Package 'autoGLM'

December 13, 2016

Type Package

Title Automated fitting, model selection, and documentation functions for GLM modelling of binary response variables.

Version 1.0.1

Author Bo Pieter Johannes Andree

AuthorAffiliation Vrije Universiteit (VU)

Maintainer <b.p.j.andree@vu.nl>

Description Methods and tools for automation of binary response GLM's. Supports linear models with identity, probit and logit link. The package is written with binomial land-use modelling using large data sets in mind, but the methods and tools stretch out to other research cases. The main function is autoGLM, which wraps around most of the other functions contained in the package. However, some general functions are included as well such as a more elaborated function for package loading, an out of memory version to lapply, memory-threshold garbage collection, sample size reduction routines, variable selection routines, and user friendly methods for logit estimation through GLM and GMM.

Depends R (>= 3.2.1)

URL https://github.com/BPJandree/AutoGLM

BugReports https://github.com/BPJandree/AutoGLM

License MIT

ByteCompile TRUE

LazyData TRUE

RoxygenNote 5.0.1

R topics documented:

`accuracy` *Miscelaneous function used in main routines.*

Description

Miscelaneous function used in main routines.

Usage

```
accuracy(testdata, fitteddata)
```

Arguments

`testdata`	a vector of occurrences.
`fitteddata`	a vector of occurrences.

Value

A numeric indicating the degree of correspondence.

Examples

```
observed <- c(1,1,1,0,0,0,0,0,0,0)
predicted <-c(1,0,1,0,0,0,0,1,1,0)

print(accuracy(observed, predicted))
```

autoGLM	*A main function of the package for automated generalized linear model fitting. Includes additional options w.r.t. the* generalizeToSpecific *command.*

Description

This function is a wrapper around the optimization and selection routines in the package and can be used for automated calibration of GLM's on semi large datasets. generalizeToSpecific is more appropiate for manual R sessions, autoGLM is more appropiate for situations when calibration takes a long time. E.g., it allows to run generalizeToSpecific over a vector of dependent variable classes, and to log and write outputs to disk.

Usage

```
autoGLM(data, reclasstable = "default", class = 1,
  outputpath = paste(getwd(), "//", sep = ""), modelname = "autoGLM",
  tracelevel = 1, actions = c("print", "return"), NAval = -9999,
  model = "logit", preselect = "lm", method = "opt.ic", crit.t = 1.64,
  crit.p = 0.05, test = "LR", KLIC = "AICc", accuracytolerance = 0.01,
  confidence.alternative = 0.9, use.share = 0.25, maxsampleruns = 50,
  memorymanagement = TRUE, returnall = FALSE, compress = FALSE,
  JIT = TRUE)
```

Arguments

data	A dataframe with a categorical response variable in the first column, and covariates in subsequent columns. Typically the product of cbind(Y,X).
reclasstable	A table that maps the first column of data into a binary response variable. By default it will be ommitted (the binary response variable will be identical to data[,1]). See also corinetable, See also reclassify.
class	The class that should be 1 in the binary response variable, all other classes in the categorical variable will be set to 0. Defaults to 1. See also reclassify.
outputpath	The location on the hard drive where output wwill be written to. Defaults to getwd().
modelname	The name of the model, will be used when writing a weightsfile. Defaults to "autoGLM". See also exportWeightsfile.
tracelevel	The amount of information to be printed. Passed on to underlying routines. Defaults to 1 for printing, set to 0 for no printing.
actions	Actions to be taken by autoGLM, by default c("print", "return"), may include any combination of c("write", "print", "log", "return"), for writing a geoDMS weightsfile, See also exportWeightsfile, printing results, writing a log file, and returning results as a list object.
NAval	Optional categorical variable that should be dropped by the reclassification scheme. See also reclassify.
model	Main model type that should be calibrated, either "lm", "probit", or "logit". See also generalizeToSpecific.

`preselect` Optional variable preselection using a first order approximation (linear model) of the logit or probit model, by specifying "lm" (default setting). See also `selectX`.

`method` The optimization strategy. Either "opt.ic" to optimize using information criteria, "opt.t" for step-wise elimination of insignificant values (statistically speaking not a sound procedure, but it will provide a parsimonious model that can be usefull as a benchmark), or "opt.h" to optimize by classical hypothesis tests. defaults to "opt.ic". See also See also `opt.ic`, `opt.t`, See also `opt.h`.

`crit.t` The t-value indicating significance when using method "opt.t", defaults to 1.64. `opt.t`.

`crit.p` the p-value used by method "opt.h" in the hypothesis tests. Defaults to 0.05. `opt.h`.

`test` The hypothesis test used by "opt.h". Defaults to "LR" for the Likelihood Ratio test. Other options are "F", for an F test for joint significance of insignificant parameters, or "Chisq" for a wald test against the Chi squared distribution. `opt.h`.

`KLIC` The information criterion used by "opt.ic", either "AIC" or "AICc", defaults to the latter. `opt.ic`.

`accuracytolerance` When aut of sample and within sample accuracy differ more than accuracytolerance, a warning will be issued, which is also logged when specifying "log" in actions. Defaults to 0.01. `accuracy`.

`confidence.alternative` See also `getSamples`, confidence level used for the alternative of dissimilar samples in the sampling routine. Defaults to .85.

