Package 'inflection'

October 13, 2022

Type Package

Title Finds the Inflection Point of a Curve

Version 1.3.6

Date 2022-06-15

Author Demetris T. Christopoulos

Maintainer Demetris T. Christopoulos <dchristop@econ.uoa.gr>

Description Implementation of methods Extremum Surface Estimator (ESE) and Extremum Distance Estimator (EDE) to identify the inflection point of a curve . Christopoulos, DT (2014) <doi:10.48550/arXiv.1206.5478> . Christopoulos, DT (2016) <https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf> . Christopoulos, DT (2016) <doi:10.2139/ssrn.3043076> .

License GPL (>= 2)

URL https://CRAN.R-project.org/package=inflection

Imports parallel, stats, graphics, grDevices

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-15 13:10:02 UTC

R topics documented:

inflection-package 2
bede 6
bese 8
check_curve 9
d2uik 11
ede 13
edeci 15
ese 16

findipiterplot 18
findipl 21
findiplist 23
lin2 25
table_01 26
table_02 26
table_03_04 27
table_05_06 28
table_08_09 29
table_10_11 30
table_13 31
table_14_15 31
table_17_18 32
table_19_20 33
uik 34

Index **36**

`inflection-package` *Finds the Inflection Point of a Curve*

Description

Implementation of methods Extremum Surface Estimator (ESE), Extremum Distance Estimator (EDE) and their iterative versions BESE and BEDE in order to identify the inflection point of a curve.

Details

The x,y data should be numeric vectors of length at least 4 without missing values.
Main functions for a convex/concave curve are:
ese(x,y,0) for ESE method, see `ese` & iterative version `bese` for BESE
ede(x,y,0) for EDE method, see `ede` & iterative version `bede` for BEDE
findiplist(x,y,0) for both ESE and EDE methods, see `findiplist`

From version 1.3.5 it is also available the implementation of Unit Invariant Knee (UIK) method for the determination of elbow or knee point of a curve. That is extremely useful in a wide range of analyses every time we want to find the optimal number of 'components' from a scree plot (PCA: components, Archetypal Analysis: archetypes, Factor A.: factors, Cluster A..: clusters and so on). In most of the cases uasage is uik(x,y), with x the vector of components, archetypes, clusters, factors or whatever and y the relevant vector Sum of Squared Errors (SSE), see `uik` and `d2uik`.

The methods are defined at [1] & [2] while a detailed reproduction of all Tables for [1] exist in a vignette. All relevant Tables exist as data, for example to load Table 1 of [1] we just use data("table_01").

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

[3]Demetris T. Christopoulos (2016). Introducing Unit Invariant Knee (UIK) As an Objective Choice for Elbow Point in Multivariate Data Analysis Techniques (March 1, 2016). https://www.ssrn.com/abstract=3043076

