

# Package ‘likelihoodAsy’

November 2, 2016

**Version** 0.45

**Priority** optional

**Title** Functions for Likelihood Asymptotics

**Author** Ruggero Bellio and Donald Pierce

**Maintainer** Ruggero Bellio <ruggero.bellio@uniud.it>

**Depends** nleqslv, Rsolnp, alabama, pracma, cond

**Description** Functions for computing the  $r$  and  $r^*$  statistics for inference on an arbitrary scalar function of model parameters, plus some code for the (modified) profile likelihood.

**Suggests** MASS, lme4, knitr

**License** GPL (>= 2)

**NeedsCompilation** no

**LazyLoad** yes

**VignetteBuilder** knitr

**Repository** CRAN

**Date/Publication** 2016-11-02 17:57:20

## R topics documented:

likelihoodAsy-package . . . . .	2
finndat . . . . .	2
logMPL . . . . .	3
logPL . . . . .	5
rstar . . . . .	8
rstar.ci . . . . .	11
rstar.ci.control . . . . .	15
<b>Index</b>	<b>17</b>

---

likelihoodAsy-package    *The R likelihoodAsy Package*

---

### Description

Some functions for likelihood asymptotics, based on the expository paper

Pierce, D.A. and Bellio, R. (2016) Modern likelihood-frequentist inference. Submitted for publication.

### Details

For a detailed introduction see the vignette (`browseVignettes("likelihoodAsy")`), which includes an R script that carries out the examples.

### Author(s)

Author: Ruggero Bellio and Donald A Pierce

Maintainer: Ruggero Bellio <ruggero.bellio@uniud.it>

---

finndat                      *Data from Finney (1947)*

---

### Description

Binomial data from a bioassay.

### Format

A data frame with 10 observations on the following 3 variables.

`z` a numeric vector of doses.

`den` a numeric vector of binomial denominators.

`y` binomial responses.

### Source

Finney, D.J. (1947). *Probit Analysis: A Statistical Treatment of the Sigmoid Response Curve*. Cambridge University Press, London and New York.

---

logMPL	<i>Modified profile likelihood computation</i>
--------	--

---

### Description

This function evaluates the Modified Profile Likelihood (MPL) for a subset of the model parameter. The result is optionally returned with a minus sign, so the function can be used directly as input to a general-purpose optimizer.

### Usage

```
logMPL(psival, data, mle, floglik, fscore=NULL, indpsi, datagen, R=500, seed=NULL,
       minus=FALSE, onestep=FALSE, jhat=NULL, trace=FALSE)
```

### Arguments

psival	A numerical vector containing the value of the parameter of interest.
data	The data as a list. All the elements required to compute the likelihood function at a given parameter value should be included in this list. The required format of such list will be determined by the user-provided function floglik.
mle	A numerical vector, containing the maximum likelihood estimate of the entire model parameter.
floglik	A function which returns the log likelihood function at a given parameter value. In particular, for a certain parameter value contained in a numerical vector theta, a call floglik(theta, data) should return a scalar numerical value, the log likelihood function at theta. Note that the parameter of interest should be a subset of the coordinates of theta.
fscore	An optional function which returns the score function at a given parameter value. It must return a numerical vector of the same length of mle. For a certain parameter value contained in a numerical vector theta, a call fscore(theta, data) should return the gradient of the log likelihood function at theta. Default is NULL, implying that numerical differentiation will be employed.
indpsi	A vector of integers in the range 1:length(theta) containing the indexes of the parameter of interest, so that the parameter of interest will be given by theta[indpsi].
datagen	A function which simulates a data set. A call datagen(theta, data) will generate a copy of the data list, with the values of the response variable replaced by a set of values simulated from the parametric statistical model assumed for the response variable.
R	The number of Monte Carlo replicates used for computing the modified profile likelihood. A positive integer, default is 500.
seed	Optional positive integer, the random seed for the Monte Carlo computation. Default is NULL.
minus	Logical. Should the modified profile likelihood be multiplied by -1? This may be useful for usage with optimizers. Default is FALSE.

onestep	Logical. If set to TRUE the constrained estimate of the nuisance parameter is replaced by a one-step approximation around the maximum likelihood estimate. Default is FALSE.
jhat	A squared matrix with dimension equal to length(mle) containing the observed information matrix evaluated at mle. It is employed only when onestep=TRUE. Default is NULL.
trace	Logical. When set to TRUE will cause the printing of the MPL value, which can be useful to monitor optimization. Default is FALSE.

