# Technical Appendix for "Dealing with Separation in Logistic Regression Models"

*Carlisle Rainey*

*March 2, 2016*

## Proof of Theorem 1

Recall Theorem 1:

> For a monotonic likelihood $p(y|\beta)$ increasing [decreasing] in $\beta_s$, proper prior distribution $p(\beta|\sigma)$, and large positive [negative] $\beta_s$, the posterior distribution of $\beta_s$ is proportional to the prior distribution for $\beta_s$, so that $p(\beta_s|y) \propto p(\beta_s|\sigma)$. More formally, $\lim\limits_{\substack{\beta_s \to \infty \\ [-\infty]}} \dfrac{p(\beta_s|y)}{p(\beta_s|\sigma)} = k$, for postive constant $k$.

*Proof.* Due to separation, $p(y|\beta)$ is monotonic increasing in $\beta_s$ to a limit $\mathscr{L}$, so that $\lim\limits_{\beta_s \to \infty} p(y|\beta_s) = \mathscr{L}$. By Bayes' rule,

$$p(\beta|y) = \frac{p(y|\beta)p(\beta|\sigma)}{\int\limits_{-\infty}^{\infty} p(y|\beta)p(\beta|\sigma)d\beta} = \frac{p(y|\beta)p(\beta|\sigma)}{\underbrace{p(y|\sigma)}_{\text{constant w.r.t. } \beta}}.$$

Integrating out the other parameters $\beta_{-s} = \langle \beta_{cons}, \beta_1, \beta_2, ..., \beta_k \rangle$ to obtain the posterior distribution of $\beta_s$,

$$p(\beta_s|y) = \frac{\int\limits_{-\infty}^{\infty} p(y|\beta)p(\beta|\sigma)d\beta_{-s}}{p(y|\sigma)}, \tag{1}$$

and the prior distribution of $\beta_s$,

$$p(\beta_s|\sigma) = \int\limits_{-\infty}^{\infty} p(\beta|\sigma)d\beta_{-s}.$$

Notice that $p(\beta_s|y) \propto p(\beta_s|\sigma)$ iff $\dfrac{p(\beta_s|y)}{p(\beta_s|\sigma)} = k$, where the constant $k \neq 0$. Thus, Theorem **??** implies that

$$\lim_{\beta_s \to \infty} \frac{p(\beta_s|y)}{p(\beta_s|\sigma)} = k$$

Substituting in Equation 1,

$$\lim_{\beta_s \to \infty} \frac{\frac{\int\limits_{-\infty}^{\infty} p(y|\beta)p(\beta|\sigma)d\beta_{-s}}{p(y|\sigma)}}{p(\beta_s|\sigma)} = k.$$

Multiplying both sides by $p(y|\sigma)$, which is constant with respect to $\beta$,

$$\lim_{\beta_s \to \infty} \frac{\int\limits_{-\infty}^{\infty} p(y|\beta)p(\beta|\sigma)d\beta_{-s}}{p(\beta_s|\sigma)} = kp(y|\sigma).$$

Setting $\int\limits_{-\infty}^{\infty} p(y|\beta)p(\beta|\sigma)d\beta_{-s} = p(y|\beta_s)p(\beta_s|\sigma)$,

$$\lim_{\beta_s \to \infty} \frac{p(y|\beta_s)p(\beta_s|\sigma)}{p(\beta_s|\sigma)} = kp(y|\sigma).$$

Canceling $p(\beta_s|\sigma)$ in the numerator and denominator,

$$\lim_{\beta_s \to \infty} p(y|\beta_s) = kp(y|\sigma).$$

Recalling that $\lim\limits_{\beta_s \to \infty} p(y|\beta) = \underline{\mathscr{L}}$ and substituting,

$$\underline{\mathscr{L}} = kp(y|\sigma),$$

which implies that $k = \dfrac{p(y|\sigma)}{\underline{\mathscr{L}}}$, which is a positive constant. $\square$

## Question and Answers

### Does the process partial prior distribution, which depends on the data, invalidate or bias the estimates or hypothesis tests?

