# Package 'TVMVP'

June 27, 2025

**Type** Package

**Title** Time-Varying Minimum Variance Portfolio

**Version** 1.0.5

**Date** 2025-06-27

**Description** Provides the estimation of a time-dependent covariance matrix of returns with the intended use for portfolio optimization. The package offers methods for determining the optimal number of factors to be used in the covariance estimation, a hypothesis test of time-varying covariance, and user-friendly functions for portfolio optimization and rolling window evaluation. The local PCA method, method for determining the number of factors, and associated hypothesis test are based on Su and Wang (2017) <doi:10.1016/j.jeconom.2016.12.004>. The approach to time-varying portfolio optimization follows Fan et al. (2024) <doi:10.1016/j.jeconom.2022.08.007>. The regularisation applied to the residual covariance matrix adopts the technique introduced by Chen et al. (2019) <doi:10.1016/j.jeconom.2019.04.025>.

**License** MIT + file LICENSE

**URL** https://github.com/erilill/TV-MVP

**BugReports** https://github.com/erilill/TV-MVP/issues

**Encoding** UTF-8

**Depends** R (>= 3.6.0)

**Imports** R6, cli, prettyunits, dplyr, ggplot2, tidyr

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

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Erik Lillrank [aut, cre] (ORCID:
<https://orcid.org/0009-0001-3345-7694>),
Yukai Yang [aut] (ORCID: <https://orcid.org/0000-0002-2623-8549>)

**Maintainer** Erik Lillrank <erik.lillrank@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-06-27 15:50:06 UTC

# Contents



---

`TVMVP-package` *TVMVP: Time-Varying Minimum Variance Portfolio Optimization*

---

## Description

The TVMVP package provides tools for estimating time-dependent covariance matrices using kernel-weighted principal component analysis. These estimates can then be used for portfolio optimization in a dynamic setting.

## Details

The method involves five steps: (1) determining the number of factors, (2) estimating kernel-weighted PCA, (3) regularizing the idiosyncratic error covariance, (4) estimating the total covariance matrix, and (5) computing optimal portfolio weights.

An optional step includes a hypothesis test to check whether the covariance matrix is time-invariant.

The local PCA method, method for determining the number of factors, and associated hypothesis test are based on Su and Wang (2017). The approach to time-varying portfolio optimization follows Fan et al. (2024). The regularisation applied to the residual covariance matrix adopts the technique introduced by Chen et al. (2019).

The methodology implemented in this package closely follows Fan et al. (2024). The original authors provide a Matlab implementation at https://github.com/RuikeWu/TV-MVP.

The default kernel function in the package is the Epanechnikov kernel. Other kernel functions can also be used, however these are not implemented in the package. In order to do this, write an R function with an integrable kernel function and use this as input in the functions with argument `kernel_func`. It should be constructed as `custom_kernel <- function(u){...}`.

Similarly, the bandwidth function which is implemented in the package is the Silverman's rule of thumb. For most functions, simply set `bandwidth` to your preferred bandwidth, however for `expanding_tvmvp`, only Silverman's is implemented in this version of the package.

## Authors and Maintainer

Authors: Erik Lillrank and Yukai Yang
Maintainer: Erik Lillrank
Department of Statistics, Uppsala University
<erik.lillrank@gmail.com>, <yukai.yang@statistik.uu.se>

## Functions

`determine_factors` Selects the optimal number of factors via an information criterion.

`hyptest` Hypothesis test for time-invariant covariance matrices. Bootstrap p-values supported.

`predict_portfolio` Optimizes portfolio weights for out-of-sample prediction of portfolio performance.

`expanding_tvmvp` Evaluates MVP performance in a expanding window framework.

`time_varying_cov` Estimates the time-varying covariance matrix.

`silverman` Silverman's rule of thumb bandwidth formula.

## Class

`TVMVP` Time Varying Minimum Variance Portfolio (TVMVP) Class.

## References

Lillrank, E. (2025). Master's thesis (PDF in inst/doc)

Su, L., & Wang, X. (2017). On time-varying factor models: Estimation and testing. Journal of Econometrics, 198(1), 84–101.

