Package ‘wle’

October 18, 2015

Title Weighted Likelihood Estimation
LazyLoad yes
LazyData yes
Version 0.9-91
Author Claudio Agostinelli <claudio.agostinelli@unitn.it>, SLATEC Common Mathematical Library <www.netlib.org/slatec>
Maintainer Claudio Agostinelli <claudio.agostinelli@unitn.it>
Depends R (>= 3.0.0), circular,
Date 2015-10-17
Description Approach to the robustness via Weighted Likelihood.
License GPL-2
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-10-18 18:41:42

R topics documented:

anova.wle.glm.root ......................................................... 3
artificial ............................................................................ 4
binary ................................................................................. 5
cavendish ........................................................................... 6
extractRoot ......................................................................... 6
hald ..................................................................................... 9
mde.vonmises ....................................................................... 9
mde.wrappednormal ............................................................ 11
mle.aic .............................................................................. 14
mle.aic.summaries ............................................................. 16
mle.cp ............................................................................... 17
mle.cp.summaries .............................................................. 18
mle.cv ............................................................................... 20
mle.cv.summaries .............................................................. 21
mle.stepwise ....................................................................... 22
R topics documented:

- mle.stepwise.summaries ................................................... 24
- plot.mle.cp ................................................................. 25
- plot.wle.cp ................................................................. 27
- plot.wle.lm ............................................................... 28
- residuals.Ancombe ....................................................... 30
- rocky ................................................................. 31
- selection ............................................................... 32
- summary.wle.glm ....................................................... 32
- wle.aic ................................................................. 34
- wle.aic.ar ............................................................... 37
- wle.aic.ar.summaries .................................................. 41
- wle.aic.summaries ..................................................... 42
- wle.ar ................................................................. 43
- wle.binomial ........................................................... 46
- wle.cp ................................................................. 47
- wle.cp.summaries ...................................................... 51
- wle.cv ................................................................. 52
- wle.cv.summaries ...................................................... 55
- wle.fracdiff ............................................................ 56
- wle.gamma ............................................................. 59
- wle.glm ................................................................. 61
- wle.glm.control ........................................................ 67
- wle.glm.summaries ..................................................... 69
- wle.glm.weights ......................................................... 70
- wle.lm ................................................................. 72
- wle.lm.summaries ...................................................... 75
- wle.negativebinomial .................................................. 77
- wle.normal ............................................................. 79
- wle.normal.mixture ..................................................... 81
- wle.normal.multi ....................................................... 83
- wle.normal.multi.summaries ......................................... 85
- wle.normal.summaries ................................................ 86
- wle.onestep ............................................................ 87
- wle.onestep.summaries ............................................... 89
- wle.poisson ........................................................... 89
- wle.smooth ........................................................... 91
- wle.stepwise .......................................................... 93
- wle.stepwise.summaries .............................................. 96
- wle.t.test ............................................................. 97
- wle.var.test .......................................................... 100
- wle.vonmises .......................................................... 102
- wle.weights ........................................................... 104
- wle.wrappednormal .................................................... 105

Index ............................................................. 108
Robust Analysis of Deviance for Generalized Linear Model Fits

Description

Compute a robust analysis of deviance table for one or more generalized linear model fits.

Usage

```r
# S3 method for class 'wle.glm.root'
anova(object, ..., dispersion = NULL, test = NULL)
```

Arguments

- `object, ...`: objects of class `wle.glm.root`, typically the result of a call to `extractRoot.wle.glm`, or a list of objects each of which a result of a call to "extractRoot.wle.glm" method.
- `dispersion`: the dispersion parameter for the fitting family. By default it is obtained from the object(s).
- `test`: a character string, (partially) matching one of "Chisq", "F" or "Gp". See `stat.anova`.

Details

Specifying a single object gives a sequential analysis of deviance table for that fit. That is, the reductions in the residual deviance as each term of the formula is added in turn are given in as the rows of a table, plus the residual deviances themselves.

If more than one object is specified, the table has a row for the residual degrees of freedom and deviance for each model. For all but the first model, the change in degrees of freedom and deviance is also given. (This only makes statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

The table will optionally contain test statistics (and P values) comparing the reduction in deviance for the row to the residuals. For models with known dispersion (e.g., binomial and Poisson fits) the robust chi-squared test is most appropriate, and for those with dispersion estimated by moments (e.g., gaussian, quasibinomial and quasipoisson fits) the Robust F test is most appropriate. Robust Mallows’ $C_p$ statistic is the residual weighted deviance plus twice the estimate of $\sigma^2$ times the residual (weighted) degrees of freedom, which is closely related to Robust AIC (and a multiple of it if the dispersion is known).

The dispersion estimate will be taken from the largest model, using the value returned by `summary.wle.glm`. As this will in most cases use a Chisquared-based estimate, the F tests are not based on the residual deviance in the analysis of deviance table shown.

Value

An object of class "anova" inheriting from class "data.frame".
Warning

The comparison between two or more models by `anova.wle.glm.root` or `anova.wleglm.list` will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R's default of `na.action = na.omit` is used, and `anova.wleglm.list` will detect this with an error.

Since in a model selection procedure and/or on an ANOVA table the weights of the WLE procedure must be that of the FULL model (and not that of the actual model) statistics on degrees of freedom, deviance and AIC are valid only if object is the FULL model.

References


See Also

`extractRoot.wle.glm`, `wle.glm`, `anova`.

Examples

```r
## --- Continuing the Example from '?wle.glm':
anova(extractRoot(wle.glm.D93))
anova(extractRoot(wle.glm.D93), test = "Cp")
anova(extractRoot(wle.glm.D93), test = "Chisq")
```

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


See Also


```
# Convert decimal base number to binary base

# Usage
binary(x, dim)

# Arguments
x       a number in decimal base.
dim     the number of digits, if missing the right number of digits is evaluated.

# Value
binary   a vector representing the 'x' number in binary base.
dicotomy 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'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


See Also


Extract a Root from a result of a wle function

Description

This function extract the information regarding one solution of the Weighted Likelihood Estimating Equation.

Usage

## S3 method for class 'wle.glm'
ejectRoot(object, root=1, ...)
Arguments

object  an object of class "wle.glm", usually, a result of a call to \texttt{wle.glm}.

root  an integer number to specify which root should be extract.

... further arguments passed to or from other methods.

Value

\texttt{extract.wle.glm} returns an object of class "extract.wle.glm.root", a (variable length) list containing at least the following components:

- \texttt{coefficients} a named vector of coefficients
- \texttt{residuals} the \textit{working} residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are \texttt{NA}.
- \texttt{fitted.values} the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
- \texttt{rank} the numeric rank of the fitted linear model.
- \texttt{family} the \texttt{family} object used.
- \texttt{linear.predictors} the linear fit on link scale.
- \texttt{deviance} up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
- \texttt{aic} Akaike's \textit{An Information Criterion}, minus twice the maximized log-likelihood plus twice the number of coefficients (so assuming that the dispersion is known).
- \texttt{null.deviance} The deviance for the null model, comparable with \texttt{deviance}. The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
- \texttt{iter} the number of iterations of IWLS used.
- \texttt{weights} the \textit{working} weights, that is the weights in the final iteration of the IWLS fit.
- \texttt{prior.weights} the weights initially supplied, a vector of 1s if none were.
- \texttt{df.residual} the residual degrees of freedom.
- \texttt{df.null} the residual degrees of freedom for the null model.
- \texttt{y} if requested (the default) the y vector used. (It is a vector even for a binomial model.)
- \texttt{x} if requested, the model matrix.
- \texttt{model} if requested (the default), the model frame.
- \texttt{converged} logical. Was the IWLS algorithm judged to have converged?
- \texttt{boundary} logical. Is the fitted value on the boundary of the attainable values?
- \texttt{wle.weights} final (robust) weights based on the WLE approach.
- \texttt{wle.asymptotic} logicals. If \texttt{TRUE} asymptotic weight based on Anscombe residual is used for the corresponding observation.
In addition, non-empty fits will have components qr, R, qraux, pivot and effects relating to the final weighted linear fit.

family the family object used.
call the matched call.
formula the formula supplied.
terms the terms object used.
data the data argument.
offset the offset vector used.
control the value of the control argument used.
method the name of the fitter function used, currently always "wle.glm.fit".
contrasts (where relevant) the contrasts used.
xlevels (where relevant) a record of the levels of the factors used in fitting.
tot.sol the number of solutions found.
not.conv the number of starting points that does not converge after the max.iter (defined using wle.glm.control) iterations are reached.
na.action (where relevant) information returned by model.frame on the special handling of NAs.

If a binomial wle.glm model was specified by giving a two-column response, the weights returned by prior.weights are the total numbers of cases (factored by the supplied case weights) and the component y of the result is the proportion of successes.

Author(s)
Claudio Agostinelli and Fatemah Al-quallaf

See Also
anova.wle.glm.root

Examples
## --- Continuing the Example from '?wle.glm':

anova(extractRoot(wle.glm.D93))
**Hald Data**

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**

```r
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**


---

**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**

```r
mde.vonmises(x, bw, mu = NULL, kappa = NULL, n = 512,
              from = circular(0), to = circular(2 * pi), lower = NULL,
              upper = NULL, method = "L-BFGS-B", lower.kappa = .Machine$double.eps,
              upper.kappa = Inf, alpha = NULL, p = 2, control.circular = list(), ...)
```

## S3 method for class 'mde.vonmises'

```r
print(x, digits = max(3,getOption("digits") - 3), ...)
```
Arguments

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

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:

- **call**: the match.call().
- **mu**: the estimate of the mean direction.
- **kappa**: the estimate of the concentration parameter.
- **dist**: the distance between the estimated density and the model.
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 = 1e-05, n = 512, from = circular(0),
to = circular(2 * pi), lower = NULL, upper = NULL,
method = "L-BFGS-B", lower.rho = 1e-06, upper.rho = 1 - 1e-06,
min.sd = 0.001, K = NULL, min.k = 10, control.circular = list(), ...)
## S3 method for class 'mde.wrappednormal'
print(x, digits = max(3, getOption("digits") - 3), ...)
Arguments

- **x**: a vector. The object is coerced to class `circular`.
- **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 `rho` is `NULL`. Default: maximum likelihood estimate.
- **alpha**: if not `NULL` overrides the value of `p`. See the next argument `p`. This is a different parameterization, `alpha=-1/2` provides Hellinger distance, `alpha=-1` provides Kullback-Leibler distance and `alpha=-2` provides Neyman’s Chi-Square distance.
- **p**: `p=2` provides Hellinger distance, `p=-1` provides Kullback-Leibler distance and `p=Inf` provides Neyman’s Chi-Square distance. It is ignored if `alpha` is not `NULL`.
- **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 `mle.wrappednormal`.
- **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 `optim` used to constrained optimization. First element for the mean direction, second element for the concentration.
- **upper**: a 2 elements vector passed to `optim` used to constrained optimization. First element for the mean direction, second element for the concentration.
- **method**: passed to `optim`.
- **lower.rho**: if `lower` is `NULL` this parameter is used to constrained optimization for the concentration parameter.
- **upper.rho**: if `upper` is `NULL` this parameter is used to constrained optimization for the concentration parameter.
- **min.sd**: minimum value for the `sd` parameter. This argument is passed to the function which determined the Maximum Likelihood estimates of the parameters. See `mle.wrappednormal`.
- **min.k**: number of elements used to approximate the density of the wrapped normal.
- **control.circular**: the attribute of the resulting object (`mu`)
- **digits**: integer indicating the precision to be used.
- **...**: further parameters in `print.mde.wrappednormal`.
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:

- `call` the match.call().
- `mu` the estimate of the mean direction.
- `rho` the estimate of the concentration parameter.
- `sd` the estimate of the standard deviation parameter.
- `dist` the distance between the estimated density and the model.
- `data` 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. Robust estimation for circular data. Computational Statistics & Data Analysis,

See Also

circular, mle.wrappednormal and wle.wrappednormal.

