Statistics 2: Computer Practical 2

Jake Ireland (1908320) Isaac Rawcliffe (1603871)

1 Processing times

The data are recorded processing times in minutes for an immigration control counter in Wakanda. Due to extensive training and technological advances, the processing time for travellers without any special circumstances is known to follow an Exponential(1) distribution: the variability is due to the specific attributes of the traveller. The distribution of processing times for travellers with special circumstances is assumed to be Exponential, but it is not known what the statistical parameter is, or what proportion of travellers have special circumstances.

2 A mixture distribution

Given the description above, the processing time of a traveller follows a "mixture model". Specifically, consider the random variable

$$X = \mathbb{I}(B = 1)Y + \mathbb{I}(B = 0)Z,$$

where $B \sim$ Bernoulli(p), $Y \sim$ Exponential(λ) and $Z \sim$ Exponential(1) are independent. The random variable B corresponds to whether the traveller has special circumstances $(B = 1)$ or not $(B = 0)$. The statistical parameters are p and λ so we write $\theta = (p, \lambda)$.

The intuition is that X is with probability p an Exponential(λ) random variable, and with probability $1 - p$ an Exponential(1) random variable.

We shall write $X \sim \text{MixExp}(p, \lambda)$. To simulate a vector of realizations of independent MixExp(p, λ) random variables, we can use the following code.

```
rmixexp <- function(n, p, lambda) {
  bs <- rbinom(n,1,p) # generate n independent Bernoulli(p) random variates
  ys <- rexp(n,lambda) # generate n independent Exponential(lambda) random variates
  zs <- rexp(n,1) # generate n independent Exponential(1) random variates
  bs*ys + (1-bs)*zs # use the definition of X
}
```

We can derive the CDF of X as follows. Let $G(\cdot; \lambda)$ be the CDF of an Exponential(λ) random variable. We have

$$\begin{aligned} F_X(x; p, \lambda) &= \mathbb{P}(X \leq x; p, \lambda) \\ &= \mathbb{P}(B = 1, X \leq x; p, \lambda) + \mathbb{P}(B = 0, X \leq x; p, \lambda) \\ &= pF_Y(x; \lambda) + (1 - p)F_Z(x) \\ &= pG(x; \lambda) + (1 - p)G(x; 1). \end{aligned}$$

It follows that the PDF of X is

$$f_X(x; p, \lambda) = pg(x; \lambda) + (1 - p)g(x; 1),$$

where $g(\cdot;\lambda)$ is the PDF of an Exponential(λ) random variable.

To evaluate the PDF at each point in a vector, we can use the following code.

```
dmixexp <- function(xs,p,lambda) {
  p*dexp(xs,lambda) + (1-p)*dexp(xs,1)
}
```

