**Package ‘ecd’**

October 3, 2017

**Type** Package  
**Title** Elliptic Lambda Distribution and Option Pricing Model  
**Version** 0.9.1  
**Date** 2017-09-30  
**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.

**Depends** R (>= 3.3.1)  
**Imports** stats, utils, Rmpfr (>= 0.6-0), gsl, RcppFaddeeva, polynom, xts, zoo, optimx, moments, stabledist, parallel, graphics, ggplot2, gridExtra, xtable, methods, yaml, RSQLite, digest  
**Suggests** knitr, testthat, roxygen2, ghyp, fOptions, shape  
**License** Artistic-2.0  
**Encoding** latin1  
**LazyData** true  
R topics documented:

'ecop-get-ld-triple-method.R' 'ecop-plot-option-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-lihnlap-distribution-method.R' 'lamp-plot-sim4-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-simulate-1-method.R'
'lamp-stable-cnt-distribution-method.R'
'lamp-stable-lambda-dist-method.R'
'lamp-stable-rnd-walk-method.R' 'levy-dlambda-method.R'
'levy-domain-coloring-method.R'
'levy-dskewed-distribution-method.R'

RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2017-10-03 19:57:50 UTC

R topics documented:

ecd-package .......................................................... 6
bootstrap.ecdb ....................................................... 6
dec ................................................................. 7
discr.ecd ............................................................ 8
dstablecnt ............................................................ 9
edc ................................................................. 10
edc-class ............................................................. 11
edc.adj_gamma ......................................................... 11
edc.asymp_stats ....................................................... 12
edc.ccdf ............................................................ 13
edc.cdf ............................................................. 14
edc.cubic ........................................................... 15
edc.cusp ............................................................ 15
edc.cusp_a2r ........................................................ 16
edc.cusp_std_moment ............................................... 17
edc.data ............................................................. 18
edc.data_stats ....................................................... 19
edc.dff2ts .......................................................... 20
edc.diff ............................................................. 21
edc.erfq ............................................................. 21
edc.estimate_const .................................................. 22
edc.fit_data ......................................................... 23
edc.fit_ts_conf .................................................... 23
R topics documented:

ecd.has_quantile ...................................................... 24
ecd.imgf ................................................................. 25
ecd.integrate ............................................................. 26
ecd.lag ................................................................. 27
ecd.manage_hist_tails ................................................. 27
ecd.max_kurtosis ....................................................... 28
ecd.mp2f ................................................................. 29
ecd.mpfr ................................................................. 29
ecd.mpfr_qagi ......................................................... 31
ecd.mpnum .............................................................. 31
ecd.ogf ................................................................. 32
ecd.pdf ................................................................. 33
ecd.polar ............................................................... 34
ecd.rational ............................................................ 35
ecd.read_csv_by_symbol ............................................... 35
ecd.read_symbol_conf .................................................. 36
ecd.sd ................................................................. 37
ecd.setup_const .......................................................... 38
ecd.solve_cusp_asym .................................................. 38
ecd.stats ............................................................... 39
ecd.toString ............................................................ 40
ecd.ts_lag_stats ....................................................... 40
ecd.uniroot ............................................................ 41
ecd.y0_isomorphic ..................................................... 42
ecdattr ................................................................. 42
ecdattr-class ........................................................... 43
ecdattr.enrich .......................................................... 44
ecdattr.pairs ............................................................ 44
ecdattr.pairs_polar .................................................... 45
ecdb ................................................................. 45
ecdb-class ............................................................. 46
ecdb.dbSendQuery ...................................................... 46
ecd.db.protectiveCommit ............................................. 47
ecdq ................................................................. 47
ecdq-class ............................................................. 48
ecdl ................................................................. 49
ecdl-class ............................................................. 50
ecdl.cdf ............................................................... 51
ecdl.const ............................................................. 52
ecdl.fixed_point_SN0_atm_ki ........................................ 52
ecdl.gamma ............................................................. 52
ecdl.imgf .............................................................. 53
ecdl.imnt .............................................................. 54
ecdl.ivol_ogf_star ..................................................... 55
ecdl.mgf_term .......................................................... 56
ecdl.mgf ............................................................... 57
ecdl.moment ........................................................... 58
ecdl.mpnum ............................................................ 59
ecdl.mu_D .............................................................. 60
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecld.ogf</td>
<td>60</td>
</tr>
<tr>
<td>ecld.ogf_star</td>
<td>61</td>
</tr>
<tr>
<td>ecld.op_Q</td>
<td>62</td>
</tr>
<tr>
<td>ecld.op_V</td>
<td>63</td>
</tr>
<tr>
<td>ecld.pdf</td>
<td>65</td>
</tr>
<tr>
<td>ecld.quartic_Qp</td>
<td>65</td>
</tr>
<tr>
<td>ecld.quartic_Qp_atm_attr</td>
<td>67</td>
</tr>
<tr>
<td>ecld.quartic_SN0_atm_ki</td>
<td>68</td>
</tr>
<tr>
<td>ecld.sd</td>
<td>68</td>
</tr>
<tr>
<td>ecld.sged_const</td>
<td>69</td>
</tr>
<tr>
<td>ecld.solve</td>
<td>70</td>
</tr>
<tr>
<td>ecld.y_slope</td>
<td>71</td>
</tr>
<tr>
<td>ecldOrEcd-class</td>
<td>72</td>
</tr>
<tr>
<td>ecop-class</td>
<td>72</td>
</tr>
<tr>
<td>ecop.bs_implied_volatility</td>
<td>73</td>
</tr>
<tr>
<td>ecop.bs_option_price</td>
<td>74</td>
</tr>
<tr>
<td>ecop.find_fixed_point_lambda_by_atm_skew</td>
<td>75</td>
</tr>
<tr>
<td>ecop.find_fixed_point_sd_by_lambda</td>
<td>76</td>
</tr>
<tr>
<td>ecop.from_symbol_conf</td>
<td>77</td>
</tr>
<tr>
<td>ecop.get_ld_triple</td>
<td>78</td>
</tr>
<tr>
<td>ecop.opt-class</td>
<td>79</td>
</tr>
<tr>
<td>ecop.plot_option</td>
<td>80</td>
</tr>
<tr>
<td>ecop.polyfit_option</td>
<td>81</td>
</tr>
<tr>
<td>ecop.read_csv_by_symbol</td>
<td>81</td>
</tr>
<tr>
<td>ecop.term_master_calculator</td>
<td>82</td>
</tr>
<tr>
<td>ecop.term_plot_3x3</td>
<td>83</td>
</tr>
<tr>
<td>ecop.vix_plot_3x3</td>
<td>84</td>
</tr>
<tr>
<td>ellipticity.ecd</td>
<td>84</td>
</tr>
<tr>
<td>history.ecdb</td>
<td>85</td>
</tr>
<tr>
<td>integrate_pdf.ecd</td>
<td>86</td>
</tr>
<tr>
<td>jinv.ecd</td>
<td>87</td>
</tr>
<tr>
<td>k2mnt</td>
<td>87</td>
</tr>
<tr>
<td>lamp</td>
<td>88</td>
</tr>
<tr>
<td>lamp-class</td>
<td>89</td>
</tr>
<tr>
<td>lamp.generate_tau</td>
<td>90</td>
</tr>
<tr>
<td>lamp.plot_sim4</td>
<td>90</td>
</tr>
<tr>
<td>lamp.qsl_fit_config</td>
<td>91</td>
</tr>
<tr>
<td>lamp.qsl_fit_plot</td>
<td>92</td>
</tr>
<tr>
<td>lamp.sd_factor</td>
<td>92</td>
</tr>
<tr>
<td>lamp.simulate1</td>
<td>93</td>
</tr>
<tr>
<td>lamp.simulate_iter</td>
<td>94</td>
</tr>
<tr>
<td>lamp.stable_rnd_walk</td>
<td>95</td>
</tr>
<tr>
<td>levy.dlambda</td>
<td>95</td>
</tr>
<tr>
<td>levy.domain_coloring</td>
<td>96</td>
</tr>
<tr>
<td>levy.dskewed</td>
<td>97</td>
</tr>
<tr>
<td>moment.ecd</td>
<td>98</td>
</tr>
<tr>
<td>numericMpfr-class</td>
<td>99</td>
</tr>
<tr>
<td>plot_2x2.ecd</td>
<td>99</td>
</tr>
</tbody>
</table>
ecd-package

