Week 14 Advanced R Data Mining

```
knitr::opts_knit$set(root.dir = "D:/Introduction-to-R/Data")
library("dplyr")      # Data manipulation
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("tidyr")      # Tidying the data
library("validate")   # Data validation
```

```
##
## Attaching package: 'validate'
```

```
## The following object is masked from 'package:dplyr':
##
##     expr
```

```
library("magrittr")   # To use for %>% (pipes)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library("party")      # Creating the decision tree
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
library("DMwR")       # Data Mining with R package
```

```
## Loading required package: lattice
```

There are many topics that would fall into advanced uses of R. This week is dedicated to the creation of an R application to do data mining. Students will explore data mining in detail in other courses, but this will give them hands on experience with the R code to do it. This is valuable for use as a data mining exercise, but it also tests the student on just about every aspect of the language both from a statistical perspective as well as a data manipulation. It serves as an excellent final test for the student.

In this week's assignment, we will walk through an entire data mining project. It will include both supervised and unsupervised examples, with the final output being a clustered dataset.

Weekly Learning Objectives

Create an R script focused on data mining with the following components:

1. Definition of the sample file and some initial statistical information.
2. The creation of a decision tree to further analyze the data.
3. Outlier detection
4. K-means clustering of the data
5. Output of the results to a text file, or separate text files for each cluster

1. Initial statistical information

Any data mining project should start with exploratory data analysis (EDA). While there could be a lot of additional steps in this section, for the purposes of this exercise we will simply load the data file and do some initial stats on it. We will also run it through validation to ensure that our data quality checks are met. In a real-world example, this would have numerous additional steps.

This will give us a general idea of the dataset, and help to determine what factors are of greatest interest.

Full Points

Load the data and perform some basic stats on it. That can be done manually, or through a combination of plots and even the use of the validation library from earlier exercises.

```
phoneRawData <- read.csv("Phone Records.csv")

# Describe the data - Summary
head(phoneRawData)
```

```
##        ID Fixed.Calls  Duration     Mobile International Broadband
## 1 C100000   229.37271 26822.505   0.636458     16.007303   0.00000
## 2 C100100   120.84861  7282.827  14.464970      7.593508   0.00000
## 3 C100200    97.45483  4385.387   4.307254     13.866595   0.00000
## 4 C100300   510.09463 50479.551 500.780837     34.524157 298.86990
## 5 C100400   401.63257 46200.675  30.025406     13.371935  69.11522
## 6 C100500   159.38233  9796.170  33.726267     12.616191   0.00000
```

```
tail(phoneRawData)
```

```
##           ID Fixed.Calls  Duration     Mobile International Broadband
## 2052 C994000   292.57861 27443.171 233.754365     67.337511  44.99276
## 2053 C995000   143.03991  9992.605   5.850008      8.204185   0.00000
## 2054 C996000   261.18645 10285.038 202.332191    226.525369   0.00000
## 2055 C997000   153.38252 17505.765  18.491076      6.583973   0.00000
## 2056 C998000   310.68905 36938.817  83.878480     67.677106   0.00000
## 2057 C999000    95.16215  7452.760  10.375127     11.433856   0.00000
```

```
summary(phoneRawData)
```

```
##        ID        Fixed.Calls         Duration          Mobile
##  C100000:   1   Min.   :   0.00   Min.   :     0   Min.   :   0.000
##  C100100:   1   1st Qu.:  80.63   1st Qu.:  5302   1st Qu.:   4.655
##  C100200:   1   Median : 133.34   Median :  9518   Median :  21.133
##  C100300:   1   Mean   : 164.76   Mean   : 13373   Mean   :  62.440
##  C100400:   1   3rd Qu.: 211.19   3rd Qu.: 16620   3rd Qu.:  72.654
##  C100500:   1   Max.   :1119.94   Max.   :198954   Max.   :1895.452
##  (Other):2051
##  International       Broadband
##  Min.   :  0.000   Min.   :    0.00
##  1st Qu.:  5.943   1st Qu.:    0.00
##  Median : 10.810   Median :    0.00
##  Mean   : 19.724   Mean   :  181.30
##  3rd Qu.: 21.327   3rd Qu.:   39.48
##  Max.   :466.076   Max.   :16130.86
##
```

```
var(phoneRawData$Fixed.Calls)
```

```
## [1] 16013.31
```

```
var(phoneRawData$Duration)
```

```
## [1] 194869186
```

```
var(phoneRawData$Mobile)
```

```
## [1] 12374.55
```

```
var(phoneRawData$International)
```

```
## [1] 1053.075
```

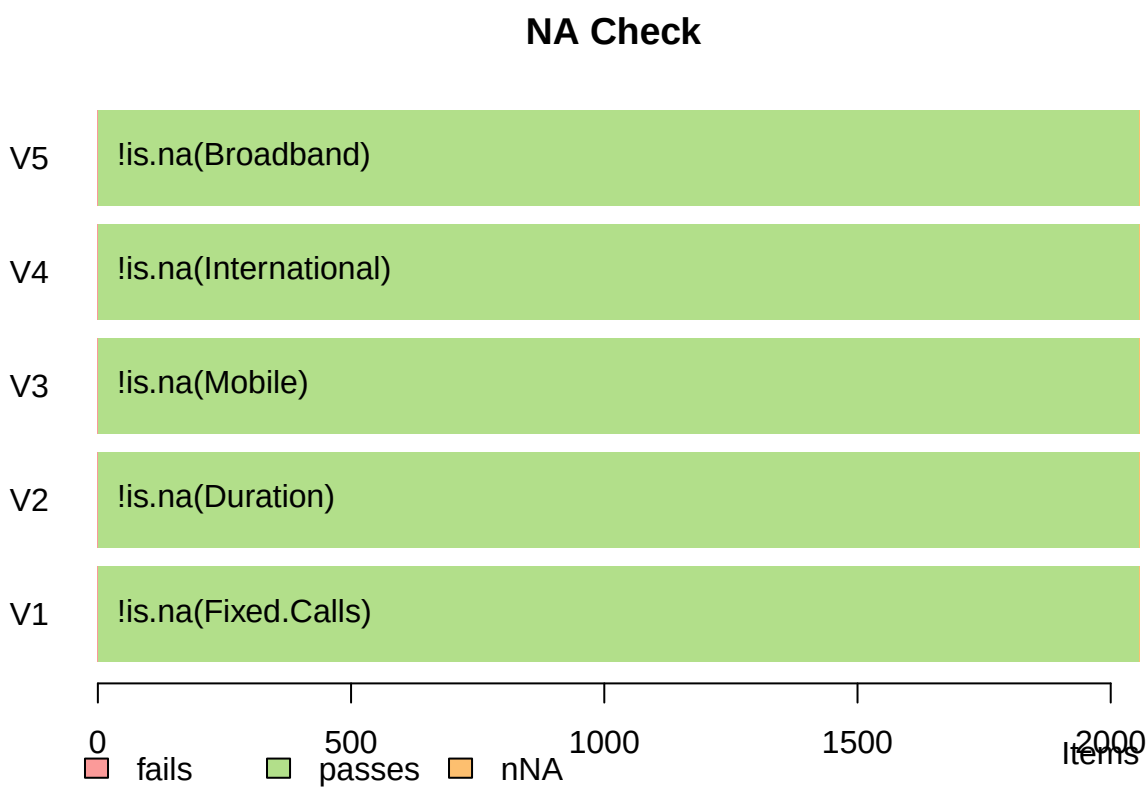
```
var(phoneRawData$Broadband)
```

```
## [1] 497895.1
```

