

Package ‘CepReg’

December 17, 2025

Title A Cepstral Model for Covariate-Dependent Time Series

Version 0.1.3

Description Modeling associations between covariates and power spectra of replicated time series using a cepstral-based semiparametric framework. Implements a fast two-stage estimation procedure via Whittle likelihood and multivariate regression. The methodology is based on Li and Dong (2025) <[doi:10.1080/10618600.2025.2473936](https://doi.org/10.1080/10618600.2025.2473936)>.

Imports MASS, Renvlp, psych

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Author Qi Xia [aut, cre],
Zeda Li [aut, ctb]

Maintainer Qi Xia <xiaqi1010@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2025-12-17 22:00:02 UTC

Contents

boot_effect	2
CepReg	4
cep_get	6
data_generator	7
effect_get	7
env_get	8
generate_sig	9
ols_get	10
perd_get	11
psi_get	11
rrr_get	12
spec_regress	13

Index	15
--------------	-----------

boot_effect

*Bootstrap Confidence Intervals for Functional Effect Curves***Description**

Computes bias-corrected percentile bootstrap confidence intervals for the intercept and covariate effect functions in cepstral-based regression models.

Usage

```
boot_effect(
  logspect,
  res,
  alpha_effect,
  beta_effect,
  X,
  nbase,
  frq1,
  frq2,
  nrank,
  ind,
  level,
  nboot,
  method = "rrr_get",
  verb = FALSE
)
```

Arguments

logspect	Matrix of estimated log-spectra.
res	Matrix of residuals from cepstral regression.
alpha_effect	Vector of estimated intercept effect function.
beta_effect	Matrix of estimated covariate effect functions.
X	Covariate matrix.
nbase	Number of cepstral basis functions.
frq1	Frequency grid used for cepstral modeling.
frq2	Frequency grid used for reconstructing spectra.
nrank	Rank for reduced-rank regression.
ind	A vector of indices indicating which covariates to compute effect confidence intervals for.
level	Confidence level.
nboot	Number of bootstrap iterations.
method	Regression method: "rrr", "ols", or "env".
verb	Logical; if TRUE, prints bootstrap iteration number.

Value

A list with:

alpha_ci Matrix with lower and upper CI for intercept effect.

beta_ci Array or matrix of lower and upper CI for covariate effects.

Examples

```
set.seed(123)
N <- 10
len <- 12
nbase <- 2
nrank <- 1
nboot <- 10
level <- 0.95
p <- 2
ind <- 1:p

Y <- matrix(rnorm(N * nbase), nrow = N, ncol = nbase)
X <- matrix(rnorm(N * p), nrow = N, ncol = p)
frq <- seq(1, nbase) / len

rrr_out <- rrr_get(X, Y, frq, nbase, nrank)

res_matrix <- rrr_out$res
if (ncol(res_matrix) != length(frq)) {
  res_matrix <- matrix(
    res_matrix[, 1:length(frq)],
    nrow = N,
    ncol = length(frq))}

eff <- effect_get(rrr_out$alph, rrr_out$bet, frq, nbase, ind)
alpha_eff <- eff$alpha_effect
beta_eff <- eff$beta_effect

logspect <- matrix(rnorm(N * length(frq)), nrow = N, ncol = length(frq))

boot_ci <- boot_effect(
  logspect = logspect,
  res = res_matrix,
  alpha_effect = alpha_eff,
  beta_effect = beta_eff,
  X = X,
  nbase = nbase,
  frq1 = frq,
  frq2 = frq,
  nrank = nrank,
  ind = ind,
  level = level,
  nboot = nboot,
  method = "rrr_get",
  verb = TRUE)
```

```

)

plot(frq, beta_eff[, 1], type = "l", col = "blue", lwd = 2,
     ylab = "Effect", xlab = "Frequency",
     main = paste("Effect Function and", level*100, "% CI for Covariate", ind[1]))
lines(frq, boot_ci[[ind[1]]][, 1], col = "red", lty = 2)
lines(frq, boot_ci[[ind[1]]][, 2], col = "red", lty = 2)
legend("topright", legend = c("Effect", "Bootstrap CI"), col = c("blue", "red"),
      lty = c(1, 2), lwd = c(2, 1))

```

CepReg

Cepstral Regression

Description

Performs cepstral regression to model frequency domain relationships between a functional response and scalar covariates. Supports ordinary least squares (OLS), reduced-rank regression (`rrr_get`), and envelope regression (ENV) methods. Automatically selects the number of cepstral basis functions via AIC.

