The profileModel R Package: Profiling Objectives for Models with Linear Predictors.

by Ioannis Kosmidis

Introduction

In a multi-parameter problem the profiles of the likelihood can be used for the construction of confidence intervals for parameters, as well as to assess features of the likelihood surface such as local maxima, asymptotes, etc., which can affect the performance of asymptotic procedures. The `profile` methods of the **R** language (**stats** and **MASS** packages) can be used for profiling the likelihood function for several classes of fitted objects, such as `glm` and `polr`. However, the methods are limited to cases where the profiles are almost quadratic in shape and can fail, for example, in cases where the profiles have an asymptote.

Furthermore, often the likelihood is replaced by an alternative objective for either the improvement of the properties of the estimator, or for computational efficiency when the likelihood has a complicated form (see, for example, Firth (1993) for a maximum penalized likelihood approach to bias-reduction, and Lindsay (1988) for composite likelihood methods, respectively). Alternatively, estimation might be performed by using a set of estimating equations which do not necessarily correspond to a unique objective to be optimized, as in quasi-likelihood estimation (Wedderburn, 1974; McCullagh, 1983) and in generalized estimating equations for models for clustered and longitudinal data (Liang and Zeger, 1986). In all of the above cases, the construction of confidence intervals can be done using the profiles of appropriate objective functions in the same way as the likelihood profiles. For example, in the case of bias-reduction in logistic regression via maximum penalized likelihood, Heinze and Schemper (2002) suggest to use the profiles of the penalized likelihood, and when estimation is via a set of estimating equations Lindsay and Qu (2003) suggest the use of profiles of appropriate quadratic score functions.

profileModel is an **R** package that provides tools to calculate, evaluate, plot and use for inference the profiles of *arbitrary* objectives for arbitrary fitted models with *linear predictors*. It allows the developers of fitting procedures to use these capabilities by simply writing a small function for the appropriate objective to be profiled. Furthermore, it provides an alternative to already implemented profiling methods, such as `profile.glm` and `profile.polr`, by extending and improving their capabilities (see, the incidence of leaf-blotch on barley example in the Examples section).

In its current version (0.5-3), the **profileModel** package has been tested and is known to work for fitted objects resulting from `lm`, `glm`, `polr`, `gee`, `geeglm`, `brglm`, `brlr`, `BTm` and `survreg`.

Supported fitted objects

profileModel aims at generality of application and thus it is designed to support most of the classes of fitted objects that are constructed according to the specifications of a fitted object as given in Chambers and Hastie (1991, Chapter 2). Specifically, the supported classes of fitted objects (`object`) should comply with the following:

1. The parameters of the model to be fitted have to be related through a linear combination of parameters and covariates. That is, only models with *linear predictors* are supported.

2. The fitting procedure that resulted in `object` has to support `offset` in `formula`. Also, `object$call` has to be the call that generated `object`.

3. `object` has to be an object which supports the method `coef` and which has `object$terms` with the same meaning as, for example, in `lm` and `glm`. `coef(object)` has to be a *vector* of coefficients where each component corresponds to a column of the model matrix

   ```
   > mf <- model.frame(object$terms,
   +     data = eval(object$call$data))
   > model.matrix(object$terms, mf,
   +     contrasts = object$contrasts)
   ```

   or maybe just `model.matrix(object)`, for fitted objects that support the `model.matrix` method.

   Exceptions to this are objects returned by the `BTm` function of the **BradleyTerry** package, where some special handling of the required objects takes place.

   Note that any or both of `data` and `contrasts` could be `NULL`. This depends on whether the `data` argument has been supplied to the fitting function and whether `object$contrasts` exists.

 ISSN 1609-3631

4. Support for a `summary` method is optional. The `summary` method is only used for obtaining the estimated asymptotic standard errors associated to the coefficients in `object`. If these standard errors are not in `coef(summary(object))[,2]`, they can be supplied by the user.

The profileModel objective functions

Given a fitted object of p parameters, the $p-1$ dimensional restricted fit is an object resulting from the following procedure:

1. Restrict a parameter β_r, $r \in \{1,\ldots,p\}$, to a specific scalar c.
2. Calculate cx_r, where x_r is the model matrix column that corresponds to β_r.
3. Estimate the remaining $p-1$ parameters with cx_r as an additional offset in the model formula.