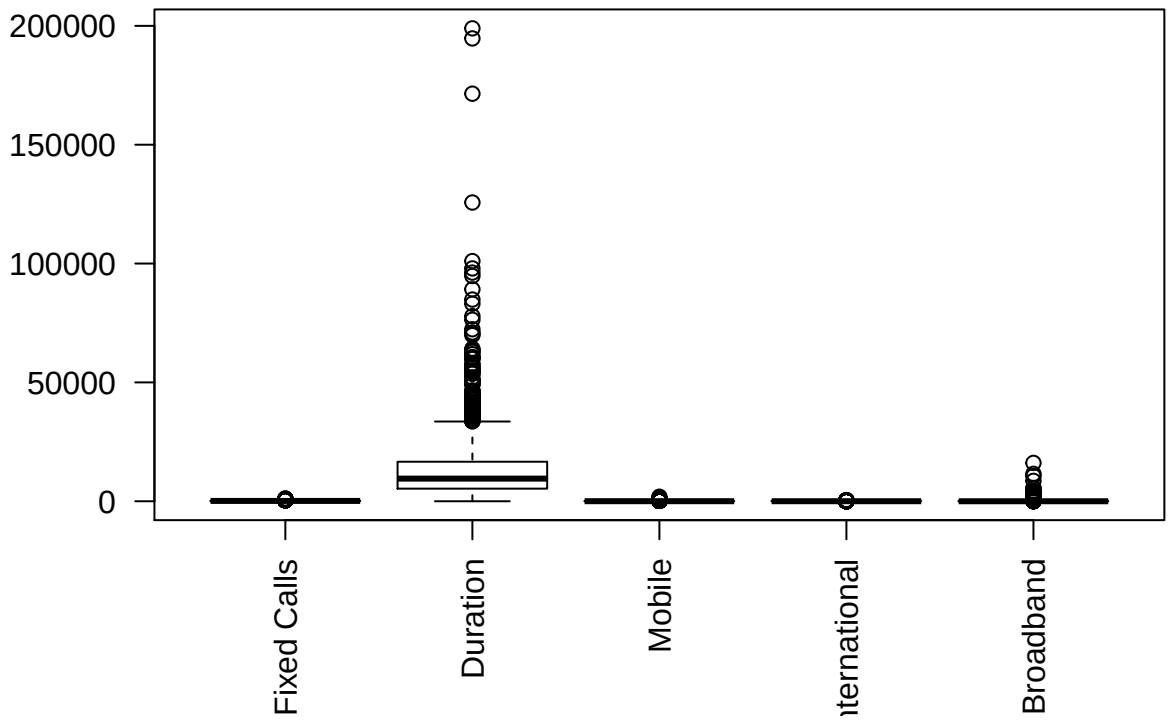
```
# Check for invalid fields
aCheck <- check_that(phoneRawData,
                     !is.na(Fixed.Calls),
                     !is.na(Duration),
                     !is.na(Mobile),
                     !is.na(International),
                     !is.na(Broadband))
print(summary(aCheck))
```

```
##   name items passes fails nNA error warning            expression
## 1   V1  2057   2057     0   0 FALSE   FALSE   !is.na(Fixed.Calls)
## 2   V2  2057   2057     0   0 FALSE   FALSE      !is.na(Duration)
## 3   V3  2057   2057     0   0 FALSE   FALSE        !is.na(Mobile)
## 4   V4  2057   2057     0   0 FALSE   FALSE !is.na(International)
## 5   V5  2057   2057     0   0 FALSE   FALSE     !is.na(Broadband)
```

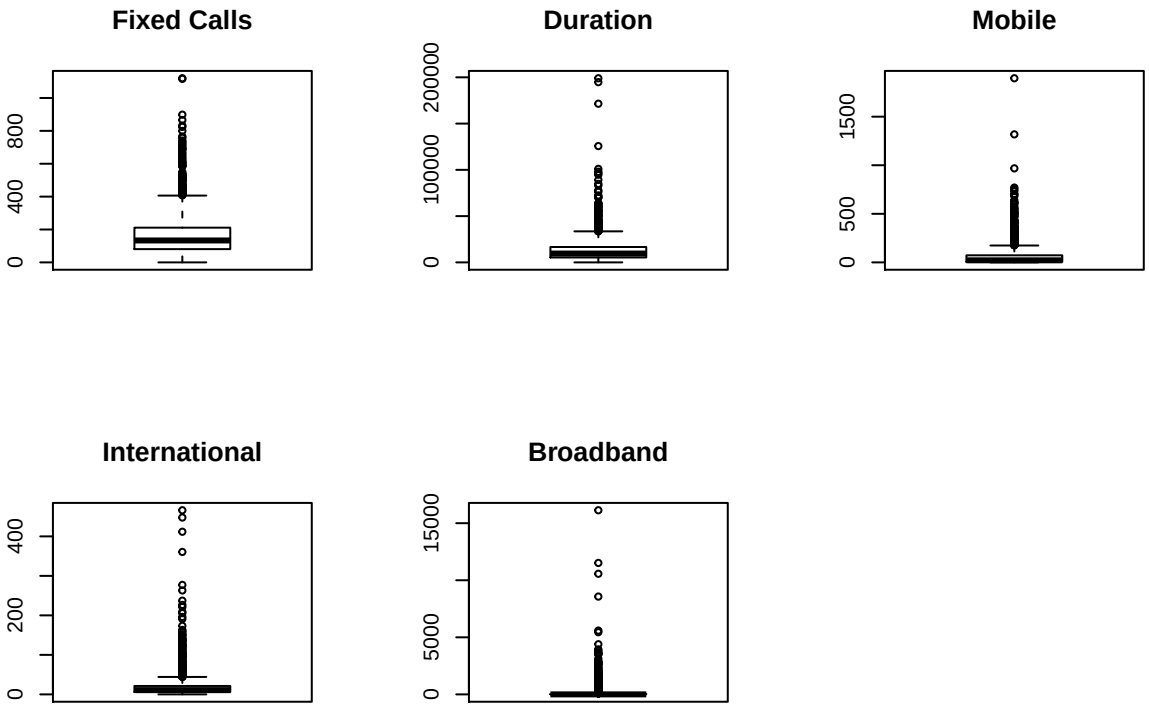
```
# Visually describe the data
barplot(aCheck, main="NA Check")
```



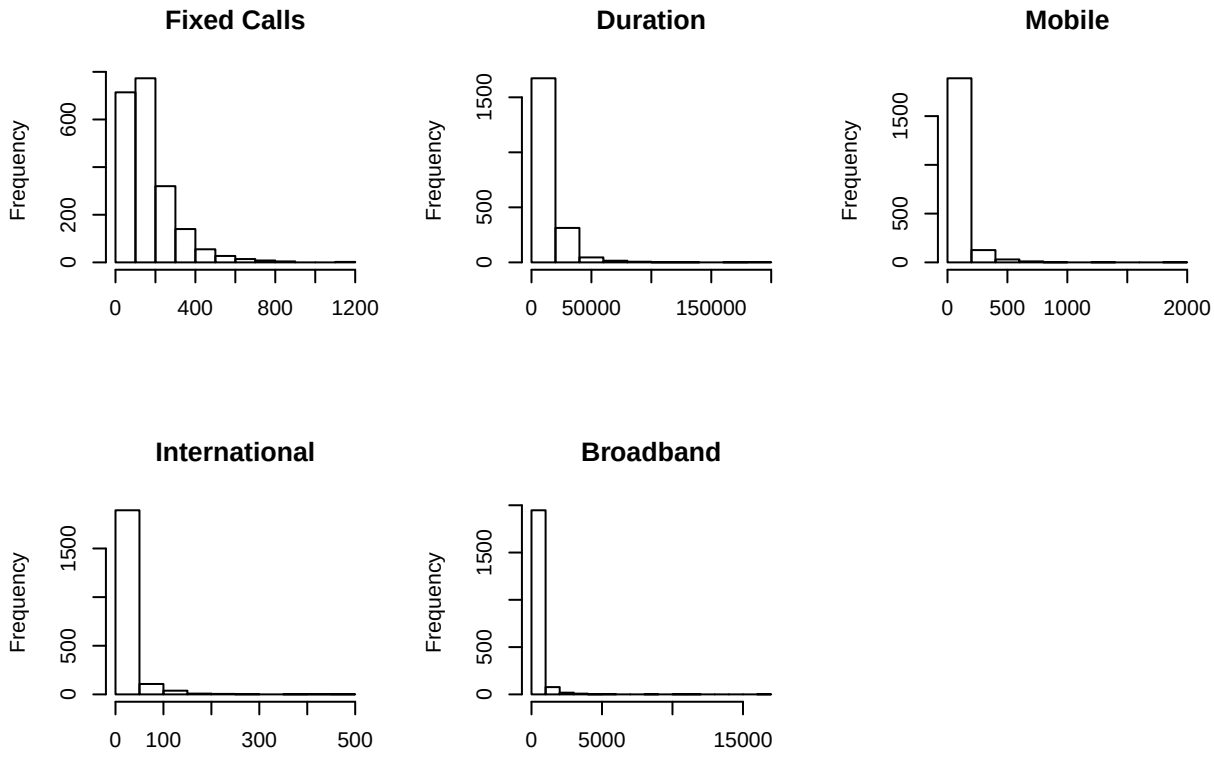

```
boxplot(phoneRawData$Fixed.Calls,
        phoneRawData$Duration,
        phoneRawData$Mobile,
        phoneRawData$International,
        phoneRawData$Broadband,
        las = 2,
        main = "Overall Comparison",
        names=c("Fixed Calls","Duration","Mobile","International","Broadband"))
```



