Package 'Emcdf'

January 13, 2018

Type Package

Title Computation and Visualization of Empirical Joint Distribution (Empirical Joint CDF)

Version 0.1.2

Author En-shuo Hsu, Jeffrey C. Miecznikowski

Maintainer En-shuo Hsu <daviden1013@gmail.com>

Description Computes and visualizes empirical joint distribution of multivariate data with optimized algorithms and multi-thread computation. There is a faster algorithm using dynamic programming to compute the whole empirical joint distribution of a bivariate data. There are optimized algorithms for computing empirical joint CDF function values for other multivariate data. Visualization is focused on bivariate data. Levelplots and wireframes are included.

License GPL-3

LazyData TRUE

Imports Rcpp (>= 0.12.8), methods

Depends R (>= 3.2.5), lattice

LinkingTo Rcpp

RoxygenNote 5.0.1

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-01-13 06:36:24 UTC

R topics documented:

Biemcdf *Computes bivariate empirical joint distribution*

Description

This function computes empirical joint distribution (joint CDF) table with dynamical programming.

Usage

```
Biemcdf(data)
```

Arguments

data	a numeric matrix with two columns.

Details

This is an optimization for bivariate data.

Value

a matrix of values of empirical joint CDF function, where rows and columns are the sorted variables. Columns are the first variable, and rows are the second variable.

Examples

```
n = 10^2
set.seed(123)
x = rnorm(n)
y = rnorm(n)
data = cbind(x, y)
Biemcdf(data)
```

coreNum *CPU core number*

Description

This function reports the total core number in CPU on the machine.

Usage

```
coreNum()
```

Details

This function is a wrapper for the C++ function std::thread::hardware_concurrency(); If core number is not detected, 0 will be returned. See "See Also" for details.

Value

an integer indicating the total number of cores in CPU.

See Also

http://www.cplusplus.com/reference/thread/thread/hardware_concurrency/

Examples

```
num = coreNum()
```

---

emcdf *Computes multivariate empirical joint distribution*

Description

This function computes empirical joint distribution (joint CDF) with single/ multi-thread.

Usage

```
emcdf(data, a)
```

Arguments

data	a numeric matrix stores data. Or an S4 object of class "emcdf_obj".
a	a numeric vector or matrix of parameters for CDF function.

Details

When data is a numeric matrix, this function computes joint empirical CDF with single thread. When data is an object of class "emcdf_obj", it computes with multi-thread. Parameter "a" must have equal length (or equal column number) as the column number of data. Both single-thread and multi-thread emcdf algorithms are faster than using the bulit-in function sum{base}. See example for simulation. Note that initializing threads and spliting data takes time though it's a one-time task. Thus for big data, big number of CDF computation, multi-thread is recommended. Yet for small data, small number of CDF computation, single thread is faster.

Value

a numeric (vector) as value(s) of empirical joint CDF function.

Examples

```
n = 10^6
set.seed(123)
x = rnorm(n)
y = rnorm(n)
z = rnorm(n)
data = cbind(x, y, z)
#The aim is to compute F(0.5,0.5,0.5) with three
#approaches and compare the performances.
#To avoid CPU noises, we repeat the computation 10 times.
#compute with R built-in function, sum()
sum_time = system.time({
  aws1 = c()
  for(i in 1:10)
    aws1[i] = sum(x <= 0.5& y <=0.5& z <=0.5)/n
})[3]

#compute with emcdf single-thread
a = matrix(rep(c(0.5, 0.5, 0.5), 10), 10, 3)
single_time = system.time({
   aws2 = emcdf(data, a)
})[3]

obj = initF(data, 4)
multi_time = system.time({
   aws3 = emcdf(obj, a)
})[3]
aws2 == aws1
aws3 == aws1
sum_time
single_time
multi_time
```

`initF` *Initialize threads and split data*

Description

This function initializes threads and splits data. Retures an S4 object of class "emcdf_obj" that is ready for parallel computation.

Usage

```
initF(data, num = 2)
```

Arguments

data	a numeric matrix stores data. Colunms as variables.
num	an integer specifies number of threads to initialize.

Details

The input data must be a numeric matrix with variables as columns. The choice of "num" is machine dependent. A reasonable number would be the total number of CPU cores - 1. Call coreNum() to get CPU core number.

Value

an S4 object of class "emcdf_obj", holding pointer to an C++ object. When passed to function emcdf(), it computes joint CDF with multi-threads.

Examples

```
n = 10^5
set.seed(123)
x = rnorm(n)
y = rnorm(n)
z = rnorm(n)
data = cbind(x, y, z)

#decide thread number
num = coreNum() - 1

#initialize threads
obj = initF(data, num)

#compute empirical CDF
emcdf(obj, c(0.5, 0.5, 0.5))
```

---

plotcdf *Plots multivariate empirical joint distribution of bivariate data*

Description

This function plots empirical joint distribution (joint CDF) with levelplot, and 3D wireframes.

Usage

```
plotcdf(data, type = "levelplot", angle = 60,
  main = paste("Bivariate CDF of", deparse(substitute(data))))
```

Arguments

data	a numeric matrix / data frame of two variables.
type	a character spicifies plot types. Must be one of "levelplot", "wireframe", or "multiple_wireframe".
angle	a numeric scalar for z axis rotation. With default = 60 degrees.
main	a character of plot title.

Details

When type = "multiple_wireframe", this function plots 8 wireframes of directions 0 to what parameter angle is. This process takes longer. When type = "levelplot", parameter angle has no effect.

Examples

```
n = 10
set.seed(123)
x = rnorm(n)
y = x^2 + 0.1*rnorm(n)
data = cbind(x, y)
plotcdf(data, type = "multiple_wireframe")
```

Index