

Package ‘wle’

April 19, 2009

Title Weighted Likelihood Estimation

LazyLoad yes

LazyData yes

Version 0.9-3

Author Claudio Agostinelli <claudio@unive.it>

Maintainer Claudio Agostinelli <claudio@unive.it>

Depends circular

Date June 11, 2006.

Description Approach to the robustness via Weighted Likelihood.

License GPL-2

Repository CRAN

Date/Publication 2006-08-25 08:15:02

R topics documented:

artificial	2
binary	3
cavendish	4
hald	4
mde.vonmises	5
mde.wrappednormal	7
mle.aic	9
mle.aic.summaries	11
mle.cp	12
mle.cp.summaries	14
mle.cv	15
mle.cv.summaries	17
mle.stepwise	18

mle.stepwise.summaries	20
plot.mle.cp	21
plot.wle.cp	22
plot.wle.lm	23
rocky	25
selection	25
wle.aic	26
wle.aic.ar	29
wle.aic.ar.summaries	32
wle.aic.summaries	33
wle.ar	34
wle.binomial	37
wle.cp	39
wle.cp.summaries	42
wle.cv	43
wle.cv.summaries	46
wle.fracdiff	47
wle.gamma	50
wle.lm	52
wle.lm.summaries	55
wle.normal	57
wle.normal.mixture	59
wle.normal.multi	62
wle.normal.multi.summaries	64
wle.normal.summaries	64
wle.onestep	65
wle.onestep.summaries	67
wle.poisson	68
wle.smooth	69
wle.stepwise	71
wle.stepwise.summaries	74
wle.t.test	75
wle.var.test	78
wle.vonmises	80
wle.weights	82
wle.wrappednormal	83

Index**86**

Description

This data set was generated by Hawkins, Bradu and Kass in the 1984 for illustrating some of the merits of a robust technique. The data set consists of 75 observations in four dimensions (one response and three explanatory variables). The first 10 observations are bad leverage points, and the next four points are good leverage points (i.e., their x are outlying, but the corresponding y fit the model quite well).

Usage

```
data(artificial)
```

Format

`artificial` is a data frame with 75 cases (rows) and 4 variables (columns) where the last column is the dependent variable, `y.artificial` and `x.artificial` (as a matrix) are also available.

Source

Hawkins, D.M., Bradu, D., and Kass, G.V. (1984) Location of several outliers in multiple regression data using elemental sets. *Technometrics*, **26**, 197–208.

See Also

Rousseeuw, P.J., and Leroy, A.M. (1987) *Robust regression and outliers detection*, Wiley.

binary

Convert decimal base number to binary base

Description

Convert decimal base number to binary base.

Usage

```
binary(x, dim)
```

Arguments

<code>x</code>	a number in decimal base.
<code>dim</code>	the number of digits, if missing the right number of digits is evaluated.

Value

<code>binary</code>	a vector representing the 'x' number in binary base.
<code>dicotomy</code>	the same as 'binary' but 'TRUE' and 'FALSE' instead of 1 and 0.

Note

the elements of ‘binary’ and ‘dicotomy’ are in reverse order.

Author(s)

Claudio Agostinelli

Examples

```
binary(2)
binary(10,dim=5)
```

cavendish

Cavendish’s determinations of the mean density of the earth Data

Description

The Cavendish’s determinations of the mean density of the earth data (relative to that of water) are 29 measures performed in the 1798 using a torsion balance devised earlier by Michell. After the sixth of these determinations, Cavendish changed his experimental apparatus by replacing a suspension wire by one that was stiffer. To further complicate matters, Cavendish erred in taking the mean of all 29 determinations by treating the value 4.88 as if it were in fact 5.88.

Usage

```
data(cavendish)
```

Format

cavendish is a vector of 29 observations, the first sixth are made before the apparatus replacement.

Source

Cavendish, H. (1900) Experiments to determine the density of the earth, *Philosophical Transactions of the Royal Society of London for the year 1798 (Part II)* **88**, 469–526, Reprinted in *The law of gravitation* (A.S. Mackenzie, ed.) American, New York.

See Also

Stigler, S.M. (1977) Do robust estimators work with *real* data? *Annals of Statistics*, **5**, 1055–1098, (with discussion).

hald

Hald Data

Description

Montgomery and Peck (1982) illustrated variable selection techniques on the Hald cement data and gave several references to other analysis. The response variable y is the *heat evolved* in a cement mix. The four explanatory variables are ingredients of the mix, i.e., x_1 : *tricalcium aluminate*, x_2 : *tricalcium silicate*, x_3 : *tetracalcium alumino ferrite*, x_4 : *dicalcium silicate*. An important feature of these data is that the variables x_1 and x_3 are highly correlated ($\text{corr}(x_1, x_3) = -0.824$), as well as the variables x_2 and x_4 (with $\text{corr}(x_2, x_4) = -0.975$). Thus we should expect any subset of (x_1, x_2, x_3, x_4) that includes one variable from highly correlated pair to do as any subset that also includes the other member.

Usage

```
data(hald)
```

Format

`hald` is a matrix with 13 observations (rows) and 5 variables (columns), the first column is the dependent variable. `y.hald` and `x.hald` are also availables.

Source

Montgomery, D.C., Peck, E.A. (1982) *Introduction to linear regression analysis*, John Wiley, New York.

mde.vonmises

von Mises Minimum Distance Estimates

Description

Computes the minimum distance estimates for the parameters of a von Mises distribution: the mean direction and the concentration parameter.

Usage

```
mde.vonmises(x, bw, mu = NULL, kappa = NULL, n = 512, from = circular(0), to = circular(2*pi))
## S3 method for class 'mde.vonmises':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>x</code>	a vector. The object is coerced to class <code>circular</code> .
<code>bw</code>	the value of the smoothing parameter.
<code>mu</code>	initial value for the mean direction. Default: maximum likelihood estimate.
<code>kappa</code>	initial value for the concentration parameter. Default: maximum likelihood estimate.
<code>n</code>	number of points used to approximate the density.
<code>from</code>	from which point in the circle the density is approximate.
<code>to</code>	to which point in the circle the density is approximate.
<code>lower</code>	a 2 elements vector passed to <code>optim</code> used to constrained optimization. First element for the mean direction, second element for the concentration.
<code>upper</code>	a 2 elements vector passed to <code>optim</code> used to constrained optimization. First element for the mean direction, second element for the concentration.
<code>method</code>	passed to <code>optim</code> .
<code>lower.kappa</code>	if <code>lower</code> is NULL this parameter is used to constrained optimization for the concentration parameter.
<code>upper.kappa</code>	if <code>upper</code> is NULL this parameter is used to constrained optimization for the concentration parameter.
<code>alpha</code>	see the next argument <code>p</code> . This is a different parameterization, <code>alpha=-1/2</code> provides Hellinger distance, <code>alpha=-1</code> provides Kullback-Leibler distance and <code>alpha=-2</code> provides Neyman's Chi-Square distance.
<code>p</code>	<code>p=2</code> provides Hellinger distance, <code>p=-1</code> provides Kullback-Leibler distance and <code>p=Inf</code> provides Neyman's Chi-Square distance. It is ignored if <code>alpha</code> is not NULL.
<code>control.circular</code>	the attribute of the resulting object (<code>mu</code>)
<code>digits</code>	integer indicating the precision to be used.
<code>...</code>	further parameters in <code>print.mde.vonmises</code> .

Details

The distance from an estimated density (by the non parametric kernel density estimator) and the model is evaluated by simple rectangular approximation. `optim` is used to performs minimization.

Value

Returns a list with the following components:

<code>call</code>	the <code>match.call()</code> .
<code>mu</code>	the estimate of the mean direction.
<code>kappa</code>	the estimate of the concentration parameter.
<code>dist</code>	the distance between the estimated density and the model.
<code>data</code>	the original supplied data converted in radians, clockwise and zero at 0.

x the 'n' coordinates of the points where the density is estimated.
 y the estimated density values.
 k the density at the model.

Author(s)

Claudio Agostinelli

References

C. Agostinelli (2006) Robust Estimation for Circular Data, under revision.

See Also

[circular](#), [mle.vonmises](#) and [wle.vonmises](#).

Examples

```
if (require(circular)) {
  set.seed(1234)
  x <- c(rvonmises(n=200, mu=circular(0), kappa=10), rvonmises(n=20, mu=circular(pi/2), ka
  res <- mde.vonmises(x, bw=500, mu=circular(0), kappa=10)
  res
  plot(circular(0), type='n', xlim=c(-1, 1.75), shrink=1.2)
  lines(circular(res$x), res$y)
  lines(circular(res$x), res$k, col=2)
  legend(1,1.5, legend=c('estimated density', 'MDE'), lty=c(1, 1), col=c(1, 2))
} else {
  cat("Please, install the package 'circular' in order to use this function.\n")
}
```

mde.wrappednormal *Wrapped Normal Minimum Distance Estimates*

Description

Computes the minimum distance estimates for the parameters of a Wrapped Normal distribution: the mean direction and the concentration parameter (and the scale parameter).

Usage

```
mde.wrappednormal(x, bw, mu = NULL, rho = NULL, sd = NULL, alpha = NULL, p = 2, tol
## S3 method for class 'mde.wrappednormal':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	a vector. The object is coerced to class <code>circular</code> .
bw	the value of the smoothing parameter.
mu	initial value for the mean direction. Default: maximum likelihood estimate.
rho	initial value for the concentration parameter. Default: maximum likelihood estimate.
sd	initial value for the standard deviation parameter. This value is used only if <code>rho</code> is <code>NULL</code> . Default: maximum likelihood estimate.
alpha	see the next argument <code>p</code> . This is a different parameterization, <code>alpha=-1/2</code> provides Hellinger distance, <code>alpha=-1</code> provides Kullback-Leibler distance and <code>alpha=-2</code> provides Neyman's Chi-Square distance.
p	<code>p=2</code> provides Hellinger distance, <code>p=-1</code> provides Kullback-Leibler distance and <code>p=Inf</code> provides Neyman's Chi-Square distance. It is ignored if <code>alpha</code> is not <code>NULL</code> .
tol	the absolute accuracy to be used to achieve convergence of the algorithm. This argument is passed to the function which determined the Maximum Likelihood estimates of the parameters. See <code>mle.wrappednormal</code> .
n	number of points used to approximate the density.
from	from which point in the circle the density is approximate.
to	to which point in the circle the density is approximate.
lower	a 2 elements vector passed to <code>optim</code> used to constrained optimization. First element for the mean direction, second element for the concentration.
upper	a 2 elements vector passed to <code>optim</code> used to constrained optimization. First element for the mean direction, second element for the concentration.
method	passed to <code>optim</code> .
lower.rho	if <code>lower</code> is <code>NULL</code> this parameter is used to constrained optimization for the concentration parameter.
upper.rho	if <code>upper</code> is <code>NULL</code> this parameter is used to constrained optimization for the concentration parameter.
min.sd	minimum value for the <code>sd</code> parameter. This argument is passed to the function which determined the Maximum Likelihood estimates of the parameters. See <code>mle.wrappednormal</code> .
K	number of elements used to approximate the density of the wrapped normal.
min.k	minimum number of elements used to approximate the density of the wrapped normal.
control.circular	the attribute of the resulting object (<code>mu</code>)
digits	integer indicating the precision to be used.
...	further parameters in <code>print.mde.wrappednormal</code> .

Details

The distance from an estimated density (by the non parametric kernel density estimator) and the model is evaluated by simple rectangular approximation. `optim` is used to performs minimization.

Value

Returns a list with the following components:

<code>call</code>	the <code>match.call()</code> .
<code>mu</code>	the estimate of the mean direction.
<code>rho</code>	the estimate of the concentration parameter.
<code>sd</code>	the estimate of the standard deviation parameter.
<code>dist</code>	the distance between the estimated density and the model.
<code>data</code>	the original supplied data converted in radians, clockwise and zero at 0.
<code>x</code>	the 'n' coordinates of the points where the density is estimated.
<code>y</code>	the estimated density values.
<code>k</code>	the density at the model.

Author(s)

Claudio Agostinelli

References

C. Agostinelli (2006) Robust Estimation for Circular Data, under revision.

See Also

[circular](#), [mle.wrappednormal](#) and [wle.wrappednormal](#).

Examples

```
if (require(circular)) {
  set.seed(1234)
  x <- c(rwrappednormal(n=200, mu=circular(0), sd=0.6), rwrappednormal(n=20, mu=circular(pi/4), sd=0.6))
  res <- mde.wrappednormal(x, bw=0.08, mu=circular(0), sd=0.6)
  res
  plot(circular(0), type='n', xlim=c(-1, 1.75), shrink=1.2)
  lines(circular(res$x), res$y)
  lines(circular(res$x), res$k, col=2)
  legend(1,1.5, legend=c('estimated density', 'MDE'), lty=c(1, 1), col=c(1, 2))
} else {
  cat("Please, install the package 'circular' in order to use this function.\n")
}
```

mle.aic

Akaike Information Criterion

Description

The Akaike Information Criterion is evaluated for each submodel.

Usage

```
mle.aic(formula, data=list(), model=TRUE, x=FALSE,
        y=FALSE, var.full=0, alpha=2, contrasts = NULL,
        se=FALSE, verbose=FALSE)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>mle.aic</code> is called from.
model, x, y	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response.)
var.full	the value of variance to be used, if 0 the variance estimated from the full model is used.
alpha	the penalized constant.
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
se	logical. if <code>TRUE</code> the returning object contains standard errors for the parameters of every model.
verbose	if <code>TRUE</code> warnings are printed.

Details

Models for `mle.aic` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

Value

`mle.aic` returns an object of class "mle.aic".

The function `summary` is used to obtain and print a summary of the results. The generic accessor functions `coefficients` and `residuals` extract coefficients and residuals returned by `mle.aic`. The object returned by `mle.aic` are:

aic	the AIC for each submodels
coefficients	the parameters estimator, one row vector for each submodel.
scale	an estimation of the error scale, one value for each submodel.
residuals	the residuals from the estimated model, one column vector for each submodel.
call	the match.call().
contrasts	
xlevels	
terms	the model frame.
model	if model=TRUE a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if x=TRUE a matrix with the explanatory variables for the full model.
y	if y=TRUE a vector with the dependent variable.
info	not well working yet, if 0 no error occurred.
se	standard errors of the parameters, one row vector for each submodel. Available only if se is TRUE.

Author(s)

Claudio Agostinelli

References

Akaike, H., (1973) Information theory and an extension of the maximum likelihood principle, in: B.N. Petrov and F. Csáki, eds., *Proc. 2nd International Symposium of Information Theory*, Akadémiai Kiadó, Budapest, 267-281.

Examples

```
library(wle)

data(hald)

cor(hald)

result <- mle.aic(y.hald~x.hald)

summary(result,num.max=10)
```

mle.aic.summaries *Summaries and methods for mle.aic*

Description

All these functions are [methods](#) for class `mle.aic` or `summary.mle.aic`.

Usage

```
## S3 method for class 'mle.aic':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'mle.aic':
print(x, digits = max(3, getOption("digits") - 3),
      num.max=max(1, nrow(x$aic)), ...)

