Package ‘TTTSizer’

June 7, 2020

Type Package

Title TTT-SiZer: A Graphic Tool For Aging Trends Recognition

Version 1.0

Author Rafael Adolfo Nozal Cañadas

Maintainer Rafael Adolfo Nozal Cañadas <rca015@uit.no>

Description A new graphic tool is presented to test aging trends based on lifetime data. The graphical test is developed by means of scale and space inference about the Total-Time-on-Test transform and its first and second derivatives. The graphic tool TTT-SiZer is defined considering nonparametric local polynomial kernel estimators and constructing the corresponding (simultaneous as well as punctual) confidence intervals around three different curves. The finite sample properties of the method are evaluated by a simulation study and the comparison with other non-graphical tests shows that the graphical test helps localize discrepancies of empirical data concerning a given hypothesized aging property, thus allowing to solve the problem locally.

License GPL-3

Encoding UTF-8

LazyData true

Imports ggplot2, reshape2, dplyr, lubridate, latex2exp, ggpubr

RoxygenNote 7.1.0

NeedsCompilation no

R topics documented:

`findBigA` *TTTSizer*

Description

Finds a single A_r(p0). This is use the function that finds all A's. (uppercase)

Usage

```
findBigA(p0, h, r, phiVector, kernelCube, poliCube)
```

Arguments

`p0`	- An index number refering to the p0 value
`h`	- An index number refering to the bandwith
`r`	- index with the ^r value. (Note, R indexing start at 1, 1 = -> ^0, 2 -> ^1, and so on)
`phiVector`	- The vector with the phi values
`kernelCube`	- The cube with all the Kh(pi-p0) values
`poliCube`	- The cube with all the (pi-p0)^n values

Value

A single float with the Ar(p0)

Note

The function needs you to calculate first the following constants

1. - a kernel cube with all the K_h(pi-p0) values
2. - a polinomial cube with all the (pi-p0)^n values
3. - a vector with all the phi(xis) values

See Also

findBigAvalues

`findBigAvalues` *TTTSizer*

Description

For a given set of data, find the A matrix of dimession i x 1

Usage

```
findBigAvalues(p0Index, hIndex, phiVector, kernelCube, poliCube)
```

Arguments

`p0Index`	- An index number refering to the p0 value
`hIndex`	- An index number refering to the bandwith
`phiVector`	- The vector with the phi values
`kernelCube`	- The cube with all the Kh(pi-p0) values
`poliCube`	- The cube with all the (pi-p0)^n values

Value

Return a vector with the (A0,A1,A2, A3) values

Note

The function needs you to calculate first the following constants

1. - a kernel cube with all the K_h(pi-p0) values
2. - a polinomial cube with all the (pi-p0)^n values
3. - a vector with all the phi(xis) values

See Also

findThetas

`findLittleA` *TTTSizer*

Description

Finds a single a_r(p0). This is use the function that finds all a's. (lowercase)

Usage

```
findLittleA(p0, h, r, kernelCube, poliCube)
```

Arguments

`p0`	- An index number refering to the p0 value
`h`	- An index number refering to the bandwith
`r`	- index with the ^r value. (Note, R indexing start at 1, 1 = -> ^0, 2 -> ^1, and so on)
`kernelCube`	- The cube with all the Kh(pi-p0) values
`poliCube`	- The cube with all the (pi-p0)^n values

Value

A single float with the Ar(p0)

Note

The function needs you to calculate first the following constants

1. - a kernel cube with all the K_h(pi-p0) values
2. - a polinomial cube with all the (pi-p0)^n values
3. - a vector with all the phi(xis) values

See Also

findThetas

`findThetas` *TTTSizer*

Description

Find thetas for a quadratic method. For a given set of data, find the variable matrix solution of theta0, theta1, and theta2

Usage

```
findThetas(bigAs, littleAs)
```

Arguments

`bigAs`	vector with the big As associated with that p0,x,p,h and kernel
`littleAs`	vector with the little As associated with that p0, p,h and kernel

Value

c(NaN, NaN, NaN) - If is not a Cramer system c(theta0, theta1, theta2) - Otherwise

`findThetasCubic` *TTTSizer*

Description

Find thetas for a cubic method. For a given set of data, find the variable matrix solution of theta0, theta1, and theta2

