Atlas-PS 3

David Atlas

September 17, 2018

Problem 1

We define the likelihood function $L(\hat{\alpha}; X)$:

$$L(\alpha; X) = \prod_{i=1}^{n} \frac{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^{x_i} \mathrm{e}^{-(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})}}{x_i!},$$

and the log-likelihood function $l(\hat{\alpha}; X)$:

$$l(\alpha; X) = \Sigma_{i=1}^{n} x_i \log \alpha_1 b_{i,1} + \alpha_2 b_{i,2} - \Sigma_{i=1}^{n} \alpha_1 b_{i,1} - \Sigma_{i=1}^{n} \alpha_2 b_{i,2} - \Sigma_{i=1}^{n} \log(x_i!).$$

a)

Derive the Newton Raphson update for finding the MLEs of α_1 and $alpha_2$.

First, we take the first derivative of l' with respect to $\hat{\alpha}$. This leaves us with a 2x1 matrix of first derivatives.

$$\begin{bmatrix} \Sigma_{i=1}^{n} \frac{x_i b_{i,1}}{\alpha_1 b_{i,1} + \alpha_2 b_{i,2}} - \Sigma_{i=1}^{n} b_{i,1} \\ \Sigma_{i=1}^{n} \frac{x_i b_{i,2}}{\alpha_1 b_{i,1} + \alpha_2 b_{i,2}} - \Sigma_{i=1}^{n} b_{i,2} \end{bmatrix}.$$

We find the Hessian:

$$\begin{bmatrix} \Sigma_{i=1}^{n} \frac{-x_i b_{i,1}^2}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-x_i b_{i,1} b_{i,2}}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} \\ \Sigma_{i=1}^{n} \frac{-x_i b_{i,1} b_{i,2}}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-x_i b_{i,2}^2}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} \end{bmatrix}$$

The Newton-Raphson update is $h = -\mathbf{l}''(\theta)^{-\mathbf{1}}\mathbf{l}'(\theta)$. We combine the two of them below:

$$h(\alpha) = - \begin{bmatrix} \Sigma_{i=1}^{n} \frac{-x_i b_{i,1}^2}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-x_i b_{i,1} b_{i,2}}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} \\ \Sigma_{i=1}^{n} \frac{-x_i b_{i,1} b_{i,2}}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-x_i b_{i,2}^2}{(\alpha_1 b_{i,1} + \alpha_2 b_{i,2})^2} \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_{i=1}^{n} \frac{x_i b_{i,1}}{\alpha_1 b_{i,1} + \alpha_2 b_{i,2}} - \Sigma_{i=1}^{n} b_{i,1} \\ \Sigma_{i=1}^{n} \frac{x_i b_{i,2}}{\alpha_1 b_{i,1} + \alpha_2 b_{i,2}} - \Sigma_{i=1}^{n} b_{i,2} \end{bmatrix}.$$

b)

Derive the Fisher Scoring update.

We take the Hessian calculated above. We site the textbook for expected value for a $X \sim \text{Poisson}(\lambda)$ distribution: $E(X) = \lambda$. We also point out that the expected value of a sum is equal to the sum of expected values, or $\Sigma_{i=1}^{n} E(X) = E(\Sigma_{i=1}^{n} x)$.

As such, we can write the Fisher Information $I(\alpha) = -\mathrm{E}(\mathrm{l}''(\alpha))$ as:

$$-\begin{bmatrix} \Sigma_{i=1}^{n} \frac{-\mathrm{E(x_i)b_{i,1}^2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-\mathrm{E(x_i)b_{i,1}b_{i,2}}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} \\ \Sigma_{i=1}^{n} \frac{-\mathrm{E(x_i)b_{i,1}b_{i,2}}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-\mathrm{E(x_i)b_{i,2}^2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} \end{bmatrix} = -\begin{bmatrix} \Sigma_{i=1}^{n} \frac{-(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})b_{i,1}^2}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})b_{i,1}b_{i,2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} \\ \Sigma_{i=1}^{n} \frac{-(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})b_{i,1}b_{i,2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} & \Sigma_{i=1}^{n} \frac{-(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})b_{i,2}^2}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})^2} \end{bmatrix}$$
$$= \begin{bmatrix} \Sigma_{i=1}^{n} \frac{b_{i,1}^2}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} & \Sigma_{i=1}^{n} \frac{b_{i,1}b_{i,2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} \\ \Sigma_{i=1}^{n} \frac{b_{i,1}b_{i,2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} & \Sigma_{i=1}^{n} \frac{b_{i,2}^2}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} \end{bmatrix}.$$