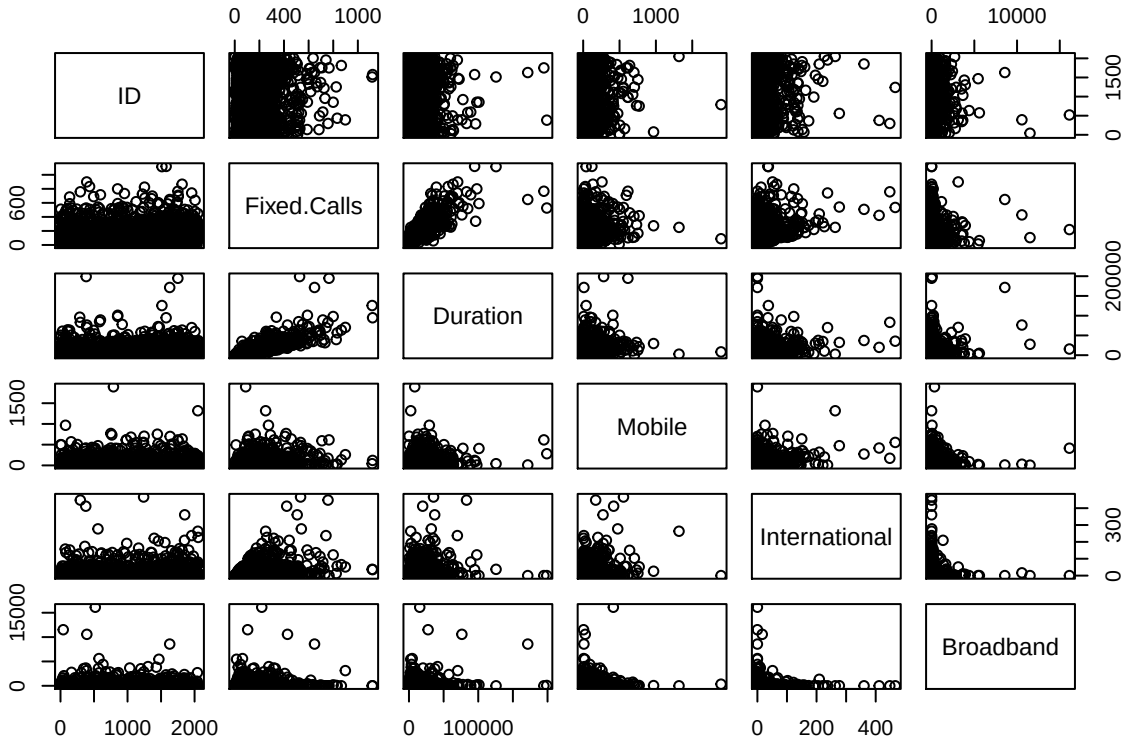

```
# Individual Boxplots
par(mfrow=c(2,3))
boxplot(phoneRawData$Fixed.Calls,main="Fixed Calls")
boxplot(phoneRawData$Duration,main="Duration")
boxplot(phoneRawData$Mobile,main="Mobile")
boxplot(phoneRawData$International,main="International")
boxplot(phoneRawData$Broadband,main="Broadband")
par(mfrow=c(1,1))   #Reset graph window
```




```
# Histograms
par(mfrow=c(2,3))
hist(phoneRawData$Fixed.Calls,main="Fixed Calls",xlab="")
hist(phoneRawData$Duration,main="Duration",xlab="")
hist(phoneRawData$Mobile,main="Mobile",xlab="")
hist(phoneRawData$International,main="International",xlab="")
hist(phoneRawData$Broadband,main="Broadband",xlab="")
par(mfrow=c(1,1))   #Reset graph window
```




```
plot(phoneRawData)
```



2. Create a decision tree

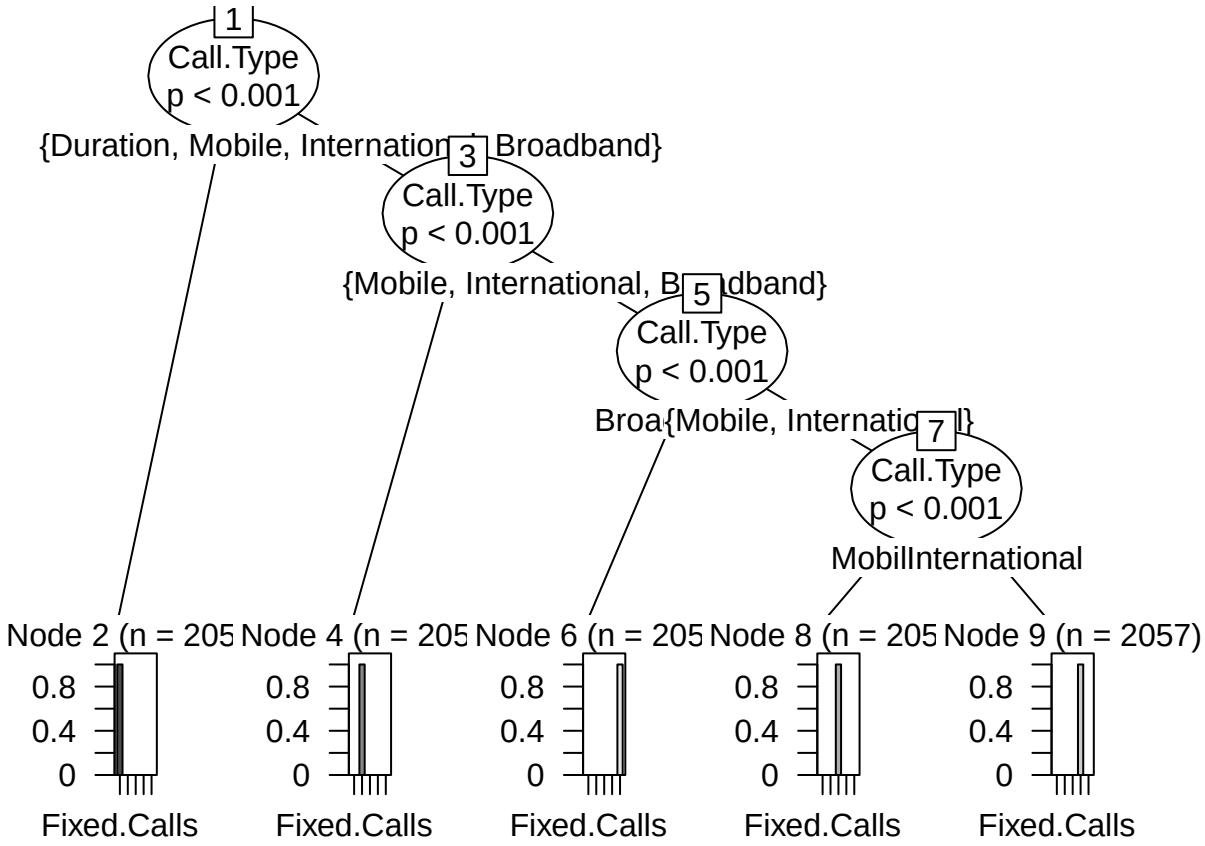
We have created decision trees earlier in the course using more of a manual approach. Now we will shift to using one of the package libraries. For this, we want to run our dataset through and build out a decision tree. This example will demonstrate the minimum necessary to produce a tree. There are a lot of additional steps that can be introduced to make it even better. A good discussion of creating decision trees in R can be found here:

https://www.datacamp.com/community/tutorials/decision-trees-R

Full Points

Generate a simple decision tree, either through a manual process as before or by using a library call such as show below.

```
phoneDataPivot <- gather(phoneRawData,key=Call.Type,value="Length",Fixed.Calls,Duration,Mobile,Internati

phoneTree <- ctree(Call.Type ~ Call.Type, data=phoneDataPivot)
plot(phoneTree, las=2)
```




```
rm(phoneDataPivot)
rm(phoneTree)
```

3. Outlier detection

Understanding outliers is crucial to a properly tuned model. There may be instances when you would include or exclude an outlier. This is another spot where a package can make it much easier. A detailed discussion on outlier detection can be found here:

