Question 1

```
rm(list=ls())
```

Implement K-means

```
library(ElemStatLearn)

nrow(zip.train)
```

```
## [1] 7291
```

```
ncol(zip.train)
```

```
## [1] 257
```

```
data <- zip.train
#data


customKmeans<-function(dataset=NA,k=NA, seed){
  if(is.na(dataset) || is.na(k)){
    stop("You must input valid parameters!!")
  }

  Eudist<-function(x,y){
    distance<-sqrt(sum((x-y)^2))
    return (distance)
  }

  set.seed(seed)

  rows.dataset<-nrow(dataset)
  continue.change=TRUE
  initPoint<-dataset[sample.int(rows.dataset,size = k),]
  formerPoint<-initPoint
  iterPoint<-matrix(0,nrow = k,ncol = ncol(dataset))


  error.matrix<-matrix(0,nrow=rows.dataset,ncol=k)
  while(continue.change){

    cluster.matrix<-matrix(0,nrow=rows.dataset,ncol=k)
    for(i in 1:rows.dataset){
      for(j in 1:k){
        error.matrix[i,j]<-Eudist(dataset[i,2:257],formerPoint[j,2:257])
      }
    }
```

```
    for(i in 1:rows.dataset){
      cluster.matrix[i,which.min(error.matrix[i,])]<-1
    }

    for(i in 1:k){
      iterPoint[i,]<-apply(dataset[which(cluster.matrix[,i] == 1),],2,"mean")
    }
    all.true<-c()

    for(i in 1:k){
      if(all(formerPoint[i,] == iterPoint[i,]) == T){
        all.true[i]<-TRUE
      }
    }
    formerPoint = iterPoint
    continue.change=ifelse(all(all.true) == T,F,T)
  }
  colnames(iterPoint)<-colnames(dataset)
  out=list()
  out[["centers"]]<-iterPoint
  out[["distance"]]<-error.matrix
  out[["cluster"]]<-rep(1,rows.dataset)
  for(i in 1:rows.dataset){
    out[["cluster"]][i]<-which(cluster.matrix[i,] == 1)
  }
  return <-out
}
```

One random initialization

```
out <- customKmeans(data, 5, 1)
result <- data.frame("data"=data, "cluster"=out$cluster)
cluster.count <- matrix(rep(0, 50), nrow=5, ncol=10)
for(i in 1:5)
{
  clusters <- subset(result, cluster==i);

  for(j in 0:9)
  {
    cluster.count[i, j+1] <- nrow(subset(clusters, data.1==j))
  }
}

print(cluster.count)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]  592    0  128  289   17  270  299    1   43     5
## [2,]  378    0   28    3    4   15   25    2    1     2
## [3,]   63 1005  456  103  206  140  319  230  190   154
## [4,]  133    0   41  137   38   47   20   48   21    20
## [5,]   28    0   78  126  387   84    1  364  287   463
```

In cluster 1, the most prevalent digits are: 0,3,6 In cluster 2, the most prevalent digits are: 0 In cluster 3, the most prevalent digits are: 1,2,6 In cluster 4, the most prevalent digits are: 0,3 In cluster 5, the most prevalent digits are: 9,4,7

Ten random initialization

```
for(seed in 1:10)
{
  out<-customKmeans(data, 5, seed);

  whole.data <- data.frame("distance"=out$distance, "cluster_result"=out$cluster, "data"=data)
  error <- 0;
  for(cluster in 1:5)
  {
    cluster.data <- subset(whole.data, cluster_result==cluster);
    error <- error+sum(cluster.data[,cluster]);
    pca <- prcomp(cluster.data[, (8:ncol(cluster.data))])

    summary(pca)

    #plot(x=dim[,1], y = dim[,2], col=cluster)

  }
  print(error)
}
```

