04\_question4

44

2022-06-02

# Q4

You have received data on patients with cardiac monitoring devices that detect atrial fibrillation events. In reporting the counts of atrial fibrillation events within one month, for some patients (Group A) the number of events is available while for the rest (Group B) you only have a binary indicator of whether the count was non-zero.

There is no information in the data set that will let you know whether the value presented is a count or a binary indicator (i.e., data from groups A and B are mixed together).

The data is available in ATFIB.txt and contains the following columns: - id: unique patient identifier - outcome: observed outcome (i.e., the direct actual count of atrial fibrillation events or the binary indicator of a non-zero count)

## (a)

Given what you know about the data collection, you realize that you could consider the data to arise from a Poisson random variable with mean for which you observe either directly (Group A), or the (Group B).

Which group each individual belongs to can be considered missing data.

Define a latent variable , independent of , as a binary indicator of whether is observed directly (i.e., indicates an observation is from Group A and indicates an observation is from Group B).

The observed outcome can then be defined in terms of , and .

Write out the likelihood for:

### i. The observed data (X).

Hint: Consider the cases - (i) X = 0: there is no event - (ii) X = 1: there is or events - (iii) X = k, k > 1: there are k events

The mixture of degenerate distribution at zero with probability and Poisson distribution with probability ; so the Poisson mixed with two point masses at 0 with and at with ; .

### ii. The complete data .

Hint: Consider the cases for when (i) Z = 1,Y = k, (ii) Z = 0, Y = 0, (iii) Z = 0, Y = k, k>0

When , we do not have the information of exact , but a Bernoulli conditional on the probability of .

## (b) Numerically optimize the observed data log-likelihood using “optim” in R,

to obtain the estimates, standard errors, and 95% confidence intervals for and .

Provide non-technical interpretations for these parameters. Comment on how you think these results would differ if you had data on , and the reason for those differences.

library(readr)  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✔ ggplot2 3.3.6 ✔ dplyr 1.0.9  
## ✔ tibble 3.1.7 ✔ stringr 1.4.0  
## ✔ tidyr 1.2.0 ✔ forcats 0.5.1  
## ✔ purrr 0.3.4

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

data\_4 <- read\_table("data/ATFIB.txt") %>%   
 janitor::clean\_names()

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## `"id"` = col\_double(),  
## `"outcome"` = col\_double()  
## )

#' Title Loglikelihood for inflated mixture model with imcomplete dataset  
#' @param lambda `numeric` a positive value as Poisson distribution parameter  
#' @param p `numeric` a positive value in [0, 1] as Bernoulli distribution parameter  
#' @param datay `data.frame` the observed dataset   
#' @return `neg\_logL` the negative log-likelihood  
#' @export  
get\_logL\_inc <- function(parameter,  
 datay) {  
 ## need to argue why lambda   
 ## and p are separated   
 p <- parameter[1] %>% as.numeric()  
 lambda <- parameter[2] %>% as.numeric()  
   
 ## subset the dataset by Y  
 y0 <- datay %>% filter(outcome == 0)  
 n0 <- nrow(y0)  
 y1 <- datay %>% filter(outcome == 1)  
 n1 <- nrow(y1)  
 yk <- datay %>% filter(outcome > 1)  
   
 ## loglikelihood for Y==0  
 logL0 <- - lambda \* n0   
 ## loglikelihood for Y==1  
 logL1 <- (log((1 - exp(-lambda)) \* (1 - p) + p \* lambda \* exp(-lambda))) \* n1  
 ## for loop to calculate Y==k  
 logLk <- 0  
 for (yi in yk$outcome) {  
 logLk <- logLk + log(p) + yi \* log(lambda) - lambda - log(gamma(yi + 1))  
 }  
 (neg\_logL <- -(logL0 + logL1 + logLk))  
 return(neg\_logL)  
}  
  
p\_hat <- optim(par = c(0.1, 2),  
 fn = get\_logL\_inc,  
 lower = c(0, 1),  
 upper = c(1, Inf),  
 datay = data\_4,  
 hessian = T)

## Warning in optim(par = c(0.1, 2), fn = get\_logL\_inc, lower = c(0, 1), upper =  
## c(1, : bounds can only be used with method L-BFGS-B (or Brent)

(par\_mean <- p\_hat$par)

## [1] 0.8440276 1.7547662

(par\_sd <- solve(p\_hat$hessian) %>% diag() %>% sqrt())

## [1] 0.06447762 0.10671729

par\_mean + 1.96 \* par\_sd

## [1] 0.9704038 1.9639321

par\_mean - 1.96 \* par\_sd

## [1] 0.7176515 1.5456003

## (c) You begin to suspect that maybe the data was actually collected consistently, and only represents counts that arise from a distribution.