See Also

ese, bese, ede, bede, findiplist, findipiterplot

Examples

```
#
## Lets create some convex/concave data based on the Fisher-Pry model
##by using 1001 not equal spaced abscissas with data right asymmetry
N=20001;
## Case I: not noisy data
#
set.seed(20190628);
x=sort(runif(N,0,15));y=5+5*tanh(x-5);
#
# t1=Sys.time();
# tese=ese(x,y,0,doparallel = TRUE);#...simple run of ESE
# t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 6.023056 secs
# tese;
##       j1   j2      chi
## ESE 4573 8272 4.834904
tede=ede(x,y,0);tede;#...simple run of EDE
##       j1   j2      chi
## EDE 4418 8867 5.000198
edeci(x,y,0);#...Run EDE and compute 95% Chebyshev c.i.
##       j1   j2      chi k  chi-5*s chi+5*s
## EDE 4418 8867 5.000198 5 4.994605 5.00579
#
# t1=Sys.time();
# eseit=bese(x,y,0,doparallel = TRUE);#...Bisection ESE (BESE)
# t2=Sys.time();
# print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 6.262982 secs
#eseit$iplast #...last estimation for inflection point
## [1] 5.000773
#eseit$iters #...all iterations done...
```

```
#        n            a         b      ESE
# 1  20001 0.0001931784 14.999900 4.835303
# 2   3668 4.4606627093  5.647031 5.053847
# 3   1567 4.6878642635  5.262619 4.975242
# 4    737 4.8696049280  5.154673 5.012139
# 5    376 4.9229470803  5.064312 4.993629
# 6    181 4.9684872106  5.038649 5.003568
# 7     82 4.9806225684  5.015416 4.998019
# 8     35 4.9924177257  5.009629 5.001023
# 9     20 4.9960624950  5.002740 4.999401
# 10    11 4.9980399851  5.001968 5.000004
#
t1=Sys.time();
edeit=bede(x,y,0);#...Bisection EDE (BEDE)
t2=Sys.time();
print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
# Time difference of 0.073102 secs
edeit$iplast #...last estimation for inflection point
## [1]  4.999941
edeit$iters #...all iterations done
##        n            a         b      EDE
## 1  20001 0.0004635123 14.999801 5.000198
## 2   4450 4.1996716394  5.799961 4.999816
## 3   2129 4.5614927786  5.438346 4.999920
## 4   1182 4.7512642248  5.249931 5.000597
## 5    656 4.8580908542  5.143351 5.000721
## 6    365 4.9175169028  5.082218 4.999868
## 7    229 4.9524894403  5.047898 5.000194
## 8    135 4.9723928177  5.027658 5.000025
## 9     77 4.9835896748  5.016056 4.999823
## 10    46 4.9904827878  5.009340 4.999912
## 11    28 4.9945713510  5.005444 5.000008
## 12    15 4.9968973151  5.003045 4.999971
## 13    11 4.9978968780  5.001986 4.999941
# t1=Sys.time();
# A=findiplist(x,y,0,doparallel=TRUE);#...Run both ESE & EDE at once...
# t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 5.143354 secs
#A
##       j1   j2      chi
## ESE 4573 8272 4.834904
## EDE 4418 8867 5.000198
## Let's plot some interesting approximately results.
# plot(x,y,type="l",xaxt="n",lwd=2);axis(1,at=seq(0,x[N]));
# lines(c(x[1],x[A[1,2]]),c(y[1],y[A[1,2]]),col="green",lty=2);
# lines(c(x[N],x[A[1,1]]),c(y[N],y[A[1,1]]),col="blue",lty=2);
# lines(c(x[1],x[N]),c(y[1],y[N]),col="black",lty=2);
# abline(v=A[,3],col=c('blue','red'),lty=2);
# points(x[A[1,1:2]],y[A[1,1:2]], type = "p",pch = 19,col="blue",font=2);
# points(x[A[2,1:2]],y[A[2,1:2]], type = "p",pch = 19,col="red",font=2);
# text(A[1,3]-0.5,0,expression(chi[S]),font=2,col='blue');
# text(A[2,3]+0.5,0,expression(chi[D]),font=2,col='red');
# grid();
```

```
#
### Case II: noisy data
#
set.seed(20190628);
x=sort(runif(N,0,15));
r=0.1;y=5+5*tanh(x-5)+rnorm(N,0,0.05);
#
# t1=Sys.time();
# tese=ese(x,y,0,doparallel = TRUE);#...simple run of ESE
# t2=Sys.time();
# print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 4.878437 secs
#tese
##       j1   j2      chi
## ESE 4684 8412 4.936269
tede=ede(x,y,0);tede;#...simple run of EDE
#       j1   j2      chi
# EDE 4190 8856 4.909071
edeci(x,y,0);#...Run EDE and compute 95% Chebyshev c.i.
##       j1   j2      chi k  chi-5*s  chi+5*s
## EDE 4190 8856 4.909071 5 4.659396 5.158746
# t1=Sys.time();
# eseit=bese(x,y,0,doparallel = TRUE);#...Bisection ESE (BESE)
# t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 6.288019 secs
#eseit$iplast#...last estimation for inflection point
## [1] 4.94072
#eseit$iters#...all iterations done...
##       n            a         b      ESE
## 1 20001 0.0004635123 14.999801 4.936269
## 2  3729 4.4285049755  5.538487 4.983496
## 3  1475 4.7297708667  5.413794 5.071782
## 4   932 4.7838335764  5.097607 4.940720
#
t1=Sys.time();
edeit=bede(x,y,0);#...Bisection EDE (BEDE)
t2=Sys.time();
print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 0.01700807 secs
edeit$iplast#...last estimation for inflection point
## [1] 5.071782
edeit$iters#...all iterations done
##       n            a         b      EDE
## 1 20001 0.0004635123 14.999801 4.909071
## 2  4667 4.2343777127  5.704010 4.969194
## 3  1938 4.7297708667  5.413794 5.071782
#
# t1=Sys.time();
# A=findiplist(x,y,0,doparallel=TRUE);#...Run both ESE & EDE at once...
# t2=Sys.time();
# print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
## Time difference of 5.037386 secs
#A
```

```
##       j1   j2      chi
## ESE 4684 8412 4.936269
## EDE 4190 8856 4.909071
#
## Let's plot some interesting approximately results.
# plot(x,y,type="l",xaxt="n",lwd=2);axis(1,at=seq(0,x[N]));
# lines(c(x[1],x[A[1,2]]),c(y[1],y[A[1,2]]),col="green",lty=2);
# lines(c(x[N],x[A[1,1]]),c(y[N],y[A[1,1]]),col="blue",lty=2);
# lines(c(x[1],x[N]),c(y[1],y[N]),col="black",lty=2);
# abline(v=A[,3],col=c('blue','red'),lty=2);
# points(x[A[1,1:2]],y[A[1,1:2]], type = "p",pch = 19,col="blue",font=2);
# points(x[A[2,1:2]],y[A[2,1:2]], type = "p",pch = 19,col="red",font=2);
# text(A[1,3]-0.5,0,expression(chi[S]),font=2,col='blue');
# text(A[2,3]+0.5,0,expression(chi[D]),font=2,col='red');
# grid();
## Close device
#dev.off()
##
## Load data used for Tables of [1]
data("table_01")
dh=table_01
plot(dh,pch=19,cex=0.1)
A=findiplist(dh$x,dh$y,0)
A
abline(v=A[1,3],col='blue')
abline(v=A[2,3],col='red')
##
```

---

bede *Bisection Extremum Distance Estimator Method*

Description

It iterates in a way similar to the well known bisection method in root finding, with the only exception that our $[a_n, b_n]$ intervals contain the inflection point now and the rule for choosing them follows definitions and Lemmas of [1], [2].