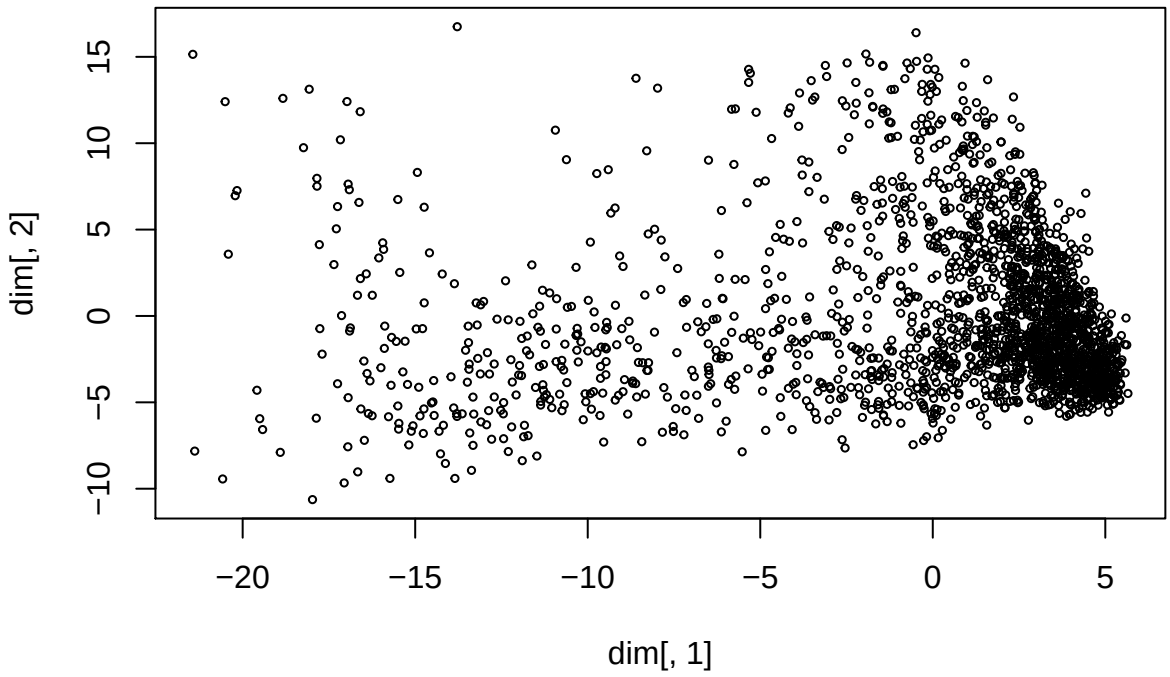
```
## [1] 85055.55
## [1] 88201.81
## [1] 105748.4
## [1] 85931.7
## [1] 91439.58
## [1] 86676.72
## [1] 81460.47
## [1] 90355.09
## [1] 102987.3
## [1] 87571.46
```

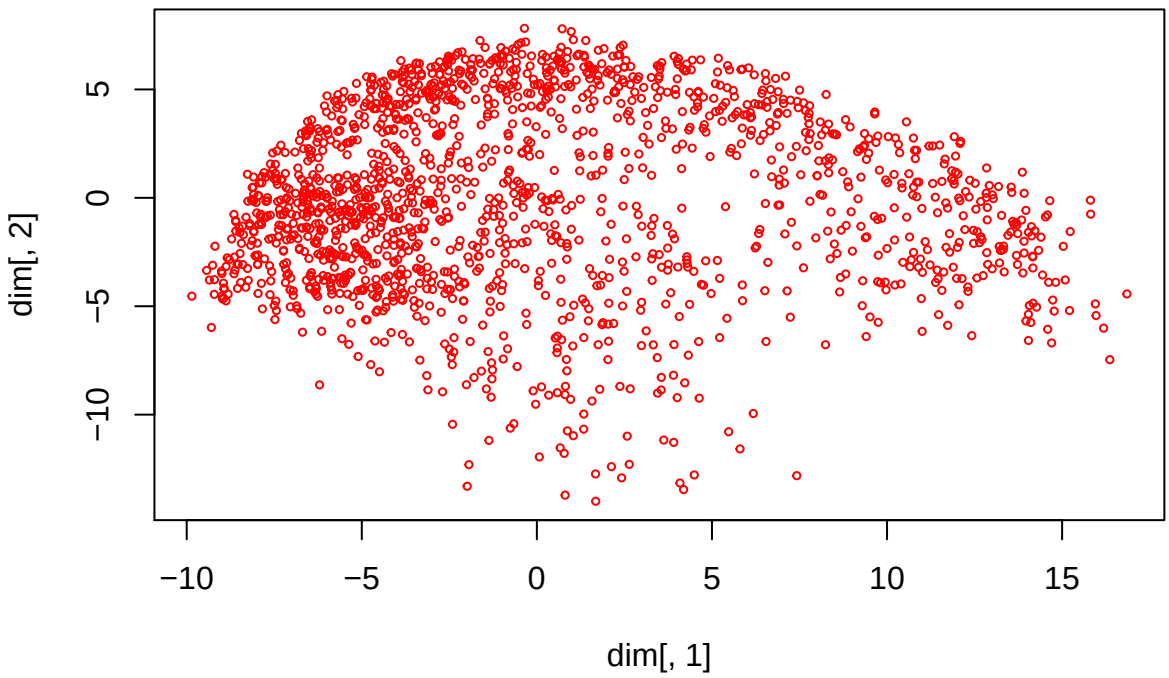
Best error occurs when seed is 7. Repeat to get plots:

