Stat 5421 Lecture Notes

Exponential Families, Part II

Charles J. Geyer

May 4, 2016

Contents

1 Existence of Maximum Likelihood Estimates

1.1 A Fact about Exponential Families

Section 1.5 of Part I of these lecture notes (Geyer, 2016, hereinafter referred to as "Part I of these notes") stresses a property of exponential

families: all distributions in the family must have the same support. Then near the end of Section 2.1 of Part I of these notes it is explictly stated that the mean value parameter μ of the binomial family of distributions, considered as an exponential family, has range $0 < \mu < n$. It then follows from the "observed equals expected property" of regular full exponential families (Theorem 3 of Part I of these notes and the discussion following following it) that the maximum likelihood estimate (MLE) for the mean value parameter is $\hat{\mu} = y$, where y is the canonical statistic of the binomial distribution (the observed number of successes in n Bernoulli trials).

It follows from these facts that when $y = 0$ or $y = n$ (we observe all successes or all failures) the MLE does not exist (this is explicitly stated by the aforementioned Theorem 3). Part II of these notes (this document) is about this issue: What do we do when the MLE for a regular full exponential family does not exist?

The binomial example already shows this is a phenomenon that cannot just be ignored. Most statistics books do ignore it, although not Agresti (2013), which has Section 6.5 devoted to this issue, which it calls, in the context of logistic regression, *complete separation* and *quasi-complete separation*. Agresti does not deal with this issue in the context general exponential families or in the limited contexts of Poisson regression or log-linear models for contingency tables. Moreover, Agresti does not provide any useful advice about what to do when the problem of MLE nonexistence arises.

1.2 History

The problem of nonexistence of the MLE in regular full exponential families has long been completely understood. The theory is found in the books of Barndorff-Nielsen (1978, pp. 154–158 and 163–164) and Brown (1986, pp. 191–202). Computing for this theory has also long been completely understood. An algorithm is found in your humble author's unpublished PhD thesis (Geyer, 1990). A modified version of this algorithm that makes use of the R package `rcdd` (Geyer, Meeden, and Fukuda, 2016) was published as Geyer (2009b).

The theory in Barndorff-Nielsen (1978) covers full exponential familes with finite support (which are automatically regular), which includes logistic regression and log-linear models for contingency tables if multinomial or product multinomial sampling is assumed. The theory in Brown (1986) covers some regular full exponential families with infinite support, including Poisson regression with log link and log-linear models for contingency tables (as well as including the results of Barndorff-Nielsen). The theory in

Chapter 4 of Geyer (1990) covers any exponential family whatsoever (full, non-full, regular or non-regular) but is not needed for models for discrete multivariate data covered in this course.

The algorithm in Chapter 2 of Geyer (1990) covers some non-full exponential families, but the algorithm in Geyer (2009b) only covers regular full exponential families satisfying the conditions of Brown (1986). But that covers everything we want in this course.

1.3 The Binomial Family Again

One can define binomial distributions with success probability equal to zero or one. Look at the probability mass function (PMF)

$$f_p(y) = \binom{n}{y} p^y (1-p)^{n-y}, \qquad y = 0, \ldots, n.$$

In case $y = 0$, this becomes

$$f_p(0) = (1-p)^n \tag{1}$$

and if we take the limit as $p \to 0$ we obtain

$$f_0(0) = 1$$

which implies that the distribution is degenerate

$$f_0(y) = \begin{cases} 1, & y = 0 \\ 0, & y \neq 0 \end{cases} \tag{2}$$

A similar limit process applied to the case $y = n$ gives

$$f_1(y) = \begin{cases} 0, & y \neq n \\ 1, & y = n \end{cases} \tag{3}$$

If we include these distributions in the statistical model, then maximum likelihood estimates do exist. And we preserve the "observed equals expected" property $\hat{\mu} = y$. The MLE distribution when we observe $y = 0$ is the distribution concentrated at zero, which has PMF (2). The MLE distribution when we observe $y = n$ is the distribution concentrated at zero, which has PMF (2).

We can see these assertions by looking at the likelihood, which is (1) in the case $y = 0$ and which is a strictly decreasing function of p. This means the likelihood is maximized at the lower end of the range of values $0 \leq p \leq 1$. The proof for the $y = n$ case is similar. We won't bother with the details.

1.4 Directions of Recession and Constancy

It turns out that this binomial example perfectly illustrates the theoretical issues that arise for arbitrary full exponential families. The only additional issue is that the geometry becomes impossible to visualize when the dimension of the model is greater than two.

MLE fail to exist when the data are at one end of the range of possible values in some direction. If y denotes the observed value of the canonical statistic and Y a random value, then we say y is extreme if there exists a vector δ in the parameter space such that $\langle Y, \delta \rangle \le \langle y, \delta \rangle$ almost surely. "Almost surely" means with probability one, and in an exponential family this does not depend on the parameter value because all distributions in the family have the same support.

Geyer (2009b, Theorem 3 and the following discussion) introduces into the exponential family literature the term *direction of recession* for a vector δ having the property discussed in the preceding paragraph. The term comes from the theory of convex sets and functions (Rockafellar, 1970, p. 69).

Recall that a direction δ in the parameter space is called a *direction of constancy* if $\langle Y, \delta \rangle$ is almost surely constant (Theorem 1 and the surrounding discussion in Part I of these notes). In this case we must have $\langle y, \delta \rangle = \langle Y, \delta \rangle$ almost surely. Thus, for comparison, we have

term	definition
direction of recession	$\langle Y, \delta \rangle \le \langle y, \delta \rangle$ almost surely
direction of constancy	$\langle Y, \delta \rangle = \langle y, \delta \rangle$ almost surely

It is clear from the alternative definition $\langle Y, \delta \rangle$ is constant almost surely that whether δ is a direction of constancy does not depend on the observed data y. As we shall see, whether δ is a direction of recession does depend on y.

1.5 Existence of MLE

Theorem 1. *The MLE in a full exponential family exists if and only if every direction of recession is a direction of constancy.*

This is Theorem 4 in Geyer (2009b). It tells us exactly when MLE exist in a full exponential family.

We can see why maximum likelihood don't exist when there is a direction of recession that is not a direction of constancy by looking at the derivative

of the function $s \mapsto l(\theta + s\delta)$ where l is the log likelihood

$$\begin{aligned} \frac{dl(\theta + s\delta)}{ds} &= \frac{d}{ds}\big[\langle y, \theta + s\delta\rangle - c(\theta + s\delta) \\ &= \langle y, \delta\rangle - \langle \mu(\theta + s\delta), \delta\rangle \end{aligned}$$

where

$$\mu(\theta) = \nabla c(\theta) = E_\theta(Y)$$

by equation (5) in Part I of these notes; which is the mean value parameter expressed as a function of the canonical parameter. Hence

$$\begin{aligned} \frac{dl(\theta + s\delta)}{ds} &= \langle y, \delta\rangle - \langle E_{\theta+s\delta}(Y), \delta\rangle \\ &= E_{\theta+s\delta}(\langle y - Y, \delta\rangle) \end{aligned}$$

(the bilinear form and the subtraction can be moved inside the expectation by linearity). We know that if δ is a direction of recession that is not a direction of constancy, then $\langle y - Y, \delta\rangle$ is nonnegative almost surely and is not zero almost surely. Thus its expectation is strictly positive. Hence we have deduced that, if δ is a direction of recession that is not a direction of constancy, then $l(\theta + s\delta)$ is a strictly increasing function of s; the likelihood keeps increasing all the way to infinity. No point θ can be the MLE because $\theta + s\delta$ has a higher likelihood whenever $s > 0$.

It is not obvious that $\theta + s\delta$ remains in the full canonical parameter space of the family as $s \to \infty$. But we can see this is so by (4) in Part I of these notes, which is

$$c(\theta) = c(\psi) + \log E_\psi\left\{e^{\langle Y, \theta - \psi\rangle}\right\} \tag{4}$$

and comes from Geyer (2009b, equation (5) there). We have (still assuming δ is a direction of recession)

$$\begin{aligned} c(\theta + s\delta) &= c(\psi) + \log E_\psi\left\{e^{\langle Y, \theta + s\delta - \psi\rangle}\right\} \\ &= c(\psi) + \log E_\psi\left\{e^{\langle Y, \theta - \psi\rangle} e^{s\langle Y, \delta\rangle}\right\} \\ &\le c(\psi) + \log E_\psi\left\{e^{\langle Y, \theta - \psi\rangle} e^{s\langle y, \delta\rangle}\right\} \\ &= c(\psi) + s\langle y, \delta\rangle + \log E_\psi\left\{e^{\langle Y, \theta - \psi\rangle}\right\} \\ &= c(\theta) + s\langle y, \delta\rangle \end{aligned}$$

And this shows that $c(\theta) < \infty$ and δ a direction of recession proves $c(\theta + s\delta) < \infty$ for all $s > 0$.

2 MLE as Limits

2.1 Limits in Directions of Recession

What when MLE don't exist? In the binomial family we saw that we needed to take limits. That works in general, at least for exponential families for discrete data.

