Package 'NFCP'

June 24, 2021

Title N-Factor Commodity Pricing Through Term Structure Estimation

Version 1.1.0

Description Commodity pricing models are (systems of) stochastic differential equations that are utilized for the valuation and hedging of commodity contingent claims (i.e. derivative products on the commodity) and other commodity related investments. Commodity pricing models that capture market dynamics are of great importance to commodity market participants in order to exercise sound investment and risk-management strategies. Parameters of commodity pricing models are estimated through maximum likelihood estimation, using available term structure futures data of a commodity. 'NFCP' (n-factor commodity pricing) provides a framework for the modeling, parameter estimation, probabilistic forecasting, option valuation and simulation of commodity prices through state space and Monte Carlo methods, risk-neutral valuation and Kalman filtering. 'NFCP' allows the commodity pricing model to consist of n correlated factors, with both random walk and mean-reverting elements. The n-factor commodity pricing model framework was first presented in the work of Cortazar and Naranjo (2006) <doi:10.1002/fut.20198>. Examples presented in 'NFCP' replicate the two-factor crude oil commodity pricing model presented in the prolific work of Schwartz and Smith (2000) <doi:10.1287/mnsc.46.7.893.12034> with the approximate term structure futures data applied within this study provided in the 'NFCP' package.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1.9001

RdMacros mathjaxr,
Rdpack

Suggests knitr,
rmarkdown

Imports FKF.SP,
LSMRealOptions,
MASS,
numDeriv,
parallel,
rgenoud,
stats,
mathjaxr,
Rdpack,
curl

VignetteBuilder knitr

R topics documented:

`American_option_value` *N-factor model American options on futures contracts valuation*

Description

Value American options on futures contracts under the parameters of an N-factor model

Usage

```
American_option_value(
  x_0,
  parameters,
  futures_maturity,
  option_maturity,
  K,
  r,
  call = FALSE,
  N_simulations,
  dt,
  orthogonal = "Power",
  degree = 2,
  verbose = FALSE,
  debugging = FALSE
)
```

Arguments

`x_0`	`vector`. Initial values of the state variables, where the length must correspond to the number of factors specified in the parameters.
`parameters`	`vector`. A named vector of parameter values of a specified N-factor model. Function `NFCP_parameters` is recommended.
`futures_maturity`	`numeric`. Time, in years, when the underlying futures contract matures.

`option_maturity`
: `numeric`. Time, in years, when the American option expires.

`K`
: `numeric`. Strike price of the American Option.

`r`
: `numeric`. Annualized risk-free interest rate.

`call`
: `logical`. Is the American option a call or put option?

`N_simulations`
: `numeric`. Total number of simulated price paths.

`dt`
: `numeric`. Discrete time step, in years, of the Monte Carlo simulation.

`orthogonal`
: `character`. The orthogonal polynomial used to approximate the continuation value of the option in the LSM simulation method. Orthogonal polynomial arguments available are: "Power", "Laguerre", "Jacobi", "Legendre", "Chebyshev", "Hermite".

`degree`
: `numeric`. The degree of polynomials used in the least squares fit.

`verbose`
: `logical`. Should additional option value information be output? see **details**.

`debugging`
: `logical` Should the simulated state variables and futures prices be output?

Details

The `American_option_value` function calculates numerically the value of American options on futures contracts within the N-factor model. An American option on a commodity futures contract gives the holder the right, but not the obligation, to buy (call) or sell (put) the underlying asset at any time before option maturity. If the American option is exercised, the option devolves into buying or selling of the underlying futures asset at the exercise price.

The 'American_option_value' function uses Monte Carlo simulation and the Least-Squares Monte Carlo (LSM) simulation approach to numerically calculate the value of American options on futures contracts under the N-factor model. LSM simulation is a method that values options with early exercise opportunities, first presented by Longstaff and Schwartz (2001). LSM simulation uses discrete time steps to approximate the value of the American option and thus technically values Bermudan-style options, converging to American option values as the size of the time step approaches zero. For more information on LSM simulation, see help('LSM_American_option') of the 'LSMRealOption' package or Longstaff and Schwartz (2001).

For a provided N-factor model,the 'American_option_value' function simulates state variables under the N-factor framework through the 'spot_price_simulate' function, developing expected futures prices from these simulated state variables. The function then uses the 'LSM_American_option' of the 'LSMRealOption' package to calculate the value of the American option with early exercise opportunities.

The number of simulations has a large influence on the standard error and accuracy of calculated option values at the cost of computational expense. Large numbers of simulations are suggested to converge upon appropriate values.

Orthogonal polynomials are used in the LSM simulation method to approximate the value of continuing to hold the American option. In general, increasing the degree of orthogonal polynomials used should increase the accuracy of results, at the cost of increased computational expense.

Value

The 'American_option_value' function by default returns a `numeric` object corresponding to the calculated value of the American option.

When `verbose = T`, 6 objects related to the American option value are returned within a `list` class object. The objects returned are:

Value	The calculated option value.
Standard Error	The standard error of the calculated option value.
Expected Timing	The expected time of early exercise.
Expected Timing SE	The standard error of the expected time of early exercise.
Exercise Probability	The probability of early exercise of the option being exercised.
Cumulative Exercise Probability	vector. The cumulative probability of option exercise at each discrete observatio

When debugging = T, an additional 2 simulation objects are returned within the list class object. These objects can have high dimensions and thus memory usage, so caution should be applied. The objects returned are:

Simulated State Variables	An array of simulated state variables. The three dimensions of the array correspond to a d
Simulated Futures Prices	A matrix of simulated futures contract price paths. Each row represents one simulated dis

References

Longstaff, F.A., and E.S. Schwartz, (2001). Valuing American Options by Simulation: A Simple Least-Squares Approach. *The Review of Financial Studies.*, 14:113-147.

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Aspinall, T., A. Gepp, G. Harris, S. Kelly, C. Southam, and B. Vanstone, (2021). LSMRealOptions: Value American and Real Options Through LSM Simulation. R package version 0.1.1.

Examples

```
# Example 1 - An American put option on a futures contract following 'GBM'
American_option_value(x_0 = log(36),
                      parameters = c(mu_rn = 0.06, sigma_1 = 0.2),
                      N_simulations = 1e2,
                      futures_maturity = 1,
                      option_maturity = 1,
                      dt = 1/50,
                      K = 40,
                      r = 0.06,
                      verbose = FALSE,
                      orthogonal = "Laguerre",
                      degree = 3)

# Example 2 - An American put option under a two-factor crude oil model:

## Step 1 - Obtain current (i.e. most recent) state vector by filtering the
## two-factor oil model:
Schwartz_Smith_oil <- NFCP_Kalman_filter(parameter_values = SS_oil$two_factor,
                                         parameter_names = names(SS_oil$two_factor),
                                         log_futures = log(SS_oil$stitched_futures),
                                         dt = SS_oil$dt,
                                         futures_TTM = SS_oil$stitched_TTM,
                                         verbose = TRUE)
```

```
##Step 2 - Calculate 'put' option price:
American_option_value(x_0 = Schwartz_Smith_oil$x_t,
                      parameters = SS_oil$two_factor,
                      futures_maturity = 2,
                      option_maturity = 1,
                      K = 20,
                      r = 0.05,
                      call = FALSE,
                      N_simulations = 1e2,
                      dt = 1/12,
                      verbose = TRUE,
                      orthogonal = "Power",
                      degree = 2)
```

`European_option_value` *N-factor model European options on futures contracts valuation*

Description

Value European Option Put and Calls under the parameters of an N-factor model.

Usage

```
European_option_value(
  x_0,
  parameters,
  futures_maturity,
  option_maturity,
  K,
  r,
  call = FALSE,
  verbose = FALSE
)
```

Arguments

`x_0`	`vector`. Initial values of the state variables, where the length must correspond to the number of factors specified in the parameters.
`parameters`	`vector`. A named vector of parameter values of a specified N-factor model. Function `NFCP_parameters` is recommended.
`futures_maturity`	`numeric`. Time, in years, when the underlying futures contract matures.
`option_maturity`	`numeric`. Time, in years, when the American option expires.
`K`	`numeric`. Strike price of the American Option.
`r`	`numeric`. Annualized risk-free interest rate.
`call`	`logical`. Is the American option a call or put option?
`verbose`	`logical`. Should additional option value information be output? see **details**.

Details

The `European_option_value` function calculates analytic expressions of the value of European call and put options on futures contracts within the N-factor model. A European option on a commodity futures contract gives the holder the right, but not the obligation, to buy (call) or sell (put) the underlying asset at option maturity. If the European option is exercised, the option devolves into buying or selling of the underlying futures asset.

State variables (i.e., the states of the factors of an N-factor model) are generally unobservable. Filtering the commodity pricing model using term structure data will provide the most recent optimal estimates of state variables, which can then be used to forecast and value European options.

Under the assumption that future futures prices are log-normally distributed under the risk-neutral process, there exist analytic expressions of the value of European call and put options on futures contracts. The value of a European option on a futures contract is given by calculating the current expected futures price and the average instantaneous variance of the futures return innovations over the life of the option.

