Package ‘termstrc’

February 20, 2015

Type Package
Title Zero-coupon Yield Curve Estimation
Version 1.3.7
Date 2013-11-03
Author Robert Ferstl, Josef Hayden
Maintainer Josef Hayden <josef.hayden@gmail.com>
Description The package offers a wide range of functions for term structure estimation based on static and dynamic coupon bond and yield data sets. The implementation focuses on the cubic splines approach of McCulloch (1971, 1975) and the Nelson and Siegel (1987) method with extensions by Svensson (1994), Diebold and Li (2006) and De Pooter (2007). We propose a weighted constrained optimization procedure with analytical gradients and a globally optimal start parameter search algorithm. Extensive summary statistics and plots are provided to compare the results of the different estimation methods. Several demos are available using data from European government bonds and yields.
URL http://R-Forge.R-project.org/projects/termstrc/
LazyLoad yes
Depends R (>= 3.0.0)
Imports lmtest, Rcpp (>= 0.10.6), rgl, sandwich, urca, zoo
LinkingTo Rcpp
License GPL (>= 2)
Repository CRAN
Repository/R-Forge/Project termstrc
Repository/R-Forge/Revision 637
Repository/R-Forge/DateTimeStamp 2013-11-03 18:53:03
Date/Publication 2013-11-04 20:20:11
NeedsonCompilation yes
R topics documented:

termstrc-package .............................................. 4
aabse .......................................................... 5
bond_prices ..................................................... 6
bond_yields .................................................... 7
create_cashflows_matrix ....................................... 7
create_maturities_matrix ...................................... 8
cSums .......................................................... 9
datadyncouponbonds ............................................. 10
duration ......................................................... 10
estimateyieldcurve ............................................ 11
estimatezcyieldcurve ......................................... 12
estim_cs ........................................................ 12
estim_cs.couponbonds ......................................... 13
estim_nss ........................................................ 15
estim_nss.couponbonds ......................................... 16
estim_nss.dyncouponbonds ..................................... 18
estim_nss.zeroyields .......................................... 19
fcontrib ........................................................ 21
fcontrib.dyntermstrc_param ................................... 21
findstartparambonds .......................................... 22
findstartparamyields .......................................... 23
forwardrates ................................................... 23
fwr_asv ......................................................... 24
fwr_dl .......................................................... 25
fwr_ns .......................................................... 26
fwr_sv .......................................................... 27
get_constraints ................................................ 28
get_grad_objfct ............................................... 28
get_grad_objfct_bonds ........................................ 28
get_objfct ...................................................... 29
get_objfct_bonds .............................................. 29
get_paramnames ................................................ 30
get_realnames ................................................ 30
gi .............................................................. 30
govbonds ....................................................... 31
grad_asv ....................................................... 32
grad_asv_bonds ............................................... 32
grad_asv_bonds_grid .......................................... 33
grad_asv_grid ................................................ 34
grad_dl ........................................................ 34
grad_dl_bonds ................................................ 35
grad_ns ........................................................ 35
grad_ns_bonds ................................................ 36
grad_ns_bonds_grid .......................................... 36
grad_ns_grid .................................................. 37
grad_sv ........................................................ 37
R topics documented:

grad_sv_bonds .................................................. 38
grad_sv_bonds_grid ............................................. 38
grad_sv_grid ..................................................... 39
impl_fwr .......................................................... 39
loss_function ..................................................... 40
maturity_range ................................................... 40
objfct_asv ........................................................ 41
objfct_asv_bonds ................................................ 41
objfct_asv_bonds_grid ......................................... 42
objfct_asv_grid ................................................... 42
objfct_dl .......................................................... 43
objfct_dl_bonds .................................................. 43
objfct_ns .......................................................... 44
objfct_ns_bonds .................................................. 44
objfct_ns_bonds_grid .......................................... 45
objfct_ns_grid .................................................... 45
objfct_sv ........................................................ 46
objfct_sv_bonds .................................................. 46
objfct_sv_bonds_grid .......................................... 47
objfct_sv_grid .................................................... 47
param .............................................................. 48
param.dyntermstrc_nss ........................................... 48
param.dyntermstrc_yields ....................................... 49
plot.df_curves ................................................... 49
plot.dyntermstrc_nss ............................................ 50
plot.dyntermstrc_param ......................................... 51
plot.dyntermstrc_yields ........................................ 51
plot.error .......................................................... 52
plot.fwr_curves .................................................. 53
plot.ir_curve ...................................................... 54
plot.spot_curves ................................................ 54
plot.spssearch ................................................... 55
plot.s_curves ..................................................... 56
plot.termstrc_cs .................................................. 57
plot.termstrc_nss ............................................... 58
plot.zeroyields ................................................... 59
postpro_bond ...................................................... 59
prepro_bond ........................................................ 60
print.couponbonds .............................................. 61
print.dyncouponbonds .......................................... 61
print.dyntermstrc_nss ......................................... 62
print.dyntermstrc_yields ....................................... 62
print.summary.dyntermstrc_nss ............................... 63
print.summary.dyntermstrc_param ............................ 63
print.summary.dyntermstrc_yields ............................ 64
print.summary.termstrc_cs ..................................... 64
print.summary.termstrc_nss .................................... 65
print.termstrc_cs ............................................... 65
Description

The package offers a wide range of functions for term structure estimation based on static and dynamic coupon bond and yield data sets. The implementation focuses on the cubic splines approach of McCulloch (1971, 1975) and the Nelson and Siegel (1987) method with extensions by Svensson (1994), Diebold and Li (2006) and De Pooter (2007). We propose a weighted constrained optimization procedure with analytical gradients and a globally optimal start parameter search algorithm. Extensive summary statistics and plots are provided to compare the results of the different estimation methods. Several demos are available using data from European government bonds and yields.

References


Index 79


---

**aабсе**

*Average Absolute Mean Error*

---

**Description**

Calculation of the average absolute mean error (AABSE). The AABSE is also called mean absolute error (MAE).

**Usage**

`aабсе(actual, estimated)`

**Arguments**

- `actual`: vector, consisting of the observed values.
- `estimated`: vector, consisting of the estimated values.

**Details**

Calculation of the AABSE according to the formula:

$$
AABSE = \frac{1}{m} ||\epsilon||_1,
$$

whereas $\epsilon$ is the vector of the yield or price errors of the bonds and $\iota$ is a column vector filled with ones. $m$ is the number of bonds, for which $\epsilon$ has been calculated.

**See Also**

`rmse`
Description

Function for the calculation of bond prices according to the chosen approach (Diebold/Li, Nelson/Siegel, Svensson) based on the cashflows and maturities matrix of the bonds.

Usage

bond_prices(method = "ns", beta, m, cf, lambda)

Arguments

- **method**: defines the desired method: "ns" for the Nelson/Siegel, "dl" for Diebold/Li, "sv" for the Svensson approach.
- **beta**: parameter vector, is linked to the chosen approach.
- **m**: maturities matrix, consists of the maturity dates which are appended to the cashflows of the bonds.
- **cf**: cashflows matrix.
- **lambda**: additional parameter for the "dl" spot rate function.

Value

Returns a list with:

- **spot_rates**: spot rates matrix
- **discount_factors**: discount factors matrix
- **bond_prices**: bond prices vector

See Also

- **spotrates**

Examples

data(govbonds)
cf <- create_cashflows_matrix(govbonds[[1]])
m <- create_maturities_matrix(govbonds[[1]])
beta <- c(0.0511,-0.0124,-0.0303,2.5429)
bond_prices(method="ns",beta,m,cf)$bond_prices
bond_yields  

**Bond Yield Calculation**

**Description**

Function for the calculation of bond yields.

**Usage**

```r
bond_yields(cashflows, m, searchint = c(-1, 1), tol = 1e-10)
```

**Arguments**

- `cashflows`: matrix with the cashflows of the bonds, including the current dirty price.
- `m`: maturity matrix of the bonds
- `searchint`: search interval for root finding.
- `tol`: desired accuracy for function `uniroot`.

**Value**

The function returns a matrix with the yields of the bonds and the associated maturities.

**See Also**

- `uniroot`

**Examples**

```r
data(govbonds)
cf_p <- create_cashflows_matrix(govbonds[[1]], include_price=TRUE)
m_p <- create_maturities_matrix(govbonds[[1]], include_price=TRUE)
bond_yields(cf_p, m_p)
```

---

create_cashflows_matrix  

**Cashflows Matrix Creation**

**Description**

Creates a matrix of cashflows for a specified group of bonds for a static bond data set. The number of rows is the number of cashflows for the bond with the longest maturity.

**Usage**

```r
create_cashflows_matrix(group, include_price = FALSE)
```
create_maturities_matrix

Arguments

- **group**: static bond data set for a certain group of bonds.
- **include_price**: if TRUE the dirty price is included (default: FALSE).

Value

Returns a matrix which consists of the calculated cashflows.

See Also

create_maturities_matrix

table

class

Description

Creates a matrix of maturities for a specified group of bonds for a static bond data set. The number of rows is the number of cashflows for the bond with the longest maturity.

Usage

```r
create_maturities_matrix(group, include_price = FALSE)
```

Arguments

- **group**: static bond data set for a certain group of bonds.
- **include_price**: if TRUE the dirty price is included (default: FALSE).

Value

The maturity matrix is returned.

See Also

create_cashflows_matrix
cSums

Examples

```r
data(govbonds)
(m <- create_maturities_matrix(govbonds[[1]]))

## maturities matrix with included maturity of the current
## dirty price, i.e., zero.
(m_p <- create_maturities_matrix(govbonds[[1]], include_price=TRUE))
```

---

<table>
<thead>
<tr>
<th>cSums</th>
<th>Form Column Sums</th>
</tr>
</thead>
</table>

Description

Calculates column sums for numeric matrices

Usage

```r
cSums(x, na.rm = FALSE, dims = 1L)
```

Arguments

- `x` matrix
- `na.rm` logical. Should missing values (including NaN) be omitted from the calculations?
- `dims` integer. Which dimensions are regarded as rows or columns to sum over

Note

The function is an optimized version of `colSums` and only used for internal calculations.

See Also

`colSums`
**datadyncouponbonds**  
*German Government Bond Data Set*

**Description**
Dynamic German government coupon bond data set

**Usage**
```r
data(datadyncouponbonds)
```

**Note**
If you use your own data set, make sure that the structure is identical to the provided data sets. Use the function `str()` to explore the data set.

