MAST30027: Modern Applied Statistics

Assignment 2 Solution 2021

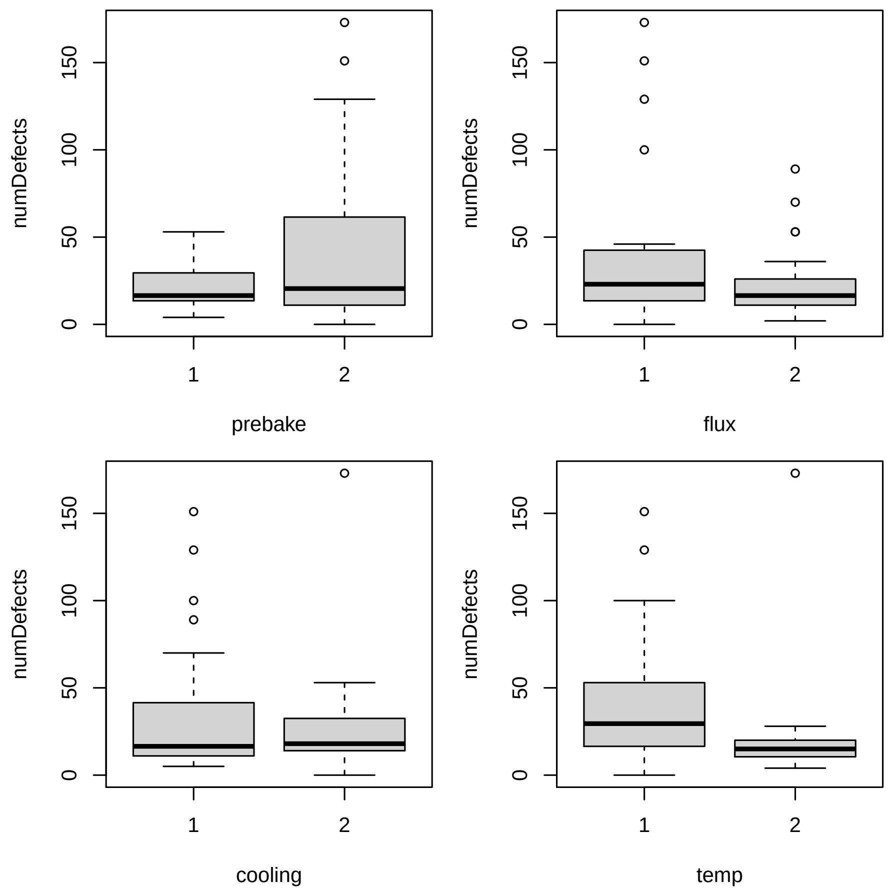
1. **Solution: There is no unique answer for this problem. This is a sample solution.**

   You can read the data using the following command.

```
> data <- read.table(file ="assignment2_prob1_2021.txt", header=TRUE)
> dim(data)

[1] 48  5

> names(data)

[1] "numDefects" "prebake"    "flux"       "cooling"    "temp"

> data$prebake <- factor(data$prebake)
> data$flux <- factor(data$flux)
> data$cooling <- factor(data$cooling)
> data$temp <- factor(data$temp)
```

   First, we visualize data by plotting the responses against the predictors.
   (Predictors are factor, so we don't need to check linearity.)

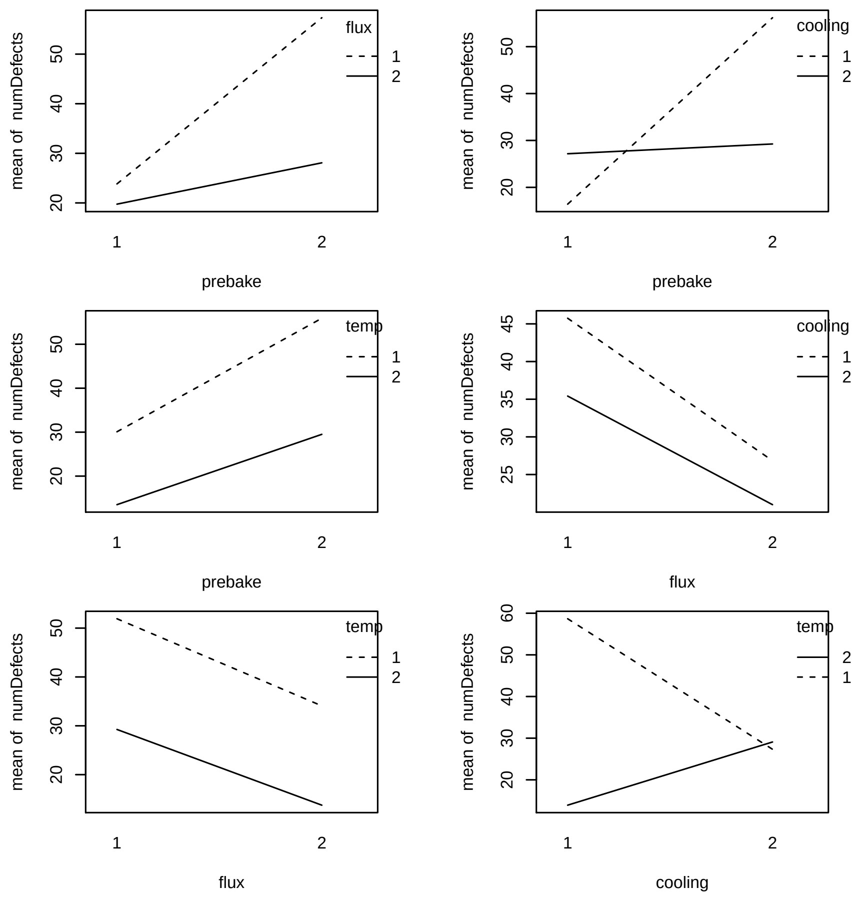
```
> par(mfrow=c(2,2), mar=c(4,4,1,1))
> plot(numDefects ~ prebake, data)
> plot(numDefects ~ flux, data)
> plot(numDefects ~ cooling, data)
> plot(numDefects ~ temp, data)
```



The variable `temp` has a clear effect on the number of defects, but for the other variables things are less clear.