The profiles of an objective should depend on the restricted fit and a **profileModel** objective function should be defined as such.

For example, if `object` was the result of a `glm` call and `restrFit` is the object corresponding to the restricted fit for $\beta_r = c$ for some $r \in \{1,\ldots,p\}$ then an appropriate objective function for the profile likelihood is

```
profDev <- function(restrFit, dispersion)
   restrFit$deviance/dispersion
```

where the `dispersion` argument has to be defined in the relevant **R** functions calling `profDev`. Once `profDev` is passed to the functions in `profileModel` i) the restricted fits for an appropriate grid of c-values are obtained using `glm` with the appropriate offsets and covariates in its `formula` argument and ii) the differences

```
profDev(restrFit) - profDev(object)
```

are calculated for each restricted fit. Note that, in the current case, where the fitting procedure is maximum likelihood and the objective is the deviance, the above difference is the log-likelihood ratio that is usually used for testing the hypothesis $\beta_r = c$, i.e.,

$$2\left\{l(\hat{\beta}_1,\ldots,\hat{\beta}_p) - l(\hat{\beta}^c_1,\ldots,\hat{\beta}^c_{r-1},c,\hat{\beta}^c_{r+1},\ldots,\hat{\beta}^c_p)\right\}, \quad (1)$$

where $\hat{\beta}^c_t$ $(t = 1,\ldots,r-1,r+1,\ldots,p)$ is the restricted maximum likelihood estimate when $\beta_r = c$, and $l(.)$ is the log-likelihood function.

As an illustrative example consider

```
> m1 <- example(birthwt, "MASS")$value
> library(profileModel)
> prof1.m1 <- profileModel(fitted = m1,
+     objective = profDev,
+     dispersion = 1)
```

Here, we use the default behaviour of the `profileModel` function; `prof1.m1$profiles` is a list of matrices, one for each estimated parameter in `m1`. The first column of each matrix contains 10 values of the corresponding parameter with the smallest and largest values being 5 estimated standard errors on the left and on the right, respectively, of the maximum likelihood estimate of the parameter in `m1`. The second column contains the values of (1) that correspond to the grid of values in the first column.

The profileModel function and class

The primary function of the **profileModel** package is the profiling function with the same name, which provides the following two main features.

Profiling the objective over a specified grid of values for the parameters of interest.

This feature can be used with generic objectives irrespective of their shape. It is most useful for the study of the shape of the profiles in a given range of parameter values. As an illustration consider the following:

```
> prof2.m1 <- update(prof1.m1,
+     which = c("age", "raceblack", "ftv1"),
+     grid.bounds = c(-2, 0, 0, 2, -1, 1),
+     gridsize = 50)
> plot(prof2.m1)
```

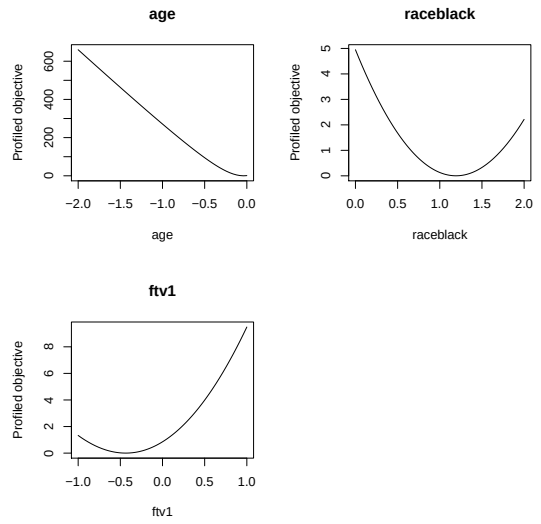

Figure 1: Profiling the log-likelihood ratio statistic over a grid of values.