---

**duration**  
*Duration, modified Duration and Duration based Weights*

**Description**
The function calculates the Macauly duration, modified duration and duration based weights.

**Usage**
```r
duration(cf_p, m_p, y)
```

**Arguments**
- `cf_p` cashflows matrix including the current dirty prices of the bonds.
- `m_p` maturity matrix, the first row is filled with zeros.
- `y` yields of the bonds.

**Details**
The duration vector is calculated using the following formula:

\[
d = \frac{\iota'(C \cdot M \cdot D)}{\iota'(C \cdot D)},
\]

whereas \( C \) is the cashflow matrix and \( M \) is the maturity matrix. \( \iota \) is a column vector filled with ones. \( (\cdot) \) denotes a elementwise matrix multiplication and \( ^{\prime} \) the transpose of a vector (matrix).

The weight \( \omega_j \) for one bond \( j \) is defined as

\[
\omega_j = \frac{1}{\sum_{i=1}^{m} d_i}.,
\]

where \( d_j \) is the duration of the \( j \)-th bond.
**Value**

The function returns a matrix with three columns, i.e., duration, modified duration and duration based weights.

**Examples**

```r
data(govbonds)
cf_p <- create_cashflows_matrix(govbonds[[1]], include_price=TRUE)
m_p <- create_maturities_matrix(govbonds[[1]], include_price=TRUE)
y <- bond_yields(cf_p, m_p)
duration(cf_p, m_p, y[,2])
```

**estimateyieldcurve**

*Estimate Zero-coupon Yield Curves*

**Description**

Estimate Zero-coupon Yield curves assuming a certain spot rate function

**Usage**

```
estimateyieldcurve(method, y, m, beta, lambda, objfct, grad_objfct, constraints, constrOptimOptions)
```

**Arguments**

- **method**: form of the spot rate function
- **y**: yields
- **m**: maturities
- **beta**: parameter vector
- **lambda**: parameter for Diebold/Li
- **objfct**: objective function
- **grad_objfct**: grad_objfct
- **constraints**: parameter constraints
- **constrOptimOptions**: solver options

**Details**

internal helper function
estimatezcyieldcurve  Estimate Zero-coupon Yield Curves

Description

Estimate Zero-coupon Yield curves assuming a certain spot rate function

Usage

estimatezcyieldcurve(method, startparam, lambda, objfct, grad_objfct, constraints, constrOptimOptions, m, cf, weights, p)

Arguments

method  form of the spot rate function
startparam  start parameter vector
lambda  parameter for Diebold/Li
objfct  objective function, e.g., sum of the weighted squared price errors
grad_objfct  gradient
constraints  constraints for the solver
constrOptimOptions  solver options
m  maturities
cf  cash flows
weights  weights
p  prices

Details

Used as internal helper function

estim_cs  Cubic Splines Term Structure Estimation

Description

Function for estimating the term structure of coupon bonds based on cubic splines.

Usage

estim_cs(bonddata, group, matrange="all", rse=TRUE)
Arguments

- **bonddata**: a data set of bonds in list format.
- **group**: vector defining the group of bonds used for the estimation, e.g., `c("GERMANY","AUSTRIA").`
- **matrange**: use "all" for no restrictions, or restrict the maturity range used for the estimation with `c(lower,upper)`.
- **rse**: TRUE (default) calculates robust standard errors for the confidence intervalls of the discount curve.

See Also

- estim_cs.couponbonds

---

**estim_cs.couponbonds**

**S3 Estim_cs Method**

Description

S3 estim.cs method for an object of the class "couponbonds". The method estimates the discount curve with the cubic splines approach by McCulloch (1975).

Usage

```r
## S3 method for class 'couponbonds'
estim_cs(bonddata, group, matrange = "all", rse = TRUE)
```

Arguments

- **bonddata**: a data set of bonds in list format.
- **group**: vector defining the group of bonds used for the estimation, e.g., `c("GERMANY","AUSTRIA").`
- **matrange**: use "all" for no restrictions, or restrict the maturity range used for the estimation with `c(lower,upper)`.
- **rse**: TRUE (default) calculates robust standard errors for the confidence intervalls of the discount curve.

Details

- **group**: The first element of the vector will be used as the reference country for the spread curve calculation. `group` can be either a vector of bond groups or a scalar.
- **bonddata**: The package is designed to work with a certain list data structure. For more information use the function `str()` to explore the structure of the example data sets.
Value

The function returns an object of the class "termstrc_cs". The object contains the following items (mainly lists):

- **group**: group of bonds (e.g. countries) used for the estimation.
- **matrange**: "none" or a vector with the maturity range.
- **n_group**: length of object group, i.e. the number of countries.
- **knotpoints**: selected knot points for the cubic splines estimation.
- **spot**: zero-coupon yield curves as object of the class "spot_curves".
- **spread**: spread curves as object of the class "s_curves".
- **forward**: forward curves as object of the class "fwr_curves".
- **discount**: discount curves as object of the class "df_curves".
- **cf**: cashflow matrices.
- **m**: maturity matrices.
- **p**: dirty prices.
- **phat**: estimated bond prices.
- **perrors**: pricing errors and maturities as object of the class "error".
- **y**: bond yields.
- **yhat**: one list for each group with the theoretical bond yields calculated with the estimated bond prices phat.
- **yerrors**: yield errors and maturities as object of the class "error".
- **alpha**: OLS coefficients of cubic splines estimation.
- **regout**: OLS estimation results as object of the class "lm".
- **rse**: robust standard errors for confidence interval calculation.

Note

For objects of the class "spot_curves", "s_curves", "df_curves", "fwr_curves", "error" appropriate plot methods are offered. For objects of the list item regout standard lm methods apply. For objects of the class "termstrc_cs" print, summary and plot methods are available. Another term structure estimation method is provided by the method `estim_nss.couponbonds`.

References


See Also

`print.termstrc_cs, summary.termstrc_cs, plot.termstrc_cs, estim_nss.couponbonds, plot.spot_curves, plot.s_curves, plot.df_curves, plot.fwr_curves, plot.error, summary.lm, plot.lm`. 
Examples

```r
# load data set
data(govbonds)

# define countries, for which the estimation
# of the zero-coupon yield curves will be carried out
group <- c("GERMANY", "AUSTRIA")

# set maturity range
matrange <- c(0, 19)

# perform estimation
x <- estim_cs(govbonds, group, matrange)

# print the obtained parameters of the estimation
print(x)

# goodness of fit measures
summary(x)

# plot the zero-coupon yield curve for each country
plot(x, errors="none")

# plot all zero-coupon yield curves together
plot(x, multiple=TRUE, errors="none")

# spread curve splot
plot(x, ctype="spread", errors="none")

# price error plot for all countries
plot(x, ctype="none")
```

---

**estim_nss**  
*Parametric Term Structure Estimation*

**Description**

Function for estimating the term structure of coupon bonds and yields, with the spot rate function of Diebold/Li, Nelson/Siegel or Svensson.

**Usage**

```r
estim_nss(dataset, ...)
```

**Arguments**

- **dataset**: object of the class "zeroyields", "couponbonds" or "dyncouponbonds"
- **...**: further arguments
estim_nss.couponbonds

See Also

estim_nss.zeroyields, estim_nss.couponbonds, estim_nss.dyncouponbonds

estim_nss.couponbonds  S3 Estim_nss Method

Description

Zero-coupon yield curve estimation with the parametric Nelson/Siegel (1987), Svensson (1994) and Diebold/Li (2006) method. The method requires an object of the class "couponbonds".

Usage

```r
## S3 method for class 'couponbonds'
estim_nss(dataset, group, matrange = "all", method = "ns",
          startparam = NULL, lambda = 0.0609 * 12, tauconstr = NULL,
          constrOptimOptions = list(control = list(maxit = 200),
                                     outer.iterations = 200, outer.eps = 1e-04),...)
```

Arguments

dataset   a static coupon bond data set of the class "couponbonds"
group     vector defining the group of bonds used for the estimation, e.g., c("GERMANY","AUSTRIA"). The spot rate curve of the first group element will be used as the reference curve for the spread curve calculation.
matrange  use "all" for no restrictions, or restrict the maturity range (in years) used for the estimation with c(lower,upper).
method    "ns" for Nelson/Siegel (default), "dl" for Diebold/Li, "sv" for Svensson or "asv" for adjusted Svensson.
startparam matrix of start parameters (number of columns is the number of parameters). If no start parameters are given, globally optimal parameters are searched automatically (default: NULL)
lambda    parameter on a yearly time scale with fixed value for "dl" spot rate function (default: 0.0609*12)
tauconstr
constrOptimOptions
...      list with solver control parameters (default: control=list(), outer.iterations=30, outer.eps.=1e-04). For further documentation please refer to optim
...      further arguments
Value

The function nelson_estim returns an object of the class "nelson". The object contains the following items (mainly lists):

group           group of bonds (e.g. countries) used for the estimation.
matrange        "none" or a vector with the maturity range.
method          estimation method ("Nelson/Siegel" or "Svensson").
startparam      calculated starparameters.
n_group         length of object group, i.e. the number of countries.
lambda          lambda parameter of "dl" spot rate function.
spsearch        detailed data from the start parameter search algorithm
spot            zero-coupon yield curves as object of the class "spot_curves".
spread          spread curves as object of the class "s_curves".
forward         forward curves as object of the class "fwr_curves".
discount        discount curves as object of the class "df_curves".
expoints        extrapolation points for Nelson/Siegel method.
cf              cashflow matrices.
m              maturity matrices.
duration        duration matrix, including the modified duration and duration based weights.
p              dirty prices.
phat            estimated bond prices.
perrors         pricing errors and maturities, object of the class "error".
ac              accrued interest
y              bond yields.
yhat            one list for each group with the theoretical bond yields calculated with the estimated bond prices phat.
yerrors         yield errors and maturities as object of the class "error".
opt_result      optimization results from optim, e.g. optimal parameters, convergence info.

Note

An error message concerning the function uniroot() is in general caused by wrongly specified start parameters or by data issues.