`use.share` Share of the data used, See also `getSamples`. Defaults to .25.

`maxsampleruns` See also `getSamples`, defaults to 50.

`memorymanagement` TRUE/FALSE indicating whether garbage collection should be forced regularly when memory usage is high. Defaults to TRUE, recommended setting for large datasets. See also `tgc`.

`returnall` TRUE, FALSE, or "writedisk" indicating whether all the outputted objects for each class should be returned in an array as produced by lapply, or whether only the final output should be returnd as an object. Specifying "writedisk" will write the objects containing results of each class as seperate .RDS files, which you can use to restore the output using readRDS(). `iapply`. Returning an array of all results can consume large amounts of memory as each object contains copies of the used datasets. When working with countrysize datasets, these array objects can easily require over 64gb of RAM. Specifying returnall = FALSE (default setting), is much more more RAM friendly as it stores results for each class in the same memory adress, overwriting previous results. Seting returnall = FALSE, will still write log files and print diagnostics to screen if specified in actions. returnall="writedisk" is the recommended setting, but it is not default. `iapply`.

`compress,` passed on to iapply. Defaults to no compression of RDS output, which is the recommended setting if computation time is valued of disk space. Keep in mind that when using large datasets, autoGLM objects can be several gigabytes in size. `iapply`.

`JIT,` logical indicating whether just-in-time compilation of internal functions should be used. Mainly for historical reasons.

Examples

```
data(ITdata)
datacorinetable)

results <- autoGLM(data=randomlogit, reclasstable=corinetable, class=0, method ="opt.ic")


# All options:
autoGLM <- function (data, reclasstable = "default", class=1, outputpath=paste(getwd(),"//", sep=""),
 modelname="autoGLM", tracelevel=1,
 actions = c("print", "return"), NAval = -9999,
 model="logit", preselect = "lm", method = "opt.ic", crit.t = 1.64, crit.p =.05,
 test = "LR", KLIC = "AICc", accuracytolerance =0.01, confidence.alternative =0.90,
 use.share = 0.25, maxsampleruns=50, memorymanagement = TRUE, returnall = FALSE,
 compress = FALSE, JIT = TRUE)
```

`corinetable` *A reclassification table for Corine land cover to a simplified aggregation used in the LUISA platform of the JRC.*

Description

A table that maps 51 Corine Land-cover classes into 8 classes and a NoData class.

- 0. Urban.
- 1. Industry.
- 2. Arable.
- 3. Permanent crops.
- 4. Pastures.
- 5. Forest.
- 6. Herbaceous cover.
- 16. Transitional.
- -9999. NoData.

Usage

```
data(corinetable)
```

`describe` *A simple function to describe a dataset.*

Description

This function returns a dataframe with variable names, their minimum and maximum values, means and standard deviations.

Usage

```
describe(X)
```

Arguments

`X`	Matrix or dataframe to be summarized.

Value

A dataframe with names and statistics.

Examples

```
someVector = 1:10
describe(someVector)

df <- data(ITdata)
description <- describe(df)
print(description)
```

`exportWeightsfile` *A function to export a csv file containing the weights file in a format that can be loaded into geoDMS applications.*

Description

This function exports a csv file containing the names of the weights, coefficients and country name in a standardized format supported by geoDMS.

Usage

```
exportWeightsfile(model, originaldata, modeldata, coefnamelist, outdir,
  modelname, filename)
```

Arguments

model	The model object from which the estimated coefficients should be extracted.
originaldata	The complete dataset from which the model data has been extracted. Used by the function to determine which variables have a zero weight.
modeldata	The dataset on which the model is estimated. Used by the function to determine which variables have a zero weight.
coefnamelist	A character vector containing the weight names that should be printed in the weights file.
outdir	The directory to which the weightsfile should be exported.
modelname	A string containing the name of the country that should be printed in the weightsfile.
filename	The name of the exported weightsfile.

Value

Any messages that may be printed by file(), writeLines() or close().

Examples

```
logitpars <- c(-0.5, 0.4, -0.3, 0.2, -0.1, 0, 0, 0, 0, 0, 0, 0)
someData <- simulateLogit(nobs=2500, pars=logitpars)
useData <- someData[,c(1:6)]
someModel <- logit(UseData)

exportWeightsfile(model = someModel, originaldata = someData, modeldata = useData,
coefnamelist = colnames(someData[-1,]), outdir = "C:\\Users\\",
modelname = "ImportantModel", filename = "important_result.csv")

data(ITdata)
data(corinetable)
ITsample <- getSamples(data = ITdata, share = 0.25, confidence.alternative=0.85, max.iter =100)
sampledIT = ITdata[ITsample,]
reclIT <- reclassify(sampledIT,  reclasstable=corinetable)
trainY <- MLtoBinomData(reclIT[,1], class=0)
trainX = reclIT[,-1]
bestX <- selectX(trainY, trainX, share = 0.25, returntype="colnames")
# total accessibility and domestic accessibility are multicollinear.
bestlogit <- logit(cbind(trainY,trainX[,bestX]))
exportWeightsfile(model = bestlogit, originaldata = ITdata, modeldata = cbind(trainY,trainX),
coefnamelist = colnames(trainX), outdir = "C:\\Users\\",
modelname = "IT", filename = "Urban_weights.csv")
```

---

generalizeToSpecific *A main function of the package to apply generalize to specific to generalized linear models.*

Description

This function is a wrapper to the functions opt.ic(), opt.t() and opt.h().