Consider a European option with strike price K and a risk-free interest rate of r_f. The option maturity is at time T_0 and futures maturity at time T_1. The particular model features a state vector of length N (i.e., N-factors) $x(t)$

The value of a European call option would thus be:

$$e^{-rT_0}.E^*[max(F(x(T_0), T_0, T_1) - K, 0)]$$

The analytic solution to call and put options are given by:

Call options:

$$e^{-rT_0}(F(x(0), 0, T_1).N(d_1) - K.N(d_2))$$

Put options:

$$e^{-rT_0}(K.N(-d_2) - F(x(0), 0, T_1).N(-d_1))$$

Where:

Where:

$$d_1 = \frac{\ln(F/K) + \frac{1}{2}v^2}{v}$$

$$d_2 = d_1 - v$$

Parameter $N(d)$ indicates cumulative probabilities for the standard normal distribution (i.e. $P(Z < d)$).

Finally, parameter v, the annualized option volatility, is given by:

$$Var^*[\ln(F(x(T_0), T_0, T_1))] \equiv v^2 = \sum_{i.j=1} e^{(-\kappa_i + \kappa_j)(T_1 - T_0)} Cov^*(x_i(T_0), x_j(T_0))$$

The annualized option volatility approaches $\sigma_1^2 T_0$ as both T_0 and T_1 increase, as most uncertainty about spot prices at futures contract maturity and option expiration are a result of uncertainty about spot prices, rather than the cost of carry (Schwartz and Smith, 2000).

The presented option valuation formulas are analogous to the Black-Scholes formulas for valuing European options on stocks that do not pay dividends

When `verbose = T`, the `European_option_value` function numerically calculates the sensitivity of option prices to underlying option and model parameters. Gradients are calculated numerically through the `grad` function of the `numDeriv` package.

Value

The European_option_value function returns a numeric value corresponding to the present value of an option when verbose = F. When verbose = T, European_option_value returns a list with three objects:

option value	Present value of the option.
annualized volatility	Annualized volatility of the option.
parameter sensitivity	Sensitivity of option value to model parameters.
greeks	Sensitivity of option value to option parameters.

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Paul Gilbert and Ravi Varadhan (2016). numDeriv: Accurate Numerical Derivatives. R package version 2016.8-1. https://CRAN.R-project.org/package=numDeriv

Examples

```
##Example 1 - A European 'put' option on a futures contract following 'GBM'

European_option_value(x_0 = log(20), parameters = c(mu_rn = 0.06, sigma_1 = 0.2),
                      futures_maturity = 1, option_maturity = 1,
                      K = 20, r = 0.06, call = FALSE, verbose = TRUE)

##Example 2 - A European put option under a two-factor crude oil model:

##Step 1 - Obtain current (i.e. most recent) state vector by filtering the
##two-factor oil model:
Schwartz_Smith_oil <- NFCP_Kalman_filter(parameter_values = SS_oil$two_factor,
                                         parameter_names = names(SS_oil$two_factor),
                                         log_futures = log(SS_oil$stitched_futures),
                                         dt = SS_oil$dt,
                                         futures_TTM = SS_oil$stitched_TTM,
                                         verbose = TRUE)

##Step 2 - Calculate 'call' option price:
European_option_value(x_0 = Schwartz_Smith_oil$x_t,
                      parameters = SS_oil$two_factor,
                      futures_maturity = 2,
                      option_maturity = 1,
                      K = 20,
                      r = 0.05,
                      call = FALSE,
                      verbose = FALSE)
```

`futures_price_forecast` *Forecast the futures prices of an N-factor model*

Description

Analytically forecast future expected Futures prices under the risk-neutral version of a specified N-factor model.

Usage

```
futures_price_forecast(
  x_0,
  parameters,
  t = 0,
  futures_TTM = 1:10,
  percentiles = NULL
)
```

Arguments

`x_0`	`vector`. Initial values of the state variables, where the length must correspond to the number of factors specified in the parameters.
`parameters`	`vector`. A named vector of parameter values of a specified N-factor model. Function `NFCP_parameters` is recommended.
`t`	`numeric`. The time point, in years, at which to forecast futures prices.
`futures_TTM`	`vector`. the time-to-maturity, in years, of futures contracts to forecast.
`percentiles`	`vector`. Optional. Probabilistic forecasting percentile intervals.

Details

Under the assumption or risk-neutrality, futures prices are equal to the expected future spot price. Additionally, under deterministic interest rates, forward prices are equal to futures prices. Let $F_{T,t}$ denote the market price of a futures contract at time t with time T until maturity. let * denote the risk-neutral expectation and variance of futures prices. The following equations assume that the first factor follows a Brownian Motion.

$$E^*[ln(F_{T,t})] = season(T) + \sum_{i=1}^{N} e^{-\kappa_i T} x_i(0) + \mu^* t + A(T-t)$$

Where:

$$A(T-t) = \mu^*(T-t) - \sum_{i=1}^{N} - \frac{1-e^{-\kappa_i(T-t)}\lambda_i}{\kappa_i} + \frac{1}{2}(\sigma_1^2(T-t) + \sum_{i.j\neq 1} \sigma_i\sigma_j\rho_{i,j}\frac{1-e^{-(\kappa_i+\kappa_j)(T-t)}}{\kappa_i+\kappa_j})$$

The variance is given by:

$$Var^*[ln(F_{T,t})] = \sigma_1^2 t + \sum_{i.j\neq 1} e^{-(\kappa_i+\kappa_j)(T-t)}\sigma_i\sigma_j\rho_{i,j}\frac{1-e^{-(\kappa_i+\kappa_j)t}}{\kappa_i+\kappa_j}$$

Value

futures_price_forecast returns a vector of expected Futures prices under a given N-factor model with specified time to maturities at time t. When percentiles are specified, the function returns a matrix with the corresponding confidence bands in each column of the matrix.

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
# Forecast futures prices of the Schwartz and Smith (2000) two-factor oil model:
## Step 1 - Run the Kalman filter for the two-factor oil model:
SS_2F_filtered <- NFCP_Kalman_filter(parameter_values = SS_oil$two_factor,
                                     parameter_names = names(SS_oil$two_factor),
                                     log_futures = log(SS_oil$stitched_futures),
                                     dt = SS_oil$dt,
                                     futures_TTM = SS_oil$stitched_TTM,
                                     verbose = TRUE)

## Step 2 - Probabilistic forecast of the risk-neutral two-factor
## stochastic differential equation (SDE):
futures_price_forecast(x_0 = SS_2F_filtered$x_t,
                       parameters = SS_oil$two_factor,
                       t = 0,
                       futures_TTM = seq(0,9,1/12),
                       percentiles = c(0.1, 0.9))
```

---

futures_price_simulate

Simulate futures prices of an N-factor model through Monte Carlo simulation

---

Description

Simulate Futures price data with dynamics that follow the parameters of an N-factor model through Monte Carlo simulation.

Usage

```
futures_price_simulate(
  x_0,
  parameters,
  dt,
  N_obs,
  futures_TTM,
  ME_TTM = NULL,
  verbose = TRUE
)
```

Arguments

x_0	vector. Initial values of the state variables, where the length must correspond to the number of factors specified in the parameters.
parameters	vector. A named vector of parameter values of a specified N-factor model. Function NFCP_parameters is recommended.
dt	numeric. Discrete time step, in years, of the Monte Carlo simulation.
N_obs	numeric. Number of discrete observations to simulate.
futures_TTM	vector or matrix. The time-to-maturity of observed futures contracts, in years, at a given observation date. This time-to-maturity can either be constant (ie. class 'vector') or variable (ie. class 'matrix') across observations. The number of rows of object 'futures_TTM' must be either 1 or equal to argument 'N_obs'. NA values are allowed.
ME_TTM	vector. the time-to-maturity groupings to consider for observed futures prices. The length of ME_TTM must be equal to the number of 'ME' parameters specified in object 'parameter_names'. The maximum of 'ME_TTM' must be greater than the maximum value of 'futures_TTM'. When the number of 'ME' parameter values is equal to one or the number of columns of object 'log_futures', this argument is ignored.
verbose	logical. Should simulated state variables be output?

Details

The futures_price_simulate function simulates futures price data using the Kalman filter algorithm, drawing from a normal distribution for the shocks in the transition and measurement equations at each discrete time step. At each discrete time point, an observation of the state vector is generated through the transition equation, drawing from a normal distribution with a covariance equal to Q_t. Following this, simulated futures prices are generated through the measurement equation, drawing from a normal distribution with covariance matrix equal to H.

Input futures_TTM can be either a matrix specifying the constant time to maturity of futures contracts to simulate, or it can be a matrix where nrow(futures_TTM) == N_obs for the time-varying time to maturity of the futures contracts to simulate. This allows for the simulation of both aggregate stitched data and individual futures contracts.

Value

futures_price_simulate returns a list with three objects when verbose = T and a matrix of simulated futures prices when verbose = F. The list objects returned are:

#'

state_vector	A matrix of Simulated state variables at each discrete time point. The columns represent each factor
futures_prices	A matrix of Simulated futures prices, with each column representing a simulated futures contract.
spot_prices	A vector of simulated spot prices

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
# Example 1 - Simulate Crude Oil with constant time-to-maturity:

simulated_futures <- futures_price_simulate(x_0 = c(log(SS_oil$spot[1,1]), 0),
                                            parameters = SS_oil$two_factor,
                                            dt = SS_oil$dt,
                                            N_obs = nrow(SS_oil$stitched_futures),
                                            futures_TTM = SS_oil$stitched_TTM)

##Simulate Crude Oil Contracts with a rolling-window of measurement error:

simulated_futures_prices <- futures_price_simulate(x_0 = c(log(SS_oil$spot[1,1]), 0),
                                                   parameters = SS_oil$two_factor,
                                                   dt = SS_oil$dt,
                                                   N_obs = nrow(SS_oil$contracts),
                                                futures_TTM = SS_oil$contract_maturities,
                                                   ME_TTM = c(1/4, 1/2, 1, 2, 5))
```

NFCP_domains *N-Factor MLE search boundaries*

Description

Generate boundaries for the domain of parameters of the N-factor model for parameter estimation.

