

Package ‘ecd’

May 9, 2022

Type Package

Title Elliptic Lambda Distribution and Option Pricing Model

Version 0.9.2.3

Date 2022-05-10

Author Stephen H-T. Lihn [aut, cre]

Maintainer Stephen H-T. Lihn <stevelihn@gmail.com>

Description Elliptic lambda distribution and lambda option pricing model

have been evolved into a framework of
stable-law inspired distributions,
such as the extended stable lambda distribution for asset return,
stable count distribution for volatility,
and Lihn-Laplace process as a leptokurtic extension of Wiener process.
This package contains functions for the computation of
density, probability, quantile, random variable, fitting procedures,
option prices, volatility smile. It also comes with sample financial data,
and plotting routines.

URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3046732

Depends R (>= 3.5.1)

Imports stats, utils, Rmpfr (>= 0.6-0), gsl, polynom, xts, zoo,
optimx, moments, stabledist, parallel, graphics, ggplot2,
gridExtra, xtable, methods, yaml, RSQLite, digest

Suggests knitr, testthat, roxygen2, shape

License Artistic-2.0

Encoding latin1

Collate 'ecd-adj-gamma-method.R' 'ecd-asymp-stats-method.R'
'ecd-ccdf-method.R' 'ecd-cdf-method.R'
'ecd-numericMpfr-class.R' 'ecdq-class.R' 'ecd-package.R'
'ecd-class.R' 'ecd-constructor.R' 'ecd-cubic-method.R'
'ecd-cusp-a2r-method.R' 'ecd-cusp-constructor.R'
'ecd-cusp-std-moment-method.R' 'ecd-data-config-internal.R'
'ecd-data-method.R' 'ecd-data-stats-method.R'
'ecd-df2ts-method.R' 'ecd-diff-method.R' 'ecd-distr-generic.R'
'ecd-distribution-method.R' 'eclid-class.R'
'ecd-eclidOrEcd-class.R' 'ecd-ellipticity-generic.R'
'ecd-erfq-method.R' 'ecd-estimate-const-method.R'
'ecd-fit-data-method.R' 'ecd-fit-ts-conf-method.R'

```
'ecd-has-quantile-method.R' 'ecd-imgf-method.R'
'ecd-integrate-method.R' 'ecd-integrate-pdf-generic.R'
'ecd-is-numericMpfr-internal.R' 'ecd-jinv-generic.R'
'ecd-lag-method.R' 'ecd-manage-hist-tails-method.R'
'ecd-max-kurtosis-method.R' 'ecd-model-internal.R'
'ecd-moment-generic.R' 'ecd-mp2f-method.R' 'ecd-mpfr-method.R'
'ecd-mpfr-qagi-method.R' 'ecd-mpnum-method.R'
'ecd-ogf-method.R' 'ecd-pdf-method.R' 'ecd-plot-2x2-generic.R'
'ecd-polar-constructor.R' 'ecd-quantilize-generic.R'
'ecd-rational-method.R' 'ecd-read-csv-by-symbol-method.R'
'ecd-read-symbol-conf-method.R' 'ecd-sd-method.R'
'ecd-setup-const-method.R' 'ecd-solve-cusp-asym-method.R'
'ecd-solve-generic.R' 'ecd-solve-sym-generic.R'
'ecd-solve-trig-generic.R' 'ecd-standardfit-method.R'
'ecd-stats-method.R' 'ecd-toString-method.R'
'ecd-ts-lag-stats-method.R' 'ecd-uniroot-method.R'
'ecd-y-slope-generic.R' 'ecd-y0-isomorphic-method.R'
'ecdattr-class.R' 'ecdattr-constructor.R'
'ecdattr-enrich-method.R' 'ecdattr-pairs-method.R'
'ecdattr-pairs-polar-method.R' 'ecdb-class.R'
'ecdb-bootstrap-generic.R' 'ecdb-constructor.R'
'ecdb-dbSendQuery-method.R' 'ecdb-ecdattr-generic.R'
'ecdb-helpers-internal.R' 'ecdb-history-generic.R'
'ecdb-protectiveCommit-method.R' 'ecdb-read-generic.R'
'ecdb-summary-generic.R' 'ecdb-write-generic.R'
'ecdq-constructor.R' 'ecld-cdf-method.R' 'ecld-const-method.R'
'ecld-constructor.R' 'ecld-fixed-point-RN0-method.R'
'ecld-gamma-method.R' 'ecld-imgf-method.R' 'ecld-imnt-method.R'
'ecld-ivol-ogf-star-method.R' 'ecld-mgf-term-method.R'
'ecld-moment-method.R' 'ecld-mpnum-method.R'
'ecld-mu-D-method.R' 'ecld-ogf-method.R'
'ecld-ogf-star-method.R' 'ecld-op-Q-method.R'
'ecld-op-V-method.R' 'ecld-pdf-method.R'
'ecld-quartic-RN0-method.R' 'ecld-sd-method.R'
'ecld-sged-method.R' 'ecld-solve-method.R'
'ecld-y-slope-method.R' 'ecld.quartic-Qp-method.R'
'ecld.quartic_Qp_atm_attr.R'
'ecop-bs-implied-volatility-method.R'
'ecop-bs-option-price-method.R' 'ecop-opt-class.R'
'ecop-class.R' 'ecop-constructor.R'
'ecop-find-fixed-point-lambda-by-atm-skew-method.R'
'ecop-find-fixed-point-sd-by-lambda-method.R'
'ecop-get-ld-triple-method.R' 'ecop-polyfit-option-method.R'
'ecop-read-csv-by-symbol.R'
'ecop-term-master-calculator-method.R'
'ecop-term-plot-3x3-method.R' 'ecop-vix-plot-3x3-method.R'
'lamp-class.R' 'lamp-constructor.R'
'lamp-generate-tau-method.R' 'lamp-k2mnt-method.R'
'lamp-laplace-distribution-method.R' 'lamp-plot-sim4-method.R'
'lamp-qsl-analytic-method.R' 'lamp-qsl-fit-config-method.R'
'lamp-qsl-fit-plot-method.R' 'lamp-sd-factor-method.R'
'lamp-simulate-iter-method.R' 'lamp-simulate1-method.R'
```

```
'lamp-stable-cnt-distribution-method.R'
'lamp-stable-lambda-dist-method.R'
'lamp-stable-rnd-walk-method.R'
'lamp-stdlap-distribution-method.R' 'levy-dlambda-method.R'
'levy-domain-coloring-method.R'
'levy-dskewed-distribution-method.R' 'qslid-fit-method.R'
'sld-class.R' 'sld-constructor.R' 'sld-sd-method.R'
```

RoxygenNote 7.1.2

NeedsCompilation no

R topics documented:

ecd-package	6
bootstrap.ecdb	6
dec	7
discr.ecd	8
dsl	8
dstablecnt	11
dstdlap	12
ecd	14
ecd-class	15
ecd.adj_gamma	16
ecd.asymp_stats	17
ecd.ccdf	17
ecd.cdf	18
ecd.cubic	19
ecd.cusp	20
ecd.cusp_a2r	21
ecd.cusp_std_moment	21
ecd.data	22
ecd.data_stats	23
ecd.df2ts	24
ecd.diff	25
ecd.erfq	26
ecd.estimate_const	26
ecd.fit_data	27
ecd.fit_ts_conf	28
ecd.has_quantile	29
ecd.imgf	29
ecd.integrate	30
ecd.lag	31
ecd.manage_hist_tails	32
ecd.max_kurtosis	32
ecd.mp2f	33
ecd.mpfr	34
ecd.mpfr_qagi	35
ecd.mppnum	36
ecd.ofg	36
ecd.pdf	37
ecd.polar	38
ecd.rational	39

ecd.read_csv_by_symbol	39
ecd.read_symbol_conf	40
ecd.sd	41
ecd.setup_const	41
ecd.solve_cusp_asym	42
ecd.stats	43
ecd.toString	43
ecd.ts_lag_stats	44
ecd.uniroot	45
ecd.y0_isomorphic	45
ecdattr	46
ecdattr-class	47
ecdattr.enrich	47
ecdattr.pairs	48
ecdattr.pairs_polar	48
ecdb	49
ecdb-class	49
ecdb.dbSendQuery	50
ecdb.protectiveCommit	50
ecdq	51
ecdq-class	51
ecld	52
ecld-class	53
ecld.cdf	55
ecld.const	56
ecld.fixed_point_SN0_atm_ki	56
ecld.gamma	57
ecld.imgf	58
ecld.imnt	59
ecld.ivol_ogf_star	60
ecld.mgf_term	61
ecld.moment	62
ecld.mpnum	62
ecld.mu_D	63
ecld.ogf	64
ecld.ogf_star	65
ecld.op_Q	66
ecld.op_V	67
ecld.pdf	68
ecld.quartic_Qp	69
ecld.quartic_Qp_atm_attr	70
ecld.quartic_SN0_atm_ki	71
ecld.sd	71
ecld.sged_const	72
ecld.solve	73
ecld.y_slope	74
ecldOrEcd-class	75
ecop-class	75
ecop.bs_implied_volatility	76
ecop.bs_option_price	77
ecop.find_fixed_point_lambda_by_atm_skew	78
ecop.find_fixed_point_sd_by_lambda	79

ecop.from_symbol_conf	80
ecop.get_ld_triple	81
ecop.opt-class	82
ecop.polyfit_option	82
ecop.read_csv_by_symbol	83
ecop.term_master_calculator	84
ecop.term_plot_3x3	85
ecop.vix_plot_3x3	86
ellipticity.ecd	86
history.ecdb	87
integrate_pdf.ecd	87
jinv.ecd	89
k2mnt	89
lamp	90
lamp-class	91
lamp.generate_tau	92
lamp.plot_sim4	92
lamp.qsl_fit_config	93
lamp.qsl_fit_plot	94
lamp.sd_factor	94
lamp.simulate1	95
lamp.simulate_iter	96
lamp.stable_rnd_walk	97
levy.dlambda	97
levy.domain_coloring	98
levy.dskeewed	99
moment.ecd	99
numericMpfr-class	100
plot_2x2.ecd	101
qsld.fit	102
qsl_kurtosis_analytic	103
quantilize.ecd	104
read.ecdb	105
rlaplace0	106
sld	107
sld-class	108
sld.sd	109
solve.ecd	110
solve_sym.ecd	110
solve_trig.ecd	111
summary.ecdb	112
write.ecdb	112
y_slope.ecd	113

ecd-package

*ecd: A package for the stable lambda distribution family.***Description**

The ecd package provides the core classes and functions for the stable lambda distribution family. The stable lambda distribution is implemented in [ds1](#) section. The lambda distribution uses the ecld namespace. SGED is considered part of ecld. (See [ecld-class](#) for definition.) The original elliptic lambda distribution uses the generic methods or ecd namespace. (See [ecd-class](#) for definition.) The option pricing API uses the ecop namespace. (See [ecop-class](#) for definition.) Most helper utilities are named under either ecd or ecld.

Author(s)

Stephen H-T. Lihn

See Also

The two main classes are [ecd-class](#) and [ecld-class](#)

bootstrap.ecdb

*Bootstrap data for the Elliptic DB (ECDB)***Description**

Main interface to generate data for ECDB based on the configuration.

Usage

```
## S3 method for class 'ecdb'
bootstrap(object, action = "all", skip.exists = TRUE)

bootstrap(object, action = "all", skip.exists = TRUE)

## S4 method for signature 'ecdb'
bootstrap(object, action = "all", skip.exists = TRUE)
```

Arguments

<code>object</code>	an object of ecdb class.
<code>action</code>	the action operating on the ecdb.
<code>skip.exists</code>	logical, if TRUE (default), skip if action already done in history.

Value

Row count.

Description

Density, distribution function, quantile function, and random generation for the univariate elliptic distribution.

Usage

```
dec(x, object = ecd())
pec(q, object = ecd())
qec(p, object = ecd(with.quantile = TRUE), debug = FALSE)
rec(n, object = ecd(with.quantile = TRUE))
```

Arguments

x	numeric vector of quantiles.
object	an object of ecd-class . To achieve high performance for qec and rec, it should be created with with.quantile=TRUE.
q	numeric vector of quantiles.
p	numeric vector of probabilities.
debug	logical, whether to print debug message, default is FALSE.
n	number of observations.

Value

dec gives the density, pec gives the distribution function, qec gives the quantile function, rec generates random deviates.

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(with.quantile=TRUE)
x <- seq(-20, 20, by=5)
dec(x,d)
pec(x,d)
p <- c(0.0001, 0.001, 0.01, 0.99, 0.999, 0.9999)
qec(p,d)
rec(100,d)
```

discr.ecd*Discriminant of the elliptic curve $y(x)$* **Description**

Discriminant of the elliptic curve $y(x)$

Usage

```
## S3 method for class 'ecd'
discr(object, no.validate = FALSE)

discr(object, no.validate = FALSE)

## S4 method for signature 'ecd'
discr(object, no.validate = FALSE)
```

Arguments

<code>object</code>	an object of ecd class
<code>no.validate</code>	logical, if TRUE, don't validate presence of beta. Default is FALSE.

Value

the discriminant

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd(-1,1)
discr(d)
```

dsl*Stable lambda distribution***Description**

Implements the stable lambda distribution (SLD) and the quartic stable lambda distribution (QSLD). Be aware of the performance concerns: (a) The cumulative density function is implemented by direct integration over the density. (b) The quantile function is implemented by root finding on cumulative density function.

Usage

```

dsI(x, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0, lambda = 4)

dqsl(x, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0)

rqsl(n, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0)

rsl(n, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0, lambda = 4)

pqsl(x, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0)

psl(x, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0, lambda = 4)

qqlsI(q, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0)

qsl(q, t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0, lambda = 4)

kqsl(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0)

ksl(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0, lambda = 4)

cfqsl(
  s,
  t = 1,
  nu0 = 0,
  theta = 1,
  convo = 1,
  beta.a = 0,
  mu = 0,
  method = "a"
)

cfsI(
  s,
  t = 1,
  nu0 = 0,
  theta = 1,
  convo = 1,
  beta.a = 0,
  mu = 0,
  lambda = 4,
  method = "a"
)

```

Arguments

x	numeric, vector of responses.
t	numeric, the time parameter, where the variance is t, default is 1.
nu0	numeric, the location parameter, default is 0.
theta	numeric, the scale parameter, default is 1.
convo	numeric, the convolution number, default is 1.

beta.a	numeric, the skewness parameter, default is 0. This number is annualized by sqrt(t).
mu	numeric, the location parameter, default is 0.
lambda	numeric, the shape parameter, default is 4.
n	numeric, number of observations.
q	numeric, vector of quantiles.
s	numeric, vector of responses for characteristic function.
method	character, method of characteristic function (CF) calculation. Default is "a". Method a uses cfstdlap x dstablecnt. Method b uses dstdlap x cfstablecnt. Method c uses direct integration on PDF up to 50 stdev. They should yield the same result.

Value

numeric, standard convention is followed: d* returns the density, p* returns the distribution function, q* returns the quantile function, and r* generates random deviates. The following are our extensions: k* returns the first 4 cumulants, skewness, and kurtosis, cf* returns the characteristic function.

Details

The stable lambda distribution is the stationary distribution for financial asset returns. It is a product of the stable count distribution and the Lihn-Laplace process. The density function is defined as

$$P_{\Lambda}^{(m)}(x; t, \nu_0, \theta, \beta_a, \mu) \equiv \int_{\nu_0}^{\infty} \frac{1}{\nu} f_L^{(m)}\left(\frac{x - \mu}{\nu}; t, \beta_a \sqrt{\nu}\right) N_{\alpha}(\nu; \nu_0, \theta) d\nu$$

where $f_L^{(m)}(.)$ is the Lihn-Laplace distribution and $N_{\alpha}(.)$ is the quartic stable count distribution. t is the time or sampling period, α is the stability index which is $2/\lambda$, ν_0 is the floor volatility parameter, θ is the volatility scale parameter, β_a is the annualized asymmetric parameter, μ is the location parameter.

The quartic stable lambda distribution (QSLD) is a specialized distribution with $\lambda = 4$ aka $\alpha = 0.5$. The PDF integrand has closed form, and all the moments have closed forms. Many financial asset returns can be fitted by QSLD precisely up to 4 standard deviations.

Author(s)

Stephen H-T. Lihn

References

For more detail, see Section 8.2 of Stephen Lihn (2017). *A Theory of Asset Return and Volatility under Stable Law and Stable Lambda Distribution*. SSRN: 3046732, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3046732.

See Also

`dstablecnt` for $N_{\alpha}(.)$, and `dstdlap` for $f_L^{(m)}(.)$.

Examples

```
# generate the quartic pdf for SPX 1-day distribution
x <- c(-0.1, 0.1, by=0.001)
pdf <- dqs1(x, t=1/250, nu0=6.92/100, theta=1.17/100, convo=2, beta=-1.31)
```

dstablecnt

Stable Count distribution

Description

Implements the stable count distribution (based on stabledist package) for stable random walk simulation. Quartic stable distribution is implemented through gamma distribution.

Usage

```
dstablecnt(x, alpha = NULL, nu0 = 0, theta = 1, lambda = NULL)

pstablecnt(x, alpha = NULL, nu0 = 0, theta = 1, lambda = NULL)

rstablecnt(n, alpha = NULL, nu0 = 0, theta = 1, lambda = NULL)

qstablecnt(q, alpha = NULL, nu0 = 0, theta = 1, lambda = NULL)

cfstablecnt(s, alpha = NULL, nu0 = 0, theta = 1, lambda = NULL)

kstablecnt(alpha = NULL, nu0 = 0, theta = 1, lambda = NULL)
```

Arguments

x	numeric, vector of responses.
alpha	numeric, the shape parameter, default is NULL. User must provide either alpha or lambda.
nu0	numeric, the location parameter, default is 0.
theta	numeric, the scale parameter, default is 1.
lambda	numeric, alternative shape parameter, default is NULL.
n	numeric, number of observations.
q	numeric, vector of quantiles.
s	numeric, vector of responses for characteristic function.

Value

numeric, standard convention is followed: d* returns the density, p* returns the distribution function, q* returns the quantile function, and r* generates random deviates. The following are our extensions: k* returns the first 4 cumulants, skewness, and kurtosis, cf* returns the characteristic function.

Details

The stable count distribution is the conjugate prior of the stable distribution. The density function is defined as

$$N_{\alpha}(\nu; \nu_0, \theta) \equiv \frac{\alpha}{\Gamma(\frac{1}{\alpha})} \frac{1}{\nu - \nu_0} L_{\alpha}\left(\frac{\theta}{\nu - \nu_0}\right), \text{ where } \nu > \nu_0.$$

where $\nu > \nu_0$. α is the stability index, ν_0 is the location parameter, and θ is the scale parameter. At $\alpha = 0.5$ aka $\lambda = 4$, it is called "quartic stable count distribution", which is a gamma distribution with shape of 3/2. It has the closed form of

$$N_{\frac{1}{2}}(\nu; \nu_0, \theta) \equiv \frac{1}{4\sqrt{\pi} \theta^{3/2}} (\nu - \nu_0)^{\frac{1}{2}} e^{-\frac{\nu-\nu_0}{4\theta}}$$

Author(s)

Stephen H-T. Lihn

References

For more detail, see Section 2.4 and Section 3.3 of Stephen Lihn (2017). *A Theory of Asset Return and Volatility under Stable Law and Stable Lambda Distribution*. SSRN: 3046732, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3046732. This distribution is also documented formally in Wikipedia: https://en.wikipedia.org/wiki/Stable_count_distribution.

Examples

```
# generate the pdf of the VIX distribution
x <- c(0, 100, by=0.1)
pdf <- dstablecnt(x, nu0=10.4, theta=1.6, lambda=4)
```

dstdlap

Standardized Laplace process and distribution

Description

Implements the standardized Laplace process and distribution. Be aware of the performance concerns: (a) The cumulative density function is implemented by direct integration over the density. (b) The quantile function is implemented by root finding on cumulative density function.

