Data Mining Programming in R Language

Coby Veal, Krunal Patel, and Jin Wang
Department of Mathematics and Computer Science
Valdosta State University, Valdosta, GA 31698, USA

ABSTRACT

Datamining has been used in Computer Science and Mathematical domains since as early as the 1960s. It has many wide reaching applications, and can be performed in many different ways. The R programming language is one such method, and will be the main method discussed in this paper.

Keywords: K-means Clustering; R Language; Linear Regression

I. What is Data Mining?

The term "data mining" is used to describe the computational process of discovering patterns in large data sets, often referred to as "big data". The goal of data mining is to take this "big data" and find the useful information in it, thus transforming it into a much more easily understood data set that can then be used in many different ways. However, the way in which the data is mined is what distinguishes data mining from a simple search method. The searching takes place in an automatic way, and to such an extent that this field is very closely tied with that of AI, or Artificial Intelligence.

Data mining itself is actually only a part of a larger process called Knowledge Discovery of Databases (KDD), which consists of five steps:

(1) Selection
(2) Pre-processing
(3) Transformation
(4) *Data Mining
(5) Interpretation/Evaluation

Many companies will tailor this strategy somewhat to fit their needs, but the basic idea doesn't change.

We won't analyze the other four steps in depth, but Data Mining can be broken down into six different types of "tasks":

(1) Anomaly Detection
(2) Association Rule learning
(3) *Clustering
(4) Classification
(5) *Regression
(6) Summarization
(7)

All of these tasks are useful, the application is what will determine which task needs to be used and when. Often times, more than one task will be used in a "mine". Now that the data mining and its applications have been covered, the question arises: How do we utilize the power of data mining?

II. The R Language

The R language is an open source standard programming language that is used heavily in statistical computing. Ross Ihaka and Robert Gentleman create the R language at the University of Auckland in New Zealand in the early to mid-1990s. Though R has many different functions and applications, we will focus on its data mining capabilities. Specifically, we will see how the R Language can be used to simulate k-means clustering, as well as how to implement basic linear regressions in R.

III. K-means Clustering in R

Clustering describes taking a set of data and grouping it based on some predetermined specifications. K-means keeps true to this idea, with the stipulation that there should be *k* number of groups.

To perform this in R, we first need an initial data set to work with. The [2]data I have chosen is one containing detailed information on

ISBN: 1-60132-430-8, CSREA Press ©

different species of the Iris plant, totaling one hundred and fifty elements. The first step is to read in the information and get it into a workable state.

```
> iris <- read.csv("~/Desktop/iris.csv")
>   View(iris)
> iris.features <- iris
> iris.features$X <- NULL
> view(iris.features)
Error: could not find function "view"
> View(iris.featrues)
Error in View : object 'iris.featrues' not found
> View(iris.features)
> results <- kmeans(iris.features, 3)
```

Typos aside, you can see that it was a matter of specifying the path, and using the assignment character '<-' to place that into the value "iris". In the initial data set, there was a column labeled "X" that listed all of the species names corresponding to each entry, which was unnecessary. This led to the creation of "iris.features", which had all of the same information, minus the 'x' column. Once that was done, the actual k-means algorithm could be run in one line, with the dataset in question on the left, followed by the number of clusters on the right. I knew how many different clusters there would be because of access to the dataset and knowledge that there were three different species. I assigned this value to "results".

```
> results
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  sepal.length sepal.width petal.length petal.width
1     6.850000    3.073684     5.742105    2.071053
2     5.901613    2.748387     4.393548    1.433871
3     5.006000    3.418000     1.464000    0.244000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [25] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [49] 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [73] 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [97] 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2
[121] 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1
[145] 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.24040
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

Running the k-means gave me my results of three clusters of sizes thirty-eight, sixty-two, and fifty. It also shows which cluster each element was mapped to under the "clustering vector:" line.

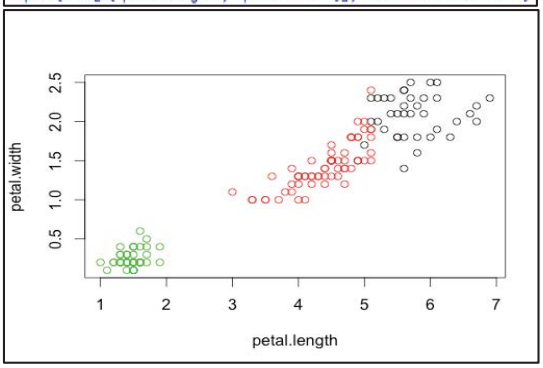
This information does not give us an idea of how accurate our clusters are, so we need to find a way to compare it to our original data set and figure out how close we are.

