

Package: aRD (via r-universe)

June 8, 2026

Type Package

Title Adjusted Risk Differences via Specifically Penalized Likelihood

Version 0.1.0

Date 2025-08-18

Description Fits a linear-binomial model using a modified Newton-type algorithm for solving the maximum likelihood estimation problem under linear box constraints. Similar methods are described in Wagenpfeil, Schöpe and Bekhit (2025, ISBN:9783111341972) `` Estimation of adjusted relative risks in log-binomial regression using the BSW algorithm". In: Mau, Mukhin, Wang and Xu (Eds.), Biokybernetika. De Gruyter, Berlin, pp. 665–676.

License GPL (>= 3)

Encoding UTF-8

URL <https://github.com/UdS-MF-IMBEI/aRD>

BugReports <https://github.com/UdS-MF-IMBEI/aRD/issues>

VignetteBuilder knitr

Depends R (>= 4.0)

Imports Matrix, matrixStats, osqp, quadprog, methods, stats, boot, checkmate

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Config/pak/sysreqs make

Repository <https://uds-mf-imbei.r-universe.dev>

Date/Publication 2025-08-26 10:48:51 UTC

RemoteUrl <https://github.com/uds-mf-imbei/ard>

RemoteRef HEAD

RemoteSha 79b2c88de4cdf248742fb5cdece571c018b44584

Contents

aRD	2
aRD-class	4
bootaRD	4
coef,aRD-method	6
confint,aRD-method	7
constr	8
gradF	9
hess	10
obj_value	10
summary,aRD-method	11
variable_selection_aRD	12
Index	14

aRD	<i>Adjusted Risk differences via specifically penalized likelihood</i>
-----	--

Description

aRD() fits a linear-binomial model using a modified Newton-type algorithm (aRD algorithm) for solving the maximum likelihood estimation problem under linear box constraints.

Usage

```
aRD(formula, data, maxit = 80000L, solver = "quadprog",
     verbose = FALSE, polish_final = TRUE, eps_abs = 1e-5,
     eps_rel = 1e-5, conswitch = 1, use_nearPD = TRUE)
```

Arguments

formula	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data	A data frame containing the variables in the model.
maxit	A positive integer giving the maximum number of iterations.
solver	A character string specifying the solver to use. Options are "osqp" () or "quadprog" ().
verbose	Logical; if TRUE, a detailed output is printed, including iteration logs, armijo steps, objective values and solver messages (e.g., from osqp).
polish_final	Logical; if TRUE and solver is "osqp", performs a final polish step.
eps_abs	Absolute tolerance for solver convergence (only used with "osqp").
eps_rel	Relative tolerance for solver convergence (only used with "osqp").
conswitch	Specifies how the constraint matrix is constructed:

- 1 (default)** Generates all possible combinations of minimum and maximum values for the predictors (excluding the intercept), resulting in 2^{m-1} constraints. This formulation constrains model predictions within the observed data range, making it suitable for both risk factor identification and prediction (prognosis).
 - 0** Uses the raw design matrix x as the constraint matrix, resulting in n constraints. This is primarily suitable for identifying risk factors, but not for prediction tasks, as predictions are not bounded to realistic ranges.
- use_nearPD Logical; if TRUE, the Hessian matrix is projected to the nearest positive definite matrix using `Matrix::nearPD()` to ensure numerical stability, especially when using solvers that require a positive definite matrix (e.g., quadprog). If FALSE, the raw Hessian is used directly, which is faster but may lead to numerical issues if the matrix is not positive definite. Default is TRUE.