Usage

```
generalizeToSpecific(model = "lm", Y, X, method = "opt.ic", KLIC = "AICc",
  crit.t = 1.64, crit.p = 0.1, test = "LR", tracelevel = 1,
  memorymanagement = TRUE)
```

Arguments

model	Either "lm" for the linear probability model, "logit" for the logistic probability model, or "probit", for the probit model. The logit and probit models are solved using Iterated Weighted Least Squares, and optimization of the logit model is significantly faster than the probit model. Defaults to "lm".
Y	A binary response variable.
X	A dataframe of multiple exogenous regressors.
method	The optimization strategy. Either "opt.ic" to optimize using information criteria, "opt.t" for step-wise elimination of insignificant values (statistically speaking not a sound procedure, but it will provide a parsimonious model that can be usefull as a benchmark), or "opt.h" to optimize by classical hypothesis tests. defaults to "opt.ic".
KLIC	the information criterion used by "opt.ic", either "AIC" or "AICc", defaults to the latter.
crit.t	The t-value indicating significance when using method "opt.t", defaults to 1.64.
crit.p	the p-value used by method "opt.h" in the hypothesis tests. Defaults to 0.05.
test	The hypothesis test used by "opt.h". Defaults to "LR" for the Likelihood Ratio test. Other options are "F", for an F test for joint significance of insignificant parameters, or "Chisq" for a wald test against the Chi squared distribution. Recommended setting is either "LR" as it is less dependent on correct estimation of the standard errors. Keep in mind that "Chisq" is an asymptotic test, anf "F" is more appropiate for small sample tests. Howver "Chisq" holds under milder conditions and should be used if no small sample theory is available for the model.
tracelevel	the amount of information to be printed. Passed on to underlying routines. Defaults to 1 for printing, set to 0 for no printing.
memorymanagement	TRUE/FALSE indicating whether garbage collection should be forec regularly when memory usage is high. Defaults to TRUE, recommended setting for large datasets.
share	between 0-1, specifying the amount of data that should be passed on to the optimization strategies. Defaults to 0.75, to improve speed. Uses getSamples() to maintain first and second moments of the data.

Value

Either a dataframe of exogenous variables, or a vector containing the collumn names indicating the optimal variables extracted from the supplied dataset.

Examples

```
randomlogit <- simulateLogit(nobs=8000, pars = c(0.5, -0.4, -0.3, 0.1, 0.05, 0.025, 0.01,
                                                 0.005, 0.005, 0.005, 0.005, 0.005, 0.005,
                                           0.0025, 0.0025, 0.0025, 0.0025, 0.0, 0.0, 0.0))
```

```
# add multicollinear vector, to see how the method responds to faulty variables.
randomlogit<-cbind(randomlogit,mcv = randomlogit[,2])

Y=randomlogit[,1]
X=randomlogit[,-1]

logit_ic <- generalizeToSpecific(model ="logit", Y, X)

logit_t <- generalizeToSpecific(model ="logit", Y, X, "opt.t")

logit_h <- generalizeToSpecific(model ="logit", Y, X, "opt.h")

probit_ic <- generalizeToSpecific(model ="probit", Y, X)
linear_ic <- generalizeToSpecific(model ="probit", Y, X)
```

`getCall2` *Returns the name of a call as string.*

Description

When called inside a function, returns the parent function's name.

Usage

```
getCall2(level = -1)
```

Arguments

`level`	0 returns "getCall2", -1 returns the name of the function in which getCall2(-1) is called etc.

Value

string

Examples

```
getCall2(0)

foo <- function(){
return(getCall2())
}
foo()

bar <- function (somevar){
foo <- function(){
return(getCall2(-2))
}
return(foo())
}
```

`getSamples` *A function to extract small samples that maintain important characteristics of the population sample.*

Description

This function returns a sample extracted from the supplied population data, that has a similar distribution to the supplied population dataset. The function is called by guessStartval() to estimate inital values for numerical optimization procedures, but can also be used directly to reduce the sample size such that computationally intensive models can be estimated on a representative sample of an entire dataset. The function makes use of var.test() to compute an F test for the ratio of sample/population variance, and t.test() to compare their means.

Usage

```
getSamples(data, share = 0.25, confidence.alternative = 0.9,
  max.iter = 50, tracelevel = 1, memorymanagement = TRUE)
```

Arguments

`data`	The population data from which a sample needs to be taken.
`share`	The size of the sample in terms of the share of the population data. Defaults to .25.
`confidence.alternative`	The confidence level used in the F and t-tests defined as the probability level at which the alternative is accepted. For confidence.alternative = .9, we need less evidence to accept the alternative hypothesis that the samples are unequal than at confidence.alternative = .95, hence .90 is stricter than .95.
`max.iter`	The maximum number of draws to be taken. The programm breaks either when a suitable sample is found or when max.iter is reached.
`tracelevel`	Similar to a verbose statement. Should information be printed during execution? defaults to 1 for printing. set to 0 for no printing.
`memorymanagement`	TRUE/FALSE indicating whether garbage collection should be forced using tgc(). Defaults to TRUE. Recommended setting for large datasets.

