4.1

Which lines correspond to the E-step, which to the M-step?

The following lines correspond to the E-step:

```
phiA = dnorm(yvar, mean = muA, sd = sdA)
phiB = dnorm(yvar, mean = muB, sd = sdB)
pA   = lambda * phiA
pB   = (1 - lambda) * phiB
ptot = pA + pB
pA   = pA / ptot
pB   = pB / ptot
```

The following lines correspond to the M-step:

```
lambda = mean(pA)
muA = weighted.mean(yvar, pA)
muB = weighted.mean(yvar, pB)
sdA = sqrt(weighted.mean((yvar - muA)^2, pA))
sdB = sqrt(weighted.mean((yvar - muB)^2, pB))
```

What does the M-step do, what does the E-step do?

In the soft-label EM algorithm, the E-step uses Bayes' Rule to gauge the probability that data point i belongs to group j using the metric $w_j^{(i)}$. Specifically:

$$w_j^{(i)} = p\left(z^{(i)} = j \,|x^{(i)};\, \phi, \mu, \Sigma\right) = \frac{p\left(x^{(i)} \mid z^{(i)}=j;\, \mu,\Sigma\right) \cdot p\left(z^{(i)}=j;\, \phi\right)}{\sum_{l=1}^{k} p\left(x^{(i)} \mid z^{(i)}=l;\, \mu,\Sigma\right) \cdot p\left(z^{(i)}=j;\, \phi\right)} \quad [1]$$

The M-step then uses the updated $w_j^{(i)}$s to update parameters of the gaussian distribution following the maximum-likelihood estimation of mean and variance. Specifically:

$$\phi_j = \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)};\;\; \mu_j = \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}};\;\; \Sigma_j = \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)}-\mu_j)(x^{(i)}-\mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}} \quad [1]$$

The EM algorithm can be proven to converge to at least a local maximum using Jensen's inequality [1].

Why do we need to compute loglik?

Because we need to compare the new and old log likelihood and calculate their difference. If the difference is smaller than the tolerance level, then we halt the program and conclude the algorithm has converged. This protects us from running an infinite loop.

Compare the result of what we are doing here to the output of the normalmixEM function from the mixtools package.

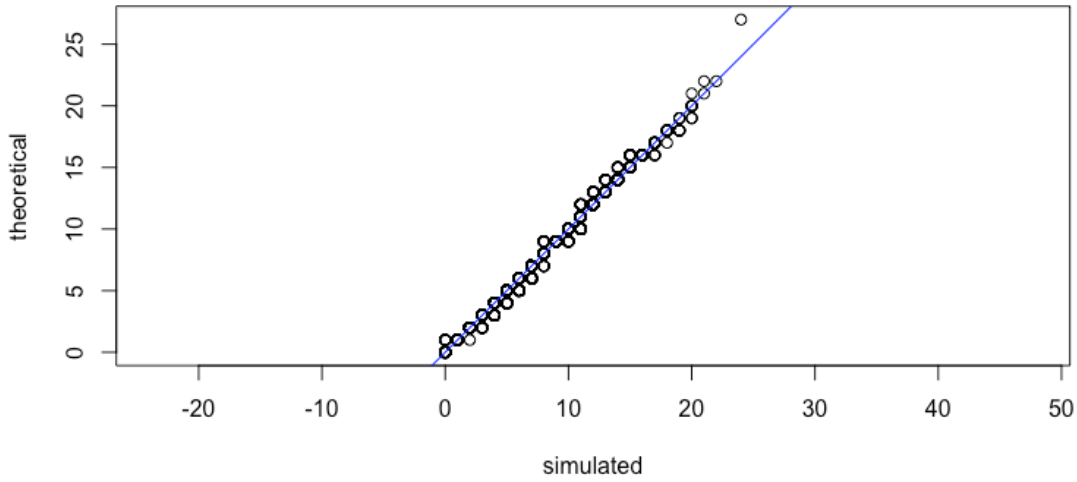
The result from *normalmixEM matches that from the textbook:*

```
> t(data.frame(results$mu,results$sigma))
                      [,1]      [,2]
results.mu    -0.16935914 0.1473216
results.sigma  0.09826529 0.1497879
```

R Code:

```
#4.1
library(mixtools)
yvar <- readRDS("../data/Myst.rds")$yvar
results <- mixtools::normalmixEM(yvar)
t(data.frame(results$mu,results$sigma))
```

