# Institute of Actuaries of India

## Subject CS2B – Risk Modelling and Survival Analysis (Paper B)

# March 2022 Examination

# INDICATIVE SOLUTION

**Introduction**

The indicative solution has been written by the Examiners with the aim of helping candidates. The solutions given are only indicative. It is realized that there could be other points as valid answers and examiner have given credit for any alternative approach or interpretation which they consider to be reasonable.

**Solution 1:**

```
library(moments)

# i)

f=function(x){
    ifelse(x[1]>150 & x[2]>50,ifelse(x[3]>300,1,0),0)
}
```

```
#OR

f=function(x){
 ifelse(x[1]>150 & x[2]>50,0,ifelse(x[3]>300,1,0))
}

#OR

f=function(x){
 ifelse(x[1]>150 & x[2]>50,ifelse(x[3]>300,0,1),0)
}

#OR

f=function(x){
 ifelse(x[1]>150 & x[2]>50,0,ifelse(x[3]>300,0,1))
}
```

(3)

```
# (ii)

f(c(180,75,350))

## [1] 1

## [1] 0

## [1] 0

## [1] 0
```

(1)

```
# (iii)

decisionData=read.csv("D:\\IAI Question Paper\\Mar22 Diet\\CS2B_Mar22_Dataset
1.csv")
model=apply(decisionData,1,f)
model

##  [1] 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0
0 0 1
## [39] 0 0 1 0 0 0 0 0 0 1 0 0
```

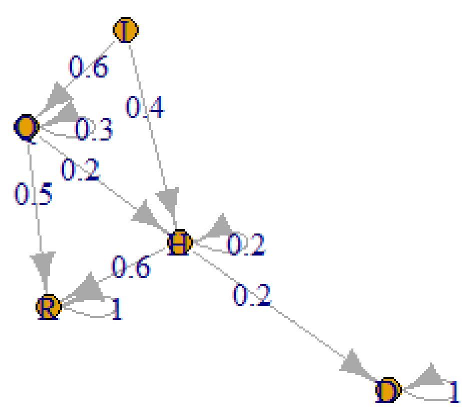
```
##[1] 0 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 1 1
1 0 0 0 0 1 1 0 1 0 1 0 0 0

##[1] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0 0 1 1

## [1] 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
                                                                           (3)


# (iv)

TP=sum(model == 1 & decisionData$OperationStatus == 1)
print(paste("TP = ", TP))

## [1] "TP =  8"

FP=sum(model == 1 & decisionData$OperationStatus == 0)
print(paste("FP = ", FP))

## [1] "FP =  5"

TN=sum(model == 0 & decisionData$OperationStatus == 0)
print(paste("TN = ", TN))

## [1] "TN =  22"

FN=sum(model == 0 & decisionData$OperationStatus == 1)
print(paste("FN = ", FN))

## [1] "FN =  15"

precision=(TP/(TP+FP))
precision

## [1] 0.6153846

recall=(TP/(TP+FN))
recall

## [1] 0.3478261

f1=(2*precision*recall)/(precision+recall)
f1

## [1] 0.4444444
                                                                           (6)


# (v)

# Lower precision value indicates the presence of larger number of False posi
tives
# Lower recall value indicates the presence of larger number of False negativ
es
# This test is not really effective at correctly identifying individuals who
require Surgery
```

```
# F1 tells how precise your classifier is (how many instances it classifies c
orrectly), as well as how robust it is (it does not miss a significant number
of instances)
# Low F1 score indicates that the model can be modified to increase its perfo
rmance
                                                                          (2)
                                                                  [15 Marks]
```

