Some computational and methodological notes on complete separation

Daniel J. Eck

6/13/2020

In this document we investigate some computational and methodological issues related to the complete separation issue in logistic regression specifically, and the canonical parameter estimated to be on the boundary problem generally.

The following references proposed an inferential solution to this boundary problem through the lens of maximum likelihood estimation:

1. Computationally efficient likelihood inference in exponential families when the maximum likelihood estimator does not exist by Daniel J. Eck and Charles J. Geyer (see here)
2. Supporting Theory and Data Analysis for "Likelihood Inference in Exponential Families and Directions of Recession" by Charles J. Geyer (see here)

Projects in this area include but are not limited to:

1. **Comparing methods.** The above references are great and all, but they have not been compared and contrasted to existing "solutions" that appear in the literature. In addition, the framework in these references is theoretical, hard to read, hard to understand, and, although software exists, our implementation is currently small in scope and has not been extensively tested in the same vein of existing "solutions".

I write existing "solutions" with solutions in quotes because, in my opinion, these existing "solutions" are more of work arounds than actual solutions. For example, when one fits a logistic regression model in classical settings ($n > p$ or $n \gg p$) that exhibited no separation problems, then one not has no reason to consider solutions to separation when separation isn't observed. Instead, one would proceed with interpreting the model output. However, when one encounters separation the game has changed, parameter estimates are on the boundary and the original logistic model is "broken" or degenerate without obvious remedies. To remedy this problem people invented approaches which retrospectively change the assumed modeling paradigm or add data in clever ways. Our philosophy is that nothing new is wrong with the modeling paradigm in the presence of separation, the problem is with observed data that is allowable under the model in a finite sample. Consistent with this philosophy, our remedy stays within the context original model and does not alter the observed data in any way. Note that this discussion can not definitively say which approaches or philospophies are better, but we emphasize that a formal comparison between methods is needed.

2. **Canonical parameter confidence intervals and predictions.** Our software implementation `glmdr` currently only provides inference for mean-valued parameters (the response of a generalized linear regression model). It would be nice to be able to expand this framework to include inference about canonical parameter estimates (the regression coefficient vectors). It would also be nice to include prediction. We currently do not provide inferences for predicted values at new data points in a regression model that is fit to data that exhibits separation. Both confidence intervals for canonical parameter estimates and a sensible methodology for making valid predictions would be useful. Some of the work on confidence intervals for canonical parameter estimates has begun with Charlie Geyer's course notes.
3. **Expand `glmdr`.** This project overlaps with the above project and also includes fixing some current issues with functionality in the `glmdr` package. These current issues are discussed in this document.
4. **Propensity score estimation.** Propensity score estimation (estimating mean-values) using a logistic regression model has a lot of usefulness in causal inference. Separation of the data can render this endeavor useless, because the estimated propensity score can be 0 or 1 when separation exists which is problematic. For an example of this problem see here for some intuition behind inverse probability weighting. See here for a more formal discussion.

A brief list of methods that handle separation

We load in the following software packages:

```
rm(list = ls())

## Eck and Geyer's implementation
library(glmdr)

## Functions to accompany A. Gelman and J. Hill,
##Data Analysis Using Regression and Multilevel/Hierarchical Models,
## Cambridge University Press, 2007.
library(arm)

## logistf: Firth's Bias-Reduced Logistic Regression
library(logistf)

## brglm2: Bias Reduction in Generalized Linear Models
library(brglm2)
```

Here are some notes from the above implementations

1. glmdr: Right now only the fitting function (glmdr) and its corresponding summary methods work. The name stands for "generalized linear models done right", where "done right" means it correctly handles cases where the maximum likelihood estimate (MLE) does not exist in the conventional sense. Only does discrete generalized linear models and only those that are exponential family (because only exponential families have good theory about existence of MLE). Also does log-linear models for contingency tables and multinomial logistic regression, which it handles as conditional distributions of Poisson regression. It provides valid hypothesis tests and confidence intervals even when the MLE are "at infinity" in terms of canonical parameters or "on the boundary" in terms of mean value parameters

2. arm: Functions to accompany A. Gelman and J. Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2007. See the function 'bayesglm': Bayesian functions for generalized linear modeling with independent normal, t, or Cauchy prior distribution for the coefficients. See the accompanying paper for more details.

