과대산포에 대한 모의실험

Jieun Shin

2022-10-24

1. 실험의 목적

- 과대산포를 허용하는 음이항 모형과 일반화 포아송 모형에서 반응변수를 생성하고, 유의수준 0.05에서 과대산포에 대한 3가지 검정 (LR, Wald, Score test)을 실시한다.
- 3가지 검정법을 사용했을 때의 기각 및 채택여부를 파악한 후, 추정된 유의수준과 검정력을 비교한다.

2. 3가지 검정 통계량

- Likelihood ratio test

$$\text{LRT} = -2[\log L(\tilde{\theta}_0) - \log L(\hat{\theta})],$$

여기서 $\tilde{\theta}_0 = (\tilde{\boldsymbol{\beta}}, 0)^T$는 포아송 회귀모형에서의 MLE이고 $\hat{\theta} = (\hat{\boldsymbol{\beta}}, \hat{\tau})^T$는 음이항 (혹은 일반화 포아송) 회귀모형에서의 MLE이다. 가설이 단측가설이므로 $\text{LR} = \sqrt{\text{LRT}} \sim N(0,1)$을 이용하여 가설검정을 실시함.

- Wald test

$$W = \frac{\hat{\tau}}{\hat{se}(\hat{\tau})},$$

현재 고려한 음이항 (혹은 일반화 포아송) 회귀모형에서의 $\hat{\tau}$에 대한 추정량과 $\hat{\tau}$의 표준오차를 이용함. $W \sim N(0,1)$를 이용하여 가설검정을 실시함.

- Score test

스코어 통계량

$$\text{Score} = \frac{\sum_{i=1}^{n}((y_i - \tilde{\mu}_i)^2 - y_i)}{2\sum_{i=1}^{n}\tilde{\mu}_i^2}$$

혹은 조정된 스코어 통계량

$$\text{Score}_a = \frac{\sum_{i=1}^{n}((y_i - \tilde{\mu}_i)^2 - y_i)}{2\sum_{i=1}^{n}\tilde{\mu}_i^2}$$

이고, 여기서 $\tilde{\mu}_i$는 포아송 회귀모형에서의 MLE이다. $\text{Score}, \text{Score}_a \sim N(0,1)$를 이용하여 가설검정을 실시함.

3. 모의실험

모의실험 디자인

- $E(Y_i) = \mu_i = \exp(\beta_0 + \beta_1 x_i)$와 τ (산포모수)를 기반으로 함.
- 설명변수 x_i는 Unif$(0,1)$에서 생성함.
- 회귀계수 β_0와 β_1의 참값은 각각 1.0와 1.0로 설정함. 이를 기반으로 μ_i의 값을 계산함.
- 반응변수 Y_i는 $NB(\mu_i, \tau)$에서 생성함. 이때 산포모수의 값은 0에서 0.1까지 0.02단위로 움직임.
- 표본 수 n은 50, 100과 200을 사용함.

- 각 모수의 값에서 총 1000번의 반복을 실시함. 각 반복에서는 3가지 검정통계량 값을 계산함.