## S3 method for class 'summary.mle.aic':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>object</code>	an object of class <code>mle.aic</code> .
<code>x</code>	an object of class <code>mle.aic</code> or <code>summary.mle.aic</code> .
<code>num.max</code>	the max number of models should be reported.
<code>digits</code>	number of digits to be used for most numbers.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.mle.aic</code>) or further arguments passed to or from other methods (in <code>print.mle.aic</code> and <code>print.summary.mle.aic</code>).

Value

`summary.mle.aic` returns a list:

<code>aic</code>	the first <code>num.max</code> best models with their AIC.
<code>num.max</code>	the number of models reported.
<code>call</code>	

Author(s)

Claudio Agostinelli

See Also

[mle.aic](#) a function for evaluate the Akaike Information Criterion.

mle.cp

*Mallows Cp***Description**

The Mallows Cp is evaluated for each submodel.

Usage

```
mle.cp(formula, data=list(), model=TRUE, x=FALSE,
       y=FALSE, var.full=0, contrasts=NULL, verbose=FALSE)
```

Arguments

<code>formula</code>	a symbolic description of the model to be fit. The details of model specification are given below.
<code>data</code>	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>mle.cp</code> is called from.
<code>model, x, y</code>	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response).
<code>var.full</code>	the value of variance to be used in the denominator of the Mallows Cp, if 0 the variance estimated from the full model is used.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>verbose</code>	if <code>TRUE</code> warnings are printed.

Details

Models for `mle.cp` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

Value

`mle.cp` returns an object of class `"mle.cp"`.

The function `summary` is used to obtain and print a summary of the results, only models below the bisector are reported. The generic accessor functions `coefficients` and `residuals` extract coefficients and residuals returned by `mle.cp`. The object returned by `mle.cp` are:

<code>cp</code>	Mallows Cp for each submodels
<code>coefficients</code>	the parameters estimator, one row vector for eac submodel.
<code>scale</code>	an estimation of the error scale, one value for each submodel.

residuals	the residuals from the estimated model, one column vector for each submodel.
call	the match.call().
contrasts	
xlevels	
terms	the model frame.
model	if model=TRUE a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if x=TRUE a matrix with the explanatory variables for the full model.
y	if y=TRUE a vector with the dependent variable.
info	not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

Mallows, C.L., (1973) Some comments on Cp, *Technometrics*, 15, 661-675.

Examples

```
library(wle)
data(hald)
cor(hald)
result <- mle.cp(y.hald~x.hald)
summary(result)
plot(result)
```

mle.cp.summaries *Summaries and methods for mle.cp*

Description

All these functions are [methods](#) for class mle.cp or summary.mle.cp.

Usage

```
## S3 method for class 'mle.cp':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'mle.cp':
print(x, digits = max(3, getOption("digits") - 3),
      num.max=max(1, nrow(x$cp)), ...)

## S3 method for class 'summary.mle.cp':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	an object of class <code>mle.cp</code> .
x	an object of class <code>mle.cp</code> or <code>summary.mle.cp</code> .
digits	number of digits to be used for most numbers.
num.max	the max number of models should be reported.
verbose	if TRUE warnings are printed.
...	additional arguments affecting the summary produced (in <code>summary.mle.cp</code>) or further arguments passed to or from other methods (in <code>print.mle.cp</code> and <code>print.summary.mle.cp</code>).

Value

`summary.mle.cp` returns a list:

cp	the first <code>num.max</code> best models with their Mallows Cp.
num.max	the number of models reported.
call	

Author(s)

Claudio Agostinelli

See Also

[mle.cp](#) a function for evaluate the Mallows Cp.

mle.cv

Cross Validation Selection Method

Description

The Cross Validation selection method is evaluated for each submodel.

Usage

```
mle.cv(formula, data=list(), model=TRUE, x=FALSE,
        y=FALSE, monte.carlo=500, split,
        contrasts=NULL, verbose=FALSE)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>mle.cv</code> is called from.
model, x, y	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response.)
monte.carlo	the number of Monte Carlo replication we use to estimate the average prediction error.
split	the size of the construction sample. When the suggested value is outside the possible range, the split size is let equal to $\max(\text{round}(\text{size}^{(3/4)}), \text{nvar} + 2)$.
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
verbose	if <code>TRUE</code> warnings are printed.

Details

Models for `mle.cv` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

Value

`mle.cv` returns an object of class "mle.cv".

The function `summary` is used to obtain and print a summary of the results.

The object returned by `mle.cv` are:

cv	the estimated prediction error for each submodels
----	---

call	the match.call().
contrasts	
xlevels	
terms	the model frame.
model	if model=TRUE a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if x=TRUE a matrix with the explanatory variables for the full model.
y	if y=TRUE a vector with the dependent variable.
info	not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

Shao, J., (1993) Linear model selection by Cross-Validation. *Journal American Statistical Association*, 88, 486-494.

Examples

```
library(wle)

data(hald)

cor(hald)

result <- mle.cv(y.hald~x.hald)

summary(result)
```

mle.cv.summaries *Summaries and methods for mle.cv*

Description

All these functions are [methods](#) for class `mle.cv` or `summary.mle.cv`.

Usage

```
## S3 method for class 'mle.cv':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'mle.cv':
print(x, digits = max(3, getOption("digits") - 3), num.max=max(1, nrow(x$cv)), ...)

## S3 method for class 'summary.mle.cv':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>object</code>	an object of class <code>mle.cv</code> .
<code>x</code>	an object of class <code>mle.cv</code> or <code>summary.mle.cv</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>num.max</code>	the max number of models should be reported.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.mle.cv</code>) or further arguments passed to or from other methods (in <code>print.mle.cv</code> and <code>print.summary.mle.cv</code>).

Value

`summary.mle.cv` returns a list:

<code>cv</code>	the first <code>num.max</code> best models with their estimated prediction error using CV.
<code>num.max</code>	the number of models reported.
<code>call</code>	

Author(s)

Claudio Agostinelli

See Also

[mle.cv](#) a function for evaluate the Cross-Validation selection criterion for linear models.

<code>mle.stepwise</code>	<i>Stepwise, Backward and Forward selection methods</i>
---------------------------	---

Description

This function performs Stepwise, Forward and Backward model selection.

Usage

```
mle.stepwise(formula, data=list(), model=TRUE, x=FALSE,
              y=FALSE, type="Forward", f.in=4.0, f.out=4.0,
              contransts=NULL, verbose=FALSE)
```

Arguments

<code>formula</code>	a symbolic description of the model to be fit. The details of model specification are given below.
<code>data</code>	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>mle.stepwise</code> is called from.
<code>model, x, y</code>	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response.)
<code>type</code>	<code>type="Stepwise"</code> : the stepwise methods is used, <code>type="Forward"</code> : the forward methods is used, <code>type="Backward"</code> : the backward method is used.
<code>f.in</code>	the in value
<code>f.out</code>	the out value
<code>contranst</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>verbose</code>	if <code>TRUE</code> warnings are printed.

Details

Models for `mle.stepwise` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

Value

`mle.stepwise` returns an object of class `"mle.stepwise"`.

The function `summary` is used to obtain and print a summary of the results.

The object returned by `mle.stepwise` are:

<code>step</code>	the selected models
<code>type</code>	the type o model selection procedure was used.
<code>f.in</code>	the value of <code>f.in</code> used.
<code>f.out</code>	the value of <code>f.out</code> used.
<code>call</code>	the <code>match.call()</code> .
<code>contrasts</code>	
<code>xlevels</code>	
<code>terms</code>	the model frame.
<code>model</code>	if <code>model=TRUE</code> a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
<code>x</code>	if <code>x=TRUE</code> a matrix with the explanatory variables for the full model.
<code>y</code>	if <code>y=TRUE</code> a vector with the dependent variable.
<code>info</code>	not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

Beale, E.M.L., Kendall, M.G., Mann, D.W., (1967) The discarding of variables in multivariate analysis, *Biometrika*, 54, 357-366.

Efroymsen, (1960) Multiple regression analysis, in *Mathematical Methods for Digital Computers*, eds. A. Ralston and H.S. Wilf, 191-203, Wiley, New York.

Garside, M.J., (1965) The best sub-set in multiple regression analysis, *Applied Statistics*, 14, 196-200.

Goldberger, A.S. and Jochems, D.B., (1961) Note on stepwise least squares, *Journal of the American Statistical Association*, 56, 105-110.

Goldberger, A.S., (1961) Stepwise least squares: Residual analysis and specification error, *Journal of the American Statistical Association*, 56, 998-1000.

Examples

```
library(wle)

data(hald)

cor(hald)

result <- mle.stepwise(y.hald~x.hald)

summary(result)
```

```
mle.stepwise.summaries
      Accessing summaries for mle.stepwise
```

Description

All these functions are [methods](#) for class `mle.stepwise` or `summary.mle.stepwise`.

Usage

```
## S3 method for class 'mle.stepwise':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'mle.stepwise':
print(x, digits = max(3, getOption("digits") - 3), num.max=max(1,nrow(x$step)), ...)

## S3 method for class 'summary.mle.stepwise':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	an object of class <code>mle.stepwise</code> .
x	an object of class <code>mle.stepwise</code> or <code>summary.mle.stepwise</code> .
digits	number of digits to be used for most numbers.
num.max	the number of the last iterations reported.
verbose	if TRUE warnings are printed.
...	additional arguments affecting the summary produced (in <code>summary.mle.stepwise</code>) or further arguments passed to or from other methods (in <code>print.mle.stepwise</code> and <code>print.summary.mle.stepwise</code>).

Value

The function `summary.mle.stepwise` returns the last `num.max` iterations, call plus:

step	the model for each iteration reported.
num.max	the number of iterations reported.
type	the type of selection procedure used.
f.in	the in value
f.out	the out value

Author(s)

Claudio Agostinelli

plot.mle.cp

Plot the Mallows Cp

Description

Plot the Mallows Cp.

Usage

```
## S3 method for class 'mle.cp':
plot(x, base.line=0, num.max=20,
      plot.it=TRUE, log.scale=FALSE,
      xlab="Number of Predictors", ylab=NULL,
      verbose=FALSE, ...)
```

Arguments

<code>x</code>	an object of class <code>mle.cp</code> .
<code>base.line</code>	the intercept of the line to split the submodels in acceptable (good) and not-acceptable (bad), (the slope is always one).
<code>num.max</code>	maximum number of submodels plotted.
<code>plot.it</code>	if TRUE the graph is plotted.
<code>log.scale</code>	if TRUE the y-axis as log10 scale.
<code>xlab</code>	a title for the x axis.
<code>ylab</code>	a title for the y axis.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	graphical parameters can be given as arguments.

Value

<code>num.good</code>	number of submodels below the <code>base.line</code>
<code>num.bad</code>	number of submodels above the <code>base.line</code>
<code>cp.good</code>	list of the submodels below the <code>base.line</code> with their Cp.
<code>cp.bad</code>	list of the submodels above the <code>base.line</code> with their Cp.

Author(s)

Claudio Agostinelli

See Also

[mle.cp](#) a function to calculate the Mallows Cp.

Examples

```
library(wle)

data(hald)

result <- mle.cp(y.hald~x.hald)

plot(result,num.max=7)
```

plot.wle.cp

Plot the Weighted Mallows Cp

Description

Plot the weighted Mallows Cp based on weighted likelihood.

Usage

```
## S3 method for class 'wle.cp':
plot(x, base.line=0, num.max=20,
      plot.it=TRUE, log.scale=FALSE,
      xlab="Number of Predictors", ylab=NULL,
      verbose=FALSE, ...)
```

Arguments

x	an object of class <code>wle.cp</code> .
base.line	the intercept of the line to split the submodels in acceptable (good) and not-acceptable (bad), (the slope is always one).
num.max	maximum number of submodels plotted.
plot.it	if TRUE the graph is plotted.
log.scale	if TRUE the y-axis as log10 scale.
xlab	a title for the x axis.
ylab	a title for the y axis.
verbose	if TRUE warnings are printed.
...	graphical parameters can be given as arguments.

Value

num.good	number of submodels below the <code>base.line</code>
num.bad	number of submodels above the <code>base.line</code>
wcp.good	list of the submodels below the <code>base.line</code> with their WCp.
wcp.bad	list of the submodels above the <code>base.line</code> with their WCp.

Author(s)

Claudio Agostinelli

References

Agostinelli, C., (1999) Robust model selection in regression via weighted likelihood methodology, *Working Paper n. 1999.4*, Department of Statistics, University of Padova.

Agostinelli, C., (1999) Robust model selection in regression via weighted likelihood methodology, submitted to *Statistics & Probability Letters*, revised december 1999.

Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Agostinelli, C., (1998) Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Società Italiana di Statistica*, Sorrento 1998.

See Also

`wle.cp` a function to calculate the Weighted Mallows Cp, `wle.lm` a function for estimating linear models with normal distribution error and normal kernel.

Examples

```
library(wle)
x.data <- c(runif(60,20,80),runif(5,73,78))
e.data <- rnorm(65,0,0.6)
y.data <- 8*log(x.data+1)+e.data
y.data[61:65] <- y.data[61:65]-4
z.data <- c(rep(0,60),rep(1,5))
plot(x.data, y.data, xlab="X", ylab="Y")
xx.data <- cbind(x.data, x.data^2, x.data^3, log(x.data+1))
result <- wle.cp(y.data~xx.data)
plot(result,num.max=15)
```

plot.wle.lm

Plots for the Linear Model

Description

The `plot.wle.lm` function plots a separate graph windows for each root. In each windows four plots are printed: residuals vs fitted, normal qq plot of the residuals, weighted residuals vs weighted fitted, normal qq plot of the weighted residuals. A summary plot is also printed: in the diagonal, the value of the weights vs position of the observations for each root; in the upper diagonal residuals vs residuals of two different roots; in the lower diagonal weights vs weights of two different roots. The roots and the graphs can be chosen by the arguments `roots`, `which.main` and `which`.

Usage

```
## S3 method for class 'wle.lm':
plot(x, roots, which=1:4, which.main, level.weight=0.5,
      ask = dev.interactive(), col=c(2, 1, 3), id.n=3, labels.id, cex.id = 0.75, verbo
```

Arguments

<code>x</code>	an object of class <code>wle.lm</code> .
<code>roots</code>	a vector specify for which roots the plots are required.
<code>which</code>	if a subset of the plots for each root is required, specify a subset of the numbers $0:4$, 0 means no plots.
<code>which.main</code>	if a subset of the plots for the main graphic is required, specify a subset of the numbers $0:roots^2$, 0 means no plots. The plots are specified by columns.
<code>level.weight</code>	value of the weight under which an observations is marked with different color.
<code>ask</code>	logical; if TRUE, the user is <i>asked</i> before each plot, see <code>par(ask=.)</code> .
<code>col</code>	a vector of 3 elements, to specify colors for the plots.
<code>id.n</code>	number of points to be labelled in some plots, starting with the ones with less weight.
<code>labels.id</code>	vector of labels, from which the labels for less weighted points will be chosen. If missing uses observation numbers.
<code>cex.id</code>	magnification of point labels.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	graphical parameters can be given as arguments.

