Package: rlmDataDriven (via r-universe)

September 14, 2024

Type Package

Title Robust Regression with Data Driven Tuning Parameter

Version 0.4.0

Author You-Gan Wang, Benoit Liquet, Aurelien Callens and Na Wang.

Maintainer You-Gan Wang <you-gan.wang@qut.edu.au>

Imports stats, MASS, tseries

Description Data driven approach for robust regression estimation in homoscedastic and heteroscedastic context. See Wang et al. (2007), <doi:10.1198/106186007X180156> regarding homoscedastic framework.

License GPL (>= 2.0)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2019-10-03 09:10:02 UTC

Contents

DD-internal *Internal Functions*

Description

Internal functions not to be used by the user.

plasma *plasma*

Description

The data are collected from 273 female patients.

Usage

```
plasma
```

Format

This data frame contains the following columns:

y plasma beta-carotene

calories calories in KJ

dietary dietary beta-carotene

Source

Jiang, Y., Wang, Y-G., Fu, L., & Wang, X. (2018). Robust Estimation Using Modified Huber's Functions With New Tails. Technometrics, in press, see <doi:10.1080/00401706.2018.1470037>.

rlmDD *Data driven robust methods*

Description

Robust estimation often relies on a dispersion function that is more slowly varying at large values than the squared function. However, the choice of tuning constant in dispersion function may impact the estimation efficiency to a great extent. For a given family of dispersion functions, we suggest obtaining the 'best' tuning constant from the data so that the asymptotic efficiency is maximized.

This library provides a robust linear regression with a tuning parameter being automatically chosen to provide the necessary resistance against outliers. The robust (loss) functions include the Huber, Tukey bisquare and the exponential loss.

Usage

```
rlmDD(yy, xx, beta0, betaR, method, plot)
```

Arguments

yy	Vector representing the response variable
xx	Design matrix of the covariates excluding the intercept in the first column
beta0	Initial parameter estimate using lm
betaR	Robust estimate of beta with a fixed tuning constant using rlm
method	Huber, Bisquare or Exponential
plot	"Y" gives a plot: the efficiency factor versus a range of tunning parameter values.

Value

The function returns a list including

esti	Value of the robust estimate
Std.Error	Standard error of the robust estimate
tunning	Optimum tunning parameter
R2	R-squared value

Author(s)

You-Gan Wang, Na Wang

References

Wang, Y-G., Lin, X., Zhu, M., & Bai, Z. (2007). Robust estimation using the Huber function with a data-dependent tuning constant. Journal of Computational and Graphical Statistics, 16(2), 468-481.

Wang, X., Jiang, Y., Huang, M., & Zhang, H. (2013). Robust variable selection with exponential squared loss. Journal of the American Statistical Association, 108, 632-643.

Wang, N., Wang, Y-G., Hu, S., Hu, Z. H., Xu, J., Tang, H., & Jin, G. (2018). Robust Regression with Data-Dependent Regularization Parameters and Autoregressive Temporal Correlations. Environmental Modeling & Assessment, in press.

See Also

rlm function from package MASS

Examples

```
library(MASS)
data(stackloss)

LS <- lm(stack.loss ~ stack.x)
RB <- rlm(stack.loss ~ stack.x, psi = psi.huber, k = 1.345)
DD1 <- rlmDD(stack.loss, stack.x, LS$coef, RB$coef, method = "Huber",
```

```
plot = "Y")

LS <- lm(stack.loss ~ stack.x)
RB <- rlm(stack.loss ~ stack.x, psi = psi.bisquare, c = 4.685)
DD2 <- rlmDD(stack.loss, stack.x, LS$coef, RB$coef, method = "Bisquare",
plot = "Y")

LS <- lm(stack.loss ~ stack.x)
RB <- rlm(stack.loss ~ stack.x, psi = psi.huber, k = 1.345)
DD3 <- rlmDD(stack.loss, stack.x, LS$coef, RB$coef, method = "Exponential",
plot = "Y")


## Plasma dataset


data(plasma)

y <- plasma$y
x <- cbind(plasma$calories, plasma$dietary)

LS <- lm(y ~ x)
RB <- rlm(y ~ x, psi = psi.huber, k = 1.345)
DD.h <- rlmDD(y, x, LS$coef, RB$coef, method = "Huber", plot = "Y")

LS <- lm(y ~ x)
RB <- rlm(y ~ x, psi = psi.bisquare, c = 4.685)
DD.b <- rlmDD(y, x, LS$coef, RB$coef, method = "Bisquare", plot = "Y")

LS <- lm(y ~ x)
RB <- rlm(y ~ x, psi = psi.huber, k = 1.345)
DD.e <- rlmDD(y, x, LS$coef, RB$coef, method = "Exponential", plot = "Y")
```

`rlmDD_het` *Robust estimation for autoregressive models with heterogeneity*

Description

Performs robust regression for autoregressive models with heterogeneity.

First, a M-estimation is performed on the data assuming that the variance is constant. The residuals of this model are used to robustly estimate the variance parameter. Then, a weighted M-estimation with variance as weight is used to update the regression parameters. These steps are repeated for different values of tuning parameter. The best tuning parameter is the one which minimizes the variance of the estimator.

Finally, lagged term are built and added to the regression model therefore accounting for temporal correlations. The loss function used is Huber's function.