Usage

```
bede(x, y, index)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
index	If data is convex/concave then index=0 If data is concave/convex then index=1

Details

It is the fastest solution for very large data sets, over one million rows.

Value

It returns a list of two elements:

`iplast`	the last EDE estimation that was found
`iters`	a matrix with 4 columns ("n", "a", "b", "EDE") that give the number of x-y pairs used at each iteration, the [a,b] range where we searched and the EDE estimated inflection point.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also the simple version `ede`, `edeci` and iterations plot using `findipiterplot`.

Examples

```
#
#Fisher-pry model with heavy noise, unequal spaces
#and 1 million cases:
N=10^6+1;
set.seed(2017-05-11);x=sort(runif(N,0,10));y=5+5*tanh(x-5)+runif(N,-1,1);
#
ptm <- proc.time()
tede=ede(x,y,0);tede;proc.time() - ptm
#         j1     j2      chi
# EDE 351061 648080 4.997139
# user  system elapsed
# 0.02    0.02    0.05
#
```

bese *Bisection Extremum Surface Estimator Method*

Description

It iterates in a way similar to the well known bisection method in root finding, with the only exception that our $[a_n, b_n]$ intervals contain the inflection point now and the rule for choosing them follows definitions and Lemmas of [1], [2]. It uses parallel computing under user request.

Usage

```
bese(x, y, index, doparallel = FALSE)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
index	If data is convex/concave then index=0 If data is concave/convex then index=1
doparallel	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

This function is suitable for making a 'fine tuning' while searching for inflection point. For very large data sets it is better using first EDE method, see ede. Then we apply BESE at a smaller range.

Value

It returns a list of two elements:

iplast	the last estimation found
iters	a matrix with 4 columns ("n", "a", "b", "ESE") that give the number of x-y pairs used at each iteration, the [a,b] range where we searched and the ESE estimated inflection point.

Note

Parallel computing was added in version 1.3

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos, Developing methods for identifying the inflection point of a convex/ concave curve. arXiv:1206.5478v2 [math.NA], https://arxiv.org/pdf/1206.5478v2.pdf, 2014

[2]Demetris T. Christopoulos, On the efficient identification of an inflection point,International Journal of Mathematics and Scientific Computing,(ISSN: 2231-5330), vol. 6(1), https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf, 2016

See Also

See also the simple version ese and iterations plot using findipiterplot.

Examples

```
#Fisher-pry model with noise and 50k cases:
N=5*10^4+1;
set.seed(2017-05-11);x=seq(0,15,length.out = N);y=5+5*tanh(x-5)+runif(N,-0.25,0.25);
#We first run BEDE to find a smaller neighborhood for inflection point
iters=bede(x,y,0)$iters;
iters;
#Now we find last interval
ab=apply(iters[dim(iters)[1],c('a','b')],2,function(xx,x){which(x==xx)},x);ab;
#Apply BESE to that
eseit=bese(x[ab[1]:ab[2]],y[ab[1]:ab[2]],0)
eseit$iplast
eseit$iters
#Or apply directly to data with doparallel=TRUE
#
#t1=Sys.time();
#eseit=bese(x,y,0,doparallel = TRUE);#...Bisection ESE (BESE)
#t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
# Time difference of 56.14608 secs
#eseit$iplast#...last estimation for inflection point
# [1] 5.0241
#eseit$iters#...all iterations done...
#       n      a       b     ESE
# 1 50001 0.0000 15.0000 4.81740
# 2  9375 4.4721  5.6505 5.06130
# 3  3929 4.7007  5.2758 4.98825
# 4  1918 4.8654  5.1828 5.02410
#Better accuracy, slightly more time, provided that there exist multi cores.
#plot(eseit$iters$ESE,type='b');abline(h=5,col='blue',lwd=3)
#
```

---

check_curve *Checks a curve and decides for its convexity type*

Description

Given a planar curve with discrete (xi,yi) points this function can find if it is convex, concave or for the sigmoid case if it is convex/concave or concave/convex.

Usage

```
check_curve(x, y)
```

Arguments

x	The numeric vector of x-abscissas
y	The numeric vector of y-abscissas

Details

It uses the function `findipl` which provides us with a consistent estimator for the surfaces left and right that are used here, see [1],[2] for more details.

Value

A list with members:

1. ctype, the convexity type of the curve
2. index, the index that can be used from other functions to identify the inflection point

Note

If we do not have a visual inspection of our data, then this function is useful because it can automate the usage of all other functions that compute the inflection point.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

[3]Bardsley, W. G. & Childs, R. E. Sigmoid curves, non-linear double-reciprocal plots and allosterism Biochemical Journal, Portland Press Limited, 1975, 149, 313-328

See Also

findipl, ese, ede, bese, bede .

Examples

```
## Lets create a really hard data set, an exxample of a "2:2 function" taken from [3]
## This function for x>0 has an inflectrion point at
## x = -1/8+(1/24)*sqrt(381) = 0.6883008876 ~0.69
## We want to see ,if function 'check_curve()' will proper classify it,
## given that we used [0.2,4] as definition range.
f=function(x){(1/8*x+1/2*x^2)/(1+1/8*x+1/2*x^2)}
x=seq(0.2,4,0.05)
y=f(x)
plot(x,y,pch=19,cex=0.5)
cc=check_curve(x,y)
cc
## $ctype
## [1] "convex_concave"
##
## $index
## [1] 0
##
## Yes it found it.
```

---

d2uik — *Implementation of UIK method to the approximation for second order derivative of data points*

Description

It finds the UIK estimation for elbow or knee point of a curve, see [1] for details, but now it applies the method to the properly prepared approximation of second derivative for data points.

