# HW2-Fangzhou Song

*Fangzhou Song*

## Coding Algorithm

a. The initialization step. Note that the parameters for each cluster, $\lambda_1$ and $\lambda_2$ have to be positive.

```
em_init=function(x){
  lam1=quantile(x,0.7)
  lam2=quantile(x,0.3)
  return(c(lam1,lam2))
}
```

b. The E-step. If you know what you are doing, this step should be straightforward. You just need to use the dpois function.

```
em_e=function(x,lam1,lam2){
  p1=dpois(x,lambda = lam1)
  p2=dpois(x,lambda = lam2)
  return(p1/(p1+p2))
}
```

c. The M-step. Use your result from the question 2.

$$\lambda_k = \frac{\sum_{i=1}^{n} \pi_{i,k} x_i}{\sum_{i=1}^{n} \pi_{i,k}}$$

```
em_m=function(x,z){
  lam1=sum(x*z)/sum(z)
  lam2=sum(x*(1-z))/sum(1-z)
  return(c(lam1,lam2))
}
```

d. Put it together

```
em_poisson=function(x,iter.max=100,conv.check=1e-4){
  init_para=em_init(x)
  lam1=init_para[1]
  lam2=init_para[2]

  previous_para=init_para

  for(t in 1:iter.max){
    e_result=em_e(x,lam1,lam2)    #E-step
    m_result=em_m(x,e_result)     #m-Step
    lam1=m_result[1]
    lam2=m_result[2]

    #stop the algorithm if we achieved convergence
    if(max(abs(m_result-previous_para)) < conv.check) break;

    previous_para=m_result
  }
```

```r
  return(list(z=e_result,
              lam=m_result))
}
```

### Check it using simulation

5. Generate random Poisson samples with the following R code, and test how well your algorithm works in finding the clusters and the cluster parameters:

```r
set.seed(233)
lambda1 <- 10
lambda2 <- 1.2

n1 <- 50
n2 <- 50

x1 <- rpois(n1,lambda1)
x2 <- rpois(n2,lambda2)

x <- c(x1,x2)
```

```r
fit1=em_poisson(x)
fit1
```

```
## $z
##   [1] 0.999999846 0.999999846 0.999943888 0.979903805 0.999998902
##   [6] 0.999999997 1.000000000 0.979903805 0.999598958 0.999943888
##  [11] 0.999999846 0.999998902 0.999999846 0.997139756 0.997139756
##  [16] 0.999943888 0.999992151 0.999943888 0.997139756 0.999998902
##  [21] 0.999943888 0.999998902 0.997139756 1.000000000 0.999598958
##  [26] 0.999943888 0.872123239 0.999999846 0.999998902 0.999992151
##  [31] 0.999992151 0.999598958 0.488203081 0.999998902 0.979903805
##  [36] 0.999999997 0.999943888 0.999943888 0.999998902 0.979903805
##  [41] 0.999992151 0.999598958 0.999598958 0.999598958 0.999998902
##  [46] 0.999943888 0.488203081 0.999998902 0.979903805 0.872123239
##  [51] 0.002603279 0.000364931 0.018319195 0.018319195 0.000364931
##  [56] 0.000364931 0.018319195 0.018319195 0.002603279 0.018319195
##  [61] 0.018319195 0.018319195 0.000364931 0.002603279 0.018319195
##  [66] 0.117714176 0.018319195 0.488203081 0.000364931 0.002603279
##  [71] 0.018319195 0.018319195 0.117714176 0.002603279 0.002603279
##  [76] 0.000364931 0.002603279 0.000364931 0.000364931 0.488203081
##  [81] 0.002603279 0.000364931 0.002603279 0.002603279 0.002603279
##  [86] 0.002603279 0.002603279 0.002603279 0.488203081 0.000364931
##  [91] 0.000364931 0.000364931 0.117714176 0.117714176 0.000364931
##  [96] 0.018319195 0.018319195 0.000364931 0.000364931 0.002603279
##
## $lam
## [1] 9.202479 1.287089
```

It shows that $\hat{\lambda}_1$=9.202479, $\hat{\lambda}_2$=1.287089, which are approximate to 10 and 1.2

Result check

```r
fit1$z > 0.5
```

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

```
##  [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [34]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [45]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE
```

Use 0.5 as cut-off. If z > 0.5, we can regrad it as the member of cluster 1, cluster 2 otherwise. It shows that most of predictions are correct.

