CSE – 3020

Data Visualization

Lab DA – 5

Name : **Anish Desai**

Reg. No. : **20BCE0461**

Slot : **L39 + L40**

Guided by : **Prof. Jyotismita Chaki**

VIT

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Question – 1 :

Perform the experimentation related to time series analysis.

Code :

Lab DA-5

Anish Desai
20BCE0461

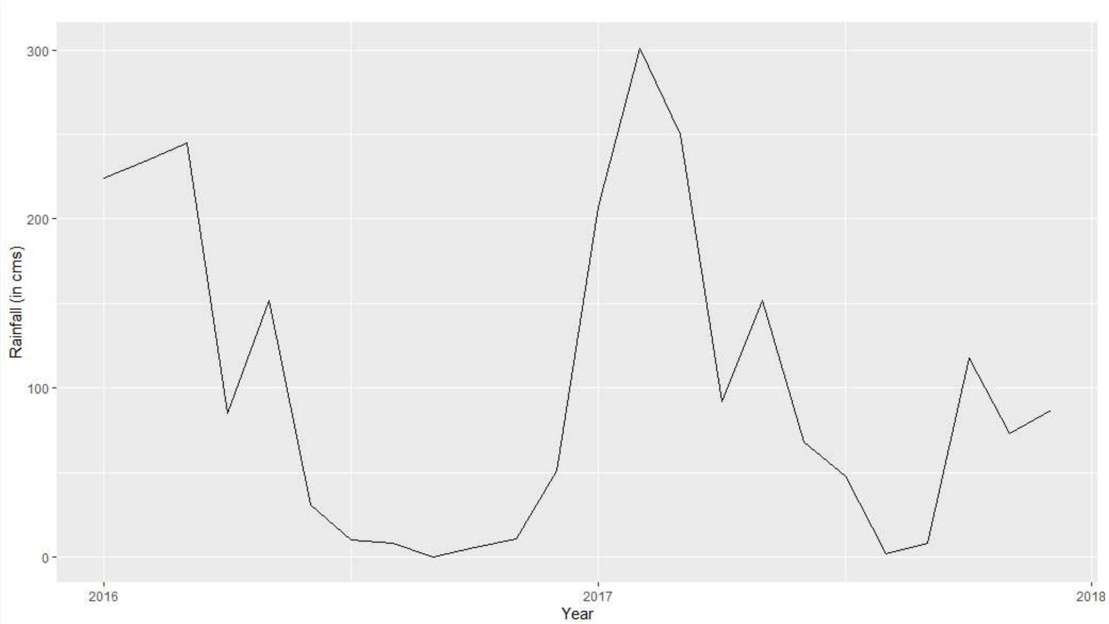
```
#Q1
#Time Series Analysis
library(forecast)
#Using 2016 and 2017 Monthly Rainfall Dataset
rainfall <- c(224.0, 234.0, 245.0, 85.0, 152.0, 31.0,
              10.0, 8.0, 0.0, 6.0, 11.0, 51.0, 207.0,
              301.0, 250.0, 92.0, 152.0, 68.0,
              48.0, 2.0, 8.0, 118.0, 73.0, 87.0)
rainfall.ts <- ts(rainfall, start=c(2016,1), frequency
                                                  = 12)
rainfall.ts
autoplot(rainfall.ts, xlab= "Year", ylab = "Rainfall
                                              (in cms)")

#Decompose it
rainfall.comp <- decompose(rainfall.ts)
#Access components
rainfall.comp $ trend
rainfall.comp $ seasonal
rainfall.comp $ random
autoplot(rainfall.comp)
```

```
#Forecast Method
#a. The Mean Method
# Use meanf() to forecast monthly rainfall in
                                                  2018
rainfall.fc <- meanf(rainfall.ts, h=12)
#Plot and summarize the forecasts
autoplot(rainfall.fc, xlab = "Month", ylab= " Rainfall
                                                  (in cms)")
summary(rainfall.fc)

#b. The Naive Method
#Use naive() to forecast monthly rainfall in
                                                  2018
rainfall.fc <- naive(rainfall.ts, h=12)
autoplot(rainfall.fc)
summary(rainfall.fc)

#c. The Simple Moving Average Method
library(smooth)
rainfall.comp <- decompose(rainfall.ts)
autoplot(rainfall.comp)
summary(rainfall.comp)
```

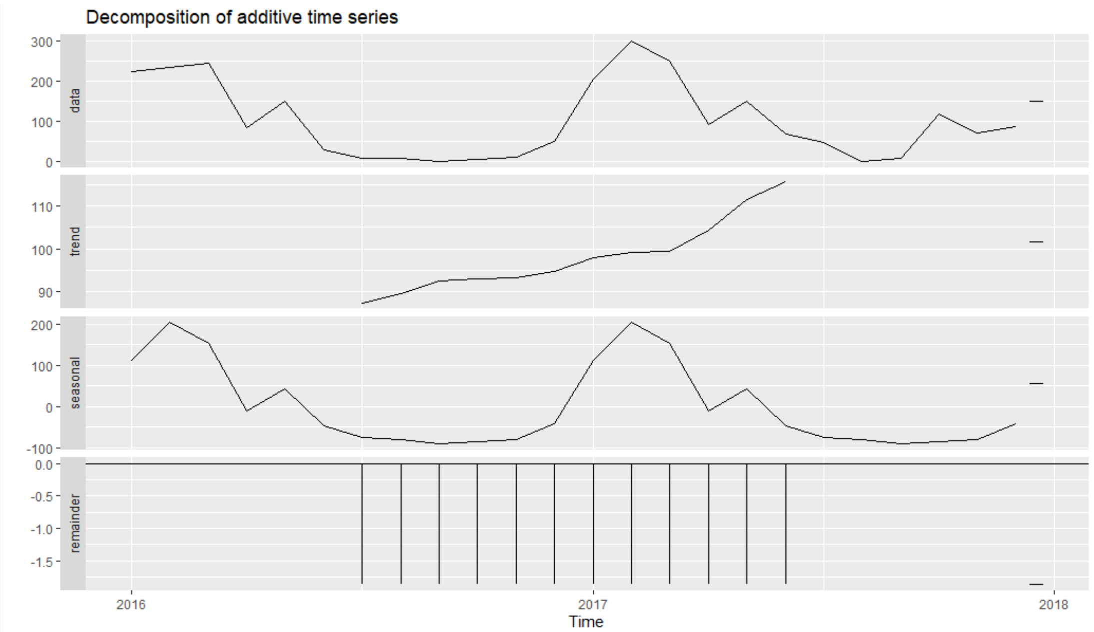
```
#Use sma() to forecast monthly rainfall in
                                        2018
rainfall.fc <- sma(rainfall.ts, order=12, h=12,
                                    silent=FALSE)

#Print model summary
summary(rainfall.fc)
#Print the forecasts
fc <- forecast(rainfall.fc)
print(fc)


#Examining the residuals
rainfall.fc <- meanf(rainfall.ts, h=12)
checkresiduals(rainfall.fc)
```

Output :

```
> rainfall.ts
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2016 224 234 245  85 152  31  10   8   0   6  11  51
2017 207 301 250  92 152  68  48   2   8 118  73  87
```

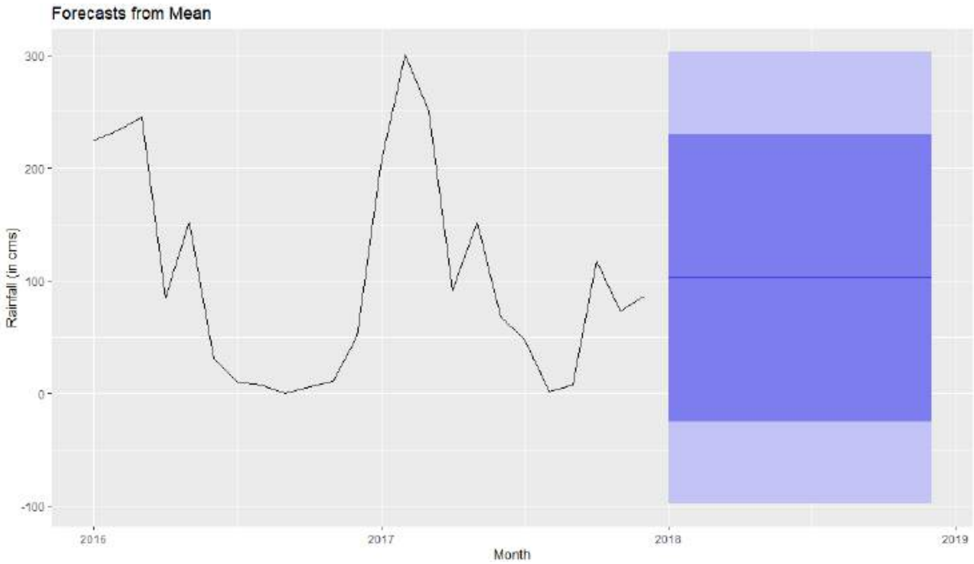



```
> rainfall.comp$trend
           Jan       Feb       Mar       Apr       May       Jun       Jul       Aug       Sep
2016        NA        NA        NA        NA        NA        NA  87.37500  89.45833  92.45833
2017  97.91667  99.25000  99.33333 104.33333 111.58333 115.66667        NA        NA        NA
           Oct       Nov       Dec
2016  92.95833  93.25000  94.79167
2017        NA        NA        NA
> rainfall.comp$seasonal
           Jan       Feb       Mar       Apr       May       Jun       Jul       Aug       Sep
2016 110.94792 203.61458 152.53125 -10.46875  42.28125 -45.80208 -75.51042 -79.59375 -90.59375
2017 110.94792 203.61458 152.53125 -10.46875  42.28125 -45.80208 -75.51042 -79.59375 -90.59375
           Oct       Nov       Dec
2016 -85.09375 -80.38542 -41.92708
2017 -85.09375 -80.38542 -41.92708
> rainfall.comp$random
           Jan       Feb       Mar       Apr       May       Jun       Jul       Aug       Sep
2016        NA        NA        NA        NA        NA        NA -1.864583 -1.864583 -1.864583
2017 -1.864583 -1.864583 -1.864583 -1.864583 -1.864583 -1.864583        NA        NA        NA
           Oct       Nov       Dec
2016 -1.864583 -1.864583 -1.864583
2017        NA        NA        NA
```