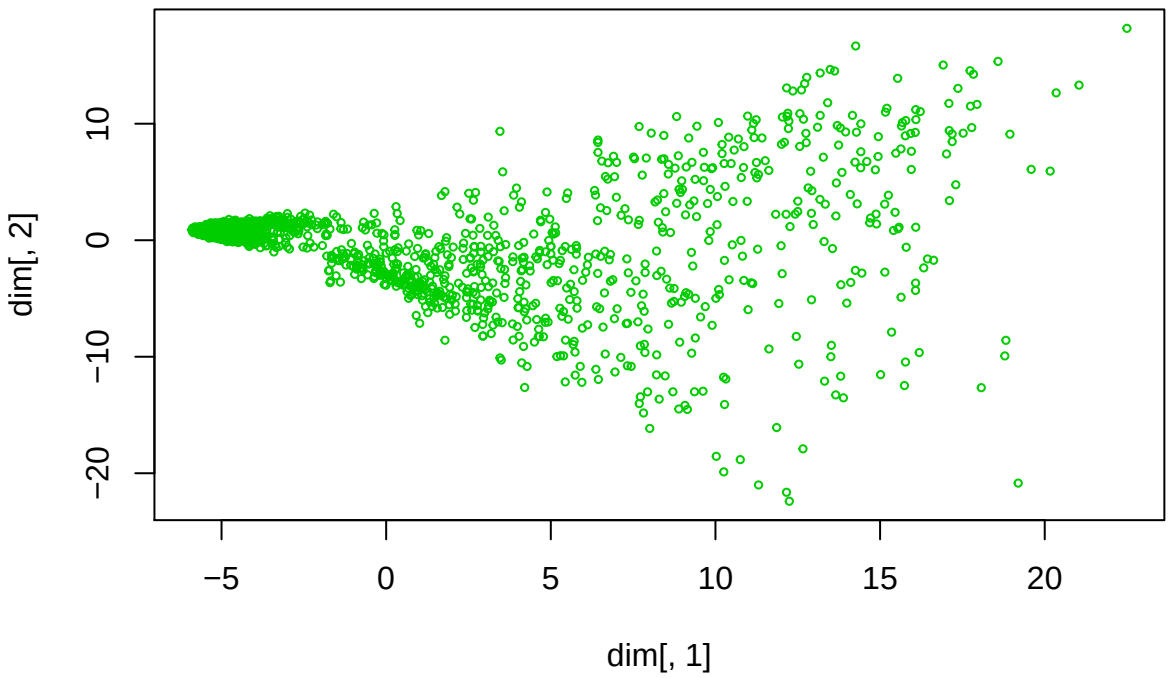
Plots

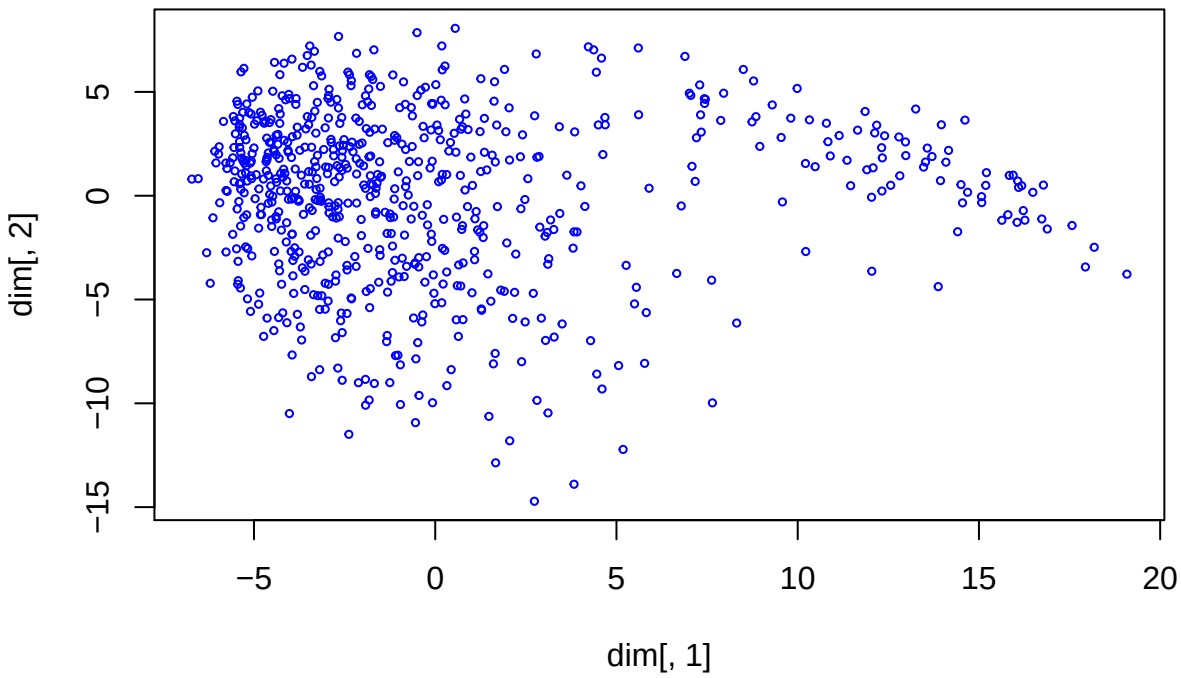
```
out<-customKmeans(data, 5, 7);

whole.data <- data.frame("distance"=out$distance, "cluster_result"=out$cluster, "data"=data)

for(cluster in 1:5)
{