Usage

```

CepReg(
  y,
  x,
  method = c("ols", "rrr", "env"),
  number_of_K,
  if_bootstrap = FALSE,
  level = NULL,
  nboot = NULL,
  ind = NULL,
  nrnk = NULL
)

```

Arguments

<code>y</code>	Numeric matrix of dimension (time points) $\times$ (samples).
<code>x</code>	Numeric matrix of scalar covariates with dimensions (samples) $\times$ (covariates).
<code>method</code>	One of "ols", "rrr", or "env" specifying the regression method.
<code>number_of_K</code>	Maximum number of cepstral basis functions to consider for AIC selection.
<code>if_bootstrap</code>	Logical; whether to compute bootstrap confidence intervals (default FALSE).
<code>level</code>	Confidence level for bootstrap intervals. Required if <code>if_bootstrap = TRUE</code> .
<code>nboot</code>	Integer; the number of bootstrap samples. Required if <code>if_bootstrap = TRUE</code> .
<code>ind</code>	Integer vector; indices of covariates for which the effect functions are to be estimated and plotted. Required if <code>if_bootstrap = TRUE</code> .
<code>nrnk</code>	Integer; the rank used for reduced-rank regression. Required when <code>method = "rrr"</code> or when bootstrapping with "rrr".

Value

A list with components:

eff A list of estimated effect functions (e.g., `alpha_effect`, `beta_effect`).

boot A list of bootstrap results including confidence intervals; NULL if `if_bootstrap = FALSE`.

fit A list containing regression coefficients, residuals, smoothed spectral estimates, and other model outputs.

Examples

```
set.seed(123)
niter <- 5
len <- 20
N <- 10
p <- 2
L <- floor(len/2)-1
frq <- (1:L)/len
mu <- rep(0, p)
rho <- 0
Sigma <- generate_sig(p, rho)

X <- MASS::mvrnorm(N, mu, Sigma)
X[,1] <- runif(N, 0, 1)

spec <- matrix(0,len,N)
for(j in 1:N){
  eta1 <- rnorm(1,0,0.5)
  eta2 <- rnorm(1,0,0.5)
  eta3 <- rnorm(1,0,0.5)
  spec[,j] <- exp(
    2*cos(2*pi*(1:len)/len) +
    X[j,1]*(2*cos(4*pi*(1:len)/len)) +
    eta1 + eta2*cos(2*pi*(1:len)/len) +
    eta3*(cos(4*pi*(1:len)/len))
  )
}

Z <- data_generater(N,len,sqrt(spec))

res_ols <- CepReg(Z, X, method = "ols", number_of_K = 5,
  if_bootstrap = TRUE, level = 0.95,
  nboot = 10, ind = 1)

eff_ols <- res_ols$eff
boot_ols <- res_ols$boot

plot(frq, eff_ols$alpha_effect, type = 'l', col = "black", xlab = "Frequency", ylab = "",
  ylim = range(c(boot_ols$alpha_ci,
    eff_ols$alpha_effect, 2*cos(2*pi*frq)+0.577)))
title(ylab = expression(alpha(omega)), line = 2, cex.lab = 1.2)
lines(frq, boot_ols$alpha_ci[, 1], col = "black", lty = 2)
```

```
lines(frq, boot_ols$alpha_ci[, 2], col = "black", lty = 2)
lines(frq, 2 * cos(2 * pi * frq) + 0.577, col = "red", lty = 1, lwd = 2)
```

cep_get

Estimate Cepstral Coefficients from Periodogram

Description

Estimates replicate-specific cepstral coefficients and smoothed log-spectra using a Fourier cosine basis and Whittle-type approximation.

Usage

```
cep_get(perd, k0, frq)
```

Arguments

perd	An matrix of periodogram.
k0	Number of cepstral coefficients.
frq	A vector of frequencies in $[0, 1]$.

Value

A list with:

f An $N \times k_0$ matrix of estimated cepstral coefficients.

ff An $N \times K$ matrix of smoothed log-spectra.