Usage

```
rlmDD_het(yy, xx, var.function = c("power", "exponential"),
phi.par = TRUE, tuning.para = NULL, step = 0.1, n.lag = NULL,
print.summary = TRUE)
```

Arguments

yy	Vector representing the response variable
xx	Design matrix of the covariates (including the intercept)
var.function	Assumed function for the variance. "power" function corresponds to $\sqrt{Var} = \sigma = \phi\|x^T\beta\|^\gamma$ and "exponential" to $\sqrt{Var} = \sigma = \phi e^{\gamma\|x^T\beta\|}$.
phi.par	If TRUE, the function estimate the phi parameter. If FALSE, phi is assumed equal to 1.
tuning.para	If NULL, the function will run the estimation procedure for a range of value between 0 and 3 and will select the tuning parameter that minimizes the variance of the estimates. The user can also indicate a value of tuning parameter: in this case the estimation procedure will be evaluated once with the selected value of the tuning parameter.
step	Only works when tuning.para = NULL, indicates the increment of the tuning parameter sequence (between 0 and 3) tested by the function. It will determine the precision of the tuning parameter. Caution : a smaller value indicates a larger number of value tested, resulting in a longer computing time.
n.lag	If NULL, a pAcf plot of the resiuduals will appear and you will have to indicate the number of lags the method has to include. The user can also give an integer corresponding to the number of lags desired.
print.summary	If TRUE, prints a summary of the estimates.

Value

The function returns a list including

coefficients	Value of the robust estimates
residuals	Residuals of the model.
p_residuals	Pearson residuals of the model.
r_residuals	Robust pearson residuals of the model : $\psi(p_r esiduals, c)$

with ψ the derivative of the loss function and c the chosen tuning parameter.

fitted values	Fitted values obtained with the robust method
vcov	Variance-covariance matrix of the estimates
summary	Summary of the model including: values, standard errors and z-values of the estimates
model	Design matrix of the model
tuningpara	When tuning.para = NULL, list containing the optimal tuning parameter, all the values of tuning parameter tested and their associated variance obtained.
varpara	Estimates of the variance parameters

Author(s)

Aurelien Callens, You-Gan Wang, Benoit Liquet

References

Callens, A., Wang, Y-G., Fu, L. & Liquet, B. (2018). Robust estimation for autoregressive models with heterogeneity. Submitted.

See Also

`rlm` function from package `MASS`

Examples

```
library(tseries)
data(ice.river)
xx <- model.matrix(flow.vat ~ prec + temp, data = ice.river)
yy <- flow.jok

least_square <- lm(flow.vat ~ prec + temp, data = ice.river)
pacf(least_square$residuals)
qqnorm(least_square$residuals)
qqline(least_square$residuals, col = "red", lwd = 2)

#With choice of optimal tuning parameter and 2 lags.
#Note that if lag = NULL, a Pacf plot will appear to help you choose
#the number of lags, you will need to input this number in the console.

model_1 <- rlmDD_het(yy, xx, var.function = "exponential",
                                    tuning.para = NULL, n.lag = 2)

pacf(model_1$p_residuals)
qqnorm(model_1$r_residuals)
qqline(model_1$r_residuals, col = "red", lwd = 2)

#For fixed number of lags and tuning parameter
model_2 <- rlmDD_het(yy, xx, var.function = "exponential",
                                   tuning.para = 1.345, n.lag = 2)
```

`whm` *Weighted M-estimation*

Description

This function performs a weighted M-estimation described by Carroll and Ruppert (1982) with the Huber loss function. First, a M-estimation is performed on the data assuming that the variance is constant. The residuals of this model are used to robustly estimate the variance parameter. Then, a weighted M-estimation with variance as weight is used to update the regression parameters. These steps are iterated until desired convergence.

Usage

```
whm(yy, xx, var.function = "power", tuning.para = 1.345, ite = 5)
```

Arguments

`yy`	Vector representing the response variable
`xx`	Design matrix of the covariates including the intercept in the first column
`var.function`	Assumed function for the variance. "`power`" function corresponds to $\sqrt{(Var)} = \sigma = \phi\|x^T\beta\|^\gamma$ and "`exponential`" to $\sqrt{(Var)} = \sigma = \phi e^{\gamma\|x^T\beta\|}$.
`tuning.para`	Value of the tuning parameter associated with the loss function.
`ite`	Number of iterations for the estimation procedure.

Value

The function returns a list including

`esti`	Value of the robust estimate
`Std.Error`	Standard error of the robust estimate
`tunning`	Optimum tunning parameter
`R2`	R-squared value

Author(s)

Aurelien Callens, You-Gan Wang, Benoit Liquet.

References

Carroll, R. J., & Ruppert, D. (1982). Robust estimation in heteroscedastic linear models. The annals of statistics, 429-441.

See Also

`rlm` function from package `MASS`

Examples

```
library(MASS)
data(stackloss)

LS <- lm(stack.loss ~ stack.x)
RB <- rlm(stack.loss ~ stack.x, psi = psi.huber, k = 1.345)

yy <- stack.loss
xx <- model.matrix(stack.loss ~ stack.x)

#With power function as variance function
WHM_p <- whm(yy, xx, var.function = "power", tuning.para = 1.345)

#With exponential function as variance function
WHM_e <- whm(yy, xx, var.function = "exponential", tuning.para = 1.345)
```

Index