In short, no.

Let me start by clarifying a potential confusion. When I write "the researcher might assess the reasonableness of the prior distribution by examining the prior distribution and asking herself whether the prior and model produce a distribution for the quantities of interest that matches her prior information," I do not mean that the researcher should compute the (post-data) posterior. Instead, I mean that the researcher should simply transform the (pre-data) prior into the quantities of interest to check whether the prior makes substantive sense. *I would consider it inappropriate to choose a prior based on the reasonableness of the posterior.*

The process does involve initially fitting a model with ML. However, this initial fit only serves one purpose: to identify the region of the (multivariate) prior distribution that really matters, which simplifies the process and allows the researcher to choose an appropriate prior for that region. Instead, I view it as almost purely Bayesian (i.e., computing a posterior based on data with informative priors), except that rather than choose a full prior distribution (almost always impossible for multivariate models), the researcher places uninformative priors on the coefficients for the non-separating explanatory variables and an informative prior on the coefficient for the separating explanatory variable.

Things then get tricky because political scientists usually think of effects in terms of predicted probabilities, first differences, or risk ratios—not in terms of the coefficients themselves. Of course, these quantities depend on the values of the coefficients for the non-separating variables. My suggestion is to use ML as a quick method to assign values to these coefficients. With those coefficients fixed as a reasonable value, it is then easy to think about the prior distribution for the separating coefficient in terms of an arbitrary quantity of interest.

The only important choice in the problem is to choose a scale parameter for the prior distribution for the coefficient of the separating variable. (Unbounded, symmetric, mean-zero priors should almost always work fine.) This choice is essentially asking: How much pooling toward zero is appropriate? We know we need some pooling—infinite estimates are not plausible. The question is how much.

Once the researcher chooses the prior (i.e., chooses the scale), she fits the model with MCMC just once and uses the MCMC simulations to perform the inference. Except for the initial ML fit to determine the important region of the prior distribution, the researcher fits the model just once. I am comfortable performing inference in the usual way, without any correction. Though the researcher is using the data twice, the first usage only points the researcher to the important region of the prior distribution (i.e., other coefficients near their MLEs).

Let me emphasize again that I would oppose evaluating the quality of the prior in terms of the reasonableness of the posterior—I would view the inferences from this approach as biased. In my view, the posterior should be computed just once (except when demonstrating the robustness of the conclusions to a range of prior distributions).

## Does Clarify adequately approximate the posterior distribution under separation?

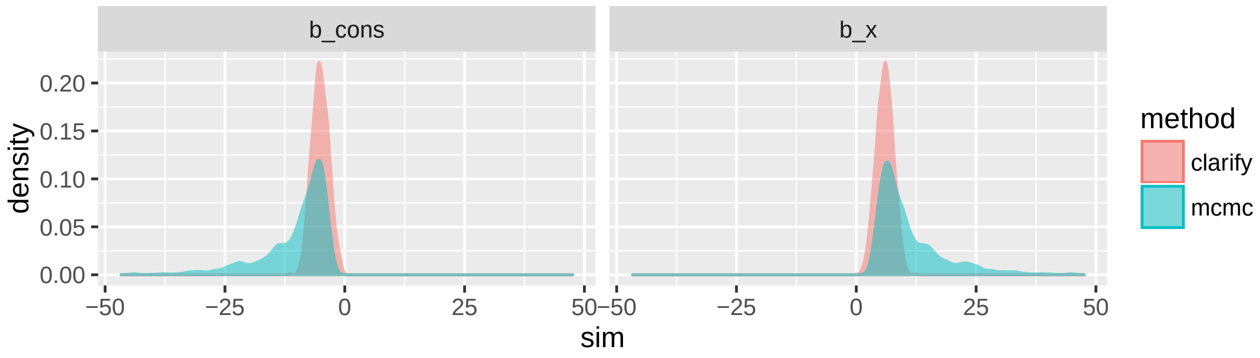
In short, no.

