R Zone 5: Chapter 19 and 20

Brandon Hufstetler, Garrett Alarcon, Jinan Andrews, Anson Cheng, Nick Forrest, and Nestor Herandez

12 August 2019

Chapter 19

INSTALL THE REQUIRED PACKAGE AND CREATE THE DATA

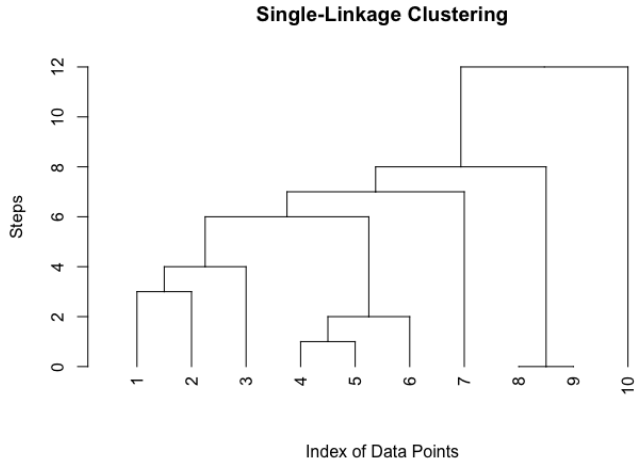
```
library(cluster)
data <- c(2, 5, 9, 15, 16, 18, 25, 33, 33, 45)
```

SINGLE-LINKAGE CLUSTERING

In the dendrogram below, the height of each linkage shows the distance between the groups it is linking. For example, There is a linkage connecting the group (1,2,3) to (4,5,6). The height of the linkage is 6. Between the two groups, the two closest neighbors (because we are using the 'single' method) are 3 (9) and 4 (15) having a distance of 6. The single linkage method is based on the minimum distance between any record in cluster A and any record in cluster B.

```
agn <- agnes(data, diss = FALSE, stand = FALSE, method = "single")

# Make and plot the dendrogram
dend_agn <- as.dendrogram(agn)
plot(dend_agn,
     xlab = "Index of Data Points",
     ylab = "Steps",
     main = "Single-Linkage Clustering")
```

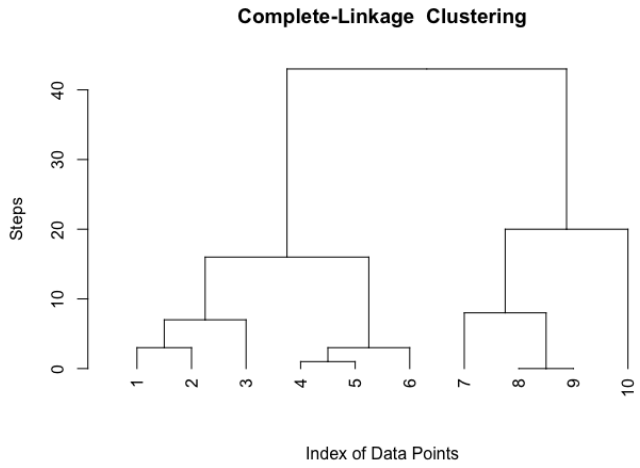


COMPLETE-LINKAGE CLUSTERING

The complete linkage method is based on the maximum distance between any record in cluster A and cluster B. Notice how observation 7 joins the (8,9) cluster here as opposed to the (1,2,3,4,5,6) cluster as seen in the single linkage method.

```
agn_complete <- agnes(data, diss = FALSE, stand = FALSE, method = "co

# Make and plot the dendrogram
dend_agn_complete <- as.dendrogram(agn_complete)
plot(dend_agn_complete,
     xlab = "Index of Data Points",
     ylab = "Steps",
     main = "Complete-Linkage  Clustering")
```