4.2



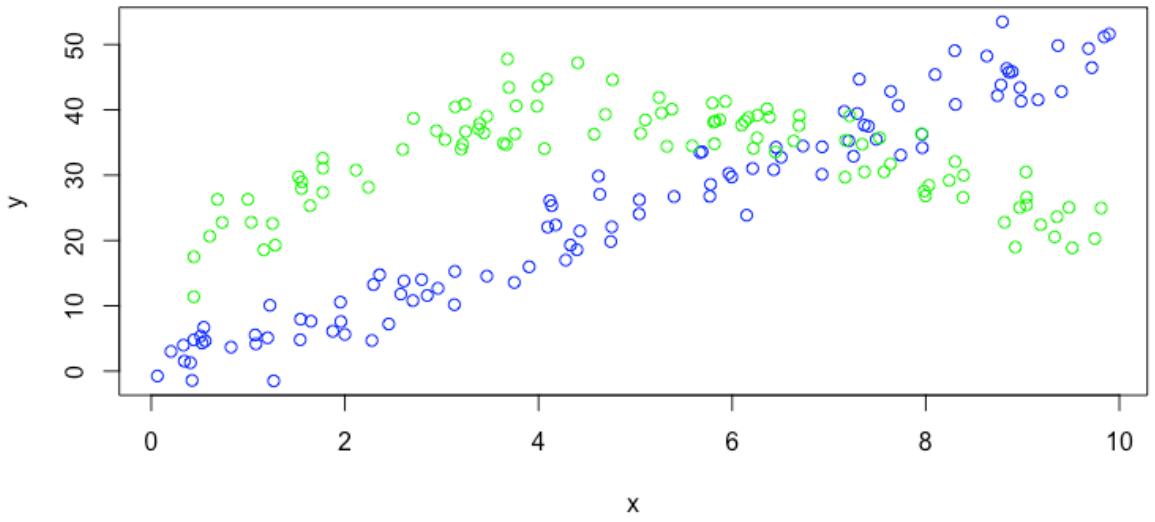
As shown above, the two distributions match closely.

R Code:

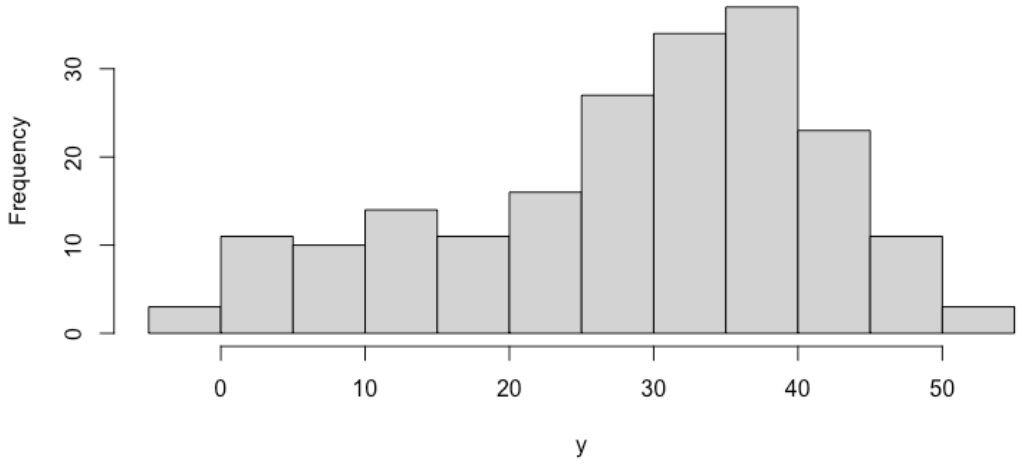
```
#original distribution
lambda <- rgamma(10000, shape = 10, rate = 3/2)
distribution1 <- rpois(length(lambda), lambda = lambda)

#theoretical distribution generated from parameters obtained from fitting in 4.4.3
p_size <- 10.06276
p_prob <- 0.6038492
distribution2 <- rnbinom(10000, p_size, p_prob)

qqplot(distribution1,distribution2, asp = 1, xlab = "simulated", ylab = "theoretical")
abline(a = 0, b = 1, col="blue")
```

4.3

(a)



From plotting data points from the two classes separately, it is evident that data points from *class = 1* have a linear relationship and data points from *class = 1* have a semi-quadratic relationship.