## Real life Application

6. Download the DJI_vol.csv dataset from Blackboard. In this dataset, you will find the total number of volatile days that the Dow Jones Index had for each month. Here, the number of volatile days is defined as the number of days in which the absolute value of the daily return is higher than 1%.

Read data

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

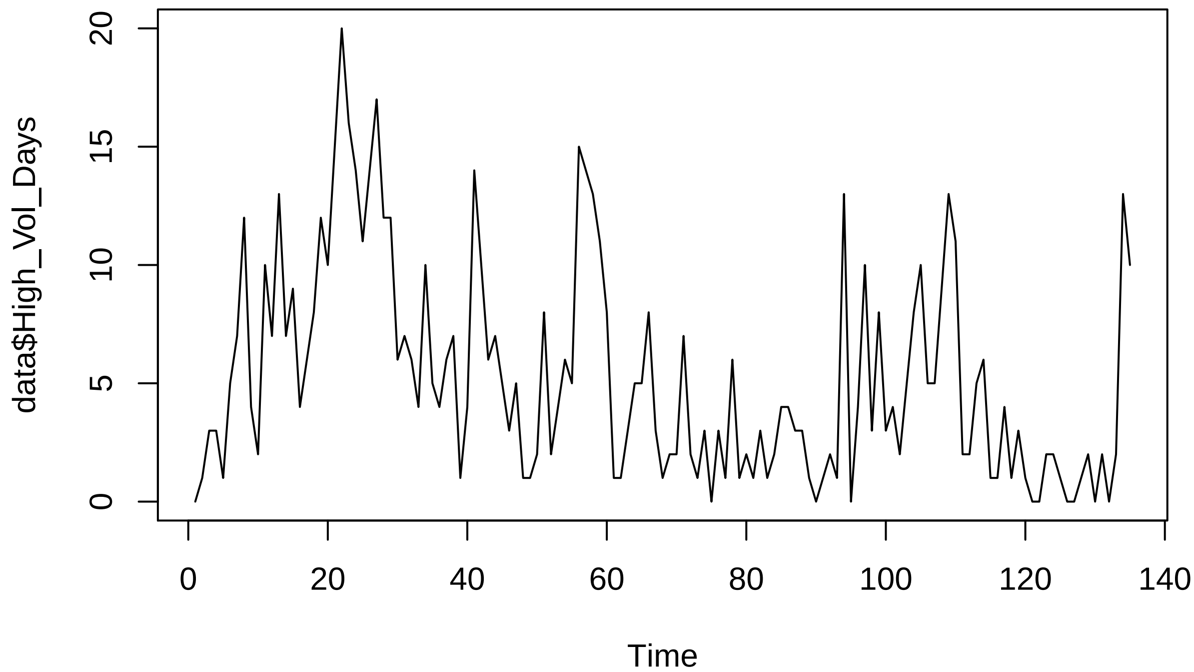
```
## The following object is masked from 'package:base':
##
##     date
```

```r
data=read_csv("DJI_vol.csv")
```

```
## Parsed with column specification:
## cols(
##   Date = col_character(),
##   High_Vol_Days = col_double()
## )
```