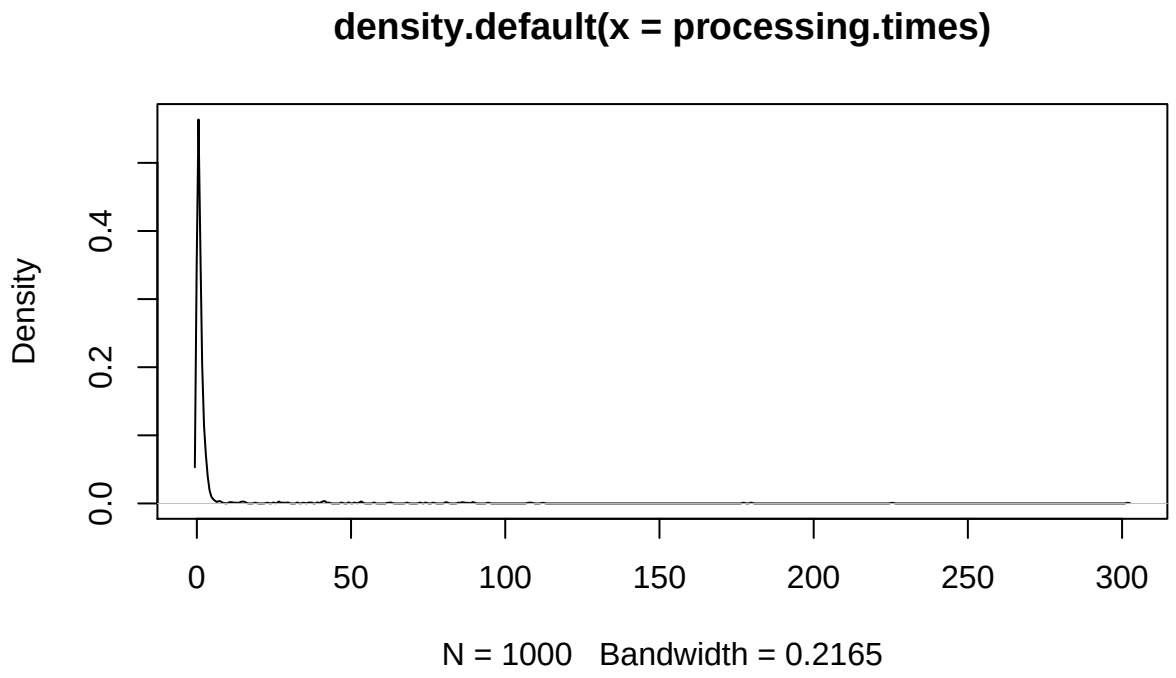
3 Data

We can load the data as follows:

```
processing.times <- read.csv("processing-times.csv")$x
```

We can see by plotting an estimate of the density of the processing time distribution that most of the times are fairly small, but there is a "long tail".

```
plot(density(processing.times))
```



The distribution is unlikely to be Exponential(λ) for any value of λ, since the empirical mean is very different from the square root of the empirical variance.

```
mean(processing.times)
```

```
## [1] 4.515289
```

```r
var(processing.times)
```

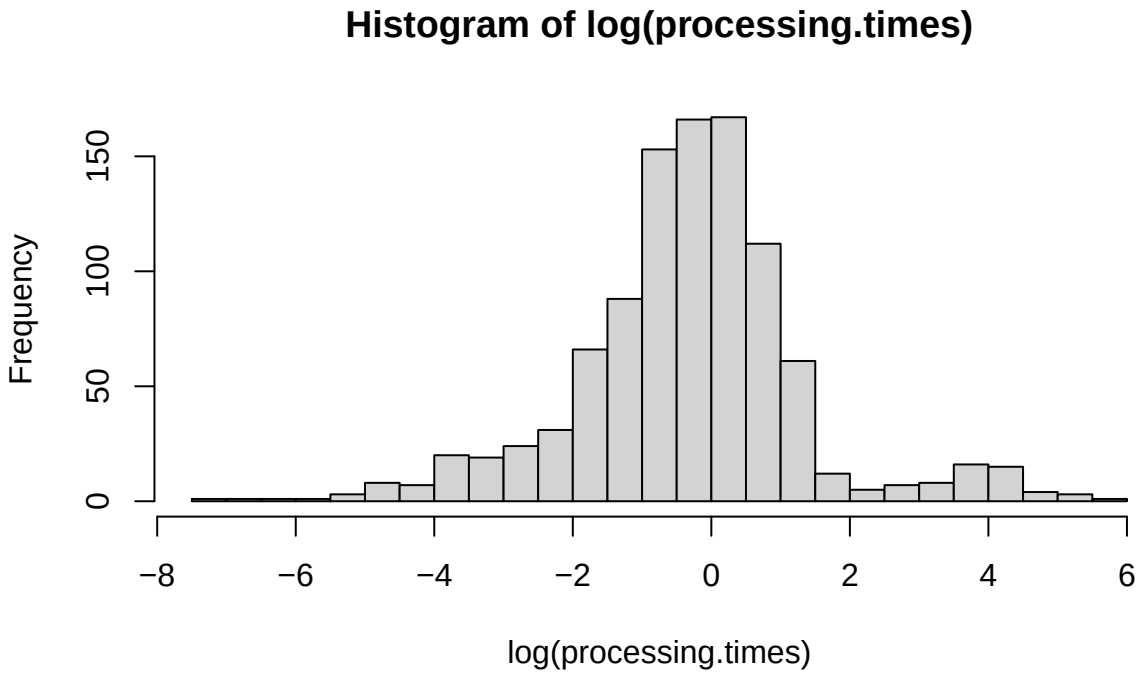
```
## [1] 363.5594
```

A histogram of the processing times data would not be very easy to interpret, since so many data points are close to 0. For the purposes of visualization (only), we can transform the model by taking $W = \log(X)$. It is then possible to obtain that the density of W is

$$f_W(w; p, \lambda) = \exp(w) f_X(\exp(w), p, \lambda).$$

This will allow to use to check visually our model fit for the transformed data.

```r
hist(log(processing.times),breaks=20)
```