For objects of the class "spot_curves", "s_curves", "df_curves", "fwr_curves", "error" appropriate plot methods are offered. For objects of the class "termstrc_nss" print, summary and plot methods are available. Another term structure estimation method is provided by the function estim_cs.
References


See Also

*print.termstrc_nss, summary.termstrc_nss, plot.termstrc_nss, estim_cs, plot.spot_curves, plot.s_curves, plot.df_curves, plot.fwr_curves, plot.error, uniroot*.

Examples

```r
## Run: demo(nss_static)

estim_nss.dyncouponbonds
```

## S3 Estim_nss method

### Description

The method performs an iterative term structure estimation procedure on a dynamic bond data set of the class "dyncouponbonds". Available methods are Nelson/Siegel, Diebold/Li and (adjusted) Svensson.

### Usage

```r
## S3 method for class 'dyncouponbonds'
estim_nss(dataset, group, matrange = "all", method = "ns",
lambda = 0.0609 * 12, tauconstr = NULL, optimtype = "firstglobal",
constrOptimOptions = list(control = list(maxit = 2000),
outer.iterations = 200, outer.eps = 1e-04), ...)
```

### Arguments

- `dataset`: dynamic bond data set of the class "dyncouponbonds".
- `group`: vector defining the group of bonds used for the estimation, e.g., c("GERMANY", "AUSTRIA").
- `matrange`: use "all" for no restrictions, or restrict the maturity range (in years) used for the estimation with c(lower, upper).
- `method`: "ns" for Nelson/Siegel (default), "dl" for Diebold/Li, "sv" for Svensson or "asv" for adjusted Svensson.
estim_nss.zeroyields

lambda parameter on a yearly time scale with fixed value for "dl" spot rate function (default: 0.0609*12)

tauconstr

optimtype use "firstglobal" for an initial search for globally optimal start parameters or "allglobal" for a search at every iteration.

constroptimoptions

list with solver control parameters (default: control=list(), outer.iterations=30, outer.eps.=1e-04). For further documentation please refer to optim

... further arguments

Details

The method iteratively applies the method "estim_nss.couponbonds".

Value

The method returns an object of the class "dyntermstrc_nss". The object is a list with sublists of the class "termstrc_nss".

See Also

estim_nss.couponbonds

Examples

### Run: demos(nss_dynamic)

---

estim_nss.zeroyields S3 estim_nss Method

---

Description

The method performs an iterative term structure estimation procedure on a dynamic yield data set of the class "zeroyields". Available methods are Nelson/Siegel, Diebold/Li and (adjusted) Svensson.

Usage

### S3 method for class 'zeroyields'
estim_nss(dataset, method = "ns",
lambda = 0.0609 *12, tauconstr = NULL, optimtype = "firstglobal",
constroptimoptions = list(control = list(),
outer.iterations = 200, outer.eps = 1e-04), ...)
estim_nss.zeroyields

Arguments

- **dataset**: dynamic bond data set of the class "zeroyields"
- **method**: "ns" for Nelson/Siegel (default), "dl" for Diebold/Li, "sv" for Svensson or "asv" for adjusted Svensson.
- **lambda**: parameter on a yearly time scale with fixed value for "dl" spot rate function (default: 0.0609*12)
- **tauconstr**: This is vector with parameters for the grid search procedure containing:
  For parametrizations except Diebold/Li, a grid search for the tau-parameter is performed. The parameters must lie within the following bounds.
  lower bound < [tau_1, tau_2] < upper bound
  The width of the grid is given by gridsize.
  tau_2 - tau_1 > tau distance
  (upper bound, lower bound, gridsize, tau distance)
- **optimtype**: use "firstglobal" for an initial search for globally optimal start parameters or "allglobal" for a search at every iteration.
- **constrOptimOptions**: list with solver control parameters (default: control=list(), outer.iterations=30, outer.eps.=1e-04). For further documentation please refer to optim

Value

The method returns an object of the class "dyntermstrc_yields". There are print, plot and summary method available.

References


Examples

```r
## Run: demo(zero yields)
```
**fcontrib**

*Plot Factor Contribution*

**Description**

The function plots the factor contribution of the parameters of the different spot rate functions at a certain point in time.

**Usage**

```r
fcontrib(x, method = "ns", lambda = 0.0609 * 12, index = 1,
          m = 1:10, ylim = NULL,...)
```

**Arguments**

- **x**
  - object of the class `dyntermstrc_param`
- **method**
  - Spot rate function, one of the following "ns","sv","dl","sv"
- **lambda**
  - additional parameter for "dl" spot rate function.
- **index**
  - specific point in time
- **m**
  - maturity spectrum for the plot, e.g., "c(min,max)"
- **ylim**
  - range of the y axis.
- **...**
  - further arguments

---

**fcontrib.dyntermstrc_param**

*S3 fcontrib Method*

**Description**

S3 fcontrib method for objects of the class "dyntermstrc_param". The function plots the factor contribution of the parameters of the different spot rate functions at a certain point in time.

**Usage**

```r
# S3 method for class 'dyntermstrc_param'
fcontrib(x, method = "ns", lambda = 0.0609 * 12, index = 1,
          m = 1:10, ylim = NULL,...)
```
Arguments

- **x**: object of the class `dyntermstrc_param`.
- **method**: Spot rate function, one of the following: "ns", "sv", "dl", "sv".
- **lambda**: Additional parameter for "dl" spot rate function.
- **index**: Specific point in time.
- **m**: Maturity spectrum for the plot, e.g., "c(min, max)."
- **ylim**: Range of the y axis.
- **...**: Further arguments.

**findstartparambonds**  
*Find Globally Optimal Startparameters*

Description

Start parameter search routine for term structure estimation based on a coupon bond data set. The algorithm searches for the parameters over a grid spanned over tau1 (tau2).

Usage

```r
findstartparambonds(p, m, cf, weights, method, tauconstr, control = list(), outer.iterations = 30, outer.eps = 1e-04)
```

Arguments

- **p**: Price vector.
- **m**: Maturites matrix.
- **cf**: Cashflows matrix.
- **weights**: Duration based weights.
- **method**: Form of the spot rate function.
- **tauconstr**: Control solver control parameters, for details see `optim`.
- **outer.iterations**: See `constrOptim`.
- **outer.eps**: See `constrOptim`.

Details

Used as internal helper function.

Value

Returns an object of the class "spsearch", which includes the startparameters and details concerning the optimization.
findstartparamyields  Find Globally Optimal Start Parameters

Description
Start parameter search routine for term structure estimation based on a yield data set. The algorithm searches for the parameters over a grid spanned over tau1 (tau2).

Usage
findstartparamyields(y, m, method, tauconstr, control = list(), outer.iterations = 30, outer.eps = 1e-04)

Arguments
- y: yields
- m: maturities
- method: type of spot rate function
- tauconstr: tau parameter constraints
- control: solver control parameters, for details see optim
- outer.iterations: see constrOptim
- outer.eps: see constrOptim

Details
Used as internal helper function

Value
Returns an object of the class "spsearch", which includes the startparameters and details concerning the optimization.

forwardrates  Forward Rate Calculation

Description
Calculates forward rates according to the Diebold/Li, Nelson/Siegel, Svensson approach.

Usage
forwardrates(method, beta, m, lambda)
Arguments

- **method**: forward rate function type: "dl" for Diebold/Li, "ns" for Nelson/Siegel, "sv" for Svensson, "asv" for adjusted Svensson.

- **beta**: parameter vector $\beta$.

- **m**: maturity or a vector of maturities.

- **lambda**: $= 1/\tau_1$, a scalar; only required for Diebold/Li forward rate function.

Value

The function returns a vector with the calculated forward rates.

See Also

- `fwr_dl`, `fwr_ns`, `fwr_sv`

Examples

```r
forwardrates(method="ns", beta=c(0.03, 0.02, 0.01, 5), m=1:30)
```

Description

The function calculates the forward rates based on a given parameter and maturity vector.

Usage

```r
fwr_asv(beta, m)
```

Arguments

- **beta**: parameter vector $\beta = (\beta_0, \beta_1, \beta_2, \tau_1, \beta_3, \tau_2)$.

- **m**: maturity or vector of maturities.

Details

The forward rate for a maturity $m$ is calculated according to the following formula:

$$f(m, \beta) = \beta_0 + \beta_1 \exp \left( - \frac{m}{\tau_1} \right) + \beta_2 \left( \frac{m}{\tau_1} \exp \left( - \frac{m}{\tau_1} \right) \right) + \beta_3 \left[ \exp \left( - \frac{m}{\tau_2} \right) + \left( \frac{2m}{\tau_2} - 1 \right) \exp \left( - \frac{2m}{\tau_2} \right) \right].$$

Value

Returns the a vector with the calculated forward rate (vector).
References

See Also
forwardrates

Examples
fwr_asv(c(0.03, 0.02, 0.01, 5, 0.01, 10), 1:30)

---

**fwr_dl**
Forward Rate Calculation according to Diebold/Li.

Description
Calculate forward rates according to Diebold/Li(2006).

Usage
fwr_dl(beta, m, lambda)

Arguments
- beta: parameter vector $\beta = (\beta_0, \beta_1, \beta_2)$.
- m: maturity or maturity vector.
- lambda: $= \frac{1}{\tau_1}$, a scalar

Details
The forward rate for a maturity $m$ is calculated according to the following formula:

$$f(m, \beta, \lambda) = \beta_0 + \beta_1 \exp(-m \lambda) + \beta_2 [(m \lambda) \exp(-m \lambda)].$$

Value
The function returns the calculated forward rate (vector).

References

See Also
fwr_sv, fwr_ns, forwardrates
Examples

```r
fwr_dl(beta=c(0.03, 0.02, 0.01), 1:30, lambda=1/5)
```

Description

Calculate forward rates according to Nelson/Siegel(1987).

Usage

```r
fwr_ns(beta, m)
```

Arguments

- **beta**: parameter vector \( \beta = (\beta_0, \beta_1, \beta_2, \tau_1) \).
- **m**: maturity or maturity vector.

Details

The forward rate for a maturity \( m \) is calculated using the following relation:

\[
f(m, \beta) = \beta_0 + \beta_1 \exp \left( -\frac{m}{\tau_1} \right) + \beta_2 \left[ \left( \frac{m}{\tau_1} \right) \exp \left( -\frac{m}{\tau_1} \right) \right].
\]

Value

The function returns the calculated forward rate (vector).

