Package: speedglm (via r-universe)

September 10, 2024

Type Package

Title Fitting Linear and Generalized Linear Models to Large Data Sets

Version 0.3-5

Date 2023-04-20

Author Marco Enea [aut, cre], Ronen Meiri [ctb] (on behalf of DMWay Analytics LTD), Tomer Kalimi [ctb] (on behalf of DMWay Analytics LTD)

Maintainer Marco Enea <marco.enea@unipa.it>

Depends Matrix, MASS, biglm

Imports methods, stats

Description Fitting linear models and generalized linear models to large data sets by updating algorithms, according to the method described in Enea (2009, ISBN: 9788861294257).

License GPL

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2023-05-06 10:10:02 UTC

Contents

speedglm-package *Fitting Linear and Generalized Linear Models to Large Data Sets*

Description

Fits Linear and Generalized Linear Models to large data sets. For data loaded in R memory the fitting is usually fast, especially if R is linked against an optimized BLAS. For data sets of size larger than R memory, the fitting is made by an updating algorithm.

Details

Package:	speedglm
Type:	Package
Version:	0.3-5
Date:	2023-04-20
Depends:	Matrix, stats, MASS
License:	GPL
LazyLoad:	yes

Author(s)

Marco Enea <marco.enea@unipa.it>, with contributions from Ronen Meiri and Tomer Kalimi (on behalf of DMWay Analytics LTD).

Maintainer: Marco Enea <marco.enea@unipa.it>

add1.speedlm *Wrappers to the* add1 *and* drop1 *methods for speedlm and speedglm objects*

Description

These are adviced to be used for speedlm and speedglm models fitted on moderately large data sets. It is also possible to use stepAIC function from package MASS.

Usage

```
## S3 method for class 'speedlm'
## S3 method for class 'speedlm'
add1(object, scope, scale = 0, test = c("none", "Chisq","F"),
                       x = NULL, k = 2, data, ...)
## S3 method for class 'speedlm'
drop1(object, scope, scale = 0, all.cols = TRUE,
                        test = c("none","Chisq", "F"), k = 2, data, ...)
## S3 method for class 'speedlm'
extractAIC(fit, scale = 0, k=2,...)
## S3 method for class 'speedlm'
nobs(object, use.fallback = FALSE, ...)

## S3 method for class 'speedglm'
## S3 method for class 'speedglm'
add1(object, scope, scale = 0, test = c("none", "LRT",
                         "Chisq", "F"), x = NULL, k = 2, ...)
## S3 method for class 'speedglm'
drop1(object, scope, scale = 0, test = c("none", "LRT",
                         "Chisq", "F"), k = 2, ...)
## S3 method for class 'speedglm'
extractAIC(fit, scale = 0, k=2,...)
## S3 method for class 'speedglm'
nobs(object, use.fallback = FALSE, ...)
```

Arguments

object	a speedlm or speedglm object for which model=TRUE was previously set to.
fit	a speedlm or speedglm object
scope	see add1 from package stats.
scale	see add1 from package stats.
all.cols	see drop1 from package stats.
test	see add1 from package stats. Currently, test "Rao" is not implemented.
x	see add1 from package stats.
k	see add1 from package stats.
data	the data that the model was previously fitted to. If not provided, these will be searched in the parent environment.
use.fallback	logical. Should fallback methods be used to try to guess the value?
...	further optional arguments.

Details

It is possible to use functions step() and stepAIC() for both speedlm and speedglm objects but objects fitted using updateWithMoreData().

Value

An object of classes "anova" and "data.frame" summarizing the differences in fit between the models.

Warnings

Note that these functions have been poorly tested and need to be checked out more carefully.

Author(s)

Ronen Meiri and Marco Enea

Examples

```
set.seed(10)
n <- 50
k <- 3
x <- round(matrix(rnorm(n * k), n, k), digits = 3)
beta <- c(0.05,0.5,0.8,1.3,1.8)
y <- drop(tcrossprod(cbind(1,x,x[,2]*x[,3]),t(beta))) + rnorm(n,,0.2)
colnames(x) <- c("s1", "s2", "s3")
da <- data.frame(y, x)

m0 <- speedlm(y ~ 1, data = da,model=TRUE, y=TRUE)
m0.1 <- add1(m0,scope=~s1+s2+s3, data = da)
m1 <- step(m0,scope=~s1+s2+s3)
m1
```

---

control *Miscellanea of functions*

Description

Utility functions for least squares estimation in large data sets.

