Package 'schwartz97'

July 2, 2014

Type Package

Version 0.0.6

Date 2011-12-18

Title A package on the Schwartz two-factor commodity model

Author Philipp Erb, David Luethi, Juri Hinz, Simon Otziger

Maintainer Marc Weibel <marc.weibel@zhaw.ch>

LazyData no

Depends R(>= 2.10), FKF(>= 0.1.0), mvtnorm, methods, RUnit

Description This package provides detailed functionality for working with the Schwartz 1997 two-factor commodity model. Essentially, it contains pricing formulas for futures and European options and the standard d/p/q/r functions for the distribution of the state variables and futures prices. In addition, a parameter estimation procedure is contained together with many utilities as filtering and plotting functionality. This package is accompanied by futures data of ten commodities.

License GPL (>= 2)

Encoding latin1

Repository CRAN

Date/Publication 2014-02-11 12:32:43

NeedsCompilation no

R topics documented:

`schwartz97-package` *Two-factor Commodity Model*

Description

This package contains an implementation of the Schwartz two-factor commodity model, that is, the joint dynamics of the spot price and the spot convenience yield according to Schwartz (1997). The parameter estimation function constitutes the core of this package. Once the parameters are estimated, futures and European call and put options can be priced, term structures can be calculated and the usual distribution operations d/p/q/r can be carried out on the state variables as well as on futures prices. The package is accompanied by a variety of utility functions, futures data of ten commodities, and two vignettes describing technical details and usage of the package.

Details

Package:	schwartz97
Type:	Package
Version:	0.0.4
Date:	2011-12-18
License:	GPL (GNU Public License), Version 2 or later

Initialization:

`schwartz2f`	Initialize a Schwartz two-factor object.

Density, distribution function, quantile function, random number generation, and trajectories of the state variables:

`dstate`	Density of the spot and the convenience yield.

pstate	Distribution of the spot and the convenience yield.
qstate	Quantile of the spot and the convenience yield.
rstate	Random number generation of the spot and the convenience yield.
simstate	Trajectory of the spot and the convenience yield.

Density, distribution function, quantile function, and random number generation of the futures price:

dfutures	Density of the futures price.
pfutures	Distribution of the futures price.
qfutures	Quantile of the futures price.
rfutures	Random number generation of the futures price.

Parameter estimation:

fit.schwartz2f	Estimate parameters of the two-factor model.
fitted	Extract the model's fitted values.
resid	Extract model residuals.

Pricing:

pricefutures	Compute arbitrage-free futures prices.
priceoption	Compute arbitrage-free European option prices.

Utilities:

coef	Extract model coefficients of schwartz2f-objects.
mean	Extract the mean of schwartz2f-objects.
vcov	Extract the covariance matrix of schwartz2f-objects.
filter.schwartz2f	Filter futures prices to get the spot price and convenience yield.
plot	Plot schwartz2f.fit-objects.
plot	Plot trajectories of schwartz2f-objects.
futures	Use data(futures) to get data of 10 commodities.

Package vignette:

The R package schwartz97 contains two vignettes:

The vignette *Technical Document* gives the necessary relations and tools to fully understand the internals of the package.

The vignette *User Guide* discusses implementation details and gives numerous examples and intuitive explanations.

Author(s)

David Luethi, Philipp Erb, Juri Hinz, Simon Otziger

Maintainer: David Luethi <luethid@gmail.com>

References

Stochastic Convenience Yield and the Pricing of Oil Contingent Claims by Rajna Gibson and Eduardo S. Schwartz
The Journal of Finance 45, 1990, 959-976

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

Pricing of Options on Commodity Futures with Stochastic Term Structures of Convenience Yields and Interest Rates by Kristian R. Miltersen and Eduardo S. Schwartz
Journal of Financial and Quantitative Analysis 33, 1998, 33-59

Valuation of Commodity Futures and Options under Stochastic Convenience Yields, Interest Rates, and Jump Diffusions in the Spot by Jimmy E. Hilliard and Jorge Reis
Journal of Financial and Quantitative Analysis 33, 1998, 61-86

`coef-method` *Extract parameters of schwartz2f objects*

Description

The function `coef` returns parameters of `schwartz2f` and a `schwartz2f.fit` objects as a list. The function `coefficients` is an alias for `coef`.

Usage

```
## S4 method for signature 'schwartz2f'
coef(object)
## S4 method for signature 'schwartz2f'
coefficients(object)

## S4 method for signature 'schwartz2f.fit'
coef(object)
## S4 method for signature 'schwartz2f.fit'
coefficients(object)
```

Arguments

object	An object from class schwartz2f or schwartz2f.fit.

Value

If object is of class schwartz2f:

s0	Commodity spot price.
delta0	Convenience yield.
mu	Drift parameter of the spot price process.
sigmaS	Diffusion parameter of the spot price process.
kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.
rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.

If object is of class schwartz2f.fit:

s0	Commodity spot price.
delta0	Convenience yield.
mu	Drift parameter of the spot price process.
sigmaS	Diffusion parameter of the spot price process.
kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.
rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.
r	Instantaneous risk-free interest rate.
lambda	Market price of convenience yield risk.
alphaT	Mean-level of the convenience yield process with respect to the equivalent martingale measure.

The model and its parameters are described in the **Details** section of the schwartz2f-class documentation and in the package vignette *Technical Document*.

Author(s)

Philipp Erb, David Luethi

See Also

schwartz2f classes.

Examples

```
# ## coef-method for schwartz2f-objects:
# coef(schwartz2f())
#
```

```
# ## coef-method for schwartz2f.fit-objects:
# ## Estimate parameters for soybean oil (but stop after 3 iterations).
# data(futures)
# fit.obj <- fit.schwartz2f(futures$soybean.oil$price, futures$soybean.oil$ttm / 260,
#                           deltat = 1 / 260, control = list(maxit = 3))
# coef(fit.obj)
```

`distribution-futures` *Schwartz two-factor Model: Distribution of Futures Prices*

Description

Density, distribution function, quantile function and random number generation of futures prices.

