rules for significance tests, interval-excludes-zero rules for bootstrap difference statistics) and reported as decision proportions across samples.

Run time scales with `iter` multiplied by the resampling budget of the base DDA call, and the underlying independence statistics are quadratic in the number of observations. Use `inner_B` to cap the inner budget. Note also that if the original DDA call used `parallelize = TRUE`, that setting is inherited by every outer iteration, so a new cluster is started `iter` times; for the small inner bootstraps typical of bagging this is usually slower than running the inner calls serially.

Value

An object of class `dda_bagging` (with subclasses `dda_bagging_indep`, `dda_bagging_vardist`, or `dda_bagging_resdist`), which contains aggregated and raw results from tests matching the initial DDA function (`dda.indep`, `dda.vardist`, or `dda.resdist`).

References

- Wiedermann, W., & von Eye, A. (2025). *Direction Dependence Analysis: Foundations and Statistical Methods*. Cambridge, UK: Cambridge University Press.
- Wilson, D. J. (2019). The harmonic mean p-value for combining dependent tests. *Proceedings of the National Academy of Sciences*, 116(4), 1195–1200.

See Also

[dda.indep](#), [dda.vardist](#), [dda.resdist](#), [dda.decisions](#)

Examples

```
set.seed(123)
n <- 200
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

## --- Fit a base DDA independence model

base_model <- dda.indep(y ~ x, pred = "x", data = d, B = 10, hetero = TRUE)

## --- Bootstrap aggregation of the base model

bagged <- dda.bagging(base_model, data = d, iter = 5, agg_stat = "mean",
  inner_B = 10, progress = FALSE)
# Note: n, B and iter are all kept small here to lower computation time.
# inner_B caps the resampling budget of each outer iteration.

print(bagged)
summary(bagged, show = c("hsic", "dcor"))

## Not run:
## --- Realistic settings; run time is substantial

base_model <- dda.indep(y ~ x, pred = "x", data = d, B = 500,
  hetero = TRUE, nlfun = 2, diff = TRUE)

bagged <- dda.bagging(base_model, data = d, iter = 200,
  agg_stat = "trimmed", trim_prob = 0.05, progress = TRUE)

print(bagged)
summary(bagged, show = c("hsic", "dcor", "bp"))
print_ols_summary(bagged)

## End(Not run)
```

dda.decisions

*Model Selection Decisions for a Fitted DDA Object***Description**

dda.decisions translates the tests stored in a fitted DDA object (dda.indep, dda.resdist, or dda.vardist) into causal model selection decisions. Significance tests are compared against alpha; difference statistics are decided by whether their bootstrap confidence interval excludes zero. The same rules are used inside dda.bagging, so simulation code can call dda.decisions directly instead of re-implementing decision logic.

Usage

```
dda.decisions(dda_result, alpha = 0.05, nlcor.adjust = c("none", "bonferroni"))
```

Arguments

dda_result	An output object from dda.indep, dda.resdist, or dda.vardist.
alpha	Numeric. Significance level used for causal model selection (default: 0.05).
nlcor.adjust	Character. Multiplicity adjustment applied to the smallest non-linear correlation p-value across the three transformations (dda.indep only). "none" (default) uses the smallest p-value directly. "bonferroni" multiplies it by the number of transformations.

Details

Throughout, the target model is $x \rightarrow y$ and the alternative model is $y \rightarrow x$; p_{yx} denotes a p-value obtained under the target model and p_{xy} a p-value obtained under the alternative model.

Separate independence tests (dda.indep: HSIC, dCor, robust Breusch-Pagan, non-linear correlation):

- $p_{yx} > \alpha$ and $p_{xy} \leq \alpha$: "Target"
- $p_{yx} \leq \alpha$ and $p_{xy} > \alpha$: "Alternative"
- both $\leq \alpha$: "Confounding"
- both $> \alpha$: "Undecided"

The non-linear correlation decision combines the three transformations by taking the smallest p-value in each direction, matching the reference implementation. That combination is not adjusted for multiplicity by default; nlcor.adjust = "bonferroni" applies a correction, which lowers the rate of decisive outcomes when neither direction is identified.

Difference statistics (HSIC, dCor, MI) use their bootstrap confidence intervals: an interval above zero speaks for the target model, below zero for the alternative model, otherwise the decision is "Undecided".