### Details

The function implements the Modified Profile Likelihood employing the approximation to sample space derivatives proposed in Skovgaard (1996). The function is designed to be used with external functions, such as optimizers and evaluators over a grid of points.

### Value

A scalar value, minus the modified profile likelihood at psival.

### References

- Severini, T.A. (2000). Likelihood Methods in Statistics. Oxford University Press.
- Skovgaard, I.M. (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

### Examples

```
# Approximating the conditional likelihood for logistic regression
# Let us define the various functions
# Log likelihood for logistic regression
loglik.logit<- function(theta, data)
{
  y <- data$y
  den <- data$den
  X <- data$X
  eta <- X %*% theta
  p <- plogis(eta)
  l <- sum(y * log(p) + (den - y) * log(1-p))
  return(l)
}
# Score function
grad.logit<- function(theta, data)
{
  y <- data$y
  den <- data$den
  X <- data$X
  eta <- X %*% theta
  p <- plogis(eta)
  out <- t(y - p * den) %*% X
  return(drop(out))
}
```

```

}
# Data generator
gendat.logit<- function(theta, data)
{
  X <- data$X
  eta <- X %*% theta
  p <- plogis(eta)
  out <- data
  out$y <- rbinom(length(data$y), size = data$den, prob = p)
  return(out)
}
# Famous crying babies data
data(babies)
mod.glm <- glm(formula = cbind(r1, r2) ~ day + lull - 1, family = binomial,
  data = babies)
data.obj <- list(y = babies$r1, den = babies$r1 + babies$r2,
  X = model.matrix(mod.glm))
# Numerical optimization of profile and modified profile log likelihoods
max.prof <- nlmnb(0, logPL, data=data.obj, thetainit=coef(mod.glm),
  floglik=loglik.logit, fscore=grad.logit, indpsi=19, minus=TRUE, trace=FALSE)
max.mpl <- nlmnb(0, logMPL, data=data.obj, mle=coef(mod.glm),
  floglik=loglik.logit, fscore=grad.logit, datagen=gendat.logit,
  indpsi=19, R=50, seed=2020, minus=TRUE, trace=FALSE)
c(max.prof$par, max.mpl$par)
# We can plot the profile likelihood and the modified profile likelihood
# R=50 suffices for the modified profile likelihood as the model is a full exp. family
psi.vals <- seq(-0.3, 3.7, l=20)
obj.prof <- sapply(psi.vals, logPL, data=data.obj, thetainit=coef(mod.glm),
  floglik=loglik.logit, fscore=grad.logit, indpsi=19, trace=FALSE)
obj.mpl <- sapply(psi.vals, logMPL, data=data.obj, mle=coef(mod.glm),
  floglik=loglik.logit, fscore=grad.logit, datagen=gendat.logit,
  indpsi=19, trace=FALSE, R=50, seed=2020)
par(pch="s")
plot(psi.vals, obj.prof - max(obj.prof), type="l", xlab=expression(psi),
  ylab="log likelihood", lwd=2, las=1)
lines(psi.vals, obj.mpl - max(obj.mpl), col="red", lwd=2)
legend("topright", col=c(1, 2), lty=1, lwd=2, legend=c("Profile", "MPL"), bty="n")

```

---

logPL

---

*Profile likelihood computation*


---

## Description

This function evaluates the profile likelihood for a subset of the model parameter. The result is optionally returned with a minus sign, so the function can be used directly as input to a general-purpose optimizer.

**Usage**

```
logPL(psival, data, thetainit, floglik, fscore=NULL, indpsi, minus=FALSE, onestep=FALSE,
      jhat=NULL, trace=FALSE)
```

