Package 'odpc'

March 2, 2022

Type Package

Title One-Sided Dynamic Principal Components

Version 2.0.5

Date 2022-02-25

Author Daniel Peña <daniel.pena@uc3m.es>,
Ezequiel Smucler <ezequiels.90@gmail.com>,
Victor Yohai <vyohai@dm.uba.ar>

Maintainer Ezequiel Smucler <ezequiels.90@gmail.com>

Description Functions to compute the one-sided dynamic
principal components ('odpc') introduced in Peña, Smucler and Yohai (2019)
<DOI:10.1080/01621459.2018.1520117>. 'odpc' is a novel dimension
reduction technique for multivariate time series, that is useful
for forecasting. These dynamic principal components are defined as
the linear combinations of the present and past values of the series
that minimize the reconstruction mean squared error.

License GPL (>= 2)

Biarch true

Imports methods, Rcpp (>= 0.12.7), forecast, parallel, doParallel,
foreach, MASS

LinkingTo Rcpp, RcppArmadillo (>= 0.7.500.0.0)

Suggests testthat

Depends R (>= 3.3.0)

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.1.0

Repository CRAN

Date/Publication 2022-03-02 19:50:02 UTC

R topics documented:

`components_odpcs` *Get One-Sided Dynamic Principal Components From an odpcs Object*

Description

Get One-Sided Dynamic Principal Components from an `odpcs` object.

Usage

```
components_odpcs(object, which_comp = 1)
```

Arguments

`object`	An object of class `odpcs`, usually the result of odpc.
`which_comp`	Numeric vector indicating which components to get. Default is 1.

Value

A list whose entries are the desired dynamic principal components.

See Also

`odpc`, `crit.odpc`, `cv.odpc`

Examples

```
T <- 200 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
 x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
fit <- odpc(x, ks = matrix(c(1, 1, 1, 0), 2, 2))
comps <- components_odpcs(fit, which_comp = c(1, 2))
```

`crit.odpc` *Automatic Choice of Tuning Parameters for One-Sided Dynamic Principal Components via the Minimization of an Information Criterion*

Description

Computes One-Sided Dynamic Principal Components, choosing the number of components and lags automatically, to minimize an information criterion.

Usage

```
crit.odpc(
  Z,
  k_list = 1:5,
  max_num_comp = 5,
  ncores = 1,
  method,
  tol = 1e-04,
  niter_max = 500
)
```

Arguments

`Z`	Data matrix. Each column is a different time series.
`k_list`	List of values of k to choose from.
`max_num_comp`	Maximum possible number of components to compute.
`ncores`	Number of cores to use in parallel computations.
`method`	A string specifying the algorithm used. Options are 'ALS', 'mix' or 'gradient'. See details in `odpc`.
`tol`	Relative precision. Default is 1e-4.
`niter_max`	Integer. Maximum number of iterations. Default is 500.

Details

We apply the same stepwise approach taken in `cv.odpc`, but now to minimize an information criterion instead of the cross-validated forecasting error. The criterion is inspired by the IC_{p3} criterion proposed in Bai and Ng (2002). Let $\widehat{\sigma}^2_{1,k}$ be the reconstruction mean squared error for the first ODPC defined using k lags. Let $T^{*,1,k} = T - 2k$. Then we choose the value $k^{*,1}$ in `k_list` that minimizes

$$BNG_{1,k} = \log\left(\widehat{\sigma}^2_{1,k}\right) + (k+1)\frac{\log\left(\min(T^{*,1,k}, m)\right)}{\min(T^{*,1,k}, m)}.$$