References


See Also

`fwr_sv, fwr_dl, forwardrates`

Examples

```r
fwr_ns(beta=c(0.03, 0.02, 0.01, 5), 1:30)
```
Forward Rate Calculation according to Svensson (1994).

Description
Calculate forward rates according to Svensson (1994).

Usage
fwr_sv(beta, m)

Arguments
beta | parameter vector \( \beta = (\beta_0, \beta_1, \beta_2, \tau_1, \beta_3, \tau_2) \).
m | maturity or vector of maturities.

Details
The forward rate for a maturity \( m \) is calculated according to the following formula:

\[
f(m, \beta) = \beta_0 + \beta_1 \exp\left(-\frac{m}{\tau_1}\right) + \beta_2 \left[\left(\frac{m}{\tau_1}\right) \exp\left(-\frac{m}{\tau_1}\right)\right] + \beta_3 \left[\left(\frac{m}{\tau_2}\right) \exp\left(-\frac{m}{\tau_2}\right)\right].
\]

Value
Returns the a vector with the calculated forward rate (vector).

References

See Also
fwr_ns, fwr_dl forwardrates

Examples
fwr_sv(c(0.03, 0.02, 0.01, 5, 0.01, 10), 1:30)
get_constraints  Constraints Selection

Description
Selection of the appropriate constraints for constrOptim()

Usage
get_constraints(method, tauconstr)

Arguments
method  term structure estimation method
tauconstr  constraints on tau parameters

get_grad_objfct  Gradient Selection Function

Description
Selects the appropriate gradient of the objective function

Usage
get_grad_objfct(method)

Arguments
method  term structure estimation method

get_grad_objfct_bonds  Gradient Selection Function

Description
Selects the appropriate gradient of the objective function for a bond data set

Usage
get_grad_objfct_bonds(method)

Arguments
method  term structure estimation method
**get_objfct**  
*Objective Function Selection*

**Description**
Based on a chosen method the objective function is selected

**Usage**

```
get_objfct(method)
```

**Arguments**

- method  
  term structure estimation method

---

**get_objfct_bonds**  
*Objective Function Selection*

**Description**
Based on a chosen method the objective function for a bond data set is selected

**Usage**

```
get_objfct_bonds(method)
```

**Arguments**

- method  
  term structure estimation method

---

**get_paramnames**  
*Parameter Names*

**Description**
Parameter Names for term structure estimation methods

**Usage**

```
get_paramnames(method)
```

**Arguments**

- method  
  form of the spot rate function, i.e., one of the following "ns", "sv", "asv", "dl"
Value

Returns a character string with the names of the elements of the parameter vector.

Examples

get_paramnames("ns")

---

get_realnames Name Conversion

Description

Converts Term Structure Method Into Real Name

Usage

get_realnames(method)

Arguments

method form of the spot rate function, i.e., "ns", "sv", "asv", "dl"

Value

Returns a character string with the real name

Examples

get_realnames("asv")

---

gi Cubic Functions

Description

Calculation of the cubic functions according to the approach of McCulloch (1975).

Usage

gi(t, T, i, s)

Arguments

t maturity.
T knot points.
i index.
s number of basis functions.
govbonds

References

description

European government bonds.

Usage
data(govbonds)

Details
The data set bonds consists of German, Austrian and French government bonds.

Note
If you use your own data set, make sure that the structure is identical to the provided data sets. Use the function str() to explore the data set.

Examples
data(govbonds)
str(govbonds)

# The following code may be used to generate an empty data set,
# which can then be filled with bond data:

ISIN <- vector()
MATUREDATE <- vector()
ISSUEDATE <- vector()
COUPONRATE <- vector()
PRICE <- vector()
ACCRUED <- vector()

CFISIN <- vector()
CF <- vector()
DATE <- vector()

CASHFLOWS <- list(CFISIN,CF,DATE)
names(CASHFLOWS) <- c("ISIN","CF","DATE")

TODAY <- vector()
mycountry1 <- list(ISIN, MATURITYDATE, ISSUEDATE, COUPONRATE, PRICE, ACCRUED, CASHFLOWS, TODAY)
mycountry2 <- list(ISIN, MATURITYDATE, ISSUEDATE, COUPONRATE, PRICE, ACCRUED, CASHFLOWS, TODAY)

names(mycountry1) <- c("ISIN", "MATURITYDATE", "ISSUEDATE", "COUPONRATE", "PRICE", "ACCRUED", "CASHFLOWS", "TODAY")
names(mycountry2) <- c("ISIN", "MATURITYDATE", "ISSUEDATE", "COUPONRATE", "PRICE", "ACCRUED", "CASHFLOWS", "TODAY")

mybonds <- list(mycountry1, mycountry2)

names(mybonds) <- c("mycountry1", "mycountry2")
class(mybonds) <- "couponbonds"

---

**grad_asv**

*Gradient of the adjusted Svensson Loss Function for Yields*

**Description**

Calculates the gradient of the adjusted Svensson Loss Function for Yields

**Usage**

```r
grad_asv(beta, m, y)
```

**Arguments**

- **beta**: Parameter of the adjusted Svensson spot rate function (for details see `spr_asv`).
- **m**: maturity vector
- **y**: yield vector

**See Also**

- `objfct_asv`, `spr_asv`

---

**grad_asv_bonds**

*adjusted Svensson Gradient Function*

**Description**

Calculates the gradient of the objective function. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on an adjusted version of Svensson.
grad_asv_bonds_grid

Usage

grad_asv_bonds(beta, m, cf, w, p)

Arguments

<table>
<thead>
<tr>
<th>beta</th>
<th>Spot rate parameter vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>maturity matrix</td>
</tr>
<tr>
<td>cf</td>
<td>cashflow matrix</td>
</tr>
<tr>
<td>w</td>
<td>weights vector</td>
</tr>
<tr>
<td>p</td>
<td>price vector</td>
</tr>
</tbody>
</table>

Value

returns the gradient vector

grad_asv_bonds_grid  adjusted Svensson Gradient Function for the Grid Search

Description

Calculates the gradient of the objective function for the grid search. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on an adjusted version of Svensson.

Usage

grad_asv_bonds_grid(beta, tau, m, cf, w, p)

Arguments

<table>
<thead>
<tr>
<th>beta</th>
<th>Spot rate parameter vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>tau</td>
<td>fixed parameters</td>
</tr>
<tr>
<td>m</td>
<td>maturity matrix</td>
</tr>
<tr>
<td>cf</td>
<td>cashflow matrix</td>
</tr>
<tr>
<td>w</td>
<td>weights vector</td>
</tr>
<tr>
<td>p</td>
<td>price vector</td>
</tr>
</tbody>
</table>

Value

returns the gradient vector
**grad_asv_grid**  
*Adjusted Svensson Gradient Function for the Grid Search*

**Description**
Calculates the gradient of the objective function for the grid search. The objective function minimizes the sum of the squared yield errors. The spot rate function is based on the adjusted version of Svensson.

**Usage**

```
grad_asv_grid(beta, tau, m, y)
```

**Arguments**
- **beta**: Spot rate function parameter vector
- **tau**: fixed parameters
- **m**: maturity vector
- **y**: yield vector

**grad_dl**  
*Gradient of the Diebold/Li Loss Function for Yields*

**Description**
Calculates the gradient of the Diebold/Li Loss Function for Yields

**Usage**

```
grad_dl(beta, lambda, m, y)
```

**Arguments**
- **beta**: Parameter of the Diebold/Li spot rate function
- **lambda**: constant parameter of the Diebold/Li spot rate function
- **m**: maturity vector
- **y**: yield vector
**grad_dl_bonds**  
*Diebold/Li Gradient function*

**Description**
Calculates the gradient of the objective function. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on Diebold/Li.

**Usage**

```r
grad_dl_bonds(beta, lambda, m, cf, w, p)
```

**Arguments**
- `beta`: Spot rate parameter vector
- `lambda`: fixed spot rate parameter
- `m`: maturity matrix
- `cf`: cashflow matrix
- `w`: weights vector
- `p`: price vector

**Value**
returns the gradient vector

---

**grad_ns**  
*Gradient of the Nelson/Siegel Loss Function for Yields*

**Description**
Calculates the gradient of the Nelson/Siegel Loss Function for Yields

**Usage**

```r
grad_ns(beta, m, y)
```

**Arguments**
- `beta`: Parameter of the Nelson/Siegel spot rate function (for details see `spr_ns`).
- `m`: maturity vector
- `y`: yield vector

**See Also**
- `objfct_ns`, `spr_ns`
grad_ns_bonds

Nelson/Siegel Gradient Function

Description

Calculates the gradient of the objective function. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on Nelson/Siegel.

Usage

grad_ns_bonds(beta, m, cf, w, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>Spot rate parameter vector</td>
</tr>
<tr>
<td>m</td>
<td>maturity matrix</td>
</tr>
<tr>
<td>cf</td>
<td>cashflow matrix</td>
</tr>
<tr>
<td>w</td>
<td>weights vector</td>
</tr>
<tr>
<td>p</td>
<td>price vector</td>
</tr>
</tbody>
</table>

Value

returns the gradient vector

grad_ns_bonds_grid

Nelson/Siegel Gradient Function for the Grid Search

Description

Calculates the gradient of the objective function for the grid search. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on Nelson/Siegel.

Usage

grad_ns_bonds_grid(beta, tau, m, cf, w, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>Spot rate parameter vector</td>
</tr>
<tr>
<td>tau</td>
<td>fixed parameters</td>
</tr>
<tr>
<td>m</td>
<td>maturity matrix</td>
</tr>
<tr>
<td>cf</td>
<td>cashflow matrix</td>
</tr>
<tr>
<td>w</td>
<td>weights vector</td>
</tr>
<tr>
<td>p</td>
<td>price vector</td>
</tr>
</tbody>
</table>
grad_ns_grid

Value
returns the gradient vector

Description
 Calculates the gradient of the objective function for the grid search. The objective function mini-
mizes the sum of the squared yield errors. The spot rate function is based on Nelson/Siegel.

Usage
 grad_ns_grid(beta, tau, m, y)

Arguments
beta Spot rate function parameter vector
tau fixed parameters
m maturity vector
y yield vector

grad_sv

Description
 Calculates the gradient of the Svensson Loss Function for Yields

Usage
 grad_sv(beta, m, y)

Arguments
beta Parameter of the Svensson spot rate function.
m maturity vector
y yield vector
**grad_sv_bonds**  *Svensson Gradient Function*

**Description**

Calculates the gradient of the objective function. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on Svensson.