Value

A sample of the population dataset that has significantly similar means and variances, or a message indicating that no suitable dat

Examples

```
getSamples (data = ITdata, share = 0.025, confidence.alternative=0.90, max.iter =100)
```

guessStartVal	*A function to efficiently obtain starting values for numerical optimization procedures. Used to initialize the "warm start" optimization routines in generalizeTospecific. The funciton itself uses a "warm start" algorithm over a growing dataset similar to a sieves estimator for an unbounded parameter space. When (quasi-)complete seperation is detected in subsamples, the starting values are returned as a vector of zeros. Only relevant when working with large datasets, (multiple times the size of the example data). Has some robustnes checks, returns a vector of zeros when the solution to the criterion is non-unique and the initial guess lands in a parameter regions of extreme values. Multicollinear values will be return with a parameter guess of 0.*

Description

This function is called by generalizeToSpecific(), but may also be called by users directly to obtain an initial gues of starting values to be passed on to easygmmlogit().

Usage

```
guessStartVal(Y, X, model = "logit", s1 = 0.25, s2 = s1, c1 = 0.85,
  c2 = c1, tracelevel = 1, memorymanagement = TRUE)
```

Arguments

Y	A binary response variable.
X	A dataset containing multiple exogenous regressors.
model	The model for which starting values should be estimated. Either "logit" or "probit" for the logit or probit model, or "gmm_nlminb" for a logit model estimated with gmm using PORT routines (reliable) or "gmm_bfgs" using the BFGS algorithm (fast, but still very slow compared to option "logit").
s1	Share of the sample used for the guess
s2	share of the subsample used to initialize the guess. If s2 =0.25, s1 =0.25, the guess is initialized at a .05 share of the entire dataset, or .25 of s1*datasize.
c1	confidence of first sample, see getSamples.
c2	confidence of subsample, see getSamples. #' @param tracelevel Whether information should be printed during execution. Defaults to 1 for printing, set to 0 for no printing.
memorymanagement	TRUE/FALSE indicating whether garbage collection should be forec regularly when memory usage is high. Defaults to TRUE, recommended setting for large datasets.

Value

A vector of coefficients that can be passed on to numerical optimization algorithms.

Examples

```
set.seed(234)

randomlogit <- simulateLogit(nobs=50000, pars = c(0.25, -0.2, -0.3, 0.1, 0.05, 0.025, 0.01,
                             0.005, 0.005, 0.005,0.005,0.0025,
                             0.0025,0.0025,0.0025,0,0,0,0,0,0))

Y=randomlogit[,1]
X=randomlogit[,-1]



# i5 4570 @ 3.2 GHz
system.time(guessStartVal(Y, X, model="logit"))
# user  system elapsed
0.29    0.00    0.30
system.time(logit(cbind(Y,X)))
# user  system elapsed
0.79    0.00    0.80
# the IWLS algorithm used for glm is already quite fast.

system.time(guessStartVal(Y, X, model="gmm_nlminb"))
# user  system elapsed
40.78    4.17    45.35
system.time(logit(cbind(Y,X), method ="gmm"))
# user  system elapsed
179.52   27.55  207.76
# the difference for gmm is quite large. It pays to do:
system.time(logit(cbind(Y,X), method ="gmm", start=guessStartVal(Y, X, model="logit")))
# user  system elapsed
160.55   21.79  182.48
```

`iapply` *lapply without memory builtup. Iterative application of a function over a vector of arguments, returning only the last result as an object or writing all output to disk.*

Description

Similar to well-know lapply, but returns only the last result as an object. Very usefull when working with big datasets. If the object to be returned is large in memory, say it contains copies of a large dataset, iapply iteratively applies a function but returns only the last output of the function application. This is useful if the function prints diagnostics to screen or saves results in a log file, but you would like to evaluate the last object to understand the output better. Additionally, the function allows you to write all the output to disk using a Serialization Interface for Single Objects. This allows you to restore any output to an object, possibly with a different name.

Usage

```
iapply(X, FUN, writedisk = FALSE, outdir = "default", name = "default",
  compress = FALSE)
```

Arguments

X	a vector of input variables similar to lapply.
FUN	a function to be applied iteratively over the input arguments. similar to FUN in lapply.
writedisk	TRUE/FALSE indicating whether output of application of FUN to elements of X should be written to disk as serialized represention in RDS files before it gets overwritten in memory. Will also write the final output to disk.
outdir	output directory where .RDS files should be written.
name	When writedisk = TRUE, output is saved with this name followed by the iteration. By default, uses getCall2(), thus RDS files are "iapply" followed by the number of th element. However storing iapply in another object, will change the output name.
compress	a logical specifying whether outputted RDS files use "gzip" compress, or one of "gzip", "bzip2" or "xz" to indicate the type of compress to be used. Defaults to no compress. You can later restore the objects using readRDS().

Value

the output of FUN(X[X[length(X)]], plus any prints, messages, warnings, errors that lapply(X, FUN) would produce. If FUN writes results to a disk, these files will be created too.

