First set the environment & load csv files，I load two data sets, one is sample from the slide and one is the iris flower data set.
Code: mydata<-read.csv("sample.csv")
Code: myflower<-read.csv("raw data.csv")

1. Implement chi-merge

I use the function chiSq ,from package “discretization” to calculate the chi-square value between two clusters. So we need to install and import this package.
Code: install.packages("discretization")
Code: library(discretization)

Below is my chiMerge algorithm , I have attached a text file in my homework report for readability:

```
chi_merge <- function(attr,freque,threshold){
  contingency_table <- table(attr,freque)
  contingency_table <- as.data.frame.matrix(contingency_table)

  print(contingency_table)
  #calculate chi value and create a table
  n = nrow(contingency_table)
  for (i in 1:(n-1) ) {
    chi_val = chiSq(contingency_table[i:(i+1),1:ncol(contingency_table)])
    if(grepl("-",rownames(contingency_table)[i])){
      lower_bound = as.numeric(strsplit(rownames(contingency_table)[i],"-")[[1]][1])
    }else{
      lower_bound = as.numeric(rownames(contingency_table)[i])
    }
    if(grepl("-",rownames(contingency_table)[(i+1)])){
      upper_bound = as.numeric(strsplit(rownames(contingency_table)[(i+1)],"-")[[1]][2])
    }else{
      upper_bound = as.numeric(rownames(contingency_table)[(i+1)])
    }
    if (!exists("chi_table")) chi_table <- c(lower_bound,upper_bound,chi_val)
    else chi_table <- rbind(chi_table,c(lower_bound,upper_bound,chi_val))
  }
  colnames(chi_table) <- c("lower_bound","upper_bound","chi_val")
  chi_table = as.data.frame(chi_table)
  min_chi = min(chi_table$chi_val)
  p_value = qchisq(1 - threshold,ncol(contingency_table)-1)
  print(chi_table)
  #merge the min chi_val entil the threshold is reached
  while(min_chi < p_value){
    n = nrow(chi_table)
    # generate new contingency_table
    for(i in 1:n){
      if(chi_table[i,3] == min_chi){
        if((i==n) || chi_table[(i+1),3] != min_chi){
          if(!exists("index_lower")) index_lower = i
          index_upper = (i+1)
          interval = paste(chi_table[index_lower,1],chi_table[i,2],sep="-")
          rownames(contingency_table)[index_lower] <- interval
          col_num = ncol(contingency_table)
          while(index_upper!=index_lower){
            for (j in 1:col_num){
              contingency_table[(index_upper-1),j] = contingency_table[(index_upper-1),j]+contingency_table[index_upper,j]
            }
            if(!exists("row_index_to_delete")){
              row_index_to_delete <- index_upper
            }
            else{
              row_index_to_delete <- c(row_index_to_delete,index_upper)
            }
            index_upper = index_upper - 1
          }
        }
        # we need to do cumulative merging if two adjacent chi_val are min value
        else{
          if(!exists("index_lower")) index_lower = i
          next
        }
      }
      if(exists("index_lower")) rm(index_lower)
    }
    contingency_table <- contingency_table[-row_index_to_delete,]
    rm(row_index_to_delete)
    print(contingency_table)
    # calculate chi value and generate chi_table
    n = nrow(contingency_table)
    rm(chi_table)
    for (i in 1:(n-1)){
      chi_val = chiSq(contingency_table[i:(i+1),1:ncol(contingency_table)])
      if(grepl("-",rownames(contingency_table)[i])){
        lower_bound = as.numeric(strsplit(rownames(contingency_table)[i],"-")[[1]][1])
      }else{
        lower_bound = as.numeric(rownames(contingency_table)[i])
      }
      if(grepl("-",rownames(contingency_table)[(i+1)])){
        upper_bound = as.numeric(strsplit(rownames(contingency_table)[(i+1)],"-")[[1]][2])
      }else{
```

```
77          upper_bound = as.numeric(rownames(contingency_table)[(i+1)])
78        }
79        if (!exists("chi_table")) chi_table <- c(lower_bound,upper_bound,chi_val)
80        else chi_table <- rbind(chi_table,c(lower_bound,upper_bound,chi_val))
81      }
82      colnames(chi_table) <- c("lower_bound","upper_bound","chi_val")
83      chi_table = as.data.frame(chi_table)
84      print(chi_table)
85      min_chi = min(chi_table$chi_val)
86    }
87    return(contingency_table)
88  }
89
90
```

The chi_merge function takes two vectors as input , one is **attr** , the data we want for discretization, one is **freque** , the class frequency of the corresponding data . The function also need a **threshold** parameter , which is needed to calculate the corresponding p values.

The output of this function is a contingency table(I print the contingency table and chi_val table every calculation in my function for debugging and better illustration) , the row names specify the cluster interval (inclusively) , the columns are the class frequency of this specific data interval. For example:

```
        Iris-setosa Iris-versicolor Iris-virginica
4.3-4.8          16               0              0
4.9               4               1              1
5-5.4            25               5              0
5.5-5.7           4              15              2
5.8-6.2           1              15             10
6.3-7             0              14             25
7.1-7.9           0               0             12
>
```

We take the first line for illustration , it says , in 4.3-4.8 interval (inclusively) , there are 16 instances of class Iris-setosa , 0 instance of Iris-versicolor and 0 instance of class Iris-virginica.

