BDCT - Lecture 2

Clustering using R

This tutorial will introduce clustering technique in R. This will follow the same recipe that was in the study guide that used Excel. It will touch upon the Wine dataset. R is powerful statistical package. Although the commands may look complicated if you are a beginner, they provide solid training of the basics that you could use to build on the future big data applications.

This tutorial is closely adopted to the Excel guide. As with any program, there are multiple ways to get at the answer - including simpler versions. This program is not necessarily the simplest, although illustrates the flexibility with R. The key take away is to gets hand on experience using R.

Lecture 2 - Clustering Wine data

Recall, the business question in the case is a company that is focused on running promotions for its customers. The question is how would it segment the customers to decide on the type of offers it would promote. The data that was provided is on the deals offered last quarter, and the transactions that the customers purchased. They are in the *OfferInformation* and *Transactions* worksheet in the data Excel file.

There are 100 customers you are looking to segment, and you have 324 transactions between them.

The outline of the program document is as follows: first the file is loaded in to R, followed by the analysis performed with the dataset. Then we work to see the results from the kmeans, and finally we look into plotting the clusters along with silhouetting.

Loading the Excel files in R

In the first step let us load packages in to R. R, by default, comes with many of the standard packages. Packages are collection of functions in R that are stored in what is referred to a library. There are many other packages available that would need to be initially downloaded and installed in the machine through some simple commands. Once they are installed, they would need to be loaded into the session to use it. These packages increase the statistical capacity of R, and also is an active part of the open source programming community through CRAN.

For e.g., to install a package the following command is used:

```
install.packages("xlsx")
install.packages("rpart")
install.packages("tidyr")
install.packages("ggplot2")
install.packages("dplyr")
```

To load the installed packages in to the particular R session:

```
library(xlsx) #Loading the package needed to read Excel file
```

```
## Loading required package: rJava
## Loading required package: xlsxjars
```

```
library(rpart) #Loading the package needed to construct and prune trees
```

```
## Warning: package 'rpart' was built under R version 3.1.2
```

```
library(dplyr)#R package to quickly join datasets
```

```
## Warning: package 'dplyr' was built under R version 3.1.2
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.1.2
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
# Uses logic from http://rpubs.com/hrbrmstr/customer-segmentation-r
# Also refer https://rstudio-pubs-static.s3.amazonaws.com/33876_1d7794d9a86647ca90c4f182df93f0e8.html
```

If the packages are already installed, then load them in R using. Note, even if you have installed the package earlier, it would need to be loaded in order to use its functions.

Reading the Excel file

We read the Excel data that was provided into the data frame *lecture2data* . The variable name *DIRNAME* stores the directory, and *XLFILENAME* contains the file name in which the excel file is on my computer. The variable *FILENAME* stores the name of the absolute file name. We also store the number of clusters we will be using for our analysis. This can be changed depending on the analysis that would be performed.

```
DIRNAME <- "/Users/rvijayaraghavan/Desktop/rajesh/Personal/Teaching/DataDrivenSrikantClass/"
XLFILENAME <- "ClusteringWorkbook.xlsx"
FILENAME <- paste0(DIRNAME,XLFILENAME)
NUMCLUSTERS <- 4
```

Reading the *OfferInformation* sheet in to Excel. Then we assign the names of the columns in to what they refer to. This is to just simplify the conventions and make the naming more easy to refer in the future. So the command *colnames(lecture2data.offers)* is asking R to update the column names of the data frame read in the prior step.

```
# Read the offer information dataset.
lecture2data.offers <- read.xlsx(FILENAME, sheetName="OfferInformation")
colnames(lecture2data.offers) <- c("offerid", "campaign", "varietal",
                                   "minqty", "discount", "origin", "pastpeak")
```

Reading the *Transactions* sheet in to Excel. Notice the sheet itself has some extra text, so the read data contains other text that is not related to the dataset. These have to be removed and you can see this starting from third column in the data, running in to the fifth column. That would leave us with two columns for the *customername* and *offerid*. The code below performs these operation.

```
# Read the transaction information dataset.

lecture2data.transactions <- read.xlsx(FILENAME,
                                       sheetName="Transactions") ##reading the transactions worksheet.

lecture2data.transactions <- lecture2data.transactions[ -c(3:5) ] ##remove unwanted columns. In this cas

colnames(lecture2data.transactions) <- c("customername",
                                         "offerid") ##set column names to customername, and offerid.
```

At this point we have read in the two sheets, but have not joined them as yet.