We can then write the Fisher Scoring update, $I(\theta)^{-1}l'(\theta)$ as:

$$\begin{bmatrix} \Sigma_{i=1}^{n} \frac{b_{i,1}^2}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} & \Sigma_{i=1}^{n} \frac{b_{i,1}b_{i,2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} \\ \Sigma_{i=1}^{n} \frac{b_{i,1}b_{i,2}}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} & \Sigma_{i=1}^{n} \frac{b_{i,2}^2}{(\alpha_1 b_{i,1}+\alpha_2 b_{i,2})} \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_{i=1}^{n} \frac{x_i b_{i,1}}{\alpha_1 b_{i,1}+\alpha_2 b_{i,2}} - \Sigma_{i=1}^{n} b_{i,1} \\ \Sigma_{i=1}^{n} \frac{x_i b_{i,2}}{\alpha_1 b_{i,1}+\alpha_2 b_{i,2}} - \Sigma_{i=1}^{n} b_{i,2} \end{bmatrix}$$

c)

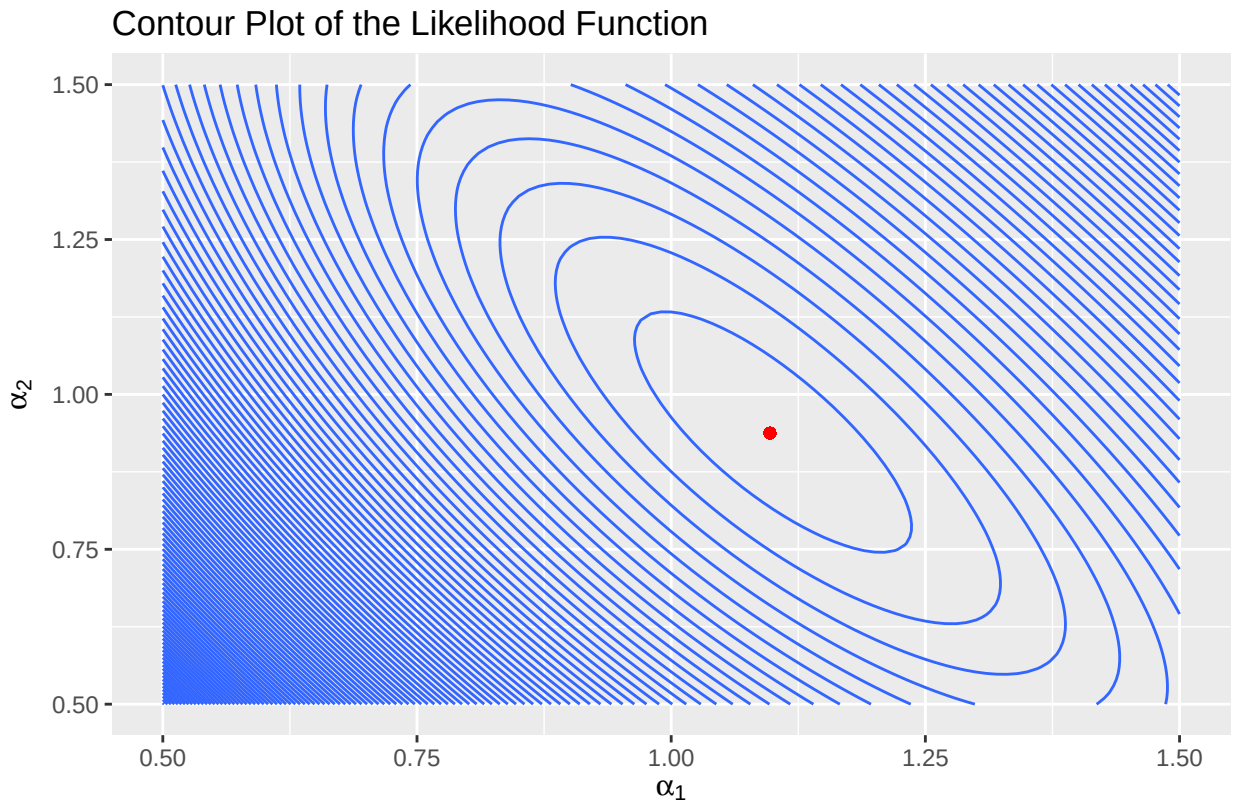
We implement Newton's Method.

```
oil <- read.table("../datasets/oilspills.dat", header=TRUE)

fprime <- function(alpha, b, x){
  return(c(sum(x * b[, 1] / (b %*% alpha)) - sum(b[, 1]),
           sum(x * b[, 2] /  (b %*% alpha)) - sum(b[, 2])))
}

f2prime <- function(alpha, b, x){
  return(-1 * matrix(c(
    sum(x * b[, 1]^2 / (b %*% alpha)^2),
    sum(x * apply(b, 1, prod) / (b %*% alpha)^2),
    sum(x * apply(b, 1, prod) / (b %*% alpha)^2),
    sum(x * b[, 2]^2 / (b %*% alpha)^2)
  ), ncol=2))
}


newtons_method <- function(fprime, f2prime, alpha0, b, x, max_iter=10000, tol=.001){
  alpha_t <- alpha0

  # Iterate through
  for (n in 1:max_iter){
    # Set stopping conditions
    if(sum((alpha0 - alpha_t)^2) < tol & n > 1){break}

    alpha0 <- alpha_t

    # Get the Newton update
    alpha_t <- alpha0 - solve(f2prime(alpha0, b, x)) %*% fprime(alpha0, b, x)
  }

  return(c(alpha_t=alpha_t, n=n))
}
```

```r
# We call the function on our dataset
alpha0 <- c(1, 1)
b <- as.matrix(oil[, c('importexport', 'domestic')])
x <- as.matrix(oil[, c('spills')])

solution <- newtons_method(fprime, f2prime, c(1, 1), b, x, tol=.00001)
```

