Package: ScatterDensity (via r-universe)

October 24, 2024

Type Package

Title Density Estimation and Visualization of 2D Scatter Plots

Version 0.0.3

Date 2023-07-18

Maintainer Michael Thrun <m.thrun@gmx.net>

Description The user has the option to utilize the two-dimensional density estimation techniques called smoothed density published by Eilers and Goeman (2004) <doi:10.1093/bioinformatics/btg454>, and pareto density which was evaluated for univariate data by Thrun, Gehlert and Ultsch, 2020 <doi:10.1371/journal.pone.0238835>. Moreover, it provides visualizations of the density estimation in the form of two-dimensional scatter plots in which the points are color-coded based on increasing density. Colors are defined by the one-dimensional clustering technique called 1D distribution cluster algorithm (DDCAL) published by Lux and Rinderle-Ma (2023) <doi:10.1007/s00357-022-09428-6>.

LazyLoad yes

Imports Rcpp, pracma

Suggests DataVisualizations, ggplot2, ggExtra, plotly, FCPS, parallelDist, secr, ClusterR, flowCore, flowWorkspace, flowGate, dplyr, tibble

Depends methods, R (>= 2.10)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

License GPL-3

Encoding UTF-8

URL https://www.deepbionics.org/

BugReports https://github.com/Mthrun/ScatterDensity/issues

Repository https://mthrun.r-universe.dev

RemoteUrl https://github.com/mthrun/scatterdensity

RemoteRef HEAD

RemoteSha b568205a9a1d0c12444803c7267acbed5ff04e0c

Contents

`ScatterDensity-package`

Density Estimation and Visualization of 2D Scatter Plots

Description

The user has the option to utilize the two-dimensional density estimation techniques called smoothed density published by Eilers and Goeman (2004) <doi:10.1093/bioinformatics/btg454>, and pareto density which was evaluated for univariate data by Thrun, Gehlert and Ultsch, 2020 <doi:10.1371/journal.pone.0238835>. Moreover, it provides visualizations of the density estimation in the form of two-dimensional scatter plots in which the points are color-coded based on increasing density. Colors are defined by the one-dimensional clustering technique called 1D distribution cluster algorithm (DDCAL) published by Lux and Rinderle-Ma (2023) <doi:10.1007/s00357-022-09428-6>.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Michael Thrun [aut, cre, cph] (<https://orcid.org/0000-0001-9542-5543>), Felix Pape [aut, rev], Luca Brinkman [aut], Quirin Stier [aut] (<https://orcid.org/0000-0002-7896-4737>)

Maintainer: Michael Thrun <m.thrun@gmx.net>

Examples

```
#Todo
```

DDCAL *Density Distribution Cluster Algorithm of [Lux and Rinderle-Ma, 2023].*

Description

DDCAL is a clustering-algorithm for one-dimensional data, which heuristically finds clusters to evenly distribute the data points in low variance clusters.

Usage

```
DDCAL(data, nClusters, minBoundary = 0.1, maxBoundary = 0.45,

numSimulations = 20, csTolerance = 0.45, csToleranceIncrease = 0.5)
```

Arguments

`data`	[1:n] Numeric vector, with the data values
`nClusters`	Scalar, number of clusters to be found
`minBoundary`	Scalar, in the range (0,1), gives the lower boundary (in percent), for the simulation. Default is 0.1
`maxBoundary`	Scalar, in the range (0,1), gives the upper boundary (in percent), for the simulation. Default is 0.45
`numSimulations`	Scalar, number of simulations/iterations of the algorithm
`csTolerance`	Scalar, in the range (0,1). Gives cluster size tolerance factor. The necessary cluster size is defined by (dataSize/nClusters - dataSize/nClusters * csTolerance). Default is 0.45
`csToleranceIncrease`	Scalar, in the range (0,1), gives the procentual increase of the csTolerance-factor, if some clusters did not reach the necessary size. Default is 0.5

Details

DDCAL creates a evenly spaced division of the min-max-normalized data from minBoundary to maxBoundary. Those divisions will be used as boundaries. The first initial clusters will be the data from min(data) to minBoundary and maxBoundary to max(data). The clusters will be extended to neighboring points, as long as the standard deviations of the clusters will be reduced. A potential clusters will be used, if they have the necessary size, given as (dataSize/nClusters - dataSize/nClusters * csTolerance). If both clusters can be used, the left cluster (which is the cluster from min(data) to minBoundary or above) is preferred. If no clusters can be found with the necessary size, then the csTolerance-factor and with it the necessary cluster size will be lowered. If a clusters is used, the next boundaries are found, which are not in the already existing clusters and the procedure is repeated with the not already clustered data, until all points are assigned to clusters.