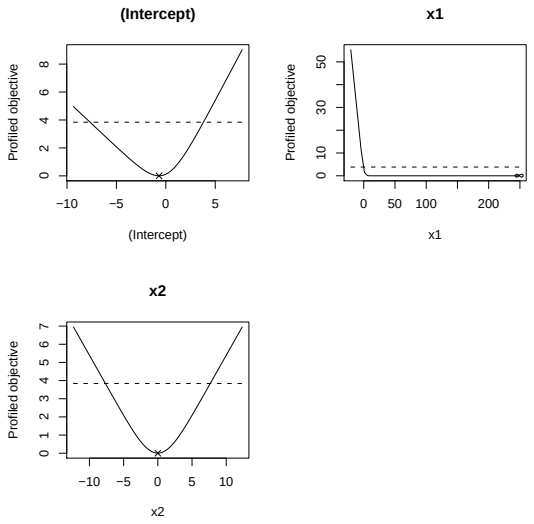

Figure 2: Profiling until the profile exceeds `qchisq(0.95,1)` (dashed line). The ML estimate for `x1` is infinity.

Profiling the objective until the value of the profile exceeds a given quantile value for the parameters of interest.

This feature relates to convex objectives. Restricted fits are calculated on the left and on the right of the estimate until the value of the profile exceeds a given quantile value for the parameters of interest. In contrast with the default `profile` methods, in **profileModel** the size of the steps to be taken on the left and on the right of the estimate depends on the shape of the profile curve through estimates of the slope of the tangents to the curve. Specifically, the steps are smaller when the profile curve seems to change fast. In this way any asymptotes in either direction can be detected (see the documentation of the `profileModel` function for detailed information on the calculation of the steps). For example, consider

```
> y <- c(1/3, 1/3, 1, 1)
> x1 <- c(0, 0, 1, 1) ;  x2 <- c(0, 1, 0, 1)
> m2 <- glm(y ~ x1 + x2, binomial)
> prof.m2 <- profileModel(m2,
+     objective = profDev,
+     quantile = qchisq(0.95, 1),
+     gridsize = 50, dispersion = 1)
> plot(prof.m2)
```

The `profileModel` procedure results in objects of class `profileModel`. S3 methods specific to `profileModel` objects are

- **Generics:**
  `print`, `update`
- **Plotting facilities:**
  `plot`, `pairs`
- **Construction of confidence intervals:**
  `profConfint`, `profZoom`, `profSmooth`

Construction of confidence intervals

The confidence-interval methods in the **profileModel** package relate to convex objectives. Asymptotic confidence intervals based on the profiles of the objective can be constructed using the `profConfint` method, which is a wrapper of two different methods for the computation of confidence intervals.

Spline smoothing via the `profSmooth` method

A smoothing spline is fitted on the points of the profiles in a `profileModel` object and then the endpoints of the confidence intervals are computed by finding the value of the parameter where the fitted spline has the value of the appropriate quantile. For example,

```
> profConfint(prof.m2, method="smooth")
                Lower    Upper
(Intercept) -7.670327 3.802680
x1          -0.507256      Inf
x2          -7.668919 7.668919
```

Since `quantile = qchisq(0.95,1)` was used for `prof.m2`, these are 95% asymptotic profile confidence intervals.

The advantage of spline smoothing is that it requires minimal computation. However, by construction the method depends on the size of the profiling grid, as well as on the shape of the profiles. The endpoints of the resultant confidence intervals are most accurate when the profiles are quadratic in shape.

Binary search via the `profZoom` method

A binary search is performed that 'zooms in' on the appropriate regions until the absolute difference of the quantile from the profile value at each end point is smaller than a specified tolerance. Construction of confidence intervals in this way requires the calculation of the value of the profiles at the candidates returned by the binary search. Thus, in contrast to smoothing, this method, whereas accurate, is computationally intensive. On the same example as for spline smoothing we have

```
> profConfint(prof.m2, method="zoom",
+     endpoint.tolerance = 1e-8)
                 Lower    Upper
(Intercept) -7.6703317 3.802770
x1          -0.8518235      Inf
x2          -7.6689352 7.668935
```

The smoothing method was accurate up to at least 3 decimal places for the endpoints of the confidence intervals for `(Intercept)` and `x1`. However, it failed to accurately estimate the left endpoint of the confidence interval for `x1`; the profile for `x1` in Figure 2 has an asymptote on the right.

