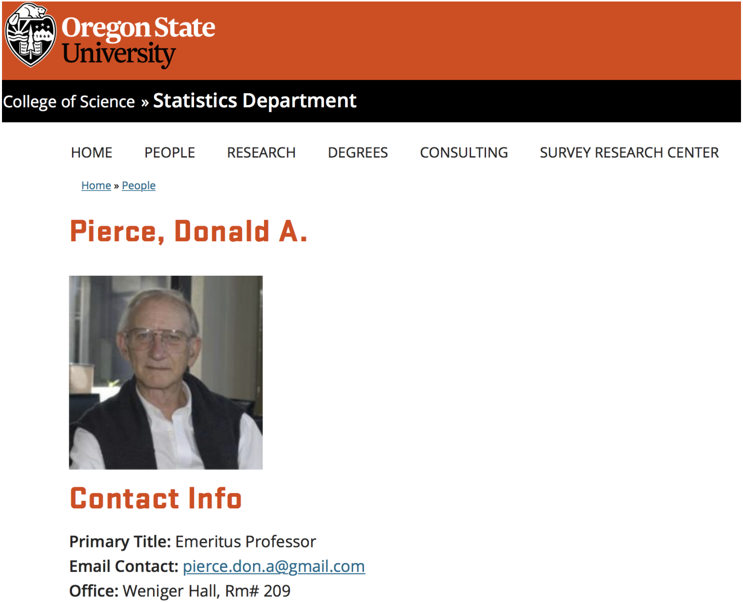
Modern likelihood-frequentist inference with the likelihoodAsy package

Ruggero Bellio

Department of Economics & Statistics, University of Udine (Italy)

Joint work with: Donald A. Pierce, Oregon State University (USA)

My co-author



Oregon State University

College of Science » Statistics Department

HOME PEOPLE RESEARCH DEGREES CONSULTING SURVEY RESEARCH CENTER

Home » People

Pierce, Donald A.

Contact Info

Primary Title: Emeritus Professor

Email Contact: pierce.don.a@gmail.com

Office: Weniger Hall, Rm# 209

Outline

- Some background on likelihood asymptotics
- The `R` package `likelihoodAsy`
- Example 1: Weibull regression
- Example 2: Binomial GLMM
- Winding up

Modern likelihood asymptotics

- Started by several researchers in the 1980s, it is well summarised in recent reviews by Severini (2000) and Skovgaard (2001)
- Pierce & Bellio (2017) tracks some of the developments, with an effort to make them accessible to a wider audience.
- The paper has an accompanying `R` package, named `likelihoodAsy` (on CRAN) which implements some of the methods.

Setting of interest and notation

- Let $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ be a sample from a parametric statistical model indexed by a p-dimensional parameter $\boldsymbol{\theta}$

$$\mathbf{y} \sim p(\mathbf{y}; \boldsymbol{\theta})$$

- $\ell(\boldsymbol{\theta}; \mathbf{y})$ is the log likelihood function and $\widehat{\boldsymbol{\theta}}$ is the Maximum Likelihood Estimate (MLE) of $\boldsymbol{\theta}$.

- Main methodology concerns inference about a scalar function of the parameters $\psi(\boldsymbol{\theta})$.

The r^* formula

- For testing $\psi(\boldsymbol{\theta}) = \psi$ we rely on the **directed deviance**

  $$r_\psi(\mathbf{y}) = \operatorname{sgn}(\widehat{\psi} - \psi) \left[2\left\{\ell(\widehat{\boldsymbol{\theta}}; \mathbf{y}) - \ell(\widehat{\boldsymbol{\theta}}_\psi; \mathbf{y})\right\}\right]^{1/2}$$

  for which $r_\psi(\mathbf{Y}) \dot\sim N(0,1)$ with 1st-order error.

