homework 1

Jieun Shin

2022-10-10

1. 과산포의 표준오차 실험

시뮬레이션 데이터는 최민옥(2017)의 모의실험 디자인으로 한다. 과대산포를 허용하는 음이항 모형에서 반응변수를 생성하고 이를 포아송회귀모형 및 음이항 회귀모형에서 추정한 후, 회귀계수 추정량 및 회귀계수 추정량의 추정량의 표준오차를 계산하기로 한다. 또한 회귀계수에 대한 가설검정을 실시한다.

- $E(Y_i) = \mu_i = \exp(\beta_0 + \beta_1 x_i)$와 τ (산포모수)를 기반으로 함.
- 설명변수 x_i는 $\text{Unif}(0, 1)$에서 생성함.
- 회귀계수 β_0와 β_1의 참값은 각각 1.2와 0.5로 설정함. 이를 기반으로 μ_i의 값을 계산함.
- 반응변수 Y_i는 $NB(\mu_i, \tau)$에서 생성함. 이때 산포모수의 값은 산포모수의 효과를 알아보기 위하여 0에서 1까지 0.1단위로 움직임.
- 표본 수 n은 200과 500을 사용함.
- 각 모수의 값에서 총 1000번의 반복을 실시함. 각 반복에서는 포아송과 음이항 회귀모형에서의 회귀계수에 대한 추정 및 가설검정을 실시하였음.

```
rm(list=ls())
# 시뮬레이션 데이터 생성
N = c(200, 500)

for(n in N){
p = 2
beta = c(1.2, 0.5)      # true coefficient

N_rep = 1000            # 반복 수
result = list()         # tau별로 저장할 공간
tau_grid = seq(0.1, 1, length.out = 10)

t = 0
for(tau in tau_grid){
  t = t + 1
  ps_theta = matrix(0, p, N_rep)  # 반복별로 theta_hat을 저장할 공간 (pois)
  ps_wald = matrix(0, 3, N_rep)     # 반복별로 wald를 저장할 공간 (pois)
  nb_theta = matrix(0, p+1, N_rep)  # 반복별로 theta_hat을 저장할 공간 (nbr)
  nb_wald = matrix(0, 3, N_rep)     # 반복별로 wald를 저장할 공간 (nbr)

    for(r in 1:N_rep){
      set.seed(r)
      x = runif(n, 0, 1)
      design_x = cbind(rep(1, n), x)  # design matrix
      mu = exp(design_x %*% beta)
      y = rnbinom(n, mu = mu, size = 1/tau)
```

```
      # fit poisson regression
      psr_fit = summary(glm(y ~ x, family = poisson()))
      ps_theta[,r] = c(psr_fit$coefficients[,1])                  # (beta0, beta1) 추정
      mu_hat = exp(design_x %*% ps_theta[,r])

      ps_wald[1,r] = psr_fit$coefficients[2,2]                # beta1 standard error estimate
      ps_wald[2,r] = psr_fit$coefficients[2,3]                # wald statistics
      CI = ps_theta[2, r] + c(-1, 1) * 1.96 * ps_wald[1,r]    # if beta1 is rejected
      ps_wald[3,r] = CI[1] <= beta[2] & beta[2] <= CI[2]      # If beta under H0 is in CI, then H0 is no

      # fit negative binomial regression
      nbr_fit = summary(glm.nb(y ~ x))

      nb_theta[,r] = c(nbr_fit$coefficients[,1], 1/nbr_fit$theta) # (beta0, beta1, tau) 추정
      mu_hat = exp(design_x %*% nb_theta[1:2, r])

      nb_wald[1,r] = nbr_fit$coefficients[2,2]                # beta1 standard error estimate
      nb_wald[2,r] = nbr_fit$coefficients[2,3]                # wald statistics
      CI = nb_theta[2, r] + c(-1, 1) * 1.96 * nb_wald[1,r]    # if beta1 is rejected
      nb_wald[3,r] = CI[1] <= beta[2] & beta[2] <= CI[2]      # If beta under H0 is in CI, then H0 is no
    }

  out = list()
  out$tau = tau
  out$ps_theta = ps_theta
  out$ps_wald = ps_wald
  out$nb_theta = nb_theta
  out$nb_wald = nb_wald
  result[[t]] = out
}

ps_beta1_mean = sapply(1:length(tau_grid), function(r) mean(result[[r]]$ps_theta[2,]))  # tau별 1000개 b
ps_beta1_bias = sapply(1:length(tau_grid), function(r) mean(result[[r]]$ps_theta[2,]) - 0.5)  # tau별 1
ps_beta1_mse = sapply(1:length(tau_grid), function(r) mean((result[[r]]$ps_theta[2,] - 0.5)^2))  # tau별

nb_beta1_mean = sapply(1:length(tau_grid), function(r) mean(result[[r]]$nb_theta[2,]))  # tau별 1000개 b
nb_beta1_bias = sapply(1:length(tau_grid), function(r) mean(result[[r]]$nb_theta[2,]) - 0.5)  # tau별 1
nb_beta1_mse = sapply(1:length(tau_grid), function(r) mean((result[[r]]$nb_theta[2,] - 0.5)^2))  # tau별

cat('n' = n, '\n')

view1 = data.frame("tau" = tau_grid,
                   "ps_beta1_mean"= ps_beta1_mean,
                   "ps_beta1_bias" = ps_beta1_bias,
                   "ps_beta1_mse"= ps_beta1_mse,
                   "nb_beta1_mean"= nb_beta1_mean,
                   "nb_beta1_bias" = nb_beta1_bias,
                   "nb_beta1_mse"= nb_beta1_mse
                   )
view1

# standard error table
ps_beta1_se = sapply(1:length(tau_grid), function(r) mean(result[[r]]$ps_wald[1,]))
```

```
ps_beta1_rej = sapply(1:length(tau_grid), function(r) 1-mean(result[[r]]$ps_wald[3,]))

nb_beta1_se = sapply(1:length(tau_grid), function(r) mean(result[[r]]$nb_wald[1,]))
nb_beta1_rej = sapply(1:length(tau_grid), function(r) 1-mean(result[[r]]$nb_wald[3,]))

view2 = data.frame("tau" = tau_grid,
                   "ps_beta1_se"= ps_beta1_se,
                   "ps_beta1_rej" = ps_beta1_rej,
                   "nb_beta1_se"= nb_beta1_se,
                   "nb_beta1_rej" = nb_beta1_rej
                   )
view2
}
```

```
## 200
## 500
```

