01\_question1

44

2022-06-02

# Q1

A trial was performed to assess the impact of an intervention on the number of times a sedentary individual engages in 10 minutes of sustained moderate-to-vigorous physically active over a week.

The study recruited participants in matched pairs (matching on age), assigning the intervention to one individual within each age pair (n total pairs).

Denote each pair using the subscript and let the number of active bouts for the person in the th pair be denoted as , , with corresponding to individual who received the intervention.

Let be the age for individuals in the th pair.

It is hypothesized that

where is an unspecified function of (age).

The investigators of the current study are uninterested in .

Assume that the distribution of age in the population follows some distribution .

If the number of pairs () observed is large and is observed densely relative to the shape of then may be estimable using the likelihood of the data.

However, if n is small or is sparsely observed, estimating both and may not be possible. This question involves deriving a conditional likelihood where you condition out the nuisance parameter(s) so that is estimable in such scenarios.

You will implement a short simulation study and apply your method to data from the study referenced above.

## (a)

In the first part of this problem, we will condition out nuisance parameters () to obtain a likelihood for which depends only on the observed data.

### i.

What is the likelihood of ? Your answer will involve the marginal distribution of , .

### ii.

What is the likelihood of ? Identify the corresponding distribution.

The summation of two Poisson distribution is also a Poisson distribution.

### iii.

What is the likelihood of ? Identify the corresponding distribution.

The distribution of is , with sample size and

The distribution for the parameter is

### iv.

What is the maximum likelihood estimator of obtained from the conditional likelihood you derived in iii?

If is the MLE of , then is the MLE of ;

## (b)

Propose a method for constructing an asymptotic 95% confidence interval for based on your result from (a) iv using maximum likelihood theory (you may use the observed information here).

The MLE is asymptotically normal under regularity conditions.

Hence, an asymptotic 95% confidence interval for can be calculated using

## (c)

A statistician colleague of yours suggests constructing of a confidence interval using simulation. Specifically, your colleague proposes the following procedure:

Fix , the type-I error rate.

* Step 1 Choose candidate interval endpoints which is dense over a range of plausible endpoints for the confidence interval.
* Step 2 For each
  + Step 2a Simulate , using the distribution you derived in (a)iii.
  + Step 2b Calculate as the MLE you derived in (a)iv.
  + Step 2c Repeat (1)-(2) many times (e.g. 2000).
  + Step 2d Let , where ) and are the and percentiles of obtained from Steps 2a-2c, respectively.
* Step 3 Define a % CI for as $$ Argue that your colleague’s algorithm is a valid approach for constructing a 95% confidence interval.

It is the same as using a stochastic process to find the empirical distribution of . According to Donsker’s theroem and Glivenko-Cantelli theorem, we know that the empirical process distribution convergence to the true distribution asymptotically. Hence we can use sampling to generate a valide confidence interval.

Write a function which implements this algorithm for arbitrary: - Type-I error - Input vectors - Candidate interval endpoints - Number of simulated datasets created in step 2c

library(tidyverse)  
  
gety <- function(seed,   
 n = 15) {  
 set.seed(seed)  
 X <- runif(n, min = 0, max = 1)  
 beta <- 2 \* exp(-2 \* X)   
 gamma <- 2.2  
   
 y1 <- map(gamma \* beta, ~rpois(1, .)) %>% unlist()  
 y2 <- map(beta, ~rpois(1, .)) %>% unlist()  
   
 simy <- cbind(y1 = y1, y2 = y2)  
 return(simy)  
}  
  
## simulate 5000 y1 y2 set.  
sim\_num <- 5000  
y\_ss15 <- map(1:sim\_num, ~gety(seed = ., n = 15))  
y\_ss100 <- map(1:sim\_num, ~gety(seed = ., n = 100))  
save(y\_ss15, y\_ss100, file = "bios\_qexam\_data\_1.Rdata")  
  