Usage

```
dstdlap(x, t = 1, convo = 1, beta = 0, mu = 0)

pstdlap(x, t = 1, convo = 1, beta = 0, mu = 0)

qstdlap(q, t = 1, convo = 1, beta = 0, mu = 0)

rstdlap(n, t = 1, convo = 1, beta = 0, mu = 0)

cfstdlap(s, t = 1, convo = 1, beta = 0, mu = 0)
```

```
kstdlap(t = 1, convo = 1, beta = 0, mu = 0)
dstdlap_poly(x, t = 1, convo = 1, beta = 0, mu = 0)
```

Arguments

x	numeric, vector of responses.
t	numeric, the time parameter, of which the variance is t.
convo	numeric, the convolution number, default is 1, which is Laplace without convolution. There is a special provision in rstdlap, where it will simulate the Wiener process if convo=Inf and beta=0.
beta	numeric, skewness parameter according to skewed lambda distribution, default is 0.
mu	numeric, location parameter, default is 0.
q	numeric, vector of quantiles.
n	numeric, number of observations.
s	numeric, vector of responses for characteristic function.

Value

numeric, standard convention is followed: d* returns the density, p* returns the distribution function, q* returns the quantile function, and r* generates random deviates. The following are our extensions: k* returns the first 4 cumulants, skewness, and kurtosis, cf* returns the characteristic function.

Details

The Lihn-Laplace distribution is the stationary distribution of Lihn-Laplace process. The density function is defined as

$$f_L^{(m)}(x; t, \beta, \mu) \equiv \frac{1}{\sqrt{\pi} \Gamma(m) \sigma_m} \left| \frac{x - \mu}{2B_0 \sigma_m} \right|^{m-\frac{1}{2}} K_{m-\frac{1}{2}} \left(\left| \frac{B_0(x - \mu)}{\sigma_m} \right| \right) e^{\frac{\beta(x - \mu)}{2\sigma_m}}$$

where

$$\sigma_m \equiv \sqrt{\frac{t}{m(2 + \beta^2)}}, \quad B_0 \equiv \sqrt{1 + \frac{1}{4}\beta^2}.$$

$K_n(x)$ is the modified Bessel function of the second kind. t is the time or sampling period, β is the asymmetric parameter, μ is the location parameter.

Author(s)

Stephen H-T. Lihn

References

For more detail, see Section 5.4 of Stephen Lihn (2017). *A Theory of Asset Return and Volatility under Stable Law and Stable Lambda Distribution*. SSRN: 3046732, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3046732.

Examples

```
# generate the pdf at time t=1 for the second convolution and beta = 0.1 for skewness
x <- c(-10, 10, by=0.1)
pdf <- dstdlap(x, t=1, convo=2, beta=0.1)
```

ecd

Constructor of ecd class

Description

Construct an ecd class by providing the required parameters. The default is the standard cusp distribution. Cusp is validated by $\text{eps} = \max(\text{Machine}\$double.\text{eps} * 1000, 1e-28)$.

Usage

```
ecd(
  alpha = 0,
  gamma = 0,
  sigma = 1,
  beta = 0,
  mu = 0,
  cusp = 0,
  lambda = 3,
  with.stats = TRUE,
  with.quantile = FALSE,
  bare.bone = FALSE,
  verbose = FALSE
)
```

Arguments

alpha	numeric, the flatness parameter. Default: 0.
gamma	numeric, the sharpness parameter. Default: 0.
sigma	numeric, the scale parameter. Must be positive. Default: 1.
beta	numeric, the skewness parameter. Default: 0.
mu	numeric, the location parameter. Default: 0.
cusp	logical, indicate type of cusp. The singular points in cusp requires special handling. Default: 0, not a cusp. 1: cusp with alpha specified. 2: cusp with gamma specified.
lambda	numeric, the leading exponent for the special model. Default: 3.
with.stats	logical, also calculate statistics, default is TRUE.
with.quantile	logical, also calculate quantile data, default is FALSE.
bare.bone	logical, skip both const and stats calculation, default is FALSE. This for debug purpose for issues on integrating $e^y(x)$.
verbose	logical, display timing information, for debugging purpose, default is FALSE.

Value

An object of ecd class

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
d <- ecd(1,1)
d <- ecd(alpha=1, gamma=1)
```

ecd-class

The ecd class

Description

This S4 class is the major object class for elliptic lambda distribution. It stores the ecd parameters, numerical constants that facilitates quadpack integration, statistical attributes, and optionally, an internal structure for the quantile function.

Slots

call The match.call slot
 alpha, gamma, sigma, beta, mu a length-one numeric. These are core ecd parameters.
 cusp a length-one numeric as cusp indicator. 0: not a cusp; 1: cusp specified by alpha; 2: cusp specified by gamma.
 lambda a length-one numeric, the leading exponent for the special model, default is 3.
 R, theta a length-one numeric. These are derived ecd parameters in polar coordinate.
 use.mpfr logical, internal flag indicating whether to use mpfr.
 const A length-one numeric as the integral of $\exp(y(x))$ that normalizes the PDF.
 const_left_x A length-one numeric marking the left point of PDF integration.
 const_right_x A length-one numeric marking the right point of PDF integration.
 stats A list of statistics, see ecd.stats for more information.
 quantile An object of ecdq class, for quantile calculation.
 model A vector of four strings representing internal classification: long_name.skew, code.long_name,
 short_name.skew, short_name. This slot doesn't have formal use yet.

Details

The elliptic lambda distribution is the early research target of what becomes the stable lambda distribution. It is inspired from elliptic curve, and is defined by a depressed cubic polynomial,

$$-y(z)^3 - (\gamma + \beta z) y(z) + \alpha = z^2,$$

where $y(z)$ must approach minus infinity as z approaches plus or minus infinity. The density function is defined as

$$P(x; \alpha, \gamma, \sigma, \beta, \mu) \equiv \frac{1}{C\sigma} e^{y(|\frac{x-\mu}{\sigma}|)},$$

and C is the normalization constant,

$$C = \int_{-\infty}^{\infty} e^{y(z)} dz,$$

where α and γ are the shape parameters, σ is the scale parameter, β is the asymmetric parameter, μ is the location parameter.

Author(s)

Stephen H. Lihn

References

For elliptic lambda distribution, see Stephen Lihn (2015). *On the Elliptic Distribution and Its Application to Leptokurtic Financial Data*. SSRN: <https://www.ssrn.com/abstract=2697911>

`ecd.adj_gamma`

Discriminant-adjusted gamma

Description

Adjust gamma by discriminant conversion formula so that the critical line is a straight 45-degree line. The inverse adjustment is also provided.

Usage

```
ecd.adj_gamma(gamma)
```

```
ecd.adj2gamma(adj_gamma)
```

Arguments

<code>gamma</code>	numeric, the gamma parameter
<code>adj_gamma</code>	numeric, the discriminant-adjusted gamma

Value

adjusted gamma (or the reverse of adjustment)

Examples

```
gamma2 <- ecd.adj_gamma(c(1,2))
gamma <- ecd.adj2gamma(c(1,2))
```

<code>ecd.asymp_stats</code>	<i>Compute asymptotic statistics of an ecd object</i>
------------------------------	---

Description

The main API for asymptotic statistics. It follows the same definition of moments, except the integral of PDF is limited to a range of quantile. That is to truncate the tails. The asymptotic kurtosis is also called truncated kurtosis.

Usage

```
ecd.asymp_stats(object, q)
ecd.asymp_kurtosis(object, q)
```

Arguments

<code>object</code>	an object of ecd class with quantile
<code>q</code>	numeric vector of quantiles

Value

a list of stats list, or a vector of kurtosis

Examples

```
## Not run:
d <- ecd(1,1, with.quantile=TRUE)
q <- 0.01
ecd.asymp_stats(d,q)
ecd.asymp_kurtosis(d,q)

## End(Not run)
```

<code>ecd.ccdf</code>	<i>Complementary CDF of ecd</i>
-----------------------	---------------------------------

Description

Complementary CDF of ecd, integration of PDF from x to Inf

Usage

```
ecd.ccdf(object, x, to.x = Inf, piece.wise = FALSE, f = NULL, verbose = FALSE)
```

Arguments

object	An object of ecd class
x	A numeric vector of x
to.x	A value or a vector of starting x, default Inf This is for internal use only.
piece.wise	Logical. If TRUE, use cumulative method for large array. Default to FALSE. Use it with a scalar to.x.
f	an optional extension to perform integral on function other than 1. This is for internal use only. You should use the respective wrappers.
verbose	logical, display timing information, for debugging purpose.

Value

The CCDF

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
x <- seq(0, 10, by=1)
ecd.ccdf(d,x)
```

ecd.cdf

CDF of ecd

Description

CDF of ecd, integration of PDF from -Inf (or a point of choice) to x

Usage

```
ecd.cdf(
  object,
  x,
  from.x = -Inf,
  piece.wise = FALSE,
  f = NULL,
  verbose = FALSE
)
```

Arguments

object	An object of ecd class
x	A numeric vector of x
from.x	A value or a vector of starting x, default -Inf
piece.wise	Logical. If TRUE, use cumulative method for large array. Default to FALSE. Use it with a scalar from.x.

f	an optional extension to perform integral on function other than 1. This is for internal use only. You should use the respective wrappers.
verbose	logical, display timing information, for debugging purpose.

Value

The CDF

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
x <- seq(-10, 10, by=1)
ecd.cdf(d,x)
ecd.cdf(d,1, from.x = -1)
```

ecd.cubic

Generate or solve the cubic polynomial for ecd

Description

Generate or solve the polynomial from ecd. This is usually transient for solve. Or it can be used for studying singular points.

Usage

```
ecd.cubic(object, x = 0, solve = TRUE)
```

Arguments

object	An object of ecd class
x	A vector of x dimension
solve	Logical, solve the polynomial, default = TRUE.

Value

list of the polynomial object, or result of solve.

Examples

```
d <- ecd()
ecd.cubic(d)
ecd.cubic(d, 0)
```

ecd.cusp*Cusp constructor of ecd class***Description**

Construct an ecd class for cusp distribution by specifying either alpha or gamma, but not both. At the moment, it can't handle beta.

Usage

```
ecd.cusp(
  alpha = NaN,
  gamma = NaN,
  sigma = 1,
  mu = 0,
  with.stats = TRUE,
  with.quantile = FALSE,
  bare.bone = FALSE,
  verbose = FALSE
)
```

Arguments

<code>alpha</code>	numeric, the flatness parameter. Default: NaN.
<code>gamma</code>	numeric, the sharpness parameter. Default: NaN.
<code>sigma</code>	numeric, the scale parameter. Must be positive. Default 1.
<code>mu</code>	numeric, the location parameter. Default: 0.
<code>with.stats</code>	logical, also calculate statistics, default is TRUE.
<code>with.quantile</code>	logical, also calculate quantile data, default is FALSE.
<code>bare.bone</code>	logical, skip both const and stats calculation, default is FALSE. This for debug purpose for issues on integrating $e^y(x)$.
<code>verbose</code>	logical, display timing information, for debugging purpose, default is FALSE.

Value

The ecd class

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd.cusp(alpha=1)
d <- ecd.cusp(gamma=-1)
```

ecd.cusp_a2r

*Conversion between alpha and gamma for cusp distribution***Description**

ecd.cusp_a2r converts from alpha to gamma. ecd.cusp_r2a converts from gamma to alpha.

Usage

```
ecd.cusp_a2r(alpha)
ecd.cusp_r2a(gamma)
```

Arguments

alpha	numeric
gamma	numeric

Value

gamma for a2r; alpha for r2a.

Author(s)

Stephen H-T. Lihn

Examples

```
gamma <- ecd.cusp_a2r(alpha=1)
alpha <- ecd.cusp_r2a(gamma=1)
```

ecd.cusp_std_moment

*The moments, characteristic function (CF), and moment generating function (MGF) of standard cusp distribution.***Description**

The moments of standard cusp distribution are calculated via Gamma function. The CF and MGF are calculated as sum of moment terms. The CF is a complex number. Since the terms in MGF is ultimately diverging, the sum is truncated before the terms are increasing.

Usage

```
ecd.cusp_std_moment(n)
ecd.cusp_std_cf(t, mu = 0, sigma = 1, rel.tol = 1e-08, show.warning = FALSE)
ecd.cusp_std_mgf(t, mu = 0, sigma = 1, rel.tol = 1e-07, show.warning = FALSE)
```

Arguments

n	integer vector specifying the n-th moments
t	numeric vector for CF and MGF
mu	length-one numeric, specifying mean for CF and MGF
sigma	length-one numeric, specifying volatility for CF and MGF
rel.tol	relative tolerance
show.warning	logical, to show warning or not.

Value

the values of the moments, CF, MGF

Examples

```
ecd.cusp_std_moment(c(2,4))
```

ecd.data

Read sample data

Description

Read sample data by specifying the symbol. The two utilities, ecd.data and ecd.data.arr, serves for slightly different purpose. ecd.data works off the xts object that has two rows: the prices and log-returns indexed by the dates. ecd.data.arr and ecd.data.ts separate the data into list of three vectors: x is the log-return, p is the prices, and d is the dates. And allows for more sophisticated call for range of dates, and different ways of slice and lag. ecd.data.arr takes symbol as input, while ecd.data.ts takes an xts object.

Usage

```
ecd.data(symbol = "dji")

ecd.data.arr(
  symbol = "dji",
  start.date = "1950-01-01",
  end.date = "2015-12-31",
  on = "days",
  lag = 1,
  drop = 0,
  repeated = TRUE,
  cache = TRUE,
  do.kurtosis = FALSE
)

ecd.data.ts(
  ts,
  start.date = "1950-01-01",
  end.date = "2015-12-31",
```

```

    on = "days",
    lag = 1,
    drop = 0,
    repeated = TRUE,
    do.kurtosis = FALSE
)

```

Arguments

symbol	character, the symbol of the time series. Default: dji
start.date, end.date	Date or character of ISO format (YYYY-MM-DD), to specify the date range, default is from 1950-01-01 to 2015-12-31. Set start.date and end.date to NULL or "" if you wish to get the entire time series.
on	character, specify the calendar interval, days, weeks, months. Default is days.
lag	integer, specify the lags of return calculation, default is 1.
drop	integer, specify number of largest outliers to drop, default is 0.
repeated	logical, specify whether to use repeated sampling or unique sampling, default is TRUE. Using "repeated" sampling can reduce noise due to insufficient sample size. This is particularly useful for larger lags.
cache	logical, use R's options memory to cache xts data, default is TRUE.
do.kurtosis	logical, if specified, calculate mean, sd, var, skewness, and kurtosis, default is FALSE.
ts	xts, the time series

Value

ecd.data returns an xts object for the time series, with two columns - "Close" and "logr". ecd.data.arr and ecd.data.ts return a list of three vectors: x is the log-return, p is the prices, and d is the dates.

Examples

```

dji <- ecd.data()
wti <- ecd.data("wti")
spx <- ecd.data.arr("spx", lag=5)

```

ecd.data_stats	<i>Statistics and histogram on log returns</i>
----------------	--

Description

Statistics and histogram on log returns are added to the xts attributes

Usage

```
ecd.data_stats(
  ts = "dji",
  breaks = 20,
  merge_tails = c(0, 0),
  with.tail = FALSE,
  tail.N1 = 7,
  tail.N2 = 5
)
```

Arguments

ts	can be either a symbol of sample data, or the xts object from sample data
breaks	A length-one numeric, breaks for generating the histogram.
merge_tails	A length-two numeric vector. The first element is how many points in the left tail of histogram to be dropped during fitting. The second element is how many points in the right tail of histogram to be dropped during fitting.
with.tail	logical, include tail statistics, mainly on asymptotic kurtosis. Default: FALSE.
tail.N1	a numeric, defining the wider range of tail statistics
tail.N2	a numeric, defining the smaller range of tail statistics

Value

The xts object containing ecd added attributes

Examples

```
dji <- ecd.data_stats(ecd.data("dji"))
dji <- ecd.data_stats("dji")
```

ecd.df2ts

Utility to standardize timeseries from data.frame to xts

Description

This utility converts the df input to an xts object with columns and statistics required for the fitting/plot utility in the ecd package. The required columns are Date, Close, logr. This utility can also be used to convert the input from Quandl.

Usage

```
ecd.df2ts(
  df,
  date_format = "%m/%d/%Y",
  dt = "Date",
  col_in = "Close",
  col_out = "Close",
  do.logr = TRUE,
  rnd.zero = 0.01
)
```

Arguments

df	Data.frame of the time serie
date_format	Character, date format of the input date column. It can be NULL to indicate no date conversion is needed. Default: "%m/%d/%Y".
dt	Character, the name of the input date column. Default: "Date"
col_in	Character, the name of the input closing price column. Default: "Close"
col_out	Character, the name of the output closing price column. Default: "Close"
do.logr	logical, if TRUE (default), produce xts object of logr; otherwise, just the col_out column.
rnd.zero	numeric, a small random factor (scaled to sd of logr) to avoid an unreal peak of zero log-returns.

Value

The xts object for the time series

Examples

```
## Not run:
ecd.df2ts(df)

## End(Not run)
```

ecd.diff

Utility to diff a vector of numeric or mpfr to get first derivative

Description

This utility uses diff to get first derivative dy/dx. but it handles mpfr vector properly

Usage

```
ecd.diff(y, x, pad = 0)
```

Arguments

y	a vector of numeric or mpfr
x	a vector of numeric or mpfr
pad	integer, to manage padding so that the output vector has the same length as the input. 0 for no padding, 1 to repeat the first element, -1 to repeat the last element.

Value

the derivative vector

Examples

```
d <- ecd.diff(c(10,20,30), c(1,2,3), pad=1)
```

ecd.erfq*Quartic scaled error function***Description**

The scaled error function in quartic pricing model that encapsulates both scaled erfi and erfc functions into a single representation. This is used to provide an elegant expression for the MGF and local option prices, $L_{c,p}$. When sgn=-1, it is $\sqrt{\pi}e^{-x^2}erfi(x)$, which twice of Dawson function. When sgn=1, it is $\sqrt{\pi}e^{x^2}erfc(x)$. ecd.erfq_sum is the summation implementation with truncation rule set forth in the quartic pricing model. It achieves high precision when $x > 4.5$.

Usage

```
ecd.erfq(x, sgn)
ecd.erfq_sum(x, sgn)
```

Arguments

<code>x</code>	numeric
<code>sgn</code>	an integer of 1 or -1

Value

The `mpfr` object

Examples

```
x <- ecd.erfq(c(5,10,15), 1)
y <- ecd.erfq(c(5,10,15), -1)
```

ecd.estimate_const*Estimate the normalization constant for an ecd object***Description**

This is an internal helper function for ecd constructor. Its main function is to estimate const using analytical formula, without any dependency on statistics and numerical integration.

Usage

```
ecd.estimate_const(object)
```

Arguments

<code>object</code>	An object of ecd class
---------------------	------------------------

Value

numeric, estimated const

Examples

```
ecd.estimate_const(ecd(100,100, sigma=0.1, bare.bone=TRUE))
```

`ecd.fit_data`

Sample data fit

Description

Fitting sample data to ecd with a starting set of parameters. This is the highest level wrapper of the fitting routine.

Usage

```
ecd.fit_data(
  symbol = "dji",
  iter = 1000,
  FIT = FALSE,
  EPS = FALSE,
  conf_file = "conf/ecd-fit-conf.yml",
  eps_file = NULL,
  qa.fit = FALSE
)
```

Arguments

<code>symbol</code>	Character. The symbol of sample data. Default: dji.
<code>iter</code>	A length-one numeric. Number of maximum iterations. Default: 1000.
<code>FIT</code>	Logical, indicating whether to call linear regression, default = FALSE
<code>EPS</code>	Logical, indicating whether to save the plot to EPS, default = FALSE
<code>conf_file</code>	File name for symbol config, default to conf/ecd-fit-conf.yml
<code>eps_file</code>	File name for eps output
<code>qa.fit</code>	Logical, qa the standardfit_fn once.

