LIKELIHOOD BASED INFERENCE ON THE BOX-COX FAMILY OF TRANSFORMATIONS: SAS AND MATLAB PROGRAMS

Scott Hyde
Department of Mathematical Sciences
Montana State University

April 7, 1999

A writing project submitted in partial fulfillment
of the requirements for the degree

Master of Science in Statistics

APPROVAL

of a writing project submitted by

SCOTT HYDE

This writing project has been read by the writing project director and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Statistics Faculty.

Date	Robert J. Boik Writing Project Director

Contents

1 Introduction 1

1.1 Conventional Linear Model 1

1.2 Box-Cox Family of Transformations 2

2 Estimation of the Transformation Parameter 3

2.1 Derivation of the Log Likelihood Function 3

3 Newton-Raphson Algorithm 6

4 Confidence Intervals for λ 8

4.1 Asymptotic Distribution of MLE 9

4.2 Inversion of LR Test 12

5 Examples 13

6 Programming Notes 17

A Appendix 20

A.1 SAS Code 20

A.2 MATLAB Code 25

A.3 Finding the limit of u_i and v_i 30

A.4 Finding the form for the observed Fisher's Total Information matrix. . 31

References 34

A SAS program to compute the MLE of the Box-Cox Transformation parameter

Abstract

Box and Cox [3] proposed a parametric family of power transformations of the data to reduce problems with non-normality and heteroscedasticity. This paper presents programs in SAS and MATLAB to compute the MLE and to compute approximate confidence intervals for the Box-Cox transformation parameter.

1 Introduction

1.1 Conventional Linear Model

In the conventional linear model, the response is modeled by a linear combination of parameters and explanatory variables plus random fluctuation. Additionally, the errors, or random fluctuation, $\boldsymbol{\varepsilon}_j$, $j = 1, \ldots, n$ are assumed to be independent and normally distributed with mean 0 and variance σ^2. In short,

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2\boldsymbol{I}).$$

Note that $\boldsymbol{y}$ is an $n \times 1$ vector, $\mathbf{X}$ is a $n \times p$ matrix, and $\boldsymbol{\beta}$ is a $p \times 1$ vector of undetermined coefficients.

Box and Cox [3] note that the above model implies several assumptions, namely

1. Linearity of Structure for $E(\boldsymbol{y})$;
2. Constancy of Error Variances;
3. Normality of Distributions; and
4. Independence of Observations.

Typically, the first and last assumptions are assumed to have been met. But, when these assumptions are not met, then conventional methods of inference, prediction, and estimation fail. Procedures are available for examining departures from these assumptions. If replications are available, then a standard lack of fit test can be performed. The Durbin-Watson test can be used to test for a serial correlation (i.e., AR(1)).

For the second of the above assumptions, there are several methods for detecting heteroscedasticity. One method consists of examining a plot of the residuals versus the predicted values. If the residual plot shows random scatter, then there is little evidence against the homogeneity of variance assumption. Nonetheless, this does not imply that homogeneity is satisfied. Other methods, including the residual plot method, can be found in Chapter 6 of Sen & Srivastava's textbook [11].

Two methods are prominent for detecting non-normality. These methods, namely probability plots and inference tests, also are described in Sen and Srivastava [11]. Probability plots are constructed by plotting the data versus quantiles γ_i of the normal distribution. More specifically, the i^{th} normal quantile is

$$\gamma_i = \Phi^{-1}\left(\frac{i - 3/8}{n + 1/4}\right).$$

When plotting γ_i versus the ordered data, a line should form if the data are normal. If the data are not normal, then other patterns will emerge. Inference tests for normality include the Shapiro-Wilk test, which seems to be the standard for small data sets. Other methods include the square of the correlation between the ordered data and the γ_i's, which was initially used as an approximation to the Shapiro-Wilk statistic. When n is large, the Kolmogorov test statistic may be used.

If the first three assumptions are not satisfied, then a transformation of the original data may help so that methods of inference, prediction, and estimation can be valid. Several different transformation choices have been discussed by various authors. One main goal is to attain approximate constant variance. References are given to other papers in Box and Cox [3], Hoyle [5], and Sakia [9]. While there are many choices for transformations, this paper discusses the Box-Cox transformation of the dependent variable.

1.2 Box-Cox Family of Transformations

The Box-Cox transformation family introduces a new parameter λ and transforms the response, $\boldsymbol{y}$, to $\boldsymbol{z} = \boldsymbol{y}^{(\boldsymbol{\lambda})}$. The transformation is as follows:

$$z = y^{(\lambda)} \quad = \begin{cases} \dfrac{y^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0; \\ \ln y & \text{if } \lambda = 0. \end{cases} \tag{1}$$

The objective of this family of transformations is to produce data which follow a normal distribution more closely than the original data. While this transformation does not guarantee that the transformed data are normal, it does reduce problems with estimation, prediction, and inference.

The transformation above can be used only on positive response variables. One transformation suggested by Box and Cox [3] which allows for negative data is

$$y^{(\boldsymbol{\lambda})} = \begin{cases} \dfrac{(y+\lambda_2)^{\lambda_1}-1}{\lambda_1} & \text{if } \lambda_1 \neq 0; \\ \ln(y+\lambda_2) & \text{if } \lambda_1 = 0. \end{cases}$$

The discussion in this paper is based on equation (1).

2 Estimation of the Transformation Parameter

Estimation of λ can be done by Bayesian methods, by likelihood-based methods, or by other methods. One estimator may be chosen over another if one is more robust. In fact, the mle of λ is not robust. A robust estimator of λ was described by Carroll [4]. Although the mle of λ is not robust, there are still situations where it is a very good estimator. For this reason, the mle of λ will be discussed in this paper.

2.1 Derivation of the Log Likelihood Function

Maximum likelihood estimation consists of the following steps. First, the likelihood function is specified. Second, the likelihood function is maximized with respect to the unknown parameters. For the models in this paper, the maximizers (i.e., mles) can be obtained by taking the derivative of the likelihood function, setting the derivative to zero, and solving for the unknown parameters.

To construct the likelihood function, the distribution of the data, $\boldsymbol{y}$, can be derived based on the assumption that the transformed data $\boldsymbol{y}^{(\boldsymbol{\lambda})}$ are normal. The density of the multivariate normal distribution (with one response variable) can be written in matrix form as

$$f(\boldsymbol{z}|\boldsymbol{\beta},\boldsymbol{\Sigma}) = \frac{\exp\left\{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right\}}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}},$$

$$\text{where} \quad \boldsymbol{z} = \boldsymbol{y}^{(\boldsymbol{\lambda})}.$$

If linearity of the structure for $E(\boldsymbol{y})$ (i.e. $\boldsymbol{\mu} = \boldsymbol{X\beta}$) and homogeneity of variance (i.e. $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$) are satisfied, then a slightly different density function emerges:

$$f(\boldsymbol{z}|\boldsymbol{\beta},\sigma^2) = \frac{\exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{z}-\boldsymbol{X\beta})'(\boldsymbol{z}-\boldsymbol{X\beta})\right\}}{(2\pi\sigma^2)^{\frac{n}{2}}}.$$

But, the pdf of $\boldsymbol{y}$, not $\boldsymbol{z}$ is needed. By multiplying the pdf of $\boldsymbol{z}$ by the Jacobian, the pdf of $\boldsymbol{y}$ is found to be

$$f(\boldsymbol{y}|\boldsymbol{\beta},\sigma^2,\lambda) = \frac{\exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{z}-\boldsymbol{X\beta})'(\boldsymbol{z}-\boldsymbol{X\beta})\right\}}{(2\pi\sigma^2)^{\frac{n}{2}}}\prod_{i=1}^{n} y_i^{\lambda-1}.$$

Consequently, the log likelihood function is

$$\ln L(\boldsymbol{\beta}, \sigma^2, \lambda|\boldsymbol{y}) = -\frac{1}{2\sigma^2}(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta}) - \frac{n}{2}\ln(2\pi\sigma^2) + (\lambda - 1)\sum_{i=1}^{n}\ln y_i.$$

The mles of $\boldsymbol{\beta}$, σ^2, and λ are found by maximizing the log likelihood function. Finding the values of the parameters that maximize the log likelihood is easily done by doing the maximization separately for each parameter and substituting its mle into the log likelihood function.

First, the mle of $\boldsymbol{\beta}$ will be found. Taking the derivative of $\ln L$ with respect to $\boldsymbol{\beta}$, setting it equal to zero, and solving for $\boldsymbol{\beta}$ yields

$$\begin{aligned}\frac{\partial \ln L(\boldsymbol{\beta}, \sigma^2, \lambda|\boldsymbol{y})}{\partial \boldsymbol{\beta}} &= -\frac{1}{2\sigma^2}(-2\boldsymbol{X}'\boldsymbol{z} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}) \overset{\text{set}}{=} \boldsymbol{0} \\ &\implies 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta} = 2\boldsymbol{X}'\boldsymbol{z} \\ &\implies \widetilde{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-}\boldsymbol{X}'\boldsymbol{z},\end{aligned}$$

where $(\boldsymbol{X}'\boldsymbol{X})^{-}$ is any generalized inverse of $\boldsymbol{X}'\boldsymbol{X}$. Substituting $\widetilde{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$ into the log likelihood function yields

$$\begin{aligned}\ln L(\sigma^2, \lambda|\boldsymbol{y}, \widetilde{\boldsymbol{\beta}}) &= -\frac{1}{2\sigma^2}(\boldsymbol{z} - \boldsymbol{H}\boldsymbol{z})'(\boldsymbol{z} - \boldsymbol{H}\boldsymbol{z}) - \frac{n}{2}\ln(2\pi\sigma^2) + (\lambda - 1)\sum_{i=1}^{n}\ln y_i \quad (2) \\ &= -\frac{1}{2\sigma^2}\boldsymbol{z}'(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{z} - \frac{n}{2}\ln(2\pi\sigma^2) + (\lambda - 1)\sum_{i=1}^{n}\ln y_i, \quad (3)\end{aligned}$$

where $H = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-}\boldsymbol{X}'$.

