final_logr.R

shayantanbanerjee

Mon Feb 26 11:29:04 2018

```
#Assume the data is stored in a dataframe called noise
library(signal)
```

```
##
## Attaching package: 'signal'
```

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

```
library(prospectr)
```

```
## Loading required package: RcppArmadillo
```

```
##
## Attaching package: 'prospectr'
```

```
## The following object is masked from 'package:signal':
##
##     resample
```

```
library(Rlof)
```

```
## Loading required package: doParallel
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
noise=read.table("/media/shayantanbanerjee/disk1/gsoc/logrr.txt",sep="\t",header=TRUE
)
#some basic information
summary(noise)
```

```
##       chr            start                 end             testSample1
##  Min.   :1.000   Min.   :        0   Min.   :    29999   Min.   :-Inf
##  1st Qu.:1.000   1st Qu.: 58402500   1st Qu.: 58432499   1st Qu.:   0
##  Median :2.000   Median :118815000   Median :118844999   Median :   0
##  Mean   :1.513   Mean   :123038738   Mean   :123068737   Mean   :-Inf
##  3rd Qu.:2.000   3rd Qu.:187927500   3rd Qu.:187957499   3rd Qu.:   0
##  Max.   :2.000   Max.   :249240000   Max.   :249269999   Max.   :   2
##   testSample2
##  Min.   : -1
##  1st Qu.:  0
##  Median :  0
##  Mean   :Inf
##  3rd Qu.:  0
##  Max.   :Inf
```

```
which(noise$testSample1==-Inf)
```

```
## [1] 428
```

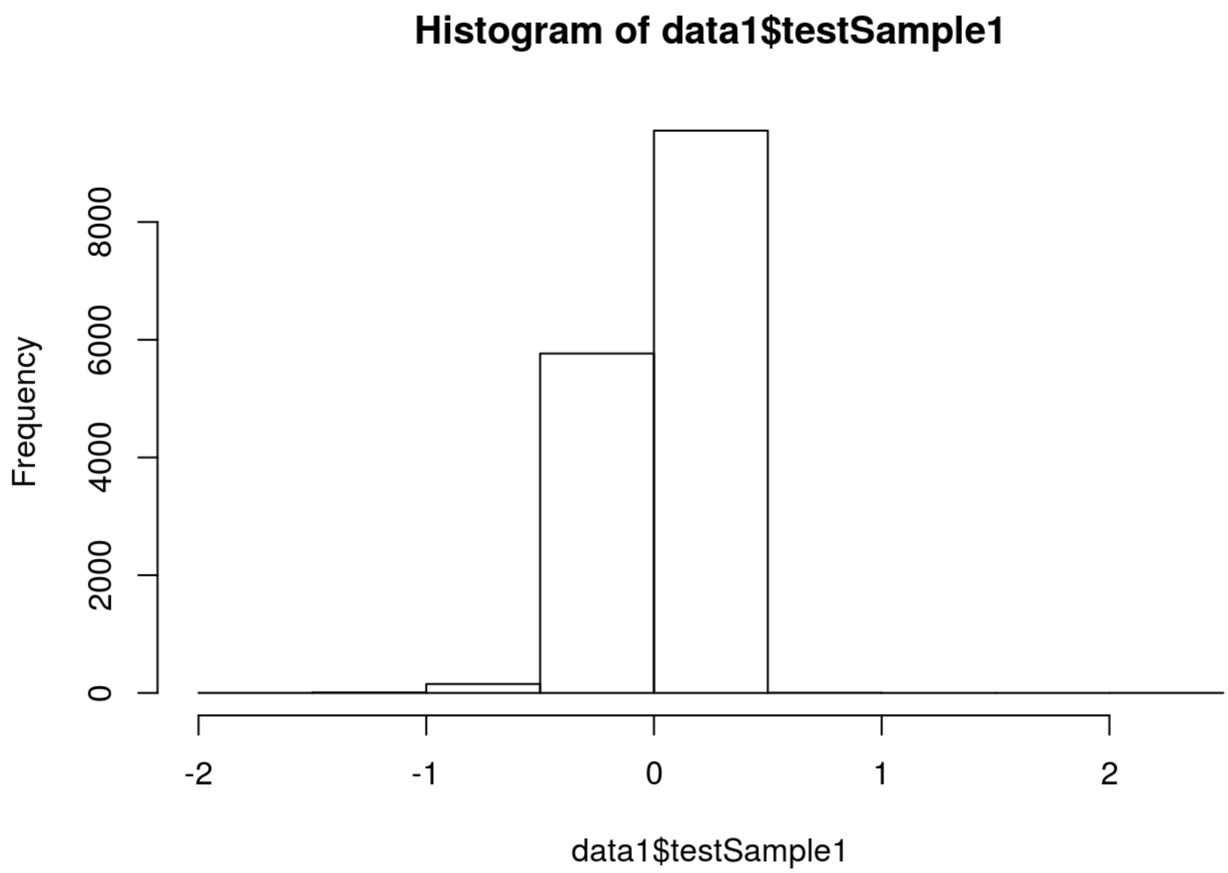
```
which(noise$testSample2==Inf)
```

```
## [1] 4229
```

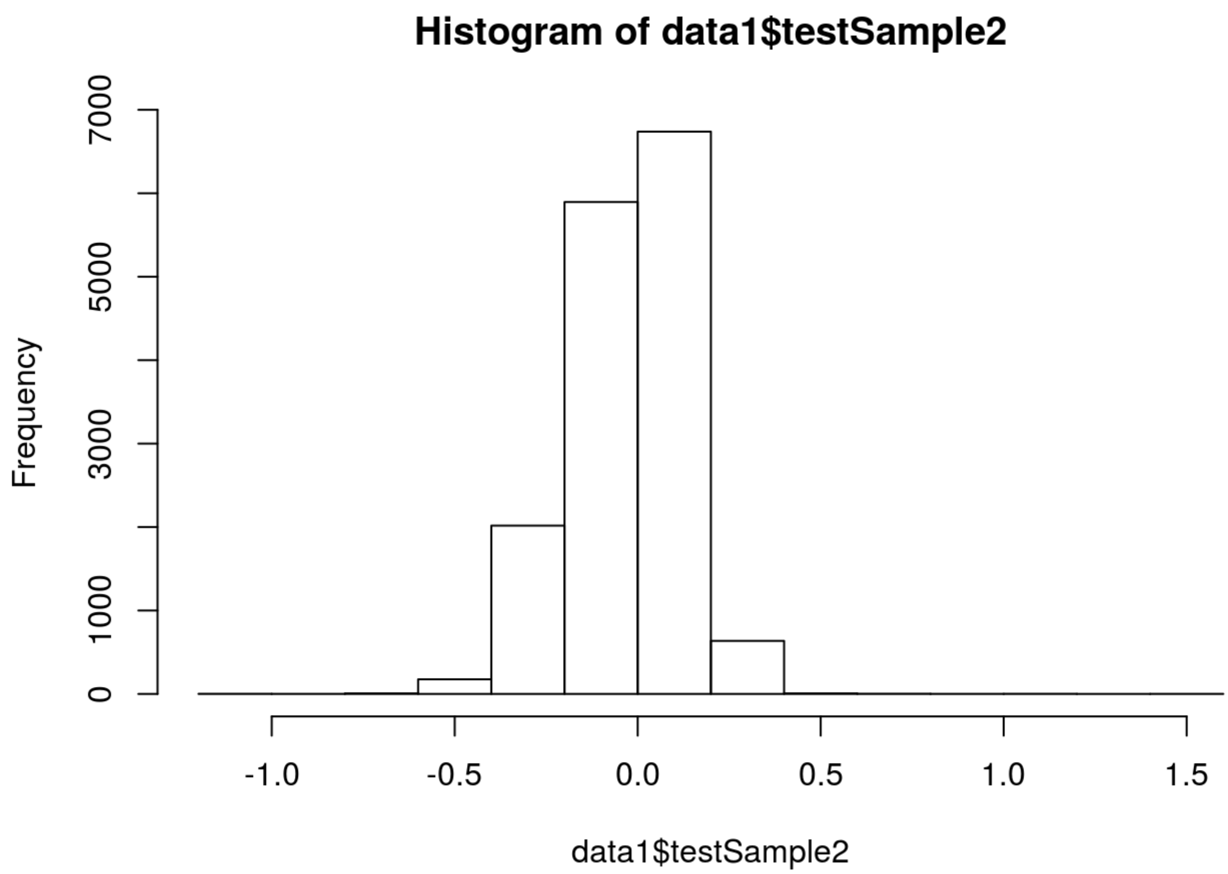
```
#Ratios can't be infinite
#When x is very large, log(x) attains infinity, since we can't get infinite read det
h, we don't conisder these rows
data1=noise[-c(428,4229),]