The solution is given as $\alpha = [1.097, .938]$, converging in 4 iterations. Below, the contour plot for the likelihood function is shown, with the red dot labelling the given solution. We see that the algorithm appears to have converged on the solution.

```r
log_likelihood <- function(alpha, b, x){
  return(sum(x * log(b %*% alpha)) - sum(alpha[1] * b[, 1])
         - sum(alpha[2] * b[, 2]) - sum(log(factorial(x))))
}

# we construct agrid of the likelihood function to plot the contours
a1 <- seq(0.5, 1.5, .01)
a2 <- seq(0.5, 1.5, .01)
alpha_space <- as.matrix(expand.grid(a1, a2)) # Create cartesian product

# Find the likelihod for all pairs
results <- data.frame(cbind(
  alpha_space, apply(
    alpha_space, 1,
      function(alpha) log_likelihood(alpha, b=b, x=x))))

# Add column names
colnames(results) <- c("alpha1", "alpha2", "likelihood")

n_bin <- 100


# Plot the contours with the solution in red
ggplot(results) +
  geom_contour(aes(x=alpha1, y=alpha2, z=likelihood), bins=n_bin) +
  geom_point(aes(x=solution[1], y=solution[2]), colour="red") +
  xlab(TeX("$\\alpha_1$")) + ylab(TeX("$\\alpha_2$")) +
  ggtitle("Contour Plot of the Likelihood Function") +
  labs(caption="Note: The solution using the Newton-Raphson method is shown in red.")
```



Note: The solution using the Newton−Raphson method is shown in red.