If a matrix is given as input data, the first column of the matrix will be used as data for the clustering

Non-finite values will not be clustered, but instead will get the cluster label NaN.

The algorithm is not garantueed to produce the given number of clusters, given in nClusters. The found number of clusters can be lower, depending on the data and input parameters.

Value

`labels` [1:n] Numeric vector, containing the labels for the input data points

Author(s)

Luca Brinkmann

References

[Lux and Rinderle-Ma, 2023] Lux, M., Rinderle-Ma, S.: DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling; Springer Journal of Classification, Vol. 40, pp. 106-144, DOI: doi:10.1007/s00357022094286, 2023.

Examples

```
# Load data
if(requireNamespace("FCPS")){
data(EngyTime, package = "FCPS")
engyTimeData = EngyTime$Data
c1 = engyTimeData[,1]
c2 = engyTimeData[,2]
}else{
c1 = rnorm(n=4000)
c2 = rnorm(n=4000,1,2)
}
# Calculate Densities

densities = SmoothedDensitiesXY(c1,c2)$Densities
# Use DDCAL to cluster the densities
labels = DDCAL(densities, 9)

# Plot Densities according to labels
my_colors = c("#000066", "#3333CC", "#9999FF", "#00FFFF", "#66FF33",
                       "#FFFF00", "#FF9900", "#FF0000", "#990000")
labels = as.factor(labels)
df = data.frame(c1, c2, labels)

if(requireNamespace("ggplot2")){
ggplot2::ggplot(df, ggplot2::aes(c1, c2, color = labels)) +
  ggplot2::geom_point() +
  ggplot2::scale_color_manual(values = my_colors)
}
```

DensityScatter.DDCAL *Scatter density plot [Brinkmann et al., 2023]*

Description

Density estimation (PDE) [Ultsch, 2005] or "SDH" [Eilers/Goeman, 2004] used for a scatter density plot, with clustering of densities with DDCAL [Lux/Rinderle-Ma, 2023] proposed by [Brinkmann et al., 2023].

Usage

```
DensityScatter.DDCAL(X, Y, nClusters = 12, Plotter = "native",
SDHorPDE = TRUE, PDEsample = 10000,
Marginals = FALSE, na.rm=TRUE,
pch = 10, Size = 1,
xlab="x", ylab="y", main = "",lwd = 2,
xlim=NULL,ylim=NULL,Polygon,BW = TRUE,Silent = FALSE, ...)
```

Arguments

X	Numeric vector [1:n], first feature (for x axis values)
Y	Numeric vector [1:n], second feature (for y axis values)
nClusters	Integer defining the number of clusters (colors) used for finding a hard color transition.
Plotter	(Optional) String, name of the plotting backend to use. Possible values are: "native" or "ggplot2"
SDHorPDE	(Optional) Boolean, if TRUE SDH is used to calculate density, if FALSE PDE is used
PDEsample	(Optional) Scalar, Sample size for PDE and/or for ggplot2 plotting. Default is 5000
Marginals	(Optional) Boolean, if TRUE the marginal distributions of X and Y will be plotted together with the 2D density of X and Y. Default is FALSE
na.rm	(Optional) Boolean, if TRUE non finite values will be removed
pch	(Optional) Scalar or character. Indicates the shape of data points, see plot() function or the shape argument in ggplot2. Default is 10
Size	(Optional) Scalar, size of data points in plot, default is 1
xlab	String, title of the x axis. Default: "X", see plot() function
ylab	String, title of the y axis. Default: "Y", see plot() function
main	(Optional) Character, title of the plot. [1:2]
lwd	(Optional) Scalar, thickness of the lines used for the marginal distributions (only needed if Marginals=TRUE), see plot(). Default = 2
xlim	(Optional) numerical vector, min and max of x values to be plottet

ylim	(Optional) numerical vector, min and max of y values to be plottet
Polygon	(Optional) [1:p,1:2] numeric matrix that defines for x and y coordinates a polygon in magenta
BW	(Optional) Boolean, if TRUE ggplot2 will use a white background, if FALSE the typical ggplot2 backgournd is used. Not needed if "native" as Plotter is used. Default is TRUE
Silent	(Optional) Boolean, if TRUE no messages will be printed, default is FALSE
...	Further plot arguments