3. brglm2: Estimation and inference from generalized linear models based on various methods for bias reduction and maximum penalized likelihood with powers of the Jeffreys prior as penalty. The 'brglmFit' fitting method can achieve reduction of estimation bias by solving either the mean bias-reducing adjusted score equations in Firth (1993) doi:10.1093/biomet/80.1.27 and Kosmidis and Firth (2009) doi:10.1093/biomet/asp055, or the median bias-reduction adjusted score equations in Kenne et al. (2016) <arXiv:1604.04768>, or through the direct subtraction of an estimate of the bias of the maximum likelihood estimator from the maximum likelihood estimates as in Cordeiro and McCullagh (1991) http://www.jstor.org/stable/2345592. See Kosmidis et al (2019) doi:10.1007/s11222-019-09860-6 for more details. Estimation in all cases takes place via a quasi Fisher scoring algorithm, and S3 methods for the construction of of confidence intervals for the reduced-bias estimates are provided. In the special case of generalized linear models for binomial and multinomial responses (both ordinal and nominal), the adjusted score approaches return estimates with improved frequentist properties, that are also always finite, even in cases where the maximum likelihood estimates are infinite (e.g. complete and quasi-complete separation). 'brglm2' also provides pre-fit and post-fit methods for detecting separation and infinite maximum likelihood estimates in binomial response generalized linear models

4. logistf: Fit a logistic regression model using Firth's bias reduction method, equivalent to penalization of the log-likelihood by the Jeffreys prior. Confidence intervals for regression coefficients can be computed by penalized profile likelihood. Firth's method was proposed as ideal solution to the problem of separation in logistic regression. If needed, the bias reduction can be turned off such that ordinary maximum likelihood logistic regression is obtained.

Initial study: Our method works in mean-value parameter inference settings

In this study we show that our (Eck and Geyer) methodology works well when inference about mean-valued parameters (success proportion in logistic regression) is desired. Our methodology **currently** only seems to works well when the separation is not the result of one explanatory variable in the presence of others. We present two examples favoring our method and one example that illustrates where improvements are needed.

Example 1 (method works): logistic regression example

In this example, we show that our model-based solution to the complete separation problem works, and that it provides narrower confidence intervals than competing methods. The data is constructed below. This data exhibits complete separation and `glm` provides a useless error message.

```
x <- 1:30
y <- c(rep(0, 12), rep(1, 11), rep(0, 7))
out1 <- glm(y ~ x + I(x^2), family = binomial, x = TRUE)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
summary(out1)
```

```
##
## Call:
## glm(formula = y ~ x + I(x^2), family = binomial, x = TRUE)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.401e-05  -2.100e-08  -2.100e-08   2.100e-08   5.637e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1084.702 420692.854  -0.003    0.998
## x              133.047  51514.292   0.003    0.998
## I(x^2)          -3.696   1431.369  -0.003    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3.9429e+01  on 29  degrees of freedom
## Residual deviance: 1.1902e-08  on 27  degrees of freedom
## AIC: 6
##
## Number of Fisher Scoring iterations: 25
```

Our `glmdr` software can fit this model correctly and it provides the user with an informative description of what happened.

```
dat <- data.frame(y = y, x = x)
out2 <- glmdr(y ~ x + I(x^2), family = "binomial", data = dat) #, maxit = 5e3)
summary(out2)
```

```
##
## MLE exists in Barndorff-Nielsen completion
## it is completely degenerate
## the MLE says the response actually observed is the only
## possible value that could ever be observed
```

```
M <- out2$modmat

## inference for estimated parameters;
## one-sided mean value-parameter CIs when on the boundary
mus_CI <- inference(out2)
mus_CI
```

```
##          lower        upper
## 1  0.00000000 6.563497e-12
## 2  0.00000000 1.915859e-10
## 3  0.00000000 4.570606e-09
## 4  0.00000000 8.919127e-08
## 5  0.00000000 1.425473e-06
## 6  0.00000000 1.869697e-05
## 7  0.00000000 2.019500e-04
## 8  0.00000000 1.806200e-03
## 9  0.00000000 1.345421e-02
## 10 0.00000000 8.242264e-02
## 11 0.00000000 3.741233e-01
## 12 0.00000000 9.482943e-01
## 13 0.05729254 1.000000e+00
## 14 0.65501231 1.000000e+00
## 15 0.86723201 1.000000e+00
## 16 0.92920573 1.000000e+00
## 17 0.95066373 1.000000e+00
## 18 0.95616871 1.000000e+00
## 19 0.95066368 1.000000e+00
## 20 0.92920575 1.000000e+00
## 21 0.86723190 1.000000e+00
## 22 0.65501179 1.000000e+00
## 23 0.05241039 1.000000e+00
## 24 0.00000000 9.478845e-01
## 25 0.00000000 3.741229e-01
## 26 0.00000000 8.242262e-02
## 27 0.00000000 1.345423e-02
## 28 0.00000000 1.806203e-03
## 29 0.00000000 2.019507e-04
## 30 0.00000000 1.869705e-05
```

