

# Package ‘SpatPCA’

May 7, 2026

**Title** Regularized Principal Component Analysis for Spatial Data

**Version** 1.3.8

**Description** Provide regularized principal component analysis incorporating smoothness, sparseness and orthogonality of eigen-functions by using the alternating direction method of multipliers algorithm (Wang and Huang, 2017, <[DOI:10.1080/10618600.2016.1157483](https://doi.org/10.1080/10618600.2016.1157483)>). The method can be applied to either regularly or irregularly spaced data, including 1D, 2D, and 3D.

**License** GPL (>= 2)

**ByteCompile** true

**BugReports** <https://github.com/egpivo/SpatPCA/issues>

**Depends** R (>= 3.4.0)

**Imports** Rcpp (>= 1.0.12), ggplot2, parallel

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** knitr, rmarkdown, testthat (>= 2.1.0), dplyr (>= 1.0.3), tidyr, fields, scico, plot3D, pracma, RColorBrewer, maps, covr, styler, V8

**SystemRequirements** GNU make

**VignetteBuilder** knitr, rmarkdown

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**URL** <https://egpivo.github.io/SpatPCA/>,  
<https://github.com/egpivo/SpatPCA>

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Wen-Ting Wang [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-3051-7302>>),  
Hsin-Cheng Huang [aut] (ORCID: <<https://orcid.org/0000-0002-5613-349X>>)

**Maintainer** Wen-Ting Wang <egpivo@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-09-28 10:40:18 UTC

## Contents

SpatPCA-package . . . . .	2
plot.spatpca . . . . .	2
predict . . . . .	3
predictEigenfunction . . . . .	4
spatpca . . . . .	5
thinPlateSplineMatrix . . . . .	8

<b>Index</b>	<b>9</b>
--------------	----------

---

SpatPCA-package	<i>Regularized Principal Component Analysis for Spatial Data</i>
-----------------	--

---

### Description

A new regularization approach to estimate the leading spatial patterns via smoothness and sparseness penalties, and spatial predictions for spatial data that may be irregularly located in space (including 1D, 2D and 3D), and obtain the spatial prediction at the designated locations.

### Details

Package:	SpatPCA
Type:	Package
Version:	1.3.3.4
Date:	2021-02-11
License:	GPL version 3

### Author(s)

Wen-Ting Wang <egpivo@gmail.com> and Hsin-Cheng Huang <hchuang@stat.sinica.edu.tw>

---

plot.spatpca	<i>Display the cross-validation results</i>
--------------	---

---

### Description

Display the M-fold cross-validation results

### Usage

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

**Arguments**

x                    An spatpca class object for plot method  
...                   Not used directly

**Value**

NULL.

**See Also**

[spatpca](#)

**Examples**

```
x_1D <- as.matrix(seq(-5, 5, length = 10))
Phi_1D <- exp(-x_1D^2) / norm(exp(-x_1D^2), "F")
set.seed(1234)
Y_1D <- rnorm(n = 100, sd = 3) %*% t(Phi_1D) + matrix(rnorm(n = 100 * 10), 100, 10)
cv_1D <- spatpca(x = x_1D, Y = Y_1D)
plot(cv_1D)
```

---

predict

*Spatial predictions on new locations*

---

**Description**

Predict the response on new locations with the estimated spatial structures.

**Usage**

```
predict(spatpca_object, x_new, eigen_patterns_on_new_site = NULL)
```

**Arguments**

spatpca\_object    An spatpca class object  
x\_new              New location matrix.  
eigen\_patterns\_on\_new\_site  
                    Eigen-patterns on x\_new

**Value**

A prediction matrix of Y at the new locations, x\_new.

**See Also**

[spatpca](#)

**Examples**

```
# 1D: artificial irregular locations
x_1D <- as.matrix(seq(-5, 5, length = 10))
Phi_1D <- exp(-x_1D^2) / norm(exp(-x_1D^2), "F")
set.seed(1234)
Y_1D <- rnorm(n = 100, sd = 3) %*% t(Phi_1D) + matrix(rnorm(n = 100 * 10), 100, 10)
removed_location <- sample(1:10, 3)
removed_x_1D <- x_1D[-removed_location]
removed_Y_1D <- Y_1D[, -removed_location]
new_x_1D <- as.matrix(seq(-5, 5, length = 20))
cv_1D <- spatpca(x = removed_x_1D, Y = removed_Y_1D, tau2 = 1:100)
predictions <- predict(cv_1D, x_new = new_x_1D)
```

---

predictEigenfunction *Spatial dominant patterns on new locations*

---

**Description**

Estimate K eigenfunctions on new locations

**Usage**

```
predictEigenfunction(spatpca_object, x_new)
```

**Arguments**

spatpca\_object An spatpca class object  
x\_new New location matrix.