Usage

```
NFCP_domains(
  parameters,
  kappa = NULL,
  lambda = NULL,
  sigma = NULL,
  mu = NULL,
  mu_rn = NULL,
  rho = NULL,
  season = NULL,
  ME = NULL,
  x_0 = NULL,
  E = NULL
)
```

Arguments

parameters	a vector of parameter names of an N-factor model. Function NFCP_parameters is recommended.
kappa	A vector of length two specifying the lower and upper bounds for the 'kappa' parameter
lambda	A vector of length two specifying the lower and upper bounds for the 'lambda' parameter

sigma	A vector of length two specifying the lower and upper bounds for the 'sigma' parameter
mu	A vector of length two specifying the lower and upper bounds for the 'mu' parameter
mu_rn	A vector of length two specifying the lower and upper bounds for the 'mu_rn' parameter
rho	A vector of length two specifying the lower and upper bounds for the 'rho' parameter
season	A vector of length two specifying the lower and upper bounds for the 'season' parameter
ME	A vector of length two specifying the lower and upper bounds for the 'ME' (i.e., measurement error) parameter
x_0	A vector of length two specifying the lower and upper bounds for the 'x_0' parameter
E	A vector of length two specifying the lower and upper bounds for the 'E' parameter

Details

The NFCP_domains function generates lower and upper bounds for the parameter estimation procedure in the format required of the 'Domains' argument of the 'genoud' function. NFCP_domains allows easy setting of custom boundaries for parameter estimation, whilst also providing default domains of parameters.

Value

A matrix of defaulted domains for the given unknown parameters. The first column corresponds to the lower bound of the allowable search space for the parameter, whilst the second column corresponds to the upper bound. These values were set to allow for the 'realistic' possible values of given parameters as well as restricting some parameters (such as variance and mean-reverting terms) from taking negative values. The format of the returned matrix matches that required by the Domains argument of the Genoud function from the package RGenoud.

References

Mebane, W. R., and J. S. Sekhon, (2011). Genetic Optimization Using Derivatives: The rgenoud Package for R. *Journal of Statistical Software*, 42(11), 1-26. URL http://www.jstatsoft.org/v42/i11/.

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Examples

```
##Specify the Schwartz and Smith (2000) two-factor model
##with fixed measurement error:
parameters_2F <- NFCP_parameters(N_factors = 2,
                                 GBM = TRUE,
                                 initial_states = TRUE,
                                 N_ME = 1)

###Generate the default 'domains' argument of 'NFCP_MLE' function:
NFCP_MLE_bounds <- NFCP_domains(parameters_2F)
```

NFCP_Kalman_filter *Filter an N-factor commodity pricing model though the Kalman filter*

Description

Given a set of parameters of the N-factor model, filter term structure data using the Kalman filter.

Usage

```
NFCP_Kalman_filter(
  parameter_values,
  parameter_names,
  log_futures,
  dt,
  futures_TTM,
  ME_TTM = NULL,
  verbose = FALSE,
  debugging = FALSE
)
```

Arguments

parameter_values	vector. Numeric parameter values of an N-factor model.
parameter_names	vector. Parameter names, where each element of parameter_names must correspond to its respective value element in object parameter_values.
log_futures	matrix. The natural logarithm of observed futures prices. Each row must correspond to quoted futures prices at a particular date and every column must correspond to a unique futures contract. NA values are allowed.
dt	numeric. Constant, discrete time step of observations, in years.
futures_TTM	vector or matrix. The time-to-maturity of observed futures contracts, in years, at a given observation date. This time-to-maturity can either be constant (ie. class 'vector') or variable (ie. class 'matrix') across observations. The number of columns of 'futures_TTM' must be identical to the number of columns of object 'log_futures'. The number of rows of object 'futures_TTM' must be either 1 or equal to the number of rows of object 'log_futures'.
ME_TTM	vector. the time-to-maturity groupings to consider for observed futures prices. The length of ME_TTM must be equal to the number of 'ME' parameters specified in object 'parameter_names'. The maximum of 'ME_TTM' must be greater than the maximum value of 'futures_TTM'. When the number of 'ME' parameter values is equal to one or the number of columns of object 'log_futures', this argument is ignored.
verbose	logical. Should additional information be output? see **values**. When verbose = F, the NFCP_Kalman_filter function is significantly faster, see **details**.
debugging	logical. Should additional filtering information be output? see **values**.

Details

NFCP_Kalman_filter applies the Kalman filter algorithm for observable log_futures prices against the input parameters of an N-factor model provided through the parameter_values and parameter_names input vectors.

The NFCP_Kalman_filter function is designed for subsequent input into optimization functions and is called within the N-factor parameter estimation function NFCP_MLE. The first input to the NFCP_Kalman_filter function is a vector of parameters of an N-factor model, with elements of this vector corresponding to the parameter names within the elements of input vector parameter_names. When logical input verbose = F, the NFCP_Kalman_filter function calls the fkf_SP function of the FKF_SP package, which itself is a wrapper of a routine of the Kalman filter written in C utilizing Sequential Processing for maximum computational efficiency (see fkf_SP for more details). When verbose = T, the NFCP_Kalman_filter instead applies a Kalman filter algorithm written in base R and outputs several other list objects, including filtered values and measures for model fit and robustness (see **Returns**)

The N-factor model The N-factor framework was first presented in the work of Cortazar and Naranjo (2006, equations 1-3). It is a risk-premium class of commodity pricing model, in which futures prices are given by discounted expected future spot prices, where these spot prices are discounted at a given level of risk-premium, known as the cost-of-carry.

The N-factor framework describes the spot price process of a commodity as the correlated sum of N state variables x_t. The 'NFCP' package also allows for a deterministic, cyclical seasonal function $season(t)$ to be considered.

When GBM = TRUE:

$$log(S_t) = season(t) + \sum_{i=1}^{N} x_{i,t}$$

When GBM = FALSE:

$$log(S_t) = E + season(t) + \sum_{i=1}^{N} x_{i,t}$$

Where GBM determines whether the first factor follows a Brownian Motion or Ornstein-Uhlenbeck process to induce a unit root in the spot price process.

When GBM = TRUE, the first factor corresponds to the spot price, and additional N-1 factors model the cost-of-carry.

When GBM = FALSE, the commodity model assumes that there is a long-term equilibrium the commodity price will tend towards over time, with model volatility a decreasing function of time. This is not the standard approach made in the commodity pricing literature (Cortazar and Naranjo, 2006).

State variables are thus assumed to follow the following processes:

When GBM = TRUE:

$$dx_{1,t} = \mu^* dt + \sigma_1 dw_1 t$$

When GBM = FALSE:

$$dx_{1,t} = -(\lambda_1 + \kappa_1 x_{1,t})dt + \sigma_1 dw_1 t$$

And:

$$dx_{i,t} =_{i \neq 1} -(\lambda_i + \kappa_i x_{i,t})dt + \sigma_i dw_i t$$

where:

$$E(w_i)E(w_j) = \rho_{i,j}$$

Additionally, the deterministic seasonal function (if specified) is given by:

$$season(t) = \sum_{i=1}(season_{i,1}cos(2i\pi) + season_{i,2}sin(2i\pi)$$

The addition of deterministic, cyclical seasonality as a function of trigonometric variables was first suggested by Hannan, Terrell, and Tuckwell (1970) and first applied to model commodities by Sørensen (2002).

The following constant parameters are defined as:

`var` μ: long-term growth rate of the Brownian Motion process.

`var` E: Constant equilibrium level.

`var` $\mu^* = \mu - \lambda_1$: Long-term risk-neutral growth rate

`var` λ_i: Risk premium of state variable i.

`var` κ_i: Reversion rate of state variable i.

`var` σ_i: Instantaneous volatility of state variable i.

`var` $\rho_{i,j} \in [-1, 1]$: Instantaneous correlation between state variables i and j.