Examples

```r
set.seed(1234)
x <- c(rwrappednormal(n=200, mu=circular(0), sd=0.6),
      rwrappednormal(n=20, mu=circular(pi/2), sd=0.1))
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))
```
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 `mle.aic` is called from.
- **model, x, y**: logicals. If TRUE 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 contrasts.arg of `model.matrix.default`.
- **se**: logical. if TRUE the returning object contains standard errors for the parameters of every model.
- **verbose**: if TRUE 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:
mle.aic

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

Examples
library(mle)
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

```r
## 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

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

Value

summary.mle.aic returns a list:

- **aic**: the first num.max best models with their AIC.
- **num.max**: the number of models reported.
- **call**

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**

```r
mle.cp(formula, data=list(), model=TRUE, x=FALSE, y=FALSE, var.full=0, 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 `mle.cp` is called from.
- `model, x, y` logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response).
- `var.full` 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.
- `contrasts` an optional list. See the `contrasts.arg` of `model.matrix.default`.
- `verbose` if TRUE 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:

- `cp` Mallows Cp 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.

Author(s)
Claudio Agostinelli

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

Examples
library(mle)
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=20L, verbose=FALSE, ...)

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

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

Arguments

- **object**: an object of class mle.cp.
- **x**: an object of class mle.cp or summary.mle.cp.
- **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 summary.mle.cp) or further arguments passed to or from other methods (in print.mle.cp and print.summary.mle.cp).

Value

summary.mle.cp returns a list:

- **cp**: the first num.max 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.
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 mle.cv is called from.
model, x, y logicals. If TRUE 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(round(size(3/4)), nvar + 2).
contrasts an optional list. See the contrasts.arg of model.matrix.default.
verbose if TRUE 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 the model frame.
xlevels the model frame.
terms if model=TRUE a matrix with first column the dependent variable and the remain
column the explanatory variables for the full model.
model if x=TRUE a matrix with the explanatory variables for the full model.
x if y=TRUE a vector with the dependent variable.
y not well working yet, if 0 no error occurred.

Author(s)
Claudio Agostinelli

References

Examples
library(mle)
data(hald)
cor(hald)
result <- mle(y.hald~x.hald)
summary(result)

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

- **object**: an object of class `mle.cv`.
- **x**: an object of class `mle.cv` or `summary.mle.cv`.
- **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 `summary.mle.cv`) or further arguments passed to or from other methods (in `print.mle.cv` and `print.summary.mle.cv`).

Value

`summary.mle.cv` returns a list:

- **cv**: the first `num.max` best models with their estimated prediction error using CV.
- **num.max**: the number of models reported.
- **call**

Author(s)

Claudio Agostinelli

See Also

- `mle.cv` a function for evaluate the Cross-Validation selection criterion for linear models.

---

**mle.stepwise**

*Stepwise, Backward and Forward selection methods*

Description

This function performs Stepwise, Forward and Backward model selection.

Usage

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

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 \texttt{mle.stepwise} is called from.
- **model, x, y**: logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response.)
- **type**: \texttt{type="Stepwise"}: the stepwise methods is used, \texttt{type="Forward"}: the forward methods is used, \texttt{type="Backward"}: the backward method is used.
- **f.in**: the in value
- **f.out**: the out value
- **contrasts**: an optional list. See the \texttt{contrasts.arg} of \texttt{model.matrix.default}
- **verbose**: if TRUE warnings are printed.

Details

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

Value

\texttt{mle.stepwise} returns an object of \texttt{class "mle.stepwise"}.

The function \texttt{summary} is used to obtain and print a summary of the results.

The object returned by \texttt{mle.stepwise} are:

- **step**: the selected models
- **type**: the type o model selection procedure was used.
- **f.in**: the value of \texttt{f.in} used.
- **f.out**: the value of \texttt{f.out} used.
- **call**: the \texttt{match.call}.
- **contrasts**: the model frame.
- **xlevels**: if \texttt{model=TRUE} a matrix with first column the dependent variable and the remain column the explanatory variables for the full model.
- **terms**: if \texttt{x=TRUE} a matrix with the explanatory variables for the full model.
- **model**: if \texttt{y=TRUE} a vector with the dependent variable.
- **info**: not well working yet, if 0 no error occurred.
Author(s)
Claudio Agostinelli

References

Examples

```r
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**

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

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

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

- **object**: an object of class `mle.stepwise`.
- **x**: an object of class `mle.stepwise` or `summary.mle.stepwise`.
- **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 `summary.mle.stepwise`) or further arguments passed to or from other methods (in `print.mle.stepwise` and `print.summary.mle.stepwise`).

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

```r
## 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

- **x**: an object of class `mle.cp`.
- **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 `base.line`
- **num.bad**: number of submodels above the `base.line`
- **cp.good**: list of the submodels below the `base.line` with their Cp.
- **cp.bad**: list of the submodels above the `base.line` with their Cp.

Author(s)

Claudio Agostinelli

See Also

- `mle.cp` a function to calculate the Mallows Cp.

Examples

```r
library(wle)
data(hald)
result <- mle.cp(y.hald~x.hald)
plot(result,num.max=7)
```
Description

Plot the weighted Mallows Cp based on weighted likelihood.

Usage

```r
## 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 `wle.cp`.
- `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 `base.line`.
- `num.bad` number of submodels above the `base.line`.
- `wcp.good` list of the submodels below the `base.line` with their W Cp.
- `wcp.bad` list of the submodels above the `base.line` with their W Cp.

Author(s)

Claudio Agostinelli
References


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")
x.data <- cbind(x.data, x.data^2, x.data^3, log(x.data+1))
result <- wle.cp(y.data~x.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, verbose=FALSE, ...)
Arguments

- **x**: an object of class `wle.lm`.
- **roots**: a vector specifying for which roots the plots are required.
- **which**: if a subset of the plots for each root is required, specify a subset of the numbers 0:4, 0 means no plots.
- **which.main**: 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.
- **level.weight**: value of the weight under which an observation is marked with different color.
- **ask**: logical; if TRUE, the user is asked before each plot, see `par(ask=.)`.
- **col**: a vector of 3 elements, to specify colors for the plots.
- **id.n**: number of points to be labelled in some plots, starting with the ones with less weight.
- **labels.id**: vector of labels, from which the labels for less weighted points will be chosen. If missing uses observation numbers.
- **cex.id**: magnification of point labels.
- **verbose**: if TRUE warnings are printed.
- **...**: 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

```r
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

residualsAnscombe  

---

## Description

Evaluate the Anscombe residuals for a given type of family in GLM.

## Usage

residualsAnscombe(y, mu, family, ...)

## Arguments

- `y` vector of the response variable
- `mu` vector of the same length as `y` with the corresponding fitted.values.
- `family` an object of class family.
- `...` not used yet.

## Details

The function performs the Anscombe transformation to obtain residuals that are asymptotically normal distributed. For the Binomial family (see Con and Snell 1968) the transformation is

\[
\beta(\frac{2}{3}, \frac{2}{3}) \times (pbeta(y/m, 2/3, 2/3) - pbeta(mu - (1 - 2*mu)/(6*m), 2/3, 2/3))/((mu^{1/6}*(1 - mu^{1/6}))/sqrt(m))
\]

where `m` is the number of trial and `y` the number of successes. For the Poisson family (see Con and Snell 1968) the transformation is

\[
(3/2 \times (y^{2/3} - (mu - 1/6)^{(2/3)})/(mu^{1/6}))
\]

while for the Gamma family (see McCullagh and Nelder 1989) the transformation is

\[
3 \times (y^{1/3} - mu^{1/3})/(mu^{1/3})
\]

and for the Inverse Gaussian family (see McCullagh and Nelder 1989) the transformation is

\[
(ln(y) - ln(mu))/\sqrt(mu)
\]

## Value

It return a vector with the Anscombe residuals.
Author(s)
Claudio Agostinelli and Fatemah Al-quallaf

References


See Also
wle.glm

Examples
```r
## Dobson (1990) Page 93: Randomized Controlled Trial:
counts <- c(18,17,15,20,10,18,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AO <- data.frame(treatment, outcome, counts))
wle.glm.D93 <- wle.glm(counts ~ outcome + treatment, family=poisson())
res <- residualsAnscombe(counts, mu=wle.glm.D93$root1$fitted.values, family=poisson())
qqnorm(res)
qqline(res)
```

---

rocky

*Rockwell hardness, 100 coils produced in sequence at a Chicago Steel Mill Data*

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


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


Description

These functions are all methods for class wle.glm or summary.wle.glm objects.