K-MEANS CLUSTERING

With k-means clustering, a number of centroids is predetermined and k records are randomly assigned as a centroid. For each record, the nearest cluster centroid is found and assign it to that cluster. Recalculate the

cluster centroids. Continue doing this until convergence or termination. In the example here, there are 8 data points assigned to 2 clusters. The centroid locations are shown along with the sum of squares for each cluster.

```
  # Create the data matrix from Table 10.1
  m <- matrix(c(1,3,3,3,4,3,5,3,1,2,4,2,1,1,2,1), byrow=TRUE, ncol = 2)
  km <- kmeans(m, centers = 2)
  km
```

```
## K-means clustering with 2 clusters of sizes 4, 4
##
## Cluster means:
##   [,1] [,2]
## 1 4.00 2.75
## 2 1.25 1.75
##
## Clustering vector:
## [1] 2 1 1 1 2 1 2 2
##
## Within cluster sum of squares by cluster:
## [1] 2.75 3.50
##  (between_SS / total_SS =  73.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Chapter 20

OPEN 'KOHONEN' PACKAGE. READ IN AND PREPARE THE DATA

The first step taken here is to create indicator variables for International Plan, Voice Mail Plan, and Churn variables, then normalize the data.

```
  setwd("~/IMGT680")
  #install.packages("kohonen")
  library(kohonen)
  churn <- read.csv(file = "churn.txt", stringsAsFactors=TRUE)
  IntPlan <- VMPlan <- Churn <- c(rep(0, length(churn$Int.l.Plan))) # F
  for (i in 1:length(churn$Int.l.Plan)) {
```

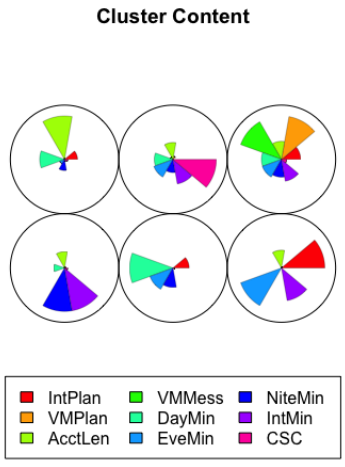
```
  if (churn$Int.l.Plan[i]=="yes") IntPlan[i] = 1
  if (churn$VMail.Plan[i]=="yes") VMPlan[i] = 1
  if (churn$Churn[i] == "True.") Churn[i] = 1
}
AcctLen <- (churn$Account.Length - mean(churn$Account.Length)) / sd(c
VMMess <- (churn$VMail.Message - mean(churn$VMail.Message)) / sd(chur
DayMin <- (churn$Day.Mins - mean(churn$Day.Mins)) / sd(churn$Day.Mins
EveMin <- (churn$Eve.Mins - mean(churn$Eve.Mins)) / sd(churn$Eve.Mins
NiteMin <- (churn$Night.Mins - mean(churn$Night.Mins)) / sd(churn$Nig
IntMin <- (churn$Intl.Mins - mean(churn$Intl.Mins)) / sd(churn$Intl.M
CSC <- (churn$CustServ.Calls - mean(churn$CustServ.Calls)) / sd(churn
```

RUN THE ALGORITHM TO GET A 3X2 KOHONEN NETWORK

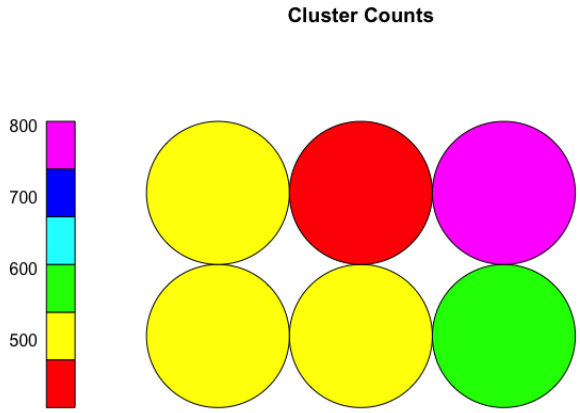
A kohonen network is a type of self-organizing map, which represents a special class of neural networks. The goal of self-organizing maps is to convert a complex high-dimensional input signal into a simpler low-dimensional discrete map. They exhibit three characteric processes: Competition, Cooperation, and Adaptation. The cluster contents shown below illustrate the component make-up of each cluster. The cluster counts show the records in each cluster. Each cluster is then plotted with a jitter function to show the makeup of each cluster. The observations with a voice mail plan are colored red. Finally, a table showing the percent of churners in each cluster is created.

