FMPH243A simulation3

Keren Hu, Jennifer Zhang

2023-11-16

Model performance

1. Poisson log-linear regression model

```r
library(MASS)
set.seed(2023)

M <- 1000
n <- 1000
beta0 <- 1
beta1 <- 1
sigma_sq_x <- 1
sigma_sq_b <- 2

beta0_1 <- log(3) + 0.5*sigma_sq_b + beta0


# Initialize vectors
beta_hats <- matrix(0, ncol = 2, nrow = M)
asy_var <- list()
Tws <- numeric(M)

for (m in 1:M) {
  # Simulate data for each iteration
  # Simulating random effects bi
  bi <- rnorm(n, mean = 0, sd = sqrt(sigma_sq_b))

  # Simulating predictor variable xi
  xi <- rnorm(n, mean = 1, sd = sqrt(sigma_sq_x))

  # Simulating correlated count responses u_it
  u <- matrix(0, nrow = n, ncol = 3)
  for (i in 1:n) {
    mu_it <- exp(beta0 + xi[i] * beta1 + bi[i])
    for (t in 1:3){
      u[i, t] <- rpois(1, mu_it)
    }
  }
  y <- rowSums(u)
```

```r
  # Fit Poisson log-linear regression model
  m1 <- glm(y ~ xi, family = poisson(link = "log"))

  # Obtain estimates and associated asymptotic variance
  beta_hats[m,] <- t(as.matrix(coef(m1)))
  var_hat <- diag(vcov(m1))[2]

  # Compute Wald statistic for testing the null hypothesis beta1 = 1
  Tw <- (beta_hats[m,2] - beta1)^2 / var_hat

  # Store results for each iteration
  asy_var[[m]] <- vcov(m1)
  Tws[m] <- Tw
}

# Compute empirical type I error rate
## compare to alpha = 0.05
mean(Tws >= qchisq(0.95, df = 1))
```

```
## [1] 0.968
```

```r
# Compute MC estimate of beta
## compare to true value (3.098612, 1)
colMeans(beta_hats)
```

```
## [1] 3.1088669 0.9884016
```

```r
# Compute MC estimate of associated asymptotic variances
Reduce("+", asy_var)/M
```

```
##               (Intercept)            xi
## (Intercept)  5.119671e-05 -2.078698e-05
## xi          -2.078698e-05  1.060122e-05
```

```r
# Compute MC estimate of empirical variances
var(beta_hats)
```

```
##             [,1]        [,2]
## [1,]  0.05491794 -0.03039946
## [2,] -0.03039946  0.01995864
```

2. NB log-linear regression model

```r
library(MASS)
set.seed(2023)

M <- 1000
n <- 1000
beta0 <- 1
```

```r
beta1 <- 1
sigma_sq_x <- 1
sigma_sq_b <- 2


# Initialize vectors to store results
beta_hats <- matrix(0, ncol = 2, nrow = M)
asy_var <- list()
Tws <- numeric(M)

errors <- list()
error_data <- list()

for (m in 1:M) {
  # Using tryCatch to catch errors
  tryCatch({
    # Simulate data for each iteration
    # Simulating random effects bi
    bi <- rnorm(n, mean = 0, sd = sqrt(sigma_sq_b))

    # Simulating predictor variable xi
    xi <- rnorm(n, mean = 1, sd = sqrt(sigma_sq_x))

    # Simulating correlated count responses u_it
    u <- matrix(0, nrow = n, ncol = 3)
    for (i in 1:n) {
      mu_it <- exp(beta0 + xi[i] * beta1 + bi[i])
      for (t in 1:3){
        u[i, t] <- rpois(1, mu_it)
      }
    }
    y <- rowSums(u)

    # Fit NB log-linear regression model
    m2 <- glm.nb(y ~ xi)

    # Obtain estimates and associated asymptotic variance
    beta_hats[m,] <- t(as.matrix(coef(m2)))
    var_hat <- diag(vcov(m2))[2]

    # Compute Wald statistic for testing the null hypothesis beta1 = 1
    Tw <- (beta_hats[m,2] - beta1)^2 / var_hat

    # Store results for each iteration
    asy_var[[m]] <- vcov(m2)
    Tws[m] <- Tw
  }, error = function(e) {
    # Store error message
    errors[[m]] <- conditionMessage(e)
    # Store dataset causing the error
    error_data[[m]] <- list(bi = bi, xi = xi, u = u, y = y)
  })
}
```

```r
# Compute empirical type I error rate
## compare to alpha = 0.05
mean(Tws >= qchisq(0.95, df = 1))
```

```
## [1] 0.273
```

```r
# Compute MC estimate of beta
## compare to true value (3.098612, 1)
colMeans(beta_hats)
```

```
## [1] 3.085951 1.003719
```

```r
# Compute MC estimate of associated asymptotic variances
asy_array <- array(unlist(asy_var), dim = c(2, 2, length(asy_var)))
apply(asy_array, c(1, 2), mean)
```

```
##              [,1]         [,2]
## [1,]  0.003569155 -0.001801211
## [2,] -0.001801211  0.001775653
```

```r
# Compute MC estimate of empirical variances
var(beta_hats)
```

```
##              [,1]         [,2]
## [1,]  0.021274859 -0.002584925
## [2,] -0.002584925  0.006801673
```