Are there any two way interactions between factors? We use interaction plots to check.

```
> par(mfrow=c(3,2))
> par(mar=c(4,4,1,4))
> with(data, interaction.plot(prebake,flux,numDefects))
> with(data, interaction.plot(prebake,cooling,numDefects))
> with(data, interaction.plot(prebake,temp,numDefects))
> with(data, interaction.plot(flux,cooling,numDefects))
> with(data, interaction.plot(flux,temp,numDefects))
> with(data, interaction.plot(cooling,temp,numDefects))
```



There are clear interactions between `cooling` and `prebake`, and between `temp` and `cooling`.

Given that we have count data, a Poisson regression model with a log link is a good place to start. Since there are interactions, we use the main effects and all two-way interactions at once, then using stepwise selection with the AIC to select a parsimonious model.

```
> pmod <- glm(numDefects ~ (prebake + flux + cooling + temp)^2,
+             family=poisson, data=data)
> pmod.f <- step(pmod, trace=0)
> summary(pmod.f)

Call:
glm(formula = numDefects ~ prebake + flux + cooling + temp +
    prebake:flux + prebake:cooling + prebake:temp + flux:temp +
    cooling:temp, family = poisson, data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.7921  -2.6541  -0.2946   1.3936  13.5042

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        3.40957    0.08598  39.653  < 2e-16 ***
prebake2           1.33040    0.09625  13.823  < 2e-16 ***
flux2             -0.09016    0.09461  -0.953   0.3406
cooling2           0.07550    0.09794   0.771   0.4408
temp2             -1.86838    0.14883 -12.554  < 2e-16 ***
```

```
prebake2:flux2    -0.51723    0.11047  -4.682 2.84e-06 ***
prebake2:cooling2 -1.40241    0.12314 -11.389  < 2e-16 ***
prebake2:temp2     0.66403    0.13185   5.036 4.75e-07 ***
flux2:temp2       -0.31964    0.11465  -2.788   0.0053 **
cooling2:temp2     1.70784    0.12439  13.730  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1450.52  on 47  degrees of freedom
Residual deviance:  626.99  on 38  degrees of freedom
AIC: 877.73

Number of Fisher Scoring iterations: 5
```

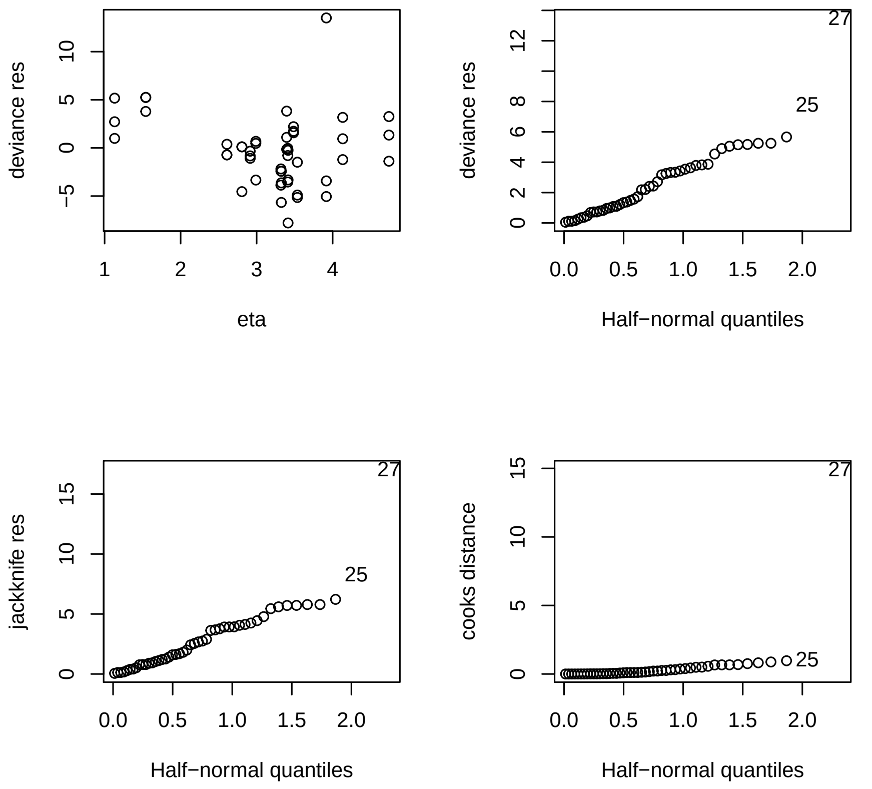
We see that the interaction term between `flux` and `cooling` was removed. We check the model adequacy.

```
> pchisq(deviance(pmod.f), pmod.f$df.residual, lower.tail=FALSE)
```

```
[1] 1.005731e-107
```

The p-value is very small, thus something is amiss with the model. The diagnostic plots might be able to tell us what is wrong.

```
> library(faraway)
> par(mfrow=c(2,2))
> plot(predict(pmod.f), residuals(pmod.f), xlab="eta", ylab="deviance res")
> halfnorm(residuals(pmod.f), ylab="deviance res")
> halfnorm(rstudent(pmod.f), ylab="jackknife res")
> halfnorm(cooks.distance(pmod.f), ylab="cooks distance")
```



Point 27 has very large Cook's distance, jackknife residual, and deviance residual, suggesting that this point might be an influential point. Thus, we will fit the model without the point 27.