Usage

```
control(B, symmetric = TRUE, tol.values = 1e-7, tol.vectors = 1e-7,
       out.B = TRUE, method = c("eigen", "Cholesky"))
cp(X, w = NULL, row.chunk = NULL, sparse = FALSE)
is.sparse(X, sparselim = .9, camp = .05)
```

Arguments

B	a squared matrix.
symmetric	logical, is B symmetric?
tol.values	tolerance to be consider eigenvalues equals to zero.
tol.vectors	tolerance to be consider eigenvectors equals to zero.
out.B	Have the matrix B to be returned?
method	the method to check for singularity. By default is "eigen", and an eigendecomposition of X'X is made. The "Cholesky" method is faster than "eigen" and does not use tolerance, but the former seems to be more stable for opportune tolerance values.
X	the model matrix.
w	a weights vector.
sparse	logical, is X sparse?
sparselim	a real in the interval [0; 1]. It indicates the minimal proportion of zeroes in the data matrix X in order to consider X as sparse

eigendec Logical. Do you want to investigate on rank of X? You may set to

row.chunk	an integer which indicates the total rows number compounding each of the first g-1 blocks. If row.chunk is not a divisor of nrow(X), the g-th block will be formed by the remaining data.
camp	the sample proportion of elements of X on which the survey will be based.

Details

Function control makes an eigendecomposition of B according established values of tolerance. Function cp makes the cross-product X'X by partitioning X in row-blocks. When an optimized BLAS, such as ATLAS, is not installed, the function represents an attempt to speed up the calculation and avoid overflows with medium-large data sets loaded in R memory. The results depending on processor type. Good results are obtained, for example, with an AMD Athlon dual core 1.5 Gb RAM by setting row.chunk to some value less than 1000. Try the example below by changing the matrix size and the value of row.chunk. If the matrix X is sparse, it will have class "dgCMatrix" (the package Matrix is required) and the cross-product will be made without partitioning. However, good performances are usually obtained with a very high zeroes proportion. Function is.sparse makes a quick sample survey on sample proportion of zeroes in X.

Value

for the function control, a list with the following elements:

XTX	the matrix product B without singularities (if there are).
rank	the rank of B
pivot	an ordered set of column indeces of B with, if the case, the last $rank + 1, ..., p$ columns which indicate possible linear combinations.

for the function cp:

`new.B` the matrix product X'X (weighted, if `w` is given).

for the function `is.sparse`:

`sparse` a logical value which indicates if the sample proportion of zeroes is greater than `sparselim`, with the sample proportion as attribute.

Author(s)

Marco ENEA

See Also

eigen, chol, qr, crossprod

Examples

```
#### example 1.

n <- 100
k <- 5
x <- round(matrix(rnorm(n*k),n,k),digits=4)
y <- rnorm(n)

# if an optimized BLAS is not installed, depending on processor type, cp() may be
# faster than crossprod() for large matrices.

a1 <- crossprod(x)
a2 <- cp(x,,row.chunk = 50)
all.equal(a1, a2)

#### example 2.1.
x[,2] <- x[,1] + 2*x[,3]  # x has rank 9

# estimation by least squares
A <- function(){
  A1 <- control(crossprod(x))
  ok <- A1$pivot[1:A1$rank]
  as.vector(solve(A1$XTX,crossprod(x[,ok],y)))
}
# estimation by QR decomposition
B <- function(){
  B1 <- qr(x)
  qr.solve(x[,B1$pivot[1:B1$rank]],y)
}
a <- A()
b <- B()

all.equal(a,b)


### example 3.
n <- 1000
```

```
fat1 <- gl(20,50)
y <- rnorm(n)
da <- data.frame(y,fat1)
m <- model.matrix(y ~ factor(fat1),data = da)
is.sparse(m)
```

`data1` *A toy dataset*

Description

The `data1` dataset has 100 rows and 4 columns.

Usage

```
data(data1)
```

Format

A data frame with 100 observations on the following 4 variables.

`y` a gamma-distributed response variable

`fat1` a four-level factor

`x1` a numeric covariate

`x2` a numeric covariate

Details

This is a toy dataset used to show how function `shglm` works.