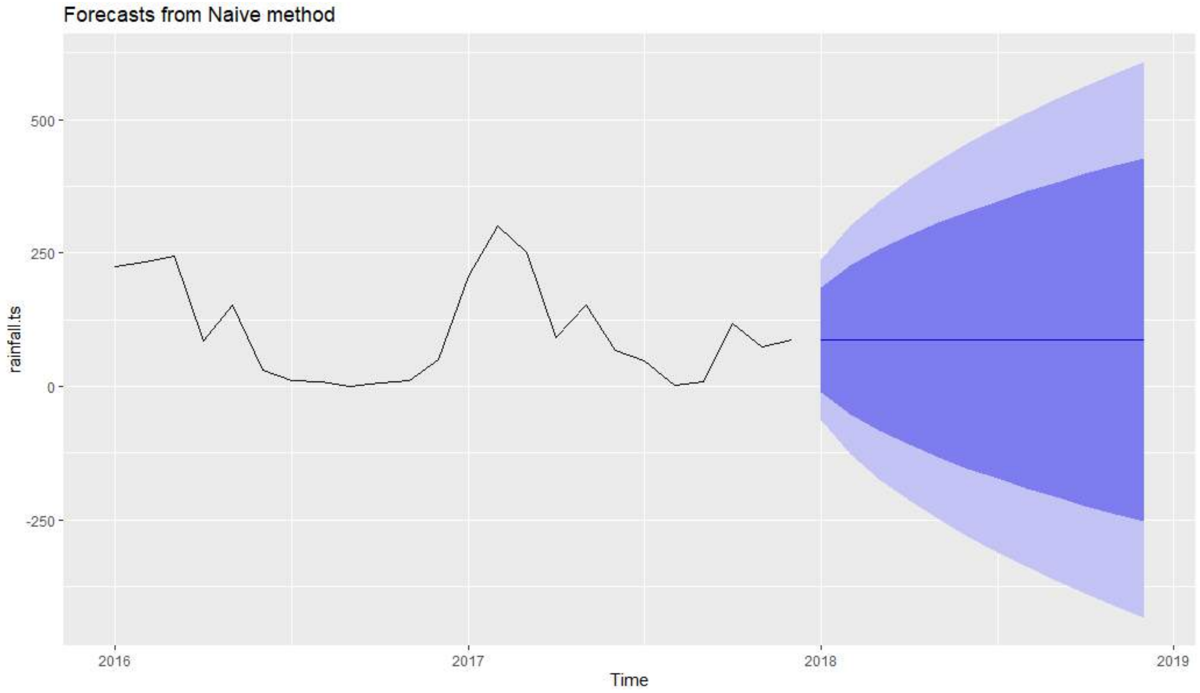
The Mean Method

```
Error measures:
                ME     RMSE      MAE  MPE MAPE     MASE      ACF1
Training set  0 93.03665 79.94792 -Inf  Inf 2.428797 0.6380974

Forecasts:
         Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
Jan 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Feb 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Mar 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Apr 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
May 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Jun 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Jul 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Aug 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Sep 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Oct 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Nov 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
Dec 2018        102.625 -25.35923 230.6092 -98.02943 303.2794
```

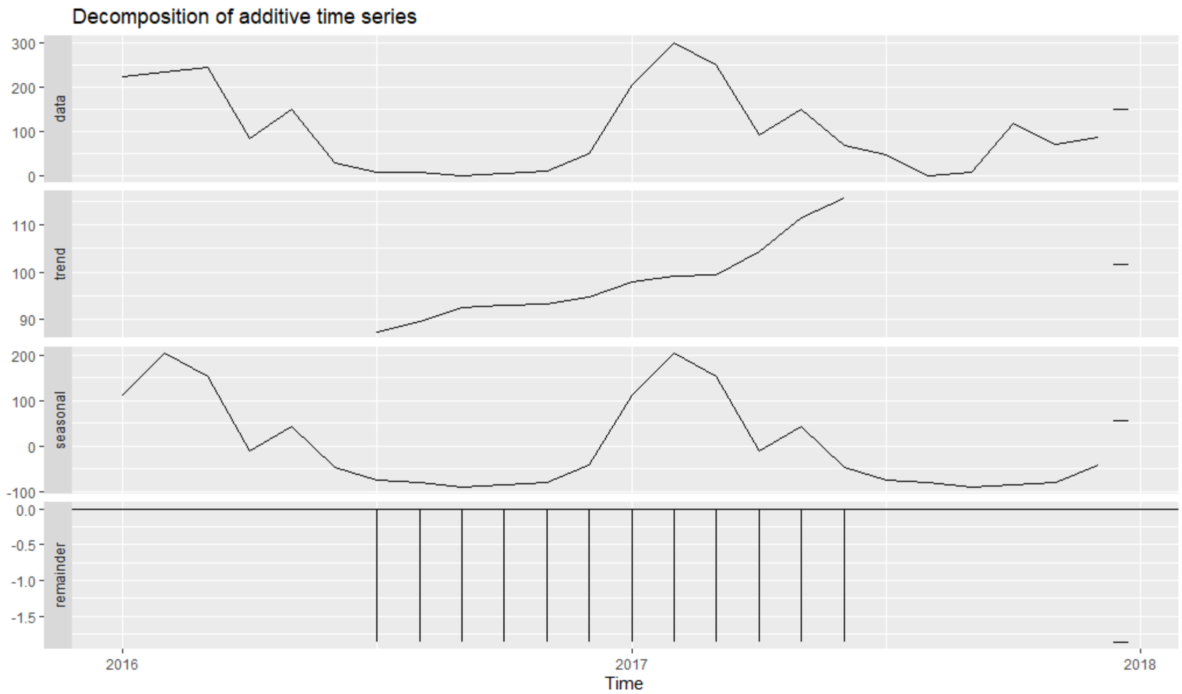


The Naïve Method

```
Error measures:
                    ME     RMSE      MAE  MPE MAPE     MASE       ACF1
Training set -5.956522 76.72282 56.30435 -Inf  Inf 1.710512 -0.08648582

Forecasts:
         Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
Jan 2018             87  -11.32425 185.3243  -63.37397 237.3740
Feb 2018             87  -52.05149 226.0515 -125.66090 299.6609
Mar 2018             87  -83.30260 257.3026 -173.45535 347.4554
Apr 2018             87 -109.64850 283.6485 -213.74793 387.7479
May 2018             87 -132.85971 306.8597 -249.24641 423.2464
Jun 2018             87 -153.84425 327.8442 -281.33949 455.3395
Jul 2018             87 -173.14152 347.1415 -310.85212 484.8521
Aug 2018             87 -191.10298 365.1030 -338.32181 512.3218
Sep 2018             87 -207.97276 381.9728 -364.12190 538.1219
Oct 2018             87 -223.92858 397.9286 -388.52423 562.5242
Nov 2018             87 -239.10465 413.1047 -411.73403 585.7340
Dec 2018             87 -253.60520 427.6052 -433.91070 607.9107
```