- Key theoretical result is the r^*-formula

  $$\Pr\left\{r_\psi(\mathbf{Y}) \leq r_\psi(\mathbf{y}); \boldsymbol{\theta} : \psi(\boldsymbol{\theta}) = \psi\right\} = \Phi\{r^*_\psi(\mathbf{y})\}\left\{1 + O\left(n^{-1}\right)\right\}$$

  where $r^*_\psi(\mathbf{y})$ is a **modified directed deviance**, introduced by Barndorff-Nielsen (1986).

- It provides fairly accurate confidence intervals and tests.

  Results similar to the **most accurate parametric bootstrap method**, with no need to any additional fitting.

Comments: on the nature of r^*

- The r^* formula is derived from the p^* formula, giving an accurate approximation to the distribution of the MLE conditional on an ancillary statistic.
- The ancillary statistic is a notional quantity. It makes the MLE to be approximately sufficient, and sufficiency provides inferential optimality.
- The r^* statistic results from the transformation of the distribution of $\widehat{\boldsymbol{\theta}}$ to that of $r_\psi(\mathbf{Y})$.

The expository paper by Pierce & Bellio (2017) provides a full account of these steps.

Key parts of the package

A major feature of the package is to organize the calculations of r^*, which are **rather challenging**. Some key parts:

1. The package implements the r^* version due to Skovgaard (1996), which requires the computation of certain expected values. A suitable **Monte Carlo** approach carries out the task.

2. A key step for computing the r statistic is to obtain the constrained MLE

   $$\widehat{\boldsymbol{\theta}}_\psi = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}}\ \ell(\boldsymbol{\theta}) \quad \text{subject to} \quad \psi(\boldsymbol{\theta}) = \psi$$

   We endorse the **augmented Lagrangian** method, which is implemented in some `R` packages, including `Rsolnp` and `alabama`

likelihoodAsy at work

Usage of the software requires **two user-supplied functions**

- A function for evaluating the log-likelihood.
- A function that generates a sample under the model assumed for the data.
- Optional A function for evaluating the log-likelihood gradient (the score function).

Example 1: Weibull regression

- Sample of n observations from a Weibull distribution, with survival function $S(t_i; \boldsymbol{\beta}, \gamma) = \exp\left[-\left\{t_i^{\gamma} \exp(\mathbf{x}_i^{\top}\boldsymbol{\beta})\right\}\right]$.

- $\boldsymbol{\theta} = (\boldsymbol{\beta}, \gamma)$, and we take ψ as the log reliability at t_0, for a given covariate vector $\mathbf{x}_0$

$$\boxed{\psi(\boldsymbol{\beta}, \gamma) = -t_0^{\gamma} \exp(\mathbf{x}_0^{\top}\boldsymbol{\beta})}$$

The two functions

```r
loglik <- function(theta, data) {
  logy <- log(data$y)
  X <- data$X
  loggam <- theta[1]
  beta <- theta[-1]
  gam <- exp(loggam)
  H <- exp(gam * logy + X %*% beta)
  out <- sum(X %*% beta + loggam + (gam - 1) * logy - H)
  return(out)
}

gendat <- function(theta, data) {
  out <- data
  X <- data$X
  n <- nrow(X)
  beta <- theta[-1]
  gam <- exp(theta[1])
  out$y <- (rexp(n) / exp(X %*% beta)) ^ (1 / gam)
  return(out)
}
```

Data list and definition of ψ

We take the `leuk` data in `MASS` package involving simple linear regression on `log(WBC)` of the log failure rate for leukemia survival.

