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. Note that all numeric variable are rescaled to have mean 0 and SD 0.5 and all indicators are rescaled to have mean 0.

```
# 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) %>%
  # recode binary predictor so that 1 perfectly predicts the outcome
  mutate(dem_governor = 1 - (gop_governor - min(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, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, ~
```

Initial Fit with Maximum Likelihood

We can then fit the model from their Figure 2 using maximum likelihood. The separation problem is immediately apparent. The `z value` and `Pr(>|z|)` columns in the `summary()` output reports the Wald z-statistic and p-value.

```
# 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
```

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

# print estimates and (Wald) p-values
summary(ml_fit)
```

```
##
## Call:
## glm(formula = f, family = binomial, data = br)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73776  -0.45518  -0.00001   0.59069   2.35004
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.71545    0.66708  -1.073    0.283
## dem_governor            -20.34924 3224.39979  -0.006    0.995
## percent_favorable_aca     0.12755    1.54920   0.082    0.934
## gop_leg                   2.42938    1.47965   1.642    0.101
## percent_uninsured         0.92303    2.23424   0.413    0.680
## bal2012                  -0.05353    0.85353  -0.063    0.950
## multiplier               -0.35474    1.19260  -0.297    0.766
## percent_nonwhite          1.43356    2.61588   0.548    0.584
## percent_metro            -2.75893    1.68666  -1.636    0.102
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.687  on 49  degrees of freedom
## Residual deviance: 31.710  on 41  degrees of freedom
## AIC: 49.71
##
## Number of Fisher Scoring iterations: 19
```

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 = .Machine$double.eps, maxit = 10^10)
```

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

```r
# print estimates and (Wald) p-values
summary(ml_fit_maxprec)
```

```
##
## Call:
## glm(formula = f, family = binomial, data = br, epsilon = .Machine$double.eps,
##     maxit = 10^10)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7378  -0.4552   0.0000   0.5907   2.3500
##
```

```
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -7.155e-01  6.671e-01  -1.073    0.283
## dem_governor          -3.435e+01  1.501e+07   0.000    1.000
## percent_favorable_aca  1.275e-01  1.549e+00   0.082    0.934
## gop_leg                2.429e+00  1.480e+00   1.642    0.101
## percent_uninsured      9.230e-01  2.234e+00   0.413    0.680
## bal2012               -5.353e-02  8.535e-01  -0.063    0.950
## multiplier            -3.547e-01  1.193e+00  -0.297    0.766
## percent_nonwhite       1.434e+00  2.616e+00   0.548    0.584
## percent_metro         -2.759e+00  1.687e+00  -1.636    0.102
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.687  on 49  degrees of freedom
## Residual deviance: 31.710  on 41  degrees of freedom
## AIC: 49.71
##
## Number of Fisher Scoring iterations: 33
```

Detecting Separation

After noticing the unusual coefficients and strangely large standard error estimates, we might use the detectseparation package to formally check that separation actually exists. The package has two methods: the pre-fit `detect_separation()` method and the post-fit `check_infinite_estimates()` method. Both methods should agree. See the helpful vignette for additional information.

```
library(detectseparation)

# pre-fit detection
ml_detect <- glm(f, data = br, family = binomial, method = "detect_separation")
ml_detect
```

```
## Implementation: ROI | Solver: lpsolve
## Separation: TRUE
## Existence of maximum likelihood estimates
##           (Intercept)          dem_governor percent_favorable_aca
##                     0                  -Inf                     0
##               gop_leg     percent_uninsured               bal2012
##                     0                     0                     0
##            multiplier      percent_nonwhite         percent_metro
##                     0                     0                     0
## 0: finite value, Inf: infinity, -Inf: -infinity
```

```
# post-fit detection
ml_detect <- glm(f, data = br, family = binomial, method = "detect_infinite_estimates")
ml_detect
```

```
## Implementation: ROI | Solver: lpsolve
## Infinite estimates: TRUE
## Existence of maximum likelihood estimates
##           (Intercept)          dem_governor percent_favorable_aca
##                     0                  -Inf                     0
##               gop_leg     percent_uninsured               bal2012
##                     0                     0                     0
```

```
##            multiplier      percent_nonwhite         percent_metro
##                     0                     0                     0
## 0: finite value, Inf: infinity, -Inf: -infinity
```