Usage

```
d2uik(x, y)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 6.
y	The numeric vector of y-abscissas, must be of length at least 6.

Details

A preprocessing step is initially done in order to ensure that data points have unique x-abscissas:

- if no multiple entries exist for the same xi's, then we proceed with initial data frame
- if multiple entries exist for the same xi's, then we average entries and proceed with the new aggregated data frame

After we take the approximation of second derivative for the discrete data points as given from the second order forward divided differences estimation. Finally we apply UIK method on the absolute values of the estimated second derivatives.

Value

It returns the x-abscissa which is the estimation for the knee point.

Warning

Please use this function as a supplementary to uik.

Note

This function has been created for robustnes reasons:

- many times the knee point is dependent on the number N of used data points
- in those cases by applying UIK method on second derivatives we obtain a result that does not depend on N

Author(s)

Demetris T. Christopoulos

References

[1] Christopoulos, Demetris T., Introducing Unit Invariant Knee (UIK) As an Objective Choice for Elbow Point in Multivariate Data Analysis Techniques (March 1, 2016). Available at SSRN: https://ssrn.com/abstract=3043076 or http://dx.doi.org/10.2139/ssrn.3043076

See Also

uik

Examples

```
## Lets create a convex data set with x from 1 to 10
x=seq(1,10,1)
y=1/x
plot(x,y)
uik1=uik(x,y)
uik1
## [1] 3
uik2=d2uik(x,y)
uik2
## [1] 3
## Lets extend it from 1 to 15
x=seq(1,15,1)
y=1/x
plot(x,y)
uik1=uik(x,y)
```

```
uik1
## [1] 4
uik2=d2uik(x,y)
uik2
## [1] 3
## We observe the d2uik remains 3
## Lets do the same job with noisy data sets:
set.seed(20190625)
x=seq(1,10,1)
y=1/x+runif(length(x),-0.02,0.02)
plot(x,y)
uik1=uik(x,y)
uik1
## [1] 3
uik2=d2uik(x,y)
uik2
## [1] 3
## Extension to 1:15
x=seq(1,15,1)
y=1/x+runif(length(x),-0.02,0.02)
plot(x,y)
uik1=uik(x,y)
uik1
## [1] 5
uik2=d2uik(x,y)
uik2
## [1] 3
## Again d2uik is stable
```

---

ede *The Extremum Distance Estimator (EDE) for finding the inflection point of a convex/concave curve*

Description

Implementation of EDE method as defined in [1] and [2] by giving a simple output of the method.

Usage

```
ede(x, y, index)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
index	If data is convex/concave then index=0 If data is concave/convex then index=1

Details

We also obtain the x_{F_1}, x_{F_2} points, see [1], [2].

Value

A matrix of size 1 x 3 is returned with elements:

`A(1,1)`	The index j_{F_1} for EDE method
`A(1,2)`	The index j_{F_2} for EDE method
`A(1,3)`	The Extremum Distance Estimator (EDE) for inflection point

Note

This function is for real big data sets, more than one million rows. It is the fastest available method, see [2] for comparison to other methods.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also the iterative version `bede` and iterations plot using `findipiterplot`.

Examples

```
#
#Fisher-pry model with heavy noise, unequal spaces
#and 1 million cases:
N=10^6+1;
set.seed(2017-05-11);x=sort(runif(N,0,10));y=5+5*tanh(x-5)+runif(N,-1,1);
#
ptm <- proc.time()
tede=ede(x,y,0);tede;proc.time() - ptm
#         j1     j2      chi
# EDE 351061 648080 4.997139
# user  system elapsed
# 0.01    0.00    0.01
#
```

edeci *An improved version of EDE that provides us with a Chebyshev confidence interval for inflection point*

Description

It computes except from the common EDE output the Chebyshev confidence interval based on Chebyshev inequality.

Usage

```
edeci(x, y, index, k = 5)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
index	If data is convex/concave then index=0 If data is concave/convex then index=1
k	According to Chebyshev's inequality we find a relevant Chebyshev confidence interval of the form $[\mu - k\,\sigma, \mu + k\,\sigma]$

Details

We define as Chebyshev confidence interval the

$$[\mu - k\,\sigma, \mu + k\,\sigma]$$

where usually k=5 because it corresponds to 96%, while an estimator of σ is given by s_D, see Eq. (29) of [2]:

$$s_D^2 = \frac{1}{2}\,s^2 = \frac{1}{2}\,\sum_{i=1}^{n}(\frac{y_i - y_{i-1}}{2})^2$$

Value

A one row matrix with elements the output of EDE, the given k and the Chebyshev c.i.

Note

This function works better if the noise is of a "zig-zag"" pattern, see [1].

Author(s)

Demetris T. Christopoulos

References

Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also the simple ede and iterative version bede.

Examples

```
#
#Gompertz model with noise, unequal spaces
#and 1 million cases:
N=10^6+1;
set.seed(2017-05-11);x=sort(runif(N,0,10));y=10*exp(-exp(5)*exp(-x))+runif(N,-0.05,0.05);
#EDE one time only
ede(x,y,0)
#         j1     j2      chi
# EDE 372064 720616 5.465584
#Not so close to the exact point
#Let's reduce the size using BEDE
iters=bede(x,y,0)$iters;iters;
#         n            a        b      EDE
# 1 1000001 2.273591e-05 9.999994 5.465584
# 2  348553 4.237734e+00 6.017385 5.127559
# 3  177899 4.573986e+00 5.499655 5.036821
#Now we choose last interval, in order for EDE to be applicable on next run
ab=apply(iters[dim(iters)[1]-1,c('a','b')],2,function(xx,x){which(x==xx)},x);ab;
#      a      b
# 423480 601378
#Apply edeci...
edeci(x[ab[1]:ab[2]],y[ab[1]:ab[2]],0)
#        j1     j2      chi k  chi-5*s  chi+5*s
# EDE 33355 126329 5.036821 5 4.892156 5.181485
#Very close to the true inflection point.
#
```

---

ese *The Extremum Surface Estimator (ESE) for finding the inflection point of a convex/concave curve*

Description

Implementation of ESE method as defined in [1] and [2] by giving a simple output of the method. Use of parallel computing under user request.

Usage

