Assignment2

Zeyi Pan

4/15/2020

Q1

The posterior distribution is:
$\pi(\theta|x) = \frac{p(X=x|\theta)\pi(\theta)}{p(x)}$ where $p(x) = \sum p(x|\theta_i)\pi(\theta_i)$
As we know, $\pi(1/4) = \pi(1/2) = \pi(3/4) = 1/3$, when $X = 4$, we get
$\pi(\theta|4) = \frac{p(X=4|\theta)\pi(\theta)}{p(4|1/4)\pi(1/4)+p(4|1/2)\pi(1/2)+p(4|3/4)\pi(3/4)}$
Because $X \sim bin(n, \theta)$, $p(x|\theta) = \binom{10}{x}\theta^x(1-\theta)^{10-x}$.
When $\theta = 1/4$, $p(4|1/4) = \binom{10}{4}(1/4)^4(3/4)^6$ and $\pi(1/4|4) = 0.397$
When $\theta = 1/2$, $p(4|1/2) = \binom{10}{4}(1/2)^4(1/2)^6$ and $\pi(1/2|4) = 0.558$
When $\theta = 3/4$, $p(4|3/4) = \binom{10}{4}(3/4)^4(1/4)^6$ and $\pi(3/4|4) = 0.044$.

```
p1 = dbinom(4, size=10, prob=1/4)
p2 = dbinom(4, size=10, prob=1/2)
p3 = dbinom(4, size=10, prob=3/4)
pi_1 = p1/(p1+p2+p3)
pi_1
```

```
## [1] 0.3974918
```

```
pi_2 = p2/(p1+p2+p3)
pi_2
```

```
## [1] 0.5583424
```

```
pi_3 = p3/(p1+p2+p3)
pi_3
```

```
## [1] 0.04416576
```

Q2

(a)

As the number of vehicle V passed a toll station during time T has Poisson distribution, $p(V) = \frac{(\frac{\lambda T}{M})^V e^{-\frac{\lambda T}{M}}}{V!}$.
Therefore, the probability that no vehicle passed during time T is $p(0) = e^{-\frac{\lambda T}{M}}$, and the probability that at least one vehicle passed will be $1 - p(0) = 1 - e^{-\frac{\lambda T}{M}}$.
The number of toll stations whichi having at least one vehicle passed has a binomial distribution, therefore $f(x|\lambda, M) = \binom{N}{x}(1 - e^{-\frac{\lambda T}{M}})^x(e^{-\frac{\lambda T}{M}})^{N-x}$ where $N = 10, T = 15$ seconds $= 1/240$ hours.

(b)

As λ and M are independent, $\pi(\lambda,p) = \pi_\lambda(\lambda)\pi_M(M)$.Because λ represents number of vehicles and M represents number of toll stations, they are discrete variables and we use summation here rather than integral. And we consider λ and M as a pair, so the subscrip will be the same, which is denoted by i. $f(x|\lambda,M)$ is as in (a).

$\pi(\lambda,M|x) = \frac{f(x|\lambda,M)\pi(\lambda,M)}{\sum f(x|\lambda_i,M_i)\pi(\lambda_i,M_i)} = \frac{\binom{N}{x}(1-e^{-\frac{\lambda T}{M}})^x(e^{-\frac{\lambda T}{M}})^{N-x}\pi_\lambda(\lambda)\pi_M(M)}{\sum \binom{N}{x}(1-e^{-\frac{\lambda_i T}{M_i}})^x(e^{-\frac{\lambda_i T}{M_i}})^{N-x}\pi_\lambda(\lambda_i)\pi_M(M_i)}$

(c)

Both M and λ have asymetric proposal distribution.
The proposal markov chain for M is given by

$$Q(M_{new}|M_{old}) = \begin{cases} 1, & M_{new} = 9, M_{old} = 8 \text{ or } 10 \\ 1/2, & M_{new} = 8, M_{old} = 9 \\ 1/2, & M_{new} = 10, M_{old} = 9 \end{cases}$$

As it just indicates the transition between two states, therefore, we can get the following probabilities:
$Q(M_{new} = 9|M_{old} = 8) = 1, Q(M_{old} = 8|M_{new} = 9) = 1/2$
$Q(M_{new} = 8|M_{old} = 9) = 1/2, Q(M_{old} = 9|M_{new} = 8) = 1$
$Q(M_{new} = 9|M_{old} = 10) = 1, Q(M_{old} = 10|M_{new} = 9) = 1/2$
$Q(M_{new} = 10|M_{old} = 9) = 1/2, Q(M_{old} = 9|M_{new} = 10) = 1$
As $\lambda_{new} = \lambda_{old} + Uniform(-10, 10)$ and there is a constraint on negative λ (set negative λ to zero), the distribution is asymetrical. This means it will be symetrical when both of states are zero or both of them are not zero. when one of the state is zero, the other one must be within 10 and then a situation that states will be absorbed by zero will happen. From -10 to 10, there are 21 points, so the denominator is 21. Then, the proposal markov chain for λ is given by

$$Q(\lambda_{new}|\lambda_{old}) = \begin{cases} \frac{11-\lambda_{old}}{21}, & \lambda_{new} = 0, \lambda_{old} = 1, ..., 9 \\ \frac{1}{21}, & \lambda_{old} = 0, \lambda_{new} = 1, ..., 9 \end{cases}$$