Value

Final ecd object

Examples

```
## Not run:
dji <- ecd.fit_data("dji", FIT=T)

## End(Not run)
```

 ecd.fit_ts_conf *Timeseries fitting utility*

Description

Fitting timeseries with provided conf as starting set of parameters.

Usage

```
ecd.fit_ts_conf(
  ts,
  conf,
  iter = 1000,
  FIT = FALSE,
  EPS = FALSE,
  eps_file = NULL,
  qa.fit = FALSE
)
```

Arguments

ts	An xts object from either ecd.data or ecd.df2ts.
conf	A nested list object, the configuration.
iter	A length-one numeric. Number of maximum iterations. Default: 1000.
FIT	Logical, indicating whether to call linear regression, default = FALSE
EPS	Logical, indicating whether to save the plot to EPS, default = FALSE
eps_file	File name for eps output
qa.fit	Logical, qa the standardfit_fn once.

Value

Final ecd object

Examples

```
## Not run:
d <- ecd.fit_ts_conf(ts, conf)

## End(Not run)
```

<code>ecd.has_quantile</code>	<i>Whether the ecd object has quantile data or not</i>
-------------------------------	--

Description

Whether the ecd object has quantile data or not. This is mostly for internal use.

Usage

```
ecd.has_quantile(object)
```

Arguments

<code>object</code>	an object of ecd class
---------------------	------------------------

Value

logical, whether the object has quantile data or not.

Author(s)

Stephen H-T. Lihn

<code>ecd.imgf</code>	<i>Incomplete MGF of ecd</i>
-----------------------	------------------------------

Description

Incomplete moment generating function (IMGF) of ecd, integration of $e^z P(z)$ for z from x to Inf. `ecd.mu_D` is simply a wrapper around MGF.

Usage

```
ecd.imgf(
  object,
  x = -Inf,
  t = 1,
  minus1 = FALSE,
  unit.sigma = FALSE,
  n.sigma = .ecd.mpfr.N.sigma,
  verbose = FALSE
)
ecd.mu_D(object)
```

Arguments

object	an object of ecd class
x	a numeric vector of x, default to -Inf
t	a numeric value for MGF, default to 1
minus1	logical, subtracting one from e^{tx}
unit.sigma	logical, transforming to unit sigma to achieve greater stability. Due to the instability of quadpack for ecd.integrate_pdf, default to TRUE. But constructing a new ecd object has significant overhead, be aware of it in performance sensitive program.
n.sigma	length-one numeric, specifying the max number of sigma to check for truncation.
verbose	logical, display timing information, for debugging purpose.

Value

The IMGF

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(0, 0, sigma=0.01)
x <- seq(0, 1, by=0.1)
ecd.imgf(d, x)
```

ecd.integrate

Wrapper to integrate numeric and mpfr

Description

The wrapper handles chooses to to use `integrate` for numeric; or to use `integrateR` for `mpfr`. Since the later doesn't allow infinity, there is a special handling to replace infinity with a large multiple of `sigma`.

Usage

```
ecd.integrate(
  object,
  f,
  lower,
  upper,
  ...,
  abs.tol = .Machine$double.eps^0.25,
  mpfr.qagi = TRUE,
  show.warning = TRUE
)
```

Arguments

object	An object of ecd class. This object can be bare-boned.
f	An R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
lower	Numeric, the lower limit of integration. Can be infinite.
upper	Numeric, the upper limit of integration. Can be infinite.
...	Addtional arguments for f.
abs.tol	numeric, the suggested absolute tolerance.
mpfr.qagi	logical, to use quadpack qagi transformation for infinity.
show.warning	logical, to suppress warnings or not.

Value

The integrate object

Author(s)

Stephen H. Lihn

ecd.lag

Utility to shift a vector of numeric or mpfr

Description

This utility is basically the same as `Hmisc::Lag`, but it handles mpfr vector properly.

Usage

```
ecd.lag(x, shift = 1, na.omit = FALSE)
```

Arguments

x	a vector of numeric or mpfr
shift	integer, cells to shift
na.omit	logical, whether to remove the NAs

Value

the shifted vector

Examples

```
x <- ecd.lag(c(1,2,3))
y <- ecd.lag(ecd.mpfr(c(1,2,3)))
```

`ecd.manage_hist_tails` *Manage histogram tails*

Description

Manage histogram tails to remove very far outliers. `histuple` is `list(hx = hist$mids, hy = hist$counts)`, which is an internal representation of histogram

Usage

```
ecd.manage_hist_tails(htu, merge_tails = c(0, 0))
```

Arguments

<code>htu</code>	list, input <code>histtuple</code>
<code>merge_tails</code>	length-two numeric vector, points to be merged for left and right tails

Value

list, `histtuple`

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
htu2 <- ecd.manage_hist_tails(htu, c(1,2))

## End(Not run)
```

`ecd.max_kurtosis` *Utility to calculate where the maximum kurtosis is on the positive j=0 line*

Description

This utility calculates the kurtosis for alpha from 2.85 to 3.00. Then the location and value of maximum kurtosis is presented.

Usage

```
ecd.max_kurtosis(jinv = 0)
```

Arguments

<code>jinv</code>	specify 0 (default) or 1728.
-------------------	------------------------------

Value

numeric vector, in which the first element is alpha, and the second element is the maximum kurtosis.

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:  
k <- ecd.max_kurtosis()  
alpha <- k[1]  
kurtosis <- k[2]  
  
## End(Not run)
```

ecd.mp2f

Wrapper to convert mpfr to numeric

Description

Convert mpfr to numeric primarily for display messages.

Usage

```
ecd.mp2f(x)
```

Arguments

x an object of `mpfr` class. If x is numeric class, it will be passed through.

Value

a numeric vector

Examples

```
x <- ecd.mp2f(ecd.mpfr(c(1,2,3)))
```

ecd.mpfr*Wrapper to convert numeric to mpfr***Description**

Convert numeric to mpfr for ecd calculations. `ecd.mp1` is the constant 1 wrapped in mpfr class. `ecd.mppi` is the function to obtain pi from Rmpfr with an optional precision. This is used to implement `ecd.erfq`. `ecd.gamma` is a wrapper on `ecl1d.gamma`, which is the incomplete gamma function. `ecd.erf` is a wrapper on `Rmpfr::erf`. `ecd.erfcx` is a wrapper on `Rmpfr::erfcx`. `ecd.erfc` is a wrapper on `Rmpfr::erfc`. This is used to implement `ecd.erfq`. `ecd.dawson` is a wrapper on `gsl::dawson`. Dawson function is used to implement `ecd.erfq`. `ecd.erfi` is the imaginary scaled error function, which is implemented through `ecd.dawson`. `ecd.devel` is a developer tool to size down intensive mpfr tests for CRAN. Set `ecd_devel` in R options or OS env to change its value.

Usage

```
ecd.mpfr(x, precBits = getOption("ecd.precBits"))

ecd.mp1

ecd.mppi(precBits = getOption("ecd.precBits"))

ecd.gamma(s, x, na.stop = TRUE)

ecd.erf(x)

ecd.erfc(x)

ecd.erfcx(x)

ecd.dawson(x)

ecd.erfi(x)

ecd.devel()
```

Arguments

- `x` a numeric vector or list. If `x` is mpfr class, it will be passed through.
- `precBits` an integer for mpfr precBits. Default is from `getOption("ecd.precBits")`.
- `s` numeric vector, for the order of incomplete gamma function
- `na.stop` logical, stop if NaN is generated. The default is TRUE.

Format

An object of class `mpfr` of length 1.

Value

The `mpfr` object

Examples

```
x <- ecd.mpfr(1)
y <- ecd.mpfr(c(1,2,3))
z <- ecd.mp1
p <- ecd.mppi()
```

`ecd.mpfr_qagi`

Utility to integrate mpfr with infinity via qagi

Description

This utility supplements `Rmpfr::integrateR` with the quadpack `qagi` method to handle integration involving infinity. `Qagi` is a transformation of $x/\sigma = (1-t)/t$ for positive x , and $x/\sigma = (t-1)/t$ for negative x . $t = 0$ is represented by `.Machine$double.eps`. This utility requires (a) lower or upper is `+/-Inf`; (b) lower and upper are of the same sign.

Usage

```
ecd.mpfr_qagi(
  object,
  f,
  lower,
  upper,
  ...,
  abs.tol = .Machine$double.eps^0.25,
  show.warning = TRUE
)
```

Arguments

<code>object</code>	an object of <code>ecd</code> class
<code>f</code>	an R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
<code>lower</code>	numeric, the lower limit of integration. Can be infinite.
<code>upper</code>	numeric, the upper limit of integration. Can be infinite.
<code>...</code>	additional arguments for <code>f</code> .
<code>abs.tol</code>	numeric, the suggested absolute tolerance.
<code>show.warning</code>	logical, to suppress warnings or not.

Value

The `integrate` object

Author(s)

Stephen H. Lihn

ecd.mpnum	<i>Wrappers for ecd to maintain consistent type between mpfr and numeric</i>
-----------	--

Description

Primarily to make sure x is converted to mpfr vector if it is not, when use.mpfr is set.

Usage

```
ecd.mpnum(object, x)
ecd.ifelse(object, test, yes, no)
ecd.sapply(object, x, FUN, ...)
ecd.mcsapply(object, x, FUN, ...)
```

Arguments

object	an object of ecd class
x	a vector of numeric or mpfr.
test	logical, test of ifelse.
yes	return values for true elements of test
no	return values for false elements of test
FUN	the function to be applied to each element of x
...	optional arguments to FUN

Value

a numeric or mpfr vector

Author(s)

Stephen H. Lihn

ecd.ogf	<i>Option generating function of ecd</i>
---------	--

Description

Option generating function (OGF) of ecd. For call, it is integration of $(e^z - e^k)P(z)$ for z from k to Inf. For put, it is integration of $(e^k - e^z)P(z)$ for z from -Inf to k.

Usage

```
ecd.ogf(object, k, otype = "c", unit.sigma = FALSE, verbose = FALSE)
```

Arguments

object	an object of ecd class
k	a numeric vector of log-strike
otype	character, specifying option type: c or p.
unit.sigma	logical, transforming to unit sigma to achieve greater stability.
verbose	logical, display timing information, for debugging purpose.

Value

The option price normalized by underlying

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(0, 0, sigma=0.01)
k <- seq(-0.1, 0.1, by=0.01)
ecd.ogf(d, k, "c")
```

ecd.pdf

Calculate the PDF of an ecd object

Description

Calculate the PDF of an ecd object

Usage

```
ecd.pdf(object, x)
```

Arguments

object	an object of ecd class
x	numeric vector of <i>x</i> dimension

Value

numeric vector of the PDF

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd()
x <- seq(-10, 10, by=1)
ecd.pdf(d,x)
```

ecd.polar*Polar constructor of ecd class***Description**

Construct an ecd class by specifying R and theta. They are converted to alpha and gamma, then passed onto the ecd constructor.

Usage

```
ecd.polar(
  R = NaN,
  theta = NaN,
  sigma = 1,
  beta = 0,
  mu = 0,
  cusp = 0,
  with.stats = TRUE,
  with.quantile = FALSE,
  bare.bone = FALSE,
  verbose = FALSE
)
```

Arguments

R	numeric, the radius parameter. Default is NaN.
theta	numeric, the angle parameter. Default: NaN.
sigma	numeric, the scale parameter. Must be positive. Default: 1.
beta	numeric, the skewness parameter. Default: 0.
mu	numeric, the location parameter. Default: 0.
cusp	logical, indicate type of cusp (0,1,2).
with.stats	logical, also calculate statistics, default is TRUE.
with.quantile	logical, also calculate quantile data, default is FALSE.
bare.bone	logical, skip both const and stats calculation, default is FALSE. This for debug purpose for issues on integrating $e^y(x)$.
verbose	logical, display timing information, for debugging purpose, default is FALSE.

Value

The ecd class

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd.polar(R=1, theta=0.5*pi)
```

<code>ecd.rational</code>	<i>Utility to convert a numeric to a rational</i>
---------------------------	---

Description

Convert a numeric x to rational p/q, which is then used for polynomial construction. It can be used for displaying the time as fraction of a year too.

Usage

```
ecd.rational(
  x,
  pref.denominator = numeric(0),
  cycles = 10,
  max.denominator = 500,
  as.character = FALSE
)
```

Arguments

<code>x</code>	numeric
<code>pref.denominator</code>	numeric, a list of preferred integer denominators to conform to, default is <code>numeric(0)</code> .
<code>cycles</code>	numeric, maximum number of steps, default is 10.
<code>max.denominator</code>	numeric, maximum denominator when the loop of trial should stop, default is 500.
<code>as.character</code>	logical, if specified, convert to character of p/q, default is FALSE.

Value

vector of two integers, representing numerator and denominator. If `as.character` is true, then return character instead of the rational pair. If `x` is a vector and `as.character` is false, return a matrix of length(`x`) by 2.

Examples

```
 pq1 <- ecd.rational(2.5)
 pq2 <- ecd.rational(1/250)
```

<code>ecd.read_csv_by_symbol</code>	<i>Read csv file of sample data</i>
-------------------------------------	-------------------------------------

Description

This is a helper utility to read sample csv file into data frame. The main use for external users is to read the option data since it has a different format than other price timeseries data.

Usage

```
ecd.read_csv_by_symbol(symbol = "dji", extdata_dir = NULL)
```

Arguments

symbol	Character for the symbol of the time series. Default: dji
extdata_dir	optionally specify user's own extdata folder

Value

The data.frame object

Author(s)

Stephen H-T. Lihn

Examples

```
dji <- ecd.read_csv_by_symbol("dji")
spx <- ecd.read_csv_by_symbol("spxoption2")
```

ecd.read_symbol_conf	<i>Read conf for sample data</i>
----------------------	----------------------------------

Description

Read conf for sample data

Usage

```
ecd.read_symbol_conf(symbol, conf_file = "conf/ecd-fit-conf.yml")
```

Arguments

symbol	Character. The symbol of sample data. Default: dji.
conf_file	File name for symbol config, default to conf/ecd-fit-conf.yml

Value

the conf object

Examples

```
## Not run:
conf <- ecd.read_symbol_conf("dji")

## End(Not run)
```

ecd.sd	<i>Standard deviation, variance, mean, skewness, and kurtosis of ecd</i>
--------	--

Description

Convenience wrappers around ecd's stats data

Usage

```
ecd.sd(object)  
ecd.var(object)  
ecd.mean(object)  
ecd.skewness(object)  
ecd.kurt(object)  
ecd.kurtosis(object)
```

Arguments

object an object of ecd class

Value

numeric or mpfr

Examples

```
d <- ecd(-1,1)  
ecd.sd(d)  
ecd.var(d)  
ecd.mean(d)  
ecd.skewness(d)  
ecd.kurt(d)
```

ecd.setup_const	<i>Integration preprocessor for an ecd object</i>
-----------------	---

Description

This is an internal helper function for ecd constructor Its main function is to determine const, const_left_x, and const_right_x during object construction.

Usage

```
ecd.setup_const(object, verbose = FALSE)
```

Arguments

- object An object of ecd class
 verbose logical, display timing information, for debugging purpose.

Value

```
list(const, const_left_x, const_right_x)
```

Author(s)

Stephen H. Lihn

Examples

```
ecd.toString(ecd(-1,1, sigma=0.1))
```

ecd.solve_cusp_asym	<i>Trigonometric solution for asymmetric cusp distribution</i>
---------------------	--

Description

The simplified trigonometric solution for $x^2 = -y^3 - \text{beta} * x * y$

Usage

```
ecd.solve_cusp_asym(x, beta)
```

Arguments

- x Array of x dimension
 beta the skew parameter

Value

Array of y

Examples

```
x <- seq(-100,100,by=0.1)
y <- ecd.solve_cusp_asym(x, beta=0.5)
```

ecd.stats	<i>Compute statistics of an ecd object</i>
-----------	--

Description

Compute statistics for m1, m2, m3, m4, mean, var, skewness, kurtosis. This is used as part of ecd constructor.

Usage

```
ecd.stats(object, asymp.q = NULL, verbose = FALSE)
```

Arguments

object	an object of ecd class
asymp.q	If specified, a length-one numeric as asymptotic quantile for the asymptotic statistics. There is a wrapper in ecd.asymp.stats
verbose	logical, display timing information, for debugging purpose.

Value

a list of m1, m2, m3, m4, mean, var, skewness, kurtosis

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(1,1)
ecd.stats(d)
```

ecd.toString	<i>String representation of ecd</i>
--------------	-------------------------------------

Description

A string representation of an ecd object. Can be used for warning or error.

Usage

```
ecd.toString(object, full = FALSE)
```

Arguments

object	An object of ecd class
full	logical, indicating if long form (multiple lines) should be rendered.

Value

character

Examples

```
ecd.toString(ecd(-1,1, sigma=0.1))
```

ecd.ts_lag_stats

Lag statistics on timeseries of log returns

Description

Lag statistics on log returns are added to the xts attributes. It takes a vector of lags and calculates the mean, stdev, var, skewness, and kurtosis for cumulative log returns of each lag. The data is stored as a list of vectors under lagstats attribute. Be aware this function uses multicore lapply.

Usage

```
ecd.ts_lag_stats(ts = "dji", lags, absolute = FALSE)
```

Arguments

- | | |
|----------|--|
| ts | the xts object from sample data. The ts must have the logr column. If a string is given, it will be replaced with sample data of the symbol. |
| lags | a numeric vector of integers greater than 0. |
| absolute | logical, if TRUE, statistics calculated on absolute log returns. Default: FALSE. |

Value

The xts object containing lagstats attribute

Examples

```
## Not run:
dji <- ecd.ts_lag_stats(ecd.data("dji"), 2)

## End(Not run)
```

<code>ecd.uniroot</code>	<i>Uniroot wrapper</i>
--------------------------	------------------------

Description

This function wraps ordinary uniroot and unirootR (from Rmpfr) to the same interface.

Usage

```
ecd.uniroot(
  f,
  lower,
  upper,
  use.mpfr = FALSE,
  tol = .Machine$double.eps^0.25,
  maxiter = 100
)
```

Arguments

<code>f</code>	the function for which the root is sought.
<code>lower</code> , <code>upper</code>	the lower and upper end points of the interval to be searched.
<code>use.mpfr</code>	logical, to use MPFR (default), or else uniroot in stats.
<code>tol</code>	the desired accuracy (convergence tolerance).
<code>maxiter</code>	the maximum number of iterations.

Value

uniroot result

Author(s)

Stephen H. Lihn

<code>ecd.y0_isomorphic</code>	<i>The analytic solution of $y(0)$ via isomorphic mapping.</i>
--------------------------------	---

Description

This utility can be called two ways: (a) specify R and theta; (b) provide the ecd object. But not at the same time.

Usage

```
ecd.y0_isomorphic(theta = NaN, R = 1, object = NULL)
```

Arguments

theta	numeric vector, the polar coordinate
R	numeric vector, the polar coordinate
object	optionally, a single ecd object

Value

the value of $y(0)$

Examples

```
t <- 45/180*pi
ecd.y0_isomorphic(t)
```

ecdattr

Constructor of ecdattr class for the Elliptic Database (ECDB)

Description

Construct an ecdattr class by providing the required parameters. This object has one-to-one correspondence to the rows in ECDATTR table. This is used primarily as object wrapper for safe update to ECDB.

Usage