Examples

```
set.seed(123)
Y <- matrix(rnorm(20 * 5), nrow = 20, ncol = 5)
len <- nrow(Y)
L <- floor(len/2)-1
frq <- (1:L)/len
perd <- per_get(Y)
result <- cep_get(perd = perd, k0 = 3, frq = frq)
```

data_generator	<i>Generate Time Series</i>
----------------	-----------------------------

Description

Simulates real-valued time series using the Cramér spectral representation and inverse FFT.

Usage

```
data_generator(N, nobs, spec)
```

Arguments

N	Number of time series to generate.
nobs	Number of time points.
spec	Spectral density matrix.

Value

Matrix of size nobs × N of generated time series

Examples

```
set.seed(123)
N <- 3
nobs <- 20
freqs <- (1:nobs) / nobs

spec <- matrix(NA, nrow = nobs, ncol = N)
for (i in 1:N) {
  spec[, i] <- exp(2 * cos(2 * pi * freqs) + rnorm(1, sd = 0.1))
}

data_generator(N = N, nobs = nobs, spec = spec)
```

effect_get	<i>Compute Functional Effects of Intercept and Covariates</i>
------------	---

Description

Projects cepstral coefficient intercept and covariate effects onto the frequency domain using the cepstral basis functions.

Usage

```
effect_get(alpha, beta, frq, nbase, ind)
```

Arguments

alpha	A numeric vector of cepstral intercept coefficients.
beta	A numeric matrix of regression coefficients.
frq	Numeric vector of frequency points in $[0, 1]$.
nbase	Number of Fourier basis functions.
ind	An integer vector indicating the indices of covariates to be included in the model.

Value

A list containing:

alpha_effect Functional intercept across frequency.

beta_effect Matrix of functional covariate effects.

Examples

```
frq <- seq(0, 1, length.out = 16)[2:8]
alpha <- rnorm(3)
beta <- matrix(rnorm(2 * 3), 2, 3)
result <- effect_get(alpha, beta, frq, nbase = 3, ind = c(1, 2))
```

env_get

Envelope Estimator for Log-Spectral Regression

Description

Fits an envelope regression model to predict cepstral coefficients from covariates.

Usage

```
env_get(X, f, frq, nbase)
```

Arguments

X	A numeric matrix of predictors ($N \times p$).
f	A numeric matrix of cepstral coefficients ($N \times \text{nbase}$).
frq	Numeric vector of frequencies in $[0, 1]$.
nbase	Number of Fourier basis functions.

Value

A list containing:

alph Intercept vector.

bet Envelope regression coefficient matrix.

spechat Estimated smoothed log-spectra.

res Residuals from envelope model.

Examples

```

library(Renvlp)

set.seed(123)
frq <- seq(0, 1, length.out = 16)[2:8]
n <- 15
p <- 3
nbase <- 5
X <- matrix(rnorm(n * p), n, p)
f <- matrix(rnorm(n * nbase), n, nbase)

u_max <- min(ncol(X), ncol(f))
cv_errors <- numeric(u_max)
for (j in 1:u_max) {
  cv_errors[j] <- cv.xenv(X, f, j, m = 5, nperm = 10)
}
optimal_u <- which.min(cv_errors)

env_result <- env_get(X, f, frq, nbase = nbase)

```

generate_sig

*Generate Exponential Correlation Covariance Matrix***Description**

Creates an $n \times n$ covariance matrix with entries $\rho^{|i-j|}$.

Usage

```
generate_sig(n, rho)
```

Arguments

n	Dimension of the covariance matrix.
rho	Correlation decay parameter.

Value

An $n \times n$ positive definite covariance matrix.

Examples

```
S <- generate_sig(5, 0.5)
```

ols_get

*Ordinary Least Squares Estimator for Log-Spectral Regression***Description**

Performs OLS regression to estimate the association between covariates and cepstral coefficients.

Usage

```
ols_get(X, f, frq, nbase)
```

Arguments

X	A numeric matrix of predictors (N x P).
f	A numeric matrix of cepstral coefficients.
frq	A vector of frequencies in $[0, 1]$.
nbase	Number of Fourier basis functions.

Value

A list containing:

alph Intercept vector.

bet OLS coefficient matrix.

spechat Estimated smoothed log-spectra.

res Matrix of residuals.