To get acceptance probability $\alpha = min(r, 1)$, we need $r = \frac{p(M',\lambda'|x)}{p(M_n,\lambda_n|x} \frac{Q(y_n|y')}{Q(y'|y_n)}$.
The first fraction $\frac{p(M',\lambda'|x)}{p(M_n,\lambda_n|x)} = \frac{f(x|M',\lambda')\pi_M(M')\pi_\lambda(\lambda')}{f(x|M_n,\lambda_n)\pi_M(M_n)\pi_\lambda(\lambda_n)}$.
$\frac{f(x|M',\lambda')}{f(x|M_n,\lambda_n)}$ in this part can be derived by the expression in (a). $\pi_M(M)$ is given. As λ is uniform, $\pi_\lambda(\lambda)$ is the same and can be removed.
The second fraction $\frac{Q(y_n|y')}{Q(y'|y_n)} = \frac{Q(M_n|M')}{Q(M'|M_n)} \frac{Q(\lambda_n|\lambda')}{Q(\lambda'|\lambda_n)}$.

$$\frac{Q(M_n|M')}{Q(M'|M_n)} = \begin{cases} 1/2, & M' = 9, M_n = 8 \text{ or } M' = 9, M_n = 10 \\ 2, & M' = 8, M_n = 9 \text{ or } M' = 10, M_n = 9 \end{cases}$$

$$\frac{Q(\lambda_n|\lambda')}{Q(\lambda'|\lambda_n)} = \begin{cases} 1, & \text{if } \lambda_n = \lambda' = 0 \text{ or } \lambda_n \neq 0 \& \lambda' \neq 0 \\ \frac{1}{11-\lambda_n}, & \text{if } \lambda_n \neq 0, \lambda' = 0 \\ 11 - \lambda_n, & \text{if } \lambda_n = 0, \lambda' \neq 0 \end{cases}$$

```
sampling.function = function(x){
  ntrace = 5000000
  m0 = 8
  lmd0 = 240*x
  m.traace = rep(0, 5000)
  lmd.trace = rep(0, 5000)
```

```r
for (i in 1:ntrace){
  if (m0 == 8){
    m1 = 9
  }else if(m0 == 9){
    m1 = as.numeric(sample(list(8, 10),1))
  }else if(m0 ==10){
    m1 = 9
  }

  temp = lmd0 + round(runif(1, -10, 10))
  if (temp < 0){
    lmd1 = 0
  }else{
    lmd1 = temp
  }

  if (m1 == 9) {
    qm.ratio = 0.5
  }else if(m1 == 8 || m1 == 10){
    qm.ratio = 2
  }

  if (lmd0 == 0 && lmd1==0){
    ql.ratio = 1
  }else if(lmd0 != 0 && lmd1 != 0){
    ql.ratio = 1
  }else if(lmd0!= 0 && lmd1 == 0){
    ql.ratio = 1/(11-lmd0)
  }else if(lmd0 == 0 && lmd1 != 0){
    ql.ratio = 11-lmd0
  }

  if (m0 == 8 && m1 == 9){
    pi.ratio = 1/2
  }else if(m0 == 9 && m1 == 10){
    pi.ratio = 1/2
  }else if(m0 == 10 && m1 == 9){
    pi.ratio = 2
  }else if(m0 == 9 && m1 == 8){
    pi.ratio = 2
  }

  l1 = -lmd1/240
  l0 = -lmd0/240

  p1.temp = 1-exp(l1/m1)
  p2.temp = 1-exp(l0/m0)
  f.ratio = dbinom(x, size=10, prob=p1.temp)/dbinom(x, size=10, prob=p2.temp)

  r=f.ratio*pi.ratio*qm.ratio*ql.ratio
  alpha = min(r, 1)

  if (runif(1) <= alpha){
```

```
      m0 = m1
      lmd0 = lmd1
    } else {
      m0 = m0
      lmd0 = lmd0
    }

    if (i%%1000==0){
      m.traace[i/1000]=m0
      lmd.trace[i/1000]=lmd0
    }
  }
  return(cbind(m.traace, lmd.trace))
}
```

(d)

Compare to the prior distribution, the posterior distribution has slight increase when $M = 9$ and 10, and a decrease when $M = 8$.

```
s1 = sampling.function(2)
a1 = length(s1[,1][s1[,1] == 8])
a2 = length(s1[,1][s1[,1] == 9])
a3 = length(s1[,1][s1[,1] == 10])
a1/length(s1[,1])
```

```
## [1] 0.5254
```

```
a2/length(s1[,1])
```

```
## [1] 0.3136
```

```
a3/length(s1[,1])
```

```
## [1] 0.161
```

```
s2 = sampling.function(4)
a1 = length(s2[,1][s2[,1] == 8])
a2 = length(s2[,1][s2[,1] == 9])
a3 = length(s2[,1][s2[,1] == 10])
a1/length(s2[,1])
```

```
## [1] 0.5294
```

```
a2/length(s2[,1])
```

```
## [1] 0.3042
```

```r
a3/length(s2[,1])
```

```
## [1] 0.1664
```

```r
s3 = sampling.function(6)
a1 = length(s3[,1][s3[,1] == 8])
a2 = length(s3[,1][s3[,1] == 9])
a3 = length(s3[,1][s3[,1] == 10])
a1/length(s3[,1])
```

```
## [1] 0.5264
```

```r
a2/length(s3[,1])
```

```
## [1] 0.2998
```

```r
a3/length(s3[,1])
```

```
## [1] 0.1738
```

```r
s4 = sampling.function(8)
a1 = length(s4[,1][s4[,1] == 8])
a2 = length(s4[,1][s4[,1] == 9])
a3 = length(s4[,1][s4[,1] == 10])
a1/length(s4[,1])
```

```
## [1] 0.511
```

```r
a2/length(s4[,1])
```

```
## [1] 0.3072
```