이번에는 전체 결과를 표로 정리해보자. 여기서는 τ별 1000개 β_1의 추청치 평균, bias, mse만 출력하였다. 여기서 포아송회귀, 음이항 회귀모형에서 모두 β_1의 bias가 매우 작으므로 불편추정치라고 할 수 있다. 따라서 회귀계수 추정에는 문제가 없다.

다음으로 표준오차의 평균 및 $H_0 : \beta_1 = 0.5$에 대한 검정 결과를 보자. 포아송회귀에서는 과대산포가 커질수록 회귀계수의 표준오차의 평균에 큰 변화가 없는 반면 음이항 회귀모형에서 회귀계수의 표준오차는 커지고 있다. 이로부터 포아송모형은 회귀계수를 과소추정하는 것을 알 수 있다. 이어서 검정 결과를 살펴보면 음이항 회귀에서 추정된 유의수준은 명목 유의수준 5%를 어느정도 유지하고 있지만 포아송회귀의 경우 과대추정하는 것을 확인할 수 있다. 따라서 데이터에서 과대산포를 무시하면 추정에는 문제가 없지만 표준오차 추정에는 문제가 발생하는 것을 알 수 있다.

2. 과대산포에 대한 모의실험

2-1. 3가지 검정통계량의 비교

과대산포를 허용하는 음이항모형에서 반응변수를 생성하고, 유의수준 0.05에서 과대산포에 대한 3가지 검정(LR, Wald, Score test)를 실시하고 기각 및 채택여부를 파악한 후, 추정된 유의수준과 검정력을 계산해보자. 이를 통해 3가지 검정의 소표본 성질을 알아보고자 한다.