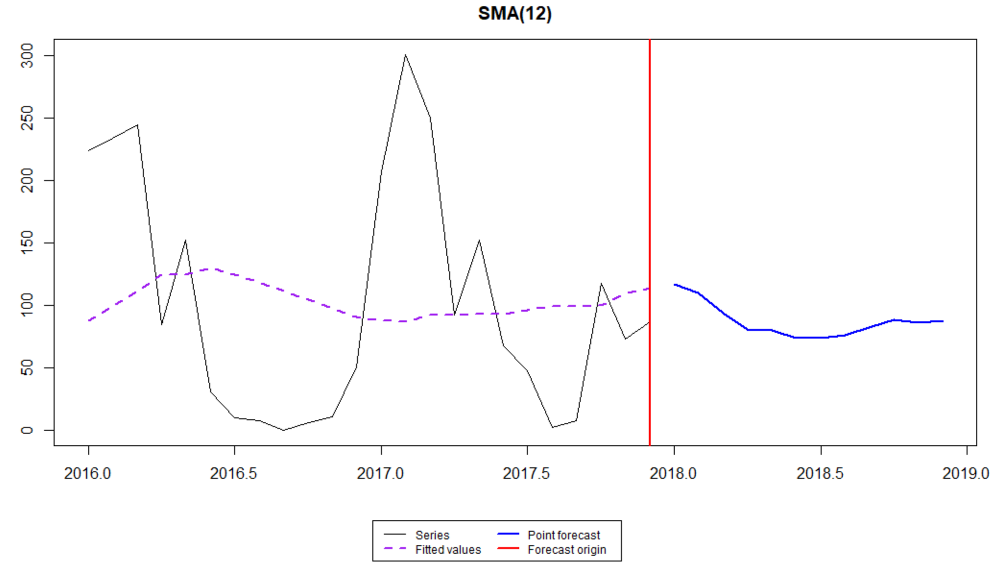
The Simple Average Moving Method



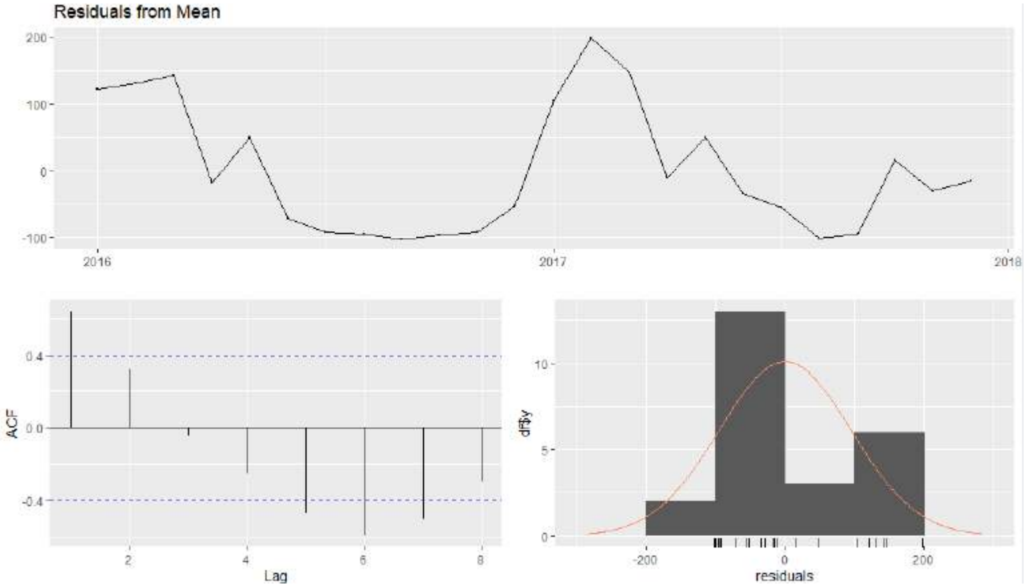
SMA Method

```
Information criteria:
     AIC     AICc      BIC     BICc
292.6244 293.1958 294.9805 295.8885
```

```
> fc <- forecast(rainfall.fc)
> print(fc)
         Point forecast Lower bound (2.5%) Upper bound (97.5%)
Jan 2018      117.16667          -97.05124            331.3846
Feb 2018      109.68056         -105.27988            324.6410
Mar 2018       93.73727         -122.09134            309.5659
Apr 2018       80.71537         -136.12770            297.5585
May 2018       79.77499         -138.25266            297.8026
Jun 2018       73.75624         -145.65348            293.1660
Jul 2018       74.23592         -146.78479            295.2566
Aug 2018       76.42225         -146.47428            299.3188
Sep 2018       82.62411         -142.45398            307.7022
Oct 2018       88.84278         -138.76893            316.4545
```



Examining the Residuals



Interpretation :

Interpretation #Q1

For Time series analysis, we have used the rainfall dataset of a particular region for the years 2016 and 2017. The subsequent line plot is visualized.

The dataset is given as input in the form of vector and is then divided using a frequency of 12, implying monthly distribution.

It is then decomposed and three important parameters are obtained :

trend : long term movements in the mean
trends of increase or decrease or stagnance of rainfall

seasonal : Repetitive seasonal fluctuation of data
Patterns that repeat with a fixed period of time.

random : Residual of original time series after seasonal and trend series are removed

The trend component in the result shows that there is general increase in rainfall from the last half of the year to the first quarter of the next year

The seasonal component in the result shows the pattern of general increase in first quarter, followed by slight variation and then a steep decrease, then an increase towards the first quarter of next year, and same cycle repeated.

Using #Mean Method, we have forecasted the rainfall of 2018. The output shows that there will be rainfall of 102.625 cms, with a 95% confidence interval of 0 to 303.2794 cms. This means we are 95% sure that the rainfall for that period will be between 0 and 303.2794 cms.

Using #Naive Method, the forecast is 87cms of rainfall, with a 95% confidence interval of 0 to 237cms in Jan 2018, ..., 0 to 607.9107cms in Dec 2018.

Hi 95 - Upper bound at 95% confidence
Lo 95 - Lower bound at 95% confidence

Lo 95 and Hi 95 is more significant as compared to Lo 80 and Hi 80.

The #SMA method works the best. It has pinpointed different forecasts for all months based on historical data.

The lower and upper bounds at 97.5% (greater) confidence interval is appreciable.

The solid blue line in the graph shows point forecasts for the months of 2018.

On examining the residuals, which is the difference between the observations and corresponding fitted values, we get to know that the rainfall has unimodal distribution, as is evident from the histogram possesing only one peak. Some of the spikes are outside the blue line implying some part of residuals info is useful for forecasting.

The line plot depicts the difference b/w observations and fitted values, the graphs of which can be seen in SMA method.

Question – 2 :

Perform the experimentation related to visualization of streaming data.

Code :

```
#Q2
# Stream Data Visualization
library(tidyverse)
library(stream)
set.seed (1000)
#DSD_Gaussians
stream <- DSD_Gaussians (K=3, d=2)
plot(stream)

#DSC_DStream
dstream <- DSC_DStream (gridsize = .1, Cm = 1.2)
update(dstream, stream, n = 500)
dstream
plot(dstream)
```

```
#k-Means Clustering
km <- DSC_kMeans(k=3)
recluster(km, dstream)
plot(km, stream, type = "both")

#DSD_BarsAndGaussians
stream2 <- DSD_BarsAndGaussians(angle = 45,
                                noise = 0.1)
plot(stream2)

# DSD_mlbenchData
stream3 <- DSD_mlbenchData("Shuttle")
stream3
plot(stream3, n=100)

#DSD_mlbenchGenerator
stream4 <- DSD_mlbenchGenerator(method = "cassini")
stream4
plot(stream4, n=500)
library("mlbench")
set.seed(1234)
Cassini <- mlbench.cassini(1000)
view(Cassini)
```

```
# DSD_Target
stream5 <- DSD_Target()
plot(stream5)

# DSD_UniformNoise
stream6 <- DSD_UniformNoise(d=2)
plot(stream6, n=100)
stream7 <- DSD_UniformNoise(d=3, range =
                rbind(c(0,1), c(0,10), c(0,5)))
plot(stream7, n=100)

set.seed(1000)
stream8 <- DSD_Gaussians(K=3, d=3, noise = .05,
                p= c(.5, .3, .1))
stream8
p <- get_points(stream8, n=5)
p
p <- get_points(stream8, n=100, class= TRUE)
head(p, n=10)
plot(stream8, n=500)
plot(stream8, n=500, method = "pc")
```