Taking the derivative of (2) with respect to σ^2 and solving the resulting equation yields

$$\begin{aligned}\frac{\partial \ln L(\sigma^2, \lambda|\boldsymbol{y}\widetilde{\boldsymbol{\beta}})}{\partial \sigma^2} &= -\frac{1}{2\sigma^4}\boldsymbol{z}'(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{z} - \frac{n}{2\sigma^2} \overset{\text{set}}{=} \boldsymbol{0} \\ &\implies \frac{\boldsymbol{z}'(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{z}}{\sigma^4} = \frac{n}{\sigma^2} \\ &\implies \hat{\sigma}^2 = \frac{\boldsymbol{z}'(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{z}}{n}.\end{aligned}$$

Accordingly, the concentrated log likelihood function is

$$\begin{aligned}\ln L(\lambda|\boldsymbol{y}) = \ln L(\lambda|\boldsymbol{y}, \widetilde{\boldsymbol{\beta}}, \hat{\sigma}^2) &= -\frac{1}{2}\frac{n\hat{\sigma}^2}{\hat{\sigma}^2} - \frac{n}{2}\ln(2\pi\hat{\sigma}^2) + (\lambda - 1)\sum_{i=1}^{n}\ln y_i \\ &= -\frac{n}{2}\ln(2\pi e) - \frac{n}{2}\ln(\hat{\sigma}^2(\boldsymbol{z})) + (\lambda - 1)\sum_{i=1}^{n}\ln y_i. \quad (4)\end{aligned}$$

Equation (4) represents a partially maximized log likelihood function that depends only on λ. Therefore, maximizing (4) with respect to λ will complete the estimation problem. The derivative of $\ln L(\lambda|\boldsymbol{y})$ can be written as

$$\frac{\partial \ln L(\lambda|\boldsymbol{y})}{\partial\lambda} = -\frac{n}{2\hat{\sigma}^2}\frac{\partial\hat{\sigma}^2(\boldsymbol{z})}{\partial\lambda} + \sum_{i=1}^{n}\ln y_i,$$

which, using the chain rule for vectors, can be simplified as

$$\frac{\partial \ln L(\lambda|\boldsymbol{y})}{\partial\lambda} = -\frac{n}{2\hat{\sigma}^2}\left(\frac{\partial\boldsymbol{z}'}{\partial\lambda}\frac{\partial\hat{\sigma}^2(\boldsymbol{z})}{\partial\boldsymbol{z}}\right) + \sum_{i=1}^{n}\ln y_i.$$

Expressions for $\frac{\partial\boldsymbol{z}'}{\partial\lambda}$ and $\frac{\partial\hat{\sigma}^2(\boldsymbol{z})}{\partial\boldsymbol{z}}$ are necessary to simplify the above expression. Because $\boldsymbol{z} = \boldsymbol{y}^{(\boldsymbol{\lambda})}$, it follows that

$$\frac{\partial z_i}{\partial\lambda} = u_i = \begin{cases} \dfrac{\lambda y_i^\lambda \ln y_i - y_i^\lambda + 1}{\lambda^2} & \text{if } \lambda \neq 0; \\ \dfrac{(\ln y_i)^2}{2} & \text{if } \lambda = 0 \end{cases} \tag{5}$$

where, as shown in appendix A.3,

$$\frac{(\ln y_i)^2}{2} = \lim_{\lambda\to 0}\frac{\partial z_i}{\partial\lambda}.$$

It is convenient to arrange these derivatives into a vector:

$$\frac{\partial\boldsymbol{z}}{\partial\lambda} = \boldsymbol{u},$$

where $\boldsymbol{u}$ is an $n \times 1$ vector consisting of $\{u_i\}$. The derivative $\frac{\partial\hat{\sigma}^2(\lambda)}{\partial\boldsymbol{z}}$ can be found by using the rule of quadratic forms found in chapter 8 of Schott [10]. The result is

$$\frac{\partial\hat{\sigma}^2(\boldsymbol{z})}{\partial\boldsymbol{z}} = \frac{1}{n}\frac{\partial\boldsymbol{z}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{\partial\boldsymbol{z}} = \frac{2(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{n}.$$

Combining the preceding gives

$$\begin{aligned} \frac{\partial\hat{\sigma}^2(\boldsymbol{z})}{\partial\lambda} &= \frac{\partial\boldsymbol{z}'}{\partial\lambda}\frac{\partial\hat{\sigma}^2(\boldsymbol{z})}{\partial\boldsymbol{z}} \\ &= \boldsymbol{u}'\left(\frac{2(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{n}\right) \\ &= \frac{2\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{n}. \end{aligned}$$

In summary,

$$\frac{\partial \ln L(\lambda|\boldsymbol{y})}{\partial\lambda} = -\frac{\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{\hat{\sigma}^2} + \sum_{i=1}^{n}\ln y_i. \tag{6}$$

Solving for λ proves to be difficult. Apparently, there is no closed form solution to the likelihood equation $\frac{\partial \ln L(\lambda|\boldsymbol{y})}{\partial\lambda} = 0$. Therefore, iterative techniques need to be used. One such method is the Newton-Raphson algorithm.

3 Newton-Raphson Algorithm

The Newton-Raphson algorithm is an iterative method used to find an unconstrained maximum or minimum, or to solve an equation. A full motivation of the subject can be found in chapter 10 of Kennedy and Gentle [6]. A synopsis of the algorithm follows.

Consider a differentiable scalar function f of an $n \times 1$ vector $\boldsymbol{\theta}$. If an exact expression for the optimizer of $f(\boldsymbol{\theta})$ does not exist, then an approximation for the optimizer of $f(\boldsymbol{\theta})$ can be derived. First, expand $f(\boldsymbol{\theta})$ in a second order Taylor Series about the vector $\boldsymbol{\theta}_0$ (an initial guess):

$$f(\boldsymbol{\theta}) \quad \approx \quad f(\boldsymbol{\theta}_0) + g(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)' H(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0), \tag{7}$$

where the gradient function is

$$g(\boldsymbol{\theta}_0) = \left[\frac{\partial f}{\partial \boldsymbol{\theta}}\right]_{\boldsymbol{\theta}=\boldsymbol{\theta}_0},$$

and the Hessian matrix is

$$H(\boldsymbol{\theta}_0) = \left[\frac{\partial^2 f}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}'}\right]_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}.$$

An approximate optimizer of $f(\boldsymbol{\theta})$ can be found by taking the derivative of the Taylor approximation of $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, setting it equal to zero, and solving for $\boldsymbol{\theta}$.

By using vector calculus [10], the derivative of (7) becomes

$$\frac{\partial f}{\partial \boldsymbol{\theta}} = g(\boldsymbol{\theta}_0) + H(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \overset{\text{set}}{=} \mathbf{0}.$$

Solving for $\boldsymbol{\theta}$ gives

$$\boldsymbol{\theta} \quad = \quad \boldsymbol{\theta}_0 - H(\boldsymbol{\theta}_0)^{-1} g(\boldsymbol{\theta}_0). \tag{8}$$

By renaming $\boldsymbol{\theta}$ as $\boldsymbol{\theta}_1$ in equation (8), an iterative process can be established, and is

$$\boldsymbol{\theta}_{k+1} \quad = \quad \boldsymbol{\theta}_k - H(\boldsymbol{\theta}_k)^{-1} g(\boldsymbol{\theta}_k), \tag{9}$$

where $\boldsymbol{\theta}_k$ is the k^{th} iteration and $\boldsymbol{\theta}_{k+1}$ is the $(k+1)^{\text{st}}$ iteration. Equation (9) should be repeatedly evaluated until convergence to the solution $\widehat{\boldsymbol{\theta}}$. Convergence will occur at a quadratic rate, which is one reason that this algorithm is often used. Nonetheless, if the initial guess is "too far" from the optimizer, then convergence may not occur. So, an "educated", rather than arbitrary guess should be used. Once convergence has occurred, the Hessian matrix can be used to determine whether $\widehat{\boldsymbol{\theta}}$ is a local minimizer or maximizer. If $H(\widehat{\boldsymbol{\theta}})$ is positive definite, then $\widehat{\boldsymbol{\theta}}$ is a minimizer of $f(\boldsymbol{\theta})$; and if $-H(\widehat{\boldsymbol{\theta}})$ is positive definite, then $\widehat{\boldsymbol{\theta}}$ is a maximizer.

Applying this general result to finding mles is simple. All that is necessary is identifying $f(\boldsymbol{\theta})$. Either the concentrated likelihood function $L(\lambda|\boldsymbol{y})$, or the concentrated log likelihood function $\ln L(\lambda|\boldsymbol{y})$ are normally used for $f(\boldsymbol{\theta})$. In this case, estimation proceeds using $f(\boldsymbol{\theta}) = \ln L(\lambda|\boldsymbol{y})$.

Note that $g(\lambda)$ already has been derived in (6). The Hessian matrix is the only remaining component that is needed to apply the Newton-Raphson method.

Before proceeding, two preliminary results are needed. First, define matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ with sizes of $m \times l$, $l \times r$, and $p \times q$ respectively. Then the derivative of the product $\boldsymbol{AB}$ with respect to $\boldsymbol{C}$ is the $mp \times rq$ matrix

$$\frac{\partial(\boldsymbol{AB})}{\partial\boldsymbol{C}} = \left(\frac{\partial\boldsymbol{A}}{\partial\boldsymbol{C}}\right)(\boldsymbol{I}_q \otimes \boldsymbol{B}) + (\boldsymbol{I}_p \otimes \boldsymbol{A})\left(\frac{\partial\boldsymbol{B}}{\partial\boldsymbol{C}}\right). \tag{10}$$

Second, the derivative of the inverse of an $v \times v$ nonsingular matrix, $\boldsymbol{A}$, whose entries are functions of a $p \times q$ matrix $\boldsymbol{C}$ is

$$\frac{\partial\boldsymbol{A}^{-1}}{\partial\boldsymbol{C}} = -(\boldsymbol{I}_p \otimes \boldsymbol{A}^{-1})\left(\frac{\partial\boldsymbol{A}}{\partial\boldsymbol{C}}\right)(\boldsymbol{I}_q \otimes \boldsymbol{A}^{-1}). \tag{11}$$

Using (10), the derivative of (6) with respect to λ' is found by identifying

$$\boldsymbol{A} = -\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z},\ \boldsymbol{B} = (\hat{\sigma}^2)^{-1},\ \text{and } \boldsymbol{C} = \lambda' = \lambda,$$

and $m = l = r = p = q = 1$. Consequently,

$$\begin{aligned}\frac{\partial^2 \ln L(\lambda|\boldsymbol{y})}{\partial\lambda^2} &= \left(\frac{\partial\boldsymbol{A}}{\partial\boldsymbol{C}}\right)\boldsymbol{B} + \boldsymbol{A}\left(\frac{\partial\boldsymbol{B}}{\partial\boldsymbol{C}}\right) \\ &= \left(\frac{\partial\left[-\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}\right]}{\partial\lambda}\right)(\hat{\sigma}^2)^{-1} - \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}\left(\frac{\partial(\hat{\sigma}^2)^{-1}}{\partial\lambda}\right). \qquad (12)\end{aligned}$$

Now, each individual derivative in (12) needs to be computed. To evaluate $\frac{\partial(\hat{\sigma}^2)^{-1}}{\partial\lambda}$, equation (11) is used by identifying

$$\boldsymbol{A} = (\hat{\sigma}^2)^{-1} \text{ and } \boldsymbol{C} = \lambda.$$

Therefore,

$$\begin{aligned}\frac{\partial(\hat{\sigma}^2)^{-1}}{\partial\lambda} &= -(\hat{\sigma}^2)^{-1}\left(\frac{\partial\hat{\sigma}^2}{\partial\lambda}\right)(\hat{\sigma}^2)^{-1} \\ &= -\frac{1}{(\hat{\sigma}^2)^2}\left(\frac{2\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{n}\right) \\ &= -\frac{2\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{n(\hat{\sigma}^2)^2}.\end{aligned}$$

Finding an expression for $\dfrac{\partial[-\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}]}{\partial\lambda}$ involves the use of (10). Note that

$$\boldsymbol{A} = -\boldsymbol{u}',\ \boldsymbol{B} = (\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z},\ \boldsymbol{C} = \lambda' = \lambda,$$