```
ese(x, y, index, doparallel = FALSE)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
index	If data is convex/concave then index=0 If data is concave/convex then index=1
doparallel	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

If data is from an unknown function and without noise then we can find the inflection point by a way similar to bisection method way for root finding. If data is noisy, then we have two consistent estimators of the trapezoidal estimated inflection point, i.e. we consistently estimate what we could find by computing the relevant areas with elementary trapezoids. This method is not so fast as ede but it can be used for a fine-tuning of the result returned be EDE.

Value

A matrix of size 1 x 3 is returned with elements:

A(1,1)	The index j-right for ESE method
A(1,2)	The index j-left for ESE method
A(1,3)	The Extremum Surface Estimator (ESE) for inflection point

Note

Use doparallel=TRUE option when you have relative large data sets (N>20000).
For large data sets (one million rows) it is better to use first ede or bede in order to locate a smaller neighbourhood of the inflection point. Then ese gives a better estimation, since it uses surfaces and not distances from total chord.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also the iterative version bese and iterations plot using findipiterplot.

Examples

```
#
#
#Fisher-pry model with heavy noise and unequal spaces, relative large data set:
#N=20001;
#set.seed(2017-05-11);x=sort(runif(N,0,10));y=5+5*tanh(x-5)+runif(N,-1,1);
#plot(x,y,type='l',ylab=expression(5+5*tanh(x-5)+epsilon[i]))
#
#t1=Sys.time();
#tese=ese(x,y,0,doparallel = TRUE);
#t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
#Time difference of 7.641404 secs
#tese;abline(v=tese[3],col='blue')
#       j1    j2      chi
# ESE 7559 12790 5.078434
#Compare with serial version (don't run):
#
# t1=Sys.time();
# tese=ese(x,y,0,doparallel = FALSE);
# t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
# #Time difference of 24.24364 secs
# tese;
#      j1    j2      chi
# ESE 7559 12790 5.078434
#
```

`findipiterplot` *A function to show implementation of BESE and BEDE methods by plotting their iterative convergence*

Description

We apply the BESE and BEDE methods, we plot results showing the bisection like convergence to the true inflection point. One plot is created for BESE and another one for BEDE. If it is possible, then we compute 95% confidence intervals for the results. Parallel computing also available under user request.

Usage

```
findipiterplot(x, y, index, plots = TRUE, ci = FALSE, doparallel = FALSE)
```

Arguments

`x`	The numeric vector of x-abscissas, must be of length at least 4.
`y`	The numeric vector of the noisy or not y-ordinates
`index`	If data is convex/concave then index=0 If data is concave/convex then index=1
`plots`	When plots=TRUE the plot commands are executed (default value = TRUE)

ci	When ci=TRUE the 95% confidence intervals are computed, if sufficient results are available (default value = FALSE)
doparallel	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

It applies iteratively when that is theoretically allowable the methods ESE, EDE, stores all useful results and according to the input computes 95% confidence intervals and plots the two sequences (BESE, BEDE). Useful for a graphical investigation of the inflection point.

Value

ans$first	The output of first run for ESE and EDE methods
ans$BESE	The vector of BESE iterations
ans$BEDE	The vector of BEDE iterations
ans$aesmout	Mean, Std Deviation and 95 % confidence interval for all BESE iterations, if possible
ans$aedmout	Mean, Std Deviation and 95 % confidence interval for all BEDE iterations, if possible
ans$xysl	A list of xy data frames containing the data used for every ESE iteration
ans$xydl	A list of xy data frames containing the data used for every EDE iteration

Note

Non direct methods have been removed in version 1.3 due to their limited functionality.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also bese and bede.

