# Package: funbootband (via r-universe)

May 19, 2026

**Title** Simultaneous Prediction and Confidence Bands for Time Series Data

**Version** 0.2.0

**Description** Provides methods to compute simultaneous prediction and confidence bands for dense time series data. The implementation builds on the functional bootstrap approach proposed by Lenhoff et al. (1999) <doi:10.1016/S0966-6362(98)00043-5> and extended by Koska et al. (2023) <doi:10.1016/j.jbiomech.2023.111506> to support both independent and clustered (hierarchical) data. Includes a simple API (see band()) and an 'Rcpp' backend for performance.

**License** GPL-3

**URL** https://github.com/koda86/funbootband-cran

**BugReports** https://github.com/koda86/funbootband-cran/issues

**Depends** R (>= 3.5)

**Imports** Rcpp, stats

**LinkingTo** Rcpp

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**VignetteBuilder** knitr

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**SystemRequirements** C++17

**Config/testthat/edition** 3

**ByteCompile** true

**Repository** https://koda86.r-universe.dev

**Date/Publication** 2025-10-14 13:36:02 UTC

**RemoteUrl** https://github.com/koda86/funbootband-cran

**RemoteRef** HEAD

**RemoteSha** c67a0f5a6495064a475a8a289a3da949329e8e3d

# Contents



---

band *Simultaneous Bands for Functional Data*

---

## Description

Create simultaneous bootstrap bands for dense functional data (rows are time points, columns are curves). Supports clustered designs via a simple cluster bootstrap when `iid = FALSE`.

## Usage

```
band(
  data,
  type = c("prediction", "confidence"),
  alpha = 0.05,
  iid = TRUE,
  id = NULL,
  B = 1000L,
  k.coef = 50L
)
```

## Arguments

| | |
|---|---|
| `data` | Numeric matrix with T rows (time) and n columns (curves). A data.frame of numeric columns is also accepted and coerced to a matrix. |
| `type` | Character, either "prediction" or "confidence". |
| `alpha` | Numeric in (0, 1). Use 0.05 for 95% bands. |
| `iid` | Logical; if FALSE, use a cluster bootstrap (requires `id` or infers clusters from column-name prefixes). |
| `id` | Optional integer/factor vector of length ncol(data) giving a cluster id for each curve (used when `iid = FALSE`). If NULL and `iid = FALSE`, clusters are inferred from column names by prefix (up to the first underscore, hyphen, or dot). |
| `B` | Integer, number of bootstrap iterations (e.g., 1000 for final results; use smaller values in examples/tests). |
| `k.coef` | Integer; number of Fourier harmonics (default 50). Automatically clamped to $\lfloor (T-1)/2 \rfloor$ based on the grid length. Larger values fit more high-frequency detail; smaller values smooth more. |

## Value

A list with elements `lower`, `mean`, `upper` (each of length T) and `meta` (a list with settings such as type, alpha, iid, B, n, T).

## References

Koska, D., Oriwol, D., & Maiwald, C. (2023). Comparison of statistical models for characterizing continuous differences between two biomechanical measurement systems. *Journal of Biomechanics*, 149, 111506. doi:10.1016/j.jbiomech.2023.111506

Lenhoff, M. W., Santner, T. J., Otis, J. C., Peterson, M. G. E., Williams, B. J., & Backus, S. I. (1999). Bootstrap prediction and confidence bands: a superior statistical method for analysis of gait data. *Gait & Posture*, 9(1), 10–17. doi:10.1016/S0966-6362(98)00043-5

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press. doi:10.1017/cbo9780511802843