Suppose now that `max_num_comp` ≥ 2 and we have computed $q-1$ dynamic principal components, $q - 1 <$ `max_num_comp`, each with $k^i_1 = k^i_2 = k^{*,i}$ lags, $i = 1, \ldots, q-1$. Let $\widehat{\sigma}^2_{q,k}$ be the reconstruction mean squared error for the fit obtained using q components, where the first $q-1$

components are defined using $k^{*,i}$, $i = 1, \ldots, q-1$ and the last component is defined using k lags. Let $T^{*,q,k} = T - \max\{2k^{*,1}, \ldots, 2k^{*,q-1}, 2k\}$. Let $k^{*,q}$ be the value in `k_list` that minimizes

$$BNG_{q,k} = \log\left(\widehat{\sigma}^2_{q,k}\right) + \left(\sum_{i=1}^{q-1}(k^{*,i}+1) + k + 1\right)\frac{\log\left(\min(T^{*,q,k}, m)\right)}{\min(T^{*,q,k}, m)}.$$

If $BNG_{q,k^{*,q}}$ is larger than $BNG_{q-1,k^{*,q-1}}$ we stop and the final model is the ODPC with $q-1$ components. Else we add the q-th component defined using $k^{*,q}$ and continue as before.

Value

An object of class odpcs, that is, a list of length equal to the number of computed components, each computed using the optimal value of k. The i-th entry of this list is an object of class `odpc`, that is, a list with entries

`f`	Coordinates of the i-th dynamic principal component corresponding to the periods $k_1 + 1, \ldots, T$.
`mse`	Mean squared error of the reconstruction using the first i components.
`k1`	Number of lags used to define the i-th dynamic principal component f.
`k2`	Number of lags of f used to reconstruct.
`alpha`	Vector of intercepts corresponding to f.
`a`	Vector that defines the i-th dynamic principal component
`B`	Matrix of loadings corresponding to f. Row number k is the vector of $k-1$ lag loadings.
`call`	The matched call.
`conv`	Logical. Did the iterations converge?

`components`, `fitted`, `plot` and `print` methods are available for this class.

References

Peña D., Smucler E. and Yohai V.J. (2017). "Forecasting Multiple Time Series with One-Sided Dynamic Principal Components." Available at https://arxiv.org/abs/1708.04705.

Bai J. and Ng S. (2002). "Determining the Number of Factors in Approximate Factor Models." Econometrica, 70(1), 191–221.

See Also

`odpc`, `cv.odpc`, `forecast.odpcs`

Examples

```
T <- 50 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
```

```
for (i in 1:m) {
  x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
# Choose parameters to perform a one step ahead forecast. Use 1 or 2 lags, only one component
fit <- crit.odpc(x, k_list = 1:2, max_num_comp = 1)
```

`crit.sparse_odpc` *Automatic Choice of Regularization Parameters for Sparse One-Sided Dynamic Principal Components using a BIC type criterion*

Description

Computes Sparse One-Sided Dynamic Principal Components, choosing the number of components and regularization parameters automatically, using a BIC type criterion.

Usage

```
crit.sparse_odpc(
  Z,
  k_list = 1:3,
  max_num_comp = 1,
  nlambda = 20,
  tol = 1e-04,
  niter_max = 500,
  eps = 0.001,
  ncores = 1
)
```

Arguments

`Z`	Data matrix. Each column is a different time series.
`k_list`	List of values of k to choose from.
`max_num_comp`	Maximum possible number of components to compute.
`nlambda`	Length of penalty sequence.
`tol`	Relative precision. Default is 1e-4.
`niter_max`	Integer. Maximum number of iterations. Default is 500.
`eps`	Between 0 and 1, used to build penalty sequence
`ncores`	Number of cores to use in parallel computations

Details

First `crit.odpc` is called to choose the number of lags and of components to use. Each component is then computed using a regularized version of the odpc objective function (see `odpc`), where the L1

norm of the $\mathbf{a}$ vector is penalized. The penalization parameter λ is chosen from a grid of candidates of size `nlambda`, seeking to minimize the following BIC type criterion

$$\log(MSE(\mathbf{a}_\lambda, \alpha_\lambda, \mathbf{B}_\lambda)) + \frac{\log(T^* m)}{T^* m} \|\mathbf{a}_\lambda\|_0,$$

where $\mathbf{a}_\lambda, \mathbf{B}_\lambda$ are the estimates associated with a given λ, m is the number of series and T^* is the number of periods being reconstructed.