**Arguments**

psival	A numerical vector containing the value of the parameter of interest.
data	The data as a list. All the elements required to compute the likelihood function at a given parameter value should be included in this list. The required format of such list will be determined by the user-provided function <code>floglik</code> .
thetainit	A numerical vector with the size of the entire model parameter, that will be used as starting point in the constrained optimization performed to obtain the maximum likelihood estimate under the null. The specific meaning of <code>thetainit</code> is determined by the specification of <code>floglik</code> .
floglik	A function which returns the log likelihood function at a given parameter value. In particular, for a certain parameter value contained in a numerical vector <code>theta</code> , a call <code>floglik(theta, data)</code> should return a scalar numerical value, the log likelihood function at <code>theta</code> . Note that the parameter of interest should be a subset of the coordinates of <code>theta</code> .
fscore	An optional function which returns the score function at a given parameter value. It must return a numerical vector of the same length as <code>thetainit</code> . For a certain parameter value contained in a numerical vector <code>theta</code> , a call <code>fscore(theta, data)</code> should return the gradient of the log likelihood function at <code>theta</code> . Default is <code>NULL</code> , implying that numerical differentiation will be employed.
indpsi	A vector of integers in the range <code>1:length(theta)</code> containing the indexes of the parameter of interest, so that the parameter of interest will be given by <code>theta[indpsi]</code> .
minus	Logical. Should the profile likelihood be multiplied by -1? This may be useful for usage with optimizers. Default is <code>FALSE</code> .
onestep	Logical. If set to <code>TRUE</code> the constrained estimate of the nuisance parameter is replaced by a one-step approximation around the maximum likelihood estimate. Default is <code>FALSE</code> .
jhat	A squared matrix with dimension equal to <code>length(mle)</code> containing the observed information matrix evaluated at <code>mle</code> . It is employed only when <code>onestep=TRUE</code> . Default is <code>NULL</code> .
trace	Logical. When set to <code>TRUE</code> will cause the printing of the MPL value, which can be useful to monitor optimization. Default is <code>FALSE</code> .

**Details**

This function is designed to be used with external functions, such as optimizers and evaluators over a grid of points.

**Value**

A scalar value, minus the profile likelihood at `psival`.

## References

Severini, T.A. (2000). Likelihood Methods in Statistics. Oxford University Press.

## Examples

```
# A negative binomial example, taken from Venables and Ripley (2002, MASS4 book)
library(MASS)
# The quine data are analysed in Section 7.4
data(quine)
# We fit a model with just the main effects
quine.nb1 <- glm.nb(Days ~ Eth + Sex + Age + Lrn, data = quine)
# The data list includes the design matrix and the response vector
quinedata<-list(X=model.matrix(quine.nb1), y=quine$Days)
# Let us define the various functions
# Log likelihood, log link
logLikNbin <- function(theta,data)
{
  y <- data$y
  X <- data$X
  eta <- X %%% theta[1:ncol(X)]
  mu <- exp(eta)
  alpha <- theta[ncol(X)+1]
  l <- sum(lgamma(y + alpha) + y * log(mu) - (alpha + y) * log(alpha + mu)
          - lgamma(alpha) + alpha * log(alpha))
  return(l)
}

# Score function
gradLikNbin <- function(theta,data)
{
  y <- data$y
  X <- data$X
  eta <- X %%% theta[1:ncol(X)]
  mu <- exp(eta)
  alpha <- theta[ncol(X)+1]
  g <- rep(0,ncol(X)+1)
  g[1:ncol(X)] <- t(y - (alpha+y)*mu / (alpha+mu)) %%% X
  g[ncol(X)+1] <- sum(digamma( y + alpha) - log(alpha + mu) - (alpha + y) / (alpha + mu)
                      - digamma(alpha) + 1 + log(alpha))
  return(g)
}

# Data generator
genDataNbin<- function(theta,data)
{
  out <- data
  X <- data$X
  eta<- X %%% theta[1:ncol(X)]
  mu <- exp(eta)
  out$y <- rnegbin(length(data$y), mu=mu, theta=theta[ncol(X)+1])
  return(out)
}
```

```
# First we refine the maximum likelihood estimates
mleFull <- optim( c(coef(quine.nb1),quine.nb1$theta), logLikNbin, gr=gradLikNbin,
                 method="BFGS", data=quinedata, control=list(fnscale=-1), hessian=TRUE)
# Then we can plot the profile likelihood
list.psi <- seq(0.90, 1.70, l=30)
list.prof <- sapply(list.psi, logPL, data=quinedata, thetainit=mleFull$par, floglik=logLikNbin,
                  fscore=gradLikNbin, indpsi=8, trace=FALSE)
plot(list.psi, list.prof-max(list.prof), type="l", xlab=expression(psi), ylab="Log likelihood")
```

---

rstar

---

*Inference on a scalar function of interest by the  $r^*$  statistic*


---

## Description

This function evaluates the  $r^*$  statistic for testing of a scalar function of interest.