Examples

```
data(data1)
```

`predict.speedglm` *Predict method for a speedglm object*

Description

`summary` The method is currently under construction but some functionalities are available.

Usage

```
## S3 method for class 'speedglm'
predict(object, newdata, type = c("link", "response"),
                           na.action = na.pass, ...)
```

Arguments

`object`	an object of class 'speedlgm'.
`newdata`	An optional data frame with new data or the original data.
`type`	Type of prediction.
`na.action`	function determining what should be done with missing values in `newdata`.
`...`	further optional arguments

Details

If `newdata` is omitted prediction are based on the data used for the fit only if argument `fitted` was previously set to `TRUE` in the speedglm object. Currently the method does not work for function `shglm`.

Value

`pred`	a vector of predictions.

Author(s)

Tomer Kalimi and Marco Enea

See Also

speedglm

Examples

```
data(data1)
mod <- speedglm(y~x1+x2+factor(fat1), data=data1, family=Gamma(log), fitted=TRUE)
predict(mod)
```

`predict.speedlm` *Predict method for a speedlm object*

Description

`summary` The method is currently under construction but some functionalities are available.

Usage

```
## S3 method for class 'speedlm'
predict(object, newdata, na.action = na.pass, ...)
```

Arguments

`object`	an object of class 'speedlm'.
`newdata`	An optional data frame with new data or the original data.
`na.action`	function determining what should be done with missing values in `newdata`.
`...`	further optional arguments

Details

If `newdata` is omitted prediction are based on the data used for the fit only if argument `fitted` was previously set to `TRUE` in the speedlm object.

Value

`predictor`	a vector of predictions.

Author(s)

Tomer Kalimi and Marco Enea

See Also

speedlm

Examples

```
data(data1)
mod <- speedglm(y~x1+x2+factor(fat1), data=data1, family=Gamma(log), fitted=TRUE)
predict(mod)
```

speedglm *Fitting Generalized Linear Models for Large Data Sets*

Description

speedglm and speedglm.wfit fit GLMs to medium-large data sets, that is those storable into the R memory. The highest performances, in terms of computation time, are obtained when R is linked against an optimized BLAS, such as ATLAS. The function shglm is for a data set stored into a file of size greater than the available memory, and takes as argument a function to manipulate connections.

Usage

```
## S3 method for class 'data.frame':
speedglm(formula,data,family=gaussian(),weights=NULL,start=NULL,
         etastart=NULL,mustart=NULL,offset=NULL,maxit=25, k=2,
         sparse=NULL,set.default=list(), trace=FALSE,
         method=c('eigen','Cholesky','qr'), model=FALSE, y=FALSE,
         fitted=FALSE,...)

## S3 method for class 'matrix':
speedglm.wfit(y, X, intercept=TRUE, weights=NULL,row.chunk=NULL,
              family=gaussian(), start=NULL, etastart=NULL,
              mustart=NULL, offset=NULL, acc=1e-08, maxit=25, k=2,
              sparselim=.9,camp=.01, eigendec=TRUE, tol.values=1e-7,
              tol.vectors=1e-7, tol.solve=.Machine$double.eps,
              sparse=NULL,method = c('eigen','Cholesky','qr'),
              trace=FALSE,...)

## S3 method for class 'function':
shglm(formula, datafun, family = gaussian(), weights.fo = NULL, start = NULL,
      etastart = NULL, mustart = NULL, offset = NULL, maxit = 25, k = 2,
      chunksize = 5000, sparse = NULL, trace = FALSE, all.levels = FALSE,
      set.default = list(),...)
```

Arguments

Most of arguments are the same of glm or bigglm but with some difference.

formula	the same of glm.
data	a data frame.
datafun	a function which uses connections. See the example below.
family	the same of glm, but it must be specified with brackets.
start	the same of glm.
weights	the same of glm.