- $E(Y_i) = \mu_i = \exp(\beta_0 + \beta_1 x_i)$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.

```
rm(list = ls())

# 시뮬레이션 데이터 생성
N = c(50, 100, 200)

for(n in N){
p = 2
beta = c(1, 1)      # true coefficient

N_rep = 1000            # 반복 수
result = list()         # tau별로 저장할 공간
```

```r
tau_grid = seq(0, 0.1, 0.02)

t = 0
for(tau in tau_grid){
  t = t + 1
  tau_hat = rep(0, N_rep)            # 반복별로 tau 추정치를 저장할 공간
  tau_LRT = tau_Wald = tau_Score = tau_adjScore = matrix(0, 2, N_rep)  # 반복별로 각 검정결과를 저장할 공간

    for(r in 1:N_rep){
      set.seed(r)
      x = runif(n, 0, 1)
      design_x = cbind(rep(1, n), x)  # design matrix
      mu = exp(design_x %*% beta)
      if(tau == 0) y = rpois(n, mu)
      if(tau > 0) y = rnbinom(n, mu = mu, size = 1/tau)

      ps_fit = summary(glm(y ~ x, family = poisson()))
      nbr_fit = summary(glm.nb(y ~ x))

      # LRT
      ps_mu_hat = exp(design_x %*% ps_fit$coefficients[,1])
      nb_mu_hat = exp(design_x %*% nbr_fit$coefficient[,1])

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

      LRT = ifelse(2*(logL_psfit - logL_nbfit) >= 0, logL_psfit - logL_nbfit, 0)
      LR = sqrt(LRT)
      if.rej = ifelse(LR > 1.645, 1, 0)     # for checking power (단측검정)

      tau_hat[r] = 1/nbr_fit$theta
      tau_LRT[,r] = c(LR, if.rej)

      # Wald test
      Wald = nbr_fit$theta/(nbr_fit$SE.theta)
      if.rej = ifelse(Wald > 1.645, 1, 0)   # for checking power (단측검정)
      tau_Wald[,r] = 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)
      tau_Score[,r] = 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)
```

```
      tau_adjScore[,r] = c(adjScore, if.rej)
    }

  out = list()
  out$tau = tau_hat
  out$tau_LRT = tau_LRT
  out$tau_Wald = tau_Wald
  out$tau_Score = tau_Score
  out$tau_adjScore = tau_adjScore
  result[[t]] = out
}

Score = sapply(1:length(tau_grid), function(r) mean(result[[r]]$tau_Score[2,]))
Score_adj = sapply(1:length(tau_grid), function(r) mean(result[[r]]$tau_adjScore[2,]))
LRT = sapply(1:length(tau_grid), function(r) mean(result[[r]]$tau_LRT[2,]))
Wald = sapply(1:length(tau_grid), function(r) mean(result[[r]]$tau_Wald[2,]))

view = data.frame('n' = n,
                  'tau' = tau_grid,
                  'Score' = Score,
                  'Score_adj' = Score_adj,
                  'LRT' = LRT,
                  'Wald' = Wald
                  )
cat('n = ', n, '\n')
print(view)
}
```

```
## n =  50
##    n  tau Score Score_adj   LRT  Wald
## 1 50 0.00 0.041     0.053 0.026 0.008
## 2 50 0.02 0.089     0.124 0.066 0.021
## 3 50 0.04 0.207     0.255 0.166 0.059
## 4 50 0.06 0.338     0.402 0.280 0.121
## 5 50 0.08 0.475     0.535 0.421 0.229
## 6 50 0.10 0.590     0.632 0.541 0.342
## n =  100
##     n  tau Score Score_adj   LRT  Wald
## 1 100 0.00 0.051     0.063 0.042 0.021
## 2 100 0.02 0.158     0.172 0.126 0.067
## 3 100 0.04 0.335     0.371 0.301 0.179
## 4 100 0.06 0.536     0.587 0.493 0.360
## 5 100 0.08 0.697     0.734 0.673 0.543
## 6 100 0.10 0.809     0.848 0.783 0.693
## n =  200
##     n  tau Score Score_adj   LRT  Wald
## 1 200 0.00 0.052     0.055 0.040 0.024
## 2 200 0.02 0.252     0.273 0.223 0.154
## 3 200 0.04 0.557     0.600 0.510 0.421
## 4 200 0.06 0.791     0.810 0.766 0.700
## 5 200 0.08 0.930     0.942 0.921 0.887
## 6 200 0.10 0.983     0.984 0.982 0.965
```

```
rm(result)
```

