The Computational Companion

to "Meaningful Hypothesis Tests Under Separation (Without Prior Information)"

Example 1 from the Paper

Wald test of the logistic regression coefficient

```
n <- 1000
x <- c(rep(0, n/2), rep(1, n/2))
y <- x
data <- data.frame(x, y)

fit <- glm(y ~ x, data = data, family = binomial)
```

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

```
arm::display(fit, detail = TRUE, digits = 3)
```

```
## glm(formula = y ~ x, family = binomial, data = data)
##             coef.est  coef.se   z value   Pr(>|z|)
## (Intercept)   -26.566 15926.349    -0.002     0.999
## x              53.132 22523.261     0.002     0.998
## ---
##   n = 1000, k = 2
##   residual deviance = 0.0, null deviance = 1386.3 (difference = 1386.3)
```

```
fit_max <- glm(y ~ x, data = data, family = binomial,
           epsilon = 10^(-100), maxit = 10^10)
```

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

```
arm::display(fit_max, detail = TRUE, digits = 3)
```

```
## glm(formula = y ~ x, family = binomial, data = data, epsilon = 10^(-100),
##     maxit = 10^10)
##             coef.est    coef.se     z value     Pr(>|z|)
## (Intercept)     -31.566 3001199.636       0.000       1.000
## x                63.132 4244337.229       0.000       1.000
## ---
##   n = 1000, k = 2
##   residual deviance = 0.0, null deviance = 1386.3 (difference = 1386.3)
```

Exact test

For the exact test, choose $p = 0.5$ as the null hypothesis to reject because it produces the largest p-value. Assuming that $p = 0.5$, then there are two outcomes that are at least as extreme as the observed data: the observed data and also perfect prediction in the opposite direction. Both have probability $\left(\frac{1}{2}\right)^{500} \times \left(\frac{1}{2}\right)^{500}$. Because these are mutual exclusive, simply mulitply them together to obtain the p-value $\frac{2}{2^{1000}}$.