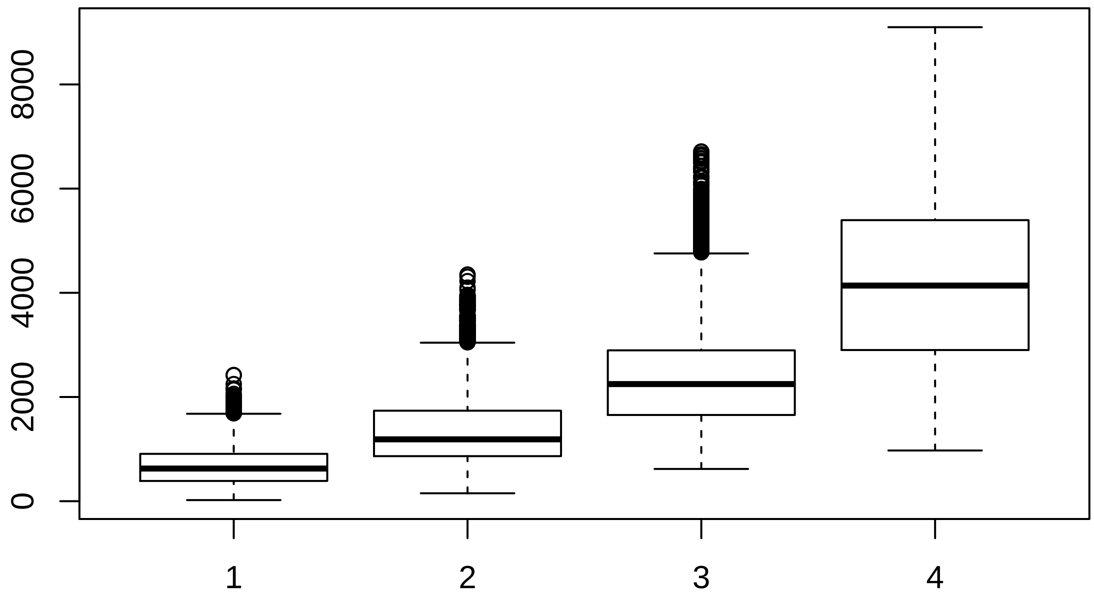
```r
a3/length(s4[,1])
```

```
## [1] 0.1818
```

(e)

The mean of posterior posibilities increase as x increases.

```r
boxplot(cbind(s1[,2], s2[,2], s3[,2], s4[,2]))
```



Q3

(a)

```
library(MASS)
carsb = Cars93[,c(4,5,6,7,8,12,13,14,15,17,19:22,25,26)]
names(carsb)
```

```
##  [1] "Min.Price"          "Price"              "Max.Price"
##  [4] "MPG.city"           "MPG.highway"        "EngineSize"
##  [7] "Horsepower"         "RPM"                "Rev.per.mile"
## [10] "Fuel.tank.capacity" "Length"             "Wheelbase"
## [13] "Width"              "Turn.circle"        "Weight"
## [16] "Origin"
```

```
carsb[,-16] = log(carsb[,-16])

fa = function(confusion.table){
  n11 = confusion.table[1,1]
  n12 = confusion.table[1,2]
  n21 = confusion.table[2,1]
  n22 = confusion.table[2,2]

  LR_p =(n11/(n11+n21))/(1-n22/(n12+n22))
  LR_m = (1-n11/(n11+n21))/(n22/(n12+n22))

  CE = (n12+n21)/(n11+n12+n21+n22)

  return(c(CE, LR_p, LR_m))
}
```

(b)

```
lda.fit = lda(Origin~., data = carsb)
pr = predict(lda.fit)
confusion.table = table(pr$class, carsb$Origin)
confusion.table
```

```
##
##           USA non-USA
##   USA      43       5
##   non-USA   5      40
```

```
fa(confusion.table)
```

```
## [1] 0.1075269 8.0625000 0.1171875
```

(c)

yes. QDA classifier appear to improve the LDA classifier. The CE devreases.

```
qda.fit = qda(Origin~., data = carsb)
pr = predict(qda.fit)
confusion.table = table(pr$class, carsb$Origin)
confusion.table
```

```
##
##           USA non-USA
##   USA      46       1
##   non-USA   2      44
```

```
fa(confusion.table)
```

```
## [1]  0.03225806 43.12500000  0.04261364
```

(d)

Without cross validation, QDA is better than LDA. With cross validation, LDA is better. Because cross validation will eliminate the effect of overfitting.

```
ldaCV = lda(Origin~., data = carsb, CV=TRUE)
pr = ldaCV$class
confusion.table = table(pr, carsb$Origin)
confusion.table
```

```
##
## pr        USA non-USA
##   USA      41       6
##   non-USA   7      39
```

```r
fa(confusion.table)
```

```
## [1] 0.1397849 6.4062500 0.1682692
```

```r
qdaCV = qda(Origin~., data = carsb, CV=TRUE)
pr = qdaCV$class
confusion.table = table(pr, carsb$Origin)
confusion.table
```

```
##
## pr          USA non-USA
##   USA        36      11
##   non-USA   12      34
```

```r
fa(confusion.table)
```

```
## [1] 0.2473118 3.0681818 0.3308824
```

Q4

(a)