**Solution 2**:

```
# i)

covid_data=read.csv("D:\\IAI Question Paper\\Mar22 Diet\\CS2B_Mar22_Dataset2.
csv")
covid_data$StateFrom = as.factor(covid_data$StateFrom)
covid_data$State.To = as.factor(covid_data$State.To)
States_From = levels(covid_data$StateFrom)
States_To = levels(covid_data$State.To)
States = unique(c(States_From,States_To))

# Transition Probabilities for each pair

transitions_Master = c()
for (i in States) {
    for (j in States) {

        transition = sum(covid_data$StateFrom == i & covid_data$State.To == j
)/sum(covid_data$StateFrom == i)
        transition = ifelse(sum(covid_data$StateFrom == i) ==0, ifelse(i ==j,
1,0),transition)
        transitions_Master = c(transitions_Master,transition)

    }

}

transition_matrix = matrix(transitions_Master, nrow=5, byrow = T, dimnames=li
st(States, States))
transition_matrix

##     H I   Q   D   R
## H 0.2 0 0.0 0.2 0.6
## I 0.4 0 0.6 0.0 0.0
## Q 0.2 0 0.3 0.0 0.5
## D 0.0 0 0.0 1.0 0.0
## R 0.0 0 0.0 0.0 1.0
                                                                          (6)

#(ii)

# The two absorbing states are "Recovered" and "Death"                    (2)

# (iii)
```

```
library(markovchain)

## Warning: package 'markovchain' was built under R version 4.0.5

## Package:   markovchain
## Version:   0.8.5-4
## Date:      2021-01-07
## BugReport: https://github.com/spedygiorgio/markovchain/issues

covid19=new("markovchain",states=c("H","I","Q", "D","R"),
                         transitionMatrix=transition_matrix)

covid19

## Unnamed Markov chain
##  A  5 - dimensional discrete Markov Chain defined by the following states:
##  H, I, Q, D, R
##  The transition matrix  (by rows)  is defined as follows:
##     H I   Q   D   R
## H 0.2 0 0.0 0.2 0.6
## I 0.4 0 0.6 0.0 0.0
## Q 0.2 0 0.3 0.0 0.5
## D 0.0 0 0.0 1.0 0.0
## R 0.0 0 0.0 0.0 1.0

plot(covid19)
```




(4)

```
# (iv)

# (a)
```

```
initialstate=c(0,1,0,0,0)
afterweek=initialstate*(covid19)
afterweek

##        H I   Q D R
## [1,] 0.4 0 0.6 0 0

## [1] 0.6
```

(1)

```
# (b)

aftertwoweeks=initialstate*(covid19*covid19)
aftertwoweeks

##        H I    Q    D    R
## [1,] 0.2 0 0.18 0.08 0.54

# Probability of either hospital or quarantine after 2 weeks
p1 = aftertwoweeks[1]+aftertwoweeks[3]
p1

## [1] 0.38
```

(2)

```
# (c)
afterthreeweeks=initialstate*(covid19*covid19*covid19)
afterthreeweeks

##          H I     Q    D    R
## [1,] 0.076 0 0.054 0.12 0.75

#Probability of recovery after 3 weeks
p2 = afterthreeweeks[5]
p2

## [1] 0.75
```

(2)

```
# (v)

# Long term probability (after 13 weeks)

# Quarantined People
initial_Q = c(0,0,1,0,0)
after_quarter = initial_Q*(covid19^13)
initial_H = c(1,0,0,0,0)
after_quarter1 = initial_H*(covid19^13)


Recovered = 20000*after_quarter[5]+10000*after_quarter1[5]
Died = 20000*after_quarter[4]+10000*after_quarter1[4]
Quarantined = 20000*after_quarter[3]+10000*after_quarter1[3]
Hospitalized = 20000*after_quarter[1]+10000*after_quarter1[1]
```

```
round(Recovered,0)

## [1] 26071

round(Died,0)

## [1] 3929

round(Hospitalized,0)

## [1] 0

round(Quarantined,0)

## [1] 0
```

(5)

