Computational Companion

to "Hypothesis Tests Under Separation"

In this computational companion, I illustrate how to compute the Wald, likelihood ratio, and score p-values using data from Barrilleaux and Rainey (2014).

Preliminary Data Work

First, I load the data from GitHub, select the variables we need (dropping the rest), and inverting the `gop_governor` indicator into an indicator of *Democratic* governors.

```
# load packages
library(tidyverse)

# load data and tidy the data
gh_data_url <- "https://raw.githubusercontent.com/carlislerainey/need/master/Data/politics_and_need_res
br <- read_csv(gh_data_url) %>%
  select(oppose_expansion, gop_governor, percent_favorable_aca, gop_leg, percent_uninsured,
         bal2012, multiplier, percent_nonwhite, percent_metro) %>%
  mutate(dem_governor = -1*gop_governor) %>%
  glimpse()
```

```
## Rows: 50
## Columns: 10
## $ oppose_expansion      <dbl> 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, ~
## $ gop_governor          <dbl> 0.4, 0.4, 0.4, -0.6, -0.6, -0.6, -0.6, -0.6, 0.4~
## $ percent_favorable_aca <dbl> -0.35384709, -0.40133316, -0.27352180, -0.474745~
## $ gop_leg               <dbl> 0.46, 0.46, 0.46, 0.46, -0.54, -0.54, -0.54, -0.~
## $ percent_uninsured     <dbl> -0.04385375, 0.56522612, 0.44341015, 0.44341015,~
## $ bal2012               <dbl> -0.192312167, 3.238509236, -0.103217193, -0.2056~
## $ multiplier            <dbl> 0.61380237, -0.49460333, 0.56837591, 0.80459351,~
## $ percent_nonwhite      <dbl> 0.119902567, 0.119902567, 0.530095558, -0.132523~
## $ percent_metro         <dbl> -0.01191702, -0.10721941, 0.30521706, -0.2431471~
## $ dem_governor          <dbl> -0.4, -0.4, -0.4, 0.6, 0.6, 0.6, 0.6, 0.6, -0.4,~
```

Intial Fit with Maximum Likelihood

We can then fit the model from their Figure 2 using maximum likelihood. The separation problem is immediately apparent.

```
# create model formula for the model shown in their Figure 2, p. 446
f <- oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg + percent_uninsured +
  bal2012 + multiplier + percent_nonwhite + percent_metro

# fit model with maximum likelihood
ml_fit <- glm(f, data = br, family = binomial)
```

```r
# print estimates and (Wald) p-values
arm::display(ml_fit, detail = TRUE)
```

```
## glm(formula = f, family = binomial, data = br)
##                       coef.est coef.se z value Pr(>|z|)
## (Intercept)             -8.86  1289.76   -0.01    0.99
## dem_governor           -20.35  3224.40   -0.01    0.99
## percent_favorable_aca    0.13     1.55    0.08    0.93
## gop_leg                  2.43     1.48    1.64    0.10
## percent_uninsured        0.92     2.23    0.41    0.68
## bal2012                 -0.05     0.85   -0.06    0.95
## multiplier              -0.35     1.19   -0.30    0.77
## percent_nonwhite         1.43     2.62    0.55    0.58
## percent_metro           -2.76     1.69   -1.64    0.10
## ---
##   n = 50, k = 9
##   residual deviance = 31.7, null deviance = 62.7 (difference = 31.0)
```

Under separation, the numerical algorithm is sensitive to numerical precision, so if we shrink the error tolerance, we obtain different coefficient estimates and standard error estimates. (Notice that the coefficient estimate gets *a little* larger, but the standard error estimate gets *a lot* larger–this is why the Wald test can never reject the null hypothesis under separation.)

```r
# fit model with maximum likelihood using maximum precision
ml_fit_maxprec <- glm(f, data = br, family = binomial, epsilon = 10^-16, maxit = 10^10)
```