Author(s)

Claudio Agostinelli

See Also

[wle.lm](#) a function for estimating linear models with normal distribution error and normal kernel.

Examples

```
library(wle)

data(artificial)

result <- wle.lm(y~x1+x2+x3, data=artificial, boot=40, group=6, num.sol=2)

result

plot(result) # all plots, default behavior

plot(result, roots=1) # only first root, one plot for window

par(mfcol=c(2,2))
plot(result, roots=1) # only first root, as usual

plot(result, roots=2, which=1, which.main=0)
# only second root, only residual vs fitted values plot

plot(result, which=1)
```

```
# main plot + residual vs fitted values plot for each root

par(mfcol=c(3,2))
plot(result, which=1)
# main plot + residual vs fitted values plot for each root all in the same window
```

rocky	<i>Rockwell hardness, 100 coils produced in sequence at a Chicago Steel Mill Data</i>
-------	---

Description

This data set is the Rockwell Hardness (measured on Rockwell "B" scale) of a sample of 100 steel coins produced in sequence in a Chicago steel mill.

Usage

```
data(rocky)
```

Format

rocky is a time serie with 100 elements.

Source

K.W. Hipel and A.I. McLeod (1994), Time series modelling of water resources and environmental systems, Elsevier, Amsterdam.

selection	<i>Selection's Data</i>
-----------	-------------------------

Description

This data set consists of 60 observations for two variables (ydata and xdata). The appropriate regression model for the first fiftyfourth observations should be like $y = x^2 + e$ the last sixth comes from a different model.

Usage

```
data(selection)
```

Format

xdata is a vector which contains x, x^2 , while ydata contains the dependent variable's observations.

Source

Agostinelli, C. (1999) Robust model selection in regression via weighted likelihood methodology, submitted to *Statistics & Probability Letters*, revised december 1999.

wle.aic

*Weighted Akaike Information Criterion***Description**

The Weighted Akaike Information Criterion.

Usage

```
wle.aic(formula, data=list(), model=TRUE, x=FALSE,
        y=FALSE, boot=30, group, var.full=0, num.sol=1,
        raf="HD", smooth=0.031, tol=10^(-6),
        equal=10^(-3), max.iter=500, min.weight=0.5,
        method="full", alpha=2, contrasts=NULL, verbose=FALSE)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>wle.aic</code> is called from.
model, x, y	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response.)
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), \text{var})$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
var.full	the value of variance to be used in the denominator of the WAIC, if 0 the variance estimated from the full model is used.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations.

<code>min.weight</code>	see details.
<code>method</code>	see details.
<code>alpha</code>	penalty value.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>verbose</code>	if TRUE warnings are printed.

Details

Models for `wle.aic` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

`min.weight`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and type. The presence of multiple roots in the full model can create some problem in the set of weights we should use. Actually, the selection of the root is done by the minimum scale error provided. Since this choice is not always the one would choose, we introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

`method`: this parameter, when set to "reduced", allows to use weights based on the reduced model. This is strongly discourage since the robust and asymptotic property of this kind of weighted AIC are not as good as the one based on `method="full"`.

Value

`wle.aic` returns an object of class "wle.aic".

The function `summary` is used to obtain and print a summary of the results. The generic accessor functions `coefficients` and `residuals` extract coefficients and residuals returned by `wle.aic`. The object returned by `wle.aic` are:

<code>waic</code>	Weighted Akaike Information Criterion for each submodels
<code>coefficients</code>	the parameters estimator, one row vector for each root found and each submodel.
<code>scale</code>	an estimation of the error scale, one value for each root found and each submodel.
<code>residuals</code>	the unweighted residuals from the estimated model, one column vector for each root found and each submodel.
<code>tot.weights</code>	the sum of the weights divide by the number of observations, one value for each root found and each submodel.
<code>weights</code>	the weights associated to each observation, one column vector for each root found and each submodel.
<code>freq</code>	the number of starting points converging to the roots.
<code>call</code>	the <code>match.call()</code> .

contrasts	
xlevels	
terms	the model frame.
model	if model=TRUE a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if x=TRUE a matrix with the explanatory variables for the full model.
y	if y=TRUE a vector with the dependent variable.
info	not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

- Agostinelli, C., (1999) Robust model selection in regression via weighted likelihood methodology, *Working Paper n. 1999.4*, Department of Statistics, University of Padova.
- Agostinelli, C., (2002) Robust model selection in regression via weighted likelihood methodology, *Statistics & Probability Letters*, 56, 289-300.
- Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.
- Agostinelli, C., Markatou, M., (1998) A one-step robust estimator for regression based on the weighted likelihood reweighting scheme, *Statistics & Probability Letters*, Vol. 37, n. 4, 341-350.
- Agostinelli, C., (1998) Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Societa' Italiana di Statistica*, Sorrento 1998.

Examples

```
library(wle)

x.data <- c(runif(60,20,80),runif(5,73,78))
e.data <- rnorm(65,0,0.6)
y.data <- 8*log(x.data+1)+e.data
y.data[61:65] <- y.data[61:65]-4
z.data <- c(rep(0,60),rep(1,5))

plot(x.data,y.data,xlab="X",ylab="Y")

xx.data <- cbind(x.data,x.data^2,x.data^3,log(x.data+1))
colnames(xx.data) <- c("X","X^2","X^3","log(X+1)")

result <- wle.aic(y.data~xx.data,boot=10,group=10,num.sol=2)

summary(result)

result <- wle.aic(y.data~xx.data+z.data,boot=10,group=10,num.sol=2)

summary(result)
```

wle.aic.ar

Weighted Akaike Information Criterion for AR models

Description

The function evaluate the Weighted Akaike Information Criterion for AutoRegressive Models. This is a robust model selection method to choose the order of an AutoRegressive model.

Usage

```
wle.aic.ar(x, order = c(1, 0), seasonal = list(order = c(0, 0), period = NA), group
```

Arguments

<code>x</code>	a univariate time series.
<code>order</code>	maximum order to investigate. A specification of the non-seasonal part of the ARI model: the two components (p, d) are the AR order and the degree of differencing.
<code>seasonal</code>	a specification of the seasonal part of the ARI model, plus the period (which defaults to <code>frequency(x)</code>).
<code>group</code>	the dimension of the bootstrap subsamples.
<code>group.start</code>	the dimension of the bootstrap subsamples used in the starting process if <code>wle.init=TRUE</code> .
<code>group.step</code>	the dimension of the bootstrap subsamples used in a step, it must be less than <code>group</code> .
<code>xreg</code>	optionally, a vector or matrix of external regressors, which must have the same number of rows as <code>x</code> .
<code>include.mean</code>	Should the ARI model include a mean term? The default is <code>TRUE</code> for undifferenced series, <code>FALSE</code> for differenced ones (where a mean would not affect the fit nor predictions).
<code>na.action</code>	function to be applied to remove missing values.
<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>tol.step</code>	the absolute accuracy to be used to achieve convergence in a step.
<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>equal.step</code>	the absolute value for which two roots are considered the same in a step. (This parameter must be greater than <code>tol.step</code>).
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>var.full</code>	An estimate of the residual variance for the full model.
<code>smooth</code>	the value of the smoothing parameter.

<code>smooth.ao</code>	the value of the smoothing parameter used in the outliers classification, default equal to <code>smooth</code> .
<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>boot.start</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots in the starting process.
<code>boot.step</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots in a step.
<code>num.sol</code>	maximum number of roots to be searched.
<code>x.init</code>	initial values, a vector with the same length of the AR order, or a number, default is 0.
<code>x.seasonal.init</code>	initial values, a vector with the same length of the SAR order, or a number, default is 0.
<code>max.iter.out</code>	maximum number of iterations in the outer loop.
<code>max.iter.in</code>	maximum number of iterations in the inner loop.
<code>max.iter.start</code>	maximum number of iterations in the starting process.
<code>max.iter.step</code>	maximum number of iterations in a step.
<code>verbose</code>	if TRUE warnings are printed.
<code>w.level</code>	the threshold used to decide if an observation could be an additive outlier.
<code>min.weights</code>	see details.
<code>population.size</code>	see details.
<code>population.choose</code>	see details.
<code>elements.random</code>	see details.
<code>wle.start</code>	if TRUE a weighted likelihood estimation is used to have a starting value.
<code>init.values</code>	a vector with initial values for the AR and seasonal AR coefficients and the innovations variance.
<code>num.max</code>	maximum number of observations can be considered as possible additive outliers.
<code>num.sol.step</code>	maximum number of roots to be searched in a step.
<code>min.weights.aic</code>	see details.
<code>approx.w</code>	logical: if TRUE an approximation is used to evaluate the weights in the outlier identification procedure.
<code>ask</code>	logical. If TRUE, in the case of multiple roots in the full model, the users is asked for selecting the root.
<code>alpha</code>	penalty value.
<code>method</code>	if "WLE" the parameters are estimated using weighted likelihood estimating equations in the reduced models, otherwise if "WLS" a weighted least squares approach is used with weights based on the full model.

Details

`min.weights`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and type. We introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

`min.weights.aic` is used as `min.weights` but in the full model. The algorithm used to classify the observations as additive outliers is made by a genetic algorithm. The `population.size`, `population.choose` and `elements.random` are parameters related to this algorithm.

The function `wle.ar.wls` is used to estimate the parameter of an autoregressive model by weighted least squares where the weights are those from the weighted likelihood estimating equation of the full model (the model with the highest order).

Value

A list of class `wle.aic.ar` with the following components:

<code>full.model</code>	the results for the full model, that is an object of class <code>wle.arima</code> see wle.ar help for further details.
<code>waic</code>	Weighted Akaike Information Criterion for each submodels.
<code>call</code>	<code>match.call</code> result.

Author(s)

Claudio Agostinelli

References

Agostinelli C, (2004) Robust Akaike Information Criterion for ARMA models, *Rendiconti per gli Studi Economici Quantitativi*, 1-14, isbn: 88-88037-10-1.

Agostinelli C., (2003) Robust time series estimation via weighted likelihood, in: *Development in Robust Statistics. International Conference on Robust Statistics 2001*, Eds. Dutter, R. and Filzmoser, P. and Rousseeuw, P. and Gather, U., Physica Verlag.

See Also

[wle.ar](#)

Examples

```
data(rocky)

res <- wle.aic.ar(x=rocky, order=c(6,0), group=50, group.start=30, method="WLS")
res
plot(res$full.model$weights)
```

```
wle.aic.ar.summaries
```

Summaries and methods for wle.aic.ar

Description

All these functions are [methods](#) for class `wle.aic.ar` or `summary.wle.aic.ar`.

Usage

```
## S3 method for class 'wle.aic.ar':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'wle.aic.ar':
print(x, digits = max(3, getOption("digits") - 3),
      num.max=max(1, nrow(x$waic)), ...)

## S3 method for class 'summary.wle.aic.ar':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>object</code>	an object of class <code>wle.aic.ar</code> .
<code>x</code>	an object of class <code>wle.aic</code> or <code>summary.wle.aic.ar</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>num.max</code>	the max number of models should be reported.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.wle.aic.ar</code>) or further arguments passed to or from other methods (in <code>print.wle.aic.ar</code> and <code>print.summary.wle.aic.ar</code>).

Value

`summary.wle.aic.ar` returns a list:

<code>waic</code>	the first <code>num.max</code> best models with their Weighted Akaike Information Criterion.
<code>num.max</code>	the number of models reported.
<code>call</code>	

Author(s)

Claudio Agostinelli

See Also

[wle.aic.ar](#) a function for evaluate the Weighted Akaike Information Criterion for autoregressive models.

wle.aic.summaries *Summaries and methods for wle.aic*

Description

All these functions are [methods](#) for class `wle.aic` or `summary.wle.aic`.

Usage

```
## S3 method for class 'wle.aic':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'wle.aic':
print(x, digits = max(3, getOption("digits") - 3),
      num.max=max(1, nrow(x$waic)), ...)

## S3 method for class 'summary.wle.aic':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>object</code>	an object of class <code>wle.aic</code> .
<code>x</code>	an object of class <code>wle.aic</code> or <code>summary.wle.aic</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>num.max</code>	the max number of models should be reported.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.wle.aic</code>) or further arguments passed to or from other methods (in <code>print.wle.aic</code> and <code>print.summary.wle.aic</code>).

Value

`summary.wle.aic` returns a list:

<code>waic</code>	the first <code>num.max</code> best models with their Weighted Akaike Information Criterion.
<code>num.max</code>	the number of models reported.
<code>call</code>	

Author(s)

Claudio Agostinelli

See Also

`wle.aic` a function for evaluate the Weighted Akaike Information Criterion in the linear models.

wle.ar

Fit Autoregressive Models to Time Series - Preliminary Version

Description

This is a preliminary version of functions for the estimation of the autoregressive parameters via Weighted Likelihood Estimating Equations and a classification algorithm. The main function is `wle.ar`, the remain functions are for internal use and they should not call by the users. They are not documented here.

Usage