  cluster.data <- subset(whole.data, cluster_result==cluster);
```

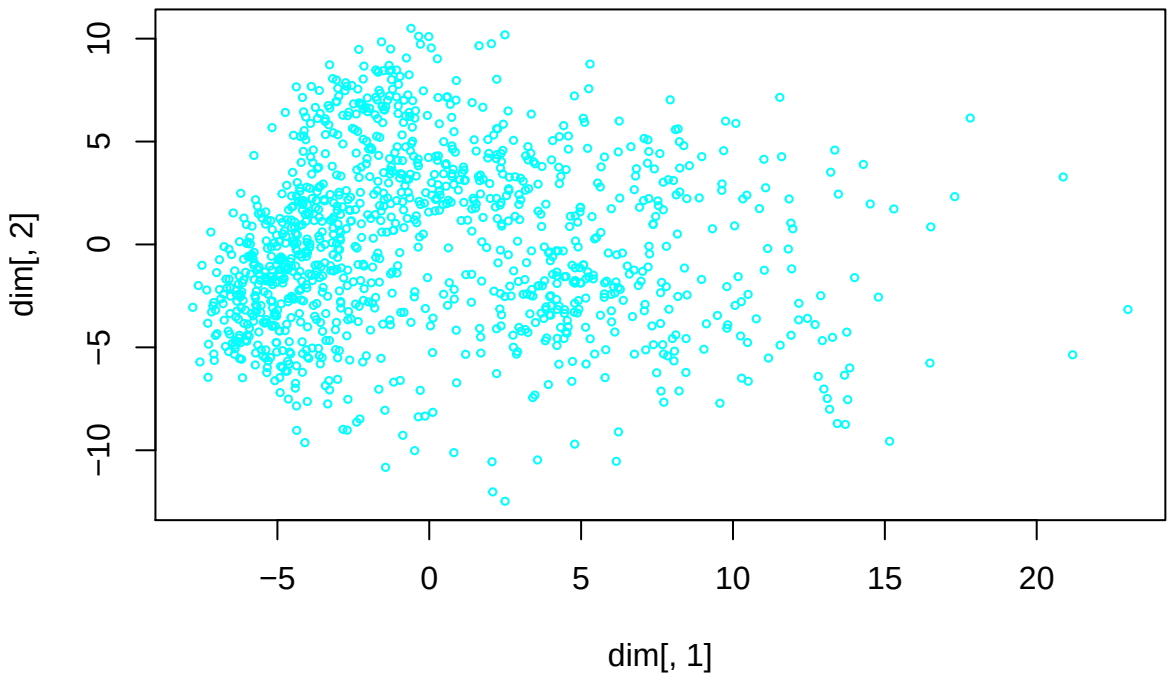
```
  pca <- prcomp(cluster.data[, (9:ncol(cluster.data))], scale. = T)

  dim <- pca$x

  plot(x=dim[,1], y = dim[,2], col=cluster, cex=0.5)
}
```