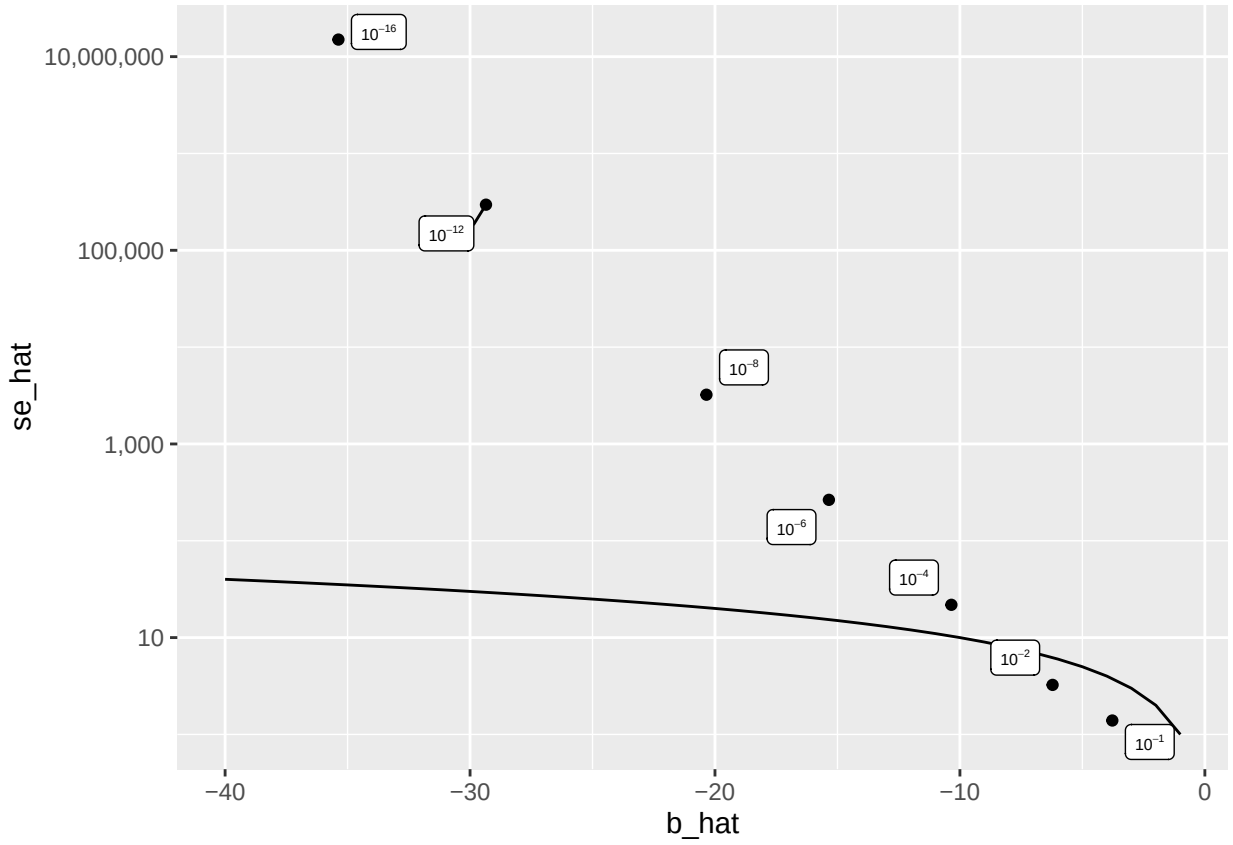
Barrilleaux and Rainey convergence

```
# load packages
library(tidyverse)
library(ggrepel)
library(scales)

# set seed
set.seed(784123)

# load data
br_df <- read_csv("data/politics_and_need_rescale.csv") %>%
  mutate(dem_governor = -1*gop_governor)

# 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

# a function to compute quantities of interest for each fit
fit_and_get_f <- function(e) {
  ml_fit  <- glm(f, data = br_df, family = binomial, epsilon = e, maxit = 10000000)
  ml0_fit <- update(ml_fit, . ~ . - dem_governor)
  ll <- as.numeric(logLik(ml_fit))       # log-likelihood of alternative model
  ll0 <- as.numeric(logLik(ml0_fit))     # log-likelihood of null model
  test_statistic <- -2*(ll0 - ll)     # wilk's test statistic
  degrees_of_freedom <- 1             # wilk's degrees-of-freedom
  lr_p <- 1 - pchisq(test_statistic, df = degrees_of_freedom)  # wilk's p-val
  res_df <- tibble(
    e = e,
    exponent = log10(e),
    b_hat = coef(ml_fit)[2],
    se_hat = sqrt(diag(vcov(ml_fit)))[2],
    wald_p = 2*pnorm((abs(coef(ml_fit))/sqrt(diag(vcov(ml_fit))))[2], lower.tail = FALSE),
    lr_p = lr_p)
  return(res_df)
}

# fit models varying the tolerance
exponent <- c(-1, -2, -4, -6, -8, -12, -16)
conv_df <- 10^exponent %>%
  map(~ fit_and_get_f(.)) %>%
  bind_rows() %>%
  mutate(e_text = paste0("10^", exponent)) %>%
  write_csv("output/br-convergence-gh.csv") %>%
  glimpse()
```

```
## Rows: 7
## Columns: 7
## $ e        <dbl> 1e-01, 1e-02, 1e-04, 1e-06, 1e-08, 1e-12, 1e-16
## $ exponent <dbl> -1, -2, -4, -6, -8, -12, -16
## $ b_hat    <dbl> -3.777654, -6.216560, -10.348540, -15.349238, -20.349242, -29~
## $ se_hat   <dbl> 1.392568e+00, 3.249130e+00, 2.181425e+01, 2.646821e+02, 3.224~
## $ wald_p   <dbl> 0.006673251, 0.055709657, 0.635219314, 0.953755625, 0.9949645~
## $ lr_p     <dbl> 0.004136258, 0.003087086, 0.002948371, 0.002945870, 0.0029458~
```

```
## $ e_text   <chr> "10^-1", "10^-2", "10^-4", "10^-6", "10^-8", "10^-12", "10^-1~
```

```
linear <- tibble(x = -1:-40) %>%
  mutate(y = -x)
ggplot(conv_df, aes(x = b_hat, y = se_hat)) +
  scale_x_continuous() +
  scale_y_log10(labels = label_comma()) +
  geom_line(data = linear, aes(x = x, y = y)) +
  geom_point() +
  geom_label_repel(aes(label = e_text), parse = TRUE, size = 2)
```



Example 1: No Separation

The code below shows how to fit a logistic regression model with the usual maximum likelihood approach and two penalized maximum likelihood approaches and compute p-values using the Wald and likelihood ratio methods.