Including additional factors within the spot-price process allow for additional flexibility (and possibly fit) to the term structure of a commodity. The N-factor model nests simpler models within its framework, allowing for the fit of different N-factor models (applied to the same term structure data), represented by the log-likelihood, to be directly compared with statistical testing possible through a chi-squared test.

Disturbances - Measurement Error:

The Kalman filtering algorithm assumes a given measure of measurement error or disturbance in the measurement equation (ie. matrix H). Measurement errors can be interpreted as error in the model's fit to observed prices, or as errors in the reporting of prices (Schwartz and Smith, 2000). These disturbances are typically assumed independent.

`var` ME_i measurement error of contract i.

where the measurement error of futures contracts ME_i is equal to `'ME_'` [i] (i.e. `'ME_1'`, `'ME_2'`, ...) specified in arguments `parameter_values` and `parameter_names`.

There are three particular cases on how the measurement error of observations can be treated in the `NFCP_Kalman_filter` function:

Case 1: Only one ME is specified. The Kalman filter assumes that the measurement error of observations are independent and identical.

Case 2: One ME is specified for every observed futures contract. The Kalman filter assumes that the measurement error of observations are independent and unique.

Case 3: A series of ME's are specified for a given grouping of maturities of futures contracts. The Kalman filter assumes that the measurement error of observations are independent and unique to their respective time-to-maturity.

Grouping of maturities for case 3 is specified through the `ME_TTM` argument. This is a vector that specifies the maximum maturity to consider for each respective ME parameter argument.

in other words, ME_1 is considered for observations with TTM less than ME_TTM[1], ME_2 is considered for observations with TTM less than ME_TTM[2], ..., etc.

The first case is clearly the simplest to estimate, but can be a restrictive assumption. The second case is clearly the most difficult to estimate, but can be an infeasible assumption when considering all available futures contracts that make up the term structure of a commodity.

Case 3 thus serves to ease the restriction of case 1, and allow the user to make the modeling of measurement error as simple or complex as desired for a given set of maturities.

Kalman Filtering

The following section describes the Kalman filter equations used to filter the N-factor model.

The Kalman filter iteration is characterised by a transition and measurement equation. The transition equation develops the vector of state variables between discretised time steps (whilst considering a given level of covariance between state variables over time). The measurement equation relates the unobservable state vector to a vector of observable measurements (whilst also considering a given level of measurement error). The typical Kalman filter algorithm is a Gaussian process state space model.

Transition Equation:

$$\hat{x}_{t|t-1} = c_t + G_t\hat{x}_{t-1} + Q_t\eta_t$$

Measurement Equation:

$$\hat{y}_t = d_t + Z_t\hat{x}_{t|t-1} + H_t\epsilon_t$$

$$t = 1, \cdots, n$$

Where η_t and ϵ_t are IID $N(0, I(m))$ and iid $N(0, I(d))$ respectively.

The state vector follows a normal distribution, $x_1 \sim N(a_1, P_1)$, with a_1 and P_1 as the mean vector and variance matrix of the initial state vector x_1, respectively.

The Kalman filter can be used for parameter estimation through the maximization of the Log-Likelihood value. See `NFCP_MLE`.

Filtering the N-factor model

let m represent the number of observations at time t

let n represent the number of factors in the N-factor model

observable futures prices: $y_t = [ln(F(t, T_1)), ln(F(t, T_2)), \cdots, ln(F(t, T_m))]'$

State vector: $x_t = [x_1t, x_2t, \cdots, x_nt]'$

Measurement error: $diag(H) = [ME_1^2, ME_2^2, \cdots, ME_n^2]$

When the number of specified ME terms is one, $s_1 = s_2 = \cdots = s_n = ME_1^2$

`var` Z is an $m \times n$ matrix, where each element $[i, j]$ is equal to:

$$Z_{i,j} = e^{-\kappa_i T_j}$$

`var` d_t is an $m \times 1$ vector:

$$d_t = [season(T_1) + A(T_1), season(T_2) + A(T_2), \cdots, season(T_m) + A(T_m)]'$$

Under the assumption that Factor 1 follows a Brownian Motion, $A(T)$ is given by:

$$A(T) = \mu^* T - \sum_{i=1}^{N} - \frac{1 - e^{-\kappa_i T}\lambda_i}{\kappa_i} + \frac{1}{2}(\sigma_1^2 T + \sum_{i.j \neq 1} \sigma_i \sigma_j \rho_{i,j} \frac{1 - e^{-(\kappa_i + \kappa_j)T}}{\kappa_i + \kappa_j})$$

`var` v_t is a $n \times 1$ vector of serially uncorrelated Guassian disturbances with $E(V_t) = 0$ and $cov(v_t) = R^2$

Where:

$diag(G_t) = [e^{-\kappa_1 \tau}, e^{-\kappa_2 \tau}, \cdots, e^{-\kappa_n \tau}]$

Where $\tau = T - t$

var w_t is an $n \times 1$ vector of serially uncorrelated Guassian disturbances where:

$$E(w_t) = 0$$

and $cov(w_t) = Q_t$

var $c_t = [\mu \Delta t, 0, \cdots, 0]'$ is an $N \times 1$ vector of the intercept of the transition equation.

var Q_t is equal to the covariance function, given by:

$$Cov_{1,1}(x_{1,t}, x_{1,t}) = \sigma_1^2 t$$

$$Cov_{i,j}(x_{i,t}, x_{j,t}) = \sigma_i \sigma_j \rho_{i,j} \frac{1 - e^{-(\kappa_i + \kappa_j)t}}{\kappa_i + \kappa_j}$$

(see also cov_func)

Penalising poorly specified models

The Kalman filter returns non-real log-likelihood scores when the prediction error variance matrix becomes singular or its determinant becomes negative. This generally occurs when a poorly specified parameter set is input, such as when measurement error is zero. Non-real log-likelihood scores can break optimization and gradients algorithms and functions. To circumvent this, the NFCP_Kalman_filter returns a heavily penalized log-likelihood score when verbose = F. Penalized log-likelihood scores are calculated by:

```
stats::runif(1,-2e6,-1e6)
```

Diffuse Kalman filtering

If the initial values of the state vector are not supplied within the parameter_names and parameter_values vectors, a 'diffuse' assumption is used within the Kalman filtering algorithm. Initial states of factors that follow an Ornstein-Uhlenbeck are assumed to equal zero. The initial state of the first factor, given that it follows a Brownian motion, is assumed equal to the first element of log_futures. This is an assumption that the initial estimate of the spot price is equal to the closest to maturity observed futures price.

The initial states of factors that follow an Ornstein-Uhlenbeck have a transient effect on future observations. This makes the diffuse assumption reasonable and further means that initial states cannot generally be accurately estimated.

Value

NFCP_Kalman_filter returns a numeric object when verbose = F, which corresponds to the log-likelihood of observations. When verbose = T, the NFCP_Kalman_filter function returns a list object of length seven with the following objects:

Log-Likelihood	Log-Likelihood of observations.
Information Criteria	vector. The Akaikie and Bayesian Information Criterion.
X_t	vector. The final observation of the state vector.
X	matrix. Optimal one-step-ahead state vector.
Y	matrix. Estimated futures prices.
V	matrix. Estimation error.
Filtered Error	matrix. positive mean error (high bias), negative mean error (low bias), mean error (
Term Structure Fit	matrix. The mean error (Bias), mean absolute error, standard deviation of error and
Term Structure Volatility Fit	matrix. Theoretical and empirical volatility of observed futures contract returns.

When debugging = T, 9 objects are returned in addition to those returned when verbose = T:

P_t	array. Covariance matrix of state variables, with the third dimension indexing across time
F_t	vector. Prediction error variance matrix, with the third dimension indexing across time
K_t	matrix. Kalman Gain, with the third dimension indexing across time
d	matrix. d_t (see **details**)
Z	matrix. Z_t (see **details**)
G_t	matrix. G_t (see **details**)
c_t	vector. C_t (see **details**)
Q_t	matrix. Q_t (see **details**)
H	matrix. H (see **details**)

References

Hannan, E. J., et al. (1970). "The seasonal adjustment of economic time series." *International economic review*, 11(1): 24-52.

Anderson, B. D. O. and J. B. Moore, (1979). *Optimal filtering* Englewood Cliffs: Prentice-Hall.

Fahrmeir, L. and G. tutz,(1994) *Multivariate Statistical Modelling Based on Generalized Linear Models.* Berlin: Springer.

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Sørensen, C. (2002). "Modeling seasonality in agricultural commodity futures." *Journal of Futures Markets: Futures, Options, and Other Derivative Products* 22(5): 393-426.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Durbin, J., and S. J. Koopman, (2012). *Time series analysis by state space methods.* Oxford university press.