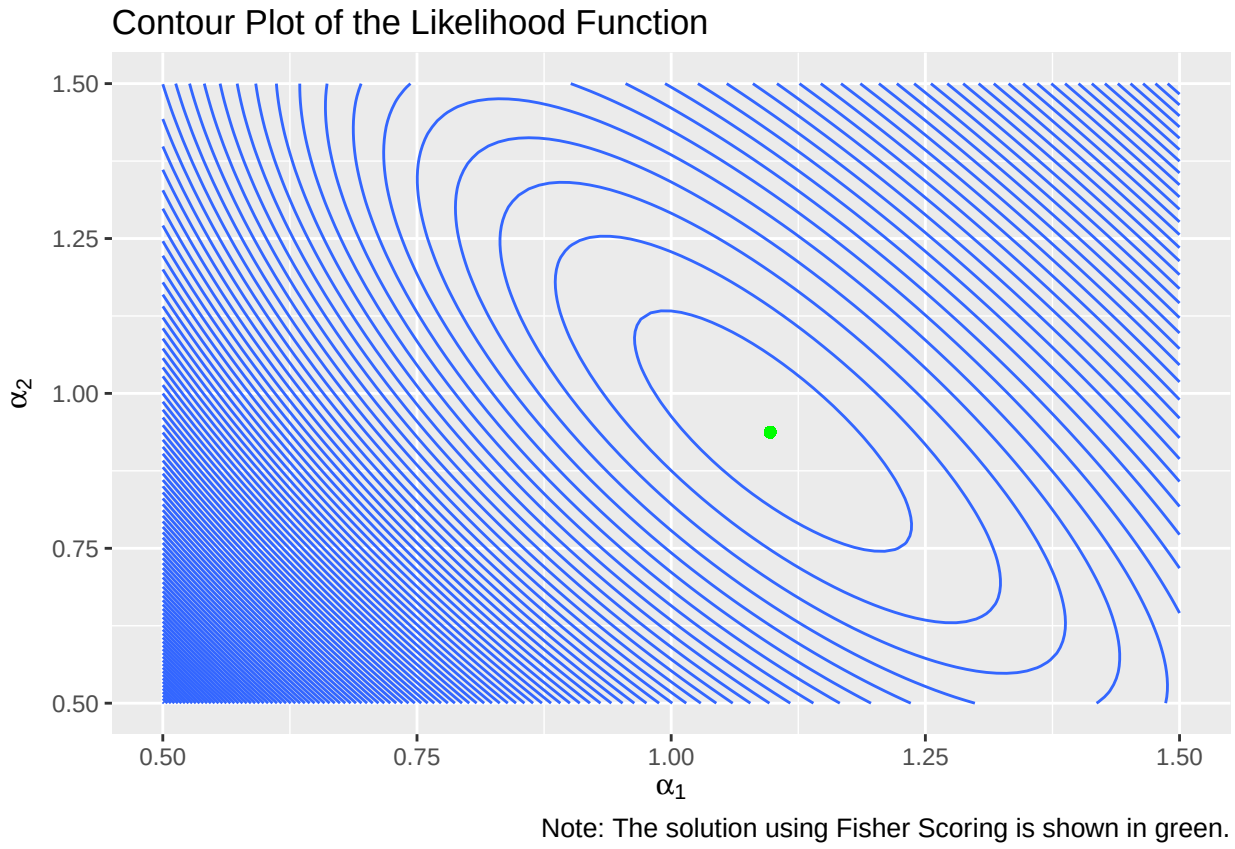
Next, we implement the Fisher Scoring algorithm.

```
# We implement the Fisher scoring update method

I <- function(alpha, b, x){
  return(matrix(c(
    sum(b[, 1]^2 / (b %*% alpha)),
    sum(apply(b, 1, prod) / (b %*% alpha)),
    sum(apply(b, 1, prod) / (b %*% alpha)),
    sum(b[, 2]^2 / (b %*% alpha))
  ), ncol=2))
}

fisher_scoring <- function(I, fprime, alpha0, b, x, max_iter=10000, tol=.001){
  alpha_t <- alpha0

  # Iterate through
  for (n in 1:max_iter){
    # Set stopping conditions
    if(sum((alpha0 - alpha_t)^2) < tol & n > 1){break}

    alpha0 <- alpha_t

    # Get the Fisher update
    alpha_t <- alpha0 + solve(I(alpha0, b, x)) %*% fprime(alpha0, b, x)
  }
```

```
  return(c(alpha_t=alpha_t, n=n))
}

# We call the function on our dataset
alpha0 <- c(1, 1)
b <- as.matrix(oil[, c('importexport', 'domestic')])
x <- as.matrix(oil[, c('spills')])

solution <- fisher_scoring(I=I, fprime=fprime, alpha0=c(1, 1), b=b, x=x, tol=.00001)
```

The solution is given as $\alpha = [1.097, .938]$, converging in 6 iterations. Below, the contour plot for the likelihood function is shown, with the green dot labelling the given solution. We see that the algorithm appears to have converged on the solution. Note that this is the same solution seen above with Newton's Algorithm. This is as expected, as the two techniques are quite similar.

```
# Plot the contours with the solution in red
ggplot(results) +
  geom_contour(aes(x=alpha1, y=alpha2, z=likelihood), bins=n_bin) +
  geom_point(aes(x=solution[1], y=solution[2]), colour="green") +
  xlab(TeX("$\\alpha_1$")) + ylab(TeX("$\\alpha_2$")) +
  ggtitle("Contour Plot of the Likelihood Function") +
  labs(caption="Note: The solution using Fisher Scoring is shown in green.")
```



Both algorithms converge to the same solution, given a starting point near the correct solution. Fisher scoring takes 2 more iterations to get there, but roughly comparable. The computation implementation ease is nearly identical, although because we used the Hessian to get the Fisher Information, it was an extra step of derivation over Newton's Method. If the Hessian was more difficult to find, Fisher Scoring would make

more sense.

e)

