Clustering in R

by Rose Condon

**Question**

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

======== Answer =========

The current situation/task is to find the best cluster K number for the famous iris dataset from UCL repo. Here are the attributes/predictors that I think should be considered.

* sepal length (in cm)
* sepal width (in cm)
* petal length (in cm)
* petal width (in cm)

And the class is categorized as: Iris Setosa, Iris Versicolor, and Iris Virginca.

**Question**

The *iris* data set iris.txt contains 150 data points.The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris ). Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

======== Implementation Steps =========

**Step1**: Load required library, read iris dataset and view data point. We can either read the cvs file from UCL repo, or just using iris data from dataset library. Both ways return same dataset.

**library(datasets)**

**d.iris <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"), header=FALSE, col.names=c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width","Species"))**

**head(d.iris)**

**data(iris)**

**head(iris)**

**summary(iris)**

Console output:

A screenshot of a cell phone

Description automatically generated

**Step2**: Load libraries for cluster algorithm and visualization

**library(tidyverse) # data manipulation**

**library(cluster) # clustering algorithms**

**library(factoextra) # clustering algorithms & visualization**

# Clustering distance measure – Euclidean or Manhattan distance, and view it

**sample\_iris = sample(nrow(iris), size = floor(nrow(iris) \* 0.3))**

**distance <- get\_dist(sample\_iris)**

**fviz\_dist(distance, gradient = list(low = "red", mid = "white", high = "blue"))**

**Step3**: Using K-Means Clustering with standard algorithm

# Scale our dataset, and specify the number of clusters (K), initial center is 3, and configuration is 25

**iris.scaled <- scale(iris[, -5])**

**k2 <- kmeans(iris.scaled, centers=3, nstart = 25)**

**str(k2)**

**# view group results**

**k2**

**# View clusters**

**fviz\_cluster(k2, iris[, -5], ellipse.type = "norm")**

Console output:

**A screenshot of a cell phone

Description automatically generated**

and cluster plot

A close up of a map

Description automatically generated

**Step 4**: Select randomly k objects, for each of the k clusters, initial cluster centers.

**k3 <- kmeans(iris.scaled, centers = 4, nstart = 25)**

**k4 <- kmeans(iris.scaled, centers = 5, nstart = 25)**

**k5 <- kmeans(iris.scaled, centers = 6, nstart = 25)**

# Generate plot for each K

**p1 <- fviz\_cluster(k2, geom = "point", data = iris.scaled) + ggtitle("k = 2")**

**p2 <- fviz\_cluster(k3, geom = "point", data = iris.scaled) + ggtitle("k = 3")**

**p3 <- fviz\_cluster(k4, geom = "point", data = iris.scaled) + ggtitle("k = 4")**

**p4 <- fviz\_cluster(k5, geom = "point", data = iris.scaled) + ggtitle("k = 5")**

# Compare these plots - group 2 plots to avoid reached elapsed time limit

**library(gridExtra)**

**grid.arrange(p1, p2, nrow = 2)**

**grid.arrange(p3, p4, nrow = 2)**

Plots: K=2 and K=3

**A close up of a map

Description automatically generated**

Plots: K=4 and K=5

**A close up of a map

Description automatically generated**

**Step 5**: Iteratively minimize the total within sum of square

Here, I use Elbow Method

**set.seed(123)**

**# function to compute total within-cluster sum of square**

**sumOfSquare <- function(k) {**

**kmeans(iris.scaled, k, nstart = 10 )$tot.withinss**

**}**

**# Compute and plot sum of square for k = 1 to k = 15**

**k.values <- 1:15**

**# extract sum of square for 2-15 clusters**

**sumOfSquare\_values <- map\_dbl(k.values, sumOfSquare)**

# We can view the plot with either plot or factoextra

**plot(k.values, sumOfSquare\_values,**

**type="b", pch = 19, frame = FALSE,**

**xlab="Clusters Number K",**

**ylab="Total sum of squares within Cluster K")**

**A close up of a device

Description automatically generated**

Or view with fviz\_nbcluster method:

**fviz\_nbclust(iris.scaled, kmeans, method = "sumOfSquare")**

**A close up of a map

Description automatically generated**

\*\*From two plots above, we can tell K=3 or K=4 has big impacts for our model. I will take k=4 as a best option\*\*