In the paper, I note that Clarify provides a "poor" approximate to the posterior distribution under separation. This is unusual. Clarify's approach *usually* produces confidence intervals that closely correspond to the exact posterior computed via MCMC. However, in the case of separation, the approximation of Clarify's simulation procedure to MCMC simulation is indeed quite poor.

I have included an example below to illustrate that separation "breaks" Clarify, at least in terms Clarify's ability to approximate the posterior distribution. In the figure below, notice that the *correct* posterior calculated by MCMC suggests that much larger values of the coefficients are plausible than the Clarify approximation suggests. (Though, as R1 notes, Clarify and the bootstrap produce similar confidence intervals.)

```r
# load packages
library(MASS)
library(arm)
library(rstanarm)
library(dplyr)
library(tidyr)
library(magrittr)
library(ggplot2)

# set seed
set.seed(208579)

# create data
n <- 100  # sample size
x <- rbinom(n, size = 1, prob = 0.5)  # single explanatory variable
y <- rbinom(n, size = 1, prob = plogis(-1 + 2*x))  # create overlapping outcome variable
y[x == 0] <- 0  # create quasi-complete separation

# approximate posterior using Clarify-like simulation
pml_fit <- bayesglm(y ~ x, family = binomial)
mu_hat <- coef(pml_fit)
Sigma_hat <- vcov(pml_fit)
clarify_sims <- mvrnorm(n = 1000, mu = mu_hat, Sigma = Sigma_hat) %>%
  data.frame("clarify")

# calculate posterior exactly with MCMC
mcmc_fit  <- stan_glm(y ~ x, family = binomial, prior = cauchy())
mcmc_sims <- as.matrix(mcmc_fit) %>%
  data.frame("mcmc")

# combine simulation into single data.frame for ploting
```

```r
names(clarify_sims) <- names(mcmc_sims) <- c("b_cons", "b_x", "method")
df <- rbind(clarify_sims, mcmc_sims) %>%
  gather(parameter, sim, -method)

# plot
ggplot(df, aes(x = sim, color = method, fill = method)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ parameter)
```



### Do the estimates of the coefficients for the non-separating explanatory variables depend on the prior?

In short, not much.

The ML estimates of the coefficients for the non-separating explanatory variables are “roughly correct.” However, the presence of separation makes a precise statement of the meaning difficult for two reasons. First, separation itself is a post-data description. Secondly, if the the separating variable is correlated with the other variables, then the estimator is somewhat biased. We know (or assume) that separation produces an overestimate of the absolute value $\beta_s$ . If we have an over-estimate of $\beta_s$ and $s$ is correlated with $x$, then the estimate of $\beta_x$ will be biased. However, my assertion is that this bias is not too large and that all we need is a rough estimate.

The simulation below verifies intuition that the bias is not large, but also confirms that the magnitude of the bias is substantively negligible, even when separation is “severe” (true coefficient is large) and the correlation between the separating and non-separating explanatory variable is large. That is, when (1) using a much more informative prior than usual (i.e., scale parameter of 1 rather than 2.5) (2) with variables more correlated than usual (rho = 0.84) and (3) much larger effects than researchers typically expect ($\beta_x = -11$), the bias is quite small. Even in this “worst case” scenario, the average estimate is about 0.38 while the true estimate is about 0.5. For more typical correlations (i.e., about 0.48), the average estimate of $\beta_x$ never drops below about 0.44. For more typical effect sizes (i.e., about $\beta_s = -6$), the average estimate of $\beta_x$ is always close to 0.5.

```r
# load packages
library(arm)

# set seed
set.seed(208579)

# simulation parameters
n <- 500 # sample size
b_cons <- c(-1, 2, 4)  # intercept
b_s <- c(-6, -9, -11)  # coefficient for separating variable
```