Usage

```
## S4 method for signature 'ANY,ANY,ANY,numeric'
dfutures(x, time = 0.1, ttm = 1, s0 = 50, delta0 = 0,
         mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
         sigmaE = 0.5, rho = 0.75, r = 0.05, lambda = 0,
         alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f'
dfutures(x, time = 0.1, ttm = 1, s0, r = 0.05,
         lambda = 0, alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f.fit'
dfutures(x, time = 0.1, ttm = 1, s0, measure = c("P", "Q"), ...)


## S4 method for signature 'ANY,ANY,ANY,numeric'
pfutures(q, time = 0.1, ttm = 1, s0 = 50, delta0 = 0,
         mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
         sigmaE = 0.5, rho = 0.75, r = 0.05, lambda = 0,
         alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f'
pfutures(q, time = 0.1, ttm = 1, s0, r = 0.05,
         lambda = 0, alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f.fit'
pfutures(q, time = 0.1, ttm = 1, s0, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,numeric'
qfutures(p, time = 0.1, ttm = 1, s0 = 50, delta0 = 0,
         mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
```

```
         sigmaE = 0.5, rho = 0.75, r = 0.05, lambda = 0,
         alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f'
qfutures(p, time = 0.1, ttm = 1, s0, r = 0.05,
         lambda = 0, alphaT = NULL, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f.fit'
qfutures(p, time = 0.1, ttm = 1, s0, measure = c("P", "Q"), ...)

## S4 method for signature 'ANY,ANY,ANY,numeric'
rfutures(n, time = 0.1, ttm = 1, s0 = 50, delta0 = 0,
         mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0, sigmaE = 0.5,
         rho = 0.75, r = 0.05, lambda = 0, alphaT = NULL, measure = c("P", "Q"))

## S4 method for signature 'ANY,ANY,ANY,schwartz2f'
rfutures(n, time = 0.1, ttm = 1, s0, r = 0.05,
         lambda = 0, alphaT = NULL, measure = c("P", "Q"))

## S4 method for signature 'ANY,ANY,ANY,schwartz2f.fit'
rfutures(n, time = 0.1, ttm = 1, s0, measure = c("P", "Q"))
```

Arguments

q, x	vector of quantiles.
p	vector of probabilities.
n	number of observations. If `length(n) > 1`, the length is taken to be the number required.
time	Time where the futures process is evaluated (relative to now).
ttm	Time to maturity (relative to now).
s0	Either a `numeric` representing the initial value of the commodity spot price or an object inheriting from class `schwartz2f`.
delta0	Initial value of the convenience yield.
mu	Drift term of commodity spot price.
sigmaS	Diffusion parameter of the spot price process.
kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.
rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.
lambda	Market price of convenience yield risk (see **Details**).
alphaT	Mean-level of the convenience yield process with respect to the equivalent martingale measure (see **Details**).

r	Instantaneous risk-free interest rate.
measure	under which the functions are computed. "P" denotes the objective measure, "Q" the risk-neutral measure (see **Details**).
...	Arguments to be passed to the functions d/p/q-norm.

Details

Futures prices depend on the spot-price and the convenience yield.

To get the real (i.e. the objective) distribution of futures prices at some date in the future the dynamics is considered under the objective measure *P*. The *P*-dynamics is

$$dS_t = (\mu - \delta_t)S_t dt + \sigma_S \tilde{} S_t \tilde{} dW_t^1,$$

$$d\delta_t = \kappa(\alpha - \delta_t)dt + \sigma_E dW_t^2$$

$$dW_t^1 \tilde{} dW_t^2 = \rho dt,$$

where W^1, W^2 are Brownian motions under the objective measure, the measure *P*.

Options on futures are evaluated based on the risk-neutral dynamics of the spot-price and the convenience yield, i.e. under the measure *Q*. The *Q*-dynamics is

$$dS_t = (r - \delta_t)S_t dt + \sigma_S \tilde{} S_t \tilde{} d\tilde{W}_t^1$$

$$d\delta_t = \kappa(\tilde{\alpha} - \delta_t)dt + \sigma_E d\tilde{W}_t^2,$$

where $\tilde{W}^1$, $\tilde{W}^2$ are Brownian motions with respect to *Q*.

$\tilde{\alpha} = \alpha - \lambda/\kappa$ where λ is the market price of convenience-yield risk. The market price of convenience yield risk can either be specified explicitly by `lambda` or implicitly by `alphaT`. The relation is `alphaT = alpha - lambda / kappa`. See the package vignette.

Value

Probabilities, densities, quantiles or samples of the log-normally distributed futures prices as `numeric`.

Note

Note that futures and forward prices coincide as the interest rate is assumed to be constant in the Schwartz two-factor model.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

Valuation of Commodity Futures and Options under Stochastic Convenience Yields, Interest Rates, and Jump Diffusions in the Spot by Jimmy E. Hilliard and Jorge Reis
Journal of Financial and Quantitative Analysis 33, 1998, 61-86

See Also

pricefutures, d/p/qstate, r/simstate

Examples

```
# ## Create a "schwartz2f"-object
# model <- schwartz2f()
#
# ## Probability
# pfutures(q = 10 * 3:9, time = 0.5, ttm = 2, model, lambda = 0.01)
#
# ## Density
# dfutures(x = c(20, 40, 100), time = 0.5, ttm = 2, model, lambda = 0.01)
#
# ## Quantile
# qfutures(p = 0.1 * 2:5, time = 0.5, ttm = 10, model, lambda = 0.01)
#
# ## Sample
# sim <- rfutures(n = 1000, time = 0.5, ttm = 5, model, lambda = 0.01)
#
# hist(sim, prob = TRUE)
# lines(seq(30, 300, length = 100),
#       dfutures(seq(30, 300, length = 100),
#                time = 0.5, ttm = 5, model, lambda = 0.01), col = "red")
#
# ## At time 0 the futures price is a deterministic function of s0 and
# ## delta0. Therefore 3 times the same value is obtained:
# rfutures(3, time = 0, ttm = 1, model, lambda = 0)
```

---

distribution-state *Schwartz two-factor Model: Distribution of the State Variables*

Description

Density, distribution function and quantile function of the state variables. The state variables are the commodity spot price `s` and the spot convenience yield `delta`. The commodity log spot price and the convenience yield follow a bivariate normal distribution.

Usage

```
## S4 method for signature 'ANY,ANY,numeric'
dstate(x, time = 1, s0 = 50, delta0 = 0,
       mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
       sigmaE = 0.5, rho = 0.75, ...)

## S4 method for signature 'ANY,ANY,schwartz2f'
dstate(x, time = 1, s0, ...)


## S4 method for signature 'ANY,ANY,ANY,numeric'
pstate(lower, upper, time = 1, s0 = 50, delta0 = 0,
       mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
       sigmaE = 0.5, rho = 0.75, ...)

## S4 method for signature 'ANY,ANY,ANY,schwartz2f'
pstate(lower, upper, time = 1, s0, ...)

## S4 method for signature 'ANY,ANY,numeric'
qstate(p, time = 1, s0 = 50, delta0 = 0,
       mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
       sigmaE = 0.5, rho = 0.75, tail = "lower.tail", ...)

## S4 method for signature 'ANY,ANY,schwartz2f'
qstate(p, time = 1, s0, tail = "lower.tail", ...)
```

Arguments

x	Vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
time	Time at which the quantity is computed (relative to time zero).
p	Probability, a scalar.
lower	The vector of lower limits of length 2. Note that first component stands for lower limit of the commodity spot price rather than the log-price.
upper	The vector of upper limits of length 2. Note that first component stands for the upper limit of the commodity spot price rather than the log-price.
s0	Either a numeric representing the initial value of the commodity spot price or an object inheriting from class schwartz2f.
delta0	Initial value of the convenience yield.
mu	enters the drift of the commodity spot price.
sigmaS	Diffusion parameter of the spot price-process.
kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.

rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.
tail	See qmvnorm of package mvtnorm.
...	Further arguments to be passed to methods of package mvtnorm.

Details

The model and its parameters are described in the **Details** section of the schwartz2f-class documentation and in the package vignette *Technical Document*.