**Value**

A matrix with K Eigenfunction values on new locations.

**See Also**

[spatpca](#)

**Examples**

```
# 1D: artificial irregular locations
x_1D <- as.matrix(seq(-5, 5, length = 10))
Phi_1D <- exp(-x_1D^2) / norm(exp(-x_1D^2), "F")
set.seed(1234)
Y_1D <- rnorm(n = 100, sd = 3) %*% t(Phi_1D) + matrix(rnorm(n = 100 * 10), 100, 10)
rm_loc <- sample(1:10, 2)
x_1Drm <- x_1D[-rm_loc]
Y_1Drm <- Y_1D[, -rm_loc]
x_1Dnew <- as.matrix(seq(-5, 5, length = 20))
```

```
cv_1D <- spatpca(x = x_1Drm, Y = Y_1Drm, tau2 = 1:100)
dominant_patterns <- predictEigenfunction(cv_1D, x_new = x_1Dnew)
```

spatpca

*Regularized PCA for spatial data***Description**

Produce spatial dominant patterns and spatial predictions at the designated locations according to the specified tuning parameters or the selected tuning parameters by the M-fold cross-validation.

**Usage**

```
spatpca(
  x,
  Y,
  M = 5,
  K = NULL,
  is_K_selected = ifelse(is.null(K), TRUE, FALSE),
  tau1 = NULL,
  tau2 = NULL,
  gamma = NULL,
  is_Y_detrended = FALSE,
  maxit = 100,
  thr = 1e-04,
  num_cores = NULL
)
```

**Arguments**

x	Location matrix ( $p \times d$ ). Each row is a location. $d$ is the dimension of locations
Y	Data matrix ( $n \times p$ ) stores the values at $p$ locations with sample size $n$ .
M	Optional number of folds for cross validation; default is 5.
K	Optional user-supplied number of eigenfunctions; default is NULL. If K is NULL or is_K_selected is TRUE, K is selected automatically.
is_K_selected	If TRUE, K is selected automatically; otherwise, is_K_selected is set to be user-supplied K. Default depends on user-supplied K.
tau1	Optional user-supplied numeric vector of a non-negative smoothness parameter sequence. If NULL, 10 tau1 values in a range are used.
tau2	Optional user-supplied numeric vector of a non-negative sparseness parameter sequence. If NULL, none of tau2 is used.
gamma	Optional user-supplied numeric vector of a non-negative tuning parameter sequence. If NULL, 10 values in a range are used.
is_Y_detrended	If TRUE, center the columns of Y. Default is FALSE.

maxit	Maximum number of iterations. Default value is 100.
thr	Threshold for convergence. Default value is $10^{-4}$ .
num_cores	Optional numeric value indicating desired cores. The request is validated but computations run sequentially. Default value is NULL.

### Details

An ADMM form of the proposed objective function is written as

$$\min_{\Phi} \|\mathbf{Y} - \mathbf{Y}\Phi\Phi'\|_F^2 + \tau_1 \text{tr}(\Phi^T \Omega \Phi) + \tau_2 \sum_{k=1}^K \sum_{j=1}^p |\phi_{jk}|,$$

subject to  $\Phi^T \Phi = \mathbf{I}_K$ , where  $\mathbf{Y}$  is a data matrix,  $\Omega$  is a smoothness matrix, and  $\Phi = \{\phi_{jk}\}$ .

### Value

A list of objects including

eigenfn	Estimated eigenfunctions at the new locations, x_new.
selected_K	Selected K based on CV. Execute the algorithm when is_K_selected is TRUE.
selected_tau1	Selected tau1.
selected_tau2	Selected tau2.
selected_gamma	Selected gamma.
cv_score_tau1	cv scores for tau1.
cv_score_tau2	cv scores for tau2.
cv_score_gamma	cv scores for gamma.
tau1	Sequence of tau1-values used in the process.
tau2	Sequence of tau2-values used in the process.
gamma	Sequence of gamma-values used in the process.
detrended_Y	If is_Y_detrended is TRUE, detrended_Y means Y is detrended; else, detrended_Y is equal to Y.
scaled_x	Input location matrix. Only scale when it is one-dimensional

### Author(s)

Wen-Ting Wang and Hsin-Cheng Huang

### References

Wang, W.-T. and Huang, H.-C. (2017). Regularized principal component analysis for spatial data. *Journal of Computational and Graphical Statistics* **26** 14-25.