http://www.rdatamining.com/examples/outlier-detection

Full Points

Include outlier detection in your script, either through a manual process or by using a package. In this example, we will use a package.

```
phoneData <- select(phoneRawData,Fixed.Calls,Duration,Mobile,International,Broadband)

# Individual variable outliers

# Fixed Calls
boxplot.stats(phoneData$Fixed.Calls)$out
```

```
##   [1]  510.0946  431.4121  410.1768  499.9146  415.4666  520.3207  442.5105
```

```
##   [8]  684.2165  588.0767  529.1635  493.0272  452.3593  512.0886  457.2598
##  [15]  757.7755  449.2006  512.1846  421.9935  526.6966  898.5445  428.5223
##  [22]  451.5816  832.5361  508.0625  688.8205  504.0934  540.9044  465.5237
##  [29]  519.0864  715.3293  443.3229  454.3862  417.1348  517.1402  439.4904
##  [36]  498.1559  513.3699  474.8487  799.6126  590.3857  531.3966  502.6369
##  [43]  550.1302  461.7152  523.3602  733.2407  535.7358  494.1699  508.3537
##  [50]  617.1443  469.0108  477.0828  725.3228  534.6829  824.3238  669.1801
##  [57]  601.2617  660.0257  697.4533  432.6316  458.3404  410.1662  456.7711
##  [64]  585.4159  604.7096  588.8775  420.8351 1116.2694  514.0053  416.3251
##  [71] 1119.9429  422.2949  427.2394  408.7459  585.4545  477.6980  648.0569
##  [78]  628.6925  409.2615  444.0672  447.6050  515.0649  494.4531  441.3599
##  [85]  475.9934  604.2485  612.5696  765.7142  707.9607  692.3592  460.7310
##  [92]  865.9984  458.7477  442.7013  507.3724  442.7802  497.6811  741.0780
##  [99]  423.5197  426.4231  478.5957  636.6896  448.4523
```

```
# Duration
boxplot.stats(phoneData$Duration)$out
```

```
##   [1]  50479.55  46200.67  60718.99  34132.81  43248.49  35025.92  49027.88
##   [8]  43989.45  64191.71  60253.01  33957.17  63321.89  39559.10  50807.99
##  [15]  44965.97  37005.42  34905.39  96195.93  83107.89  39570.39  33664.55
##  [22] 198954.01  70353.16  76358.83  42747.04  38567.27  56536.61  41764.95
##  [29]  41140.36  62811.25  53754.43  45198.15  36419.16  84853.08  89158.59
##  [36]  36218.76  34550.92  35233.92  34457.06  50183.21  34665.98  50954.22
##  [43]  34426.53  97948.54 101031.34  41138.24  35692.41  36453.11  34437.60
##  [50]  77869.28  37086.57  44537.15  46503.85  38350.07  52100.29  34205.75
##  [57]  37210.84  35713.58  37187.17  57598.18  39405.44  35459.29  46553.84
##  [64]  39550.19  60469.99  37262.20  41219.35  33650.56  42961.79  36328.47
##  [71]  38527.36  36726.01  57858.13  50955.88  56360.07  37488.96  40176.76
##  [78]  34671.22  41304.39  35017.10  46076.09  49153.86  72355.02  37480.90
##  [85]  34214.48  71016.73 125687.44  39233.36  44951.05  37727.17  39986.21
##  [92]  94767.23  36460.10  53642.49  55073.95 171457.03  49846.85  33845.94
##  [99]  34378.68  34741.75  43008.34  37240.32  42017.17  61868.45  54893.97
## [106]  37601.18  44268.05  54733.72  56128.22  39080.80  36626.15 194720.68
## [113]  60098.33  39417.27  36394.39  63569.62  37146.35  41202.84  54295.79
## [120]  44475.75  44141.51  36251.99  35898.16  35243.55  35087.08  69860.74
## [127]  35307.92  35819.60  33992.08  59823.01  34697.10  34391.87  36938.82
```

```
# Mobile
boxplot.stats(phoneData$Mobile)$out
```

```
##   [1]  500.7808  495.1408  262.5925  178.5279  176.3018  275.5282  304.5601
##   [8]  215.7571  967.9738  194.5141  179.1284  358.1257  208.7966  210.8852
##  [15]  193.4156  354.7940  248.1233  569.6558  214.7173  208.1975  373.2367
##  [22]  254.0768  277.6692  430.6860  182.6844  248.7305  504.1272  240.7314
##  [29]  188.4667  264.3361  197.8488  249.0039  401.5321  211.1898  258.5801
##  [36]  338.3639  418.8391  232.3999  281.4274  190.1719  258.8800  216.4396
##  [43]  247.3953  283.3452  341.1602  181.7156  433.8474  199.3745  227.3791
##  [50]  213.6957  279.0880  188.3380  417.0223  238.0998  366.2269  414.8489
##  [57]  204.2879  301.7655  475.0654  182.8653  181.2174  204.8637  200.8830
##  [64]  214.0601  205.9551  258.9386  315.1326  369.9619  291.8397  180.7537
##  [71]  240.0194  284.8148  276.8272  197.1460  367.1938  178.9018  769.8512
##  [78]  418.9714  316.8730  213.0565  734.1001  240.8658  313.1207  333.7871
```

```
##  [85]  297.4840 1895.4521  190.7339  221.7066  200.9393  203.1727  412.1926
##  [92]  222.8579  252.9479  407.6104  201.9233  182.3208  359.3965  225.9057
##  [99]  214.6877  411.4214  190.8865  613.6871  615.2694  274.2249  283.9705
## [106]  243.5084  257.9496  196.1149  260.4415  219.0225  199.8222  681.6561
## [113]  215.5568  524.4022  231.1335  412.9995  215.0129  224.4124  487.7329
## [120]  346.4502  349.1918  180.9333  549.6040  211.2606  189.2621  195.0781
## [127]  178.8136  242.9765  310.2514  222.6470  192.6925  193.8965  214.7511
## [134]  316.7680  548.8689  554.0751  194.5018  256.4255  704.0194  196.6506
## [141]  248.0296  413.4198  178.7452  187.9188  179.5003  247.5009  241.9742
## [148]  303.5140  271.1078  280.0179  221.0954  192.0734  377.3726  203.5249
## [155]  320.7104  268.9081  752.7183  525.4477  476.1832  228.9348  426.5762
## [162]  222.3288  430.7294  208.4878  361.9385  239.2908  303.8369  207.3935
## [169]  697.9616  394.7706  509.5626  263.9655  183.9951  517.2174  219.6122
## [176]  222.5008  615.3138  593.7251  289.3871  267.0892  277.7181  271.4794
## [183]  183.8504  233.5746  638.5853  349.6143  461.9855  315.3434  318.9656
## [190]  273.0260  198.4020  379.3172  348.9656  192.9847  194.1032  301.3737
## [197]  214.8342  234.7029  284.8975  177.7664  195.6876  221.1400  260.1706
## [204]  183.7712  234.6136  288.4186  278.9933  196.7130  181.1925  197.0675
## [211]  210.1537 1317.8586  233.7544  202.3322
```

```
# International
boxplot.stats(phoneData$International)$out
```