Perform an appropriate statistical test to formally test this assumption for your data. State the null and alternative hypotheses for your test, and the asymptotic distribution of the test statistic under the null. Present your conclusions based on this test.

The Poisson model can be regarded as a special nested case of the mixed model in (a); hence we can use likelihood ratio test to test the hypothesis.

#' Title Negative likelihood for Poisson regression  
#' @param lambda `numeric` a positive value as Poisson distribution parameter  
#' @param datay `data.frame` the observed dataset   
#' @return `-logLk` the negative log-likelihood  
#' @export  
get\_logL\_pois <- function(lambda,  
 datay) {  
 logLk <- 0  
 for (yi in data\_4$outcome) {  
 logLk <- logLk + yi \* log(lambda) - lambda - log(gamma(yi + 1))  
 }  
 return(-logLk)  
}  
  
lambda\_pois <- optim(par = 1,   
 fn = get\_logL\_pois,  
 method = "Brent",  
 lower = 0,  
 upper = 6,  
 datay = data\_4)  
lambda\_pois$par

## [1] 1.612

logL\_mix <- get\_logL\_inc(c(1.000, 1.868),  
 datay = data\_4)  
  
  
logL\_poisson <- get\_logL\_pois(lambda = 1.612,  
 datay = data\_4)  
  
# logL\_mix; logL\_poisson  
  
2 \* abs(logL\_mix - logL\_poisson) %>%   
 pchisq(df = 1, lower.tail = FALSE)

## [1] 0.06391618

## (d) Design and conduct a simulation study that evaluates the efficiency of estimation using the incomplete data versus the complete data (i.e., the incomplete data augmented with a column for ). Specifically, report bias and efficiency for estimating . Fix . Vary sample size and to generate data from 6 settings. Simulate a reasonable number of replicates.

* Two random variables are simulated as with method and with method, with
* is the half width
* Let and denote the amount of computational effort required to produce one sample of and , respectively.
* Then the total effort expended by and , respectively, to achieve a half width
* is more efficient than if . Note that if and only if .
* is used to qualify and

### i. Submit commented code that describes (i) data generation process, (ii) estimation, and (iii) computation of metrics.

lambda <- 1  
p <- 0.8  
N <- 100  
  
#' Title Simulate data for mixture model  
#'  
#' @param lambda `numeric` parameter for Poisson as Y|Z==1 distribution  
#' @param p `numeric` parameter for Bernoulli as Z's distribution  
#' @param N `integer` for sample size  
#' @param seed `integer` set seed  
#' @return `data\_sim` a simulated dataset  
#' @export  
  
get\_sim <- function(lambda = 1,  
 p = 0.5,  
 N = 100,  
 seed) {  
 set.seed(seed)  
 ## number of Z == 1  
 Z1 <- rbinom(1, N, p)  
 ## number of Z == 0  
 Z0 <- N - Z1  
 ## Vector = Z  
 Z <- c(rep(1, Z1), rep(0, Z0))  
 ## Y|Z==1 Poisson(\lambda)  
 Y <- c(rpois(Z1, lambda),   
 ## Y|Z==0 Bernoulli(1 - \exp(-\lambda))  
 unlist(map(rep(1, Z0),   
 ~rbinom(1, ., (1 - exp(-lambda))))))  
 data\_sim <- data.frame(Z = Z, Y = Y)  
 return(data\_sim)  
}  
sim\_num <- 1000  
sim\_N100\_p02 <- map(1:sim\_num, ~get\_sim(seed = ., N = 100, p = 0.2))  
sim\_N100\_p05 <- map(1:sim\_num, ~get\_sim(seed = ., N = 100, p = 0.5))  
sim\_N100\_p08 <- map(1:sim\_num, ~get\_sim(seed = ., N = 100, p = 0.8))  
sim\_N1k\_p02 <- map(1:sim\_num, ~get\_sim(seed = ., N = 1000, p = 0.2))  
sim\_N1k\_p05 <- map(1:sim\_num, ~get\_sim(seed = ., N = 1000, p = 0.5))  
sim\_N1k\_p08 <- map(1:sim\_num, ~get\_sim(seed = ., N = 1000, p = 0.8))  
  
save(sim\_N100\_p02, sim\_N100\_p05, sim\_N100\_p08,  
 sim\_N1k\_p02, sim\_N1k\_p05, sim\_N1k\_p08,  
 file = "bios\_qexam\_question\_4.Rdata")

load("bios\_qexam\_question\_4.Rdata")