Fan, Q., Wu, R., Yang, Y., & Zhong, W. (2024). Time-varying minimum variance portfolio. Journal of Econometrics, 239(2), 105339.

Chen, J., Li, D., & Linton, O. (2019). A new semiparametric estimation approach for large dynamic covariance matrices with multiple conditioning variables. Journal of Econometrics, 212(1), 155–176.

## Author(s)

**Maintainer**: Erik Lillrank <erik.lillrank@gmail.com> (ORCID)

Authors:

- Yukai Yang <yukai.yang@statistik.uu.se> (ORCID)

## See Also

Useful links:

- https://github.com/erilill/TV-MVP
- Report bugs at https://github.com/erilill/TV-MVP/issues

`comp_expected_returns` *Function to compute expected returns using a simple model selection approach*

## Description

Function to compute expected returns using a simple model selection approach

## Usage

```
comp_expected_returns(returns, horizon)
```

## Arguments

| | |
|---|---|
| `returns` | T times p matrix of returns |
| `horizon` | Length of forecasting horizon |

`determine_factors` *Determine the Optimal Number of Factors via an Information Criterion*

## Description

This function selects the optimal number of factors for a local principal component analysis (PCA) model of asset returns. It computes an BIC-type information criterion (IC) for each candidate number of factors, based on the sum of squared residuals (SSR) from the PCA reconstruction and a penalty term that increases with the number of factors. The optimal number of factors is chosen as the one that minimizes the IC. The procedure is available either as a stand-alone function or as a method in the 'TVMVP' R6 class.

## Usage

```
determine_factors(returns, max_m, bandwidth = silverman(returns))
```

## Arguments

| | |
|---|---|
| `returns` | A numeric matrix of asset returns with dimensions $T \times p$. |
| `max_m` | Integer. The maximum number of factors to consider. |
| `bandwidth` | Numeric. Kernel bandwidth for local PCA. Default is Silverman's rule of thumb. |

## Details

Two usage styles:

```
# Function interface
determine_factors(returns, max_m = 5)

# R6 method interface
tv <- TVMVP$new()
tv$set_data(returns)
tv$determine_factors(max_m = 5)
tv$get_optimal_m()
tv$get_IC_values()
```

When using the method form, if 'max_m' or 'bandwidth' are omitted, they default to values stored in the object. Results are cached and retrievable via class methods.

For each candidate number of factors $m$ (from 1 to `max_m`), the function:

1. Performs a local PCA on the returns at each time point $r = 1, \ldots, T$ using $m$ factors.
2. Computes a reconstruction of the returns and the corresponding residuals:

$$\text{Residual}_r = R_r - F_r \Lambda_r,$$

   where $R_r$ is the return at time $r$, and $F_r$ and $\Lambda_r$ are the local factors and loadings, respectively.
3. Computes the average sum of squared residuals (SSR) as:

$$V(m) = \frac{1}{pT} \sum_{r=1}^{T} \|\text{Residual}_r\|^2.$$

4. Adds a penalty term that increases with $R$:

$$\text{Penalty}(m) = m \frac{(p + T\text{bandwidth})}{(pT\text{bandwidth})} \log\left(\frac{pT\text{bandwidth}}{(p + T\text{bandwidth})}\right).$$

5. The information criterion is defined as:

$$\text{IC}(m) = \log\big(V(m)\big) + \text{Penalty}(m).$$

The optimal number of factors is then chosen as the value of $m$ that minimizes $\text{IC}(m)$.

## Value

A list with:

- `optimal_m`: Integer. The optimal number of factors.
- `IC_values`: Numeric vector of IC values for each candidate $m$.

## References

Su, L., & Wang, X. (2017). On time-varying factor models: Estimation and testing. Journal of Econometrics, 198(1), 84–101.

## Examples

```
set.seed(123)
returns <- matrix(rnorm(100 * 30), nrow = 100, ncol = 30)

# Function usage
result <- determine_factors(returns, max_m = 5)
print(result$optimal_m)
print(result$IC_values)

# R6 usage
tv <- TVMVP$new()
tv$set_data(returns)
tv$determine_factors(max_m = 5)
tv$get_optimal_m()
tv$get_IC_values()
```

---

# epanechnikov_kernel *Epanechnikov Kernel Function*

---

## Description

This function computes the value of the Epanechnikov kernel for a given input $u$. The Epanechnikov kernel is a popular choice in kernel density estimation due to its optimal properties in minimizing mean integrated squared error.