4 Estimation

We can use the `"L-BFGS-B"` method of the `optim` function to write a function `ml.estimate` that numerically maximizes the log-likelihood function, and returns an approximate maximum likelihood estimate of $\theta = (p, \lambda)$.

```r
# compute the score associated with all the observations
score <- function(theta,xs) {
  p <- theta[1]
  lambda <- theta[2]
  v1 <- sum((dexp(xs,lambda)-dexp(xs,1))/dmixexp(xs,p,lambda))
  v2 <- sum(p*exp(-lambda*xs)*(1-lambda*xs)/dmixexp(xs,p,lambda))
```

```
  c(v1,v2)
}

# obtain the maximum likelihood estimate numerically
# uses the log-likelihood function and also the gradient (the score)
# the use of the gradient makes the algorithm faster
ml.estimate <- function(xs) {
  ell <- function(theta) {
    p <- theta[1]
    lambda <- theta[2]
    sum(log(dmixexp(xs,p,lambda)))
  }
  neg.ell <- function(theta) {
    -ell(theta)
  }
  g <- function(theta) {
    -score(theta,xs)
  }
  out <- optim(c(0.1,0.1),neg.ell,gr = g,method="L-BFGS-B",
               lower=c(1e-5,1e-5),upper=c(1-1e-5,Inf))$par
  p.hat <- out[1]
  lambda.hat <- out[2]
  c(p.hat, lambda.hat)
}
```

We can compute the ML estimate of θ for the data.

```
ml.estimate(processing.times)
```

```
## [1] 0.06771554 0.01894260
```

Question 1. [2 marks] Derive the method of moments estimators of p and λ, and report the estimates for this dataset.

I suggest that you write a function `mom.estimate` that takes as input some data and returns a vector containing the estimated values of p and λ. For example, something like this:

```
mom.estimate <- function(xs) {
  ## your code here
  p.hat <- 0 # change this line
  lambda.hat <- 0 # change this line
  c(p.hat, lambda.hat)
}
```

— **Solution** —

We have $X \sim \text{MixExp}(p, \lambda)$ with $f(x; p, \lambda) = p\lambda e^{-\lambda x} + (1-p)e^{-x}$. To calculate the MOM estimates for p and λ we first need to calculate the first and second moments of X. For the first moment we have

$$\begin{aligned}\mathbb{E}(X) &= \int_{-\infty}^{\infty} x f_X(x) dx \\ &= \int_0^{\infty} x(p\lambda e^{-\lambda x} + (1-p)e^{-x}) dx \\ &= p\lambda \int_0^{\infty} x e^{-\lambda x} dx + (1-p) \int_0^{\infty} x e^{-x} dx\end{aligned}$$

Now we can show $\int_0^{\infty} x e^{-\lambda x} dx = \frac{1}{\lambda^2}$ using integration by parts and L'hôpital's rule. Hence $\mathbb{E}(X) = \frac{p}{\lambda} + (1-p)$. For the second moment we have

$$\begin{aligned}\mathbb{E}(X^2) &= \int_{-\infty}^{\infty} x^2 f_X(x) dx \\ &= \int_0^{\infty} x^2(p\lambda e^{-\lambda x} + (1-p)e^{-x}) dx \\ &= p\lambda \int_0^{\infty} x^2 e^{-\lambda x} dx + (1-p) \int_0^{\infty} x^2 e^{-x} dx\end{aligned}$$

Similarly we have $\int_0^{\infty} x^2 e^{-\lambda x} dx = \frac{2}{\lambda^3}$ again using integration by parts and L'hôpital's rule. Hence $\mathbb{E}(X^2) = \frac{2p}{\lambda^2} + 2(1-p)$. Now we define for $j \in \mathbb{Z}_{>0}$

$$m_j = \frac{1}{n} \sum_{i=1}^{n} x_i^j$$

where our data takes the form $x_1, x_2, \ldots, x_n$. We now solve the system of equations

$$\begin{cases} \frac{p}{\lambda} + (1-p) = \mathbb{E}(X) = m_1 \\ \frac{2p}{\lambda^2} + 2(1-p) = \mathbb{E}(X^2) = m_2 \end{cases}.$$

By substitution we get

$$p = \frac{m_1 - 1}{\lambda^{-1} - 1}$$

and

$$(m_2 - 2m_1)\lambda^2 + (2 - m_2)\lambda + 2m_1 - 2 = 0$$

which solved together give us our MOM estimates. We factorise the quadratic to get

$$(m_2 - 2m_1)\lambda^2 + (2 - m_2)\lambda + 2m_1 - 2 = (\lambda - 1)[(m_2 - 2m_1)\lambda + 2 - 2m_1]$$