The next step is creating the vector of 1 and 0 for the offer that was taken by the customer. This was created in Excel using pivot tables. We accomplish this using a number of steps. In the first step, we create a new column *isoffer* in the *Transactions* that takes a value 1. Then we merge the dataset offer and transaction information. We then set every other value to 0 for the offers that was not taken. That is, we first set the value 1, and then spread this value across other transactions so that other values are 0. At this point, we use the vectors of values that we just need for our analysis.

```
#initialize the variables

clustering.vector.inp <- NULL
clustering.vector <- NULL
clustering.vector.dataset <- NULL


# set the variable isoffer to 1. This is the first step in setting 1 for the offer corresponding to the
lecture2data.transactions$isoffer <- 1

# Now let us link the two sheets, and store this in the dataset called clustering.vector.dataset
clustering.vector.dataset <- dplyr::left_join(lecture2data.transactions, lecture2data.offers)
```

```
## Joining by: "offerid"
```

```
head(clustering.vector.dataset) ##look at the first five elements of the dataset.
```

```
##   customername offerid isoffer  campaign   varietal minqty discount
## 1        Smith       2       1   January Pinot Noir     72       17
## 2        Smith      24       1 September Pinot Noir      6       34
## 3      Johnson      17       1      July Pinot Noir     12       47
## 4      Johnson      24       1 September Pinot Noir      6       34
## 5      Johnson      26       1   October Pinot Noir    144       83
```

```
## 6     Williams      18       1      July  Espumante      6       50
##      origin pastpeak
## 1    France    FALSE
## 2     Italy    FALSE
## 3   Germany    FALSE
## 4     Italy    FALSE
## 5 Australia    FALSE
## 6    Oregon    FALSE
```

```
clustering.vector <- clustering.vector.dataset[,(1:3)] ##take the first three columns that are sufficien
head(clustering.vector)
```

```
##   customername offerid isoffer
## 1        Smith       2       1
## 2        Smith      24       1
## 3      Johnson      17       1
## 4      Johnson      24       1
## 5      Johnson      26       1
## 6     Williams      18       1
```

```
clustering.vector <-  tidyr::spread(clustering.vector,offerid,isoffer) ##spread the value of 1 across o
clustering.vector [is.na(clustering.vector )] <- 0

nrow(clustering.vector) #number of rows for input vector
```

```
## [1] 100
```

```
ncol(clustering.vector) #number of columns
```

```
## [1] 33
```

You will see there are 100 rows that correspond to the customers, and 32 offers in all.

Performing K Means

After creating the merged data set, we are ready to apply the kmeans algorithm in R. The command in R is *kmeans*. This function takes the argument of all columns that we would run the similarity measures on. The R *kmeans* command takes only numeric vectors – so in our case it would be just the 32 point (dimension) vector of 1s and 0s that correspond to the offers made by all the customers. These corresponds to cells $L2$ to $DG333$ in the Excel workbook in the *4MC* sheet that was done in class.

Similarly in our case, that would meanin the data set we created, the first column that identifies the index of the customer alone has to be removed. Note that we assigned *NUMCLUSTERS* to 4 earlier in our program.

The R command to generate kmeans is *kmeans()* . And *clusering.vector* has the input data, along with the index of the customer.

```
kmeans.result <- kmeans(clustering.vector.inp,NUMCLUSTERS)
```