Examples

```
f <- function(x){
print (x)
return(x)
}
x=1:10
# this will print all elements of x, finaloutput will only contain the last element of x.
finaloutput <- iapply (x, f)
print(finaloutput)

# this will print all elements of x, create 10 RDS files that you can use to restore f(x),
# finaloutput will contain the last element of x.
finaloutput <- iapply (x, f, writedisk = TRUE, outdir = "C:\\Users\\")
print(finaloutput)

# this will do the same, but stores names as "binomials1", "binomials2" etc.
binomials <- iapply
finaloutput <- binomials (x, f, writedisk = TRUE, outdir = "C:\\Users\\")
print(finaloutput)
```

IC *A simple function to extract information criteria from a model object.*

Description

This function returns a Kullback-Leibler Information Criterion.

Usage

```
IC(model, KLIC = "AICc", sigma.is.estimated = FALSE)
```

Arguments

model	A model object that is supported by AIC(). For example produced by lm(), glm(), or speedglm().
KLIC	either "AIC" for the AIC, or "AICc" for the corrected criterion. defaults to "AICc".
sigma.is.estimated	boolean indicating whether variance has been estimated as part of the parameters (Likelihood), or not (LS), If FALSE, number of parameters is increased by 1 for comparison with MLE objects. Defaults to FALSE. Should not impact the results when models of the same type are compared, but the option has been added to allow users to change the number of counted parameters when comparing MLE results with LS results.

Value

numeric value.

Examples

```
somepars <- c(-0.5, 0.4, -0.3, 0.2, -0.1, 0, 0, 0)
someData <- simulateLogit(nobs=2500, pars=somepars)
someModel <- logit(someData)
IC(someModel, KLIC="AIC")
AIC(someModel) # compare
IC(someModel) # AICc
```

---

is.speed *Miscelaneous function used in main routines.*

Description

Miscelaneous function used in main routines.

Usage

```
is.speed(model)
```

Arguments

model	a modelobject.

Value

Boolean indicating whether supplied object is of class speedglm from package speedglm.

Examples

```
y=c(1,0,1,0,1,0)
x=c(0,0,1,1,1,0)
is.speed(lm(y~x))
```

ITdata — *A sample dataset produced with getSamples with 157481 obersvations representing 100x100 meter resolution grid cells of Corine land cover data for Italy with 26 exogenous variables. Variable names starting with a "w" indicate spatial lags.*

Description

To inspect the data, describe(data(ITdata))

Usage

```
data(ITdata)
```

Format

a data frame with 157481 rows and 27 columns

logistic — *The logistic function.*

Description

Called by various routines in the package. May also be used to predict fitted conditional probabilities by supplying a set of variables and corresponding coefficients estimated from a logit model.

Usage

```
logistic(theta, data)
```

Arguments

theta	A vector of coefficients.
data	A dataframe of multiple exogenous regressors.

Value

A vector of values produced by a logistic formula under specified parameters and data.

Examples

```
logitdata <- simulateLogit(1000, c(1,0.5,-0.5,-0.3))
model <- logit(logitdata)
pars <- coef(model)
# predict with logistic function
predicted <- logistic(pars, cbind(1,logitdata[,-1]))
# compare with data
describe(logitdata[,1])
# compare with predict function from R
describe(fitted(model))
```

logit *Simple wrapper to estimate a binomial logit model by IWLS or GMM.*

Description

This function returns an estimated logit model.

Usage

```
logit(Z, method = "IWLS", start = "default", maximizer = "nlminb",
  wmatrix = "ident", gmmtype = "iterative")
```

Arguments

Z	dataset with dependent variable in the first column and explanatory variables in other columns.
method	either IWLS to estimate a glm by Iterated Weighted Least Squares, or GMM to estimate with method of moments.
start	starting values, defaults to vector of zeros for IWLS and guessStartVal for gmm.
maximizer,	maximizer used for GMM, either "nlminb" for port routines or "BFGS".
wmatrix	Defaults to "ident" for identity matrix weighting of the gmm estimator. Other option is "optimal" for inverse covariance weighting. See package "gmm" for details.
gmmtype,	Either "twostep" for the two step GMM proposed by Hansen(1982), and the "cue" and "iterative" for respectively the continuously and the iteratively updated methods proposed by Hansen, Eaton et Yaron (1996). See package "gmm" for details. Defaults to "iterative".

Value

A logit model fitted by glm()

Examples

```
set.seed(10101)
Z <- simulateLogit(nobs=1000, pars =c(0.25,-0.5, 0.1, -0.1, 0, 0))

# Iterated Weighted Least Squares
test<- logit(Z)
# GMM using port routines and the results of the IWLS as starting values
test2<- logit(Z, method="gmm", start=coef(test))
# GMM using BFGS and starting at correct parameters
test3<- logit(Z, method="gmm", maximizer="BFGS", start =c(0, 0.25,-0.5, 0.1, -0.1, 0, 0))
test4<- logit(Z, method="gmm")
```

MLtoBinomData — *Miscelaneous function used in main routines.*

Description

Miscelaneous function used in main routines.

Usage

```
MLtoBinomData(data, class)
```

Arguments

data	factorial or numeric data of a categorial nature.
class	the category that should be reclassified into 1, all other values return as 0.

Value

a binary dataset.