The classification error whould be 0.34.

```r
pima1 = rbind(Pima.tr)[,c(2,3,4,5,6,8)]
names(pima1)
```

```
## [1] "glu"  "bp"   "skin" "bmi"  "ped"  "type"
```

```r
#determine frequency
freq_yes = sum(pima1$type == "Yes")/length(pima1$type)
freq_no = sum(pima1$type == "No")/length(pima1$type)
# the classification error whould be 0.34
```

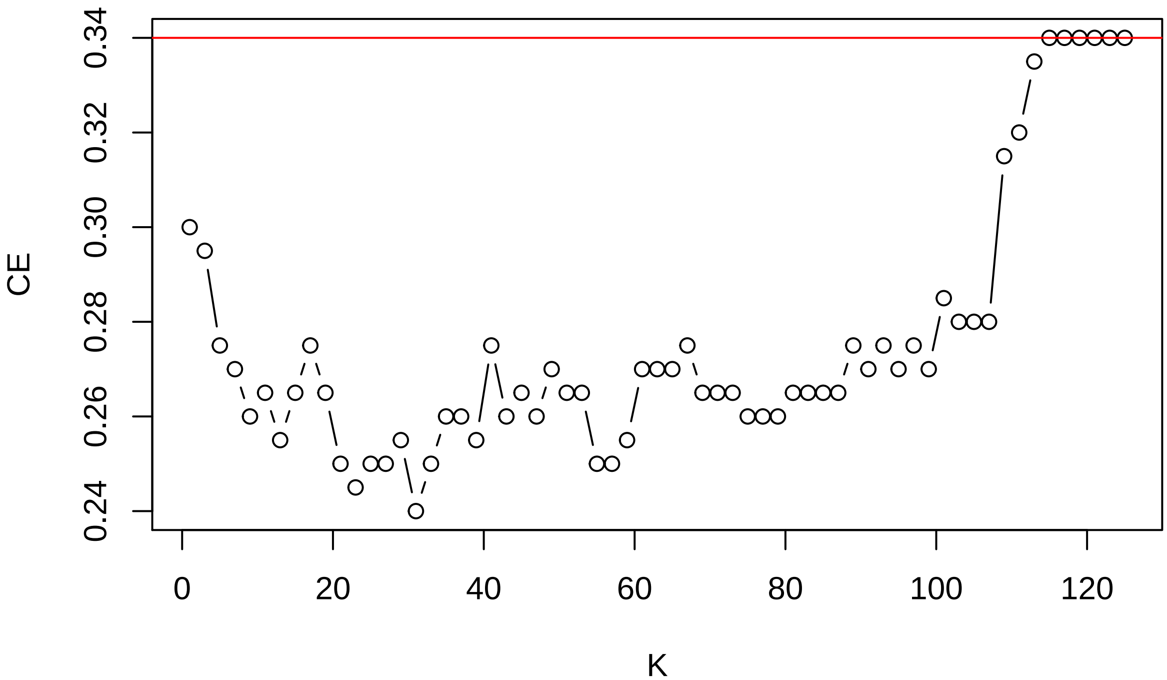
(b)

```r
library(class)
knn.function = function(k.list, train, gr) {
  pr.table = matrix(NA,length(k.list),3)
  colnames(pr.table) = list("CE", "LR+", "LR-")
  for (i in 1:length(k.list)) {
    knn.fit = knn.cv(train,gr,k=k.list[i],use.all=T)
    c_table = table(knn.fit,gr)
    pr.table[i,] = fa(c_table)
  }
  return(pr.table)
}
```

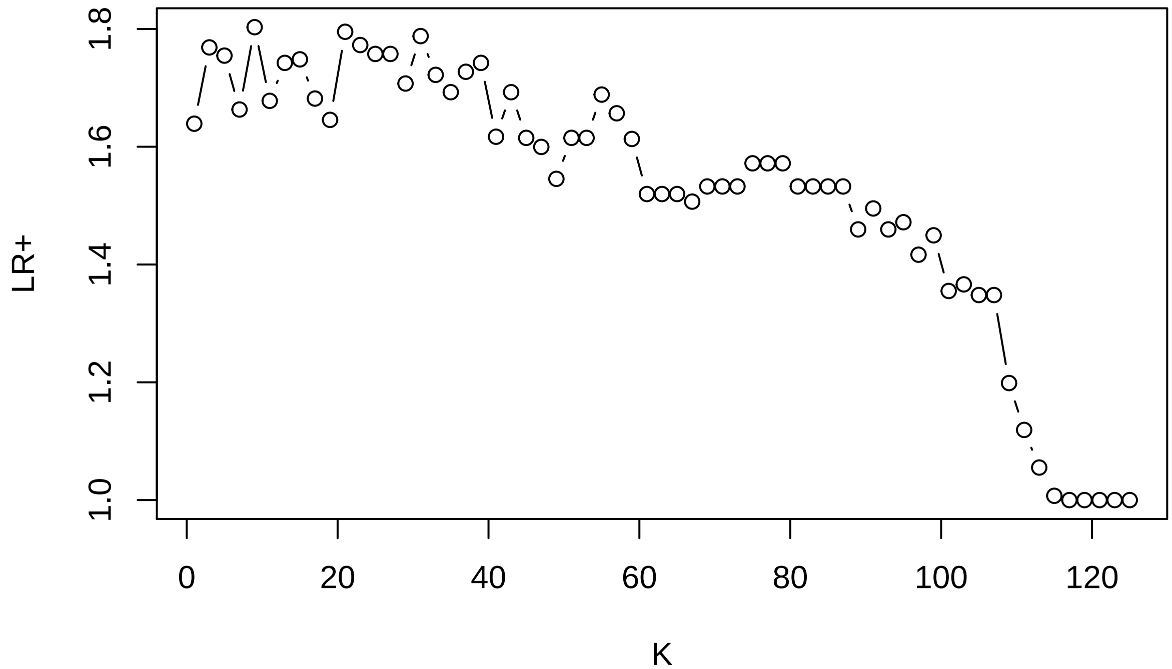
(c)

The results are as below.