Value

An object of S4 class "aRD" containing the following slots:

call	An object of class "call".
formula	An object of class "formula".
coefficients	A numeric vector containing the estimated model parameters.
iter	A positive integer indicating the number of iterations.
converged	A logical constant that indicates whether the model has converged.
y	A numerical vector containing the dependent variable of the model.
x	The model matrix.
data	A data frame containing the variables in the model.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

References

Wagenpfeil S, Schöpe J, Bekhit A (2025) Estimation of adjusted relative risks in log-binomial regression using the BSW algorithm. In: Mau J, Mukhin S, Wang G, Xu S (Eds.) Biokybernetika. DE GRUYTER, Berlin, Germany, pp. 665–676.

Wagenpfeil S (1991) Implementierung eines SQP-Verfahrens mit dem Algorithmus von Ritter und Best. Diplomarbeit, TUM, Munich, Germany

Stellato B, Banjac G, Goulart P, Boyd S, Bansal V (2024). *osqp: Quadratic Programming Solver using the 'OSQP' Library*. R package version 0.6.3.3. doi:10.32614/CRAN.package.osqp. Available at: <https://CRAN.R-project.org/package=osqp>.

Turlach BA, Weingessel A, Moler C (2019). *quadprog: Functions to Solve Quadratic Programming Problems*. R package version 1.5-8. doi:10.32614/CRAN.package.quadprog. Available at: <https://CRAN.R-project.org/package=quadprog>.

Examples

```

set.seed(123)
n <- 100
x <- rnorm(n, 50, 1)
y <- rbinom(n, 1, -1.5 + 0.04 * x)
fit <- aRD(formula = y ~ x, data = data.frame(y = y, x = x),
  solver = "quadprog", verbose = TRUE, maxit = 80000L, conswitch = 1)
summary(fit)

```

aRD-class	<i>S4 Class "aRD"</i>
-----------	-----------------------

Description

S4 Class "aRD"

Slots

`call` An object of class "call".

`formula` An object of class "formula".

`coefficients` A numeric vector containing the estimated model parameters.

`iter` A positive integer indicating the number of iterations.

`converged` A logical constant that indicates whether the model has converged.

`y` A numeric vector containing the dependent variable of the model.

`x` The model matrix.

`data` A data frame containing the variables in the model.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenfeil

bootaRD	<i>Bootstrapping for Risk Differences estimated by aRD()</i>
---------	--

Description

`bootaRD()` applies nonparametric bootstrapping to an object of class "aRD" and computes bias-corrected accelerated confidence intervals (BCa) for the estimated risk differences.

Usage

```

bootaRD(object, ci_level = 0.95, R = 1000L, maxit = NULL, verbose = NULL,
  solver = NULL, eps_abs = NULL, eps_rel = NULL,
  polish_final = NULL, conswitch = NULL, use_nearPD = NULL)

```

Arguments

object	An object of the class "aRD".
ci_level	A value between 0 and 1 indicating the confidence interval. Provides bias-corrected accelerated bootstrap confidence intervals of the original estimated model parameters of aRD().
R	A positive integer greater than or equal to 1000 giving the number of bootstrap replicates.
maxit	A positive integer giving the maximum number of iterations in the aRD() algorithm. If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
verbose	Logical; if TRUE, a detailed output is printed, including iteration logs, objective values and solver messages (e.g., from osqp). If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
solver	A character string specifying the solver to use. Options are "osqp" or "quadprog". If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
eps_abs	Absolute tolerance for solver convergence (only used with "osqp"). If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
eps_rel	Relative tolerance for solver convergence (only used with "osqp"). If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
polish_final	Logical; if TRUE and solver is "osqp", performs a final polish step. If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
conswitch	Specifies how the constraint matrix is constructed: 1 (default) Generates all possible combinations of minimum and maximum values for the predictors (excluding the intercept), resulting in 2^{m-1} constraints. This formulation constrains model predictions within the observed data range, making it suitable for both risk factor identification and prediction (prognosis). 0 Uses the raw design matrix x as the constraint matrix, resulting in n constraints. This is primarily suitable for identifying risk factors, but not for prediction tasks, as predictions are not bounded to realistic ranges. If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().
use_nearPD	Logical; if TRUE, the Hessian matrix is projected to the nearest positive definite matrix using <code>Matrix::nearPD()</code> to ensure numerical stability, especially when using solvers that require a positive definite matrix (e.g., quadprog). If FALSE, the raw Hessian is used directly, which is faster but may lead to numerical issues if the matrix is not positive definite. Default is TRUE. If NULL (the default), the value stored in the aRD object is passed internally to bootaRD().