We now fit using the defaults of the `bayesglm` function `arm` package.

```
## fit using bayesglm
out3 <- bayesglm (y ~ x + I(x^2), family=binomial(link="logit"))
display (out3)
```

```
## bayesglm(formula = y ~ x + I(x^2), family = binomial(link = "logit"))
##             coef.est coef.se
## (Intercept) -23.99     8.16
## x             2.99     0.97
## I(x^2)       -0.08     0.03
## ---
## n = 30, k = 3
## residual deviance = 7.4, null deviance = 39.4 (difference = 32.0)
```

Our method produces smaller confidence intervals than the defaults in `bayesglm`.

```
canon_out3 <- predict(out3, type = "link", se.fit = TRUE)
lwr_out3 <- canon_out3$fit - 1.96 * canon_out3$se.fit
upr_out3 <- canon_out3$fit + 1.96 * canon_out3$se.fit
mus_CI_out3 <- cbind(invlogit(lwr_out3), invlogit(upr_out3))

## total length of our CIs
sum(as.matrix(mus_CI) %*% c(-1,1))
```

```
## [1] 5.970135
```

```
## total length of bayesglm CIs
sum(as.matrix(mus_CI_out3) %*% c(-1,1))
```

```
## [1] 11.11135
```

–>

We now fit using various defaults of the `brglmFit` fitting method. See their help page for more details.

```
## fit using brglmFit function;
## summary calls are ignored
out4 <- glm(y ~ x + I(x^2), family=binomial(link="logit"))
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
out4_AS_mixed <- update(out4, method = "brglmFit", type = "AS_mixed")
```

```
## Warning: brglmFit: fitted probabilities numerically 0 or 1 occurred
```

```
out4_AS_mean <- update(out4, method = "brglmFit", type = "AS_mean")
```

```
## Warning: brglmFit: fitted probabilities numerically 0 or 1 occurred
```

```
out4_AS_median <- update(out4, method = "brglmFit", type = "AS_median")
```

```
## Warning: brglmFit: fitted probabilities numerically 0 or 1 occurred
```

```
out4_AS_MPL_Jeffreys <- update(out4, method = "brglmFit", type = "MPL_Jeffreys")
```

```
## Warning: brglmFit: fitted probabilities numerically 0 or 1 occurred
```

```
out4_AS_correction <- update(out4, method = "brglmFit", type = "correction")
```

```
## Warning: brglmFit: fitted probabilities numerically 0 or 1 occurred
```

Our method produces smaller confidence intervals than any of the `brglmFit` fitting methods.

```
## AS_mixed fit: the mixed bias-reducing score adjustments
## in Kosmidis et al (2019)
canon_out4_AS_mixed <- predict(out4_AS_mixed, type = "link", se.fit = TRUE)
lwr_out4_AS_mixed <- canon_out4_AS_mixed$fit - 1.96 * canon_out4_AS_mixed$se.fit
upr_out4_AS_mixed <- canon_out4_AS_mixed$fit + 1.96 * canon_out4_AS_mixed$se.fit
mus_CI_out4_AS_mixed <-
  cbind(invlogit(lwr_out4_AS_mixed), invlogit(upr_out4_AS_mixed))
sum(as.matrix(mus_CI) %*% c(-1,1))
```

```
## [1] 5.970135
```

```
sum(as.matrix(mus_CI_out4_AS_mixed) %*% c(-1,1))
```

```
## [1] 15.87614
```

```
## AS_mean fit: the mean bias-reducing score adjustments
## in Firth, 1993 and Kosmidis & Firth, 2009.
canon_out4_AS_mean <- predict(out4_AS_mean, type = "link", se.fit = TRUE)
lwr_out4_AS_mean <- canon_out4_AS_mean$fit - 1.96 * canon_out4_AS_mean$se.fit
upr_out4_AS_mean <- canon_out4_AS_mean$fit + 1.96 * canon_out4_AS_mean$se.fit
mus_CI_out4_AS_mean <-
  cbind(invlogit(lwr_out4_AS_mean), invlogit(upr_out4_AS_mean))
sum(as.matrix(mus_CI) %*% c(-1,1))
```

```
## [1] 5.970135
```

```
sum(as.matrix(mus_CI_out4_AS_mean) %*% c(-1,1))
```

```
## [1] 15.87614
```

```
## AS_median fit: the median-bias reducing score adjustments
## in Kenne Pagui et al. (2017)
canon_out4_AS_median <- predict(out4_AS_median, type = "link", se.fit = TRUE)
lwr_out4_AS_median <- canon_out4_AS_median$fit - 1.96 * canon_out4_AS_median$se.fit
upr_out4_AS_median <- canon_out4_AS_median$fit + 1.96 * canon_out4_AS_median$se.fit
mus_CI_out4_AS_median <-
  cbind(invlogit(lwr_out4_AS_median), invlogit(upr_out4_AS_median))
sum(as.matrix(mus_CI) %*% c(-1,1))
```

```
## [1] 5.970135
```

```
sum(as.matrix(mus_CI_out4_AS_median) %*% c(-1,1))
```

```
## [1] 26.35001
```

```
## AS_MPL_Jeffreys fit: maximum penalized likelihood
## with powers of the Jeffreys prior as penalty.
canon_out4_AS_MPL_Jeffreys <- predict(out4_AS_MPL_Jeffreys, type = "link", se.fit = TRUE)
lwr_out4_AS_MPL_Jeffreys <- canon_out4_AS_MPL_Jeffreys$fit - 1.96 * canon_out4_AS_MPL_Jeffr
upr_out4_AS_MPL_Jeffreys <- canon_out4_AS_MPL_Jeffreys$fit + 1.96 * canon_out4_AS_MPL_Jeffr
mus_CI_out4_AS_MPL_Jeffreys <-
  cbind(invlogit(lwr_out4_AS_MPL_Jeffreys), invlogit(upr_out4_AS_MPL_Jeffreys))
sum(as.matrix(mus_CI) %*% c(-1,1))
```

```
## [1] 5.970135
```

```
sum(as.matrix(mus_CI_out4_AS_MPL_Jeffreys) %*% c(-1,1))
```

```
## [1] 15.87614
```

```
## AS_correction fit: maximum penalized likelihood
## with powers of the Jeffreys prior as penalty.
## does not fit
#canon_out4_AS_correction <- predict(out4_AS_correction, type = "link", se.fit = TRUE)
#lwr_out4_AS_correction <- canon_out4_AS_correction$fit - 1.96 * canon_out4_AS_correction$s
#upr_out4_AS_correction <- canon_out4_AS_correction$fit + 1.96 * canon_out4_AS_correction$s
#mus_CI_out4_AS_correction <-
#  cbind(invlogit(lwr_out4_AS_correction), invlogit(upr_out4_AS_correction))
#sum(as.matrix(mus_CI) %*% c(-1,1))
#sum(as.matrix(mus_CI_out4_AS_correction) %*% c(-1,1))
```