Examples

```
##Example 1 - complete, stitched data.
##Replicating the Schwartz and Smith (2000)
##Two-Factor commodity pricing model applied to crude oil:

SS_stitched_filtered <- NFCP_Kalman_filter(
parameter_values = SS_oil$two_factor,
parameter_names = names(SS_oil$two_factor),
log_futures = log(SS_oil$stitched_futures),
futures_TTM = SS_oil$stitched_TTM,
## maturity groupings need not be considered here:
ME_TTM = NULL,
dt = SS_oil$dt,
verbose = FALSE)


##Example 2 - incomplete, contract data.
##Replicating the Schwartz and Smith (2000)
##Two-Factor commodity pricing model applied to all available
##crude oil contracts:

SS_2F <- SS_oil$two_factor
##omit stitched contract white noise
```

```
SS_2F <- SS_2F[!grepl("ME",
             names(SS_2F))]

# Evaluate two different measurement errors
SS_2F[c("ME_1", "ME_2")] <- c(0.01, 0.04)

## Separate measurement error into two different maturity groupings
SS_ME_TTM <- c(1,3)
## ME_1 is applied for observed contracts with less than one year
## maturity, whilst ME_2 considers contracts with maturity greater
## than one year, and less than three years

#Kalman filter
SS_contract_filtered <- NFCP_Kalman_filter(
parameter_values = SS_2F,
parameter_names = names(SS_2F),
## All available contracts are considered
log_futures = log(SS_oil$contracts),
## Respective 'futures_TTM' of these contracts are input:
futures_TTM = SS_oil$contract_maturities,
ME_TTM = SS_ME_TTM,
dt = SS_oil$dt,
verbose = FALSE)
```

NFCP_MLE *N-factor model parameter estimation through the Kalman filter and maximum likelihood estimation*

Description

The NFCP_MLE function performs parameter estimation of commodity pricing models under the N-factor framework of Cortazar and Naranjo (2006). It uses term structure futures data and estimates unknown parameters through maximum likelihood estimation. NFCP_MLE allows for missing observations, a variable number of state variables, deterministic seasonality and a variable number of measurement error terms.

Usage

```
NFCP_MLE(
  log_futures,
  dt,
  futures_TTM,
  N_factors,
  N_season = 0,
  N_ME = 1,
  ME_TTM = NULL,
  GBM = TRUE,
  estimate_initial_state = FALSE,
  cluster = FALSE,
  ...
)
```

Arguments

log_futures	matrix. The natural logarithm of observed futures prices. Each row must correspond to quoted futures prices at a particular date and every column must correspond to a unique futures contract. NA values are allowed.
dt	numeric. Constant, discrete time step of observations, in years.
futures_TTM	vector or matrix. The time-to-maturity of observed futures contracts, in years, at a given observation date. This time-to-maturity can either be constant (ie. class 'vector') or variable (ie. class 'matrix') across observations. The number of columns of 'futures_TTM' must be identical to the number of columns of object 'log_futures'. The number of rows of object 'futures_TTM' must be either 1 or equal to the number of rows of object 'log_futures'.
N_factors	numeric. Number of state variables in the spot price process.
N_season	numeric. The number of deterministic, cyclical seasonal factors to include in the spot price process.
N_ME	numeric. The number of independent measuring errors of observable futures contracts to consider in the Kalman filter.
ME_TTM	vector. the time-to-maturity groupings to consider for observed futures prices. The length of ME_TTM must be equal to the number of 'ME' parameters specified in object 'parameter_names'. The maximum of 'ME_TTM' must be greater than the maximum value of 'futures_TTM'. When the number of 'ME' parameter values is equal to one or the number of columns of object 'log_futures', this argument is ignored.
GBM	logical. When TRUE, factor 1 of the model is assumed to follow a Brownian Motion, inducing a unit-root in the spot price process.
estimate_initial_state	logical. Should the initial state vector be specified as unknown parameters of the commodity pricing model? These are generally estimated with low precision (see **details**).
cluster	cluster. An optional object returned by one of the makeCluster commands in the parallel package to allow for parameter estimation to be performed across multiple cluster nodes.
...	additional arguments to be passed into the genoud genetic algorithm numeric optimization. These can highly influence the maximum likelihood estimation procedure. See help(genoud)

Details

The NFCP_MLE function facilitates parameter estimation of commodity pricing models under the N-factor framework through the Kalman filter and maximum likelihood estimation. NFCP_MLE uses genetic algorithms through the genoud function of the rgenoud package to numerically optimize the log-likelihood score returned from the NFCP_Kalman_filter function.

Parameter estimation of commodity pricing models can involve a large number of observations, state variables and unknown parameters. It also features an objective log-likelihood function that is nonlinear and discontinuous with respect to model parameters. NFCP_MLE is designed to perform parameter estimation as efficiently as possible, maximizing the likelihood of attaining a global optimum.

Arguments passed to the genoud function can greatly influence estimated parameters as well as computation time and must be considered when performing parameter estimation. All arguments of the genoud function may be passed through the NFCP_MLE function.

When grad is not specified, the grad function from the numDeriv package is called to approximate the gradient within the genoud optimization algorithm through Richardsons extrapolation.

Richardsons extrapolation is regarded for its ability to improve the approximation of estimation methods, which may improve the likelihood of obtained a global maxmimum estimate of the log-likelihood.

The population size can highly influence parameter estimates at the expense of increased computation time. For commodity pricing models with a large number of unknown parameters, large population sizes may be necessary to maximize the estimation process.

NFCP_MLE by default performs boundary constrained optimization of log-likelihood scores and does not allow does not allow for out-of-bounds evaluations within the genoud optimization process, preventing candidates from straying beyond the bounds provided by argument Domains.

When Domains is not specified, the default bounds specified by the NFCP_domains function are used. The size of the search domains of unknown parameters can highly influence the computation time of the NFCP_MLE function, however setting domains that are too restrictive may result in estimated parameters returned at the upper or lower bounds. Custom search domains can be used through the NFCP_domains function and subsequently the Domains argument of this function.

Finally, the maximum likelihood estimation process of parameters provides no in-built guarantee that the estimated parameters of commodity models are financially sensible results. When the commodity model has been over-parameterized (i.e., the number of factors N specified is too high) or the optimization algorithm has failed to attain a global maximum likelihood estimate, estimated parameters may be irrational.

Evidence of irrational parameter estimates include correlation coefficients that are extremely large (e.g., > 0.95 or < -0.95), risk-premiums or drift terms that are unrealistic, filtered state variables that are unrealistic and extremely large/small mean-reverting terms with associated large standard errors.

Irrational parameter estimates may indicate that the number of stochastic factors (i.e., N_factors) of the model or number of seasonal factors (i.e., N_season) are too high.

The N-factor model The N-factor framework was first presented in the work of Cortazar and Naranjo (2006, equations 1-3). It is a risk-premium class of commodity pricing model, in which futures prices are given by discounted expected future spot prices, where these spot prices are discounted at a given level of risk-premium, known as the cost-of-carry.

The N-factor framework describes the spot price process of a commodity as the correlated sum of N state variables x_t. The 'NFCP' package also allows for a deterministic, cyclical seasonal function $season(t)$ to be considered.

When GBM = TRUE:

$$log(S_t) = season(t) + \sum_{i=1}^{N} x_{i,t}$$

When GBM = FALSE:

$$log(S_t) = E + season(t) + \sum_{i=1}^{N} x_{i,t}$$

Where GBM determines whether the first factor follows a Brownian Motion or Ornstein-Uhlenbeck process to induce a unit root in the spot price process.

When GBM = TRUE, the first factor corresponds to the spot price, and additional N-1 factors model the cost-of-carry.

When GBM = FALSE, the commodity model assumes that there is a long-term equilibrium the commodity price will tend towards over time, with model volatility a decreasing function of time. This is not the standard approach made in the commodity pricing literature (Cortazar and Naranjo, 2006).

State variables are thus assumed to follow the following processes:

When `GBM = TRUE`:

$$dx_{1,t} = \mu^* dt + \sigma_1 dw_1 t$$

When `GBM = FALSE`:

$$dx_{1,t} = -(\lambda_1 + \kappa_1 x_{1,t})dt + \sigma_1 dw_1 t$$

And:

$$dx_{i,t} =_{i \neq 1} -(\lambda_i + \kappa_i x_{i,t})dt + \sigma_i dw_i t$$

where:

$$E(w_i)E(w_j) = \rho_{i,j}$$

Additionally, the deterministic seasonal function (if specified) is given by:

$$season(t) = \sum_{i=1} (season_{i,1} cos(2i\pi) + season_{i,2} sin(2i\pi)$$

The addition of deterministic, cyclical seasonality as a function of trigonometric variables was first suggested by Hannan, Terrell, and Tuckwell (1970) and first applied to model commodities by Sørensen (2002).

The following constant parameters are defined as:

`var` μ: long-term growth rate of the Brownian Motion process.

`var` E: Constant equilibrium level.

`var` $\mu^* = \mu - \lambda_1$: Long-term risk-neutral growth rate

`var` λ_i: Risk premium of state variable i.

`var` κ_i: Reversion rate of state variable i.

`var` σ_i: Instantaneous volatility of state variable i.

`var` $\rho_{i,j} \in [-1, 1]$: Instantaneous correlation between state variables i and j.