```
wle.ar(x, order=c(1, 0), seasonal=list(order = c(0, 0),
  period = NA), group, group.start, group.step=group.start,
  xreg=NULL, include.mean=TRUE, na.action=na.fail,
  tol=10-6, tol.step=tol, equal=10-3, equal.step=equal,
  raf="HD", smooth=0.0031, smooth.ao=smooth, boot=10,
  boot.start=10, boot.step=boot.start, num.sol=1, x.init=0,
  x.seasonal.init=0, max.iter.out=20, max.iter.in=50,
  max.iter.start=200, max.iter.step=500, verbose=FALSE,
  w.level=0.4, min.weights=0.5, population.size=10,
  population.choose=5, elements.random=2, wle.start=FALSE,
  init.values=NULL, num.max=NULL, num.sol.step=2, approx.w=TRUE)
```

Arguments

<code>x</code>	a univariate time series.
<code>order</code>	a specification of the non-seasonal part of the ARI model: the two components (p, d) are the AR order and the degree of differencing.
<code>seasonal</code>	a specification of the seasonal part of the ARI model, plus the period (which defaults to <code>frequency(x)</code>).
<code>group</code>	the dimension of the bootstrap subsamples.
<code>group.start</code>	the dimension of the bootstrap subsamples used in the starting process if <code>wle.init=TRUE</code> .
<code>group.step</code>	the dimension of the bootstrap subsamples used in a step, it must be less than <code>group</code> .
<code>xreg</code>	optionally, a vector or matrix of external regressors, which must have the same number of rows as <code>x</code> .
<code>include.mean</code>	Should the ARI model include a mean term? The default is <code>TRUE</code> for undifferenced series, <code>FALSE</code> for differenced ones (where a mean would not affect the fit nor predictions).
<code>na.action</code>	function to be applied to remove missing values.

<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>tol.step</code>	the absolute accuracy to be used to achieve convergence in a step.
<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>equal.step</code>	the absolute value for which two roots are considered the same in a step. (This parameter must be greater than <code>tol.step</code>).
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>smooth</code>	the value of the smoothing parameter.
<code>smooth.ao</code>	the value of the smoothing parameter used in the outliers classificaton, default equal to <code>smooth</code> .
<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>boot.start</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots in the starting process.
<code>boot.step</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots in a step.
<code>num.sol</code>	maximum number of roots to be searched.
<code>num.sol.step</code>	maximum number of roots to be searched in a step.
<code>x.init</code>	initial values, a vector with the same length of the AR order, or a number, default is 0.
<code>x.seasonal.init</code>	initial values, a vector with the same length of the SAR order, or a number, default is 0.
<code>max.iter.out</code>	maximum number of iterations in the outer loop.
<code>max.iter.in</code>	maximum number of iterations in the inner loop.
<code>max.iter.start</code>	maximum number of iterations in the starting process.
<code>max.iter.step</code>	maximum number of iterations in a step.
<code>w.level</code>	the threshold used to decide if an observation could be an additive outlier.
<code>population.size</code>	see details.
<code>population.choose</code>	see details.
<code>elements.random</code>	see details.
<code>num.max</code>	maximum number of observations can be considered as possible additive outliers.
<code>wle.start</code>	if TRUE a weighted likelihood estimation is used to have a starting value.
<code>init.values</code>	a vector with initial values for the AR and seasonal AR coefficients and the innovations variance.

<code>verbose</code>	if TRUE warnings are printed.
<code>min.weights</code>	see details.
<code>approx.w</code>	logical: if TRUE an approximation is used to evaluate the weights in the outlier identification procedure.

Details

`min.weight`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and type. We introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

The algorithm used to classify the observations as additive outliers is made by a genetic algorithm. The `population.size`, `population.choose` and `elements.random` are parameters related to this algorithm.

Value

<code>coef</code>	a vector of AR and regression coefficients.
<code>sigma2.coef</code>	the estimated variance matrix of the coefficients <code>coef</code> .
<code>sigma2</code>	the WLE of the innovations variance.
<code>arma</code>	a compact form of the specification, as a vector giving the number of AR, MA=0, seasonal AR and seasonal MA=0 coefficients, plus the period and the number of non-seasonal and seasonal differences.
<code>resid</code>	the residuals.
<code>resid.with.ao</code>	the residuals with the additive outliers effects.
<code>resid.without.ao</code>	the residuals without the additive outliers effects.
<code>x.ao</code>	the time series without the additive outliers effects.
<code>call</code>	the matched call.
<code>series</code>	the name of the series <code>x</code> .
<code>weights</code>	the weights.
<code>weights.with.ao</code>	the weights with the additive outliers effects.
<code>weights.without.ao</code>	the weights without the additive outliers effects
<code>tot.sol</code>	the number of solutions found.
<code>not.conv</code>	the number of starting points that does not converge after the <code>max.iter.out</code> iteration are reached.
<code>ao.position</code>	the position of the additive outliers.

Author(s)

Claudio Agostinelli

References

Agostinelli C., (2001) Robust time series estimation via weighted likelihood: some preliminary results, *Working Paper n. 2001.3* Department of Statistics, University of Padova.

Agostinelli C., (2003) Robust time series estimation via weighted likelihood, in: *Development in Robust Statistics. International Conference on Robust Statistics 2001*, Eds. Dutter, R. and Filzmoser, P. and Rousseeuw, P. and Gather, U., Physica Verlag.

Examples

```
data(lh)
wle.ar(x=lh, order=c(3,0), group=30)
```

wle.binomial

Robust Estimation in the Binomial Model

Description

wle.binomial is used to robust estimate the proportion parameters via Weighted Likelihood.

Usage

```
wle.binomial(x, size, boot=30, group, num.sol=1, raf="HD",
             tol=10-6, equal=10-3, max.iter=500,
             verbose=FALSE)
```

Arguments

x	a vector contain the number of success in each size trials.
size	number of trials.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{length}(x)/4), 2)$.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than tol).
max.iter	maximum number of iterations.
verbose	if TRUE warnings are printed.

Value

wle.binomial returns an object of class "wle.binomial".

Only print method is implemented for this class.

The object returned by wle.binomial are:

p	the estimator of the proportion parameter, one value for each root found.
tot.weights	the sum of the weights divide by the number of observations, one value for each root found.
weights	the weights associated to each observation, one column vector for each root found.
f.density	the non-parametric density estimation.
m.density	the smoothed model.
delta	the Pearson residuals.
call	the match.call().
tot.sol	the number of solutions found.
not.conv	the number of starting points that does not converge after the max.iter iteration are reached.

Author(s)

Claudio Agostinelli

References

Markatou, M., Basu, A., and Lindsay, B.G., (1997) Weighted likelihood estimating equations: The discrete case with applications to logistic regression, *Journal of Statistical Planning and Inference*, 57, 215-232.

Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Examples

```
library(wle)

set.seed(1234)

x <- rbinom(20,p=0.2,size=10)
wle.binomial(x,size=10)

x <- c(rbinom(20,p=0.2,size=10),rbinom(10,p=0.9,size=10))
wle.binomial(x,size=10)
```

wle.cp

*Weighted Mallows Cp***Description**

The Weighted Mallows Cp is evaluated for each submodel.

Usage

```
wle.cp(formula, data=list(), model=TRUE, x=FALSE,
       y=FALSE, boot=30, group, var.full=0, num.sol=1,
       raf="HD", smooth=0.031, tol=10^(-6),
       equal=10^(-3), max.iter=500, min.weight=0.5,
       method="full", alpha=2, contrasts=NULL, verbose=FALSE)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>wle.cp</code> is called from.
model, x, y	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response.)
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), \text{var})$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
var.full	the value of variance to be used in the denominator of the WCP, if 0 the variance estimated from the full model is used.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations.
min.weight	see details.
method	see details.
alpha	penalty value.
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
verbose	if TRUE warnings are printed.

Details

Models for `wle.cp` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

`min.weight`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and type. The presence of multiple roots in the full model can create some problem in the set of weights we should use. Actually, the selection of the root is done by the minimum scale error provided. Since this choice is not always the one would choose, we introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

`method`: this parameter, when set to "reduced", allows to use weights based on the reduced model. This is strongly discourage since the robust and asymptotic property of this kind of weighted Cp are not as good as the one based on `method="full"`.

Value

`wle.cp` returns an object of `class "wle.cp"`.

The function `summary` is used to obtain and print a summary of the results. The generic accessor functions `coefficients` and `residuals` extract coefficients and residuals returned by `wle.cp`. The object returned by `wle.cp` are:

<code>wcp</code>	Weighted Mallows Cp for each submodels
<code>coefficients</code>	the parameters estimator, one row vector for each root found and each submodel.
<code>scale</code>	an estimation of the error scale, one value for each root found and each submodel.
<code>residuals</code>	the unweighted residuals from the estimated model, one column vector for each root found and each submodel.
<code>tot.weights</code>	the sum of the weights divide by the number of observations, one value for each root found and each submodel.
<code>weights</code>	the weights associated to each observation, one column vector for each root found and each submodel.
<code>freq</code>	the number of starting points converging to the roots.
<code>call</code>	the <code>match.call()</code> .
<code>contrasts</code>	
<code>xlevels</code>	
<code>terms</code>	the model frame.
<code>model</code>	if <code>model=TRUE</code> a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
<code>x</code>	if <code>x=TRUE</code> a matrix with the explanatory variables for the full model.

y if y=TRUE a vector with the dependent variable.
 info not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

- Agostinelli, C., (1999). Robust model selection in regression via weighted likelihood methodology, *Working Paper n. 1999.4*, Department of Statistics, University of Padova.
- Agostinelli, C., (2002). Robust model selection in regression via weighted likelihood methodology, *Statistics & Probability Letters*, 56, 289-300.
- Agostinelli, C., (1998). Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.
- Agostinelli, C., Markatou, M., (1998). A one-step robust estimator for regression based on the weighted likelihood reweighting scheme, *Statistics & Probability Letters*, Vol. 37, n. 4, 341-350.
- Agostinelli, C., (1998). Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Società Italiana di Statistica*, Sorrento 1998.

See Also

[wle.smooth](#) an algorithm to choose the smoothing parameter for normal distribution and normal kernel, [wle.lm](#) a function for estimating linear models with normal distribution error and normal kernel.

Examples

```
library(wle)

x.data <- c(runif(60,20,80),runif(5,73,78))
e.data <- rnorm(65,0,0.6)
y.data <- 8*log(x.data+1)+e.data
y.data[61:65] <- y.data[61:65]-4
z.data <- c(rep(0,60),rep(1,5))

plot(x.data,y.data,xlab="X",ylab="Y")

xx.data <- cbind(x.data,x.data^2,x.data^3,log(x.data+1))
colnames(xx.data) <- c("X","X^2","X^3","log(X+1)")

result <- wle.cp(y.data~xx.data,boot=10,group=10,num.sol=2)

summary(result)

plot(result,num.max=15)

result <- wle.cp(y.data~xx.data+z.data,boot=10,group=10,num.sol=2)

summary(result)
```

```
plot(result,num.max=15)
```

```
wle.cp.summaries Summaries and methods for wle.cp
```

Description

All these functions are [methods](#) for class `wle.cp` or `summary.wle.cp`.

Usage

```
## S3 method for class 'wle.cp':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'wle.cp':
print(x, digits = max(3, getOption("digits") - 3),
      num.max=max(1, nrow(x$wcp)), ...)

## S3 method for class 'summary.wle.cp':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>object</code>	an object of class <code>wle.cp</code> .
<code>x</code>	an object of class <code>wle.cp</code> or <code>summary.wle.cp</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>num.max</code>	the max number of models should be reported.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.wle.cp</code>) or further arguments passed to or from other methods (in <code>print.wle.cp</code> and <code>print.summary.wle.cp</code>).

Value

`summary.wle.cp` returns a list:

<code>wcp</code>	the first <code>num.max</code> best models with their Weighted Mallows Cp.
<code>num.max</code>	the number of models reported.
<code>call</code>	

Author(s)

Claudio Agostinelli

See Also

[wle.cp](#) a function for evaluate the Weighted Mallows Cp in the linear models.

Description

The Weighted Cross-Validation methods is used to choose the best linear model.

Usage

```
wle.cv(formula, data=list(), model=TRUE, x=FALSE,
       y=FALSE, monte.carlo=500, split, boot=30,
       group, num.sol=1, raf="HD", smooth=0.031,
       tol=10-6, equal=10-3, max.iter=500,
       min.weight=0.5, contrasts=NULL, verbose=FALSE)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>wle.cv</code> is called from.
model, x, y	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response.)
monte.carlo	the number of Monte Carlo replication we use to estimate the average prediction error.
split	the size of the construction sample. When the suggested value is outside the possible range, the split size is let equal to $\max(\text{round}(\text{size}^{3/4}), \text{var} + 2)$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), \text{var})$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations.
min.weight	see details.
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
verbose	if <code>TRUE</code> warnings are printed.

Details

Models for `wle.cv` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

`min.weight`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and type. The presence of multiple roots in the full model can create some problem in the set of weights we should use. Actually, the selection of the root is done by the minimum scale error provided. Since this choice is not always the one would choose, we introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

Value

`wle.cv` returns an object of class `"wle.cv"`.

The function `summary` is used to obtain and print a summary of the results. The generic accessor functions `coefficients` and `residuals` extract coefficients and residuals returned by `wle.cv`. The object returned by `wle.cv` are:

<code>wcv</code>	Weighted Cross-Validation for each submodels
<code>coefficients</code>	the parameters estimator, one row vector for each root found in the full model.
<code>scale</code>	an estimation of the error scale, one value for each root found in the full model.
<code>residuals</code>	the unweighted residuals from the estimated model, one column vector for each root found in the full model.
<code>tot.weights</code>	the sum of the weights divide by the number of observations, one value for each root found in the full model.
<code>weights</code>	the weights associated to each observation, one column vector for each root found in the full model.
<code>freq</code>	the number of starting points converging to the roots.
<code>index</code>	position of the root used for the weights.
<code>call</code>	the <code>match.call()</code> .
<code>contrasts</code>	
<code>xlevels</code>	
<code>terms</code>	the model frame.
<code>model</code>	if <code>model=TRUE</code> a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
<code>x</code>	if <code>x=TRUE</code> a matrix with the explanatory variables for the full model.
<code>y</code>	if <code>y=TRUE</code> a vector with the dependent variable.
<code>info</code>	not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

Agostinelli, C., (1999). Robust model selection by Cross-Validation via weighted likelihood methodology, *Working Paper n. 1999.37*, Department of Statistics, University of Padova.

Agostinelli, C., (1998). Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Agostinelli, C., Markatou, M., (1998). A one-step robust estimator for regression based on the weighted likelihood reweighting scheme, *Statistics & Probability Letters*, Vol. 37, n. 4, 341-350.

Agostinelli, C., (1998). Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Società Italiana di Statistica*, Sorrento 1998.

See Also

[wle.smooth](#) an algorithm to choose the smoothing parameter for normal distribution and normal kernel, [wle.lm](#) a function for estimating linear models with normal distribution error and normal kernel.

Examples

```
library(wle)

set.seed(1234)

x.data <- c(runif(60,20,80),runif(5,73,78))
e.data <- rnorm(65,0,0.6)
y.data <- 8*log(x.data+1)+e.data
y.data[61:65] <- y.data[61:65]-4
z.data <- c(rep(0,60),rep(1,5))

plot(x.data,y.data,xlab="X",ylab="Y")

xx.data <- cbind(x.data,x.data^2,x.data^3,log(x.data+1))
colnames(xx.data) <- c("X","X^2","X^3","log(X+1)")

result <- wle.cv(y.data~xx.data,boot=20,num.sol=2)

summary(result)

result <- wle.cv(y.data~xx.data+z.data,boot=20,num.sol=2,
monte.carlo=1000,split=50)

summary(result)
```

wle.cv.summaries *Summaries and methods for wle.cv*

Description

All these functions are [methods](#) for class `wle.cv` or `summary.wle.cv`.