Examples

```
y = c(1,2,3,4,1,2,3,4,1,2,3,4)
```

normalize — *A simple function to normalize a dataset of variables to the 0-1 interval.*

Description

This function returns a dataframe with normalized variables.

Usage

```
normalize(X)
```

Arguments

X	Matrix or dataframe to be normalized.

Value

A dataframe with normalized values ranging from 0 to 1.

Examples

```
someVector = 1:10
normalize(someVector)
df <- data(ITdata)
normalizedIT <- normalize(df)
describe(normalizedIT)
```

opt.glm *Core function used by autoGLM, for details and usage see autoGLM.*

Description

Core function used by autoGLM, for details and usage see autoGLM.

Usage

```
opt.glm(data, reclasstable = "default", class = 1,
  outputpath = paste(getwd(), "//", sep = ""), modelname = "autoGLM",
  tracelevel = 1, actions = c("print", "return"), NAval = -9999,
  model = "logit", preselect = "lm", method = "opt.ic", crit.t = 1.64,
  crit.p = 0.1, test = "LR", KLIC = "AICc", accuracytolerance = 0.025,
  confidence.alternative = 0.85, use.share = 0.25, maxsampleruns = 50,
  memorymanagement = TRUE)
```

Arguments

data	See also autoGLM.
reclasstable	autoGLM.
class	autoGLM.
outputpath	autoGLM.
modelname	autoGLM.
tracelevel	autoGLM.
actions	autoGLM.
NAval	autoGLM.
model	autoGLM.
preselect	autoGLM.
method	autoGLM.
crit.t	autoGLM.
crit.p	autoGLM.
test	autoGLM.
KLIC	autoGLM.
accuracytolerance	autoGLM.
confidence.alternative	autoGLM.
use.share	autoGLM.
maxsampleruns	autoGLM.
memorymanagement	autoGLM.

Value

a model object.

Examples

```
randomlogit <- simulateLogit(nobs=5000, pars = c(0.5, -0.4, -0.3, 0.1, 0, 0, 0, 0, 0, 0))
opt.glm(randomlogit, outputpath="C:\\users\\")
```

---

opt.h *Optimization routine based on hypothesis testing.*

Description

This function uses hypothesis tests to optimize a glm. If method is "joint", a joint significance tests of the full model against the model with only paramaeters that are significant in the full model is performed. If the insignificant variables are jointly significant, the alternative specification is returned. If the insignificant variables are jointly insignificant, the parameter with the largest p-value is dropped, and the test is repeated. If method is "single", the hypothesis tests are made between the full model and a model with one variable less. The tests are repeated till removing variables is not supported by the specified test. Single wald tests are equal to t-tests, F-tests are finite sample tests. Note that optimizing by single parameter tests is restrictive, for a discussion look up bonferonni corrections. Only the joint tests are available through autoGLM, for single tests opt.t is available. Though both methods use hypothesis testing a a decision criterion, the returned models may differ. This is a common result of hypothesis tests, and is one argument why model selection may be based on information criteria. for a discussion see for example "Comments on testing economic theories and the use of model selection criteria" by Granger, King and White (1995).

Usage

```
opt.h(model, Y, X, returntype = "model", tracelevel = 0, crit.p = 0.1,
  test = "LR", method = "joint", memorymanagement = FALSE)
```

Arguments

model	The model to be optimized. Supports "lm" for the linear probability model, "logit" for the logistic probability model, and "probit" for the probit model.
Y	The binary response variable.
X	A dataframe with collumns of exogenous regressors.
returntype	"model", "data", or "colnames"
tracelevel	level of printing.
crit.p	p value used in hypothesis tests.
test	type of test, either "LR", "F", or "Chisq".
memorymanagement	logical, indicating whether memory should be more actively managed.

Value

"model", "data", or "colnames", to be specified in returnype.

Examples

```
pars = c(0.5, -0.4, -0.3, 0.1, 0.05, 0.025, 0, 0, 0, 0,0,0,0)
randomlogit <- simulateLogit(nobs=8000, pars = pars)
Y=randomlogit[,1]
X=randomlogit[,-1]
hmod <- opt.h(model="logit", Y, X, returntype="model", tracelevel=1, crit.p=0.05, test="LR")
```

`opt.ic` *Optimization routine based on information criteria.*

Description

This function optimizes a model using information critera as a decision rule. This solves many of the problems related to model selection based on hypothesis tests, see also `opt.h`. In finite samples, the AIC is known to favor large models. The corrected AIC is a slightly stricter measure.

Usage

```
opt.ic(model, Y, X, KLIC = "AICc", returntype = "model", tracelevel = 0,
  memorymanagement = FALSE)
```

Arguments

`model`	The model to be optimized. Supports "lm" for the linear probability model, "logit" for the logistic probability model, and "probit" for the probit model.
`Y`	The binary response variable.
`X`	A dataframe with collumns of exogenous regressors.
`KLIC`	information criterion to be used, "AIC" or "AICc", See also `IC`
`returntype`	"model", "data", or "colnames"
`tracelevel`	level of printing.
`memorymanagement`	logical, indicating whether memory should be more actively managed.

Value

"model", "data", or "colnames", to be specified in returnype.