Examples

Profile likelihood for `survreg` fitted objects

To illustrate the generality of application of the **profileModel** package we consider the example in the help file of the `survreg` function in the **survival** package.

```
> library(survival)
> m3 <- survreg(
+     Surv(futime, fustat) ~ ecog.ps + rx,
+     ovarian, dist= 'weibull', scale = 1)
```

For the parameters `(Intercept)`, `ecog.ps` and `rx` we calculate the values of the log-likelihood ratio statistic in (1), plot the profiles and obtain asymptotic confidence intervals based on the profiles. According to the rules for the construction of objectives in previous sections and to the available quantities in a `survreg` object, an appropriate **profileModel** objective could be defined as

```
> profLogLik <- function(restrFit) {
+     -2*restrFit$loglik[2]
+ }
```

In the case of a `survreg` object, the estimated asymptotic standard errors for the estimates are given by `summary(m3)$table[,2]`. We have

```
> prof.m3 <- profileModel(m3,
+     quantile=qchisq(0.95,1),
+     objective = profLogLik,
+     stdErrors = summary(m3)$table[,2],
+     gridsize=20)
```

Using binary search, the profile likelihood 95% asymptotic confidence intervals are

```
> profConfint(prof.m3,
+     method="zoom",
+     endpoint.tolerance = 1e-8)
                Lower     Upper
(Intercept)  4.5039809 9.7478481
ecog.ps     -1.6530569 0.7115453
rx          -0.5631772 1.8014250
```

Note that the 95% Wald asymptotic confidence intervals

```
> confint(m3)
                 2.5 %    97.5 %
(Intercept)  4.3710056 9.5526696
ecog.ps     -1.5836210 0.7173517
rx          -0.5689836 1.7319891
```

are similar because of the almost quadratic shape of the profiles, or equivalently of the apparent linearity of the signed square roots of the profiles as can be seen in Figure 3.

```
> plot(prof.m3, signed=TRUE)
```

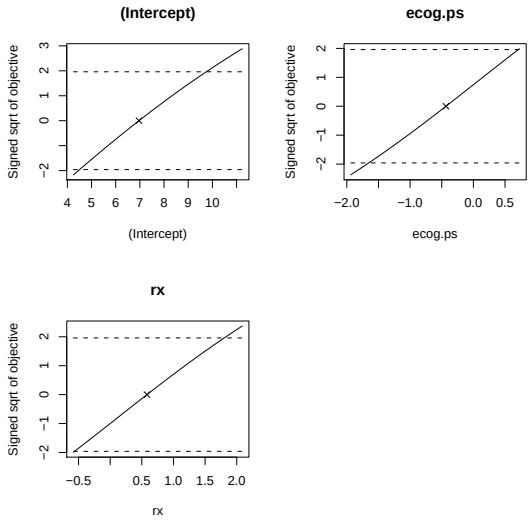

Figure 3: Signed square roots of the profiles for `m3`.

Incidence of leaf-blotch on barley

As as demonstration of the improvement over the default `profile` methods, as well as, of the generality of the **profileModel** package in terms of profiled objectives, we consider the leaf-blotch on barley study found in McCullagh and Nelder (1989, Section 9.4.2).

A linear logistic regression model with main effects is chosen to describe the variety and site effects and estimation is performed using quasi-likelihood with variance function $\mu^2(1-\mu)^2$ (see Wedderburn, 1974). The family object to be used along with `glm` is implemented through the `wedderburn` function of the **gnm** package.

```
> library(gnm)
> data(barley)
> m4 <- glm(y ~ site + variety,
+     family = wedderburn, data = barley)
```

In this case, an appropriate **profileModel** objective is the quasi-deviance (i.e. calculate the values of (1) with $l(.)$ replaced by the corresponding quasi-likelihood function).

```
> profPars <- paste("variety", c(2:9,"X"),
+     sep = "")
> prof1.m4 <- profileModel(m4,
+     objective = profDev,
+     quantile = qchisq(0.95, 1),
+     which = profPars,
+     gridsize = 20,
+     dispersion = summary(m4)$dispersion)
> cis1.m4 <- profConfint(prof1.m4)
> cis1.m4
              Lower    Upper
variety2 -1.4249881 0.5047550
variety3 -0.8653948 1.0327784
variety4  0.0086728 1.8875857
variety5  0.4234367 2.2604532
variety6  0.3154417 2.3844366
variety7  1.4351908 3.2068870
variety8  2.3436347 4.1319455
variety9  2.1698223 4.0626864
varietyX  2.9534824 4.7614612
> plot(prof1.m4, cis= cis1.m4)
```