ecd: A package for the elliptic distribution.

Description

The ecd package provides the core class and functions to calculate the elliptic distribution. They are either generic or use ecd namespace. The lambda distribution is using ecld namespace. SGED is considered part of ecld. The option pricing API is using ecop namespace.

Author(s)

Stephen H-T. Lihn

bootstrap.ecdb

Bootstrap data for the Elliptic DB (ECDB)

Description

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

Usage

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

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

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

*Arguments*

- **object**: an object of ecdb class.
- **action**: the action operating on the ecdb.
- **skip.existing**: logical, if TRUE (default), skip if action already done in history.

*Value*

Row count.

---

**dec**

*The Elliptic Distribution*

---

**Description**

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

**Usage**

```r
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 ecdb 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)
de(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

```r
## 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

- `object` an object of ecd class
- `no.validate` logical, if TRUE, don’t validate presence of beta. Default is FALSE.

Value

the discriminant

Author(s)

Stephen H-T. Lihn

Examples

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

Implements some aspects of 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.

Author(s)

Stephen H-T. Lihn
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

<table>
<thead>
<tr>
<th>alpha</th>
<th>numeric, the flatness parameter. Default: 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>gamma</td>
<td>numeric, the sharpness parameter. Default: 0.</td>
</tr>
<tr>
<td>sigma</td>
<td>numeric, the scale parameter. Must be positive. Default: 1.</td>
</tr>
<tr>
<td>beta</td>
<td>numeric, the skewness parameter. Default: 0.</td>
</tr>
<tr>
<td>mu</td>
<td>numeric, the location parameter. Default: 0.</td>
</tr>
<tr>
<td>cusp</td>
<td>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.</td>
</tr>
<tr>
<td>lambda</td>
<td>numeric, the leading exponent for the special model. Default: 3.</td>
</tr>
<tr>
<td>with.stats</td>
<td>logical, also calculate statistics, default is TRUE.</td>
</tr>
<tr>
<td>with.quantile</td>
<td>logical, also calculate quantile data, default is FALSE.</td>
</tr>
<tr>
<td>bare.bone</td>
<td>logical, skip both const and stats calculation, default is FALSE. This for debug purpose for issues on integrating $e^y(x)$.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical, display timing information, for debugging purpose, default is FALSE.</td>
</tr>
</tbody>
</table>

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 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, codelong_name, short_name.skew, short_name. This slot doesn’t have formal use yet.

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

edc.adj.gamma(gamma)

edc.adj2gamma(adj.gamma)
ecd.asymp_stats

Arguments

- gamma: numeric, the gamma parameter
- adj.gamma: numeric, the discriminant-adjusted gamma

Value

- adjusted gamma (or the reverse of adjustment)

Examples

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

ecd.asymp.stats

*Compute asymptotic statistics of an ecd object*

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

```r
ecd.asymp.stats(object, q)
ecd.asymp.kurtosis(object, q)
```

Arguments

- object: an object of ecd class with quantile
- q: numeric vector of quantiles

Value

- a list of stats list, or a vector of kurtosis

Examples

```r
# 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)
```
ecd.ccdf

Complementary CDF of ecd

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)
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**

```r
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**

```r
d <- ecd()
ed.cubic(d)
ed.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**

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

alpha numeric, the flatness parameter. Default: NaN.
gamma numeric, the sharpness parameter. Default: NaN.
sigma numeric, the scale parameter. Must be positive. Default 1.
mu numeric, the location parameter. Default: 0.
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 $e^\gamma(x)$.
verbose 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)

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 outliners 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.
**ecd.data_stats**

**Examples**

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

---

**ecd.data_stats**  
*Statistics and histogram on log returns*

**Description**

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

**Usage**

```r
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**

```r
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 require 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**
```r
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 enencaptulates both scaled \( \text{erfi} \) and \( \text{erfc} \) functions into a single representation. This is used to provide an elegant expression for the MGF and local option prices, \( \text{L}_{c,p} \). When \( \text{sgn} = -1 \), it is \( \sqrt{\pi} e^{-x^2} \text{erfi}(x) \), which twice of Dawson function. When \( \text{sgn} = 1 \), it is \( \sqrt{\pi} e^{x^2} \text{erfc}(x) \). 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)
ecd.estimate_const

Arguments

- x numeric
- sgn an integer of 1 or -1

Value

The mpfr object

Examples

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

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

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

Arguments

- object An object of ecd class

Value

numeric, estimated const

Examples

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

- **symbol** Character. The symbol of sample data. Default: dji.
- **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
- **conf_file** File name for symbol config, default to conf/ecd-fit-conf.yml
- **eps_file** File name for eps output
- **qa.fit** Logical, qa the standard fit_fn once.

Value

Final ecd object

Examples

```r
## 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)
ecd.has_quantile

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

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

ecd.has_quantile  Whether the ecd object has quantile data or not

Description

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

Usage

ecd.has_quantile(object)

Arguments

- **object**: an object of ecd class

Value

logical, whether the object has quantile data or not.