We can conveniently update the coefficient estimates with the output of `detect_separation()` or `check_infinite_estimates()`.

```
# print coefficient estimates
coef(ml_fit) + coef(ml_detect)
```

```
##           (Intercept)          dem_governor percent_favorable_aca
##           -0.71544693                  -Inf            0.12754770
##               gop_leg     percent_uninsured               bal2012
##            2.42938293            0.92303145           -0.05353412
##            multiplier      percent_nonwhite         percent_metro
##           -0.35474333            1.43356127           -2.75893137
```

```
# adjust estimates in texreg
texreg::screenreg(list(ml_fit, ml_fit),
                  override.coef = list(coef(ml_fit),
                                       coef(ml_fit) + coef(ml_detect)))
```

```
##
## ========================================
##                         Model 1    Model 2
## ----------------------------------------
## (Intercept)               -0.72    -0.72
##                           (0.67)   (0.67)
## dem_governor             -20.35     -Inf
##                         (3224.40)
## percent_favorable_aca      0.13     0.13
##                           (1.55)   (1.55)
## gop_leg                    2.43     2.43
##                           (1.48)   (1.48)
## percent_uninsured          0.92     0.92
##                           (2.23)   (2.23)
## bal2012                   -0.05    -0.05
##                           (0.85)   (0.85)
## multiplier                -0.35    -0.35
##                           (1.19)   (1.19)
## percent_nonwhite           1.43     1.43
##                           (2.62)   (2.62)
## percent_metro             -2.76    -2.76
##                           (1.69)   (1.69)
## ----------------------------------------
## AIC                       49.71    49.71
## BIC                       66.92    66.92
## Log Likelihood           -15.86   -15.86
## Deviance                  31.71    31.71
## Num. obs.                 50       50
## ========================================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

Likelihood Ratio and Score Tests

As a first step, we might want to obtain a p-value for the coefficient of `dem_governor` that is reasonable, but without turning immediately to penalized estimation, since Rainey (2016) shows that the inferences can be sensitive to the choice of prior. The likelihood ratio and score tests work well without a prior distribution or penalty, so they offer a principled, frequentist alternative to the p-values from penalized and Bayesian estimators.

Likelihood Ratio

The below code computes the likelihood ratio test for the variable `dem_governor`.

```
# 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
```

The code below computes the *same* likelihood ratio p-value by supplying the same `test = "LRT"` argument to `anova()`.

```
# likelihood ratio test, 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. This function takes the unrestricted fit as the first argument and the name of the variable to be dropped in the restricted model as the second argument.

```
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
```

Score Test

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

```
# score 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
```