Value

An object of class odpcs, that is, a list of length equal to the number of computed components, each computed using the optimal value of k. The i-th entry of this list is an object of class `odpc`, that is, a list with entries

`f`	Coordinates of the i-th dynamic principal component corresponding to the periods $k_1 + 1, \ldots, T$.
`mse`	Mean squared error of the reconstruction using the first i components.
`k1`	Number of lags used to define the i-th dynamic principal component f.
`k2`	Number of lags of f used to reconstruct.
`alpha`	Vector of intercepts corresponding to f.
`a`	Vector that defines the i-th dynamic principal component
`B`	Matrix of loadings corresponding to f. Row number k is the vector of $k-1$ lag loadings.
`call`	The matched call.
`conv`	Logical. Did the iterations converge?
`lambda`	Regularization parameter used for this component

`components`, `fitted`, `plot` and `print` methods are available for this class.

References

Peña D., Smucler E. and Yohai V.J. (2017). "Forecasting Multiple Time Series with One-Sided Dynamic Principal Components." Available at https://arxiv.org/abs/1708.04705.

See Also

`odpc`, `crit.odpc`, `forecast.odpcs`

Examples

```
T <- 50 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
  x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
fit <- crit.sparse_odpc(x, k_list = 1, ncores = 1)
```

cv.odpc *Automatic Choice of Tuning Parameters for One-Sided Dynamic Principal Components via Cross-Validation*

Description

Computes One-Sided Dynamic Principal Components, choosing the number of components and lags automatically, to minimize an estimate of the forecasting mean squared error.

Usage

```
cv.odpc(
  Z,
  h,
  k_list = 1:5,
  max_num_comp = 5,
  window_size,
  ncores_k = 1,
  ncores_w = 1,
  method,
  tol = 1e-04,
  niter_max = 500,
  train_tol = 0.01,
  train_niter_max = 100
)
```

Arguments

Z	Data matrix. Each column is a different time series.
h	Forecast horizon.
k_list	List of values of k to choose from.
max_num_comp	Maximum possible number of components to compute.
window_size	The size of the rolling window used to estimate the forecasting error.
ncores_k	Number of cores to parallelise over k_list.
ncores_w	Number of cores to parallelise over the rolling window (nested in k_list).
method	A string specifying the algorithm used. Options are 'ALS', 'mix' or 'gradient'. See details in odpc.
tol	Relative precision. Default is 1e-4.
niter_max	Integer. Maximum number of iterations. Default is 500.
train_tol	Relative precision used in cross-validation. Default is 1e-2.
train_niter_max	Integer. Maximum number of iterations used in cross-validation. Default is 100.

Details

We assume that for each component $k_1^i = k_2^i$, that is, the number of lags of $\mathbf{z}_t$ used to define the dynamic principal component and the number of lags of $\widehat{f}_t^i$ used to reconstruct the original series are the same. The number of components and lags is chosen to minimize the cross-validated forecasting error in a stepwise fashion. Suppose we want to make h-steps ahead forecasts. Let $w =$ `window_size`. Then given $k \in$ `k_list` we compute the first ODPC defined using k lags, using periods $1, \ldots, T - h - t + 1$ for $t = 1, \ldots, w$, and for each of these fits we compute an h-steps ahead forecast and the corresponding mean squared error $E_{t,h}$. The cross-validation estimate of the forecasting error is then

$$\widehat{MSE}_{1,k} = \frac{1}{w} \sum_{t=1}^{w} E_{t,h}.$$

We choose for the first component the value $k^{*,1}$ that minimizes $\widehat{MSE}_{1,k}$. Then, we fix the first component computed with $k^{*,1}$ lags and repeat the procedure with the second component. If the optimal cross-validated forecasting error using the two components, $\widehat{MSE}_{2,k^{*,2}}$ is larger than the one using only one component, $\widehat{MSE}_{1,k^{*,1}}$, we stop and output as a final model the ODPC computed using one component defined with $k^{*,1}$ lags; otherwise, if `max_num_comp` ≥ 2 we add the second component defined using $k^{*,2}$ lags and proceed as before.