Examples

```
#
#Lets create some convex/concave data based on the Fisher-Pry model, without noise
f=function(x){5+5*tanh(x-5)};xa=0;xb=15;
set.seed(12345);x=sort(runif(5001,xa,xb));y=f(x);
#
t1=Sys.time();
a<-findipiterplot(x,y,0,TRUE,TRUE,FALSE);
t2=Sys.time();print(as.POSIXlt(t2, "GMT")-as.POSIXlt(t1, "GMT"),quote=F);
#Time difference of 2.692897 secs
#Lets see available results
ls(a)
# [1] "aedmout" "aesmout" "BEDE"    "BESE"    "first"   "xydl"    "xysl"
a$first;#Show first solution
#       i1   i2  chi_S,D
# ESE 1128 2072 4.835633
# EDE 1091 2221 4.999979
a$BESE;#Show ESE iterations
#                       1        2        3        4        5        6        7        8
# ESE iterations 4.835633 5.054775 4.978086 5.011331 4.993876 5.003637 4.998145 4.999782
a$BEDE;#Show EDE iterations
#                       1        2        3        4        5        6        7        8
# EDE iterations 4.999979 4.996327 4.997657 5.001511 4.996464 5.000629 4.999149 4.999885
#        9       10
# 5.000082 4.999782
a$aesmout;#Statistics and 95%c c.i. for ESE
#                mean      sdev   95%(l)   95%(r)
# ESE method 4.984408 0.0640699 4.930844 5.037972
a$aedmout;#Statistics and 95%c c.i. for EDE
#                mean        sdev   95%(l) 95%(r)
# EDE method 4.999146 0.001753223 4.997892 5.0004
#
#Look how bisection based method (BESE) converges in 8 steps...
#
lapply(a$xysl,summary);
# [[1]]
# x                  y
# Min.   : 0.006405   Min.   : 0.00046
# 1st Qu.: 3.802278   1st Qu.: 0.83521
# Median : 7.583006   Median : 9.94325
# Mean   : 7.504537   Mean   : 6.68942
# 3rd Qu.:11.240944   3rd Qu.: 9.99996
# Max.   :14.994895   Max.   :10.00000
#
#...
#
#
# [[8]]
# x              y
# Min.   :4.978   Min.   :4.891
# 1st Qu.:4.988   1st Qu.:4.938
# Median :5.004   Median :5.018
```

```
# Mean   :4.999   Mean   :4.997
# 3rd Qu.:5.009   3rd Qu.:5.043
# Max.   :5.018   Max.   :5.090
#
# and BEDE in 10 steps:
#
lapply(a$xydl,summary)
# [[1]]
# x                  y
# Min.   : 0.006405   Min.   : 0.00046
# 1st Qu.: 3.802278   1st Qu.: 0.83521
# Median : 7.583006   Median : 9.94325
# Mean   : 7.504537   Mean   : 6.68942
# 3rd Qu.:11.240944   3rd Qu.: 9.99996
# Max.   :14.994895   Max.   :10.00000
#
# ...
#
# [[10]]
# x               y
# Min.   :4.982   Min.   :4.911
# 1st Qu.:4.993   1st Qu.:4.965
# Median :5.004   Median :5.019
# Mean   :5.001   Mean   :5.007
# 3rd Qu.:5.009   3rd Qu.:5.045
# Max.   :5.018   Max.   :5.090
#
# See also the pdf plots 'ese_iterations.pdf' and 'ede_iterations.pdf'
```

`findipl` *Finds the s-left and s-right for a given internal point x[j]*

Description

From the definitions (1.3), (2.2) of references [1], [2] it is necessary to find s_l and s_r in order to estimate the Extremum Surface Estimator (ESE) of the inflection point.

Usage

```
findipl(x, y, j)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
j	The data index j such that x=x_j

Value

A list is returned that contains

j	The data index j such that x=x_j
x	The corresponding x-abscissa
sl	The value of s-left
sr	The value of s-right

Note

This small function is used when we are scanning for the position of inflection point in ESE method.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also ese.

Examples

```
#
#Lets create some data based on the Fisher-Pry model, without noise:
x<-seq(0,10,by=0.1);y<-5*(1+tanh(x-5));
tese=ese(x,y,0);tese;
#     j1 j2 chi
# ESE 39 63   5
N<-length(x);N
# [1] 101
#We know that total symmetry exists, so for the middle point it is better to compute |sl|=|sr|
j=(N-1)/2+1;j
# [1] 51
#Define the left and right chord:
fl<-function(t){y[1] + (y[j] - y[1]) * (t - x[1]) / (x[j] - x[1])}
fr<-function(t){y[j] + (y[N] - y[j]) * (t - x[j]) / (x[N] - x[j])}
#Find the s-left and s-right:
LR<-findipl(x,y,j);LR;
# [1] 51.000000  5.000000 -9.031459  9.031459
#Show all results in a plot:
```

```
plot(x,y,type="l",col="red")
lines(c(x[1],x[j]),c(y[1],y[j]),type="l",col="green")
lines(c(x[N],x[j]),c(y[N],y[j]),type="l",col="blue")
points(x[j],y[j], type = "p",pch = 19,col="black")
text(2.5,1,round(LR[3],digits=2))
text(6.5,7.5,round(LR[4],digits=2))
#The two surfaces are indeed absolutely equal |sl|=|sr|
#
```

`findiplist` *The Extremum Surface Estimator (ESE) and Extremum Distance Estimator (EDE) methods for finding the inflection point of a convex/concave curve.*

Description

Given the $(x_i, y_i), i = 1, ..., N$ noisy or not data we want to estimate the inflection point of the corresponding curve. The curve can be convex before the inflection point and then concave or vice versa. The ESE and EDE methods are applied and the results are returned as a matrix. Parallel computing is applied under request.

Usage

```
findiplist(x, y, index, doparallel = FALSE)
```

Arguments

`x`	The numeric vector of x-abscissas, must be of length at least 4.
`y`	The numeric vector of the noisy or not y-ordinates, must be of length at least 4.
`index`	If data is convex/concave then index=0 If data is concave/convex then index=1
`doparallel`	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

If data is from an unknown function and without error then we can find the inflection point in a way similar to that of bisection method' s way for a root. If data is noisy, then we have two consistent estimators of the inflection point.

Value

A matrix of size 2 x 3 is returned with elements:

`A(1,1)`	The index j-right for ESE method
`A(1,2)`	The index j-left for ESE method
`A(1,3)`	The Extremum Surface Estimator (ESE) for inflection point

A(2,1)	The index j1 for EDE method
A(2,2)	The index j2 for EDE method
A(2,3)	The Extremum Distance Estimator (EDE) for inflection point, if this method is applicable

Note

It is a simple implementation of both ESE and EDE methods to a data set of at least 4 xy-pairs.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. https://arxiv.org/pdf/1206.5478v2.pdf

[2]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

See also the simple versions ese and ede.

Examples