We summarize these findings in the table below:

```
cbind(c("glmdr","Bayesglm","AS_mixed","AS_mean","AS_median","AS_MPL_Jeffreys"), round(c(sum
  sum(as.matrix(mus_CI_out3) %*% c(-1,1)),
  sum(as.matrix(mus_CI_out4_AS_mixed) %*% c(-1,1)),
  sum(as.matrix(mus_CI_out4_AS_mean) %*% c(-1,1)),
  sum(as.matrix(mus_CI_out4_AS_median) %*% c(-1,1)),
  sum(as.matrix(mus_CI_out4_AS_MPL_Jeffreys) %*% c(-1,1))), 3))
```

```
##      [,1]              [,2]
## [1,] "glmdr"           "5.97"
## [2,] "Bayesglm"        "11.111"
## [3,] "AS_mixed"        "15.876"
## [4,] "AS_mean"         "15.876"
## [5,] "AS_median"       "26.35"
## [6,] "AS_MPL_Jeffreys" "15.876"
```

We now fit using various defaults of the `logistf` function.

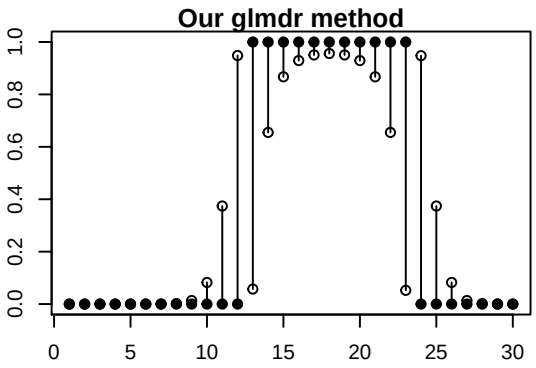
```
## fit using Firth's bias-reduced logistic regression
out5 <- logistf(y ~ x + I(x^2), family=binomial(link="logit"))
#summary(out5)
#M %*% out5$coefficients
```

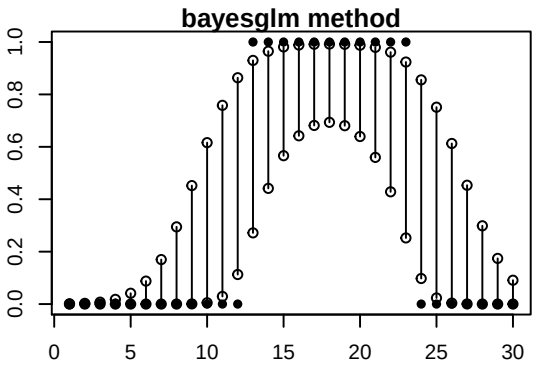
The functionality in `logistf` does not integrate easily to the mean-value parameterization. Therefore, we do not consider `logistf` past this point.