This method can be computationally costly, especially for large values of the `window_size`. Ideally, the user should set `n_cores_k` equal to the length of `k_list` and `n_cores_w` equal to `window_size`; this would entail using `n_cores_k` times `n_cores_w` cores in total.

Value

An object of class odpcs, that is, a list of length equal to the number of computed components, each computed using the optimal value of k. The i-th entry of this list is an object of class `odpc`, that is, a list with entries

`f`	Coordinates of the i-th dynamic principal component corresponding to the periods $k_1 + 1, \ldots, T$.
`mse`	Mean squared error of the reconstruction using the first i components.
`k1`	Number of lags used to define the i-th dynamic principal component f.
`k2`	Number of lags of f used to reconstruct.
`alpha`	Vector of intercepts corresponding to f.
`a`	Vector that defines the i-th dynamic principal component
`B`	Matrix of loadings corresponding to f. Row number k is the vector of $k - 1$ lag loadings.
`call`	The matched call.
`conv`	Logical. Did the iterations converge?

`components`, `fitted`, `plot` and `print` methods are available for this class.

References

Peña D., Smucler E. and Yohai V.J. (2017). "Forecasting Multiple Time Series with One-Sided Dynamic Principal Components." Available at https://arxiv.org/abs/1708.04705.

See Also

odpc, crit.odpc, forecast.odpcs

Examples

```
T <- 50 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
  x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
# Choose parameters to perform a one step ahead forecast. Use 1 or 2 lags, only one component
# and a window size of 2 (artificially small to keep computation time low). Use two cores for the
# loop over k, two cores for the loop over the window
fit <- cv.odpc(x, h=1, k_list = 1:2, max_num_comp = 1, window_size = 2, ncores_k = 2, ncores_w = 2)
```

---

fitted.odpcs *Get Reconstructed Time Series From an odpcs Object*

Description

Get reconstructed time series from an odpcs object.

Usage

```
  ## S3 method for class 'odpcs'
fitted(object, num_comp = 1, ...)
```

Arguments

object	An object of class odpcs, usually the result of odpc.
num_comp	Integer indicating how many components to use for the reconstrucion. Default is 1.
...	Additional arguments for compatibility.

Value

A matrix that is the reconstruction of the original series.

See Also

odpc, crit.odpc, cv.odpc

Examples

```
T <- 200 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
 x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
fit <- odpc(x, ks = matrix(c(1, 1, 1, 0), 2, 2))
recons <- fitted(fit, num_comp = 2)
```

`forecast.odpcs` *Get Forecast From an odpcs Object*

Description

Get forecasts from an odpcs object.

Usage

```
  ## S3 method for class 'odpcs'
forecast(object, h, Z = NULL, add_residuals = FALSE, ...)
```

Arguments

object	An object of class odpcs, usually the result of odpc.
h	Integer. Number of periods for forecasting.
Z	Original data. Only used if add_residuals = TRUE.
add_residuals	Logical. Should the forecasts of the reconstruction residuals be added to the final forecast? Default is FALSE.
...	Additional arguments to be passed to auto.arima.