Examples

```
frq <- seq(0, 1, length.out = 16)[2:8]
n <- 10
p <- 3
nbase <- 5
X <- matrix(rnorm(n * p), n, p)
f <- matrix(rnorm(n * nbase), n, nbase)

ols_result <- ols_get(X, f, frq, nbase)
```

perd_get	<i>Compute the Periodogram of Multivariate Time Series</i>
----------	--

Description

This function computes the periodogram for each time series in the input matrix.

Usage

```
perd_get(Y)
```

Arguments

Y	A numeric matrix of dimension $T \times N$, where each column is a univariate time series.
---	---

Value

A numeric matrix of dimension $N \times L$, where each row is the periodogram of a time series.

Examples

```
set.seed(123)
Y <- matrix(rnorm(20), ncol = 4)
perd <- perd_get(Y)
```

psi_get	<i>Generate a Fourier Cosine Basis Matrix for Log-Spectral Modeling</i>
---------	---

Description

Constructs a matrix of Fourier cosine basis functions evaluated at a given frequency grid. Used in cepstral smoothing of log-spectra.

Usage

```
psi_get(k0, frq)
```

Arguments

k0	Number of cepstral basis function.
frq	A vector of frequencies in $[0, 1]$.

Value

A $k0 \times \text{length}(\text{frq})$ matrix of basis function.

Examples

```
set.seed(123)
frq<-seq(0,1, length.out=5)
psi<-psi_get(k0=3, frq)
```

rrr_get

*Reduced-Rank Regression on Cepstral Coefficients***Description**

Fits a reduced-rank regression (RRR) between covariates and cepstral coefficients using a specified maximum rank, and reconstructs log-spectra.

Usage

```
rrr_get(X, f, frq, nbase, nrank)
```

Arguments

X	A numeric matrix of predictors (N x P).
f	A numeric matrix of cepstral coefficients.
frq	A vector of frequencies in $[0, 1]$.
nbase	Number of Fourier basis functions.
nrank	Fixed Rank for the reduced-rank regression.

Value

A list containing:

alph Estimated intercept vector.
 bet Estimated coefficient matrix.
 spechat Estimated log-spectra.
 res Matrix of residuals.

Examples

```
set.seed(123)
frq <- seq(0, 1, length.out = 16)[2:8]
n <- 5
p <- 2
nbase <- 2

X <- matrix(rnorm(n * p), n, p)

true_beta <- matrix(rnorm(p * nbase), p, nbase)
alph <- rnorm(nbase)
```

```
f <- X %*% true_beta + matrix(alph, n, nbase, byrow = TRUE) +
  matrix(rnorm(n * nbase), n, nbase)

rrr <- rrr_get(X, f, frq, nbase = nbase, nrank = 1)
```

spec_regress

*Fisher Scoring Algorithm For Estimating Cepstral Coefficients***Description**

Estimates replicate-specific cepstral coefficients and corresponding smoothed log-spectra using a Whittle likelihood approximation.

Usage

```
spec_regress(perd, psi, Wmat, k0)
```

Arguments

perd	An $N \times K$ matrix of periodogram.
psi	A matrix of cepstral basis functions of dimension $k_0 \times K$.
Wmat	The inverse Gram matrix of the basis functions.
k0	Number of cepstral basis function

Value

A list with:

f An $N \times k_0$ matrix of estimated cepstral coefficients.

ff An $N \times K$ matrix of smoothed log-spectra.

Examples

```
set.seed(123)
N <- 5
len <- 20
L <- floor(len/2) - 1
frq <- (1:L) / len

Y <- matrix(rnorm(len * N), nrow = len, ncol = N)

perd <- perd_get(Y)

k0 <- 3
psi <- psi_get(k0, frq)

Wmatin <- matrix(0, k0, k0)
for (j in 1:ncol(psi)) {
```

```
    Wmatin <- Wmatin + psi[, j] %*% t(psi[, j])
  }
  Wmat <- solve(Wmatin)

  out <- spec_regress(perd, psi, Wmat, k0)
```

Index

boot_effect, [2](#)

cep_get, [6](#)

CepReg, [4](#)

data_generator, [7](#)

effect_get, [7](#)

env_get, [8](#)

generate_sig, [9](#)

ols_get, [10](#)

perd_get, [11](#)

psi_get, [11](#)

rrr_get, [12](#)

spec_regress, [13](#)