**# (vi)**

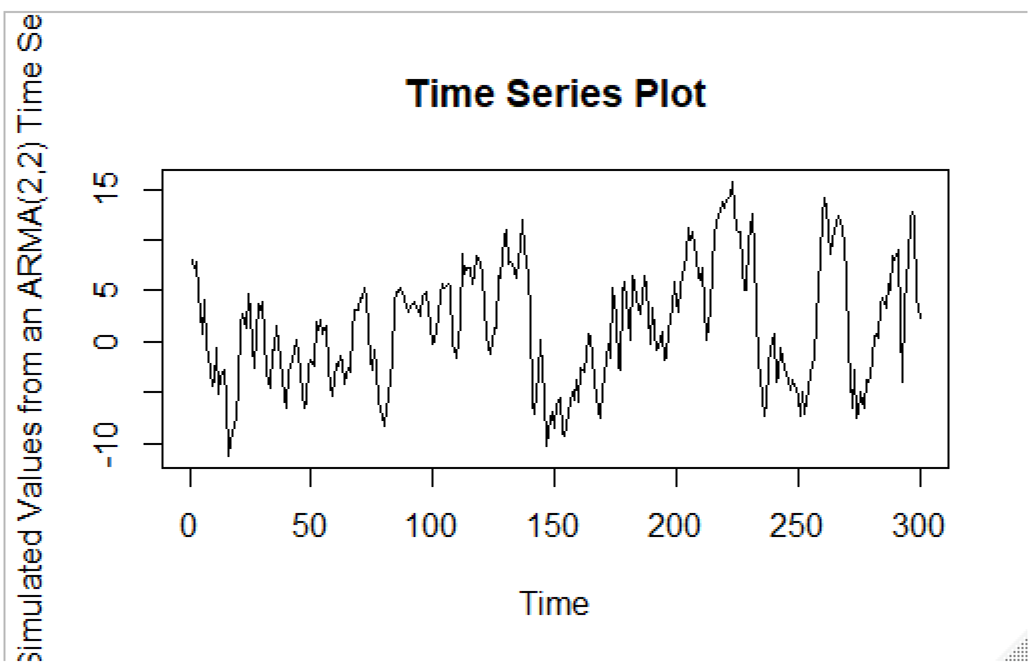
```
# In the long run, everyone needs to reach a steady state probabilities.
# As there are two absorbing states, one of the two is the destination
# The other states are completely zeroes
```

(3)
**[25 Marks]**

**Solution 3:**

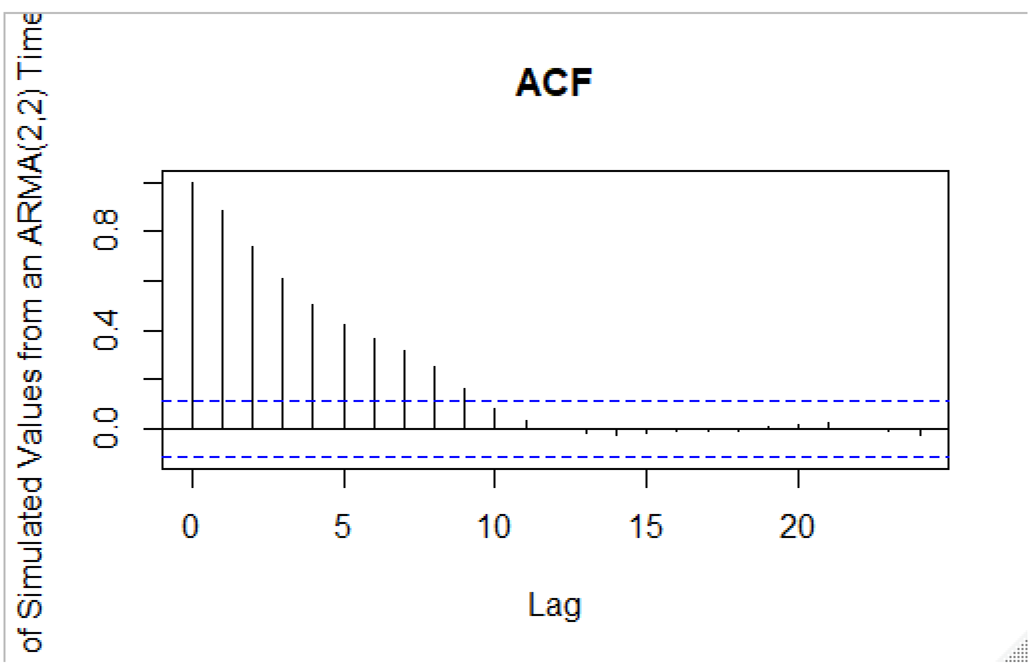
***(i)***

```
set.seed(100)
z=2+arima.sim(n=300,list(ar=c(0.8,0.1),ma=c(0.4,0.1),sd=7^0.5))
plot(z, xlab = "Time", ylab = "Simulated Values from an ARMA(2,2) Time Series
", main = "Time Series Plot")
```