dim[, 2]
5
0
−5
−10
−10
−5
0
5
10
15
dim[, 1]


dim[, 2]
10
0
−10
−20
−5
0
5
10
15
20
dim[, 1]


dim[, 2]
5
0
−5
−10
−15
−5
0
5
10
15
20
dim[, 1]



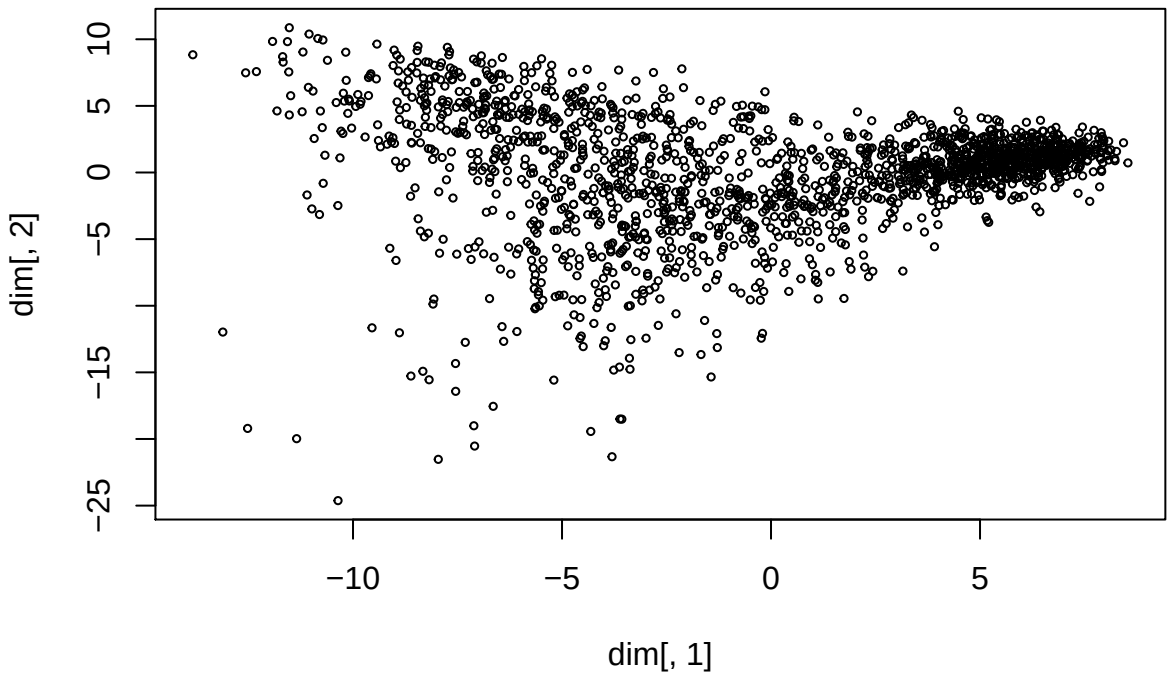
Compare with built-in k-means

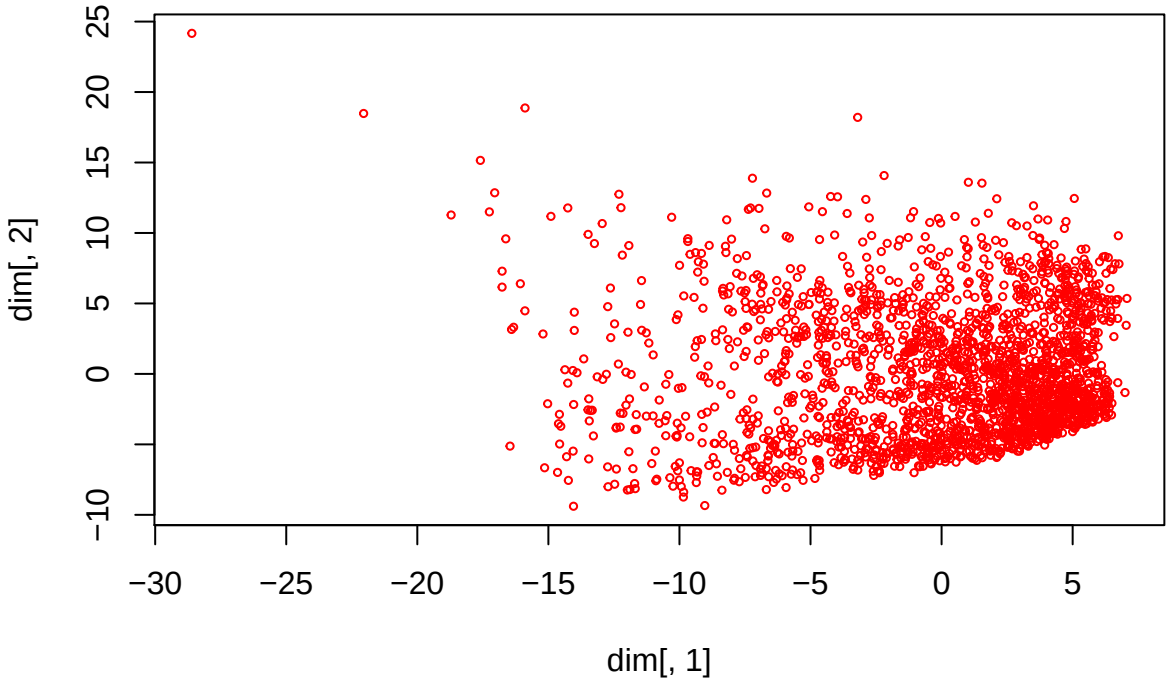
```
set.seed(1)

cl <- kmeans(data[, (1:ncol(data))], centers=5)

whole.data <- data.frame("cluster"=cl$cluster, "data"=data )