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

```r
# print estimates and (Wald) p-values
arm::display(ml_fit_maxprec, detail = TRUE)
```

```
## glm(formula = f, family = binomial, data = br, epsilon = 10^-16,
##     maxit = 10^10)
##                       coef.est    coef.se     z value     Pr(>|z|)
## (Intercept)                -14.87  6002399.27        0.00        1.00
## dem_governor               -35.37 15005998.18        0.00        1.00
## percent_favorable_aca        0.13        1.55        0.08        0.93
## gop_leg                      2.43        1.48        1.64        0.10
## percent_uninsured            0.92        2.23        0.41        0.68
## bal2012                     -0.05        0.85       -0.06        0.95
## multiplier                  -0.35        1.19       -0.30        0.77
## percent_nonwhite             1.43        2.62        0.55        0.58
## percent_metro               -2.76        1.69       -1.64        0.10
## ---
##   n = 50, k = 9
##   residual deviance = 31.7, null deviance = 62.7 (difference = 31.0)
```

Penalized Maximum Likelihood

As an initial solution, we might try logistic regression with a Jeffreys or Cauchy prior. The Wald p-values from these penalized estimators are reasonable, but Rainey (2016) shows that the inferences depend on the penalty the researcher chooses. While we should not draw strong conclusions from this, the estimate using Jeffreys prior is not statistically significant, but the estimate using the Cauchy prior is statistically significant.

```r
# using jeffreys prior
pml_fit_jeffreys <- brglm::brglm(f, family = binomial, data = br)
summary(pml_fit_jeffreys)
```

```
##
## Call:
## brglm::brglm(formula = f, family = binomial, data = br)
##
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -1.4957     0.6040  -2.476   0.0133 *
## dem_governor           -2.6766     1.4208  -1.884   0.0596 .
## percent_favorable_aca  -0.1384     1.3133  -0.105   0.9161
## gop_leg                 1.6182     1.1737   1.379   0.1680
## percent_uninsured       0.1801     1.1271   0.160   0.8730
## bal2012                -0.1231     0.7252  -0.170   0.8652
## multiplier             -0.3265     1.0181  -0.321   0.7485
## percent_nonwhite        1.5620     1.2078   1.293   0.1959
## percent_metro          -1.8196     1.1879  -1.532   0.1256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 46.975  on 49  degrees of freedom
## Residual deviance: 34.365  on 41  degrees of freedom
## Penalized deviance: 32.26169
## AIC:  52.365
```

```r
# using cauchy prior
pml_fit_cauchy <- arm::bayesglm(f, family = binomial, data = br)
summary(pml_fit_cauchy)
```

```
##
## Call:
## arm::bayesglm(formula = f, family = binomial, data = br)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.52844  -0.57915  -0.09985   0.70392   2.01708
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -1.9129     0.7585  -2.522   0.0117 *
## dem_governor           -3.3791     1.6307  -2.072   0.0382 *
## percent_favorable_aca  -0.2085     1.0351  -0.201   0.8404
## gop_leg                 1.6956     1.0608   1.598   0.1100
## percent_uninsured       0.5998     1.0779   0.556   0.5779
## bal2012                 0.1548     0.7508   0.206   0.8367
## multiplier             -0.1624     0.8766  -0.185   0.8531
## percent_nonwhite        0.9340     1.2449   0.750   0.4531
## percent_metro          -1.4595     1.0439  -1.398   0.1621
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.687  on 49  degrees of freedom
## Residual deviance: 33.311  on 41  degrees of freedom
## AIC: 51.311
##
## Number of Fisher Scoring iterations: 22
```

However, the likelihood ratio and score tests work well without a prior distribution or penalty, so they offer a principled, frequentist alternative to penalized and Bayesian estimators.