Time-series plot

```r
ts.plot(data$High_Vol_Days)
```




Cluster estimates

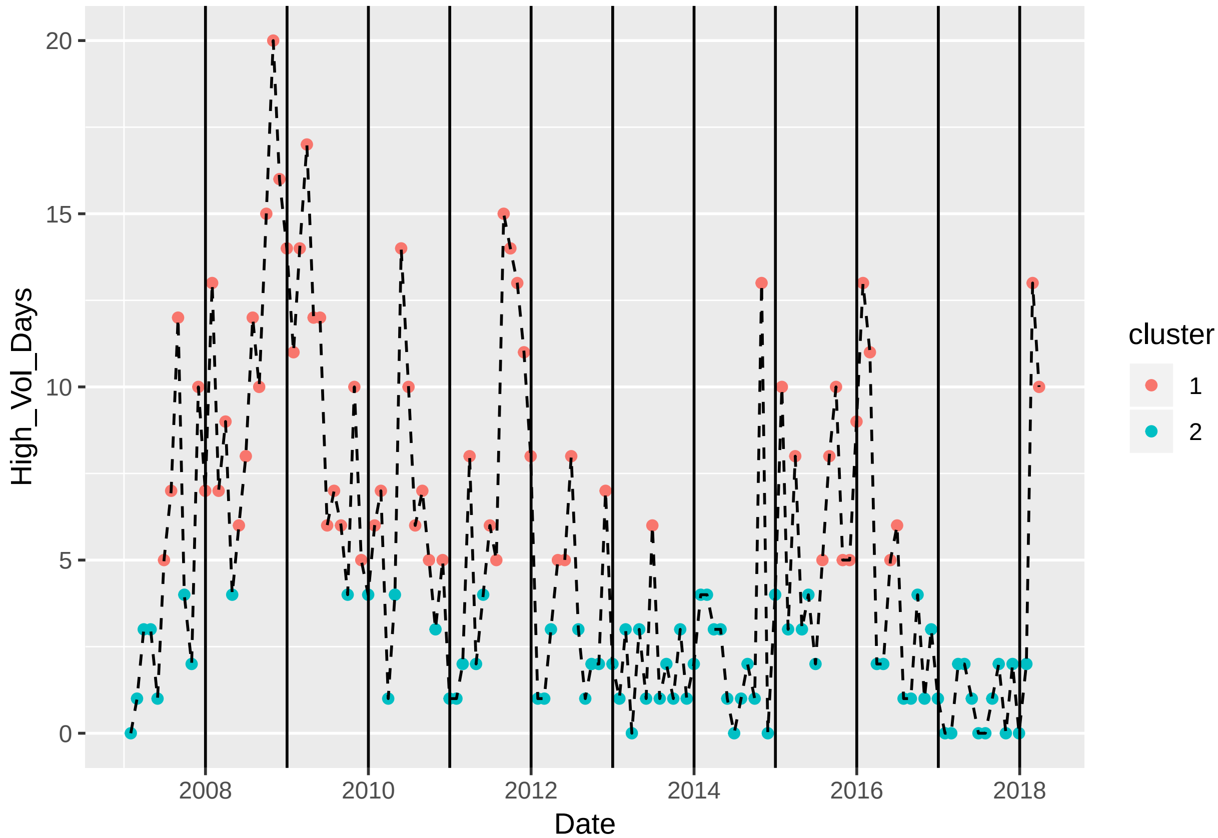
```
result=em_poisson(data$High_Vol_Days)
result$lam
```

```
## [1] 9.356897 2.126580
```

Plot the cluster assignments in time

```
data_2=cbind(data,result$z)
data_3=data_2 %>%
  mutate(
    cluster=as.factor(if_else(result$z > 0.5,1,2)),
    Date=ymd(Date)
  )
ggplot(data = data_3)+
  geom_point(mapping = aes(x=Date,y=High_Vol_Days,color=cluster))+
  geom_line(mapping = aes(x=Date,y=High_Vol_Days),linetype=2)+
  geom_vline(xintercept = ymd(c("2008-01-01",
                                "2009-01-01",
                                "2010-01-01",
                                "2011-01-01",
                                "2012-01-01",
                                "2013-01-01",
                                "2014-01-01",
                                "2015-01-01",
                                "2016-01-01",
                                "2017-01-01",
```

```
"2018-01-01")))
```




As can be seen from the plot, the boundary bewtween cluster 1 and 2 is 5. Before 2012, the cluster 1 is majority. However, the data after 2012 mostly belongs to cluster 2.

In my opinion, it is not suprised to see that there is no such a significant relationship between cluster and time because clustering is an unsupervised algorithm. During the process, we only try to analyze or explore some pattern in High_Vol_Days variable without any other information included.