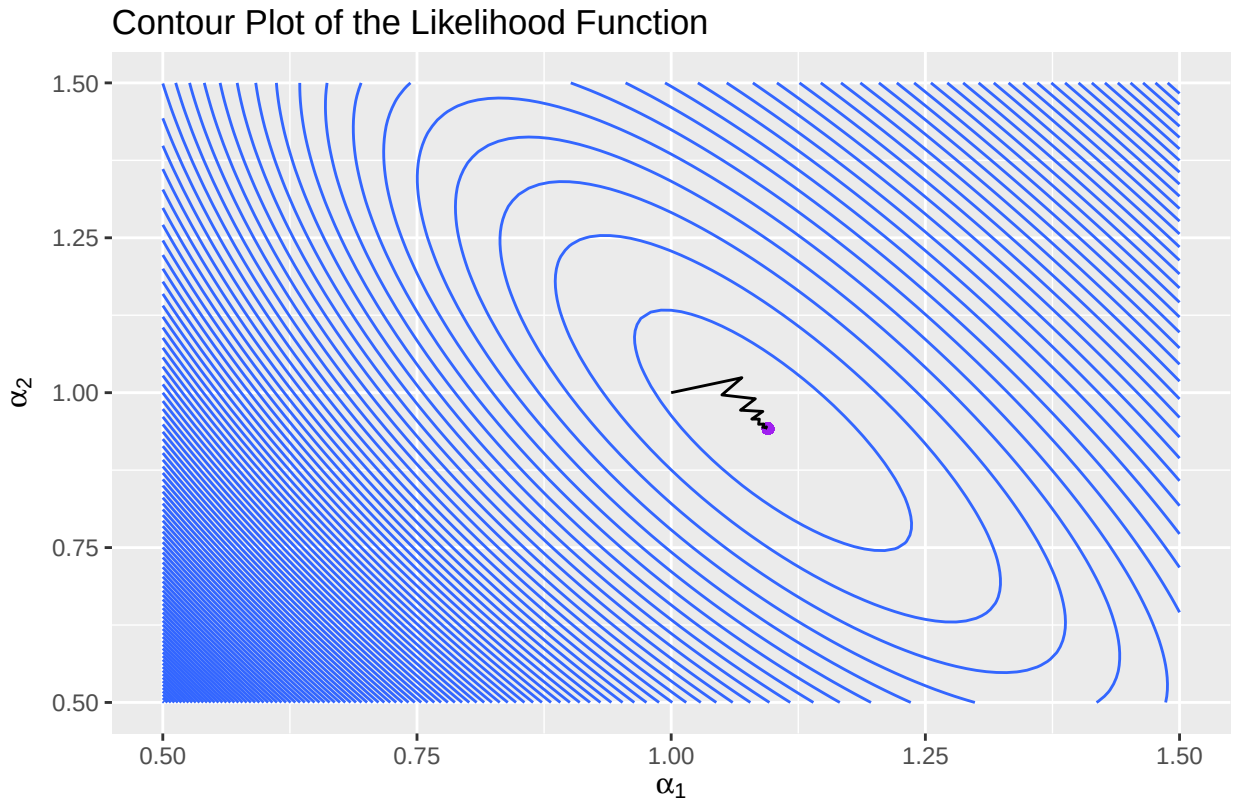
Next, we implement the method of steepest ascent, and apply it to the problem. We already have the first derivative of the log-likelihood function above, and so we do not need any further derivation to use the steepest ascent algorithm with backtracking.

```
steepest_ascent <- function(f, fprime, alpha0, b, x, max_iter=10000, tol=.001){
  alpha_t <- alpha0
  path <- matrix(ncol=length(alpha0), nrow=0)

  # Iterate through
  for (n in 1:max_iter){

    # Set stopping conditions
    if(sum((alpha0 - alpha_t)^2) < tol & n > 1){break}

    alpha0 <- alpha_t

    # Reset the backtrack scaling to 1
    backtrack <- 1

    # Get the update h
    fprime_x <- fprime(alpha0, b, x)
    M <- diag(length(fprime_x))
    h_t <- backtrack * (solve(M) %*% fprime_x)

    # While the next point would be negative, backtrack (divide by 2)
    while(f(alpha0 + h_t, b, x) < f(alpha0, b, x)){
      backtrack <- backtrack / 2
      h_t <- backtrack * (solve(M) %*% fprime_x)
    }

    # Iterate to the next point
    alpha_t <- alpha0 + h_t
    path <- rbind(path, t(alpha0))
  }

  return(list(path=path, n=n, alpha_t=alpha_t))
}

solutions <- steepest_ascent(log_likelihood, fprime, c(1, 1), b=b, x=x, tol=.00001)
```

The solver works effectively, finding $\alpha = [1.095, .941]$ in 14 iterations. This is a bit slower than the methods above (Newton's and Fisher Scoring). However, the derivation here is much easier, as you only need the first derivative and the likelihood function (no second derivative or Fisher Information). Below, we plot the solution, and the path taken to get there.

```
# Plot the contours with the solution in red
ggplot(results) +
  geom_contour(aes(x=alpha1, y=alpha2, z=likelihood), bins=n_bin) +
  geom_point(aes(x=solutions$alpha_t[1], y=solutions$alpha_t[2]), colour="purple") +
  geom_path(data=data.frame(solutions$path), aes(x=X1, y=X2)) +
```

```
  xlab(TeX("$\\alpha_1$")) + ylab(TeX("$\\alpha_2$")) +
  ggtitle("Contour Plot of the Likelihood Function") +
  labs(caption="Note: The solution using steepest ascent is shown in purple, with the path taken to get
```