## Examples

```
## i.i.d. example

set.seed(1)
T <- 200
n <- 10
x <- seq(0, 1, length.out = T)

# Simulate smooth Gaussian-process-like curves of equal length
mu  <- 10 * sin(2 * pi * x)
ell <- 0.12; sig <- 3
Kmat <- outer(x, x, function(s, t) sig^2 * exp(-(s - t)^2 / (2 * ell^2)))
ev <- eigen(Kmat + 1e-8 * diag(T), symmetric = TRUE)
Z  <- matrix(rnorm(T * n), T, n)
Y  <- mu + ev$vectors %*% (sqrt(pmax(ev$values, 0)) * Z)
Y  <- Y + matrix(rnorm(T * n, sd = 0.2), T, n) # observation noise

# Fit prediction and confidence bands
fit_pred <- band(Y, type = "prediction", alpha = 0.11, iid = TRUE, B = 1000L, k.coef = 50L)
fit_conf <- band(Y, type = "confidence", alpha = 0.11, iid = TRUE, B = 1000L, k.coef = 50L)

# Plot the results
x_idx <- seq_len(fit_pred$meta$T)
ylim  <- range(c(Y, fit_pred$lower, fit_pred$upper), finite = TRUE)

plot(x_idx, fit_pred$mean, type = "n", ylim = ylim,
     xlab = "Index (Time)", ylab = "Amplitude",
     main = "Simultaneous bands (i.i.d.)")

matlines(x_idx, Y, col = "gray70", lty = 1, lwd = 1)
polygon(c(x_idx, rev(x_idx)), c(fit_pred$lower, rev(fit_pred$upper)),
        col = grDevices::adjustcolor("steelblue", alpha.f = 0.25), border = NA)
polygon(c(x_idx, rev(x_idx)), c(fit_conf$lower, rev(fit_conf$upper)),
        col = grDevices::adjustcolor("gray40", alpha.f = 0.3), border = NA)
lines(x_idx, fit_pred$mean, col = "black", lwd = 1)

## clustered (hierarchical) example

set.seed(2)
T <- 200
```

```
m <- c(5, 5)
x <- seq(0, 1, length.out = T)

# Cluster-specific means
mu <- list(
  function(z) 8 * sin(2 * pi * z),
  function(z) 8 * cos(2 * pi * z)
)

# Generate curves with smooth within-cluster variation
Bm <- cbind(sin(2 * pi * x), cos(2 * pi * x))
gen_curve <- function(k) {
  sc <- rnorm(ncol(Bm), sd = c(2.0, 1.5))
  mu[[k]](x) + as.vector(Bm %*% sc)
}

Ylist <- lapply(seq_along(m), function(k) {
  sapply(seq_len(m[k]), function(i) gen_curve(k) + rnorm(T, sd = 0.6))
})
Y <- do.call(cbind, Ylist)
colnames(Y) <- unlist(mapply(
  function(k, mk) paste0("C", k, "_", seq_len(mk)),
  seq_along(m), m
))


# Fit prediction and confidence bands
fit_pred <- band(Y, type = "prediction", alpha = 0.11, iid = FALSE, B = 1000L, k.coef = 50L)
fit_conf <- band(Y, type = "confidence", alpha = 0.11, iid = FALSE, B = 1000L, k.coef = 50L)

# Plot the results
x_idx <- seq_len(fit_pred$meta$T)
ylim   <- range(c(Y, fit_pred$lower, fit_pred$upper), finite = TRUE)

plot(x_idx, fit_pred$mean, type = "n", ylim = ylim,
     xlab = "Index (Time)", ylab = "Amplitude",
     main = "Simultaneous bands (clustered)")

matlines(x_idx, Y, col = "gray70", lty = 1, lwd = 1)
polygon(c(x_idx, rev(x_idx)), c(fit_pred$lower, rev(fit_pred$upper)),
        col = grDevices::adjustcolor("steelblue", alpha.f = 0.25), border = NA)
polygon(c(x_idx, rev(x_idx)), c(fit_conf$lower, rev(fit_conf$upper)),
        col = grDevices::adjustcolor("gray40", alpha.f = 0.3), border = NA)
lines(x_idx, fit_pred$mean, col = "black", lwd = 1)
```

# Index