Package: IndexConstruction (via r-universe)

October 29, 2026

Type Package

Title Index Construction for Time Series Data

Version 0.2-1

Date 2026-09-09

LazyLoad yes

LazyData true

Depends R (>= 2.10)

Imports KernSmooth, fGarch, lubridate, xts, RcppBDT, zoo

Description Derivation of indexes for benchmarking purposes. A methodology with flexible number of constituents is implemented. Also functions for market capitalization and volume weighted indexes with fixed number of constituents are available. The main function of the package, indexComp(), provides the derived index, suitable for analysis purposes. The functions indexUpdate(), indexMemberSelection() and indexMembersUpdate() are components of indexComp() and enable one to construct and continuously update an index, e.g. for display on a website. The methodology behind the functions provided gets introduced in Trimborn and Haerdle (2018) <doi:10.1016/j.jempfin.2018.08.004>.

License GPL (>=3)

RoxygenNote 7.3.2

Repository https://simontrimborn.r-universe.dev

Date/Publication 2026-09-29 12:37:17 UTC

RemoteUrl https://github.com/simontrimborn/indexconstruction

RemoteRef HEAD

RemoteSha 4ee5a8a52c95ffcce115fae1c16c89f04b90328c

Contents

indexComp 2

indexMemberSelection 5
indexMembersUpdate 7
indexUpdate 8
market 9
price 9
relativeWeights 10
switchDates 10
vol 11

Index **13**

`indexComp` *Index derivation for price and liquidity indices*

Description

`indexComp` derives an Index from the given price and market capitalization or liquidity data. The number of constituents can be fixed or being chosen flexible based on the methodology from Trimborn and Haerdle (2018). This is the main function of the package. The derived index is meant for analysis purposes. For a continuous updating and display of an index on a website, please refer to the remaining functions.

Usage

```
indexComp(market, price, vol = NULL, weighting = "market", weighting.all = "market",
ic = "AIC", eval.seq = c("sequential", "all.together"),
optimum = c("local", "global"), start.const = 1, steps = 1, fixed.value = NULL,
base.value = 1000, derivation.period = 1, derivation.period.ic = 3, days.line)
```

Arguments

`market`	An xts object with the market capitalization data. The default is `NULL`, an entry is necessary if weighting is set to `"market"`. If `"equal"` is chosen, the market cap will not be used in the calculation.
`price`	An xts object with the price data. An entry is always required.
`vol`	An xts object with the trading volume (liquidity) data. The default is `NULL`, an entry is necessary if weighting is set to `"volume"`.
`weighting`	The weighting scheme to be applied. `"market"` refers to weighting by market capitalization, `"volume"` refers to weighting by trading volume, and `"equal"` refers to equal weighting.
`weighting.all`	The weighting scheme to be applied to the full market index. `"market"` refers to weighting by market capitalization, `"volume"` refers to weighting by trading volume, and `"equal"` refers to equal weighting.
`ic`	Information Criterion to be used for the evaluation of the appropriate index to be used. Possible entries are `"AIC"`, `"GCV"`, `"GFCV"`, `"SH"`, `"Cp"` and `"FPE"`.

`eval.seq` Indicates how the evaluation of the candidate indices by the ic shall be performed. `"all.together"` evaluates all indices against each other, `"sequential"` evaluates always two consecutive indices against each other.

`optimum` Define how to choose the optimal index. Either a `"local"` optimum is chosen, thus the derivation stops the first time the results become worse under the chosen `ic`, or a `"global"` optimum is chosen, thus all indices are derived and the best fitting one under the `ic` is chosen.

`start.const` The number of constituents to start constructing the indices with. The default is `1`.

`steps` The step width for the number of constituents to construct the next index from. The default is `1`.

`fixed.value` In case no `ic` for the number of constituents for the index shall be applied, give the number of constituents the index shall contain. In that case, `"ic"`, `"eval.seq"`, `"optimum"`, `"start.const"` and `"steps"` are inactive parameters. The default is `NULL`.

`base.value` The starting value for the index. The default is `1000`.

`derivation.period` The number of month after which the weights of the index are reallocated. The default is `1`.

`derivation.period.ic` The number of month after which the composition of the index is derived again, thus the number of constituents is reevaluated. The default is `3`.

`days.line` The days of the month to perform the recalculation on. Can be calculated from switchDates.

Details

For equal weighting, the order is not decided by largest market capitalization or largest trading volume. Instead, the order in which assets are added to the computation is by largest contribution to the R-squared on the last periods data. For a large number of assets, this computation can take a while during the periodic recomputation. In case two or more assets have similar R-square, the first one in the dataset with respective R-square contribution is added.