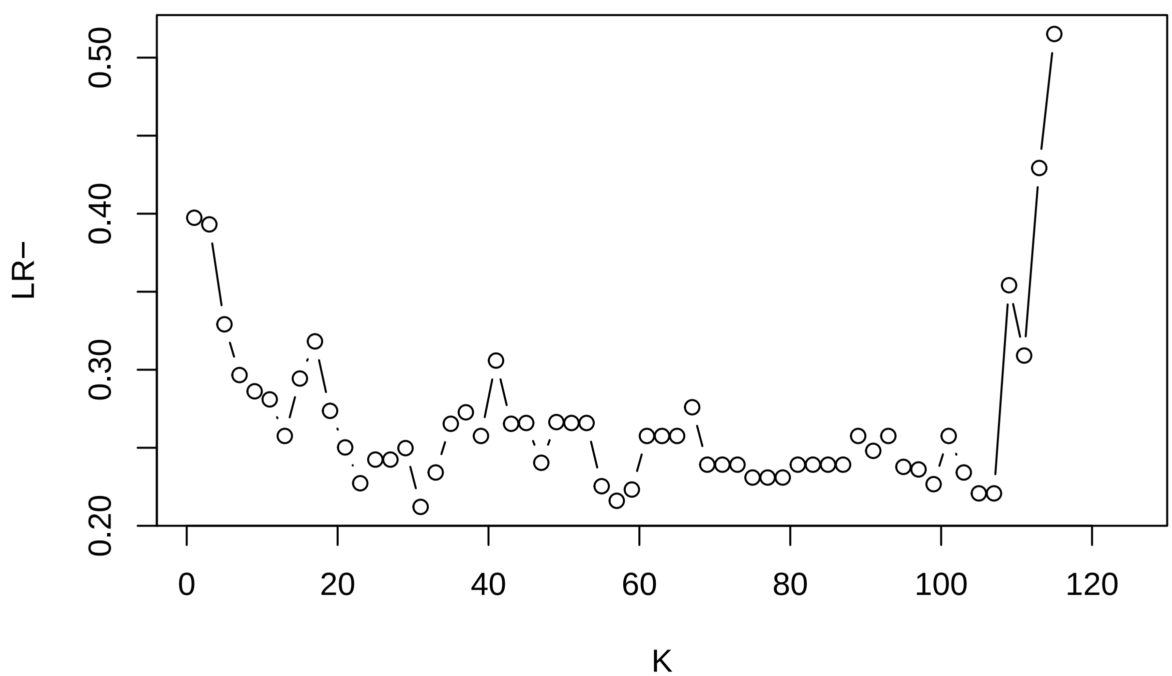
```
k.list = seq(1,125,2)
cv.table = knn.function(k.list, pima1[,1:5], pima1[,6])
# plot CE vs K
plot(k.list,cv.table[,1],type="b",xlab="K",ylab="CE")
abline(h=0.34, col="red")
```




```
plot(k.list,cv.table[,2],type="b",xlab="K",ylab="LR+")
```




```
plot(k.list,cv.table[,3],type="b",xlab="K",ylab="LR-")
```




```
# min CE
print("min CE")
```

```
## [1] "min CE"
```

```r
min(cv.table[,1],na.rm=TRUE)
```

```
## [1] 0.24
```

```r
k.list[cv.table[,1]==min(cv.table[,1], na.rm=TRUE)]
```

```
## [1] 31
```

```r
# min and max for LR+
print("min and max LR+")
```

```
## [1] "min and max LR+"
```

```r
min(cv.table[,2], na.rm=TRUE)
```

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

```r
k.list[na.omit(cv.table[,2])==min(cv.table[,2], na.rm=TRUE)]
```

```
## [1] 117 119 121 123 125
```

```r
max(cv.table[,2], na.rm=TRUE)
```

```
## [1] 1.80303
```

```r
k.list[na.omit(cv.table[,2])==max(cv.table[,2], na.rm=TRUE)]
```

```
## [1] 9
```

```r
# min and max for LR-
print("min and max LR-")
```

```
## [1] "min and max LR-"
```

```r
min(cv.table[,3], na.rm=TRUE)
```

```
## [1] 0.2121212
```

```r
k.list[na.omit(cv.table[,3])==min(cv.table[,3], na.rm=TRUE)]
```

```
## [1] 31
```

```r
max(cv.table[,3], na.rm=TRUE)
```

```
## [1] 0.5151515
```