The above methods rely on the functions pmvnorm, dmvnorm, and qmvnorm of the package mvtnorm.

Value

dstate and pstate return a numeric, qstate returns the output of qmvnorm as a list.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

See Also

schwartz2f-class description, rstate and simstate for random number generation, constructors schwartz2f and fit.schwartz2f.

Examples

```
# ## Create a "schwartz2f"-object
# model <- schwartz2f()
#
# ## Probability
# pstate(lower = c(0, -Inf), upper = c(45, 0.01), time = 1, model)
#
# ## Density
# dstate(x = c(50, 0.03), time = 2, model)
# dstate(x = rbind(c(50, 0.03), c(50, 0.1)), time = 2, model) # x is a matrix
#
# ## Quantile
# qstate(p = 0.5, s0 = model)
#
# ## Generate random numbers
# object <- schwartz2f(alpha = 0.05)
# samples <- rstate(1000, time = 2, object)
# ## ...and plot histograms
# par(mfrow = c(2, 1))
# hist(samples[,1])
# abline(v = mean(object, time = 2)[1], col = "red")
# hist(samples[,2])
# abline(v = mean(object, time = 2)[2], col = "red")
```

filter-futures *Schwartz two-factor Model: Filter futures data*

Description

Filter a series of futures prices to obtain state variables.

Usage

```
## S4 method for signature 'ANY,ANY,numeric'
filter.schwartz2f(data, ttm, s0 = 50, delta0 = 0,
                  mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0, sigmaE = 0.5,
                  rho = 0.75, r = 0.05, lambda = 0, alphaT = NULL,
                  deltat = 1/260, meas.sd = rep(1e-3, ncol(data)),
                  P0 = 0.5 * diag(c(log(s0), abs(delta0))))

## S4 method for signature 'ANY,ANY,schwartz2f'
filter.schwartz2f(data, ttm, s0,
                  r = 0.05, lambda = 0, alphaT = NULL, deltat = 1/260,
                  meas.sd = rep(1e-3, ncol(data)),
                  P0 = 0.1 * diag(2))

## S4 method for signature 'ANY,ANY,schwartz2f.fit'
filter.schwartz2f(data, ttm, s0)
```

Arguments

data	A matrix with futures prices.
ttm	A matrix with the corresponding time to maturity (see **Details**).
s0	Either a numeric representing the initial value of the commodity spot price or an object inheriting from class schwartz2f.
delta0	Initial value of the convenience yield.
mu	enters the drift of the commodity spot price.
sigmaS	Diffusion parameter of the spot price process.
kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.
rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.
r	Instantaneous risk-free interest rate.
lambda	Market price of convenience yield risk.

alphaT	Mean-level of the convenience yield process with respect to the equivalent martingale measure.
deltat	Time increment.
meas.sd	The standard deviation of the measurement equation (see **Details** section of fit.schwartz2f).
P0	Variance of the state variables s0 and P0.

Details

The elements of data and ttm have the following interpretation: data[i,j] denotes the futures price whose time to maturity was ttm[i,j] when it was observed. The unit of ttm is defined by deltat.

deltat is the time between observation data[i,j] and data[i + 1,j]. It is assumed to be constant, i.e., that data is a regular time-series.

Value

A list with components:

state	A matrix with colnames are "S" and "delta" giving the the expected spot price and the convenience yield.
fkf.obj	The filter output from the package fkf. Note that the log of the commodity spot price is filtered.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

See Also

fit.schwartz2f, pricefutures.

Examples

```
# data(futures)
#
# ## Estimate parameters for coffee data (stop after 20 iterations)
# fit.obj <- fit.schwartz2f(futures$coffee$price, futures$coffee$ttm / 260,
#                           deltat = 1 / 260, control = list(maxit = 20))
#
# ## Filter the futures data to get the spot price and the convenience yield.
# filtered <- filter.schwartz2f(futures$coffee$price, futures$coffee$ttm / 260, fit.obj)
#
# ## ...and plot it.
```

```
# par(mfrow = c(2, 1))
# plot(filtered$state[,1], ylab = "Spot price", type = "l")
# lines(futures$coffee$price[,1], col = "red") # Closest to maturity futures
# plot(filtered$state[,2], ylab = "Convenience yield", type = "l")
# abline(h = 0)
```

`fitted-futures` *Extract Model Fitted Values*

Description

Function to extract fitted values from `schwartz2f.fit`-objects.

Usage

```
## S4 method for signature 'schwartz2f.fit'
fitted(object, data, ttm)
```

Arguments

`object`	A `schwartz2f.fit`-object returned from `schwartz2f.fit`.
`data`	A matrix with futures prices.
`ttm`	A matrix with the corresponding times to maturity (see **Details**).

Details

The elements of `data` and `ttm` have the following interpretation: `data[i,j]` denotes the futures price whose time to maturity was `ttm[i,j]` when it was observed. The unit of `ttm` was defined by the argument `deltat` of `fit.schwartz2f`.

Value

A matrix containing the fitted values of the same dimension as `data`.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

See Also

`fit.schwartz2f`, `schwartz2f.fit`-class, `resid`.

Examples

```
# data(futures)
#
# ## Estimate parameters for lumber data. Fit only 'mu', 'sigmaS',
# ## 'sigmaE', and 'rho' (and stop after 100 iterations).
# fit.obj <- fit.schwartz2f(futures$lumber$price, futures$lumber$ttm / 260,
#                           opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = TRUE,
#                                        sigmaS = TRUE, kappa = FALSE, alpha = FALSE,
#                                        sigmaE = TRUE, rho = TRUE, lambda = FALSE),
#                           alpha = 0, kappa = 1.2, lambda = 0,
#                           deltat = 1 / 260, control = list(maxit = 100))
# fit.obj
#
# ## Get the fitted values
# fitted.futures <- fitted(fit.obj, futures$lumber$price, futures$lumber$ttm / 260)
#
# par(mfrow = c(1, 3))
# ## Plot futures prices
# plot(as.ts(futures$lumber$price), plot.type = "single", ylab = "Futures prices",
#      col = gray(seq(0.1, 0.9, length = 4)))
# ## Plot fitted values
# plot(as.ts(fitted.futures), plot.type = "single", ylab = "Fitted values",
#      col = gray(seq(0.1, 0.9, length = 4)))
# ## Plot relative difference
# plot(as.ts((fitted.futures - futures$lumber$price) / fitted.futures), plot.type = "single",
#      ylab = "Relative difference", col = gray(seq(0.1, 0.9, length = 4)))
```

`futures-data` *Daily futures prices*

Description

Futures prices, time to maturity, open interest, volume, underlying tickers, and last trade date of ten different commoditites: corn, wheat, soybean, soybean meal, soybean oil, lumber, live cattle, coffee, heating oil, copper.

There are, depending on the liquidity of the commodity, between 4 and 10 'clean' closest to maturity futures price series.

Usage

```
data(futures)
```

Format

A list containing ten commodities as lists: "corn", "wheat", "soybean", "soybean.meal", "soybean.oil", "lumber", "live.cattle", "coffee", "heating.oil", "copper".