Usage

```
## S3 method for class 'wle.cv':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'wle.cv':
print(x, digits = max(3, getOption("digits") - 3),
      num.max=max(1, nrow(x$wcv)), ...)

## S3 method for class 'summary.wle.cv':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>object</code>	an object of class <code>wle.cv</code> .
<code>x</code>	an object of class <code>wle.cv</code> or <code>summary.wle.cv</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>num.max</code>	the max number of models should be reported.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.wle.cv</code>) or further arguments passed to or from other methods (in <code>print.wle.cv</code> and <code>print.summary.wle.cv</code>).

Value

`summary.wle.cv` returns a list:

<code>wcv</code>	the first <code>num.max</code> best models with their estimated prediction error using WCV.
<code>num.max</code>	the number of models reported.
<code>call</code>	

Author(s)

Claudio Agostinelli

See Also

[wle.cv](#) a function for evaluate the Weighted Cross Validation criterion in the linear models.

wle.fracdiff

*Fit Fractional Models to Time Series - Preliminary Version***Description**

This is a preliminary version of functions for the estimation of the fractional parameter via Weighted Likelihood Estimating Equations and a classification algorithm. The main function is `wle.fracdiff`, the remain functions are for internal use and they should not call by the users. They are not documented here.

Usage

```
wle.fracdiff(x, lower, upper, M, group, na.action=na.fail,
tol=10-6, equal=10-3, raf="HD", smooth=0.0031, smooth.ao=smooth,
boot=10, num.sol=1, x.init=rep(0,M), use.uniroot=FALSE, max.iter.out=20,
max.iter.in=100, max.iter.step=5000, max.iter.start=max.iter.step,
verbose=FALSE, w.level=0.4, min.weights=0.5, init.values=NULL,
num.max=length(x), include.mean=FALSE, ao.list=NULL, elitist=5,
size.generation=5, size.population=10, type.selection="roulette",
prob.crossover=0.8, prob.mutation=0.02, type.scale="none", scale.c=2)
```

Arguments

<code>x</code>	a univariate time series.
<code>lower</code>	the lower end point of the interval to be searched.
<code>upper</code>	the upper end point of the interval to be searched.
<code>M</code>	the order of the finite memory process used to estimate the d parameter.
<code>group</code>	the dimension of the bootstrap subsamples.
<code>na.action</code>	function to be applied to remove missing values.
<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>smooth</code>	the value of the smoothing parameter.
<code>smooth.ao</code>	the value of the smoothing parameter used in the outliers classification, default equal to <code>smooth</code> .
<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>num.sol</code>	maximum number of roots to be searched.
<code>x.init</code>	initial values, a vector with the same length of the M parameter, or a number, default is 0.

<code>use.uniroot</code>	default: FALSE, if TRUE in each step the weighted likelihood estimating equations is solved, otherwise, a maximization is performed on a weighted log-likelihood function with fixed weights. The estimators obtain with the two methods is the same.
<code>max.iter.out</code>	maximum number of iterations in the outer loop.
<code>max.iter.in</code>	maximum number of iterations in the inner loop.
<code>max.iter.step</code>	maximum number of iterations in a step.
<code>max.iter.start</code>	maximum number of iterations in the starting process.
<code>verbose</code>	if TRUE warnings are printed.
<code>w.level</code>	the threshold used to decide if an observation could be an additive outlier.
<code>init.values</code>	a vector with initial values for the d and the innovations variance.
<code>num.max</code>	maximum number of observations can be considered as possible additive outliers.
<code>include.mean</code>	Should the model include a mean term? The default is TRUE.
<code>ao.list</code>	possible list of pattern of additive outliers.
<code>min.weights</code>	see details.
<code>size.population</code>	see details.
<code>size.generation</code>	see details.
<code>prob.crossover</code>	see details.
<code>prob.mutation</code>	see details.
<code>type.scale</code>	see details.
<code>type.selection</code>	see details.
<code>elitist</code>	see details.
<code>scale.c</code>	see details.

Details

`min.weight`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and type. We introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

The algorithm used to classify the observations as additive outliers is a simple genetic algorithm as described in Goldberg (1989). The `size.population`, `size.generation`, `type.selection`, `prob.crossover`, `prob.mutation`, `type.scale`, `type.selection`, `elitist` and `scale.c` are parameters related to this algorithm.

Value

<code>d</code>	the WLE of the fractional parameter.
<code>sigma2</code>	the WLE of the innovations variance.
<code>x.mean</code>	the WLE of the mean.
<code>resid</code>	the residuals.
<code>resid.without.ao</code>	the residuals with the additive outliers effects.
<code>resid.with.ao</code>	the residuals without the additive outliers effects.
<code>x.ao</code>	the time series without the additive outliers effects.
<code>call</code>	the matched call.
<code>weights</code>	the weights.
<code>weights.with.ao</code>	the weights with the additive outliers effects.
<code>weights.without.ao</code>	the weights without the additive outliers effects
<code>tot.sol</code>	the number of solutions found.
<code>not.conv</code>	the number of starting points that does not converge after the <code>max.iter.out</code> iteration are reached.
<code>ao.position</code>	the position of the additive outliers.

Author(s)

Claudio Agostinelli

References

Agostinelli C., Bisaglia L., (2002) Robust estimation of ARFIMA processes, manuscript.
 Goldberg, David E., (1989) Genetic Algorithms in Search, Optimization and Machine Learning.
 Addison-Wesley Pub. Co. ISBN: 0201157675

Examples

```
require(stats)
if (require(fracdiff)) {
  res <- fracdiff(Nile)
  res$d
}
set.seed(1234)
resw <- wle.fracdiff(Nile, M=100, include.mean=TRUE, lower=0.01, upper=0.96, group=20)
resw$d
resw$sigma2
resw$x.mean

x <- Nile
x[50] <- x[50]+4*sd(x)
```

```

if (require(fracdiff)) {
  res <- fracdiff(x)
  res$d
}

set.seed(1234)
resw <- wle.fracdiff(x, M=100, include.mean=TRUE, lower=0.01, upper=0.96, group=40)
resw$d
resw$sigma2
resw$x.mean
resw$ao.position

```

wle.gamma

*Robust Estimation in the Gamma model***Description**

wle.gamma is used to robust estimate the shape and the scale parameters via Weighted Likelihood, when the majority of the data are from a gamma distribution.

Usage

```

wle.gamma(x, boot=30, group, num.sol=1, raf="HD", smooth=0.008,
          tol=10-6, equal=10-3, max.iter=500,
          shape.int=c(0.01, 100), use.smooth=TRUE, tol.int,
          verbose=FALSE, maxiter=1000)

```

Arguments

x	a vector contain the observations.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), \text{var})$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations for the main function.

<code>shape.int</code>	a 2 dimension vector for the interval search of the shape parameter.
<code>use.smooth</code>	if FALSE the unsmoothed model is used. This is usefull when the integration routine does not work well.
<code>tol.int</code>	the absolute accuracy to be used in the integration routine. The default value is $tol * 10^{-4}$.
<code>verbose</code>	if TRUE warnings are printed.
<code>maxiter</code>	maximum number of iterations. This value is passed to <code>uniroot</code> function.

Details

The gamma is parametrized as follows ($\alpha = scale, \omega = shape$):

$$f(x) = 1/(\alpha^\omega \Gamma(\omega)) x^{\omega-1} e^{-x/\alpha}$$

for $x > 0, \alpha > 0$ and $\omega > 0$.

The function use `uniroot` to solve the estimating equation for *shape*, errors from `uniroot` are handled by `try`. If errors occurs then the function returns NA.

You can use `shape.int` to avoid them. It also use a fortran routine (`dqagp`) to calculate the smoothed model, i.e., evaluate the integral. Sometime the accuracy is not satisfactory, you can use `use.smooth=FALSE` to have an approximate estimation using the model instead of the smoothed model.

The Folded Normal distribution is use as kernel. The bandwidth is $smooth * shape / scale^2$.

Value

`wle.gamma` returns an object of `class "wle.gamma"`.

Only print method is implemented for this class.

The object returned by `wle.gamma` are:

<code>shape</code>	the estimator of the shape parameter, one value for each root found.
<code>scale</code>	the estimator of the scale parameter, one value for each root found.
<code>rate</code>	the estimator of the rate parameter (1/scale), one value for each root found.
<code>tot.weights</code>	the sum of the weights divide by the number of observations, one value for each root found.
<code>weights</code>	the weights associated to each observation, one column vector for each root found.
<code>f.density</code>	the non-parametric density estimation.
<code>m.density</code>	the smoothed model.
<code>delta</code>	the Pearson residuals.
<code>call</code>	the <code>match.call()</code> .
<code>tot.sol</code>	the number of solutions found.
<code>not.conv</code>	the number of starting points that does not converge after the <code>max.iter</code> iteration are reached.

Author(s)

Claudio Agostinelli

References

Markatou, M., Basu, A. and Lindsay, B.G., (1998). Weighted likelihood estimating equations with a bootstrap root search, *Journal of the American Statistical Association*, 93, 740-750.

Agostinelli, C., (1998). Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Examples

```
library(wle)

x <- rgamma(n=100, shape=2, scale=2)

wle.gamma(x)

x <- c(rgamma(n=30, shape=2, scale=2), rgamma(n=100, shape=20, scale=20))

wle.gamma(x, boot=10, group=10, num.sol=2) # depending on the sample, one or two roots.
```

wle.lm

Fitting Linear Models using Weighted Likelihood

Description

wle.lm is used to fit linear models via Weighted Likelihood, when the errors are iid from a normal distribution with null mean and unknown variance. The carriers are considered fixed. Note that this estimator is robust against the presence of bad leverage points too.

Usage

```
wle.lm(formula, data=list(), model=TRUE, x=FALSE,
        y=FALSE, boot=30, group, num.sol=1, raf="HD",
        smooth=0.031, tol=10^(-6), equal=10^(-3),
        max.iter=500, contrasts=NULL, verbose=FALSE)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given below.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which wle.lm is called from.
model, x, y	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response.)

<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>group</code>	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), \text{var})$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
<code>num.sol</code>	maximum number of roots to be searched.
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>smooth</code>	the value of the smoothing parameter.
<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>max.iter</code>	maximum number of iterations.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>verbose</code>	if TRUE warnings are printed.

Details

Models for `wle.lm` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

Value

`wle.lm` returns an object of class `"wle.lm"`.

The function `summary` is used to obtain and print a summary of the results. The generic accessor functions `coefficients`, `residuals` and `fitted.values` extract coefficients, residuals and fitted values returned by `wle.lm`.

The object returned by `wle.lm` are:

<code>coefficients</code>	the parameters estimator, one row vector for each root found.
<code>standard.error</code>	an estimation of the standard error of the parameters estimator, one row vector for each root found.
<code>scale</code>	an estimation of the error scale, one value for each root found.
<code>residuals</code>	the unweighted residuals from the estimated model, one column vector for each root found.
<code>fitted.values</code>	the fitted values from the estimated model, one column vector for each root found.

tot.weights	the sum of the weights divide by the number of observations, one value for each root found.
weights	the weights associated to each observation, one column vector for each root found.
f.density	the non-parametric density estimation.
m.density	the smoothed model.
delta	the Pearson residuals.
freq	the number of starting points converging to the roots.
tot.sol	the number of solutions found.
not.conv	the number of starting points that does not converge after the <code>max.iter</code> iteration are reached.
call	the <code>match.call()</code> .
contrasts	
xlevels	
terms	the model frame.
model	if <code>model=TRUE</code> a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if <code>x=TRUE</code> a matrix with the explanatory variables for the full model.
y	if <code>y=TRUE</code> a vector with the dependent variable.
info	not well working yet, if 0 no error occurred.

Author(s)

Claudio Agostinelli

References

Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Agostinelli, C., Markatou, M., (1998) A one-step robust estimator for regression based on the weighted likelihood reweighting scheme, *Statistics & Probability Letters*, Vol. 37, n. 4, 341-350.

Agostinelli, C., (1998) Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Società Italiana di Statistica*, Sorrento 1998.

See Also

[wle.smooth](#) an algorithm to choose the smoothing parameter for normal distribution and normal kernel.

Examples

```

library(wle)
# You can find this data set in:
# Hawkins, D.M., Bradu, D., and Kass, G.V. (1984).
# Location of several outliers in multiple regression data using
# elemental sets. Technometrics, 26, 197-208.
#
data(artificial)

result <- wle.lm(y.artificial~x.artificial,boot=40,num.sol=3)

summary(result)

plot(result)

```

wle.lm.summaries *Accessing Linear Model Fits for wle.lm*

Description

All these functions are [methods](#) for class `wle.lm` or `summary.wle.lm`.

Usage

```

## S3 method for class 'wle.lm':
coef(object, ...)
## S3 method for class 'wle.lm':
formula(x, ...)
## S3 method for class 'wle.lm':
fitted(object, ...)
## S3 method for class 'wle.lm':
model.frame(formula, data, na.action, ...)
## S3 method for class 'wle.lm':
summary(object, root="ALL", ...)
## S3 method for class 'wle.lm.root':
summary(object, root=1, ...)

## S3 method for class 'wle.lm':
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'summary.wle.lm':
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars= getOption("show.signif.stars"), ...)

## S3 method for class 'summary.wle.lm.root':
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars= getOption("show.signif.stars"), ...)

```

Arguments

<code>object</code>	an object of class <code>wle.lm</code> .
<code>x</code>	an object of class <code>wle.lm</code> or <code>summary.wle.lm</code> .
<code>formula</code>	a model formula
<code>data</code>	<code>data.frame</code> , <code>list</code> , <code>environment</code> or object coercible to <code>data.frame</code> containing the variables in <code>formula</code> .
<code>na.action</code>	how NAs are treated. The default is <code>first</code> , any <code>na.action</code> attribute of <code>data</code> , second a <code>na.action</code> setting of <code>options</code> , and third <code>na.fail</code> if that is unset. The “factory-fresh” default is <code>na.omit</code> .
<code>root</code>	the root to be printed, in <code>summary.wle.lm</code> it could be "ALL", all the roots are printed, or a vector of integers.
<code>digits</code>	number of digits to be used for most numbers.
<code>signif.stars</code>	logical; if TRUE, P-values are additionally encoded visually as “significance stars” in order to help scanning of long coefficient tables. It defaults to the <code>show.signif.stars</code> slot of <code>options</code> .
<code>...</code>	additional arguments.

Details

`print.summary.wle.lm` and `print.summary.wle.lm.root` tries formatting for each root the coefficients, standard errors, etc. and additionally gives “significance stars” if `signif.stars` is TRUE.

The generic accessor functions `coefficients`, `fitted.values`, `residuals` and `weights` can be used to extract various useful features of the value returned by `wle.lm`.

Value

The function `summary.wle.lm` (the `summary.wle.lm.root` do the same for just one selected root) computes and returns, for each selected root, a list of summary statistics of the fitted linear model given in `object`, using the components (list elements) `"call"` and `"terms"` from its argument, plus

<code>residuals</code>	the <i>weighted</i> residuals, the usual residuals rescaled by the square root of the weights given by <code>wle.lm</code> .
<code>coefficients</code>	a $p \times 4$ matrix with columns for the estimated coefficient, its standard error, weighted-t-statistic and corresponding (two-sided) p-value.
<code>sigma</code>	the square root of the estimated variance of the random error.
<code>df</code>	degrees of freedom, a 3-vector $(p, \sum weights - p, p^*)$.
<code>fstatistic</code>	a 3-vector with the value of the weighted-F-statistic with its numerator and denominator degrees of freedom.
<code>r.squared</code>	R^2 , the “fraction of variance explained by the model”.
<code>adj.r.squared</code>	the above R^2 statistic “ <i>adjusted</i> ”, penalizing for higher p .
<code>root</code>	the label of the root reported.