Separate normality tests on residuals (dda.resdist, D'Agostino and Anscombe-Glynn) depend on the prob.trans setting of the fitted object. Under prob.trans = FALSE:

- $p_{yx} > \alpha$ and $p_{xy} \leq \alpha$: "Target"
- $p_{yx} \leq \alpha$ and $p_{xy} > \alpha$: "Alternative"
- otherwise: "Undecided"

Under `prob.trans = TRUE` the roles are reversed ($p_{yx} \leq \alpha$ and $p_{xy} > \alpha$ speaks for the target model). The skewness and kurtosis differences reverse in the same way, so that under `prob.trans = TRUE` a difference below zero speaks for the target model. Co-skewness, co-kurtosis and the likelihood-ratio approximations never reverse: a difference above zero speaks for the target model under both settings.

When `dda.resdist` is called with `B = 0` no bootstrap interval is available for the skewness and kurtosis differences. The asymptotic difference test stored alongside them is used instead, combining its p-value with the sign of the difference.

Separate normality tests on observed variables (`dda vardist`):

- outcome $p > \alpha$ and predictor $p \leq \alpha$: "Target"
- outcome $p \leq \alpha$ and predictor $p > \alpha$: "Alternative"
- otherwise: "Undecided"

All higher-moment difference intervals in `dda vardist` use the interval-excludes-zero rule with values above zero speaking for the target model.

Value

A named character vector of decisions, one element per test available in the object. Values are "Target", "Alternative", "Undecided", "Confounding" (separate independence tests only), or NA when a test result is missing.

See Also

[dda.bagging](#)

Examples

```
set.seed(123)
n <- 200
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

fit <- dda.vardist(y ~ x, pred = "x", data = d, B = 50)
dda.decisions(fit, alpha = 0.05)
```

Description

dda.indep evaluates asymmetries of predictor-error independence of causally competing models ($y \sim x$ vs. $x \sim y$). print returns DDA test statistics associated with dda.indep objects.

Usage

```
dda.indep(
  formula,
  pred = NULL,
  data = list(),
  nlfun = NULL,
  hetero = FALSE,
  hsic.method = "gamma",
  diff = FALSE,
  B = 200,
  boot.type = "perc",
  conf.level = 0.95,
  parallelize = FALSE,
  cores = 1,
  robust = FALSE
)

## S3 method for class 'dda.indep'
print(x, ...)
```

Arguments

formula	Symbolic formula of the model to be tested or a lm object.
pred	A character indicating the variable name of the predictor which serves as the outcome in the alternative model.
data	An optional data frame containing the variables in the model (by default variables are taken from the environment which dda.indep is called from).
nlfun	Either a numeric value or a function of .Primitive type used for non-linear correlation tests. When nlfun is numeric the value is used in a power transformation.
hetero	A logical value indicating whether separate homoscedasticity tests (i.e., standard and robust Breusch-Pagan tests) should be computed.
hsic.method	A character indicating the inference method for the Hilbert-Schmidt Independence Criterion (HSIC). Must be one of the four specifications c("gamma", "eigenvalue", "bootstrap", "permutation"). hsic.method = "gamma" is the default.

<code>diff</code>	A logical value indicating whether differences in HSIC, Distance Correlation (dCor), and MI values should be computed. Bootstrap confidence intervals are computed using <code>B</code> bootstrap samples.
<code>B</code>	Number of permutations for separate dCor tests and number of resamples if <code>hsic.method = c("bootstrap", "permutation")</code> or <code>diff = TRUE</code> .
<code>boot.type</code>	A vector of character strings representing the type of bootstrap confidence intervals. Must be one of the two specifications <code>c("perc", "bca")</code> . <code>boot.type = "perc"</code> is the default.
<code>conf.level</code>	Confidence level for bootstrap confidence intervals.
<code>parallelize</code>	A logical value indicating whether bootstrapping is performed on multiple cores. Only used if <code>diff = TRUE</code> .
<code>cores</code>	A numeric value indicating the number of cores. Only used if <code>parallelize = TRUE</code> .
<code>robust</code>	A logical value indicating whether Siegel's (1982) repeated median estimator should be used for model estimation. If <code>robust = TRUE</code> repeated median estimation is applied for the causally competing models, otherwise ordinary least squares (OLS) estimation is used.
<code>x</code>	An object of class <code>dda.indep</code> when using <code>print</code> .
<code>...</code>	Additional arguments to be passed to the function.

Value

An object of class `dda.indep` containing the results of DDA independence tests.

References

Wiedermann, W., & von Eye, A. (2025). *Direction Dependence Analysis: Foundations and Statistical Methods*. Cambridge, UK: Cambridge University Press.

See Also

[cdda.indep](#) for a conditional version.