```
> table(iris$X, results$cluster)

                 1  2  3
  Iris-setosa    0  0 50
  Iris-versicolor  2 48  0
  Iris-virginica  36 14  0
```

The best way to do this is to make a table of the information, using the species for the rows and the clusters as the columns. As seen above, it appears that the clusters were fairly accurate, with the "Iris-setosa" species getting cluster three all to itself. There is some overlapping, but that is all right. Now that we've analyzed the "results" in comparison to the original data set, perhaps it would be useful to get a look at the graph of the k-means to get a better idea of their distribution. For this demonstration, the plot will use the petal.length for the x-axis and the petal.width for the y-axis.

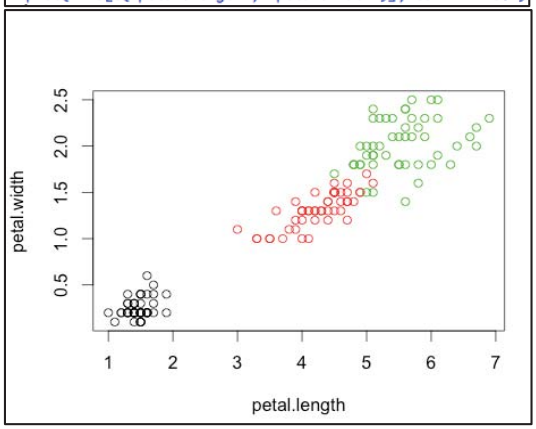
```
> plot(iris[c("petal.length", "petal.width")], col = results$cluster)
```



It would appear that cluster 3 is the green cluster corresponding to "setosa" species. It seems to have a short petal length and petal width. The red cluster corresponds to cluster 2, would appear to be mostly "versicolor", having a medium length and width petal between the three. The black cluster corresponds with cluster 3 above, and is made up of mostly "virginica", though there are quite a few "versicolor" as well.

We can compare this graph to the graph of the orginal data set sorted by species to see how close our clusters were, in a similar way to the comparison of the table above.

```
> plot(iris[c("petal.length", "petal.width")], col = iris$X)
```



Comparing the two, it would appear that the clusters were very accurate, though there were a few outliers that were unable to be mapped correctly. Considering the nature of the data set, plantlife, it is impossible to eliminate them completely due to the variance in which plants can grow. However, our clusters give us a very good idea of the relationship between each species and the length and width of their petals.

IV. Linear Regression in R

A good understanding of how to implement basic K-means clustering in R, and how to analyze your data should have been acquired from the previous section. However, a clustering analysis is not always the ideal technique to use when attempting to compare and analyze data. Another very popular and powerful technique is linear regression, which R has built in functionality for.

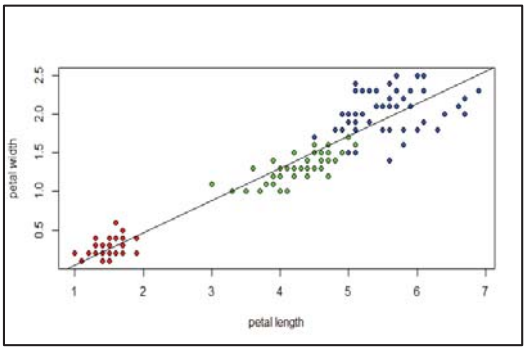
As with any data analysis problem, we must first read in the data we will be using.