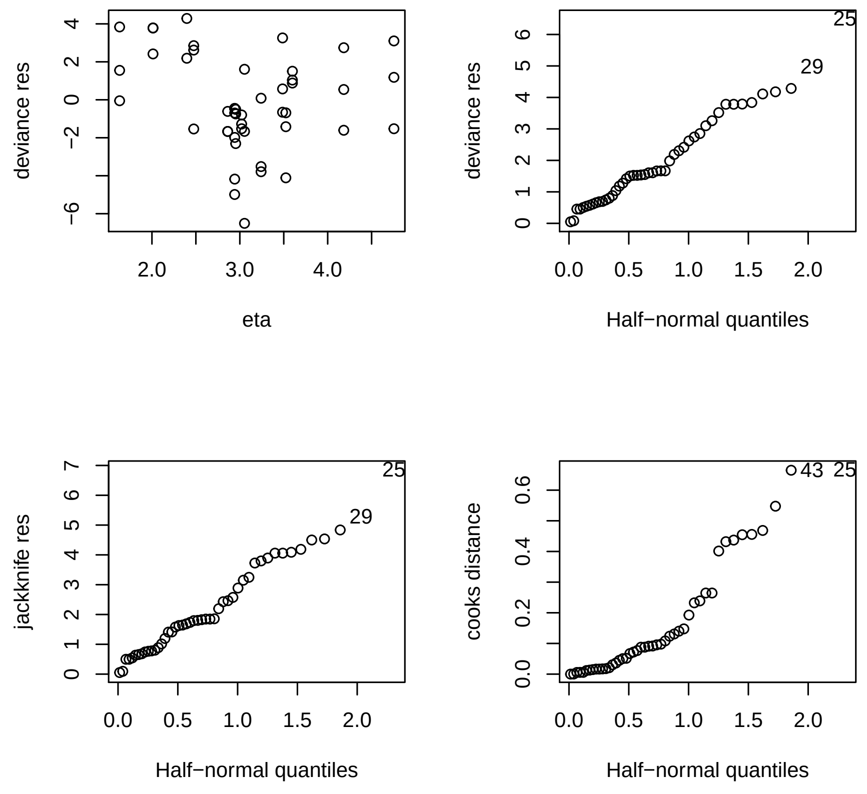
```
> pmod2 <- glm(numDefects ~ (prebake + flux + cooling + temp)^2,
+              family=poisson, subset=-27, data=data)
> pmod2.f <- step(pmod2, trace=0)
> summary(pmod2.f)

Call:
glm(formula = numDefects ~ prebake + flux + cooling + temp +
    prebake:cooling + flux:cooling + flux:temp + cooling:temp,
    family = poisson, data = data, subset = -27)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.5090  -1.5709  -0.5039   1.5787   4.2854

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        3.52381    0.07795  45.205  < 2e-16 ***
prebake2           1.23003    0.08099  15.187  < 2e-16 ***
flux2             -0.57083    0.07473  -7.638 2.20e-14 ***
cooling2           0.07465    0.10984   0.680    0.497
temp2             -1.51196    0.10010 -15.104  < 2e-16 ***
prebake2:cooling2 -1.77525    0.12359 -14.364  < 2e-16 ***
flux2:cooling2     0.45881    0.11527   3.980 6.88e-05 ***
```

```
flux2:temp2         0.19085    0.12844   1.486    0.137
cooling2:temp2      0.85535    0.12840   6.662 2.71e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1137.32  on 46  degrees of freedom
Residual deviance:  289.02  on 38  degrees of freedom
AIC: 530.77

Number of Fisher Scoring iterations: 5
```

```
> pchisq(deviance(pmod2.f), pmod2.f$df.residual, lower.tail=FALSE)
```

```
[1] 2.343993e-40
```

```
> par(mfrow=c(2,2))
> plot(predict(pmod2.f), residuals(pmod2.f), xlab="eta", ylab="deviance res")
> halfnorm(residuals(pmod2.f), ylab="deviance res")
> halfnorm(rstudent(pmod2.f), ylab="jackknife res")
> halfnorm(cooks.distance(pmod2.f), ylab="cooks distance")
```



The diagnostic plots look fine now. However, we still observe that p-value of model adequacy test is low. Let’s check overdispersion.

```
> (phi <- sum(residuals(pmod2.f, type="pearson")^2/pmod2.f$df.residual))
```

```
[1] 7.355161
```

The large value of $\hat{\phi}$ suggests that overdispersion occurs. Thus, we fit the data by quasi-Poisson model.

```
> pmod.q <- glm(numDefects ~ prebake + flux + cooling + temp + prebake:cooling +
+               flux:cooling + flux:temp + cooling:temp,
+               family=quasipoisson, subset=-27, data=data)
> summary(pmod.q)

Call:
glm(formula = numDefects ~ prebake + flux + cooling + temp +
    prebake:cooling + flux:cooling + flux:temp + cooling:temp,
    family = quasipoisson, data = data, subset = -27)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.5090  -1.5709  -0.5039   1.5787   4.2854

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        3.52381    0.21141  16.668  < 2e-16 ***
prebake2           1.23003    0.21966   5.600 2.02e-06 ***
flux2             -0.57083    0.20268  -2.816  0.00766 **
cooling2           0.07465    0.29790   0.251  0.80349
temp2             -1.51196    0.27148  -5.569 2.22e-06 ***
prebake2:cooling2 -1.77525    0.33519  -5.296 5.25e-06 ***
flux2:cooling2     0.45881    0.31261   1.468  0.15042
flux2:temp2        0.19085    0.34833   0.548  0.58696
cooling2:temp2     0.85535    0.34822   2.456  0.01872 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 7.355163)

    Null deviance: 1137.32  on 46  degrees of freedom
Residual deviance:  289.02  on 38  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

From the p-values in the summary table, `flux2:cooling2` and `flux2:temp2` are not significant. So we will consider a model without these two terms and compare the two models using F-test.

