

Inference for Sparse Spectral Precision Matrices with `cxreg`

Younghoon Kim

Navonil Deb

Sumanta Basu

July 09, 2026

Introduction

This vignette describes the inference pipeline for sparse spectral precision matrices (SPMs) implemented in `cxreg`. The methodology is developed in (2026), referred to as DEB26 throughout. We focus on practical usage of the functions.

The spectral precision matrix $\Theta(\omega)$ at frequency $\omega \in (-\pi, \pi]$ is the inverse of the spectral density matrix $f(\omega)$ at the same given frequency ω , and plays the role of a precision matrix in the spectral domain: its zero and nonzero off-diagonal entries encode the absence or presence of frequency-specific conditional dependencies among time series components.

The inference pipeline in `cxreg` consists of five steps:

1. **Data-driven bandwidth selection:** `select_m()` selects the smoothing bandwidth m via a generalised cross-validation (GCV) criterion.
2. **Spectral density estimation:** `dft.all()` and `fhat_at()` compute the discrete Fourier transforms (DFTs) and the smoothed periodogram $\hat{f}(\omega_j)$.
3. **Debiased estimation:** `decglasso()` constructs the debiased complex graphical lasso (deCGLASSO) estimator $\tilde{\Theta}_j$ from the sparse complex graphical lasso (CGLASSO).
4. **Variance estimation:** `var.cov()` estimates the asymptotic variance and pseudovariance of $\tilde{\Theta}_j$ using either a plug-in or heteroscedasticity and autocorrelation consistent (HAC) estimator.
5. **Inference:** `spec.test()` constructs entry-wise confidence ellipsoids and test statistics; `spec.fdr()` applies multiple hypothesis testing with controlled false discovery rate (FDR).

Background

Spectral density and the DFT

For a p -dimensional weakly stationary time series $\{X_t\}_{t=1}^n$, the discrete Fourier transform (DFT) at a Fourier frequency $\omega_j = 2\pi j/n$ is

$$d_j := d(\omega_j) = \frac{1}{\sqrt{n}} \sum_{t=1}^n X_t \exp(-it\omega_j).$$

Asymptotically, $d_j \sim \mathcal{N}_{\mathbb{C}}(0, f(\omega_j))$. The smoothed periodogram at ω_j with half-bandwidth m is

$$\hat{f}(\omega_j) = \frac{1}{2\pi(2m+1)} \sum_{k \in W_m(j)} d_k d_k^*,$$

where $W_m(j) = \{k : |k - j| \leq m\}$ is the set of $2m + 1$ neighboring frequencies around ω_j .

The CGLASSO and deCGLASSO estimators

The CGLASSO estimator $\hat{\Theta}_j$ is the minimizer of the optimization problem

$$\hat{\Theta} = \arg \min_{\Theta \in \mathcal{H}_{++}^p} \left\{ -\log \det \Theta + \text{trace}(\hat{f}_j \Theta) + \lambda \|\Theta^-\|_1 \right\},$$

where $\|\Theta^-\|_1 = \sum_{a \neq b} |\Theta_{a,b}|$ penalizes off-diagonal entries of Θ . In high dimensions, $\hat{\Theta}_j$ has a regularization bias due to the penalty term. The deCGLASSO estimator removes this bias through the computation (2026):

$$\tilde{\Theta}_j = 2\hat{\Theta}_j - \hat{\Theta}_j \hat{f}_j \hat{\Theta}_j.$$

Under suitable regularity conditions, (2026) showed that $\sqrt{2m+1}(\tilde{\Theta}_j - \Theta_j)_{ab}$ is asymptotically bivariate normal with a covariance matrix constructed by using Θ_j .