For more details, please see the methodology section of the paper Trimborn and Haerdle (2018).

Value

An object of the `class` IndexConstruction with the components

`results` A list containing the results of the model fitting

`index` The optimal index

`totalIndex` The index of all constituents

`totalIndexRebased` The index of all constituents rebased at the index each time after altering the number of index constituents which is useful for comparisons with the market

`assets` A list containing the assets considered for index construction in each period

`weights` A list containing the weights assigned to the selected index constituents in each period

`weightsRelative` A list containing the relative weights assigned to the selected index constituents in each period

`inputs` A list containing the inputs for model fitting

`marketCap` The provided dataset of the market capitalization of each asset for index construction

`price` The provided dataset of the price series of each asset for index construction

`tradingVolume` The provided dataset of the trading volume of each asset for index construction

`daysDerivation` The provided vector of dates on which to rederive the index weights and number of index constituents

`weighting` The selected weighting scheme

`weighting.all` The selected weighting.all scheme

`ic` The selected ic

`eval.seq` The selected eval.seq scheme

`optimum` The selected optimization scheme

`start.const` The selected number of starting constituents for the index

`steps` The selected step size for the selection of the constituents for the index

`derivation.period`
The selected period for rederivation of the weights of the index constituents

`derivation.period.ic`
The selected period for rederivation of the number of index constituents

References

Trimborn, S. and Haerdle, W.K. (2018). CRIX an Index for cryptocurrencies, *Journal of Empirical Finance* 49, pp. 107-122. doi:10.1016/j.jempfin.2018.08.004

Examples

```
data(CryptoData)

price = price["2014-03-31::2015-01-31"]
market = market["2014-03-31::2015-01-31"]
vol = vol["2014-03-31::2015-01-31"]
days.line = switchDates(price, specificDate = "1")

indexComp(market = market, price = price, vol = vol, weighting = "market",
weighting.all = "market", ic = "AIC", eval.seq = "sequential", optimum = "local",
start.const = 5, steps = 5, days.line = days.line)
```

indexMemberSelection *Number of Index Members Derivation*

Description

indexMemberSelection derives the number of index members for the coming period based on an Information Criterion, e.g. AIC. The methodology is according to Trimborn and Haerdle (2018). The method derives the new weights according to the specifications of the weight reevaluation. The function expects the data period provided to be twice the number of months specified in derivation.period.ic. In case of a mismatch, a warning is given. This function is meant for continuous updating and display of an index on a website. For the derivation of an index for analysis purposes, please refer to the function "indexComp".

Usage

```
indexMemberSelection(market, price, vol, weighting = "market",
weighting.all = "market", ic = "AIC", eval.seq = c("sequential", "all.together"),
optimum = c("local", "global"), start.const = 1, steps = 1, fixed.value = NULL,
derivation.period = 1, derivation.period.ic = 3, base.value = 1000, days.line)
```

Arguments

market	An xts object with the market capitalization data. The default is NULL, an entry is necessary if weighting is set to "market".
price	An xts object with the price data. An entry is always required.
vol	An xts object with the trading volume (liquidity) data. The default is NULL, an entry is necessary if weighting is set to "volume".
weighting	The weighting scheme to be applied. "market" refers to weighting by market capitalization, "volume" refers to weighting by trading volume.
weighting.all	The weighting scheme to be applied to the full market index. "market" refers to weighting by market capitalization, "volume" refers to weighting by trading volume.
ic	Information Criterion to be used for the evaluation of the appropriate index to be used. Possible entries are "AIC", "GCV", "GFCV", "SH", "Cp" and "FPE".
eval.seq	Indicates how the evaluation of the candidate indices by the ic shall be performed. "all.together" evaluates all indices against each other, "sequential" evaluates always two consecutive indices against each other.
optimum	Define how to choose the optimal index. Either a "local" optimum is chosen, thus the derivation stops the first time the results become worse under the chosen ic, or a "global" optimum is chosen, thus all indices are derived and the best fitting one under the ic is chosen.
start.const	The number of constituents to start constructing the indices with. The default is 1.
steps	The step width for the number of constituents to construct the next index from. The default is 1.

`fixed.value` In case no ic for the number of constituents for the index shall be applied, give the number of constituents the index shall contain. In that case, `"ic"`, `"eval.seq"`, `"optimum"`, `"start.const"` and `"steps"` are inactive parameters. The default is `NULL`.