#' Title Loglikelihood for inflated mixture model with imcomplete dataset  
#' @param lambda `numeric` a positive value as Poisson distribution parameter  
#' @param p `numeric` a positive value in [0, 1] as Bernoulli distribution parameter  
#' @param datay `data.frame` the observed dataset   
#' @return `neg\_logL` the negative log-likelihood  
#' @export  
get\_logL\_incomplete <- function(parameter,  
 datayz) {  
 ## need to argue why lambda   
 ## and p are seperated   
 p <- parameter[1] %>% as.numeric()  
 lambda <- parameter[2] %>% as.numeric()  
   
 ## subset the dataset by Y  
 y0 <- datayz %>% filter(Y == 0)  
 n0 <- nrow(y0)  
 y1 <- datayz %>% filter(Y == 1)  
 n1 <- nrow(y1)  
 yk <- datayz %>% filter(Y > 1)  
   
 ## loglikelihood for Y==0  
 logL0 <- -lambda \* n0   
 ## loglikelihood for Y==1  
 logL1 <- (log((1 - exp(-lambda)) \* (1 - p) + p \* lambda \* exp(-lambda))) \* n1  
 ## for loop to calculate Y==k  
 logLk <- 0  
 for (yi in yk$Y) {  
 logLk <- logLk + log(p) + yi \* log(lambda) - lambda - log(gamma(yi + 1))  
 }  
 (neg\_logL <- -(logL0 + logL1 + logLk))  
 return(neg\_logL)  
}  
  
# get\_logL\_incomplete(c(0.5, 2), sim\_N100\_p02[[5]])

get\_logL\_complete <- function(parameter,  
 datayz) {  
 ## need to argue why lambda   
 ## and p are seperated   
 p <- parameter[1] %>% as.numeric()  
 lambda <- parameter[2] %>% as.numeric()  
   
 p1 <- exp(-lambda) \* (1 - p)  
 p2 <- (1 - exp(-lambda)) \* (1 - p)  
   
 ## subset the dataset by Y  
 z1 <- datayz %>% filter(Z == 1)  
 z0y1 <- datayz %>% filter(Y == 1, Z == 0)  
 ny1 <- nrow(z0y1)  
 z0y0 <- datayz %>% filter(Y == 0, Z == 0)  
 ny0 <- nrow(z0y0)  
   
 ## loglikelihood for Y==0 Z==0  
 logLy0 <- ny0 \* log(p1)  
 ## loglikelihood for Y==1 Z==0  
 logLy1 <- ny1 \* log(p2)  
 ## for loop to calculate Y==k  
 logLk <- 0  
 for (yi in z1$Y) {  
 logLk <- logLk + log(p) + yi \* log(lambda) - lambda - log(gamma(yi + 1))  
 }  
 (neg\_logL <- -(logLy0 + logLy1 + logLk))  
 return(neg\_logL)  
}

get\_optim <- function(datayz) {  
 ## get the estimation of parameters  
 par\_cpl <- optim(par = c(0.1, 1),   
 fn = get\_logL\_complete,  
 lower = c(0, 0),  
 upper = c(1, Inf),  
 datayz = datayz)  
 ## get the estimation of parameters  
 par\_icp <- optim(par = c(0.1, 1),   
 fn = get\_logL\_incomplete,  
 lower = c(0, 0),  
 upper = c(1, Inf),  
 datayz = datayz)  
 result <- c(unlist(par\_icp$par),   
 unlist(par\_cpl$par))   
 return(result)  
}  
  
## get the simulations estimation for all the dataset   
result\_N100\_p02 <- map\_dfc(sim\_N100\_p02, ~try(get\_optim(.))) %>% select\_if(is.numeric)  
result\_N100\_p05 <- map\_dfc(sim\_N100\_p05, ~try(get\_optim(.))) %>% select\_if(is.numeric)  
result\_N100\_p08 <- map\_dfc(sim\_N100\_p08, ~try(get\_optim(.))) %>% select\_if(is.numeric)  
  
result\_N1k\_p02 <- map\_dfc(sim\_N1k\_p02, ~try(get\_optim(.))) %>% select\_if(is.numeric)  
result\_N1k\_p05 <- map\_dfc(sim\_N1k\_p05, ~try(get\_optim(.))) %>% select\_if(is.numeric)  
result\_N1k\_p08 <- map\_dfc(sim\_N1k\_p08, ~try(get\_optim(.))) %>% select\_if(is.numeric)  
  
save(result\_N100\_p02, result\_N1k\_p02,  
 result\_N100\_p05, result\_N1k\_p05,  
 result\_N100\_p08, result\_N1k\_p08,  
 file = "bios\_qexam\_quesion\_4\_result.Rdata")