## Usage

```
epanechnikov_kernel(u)
```

## Arguments

| | |
|---|---|
| u | A numeric vector of points at which the kernel is evaluated. |

## Details

The Epanechnikov kernel is defined as:

$$K(u) = \begin{cases} \frac{3}{4}(1 - u^2) & \text{if } |u| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

This function applies the above formula to each element of the input vector u.

## Value

A numeric vector of kernel values corresponding to each input u.

## Examples

```
# Define a range of u values
u_values <- seq(-1.5, 1.5, by = 0.1)

# Compute Epanechnikov kernel values
kernel_values <- epanechnikov_kernel(u_values)
```

---

`expanding_tvmvp` *#' Expanding Window Time-Varying Minimum Variance Portfolio Optimization*

---

## Description

This function performs time-varying minimum variance portfolio (TV-MVP) optimization using time-varying covariance estimation based on Local Principal Component Analysis (Local PCA). The optimization is performed over a Expanding window, with periodic rebalancing. The procedure is available either as a stand-alone function or as a method in the ‘TVMVP‘ R6 class.

## Usage

```
expanding_tvmvp(
  obj,
  initial_window,
  rebal_period,
  max_factors,
  return_type = "daily",
  kernel_func = epanechnikov_kernel,
  rf = NULL,
  M0 = 10,
  rho_grid = seq(0.005, 2, length.out = 30),
  floor_value = 1e-12,
  epsilon2 = 1e-06
)
```

## Arguments

| | |
|---|---|
| `obj` | An object of class TVMVP with the data. |
| `initial_window` | An integer specifying the number of periods used in the initial estimation window. |
| `rebal_period` | An integer specifying the number of periods between portfolio rebalancing (e.g., weekly, monthly). |
| `max_factors` | An integer indicating the maximum number of latent factors to consider in the factor model. |
| `return_type` | A string indicating the return frequency. Options: ‘"daily"‘, ‘"weekly"‘, or ‘"monthly"‘. Used for annualization of evaluation metrics. |

| | |
|---|---|
| kernel_func | A kernel function to be used in the local PCA procedure. Default is 'epanechnikov_kernel'. |
| rf | A numeric vector or single value representing the risk-free rate. If 'NULL', it defaults to zero. |
| M0 | An integer specifying the number of observations to leave out between the two sub-samples for the adaptive thresholding of the residual covariance estimation. |
| rho_grid | A numeric sequence specifying the grid of rho values for threshold selection in covariance shrinkage. Default is 'seq(0.005, 2, length.out = 30)'. |
| floor_value | A small positive value to ensure numerical stability in the covariance matrix. Default is '1e-12'. |
| epsilon2 | A small positive value used in the adaptive thresholding of the residual covariance estimation. Default is '1e-6'. |

## Details

Two usage styles: #'

```
# Function interface
results <- expanding_tvmvp(
  obj = tv,
  initial_window = 50,
  rebal_period = 20,
  max_factors = 3,
  return_type = "daily",
  rf = NULL
)

# R6 method interface
tv <- TVMVP$new()
tv$set_data(returns)
results <- tv$expanding_tvmvp(
  initial_window = 50,
  rebal_period = 20,
  max_factors = 3,
  return_type = "daily",
  rf = NULL)
```

The function implements a Expanding time-varying PCA approach to estimate latent factor structures and uses a sparse residual covariance estimation method to improve covariance matrix estimation. The covariance matrix is used to determine the global minimum variance portfolio (MVP), which is rebalanced periodically according to the specified 'rebal_period'. The number of factors is determined by a BIC-type information criterion using the function 'determine_factors', updated yearly. The bandwidth is determined by Silverman's rule of thumb, updated each rebalancing period.

If 'rf' is 'NULL', the risk-free rate is assumed to be zero.