$m = r = p = q = 1$, and $l = n$. As a result,

$$\begin{aligned}\frac{\partial\left[-\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}\right]}{\partial\lambda} &= \left(-\frac{\partial\boldsymbol{u}'}{\partial\lambda}\right)(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z} - \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\left(\frac{\partial\boldsymbol{z}}{\partial\lambda}\right)\\ &= \boldsymbol{v}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z} - \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{u},\end{aligned}$$

where $\boldsymbol{v}$ is the $n \times 1$ vector containing

$$v_i = -\frac{\partial\boldsymbol{u}_i}{\partial\lambda} = \begin{cases}\dfrac{2-\left[(\lambda\ln y_i)^2 - 2\lambda\ln y_i + 2\right]y_i^\lambda}{\lambda^3} & \text{if } \lambda \neq 0;\\ -\dfrac{(\ln y_i)^3}{3} & \text{if } \lambda = 0,\end{cases} \tag{13}$$

and, as shown in appendix A.3,

$$\frac{(\ln y_i)^3}{3} = -\lim_{\lambda\to 0}\frac{\partial u_i}{\partial\lambda}.$$

In summary,

$$\begin{aligned}\frac{\partial^2\ln L(\lambda|\boldsymbol{y})}{\partial\lambda^2} &= \left(\frac{\partial\left[-\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}\right]}{\partial\lambda}\right)(\hat{\sigma}^2)^{-1} - \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}\left(\frac{\partial(\hat{\sigma}^2)^{-1}}{\partial\lambda}\right)\\ &= \frac{\boldsymbol{v}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z} - \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{u}}{\hat{\sigma}^2} + \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}\left(\frac{2\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{n(\hat{\sigma}^2)^2}\right).\end{aligned} \tag{14}$$

After simplifying (14), the equation of the Hessian can be written as

$$H(\lambda) = \frac{\partial^2\ln L}{\partial\lambda^2} = \frac{\boldsymbol{v}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z} - \boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{u}}{\hat{\sigma}^2} + \frac{2}{n}\left(\frac{\boldsymbol{u}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{z}}{\hat{\sigma}^2}\right)^2. \tag{15}$$

In conclusion, the mle of λ can be found using the Newton-Raphson algorithm (9), with the gradient defined in (6), and the Hessian defined in (15).

4 Confidence Intervals for λ

While there are a variety of methods to find confidence intervals for parameters, only two are discussed in this paper. One is based on the asymptotic distribution of mles, whereas the other is based on the inversion of the LR Test. Both yield approximate $(1-\alpha)100\%$ confidence intervals.

4.1 Asymptotic Distribution of MLE

Finding a confidence interval using the asymptotic distribution of maximum likelihood estimators is similar to the pivotal method for finding confidence intervals for location parameters. Let $\widehat{\boldsymbol{\theta}}$ be the mle of $\boldsymbol{\theta}$, a $p\times 1$ vector of parameters. Because the distribution of $\widehat{\boldsymbol{\theta}}$ is asymptotically normal ([7], pg 198), an approximate $(1-\alpha)100\%$ confidence interval for one of the parameters in $\boldsymbol{\theta}$ can be found using

$$\hat{\theta}_i \pm z^{\star}\widehat{\mathrm{SE}}(\hat{\theta}_i),$$

where $z^{\star}$ is the $\left(1-\frac{\alpha}{2}\right)100^{\text{th}}$ percentile of the standard normal distribution, and $\widehat{\mathrm{SE}}(\hat{\theta}_i)$ is an estimate of the standard error of $\hat{\theta}_i$. Once the mle is known, the standard errors are all that remain in constructing the intervals. By finding the estimates to the parameters of the asymptotic distribution of $\widehat{\boldsymbol{\theta}}$, expressions for the standard errors can be found as the square root of the diagonal entries of the estimated covariance matrix. First, the distribution of $\widehat{\boldsymbol{\theta}}$ will be stated, followed by a derivation of the approximate asymptotic distribution of $\widehat{\lambda}$.

Designate the log likelihood function as $\ell(\boldsymbol{\theta} \mid \boldsymbol{y})$. Define $\boldsymbol{U}$ as

$$\boldsymbol{U} = \frac{\partial \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\theta}},$$

and denote the observed Fisher's total information matrix as

$$\widehat{\boldsymbol{J}}(\boldsymbol{\theta}) = -\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{\theta}'} = -\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}'}.$$

Then Fisher's total information matrix, $\boldsymbol{J}$, is defined as

$$\boldsymbol{J}(\boldsymbol{\theta}) = E[\widehat{\boldsymbol{J}}(\boldsymbol{\theta})] = E(\boldsymbol{U}\boldsymbol{U}').$$

The distribution of $\sqrt{n}(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})$, as $n \to \infty$, converges to a normal distribution. In particular,

$$\left[\frac{\boldsymbol{J}(\boldsymbol{\theta})}{n}\right]^{\frac{1}{2}} \sqrt{n}(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \xrightarrow{\text{dist}} N(\mathbf{0}, \boldsymbol{I}), \tag{16}$$

where $\boldsymbol{J}(\boldsymbol{\theta})$ is Fisher's total information matrix [7]. Nonetheless, $\boldsymbol{J}(\boldsymbol{\theta})$ is typically not known, yet by substituting an $n^{-\frac{1}{2}}$ consistent estimator, $\frac{\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})}{n}$ for $\frac{\boldsymbol{J}(\boldsymbol{\theta})}{n}$ in equation (16), the same result is obtained. Accordingly,

$$\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})^{\frac{1}{2}}(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \xrightarrow{\text{dist}} N(\mathbf{0}, \boldsymbol{I}) \text{ and } \widehat{\boldsymbol{\theta}} \mathrel{\dot\sim} N(\boldsymbol{\theta}, \widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})^{-1}).$$

In this paper,

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\beta} \\ \sigma^2 \\ \lambda \end{pmatrix}$$

and

$$
\begin{aligned}
-\widehat{\boldsymbol{J}}(\boldsymbol{\theta}) &= \begin{pmatrix} \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}'} & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}\, \partial \sigma^2} & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}\, \partial \lambda} \\ \left(\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}\, \partial \sigma^2}\right)' & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial (\sigma^2)^2} & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \sigma^2\, \partial \lambda} \\ \left(\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}\, \partial \lambda}\right)' & \left(\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \sigma^2\, \partial \lambda}\right)' & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \lambda^2} \end{pmatrix} \qquad (17) \\
&= \frac{1}{\sigma^2} \begin{pmatrix} \boldsymbol{X}'\boldsymbol{X} & \dfrac{\boldsymbol{X}'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^2} & -\boldsymbol{X}'\boldsymbol{u} \\ \dfrac{(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{X}}{\sigma^2} & \dfrac{(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^4} - \dfrac{n}{\sigma^2} & -\dfrac{\boldsymbol{u}'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^2} \\ -\boldsymbol{u}'\boldsymbol{X} & -\dfrac{(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{u}}{\sigma^2} & \boldsymbol{u}'\boldsymbol{u} - \boldsymbol{v}'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta}) \end{pmatrix}.
\end{aligned}
$$

Appendix A.4 gives details about the expression for $\widehat{\boldsymbol{J}}(\boldsymbol{\theta})$.

The square root of the diagonal entries of $\widehat{\boldsymbol{J}}(\boldsymbol{\theta})^{-1}$ is an estimate of the standard error for each of the estimators. An estimator for the standard error of $\widehat{\lambda}$ is of particular interest and can be found using inverses of partitioned matrices. First, partition $-\widehat{\boldsymbol{J}}(\boldsymbol{\theta})$ as

$$
\begin{aligned}
-\widehat{\boldsymbol{J}}(\boldsymbol{\theta}) &= \frac{1}{\sigma^2} \left(\begin{array}{cc|c} \boldsymbol{X}'\boldsymbol{X} & \dfrac{\boldsymbol{X}'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^2} & -\boldsymbol{X}'\boldsymbol{u} \\ \dfrac{(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{X}}{\sigma^2} & \dfrac{(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^4} - \dfrac{n}{\sigma^2} & -\dfrac{\boldsymbol{u}'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^2} \\ \hline -\boldsymbol{u}'\boldsymbol{X} & -\dfrac{(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{u}}{\sigma^2} & \boldsymbol{u}'\boldsymbol{u} - \boldsymbol{v}'(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta}) \end{array}\right) \\
&= \frac{1}{\sigma^2} \begin{pmatrix} \widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{11} & \widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{12} \\ \widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{21} & \widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{22} \end{pmatrix}.
\end{aligned}
$$

Then one estimator for $\mathrm{Var}(\widehat{\lambda})$ is

$$
\widehat{\mathrm{Var}}(\widehat{\lambda}) = \sigma^2 \left(\widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{22} - \widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{21} \left[\widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{11} \right]^{-1} \widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{12} \right)^{-1}.
$$

Two problems exists with this estimator of $\mathrm{Var}(\widehat{\lambda})$. First, $\widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{11}$ is singular, and second, each $\widehat{\boldsymbol{J}}(\boldsymbol{\theta})_{ij}$ matrix depends on unknown parameters. The latter problem can averted by substituting the mles for $\boldsymbol{\beta}$ and σ^2. The inverse of $\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11}$ does not exist, but a generalized inverse does exist and $\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{21} \left[\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11} \right]^{-} \widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{12}$ does not depend on the choice of the generalized inverse. Simplifications to $\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11}$ occur because

$$
\frac{\boldsymbol{X}'(\boldsymbol{z} - \boldsymbol{X}\widetilde{\boldsymbol{\beta}})}{\hat{\sigma}^2} = \frac{\overbrace{\boldsymbol{X}'(\boldsymbol{I} - \boldsymbol{H})}^{\mathbf{0}}\, \boldsymbol{z}}{\hat{\sigma}^2} = \mathbf{0},
$$

and

$$\frac{(\boldsymbol{z}-\boldsymbol{X}\widetilde{\boldsymbol{\beta}})'(\boldsymbol{z}-\boldsymbol{X}\widetilde{\boldsymbol{\beta}})}{(\hat{\sigma}^2)^2}-\frac{n}{2\hat{\sigma}^2}=\frac{1}{\hat{\sigma}^2}\left[\frac{n\hat{\sigma}^2}{\hat{\sigma}^2}-\frac{n}{2}\right]=\frac{n}{2\hat{\sigma}^2}.$$

Simplified, $\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11}$ becomes

$$\begin{pmatrix} \boldsymbol{X}'\boldsymbol{X} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{n}{2\hat{\sigma}^2} \end{pmatrix}.$$

One choice for $\left[\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11}\right]^-$ is

$$\begin{pmatrix} (\boldsymbol{X}'\boldsymbol{X})^- & \boldsymbol{0} \\ \boldsymbol{0} & \frac{2\hat{\sigma}^2}{n} \end{pmatrix}.$$

As a result, $\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{21}\left[\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11}\right]^-\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{12}$ simplifies to

$$\begin{aligned}
&= \begin{pmatrix} \hat{\boldsymbol{u}}'\boldsymbol{X} & \dfrac{\hat{\boldsymbol{z}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{u}}}{\hat{\sigma}^2} \end{pmatrix} \begin{pmatrix} (\boldsymbol{X}'\boldsymbol{X})^- & \boldsymbol{0} \\ \boldsymbol{0} & \dfrac{2\hat{\sigma}^2}{n} \end{pmatrix} \begin{pmatrix} \boldsymbol{X}'\hat{\boldsymbol{u}} \\ \dfrac{\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}}}{\hat{\sigma}^2} \end{pmatrix} \\
&= \begin{pmatrix} \hat{\boldsymbol{u}}'\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^- & \dfrac{2\hat{\boldsymbol{z}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{u}}}{n} \end{pmatrix} \begin{pmatrix} \boldsymbol{X}'\hat{\boldsymbol{u}} \\ \dfrac{\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}}}{\hat{\sigma}^2} \end{pmatrix} \\
&= \hat{\boldsymbol{u}}'\underbrace{\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^-\boldsymbol{X}'}_{\boldsymbol{H}}\hat{\boldsymbol{u}}+\frac{2(\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}})^2}{n\hat{\sigma}^2},
\end{aligned}$$

where $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{z}}$ are $\boldsymbol{u}$ and $\boldsymbol{z}$ of (5) and (1) in which $\widehat{\lambda}$ is substituted for λ.