Examples

```
set.seed(123)
n <- 500
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

result <- dda.indep(y ~ x, pred = "x", data = d, parallelize = TRUE, cores = 2,
  nlfun = 2, B = 50, hetero = TRUE, diff = TRUE, robust = FALSE)

print(result)

## Not run:
```

```
## --- Realistic settings; run time is substantial

result <- dda.indep(y ~ x, pred = "x", data = d, parallelize = TRUE,
  cores = 2, nlfun = 2, B = 500, hetero = TRUE, diff = TRUE)

print(result)

## --- Permutation-based HSIC inference

dda.indep(y ~ x, pred = "x", data = d, hsic.method = "permutation", B = 500)

## --- Siegel (1982) repeated median estimation

dda.indep(y ~ x, pred = "x", data = d, robust = TRUE, hetero = TRUE, B = 500)

## End(Not run)

print(result)
```

dda.resdist

Direction Dependence Analysis: Residual Distributions

Description

dda.resdist evaluates patterns of asymmetry of error distributions of causally competing models ($y \sim x$ vs. $x \sim y$). print returns DDA test statistics associated with dda.resdist objects.

Usage

```
dda.resdist(
  formula,
  pred = NULL,
  data = list(),
  B = 200,
  boot.type = "perc",
  prob.trans = FALSE,
  conf.level = 0.95,
  robust = FALSE
)

## S3 method for class 'dda.resdist'
print(x, ...)
```

Arguments

formula	Symbolic formula of the target model to be tested or an lm object.
pred	Variable name of the predictor which serves as the outcome in the alternative model.

<code>data</code>	An optional data frame containing the variables in the model (by default variables are taken from the environment which <code>dda.resdist</code> is called from).
<code>B</code>	Number of bootstrap samples.
<code>boot.type</code>	A vector of character strings representing the type of bootstrap confidence intervals. Must be one of the two values <code>c("perc", "bca")</code> ; <code>boot.type = "perc"</code> is the default.
<code>prob.trans</code>	A logical value indicating whether a probability integral transformation should be performed prior computation of skewness and kurtosis tests.
<code>conf.level</code>	Confidence level for bootstrap confidence intervals.
<code>robust</code>	A logical value indicating whether Siegel's (1982) repeated median estimator should be used for model estimation. If <code>robust = TRUE</code> repeated median estimation is applied for the causally competing models, otherwise ordinary least squares (OLS) estimation is used.
<code>x</code>	An object of class <code>dda.resdist</code> when using <code>print</code> .
<code>...</code>	Additional arguments to be passed to the method.

Value

An object of class `dda.resdist` containing the results of direction dependence tests of error distributions.

References

Wiedermann, W., & von Eye, A. (2025). *Direction Dependence Analysis: Foundations and Statistical Methods*. Cambridge, UK: Cambridge University Press.

Examples

```
set.seed(123)
n <- 500
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

result <- dda.resdist(y ~ x, pred = "x", data = d,
  B = 50, conf.level = 0.90, prob.trans = TRUE, robust = FALSE)

print(result)

## Not run:
## --- Realistic settings; run time is substantial

result <- dda.resdist(y ~ x, pred = "x", data = d, B = 500,
  boot.type = "bca", conf.level = 0.95, prob.trans = TRUE)

print(result)

## --- Siegel (1982) repeated median estimation
```

```

dda.resdist(y ~ x, pred = "x", data = d, B = 500, robust = TRUE)

## End(Not run)

## Not run:
print(result)

## End(Not run)

```

dda.vardist

Direction Dependence Analysis: Variable Distributions

Description

dda.vardist evaluates patterns of asymmetry of variable distributions for causally competing models ($y \sim x$ vs. $x \sim y$). print returns DDA test statistics associated with dda.vardist objects.

Usage

```

dda.vardist(
  formula,
  pred = NULL,
  data = list(),
  B = 200,
  boot.type = "perc",
  conf.level = 0.95
)

## S3 method for class 'dda.vardist'
print(x, ...)

```

Arguments

formula	Symbolic formula of the model to be tested or a lm object.
pred	Variable name of the predictor which serves as the outcome in the alternative model.
data	An optional data frame containing the variables in the model (by default variables are taken from the environment which dda.vardist is called from).
B	Number of bootstrap samples.
boot.type	A character indicating the type of bootstrap confidence intervals. Must be one of the two specifications c("perc", "bca"). boot.type = "perc" is the default.
conf.level	Confidence level for bootstrap confidence intervals.
x	An object of class dda.vardist when using print.
...	Additional arguments to be passed to the function.