## Value

An R6 object of class `ExpandingWindow` with the following accessible elements:

`summary` A data frame of summary statistics for the TV-MVP and equal-weight portfolios, including cumulative excess return (CER), mean excess returns (MER), Sharpe ratio (SR), and standard deviation (SD) (raw and annualized).

`TVMVP` A list containing rebalancing dates, estimated portfolio weights, and excess returns for the TV-MVP strategy.

`Equal` A list with similar structure for the equal-weight portfolio.

## References

Lillrank, E. (2025). Master's thesis (PDF in inst/doc)

Fan, Q., Wu, R., Yang, Y., & Zhong, W. (2024). Time-varying minimum variance portfolio. Journal of Econometrics, 239(2), 105339.

## Examples

```
# Generate random returns for 20 assets over 100 periods
set.seed(123)
returns <- matrix(rnorm(20*100), nrow = 100, ncol = 20)

# Initialize object
tv <- TVMVP$new()
tv$set_data(returns)

# Run Expanding TV-MVP optimization
results <- expanding_tvmvp(
  obj = tv,
  initial_window = 50,
  rebal_period = 20,
  max_factors = 3,
  return_type = "daily",
  kernel_func = epanechnikov_kernel,
  rf = NULL
)

# R6 method interface
results <- tv$expanding_tvmvp(
  initial_window = 50,
  rebal_period = 20,
  max_factors = 3,
  return_type = "daily",
  rf = NULL)

# Print summary
print(results)

# Plot cumulative log excess returns
plot(results)
```

`get_object_size` *the function will return the size of obj and it is smart in the sense that it will choose the suitable unit*

## Description

the function will return the size of obj and it is smart in the sense that it will choose the suitable unit

## Usage

```
get_object_size(obj)
```

## Arguments

| | |
|---|---|
| `obj` | Object |

`hyptest` *Test for Time-Varying Covariance via Local PCA and Bootstrap*

## Description

This function performs a hypothesis test for time-varying covariance in asset returns based on Su and Wang (2017). It first standardizes the input returns and then computes a time-varying covariance estimator using a local principal component analysis (Local PCA) approach. The test statistic $J_{pT}$ is computed and its significance is assessed using a bootstrap procedure. The procedure is available either as a stand-alone function or as a method in the 'TVMVP' R6 class.

## Usage

```
hyptest(returns, m, B = 200, kernel_func = epanechnikov_kernel)
```

## Arguments

| | |
|---|---|
| `returns` | A numeric matrix of asset returns with dimensions $Tp$ (time periods by assets). |
| `m` | Integer. The number of factors to extract in the local PCA. See `determine_factors`. |
| `B` | Integer. The number of bootstrap replications to perform. Default is 200 |
| `kernel_func` | Function. A kernel function for weighting observations in the local PCA. Default is `epanechnikov_kernel`. |

## Details

Two usage styles:

```
# Function interface
hyptest(returns, m=2)

# R6 method interface
tv <- TVMVP$new()
tv$set_data(returns)
tv$determine_factors(max_m=5)
tv$hyptest()
tv
tv$get_bootstrap()         # prints bootstrap test statistics
```

When using the method form, if 'm' are omitted, they default to values stored in the object. Results are cached and retrievable via class methods.

The function follows the steps below:

1. Standardizes the returns.
2. Computes the optimal bandwidth using the Silverman rule.
3. Performs a local PCA on the standardized returns to extract local factors and loadings.
4. Computes a global factor model via singular value decomposition (SVD) to obtain global factors.
5. Calculates residuals by comparing the local PCA reconstruction to the standardized returns.
6. Computes a test statistic $J_{pT}$ based on a function of the residuals and covariance estimates as:

$$\hat{J}_{pT} = \frac{Tp^{1/2}h^{1/2}\hat{M} - \hat{\mathbb{B}}_{pT}}{\sqrt{\hat{\mathbb{V}}_{pT}}},$$

where:

$$\hat{M} = \frac{1}{pT}\sum_{i=1}^{p}\sum_{t=1}^{T}\left(\hat{\lambda}_{it}'\hat{F}_t - \tilde{\lambda}_{i0}'\tilde{F}_t\right),$$