. So our solutions for lambda are $\lambda = 1$ and $\lambda = \frac{2m_1 - 2}{m_2 - 2m_1}$. Note $\lambda = 1$ leaves p undefined so $\lambda = \frac{2m_1 - 2}{m_2 - 2m_1}$ only.

```
mom.estimate <- function(xs) {
  # calculate moments
  m1 <- mean(xs)
  m2 <- mean(xs^2)
```

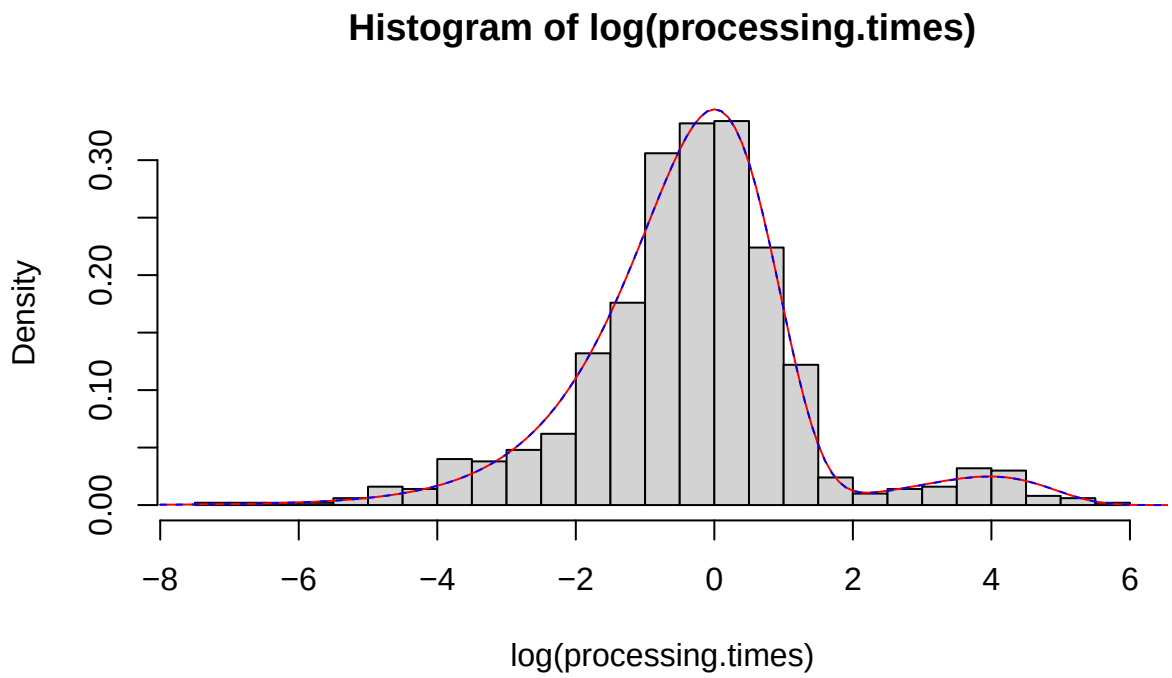
```r
  # calculate estimates
  lambda.hat <- (2*m1 - 2) / (m2 - 2*m1)
  p.hat <- (m1 - 1) / ((1 / lambda.hat) - 1)

  # return estimates
  c(p.hat, lambda.hat)
}

cat("(p.hat, lambda.hat) = ", mom.estimate(processing.times))
```

```
## (p.hat, lambda.hat) =  0.06724624 0.01877057
```

— **End of Solution** —

For each estimate, we can visualize the fitted (transformed) distribution's PDF alongside the histogram of the log-transformed data.

```r
theta.ml <- ml.estimate(processing.times)
theta.mom <- mom.estimate(processing.times)
hist(log(processing.times),breaks=20,freq=FALSE)
vs <- seq(-8,8,0.1)
lines(vs,exp(vs)*dmixexp(exp(vs),theta.ml[1],theta.ml[2]),col="red")
lines(vs,exp(vs)*dmixexp(exp(vs),theta.mom[1],theta.mom[2]),col="blue",lty='dashed')
```



Question 2. [2 marks] Run a simulation study and plot the density estimates of the ML and MoM estimators for both p and λ when $\theta = \theta^* = (0.05, 0.02)$ and $n = 1000$.