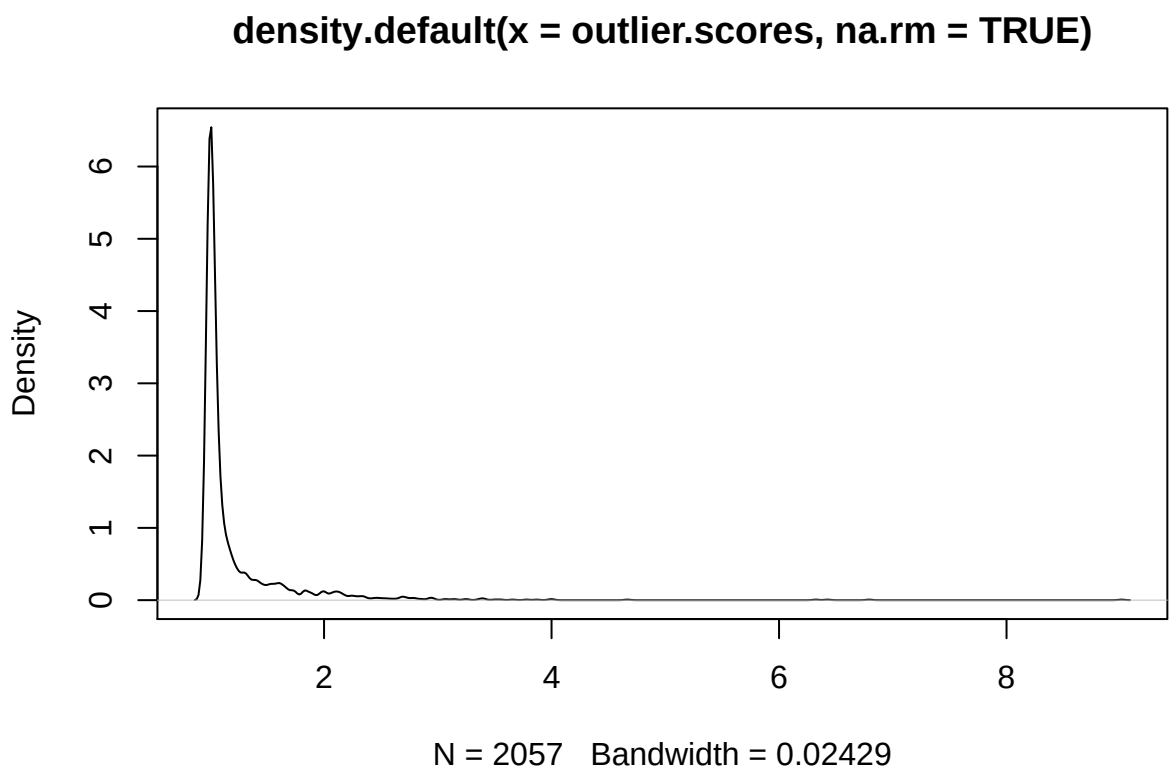
```
##   [1]  56.47520  55.99629 157.36762 137.25988  44.57502  50.56882  59.22641
##   [8] 122.56456 152.04580  45.89480  53.27160  54.86798  44.42676  45.53518
##  [15]  46.45026  71.43556  98.03969  54.80378  51.98473  54.24984  63.49157
##  [22]  94.19162 119.36878  63.11927  85.20440  53.59437  47.07464  72.58244
##  [29]  48.04432 128.10041  68.23110  67.89610 447.69317  47.53516  55.44293
##  [36]  57.41286 173.48074 411.70003  61.13883  51.22502  52.48218 136.47735
##  [43]  87.88788  56.30349  85.26216  69.70288  63.46313 142.07557  53.18622
##  [50]  62.34023  51.16395  63.03384  47.93140  57.38418  87.80176 277.12903
##  [57]  49.82380  58.32063  80.54634 129.99614 134.43689  52.25695  59.32467
##  [64] 123.89216  95.19937 126.45357  79.84782  52.12751  45.27465  71.09889
##  [71] 136.97406  51.66760 101.26510  67.57767  48.67016  46.82429  91.78877
##  [78] 123.02311  61.15702  52.87878  97.14927  59.19107  74.04177 142.02361
##  [85] 133.71636 190.90038  52.99529 111.01693  60.83938  47.17763 121.79820
##  [92]  99.97682 106.66113 163.12538  49.00760 125.42358  69.77954 116.73083
##  [99]  48.82816 466.07562 103.06939  45.17687  69.58201  95.47648  56.01931
## [106]  48.09351 106.75947  55.77966 137.88596 136.07840 109.01031 221.28597
## [113]  50.12862  94.40822  50.00842  81.91891  54.14422  61.99626  58.65719
## [120]  60.63120 205.67245  65.18283 100.49221  44.58891  45.20154  48.41369
## [127]  49.29329  68.93000  55.62831  47.50751  49.17419 196.07056  65.80307
## [134]  76.70098 116.92633 102.99612  64.90348  52.27677  67.23522  91.30044
## [141] 147.53295 102.33705  51.44829  97.55914  54.08118  94.16749  66.92000
## [148]  59.45487  66.31452  65.66229 139.14828 123.37560  49.55102 138.56309
## [155]  54.07445 113.97410  48.42030  53.56120  69.71931 159.99350  46.00714
## [162]  67.96598 108.12449  65.06518 149.99143  82.33830  70.86334  64.43596
## [169] 209.26211 122.72246 360.59242  56.93610  51.38663  46.40384 151.47856
## [176]  81.60964  66.30729  61.23806  98.05830 237.25358  49.37891  73.68045
## [183]  48.39007 115.65304  61.05652  70.37009  75.31747  52.47448  62.89114
## [190]  80.35079 263.09535  67.33751 226.52537  67.67711
```

```r
# Broadband
boxplot.stats(phoneData$Broadband)$out
```

```
##   [1]   298.8699   176.2194   882.1243   239.7254   194.6768  1171.5956
##   [7]  1630.8055   448.5648 11515.4155   586.5956  2070.7638   415.6034
##  [13]   177.4830   238.7118  1268.9227   229.6649   348.3830   544.2799
##  [19]   779.4260   752.1689   591.8501   260.0999   857.6930   210.1464
##  [25]   102.9108  2767.3056   428.2134   256.0132   387.8958  1027.1631
##  [31]   240.4011   295.1431  1647.1407   594.0502  1440.2453   149.2298
##  [37]   426.1617   340.9276  1348.6700   104.4849   255.6898   108.9520
##  [43]  1644.2324   368.8631   131.0050   575.6486   214.0288  2645.7512
##  [49]   553.5800  3529.0547   341.6590   673.2021   102.1093   853.5356
##  [55]   636.6742   161.8472   691.8967  3646.3595  1725.4161   270.4349
##  [61]  3117.9002 10567.6396   287.6202  1880.3384  1612.0847   229.8956
##  [67]   359.0350   399.6889   533.1737   287.3233   342.5378   332.4420
##  [73]   649.3090   122.7752   218.2364   546.6547  1405.2932  1202.9800
##  [79]   978.7576   611.4754   104.5983   200.2600  1366.5839   763.7900
##  [85]   798.1965   403.5844   355.0962   439.4737   149.0217   660.5162
##  [91] 16130.8584  2067.7427   155.5430   283.6185   186.2746   159.4196
##  [97]  1558.4625  3564.2916   674.1883  2651.2828  5589.6604   202.8613
## [103]   265.7615   474.5812   765.7827   121.1172   181.7386   746.9508
## [109]  1129.4159  1172.5777   326.8394  1684.2268   141.4775   130.9731
## [115]   257.6480  4391.9129  1364.2325   213.5906  1924.1415   647.2870
## [121]   321.6712   193.2702   235.7429  1398.5909  2887.2541   267.7902
## [127]  1614.3971  1097.4552   333.4411   283.0478  1127.1796   626.9289
## [133]   340.1749   544.6479   200.8349   783.8108   643.6562   959.8850
## [139]   203.4354  1685.7695  2834.6761  1047.5346   653.5152   144.7575
## [145]   230.8728  1120.3934   239.9369   270.2834   215.3416   247.1747
## [151]   185.4953   335.1275  1914.1190   134.0576  1270.0317   339.0496
## [157]   145.5411   298.0549   145.2506   111.7793  2530.6060  2155.4682
## [163]   294.9758   263.5890   705.3659   188.1650   199.4178  2702.0398
## [169]   107.7256   129.8116   251.5254   717.8159   127.2634   331.8079
## [175]   220.2469  1209.5457   534.7895   150.5912   479.6019   247.7422
## [181]   503.4900  1300.7885   270.5607   186.0762   157.8382   520.5307
## [187]   173.6179   101.1419   114.0147   540.3501   562.1192   556.9053
## [193]   401.5194   113.3206  1116.9242   330.8939   487.0823  1582.5480
## [199]  1339.7070  1460.2030   986.6439  3657.9380   717.5289   249.9871
## [205]  1266.3272  1160.7265  1307.0582   743.6476   953.3102   142.4682
## [211]   207.4011  1867.8703   506.6382   191.7865   758.2762  1227.8313
## [217]   270.9734   342.4468   423.6196   546.0165   789.9973   427.7768
## [223]   225.0328   121.7919   769.2511   120.9534   333.8290   145.8380
## [229]   329.9690  1744.4659   191.7789   692.0949   423.3184   471.7050
## [235]  1153.3819   516.6120  2051.8189  1134.7174  1827.3733  1900.3462
## [241]   147.2946   192.8783   758.7245   813.8307   114.7987   535.9033
## [247]   551.0142   138.9427   136.7039  1117.6255   335.3312   285.7674
## [253]  1180.7119   327.1166   146.3626   298.2176  3497.2740  2324.4320
## [259]   230.8631   222.5729   329.6189   780.9269   397.5352   777.4558
## [265]  1649.1091  1082.5934  1442.2081   597.4311   666.6795  2871.9820
## [271]  1362.0965   514.6560   916.1782   658.6601  1042.6717   560.9082
## [277]   488.2513   163.9595   738.3382   384.3228   742.5178   524.7141
## [283]   273.2766   346.3578  1153.3209  1332.8232   588.4576  1387.7035
## [289]   223.9367   866.8280   308.1041  3941.6366   107.4236   261.6446
## [295]  1482.5129   634.1531   101.0039   292.5296   649.8049   503.5204
```

```
## [301]   174.2021   430.7931   249.8175  1075.4171   429.6144   288.3914
## [307]   451.7961   187.2749   635.3655   148.1800  3825.0787   828.7880
## [313]   809.1448   564.1652   434.5501   316.3750  5446.1249   456.2740
## [319]   422.7375   876.9963  1548.6649   467.1255   162.5614   124.5686
## [325]   191.4506  1339.0021   435.5793  1348.4011   537.5324   194.6074
## [331]  1324.1360   841.5414   138.2321   274.7495   114.7818  1597.0646
## [337]  1048.8601   342.7494   311.5578  2249.8258   336.0640   135.1261
## [343]  1757.0632  1765.9699   320.1145   351.1374   571.8041   787.4947
## [349]   983.2474  1249.1557  8569.4811   707.4488   422.4585   112.3696
## [355]   171.0418  1056.9626  2436.1146   144.0490   585.3908  1659.5545
## [361]   104.1399   201.3733   614.9880  2039.2539  1223.5393  1030.3802
## [367]   349.0919   109.7161   154.3834   165.6984   106.5721   147.5655
## [373]   333.0476   103.6142   711.5150   425.7722   534.7880   132.3319
## [379]   680.4822   245.7004   708.9539  1629.0745   219.5080   159.2929
## [385]   177.7507  2315.5307  1492.9583  1335.2844   121.6526  1188.9339
## [391]   703.5088   926.2147   163.8462   109.3810   163.0653   153.5453
## [397]  1027.6406  1565.3210   626.4741   626.7456   437.5249   671.7774
## [403]  1631.1595   117.6500   462.7768   769.3510   116.5336   112.2081
## [409]   151.0123   480.9716   160.6841   337.1716   342.8642   443.9177
## [415]   688.7616   241.3561   237.7011   265.8122   897.3495   347.1103
## [421]   101.0409   907.1204  2693.2557
```

```r
# Multi-variant outlier detection
outlier.scores <- lofactor(phoneData, k=10)
plot(density(outlier.scores, na.rm=TRUE))
```



```
# Top Multi-variant Outliers
outliersOrdered <- order(outlier.scores, decreasing=T)[1:100]

# Location of the top outliers
outliersOrdered
```

```
##   [1]  519   41 2049  790 1199  646  383 1751  566 2042 1194  835  431 1445 1595
##  [16] 1230 1275 1629  537 1274  988 1068 1063 1038  218  980 1843  600 1539 1139
##  [31]  501  473 1060 1586 1796  492 1082 1912   18 1687  164 1114  751  377 1948
##  [46]  235 2054 1280 1468  813  781 1328  422 1997 2038  725  845  350 1983 1583
##  [61]  499  709 1440  825 1571  506 1656  711  412  364  579  632  806  159 1956
##  [76] 1227  761 1233  209   76  155  739 1225 1513 1204  622  441 1144 1850 1591
##  [91] 1264  378 1122 1248   13 1891  574  821 1429 1128
```

```
phoneData.outliersRemoved <- phoneData[-outliersOrdered,]        # Not used but kept as an example
```

4. Perform a K-means clustering

Having worked out the outliers, it is now time to run it through a clustering algorithm. The k-means is probably the most common one for this type of task. For this example, we will wrap it in a function and perform successive plots. Then we will perform primary component analysis (PCA) to determine the variables for clustering. Based on our results from the first two steps, it looks like a cluster size of 4 is appropriate for this data. We will run the kmeans and then graph out those results.