$$\hat{\mathbb{B}}_{pT} = \frac{h^{1/2}}{T^2p^{1/2}}\sum_{i=1}^{p}\sum_{t=1}^{T}\sum_{s=1}^{T}\left(k_{h,st}\hat{F}_s'\hat{F}_t - \tilde{F}_s'\tilde{F}_t\right)^2\hat{e}_{is}^2,$$

and

$$\hat{\mathbb{V}}_{pT} = \frac{2}{phT^2}\sum_{1\le s\neq r\le T}\bar{k}_{sr}^2\left(\hat{F}_s'\hat{\Sigma}_F\hat{F}_r\right)^2\left(\hat{e}_r'\hat{e}_s\right)^2.$$

7. A bootstrap procedure is then used to compute the distribution of $J_{pT}$ and derive a p-value.

The function prints a message indicating the strength of evidence for time-varying covariance based on the p-value.

## Value

A list containing:

| | |
|---|---|
| J_NT | The test statistic $J_{pT}$ computed on the original data. |
| p_value | The bootstrap p-value, indicating the significance of time variation in covariance. |
| J_pT_bootstrap | A numeric vector of bootstrap test statistics from each replication. |

## References

Su, L., & Wang, X. (2017). On time-varying factor models: Estimation and testing. Journal of Econometrics, 198(1), 84–101

## Examples

```
# Simulate some random returns (e.g., 100 periods, 30 assets)
set.seed(123)
returns <- matrix(rnorm(100*30, mean = 0, sd = 0.02), nrow = 100, ncol = 30)

# Test for time-varying covariance using 3 factors and 10 bootstrap replications
test_result <- hyptest(returns, m = 3, B = 10, kernel_func = epanechnikov_kernel)

# Print test statistic and p-value
print(test_result$J_NT)
print(test_result$p_value)

# Or use R6 method interface
tv <- TVMVP$new()
tv$set_data(returns)
tv$determine_factors(max_m=5)
tv$hyptest(iB = 10, kernel_func = epanechnikov_kernel)
tv
tv$get_bootstrap()         # prints bootstrap test statistics
```

---

# localPCA *Perform Local PCA Over Time*

---

## Description

This function performs a local principal component analysis (PCA) on asset returns for each time period, aggregating the results over time. It calls an internal function `local_pca()` at each time index to extract local factors, loadings, and one-step-ahead factor estimates, and stores these results in lists. It uses previously computed factors to align the sign of the new factors.

## Usage

```
localPCA(returns, bandwidth, m, kernel_func = epanechnikov_kernel)
```

## Arguments

| | |
|---|---|
| `returns` | A numeric matrix of asset returns with dimensions $Tp$, where $T$ is the number of time periods and $p$ is the number of assets. |
| `bandwidth` | Numeric. The bandwidth parameter used in the kernel weighting for the local PCA. |
| `m` | Integer. The number of factors to extract. |
| `kernel_func` | Function. The kernel function used for weighting observations. Default is `epanechnikov_kernel`. |

## Details

The function processes the input returns over $T$ time periods by iteratively calling the `local_pca()` function. For each time $t_i$:

1. Kernel weights are computed using the specified `kernel_func` and `bandwidth`.
2. The returns are weighted by the square root of these kernel weights.
3. An eigen decomposition is performed on the weighted returns' covariance matrix to extract the top $m$ eigenvectors, which are scaled by sqrt(T) to form the local factors.
4. The signs of the new factors are aligned with those of the previous factors.
5. The factor loadings are computed by projecting the weighted returns onto the local factors, normalized by $T$.
6. A second pass computes a one-step-ahead factor estimate for the current time period.

## Value

A list with the following components:

- `factors`: A list of length $T$, where each element is a $Tm$ matrix of local factors.
- `loadings`: A list of length $T$, where each element is a $pm$ matrix of factor loadings.
- `m`: The number of factors extracted.
- `weights`: A list of length $T$, where each element is a vector of kernel weights used at that time point.
- `f_hat`: A $Tm$ matrix of one-step-ahead factor estimates.

## References

Su, L., & Wang, X. (2017). On time-varying factor models: Estimation and testing. Journal of Econometrics, 198(1), 84–101.