Usage

```r
# S3 method for class 'wle.glm'
summary(object, root = 1, dispersion = NULL,
correlation = FALSE, symbolic.cor = FALSE, ...)

# S3 method for class 'summary.wle.glm'
print(x, digits = max(3, getOption("digits") - 3),
symbolic.cor = x$symbolic.cor,
signif.stars = getOption("show.signif.stars"), ...)
```
Arguments

object: an object of class "wle.glm", usually, a result of a call to \texttt{wle.glm}.

root: an integer number to specify for which root the summary should be reported.

x: an object of class "summary.wle.glm", usually, a result of a call to \texttt{summary.glm}.

dispersion: the dispersion parameter for the family used. Either a single numerical value or \texttt{NULL} (the default), when it is inferred from \texttt{object} (see 'Details').

correlation: logical; if \texttt{TRUE}, the correlation matrix of the estimated parameters is returned and printed.

digits: the number of significant digits to use when printing.

symbolic.cor: logical. If \texttt{TRUE}, print the correlations in a symbolic form (see \texttt{symnum}) rather than as numbers.

signif.stars: logical. If \texttt{TRUE}, 'significance stars' are printed for each coefficient.

... further arguments passed to or from other methods.

Details

\texttt{print.summary.wle.glm} tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives 'significance stars' if \texttt{signif.stars} is \texttt{TRUE}. The coefficients component of the result gives the estimated coefficients and their estimated standard errors, together with their ratio. This third column is labelled \texttt{t ratio} if the dispersion is estimated, and \texttt{z ratio} if the dispersion is known (or fixed by the family). A fourth column gives the two-tailed p-value corresponding to the \texttt{t} or \texttt{z} ratio based on a Student t or Normal reference distribution. (It is possible that the dispersion is not known and there are no residual degrees of freedom from which to estimate it. In that case the estimate is NaN.)

Aliased coefficients are omitted in the returned object but restored by the \texttt{print} method.

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print \texttt{summary(object)\$correlation} directly.

The dispersion of a GLM is not used in the fitting process, but it is needed to find standard errors. If \texttt{dispersion} is not supplied or \texttt{NULL}, the dispersion is taken as 1 for the binomial and Poisson families, and otherwise estimated by the residual Chi-squared statistic (calculated from cases with non-zero weights) divided by the residual degrees of freedom.

\texttt{summary} can be used with Gaussian \texttt{wle.glm} fits to handle the case of a linear regression with known error variance, something not handled by \texttt{summary.wle.lm}.

Value

\texttt{summary.wle.glm} returns an object of class "summary.wle.glm", a list with components

\begin{itemize}
  \item \texttt{call} \hspace{1cm} the component from \texttt{object}.
  \item \texttt{family} \hspace{1cm} the component from \texttt{object}.
  \item \texttt{deviance} \hspace{1cm} the component from \texttt{object}.
  \item \texttt{contrasts} \hspace{1cm} the component from \texttt{object}.
  \item \texttt{df.residual} \hspace{1cm} the component from \texttt{object}.
\end{itemize}
null.deviance  the component from object.
df.null       the component from object.
deviance.resid the deviance residuals; see `residuals.glm`.
coefficients  the matrix of coefficients, standard errors, z-values and p-values. Aliased coefficients are omitted.
aliased       named logical vector showing if the original coefficients are aliased.
dispersion    either the supplied argument or the inferred/estimated dispersion if the latter is `null`.
df            a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of non-aliased coefficients.
cov.unscaled  the unscaled (\( \text{dispersion} = 1 \)) estimated covariance matrix of the estimated coefficients.
cov.scaled    ditto, scaled by `dispersion`.
correlation   (only if `correlation` is true.) The estimated correlations of the estimated coefficients.
symbolic.cor  (only if `correlation` is true.) The value of the argument `symbolic.cor`.

Warnings
Since in a model selection procedure and/or on an ANOVA table the weights of the WLE procedure must be that of the FULL model (and not that of the actual model) statistics on degrees of freedom, deviance and AIC are valid only if this is the FULL model.

See Also
`wle.glm`, `summary`.

Examples
```r
## --- Continuing the Example from '?wle.glm':
summary(wle.glm.D93)
```

wle.aic  Weighted Akaike Information Criterion

Description
The Weighted Akaike Information Criterion.

Usage
```r
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 wle.aic 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 \text{size} is the number of observations and \text{var} 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 tol).

- **max.iter**: maximum number of iterations.

- **min.weight**: see details.

- **method**: see details.

- **alpha**: penalty value.

- **contrasts**: an optional list. See the contrasts.arg of model.matrix.default.

- **verbose**: 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:

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


Examples

```r
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, 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", var.full = 0, 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, min.weights.aic = 0.5, approx.w = TRUE,
  ask = TRUE, alpha = 2, method = "WLS")

Arguments

x a univariate time series.

order 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.

seasonal a specification of the seasonal part of the ARI model, plus the period (which
defaults to frequency(x)).

group the dimension of the bootstrap subsamples.

group.start the dimension of the bootstrap subsamples used in the starting process if wle.init=TRUE.

group.step the dimension of the bootstrap subsamples used in a step, it must be less than
group.

xreg optionally, a vector or matrix of external regressors, which must have the same
number of rows as x.

include.mean Should the ARI model include a mean term? The default is TRUE for undiffer-
enced series, FALSE for differenced ones (where a mean would not affect the fit nor predictions).

na.action function to be applied to remove missing values.

tol the absolute accuracy to be used to achieve convergence of the algorithm.

tol.step the absolute accuracy to be used to achieve convergence in a step.

equal the absolute value for which two roots are considered the same. (This parameter
must be greater than tol).

equal.step the absolute value for which two roots are considered the same in a step. (This
parameter must be greater than tol.step).

raf type of Residual adjustment function to be use: raf="HD": Hellinger Distance
RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCH2": Sym-
metric Chi-Squared Disparity RAF.

var.full An estimate of the residual variance for the full model.

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

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

Author(s)

Claudio Agostinelli

References


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

```r
## 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

- `object` an object of class wle.aic.ar.
- `x` an object of class wle.aic or summary.wle.aic.ar.
- `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 summary.wle.aic.ar) or further arguments passed to or from other methods (in print.wle.aic.ar and print.summary.wle.aic.ar).

Value

summary.wle.aic.ar returns a list:

- `waic` the first num.max best models with their Weighted Akaike Information Criterion.
- `num.max` the number of models reported.
- `call`

Author(s)

Claudio Agostinelli

See Also

- `wle.aic.ar` a function for evaluate the Weighted Akaike Information Criterion for autoregressive models.
Description

All these functions are methods for class wle.aic or summary.wle.aic.

Usage

```r
## 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

- `object` an object of class wle.aic.
- `x` an object of class wle.aic or summary.wle.aic.
- `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 summary.wle.aic) or further arguments passed to or from other methods (in print.wle.aic and print.summary.wle.aic).

Value

`summary.wle.aic` returns a list:

- `waic` the first `num.max` best models with their Weighted Akaike Information Criterion.
- `num.max` the number of models reported.
- `call`

Author(s)

Claudio Agostinelli

See Also

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

x a univariate time series.
order 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.
seasonal a specification of the seasonal part of the ARI model, plus the period (which defaults to frequency(x)).
group the dimension of the bootstrap subsamples.
group.start the dimension of the bootstrap subsamples used in the starting process if wle.init=TRUE.
group.step the dimension of the bootstrap subsamples used in a step, it must be less than group.
xreg optionally, a vector or matrix of external regressors, which must have the same number of rows as x.
include.mean Should the ARI model include a mean term? The default is TRUE for undifferenced series, FALSE for differenced ones (where a mean would not affect the fit nor predictions).
na.action function to be applied to remove missing values.
tol the absolute accuracy to be used to achieve convergence of the algorithm.
tol.step the absolute accuracy to be used to achieve convergence in a step.
equal the absolute value for which two roots are considered the same. (This parameter must be greater than tol).
equal.step the absolute value for which two roots are considered the same in a step. (This parameter must be greater than tol.step).
raf type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCH2": Symmetric Chi-Squared Disparity RAF.
smooth the value of the smoothing parameter.
smooth.ao the value of the smoothing parameter used in the outliers classification, default equal to smooth.
boot the number of starting points based on bootstrap subsamples to use in the search of the roots.
boot.start the number of starting points based on bootstrap subsamples to use in the search of the roots in the starting process.
boot.step the number of starting points based on bootstrap subsamples to use in the search of the roots in a step.
num.sol maximum number of roots to be searched.
num.sol.step maximum number of roots to be searched in a step.
x.init initial values, a vector with the same length of the AR order, or a number, default is 0.
x.seasonal.init initial values, a vector with the same length of the SAR order, or a number, default is 0.
max.iter.out maximum number of iterations in the outer loop.
max.iter.in maximum number of iterations in the inner loop.
max.iter.start maximum number of iterations in the starting process.
max.iter.step maximum number of iterations in a step.
w.level the threshold used to decide if an observation could be an additive outlier.
population.size see details.
population.choose see details.
elements.random see details.
num.max maximum number of observations can be considered as possible additive outliers.
wle.start if TRUE a weighted likelihood estimation is used to have a starting value.
init.values a vector with initial values for the AR and seasonal AR coefficients and the innovations variance.
verbose if TRUE warnings are printed.
min.weights see details.
approx.w 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coef</td>
<td>a vector of AR and regression coefficients.</td>
</tr>
<tr>
<td>sigma2.coef</td>
<td>the estimated variance matrix of the coefficients coef.</td>
</tr>
<tr>
<td>sigma2</td>
<td>the WLE of the innovations variance.</td>
</tr>
<tr>
<td>arma</td>
<td>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.</td>
</tr>
<tr>
<td>resid</td>
<td>the residuals.</td>
</tr>
<tr>
<td>resid.with ao</td>
<td>the residuals with the additive outliers effects.</td>
</tr>
<tr>
<td>resid.without ao</td>
<td>the residuals without the additive outliers effects.</td>
</tr>
<tr>
<td>x.ao</td>
<td>the time series without the additive outliers effects.</td>
</tr>
<tr>
<td>call</td>
<td>the matched call.</td>
</tr>
<tr>
<td>series</td>
<td>the name of the series x.</td>
</tr>
<tr>
<td>weights</td>
<td>the weights.</td>
</tr>
<tr>
<td>weights.with ao</td>
<td>the weights with the additive outliers effects.</td>
</tr>
<tr>
<td>weights.without ao</td>
<td>the weights without the additive outliers effects</td>
</tr>
<tr>
<td>tot.sol</td>
<td>the number of solutions found.</td>
</tr>
<tr>
<td>not.conv</td>
<td>the number of starting points that does not converge after the max.iter.out iteration are reached.</td>
</tr>
<tr>
<td>ao.position</td>
<td>the position of the additive outliers.</td>
</tr>
</tbody>
</table>