```
> pmod.q.r <- glm(numDefects ~ prebake + flux + cooling + temp + prebake:cooling +
+                   cooling:temp, family=quasipoisson, subset=-27, data=data)
> anova(pmod.q.r, pmod.q, test="F")

Analysis of Deviance Table

Model 1: numDefects ~ prebake + flux + cooling + temp + prebake:cooling +
    cooling:temp
Model 2: numDefects ~ prebake + flux + cooling + temp + prebake:cooling +
    flux:cooling + flux:temp + cooling:temp
  Resid. Df Resid. Dev Df Deviance    F Pr(>F)
```

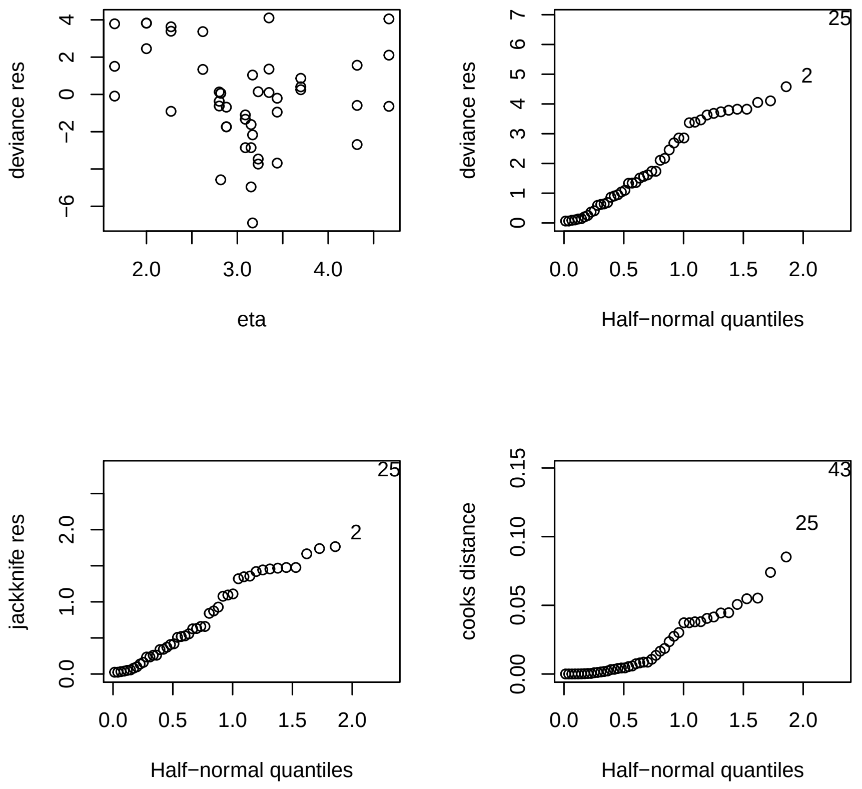
```
1        40     309.76
2        38     289.02  2   20.741 1.41 0.2566
```

The test shows that we should remove these two terms.

```
> summary(pmod.q.r)

Call:
glm(formula = numDefects ~ prebake + flux + cooling + temp +
    prebake:cooling + cooling:temp, family = quasipoisson, data = data,
    subset = -27)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-6.893  -1.676  -0.090   1.432   4.104

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.4384     0.2091  16.448  < 2e-16 ***
prebake2            1.2300     0.2212   5.561 1.95e-06 ***
flux2              -0.3502     0.1494  -2.344   0.0241 *
cooling2            0.2601     0.2675   0.972   0.3367
temp2              -1.4388     0.2351  -6.121 3.19e-07 ***
prebake2:cooling2  -1.7608     0.3375  -5.218 5.90e-06 ***
cooling2:temp2      0.8914     0.3472   2.567   0.0141 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 7.458044)

    Null deviance: 1137.32  on 46  degrees of freedom
Residual deviance:  309.76  on 40  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

Now the `summary` looks fine. Note the main effect `cooling` is not significant. However, the interaction terms contains `cooling`. Hence, we will keep the main effect.

```
> par(mfrow=c(2,2))
> plot(predict(pmod.q.r), residuals(pmod.q.r), xlab="eta", ylab="deviance res")
> halfnorm(residuals(pmod.q.r), ylab="deviance res")
> halfnorm(rstudent(pmod.q.r), ylab="jackknife res")
> halfnorm(cooks.distance(pmod.q.r), ylab="cooks distance")
```



The diagnostic plots of the resulting model also look fine.

`flux` is the only main effect which does not interact with any other factors. It has a negative effect on the number of defects since its coefficient is negative. `prebake` has a positive main effect on the number of defects. However, the coefficient of the interaction between `prebake` and `cooling` is negative and the effect is larger than the main effect of `prebake`. This means that the effect of `prebake` is negative when `cooling = 2`. (Other interpretations are acceptable as long as they are well explained.)

2. **Solution:**

   (a) Let $\theta = (\pi_1, \pi_2, \lambda_1, \lambda_2, \lambda_3)$ and $n = 300$.

$$\begin{aligned} p(X_1, ...X_n, Z_1, ..., Z_n|\theta) &= \prod_{i=1}^{n} p(X_i|Z_i, \theta)p(Z_i|\theta) \\ &= \prod_{i=1}^{n}\prod_{k=1}^{3}[p(X_i|Z_i = k, \theta)p(Z_i = k|\theta)]^{I_{(Z_i=k)}}. \end{aligned}$$

$$\log[p(X_1, ...X_n, Z_1, ..., Z_n|\theta)] = \sum_{i=1}^{n}\sum_{k=1}^{3} I_{(Z_i=k)}[\log p(X_i|Z_i = k, \theta) + \log p(Z_i = k|\theta)].$$

$$
\begin{aligned}
Q(\theta,\theta^0) &= E_{Z|X,\theta^0}[\log p(X_1,...X_n,Z_1,...,Z_n|\theta)]\\
&= \sum_{i=1}^{n}\sum_{k=1}^{3} p(Z_i=k|X_i,\theta^0)[\log p(X_i|Z_i=k,\theta)+\log p(Z_i=k|\theta)]\\
&= \sum_{i=1}^{n}\sum_{k=1}^{3} p(Z_i=k|X_i,\theta^0)[X_i\log\lambda_k-\lambda_k-\log X_i!+\log\pi_k],
\end{aligned}
$$

where $\pi_3 = 1-\pi_1-\pi_2$.