counts <- matrix(rep(0, 50), nrow=5, ncol=10)

for(i in 1:5)
{
  for(j in 0:9)
  {
    counts[i,j+1] = nrow(subset(subset(whole.data, cluster==i), data.1==j))

  }
}

counts
```

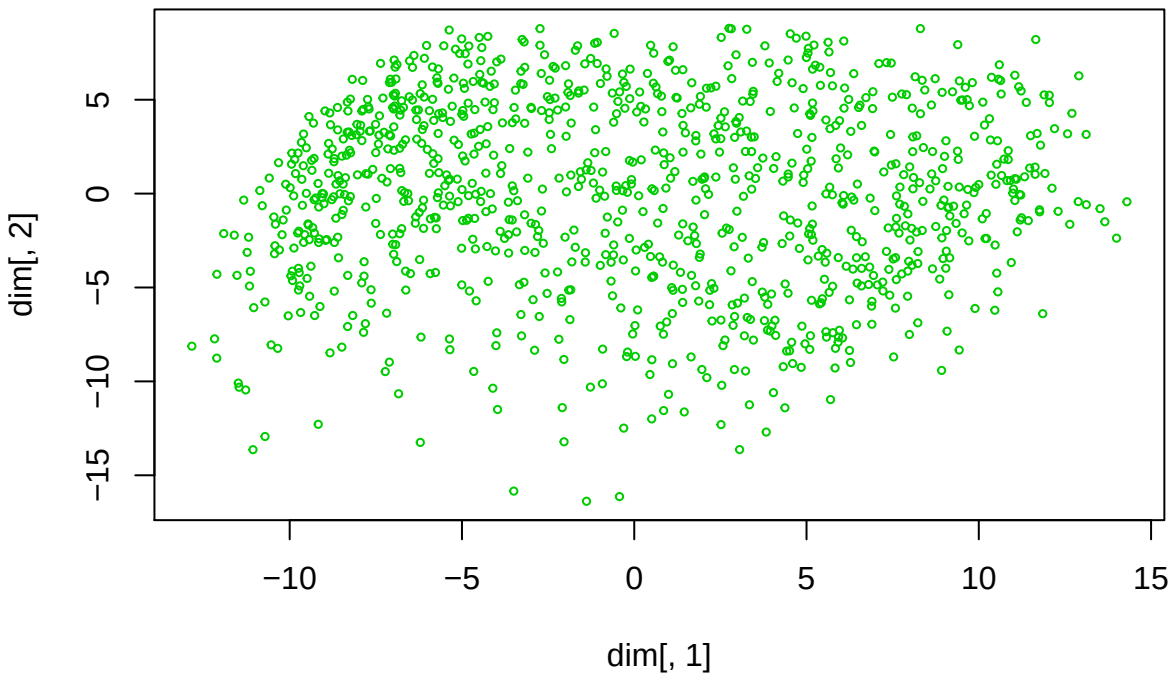
```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]  194    1  616   26  200  125  640    5   14     2
```