weights.fo	weights for the response. It must be specified as a formula (see the example below).
etastart	the same of glm.
mustart	the same of glm.
offset	the same of glm.
intercept	the same of glm.
X	the same of x in glm.fit.
y	the same of glm and glm.fit.
maxit	the same of glm.
k	numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC.
trace	logical. Do you want to be informed about the model estimation progress?
sparse	logical. Is the model matrix sparse? By default is NULL, so a quickly sample survey will be made.
chunksize	an integer indicates the number of rows of the data file to read at time.
all.levels	logical, are all factor's levels present in each data chunk?
set.default	a list in which to specify the below parameters.
sparselim	a real in the interval [0, 1]. It indicates the minimal proportion of zeroes in the data matrix X in order to consider X as sparse.
camp	see the function is.sparse.
eigendec	logical. Do you want to check the rank of X? You may set it to false if you are sure that X is full rank.
row.chunk	an integer, see the function cp for details.
acc	tolerance to be used for the estimation.
tol.solve	see the function solve.
tol.values	see the function control.
tol.vectors	see the function control.
method	the method chosen to detect for singulatity.
model	logical. If TRUE the model frame will be returned.
fitted	logical. If TRUE the fitted values will be returned.
...	further optional arguments.

Details

The function shglm works like biglm, but it checks for singularity and does not impose restrictions on factors. Since during the IWLS estimation shglm uses repeated accesses to data file stored, for example, into the hard disk, the estimation time could be very long. Unlike from glm or biglm, the functions of class 'speedglm' do not use the QR decomposition, but directly solve the equations in the form of Iterative(-ly) (Re-)Weighted Least Squares (IWLS). The memory size of an object of class 'speedglm' is $O(p^2)$, where p is the number of covariates, unless one or more of argument model, y and fitted are set to TRUE. If an optimized BLAS is not installed, an attempt to speed

up calculations might be done by setting `row.chunk` to some value, usually less than 1000, in `set.default`. See the function cp for details.
If the model matrix is (very) sparse, the package `Matrix` could be used. Note that if method 'qr' is chosen, then the qr decomposition will not be applied on matrix X, as in `lm`, but on X'WX.

Value

`coefficients`	the estimated coefficients.
`logLik`	the log likelihood of the fitted model.
`iter`	the number of iterations of IWLS used.
`tol`	the maximal value of tolerance reached.
`convergence`	a logical value which indicates if convergence was reached.
`family`	the family object used.
`link`	the link function used.
`df`	the degrees of freedom of the model.
`XTX`	the product X'X (weighted, if the case).
`dispersion`	the estimated dispersion parameter of the model.
`ok`	the set of column indeces of the model matrix where the model has been fitted.
`rank`	the rank of the model matrix.
`RSS`	the estimated residual sum of squares of the fitted model.
`aic`	the estimated Akaike Information Criterion.
`sparse`	a logical value which indicates if the model matrix is sparse.
`deviance`	the estimated deviance of the fitted model.
`nulldf`	the degrees of freedom of the null model.
`nulldev`	the estimated deviance of the null model.
`ngoodobs`	the number of non-zero weighted observations.
`n`	the number of observations.
`intercept`	a logical value which indicates if an intercept has been used.
`terms`	the terms object used.
`call`	the matched call.
`model`	Either NULL or, if `model` was previously set to TRUE, the model frame.
`y`	Either NULL or, if `y` was previously set to TRUE, the response variable.
`linear.predictors`	Either NULL or, if `fitted` was previously set to TRUE, the fitted values.
`offset`	the model offset.

Note

All the above functions make an object of class 'speedglm'.
In the current package version, arguments `start`, `mustart` and `etastart` of function `shglm` have been disabled. These will be restored in future.

Author(s)

Marco Enea. Ronen Meiri contributed with method 'qr'

References

Enea, M. (2009) Fitting Linear Models and Generalized Linear Models with large data sets in R. In *book of short papers, conference on "Statistical Methods for the analysis of large data-sets", Italian Statistical Society*, Chieti-Pescara, 23-25 September 2009, 411-414. ISBN:978-88-6129-425-7

Bates, D. (2009) Comparing Least Square Calculations. Technical report.

Lumley, T. (2009) biglm: bounded memory linear and generalized linear models. *R package version 0.7*. https://CRAN.R-project.org/package=biglm.