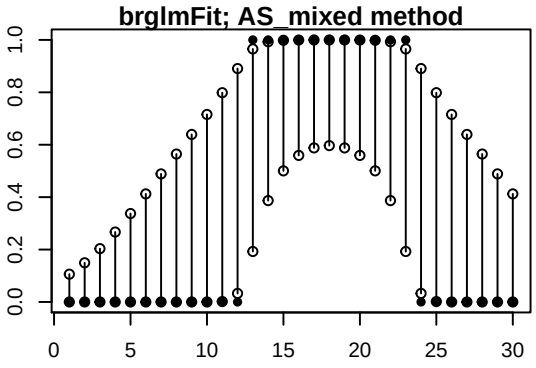
We now visualize the different confidence intervals.

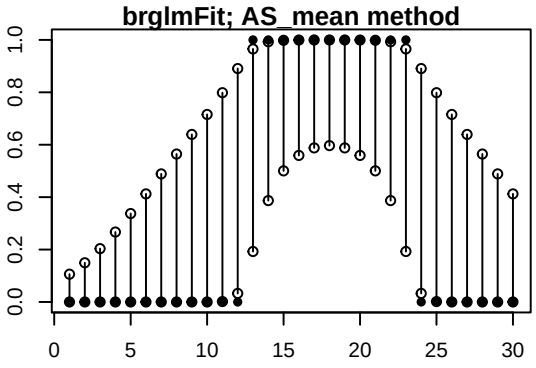
```
par(mfrow = c(3,2), mar = c(5, 4, 1, 1) + 0.1)
plot(x, y, ylim = c(0,1), pch = 16, main = "Our glmdr method",
     ylab = "", xlab = "")
points(x, mus_CI[, 1])
points(x, mus_CI[, 2])
segments(x, mus_CI[, 1], x, mus_CI[, 2])

plot(x, y, ylim = c(0,1), pch = 16, main = "bayesglm method",
     ylab = "", xlab = "")
points(x, mus_CI_out3[, 1])
points(x, mus_CI_out3[, 2])
segments(x, mus_CI_out3[, 1], x, mus_CI_out3[, 2])

plot(x, y, ylim = c(0,1), pch = 16, main = "brglmFit; AS_mixed method",
     ylab = "", xlab = "")
points(x, mus_CI_out4_AS_mean[, 1])
points(x, mus_CI_out4_AS_mean[, 2])
segments(x, mus_CI_out4_AS_mean[, 1], x, mus_CI_out4_AS_mean[, 2])

plot(x, y, ylim = c(0,1), pch = 16, main = "brglmFit; AS_mean method",
     ylab = "", xlab = "")
points(x, mus_CI_out4_AS_mixed[, 1])
points(x, mus_CI_out4_AS_mixed[, 2])
segments(x, mus_CI_out4_AS_mixed[, 1], x, mus_CI_out4_AS_mixed[, 2])

plot(x, y, ylim = c(0,1), pch = 16, main = "brglmFit; AS_median method",
     ylab = "", xlab = "")
points(x, mus_CI_out4_AS_median[, 1])
points(x, mus_CI_out4_AS_median[, 2])
```

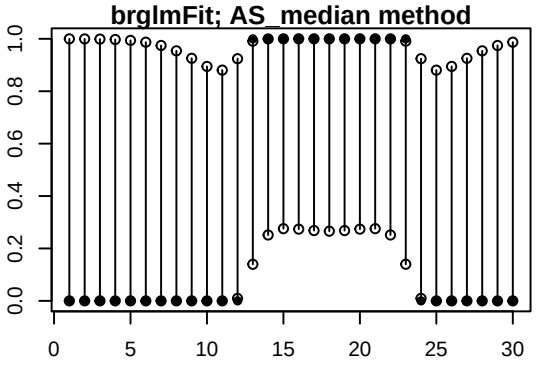
```r
segments(x, mus_CI_out4_AS_median[, 1], x, mus_CI_out4_AS_median[, 2])

plot(x, y, ylim = c(0,1), pch = 16, main = "brglmFit; MPL_Jeffreys method",
     ylab = "", xlab = "")
points(x, mus_CI_out4_AS_MPL_Jeffreys[, 1])
points(x, mus_CI_out4_AS_MPL_Jeffreys[, 2])
segments(x, mus_CI_out4_AS_MPL_Jeffreys[, 1],
         x, mus_CI_out4_AS_MPL_Jeffreys[, 2])

#plot(x, y, ylim = c(0,1), pch = 16, ylab = "", xlab = "")
#points(x, mus_CI_out4_AS_correction[, 1])
#points(x, mus_CI_out4_AS_correction[, 2])
#segments(x, mus_CI_out4_AS_correction[, 1],
#         x, mus_CI_out4_AS_correction[, 2])
```