Each list contains six dimnamed matrices:

`price` Daily futures prices.

`ttm` The time to maturity of the futures contracts in units of days (see **Details**.)

`oi` Open interest.

`vol` Volume.

`underl.tickers` Underlying tickers / contracts.

`last.trade.dt` Last trade date as character in the ISO 8601 international standard format.

The i-th column of each matrix contains data for the i-th closest to maturity contract. The i-th column name is the ticker of the i-th generic futures.

Commodity	# Contracts	Exchange	Start date	End date
Corn	6	CBOT	1997-01-02	2010-04-07
Wheat	5	CBOT	1995-01-03	2010-04-07
Soybean	7	CBOT	1995-01-03	2010-04-07
Soybean meal	6	CBOT	2000-01-03	2010-04-07
Soybean oil	6	CBOT	1995-01-03	2010-04-07
Lumber	4	CME	1995-01-03	2010-04-07
Live cattle	6	CME	2004-07-01	2010-04-07
Coffee	5	ICE	1995-01-03	2010-04-07
Heating oil	10	NYMEX	1995-01-03	2010-03-31
Copper	8	COMEX	1996-01-02	2010-02-24

Details

The elements of `price` and `ttm` have the following interpretation: `price[i,j]` denotes the futures price whose time to maturity was `ttm[i,j]` days when it was observed.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

See Also

`futuresplot`, `fit.schwartz2f`.

Examples

```
# data(futures)
#
# ## Plot forward curves of lumber
# futuresplot(futures$lumber, type = "forward.curve")
#
# ## Plot time to maturity of heating oil data
# futuresplot(futures$heating.oil, type = "ttm")
#
```

```
# ## Make 'futures' weekly, take Wednesday data
# futures.w <- rapply(futures, function(x)x[format(as.Date(rownames(x)), "%w") == 3,],
#                     classes = "matrix", how = "list")
#
# ## Make 'futures' monthly, take the 28th day of the month
# futures.m <- rapply(futures, function(x)x[format(as.Date(rownames(x)), "%d") == 28,],
#                     classes = "matrix", how = "list")
#
# ## Plot weekly lumber and monthly soybean data
# futuresplot(futures.w$lumber, type = "forward.curve", main = "Lumber")
# futuresplot(futures.m$soybean, type = "forward.curve", main = "Soybean")
#
# ## Convert to zoo-objects:
# require(zoo)
# futures.zoo <- rapply(futures, function(x)zoo(x, as.Date(rownames(x))),
#                       classes = "matrix", how = "list")
#
# ## ...and plot it nicely using plot.zoo:
# plot(futures.zoo$heating.oil$ttm)
# plot(futures.zoo$wheat$vol)
# plot(futures.zoo$copper$oi)
#
# ## Estimate soybean meal parameters (stop after 100 iterations).
# ## ttm (time-to-maturity) is divided by 260 as it is in unit of days and
# ## deltat = 1/52 because weekly price observations are used.
# soybean.meal.fit <- fit.schwartz2f(data = futures.w$soybean.meal$price,
#                                    ttm = futures.w$soybean.meal$ttm / 260,
#                                    deltat = 1 / 52, r = 0.04,
#                                    control = list(maxit = 100))
# soybean.meal.fit
```

`futures-plot` *Visualization of Futures Data*

Description

Visualization of historical commodity futures prices and remaining time to maturity. This function is intended to be fed with the futures data contained in this package (see `futures-data`).

Usage

```
futuresplot(futures, type = c("forward.curve", "ttm"), ...)
```

Arguments

`futures`	A `list` with elements `price` and `ttm`. Usually an element of `futures`.
`type`	What shall be plotted. `"forward.curve"` or `"ttm"` (time to maturity).
`...`	Optional arguments passed to `plot`.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

See Also

futures-data

Examples

```
# data(futures)
#
# ## Plot time to maturity of corn data
# futuresplot(futures$corn, type = "ttm")
#
# ## Plot forward curves of wheat data since Jan 2010
# wheat.2010 <- lapply(futures$wheat,
#                      function(x)x[as.Date(rownames(x)) > "2010-01-01",])
# futuresplot(wheat.2010, type = "forward.curve")
```

---

mean-vcov-methods *Expected value and variance-covariance*

Description

The function mean returns the expected value of the spot price and the convenience yield for some time in the future. The function vcov returns the covariance matrix of the *log* spot price and the convenience yield.

Usage

```
## S4 method for signature 'schwartz2f'
mean(x, time = 1)

## S4 method for signature 'schwartz2f'
vcov(object, time = 1)
```

Arguments

x, object	An object inheriting from class schwartz2f.
time	The point in time for which the mean or covariance matrix is computed. mean accepts a vector, vcov a scalar only.

Value

Either the expected value or variance-covariance matrix.

Note

vcov returns the variance-covariance matrix of [*log* spot price, convenience yield](time).

Author(s)

David Luethi

See Also

schwartz2f to create Schwartz 2 factor objects, plot to show trajectories of the state variables including means and standard deviations.

Examples

```
# mean(schwartz2f(mu = 0.1), time = 1)
#
# mean(schwartz2f(mu = 0.2), time = 0:3)
#
# vcov(schwartz2f(), time = 10)
#
# ## Plot a schwartz2f-object including means and standard deviations
# plot(schwartz2f(sigmaE = 0.1), n = 50, time = 5, dt = 1 / 52)
```

---

parameter-estimation *Schwartz 1997 two factor parameter estimation*

Description

Fit the Schwartz 1997 two factor commodity model to futures data.

Usage

```
fit.schwartz2f(data, ttm, deltat = 1 / 260,
               s0 = data[1,1], delta0 = 0,
               mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0, sigmaE = 0.3,
               rho = 0.7, lambda = 0,
               meas.sd = rep(0.1, ncol(data)),
              opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = TRUE, sigmaS = TRUE,
                           kappa = TRUE, alpha = TRUE, sigmaE = TRUE,
                           rho = TRUE, lambda = FALSE),
               opt.meas.sd = c("scalar", "all", "none"),
               r = 0.03, silent = FALSE, ...)
```

Arguments

data	A matrix with futures prices. NA-values are allowed.
ttm	A matrix with the corresponding time to maturity (see **Details**).
deltat	Time increment (see **Details**).
s0	Initial value of the commodity spot price.
delta0	Initial value of the convenience yield.
mu	Initial value of the drift parameter of the commodity spot price.
kappa	Initial parameter of the speed of mean-reversion of the convenience yield process.
alpha	Initial parameter of the mean-level of the convenience yield process.
sigmaS	Initial parameter of the diffusion parameter of the spot price process.
sigmaE	Initial parameter of the diffusion parameter of the convenience yield process.
rho	Initial parameter of the correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.
meas.sd	Initial parameter of the standard deviation of the measurement equation (see **Details**).
lambda	Initial value of the market price of convenience yield risk.
opt.pars	A logical vector giving the parameters which shall be fitted. The order is as given in the function header, names are discarded.
opt.meas.sd	States how the standard deviation of the measurement equation should be treated (see **Details**).
r	Instantaneous risk-free interest rate.
silent	If FALSE the log-likelihood and parameters will be printed in each iteration.
...	Arguments passed to optim.