dim(data1)
```

```
## [1] 15486     5
```

```
#Distribution of each sample
hist(data1$testSample1,breaks=10)
```




```
hist(data1$testSample2,breaks=10)
```



```
#checking for positive and negative values
length(which(data1$testSample1>0))
```

```
## [1] 9558
```

```
length(which(data1$testSample1<=0))
```

```
## [1] 5928
```

```
#Correlation between two samples
cor(data1$testSample1,data1$testSample2)
```

```
## [1] -0.6616517
```

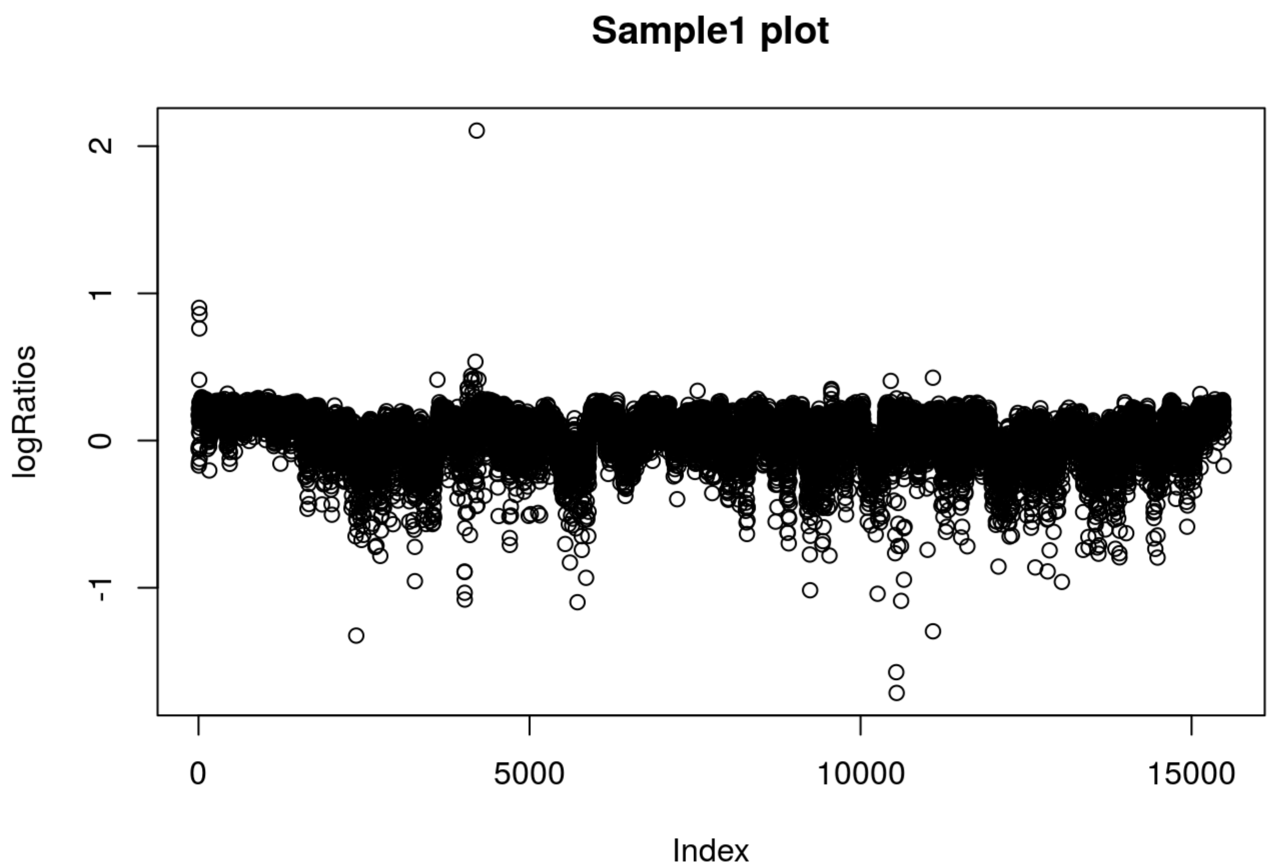
```
#This gives the cross xorrelation between two signals.There is a strong negative corr
elation between the two signals
#According to the documentation SCoNEs rational is the logR ratio
#between normal and tumor read depth is asignal which is composed of mixture of sever
al gaussian for each copy number state and somehow for technical and artifical noise.