From the histogram of *yn* alone, it is difficult to reach the conclusion that the population is made of two mixtures.

(b)

```
m1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)
m1
```

(c)

The estimated parameters of the mixture components are as follows:

```
> m1@components
$Comp.1
$Comp.1[[1]]
$coef
(Intercept)           x      I(x^2)
-0.20991024  4.81742390  0.03620068

$sigma
[1] 3.475761



$Comp.2
$Comp.2[[1]]
$coef
(Intercept)           x      I(x^2)
 14.7165136   9.8465004  -0.9683288

$sigma
[1] 3.480335
```

This matches our expectation from observing the plot in part *(a)*. The blue part is component 1 and will have its intercept at −0.2099. Its linear coefficient is significantly larger than its quadratic counterpart, hence it is mostly linear in shape. On the other hand, the green part is component 2 and will have its intercept at 14.7165. Its quadratic coefficient is non-negligible and its negative value further confirms its concave shape.

The truth table is as follows:

```
> truthTable
             x         yn class cluster
1   4.17663259 22.3803787     1       1
2   1.20163055  5.1115746     1       1
3   2.29500635 13.2510576     1       1
4   5.96586786 30.2852404     1       1
5   2.35808338 14.7645078     1       1
6   7.63706104 42.8337601     1       1
7   6.45246460 34.2667327     1       1
8   9.40163061 42.8138796     1       1
9   8.29927115 49.0659954     1       1
10  4.33053110 19.3005980     1       1
11  4.61978652 29.8703835     1       1
12  9.71513912 46.4716001     1       1
13  2.27872578  4.6923751     1       1
14  6.92897805 34.3449694     1       1
15  7.25327696 32.9045352     1       2
16  0.40434891  1.2872814     1       1
17  9.36491266 49.8382514     1       1
18  2.57714695 11.8085276     1       1
19  7.74066373 33.0757635     1       2
20  9.89474383 51.6069735     1       1
21  0.55744518  4.6421303     1       1
22  1.54234713  7.9628999     1       1
23  8.09633770 45.4036015     1       1
24  1.08087069  4.1806895     1       1
25  3.13088989 10.1874921     1       1
26  7.36439054 37.6754354     1       1
27  8.85874283 45.7929465     1       1
28  5.66849986 33.4717028     1       1
29  6.50576695 32.7460204     1       1
30  1.53762205  4.8097527     1       1
```

The clustering algorithm correctly matched 95% of the datapoints from

mean(abs(truthTable$class - truthTable$cluster))

The summary of object *m1* shows that the log likelihood is negative enough to reject the null hypothesis that the two clusters come from the same underlying distribution. In other words, we can be fairly certain that the data is made of two distinct mixtures generated from $y = ax + bx^2$.

```
> summary(m1)

Call:
flexmix(formula = yn ~ x + I(x^2), data = NPreg, k = 2)

       prior size post>0 ratio
Comp.1 0.506  100    141 0.709
Comp.2 0.494  100    145 0.690

'log Lik.' -642.5451 (df=9)
AIC: 1303.09   BIC: 1332.775
```

(d)

As shown in the plot above, the clustering matches closely to what is reported in part *(a)*.

R Code:

```
#4.3
library("flexmix")
```

```
data("NPreg")

#a
NPreg1 <- dplyr::filter(NPreg,class==1)
NPreg2 <- dplyr::filter(NPreg,class==2)

plot(NPreg1$x,NPreg1$yn,col="blue",xlab = "x", ylab = "y")
points(NPreg2$x,NPreg2$yn,col="green")
hist(NPreg$yn,main = "",xlab="y")

#b
m1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)
m1

#c
m1@components
NPreg_clustered <- data.frame(NPreg, "cluster" = as.factor(clusters(m1)))
truthTable <- NPreg_clustered[,c("x","yn","class","cluster")]
class(truthTable)
mean(abs(truthTable$class - truthTable$cluster))
summary(m1)

#d
library("ggplot2")
ggplot(data = truthTable, aes(x = x, y = yn)) + geom_point(aes(color=cluster)) + labs(x="x",y="y")
```

References

[1] A. Ng, “Lecture 14 - Expectation-Maximization Algorithms | Stanford CS229: Machine Learning (Autumn 2018) - YouTube,” 2017. [Online]. Available: https://www.youtube.com/watch?v=rVfZHWTwXSA. [Accessed: 14-Jun-2020].