Package ‘capr’

January 24, 2026

Title Covariate Assisted Principal Regression

Version 0.2.0

Author Xi Luo [aut, cre],
Yi Zhao [aut],
Brian Caffo [aut]

Maintainer Xi Luo <xi.rossi.luo@gmail.com>

Description Covariate Assisted Principal Regression (CAPR) for multiple covariance-matrix outcomes. The method identifies (principal) projection directions that maximize the log-likelihood of a log-linear regression model of the covariates. See Zhao et al. (2021), ``Covariate Assisted Principal Regression for Covariance Matrix Outcomes'' <doi:10.1093/biostatistics/kxz057>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, MASS

Suggests testthat, roxygen2

SystemRequirements C++17

URL https://github.com/rluo/capr

BugReports https://github.com/rluo/capr/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-01-24 10:50:02 UTC

Contents

---

capr — *Covariate Assisted Principal (CAP) Regression*

Description

Fits CAP components sequentially for principal direction vectors $\gamma^{(k)}$ and regression coefficients $\beta^{(k)}$, $k = 1, \ldots, K$. Each component is estimated via a flip-flop algorithm with optional orthogonalization of successive directions.

Usage

```
capr(
  S,
  X,
  K,
  B.init = NULL,
  Gamma.init = NULL,
  weight = NULL,
  max_iter = 200L,
  tol = 1e-06,
  orth = TRUE,
  n.init = 10L
)
```

Arguments

`S`	Numeric 3D array of size $p \times p \times n$ (for example, a stack of covariance matrices).
`X`	Numeric matrix $n \times q$ (design matrix), created for example by `model.matrix()`.
`K`	Integer scalar; number of components ($K \geq 1$).
`B.init`	Initial value of the coefficient array $B \in \mathbb{R}^{q \times n.init \times K}$ (default: zero 3D array).
`Gamma.init`	Initial value of the principal direction array $\Gamma \in \mathbb{R}^{p \times n.init \times K}$ (default: random Gaussian 3D array).
`weight`	Numeric vector of length n (default `rep(1, n)`); each element should be proportional to the sample size for the corresponding slice $S[,,i]$.
`max_iter`	Integer scalar; maximum flip-flop iterations per component (default 200).
`tol`	Positive numeric scalar; convergence tolerance (default 1e-6).

`orth` Logical scalar; if `TRUE` (default), enforce orthogonality of successive $\gamma^{(k)}$. If `FALSE`, no orthogonalization is performed (which may yield identical components).

`n.init` Integer scalar; number of random initializations (default 10). If `B.init` and `Gamma.init` are both supplied, `n.init` is ignored.

Details

For component k, CAP solves

$$\min_{\beta^{(k)},\gamma^{(k)}} \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^\top \beta^{(k)})T_i + \frac{1}{2}\sum_{i=1}^{n}\gamma^{(k)\top} S_i \gamma^{(k)} \exp(-\mathbf{x}_i^\top \beta^{(k)})$$

subject to

$$\gamma^{(k)\top} H \gamma^{(k)} = 1$$

and, for $k > 1$,

$$\Gamma^{(k-1)\top}\gamma^{(k)} = \mathbf{0}.$$

Here T_i denotes the weight for slice i, S_i is the i-th covariance slice, and H is the positive definite matrix used for the orthogonality constraint (see Zhao et al., 2021). The algorithm fits $\gamma^{(k)}$ and $\beta^{(k)}$ sequentially with multiple random initializations and returns the solution pair that minimizes the negative log-likelihood.

Value

A list of class `capr` with:

`B`	numeric matrix $q \times K$ whose k-th column stores $\beta^{(k)}$
`Gamma`	numeric matrix $p \times K$ whose k-th column stores $\gamma^{(k)}$
`loglike`	negative log-likelihood, up to constant scaling and shift
`S`	3D array used for fitting
`X`	design matrix used for fitting
`weight`	weight values used for fitting

References

Zhao, Y., Wang, B., Mostofsky, S. H., Caffo, B. S., & Luo, X. (2021). "Covariate assisted principal regression for covariance matrix outcomes." *Biostatistics*, 22(3), 629-645.

Examples

```
simu.data <- simu.capr(seed = 123L, n = 120L)
K <- 2L
fit <- capr(
    S = simu.data$S,
    X = simu.data$X,
    K = K
)
print(fit)
```

capr.boot *Bootstrap confidence intervals for CAP coefficients*

Description

Generates bootstrap inference for the CAP regression coefficients while holding the fitted directions Γ fixed. Each replicate samples the covariance slices $S[,,i]$ with replacement, projects them onto the fixed directions to obtain component-specific variances, and re-solves the $\beta^{(k)}$ equations. Quantile-based confidence intervals are returned for every predictor/component pair.