Author(s)

Claudio Agostinelli

References

Examples

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

Description

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

Usage

```r
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 boostrap subsamples to use in the search of the roots.
- `group` : the dimension of the bootstrap subsamples. The default value is `max(round(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.
wle.cp

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tot.weights</td>
<td>the sum of the weights divide by the number of observations, one value for each root found.</td>
</tr>
<tr>
<td>weights</td>
<td>the weights associated to each observation, one column vector for each root found.</td>
</tr>
<tr>
<td>f.density</td>
<td>the non-parametric density estimation.</td>
</tr>
<tr>
<td>m.density</td>
<td>the smoothed model.</td>
</tr>
<tr>
<td>delta</td>
<td>the Pearson residuals.</td>
</tr>
<tr>
<td>call</td>
<td>the match.call().</td>
</tr>
<tr>
<td>tot.sol</td>
<td>the number of solutions found.</td>
</tr>
<tr>
<td>not.conv</td>
<td>the number of starting points that does not converge after the max.iter iteration are reached.</td>
</tr>
</tbody>
</table>

**Author(s)**

Claudio Agostinelli

**References**


**Examples**

```r
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

```r
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 wle.cp 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 size is the number of observations and var 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:
  - raf="HD": Hellinger Distance RAF,
  - raf="NED": Negative Exponential Disparity RAF,
  - raf="SCH2": 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.
- **min.weight**: see details.
- **method**: see details.
- **alpha**: penalty value.
- **contrasts**: an optional list. See the contrasts.arg of model.matrix.default.
- **verbose**: if TRUE warnings are printed.

Details

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

\texttt{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 we would choose, we introduce the \texttt{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.

\texttt{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 \textit{Cp} are not as good as the one based on \texttt{method="full"}.

\textbf{Value}

\texttt{wle.cp} returns an object of \texttt{class \"wle.cp\"}. The function \texttt{summary} is used to obtain and print a summary of the results. The generic accessor functions \texttt{coefficients} and \texttt{residuals} extract coefficients and residuals returned by \texttt{wle.cp}. The object returned by \texttt{wle.cp} are:

\begin{itemize}
  \item \texttt{wcp}: Weighted Mallows Cp for each submodels
  \item \texttt{coefficients}: the parameters estimator, one row vector for each root found and each submodel.
  \item \texttt{scale}: an estimation of the error scale, one value for each root found and each submodel.
  \item \texttt{residuals}: the unweighted residuals from the estimated model, one column vector for each root found and each submodel.
  \item \texttt{tot.weights}: the sum of the weights divide by the number of observations, one value for each root found and each submodel.
  \item \texttt{weights}: the weights associated to each observation, one column vector for each root found and each submodel.
  \item \texttt{freq}: the number of starting points converging to the roots.
  \item \texttt{call}: the match.call().
  \item \texttt{contrasts}
  \item \texttt{xlevels}
  \item \texttt{terms}
  \item \texttt{model}
  \item \texttt{x}
  \item \texttt{y}
  \item \texttt{info}
\end{itemize}

\textbf{Author(s)}

Claudio Agostinelli
References


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)
Description

All these functions are methods for class wle.cp or summary.wle.cp.

Usage

```r
## 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

- `object`: an object of class wle.cp.
- `x`: an object of class wle.cp or summary.wle.cp.
- `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 summary.wle.cp) or further arguments passed to or from other methods (in print.wle.cp and print.summary.wle.cp).

Value

summary.wle.cp returns a list:

- `wcp`: the first num.max best models with their Weighted Mallows Cp.
- `num.max`: the number of models reported.
- `call`

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

\[
\text{wle.cv(formula, data=list(), model=TRUE, x=FALSE, y=FALSE, monte.carlo=500, split, boot=30, group, num.sol=1, raf="HD", smooth=0.031, tol=10^{-6}, equal=10^{-3}, max.iter=500, min.weight=0.5, contrasts=NULL, type=c('fast', 'slow'), verbose=FALSE)}
\]

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 \texttt{wle.cv} is called from.
- **model, x, y**: logicals. If TRUE 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 \text{size} is the number of observations and \text{var} 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 \text{size} is the number of observations and \text{var} is the number of variables.
- **num.sol**: maximum number of roots to be searched.
- **raf**: type of Residual adjustment function to be use:
  - \texttt{raf="HD"}: Hellinger Distance RAF,
  - \texttt{raf="NED"}: Negative Exponential Disparity RAF,
  - \texttt{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 \texttt{tol}).
- **max.iter**: maximum number of iterations.
Details

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

\texttt{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 \texttt{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

\texttt{wle.cv} returns an object of class \texttt{"wle.cv"}.

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

\begin{itemize}
  \item \texttt{wcv} \hspace{1cm} Weighted Cross-Validation for each submodels
  \item \texttt{coefficients} \hspace{1cm} the parameters estimator, one row vector for each root found in the full model.
  \item \texttt{scale} \hspace{1cm} an estimation of the error scale, one value for each root found in the full model.
  \item \texttt{residuals} \hspace{1cm} the unweighted residuals from the estimated model, one column vector for each root found in the full model.
  \item \texttt{tot.weights} \hspace{1cm} the sum of the weights divide by the number of observations, one value for each root found in the full model.
  \item \texttt{weights} \hspace{1cm} the weights associated to each observation, one column vector for each root found in the full model.
  \item \texttt{freq} \hspace{1cm} the number of starting points converging to the roots.
  \item \texttt{index} \hspace{1cm} position of the root used for the weights.
  \item \texttt{call} \hspace{1cm} the match.call().
  \item \texttt{contrasts} \hspace{1cm} an optional list. See the \texttt{contrasts.arg} of \texttt{model.matrix.default}.
  \item \texttt{xlevels} \hspace{1cm} see details.
\end{itemize}
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


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(rnorm(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)

---

**Description**

All these functions are methods for class `wle.cv` or `summary.wle.cv`.

