Package 'AirScreen'

July 31, 2025

Type Package

Title Feature Screening via Adaptive Iterative Ridge (Air-HOLP and Air-OLS)

Version 0.1.0

Description Implements two complementary high-dimensional feature screening methods, Adaptive Iterative Ridge High-dimensional Ordinary Least-squares Projection (Air-HOLP, suitable when the number of predictors p is greater than or equal to the sample size n) and Adaptive Iterative Ridge Ordinary Least Squares (Air-OLS, for n greater than p). Also provides helper functions to generate compound-symmetry and AR(1) correlated data, plus a unified Air() front end and a summary method. For methodological details see Joudah, Muller and Zhu (2025) <doi:10.1007/s11222-025-10599-6>.

License MIT + file LICENSE

Encoding UTF-8

Imports stats

RoxygenNote 7.3.2

URL https://github.com/Logic314/Air-HOLP

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Ibrahim Joudah [aut, cre] (ORCID: <https://orcid.org/0000-0002-0397-6266>), Samuel Muller [aut] (ORCID: <https://orcid.org/0000-0002-3087-8127>), Houying Zhu [aut] (ORCID: <https://orcid.org/0000-0002-2515-7413>)

Maintainer Ibrahim Joudah <ibrahim.joudah@mq.edu.au>

Repository CRAN

Date/Publication 2025-07-31 10:00:43 UTC

Contents

`Air` *Unified Interface for* `AirHOLP` *and* `AirOLS`

Description

`Air` is a high-level wrapper that applies either the `AirHOLP` or `AirOLS` methods based on data dimensions. It returns an `AirResult` object ready for inspection with `summary`.

Usage

```
Air(
  X,
  y,
  m = NULL,
  screening_threshold = NULL,
  penalty = 10,
  penalty_type = c("adaptive", "fixed", "both"),
  method = c("auto", "AirHOLP", "AirOLS")
)
```

Arguments

`X`	Numeric predictor matrix ($n \times p$).
`y`	Numeric response vector of length n.
`m`	Integer specifying the number of coefficients retained at each adaptive-penalty iteration (default $n/\log(n)$ capped at $p-1$).
`screening_threshold`	Integer specifying the number of screened features to display in summary.
`penalty`	Numeric scalar or vector with ridge penalty value(s) (default 10).
`penalty_type`	One of `"adaptive"`, `"fixed"`, or `"both"` (default `"adaptive"`).
`method`	`"auto"`, `"AirHOLP"`, or `"AirOLS"` (default `"auto"`).

Details

- When `method = "auto"` (default), `AirHOLP` is used if $p \geq n$; otherwise `AirOLS` is chosen.
- `penalty_type` chooses whether the ridge penalty is selected adaptively, fixed at the supplied value(s), or both (returning two sets of ranks).
- Air checks the validity of inputs and substitute by defaults when invalid.

Value

An object of class `"AirResult"`, a named list that may contain the following elements (depending on `penalty_type`):

`order`, `order_adaptive`, `order_fixed` Integer vector of feature indices sorted by absolute Air-HOLP or Air-OLS score, from largest to smallest.

`rank`, `rank_adaptive`, `rank_fixed` Integer vector of feature ranks matching `order`.

`Beta`, `Beta_adaptive`, `Beta_fixed` Numeric vector of Air-HOLP or Air-OLS coefficient estimates.

`penalty`, `penalty_adaptive`, `penalty_fixed` Final ridge penalty value(s) used.

The helper `summary.AirResult` prints a concise summary.

References

Joudah, I., Muller, S., and Zhu, H. (2025). "Air-HOLP: Adaptive Regularized Feature Screening for High-Dimensional Data." *Statistics and Computing*. doi:10.1007/s11222025105996

Examples

```
## simple example (p > n  ->  AirHOLP)
set.seed(314)
X <- matrix(rnorm(100000), nrow = 200, ncol = 500)
y <- X[, 1] + X[, 2] + X[, 3] + X[, 4] + 3*rnorm(200)
result <- Air(X, y, penalty_type = "both")
summary(result)

## multiple fixed penalty values
result2 <- Air(X, y, penalty_type = "fixed", penalty = c(1, 100))
summary(result2)
```

`AirHOLP` *Adaptive Iterative Ridge HOLP Screening*

Description

This function ranks features with the Adaptive Iterative Ridge High-dimensional Ordinary Least-squares Projection (Air-HOLP) method of Joudah *et al.* (2025) and returns both the per-feature ranks and the ordered feature indices. AirHOLP is intended for the high-dimensional case $p \geq n$. When $n > p$, use `AirOLS` instead.