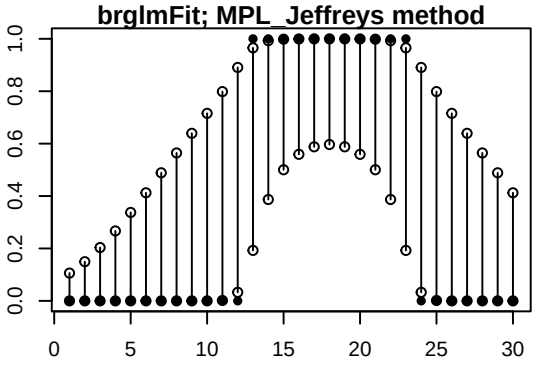

Our glmdr method
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


bayesglm method
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


brglmFit; AS_mixed method
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


brglmFit; AS_mean method
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


brglmFit; AS_median method
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


brglmFit; MPL_Jeffreys method
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30

Example 2 (method works): Poisson regression example

Our method also works for Poisson regression. In this example, the regression model is not completely degenerate.

```
## Poisson regression example from exponential family paper
data(catrec)
gout <- glm(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
              family = "poisson", data = catrec)
gout2 <- glmdr(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
              family = "poisson", data = catrec)
mus_CI_catrec <- inference(gout2)
mus_CI_catrec
```

```
##     lower      upper
## 1       0 0.28630976
## 9       0 0.14082947
## 20      0 0.21996699
## 28      0 0.42095570
## 33      0 0.08946242
## 41      0 0.09376644
## 52      0 0.19302341
## 60      0 0.28869770
## 65      0 0.10631113
## 73      0 0.11415034
## 84      0 0.09128766
## 92      0 0.26461098
## 97      0 0.06669488
## 105     0 0.15477613
## 116     0 0.14096916
## 124     0 0.32392016
```

We can see which indices of the response vector are constrained to be on the boundary of their support ($\hat{y} = 0$).

```
linearity_catrec <- gout2$linearity
which(!linearity_catrec)
```

```
##   1   9  20  28  33  41  52  60  65  73  84  92  97 105 116 124
##   1   9  20  28  33  41  52  60  65  73  84  92  97 105 116 124
```