Usage

```
findThetasCubic(bigAs, littleAs)
```

Arguments

`bigAs`	vector with the big As associated with that p0,x,p,h and kernel
`littleAs`	vector with the little As associated with that p0, p,h and kernel

Value

c(NaN, NaN, NaN) - If is not a Cramer system c(theta0, theta1, theta2) - Otherwise

---

`findVariancePhiWithKernel` *TTTSizer*

Description

Find the variance for a given derivate with a given kernelCube

Usage

```
findVariancePhiWithKernel(
  h,
  p0,
  kernelBarCube,
  sigmaMatrix,
  weightMatrix,
  order
)
```

Arguments

`h`	- An index number refering to the bandwith.
`p0`	- An index number refering to the p0 value.
`kernelBarCube`	- A matrix with all the Kbar_h(pi-p0) combinations.
`sigmaMatrix`	- A Variance/Coovariance matrix.
`weightMatrix`	- A weight matrix object.
`order`	- The order of the kernel cube you want to generate

Value

The variance value that correspond to the given indexes

`findVariancePhiWithKernelCubic` *TTTSizer*

Description

Find the variance for a given derivate with a given kernelCube (cubic version)

Usage

```
findVariancePhiWithKernelCubic(
  h,
  p0,
  kernelBarCube,
  sigmaMatrix,
  weightMatrix,
  order
)
```

Arguments

`h`	- An index number refering to the bandwith.
`p0`	- An index number refering to the p0 value.
`kernelBarCube`	- A matrix with all the Kbar_h(pi-p0) combinations.
`sigmaMatrix`	- A Variance/Coovariance matrix.
`weightMatrix`	- A weight matrix object.
`order`	- The order of the kernel cube you want to generate

Value

The variance value that correspond to the given indexes

`generateKernelBarHCube` *TTTSizer*

Description

For a given kernelCube and polinomialCube, and a's matrix, generate all possible Kbar_h(pi-p0) combinations. This is only for the quadratic version, thus you only need 5 matrices

Usage

```
generateKernelBarHCube(
  kernelCube,
  poliCube,
  a0Matrix,
  a1Matrix,
  a2Matrix,
  a3Matrix,
  a4Matrix,
  order
)
```

Arguments

`kernelCube`	- The cube with all the Kh(pi-p0) values
`poliCube`	- The cube with all the (pi-p0)^n values
`order`	- The order of the kernel cube you want to generate
`anMatrix`	- A matrix with the pi,h values of the a_n's

Value

A list of matrixs with all the Kbar_h(pi-p0) combinations pre-calculated. Where the index of the list represent the h value, the row is pi and the column p0

Note

The function needs you to calculate first the following constants

1. - a kernel cube with all the K_h(pi-p0) values
2. - a polinomial cube with all the (pi-p0)^n values

`generateKernelBarHCubeCubic` *TTTSizer*

Description

For a given kernelCube and polinomialCube, and a's matrix, generate all possible Kbar_h(pi-p0) combinations. This is for the cubic version, and thus you need the 7 matrices. In the paper, this is notated as K tilde

Usage

```
generateKernelBarHCubeCubic(
  kernelCube,
  poliCube,
  a0Matrix,
  a1Matrix,
  a2Matrix,
  a3Matrix,
  a4Matrix,
  a5Matrix,
  a6Matrix,
  order
)
```

Arguments

`kernelCube`	- The cube with all the Kh(pi-p0) values
`poliCube`	- The cube with all the (pi-p0)^n values
`order`	- The order of the kernel cube you want to generate
`anMatrix`	- A matrix with the pi,h values of the a_n's

Value

A list of matrixs with all the Kbar_h(pi-p0) combinations pre-calculated. Where the index of the list represent the h value, the row is pi and the column p0

Note

The function needs you to calculate first the following constants

1. - a kernel cube with all the K_h(pi-p0) values
2. - a polinomial cube with all the (pi-p0)^n values

`generateKernelHCube` *TTTSizer*

Description

For a given vector of pi's and h's, generate all possible K_h(pi-p0) combinations.