Problem 2

a)

Show that the complete-data log-likelihood function is $\log \pi \Sigma_{i=1}^n z_i + \Sigma_{i=1}^n z_i \log \phi(x_i; \mu_1, \sigma_1^2) + \log(1-\pi)(n - \Sigma_{i=1}^n z_i) + \Sigma_{i=1}^n (1-z_i) \log \phi(x_i; \mu_2, \sigma_2^2)$

Given the complete-data density for Y

$$p(x_i, z_i|\theta) = [\pi\phi(x; \mu_1, \sigma_1^2)]^{z_i}[(1-\pi)\phi(x_i; \mu_2, \sigma_2^2)]^{(1-z_i)},$$

we can write the complete log-likelihood function for θ as the product of n random draws from the distribution of Y:

$$L(\theta|x, z) = \prod_{i=1}^{n} [\pi\phi(x_i; \mu_1, \sigma_1^2)]^{z_i}[(1-\pi)\phi(x_i; \mu_2, \sigma_2^2)]^{(1-z_i)}.$$

We take can then write the log-likelihood

$$
\begin{aligned}
l(\theta|x,z) &= \Sigma_{i=1}^{n} \log(\pi\phi(x_i;\mu_1,\sigma_1^2))^{z_i} + \Sigma_{i=1}^{n} \log((1-\pi)\phi(x_i;\mu_2,\sigma_2^2))^{(1-z_i)} \\
&= \Sigma_{i=1}^{n} z_i \log\pi + \Sigma_{i=1}^{n} z_i \log\phi(x_i;\mu_1,\sigma_1^2) + \Sigma_{i=1}^{n}(1-z_i)\log(1-\pi) + \Sigma_{i=1}^{n}(1-z_i)\log\phi(x_i;\mu_2,\sigma_2^2) \\
&= \log\pi\Sigma_{i=1}^{n} z_i + \Sigma_{i=1}^{n} z_i \log\phi(x_i;\mu_1,\sigma_1^2) + \log(1-\pi)(n-\Sigma_{i=1}^{n} z_i) + \Sigma_{i=1}^{n}(1-z_i)\log\phi(x_i;\mu_2,\sigma_2^2)
\end{aligned}
$$

b)

Find $Q(\theta|\theta^{(k)})$.

We then can write $Q(\theta|\theta^{(k)})$ as the expected value of the $l(\theta|Y)$ with respect to the distribution of Z. As X is observed, all terms not dependant on Z are constant. We can write

$$
\begin{aligned}
Q(\theta|\theta^{(k)} &= \mathrm{E}[\log \mathrm{f_Y}(\mathrm{y}|\theta)|\mathrm{x},\theta^{(\mathrm{k})}] \\
&= E[\log\pi\Sigma_{i=1}^{n} z_i + \Sigma_{i=1}^{n} z_i \log\phi(x_i;\mu_1,\sigma_1^2) + \log(1-\pi)(n-\Sigma_{i=1}^{n} z_i) + \Sigma_{i=1}^{n}(1-z_i)\log\phi(x_i;\mu_2,\sigma_2^2)] \\
&= \log\pi\Sigma_{i=1}^{n} E[Z_i] + \Sigma_{i=1}^{n} E[Z_i]\log\phi(x_i;\mu_1,\sigma_1^2) + \log(1-\pi)(n-\Sigma_{i=1}^{n} E[Z_i]) + \Sigma_{i=1}^{n}(1-E[Z_i])\log\phi(x_i;\mu_2,\sigma_2^2),
\end{aligned}
$$

where $E[Z_i]$ is conditional on x and $\theta^{(k)}$.

c)

As $z_i \in 0, 1, \forall i \in \mathbb{N}$, we can say that

$$
\begin{aligned}
E[Z_i|x_i,\theta^{(k)}] &= 0 \times P[Z_i = 0|x_i,\theta^{(k)}] + 1 \times P[Z_i = 1|x_i,\theta^{(k)}] \\
&= P[Z_i = 1|x_i,\theta^{(k)}].
\end{aligned}
$$

Additionally, we can say that

$$
P[Z_i = 1|x_i,\theta^{(k)}] = \frac{P(Z_i = 1|x_i,\theta^{(k)})}{P(Z_i = 1|x_i,\theta^{(k)}) + P(Z_i = 0|x_i,\theta^{(k)})}
$$

where $P(Z_i = 1|x_i,\theta^{(k)}) + P(Z_i = 0|x_i,\theta^{(k)}) = 1$.