#Thus for sample 1, there are more number of data points with positive log ratios(CAS
ES WHERE TUMOUR READ DEPTH IS GREATER THAN NORMAL)
length(which(data1$testSample2>0))
```

```
## [1] 7389
```

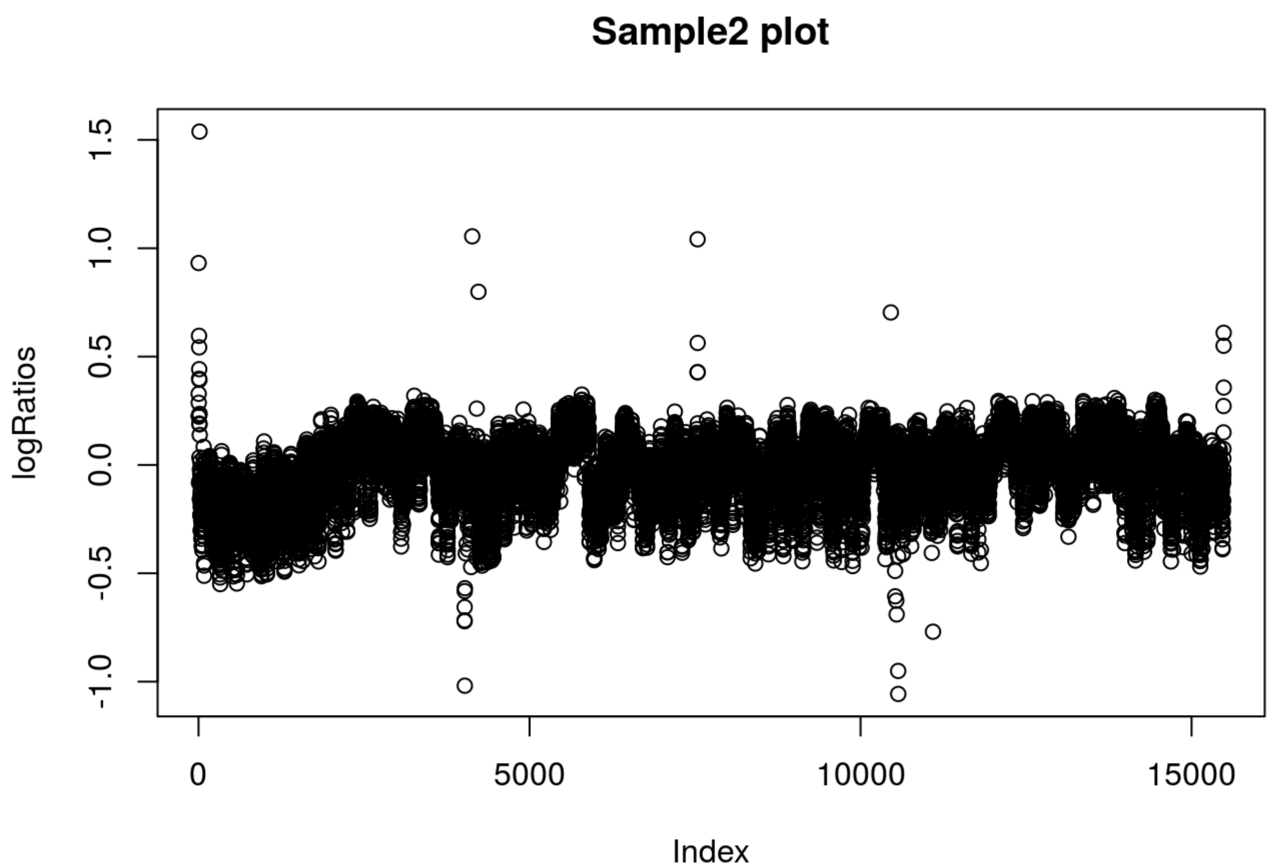
```
length(which(data1$testSample2<=0))
```

```
## [1] 8097
```

```
#Thus for sample 2, there are more number of data points with negative log ratios(CAS
ES WHERE TUMOUR READ DEPTH IS LESSER THAN NORMAL)
#plotting individual sample signals to look for noise
plot(data1$testSample1,ylab="logRatios",main="Sample1 plot")