Usage

```
AirHOLP(
  X,
  y,
  Threshold = min(ncol(X) - 1, ceiling(nrow(X)/log(nrow(X)))),
  r0 = 10,
  adapt = TRUE,
  iter = 10,
  Lambda,
  Un,
  XUn
)
```

Arguments

X	Numeric predictor matrix of dimension $n \times p$.
y	Numeric response vector of length n.
Threshold	Integer specifying the number of coefficients retained at each adaptive-penalty iteration (default $n/\log(n)$ capped at $p-1$).
r0	Numeric initial ridge penalty (default 10).
adapt	Logical; set to TRUE (default) to enable adaptive penalty selection.
iter	Integer; maximum number of iterations for adaptive-penalty selection (default 10).
Lambda	Eigenvalues of XX^T, if missing the function will compute it.
Un	Eigenvectors of XX^T, if missing the function will compute it.
XUn	X transpose times Un, if missing the function will compute it.

Details

The Threshold parameter controls how many coefficients are kept at each iteration of the adaptive-penalty procedure. The default value $\lceil n/\log(n) \rceil$ performs well in most settings; changing it can reduce stability, so we recommend keeping the default unless you have a specific reason to adjust it. The parameters Lambda, Un, and XUn are helpful to run AirHOLP on 2 or more different y vectors for the same X (to avoid repeated heavy computations).

Value

An object of class AirResult containing

order_r Integer vector of feature indices sorted by absolute Air-HOLP score, from largest to smallest.

index_r Integer vector of feature ranks matching order_r.

Beta_r Numeric vector of Air-HOLP coefficient estimates.

r Final ridge-penalty value used.

iter_last Number of iterations performed for adaptive penalty selection.

References

Joudah, I., Muller, S., and Zhu, H. (2025). "Air-HOLP: Adaptive Regularized Feature Screening for High-Dimensional Data." *Statistics and Computing*. doi:10.1007/s11222025105996

Examples

```
# Example 1 (default parameters)
set.seed(314)
X <- matrix(rnorm(10000), nrow = 50, ncol = 200)
y <- X[, 1] + X[, 10] + rnorm(50)
result <- AirHOLP(X, y)
str(result)
result$order_r[1:7] # the top 7 features
result$index_r[c(1, 10),] # ranks of the true features (x1, and x10)

# Example 2 (multiple responses, same X)
set.seed(314)
X <- matrix(rnorm(2000000), nrow = 1000, ncol = 2000)
y1 <- X[, 1] + X[, 2] + 6*rnorm(1000)
y2 <- X[, 1] - X[, 2] + 12*rnorm(1000)
y3 <- X[, 1] + X[, 2] - X[, 3] + 3*rnorm(1000)
y4 <- X[, 1] - X[, 2] + X[, 3] + 9*rnorm(1000)
XXT <- tcrossprod(X)
eXXT <- eigen(XXT)
Lambda <- eXXT$values
Un <- eXXT$vectors
XUn <- crossprod(X,Un)
result1 <- AirHOLP(X, y1, Lambda = Lambda, Un = Un, XUn = XUn)
result1$order_r[1:7] # the top 7 features
result1$index_r[1:2,] # ranks of the true features (x1 and x2)
result2 <- AirHOLP(X, y2, Lambda = Lambda, Un = Un, XUn = XUn)
result2$order_r[1:7] # the top 7 features
result2$index_r[1:2,] # ranks of the true features (x1 and x2)
result3 <- AirHOLP(X, y3, Lambda = Lambda, Un = Un, XUn = XUn)
result3$order_r[1:7] # the top 7 features
result3$index_r[1:3,] # ranks of the true features (x1, x2, and x3)
result4 <- AirHOLP(X, y4, Lambda = Lambda, Un = Un, XUn = XUn)
result4$order_r[1:7] # the top 7 features
result4$index_r[1:3,] # ranks of the true features (x1, x2, and x3)

# Example 3 (multiple fixed penalties)
set.seed(314)
X <- matrix(rnorm(10000), nrow = 100, ncol = 200)
y <- X[, 1] - X[, 2] + X[, 3] + 2*rnorm(100)
result <- AirHOLP(X, y, r0 = c(1, 100, 10000), adapt = FALSE)
str(result)
result$order_r0[1:7,] # the top 7 features (for each penalty)
result$index_r0[1:3,] # ranks of the true features (x1, x2, and x3)
```