**Usage**

```r
## 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$x)$wcv)), ...)

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

**Arguments**

- `object` an object of class `wle.cv`.
- `x` an object of class `wle.cv` or `summary.wle.cv`.
- `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 `summary.wle.cv`) or further arguments passed to or from other methods (in `print.wle.cv` and `print.summary.wle.cv`).

**Value**

- `summary.wle.cv` returns a list:
  - `wcv` the first `num.max` best models with their estimated prediction error using WCV.
  - `num.max` the number of models reported.
  - `call`

**Author(s)**

Claudio Agostinelli
See Also

wle.cv a function for evaluate the Weighted Cross Validation criterion in the linear models.

wle.fractiondiff

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.fractiondiff, the remain functions are for internal use and they should not call by the users. They are not documented here.

Usage

wle.fractiondiff(x, lower, upper, M, group, na.action=na.fail, lower=10^(-6), equal=10^(-3), raf="HD", smooth=0.0031, smooth.ao=smooth, boot=10, num.sol=1, x.init=rep(0,M), use.unirout=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

x a univariate time series.
lower the lower end point of the interval to be searched.
upper the upper end point of the interval to be searched.
M the order of the finite memory process used to estimate the d parameter.
group the dimension of the bootstrap subsamples.
na.action function to be applied to remove missing values.
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).
raf type of Residual adjustment function to be use: raf="HD": Hellinger Distance RAF, raf="NED": Negative Exponential Disparity RAF, raf="SCH2": Symmetric Chi-Squared Disparity RAF.
smooth the value of the smoothing parameter.
smooth.ao the value of the smoothing parameter used in the outliers classification, default equal to smooth.
boot the number of starting points based on bootstrap subsamples to use in the search of the roots.
num.sol maximum number of roots to be searched.
x.init initial values, a vector with the same length of the M parameter, or a number, default is 0.
use.unirroot 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.
max.iter.out maximum number of iterations in the outer loop.
max.iter.in maximum number of iterations in the inner loop.
max.iter.step maximum number of iterations in a step.
max.iter.start maximum number of iterations in the starting process.
verbose if TRUE warnings are printed.
w.level the threshold used to decide if an observation could be an additive outlier.
ininit.values a vector with initial values for the d and the innovations variance.
num.max maximum number of observations can be considered as possible additive outliers.
include.mean Should the model include a mean term? The default is TRUE.
ao.list possible list of pattern of additive outliers.
min.weights see details.
size.population see details.
size.generation see details.
prob.crossover see details.
prob.mutation see details.
type.scale see details.
type.selection see details.
elitist see details.
scale.c 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

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

Author(s)

Claudio Agostinelli

References


Examples

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
## Not run:
x <- Nile
x[50] <- x[50]+4*sd(x)

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

resw$sigma2
resw$x.mean
resw$s0.position

## End(Not run)

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(round(size/4), var)
  where size is the number of observations and var 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 for the main function.

shape.int
  a 2 dimension vector for the interval search of the shape parameter.

use.smooth
  if FALSE the unsmoothed model is used. This is usefull when the integration routine does not work well.

tol.int
  the absolute accuracy to be used in the integration routine. The default value is tol * 10^{-4}.

verbose
  if TRUE warnings are printed.

maxiter
  maximum number of iterations. This value is passed to uniroot function.
Details

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

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

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

The function use `uniroot` to solve the estimating equation for $\text{shape}$, 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 bandwith is $smooth \times \text{shape}/\text{scale}^2$.

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:

- `shape`: the estimator of the shape parameter, one value for each root found.
- `scale`: the estimator of the scale parameter, one value for each root found.
- `rate`: the estimator of the rate parameter (1/scale), 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


wle.glm

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.glm

Robust Fitting Generalized Linear Models using Weighted Likelihood

Description

wle.glm is used to robustly fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Usage

wle.glm(formula, family = binomial, data, weights, subset, na.action, start = NULL, etastart, mustart, offset, control = list(glm = glm.control(...), wle = wle.glm.control()), model = TRUE, method = "wle.glm.fit", x = FALSE, y = TRUE, contrasts = NULL, dist.method = "euclidean", ...)
wle.glm.fit(x, y, weights = NULL, wle.weights = rep(1, NROW(y)), start = NULL, etastart = NULL, mustart = NULL, offset = rep(0, NROW(y)), family = gaussian(), control = list(glm=glm.control(), wle=wle.glm.control()), dist.method='euclidean', intercept = TRUE, dispersion = NULL)

## S3 method for class 'wle.glm'
weights(object, type = c("prior", "working", "wle"), root="all", ...)

Arguments

formula an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under ‘Details’.

family a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
data: an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `wle.glm` is called.

weights: an optional vector of ‘prior weights’ to be used in the fitting process. Should be `NULL` or a numeric vector.

subset: an optional vector specifying a subset of observations to be used in the fitting process.

na.action: a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The ‘factory-fresh’ default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.

start: starting values for the parameters in the linear predictor.

etastart: starting values for the linear predictor.

mustart: starting values for the vector of means.

offset: this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.

control: a list with two components of parameters for controlling the fitting process. The first component (`glm`) is set using the function `glm.control` while the second component (`wle`)) is set using the function `wle.glm.control` and it is used to set the parameters regarding the behaviour of the robust method. See the documentation of these functions for details.

model: a logical value indicating whether `model.frame` should be included as a component of the returned value.

method: the method to be used in fitting the model. The default method "wle.glm.fit" uses iteratively reweighted least squares (IWLS). The only current alternative is "model.frame" which returns the model frame and does no fitting.

x, y: For `wle.glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.

For `wle.glm.fit`: x is a design matrix of dimension n * p, and y is a vector of observations of length n.

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

dist.method: distance method passed to `dist` to measure the distance between predictor rows.

intercept: logical. Should an intercept be included in the null model?

dispersion: numeric or `NULL`. If provided used as starting value.

object: an object inheriting from class "wle.glm".

type: character, partial matching allowed. Type of weights to extract from the fitted model object.

root: character ("all") or a number. For which solutions the weights are reported.
**wle.glm**

**wle.weights** For *wle.glm.fit* these are weights used in the iterative algorithm evaluated at each step by *wle.glm.weights*.

... For *wle.glm* arguments to be passed by default to *glm.control*: see argument control.

For weights: further arguments passed to or from other methods.

**Details**

A typical predictor has the form \( \text{response} \sim \text{terms} \) where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. For binomial and quasibinomial families the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures. A terms specification of the form \( \text{first} + \text{second} \) indicates all the terms in first together with all the terms in second with any duplicates removed.

A specification of the form \( \text{first}:\text{second} \) indicates the the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification \( \text{first}*\text{second} \) indicates the cross of first and second. This is the same as \( \text{first} + \text{second} + \text{first}:\text{second} \).

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers \( w_i \), that each response \( y_i \) is the mean of \( w_i \) unit-weight observations. In case of binomial GLM prior weights CAN NOT be used to give the number of trials when the response is the proportion of successes; in this situation please submit the response variable as two columns (first column successes, second column unsuccesses). They would rarely be used for a Poisson GLM.

*wle.glm.fit* is the workhorse function: it is not normally called directly but can be more efficient where the response vector and design matrix have already been calculated. However, this function needs starting values and does not look for possible multiple roots in the system of equations.

If more than one of etastart, start and mustart is specified, the first in the list will be used. It is often advisable to supply starting values for a quasi family, and also for families with unusual links such as gaussian("log").

All of weights, subset, offset, etastart and mustart are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula.

For the background to warning messages about ‘fitted probabilities numerically 0 or 1 occurred’ for binomial GLMs, see Venables & Ripley (2002, pp. 197–8).

Multiple roots may occur if the asymptotic weights are used or in the case of continuous models. The function implements the bootstrap root serach approach described in Markatou, Basu and Lindsay (1998) in order to find these roots.

**Value**

*wle.glm* returns an object of class inheriting from "wle.glm".

The function summary (i.e., summary.wle.glm) can be used to obtain or print a summary of the results and the function anova (i.e., anova.wle.glm.root) to produce an analysis of variance table.
The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` can be used to extract various useful features of the value returned by `wle.glm`.

`weights` extracts a vector of weights, one for each case/root in the fit (after subsetting and `na.action`). An object of class "wle.glm" is a (variable length) list containing at least the following components:

- `root1` which is a list with the following components:
  - `coefficients` a named vector of coefficients
  - `residuals` the working residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are NA.
  - `fitted.values` the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
  - `rank` the numeric rank of the fitted linear model.
  - `family` the family object used.
  - `linear.predictors` the linear fit on link scale.
  - `deviance` up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
  - `aic` Akaike’s An Information Criterion, minus twice the maximized log-likelihood plus twice the number of coefficients (so assuming that the dispersion is known).
  - `null.deviance` The deviance for the null model, comparable with `deviance`. The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
  - `iter` the number of iterations of IWLS used.
  - `weights` the working weights, that is the weights in the final iteration of the IWLS fit.
  - `prior.weights` the weights initially supplied, a vector of 1s if none were.
  - `df.residual` the residual degrees of freedom.
  - `df.null` the residual degrees of freedom for the null model.
  - `y` if requested (the default) the y vector used. (It is a vector even for a binomial model.)
  - `x` if requested, the model matrix.
  - `model` if requested (the default), the model frame.
  - `converged` logical. Was the IWLS algorithm judged to have converged?
  - `boundary` logical. Is the fitted value on the boundary of the attainable values?
  - `wle.weights` final (robust) weights based on the WLE approach.
  - `wle.asymptotic` logicals. If TRUE asymptotic weight based on Anscombe residual is used for the corresponding observation.

In addition, non-empty fits will have components `qr`, `R`, `qraux`, `pivot` and `effects` relating to the final weighted linear fit.

and the following components:
family the family object used.
call the matched call.
formula the formula supplied.
terms the terms object used.
data the data argument.
offset the offset vector used.
control the value of the control argument used.
method the name of the fitter function used, currently always "wle.glm.fit".
contrasts (where relevant) the contrasts used.
xlevels (where relevant) a record of the levels of the factors used in fitting.
tot.sol the number of solutions found.
not.conv the number of starting points that does not converge after the max.iter (defined using wle.glm.control) iterations are reached.
na.action (where relevant) information returned by model.frame on the special handling of NAs.

Objects of class "wle.glm" are normally of class "wle.glm".

If a binomial wle.glm model was specified by giving a two-column response, the weights returned by prior.weights are the total numbers of cases (factored by the supplied case weights) and the component y of the result is the proportion of successes.

In case of multiple roots (i.e. tot.sol > 1) then objects of the same form as root1 are reported with names root2, root3 and so on until tot.sol.

Warnings

Since in a model selection procedure and/or on an ANOVA table the weights of the WLE procedure must be that of the FULL model (and not that of the actual model) statistics on degrees of freedom, deviance and AIC are valid only if this is the FULL model.

Author(s)

Claudio Agostinelli and Fatemah Al-quallaf

References


See Also

`anova.wle.glm`, `summary.wle.glm`, etc. for `wle.glm` methods, and the generic functions `anova`, `summary`, `effects`, `fitted.values`, and `residuals`.

`wle.lm` for robust non-generalized linear models for ‘general’ linear models.

Examples

```r
## Dobson (1990) Page 93: Randomized Controlled Trial:
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
wle.glm.D93 <- wle.glm(counts ~ outcome + treatment, family=poisson(), x=TRUE, y=TRUE)
wle.glm.D93$extractRoot(wle.glm.D93)
summary(wle.glm.D93)

## Not run:
## Support for gaussian family not provided yet!
## an example with offsets from Venables & Ripley (2002, p.189)
utils::data(anorexia, package="MASS")

anorex.2 <- wle.glm(Postwt ~ Prewt + Treat + offset(Prewt),
                     family = gaussian, data = anorexia)

anorex.2
summary(anorex.2)

## End(Not run)

## Not run:
# Gamma family is not yet implemented!
# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))
wlot1 <- wle.glm(lot1 ~ log(u), data=clotting, family=Gamma,
                control=list(glm=glm.control(), wle=wle.glm.control(use.asymptotic=1)))
wlot2 <- wle.glm(lot2 ~ log(u), data=clotting, family=Gamma,
                control=list(glm=glm.control(), wle=wle.glm.control(use.asymptotic=1)))
wlot1
wlot2
summary(wlot1)
```
**wle.glm.control**

Auxiliary for Controlling GLM Robust Fitting

**Description**

Auxiliary function as user interface for glm robust fitting. Typically only used when calling `wle.glm` or `wle.glm.fit`.

**Usage**

```r
wle.glm.control(boot = 30, group = NULL, num.sol = 1,
rafi = c("GKL", "PWD", "HD", "NED", "SCHI2"), tau = 0.1,
cutpoint = 0, powerdown = 1, delta = NULL, smooth = NULL,
asy.smooth = 0.031, tol = 10^(-6), equal = 10^(-3),
max.iter = 500, window.size = NULL, use.asymptotic = NULL,
use.smooth = TRUE, mle.dispersion = FALSE, verbose = FALSE)
```

**Arguments**

- `boot` integer. Number of starting points based on bootstrap subsamples to use in the search of the roots.
- `group` integer. Dimension of the bootstrap subsamples. The default value is $\text{max}(\text{round}(\text{size}/2), \text{var} + 1)$ where `size` is the number of observations and `var` is the number of predictors.
- `num.sol` integer. Maximum number of roots to be searched.
- `rafi` type of Residual adjustment function to be used:
  - raf="GKL": Generalized Kullback-Leibler family RAF (see details),
  - raf="PWD": Power Divergence family RAF (see details),
  - raf="HD": Hellinger Distance RAF,
  - raf="NED": Negative Exponential Disparity RAF,
  - raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
- `tau` positive real. Used in selecting the member of the RAF family in the case of GKL or PWD.
- `cutpoint` a value in the interval [0,1].
- `powerdown` a non negative number.
- `delta` between (0,1). Used in the construction of the weights for the Binomial family.
- `smooth` the value of the smoothing parameter; used in the evaluation of weights in the case of continuous models.
- `asy.smooth` the value of the smoothing parameter; used in the evaluation of asymptotic weights. or in the case of continuous models.
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. Two roots are compared using the corresponding final weights.
max.iter maximum number of iterations.
window.size positive real or NULL. The observations with a distance, in the predictors space, less than this threshold are used to estimate the conditional distribution for a given level of the predictor.
use.asymptotic integer or NULL. The minimum number of observations for the level of the predictors under which asymptotic weights are used.
use.smooth if TRUE the smoothed model is used in the computation of the Pearson Residuals. For now, the option is used only for the Gamma family.
mle.dispersion if TRUE the weighted likelihood estimator for dispersion is used otherwise the weighted chi-squared statistics is used.
verbose if TRUE warnings are printed.

Details
The Generalized Kullback-Leibler family RAF is defined as:
\[
\ln(tau \ast x + 1)/tau
\]
for \(tau > 0\).
The Power Divergence family RAF is defined as:
\[
tau \ast ((x + 1)({1/tau}) - 1)
\]
for \(0 < tau < Inf\) while
\[
\ln(x + 1)
\]
for \(tau = Inf\).

Value
A list with the arguments as components.

Author(s)
Claudio Agostinelli and Fatemah Alqallaf

References

See Also
wle.glm
Examples

### A variation on example(wle.glm):