Value

An object of class `dda.vardist` containing the results of direction dependence tests of variable distributions.

References

Wiedermann, W., & von Eye, A. (2025). *Direction Dependence Analysis: Foundations and Statistical Methods*. Cambridge, UK: Cambridge University Press.

See Also

[cdda.vardist](#) for a conditional version.

Examples

```
set.seed(123)
n <- 500
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

result <- dda.vardist(y ~ x, pred = "x", data = d, B = 50)
# Note: Only 50 bootstrap samples are used here to lower computation time

## Not run:
## --- Realistic settings; run time is substantial

result <- dda.vardist(y ~ x, pred = "x", data = d, B = 500,
  boot.type = "bca", conf.level = 0.95)

print(result)

## End(Not run)

print(result)
```

```
print.dda_bagging_indep
```

Print Methods for Bootstrap Aggregated DDA Objects

Description

`print` returns aggregated test statistics and summaries of bootstrap aggregated Direction Dependence Analysis (DDA) objects. The function supports independence properties (obtained from `dda.indep`), residual distributions (obtained from `dda.resdist`), variable distributions (obtained from `dda.vardist`), and OLS summaries (`print_ols_summary`).

Usage

```
## S3 method for class 'dda_bagging_indep'
print(
  x,
  agg_stat = NULL,
  trim_prob = 0.1,
  win_prob = 0.1,
  digits = 4,
  alpha = 0.05,
  ...
)

## S3 method for class 'dda_bagging_resdist'
print(x, agg_stat = NULL, trim_prob = 0.1, win_prob = 0.1, digits = 4, ...)

## S3 method for class 'dda_bagging_vardist'
print(x, agg_stat = NULL, trim_prob = 0.1, win_prob = 0.1, digits = 4, ...)

print_ols_summary(
  object,
  agg_stat = NULL,
  trim_prob = 0.1,
  win_prob = 0.1,
  digits = 4,
  ...
)
```

Arguments

<code>x</code>	An object of class <code>dda_bagging_indep</code> , <code>dda_bagging_resdist</code> , or <code>dda_bagging_vardist</code> .
<code>agg_stat</code>	Character. Specifies the method used for aggregating test statistics and coefficients across bootstrap samples. Must be one of the following specifications <code>c("mean", "median", "trimmed", "winsorized", "midhinge", "tukey")</code> . If <code>NULL</code> , the method established in <code>dda.bagging()</code> is used.
<code>trim_prob</code>	Numeric. Proportion of observations to be trimmed on each side of the sampling distribution when <code>agg.stat = "trimmed"</code> (default: 0.10).
<code>win_prob</code>	Numeric. Proportion of observations to be winsorized on each side of the sampling distribution when <code>agg.stat = "winsorized"</code> (default: 0.10).
<code>digits</code>	Integer. Number of digits used for rounding (default: 4).
<code>alpha</code>	Numeric. Significance level used for causal model selection (default: 0.05).
<code>...</code>	Additional arguments passed to <code>print</code> .
<code>object</code>	Aggregated bagging output, used by <code>print_ols_summary</code> .

Value

Returns a summary of the bootstrap aggregated DDA object.

Examples

```

set.seed(123)
n <- 200
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

base_model <- dda.indep(y ~ x, pred = "x", data = d, B = 20,
  hetero = TRUE, nlfun = 2, diff = TRUE)
bagged <- dda.bagging(base_model, data = d, iter = 10, inner_B = 20,
  agg_stat = "mean", progress = FALSE)

# Default print
print(bagged)

# Override aggregation method at print time
print(bagged, agg_stat = "trimmed", trim_prob = 0.05)

# Print aggregated OLS summaries
print_ols_summary(bagged)

set.seed(123)
n <- 200
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

base_rd <- dda.resdist(y ~ x, pred = "x", data = d, B = 20)
bagged_rd <- dda.bagging(base_rd, data = d, iter = 10, inner_B = 20,
  progress = FALSE)
print(bagged_rd)
print(bagged_rd, agg_stat = "median")

set.seed(123)
n <- 200
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

base_vd <- dda.vardist(y ~ x, pred = "x", data = d, B = 20)
bagged_vd <- dda.bagging(base_vd, data = d, iter = 10, inner_B = 20,
  progress = FALSE)
print(bagged_vd)
print(bagged_vd, agg_stat = "winsorized", win_prob = 0.10)

```

summary.dda_bagging_indep

Summary Methods for Bootstrap Aggregated DDA Objects

Description

summary returns the proportion of model selection decisions (i.e., target model, alternative model, or presence of hidden confounding) across the bootstrap samples for bootstrap aggregated Direction Dependence Analysis (DDA) objects.

