Package 'kendallRandomWalks'

October 13, 2022

Title Simulate and Visualize Kendall Random Walks and Related Distributions

Version 0.9.4

Description Kendall random walks are a continuous-space Markov chains generated by the Kendall generalized convolution. This package provides tools for simulating these random walks and studying distributions related to them. For more information about Kendall random walks see Jasiulis-Gołdyn (2014) <arXiv:1412.0220>.

Depends R (>= 3.3)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports ggplot2, dplyr, tibble, actuar

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, testthat, covr

VignetteBuilder knitr

NeedsCompilation no

Author Mateusz Staniak [aut, cre]

Maintainer Mateusz Staniak <mateusz.staniak@math.uni.wroc.pl>

Repository CRAN

Date/Publication 2018-08-04 21:40:02 UTC

R topics documented:

`dkend` *PDF of Kendall stable distribution*

Description

PDF of Kendall stable distribution

Usage

```
dkend(m_alpha)
```

Arguments

`m_alpha`	function giving moments of order alpha of step dist.

Value

function that returns values of the PDF

Examples

```
dKend <- dkend(function(x) 1)
# Step distribution: delta_{1}
dKendall <- dKend(1:10, 0.5)
# Values of PDF for arguments 1:10 and alpha = 0.5
```

---

estimate_stable_alpha *Fit alpha parameter using MLE for distribution with one parameter*

Description

Fit alpha parameter using MLE for distribution with one parameter

Usage

```
estimate_stable_alpha(data)
```

Arguments

data	Numeric vector

Value

list with optimal parameter and loglikelihood value

---

fit_kendall *Fit stable Kendall distribution for given data and m_alpha function.*

Description

Fit stable Kendall distribution for given data and m_alpha function.

Usage

```
fit_kendall(data)
```

Arguments

data	Numeric vector of observation to which the distribution will be fitted.

Value

fitted quantiles

fit_separate — *Function for fitting stable Kendall distribution separately to two parts of data*

Description

Function for fitting stable Kendall distribution separately to two parts of data

Usage

```
fit_separate(data, separation_point)
```

Arguments

data	Numeric vector. Observation to which the distribution will be fitted.
separation_point	Order above which data (quantiles) will be separated.

Value

List of class kendall_fit with estimated and theoretical quantiles and estimated parameters.

fit_stable_alpha — *Fit stable Kendall distribution with one parameter (alpha)*

Description

Fit stable Kendall distribution with one parameter (alpha)

Usage

```
fit_stable_alpha(data)
```

Arguments

data	Numeric vector of data.

Value

list of type kendall_fit

full_loglik_gradient *Gradient of minus loglikelihood for stable Kendall distribution with 3 parameters.*

Description

Gradient of minus loglikelihood for stable Kendall distribution with 3 parameters.

Usage

```
full_loglik_gradient(data)
```

Arguments

data	numeric vector of observation.

Value

Function of one argument of length 3 (alpha, location, scale).

full_minus_loglik *Negative loglikelihood for stable Kendall distr. with 3 parameters.*

Description

Negative loglikelihood for stable Kendall distr. with 3 parameters.

Usage

```
full_minus_loglik(data)
```

Arguments

data	Dataset for which the loglikelihood will be calculated.

Value

numeric, value of loglikelihood

g_function — *Function G(t) - Williamson transform taken at point 1/t.*

Description

This function return only approximated values. To check their precisions use g_function_single function with an argument of length 1.

Usage

```
g_function(t, alpha, density)
```

Arguments

t	Argument to the function.
alpha	Value of the alpha parameter.
density	Density function of the step distribution.

Value

Object of class "integrate"

Examples

```
g_function(1:5, 0.75, dnorm)
```

g_function_single — *Function G(t) - Williamson transform taken at point 1/t.*

Description

This function return the whole "integrate" object, so precision of the approximation can be checked.

Usage

```
g_function_single(t, alpha, density)
```

Arguments

t	Argument to the function.
alpha	Value of the alpha parameter.
density	Density function of the step distribution.