Theorem 2. *For a full exponential family for discrete data having PMF given by*

$$f_\theta(\omega) = e^{\langle Y(\omega), \theta \rangle - c(\theta) + h(\omega)}, \qquad \omega \in \Omega, \tag{5}$$

if δ is a direction of recession for observed value y of the canonical statistic Y which takes values in $\mathbb{R}^p$ and

$$H_\delta = \{\, w \in \mathbb{R}^p : \langle w - y, \delta \rangle = 0 \,\}, \tag{6}$$

then

$$f_{\theta + s\delta}(\omega) \to f_\theta(\omega \mid Y \in H_\delta) \qquad \text{as } s \to \infty. \tag{7}$$

This is Theorem 6 in Geyer (2009b). In equation (5) ω is the whole data which may or may not be the canonical statistic Y and Ω is the sample space (the set of all possible values of ω). The conditional distribution that is the limiting distribution in (7) has PMF

$$f_\theta(\omega \mid Y \in H_\delta) = \begin{cases} f_\theta(\omega) / \operatorname{pr}_\theta(Y \in H_\delta), & Y(\omega) \in H_\delta \\ 0, & \text{otherwise} \end{cases}$$

It is important to remember that the limiting distribution in the theorem can be thought of in two different ways: it can be derived by taking a limit or by conditioning. No matter which one we choose to emphasize by notation or terminology, it is also the other one. It is both a limit distribution and a conditional distribution.

If Θ is the full canonical parameter space of the original model (before limits or conditioning), then the statistical model having densities

$$f_\theta(\,\cdot \mid Y \in H_\delta), \qquad \theta \in \Theta,$$

is an exponential family, because the log likelihood is

$$l_\delta(\theta) = \langle y, \theta \rangle - c(\theta) - \log \operatorname{pr}_\theta(Y \in H_\delta)$$

so we see we have a new exponential family with the same canonical statistic y and the same canonical parameter θ but a new cumulant function

$$c_\delta(\theta) = c(\theta) + \log \operatorname{pr}_\theta(Y \in H_\delta).$$

Geyer (2009b) calls this exponential family the *limiting conditional model* (LCM). For contrast, call the original model (before limits or conditioning) the OM.

It is implicit that we have taken limits in a direction of recession δ. If we wanted to be more specific, we could write LCM_δ to indicate that the LCM can depend on the direction in which we take limits. But, as we shall soon see, we usually do not need to consider more than one δ or more than one LCM.

Now suppose δ is a direction of recession that is not a direction of constancy (for the OM). This means that distributions in the original model are not concentrated on H_δ. Hence $\text{pr}_\theta(Y \in H_\delta) < 1$ and $\log \text{pr}_\theta(Y \in H_\delta) < 0$. So, if we compare log likelihoods l for the original model and l_δ for the LCM, we see that

$$l(\theta) < l_\delta(\theta), \qquad \theta \in \Theta,$$

and the maximum of the log likelihood is always at least as large in the LCM as in the OM. Hence the MLE for the LCM is always the MLE for the model that is the union of the OM and LCM. Hence, if the MLE exists for the LCM, then we have found the MLE as a limit of distributions in the OM.

And if the MLE does not exist for the LCM, then it has a direction of recession that is not a direction of constancy, and we can repeat the process. Eventually we must get an MLE in an LCM. And we find the MLE as a limit of limits or limit of limits of limits, etc. of distributions in the OM.

2.2 Generic Directions of Recession

Geyer (2009b) not only introduces the concepts of directions of recession and constancy to exponential family theory but also the concept of a *generic direction of recession* (GDOR). This is a direction of recession that is in the relative interior of the set of all directions of recession (Geyer, 2009b, Section 3.6), but we need not understand that definition because we are going to use Theorems in Geyer (2009b) to characterize it that do not obviously refer to the definition. Under a regularity condition of Brown (1986) that always holds for logistic regression, Poisson regression, and log-linear models for contingency tables there is always a GDOR δ whenever the MLE for the OM does not exist, and if we take limits in this direction δ the MLE for the LCM is guaranteed to exist. So we never have to take limits of limits if the first direction is a GDOR.

2.3 Example I

Consider a binomial family. This is a one-dimensional exponential family. Suppose the observed data are $y = 0$. Then there is a direction of recession that it not a direction of constancy of the original family. A direction vector in one-dimensional space is just a number. If $\delta < 0$, then $Y\delta \le y\delta = 0$ for all possible values Y. If we take limits $\theta + s\delta$ as $s \to \infty$, this is the same as taking limits as $\theta \to -\infty$. Since the map from canonical to mean value parameter is (for the binomial family)

$$\mu = \frac{n}{1 + \exp(-\theta)}$$

(this follows from (10) in Part I of these notes), so $\mu \to 0$ as $\theta \to -\infty$. And the LCM in the direction δ contains only one distribution the distribution concentrated at $y = 0$, which is (by our previous analysis) the MLE.

Similarly, if we observe $y = n$, then any $\delta > 0$ is a direction of recession that is not a direction of constancy, and, if we take limits in this direction, then the LCM in this direction δ contains only one distribution the distribution concentrated at $y = n$, which is the MLE for these data.

2.4 Canonical Affine Submodels

Section 4.6 of these notes introduces the concept of canonical affine submodels of an exponential family statistical model. These too are exponential families. They have canonical parameter β related to the canonical parameter of the original family (called the saturated model in this context) by

$$\theta = a + M\beta$$

where a is a known vector (the *offset vector*) and M is a known matrix (the *model matrix*), where "known" means perhaps a function of predictor variables but not a function of the response variable.

A canonical affine submodel is itself a regular full exponential family with canonical parameter β and canonical statistic $M^T y$, where y is the saturated model canonical statistic (the response vector).

Thus all of our theory holds for canonical affine submodels. We just need to replace y with $M^T y$ and Y with $M^T Y$ wherever they occur.

2.5 Example II

Agresti (2013, Section 6.5.1) introduces the notation of complete separation with the following example

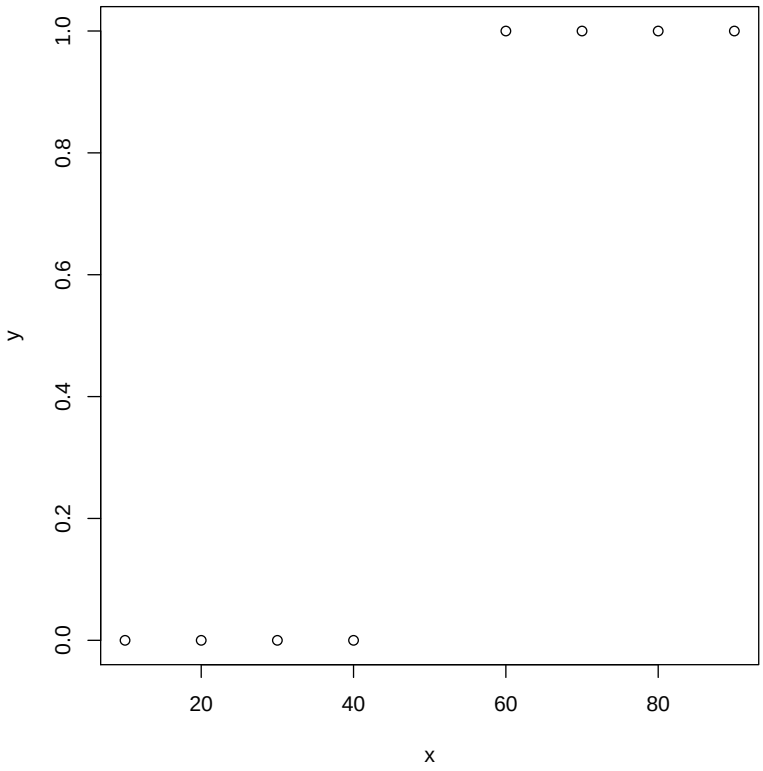

Figure 1: Some Logistic Regression Data.

```
> x <- seq(10, 90, 10)
> x <- x[x != 50]
> x

[1] 10 20 30 40 60 70 80 90

> y <- as.numeric(x > 50)
> y

[1] 0 0 0 0 1 1 1 1
```

Figure 1 shows these data.

Suppose we want to do "simple" logistic regression (one predictor x plus intercept, so the model is two-dimensional). Our theory tells us that we must look at the set of all possible values of the canonical statistic $M^T y$

where M is the model matrix for this model, which has two columns: the first column is all ones (the "intercept" column) and the second column is x. So let's find that set. There are 2^n possible values where n is the dimension of the response vector because each component of y can be either zero or one. The following code makes all of those vectors.

```
> yy <- NULL
> n <- length(y)
> for (i in 1:n) {
+     j <- 2^(i - 1)
+     k <- 2^n / j / 2
+     yy <- cbind(rep(rep(0:1, each = j), times = k), yy)
+ }

> head(yy)

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    0    0    0    0    0    0    0
[2,]    0    0    0    0    0    0    0    1
[3,]    0    0    0    0    0    0    1    0
[4,]    0    0    0    0    0    0    1    1
[5,]    0    0    0    0    0    1    0    0
[6,]    0    0    0    0    0    1    0    1

> dim(yy)

[1] 256   8
```

For those who know how to count in binary, row i is $i-1$ expressed in binary. Have you heard the joke: there are two kinds of people in this world, those who divide everything into two kinds and those who don't? And its nerd version: there are 10 kinds of people in this world, those who know binary and those who don't?