Details

The DensityScatter.DDCAL function generates the density of the xy data as a z coordinate. Afterwards xyz will be plotted as a contour plot. It assumens that the cases of x and y are mapped to each other meaning that a cbind(x,y) operation is allowed. The colors for the densities in the contour plot are calculated with DDCAL, which produces clusters to evenly distribute the densities in low variance clusters.

In the case of "native" as Plotter, the handle returns NULL because the basic R functon plot() is used

Value

If "ggplot2" as Plotter is used, the ggobj is returned

Note

Support for plotly will be implemented later

Author(s)

Luca Brinkmann, Michael Thrun

References

[Ultsch, 2005] Ultsch, A.: Pareto density estimation: A density estimation for knowledge discovery, In Baier, D. & Werrnecke, K. D. (Eds.), Innovations in classification, data science, and information systems, (Vol. 27, pp. 91-100), Berlin, Germany, Springer, 2005.

[Eilers/Goeman, 2004] Eilers, P. H., & Goeman, J. J.: Enhancing scatterplots with smoothed densities, Bioinformatics, Vol. 20(5), pp. 623-628. 2004.

[Lux/Rinderle-Ma, 2023] Lux, M. & Rinderle-Ma, S.: DDCAL: Evenly Distributing Data into Low Variance Clusters Based on Iterative Feature Scaling, Journal of Classification vol. 40, pp. 106-144, 2023.

[Brinkmann et al., 2023] Brinkmann, L., Stier, Q., & Thrun, M. C.: Computing Sensitive Color Transitions for the Identification of Two-Dimensional Structures, Proc. Data Science, Statistics & Visualisation (DSSV) and the European Conference on Data Analysis (ECDA), p.109, Antwerp, Belgium, July 5-7, 2023.

Examples

```
# Create two bimodial distributions
x1=rnorm(n = 7500,mean = 0,sd = 1)
y1=rnorm(n = 7500,mean = 0,sd = 1)
x2=rnorm(n = 7500,mean = 2.5,sd = 1)
y2=rnorm(n = 7500,mean = 2.5,sd = 1)
x=c(x1,x2)
y=c(y1,y2)

DensityScatter.DDCAL(x, y, Marginals = TRUE)
```

---

inPSphere2D *2D data points in Pareto Sphere*

Description

This function determines the 2D data points inside a ParetoSphere with ParetoRadius.

Usage

```
inPSphere2D(data, paretoRadius=NULL)
```

Arguments

data	numeric matrix of data.
paretoRadius	numeric value. radius of P-spheres. If not given, calculate by the function 'paretoRad'

Value

numeric vector with the number of data points inside a P-sphere with ParetoRadius.

Author(s)

Felix Pape

`InteractiveGate` *Interactive manual gating of scatter pplots*

Description

Allows to draw a polygon around a specifc area in a two-dimensional scatter plot

Usage

```
InteractiveGate(Data, GateVars, Gatename = "Leukocytes", PlotIt = FALSE)
```

Arguments

`Data`	[1:n,1:d] numerical matrix of n cases and d variables. each variable has a name accessible via colnames(Data)
`GateVars`	[1:2] character vector of columns that are selected
`Gatename`	name of the gate
`PlotIt`	TRUE: plots the result

Details

Only polygon gates can be used in the interactive tool.

Value

a LIST,

`Polygon`	[1:p,1:2] numerical matrix of coordinates for variables named by`GateVars` defining the polygon
`Index`	indices of data points within the polygon
`DataInGate`	[1:m,1:d] numerical matrix of n cases and d variables of data points within the polygon

Author(s)

Michael Thrun

See Also

`gs_gate_interactive`

Examples

```
Disk=""
#path=ReDi("AutoGating/09Originale",Disk)
#V=ReadLRN("110001_T1_NB_d11_N33k",path)
#Data=V$Data
#V=InteractiveGate(Data,GateVars = c( "SS", "FS"),PlotIt = T)
```

`InteractiveSequentialGates` *Interactive Sequential Gates*

Description

Performs several consecutive gates interactively on the same data. Only polygon gates can be used in the interactive tool.