```
# Make the variables into one matrix, and make sure the records are t
dat <- t(rbind(IntPlan, VMPlan, AcctLen, VMMess, DayMin, EveMin, Nite
som.6 <- som(dat, grid = somgrid(3, 2), rlen = 170, alpha = c(0.3, 0.

# Plot the make-up of each cluster
plot(som.6, type = c("codes"), palette.name = rainbow, main = "Cluste
```




```
# Plot the counts in each cluster
plot(som.6, type = c("counts"), palette.name = rainbow, main = "Clust
```



PLOT MAKE-UP OF CLUSTERS

```
  som.6$unit.classif # Winning Clusters
```

```
##    [1] 6 6 4 2 5 2 6 4 3 6 5 2 4 5 5 5 6 5 6 1 3 5 4 4 3 3 6 5 2 3 1
##   [35] 6 6 6 3 6 3 5 6 1 4 2 6 5 4 5 3 2 2 1 3 5 1 4 6 3 4 4 6 6 6 2
##   [69] 4 5 1 6 4 2 2 2 2 5 2 1 3 1 6 6 3 6 5 3 6 3 3 4 1 2 5 4 3 5 4
##  [103] 4 1 1 5 6 6 6 4 3 1 5 1 1 6 3 2 6 2 2 6 5 4 4 1 5 6 3 6 5 2 2
##  [137] 5 5 6 6 3 6 6 2 1 5 3 2 4 6 3 1 2 2 5 3 2 6 2 6 3 1 6 1 4 6 6
##  [171] 3 6 1 3 4 2 3 3 5 2 5 5 6 2 4 2 1 2 4 2 3 1 2 3 6 6 1 4 6 1 6
##  [205] 3 6 1 5 1 5 2 3 6 6 3 5 3 5 3 4 5 6 2 2 1 2 5 6 5 3 2 6 5 3 2
##  [239] 4 2 4 1 2 2 1 2 6 1 5 6 6 5 5 6 3 6 1 3 1 3 6 1 3 1 4 6 6 1 6
##  [273] 6 5 6 5 4 6 6 5 4 2 6 6 2 6 4 1 1 2 6 4 2 6 1 4 3 1 5 2 1 2 3
##  [307] 2 5 3 4 2 6 2 3 1 3 6 6 4 6 2 6 3 3 1 6 6 6 3 3 3 2 5 3 2 4 3
##  [341] 1 1 6 6 6 4 6 4 6 5 5 6 3 1 2 1 4 6 6 6 4 1 6 1 4 4 5 1 3 3 4
##  [375] 1 6 4 2 3 4 5 1 4 6 3 5 1 2 3 1 2 3 5 4 2 6 3 2 2 3 6 1 6 4 5
##  [409] 3 4 2 3 2 4 1 2 4 4 6 3 6 2 3 6 6 4 4 6 5 3 2 3 1 3 6 6 3 2 3
##  [443] 6 4 1 3 5 2 3 5 3 6 2 6 2 3 1 3 6 4 2 1 4 1 6 5 6 1 6 6 2 3 4
##  [477] 2 6 1 1 3 4 1 3 5 2 3 2 4 3 1 3 3 6 2 4 6 3 5 3 6 6 1 6 1 1 6
##  [511] 4 3 2 6 6 3 3 2 3 5 4 5 5 3 3 4 2 1 1 3 6 2 6 6 5 3 6 4 4 3 6
##  [545] 6 6 2 6 6 6 5 4 2 3 3 6 3 6 2 1 1 3 6 1 6 5 6 2 2 6 2 1 3 6 2
##  [579] 6 3 2 6 4 1 3 1 2 4 5 3 3 3 1 6 1 6 5 1 3 6 3 3 4 3 2 2 3 6 2
##  [613] 5 6 6 6 6 3 4 1 1 4 6 4 4 3 6 5 3 2 1 4 6 6 1 5 4 6 6 1 2 2 3
##  [647] 6 4 6 2 3 2 6 6 6 5 1 4 2 4 2 6 2 4 4 6 4 2 4 6 6 6 4 5 4 2 2
##  [681] 1 1 6 4 5 6 4 1 2 2 1 4 5 6 5 1 4 3 2 2 4 3 1 6 4 3 2 6 6 4 5
##  [715] 4 3 6 2 4 1 6 5 3 5 6 1 1 2 6 6 5 2 2 6 6 4 5 1 3 2 6 3 5 4 2
##  [749] 3 3 6 6 6 1 1 3 2 3 6 1 3 2 4 6 6 4 3 6 5 3 4 5 2 4 1 3 6 6 5
##  [783] 6 1 6 5 2 2 4 5 6 3 6 1 3 3 6 3 6 6 6 2 2 1 2 2 4 6 3 4 3 2 4
##  [817] 2 4 1 1 1 1 1 6 4 3 4 3 2 3 1 1 2 3 1 3 6 6 6 6 3 5 3 1 6 6 1
```