## Usage

```
rstar(data, thetainit, floglik, fscore=NULL, fpsi, psival, datagen, R=1000, seed=NULL,
      trace=TRUE, ronly=FALSE, psidesc=NULL, constr.opt="solnp")
```

## Arguments

data	The data as a list. All the elements required to compute the log likelihood function at a given parameter value should be included in this list.
thetainit	A numerical vector containing the initial value for the parameter of the model. It will be used as starting point in the numerical optimization of the log likelihood function.
floglik	A function which returns the log likelihood function at a given parameter value. In particular, for a certain parameter value contained in a numerical vector theta, a call floglik(theta, data) should return a scalar numerical value, the log likelihood function at theta.
fscore	An optional function which returns the score function at a given parameter value. It must return a numerical vector of the same length of thetainit. For a certain parameter value contained in a numerical vector theta, a call fscore(theta, data) should return the gradient of the log likelihood function at theta. Default is NULL, implying that numerical differentiation will be employed.
fpsi	A function which specifies the parameter of interest. A call fpsi(theta) should return a scalar value.
psival	A numerical scalar value containing the value of the parameter of interest under testing.
datagen	A function which simulates a data set. A call datagen(theta, data) will generate a copy of the data list, with the values of the response variable replaced by a set of values simulated from the parametric statistical model assumed for the response variable.



R	The number of Monte Carlo replicates used for computing the $r^*$ statistic. A positive integer, default is 1000.
seed	Optional positive integer, the random seed for the Monte Carlo computation. Default is NULL.
trace	Logical. When set to TRUE will cause some information on the computation to be printed. Default is FALSE.
ronly	Logical. If set to TRUE the computation of the $r^*$ statistic will be skipped, and only the value of the signed likelihood ratio test statistic $r$ will be returned by the procedure, without any Monte Carlo computation. Default is FALSE.
psidesc	An optional character string describing the nature of the parameter of interest. Default is NULL.
constr.opt	Constrained optimizer used for maximizing the log likelihood function under the null hypothesis. Possible values are "solnp" or "alabama", with the former employing the solnp function from package Rsolnp and the latter the constrOptim.nl from the package alabama. Defaults is "solnp".

### Details

The function computes the  $r^*$  statistic proposed by Skovgaard (1996) for accurate computation of the asymptotic distribution of the signed likelihood ratio test for a scalar function of interest.

The function requires the user to provide three functions defining the log likelihood function, the scalar parametric function of interest, and a function for generating a data set from the assumed statistical model. A further function returning the gradient of the log likelihood is not required, but if provided it will speed up the computation.

When `ronly = TRUE` the function returns the value of the signed likelihood ratio test statistic  $r$  only.

The function handles also one-parameter models.

### Value

The returned value is an object of class "rstar", containing the following components:

r	The observed value the signed likelihood ratio test statistic $r$ for testing $f_{\psi}(\theta) = \psi$ .
NP, INF	The Nuisance Parameter adjustment (NP) and the Information adjustment (INF) from the decomposition of the $r^*$ - $r$ adjustment. The former is not computed for one-parameter models. Neither one is computed when <code>ronly = TRUE</code> .
rs	The observed value of the $r^*$ statistic. Not computed when <code>ronly = TRUE</code> .
theta.hat	The maximum likelihood estimate of the parameter $\theta$ , the argument of the <code>floglik</code> , <code>fscore</code> , <code>datagen</code> and <code>fpsi</code> functions.
info.hat	The observed information matrix evaluated at <code>theta.hat</code> . Not computed when <code>ronly = TRUE</code> .
se.theta.hat	The estimated standard error of <code>theta.hat</code> . Not computed when <code>ronly = TRUE</code> .
psi.hat	The parameter of interest evaluated at <code>theta.hat</code> .
se.psi.hat	The estimated standard error for the parameter of interest. Not computed when <code>ronly = TRUE</code> .

theta.hyp	The constrained estimate of the parameter, under the null hypothesis $f_{\psi}(\theta)=\psi_{\text{sival}}$ .
psi.hyp	The value under testing, equal to $\psi_{\text{sival}}$ .
seed	Random seed used for Monte Carlo trials. Not returned when <code>ronly = TRUE</code> .
psidesc	A character string describing the nature of the parameter of interest.
R	Number of Monte Carlo replicates used for computing the $r^*$ statistic. Not returned when <code>ronly = TRUE</code> .