Usage

```
InteractiveSequentialGates(Data, GateVarsMat, Gatenames, PlotIt = FALSE)
```

Arguments

`Data`	[1:n,1:d] numerical matrix of n cases and d variables. each variable has a name accessible via colnames(Data)
`GateVarsMat`	[1:2,1:c] character matrix of columns that are selected, each rwo represents one gate
`Gatenames`	[1:c] character vector of names of the sequential gates
`PlotIt`	TRUE: plots the last gate on the prior of last data

Details

In flow cytometry, the process of setting sequential or consecutive gates is known as interactive gating. Interactive gating involves analyzing and selecting specific populations of cells or particles based on their characteristics in a step-by-step manner.

Once the initial gate is set, you can further refine the analysis by examining the population within that gate. By assessing the patterns and characteristics of the cells within the initial gate, you can determine which parameters are relevant for subsequent gating.

Based on the analysis of the first gate, you can then create a second gate or subsequent gates to isolate more specific subsets of cells. This process involves creating new scatter plots based on the relevant parameters identified earlier. The successive gates are usually set based on the characteristics observed within the previously gated populations.

Value

a LIST,

`Polygons`	[1:c] list named by Gatenames, each element is a [1:p,1:2] numerical matrix of coordinates for variables named by`GateVars` defining the polygon
`Indices`	[1:c] list named by Gatenames, each element defines the indices of data points within the polygon with regards to the data of the prior polygon (gate)
`DataInGates`	[1:c] list named by Gatenames, each element is a [1:m,1:d] numerical matrix of n cases and d variables of data points within the current polygon

Author(s)

Michael Thrun

See Also

gs_gate_interactive

Examples

```
Disk=""
#path=ReDi("AutoGating/09Originale",Disk)
#read data
#filename="110001_T1_NB_d11_N33k"
#V=dbt.DataIO::ReadLRN(filename,path)
#Data=V$Data
#find out names of given parameters
#colnames(Data)
#select parameters, each two define one gate
#GateVarsMat=rbind(c("SS", "FS"),c("CD45", "CD19"))
#name the gates, the first one is overall data, every next one is a consecutive gate
#base on the gated information of the previous gate
#Gatenames=c("Lymphocytes","Leukocytes")
#apply interactive function
#GateInfo=ScatterDensity::InteractiveSequentialGates(Data,
#GateVarsMat=GateVarsMat,Gatenames =Gatenames#,PlotIt = T)
```

PDEscatter *Scatter Density Plot*

Description

Concept of Pareto density estimation (PDE) proposed for univsariate data by [Ultsch, 2005] and comparet to varius density estimation techniques by [Thrun et al., 2020] for univariate data is here applied for a scatter density plot. It was also applied in [Thrun and Ultsch, 2018] to bivariate data, but is not yet compared to other techniques.

Usage

```
PDEscatter(x,y,SampleSize,

na.rm=FALSE,PlotIt=TRUE,ParetoRadius,sampleParetoRadius,

NrOfContourLines=20,Plotter='native', DrawTopView = TRUE,

xlab="X", ylab="Y", main="PDEscatter",

xlim, ylim, Legendlab_ggplot="value")
```

Arguments

x	Numeric vector [1:n], first feature (for x axis values)
y	Numeric vector [1:n], second feature (for y axis values)
SampleSize	Numeric m, positiv scalar, maximum size of the sample used for calculation. High values increase runtime significantly. The default is that no sample is drawn
na.rm	Function may not work with non finite values. If these cases should be automatically removed, set parameter TRUE
ParetoRadius	Numeric, positiv scalar, the Pareto Radius. If omitted (or 0), calculate by paretoRad.
sampleParetoRadius	Numeric, positiv scalar, maximum size of the sample used for estimation of "kernel", should be significantly lower than SampleSize because requires distance computations which is memory expensive
PlotIt	TRUE: plots with function call FALSE: Does not plot, plotting can be done using the list element Handle -1: Computes density only, does not perfom any preperation for plotting meaning that Handle=NULL
NrOfContourLines	Numeric, number of contour lines to be drawn. 20 by default.
Plotter	String, name of the plotting backend to use. Possible values are: "native", "ggplot", "plotly"
DrawTopView	Boolean, True means contur is drawn, otherwise a 3D plot is drawn. Default: TRUE
xlab	String, title of the x axis. Default: "X", see plot() function
ylab	String, title of the y axis. Default: "Y", see plot() function
main	string, the same as "main" in plot() function
xlim	see plot() function
ylim	see plot() function
Legendlab_ggplot	String, in case of Plotter="ggplot" label for the legend. Default: "value"