Author(s)

Stephen H-T. Lihn
**ecd.imgf**

**Incomplete MGF of ecd**

**Description**

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

**Usage**

```r
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 $-\text{Inf}$
- **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**

```r
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.
...   Additional 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
- htu: list, input histuple
- merge_tails: length-two numeric vector, points to be merged for left and right tails

Value
list, histuple
ecd.max_kurtosis

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
jinv 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
```r
## Not run:
k <- ecd.max_kurtosis()
appe <- 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 ecd.gamma, which is the incomplete gamma function. ecd.erf is a wrapper on Rmpfr::erf; or RcppFaddeeva::erf when it's complex. ecd.erfcx is a wrapper on Rmpfr::erfcx; or RcppFaddeeva::erfcx when it's complex. ecd.erfc is a wrapper on Rmpfr::erfc; or RcppFaddeeva::erfc when it's complex. This is used to implement ecd.erfq. ecd.dawson is a wrapper on gsl::dawson; or RcppFaddeeva::Dawson when it's complex. Dawson function is used to implement ecd.erfq. ecd.erfi is the imaginary scaled error function, which is implemented through ecd.dawson. Or RcppFaddeeva::erfi when it's complex. 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 $+/-\text{Inf}$; (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

- object: an object of ecd class
- 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.
- ...: additional arguments for $f$.
- abs.tol: numeric, the suggested absolute tolerance.
- show.warning: logical, to suppress warnings or not.

Value

The integrate object

Author(s)

Stephen H. Lihn

ecd.mpnum

Wrappers for ecd 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
ecd.mprnum(object, x)
edc.ifelse(object, test, yes, no)
edc.sapply(object, x, FUN, ...)
edc.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

Option generating function of ecd

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

Usage
edc.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.
Description

Calculate the PDF of an ecd object

Usage

ecd.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

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)
ecd.rational

Utility to convert a numeric to a rational

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

  x         numeric
  pref.denominator
             numeric, a list of preferred integer denominators to conform to, default is numeric(0).
  cycles    numeric, maximum number of steps, default is 10.
  max.denominator
             numeric, maximum denominator when the loop of trial should stop, default is 500.
  as.character 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)

cd.read_csv_by_symbol

Read csv file of sample data

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.
ecd.read_symbol_conf

**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  

**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**

```r
## Not run:
conf <- ecd.read_symbol_conf("dji")
## End(Not run)
```
Standard deviation, variance, mean, skewness, and kurtosis of ecd

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)
edc.sd(d)
edc.var(d)
edc.mean(d)
edc.skewness(d)
edc.kurt(d)
### ecd.solve_cusp_asym

**Description**

The simplified trigonometric solution for \( x^2 = -y^3 - \beta x^* y \)

**Usage**

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

**Arguments**

- `x`: Array of x dimension
- `beta`: the skew parameter

---

### ecd.setup_const

**Integration preprocessor for an ecd object**

**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**

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

**Arguments**

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

**Value**

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

**Author(s)**

Stephen H. Lihn

**Examples**

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

Value

Array of y

Examples

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

d <- ecd.stats(d)

ecd.stats

**Compute statistics of an ecd object**

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

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

String representation of ecd

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

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

## End(Not run)
```

---

**ecd.unirroot**  
*Uniroot wrapper*

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

uniroot result

**Author(s)**

Stephen H. Lihn
ecd.y0_isomorphic

The analytic solution of $y(0)$ via isomorphic mapping.

**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**

```r
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

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

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  Enrich a basic ecdattr object

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

p 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

alpha, gamma numeric vectors
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

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 $rL\theta$. If $cusp$ is 1, data points are on the critical line specified by $r$.

**Usage**

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

**Arguments**

- **R**, **theta**: numeric vectors
- **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**

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 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. Special 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**

```r
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
**ecdb.protectiveCommit**

**Value**

a result set object

**Author(s)**

Stephen H-T. Lihn

---

**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

---

**ecdq**

*Constructor of ecdq class*

---

**Description**

Construct an ecdq class by providing the required parameters.

**Usage**

ecdq(ecd, verbose = FALSE)

**Arguments**

ecd An object of ecd class

verbose logical, display timing information, for debugging purpose.
Value

An object of ecdq class

Author(s)

Stephen H. Lihn

Examples

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

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.
ecld Constructor of ecld class

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 vanila 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

- **lambda**: numeric, the lambda parameter. Must be positive. Default: 3.
- **sigma**: numeric, the scale parameter. Must be positive. Default: 1.
- **beta**: numeric, the skewness parameter. Default: 0.
- **mu**: numeric, the location parameter. Default: 0.
- **epsilon**: 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_0. 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)

ecld-class  An S4 class to represent the lambda distribution

Description
The ecld class serves as an object-oriented interface for the lambda distribution. The ecld prefix will also be 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 ecld 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

**Author(s)**
Stephen H. Lihn

---

**ecld.cdf**

*CDF and CCDF of ecld*

**Description**

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

**Usage**

```r
ecld.cdf(object, x)
ecld.ccdf(object, x)
ecld.cdf_integrate(object, x)
ecld.cdf_gamma(object, x)
```

**Arguments**

- `object` an object of ecld class
- `x` a numeric vector of x

**Value**

The CDF or CCDF vector

**Author(s)**
Stephen H. Lihn

**Examples**

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

Analytic solution of the normalization constant for lambda distribution

Description
The normalization constant $C$. SGED is supported.

Usage
eclld.const(object)

Arguments

object an object of eclld class

Value
numeric

Author(s)
Stephen H. Lihn

Examples

ld <- eclld(3)
eclld.const(ld)

eclld.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/stddev, ATM skew of $Q_c$, and the ratio of lambda to ATM skew under the RNO measure in the fixed point model.