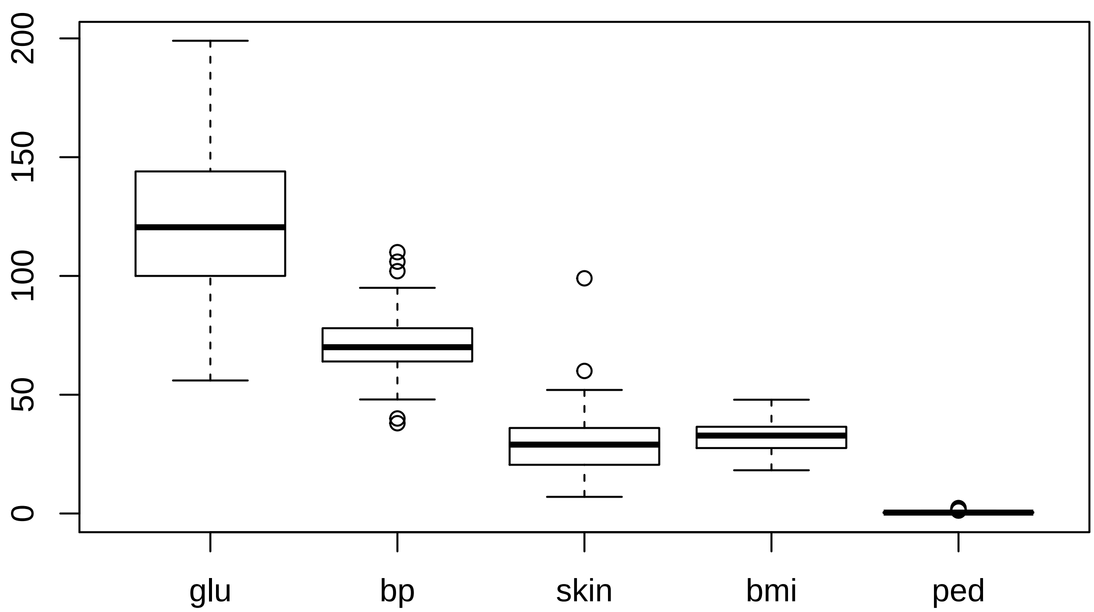
```r
k.list[na.omit(cv.table[,3])==max(cv.table[,3], na.rm=TRUE)]
```

```
## [1] 115
```

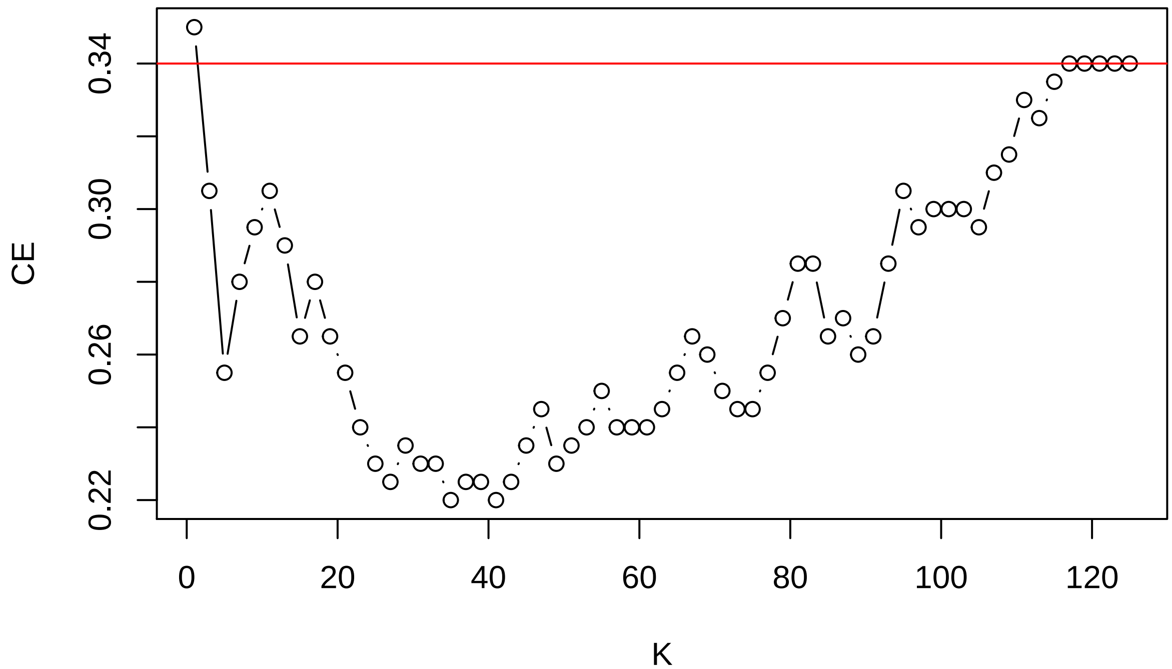
(d)

The CE is slightly lower than before. Scaling is necessary here because the outcome will be affected by the magnitude of variables. To be specific, it will be biased toward variables with higher magnitude.