Usage

```
capr.boot(
  fit,
  nboot = 1000L,
  level = 0.95,
  max_iter = 100L,
  tol = 1e-06,
  seed = NULL
)
```

Arguments

`fit`	A `capr` fit containing components `B` and `Gamma`.
`nboot`	Integer; number of bootstrap replicates.
`level`	Confidence level for the returned intervals.
`max_iter`	Maximum Newton iterations for solving β.
`tol`	Convergence tolerance for the Newton solver.
`seed`	Optional integer seed for reproducibility.

Value

A list of class `capr.boot` with:

`beta`	bootstrap average of β with dimension $q \times K$
`ci_lower`, `ci_upper`	Matrices $q \times K$ with the lower/upper confidence limits.
`level`	The requested confidence level.

Examples

```
simu.data <- simu.capr(seed = 123L, n = 120L)
K <- 3L
fit <- capr(
    S = simu.data$S,
    X = simu.data$X,
```

```
    K = K
)
capr.boot(fit, nboot = 10L, level = 0.95, seed = 42L)
```

---

cosine_similarity *Cosine similarity between numeric vectors*

Description

Computes the cosine of the angle between two numeric vectors. Both vectors must have equal length and non-zero Euclidean norms.

Usage

```
cosine_similarity(a, b, eps = 1e-12)
```

Arguments

a, b	Numeric vectors of equal length.
eps	Non-negative numeric tolerance used to guard against division by zero. Defaults to 1e-12.

Value

A scalar double value in [-1, 1] representing the cosine similarity between a and b.

Examples

```
cosine_similarity(c(1, 2, 3), c(1, 2, 3))
cosine_similarity(c(1, 0), c(0, 1))
cosine_similarity(c(1, 2), c(-1, -2))
```

FG *Flury-Gautschi Common Principal Components*

Description

Implements the Flury & Gautschi (1986) (FG) iterative algorithm and a variant to estimate a common loading matrix across multiple covariance matrices. Each iteration cycles over all ordered pairs of variable indices and updates a (2 x 2) rotation so that the transformed matrices share diagonal structure.

Usage

```
FG(cov_array, p = NULL, m = NULL, maxit = 30L)

FG2(cov_array, p = NULL, m = NULL, maxit = 30L)
```

Arguments

`cov_array`	Numeric 3D array of shape $pxpxm$ containing covariance matrices in its m slices.
`p`	Optional integer specifying the matrix dimension; defaults to `dim(cov_array)[1]`.
`m`	Optional integer specifying the number of matrices/slices; defaults to `dim(cov_array)[3]`.
`maxit`	Integer scalar; number of outer iterations of the algorithm.

Details

Two solvers are exported:

`FG()` The original FG algorithm.

`FG2()` An alternative algorithm by Eslami et al. (2013).

Value

A pxp numeric matrix of estimated common loadings.

References

Flury, B. N. (1984). "Common Principal Components in k Groups." *Journal of the American Statistical Association*, 79, 892-898.

Flury, B. N., & Gautschi, W. (1986). "An Algorithm for Simultaneous Orthogonal Transformation of Several Positive Definite Symmetric Matrices to Nearly Diagonal Form." *SIAM Journal on Scientific and Statistical Computing*, 7(1), 169-184.

Eslami, A., Qannari, E. M., Kohler, A., & Bougeard, S. (2013). "General Overview of Methods of Analysis of Multi-Group Datasets." *Revue des Nouvelles Technologies de l'Information*, 25, 108-123.

Examples

```
set.seed(1)
p <- 3
m <- 4
mats <- replicate(m,
    {
        A <- matrix(rnorm(p * p), p, p)
        crossprod(A)
    },
    simplify = FALSE
)
cov_cube <- array(NA_real_, dim = c(p, p, m))
for (k in 1:m) cov_cube[, , k] <- mats[[k]]
FG(cov_cube, maxit = 5)
FG2(cov_cube, maxit = 5)
```

`log_deviation_from_diagonality` *Log deviation from diagonality*

Description

Evaluates the Flury-Gautschi log-deviation criterion for a collection of covariance matrices transformed by a loading matrix.

Usage

```
log_deviation_from_diagonality(S_cube, nval, B)
```

Arguments

S_cube	Numeric 3D array of shape `p x p x n` containing covariance matrices in its slices.
nval	Numeric vector of length `n` giving weights for each matrix.
B	Numeric `p x p` orthonormal matrix applied to the covariance slices.