Usage

```
generateKernelHCube(piVector, p0Vector, hVector, kernel)
```

Arguments

`piVector`	The vector with all the frecuencies (can be a vector of size one)
`p0Vector`	The vector with only the p0 values (can be a vector of size one)
`hVector`	The vector with only the h's values (can be a vector of size one)
`kernel`	The selected kernel function ("gaussian", "biweight", "triweight", "epanechnikov")

Value

A list of matrixs with all the K_h(pi-p0) combinations pre-calculated. Where the index of the list represent the h value, the row is pi and the column p0

The row is the h, the column is the sum(K_h(pi-p0))

`generatePolinomialCube` *TTTSizer*

Description

For a given vector of pi's and generate all possible (pi-p0)^n combinations. Where n=0,1,2,3,4,5,6

Usage

```
generatePolinomialCube(piVector, p0Vector)
```

Arguments

`piVector`	The vector with all the frecuencies (can be a vector of size one)
`p0Vector`	The vector with only the p0 values (can be a vector of size one)

Value

A list of matrix with all the(pi-p0)^n combinations pre-calculated. The index of the list is n, the row of the matrix is pi and the column is p0

`generateVarianceXVectorBOOTSTRAP` *TTTSizer*

Description

Generate the Variance/Coovariance matrix with a bootstrap algorithm

Usage

```
generateVarianceXVectorBOOTSTRAP(x, bootFactor = 100)
```

Arguments

`x`	The original vector with your data points.
`bootFactor`	The number of bootstrap iterations, default is 100

Value

A matrix with all the coovariances and variances. The main diagonal contains the variance vector

generateWeightMatrix *TTTSizer*

Description

Generate a weight matrix object of size n x n with this format

1 0 0 ... 0 1/n (n-1)/n 0 ... 0 1/n 1/n (n-2)/n ... 0 ... 1/n 1/n 1/n ... 1/n

Usage

```
generateWeightMatrix(n)
```

Arguments

`n`	Size of the matrix

Value

A matrix of floats as specified in the description

getDataFromFile *TTTSizer*

Description

Read a file with numbers and return them as vector

Usage

```
getDataFromFile(filePath)
```

Arguments

`filePath`	The file path of the file

Value

A vector with the numbers

Examples

```
getDataFromFile("/home/me/myNumbers.txt")
```

`hello` *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`kernelFunction` *TTTSizer*

Description

For a given number, return the value of the kernel function, with the specified kernel

Usage

```
kernelFunction(x, kernel = "gaussian")
```

Arguments

`x`	(float) A value for the symetric kernel function
`kernel`	The selected kernel function **gaussian** exp(-(x^2) / 2) / sqrt(2 * pi) **biweight** (abs(x)<=1)*15/16*(1-x^2)^2 **triweight** (abs(x)<=1)*35/32*(1-x^2)^3 **epanechnikov** (abs(x)<=1)*((1-x^2)*3/4)

Value

(float) The kernel function result.

Examples

```
kernelFunction(1.5)
          kernelFunction(0, kernel="biweight")
```

`kernelHFunction` *TTTSizer*

Description

Correct the kernel of a given data with a given badwith h and k

Usage

```
kernelHFunction(x, h, kernel)
```

Arguments

`x`	the data to find the kernel, can be a vector
`h`	Any given bandwith
`kernel`	The selected kernel function ("gaussian", "biweight", "triweight", "epanechnikov")

Value

a float vector of size length(x) with the result of the kernel

Examples

```
kernelHFunction(1,2,"gaussian")
```

`summaryColors` *TTTSizer*

Description

After running a TTT function, summarize the results of each SiZer map into a dataframe. - The first column of the dataframe is the color name - The second column of the dataframe is proportion of pixels for that SiZer map. - Notice that rows are group 4 by 4. The first 4 is the SiZer-0, the next 4 SiZer-1, and the last 4 the SiZer-2

Usage

```
summaryColors(sizerData)
```

Arguments

`sizerData`	The summary returned by the TTT function

Value

A 12 x 2 dataframe with the color info summarized.

ttt	*For a given array of data, creates several TTT-SiZer plots - Plot with the raw data using density a density plot. - Plot with the phi vector. - The Theta / Phi_h vector / Family plots for Theta0, Theta1, Theta2 - All theta plot together in the same image - The SiZer 0, SiZer 1, and SiZer 2 plots - All SiZer plot together in the same image with the family plot Theta0 - The SiZer 0, SiZer 1, and SiZer 2 plots with the z quantiles instead of strict categorical pixels - The ESS (Effective Sample Space)*

Description

For a given array of data, creates several TTT-SiZer plots - Plot with the raw data using density a density plot. - Plot with the phi vector. - The Theta / Phi_h vector / Family plots for Theta0, Theta1, Theta2 - All theta plot together in the same image - The SiZer 0, SiZer 1, and SiZer 2 plots - All SiZer plot together in the same image with the family plot Theta0 - The SiZer 0, SiZer 1, and SiZer 2 plots with the z quantiles instead of strict categorical pixels - The ESS (Effective Sample Space)

