```r
getwd()
Series1 = read.csv("Series1.csv")
Series2 = read.csv("Series2.csv")

#Functions that are used in this assignment
getest = function(x, y, m, beta_ini, lambda_ini, ..., eps = 10^(-10)){
  beta = beta_ini
  lambda = lambda_ini
  xi = c(beta, lambda)
  iter = 0

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

    eta_mu = lambda / (1 - mu) / (1 - (1 - mu)^lambda)
    w = m / (eta_mu^2 * mu * (1 - mu))
    eta_lambda = -log(1 - mu) / (1 - (1 - mu)^lambda) - 1 / lambda

    X_A = cbind(x, -eta_lambda)
    W = diag(c(w))
    z = (y - mu) * eta_mu

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

    if(norm(xi - xi_2) < 10^(-10))
      break
    else{
      beta = c(xi_2)[1:2]
      lambda = c(xi_2)[3]
      xi = c(beta, lambda)
    }
  }
  return(list(beta = c(xi_2)[1:2], lambda = c(xi_2)[3], iteration = iter))
}

#Series = Series2
#link_ini = "cloglog"

glmfit = function(Series, link_ini){
  model_ini = glm(deathrate ~ logCS2, data = Series, family = binomial(link_ini), weights = Total)
  beta_ini = model_ini$coefficients
  if(link_ini == "logit") lambda_ini = 1
  if(link_ini == "cloglog") lambda_ini = 0.001

  x = cbind(1, Series$logCS2)
  y = Series$deathrate
  m = Series$Total

  eps = 10^(-10)

  res = getest(x, y, m, beta_ini, lambda_ini)
  beta = res$beta
  lambda = res$lambda

  model_logit = glm(deathrate ~ logCS2, data = Series, family = binomial("logit"), weights = Total)
  chi2statistics = 2 * (loglik(lambda, beta, x, y, m)- loglik(1, model_logit$coefficients, x, y, m))

  return(list(beta = beta, lambda = lambda, loglik = loglik(lambda, beta, x, y, m),
              loglik_logit = loglik(1, model_logit$coefficients, x, y, m), chi2 = chi2statistics,
              iter = res$iteration))
}

loglik = function(lambda, beta, x, y, m){
  eta = x %*% beta
  mu = 1 - 1 / (lambda * exp(eta) + 1)^(1 / lambda)
  if(any(mu == 1)) return(Inf)
  else return(sum(m * (y * log(mu / (1 - mu)) + log(1 - mu))))
}

# Prob 1
fit1 = glmfit(Series1, "logit")
#glmfit(Series2, "logit")
fit2 = glmfit(Series2, "cloglog")
qchisq(0.95, 1) # Series1: do not reject H0; Series2: reject H0

# Prob 2
Series_combined = rbind(Series1, Series2)
fit3 = glmfit(Series_combined, "cloglog")
2 * (fit1$loglik + fit2$loglik - fit3$loglik)
qchisq(0.95, 3) # Do not reject H0

# Prob 3
2 * 6 - 2 * (fit1$loglik + fit2$loglik)
2 * 3 - 2 * fit3$loglik
2 * 4 - 2 * (fit1$loglik_logit + fit2$loglik_logit)
2 * 2 - 2 * fit3$loglik_logit
#Series_combined; estimated link

beta = fit3$beta
lambda = fit3$lambda
x_vec = seq(from = 3.8, to = 4.4, length = 100)
eta_vec = beta[1] + x_vec * beta[2]
mu_vec = 1 - 1 / (lambda * exp(eta_vec) + 1)^(1 / lambda)

plot(mu_vec ~ x_vec, type = "l", xlab = "logCS2", ylab = "estimated probability",
     main = "Estimated response curve")
sprintf("eta = %g + %g * x; lambda = %g", beta[1], beta[2], lambda)
```

```r
mu_x = - beta[1] / beta[2]
sigma_x = 1 / beta[2]
G_vec = 1 - 1 / (lambda * exp((x_vec - mu_x) / sigma_x) + 1)^(1 / lambda)

plot(G_vec ~ x_vec, type = "l", xlab = "logCS2", ylab = "estimated probability",
     main = "Estimated tolerance distribution")

# Prob 4
x = Series_combined$logCS2
y = Series_combined$deathrate
m = Series_combined$Total
eta = beta[1] + x * beta[2]
mu = 1 - 1 / (lambda * exp(eta) + 1)^(1 / lambda)

sum(2 * m * (y * log(y / mu) + (1 - y) * log((1 - y) / (1 - mu))), na.rm = TRUE)
qchisq(0.95, 13 - 3) # Do not reject H0

di = 2 * m * (y * log(y / mu) + (1 - y) * log((1 - y) / (1 - mu)))
plot(y = sqrt(di[!is.na(di)]) * sign(y[!is.na(di)] - mu[!is.na(di)]), x = y[!is.na(di)],
     ylab = "Deviance Residual", xlab = "Fitted Value", main = "residual plot")
```