Using the given density for Y, and conditioning on $\theta = (\pi,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2)$, we can say that

$$
\begin{aligned}
P[Z_i = 1|x_i,\theta^{(k)}] &= [\pi^{(k)}\phi(x_i;\mu_1^{(k)},(\sigma_1^2)^{(k)})]^1[(1-\pi^{(k)})\phi(x_i;\mu_2^{(k)},(\sigma_2^2)^{(k)})]^0 \\
&= \pi^{(k)}\phi(x_i;\mu_1^{(k)},(\sigma_1^2)^{(k)})
\end{aligned}
$$

and

$$
\begin{aligned}
P[Z_i = 0|x_i,\theta^{(k)}] &= [\pi^{(k)}\phi(x_i;\mu_1^{(k)},(\sigma_1^2)^{(k)})]^0[(1-\pi^{(k)})\phi(x_i;\mu_2^{(k)},(\sigma_2^2)^{(k)})]^1 \\
&= (1-\pi^{(k)})\phi(x_i;\mu_2^{(k)},(\sigma_2^2)^{(k)})
\end{aligned}
$$

Combining the two, we can write that

$$
E[Z_i|x_i,\theta^{(k)}] = \frac{\pi^{(k)}\phi(x_i;\mu_1^{(k)},(\sigma_1^2)^{(k)})}{\pi^{(k)}\phi(x_i;\mu_1^{(k)},(\sigma_1^2)^{(k)}) + (1-\pi^{(k)})\phi(x_i;\mu_2^{(k)},(\sigma_2^2)^{(k)})}
$$

d)

To find $\pi^{(k)}$, we must solve $\frac{\partial Q(\theta|\theta^{(k)})}{\partial \pi} = 0$. Using our equation for $Q(\theta|\theta^{(k)})$, we can write can write

$$\frac{\partial Q(\theta|\theta^{(k)})}{\partial \pi} = \frac{\Sigma_{i=1}^{n} E[Z_i]}{\pi} - \frac{n - \Sigma_{i=1}^{n} E[Z_i]}{1-\pi}.$$

Setting the derivative equal to zero and solving for $\hat{\pi}$ gives us

$$\begin{aligned}
0 &= \frac{\Sigma_{i=1}^{n} E[Z_i]}{\hat{\pi}} - \frac{n - \Sigma_{i=1}^{n} E[Z_i]}{1-\hat{\pi}} \\
&\implies \hat{\pi}(n - \Sigma_{i=1}^{n} E[Z_i]) = \Sigma_{i=1}^{n} E[Z_i] - \hat{\pi}\Sigma_{i=1}^{n} E[Z_i] \\
&\implies \hat{\pi} n = \Sigma_{i=1}^{n} E[Z_i] \\
&\implies \hat{\pi} = \frac{\Sigma_{i=1}^{n} E[Z_i]}{n} = \frac{\eta^{(k)}}{n}
\end{aligned}$$

e)

To find the updates for each of the parameters, we take the partial derivatives for each of the unknown parameters and find their roots. We use the proportionality given in the assignment.

Solve for $\mu_1^{(k+1)}$

$$\begin{aligned}
0 = \frac{\partial Q(\theta|\theta^{(k)})}{\partial \mu_1} &= \Sigma_{i=1}^{n} E[Z_i] \frac{x_i - \hat{\mu 1}_1}{\sigma_1^2} \\
&\implies \Sigma_{i=1}^{n} E[Z_i] x_i = \hat{\mu_1} \Sigma_{i=1}^{n} E[Z_i] \\
&\implies \hat{\mu_1} = \frac{\Sigma_{i=1}^{n} E[Z_i] x_i}{\Sigma_{i=1}^{n} E[Z_i]} = \frac{1}{\eta^{(k)}} \Sigma_{i=1}^{n} \eta_i^{(k)} x_i
\end{aligned}$$

