04_question4

44

2022-06-02

Q4 You have received data on patients with cardiac monitoring devices that detect atrial fibrillation events. I

n 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 Y with mean λ for which you observe either Y directly (Group A), or the $I(Y > 0)$ (Group B).

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

Define a latent variable $Z \sim Bernoulli(p)$, independent of Y, as a binary indicator of whether Y is observed directly (i.e., $Z = 1$ indicates an observation is from Group A and $Z = 0$ indicates an observation is from Group B).

The observed outcome X can then be defined in terms of $Y \sim Poisson(\lambda)$, $I(Y > 0)$ and Z.

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 $= 1$ or > 1 events

(iii) X = k, k > 1: there are k events

$$Y_i|Z_i = 1 \sim Poisson(\lambda)$$
$$P(Y = 0) = e^{-\lambda}$$
$$P(Y = 0|Z = 1) = e^{-\lambda}$$
$$P(Y = 1|Z = 1) = \lambda e^{-\lambda}$$
$$Y_i|Z_i = 0 \sim Bernoulli\left(1 - e^{-\lambda}\right)$$
$$P(Y = 0|Z = 0) = e^{-\lambda}$$
$$P(Y = 1|Z = 0) = 1 - e^{-\lambda}$$
$$Z_i \sim Bernoulli(p_3)$$

The mixture of degenerate distribution at zero with probability p_1 and Poisson distribution with probability $1 - p_1$; the Poisson mixed with two-point masses at 0 with p_1 and at 1 with p_2; $p_3 = 1 - p_1 - p_2$.

$$\begin{aligned}
P(Y = y) &= P(Y|Z = 0) \times P(Z = 0) + P(Y|Z = 1) \times P(Z = 1) \\
&= \begin{cases} p_1 + p_3 e^{-\lambda} = e^{-\lambda} & y = 0 \\ p_2 + p_3 \lambda e^{-\lambda} & y = 1 \\ 0 + p_3 \dfrac{\lambda^y e^{-\lambda}}{y!} & y > 1 \end{cases} \\
p_1 &= P(Y = 0|Z = 0) \times P(Z = 0) = \frac{\lambda^0 e^{-\lambda}}{0!} p_3 = e^{-\lambda}(1 - p_3) \\
p_2 &= P(Y = 1|Z = 0) \times P(Z = 0) = \left(1 - e^{-\lambda}\right)(1 - p_3) \\
p_1 &+ p_2 + p_3 = 1 \\
L(p_3, \lambda|Y) &= \prod_{i \in \{y_i = 0\}} e^{-\lambda} \prod_{i \in \{y_i = 1\}} \left(1 - e^{-\lambda}\right)(1 - p_3) + p_3 \lambda e^{-\lambda} \prod_{i \in \{y_i > 1\}} p_3 \frac{\lambda^y e^{-\lambda}}{y!}
\end{aligned}$$

ii. The complete data (Y, Z).

Hint: Consider the cases for when

(i) Z = 1,Y = k,

(ii) Z = 0, Y = 0,

(iii) Z = 0, Y = k, k>0

$$P(Y,Z) = P(Y|Z) \times P(Z) = \begin{cases} P(Y=0|Z=1) \times P(Z=1) = e^{-\lambda} p_3 \\ P(Y=1|Z=1) \times P(Z=1) = \lambda e^{-\lambda} p_3 \\ P(Y=k|Z=1) \times P(Z=1) = \dfrac{\lambda^k e^{-\lambda}}{k} p_3 \\ P(Y=0|Z=0) \times P(Z=0) = e^{-\lambda}(1-p_3) \\ P(Y=k|Z=0) \times P(Z=0) = 0 \\ P(Y=1|Z=0) \times P(Z=0) = \left(1-e^{-\lambda}\right)(1-p_3) \end{cases}$$

$$\begin{aligned} L\left(p_3, \lambda|(Y,Z)\right) &= \prod_{i \in \{z_i=1\}} \frac{\lambda_i^y e^{-\lambda}}{y_i} p_3 \prod_{i \in \{z_i=0, y_i=0\}} e^{-\lambda}(1-p_3) \prod_{i \in \{z_i=0, y_i=1\}} \left(1-e^{-\lambda}\right)(1-p_3) \\ &= \prod_{i \in \{z_i=1\}} \frac{\lambda_i^y e^{-\lambda}}{y_i} p_3 \prod_{i \in \{z_i=0, y_i=0\}} p_1 \prod_{i \in \{z_i=0, y_i=1\}} p_2 \end{aligned}$$

When $Z = 0$, we do not have the information of exact Y, but a Bernoulli distribution conditional on the probability of $Z = 0$.