Usage
eclld.fixed_point_SN0_atm_ki(lambda)
eclld.fixed_point_SN0_rhoe_sd(lambda)
eclld.fixed_point_SN0_atm_ki_sd()
eclld.fixed_point_SN0_skew(lambda, atm_ki = NULL)
eclld.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, $e^{-x} x^{s-1} \text{hyper} F_0(1, 1-s; -1/x)$. It is mainly used in for star OGF $L^*(k; \lambda)$. ecld.gamma_2F0 is simply $\text{hyper} F_0(1, 1-s; -1/x)$, 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
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

```r
ecl.d.imgf(object, k, otype = "c", RN = TRUE)
ecl.d.imgf_quartic(object, k, otype = "c", RN = TRUE)
ecl.d.imgf_gamma(object, k, otype = "c", RN = TRUE)
ecl.d.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

```r
ld <- ecld(sigma=0.01)
ecl.d.imgf(ld, 0)
```
### eclud.imnt

**Incomplete moment (imnt) of eclud**

**Description**

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

**Usage**

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

**Arguments**

- **object**
  - an object of eclud class
- **ki**
  - numeric vector of normalized log-strike, \((k-mu)/\sigma\)
- **order**
  - numeric. Order of the moment to be computed. For `eclud.imnt_sum`, this is the maximum order to be truncated. For small \(\sigma\) at \(\lambda=3\), this can be simply 2. If \(\text{Inf}\), 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**

```r
ld <- eclud(sigma=0.01*ecd.mp1)
ki <- seq(-0.1, 0.1, by=0.01)
eclud.imnt(ld, ki, 1)
```
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 \( \sigma_1 \).

Author(s)

Stephen H-T. Lihn

Examples

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

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

Usage

ecl.d.mgf_term(object, order, t = 1)
ecl.d.mgf_term_original(object, order, t = 1)
ecl.d.mgf_diterm(object, order, t = 1)
ecl.d.mgf_trunc(object, t = 1)
ecl.d.mgf_trunc_max_sigma(object, order = 1)

Arguments

- object: an object of ecd class
- order: numeric. Order of the term (moment). Order can be a vector.
- t: numeric, for MGF

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

ld <- ecld(3, sigma=0.01*ecd.mp1)
ecl.d.mgf_trunc(ld)
The moments and MGF of ecd

Description

Compute the moments and MGF of ecd for μ=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

object an object of ecd class
order numeric, order of the moment to be computed
ignore.mu logical, disregard μ; otherwise, stop if μ is not zero.
t 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)
Wrappers for ecd 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 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-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**

```r
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**

```r
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.gof(object, k, otype = "c", RN = TRUE)
ecld.gof_quartic(object, k, otype = "c", RN = TRUE)
ecld.gof_integrate(object, k, otype = "c", RN = TRUE)
ecld.gof_gamma(object, k, otype = "c", RN = TRUE)
ecld.gof_imnt_sum(object, k, order, otype = "c", RN = TRUE)
ecld.gof_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.gof(ld, k)

ecld.gof_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.gof_star(object, ki)

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

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

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

ecld.gof_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.gof_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, -(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 \( \rho, \epsilon \)

ki           numeric, a vector of \( \sigma \)-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 nor-
malized 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 \( \times \) 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_0(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 ectl 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 ectl 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
**Description**

Calculate the PDF of an ecd object

**Usage**

```r
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**

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

---

**ecd.quartic.Qp**

*The ATM volatility and skew of \( Q_p \) in quartic model*

**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:
```
1d <- ecld.quartic(sigma=0.001*ecd.mp1, epsilon=0, rho=0, mu_plus=0)
ecld.quartic_Qp_atm_ki(1d, lower=-12, upper=-11)
ecld.quartic_Qp_atm_skew(1d, lower=-12, upper=-11)
```

## End(Not run)
ecld.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)/stdev. The output fields are: atm_ki, atm_kew, atm_vol, rho, and rho_ratio - rho/stdev.

Usage

ecld.quartic_Qp_atm_attr(df)
ecld.quartic_model_sample(dt, ttm, skew_adjusted = TRUE)
ecld.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 intercep, else use the tercep 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 + 8*ttm
mupr <- -(ecld.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)
eclld.quartic_Qp_atm_attr(df)

## End(Not run)
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 RN0 measure.

Usage

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

Arguments

sigma numeric, the volatility parameter

Value

numeric

Author(s)

Stephen H-T. Lihn

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**

```r
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

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

<table>
<thead>
<tr>
<th>ecld.y_slope</th>
<th><em>Analytic solution for the slope of ( y(x) ) in lambda distribution</em></th>
</tr>
</thead>
</table>

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

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

object an object of ecld 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.
ecop.bs_implied_volatility

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 $\sigma_{BS}$ 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 $\sigma_{BS}$.

Examples

```r
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%
```

date: 2023-02-20

Description

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

Usage

```r
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

- `ivol` numeric vector of implied volatility
- `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, c or p. Default is c.

Value

The call/put prices
Examples

```r
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)
```

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

```r
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.
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 encapulates 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

Description

Read conf for option sample data and fitting parameters

Usage

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

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

```r
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

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

## End(Not run)
ecop.get_ld_triple  

Get triple list of ecld objects by stdev

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

- lambda: numeric, the lambda parameter. Must be positive. Default: 3.
- sd: numeric, the stdev parameter. Must be positive. Default: 1.
- beta: numeric, the skewness parameter. Default: 0.
- mu_plus_ratio: numeric, numeric, excess value in addition to mu_D, relative to the stdev. Default: 0.
- epsilon_ratio: numeric, epsilon ratio relative to the stdev. Default: 0.
- atm_imp_k: 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, -(atm_imp_k - mu). Default: NaN. the rho slot in ld1 is populated with the value.
- fn_shift: function, takes an ecld object and return the fixed point shift, -(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 atm_imp_k 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.plot_option  
__Plot option chain charts using conf from option sample data__

**Description**

This utility produces standardized plots of 3. The first plot is the option state price and fits. The second plot is the log-slope of option state prices and fits. The third plot is the implied volatility and fits.

**Usage**

```
ecop.plot_option(object, otype, simulate = TRUE, do.polyfit = TRUE)
```

**Arguments**

- **object**: an ecop object with conf
- **otype**: option type
- **simulate**: logical, if TRUE, simulate according to lambda transformation and lambda distribution.
- **do.polyfit**: logical, if TRUE, use polyfit to enhance the resolution on the peak of the log-slope curve. In some cases, there aren’t enough data points for polyfit, use this parameter to turn the feature off.