Author(s)

Claudio Agostinelli

See Also

`wle.lm` a function for estimating linear models with normal distribution error and normal kernel, `plot.wle.lm` for plot method.

Examples

```
library(wle)
# You can find this data set in:
# Hawkins, D.M., Bradu, D., and Kass, G.V. (1984).
# Location of several outliers in multiple regression data using
# elemental sets. Technometrics, 26, 197-208.
#
data(artificial)

result <- wle.lm(y.artificial~x.artificial,boot=40,group=6,num.sol=3)

#summary only for the first root
summary(result,root=1)
#summary for all the roots
summary(result,root="ALL")
```

wle.normal

Robust Estimation in the Normal Model

Description

`wle.normal` is used to robust estimate the location and the scale parameters via Weighted Likelihood, when the sample is iid from a normal distribution with unknown mean and variance.

Usage

```
wle.normal(x, boot=30, group, num.sol=1, raf="HD",
           smooth=0.003, tol=10-6, equal=10-3,
           max.iter=500, verbose=FALSE)
```

Arguments

<code>x</code>	a vector contain the observations.
<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>group</code>	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), 2)$ where <i>size</i> is the number of observations.
<code>num.sol</code>	maximum number of roots to be searched.

raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations.
verbose	if TRUE warnings are printed.

Value

`wle.normal` returns an object of class `"wle.normal"`.

Only `print` method is implemented for this class.

The object returned by `wle.normal` are:

location	the estimator of the location parameter, one value for each root found.
scale	the estimator of the scale parameter, one value for each root found.
residuals	the residuals associated to each observation, one column vector for each root found.
tot.weights	the sum of the weights divide by the number of observations, one value for each root found.
weights	the weights associated to each observation, one column vector for each root found.
f.density	the non-parametric density estimation.
m.density	the smoothed model.
delta	the Pearson residuals.
freq	the number of starting points converging to the roots.
call	the <code>match.call()</code> .
tot.sol	the number of solutions found.
not.conv	the number of starting points that does not converge after the <code>max.iter</code> iteration are reached.

Author(s)

Claudio Agostinelli

References

Markatou, M., Basu, A. and Lindsay, B.G., (1998) Weighted likelihood estimating equations with a bootstrap root search, *Journal of the American Statistical Association*, 93, 740-750.

Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Arguments

x	a vector contain the observations.
m	numbers of components.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), 2)$ where <i>size</i> is the number of observations.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations.
all.comp	try to find all the components.
min.size	see details
min.weights	see details
boot.start	the number of starting points for the starting process.
group.start	the dimension of the bootstrap subsamples in the starting process. The default value is $\max(\text{round}(\text{group}/4), 2)$.
tol.start	the absolute accuracy to be used to achieve convergence of the algorithm in the starting process.
equal.start	the absolute value for which two roots are considered the same in the starting process. (This parameter must be greater than <code>tol.start</code>).
smooth.start	the value of the smoothing parameter in the starting process.
max.iter.start	maximum number of iterations in the starting process.
max.iter.boot	maximum number of iterations of the starting process.
verbose	if TRUE warnings are printed.

Details

this function use an iterative procedure to evaluate starting points. First, using `wle.normal` we try to find the biggest components, then we discard each observation with weight greater than `min.weights`. The `wle.normal` is run on the remain observations if the ratio between the number of observations and the original sample size is greater than `min.size`. The convergence of the algorithm is determined by the difference between two iterations. This stopping rule could have some problems, as soon as possible it will replace with the one proposed in Markatou (2000) pag. 485 (5).

Value

wle.normal.mixture returns an object of class "wle.normal.mixture".

Only print method is implemented for this class.

The objects returned by wle.normal.mixture are:

location	the estimator of the location parameters, one vector for each root found.
scale	the estimator of the scale parameters, one vector for each root found.
pi	the estimator of the proportion parameters, one vector for each root found.
tot.weights	the sum of the weights, divide by the number of observations, one value for each root found.
weights	the weights associated to each observation, one column vector for each root found.
f.density	the non-parametric density estimation.
m.density	the smoothed model.
delta	the Pearson residuals.
freq	the number of starting points converging to the roots.
tot.sol	the number of solutions found.
not.conv	the number of starting points that does not converge after the max.iter iteration are reached.
call	the match.call().

Author(s)

Claudio Agostinelli

References

Markatou, M., (2000) Mixture models, robustness and the weighted likelihood methodology, *Biometrics*, 56, 483-486.

Markatou, M., (2001) A closer look at the weighted likelihood in the context of mixtures, *Probability and Statistical Models with Applications*, Charalambides, C.A., Koutras, M.V. and Balakrishnan, N. (eds.), Chapman and Hall/CRC, 447-467.

Examples

```
library(wle)
set.seed(1234)
x <- c(rnorm(150, 0, 1), rnorm(50, 15, 2))
wle.normal.mixture(x, m=2, group=50, group.start=2, boot=5, num.sol=3)
wle.normal(x, group=2, boot=10, num.sol=3)
```

wle.normal.multi *Robust Estimation in the Normal Multivariate Model*

Description

wle.normal.multi is used to robust estimate the location and the covariance matrix via Weighted Likelihood, when the sample is iid from a normal multivariate distribution with unknown means and variance matrix.

Usage

```
wle.normal.multi(x, boot=30, group, num.sol=1,
                 raf="HD", smooth, tol=10^(-6),
                 equal=10^(-3), max.iter=500,
                 verbose=FALSE)
```

Arguments

x	a matrix contain the observations.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), (\text{var} * (\text{var} + 1)/2 + \text{var}))$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than tol).
max.iter	maximum number of iterations.
verbose	if TRUE warnings are printed.

Value

wle.normal.multi returns an object of class "wle.normal.multi".

Only print method is implemented for this class.

The object returned by wle.normal.multi are:

location	the estimator of the location parameters, one vector for each root found.
----------	---


```
barplot(result$weights,col=2,xlab="Observations",
        ylab="Weights",ylim=c(0,1),
        names.arg=seq(1:length(result$weights)))
```

```
wle.normal.multi.summaries
```

Summaries and methods for wle.normal.multi

Description

Until now, only `print methods` is available for class `wle.normal.multi`. `print.wle.normal.multi` print nicely the output.

Usage

```
## S3 method for class 'wle.normal.multi':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>x</code>	an object of class <code>wle.normal.multi</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>...</code>	further arguments passed to or from other methods.

Author(s)

Claudio Agostinelli

See Also

`wle.normal.multi` a function for estimating normal multivariate location and scale models.

```
wle.normal.summaries
```

Summaries and methods for wle.normal

Description

Until now, only `print methods` is available for class `wle.normal`. `print.wle.normal` print nicely the output.

Usage

```
## S3 method for class 'wle.normal':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x` an object of class `wle.normal`.
`digits` number of digits to be used for most numbers.
`...` further arguments passed to or from other methods.

Author(s)

Claudio Agostinelli

See Also

[wle.normal](#) a function for estimating normal location and scale models.

wle.onestep

A One-Step Weighted Likelihood Estimator for Linear model

Description

This function evaluate the One-step weighted likelihood estimator for the regression and scale parameters.

Usage

```
wle.onestep(formula, data=list(), model=TRUE, x=FALSE,
             y=FALSE, ini.param, ini.scale, raf="HD",
             smooth=0.031, num.step=1,
             contrasts=NULL, verbose=FALSE)
```

Arguments

`formula` a symbolic description of the model to be fit. The details of model specification are given below.

`data` an optional data frame containing the variables in the model. By default the variables are taken from the environment which `wle.stepwise` is called from.

`model, x, y` logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response.)

`ini.param` starting values for the coefficients.

`ini.scale` starting values for the scale parameters.

`raf` type of Residual adjustment function to be use:
`raf="HD"`: Hellinger Distance RAF,
`raf="NED"`: Negative Exponential Disparity RAF,
`raf="SCHI2"`: Symmetric Chi-Squared Disparity RAF.

`smooth` the value of the smoothing parameter.

`num.step` number of the steps.

`contrasts` an optional list. See the `contrasts.arg` of `model.matrix.default`.

`verbose` if `TRUE` warnings are printed.

Value

wle.onestep returns an object of class "wle.onestep".

Only print method is implemented for this class.

The object returned by wle.onestep are:

coefficients	the parameters estimator.
standard.error	an estimation of the standard error of the parameters estimator.
scale	an estimation of the error scale.
residuals	the unweighted residuals from the estimated model.
fitted.values	the fitted values from the estimated model.
tot.weights	the sum of the weights divide by the number of observations.
weights	the weights associated to each observation.
f.density	the non-parametric density estimation.
m.density	the smoothed model.
delta	the Pearson residuals.
call	the match.call().
contrasts	
xlevels	
terms	the model frame.
model	if model=TRUE a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if x=TRUE a matrix with the explanatory variables for the full model.
y	if y=TRUE a vector with the dependent variable.

Author(s)

Claudio Agostinelli

References

- Agostinelli, C., (1997) A one-step robust estimator based on the weighted likelihood methodology, *Working Paper n. 1997.16*, Department of Statistics, University of Padova.
- Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.
- Agostinelli, C., Markatou, M., (1998) A one-step robust estimator for regression based on the weighted likelihood reweighting scheme, *Statistics & Probability Letters*, Vol. 37, n. 4, 341-350.
- Agostinelli, C., (1998) Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Società Italiana di Statistica*, Sorrento 1998.

See Also

[wle.smooth](#) an algorithm to choose the smoothing parameter for normal distribution and normal kernel, [wle.lm](#) a function for estimating linear models with normal distribution error and normal kernel.

Examples

```
#library(wle)
#library(lqs)

#data(artificial)

#result.lts <- lqs(y.artificial~x.artificial,
#                 method = "lts")

#result.wle <- wle.onestep(y.artificial~x.artificial,
#                          ini.param=result.lts$coefficients,
#                          ini.scale=result.lts$scale[1])

#result.wle
```

wle.onestep.summaries

Summaries and methods for wle.onestep

Description

Until now, only `print.methods` is available for class `wle.onestep`. `print.wle.onestep` print nicely the output.

Usage

```
## S3 method for class 'wle.onestep':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

<code>x</code>	an object of class <code>wle.onestep</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>...</code>	further arguments passed to or from other methods.

Author(s)

Claudio Agostinelli

See Also

[wle.onestep](#) a function for one-step estimation in linear models.

wle.poisson *Robust Estimation in the Poisson Model*

Description

wle.poisson is used to robust estimate the lambda parameters in the poisson model via Weighted Likelihood.

Usage

```
wle.poisson(x, boot=30, group, num.sol=1, raf="HD",
            tol=10-6, equal=10-3,
            max.iter=500, verbose=FALSE)
```

Arguments

x	a vector contain the number of success.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{length}(x)/4), 2)$.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than tol).
max.iter	maximum number of iterations.
verbose	if TRUE warnings are printed.

Value

wle.poisson returns an object of class "wle.poisson".

Only print method is implemented for this class.

The object returned by wle.poisson are:

lambda	the estimator of the lambda parameter, one value for each root found.
tot.weights	the sum of the weights divide by the number of observations, one value for each root found.
weights	the weights associated to each observation, one column vector for each root found.
f.density	the non-parametric density estimation.

m.density	the smoothed model.
delta	the Pearson residuals.
call	the match.call().
tot.sol	the number of solutions found.
not.conv	the number of starting points that does not converge after the <code>max.iter</code> iteration are reached.

Author(s)

Claudio Agostinelli

References

Markatou, M., Basu, A., and Lindsay, B.G., (1997) Weighted likelihood estimating equations: The discrete case with applications to logistic regression, *Journal of Statistical Planning and Inference*, 57, 215-232.

Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.

Examples

```
library(wle)

set.seed(1234)

x <- rpois(40, 5)
wle.poisson(x)

x <- c(rpois(40, 5), rpois(10, 20))
wle.poisson(x)
```

wle.smooth

Bandwidth selection for the normal kernel and normal model.

Description

The bandwidth of the kernel is choose for normal model and normal kernel in such a way a contaminated point `costant` times away from the mean of the distribution in scale units and mass level has a weight no bigger than `weight`.

Usage

```
wle.smooth(weight=0.31, costant=3, level=0.2,
           dimension=1, raf="HD", interval=c(0.00001, 0.5),
           tol=10^-6, max.iter=1000)
```

Arguments

<code>weight</code>	weights associated to an observation that is <code>constant</code> scale units away from the mean of the distribution.
<code>constant</code>	times the contaminated point mass is away from the mean of the distribution in scale units.
<code>level</code>	mass of the contaminated point.
<code>dimension</code>	dimension of the normal distribution.
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>interval</code>	interval from which to search the root.
<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>max.iter</code>	maximum number of iterations.

Details

The `wle.smooth` use `uniroot` function to solve the non linear equation. No handling error is provided yet. For the Symmetric Chi-Squared Disparity RAF you should use `weight=0.2` and `interval=c(0.1, 1)` to have a solution.

Value

`wle.smooth` returns an object of `class` "wle.smooth".

Only print method is implemented for this class.

The object returned by `wle.smooth` is a list with four components: `root` and `f.root` give the location of the root and the value of the function evaluated at that point. `iter` and `estim.prec` give the number of iterations used and an approximate estimated precision for root.

`root` is the value of the bandwidth.

Author(s)

Claudio Agostinelli

References

- Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D. thesis*, Department of Statistics, University of Padova.
- Markatou, M., Basu, A. and Lindsay, B.G. (1998) Weighted likelihood estimating equations with a bootstrap root search. *Journal of the American Statistical Association*, 93, 740-750.
- Agostinelli, C., and Markatou, M., (2001) Test of hypotheses based on the Weighted Likelihood Methodology, *Statistica Sinica*, vol. 11, n. 2, 499-514.

See Also

`uniroot`, `uniroot`: one dimensional root finding.

Examples

```
library(wle)

wle.smooth()
```

`wle.stepwise` *Weighted Stepwise, Backward and Forward selection methods*

Description

This function performs Weighted Stepwise, Forward and Backward model selection.

Usage