Further, an explicit expression for $\widehat{\text{Var}}(\widehat{\lambda})$ can be derived:

$$\begin{aligned}
\widehat{\text{Var}}(\widehat{\lambda}) &= \hat{\sigma}^2\left(\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{22}-\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{21}\left[\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{11}\right]^-\widehat{\boldsymbol{J}}(\widehat{\boldsymbol{\theta}})_{12}\right)^{-1} \\
&= \hat{\sigma}^2\left[\hat{\boldsymbol{u}}'\hat{\boldsymbol{u}}-\hat{\boldsymbol{v}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}}-\left(\hat{\boldsymbol{u}}'\boldsymbol{H}\hat{\boldsymbol{u}}+\frac{2(\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}})^2}{n\hat{\sigma}^2}\right)\right]^{-1} \\
&= \hat{\sigma}^2\left[\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{u}}-\hat{\boldsymbol{v}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}}-\frac{2(\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}})^2}{n\hat{\sigma}^2}\right]^{-1} \\
&= -\left[\frac{\hat{\boldsymbol{v}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}}-\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{u}}}{\hat{\sigma}^2}+\frac{2}{n}\left(\frac{\hat{\boldsymbol{u}}'(\boldsymbol{I}-\boldsymbol{H})\hat{\boldsymbol{z}}}{\hat{\sigma}^2}\right)^2\right]^{-1} \\
&= -H(\widehat{\lambda})^{-1},
\end{aligned}$$

where $H(\widehat{\lambda})$ is given in (15). Accordingly,

$$\widehat{\mathrm{SE}}(\widehat{\lambda}) = \frac{1}{\sqrt{-H(\widehat{\lambda})}}.$$

In conclusion, an approximate $(1-\alpha)100\%$ confidence interval for λ can be found using

$$\widehat{\lambda} \pm z^{\star}\widehat{\mathrm{SE}}(\widehat{\lambda})$$

where $z^{\star}$ is the $\left(1-\frac{\alpha}{2}\right)100^{\text{th}}$ percentile of the standard normal distribution.

4.2 Inversion of LR Test

An alternative method for finding a confidence interval for λ is based on the inversion of the likelihood ratio test of $\mathrm{H}_o : \lambda = \lambda_0$ vs $\mathrm{H}_a : \lambda \neq \lambda_0$. All values of λ_0 which yield a *Fail to Reject* decision are included in the confidence interval.

To implement this inversion, the deviance statistic is used. For λ, the deviance is defined as

$$D(\lambda) = -2\left[\ell(\lambda \mid \boldsymbol{y}) - \ell(\widehat{\lambda} \mid \boldsymbol{y})\right],$$

where $\ell(\lambda \mid \boldsymbol{y})$ is the concentrated log likelihood function given in (4). The deviance has an asymptotic χ^2 distribution [7]. More specifically, if H_o is true, then

$$D(\lambda) \xrightarrow{\text{dist}} \chi^2_1,$$

as $n \to \infty$/ Thus, a $(1-\alpha)100\%$ confidence interval for λ can be found by solving

$$D(\lambda) \quad = \quad \chi^2_{1-\alpha,1} \tag{18}$$

for λ. A graphical description of the confidence interval is displayed in Figure 1.

The values of λ which solve (18) can be found using the Newton-Raphson algorithm for roots given in chapter 5 of Kennedy and Gentle [6]. The equation to solve is

$$f(\lambda) = \ell(\lambda \mid \boldsymbol{y}) - \ell(\widehat{\lambda} \mid \boldsymbol{y}) + \frac{1}{2}\chi^2_{1-\alpha,1} = 0.$$

Here $f'(\lambda) = \frac{\partial \ell(\lambda|\boldsymbol{y})}{\partial \lambda}$, and is given by equation (6). Thus, the iterative equation for solving for λ is

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)}.$$

To implement this algorithm, initial guesses are required. The endpoints of the confidence interval generated by the asymptotic distribution given in the last section work well.

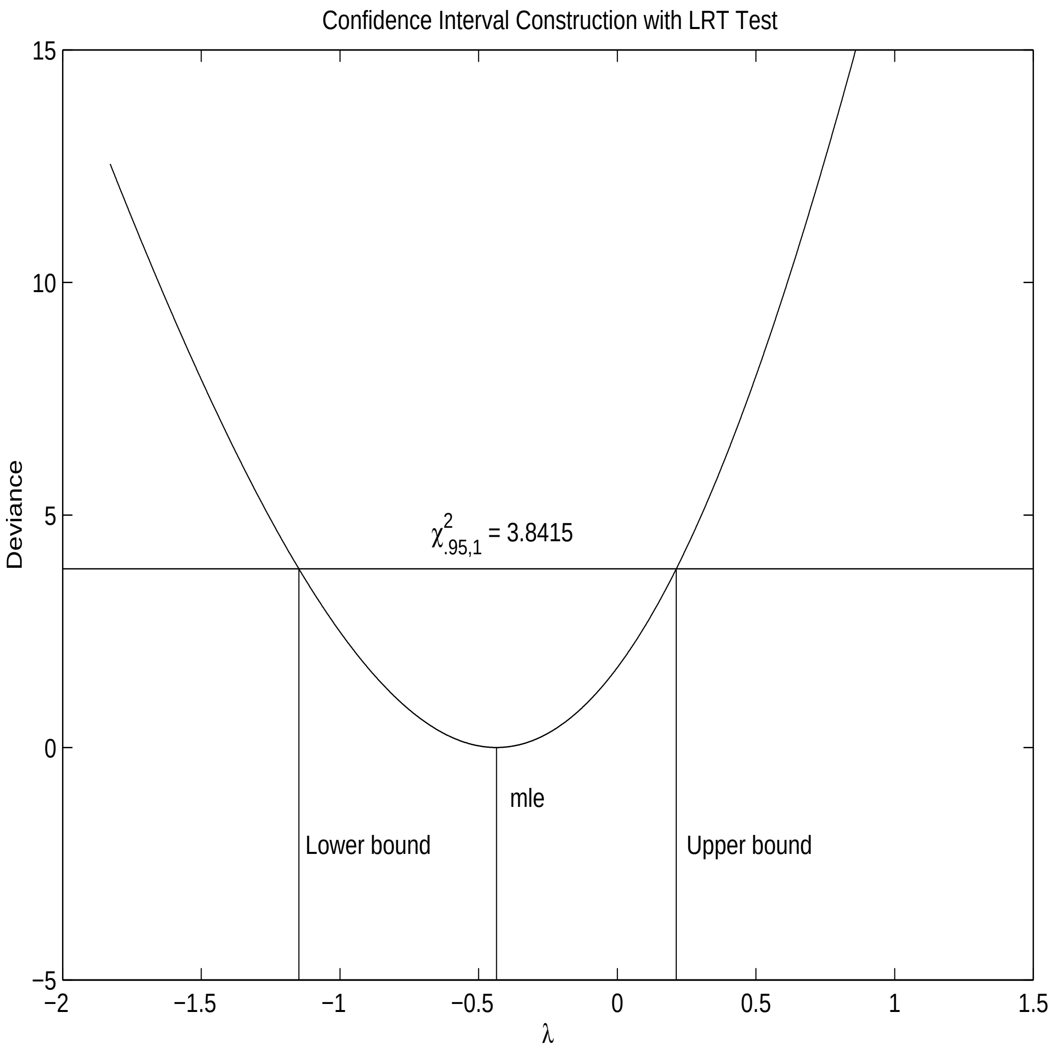

Figure 1: Matlab plot illustrating inversion of LRT confidence interval

5 Examples

To demonstrate the SAS and MATLAB programs, three data sets will be used. The first data set comes from Neter, Wasserman and Kutner [8]. Measurements of the plasma level of polyamine for 25 healthy children with varying ages were taken.

The general SAS program is given in Appendix A.1. To use the program for this data set, several modifications of the program are required. The correct filename and variable names in the `input` statement should be specified in the `data` step:

```
data in;
    infile 'dataplasma';
    input age plasma;
```

Next, the model should be specified in the `glmmod` procedure. The model is specified in the same manner as in the `glm` procedure. The model for the Neter data is a one way classification. The modified code is

```
proc glmmod outdesign=design noprint;
    class age;
    model plasma = age;
run;
```

The last thing to change or omit is the `title2` statement in the `gplot` procedure. The SAS Box-Cox transformation program produces the following output:

```
                           OUTPUT
                           ML Estimate   -0.43576
                           Lower Bound   -1.14820
                           Upper Bound    0.21272

        Bounds are based on an approximate  95% Confidence Interval

    Using the theoretical result that -2[loglik(lambda0) - loglik(mle)]

    is distributed approximately as a one degree of freedom chi-squared

    random variable when the null hypotheses (lambda=lambda0) is true.
```

Neter reported the mle for λ to be between $-.4$ and $-.5$, which is consistent with the SAS result. Because the 95% confidence interval given above does not contain 1, the LR test concludes that a transformation is needed. A plot of $\ln L(\lambda|\boldsymbol{y})$ versus λ is shown in Figure 2. The SAS code to produce this plot is in Appendix A.1.