First we define the `data` list and the parameter of interest

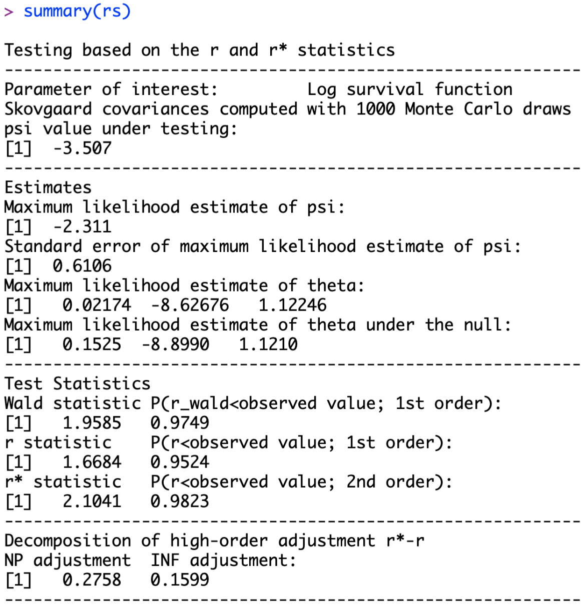
```r
library(MASS)
data(leuk)
leuk.p <- subset(leuk, ag=="present")
X <- model.matrix(~log(wbc, base=10),  data=leuk.p)
data.list <- list(X = X, y = leuk.p$time)

psifcn <- function(theta) {
  beta <- theta[-1]
  gam <- exp(theta[1])
  y0 <- 130; x0 <- 4
  psi <- -(y0 ^ gam) * exp(beta[1] + x0 * beta[2])
  return(psi)
}
```

The rstar function

The main function is rstar, here used for testing $\psi = \log(0.03)$

```r
library(likelihoodAsy)
rs <- rstar(data=data.list, thetainit = c(0, 0, 0),
           floglik = loglik,  fpsi = psifcn, psival = log(0.03),
           datagen = gendat, psidesc="Log survival function")
```

Summary function for tests



```
> summary(rs)

Testing based on the r and r* statistics
-----------------------------------------------------------
Parameter of interest:         Log survival function
Skovgaard covariances computed with 1000 Monte Carlo draws
psi value under testing:
[1]  -3.507
-----------------------------------------------------------
Estimates
Maximum likelihood estimate of psi:
[1]  -2.311
Standard error of maximum likelihood estimate of psi:
[1]  0.6106
Maximum likelihood estimate of theta:
[1]   0.02174  -8.62676   1.12246
Maximum likelihood estimate of theta under the null:
[1]   0.1525  -8.8990   1.1210
-----------------------------------------------------------
Test Statistics
Wald statistic P(r_wald<observed value; 1st order):
[1]   1.9585   0.9749
r statistic    P(r<observed value; 1st order):
[1]   1.6684   0.9524
r* statistic   P(r<observed value; 2nd order):
[1]   2.1041   0.9823
-----------------------------------------------------------
Decomposition of high-order adjustment r*-r
NP adjustment  INF adjustment:
[1]   0.2758   0.1599
-----------------------------------------------------------
```

The `rstar.ci` function

It computes 99%, 95% and 90% confidence intervals based on r^* and r, trying to determine automatically a suitable grid of values where to evaluate the two statistics.

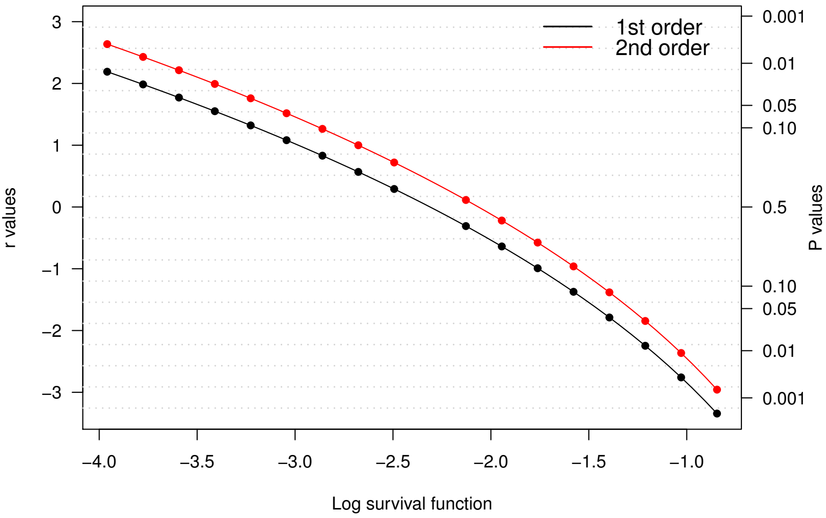
```
rs.ci <- rstar.ci(data=data.list, thetainit = c(0, 0, 0),
            floglik = loglik,  fpsi = psifcn,
            datagen = gendat, psidesc="Log survival function")
```

There are `summary` and `plot` functions for the resulting object.