Solve for $(\sigma_1^2)^{(k+1)}$

$$\begin{aligned}
0 = \frac{\partial Q(\theta|\theta^{(k)})}{\partial \sigma_1^2} &= \Sigma_{i=1}^{n} E[Z_i] \left(-\frac{1}{2\sigma_1^2} + \frac{(x_i - \mu_1)^2}{2(\sigma_1^2)^2} \right) \\
&\implies 0 = \Sigma_{i=1}^{n} E[Z_i] \left(-\frac{1}{2\hat{\sigma_1^2}} + \frac{(x_i - \mu_1)^2}{(2\hat{\sigma_1^2})^2} \right) \\
&\implies \frac{1}{2\hat{\sigma_1^2}} \Sigma_{i=1}^{n} E[Z_i] = \Sigma_{i=1}^{n} E[Z_i] \frac{(x_i - \mu_1)^2}{2(\hat{\sigma_1^2})^2} \\
&\implies \hat{\sigma_1^2} \Sigma_{i=1}^{n} E[Z_i] = \Sigma_{i=1}^{n} E[Z_i](x_i - \mu_1)^2 \\
&\implies \hat{\sigma_1^2} \Sigma_{i=1}^{n} E[Z_i] = \Sigma_{i=1}^{n} E[Z_i](x_i - \mu_1)^2 \\
&\implies \hat{\sigma_1^2} = \frac{\Sigma_{i=1}^{n} E[Z_i](x_i - \mu_1)^2}{\Sigma_{i=1}^{n} E[Z_i]} = \frac{1}{\eta^{(k)}} \Sigma_{i=1}^{n} \eta_i^{(k)} (x_i - \mu_1)^2
\end{aligned}$$

Solve for $\mu_2^{(k+1)}$

$$
\begin{aligned}
0 = \frac{\partial Q(\theta|\theta^{(k)})}{\partial \mu_2} &= \Sigma_{i=1}^{n}(1 - E[Z_i])\frac{x_i - \hat{\mu_2}}{\sigma_2^2} \\
&\implies \hat{\mu_2}\Sigma_{i=1}^{n}(1 - E[Z_i]) = \Sigma_{i=1}^{n}(1 - E[Z_i])x_i \\
&\implies \hat{\mu_2} = \frac{\Sigma_{i=1}^{n}(1 - E[Z_i])x_i}{\Sigma_{i=1}^{n}(1 - E[Z_i])} \\
&= \frac{1}{n - \Sigma_{i=1}^{n}E[Z_i]}\Sigma_{i=1}^{n}(1 - E[Z_i])x_i = \frac{1}{n - \eta^{(k)}}\Sigma_{i=1}^{n}(1 - \eta_i^{(k)})x_i
\end{aligned}
$$

Solve for $(\sigma_2^2)^{(k+1)}$

$$
\begin{aligned}
\frac{\partial Q(\theta|\theta^{(k)})}{\partial \sigma_2^2} &= \Sigma_{i=1}^{n}(1 - E[Z_i])[-\frac{1}{2\sigma_2^2} + \frac{(x_i - \mu_2)^2}{2(\sigma_2^2)^2}] \\
&\implies \frac{1}{2\hat{\sigma_2^2}}\Sigma_{i=1}^{n}(1 - E[Z_i]) = \frac{1}{2(\hat{\sigma_2^2})}\Sigma_{i=1}^{n}(1 - E[Z_i])(x_i - \mu_2^{(k)})^2 \\
&\implies \hat{\sigma_2^2} = \frac{\Sigma_{i=1}^{n}(1 - E[Z_i])(x_i - \mu_2^{(k)})^2}{\Sigma_{i=1}^{n}(1 - E[Z_i])} \\
&= \frac{1}{n - \eta^{(k)}}\Sigma_{i=1}^{n}(1 - \eta_i^{(k)})(x_i - \mu_2^{(k)})^2
\end{aligned}
$$

f)

Let `q` be a function that returns the expectation $Q(\theta|\theta^{(t)})$. We found the closed form of this equation in Part C above, so we simply code this function from that closed form. This gives us values for the vector $\hat{z}$

Let `q_updates` be a function that finds $\arg\max_\theta Q(\theta|\theta^{(t)})$ for vectors $\theta^{(t)}$ and $\hat{z}$. As we found each of the first derivative roots above, we simply code those closed form solutions into the function and return their calculated values.

Let `x` be the observations of the variable x.

Let `theta0` be a reasonable set of starting values for the elements of θ.

Let `epsilon` be the convergence value, such that when the Euclidean distance between values $\theta^{(t)}$ and $\theta^{(t+1)}$ are less than `epsilon`, the algorithm has converged, and stops iterating.

```
EM Algorithm(q, q_updates, x, theta0, epsilon)
  # Assign a value for theta_t that won't be convergent
  theta_t = theta0
  theta0 = theta_t + 1000

  # While convergence is not met
  while sum((theta0 - theta_t) ^ 2) < epsilon do
    # Update the value with the next iteration
    theta0 = theta_t

    # The expectation step
    zhat = q(theta0)

    # The maximization step
```

```
    theta_t = q_updates(theta0)

  return theta_t
```

Given the closed form solutions derived in earlier steps, the algorithm is extremely simple, as most of the heavy lifting has been done above manually with the calculus of the earlier steps.