The second example is a biological experiment using a 3×4 factorial design with replication, and can be found in Box and Cox [3]. The two factors in this experiment are poisons with three levels and treatments with four levels. Every combination of poison and treatments was assigned to four animals. The survival time for each animal was recorded. Because this is a two way classification with no interaction, the model statement is

```
proc glmmod outdesign=design noprint;
    class poison treatment;
    model survival = poison treatment;
run;
```

When run initially, the SAS program gives an overflow error and does not converge. Changing the initial guess enables the program to converge. From a starting point of $\lambda = .5$, SAS generates

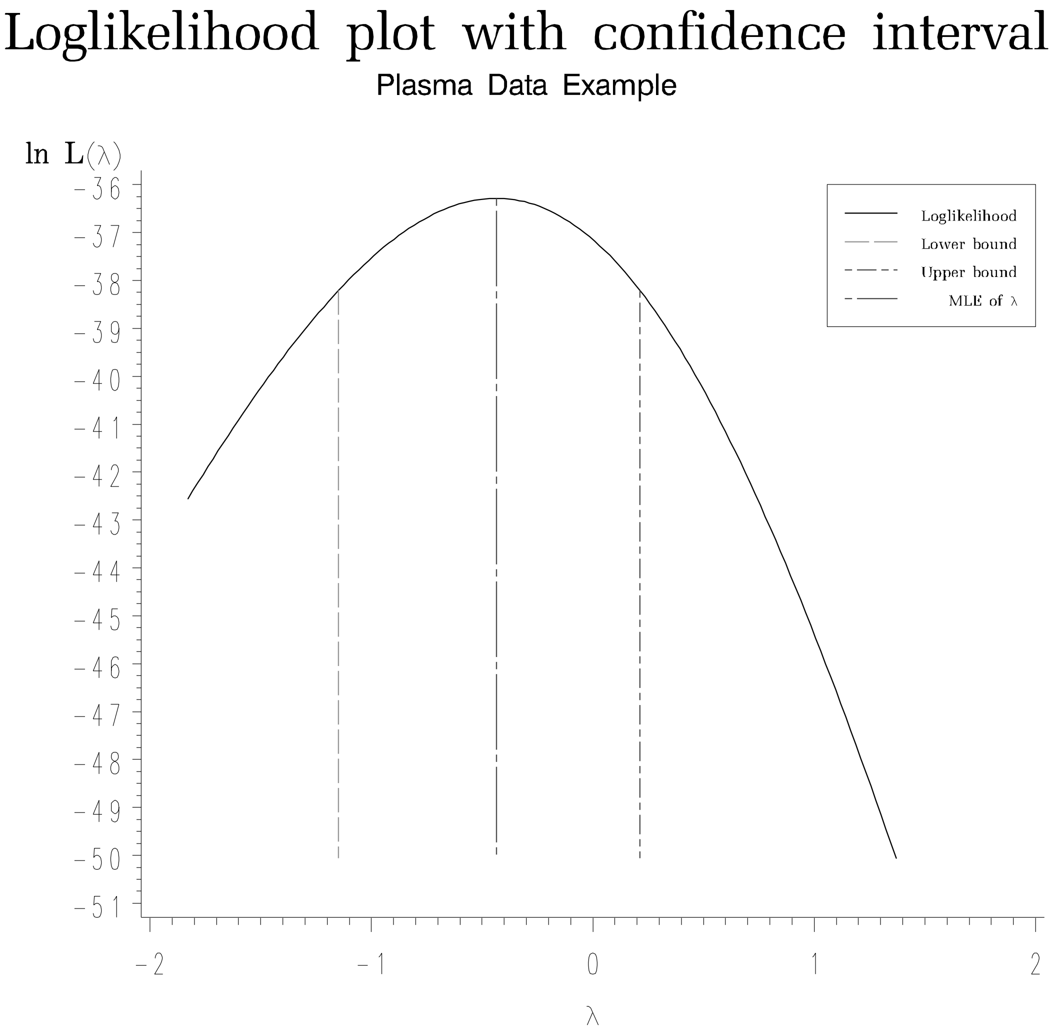

Figure 2: SAS plot for Plasma example

```
OUTPUT
ML Estimate   -0.75016
Lower Bound   -1.13803
Upper Bound   -0.35609
```

Box and Cox give similar results in their paper. In fact, Box and Cox mention that the estimate for λ agrees with the assumption that the response and explanatory variables should be inversely proportional. A plot of the log likelihood is shown in Figure 3.

The third data set is taken from Batchelder, Rogers, and Walker [1], but also was described in Boik and Marasinghe [2]. It is a three way classification with no third order interaction. The factors in this example are mulches (three levels), years (three levels), and types of soil (four levels). Tobacco yield for each combination of the factors was measured. SAS output for this example is

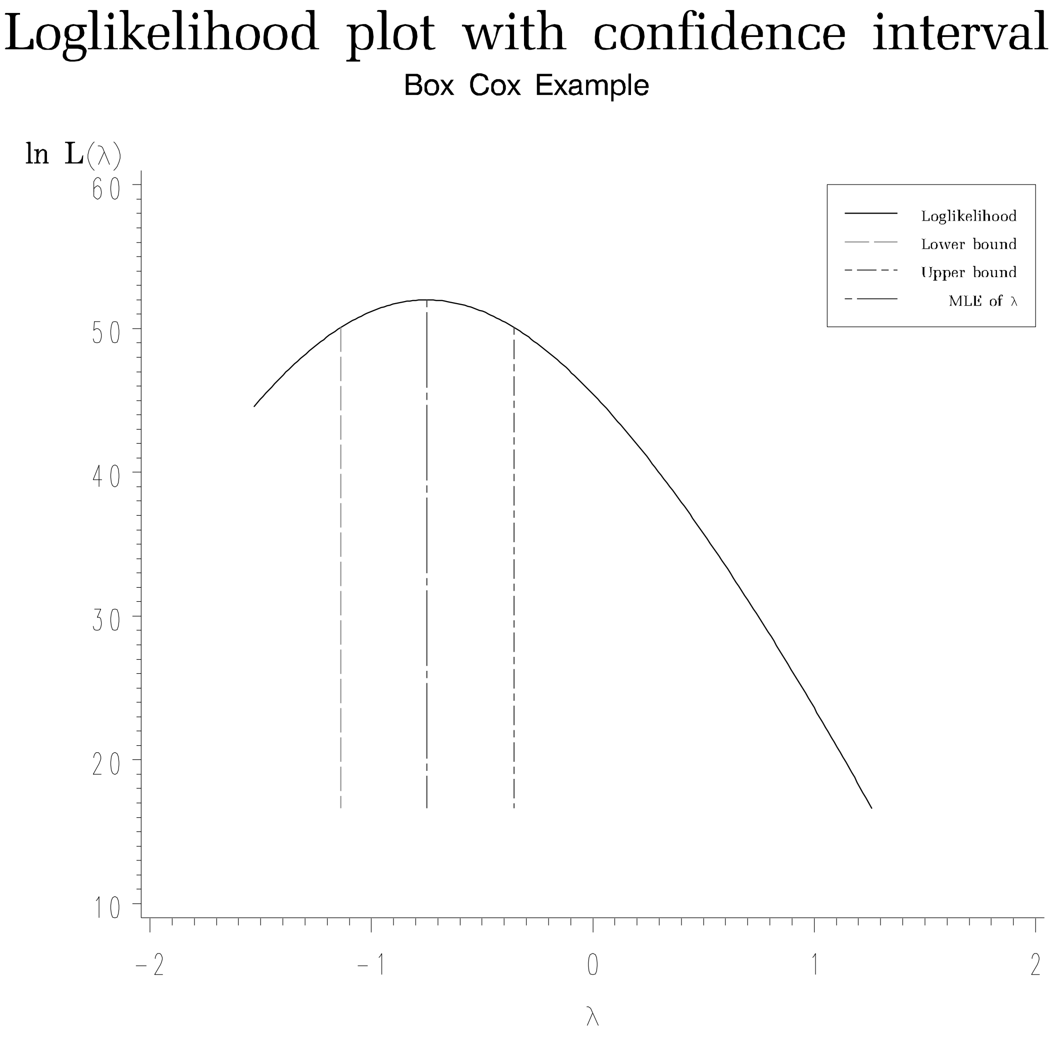

Figure 3: SAS plot for biological experiment

```
OUTPUT
ML Estimate    0.68951
Lower Bound    0.13732
Upper Bound    1.34040
```

This corresponds exactly to the results published in Boik and Marasinghe [2]. A graph of the log likelihood is displayed in Figure 4.

Using MATLAB is more difficult than using SAS because it is more tedious to specify the design matrix. Nonetheless, the matlab commands `dummyvar` and `x2fx` are helpful when generating design matrices. In fact, results for some simple models can easily be constructed. Figure 5 shows the corresponding output for the plasma data. MATLAB code is given in Appendix A.2.

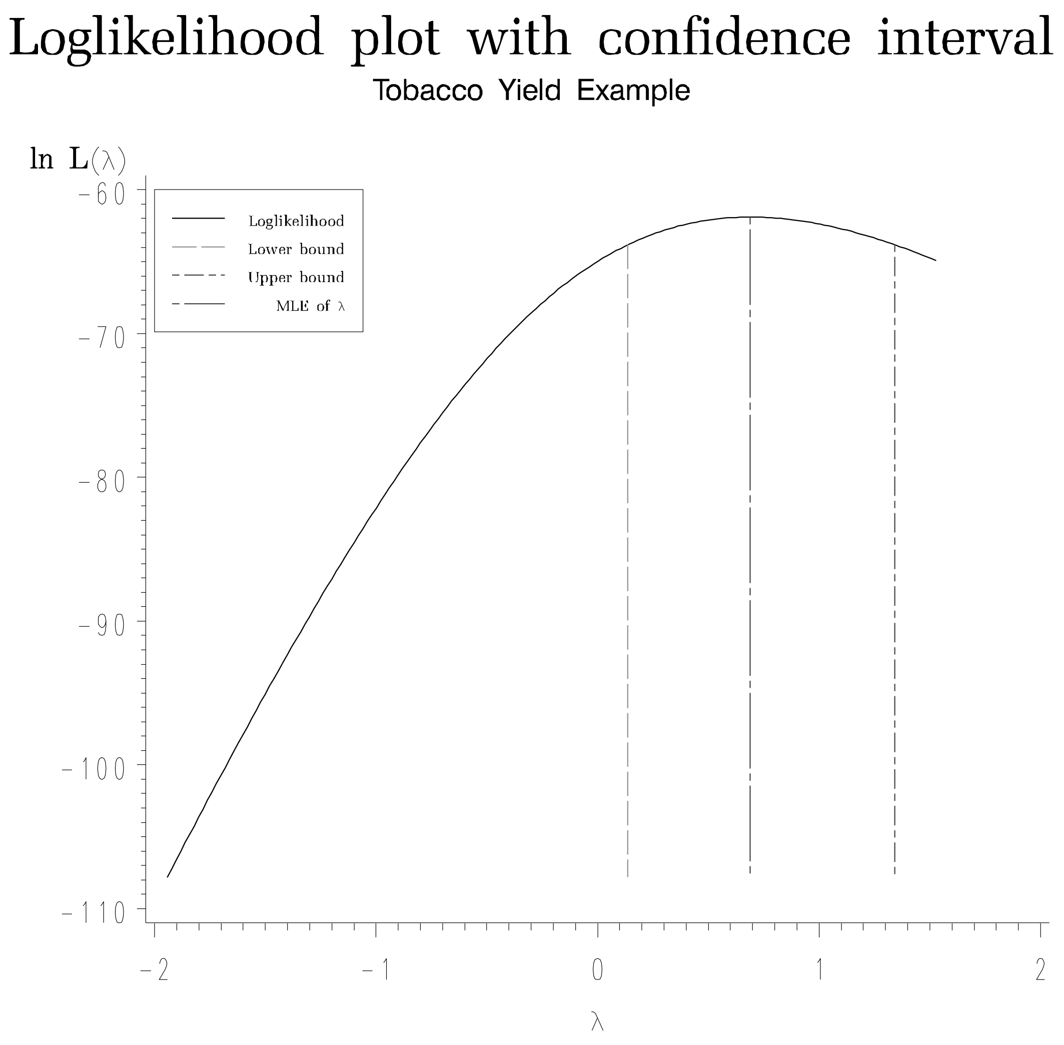

Figure 4: Sas plot for Tobacco example

6 Programming Notes

Several programming strategies were implemented to efficiently find the mle and the confidence interval.