Value

An object of class "aRD_boot", which is a list containing:

Call_aRD The original call to the aRD() function used to fit the model.

- Successful_Bootstraps** The number of bootstrap replicates that were completed successfully.
- message** A character string with a status message indicating how many bootstrap samples succeeded.
- Coefficients** A matrix with the original estimated model parameters (Orig. Est.), the mean of the bootstrap estimates (Boot. Est.), the standard error of the bootstrap estimates (Boot. SE), the difference between the bootstrap mean and the original estimate (bias), the Risk Difference (equal to the estimate; RD), and the bias-corrected accelerated confidence intervals at the specified level.
- Bootstrap_Object** An object of class "boot" (from the **boot** package) containing the full bootstrap output, including replicates and metadata. This can be used for further analyses or plotting.

Author(s)

Julius Johannes Weise, Thomas Wolf, Stefan Wagenpfeil

Examples

```
set.seed(123)
x <- rnorm(100, 50, 10)
y <- rbinom(100, 1, exp(-4 + x * 0.04))
library(aRD)
fit <- aRD(formula = y ~ x, data = data.frame(y = y, x = x))
result <- bootaRD(fit, ci_level = 0.90)
print(result)
```

coef, aRD-method

Extracting the estimated model parameters of aRD()

Description

For objects of class "aRD", coef() extracts the estimated model parameters of aRD().

Usage

```
## S4 method for signature 'aRD'
coef(object)
```

Arguments

object An object of class "aRD".

Value

A numeric vector containing the estimated model parameters.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

confint,aRD-method	<i>Estimating confidence intervals of the estimated model parameters of aRD()</i>
--------------------	---

Description

For objects of class "aRD", confint() estimates confidence intervals of the estimated model parameters of aRD().

Usage

```
## S4 method for signature 'aRD'
confint(
  object,
  parm,
  level = 0.95,
  method = "wald",
  R = 1000L,
  maxit = NULL,
  verbose = NULL,
  solver = NULL,
  eps_abs = NULL,
  eps_rel = NULL,
  polish_final = NULL,
  conswitch = NULL,
  use_nearPD = NULL
)
```

Arguments

object	An object of class "aRD".
parm	A specification of which model parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all model parameters are considered.
level	A numeric value that indicates the level of confidence.
method	A character giving the estimation method of the confidence intervals ("bca" or "wald").
R	A positive integer giving the number of bootstrap replicates.
maxit	A positive integer giving the maximum number of iterations.
verbose	Logical; if TRUE, a detailed output is printed, including iteration logs, objective values and solver messages (e.g., from osqp).

solver	A character string specifying the solver to use. Options are "osqp" or "quadprog".
eps_abs	Absolute tolerance for solver convergence (only used with "osqp").
eps_rel	Relative tolerance for solver convergence (only used with "osqp").
polish_final	Logical; if TRUE and solver is "osqp", performs a final polish step.
conswitch	Specifies how the constraint matrix is constructed: 1 (default) Generates all possible combinations of minimum and maximum values for the predictors (excluding the intercept), resulting in 2^{m-1} constraints. This formulation constrains model predictions within the observed data range, making it suitable for both risk factor identification and prediction (prognosis). 0 Uses the raw design matrix x as the constraint matrix, resulting in n constraints. This is primarily suitable for identifying risk factors, but not for prediction tasks, as predictions are not bounded to realistic ranges.
use_nearPD	Logical; if TRUE, the Hessian matrix is projected to the nearest positive definite matrix using <code>Matrix::nearPD()</code> to ensure numerical stability, especially when using solvers that require a positive definite matrix (e.g., quadprog). If FALSE, the raw Hessian is used directly, which is faster but may lead to numerical issues if the matrix is not positive definite. Default is TRUE.