```
#Lets create some convex/concave data based on the Fisher-Pry model
#by using 1001 not equal spaced abscissas with data right asymmetry
N=1001;
#Case I: data without noise
set.seed(2017-05-11);x=sort(runif(N,0,15));y=5+5*tanh(x-5);
A=findiplist(x,y,0);A;
#      j1  j2      chi
# ESE 242 438 4.848448
# EDE 228 478 5.000907
plot(x,y,type="l",xaxt="n",lwd=2);axis(1,at=seq(0,15));
abline(v=A[,3],col=c("blue","red"))
text(A[1,3]-0.5,0,expression(chi[S]),font=2);
text(A[2,3]+0.5,0,expression(chi[D]),font=2);
grid();
#
###Case II: noisy data
set.seed(2017-05-11);x=sort(runif(N,0,15));y=5+5*tanh(x-5)+rnorm(N,0,0.05);
A=findiplist(x,y,0);A;
#      j1  j2      chi
# ESE 245 437 4.853798
# EDE 245 469 5.030581
```

```
plot(x,y,type="l",xaxt="n",lwd=2);axis(1,at=seq(0,15));
abline(v=A[,3],col=c("blue","red"))
text(A[1,3]-0.5,0,expression(chi[S]),font=2);
text(A[2,3]+0.5,0,expression(chi[D]),font=2);
grid();
#
```

---

lin2 *Linear function defined from two planar points (x1,y1) and (x2,y2)*

Description

It gives the value of the linear function defined from two points (x1,y1) and (x2,y2) at any x

Usage

```
lin2(x1, y1, x2, y2, x)
```

Arguments

x1	the x - abscissa of the first point
y1	the y - ordinate of the first point
x2	the x - abscissa of the second point
y2	the y - ordinate of the second point
x	the x - point where we compute the value of the linear function

Value

It returns the value of the linear function from (x1,y1) to (x2,y2) at an arbitrary x.

Note

The value of a linear function is built by this way for R to be faster than a two-vectors call.

Author(s)

Demetris T. Christopoulos.

Examples

```
x1<-1
y1<-3
x2<-5
y2<-7
x<-10
ylin<-lin2(x1,y1,x2,y2,x)
print(ylin)
```

table_01 *Fisher-Pry sigmoid with total symmetry and no error*

Description

Data used for creating Table 1 of arXiv:1206.5478v2

Usage

```
data("table_01")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 1: Fisher-Pry sigmoid, p=5, total symmetry, n=500, no-error

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_01")
dh=table_01
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
```

table_02 *Fisher-Pry sigmoid with total symmetry and error ~ U(-0.5,0.05)*

Description

Data used for creating Table 2 of arXiv:1206.5478v2

Usage

```
data("table_02")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Fisher-Pry sigmoid, p=5, data symmetry, [a, b] = [2, 8], n=500, error r=0.05

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_02")
dh=table_02
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
```

`table_03_04` *Fisher-Pry sigmoid with data left asymmetry and no error*

Description

Data used for creating Table 3 and 4 of arXiv:1206.5478v2

Usage

```
data("table_03_04")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 3: Fisher-Pry sigmoid, p=5, data left symmetry, [a, b] = [4.2, 8], n=500, no error

Table 4: ESE & EDE iterations for Fisher-Pry sigmoid, p=5, data left asymmetry, [a, b] = [4.2, 8], n=500, no-error

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_03_04")
dh=table_03_04
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

---

table_05_06 *Fisher-Pry sigmoid with data left asymmetry and no error ~ U(-0.05,0.05)*

Description

Data used for creating Table 5 and 6 of arXiv:1206.5478v2

Usage

```
data("table_05_06")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 5: Fisher-Pry sigmoid, p=5, data left symmetry, [a, b] = [4.2, 8], n=500, error r=0.05

Table 6: ESE & EDE iterations for Fisher-Pry sigmoid, p=5, data left asymmetry, [a, b] = [4.2, 8], n=500, error r=0.05

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_05_06")
dh=table_05_06
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

---

table_08_09 *Gompertz non-symmetric sigmoid with no error*

Description

Data used for creating Table 8 and 9 of arXiv:1206.5478v2

Usage

```
data("table_08_09")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 8: Gompertz sigmoid, p=5, asymmetry, n=500, no-error

Table 9: ESE & EDE iterations for Gompertz sigmoid, p=5, asymmetry, [a, b] = [3.5, 8], n=500, no error

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_08_09")
dh=table_08_09
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

table_10_11 *Gompertz non-symmetric sigmoid with error ~ U(-0.05,0.05)*

Description

Data used for creating Table 10 and 11 of arXiv:1206.5478v2

Usage

```
data("table_10_11")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 10: Gompertz sigmoid, p=5, asymmetry, [a, b] = [3.5, 8], n=500, error r=0.05

Table 11: ESE & EDE iterations for Gompertz sigmoid, p=5, asymmetry, [a, b] = [3.5, 8], n=500, error r=0.05

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_10_11")
dh=table_10_11
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

table_13 *A 3rd order polynomial with total symmetry and no error*

Description

Data used for creating Table 13 of arXiv:1206.5478v2

Usage