First, the SAS program uses the `glmmod` procedure to produce the design matrix for the specified model. This makes it easier to use than MATLAB. The first variable in the design data set is the response variable, and the remaining variables are the columns of the design matrix. The `iml` procedure constructs the design matrix from this data set.

Second, there are several user adjustable variables which can be changed. The first is the initial guess for λ. Sometimes, the initial guess for λ should be different that 1 to ensure convergence (e.g., the Box Cox biological example). Additionally, the

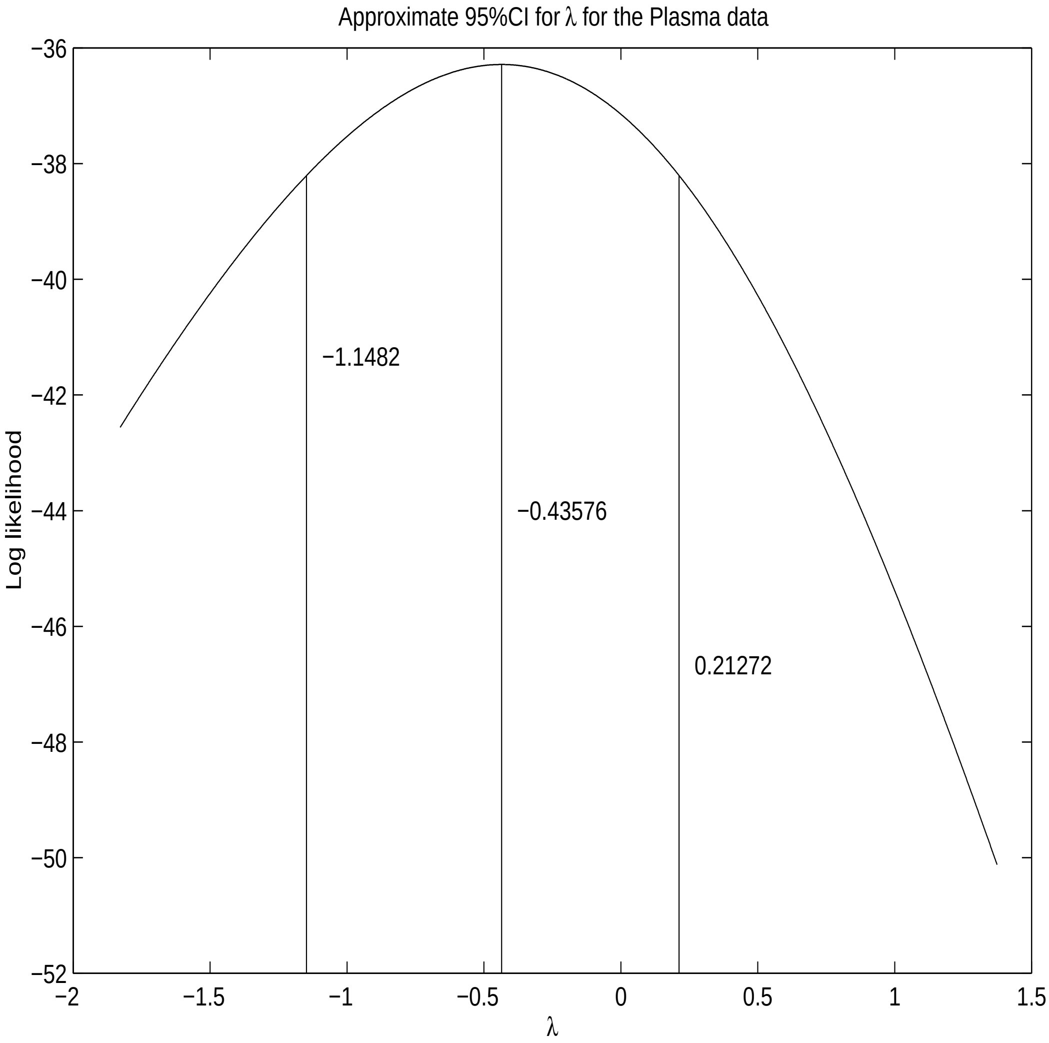

Figure 5: Matlab plot for Plasma data

confidence level can be changed if a confidence interval other than 95% is desired. The variables `small` and `tol` can be specified according to the user's preferences.

The last strategy implemented was using the full rank singular value decomposition to save on memory storage. Many of the quadratic forms derived include the matrix $\boldsymbol{I}-\boldsymbol{H}$ (e.g. $\boldsymbol{v}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{u}$). Rather than forming the $n\times n$ matrix $\boldsymbol{I}-\boldsymbol{H}$, $\boldsymbol{H}$ is decomposed into

$$\boldsymbol{H}=\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-}\boldsymbol{X}'=\boldsymbol{X}\overbrace{\boldsymbol{S}\boldsymbol{D}^{-1}\boldsymbol{S}'}^{(\boldsymbol{X}'\boldsymbol{X})^{-}}\boldsymbol{X}'=\overbrace{\boldsymbol{X}\boldsymbol{S}\boldsymbol{D}^{-\frac{1}{2}}}^{\boldsymbol{F}}\overbrace{\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{S}'\boldsymbol{X}'}^{\boldsymbol{F}'}=\boldsymbol{F}\boldsymbol{F}',$$

where $\boldsymbol{D}$ is an $r\times r$ diagonal matrix with non-negative entries, $\boldsymbol{S}$ is an $n\times r$ matrix, and $r=\operatorname{rank}(\boldsymbol{X})$. As a result, $\boldsymbol{F}$ is an $n\times r$ matrix. By factoring $\boldsymbol{H}$ this way, quadratic forms can be rewritten as

$$\boldsymbol{v}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{u}=\boldsymbol{v}'(\boldsymbol{I}-\boldsymbol{F}\boldsymbol{F}')\boldsymbol{u}=\boldsymbol{v}'\boldsymbol{u}-(\boldsymbol{v}'\boldsymbol{F})(\boldsymbol{F}'\boldsymbol{u}).$$

This reduces the storage space from n^2 to nr. Because r is typically much smaller than n, this saves a relatively large amount of space.

A Appendix

A.1 SAS Code

To execute this code on a different data set, the following changes are required. First, in the `data` statements, the `infile` filename, the `input` variable names, and the `model` specified in `proc glmmod` must be modified appropriately.

Second, the user defined variables can be changed. The initial guess for λ may need to be different than 1 in order for convergence of the estimate. Also, changes in the confidence level are made here.

Third, in the `gplot` procedure, any number of changes can be made, with the minimum being the `title2` statement, which should be suitably changed.

```
dm 'log;clear;out;clear;'; /*This clears the log and output windows*/

/*SAS program for finding the maximum likelihood estimate of the */
/*Box-Cox transformation parameter assuming normality of the     */
/*transformed data and homogeneity of variance                   */

/*===============================================================*/
data in;
    infile 'datatobacco';
    input mulch year soil response;

/* proc glmmod is used only to construct the design matrix, which*/
/* is used in proc iml.  You will need to specify the model in   */
/*proc glmod.  Only one response variable can be used.           */
proc glmmod outdesign=design noprint;
    class mulch year soil;
    model response = mulch year soil
                     mulch*year mulch*soil year*soil;
run;
/*===============================================================*/

proc iml;
  /*This section of the code breaks up the data set, design,     */
  /*into the response variable (y) and the design matrix (X)     */
  use design;
  names = contents();
  p = nrow(names);
  response = names[1,];
  dematrix = names[2:p,1];
  read all var response into y;
  read all var dematrix into X;
  n=nrow(X);

/*=======================================================================*/
  /*User Adjustable variables*/
  lam = 1;                      /*Initial guess for the mle of lambda.   */
  ConLevel = .95;               /*Specify the Confidence Level here.     */
  small = 1.e-10;               /*Numbers smaller than this are zero.    */
  tol = 1.e-10;                 /*Estimate error tolerances.             */
/*=======================================================================*/

  /* Constants used throughout the program */
  XPX = X‘*X;
  call svd(UU,QQ,V,XPX);
  r = max(loc(QQ>small));       /*rank of XPX*/
  U = UU[,1:r];
  Q = QQ[1:r];
  F = X*U*diag(Q##-.5);         /*Note that H = X*ginv(X‘*X)*X‘ = F*F‘*/
  free UU U QQ Q V XPX;         /*Free the space. These won‘t be used again.*/
  pi = arcos(-1);               /*This is pi*/
  logy = log(y);
  slogy = logy[+];              /*Sum of the logy‘s*/
  one = J(n,1);

  /* This loop generates both the maximum likelihood estimate of lambda*/
  /* and the ConLevel*100% confidence interval for lambda.             */
  do i = 1 to 3;
    step = 1;
    if i = 2 then
      do;
        const = -n/2*(log(2*pi) + 1)-slogy-loglik+cinv(ConLevel,1)/2;
        iguess = probit((1+ConLevel)/2)*sqrt(-1/H);
      end;
    /*This statement chooses the initial guesses for finding the lower*/
    /*and upper bound on lambda*/
    if i ^= 1 then
      if i=2 then lam = mlelam - iguess;
      else lam = mlelam + iguess;

    do until(abs(step) < tol);
      if abs(lam) < small then  /*This specifies the limit of Z, U, and V*/
        do;                     /*when lambda approaches zero (undefined */
          Z=logy;               /*otherwise).                            */
          U=logy##2/2;
          if i=1 then V=-logy##3/3;
        end;
      else                      /*Normal values for Z,U, and V when */
        do;                     /*lambda is not zero.               */
          ylam = y##lam;
          ly=lam*logy;
          Z=(ylam-one)/lam;
          U=((ly-one)#ylam + one)/lam##2;
          if i=1 then V=(2-(ly##2-2*ly+2*one)#ylam)/lam##3;
        end;

      /*F is used to save storage space.  Instead of n^2 storage*/
      /*space needed only n*r is needed where r=rank(X)         */
      fz = F`*Z;
      fu = F`*U;
      /*(ms) mle of the variance of the transformed data*/
      ms = (Z`*Z-fz`*fz)/n;     /*also Z`*(I-P)*Z\n   */
      JJ = (U`*Z-fu`*fz)/ms;    /*also U`*(I-P)*Z\ms  */

      g = -JJ + slogy;          /*First derivative of the loglik function*/
      if i=1 then
        do;
          /*Second Derivative (or the Hessian) of the loglik function (H)*/
          /*Part in parentheses easier to write as V`(I-P)Z-U`(I-P)*U    */
          H = ((V`*Z-V`*F*fz) - (U`*U-fu`*fu))/ms + 2/n*JJ##2;
          step = -g/H;          /*step for mle*/
        end;
      else step = -(const-n/2*log(ms)+lam*slogy)/g; /*step for CI estimate*/
      lam = lam + step;         /*Update estimate*/
    end;
    if i = 1 then mlelam=lam;
    else if i = 2 then lci=lam;
         else uci=lam;
    /*Save Loglikelihood for later use*/
    lglik = -n/2*(log(2*pi) + 1 + log(ms)) + (lam-1)*slogy;
    if i = 1 then loglik = lglik;
    else ciloglik = lglik;
  end;

  /*Print out the estimates for the mle and the CI of lambda*/
  output = mlelam // lci // uci;
  row = { "ML Estimate" "Lower Bound" "Upper Bound" };
  print output[rowname=row format=10.5];
  clevel=ConLevel*100;
  dum1 = 'Bounds are based on an approximate ';
  dum2 = char(clevel,2);
```

```
  dum3 = ’% Confidence Interval  ’;
  _ = concat(dum1,concat(dum2,dum3));
  print _;
  print ’Using the theoretical result that -2[loglik(lambda0) - loglik(mle)]’;
  print ’is distributed approximately as a one degree of freedom chi-squared’;
  print ’random variable when the null hypotheses (lambda=lambda0) is true. ’;

  /*The following statements form the data for a graph of the Log-likelihood */
  /*function along with the (ConLevel*100)% Conifidence Interval             */
  spread = (uci-lci)/6;         /* approximate spread  */
  maxci=max(abs(lci),abs(uci));
  lbs = -maxci-3*spread;        /* Lowerbound for plot */
  ubs = maxci+spread;           /* Upperbound for plot */
  ss = spread/10;               /* Stepsize for plot   */

  /*Generation of the data is here in the do loop*/
  do lam = lbs to ubs by ss;
    if abs(lam) < small then
      Z = logy;
    else
      Z = (y##lam-1)/lam;
    ms = (Z‘*Z-Z‘*F*F‘*Z)/n;        /*ms=Z‘(I-H)Z\n*/
    lglik = -n/2*(log(2*pi) + 1 + log(ms)) + (lam-1)*slogy;
    dummy = dummy // ( lam || lglik || 1 );
  end;

  /*The next four lines form the data for plotting the confidence interval*/
  /*and the mle                                                           */
  miny = min(dummy[,2]);
  dummy = dummy // ( lci || ciloglik || 2 ) // (lci || miny || 2);
  dummy = dummy // ( uci || ciloglik || 3 ) // (uci || miny || 3);
  dummy = dummy // ( mlelam || loglik || 4) // (mlelam || miny || 4);
  create plotdata from dummy [ colname = { Lambda LogLikel graph } ];
  append from dummy;
  close plotdata;

/*This is the main SAS code for doing the Plot of the */
/*Log Likelihood Function*/
/* goptions device=psepsf gsfname=temp gsflen=132 hsize=6in vsize=6in
    gsfmode=replace;  */
symbol1 color=black interpol=join width=2 value=none height=3;
symbol2 color=green interpol=join width=2 value=none line=4 height=3;
symbol3 color=red interpol=join width=2 value=none line=8 height=3;
symbol4 color=red interpol=join width=2 value=none line=10 height=3;
axis1 label=(font=greek ’l’);
```

```
axis2 label=(font=centx 'ln L' font=greek '(l)');
legend across=1 position=(top inside left) label=none cborder=black
  value=(font=zapf height=.5 tick=1 'Loglikelihood' tick=2 'Lower bound'
  tick=3 'Upper bound' tick=4 'MLE of ' font=greek 'l') mode=protect
  shape=symbol(3,1);
proc gplot data=plotdata;
  plot LogLikel*Lambda=graph /legend=legend1 haxis=axis1 vaxis=axis2;
  title font=zapf 'Loglikelihood plot with confidence interval';
/*================================================================*/
  title2 font=swiss 'Subtitle';
/*================================================================*/
run;
quit;
```