```
ecdattr(alpha, gamma = NaN, cusp = 0, use.mpfr = FALSE)
```

Arguments

alpha	numeric, must be an integer after multiplied by 1000000.
gamma	numeric, must be an integer after multiplied by 1000000. NaN if cusp is 1.
cusp	numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
use.mpfr	logical, whether to use mpfr for ecd object, default is FALSE.

Value

an object of ecdattr class

Examples

```
a <- ecdattr(1,1)
b <- ecdattr(alpha=1, cusp=1)
```

ecdattr-class	<i>An S4 class to represent the ecdattr row in the Elliptic Database (ECDB)</i>
---------------	---

Description

The ecdattr class serves as an object-oriented interface between R and ECDB. This class is used extensively during the `bootstrap` process. A list of light-weight ecdattr objects is created first by `ecdattr.pairs` function, then the `ecdattr.enrich` function is invoked in parallel to calculate additional ecd attributes.

Slots

- `call` the match.call slot
- `alpha` numeric
- `gamma` numeric. When cusp is 1, gamma is derived.
- `cusp` numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
- `use.mpfr` logical, whether to use mpfr for ecd object.
- `enriched` logical. If TRUE, it indicates the object has been enriched with ecd attributes.
- `alpha_m` numeric, $\alpha * 1000000$.
- `gamma_m` numeric, $\gamma * 1000000$.
- `ecd` an object of ecd class.
- `attr` list of attributes. They are NULL protected for SQLite.

ecdattr.enrich	<i>Enrich a basic ecdattr object</i>
----------------	--------------------------------------

Description

It takes a basic ecdattr object, enrich it with ecd attributes. This function is computationally heavy. So the objects are often wrapped in a list and computed via `parallel::mclapply`.

Usage

```
ecdattr.enrich(p)
```

Arguments

<code>p</code>	a basic ecdattr object
----------------	------------------------

Value

an enriched ecdattr object	
----------------------------	--

`ecdattr.pairs` *Create a list of basic ecdattr objects*

Description

The list is created by the Cartesian product between alpha and gamma. This contains the data points of a rectangular area defined by alpha, gamma. If cusp is 1, data points are on the critical line specified by alpha.

Usage

```
ecdattr.pairs(alpha, gamma, cusp = 0, use.mpfr = FALSE)
```

Arguments

<code>alpha, gamma</code>	numeric vectors
<code>cusp</code>	numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
<code>use.mpfr</code>	logical, whether to use mpfr for ecd object, default is FALSE.

Value

a list of basic ecdattr objects.

`ecdattr.pairs_polar` *Create a list of basic ecdattr objects in polar coordinate*

Description

The list is created by the Cartesian product between R and theta. This contains the data points of a circular area defined by R, theta. If cusp is 1, data points are on the critical line specified by R.

Usage

```
ecdattr.pairs_polar(R, theta, cusp = 0, use.mpfr = FALSE)
```

Arguments

<code>R, theta</code>	numeric vectors
<code>cusp</code>	numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
<code>use.mpfr</code>	logical, whether to use mpfr for ecd object, default is FALSE.

Value

a list of basic ecdattr objects.

ecdb*Constructor of ecdb class for the elliptic database*

Description

Construct an ecdb class by providing the required parameters. The default is to use the internal database location. But the internal db is limited in size. The the elliptic database stores the stdev, kurtosis, discriminant, j-invariant, and ellipticity. for alpha and gamma between -100 and 100. Step size is 1 for -100 to 100; 0.25 for -50 to 50; 0.1 for -10 to 10; 0.025 between -6 and 1. Speical lines with step size of 0.001 for j0 and j1728 between -10 and 10; 0.01 for kmax and critical between 0 and 100. For asym1X, step size is 10 from 100 to 1000. For asym2X, step size is 100 from 1000 to 10000. For asym3X, step size is 1000 from 10000 to 60000. For polar-q1, step size is 0.025 from 0 to 20 for log2(R), and integer angles, 0-89.

Usage

```
ecdb(file = NULL, newdb = FALSE)
```

Arguments

file	Character, the full path to an elliptic database. Use "internal" to force the usage of the internal db.
newdb	Logical. If TRUE, remove existing db and create a new one. Default: FALSE.

Value

An object of ecdb class

Examples

```
db <- ecdb("internal")
```

ecdb-class

setClass for ecdb class

Description

setClass for ecdb class

Slots

call the match.call slot
 file character, the full path to an elliptic database.
 conn an object of SQLiteConnection class.
 is.internal logical, whether the connected db is internal.
 conf list of configuration for data generation assigned by the constructor. Typical user should not have to modify this list unless you need to generate more data for advanced research.

Author(s)

Stephen H-T. Lihn

ecdb.dbSendQuery *Send query to the elliptic database*

Description

This API is used for write operations such as CREATE and INSERT.

Usage

```
ecdb.dbSendQuery(db, statement, ...)
```

Arguments

db	an object of ecdb class
statement	character, the SQL statement
...	database-specific parameters may be specified here

Value

a result set object

Author(s)

Stephen H-T. Lihn

ecdb.protectiveCommit *Protective commit*

Description

Protective commit after sending query to the elliptic database.

Usage

```
ecdb.protectiveCommit(db)
```

Arguments

db	an object of ecdb class
----	-------------------------

Value

The db object

Author(s)

Stephen H-T. Lihn

<code>ecdq</code>	<i>Constructor of ecdq class</i>
-------------------	----------------------------------

Description

Construct an ecdq class by providing the required parameters.

Usage

```
ecdq(ecd, verbose = FALSE)
```

Arguments

<code>ecd</code>	An object of ecd class
<code>verbose</code>	logical, display timing information, for debugging purpose.

Value

An object of ecdq class

Author(s)

Stephen H. Linh

Examples

```
## Not run:
d <- ecd()
dq <- ecdq(d)

## End(Not run)
```

<code>ecdq-class</code>	<i>setClass for ecdq class</i>
-------------------------	--------------------------------

Description

setClass for ecdq class, the quantile generator

Slots

- `call` the match.call slot
- `xseg.from,xseg.to` numeric vectors. The from and to for each x segment.
- `cseg.from,cseg.to` numeric vectors. The from and to for each cdf segment.
- `cseg.min,cseg.max` numeric. The min and max of cdf segments.
- `N_seg` numeric. Number of segments.
- `cdf.fit` A vector of lm object, one for each segment.
- `x_left_tail,x_right_tail` numeric. The starting x of left and right tails.
- `fit.left,fit.right` objects of lm class for fitting the tails.
- `conf` list of miscellaneous configurations. For debugging purpose.

<code>ecld</code>	<i>Constructor of ecld class</i>
-------------------	----------------------------------

Description

Construct an `ecld-class` by providing the required parameters. The default is the standard symmetric cusp distribution. The default also doesn't calculate any ecd extension. `ecld.from` allows you to pass the parameters from an existing ecd object. `ecld.validate` checks if an object is ecld class. `ecld.quartic` is a convenient constructor designed for quartic distribution. `ecld.from_sd` calculates sigma from a given sd and renders a vanilla ecld object.

Usage

```
ecld(
  lambda = 3,
  sigma = 1,
  beta = 0,
  mu = 0,
  epsilon = NaN,
  rho = NaN,
  with.ecd = FALSE,
  with.mu_D = FALSE,
  with.RN = FALSE,
  is.sged = FALSE,
  verbose = FALSE
)

ecld.from(
  object,
  with.ecd = FALSE,
  with.mu_D = FALSE,
  with.RN = FALSE,
  verbose = FALSE
)

ecld.validate(object, sged.allowed = FALSE, sged.only = FALSE)

ecld.quartic(sigma, epsilon, rho, mu_plus_ratio = NaN, mu_plus = NaN)

ecld.from_sd(lambda = 3, sd = 1, beta = 0, mu = 0)
```

Arguments

<code>lambda</code>	numeric, the lambda parameter. Must be positive. Default: 3.
<code>sigma</code>	numeric, the scale parameter. Must be positive. Default: 1.
<code>beta</code>	numeric, the skewness parameter. Default: 0.
<code>mu</code>	numeric, the location parameter. Default: 0.
<code>epsilon</code>	The supplemental residual premium for lambda transformation. It is default to NaN in ecld constructor since its meaning is not defined.

rho	The supplemental momentum shift for lambda transformation. It is default to NaN in ecld constructor since its meaning is not defined.
with.ecd	logical, also calculate the ecd object, default is FALSE.
with.mu_D	logical, also calculate the ecd risk-neutral drift, default is FALSE. If TRUE, this flag supercedes with.ecd. Also mu must set to zero.
with.RN	logical, also calculate the risk-neutral ecd object, default is FALSE. If TRUE, this flag supercedes with.mu_D.
is.sged	logical, if TRUE, interpret parameters as SGED.
verbose	logical, display timing information, for debugging purpose, default is FALSE.
object	an object of ecld class
sged.allowed	logical, used in ecld.validate to indicate if the function allows SGED.
sged.only	logical, used in ecld.validate to indicate if the function is only for SGED.
mu_plus, mu_plus_ratio	numeric, excess value in addition to mu_D. When ratio is provided, it is relative to the stdev.
sd	numeric, the scale parameter expressed in stdev instead of sigma. Internally, It is converted to sigma via uniroot on ecld.sd. Must be positive. Default: 1.

Value

an object of ecld class

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld()
ld <- ecld(2, 0.01)
ld <- ecld.from_sd(3, 0.1)
```

Description

The ecld class serves as an object-oriented interface for the lambda distribution. The ecld prefix is also used as the namespace for many analytic formulai derived in lambda distribution, especially when lambda = 1,2,3. Because of the extensive use of analytic formulai and enhanced precision through the unit distribution, MPFR is not needed in most cases. This makes option pricing calculation in ecld much faster than its counterpart built on the more general-purpose ecd library.

Slots

call the match.call slot
 lambda numeric
 sigma numeric
 beta numeric
 mu numeric
 use.mpfr logical, whether to use mpfr for ecl^d object. If any of the above parameters is mpfr, then this flag is set to TRUE.
 is.sged logical, if TRUE, interpret parameters as SGED.
 ecd the companion object of ecd class (optional)
 mu_D the risk-neutral drift, optional, but preferred to have value if the object is to engage with OGF calculation.
 epsilon the residual risk, optional as a storage for lambda transformation
 rho the momentum shift, optional as a storage for lambda transformation
 ecd_RN the risk-neutral companion object of ecd class (optional)
 status numeric, bitmap recording the state of the calculation layers. 1: bare bone; 2: ecd; 4: mu_D; 8: ecd_RN

Details

The lambda distribution is defined by a depressed polynomial of λ -th order,

$$|y(z)|^\lambda + \dots - \beta z y(z) = z^2$$

where $y(z)$ must approach minus infinity as z approaches plus or minus infinity. The density function is defined as

$$P(x; \lambda, \sigma, \beta, \mu) \equiv \frac{1}{C\sigma} e^{y(|\frac{x-\mu}{\sigma}|)},$$

and C is the normalization constant,

$$C = \int_{-\infty}^{\infty} e^{y(z)} dz,$$

where λ is the shape parameter, σ is the scale parameter, β is the asymmetric parameter, μ is the location parameter.

The distribution is symmetric when $\beta = 0$, which becomes

$$P(x; \lambda, \sigma, \mu) \equiv \frac{1}{\lambda \Gamma(\frac{2}{\lambda}) \sigma} e^{-|\frac{x-\mu}{\sigma}|^{\frac{2}{\lambda}}}.$$

This functional form is not unfamiliar and has appeared under several names, such as generalized normal distribution and power exponential distribution, where $\lambda < 2$.

However, we are most interested in $\lambda \geq 2$, which is called the "local regime". In this regime, the MGF diverges which requires regularization aka truncation of the right tail. The λ option model pays special attention to $\lambda = 2, 3, 4$ where many closed form solutions can be obtained. In particular, SPX options fit best at $\lambda = 4$, which is called "quartic lambda".

Since option model often has to deal with very small numbers which are closed to the machine error of double precision calculation, the method supports MPFR. As soon as one of the ecl^d parameters becomes MPFR (by simply multiplying ecd.mp1), the subsequent calculations will use MPFR.

Author(s)

Stephen H. Lihn

References

For lambda distribution and option pricing model, see Stephen Lihn (2015). *The Special Elliptic Option Pricing Model and Volatility Smile*. SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2707810.

Closed form solutions are derived in Stephen Lihn (2016). *Closed Form Solution and Term Structure for SPX Options*. SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2805769 and

Stephen Lihn (2017). *From Volatility Smile to Risk Neutral Probability and Closed Form Solution of Local Volatility Function*. SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2906522

eclt.cdf

CDF and CCDF of eclt

Description

The analytic solutions for CDF and CCDF of eclt, if available. eclt.cdf_gamma is a sub-module with the CDF expressed as incomplete gamma function. SGED is supported only in eclt.cdf and eclt.ccdf.

Usage

```
eclt.cdf(object, x)
eclt.ccdf(object, x)
eclt.cdf_integrate(object, x)
eclt.cdf_gamma(object, x)
```

Arguments

object	an object of eclt class
x	a numeric vector of x

Value

The CDF or CCDF vector

Author(s)

Stephen H. Lihn

Examples

```
ld <- eclt(sigma=0.01*ecd.mp1)
x <- seq(-0.1, 0.1, by=0.01)
eclt.cdf(ld,x)
```

`ecld.const`*Analytic solution of the normalization constant for lambda distribution***Description**

The normalization constant C . SGED is supported.

Usage

```
ecld.const(object)
```

Arguments

<code>object</code>	an object of <code>ecld</code> class
---------------------	--------------------------------------

Value

`numeric`

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(3)
ecld.const(ld)
```

`ecld.fixed_point_SN0_atm_ki`*The ATM RNO related constants and calculations in fixed point model***Description**

Computes the small sigma limit of ATM location, rho/stdev, ATM skew of Q_c , and the ratio of lambda to ATM skew under the RNO measure in the fixed point model.

Usage

```
ecld.fixed_point_SN0_atm_ki(lambda)
ecld.fixed_point_SN0_rho_sd(lambda)
ecld.fixed_point_SN0_atm_ki_sd()
ecld.fixed_point_SN0_skew(lambda, atm_ki = NULL)
ecld.fixed_point_SN0_lambda_skew_ratio(lambda, atm_ki = NULL)
```

Arguments

lambda	numeric the lambda parameter.
atm_ki	numeric optional and experimental, use it as override. This is for experimental purpose, default is NULL. A typical override is the sd/sigma.

Value

numeric

Author(s)

Stephen H-T. Lihn

ecld.gamma

*Incomplete gamma function and asymptotic expansion***Description**

ecld.gamma is the wrapper for incomplete gamma function $\Gamma(s, x)$. It is mainly to wrap around pgamma. And ecld.gamma_hgeo is the asymptotic expansion of $\Gamma(s, x)$ using hypergeometric series, $e^{-x} x^{s-1} {}_2F_0(1, 1-s; ; -1/x)$. It is mainly used in for star OGF $L^*(k; \lambda)$. ecld.gamma_2F0 is simply ${}_2F_0(1, 1-s; ; -1/x)$, which is used in the star OGF expansion.

Usage

```
ecld.gamma(s, x = 0, na.stop = TRUE)
ecld.gamma_hgeo(s, x, order)
ecld.gamma_2F0(s, x, order)
```

Arguments

s	numeric vector, for the order of incomplete gamma function
x	numeric or MPFR vector
na.stop	logical, stop if NaN is generated. The default is TRUE.
order	numeric, the order of the power series

Value

numeric

Author(s)

Stephen H-T. Lihn

ecld.imgf*Incomplete moment generating function (IMGF) of ecld***Description**

The analytic solutions for IMGF of ecld, if available. Note that, by default, risk neutrality is honored. However, you must note that when fitting market data, this is usually not true. SGED is supported.

Usage

```
ecld.imgf(object, k, otype = "c", RN = TRUE)
ecld.imgf_quartic(object, k, otype = "c", RN = TRUE)
ecld.imgf_gamma(object, k, otype = "c", RN = TRUE)
ecld.imgf_integrate(object, k, otype = "c", RN = TRUE)
```

Arguments

object	an object of ecld class
k	a numeric vector of log-strike
otype	character, specifying option type: c (default) or p.
RN	logical, use risk-neutral assumption for mu_D

Value

numeric, incomplete MGF

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01)
ecld.imgf(ld, 0)
```

ecld.imnt*Incomplete moment (imnt) of ecld*

Description

The analytic solutions for imnt of ecld, if available. Note that, by default, risk neutrality is honored. `ecld.imnt_sum` provides an alternative method to calculate IMGF.

Usage

```
ecld.imnt(object, ki, order, otype = "c")
ecld.imnt_integrate(object, ki, order, otype = "c")
ecld.imnt_sum(object, ki, order, otype = "c")
```

Arguments

object	an object of ecld class
ki	numeric vector of normalized log-strike, $(k-\mu)/\sigma$
order	numeric. Order of the moment to be computed. For <code>ecld.imnt_sum</code> , this is the maximum order to be truncated. For small sigma at lambda=3, this can be simply 2. If Inf, the slope truncation procedure will be used to determine the maximum order. However, due to the numeric limit of pgamma, it is capped at 100.
otype	character, specifying option type: c (default) or p.

Value

numeric vector

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
ki <- seq(-0.1, 0.1, by=0.01)
ecld.imnt(ld,ki, 1)
```

ecld.ivol_ogf_star *Calculate implied volatility using star OGF and small sigma formula*

Description

Calculate implied volatility using star OGF and small sigma formula. SGED is not supported yet.

Usage

```
ecld.ivol_ogf_star(
  object,
  ki,
  epsilon = 0,
  otype = "c",
  order.local = Inf,
  order.global = Inf,
  ignore.mu = FALSE
)
```

Arguments

object	an object of ecld class
ki	a numeric vector of log-strike
epsilon	numeric, small asymptotic premium added to local regime
otype	option type
order.local	numeric, order of the hypergeometric series to be computed for local regime. Default is Inf, use the incomplete gamma. When it is NaN, L* value is suppressed.
order.global	numeric, order of the hypergeometric series to be computed for global regime. Default is Inf, use the incomplete gamma. If NaN, then revert to OGF.
ignore.mu	logical, ignore exp(mu) on both sides, default is FALSE.

Value

The state price of option in star OGF terms. For `ecld.ivol_ogf_star`, it is σ_1 .

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.001)
ecld.ivol_ogf_star(ld, 0)
```

<code>ecld.mgf_term</code>	<i>The term structure of ecld symmetric MGF</i>
----------------------------	---

Description

`ecld.mgf_term` and `ecld.mgf_diterm` are the term and derivative of the term by order (n) in the summation of MGF. Since `ecld.mgf_term` uses `lgamma` instead of `gamma` itself, `ecld.mgf_term_original` is to preserve the original formula. `ecld.mgf_trunc` uses `ecld.mgf_diterm` to locate the truncation of MGF terms. `ecld.mgf_trunc_max_sigma` locates the maximum sigma that keeps MGF finite for each lambda. SGED is supported.

Usage

```
ecld.mgf_term(object, order, t = 1)

ecld.mgf_term_original(object, order, t = 1)

ecld.mgf_diterm(object, order, t = 1)

ecld.mgf_trunc(object, t = 1)

ecld.mgf_trunc_max_sigma(object, order = 1)
```

Arguments

<code>object</code>	an object of ecd class
<code>order</code>	numeric. Order of the term (moment). Order can be a vector.
<code>t</code>	numeric, for MGF

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(3, sigma=0.01*ecd.mp1)
ecld.mgf_trunc(ld)
```

ecld.moment*The moments and MGF of ecld***Description**

Compute the moments and MGF of ecld for `mu=0` (centered), via analytical result whenever is available. SGED is supported.

Usage

```
ecld.moment(object, order, ignore.mu = TRUE)
ecld.mgf(object, t = 1)
ecld.mgf_by_sum(object, t = 1)
ecld.mgf_quartic(object, t = 1)
```

Arguments

<code>object</code>	an object of ecd class
<code>order</code>	numeric, order of the moment to be computed
<code>ignore.mu</code>	logical, disregard mu; otherwise, stop if mu is not zero.
<code>t</code>	numeric, for MGF

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(lambda=3, sigma=0.01*ecd.mp1)
ecld.moment(ld, 2)
ecld.mgf(ld)
```

ecld.mpnum*Wrappers for ecld to maintain consistent type between mpfr and numeric***Description**

Primarily to make sure `x` is converted to mpfr vector if it is not, when `use.mpfr` is set.