Details

The elements of data and ttm have the following interpretation: data[i,j] denotes the futures price whose time to maturity was ttm[i,j] when it was observed (in units of deltat).

The time increment between observation data[i,j] and data[i + 1,j] is denoted with deltat. Note that this specification requires a regularly spaced data series.

opt.meas.sd specifies how measurement uncertainty is treated in the fit: According to the model there should be a one-to-one correspondence between the spot and the futures price. In reality, the term structure does not fully match for any set of parameters. This is reflected in the measurement uncertainty-vector meas.sd. All components of meas.sd can be fitted. However, it might be sufficient to fit only a scalar where the measurement uncertainty is parametrized by scalar * meas.sd. In this case define the vector meas.sd and set opt.meas.sd to "scalar". meas.sd can be set to a vector with each component set to, e.g., 2%, giving each point in the term structure equal weight. Another reasonable specification takes open interest or volumes into account: The higher the volume, the higher the weight and therefore the lower the corresponding component of meas.sd. If all components of meas.sd shall be fitted choose "all". If the measurement uncertainty is known set meas.sd to "none". Note that the measurement errors are assumed to be independent in this implementation (even though the model and the filter do not require independence).

The model and its parameters are described in the **Details** section of the schwartz2f-class documentation and in the package vignette *Technical Document*.

Value

An object of class schwartz2f.fit.

Note

Parameter estimation is statistically fragile and computationally demanding. Multiple local maxima of the likelihood may exist which can result in absurd parameter estimates as, e.g., a yearly drift of 300% and or a market price of convenience yield risk of -200%. Therefore, a reasonable parameter estimation is most likely an iteration where several initial values are used and different combinations of parameters are held constant during estimation. Also, simulation studies showed that a fairly large sample is required to get adequate estimates (e.g. 20000 daily observations, depending on the number of parameters which shall be estimated). For this reason the default is to hold `s0`, `delta0`, and `lambda` constant.

Several utility functions may help to investigate the fit (see e.g. fitted, resid, plot, coef).

The fitting procedure generally requires a large number of iterations to achieve a reasonable tolerance level. Each optimization iteration involves the filtering of the data set by the Kalman filter. Therefore, an efficient implementation of the Kalman filter is key. Hence, the fkf function of the package FKF can be considered as the backbone of the estimation procedure.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

See Also

schwartz2f.fit class, fitted, resid, plot, coef), pricefutures, futures-data.

Examples

```
# data(futures)
#
# ## Estimate parameters for wheat data.
# ## (little precision required with reltol = 1e-3)
# fit.obj <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
#                           deltat = 1 / 260, control = list(reltol = 1e-3))
#
# ## See how parameter values evolved during the fit
# plot(fit.obj, type = "trace.pars")
#
```

```
# ## Do the same but with lower tolerance level:
# high.precision.fit <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
#                                      control = list(maxit = 3000, reltol = 5e-8))
#
# high.precision.fit
#
# plot(high.precision.fit, type = "trace.pars") # ...concerning parameter evolution.
#
# ## Alpha becomes nonsensically high, kappa (speed of mean-reversion
# ## of the convenience yield) goes to zero. Solution: Choose different
# ## initial values and/or hold kappa constant at 1.
#
# constrained.fit <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
#                                   opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = TRUE,
#                                            sigmaS = TRUE, kappa = FALSE, alpha = TRUE,
#                                             sigmaE = TRUE, rho = TRUE, lambda = TRUE),
#                                   alpha = 0, kappa = 1,
#                                   control = list(maxit = 3000, reltol = 5e-8))
#
# constrained.fit
#
# plot(constrained.fit, type = "trace.pars")
#
# ## The above parameters based on a fit - where kappa was held constant at 1 -
# ## look more reasonable.
#
# ## These residuals should be iid standard normal
# model.resid <- resid(fit.obj, data = futures$wheat$price, ttm = futures$wheat$ttm / 260,
#                      type = "filter.std")
# acf(model.resid, na.action = na.pass)
# par(mfrow = c(3, 2))
# apply(model.resid, 2, function(x)plot(density(na.omit(x))))
# ## ...but are anything but iid standard normal.
#
# ## ...though the fitted values look fine:
# fitted.futures <- fitted(fit.obj, futures$wheat$price, futures$wheat$ttm / 260)
# par(mfrow = c(1, 3))
# ### Plot futures prices
# plot(as.ts(futures$wheat$price), plot.type = "single", ylab = "Futures prices",
#      col = gray(seq(0.1, 0.9, length = 4)))
# ## Plot fitted values
# plot(as.ts(fitted.futures), plot.type = "single", ylab = "Fitted values",
#      col = gray(seq(0.1, 0.9, length = 4)))
# ## Plot relative difference
# plot(as.ts((fitted.futures - futures$wheat$price) / fitted.futures), plot.type = "single",
#      ylab = "Relative difference", col = gray(seq(0.1, 0.9, length = 4)))
#
#
# ## Try with kappa = 1, alpha = 0, and flexible standard deviationss of
# ## the measurement errors. Stop at 200 iterations.
# fit.obj.2 <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
#                             opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = TRUE,
#                                          sigmaS = TRUE, kappa = FALSE, alpha = FALSE,
```

```
#                                         sigmaE = TRUE, rho = TRUE, lambda = TRUE),
#                             alpha = 0, kappa = 1, opt.meas.sd = "all",
#                             deltat = 1 / 260, control = list(maxit = 200))
#
#
# model.resid.2 <- resid(fit.obj.2, data = futures$wheat$price, ttm = futures$wheat$ttm / 260,
#                        type = "filter.std")
# ## Residuals seem to be better:
# acf(model.resid.2, na.action = na.pass)
# par(mfrow = c(3, 2))
# apply(model.resid.2, 2, function(x)plot(density(na.omit(x))))
#
#
# ## The schwartz2f.fit-object 'fit.obj' can be used to do further calculations as
# ## pricing a put option on the wheat futures which matures in 1.1
# ## years. The option expires in 1 year. The strike price is the
# ## expected futures price in 1.1 years.
# priceoption("put", time = 1, Time = 1.1, K = pricefutures(1.1, fit.obj),
#             fit.obj)
#
#
#
# ## Parameter estimation for weekly soybean meal data:
# ## Get Wednesday observations:
# futures.w <- rapply(futures, function(x)x[format(as.Date(rownames(x)), "%w") == 3,],
#                     classes = "matrix", how = "list")
#
# ## Estimate soybean meal parameters (stop after 500 iterations).
# ## ttm (time-to-maturity) is divided by 260 as it is in unit of days and
# ## deltat = 1/52 because weekly price observations are used.
# ## Estimate all measurement error standard deviations (opt.meas.sd == "all"),
# ## but hold kappa, alpha, and lambda constant.
# soybean.meal.fit <- fit.schwartz2f(data = futures.w$soybean.meal$price,
#                                    ttm = futures.w$soybean.meal$ttm / 260,
#                                    opt.meas.sd = "all",
#                                    mu = 0, kappa = 1, alpha = 0.03, r = 0.04,
#                                    opt.pars = c(s0 = FALSE, delta0 = FALSE, mu = TRUE,
#                                            sigmaS = TRUE, kappa = FALSE, alpha = FALSE,
#                                             sigmaE = TRUE, rho = TRUE, lambda = FALSE),
#                                    deltat = 1 / 52, control = list(maxit = 500))
#
# soybean.meal.fit
#
# plot(soybean.meal.fit, type = "trace.pars") # plot the parameter evolution
#
# ## Plot real and predicted forward curves:
# par(mfrow = c(1, 2))
# futuresplot(futures.w$soybean.meal, type = "forward.curve")
# plot(soybean.meal.fit, type = "forward.curve", data = futures.w$soybean.meal$price,
#      ttm = futures.w$soybean.meal$ttm / 260)
```