**Value**

The ecop.opt object

**Author(s)**

Stephen H-T. Lihn

**Examples**

```r
## Not run:
op <- ecop.from_symbol_conf("spx2_1d")
par(mfcol=c(3,2))
ecop.plot_option(op, otype="c")
ecop.plot_option(op, otype="p")
## End(Not run)
```
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")

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 calcuate teh 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

date_str  character in the form of YYYY-MM-DD
option_data  dataframe, read from ecop.read_csv_by_symbol
result  dataframe, the VIX optimx result
result_avg  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

object  An object of ecd class

tol  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)

Description

List of unique history reflecting the bootstrap activities.

Usage

```r
## S3 method for class 'ecd'
history(object)

history(object)

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

Arguments

object  an object of ecd class.

Value

list of history

Author(s)

Stephen H-T. Lihn
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

object
An object of ecd class

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.

... Additional arguments for f.

show.warning
logical, display warning messages.

verbose
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)
### jinv.ecd

**J-invariant of the elliptic curve** \(y(x)\)

**Description**

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

**Usage**

```r
## 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**
- `object`: an object of `ecd` class
- `no.validate`: logical, if TRUE, don’t validate presence of \(beta\). Default is FALSE.

**Value**

the j-invariant

**Author(s)**

Stephen H-T. Lihn

**Examples**

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

### k2mnt

**Conversion between cumulants and moments**

**Description**

Implements conversion between the first four cumulants and moments

**Usage**

```r
k2mnt(k)
mnt2k(m)
```
Arguments

- \( k \) numeric, first four cumulants.
- \( m \) 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

```r
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

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

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  Generate tau from stable distribution

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)
lpl <- lamp.generate_tau(lp)
lpl@tau

lamp.plot_sim4  Plot the simulation result in standard layout

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

---

**Description**

Read QLD 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.sd_factor

Calculate sd adjustment factor

Description

Calculate sd adjustment factor. For L2 random walk, it is the power of \(\frac{1}{1+\alpha/2}\). 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)
lamp.simulate1

**Arguments**

object  
an object of lamp class

**Value**

numeric, the sd factor

**Author(s)**

Stephen H-T. Lihn

---

**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

object
  an object of lamp class

use.mc
  numeric, number of cores for parallel simulations. Default is 4.

sim.length
  numeric, number of Z to simulate. Default is 1000.

reset.cache
  logical, to reset simulation cache or not prior the run. Default is FALSE.

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

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

plot.util
  function, interface to plot simulation results. Default is lamp.plot_sim4.

cpu.temperature
  numeric, temperature above which is overhead. Default is 68.

cpu.temperature.util
  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: 1. Laplace(0,1). No skewess. 2. Experimental Laplace random walk via Gauss-Laplace transmutation. 22. Normal distribution N(0, sqrt(n))*epsilon. No skewess. 2. Binomial random walk, b*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

\[
\begin{align*}
x &= \text{seq}(1, 10) \\
y &= \text{levy.dlambda}(x)
\end{align*}
\]

levy.domain_coloring  Domain coloring of Laplace kernel of lambda distribution

Description

Domain coloring on the complex plane of Laplace kernel of lambda distribution, \( \exp(\text{i}tx) \ P(x) \), 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

\[
\text{levy.domain_coloring}(t, \text{rec}, n = 200, \lambda = 4)
\]

Arguments

- \( t \): numeric or complex
- \( \text{rec} \): numeric, define the rectangle of plot in the order of \((x_1, x_2, y_1, y_2)\)
- \( n \): 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 \text{grid.arrange}()

Author(s)

Stephen H. Lihn
### Examples

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

### 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

```r
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

```r
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 \(-\text{Inf}\) and \text{Inf}. The \text{asymp.lower} and \text{asymp.upper} parameters are used for asymptotic statistics, to study the effect of finite observations.

Usage
```r
## S3 method for class 'ecd'
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.
- **asymp.lower**: numeric, lower bound for asymptotic statistics, default: \(-\text{Inf}\).
- **asymp.upper**: numeric, upper bound for asymptotic statistics, default: \text{Inf}.
- **verbose**: logical, display timing information, for debugging purpose.

Value
Numeric. The moment.

Author(s)
Stephen H. Lihn

Examples
```r
d <- ecd()
moment(d, 2)
```
**numericMpfr-class**

**The numericMpfr class**

**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**

**Standard 2x2 plot for sample data**

**Description**

Standard 2x2 plot for sample data

**Usage**

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

```r
## 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**

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

```r
## End(Not run)
```
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

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

```r
## 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

```r
rlihnlap(n, b = 1)
dlaplace0(x, b = 1)
```

Arguments

- `n` numeric, number of observations.
- `b` numeric, the scale parameter, where the variance is $2b^2$.
- `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

---

**rlihnlap**

*Liht-Laplace process and distribution*

Description

Implements some aspects of Liht-Laplace process

Usage

```r
rlihnlap(n, t = 1, convo = 1, beta = 0, mu = 0)
dlihnlap(x, t = 1, convo = 1, beta = 0, mu = 0)
cflihnlap(s, t = 1, convo = 1, beta = 0, mu = 0)
klihnlap(t = 1, convo = 1, beta = 0, mu = 0)
dlihnlap_poly(x, t = 1, convo = 1, beta = 0, mu = 0)
```

Arguments

- `n` numeric, number of observations.
- `t` numeric, the time parameter, of which the variance is $t$.
- `convo` numeric, the convolution number, default is 1, which is without convolution.
- `beta` numeric, skewness parameter according to skewed lambda distribution, default is 0.
\textit{rqsl}

\textbf{Description}

Implements some aspects of the stable lambda (SL) distribution

\textbf{Usage}

\begin{verbatim}
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)
dsl(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)
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)
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)
\end{verbatim}
Arguments

- **n**: numeric, number of observations.
- **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.
- **x**: numeric, vector of responses.
- **nu**: numeric, vector of nu in the pdf integrand, starting from 0 (not nu0).
- **s**: numeric, vector of responses for characteristic function.
- **method**: character, method of characteristic function (CF) calculation. Default is "a". Method a uses cflihnlap x dstablecnt. Method b uses dlihnlap 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.

Author(s)

Stephen H-T. Lihn
**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**

```r
## 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**

```r
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**

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

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

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

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

Description

Summary for the Elliptic DB (ECDB)

Usage

```r
## S3 method for class 'ecdb'
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

```r
summary(ecdb())
```
write.ecdb  Write API for the ecdb for a list of basic ecdattr objects

Description

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

Usage

```r
## 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  Slope of \( y(x) \)

Description

Slope of \( y(x) \), that is, \( \frac{dy}{dx} \).

