```r
getwd()
meat = read.csv("meatstoredat.txt", sep = " ")

getest = function(x, y, beta_ini, ..., eps = 10^(-10)){
  beta = beta_ini
  iter = 0

  while(TRUE){
    iter = iter + 1
    eta = x %*% beta
    mu = 1 - 1 / (1 + exp(eta))
    #mu = ifelse(abs(mu) > eps, mu, eps)
    #mu = ifelse(abs(1 - mu) > eps, mu, 1 - eps)

    eta_mu = 1 / mu + 1 / (1 - mu)
    w = 1 / (eta_mu^2 * mu * (1 - mu))

    W = diag(c(w))
    z = (y - mu) * eta_mu

    beta_2 = beta + solve(t(x) %*% W %*% x, t(x) %*% W %*% z)
    #xi_2 = solve(t(x) %*% W %*% x, t(x) %*% W %*% (x %*% xi + z))
    if(any(is.na(beta_2))){
      stop("Fatal error:: NA's generated")
    }

    if(norm(beta - beta_2) < 10^(-10)){
      beta = beta_2
      break
    }
    else{
      beta =  beta_2
    }
  }
  eta = x %*% beta
  mu = 1 - 1 / (1 + exp(eta))
  Vmu = mu * (1 - mu)
  phi = mean((y - mu)^2 / Vmu)

  return(list(beta = beta, phi = phi, iteration = iter))
}


getest2 = function(x, y, gamma_ini, phi_ini, ..., eps = 10^(-10)){
  iter = 0
  theta = c(gamma_ini, phi_ini)

  while(TRUE){
    gamma = theta[1:2]
    phi = theta[3]
    iter = iter + 1
    eta = x %*% gamma
    mu = 1 - 1 / (1 + exp(eta))
```

```
#mu = ifelse(abs(mu) > eps, mu, eps)
#mu = ifelse(abs(1 - mu) > eps, mu, 1 - eps)

eta_mu = 1 / mu + 1 / (1 - mu)

alpha = (1 / phi - 1) * mu
beta = (1 / phi - 1) * (1 - mu)

l_alpha = log(y) +  digamma(alpha + beta) - digamma(alpha)
l_beta = log(1 - y) + digamma(alpha + beta) - digamma(beta)

mu_eta = mu * (1 - mu)
l_gamma = (1 / phi - 1) * t(x) %*% ((l_alpha - l_beta) * mu_eta)
l_phi = -sum(l_alpha * mu / phi^2 + l_beta * (1 - mu) / phi^2)

l_alpha2 = trigamma(alpha + beta) - trigamma(alpha)
l_beta2 = trigamma(alpha + beta) - trigamma(beta)

l_theta = c(l_gamma, l_phi)

mu_eta2 = mu * (1 - mu) * (1 - 2 * mu)
alpha_mu = 1 / phi - 1

w11 = alpha_mu^2 * ((l_alpha2 + l_beta2) * mu_eta^2) + alpha_mu *
  (l_alpha - l_beta) * mu_eta2
l_gamma2 = t(x) %*% diag(c(w11)) %*% x
l_phi2 = sum(l_alpha2 * mu^2 / phi^4 + l_alpha * 2 * mu / phi^3 +
  l_beta2 * (1 - mu)^2 / phi^4 + l_beta * 2 * (1 - mu) / phi^3)
l_gammaphi = t(x) %*% (alpha_mu * (-l_alpha2*mu + l_beta2*(1 - mu)) *
                         mu_eta / phi^2 + (-l_alpha + l_beta) * mu_eta / phi^2)

l_theta2 = rbind(cbind(l_gamma2, l_gammaphi), c(l_gammaphi, l_phi2))

#print(iter)
#print(l_theta)
#print(l_theta2)

theta_2 = c(gamma, phi) - solve(l_theta2, l_theta)

logliktmp = loglik(gamma, phi, x, y, length(y))
logliktmp2 = loglik(theta_2[1:2], theta_2[3], x, y, length(y))

if(any(is.na(theta_2))){
  stop("Fatal error:: NA's generated")
}