Likelihood Ratio Test

The code computes the likelihood ratio test for the variable `dem_governor`. While it's possible to perform a likelihood-ratio test for each variable in the model, I've chosen to focus on a single variable. The single-variable approach aligns with the logic of the tests (i.e., an unrestricted model versus a restricted model) and clarifies that the test is not the standard Wald test.

```r
# fit unrestricted model
f <- oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg + percent_uninsured +
  bal2012 + multiplier + percent_nonwhite + percent_metro
ml_fit <- glm(f, data = br, family = binomial)

# fit the restricted model (omit dem_governor variable)
ml_fit0 <- update(ml_fit, . ~ . - dem_governor)

# likelihood-ratio test
anova(ml_fit0, ml_fit, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: oppose_expansion ~ percent_favorable_aca + gop_leg + percent_uninsured +
##     bal2012 + multiplier + percent_nonwhite + percent_metro
## Model 2: oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg +
##     percent_uninsured + bal2012 + multiplier + percent_nonwhite +
##     percent_metro
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        42     40.551
## 2        41     31.710  1   8.8407 0.002946 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# or alternatively
anova(ml_fit0, ml_fit, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: oppose_expansion ~ percent_favorable_aca + gop_leg + percent_uninsured +
##     bal2012 + multiplier + percent_nonwhite + percent_metro
## Model 2: oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg +
##     percent_uninsured + bal2012 + multiplier + percent_nonwhite +
##     percent_metro
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        42     40.551
```

```
## 2        41     31.710  1   8.8407 0.002946 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For a slightly more convenient syntax, we can use the `lrtest()` function in the lmtest package.

```
lmtest::lrtest(ml_fit, "dem_governor")  # specify name of variable to omit in the restricted model
```

```
## Likelihood ratio test
##
## Model 1: oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg +
##     percent_uninsured + bal2012 + multiplier + percent_nonwhite +
##     percent_metro
## Model 2: oppose_expansion ~ percent_favorable_aca + gop_leg + percent_uninsured +
##     bal2012 + multiplier + percent_nonwhite + percent_metro
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   9 -15.855
## 2   8 -20.276 -1 8.8407   0.002946 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternatively, we can use the `lr.test()` function in the mdscore package, though this requires fitting both models manually.

```
mdscore::lr.test(ml_fit0, ml_fit)
```

```
## $LR
## [1] 8.840705
##
## $pvalue
## [1] 0.002945853
##
## attr(,"class")
## [1] "lrt.test"
```

Score Test

The code below computes the score test for the variable `dem_governor`.

```
# fit unrestricted model
f <- oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg + percent_uninsured +
  bal2012 + multiplier + percent_nonwhite + percent_metro
ml_fit <- glm(f, data = br, family = binomial)

# fit the restricted model (omit dem_governor variable)
ml_fit0 <- update(ml_fit, . ~ . - dem_governor)

# likelihood-ratio test
anova(ml_fit0, ml_fit, test = "Rao")
```

```
## Analysis of Deviance Table
##
## Model 1: oppose_expansion ~ percent_favorable_aca + gop_leg + percent_uninsured +
##     bal2012 + multiplier + percent_nonwhite + percent_metro
## Model 2: oppose_expansion ~ dem_governor + percent_favorable_aca + gop_leg +
##     percent_uninsured + bal2012 + multiplier + percent_nonwhite +
```

```
##     percent_metro
##   Resid. Df Resid. Dev Df Deviance    Rao Pr(>Chi)
## 1        42     40.551
## 2        41     31.710  1   8.8407 6.8156 0.009037 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternatively, we can use the `glm.scoretest()` function in the statmod package or the `mdscore` function in the mdscore package, though these methods are slightly more tedious.

```
mm <- model.matrix(ml_fit, data = br)
score <- statmod::glm.scoretest(ml_fit0, x2 = mm[, 2])
2*(1 - pnorm(abs(score))) # p-value
```

```
## [1] 0.009036665
```

```
mm <- model.matrix(ml_fit, data = br)
score <- mdscore::mdscore(ml_fit0, X1 = mm[, 2])
summary(score)
```

```
##                Df  Value  P-value
## Score           1   6.82   0.0090
## Modified score  1   6.14   0.0132
```