```r
## Annette Dobson's example ...
counts <- c(18,17,15,20,10,20,12,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
oo <- options(digits = 12) # to see more when tracing:
wle.glm.D93X <- wle.glm(counts ~ outcome + treatment, family=poisson(),
                           control=list(glm=wle.glm.control(trace = TRUE),
                           wle=wle.glm.control(raf='GKL', tau=0.15))
options(o)
coef(wle.glm.D93X)
```

### `wle.glm.summaries`

**Accessing Generalized Linear Model Robust Fits**

**Description**

These functions are all methods for class `glm` or `summary.glm` objects.

**Usage**

```r
## S3 method for class 'wle.glm'
family(object, ...)
```

```r
## S3 method for class 'wle.glm'
residuals(object, type = c("deviance", "pearson", "working", "response", "partial"), root="all", ...)```

**Arguments**

- `object`: an object of class `glm`, typically the result of a call to `glm`.
- `type`: the type of residuals which should be returned. The alternatives are: "deviance" (default), "pearson", "working", "response", and "partial".
- `root`: vector of integer or characters. If "all" the residuals of all roots are reported, otherwise the position of the root should be supplied.
- `...`: further arguments passed to or from other methods.

**Details**

The references define the types of residuals: Davison & Snell is a good reference for the usages of each.

The partial residuals are a matrix of working residuals, with each column formed by omitting a term from the model.
How residuals treats cases with missing values in the original fit is determined by the `na.action` argument of that fit. If `na.action = na.omit` omitted cases will not appear in the residuals, whereas if `na.action = na.exclude` they will appear, with residual value NA. See also `naresid`.

For fits done with `y = FALSE` the response values are computed from other components.

**See Also**

`wle.glm`, `anova.wle.glm.root`; the corresponding *generic* functions, `summary.wle.glm`, `coef`, `deviance`, `df.residual`, `effects`, `fitted`, `residuals`.

---

**wle.glm.weights**  
*Weights based on Weighted Likelihood for the GLM model*

**Description**

Evaluate the weights for a given GLM model

**Usage**

```r
wle.glm.weights(y, x, fitted.values, family = gaussian(),
    dispersion = 1, raf = "GKL", tau = 0.1, smooth = NULL,
    asy.smooth=0.031, window.size = NULL, use.asymptotic = NULL,
    use.smooth=TRUE, tol=10^(-6), dist.method = "euclidean",
    cutpoint = 0, powerdown = 1)
```

**Arguments**

- `y`: `y` is a vector of observations of length `n`.
- `x`: `x` is a design matrix of dimension `n * p`.
- `fitted.values`: the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function. Often obtain as a result of `wle.glm.fit` call.
- `family`: a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)
- `dispersion`: value of the dispersion parameter. Used only in the Gamma family for now.
- `raf`: type of Residual adjustment function to be used:
  - `raf = "GKL"`: Generalized Kullback-Leibler family RAF (see details),
  - `raf = "PWD"`: Power Divergence family RAF (see details),
  - `raf = "HD"`: Hellinger Distance RAF,
  - `raf = "NED"`: Negative Exponential Disparity RAF,
  - `raf = "SCH12"`: Symmetric Chi-Squared Disparity RAF.
- `tau`: positive real. Used in selecting the member of the RAF family in the case of GKL or PWD.
- `smooth`: the value of the smoothing parameter; used in the case of continuous models.
asy.smooth 
the value of the smoothing parameter; used in the evaluation of asymptotic weights.

window.size 
positive real or NULL. The observations with a distance, in the predictors space, less than this threshold are used to estimate the conditional distribution for a given level of the predictor.

use.asymptotic 
integer or NULL. The minimum number of observations for the level of the predictors under which asymptotic weights are used.

use.smooth 
if TRUE the smoothed model is used in the computation of the Pearson Residuals. For now, the option is used only for the Gamma family.

tol 
the tolerance used in the numerical calculations. For now, the option is used only for the Gamma family.

dist.method 
distance method passed to dist to measure the distance between predictor rows.

cutpoint 
a value in the interval [0,1].

powerdown 
a non negative number.

Value
A list with two components

weights 
the weights associated to the observations.

asy 
logical. If TRUE the corresponding weights is evaluated using asymptotic considerations based on Anscombe residuals.

Author(s)
Claudio Agostinelli and Fatemah Al-quallaf

References

See Also
wle.glm

Examples

# tau=0.1
wgr.D93 <- extractRoot(wle.glm.D93)
# tau=0.2
wlwgr.D93 <- wle.glm.weights(y = wgr.D93$y, x = wgr.D93$x,
fitted.values = wgr.D93$fitted.values, family = wgr.D93$family,
raf = "GKL", tau = 0.2, smooth = 0.031, window.size = NULL,
use.asymptotic = NULL, dist.method = "euclidean")
# tau=0.3
w2wgr.D93 <- wle.glm.weights(y = wgr.D93$y, x = wgr.D93$x,
**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**

```r
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.)
- **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(round(size/4), var) where size is the number of observations and var is the number of variables.
- **num.sol**: maximum number of roots to be searched.
- **raf**: type of Residual adjustment function to be used: 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.

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

verbose 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:

coefficients the parameters estimator, one row vector for each root found.

standard.error an estimation of the standard error of the parameters estimator, one row vector for each root found.

scale an estimation of the error scale, one value for each root found.

residuals the unweighted residuals from the estimated model, one column vector for each root found.

fitted.values 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 max.iter iterations are reached.

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

See Also
wle.smooth an algorithm to choose the smoothing parameter for normal distribution and normal kernel.

Examples

```r
library(wle)
# You can find this data set in:
# 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)
```
Description

All these functions are methods for class wle.lm or summary.wle.lm.

Usage

```r
## 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

- `object` an object of class wle.lm.
- `x` an object of class wle.lm or summary.wle.lm.
- `formula` a model formula
- `data` data.frame, list, environment or object coercible to data.frame containing the variables in formula.
- `na.action` how NAs are treated. The default is first, any na.action attribute of data, second a na.action setting of options, and third na.fail if that is unset. The “factory-fresh” default is na.omit.
- `root` the root to be printed, in summary.wle.lm it could be "ALL", all the roots are printed, or a vector of integers.
digits

number of digits to be used for most numbers.

signif.stars

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 show.signif.stars slot of options.

... 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

residuals

the weighted residuals, the usual residuals rescaled by the square root of the weights given by wle.lm.

coefficients

a $p \times 4$ matrix with columns for the estimated coefficient, its standard error, weighted-t-statistic and corresponding (two-sided) p-value.

sigma

the square root of the estimated variance of the random error.

df

degrees of freedom, a 3-vector $(p, \sum \text{weights} - p, p)$. 

fstatistic

a 3-vector with the value of the weighted-F-statistic with its numerator and denominator degrees of freedom.

r.squared

$R^2$, the "fraction of variance explained by the model".

adj.r.squared

the above $R^2$ statistic "adjusted", penalizing for higher $p$.

root

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:
# Location of several outliers in multiple regression data using
# elemental sets. Technometrics, 26, 197-208.
```r
# data(artificial)
result <- wle.nlm(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")
```

---

**Description**

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

**Usage**

```r
wle.negativebinomial(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 containing the number of failures which occur in a sequence of Bernoulli trials before a target number of successes `size` is reached.
- **size**: target number of successes.
- **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(round(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.negativebinomial` returns an object of class "`wle.negativebinomial`".

Only print method is implemented for this class.

The object returned by `wle.negativebinomial` 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


Examples

```r
library(wle)

set.seed(1234)

x <- rnbinom(20, size=10, prob=0.2)
wle.negativebinomial(x, size=10)

x <- c(rnbinom(20, size=10, prob=0.2),rnbinom(10, size=10, p=0.9))
result <- wle.negativebinomial(x, size=10)
print(result)
plot(result$weights)
```
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

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 $\max(\text{round}(\text{size}/4), 2)$ where $\text{size}$ 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 tol).
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 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

See Also
wle.smooth an algorithm to choose the smoothing parameter for normal distribution and normal kernel.

Examples
library(wle)
data(cavendish)
result <- wle.normal(cavendish)
result
test <- wle.normal(cavendish,boot=20,num.sol=1)
barplot(result$weights,col=2,xlab="Observations", ylab="Weights",ylab=c(0,1), names.arg=seq(1:length(result$weights)))
**Description**

`wle.normal.mixture` is a preliminary version; it is used to robust estimate the location, scale and proportion parameters via Weighted Likelihood, when the sample is iid from a normal mixture univariate distribution with known m number of components.

**Usage**

```r
wle.normal.mixture(x, m, boot=5, group, num.sol=1, raf="HD", smooth=0.003, tol=10^(-15), equal=10^(-2), max.iter=1000, all.comp=TRUE, min.size=0.02, min.weights=0.3, boot.start=10, group.start=3, tol.start=10^(-6), equal.start=10^(-3), smooth.start=0.003, max.iter.start=500, max.iter.boot=25, verbose=FALSE)
```