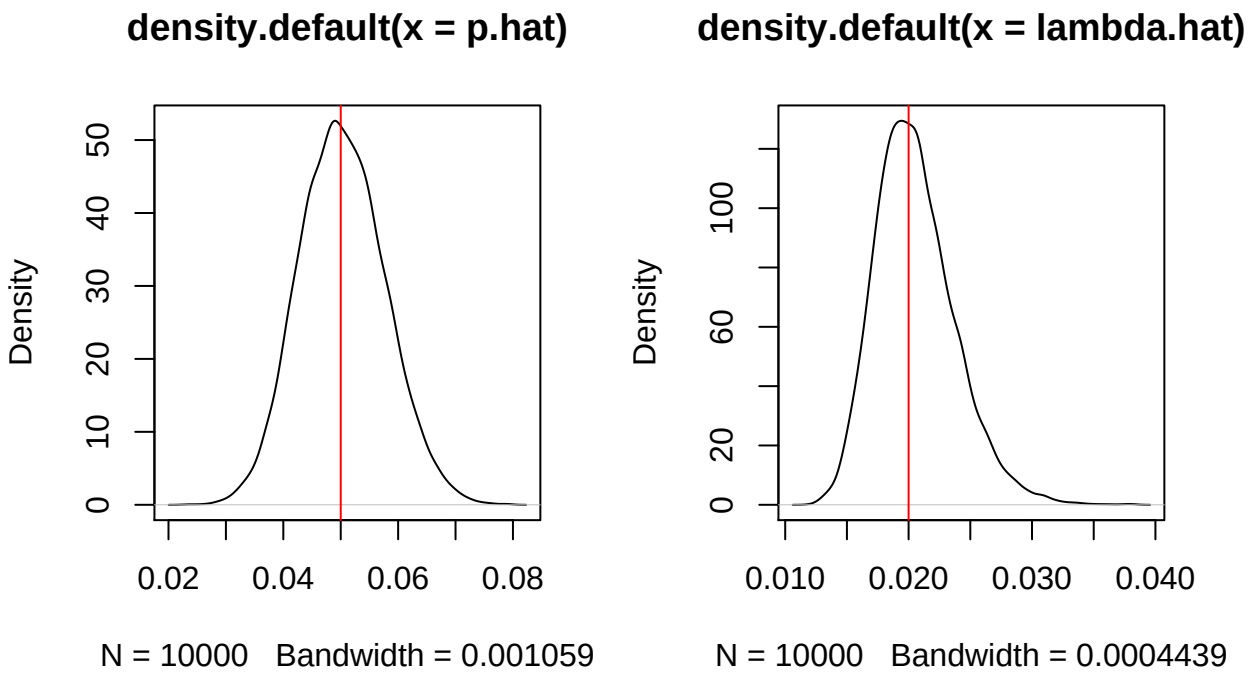
— **Solution** — For the simulation study, we create random numbers for the Mixed Exponential distribution which we arrange in a matrix where each row represents a new sample of size 1000. Then we will apply an

estimator function to each row to calculate a value for $\hat{p}$ and $\hat{\lambda}$ for each sample. Then we can plot histograms of these values representing their respective density estimates.

```
B <- 10000 # number of samples
n <- 1000 # size of sample
theta <- c(0.05, 0.02)

estimator.density.plot <- function(n, B, theta, estimator) {
  # generate nB random numbers.
  big.sample <- rmixexp(n*B, theta[1], theta[2])
  # arranged random numbers in a B*n matrix, each row represents a different sample
  samples <- matrix(big.sample, nrow=B)

  # apply the given estimator function to each row and replace with the estimates
  # output is 2*B matrix
  estimates <- apply(samples, 1, estimator)

  # split into estimates for each parameter
  p.hat = estimates[1,]
  lambda.hat = estimates[2,]

  # plot density plots
  par(mfrow=c(1,2))
  plot(density(p.hat))
  #hist(p.hat,breaks=50,freq=FALSE)
  abline(v=theta[1], col="red")
  #hist(lambda.hat,breaks=50,freq=FALSE)
  plot(density(lambda.hat))
  abline(v=theta[2], col="red")

}
```