Usage

```
ecld.mpnum(object, x)
ecld.ifelse(object, test, yes, no)
ecld.sapply(object, x, FUN, ...)
ecld.mclapply(object, x, FUN, ...)
```

Arguments

object	an object of ecld class
x	a vector of numeric or mpfr.
test	logical, test of ifelse.
yes	return values for true elements of test
no	return values for false elements of test
FUN	the function to be applied to each element of x
...	optional arguments to FUN

Value

a numeric or mpfr vector

Author(s)

Stephen H-T. Lihn

ecld.mu_D

mu_D of ecld

Description

The analytic solutions for risk-neutral drift. If analytic form doesn't exist, it uses integral of unit distribution. This is different from ecld.mgf where series summation is used.

Usage

```
ecld.mu_D(object, validate = TRUE)
ecld.mu_D_quartic(object)
ecld.mu_D_by_sum(object)
ecld.mu_D_integrate(object, validate = TRUE)
```

Arguments

object	an object of ecld class
validate	logical, if true (default), stop when the result is NaN or infinite.

Value

numeric

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
ecld.mu_D(ld)
```

ecld.ogf

Option generating function (OGF) of ecld

Description

The analytic solutions for OGF of ecld, if available. Note that, by default, risk neutrality is honored. However, you must note that when fitting market data, this is usually not true. It is also more preferable that input object already contains mu_D. It is more consistent and saves time.

Usage

```
ecld.ogf(object, k, otype = "c", RN = TRUE)

ecld.ogf_quartic(object, k, otype = "c", RN = TRUE)

ecld.ogf_integrate(object, k, otype = "c", RN = TRUE)

ecld.ogf_gamma(object, k, otype = "c", RN = TRUE)

ecld.ogf_imnt_sum(object, k, order, otype = "c", RN = TRUE)

ecld.ogf_log_slope(object, k, otype = "c", RN = TRUE)
```

Arguments

object	an object of ecld class
k	a numeric vector of log-strike
otype	character, specifying option type: c (default) or p.
RN	logical, use risk-neutral assumption for mu_D
order	numeric, order of the moment to be computed

Value

The state price of option

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
k <- seq(-0.1, 0.1, by=0.05)
ecld.ogf_star(ld,k)
```

ecld.ogf_star

Star OGF of ecld

Description

The star OGF of ecld is the limiting OGF for small sigma. It only depends on the normalized k and lambda. Its dependency on sigma and mu is removed. SGED is not supported yet.

Usage

```
ecld.ogf_star(object, ki)

ecld.ogf_star_hgeo(object, ki, order = 4)

ecld.ogf_star_exp(object, ki, order = 3)

ecld.ogf_star_gamma_star(object, ki, order = 6)

ecld.ogf_star_analytic(object, ki)
```

Arguments

object	an object of ecld class
ki	a numeric vector of log-strike
order	numeric, order of the hypergeometric series to be computed

Value

The state price of option in star OGF terms.

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.001*ecd.mp1)
ki <- seq(1, 5, by=1)
ecld.ogf_star(ld, ki)
```

ecld.op_Q*The Q operator in option pricing model*

Description

The Q operator generates the normalized implied volatility $\sigma_1(k)/\sigma$. ecld.op_Q_skew calculates the skew in Q space by ki and +/- dki/2. ecld.op_Q_skew_by_k_lm calculates the skew in Q space by lm on a vector of k. ki is derived internally from $(k-\mu-\rho)/\sigma$. ecld.fixed_point_atm_Q_left is the left hand side of fixed point ATM hypothesis. ecld.fixed_point_atm_Q_right is the right hand side of fixed point ATM hypothesis, assuming shift is stored in rho. ecld.fixed_point_atm_ki is the ATM ki in fixed point ATM hypothesis. assuming shift is stored in rho. ecld.fixed_point_shift is the utility for the standard shift algorithm, $-(\text{atm_imp_k} - \mu)$.

Usage

```
ecld.op_Q(object, ki, otype = "c")
ecld.op_Q_skew(object, ki, dki = 0.1, otype = "c")
ecld.op_Q_skew_by_k_lm(object, k, otype = "c")
ecld.fixed_point_atm_Q_left(object, otype = "c")
ecld.fixed_point_atm_ki(object)
ecld.fixed_point_atm_Q_right(object)
ecld.fixed_point_shift(object, atm_imp_k)
```

Arguments

object	an object of ecld class with built-in ρ, ϵ
ki	numeric, a vector of σ -normalized log-strike
otype	character, specifying option type: c (default) or p.
dki	numeric, delta of ki for calculating slope
k	numeric, a vector of log-strike
atm_imp_k	numeric, the ATM implied log-strike. It is derived from ATM volatility times square root of time to expiration.

Value

a numeric vector, representing Q or skew of Q. For ecld.fixed_point_atm_ki, it is ATM ki. For ecld.fixed_point_shift, it is the shift.

Author(s)

Stephen H. Lihn

ecld.op_V

The O, V, U operators in option pricing model

Description

The O operator takes a vector of implied volatility $\sigma_1(k)$ and transforms them to a vector of normalized option prices. The V operator takes a vector of normalized option prices and transforms them to a vector of implied volatility $\sigma_1(k)$. If ttm is provided, $\sigma_1(k)$ will be divided by square root of 2 ttm and yield Black-Scholes implied volatility. The U operator calculates the log-slope of the option prices. The op_VL_quartic operator is the quartic composite of V x OGF, assuming epsilon and rho are deposited in the ecld object. The RN parameter for OGF is not available here. It is always assumed to be FALSE.

Usage

```
ecld.op_V(
  L,
  k,
  otype = "c",
  ttm = NaN,
  rho = 0,
  stop.on.na = FALSE,
  use.mc = TRUE
)
ecld.op_O(sigma1, k, otype = "c", rho = 0)
ecld.op_U_lag(L, k, sd, n = 2)
ecld.op_VL_quartic(
  object,
  k,
  otype = "c",
  ttm = NaN,
  stop.on.na = FALSE,
  use.mc = TRUE
)
```

Arguments

L	numeric, a vector of normalized local option prices
k	numeric, a vector of log-strike
otype	character, specifying option type: c (default) or p.
ttm	numeric, time to expiration (maturity), measured by fraction of year. If specified, V operator will adjust $\sigma_1(k)$ to Black-Scholes implied volatility. Default is NaN.
rho	numeric, specify the shift in the global mu.
stop.on.na	logical, to stop if fails to find solution. Default is to use NaN and not stop.
use.mc	logical, to use mclapply, or else just use for loop. Default is TRUE. For loop option is typically for debugging.

sigma1	numeric, a vector of implied volatility (without T)
sd	numeric, the stdev of the distribution. Instead, if an ecld or ecd object is provided, the stdev will be calculated from it.
n	numeric, number of lags in ecld.op_U_lag.
object	an object of ecld class created from ecld.quartic. This object contains the full quartic lambda model spec in order to be used in ecld.op_VL_quartic

Value

a numeric vector

Author(s)

Stephen H. Lihn

ecld.pdf

Calculate the PDF of an ecld object

Description

Calculate the PDF of an ecld object

Usage

```
ecld.pdf(object, x)
```

Arguments

object	an object of ecd class
x	numeric vector of x dimension

Value

numeric vector of the PDF

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(lambda=3)
x <- seq(-10, 10, by=1)
ecld.pdf(ld,x)
```

<code>ecld.quartic_Qp</code>	<i>The ATM volatility and skew of Q_p in quartic model</i>
------------------------------	---

Description

Compute the ATM location and ATM skew of Q_p in quartic model.

Usage

```
ecld.quartic_Qp(object, ki)

ecld.quartic_Q(object, ki, otype)

ecld.quartic_Qp_atm_ki(object, lower = -50, upper = -1.37)

ecld.quartic_Qp_rho(object, atm_ki = NaN, lower = -50, upper = -1.37)

ecld.quartic_Qp_skew(object, ki, dki = 0.1)

ecld.quartic_Qp_atm_skew(object, dki = 0.1, lower = -50, upper = -1.37)
```

Arguments

object	an object of ecd class
ki	numeric, order of the moment to be computed
otype	character, specifying option type with either c or p.
lower	numeric, optional value to specify the lower bound of ATM root finding. This is often needed when the smile is collapsed in the left wing.
upper	numeric, optional value to specify the upper bound of ATM root finding. This is often needed when the smile is collapsed significantly in the right wing.
atm_ki	numeric, if provided, take it as is without calculating again
dki	numeric, delta of ki for calculating slope

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
ld <- ecld.quartic(sigma=0.001*ecd.mp1, epsilon=0, rho=0, mu_plus=0)
ecld.quartic_Qp_atm_ki(ld, lower=-12, upper=-11)
ecld.quartic_Qp_atm_skew(ld, lower=-12, upper=-11)

## End(Not run)
```

eclt.quartic_Qp_atm_attr*Calculate ATM attributes from key quartic parameters***Description**

This utility takes a data frame of key quartic parameters, and generates several key ATM attributes. Input fields are: ttm - time to expiration, sigma - term structure of sigma, epsilon_ratio - term structure of epsilon/sigma, mu_plus_ratio - term structure of $(\mu_p - \mu_D)/\text{stdev}$. The output fields are: atm_ki, atm_kew, atm_vol, rho, and rho_ratio - rho/stdev.

Usage

```
eclt.quartic_Qp_atm_attr(df)

eclt.quartic_model_sample(dt, ttm, skew_adjusted = TRUE)

eclt.quartic_model_sample_attr(dt, ttm, target_file, skew_adjusted = TRUE)
```

Arguments

df	data.frame
dt	character, one of three sample dates used in the quartic model paper (YYYY-MM-DD)
ttm	numeric, list of time to expiration (T=1 for one year)
skew_adjusted	logical, if true, use skew adjusted T=0 intercept, else use the intercept from linear fit. Default is TRUE.
target_file	character, file location to cache the attribute data (to avoid lengthy repetitions)

Value

data.frame

Author(s)

Stephen H-T. Lihn

Examples

```
ttm <- seq(sqrt(90), sqrt(365), length.out=3)^2 / 365
epsr = 0.014 + 0*ttm
mupr <- -(eclt.quartic_SN0_max_RNV() - 0.2*sqrt(ttm))
## Not run:
df <- data.frame(ttm=ttm, sigma=0.2*sqrt(ttm/120), mu_plus_ratio=mupr, epsilon_ratio=epsr)
eclt.quartic_Qp_atm_attr(df)

## End(Not run)
```

`ecld.quartic_SN0_atm_ki`

The ATM RNO related constants and calculations in quartic model

Description

Computes the small sigma limit of ATM location, rho/stdev, and ATM skew of Q_p under the RNO measure in quartic model. Computes the maximum risk-neutral violation as an extension of RNO measure.

Usage

```
ecld.quartic_SN0_atm_ki()
ecld.quartic_SN0_rho_stdev()
ecld.quartic_SN0_skew()
ecld.quartic_SN0_max_RNV(sigma = 0)
```

Arguments

<code>sigma</code>	numeric, the volatility parameter
--------------------	-----------------------------------

Value

numeric

Author(s)

Stephen H-T. Lihn

`ecld.sd`

Compute statistics analytically for an ecld object

Description

Compute statistics for mean, var, skewness, kurtosis, from the known analytical result. SGED is supported.

Usage

```
ecld.sd(object)
ecld.var(object)
ecld.mean(object)
ecld.skewness(object)
```

```
ecld.kurtosis(object)
ecld.kurt(object)
```

Arguments

object an object of ecld class

Value

numeric or mpfr

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(3)
ecld.sd(ld)
ecld.var(ld)
ecld.mean(ld)
ecld.skewness(ld)
ecld.kurt(ld)
```

ecld.sged_const *The integral solutions of SGED*

Description

These integrals are mainly used as validation to analytic solutions. If you must use them, be mindful of their slower speeds.

Usage

```
ecld.sged_const(object)
ecld.sged_cdf(object, x)
ecld.sged_moment(object, order)
ecld.sged_mgf(object, t = 1)
ecld.sged_imgf(object, k, t = 1, otype = "c")
ecld.sged_ogf(object, k, otype = "c")
```

Arguments

object	an sged object of ecld class
x	a numeric vector of x
order	numeric, order of the moment to be computed
t	numeric, for MGF and IMGF
k	a numeric vector of log-strike
otype	character, specifying option type: c (default) or p.

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(3)
ecld.const(ld)
```

ecld.solve

*Analytic solution for $y(x)$ in lambda distribution***Description**

Analytic solution for $y(x)$ if available. $ecld.laplace_B$ is a utility function for the slopes of a skew Laplace distribution at lambda=2: B^+ and B^- with $B^0/2 = B^+ + B^-$. If sigma is provided, B notation is expanded for IMGF where $B_\sigma^+ B_\sigma^- = \exp(\mu_D)$. SGED is supported.

Usage

```
ecld.solve(a, b, ...)
ecld.laplace_B(beta, sgn = 0, sigma = 0)
ecld.solve_quartic(a, b, ...)
ecld.solve_by_poly(a, b, ...)
ecld.solve_isomorphic(a, b, ...)
```

Arguments

a	an object of ecld class
b	a vector of x values
...	Not used. Only here to match the generic signature.
beta	the skew parameter
sgn	sign of -1, 0, +1
sigma	the scale parameter, optional

Value

A vector for $y(x)$

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
x <- seq(-0.1, 0.1, by=0.01)
ecld.solve(ld,x)
```

ecld.y_slope

Analytic solution for the slope of $y(x)$ in lambda distribution

Description

Analytic solution for the slope of $y(x)$ if available. *ecld.y_slope_trunc* calculates the MGF truncation point where $dy/dx + t = 1$. SGED is supported.

Usage

```
ecld.y_slope(object, x)
ecld.y_slope_trunc(object, t = 1)
```

Arguments

object	an object of <i>ecld</i> class
x	a vector of x values
t	numeric, for MGF truncation

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
x <- seq(-0.1, 0.1, by=0.01)
ecld.y_slope(ld,x)
ecld.y_slope_trunc(ld)
```

ecldOrEcd-class *The ecldOrEcd class*

Description

The S4 class union of ecld and ecd, primarily used to define slot in ecop.opt class. Its usage is rather cumbersome, so the end user should avoid it as much as possible.

ecop-class *An S4 class to represent the top-level option model*

Description

The ecop class serves as an object-oriented container for the option pricing model. It does have a specific purpose at the moment - that is, to produce all the data for the charts of the paper, based on CBOE data structure. Therefore, user may not find it general enough. That probably will be the case for the time being until more popularity calls for a more generic container.

Slots

call the match.call slot
conf list, configuration
key character
symbol character
datadate Date
days numeric, days between datadate and expiry date
ttm numeric, time to maturity in days/365
int_rate numeric
div_yield numeric
put_data the put data of ecop.opt class
call_data the call data of ecop.opt class
put_conf list, the put configuration
call_conf list, the call configuration

Author(s)

Stephen H-T. Lihn

ecop.bs_implied_volatility*Implied volatility of Black-Sholes model*

Description

This is the standard library to calculate implied volatility σ_{BS} in Black-Sholes model. There is no external dependency on elliptic distribution.

Usage

```
ecop.bs_implied_volatility(
  V,
  K,
  S,
  ttm,
  int_rate = 0,
  div_yield = 0,
  otype = "c",
  stop.on.na = FALSE,
  use.mc = TRUE
)
```

Arguments

V	numeric vector of option prices
K	numeric vector of strike prices
S	length-one numeric for underlying price
ttm	length-one numeric for time to maturity, in the unit of days/365.
int_rate	length-one numeric for risk-free rate, default to 0.
div_yield	length-one numeric for dividend yield, default to 0.
otype	character, specifying option type: c or p.
stop.on.na	logical, to stop if fails to find solution. Default is to use NaN and not stop.
use.mc	logical, to use mclapply (default), or else just use for loop. For loop option is typically for debugging.

Value

The implied volatility σ_{BS} .

Examples

```
V <- c(1.8, 50)
K <- c(2100, 2040)
S <- 2089.27
T <- 1/365
y <- 0.019
ecop.bs_implied_volatility(V, K, S, ttm=T, div_yield=y, otype="c")
# expect output of 12.8886% and 29.4296%
```

`ecop.bs_option_price` *Calculate option price from implied volatility in Black-Sholes model*

Description

This is the standard library to calculate option price from implied volatility σ_{BS} in Black-Sholes model. There is no external dependency on elliptic distribution.

Usage

```
ecop.bs_option_price(ivol, K, S, ttm, int_rate = 0, div_yield = 0, otype = "c")
ecop.bs_call_price(ivol, K, S, ttm, int_rate = 0, div_yield = 0)
ecop.bs_put_price(ivol, K, S, ttm, int_rate = 0, div_yield = 0)
```

Arguments

<code>ivol</code>	numeric vector of implied volatility
<code>K</code>	numeric vector of strike prices
<code>S</code>	length-one numeric for underlying price
<code>ttm</code>	length-one numeric for time to maturity, in the unit of days/365.
<code>int_rate</code>	length-one numeric for risk-free rate, default to 0.
<code>div_yield</code>	length-one numeric for dividend yield, default to 0.
<code>otype</code>	character, c or p. Default is c.

Value

The call/put prices

Examples

```
ivol <- c(0.128886, 0.294296)
K <- c(2100, 2040)
S <- 2089.27
T <- 1/365
y <- 0.019
ecop.bs_option_price(ivol, K, S, ttm=T, div_yield=y, otype="c")
# expect output of c(1.8, 50)
```

```
ecop.find_fixed_point_lambda_by_atm_skew
```

Utility to find the fixed point lambda that matches ATM skew

Description

This utility finds the fixed point lambda from larger lambda to smaller lambda until the calculated ATM skew is smaller than ATM skew from data. It uses ecop.find_fixed_point_sd_by_lambda to locate stdev. Other smile related parameters are abstracted away via the closure function fn_get_ld1. This utility is used primarily to solve the fixed point ATM hypothesis (for VIX option smile). Note that this utility alone is not the full solution. Another utility is needed to match the two tails (via mu and epsilon). This utility doesn't handle beta either.

Usage

```
ecop.find_fixed_point_lambda_by_atm_skew(
  fn_get_ld1,
  lambda,
  step,
  atm_skew,
  k_atm,
  ttm,
  otype = "c",
  verbose = TRUE,
  msg_prefix = "",
  min_lambda = 1.1
)
```

Arguments

fn_get_ld1	function, takes stdev, lambda, beta as input, return ld1 object via ecop.get_ld_triple. This closure function encapsulates mu_plus_ratio, epsilon_ratio, atm_imp_k.
lambda	numeric, the lambda parameter.
step	numeric, increment to decrease lambda.
atm_skew	numeric, ATM skew from data.
k_atm	a vector of numeric, range of log-strike to calculate ATM skew via lm.
ttm	numeric, time to expiration, with 1 representing 1 year (365 days).
otype	character, option type. Default: "c".
verbose	boolean, print debug message. Default: FALSE.
msg_prefix	character, command line message prefix. Default: "".
min_lambda	numeric, do not try lambda lower than this and return it. Default is 1.1.

Value

numeric, representing lambda.

Author(s)

Stephen H-T. Lihn

ecop.find_fixed_point_sd_by_lambda

Utility to find the fixed point stdev when lambda is given

Description

This utility finds the fixed point stdev when lambda is given. Other smile related parameters are abstracted away via the closure function fn_get_ld1. This utility is used primarily to solve the fixed point ATM hypothesis (for VIX option smile). Note that this utility alone is not the full solution. Another utility is needed to match the ATM skew, and the two tails (via mu and epsilon). fn_get_ld1 should have the functional signature of fn_get_ld1(sd, lambda, beta=0) and returns an ecld object accordingly.

Usage

```
ecop.find_fixed_point_sd_by_lambda(  
  fn_get_ld1,  
  lambda,  
  beta = 0,  
  otype = "c",  
  verbose = FALSE  
)
```

Arguments

fn_get_ld1	function, takes stdev, lambda, beta as input, return ld1 object via ecop.get_ld_triple. This closure function encapsulates mu_plus_ratio, epsilon_ratio, atm_imp_k.
lambda	numeric, the lambda parameter. Must be positive. Default: 3.
beta	numeric, the skewness parameter. Default: 0.
otype	character, option type. Default: "c".
verbose	boolean, print debug message. Default: FALSE.

Value

numeric, representing stdev.

Author(s)

Stephen H-T. Lihn

`ecop.from_symbol_conf` *Constructor of ecop class by read conf for option sample data*

Description

Read conf for option sample data and fitting parameters

Usage

```
ecop.from_symbol_conf(
  key,
  conf_file = "conf/ecop-fit-conf.yml",
  conf_data = NULL,
  extdata_dir = NULL
)

ecop.read_symbol_conf(key, conf_file = "conf/ecop-fit-conf.yml")

ecop.build_opt(ecop, df, otype)
```

Arguments

key	character. The top-level key in conf
conf_file	file name for symbol config, default to conf/ecld-fit-conf.yml
conf_data	optionally feed config through a list. If this is not null, this takes priority and conf_file will be ignored.
extdata_dir	optionally specify user's own extdata folder
ecop	an ecop object with conf
df	dataframe of a single closing date and time to maturity
otype	option type

Value

the ecop object

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
conf <- ecop.read_symbol_conf("spx2_1d")
op <- ecop.from_symbol_conf("spx2_1d")

## End(Not run)
```

<code>ecop.get_ld_triple</code>	<i>Get triple list of ecld objects by stdev</i>
---------------------------------	---

Description

Construct triple list of ecld objects by stdev, with lambda, and ratios related to stdev. This utility is used primarily in fixed point ATM hypothesis (when simulating VIX option smile).