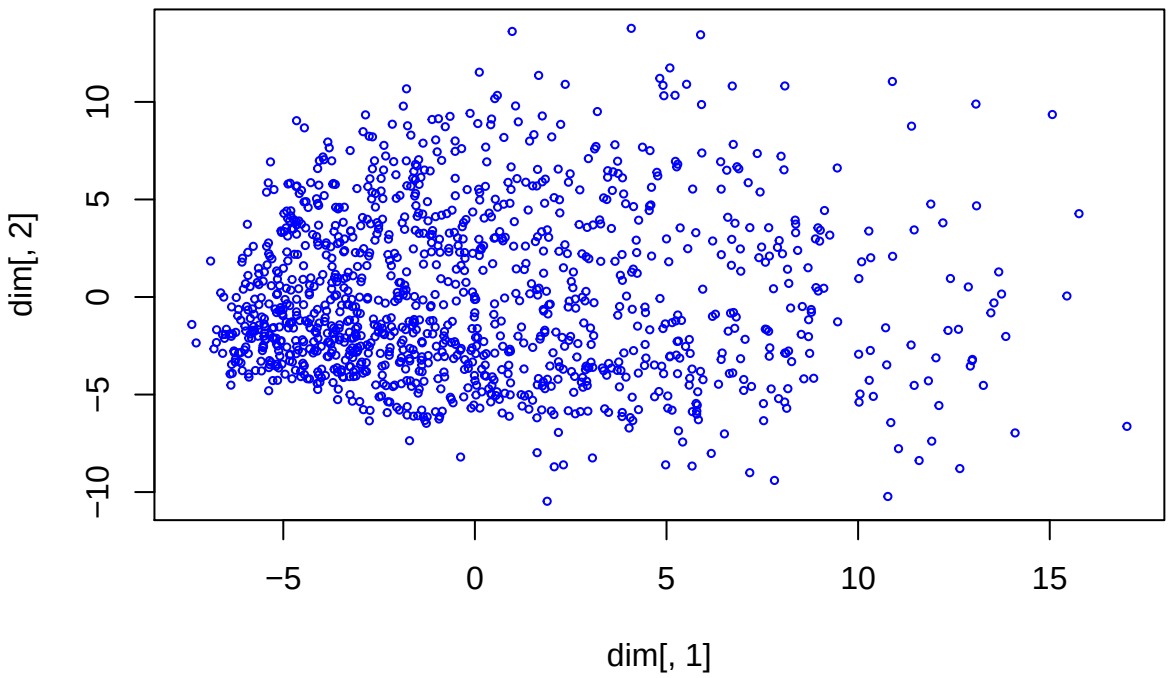
```
## [2,]  986    0   23    2    3    6   16    0    1    0
## [3,]    0 1004    7    1   43    0    3    0    1    0
## [4,]   14    0   82  618    4  390    3    0   87    0
## [5,]    0    0    3   11  402   35    2  640  439  642
```