For those who don't, the following code shows that every row of `yy` is different, every row contains only zeros and ones, and there are 2^n rows.

```
> fred <- apply(yy, 1, paste, collapse = "")
> head(fred)

[1] "00000000" "00000001" "00000010" "00000011" "00000100"
[6] "00000101"
```

```
> length(unique(fred)) == length(fred)

[1] TRUE

> all(apply(yy, 1, function(x) all(x %in% 0:1)))

[1] TRUE

> nrow(yy) == 2^n

[1] TRUE
```

But there are not so many distinct values of the submodel canonical statistic.

```
> m <- cbind(1, x)
> mtyy <- t(m) %*% t(yy)
> t1 <- mtyy[1, ]
> t2 <- mtyy[2, ]
> t1.obs <- sum(y)
> t2.obs <- sum(x * y)
```

Figure 2 shows these possible values of the submodel canonical statistic.

And now we are stuck. Figure 2 seems to show that the observed data vector is an extreme value, but we cannot easily figure out the direction of recession.

2.6 Tangent Vectors

Vectors $Y(\omega) - y$, where y is the observed value of the canonical statistic vector and $Y(\omega)$ are other possible values of the canonical statistic vector, are called *tangent vectors* (Geyer, 2009b, explains the reason they have this name). If

$$V = \{ v_i : i \in I \}$$

is the set of tangent vectors, then the set of a nonnegative combinations of them, vectors of the form

$$\sum_{i \in A} a_i v_i$$

where A is a finite set and $a_i \geq 0$ for all i, is called the *tangent cone*. It is denoted $\operatorname{con}(\operatorname{pos} T)$ in Geyer (2009b).

Figure 3 shows the tangent vectors and tangent cone. The points in Figures 2 and 3 are the same except in Figure 3 they are moved so the one

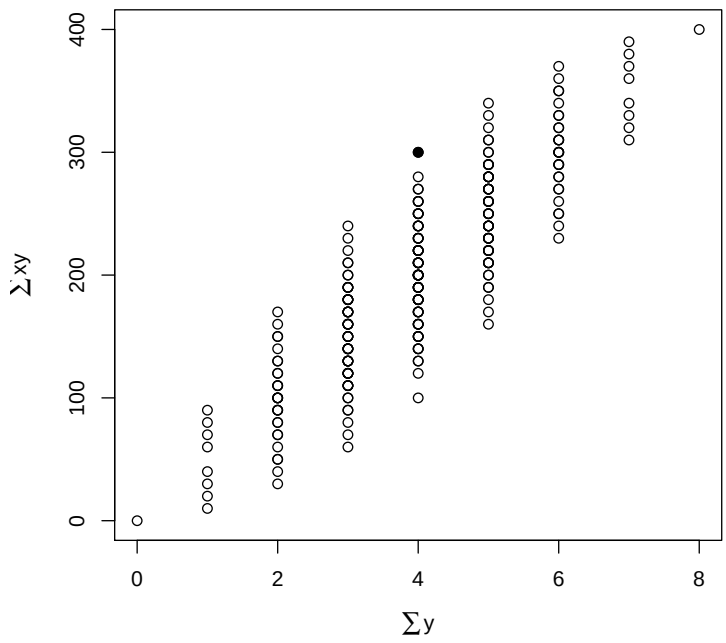

Figure 2: Possible values of the submodel canonical statistic vector $M^T y$ for the data shown in Figure 1. Solid dot is the observed value of the submodel canonical statistic vector.

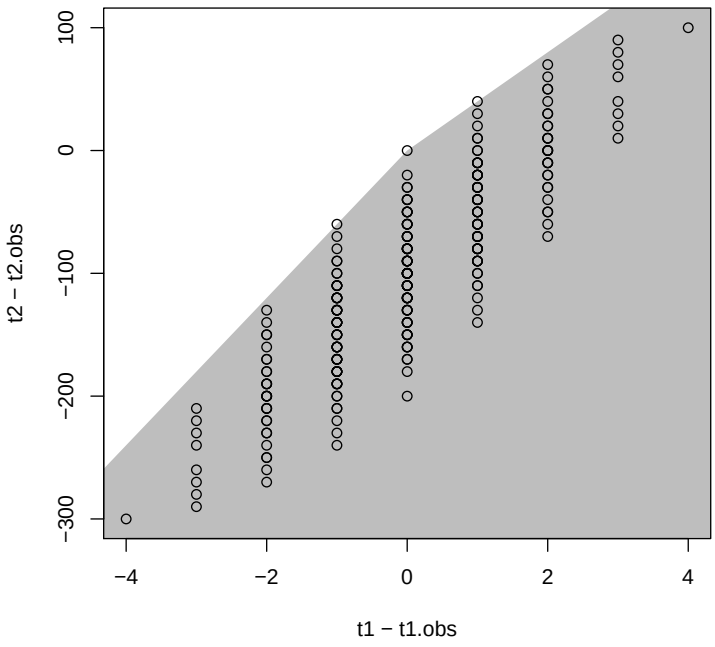

Figure 3: Tangent vectors and tangent cone for data shown in Figure 1. Dots are tangent vectors, gray region is tangent cone.

corresponding to the observed value of the canonical statistic is the origin $(0, 0)$. The gray area is the tangent cone (set of all nonnegative combinations of tangent vectors).

We are interested in the case where a finite subset of the tangent vectors gives the same tangent cone, that is, when S is a finite subset of T such that $\operatorname{con}(\operatorname{pos} S) = \operatorname{con}(\operatorname{pos} T)$. This is obviously the case, when the statistical model has finite support so T is finite, as in logistic regression and log-linear models for contingency tables with multinomial or product multinomial sampling. As we shall see, it is also the case for Poisson regression with log link and for log-linear models for contingency tables with Poisson sampling.

For generalized linear models (GLM) we do not need all the tangent vectors. For the saturated model, tangent vectors $Y(\omega) - y$ such that $Y(\omega)$ and y differ only in one coordinate are enough to generate the whole tangent cone (Geyer, 2009b, Section 3.11). Moreover, if V_{sat} is a set of vectors generating the tangent cone for the saturated model, then

$$V_{\text{sub}} = \{\, M^T v : v \in V_{\text{sat}} \,\}$$

is a set of vectors generating the tangent cone for the canonical affine submodel with model matrix M (Geyer, 2009b, Section 3.10).

So now we need to learn how to find these tangent vectors for the saturated model.

First consider logistic regression when y has Bernoulli components (zero-or-one-valued). If the observed value of y_i is 0, the only other possible value is 1, so the vector e_i, which has all coordinates equal to 0 except the i-th component, which is 1, is a tangent vector. Similar reasoning says $-e_i$ is a tangent vector if $y_i = 1$.

Second consider logistic regression when y has binomial components. Now we have not only components y_i of the response vector but sample sizes n_i that go with them. We see that if $y_i = 0$, then e_i is a tangent vector (as before), and if $y_i = n_i$, then $-e_i$ is a tangent vector (as before), but now we also have the case $0 < y_i < n_i$ in which case it is possible to change the i-th coordinate either up or down, so both e_i and $-e_i$ are tangent vectors.

Third consider Poisson regression with log link. Just like in the binomial case we have e_i is a tangent vector when $y_i = 0$, and both e_i and $-e_i$ are tangent vectors when $0 < y_i < \infty$. Since there is no upper bound to the range of a Poisson random variable, there is no case where only $-e_i$ is a tangent vector.

2.7 Calculating the Linearity

We now want to calculate a GDOR, but that calculation proceeds in two steps in the algorithm of Geyer (2009b). First we need to find the *linearity* of the tangent cone (Geyer, 2009b, Section 3.12), which is the smallest vector subspace contained in the tangent cone, although Geyer (2009b) also (somewhat sloppily) uses the same term for a set of vectors spanning this vector subspace.

If V_{sub} is a set of vectors generating the tangent cone for the canonical affine submodel, then there is an R function `linearity` in the R package `rcdd` that calculates

$$L_{\text{sub}} = \{\, v \in V_{\text{sub}} : -v \in \text{con}(\text{pos}\, V_{\text{sub}}) \,\}. \tag{8}$$

This is the set of all the given tangent vectors that are in the linearity of the tangent cone. They also span it, hence determine it.

If we use L_{sub} as defined in (8) to denote a set of tangent vectors. Then the linearity considered as a vector space is denoted $\text{span}\, L_{\text{sub}}$.

The linearity is useful for three reasons. We have $Y \in H_\delta$ if and only if $Y - y \in \text{span}\, L_{\text{sub}}$. So the linearity tells us the support of the LCM. We also need to know what the linearity is in order to calculate a GDOR. Finally, the linearity tells whether the MLE exists or not. It exists if and only if $L_{\text{sub}} \neq V_{\text{sub}}$ (Geyer, 2009b, Theorem 4).

So let us calculate the linearity for our example, the data shown in Figure 1. We follow Section 4.1 of Geyer (2008).

```
> library(rcdd)
> tanv <- m
> tanv[y == 1, ] <- (-tanv[y == 1, ])
> vrep <- makeV(rays = tanv)
> lout <- linearity(d2q(vrep), rep = "V")
> lout

integer(0)
```

$M^T e_i$ is just the i-th row of M, so the rows of `m` are either tangent vectors or -1 times tangent vectors. So we assign `tanv` to be `m` and then adjust the signs. For rows of `m` such that corresponding component of `y` is equal to one, we need to change the sign. So the second and third lines of the code chunk above make `tanv` a matrix whose rows are the elements of $V_{\texttt{sub}}$. Then next two lines are idiomatic usage of the R package `rcdd`. The result

`lout` is an integer vector giving the indices of the tangent vectors in the linearity, that is, `tanv[linearity, ]` is a basis for the linearity considered as a vector subspace.

