Package 'optMaxlik'

April 3, 2017

Type Package

Title What the Package Does (Title Case)

Version 0.1.0

Description Extension to the maxLik package for numerical minimzation of Kullback-Leibler divergence. Applies a general to specific approach to custom Likelihood functions.

Depends R (>= 3.2.1),
maxLik(>= 1.3.0)

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

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

License MIT

ByteCompile TRUE

LazyData TRUE

RoxygenNote 5.0.1

Imports maxLik

R topics documented:

aicc *Corrected AIC*

Description

Compute the corrected AIC from a maxLik object.

Usage

```
aicc(x, n = Inf, penalty = 0)
```

Arguments

x	maxLik object
n	number of observations for correction. Defaults to Inf, which coincides with the AIC.
penalty	numeric that should be added to the log likelihood to adjust when using penalized likelihood or regularized estimators.

Value

numeric

areFixed *Mainly auxiliary. Returns a vector indicating which parameters are fixed at zero.*

Description

Returns a vector indicating which parameters are fixed at zero.

Usage

```
areFixed(x)
```

Arguments

x	vector

Value

vector

as.numeric.matrix *Mainly auxiliary. Same as as.numeric() but for matrices.*

Description

Function to return a matrix with numericals.

Usage

```
## S3 method for class 'matrix'
as.numeric(mat)
```

Arguments

mat	matrix

Value

matrix

breads *Get the breads to build standard errors.*

Description

Computes the inverse Fischer Information matrix

Usage

```
breads(model)
```

Arguments

model	maxLik object

Value

matrix

dropna *Drop NA values from a vector.*

Description

Removes NA values from a vector.

Usage

```
dropna(x)
```

Arguments

x	vector

Value

vector

freePars *Get number of unrestricted parameters from a maxLik object.*

Description

Get number of unrestricted parameters from a maxLik object.

Usage

```
freePars(x)
```

Arguments

x	maxLik object,

Value

numeric

insert.at — *Mainly auxiliary. Inserts the content of an object into the supplied vector.*

Description

Function to insert values into a vector.

Usage

```
insert.at(a, pos, ...)
```

Arguments

a	vector in which to insert a value
pos	numeric indicating the position where to insert a value.
...	objects to be inserted.

Value

vector

Examples

```
a <- 1:10
insert.at(a, 5, 0)
```

opt.maxLik — *Main function of the package. Automated general to specific approach for custom Likelihood functions.*

Description

Numerical minimization of AICc by optimization of a likelihood function and automated constraining of parameters.

Usage

```
opt.maxLik(LL, start, initialfix = numeric(), nobs = Inf, method = "BFGS",
  penalized = numeric(), pw = 2)
```

Arguments

LL	a log likelihood function.
start	named vector of starting values.
initialfix	vector indicating the parameters that should be treated as constants with values supplied in the start vector.
nobs	number of observations to be used in the corrction of the AIC(c), defaults to Inf.

method	numerical algorithm. See maxLik package. Defaults to BFGS
penalized	vector indicating which parameters are penalized in your likelihood function. Will be used to compute adjustments for the AIC(c).
pw	the weight of you penalty. Supported penalties are abs(parameter value)^pw.

Value

maxLik object of final model.

Examples

```
library(maxLik)
library(compiler)
# Fill a matrix with some random data.
mydata<-matrix(rnorm(1000), ncol=10, nrow=100)

# Create you own Log likelihood function.
crossectionalARMA_44 <- function(pars, ret="LL", Y){
 data=Y
 Log.L <-numeric()[1:T]
 b0    <-pars[1]

 B.y   <-pars[2]
 B.y2  <-pars[3]
 B.y3  <-pars[4]
 B.y4  <-pars[5]

 B.e   <-pars[6]
 B.e2  <-pars[7]
 B.e3  <-pars[8]
 B.e4  <-pars[9]

 nu    <-pars[10]
 sigma <-pars[11]

 df = as.numeric.matrix(data)
 T=nrow(df)
 N=ncol(df)

 p=4

 e = matrix(0,T,N)
 e[1:p,] <- 0 ## Initialize with e1 = 0

 Log.L[1:p]<-0

 A =N*log((gamma((nu+1)/2))/(((pi*(nu-2))^0.5)*gamma(nu/2))) - 0.5*N*log(max(0,sigma)^2)
 A2=((nu+1)/2)
 A3=(max(0,sigma)^2*(nu-2))

 for (t in (p+1):T) {
  y = df[t,]#as.numeric(matrix(c(df[t,])))
  ymin =  df[t-1,]#as.numeric(matrix(c(df[t-1,])))
  ymin2 =  df[t-2,]#as.numeric(matrix(c(df[t-2,])))
  ymin3 =  df[t-3,]#as.numeric(matrix(c(df[t-3,])))
  ymin4 =  df[t-4,]#as.numeric(matrix(c(df[t-4,])))
```

```
  emin=e[t-1,]
  emin2=e[t-2,]
  emin3=e[t-3,]
  emin4=e[t-4,]

   MA = B.e*emin + B.e2*emin2 + B.e3*emin3 + B.e4*emin4
  e[t,] = y - b0 - B.y*ymin - B.y2*ymin2 - B.y3*ymin3 - B.y4*ymin4 - MA

  }


 Log.L <- A - A2*(rowSums(log(1+ (e)^2 / A3)))
 Log.L[1:p]<-0

 if(ret=="e"){return(e)}else if (ret =="LLvec") {return(Log.L)} else(sum(Log.L))

}

# Optionally compile your function

crossectionalARMA_44<-cmpfun(crossectionalARMA_44)

# opt.Maxlik takes only functions that have only a parameter vector as input.
# fix the data argument.

LL <- function(pars){crossectionalARMA_44(pars, ret="LL", Y=mydata)}

# create a vector of names parameter starting values.

start =c(b0=0, b1=0, b2=0, b3=0,b4=0, ma1=0, ma2=0, ma3=0, ma4=0, nu=120, sigma=sd(mydata))

# t-estimation
results <- opt.maxLik (LL=LL, start=start,initialfix=c(), nobs=dim(mydata)[1]*dim(mydata)[2])
summary(results)

estimates=coef(results)

parsToOriginal(estimates, start)

# approximate normal estiamtion by fixing the degrees of freedom
results2 <- opt.maxLik (LL=LL, start=start,initialfix=c(10), nobs=dim(mydata)[1]*dim(mydata)[2])
summary(results2)

estimates2=coef(results2)

parsToOriginal(estimates2, start)
parsToOriginal2(estimates2, start)
```

`parsToOriginal` *Mainly auxiliary. Function to convert final parameters back into the format of the starting parameter vectors.*

Description

Function to convert final parameters back into the format of the starting parameter vectors. Sets parameters that are contstrained in the final results to zero.

Usage

```
parsToOriginal(results, start)
```

Arguments

results	vector of parameters
start	vector of the original starting values.

Value

vector

`parsToOriginal2` *Mainly auxiliary. Function to convert final parameters back into the format of the starting parameter vectors.*

Description

Function to convert final parameters back into the format of the starting parameter vectors. Sets parameters that are contstrained in the final results to supplied starting values.

Usage

```
parsToOriginal2(results, start)
```

Arguments

results	vector of parameters
start	vector of the original starting values.

Value

vector

`se` *Get standard errors.*

Description

Computes the standard errors from the Hessian.

Usage

```
se(model)
```

Arguments

`model`	maxLik object

Value

matrix

Index