**Step 6**: Select suggested value of k, compute k-means clustering

**k\_best <- 4**

**set.seed(123)**

**final <- kmeans(iris.scaled, k\_best, nstart = 25)**

**print(final)**

Console output:

A screenshot of a cell phone

Description automatically generated

We can also view with factoextra method fxiz\_cluster

**fviz\_cluster(final, data = iris.scaled, ellipse.type = "norm")**

**A close up of a map

Description automatically generated**

**Question**

Using crime data from the file uscrime.txt (http://www.statsci.org/data/general/uscrime.txt, description at http://www.statsci.org/data/general/uscrime.html), test to see whether there are any outliers in the last column (number of crimes per 100,000 people). Use the grubbs.test function in the outliers package in R.

======== Implementation Steps =========

**Step 1**: install library, load data from http://www.statsci.org/data/general/uscrime.txt and view data

**library(readr)**

**file\_path <- "http://www.statsci.org/data/general/uscrime.txt"**

**crime\_df <- read.table(file=file\_path, sep="\t", header=T, stringsAsFactors = F)**

**crime\_df <- na.omit(crime\_df) # remove missing values**

**tail(crime\_df)**

Console output:

A screenshot of a cell phone

Description automatically generated

**Step 2**: install and load required library outliers.

I will use **grubbs.test** to test different type and options and I try to find if there is an outlier in the last column - number of crimes per 100,000 people.

**library(outliers)**

**grubbs.test(crime\_df[,16], type = 11, opposite = FALSE, two.sided = FALSE )**

**# type=10 is a test for one outlier,**

**grubbs.test(crime\_df[,16], type=10, opposite = TRUE)**

**grubbs.test(crime\_df[,16], type=11) # type=11 is a test for two outliers on opposite tails**

Console output:

A screenshot of a cell phone

Description automatically generated

Grubbs's test is based on the assumption of normal distribution. The probability p-value determines whether an outlier exists.

From test results above, there are two outliers, value 342 and 1993. Let’s verify them with min and max from summary

# Verify the grubbs.test result with a five-number summary

**summary(crime\_df$Crime)**

Console output:



**Step 3**: Create a boxplot and whisker plot to view outliers.

**boxplot(crime\_df[,16], ylab ='Crime Values', xlab="Crimes per 100,000 people", main = 'Crime Outliers')**

Boxplot:

A screenshot of a cell phone

Description automatically generated

**Question**

Describe a situation or problem from your job, everyday life, current events, etc., for which a Change Detection model would be appropriate. Applying the CUSUM technique, how would you choose the critical value and the threshold?

======== Answer =========

According to Wiki, In statistical quality control, the **CUSUM** (or **cumulative sum control chart**) is a sequential analysis technique developed by E. S. Page of the University of Cambridge. **Control charts**, is a statistical process also known as **Shewhart charts** or **process-behavior charts**, is used to determine if a process is in a state of control.

I think CUSUM will work well in transportation industry, specially for cross-board / Region or long-distance transportation. I had worked a blockchain project for two transportation companies, they wanted to implement a tracking system for the shipping status. All status’ are defined separately for different states, and different status (critical value) will have impacts (good or bad) for the coming actions – the change directions.

As the result, we created a state machine engine. It collects the current status for each shipping states, automatically provide the insights for business operator (we call this “digital worker”) to proceed the next action in the transportation workflow. The result is positive.

**Question**

1. Using July through October daily-high-temperature data for Atlanta for 1996 through 2015, use a CUSUM approach to identify when unofficial summer ends (i.e., when the weather starts cooling off) each year. You can get the data that you need from the file temps.txt or online, for example at http://www.iweathernet.com/atlanta-weather-records or https://www.wunderground.com/history/airport/KFTY/2015/7/1/CustomHistory.html . You can use R if you’d like, but it’s straightforward enough that an Excel spreadsheet can easily do the job too.

2. Use a CUSUM approach to make a judgment of whether Atlanta’s summer climate has gotten warmer in that time (and if so, when).

======== Implementation Steps =========

**Step 1**: Load weather data

**wdata <- read.table("temps.txt", stringsAsFactors = FALSE, header = FALSE) # has V1 - VN, and Year**

**head(wdata)**

**nrow(wdata**

Console output:

A screenshot of a computer

Description automatically generated

So, there are 124 records in dataset.