`predict_portfolio` *Predict Optimal Portfolio Weights Using Time-Varying Covariance Estimation*

## Description

This function estimates optimal portfolio weights using a time-varying covariance matrix derived from Local Principal Component Analysis (Local PCA). The procedure is available either as a stand-alone function or as a method in the 'TVMVP' R6 class. It computes the following portfolios:

1. Global Minimum Variance Portfolio (MVP)
2. Maximum Sharpe Ratio Portfolio (if `max_SR = TRUE`)
3. Return-Constrained Minimum Variance Portfolio (if `min_return` is provided)

## Usage

```
predict_portfolio(
  obj,
  horizon = 1,
  max_factors = 3,
  kernel_func = epanechnikov_kernel,
  min_return = NULL,
  max_SR = NULL,
  rf = NULL
)
```

## Arguments

| | |
|---|---|
| `obj` | An object of class TVMVP with the data. |
| `horizon` | Integer. Investment horizon over which expected return and risk are computed. Default is 1. |
| `max_factors` | Integer. The number of latent factors to consider in the Local PCA model. Default is 3. |
| `kernel_func` | Function. Kernel used for weighting observations in Local PCA. Default is `epanechnikov_kernel`. |
| `min_return` | Optional numeric. If provided, the function computes a Return-Constrained Portfolio that targets this minimum return. |
| `max_SR` | Logical. If TRUE, the Maximum Sharpe Ratio Portfolio is also computed. Default is `NULL`. |
| `rf` | Numeric. Log risk-free rate. If `NULL`, defaults to 0. |

## Details

Two usage styles:

#'

```
# R6 method interface
tv <- TVMVP$new()
tv$set_data(returns)
tv$determine_factors(max_m=5)
prediction <- tv$predict_portfolio(horizon = 1, min_return = 0.01, max_SR = TRUE)

#' # Function interface
prediction <- predict_portfolio(obj, horizon = 5, m = 2, min_return = 0.01, max_SR=TRUE)
```

The methods can then be used on `prediction` to retrieve the weights.

The function estimates a time-varying covariance matrix using Local PCA:

$$\hat{\Sigma}_{r,t} = \hat{\Lambda}_t \hat{\Sigma}_F \hat{\Lambda}_t' + \tilde{\Sigma}_e$$

Where $\hat{\Lambda}_t$ is the factor loadings at time t, $\hat{\Sigma}_F$ is the factor covariance matrix, and $\tilde{\Sigma}_e$ is regularized covariance matrix of the idiosyncratic errors.

It forecasts asset-level expected returns using a simple ARIMA model selection procedure and uses these in portfolio optimization. Optimization strategies include:

- Global minimum variance (analytical)
- Maximum Sharpe ratio (if `max_SR = TRUE`)
- Minimum variance with expected return constraint (if `min_return` is provided)

## Value

An object of class `PortfolioPredictions` (an R6 object) with:

`summary` A data frame of evaluation metrics (expected return, risk, Sharpe ratio) for all computed portfolios.

`MVP` A list containing the weights, expected return, risk, and Sharpe ratio for the Global Minimum Variance Portfolio.

`max_SR` (Optional) A list with metrics for the Maximum Sharpe Ratio Portfolio.

`MVPConstrained` (Optional) A list with metrics for the Return-Constrained Portfolio.

## Methods

The returned object includes:

- `$print()`: Nicely prints summary and portfolio access information.
- `$getWeights(method = c("MVP", "max_SR", "MVPConstrained"))`: Retrieves the weights for the selected portfolio.

## References

Lillrank, E. (2025). Master's thesis (PDF in inst/doc)

Fan, Q., Wu, R., Yang, Y., & Zhong, W. (2024). Time-varying minimum variance portfolio. Journal of Econometrics, 239(2), 105339.