Here the result is a vector of length zero, which says the empty set of vectors spans the linearity, which means it is the trivial vector subspace $\{0\}$ that has only one point. We could actually see this in Figure 3, the gray area is a pointed cone, so it contains only the trivial subspace.

So this tells us that the support of the LCM for this example contains only one point. The MLE distribution is completely degenerate, concentrated at y. The MLE distribution says the only data we could have observed is what we did observe; no other data values were possible. Before anyone decides this is weird, let me remind you this is only an estimate, and, as always, estimates are not parameters. This degeneracy causes no problem so long as we don't overinterpret it.

This complete degeneracy of the MLE distribution is what Agresti calls "complete separation."

2.8 Calculating Generic Directions of Recession

If $L_{\text{sub}} \neq V_{\text{sub}}$ the MLE does not exist in the OM, and we need to calculate a GDOR. In this we follow Geyer (2009b, Section 3.13). A vector δ in the parameter space is a GDOR if and only if

$$\langle v, \delta \rangle = 0, \qquad v \in L_{\text{sub}} \tag{9a}$$

$$\langle v, \delta \rangle < 0, \qquad v \in V_{\text{sub}} \setminus L_{\text{sub}} \tag{9b}$$

and we can find one such δ by solving the following linear program

$$\begin{aligned}
&\text{maximize} \\
&\quad \varepsilon \\
&\text{subject to} \\
&\qquad \varepsilon \leq 1 \\
&\langle v, \delta \rangle = 0, && v \in L_{\text{sub}} \\
&\langle v, \delta \rangle \leq -\varepsilon, && v \in V_{\text{sub}} \setminus L_{\text{sub}}
\end{aligned}$$

where δ is a vector, ε is a scalar, and (δ, ε) denotes a vector of length one more than the length of δ. This vector is the vector of variables of the linear program. The δ part of the solution is a generic direction of recession. The ε part does not matter.

So we solve this linear program to calculate the GDOR, still following Section 4.1 of Geyer (2008).

```
> p <- ncol(tanv)
> hrep <- cbind(0, 0, -tanv, -1)
> hrep <- rbind(hrep, c(0, 1, rep(0, p), -1))
> objv <- c(rep(0, p), 1)
> pout <- lpcdd(d2q(hrep), d2q(objv), minimize = FALSE)
> names(pout)

[1] "solution.type"   "primal.solution" "dual.solution"
[4] "optimal.value"

> pout$solution.type

[1] "Optimal"

> gdor <- q2d(pout$primal.solution[1:p])
> gdor

[1] -5.0  0.1
```

The code chunk above is not general. It assumes the linearity is trivial, as in the particular example we are working on. More on this later.

So now we have a GDOR, we should put that on the plot, but we cannot. The reason is that δ is a vector in the parameter space (as we have been saying over and over), but the space plotted in Figure 2 is the sample space for the canonical statistic vector. (I tried. There is no way to draw δ into Figure 2.) What we can do is add H_δ. See Figure 4. The fact that the only possible value of the canonical statistic vector that is on H_δ is the observed value y again tells us that the LCM is completely degenerate, concentrated at the observed value.

2.9 Summary

So now we have the MLE for the LCM, which is the limit of distributions in the OM that maximizes the likelihood. We should say that it is somewhat unusual.

The MLE of the mean value parameter satisfies the "observed equals expected" property $\hat{\mu} = y$. That is because it is the MLE in the LCM.

The MLE of the canonical parameter is very weird. Firstly, it doesn't exist. Second, we can think of it as a point at infinity. Start at the MLE for the canonical parameter in the LCM (which does exist) and head to infinity in the direction of the GDOR. The likelihood in the OM increases all the

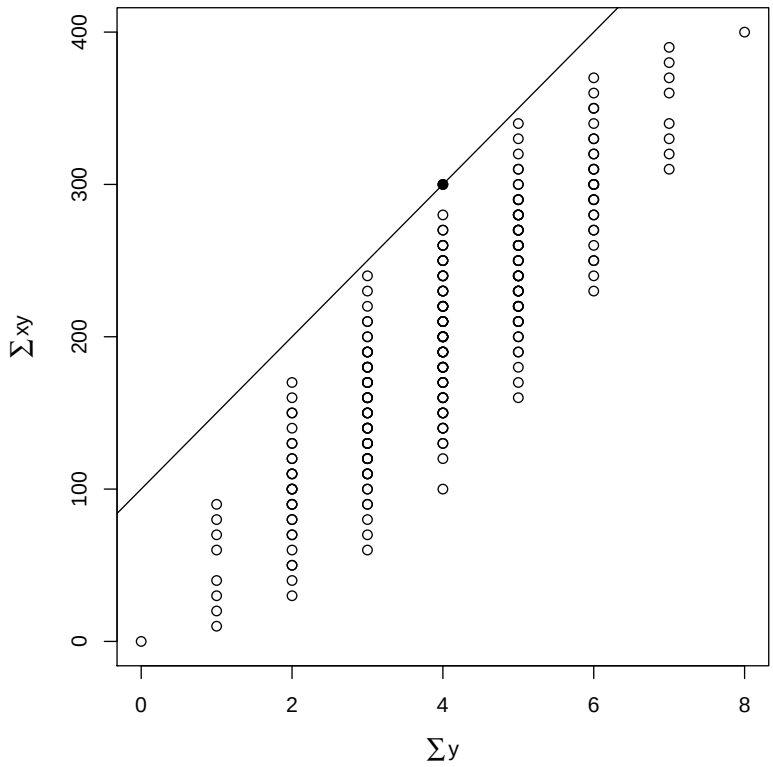

Figure 4: Possible values of the submodel canonical statistic vector $M^T y$ for the data shown in Figure 1. Solid dot is the observed value of the submodel canonical statistic vector. Solid line is the hyperplane H_δ on which the LCM is concentrated.

way but does asymptote at the supremum of the likelihood. The supremum is never achieved (that is why we say "supremum" rather than "maximum"), but we do converge to it.

In Example II the MLE in the LCM is any point in $\mathbb{R}^2$. Since the LCM is completely degenerate, every direction in the parameter space is a direction of constancy (for the LCM) so every point in $\mathbb{R}^2$ corresponds to the same distribution. This is not surprising since there is only one distribution in the LCM (the one concentrated at the observed data). Thus we can think of the MLE for the canonical parameter as start anywhere and head to infinity in the direction of the GDOR.

2.10 Statistics

Hypothesis tests and confidence intervals become difficult and hard to understand when the MLE does not exist in the OM. We will say a lot more about both, but will creep up on them slowly, waiting until we have some interesting examples.

But there is one statistical procedure we can do on Example II. The usual Wilks and Rao hypothesis tests work just fine when n is large (their assumptions are satisfied) when the MLE exists for the null hypothesis. There is no need for the MLE to exist for the alternative hypothesis. The reason is that the Rao test statistic does not use the MLE for the alternative hypothesis, and the Wilks test only seems to use the MLE for both hypotheses. If Θ_0 and Θ_1 are the two hypotheses and l_n is the log likelihood, the Wilks test statistic can be written

$$T_n = 2 \left(\sup_{\theta \in \Theta_1} l_n(\theta) \right) - 2 \left(\sup_{\theta \in \Theta_0} l_n(\theta) \right)$$

and we can calculate the supremum of the log likelihood even when the MLE does not exist (just keep going uphill on the log likelihood until the increases in the steps get very small).

Hence

```
> gout.0 <- glm(y ~ 1, family = binomial)
> gout.1 <- glm(y ~ x, family = binomial)
> anova(gout.0, gout.1, test = "Chisq")

Analysis of Deviance Table

Model 1: y ~ 1
```

```
Model 2: y ~ x
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1         7      11.09
2         6       0.00  1    11.09 0.0008678 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

is completely valid (except for n being small). So it is clear that just using a smaller model for which the MLE exists is a non-starter. We have to use the theory of this document when we can prove that no model for which the MLE exists fits the data.

Similarly,

```
> add1(gout.0, ~ x, test = "Rao")

Single term additions

Model:
y ~ 1
       Df Deviance   AIC Rao score Pr(>Chi)
<none>       11.09 13.09
x       1     0.00  4.00    6.6667 0.009823 **
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

is just as valid. The fact that these two test do not agree about the P-value is because n is small.

The R functions `glm` and `add1` issue warnings

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

(which Sweave does not capture). These can be ignored in this case. We know what we are doing. The tests are valid (except for smallness of n) because the MLE exists for the null hypothesis (the one with the formula `y ~ 1`).

Before we leave, save the GDOR (the next example will clobber it). Also save the model.

```
> ex.ii <- list(x = x, y = y, m = m)
> ex.ii.gdor <- gdor
```

2.11 Example III

This is the other example mentioned in Agrest. If we add two data points to the data for Example II with $x = 50$ and one success and one failure,

```
> x <- c(x, 50, 50)
> y <- c(y, 0, 1)
```

then we will not have a completely degenerate LCM and will have to work a little harder. Agresti calls this case quasi-complete separation.

We find the linearity as before

```
> m <- cbind(1, x)
> tanv <- m
> tanv[y == 1, ] <- (-tanv[y == 1, ])
> vrep <- makeV(rays = tanv)
> lout <- linearity(d2q(vrep), rep = "V")
> lout

[1]  9 10

> tanv[lout, ]

          x
[1,]  1  50
[2,] -1 -50
```

We already know (by definition) that if v is in the linearity, so is $-v$. So the linearity is actually one-dimensional here. Again, the fact that not all tangent vectors are in the linearity means that the MLE in the OM does not exist.