I begin in a situation without separation.

Load the Data

```
# load packages
library(tidyverse)
library(brglm)
library(arm)
```

```
# Zelig not loaded, but needed below
# arm not loaded, but needed below

# load turnout data from from Zelig package
data(turnout, package = "Zelig")

# quick overview of data
glimpse(turnout)
```

```
## Rows: 2,000
## Columns: 5
## $ race    <fct> white, white, white, white, white, white, white, white, white,~
## $ age     <int> 60, 51, 24, 38, 25, 67, 40, 56, 32, 75, 46, 52, 22, 60, 24, 30~
## $ educate <dbl> 14, 10, 12, 8, 12, 12, 12, 10, 12, 16, 15, 12, 12, 12, 14, 10,~
## $ income  <dbl> 3.3458, 1.8561, 0.6304, 3.4183, 2.7852, 2.3866, 4.2857, 9.3205~
## $ vote    <int> 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,~
```

Fit the Models

```
# fit a logistic regression model
formula <- vote ~ age + race
ml_fit <- glm(formula, family = binomial, data = turnout)
pml_fit_jeffreys <- brglm(formula, family = binomial, data = turnout)
pml_fit_cauchy <- bayesglm(formula, family = binomial, data = turnout)

# combine fits into a list
fit_list <- list("ML" = ml_fit,
                 "PML: Jeffreys" = pml_fit_jeffreys,
                 "PML: Cauchy" = pml_fit_cauchy)

# print a summary table of logistic regression fit
texreg::screenreg(fit_list)
```

```
##
## ============================================================
##                  ML            PML: Jeffreys  PML: Cauchy
## ------------------------------------------------------------
## (Intercept)         0.04          0.04           0.04
##                    (0.18)        (0.18)         (0.18)
## age                 0.01 ***      0.01 ***       0.01 ***
##                    (0.00)        (0.00)         (0.00)
## racewhite           0.65 ***      0.65 ***       0.64 ***
##                    (0.13)        (0.13)         (0.13)
## ------------------------------------------------------------
## AIC              2234.82       2234.82        2234.82
## BIC              2251.62       2251.62        2251.62
## Log Likelihood  -1114.41      -1114.41       -1114.41
## Deviance         2228.82       2228.82        2228.82
## Num. obs.        2000          2000           2000
## ============================================================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

The table above presents Wald p-values using stars.

We can see the exact p-values using `summary()`.

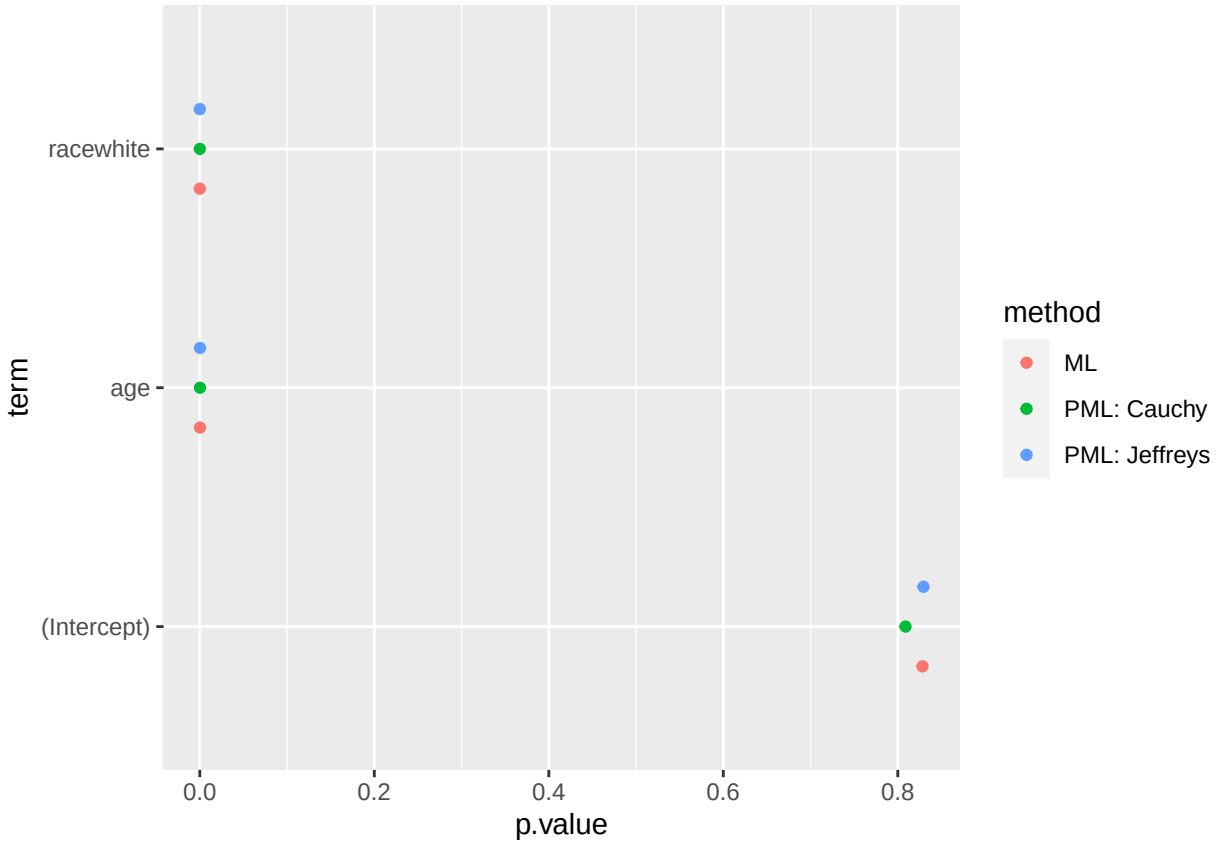
We can use broom's `tidy()` function to get the p-values into a data frame