There are print and summary methods for this class.

## References

The method implemented in this function was proposed in

Skovgaard, I.M. (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

For a general review

Severini, T.A. (2000). Likelihood Methods in Statistics. Oxford University Press.

## See Also

[rstar.ci](http://rstar.ci).

## Examples

```
# Autoregressive model of order 1
# We use the lh data from MASS
library(MASS)
data(lh)
dat.y <- list(y=as.numeric(lh))
# First let us define the function returning the log likelihood function
# We employ careful parameterizations for the correlation and variance to
# avoid numerical problems
likAR1 <- function(theta, data)
{
  y <- data$y
  mu <- theta[1]
  phi <- theta[2] ### phi is log(sigma)
  sigma2 <- exp(phi*2)
  z <- theta[3]   ### z is Fisher's z transform for rho
  rho <- (exp(2*z)-1) / (1 + exp(2*z))
  n <- length(y)
  Gamma1 <- diag(1+c(0,rep(rho^2,n-2),0))
  for(i in 2:n)
    Gamma1[i,i-1]<- Gamma1[i-1,i] <- -rho
  lik <- -n/2 * log(sigma2) + 0.5 * log(1-rho^2) -1/(2*sigma2) *
    mahalanobis(y, rep(mu,n), Gamma1, inverted = TRUE)
  return(lik)
}
# We need a function for simulating a data set
genDataAR1 <- function(theta, data)
```

```

{
  out <- data
  mu <- theta[1]
  sigma <- exp(theta[2])
  z <- theta[3]
  rho <- (exp(2*z)-1) / (1 + exp(2*z))
  n <- length(data$y)
  y <- rep(0,n)
  y[1] <- rnorm(1,mu,s=sigma*sqrt(1/(1-rho^2)))
  for(i in 2:n)
    y[i] <- mu + rho * (y[i-1]-mu) + rnorm(1) * sigma
  out$y <- y
  return(out)
}
# For inference on the mean parameter we need a function returning the first component of theta
psifcn.mu <- function(theta) theta[1]
# Now we can call the function
rs.mu <- rstar(dat.y, c(0,0,0), likAR1, fpsci=psifcn.mu, psival=2, datagen=genDataAR1, R=1000,
               trace=TRUE, psidesc="mean parameter")
summary(rs.mu)

```

rstar.ci

*Confidence intervals on a scalar function of interest by the  $r^*$  statistic*

## Description

This function obtains confidence intervals for a scalar function of interest, based on the  $r^*$  statistic.

## Usage

```

rstar.ci(data, thetainit, floglik, fscore=NULL, fpsci, datagen, R=1000, seed=NULL,
         ronly=FALSE, psidesc=NULL, constr.opt="solnp", lower=NULL, upper=NULL,
         control=list(...), ...)

```

## Arguments

data	The data as a list. All the elements required to compute the likelihood function at a given parameter value should be included in this list.
thetainit	A numerical vector containing the initial value for the parameter of the model. It will be used as starting point in the numerical optimization of the likelihood function.
floglik	A function which returns the log likelihood function at a given parameter value. In particular, for a certain parameter value contained in a numerical vector theta, a call floglik(theta, data) should return a scalar numerical value, the log likelihood function at theta.
fscore	An optional function which returns the score function at a given parameter value. It must return a numerical vector of the same length of thetainit. For a certain parameter value contained in a numerical vector theta, a call

	fscore(theta, data) should return the gradient of the log likelihood function at theta. Default is NULL, implying that numerical differentiation will be employed.
fpsi	A function which specifies the parameter of interest. A call fpsi(theta) should return a scalar value.
datagen	A function which simulates a data set. A call datagen(theta, data) will generate a copy of the data list, with the values of the response variable replaced by a set of values simulated from the parametric statistical model assumed for the response variable.
R	The number of Monte Carlo replicates used for computing the $r^*$ statistic. A positive integer, default is 1000.
seed	Optional positive integer, the random seed for the Monte Carlo computation. Default is NULL.
ronly	Logical. If set to TRUE the computation of the $r^*$ statistic will be skipped, and only the value of the signed likelihood ratio test statistic $r$ will be returned by the procedure, without any Monte Carlo computation. Default is FALSE.
psidesc	An optional character string describing the nature of the parameter of interest. Default is NULL.
constr.opt	Constrained optimizer used for maximizing the log likelihood function under the null hypothesis. Possible values are "solnp" or "alabama", with the former employing the solnp function from package Rsolnp and the latter the constrOptim.nl from the package alabama. Defaults is "solnp".
lower, upper	Optional numeric values defining the lower/upper limit of a grid of points for the parameter of interest, where the $r^*$ statistic will be evaluated. Default is NULL.
control	A list of parameters for controlling the computation of confidence intervals. See <a href="#">rstar.ci.control</a> .
...	Arguments to be used to form the default control argument if it is not supplied directly.