Value

Object of class "integrate"

Examples

```
g_function_single(5, 0.26, dnorm)
```

---

kendallRandomWalks *kendallRandomWalks: explore and visualize Kendall random walks.*

Description

Kendall random walks are Markov processes generated by the Kendall convolution. This package helps simulate and visualize these random walks and associated distributions. It also provides function to fit these distributions to data.

Important functions

`simulate_kendall_rw` simulates Kendall random walks.

`transform_kendall_rw` applies scaling and shift to simulated Kendall r.w-s.

`ladder_moment` estimates the distribution of first ladder moment by simulating Kendall random walks.

`ladder_height` estimates the distribution of first ladder height by simulating Kendall random walks.

`ladder_moment_pmf` computes the PMF of the distribution of first ladder moment.

`g_function` Finds the value of G(t) numerically.

`pkend`, `dkend`, `qkend`, `rkend` give CDF, PDF, quantile function and random numbers from stable Kendall distribution.

---

kendall_loglik *Log-likelihood for stable kendall distribution with m_alpha = 1*

Description

Log-likelihood for stable kendall distribution with m_alpha = 1

Usage

```
kendall_loglik(alpha, x)
```

Arguments

alpha	alpha parameter of the Kendall random walk
x	numeric vector of observations

Value

numeric

---

ladder_height *Estimate the distribution of first ladder height for given level*

Description

NA is returned if the level wasn't crossed. Printing the resulting object will give summary of the estimated distribution and information whether level wasn't crossed in some simulations. This information can be used to pick the right trajectory length for the given level.

Usage

```
ladder_height(simulations, level)
```

Arguments

simulations	kendall_simulation object
level	Positive numeric

Value

tibble

Examples

```
{
  kendall_rw <- simulate_kendall_rw(100, 100, runif, 0.5)
  estim_ladder <- ladder_height(kendall_rw, 1000)
  estim_ladder
}
```

ladder_moment *Estimate the distribution of first ladder moment for given level*

Description

NA is returned if the level wasn't crossed. Printing the resulting object will give summary of the estimated distribution and information whether level wasn't crossed in some simulations. This information can be used to pick the right trajectory length for the given level.

Usage

```
ladder_moment(simulations, level)
```

Arguments

simulations	kendall_simulation object
level	Positive numeric

Value

tibble

Examples

```
{
kendall_rw <- simulate_kendall_rw(100, 100, runif, 0.5)
estim_ladder <- ladder_moment(kendall_rw, 1000)
estim_ladder
}
```

ladder_moment_pmf *Distribution of the first ladder moment.*

Description

Distribution of the first ladder moment.

Usage

```
ladder_moment_pmf(n, level, alpha, step_cdf, step_pdf)
```

Arguments

n	Argument to the PDF.
level	Level a to be crossed.
alpha	Alpha parameter of Kendall random walk.
step_cdf	CDF of the step distribution.
step_pdf	PDF of the step distribution.

Value

Value of PMF of the distribution of first ladder moment

Examples

```
prob <- ladder_moment_pmf(10, 1000, 0.5, pnorm, dnorm)
prob
```

---

mutate_kendall_rw *Mutate each trajectory.*

Description

Mutate each trajectory.

Usage

```
mutate_kendall_rw(simulations, mutate_function, df = T)
```

Arguments

simulations	Object of class kendall_simulation.
mutate_function	Function that will be applied to each trajectory.
df	If TRUE, a d.f will be returned, if FALSE, simulations in the kendall_simulation object passed in simulations argument will be replaced by the result of mutate_function.

Value

data frame or a list (of class kendall_simulation)

pkend *CDF of Kendall stable distribution*

Description

CDF of Kendall stable distribution

Usage

```
pkend(m_alpha)
```

Arguments

m_alpha	function giving moments of order alpha of step dist.