Step 1 — Generate a multivariate white noise process

We illustrate the pipeline using a multivariate Gaussian white noise process $X_t \sim \text{WN}(0, \Sigma)$, $t = 1, \dots, n$, with a sparse tridiagonal precision matrix $\Theta = \Sigma^{-1}$. Note that the spectral density of white noise process is constant across all frequencies:

$$f(\omega) = \frac{1}{2\pi} \Sigma, \quad \omega \in (-\pi, \pi],$$

so the true SPM is also constant over frequencies:

$$\Theta(\omega) = f(\omega)^{-1} = 2\pi \Sigma^{-1}.$$

```
library(cxreg)
library(mvtnorm)

set.seed(1010)
p <- 10      # number of variables
n <- 500     # sample size

# True precision matrix: tridiagonal
C <- diag(0.7, p)
C[row(C) == col(C) + 1] <- 0.3
C[row(C) == col(C) - 1] <- 0.3
Sigma <- solve(C)

# True SPM: constant across all frequencies
Theta_true <- 2*pi*C

# Generate white noise observations
X <- rmvnorm(n = n, mean = rep(0,p), sigma=Sigma)
cat("Data dimensions:", dim(X), "\n")

## Data dimensions: 500 10
```

```
cat("True SPM (1:4, 1:4):\n")
```

```
## True SPM (1:4, 1:4):
```

```
print(round(Re(Theta_true[1:4, 1:4]), 3))
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 4.398 1.885 0.000 0.000
## [2,] 1.885 4.398 1.885 0.000
## [3,] 0.000 1.885 4.398 1.885
## [4,] 0.000 0.000 1.885 4.398
```

Step 2 — Bandwidth selection via GCV

`select_m()` searches over a grid of half-bandwidths and selects the one that minimizes a GCV criterion based on the deviance of the diagonal periodogram (2001). The GCV criterion provides a principled data-driven choice of the half-bandwidth m .

```
bw_sel <- select_m(X, verbose = FALSE)
m <- bw_sel$m_opt
cat("Selected bandwidth: m =", m, "(full bandwidth:", 2*m+1, ")\n")
```

```
## Selected bandwidth: m = 67 (full bandwidth: 135 )
```

```
bw_sel$gcv_table
```

```
##      m  bw      lambda deviance      gcv gcv_denom
## 1 11  23 0.3164054 0.5397535 0.5899372 0.9149338
## 2 16  33 0.2641502 0.5537477 0.5888977 0.9403122
## 3 22  45 0.2262047 0.5507499 0.5760685 0.9560494
## 4 27  55 0.2046098 0.5522657 0.5729094 0.9639669
## 5 33  67 0.1853832 0.5569039 0.5739076 0.9703720
## 6 44  89 0.1608470 0.5579496 0.5707023 0.9776543
## 7 55 111 0.1440278 0.5585350 0.5687363 0.9820631
## 8 67 135 0.1305993 0.5603018 0.5686957 0.9852401
```

Step 3 — Compute the DFT and smoothed periodogram

We target $j = \lfloor n/4 \rfloor$, corresponding to $\omega \approx \pi/2$.

```
j <- floor(n / 4)
dft <- dft.all(X) # full DFT matrix: n x p
fhat <- fhat_at(dft, j = j, m = m) # smoothed periodogram: p x p
cat("Frequency index j =", j, "(omega =", round(2*pi*j/n, 3), ")\n")
```

```
## Frequency index j = 125 (omega = 1.571 )
```

```
# For white noise: fhat should be close to Sigma / (2*pi)
cat("Max deviation of Re(fhat) from Sigma/(2pi):", round(max(abs(Re(fhat) - Sigma / (2*pi))), 4), "\n")
```

```
## Max deviation of Re(fhat) from Sigma/(2pi): 0.0869
```

Step 4 — Fit CGLASSO and compute the deCGLASSO estimator

We fit the CGLASSO path using the EBIC stopping criterion, then apply one-step debiasing:

```
# Fit CGLASSO
fit <- cglasso(S = fhat, m = m, type = "II")
```

```
## The algorithm was terminated at 39 th lambda
```

```
cat("EBIC-selected lambda index:", fit$min_index, "\n")
```

```
## EBIC-selected lambda index: 23
```

```
cat("Lambda selected:", round(fit$lambda_grid[fit$min_index], 4), "\n")
```

```
## Lambda selected: 0.065
```

```
# One-step debiasing to produce deCGLASSO
deb <- decglasso(object = fit, fhat = fhat)
cat("Class of deCGLASSO output:", class(deb), "\n")
```

```
## Class of deCGLASSO output: decglasso
```

```
Theta_tilde <- deb$Theta_tilde
```