## Details

The function obtains 90%, 95% and 99% two-sided confidence intervals for the scalar function of interest based on the  $r^*$  statistic.

The function requires the user to provide three functions defining the log likelihood function, the scalar parametric function of interest, and a function for generating a data set from the assumed statistical model. A further function returning the gradient of the log likelihood is not required, but if provided it will speed up the computation.

When `ronly = TRUE` the function literally returns the value of the signed likelihood ratio test statistic  $r$  only. The function handles also one-parameter models.

The function provides two different strategies to obtain the various confidence intervals. The default strategy, invoked by leaving either `lower` or `upper` to NULL, starts from the MLE and moves away in a stepwise fashion, until the  $r^*$  statistic crosses the standard normal quantiles corresponding to the 99% two-sided confidence interval. It is crucial to start the search a bit away from the MLE, where the  $r^*$  is singular, and this is regulated by the `away` argument of the [rstar.ci.control](#) function. The first strategy may fail to cross the target normal quantiles when the profile likelihood has an upper

asymptote. For such cases, and for any other instances when the output of the default strategy is deemed not satisfactory, it is possible to specify the range of a grid of values where the  $r^*$  statistic will be evaluated. The lower and upper argument specify the lower and upper limit of such grid, whereas the number of points is controlled by the npoints of the [rstar.ci.control](#) function.

## Value

The returned value is an object of class "rstarci", containing the following components:

psivals	A list of values for the parameter of interest for which the $r$ and $r^*$ statistics have been evaluated.
rvals	A numerical list containing the values of the $r$ statistic evaluated at each element of psivals.
NPvals, INFvals	Numerical lists containing the values of the Nuisance Parameter adjustment (NP) and the Information adjustment (INF) from the decomposition of the $r^*$ - $r$ adjustment, for each of the psivals values. Not computed when ronly = TRUE.
rsvals	The observed value of the $r^*$ statistic at each element of psivals. Not computed when ronly = TRUE.
CIr	A 3 x 2 matrix containing the 90%, 95% and 99% confidence intervals for the parameter of interest (first, second and third row respectively) based on the first-order $r$ statistic.
CIrs	A 3 x 2 matrix containing the 90%, 95% and 99% confidence intervals for the parameter of interest (first, second and third row respectively) based on the $r^*$ statistic. Not computed when ronly = TRUE.
seed	Random seed used for Monte Carlo replicates used for computing the $r^*$ statistic. Not returned when ronly = TRUE.
psidesc	A character string describing the nature of the parameter of interest.
R	Number of Monte Carlo replicates used for computing the $r^*$ statistic. Not returned when ronly = TRUE.

There are print, summary and plot methods for this class.

## References

The method implemented in this function was proposed in  
 Skovgaard, I.M. (1996). An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.  
 For a general review  
 Severini, T.A. (2000). *Likelihood Methods in Statistics*. Oxford University Press.

## See Also

[rstar](#), [rstar.ci.control](#).