Usage

```
## S3 method for class 'dda_bagging_indep'
summary(object, show = NULL, digits = 2, ...)

## S3 method for class 'dda_bagging_varDIST'
summary(object, show = NULL, moment = NULL, digits = 2, ...)

## S3 method for class 'dda_bagging_resDIST'
summary(object, show = NULL, moment = NULL, digits = 2, ...)
```

Arguments

object	An object of class dda_bagging_indep, dda_bagging_varDIST, or dda_bagging_resDIST.
show	Character vector specifying the DDA statistics used to perform model selection. The function accepts the following specifications c("all", "hsic", "dcOR", "mi", "bp", "nlcor", "skew", "kurt", "coskew", "cokurt"). If NULL, shows default statistics based on the object type.
digits	Integer. Number of decimal places to print for proportions (default: 2).
...	Additional arguments passed to summary.
moment	Numeric vector indicating which moments to include in the summary (e.g., c(3, 4)). Applicable only to dda_bagging_varDIST and dda_bagging_resDIST.

Value

Invisibly returns the original object.

summary_ols

*Print Summary for OLS Models of Bootstrap Aggregated DDA Objects***Description**

summary_ols returns aggregated ordinary least squares (OLS) regression summaries for the causally competing target and alternative models of a bootstrap aggregated DDA object. Regression coefficients and standard errors are aggregated across bootstrap samples using the method specified in dda.bagging() or overridden via agg.stat.

Usage

```
summary_ols(
  object,
  agg_stat = NULL,
  trim_prob = 0.1,
  win_prob = 0.1,
  digits = 4,
  ...
)
```

Arguments

object	Output from dda.bagging().
agg_stat	Character. Specifies the method used for aggregating test statistics and coefficients across bootstrap samples. Must be one of the following specifications c("mean", "median", "trimmed", "winsorized", "midhinge", "tukey"). If NULL, the function uses the method applied with dda.bagging().
trim_prob	Numeric. Proportion of observations to be trimmed on each side of the sampling distribution when agg.stat = "trimmed" (default: 0.10).
win_prob	Numeric. Proportion of observations to be winsorized on each side of the sampling distribution when agg.stat = "winsorized" (default: 0.10).
digits	Integer. Number of digits used for rounding.
...	Additional arguments passed to print.

Examples

```
set.seed(123)
n <- 200
x <- rchisq(n, df = 4) - 4
e <- rchisq(n, df = 3) - 3
y <- 0.5 * x + e
d <- data.frame(x, y)

base_model <- dda.indep(y ~ x, pred = "x", data = d, B = 20)
bagged <- dda.bagging(base_model, data = d, iter = 10, inner_B = 20,
```

```
        progress = FALSE)

# Print aggregated OLS coefficients for target and alternative models
summary_ols(bagged)

# Override aggregation method
summary_ols(bagged, agg_stat = "median")
```

Index

`cdda.indep`, [2](#), [15](#)
`cdda.vardist`, [6](#), [19](#)

`dda.bagging`, [9](#), [13](#)
`dda.decisions`, [10](#), [11](#), [12](#)
`dda.indep`, [5](#), [11](#), [14](#)
`dda.resdist`, [11](#), [16](#)
`dda.vardist`, [8](#), [11](#), [18](#)

`plot.cdda.indep (cdda.indep)`, [2](#)
`plot.cdda.vardist (cdda.vardist)`, [6](#)
`print.cdda.indep (cdda.indep)`, [2](#)
`print.cdda.vardist (cdda.vardist)`, [6](#)
`print.dda.indep (dda.indep)`, [14](#)
`print.dda.resdist (dda.resdist)`, [16](#)
`print.dda.vardist (dda.vardist)`, [18](#)
`print.dda_bagging_indep`, [19](#)
`print.dda_bagging_resdist`
 (`print.dda_bagging_indep`), [19](#)
`print.dda_bagging_vardist`
 (`print.dda_bagging_indep`), [19](#)
`print_ols_summary`
 (`print.dda_bagging_indep`), [19](#)

`summary.cdda.indep (cdda.indep)`, [2](#)
`summary.cdda.vardist (cdda.vardist)`, [6](#)
`summary.dda_bagging_indep`, [22](#)
`summary.dda_bagging_resdist`
 (`summary.dda_bagging_indep`), [22](#)
`summary.dda_bagging_vardist`
 (`summary.dda_bagging_indep`), [22](#)
`summary_ols`, [23](#)