The results from the kmeans are stored in the variable *kmeans.result* . The *kmeans* also returns other statistics, including the clusters that were generated for each of the customers. They are accessed specifying the command corresponding to the variable that we need such as cluster: *kmeans.result$cluster*, center using : *kmeans.result$centers* . For the full list refer to the output that is run by calling the output *kmeans.result*.

```r
set.seed(1234) ##set so that the centroids generated through kmeans dont flip after every run in one ses

clustering.vector.inp <- clustering.vector[,-1] #input for kmeans, all columns apart from the first-col

head(clustering.vector.inp)
```

```
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
## 2 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
## 3 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0
## 4 0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5 0 0 0 0 0 0 1 0 0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
## 6 0 0 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0
##   28 29 30 31 32
## 1  0  1  1  0  0
## 2  0  0  0  0  0
## 3  0  0  0  0  0
## 4  0  0  1  0  0
## 5  0  0  0  1  0
## 6  0  0  0  1  0
```

```r
kmeans.result <- kmeans(clustering.vector.inp,NUMCLUSTERS)

kmeans.result
```

```
## K-means clustering with 4 clusters of sizes 16, 33, 29, 22
##
## Cluster means:
##         1      2       3      4      5       6      7       8       9
## 1 0.06250 0.3750 0.00000 0.0000 0.0000 0.00000 0.0000 0.00000 0.00000
## 2 0.00000 0.0000 0.00000 0.0000 0.0000 0.03030 0.4242 0.48485 0.00000
## 3 0.24138 0.1379 0.03448 0.2414 0.1379 0.03448 0.0000 0.03448 0.27586
## 4 0.09091 0.0000 0.22727 0.2273 0.0000 0.45455 0.2273 0.13636 0.09091
##       10     11     12     13     14      15     16     17      18     19
## 1 0.0625 0.0000 0.0625 0.0000 0.0000 0.00000 0.0625 0.4375 0.00000 0.0000
## 2 0.0303 0.0000 0.0303 0.1818 0.0000 0.00000 0.0000 0.0000 0.39394 0.0000
## 3 0.0000 0.4483 0.1034 0.0000 0.2069 0.13793 0.1379 0.0000 0.00000 0.0000
## 4 0.2273 0.0000 0.0000 0.0000 0.1364 0.09091 0.0000 0.0000 0.04545 0.2273
##        20     21     22      23   24     25     26      27     28      29
## 1 0.00000 0.0000 0.0000 0.06250 0.75 0.0000 0.7500 0.06250 0.0000 0.00000
## 2 0.00000 0.0303 0.0000 0.00000 0.00 0.0000 0.0000 0.03030 0.0303 0.48485
## 3 0.17241 0.0000 0.3448 0.06897 0.00 0.2069 0.1034 0.06897 0.1724 0.03448
## 4 0.04545 0.1364 0.5000 0.09091 0.00 0.0000 0.0000 0.22727 0.0000 0.00000
##        30     31      32
## 1 0.00000 0.0000 0.00000
## 2 0.48485 0.0000 0.00000
## 3 0.17241 0.1034 0.06897
## 4 0.04545 0.6364 0.09091
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   2   3   1   2   4   4   1   2   3   2   3   1   2   3   3   1   3   1
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
```

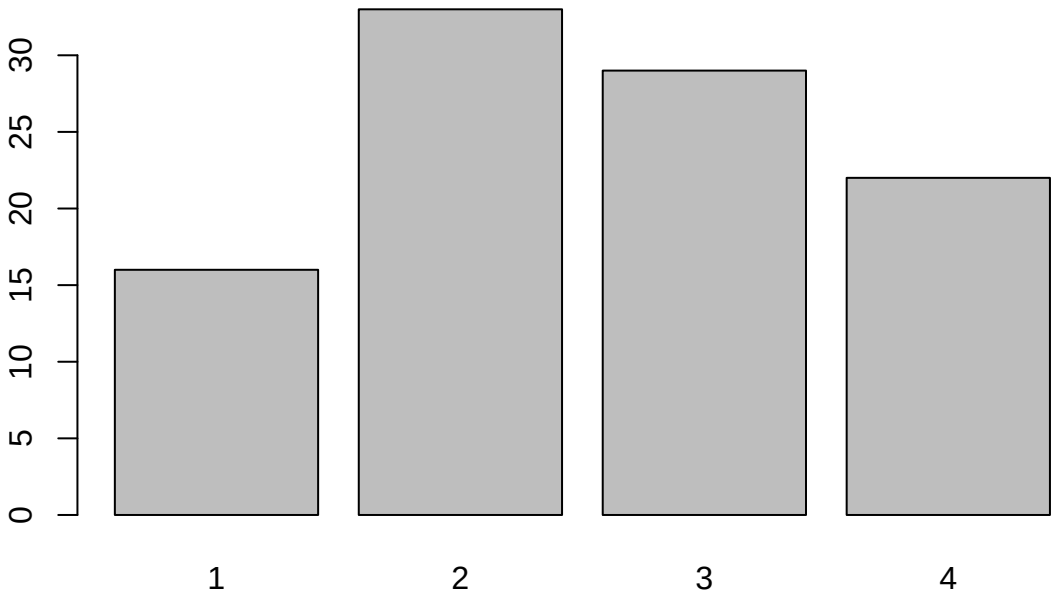
```
##   2   3   2   2   4   3   1   3   3   3   4   1   2   2   3   4   2   2
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   3   2   3   2   1   1   2   4   2   4   2   4   3   3   3   4   3   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   4   2   1   2   2   4   4   2   3   2   2   1   1   3   4   3   3   3
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   4   2   4   2   1   4   2   1   3   3   4   1   2   2   2   3   3   2
##  91  92  93  94  95  96  97  98  99 100
##   4   2   3   2   3   4   2   4   4   4
##
## Within cluster sum of squares by cluster:
## [1] 19.31 51.39 83.38 58.95
##  (between_SS / total_SS =  24.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Let us see a barplot to see how many customers are segmented to each of the clusters. We use the command *barplot* for the purpose. First, to show the statistics of the clusters that are assigned:

```
tab1 <- table(kmeans.result$cluster) #to show statistics of how the clusters are assigned across custom
tab1 # show the clusters
```