A detailed discussion of k-means clustering in R can be found at this link:

https://www.datacamp.com/community/tutorials/k-means-clustering-r

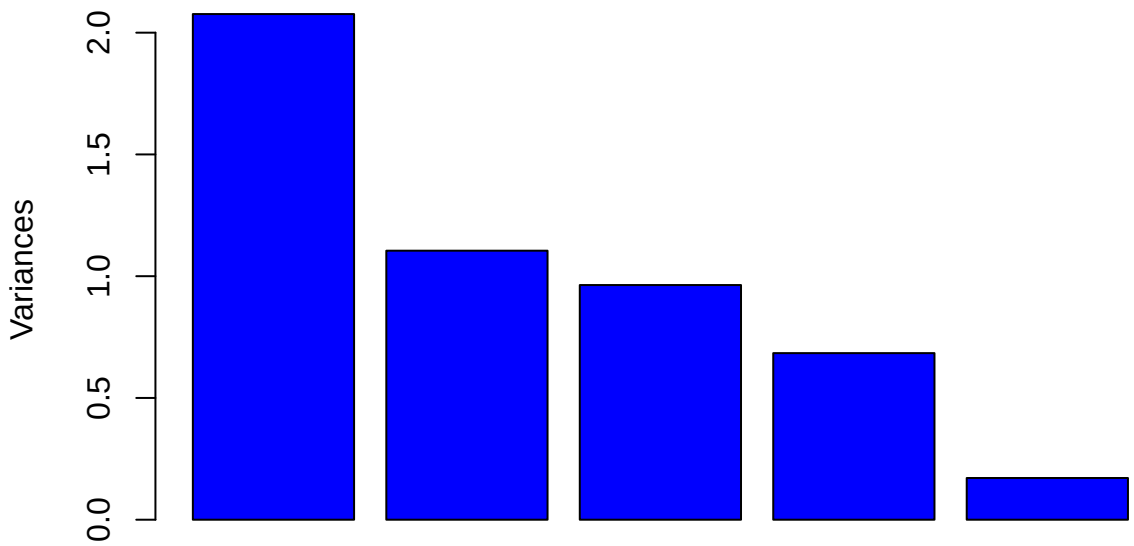
Full Points

Perform a k-means clustering. To do that, you will have to run a PCA analysis to determine which variables are appropriate, and you will have to perform a few plots to display it. The expectation is that it will be relatively accurate. It is not expected to be fully tuned.

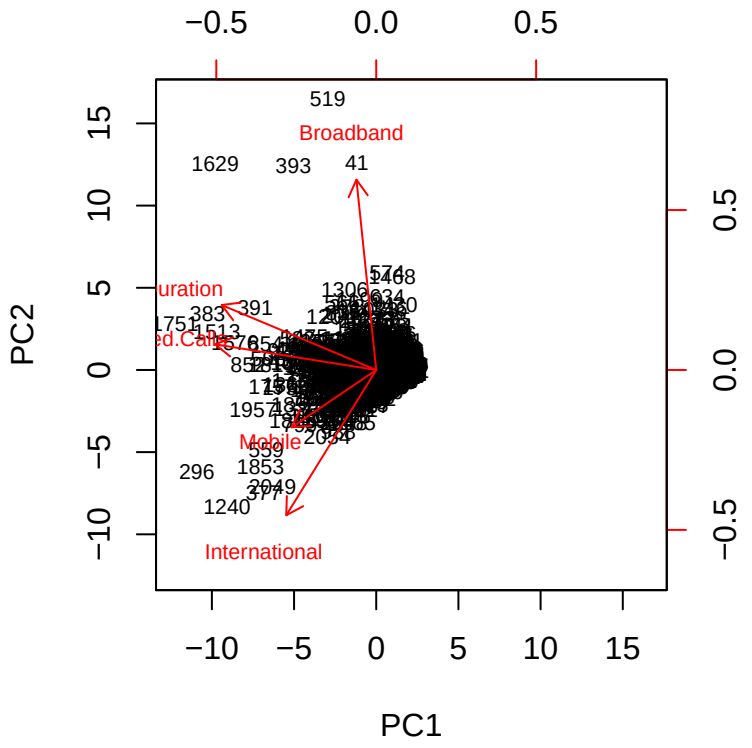
```
wssplot <- function(wssData, numClust=12, seed=12345){
  wss <- (nrow(wssData)-1)*sum(apply(wssData,2,var))
  for (i in 2:numClust){
    set.seed(seed)
    wss[i] <- sum(kmeans(wssData, centers=i)$withinss)
  }
  plot(1:numClust, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")
}

# Run PCA analysis to identify variables for clustering
model <- prcomp(phoneData,center=TRUE, scale = TRUE)
summary(model)
```

```
## Importance of components:
##                            PC1    PC2    PC3    PC4     PC5
## Standard deviation     1.4410 1.0510 0.9816 0.8271 0.41390
## Proportion of Variance 0.4153 0.2209 0.1927 0.1368 0.03426
## Cumulative Proportion  0.4153 0.6362 0.8289 0.9657 1.00000
```