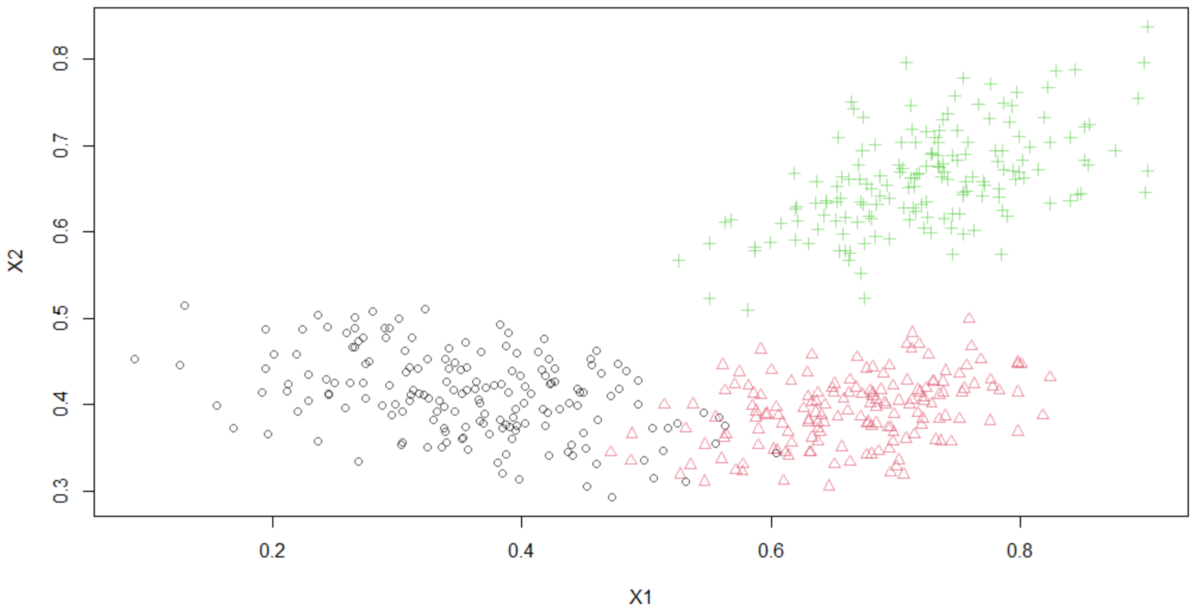
```
# DSD_ Benchmark
set.seed(1000)
stream <- DSD_ Benchmark(1)
stream


# Animation
library('animation')
reset_stream(stream)
animate_data (stream, n= 10000, horizon =100,
                    xlim = c(0,1), ylim = c(0,1))

animation :: ani.options( interval = .1)
ani.replay()
saveHTML(ani.replay())
saveGIF(ani.replay())

# Outlier generating data streams
set.seed(1000)
stream <- DSD_Gaussians (k=3, d=2, outliers =4,
              outlier_options = list (outlier_horizon
               = 10000), separation = 0.3,
               space_limit = c(0,1))
```

```
reset_stream(stream)
P <- get_points (stream, n=10000, outlier= TRUE)
head(P)
out_marks <- attr(P, "outlier")
sum(out_marks)
which (out_marks)

# Advanced statistical data streams
set.seed (1000)
stream1 <- DSD_Gaussians (K=3, d=2,
                          variance_limit = 0.2,
                          space_limit = c(0,5))
plot (stream1)
set.seed (1000)
stream2 <- DSD_Gaussians (K=3, d=2,
                          variance_limit =2,
                          space_limit = c(0,5))
plot (stream2)
```

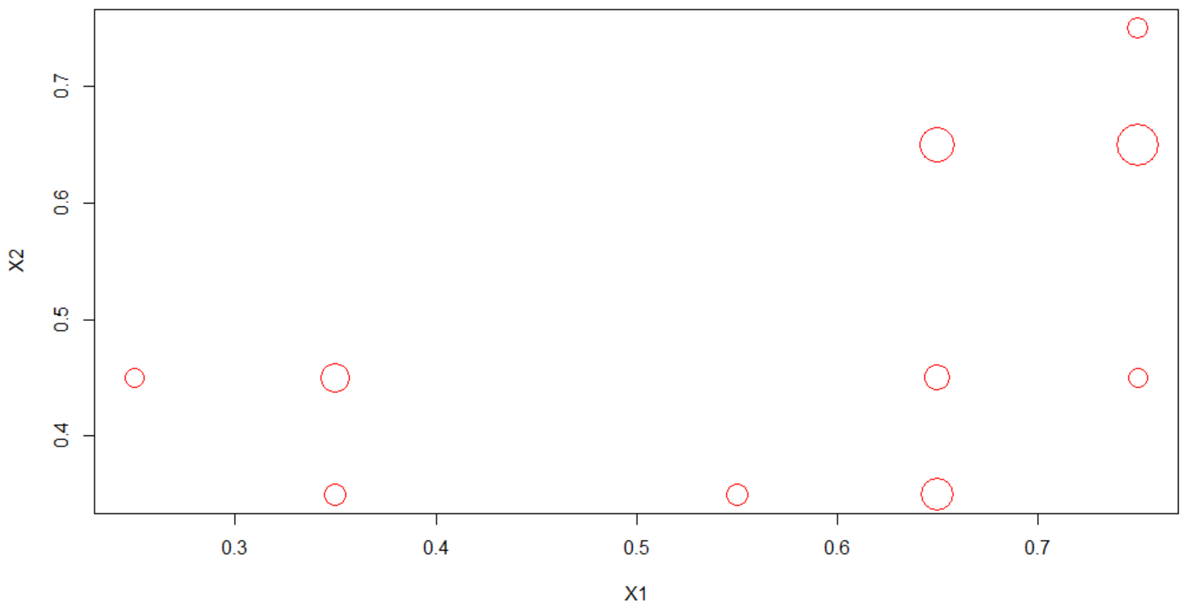
```
set.seed(1000)
stream1 <- DSD_Gaussians(k=5, d=2,
                         variance_limit = 0.2,
                         space_limit = c(0,7),
                         separation_type = "Mahalanobis",
                         separation = 4)

plot(stream1)

set.seed(1000)
stream2 <- DSD_Gaussians(k=5, d=2,
                         variance_limit = 0.2,
                         space_limit = c(0,15),
                         separation_type = "Mahalanobis",
                         separation = 10)

plot(stream2)
```

Output :