```
##
##  1  2  3  4
## 16 33 29 22
```

```
barplot(tab1) #to show a bar plot with the result from above.
```



Note that the original input that was passed as argument to the *kmeans* function was just the vector of 0s and 1s. For e.g.,

```
head(clustering.vector.inp)
```

```
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
## 2 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  1
## 3 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0
## 4 0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 5 0 0 0 0 0 0 1 0 0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
## 6 0 0 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0
##   28 29 30 31 32
## 1  0  1  1  0  0
## 2  0  0  0  0  0
## 3  0  0  0  0  0
## 4  0  0  1  0  0
## 5  0  0  0  1  0
## 6  0  0  0  1  0
```

To add more context, we need to merge the clusters to this and the corresponding customers. That will allow us to easily see how the customers look, and the connection between them.

```
# series of steps to join the results from the kmeans -- that is the cluster name to the corresponding


clustering.vector <- merge(clustering.vector,kmeans.result$cluster , by=0, all=TRUE)
names(clustering.vector)[names(clustering.vector) == 'y'] <- 'cluster'
clustering.vector <- left_join(clustering.vector,clustering.vector.dataset) #joining the two dataset us
```

```
## Joining by: "customername"
```

Let us now see the customers in each of the clusters. We use the command *filter* along with specification of the cluster name. So for example *filter(clustering.vector, cluster == 1)* returns all rows that have cluster value as 1, that is all customers who are in cluster 1. For cluster 1, we show the entire data set to give an understanding of easy to interpret how similar they are. Then we show the customer name alone for the corresponding cluster.

```
cluster.show <- filter(clustering.vector, cluster == 1)

cluster.show ##give the entire set of columns based on filtered by cluster 1. This shows all transactio
```

```
##    Row.names customername 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
## 1         12     Campbell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 2         12     Campbell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 3         12     Campbell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 4         16         Cook 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 5         16         Cook 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 6         18          Cox 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 7         18          Cox 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 8         18          Cox 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 9         18          Cox 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 10        25       Flores 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 11        25       Flores 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
```

```
## 12          3     Anderson 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 13          3     Anderson 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 14         30         Gray 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  1  0  0  0
## 15         30         Gray 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  1  0  0  0
## 16         30         Gray 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  1  0  0  0
## 17         41      Jenkins 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 18         41      Jenkins 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 19         42      Johnson 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 20         42      Johnson 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 21         42      Johnson 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 22         54        Moore 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 23         54        Moore 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 24         57       Morris 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 25         57       Morris 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 26         57       Morris 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 27         66     Peterson 1 1 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
## 28         66     Peterson 1 1 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
## 29         66     Peterson 1 1 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
## 30         66     Peterson 1 1 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
## 31         66     Peterson 1 1 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
## 32         66     Peterson 1 1 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0
## 33         67     Phillips 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 34         67     Phillips 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 35          7         Bell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 36          7         Bell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 37          7         Bell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 38          7         Bell 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  1  0  0
## 39         77    Rodriguez 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 40         77    Rodriguez 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 41         80      Russell 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 42         84        Smith 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
## 43         84        Smith 0 1 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0
##    20 21 22 23 24 25 26 27 28 29 30 31 32 cluster offerid isoffer
## 1   0  0  0  0  1  0  1  0  0  0  0  0  0       1       2       1
## 2   0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 3   0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 4   0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 5   0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 6   0  0  0  0  1  0  1  0  0  0  0  0  0       1       2       1
## 7   0  0  0  0  1  0  1  0  0  0  0  0  0       1      17       1
## 8   0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 9   0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 10  0  0  0  0  1  0  0  0  0  0  0  0  0       1      17       1
## 11  0  0  0  0  1  0  0  0  0  0  0  0  0       1      24       1
## 12  0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 13  0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 14  0  0  0  0  0  0  1  0  0  0  0  0  0       1      12       1
## 15  0  0  0  0  0  0  1  0  0  0  0  0  0       1      16       1
## 16  0  0  0  0  0  0  1  0  0  0  0  0  0       1      26       1
## 17  0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 18  0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 19  0  0  0  0  1  0  1  0  0  0  0  0  0       1      17       1
## 20  0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 21  0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
```