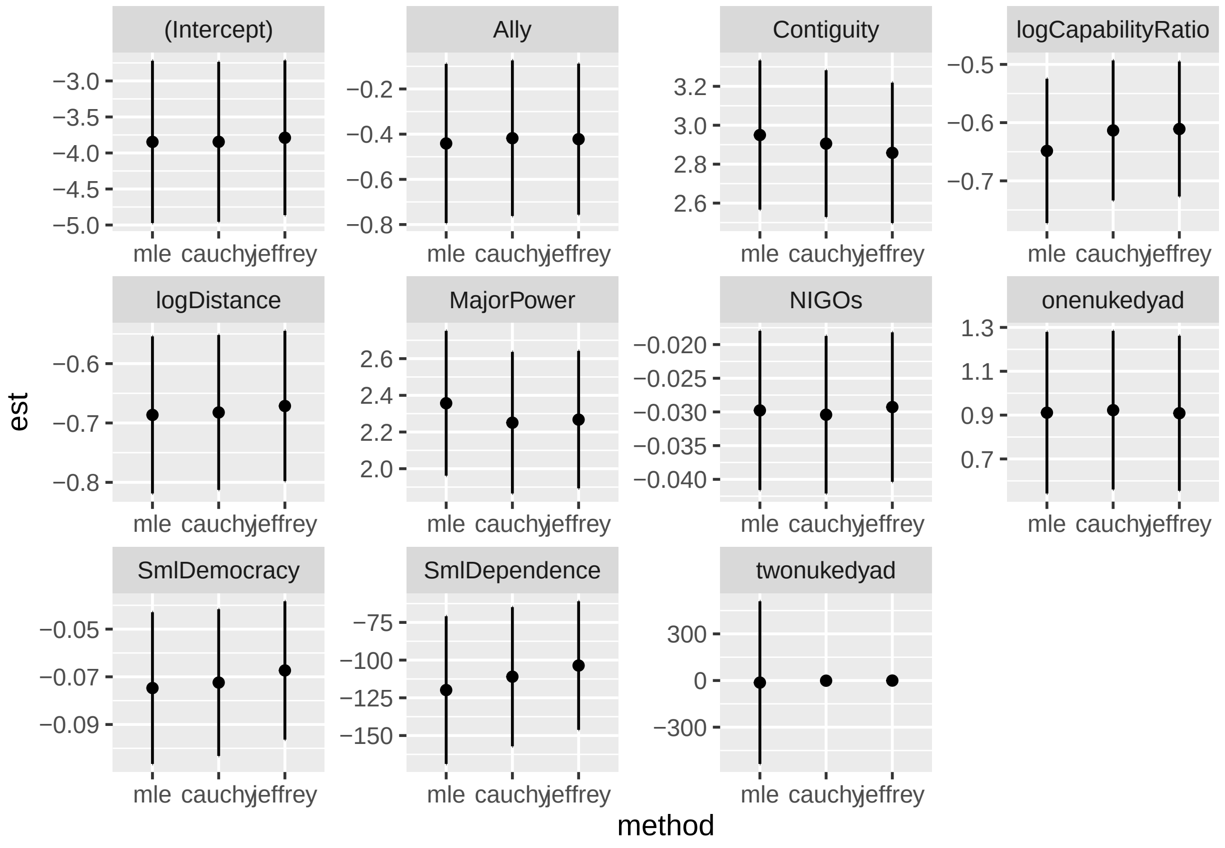
```r
b_x <- 0.5  # coefficient for other variable
rho_star <- c(1, 2, 3)  # parameter controlling the correlation between x and s
                        # note: (1 ~> 0.48, 2 ~> 0.74, 3 ~> 0.84)
prior_scale <- c(10, 2.5, 1)

# create explanatory variabls
s <- c(rep(0, n/2), rep(1, n/2)) # separating variable

# monte carlo simulation
n_sims <- 1000  # number of monte carlo simulations
hat_b_x <- array(NA, c(n_sims, length(rho_star), length(b_s), length(prior_scale)),  # holder for coeff
                 dimnames = list(NULL,
                                 paste("rho_star =", rho_star),
                                 paste("b_s =", b_s),
                                 paste("prior_scale =", prior_scale)))
pred_pr <- hat_b_x  # holder for predicted probability estimates
for (i in 1:n_sims) {  # loop to do monte carlo simulations
  for (j in 1:length(rho_star)) {
    for (k in 1:length(b_s)) {
      for (m in 1:length(prior_scale)) {
        x <- rnorm(n) + rho_star[j]*s  # explanatory variable, correlated with s (rho about 0.44)
        p <- plogis(b_cons[k] + b_s[k]*s + b_x*x)  # probability of an event
        y <- rbinom(n, 1, p)  # simulate outcome variable
        if (sum(y[s == 1]) == 0) {  # only if separation occurs
          mle <- bayesglm(y ~ s + x, family = binomial,  # fit model
                          prior.scale = c(prior_scale[m], Inf), prior.df = 1,
                          prior.scale.for.intercept = 10, prior.df.for.intercept = 1)
          hat_b_x[i, j, k, m] <- coef(mle)[3]  # store coefficient estimate
          pred_pr[i, j, k, m] <- predict(mle, # store predicted probability estimate for s = 0, x = 1
                                         newdata = data.frame(s = 0, x = 1),
                                         type = "response")
        }
      }
    }
  }
}

# calculate average estimate for each set of simulation parameters
round(apply(hat_b_x, c(2, 4, 3), mean, na.rm = TRUE), 2)
```

```
## , , b_s = -6
##
##                prior_scale = 10 prior_scale = 2.5 prior_scale = 1
## rho_star = 1               0.51              0.49            0.50
## rho_star = 2               0.51              0.49            0.49
## rho_star = 3               0.51              0.49            0.50
##
## , , b_s = -9
##
##                prior_scale = 10 prior_scale = 2.5 prior_scale = 1
## rho_star = 1               0.50              0.49            0.50
## rho_star = 2               0.49              0.48            0.50
## rho_star = 3               0.48              0.49            0.48
```

```
##
## , , b_s = -11
##
##                prior_scale = 10 prior_scale = 2.5 prior_scale = 1
## rho_star = 1              0.49              0.44            0.44
## rho_star = 2              0.40              0.39            0.39
## rho_star = 3              0.45              0.43            0.38
```