Examples

```
randomlogit <- simulateLogit(nobs=500, pars = c(0.5, -0.4, -0.3, 0.1, 0.05, 0.025, 0, 0, 0, 0))
Y=randomlogit[,1]
X=randomlogit[,-1]
opt.ic(model="lm", Y, X)
```

opt.t *Optimization routine based on step-wise elimenation using t-values.*

Description

This function searches

Usage

```
opt.t(model, Y, X, returntype = "model", tracelevel = 0, crit.t = 1.64,
  memorymanagement = FALSE)
```

Arguments

model	The model to be optimized. Supports "lm" for the linear probability model, "logit" for the logistic probability model, and "probit" for the probit model.
Y	The binary response variable.
X	A dataframe with collumns of exogenous regressors.
returntype	"model", "data", or "colnames"
tracelevel	level of printing.
crit.t	t-value used for significance test.
memorymanagement	logical, indicating whether memory should be more actively managed.

Value

"model", "data", or "colnames", to be specified in returnype.

Examples

```
randomlogit <- simulateLogit(nobs=500, pars = c(0.5, -0.4, -0.3, 0.1, 0.05, 0.025, 0, 0, 0, 0))
Y=randomlogit[,1]
X=randomlogit[,-1]
opt.t(model="lm", Y, X)
```

pkgTest *An function to load and install packages.*

Description

This function checks if a package is installed before loading it, and pulls it from the nearest CRAN server if the package is missing.

Usage

```
pkgTest(package, silent = FALSE)
```

Arguments

package	Character string indicating th package to be loaded. Accepts a vector of packages to load multiple packages.
silent	Should startup messages printed by the packages be surpressed? Will not surpress warnings or errors. Defaults to FALSE.

Value

The messages printed by the loaded package, or a message if the package is not found.

Examples

```
verbose_packages <- c("gmm", "foreign", "sp")
silent_packages <- c("data.table", "compiler", "speedglm")

pkgTest(verbose_packages)
pkgTest(silent_packages)
```

PtoBin *Miscelaneous function used in main routines.*

Description

Miscelaneous function used in main routines.

Usage

```
PtoBin(Pmap)
```

Arguments

Pmap	a vector of probabilities.

Value

A binary vector indicating occurrences.

Examples

```
probabilities<-c(1:100)/100
occurrences<-PtoBin(probabilities)
```

reclassify *A function for reclassification of integer data.*

Description

This function works similar to ArcMap's raster reclassify function, only fast and without starting hell on earth.

Usage

```
reclassify(LUdata, reclasstable = "default", JIT = TRUE, dropNA = TRUE,
  NAval = "default")
```

Arguments

LUdata	A dataset (vector, matrix or dataframe) that contains data to be reclassified, e.g., CORINE land cover data. If an object with multiple collumns is supplied, the function will reclassify only the first collumn.
reclasstable	the path to a .csv file containing the reclass table. By default, the function loads "corinetable", supplied with the package. This table maps Corine land-cover into the Classification scheme used in the LUISA framework of the Joint European Research centre. To inspect the table: data(corinetable); describe(corinetable).
JIT	a boolean indicating whether Just in Time compilation should be used. Defaults to TRUE.
dropNA	TRUE/FALSE indicating whether values that are reclassified into NoData should be dropped from the output. By default TRUE.
NAval	the value in the reclasstable that corresponds to noData. By default the function assumes that the classes have positive values, and that that the lowest negative value corresponds to the class that should be dropped (e.g., if your relcass table maps 1,2,3 into 1,2,-9999, the function will return a dataset with classes 1,2., and treat 3 as NoData, dropping it unless specified to keep NoData in the output).

Value

A vector of reclassified values, or matrix with the firs column being reclassified.

Examples

```
A=c(1,2,3,4,5,6,7,8,9)
B=c(9,8,7,6,5,4,3,2,1)
reclasstable=cbind(A,B)

reclassify(A, reclasstable)
# by default, the data is not reclassified
reclassify(A)

# however, if dropNA is TRUE, thge default NA value is dropped (the most negative value)
C = c(1,2,3,4,5,6,7,8,9,-9999)
reclassify(C)
```

```
data(ITdata)
data(corinetable)


# reclassify a vector of land-use data using the corine_to_LUISA scheme:
landcover <- reclassify(LUdata=ITdata[,1], reclasstable=corinetable, JIT=TRUE)

#reclassify a multi-collumn dataset in which the first collumn represents factorial data:
reclass_IT <- reclassify(LUdata=ITdata, reclasstable=corinetable, JIT=TRUE)



# the LUISA codes are a simplification and drop certain land cover classes. By default,
# the categories reclassified into the lowest negative values, are dropped.
# If you wish to keep all categories:
reclass_IT <- reclassify(LUdata=ITdata, reclasstable=corinetable, dropNA=FALSE)
```

`selectX` *A main function of the package for variable selection based on moidel type and a generalize to specific approach.*

Description

This function is a wrapper to the functions bestlinearX(), bestlogitX() and bestprobitX(), with an additional option to call getSamples for improved speed. Take into account that sampling itself takes time, such that total computational burden is a trade-off between the load of the getSample function and the model optimization itself.