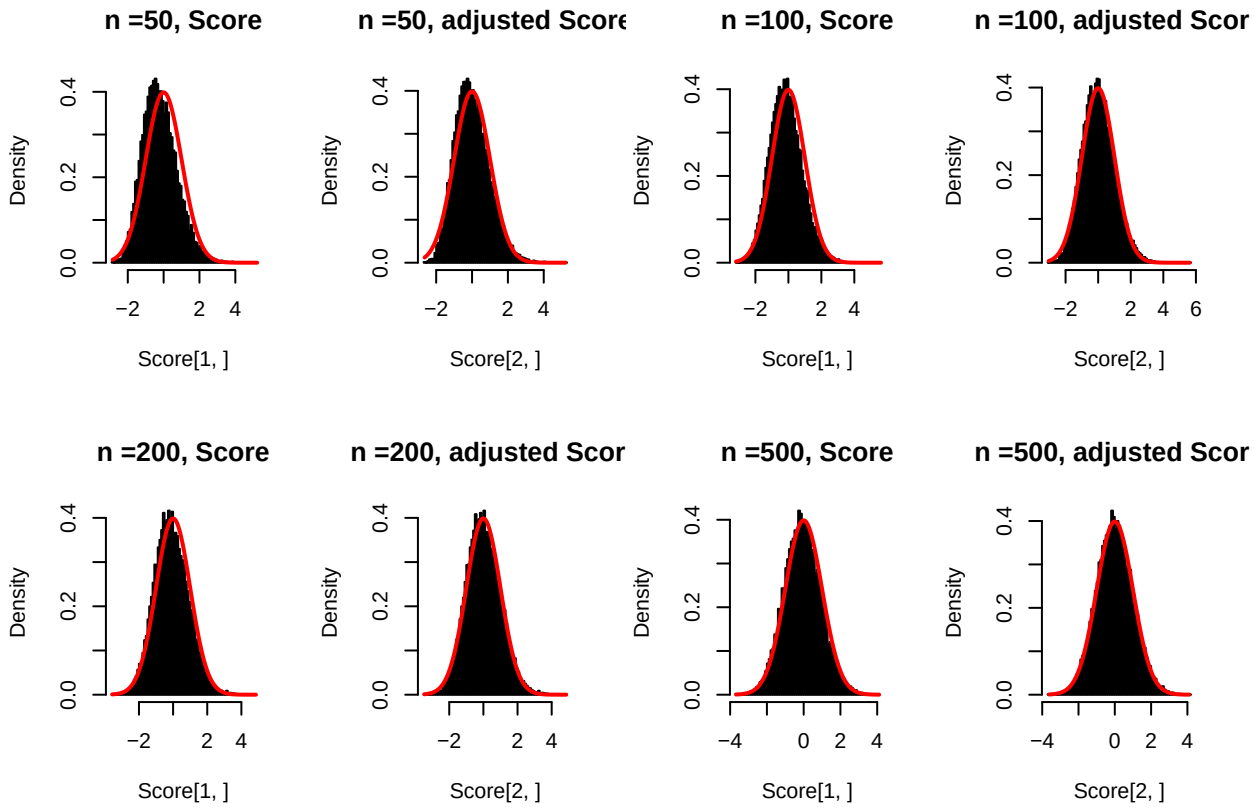
표본 크기 n과 τ에 따른 전체 결과를 보자. 각 표본크기에서 $\tau = 0$인 경우 추정된 유의수준을, $\tau > 0$인 경우 추정된 검정력을 나타낸다. 사전에 지정한 명목 유의수준 0.05와 비교하면, 추정된 유의수준은 LRT와 Wald검정에서 과소추정됨을 보이는 반면 Score검정에서는 비교적 0.05에 근접한 것을 볼 수 있다. 그리고 LRT검정보다 Wald검정에서 다소 높은 값을 보이고, Score검정보다 조정된 Score검정에서 다소 높은 값을 보인다. 이를 통해 검정력 또한 LRT와 Wald검정에서 과소추정될 것으로 예상된다.