```r
wle.normal.mixture.start(x, m, boot=5, group, raf="HD", smooth=0.003, tol=10^(-15), equal=10^(-2), min.size=0.02, min.weights=0.3, boot.start=20, group.start=3, max.iter.start=500, max.iter.boot=20, verbose=FALSE)
```

**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 size 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 tol).
- **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 \( \text{max}(\text{round}(\text{group}/4), 2) \).
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 tol.start).
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 \texttt{wle.normal} we try to find the biggest components, then we discard each observation with weight greater than min.weights. The \texttt{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

\texttt{wle.normal.mixture} returns an object of class \texttt"wle.normal.mixture". Only print method is implemented for this class.
The objects returned by \texttt{wle.normal.mixture} are:

\begin{itemize}
\item \texttt{location} the estimator of the location parameters, one vector for each root found.
\item \texttt{scale} the estimator of the scale parameters, one vector for each root found.
\item \texttt{pi} the estimator of the proportion parameters, one vector for each root found.
\item \texttt{tot.weights} the sum of the weights, divide by the number of observations, one value for each root found.
\item \texttt{weights} the weights associated to each observation, one column vector for each root found.
\item \texttt{f.density} the non-parametric density estimation.
\item \texttt{m.density} the smoothed model.
\item \texttt{delta} the Pearson residuals.
\item \texttt{freq} the number of starting points converging to the roots.
\item \texttt{tot.sol} the number of solutions found.
\end{itemize}
\textit{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.

**Arguments**

- \texttt{x} a matrix contain the observations.
- \texttt{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 \( \text{size} \) is the number of observations and \( \text{var} \) is the number of variables.

num.sol maximum number of roots to be searched.

raf type of Residual adjustment function to be use:

- \( \text{raf="HD"} \): Hellinger Distance RAF,
- \( \text{raf="NED"} \): Negative Exponential Disparity RAF,
- \( \text{raf="SCHI2"} \): 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 \( \text{tol} \)).

max.iter maximum number of iterations.

verbose if TRUE warnings are printed.

Value

\( \text{wle.normal.multi} \) returns an object of \textbf{class} "\text{wle.normal.multi}".

Only print method is implemented for this class.

The object returned by \( \text{wle.normal.multi} \) are:

- \textbf{location} the estimator of the location parameters, one vector for each root found.
- \textbf{variance} the estimator of the covariance matrix, one matrix for each root found.
- \textbf{tot.weights} the sum of the weights divide by the number of observations, one value for each root found.
- \textbf{weights} the weights associated to each observation, one column vector for each root found.
- \textbf{f.density} the non-parametric density estimation.
- \textbf{m.density} the smoothed model.
- \textbf{delta} the Pearson residuals.
- \textbf{freq} the number of starting points converging to the roots.
- \textbf{tot.sol} the number of solutions found.

\textbf{call} the \texttt{match.call()}.

\textbf{not.conv} the number of starting points that does not converge after the \texttt{max.iter} iteration are reached.

Author(s)

Claudio Agostinelli
References


See Also

**wle.smooth** an algorithm to choose the smoothing parameter for normal distribution and normal kernel.

Examples

```r
library(wle)

data(iris)

smooth <- wle.smooth(dimension=4, constant=4, weight=0.5, interval=c(0.3, 0.7))

x.data <- as.matrix(iris[,5]=="virginica", 1:4])

result <- wle.normal.multi(x.data, boot=20, group=21,
num.sol=3, smooth=smooth$root)

result

result <- wle.normal.multi(x.data, boot=20, group=21,
num.sol=1, smooth=smooth$root)

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

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

- **x**: an object of class `wle.normal.multi`.
- **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.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

```r
## 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.
A One-Step Weighted Likelihood Estimator for Linear model

Description

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

Usage

```r
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 in 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

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,
wle.onestep.summaries

# 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

x an object of class wle.onestep.
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.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

\texttt{wle.poisson(x, boot=30, group, num.sol=1,}
\texttt{raf=c("HD", "NED", "GKL", "PWD", "SCHI2"),}
\texttt{tau=NULL, tol=10^(-6), equal=10^(-3),}
\texttt{max.iter=500, verbose=FALSE)}

Arguments

- **x**: a vector containing 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 \( \text{max}(\text{round}(\text{length}(x)/4), 2) \).
- **num.sol**: maximum number of roots to be searched.
- **raf**: type of Residual adjustment function to be used:
  - raf="HD": Hellinger Distance RAF,
  - raf="NED": Negative Exponential Disparity RAF,
  - raf="GKL": Generalized Kullback-Leibler RAF family with parameter \( \tau \).
  - raf="PWD": Power Divergence Measure RAF family with parameter \( \tau \).
  - raf="SCHI2": Symmetric Chi-Squared Disparity RAF.
- **tau**: this is to set the member inside the GKL and PWD family. It must be in [0,1] for GKL and in [-1, Inf] for PWD.
- **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 \( \text{tol} \)).
- **max.iter**: maximum number of iterations.
- **verbose**: if \texttt{TRUE} warnings are printed.

Value

\texttt{wle.poisson} returns an object of \texttt{class} "wle.poisson".

Only print method is implemented for this class.

The object returned by \texttt{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 do not converge after the \texttt{max.iter} iteration are reached.
**wle.smooth**

**Author(s)**
Claudio Agostinelli

**References**


**Examples**
```r
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**
```r
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**
- `weight` weights associated to an observation that is costant scale units away from the mean of the distribution.
- `costant` times the contaminated point mass is away from the mean of the distribution in scale units.
- `level` mass of the contaminated point.
- `dimension` dimension of the normal distribution.
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.
interval  interval from which to search the root.
tol  the absolute accuracy to be used to achieve convergence of the algorithm.
max.iter  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 interavl=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

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**

```r
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**

- `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.)
- `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(round(size/4), var)` where `size` is the number of observations and `var` 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.
- `min.weight`: see details.
- `type`: `type="Stepwise"`: the weighted stepwise methods is used, `type="Forward"`: the weighted forward methods is used, `type="Backward"`: the weighted backward method is used.
f.in the in value
f.out the out value
method method="WLS": the submodel parameters are estimated by weighted least square with weights from the weighted likelihood estimator on the full model.
method="WLE": the submodel parameters are estimated by weighted likelihood estimators.
contrasts an optional list. See the contrasts.arg of model.matrix.default.
verbose if TRUE warnings are printed.

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

\texttt{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 we would choose, we introduce the \texttt{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
\texttt{wle.stepwise} returns an object of \texttt{class} "wle.stepwise".
The function \texttt{summary} is used to obtain and print a summary of the results. The generic accessor functions \texttt{coefficients} and \texttt{residuals} extract coefficients and residuals returned by \texttt{wle.stepwise}.
The object returned by \texttt{wle.stepwise} are:
\begin{itemize}
  \item \texttt{wstep} the iterations with the model selected.
  \item \texttt{coefficients} the parameters estimator, one row vector for each root found in the full model.
  \item \texttt{scale} an estimation of the error scale, one value for each root found in the full model.
  \item \texttt{residuals} the unweighted residuals from the estimated model, one column vector for each root found in the full model.
  \item \texttt{tot.weights} the sum of the weights divide by the number of observations, one value for each root found in the full model.
  \item \texttt{weights} the weights associated to each observation, one column vector for each root found in the full model.
  \item \texttt{freq} the number of starting points converging to the roots.
  \item \texttt{index} position of the root used for the weights.
\end{itemize}
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


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

- object: an object of class wle.stepwise.
- x: an object of class wle.stepwise or summary.wle.stepwise.
- 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 summary.wle.stepwise) or further arguments passed to or from other methods (in print.wle.stepwise and print.summary.wle.stepwise).

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.
wle.t.test

Value

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

- `wstep` 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

---

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

- `x` a numeric vector of data values.
- `y` an optional numeric vector data values.
- `alternative` character specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
- `mu` a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
- `paired` a logical indicating whether you want a paired weighted t-test.
- `var.equal` a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch approximation to the degrees of freedom is used.
- `conf.level` confidence level of the interval.
- `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 \( \text{max}(\text{round}(\text{size}/4), 2) \) where \( \text{size} \) is the number of observations.

num.sol: maximum number of roots to be searched.

raf: type of Residual adjustment function to be use:
- \text{raf}="\text{HD}": Hellinger Distance RAF,
- \text{raf}="\text{NED}": Negative Exponential Disparity RAF,
- \text{raf}="\text{SCH}\_2": 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 \( \text{tol} \)).

max.iter: maximum number of iterations.

Details:
If \text{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 \text{paired} is TRUE). If \text{var.equal} is TRUE then the pooled estimate of the variance is used. By default, if \text{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:

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


Examples

```r
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)
```
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

Examples

set.seed(2345)
x <- rnorm(50,0,1)
y <- rnorm(50,10,1)
res.x <- wle.norm(x,group=5)
res.y <- wle.norm(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.norm(x,group=5,num.sol=2)
res.y <- wle.norm(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**

```r
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())
```

**Arguments**

- `x`: a vector. The object is coerced to class `circular`.
- `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 `tol`).
- `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.
- `min.kappa`: minimum value for the concentration parameter.
- `use.smooth`: logical, if TRUE a smoothed model is used, default is TRUE.
alpha if not NULL overrides the value of p. This is a different parameterization, alpha=-1/2 provides Hellinger Distance RAF, alpha=-1 provides Kullback-Leibler RAF and alpha=-2 provides Neyman's Chi-Square RAF.

p this parameter works only when raf="H0". p=2 provides Hellinger Distance RAF, p=-1 provides Kullback-Leibler RAF and p=1/2 provides Neyman's Chi-Square RAF.

verbose logical, if TRUE warnings are printed.

control.circular the attribute of the resulting object (mu)

digits integer indicating the precision to be used.

... further parameters in print.wle.vonmises.

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:

call the match.call().

mu the estimate of the mean direction or the value supplied. If num.sol > 1 then mu may have length greater than 1, i.e, one value for each root found.

kappa the estimate of the concentration parameter or the value supplied. If num.sol > 1 then kappa may have length greater than 1, i.e, 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.

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

See Also

circular, mle.vonmises.

Examples

```r
x <- c(rvonmises(n=50, mu=circular(0), kappa=10), rvonmises(n=5, mu=circular(pi/2), kappa=20))
wle.vonmises(x, smooth=20, group=5)
```

### 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

```r
wle.weights(x, y=NULL, smooth=0.0031, sigma2, raf=1,
             location=FALSE, max.iter=1000, tol=10^(-6))
```

### 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="SCH2"`: 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 achived.

### Author(s)

Claudio Agostinelli
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 circular.
mu if a values if provided the parameter is considered known.
rho if a values if provided the parameter (and sd) is considered known.
sd if a values if provided the parameter (and rho) 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:
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).
min.sd minimum value for the sd parameter.
min.k minimum number of elements used to approximate the density of the wrapped normal.
max.iter maximum number of iterations.
use.smooth logical, if TRUE a smoothed model is used, default is TRUE.
alpha

if not NULL overrides the value of p. This is a different parameterization, alpha=-1/2 provides Hellinger Distance RAF, alpha=-1 provides Kullback-Leibler RAF and alpha=-2 provides Neyman’s Chi-Square RAF.

p

this parameter works only when raf="HD". p=2 provide Hellinger Distance RAF, p=-1 provide Kullback-Leibler RAF and p=Inf provide Neyman’s Chi-Square RAF.

verbose

logical, if TRUE warnings are printed.

control.circular

the attribute of the resulting objects (mu)

digits

integer indicating the precision to be used.

... further parameters in print.wle.vonmises.

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:

call

the match.call().

mu

the estimate of the mean direction or the value supplied. If num.sol > 1 then mu may have length greater than 1, i.e., one value for each root found.

rho

the estimate of the concentration parameter or the value supplied. If num.sol > 1 then rho may have length greater than 1, i.e., one value for each root found.

sd

the estimate of the standard deviation parameter or the value supplied. If num.sol > 1 then sd may have length greater than 1, i.e., 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.

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


See Also

circular, mle.wrappednormal.

Examples