**Usage**

```
grad_sv_bonds(beta, m, cf, w, p)
```

**Arguments**

- `beta`: Spot rate parameter vector
- `m`: maturity matrix
- `cf`: cashflow matrix
- `w`: weights vector
- `p`: price vector

**Value**

returns the gradient vector

---

**grad_sv_bonds_grid**  *Svensson Gradient Function for the Grid Search*

**Description**

Calculates the gradient of the objective function for the grid search. The objective function minimizes the sum of the weighted squared price errors. The spot rate function is based on Svensson.

**Usage**

```
grad_sv_bonds_grid(beta, tau, m, cf, w, p)
```

**Arguments**

- `beta`: Spot rate parameter vector
- `tau`: fixed parameters
- `m`: maturity matrix
- `cf`: cashflow matrix
- `w`: weights vector
- `p`: price vector
Value

returns the gradient vector

grad_sv_grid

Svensson Gradient Function for the Grid Search

Description

Calculates the gradient of the objective function for the grid search. The objective function minimizes the sum of the squared yield errors. The spot rate function is based on Svensson.

Usage

grad_sv_grid(beta, tau, m, y)

Arguments

- **beta**: Spot rate function parameter vector
- **tau**: fixed parameters
- **m**: maturity vector
- **y**: yield vector

impl_fwr

Implied Forward Rate Calculation

Description

Calculates the implied forward rates from given spot rates.

Usage

impl_fwr(m, s)

Arguments

- **m**: maturity vector.
- **s**: spot rate vector.

Details

Implied forward rates can be calculated using the following relationship:

\[ f(t', T) = \frac{s(m_T)m_T - s(m_{t'})m_{t'}}{m_T - m_{t'}} \]

whereas \( s(m_T), s(m_{t'}) \) is the spot rate for a maturity \( m_T, m_{t'} \) respectively.
Value
The function returns the calculated forward rate vector.

Examples

```r
s <- spr_ns(c(0.03, 0.02, 0.01, 5), 1:30)
impl_fwr(s, m=1:30)
```

---

### loss_function

*Loss Function used for the Term Structure Estimation*

**Description**
The loss function defines the objective function used for the optimisation. In case of term structure estimation the objective function is the sum of the weighted squared price errors.

**Usage**

```r
loss_function(p, phat, omega)
```

**Arguments**

- `p`: vector of observed prices.
- `phat`: vector of estimated prices.
- `omega`: weights vector, e.g., duration based weights.

---

### maturity_range

*Restricting a Bond Dataset*

**Description**
The function restricts a bond data set to a specified maturity range.

**Usage**

```r
maturity_range(bonddata, lower, upper)
```

**Arguments**

- `bonddata`: bond data set.
- `lower`: lower bound of maturity spectrum.
- `upper`: upper bound of maturity spectrum.

**Note**
Internal helper function.
**objfct_asv**

*Adjusted Svensson Loss Function for Yields*

**Description**

Calculates the sum of the squared spot rate error.

**Usage**

```r
objfct_asv(beta, m, y)
```

**Arguments**

- `beta`: Parameter vector of the adjusted Svensson spot rate function (for details see: `spr_asv`).
- `m`: maturity vector
- `y`: observed yield vector

**See Also**

`spotrates`, `spr_asv`

---

**objfct_asv_bonds**

*Adjusted Svensson Loss Function for Bonds*

**Description**

Calculates the sum of the weighted squared price error.

**Usage**

```r
objfct_asv_bonds(beta, m, cf, w, p)
```

**Arguments**

- `beta`: Parameter vector of the adjusted Svensson spot rate function
- `m`: maturity matrix
- `cf`: cashflow matrix
- `w`: weights
- `p`: price vector
**objfct_asv_bonds_grid**  
*Adjusted Svensson Grid Loss Function for Bonds*

**Description**
Calculates the sum of the weighted squared price error.

**Usage**

```r
objfct_asv_bonds_grid(beta, tau, m, cf, w, p)
```

**Arguments**

- **beta**: Beta parameters of adjusted Svensson spot rate function
- **tau**: Tau parameters of adjusted Svensson spot rate function
- **m**: maturity matrix
- **cf**: cashflow matrix
- **w**: weights vector
- **p**: price vector

**objfct_asv_grid**  
*Adjusted Svensson Grid Loss Function for Yields*

**Description**
Calculates the sum of the squared yield error.

**Usage**

```r
objfct_asv_grid(beta, tau, m, y)
```

**Arguments**

- **beta**: Beta parameters of adjusted Svensson spot rate function
- **tau**: Tau parameters of adjusted Svensson spot rate function
- **m**: maturity vector
- **y**: yield vector
**Diebold/Li Loss Function for Yields**

**Description**
Calculates the sum of the squared spot rate error.

**Usage**

```r
objfct_dl(beta, lambda, m, y)
```

**Arguments**
- `beta`: Parameter vector of the Diebold/Li spot rate function (for details see: `spr_dl`)
- `lambda`: Fixed spot rate function parameter
- `m`: maturity vector
- `y`: observed yield vector

**Diebold/Li Loss Function for Bonds**

**Description**
Calculates the sum of the weighted squared price error.

**Usage**

```r
objfct_dl_bonds(beta, lambda, m, cf, w, p)
```

**Arguments**
- `beta`: Parameter vector of the Diebold/Li spot rate function
- `lambda`: Lambda of the Diebold/Li spot rate function
- `m`: maturity matrix
- `cf`: cashflow matrix
- `w`: weights vector
- `p`: price vector
objfct_ns  
Nelson/Siegel Loss Function for Yields

**Description**
Calculates the sum of the squared spot rate error.

**Usage**
```
objfct_ns(beta, m, y)
```

**Arguments**
- **beta**: Parameter vector of the Nelson/Siegel spot rate function (for details see: `spr_ns`),
- **m**: maturity vector
- **y**: observed yield vector

**See Also**
- `spotrates`, `spr_ns`

---

objfct_ns_bonds  
Nelson/Siegel Loss Function for Bonds

**Description**
Calculates the sum of the weighted squared price error.

**Usage**
```
objfct_ns_bonds(beta, m, cf, w, p)
```

**Arguments**
- **beta**: Parameter vector of the Nelson/Siegel spot rate function
- **m**: maturity matrix
- **cf**: cashflow matrix
- **w**: weights
- **p**: price vector
**objfct_ns_bonds_grid**  
*Nelson/Siegel Grid Loss Function for Bonds*

**Description**
Calculates the sum of the weighted squared price error.

**Usage**

```
objfct_ns_bonds_grid(beta, tau, m, cf, w, p)
```

**Arguments**
- `beta`: Beta parameters of the Svensson spot price function
- `tau`: Tau parameters of the Svensson spot price function
- `m`: maturities matrix
- `cf`: cash flows matrix
- `w`: weights vector
- `p`: price vector

---

**objfct_ns_grid**  
*Nelson/Siegel Grid Loss Function for Yields*

**Description**
Calculates the sum of the squared yield error.

**Usage**

```
objfct_ns_grid(beta, tau, m, y)
```

**Arguments**
- `beta`: Beta parameters of the Nelson/Siegel spot rate function
- `tau`: Tau parameter of Nelson/Siegel spot rate function
- `m`: maturity vector
- `y`: yield vector
objfct_sv  

Svensson Loss Function for Yields

Description

Calculates the sum of the squared spot rate error.

Usage

objfct_sv(beta, m, y)

Arguments

- **beta**: Parameter vector of the Svensson spot rate function (for details see: spr_sv).
- **m**: maturity vector
- **y**: observed yield vector

See Also

spotrates, spr_sv

objfct_sv_bonds  

Svensson Loss Function for Bonds

Description

Calculates the sum of the weighted squared price error.

Usage

objfct_sv_bonds(beta, m, cf, w, p)

Arguments

- **beta**: Parameter vector of the Svensson spot rate function.
- **m**: maturity matrix
- **cf**: cashflow matrix
- **w**: weights
- **p**: price vector
**Description**
Calculates the sum of the weighted squared price error.

**Usage**
objfct_sv_bonds_grid(beta, tau, m, cf, w, p)

**Arguments**
- `beta`: Beta parameters of the Svensson spot price function
- `tau`: Tau parameters of the Svensson spot price function
- `m`: maturities matrix
- `cf`: cash flows matrix
- `w`: weights vector
- `p`: price vector

**Description**
Calculates the sum of the squared yield error.

**Usage**
objfct_sv_grid(beta, tau, m, y)

**Arguments**
- `beta`: Beta parameters of the Svensson spot rate function
- `tau`: Tau parameters of the Svensson spot rate function
- `m`: maturity vector
- `y`: yield vector
Term Structure Parameter Extraction

Description
The function extracts the estimated term structure parameters.

Usage
param(x, ...)

Arguments
x
- object of the class "dyntermstrc_yields" or "dyntermstrc_nss"

... further arguments

Details
For the class "dyntermstrc_param" print, summary and plot methods are offered.

Value
Returns a list of the class "dyntermstrc_param"

See Also
param.dyntermstrc_nss, param.dyntermstrc_yields, summary.dyntermstrc_param, plot.dyntermstrc_param
**Description**

The function extracts the estimated term structure parameters from an object of the class "dyntermstrc_yields".

**Usage**

```r
## S3 method for class 'dyntermstrc_yields'
param(x, ...)
```

**Arguments**

- `x` object of the class "dyntermstrc_yields"
- `...` further arguments

**See Also**

`param`

---

**Description**

S3 plot method for an object of the class "df_curves".

**Usage**

```r
## S3 method for class 'df_curves'
plot(x, multiple = FALSE,
     ylim = c(range(mapply(function(i) range(x[[i]][, 2]),
                  seq(x))))) * 100, xlab = c(), type = "l", lty = 1, lwd = 2,
     expoints = NULL, ylab = "Discount factor (percent)",
     xlab = "Maturity (years)", main = "Discount factor curves", ...)
```
Arguments

- `x`: object of the class "df_curves".
- `multiple`: if TRUE all discount factor curves are plotted together (default: FALSE).
- `ylim`: the y limits of the plot, for details see `plot.default`.
- `xlim`: the x limits of the plot, for details see `plot.default`.
- `type`: 1-character string giving the type of plot desired, for details see `plot.default`.
- `lty`: the line type, for details see `par`.
- `lwd`: the line width, for details see `par`.
- `expoints`: extrapolation points (default: NULL).
- `ylab`: a label for the y axis, for details see `plot.default`.
- `xlab`: a label for the x axis, for details see `plot.default`.
- `main`: a main title for the plot, for details see `title`.
- `...`: other graphical parameters, see `par`.