2-2. 스코어 검정에서 조정계수의 효과 파악하기

- 귀무가설 하에서 ($\tau = 0$) 반응변수를 생성시킨 후, 스코어 검정 및 조정된 스코어 검정통계량의 분위수를 20,000번의 반복을 통하여 추정하고, $N(0,1)$의 분위수와 비교하여 표준정규분포 근사가 정확하게 이루어지는지 체크함.
- 스코어 검정 및 조정된 스코어 검정통계량의 히스토그램을 $N(0,1)$과 비교함.
- 유의수준 1/%, 5/%, 10/%에서 가설검정을 실시하여 추정된 유의수준이 명목 유의수준에 잘 부합하는지 체크함.

```
rm(list = ls())
par(mfrow = c(2,4))

# 시뮬레이션 데이터 생성
N = c(50, 100, 200, 500)
qval = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99)

Score_result = adjScore_result = matrix(0, 9, 4)
t = 0
for(n in N){
  t = t + 1

  p = 2
  beta = c(1, 1)      # true coefficient
  tau = 0
  N_rep = 20000       # 반복 수

  Score = sapply(1:N_rep, function(r){
    set.seed(r)
    x = runif(n, 0, 1)
    design_x = cbind(rep(1, n), x)  # design matrix
    mu = exp(design_x %*% beta)
    y = rpois(n, mu)

    ps_fit = summary(glm(y ~ x, family = poisson()))   # fit poisson regression
    ps_mu_hat = exp(design_x %*% ps_fit$coefficients[,1])

    # Score test
    num = sum((y-ps_mu_hat)^2 - y)
    den = sqrt(2*sum(ps_mu_hat^2))
    Score = num/den
    Score_test = ifelse(Score > 1.645, 1, 0) # for checking power (단측검정)

    # 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 = num/den
    adjScore_test = ifelse(adjScore > 1.645, 1, 0) # for checking power (단측검정)

    return(cbind(Score, adjScore))
  })

  hist(Score[1,], breaks = 100, main = paste0("n =", n, ", Score"), freq = FALSE)
  curve(dnorm(x), from = min(Score[1,]), to = max(Score[2,]), add = T, col = 'red', lwd = 2)

  hist(Score[2,], breaks = 100, main = paste0("n =", n, ", adjusted Score"), freq = FALSE)
  curve(dnorm(x), from = min(Score[2,]), to = max(Score[2,]), add = T, col = 'red', lwd = 2)

  Score_qan = quantile(Score[1,], probs = qval)
  adjScore_qan = quantile(Score[2,], probs = qval)

  Score_result[,t] = Score_qan
  adjScore_result[,t] = adjScore_qan

}
```




```
cat("스코어검정 \n")
```

```
## 스코어검정
```

```r
Score_result = cbind(Score_result, qnorm(qval))
view1 = as.data.frame(Score_result)
colnames(view1) = c(N, 'N(0,1)')
rownames(view1) = qval
view1
```

```
##              50        100        200         500     N(0,1)
## 0.01 -2.0910414 -2.1755785 -2.2275202 -2.28947065 -2.3263479
## 0.05 -1.6366901 -1.6672469 -1.6503237 -1.67587216 -1.6448536
## 0.1  -1.3728102 -1.3649957 -1.3353422 -1.32283474 -1.2815516
## 0.25 -0.8889720 -0.8321162 -0.7888932 -0.74498930 -0.6744898
## 0.5  -0.2874525 -0.1938467 -0.1468523 -0.08631841  0.0000000
## 0.75  0.3966374  0.4837911  0.5514408  0.59192522  0.6744898
## 0.9   1.0553728  1.1370856  1.1999705  1.21819015  1.2815516
## 0.95  1.4942312  1.5808066  1.6059014  1.61784001  1.6448536
## 0.99  2.3909209  2.3962820  2.4390092  2.36892468  2.3263479
```

```r
cat("조정된 스코어검정 \n")
```

```
## 조정된 스코어검정
```

```r
adjScore_result = cbind(adjScore_result, qnorm(qval))
view2 = as.data.frame(adjScore_result, qnorm(qval))
colnames(view2) = c(N, 'N(0,1)')
rownames(view2) = qval
view2
```

```
##               50         100         200         500     N(0,1)
## 0.01 -1.88973614 -2.03407328 -2.12596325 -2.22545381 -2.3263479
## 0.05 -1.43601671 -1.52459292 -1.54989550 -1.61253646 -1.6448536
## 0.1  -1.17088446 -1.22342307 -1.23402881 -1.25909282 -1.2815516
## 0.25 -0.68610835 -0.68922613 -0.68842025 -0.68113284 -0.6744898
## 0.5  -0.08587617 -0.05079518 -0.04573167 -0.02244765  0.0000000
## 0.75  0.59977332  0.62604284  0.65250986  0.65575774  0.6744898
## 0.9   1.25737825  1.27874659  1.30068374  1.28214442  1.2815516
## 0.95  1.69404047  1.72391914  1.70700885  1.68151906  1.6448536
## 0.99  2.59436138  2.53861645  2.54083054  2.43226428  2.3263479
```