We can see that our methodology produces smaller confidence intervals for the mean-value parameters than the competing methods.

```
gout3 <- bayesglm (y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
                   family = poisson(link="log"), data = catrec)
#probs_out3 <- predict(out3, type = "response", se.fit = TRUE)

canon_gout3 <- predict(gout3, type = "link", se.fit = TRUE)
lwr_gout3 <- canon_gout3$fit - 1.96 * canon_gout3$se.fit
upr_gout3 <- canon_gout3$fit + 1.96 * canon_gout3$se.fit
mus_CI_gout3 <- cbind(exp(lwr_gout3), exp(upr_gout3))[!linearity_catrec, ]

sum(as.matrix(mus_CI_catrec) %*% c(-1,1))
```

```
## [1] 2.995732
```

```
sum(as.matrix(mus_CI_gout3) %*% c(-1,1))
```

```
## [1] 3432.42
```

```r
## fit using brglmFit function
gout4 <- glm(y ~ (v1 + v2 + v3 + v4 + v5 + v6 + v7)^3,
             family = poisson(link="log"), data = catrec)
gout4_AS_mixed <- update(gout4, method = "brglmFit", type = "AS_mixed")
gout4_AS_mean <- update(gout4, method = "brglmFit", type = "AS_mean")
gout4_AS_median <- update(gout4, method = "brglmFit", type = "AS_median")
gout4_AS_MPL_Jeffreys <- update(gout4, method = "brglmFit",
                                type = "MPL_Jeffreys")

## AS_mixed fit: the mixed bias-reducing score adjustments
## in Kosmidis et al (2019)
canon_gout4_AS_mixed <- predict(gout4_AS_mixed, type = "link", se.fit = TRUE)
lwr_gout4_AS_mixed <- canon_gout4_AS_mixed$fit - 1.96 * canon_gout4_AS_mixed$se.fit
upr_gout4_AS_mixed <- canon_gout4_AS_mixed$fit + 1.96 * canon_gout4_AS_mixed$se.fit
mus_CI_gout4_AS_mixed <-
  cbind(invlogit(lwr_gout4_AS_mixed), invlogit(upr_gout4_AS_mixed))
sum(as.matrix(mus_CI_catrec) %*% c(-1,1))
```

```
## [1] 2.995732
```

```r
sum(as.matrix(mus_CI_gout4_AS_mixed[!linearity_catrec, ]) %*% c(-1,1))
```

```
## [1] 5.57936
```

```r
## AS_mean fit: the mean bias-reducing score adjustments
## in Firth, 1993 and Kosmidis & Firth, 2009.
canon_gout4_AS_mean <- predict(gout4_AS_mean, type = "link", se.fit = TRUE)
lwr_gout4_AS_mean <- canon_gout4_AS_mean$fit - 1.96 * canon_gout4_AS_mean$se.fit
upr_gout4_AS_mean <- canon_gout4_AS_mean$fit + 1.96 * canon_gout4_AS_mean$se.fit
mus_CI_gout4_AS_mean <-
  cbind(invlogit(lwr_gout4_AS_mean), invlogit(upr_gout4_AS_mean))
sum(as.matrix(mus_CI_catrec) %*% c(-1,1))
```

```
## [1] 2.995732
```

```r
sum(as.matrix(mus_CI_gout4_AS_mean[!linearity_catrec, ]) %*% c(-1,1))
```

```
## [1] 5.57936
```

```r
## AS_median fit: the median-bias reducing score adjustments
## in Kenne Pagui et al. (2017)
canon_gout4_AS_median <- predict(gout4_AS_median, type = "link", se.fit = TRUE)
lwr_gout4_AS_median <- canon_gout4_AS_median$fit - 1.96 * canon_gout4_AS_median$se.fit
upr_gout4_AS_median <- canon_gout4_AS_median$fit + 1.96 * canon_gout4_AS_median$se.fit
mus_CI_gout4_AS_median <-
  cbind(invlogit(lwr_gout4_AS_median), invlogit(upr_gout4_AS_median))
sum(as.matrix(mus_CI_catrec) %*% c(-1,1))
```

```
## [1] 2.995732
```

```r
sum(as.matrix(mus_CI_gout4_AS_median[!linearity_catrec, ]) %*% c(-1,1))
```

```
## [1] 7.848909
```

```r
## AS_MPL_Jeffreys fit: maximum penalized likelihood
## with powers of the Jeffreys prior as penalty.
canon_gout4_AS_MPL_Jeffreys <- predict(gout4_AS_MPL_Jeffreys, type = "link", se.fit = TRUE)
lwr_gout4_AS_MPL_Jeffreys <- canon_gout4_AS_MPL_Jeffreys$fit - 1.96 * canon_gout4_AS_MPL_Je
```

```
upr_gout4_AS_MPL_Jeffreys <- canon_gout4_AS_MPL_Jeffreys$fit + 1.96 * canon_gout4_AS_MPL_Je
mus_CI_gout4_AS_MPL_Jeffreys <-
  cbind(invlogit(lwr_gout4_AS_MPL_Jeffreys), invlogit(upr_gout4_AS_MPL_Jeffreys))
sum(as.matrix(mus_CI_catrec) %*% c(-1,1))
```

```
## [1] 2.995732
```

```
sum(as.matrix(mus_CI_gout4_AS_MPL_Jeffreys[!linearity_catrec, ]) %*% c(-1,1))
```

```
## [1] 5.57936
```

We summarize these findings in the table below:

```
cbind(c("glmdr","Bayesglm","AS_mixed","AS_mean","AS_median","AS_MPL_Jeffreys"), round(c(sum
  sum(as.matrix(mus_CI_gout3) %*% c(-1,1)),
  sum(as.matrix(mus_CI_gout4_AS_mixed) %*% c(-1,1)),
  sum(as.matrix(mus_CI_gout4_AS_mean) %*% c(-1,1)),
  sum(as.matrix(mus_CI_gout4_AS_median) %*% c(-1,1)),
  sum(as.matrix(mus_CI_gout4_AS_MPL_Jeffreys) %*% c(-1,1))), 3))
```

```
##      [,1]              [,2]
## [1,] "glmdr"           "2.996"
## [2,] "Bayesglm"        "3432.42"
## [3,] "AS_mixed"        "29.093"
## [4,] "AS_mean"         "29.093"
## [5,] "AS_median"       "31.258"
## [6,] "AS_MPL_Jeffreys" "29.093"
```

Example 3 (method does not work)

This example shows a setting where `glmdr` does not seem to work well. In this example we see that the `inference` function in the `glmdr` package estimates mean-value parameters to be exactly on the boundary (the endpoints of the 95% CI are essentially the same).

```
data(endometrial)
m_endo <- glm(HG ~., family = binomial(link = "logit"),
              data = endometrial)
m_endo_glmdr <- glmdr(HG ~., family = "binomial", data = endometrial)
mus_CI_endo <- inference(m_endo_glmdr)
mus_CI_endo
```

```
##        lower upper
## 22 0.9999999     1
## 23 1.0000000     1
## 24 1.0000000     1
## 25 1.0000000     1
## 26 1.0000000     1
## 48 1.0000000     1
## 49 0.9999997     1
## 50 1.0000000     1
## 51 1.0000000     1
## 71 1.0000000     1
## 75 1.0000000     1
## 76 1.0000000     1
## 78 1.0000000     1
```