```
## 22  0  0  0  0  1  0  0  0  0  0  0  0  0       1      17       1
## 23  0  0  0  0  1  0  0  0  0  0  0  0  0       1      24       1
## 24  0  0  0  0  1  0  1  0  0  0  0  0  0       1      17       1
## 25  0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 26  0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 27  0  0  0  1  0  0  1  1  0  0  0  0  0       1       1       1
## 28  0  0  0  1  0  0  1  1  0  0  0  0  0       1       2       1
## 29  0  0  0  1  0  0  1  1  0  0  0  0  0       1      10       1
## 30  0  0  0  1  0  0  1  1  0  0  0  0  0       1      23       1
## 31  0  0  0  1  0  0  1  1  0  0  0  0  0       1      26       1
## 32  0  0  0  1  0  0  1  1  0  0  0  0  0       1      27       1
## 33  0  0  0  0  1  0  0  0  0  0  0  0  0       1      17       1
## 34  0  0  0  0  1  0  0  0  0  0  0  0  0       1      24       1
## 35  0  0  0  0  1  0  1  0  0  0  0  0  0       1       2       1
## 36  0  0  0  0  1  0  1  0  0  0  0  0  0       1      17       1
## 37  0  0  0  0  1  0  1  0  0  0  0  0  0       1      24       1
## 38  0  0  0  0  1  0  1  0  0  0  0  0  0       1      26       1
## 39  0  0  0  0  0  0  1  0  0  0  0  0  0       1       2       1
## 40  0  0  0  0  0  0  1  0  0  0  0  0  0       1      26       1
## 41  0  0  0  0  0  0  1  0  0  0  0  0  0       1      26       1
## 42  0  0  0  0  1  0  0  0  0  0  0  0  0       1       2       1
## 43  0  0  0  0  1  0  0  0  0  0  0  0  0       1      24       1
##     campaign   varietal minqty discount       origin pastpeak
## 1    January Pinot Noir     72       17       France    FALSE
## 2  September Pinot Noir      6       34        Italy    FALSE
## 3    October Pinot Noir    144       83    Australia    FALSE
## 4  September Pinot Noir      6       34        Italy    FALSE
## 5    October Pinot Noir    144       83    Australia    FALSE
## 6    January Pinot Noir     72       17       France    FALSE
## 7       July Pinot Noir     12       47      Germany    FALSE
## 8  September Pinot Noir      6       34        Italy    FALSE
## 9    October Pinot Noir    144       83    Australia    FALSE
## 10      July Pinot Noir     12       47      Germany    FALSE
## 11 September Pinot Noir      6       34        Italy    FALSE
## 12 September Pinot Noir      6       34        Italy    FALSE
## 13   October Pinot Noir    144       83    Australia    FALSE
## 14       May   Prosecco     72       83    Australia    FALSE
## 15      June     Merlot     72       88   California    FALSE
## 16   October Pinot Noir    144       83    Australia    FALSE
## 17 September Pinot Noir      6       34        Italy    FALSE
## 18   October Pinot Noir    144       83    Australia    FALSE
## 19      July Pinot Noir     12       47      Germany    FALSE
## 20 September Pinot Noir      6       34        Italy    FALSE
## 21   October Pinot Noir    144       83    Australia    FALSE
## 22      July Pinot Noir     12       47      Germany    FALSE
## 23 September Pinot Noir      6       34        Italy    FALSE
## 24      July Pinot Noir     12       47      Germany    FALSE
## 25 September Pinot Noir      6       34        Italy    FALSE
## 26   October Pinot Noir    144       83    Australia    FALSE
## 27   January     Malbec     72       56       France    FALSE
## 28   January Pinot Noir     72       17       France    FALSE
## 29     April   Prosecco     72       52   California    FALSE
## 30 September Chardonnay    144       39 South Africa    FALSE
## 31   October Pinot Noir    144       83    Australia    FALSE
```

```
## 32   October  Champagne     72       88  New Zealand    FALSE
## 33      July Pinot Noir     12       47      Germany    FALSE
## 34 September Pinot Noir      6       34        Italy    FALSE
## 35   January Pinot Noir     72       17       France    FALSE
## 36      July Pinot Noir     12       47      Germany    FALSE
## 37 September Pinot Noir      6       34        Italy    FALSE
## 38   October Pinot Noir    144       83    Australia    FALSE
## 39   January Pinot Noir     72       17       France    FALSE
## 40   October Pinot Noir    144       83    Australia    FALSE
## 41   October Pinot Noir    144       83    Australia    FALSE
## 42   January Pinot Noir     72       17       France    FALSE
## 43 September Pinot Noir      6       34        Italy    FALSE
```

```
cluster.show$customername ##gives just the customer name from the above.
```

```
##  [1] Campbell  Campbell  Campbell  Cook      Cook      Cox       Cox
##  [8] Cox       Cox       Flores    Flores    Anderson  Anderson  Gray
## [15] Gray      Gray      Jenkins   Jenkins   Johnson   Johnson   Johnson
## [22] Moore     Moore     Morris    Morris    Morris    Peterson  Peterson
## [29] Peterson  Peterson  Peterson  Peterson  Phillips  Phillips  Bell
## [36] Bell      Bell      Bell      Rodriguez Rodriguez Russell   Smith
## [43] Smith
## 100 Levels: Adams Allen Anderson Bailey Baker Barnes Bell ... Young
```