A.2 MATLAB Code

The MATLAB function below requires the design matrix, the response variable, and the confidence level. A fourth input argument can be given to change the initial guess for λ. If left out, $\lambda = 1$ will be used. The function returns $\widehat{\lambda}$, the upper and lower endpoints of the confidence interval, $\ln L(\widehat{\lambda})$, plus the transformed estimates of the response, $\boldsymbol{\beta}$, and σ^2. The MATLAB command

```
[mlelam,lci,uci] = box(X,y,ConLevel)
```

will return only λ, and the confidence intervals. The function also plots the log likelihood function.

```
function [mlelam,lci,uci,loglik,Zmle,Bmle,mSmle] = box(X,y,ConLevel,lam)
% Scott Hyde  December 27, 1998
% Matlab program for finding the maximum likelihood estimate of the
% Box-cox transformation parameter assuming normality of the transformed data
% and homogeneity of variance
%
% Inputs
%    X        (the design matrix)
%    y        (the data vector (response variable))
%    ConLevel (Specify the (Conlevel*100)% confidence level to be found
%    lam      (Initial guess for mle) (if not given, default is 1)
%
% Outputs
%    mlelam
%    lci, uci (upper and lower limits for an approximate ConLevel CI for lambda
%    loglik   (the associated log likelihood)
%    Zmle     (transformed data)
%    Bmle     (transformed estimates of the parameters)
%    mSmle    (transformed common variance)
%
[n,p] = size(y);

% User Adjustable variables
if nargin == 3,
  lam = 1;                           %Initial guess for mle of lambda if not
end;                                 %specified.
small = 1.e-10;                      %Numbers smaller than this are zero.
tol = 1.e-10;                        %Number of decimal places of accuracy.

%Constants used everywhere
XPX = X'*X;
```

```
[U,Q,V] = svd(XPX);
q=diag(Q);
r = rank(XPX);
G = U(:,1:r)*diag(q(1:r).^-.5);      %Note that pinv(X’*X) = G*G’;
F=X*G;                               %Note that H = X*pinv(X’*X)*X’ = F*F’
logy = log(y);
slogy = sum(logy);
%----------------------------------------------------------------------
%This loop generates both the maximum likelihood estimate of lambda
%and the approximate ConLevel*100% confidence interval for lambda
%(generated by inverting the likelihood ratio test).
%----------------------------------------------------------------------
for i = 1:3
  step = 1;
  if i == 2
    %const includes the parts of the deviance that are constant, which
    %makes programming the solution to -2[l(lam) - l(mle(lam))] =
    %chi2inv(Conlevel,1) easier. Chi2inv(Conlevel,1) is the ConLevel th
    %percentile of the Chi-squared dist with 1 df.

    const = -n/2*(log(2*pi)+1)-slogy-loglik+chi2inv(ConLevel,1)/2;

    %Since (mlelam - lam) is approx dist as N(0,v) where v=-1/H, then
    %good starting values to begin the iteration to find the confidence
    %interval are mlelam +- cv*sqrt(v)

    iguess = norminv((1+ConLevel)/2,0,1)*sqrt(-1/H);
  end;

  if i ~= 1
    if i == 2
      lam = mlelam - iguess;     %Initial guess for lower confidence bound
    else
      lam = mlelam + iguess;     %Initial guess for upper confidence bound
    end;
  end;

  while abs(step) > tol,

    %preliminary computations
    if abs(lam) < small
      Z = logy;                    %These specify the limit of Z, U, and V
      U = logy.^2/2;               %when lamda approaches zero. (Undefined
      if i==1                      %otherwise)
        V = logy.^3/3;
      end;
    else
      Z = (y.^lam-1)/lam;          %When lamda is not zero, these are the
      ly = lam*logy;               %computations for Z, U, and V.
      U = ((ly-1).*y.^lam + 1)/lam.^2;
      if i==1
        V = (2-(ly.^2-2*ly+2).*y.^lam)/lam.^3;
      end;
    end;

    %F is used to save storage space.  Instead of n^2 storage space needed
    %only n*r is needed where r=rank(X). (r is usually substatially less
    %than n)
    fz = F'*Z;
    fu = F'*U;

    %mS is the mle of the variance of the transformed data.
    mS = (Z'*Z-fz'*fz)/n;          %also Z'*(I-P)*Z/n
    JJ = (U'*Z-fu'*fz)/mS;         %also U'*(I-P)*Z/mS

    g = -JJ + slogy;               %First derivative of loglik function
    if i == 1
      %Second Derivative (or the Hessian) of the loglik function
      %Part in parentheses easier to write as V`(I-P)Z-U`(I-P)*U
      H = ((V'*Z-V'*F*fz) - (U'*U-fu'*fu))/mS + 2/n*JJ.^2;
      step = -g/H;  %Step for mle
    else
      step = -(const-n/2*log(mS)+lam*slogy)/g; %Step for CI estimates
    end;
    lam = lam + step;              %Update Estimate
  end;

  %First loop saves mlelam, second loop, the lci, and third, the uci
  switch i
    case 1, mlelam = lam;
    case 2, lci = lam;
    case 3, uci = lam;
  end;

  lglik = -n/2*(log(2*pi) + 1 + log(mS)) + (lam-1)*slogy;
  %Save mles of beta, sigma^2 and transformed data
  %Also, save loglikelihood for later use
  if i == 1
    Bmle = G*G'*(X'*Z);           %Estimate of parameters
    mSmle = mS;
```

```
    Zmle = Z;
    loglik = lglik;
  else
    ciloglik = lglik;
  end;
end;
%------------------------------------------------------------------
% Graph the Loglikelihood along with the (ConLevel*100)% confidence interval
spread = (uci-lci)/6;
maxci = max(abs(uci),abs(lci));
x=(-(maxci+3*spread)):spread/2:(maxci+spread);
f=[];
[dum,p] = size(x);

% Get the data ready for plotting.
for i=1:p
  if abs(x(i)) < small
    Z = logy;
  else
    Z = (y.^x(i)-1)/x(i);
  end;
  ms = (Z'*Z-Z'*F*F'*Z)/n;
  lglik = -n/2*(log(2*pi) + 1 + log(ms)) + (x(i)-1)*slogy;
  f = [ f , lglik ];
end;
plot(x,f);                                          %plot the data
ax = axis;                                          %axis parameters
lciline = [ lci ax(3) ; lci ciloglik ];
uciline = [ uci ax(3) ; uci ciloglik ];
mleline = [ mlelam ax(3) ; mlelam loglik ];
line(lciline(:,1),lciline(:,2));                    %plot lower ci line
line(uciline(:,1),uciline(:,2));                    %plot upper ci line
line(mleline(:,1),mleline(:,2));                    %plot mle line
middle = (ax(3)+ax(4))/2;
ysprd = (ax(4)-ax(3))/60;
text(lci+spread/4,middle+10*ysprd,num2str(lci));    %print lci on graph
text(uci+spread/4,middle-10*ysprd,num2str(uci));    %print uci on graph
text(mlelam+spread/4,loglik+ysprd,num2str(mlelam)); %print mle on graph
xlabel('\lambda');
ylabel('Log likelihood');
endgraph='CI for \lambda for the Plasma data'
[titlegraph, errmg] = sprintf('Approximate %2.5g%%',ConLevel*100);
titlegraph = strcat(titlegraph,endgraph);
title(titlegraph);
```

A.3 Finding the limit of u_i and v_i

When $\lambda = 0$, both the expression for u_i and v_i are in the indeterminate form of $\left(\frac{0}{0}\right)$. Yet, when programming, the limit of u_i and v_i as $\lambda \to 0$ must be found to avoid problems with roundoff error. Hence the additions to equations (5) and (13).