```
##  [851] 4 5 5 5 6 2 1 4 6 5 4 1 6 1 4 5 6 1 2 4 1 5 6 1 5 3 3 4 2 6 2
##  [885] 1 6 6 4 2 1 3 4 6 4 5 3 1 6 1 6 6 2 5 2 2 5 3 1 5 6 6 5 1 6 4
##  [919] 2 4 4 2 1 1 2 1 4 6 4 2 1 1 3 3 4 5 6 2 1 1 3 2 4 1 6 1 2 6 1
##  [953] 5 6 4 6 6 3 6 1 2 2 5 6 6 3 6 2 6 3 3 2 2 3 5 2 1 5 5 4 5 6 6
##  [987] 4 6 1 4 6 6 6 1 6 4 4 3 3 4 4 1 6 6 6 2 3 4 6 4 2 3 4 1 1 4 2
## [1021] 6 3 4 3 3 6 6 1 2 1 6 2 2 2 3 3 6 3 5 1 5 6 6 5 5 4 4 3 2 1 6
## [1055] 1 4 2 6 3 4 1 6 4 1 6 6 6 1 3 2 3 4 6 2 4 3 6 1 2 4 4 2 5 5 5
## [1089] 6 3 3 3 4 4 4 2 3 6 2 4 6 2 5 1 5 6 1 6 4 4 3 1 1 3 6 3 5 3 1
## [1123] 4 6 4 1 6 6 6 3 1 6 6 5 3 6 6 6 3 4 6 4 5 6 4 2 4 6 4 1 5 6 2
## [1157] 3 4 2 3 6 2 4 6 6 2 5 1 4 3 3 3 4 1 6 6 5 1 6 6 2 6 4 6 2 5 3
## [1191] 4 4 5 5 1 3 4 5 6 6 6 6 1 6 2 4 6 6 4 6 2 2 3 1 3 4 5 6 6 2 3
## [1225] 4 3 3 6 6 4 2 2 2 1 3 5 3 5 6 2 6 5 5 6 6 3 5 4 1 2 4 5 4 5 6
## [1259] 1 1 6 2 5 6 3 6 2 2 3 6 3 6 5 5 3 6 2 6 4 3 1 1 5 4 5 6 6 4 2
## [1293] 3 4 3 2 2 6 6 2 5 4 4 2 3 3 5 5 3 1 3 1 3 4 4 4 2 6 4 2 6 2 2
## [1327] 4 4 3 4 6 5 6 6 4 1 5 2 2 5 3 6 6 2 2 5 3 2 1 6 2 3 1 1 3 6 1
## [1361] 1 1 5 6 1 4 1 2 5 2 1 4 1 5 6 6 2 6 6 2 1 3 6 6 1 3 6 4 5 3 3
## [1395] 3 1 6 3 6 5 4 3 2 4 6 5 3 5 4 6 1 4 6 6 4 3 2 3 6 2 1 2 6 6 1
## [1429] 3 6 6 1 1 2 6 3 6 6 1 3 4 6 2 4 1 1 6 4 5 5 3 6 1 6 6 3 6 4 4
## [1463] 2 2 5 1 4 6 6 6 6 3 2 2 1 1 6 3 4 1 3 6 2 1 1 1 1 6 4 3 2 1 3
## [1497] 6 4 3 5 3 2 5 6 1 4 4 1 5 6 2 2 2 2 1 4 2 4 3 4 6 5 2 4 3 3 6