```
wle.stepwise(formula, data=list(), model=TRUE, x=FALSE,
             y=FALSE, boot=30, group, num.sol=1, raf="HD",
             smooth=0.031, tol=10-6, equal=10-3,
             max.iter=500, min.weight=0.5, type="Forward",
             f.in=4.0, f.out=4.0, method="WLE",
             contrasts=NULL, verbose=FALSE)
```

Arguments

<code>formula</code>	a symbolic description of the model to be fit. The details of model specification are given below.
<code>data</code>	an optional data frame containing the variables in the model. By default the variables are taken from the environment which <code>wle.stepwise</code> is called from.
<code>model, x, y</code>	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response.)
<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>group</code>	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), \text{var})$ where <i>size</i> is the number of observations and <i>var</i> is the number of variables.
<code>num.sol</code>	maximum number of roots to be searched.
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>smooth</code>	the value of the smoothing parameter.
<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.

<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>max.iter</code>	maximum number of iterations.
<code>min.weight</code>	see details.
<code>type</code>	<code>type="Stepwise"</code> : the weighted stepwise methods is used, <code>type="Forward"</code> : the weighted forward methods is used, <code>type="Backward"</code> : the weighted backward method is used.
<code>f.in</code>	the in value
<code>f.out</code>	the out value
<code>method</code>	<code>method="WLS"</code> : the submodel parameters are estimated by weighted least square with weights from the weighted likelihood estimator on the full model. <code>method="WLE"</code> : the submodel parameters are estimated by weighted likelihood estimators.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>verbose</code>	if TRUE warnings are printed.

Details

Models for `wle.stepwise` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first+second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first+second+first:second`.

`min.weight`: the weighted likelihood equation could have more than one solution. These roots appear for particular situation depending on contamination level and `type`. The presence of multiple roots in the full model can create some problem in the set of weights we should use. Actually, the selection of the root is done by the minimum scale error provided. Since this choice is not always the one would choose, we introduce the `min.weight` parameter in order to choose only between roots that do not down weight everything. This is not still the optimal solution, and perhaps, in the new release, this part will be change.

Value

`wle.stepwise` returns an object of class `"wle.stepwise"`.

The function `summary` is used to obtain and print a summary of the results. The generic accessor functions `coefficients` and `residuals` extract coefficients and residuals returned by `wle.stepwise`.

The object returned by `wle.stepwise` are:

<code>wstep</code>	the iterations with the model selected.
<code>coefficients</code>	the parameters estimator, one row vector for each root found in the full model.
<code>scale</code>	an estimation of the error scale, one value for each root found in the full model.

residuals	the unweighted residuals from the estimated model, one column vector for each root found in the full model.
tot.weights	the sum of the weights divide by the number of observations, one value for each root found in the full model.
weights	the weights associated to each observation, one column vector for each root found in the full model.
freq	the number of starting points converging to the roots.
index	position of the root used for the weights.
call	the match.call().
contrasts	
xlevels	
terms	the model frame.
model	if model=TRUE a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
x	if x=TRUE a matrix with the explanatory variables for the full model.
y	if y=TRUE a vector with the dependent variable.
info	not well working yet, if 0 no error occurred.
type	"Stepwise": the weighted stepwise methods is used, "Forward": the weighted forward methods is used, "Backward": the weighted backward method is used.
f.in	the in value.
f.out	the out value.
method	if "WLS" the submodel parameters are estimated by weighted least square with weights from the weighted likelihood estimator on the full model else if "WLE" the submodel parameters are estimated by weighted likelihood estimators.

Author(s)

Claudio Agostinelli

References

- Agostinelli, C., (2000) Robust stepwise regression, Working Paper n. 2000.10 del Dipartimento di Scienze Statistiche, Università di Padova, Padova.
- Agostinelli, C., (2002) Robust stepwise regression, *Journal of Applied Statistics* 29, 6, 825-840.
- Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova.
- Agostinelli, C., (1998) Verosimiglianza pesata nel modello di regressione lineare, *XXXIX Riunione scientifica della Società Italiana di Statistica*, Sorrento 1998.

See Also

[wle.smooth](#) an algorithm to choose the smoothing parameter for normal distribution and normal kernel, [wle.lm](#) a function for estimating linear models with normal distribution error and normal kernel.

Examples

```

library(wle)

# You can find this dataset in:
# Agostinelli, C., (2002). Robust model selection in regression
# via weighted likelihood methodology, Statistics &
# Probability Letters, 56, 289-300.

data(selection)

result <- wle.stepwise(ydata~xdata, boot=100, group=6, num.sol=3,
min.weight=0.8, type="Stepwise", method="WLS")

summary(result)

```

```
wle.stepwise.summaries
```

Accessing summaries for wle.stepwise

Description

All these functions are [methods](#) for class `wle.stepwise` or `summary.wle.stepwise`.

Usage

```

## S3 method for class 'wle.stepwise':
summary(object, num.max=20, verbose=FALSE, ...)

## S3 method for class 'wle.stepwise':
print(x, digits = max(3, getOption("digits") -
3), num.max=max(1, nrow(x$wstep)), ...)

## S3 method for class 'summary.wle.stepwise':
print(x, digits = max(3, getOption("digits") - 3), ...)

```

Arguments

<code>object</code>	an object of class <code>wle.stepwise</code> .
<code>x</code>	an object of class <code>wle.stepwise</code> or <code>summary.wle.stepwise</code> .
<code>digits</code>	number of digits to be used for most numbers.
<code>num.max</code>	the number of the last iterations reported.
<code>verbose</code>	if TRUE warnings are printed.
<code>...</code>	additional arguments affecting the summary produced (in <code>summary.wle.stepwise</code>) or further arguments passed to or from other methods (in <code>print.wle.stepwise</code> and <code>print.summary.wle.stepwise</code>).

Details

The generic accessor functions `coefficients`, `fitted.values`, `residuals` and `weights` can be used to extract various useful features of the value returned by `wle.stepwise`.

Value

The function `summary.wle.stepwise` returns the last `num.max` iterations, call plus:

<code>wstep</code>	the model for each iteration reported.
<code>num.max</code>	the number of iterations reported.
<code>type</code>	the type of selection procedure used.
<code>f.in</code>	the in value.
<code>f.out</code>	the out value.

Author(s)

Claudio Agostinelli

`wle.t.test`

Weighted Likelihood Student's t-Test

Description

`wle.t.test` performs one and two sample Weighted Likelihood t-tests on vectors of data. This is a robust version of the classical t-test. It should be used when the majority of the data follows a normal distribution.

Usage

```
wle.t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
           mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95,
           boot=30, group, num.sol=1, raf="HD", smooth=0.003,
           tol=10-6, equal=10-3, max.iter=500)
```

Arguments

<code>x</code>	a numeric vector of data values.
<code>y</code>	an optional numeric vector data values.
<code>alternative</code>	character specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
<code>mu</code>	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
<code>paired</code>	a logical indicating whether you want a paired weighted t-test.

<code>var.equal</code>	a logical variable indicating whether to treat the two variances as being equal. If <code>TRUE</code> then the pooled variance is used to estimate the variance otherwise the Welch approximation to the degrees of freedom is used.
<code>conf.level</code>	confidence level of the interval.
<code>boot</code>	the number of starting points based on bootstrap subsamples to use in the search of the roots.
<code>group</code>	the dimension of the bootstrap subsamples. The default value is $\max(\text{round}(\text{size}/4), 2)$ where <i>size</i> is the number of observations.
<code>num.sol</code>	maximum number of roots to be searched.
<code>raf</code>	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
<code>smooth</code>	the value of the smoothing parameter.
<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>max.iter</code>	maximum number of iterations.

Details

If `paired` is `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are removed (in pairs if `paired` is `TRUE`). If `var.equal` is `TRUE` then the pooled estimate of the variance is used. By default, if `var.equal` is `FALSE` then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

Value

The function return a list of class `"wle.t.test"` with the following components:

<code>test</code>	A list with two dimensions. Each cell is related with a combination of 'x', 'y' roots. In each cell a list of class <code>"htest"</code> containing the following components: <code>statistic</code> the value of the t-statistic. <code>parameters</code> the degrees of freedom for the t-statistic. <code>p.value</code> the p-value for the test. <code>conf.int</code> a confidence interval for the mean appropriate to the specified alternative hypothesis. <code>estimate</code> the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test. <code>null.value</code> the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test. <code>alternative</code> a character string describing the alternative hypothesis. <code>method</code> a character string indicating what type of t-test was performed.
-------------------	---

```

data.name a character string giving the name(s) of the data.
x.weights the weights related to the 'x' data.
y.weights the weights related to the 'y' data.
x.root the number of the 'x' root.
y.root the number of the 'y' root.
x.tot.sol the number of solutions for the dataset 'x'.
y.tot.sol the number of solutions for the dataset 'y' or 1.
call the match.call().
paired a logical indicating whether is a paired weighted t-test.
x 'x' data.
y 'y' data or NULL.

```

Author(s)

Claudio Agostinelli

References

Agostinelli, C., (1998) Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova (in italian).

Agostinelli, C., (2002) Un approccio alla verifica d'ipotesi robusta basato sulla funzione di verosimiglianza pesata - Robust Testing Hypotheses via Weighted Likelihood function, *Statistica*, Anno LXII, 1, 87-110.

Agostinelli, C., and Markatou, M., (2001) Test of hypotheses based on the Weighted Likelihood Methodology, *Statistica Sinica*, vol. 11, n. 2, 499-514.

Examples

```

library(wle)

set.seed(1234)

x <- rnorm(20,0,1)
y <- rnorm(20,6,1)

t.test(x,y) # P < 2.2e-16
wle.t.test(x,y,group=5) # P < 2.2e-16

t.test(x,y=c(y,250)) # P = 0.1419 -- NOT significant anymore
wle.t.test(x,y=c(y,250),group=5) # P < 2.2e-16 -- still significant
set.seed(1234)

# three roots for 'x' and three roots for 'y'
# with nine t-test value
res <- wle.t.test(x=c(rnorm(40,0,1),rnorm(40,10,1)),
                 y=c(rnorm(40,0,1),rnorm(40,10,1)),
                 group=4,num.sol=3,boot=100)

```

```

print(res) # print ALL the t-test
print(res,x.root=1,y.root=1) # print the test associated to the
                             # x.root=1,y.root=1

root.1.1 <- res$test[[1]][[1]] # access to the object associated
                              # to the x.root=1,y.root=1

names(root.1.1)

set.seed(1234)

# one root and NOT significant t-test
wle.t.test(x=c(rnorm(40,0,1),rnorm(40,10,1)),
           y=c(rnorm(40,0,1),rnorm(40,10,1)),
           group=4,num.sol=3,boot=100,paired=TRUE)

```

wle.var.test	<i>Weighted F Test to Compare Two Variances</i>
--------------	---

Description

Performs an Weighted F test to compare the variances of two samples from normal populations. The WF-test is based on weighted likelihood.

Usage

```

wle.var.test(x, y, ratio = 1, alternative = c("two.sided", "less", "greater"),
            conf.level = 0.95, x.root=1, y.root=1)

```

Arguments

x, y	fitted linear model objects (inheriting from class "wle.lm") or fitted normal model objects (inheriting from class "wle.normal").
ratio	the hypothesized ratio of the population variances of x and y.
alternative	the alternative hypothesis; must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
conf.level	confidence level for the returned confidence interval.
x.root	the 'x' root used.
y.root	the 'y' root used.

Details

The null hypothesis is that the ratio of the variances in the data to which the normal model ([wle.normal](#)) or linear models ([wle.lm](#)) x and y were fitted, is equal to ratio.

Value

A list with class "htest" containing the following components:

statistic	the value of the WF test statistic.
parameter	the degrees of the freedom of the WF distribution of the test statistic.
p.value	the p-value of the test.
conf.int	a confidence interval for the ratio of the population variances.
estimate	the ratio of the sample variances from x and y .
null.value	the ratio of population variances under the null.
alternative	a character string describing the alternative hypothesis.
method	the string "WF test to compare two variances".
data.name	a character string giving the names of the data.

Author(s)

Claudio Agostinelli

References

Agostinelli, C., (1998). Inferenza statistica robusta basata sulla funzione di verosimiglianza pesata: alcuni sviluppi, *Ph.D Thesis*, Department of Statistics, University of Padova (in italian).

Agostinelli, C., (2001) Un approccio robusto alla verifica d'ipotesi basato sulla funzione di verosimiglianza pesata - Robust Testing Hypotheses via Weighted Likelihood function, in press *Statistica*, (in italian).

Agostinelli, C., and Markatou, M., (2001) Test of hypotheses based on the Weighted Likelihood Methodology, *Statistica Sinica*, vol. 11, n. 2, 499-514.

Examples

```
set.seed(2345)

x <- rnorm(50,0,1)
y <- rnorm(50,10,1)

res.x <- wle.normal(x,group=5)
res.y <- wle.normal(y,group=5)

wle.var.test(res.x, res.y) # Do x and y have the same variance?

set.seed(2345)

x <- c(rnorm(50,0,1),rnorm(20,10,1))
y <- c(rnorm(50,10,1),rnorm(10,0,5))

res.x <- wle.normal(x,group=5,num.sol=2)
res.y <- wle.normal(y,group=5)
```

```

res.x
wle.var.test(res.x, res.y, x.root=1)
if (res.x$tot.sol>1) wle.var.test(res.x, res.y, x.root=2)

```

wle.vonmises *von Mises Weighted Likelihood Estimates*

Description

Computes the weighted likelihood estimates for the parameters of a von Mises distribution: the mean direction and the concentration parameter.

Usage

```

wle.vonmises(x, boot = 30, group, num.sol = 1, raf = "HD", smooth, tol =
10^(-6), equal = 10^(-3), max.iter = 500, bias = FALSE, mle.bias =
FALSE, max.kappa = 500, min.kappa = 0.01, use.smooth = TRUE, alpha =
NULL, p = 2, verbose = FALSE, control.circular = list())
## S3 method for class 'wle.vonmises':
print(x, digits = max(3, getOption("digits") - 3), ...)

```

Arguments

x	a vector. The object is coerced to class <code>circular</code> .
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.
equal	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
max.iter	maximum number of iterations.
bias	logical, if TRUE, the estimate for kappa is computed with a bias corrected method. Default is FALSE, i.e. no bias correction.
mle.bias	logical, if TRUE a bias corrected method is used to estimate the concentration parameter for the initial values.
max.kappa	maximum value for the concentration parameter.

<code>min.kappa</code>	minimum value for the concentration parameter.
<code>use.smooth</code>	logical, if TRUE a smoothed model is used, default is TRUE.
<code>alpha</code>	see the next argument <code>p</code> . This is a different parameterization, <code>alpha=-1/2</code> provides Hellinger Distance RAF, <code>alpha=-1</code> provides Kullback-Leibler RAF and <code>alpha=-2</code> provides Neyman's Chi-Square RAF.
<code>p</code>	this parameter works only when <code>raf="HD"</code> . <code>p=2</code> provides Hellinger Distance RAF, <code>p=-1</code> provides Kullback-Leibler RAF and <code>p=Inf</code> provides Neyman's Chi-Square RAF.
<code>verbose</code>	logical, if TRUE warnings are printed.
<code>control.circular</code>	the attribute of the resulting object (<code>mu</code>)
<code>digits</code>	integer indicating the precision to be used.
<code>...</code>	further parameters in <code>print.wle.vonmises</code> .

Details

Parameters `p` and `raf` will be change in the future. See the reference below for the definition of all the RAF.

Value

Returns a list with the following components:

<code>call</code>	the <code>match.call()</code> .
<code>mu</code>	the estimate of the mean direction or the value supplied. If <code>num.sol > 1</code> then <code>mu</code> may have length greater than 1, i.e, one value for each root found.
<code>kappa</code>	the estimate of the concentration parameter or the value supplied. If <code>num.sol > 1</code> then <code>kappa</code> may have length greater than 1, i.e, one value for each root found.
<code>tot.weights</code>	the sum of the weights divide by the number of observations, one value for each root found.
<code>weights</code>	the weights associated to each observation, one column vector for each root found.
<code>f.density</code>	the non-parametric density estimation.
<code>m.density</code>	the smoothed model.
<code>delta</code>	the Pearson residuals.
<code>tot.sol</code>	the number of solutions found.
<code>not.conv</code>	the number of starting points that does not converge after the <code>max.iter</code> iteration are reached.

Author(s)

Claudio Agostinelli

References

C. Agostinelli (2006) Robust Estimation for Circular Data, under revision.

See Also

[circular](#), [mle.vonmises](#).

Examples