Step 5 — Estimate the asymptotic variance

```
vc_plug <- var.cov(Theta = Theta_tilde, X = X, j = j, m = m, type = "plug-in")
vc_hac <- var.cov(Theta = Theta_tilde, X = X, j = j, m = m, type = "HAC")
cat("Fields:", names(vc_plug), "\n")
```

```
## Fields: Vre Vim Cov V PV type
```

The output contains:

- **Vre**: estimated variance of $\text{Re}(\tilde{\Theta}_{ab})$, a $p \times p$ matrix.
- **Vim**: estimated variance of $\text{Im}(\tilde{\Theta}_{ab})$, a $p \times p$ matrix.
- **Cov**: estimated covariance of Re and Im parts, a $p \times p$ matrix.
- **V**: complex variance σ^2 , a $p \times p$ complex matrix.
- **PV**: pseudovariance δ^2 , a $p \times p$ complex matrix.

These satisfy $\text{Vre} = \frac{1}{2}\text{Re}(V + PV)$, $\text{Vim} = \frac{1}{2}\text{Re}(V - PV)$, and $\text{Cov} = \frac{1}{2}\text{Im}(PV)$. Compare the two estimators on the (1, 2) entry:

```
cat("Entry (1,2) variance (Re part):\n")
```

```
## Entry (1,2) variance (Re part):
```

```
cat("Plug-in Vre:", round(vc_plug$Vre[1,2], 6), "\n")
```

```
## Plug-in Vre: 11.89146
```

```
cat("HAC Vre:", round(vc_hac$Vre[1,2], 6), "\n")
```

```
## HAC Vre: 12.61249
```

Step 6 — Entry-wise tests and confidence regions

`spec.test()` computes for each off-diagonal entry (a, b) :

- Z-statistics for $\text{Re}(\tilde{\Theta}_{ab})$ and $\text{Im}(\tilde{\Theta}_{ab})$.
- The Mahalanobis chi-squared statistic $(2m+1)\hat{D}_{j,m}^{(a,b)}(0) \sim \chi_2^2$ under $H_0 : \Theta_{ab} = 0$.
- Half-widths of the marginal confidence intervals (CI) for the real and imaginary parts, and areas of the joint 95% confidence region (ellipsoids) are produced.

```
alpha <- 0.2
st_plug <- spec.test(Est = Theta_tilde, varcov = vc_plug, m = m, alpha = alpha)
st_hac <- spec.test(Est = Theta_tilde, varcov = vc_hac, m = m, alpha = alpha)

cat("Chi-squared statistic at (1,2) [true edge]:\n")
```

```
## Chi-squared statistic at (1,2) [true edge]:
```

```
cat("Plug-in:", round(st_plug$Chi_sq[1,2], 3), "\n")
```

```
## Plug-in: 57.943
```

```
cat("HAC:", round(st_hac$Chi_sq[1,2], 3), "\n")
```

```
## HAC: 54.901
```

```
cat("Chi-sq(2, 0.95) critical value:", round(qchisq(0.95, 2), 3), "\n")
```

```
## Chi-sq(2, 0.95) critical value: 5.991
```

```
cat("\n Chi-squared statistic at (1,3) [true null edge]:\n")
```

```
##
```

```
## Chi-squared statistic at (1,3) [true null edge]:
```

```
cat("Plug-in:", round(st_plug$Chi_sq[1,3], 3), "\n")
```

```
## Plug-in: 0.68
```

```
cat("HAC:", round(st_hac$Chi_sq[1,3], 3), "\n")
```

```
## HAC: 0.812
```

The CI half-widths for the real and imaginary parts at the (1,2) entry:

```
cat("80% CI half-width at (1,2):\n")
```

```
## 80% CI half-width at (1,2):
```

```
cat("Re part (plug-in):", round(st_plug$wing_re[1,2], 4), "\n")
```

```
## Re part (plug-in): 0.3804
```

```
cat("Im part (plug-in):", round(st_plug$wing_im[1,2], 4), "\n")
```

```
## Im part (plug-in): 0.3038
```