## Examples

```
set.seed(123)
returns <- matrix(rnorm(200 * 20, mean = 0, sd = 0.02), ncol = 20)

# Initialize object
tv <- TVMVP$new()
tv$set_data(returns)

# Optimize weights and predict returns
result <- predict_portfolio(
  tv,
  horizon = 5,
  m = 3,
  min_return = 0.02,
  max_SR = TRUE
)

# Print the portfolio performance summary
print(result)

# Access MVP weights
result$getWeights("MVP")

# Access Max Sharpe weights (if computed)
result$getWeights("max_SR")

# Or use R6 method interface
tv$determine_factors(max_m=5)
prediction <- tv$predict_portfolio(horizon = 1, min_return)
prediction
prediction$getWeights("MVPConstrained")
```

---

`silverman` *Compute Bandwidth Parameter Using Silverman's Rule of Thumb*

---

## Description

This function calculates the bandwidth parameter for kernel functions using Silverman's rule of thumb, which is commonly used in kernel density estimation to determine an appropriate bandwidth. The procedure is available either as a stand-alone function or as a method in the 'TVMVP' R6 class.

## Usage

```
silverman(returns)
```

## Arguments

| | |
|---|---|
| returns | A numeric matrix of asset returns with $T$ rows (time periods) and $p$ columns (assets). |

## Details

Silverman's rule of thumb for bandwidth selection is given by:

$$bandwidth = \frac{2.35}{\sqrt{12}} \times T^{-0.2} \times p^{-0.1}$$

where $T$ is the number of time periods and $p$ is the number of assets.

Two usage styles:

```
# Function interface
bw <- silverman(returns)

# R6 method interface
tv <- TVMVP$new()
tv$set_data(returns)
tv$silverman()
```

## Value

A numeric value representing the computed bandwidth parameter based on Silverman's rule.

## Examples

```
# Simulate data for 50 assets over 200 time periods
set.seed(123)
T <- 200
p <- 50
returns <- matrix(rnorm(T * p, mean = 0.001, sd = 0.02), ncol = p)

# Compute bandwidth using Silverman's rule of thumb
bw <- silverman(returns)
print(bw)

tv <- TVMVP$new()
tv$set_data(returns)
tv$silverman()
```

`time_varying_cov` *Estimate Time-Varying Covariance Matrix Using Local PCA*

## Description

This function estimates a time-varying covariance matrix using local principal component analysis and the soft thresholding for residual shrinkage. By default, only the total covariance matrix is returned. Optionally, the user can retrieve all intermediate components of the estimation process. The procedure is available either as a stand-alone function or as a method in the 'TVMVP' R6 class.

## Usage

```
time_varying_cov(
  obj,
  max_factors = 3,
  kernel_func = epanechnikov_kernel,
  M0 = 10,
  rho_grid = seq(0.005, 2, length.out = 30),
  floor_value = 1e-12,
  epsilon2 = 1e-06,
  full_output = FALSE
)
```

## Arguments

| | |
|---|---|
| `obj` | An object of class TVMVP with the data. |
| `max_factors` | The number of factors to use in local PCA. |
| `kernel_func` | The kernel function to use (default is `epanechnikov_kernel`). |
| `M0` | Integer. The number of observations to leave out between the two sub-samples in the adaptive thresholding procedure. Default is 10. |
| `rho_grid` | A numeric vector of candidate shrinkage parameters $\rho$ used in `adaptive_poet_rho`. Default is `seq(0.005, 2, length.out = 30)`. |
| `floor_value` | A small positive number specifying the lower bound for eigenvalues in the final positive semidefinite repair. Default is `1e-12`. |
| `epsilon2` | A small positive tuning parameter for the adaptive thresholding. Default is `1e-6`. |
| `full_output` | Logical; if `TRUE`, returns all components of the estimation. |

## Details

The function estimates a time-varying covariance matrix using Local PCA:

$$\hat{\Sigma}_{r,t} = \hat{\Lambda}_t \hat{\Sigma}_F \hat{\Lambda}_t' + \tilde{\Sigma}_e$$

Where $\hat{\Lambda}_t$ is the factor loadings at time t, $\hat{\Sigma}_F$ is the factor covariance matrix, and $\tilde{\Sigma}_e$ is regularized covariance matrix of the idiosyncratic errors.