```
## Similarly repeat for customer 2
cluster.show <- filter(clustering.vector, cluster == 2)
cluster.show$customername
```

```
##  [1] Adams     Adams     Adams     Brown     Brown     Brown     Carter
##  [8] Carter    Carter    Carter    Cruz      Cruz      Diaz      Diaz
## [15] Diaz      Diaz      Edwards   Edwards   Green     Gutierrez Gutierrez
## [22] Gutierrez Gutierrez Hernandez Hernandez Hill      Hill      Hill
## [29] Hill      Hughes    Hughes    Hughes    Hughes    Hughes    Bailey
## [36] Bailey    James     James     James     James     James     Jones
## [43] King      King      King      King      Lewis     Lewis     Lewis
## [50] Morgan    Morgan    Murphy    Murphy    Myers     Myers     Ortiz
## [57] Perez     Perez     Perry     Perry     Perry     Perry     Rivera
## [64] Rivera    Robinson  Robinson  Ross      Ross      Bennett   Bennett
## [71] Stewart   Stewart   Stewart   Sullivan  Sullivan  Sullivan  Taylor
## [78] Taylor    Taylor    Taylor    Torres    Walker    Walker    Watson
## [85] Watson    Wilson    Wilson
## 100 Levels: Adams Allen Anderson Bailey Baker Barnes Bell ... Young
```

```
## Similar for customer 3
cluster.show <- filter(clustering.vector, cluster == 3)
cluster.show$customername
```

```
##   [1] Butler   Butler   Butler   Butler   Butler   Clark    Clark
##   [8] Clark    Clark    Collins  Collins  Cooper   Cooper   Cooper
##  [15] Cooper   Allen    Allen    Davis    Davis    Davis    Fisher
##  [22] Fisher   Fisher   Fisher   Fisher   Fisher   Fisher   Foster
##  [29] Foster   Foster   Foster   Foster   Garcia   Garcia   Gomez
```

```
##  [36] Gomez    Gomez    Gomez    Hall     Hall     Howard   Howard
##  [43] Howard   Jackson  Jackson  Jackson  Jackson  Jackson  Lopez
##  [50] Lopez    Lopez    Lopez    Lopez    Martin   Martin   Martin
##  [57] Martinez Martinez Martinez Mitchell Mitchell Parker   Parker
##  [64] Parker   Parker   Parker   Powell   Ramirez  Reed     Reed
##  [71] Reyes    Reyes    Sanchez  Sanchez  Sanchez  Sanchez  Sanchez
##  [78] Sanchez  Sanchez  Sanders  Sanders  Sanders  Sanders  Sanders
##  [85] Sanders  Sanders  Sanders  Sanders  Thomas   Thomas   Thomas
##  [92] Thomas   Thomas   Thomas   Thompson Thompson Thompson Thompson
##  [99] Brooks   Brooks   Brooks   Brooks   Ward     White    White
## [106] White    White
## 100 Levels: Adams Allen Anderson Bailey Baker Barnes Bell ... Young
```

```
## Similar for customer 4
cluster.show <- filter(clustering.vector, cluster == 4)
cluster.show$customername
```

```
##  [1] Young      Young      Young      Young      Young      Young
##  [7] Evans      Evans      Gonzalez   Gonzalez   Harris     Harris
## [13] Harris     Harris     Harris     Harris     Kelly      Kelly
## [19] Kelly      Kelly      Lee        Lee        Lee        Lee
## [25] Lee        Long       Long       Long       Baker      Baker
## [31] Baker      Baker      Miller     Miller     Miller     Miller
## [37] Miller     Miller     Miller     Morales    Morales    Morales
## [43] Morales    Morales    Morales    Barnes     Barnes     Barnes
## [49] Barnes     Nelson     Nelson     Nelson     Nelson     Nguyen
## [55] Nguyen     Price      Price      Price      Price      Richardson
## [61] Richardson Richardson Roberts    Rogers     Rogers     Rogers
## [67] Rogers     Rogers     Rogers     Scott      Scott      Scott
## [73] Turner     Turner     Turner     Turner     Williams   Williams
## [79] Williams   Wood       Wood       Wood       Wood       Wright
## [85] Wright     Wright     Wright
## 100 Levels: Adams Allen Anderson Bailey Baker Barnes Bell ... Young
```

The above commands give us a good sense of the customers who were in different clusters, thus helping the business decisions. It gives a simple way to understand the features that make them similar.

Within cluster estimates

Recall that we also calculated within cluster sum of squares to get an understanding of their fit. *kmeans* also returns variable *withinss* with the corresponding values. That is to get the vector of with-in cluster of the sum of squares, use the *$withinss* value that is returned. This returns sum of squares as one component per cluster. This is accessed using the following command.