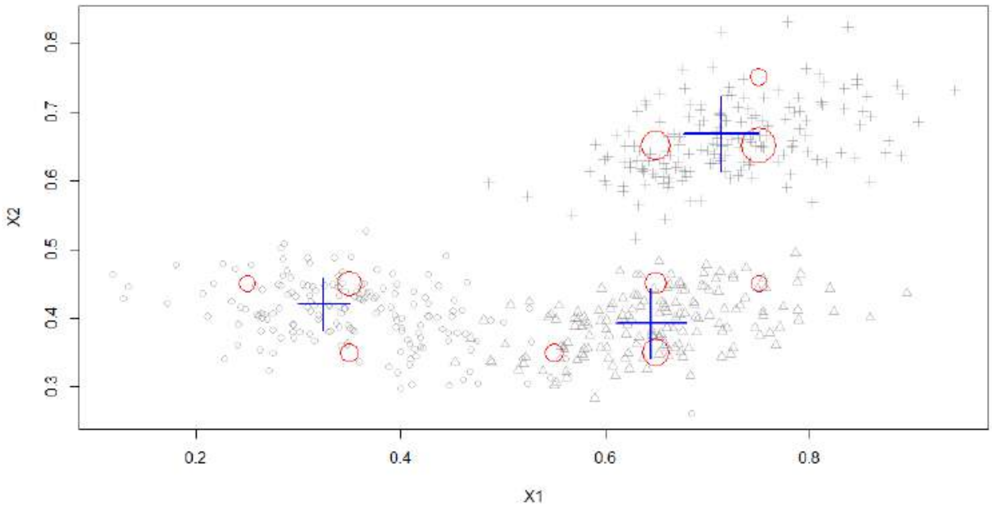
DSD_Gaussians



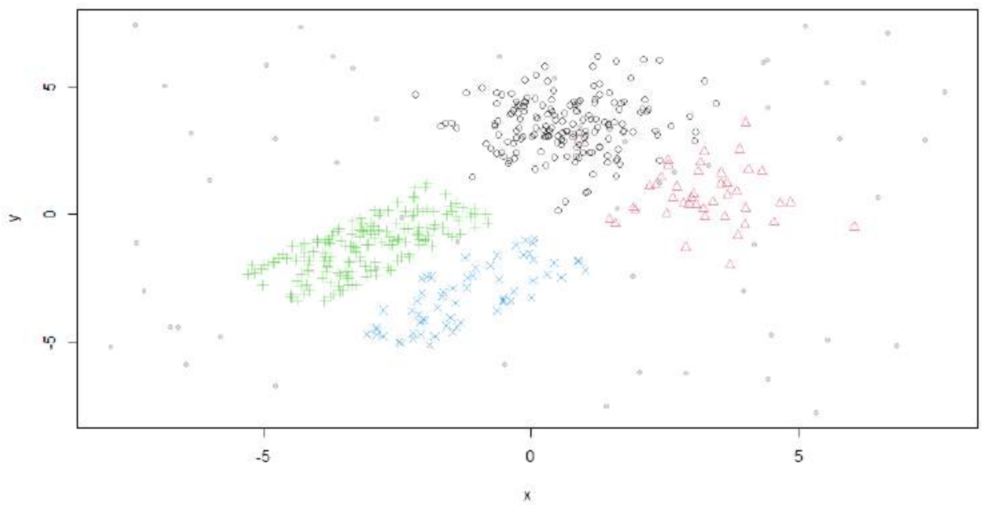
DSC_DStream



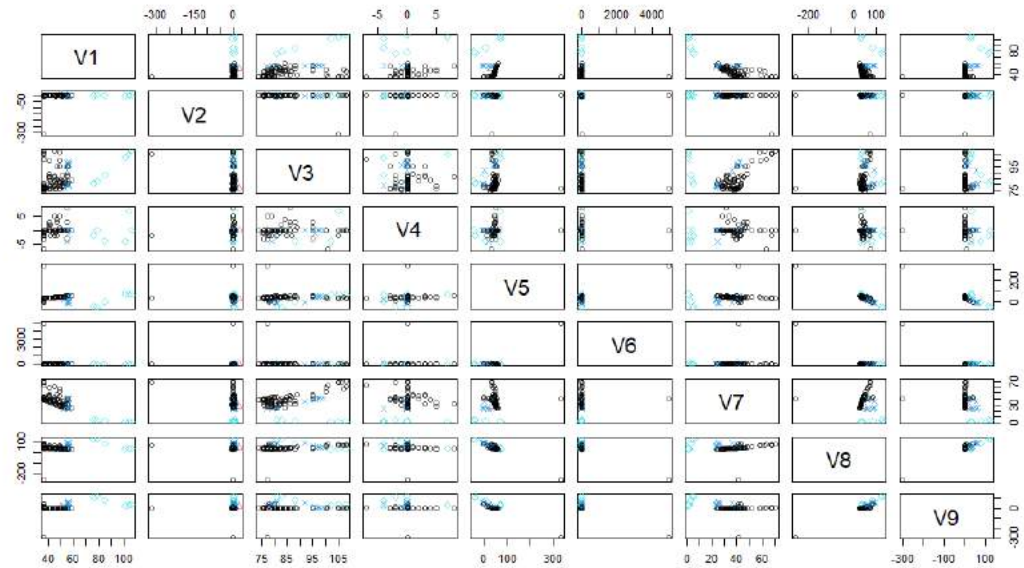
DSC_Kmeans



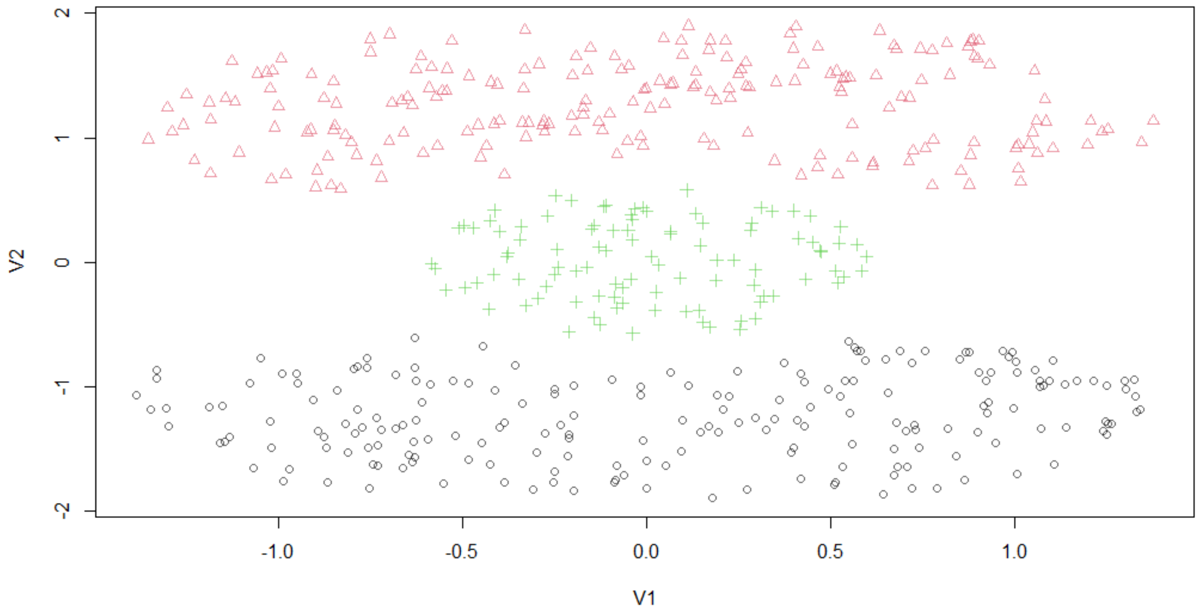
DSD_BarsAndGaussians



DSD_mlbenchData



DSD_mlbenchGenerator



Cassini

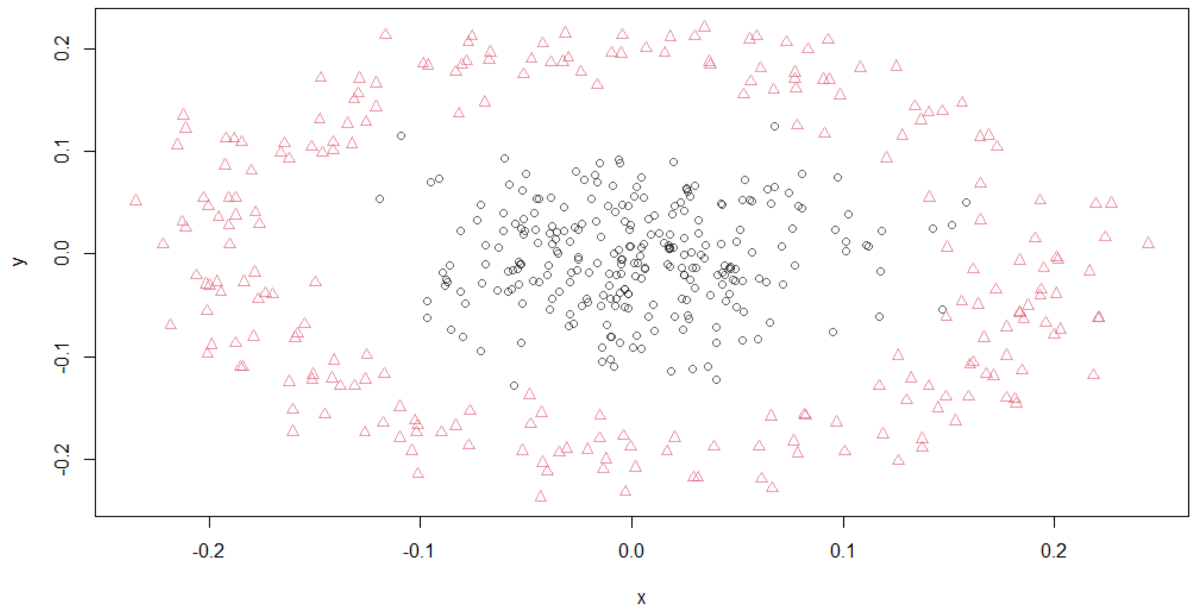
LabDA5.R × | Cassini ×

Filter

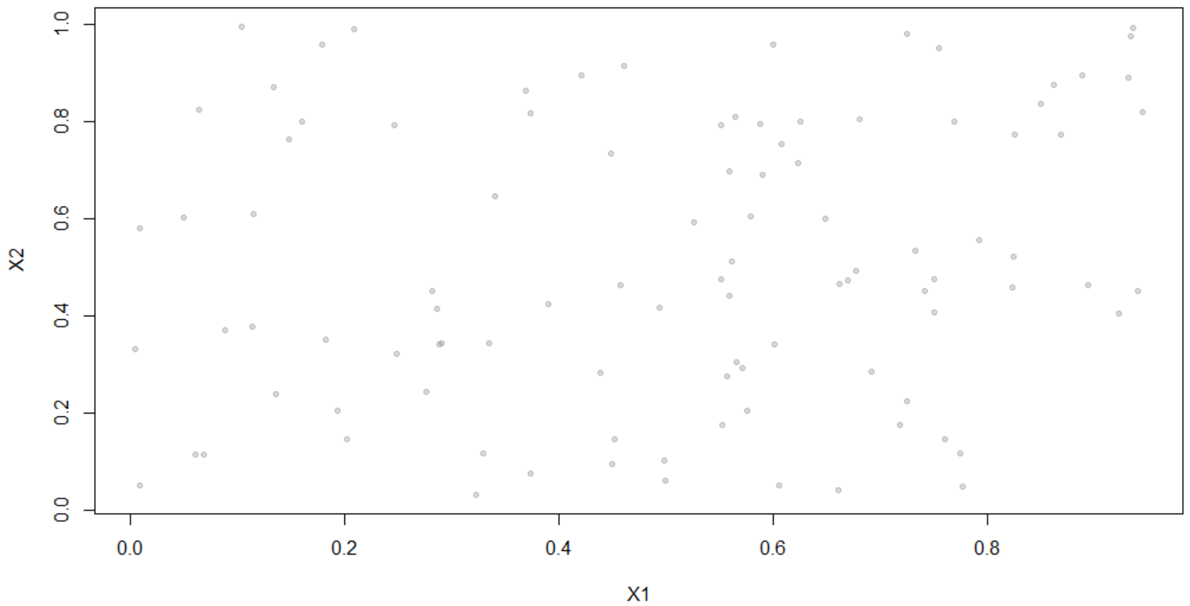
	x.1	x.2	classes
390	-0.666769904	-1.6944355	1
391	-0.898591719	-1.0651033	1
392	-0.925073671	-0.6276072	1
393	0.383088091	-0.7737175	1
394	-0.118735244	-1.3380159	1
395	0.569928076	-1.3537435	1
396	-0.254177770	-1.8574946	1
397	-0.396988234	-1.5006498	1
398	0.174372763	-1.6067634	1
399	0.774992774	-0.6294312	1
400	0.459681464	-1.6116054	1
401	0.900276876	1.4799737	2
402	0.051264218	1.4544373	2
403	-0.537675977	1.5313513	2
404	-0.329016929	1.4487847	2
405	0.821191215	0.9804505	2
406	-0.841843498	1.3126885	2
407	-0.919095283	0.9359875	2
408	0.342833480	1.0454081	2