See Also

speedlm, bigglm, glm

Examples

```
# The following comparison among glm(), bigglm() and speedglm() cannot be considered rigorous
# and exhaustive, but it is only to give an idea of the computation time.
# It may take a long time.
library(biglm)
n<-10000
k<-70
y <- rgamma(n,1.5,1)
x <-round( matrix(rnorm(n*k),n,k),digits=3)
colnames(x) <-paste("s",1:k,sep = "")
da<- data.frame(y,x)
fo <- as.formula(paste("y~",paste(paste("s",1:k,sep=""),collapse="+")))

system.time(m1 <- glm(fo,data=da,family=Gamma(log)))
system.time(m2 <- bigglm(fo,data=da,family=Gamma(log)))
system.time(m3 <- speedglm(fo,data=da,family=Gamma(log)))

# You may also try speedglm when R is linked against an optimized BLAS,
# otherwise try to run the following function. In some computers, it is
# faster for large data sets.
system.time(m4 <- speedglm(fo,data=da,family=Gamma(log),set.default=list(row.chunk=50)))

##################

## An example of function using a connection to an out-memory file
## This is a slightly modified version of the function from the bigglm's help page
 make.data<-function(filename, chunksize,...){
     conn<-NULL
     function(reset=FALSE){
     if(reset){
```

```
        if(!is.null(conn)) close(conn)
        conn<<-file(filename,open="r")
      } else{
        rval<-read.table(conn, nrows=chunksize,...)
        if ((nrow(rval)==0)) {
             close(conn)
             conn<<-NULL
             rval<-NULL
        }
        return(rval)
      }
  }
}


# data1 is a small toy dataset
data(data1)
tmp_data1<-tempfile("data1",fileext=".txt")
write.table(data1,tmp_data1,row.names=FALSE,col.names=FALSE)

da<-make.data(tmp_data1,chunksize=50,col.names=c("y","fat1","x1","x2"))

# Caution! make sure to close the connection once you have run command #1
da(reset=TRUE) #1: opens the connection to "data1.txt"
da(reset=FALSE) #2: reads the first 50 rows (out of 100) of the dataset
da(reset=FALSE) #3: reads the second 50 rows (out of 100) of the dataset
da(reset=FALSE) #4: is NULL: this latter command closes the connection

# fat1 is a factor with four levels
b1<-shglm(y~factor(fat1)+x1,weights=~I(x2^2),datafun=da,family=Gamma(log))
b2<-bigglm(y~factor(fat1)+x1,weights=~I(x2^2),data=da,family=Gamma(log))
summary(b1)
summary(b2)
```

---

speedlm *Fitting Linear Models to Large Data Sets*

Description

The functions of class 'speedlm' may speed up the fitting of LMs to large data sets. High performances can be obtained especially if R is linked against an optimized BLAS, such as ATLAS.

Usage

```
# S3 method of class 'data.frame'
speedlm(formula, data, weights = NULL, offset = NULL, sparse = NULL,
        set.default = list(), method=c('eigen','Cholesky','qr'),
        model = FALSE, y = FALSE, fitted = FALSE, subset=NULL, ...)
```

```
# S3 method of class 'matrix'
speedlm.fit(y, X, intercept = FALSE, offset = NULL, row.chunk = NULL,
            sparselim = 0.9, camp = 0.01, eigendec = TRUE,
            tol.solve = .Machine$double.eps, sparse = NULL, tol.values = 1e-07,
            tol.vectors = 1e-07, method=c('eigen','Cholesky','qr'), ...)

speedlm.wfit(y, X, w, intercept = FALSE, offset = NULL, row.chunk = NULL,
             sparselim = 0.9, camp = 0.01, eigendec = TRUE,
             tol.solve = .Machine$double.eps, sparse = NULL, tol.values = 1e-07,
             tol.vectors = 1e-07, method=c('eigen','Cholesky','qr'), ...)

# S3 method of class 'speedlm' (object) and 'data.frame' (data)
## S3 method for class 'speedlm'
update(object, formula, data, add=TRUE, evaluate=TRUE,
                           subset=NULL, offset=NULL, weights=NULL,...)

# S3 method of class 'speedlm' (object) and 'data.frame' (data)
updateWithMoreData(object, data, weights = NULL, offset = NULL, sparse = NULL,
                   all.levels = FALSE, set.default = list(), subset=NULL,...)
```

Arguments

Most of arguments are the same of functions lm but with some difference.