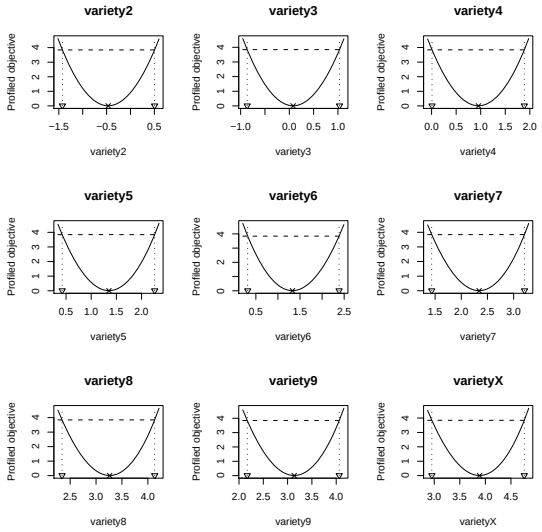

Figure 4: Profiles for `m4`.

In contrast to `profileModel`, the default `profile` method of the **MASS** package for `glm` fitted objects fails to calculate the correct profiles in this example; the profile confidence intervals using the default `profile` method are

```
> confint(m4)[profPars, ]
Waiting for profiling to be done...
               2.5 %     97.5 %
variety2 -1.42315422  0.5048776
variety3 -0.86535806  1.0328576
variety4 -0.03998811  0.8842478
variety5 -2.23635211 -2.2011092
variety6  0.31534770  2.3847282
variety7  1.41342757  1.7276810
variety8  2.31977238  2.6500143
variety9  2.15012980  2.5229588
varietyX  2.93050444  3.2755633
There were 36 warnings (use warnings()
to see them)
```

which clearly disagree with the confidence intervals obtained above (the reader is also referred to the example in the documentation of the `profConfint` function). The produced warnings give a clue for the misbehaviour; some of the restricted fits do not converge to the maximum quasi-likelihood estimates and so wrong values for the profiled objective are returned. This is most probably caused by bad starting values, which, in turn, depend on the size of the steps taken on the left and on the right of each estimate in the default `profile` method. In `profileModel`, the dependence of the size of the steps on the quasi-likelihood profiles avoids such issues.[1]

In addition to the `plot` method, the **profileModel** package provides a `pairs` method for the construction of pairwise profile trace plots (see, Venables and Ripley, 2002, for a detailed description). This method

[1] The `profile` method in the **MASS** package for `glm` objects was improved in May 2008, after the output for this example was generated.

is a minor modification of the original `pairs` method for `profile` objects in the **MASS** package. The modification was made under the GPL licence version 2 or greater and with the permission of the original authors, in order to comply with objects of class `profileModel`.

```
> pairs(prof1.m4)
```

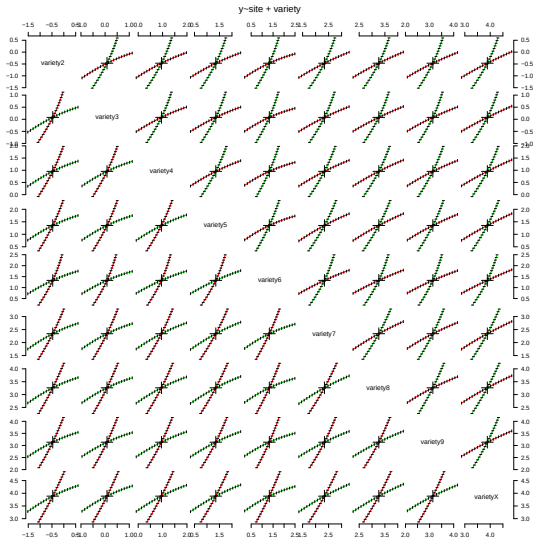

Figure 5: Pairwise profile trace plots for `m4`.

The direction of the pairwise profile trace plots in Figure 5 agrees with the fact that all the correlations amongst the variety effects are exactly $1/2$.