Next we determine the GDOR. The R code now has to account for the linearity not being trivial.

```
> p <- ncol(tanv)
> hrep <- cbind(0, 0, -tanv, -1)
> hrep[lout, 1] <- 1
> hrep[lout, p + 3] <- 0
> hrep <- rbind(hrep, c(0, 1, rep(0, p), -1))
> objv <- c(rep(0, p), 1)
> pout <- lpcdd(d2q(hrep), d2q(objv), minimize = FALSE)
> names(pout)
```

```
[1] "solution.type"   "primal.solution" "dual.solution"
[4] "optimal.value"
```

```
> pout$solution.type
```

```
[1] "Optimal"
```

```
> gdor <- q2d(pout$primal.solution[1:p])
> gdor
```

```
[1] -5.0  0.1
```

Now we make a figure like Figure 4 except for these data. It is Figure 5. We can see that the support of the LCM contains three points, because there are three points on the line in Figure 5.

Another way to figure out the support of the LCM that does not involve looking at a picture is to find the GDOR for the saturated model canonical parameter. We find the LCM by taking limits as $s \to \infty$ of distributions with submodel canonical parameter $\beta + s\delta$, which corresponds to the saturated model canonical parameter

$$M(\beta + s\delta) = \theta + s\eta,$$

where $\theta = M\beta$ and $\eta = M\delta$. Thus η is a GDOR in the saturated model canonical parameterization.

For any component of η that is positive, the corresponding component of the response vector y is fixed at the upper limit of its range in the LCM. For any component of η that is negative, the corresponding component of the response vector y is fixed at the lower limit of its range in the LCM. Such fixed components must agree with the observed data. Let's check.

```
> eta <- m %*% gdor
> y[eta < 0]
```

```
[1] 0 0 0 0
```

```
> y[eta > 0]
```

```
[1] 1 1 1 1
```

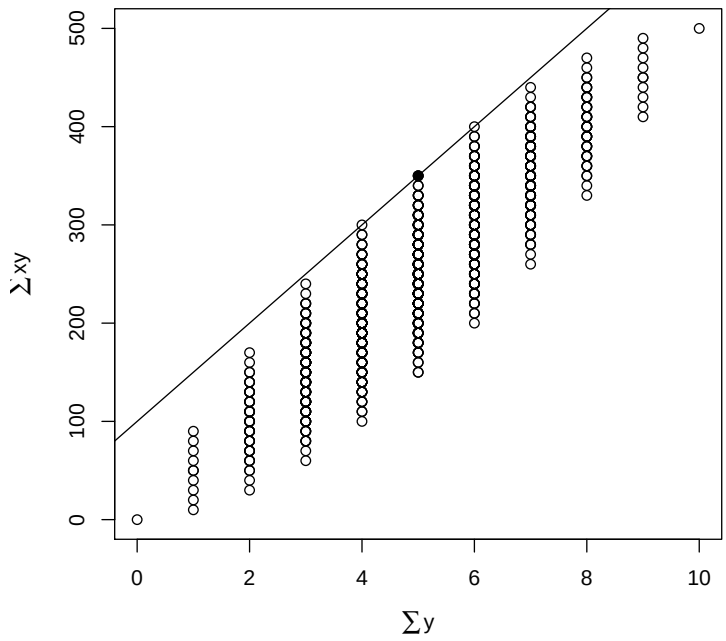

Figure 5: Possible values of the submodel canonical statistic vector $M^T y$ for the data of Example III. Solid dot is the observed value of the submodel canonical statistic vector. Solid line is the hyperplane H_δ on which the LCM is concentrated.

2.12 A Digression about Computer Arithmetic

That was an unexpected disaster (a bug). But it should not have been.

```
> as.vector(eta)
```

```
 [1] -4 -3 -2 -1  1  2  3  4  0  0
```

Those really small components of `eta` are probably zero, but R has no way to know that.

Every time we say `library(rcdd)` it prints

```
If you want correct answers, use rational arithmetic.
See the Warnings sections added to help pages for
    functions that do computational geometry.
```

We used rational arithmetic in the computations done by `rcdd` but then we converted back to ordinary arithmetic. That's what the R function `q2d` did when we computed the GDOR. Had we not done this, we would have had an exact GDOR

```
> gdor.exact <- pout$primal.solution[1:p]
> gdor.exact
```

```
[1] "-5"   "1/10"
```

```
> all(m == round(m))
```

```
[1] TRUE
```

```
> eta.exact <- qmatmult(d2q(m), cbind(gdor.exact))
> as.vector(eta.exact)
```

```
 [1] "-4" "-3" "-2" "-1" "1"  "2"  "3"  "4"  "0"  "0"
```

What happened to the ordinary computer arithmetic calculation?

```
> d2q(q2d(gdor.exact))
```

```
[1] "-5"
[2] "7205759403792793/72057594037927936"
```

The problem is that 1/10 is a "round number" in decimal arithmetic but not a "round number" in binary arithmetic, which computers use. This is a widespread issue that every user of computers for anything numerical should understand, but most don't. We quote from the R FAQ (Hornik, 2016)

7.31 Why doesn't R think these numbers are equal?

The only numbers that can be represented exactly in R's numeric type are integers and fractions whose denominator is a power of 2. Other numbers have to be rounded to (typically) 53 binary digits accuracy. As a result, two floating point numbers will not reliably be equal unless they have been computed by the same algorithm, and not always even then. For example

```
R> a <- sqrt(2)
R> a * a == 2
[1] FALSE
R> a * a - 2
[1] 4.440892e-16
```

The function `all.equal()` compares two objects using a numeric tolerance of `.Machine$double.eps ^ 0.5`. If you want much greater accuracy than this you will need to consider error propagation carefully.

For more information, see e.g. David Goldberg (1991), "What Every Computer Scientist Should Know About Floating-Point Arithmetic", *ACM Computing Surveys*, **23/1**, 5–48, ...

To quote from "The Elements of Programming Style" by Kernighan and Plauger:

> 10.0 times 0.1 is hardly ever 1.0.

Let's try it.

```
> 10 * 0.1 == 1.0
```

```
[1] TRUE
```

That's tricky. I don't know how R manages to do that. Let's try another.

```
> 0.1 + 0.2 + 0.3 + 0.4 == 1.0
```

```
[1] TRUE
```

Here is an example from the help page for the `==` operator obtained by doing `?"=="` in R.

```
> x1 <- 0.5 - 0.3
> x2 <- 0.3 - 0.1
> x1 == x2          # FALSE on most machines
```

```
[1] FALSE
```

2.13 Example III Continued

Now the check that `eta.exact` does the right thing.

```
> y[qsign(eta.exact) < 0]

[1] 0 0 0 0

> y[qsign(eta.exact) > 0]

[1] 1 1 1 1
```

So we are finally in position to find the MLE for the LCM.

```
> gout.lcm <- glm(y ~ x, family = binomial, subset = qsign(eta.exact) == 0)
> summary(gout.lcm)

Call:
glm(formula = y ~ x, family = binomial, subset = qsign(eta.exact) ==
    0)

Deviance Residuals:
     9      10
-1.177   1.177

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 4.710e-16  1.414e+00       0        1
x                  NA         NA      NA       NA

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2.7726  on 1  degrees of freedom
Residual deviance: 2.7726  on 1  degrees of freedom
AIC: 4.7726

Number of Fisher Scoring iterations: 2
```

Because the model is partially degenerate, it drops one parameter. Although R reports the value as `NA`, this actually means zero, because dropping a predictor variable is the same as setting its regression coefficient to be zero.

The "intercept" parameter is also zero. The reported value `4.71e-16` is just more inexactness of computer arithmetic.

The only components of the response vector that are not fixed in the LCM are the two components that have the corresponding components of x equal to 50. There is one success and one failure, so we estimate $1/2$ for the success probability, and $\text{logit}(1/2) = 0$.

3 Confidence Intervals and Regions

3.1 Submodel Canonical Parameters

A confidence region is like a confidence interval except that it covers a vector parameter. In this section we do confidence regions for the canonical parameter vector of a canonical affine submodel, also known as the "coefficients" in the terminology of the R function `glm`.

We know from Wilk's theorem that, assuming the "usual regularity conditions,"

$$2 \left[\sup_{\beta \in B} l_n(\beta) - l_n(\beta_0) \right] \approx \text{chi-square}(p),$$

where l_n is the log likelihood, p is the dimension of β, and B is the parameter space for β. If we let $\chi^2(p)_\alpha$ denote the $1 - \alpha$ quantile of the chi-square(p) distribution, and let

$$c = \sup_{\beta \in B} l_n(\beta) - \chi^2(p)_\alpha$$

then

$$\{ \, \beta \in B : l_n(\beta) \geq c \, \}$$

is a large sample (asymptotic) confidence region for β with (approximate) coverage probability $1 - \alpha$.