This does matter even when we specify α to be very small. The following is worse, now there are NAs.

```
mus_CI_endo2 <- inference(m_endo_glmdr, alpha = 0.0001)
mus_CI_endo2
```

```
##        lower upper
## 22 0.9999997     1
## 23        NA     1
## 24 1.0000000     1
## 25 0.9999998     1
## 26 1.0000000     1
## 48        NA     1
## 49        NA     1
## 50 1.0000000     1
## 51 1.0000000     1
## 71        NA     1
## 75 1.0000000     1
## 76 1.0000000     1
## 78        NA     1
```

The `inference` function does not appear to be working well in this example. I have looked into this example and think that the problem is due to our implementation in the `inference` function and not the `glmdr` fitting function. At the present moment, the problem seems to be caused by the existence of a unit null eigen vector in the Fisher Information matrix.

```
nulls <- m_endo_glmdr$nulls
round(nulls, 5)
```

```
##      [,1]
## [1,]    0
```

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

This means that the separation is an artifact of separation in one of the predictor variables. The follow shows that the response is observed to be 1 whenever the predictor `NV` is 1.

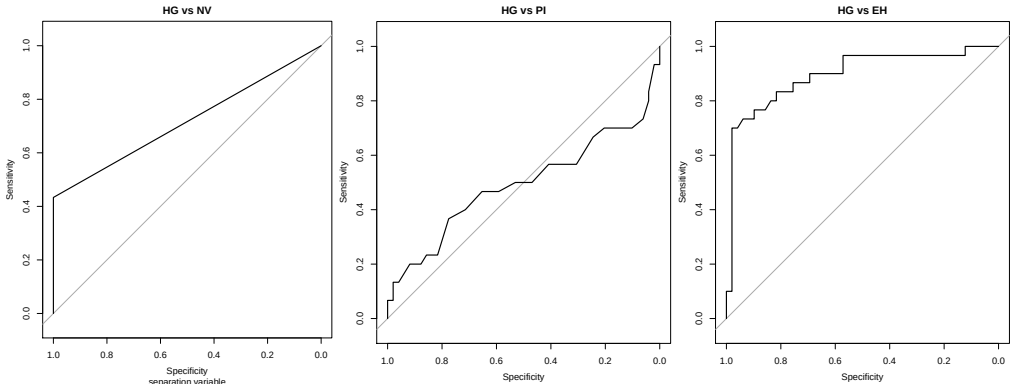
```
endometrial[!m_endo_glmdr$linearity, ]
```

```
##    NV PI   EH HG
## 22  1 38 0.97  1
## 23  1 22 1.14  1
## 24  1  7 0.88  1
## 25  1 25 0.91  1
## 26  1 15 0.58  1
## 48  1 22 1.44  1
## 49  1 40 1.18  1
## 50  1  5 0.93  1
## 51  1  0 1.17  1
## 71  1 49 0.27  1
## 75  1 11 1.01  1
## 76  1 21 0.98  1
## 78  1 19 1.02  1
```

Our methodology does not seem to work when this is the case.

ROC curves

```
library(pROC)
foo <- roc(HG ~., data = endometrial)
```



Deviance

We investigate the deviance in models with complete separation. The deviance statistic is

$$D = -2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{p}_i}\right) + (1-y_i)\log\left(\frac{1-y_i}{1-\hat{p}_i}\right)\right]$$

```
x <- 1:30
y <- c(rep(0, 12), rep(1, 11), rep(0, 7))
out1 <- glm(y ~ x + I(x^2), family = binomial, x = TRUE)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
summary(out1)
```

```
##
## Call:
## glm(formula = y ~ x + I(x^2), family = binomial, x = TRUE)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.401e-05  -2.100e-08  -2.100e-08   2.100e-08   5.637e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1084.702 420692.854  -0.003    0.998
## x              133.047  51514.292   0.003    0.998
## I(x^2)          -3.696   1431.369  -0.003    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3.9429e+01  on 29  degrees of freedom
## Residual deviance: 1.1902e-08  on 27  degrees of freedom
## AIC: 6
##
## Number of Fisher Scoring iterations: 25
```

```
names(out1)
```

```
##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "x"                 "call"              "formula"
## [25] "terms"             "data"              "offset"
## [28] "control"           "method"            "contrasts"
## [31] "xlevels"
```

```
out1$null.deviance
```

```
## [1] 39.42947
```

```
out1$deviance
```

```
## [1] 1.190244e-08
```

```
p1 <- predict(out1, type = "response")
```

```
dat <- data.frame(y = y, x = x)
out2 <- glmdr(y ~ x + I(x^2), family = "binomial", data = dat) #, maxit = 5e3)
summary(out2)
```

```
##
## MLE exists in Barndorff-Nielsen completion
## it is completely degenerate
## the MLE says the response actually observed is the only
## possible value that could ever be observed
```

```
out2$linearity
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE
```