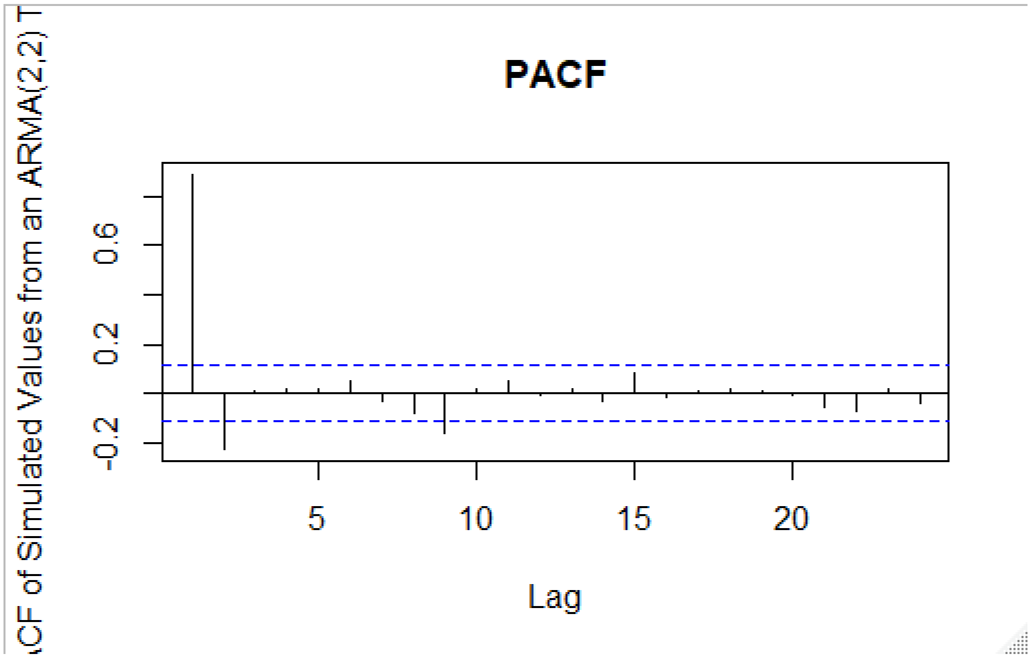



(4)

```
# (ii)

# (a)
mean (z)

## [1] 1.520674

sd(z)

## [1] 5.865453

# (b)
mean(z[1:150])

## [1] 0.678429

sd(z[1:150])

## [1] 4.997272

# (c)
mean(z[151:300])

## [1] 2.362919

sd(z[151:300])

## [1] 6.529698
                                                                         (6)

# (iii)

acf(z, xlab = "Lag", ylab = "ACF of Simulated Values from an ARMA(2,2) Time S
eries", main = "ACF")
```



```
pacf(z, xlab = "Lag", ylab = "Partial ACF of Simulated Values from an ARMA(2,
2) Time Series", main = "PACF")
```




(4)

```
# (iv)

# The series appears stationary as there are no obvious trends or cycles in t
he graph of the series and it appears to have constant mean.
# However, from (ii), it appears that mean of the two subsets of the data is
very different.
# Working with a larger subset takes the mean values closer to constant thus
revealing in stationarity.
# For a stationary time series, the ACF will drop to zero relatively quickly,
while the ACF of non-stationary data decreases slowly
# Hence, we confirm that the series is stationary
```

(3)

```
# (v)

# PACF becomes insignificant after two lags and ACF goes gradually to zero in
dicating the strong presence of AR(2)process compared to an ARMA (2,2) models
# In ARMA(2,2) model, we should observe, ACF and PACF to gradually go down to
zero after a few lags
```

(3)