함수 코드 작성

```
dgenpois = function(x, mu, tau){
  (mu/(1+tau*mu))^x * (1+tau*x)^{x-1} / factorial(x) * exp(-mu*(1+tau*x)/(1+tau*mu))
}

logL = function(y, mu, tau){
  val = y* log(mu/(1+tau*mu)) + (y-1)*log(1+tau*y) - mu*(1+tau*y)/(1+tau*mu)-log(factorial(y))
  return(sum(val))
}

Dvec = function(x, y, mu, tau){
  n = dim(x)[1]
  p = dim(x)[2]

  dbeta = c()
  for(j in 1:p){
    dbeta[j] = sum( x[,j] * (y-mu) / (1 + tau * mu)^2 )
  }

  dtau = sum( -(y*mu)/(1+tau*mu) + y*(y-1)/(1+tau*y) - mu*(y-mu)/(1+tau*mu)^2 )

  return(c(dbeta, dtau))
}


###

Hmat = function(x, y, mu, tau){
  n = dim(x)[1]
  p = dim(x)[2]

  ddbeta = matrix(0, p, p)
  for(j in 1:p){
    for(k in 1:p){
      ddbeta[j, k] = - sum( (1+2*tau*y - tau*mu)*mu / (1+tau*mu)^3 * x[,j] * x[,k] )
    }
  }

  ddbetatau = c()
  for(j in 1:p){
    ddbetatau[j] = - sum( 2*(y-mu)*mu / (1+tau*mu)^3 * x[,j] )
  }

  ddtau = sum( (3*y*mu^2 + mu^3*(tau*y-2))/(1+tau*mu)^3 - y^2*(y-1)/(1+tau*y)^2 )

  Hessian = cbind(rbind(ddbeta, ddbetatau), c(ddbetatau, ddtau))
  rownames(Hessian) = NULL

  return( Hessian )
}

# 뉴튼랩슨 알고리즘
```

```r
gp_NRoptim = function(x, y){
  xx = scale(x)
  X = cbind(1, xx)

  n = dim(X)[1]
  p = dim(X)[2]

  max_iter = 1000
  eps = 1e-5

  theta = runif(p+1, 0.1, 0.2)  # (beta, tau)
  theta_new = 100

  t = 0
  while(sum((theta-theta_new)^2) > eps || t < max_iter){
    t = t + 1

    if(t != 1){  # update
      theta = theta_new
    }
    mu_new = exp(X %*% theta[1:p])  # theta[1:p] is mu
    tau_new = theta[p+1]            # theta[p+1] is tau

    D = Dvec(X, y, mu_new, tau_new)
    H = Hmat(X, y, mu_new, tau_new)

    theta_new = theta - 0.02*solve(H) %*% D

    # cat("iteration =", t, "|| theta_new =", theta_new, '\n')
    # cat("loss =", norm(theta-theta_new), '\n')
  }
  theta_new = as.vector(theta_new)

  mu_new = exp(X %*% theta[1:p])
  tau_new = theta_new[p+1]

  cov = solve(-Hmat(X, y, mu_new, tau_new))  # 공분산행렬
  SE = sqrt(diag(cov))

  out = list()
  out$designX = X
  out$beta = theta[1:p]
  out$mu = as.vector(mu_new)
  out$tau = tau_new
  out$cov = cov
  out$SE = SE
  out$Tstat = theta_new/SE
  pval = ifelse(out$Tstat > 0, - out$Tstat, out$Tstat)
  out$pvalue = pnorm(pval)*2

  return(out)
}
```

```r
##

NB_test = function(x, n, mu, tau){

  if(tau == 0) y = rpois(n, mu)   # generate the data

  if(tau > 0) y = rnbinom(n, mu = mu, size = 1/tau)

  # fit models
  ps_fit = summary(glm(y ~ x, family = poisson()))
  nbr_fit = try(summary(glm.nb(y ~ x)))

  # design matrix
  design_x = cbind(1, x)
  ps_mu_hat = exp(design_x %*% ps_fit$coefficients[,1])
  nb_mu_hat = exp(design_x %*% nbr_fit$coefficient[,1])

  # -2loglik
  logL_psfit = ps_fit$aic - 2 * 2
  logL_nbfit = nbr_fit$aic - 2 * 3