`base.value` The starting value for the index. The default is `1000`.

`derivation.period` The number of month after which the weights of the index are reallocated. The default is `1`.

`derivation.period.ic` The number of month after which the composition of the index is derived again, thus the number of constituents is reevaluated. The default is `3`.

`days.line` The days of the month to perform the recalculation on. Can be calculated from SwitchDates.

Details

`indexMemberSelection` derives the number of index members for the coming period based on an Information Criterion, e.g. AIC. The methodology is according to Trimborn and Haerdle (2018). The method derives the new weights according to the specifications of the weight reevaluation. The function expects the data period provided to be twice the number of months specified in derivation.period.ic. In case of a mismatch, a warning is given. The data from the first period are used to derived the likelihood, the second period is used for out-of-sample derivation of the number of constituents. Hence for a 3 month reevaluation period, 6 month of data are required by this function. For more details, please see the methodology section of the paper Trimborn and Haerdle (2018).

Value

Returns the number of index members for application in the next period.

References

Trimborn, S. and Haerdle, W.K. (2018). CRIX an Index for cryptocurrencies, *Journal of Empirical Finance* 49, pp. 107-122. doi:10.1016/j.jempfin.2018.08.004

Examples

```
data(CryptoData)

price = price["2016-07-31::2017-01-31"]
market = market["2016-07-31::2017-01-31"]
vol = vol["2016-07-31::2017-01-31"]
days.line = switchDates(price, specificDate = "1")

indexMemberSelection(market = market, price = price, vol = vol,
weighting = "market", weighting.all = "market", ic = "AIC", eval.seq = "sequential",
optimum = "local", start.const = 5, steps = 5, days.line = days.line)
```

`indexMembersUpdate` *Reevaluation of Index constituents weights*

Description

`indexMembersUpdate` derives the new weights for the coming period. The methodology is according to Trimborn and Haerdle (2018). The method derives the new weights over the data period provided. The data input defines the length of the period, hence it can be different from full month. This function is meant for continuous updating and display of an index on a website. For the derivation of an index for analysis purposes, please refer to the function `"indexComp"`.

Usage

```
indexMembersUpdate(market, price, vol, weighting, index.const, last.value)
```

Arguments

`market`	An xts object with the market capitalization data. The default is `NULL`, an entry is necessary if weighting is set to `"market"`.
`price`	An xts object with the price data. An entry is always required.
`vol`	An xts object with the trading volume (liquidity) data. The default is `NULL`, an entry is necessary if weighting is set to `"volume"`.
`weighting`	The weighting scheme to be applied. `"market"` refers to weighting by market capitalization, `"volume"` refers to weighting by trading volume.
`index.const`	Number of Index constituents. The number can be derived from `indexComp`, `indexMemberSelection` or be chosen by alternative means.
`last.value`	The last index value before rederivation.

Details

`indexMembersUpdate` derives the new weights for the coming period. The methodology is according to Trimborn and Haerdle (2018). The method derives the new weights over the data period provided. The data input defines the length of the period, hence it can be different from full month. For more details, please see the methodology section of the paper Trimborn and Haerdle (2018).

Value

A list, entry 1 is the ordered names of index members, entry 2 the respective consideration of the index constituents, entry 3 the weights of the index members which gives multiplied with entry 2 the actual weight and entry 4 the new divisor of the index.

References

Trimborn, S. and Haerdle, W.K. (2018). CRIX an Index for cryptocurrencies, *Journal of Empirical Finance* 49, pp. 107-122. doi:10.1016/j.jempfin.2018.08.004

Examples

```
data(CryptoData)

price = price["2017-01-01::2017-01-31"]
market = market["2017-01-01::2017-01-31"]
vol = vol["2017-01-01::2017-01-31"]
indexMembersUpdate(market = market, price = price, vol = vol,
weighting = "market", index.const = 5, last.value = 1000)
```

---

indexUpdate *Updating an existing index with new index values*

Description

`indexUpdate` derives the next values of an Index from the given price, weights and its divisor. This function is meant for continuous updating and display of an index on a website. For the derivation of an index for analysis purposes, please refer to the function `"indexComp"`.

Usage

```
indexUpdate(price, index.weights, divisor)
```

Arguments

`price`	An xts object with the price data. An entry is always required.
`index.weights`	A vector with the absolute weights expressed as number of shares of each asset. The weights are provided by `indexComp`. They can be also easily derived from the market capitalization by dividing with the respective price.
`divisor`	The divisor required for the index derivation. The divisor is provided by `indexComp`. For details on its derivation, see Trimborn and Haerdle (2018).