Details

The PDEscatter function generates the density of the xy data as a z coordinate. Afterwards xyz will be plotted either as a contour plot or a 3d plot. It assumens that the cases of x and y are mapped to each other meaning that a cbind(x,y) operation is allowed. This function plots the PDE on top of a scatterplot. Variances of x and y should not differ by extreme numbers, otherwise calculate the percentiles on both first. If DrawTopView=FALSE only the plotly option is currently available. If another option is chosen, the method switches automatically there.

The method was succesfully used in [Thrun, 2018; Thrun/Ultsch 2018].

PlotIt=FALSE is usefull if one likes to perform adjustements like axis scaling prior to plotting with **ggplot2** or **plotly**. In the case of "native"" the handle returns NULL because the basic R functon plot() is used

Value

List of:

X	Numeric vector [1:m],m<=n, first feature used in the plot or the kernels used
Y	Numeric vector [1:m],m<=n, second feature used in the plot or the kernels used
Densities	Numeric vector [1:m],m<=n, Number of points within the ParetoRadius of each point, i.e. density information
Matrix3D	1:n,1:3] marix of x,y and density information
ParetoRadius	ParetoRadius used for PDEscatter
Handle	Handle of the plot object. Information-string if native R plot is used.

Note

MT contributed with several adjustments

Author(s)

Felix Pape

References

[Thrun/Ultsch, 2018] Thrun, M. C., & Ultsch, A. : Effects of the payout system of income taxes to municipalities in Germany, in Papiez, M. & Smiech,, S. (eds.), Proc. 12th Professor Aleksander Zelias International Conference on Modelling and Forecasting of Socio-Economic Phenomena, pp. 533-542, Cracow: Foundation of the Cracow University of Economics, Cracow, Poland, 2018.

[Ultsch, 2005] Ultsch, A.: Pareto density estimation: A density estimation for knowledge discovery, In Baier, D. & Werrnecke, K. D. (Eds.), Innovations in classification, data science, and information systems, (Vol. 27, pp. 91-100), Berlin, Germany, Springer, 2005.

[Thrun et al., 2020] Thrun, M. C., Gehlert, T. & Ultsch, A.: Analyzing the Fine Structure of Distributions, PLoS ONE, Vol. 15(10), pp. 1-66, DOI doi:10.1371/journal.pone.0238835, 2020.

Examples

```
#taken from [Thrun/Ultsch, 2018]
if(requireNamespace("DataVisualizations")){
data("ITS",package = "DataVisualizations")
data("MTY",package = "DataVisualizations")
Inds=which(ITS<900&MTY<8000)
plot(ITS[Inds],MTY[Inds],main='Bimodality is not visible in normal scatter plot')


PDEscatter(ITS[Inds],MTY[Inds],xlab = 'ITS in EUR',

ylab ='MTY in EUR' ,main='Pareto Density Estimation indicates Bimodality' )


}
```

PointsInPolygon *PointsInPolygon*

Description

Defines a Cls based on points in a given polygon.

Usage

```
PointsInPolygon(Points, Polygon, PlotIt = FALSE, ...)
```

Arguments

Points	[1:n,1:2] xy cartesian coordinates of a projection
Polygon	Numerical matrix of 2 columns defining a closed polygon
PlotIt	TRUE: Plots marked points
...	BMUorProjected: Default == FALSE, If TRUE assuming BestMatches of ESOM instead of Projected Points main: title of plot Further Plotting Arguments,xlab etc used in Classplot

Details

We assume that polygon is closed, i.e., that the last point connects to the fist point

Value

Numerical classification vector Cls with 1 = outside polygon and 2 = inside polygon

Author(s)

Michael Thrun

See Also

Classplot

Examples

```
XY=cbind(runif(80,min = -1,max = 1),rnorm(80))
#closed polygon
polymat <- cbind(x = c(0,1,1,0), y = c(0,0,1,1))
Cls=PointsInPolygon(XY,polymat,PlotIt = TRUE)
```

PolygonGate *PolygonGate*

Description

A specific Gate defined by xy coordinates that result in a closed polygon is applied to the flowcytometry data.