See Also

- `plot.fwr_curves`, `plot.s_curves`, `plot.spot_curves`

---

**Description**

Plot method for objects of the class "dyntermstrc_nss". The method plots the estimated three-dimensional spot rate curve.

**Usage**

```r
## S3 method for class 'dyntermstrc_nss'
plot(x, range = c(0, 20), ...)
```

**Arguments**

- `x`: object of the class "dyntermstrc_nss"
- `range`: maturity range, e.g., `c(0, 20)` (default)
- `...`: further arguments
plot.dyntermstrc_param

S3 Plot Method

Description
Plot method for objects of the class "dyntermstrc_param". The method is able to plot the time series of the parameter levels and first differences and the empirical (partial) autocorrelation function.

Usage
```r
## S3 method for class 'dyntermstrc_param'
plot(x, type = "param", ...)"```

Arguments
- `x` object of the class "dyntermstrc_param"
- `type` "param" (default) for the parameters, "diffparam" for the parameter differences and "acf" for the plot of the (partial) autocorrelation function of the parameters.
- `...` further arguments

plot.dyntermstrc_yields

S3 Plot Method

Description
Plot method for object of the class "dyntermstrc_yields". The method plot the estimated three-dimensional spot rate curve.

Usage
```r
## S3 method for class 'dyntermstrc_yields'
plot(x, ...)"```

Arguments
- `x` object of the class "dyntermstrc_yields".
- `...` further arguments.
Description

S3 plot method for an object of the class `error`.

Usage

```r
## S3 method for class 'error'
plot(x, type = "b", main = "", mar = c(7, 6, 6, 2) + 0.1,
oma = c(4, 2, 2, 2) + 0.1, ylab = "Error", ...)
```

Arguments

- `x`: object of the class `error`.
- `type`: 1-character string giving the type of plot desired, for details see `plot.default`.
- `main`: a main title for the plot, for details see `title`.
- `mar`: A numerical vector of the form `c(bottom, left, top, right)` which gives the number of lines of margin to be specified on the four sides of the plot, for details see `par`.
- `oma`: A vector of the form `c(bottom, left, top, right)` giving the size of the outer margins in lines of text.
- `ylab`: a label for the y axis, for details see `plot.default`.
- `...`: other graphical parameters, see `par`.

Details

Absolute yield and price errors as a result of the term structure estimation can be plotted. The scaling of the x axis depends on the maturity of the bonds, each bond is labeled with its ISIN number. The error plots seems especially useful in identifying misspriced bonds. For removing them, the function `rm_bond` may be applied.

See Also

- `rm_bond`
plot.fwr_curves  S3 Plot Method

Description

S3 plot method for an object of the class "fwr_curves".

Usage

## S3 method for class 'fwr_curves'
plot(x, multiple = FALSE, ylim = c(range(mapply(function(i) range(x[[i]][, 2]), seq(x))) * 100, xlim = c(), type = "l", lty = 1, lwd = 2, expoints = NULL, ylab = "Forward rate (percent)", xlab = "Maturity (years)", main = "Forward rate curves", ...)

Arguments

- `x` object of the class "fwr_curves".
- `multiple` if TRUE all forward rate curves are plotted together (default: FALSE).
- `ylim` the y limits of the plot, for details see `plot.default`.
- `xlim` the x limits of the plot, for details see `plot.default`.
- `type` 1-character string giving the type of plot desired, for details see `plot.default`.
- `lty` the line type, for details see `par`.
- `lwd` the line width, for details see `par`.
- `expoints` extrapolation points (default: NULL).
- `ylab` a label for the y axis, for details see `plot.default`.
- `xlab` a label for the x axis, for details see `plot.default`.
- `main` a main title for the plot, for details see `title`.
- ... other graphical parameters, see `par`.

See Also

plot.df_curves, plot.s_curves, plot.spot_curves
plot.ir_curve S3 Plot Method

Description

S3 plot method for an object of the class "ir_curve".

Usage

```r
## S3 method for class 'ir_curve'
plot(x, ylim = c(), xlim = c(), lwd = 2, type = "l",
xlab = "Maturity (years)", ylab = "Zero-coupon yields (in percent)",
col = "steelblue", lty = 1, ...)
```

Arguments

- **x**: object of the class "ir_curve".
- **ylim**: the y limits of the plot, for details see `plot.default`.
- **xlim**: the x limits of the plot, for details see `plot.default`.
- **lwd**: the line width, for details see `par`.
- **type**: 1-character string giving the type of plot desired, for details see `plot.default`.
- **xlab**: a label for the x axis, for details see `plot.default`.
- **ylab**: a label for the y axis, for details see `plot.default`.
- **col**: the colors for lines and points.
- **lty**: the line type, for details see `par`.
- **...**: other graphical parameters, see `par`.

plot.spot_curves S3 Plot Method

Description

S3 plot method for an object of the class "spot_curves".

Usage

```r
## S3 method for class 'spot_curves'
plot(x, multiple = FALSE,
ylim = c(range(mapply(function(i) range(x[[i]][[2]], 2]),
seq(x)))) * 100, xlim = c(), type = "l", lty = 1,
lwd = 2, expoints = NULL, ylab = "Zero-coupon yields (percent)",
xlab = "Maturity (years)", main = "Zero-coupon yield curves", ...)
```
plot.spsearch

Arguments

- **x**: object of the class "spot_curves".
- **multiple**: if TRUE all zero-coupon yield curves are plotted together (default: FALSE).
- **ylim**: the y limits of the plot, for details see `plot.default`.
- **xlim**: the x limits of the plot, for details see `plot.default`.
- **type**: 1-character string giving the type of plot desired, for details see `plot.default`.
- **lty**: the line type, for details see `par`.
- **lwd**: the line width, for details see `par`.
- **expoints**: extrapolation points (default: NULL.)
- **ylab**: a label for the y axis, for details see `plot.default`.
- **xlab**: a label for the x axis, for details see `plot.default`.
- **main**: a main title for the plot, for details see `title`.
- **...**: other graphical parameters, see `par`.

See Also

`plot.df_curves, plot.fwr_curves, plot.s_curves`

plot.spsearch  S3 Plot Method

Description

S3 plot method for objects of the class "spsearch". The methods plot details on the objective function of the start parameter search.

Usage

```r
## S3 method for class 'spsearch'
plot(x, main = "Start parameter search", rgl=TRUE, ...)
```

Arguments

- **x**: object of the class "spsearch".
- **main**: title.
- **rgl**: if TRUE (default) the rgl device will be used for the plot.
- **...**: further arguments.
Description

S3 plot method for an object of the class "s_curves".

Usage

```r
## S3 method for class 's_curves'
plot(x, xlim = c(range(mapply(function(i) range(x[[i]][, 1]), seq(x)))),
     ylim = c(range(mapply(function(i) range(x[[i]][, 2]),
                      seq(x))))) * 10000, expoints = NULL,
     xlab = "Maturity (years)", ylab = "Spread (basis points)",
     lwd = 2, lty = 1, main = "Spread curves", ...)
```

Arguments

- `x`: object of the class "s_curves".
- `ylim`: the y limits of the plot, for details see `plot.default`.
- `xlim`: the x limits of the plot, for details see `plot.default`.
- `lty`: the line type, for details see `par`.
- `lwd`: the line width, for details see `par`.
- `expoints`: extrapolation points (default: NULL).
- `ylab`: a label for the y axis, for details see `plot.default`.
- `xlab`: a label for the x axis, for details see `plot.default`.
- `main`: a main title for the plot, for details see `title`.
- `...`: other graphical parameters, see `par`.

Details

The spread curves (the difference of zero-coupon yield curves) are plotted, if at least two groups of bonds were specified.

See Also

- `plot.df_curves`, `plot.fwr_curves`, `plot.spot_curves`
Description

S3 plot method for an object of the class "termstrc_cs".

Usage

```
## S3 method for class 'termstrc_cs'
plot(x, matrange = c(min(mapply(function(i) min(x$sy[[i]][, 1]),
seq(x$n_group))), max(mapply(function(i) max(x$sy[[i]][, 1]),
seq(x$n_group)))), multiple = FALSE,
ctype = "spot", lwd=2, lty=1, type = "l",
errors = "none", inset = c(0.1, 0.3), ask=TRUE, ...)
```

Arguments

- `x`: object of the class "termstrc_cs".
- `matrange`: maturity range for the plot, e.g. c(2,10).
- `multiple`: if TRUE all curves are plotted together (default: FALSE).
- `ctype`: parameter setting for the desired curve type, "spot" ("forward", "discount", "spread") for the spot rate (forward rate, discount factor, spread) curves. Use "none" if no curve plot is desired.
- `errors`: Specify the type of the error plot. If "price" ("yield") the price (yield) errors will be plot. Use "none" if no error plot is desired.
- `lwd`: the line width, for details see `par`.
- `lty`: the line type, for details see `par`.
- `type`: 1-character string giving the type of plot desired, for details see `plot.default`.
- `inset`: inset distance(s) from the margins as a fraction of the plot region, for details see `legend`.
- `ask`: if TRUE (and the R session is interactive) the user is asked for input, before a new figure is drawn, see `par` for details.
- `...`: other graphical parameters, see `par`.

Details

Depending on the choice of the curve type ("spot", "forward", "discount", "spread") the corresponding curves will be plotted. Either separately or together (multiple = TRUE). If the curves are plotted separately also the knot points used for the estimation of the cubic splines and the yield-to-maturities will be plotted. In addition, with a zero-coupon yield curve plot the 95 % confidence interval of the curve will be plotted. To ease the analysis of the goodness of the estimation, severeral error plots for the yield and price error are offered.
Description

S3 plot method for an object of the class "termstrc_nss".