2-3. 각 검정의 추정된 유의수준 비교하기

```r
rm(list = ls())

# 시뮬레이션 데이터 생성
N = c(50, 100, 200, 500)
Alpha = qnorm(c(0.01, 0.05, 0.1, 0.2), lower.tail = FALSE)

Score_alpha_est = adjScore_alpha_est = matrix(0, 4, 4)
t = 0
for(n in N){
  t = t + 1

  p = 2
  beta = c(1, 1)      # true coefficient
  tau = 0
  N_rep = 1000       # 반복 수
```

```
  a = 0
  for(alpha in Alpha){
    a = a + 1
    Score = sapply(1:N_rep, function(r){
      set.seed(r)
      x = runif(n, 0, 1)
      design_x = cbind(rep(1, n), x)  # design matrix
      mu = exp(design_x %*% beta)
      y = rpois(n, mu)

      ps_fit = summary(glm(y ~ x, family = poisson()))   # fit poisson regression
      ps_mu_hat = exp(design_x %*% ps_fit$coefficients[,1])

      # Score test
      num = sum((y-ps_mu_hat)^2 - y)
      den = sqrt(2*sum(ps_mu_hat^2))
      Score = num/den
      Score_test = ifelse(Score > alpha, 1, 0) # for checking power (단측검정)

      # 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 = num/den
      adjScore_test = ifelse(adjScore > alpha, 1, 0) # for checking power (단측검정)

      return(cbind(Score_test, adjScore_test))
    })

    Score_alpha_est[t, a] = mean(Score[1,])
    adjScore_alpha_est[t, a] = mean(Score[2,])
  }
}

cat("스코어검정 \n")
```

```
## 스코어검정
```

```
view1 = as.data.frame(Score_alpha_est)
colnames(view1) = c(0.01, 0.05, 0.1, 0.2)
rownames(view1) = N
view1
```

```
##      0.01  0.05   0.1   0.2
## 50  0.012 0.041 0.067 0.126
## 100 0.019 0.051 0.083 0.154
## 200 0.019 0.052 0.083 0.172
## 500 0.009 0.043 0.080 0.165
```

```
cat("조정된 스코어검정 \n")
```

```
## 조정된 스코어검정
```

```
view2 = as.data.frame(adjScore_alpha_est)
colnames(view2) = c(0.01, 0.05, 0.1, 0.2)
rownames(view2) = N
view2
```

```
##      0.01  0.05   0.1   0.2
## 50  0.016 0.053 0.091 0.165
## 100 0.023 0.064 0.100 0.189
## 200 0.020 0.055 0.106 0.191
## 500 0.010 0.050 0.094 0.190
```

3. 일반화 포아송 회귀모형

일반화 포아송 모형을 구현하여 fitted frequency를 계산하고, 포아송 회귀모형과 음이항 회귀모형의 결과와 비교하는 것을 목표로 한다.

3-1. 모수 추정하기

일반화 포아송 회귀모형을 구현하기 위해 trip 데이터를 사용하였고, 모수 β와 τ는 뉴튼-랩슨 방법으로 추정하였다.

```
library(readxl)
```

```
## Warning: 패키지 'readxl'는 R 버전 4.1.3에서 작성되었습니다
```

```
trip = data.frame(read_excel("C:\\Users\\jieun shin\\Downloads\\trip_data.xlsx"))
y = trip$trips   # response variable
x = trip[,-1]
X = cbind(1, x)  # design matrix
rm(trip)

pgenpois = function(x, mu, tau){
  (mu/(1+tau*mu+1e-10))^x * (1+tau*x)^{x-1+1e-10} / factorial(x+1e-10) * exp(-mu*(1+tau*x)/(1+tau*mu+1e
}

logL = function(x, mu, tau){
  sum( x * log(mu/(1+tau*mu)) + (x-1) * log(1+tau*x) - mu*(1+tau*x)/(1+tau*mu) - log(x) )
}

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))
}


###
```

```r
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
NewtonRaphson = 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 = c(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$tau = tau_new
  out$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)
}
```