Usage

```
ecop.get_ld_triple(
  lambda = 3,
  sd = 1,
  beta = 0,
  mu_plus_ratio = 0,
  epsilon_ratio = 0,
  atm_imp_k = NaN,
  fn_shift = NULL
)
```

Arguments

<code>lambda</code>	numeric, the lambda parameter. Must be positive. Default: 3.
<code>sd</code>	numeric, the stdev parameter. Must be positive. Default: 1.
<code>beta</code>	numeric, the skewness parameter. Default: 0.
<code>mu_plus_ratio</code>	numeric, numeric, excess value in addition to <code>mu_D</code> , relative to the stdev. Default: 0.
<code>epsilon_ratio</code>	numeric, epsilon ratio relative to the stdev. Default: 0.
<code>atm_imp_k</code>	numeric, ATM implied log-strike. It is derived from ATM volatility times square root of time to expiration. If provided, it is used to calculate the fixed point shift, $-(\text{atm_imp_k} - \mu)$. Default: NaN. the rho slot in ld1 is populated with the value.
<code>fn_shift</code>	function, takes an ecld object and return the fixed point shift, $-(\text{atm_imp_k} - \mu)$. the rho slot in ld1 is populated with the value from this function. This serves as secondary method if you don't want to provide <code>atm_imp_k</code> directly.

Value

a triple list of ecld objects. ld0 has mu=0 as vanila object; ld1 has mu and rho as prescribed; ld2 has mu=`mu_D`.

Author(s)

Stephen H-T. Lihn

Examples

```
lds <- ecop.get_ld_triple(3, 0.1)
ld1 <- lds$ld1
```

ecop.opt-class

*An S4 class to represent the option data and model calculation***Description**

The ecop.opt class serves as an object-oriented container for the type-specific (p or c) option data.

Slots

- `call` the match.call slot
- `otype` character, option type
- `range.from` numeric, starting price range
- `range.to` numeric, ending price range
- `momentum` numeric, momentum for translation (T) operator
- `epsilon` numeric, asymptotic premium
- `k_cusp` numeric, the suggested cusp location for poly fit of prices
- `ecldOrEcd` the ecld/ecd class to calculate theoretical values in local regime
- `S` underlying price, this can be overridden by conf
- `S_raw` underlying price (before override)
- `strike` strike price
- `k` log-strike price
- `V_last` last option price
- `V_bid` bid option price
- `V_ask` ask option price
- `V` finalized option price (likely mid-point)
- `IV` implied volatility from the vendor

Author(s)

Stephen H. Lihn

ecop.polyfit_option

*Poly fit on option prices***Description**

The poly fits on logarithm of option prices are performed for each side of the suggested cusp (specified by `k.cusp`). This utility is used mainly to remove the market data noise for the calculation of log-slope of option prices.

Usage

```
ecop.polyfit_option(k, V, k.cusp, k.new, degree.left = 6, degree.right = 6)
```

Arguments

k	numeric, vector of log-strike
V	numeric, vectors of option prices
k.cusp	length-one numeric, the suggested cusp location
k.new	numeric, vector of log-strike to evaluate the poly fit
degree.left	length-one numeric, specifying the degree of poly fit for the left tail
degree.right	length-one numeric, specifying the degree of poly fit for the right tail

Value

The state prices from the poly fit

Author(s)

Stephen H-T. Lihn

ecop.read_csv_by_symbol
Read option data csv

Description

Read option data csv into dataframe. The dataframe is enriched with Date, expiration_date, days.

Usage

```
ecop.read_csv_by_symbol(symbol, extdata_dir = NULL)
ecop.enrich_option_df(df)
```

Arguments

symbol	character, option data symbol
extdata_dir	optionally specify user's own extdata folder
df	dataframe, it is assumed to be in CBOE heading format

Value

dataframe

Author(s)

Stephen H-T. Lihn

Examples

```
df <- ecop.read_csv_by_symbol("spxoption2")
```

ecop.term_master_calculator
*Master calculator for all the analytics of volatility smiles required for
a date*

Description

This is all-in-one calculator. The inputs are symbol, date (YYYY-MM-DD), and quartic config file location, and the optional external data directory. The data structure and documentation here are really rough. They are used to calculate the data needed for the quartic paper. They need to be polished and refined after the quartic paper is released.

Usage

```
ecop.term_master_calculator(
  symbol,
  date_str,
  int_rate = 0,
  div_yield = 0,
  config_file = NULL,
  extdata_dir = NULL
)

ecop.smile_data_calculator(idx, df_day, master, int_rate, div_yield, otype)

ecop.term_atm(opt)
```

Arguments

symbol	character pointing to the standard option data file
date_str	character in the form of YYYY-MM-DD
int_rate	numeric, the interest rate used to calculate BS implied volatility from market data
div_yield	numeric, the dividend yield used to calculate BS implied volatility from market data
config_file	character, config file from the quarter optimx fit
extdata_dir	character, external data directory
idx	integer, indicating the index of the option chain
df_day	data frame for the day
master	the list structure from the output of ecop.term_master_calculator
otype	character, option type of p or c
opt	the list structure from the output of ecop.smile_data_calculator

Value

The nested list containing all analytics of volatility smiles for a date. The first level keys are the date strings. The first level attributes are quartic.config which is a data frame, lists of days, volumes, classes, and values of undl_price, max_idx.

ecop.term_plot_3x3 *Produce 3x3 plot of volatility smiles for a date*

Description

This utility produces 3x3 plot of volatility smiles for a date. It is used for the term structure paper.

Usage

```
ecop.term_plot_3x3(  
  term_data,  
  date_str,  
  trim_points = 151,  
  target_days = NULL,  
  add.first.day = TRUE,  
  show.put.bid = FALSE  
)  
  
ecop.term_target_days_default  
  
ecop.term_realized_days(target_days, days)  
  
ecop.term_idx_range(realized_days, days)
```

Arguments

term_data	term structure data for one date, produced from ecop.term_master_calculator
date_str	character in the form of YYYY-MM-DD
trim_points	integer, specifying number of data points to present in the plots
target_days	list of ceiling days for the plot
add.first.day	logic, whether to add the first expiration date to target_days. Default is TRUE.
show.put.bid	logic, show bid smile for put option. Default is FALSE.
days	list of days to expiration from market data
realized_days	list of days realized for the plot

Format

An object of class numeric of length 8.

Value

The 3x3 plot

`ecop.vix_plot_3x3` *Produce 3x3 plot of VIX volatility smiles for a date*

Description

This utility produces 3x3 plot of volatility smiles for a date. It is used for the VIX option paper.

Usage

```
ecop.vix_plot_3x3(date_str, option_data, result, result_avg)
```

Arguments

<code>date_str</code>	character in the form of YYYY-MM-DD
<code>option_data</code>	dataframe, read from <code>ecop.read_csv_by_symbol</code>
<code>result</code>	dataframe, the VIX optimx result
<code>result_avg</code>	dataframe, the VIX optimx result using average lambda for all expirations

Value

The 3x3 plot

`ellipticity.ecd` *Ellipticity of ecd object*

Description

Ellipticity of ecd object, defined as half of the distance between the two elliptic points.

Usage

```
## S3 method for class 'ecd'
ellipticity(object, tol = 1e-04)

ellipticity(object, tol = 1e-05)

## S4 method for signature 'ecd'
ellipticity(object, tol = 1e-04)
```

Arguments

<code>object</code>	An object of ecd class
<code>tol</code>	Numeric, the tolerance of precision during subdivision. Default: 1e-4 of stdev.

Value

a list with 3 major numbers: `xe1`= negative `x_e`, `xe2`= positive `x_e`, `avg`= ellipticity

Examples

```
d <- ecd(0,1)
ellipticity(d)
```

history.ecdb*List of history in the Elliptic DB*

Description

List of unique history reflecting the bootstrap activities.

Usage

```
## S3 method for class 'ecdb'
history(object)

history(object)

## S4 method for signature 'ecdb'
history(object)
```

Arguments

object an object of eedb class.

Value

list of history

Author(s)

Stephen H-T. Lihn

integrate_pdf.ecd*Integrate a function with PDF of the distribution*

Description

Integrate a function with PDF of the distribution. The integration is separated into three segments to ensure convergence.

Usage

```
## S3 method for class 'ecd'
integrate_pdf(
  object,
  f,
  lower,
  upper,
  ...,
  show.warning = TRUE,
  verbose = FALSE
)

integrate_pdf(object, f, lower, upper, ...)

## S4 method for signature 'ecd'
integrate_pdf(
  object,
  f,
  lower,
  upper,
  ...,
  show.warning = TRUE,
  verbose = FALSE
)
```

Arguments

<code>object</code>	An object of <code>ecd</code> class
<code>f</code>	An R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
<code>lower</code>	Numeric, the lower limit of integration. Can be infinite.
<code>upper</code>	Numeric, the upper limit of integration. Can be infinite.
<code>...</code>	Addtional arguments for <code>f</code> .
<code>show.warning</code>	logical, display warning messages.
<code>verbose</code>	logical, display timing information, for debugging purpose.

Value

A list of class "integrate".

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
integrate_pdf(d, function(x){x^2}, -Inf, Inf)
```

<code>jinv.ecd</code>	<i>J-invariant of the elliptic curve $y(x)$</i>
-----------------------	--

Description

J-invariant of the elliptic curve $y(x)$

Usage

```
## S3 method for class 'ecd'
jinv(object, no.validate = FALSE)

jinv(object, no.validate = FALSE)

## S4 method for signature 'ecd'
jinv(object, no.validate = FALSE)
```

Arguments

<code>object</code>	an object of ecd class
<code>no.validate</code>	logical, if TRUE, don't validate presence of beta. Default is FALSE.

Value

the j-invariant

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd(1,1)
j <- jinv(d)
```

<code>k2mnt</code>	<i>Conversion between cumulants and moments</i>
--------------------	---

Description

Implements conversion between the first four cumulants and moments

Usage

```
k2mnt(k)
mnt2k(m)
```

Arguments

<code>k</code>	numeric, first four cumulants.
<code>m</code>	numeric, first four moments.

Value

numeric

Author(s)

Stephen H-T. Lihn

`lamp`

Constructor of lamp class

Description

Construct an lamp class by providing the required parameters. The default is the unit quartic lambda process.

Usage

```
lamp(
  lambda = NaN,
  T.inf = 86400 * 1000,
  rnd.n = 1e+06,
  alpha = NaN,
  beta = 0,
  rnd.walk = 1,
  sd = NaN,
  sd.method = 0,
  N.lower = 0,
  N.upper = 1000,
  file = character(0)
)
```

Arguments

<code>lambda</code>	numeric, the lambda parameter. Must be positive. Default is NaN.
<code>T.inf</code>	numeric, the infinite bound to cut off Levy sums. Default is 86400000.
<code>rnd.n</code>	numeric, the length of one rnd call. Default is 1000000.
<code>alpha</code>	numeric, optional, if you don't like to use lambda. Default is NaN. Either lambda or alpha must be specified with a positive number.
<code>beta</code>	numeric, the skewness parameter. Default: 0.
<code>rnd.walk</code>	numeric, random walk method, 1: Laplace, 2: Binomial/normal. Default is 1.
<code>sd</code>	numeric, standard deviation adjustment. No adjustment if NaN. Default is NaN.
<code>sd.method</code>	numeric, methodology of sd adjustment. 0 means in scale parameter, 1 means in Levy sums. Default is 0.

N.lower	numeric, the lower bound of N to truncate the boundary effect. Default is 0.
N.upper	numeric, the upper bound of N to limit the outliers. Default is 1000.
file	character, file path to save the object and simulation result. Default is character(0).

Value

an object of lamp class

Author(s)

Stephen H-T. Lihn

Examples

```
lp <- lamp(4, T.inf=86400*1000000)
```

lamp-class

An S4 class to represent the lambda process

Description

The lamp class serves as an object-oriented interface for the lambda process. The main purpose of the class is to store all the parameters required for simulation.

Slots

call	the match.call slot.
lambda	numeric, lambda index of lambda process, which is 2/alpha.
alpha	numeric, stable alpha. This is derived from lambda for convenience reason.
beta	numeric, stable beta.
pm	numeric, parameterization, default to 1.
rnd.walk	numeric, Random walk method. Default is 1.
sd	numeric, standard deviation adjustment. No adjustment if NaN.
sd.method	numeric, methodology of sd adjustment. 0 means in scale parameter, 1 means in Levy sums.
T.inf	numeric, the infinite bound to cut off the Levy sums.
rnd.n	numeric, the length of one rnd call.
N.lower	numeric, the lower bound of N to truncate the boundary effect. Default is 0.
N.upper	numeric, the upper bound of N to limit the outliers. Default is 1000.
use.mpfr	logical, use Mpfr for high precision sums.
file	character, file path to save the object and simulation result.
tau	numeric, storage for the stable random variables.
tau_i	numeric, for internal use, length or index of tau.
Z_i	numeric, length of Z.
Z	numeric, simulation result of the lambda process, Z.
B	numeric, simulation result of the binomial process, B.
N	numeric, simulation result of the count process, N.
tm	POSIXct, timestamp of simulation.

Author(s)

Stephen H. Lihn

lamp.generate_tau	<i>Generate tau from stable distribution</i>
-------------------	--

Description

Generate tau, a random sequence representing the stable random walk process.

Usage

```
lamp.generate_tau(object)
```

Arguments

object	an object of lamp class
--------	-------------------------

Value

an object of lamp class with tau populated, tau_i is set to 1.

Author(s)

Stephen H-T. Lihn

Examples

```
lp <- lamp(4, rnd.n=10)
lp1 <- lamp.generate_tau(lp)
lp1@tau
```

lamp.plot_sim4	<i>Plot the simulation result in standard layout</i>
----------------	--

Description

Plot the simulation result in standard layout, with 4 or 6 charts The PDF and log(PDF) histogram of Z, the lambda process. The log(PDF) histogram of N, the stable count process. The log(PDF) histogram of B, the binomial random walk process. The 6-chart plot also includes the asymptotic kurtosis and stdev vs the bps of data points dropped in Z.

Usage

```
lamp.plot_sim4(object)
```

```
lamp.plot_sim6(object)
```

Arguments

object an object of lamp class

Value

an object of lamp class

Author(s)

Stephen H-T. Lihn

lamp.qsl_fit_config *Read QSLD fit config*

Description

Read QSLD fit config for plot or custom fit utility. The xtable print utility is also provided to generate high quality latex output for publication purpose.

Usage

```
lamp.qsl_fit_config(key = NULL, extdata_dir = NULL, filename = NULL)  
lamp.qsl_fit_config_xtable(df)
```

Arguments

key character, the top-level key for config, default to NULL.
extdata_dir optionally specify user's own extdata folder, default is NULL.
filename character, optionally specify user's own config file name, default is NULL.
df the data frame generated from `lamp.qsl_fit_config`.

Value

The data.frame object for the config

Examples

```
c <- lamp.qsl_fit_config()
```

`lamp.qsl_fit_plot` *Plot the fit to asset returns using quartic stable lambda distribution*

Description

Plot the fit to asset returns using quartic stable lambda distribution

Usage

```
lamp.qsl_fit_plot(
  key,
  debug = FALSE,
  plot.type = c("pdf", "log-pdf"),
  qqplot.n = 1e+06,
  extdata_dir = NULL,
  filename = NULL
)
```

Arguments

key	character, the key(s) to retrieve configuration for plot. If key is provided as single-row data frame, then it will be used directly as config.
debug	logical, if true, print debug information. Default is FALSE.
plot.type	character, type of plot: pdf, log-pdf, qqplot. Default is c("pdf", "log-pdf").
qqplot.n	numeric, specify number of QSLD simulations for qqplot utility, default is 1000000.
extdata_dir	optionally specify user's own extdata folder, default is NULL.
filename	character, optionally specify user's own config file name, default is NULL.

Value

returns a list of each key and its data and config blocks as nested list

Author(s)

Stephen H-T. Lihn

`lamp.sd_factor` *Calculate sd adjustment factor*

Description

Calculate sd adjustment factor. For L2 random walk, it is the power of $1/(1+\alpha/2)$. For L1 random walk, it is the power of 1. This factor can be used to adjust either the scale parameter of the stable distribution or T.inf that cuts off the Levy sums.

Usage

```
lamp.sd_factor(object)
```

Arguments

object an object of lamp class

Value

numeric, the sd factor

Author(s)

Stephen H-T. Lihn

lamp.simulate1 *Simulate one sequence of lambda process from stable distribution*

Description

Simulate lambda process from one random sequence representing the stable random walk process.

Usage

```
lamp.simulate1(object, drop = 10, keep.tau = 1)
```

Arguments

object an object of lamp class

drop numeric, number of tau to discard at the end. Default is 10.

keep.tau numeric, 0 to clean up, 1 to return unused tau, 2 to return all tau. Default is 1.

Value

an object of lamp class with Z, B, N populated

Author(s)

Stephen H-T. Lihn

Examples

```
lp <- lamp(4, T.inf=8640, rnd.n=100000)
lp1 <- lamp.simulate1(lp)
```

lamp.simulate_iter *Simulate lambda process from stable distribution iteratively*

Description

Simulate lambda process from stable distribution iteratively until target length of result is reached. It uses multi-core capability to run `lamp.simulate1` in parallel. If file slot is specified, simulation result will be persisted to it periodically. A plot interface is provided to monitor the progress. A CPU temperature interface is provided to control CPU from overheating.