plot.fit-method *Plot Schwartz two-factor fit-objects*

Description

This function plots the parameter evolution during the fit, the filtered state variables (i.e. the spot price and the convenience yield), forward curves, or trajectories of the state variables.

Usage

```
## S4 method for signature 'schwartz2f.fit,missing'
plot(x, type = c("trace.pars", "state", "forward.curve", "sim"),
     data, ttm, ...)
```

Arguments

x	A schwartz2f.fit object.
type	What shall be plotted (see **Details**).
data	A matrix containing futures prices to which parameters were fitted.
ttm	A matrix with the corresponding time to maturity (see **Details**).
...	Arguments passed to plot.

Details

If `type == "trace.pars"`, the parameter evolution of the estimation is plotted. The horizontal lines denote the final value.
If `type == "state"`, the filtered state variables are plotted and overlaid with the futures prices.
If `type == "forward.curve"`, fitted forward curves are plotted.
If `type == "sim"`, a bunch of simulated trajectories of the state variables are plotted.

The elements of `data` and `ttm` have the following interpretation: `data[i,j]` denotes the futures price whose time to maturity was `ttm[i,j]` when it was observed. The time unit was defined by the argument `deltat` of the function `fit.schwartz2f` (stored in `x@deltat`).

Author(s)

David Luethi

See Also

`fit.schwartz2f` for parameter estimation, `plot`-method for `schwartz2f`-objects.

Examples

```
# data(futures)
#
# ## Estimate parameters for lumber data (stop after 100 iterations)
# fit.obj <- fit.schwartz2f(futures$lumber$price, futures$lumber$ttm / 260,
#                           deltat = 1 / 260,
#                           control = list(maxit = 100))
#
# ## Plot parameter evolution
# plot(fit.obj, type = "trace.pars")
#
# ## Plot the state variables
# plot(fit.obj, type = "state", data = futures$lumber$price,
#      ttm = futures$lumber$ttm / 260)
#
# ## Plot fitted and real forward curves of wheat data since Jan 2010.
# lumber.1995 <- lapply(futures$lumber, function(x)x[as.Date(rownames(x)) < "2000-01-01",])
# par(mfrow = c(1, 2))
# plot(fit.obj, type = "forward.curve", data = lumber.1995$price,
#      ttm = lumber.1995$ttm / 260)
# futuresplot(lumber.1995)
#
# ## Plot trajectories from the state variables
# plot(fit.obj, type = "sim")
```

`plot.state-method` *Plot Schwartz two-factor trajectories*

Description

This function plots trajectories of the Schwartz two-factor model including the means and confidence intervals at 99%, 95% and 90% levels.

Usage

```
## S4 method for signature 'schwartz2f,missing'
plot(x, n = 100, time = 2, dt = 1/52)
```

Arguments

`x`	A `schwartz2f`-object.
`n`	Number of trajectories.
`time`	Time span of the simulation.
`dt`	Time step.

Author(s)

David Luethi

See Also

schwartz2f constructor.

Examples

```
# object <- schwartz2f(s0 = 1, mu = 0.1, sigmaS = 0.2,
#                      delta0 = 0, kappa = 2, alpha = 0.05, sigmaE = 0.1,
#                      rho = 0.5)
#
# plot(object, n = 50, time = 2, dt = 1/52)
```

---

pricing-futures *Schwartz two-factor Model: Futures Prices*

Description

Compute arbitrage-free futures prices.

Usage

```
## S4 method for signature 'ANY,numeric'
pricefutures(ttm = 1, s0 = 50, delta0 = 0, sigmaS = 0.3,
             kappa = 1, alpha = 0, sigmaE = 0.5, rho = 0.75,
             r = 0.03, lambda = 0, alphaT = NULL)

## S4 method for signature 'ANY,schwartz2f'
pricefutures(ttm = 1, s0, r = 0.03,
             lambda = 0, alphaT = NULL)

## S4 method for signature 'ANY,schwartz2f.fit'
pricefutures(ttm = 1, s0)
```

Arguments

ttm	Time to maturity.
s0	Either a numeric representing the initial value of the commodity spot price or an object inheriting from class schwartz2f.
delta0	Initial value of the convenience yield.
sigmaS	Diffusion parameter of the spot price-process.

kappa	Speed of mean-reversion of the convenience-yield process.
alpha	Mean-level of the convenience-yield process.
sigmaE	Diffusion parameter of the convenience-yield process.
rho	Correlation coefficient between the Brownian motion driving the spot-price and the convenience-yield process.
r	Instantaneous risk-free interest rate.
lambda	Market price of convenience yield risk (see **Details**).
alphaT	Mean-level of the convenience yield process with respect to the equivalent martingale measure (see **Details**).

Details

The model and its parameters are described in the **Details** section of the schwartz2f-class documentation and in the package vignette *Technical Document*.

Value

A numeric containing futures prices.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

Valuation of Commodity Futures and Options under Stochastic Convenience Yields, Interest Rates, and Jump Diffusions in the Spot by Jimmy E. Hilliard and Jorge Reis
Journal of Financial and Quantitative Analysis 33, 1998, 61-86

See Also

priceoption to price options, d/p/q/rfutures to work with futures, schwartz2f-constructor, fit.schwartz2f for parameter estimation, futures-data.

Examples

```
# ## function call by atomic arguments
# forward.curve <- pricefutures(ttm = 0.2 * 1:10, s0 = 10, delta0 = 0,
#                               alpha = 0, lambda = 0.02, r = 0)
# plot(forward.curve, type = "b")
#
# ## function call via schwartz2f-object.
# obj <- schwartz2f(delta0 = 0, sigmaE = 1e-5) # Make convenience yield inactive
```

```
# forward.curve <- pricefutures(ttm = 0.2 * 1:10, s0 = obj, r = 0, alphaT = 0)
# plot(forward.curve, type = "b")
```

---

pricing-options *Schwartz two-factor Model: European Option Prices*

Description

Compute arbitrage-free prices of European call and put options on commodity futures contracts.