3-2. fitted value 비교하기

위에서 만든 함수로 일반화 포아송 회귀모형을 적합하고 그 결과를 확인한다. 이어서 세 가지 회귀분석으로부터 계수값을 추정하여 그 결과를 비교한다.

```
n = dim(x)[1]
p = dim(x)[2]
gp_fit = NewtonRaphson(as.matrix(x, n, p), y)  # fit generalized poisson regression

view1 = data.frame(param = round(c(gp_fit$beta, gp_fit$tau), 5),
                   SE = round(gp_fit$SE, 5),
                   Tstat = round(gp_fit$Tstat, 5),
                   pvalue = round(gp_fit$pvalue, 5))

view1
```

```
##      param      SE     Tstat  pvalue
## 1 -0.70613 0.09377  -7.53018 0.00000
## 2  1.47766 0.11140  13.26448 0.00000
## 3  0.31993 0.08618   3.71244 0.00021
## 4 -0.02031 0.09419  -0.21559 0.82931
## 5  0.11980 0.08868   1.35090 0.17673
## 6  3.85345 1.08927   3.53765 0.00040
## 7 -5.37036 0.52642 -10.20173 0.00000
## 8  1.30809 0.76647   1.70665 0.08789
## 9  0.40297 0.03760  10.71675 0.00000
```

```
# fitted cell frequency
fitt_freq = function(theta, type){
  X = cbind(1, scale(x))
```

```
  if(type == "genpois") mu = exp(X %*% theta[1:(p+1)]); tau = theta[p+2]
  if(type == "pois") mu = exp(X %*% theta)
  if(type == "nb") mu = exp(X %*% theta[1:(p+1)]); tau = 1/theta[p+2]

  tb = table(y)
  c_val = sort(unique(y))
  cnt = c(tb[1], tb[2], tb[3], tb[4], tb[5], tb[6], sum(tb[7:9]), sum(tb[10:12]), sum(tb[13]), sum(tb[14

  if(type == "genpois") ffreq = sapply(0:17, function(k) sum(sapply(1:n, function(j) pgenpois(k, mu[j],
  if(type == "pois") ffreq = sapply(0:17, function(k) sum(sapply(1:n, function(j) dpois(k, mu[j]))))
  if(type == "nb") ffreq = sapply(0:17, function(k) sum(sapply(1:n, function(j) dnbinom(k, mu = mu[j], s

  ffreq2 = c(ffreq[1], ffreq[2], ffreq[3], ffreq[4], ffreq[5], ffreq[6], sum(ffreq[7:9]), sum(ffreq[10:1
             sum(ffreq[13:14]), sum(ffreq[15:18]), n-sum(ffreq), n)

  view = data.frame(count = c(0, 1, 2, 3, 4, 5, '6-8', '9-11', '12-14', '15-17', '18+', 'total'),
                    freq = cnt,
                    fitted_freq = ffreq2)

  return(view)
}

ps_fit = summary(glm(y ~ as.matrix(x), family = poisson()))
nbr_fit = summary(glm.nb(y ~ as.matrix(x)))

theta_genpois = c(gp_fit$beta, gp_fit$tau)
theta_ps = ps_fit$coefficients[,1]
theta_nb = c(nbr_fit$coefficients[,1], nbr_fit$theta)

freq_genpois = fitt_freq(theta_genpois, type = "genpois")
freq_pois = fitt_freq(theta_ps, type = "pois")
freq_nb = fitt_freq(theta_nb, type = "nb")

view = data.frame(freq_genpois, fitted_pois = freq_pois$fitted_freq, fitted_nb = freq_nb$fitted_freq)
view
```

```
##    count freq fitted_freq  fitted_pois   fitted_nb
## 1      0  417  546.358803 220.26800877 468.8791172
## 2      1   68   50.988710 191.75580234 109.5065968
## 3      2   38   17.351543 109.82650833  35.4521293
## 4      3   34    8.804218  57.78527625  15.0439332
## 5      4   17    5.431173  30.89263984   7.5362378
## 6      5   13    3.747013  16.89684058   4.1850433
## 7    6-8   21    6.663591  16.53672819   5.0726966
## 8   9-11   16    3.669259   1.90319001   1.5805360
## 9  12-14    5    1.682520   0.11827735   0.5430162
## 10 15-17   15    2.386790   0.01658773   0.7281279
## 11   18+   15   11.916380  13.00014061  10.4725657
## 12 total  659  659.000000 659.00000000 659.0000000
```

```
barmatrix = rbind(freq_genpois[-12,2],
                  freq_genpois[-12,3],
                  freq_pois[-12,3],
                  freq_nb[-12,3])
```

```
rownames(barmatrix) = c("freq", "genpois", "pois", "NB")
colnames(barmatrix) = c(0, 1, 2, 3, 4, 5, '6-8', '9-11', '12-14', '15-17', '18+')
barplot(barmatrix, beside = T, legend.text = TRUE)
```