Let's try it. First we need the log likelihood function, which we do for Bernoulli regression only because that enough for our two examples.

```
> check <- function(model) {
+     stopifnot(is.list(model))
+     stopifnot(c("x", "y", "m") %in% names(model))
+     stopifnot(is.matrix(model$m))
+     stopifnot(nrow(model$m) == length(model$y))
+     stopifnot(nrow(model$m) == length(model$x))
+     stopifnot(model$y %in% c(0, 1))
+ }
```

```
> logl <- function(beta, model) {
+     check(model)
+     m <- model$m
+     y <- model$y
+     eta <- as.numeric(m %*% beta)
+     logp <- ifelse(eta < 0, eta - log1p(exp(eta)), - log1p(exp(- eta)))
+     logq <- ifelse(eta < 0, - log1p(exp(eta)), - eta - log1p(exp(- eta)))
+     sum(y * logp) + sum((1 - y) * logq)
+ }
> score <- function(beta, model) {
+     check(model)
+     m <- model$m
+     y <- model$y
+     eta <- as.vector(m %*% beta)
+     p <- 1 / (1 + exp(- eta))
+     q <- 1 / (1 + exp(eta))
+     f <- ifelse(y == 1, q, -p)
+     foo <- rbind(f) %*% m
+     dimnames(foo) <- NULL
+     foo
+ }
```

Check numerically that the score function (derivative of log likelihood) seems to be all right.

```
> beta <- rnorm(2, sd = 0.1)
> ex.iii <- list(x = x, y = y, m = m)
> logl(beta, ex.iii)

[1] -5.786526

> score(beta, ex.iii)

          [,1]      [,2]
[1,] -1.674873 -10.56689

> library(numDeriv)
> grad(logl, beta, model = ex.iii)

[1]  -1.674873 -10.566895
```

Now we need the maximized value of the log likelihood

```
> foo <- function(beta) {
+     fred <- (- logl(beta, ex.iii))
+     attr(fred, "gradient") <- (- score(beta, ex.iii))
+     fred
+ }
> foo(beta)

[1] 5.786526
attr(,"gradient")
         [,1]     [,2]
[1,] 1.674873 10.56689

> nout <- nlm(foo, beta)
> nout$code <= 2

[1] TRUE

> sup.logl <- (- nout$minimum)
> sup.logl

[1] -1.386294

> alpha <- 0.05
> crit <- qchisq(alpha, df = 2, lower.tail = FALSE) / 2
> crit

[1] 2.995732
```

In order to trace the boundary of the critical region, we need to find a starting point on the boundary.

```
> library(alabama)
> confun <- function(beta) sup.logl - logl(beta, ex.iii) - crit
> conjac <- function(beta) (- score(beta, ex.iii))
> aout <- auglag(rep(0, 2), function(beta) sum(beta^2) / 2,
+     function(beta) beta, heq = confun, heq.jac = conjac,
+     control.outer = list(trace = FALSE))
> stopifnot(aout$convergence == 0)
> beta <- aout$par
> confun(beta)

[1] -2.441516e-09
```

There seem to be no R packages that do what we want here, so we just invent our own algorithm. From the current point on the curve β we do the following, letting h stand for the function (`confun` in R) that implicitly defines the curve we are following, the set of β such that $h(\beta) = 0$. We let ε stand for an arbitrary small positive number.

1. $g := \nabla h(\beta)$.
2. $v := (-g_2, g_1)$ { rotate g by 90° }.
3. $v := v/\|v\|$.
4. $\beta := \beta + \varepsilon \cdot v$.

Now we have moved a ways along the curve (about ε in whatever norm $\|\cdot\|$ indicates), or, more precisely, "almost" along the curve, because we took a step along the straight line tangent to the curve.

We do Newton iterations to get back onto the curve. Again letting $g = \nabla h(\beta)$, we try to find the s such that $h(\beta + sg) = 0$. The Taylor series up to first derivative terms is

$$h(\beta + sg) \approx h(\beta) + s\langle \nabla h(\beta), g \rangle.$$

Setting this equal to zero and solving for s gives

$$s = -\frac{h(\beta)}{\|\nabla h(\beta\|^2},$$

where now we have to be using the Euclidean norm

$$\|g\|^2 = \langle g, g \rangle.$$

Then we set $\beta = \beta + s\nabla h(\beta)$ and then we iterate to convergence (till we are on the curve again). Let τ be another small number (the convergence tolerance). Then we continue our algorithm

5. Repeat the following steps.
   (a) $f := h(\beta)$.
   (b) If ($\|f\| < \tau$) stop the repetition.
   (c) $g := \nabla h(\beta)$.
   (d) $s := -f/\|g\|^2$.
   (e) $\beta := \beta + sg$.

And all of that takes us just one step along the curve. Then we repeat the whole thing over and over, taking many steps to trace the whole curve. Since the curve goes to infinity when the MLE does not exist in the conventional sense, we stop when we have gone as far as we want to plot.

```
> fred <- function(confun, conjac, epsilon, max.beta, tau = 1e-6) {
+
+     norm <- function(beta) sqrt(sum(beta^2))
+     is.out <- function(beta) max(abs(beta)) > max.beta
+
+     aout <- auglag(rep(0, 2), function(beta, ...) sum(beta^2) / 2,
+         function(beta, ...) beta, heq = confun, heq.jac = conjac,
+         control.outer = list(trace = FALSE))
+         stopifnot(aout$convergence == 0)
+
+     beta <- aout$par
+     betas <- aout$par
+     repeat {
+         g <- as.vector(conjac(beta))
+         v <- c(- g[2], g[1])
+         v <- v / norm(v)
+         beta <- beta + v * epsilon
+         repeat {
+             f <- confun(beta)
+             if (norm(f) < tau) break
+             g <- as.vector(conjac(beta))
+             s <- (- f) / sum(g^2)
+             beta <- beta + s * g
+         }
+         if (is.out(beta)) break
+         betas <- rbind(betas, beta)
+     }
+     beta <- aout$par
+     repeat {
+         g <- as.vector(conjac(beta))
+         v <- c(g[2], - g[1])
+         v <- v / norm(v)
+         beta <- beta + v * epsilon
+         repeat {
+             f <- confun(beta)
```

```
+             if (norm(f) < tau) break
+             g <- as.vector(conjac(beta))
+             s <- (- f) / sum(g^2)
+             beta <- beta + s * g
+         }
+         if (is.out(beta)) break
+         betas <- rbind(beta, betas)
+     }
+
+     rownames(betas) <- NULL
+     colnames(betas) <- c("x", "y")
+     betas
+ }
> trace.ex.iii <- fred(confun, conjac, 0.01, 6)
```

The result is shown in Figure 6. This figure shows more than ever that canonical parameters are meaningless (compare Sections 6.3 and 6.7.11 of Part I of these notes). The confidence region is unbounded, going all the way to infinity. We could convert this confidence region to one-sided intervals for the separate parameters, $(-\infty, -1.42)$ for β_1 and $(0.033, +\infty)$ for β_2, but when we convert those intervals back to a confidence region with sides parallel to the axes, it is infinitely larger than the confidence region shown in the figure.

For comparison, we also do the corresponding region for Example II.

```
> qux <- function(beta) {
+     fred <- (- logl(beta, ex.ii))
+     attr(fred, "gradient") <- (- score(beta, ex.ii))
+     fred
+ }
> nout <- nlm(qux, beta)
> nout$code <= 2

[1] TRUE

> sup.logl <- (- nout$minimum)
> sup.logl

[1] -1.077311e-08

> confun <- function(beta) sup.logl - logl(beta, ex.ii) - crit
> conjac <- function(beta) (- score(beta, ex.ii))
> trace.ex.ii <- fred(confun, conjac, 0.01, 6)
```

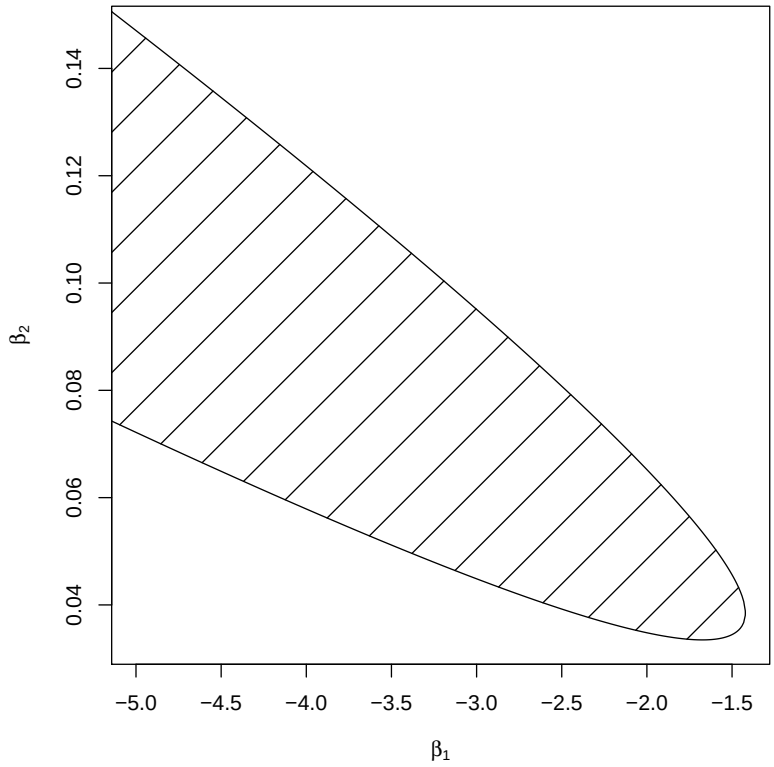

Figure 6: 95% Confidence Region for Submodel Canonical Parameter (a. k. a., Coefficients) for Example III.