```
kmeans.result$withinss
```

```
## [1] 19.31 51.39 83.38 58.95
```

Advanced – Graphics and Silhouette

In the following section, we will discuss some graphics option as to how would one would plot the output from the clusters. These commands are somewhat advanced, but give a sense of the options available in R.

The key to note is that amount of code that is needed to get the output. Usually it is few lines that give you robust set of results while prototyping.

Plotting output

First let us install the packages that we would need for graphics options. We illustrate two different methods of plotting the output, and include two packages that are specifically used for this purpose. If you dont have the packages already installed, run the following two lines of code to install and then load them. If you are running the .Rmd file, replace *{r eval = FALSE}* with *{r}* .

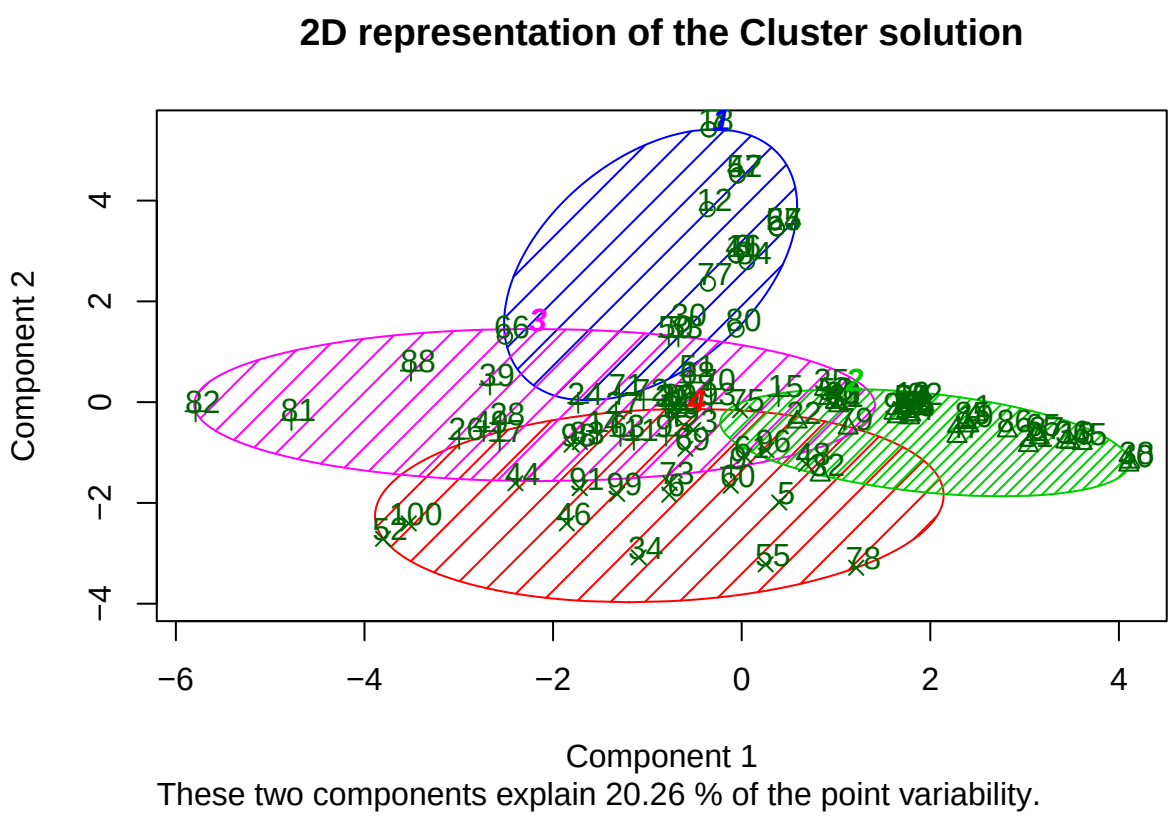
```
install.packages('cluster')
devtools::install_github("sinhrks/ggfortify")
```

Using the *cluster* package.

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.1.2
```

```
clusplot(clustering.vector.inp, kmeans.result$cluster, main='2D representation of the Cluster solution'
```