  # LRT
  LR = ifelse((logL_psfit - logL_nbfit) >= 0, logL_psfit - logL_nbfit, 0)  # nb LR test
  LR = sqrt(LR)
  if.rej = ifelse(LR > 1.645, 1, 0)     # for checking power (단측검정)
  LRT = c(LR, if.rej)

  tau_est = 1/nbr_fit$theta

  # Wald test
  wald = nbr_fit$theta/(nbr_fit$SE.theta)
  if.rej = ifelse(wald > 1.645, 1, 0)   # for checking power (단측검정)
  Wald = c(wald, if.rej)

  # Score test
  num = sum((y-ps_mu_hat)^2 - y)
  den = sqrt(2*sum(ps_mu_hat^2))

  score = ifelse(num/den >= 0, num/den, 0)
  if.rej = ifelse(score > 1.645, 1, 0) # for checking power (단측검정)
  Score = c(score, if.rej)

  # adjusted Score test
  W = diag(c(ps_mu_hat))
  H = sqrt(W) %*% design_x %*% solve( t(design_x) %*% W %*% design_x) %*% t(design_x) %*% sqrt(W)
  h = diag(H)

  num = sum((y-ps_mu_hat)^2 - y + h * ps_mu_hat)
  den = sqrt(2*sum(ps_mu_hat^2))
  adjscore =ifelse(num/den >= 0, num/den, 0)
  if.rej = ifelse(adjscore > 1.645, 1, 0) # for checking power (단측검정)
  adjScore = c(adjscore, if.rej)
```

```
  out = list()
  out$tau_est = tau_est
  out$LRT = LRT
  out$Wald = Wald
  out$Score = Score
  out$adjScore = adjScore

  return(out)
}


gp_test = function(x, n, mu, tau){
  if(tau == 0) y = rpois(n, mu)   # generate the data

  if(tau > 0) y = rgenpois2(n, meanpar = mu, disppar = tau)

  # fit models
  ps_fit = summary(glm(y ~ x, family = poisson()))
  gp_fit = gp_NRoptim(x, y)  # fit generalized poisson regression

  design_x = cbind(1, x)  # design matrix
  ps_mu_hat = exp(design_x %*% ps_fit$coefficients[,1])
  gp_mu_hat = gp_fit$mu

  tau_est = ifelse(gp_fit$tau >= 0, gp_fit$tau, 0)

  # -2loglik
  logL_psfit = ps_fit$aic - 2 * 2
  logL_gpfit = -2 *logL(y, gp_mu_hat, tau_est)  # -2 loglik

  # LRT
  LR = ifelse((logL_psfit - logL_gpfit) >= 0, logL_psfit - logL_gpfit, 0)  # nb LR test
  LR = sqrt(LR)
  if.rej = ifelse(LR > 1.645, 1, 0)     # for checking power (단측검정)
  LRT = c(LR, if.rej)

  # Wald test
  wald = gp_fit$tau/(gp_fit$SE[3])
  if.rej = ifelse(wald > 1.645, 1, 0)   # for checking power (단측검정)
  Wald = c(wald, if.rej)

  # Score test
  num = sum((y-ps_mu_hat)^2 - y)
  den = sqrt(2*sum(ps_mu_hat^2))

  score = ifelse(num/den >= 0, num/den, 0)
  if.rej = ifelse(score > 1.645, 1, 0) # for checking power (단측검정)
  Score = c(score, if.rej)

  # adjusted Score test
  W = diag(c(ps_mu_hat))
  H = sqrt(W) %*% design_x %*% solve( t(design_x) %*% W %*% design_x) %*% t(design_x) %*% sqrt(W)
  h = diag(H)
```

```
  num = sum((y-ps_mu_hat)^2 - y + h * ps_mu_hat)
  den = sqrt(2*sum(ps_mu_hat^2))
  adjscore =ifelse(num/den >= 0, num/den, 0)
  if.rej = ifelse(adjscore > 1.645, 1, 0) # for checking power (단측검정)
  adjScore = c(adjscore, if.rej)

  out = list()
  out$tau_est = tau_est
  out$LRT = LRT
  out$Wald = Wald
  out$Score = Score
  out$adjScore = adjScore

  return(out)
}
```