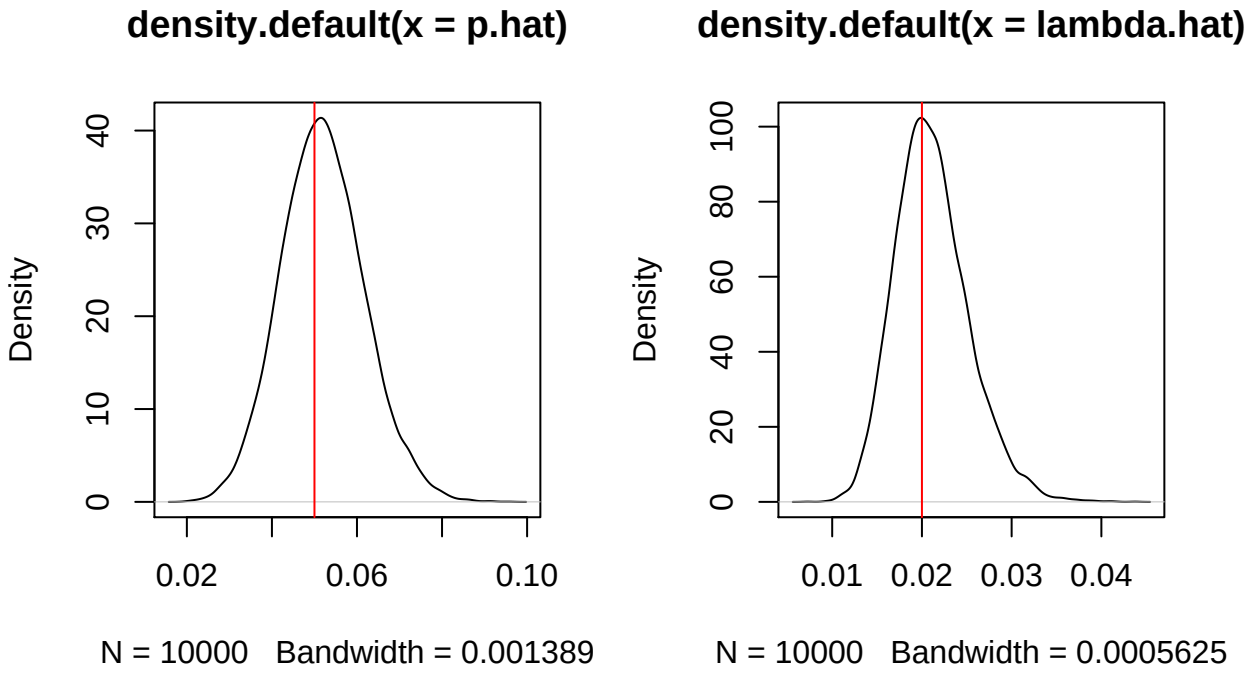
For the ML estimate we have the following density plots.

```
estimator.density.plot(n,B,theta,ml.estimate)
```



For the MOM estimate we have the following density plots.

```
estimator.density.plot(n,B,theta,mom.estimate)
```



— End of Solution —

5 Confidence Intervals

In lectures we have seen that for regular statistical models with a one-dimensional parameter θ, the ML estimator $\hat{\theta}_n$ is *asymptotically normal* with

$$I_n(\theta)^{1/2}(\hat{\theta}_n - \theta) = \sqrt{nI(\theta)}(\hat{\theta}_n - \theta) \to_{\mathcal{D}(\cdot;\theta)} Z \sim N(0,1).$$

This convergence in distribution justifies the construction of Wald confidence intervals for θ.

In this computer practical, the statistical model has a 2-dimensional parameter. Under appropriate regularity assumptions, the ML estimator $\hat{\theta}_n$ is *asymptotically (multivariate) normal* in the sense that $I_n(\theta)^{1/2}(\hat{\theta}_n - \theta)$ converges in distribution to a vector of 2 independent standard normal random variables, where $I_n(\theta)$ is the Fisher information *matrix*

$$I_n(\theta) = -\mathbb{E}[\nabla^2 \ell(\theta; X_1, \ldots, X_n); \theta],$$

where the expectation is taken element-wise. That is, the ijth entry of $I_n(\theta)$ is the negative expectation of the corresponding ijth second order partial derivative of the log-likelihood.

One can deduce from this multivariate asymptotic normality that for $j \in \{1, \ldots, 2\}$,

$$\frac{\hat{\theta}_{n,j} - \theta_j}{\sqrt{(I_n(\theta)^{-1})_{jj}}} \to_{\mathcal{D}(\cdot;\theta)} Z \sim N(0,1),$$

where $\hat{\theta}_{n,j}$ denotes the jth component of $\hat{\theta}_n$ and θ_j denotes the jth component of θ.