Usage

```
## S4 method for signature 'ANY,ANY,ANY,ANY,numeric'
priceoption(type = c("call", "put"), time = 0.5, Time = 1, K = 40,
            g0 = 50, sigmaS = 0.3, kappa = 1, sigmaE = 0.5,
            rho = 0.75, r = 0.03)

## S4 method for signature 'ANY,ANY,ANY,ANY,schwartz2f'
priceoption(type = c("call", "put"),
            time = 0.5, Time = 1, K = 40,
            g0, r = 0.03, lambda = 0, alphaT = NULL)

## S4 method for signature 'ANY,ANY,ANY,ANY,schwartz2f.fit'
priceoption(type = c("call", "put"),
            time = 0.5, Time = 1, K = 40, g0)
```

Arguments

type	Either a European "call" or a "put" option on a futures contract.
time	Exercise time of the option.
Time	Maturity date of the underlying futures (see **Details**).
K	Strike price.
g0	The current *futures price* or an object inheriting from class schwartz2f.
sigmaS	Diffusion parameter of the spot price-process.
kappa	Speed of mean-reversion of the convenience-yield process.
sigmaE	Diffusion parameter of the convenience-yield process.
rho	Correlation coefficient between the Brownian motion driving the spot-price and the convenience-yield process.
r	Instantaneous risk-free interest rate.
lambda	Market price of convenience yield risk (see **Details**).
alphaT	Mean-level of the convenience yield process with respect to the equivalent martingale measure (see **Details**).

Details

The price of an option on the spot commodity is obtained by setting `time == Time`. This is because of the convergence of the futures price towards the spot price at maturity. In general the option expires before the futures contract (`time < Time`).

If `g0` is either of class `schwartz2f` or class `schwartz2f.fit` the futures price `g0` is computed first and then plugged into the pricing function with signature `ANY,ANY,ANY,ANY,numeric`.

The model and its parameters are described in the **Details** section of the `schwartz2f`-class documentation and in the package vignette *Technical Document*.

Value

A `numeric` containing the option prices.

Note

Since the two-factor model assumes a constant interest rate, futures and forwards always have the same value and therefore also any derivative of futures or forwards.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

Pricing of Options on Commodity Futures with Stochastic Term Structures of Convenience Yields and Interest Rates by Kristian R. Miltersen and Eduardo S. Schwartz
Journal of Financial and Quantitative Analysis 33, 1998, 33-59

Valuation of Commodity Futures and Options under Stochastic Convenience Yields, Interest Rates, and Jump Diffusions in the Spot by Jimmy E. Hilliard and Jorge Reis
Journal of Financial and Quantitative Analysis 33, 1998, 61-86

See Also

`pricefutures` to price futures, `d/p/q/rfutures` to work with futures, `schwartz2f`-constructor, `fit.schwartz2f` for parameter estimation, `futures-data`.

Examples

```
# ## The option expires in 0.5 years and the futures contract in 1 year.
# priceoption(type = "call", time = 0.5, Time = 1, K = 40, g0 = 50)
```

```
#
# ## The price of a European put option on the spot which expires in 2.5
# ## years.
# priceoption(type = "put", time = 2.5, Time = 2.5, K = 900, lambda = 0.02,
#             g0 = schwartz2f(s0 = 1000))
```

---

rand-state *Schwartz two-factor Model: Sampling from the State Variables*

Description

Random number and trajectory generation of the state variables. The state variables are the commodity spot price `s0` and the spot convenience yield `delta0`. The commodity log spot price and the convenience yield follow a bivariate normal distribution.

Usage

```
## S4 method for signature 'ANY,ANY,numeric'
rstate(n, time = 1, s0 = 50, delta0 = 0,
       mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
       sigmaE = 0.5, rho = 0.75, method = "chol")

## S4 method for signature 'ANY,ANY,schwartz2f'
rstate(n, time = 1, s0, method = "chol")

## S4 method for signature 'ANY,ANY,numeric'
simstate(n, time = 1, s0 = 50, delta0 = 0,
         mu = 0.1, sigmaS = 0.3, kappa = 1, alpha = 0,
         sigmaE = 0.5, rho = 0.75, method = "chol")

## S4 method for signature 'ANY,ANY,schwartz2f'
simstate(n, time = 1, s0, method = "chol")
```

Arguments

`n`	Number of observations.
`time`	at which random numbers of the state variables are drawn (`rstate`) or horizon of the trajectory (`simstate`) relative to now.
`s0`	Either a `numeric` representing the initial value of the commodity spot price or an object inheriting from class `schwartz2f`.
`delta0`	Initial value of the convenience yield.
`mu`	enters the drift of the commodity spot price.
`sigmaS`	Diffusion parameter of the spot price-process.

kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.
rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.
method	See rmvnorm of package mvtnorm.

Details

The model and its parameters are described in the **Details** section of the schwartz2f-class documentation and in the package vignette *Technical Document*.

The above methods rely on the functions pmvnorm, dmvnorm, qmvnorm and rmvnorm of the package mvtnorm.

Value

Samples or trajectories of the commodity spot price and instantaneous spot convenience yield as matrix.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

See Also

schwartz2f-class description. d/p/q/state for the density, distribution, and quantile function of the state variables.

Examples

```
# ## Create a "schwartz2f"-object
# model <- schwartz2f()
#
# ## and sample from its distribution at time = 2.5.
# sim <- rstate(n = 1000, s0 = model, time = 2.5)
# par(mfrow = c(1, 2))
# hist(sim[,1], main = "Distribution of Spot Price")
# hist(sim[,2], main = "Distribution of Convenience Yield")
#
#
# ## Create a trajectory over a 6 years horizon sampled on a weekly basis.
# trajectory <- simstate(6 * 52, time = 6, s0 = model)
# par(mfrow = c(1, 2))
# plot(trajectory[,1], main = "Spot Price", type = "l")
# plot(trajectory[,2], main = "Convenience Yield", type = "l")
```

resid-futures *Extract Model Residuals*

Description

Function to extract model residuals from schwartz2f.fit-objects.

Usage

```
## S4 method for signature 'schwartz2f.fit'
resid(object, data, ttm, type = c("filter", "filter.std", "real"))
```

Arguments

object	A schwartz2f.fit-object returned from schwartz2f.fit.
data	A matrix with futures prices.
ttm	A matrix with the corresponding time to maturity (see **Details**).
type	What kind of residuals shall be returned (see **Details**).

Details

If type == "filter", then the residuals from the measurement equation are returned. If type == "filter.std", standardized residuals from the measurement equation are returned (note that these residuals should be standard multivariate normal). If type == "real", the difference between the observed futures prices and the fitted values (see fitted) are returned.

The model and its parameters are described in the **Details** section of the schwartz2f-class documentation and in the package vignette *Technical Document*.

Value

A matrix containing the residuals.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

See Also

fit.schwartz2f, schwartz2f.fit-class, fitted.