Basically the function chi_merge do two things , first , generate the **contingency table** of **attr** and **freque** . Then generate the corresponding **chi_value** table . which is needed to update our original contingency table. I will use the example in the slides for demonstration:

Step 1 : Generate the contingency table of **attr** F and **freque** K

F	K=1	K=2
1	1	0
3	0	1
7	1	0
8	1	0
9	1	0
11	0	1
23	0	1
37	1	0
39	0	1

45	1	0
46	1	0
59	1	0

Step 2: Generate the chi_val table between clusters in the contingency table. We need to calculate the chi_square value for clusters pairs in the contingency table.

Chi_table	Lower_bound	Upper_bound	Chi_val
	1	3	1.9996
	3	7	1.9996
	7	8	0.0000
	8	9	0.0000
	9	11	1.9996
	11	23	0.0000
	23	37	1.9996
	37	39	1.9996
	39	45	1.9996
	45	46	0.0000
	46	59	0.0000

In this table **lower_bound** means the lower bound of the first cluster , **upper_bound** means the upper bound of the second cluster. **Chi_val** is the chi value of these two clusters. We will then find the min value of chi-square.

Step 3:If the min value of chi-square is below corresponding p_value, merge these two clusters into one cluster. And modify the original contingency table.

Before:

F	K=1	K=2
1	1	0
3	0	1
7-9	1+1+1	0+0+0
~~8~~	~~1~~	~~0~~
~~9~~	~~1~~	~~0~~
11-23	0+0	1+1
~~23~~	~~0~~	~~1~~
37	1	0
39	0	1
45-59	1+1+1	0+0+0
~~46~~	~~1~~	~~0~~
~~59~~	~~1~~	~~0~~

After:

F	K=1	K=2
1	1	0
3	0	1
7-9	3	0
11-23	0	2

37	1	0
39	0	1
45-59	3	0

Step 4: Generate new chi_table for our updated contingency table.

Chi_table	Lower_bound	Upper_bound	Chi_val
	1	3	1.9996
	3	9	3.999333
	7	23	4.999567
	11	37	2.999500
	37	39	1.999600
	39	59	3.999333

Step 5: Then we will do Step 3 and Step 4 again and again until the min chi_val is above our p_value , which means the adjacent clusters are different enough and we do not need to merge them into one cluster.

So much for the illustration of my chi_merge algorithm.

2. Discretize sepal length in cm using 5 different thresholds using chi-merge (0.10 0.05 0.025 0.02 0.01)

Code: chi_merge(myflower$sepal.length,myflower$class,0.1)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
4.3-4.8          16               0              0
4.9               4               1              1
5-5.4            25               5              0
5.5-5.7           4              15              2
5.8-6.2           1              15             10
6.3-7             0              14             25
7.1-7.9           0               0             12
> 
```

Code: chi_merge(myflower$sepal.length,myflower$class,0.05)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
4.3-5.4          45               6              1
5.5-5.7           4              15              2
5.8-7             1              29             35
7.1-7.9           0               0             12
> 
```

Code: chi_merge(myflower$sepal.length,myflower$class,0.025)

Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
4.3-5.4          45               6              1
5.5-5.7           4              15              2
5.8-7             1              29             35
7.1-7.9           0               0             12
>
```

Code: chi_merge(myflower$sepal.length,myflower$class,0.02)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
4.3-5.4          45               6              1
5.5-5.7           4              15              2
5.8-7             1              29             35
7.1-7.9           0               0             12
>
```

Code: chi_merge(myflower$sepal.length,myflower$class,0.01)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
4.3-5.4          45               6              1
5.5-5.7           4              15              2
5.8-7.9           1              29             47
>
```

3. Discretize petal length in cm using 5 different thresholds using chi-merge (0.10 0.05 0.025 0.02 0.01)

Code: chi_merge(myflower$petal.length,myflower$class,0.1)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
1-1.9            50               0              0
3-4.7             0              44              1
4.8-5.1           0               6             15
5.2-6.9           0               0             34
>
```

Code: chi_merge(myflower$petal.length,myflower$class,0.05)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
1-1.9            50               0              0
3-4.7             0              44              1
4.8-5.1           0               6             15
5.2-6.9           0               0             34
>
```

Code: chi_merge(myflower$petal.length,myflower$class,0.025)

Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
1-1.9            50               0              0
3-4.7             0              44              1
4.8-5.1           0               6             15
5.2-6.9           0               0             34
> 
```

Code: chi_merge(myflower$petal.length,myflower$class,0.02)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
1-1.9            50               0              0
3-4.7             0              44              1
4.8-5.1           0               6             15
5.2-6.9           0               0             34
> 
```

Code: chi_merge(myflower$petal.length,myflower$class,0.01)
Output:

```
        Iris-setosa Iris-versicolor Iris-virginica
1-1.9            50               0              0
3-4.7             0              44              1
4.8-5.1           0               6             15
5.2-6.9           0               0             34
> 
```