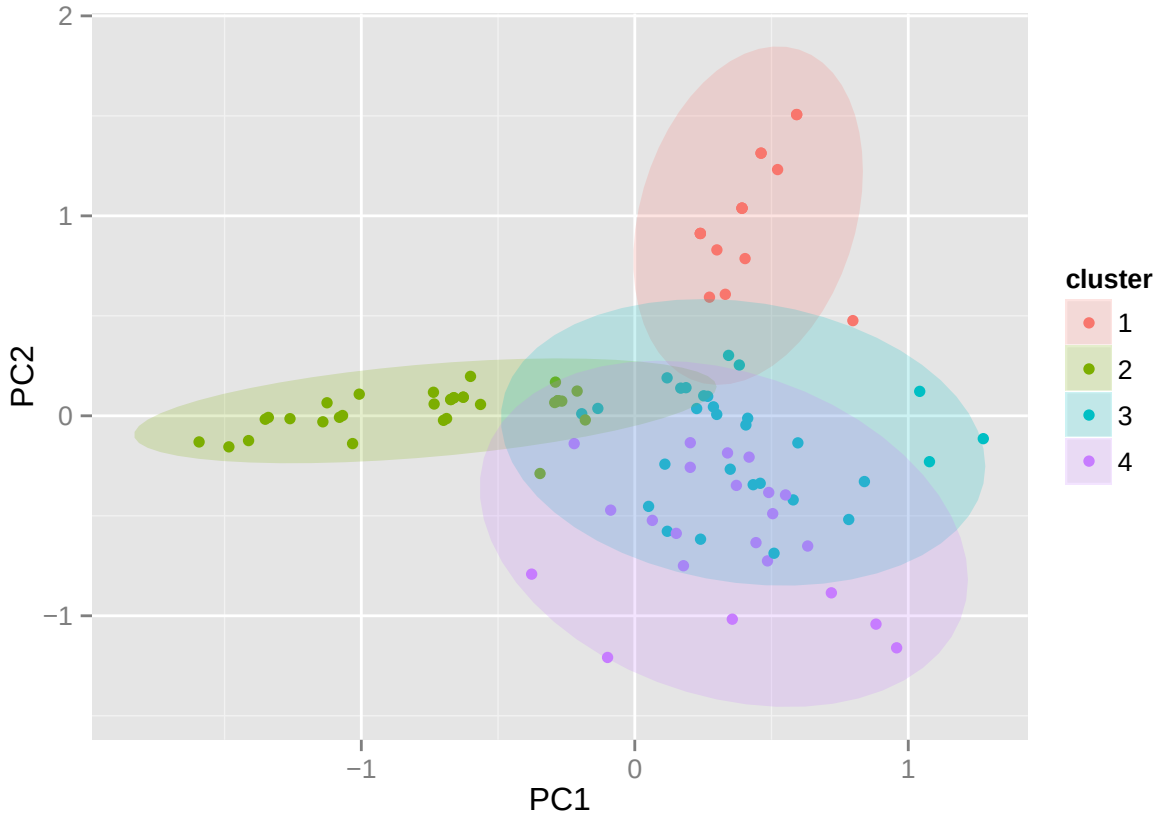


Using the *ggfortify* package.

```
library(ggfortify)
```

```
## Loading required package: proto
```

```
autoplot(kmeans.result, data=clustering.vector.inp, frame=TRUE, frame.type='norm')
```



Silhouetting

In the next step, we run the *kmeans* algorithm multiple times and see how the silhouette value looks like. This will help us to give an idea of the K that we would pick. We repeat this for all values from 2 to 10. That is we see the siluoette values for each cluster between 2 to 10.

We run the program in the form of a loop using the *for* statement in R. The following statement illustrates a loop. In this case, the program runs the code between the pair of { and }, for each value of i from 2 to MAX.

```
for (i in 2:MAX) ##run the loop from 2 to MAX. i value gets incremented in each iteration.
```

The idea in our case is that we run *kmeans* algorithm for number of clusters = 2, calculate the silouette value. Then we run the program for number of cluster = 3, calculate the silhouette value, and we keep repeating this for all values until MAX. In our case, we set MAX to 10 but that can be changed to any value. In that way, one can evaluate what would be a good cluster to pick from between 2 and 10 depending on the silhouette value.

```
MAX <- 10 #Maximum number of clusters to try
sil <- NULL

for (i in 2:MAX) #Loop runs for all value of i between 2 and MAX.
{
      temp.kmeans <- kmeans(clustering.vector.inp,
                            centers=i) #Performs K-means clustering for the corresponding value of i.
```

```
      temp.clusters <- temp.kmeans$cluster #get the clusters from the output.

      temp.silval <- silhouette(temp.clusters,
                                dist(clustering.vector.inp)) # calculate the silhouette value and store

      sil[i] <- mean(temp.silval[,3]) #store the mean silhouette value that is used for plotting the va
}
```

Plotting the silhouette values

Finally, plot the silhouette values that was created in the previous step:

```
plot(1:MAX, sil)
lines(1:MAX, sil)
```