if(norm(theta - theta_2, "2") < 10^(-10) & norm(logliktmp - logliktmp2, "2") < 10^(-10)){
  theta = theta_2
  break
}
else{
  theta =  theta_2
}
```

```
  }
  gamma = theta_2[1:2]
  phi = theta_2[3]
  logliktmp = loglik(gamma, phi, x, y, length(y))

  return(list(gamma = gamma, phi = phi, iteration = iter, gradient = l_theta,
              loglik = logliktmp))
}

res = getest2(x, y, gamma, phi)
loglik(res$gamma, res$phi, x, y, n)
eta = x %*% beta
mu = 1 - 1 / (1 + exp(eta))


glmfit = function(score, time){
  model_ini = glm(score ~ time, family = binomial("logit"))
  gamma_ini = model_ini$coefficients
  muhat = model_ini$fitted.values
  Vmuhat = muhat * (1 - muhat)
  phi_ini = mean((y - muhat)^2 / Vmuhat)
  x = cbind(1, time)
  y = score
  y = ifelse(y == 1, 1 - 10^(-5), y)
  y = ifelse(y == 0, 10^(-5), y)
  n = length(y)
  res = getest2(x, y, gamma_ini, phi_ini)
  gamma = res$gamma
  phi = res$phi

  #loglik(gamma, phi, x, y, n)

  optim_res = optim(function(pars) return(-loglik(pars[1:2], pars[3], x,
                                                  , length(y))), par = c(beta, phi),
                    method = "L-BFGS-B", upper = c(Inf, Inf, 1),
                    lower = c(-Inf, -Inf, 0))

  #gamma = optim_res$par[c(1, 2)]
  #phi = optim_res$par[3]
  return(list(gamma = gamma, phi = phi, loglik = res$loglik, iter = res$iteration,
              gradient = res$gradient))
}



loglik = function(gamma, phi, x, y, n){
  eta = x %*% gamma
  mu = 1 - 1 / (1 + exp(eta))
  alpha = (1 / phi - 1) * mu
  beta = (1 / phi - 1) * (1 - mu)
  #print(sapply(1:n, function(i) dbeta(y[i], alpha[i], beta[i], log = TRUE)))
  if(any(mu == 1)) return(Inf)
  else return(sum(sapply(1:n, function(i) dbeta(y[i], alpha[i], beta[i], log = TRUE))))
```

```r
}

# Prob2
fit1 = glmfit(meat$score1, meat$time)
fit2 = glmfit(meat$score2, meat$time)
fit3 = glmfit(c(meat$score1, meat$score2), c(meat$time, meat$time))

# Prob3
2 * (fit1$loglik + fit2$loglik - fit3$loglik)
qchisq(0.95, 3) # Reject H0

# Prob4
gamma1 = fit1$gamma
phi1 = fit1$phi
eta1 = sum(c(1, 14) * gamma1)
mu1 = 1 - 1 / (1 + exp(eta1))

alpha1 = (1 / phi1 - 1) * mu1
beta1 = (1 / phi1 - 1) * (1 - mu1)

gamma2 = fit2$gamma
phi2 = fit2$phi
eta2 = sum(c(1, 14) * gamma2)
mu2 = 1 - 1 / (1 + exp(eta2))

alpha2 = (1 / phi2 - 1) * mu2
beta2 = (1 / phi2 - 1) * (1 - mu2)

integrate(function(y2) return(pbeta(y2, alpha1, beta1) *
                                dbeta(y2, alpha2, beta2)), lower = 0, upper = 1)

# (c)
B = 100000
sumres = 0
for(simnum in 1:B){
  sumres = sumres + ifelse(rbeta(1, alpha1, beta1) < rbeta(1, alpha2, beta2), 1, 0)
}
sumres / B

# (d)
1 / 4 * (1.96 / 0.001)^2
```