(b) Let $\theta^0 = (\pi_1^0,\pi_2^0,\lambda_1^0,\lambda_2^0,\lambda_3^0)$.
E-step: For $k\in\{1,2\}$,

$$
\begin{aligned}
p(Z_i=k|X_i,\theta^0) &= \frac{p(Z_i=k,X_i|\theta^0)}{p(X_i|\theta^0)}\\
&= \frac{p(X_i|Z_i=k,\theta^0)p(Z_i=k|\theta^0)}{\sum_{s=1}^{3}p(X_i|Z_i=s,\theta^0)p(Z_i=s|\theta^0)}\\
p(Z_i=3|X_i,\theta^0) &= 1-p(Z_i=1|X_i,\theta^0)-p(Z_i=2|X_i,\theta^0),
\end{aligned}
$$

where $p(X_i|Z_i=k,\theta^0)=\frac{(\lambda_k^0)^{X_i}e^{-\lambda_k^0}}{X_i!}$, $p(Z_i=k|\theta^0)=\pi_k^0$ and $p(Z_i=3|\theta^0)=1-\pi_1^0-\pi_2^0$.

(c) M-step:

$$
\begin{aligned}
\frac{\partial Q(\theta,\theta^0)}{\partial\pi_1} &= \sum_{i=1}^{n}\Big[\frac{p(Z_i=1|X_i,\theta^0)}{\pi_1}-\frac{p(Z_i=3|X_i,\theta^0)}{1-\pi_1-\pi_2}\Big]\\
&= \frac{(1-\pi_1-\pi_2)\sum_{i=1}^{n}p(Z_i=1|X_i,\theta^0)-\pi_1\sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0)}{\pi_1(1-\pi_1-\pi_2)}=0 \quad (1)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial Q(\theta,\theta^0)}{\partial\pi_2} &= \sum_{i=1}^{n}\Big[\frac{p(Z_i=2|X_i,\theta^0)}{\pi_2}-\frac{p(Z_i=3|X_i,\theta^0)}{1-\pi_1-\pi_2}\Big]\\
&= \frac{(1-\pi_1-\pi_2)\sum_{i=1}^{n}p(Z_i=2|X_i,\theta^0)-\pi_2\sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0)}{\pi_1(1-\pi_1-\pi_2)}=0. \quad (2)
\end{aligned}
$$

Let $\pi_3=1-\pi_1-\pi_2$. From (1) and (2), we obtain

$$
\begin{aligned}
\sum_{i=1}^{n}p(Z_i=1|X_i,\theta^0)\pi_3 &= \sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0)\pi_1, \quad (3)\\
\sum_{i=1}^{n}p(Z_i=2|X_i,\theta^0)\pi_3 &= \sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0)\pi_2. \quad (4)
\end{aligned}
$$

Taking sum of (3) and (4), we have

$$
\pi_3\sum_{i=1}^{n}\Big[p(Z_i=1|X_i,\theta^0)+p(Z_i=2|X_i,\theta^0)\Big]=(1-\pi_3)\sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0),
$$

$$
\hat{\pi}_3=\frac{\sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0)}{n}.
$$

From (3), we have

$$
\hat{\pi}_1=\frac{1}{\sum_{i=1}^{n}p(Z_i=3|X_i,\theta^0)}\hat{\pi}_3\sum_{i}p(Z_i=1|X_i,\theta^0)=\frac{\sum_{i=1}^{n}p(Z_i=1|X_i,\theta^0)}{n}.
$$

From (4), we have

$$\hat{\pi}_2 = \frac{1}{\sum_{i=1}^n p(Z_i = 3|X_i, \theta^0)} \hat{\pi}_3 \sum_i^n p(Z_i = 2|X_i, \theta^0) = \frac{\sum_{i=1}^n p(Z_i = 2|X_i, \theta^0)}{n}.$$

For $k = 1, 2, 3$,

$$\begin{aligned}\frac{\partial Q(\theta, \theta^0)}{\partial \lambda_k} &= \sum_{i=1}^n p(Z_i = k|X_i, \theta^0)\Big[\frac{X_i}{\lambda_k} - 1\Big] \\ &= \sum_{i=1}^n p(Z_i = k|X_i, \theta^0)\frac{X_i}{\lambda_k} - \sum_{i=1}^n p(Z_i = k|X_i, \theta^0) = 0. \\ \hat{\lambda}_k &= \frac{\sum_{i=1}^n p(Z_i = k|X_i, \theta^0)X_i}{\sum_{i=1}^n p(Z_i = k|X_i, \theta^0)}.\end{aligned}$$