Notice that $(I_n(\theta)^{-1})_{jj}$ is the jth diagonal entry of the inverse of the Fisher information matrix, and is not in general equal to $(I_n(\theta)_{jj})^{-1}$, the inverse of the jth diagonal entry of the Fisher information matrix[1]. In `R` you can compute numerically the inverse of a matrix using the `solve` command.

As in the one-dimensional parameter case, one can replace $I_n(\theta)$ with the observed Fisher information matrix

$$nJ_n(\hat{\theta}_n) = -\sum_{i=1}^{n} \nabla^2 \ell(\hat{\theta}_n; X_1, \ldots, X_n),$$

to obtain

$$\frac{\hat{\theta}_{n,j} - \theta_j}{\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}}} \to_{\mathcal{D}(\cdot;\theta)} Z \sim N(0,1).$$

Question 3. [3 marks] This is an example of a model where the Fisher information is not a simple function. However, the Hessian matrix of second-order partial derivatives of the log-likelihood for a single observation x

$$\nabla^2 \ell(\theta; x)_{ij} = \frac{\partial^2}{\partial\theta_i \partial\theta_j} \ell(\theta; x),$$

can be computed using the following function:

```
hessian <- function(theta,x) {
  p <- theta[1]
  lambda <- theta[2]
  H <- matrix(0,2,2)
  H[1,1] <- - (dexp(x,lambda)-dexp(x,1))^2/dmixexp(x,p,lambda)^2
  H[1,2] <- exp(-(lambda+1)*x)*(1-lambda*x)/dmixexp(x,p,lambda)^2
  H[2,1] <- H[1,2]
  H[2,2] <- - exp(-lambda*x)*p*(p*exp(-lambda*x)+exp(-x)*x*(p-1)*(lambda*x-2))/
    dmixexp(x,p,lambda)^2
  H
}
```

[1] This is statistically interesting, as it captures the fact that our estimators are less precise in the presence of more unknown parameters.

Compute the observed Fisher information matrix, and use it to construct asymptotically exact 95% observed confidence intervals for p and λ.

I recommend that you write a function that computes the observed information matrix.

```
observed.info <- function(theta,xs) {
  # your code
}
```

I also recommend that you write a function that computes the observed confidence intervals' end points, which should call the `observed.info` function.

```
compute.CI.endpoints <- function(xs,alpha) {
  # your code
  L.p <- 0 # obviously wrong
  U.p <- 1 # obviously wrong
  L.lambda <- 0 # obviously wrong
  U.lambda <- 1 # obviously wrong
  list(p=c(L.p,U.p),lambda=c(L.lambda,U.lambda))
}
```

Finally, you may find it helpful to know that you can use the `solve` function to compute the inverse of a matrix.

— **Solution** —

As $nJ_n(\hat{\theta}_n) = -\sum_{i=1}^{n} \nabla^2 \ell(\hat{\theta}_n; X_1, \ldots, X_n)$, we can compute the hessian of the likelihood function for each observation, then take the negative sum to get the observed information.

For the confidence interval we have

$$\mathbb{P}(z_{1-\alpha/2} \leq \frac{\hat{\theta}_{n,j} - \theta_j}{\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}}} \leq z_{\alpha/2}) = \mathbb{P}(\frac{\hat{\theta}_{n,j} - \theta_j}{\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}}} \leq z_{\alpha/2}) - \mathbb{P}(\frac{\hat{\theta}_{n,j} - \theta_j}{\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}}} \leq z_{1-\alpha/2}) = 1-\alpha.$$

Also

$$\left\{ z_{1-\alpha/2} \leq \frac{\hat{\theta}_{n,j} - \theta_j}{\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}}} \leq z_{\alpha/2} \right\} = \left\{ \hat{\theta}_{n,j} - z_{\alpha/2}\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}} \leq \theta_j \leq \hat{\theta}_{n,j} + z_{\alpha/2}\sqrt{((nJ_n(\hat{\theta}_n))^{-1})_{jj}} \right\}.$$