```



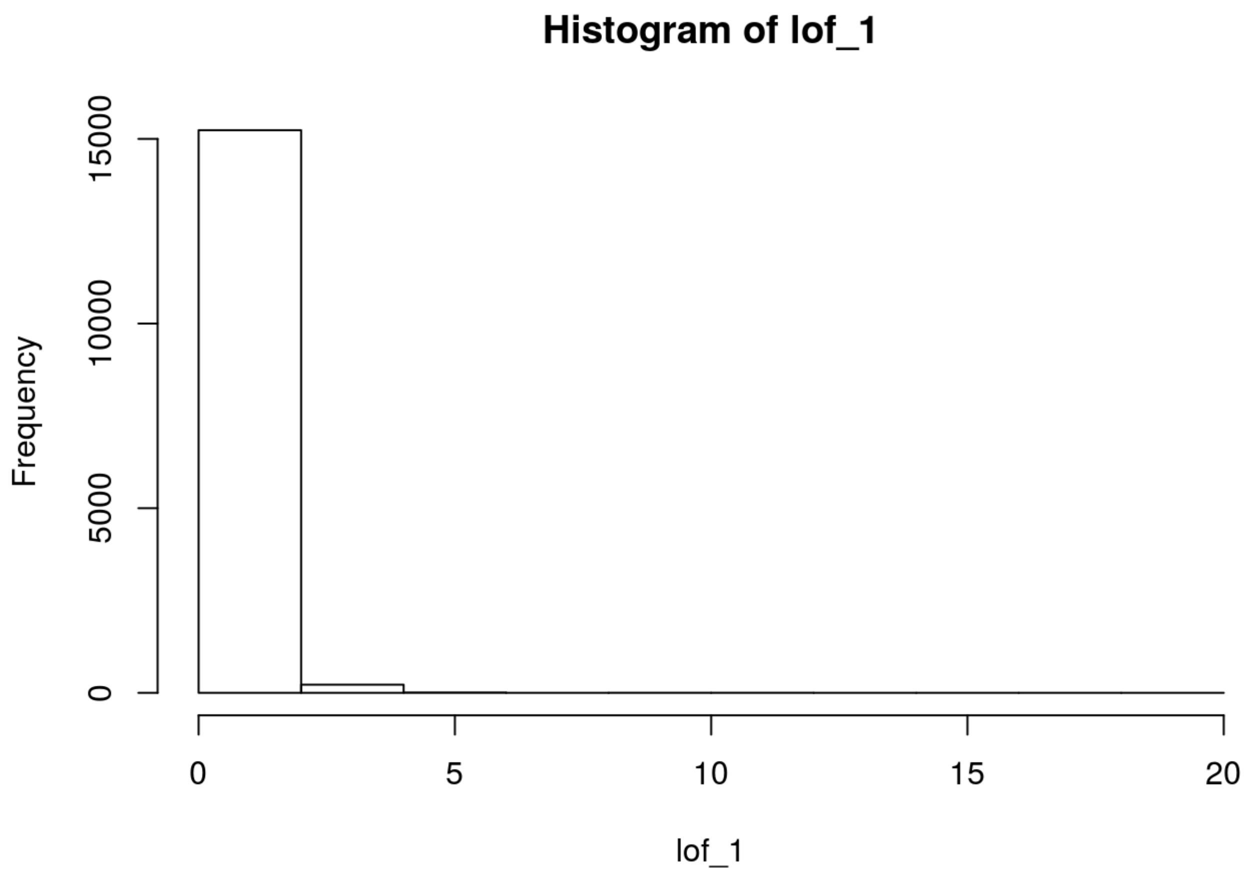

```
plot(data1$testSample2,ylab="logRatios",main="Sample2 plot")
```



```
#Below listed are two different methods for tackling noise
##METHOD 1 (OUTLIER DETECTION)
#For both the plots there are a lot of points which looks like outliers. We must desi
gn an efficient way to identify those
#For the vast amount of times, outliers are noise, which are points that are very dif
ferent from the true data
#LOF compares the local density of an point to the local densities of its neighbors.