Usage

```r
## S3 method for class 'termstrc_nss'
plot(x, matrange = c(min(mapply(function(i) min(x$y[[i]][, 1]),
    seq(x$n_group))), max(mapply(function(i) max(x$y[[i]][, 1]),
    seq(x$n_group))), multiple = FALSE, expoints = unlist(x$expoints),
cotype = "spot", errors = "none", lwd = 2, lty = 1, type ="l",
inset = c(0.8, 0.1), ask = TRUE, ...)
```

Arguments

- `x` object of the class "termstrc_nss".
- `matrange` maturity range for the plot, e.g., c(2,10). Only a range within the maturity range of the estimation is allowed.
- `multiple` if TRUE all curves are plotted together (default: FALSE).
- `expoints` extrapolation points (default: NULL).
- `cotype` parameter setting for the desired curve type, "spot" ("forward", "discount", "spread") for the spot rate (forward rate, discount factor, spread) curves. Use "none" if no curve plot is desired.
- `errors` Specify the type of the error plot. If "price" ("yield") the pricing (yield) errors will be plotted. Use "none" if no error plot is desired.
- `lwd` the line width, for details see `par`.
- `lty` the line type, for details see `par`.
- `type` 1-character string giving the type of plot desired, for details see `plot.default`.
- `inset` inset distance(s) from the margins as a fraction of the plot region, for details see `legend`.
- `ask` if TRUE (and the R session is interactive) the user is asked for input, before a new figure is drawn, see `par` for details.
- `...` other graphical parameters, see `par`.

See Also

- `plot.df_curves`, `plot.error`, `plot.fwr_curves`, `plot.ir_curve`, `plot.s_curves`, `plot.spot_curves`, `plot.termstrc_cs`
Details

Depending on the choice of the curve type ("spot", "forward", "discount", "spread") the corresponding curves will be plot. Either separately or together (multiple = TRUE). If the curves are plotted together a dashed line indicates that the corresponding curve has been extrapolated. In addition, with a separate zero-coupon yield curve plot the yield-to-maturity will be plot. To ease the analysis of the goodness of the estimation, several error plots are offered.

See Also

plot.df_curves, plot.error, plot.fwr_curves, plot.ir_curve, plot.s_curves, plot.spot_curves, plot.termstrc_nss

plot.zeroyields  S3 Plot Method

Description

S3 plot method for objects of the class "zeroyields". The method plots the estimated three-dimensional spot rate curve.

Usage

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

Arguments

x  
object of the class "zeroyields"

...  
进一步的 arguments

postpro_bond  Post Processing of Term Structure Estimation Results

Description

The function calculates based on the term structure estimation results the errors for prices and yields and different curves (spot, forward, discount curve).

Usage

postpro_bond(opt_result, m, cf, sgroup, n_group, y, p, ac, m_p, method, lambda)
Arguments

- `opt_result` parameter vector
- `m` maturities matrices
- `cf` cashflows matrices
- `sgroup` sequence of the group length
- `n_group` length of the group
- `y` yield-to-maturity matrices
- `p` dirty price vectors
- `ac` accrued interest vectors
- `m_p` maturity matrices including the maturities for the current dirty prices
- `method` form of the spot rate function
- `lambda` additional parameter for the Diebold/Li spot rate function

Note

Used as internal helper function

---

**prepro_bond**  
_Bonddata preprocess function_

Description

Preprocessing a static coupon bond data set, i.e., calculation of cashflows, maturities matrices, price, accrued interest vectors, yield-to-maturity and duration matrices.

Usage

`prepro_bond(group, bonddata, matrange = "all")`

Arguments

- `group` character, specifies group of a bond data set.
- `bonddata` Static bond data set.
- `matrange` bond data set is filtered according to chosen maturity spectrum \(c(min, max)\).

Value

- `n_group` group length
- `sgroup` sequence of the group length
- `cf` list with cashflows matrices
- `cf_p` list with cashflows matrices including the current dirty prices
- `m` list with maturities matrices
print.couponbonds

m_p  list with cashflows matrices including the maturities of the current dirty prices
p    list with the dirty price vectors
ac   list with the accrued interest vectors
y    list with the yield-to-maturity matrices
duration list with the duration, duration based weights matrices
timestamp date of the data

Examples

data(govbonds)
bdata <- prepro_bond("GERMANY", govbonds, c(0,10))
## print maturites matrix
bdata$m

print.couponbonds   S3 Print Method

Description

Prints basic information of an coupon bond data set.

Usage

## S3 method for class 'couponbonds'
print(x, ...)

Arguments

x object of the class "couponbonds"
...

futher arguments

print.dyncouponbonds   S3 Print Method

Description

The method prints basic information of a dynamic bond data set.

Usage

## S3 method for class 'dyncouponbonds'
print(x, ...)

Arguments

x object of the class "dyncouponbonds"
...

further arguments
print.dyntermstrc_nss  S3 Print Method

Description

S3 print method for objects of the class "dyntermstrc_nss". The basic parameter and a summary of the estimated term structure parameters are printed.

Usage

```r
## S3 method for class 'dyntermstrc_nss'
print(x, ...)
```

Arguments

- `x` object of the class "dyntermstrc_nss"
- `...` further arguments

print.dyntermstrc_yields  S3 Print Method

Description

S3 print method for objects of the class "dyntermstrc_yields". The method prints information from the term structure estimation and a summary of the estimated parameters.

Usage

```r
## S3 method for class 'dyntermstrc_yields'
print(x, ...)
```

Arguments

- `x` object of the class "dyntermstrc_yields"
- `...` further arguments
print.summary.dyntermstrc_nss

S3 Print Method

Description
Print method for objects of the class "summary.dyntermstrc_nss"

Usage
## S3 method for class 'summary.dyntermstrc_nss'
print(x, ...)

Arguments
x  object of the class "summary.dyntermstrc"
...

further arguments

print.summary.dyntermstrc_param

S3 Print Method

Description
S3 print method for an object of the class "summary.dyntermstrc_param"

Usage
## S3 method for class 'summary.dyntermstrc_param'
print(x, ...)

Arguments
x  object of the class "summary.dyntermstrc_param"
...

further arguments
print.summary.termstrc_yields
S3 Print Method

Description
S3 print method for objects of the class "summary.dyntermstrc_yields".

Usage
### S3 method for class 'summary.dyntermstrc_yields'
print(x, ...)

Arguments
x object of the class "summary.dyntermstrc_yields"
...
  further arguments.

print.summary.termstrc_cs
S3 Print Method

Description
S3 print method for an object of the class "summary.termstrc_cs".

Usage
### S3 method for class 'summary.termstrc_cs'
print(x, ...)

Arguments
x object of the class "summary.termstrc_cs".
...
  other arguments.
print.summary.termstrc_nss

S3 Print Method

Description

S3 print method for an object of the class "summary.termstrc_nss".

Usage

```r
## S3 method for class 'summary.termstrc_nss'
print(x, ...)
```

Arguments

- `x` object of the class "summary.termstrc_nss".
- `...` other arguments.

print.termstrc_cs

S3 Print Method for termstrc_cs

Description

S3 print method for an object of the class "termstrc_cs".

Usage

```r
## S3 method for class 'termstrc_cs'
print(x, ...)
```

Arguments

- `x` object of the class "termstrc_cs".
- `...` other arguments.

Details

The print method for an object of the class "termstrc_cs" prints the parameter estimates and the associated (robust) standard errors of the cubic spline functions.

See Also

- `plot.termstrc_cs`
- `summary.termstrc_cs`
### print.termstrc_nss  
**S3 Print Method**

#### Description
Print method for objects of the class "termstrc_nss".

#### Usage
```r
## S3 method for class 'termstrc_nss'
print(x, ...)
```

#### Arguments
- `x` objects of the class "termstrc_nss"
- `...` further arguments

#### Details
The print method for an object of the class "nelson" prints important input parameters of the optimisation and the results (the optimal parameter vector).

### print.zeroyields  
**S3 Print Method**

#### Description
S3 print method for objects of the class "zeroyields". The method prints basic information of a zeroyields data set.

#### Usage
```r
## S3 method for class 'zeroyields'
print(x, ...)
```

#### Arguments
- `x` object of the class "zeroyields"
- `...` further arguments
rmse  

**Root Mean Squared Error**

**Description**
Calculates the root mean squared error (RMSE).

**Usage**
```
rmse(actual, estimated)
```

**Arguments**
- `actual`  vector, consisting of the observed values.
- `estimated`  vector, consisting of the estimated values.

**Details**
Calculation of the RMSE according to the formula:

\[
RMSE = \sqrt{\frac{1}{m} \epsilon^2 \iota},
\]

whereas \( \epsilon \) is the vector of the yield or price errors of the bonds and \( \iota \) is a column vector filled with ones. \( m \) is the number of bonds, for which \( \epsilon \) has been calculated.

**See Also**
- `aabse`

---

rm_bond  

**Bond Removal Function**

**Description**
Specified bonds and their associated data are removed from a static or dynamic bond data set.

**Usage**
```
rm_bond(bonddata, group, ISIN)
```

**Arguments**
- `bonddata`  bond data set of the class "couponbond" or "dyncouponbond"
- `group`  the group where the bonds to be removed belong to.
- `ISIN`  the ISIN numbers of the bonds to remove.
**Description**

Specified bonds and their associated data are removed from a static bond data set of the class "couponbonds".

**Usage**

```r
## S3 method for class 'couponbonds'
rm_bond(bonddata, group, ISIN)
```

**Arguments**

- `bonddata`: bond data set.
- `group`: the group where the bonds to be removed belong to.
- `ISIN`: the ISIN numbers of the bonds to remove.

**Value**

The function returns the new bond data set.

**Examples**

```r
data(govbonds)
newgovbonds <- rm_bond(govbonds, "GERMANY", "DE0001135158")
```

---

**Description**

Specified bonds and their associated data are removed from a dynamic bond data set of the class "dyncouponbonds".

**Usage**

```r
## S3 method for class 'dyncouponbonds'
rm_bond(bonddata, group, ISIN)
```

**Arguments**

- `bonddata`: bond data set.
- `group`: the group where the bonds to be removed belong to.
- `ISIN`: the ISIN numbers of the bonds to remove.
Function for the Calculation of the Spot Rates

Description
The function calculates the spot rates for the chosen spot rate function (Diebold/Li, Nelson/Siegel, Svensson), a provided maturity and parameter vector.