Both Tests for All Variables in the Model

The researcher only needs to compute the likelihood ratio or score tests for the separating variable (`dem_governor` in this case). However, the `summarylr()` function reports likelihood ratio and/or score tests for all coefficients.

```r
# ml with default precision
print(glmglrt::summarylr(ml_fit, force = TRUE, keep.wald = TRUE,
                         method = c("LRT", "Rao")), signif.stars = FALSE)
```

```
##
## Call:
## glm(formula = f, family = binomial, data = br)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73776  -0.45518  -0.00001   0.59069   2.35004
##
## Coefficients:
##                          Estimate Std. Error z value   Pr(>|z|) LRT P-value
## (Intercept)           -7.154e-01  6.671e-01  -1.073  2.835e-01    0.237686
## dem_governor          -2.035e+01  3.224e+03  -0.006  9.950e-01    0.002946
## percent_favorable_aca  1.275e-01  1.549e+00   0.082  9.344e-01    0.934332
## gop_leg                2.429e+00  1.480e+00   1.642  1.006e-01    0.062830
## percent_uninsured      9.230e-01  2.234e+00   0.413  6.795e-01    0.677801
## bal2012               -5.353e-02  8.535e-01  -0.063  9.500e-01    0.950431
## multiplier            -3.547e-01  1.193e+00  -0.297  7.661e-01    0.765817
## percent_nonwhite       1.434e+00  2.616e+00   0.548  5.837e-01    0.580658
## percent_metro         -2.759e+00  1.687e+00  -1.636  1.019e-01    0.055583
##                       Rao P-value
## (Intercept)               0.26466
## dem_governor              0.00904
## percent_favorable_aca     0.93437
## gop_leg                   0.07281
## percent_uninsured         0.67838
## bal2012                   0.94995
## multiplier                0.76564
## percent_nonwhite          0.58162
## percent_metro             0.07554
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.687  on 49  degrees of freedom
## Residual deviance: 31.710  on 41  degrees of freedom
## AIC: 49.71
##
## Number of Fisher Scoring iterations: 19
```

```r
# ml with maximum precision
print(glmglrt::summarylr(ml_fit_maxprec, force = TRUE, keep.wald = TRUE,
                         method = c("LRT", "Rao")), signif.stars = FALSE)
```

```
##
## Call:
## glm(formula = f, family = binomial, data = br, epsilon = .Machine$double.eps,
##     maxit = 10^10)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7378  -0.4552   0.0000   0.5907   2.3500
##
```

```
## Coefficients:
##                         Estimate Std. Error z value   Pr(>|z|) LRT P-value
## (Intercept)           -7.154e-01  6.671e-01  -1.073  2.835e-01    0.237686
## dem_governor          -3.435e+01  1.501e+07   0.000  1.000e+00    0.002946
## percent_favorable_aca  1.275e-01  1.549e+00   0.082  9.344e-01    0.934332
## gop_leg                2.429e+00  1.480e+00   1.642  1.006e-01    0.062830
## percent_uninsured      9.230e-01  2.234e+00   0.413  6.795e-01    0.677801
## bal2012               -5.353e-02  8.535e-01  -0.063  9.500e-01    0.950431
## multiplier            -3.547e-01  1.193e+00  -0.297  7.661e-01    0.765817
## percent_nonwhite       1.434e+00  2.616e+00   0.548  5.837e-01    0.580658
## percent_metro         -2.759e+00  1.687e+00  -1.636  1.019e-01    0.055583
##                       Rao P-value
## (Intercept)               0.26466
## dem_governor              0.00904
## percent_favorable_aca     0.93437
## gop_leg                   0.07281
## percent_uninsured         0.67838
## bal2012                   0.94996
## multiplier                0.76564
## percent_nonwhite          0.58162
## percent_metro             0.07554
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 62.687  on 49  degrees of freedom
## Residual deviance: 31.710  on 41  degrees of freedom
## AIC: 49.71
##
## Number of Fisher Scoring iterations: 33
```

Penalized Maximum Likelihood

To obtain reasonable point estimates and compute meaningful quantities of interest, the researcher needs to use penalized estimatioon. For example, they might use logistic regression with a Jeffreys or Cauchy prior. I do not illustrate it here, but Stan provides a powerful tool for MCMC simulation, especially when interfaced with R using the `rstan`, `cmdstanr`, `rstanarm`, or `brms` packages.

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.

```
# 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)            -0.4250     0.5133  -0.828   0.4077
```

```
## 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
```

```
# 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
## -1.52848  -0.57930  -0.09998   0.70389   2.01680
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -0.5610     0.5240  -1.071   0.2843
## dem_governor           -3.3771     1.6294  -2.073   0.0382 *
## percent_favorable_aca  -0.2086     1.0350  -0.202   0.8403
## gop_leg                 1.6953     1.0607   1.598   0.1100
## percent_uninsured       0.5995     1.0777   0.556   0.5780
## bal2012                 0.1548     0.7508   0.206   0.8367
## multiplier             -0.1625     0.8766  -0.185   0.8530
## percent_nonwhite        0.9341     1.2447   0.751   0.4529
## percent_metro          -1.4592     1.0438  -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: 33.313  on 41  degrees of freedom
## AIC: 51.313
##
## Number of Fisher Scoring iterations: 22
```

Beyond the coefficient estimates and p-values, the researcher can use either of these approaches to obtain reseanable quantities of interest. Researchers should be wary, though, that default penalties are not suitable for all substantive applications, so careful thought about the prior distribution or robustness checks are

warranted.

I do not illustrate it here, but researchers can use the informal posterior simulation suggested by King, Tomz, and Wittenberg (2001) to simulate the model coefficients and then transform these into simulations of the quantities of interest. The `sim()` function in the arm package simulates the coefficients.