Usage

```
PolygonGate(Data, Polygon, GateVars,  PlotIt = FALSE, PlotSampleSize = 1000)
```

Arguments

Data	numerical matrix n x d
Polygon	numerical marix of two columns defining the coordiantes of the polygon. polygon assumed to be closed, i.e.,last coordinate connects to first coordinate.
GateVars	vector, either column index in Data of X and Y coordinates of gate or its variable names as string
PlotIt	if TRUE: plots a sample of data in the two selected variables and marks point inside the gate as yellow and outside as magenta
PlotSampleSize	size pof the plottet sample

Details

Gates are alwaxs two dimensional, i.e., require two filters, although all dimensions of data are filted by the gates. Only high-dimensional points inside the polygon (gate) are given back

Value

list of

DataInGate	m x d numerical matrix with m<=n of data points inside the gate
InGateInd	index of length m for the datapoints in original matrix

Note

if GateVars is not found a text is given back which will state this issue

Author(s)

Michael Thrun

See Also

PointsInPolygon

Examples

```
Data <- matrix(runif(1000), ncol = 10)
colnames(Data)=paste0("GateVar",1:ncol(Data))
poly <- cbind(x = c(0.2,0.5,0.8), y = c(0.2,0.8,0.2))
#set plotit TRUE for understanding the example

#Select indect
V=PolygonGate(Data,poly,c(5,8),PlotIt=FALSE,100)

#select var name
V=PolygonGate(Data,poly,c("GateVar5","GateVar8"),PlotIt=FALSE,100)
```

SmoothedDensitiesXY *Smoothed Densities X with Y*

Description

Density is the smothed histogram density at [X,Y] of [Eilers/Goeman, 2004]

Usage

```
SmoothedDensitiesXY(X, Y, nbins, lambda, Xkernels, Ykernels, PlotIt = FALSE)
```

Arguments

X	Numeric vector [1:n], first feature (for x axis values)
Y	Numeric vector [1:n], second feature (for y axis values), nbins= nxy => the nr of bins in x and y is nxy nbins = c(nx,ny) => the nr of bins in x is nx and for y is ny
nbins	number of bins, nbins =200 (default)
lambda	smoothing factor used by the density estimator or c() default: lambda = 20 which roughly means that the smoothing is over 20 bins around a given point.
Xkernels	bin kernels in x direction are given
Ykernels	bin kernels y direction are given
PlotIt	FALSE: no plotting, TRUE: simple plot

Details

lambda has to chosen by the user and is a sensitive parameter.

Value

List of:

`Densities`	numeric vector [1:n] is the smothed density in 3D
`Xkernels`	numeric vector [1:nx], nx defined by `nbins`, such that mesh(Xkernels,Ykernels,F) form the (not NaN) smothed densisties
`Ykernels`	numeric vector [1:ny], nx defined by `nbins`, such that mesh(Xkernels,Ykernels,F) form the (not NaN) smothed densisties
`hist_F_2D`	matrix [1:nx,1:ny] beeing the smoothed 2D histogram
`ind`	an index such that `Densities = hist_F_2D[ind]`

Author(s)

Michael Thrun

References

[Eilers/Goeman, 2004] Eilers, P. H., & Goeman, J. J.: Enhancing scatterplots with smoothed densities, Bioinformatics, Vol. 20(5), pp. 623-628.DOI: doi:10.1093/bioinformatics/btg454, 2004.

Examples

```
if(requireNamespace("DataVisualizations")){
data("ITS",package = "DataVisualizations")
data("MTY",package = "DataVisualizations")
Inds=which(ITS<900&MTY<8000)
V=SmoothedDensitiesXY(ITS[Inds],MTY[Inds])
}else{
#sample random data
ITS=rnorm(1000)
MTY=rnorm(1000)
V=SmoothedDensitiesXY(ITS,MTY)
}
```

Index