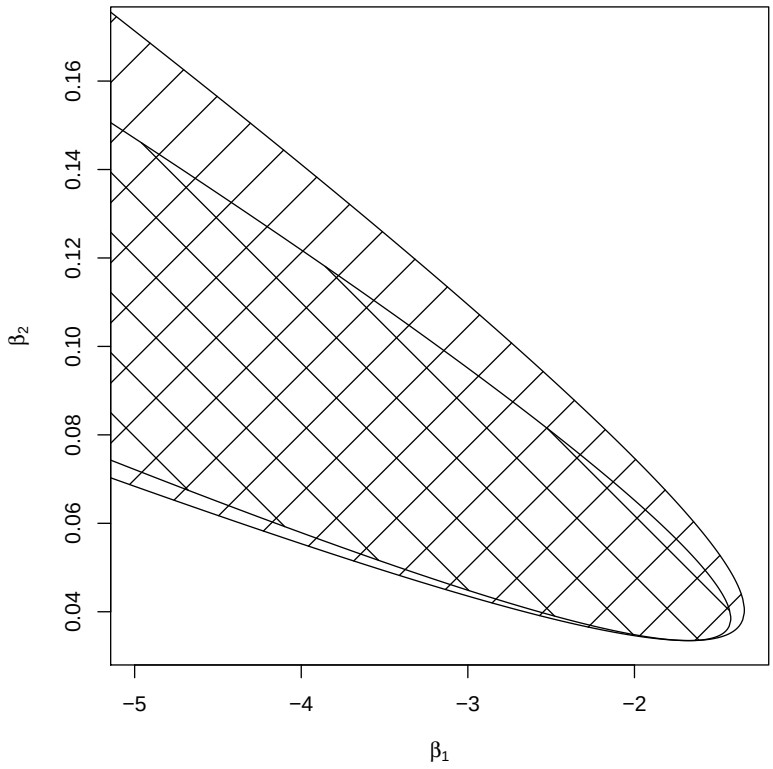

Figure 7: 95% Confidence Regions for Submodel Canonical Parameter (a. k. a., Coefficients) for Examples II and III. Region for Example II is larger, shaded at 45° angle. Region for Example II is smaller, shaded at −45° angle.

Figure 7 shows this confidence region.

3.2 Mean Value Parameters

Geyer (2009b, Section 3.16) gives a recipe for making confidence intervals for mean value parameters that are fixed in the LCM.

We will not follow that but continue with likelihood-based confidence regions and intervals. For intervals we follow the same procedure as was mention in the preceding section. First form the confidence region (at least conceptually), then calculate intervals by finding the smallest and largest values of the parameter of interest in that region. This recipe actually works for any function of the parameter vector not just for components of the parameter vector.

When doing confidence intervals, we can have intervals that have 95% simultaneous coverage, which is what we would get if we maximize over the confidence regions shown in Figures 6 and 7, or we can have intervals that have 95% non-simultaneous coverage. The latter sort of interval is what is usually done. Users like them because they are smaller (or perhaps users don't like them but programmers think users like them so that is what they provide). But they invite misinterpretation. It is hard to understand that the assertion is that there is (approximately, for large n) 95% probability that any particular interval covers the parameter it is supposed to capture, but perhaps much less than 95% probability that all the intervals cover the parameters they are supposed to capture.

Nevertheless, we follow tradition and provide both. To get non-simultaneous intervals we simply change the critical value, using the critical value for one degree of freedom.

```
> crit.non.simultaneous <- qchisq(alpha, df = 1, lower.tail = FALSE) / 2
> crit.non.simultaneous
```

```
[1] 1.920729
```

In forming these intervals, we do not actually form the confidence region and then maximize a function over it. Instead we maximize and minimize the scalar parameter of interest subject to the constraint that the parameter vector lie in the confidence region.

Let's try that out for the data of Example II and parameter μ_1. We know the confidence interval will be one-sided. So we don't need to calculate the lower bound of the interval. Since the optimization routine we will use (function `auglag` in package `alabama`, which we used above) only minimizes, to maximize we minimize minus the function.

```
> objfun <- function(beta, model, i) {
+     check(model)
+     stopifnot(is.numeric(i))
+     stopifnot(length(i) == 1)
+     stopifnot(i %in% 1:nrow(model$m))
+     m <- model$m
+     y <- model$y
+     theta <- m %*% beta
+     p <- 1 / (1 + exp(- theta[i]))
+     q <- 1 / (1 + exp(theta[i]))
+     if(y[i] == 1) return(p) else return(q)
+ }
> objgrd <- function(beta, model, i) {
+     check(model)
+     stopifnot(is.numeric(i))
+     stopifnot(length(i) == 1)
+     stopifnot(i %in% 1:nrow(model$m))
+     m <- model$m
+     y <- model$y
+     theta <- m %*% beta
+     p <- 1 / (1 + exp(- theta[i]))
+     q <- 1 / (1 + exp(theta[i]))
+     g <- as.vector(p * q * m[i, ])
+     if(y[i] == 1) return(g) else return(- g)
+ }
> beta <- rnorm(2, sd = 0.1)
> objfun(beta, ex.ii, 1)

[1] 0.2232844

> objgrd(beta, ex.ii, 1)

[1] -0.1734285 -1.7342850

> grad(objfun, beta, model = ex.ii, i = 1)

[1] -0.1734285 -1.7342850

> confun <- function(beta, ...) crit - (sup.logl - logl(beta, ex.ii))
> conjac <- function(beta, ...) score(beta, ex.ii)
> confun(beta)
```

```
[1] -8.952064

> conjac(beta)

         [,1]      [,2]
[1,] -3.64546 -94.53861

> grad(confun, beta)

[1]  -3.64546 -94.53861

> beta.start <- colMeans(trace.ex.ii)
> aout <- auglag(beta.start, objfun, objgrd, hin = confun,
+     hin.jac = conjac, model = ex.ii, i = 1,
+     control.outer = list(trace = FALSE, method = "nlminb"))
> aout$convergence == 0

[1] TRUE

> confun(aout$par)

[1] 1.002372e-09

> objgrd(aout$par, ex.ii, 1) / conjac(aout$par)

          [,1]      [,2]
[1,] 0.1646102 0.1646101
```

It looks like it is working. At the solution we should have the gradient of the objective function proportional to the gradient of the constraint function if the constraint is zero (meaning the solution is on the boundary). This implies the only directions downhill on the objective function take us out of the constraint set.

So let's see what we get when applied to all the mean value parameters.

```
> ci.raw <- rep(NA, length(ex.ii$y))
> for (i in seq(along = ci.raw)) {
+     aout <- auglag(beta.start, objfun, objgrd, hin = confun,
+         hin.jac = conjac, model = ex.ii, i = i,
+         control.outer = list(trace = FALSE, method = "nlminb"))
+     stopifnot(aout$convergence == 0)
+     ci.raw[i] <- aout$value
+ }
> ci.raw
```

```
[1] 0.71474998 0.60596408 0.42917079 0.05000004 0.05000000
[6] 0.42917079 0.60596408 0.71474998

> ci.low <- ifelse(ex.ii$y == 1, ci.raw, 0)
> ci.hig <- ifelse(ex.ii$y == 1, 1, 1 - ci.raw)
> ci.low

[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0500000
[6] 0.4291708 0.6059641 0.7147500

> ci.hig

[1] 0.2852500 0.3940359 0.5708292 0.9500000 1.0000000
[6] 1.0000000 1.0000000 1.0000000
```

Figure 8 shows these simultaneous confidence intervals for μ.

Now let's redo, non-simultaneously. The only thing we need to change is `confun`.

```
> confun <- function(beta, ...) crit.non.simultaneous -
+     (sup.logl - logl(beta, ex.ii))
> ci.raw <- rep(NA, length(ex.ii$y))
> for (i in seq(along = ci.raw)) {
+     aout <- auglag(beta.start, objfun, objgrd, hin = confun,
+         hin.jac = conjac, model = ex.ii, i = i,
+         control.outer = list(trace = FALSE, method = "nlminb"))
+     stopifnot(aout$convergence == 0)
+     ci.raw[i] <- aout$value
+ }
> ci.raw

[1] 0.8822385 0.7866289 0.6078324 0.1465001 0.1465002
[6] 0.6078324 0.7866289 0.8822385

> ci.low <- ifelse(ex.ii$y == 1, ci.raw, 0)
> ci.hig <- ifelse(ex.ii$y == 1, 1, 1 - ci.raw)
> ci.low

[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.1465002
[6] 0.6078324 0.7866289 0.8822385

> ci.hig
```

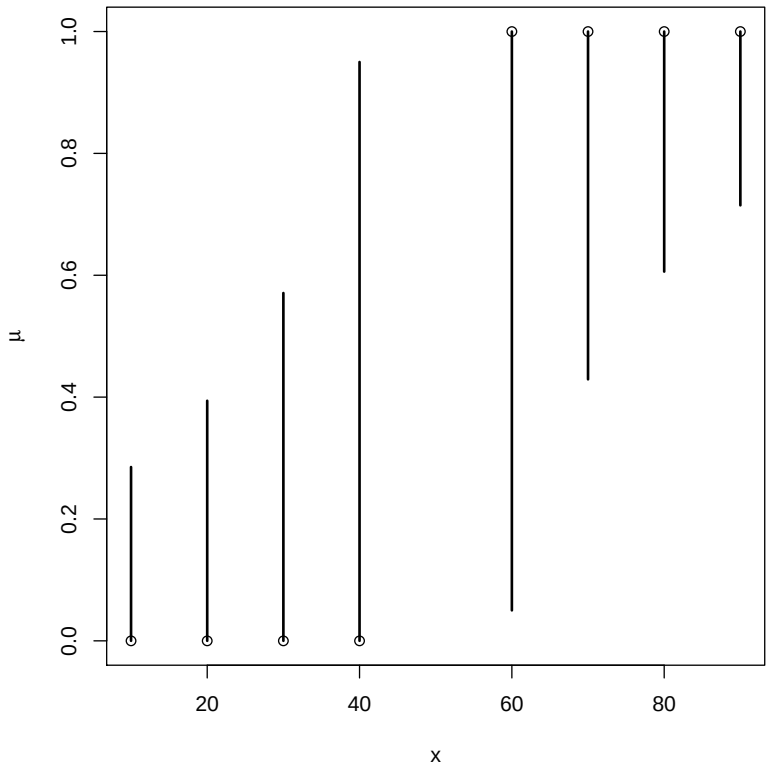

Figure 8: Logistic Regression Data (dots) for Example II. With 95% simultaneous likelihood-based confidence intervals for the success probabilities (bars).