For white noise process, the imaginary part of $\Theta(\omega)$ is zero, so we expect the imaginary CI to contain zero and the real CI to contain $2\pi(\Sigma^{-1})_{ab}$:

```
# Check if true value falls inside the 95% CI
true_re_12 <- Re(Theta_true[1,2])
est_re_12 <- Re(Theta_tilde[1,2])
hw_re_12 <- st_plug$wing_re[1,2]
cat("True Re(Theta_12):", round(true_re_12, 4), "\n")
```

```
## True Re(Theta_12): 1.885
```

```
cat("80% CI: [", round(est_re_12 - hw_re_12, 4), ",", round(est_re_12 + hw_re_12, 4), "]\n")
```

```
## 80% CI: [ 1.8757 , 2.6365 ]
```

```
cat("True value inside CI:", abs(est_re_12 - true_re_12) <= hw_re_12, "\n")
```

```
## True value inside CI: TRUE
```

Step 7 — FDR-controlled support recovery

`spec.fdr()` applies the thresholding procedure of (2013) to control the FDR across all $p(p-1)/2$ off-diagonal entries simultaneously.

```
fdr_plug <- spec.fdr(Chi_sq = st_plug$Chi_sq, alpha = alpha, diag = FALSE)
fdr_hac <- spec.fdr(Chi_sq = st_hac$Chi_sq, alpha = alpha, diag = FALSE)

cat("Number of significant edges (plug-in):", sum(fdr_plug$Decision[upper.tri(fdr_plug$Decision)]), "\n")
```

```
## Number of significant edges (plug-in): 9
```

```
cat("Number of significant edges (HAC):", sum(fdr_hac$Decision[upper.tri(fdr_hac$Decision)]), "\n")
```

```
## Number of significant edges (HAC): 10
```

```
cat("FDR threshold tau (plug-in):", round(fdr_plug$tau, 3), "\n")
```

```
## FDR threshold tau (plug-in): 6.46
```

For the white noise process, the true edge set of $\Theta(\omega) = 2\pi\Sigma^{-1}$ is the same at every frequency. Compare the discoveries to the truth:

```
# True support: nonzero off-diagonal entries of C (same as Theta_true up to scale)
true_support <- (C!=0)*1L
diag(true_support) <- 0

# Confusion matrix (plug-in)
dec <- fdr_plug$Decision
TP <- sum(dec == 1 & true_support == 1 & upper.tri(dec))
FP <- sum(dec == 1 & true_support == 0 & upper.tri(dec))
FN <- sum(dec == 0 & true_support == 1 & upper.tri(dec))

cat("True edges in upper triangle:", sum(true_support[upper.tri(true_support)]), "\n")
```

```
## True edges in upper triangle: 9
```

```
cat("True Positives:", TP, "\n")
```

```
## True Positives: 9
```

```
cat("False Positives:", FP, "\n")
```

```
## False Positives: 0
```

```
cat("False Negatives:", FN, "\n")
```

```
## False Negatives: 0
```

```
cat("Empirical FDR:", round(FP / max(TP + FP, 1), 3), "\n")
```

```
## Empirical FDR: 0
```

```
cat("Power:", round(TP / max(TP + FN, 1), 3), "\n")
```

```
## Power: 1
```

Summary of functions

Step	Function	Description
1	<code>select_m()</code>	GCV bandwidth selection
2	<code>dft.all()</code>	Full DFT matrix
2	<code>fhat_at()</code>	Smoothed periodogram at frequency j
3	<code>cglasso()</code>	CGLASSO estimator path
4	<code>decglasso()</code>	One-step debiasing
5	<code>var.cov()</code>	Asymptotic variance estimation (plug-in or HAC)
6	<code>spec.test()</code>	Chi-squared statistics and CI half-widths
7	<code>spec.fdr()</code>	FDR-controlled support recovery

References

- Deb, Navonil, Younghoon Kim, and Sumanta Basu. 2026. “Inference for High-Dimensional Sparse Spectral Precision Matrices.” *arXiv Preprint arXiv:2606.07986*.
- Liu, Weidong. 2013. “Gaussian Graphical Model Estimation with False Discovery Rate Control.” *The Annals of Statistics* 41 (6): 2948–78.
- Ombao, Hernando C, Jonathan A Raz, Robert L Strawderman, and Rainer Von Sachs. 2001. “A Simple Generalised Crossvalidation Method of Span Selection for Periodogram Smoothing.” *Biometrika* 88 (4): 1186–92.