formula	the same of function lm.
data	the same of function lm.
weights	the same of function lm.
w	model weights.
intercept	a logical value which indicates if an intercept is used.
offset	the same of function lm.
X	the same of x in function lm.
y	the same of lm,lm.wfit and lm.fit .
sparse	logical. Is the model matrix sparse? By default is NULL, so a quickly sample survey will be made.
set.default	a list in which to specify the parameters to pass to the functions cp, control and is.sparse.
sparselim	a value in the interval [0, 1]. It indicates the minimal proportion of zeroes, in the model matrix X, in order to consider X as sparse.
camp	see function is.sparse.
eigendec	logical. Do you want to investigate on rank of X? You may set it to false if you are sure that X is full rank.
row.chunk	an integer, see the function cp for details.
tol.solve	see function solve.
tol.values	see function control.

tol.vectors	see function control.
method	the method used to check for singularity. The default is 'eigen'. See details
object	an object of class 'speedlm'.
all.levels	are all levels of eventual factors present in each data chunk? If so, set all.levels to true to speed up the fitting.
model	logical. Should the model frame be returned?
fitted	logical. Should the fitted values be returned?
subset	the same of function lm
add	logical. Are additional data coming from a new chunk provided?
evaluate	logical. If true evaluate the new call else return the call.
...	further optional arguments.

Details

Unlikely from lm or biglm, the functions of class 'speedlm' do not use the QR decomposition of the X-matrix, but directly solve the normal equations. In some extreme case, this might have some problem of numerical stability, but it may take advantage from the use of an optimized BLAS. To check for singularity, three options are available. Method "eigen" performs an eigendecomposition of X'X. The 'Cholesky' method is faster than "eigen" and does not use tolerance, but the former seems to be more stable for opportune tolerance values. Method 'qr' does not perform a QR decomposition directly on matrix X, but on X'WX. The memory size of an object of class 'speedlm' is $O(p^2)$, where p is the number of covariates. If an optimized BLAS library is not installed, an attempt to speed up calculations may be done by setting row.chunk to some value, usually less than 1000, in set.default. See the function cp for details. Factors are permitted without limitations.
In the most recent versions, function update.speedlm is now a wrapper to call either updateWithMoreData (the new name of the old update.speedlm, for additional data chunks), or update from package stats.

Value

coefficients	the estimated coefficients.
df.residual	the residual degrees of freedom.
XTX	the product X'X (weighted, if the case).
A	the product X'X (weighted, if the case) not checked for singularity.
Xy	the product X'y (weighted, if the case).
ok	the set of column indeces of the model matrix where the model has been fitted.
rank	the numeric rank of the fitted linear model.
pivot	see the function control.
RSS	the estimated residual sums of squares of the fitted model.
sparse	a logical value indicating if the model matrix is sparse.
deviance	the estimated deviance of the fitted model.
weigths	the weights used in the last updating.

zero.w	the number of non-zero weighted observations.
nobs	the number of observations.
nvar	the number of independent variables.
terms	the terms object used.
intercept	a logical value which indicates if an intercept has been used.
call	the matched call.
model	Either NULL or the model frame, if model was previously set to TRUE.
y	Either NULL or the response variable, if y was previously set to TRUE.
fitted.values	Either NULL or the fitted values, if fitted was previously set to TRUE.
offset	the model offset.
...	others values necessary to update the estimation.

Note

All the above functions make an object of class 'speedlm'.

Author(s)

Marco Enea, with contribution from Ronen Meiri.

References

Enea, M. (2009) Fitting Linear Models and Generalized Linear Models With Large Data Sets in R. In *book of short papers, conference on "Statistical Methods for the analysis of large data-sets", Italian Statistical Society*, Chieti-Pescara, 23-25 September 2009, 411-414. ISBN:978-88-6129-425-7

Klotz, J.H. (1995) Updating Simple Linear Regression. *Statistica Sinica*, **5**, 399-403.

Bates, D. (2009) Comparing Least Square Calculations. Technical report.

Lumley, T. (2009) biglm: bounded memory linear and generalized linear models. *R package version 0.7* https://CRAN.R-project.org/package=biglm.