(d) Implement the EM algorithm.

```
> # w.init : initial value for pi
> # lambda.init : initial value for lambda
> # epsilon : stop if the change of the incomplete log-likelihood is less than epsilon
> # max.iter : maximum number of EM-iterations
> mixture.EM <- function(X, w.init, lambda.init, epsilon=1e-5, max.iter=100) {
+
+   w.curr = w.init
+   lambda.curr = lambda.init
+
+   # store incomplete log-likehoods for each iteration
+   log_liks = c()
+
+   # compute incomplete log-likehoods using initial values of parameters.
+   log_liks = c(log_liks, compute.log.lik(X, w.curr, lambda.curr)$ill)
+
+   # set the change in incomplete log-likelihood with 1
+   delta.ll = 1
+
+   # number of iteration
+   n.iter = 1
+
+   # If the log-likelihood has changed by less than epsilon, EM will stop.
+   while((delta.ll > epsilon) & (n.iter <= max.iter)){
+
+     # run EM step
+     EM.out = EM.iter(X, w.curr, lambda.curr)
+
+     # replace the current value with the new parameter estimate
+     w.curr = EM.out$w.new
+     lambda.curr = EM.out$lambda.new
+
+     # incomplete log-likehoods with new parameter estimate
+     log_liks = c(log_liks, compute.log.lik(X, w.curr, lambda.curr)$ill)
+
+     # compute the change in incomplete log-likelihood
+     delta.ll = log_liks[length(log_liks)]  - log_liks[length(log_liks)-1]
+
+     # increase the number of iteration
+     n.iter = n.iter + 1
+
+   }
+
+   return(list(w.curr=w.curr, lambda.curr=lambda.curr, log_liks=log_liks))
+
+ }
> EM.iter <- function(X, w.curr, lambda.curr) {
+
+   # E-step: compute E_{Z|X,\theta_0}[I(Z_i = k)]
+
+   # for each sample $X_i$, compute $P(X_i, Z_i=k)$
+   prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z
+
+   # compute P(Z_i=k | X_i)
+   P_ik = prob.x.z / rowSums(prob.x.z)
+
+   # M-step
+   w.new = colSums(P_ik)/sum(P_ik)  # sum(P_ik) is equivalent to sample size
+   lambda.new = colSums(P_ik*X)/colSums(P_ik)
+
+   return(list(w.new=w.new, lambda.new=lambda.new))
+ }
> # for each sample $X_i$, compute $P(X_i, Z_i=k)$
> compute.prob.x.z <- function(X, w.curr, lambda.curr) {
+
+   # for each sample $X_i$, compute $P(X_i, Z_i=k)$.
+   # store these values in the columns of L:
+   L = matrix(NA, nrow=length(X), ncol= length(w.curr))
+   for(k in seq_len(ncol(L))) {
+     L[, k] = dpois(X, lambda.curr[k])*w.curr[k]
+   }
+
+   return(list(prob.x.z=L))
+ }
> # Compute incomplete log-likehoods
> compute.log.lik <- function(X, w.curr, lambda.curr) {
+
+   # for each sample $X_i$, compute $P(X_i, Z_i=k)$
+   prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z
+
+   # incomplete log-likehoods
+   ill = sum(log(rowSums(prob.x.z)))
+
+   return(list(ill=ill))
+ }
```

Run the EM algorithm two times with the two initial values provided in the problem.

```
> # read the data
> X = scan(file="assignment2_prob2_2021.txt", what=double())
> EM1 <- mixture.EM(X, w.init=c(0.3,0.3,0.4), lambda.init=c(3, 20, 35),
+                   epsilon=1e-5, max.iter=100)
> EM2 <- mixture.EM(X, w.init=c(0.1,0.2,0.7), lambda.init=c(5, 25, 40),
+                   epsilon=1e-5, max.iter=100)
> check = rbind(c(EM1$w.curr, EM1$lambda.curr),
+                c(EM2$w.curr, EM2$lambda.curr))
> colnames(check) = c('pi_1', 'pi_2', 'pi_3', 'lambda_1', 'lambda_2', 'lambda_3')
> rownames(check) = c('EM1', 'EM2')
> check
```

```
         pi_1      pi_2      pi_3 lambda_1 lambda_2 lambda_3
EM1 0.2491314 0.2497808 0.5010877 5.167479 18.09393 36.93918
EM2 0.2491307 0.2497786 0.5010907 5.167466 18.09381 36.93911
```

```
> print(EM1$log_liks[length(EM1$log_liks)], digits=16)
```

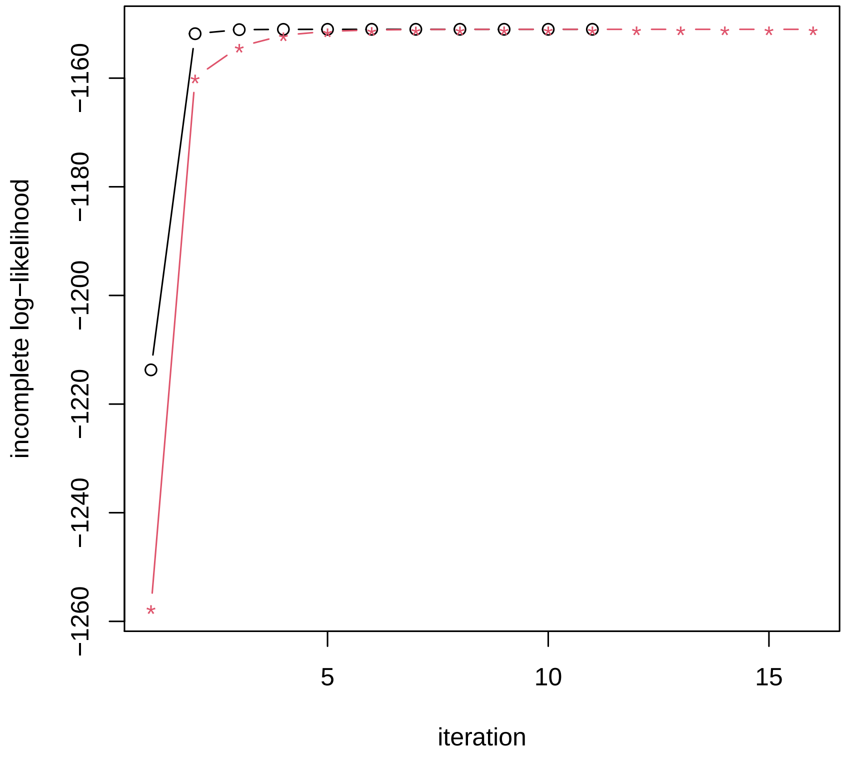
```
[1] -1151.01487128212
```

```
> print(EM2$log_liks[length(EM2$log_liks)], digits=16)
```

```
[1] -1151.014870863243
```

Estimates from the two EM runs are very similar and they have very similar incomplete log-likelihoods. So I will report one from the second run as it has slightly higher incomplete log-likelihoods. MLEs for the parameters are $\hat{\pi}_1 = 0.2491307, \hat{\pi}_2 = 0.2497786, \hat{\lambda}_1 = 5.167466, \hat{\lambda}_2 = 18.09381, \hat{\lambda}_3 = 36.93911$.
Check that the incomplete log-likelihoods increases at each step by plotting them

```
> plot(EM1$log_liks, type='b',
+      xlim=c(1,max(length(EM1$log_liks), length(EM2$log_liks))),
+      ylim=range(c(EM1$log_liks, EM2$log_liks)),
+      ylab='incomplete log-likelihood', xlab='iteration')
> points(EM2$log_liks, type='b', pch='*', col=2)
```