y1 <- map(y\_ss15, ~mean(.[, 1])) %>% unlist()  
y2 <- map(y\_ss15, ~mean(.[, 2])) %>% unlist()  
mean(y1); mean(y2); mean(y1) / mean(y2); sqrt((mean(y1) + mean(y2))^3 \* mean(y2) / (mean(y1)^3 \* 15))  
mean(y1) / mean(y2) + 1.96 \* sqrt((mean(y1) + mean(y2))^3 \* mean(y2) / (mean(y1)^3 \* 15))  
mean(y1) / mean(y2) - 1.96 \* sqrt((mean(y1) + mean(y2))^3 \* mean(y2) / (mean(y1)^3 \* 15))  
power.t.test(n = 15, delta = 2.2 -1, sd = 0.421, sig.level = 0.05)  
  
y1 <- map(y\_ss100, ~mean(.[, 1])) %>% unlist()  
y2 <- map(y\_ss100, ~mean(.[, 2])) %>% unlist()  
mean(y1); mean(y2); mean(y1) / mean(y2); sqrt((mean(y1) + mean(y2))^3 \* mean(y2) / (mean(y1)^3 \* 100))  
mean(y1) / mean(y2) + 1.96 \* sqrt((mean(y1) + mean(y2))^3 \* mean(y2) / (mean(y1)^3 \* 100))  
mean(y1) / mean(y2) - 1.96 \* sqrt((mean(y1) + mean(y2))^3 \* mean(y2) / (mean(y1)^3 \* 100))  
power.t.test(n = 100, delta = 1.2, sd = 0.163, sig.level = 0.05)

library(tidyverse)

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

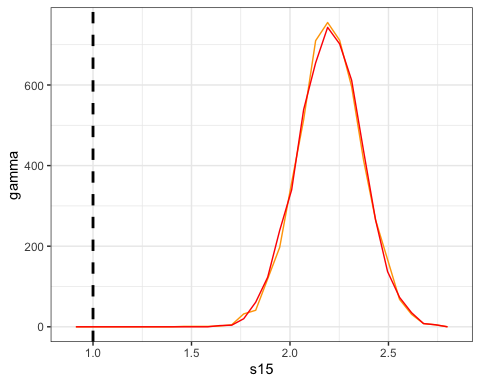
## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.7 ✔ dplyr 1.0.9  
## ✔ tidyr 1.2.0 ✔ stringr 1.4.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.1

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

s15 <- rnorm(n = 5000, mean = 2.2, sd = 0.163)  
s100 <- rnorm(n = 5000, mean = 2.2, sd = 0.163)  
ggplot() +  
 geom\_freqpoly(aes(s15), color = "orange") +  
 geom\_freqpoly(aes(s100), color = "red") +  
 geom\_vline(xintercept = 1,  
 linetype="dashed",   
 color = "black",   
 size = 1) +  
 ylab("gamma") +   
 theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



load("bios\_qexam\_data\_1.Rdata")  
  
## function for question 1d  
get\_gamma <- function(y,  
 alpha = 0.05,  
 gamma\_min = 0,  
 gamma\_max = 5.5,  
 gamma\_step = 0.01,  
 N = 2000) {  
 ## sum of y1 y2  
 y1 <- y[, 1]  
 y2 <- y[, 2]  
 y12 <- y1 + y2  
 gamma\_int <- seq(gamma\_min, gamma\_max, by = gamma\_step)  
 gamma\_hat <- sum(y1) / sum(y2)  
 ## emptry matrix to store the values  
 gamma\_mle <- matrix(NA, nrow = N, ncol = length(gamma\_int))  
  
 for (j in seq\_along(1:N)) {  
 for (i in seq\_along(1:length(gamma\_int))) {  
 phat <- gamma\_int[i] / (1 + gamma\_int[i])  
 ## simulate y1 | y1+y2 2000 times  
 # ya3 <- map(y12, ~rbinom(n = 1, size = .,   
 # prob = phat)) %>% unlist()  
 ya3 <- rbinom(nrow(y1),  
 size = unlist(y1 + y2),  
 prob = phat)  
 gamma\_mle[j, i] <- sum(ya3) / sum(y12 - ya3)   
 }  
 }  
   