## [1531] 2 5 6 1 1 1 3 5 5 4 1 6 1 1 4 6 1 3 2 1 3 4 3 2 4 6 1 6 6 4 3
## [1565] 4 1 6 1 6 6 1 4 1 1 6 6 5 3 3 6 6 6 1 6 1 1 3 1 1 2 6 6 2 2 6
## [1599] 6 1 6 3 6 6 3 3 1 5 6 1 5 6 4 2 3 1 1 1 6 5 6 6 3 3 2 6 4 2 1
## [1633] 3 1 3 5 6 2 5 1 5 2 3 1 4 6 4 1 1 3 6 1 4 3 3 4 6 4 4 6 5 6 5
## [1667] 3 3 4 6 6 1 1 5 2 3 3 3 1 2 3 3 6 6 3 5 6 5 5 6 4 2 2 6 5 1 4
## [1701] 6 2 5 6 4 6 1 5 5 5 4 6 5 5 1 6 6 1 1 1 4 2 3 3 4 1 3 1 5 6 3
## [1735] 3 3 2 4 6 2 6 2 1 2 6 2 1 3 6 5 2 4 1 2 3 6 6 2 1 6 2 6 4 6 2
## [1769] 3 3 1 6 3 6 2 6 2 5 6 3 3 4 2 6 3 6 1 6 6 4 4 3 4 2 2 6 1 6 2
## [1803] 5 3 1 5 4 5 5 1 3 1 4 2 3 6 3 4 1 4 1 6 3 6 6 5 5 1 3 6 2 5 3
## [1837] 3 3 6 1 6 6 2 1 6 1 6 6 4 1 6 5 6 4 2 3 3 3 3 3 2 6 4 3 3 5 3
## [1871] 2 2 1 6 2 3 2 2 2 1 5 2 6 6 3 2 4 6 3 1 5 1 4 2 1 6 1 6 4 6 3
## [1905] 6 2 6 6 6 6 4 5 5 2 6 4 1 1 4 5 2 5 5 1 1 1 2 5 3 1 4 2 2 2 2
## [1939] 6 3 1 4 2 1 6 3 2 2 4 3 5 6 4 4 2 4 6 4 5 6 1 3 5 3 4 4 2 5 2
## [1973] 3 5 5 2 6 2 5 2 1 2 3 5 5 6 3 3 1 6 1 3 2 2 3 6 3 5 3 3 5 2 1
## [2007] 6 2 6 3 3 5 4 3 5 2 5 6 6 6 2 6 3 1 6 3 1 6 2 3 6 5 3 4 1 5 4
## [2041] 1 6 6 5 6 5 1 5 6 6 4 2 4 3 3 1 5 1 1 2 6 6 3 1 1 3 6 5 3 2 3
## [2075] 3 6 5 1 4 1 3 1 6 5 6 1 3 5 3 3 6 2 2 3 4 3 6 2 3 2 6 4 2 2 3
## [2109] 3 6 6 1 5 2 5 1 1 5 2 2 1 3 6 4 1 6 2 5 6 6 2 6 4 6 1 3 3 4 5
## [2143] 3 5 4 1 2 5 3 6 4 1 6 3 1 4 5 4 3 1 3 1 6 2 5 3 4 4 4 4 1 2 4
## [2177] 2 1 3 6 4 6 6 3 6 1 2 5 3 5 6 4 2 4 6 6 3 4 4 2 4 1 6 5 2 1 3
```