**[20 Marks]**

**Solution 4:**

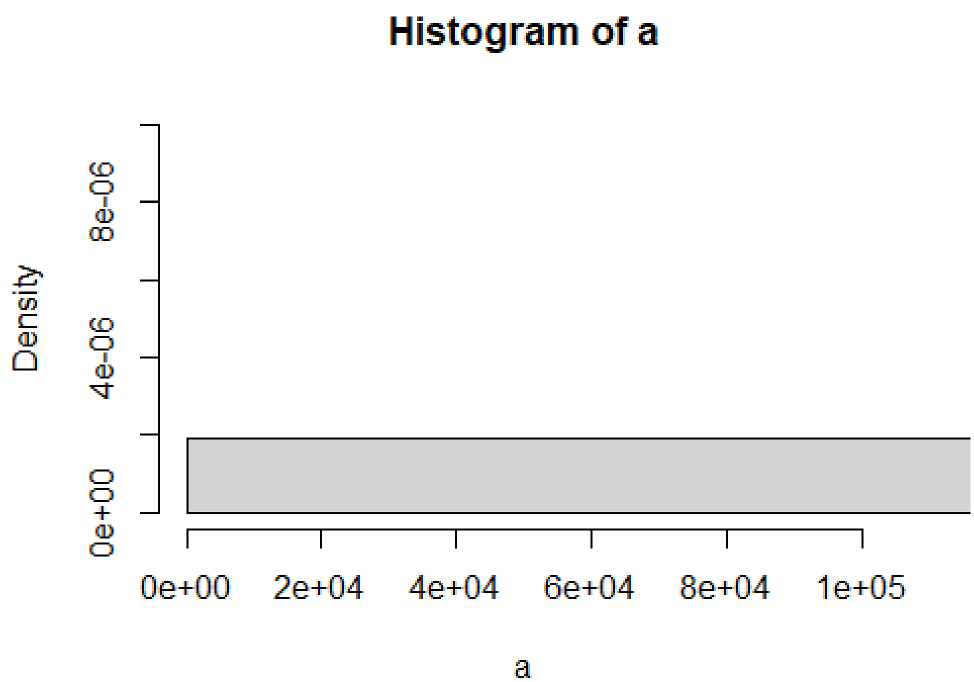
```
# i)
# a)

mu = 10
```

```
var = 4
dlnorm(5000,mu,sqrt(var))

## [1] 3.030741e-05

# The likelihood that the claim will be of 5000 is 0.000030
                                                                            (2)


# b)

1-plnorm(5000,mu,sqrt(var))

## [1] 0.7707756

# probability that the claim payout will be greater than 5000 is 0.77
                                                                            (2)


# c)
qlnorm(0.9,mu,sqrt(var))

## [1] 285815.9

qlnorm(0.99,mu,sqrt(var))

## [1] 2309856

# maximum claim payout in the confidence interval [0.9,0.99] is [285815.9, 23
09856]
                                                                             (3)

# d)
qlnorm(0.5,mu,sqrt(var))

## [1] 22026.47
#Median of Z is 22026.47
                                                                             (2)

# e)
qlnorm(0.75,mu,sqrt(var))-qlnorm(0.25,mu,sqrt(var))

## [1] 79162.81

# Interquartile range for Z is 79162.81
                                                                             (2)

# ii)
a)
set.seed(50)
a=rlnorm(100,mu,sqrt(var))
mean(a)

## [1] 121227.3

median(a)

## [1] 22785.05
```

```
sd(a)

## [1] 481295.3

skewness(a)

## [1] 8.163985

sk = function(x) sum((x-mean(x))^3)/(100*(sd(x))^3)
sk(a)

## [1] 8.041831
```

(4)

```
# (b)
hist(a, freq = FALSE,xlim = c(0,100000),ylim = c(0,0.00001))
```




```
#hist(a,freq = FALSE)
```

(5)

**[20 Marks]**

**Solution 5:**

**# (i)**

For the Nelson-Aalen model the estimated integrated hazard is calculated as:

$$\hat{\Lambda}(t) = \sum_{t_j \le t} \frac{d_j}{n_j}$$

The estimated variance of the estimator of the integrated hazard is calculated as:

$$\text{var}\left[\hat{\Lambda}(t)\right] = \sum_{t_j \le t} \frac{d_j(n_j - d_j)}{n_j^3}$$