Value

function function giving values of CDF of Kendall stable distribution

Examples

```
pKend <- pkend(function(x) 1)
# Step distribution: delta_{1}
pKendall <- pKend(1:10, 0.5)
# Values of CDF for arguments 1:10 and alpha = 0.5
```

pkendSym *CDF of symmetrical Kendall stable distribution*

Description

CDF of symmetrical Kendall stable distribution

Usage

```
pkendSym(m_alpha)
```

Arguments

m_alpha	function giving moments of order alpha of step dist.

Value

function function giving values of CDF of Kendall stable distribution

Examples

```
pKend <- pkendSym(function(x) 1)
# Step distribution: delta_{1}
pKendall <- pKend(1:10, 0.5)
# Values of CDF for arguments 1:10 and alpha = 0.5
```

`plot.kendall_barrier_crossing` *Generic function for plotting results of ladder_moment function.*

Description

Generic function for plotting results of ladder_moment function.

Usage

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

Arguments

`x`	kendall_barrier_crossing object
`...`	Additional arguments

Value

ggplot2

`plot.kendall_fit` *QQ-plot for the result of fitting stable Kendall distribtion.*

Description

QQ-plot for the result of fitting stable Kendall distribtion.

Usage

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

Arguments

`x`	List returned by fit_separate or fit_kendall function.
`...`	Aditional arguments.

Value

ggplot2 object

`plot.kendall_simulation` *Generic function that draws simulated trajectories of Kendall random walk*

Description

Generic function that draws simulated trajectories of Kendall random walk

Usage

```
## S3 method for class 'kendall_simulation'
plot(x, max_x = NULL, max_id = NULL,
  level = NULL, ...)
```

Arguments

`x`	object returned by normalising_sequences function.
`max_x`	maximum value on x axis.
`max_id`	Number of trajectories to plot. If NULL, all paths will be plotted.
`level`	Y-axis value which will be marked (level to be crossed).
`...`	Other arguments

Value

ggplot2 object

`plot.kendall_summary` *Plot summary of Kendall random walk simulations.*

Description

Plot summary of Kendall random walk simulations.

Usage

```
## S3 method for class 'kendall_summary'
plot(x, ..., type = "density")
```

Arguments

x	Object of class kendall_summary
...	Optional arguments, currently ignored
type	Type of the plot: density, histogram or boxplot

Value

ggplot2 object

---

print.kendall_barrier_crossing

Generic function for printing result of ladder_moment function

---

Description

Generic function for printing result of ladder_moment function

Usage

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

Arguments

x	kendall_barrier_crossing object
...	Additional arguments

Value

invisible x

---

print.kendall_simulation

Generic function that prints information about simulated Kendall random walk

---

Description

Generic function that prints information about simulated Kendall random walk

Usage

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

Arguments

x	Object returned by simulate_kendall_rw or transform_kendall_rw function.
...	Other arguments.

---

print.kendall_summary *Print summary of Kendall random walk simulations.*

Description

Print summary of Kendall random walk simulations.

Usage

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

Arguments

x	Object of type kendall_summary
...	Optional parameters, currently ignored

---

qkend *Quantiles of Kendall stable distribution*

Description

Quantiles of Kendall stable distribution

Usage

```
qkend(m_alpha)
```

Arguments

m_alpha	function giving moments of order alpha of step dist.

Value

function function returning quantiles of given orders

Examples

```
qKend <- qkend(function(x) 1)
# Step distribution: delta_{1}
qKendall <- qKend(c(0.1, 0.9), 0.5)
# Quantiles of order 0.1 and 0.9 for alpha = 0.5
```

qkendSym *Quantiles of symmetrical Kendall stable distribution*

Description

Quantiles of symmetrical Kendall stable distribution

Usage

```
qkendSym(m_alpha)
```

Arguments

m_alpha	function giving moments of order alpha of step dist.