Usage

```
lamp.simulate_iter(
  object,
  use.mc = 4,
  sim.length = 1000,
  reset.cache = FALSE,
  drop = 10,
  keep.tau = 1,
  plot.util = lamp.plot_sim6,
  cpu.temperature = 68,
  cpu.temperature.util = NULL
)
```

Arguments

<code>object</code>	an object of <code>lamp</code> class
<code>use.mc</code>	numeric, number of cores for parallel simulations. Default is 4.
<code>sim.length</code>	numeric, number of Z to simulate. Default is 1000.
<code>reset.cache</code>	logical, to reset simulation cache or not prior the run. Default is FALSE.
<code>drop</code>	numeric, number of tau to discard at the end per iteration. Default is 10.
<code>keep.tau</code>	numeric, 0 to clean up, 1 to return unused tau, 2 to return all tau. Default is 1.
<code>plot.util</code>	function, interface to plot simulation results. Default is <code>lamp.plot_sim4</code> .
<code>cpu.temperature</code>	numeric, temperature above which is overhead. Default is 68.
<code>cpu.temperature.util</code>	function, interface to get CPU temperature. Default is NULL.

Value

an object of `lamp` class with Z, B, N populated

Author(s)

Stephen H-T. Lihn

`lamp.stable_rnd_walk` *Calculate the stable random walk*

Description

Calculate the stable random walk. There are 4 types of random walk you can specify: 11. Laplace(0,1). No skewess. 1. Experimental Laplace random walk via Gauss-Laplace transmutation. 22. Normal distribution $N(0, \sqrt{n}) * \text{epsilon}$. No skewess. 2. Binomial random walk, $b * \text{epsilon}$. This can produce skewness.

Usage

```
lamp.stable_rnd_walk(object, n, b)
```

Arguments

object	an object of lamp class
n	numeric, number of items in Levy sums
b	numeric, cumulative sum of signs in Levy sums

Value

numeric, the value of the random walk

Author(s)

Stephen H-T. Lihn

`levy.dlambda` *Standard Lambda distribution*

Description

Standard Lambda distribution PDF that can take complex argument.

Usage

```
levy.dlambda(x, lambda = 4)
```

Arguments

x	numeric, complex, mpfr, mpfc
lambda	numeric. Default is 4, the quartic distribution.

Value

PDF in the same type as x

Author(s)

Stephen H. Lihn

Examples

```
x = seq(1,10)
y = levy.dlambda(x)
```

levy.domain_coloring *Domain coloring of Laplace kernel of lambda distribution*

Description

Domain coloring on the complex plane of Laplace kernel of lambda distribution, $\exp(ixt) P(x)$, where $P(x)$ is the PDF of a lambda distribution. This is a visualization utility to get insight how the Laplace transform works for lambda distribution. The behavior on the complex plane is deeply associated with the MGF, the skew Levy distribution, and the SaS distribution.

Usage

```
levy.domain_coloring(t, rec, n = 200, lambda = 4)
```

Arguments

t	numeric or complex
rec	numeric, define the rectangle of plot in the order of (x1, x2, y1, y2)
n	numeric, number of points per axis. Default is 200. Use 1000 for better resolution.
lambda	numeric. Default is 4, and is the only value allowed.

Value

return value of call to `grid.arrange()`

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
levy.domain_coloring(0.1, c(-25, 50, -50, 50))
levy.domain_coloring(0.1i, c(-25, 25, -25, 25))

## End(Not run)
```

levy.dskewed

*Skewed Levy distribution in Levy statistics***Description**

Skewed Levy distribution PDF. In our context, "skewed" means "completed asymmetric alpha-stable", or called "one-sided alpha-stable". And we use lambda = 2/alpha.

Usage

```
levy.dskewed(x, lambda = 4)
```

Arguments

x	numeric, complex, mpfr, mpfc
lambda	numeric. Default is 4, the Levy distribution as is generally called.

Value

PDF in the same type as x

Author(s)

Stephen H. Lihn

Examples

```
x = seq(1,10)
y = levy.dskewed(x)
```

moment.ecd

*Compute the moment of ecd via integration***Description**

Compute the moment of ecd via integration between -Inf and Inf. The asymp.lower and asymp.upper parameters are used for asymptotic statistics, to study the effect of finite observations.

Usage

```
## S3 method for class 'ecd'
moment(
  object,
  order,
  center = FALSE,
  asymp.lower = -Inf,
  asymp.upper = Inf,
  verbose = FALSE
)
```

```

moment(
  object,
  order,
  center = FALSE,
  asymp.lower = -Inf,
  asymp.upper = Inf,
  verbose = FALSE
)

## S4 method for signature 'ecd'
moment(
  object,
  order,
  center = FALSE,
  asymp.lower = -Inf,
  asymp.upper = Inf,
  verbose = FALSE
)

```

Arguments

object	an object of ecd class
order	numeric. Order of the moment to be computed
center	logical. If set to TRUE, calculate central moments. Default: FALSE.
asympt.lower	numeric, lower bound for asymptotic statistics, default: -Inf.
asympt.upper	numeric, upper bound for asymptotic statistics, default: Inf.
verbose	logical, display timing information, for debugging purpose.

Value

Numeric. The moment.

Author(s)

Stephen H. Lihn

Examples

```

d <- ecd()
moment(d, 2)

```

Description

The S4 class union of numeric and mpfr, primarily used to define slots in ecd class. The use of MPFR does not necessarily increase precision. Its major strength in ecd is ability to handle very large numbers when studying asymptotic behavior, and very small numbers caused by small sigma when studying high frequency option data. Since there are many convergence issues with integrating PDF using native integrateR library, the ecd package adds many algorithms to improve its performance. These additions may decrease precision (knowingly or unknowingly) for the sake of increasing performance. More research is certainly needed in order to cover a vast range of parameter space!

plot_2x2.ecd	<i>Standard 2x2 plot for sample data</i>
--------------	--

Description

Standard 2x2 plot for sample data

Usage

```
plot_2x2.ecd(object, ts, EPS = FALSE, eps_file = NA)

plot_2x2(object, ts, EPS = FALSE, eps_file = NA)

## S4 method for signature 'ecd'
plot_2x2(object, ts, EPS = FALSE, eps_file = NA)
```

Arguments

object	An object of ecd class.
ts	The xts object for the timeseries.
EPS	Logical, indicating whether to save the plot to EPS, default = FALSE
eps_file	File name for eps output

Examples

```
## Not run:
plot_2x2(d, ts)

## End(Not run)
```

qsld.fit*Fit observations to QSLD via MLE*

Description

This utility will fit the observations to qsld with MLE using optimx. There are three features: First, it has the ability to provide initial estimate to save the user from the headache of guessing. Second, the user has the flexibility of fixing convolution, beta.a, and/or nu0/theta ratio. And the user can ask the utility to estimate mu as well. Third, the MLE optimization comes with two flavors: (a) the log-likelihood calculated from the PDF of all observations; or (b) the log-likelihood calculated from the histogram of the observations. The later is much faster than the former. One can use the later to obtain a good estimate, then feed it into the former, if necessary. This utility also comes with an LMS regression method called "pdf.lms". This option can be used as a preprocessor to the MLE methods. It regresses the theoretical PDF against the empirical PDF obtained from the histogram, and minimizes the LMS of PDF difference within 2-stdev and log(PDF) difference within 4-stdev.

Usage

```
qsld.fit(
  x,
  breaks,
  init.qsld,
  method = "fast.mle",
  fix.convo = NaN,
  fix.beta.a = NaN,
  fix.nu0.ratio = NaN,
  derive.mu = TRUE,
  plot.interval = 20,
  verbose.interval = 20,
  itnmax = 500
)
```

Arguments

<code>x</code>	numeric, the observation of log-returns.
<code>breaks</code>	numeric, the breaks for the histogram of observations. For lme, this parameter is only for display purpose.
<code>init.qsld</code>	an object of sld class as an initial qsld guess. The user can request the utility to estimate the initial parameters by setting nu0 to NaN. However, t must be provided.
<code>method</code>	character, optimization algorithm to use, it could be either <code>fast.mle</code> , <code>mle</code> , or <code>pdf.lms</code> . Default is <code>fast.mle</code> .
<code>fix.convo</code>	numeric, fix convolution to a specific number, default is NaN.
<code>fix.beta.a</code>	numeric, fix annualized beta to a specific number, default is NaN.
<code>fix.nu0.ratio</code>	numeric, fix ν_0/θ ratio to a specific number, default is NaN.
<code>derive.mu</code>	logical, if specified, to derive mu automatically, default is TRUE.
<code>plot.interval</code>	numeric, interval of iterations to plot the fit, default is 20. If set to zero, the plot is disabled.

```

verbose.interval
    numeric, interval of iterations to print verbose message, default is 20. If set to
    zero, the verbose message is disabled.
itnmax
    numeric, specify maximum iterations for optimx, default is 500.

```

Value

a list of two components: qslld as an object of sld class representing the QSLD fit; optimx.out storing the raw output from optimx.

Author(s)

Stephen H-T. Lihn

Examples

```

## Not run:
x <- ecd.data.arr("spx", lag=1, drop=1)$x
breaks <- 200
t <- 1/250
d <- qslld(t=t)
d@nu0 <- NaN # request utility to estimate

## End(Not run)

```

qsl_kurtosis_analytic *Analytic solutions on the statistics of quartic stable lambda distribution*

Description

Several analytic solutions on the statistics of quartic stable lambda distribution (QSLD) are implemented. These functions provide precise validation on the distribution.

Usage

```

qsl_kurtosis_analytic(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0)

qsl_skewness_analytic(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0)

qsl_variance_analytic(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0)

qsl_std_pdf0_analytic(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0)

qsl_pdf_integrand_analytic(
  x,
  nu,
  t = 1,
  nu0 = 0,
  theta = 1,
  convo = 1,
  beta.a = 0,
  mu = 0
)

```

Arguments

t	numeric, the time parameter, where the variance is t, default is 1.
nu0	numeric, the location parameter, default is 0.
theta	numeric, the scale parameter, default is 1.
convo	numeric, the convolution number, default is 1.
beta.a	numeric, the skewness parameter, default is 0. This number is annualized by $\text{sqrt}(t)$.
x	numeric, vector of responses.
nu	numeric, vector of nu in the pdf integrand, starting from 0 (not nu0).
mu	numeric, the location parameter, default is 0.

Value

numeric

Author(s)

Stephen H-T. Lihn

References

For more detail, see Appendix C of Stephen Lihn (2017). *A Theory of Asset Return and Volatility under Stable Law and Stable Lambda Distribution*. SSRN: 3046732, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3046732.

Examples

```
# obtain the variance for SPX 1-day distribution
var <- qsl_variance_analytic(t=1/250, nu0=6.92/100, theta=1.17/100, convo=2, beta=-1.31)
```

quantilize.ecd	Add the quantile data to the ecd object
----------------	---

Description

Add the quantile data to the ecd object if it is not created yet.

Usage

```
## S3 method for class 'ecd'
quantilize(object, show.warning = FALSE)

quantilize(object, show.warning = FALSE)

## S4 method for signature 'ecd'
quantilize(object, show.warning = FALSE)
```

Arguments

- object an object of ecd class
 show.warning logical, if TRUE, display a warning message. Default is FALSE.

Value

an object of ecd class with a newly generated ecdq object.

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
d <- ecd(-1,1)
quantilize(d)

## End(Not run)
```

read.ecdb

Read API for the ecdb

Description

Read ecdb into data.frame. This can be accomplished by either specifying the range of alpha, gamma or the cartesian product of alpha, gamma point by point, or both. If both are specified, it follows a similar logic as plot how x,y is scoped by xlim, ylim.

Usage

```
## S3 method for class 'ecdb'
read(
  object,
  alpha = NULL,
  gamma = NULL,
  alim = NULL,
  glim = NULL,
  cusp = 0,
  polar_ext = FALSE
)

read(
  object,
  alpha = NULL,
  gamma = NULL,
  alim = NULL,
  glim = NULL,
  cusp = 0,
  polar_ext = FALSE
)
```

```
## S4 method for signature 'ecdb'
read(
  object,
  alpha = NULL,
  gamma = NULL,
  alim = NULL,
  glim = NULL,
  cusp = 0,
  polar_ext = FALSE
)
```

Arguments

object	an object of ecdb class
alpha, gamma	numeric vectors of points for cartesian product
alim, glim	length-two numeric vectors of min and max range
cusp	numeric. Type of cusp. Only 0 and 1 are allowed. If cusp=1, read cusp data on the critical line. Reading cusp data must be done from the alpha side. Default: 0.
polar_ext	logical, for polar coordinate extension: R, theta, angle. Default: FALSE.

Value

The data.frame from ECDATTR table.

rlaplace0 *Laplace distribution*

Description

Implements some aspects of Laplace distribution (based on stats package) for stable random walk simulation.

Usage

```
rlaplace0(n, b = 1)
dlaplace0(x, b = 1)
```

Arguments

n	numeric, number of observations.
b	numeric, the scale parameter, where the variance is 2*b^2.
x	numeric, vector of responses.

Value

numeric, standard convention is followed: d* returns the density, p* returns the distribution function, q* returns the quantile function, and r* generates random deviates.

Author(s)

Stephen H-T. Lihn

sld	<i>Constructor of sld class</i>
-----	---------------------------------

Description

Construct an [sld-class](#) by providing the required parameters. The qsld constructor is also provided by hiding and defaulting lambda to 4.

Usage

```
sld(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0, lambda = 4)  
qsld(t = 1, nu0 = 0, theta = 1, convo = 1, beta.a = 0, mu = 0)
```

Arguments

t	numeric, the time parameter. Must be positive. Default: 1.
nu0	numeric, the floor volatility parameter. Must be positive. Default: 0.
theta	numeric, the volatility scale parameter. Must be positive. Default: 1.
convo	numeric, the convolution parameter. Must be positive. Default: 1.
beta.a	numeric, the skewness parameter. Default: 0.
mu	numeric, the location parameter. Default: 0.
lambda	numeric, the lambda parameter. Must be positive. Default: 4.

Value

an object of sld class

Author(s)

Stephen H-T. Lihn

Examples

```
d <- sld()  
d <- qsld()
```

sld-class*An S4 class to represent the stable lambda distribution*

Description

The `sld` class serves as an object-oriented interface for the stable lambda distribution. The `sld` prefix is also used as the namespace for the related analytic formulae derived in stable lambda distribution.

Slots

```
call  the match.call slot
t    numeric
nu0  numeric
theta numeric
convo numeric
beta.a numeric
mu   numeric
lambda numeric, this is default to 4.
```

Details

See [ds1](#) for definition of stable lambda distribution.

Author(s)

Stephen H. Lihn

References

For more detail, see Section 8.2 of Stephen Lihn (2017). *A Theory of Asset Return and Volatility under Stable Law and Stable Lambda Distribution*. SSRN: 3046732, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3046732.

See Also

[dstablecnt](#) for $N_\alpha(\cdot)$, and [dstdlap](#) for $f_L^{(m)}(\cdot)$.

sld.sd*Compute statistics analytically for an sld object*

Description

Compute statistics for mean, var, skewness, kurtosis for SLD. These functions are just wrappers on [ks1](#). If you need to calculate the statistics in quantity, you should use [ks1](#) or [kqsl](#) directly.

Usage

```
sld.sd(object)  
sld.var(object)  
sld.mean(object)  
sld.skewness(object)  
sld.kurtosis(object)  
sld.kurt(object)
```

Arguments

object an object of sld class

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
d <- qsld(nu0=10.4, theta=1.6, convo=2)  
sld.sd(d)  
sld.var(d)  
sld.mean(d)  
sld.skewness(d)  
sld.kurt(d)
```

solve.ecd

Solve the elliptic curve $y(x)$ **Description**

Solve the elliptic curve $y(x)$ by constructing a cubic polynomial from ecd object. Then solve it and take the smallest real root.

Usage

```
## S3 method for class 'ecd'
solve(a, b, ...)

## S4 method for signature 'ecd'
solve(a, b, ...)
```

Arguments

a	An object of ecd class
b	A vector of x values
...	Not used. Only here to match the generic signature.

Value

A vector of roots for $y(x)$

Examples

```
d <- ecd()
x <- seq(-100,100,by=0.1)
y <- solve(d,x)
```

solve_sym.ecd

*Analytic solution for a symmetric elliptic curve***Description**

Analytic solution for a symmetric elliptic curve $y(x)$

Usage

```
## S3 method for class 'ecd'
solve_sym(object, x)

solve_sym(object, x)

## S4 method for signature 'ecd'
solve_sym(object, x)
```

Arguments

object	an object of ecd class
x	array of x dimension

Value

array of y

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd()
x <- seq(-100,100,by=0.01)
y <- solve_sym(d,x)
```

solve_trig.ecd *Trigonometric solution for a elliptic curve*

Description

Use Chebyshev trigonometry for a depressed cube to solve a elliptic curve $y(x)$.

Usage

```
## S3 method for class 'ecd'
solve_trig(object, x)

solve_trig(object, x)

## S4 method for signature 'ecd'
solve_trig(object, x)
```

Arguments

object	an object of ecd class
x	array of x dimension

Value

array of y

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd()
x <- seq(-100,100,by=0.1)
y <- solve_trig(d,x)
```

summary.ecdb	<i>Summary for the Elliptic DB (ECDB)</i>
--------------	---

Description

Summary for the Elliptic DB (ECDB)

Usage

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

summary(object, ...)

## S4 method for signature 'ecdb'
summary(object, ...)
```

Arguments

object	an object of ecdb class.
...	more arguments for summary. Currently not used.

Author(s)

Stephen H-T. Lihn

Examples

```
summary(ecdb())
```

write.ecdb	<i>Write API for the ecdb for a list of basic ecdattr objects</i>
------------	---

Description

It takes a list of basic ecdattr objects, enrich them in parallel, then save them to ecdb.

Usage

```
## S3 method for class 'ecdb'
write(x, object)

write(x, object)

## S4 method for signature 'list,ecdb'
write(x, object)
```

Arguments

x	a list of basic ecdattr objects
object	an object of ecdb class

Value

The row count

y_slope.ecd	<i>Slope of y(x)</i>
-------------	----------------------

Description

Slope of $y(x)$, that is, dy/dx .