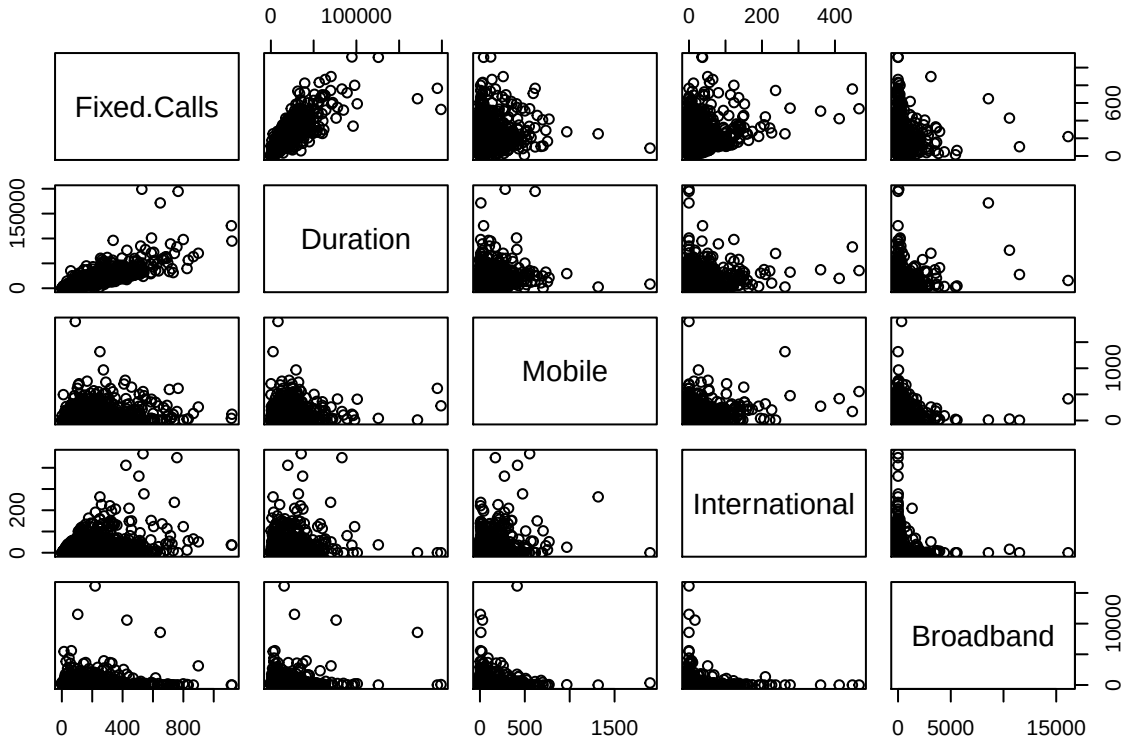
```r
plot(model,main="",col="blue")
```



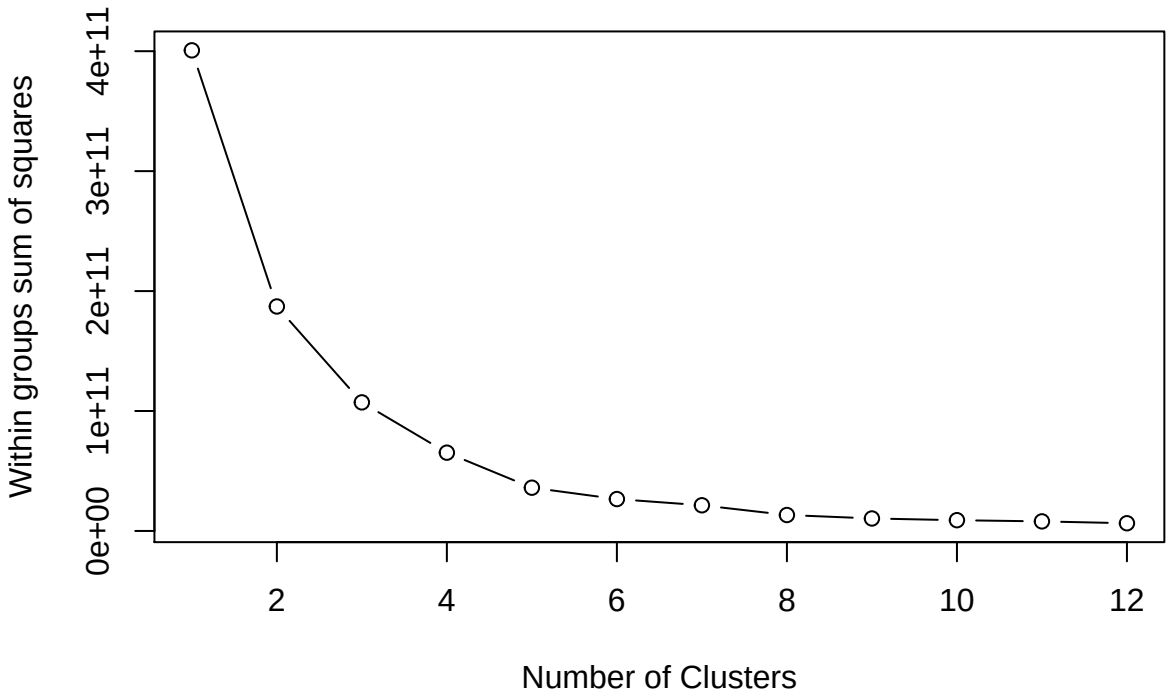

```r
biplot(model,scale=0, cex=.7)
```




```
pairs(phoneData)
```




```
# Remove Broadband & International
phoneDataScale <- select(phoneData,Fixed.Calls,Duration,Mobile)

# Estimate the number of clusters for Kmeans
wssplot(phoneDataScale)
```