Usage

```
ttt(
  myData,
  xgrid = 401,
  ygrid = 11,
  hMin = 0,
  hMax = 1,
  kernel = "gaussian",
  myMethod = "quadratic",
  quantileMethod = "normal",
  alpha = 0.05,
  ESSLimit = 5,
  bootstrapSample = 500,
  savePlots = TRUE,
  saveCSV = TRUE,
  saveLog = FALSE,
  blackAndWhite = FALSE
)
```

Arguments

`myData`	array with the data. Doesn't need to be sorted.
`xgrid`	number of estimation points. Default is 401. Beware that this could take some time to calculate.
`ygrid`	how many h we are we going to generate. Default is 11.

`hMin` the minimum h you want to try.
Default is 1/(ygrid-1).
If you set a minimum bigger than the maximum, or smaller than 0, the default will be used instead

`hMax` the maximum h you want to try.
Default is 1.
If you set a maximum smaller than the minimum, or bigger than 1, the default will be used instead

`kernel` which kernel do you want:
"epanechnikov" "biweight" "triweight"

`myMethod` which type of interpolation do you use:
"quadratic" (DEFAULT) "cubic"

`quantileMethod`
You can choose which quantile method to use when calculating the confident intervals.
"normal" (DEFAULT)
- Classic Z-score based on a normal distribution. For this method you can specify the alpha parameter. For example, for alpha 0.05, you will get a Z of 1.96
"simultaneous"
- It uses a Z-score different for each pixel, depending on the h value for that pixel.

`alpha` Level of significance for the confident intervals.
Default is 0.05
See the quantileMethod for more info

`ESSLimit` Effective Sample Space. (Default = 5)
How many numbers do you need to have around to be a valid result.

`bootstrapSample`
If you choose to use the variance with the bootstrap method, you can specify the number of sample use for the bootstrap
Default is 100

`savePlots` Generate all the plots and save them into the result folder (Default = TRUE)

`saveCSV` Save all numbers used to generate all the plots into a CSV file (Default = TRUE)

`saveLog` Save every single calculation into a TXT file. File grow exponential, and is around 20MB for 400 x 10 run. (Default = FALSE) Use this only for debuggin.

`blackAndWhite`
Save your plots with a black and white theme (Default = FALSE)

Value

The function itself, return the raw data of all the calculations in a dataframe with this columns:

This two numbers correspond which each pixel in the SiZer plots. So each row of the dataframe represent the information in each pixels:

p0 - A value between 0 and 1 h - The softener used for the kernel

The values for the function, first derivative, and second derivative.

phiZero phiOne phiTwo

The variance for that given derivatives

zeroVariance firstVariance secondVariance

The Effective Sample Space value for that pixel.

ESS - A value between 0 and infinity.

These are the limits on the left and the right, for the confident interval in each derivative.

LeftIntervalZero LeftIntervalFirst LeftIntervalSecond RightIntervalZero RightIntervalFirst RightIntervalSecond

This are all boolean values and tell you whether the average is inside the confident interval, under the lowest limit, or above the upper limit. For each of the derivatives.

ZeroInZero ZeroInFirst (average inside) ZeroInSecond

ZeroSmallerZero ZeroSmallerFirst (average under) ZeroSmallerSecond

ZeroBiggerZero ZeroBiggerFirst (average above) ZeroBiggerSecond

Which color correspond in the SiZer map

ColorCodeZero ColorCodeFirst ColorCodeSecond

In the continuos SiZer maps, this tells you how many sigmas away is the derivative which respect the average

distanceZero distanceFirst distanceSecond

The phi vector value that is assigned to that p0 value. This column is redundant and it repeat itself each time the same p0 appear in a row.

PhiVector

Also; several plots will appear into the result folder, if the savePlot option was set to TRUE The result folder is named as the timestamp of the moment you run the script.

Examples

```
This should take about 40 seconds if you run it with a CPU made from a toaster:

xgrid = 50
ygrid = 11
myData  = getRandomData(50,"gamma", 1/5, 5)

ttt(myData, xgrid, ygrid, kernel = "gaussian", myMethod = "quadratic", variance = "bootst
```