L'Hopital's rule can be used to find these limits. First, $\lim\limits_{\lambda \to 0} u_i$ yields

$$\begin{aligned}
\lim_{\lambda \to 0} u_i &= \lim_{\lambda \to 0} \frac{(\lambda \ln y_i - 1) y_i^{\lambda} + 1}{\lambda^2} \\
&= \lim_{\lambda \to 0} \frac{(\lambda \ln y_i - 1) y_i^{\lambda} \ln y_i + y_i^{\lambda} \ln y_i}{2\lambda} \\
&= \lim_{\lambda \to 0} \frac{\lambda y_i^{\lambda} (\ln y_i)^2}{2\lambda} \\
&= \lim_{\lambda \to 0} \frac{y_i^{\lambda} (\ln y_i)^2}{2} \\
&= \frac{(\ln y_i)^2}{2}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\lim_{\lambda \to 0} v_i &= \lim_{\lambda \to 0} \frac{2 - \left[2\lambda \ln y_i - (\lambda \ln y_i)^2 - 2\right] y_i^{\lambda}}{\lambda^3} \\
&= -\lim_{\lambda \to 0} \frac{\left[2\lambda \ln y_i - (\lambda \ln y_i)^2 - 2\right] y_i^{\lambda} \ln y_i + \left[2 \ln y_i + 2\lambda (\ln y_i)^2\right] y_i^{\lambda}}{3\lambda^2} \\
&= -\lim_{\lambda \to 0} \frac{y_i^{\lambda} \lambda^2 (\ln y_i)^3}{3\lambda^2} \\
&= -\lim_{\lambda \to 0} \frac{y_i^{\lambda} (\ln y_i)^3}{3} \\
&= -\frac{(\ln y_i)^3}{3}.
\end{aligned}$$

A.4 Finding the form for the observed Fisher's Total Information matrix.

To construct the observed Fisher's Total Information matrix,$\widehat{\boldsymbol{J}}(\boldsymbol{\theta})$, the second partial derivatives of $\ell(\boldsymbol{\theta} \mid \boldsymbol{y})$ are needed. Recall

$$\begin{aligned}
\ell(\boldsymbol{\theta} \mid \boldsymbol{y}) &= -\frac{1}{2\sigma^2}(\boldsymbol{z}-\boldsymbol{X\beta})'(\boldsymbol{z}-\boldsymbol{X\beta}) - \frac{n}{2}\ln(2\pi\sigma^2) + (\lambda-1)\sum_{i=1}^{n}\ln y_i \\
&= -\frac{1}{2\sigma^2}(\boldsymbol{z}'\boldsymbol{z}-\boldsymbol{z}'\boldsymbol{X\beta}-\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{z}-\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{X\beta}) - \frac{n}{2}\ln(2\pi\sigma^2) + (\lambda-1)\sum_{i=1}^{n}\ln y_i.
\end{aligned}$$

Taking the first partial derivatives yields

$$\begin{aligned}
\frac{\partial \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}} &= -\frac{1}{2\sigma^2}(-\boldsymbol{X}'\boldsymbol{z}-\boldsymbol{X}'\boldsymbol{z}+2\boldsymbol{X}'\boldsymbol{X\beta}) \\
&= -\frac{1}{\sigma^2}(-\boldsymbol{X}'\boldsymbol{z}+\boldsymbol{X}'\boldsymbol{X\beta}), \\
\frac{\partial \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \sigma^2} &= \frac{1}{2\sigma^4}(\boldsymbol{z}-\boldsymbol{X\beta})'(\boldsymbol{z}-\boldsymbol{X\beta}) - \frac{n}{2\sigma^2},
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \lambda} &= -\frac{1}{2\sigma^2}\frac{\partial(\boldsymbol{z}-\boldsymbol{X\beta})'(\boldsymbol{z}-\boldsymbol{X\beta})}{\partial \lambda} + \sum_{i=1}^{n}\ln y_i \\
&= -\frac{1}{2\sigma^2}\frac{\partial \boldsymbol{z}'}{\partial \lambda}\frac{\partial\left[(\boldsymbol{z}'\boldsymbol{z}-\boldsymbol{z}'\boldsymbol{X\beta}-\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{z}-\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{X\beta})\right]}{\partial \boldsymbol{z}} + \sum_{i=1}^{n}\ln y_i \\
&= -\frac{1}{2\sigma^2}\boldsymbol{u}'(2\boldsymbol{z}-2\boldsymbol{X\beta}) + \sum_{i=1}^{n}\ln y_i \\
&= -\frac{1}{\sigma^2}\boldsymbol{u}'(\boldsymbol{z}-\boldsymbol{X\beta}) + \sum_{i=1}^{n}\ln y_i.
\end{aligned}$$

Second partial derivatives are

$$\begin{aligned}
\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \boldsymbol{\beta}'\,\partial \boldsymbol{\beta}} &= -\frac{\boldsymbol{X}'\boldsymbol{X}}{\sigma^2}, \\
\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial \sigma^2\,\partial \boldsymbol{\beta}} &= \frac{1}{\sigma^4}(-\boldsymbol{X}\boldsymbol{z}+\boldsymbol{X}'\boldsymbol{X\beta}) \\
&= -\frac{\boldsymbol{X}'(\boldsymbol{z}-\boldsymbol{X\beta})}{\sigma^4},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda\,\partial\boldsymbol{\beta}} &= \frac{1}{\sigma^2}\frac{\partial(\boldsymbol{X}\boldsymbol{z})}{\partial\lambda} \\
&= \frac{1}{\sigma^2}\frac{\partial z'}{\partial\lambda}\frac{\partial(\boldsymbol{X}\boldsymbol{z})}{\partial\boldsymbol{z}} \\
&= \frac{\boldsymbol{u}'\boldsymbol{X}}{\sigma^2} = \frac{\boldsymbol{X}'\boldsymbol{u}}{\sigma^2},
\end{aligned}$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial(\sigma^2)^2} = -\frac{1}{\sigma^6}(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta}) + \frac{n}{2\sigma^4},$$

$$\begin{aligned}
\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda\,\partial\sigma^2} &= \frac{1}{2\sigma^4}\frac{\partial(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})}{\partial\lambda} \\
&= \frac{1}{2\sigma^4}\left[2\boldsymbol{u}'(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})\right] \\
&= \frac{\boldsymbol{u}'(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})}{\sigma^4},
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda^2} &= \frac{1}{\sigma^2}\frac{\overbrace{\partial\left[-\boldsymbol{u}'(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})\right]}^{\text{product rule}}}{\partial\lambda} \\
&= \frac{1}{\sigma^2}\left[-\frac{\partial\boldsymbol{u}'}{\partial\lambda}(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta}) - \boldsymbol{u}'\frac{\partial(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta})}{\partial\lambda}\right] \\
&= \frac{\boldsymbol{v}'(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta}) - \boldsymbol{u}'\boldsymbol{u}}{\hat{\sigma}^2}.
\end{aligned}$$

Now, recall,

$$-\widehat{\boldsymbol{J}}(\boldsymbol{\theta}) = \begin{pmatrix}
\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\boldsymbol{\beta}'\,\partial\boldsymbol{\beta}} & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\sigma^2\,\partial\boldsymbol{\beta}} & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda\,\partial\boldsymbol{\beta}} \\
\left(\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\sigma^2\,\partial\boldsymbol{\beta}}\right)' & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial(\sigma^2)^2} & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda\,\partial\sigma^2} \\
\left(\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda\,\partial\boldsymbol{\beta}}\right)' & \left(\dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda\,\partial\sigma^2}\right)' & \dfrac{\partial^2 \ell(\boldsymbol{\theta} \mid \boldsymbol{y})}{\partial\lambda^2}
\end{pmatrix}.$$

So substituting in the second partials results in

$$= \frac{1}{\sigma^2}\begin{pmatrix} \boldsymbol{X}'\boldsymbol{X} & \dfrac{\boldsymbol{X}'(\boldsymbol{z}-\boldsymbol{X\beta})}{\sigma^2} & -\boldsymbol{X}'\boldsymbol{u} \\ \left(\dfrac{\boldsymbol{X}'(\boldsymbol{z}-\boldsymbol{X\beta})}{\sigma^2}\right)' & \dfrac{(\boldsymbol{z}-\boldsymbol{X\beta})'(\boldsymbol{z}-\boldsymbol{X\beta})}{\sigma^4}-\dfrac{n}{\sigma^2} & -\dfrac{\boldsymbol{u}'(\boldsymbol{z}-\boldsymbol{X\beta})}{\sigma^2} \\ (-\boldsymbol{X}'\boldsymbol{u})' & \left(-\dfrac{\boldsymbol{u}'(\boldsymbol{z}-\boldsymbol{X\beta})}{\sigma^2}\right)' & \boldsymbol{u}'\boldsymbol{u}-\boldsymbol{v}'(\boldsymbol{z}-\boldsymbol{X\beta}) \end{pmatrix},$$

which is equivalent to equation (17).

References

[1] Batchelder, A. R. , Rogers, M. J. , & Walker, J. P. (1966). Effects of Subsoil Management Practices on Growth of Flue-Cured Tobacco. *Agronomy Journal*, **58** 345–347. 15

[2] Boik, R. J. & Marasinghe, M. G. (1989). Analysis of Nonadditive Multiway Classifications. *Journal of the American Statistical Association*, **84** 1059–1064. 15, 16

[3] Box, G. E. P. & Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society*, **2** 211–252. 1, 2, 3, 14

[4] Carroll, R. J. (1980). A Robust Method for Testing Transformations to Achieve Approximate Normality. *Journal of the Royal Statistical Society* **42** 71–78. 3

[5] Hoyle, M. H. (1973). Transformations–An Introduction and a Bibliography. *Statistical Review*, **41** 203–223. 2

[6] Kennedy, W. J. & Gentle, J. E. (1980). *Statistical Computing*. New York: Marcel Dekker. 6, 12

[7] Lindsey, J. K. (1996). *Parametric Statistical Inference*. New York: Oxford University Press. 9, 12

[8] Neter, J., Wasserman, W., & Kutner, M. H. (1990). *Applied Linear Statistical Models, Third Edition*. Boston: R. R. Donnelly Sons & Company. 13

[9] Sakia, R. M. (1992). The Box-Cox Transformation Technique: A Review. *The Statistician*, **41** 169–178. 2

[10] Schott, J. R. (1997). *Matrix Analysis For Statistics*. New York: John Wiley & Sons. 5, 6

[11] Sen, A. K. & Srivastava, M. S. (1990). *Regression Analysis: Theory, Methods, and Applications*. New York: Springer-Verlag. 2