3. Semiparametric log-linear model with the working variance

```r
library(MASS)
library(geepack)
library(gee)
set.seed(2023)

M <- 1000
n <- 1000
beta0 <- 1
beta1 <- 1
sigma_sq_x <- 1
sigma_sq_b <- 2

beta0_1 <- log(3) + 0.5*sigma_sq_b + beta0


# Initialize vectors to store results
beta_hats <- matrix(0, ncol = 2, nrow = M)
asy_var <- list()
Tws <- numeric(M)
```

```r
for (m in 1:M) {
  # Simulate data for each iteration
  # Simulating random effects bi
  bi <- rnorm(n, mean = 0, sd = sqrt(sigma_sq_b))

  # Simulating predictor variable xi
  xi <- rnorm(n, mean = 1, sd = sqrt(sigma_sq_x))

  # Simulating correlated count responses u_it
  u <- matrix(0, nrow = n, ncol = 3)
  for (i in 1:n) {
    mu_it <- exp(beta0 + xi[i] * beta1 + bi[i])
    for (t in 1:3){
      u[i, t] <- rpois(1, mu_it)
    }
  }
  y <- rowSums(u)

  # Create a data frame for GEE modeling
  df <- data.frame(y = y, xi = xi, id = seq(1,n))

  # Fit GEE regression model
  m3 <- gee(y ~ xi, data = df, family = "poisson")

  # Obtain estimates and associated asymptotic variance
  beta_hats[m,] <- m3$coefficients
  var_hat <-m3$robust.variance["xi", "xi"]

  # Compute Wald statistic for testing the null hypothesis beta1 = 1
  Tw <- (beta_hats[m, 2] - beta1)^2 / var_hat

  # Store results for each iteration
  asy_var[[m]] <- m3$robust.variance
  Tws[m] <- Tw
}
```

```r
# Compute empirical type I error rate
## compare to alpha = 0.05
mean(Tws >= qchisq(0.95, df = 1))
```

```
## [1] 0.162
```

```r
# Compute MC estimate of beta
## compare to true value (, 1)
colMeans(beta_hats)
```

```
## [1] 3.1088669 0.9884016
```

```r
# Compute MC estimate of associated asymptotic variances
Reduce("+", asy_var)/M
```

```
##             (Intercept)          xi
```

```
## (Intercept)  0.04445303 -0.02357939
## xi          -0.02357939  0.01528996
```

```r
# Compute MC estimate of empirical variances
var(beta_hats)
```

```
##             [,1]        [,2]
## [1,]  0.05491794 -0.03039946
## [2,] -0.03039946  0.01995864
```