```
> iris = read.csv("C:/Users/colby/Desktop/iris.csv")
> View(iris)
> summary(iris)
  sepal.length    sepal.width     petal.length    petal.width                X
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

The previous data set seemed to work well and have a strong correlation, so that is the data that will be used for this as well. Since the contents of the data set has already been discussed, it will not be covered again here. A summary of the important information from the data set has been included in the figure above.

```
> lm(petal.width ~ petal.length, data=iris)$coefficients
 (Intercept) petal.length
  -0.3665140    0.4164191
>
> plot(iris$petal.length, iris$petal.width, pch=21, bg=c("red","green","blue")[unclass(iris$X)],
 xlab="petal length", ylab="petal width")
```

Now that the data is in a usable state, the first thing that must be done is to make a linear model out of the data. This is done with the "lm" command. Within that command, we must specify what elements of that data exactly should be used to make the model. In keeping with the previous test on k-means, we will analyze the petal width in conjunction with petal height, using the "~" symbol to denote the nature of the relationship. Once the relationship is established, we can easily plot the data.



Ignoring the black line for a moment, we can see that we get a generally similar graph as the one we got before, which a byproduct of the linear model function.

```
> abline(lm(petal.width ~ petal.length, data=iris)$coefficients, col="black")
```

With the "abline" function, a regression line can easily be added that corresponds to the trend found in the data in relation to petal width and petal length.

```
> summary(lm(petal.width ~ petal.length, data=iris))

Call:
lm(formula = petal.width ~ petal.length, data = iris)

Residuals:
     Min       1Q   Median
-0.56543 -0.12409 -0.01647
      3Q      Max
 0.13251  0.64278

Coefficients:
             Estimate Std. Error
(Intercept)  -0.366514   0.039889
petal.length  0.416419   0.009613
             t value Pr(>|t|)
(Intercept)   -9.188 3.35e-16 ***
petal.length  43.320  < 2e-16 ***
---
Signif. codes:
  0 '***' 0.001 '**' 0.01 '*'
  0.05 '.' 0.1 ' ' 1

Residual standard error: 0.207 on 148 degrees of freedom
Multiple R-squared:  0.9269,    Adjusted R-squared:  0.9264
F-statistic:  1877 on 1 and 148 DF,  p-value: < 2.2e-16
```

Using the summary function again, a lot of useful data from the regression can be obtained. Each piece of data is useful in certain situations, but for the purpose of checking how well our regression line matches the linear model can be seen in the p-value. Generally, a good p-value is less than 0.5, which would tell you that your data is useful. As you can see, ours is displayed as p-value: < 2.2e-16. This is actually the smallest value that R can produce, which means that we have a very good relationship between petal length and petal width.

V. Summary and Conclusion

Datamining is a very broad topic, with research ongoing in a lot of different areas. What has been presented here is nothing more than a snapshot of a few different methods that "big data" can be analyzed, and how to perform these analytics in the R language.

R is a great tool for performing research on datasets of all sizes, and gives a plethora of useful functions that make the analytics very easy to perform. In most cases this can be done in a few lines! However, R is just that; a tool. Datamining does still rely on the user to know how to reach their desired result. What it excels at is simplifying the process of coding and offering many different tools and methods to obtain the desired result.

References

1. Kiri Wagstaff, Clair Cardie, Seth Rogers, Stefan Schroedl (2001) *Constrained K-means Clustering with Backround Knowledge (*18th Internation Conference on Machine Language p.577-584)
2. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
3. Leemis, Lawrence , *Learning Base R*. The College of William and Mary1 ed , Vol . 1.2015.
4. Yau, C , "Simple Linear Regression in R". RetrievedDecember , 2015 Available: http://www.r-tutor.com/elementary-statistics/simple-linear-regression
5. Kabacoff, R , "Cluster Analysis". RetrievedDecember , 2015 Available: http://www.statmethods.net/advstats/cluster.html