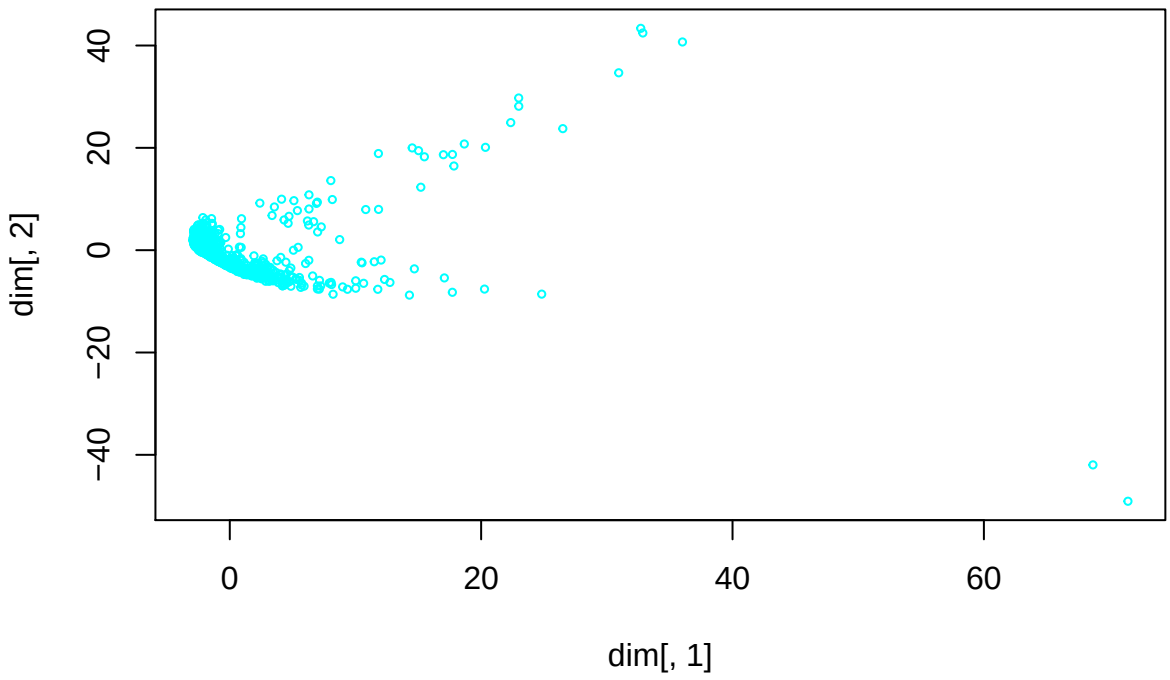
```
cl2<-kmeans(data, 5, 10);

whole.data <- data.frame("cluster_result"=cl2$cluster, "data"=data)

for(cluster in 1:5)
{
  cluster.data <- subset(whole.data, cluster_result==cluster);
  cluster.data <- cluster.data[,!apply(cluster.data, MARGIN = 2, function(x) max(x, na.rm = TRUE) == mi

  pca <- prcomp(cluster.data[, (1:ncol(cluster.data))], scale.=TRUE)

  dim <- pca$x

  plot(x=dim[,1], y = dim[,2], col=cluster, cex=0.5)

}
```




dim[, 2]
25
20
15
10
5
0
−10
−30
−25
−20
−15
−10
−5
0
5
dim[, 1]


dim[, 2]
5
0
−5
−10
−15
−10
−5
0
5
10
15
dim[, 1]


dim[, 2]
10
5
0
−5
−10
−5
0
5
10
15
dim[, 1]




```
counts <- matrix(rep(0, 50), nrow=5, ncol=10)

for(i in 1:5)
{
  for(j in 0:9)
  {
    counts[i,j+1] = nrow(subset(subset(whole.data, cluster_result==i), data.1==j))

  }
}

counts
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]  194    1  616   26  200  125  640    5   14     2
## [2,]    0    0    3   11  402   35    2  640  439   642
## [3,]  986    0   23    2    3    6   16    0    1     0
## [4,]   14    0   82  618    4  390    3    0   87     0
## [5,]    0 1004    7    1   43    0    3    0    1     0
```

Compare by count of digits in each plot

1. My implementation of kmeans:

cluster	0	1	2	3	4	5	6	7	8	9
1	592	0	128	289	17	27	299	1	43	5
2	378	0	28	3	4	15	25	2	1	2
3	63	1005	456	103	206	140	319	230	190	154
4	133	0	41	137	38	47	20	48	21	20
5	28	0	78	126	387	84	1	364	287	463

2.

cluster	0	1	2	3	4	5	6	7	8
1	194	1	616	26	200	125	640	5	14	2
2	986	0	23	2	3	6	16	0	1	0
3	0	1004	7	1	43	0	3	0	1	0
4	14	0	82	618	4	390	3	0	87	0
5	0	0	3	11	402	35	2	640	439	642

3.

cluster	0	1	2	3	4	5	6	7	8
1	194	1	616	26	200	125	640	5	14	2
2	0	0	3	11	402	35	2	640	439	642
3	986	0	23	2	3	6	16	0	1	0
4	14	0	82	618	4	390	3	0	87	0
5	0	1004	7	1	43	0	3	0	1	0