3. **Solution:**

(a) Let $\theta = (\pi_1, \pi_2, \lambda_1, \lambda_2, \lambda_3)$, $n = 300$ and $n' = 100$.

$$\begin{aligned}
&p(X_1, ...X_n, Z_1, ..., Z_n, X_{n+1}, ..., X_{n+n'}|\theta)\\
=&\prod_{i=1}^{n} p(X_i|Z_i, \theta)p(Z_i|\theta) \prod_{i=n+1}^{n+n'} p(X_i|\theta)\\
=&\prod_{i=1}^{n}\prod_{k=1}^{3}[p(X_i|Z_i = k, \theta)p(Z_i = k|\theta)]^{I_{(Z_i=k)}} \prod_{i=n+1}^{n+n'} p(X_i|\theta).
\end{aligned}$$

$$\begin{aligned}
&\log[p(X_1, ...X_n, Z_1, ..., Z_n, X_{n+1}, ..., X_{n+n'}|\theta)]\\
=&\sum_{i=1}^{n}\sum_{k=1}^{3} I_{(Z_i=k)}[\log p(X_i|Z_i = k, \theta) + \log p(Z_i = k|\theta)] + \sum_{i=n+1}^{n+n'} \log p(X_i|\theta)
\end{aligned}$$

$$\begin{aligned}
Q(\theta, \theta^0) =& E_{Z|X,\theta^0}[\log p(X_1, ...X_n, Z_1, ..., Z_n, X_{n+1}, ..., X_{n+n'}|\theta)]\\
=&\sum_{i=1}^{n}\sum_{k=1}^{3} p(Z_i = k|X_i, \theta^0)[\log p(X_i|Z_i = k, \theta) + \log p(Z_i = k|\theta)] + \sum_{i=n+1}^{n+n'} \log p(X_i|\theta)\\
=&\sum_{i=1}^{n}\sum_{k=1}^{3} p(Z_i = k|X_i, \theta^0)[X_i \log \lambda_k - \lambda_k - \log X_i! + \log \pi_k] + \sum_{i=n+1}^{n+n'} [X_i \log \lambda_2 - \lambda_2 - \log X_i!],
\end{aligned}$$

where $\pi_3 = 1 - \pi_1 - \pi_2$.

(b) E-step: same as the E-step in the solution for the problem 2 (b).
M-step: same as the M-step in the solution for the problem 2 (c) except for the following:

$$\begin{aligned}
\frac{\partial Q(\theta, \theta^0)}{\partial \lambda_2} =& \sum_{i=1}^{n} p(Z_i = 2|X_i, \theta^0)\Big[\frac{X_i}{\lambda_2} - 1\Big] + \sum_{i=n+1}^{n+n'} \Big[\frac{X_i}{\lambda_2} - 1\Big]\\
=& \sum_{i=1}^{n} p(Z_i = 2|X_i, \theta^0)\frac{X_i}{\lambda_2} - \sum_{i=1}^{n} p(Z_i = 2|X_i, \theta^0) + \sum_{i=n+1}^{n+n'} \frac{X_i}{\lambda_2} - n' = 0.
\end{aligned}$$

Hence,

$$\hat{\lambda}_2 = \frac{\sum_{i=1}^{n} p(Z_i = 2|X_i, \theta^0)X_i + \sum_{i=n+1}^{n+n'} X_i}{\sum_{i=1}^{n} p(Z_i = 2|X_i, \theta^0) + n'}.$$

(c) Implement the EM algorithm.

```
> # X  : X_1, ..., X_300  which follow a mixture of Poisson distribution
> # X0 : X_301, .., X_400 which follow a Poisson distribution
> mixture.EM <- function(X, X0, w.init, lambda.init, epsilon=1e-5, max.iter=100) {
+
+   w.curr = w.init
+   lambda.curr = lambda.init
+
+   # store incomplete log-likehoods for each iteration
+   log_liks = c()
+
+   # compute incomplete log-likehoods using initial values of parameters.
+   log_liks = c(log_liks, compute.log.lik(X, X0, w.curr, lambda.curr)$ill)
+
+   # set the change in incomplete log-likelihood with 1
+   delta.ll = 1
+
+   # number of iteration
+   n.iter = 1
+
+   # If the log-likelihood has changed by less than epsilon, EM will stop.
+   while((delta.ll > epsilon) & (n.iter <= max.iter)){
+
+     # run EM step
+     EM.out = EM.iter(X, X0, w.curr, lambda.curr)
+
+     # replace the current value with the new parameter estimate
+     w.curr = EM.out$w.new
+     lambda.curr = EM.out$lambda.new
+
+     # incomplete log-likehoods with new parameter estimate
+     log_liks = c(log_liks, compute.log.lik(X, X0, w.curr, lambda.curr)$ill)
+
+     # compute the change in incomplete log-likelihood
+     delta.ll = log_liks[length(log_liks)]  - log_liks[length(log_liks)-1]
+
+     # increase the number of iteration
+     n.iter = n.iter + 1
+   }
+
+   return(list(w.curr=w.curr, lambda.curr=lambda.curr, log_liks=log_liks))
+
+ }
> EM.iter <- function(X, X0, w.curr, lambda.curr) {
+
+   # E-step: compute E_{Z|X,\theta_0}[I(Z_i = k)]
+
+   # for each sample $X_i$, compute $P(X_i, Z_i=k)$
+   prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z
+
+   # compute P(Z_i=k | X_i)
+   P_ik = prob.x.z / rowSums(prob.x.z)
+
+   # M-step
+   w.new = colSums(P_ik)/sum(P_ik)  # sum(P_ik) is equivalent to sample size
+   ### Change!!!
+   lambda.new = rep(NA, length(w.new))
+   lambda.new[c(1,3)] = (colSums(P_ik*X)/(colSums(P_ik)))[c(1,3)]
+   lambda.new[2] = (colSums(P_ik*X)[2] + sum(X0))/( colSums(P_ik)[2] + length(X0))
+
+   return(list(w.new=w.new, lambda.new=lambda.new))
+ }
> # for each sample $X_i$, compute $P(X_i, Z_i=k)$
> compute.prob.x.z <- function(X, w.curr, lambda.curr) {
+
+   # for each sample $X_i$, compute $P(X_i, Z_i=k)$.
+   # Store these values in the columns of L:
+   L = matrix(NA, nrow=length(X), ncol= length(w.curr))
+   for(k in seq_len(ncol(L))) {
+     L[, k] = dpois(X, lambda.curr[k])*w.curr[k]
+   }
+
+   return(list(prob.x.z=L))
+ }
> # Compute incomplete log-likehoods
> compute.log.lik <- function(X, X0, w.curr, lambda.curr) {
+
+   # for each sample $X_i$, compute $P(X_i, Z_i=k)$
+   prob.x.z = compute.prob.x.z(X, w.curr, lambda.curr)$prob.x.z
+
+   # incomplete log-likehoods
+   ill = sum(log(rowSums(prob.x.z))) + sum(log(dpois(X0, lambda.curr[2])))
+
+   return(list(ill=ill))
+ }
```