Usage
spotrates(method, beta, m, lambda)

Arguments
- beta: parameter vector $\beta$.
- m: maturity or a vector of maturities.
- lambda: $= 1/\tau_1$, a scalar; only required for Diebold/Li spot rate function.

Value
Returns a vector with the calculated spot rates.

See Also
spr_dl, spr_ns, spr_sv

Examples
spotrates(method="ns",beta=c(0.03,0.02,0.01,5),m=1:30)
Adjusted Svensson Spot rate function

Description

This function calculates the spot rates for certain maturity dates and a parameter vector according to an adjusted version of Svensson (1994).

Usage

spr_asv(beta, m)

Arguments

- **beta**: a vector of parameters \( \beta = (\beta_0, \beta_1, \beta_2, \tau_1, \beta_3, \tau_2) \).
- **m**: one maturity (or a vector of maturities).

Details

The adjusted Svensson spot rate function is defined as:

\[
s(m, \beta) = \beta_0 + \beta_1 \frac{1 - \exp\left(-\frac{m}{\tau_1}\right)}{\frac{m}{\tau_1}} + \beta_2 \left( \frac{1 - \exp\left(-\frac{m}{\tau_1}\right)}{\frac{m}{\tau_1}} - \exp\left(-\frac{m}{\tau_1}\right) \right) + \beta_3 \left( \frac{1 - \exp\left(-\frac{m}{\tau_2}\right)}{\frac{m}{\tau_2}} - \exp\left(-\frac{2m}{\tau_2}\right) \right)
\]

Value

Returns a vector consisting of the calculated spot rates.

References


Examples

spr_asv(c(0.07, 0.03, 0.05, 2, 0.08, 7), 1:30)
Description

This function calculates the spot rates for certain maturity dates and a parameter vector according to Diebold/Li (2006).

Usage

spr_dl(beta, m, lambda)

Arguments

beta a vector of parameters $\beta = (\beta_0, \beta_1, \beta_2)$.
m one maturity (or a vector of maturities).
lambda $= \frac{1}{\tau_1}$, a scalar

Details

The spot rate according to Diebold/Li for a maturity $m$ is defined as:

$$s(m, \beta, \lambda) = \beta_0 + \beta_1 \frac{1 - \exp(-m\lambda)}{m\lambda} + \beta_2 \left( \frac{1 - \exp(-m\lambda)}{m\lambda} - \exp(-m\lambda) \right).$$

Value

Returns a vector consisting of the calculated spot rates.

References


See Also

codespr_ns

Examples

spr_dl(c(0.1, 0.03, 0.01), 1:30, 0.0609)
**spr_ns**

**Spot Rate Function according to Nelson and Siegel**

**Description**

This function calculates the spot rates for certain maturity dates and a parameter vector according to Nelson/Siegel (1987).

**Usage**

```r
spr_ns(beta, m)
```

**Arguments**

- `beta`: a vector of parameters $\beta = (\beta_0, \beta_1, \beta_2, \tau_1)$.
- `m`: one maturity (or a vector of maturities).

**Details**

The spot rate according to Nelson/Siegel for a maturity $m$ is defined as:

$$s(m, \beta) = \beta_0 + \beta_1 \frac{1 - \exp(-\frac{m}{\tau_1})}{\tau_1} + \beta_2 \left( \frac{1 - \exp(-\frac{m}{\tau_1})}{\tau_1} - \exp(-\frac{m}{\tau_1}) \right).$$

**Value**

Returns a vector consisting of the calculated spot rates.

**References**


**Examples**

```r
spr_ns(rep(0.01, 4), 1:30)
```
**Description**

This function calculates the spot rates for certain maturity dates and a parameter vector according to Svensson (1994).

**Usage**

```r
tspr_sv(beta, m)
```

**Arguments**

- `beta`: a vector of parameters $\beta = (\beta_0, \beta_1, \beta_2, \tau_1, \beta_3, \tau_2)$.
- `m`: one maturity (or a vector of maturities).

**Details**

The spot rate according to Svensson for a maturity $m$ is calculated using the following function:

$$s(m, \beta) = \beta_0 + \beta_1 \cdot \frac{1 - \exp\left(-\frac{m}{\tau_1}\right)}{\frac{m}{\tau_1}} + \beta_2 \left( \frac{1 - \exp\left(-\frac{m}{\tau_1}\right)}{\frac{m}{\tau_1}} - \exp\left(-\frac{m}{\tau_1}\right) \right) + \beta_3 \left( \frac{1 - \exp\left(-\frac{m}{\tau_2}\right)}{\frac{m}{\tau_2}} - \exp\left(-\frac{m}{\tau_2}\right) \right)$$

**Value**

Returns a vector consisting of the calculated spot rates.

**References**


**Examples**

```r
tspr_sv(c(0.07, 0.3, 0.05, 2, 0.08, 7), 1:30)
```
Summary method for objects of the class "dyntermstrc_nss". The average RMSE and AABSE for the prices and yields is calculated. Additionally convergence information and the message from the used solver is printed.

Usage

```r
## S3 method for class 'dyntermstrc_nss'
summary(object, ...)
```

Arguments

- `object` object of the class "dyntermstrc_nss".
- `...` further arguments

Value

The method returns an object of the class "summary.dyntermstrc_nss".

S3 summary method for objects of the class "dyntermstrc_param".

Usage

```r
## S3 method for class 'dyntermstrc_param'
summary(object, type = "none", lags = 1, selectlags = "Fixed", ...)
```

Arguments

- `object` object of the class "dyntermstrc_param".
- `type` use "trend" and a trend is considered for the unit root test (default: "none").
- `lags` number of lags for unit root test function `ur.df` from package urca (default:1)
- `selectlags` leg selection flag for function `ur.df` from package urca (default: "Fixed").
- `...` further arguments
Summary Method for objects of the class "dyntermstrc_yields". The mean RMSE and AABSE of the yields is calculated.

### Usage

```r
# S3 method for class 'dyntermstrc_yields'
summary(object, ...)
```

### Arguments

- `object`: object of the class "dyntermstrc_yields"
- `...`: further arguments

Summary Method for objects of the class "termstrc_cs".

### Usage

```r
# S3 method for class 'termstrc_cs'
summary(object,...)
```

### Arguments

- `object`: object of the class "termstrc_cs".
- `...`: other arguments.
The summary method for an object of the class "termstrc_cs" calculates goodness of fit statistics (RMSE, AABSE) of the price and yield errors. Additionally, summary statistics of the regression analysis of the parameters are printed.

See Also

plot.termstrc_cs, print.termstrc_cs, rmse, aabse, summary.lm

S3 Summary Method

S3 summary method for objects of the class "termstrc_nss".

Usage

## S3 method for class 'termstrc_nss'
summary(object,...)

Arguments

object  object of the class "termstrc_nss".
...
other arguments.

Details

The summary method for an object of the class "termstrc_nss" prints the solution of the goodness of fit statistics (RMSE, AABSE) of the optimisation. Moreover a convergence information of the used optimiser (optim) is printed.

See Also

nlminb, plot.termstrc_nss, print.termstrc_nss, rmse, aabse
**summary.zeroyields**

---

### S3 Summary Method

**Description**

S3 summary method for objects of the class "zeroyields". The method calculates basic summary statistics of the data.

**Usage**

```r
## S3 method for class 'zeroyields'
summary(object, ...)
```

**Arguments**

- `object`: object of the class "zeroyields"
- `...`: further arguments

**Value**

returns an object of the class "summary.zeroyields"

---

### Zeroyields Data Set Generation

**Description**

The function generates a zeroyield data set out of yield, date and maturities data.

**Usage**

```r
zeroyields(maturities, yields, dates)
```

**Arguments**

- `maturities`: maturities vector of the yields
- `yields`: yields matrix
- `dates`: vector of the observations dates in the format "

**Value**

returns a list, which belongs to the class "zeroyields". For the class plot, print and summary methods are offered.

**See Also**

`print.zeroyields, summary.zeroyields, plot.zeroyields`
<table>
<thead>
<tr>
<th>zyields</th>
<th><strong>Zero Coupon Yield Data Set</strong></th>
</tr>
</thead>
</table>

**Description**

Zero Coupon Yield Data Set

**Usage**

```r
data(zyields)
```

**Note**

If you use your own data set, make sure that the structure is identical to the provided data set. Use the function `str()` to explore the data set.
Index

*Topic datasets
datadyncouponbonds, 10
govbonds, 31
zyields, 78

*Topic package
termstrc-package, 4

aabse, 5, 67, 76

bond_prices, 6
bond_yields, 7
colSums, 9
constrOptim, 22, 23
create_cashflows_matrix, 7, 8
create_maturities_matrix, 8, 8
cSums, 9
datadyncouponbonds, 10
duration, 10
estim_cs, 12, 17, 18
estim_cs.couponbonds, 13, 13
estim_nss, 15
estim_nss.couponbonds, 14, 16, 16
estim_nss.dycouponbonds, 16, 18
estim_nss.zero yields, 16, 19
estimateyieldcurve, 11
estimatezcyieldcurve, 12

fcontrib, 21
fcontrib.dyntermstrc_param, 21
findstartparambonds, 22
findstartparamyields, 23
forwardrates, 23, 25–27
fwr_asv, 24
fwr_dl, 24, 25, 26, 27
fwr_ns, 24, 25, 26, 27
fwr_sv, 24–26, 27
get_constraints, 28

get_grad_objfct, 28
get_grad_objfct_bonds, 28
get_objfct, 29
get_objfct_bonds, 29
gi, 30
govbonds, 31
grad_asv, 32
grad_asv_bonds, 32
grad_asv_bonds_grid, 33
grad_asv_grid, 34
grad_dl, 34
grad_dl_bonds, 35
grad_ns, 35
grad_ns_bonds, 36
grad_ns_bonds_grid, 36
grad_ns_grid, 37
grad_sv, 37
grad_sv_bonds, 38
grad_sv_bonds_grid, 38
grad_sv_grid, 39

impl_fwr, 39
legend, 57, 58
loss_function, 40

maturity_range, 40

nlminb, 76

objfct_asv, 32, 41
objfct_asv_bonds, 41
objfct_asv_bonds_grid, 42
objfct_asv_grid, 42
objfct_dl, 43
objfct_dl_bonds, 43
objfct_ns, 35, 44
objfct_ns_bonds, 44
objfct_ns_bonds_grid, 45