Summary for confidence intervals

```
> summary(rs.ci)
Confidence interval calculations based on likelihood asymptotics
-----------------------------------------------------------------------------
Parameter of interest:         Log survival function
Calculations based on a grid of 17 points
Skovgaard covariances computed with 1000 Monte Carlo draws
-----------------------------------------------------------------------------
1st-order
         90%                         95%                         99%
( -3.487   ,  -1.456 )       ( -3.755   ,  -1.324 )       ( -4.306   ,  -1.091 )
2nd-order
         90%                         95%                         99%
( -3.152   ,  -1.294 )       ( -3.398   ,  -1.174 )       ( -3.9213   ,  -0.9648 )
-----------------------------------------------------------------------------
Decomposition of high-order adjustment
Nuisance parameter adjustment (NP)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.2203   0.2403   0.2548   0.2509   0.2634   0.2711
Information adjustment (INF)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.1545   0.1575   0.1581   0.1576   0.1584   0.1586
-----------------------------------------------------------------------------
```

Profile plots (p-value functions)



Example 2: Logistic regression with random intercepts

- A standard model for clustered binomial data, assuming

$$\log\{p_{ij}/(1-p_{ij})\} = \mathbf{x}_{ij}^\top \boldsymbol{\beta} + u_i\,, \qquad u_i \sim N(0,\sigma^2)$$

- The log-likelihood function requires the integrate of the random effects. One possible approach uses Gaussian quadrature, and this is done in the package vignette.

- Here we use instead the `TMB` package for automatic differentiation (`https://github.com/kaskr/adcomp`), that requires to define a `C++` template (a `.cpp` file) for $-\ell(\boldsymbol{\theta};\mathbf{y})$.

- `TMB` performs random effects integration via the Laplace approximation

logistic.cpp

```cpp
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() () {
  /* Data section */
  DATA_INTEGER(n);  //number of groups
  DATA_IVECTOR(m); //group size
  DATA_MATRIX(y);  //first column n succ, second n trials
  DATA_MATRIX(X); //design matrix

  /* Parameter section */
  PARAMETER_VECTOR(beta);
  PARAMETER(lsigma);
  PARAMETER_VECTOR(u);

  using namespace density;

  Type nll = 0.0;
  nll -= sum(dnorm(u, Type(0), exp(lsigma), true));
  vector<Type> Xbeta = X * beta;

  int ind = 0;
  for(int i=0;i<n;i++){
   for(int j=0;j<m(i);j++){
     int k = j + ind;
     Type eta = Xbeta(k) + u(i);
     Type prob =  exp(eta) / (1 + exp(eta));
     nll -= dbinom(y(k,0), y(k,1), prob, true);
     }
    ind += m(i);
   }
   return nll;
}
```

Functions for log likelihood and gradient

A call to `TMB::MakeADFun` does all the job

```
logLik <- function(theta, data) {
 obj0 <- TMB::MakeADFun(data=data$ADdata,
               parameters=data$ADparameters,
              DLL="logistic", random=c("u"), silent=TRUE)
 return(-obj0$fn(theta))
}

grLik <- function(theta, data) {
 obj0 <- TMB::MakeADFun(data=data$ADdata,
               parameters=data$ADparameters,
               DLL="logistic", random=c("u"), silent=TRUE)
 return(-obj0$gr(theta))
}
```

Data list

We consider data on a multicenter clinical trial of two topical cream treatments from Beitler and Landis (1985, BMCS), and fit a logistic regression model for the treatment indicator, with random intercepts for clinic.