Details

confinf provides Wald (default) and bias-corrected accelerated bootstrap confidence intervals of the estimated model parameters of `aRD()`.

Value

A matrix with columns giving the lower and upper confidence limits of each estimated model parameter.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

constr *Setting the linear inequality constraints for aRD()*

Description

`constr()` sets the linear inequality constraints for `aRD()`.

Usage

```
constr(x, version = 1)
```

Arguments

x A model matrix.
version switch for constraints

Value

A matrix containing the linear inequality constraints for `aRD()`.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

gradF *Deriving the first derivatives of the log likelihood function of the linear-binomial model in aRD()*

Description

`gradF()` derives the first derivatives of the log likelihood function of the linear-binomial model.

Usage

```
gradF(theta, y, x)
```

Arguments

theta A numeric vector containing the initial values of the model parameters.
y A numeric vector containing the dependent variable of the model.
x The model matrix.

Value

A numeric vector containing the first derivatives of the log likelihood function of the linear-binomial model.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

hess	<i>Deriving the second partial derivatives of the log likelihood function of the linear-binomial model in aRD() (Hessian matrix)</i>
------	--

Description

aRD() derives the second partial derivatives of the log likelihood function of the linear-binomial model.

Usage

```
hess(theta, y, x)
```

Arguments

theta	A numeric vector containing the initial values of the model parameters.
y	A numeric vector containing the dependent variable of the model.
x	The model matrix.

Value

A numeric matrix containing the second partial derivatives of the log likelihood function of the linear-binomial model (Hessian matrix).

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

obj_value	<i>Log-Likelihood Function Value for the linear-binomial model in aRD()</i>
-----------	---

Description

obj_value() computes the value of the log-likelihood function for the linear-binomial model at a given parameter vector theta.

Usage

```
obj_value(theta, y, x)
```

Arguments

theta	A numeric vector containing the initial values of the model parameters.
y	A numeric vector containing the dependent variable of the model.
x	The model matrix.

Value

Numeric scalar representing the value of the log-likelihood function evaluated at theta.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

summary, aRD-method	<i>Summarizing the estimated model parameters of aRD()</i>
---------------------	--

Description

For objects of class "aRD", summary() summarizes the estimated model parameters of aRD().

Usage

```
## S4 method for signature 'aRD'
summary(object)
```

Arguments

object An object of class "aRD".

Value

A list containing the following elements:

coefficients	A numeric vector containing the estimated model parameters.
std.err	A numeric vector containing the estimated standard errors of the model parameters.
z.value	A numeric vector containing the estimated z test statistic of the model parameters.
p.value	A numeric vector containing the estimated p values of the model parameters.

Author(s)

Thomas Wolf, Julius Johannes Weise, Stefan Wagenpfeil

 variable_selection_aRD

Variable Selection (Forward or Backward) for linear-Binomial Models

Description

Performs forward or backward variable selection based on Wald test p-values for models estimated using `aRD()`. In each step, a new model is fitted using `aRD()`, and variables are added or removed based on the significance level defined by `alpha`.