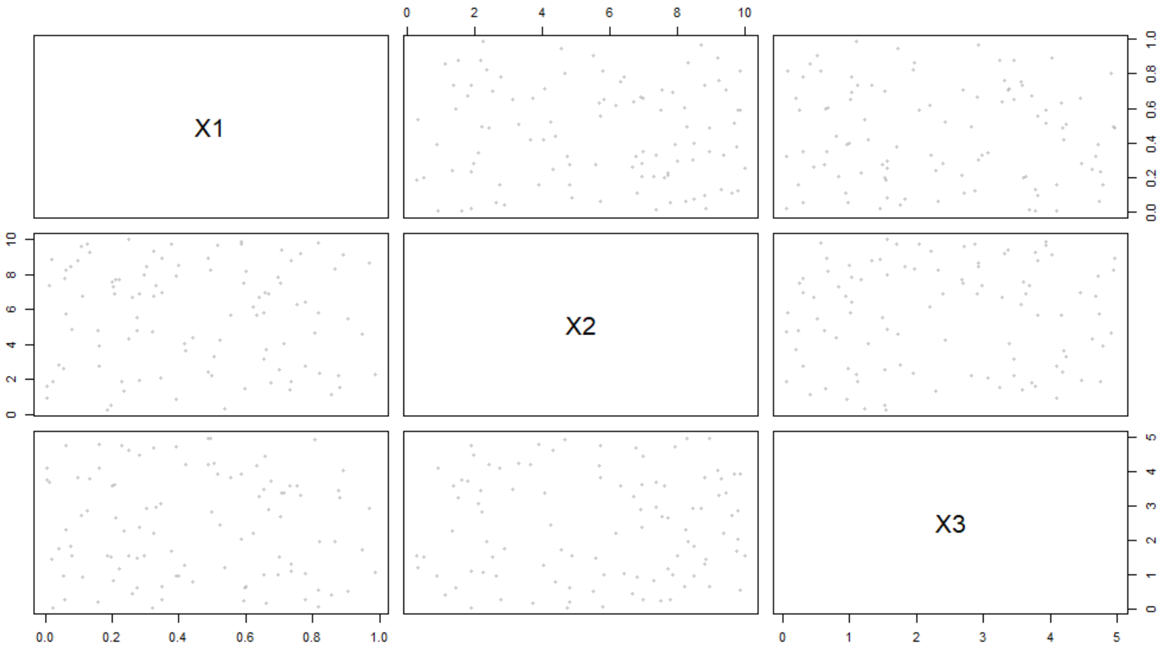
Showing 390 to 408 of 1,000 entries, 3 total columns