```
## [2211] 2 6 6 3 3 2 2 6 5 6 5 4 6 5 1 1 3 2 2 6 4 6 1 1 4 1 1 5 1 2 3
## [2245] 4 6 3 3 5 5 2 2 3 1 6 4 2 6 2 4 6 5 6 6 2 6 6 2 6 6 1 1 1 2 6
## [2279] 2 1 3 6 5 5 1 1 6 2 2 6 5 3 3 6 2 4 3 3 3 6 3 1 1 4 6 1 2 6 2
## [2313] 1 4 6 3 1 5 3 6 3 1 6 2 5 2 1 5 1 3 3 6 3 2 5 5 6 5 1 3 6 2 1
## [2347] 4 6 6 2 4 4 3 3 2 1 2 3 3 6 2 5 2 1 6 6 6 2 2 5 5 1 1 2 4 3 2
## [2381] 5 6 3 6 2 1 4 5 4 4 3 6 3 6 5 5 4 2 4 4 6 3 5 3 3 6 3 6 2 3 3
## [2415] 2 5 3 3 1 2 3 5 6 2 6 6 1 1 5 6 6 6 5 2 3 6 3 6 3 4 4 1 6 1 5
## [2449] 4 5 1 4 3 6 3 3 5 6 6 5 1 2 1 4 6 5 6 2 6 5 4 6 3 6 6 2 1 3 6
## [2483] 4 6 6 6 2 4 2 5 1 3 2 5 4 1 6 6 1 3 6 3 1 4 3 5 4 1 6 3 1 4 6
## [2517] 1 1 2 2 6 4 3 5 2 3 1 4 1 4 2 4 4 3 3 4 2 4 3 6 6 3 2 6 3 4 3
## [2551] 2 3 2 5 4 4 1 1 3 6 2 6 4 1 3 4 1 2 5 5 6 2 4 4 3 1 6 4 4 2 3
## [2585] 5 5 3 5 6 3 6 2 5 3 2 2 1 3 4 2 1 6 3 2 5 6 3 5 6 1 6 3 2 2 3
## [2619] 6 6 1 6 5 4 1 1 6 3 5 6 3 6 6 2 1 3 4 5 3 6 6 6 6 5 3 2 5 2 6
## [2653] 1 6 2 2 2 5 3 1 5 4 3 1 6 5 1 6 4 4 4 4 2 4 3 2 3 2 2 4 6 1 3
## [2687] 6 6 3 1 2 1 4 4 3 2 2 6 3 3 4 2 4 1 1 5 3 3 2 6 2 3 3 6 6 6 6
## [2721] 1 2 3 6 4 1 4 2 3 3 6 1 5 4 6 6 6 1 4 3 4 2 6 2 2 6 5 4 4 3 4
## [2755] 1 3 3 2 1 3 1 1 5 6 6 1 3 3 6 1 6 6 5 6 6 6 6 2 3 4 1 1 5 6 5
## [2789] 4 6 6 4 3 6 4 3 1 4 1 5 2 5 3 5 3 6 2 2 6 4 6 1 6 1 3 6 1 3 4
## [2823] 3 3 2 3 4 5 5 6 1 2 5 4 4 6 3 1 6 1 6 3 6 3 4 2 1 6 4 4 3 4 1
## [2857] 4 4 6 1 1 5 3 2 4 6 1 1 1 6 3 4 4 2 2 6 1 4 4 6 4 3 1 4 4 4 3
## [2891] 4 1 6 6 5 2 2 6 3 6 3 5 2 5 4 4 2 3 4 4 4 3 4 4 6 5 1 2 2 4 3
## [2925] 2 4 2 6 4 5 4 2 1 3 1 4 2 6 3 1 2 6 2 6 3 3 6 4 2 4 1 1 5 5 5
## [2959] 5 4 5 5 2 6 2 4 4 3 3 3 6 4 6 3 4 1 1 4 1 5 5 1 1 4 3 2 6 5 5
## [2993] 6 4 2 6 6 6 2 2 1 3 4 1 3 3 1 4 6 6 3 6 1 4 4 6 1 5 1 1 6 1 6
## [3027] 5 6 1 6 5 1 4 3 6 5 5 2 6 3 1 2 4 4 3 2 3 3 1 5 5 3 1 2 4 4 4
## [3061] 1 1 6 6 5 6 4 1 6 6 5 6 2 6 3 6 2 6 1 5 3 5 5 1 3 4 5 2 6 6 1
## [3095] 1 6 6 4 3 3 1 6 6 1 2 2 6 1 6 4 6 6 5 2 1 5 4 3 6 1 6 4 5 6 6
## [3129] 5 6 6 5 4 3 5 1 6 1 3 2 5 6 3 5 5 5 3 5 2 2 3 2 4 2 6 1 5 5 2
## [3163] 3 6 2 6 2 4 4 5 3 1 2 1 6 6 4 6 3 4 1 5 5 3 2 1 5 6 4 2 5 2 6
## [3197] 1 2 6 4 4 6 5 6 3 2 3 6 2 6 3 1 6 1 4 6 4 6 6 6 4 3 6 1 5 2 2
## [3231] 2 6 4 4 6 6 6 3 5 6 2 4 6 5 2 5 6 4 1 4 3 2 3 6 6 4 1 1 4 3 3
## [3265] 6 2 6 3 2 3 3 3 2 1 2 6 6 1 1 6 5 2 6 5 6 1 6 6 3 2 4 6 4 3 3
## [3299] 3 1 6 2 3 6 5 6 6 5 1 5 1 3 1 4 3 4 6 4 3 1 5 1 5 5 4 1 3 1 6
## [3333] 6
```