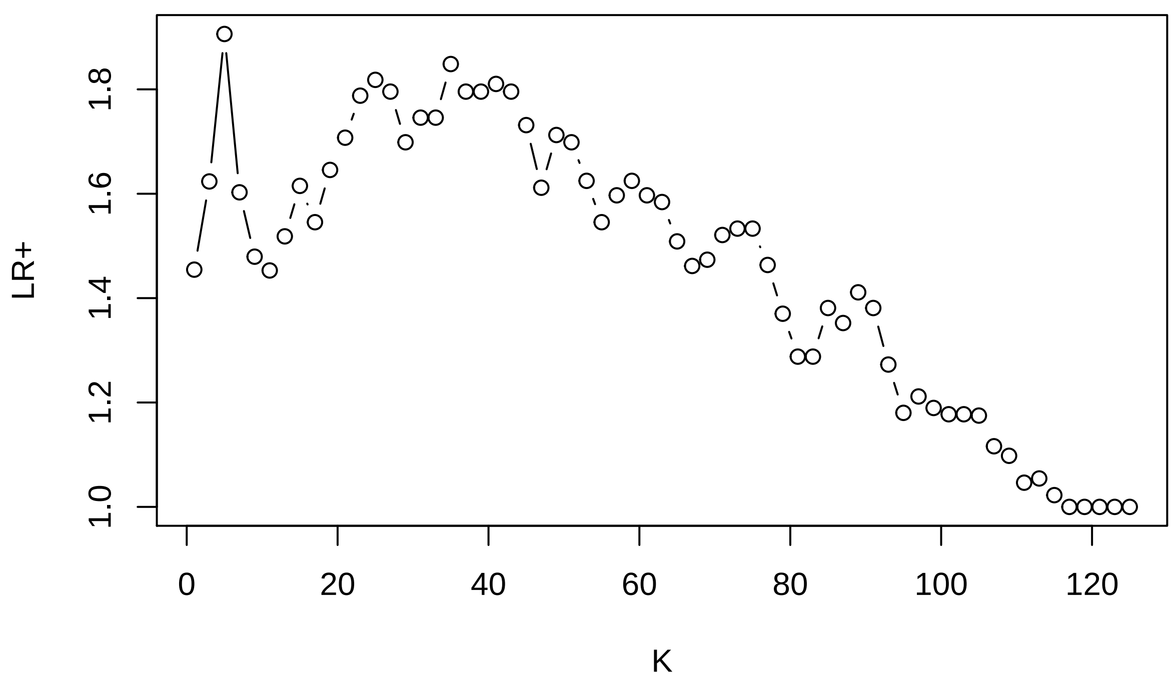
```r
# side by side boxplot
boxplot(pima1[,1:5])
```



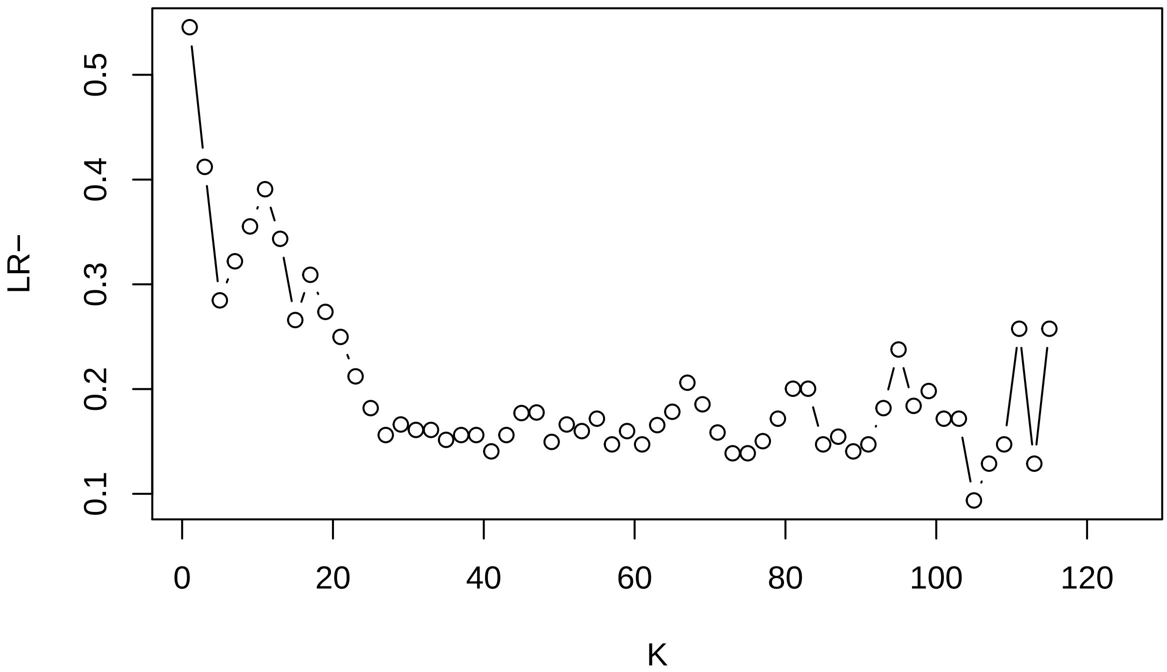

```r
# normalize
normalization.function = function(m){
  m.new = m
  for (i in 1:(length(m)-1)){
    m.new[,i] = (m[,i]-mean(m[,i], na.rm = TRUE))/sd(m[,i], na.rm = TRUE)
  }
  return (m.new)
}
pima.norm = normalization.function(pima1)

# repeat part c
k.list = seq(1,125,2)
new.table = knn.function(k.list, pima.norm[,1:5], pima.norm[,6])
# plot CE vs K
plot(k.list,new.table[,1],type="b",xlab="K",ylab="CE")
abline(h=0.34, col="red")
```




```
plot(k.list,new.table[,2],type="b",xlab="K",ylab="LR+")
```




```
plot(k.list,new.table[,3],type="b",xlab="K",ylab="LR-")
```




```r
# min CE
print("min CE")
```

```
## [1] "min CE"
```

```r
min(new.table[,1],na.rm=TRUE)
```

```
## [1] 0.22
```

```r
k.list[new.table[,1]==min(new.table[,1], na.rm=TRUE)]
```

```
## [1] 35 41
```

```r
# min and max for LR+
print("min and max LR+")
```

```
## [1] "min and max LR+"
```

```r
min(new.table[,2], na.rm=TRUE)
```