(b) Numerically optimize the observed data log-likelihood using "optim" in R, to obtain the estimates, standard errors, and 95% confidence intervals for λ and p.

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

λ expected mean as 1.755, standard deviation as 0.107, and the 95% CI as (1.546, 1.964).

λ has a higher possibility at 1.755 than other values, if we randomly simulate a confidence interval (1.546, 1.964) will have 95% chance to cover the true λ value. In a less formal way to interpret the results, we are 95% confident that the population parameter is between (1.546, 1.964), The later interpretation will mislead the reader to regard the true parameter value as random and regard the interval as fixed.

p expected mean as 0.844, standard deviation as 0.064, and the 95% Cl as (0.718, 0.970)

p has a comparatively higher possibility at 0.844 and if we randomly simulate a confidence interval (0.718, 0.970) will have 95% chance to cover the true p value. In a less formal way to interpret the results, we are 95% confident that the population parameter is between (0.718, 0.970) The later interpretation will mislead the reader to regard the true parameter value as random and regard the interval as fixed.

When $Z = 0$, we do not have the information of exact Y, but a Bernoulli conditional on the probability of $Z = 0$. The more Z = 0 the less informative the data could be applied. Because we cannot observe Z, then we have to estimate what is the p for the distribution of Z. Therefore, if the probability p is optimized as 1, then there is no Z =0. If we know the true value of Z other than estimation, we will have more accurate and less biased estimation for both λ and p.

$$L(p_3, \lambda|Y) = \prod_{i\in\{y_i=0\}} e^{-\lambda} \prod_{i\in\{y_i=1\}} \left(1-e^{-\lambda}\right)(1-p_3) + p_3\lambda e^{-\lambda} \prod_{i\in\{y_i>1\}} p_3 \frac{\lambda^y e^{-\lambda}}{y!}$$

$$logL(p_3, \lambda|Y) = -\sum_{i\in\{y_i=0\}} \lambda + \sum_{i\in\{y_i=1\}} log\left(\left(1-e^{-\lambda}\right)(1-p_3) + p_3\lambda e^{-\lambda}\right) + \sum_{i\in\{y_i>1\}} log(p_3) + ylog(\lambda) - \lambda - log(y!)$$

```
#' 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
```

```r
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)
}


## Using optim to get the estimation of p and lambda
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$par
par_sd <- solve(p_hat$hessian) %>% diag() %>% sqrt()
par_mean + 1.96 * par_sd

## [1] 1.181429 2.076756

par_mean - 1.96 * par_sd

## [1] 0.8185713 1.6592201
```

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

$$\Lambda = -2log\left[\frac{sup_{\theta\in\Theta_0}L(\theta)}{sup_{\theta\in\Theta}L(\theta)}\right] \sim \chi^2$$

Null Hypothesis: the inflated mixture model is the same as the Poisson model

$\theta = \theta_0 : p = 1, \lambda = \lambda_0,$

$$\Lambda = -2log\left[\frac{sup_{\theta\in\Theta_0}L(\theta)}{sup_{\theta\in\Theta}L(\theta)}\right] = 0$$

Alternative Hypothesis: the inflated mixture model is not the same as the Poisson model

$\theta \neq \theta_0 : p \neq 1, \lambda \neq \lambda_0,$

$$\Lambda = -2log\left[\frac{sup_{\theta\in\Theta_0}L(\theta)}{sup_{\theta\in\Theta}L(\theta)}\right] \neq 0$$