Including additional factors within the spot-price process allow for additional flexibility (and possibly fit) to the term structure of a commodity. The N-factor model nests simpler models within its framework, allowing for the fit of different N-factor models (applied to the same term structure data), represented by the log-likelihood, to be directly compared with statistical testing possible through a chi-squared test. The AIC or BIC can also be used to compare models.

Disturbances - Measurement Error:

The Kalman filtering algorithm assumes a given measure of measurement error or disturbance in the measurement equation (ie. matrix H). Measurement errors can be interpreted as error in the model's fit to observed prices, or as errors in the reporting of prices (Schwartz and Smith, 2000). These disturbances are typically assumed independent.

`var` ME_i measurement error of contract i.

where the measurement error of futures contracts ME_i is equal to `'ME_'` [i] (i.e. `'ME_1'`, `'ME_2'`, ...) specified in arguments `parameter_values` and `parameter_names`.

There are three particular cases on how the measurement error of observations can be treated in the `NFCP_Kalman_filter` function:

Case 1: Only one ME is specified. The Kalman filter assumes that the measurement error of observations are independent and identical.

Case 2: One ME is specified for every observed futures contract. The Kalman filter assumes that the measurement error of observations are independent and unique.

Case 3: A series of ME's are specified for a given grouping of maturities of futures contracts. The Kalman filter assumes that the measurement error of observations are independent and unique to their respective time-to-maturity.

Grouping of maturities for case 3 is specified through the `ME_TTM` argument. This is a vector that specifies the maximum maturity to consider for each respective ME parameter argument.

in other words, ME_1 is considered for observations with TTM less than ME_TTM[1], ME_2 is considered for observations with TTM less than ME_TTM[2], ..., etc.

The first case is clearly the simplest to estimate, but can be a restrictive assumption. The second case is clearly the most difficult to estimate, but can be an infeasible assumption when considering all available futures contracts that make up the term structure of a commodity.

Case 3 thus serves to ease the restriction of case 1, and allow the user to make the modeling of measurement error as simple or complex as desired for a given set of maturities.

Diffuse Kalman filtering

If the initial values of the state vector are not supplied within the `parameter_names` and `parameter_values` vectors, a 'diffuse' assumption is used within the Kalman filtering algorithm. Initial states of factors that follow an Ornstein-Uhlenbeck are assumed to equal zero. The initial state of the first factor, given that it follows a Brownian motion, is assumed equal to the first element of `log_futures`. This is an assumption that the initial estimate of the spot price is equal to the closest to maturity observed futures price.

The initial states of factors that follow an Ornstein-Uhlenbeck have a transient effect on future observations. This makes the diffuse assumption reasonable and further means that initial states cannot generally be accurately estimated.

Value

`NFCP_MLE` returns a `list` with 10 objects. 9 objects are returned when the user has specified not to calculate the hessian matrix at solution.

`MLE`	`numeric` The Maximum-Likelihood-Estimate of the solution.
`estimated_parameters`	`vector`. Estimated parameters.
`standard_errors`	`vector`. Standard error of the estimated parameters. Returned only when `hessian =`
`Information Criteria`	`vector`. The Akaikie and Bayesian Information Criterion.
`x_t`	`vector`. The final observation of the state vector.
`X`	`matrix`. Optimal one-step-ahead state vector.
`Y`	`matrix`. Estimated futures prices.
`V`	`matrix`. Estimation error.
`Filtered Error`	`matrix`. positive mean error (high bias), negative mean error (low bias), mean error (
`Term Structure Fit`	`matrix`. The mean error (Bias), mean absolute error, standard deviation of error and
`Term Structure Volatility Fit`	`matrix`. Theoretical and empirical volatility of observed futures contract returns
`proc_time`	`list`. The real and CPU time (in seconds) the `NFCP_MLE` function has taken.
`genoud_value`	`list`. Outputs of `genoud`.

References

Hannan, E. J., et al. (1970). "The seasonal adjustment of economic time series." *International economic review*, 11(1): 24-52.

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Sørensen, C. (2002). "Modeling seasonality in agricultural commodity futures." *Journal of Futures Markets: Futures, Options, and Other Derivative Products* 22(5): 393-426.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Mebane, W. R., and J. S. Sekhon, (2011). Genetic Optimization Using Derivatives: The rgenoud Package for R. *Journal of Statistical Software*, 42(11), 1-26. URL http://www.jstatsoft.org/v42/i11/.

Examples

```
# Estimate a 'one-factor' geometric Brownian motion model:
Oil_1F_estimated_model <- NFCP_MLE(
## Arguments
log_futures = log(SS_oil$contracts)[1:20,1:5],
dt = SS_oil$dt,
futures_TTM= SS_oil$contract_maturities[1:20,1:5],
N_factors = 1, N_ME = 1,
## Genoud arguments:
pop.size = 4, print.level = 0, gr = NULL,
max.generations = 0)
```

NFCP_parameters *Specify the constant parameters of an N-factor model*

Description

the NFCP_parameters function specifies the parameters of a commodity pricing model under the N-factor framework first described by Cortazar and Naranjo (2006). This function is a recommended starting position for the application of N-factor models within the NFCP package.

Usage

```
NFCP_parameters(
  N_factors,
  GBM,
  initial_states,
  N_ME,
  N_season = 0,
  verbose = TRUE
)
```

Arguments

N_factors	numeric. Number of state variables in the spot price process.
GBM	logical. If GBM = T, factor 1 of the model is assumed to follow a Brownian Motion, inducing a unit-root in the spot price process.
initial_states	logical. If initial_states = T, the initial state vector is specified as unknown parameters of the commodity pricing model.
N_ME	numeric. The number of independent measuring errors of observable futures contracts to consider in the Kalman filter.
N_season	numeric. The number of deterministic, cyclical seasonal factors to include in the spot price process.
verbose	logical. If verbose = T, the stochastic differential equation of the spot price process is printed when the function is called.

Details

The N-factor model The N-factor framework was first presented in the work of Cortazar and Naranjo (2006, equations 1-3). It is a risk-premium class of commodity pricing model, in which futures prices are given by discounted expected future spot prices, where these spot prices are discounted at a given level of risk-premium, known as the cost-of-carry.

The N-factor framework describes the spot price process of a commodity as the correlated sum of N state variables x_t. The 'NFCP' package also allows for a deterministic, cyclical seasonal function $season(t)$ to be considered.

When `GBM = TRUE`:

$$log(S_t) = season(t) + \sum_{i=1}^{N} x_{i,t}$$

When `GBM = FALSE`:

$$log(S_t) = E + season(t) + \sum_{i=1}^{N} x_{i,t}$$

Where `GBM` determines whether the first factor follows a Brownian Motion or Ornstein-Uhlenbeck process to induce a unit root in the spot price process.

When `GBM = TRUE`, the first factor corresponds to the spot price, and additional N-1 factors model the cost-of-carry.

When `GBM = FALSE`, the commodity model assumes that there is a long-term equilibrium the commodity price will tend towards over time, with model volatility a decreasing function of time. This is not the standard approach made in the commodity pricing literature (Cortazar and Naranjo, 2006).

State variables are thus assumed to follow the following processes:

When `GBM = TRUE`:

$$dx_{1,t} = \mu^* dt + \sigma_1 dw_1 t$$

When `GBM = FALSE`:

$$dx_{1,t} = -(\lambda_1 + \kappa_1 x_{1,t})dt + \sigma_1 dw_1 t$$

And:

$$dx_{i,t} =_{i \neq 1} -(\lambda_i + \kappa_i x_{i,t})dt + \sigma_i dw_i t$$

where:

$$E(w_i)E(w_j) = \rho_{i,j}$$

Additionally, the deterministic seasonal function (if specified) is given by:

$$season(t) = \sum_{i=1} (season_{i,1} cos(2i\pi) + season_{i,2} sin(2i\pi)$$

The addition of deterministic, cyclical seasonality as a function of trigonometric variables was first suggested by Hannan, Terrell, and Tuckwell (1970) and first applied to model commodities by Sørensen (2002).

The following constant parameters are defined as:

`var` μ: long-term growth rate of the Brownian Motion process.

`var` E: Constant equilibrium level.

`var` $\mu^* = \mu - \lambda_1$: Long-term risk-neutral growth rate

`var` λ_i: Risk premium of state variable i.

var κ_i: Reversion rate of state variable i.

var σ_i: Instantaneous volatility of state variable i.

var $\rho_{i,j} \in [-1, 1]$: Instantaneous correlation between state variables i and j.

Including additional factors within the spot-price process allow for additional flexibility (and possibly fit) to the term structure of a commodity. The N-factor model nests simpler models within its framework, allowing for the fit of different N-factor models (applied to the same term structure data), represented by the log-likelihood, to be directly compared with statistical testing possible through a chi-squared test.

Disturbances - Measurement Error:

The Kalman filtering algorithm assumes a given measure of measurement error or disturbance in the measurement equation (ie. matrix H). Measurement errors can be interpreted as error in the model's fit to observed prices, or as errors in the reporting of prices (Schwartz and Smith, 2000). These disturbances are typically assumed independent by the commodity pricing literature.

var ME_i measurement error of contract i.

where the measurement error of futures contracts ME_i is equal to `'ME_'` [i] (i.e. `'ME_1'`, `'ME_2'`, ...) specified in arguments `parameter_values` and `parameter_names`.