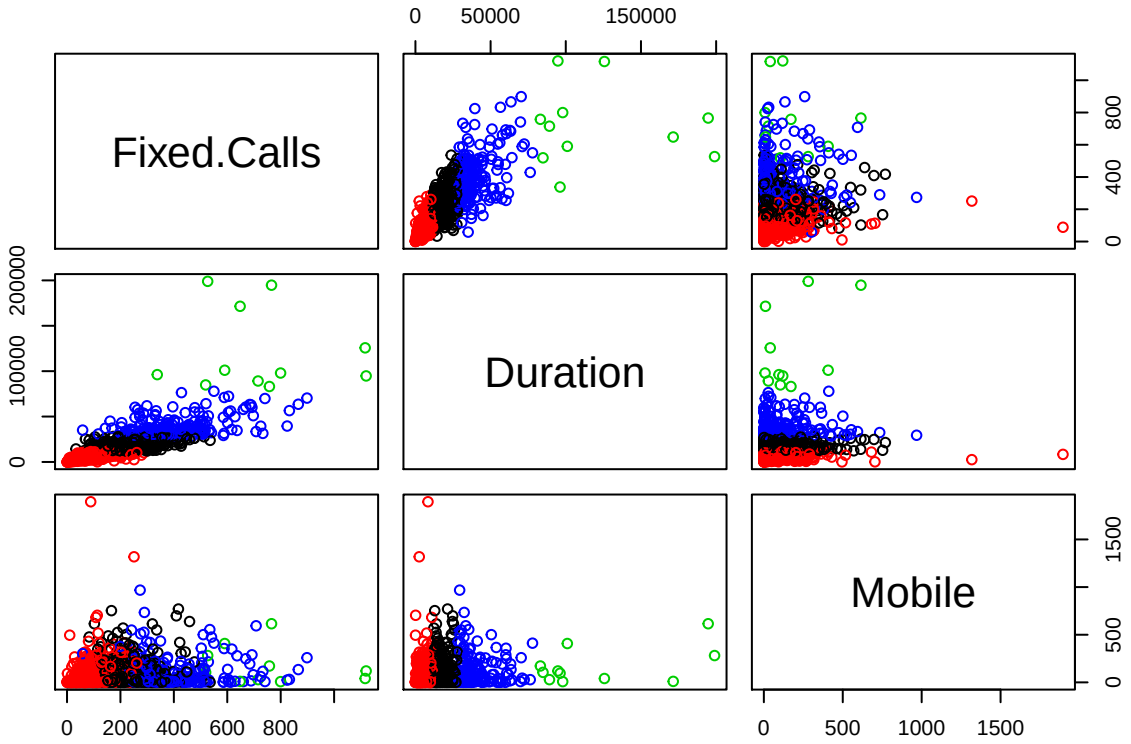



```
# Create clusters
(clust <- kmeans(phoneDataScale, 4))
```

```
## K-means clustering with 4 clusters of sizes 629, 1220, 11, 197
##
## Cluster means:
##   Fixed.Calls   Duration    Mobile
## 1   217.13234  17496.188  76.90998
## 2    95.33229   6102.147  46.48332
## 3   717.88565 121625.616 171.29259
## 4   396.63700  39191.862 108.98351
##
## Clustering vector:
##    [1] 1 2 2 4 4 2 2 2 2 2 2 2 2 4 1 2 1 2 1 2 1 1 2 1 2 2 2 2 4 1 2 2 2 4 4 1 1
##   [38] 2 2 1 1 2 1 2 2 2 2 1 2 2 1 2 1 2 1 2 2 4 1 1 2 2 2 4 2 4 2 2 2 1 2 1 2 2
##   [75] 4 1 1 2 2 2 2 1 2 4 4 2 2 2 4 2 2 2 1 1 1 2 1 1 2 1 2 1 2 2 2 2 4 2 2 2 2
##  [112] 1 1 2 1 4 2 2 1 1 2 1 2 1 2 2 2 2 2 1 1 1 2 4 1 4 2 4 2 2 2 2 1 4 4 2 2 2
##  [149] 1 2 2 2 2 1 2 1 2 2 1 2 2 4 2 4 2 2 2 2 2 2 1 2 1 1 2 2 2 1 2 1 2 2 2 1 4
##  [186] 2 1 1 1 2 1 2 2 2 1 2 1 2 4 2 1 2 2 2 2 2 4 1 2 1 1 2 2 2 1 1 2 2 2 2 2 2
##  [223] 2 4 1 1 1 1 1 2 2 2 1 2 2 2 2 2 1 1 1 2 4 2 2 1 1 2 2 2 2 1 2 4 2 2 4 2 2
##  [260] 4 1 2 2 2 2 2 2 1 1 2 2 2 1 2 1 1 2 2 1 2 2 2 1 2 2 2 3 1 1 2 2 1 2 1 2 3
##  [297] 1 1 2 2 1 1 2 2 2 1 2 1 2 1 2 2 1 2 2 2 2 2 2 2 1 2 2 1 4 1 1 1 2 2 1 1 4
##  [334] 1 1 2 2 2 2 1 1 2 2 4 2 2 2 1 2 2 1 2 2 1 2 2 1 4 2 1 2 1 2 2 2 1 2 2 2 2
##  [371] 2 1 4 2 2 2 1 1 2 2 1 2 3 2 2 1 2 2 2 1 4 1 4 2 2 1 2 2 1 2 2 2 1 2 2 2 2
##  [408] 1 2 2 1 2 2 1 2 2 1 4 1 2 2 2 4 2 2 1 2 2 2 2 2 1 2 4 2 2 2 1 4 2 2 2 2 2
##  [445] 2 2 1 2 1 2 2 2 1 1 2 2 2 2 2 1 1 2 1 1 1 2 2 2 4 2 1 1 1 2 2 2 2 1 2 2 2
```

```
##  [482] 1 2 2 2 2 2 2 2 2 1 1 1 1 4 1 2 1 2 2 1 2 2 1 2 2 2 2 2 2 1 2 4 1 2 1 2 2
##  [519] 1 2 2 2 2 1 2 2 1 1 2 4 1 2 1 2 2 2 2 2 2 1 1 2 1 2 2 1 1 1 2 2 2 1 2 1 1
##  [556] 2 2 1 4 2 2 2 2 1 2 4 2 2 2 4 2 4 1 2 2 2 2 1 2 2 2 2 3 1 2 2 1 2 2 2 1 2
##  [593] 2 2 2 3 2 4 2 1 2 2 4 2 1 1 1 1 2 2 2 1 1 2 2 1 2 1 2 1 2 1 1 4 2 2 1 2 1
##  [630] 2 2 2 2 2 2 2 2 2 2 2 4 2 2 1 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1
##  [667] 2 1 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 1 1 1 1 2 1 1 2 1 2 2 1 1 1 1 2
##  [704] 2 1 1 2 2 2 2 2 2 2 1 1 2 4 2 4 2 2 1 2 2 1 1 1 2 2 2 1 2 2 2 2 2 2 2 1 2
##  [741] 1 2 2 1 2 2 2 2 2 2 2 1 4 1 2 1 2 2 2 4 4 2 2 2 2 2 2 4 2 1 1 1 2 1 1 2 1 4 2
##  [778] 2 2 2 2 1 1 1 1 2 1 2 4 2 2 1 4 1 2 2 1 2 2 2 2 2 2 1 1 4 2 2 1 2 2 2 2 2 1
##  [815] 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 1 2 2 4 2 2 1 2 2 2 2 2 1 2 2 2 1 1 2 2 2
##  [852] 3 2 3 1 1 2 1 1 1 1 2 2 2 2 2 2 1 1 4 2 4 1 2 4 2 1 2 1 2 1 2 2 2 2 2 2 2 1 1
##  [889] 2 1 2 2 1 2 1 4 1 2 2 2 2 2 2 4 2 1 4 2 2 1 2 1 2 1 1 2 2 2 2 2 1 2 4 1 2 2 4
##  [926] 1 2 2 2 2 2 1 2 2 2 1 1 1 1 2 1 1 1 2 2 1 1 4 1 2 1 1 2 1 4 1 2 2 2 2 1 1 2 4
##  [963] 2 2 2 1 2 1 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 2 2 2 2 2 1 2 1 2 1 2 2 2 2 2
## [1000] 1 2 2 1 2 4 4 2 1 2 2 2 2 2 2 2 1 4 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1
## [1037] 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 1 1 2 1 1 2 2 2 1 2 2 1 2 2 4 2 2
## [1074] 4 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2
## [1111] 2 2 1 2 1 2 2 2 2 1 2 2 1 1 1 4 2 1 1 4 1 1 1 2 1 2 2 1 1 2 1 4 2 2 2 2 2 2 2 2 2
## [1148] 2 1 2 4 2 2 2 1 2 4 2 2 2 2 2 1 2 2 2 1 2 2 2 2 1 2 2 1 2 2 2 2 2 1 1 2 2 2 2 2 2
## [1185] 2 4 2 2 2 1 2 2 2 2 2 2 2 2 2 2 4 1 4 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2
## [1222] 4 4 1 1 2 1 2 2 2 1 2 1 2 2 2 2 2 2 2 2 4 1 1 2 1 2 4 1 2 2 2 2 2 2 4 2 2 2 2 2 4
## [1259] 4 2 2 2 2 1 1 2 4 2 2 2 2 2 2 2 1 2 2 2 2 1 1 1 1 1 1 1 1 4 2 2 2 2 2 1 2 2 2 2 1 2 2
## [1296] 2 2 4 2 2 1 1 1 2 1 2 4 2 2 1 1 2 2 2 2 1 2 2 2 2 1 4 1 2 2 2 1 2 4 1 2 2 2 2 2 2 1 2
## [1333] 2 2 2 1 2 1 2 1 2 2 2 2 4 4 2 2 2 2 1 1 2 2 1 2 1 2 4 1 2 4 1 1 2 2 2 1 2 1 4
## [1370] 2 1 1 4 2 2 2 2 4 1 1 1 2 2 2 2 4 2 4 4 1 1 2 2 1 1 2 1 2 2 2 4 1 1 2 2 2 1 4 2 2
## [1407] 2 4 1 4 4 2 2 2 1 1 2 4 4 1 2 2 2 2 2 1 4 1 1 2 2 2 2 1 4 2 2 1 2 2 2 2 1 2 2 2 2 1 2
## [1444] 1 1 2 2 2 4 2 1 1 4 2 4 2 4 1 4 1 1 2 2 2 2 2 1 2 2 2 2 1 4 2 2 2 2 4 1 1 1 2 2
## [1481] 2 2 2 1 1 2 1 1 2 2 2 1 2 2 2 2 4 2 2 2 1 1 1 2 2 2 1 1 2 2 2 2 2 2 2 2 1 3 1 2 2 2 1
## [1518] 2 2 1 4 2 2 1 1 2 2 2 2 2 2 1 2 2 1 1 1 2 2 2 1 2 2 2 1 2 1 2 2 2 1 4 4 2 2 2 2 2 4 1 2
## [1555] 2 2 2 4 4 2 2 4 4 1 2 2 1 2 2 2 2 1 1 2 2 2 1 1 2 3 2 1 1 1 1 2 2 2 2 2 1 2 2 1 1 4 2
## [1592] 2 4 2 2 2 2 1 2 4 1 4 2 2 2 2 1 2 1 2 2 2 2 4 1 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 4 2 1 2
## [1629] 3 1 2 4 1 2 2 1 2 1 2 4 1 4 1 1 1 2 2 1 2 2 2 1 2 1 2 2 2 1 4 4 2 2 2 2 2 1 1 2 1 2
## [1666] 4 2 1 1 2 1 1 2 1 2 2 1 1 2 4 4 1 2 4 2 2 2 2 2 4 1 2 2 2 1 1 4 1 2 2 1 1 2 2 2 2 1
## [1703] 2 1 1 2 2 2 4 1 2 1 4 2 1 2 2 1 2 2 2 1 4 2 2 2 2 1 2 2 1 1 1 4 4 2 2 1 1 2 2 2 1 4
## [1740] 2 4 4 4 1 2 2 2 2 2 2 1 3 2 2 2 4 2 4 2 2 2 2 2 4 1 1 2 2 2 2 1 4 2 2 2 2 2 1 2 2
## [1777] 1 2 2 2 2 2 1 1 2 1 2 2 1 1 2 2 2 1 2 2 2 1 2 2 4 2 2 2 1 1 1 2 2 2 1 1 2 1 2 4 2 2
## [1814] 4 1 2 1 2 2 2 1 2 2 2 1 2 1 2 1 2 2 2 2 1 2 2 2 2 1 2 1 1 2 1 2 4 2 2 2 2 1 1 2 2 2 1
## [1851] 1 1 4 1 4 2 2 2 2 1 4 2 1 2 2 2 2 1 2 2 2 1 2 4 2 1 2 4 2 2 2 2 2 2 2 2 1 2 4 2 2 2 1
## [1888] 2 2 2 1 1 2 2 2 1 2 2 2 2 2 1 2 1 4 1 1 2 2 2 1 2 1 2 2 2 1 2 4 2 1 2 1 2 2 2 2 2 1 2
## [1925] 2 1 2 2 4 2 2 2 2 1 2 4 2 1 2 2 1 4 4 1 1 2 2 2 2 1 2 2 2 1 2 2 2 2 1 2 2 4 1 2 4 1
## [1962] 2 1 1 2 4 2 2 2 2 2 1 2 2 2 1 2 4 2 2 2 1 1 4 1 2 4 1 1 2 2 2 2 2 1 2 2 2 1 1 2 1 4
## [1999] 2 1 1 2 2 2 4 2 2 4 4 1 2 4 1 2 1 1 2 1 2 1 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 2 1 2
## [2036] 2 2 1 2 2 2 2 4 2 1 2 2 1 1 1 2 2 2 2 1 2 2 2 1 4 2
##
## Within cluster sum of squares by cluster:
## [1] 12085382322 10290690581 20065889372 22827998235
##  (between_SS / total_SS =  83.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
plot(phoneDataScale, col = clust$cluster)                    # Visually check clusters
points(clust$centers, col = 1:4, pch = 8, cex = 2)
```




```
# Add cluster data to the phone data
phoneDataCluster <- cbind(phoneRawData,clust$cluster)

# Fix clust$cluster name
colnames(phoneDataCluster) <- c(names(phoneRawData[1,1:6]),"Cluster")
head(phoneDataCluster)
```

```
##        ID Fixed.Calls  Duration     Mobile International Broadband Cluster
## 1 C100000   229.37271 26822.505   0.636458     16.007303   0.00000       1
## 2 C100100   120.84861  7282.827  14.464970      7.593508   0.00000       2
## 3 C100200    97.45483  4385.387   4.307254     13.866595   0.00000       2
## 4 C100300   510.09463 50479.551 500.780837     34.524157 298.86990       4
## 5 C100400   401.63257 46200.675  30.025406     13.371935  69.11522       4
## 6 C100500   159.38233  9796.170  33.726267     12.616191   0.00000       2
```

5. Output to a text file

The final step is probably the easiest. Write the results out to a file, indicating the cluster for each data element.

Full Points

The output should include the data element and the cluster.

```
# Sort data
phoneDataCluster <- arrange(phoneDataCluster,Cluster,ID)
head(phoneDataCluster)
```

```
##        ID Fixed.Calls Duration     Mobile International  Broadband Cluster
## 1 C100000    229.3727 26822.51   0.636458     16.007303   0.000000       1
## 2 C101400    173.3577 20275.93  65.225693      8.374202   5.565405       1
## 3 C101600    230.7237 14939.94  18.419345      5.545259 882.124269       1
## 4 C101800    202.4080 25779.79   0.791364      7.029795   0.000000       1
## 5 C102000    153.1065 13368.85  14.814417     15.851601 194.676754       1
## 6 C102100    157.8083 13727.61 149.224959     22.808399   0.000000       1
```

```
write.csv(phoneDataCluster, file = "Phone Records with Cluster.csv")
```