## Examples

```
# A negative binomial example, taken from Venables and Ripley (2002, MASS4 book)
library(MASS)
# The quine data are analysed in Section 7.4
data(quine)
# We fit a model with just the main effects
quine.nb1 <- glm.nb(Days ~ Eth + Sex + Age + Lrn, data = quine)
# The data list includes the design matrix and the response vector
quinedata <- list(X=model.matrix(quine.nb1), y=quine$Days)
# Let us define the required functions
# Log likelihood, log link
logLikNbin <- function(theta,data)
{
  y <- data$y
  X <- data$X
  eta <- X %*% theta[1:ncol(X)]
  mu <- exp(eta)
  alpha <- theta[ncol(X)+1]
  l <- sum(lgamma(y + alpha) + y * log(mu) - (alpha + y) * log(alpha + mu)
          - lgamma(alpha) + alpha * log(alpha))
  return(l)
}
# Score function
gradLikNbin <- function(theta,data)
{
  y <- data$y
  X <- data$X
  eta <- X %*% theta[1:ncol(X)]
  mu <- exp(eta)
  alpha <- theta[ncol(X)+1]
  g <- rep(0,ncol(X)+1)
  g[1:ncol(X)] <- t(y - (alpha+y)*mu / (alpha+mu)) %*% X
  g[ncol(X)+1] <- sum(digamma( y + alpha) - log(alpha + mu) - (alpha + y) / (alpha + mu)
                      - digamma(alpha) + 1 + log(alpha))
  return(g)
}
# Data generator
genDataNbin <- function(theta,data)
{
  out <- data
  X <- data$X
  eta<- X %*% theta[1:ncol(X)]
  mu <- exp(eta)
  out$y <- rnegbin(length(data$y), mu=mu, theta=theta[ncol(X)+1])
  return(out)
}
# Confidence intervals for the coefficient of EthN
## Not run:
obj <- rstar.ci(quinedata, thetainit=c(coef(quine.nb1),quine.nb1$theta), floglik=logLikNbin,
               datagen=genDataNbin, fscore=gradLikNbin, fpsi=function(theta) theta[2], R=1000,
               psidesc="Coefficient of EthN")
print(obj)
```

```

summary(obj)
plot(obj)
# Confidence intervals for the overdispersion parameter
obj <- rstar.ci(quinedata, thetainit=c(coef(quine.nb1),quine.nb1$theta), floglik=logLikNbin,
               datagen=genDataNbin, fscore=gradLikNbin, fpsi=function(theta) theta[8], R=1000,
               psidesc="Overdispersion parameter")
summary(obj)
plot(obj)

## End(Not run)

```

---

rstar.ci.control	<i>Auxiliary function for controlling computation of <math>r^*</math>-based confidence intervals</i>
------------------	--

---

## Description

Auxiliary function for `rstar.ci`.

## Usage

```
rstar.ci.control(npoints=10, away=0.3, stepsizefac=3, maxstep=50, trace=TRUE)
```

## Arguments

<code>npoints</code>	Integer giving the number of points at which the $r^*$ and $r$ statistics will be evaluated away from the MLE in each direction when both lower and upper are not null. When either lower or upper are NULL, such value is only roughly proportional to the number of evaluation points. Default is 10.
<code>away</code>	Positive value indicating how far from the MLE the computation of the $r^*$ and $r$ statistics will be started, expressed in units of standard error of the scalar function of interest. Default is 0.3.
<code>stepsizefac</code>	Positive value used to determine the stepsize of the confidence interval algorithm when either lower or upper are null. In particular, the stepsize is given by <code>stepsizefac/npoints</code> times the standard error of the scalar function of interest. Default is 3.
<code>maxstep</code>	Integer giving the maximum number of steps that will be taken for crossing the normal quantiles for a 99 confidence interval for the $r^*$ statistic. Default is 50.
<code>trace</code>	Logical indicating if output should be produced during the computation. Default is TRUE.

## Value

A list with components named as the arguments.

## See Also

[rstar.ci](#).

**Examples**

```
# A variation on example(rstar.ci):  
## Not run:  
obj <- rstar.ci(quinedata, thetainit=c(coef(quine.nb1),quine.nb1$theta), floglik=logLikNbin,  
              datagen=genDataNbin, fscore=gradLikNbin, fpsf=function(theta) theta[2], R=1000,  
              psidesc="Coefficient of EthN", npoints=5, away=0.1)  
plot(obj)  
  
## End(Not run)
```



# Index

\*Topic **datasets**

finndat, [2](#)

\*Topic **htest**

logMPL, [3](#)

logPL, [5](#)

rstar, [8](#)

rstar.ci, [11](#)

rstar.ci.control, [15](#)

\*Topic **package**

likelihoodAsy-package, [2](#)

finndat, [2](#)

likelihoodAsy (likelihoodAsy-package), [2](#)

likelihoodAsy-package, [2](#)

logMPL, [3](#)

logPL, [5](#)

rstar, [8](#), [13](#)

rstar.ci, [10](#), [11](#), [15](#)

rstar.ci.control, [12](#), [13](#), [15](#)