An alternative to the quasi-likelihood function for the construction of asymptotic confidence intervals in quasi-likelihood estimation is the quadratic form

$$Q^2(\beta) = s(\beta)^T i^{-1}(\beta) s(\beta)\,, \tag{2}$$

where $\beta = (\beta_1, \ldots, \beta_p)$ is the parameter vector, $s(\beta)$ is the vector of the quasi-score functions and $i(\beta) = \mathrm{cov}(s(\beta))$. A review on test statistics of the above form and their properties can be found in Lindsay and Qu (2003). The quadratic form (2) could be used when there is not a unique quasi-likelihood function corresponding to the set of quasi-score functions. The profile test statistic for the hypothesis $\beta_r = c$ for some $r \in \{1, \ldots, p\}$ has the form

$$Q^2(\hat\beta_1^c, \ldots, \hat\beta_{r-1}^c, c, \hat\beta_{r+1}^c, \ldots, \hat\beta_p^c) - Q^2(\hat\beta_1, \ldots, \hat\beta_p)\,, \tag{3}$$

where $\hat\beta_t^c$ $(t = 1, \ldots, r-1, r+1, \ldots, p)$ are the restricted quasi-likelihood estimates when $\beta_r = c$.

The appropriate **profileModel** objective to calculate (3) is implemented through the `RaoScoreStatistic` function, on which more details can be found in the documentation of **profileModel**. For `m4` we have

```
> prof2.m4 <- profileModel(m4,
+     objective = RaoScoreStatistic,
+     quantile = qchisq(0.95, 1),
+     which = profPars,
+     gridsize = 20, X = model.matrix(m4),
+     dispersion = summary(m4)$dispersion)
> profConfint(prof2.m4)
              Lower     Upper
variety2 -1.4781768 0.5971332
variety3 -0.9073192 1.0991394
variety4 -0.0401468 1.9064558
variety5  0.3890679 2.2472880
variety6  0.2144497 2.6012640
variety7  1.4212454 3.1476794
variety8  2.3107389 4.0685900
variety9  2.0921331 4.0503524
varietyX  2.9017855 4.6967597
```

Summary

In this article we introduce and demonstrate the main capabilities of the **profileModel** package. The described facilities, along with other complementary functions included in the package were developed with the following aims:

- **Generality:** the package provides a unified approach to profiling objectives. It supports most of the classes of fitted objects with linear predictors that are constructed according to the specifications of a fitted object as given by Chambers and Hastie (1991, Chapter 2).

- **Embedding:** the developers of current and new fitting procedures can have direct access to profiling capabilities. The only requirement is authoring a simple function that calculates the value of the appropriate objective to be profiled. Furthermore, each function of the **profileModel** package is designed to perform a very specific task. For example, `prelim.profiling` relates only to convex objectives and results in the grid of parameter values which should be used in order for the profile to cover a specific value (usually a quantile). Such a modular design enables developers to use only the relevant functions for their specific application (see, for example, the documentation of the `profile.brglm` and `confint.brglm` methods of the **brglm** package).

- **Computational stability:** all the facilities have been developed with computational stability in mind, in order to provide an alternative that improves and extends the capabilities of already available `profile` methods.

The result, we hope, is a flexible and extensible package that will be useful both to other package authors and to end-users of **R**'s statistical modelling functions.

Bibliography

J. M. Chambers and T. Hastie. *Statistical Models in S*. Chapman & Hall, 1991.

D. Firth. Bias reduction of maximum likelihood estimates. *Biometrika*, 80(1):27–38, 1993.

G. Heinze and M. Schemper. A solution to the problem of separation in logistic regression. *Statistics in Medicine*, 21:2409–2419, 2002.

K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73: 13–22, 1986.

B. G. Lindsay. Composite likelihood methods. In N. U. Prabhu, editor, *Statistical Inference from Stochastic Processes*, pages 221–239. American Mathematical Society, 1988.

B. G. Lindsay and A. Qu. Inference functions and quadratic score tests. *Statistical Science*, 18(3):394–410, 2003.

P. McCullagh. Quasi-likelihood functions. *The Annals of Statistics*, 11:59–67, 1983.

P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 2nd edition, 1989.

W. N. Venables and B. D. Ripley. Statistics complements to Modern Applied Statistics with S, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4/VR4stat.pdf`.

R. W. M. Wedderburn. Quasi-likelihood functions, generalized linear models, and the Gauss-Newton method. *Biometrika*, 61:439–447, 1974.

Ioannis Kosmidis
Department of Statistics
University of Warwick
Coventry
CV4 7AL
United Kingdom
`i.kosmidis@warwick.ac.uk`