DSD_Target

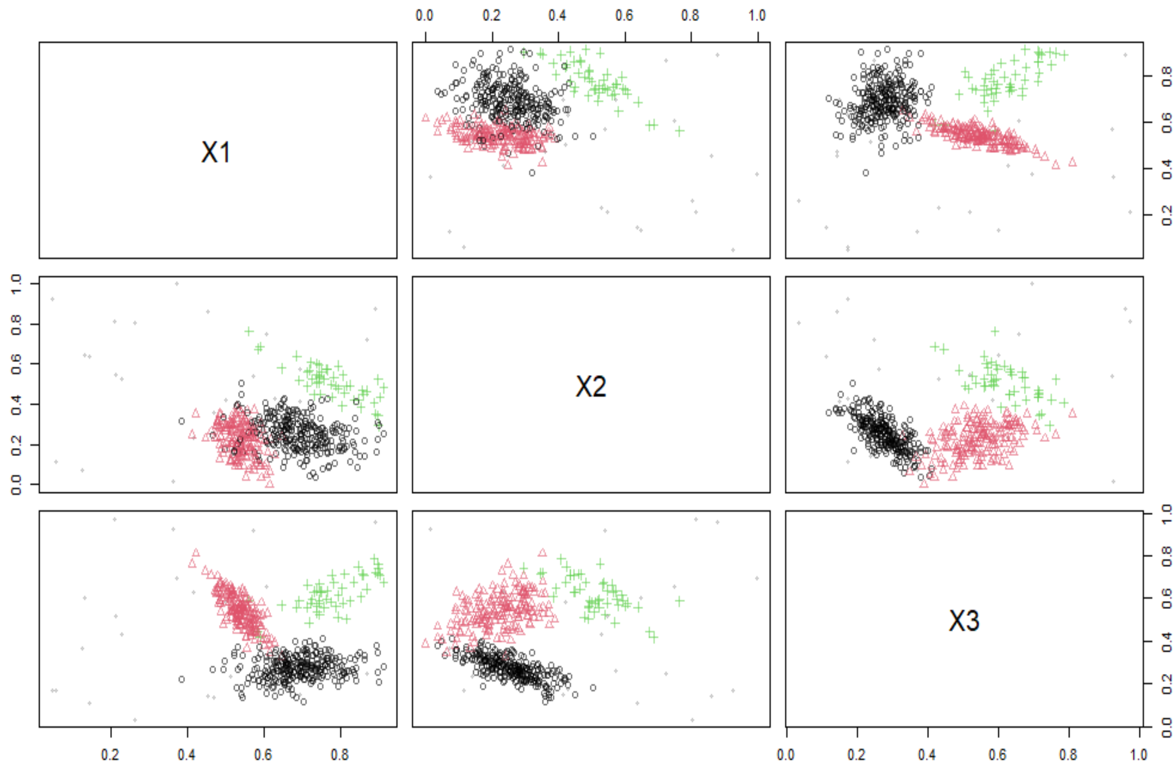


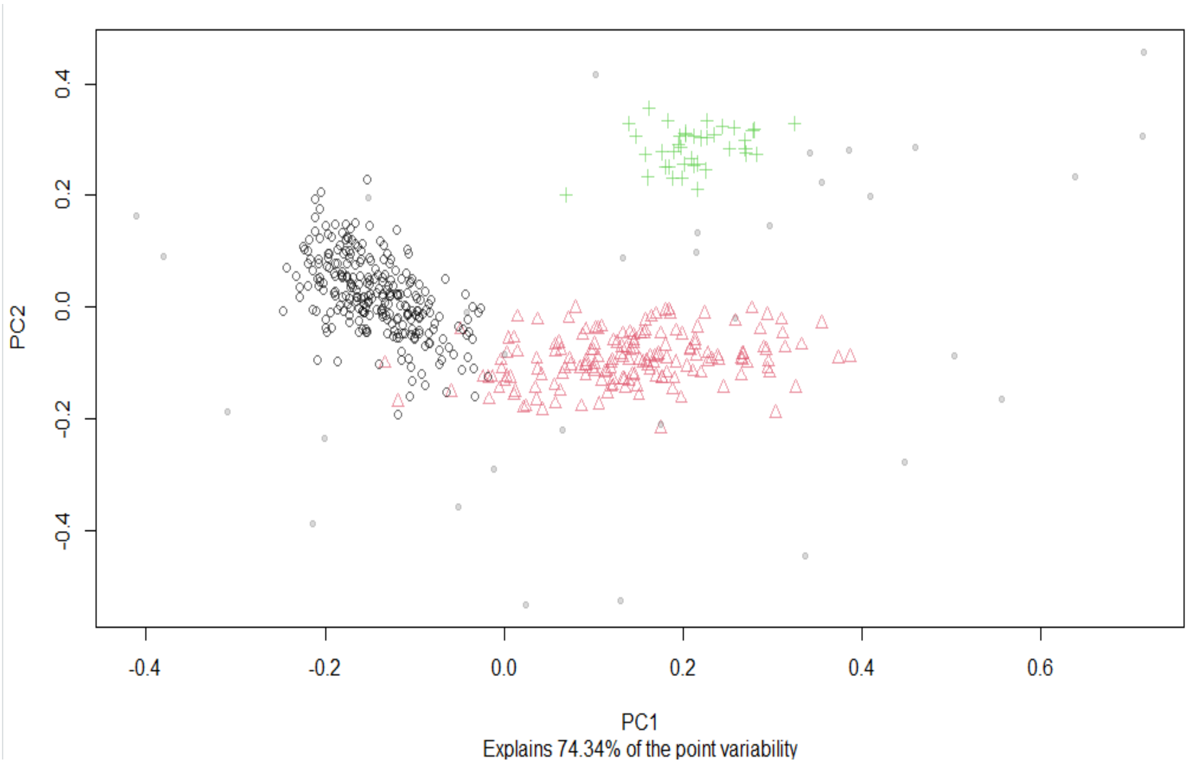
DSD_UniformNoise



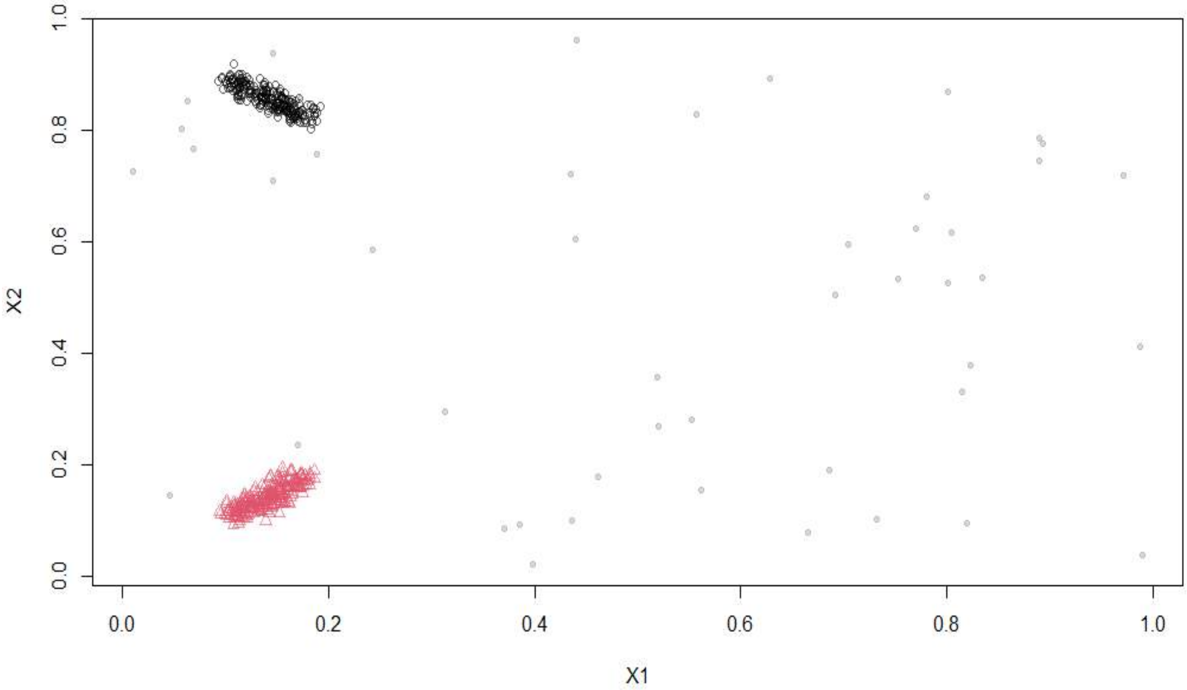



```
> p <- get_points(stream8, n = 5)
> p
         X1        X2        X3
1 0.7195163 0.2740838 0.2828963
2 0.5558063 0.2206600 0.5298132
3 0.5393841 0.2036573 0.5496762
4 0.5848149 0.2033484 0.3809868
5 0.8954718 0.4628273 0.7422705
> p <- get_points(stream8, n = 100, class = TRUE)
> head(p, n = 10)
          X1        X2        X3 class
1  0.7405084 0.4446734 0.2358457     1
2  0.5893800 0.3944584 0.1881733     1
3  0.7139894 0.2890633 0.2693328     1
4  0.7328402 0.2212008 0.3735102     1
5  0.6103774 0.3471980 0.2174912     1
6  0.7602311 0.2082705 0.3053087     1
7  0.7463360 0.2699001 0.3570198    NA
8  0.8170129 0.2040181 0.2849032     1
9  0.5741575 0.2501142 0.5661521     2
10 0.6742316 0.2714565 0.2032578     1
```

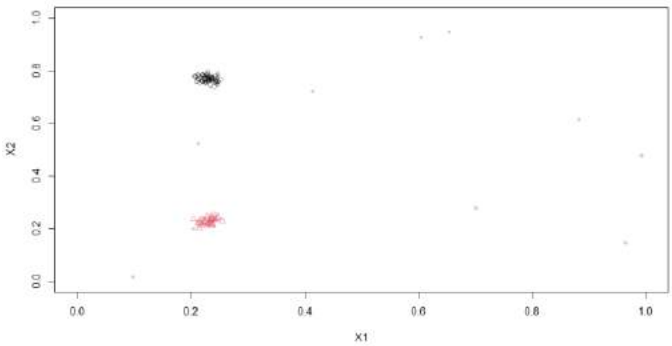

X1
X2
X3

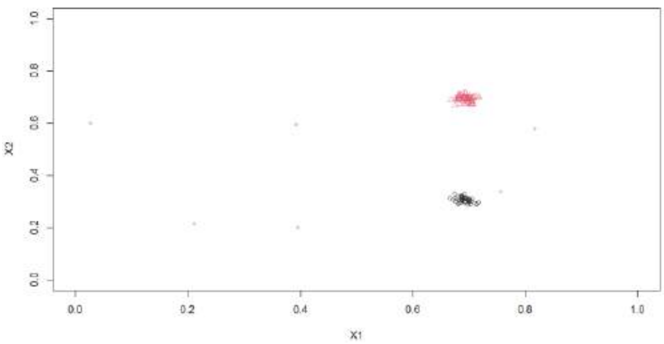

PC2
PC1
Explains 74.34% of the point variability

DSD_Benchmark



Animation



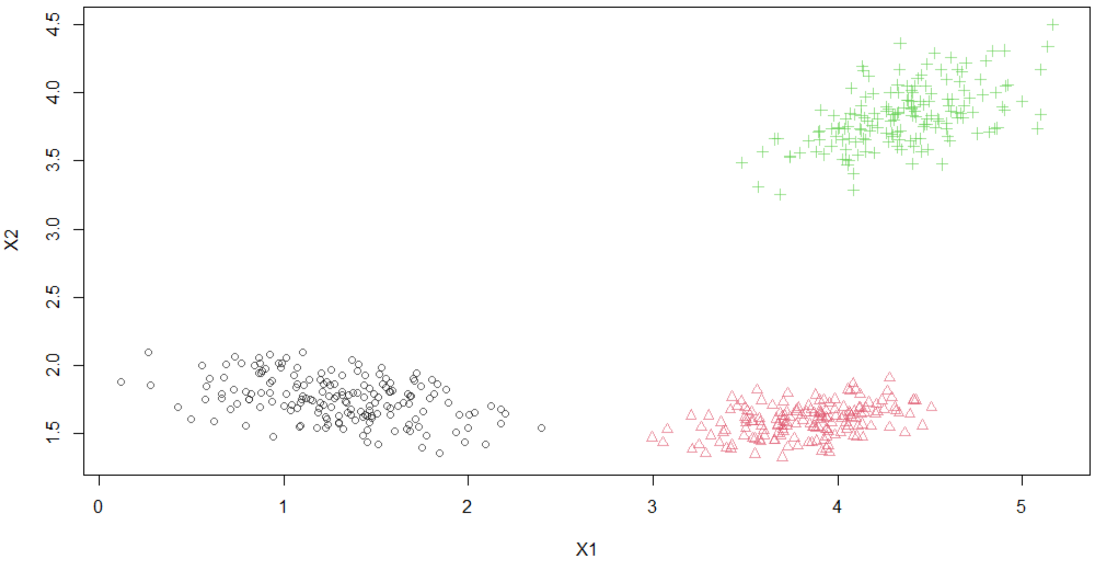


Screenshots at different instances

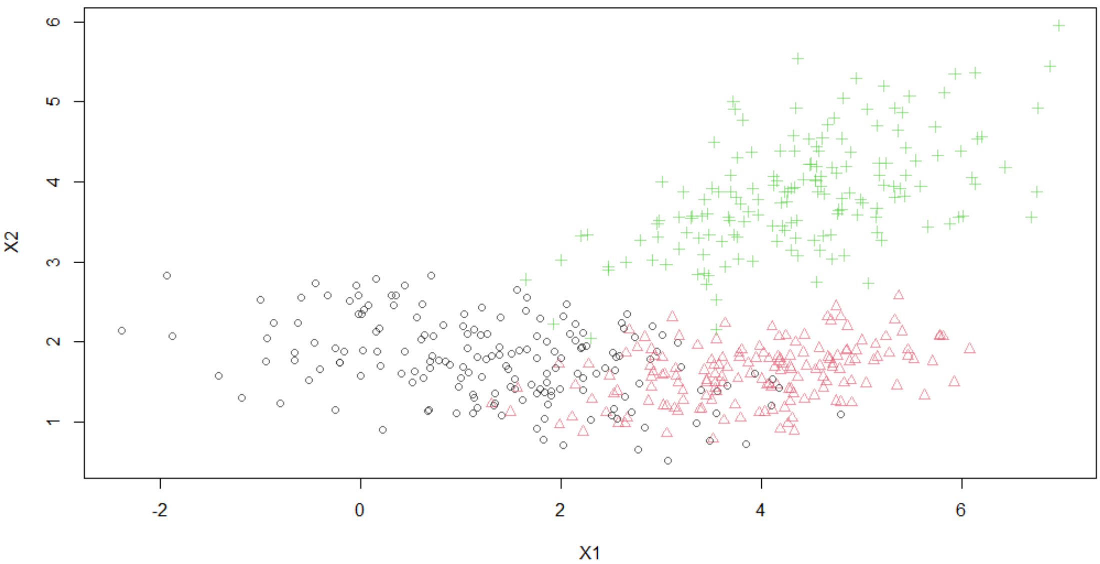
Outlier generating data streams

```
> head(p)
         X1        X2
1 0.2109460 0.3513839
2 0.7703358 0.7727423
3 0.9384046 0.8509380
4 0.3511577 0.3456761
5 0.6941820 0.3020629
6 0.7031516 0.2835118
> out_marks <- attr(p, "outlier")
> sum(out_marks)
[1] 4
> which(out_marks)
[1] 2250 2535 4647 4919
```