```
if (require(circular)) {
  x <- c(rvonmises(n=50, mu=circular(0), kappa=10), rvonmises(n=5, mu=circular(pi/2), kappa=10))
  wle.vonmises(x, smooth=20, group=5)
} else {
  cat("Please, install the package 'circular' in order to use this function.\n")
}
```

wle.weights

Weights based on Weighted Likelihood for the normal model

Description

This function evaluated the weights for the vector ‘x’ using the vector ‘y’ in the estimation of the density by the kernel density estimator.

Usage

```
wle.weights(x, y=NULL, smooth=0.0031, sigma2, raf=1, location=FALSE, max.iter=1000,
```

Arguments

x	the data set for which the weights would be calculate.
y	the data set used to calculate the weights.
smooth	the value of the smoothing parameter.
sigma2	an estimate of the variance.
raf	type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
location	if TRUE the location is estimated. Only available when y=NULL.
max.iter	maximum number of iterations.
tol	the absolute accuracy to be used to achieve convergence of the algorithm.

Value

weights	the weights associated to the x vector.
location	the location.
conv	TRUE if the convergence is achieved.

Author(s)

Claudio Agostinelli

wle.wrappednormal *Wrapped Normal Weighted Likelihood Estimates*

Description

Computes the weighted likelihood estimates for the parameters of a Wrapped Normal distribution: the mean direction and the concentration parameter (and the scale parameter).

Usage

```
wle.wrappednormal(x, mu, rho, sd, K, boot = 30, group, num.sol = 1, raf = "HD",
  smooth = 0.0031, tol = 10^(-6), equal = 10^(-3), min.sd = 0.001,
  min.k = 10, max.iter = 100, use.smooth = TRUE, alpha=NULL, p = 2,
  verbose = FALSE, control.circular=list())

## S3 method for class 'wle.wrappednormal':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	a vector. The object is coerced to class <code>circular</code> .
mu	if a values if provided the parameter is considered known.
rho	if a values if provided the parameter (and <code>sd</code>) is considered known.
sd	if a values if provided the parameter (and <code>rho</code>) is considered known.
K	number of elements used to approximate the density of the wrapped normal.
boot	the number of starting points based on bootstrap subsamples to use in the search of the roots.
group	the dimension of the bootstrap subsamples.
num.sol	maximum number of roots to be searched.
raf	type of Residual adjustment function to be use: <code>raf="HD"</code> : Hellinger Distance RAF, <code>raf="NED"</code> : Negative Exponential Disparity RAF, <code>raf="SCHI2"</code> : Symmetric Chi-Squared Disparity RAF.
smooth	the value of the smoothing parameter.

<code>tol</code>	the absolute accuracy to be used to achieve convergence of the algorithm.
<code>equal</code>	the absolute value for which two roots are considered the same. (This parameter must be greater than <code>tol</code>).
<code>min.sd</code>	minimum value for the <code>sd</code> parameter.
<code>min.k</code>	minimum number of elements used to approximate the density of the wrapped normal.
<code>max.iter</code>	maximum number of iterations.
<code>use.smooth</code>	logical, if TRUE a smoothed model is used, default is TRUE.
<code>alpha</code>	see the next argument <code>p</code> . This is a different parameterization, <code>alpha=-1/2</code> provides Hellinger Distance RAF, <code>alpha=-1</code> provides Kullback-Leibler RAF and <code>alpha=-2</code> provides Neyman's Chi-Square RAF.
<code>p</code>	this parameter works only when <code>raf="HD"</code> . <code>p=2</code> provide Hellinger Distance RAF, <code>p=-1</code> provide Kullback-Leibler RAF and <code>p=Inf</code> provide Neyman's Chi-Square RAF.
<code>verbose</code>	logical, if TRUE warnings are printed.
<code>control.circular</code>	the attribute of the resulting objects (<code>mu</code>)
<code>digits</code>	integer indicating the precision to be used.
<code>...</code>	further parameters in <code>print.wle.vonmises</code> .

Details

Parameters `p` and `raf` will be change in the future. See the reference below for the definition of all the RAF.

Value

Returns a list with the following components:

<code>call</code>	the <code>match.call()</code> .
<code>mu</code>	the estimate of the mean direction or the value supplied. If <code>num.sol > 1</code> then <code>mu</code> may have length greater than 1, i.e, one value for each root found.
<code>rho</code>	the estimate of the concentration parameter or the value supplied. If <code>num.sol > 1</code> then <code>rho</code> may have length greater than 1, i.e, one value for each root found.
<code>sd</code>	the estimate of the standard deviation parameter or the value supplied. If <code>num.sol > 1</code> then <code>sd</code> may have length greater than 1, i.e, one value for each root found.
<code>tot.weights</code>	the sum of the weights divide by the number of observations, one value for each root found.
<code>weights</code>	the weights associated to each observation, one column vector for each root found.
<code>f.density</code>	the non-parametric density estimation.
<code>m.density</code>	the smoothed model.
<code>delta</code>	the Pearson residuals.
<code>tot.sol</code>	the number of solutions found.
<code>not.conv</code>	the number of starting points that does not converge after the <code>max.iter</code> iteration are reached.

Author(s)

Claudio Agostinelli

References

C. Agostinelli (2003) Robust Estimation for Circular Data, manuscript.

See Also

[circular](#), [mle.wrappednormal](#).

Examples

```
if (require(circular)) {  
  x <- c(rwrappednormal(n=50, mu=circular(0), sd=1), rwrappednormal(n=5, mu=circular(pi/2),  
    wle.wrappednormal(x, smooth=1/20, group=5)  
} else {  
  cat("Please, install the package 'circular' in order to use this function.\n")  
}
```

Index

- *Topic **arith**
 - binary, 2
- *Topic **datasets**
 - artificial, 1
 - cavendish, 3
 - hald, 3
 - rocky, 24
 - selection, 24
- *Topic **htest**
 - wle.t.test, 74
 - wle.var.test, 77
- *Topic **models**
 - wle.binomial, 36
 - wle.gamma, 49
 - wle.normal, 56
 - wle.normal.mixture, 58
 - wle.normal.multi, 61
 - wle.normal.multi.summaries, 63
 - wle.normal.summaries, 63
 - wle.poisson, 67
- *Topic **multivariate**
 - wle.normal.multi, 61
 - wle.normal.multi.summaries, 63
- *Topic **regression**
 - mle.aic, 8
 - mle.aic.summaries, 10
 - mle.cp, 11
 - mle.cp.summaries, 13
 - mle.cv, 14
 - mle.cv.summaries, 16
 - mle.stepwise, 17
 - mle.stepwise.summaries, 19
 - plot.mle.cp, 20
 - plot.wle.cp, 21
 - plot.wle.lm, 22
 - wle.aic, 25
 - wle.aic.summaries, 32
 - wle.cp, 38
 - wle.cp.summaries, 41
 - wle.cv, 42
 - wle.cv.summaries, 45
 - wle.lm, 51
 - wle.lm.summaries, 54
 - wle.onestep, 64
 - wle.onestep.summaries, 66
 - wle.stepwise, 70
 - wle.stepwise.summaries, 73
- *Topic **robust**
 - mde.vonmises, 4
 - mde.wrappednormal, 6
 - plot.wle.cp, 21
 - plot.wle.lm, 22
 - wle.aic, 25
 - wle.aic.ar, 28
 - wle.aic.ar.summaries, 31
 - wle.aic.summaries, 32
 - wle.ar, 33
 - wle.binomial, 36
 - wle.cp, 38
 - wle.cp.summaries, 41
 - wle.cv, 42
 - wle.cv.summaries, 45
 - wle.fracdiff, 46
 - wle.gamma, 49
 - wle.lm, 51
 - wle.lm.summaries, 54
 - wle.normal, 56
 - wle.normal.mixture, 58
 - wle.normal.multi, 61
 - wle.normal.multi.summaries, 63
 - wle.normal.summaries, 63
 - wle.onestep, 64
 - wle.onestep.summaries, 66
 - wle.poisson, 67
 - wle.smooth, 68

- wle.stepwise, 70
- wle.stepwise.summaries, 73
- wle.t.test, 74
- wle.var.test, 77
- wle.vonmises, 79
- wle.weights, 81
- wle.wrappednormal, 82
- *Topic ts**
 - wle.aic.ar, 28
 - wle.aic.ar.summaries, 31
 - wle.ar, 33
 - wle.fracdiff, 46
- artificial, 1
- binary, 2
- cavendish, 3
- circular, 4–6, 8, 79, 81, 82, 84
- class, 9, 12, 15, 17, 26, 37, 39, 43, 50, 52, 57, 60, 61, 65, 67, 69, 71
- coef.wle.lm (*wle.lm.summaries*), 54
- coefficients, 74
- fitted.values, 74
- fitted.wle.lm (*wle.lm.summaries*), 54
- formula.wle.lm (*wle.lm.summaries*), 54
- hald, 3
- mde.vonmises, 4
- mde.wrappednormal, 6
- methods, 10, 13, 16, 19, 31, 32, 41, 45, 54, 63, 66, 73
- mle.aic, 8, 11
- mle.aic.summaries, 10
- mle.cp, 11, 14, 20
- mle.cp.summaries, 13
- mle.cv, 14, 16
- mle.cv.summaries, 16
- mle.stepwise, 17
- mle.stepwise.summaries, 19
- mle.vonmises, 5, 81
- mle.wrappednormal, 7, 8, 84
- model.frame.wle.lm (*wle.lm.summaries*), 54
- na.fail, 55
- na.omit, 55
- options, 55
- par, 23
- plot.mle.cp, 20
- plot.wle.cp, 21
- plot.wle.lm, 22, 56
- print.mde.vonmises (*mde.vonmises*), 4
- print.mde.wrappednormal (*mde.wrappednormal*), 6
- print.mle.aic (*mle.aic.summaries*), 10
- print.mle.cp (*mle.cp.summaries*), 13
- print.mle.cv (*mle.cv.summaries*), 16
- print.mle.stepwise (*mle.stepwise.summaries*), 19
- print.summary.mle.aic (*mle.aic.summaries*), 10
- print.summary.mle.cp (*mle.cp.summaries*), 13
- print.summary.mle.cv (*mle.cv.summaries*), 16
- print.summary.mle.stepwise (*mle.stepwise.summaries*), 19
- print.summary.wle.aic (*wle.aic.summaries*), 32
- print.summary.wle.aic.ar (*wle.aic.ar.summaries*), 31
- print.summary.wle.cp (*wle.cp.summaries*), 41
- print.summary.wle.cv (*wle.cv.summaries*), 45
- print.summary.wle.lm (*wle.lm.summaries*), 54
- print.summary.wle.stepwise (*wle.stepwise.summaries*), 73
- print.wle.aic (*wle.aic.summaries*), 32
- print.wle.aic.ar (*wle.aic.ar.summaries*), 31
- print.wle.binomial (*wle.binomial*), 36

- print.wle.cp(*wle.cp.summaries*),
41
- print.wle.cv(*wle.cv.summaries*),
45
- print.wle.gamma(*wle.gamma*), 49
- print.wle.lm(*wle.lm.summaries*),
54
- print.wle.normal
(*wle.normal.summaries*), 63
- print.wle.normal.mixture
(*wle.normal.mixture*), 58
- print.wle.normal.multi
(*wle.normal.multi.summaries*),
63
- print.wle.onestep
(*wle.onestep.summaries*), 66
- print.wle.poisson(*wle.poisson*),
67
- print.wle.smooth(*wle.smooth*), 68
- print.wle.stepwise
(*wle.stepwise.summaries*),
73
- print.wle.t.test(*wle.t.test*), 74
- print.wle.vonmises
(*wle.vonmises*), 79
- print.wle.wrappednormal
(*wle.wrappednormal*), 82

- residuals, 74
- rocky, 24

- selection, 24
- summary.mle.aic
(*mle.aic.summaries*), 10
- summary.mle.cp
(*mle.cp.summaries*), 13
- summary.mle.cv
(*mle.cv.summaries*), 16
- summary.mle.stepwise
(*mle.stepwise.summaries*),
19
- summary.wle.aic
(*wle.aic.summaries*), 32
- summary.wle.aic.ar
(*wle.aic.ar.summaries*), 31
- summary.wle.cp
(*wle.cp.summaries*), 41
- summary.wle.cv
(*wle.cv.summaries*), 45

- summary.wle.lm
(*wle.lm.summaries*), 54
- summary.wle.stepwise
(*wle.stepwise.summaries*),
73

- try, 50

- uniroot, 50, 70

- weights, 74
- weights.wle.lm
(*wle.lm.summaries*), 54
- wle.aic, 25, 33
- wle.aic.ar, 28, 32
- wle.aic.ar.summaries, 31
- wle.aic.summaries, 32
- wle.ar, 30, 33
- wle.ar.wls(*wle.aic.ar*), 28
- wle.binomial, 36
- wle.cp, 22, 38, 41
- wle.cp.summaries, 41
- wle.cv, 42, 45
- wle.cv.summaries, 45
- wle.fracdiff, 46
- wle.gamma, 49
- wle.lm, 22, 23, 40, 44, 51, 56, 66, 72, 77
- wle.lm.summaries, 54
- wle.normal, 56, 59, 64, 77
- wle.normal.mixture, 58
- wle.normal.multi, 61, 63
- wle.normal.multi.summaries, 63
- wle.normal.summaries, 63
- wle.onestep, 64, 66
- wle.onestep.summaries, 66
- wle.poisson, 67
- wle.smooth, 40, 44, 53, 58, 62, 66, 68, 72
- wle.stepwise, 70
- wle.stepwise.summaries, 73
- wle.t.test, 74
- wle.var.test, 77
- wle.vonmises, 5, 79
- wle.weights, 81
- wle.wrappednormal, 8, 82

- x.artificial(*artificial*), 1
- x.hald(*hald*), 3
- xdata(*selection*), 24

- y.artificial(*artificial*), 1

`y.hald(hald), 3`
`ydata(selection), 24`