To further illustrate that the choice of prior matters less for the non-separating variables, I reproduced Bell and Miller's estimates and standard errors using MLE, Gelman's suggested Cauchy prior, and Zorn's suggested Jeffreys' prior. I plotted the coefficient estimates and asymptotic standard errors below. Notice that the estimates and standard errors are essentially identical across the plots.

```r
# load packages
library(ggplot2)

# load barrilleaux and rainey data
bm <- readr::read_csv("bm.csv")

# model formula
# set formula
f <- warl2 ~ onenukedyad + twonukedyad + logCapabilityRatio +
  Ally + SmlDemocracy + SmlDependence + logDistance +
  Contiguity + MajorPower + NIGOs

# fit models
m1 <- glm(f, data = bm, family = binomial)  # mle
m2 <- arm::bayesglm(f, data = bm, family = binomial)  # cauchy prior
m3 <- logistf::logistf(f, data = bm, family = binomial)  # jeffreys' prior

m1_df <- data.frame(method = "mle",
                    coef = names(coef(m1)),
                    est = coef(m1),
                    se = sqrt(diag(vcov(m1))))
m2_df <- data.frame(method = "cauchy",
                    coef = names(coef(m2)),
                    est = coef(m2),
                    se = sqrt(diag(vcov(m2))))
m3_df <- data.frame(method = "jeffrey",
                    coef = names(coef(m3)),
                    est = coef(m3),
                    se = sqrt(diag(vcov(m3))))

df <- rbind(m1_df, m2_df, m3_df)

ggplot(df, aes(x = method, y = est, ymin = est - se, ymax = est + se)) +
  geom_point() +
  geom_errorbar(width = 0) +
  facet_wrap(~ coef, scales = "free")
```



(Intercept)
Ally
Contiguity
logCapabilityRatio
logDistance
MajorPower
NIGOs
onenukedyad
SmlDemocracy
SmlDependence
twonukedyad
est
method
mle
cauchy
jeffrey