See Also

summary.speedlm,speedglm, lm, and biglm

Examples

```
data(data1)
da <- data1
do1 <- da[1:30,]
do2 <- da[31:70,]
do3 <- da[71:100,]
```

```
m1 <- speedlm(y ~ factor(fat1) + x1 + x2, data = do1)
m1 <- update(m1, data = do2)
m1 <- update(m1, data = do3)

m2 <- lm(y ~ factor(fat1) + x1 + x2, data = data1)
summary(m1)
summary(m2)



# as before but recursively
make.data <- function(filename, chunksize,...){
    conn <- NULL
    function(reset=FALSE, header=TRUE){
      if(reset){
        if(!is.null(conn)) close(conn)
         conn<<-file(filename,open="r")
      } else{
        rval <- read.table(conn, nrows=chunksize,header=header,...)
        if (nrow(rval)==0) {
          close(conn)
          conn<<-NULL
          rval<-NULL
        }
        return(rval)
      }
    }
}

tmp_da<-tempfile("da",fileext=".txt")
write.table(da,tmp_da,col.names=TRUE,row.names=FALSE,quote=FALSE)

dat <- make.data(tmp_da,chunksize=30,col.names=c("y","fat1","x1", "x2"))
dat(reset=TRUE)
da2 <- dat(reset=FALSE)

# the first model runs on the first 30 rows.
m3 <- speedlm(y ~ factor(fat1) + x1 + x2, data=da2)

# the last three models run on the subsequent 30, 30 and 10 rows, respectively
for (i in 1:3){
  da2 <- dat(reset=FALSE, header=FALSE)
  m3 <- update(m3, data=da2, add=TRUE)
}
all.equal(coef(m1),coef(m3))
```

---

summary.speedglm *Methods to summarize Generalized Linear Models fits*

Description

summary method for the class 'speedglm'.

Usage

```
## S3 method for class 'speedglm'
summary(object,correlation=FALSE,...)
## S3 method for class 'speedglm'
coef(object,...)
## S3 method for class 'speedglm'
vcov(object,...)
## S3 method for class 'speedglm'
logLik(object,...)
## S3 method for class 'speedglm'
AIC(object,...)
```

Arguments

object	an object of class 'speedglm'.
correlation	logical. Do you want to print the correlation matrix? By default it is false.
...	further optional arguments

Value

coefficients	the matrix of coefficients, standard errors, z-statistics and two-side p-values.
df.residual	the component from object.
df.null	the component from object.
null.deviance	the component from object.
deviance	the component from object.
family	the component from object.
call	the component from object.
AIC	the Akaike Information Criterion.
RSS	Residuals sums of squares.
correlation	(only if correlation is true.) The correlations of the estimated coefficients.
logLik	the log-likelihood value.
rank	the component from object.
dispersion	the estimated dispersion parameter of the fitted model.
convergence	the component from object.
iter	the component from object.
tol	the component from object.

Author(s)

Marco ENEA

See Also

speedglm

Examples

```
data(data1)
mod <- speedglm(y~x1+x2+factor(fat1), data=data1, family=Gamma(log))
summary(mod)
```

`summary.speedlm` *Methods to summarize Linear Models fits*

Description

`summary` method for class 'speedlm'.

Usage

```
## S3 method for class 'speedlm'
summary(object, correlation = FALSE,...)
## S3 method for class 'speedlm'
coef(object,...)
## S3 method for class 'speedlm'
vcov(object,...)
## S3 method for class 'speedlm'
logLik(object,...)
## S3 method for class 'speedlm'
AIC(object,...,k = 2)
```

Arguments

`object`	an object of class 'speedlm'.
`correlation`	logical. Do you want to print the correlation matrix? By default it is false.
`k`	numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC.
`...`	further optional arguments

Value

`coefficients`	the matrix of coefficients, standard errors, t-statistics and two-side p-values.
`rdf`	degrees of freedom of the fitted model. It is a component from `object`.
`call`	the component from `object`.
`r.squared`	R^2, the fraction of variance explained by the model.
`adj.r.squared`	the "adjusted" R^2 statistic, penalizing for higher p.

fstatistic	(for models including non-intercept terms) a 3-vector with the value of the F-statistic with its numerator and denominator degrees of freedom.
f.pvalue	p-value of the F-statistic.
RSS	Residual sum of squares.
var.res	estimated variance of residuals.
rank	the component from object.
correlation	(only if correlation is true) the correlations of the estimated parameters.
...	the results from the functions logLik, AIC and vcov.

Author(s)

Marco ENEA

See Also

speedlm

Examples

```
data(data1)
m <- speedlm(y ~ x1 + x2 + factor(fat1), data1)
summary(m)
```

Index