Details

Suppose q dynamic principal components were fitted to the data, each with (k_1^i, k_2^i) lags, $i = 1, \ldots, q$. Let $\widehat{\mathbf{f}}_T^i$ be the vector with the estimated values for the i-th dynamic principal component and $\widehat{\mathbf{B}}^i$, $\widehat{\boldsymbol{\alpha}}^i$ be the corresponding loadings and intercepts. Forecasts of the series are built by first fitting a SARIMA model to the components using auto.arima and getting their forecasts using forecast.Arima. Let $\widehat{f}_{T+h|T}^i$ for $h > 0$ be the forecast of f_{T+h}^i with information until time T. Then the h-steps ahead forecast of $\mathbf{z}_T$ is obtained as

$$\widehat{z}_{T+h|T,j} = \sum_{i=1}^{q} \left(\widehat{\alpha}_j^i + \sum_{v=0}^{k_2^i} \widehat{b}_{v,j}^i \widehat{f}_{T+h-v|T}^i \right) \quad j = 1, \ldots, m.$$

If add_residuals = TRUE, univariate SARIMA models are fitted to the residuals of the reconstruction, and their forecasts are added to the forecasts described above.

Value

A matrix that is the h-steps ahead forecast of the original series.

See Also

odpc, crit.odpc, cv.odpc, components_odpcs, auto.arima, forecast.Arima

Examples

```
T <- 201 #length of series
m <- 10 #number of series
set.seed(1234)
f <- matrix(0, 2 * T + 1, 1)
v <- rnorm(2 * T + 1)
f[1] <- rnorm(1)
theta <- 0.7
for (t in  2:(2 * T)){
  f[t] <- theta * f[t - 1] + v[t]
}
f <- f[T:(2 * T)]
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
  x[, i] <- sin(2 * pi * (i/m)) * f[1:T] + cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
fit <- odpc(x[1:(T - 1), ], ks = c(1))
forecasts <- forecast.odpcs(fit, h = 1)
mse <- mean((x[T, ] - forecasts)**2)
mse
```

---

odpc *Fitting of One-Sided Dynamic Principal Components*

Description

Computes One-Sided Dynamic Principal Components for a given number of lags.

Usage

```
odpc(Z, ks, method, tol = 1e-04, niter_max = 500)
```

Arguments

Z	Data matrix. Each column is a different time series.
ks	Matrix or vector of integers. If a matrix, each row is the vector with number of lags to use for each component. First column has the number of lags used to define the dynamic principal component (k_1), second column has the number of lags of the dynamic principal component used to reconstruct the series (k_2). If a vector, its entries are taken as both k_1 and k_2 for each component

`method`	A string specifying the algorithm used. Options are 'ALS', 'mix' or 'gradient'. See details below.
`tol`	Relative precision. Default is 1e-4.
`niter_max`	Integer. Maximum number of iterations. Default is 500.

Details

Consider the vector time series $\mathbf{z}_1, \ldots, \mathbf{z}_T$, where $\mathbf{z}_t = (z_{t,1}, \ldots, z_{t,m})'$. Let $\mathbf{a} = (\mathbf{a}_0', \ldots, \mathbf{a}_{k_1}')'$, where $\mathbf{a}_h' = (a_{h,1}, ..., a_{h,m})$, be a vector of dimension $m(k_1+1) \times 1$, let $\boldsymbol{\alpha}' = (\alpha_1, \ldots, \alpha_m)$ and $\mathbf{B}$ the matrix that has coefficients $b_{h,j}$ and dimension $(k_2+1) \times m$. Consider

$$f_t = \sum_{j=1}^{m} \sum_{h=0}^{k_1} a_{h,j} z_{t-h,j} \quad t = k_1+1, \ldots, T,$$

and suppose we use f_t and k_2 of its lags to reconstruct the series as

$$z_{t,j}^R(\mathbf{a}, \boldsymbol{\alpha}, \mathbf{B}) = \alpha_j + \sum_{h=0}^{k_2} b_{h,j} f_{t-h}.$$