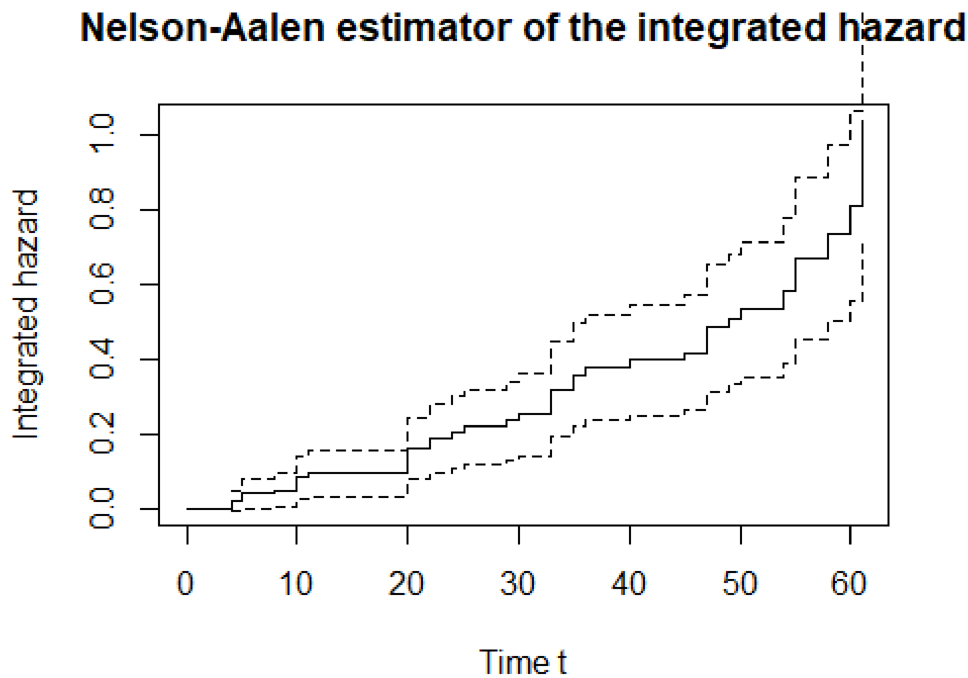
(2)

```
# (ii)

data = read.csv("D:\\IAI Question Paper\\Mar22 Diet\\CS2B_Mar22_Dataset3.csv"
)
names(data) = c("j","tj","nj","dj")
data$lambda = cumsum(data$dj/data$nj)
data$sdlambda=sqrt(cumsum(data$dj*(data$nj-data$dj)/data$nj^3))
data

data$varlambda=cumsum(data$dj*(data$nj-data$dj)/data$nj^3)
data

##     j tj  nj dj     lambda    varlambda
## 1   1  4 100  2 0.02000000 0.0001960000
## 2   2  5  98  2 0.04040816 0.0003999966
## 3   3  8  96  1 0.05082483 0.0005073733
## 4   4 10  90  3 0.08415816 0.0008653980
## 5   5 11  87  1 0.09565242 0.0009959972
## 6   6 15  86  0 0.09565242 0.0009959972
## 7   7 20  76  5 0.16144189 0.0018046975
## 8   8 22  71  2 0.18961090 0.0021902682
## 9   9 24  67  1 0.20453628 0.0024097101
## 10 10 25  66  1 0.21968779 0.0026358002
## 11 11 29  61  1 0.23608123 0.0029001395
## 12 12 30  60  1 0.25274790 0.0031732876
## 13 13 33  58  4 0.32171342 0.0042803441
## 14 14 35  54  2 0.35875046 0.0049408125
## 15 15 36  52  1 0.37798122 0.0053035230
## 16 16 40  50  1 0.39798122 0.0056955230
## 17 17 45  49  1 0.41838939 0.0061035163
## 18 18 47  45  3 0.48505605 0.0074862324
## 19 19 49  42  1 0.50886558 0.0080396283
## 20 20 50  40  1 0.53386558 0.0086490033
## 21 21 54  38  2 0.58649716 0.0099611480
## 22 22 55  35  3 0.67221144 0.0122002150
## 23 23 58  30  2 0.73887811 0.0142742891
## 24 24 60  27  2 0.81295218 0.0168145523
## 25 25 61  22  5 1.04022491 0.0247972720
```

(6)

```
# (iii)
plot(c(0,data$tj),c(0,data$lambda),type="s",main="Nelson-Aalen estimator of t
he integrated hazard",xlab="Time t",ylab="Integrated hazard")
```

```
lines(data$tj,data$lambda-1.96*data$sdlambda,type="s",lty=2)
lines(data$tj,data$lambda+1.96*data$sdlambda,type="s",lty=2)
```




(6)

```
# (iv)

data$Survival_KM = cumprod(1-data$dj/data$nj)
data$Survival_KM

##  [1] 0.9800000 0.9600000 0.9500000 0.9183333 0.9077778 0.9077778 0.8480556
##  [8] 0.8241667 0.8118657 0.7995647 0.7864571 0.7733494 0.7200150 0.6933478
## [15] 0.6800142 0.6664139 0.6528136 0.6092927 0.5947857 0.5799161 0.5493942
## [22] 0.5023033 0.4688164 0.4340892 0.3354326
```

(3)

```
#(v)

# The inequality states that:

# Survival function of Kaplan Meir SKM(t) < survival function of Nelson Aalen
SNA(t)

# To demonstrate the inequality:

data$Survival_NA = exp(-data$lambda)
sum(data$Survival_KM<data$Survival_NA)

## [1] 25
```

```
# Since all the values are true, the inequality is proved
```

(3)

**[20 Marks]**

*****************