Details

For more details, please see the methodology section of the paper Trimborn and Haerdle (2018).

Value

The next value(s) of the Index.

References

Trimborn, S. and Haerdle, W.K. (2018). CRIX an Index for cryptocurrencies, *Journal of Empirical Finance* 49, pp. 107-122. doi:10.1016/j.jempfin.2018.08.004

Examples

```
data(CryptoData)

const.names = c("btc", "eth", "xrp", "ltc", "xmr")
index.weights = c(16136712, 88440036, 36856524148, 49589181, 13859864)
divisor = 17185084

indexUpdate(price["2017-02-01", const.names], index.weights = index.weights, divisor = divisor)
```

---

market *Market capitalization data for Cryptocurrencies.*

Description

The dataset contains market capitalization information for cryptocurrencies.

Usage

```
data(CryptoData)
```

Format

A dataset with a xts matrix. Load the R library xts for proper visualization of the dataset.

Source

The dataset was provided by CoinGecko. Up-to-date data are accessible via https://www.coingecko.com/en/api.

---

price *Pricing data for Cryptocurrencies.*

Description

The dataset contains pricing information for cryptocurrencies.

Usage

```
data(CryptoData)
```

Format

A dataset with a xts matrix. Load the R library xts for proper visualization of the dataset.

Source

The dataset was provided by CoinGecko. Up-to-date data are accessible via https://www.coingecko.com/en/api.

relativeWeights *Retrieving the relative weights of the assets in the index*

Description

`relativeWeights` retrieves the relative weights of the assets in the index from the absolute weights expressed in shares of the assets. The latter is a direct output of `indexComp`.

Usage

```
relativeWeights(price, index.weights)
```

Arguments

`price`	An xts object with the price data. An entry is always required.
`index.weights`	A vector with the absolute weights expressed as number of shares of each asset. The weights are provided by `indexComp`. They can be also easily derived from the market capitalization by dividing with the respective price.

Value

The relative weights of the assets in the Index.

References

Trimborn, S. and Haerdle, W.K. (2018). CRIX an Index for cryptocurrencies, *Journal of Empirical Finance* 49, pp. 107-122. doi:10.1016/j.jempfin.2018.08.004

Examples

```
data(CryptoData)

const.names = c("btc", "eth", "xrp", "ltc", "xmr")
index.weights = c(16136712, 88440036, 36856524148, 49589181, 13859864)

relativeWeights(price = price["2017-02-01", const.names], index.weights = index.weights)
```

switchDates *Deriving the dates on which the index constituents are going to be reevaluated*

Description

`switchDates` derives the dates on which the index constituents are going to be reevaluated.

Usage

```
switchDates(price, specificDate = NULL, WeekDay = NULL, Appearance = 1)
```

Arguments

price	An xts object with the price data. An entry is always required.
specificDate	A specific date of each month on which the index members get reevaluated. A common date would be the 1st of each month or the 15th of each month. specificDate is dominating WeekDay.
WeekDay	Only active when specificDate is NULL. A specific weekday of each month on which the index members get reevaluated. The input has to be a character describing the weekday in English. By default the first weekday with this appearancce is returned. The argument Appearance defines if it is the 1st, 2nd or another appearance of this weekday. E.g. the 3rd Friday of each month can be returned.
Appearance	Defines if the 1st, 2nd or another appearance of a weekday gets returned. E.g. the 3rd Friday of each month can be returned. Only active when specificDate is NULL. The argument works in combination with WeekDay.

Value

A vector of class date with the respective dates on which the index members become reevaluated. This is a necessary input to IndexComp.

References

Trimborn, S. and Haerdle, W.K. (2018). CRIX an Index for cryptocurrencies, *Journal of Empirical Finance* 49, pp. 107-122. doi:10.1016/j.jempfin.2018.08.004

Examples

```
data(CryptoData)

switchDates(price, specificDate = "1")
```

---

vol *Volume data for Cryptocurrencies.*

Description

The dataset contains trading volume information for cryptocurrencies.

Usage

```
data(CryptoData)
```

Format

A dataset with a xts matrix. Load the R library xts for proper visualization of the dataset.

Source

The dataset was provided by CoinGecko. Up-to-date data are accessible via https://www.coingecko.com/en/api.

Index

indexComp, 2
indexMemberSelection, 5
indexMembersUpdate, 7
indexUpdate, 8

market, 9

price, 9

relativeWeights, 10

switchDates, 10

vol, 11