 ## get F(0.25) and F(0.975)  
 q025 <- apply(gamma\_mle, 2,   
 quantile, probs = alpha / 2,  
 na.rm = TRUE)  
  
 q975 <- apply(gamma\_mle, 2,   
 quantile, probs = 1 - alpha / 2,   
 na.rm = TRUE)  
 ## check whether gamma\_int is located inside F025 F975  
 gamma\_ind <- ifelse((gamma\_hat > q025) & (gamma\_hat < q975), 1, 0)  
 ## find the smallest and largest gamma\_int  
 lower <- min(gamma\_int[which(gamma\_ind == 1)])  
 upper <- max(gamma\_int[which(gamma\_ind == 1)])  
 return(c(lower, upper))  
}  
  
# get\_gamma(y\_ss100[[1]])

## parallel programming  
library(furrr)  
library(parallel)  
detectCores()  
  
## use 6 cores  
plan(multisession, workers = 6)  
finall\_5k\_N15 <- future\_map\_dfc(y\_ss15, ~get\_gamma(y = .))  
finall\_5k\_N100 <- future\_map\_dfc(y\_ss100, ~get\_gamma(y = .))  
  
save(finall\_5k\_N15, finall\_5k\_N100, finall\_per,  
 file = "bios\_qexam\_question1.Rdata")

load("bios\_qexam\_question1.Rdata")  
library(matrixStats)  
rowMeans(as.matrix(finall\_5k\_N15))  
rowMeans(as.matrix(finall\_5k\_N100))

## (d)

Implement a fully reproducible simulation study which assesses the performance of the confidence intervals obtained in (b) and (c). Specifically, implement a simulation study as follows:

* ,
* ,
* ,
* ,
* is an equally spaced grid on with interval length of
* Simulate datasets in Step 2c of the algorithm from (c)

Estimate 95% confidence intervals using the approaches you derived in both (b) and (c) in 5000 simulated datasets and evaluate power for testing the null hypothesis . Compare approaches on coverage probability and power. Summarize your results in 3-4 sentences as if for a statistical journal. Support your claims using 1-2 tables or figures. Provide well commented code which reproduces your simulation study and recreates your tables/figures.

The confidence interval for sample size 15 is [1.89, 3.11]

The confidence interval for sample size 100 is [1.95, 2.97]

## (e)

Apply your methods to the dataset “MVPA intervention.csv”, a data matrix which contains two columns, “Y1” and “Y2”, corresponding to the number of 10 minute bouts of moderate-to-vigorous physical activity in matched pairs. Estimate and construct 95% confidence intervals using both of the approaches you derived in (b) and (c).

Write a brief, non-technical, summary of both methods as if communicating to a non-statistician, recommend and justify one approach, and conclude with an interpretation of the estimated coefficient and confidence interval.

library(tidyverse)  
y <- read\_csv("data/MVPA\_intervention.csv")  
y1 <- y$Y1  
y2 <- y$Y2  
y1\_bar <- mean(y1)  
y2\_bar <- mean(y2)  
## mle   
(mean <- y1\_bar / y2\_bar)  
## sd   
## use observed information  
(gamma <- y1\_bar / y2\_bar)  
Inof <- ((2 \* gamma + 1) / (gamma^2 \* (1 + gamma)^2) \* sum(y1) - 1 / (gamma + 1)^2 \* sum(y2)) / 15  
(sd <- (1 / Inof) %>% sqrt()) / sqrt(15); sd  
## use phat  
sd0 <- sqrt((y1\_bar + y2\_bar) \* y1\_bar / y2\_bar^3) / sqrt(15)  
## 95% CI  
mean - 1.96 \* sd0  
mean + 1.96 \* sd0   
  
get\_gamma(y = y, gamma\_max = 15)

## use permutation to get the confidence interval   
y\_permu <- map(1:1000, ~sample\_n(y, size = 15, replace = TRUE))  
  
## parallel programming  
library(furrr)  
library(parallel)  
  
## use 6 cores  
plan(multisession, workers = 6)  
finall\_per <- future\_map\_dfc(y\_permu, ~get\_gamma(y = ., gamma\_max = 10))  
rowMedians(as.matrix(finall\_per))  
## [1] 1.67 9.95  
## [1] 1.825 8.640