```
library(broom)

fits <- fit_list %>%
  map(~ tidy(.)) %>%  # create a data frame of predictor-level info
  imap(~ mutate(.x, method = .y)) %>%  # add a variable indicating estimation method
  bind_rows() %>%  # combine the fits into a single data frame
  glimpse()
```

```
## Rows: 9
## Columns: 6
## $ term      <chr> "(Intercept)", "age", "racewhite", "(Intercept)", "age", "ra~
## $ estimate  <dbl> 0.03836527, 0.01126321, 0.64555103, 0.03809500, 0.01121645, ~
## $ std.error <dbl> 0.176920299, 0.003053245, 0.134481912, 0.176854330, 0.003051~
## $ statistic <dbl> 0.2168506, 3.6889302, 4.8002815, 0.2154032, 3.6758361, 4.807~
## $ p.value   <dbl> 8.283248e-01, 2.251990e-04, 1.584428e-06, 8.294529e-01, 2.37~
## $ method    <chr> "ML", "ML", "ML", "PML: Jeffreys", "PML: Jeffreys", "PML: Je~
```

We can use this data frame to plot the p-values, for example.

```
ggplot(fits, aes(y = p.value, x = term, color = method)) +
  geom_point(position = position_dodge(width = 1/2)) +
  coord_flip()
```



While the Wald test is convenient because it requires fitting just one model, it makes sense to use the likelihood-ratio test in some cases. For example, the likelihood ratio test gives a reasonable result in the case

of separation while the Wald test does not.

The code below uses base R function to compute the likelihood ratio tests for the variable `race`. While it's possible to perform a likelihood-ratio test for each variable in the model (and the constant), I've chosen to focus on single variables. 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.

```
# fit the restricted model (omit race variable)
ml_fit0 <- glm(vote ~ age, family = binomial, data = turnout)
# or alternatively
ml_fit0 <- update(ml_fit, . ~ . - race)

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

```
## Analysis of Deviance Table
##
## Model 1: vote ~ age
## Model 2: vote ~ age + race
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      1998     2250.9
## 2      1997     2228.8  1   22.097 2.592e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
## Analysis of Deviance Table
##
## Model 1: vote ~ age
## Model 2: vote ~ age + race
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      1998     2250.9
## 2      1997     2228.8  1   22.097 2.592e-06 ***
## ---
## 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.

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

```
## Likelihood ratio test
##
## Model 1: vote ~ age + race
## Model 2: vote ~ age
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   3 -1114.4
## 2   2 -1125.5 -1 22.097  2.592e-06 ***
## ---
## 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.

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

```
## $LR
## [1] 22.09724
##
## $pvalue
## [1] 2.591821e-06
##
## attr(,"class")
## [1] "lrt.test"
```