```
  som.6$grid$pts # Plot locations
```

```
##      x y
## [1,] 1 1
## [2,] 2 1
## [3,] 3 1
```

```
## [4,] 1 2
## [5,] 2 2
## [6,] 3 2
```

```
  coords <- matrix(0, ncol = 2, nrow = dim(dat)[1])
  for(i in 1:dim(dat)[1]){
    coords[i,] <-
      som.6$grid$pts[som.6$unit.classif[i],]

  }
  pchVMPlan <- ifelse(dat[,2]==0 , 1, 16)
  colVMPlan <- ifelse(dat[,2]==0 , 1, 2)
  plot(jitter(coords), main = "Kohonen Network colored by VM Plan",
       col = colVMPlan,
       pch = pchVMPlan)
```

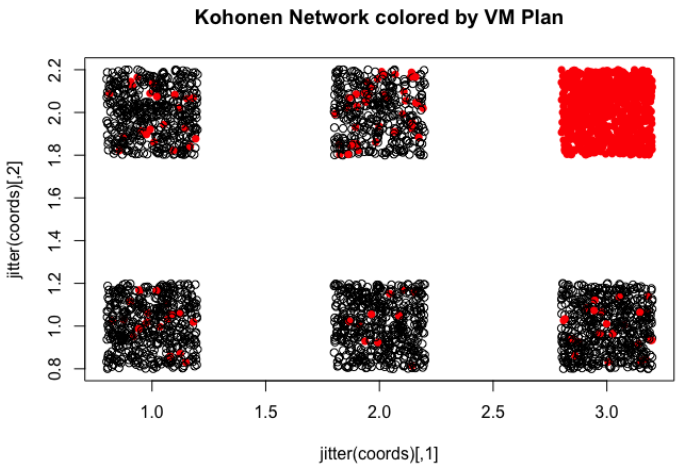


TABLE OF PERCENT CHURN BY CLUSTER

```
  c.table <- table(Churn, som.6$unit.classif)
  round(prop.table(c.table, 2)*100, 2)
```

```
##
## Churn     1     2     3     4     5     6
##     0 90.87 73.95 91.69 89.72 66.42 92.05
##     1  9.13 26.05  8.31 10.28 33.58  7.95
```