AirOLS *Adaptive Iterative Ridge OLS Screening*

Description

This function ranks features with the Adaptive Iterative Ridge Ordinary Least Squares (Air-OLS) method of Joudah *et al.* (2025) and returns both the per-feature ranks and the ordered feature indices. AirHOLP is intended for the high-dimensional case $n \geq p$. When $p > n$, use `AirHOLP` instead.

Usage

```
AirOLS(
  X,
  y,
  Threshold = min(ncol(X) - 1, ceiling(nrow(X)/log(nrow(X)))),
  r0 = 10,
  adapt = TRUE,
  iter = 10,
  Lambda,
  Up,
  XUp
)
```

Arguments

`X`	Numeric predictor matrix of dimension $n \times p$.
`y`	Numeric response vector of length n.
`Threshold`	Integer specifying the number of coefficients retained at each adaptive-penalty iteration (default $n/\log(n)$ capped at $p-1$).
`r0`	Numeric initial ridge penalty (default 10).
`adapt`	Logical; set to `TRUE` (default) to enable adaptive penalty selection.
`iter`	Integer; maximum number of iterations for adaptive-penalty selection (default 10).
`Lambda`	Eigenvalues of $X^T X$, if missing the function will compute it.
`Up`	Eigenvectors of $X^T X$, if missing the function will compute it.
`XUp`	`X` times `Up`, if missing the function will compute it.

Details

The `Threshold` parameter controls how many coefficients are kept at each iteration of the adaptive-penalty procedure. The default value $\lceil n/\log(n) \rceil$ performs well in most settings; changing it can reduce stability, so we recommend keeping the default unless you have a specific reason to adjust it. The parameters Lambda, `Up`, and `XUp` are helpful to run `AirOLS` on 2 or more different `y` vectors for the same `X` (to avoid repeated heavy computations).

Value

An object of class `AirResult` containing

`order_r` Integer vector of feature indices sorted by absolute Air-OLS score, from largest to smallest.

`index_r` Integer vector of feature ranks matching `order_r`.

`Beta_r` Numeric vector of Air-OLS coefficient estimates.

`r` Final ridge-penalty value used.

`iter_last` Number of iterations performed for adaptive penalty selection.

References

Joudah, I., Muller, S., and Zhu, H. (2025). "Air-HOLP: Adaptive Regularized Feature Screening for High-Dimensional Data." *Statistics and Computing*. doi:10.1007/s11222025105996