Usage

```
selectX(Y, X, model = "lm", returntype = "data", method = "opt.ic",
  KLIC = "AICc", crit.t = 1.64, crit.p = 0.05, test = "LR",
  share = 0.75, confidence.alternative = 0.85, max.iter = 50,
  tracelevel = 1, memorymanagement = TRUE)
```

Arguments

`Y`	A binary response variable.
`X`	A dataframe of multiple exogenous regressors.
`model`	Either "lm" for the linear probability model, "logit" for the logistic probability model, or "probit", for the probit model. The logit and probit models are solved using Iterated Weighted Least Squares, and optimization of the logit model is significantly faster than the probit model. Defaults to "lm".
`returntype`	Either "data" to return a dataset, or colnames" to only return the collumn names of the variables that are used in the optimal model. "data" by default.
`method`	The optimization strategy. Either "opt.ic" to optimize using information criteria, "opt.t" for step-wise elimination of insignificant values (statistically speaking not a sound procedure, but it will provide a parsimonious model that can be usefull as a benchmark), or "opt.h" to optimize by classical hypothesis tests. defaults to "opt.ic".

KLIC — the information criterion used by "opt.ic", either "AIC" or "AICc", defaults to the latter.

crit.t — The t-value indicating significance when using method "opt.t", defaults to 1.64.

crit.p — the p-value used by method "opt.h" in the hypothesis tests. Defaults to 0.05.

test — The hypothesis test used by "opt.h". Defaults to "LR" for the Likelihood Ratio test. Other options are "F", for an F test for joint significance of insignificant parameters, or "Chisq" for a wald test against the Chi squared distribution. Recommended setting is either "LR" as it is less dependent on correct estimation of the standard errors. Keep in mind that "Chisq" is an asymptotic test, anf "F" is more appropiate for small sample tests. Howver "Chisq" holds under milder conditions and should be used if no small sample theory is available for the model.

share — between 0-1, specifying the amount of data that should be passed on to the optimization strategies. Defaults to 0.75, to improve speed. Uses getSamples() to maintain first and second moments of the data.

confidence.alternative — passed on to getSample. Defaults to .85.

max.iter — passed on to getSample. Defaults to 50.

tracelevel — the amount of information to be printed. Passed on to underlying routines. Defaults to 1 for printing, set to 0 for no printing.

memorymanagement — TRUE/FALSE indicating whether garbage collection should be forec regularly when memory usage is high. Defaults to TRUE, recommended setting for large datasets.

Value

Either a dataframe of exogenous variables, or a vector containing the collumn names indicating the optimal variables extracted from the supplied dataset.

Examples

```
# load data
data(ITdata)
data(corinetable)
#Grab a sample (optional).
sample <- ITdata[getSamples(ITdata, share =.05),]
# Reclassify
catITdata <- reclassify(sample, reclasstable = corinetable)
# create a binary response dataset.
Y <- MLtoBinomData(catITdata[,1], class =1)
X <- catITdata[,-1]
selectX(Y, X, model ="lm", returntype = "colnames", method = "opt.t")
bestX <- selectX(Y, X)
describe(bestX)
```

simulateLogit *A function to generate random data from a logit model.*

Description

This function generates random data simulated from a logit model based on the parameters supplied.

Usage

```
simulateLogit(nobs = 2500, pars = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
```

Arguments

nobs	number of observations that the returned dataset should include. Defaults to 2500.
pars	A vector of parameters to be used in the data generating process. Regressors may have a 0 coefficient. Defaults to 0 for all variables.

Value

A simulated dataset from a logistic probability model with nobs observations, a binary response variable, and nxregs regressors ranging from 0 to 100.

Examples

```
logitpars <- c(-0.5, 0.4, -0.3, 0.2, -0.1, 0, 0, 0, 0, 0, 0, 0)
logit_data <- simulateLogit(nobs=2500, pars=logitpars)

random_binaryResponse_data <- simulateLogit(nobs=500)
```

tgc *A function for memory management.*

Description

A call to tgc() forces a garbage collection if memory usage is above a certain threshold, by default when approximately half of the available mememory is in use by R. Calling tgc(0) performs a standard garbage collection and is therefore the same as gc() with the additional option to delete the output of gc() from memory, such that the call is entirely silent. The function makes a request to the OS (windows only support) to retun memory usage statistics, if this call fails garbage collection will be forced even if memory usage is below the supplied threshold.

Usage

```
tgc(maxmemshare = 0.5, verbose = FALSE, emptythrash = TRUE)
```

Arguments

maxmemshare	Numeric between 0 and 1 indicating the threshold share of memory usage that triggers garbage collection.
verbose	Should diagnostics be printed? FALSE by default.
emptythrash	Should the object that containsthe output from the garbage collection, including messages, be deleted from memory? TRUE by default. If TRUE, tgc() is silent.

Value

Prints diagnostics of the garbage collection, or silent when emptytrash is TRUE.

Examples

```
tgc()

tgc(maxmemshare=0, verbose = FALSE, emptytrash = TRUE)

tgc(0, TRUE, FALSE)
```

---

withinInterval *A function that determines whether a point lies in a specified interval.*

Description

A function that determines whether a point lies in a specified interval.

Usage

```
withinInterval(x, interval)
```

Arguments

x	numeric.
interval	vector of length 2 specifying the outer values of the interval.

Value

TRUE/FALSE

Examples

```
withinInterval(5,c(1,10))
```

Index