모의실험 결과

```
N = c(50, 100, 200)
beta = c(1, 1)          # true coefficient
tau_grid = seq(0.1, 1, length.out = 10)
N_rep = 1000            # 반복 수

for(n in N){
  p = 2

  result = list()         # tau별로 저장할 공간
  tau_grid = seq(0, 0.1, 0.02)

  t = 0
  for(tau in tau_grid){
    t = t + 1

    tau_est_nb = tau_est_gp = rep(0, N_rep)
    LRT_nb = Wald_nb = matrix(0, nrow = N_rep, ncol = 2)
    LRT_gp = Wald_gp = matrix(0, nrow = N_rep, ncol = 2) # 반복별로 각 검정결과를 저장할 공간
    Score = adjScore = matrix(0, nrow = N_rep, ncol = 2)

    for(r in 1:N_rep){
      set.seed(r)
      x = runif(n, 0, 1)
      xx = cbind(1, x)
      mu = exp(xx %*% beta)
      nb_result = NB_test(x, n, mu, tau)
      gp_result = gp_test(x, n, mu, tau)

      tau_est_nb[r] = nb_result$tau_est
      tau_est_gp[r] = gp_result$tau_est

      LRT_nb[r,] = nb_result$LRT
      LRT_gp[r,] = gp_result$LRT

      Wald_nb[r,] = nb_result$Wald
      Wald_gp[r,] = gp_result$Wald
```

```
      Score[r,] = nb_result$Score
      adjScore[r,] = gp_result$adjScore
    }

    out = list()
    out$tau_est_nb = tau_est_nb
    out$tau_est_gp = tau_est_gp

    out$LRT_nb = LRT_nb
    out$LRT_gp = LRT_gp

    out$Wald_nb = Wald_nb
    out$Wald_gp = Wald_gp

    out$Score = Score
    out$adjScore = adjScore
    result[[t]] = out
  }

  Score = sapply(1:length(tau_grid), function(r) mean(result[[r]]$Score[,2]))
  Score_adj = sapply(1:length(tau_grid), function(r) mean(result[[r]]$adjScore[,2]))
  LRT_nb = sapply(1:length(tau_grid), function(r) mean(result[[r]]$LRT_nb[,2]))
  Wald_nb = sapply(1:length(tau_grid), function(r) mean(result[[r]]$Wald_nb[,2]))
  LRT_gp = sapply(1:length(tau_grid), function(r) mean(result[[r]]$LRT_gp[,2]))
  Wald_gp = sapply(1:length(tau_grid), function(r) mean(result[[r]]$Wald_gp[,2]))

  view = data.frame('n' = n,
                    'tau' = tau_grid,
                    'Score' = Score,
                    'Score_adj' = Score_adj,
                    'LRT_nb' = LRT_nb,
                    'Wald_nb' = Wald_nb,
                    'LRT_gp' = LRT_gp,
                    'Wald_gp' = Wald_gp
  )
  kable(view) %>% print
}
```

```
##
##
## |  n|  tau| Score| Score_adj| LRT_nb| Wald_nb| LRT_gp| Wald_gp|
## |--:|----:|-----:|---------:|------:|-------:|------:|-------:|
## | 50| 0.00| 0.041|     0.062|  0.026|   0.008|  0.036|   0.016|
## | 50| 0.02| 0.089|     0.261|  0.066|   0.021|  0.181|   0.114|
## | 50| 0.04| 0.207|     0.566|  0.166|   0.059|  0.444|   0.318|
## | 50| 0.06| 0.338|     0.776|  0.280|   0.121|  0.701|   0.601|
## | 50| 0.08| 0.475|     0.913|  0.421|   0.229|  0.864|   0.787|
## | 50| 0.10| 0.590|     0.968|  0.541|   0.342|  0.948|   0.915|
##
##
## |   n|  tau| Score| Score_adj| LRT_nb| Wald_nb| LRT_gp| Wald_gp|
## |---:|----:|-----:|---------:|------:|-------:|------:|-------:|
## | 100| 0.00| 0.051|     0.055|  0.042|   0.021|  0.035|   0.017|
```

```
## | 100| 0.02| 0.158|     0.420|  0.126|   0.067|  0.337|   0.261|
## | 100| 0.04| 0.335|     0.821|  0.301|   0.179|  0.763|   0.687|
## | 100| 0.06| 0.536|     0.968|  0.493|   0.360|  0.949|   0.930|
## | 100| 0.08| 0.697|     0.995|  0.673|   0.543|  0.993|   0.989|
## | 100| 0.10| 0.809|     1.000|  0.783|   0.693|  1.000|   0.999|
##
##
## |   n|  tau| Score| Score_adj| LRT_nb| Wald_nb| LRT_gp| Wald_gp|
## |---:|----:|-----:|---------:|------:|-------:|------:|-------:|
## | 200| 0.00| 0.052|     0.059|  0.040|   0.024|  0.040|   0.030|
## | 200| 0.02| 0.252|     0.618|  0.223|   0.154|  0.556|   0.485|
## | 200| 0.04| 0.557|     0.967|  0.510|   0.421|  0.946|   0.937|
## | 200| 0.06| 0.791|     0.998|  0.766|   0.700|  0.998|   0.998|
## | 200| 0.08| 0.930|     1.000|  0.921|   0.887|  1.000|   1.000|
## | 200| 0.10| 0.983|     1.000|  0.982|   0.965|  1.000|   1.000|
```

과대산포가 없는 $\hat{\tau} = 0$의 경우, LR 혹은 Wald검정은 추정된 유의수준이 명목 유의수준 0.05보다 작은 경향을 보인다. 이를 통해 과대산포가 존재하는 $\hat{\tau} > 0$에서도 검정력을 과소추정 할 것으로 예상된다. 스코어 검정은 0.05에 가깝게 추정하고 있다. 그리고 LR검정보다는 Wald검정이 더 유의수준을 과소추정하는 경향이 보인다.