### See Also

[predict](#)

## Examples

```

# The following examples run sequentially.
## 1D: regular locations
x_1D <- as.matrix(seq(-5, 5, length = 50))
Phi_1D <- exp(-x_1D^2) / norm(exp(-x_1D^2), "F")
set.seed(1234)
Y_1D <- rnorm(n = 100, sd = 3) %*% t(Phi_1D) + matrix(rnorm(n = 100 * 50), 100, 50)
cv_1D <- spatpca(x = x_1D, Y = Y_1D)
plot(x_1D, cv_1D$eigenfn[, 1], type = "l", main = "1st eigenfunction")
lines(x_1D, svd(Y_1D)$v[, 1], col = "red")
legend("topleft", c("SpatPCA", "PCA"), lty = 1:1, col = 1:2)

## 2D: Daily 8-hour ozone averages for sites in the Midwest (USA)
if (requireNamespace("fields", quietly = TRUE) &&
    requireNamespace("pracma", quietly = TRUE) &&
    requireNamespace("maps", quietly = TRUE)) {
  library(fields)
  library(pracma)
  library(maps)
  data(ozone2)
  x <- ozone2$lon.lat
  Y <- ozone2$y
  date <- as.Date(ozone2$date, format = "%y%m%d")
  rmna <- !colSums(is.na(Y))
  YY <- matrix(Y[, rmna], nrow = nrow(Y))
  YY <- detrend(YY, "linear")
  xx <- x[rmna, ]
  cv <- spatpca(x = xx, Y = YY)
  quilt.plot(xx, cv$eigenfn[, 1])
  map("state", xlim = range(xx[, 1]), ylim = range(xx[, 2]), add = TRUE)
  map.text("state", xlim = range(xx[, 1]), ylim = range(xx[, 2]), cex = 2, add = TRUE)
  plot(date, YY %*% cv$eigenfn[, 1], type = "l", ylab = "1st Principal Component")
  ### new locations
  new_p <- 200
  x_lon <- seq(min(xx[, 1]), max(xx[, 1]), length = new_p)
  x_lat <- seq(min(xx[, 2]), max(xx[, 2]), length = new_p)
  xx_new <- as.matrix(expand.grid(x = x_lon, y = x_lat))
  eof <- spatpca(x = xx,
                Y = YY,
                K = cv$selected_K,
                tau1 = cv$selected_tau1,
                tau2 = cv$selected_tau2)
  predicted_eof <- predictEigenfunction(eof, xx_new)
  quilt.plot(xx_new,
            predicted_eof[, 1],
            nx = new_p,
            ny = new_p,
            xlab = "lon.",
            ylab = "lat.")
  map("state", xlim = range(x_lon), ylim = range(x_lat), add = TRUE)
  map.text("state", xlim = range(x_lon), ylim = range(x_lat), cex = 2, add = TRUE)
}

```

```

}
## 3D: regular locations
p <- 10
x <- y <- z <- as.matrix(seq(-5, 5, length = p))
d <- expand.grid(x, y, z)
Phi_3D <- rowSums(exp(-d^2)) / norm(as.matrix(rowSums(exp(-d^2))), "F")
Y_3D <- rnorm(n = 100, sd = 3) %*% t(Phi_3D) + matrix(rnorm(n = 100 * p^3), 100, p^3)
cv_3D <- spatpca(x = d, Y = Y_3D, tau2 = seq(0, 1000, length = 10))
library(plot3D)
library(RColorBrewer)
cols <- colorRampPalette(brewer.pal(9, "Blues"))(p)
isosurf3D(x, y, z,
          colvar = array(cv_3D$eigenfn[, 1], c(p, p, p)),
          level = seq(min(cv_3D$eigenfn[, 1]), max(cv_3D$eigenfn[, 1]), length = p),
          ticktype = "detailed",
          colkey = list(side = 1),
          col = cols)

```

---

thinPlateSplineMatrix *Thin-plane spline matrix*

---

### Description

Produce a thin-plane spline matrix based on a given location matrix

### Usage

```
thinPlateSplineMatrix(location)
```

### Arguments

location            A location matrix

### Value

A thin-plane spline matrix

### Examples

```

pesudo_sequence <- seq(-5, 5, length = 5)
two_dim_location <- as.matrix(expand.grid(x = pseudo_sequence, y = pseudo_sequence))
thin_plate_matrix <- thinPlateSplineMatrix(two_dim_location)

```

# Index

`plot.spatpca`, 2  
`predict`, 3, 6  
`predictEigenfunction`, 4  
  
`spatpca`, 3, 4, 5  
SpatPCA-package, 2  
  
`thinPlateSplineMatrix`, 8