Value

Numeric scalar value equal to $\sum_i n_i(\log\det\operatorname{diag}(B^\top S_i B) - \log\det(B^\top S_i B))/(\sum_i n_i)$.

Examples

```
covs <- array(diag(2), dim = c(2, 2, 1))
log_deviation_from_diagonality(covs, 1, diag(2))
```

`plot.capr` *Plot deviation diagnostics by component count*

Description

For a fitted CAP regression, plots two diagnostics across the first K components: (1) the negative log-likelihood returned by `capr()` and (2) the log deviation-from-diagonality (DfD) for the loading matrix formed by the first k directions. Both curves help assess the gain from adding components.

Usage

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

Arguments

`x`	A `capr` object returned by `capr()`.
`...`	Additional arguments passed to `graphics::plot()` and applied to both panels (for example, `pch`, `col`, or axis limits).

Details

The DfD criterion for the first k directions $\Gamma^{(k)}$ is

$$\mathrm{DfD}(\Gamma^{(k)}) = \left(\prod_{i=1}^{n} \nu\left(\Gamma^{(k)\top} S_i \Gamma^{(k)} / T_i\right)^{T_i} \right)^{1/\sum_i T_i},$$

where

$$\nu(A) = \frac{\det\{\mathrm{diag}(A)\}}{\det(A)}$$

for a positive definite matrix A. The curve shows $\log \mathrm{DfD}(\Gamma^{(k)})$. A common choice for k is the last point before a sudden jump in the negative log-likelihood or log-DfD curve.

Value

Invisibly returns the numeric vector of log deviation values (one per component).

See Also

`log_deviation_from_diagonality()`

Examples

```
sim <- simu.capr(seed = 123L, n = 120L)
fit <- capr(S = sim$S, X = sim$X, K = 3L)
plot(fit)
```

---

print.capr *Print method for CAP regression fits*

Description

Formats the coefficient matrix $\hat{B}$ returned by `capr()` in a linear-regression style table, showing the estimate for each predictor and component.

Usage

```
## S3 method for class 'capr'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	An object of class `capr`, typically the result of `capr()`.
`digits`	Number of significant digits to show when printing numeric values.
`...`	Additional arguments passed on to `print.data.frame()`.

Value

The input object `x`, invisibly.

Examples

```
simu.data <- simu.capr(seed = 123L, n = 120L)
K <- 2L
fit <- capr(
    S = simu.data$S,
    X = simu.data$X,
    K = K
)
print(fit)
```

`print.capr.boot` *Print method for* `capr.boot` *objects*

Description

Displays bootstrap coefficient estimates and their confidence intervals component by component as compact tables.

Usage

```
## S3 method for class 'capr.boot'
print(x, digits = max(4L, getOption("digits") - 4L), ...)
```

Arguments

`x`	An object of class `capr.boot`, typically produced by `capr.boot()`.
`digits`	Number of significant digits to show when printing numeric values.
`...`	Additional arguments passed on to `print.data.frame()`.

Value

The input object `x`, invisibly.

Examples

```
simu.data <- simu.capr(seed = 123L, n = 120L)
K <- 2L
fit <- capr(
    S = simu.data$S,
    X = simu.data$X,
    K = K
)
fit.boot <- capr.boot(
    fit = fit,
    nboot = 10L,
    max_iter = 20L,
    tol = 1e-6,
    seed = 123L
)
print(fit.boot)
```

`simu.capr` *Simulate covariance matrices compatible with* `capr()`

Description

Generates a simple synthetic dataset for CAP regression consisting of a covariance cube, design matrix, and the latent orthogonal directions used to build the covariance slices.

Usage

```
simu.capr(seed = 123L, n = 120L)
```

Arguments

`seed`	Integer seed used for reproducibility.
`n`	Number of observations (slices) to generate.

Value

A list with components:

`S`	Array of dimension $p \times p \times n$ holding the simulated covariance matrices.
`X`	Design matrix of size $n \times 2$ with an intercept and a Bernoulli covariate.
`Q`	Orthogonal matrix whose columns are the latent directions.
`BetaMat`	True coefficient matrix used to form the eigenvalues.
`H`	Average covariance matrix $\frac{1}{n}\sum_i S_i$.
`p, n`	The dimension and sample size supplied to the generator.

Examples

```
sim <- simu.capr(seed = 10, n = 50)
str(sim$S)
```

Index