Usage

```
variable_selection_aRD(model, selection = c("backward", "forward"),
                      alpha = 0.157, print_models = FALSE, maxit = NULL,
                      verbose = NULL, solver = NULL, eps_abs = NULL,
                      eps_rel = NULL, polish_final = NULL, conswitch = NULL,
                      use_nearPD = NULL)
```

Arguments

<code>model</code>	A model object from <code>aRD()</code> with full data and formula.
<code>selection</code>	Character string, either "backward" or "forward". Determines the direction of model selection. If not specified, backward elimination is performed by default.
<code>alpha</code>	P-value threshold for variable inclusion (forward) or exclusion (backward). Defaults to 0.157, as recommended by Heinze, G., Wallisch, C., & Dunkler, D. (2018).
<code>print_models</code>	Logical; whether to print each model during selection. Defaults to FALSE.
<code>maxit</code>	Maximum number of iterations in the <code>aRD()</code> algorithm. If NULL, defaults to 200L or value from original model call.
<code>verbose</code>	Logical; if TRUE, shows detailed solver output. If NULL, taken from original model or defaults to FALSE.
<code>solver</code>	Solver to be used in <code>aRD()</code> , default is "osqp" unless specified in the original model.
<code>eps_abs</code>	Numeric. Absolute tolerance used as a convergence criterion for the solver. If not specified, taken from original model or defaults to $1e-5$.
<code>eps_rel</code>	Numeric. Relative tolerance used as a convergence criterion for the solver. If not specified, taken from original model or defaults to $1e-5$.
<code>polish_final</code>	Logical; if TRUE and solver is "osqp", performs a final polish step.
<code>conswitch</code>	Specifies how the constraint matrix is constructed: 1 (default) Generates all possible combinations of minimum and maximum values for the predictors (excluding the intercept), resulting in 2^{m-1} constraints. This formulation constrains model predictions within the observed data range, making it suitable for both risk factor identification and prediction (prognosis).

- 0** Uses the raw design matrix x as the constraint matrix, resulting in n constraints. This is primarily suitable for identifying risk factors, but not for prediction tasks, as predictions are not bounded to realistic ranges.
- use_nearPD** Logical; if TRUE, the Hessian matrix is projected to the nearest positive definite matrix using `Matrix::nearPD()` to ensure numerical stability, especially when using solvers that require a positive definite matrix (e.g., `quadprog`). If FALSE, the raw Hessian is used directly, which is faster but may lead to numerical issues if the matrix is not positive definite. Default is TRUE.

Value

An object of class "aRD_selection", which is a list containing:

- final_model** An object of class aRD representing the final model selected through the variable selection process.
- model_list** A list of intermediate aRD model objects fitted during each step of the selection.
- skipped_models** A named list of models that failed to converge and were skipped during the selection. Each entry includes the attempted formula.
- final_formula** The final model formula used in the last step.
- EPV** Estimated events-per-variable (EPV) of the final model, used as a diagnostic for model stability.
- warnings** Optional warning messages about convergence issues or model stability (e.g., low EPV or skipped variables).

Author(s)

Julius Johannes Weise, Thomas Wolf, Stefan Wagenpfeil

References

Heinze, G., Wallisch, C., & Dunkler, D. (2018). Variable selection – A review and recommendations for the practicing statistician. *Biometrical Journal*, 60(3), 431–449.

Examples

```
set.seed(123)
x1 <- rnorm(500, 50, 10)
x2 <- rnorm(500, 30, 5)
x3 <- rnorm(500, 40, 8)
x4 <- rnorm(500, 60, 12)
logit <- (-4 + x1 * 0.04 + x3 * 0.04)
p <- 1 / (1 + exp(-logit))
y <- rbinom(500, 1, p)
df <- data.frame(y, x1, x2, x3, x4)
fit <- aRD(formula = y ~ x1 + x2 + x3 + x4, data = df)
result <- variable_selection_aRD(fit, selection = "forward", alpha = 0.1)
print(result)
```

Index

[aRD](#), [2](#)

[aRD-class](#), [4](#)

[bootaRD](#), [4](#)

[coef](#), [aRD-method](#), [6](#)

[confint](#), [aRD-method](#), [7](#)

[constr](#), [8](#)

[gradF](#), [9](#)

[hess](#), [10](#)

[obj_value](#), [10](#)

[summary](#), [aRD-method](#), [11](#)

[variable_selection_aRD](#), [12](#)