Usage

```r
## 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

*Topic ATM
  ecdl.fixed_point_SN0_atm_ki, 52
  ecdl.quartic_Qp, 65
  ecdl.quartic_SN0_atm_ki, 68
*Topic Domain-Coloring
  levy.domain_coloring, 96
*Topic Laplace
  rlaplace0, 101
*Topic Lihn-Laplace
  rlihnlap, 102
*Topic Modified-Lambda
  rqsl, 103
*Topic PDF
  levy.dlambda, 95
  levy.dskewed, 97
*Topic QSLD
  lamp.qsl_fit_plot, 92
*Topic Q
  ecdl.op_q, 62
*Topic Stable
  dstablecnt, 9
*Topic analytic
  ecd.solve_cusp_asym, 38
*Topic cdf
  ecd.ccdf, 13
  ecd.cdf, 14
  ecd.imgf, 25
  ecdcdf, 51
*Topic class
  ecd-class, 11
  ecdq-class, 48
*Topic constructor
  ecd, 10
  ecd-class, 11
  ecd.cusp, 15
  ecd.estimate_const, 22
  ecd.polar, 34
  ecd.setup_const, 38
  ecd.to_string, 40
  ecdattr, 42
  ecdb, 45
  ecdb, 47
  ecd-class, 48
  ecdl, 49
  ecop.from_symbol.conf, 77
  ecop.get ld triple, 78
  lamp, 88
*Topic cubic
  ecd.cubic, 15
*Topic cusp
  ecd.cusp, 15
  ecd.cusp_a2r, 16
  ecd.cusp_std_moment, 17
*Topic datasets
  ecd.mpfr, 29
  ecop.term_plot_3x3, 83
*Topic data
  ecd.manage_hist_tails, 27
  ecd.read_csv_by_symbol, 35
  ecop.read_csv_by_symbol, 81
  lamp.qsl_fit_config, 91
*Topic discriminant
  ecd.adj_gamma, 11
*Topic distribution
  dec, 7
  ecd.has_quantile, 24
  ecd.pdf, 33
  ecd.pdf, 65
  quantize.ecd, 100
*Topic ecddat
  ecdattr, 42
  ecdattr-class, 43
  ecdattr.enrich, 44
  ecdattr.pairs, 44
  ecdattr.pairs_polar, 45
*Topic ecdb
  bootstrap.ecdb, 6
  ecd-class, 46
ecdb.dbSendQuery, 46
ecdb.protectiveCommit, 47
history.ecdb, 85
read.ecdb, 101
summary.ecdb, 107
write.ecdb, 108
*Topic ecdq
  ecdq-class, 48
*Topic ecd
  ecd.cusp, 15
ecd.cusp_std_moment, 17
ecd.polar, 34
*Topic ecdl
ecdl-class, 50
ecdl.const, 52
*Topic ecop
  ecop-class, 72
  ecop.bs IMPLIED_voLatility, 73
  ecop.bs_option_price, 74
  ecop.opt-class, 79
  ecop.polyfit_option, 81
*Topic elliptic-curve
  y_slope.ecd, 108
*Topic ellipticity
  ellipticity.ecd, 84
*Topic fit
  ecd.fit_data, 23
  ecd.fit_ts_conf, 23
  ecd.read_symbol_conf, 36
*Topic fixed-point
  ecop.find_fixed_point_lambda_by_atm_skew, 75
  ecop.find_fixed_point_sd_by_lambda, 76
*Topic gamma
ecld.gamma, 53
*Topic integrate
  integrate_pdf.ecd, 86
*Topic lamp
  lamp-class, 89
*Topic mgf
ecld.imgf, 54
ecld.imnt, 55
*Topic moments
  k2mnt, 87
*Topic moment
  ecdl.mgf_term, 57
ecld.moment, 58
moment.ecd, 98
*Topic ogf
ecld.ivol_ogf_star, 56
ecld.ogf, 60
ecld.ogf_star, 61
ecld.op_v, 63
*Topic option-pricing
ecld.mu_D, 60
*Topic option
ecld.imgf, 25
ecld.ogf, 32
*Topic pdf
ecld.pdf, 33
ecld.pdf, 65
integrate_pdf.ecd, 86
*Topic plot
eclp.plot_option, 80
lamp.plot_sim4, 90
plot_zx2.ecd, 99
*Topic polynomial
  solve.ecd, 105
*Topic quartic
  ecdl.quartic_Qp_atm_attr, 67
*Topic sample-data
ecd.data, 18
ecd.data_stats, 19
ecd.df2ts, 20
ecd.fit_data, 23
ecd.read_symbol_conf, 36
ecd.ts_lag_stats, 40
*Topic sample
  lamp.qsl_fit_config, 91
*Topic sged
ecld.sged_const, 69
*Topic simulation
  lamp.generate_tau, 90
  lamp.sd_factor, 92
  lamp.simulate1, 93
  lamp.simulate_iter, 94
  lamp.stable_rnd_walk, 95
*Topic solve
  ecd.rational, 35
ecd.y0_isomorphic, 42
ecld.solve, 70
solve.ecd, 105
solve_sym.ecd, 105
solve_trig.ecd, 106
*Topic statistics
ecd.asymp.stats, 12
edc.data_stats, 19
edc.stats, 39
edc.ts_lag_stats, 40
edcl.sd, 68

*Topic stats
  discr.ecd, 8
edc.sd, 37
jinv.ecd, 87

*Topic structure
  ecop.term_master_calculator, 82
edc.term_plot_3x3, 83
edc.vix_plot_3x3, 84

*Topic timeseries
  ecd.data, 18
edc.df2ts, 20
edc.fit_ts_conf, 23

*Topic utility
  ecd.diff, 21
edc.erf, 21
edc.integrate, 26
edc.lag, 27
edc.max_kurtosis, 28
edc.mpz, 29
edc.mpfr, 29
edc.mpfr_qagi, 31
edc.mpnum, 31
edc.uniqroot, 41
edc.mpnum, 59

*Topic xts
  ecd.data, 18
edc.df2ts, 20

*Topic y_slope
  ecd.y_slope, 71

bootstrap, 43
bootstrap (bootstrap.ecdb), 6
bootstrap.ecdb-method (bootstrap.ecdb), 6
bootstrap.ecdb, 6
cflihnlap (rlihnlap), 102