**Step 2**: Compute the estimates needed for running the chart - in this case μ̂ and σ - mean and standard deviation

**library(spcadjust)**

**chart <- new("SPCCUSUM",model=SPCModelNormal(Delta=1));**

**xihat <- xiofdata(chart,X)**

**str(xihat**)

Console output:

A close up of a logo

Description automatically generated

**Step 3**: Compute a threshold that with roughly 90% probability results in an average run length of at least 100 in control

**cal <- SPCproperty(data=X,nrep=50,**

**property="calARL",chart=chart,params=list(target=nrow(wdata)),quiet=TRUE)**

**cal**

Console output:

A screenshot of a cell phone

Description automatically generated

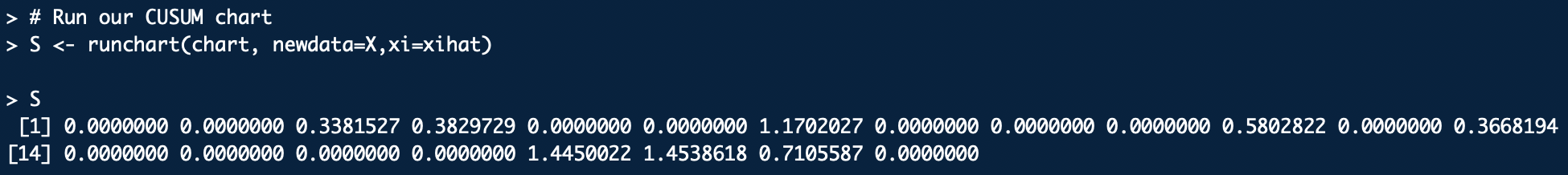
**Step 4**: Run the CUSUM chart and view plot for thresholds

# Run our CUSUM chart

**S <- runchart(chart, newdata=X,xi=xihat)**

**S**

Console output:



# View the plot for thresholds

**plot(S,ylab=expression(S[t]),xlab="Times Intervals",type="l",ylim=range(S,cal@res+1,cal@raw))**

**lines(c(0,100),rep(cal@res,2),col="red")**

**lines(c(0,100),rep(cal@raw,2),col="blue")**

**legend("topleft",c("Adjusted Threshold","Unadjusted Threshold"),col=c("red","blue"),lty=1)**

Plot:

A screenshot of a cell phone

Description automatically generated

**Step 5**: Create a new data frame base on the original one, calculate the average temperature then add to a new column

**wdata\_new <- cbind(wdata)**

**wdata\_new$Avg\_Temp = rowMeans(wdata\_new[,c(-1)])**

**head(wdata\_new)**

Console output:

A screen shot of a computer

Description automatically generated

# I will also need mean for each row and a constant C for small adjustment

**wdata\_mean <- mean(rowMeans(wdata\_new[,c(-1)]))**

**# cat(wdata\_mean)**

**avg\_v <- wdata\_new$Avg\_Temp**

**C <- 4**

**avg\_v\_minus\_i <- avg\_v - wdata\_mean - C**

**# head(avg\_v\_minus\_i)**

# Now I can find the indicator(s) to check if the weather temperature is on changing state and need to take action (change direction)

**cusum <- rep(0,nrow(wdata)) # initial new vect with all zeros**

**for (i in 1:nrow(wdata))**

**{**

**indicator <- cusum[i] + avg\_v\_minus\_i[i]**

**ifelse(indicator > 0, cusum[i+1] <- indicator, cusum[i+1] <- 0)**

**}**

**cat(cusum)**

**dates <- which(cusum >= 83) # Set 83 as threshole base on value of mean**

**cat(dates)**

Console output:

A screenshot of a cell phone

Description automatically generated

**Step 6**: Let’s find out the actual date cusum pointing to; these dates indicates the temperate could reach some level high and action should take.

**v <- character()**

**for (i in 1:nrow(wdata)) {**

**for(j in 1:length(dates)) {**

**if(i==dates[j]) {**

**v[j] <- wdata\_new[i, 1]**

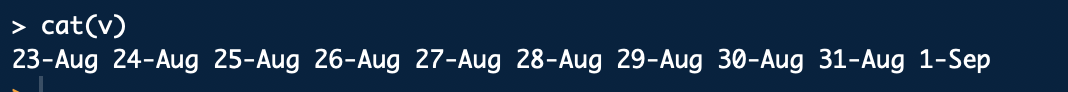
**}**

**}**

**}**

**cat(v)**

Console output:



Thanks for your time!