Two usage styles:

```
# Function interface
tv <- TVMVP$new()
tv$set_data(returns)
cov <- time_varying_cov(tv, m = 5)

# R6 method interface
tv$determine_factors(max_m = 5)
cov <- tv$time_varying_cov()
```

## Value

By default, a covariance matrix. If `full_output = TRUE`, a list containing:

- `total_cov` – the estimated covariance matrix,
- `residual_cov` – the residual (idiosyncratic) covariance,
- `loadings` – estimated factor loadings,
- `best_rho` – optimal shrinkage parameter,
- `naive_resid_cov` – residual covariance before shrinkage

## References

Lillrank, E. (2025). Master's thesis (PDF in inst/doc)

Chen, J., Li, D., & Linton, O. (2019). A new semiparametric estimation approach for large dynamic covariance matrices with multiple conditioning variables. Journal of Econometrics, 212(1), 155–176.

Fan, Q., Wu, R., Yang, Y., & Zhong, W. (2024). Time-varying minimum variance portfolio. Journal of Econometrics, 239(2), 105339.

## Examples

```
set.seed(123)
returns <- matrix(rnorm(100 * 30), nrow = 100, ncol = 30)

# Initialize object
tv <- TVMVP$new()
tv$set_data(returns)

# Using function interface
time_varying_cov(obj = tv, m = 3)

# Or using R6 method
tv$time_varying_cov(m=3)
```

TVMVP *Time Varying Minimum Variance Portfolio (TVMVP) Class*

## Description

This class implements a time-varying minimum variance portfolio using locally smoothed principal component analysis (PCA) to estimate the time-dependent covariance matrix.

This class provides a flexible interface to:

- Set return data (`$set_data()`)
- Determine optimal number of factors (`$determine_factors()`)
- Conduct test of constant factor loadings (`$hyptest()`)
- Time-dependent covariance estimation (`$time_varying_cov()`)
- Portfolio optimization (`$predict_portfolio()`)
- Expanding window evaluation (`$expanding_tvmvp()`)
- Extract cached results (`$get_optimal_m()`, `$get_IC_values()`, `$get_bootstrap()`)

Looking for package description? See TVMVP-package.

## Usage

```
# Initial object of class TVMVP
tv <- TVMVP$new()

# Set data
tv$set_data(returns) # Returns must be T times p matrix

# Determine number of factors
tv$determine_factors(max_m=10)

# Test for constant loadings
tv$hyptest()

# Estimate time-dependent covariance matrix
cov <- tv$time_varying_cov()

# Evaluate TVMVP performance on historical data
mvp_results <- tv$expanding_tvmvp(
                 initial_window = 60,
                 rebal_period   = 5,
                 max_factors    = 10,
                 return_type    = "daily")

# Make out-of-sample prediction and compute weights
predictions <- tv$predict_portfolio(horizon=5,
```

```
                                    min_return = 0.01,
                                    max_SR = TRUE)
# Extract weights
predictions$getWeights("MVP")
#'
```

## Arguments

**data** $Tp$ (time periods by assets) matrix of returns.

**bandwidth** Numerical. Bandwidth parameter used for local smoothing in the local PCA

**max_m** Integer. Maximum number of factors to be tested when determining the optimal number of factors.

**optimal_m** The optimal number of factors to use in covariance estimation.

## Methods

`$new(data = NULL)` Initialize object of class TVMVP. Optionally pass returns matrix.

`$set_data(data)` Set the data. Must be $Tp$ (time periods by assets) matrix.

`$get_data()` Get the data.

`$set()` Manually set arguments of the object.

`$determine_factors()` Determines optimal number of factors based on BIC-type information criterion.

`$get_optimal_m{}` Prints optimal number of factors, `optimal_m`.

`$get_IC_values()` Prints IC-values for the different number of factors tested using `determine_factors`.

`$hyptest()` Hypothesis test of constant loadings.

`$get_bootstrap()` Prints bootstrap test statistics from the hypothesis test.

`$predict_portfolio()` Optimizes portfolio weights for out-of-sample prediction of portfolio performance.

`$expanding_tvmvp()` Evaluates MVP performance in a expanding window framework.

`$time_varying_cov()` Estimates the time-varying covariance matrix.

`$silverman()` Silverman's rule of thumb bandwidth formula.

# Index