### ii. Summarize your results in a short report (2-3 paragraphs) and 1-2 figures or summary tables. Give hypotheses, conclusions, and recommendations based on what you have found.

library(matrixStats)

##   
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':  
##   
## count

load("bios\_qexam\_quesion\_4\_result.Rdata")  
  
## the mean of simulation  
rowMeans(result\_N100\_p02)

## [1] 0.2153281 1.0086950 0.2031255 1.0107250

rowMeans(result\_N100\_p05)

## [1] 0.5051958 1.0010439 0.5000312 0.9975965

rowMeans(result\_N100\_p08)

## [1] 0.7948531 1.0066151 0.7988300 0.9995712

rowMeans(result\_N1k\_p02)

## [1] 0.2008904 0.9921957 0.2003646 0.9919381

rowMeans(result\_N1k\_p05)

## [1] 0.5036064 0.9892691 0.5002055 0.9900653

rowMeans(result\_N1k\_p08)

## [1] 0.7999319 0.9927312 0.7994127 0.9921347

## the bias  
rowMeans(result\_N100\_p02) - c(0.2, 1, 0.2, 1)

## [1] 0.015328095 0.008694952 0.003125487 0.010724969

rowMeans(result\_N100\_p05) - c(0.2, 1, 0.2, 1)

## [1] 0.305195811 0.001043936 0.300031189 -0.002403478

rowMeans(result\_N100\_p08) - c(0.2, 1, 0.2, 1)

## [1] 0.5948531022 0.0066151001 0.5988299979 -0.0004288338

rowMeans(result\_N1k\_p02) - c(0.2, 1, 0.2, 1)

## [1] 0.0008903812 -0.0078042924 0.0003645704 -0.0080619160

rowMeans(result\_N1k\_p05) - c(0.2, 1, 0.2, 1)

## [1] 0.303606393 -0.010730934 0.300205548 -0.009934744

rowMeans(result\_N1k\_p08) - c(0.2, 1, 0.2, 1)

## [1] 0.599931910 -0.007268794 0.599412735 -0.007865296

## the variance  
(sd\_N100\_p02 <- matrixStats::rowSds(as.matrix(result\_N100\_p02)))

## [1] 0.08548515 0.13318408 0.03979703 0.12625713

(sd\_N100\_p05 <- matrixStats::rowSds(as.matrix(result\_N100\_p05)))

## [1] 0.13841674 0.12854886 0.04915667 0.11601187

(sd\_N100\_p08 <- matrixStats::rowSds(as.matrix(result\_N100\_p08)))

## [1] 0.14558867 0.12010363 0.04026091 0.10690733

(sd\_N1k\_p02 <- matrixStats::rowSds(as.matrix(result\_N1k\_p02)))

## [1] 0.02683510 0.03452663 0.01293083 0.03223520

(sd\_N1k\_p05 <- matrixStats::rowSds(as.matrix(result\_N1k\_p05)))

## [1] 0.04165791 0.03279650 0.01686132 0.02936739

(sd\_N1k\_p08 <- matrixStats::rowSds(as.matrix(result\_N1k\_p08)))

## [1] 0.05308074 0.03317989 0.01295586 0.02632188

## ARE efficiency   
## for p  
(sd\_N100\_p02[[1]] / sd\_N100\_p02[[3]])^2

## [1] 4.614027

(sd\_N100\_p05[[1]] / sd\_N100\_p05[[3]])^2

## [1] 7.928889

(sd\_N100\_p08[[1]] / sd\_N100\_p08[[3]])^2

## [1] 13.07639

(sd\_N1k\_p02[[1]] / sd\_N1k\_p02[[3]])^2

## [1] 4.306791

(sd\_N1k\_p05[[1]] / sd\_N1k\_p02[[3]])^2

## [1] 10.37868

(sd\_N1k\_p08[[1]] / sd\_N1k\_p08[[3]])^2

## [1] 16.78578

## for lambda  
(sd\_N100\_p02[[2]] / sd\_N100\_p02[[4]])^2

## [1] 1.112738

(sd\_N100\_p05[[2]] / sd\_N100\_p05[[4]])^2

## [1] 1.227811

(sd\_N100\_p08[[2]] / sd\_N100\_p08[[4]])^2

## [1] 1.26211

(sd\_N1k\_p02[[2]] / sd\_N1k\_p02[[4]])^2

## [1] 1.147223

(sd\_N1k\_p05[[2]] / sd\_N1k\_p02[[4]])^2

## [1] 1.035128

(sd\_N1k\_p08[[2]] / sd\_N1k\_p08[[4]])^2

## [1] 1.588971