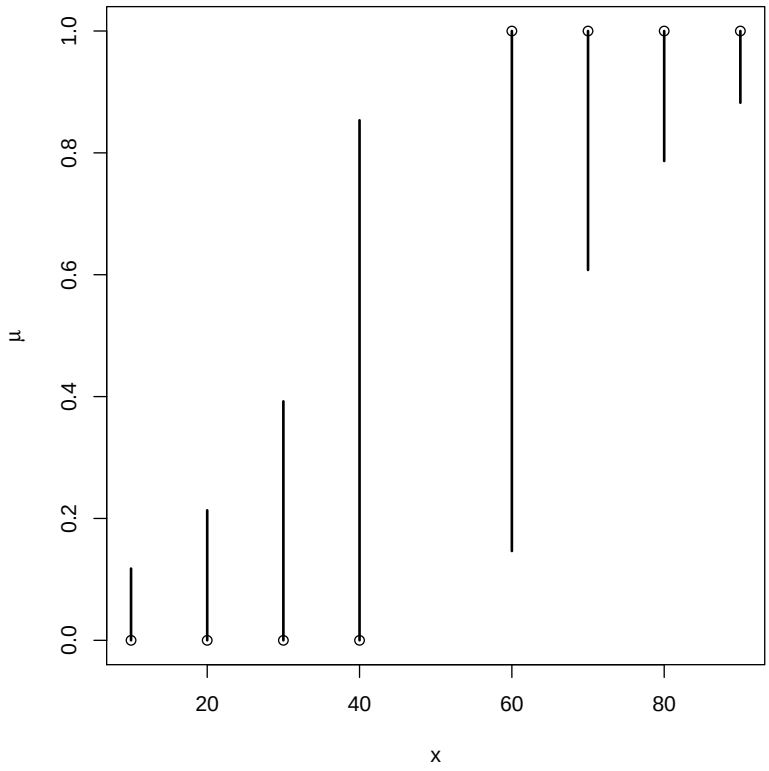

Figure 9: Logistic Regression Data (dots) for Example II. With 95% non-simultaneous likelihood-based confidence intervals for the success probabilities (bars).

```
[1] 0.1177615 0.2133711 0.3921676 0.8534999 1.0000000
[6] 1.0000000 1.0000000 1.0000000
```

Figure 9 shows these non-simultaneous confidence intervals for μ. These are a lot tighter but may be subject to misinterpretation as simultaneous intervals.

Now we do the same for Example III.

```
> confun <- function(beta, ...) crit - (sup.logl - logl(beta, ex.iii))
> conjac <- function(beta, ...) score(beta, ex.iii)
> ci.raw <- rep(NA, length(ex.iii$y))
> for (i in seq(along = ci.raw)) {
+     aout <- auglag(beta.start, objfun, objgrd, hin = confun,
```

```
+         hin.jac = conjac, model = ex.iii, i = i,
+         control.outer = list(trace = FALSE, method = "nlminb"))
+     stopifnot(aout$convergence == 0)
+     ci.raw[i] <- aout$value
+ }
> ci.raw

 [1] 0.92917552 0.85956225 0.72800115 0.48279350 0.48279351
 [6] 0.72800115 0.85956225 0.92917552 0.05278641 0.05278640

> xx <- sort(unique(ex.iii$x))
> ci.low <- rep(NA, length(xx))
> ci.hig <- rep(NA, length(xx))
> for (i in seq(along = xx)) {
+     j <- which(ex.iii$x == xx[i])
+     yy <- ex.iii$y[j]
+     if (all(yy == 1)) {
+         ci.hig[i] <- 1
+         ci.low[i] <- min(ci.raw[j])
+     } else if (all(yy == 0)) {
+         ci.low[i] <- 0
+         ci.hig[i] <- max(1 - ci.raw[j])
+         ci.low[i] <- 0
+     } else {
+         ci.low[i] <- min(ci.raw[j])
+         ci.hig[i] <- max(1 - ci.raw[j])
+     }
+ }
> xx

[1] 10 20 30 40 50 60 70 80 90

> ci.low

[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0527864
[6] 0.4827935 0.7280011 0.8595622 0.9291755

> ci.hig

[1] 0.07082448 0.14043775 0.27199885 0.51720650 0.94721360
[6] 1.00000000 1.00000000 1.00000000 1.00000000
```

Figure 8 shows these simultaneous confidence intervals for μ.

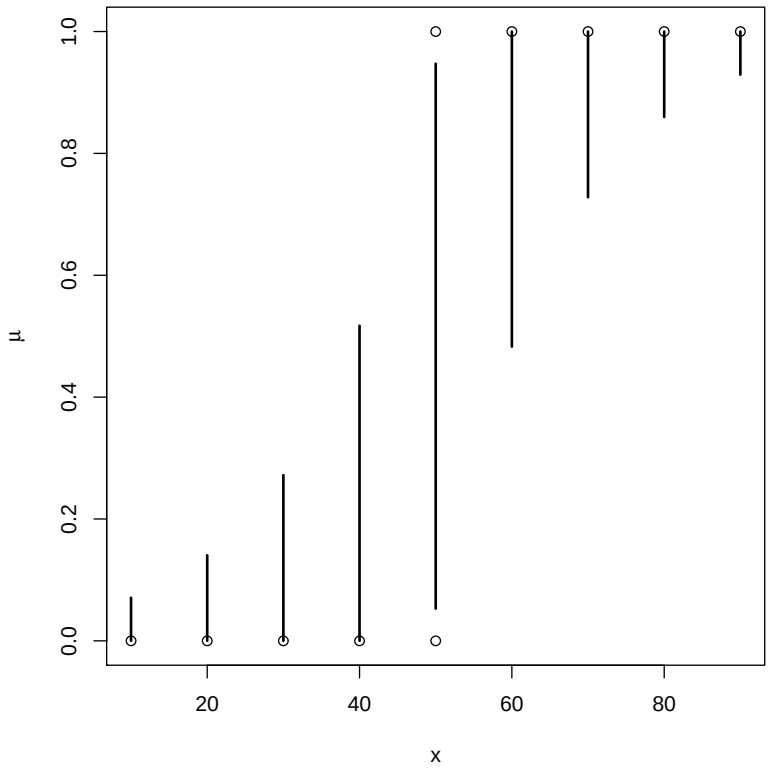

Figure 10: Logistic Regression Data (dots) for Example III. With 95% simultaneous likelihood-based confidence intervals for the success probabilities (bars).

References

Agresti, A. (2013). *Categorical Data Analysis*, third edition. John Wiley & Sons, Hoboken, NJ.

Barndorff-Nielsen, O. (1978). *Information and Exponential Families*. John Wiley & Sons, Chichester, England.

Brown, L. D. (1986). *Fundamentals of Statistical Exponential Families: with Applications in Statistical Decision Theory*. Institute of Mathematical Statistics, Hayward, CA.

Geyer, C. J. (1990). Likelihood and Exponential Families. PhD thesis, University of Washington. `http://purl.umn.edu/56330`.

Geyer, C. J. (2008). Supporting theory and data analysis for "likelihood inference in exponential families and directions of recession". Technical Report 672, School of Statistics, University of Minnesota. `http:www.stat.umn.edu/geyer/gdor/phaseTR.pdf`.

Geyer, C. J. (2009a). More supporting data analysis for "likelihood inference in exponential families and directions of recession". Technical Report 673, School of Statistics, University of Minnesota. `http:www.stat.umn.edu/geyer/gdor/phase2TR.pdf`.

Geyer, C. J. (2009b). Likelihood inference in exponential families and directions of recession. *Electronic Journal of Statistics*, **3**, 259–289.

Geyer, C. J. (2016). Exponential families, part I. Lecture notes for Stat 5421 (categorical data analysis). `http://www.stat.umn.edu/geyer/5421/notes/expfam.pdf`.

Geyer, C. J., and Meeden, G. D. (2005). Fuzzy and randomized confidence intervals and P-values (with discussion). *Statistical Science*, **20**, 358–387.

Geyer, C. J., Meeden, G. D., and Fukuda, K. (2016). R package `rcdd` (Computational Geometry), version 1.1-10. `http://CRAN.R-project.org/package=rcdd`.

Hornik, K. (2016). The R FAQ. `https://CRAN.R-project.org/doc/FAQ/R-FAQ.html`.

Rockafellar, R. T. 1970. *Convex Analysis*. Princeton University Press, Princeton, NJ.

Varadhan, R. (2015). R package `alabama` (Constrained Nonlinear Optimization), version 2015.3-1. `https://CRAN.R-project.org/package=alabama`.