Let

$$MSE(\mathbf{a}, \boldsymbol{\alpha}, \mathbf{B}) = \frac{1}{T-(k_1+k_2)} \sum_{j=1}^{m} \sum_{t=(k_1+k_2)+1}^{T} (z_{t,j} - z_{t,j}^R(\mathbf{a}, \boldsymbol{\alpha}, \mathbf{B}))^2$$

be the reconstruction MSE. The first one-sided dynamic principal component is defined as the series

$$\widehat{f_t} = \sum_{j=1}^{m} \sum_{h=0}^{k_1} \widehat{a}_{h,j} z_{t-h,j} \quad t = k_1+1, \ldots, T,$$

for optimal values $(\widehat{\mathbf{a}}, \widehat{\boldsymbol{\alpha}}, \widehat{\mathbf{B}})$ that satisfy

$$MSE(\widehat{\mathbf{a}}, \widehat{\boldsymbol{\alpha}}, \widehat{\mathbf{B}}) = \min_{\|\mathbf{a}\|=1, \boldsymbol{\alpha}, \mathbf{B}} MSE(\mathbf{a}, \boldsymbol{\alpha}, \mathbf{B}).$$

The second one-sided dynamic principal component is defined similarly, but now the residuals of the first one-sided dynamic principal component are to be reconstructed.

If method = 'ALS', an Alternating Least Squares type algorithm is used to compute the solution. If 'mix' is chosen, in each iteration Least Squares is used to compute the matrix of loadings and intercepts, but one iteration of Coordinate Descent is performed to compute the vector a that defines the dynamic principal component. If method = 'gradient', in each iteration Least Squares is used to compute the matrix of loadings and intercepts, but one iteration of Gradient Descent is performed to compute the vector a that defines the dynamic principal component. By default, 'ALS' is used when the number of series is less than 10, else 'gradient' is used.

Value

An object of class `odpcs`, that is, a list of length equal to the number of computed components. The i-th entry of this list is an object of class `odpc`, that is, a list with entries

f	Coordinates of the i-th dynamic principal component corresponding to the periods $k_1 + 1, \ldots, T$.
mse	Mean squared error of the reconstruction using the first i components.
k1	Number of lags used to define the i-th dynamic principal component f.
k2	Number of lags of f used to reconstruct.
alpha	Vector of intercepts corresponding to f.
a	Vector that defines the i-th dynamic principal component
B	Matrix of loadings corresponding to f. Row number k is the vector of $k-1$ lag loadings.
call	The matched call.
conv	Logical. Did the iterations converge?

components, fitted, plot and print methods are available for this class.

References

Peña D., Smucler E. and Yohai V.J. (2019). "Forecasting Multiple Time Series with One-Sided Dynamic Principal Components." Journal of the American Statistical Association.

See Also

crit.odpc, cv.odpc, plot.odpc, fitted.odpcs, components_odpcs, forecast.odpcs

Examples

```
T <- 200 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
   x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
fit <- odpc(x, ks = c(1))
fit
```

---

plot.odpc *Plot One-Sided Dynamic Principal Components*

Description

Plots an odpc object.

Usage

```
## S3 method for class 'odpc'
plot(x, which = 'Component', which_load = 0, ...)
```

Arguments

x	An object of class odpc, usually one of the entries of the result of odpc.
which	String. Indicates what to plot, either 'Component' or 'Loadings'. Default is 'Component'.
which_load	Lag number indicating which loadings should be plotted. Only used if which = 'Loadings'. Default is 0.
...	Additional arguments to be passed to the plotting functions.

See Also

odpc

Examples

```
T <- 200 #length of series
m <- 10 #number of series
set.seed(1234)
f <- rnorm(T + 1)
x <- matrix(0, T, m)
u <- matrix(rnorm(T * m), T, m)
for (i in 1:m) {
   x[, i] <- 10 * sin(2 * pi * (i/m)) * f[1:T] + 10 * cos(2 * pi * (i/m)) * f[2:(T + 1)] + u[, i]
}
fit <- odpc(x, ks = c(1))
plot(fit[[1]], xlab = '', ylab = '')
```

Index