cfql (rqs1), 103
cfsl (rqs1), 103
cfstablecnt (dtablecnt), 9
dec, 7
discr (discr.ecd), 8
discr.ecd-method (discr.ecd), 8
discr.ecd, 8
dlaplace0 (rlaplace0), 101
dlihnlap (rlihnlap), 102
dlihnlap_poly (rlihnlap), 102
dqsl (rqs1), 103
dsl (rqs1), 103
dtablecnt, 9
edc, 10
edc-class, 11
edc-package, 6
edc.adj2gamma (ecd.adj.gamma), 11
edc.adj.gamma, 11
edc.asymp_kurtosis (ecd.asymp.stats), 12
edc.asymp.stats, 12, 39
edc.cccdf, 13
edc.cdf, 14
edc.cubic, 15
edc.cusp, 15
edc.cusp_a2r, 16
edc.cusp_r2a (ecd.cusp_a2r), 16
edc.cusp_std_cf (ecd.cusp_std_moment), 17
edc.cusp_std_mgf (ecd.cusp_std_moment), 17
edc.cusp_std_moment, 17
edc.data, 18
edc.data_stats, 19
edc.dawson (ecd.mpfr), 29
edc.devel (ecd.mpfr), 29
edc.df2ts, 20
edc.diff, 21
edc.erf (ecd.mpfr), 29
edc.erfc (ecd.mpfr), 29
edc.erfcx (ecd.mpfr), 29
edc.erfi (ecd.mpfr), 29
edc.erf, 21
edc.erf_q (ecd.erf), 21
edc.erf_q (ecd.erf), 21
edc.fit_data, 23
edc.fit_ts_conf, 23
edc.gamma (ecd.mpfr), 29
edc.has_quantile, 24
edc.ifelse (ecd.mpnum), 31
edc.imgf, 25
edc.integrate, 26
ecd.kurt (ecd.sd), 37
ecd.kurtosis (ecd.sd), 37
ecd.lag, 27
ecd.manage_hist_tails, 27
ecd.max_kurtosis, 28
ecd.mcsapply (ecd.mpnum), 31
ecd.mean (ecd.sd), 37
ecd.mp1 (ecd.mpfr), 29
ecd.mp2f, 29
ecd.mpfr, 29
ecd.mpfr_qagi, 31
ecd.mpnum, 31
ecd.mppi (ecd.mpfr), 29
ecd.mu_D (ecd.imgf), 25
ecd.ofg, 32
ecd.pdf, 33
ecd.polar, 34
ecd.rational, 35
ecd.read_csv_by_symbol, 35
ecd.read_symbol_conf, 36
ecd.sapply (ecd.mpnum), 31
ecd.sd, 37
ecd.setup_const, 38
ecd.skewness (ecd.sd), 37
ecd.solve_cusp_asym, 38
ecd.stats, 39
ecd.toString, 40
ecd.ts_lag_stats, 40
ecd.unirroot, 41
ecd.var (ecd.sd), 37
ecd.y0_isomorphic, 42
ecdattr, 42
ecdattr-class, 43
ecdattr.enrich, 43, 44
ecdattr.pairs, 43, 44
ecdattr.pairs_polar, 45
ecdb, 45
ecdb-class, 46
ecdb.dbSendQuery, 46
ecdb.protectionCommit, 47
ecdq, 47
ecdq-class, 48
ecd, 49
ecd-class, 50
ecd.cccdf (ecd.cdf), 51
ecd.cdf, 51
ecd.cdf_gamma (ecd.cdf), 51
ecd.cdf_integrate (ecd.cdf), 51
ecd.const, 52
ecd.fixed_point_atm_ki (ecd.op_Q), 62
ecd.fixed_point_atm_Q_left (ecd.op_Q), 62
ecd.fixed_point_atm_Q_right (ecd.op_Q), 62
ecd.fixed_point_shift (ecd.op_Q), 62
ecd.fixed_point_SN0_atm_ki, 52
ecd.fixed_point_SN0_atm_ki_sd (ecd.fixed_point_SN0_atm_ki), 52
ecd.fixed_point_SN0_lambda_skew_ratio (ecd.fixed_point_SN0_atm_ki), 52
ecd.fixed_point_SN0_rhoe_sd (ecd.fixed_point_SN0_atm_ki), 52
ecd.fixed_point_SN0_skew (ecd.fixed_point_SN0_atm_ki), 52
ecd.gamma, 53
ecd.gamma_2F0 (ecd.gamma), 53
ecd.gamma_hgeo (ecd.gamma), 53
ecd.ifelse (ecd.mpnum), 59
ecd.imgf, 54
ecd.imgf.gamma (ecd.imgf), 54
ecd.imgf_integrate (ecd.imgf), 54
ecd.imgf_quartic (ecd.imgf), 54
ecd.imnt, 55
ecd.imnt_integrate (ecd.imnt), 55
ecd.imnt_sum (ecd.imnt), 55
ecd.ivol_ogf_star, 56
ecd.kurt (ecd.sd), 68
ecd.kurtosis (ecd.sd), 68
ecd.laplace_B (ecd.solve), 70
ecd.mclapply (ecd.mpnum), 59
ecd.mean (ecd.sd), 68
ecd.mgf (ecd.moment), 58
ecd.mgf_by_sum (ecd.moment), 58
ecd.mgf_diterm (ecd.mgf_term), 57
ecd.mgf_quartic (ecd.moment), 58
ecd.mgf_term, 57
ecd.mgf_term_original (ecd.mgf_term), 57
ecd.mgf_trunc (ecd.mgf_term), 57
ecd.mgf_trunc_max_sigma (ecd.mgf_term), 57
ecd.moment, 58
ecld.mpnum, 59
ecld.mu_D, 60
ecld.mu_D_by_sum (ecld.mu_D), 60
ecld.mu_D_integrate (ecld.mu_D), 60
ecld.mu_D_quartic (ecld.mu_D), 60
ecld.ofg, 60
ecld.ofg.gamma (ecld.ofg), 60
ecld.ofg.innt_sum (ecld.ofg), 60
ecld.ofg.integrate (ecld.ofg), 60
ecld.ofg.log_slope (ecld.ofg), 60
ecld.ofg_quartic (ecld.ofg), 60
ecld.ofg_star, 61
ecld.ofg_star_analytic (ecld.ofg_star), 61
ecld.ofg_star_exp (ecld.ofg_star), 61
ecld.ofg_star_gamma_star (ecld.ofg_star), 61
ecld.ofg_star_hgeo (ecld.ofg_star), 61
ecld.op_o (ecld.op_v), 63
ecld.op_Q (ecld.op_V), 62
ecld.op_Q_skew (ecld.op_Q), 62
ecld.op_Q_skew_by_k_lm (ecld.op_Q), 62
ecld.op_U_lag (ecld.op_V), 63
ecld.op_V, 63
ecld.op_VL_quartic (ecld.op_V), 63
ecld.pdf, 65
ecld.quartic_model_sample (ecld.quartic_Qp_atm_attr), 67