Value

function function returning quantiles of given orders

Examples

```
qKend <- qkendSym(function(x) 1)
# Step distribution: delta_{1}
qKendall <- qKend(c(0.1, 0.9), 0.5)
# Quantiles of order 0.1 and 0.9 for alpha = 0.5
```

Qn *Helper function*

Description

Helper function

Usage

```
Qn(x, y, alpha)
```

Arguments

x	numeric
y	numeric
alpha	numeric, parameter of Kendall random walk

Value

0 or 1 with probability depending on x, y, alpha

rkend *Pseudo-random number from Kendall stable distribution*

Description

Pseudo-random number from Kendall stable distribution

Usage

```
rkend(m_alpha)
```

Arguments

m_alpha	function giving moments of order alpha of step dist.

Value

function return n numbers genereted from Kendall stable dist.

Examples

```
rKend <- rkend(function(x) 1)
# Step distribution: delta_{1}
rKendall <- rKend(10, 0.5)
# Ten random number from stable Kendall distribution with alpha = 0.5
```

simulateOneTrajectory *Simulate one trajectory ofa Kendall random walk*

Description

Simulate one trajectory ofa Kendall random walk

Usage

```
simulateOneTrajectory(trajectory_length, step_dist, alpha,
  symmetric = FALSE, ...)
```

Arguments

trajectory_length	Number of samples to simulate.
step_dist	Function that returns random numbers from step distribution.
alpha	Alpha parameter of the random walk
symmetric	If TRUE, random walk on the whole real line will be simulated.
...	Additional parameters to step distribution.

Value

Generated path of the random walk.

`simulate_kendall_rw` *Simulate multiple trajectories of Kendall random walk*

Description

Object returned by this has print and plot methods.

Usage

```
simulate_kendall_rw(number_of_simulations, trajectory_length, step_dist,
  alpha, symmetric = FALSE, ...)
```

Arguments

`number_of_simulations`	number of trajectories to generate.
`trajectory_length`	length of trajectories.
`step_dist`	function returning random numbers from step dist.
`alpha`	alpha parameter.
`symmetric`	If TRUE, random walk on the whole real line will be simulated.
`...`	parameters for step distribution.

Value

Object of class kendall_simulation. It is a list that consists of

`simulation`	Tibble with simulation id and simulated values,
`step_distribution`	Name of the step distribution,
`alpha`	Value of alpha parameter,
`is_symmetric`	Logical value indicating if this is a symmetric Kendall R.W.

Examples

```
kendall_simulations <- simulate_kendall_rw(10, 1000, runif, 0.5)
# Kendall R.W. on positive half-line with uniform step distribution - 10 trajectories.
only_simulations <- kendall_simulations$simulation # tibble with simulated values
kendall_simulations
```

summarise_kendall_rw *Calculate some characteristic for every simulated instance.*

Description

Calculate some characteristic for every simulated instance.

Usage

```
summarise_kendall_rw(simulations, summary_function)
```

Arguments

simulations	Object of class kendall_simulation.
summary_function	Function that will be applied to each trajectory.

Value

data frame of class "kendall_summary".

transform_kendall_rw *Transforming (scaling and shifting) Kendall random walks*

Description

If one trajectory has length n, an_seq and bn_seq arguments should be sequnces of length n. Object returned by this function has plot and print methods.

Usage

```
transform_kendall_rw(simulations, an_seq = 1, bn_seq = 0)
```

Arguments

simulations	tibble returned by simulation function
an_seq	sequence that the trajectories will be multiplied by
bn_seq	sequence that will be substracted from scaled trajectory

Value

List like in simulate_kendall_rw function after transforming trajectories.

Examples

```
kendall_simulations <- simulate_kendall_rw(10, 1000, runif, 0.5)
scaled_kendall <- transform_kendall_rw(kendall_simulations, (1:1000)^(-2))
scaled_kendall # kendall random walked scaled by the sequence n^(-1/alpha)
scaled_data <- scaled_kendall$simulation # simulated values
plot(scaled_kendall)
```

U *Helper function*

Description

Helper function

Usage

```
U(x, y)
```

Arguments

x	numeric
y	numeric

Value

sign of the argument whose abs. val. is bigger

Z *Helper function: min/max*

Description

Helper function: min/max

Usage

```
Z(x, y)
```

Arguments

x	numeric
y	numeric

Value

min of arguments divided by max of arguments

Index