```r
x <- c(rwrappednormal(n=50, mu=circular(0), sd=1), rwrappednormal(n=5, mu=circular(pi/2), sd=0.5))
wle.wrappednormal(x, smooth=1/20, group=5)
```
Index

*Topic **arith**
  binary, 5

*Topic **datasets**
  artificial, 4
cavendish, 6
hald, 9
rocky, 31
selection, 32

*Topic **htest**
wle.t.test, 97
wle.var.test, 100

*Topic **models**
anova.wle.glm.root, 3
summary.wle.glm, 32
wle.binomial, 46
wle.gamma, 59
wle.glm, 61
wle.glm.control, 67
wle.glm.summaries, 69
wle.negativebinomial, 77
wle.normal, 79
wle.normal.mixture, 81
wle.normal.multi, 83
wle.normal.multi.summaries, 85
wle.normal.summaries, 86
wle.poisson, 89

*Topic **multivariate**
wle.normal.multi, 83
wle.normal.multi.summaries, 85

*Topic **regression**
anova.wle.glm.root, 3
mle.aic, 14
mle.aic.summaries, 16
mle.cp, 17
mle.cp.summaries, 18
mle.cv, 20
mle.cv.summaries, 21
mle.stepwise, 22
mle.stepwise.summaries, 24
plot.mle.cp, 25
plot.wle.cp, 27
plot.wle.lm, 28
residualsAncombe, 30
summary.wle.glm, 32
wle.aic, 34
wle.aic.summaries, 42
wle.cp, 47
wle.cp.summaries, 51
wle.cv, 52
wle.cv.summaries, 55
wle.glm, 61
wle.glm.summaries, 69
wle.lm, 72
wle.lm.summaries, 75
wle.onestep, 87
wle.onestep.summaries, 89
wle.stepwise, 93
wle.stepwise.summaries, 96

*Topic **robust**
anova.wle.glm.root, 3
mde.vonmises, 9
mde.wrappednormal, 11
plot.wle.cp, 27
plot.wle.lm, 28
summary.wle.glm, 32
wle.aic, 34
wle.aic.ar, 37
wle.aic.ar.summaries, 41
wle.aic.summaries, 42
wle.ar, 43
wle.binomial, 46
wle.cp, 47
wle.cp.summaries, 51
wle.cv, 52
wle.cv.summaries, 55
wle.fracdiff, 56
wle.gamma, 59
wle.glm, 61
INDEX

wle.glm.weights, 70
wle.lm, 72
wle.lm.summaries, 75
wle.negativebinomial, 77
wle.normal, 79
wle.normal.mixture, 81
wle.normal.multi, 83
wle.normal.multi.summaries, 85
wle.normal.summaries, 86
wle.onestep, 87
wle.onestep.summaries, 89
wle.poisson, 89
wle.smooth, 91
wle.stepwise, 93
wle.stepwise.summaries, 96
wle.t.test, 97
wle.var.test, 100
wle.vonmises, 102
wle.weights, 104
wle.wrappernormal, 105

*Topic ts
wle.aic.ar, 37
wle.aic.ar.summaries, 41
wle.ar, 43
wle.fracdiff, 56

*Topic utilities
extractRoot, 6
extractRoot.wle.glm, 3, 4
factor, 63
family, 7, 8, 30, 61, 64, 65, 70
family.wle.glm (wle.glm.summaries), 69
fitted, 70
fitted.values, 66, 96
fitted.wle.lm (wle.lm.summaries), 75
formula, 61
formula.wle.lm (wle.lm.summaries), 75
glm, 69
glm.control, 62, 63
hald, 9
mde.vonmises, 9
mde.wrappernormal, 11
methods, 16, 18, 21, 24, 32, 41, 42, 51, 55, 69,
75, 85, 86, 89, 96
mle.aic, 14, 16
mle.aic.summaries, 16
mle.cp, 17, 19, 26
mle.cp.summaries, 18
mle.cv, 20, 22
mle.cv.summaries, 21
mle.stepwise, 22
mle.stepwise.summaries, 24
mle.vonmises, 11, 104
mle.wrappernormal, 12, 13, 107
model.frame, 8, 65
model.frame.wle.lm (wle.lm.summaries), 75
model.offset, 62
na.exclude, 62
na.fail, 62, 75
na.omit, 62, 75
na.resid, 70
offset, 62
options, 62, 75, 76
par, 29
plot.mle.cp, 25
plot.wle.cp, 27
plot.wle.lm, 28, 76
print.mde.vonmises (mde.vonmises), 9

anova, 4, 63, 66
anova.wle.glm.root, 3, 8, 63, 66, 70
anova.wle.lm (anova.wle.glm.root), 3
artificial, 4
as.data.frame, 62

binary, 5
binomial, 8, 65

cavendish, 6
circular, 10–13, 102, 104, 105, 107
class, 14, 17, 20, 23, 36, 46, 49, 53, 60, 73,
78, 79, 82, 84, 87, 90, 92, 94
coeff, 70
coeff.wle.lm (wle.lm.summaries), 75
coefficients, 64, 96
defiance, 70
df.residual, 70
dist, 62, 71
effects, 66, 70
print.mde.weptternormal
  (mde.weptternormal), 11
print.mle.aic (mle.aic.summaries), 16
print.mle.cp (mle.cp.summaries), 18
print.mle.cv (mle.cv.summaries), 21
print.mle.stepwise
  (mle.stepwise.summaries), 24
print.summary.mle.aic
  (mle.aic.summaries), 16
print.summary.mle.cp
  (mle.cp.summaries), 18
print.summary.mle.cv
  (mle.cv.summaries), 21
print.summary.mle.stepwise
  (mle.stepwise.summaries), 24
print.summary.wle.aic
  (wle.aic.summaries), 42
print.summary.wle.aic.ar
  (wle.aic.ar.summaries), 41
print.summary.wle.cp
  (wle.cp.summaries), 51
print.summary.wle.cv
  (wle.cv.summaries), 55
print.summary.wle.glm
  (summary.wle.glm), 32
print.summary.wle.lm
  (wle.lm.summaries), 75
print.summary.wle.stepwise
  (wle.stepwise.summaries), 96
print.wle.aic (wle.aic.summaries), 42
print.wle.aic.ar
  (wle.aic.ar.summaries), 41
print.wle.binomial (wle.binomial), 46
print.wle.cp (wle.cp.summaries), 51
print.wle.cv (wle.cv.summaries), 55
print.wle.gamma (wle.gamma), 59
print.wle.glm (wle.glm), 61
print.wle.lm (wle.lm.summaries), 75
print.wle.negativebinomial
  (wle.negativebinomial), 77
print.wle.normal
  (wle.normal.summaries), 86
print.wle.normal.mixture
  (wle.normal.mixture), 81
print.wle.normal.multi
  (wle.normal.multi.summaries), 85
print.wle.onestep
  (wle.onestep.summaries), 89
print.wle.poisson (wle.poisson), 89
print.wle.smooth (wle.smooth), 91
print.wle.stepwise
  (wle.stepwise.summaries), 96
print.wle.t.test (wle.t.test), 97
print.wle.vonmises (wle.vonmises), 102
print.wle.weptternormal
  (wle.weptternormal), 105
quasi, 63
residuals, 66, 70, 96
residuals.glm, 34
residuals.wle.glm (wle.glm.summaries), 69
residualsAnscombe, 30
rocky, 31
selection, 32
stat.anova, 3
summary, 34, 63, 66
summary.mle.aic (mle.aic.summaries), 16
summary.mle.cp (mle.cp.summaries), 18
summary.mle.cv (mle.cv.summaries), 21
summary.mle.stepwise
  (mle.stepwise.summaries), 24
summary.wle.aic (wle.aic.summaries), 42
summary.wle.aic.ar
  (wle.aic.ar.summaries), 41
summary.wle.cp (wle.cp.summaries), 51
summary.wle.cv (wle.cv.summaries), 55
summary.wle.glm (summary.wle.glm), 32
summary.wle.lm (wle.lm.summaries), 75
summary.wle.stepwise
  (wle.stepwise.summaries), 96
symnum, 33
terms, 8, 65
try, 60
uniroot, 59, 60, 92
weights, 96
weights.wle.glm (wle.glm), 61
weights.wle.lm (wle.lm.summaries), 75
wle.aic, 34, 42
wle.aic.ar, 37, 41
wle.aic.ar.summaries, 41
INDEX

wle.aic.summaries, 42
wle.ar, 40, 43
wle.ar.wls (wle.aic.ar), 37
wle.binomial, 46
wle.cp, 28, 47, 51
wle.cp.summaries, 51
wle.cv, 52, 56
wle.cv.summaries, 55
wle.fracdiff, 56
wle.gamma, 59
wle.glm, 4, 7, 31, 33, 34, 61, 67, 68, 70, 71
wle.glm.control, 62, 67
wle.glm.fit, 67, 70
wle.glm.summaries, 69
wle.glm.weights, 70
wle.lm, 28, 29, 50, 54, 66, 72, 76, 88, 95, 100
wle.lm.summaries, 75
wle.negativebinomial, 77
wle.normal, 79, 82, 86, 100
wle.normal.mixture, 81
wle.normal.multi, 83, 86
wle.normal.multi.summaries, 85
wle.normal.summaries, 86
wle.onestep, 87, 89
wle.onestep.summaries, 89
wle.poisson, 89
wle.smooth, 50, 54, 74, 80, 85, 88, 91, 95
wle.stepwise, 93
wle.stepwise.summaries, 96
wle.t.test, 97
wle.var.test, 100
wle.vonmises, 11, 102
wle.weights, 104
wle.wrappednormal, 13, 105

x.artificial (artificial), 4
x.hald (hald), 9
xdata (selection), 32

y.artificial (artificial), 4
y.hald (hald), 9
ydata (selection), 32