ecld.quartic_model_sample_attr (ecld.quartic_Qp_atm_attr), 67
ecld.quartic_Q (ecld.quartic_Qp), 65
ecld.quartic_Qp, 65
ecld.quartic_Qp_atm_attr, 67
ecld.quartic_Qp_atm_ki (ecld.quartic_Qp), 65
ecld.quartic_Qp_atm_skew (ecld.quartic_Qp), 65
ecld.quartic_Qp_rho (ecld.quartic_Qp), 65
ecld.quartic_Qp_skew (ecld.quartic_Qp), 65
ecld.quartic_sn0_atm_ki, 68
ecld.quartic_sn0_max_RNV (ecld.quartic_sn0_atm_ki), 68
ecld.quartic_sn0_rho_stdev (ecld.quartic_sn0_atm_ki), 68
ecld.quartic_sn0_skew (ecld.quartic_sn0_atm_ki), 68
ecld.sapply (ecld.mpnum), 59
ecld.sd, 68
ecld.sged_cdf (ecld.sged_const), 69
ecld.sged_const, 69
ecld.sged_imgf (ecld.sged_const), 69
ecld.sged_mgf (ecld.sged_const), 69
ecld.sged_moment (ecld.sged_const), 69
ecld.sged_ogf (ecld.sged_const), 69
ecld.skewness (ecld.sd), 68
ecld.solve, 70
ecld.solve_by_poly (ecld.solve), 70
ecld.solve_isomorphic (ecld.solve), 70
ecld.solve_quartic (ecld.solve), 70
ecld.var (ecld.sd), 68
ecld.y_slope, 71
ecld.y_slope_trunc (ecld.y_slope), 71
ecld0rEcd-class, 72
ecop-class, 72
ecop.bs_call_price (ecop.bs_option_price), 74
ecop.bs_implied_volatility, 73
ecop.bs_option_price, 74
ecop.bs_put_price (ecop.bs_option_price), 74
ecop.build_opt (ecop.from_symbol_conf), 77
ecop.enrich_option_df (ecop.read_csv_by_symbol), 81
ecop.find_fixed_point_lambda_by_atm_skew, 75
ecop.find_fixed_point_sd_by_lambda, 76
ecop.from_symbol_conf, 77
ecop.get_ld_triple, 78
ecop.opt-class, 79
ecop.plot_option, 80
ecop.polyfit_option, 81
ecop.read_csv_by_symbol, 81
ecop.read_symbol_conf (ecop.from_symbol_conf), 77
ecop.smile_data_calculator (ecop.term_master_calculator), 82
ecop.term_atm (ecop.term_master_calculator), 82
ecop.term_idx_range (ecop.term_plot_3x3), 83
ecop.term_master_calculator, 82
ecopNterm_plot_SxS, 83
ecopNterm_realized_days
(ecopNterm_plot_SxS), 83
ecopNterm_target_days_default
(ecopNterm_plot_SxS), 83
ecopNvix_plot_SxS, 84
ellipticity (ellipticity.ecd), 84
ellipticity,ecd-method
(ellipticity.ecd), 84
ellipticity.ecd, 84
history (history.ecdb), 85
history,ecdb-method (history.ecdb), 85
history.ecdb, 85
integrate_pdf (integrate_pdf.ecd), 86
integrate_pdf,ecd-method
(integrate_pdf.ecd), 86
integrate_pdf.ecd, 86
jinv (jinv.ecd), 87
jinv,ecd-method (jinv.ecd), 87
jinv.ecd, 87
k2mnt, 87
klihnlap (rlihnlap), 102
kqsl (rqsl), 103
ksl (rqsl), 103
kstablecnt (dstablecnt), 9
lamp, 88
lamp-class, 89
lamp.generate_tau, 90
lamp.plot_sim4, 90
lamp.plot_sim6 (lamp.plot_sim4), 90
lamp.qsl_fit_config, 91
lamp.qsl_fit_config.xtable
(lamp.qsl_fit_config), 91
lamp.qsl_fit_plot, 92
lamp.sd_factor, 92
lamp.simulate1, 93
lamp.simulate_iter, 94
lamp.stable_rnd_walk, 95
levy.dlambda, 95
levy.domain_coloring, 96
levy.dskewed, 97
mnt2k (k2mnt), 87
moment (moment.ecd), 98
moment,ecd-method (moment.ecd), 98
moment.ecd, 98
numericMfr-class, 99
pec (dec), 7
plot_2x2 (plot_2x2.ecd), 99
plot_2x2,ecd-method (plot_2x2.ecd), 99
plot_2x2.ecd, 99
pstablecnt (dstablecnt), 9
qec (dec), 7
qsl_kurtosis_analytic (rqsl), 103
qsl_pdf_integrand_analytic (rqsl), 103
qsl_skewness_analytic (rqsl), 103
qsl_std_pdf0_analytic (rqsl), 103
qsl_variance_analytic (rqsl), 103
qstablecnt (dstablecnt), 9
quantilize (quantilize.ecd), 100
quantilize,ecd-method (quantilize.ecd), 100
quantilize.ecd, 100
read (read.ecdb), 101
read,ecdb-method (read.ecdb), 101
read.ecdb, 101
rec (dec), 7
rlaplace0, 101
rlihnlap, 102
rqsl, 103
rsl (rqsl), 103
rstablecnt (dstablecnt), 9
solve,ecd-method (solve.ecd), 105
solve.ecd, 105
solve_sym (solve_sym.ecd), 105
solve_sym,ecd-method (solve_sym.ecd), 105
solve_sym.ecd, 105
solve_trig (solve_trig.ecd), 106
solve_trig,ecd-method (solve_trig.ecd), 106
solve_trig.ecd, 106
summary (summary.ecdb), 107
summary,ecdb-method (summary.ecdb), 107
summary.ecdb, 107
write (write.ecdb), 108
write,list,ecdb-method (write.ecdb), 108
write.ecdb, 108
y_slope (y_slope.ecd), 108
y_slope, ecd-method (y_slope.ecd), 108
y_slope.ecd, 108