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

```r
k.list[na.omit(new.table[,2])==min(new.table[,2], na.rm=TRUE)]
```

```
## [1] 117 119 121 123 125
```

```r
max(new.table[,2], na.rm=TRUE)
```

```
## [1] 1.906061
```

```r
k.list[na.omit(new.table[,2])==max(new.table[,2], na.rm=TRUE)]
```

```
## [1] 5
```

```r
# min and max for LR-
print("min and max LR-")
```

```
## [1] "min and max LR-"
```

```r
min(new.table[,3], na.rm=TRUE)
```

```
## [1] 0.09366391
```

```r
k.list[na.omit(new.table[,3])==min(new.table[,3], na.rm=TRUE)]
```

```
## [1] 105
```

```r
max(new.table[,3], na.rm=TRUE)
```

```
## [1] 0.5454545
```

```r
k.list[na.omit(new.table[,3])==max(new.table[,3], na.rm=TRUE)]
```

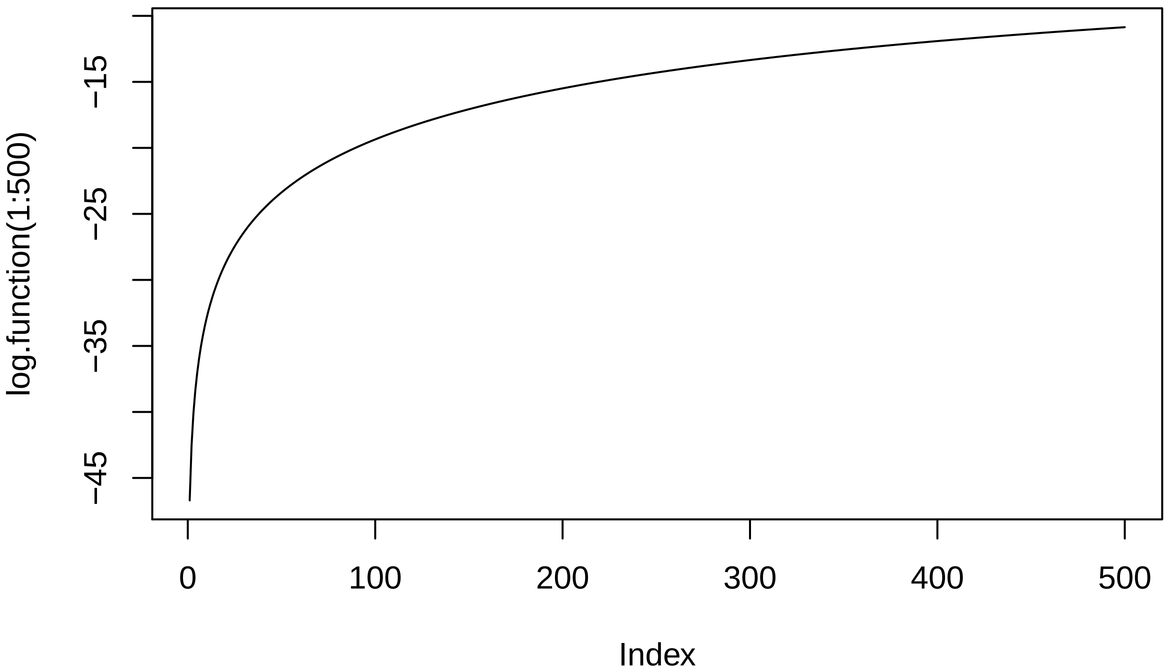
```
## [1]   1 117
```

Q5

(a)

The log-likelihood function is
$L(\lambda; x) = xlog(1 - e^{-\frac{\lambda}{2400}}) + (10 - x)log(e^{-\frac{\lambda}{2400}}) + C$

```r
log.function = function(lmd){
  6*log(1-exp(-lmd/2400))+(10-6)*log(exp(-lmd/2400))
}
plot(log.function(1:500), type = "l")
```



(b)

The sufficient conditions for optimization is
(1) if $f'(\lambda) = 0$, then λ is a stationary point of f.
(2) if $f'(\lambda) = 0$ and $f''(\lambda) < 0$, then λ is a local maximum of f.

Take the first derivative, we get
$\frac{dL(\lambda;x)}{d\lambda} = \frac{x}{1-e^{-\frac{\lambda}{2400}}} + \frac{10-x}{1-e^{-\frac{\lambda}{2400}}} = 0$
Therefore, $\hat{\lambda} = 2400log\frac{10}{10-x}$.

```r
lmd.MLE = function(x){
  2400*log(10/(10-x))
}

lmd.table = matrix(lmd.MLE(1:9), ncol=9, byrow=TRUE)
rownames(lmd.table) = "MLE"
colnames(lmd.table) = c(1:9)
lmd.table = as.table(lmd.table)
lmd.table
```

```
##            1         2         3         4         5         6         7
## MLE  252.8652  535.5445  856.0199 1225.9815 1663.5532 2199.0978 2889.5347
##            8         9
## MLE 3862.6510 5526.2042
```

(c)

When $x = 10$, $\hat{\lambda} = 2400log\frac{10}{10-x}$ will not exist. In this case, the log-likelihood funtion will be $L(\lambda; x) = 10log(1-e^{-\frac{\lambda}{2400}})+C$. Since log is an increasing function, as λ increases, the function will increases. Therefore, λ should take the maximum possible value.