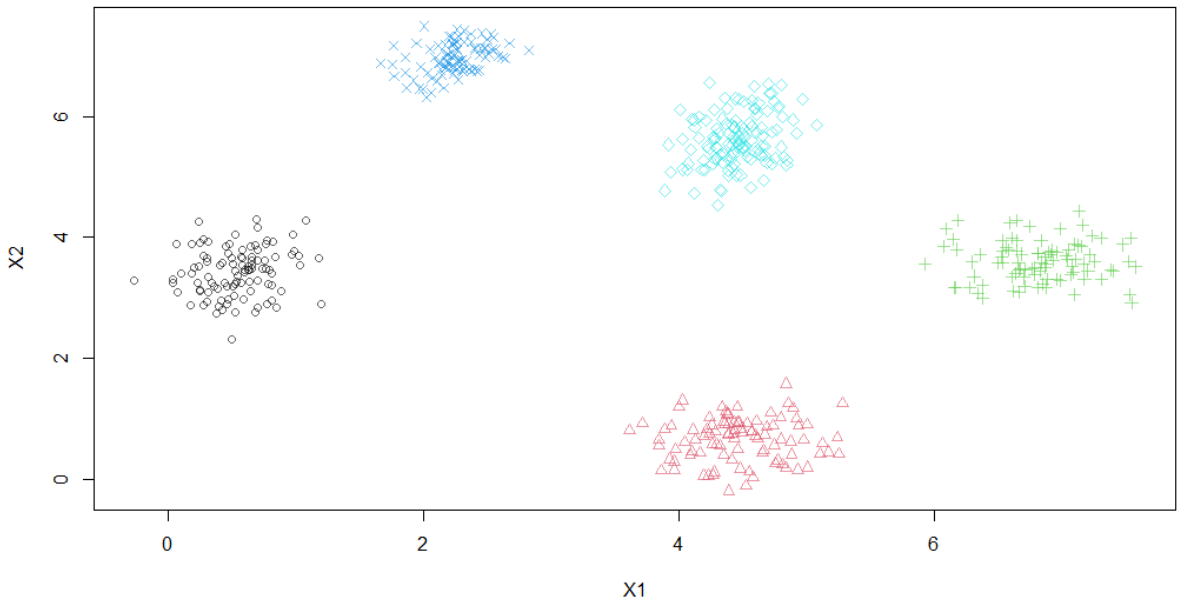
Advanced statistical data streams



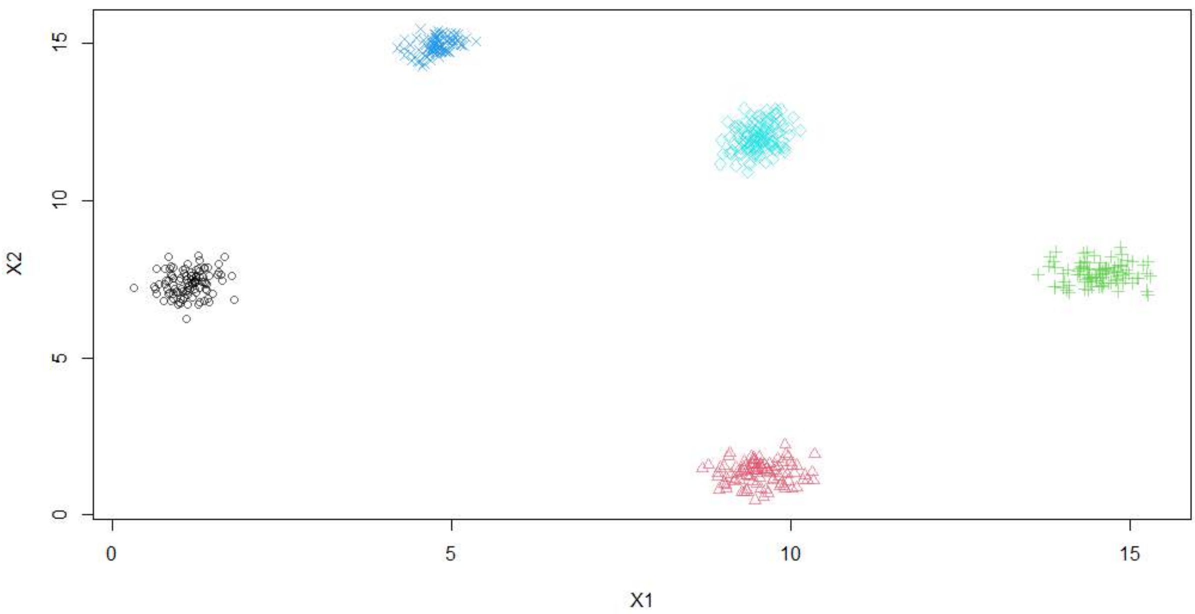
Low Variance Limit



High Variance Limit



Stream 1



Stream 2

Interpretation :

Interpretation #Q2

The seed generator is set to get reproducible results, generates random numbers.

DSD_Gaussians is a data stream generator that produces a data stream with a mixture of static Gaussians. The two parameters used 'K' and 'd' denote the number of clusters and the number of dimensions respectively. As we can see from the plot, there are 3 static gaussian clusters and it is a 2-D plot.

DSC_DStream implements the D-Stream data clustering algorithm. 'gridsize' denotes the size of grid cells and 'Cm' denotes density threshold used to detect dense grids as a proportion of the average expected density (Cm >1). Thus, we have obtained 10 grids of size 0.1 whose density is 1.2.

DSC_Kmeans implements K-means algorithm for reclustering a set of micro-clusters. The parameter 'K' denotes the number of clusters. As we can see from the graph, three new clusters have been formed, denoted by '+'.

DSD_BarsAndGaussians creates the shape of two bars and two Gaussian clusters with different density. 'angle' is the rotation in degrees. As we can see, the two bars and two gaussian clusters have been formed at an angle of 45°, all with different density.

DSD_mlbenchData provides stream interface for datasets from the mlbench package. In our code, we have used the dataset 'Shuttle'. Other datasets that can be accessed include 'Glass', 'DNA', 'Ionosphere', 'Sonar', 'Vowel' to name a few. The dataset stream is then visualized. The 'Shuttle' dataset contains 9 attributes, all of which are numerical, first one being time and last column being class with 7 levels.

DSD_mlbenchGenerator is a data stream generator class that interfaces data generators found in mlbench.

We have used 'Cassini'. The inputs of the Cassini problem are uniformly distributed on 2D space within 3 structures. The 2 external structures are banana-shaped, middle structure being circle.

DSD_Target is a data stream generator that generates a data stream in the shape of a target. It has a single Gaussian cluster in the center and a ring that surrounds it.

DSD_UniformNoise produces uniform noise in a d-dimensional unit (hyper) cube.

In the first plot, a 2D uniform noise is produced. In the second plot, 3 2-D 3x3 matrices with the given ranges of [0,1], [0,5] and [0,10] with uniform noise are produced.

Next, using DSD_Gaussians, data stream with a mixture of static Gaussians is generated. The data streams are plotted, with the noise points plotted using gray dots. The second graph visualizes the same data stream using its first two principal components

DSD_Benchmark generates several dynamic streams intended to be benchmarks to compare data stream clustering algorithms.

Library 'animation' and function 'animate_data' is used to generate an animation of data stream using the generated streams of DSD_Benchmark. This animation can be replayed or can be saved as an animation embedded in a HTML doc or an image GIF.

Next part uses DSD_Gaussians to generate data streams and identify outliers. We have generated a data stream of 10,000 points, of which 6 are displayed using head(). We then obtain the number of outliers and the SNo. of the outliers.

In the Advanced statistical data streams, we can see that as the Variance limit increases, the clusters overlap.

Thus, we use the statistical distance 'Maha lanobis' to separate the clusters, i.e., sufficiently spaced so as to prevent them from overlapping.

--------------------------------------Thank you---------------------------------------