```
data("table_13")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 13: 3rd order polynomial, total symmetry, p=2.5, n=500, no-error

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_13")
dh=table_13
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

table_14_15 *A 3rd order polynomial with total symmetry and error ~ U(-2,2)*

Description

Data used for creating Table 14 and 15 of arXiv:1206.5478v2

Usage

```
data("table_14_15")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 14: Symmetric 3rd order polynomial, total symmetry, p=2.5, n=500, error r=2.0

Table 15: ESE iterations for 3rd order polynomial, p=2.5, total symmetry, n=500, error r=2.0

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_14_15")
dh=table_14_15
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
```

---

table_17_18 *A 3rd order polynomial with data right symmetry and no error*

Description

Data used for creating Table 17 and 18 of arXiv:1206.5478v2

Usage

```
data("table_17_18")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 17: Symmetric 3rd order polynomial, data right asymmetry, p=2.5, n=500, [-2, 8], no-error

Table 18: ESE & EDE iterations for 3rd order polynomial, p=5, p=2.5, n=500, [-2, 8], no-error

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_17_18")
dh=table_17_18
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

---

table_19_20 *A 3rd order polynomial with data right symmetry and error ~ U(-2,2)*

Description

Data used for creating Table 19 and 20 of arXiv:1206.5478v2

Usage

```
data("table_19_20")
```

Format

A data frame with 501 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Details

Table 19: Symmetric 3rd order polynomial, data right asymmetry, p=2.5, n=500, [-2, 8], error r=2.0

Table 20: ESE & EDE iterations for 3rd order polynomial, p=2.5, n=500, [-2, 8], error r=2.0

References

Christopoulos, DT (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]

Examples

```
data("table_19_20")
dh=table_19_20
plot(dh,pch=19,cex=0.1)
findiplist(dh$x,dh$y,0)
bese(dh$x,dh$y,0)
bede(dh$x,dh$y,0)
```

uik *Implementation of Unit Invariant Knee (UIK) method for finding the knee point of a curve*

Description

It finds the UIK estimation for elbow or knee point of a curve, see [1] for details.

Usage

```
uik(x, y)
```

Arguments

x	The numeric vector of x-abscissas, must be of length at least 4.
y	The numeric vector of y-abscissas, must be of length at least 4.

Details

Given the x, y numeric vectors it first checks the curve by using check_curve and classifies it as convex, concave or convex/concave, concave/convex.

Value

It returns the x-abscissa which is the UIK estimation for the knee point.

Author(s)

Demetris T. Christopoulos

References

[1] Christopoulos, Demetris T., Introducing Unit Invariant Knee (UIK) As an Objective Choice for Elbow Point in Multivariate Data Analysis Techniques (March 1, 2016). Available at SSRN: https://ssrn.com/abstract=3043076 or http://dx.doi.org/10.2139/ssrn.3043076

See Also

check_curve and d2uik

Examples

```
## Lets create a convex data set
x=seq(1,10,0.05)
y=1/x
plot(x,y)
knee=uik(x,y)
knee
```

```
## [1] 3.15
abline(v=knee)
## Lets add noise to them now
set.seed(20190625)
x=seq(1,10,0.05)
y=1/x+runif(length(x),-0.02,0.02)
plot(x,y)
knee=uik(x,y)
knee
## [1] 3.3
abline(v=knee)
```

Index

∗ **bede**
bede, 6
∗ **bese**
bese, 8
∗ **bisection**
bede, 6
bese, 8
findipiterplot, 18
∗ **concave/convex**
check_curve, 9
∗ **concave**
check_curve, 9
∗ **convex/concave**
check_curve, 9
∗ **convex**
check_curve, 9
∗ **d2uik**
d2uik, 11
∗ **datasets**
table_01, 26
table_02, 26
table_03_04, 27
table_05_06, 28
table_08_09, 29
table_10_11, 30
table_13, 31
table_14_15, 31
table_17_18, 32
table_19_20, 33
∗ **edeci**
edeci, 15
∗ **ede**
ede, 13
findipiterplot, 18
findiplist, 23
∗ **ese**
ese, 16
findipiterplot, 18
findiplist, 23
∗ **findipl**
findipl, 21
∗ **inflection**
findipiterplot, 18
findiplist, 23
inflection-package, 2
∗ **iterative**
bede, 6
bese, 8
findipiterplot, 18
∗ **knee**
d2uik, 11
uik, 34
∗ **line**
lin2, 25
∗ **uik**
d2uik, 11
uik, 34

bede, *2, 3*, 6, *11*, *14*, *16, 17*, *19*
bese, *2, 3*, 8, *11*, *17*, *19*

check_curve, 9, *34*

d2uik, *2*, 11, *34*

ede, *2, 3*, *7, 8*, *11*, 13, *16, 17*, *24*
edeci, *7*, 15
ese, *2, 3*, *9*, *11*, 16, *22*, *24*

findipiterplot, *3*, *7*, *9*, *14*, *17*, 18
findipl, *10, 11*, 21
findiplist, *2, 3*, 23

inflection *(*inflection-package*)*, 2
inflection-package, 2

lin2, 25

table_01, 26
table_02, 26

table_03_04, 27
table_05_06, 28
table_08_09, 29
table_10_11, 30
table_13, 31
table_14_15, 31
table_17_18, 32
table_19_20, 33

uik, *2*, *12*, 34