Run the EM algorithm two times with the two initial values provided in the problem.

```
> X0 = scan(file="assignment2_prob3_2021.txt", what=double())
> EM1 <- mixture.EM(X, X0, w.init=c(0.3,0.3,0.4), lambda.init=c(3, 20, 35),
+                   epsilon=1e-5, max.iter=100)
> EM2 <- mixture.EM(X, X0, w.init=c(0.1,0.2,0.7), lambda.init=c(5, 25, 40),
+                   epsilon=1e-5, max.iter=100)
> check = rbind(c(EM1$w.curr, EM1$lambda.curr),
+               c(EM2$w.curr, EM2$lambda.curr))
> colnames(check) = c('pi_1', 'pi_2', 'pi_3', 'lambda_1', 'lambda_2', 'lambda_3')
> rownames(check) = c('EM1', 'EM2')
> check
```

```
         pi_1      pi_2      pi_3 lambda_1 lambda_2 lambda_3
EM1 0.2461850 0.2447837 0.5090313 5.118187 17.36453 36.76509
EM2 0.2461848 0.2447753 0.5090399 5.118186 17.36442 36.76488
```

```
> print(EM1$log_liks[length(EM1$log_liks)], digits=16)
```

```
[1] -1437.052845923898
```

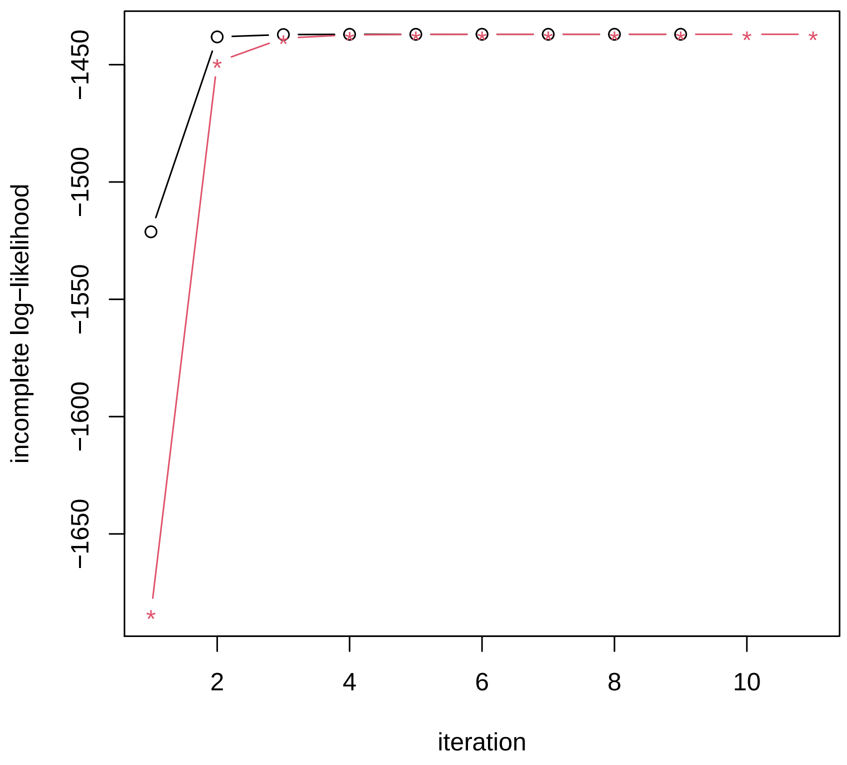
```
> print(EM2$log_liks[length(EM2$log_liks)], digits=16)
```

```
[1] -1437.052845374368
```

Estimates from the two EM runs are very similar and they have very similar incomplete log-likelihoods. So I will report one from the second run as it has slightly higher incomplete log-likelihoods. MLEs for the parameters are $\hat{\pi}_1 = 0.2461848, \hat{\pi}_2 = 0.2447753, \hat{\lambda}_1 = 5.118186, \hat{\lambda}_2 = 17.36442, \hat{\lambda}_3 = 36.76488$.

Check that the incomplete log-likelihoods increases at each step by plotting them

```
> plot(EM1$log_liks, type='b',
+      xlim=c(1,max(length(EM1$log_liks), length(EM2$log_liks))),
+      ylim=range(c(EM1$log_liks, EM2$log_liks)),
+      ylab='incomplete log-likelihood', xlab='iteration')
> points(EM2$log_liks, type='b', pch='*',  col=2)
```



incomplete log−likelihood
−1450
−1500
−1550
−1600
−1650
2
4
6
8
10
iteration

4. **NA.**