Usage

```
## S3 method for class 'ecd'  
y_slope(object, x)  
  
y_slope(object, x)  
  
## S4 method for signature 'ecd'  
y_slope(object, x)
```

Arguments

object	an object of ecd class
x	a numeric vector of x dimension

Value

a numeric vector of dy/dx

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(0,1)  
x <- seq(-20,20,by=0.01)  
yp <- y_slope(d,x)
```

Index

- * **ATM**
 - ecld.fixed_point_SN0_atm_ki, 56
 - ecld.quartic_Qp, 69
 - ecld.quartic_SN0_atm_ki, 71
- * **Domain-Coloring**
 - levy.domain_coloring, 98
- * **Laplace**
 - dstdlap, 12
 - rlaplace0, 106
- * **PDF**
 - levy.dlambda, 97
 - levy.dskeewed, 99
- * **QSLD**
 - lamp.qsl_fit_plot, 94
- * **Q**
 - ecld.op_0, 66
- * **Stable**
 - dstablecnt, 11
- * **analytic**
 - ecd.solve_cusp_asym, 42
- * **cdf**
 - ecd.ccdf, 17
 - ecd.cdf, 18
 - ecd.imgf, 29
 - ecld.cdf, 55
- * **class**
 - ecd-class, 15
 - ecdq-class, 51
- * **constructor**
 - ecd, 14
 - ecd-class, 15
 - ecd.cusp, 20
 - ecd.estimate_const, 26
 - ecd.polar, 38
 - ecd.setup_const, 41
 - ecd.toString, 43
 - ecdattr, 46
 - ecdb, 49
 - ecdq, 51
 - ecdq-class, 51
 - ecld, 52
 - ecop.from_symbol_conf, 80
 - ecop.get_ld_triple, 81
- * **lamp**
 - sld, 107
- * **cubic**
 - ecd.cubic, 19
- * **cusp**
 - ecd.cusp, 20
 - ecd.cusp_a2r, 21
 - ecd.cusp_std_moment, 21
- * **datasets**
 - ecd.mpfr, 34
 - ecop.term_plot_3x3, 85
- * **data**
 - ecd.manage_hist_tails, 32
 - ecd.read_csv_by_symbol, 39
 - ecop.read_csv_by_symbol, 83
 - lamp.qsl_fit_config, 93
- * **discriminant**
 - ecd.adj_gamma, 16
- * **distribution**
 - dec, 7
 - ecd.has_quantile, 29
 - ecd.pdf, 37
 - ecld.pdf, 68
 - quantilize.ecd, 104
- * **ecdattr**
 - ecdattr, 46
 - ecdattr-class, 47
 - ecdattr.enrich, 47
 - ecdattr.pairs, 48
 - ecdattr.pairs_polar, 48
- * **ecdb**
 - bootstrap.ecdb, 6
 - ecdb-class, 49
 - ecdb.dbSendQuery, 50
 - ecdb.protectiveCommit, 50
 - history.ecdb, 87
 - read.ecdb, 105
 - summary.ecdb, 112
 - write.ecdb, 112
- * **ecdq**
 - ecdq-class, 51
- * **ecd**
 - ecd.cusp, 20

- ecd.cusp_std_moment, 21
- ecd.polar, 38
- * **ecld**
 - ecld-class, 53
 - ecld.const, 56
 - sld-class, 108
- * **ecop**
 - ecop-class, 75
 - ecop.bs_implied_volatility, 76
 - ecop.bs_option_price, 77
 - ecop.opt-class, 82
 - ecop.polyfit_option, 82
- * **elliptic-curve**
 - y_slope.ecd, 113
- * **ellipticity**
 - ellipticity.ecd, 86
- * **fit**
 - ecd.fit_data, 27
 - ecd.fit_ts_conf, 28
 - ecd.read_symbol_conf, 40
 - qsld.fit, 102
- * **fixed-point**
 - ecop.find_fixed_point_lambda_by_atm_skew, 78
 - ecop.find_fixed_point_sd_by_lambda, 79
- * **gamma**
 - ecld.gamma, 57
- * **integrate**
 - integrate_pdf.ecd, 87
- * **lamp**
 - lamp-class, 91
- * **mgf**
 - ecld.imgf, 58
 - ecld.imnt, 59
- * **moments**
 - k2mnt, 89
- * **moment**
 - ecld.mgf_term, 61
 - ecld.moment, 62
 - moment.ecd, 99
- * **ogf**
 - ecld.ivol_ogf_star, 60
 - ecld.ogf, 64
 - ecld.ogf_star, 65
 - ecld.op_V, 67
- * **option-pricing**
 - ecld.mu_D, 63
- * **option**
 - ecd.imgf, 29
 - ecd.ogf, 36
- * **pdf**
 - ecd.pdf, 37
 - ecld.pdf, 68
 - integrate_pdf.ecd, 87
- * **plot**
 - lamp.plot_sim4, 92
 - plot_2x2.ecd, 101
- * **polynomial**
 - solve.ecd, 110
- * **quartic**
 - ecld.quartic_Qp_atm_attr, 70
- * **sample-data**
 - ecd.data, 22
 - ecd.data_stats, 23
 - ecd.df2ts, 24
 - ecd.fit_data, 27
 - ecd.read_symbol_conf, 40
 - ecd.ts_lag_stats, 44
- * **sample**
 - lamp.qsl_fit_config, 93
- * **sged**
 - ecld.sged_const, 72
- * **simulation**
 - lamp.generate_tau, 92
 - lamp.sd_factor, 94
 - lamp.simulate1, 95
 - lamp.simulate_iter, 96
 - lamp.stable_rnd_walk, 97
- * **solve**
 - ecd.rational, 39
 - ecd.y0_isomorphic, 45
 - ecld.solve, 73
 - solve.ecd, 110
 - solve_sym.ecd, 110
 - solve_trig.ecd, 111
- * **stable-Lambda**
 - dsl, 8
 - qsl_kurtosis_analytic, 103
- * **statistics**
 - ecd.asymp_stats, 17
 - ecd.data_stats, 23
 - ecd.stats, 43
 - ecd.ts_lag_stats, 44
 - ecld.sd, 71
 - sld.sd, 109
- * **stats**
 - discr.ecd, 8
 - ecd.sd, 41
 - jinv.ecd, 89
- * **structure**
 - ecop.term_master_calculator, 84
 - ecop.term_plot_3x3, 85
 - ecop.vix_plot_3x3, 86

* **term**
 ecop.term_master_calculator, 84
 ecop.term_plot_3x3, 85
 ecop.vix_plot_3x3, 86

* **timeseries**
 ecd.data, 22
 ecd.df2ts, 24
 ecd.fit_ts_conf, 28

* **utility**
 ecd.diff, 25
 ecd.erfq, 26
 ecd.integrate, 30
 ecd.lag, 31
 ecd.max_kurtosis, 32
 ecd.mp2f, 33
 ecd.mpfr, 34
 ecd.mpfr_qagi, 35
 ecd.mpnum, 36
 ecd.uniroot, 45
 eclib.mpnum, 62

* **xts**
 ecd.data, 22
 ecd.df2ts, 24

* **y_slope**
 ecd.y_slope, 74

bootstrap, 47
 bootstrap(bootstrap.ecdb), 6
 bootstrap.ecdb-method (bootstrap.ecdb),
 6
 bootstrap.ecdb, 6

cfqsl(dsl), 8
 cfsl(dsl), 8
 cfstablecnt(dstablecnt), 11
 cfstdlap(dstdlap), 12

dec, 7
 discr(discr.ecd), 8
 discr.ecd-method (discr.ecd), 8
 discr.ecd, 8
 dlaplace0(rlaplace0), 106
 dqsl(dsl), 8
 dsl, 6, 8, 108
 dstablecnt, 10, 11, 108
 dstdlap, 10, 12, 108
 dstdlap_poly(dstdlap), 12

ecd, 14
 ecd-class, 15
 ecd-package, 6
 ecd.adj2gamma(ecd.adj_gamma), 16
 ecd.adj_gamma, 16

ecd.asymp_kurtosis(ecd.asymp_stats), 17
 ecd.asymp_stats, 17, 43
 ecd.ccdf, 17
 ecd.cdf, 18
 ecd.cubic, 19
 ecd.cusp, 20
 ecd.cusp_a2r, 21
 ecd.cusp_r2a(ecd.cusp_a2r), 21
 ecd.cusp_std_cf(ecd.cusp_std_moment),
 21
 ecd.cusp_std_mgf(ecd.cusp_std_moment),
 21
 ecd.cusp_std_moment, 21
 ecd.data, 22
 ecd.data_stats, 23
 ecd.dawson(ecd.mpfr), 34
 ecd.devel(ecd.mpfr), 34
 ecd.df2ts, 24
 ecd.diff, 25
 ecd.erf(ecd.mpfr), 34
 ecd.erfc(ecd.mpfr), 34
 ecd.erfcx(ecd.mpfr), 34
 ecd.erfi(ecd.mpfr), 34
 ecd.erfq, 26
 ecd.erfq_sum(ecd.erfq), 26
 ecd.estimate_const, 26
 ecd.fit_data, 27
 ecd.fit_ts_conf, 28
 ecd.gamma(ecd.mpfr), 34
 ecd.has_quantile, 29
 ecd.ifelse(ecd.mpnum), 36
 ecd.imgf, 29
 ecd.integrate, 30
 ecd.kurt(ecd.sd), 41
 ecd.kurtosis(ecd.sd), 41
 ecd.lag, 31
 ecd.manage_hist_tails, 32
 ecd.max_kurtosis, 32
 ecd.mcsapply(ecd.mpnum), 36
 ecd.mean(ecd.sd), 41
 ecd.mp1(ecd.mpfr), 34
 ecd.mp2f, 33
 ecd.mpfr, 34
 ecd.mpfr_qagi, 35
 ecd.mpnum, 36
 ecd.mppi(ecd.mpfr), 34
 ecd.mu_D(ecd.imgf), 29
 ecd.ogf, 36
 ecd.pdf, 37
 ecd.polar, 38
 ecd.rational, 39
 ecd.read_csv_by_symbol, 39

ecd.read_symbol_conf, 40
 ecd.sapply (ecd.mpnum), 36
 ecd.sd, 41
 ecd.setup_const, 41
 ecd.skewness (ecd.sd), 41
 ecd.solve_cusp_asym, 42
 ecd.stats, 43
 ecd.toString, 43
 ecd.ts_lag_stats, 44
 ecd.uniroot, 45
 ecd.var (ecd.sd), 41
 ecd.y0_isomorphic, 45
 ecdattr, 46
 ecdattr-class, 47
 ecdattr.enrich, 47, 47
 ecdattr.pairs, 47, 48
 ecdattr.pairs_polar, 48
 ecdb, 49
 ecdb-class, 49
 ecdb.dbSendQuery, 50
 ecdb.protectiveCommit, 50
 ecdq, 51
 ecdq-class, 51
 ecld, 52
 ecld-class, 53
 ecld.ccdf (ecld.cdf), 55
 ecld.cdf, 55
 ecld.cdf_gamma (ecld.cdf), 55
 ecld.cdf_integrate (ecld.cdf), 55
 ecld.const, 56
 ecld.fixed_point_atm_ki (ecld.op_Q), 66
 ecld.fixed_point_atm_Q_left
 (ecld.op_Q), 66
 ecld.fixed_point_atm_Q_right
 (ecld.op_Q), 66
 ecld.fixed_point_shift (ecld.op_Q), 66
 ecld.fixed_point_SN0_atm_ki, 56
 ecld.fixed_point_SN0_atm_ki_sd
 (ecld.fixed_point_SN0_atm_ki),
 56
 ecld.fixed_point_SN0_lambda_skew_ratio
 (ecld.fixed_point_SN0_atm_ki),
 56
 ecld.fixed_point_SN0_rho_sd
 (ecld.fixed_point_SN0_atm_ki),
 56
 ecld.fixed_point_SN0_skew
 (ecld.fixed_point_SN0_atm_ki),
 56
 ecld.gamma, 57
 ecld.gamma_2F0 (ecld.gamma), 57
 ecld.gamma_hgeo (ecld.gamma), 57
 ecld.ifelse (ecld.mpnum), 62
 ecld.imgf, 58
 ecld.imgf_gamma (ecld.imgf), 58
 ecld.imgf_integrate (ecld.imgf), 58
 ecld.imgf_quartic (ecld.imgf), 58
 ecld.imnt, 59
 ecld.imnt_integrate (ecld.imnt), 59
 ecld.imnt_sum (ecld.imnt), 59
 ecld.ivol_ogf_star, 60
 ecld.kurt (ecld.sd), 71
 ecld.kurtosis (ecld.sd), 71
 ecld.laplace_B (ecld.solve), 73
 ecld.mclapply (ecld.mpnum), 62
 ecld.mean (ecld.sd), 71
 ecld.mgf (ecld.moment), 62
 ecld.mgf_by_sum (ecld.moment), 62
 ecld.mgf_diterm (ecld.mgf_term), 61
 ecld.mgf_quartic (ecld.moment), 62
 ecld.mgf_term, 61
 ecld.mgf_term_original (ecld.mgf_term),
 61
 ecld.mgf_trunc (ecld.mgf_term), 61
 ecld.mgf_trunc_max_sigma
 (ecld.mgf_term), 61
 ecld.moment, 62
 ecld.mpnum, 62
 ecld.mu_D, 63
 ecld.mu_D_by_sum (ecld.mu_D), 63
 ecld.mu_D_integrate (ecld.mu_D), 63
 ecld.mu_D_quartic (ecld.mu_D), 63
 ecld.ogf, 64
 ecld.ogf_gamma (ecld.ogf), 64
 ecld.ogf_imnt_sum (ecld.ogf), 64
 ecld.ogf_integrate (ecld.ogf), 64
 ecld.ogf_log_slope (ecld.ogf), 64
 ecld.ogf_quartic (ecld.ogf), 64
 ecld.ogf_star, 65
 ecld.ogf_star_analytic (ecld.ogf_star),
 65
 ecld.ogf_star_exp (ecld.ogf_star), 65
 ecld.ogf_star_gamma_star
 (ecld.ogf_star), 65
 ecld.ogf_star_hgeo (ecld.ogf_star), 65
 ecld.op_0 (ecld.op_V), 67
 ecld.op_Q, 66
 ecld.op_Q_skew (ecld.op_Q), 66
 ecld.op_Q_skew_by_k_lm (ecld.op_Q), 66
 ecld.op_U_lag (ecld.op_V), 67
 ecld.op_V, 67
 ecld.op_VL_quartic (ecld.op_V), 67
 ecld.pdf, 68
 ecld.quartic_model_sample

ecld.quartic_Qp_atm_attr, 70
 ecld.quartic_model_sample_attr
 (ecld.quartic_Qp_atm_attr), 70
 ecld.quartic_Q (ecld.quartic_Qp), 69
 ecld.quartic_Qp, 69
 ecld.quartic_Qp_atm_attr, 70
 ecld.quartic_Qp_atm_ki
 (ecld.quartic_Qp), 69
 ecld.quartic_Qp_atm_skew
 (ecld.quartic_Qp), 69
 ecld.quartic_Qp_rho (ecld.quartic_Qp),
 69
 ecld.quartic_Qp_skew (ecld.quartic_Qp),
 69
 ecld.quartic_SN0_atm_ki, 71
 ecld.quartic_SN0_max_RNV
 (ecld.quartic_SN0_atm_ki), 71
 ecld.quartic_SN0_rho_stdev
 (ecld.quartic_SN0_atm_ki), 71
 ecld.quartic_SN0_skew
 (ecld.quartic_SN0_atm_ki), 71
 ecld.sapply (ecld.mpnum), 62
 ecld.sd, 71
 ecld.sged_cdf (ecld.sged_const), 72
 ecld.sged_const, 72
 ecld.sged_imgf (ecld.sged_const), 72
 ecld.sged_mgf (ecld.sged_const), 72
 ecld.sged_moment (ecld.sged_const), 72
 ecld.sged_ogf (ecld.sged_const), 72
 ecld.skewness (ecld.sd), 71
 ecld.solve, 73
 ecld.solve_by_poly (ecld.solve), 73
 ecld.solve_isomorphic (ecld.solve), 73
 ecld.solve_quartic (ecld.solve), 73
 ecld.var (ecld.sd), 71
 ecld.y_slope, 74
 ecld.y_slope_trunc (ecld.y_slope), 74
 ecldOrEcd-class, 75
 ecop-class, 75
 ecop.bs_call_price
 (ecop.bs_option_price), 77
 ecop.bs_implied_volatility, 76
 ecop.bs_option_price, 77
 ecop.bs_put_price
 (ecop.bs_option_price), 77
 ecop.build_opt (ecop.from_symbol_conf),
 80
 ecop.enrich_option_df
 (ecop.read_csv_by_symbol), 83
 ecop.find_fixed_point_lambda_by_atm_skew,
 78
 ecop.find_fixed_point_sd_by_lambda, 79
 ecop.from_symbol_conf, 80
 ecop.get_ld_triple, 81
 ecop.opt-class, 82
 ecop.polyfit_option, 82
 ecop.read_csv_by_symbol, 83
 ecop.read_symbol_conf
 (ecop.from_symbol_conf), 80
 ecop.smile_data_calculator
 (ecop.term_master_calculator),
 84
 ecop.term_atm
 (ecop.term_master_calculator),
 84
 ecop.term_idx_range
 (ecop.term_plot_3x3), 85
 ecop.term_master_calculator, 84
 ecop.term_plot_3x3, 85
 ecop.term_realized_days
 (ecop.term_plot_3x3), 85
 ecop.term_target_days_default
 (ecop.term_plot_3x3), 85
 ecop.vix_plot_3x3, 86
 ellipticity (ellipticity.ecd), 86
 ellipticity, ecd-method
 (ellipticity.ecd), 86
 ellipticity.ecd, 86
 history (history.ecdb), 87
 history, ecdb-method (history.ecdb), 87
 history.ecdb, 87
 integrate_pdf (integrate_pdf.ecd), 87
 integrate_pdf, ecd-method
 (integrate_pdf.ecd), 87
 integrate_pdf.ecd, 87
 jinv (jinv.ecd), 89
 jinv, ecd-method (jinv.ecd), 89
 jinv.ecd, 89
 k2mnt, 89
 kql, 109
 kql (dsl), 8
 ksl, 109
 ksl (dsl), 8
 kstablecnt (dstablecnt), 11
 kstdlap (dstdlap), 12
 lamp, 90
 lamp-class, 91
 lamp.generate_tau, 92
 lamp.plot_sim4, 92
 lamp.plot_sim6 (lamp.plot_sim4), 92

lamp.qsl_fit_config, 93
lamp.qsl_fit_config_xtable
 (lamp.qsl_fit_config), 93
lamp.qsl_fit_plot, 94
lamp.sd_factor, 94
lamp.simulate1, 95
lamp.simulate_iter, 96
lamp.stable_rnd_walk, 97
levy.dlambda, 97
levy.domain_coloring, 98
levy.dskewed, 99

mnt2k (k2mnt), 89
moment (moment.ecd), 99
moment, ecd-method (moment.ecd), 99
moment.ecd, 99

numericMpfr-class, 100

pec (dec), 7
plot_2x2 (plot_2x2.ecd), 101
plot_2x2, ecd-method (plot_2x2.ecd), 101
plot_2x2.ecd, 101
pqsl (dsl), 8
psl (dsl), 8
pstablecnt (dstablecnt), 11
pstlap (dstlap), 12

qec (dec), 7
qqsl (dsl), 8
qsl (dsl), 8
qsl_kurtosis_analytic, 103
qsl_pdf_integrand_analytic
 (qsl_kurtosis_analytic), 103
qsl_skewness_analytic
 (qsl_kurtosis_analytic), 103
qsl_std_pdf0_analytic
 (qsl_kurtosis_analytic), 103
qsl_variance_analytic
 (qsl_kurtosis_analytic), 103
qsld (sld), 107
qsld.fit, 102
qstablecnt (dstablecnt), 11
qstdlap (dstlap), 12
quantilize (quantilize.ecd), 104
quantilize, ecd-method (quantilize.ecd),
 104
quantilize.ecd, 104

read (read.ecdb), 105
read, ecdb-method (read.ecdb), 105
read.ecdb, 105
rec (dec), 7

rlaplace0, 106
rqsl (dsl), 8
rsl (dsl), 8
rstablecnt (dstablecnt), 11
rstdlap (dstlap), 12

sld, 107
sld-class, 108
sld.kurt (sld.sd), 109
sld.kurtosis (sld.sd), 109
sld.mean (sld.sd), 109
sld.sd, 109
sld.skewness (sld.sd), 109
sld.var (sld.sd), 109
solve, ecd-method (solve.ecd), 110
solve.ecd, 110
solve_sym (solve_sym.ecd), 110
solve_sym, ecd-method (solve_sym.ecd),
 110
solve_sym.ecd, 110
solve_trig (solve_trig.ecd), 111
solve_trig, ecd-method (solve_trig.ecd),
 111
solve_trig.ecd, 111
summary (summary.ecdb), 112
summary, ecdb-method (summary.ecdb), 112
summary.ecdb, 112

write (write.ecdb), 112
write.list, ecdb-method (write.ecdb), 112
write.ecdb, 112

y_slope (y_slope.ecd), 113
y_slope, ecd-method (y_slope.ecd), 113
y_slope.ecd, 113