Examples

```
# Example 1 (default parameters)
set.seed(314)
X <- matrix(rnorm(10000), nrow = 200, ncol = 50)
y <- X[, 1] + X[, 10] + 2*rnorm(200)
result <- AirOLS(X, y)
str(result)
result$order_r[1:7] # the top 7 features
result$index_r[c(1, 10),] # ranks of the true features (x1, and x10)

# Example 2 (multiple responses, same X)
set.seed(314)
X <- matrix(rnorm(2000000), nrow = 2000, ncol = 1000)
y1 <- X[, 1] + X[, 2] + 6*rnorm(2000)
y2 <- X[, 1] - X[, 2] + 12*rnorm(2000)
y3 <- X[, 1] + X[, 2] - X[, 3] + 5*rnorm(2000)
y4 <- X[, 1] - X[, 2] + X[, 3] + 10*rnorm(2000)
XTX <- crossprod(X)
eXTX <- eigen(XTX)
Lambda <- eXTX$values
Up <- eXTX$vectors
XUp <- X%*%Up
result1 <- AirOLS(X, y1, Lambda = Lambda, Up = Up, XUp = XUp)
result1$order_r[1:7] # the top 7 features
result1$index_r[1:2,] # ranks of the true features (x1 and x2)
result2 <- AirOLS(X, y2, Lambda = Lambda, Up = Up, XUp = XUp)
result2$order_r[1:7] # the top 7 features
result2$index_r[1:2,] # ranks of the true features (x1 and x2)
result3 <- AirOLS(X, y3, Lambda = Lambda, Up = Up, XUp = XUp)
result3$order_r[1:7] # the top 7 features
result3$index_r[1:3,] # ranks of the true features (x1, x2, and x3)
result4 <- AirOLS(X, y4, Lambda = Lambda, Up = Up, XUp = XUp)
result4$order_r[1:7] # the top 7 features
result4$index_r[1:3,] # ranks of the true features (x1, x2, and x3)
```

```
# Example 3 (multiple fixed penalties)
set.seed(314)
X <- matrix(rnorm(10000), nrow = 200, ncol = 100)
y <- X[, 1] - X[, 2] + X[, 3] + 3*rnorm(200)
result <- AirOLS(X, y, r0 = c(1, 100, 10000), adapt = FALSE)
str(result)
result$order_r0[1:7,] # the top 7 features for each penalty
result$index_r0[1:3,] # ranks of the true features (x1, x2, and x3)
```

`newton` *Newton-Raphson root finder*

Description

Newton-Raphson root finder

Usage

```
newton(f, fp, x, tol = 0.001, m = 100)
```

Arguments

`f`	Function whose root is sought.
`fp`	Derivative function of `f`.
`x`	Numeric starting value.
`tol`	Convergence tolerance (default `1e-3`).
`m`	Maximum number of iterations (default 100).

Details

Iterates $x_{new} = x - f(x)/f'(x)$ until the change is below `tol` or `m` iterations are reached (then issues a warning).

Value

The estimated root.

Examples

```
# Solve x^2 - 2 = 0
newton(function(x) x^2 - 2, function(x) 2*x, 1)
```

rnormAR1 *Generate normal samples (Autoregressive AR(1) correlation)*

Description

rnormAR1 efficiently generates samples from a multivariate normal distribution with AR(1) correlation $cor(x_i, x_j) = \rho^{|i-j|}$.

Usage

```
rnormAR1(n, p, rho = 0.5, means = 0, variances = 1)
```

Arguments

n	Number of observations.
p	Number of variables.
rho	Autoregressive correlation coefficient.
means	Numeric vector of feature means (length 1 or p).
variances	Numeric vector of feature variances (length 1 or p).

Value

A numeric $n \times p$ matrix with the specified correlation, means, and variances.

Examples

```
X1 <- rnormAR1(10, 5)
X2 <- rnormAR1(10, 5, rho = 0.3, means = 2, variances = 4)
X3 <- rnormAR1(10, 5, rho = 0.4, means = 1:5, variances = 3:7)
```

rnormCS *Generate normal samples (Compound Symmetry)*

Description

rnormCS efficiently generates samples from a multivariate normal distribution with compound symmetry correlation structure (all features equally correlated).

Usage

```
rnormCS(n, p, rho = 0.5, means = 0, variances = 1)
```

Arguments

n	Number of observations.
p	Number of features.
rho	Common correlation coefficient.
means	Numeric vector of feature means (length 1 or p).
variances	Numeric vector of feature variances (length 1 or p).

Value

A numeric $n \times p$ matrix with the specified correlation, means, and variances.

Examples

```
X1 <- rnormCS(10, 5)
X2 <- rnormCS(10, 5, rho = 0.3, means = 2, variances = 4)
X3 <- rnormCS(10, 5, rho = 0.4, means = 1:5, variances = 3:7)
```

---

summary.AirResult *Summarise an* AirResult *Object*

Description

Produces a compact, human-readable summary of the ranking results returned by Air.

Usage

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

Arguments

object	An object of class "AirResult".
...	Additional arguments (ignored; included for S3 compatibility).

Value

Invisibly returns object. The function is invoked for its printing side-effect.

Examples

```
set.seed(314)
X <- matrix(rnorm(500), 50, 100)
y <- X[, 1] - 2*X[, 3] + rnorm(50)
res <- Air(X, y, penalty_type = "both")
summary(res)
```

Index