Examples

```
# data(futures)
#
# ## Estimate parameters for live.cattle data.
# ## (little precision required with reltol = 1e-3)
# fit.obj <- fit.schwartz2f(futures$live.cattle$price, futures$live.cattle$ttm / 260,
#                           deltat = 1 / 260,
#                           control = list(maxit = 100, reltol = 1e-3))
#
# ## Standardized residuals
# resid.std <- resid(fit.obj, data = futures$live.cattle$price, ttm =
#                      futures$live.cattle$ttm / 260, type = "filter.std")
# acf(resid.std, na.action = na.pass) # ...are not independent
#
# ## Real differences
# resid.real <- resid(fit.obj, data = futures$live.cattle$price, ttm =
#                       futures$live.cattle$ttm / 260, type = "real")
#
# plot(as.ts(resid.real / futures$live.cattle$price)) # ...are 'relatively' accurate.
```

---

schwartz2f-class-hierarchy

Classes schwartz2f and schwartz2f.fit

Description

The `schwartz2f` class stores parameters which determine initial values and the dynamics of the state variables. The class `schwartz2f.fit` inherits from the `schwartz2f` class. The class `schwartz2f.fit` adds slots which contain data regarding the estimation procedure and parameters of the risk-neutral dynamics. In particular, it adds the market price of convenience yield risk `lambda` and the interest rate.

Objects from the Class

Objects should only be created by calls to the constructors `schwartz2f` and `fit.schwartz2f`.

Slots

Slots of class "schwartz2f":

`call`: The function-call of class `call`.

`s0`: Initial commodity spot-price of class `numeric`.

`delta0`: Initial value of the convenience yield of class `numeric`.

`mu`: Enters the drift of the commodity spot price (under the objective measure, see **Details**) of class `numeric`.

`sigmaS`: Diffusion parameter of the spot price process of class `numeric`.

`kappaE`: Speed of mean-reversion of the convenience-yield process of class `numeric`.

`alpha`: Mean level of the convenience-yield process of class `numeric`.

`sigmaE`: Diffusion parameter of the convenience-yield process of class `numeric`.

`rhoSE`: Correlation between the two Brownian motions which drive the spot price and convenience-yield processes of class `numeric`.

Slots added by class "schwartz2f.fit":

`n.iter`: The number of iterations of class `numeric`.

`llh`: The log likelihood value of class `numeric`.

`converged`: A `logical` stating whether the fit converged or not.

`error.code`: An error code of class `numeric`. The value of `optim`'s "convergence". If an unknown error occurs the value -1 is returned.

`error.message`: An error message of class `character`, if any.

`fitted.params`: A `logical` vector stating which parameters were fitted.

`trace.pars`: Contains the parameter value evolution during the estimation procedure of class `matrix`.

`r`: The risk-free interest rate of class `numeric`.

`alphaT`: The mean-value of the convenience yield process under the equivalent martingale measure of class `numeric` (see **Details**).

`lambda`: The market price of convenience yield risk of class `numeric`.

`meas.sd`: The standard deviation of the measurement equantion of class `numeric`.

`deltat`: The time-increment of the transition equation of class `numeric`.

Details

The joint dynamics of the spot-price and the convenience yield are given by the stochastic differential equations

$$dS_t = (\mu - \delta_t)S_t dt + \sigma_S \tilde{} S_t \tilde{} dW_t^1,$$

$$d\delta_t = \kappa(\alpha - \delta_t)dt + \sigma_E dW_t^2$$

$$dW_t^1 \tilde{} dW_t^2 = \rho dt,$$

where W^1, W^2 are Brownian motions under the objective measure.

Under an equivalent martingale measure (the pricing measure) the dynamics is

$$dS_t = (r - \delta_t)S_t dt + \sigma_S \tilde{} S_t \tilde{} d\tilde{W}_t^1$$

$$d\delta_t = \kappa(\tilde{\alpha} - \delta_t)dt + \sigma_E d\tilde{W}_t^2,$$

where $\tilde{W}^1$, $\tilde{W}^2$ are Brownian motions with respect to the martingale measure.

$\tilde{\alpha} = \alpha - \lambda/\kappa$ where λ is the market price of convenience-yield risk.

Extends

Class "schwartz2f.fit" extends class "schwart2factor", directly.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

Stochastic Convenience Yield and the Pricing of Oil Contingent Claims by Rajna Gibson and Eduardo S. Schwartz
The Journal of Finance 45, 1990, 959-976

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

Pricing of Options on Commodity Futures with Stochastic Term Structures of Convenience Yields and Interest Rates by Kristian R. Miltersen and Eduardo S. Schwartz
Journal of Financial and Quantitative Analysis 33, 1998, 33-59

Valuation of Commodity Futures and Options under Stochastic Convenience Yields, Interest Rates, and Jump Diffusions in the Spot by Jimmy E. Hilliard and Jorge Reis
Journal of Financial and Quantitative Analysis 33, 1998, 61-86

See Also

`schwartz2f` to initialize `schwartz2f`-objects. `fit.schwartz2f` to fit the two-factor model to data and get a `schwartz2f.fit` object, `schwartz97-package` for an overview.

Examples

```
# obj <- schwartz2f() # create an object of class schwartz2f
# obj          # print it
# coef(obj)    # get coefficients
# unclass(obj) # see the slots
#
# ## create an object of class schwartz2f.fit
# data(futures)
# fit.obj <- fit.schwartz2f(futures$wheat$price, futures$wheat$ttm / 260,
#                           deltat = 1 / 260, control = list(maxit = 3))
# fit.obj          # print it
# coef(fit.obj)    # get coefficients
# unclass(fit.obj) # see the slots
```

schwartz2f-constructor

Create schwartz2f objects

Description

Create objects of class schwartz2f.

Usage

```
schwartz2f(s0 = 100, delta0 = 0, mu = 0.1, sigmaS = 0.3,
           kappa = 1, alpha = 0, sigmaE = 0.3, rho = 0.5)
```

Arguments

s0	Initial value of the commodity spot price.
delta0	Initial value of the convenience yield.
mu	enters the drift of the commodity spot price.
sigmaS	Diffusion parameter of the spot price-process.
kappa	Speed of mean-reversion of the convenience yield process.
alpha	Mean-level of the convenience yield process.
sigmaE	Diffusion parameter of the convenience yield process.
rho	Correlation coefficient between the Brownian motion driving the spot price and the convenience yield process.

Details

The dynamics of the Schwartz two-factor model is explained in the schwartz2f class documentation or in the package vignette in the doc-folder.

Value

An object of class schwartz2f.

Author(s)

Philipp Erb, David Luethi, Juri Hinz

References

The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging by Eduardo S. Schwartz
Journal of Finance 52, 1997, 923-973

See Also

fit.schwartz2f for parameter estimation. d/p/q/r/simstate for the density, distribution, and quantile function of the state variables and random number generation. plot-method for schwartz2f-objects.

Examples

```
# ## Initialize a 'schwartz2f' object with high convenience yield volatility:
# obj <- schwartz2f(sigmaE = 0.7)
#
# plot(obj) # plot it
#
# rstate(10, time = 1, s0 = obj) # generate 10 random variates.
#
# ## Get the probability of the event 'the spot price is >= 100 and the
# ## convenience yield is >= 0':
# pstate(c(0, -Inf), c(100, 0), time = 10, s0 = obj)
```

Index