With likelihood ratio statistics, we performed Chi square test and get p-value = 0.064 > 0.05 marginally larger than the critical value. Hence, we claim the inflated mixture model is not statistically significantly different from Poisson model. However due to the small sample size and marginally acceptance of the null hypothesis, I will guess further investigation with higher power.

```
#' 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 Z).

Specifically, report bias and efficiency for estimating λ. Fix $\lambda = 1$. Vary sample size ($N = 100,1000$) and p ($p = 0.2,0.5,0.8$) to generate data from 6 settings. Simulate a reasonable number of replicates.

$$HW = z_{1-\alpha/2}\sqrt{\frac{Var(Y)}{n}}$$

$$n_w = (\frac{z_{1-\alpha/2}}{HW})^2 Var(W)$$

$$n_y = (\frac{z_{1-\alpha/2}}{HW})^2 Var(Y)$$

$$TE_w = (\frac{z_{1-\alpha/2}}{HW})^2 Var(W)E_w$$

$$TE_y = (\frac{z_{1-\alpha/2}}{HW})^2 Var(Y)E_y$$

$$ARE(W,Y) = Var(Y)/Var(W)$$

- Two random variables are simulated as W with M_w method and Y with M_y method, with $E[W] = E[Y] = \theta$
- HW is the half width
- Let E_w and E_y denote the amount of computational effort required to produce one sample of W and Y, respectively.
- Then the total effort expended by M_w and M_y, respectively, to achieve a half width HW
- M_w is more efficient than M_y if $TE_w < TE_y$. Note that $TE_w < TE_y$ if and only if $Var(W)E_w < Var(Y)E_y$.
- $ARE(W,Y)$ is used to qualify efficiency of E_y and E_w

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

The data generating function is `get_sim()`;

The incomplete likelihood without information of Z is the function `get_log_L_incomplete()`;

The complete likelihood with information of both Y and Z is the function `get_log_L_complete()`;

Function `get_optim()` is to use `optim` function to get the estimation of parameter (p, λ).

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.

In this study, we are trying to find whether we can get accurate estimation of parameters in both complete and incomplete data. The null hypothesis will be that both methods can provide reasonable and accurate estimation on parameters. As we see in the table below, indeed, both methods provide reasonable estimation with small bias and standard deviations. Comparatively the complete data with both Z and Y observed can provide less biased and smaller standard-deviation estimation on the parameters. For complete dataset is less biased and more efficient on p estimation. The bias and efficiency for lambda is similar in both complete and incomplete cases. This is more obvious in the cases of smaller p value, which indicates there is higher change of missing true Y observations.

In this study, we generated dataset with fixed $\lambda = 1$. Vary sample size ($N = 100,1000$) and p ($p = 0.2,0.5,0.8$) to generate data from 6 settings. In both complete and incomplete cases, the larger the sample size, the smaller of both biases and standard error of the parameters. When it comes to Z = 0, we are missing the information of observing true Y values. With larger value of p, the higher of the chance of Z =1, which means to observe the real value of Y. Hence, we can get more information from the data. This directly results less biased and less deviated estimation of p. However, the effect of larger p value does not affect the estimation of lambda.

In conclusion, ideally, we prefer the hypothesis test on a large sample size with more Z = 1 observations. The complete dataset can provide more information than the incomplete dataset, with less biased and less variated estimations.

Settingup		Imcomplete						Complete					
Sample Size	p	Mean[p]	Bias[p]	SD[p]	Mean[lambda]	Bias[[lambda]	SD[lambda]	Mean[p]	Bias[p]	SD[p]	Mean[lambda]	Bias[lambda]	SD[lambda]
N100	p = 0.2	0.215	0.015	0.085	1.009	0.009	0.133	0.203	0.003	0.040	1.011	0.011	0.126
	p = 0.5	0.505	0.005	0.138	1.001	0.001	0.129	0.500	0.000	0.049	0.998	0.002	0.116
	p = 0.8	0.795	0.005	0.146	1.007	0.007	0.120	0.799	0.001	0.040	1.000	0.000	0.107
N1000	p = 0.2	0.201	0.001	0.027	0.992	0.008	0.035	0.200	0.000	0.013	0.992	0.008	0.032
	p = 0.5	0.504	0.004	0.042	0.989	0.011	0.033	0.500	0.000	0.017	0.990	0.010	0.029
	p = 0.8	0.800	0.000	0.053	0.993	0.007	0.033	0.799	0.001	0.013	0.992	0.008	0.026

```
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
```

```r
  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()
```

```r
  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")

## question ii

library(matrixStats)

##
## Attaching package: 'matrixStats'

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

```r
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
```