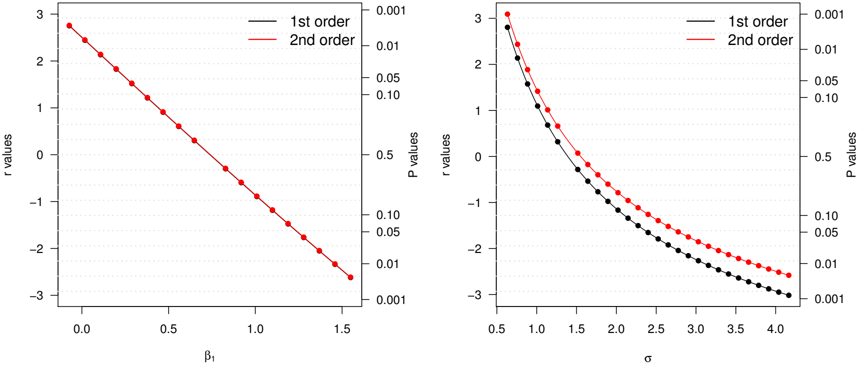
The `data` list involves all the quantity needed for `TMB::MakeADFun`

```r
multicenter <- read.table("multicenter.txt", h =TRUE)
N <- length(unique(multicenter$ID))
M <- table(multicenter$ID)
Y <- cbind(multicenter$y, multicenter$den)
X <- model.matrix(~I(trt==1), multicenter)
mydat <- list(n=N, m=M, y=as.matrix(Y), X=as.matrix(X))
parameters <- list(beta=rep(0,2), lsigma=0, u=rep(0,N))

data.list <- list(ADdata=mydat, ADparameters = parameters)
```

Function that generates a sample

```
gendat <- function(theta, data) {
   out <- data
   X <- data$ADdata$X
   p <- ncol(X)
   n <- data$ADdata$n
   beta <- theta[1:p]
   sigma <-  exp(theta[p+1])
   u <- rnorm(n) * sigma
   linpred <- X %*% beta + u
   out$data$y[,1] <- rbinom(nrow(X), size=out$ADdata$y[,2],
                            prob=plogis(linpred))
   return(out)
 }
```

Profile plots for $\psi(\boldsymbol{\theta}) = \beta_1$ and $\psi(\boldsymbol{\theta}) = \sigma$



Winding up

- `likelihoodAsy` makes some arcane inferential methods readily accessible, and it is a handy alternative to bootstrapping.
- Currently, there is some support for the **modified profile likelihood** (Severini, 2000) for multidimensional ψ, but with room for improvement.
- We would like to add some further functions for constrained optimisation.
- Better integration with the `TMB` package would be helpful.

References: Theory

- Beitler, P.J. & Landis, J. R. (1985). A mixed-effects model for categorical data. *Biometrics*, 41, 991-1000.
- **Pierce, D.A. & Bellio, R. (2017). Modern likelihood-frequentist inference. Int. Statist. Rev., 85, 519-541.**
- Severini, T.A. (2000). *Likelihood Methods in Statistics*. OUP, Oxford.
- Skovgaard, I.M. (1996). An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, 2, 145-65.
- Skovgaard, I.M. (2001). Likelihood asymptotics. *Scand. J. Statist.*, 28, 3-32.

References: R packages

- Ghalanos, A. & Theussl, S. (2015). `Rsolnp`: General non-linear optimization using augmented Lagrange multiplier method. R package version 1.16.
- Kristensen, K., Nielsen, A., Berg, C.W., Skaug, H. & Bell, B.M. (2016). `TMB`: Automatic differentiation and Laplace approximation. *J. Statist. Software*, 70, 1-21.
- Varadhan, R. (2015). `alabama`: Constrained nonlinear optimization. R package version 2015.3-1.
- Venables, W. N. & Ripley, B. D. (2002). Modern Applied Statistics with S. Fourth Edition. Springer, New York.

Thank you for your attention !

`http://ruggerobellio.weebly.com`