There are three particular cases on how the measurement error of observations can be treated in the `NFCP_Kalman_filter` function:

Case 1: Only one ME is specified. The Kalman filter assumes that the measurement error of observations are independent and identical.

Case 2: One ME is specified for every observed futures contract. The Kalman filter assumes that the measurement error of observations are independent and unique.

Case 3: A series of ME's are specified for a given grouping of maturities of futures contracts. The Kalman filter assumes that the measurement error of observations are independent and unique to their respective time-to-maturity.

Grouping of maturities for case 3 is specified through the `ME_TTM` argument. This is a vector that specifies the maximum maturity to consider for each respective ME parameter argument.

in other words, ME_1 is considered for observations with TTM less than ME_TTM[1], ME_2 is considered for observations with TTM less than ME_TTM[2], ..., etc.

The first case is clearly the simplest to estimate, but can be a restrictive assumption. The second case is clearly the most difficult to estimate, but can be an infeasible assumption when considering all available futures contracts that make up the term structure of a commodity.

Case 3 thus serves to ease the restriction of case 1, and allow the user to make the modeling of measurement error as simple or complex as desired for a given set of maturities.

Value

A vector of parameter names for a specified N-factor spot price process. This vector is ideal for application within many other functions within the `NFCP` package

References

Hannan, E. J., et al. (1970). "The seasonal adjustment of economic time series." *International economic review* 11(1): 24-52.

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Sørensen, C. (2002). "Modeling seasonality in agricultural commodity futures." *Journal of Futures Markets: Futures, Options, and Other Derivative Products* 22(5): 393-426.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
##Generate parameter of a Two-factor model Crude Oil model
##as first presented by Schwartz and Smith (2000):
two_factor_parameters <- NFCP_parameters(N_factors = 2,
                                        GBM = TRUE,
                                        initial_states = FALSE,
                                        N_ME = 5)
print(two_factor_parameters)
```

---

spot_price_forecast *Forecast spot prices of an N-factor model*

Description

Analytically forecast expected spot prices following the "true" process of a given n-factor stochastic model

Usage

```
spot_price_forecast(x_0, parameters, t, percentiles = NULL)
```

Arguments

x_0	vector. Initial values of the state variables, where the length must correspond to the number of factors specified in the parameters.
parameters	vector. A named vector of parameter values of a specified N-factor model. Function NFCP_parameters is recommended.
t	vector. Discrete time points, in years, to forecast spot prices
percentiles	vector. Optional. Probabilistic forecasting percentile intervals.

Details

Future expected spot prices under the N-factor model can be forecasted through the analytic expression of expected future prices under the "true" N-factor process.

Given that the log of the spot price is equal to the sum of the state variables (equation 1), the spot price is log-normally distributed with the expected prices given by:

$$E[S_t] = exp(E[ln(S_t)] + \frac{1}{2}Var[ln(S_t)])$$

Where:

$$E[ln(S_t)] = season(t) + \sum_{i=1}^{N} e^{-(\kappa_i t)} x_i(0) + \mu t$$

Where $\kappa_i = 0$ when GBM=T and $\mu = 0$ when GBM = F

$$Var[ln(S_t)] = \sigma_1^2 t + \sum_{i.j \neq 1} \sigma_i \sigma_j \rho_{i,j} \frac{1 - e^{-(\kappa_i + \kappa_j)t}}{\kappa_i + \kappa_j}$$

and thus:

$$E[S_t] = exp(season(t) + \sum_{i=1}^{N} e^{-\kappa_i t} x_i(0) + (\mu + \frac{1}{2}\sigma_1^2)t + \frac{1}{2} \sum_{i.j \neq 1} \sigma_i \sigma_j \rho_{i,j} \frac{1 - e^{-(\kappa_i + \kappa_j)t}}{\kappa_i + \kappa_j})$$

Under the assumption that the first factor follows a Brownian Motion, in the long-run expected spot prices grow over time at a constant rate of $\mu + \frac{1}{2}\sigma_1^2$ as the $e^{-\kappa_i t}$ and $e^{-(\kappa_i + \kappa_j)t}$ terms approach zero.

An important consideration when forecasting spot prices using parameters estimated through maximum likelihood estimation is that the parameter estimation process takes the assumption of risk-neutrality and thus the true process growth rate μ is not estimated with a high level of precision. This can be shown from the higher standard error for μ than other estimated parameters, such as the risk-neutral growth rate μ^*. See Schwartz and Smith (2000) for more details.

Value

`spot_price_forecast` returns a vector of expected future spot prices under a given N-factor model at specified discrete future time points. When `percentiles` are specified, the function returns a matrix with the corresponding confidence bands in each column of the matrix.

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
# Forecast the Schwartz and Smith (2000) two-factor oil model:

##Step 1 - Kalman filter of the two-factor oil model:
SS_2F_filtered <- NFCP_Kalman_filter(SS_oil$two_factor,
                                     names(SS_oil$two_factor),
                                     log(SS_oil$stitched_futures),
                                     SS_oil$dt,
                                     SS_oil$stitched_TTM,
                                     verbose = TRUE)

##Step 2 - Probabilistic forecast of N-factor stochastic differential equation (SDE):
spot_price_forecast(x_0 = SS_2F_filtered$x_t,
                    parameters = SS_oil$two_factor,
                    t = seq(0,9,1/12),
                    percentiles = c(0.1, 0.9))
```

spot_price_simulate *Simulate spot prices of an N-factor model through Monte Carlo simulation*

Description

Simulate risk-neutral price paths of an an N-factor commodity pricing model through Monte Carlo Simulation.

Usage

```
spot_price_simulate(
  x_0,
  parameters,
  t = 1,
  dt = 1,
  N_simulations = 2,
  antithetic = TRUE,
  verbose = FALSE
)
```

Arguments

x_0	vector. Initial values of the state variables, where the length must correspond to the number of factors specified in the parameters.
parameters	vector. A named vector of parameter values of a specified N-factor model. Function NFCP_parameters is recommended.
t	numeric. Number of years to simulate.
dt	numeric. Discrete time step, in years, of the Monte Carlo simulation.
N_simulations	numeric. The total number of Monte Carlo simulations.
antithetic	logical. Should antithetic price paths be simulated?
verbose	logical. Should simulated state variables be output?

Details

The spot_price_simulate function is able to quickly and efficiently simulate a large number of state variables and risk-neutral price paths of a commodity following the N-factor model. Simulating risk-neutral price paths of a commodity under an N-factor model through Monte Carlo simulations allows for the valuation of commodity related investments and derivatives, such as American options and real Options through dynamic programming methods. The spot_price_simulate function quickly and efficiently simulates an N-factor model over a specified number of years, simulating antithetic price paths as a simple variance reduction technique. The spot_price_simulate function uses the mvrnorm function from the MASS package to draw from a multivariate normal distribution for the correlated simulation shocks of state variables.

The N-factor model stochastic differential equation is given by:

Brownian Motion processes (ie. factor one when GBM = T) are simulated using the following solution:

$$x_{1,t+1} = x_{1,t} + \mu^* \Delta t + \sigma_1 \Delta t Z_{t+1}$$

Where Δt is the discrete time step, μ^* is the risk-neutral growth rate and σ_1 is the instantaneous volatility. Z_t represents the independent standard normal at time t.

Ornstein-Uhlenbeck Processes are simulated using the following solution:

$$x_{i,t} = x_{i,0}e^{-\kappa_i t} - \frac{\lambda_i}{\kappa_i}(1 - e^{-\kappa_i t}) + \int_0^t \sigma_i e^{\kappa_i s} dW_s$$

Where a numerical solution is obtained by numerically discretising and approximating the integral term using the Euler-Maruyama integration scheme:

$$\int_0^t \sigma_i e^{\kappa_i s} dW_s = \sum_{j=0}^{t} \sigma_i e^{\kappa_i j} dW_s$$

Finally, deterministic seasonality is considered within the spot prices of simulated price paths.