which gives us our exact $1-\alpha$ confidence interval for θ_j.

```
observed.info <- function(theta,xs) {
  n <- length(xs)

  # create an empty 2*2 matrix
  J <- matrix(0,2,2)
  for (x in xs) {
    # subtract the hessian of likelihood function for the observation x
    H <- hessian(theta, x)
    J <- J - H
  }

  J
}

observed.info(ml.estimate(processing.times), processing.times)
```

```
##           [,1]      [,2]
## [1,] 13829.50  -6258.51
## [2,] -6258.51 169193.74
```

```
compute.CI.endpoints <- function(xs,alpha) {
  # compute the ML estimate of the observations
  theta.hat <- ml.estimate(xs)
  # compute the observed information
  obs.info <- observed.info(theta.hat, xs)
  # take the inverse of the observed information
  inv.obs.info <- solve(obs.info)

  # compute confidence intervals for p and lambda
  # NOTE: absolute value used to prevent negative diagonal entries (for Q4)
  L.p <- theta.hat[1] - qnorm(1-(alpha/2))*sqrt(abs(inv.obs.info[1,1]))
  U.p <- theta.hat[1] + qnorm(1-(alpha/2))*sqrt(abs(inv.obs.info[1,1]))
  L.lambda <- theta.hat[2] - qnorm(1-(alpha/2))*sqrt(abs(inv.obs.info[2,2]))
  U.lambda <- theta.hat[2] + qnorm(1-(alpha/2))*sqrt(abs(inv.obs.info[2,2]))
  list(p=c(L.p,U.p),lambda=c(L.lambda,U.lambda))
}

compute.CI.endpoints(processing.times, 0.05)
```

```
## $p
## [1] 0.05090775 0.08452333
##
## $lambda
## [1] 0.01413729 0.02374792
```

— **End of Solution** —

Question 4. [3 marks] Using simulations, approximate the coverage of the asymptotically exact $1-\alpha$ confidence intervals corresponding to your answer to Question 3, when $(p,\lambda) = (0.05, 0.02)$ and for $n \in \{50, 100, 1000\}$. Comment on the coverage of the confidence intervals and explain in words why the approximate coverage you report is approximate.

I recommend that you write a function

```
approximate.coverage <- function(n,p,lambda,trials) {
  # your code
  coverage.p <- 0 # hopefully wrong
  coverage.lambda <- 0 # hopefully wrong
  c(coverage.p,coverage.lambda)
}
```

You may encounter in your simulations observed Fisher information matrices whose inverse has negative diagonal entries. Obviously, this is not a good estimate of the Fisher information. I recommend that to avoid trying to take the square root of a negative number that you consider the absolute value of the inverse's diagonal entries when constructing your observed confidence intervals.

— **Solution** —

The coverage for θ_j is defined as

$$\mathbb{P}(\theta_j \in [L_j(\mathbf{x}), U_j(\mathbf{x})])$$

We can approximate this as

$$\mathbb{P}(\theta_j \in [L_j(\mathbf{x}), U_j(\mathbf{x})]) \approx \frac{C}{T}$$

where C is the number of times θ_j falls in the interval and T is the number of trials in a simulation study.

```
approximate.coverage <- function(n,p,lambda,trials, alpha) {
  count.in.CI.p <- 0
  count.in.CI.lambda <- 0
  for (i in 1:trials) {
    # compute sample of size n from a mixed exponential distribution
    sample <- rmixexp(n, p, lambda)
    # estimate the confidence intervals for the sample
    CI <- compute.CI.endpoints(sample, alpha)
    # check if p and lambda fall in their respective intervals
    if (CI$p[1] <= p && p <= CI$p[2]) {
      count.in.CI.p <- count.in.CI.p + 1
    }
    if (CI$lambda[1] <= lambda && lambda <= CI$lambda[2]){
      count.in.CI.lambda <- count.in.CI.lambda + 1
    }
  }

  c(count.in.CI.p/trials, count.in.CI.lambda/trials)

}
approximate.coverage(50, 0.05, 0.02, 1000, 0.05)
```

```
## [1] 0.999 0.883
```

```
approximate.coverage(100, 0.05, 0.02, 1000, 0.05)
```

```
## [1] 0.945 0.942
```

```
approximate.coverage(1000, 0.05, 0.02, 1000, 0.05)
```

```
## [1] 0.957 0.958
```

As n increases, the coverage tends to $1 - \alpha$ which we would expect as the observed information will tend to the Fisher Information. This is still an approximate value for the coverage because n is finite.

— **End of Solution** —

6 Epilogue

This computer practical involves a slightly more complicated distribution for the data than the standard distributions typically used for communicating the basic ideas. The use of more sophisticated models is important when applying statistical ideas to real world data. As in this case, it is not uncommon that the ML estimate has to be computed numerically and that the Fisher information is not a straightforward function to calculate. Similarly, most real-world models have several statistical parameters, and the use of two parameters here is therefore a gentle introduction to what you would do in more realistic statistical analyses.