Value

`spot_price_simulate` returns a list when `verbose = T` and a matrix of simulated price paths when `verbose = F`. The returned objects in the list are:

`State_Variables`	A matrix of simulated state variables for each factor is returned when `verbose = T`. The number of fa
`Prices`	A matrix of simulated price paths. Each column represents one simulated price path and each row rep

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
# Example 1
## Simulate a geometric Brownian motion (GBM) process:
simulated_spot_prices <- spot_price_simulate(
 x_0 = log(20),
 parameters = c(mu_rn = (0.05 - (1/2) * 0.2^2), sigma_1 = 0.2),
 t = 1,
 dt = 1/12,
 N_simulations = 1e3)

# Example 2
## Simulate the Short-Term/Long-Term model:

### Step 1 - Obtain contemporary state variable estimates through the Kalman Filter:
SS_2F_filtered <- NFCP_Kalman_filter(parameter_values = SS_oil$two_factor,
                                     parameter_names = names(SS_oil$two_factor),
                                     log_futures = log(SS_oil$stitched_futures),
                                     dt = SS_oil$dt,
                                     futures_TTM = SS_oil$stitched_TTM,
```

```
                                    verbose = TRUE)

### Step 2 - Use these state variable estimates to simulate futures spot prices:
simulated_spot_prices <- spot_price_simulate(
 x_0 = SS_2F_filtered$x_t,
 parameters = SS_oil$two_factor,
 t = 1,
 dt = 1/12,
 N_simulations = 1e3,
 antithetic = TRUE,
 verbose = TRUE)
```

SS_oil *Crude oil term structure futures data (1990 - 1995)*

Description

The SS_oil list object features the approximate weekly observations of Crude Oil (WTI) futures contracts used to develop a two-factor commodity pricing model within the prominent work of Schwartz and Smith (2000) titled: "Short-Term Variations and long-Term Dynamics in Commodity Prices". The two-factor commodity pricing model presented within this study is also included. The SS_oil list object is used extensively within the NFCP package to provide working examples and showcase the features of the package.

Usage

```
data(SS_oil)
```

Format

A list Containing eight objects:

contracts A data frame with 268 rows and 82 columns. Each column represents a Crude Oil futures contract, and each row represents a closing weekly price for that futures contract. Observation dates of the contract object are weekly in frequency from 1990-02-06 to 1995-02-14. Contracts without observations on a particular date are represented as NA.

stitched_futures Schwartz and Smith (2000) applied stitched contract observation data to estimate commodity pricing models, which are approximated within this object. The stitched_futures object was developed using the stitch_contracts function (see stitch_contracts examples for more details). Contracts were stitched according to the contract numbers specified within the object stitched_TTM. stitched_futures is identical to the futures data made available within the MATLAB program "SchwartzSmithModel" developed by Goodwin (2013).

spot A data.frame of spot prices of Crude Oil. weekly in frequency from 1990-02-06 to 1995-02-14.

final_trading_days Named vector listing the final trading days of each observed futures contract within the contracts object. Each element of final_trading_days corresponds to a column of the contracts object. The final trading day of a futures contract is used to calculate the number of business days from a given observation to the maturity of the contract (ie. a contract time to maturity).

contract_maturities A data frame with identical dimensions to the `contracts` data frame. This data frame lists the time to maturity of a given futures contract in years at each observation point. This is identical to the number of business days (in years) between the observed date and the final trading day of a particular futures contract. The maturity matrix assumes 262 trading days a year. If the contract is not yet available or has expired, the `contract_maturities` element is `NA`.

stitched_TTM A vector corresponding to the constant time to maturities that was assumed within the original study of Schwartz and Smith (2000).

dt The discrete time step used to estimate parameters with this data. The time step is 5/262, which represents a weekly frequency of observations where each weekday is a business day (ie. there are no business days on weekends).

two_factor The crude oil two-factor commodity pricing model parameters presented within the work of Schwartz and Smith (2000). These parameter estimates are prolific, benchmarked within several subsequent publications.

References

Dominice Goodwin (2013). Schwartz-Smith 2-factor model - Parameter estimation (https://www.mathworks.com/matlab schwartz-smith-2-factor-model-parameter-estimation), MATLAB Central File Exchange. Retrieved November 21, 2020.

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

`stitch_contracts` *Stitch futures contracts*

Description

Aggregate futures contract price data by stitching according to either approximate maturities and rollover frequency or contract number from closest maturity.

Usage

```
stitch_contracts(
  futures,
  futures_TTM = NULL,
  maturity_matrix = NULL,
  rollover_frequency = NULL,
  contract_numbers = NULL,
  verbose = FALSE
)
```

Arguments

`futures`	Contract futures price data. Each row of `Futures` should represent one observation of futures prices and each column should represent one quoted futures contract. NA's in `Futures` are allowed, representing missing observations.
`futures_TTM`	A `vector` of contract maturities to stitch

`maturity_matrix`
: The time-to-maturity (in years) for each contract at each given observation point. The dimensions of `maturity_matrix` should match those of `Futures`

`rollover_frequency`
: the frequency (in years) at which contracts should be rolled over

`contract_numbers`
: A `vector` of contract numbers offset from the closest-to-maturity contract at which to stitch contracts.

`verbose`
: `logical`. Should additional information be output? see **details**

Details

This function aggregates a set of futures contract data by stitching contract data over an observation period, resulting in a set of futures observations that is 'complete' (ie. Does not feature missing observations). Aggregated futures data benefit from several computational efficiencies compared to raw contract data, but results in the loss of futures price information.

There are two methods of the `stitch_contracts` function that can be utilized the stitch contracts:

Method 1

`stitch_contracts(futures,contract_numbers,verbose = T)` Futures data may be aggregated by stitching prices according to maturity matching. This method requires the inputs `futures_TTM`, `maturity_matrix` and `rollover_frequency`. This method stitched contracts by matching the observation prices according to which contract has the closest time-to-maturity of the desired maturity specified in `futures_TTM`. Contracts are rolled over at the frequency specified in `rollover_frequency`.

Method 2

`stitch_contracts(futures,futures_TTM,maturity_matrix,rollover_frequency,verbose = T)` Futures data may be stitched according to the contract numbers offset from the closest-to-maturity contract. This method requires only the input `contract_numbers` specifying which contracts should be included. This method is most appropriate when the maturity of available contracts are consistent (ie. contracts expire every month or three months).

Value

`stitch_contracts` returns a matrix of stitched futures prices if `verbose = T` and a list with two or three objects otherwise (see below).

`prices` A data frame of Stitched futures prices. Each row represents an observation of the specified contracts.
`maturities` A data frame of the time-to-maturity of observed futures prices. Each row represents an observation of the
`tickers` A data frame of the named columns of observed futures prices (e.g. contract tickers). Returned only when

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
##These examples approximately replicate the Crude Oil data utilized within the
##prominent work of Schwartz and Smith (2000):

###Method 1 - Stitch crude oil contracts according to maturity matching:
SS_stitched_M1 <- stitch_contracts(futures = SS_oil$contracts,
                                   futures_TTM = c(1, 5, 9, 13, 17)/12,
                                   maturity_matrix = SS_oil$contract_maturities,
                                   rollover_frequency = 1/12, verbose = TRUE)

###Method 2 - Stitch crude oil contracts according to nearest contract numbers:
SS_stitched_M2 <- stitch_contracts(futures = SS_oil$contracts,
                                   contract_numbers = c(1, 5, 9, 13, 17), verbose = TRUE)
```

`TSfit_volatility` *Calculate the volatility term structure of futures returns*

Description

Estimate the theoretical and empirical volatility term structure of futures returns

Usage

```
TSfit_volatility(parameters, futures, futures_TTM, dt)
```

Arguments

`parameters`	`vector`. A named vector of parameters of an N-factor model. Function `NFCP_parameters` is recommended.
`futures`	`matrix`. Historical observes futures price data. Each column must correspond to a listed futures contract and each row must correspond to a discrete observation of futures contracts. NA's are permitted.
`futures_TTM`	`vector`. Each element of 'futures_TTM' must correspond to the time-to-maturity from the current observation point of futures contracts listed in object 'futures'.
`dt`	`numeric`. Constant, discrete time step of observations, in years.

Details

The fit of an N-factor models theoretical volatility term structure of futures returns to those obtained directly from observed futures prices can be used as a measure of robustness for the models ability to explain the behaviour of a commodities term structure.

The theoretical model volatility term structure of futures returns is given by the following equation:

$$\sigma_F(\tau) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sigma_i\sigma_j\rho_{i,j}e^{-(\kappa_i+\kappa_j)\tau}$$

Under the case that $\kappa_1 = 0$, the model volatility term structure converges to σ_1^2 as τ grows large.

The empirical volatility term structure of futures returns is given by:

$$\hat{\sigma}_F^2(\tau) = \frac{1}{\Delta t}\sum_{i=1}^{N}(log(F(t_i,\tau)/F(t_i-\Delta t,\tau)) - \bar{\mu})^2$$

According to Cortazar and Naranjo (2006): "A larger number of factors gives more flexibility to adjust first and second moments simultaneously, hence explaining why (a) four-factor (may) outperform (a) three-factor one in fitting the volatility term structure."

Value

`TSfit_volatility` returns a matrix with the theoretical and empirical volatility term structure of futures returns, with the number of columns of this matrix coinciding with the number of input futures contracts.

References

Schwartz, E. S., and J. E. Smith, (2000). Short-Term Variations and Long-Term Dynamics in Commodity Prices. *Manage. Sci.*, 46, 893-911.

Cortazar, G., and L. Naranjo, (2006). An N-factor Gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3), 243-268.

Examples

```
# Test the volatility term structure fit of the Schwartz-Smith two-factor model on crude oil:
V_TSFit <- TSfit_volatility(
parameters = SS_oil$two_factor,
futures = SS_oil$stitched_futures,
futures_TTM = SS_oil$stitched_TTM,
dt = SS_oil$dt)
```

Index