 It is an unsupervised approach and no labelling is necessary
#Points that have a substantially lower density than their neighbors are considered o
utliers.
#Calling lof() on unique values only
lof_1=lof(unique(data1$testSample1),k=5) #neighborhood size considered is 5 for sampl
e 1
lof_2=lof(unique(data1$testSample2),k=5) #neighborhood size considered is 5 for sampl
e 2
#distribution of outlier factors
summary(lof_1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8163  0.9811  1.0300  1.1030  1.1110 18.9400
```

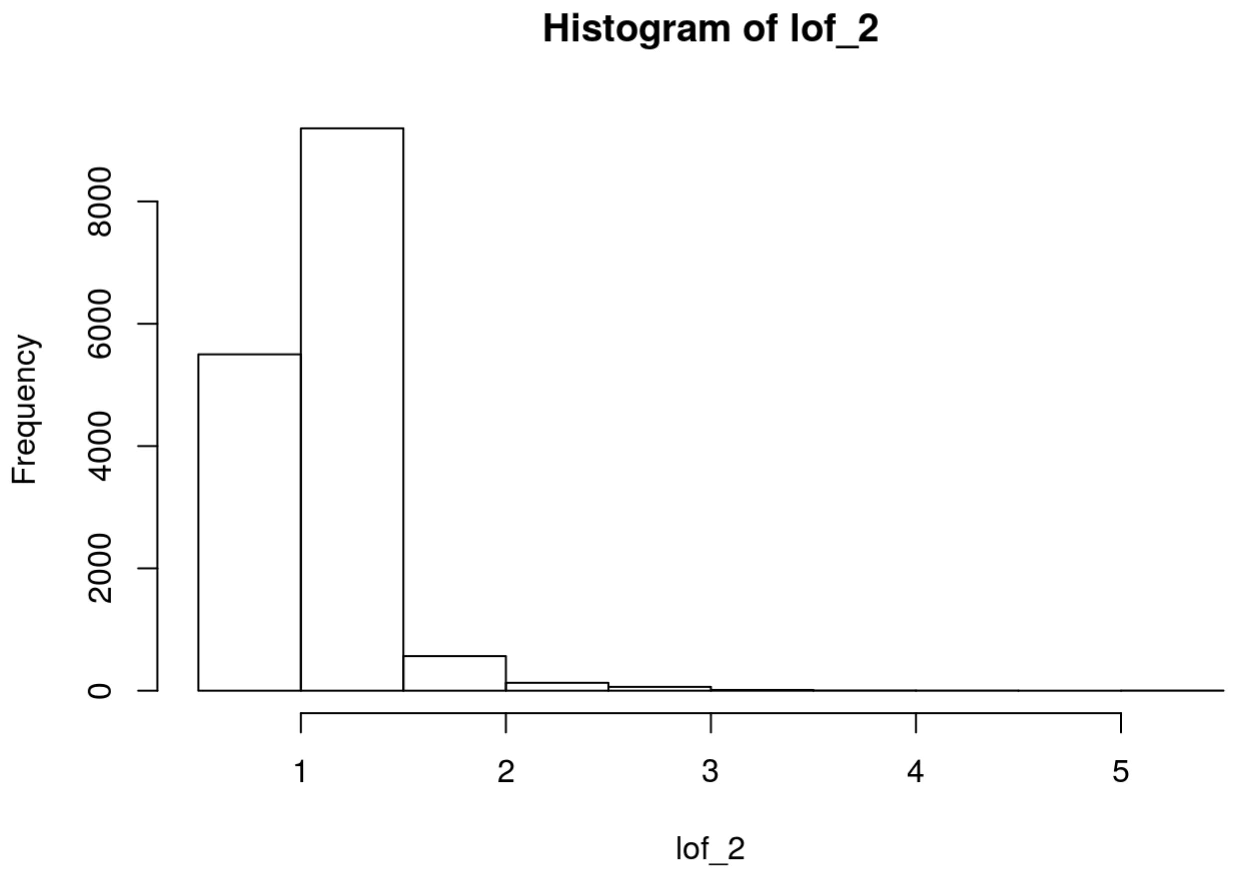
```
hist(lof_1, breaks=10)
```



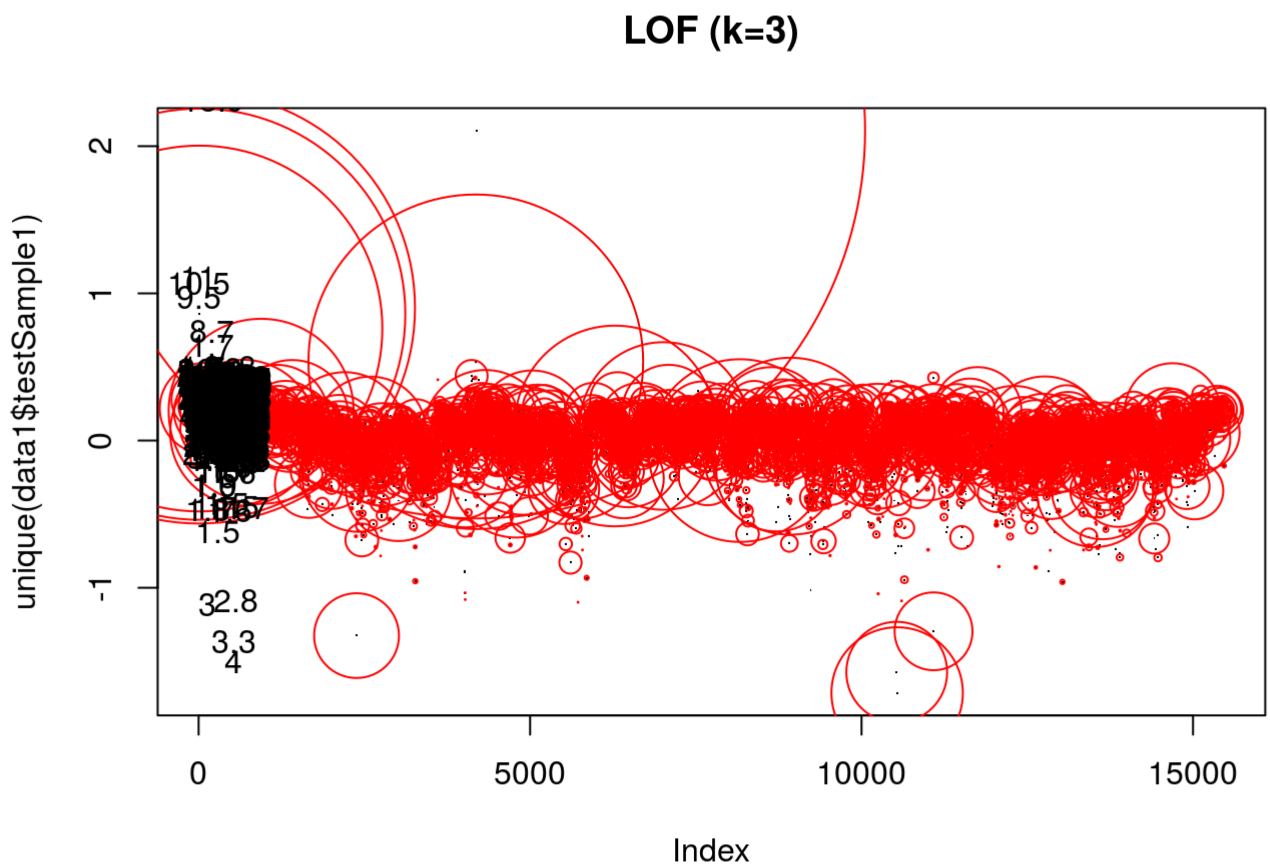

```
summary(lof_2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8211  0.9811  1.0280  1.0970  1.1120  5.4780
```

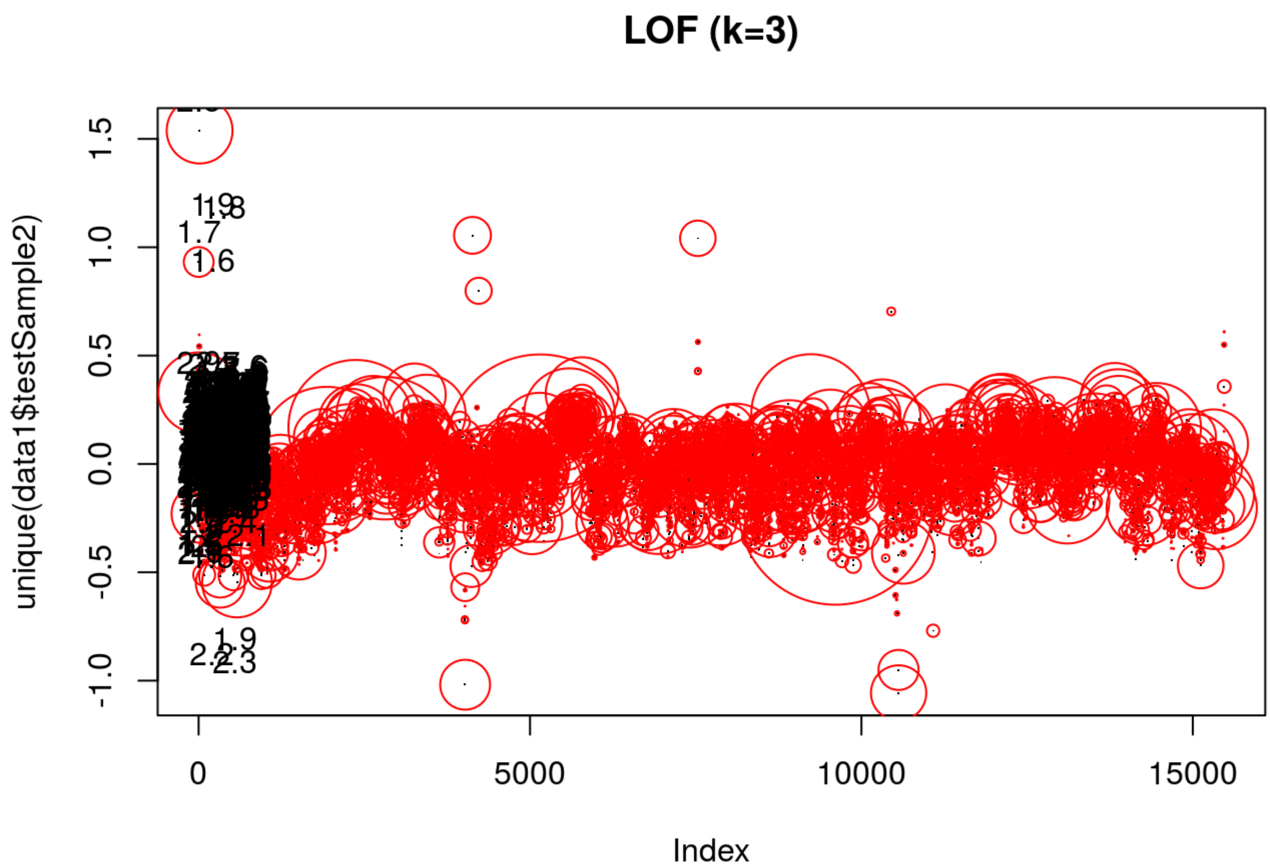
```
hist(lof_2, breaks=10)
```




```
#Both the means are approximately centered around 1
#An lof value close to 1 indicates that an object is comparable to its neighbours and
 are not outliers
#A value below {\displaystyle 1} 1 indicates a denser region (which would be an inlie
r)
#while values significantly larger than 1 indicate outliers.

#although not so informative, the below plots show point size proportional to lof
### point size is proportional to LOF
plot(unique(data1$testSample1), pch = ".", main = "LOF (k=3)")
points(unique(data1$testSample1), cex = (lof_1-1)*3, pch = 1, col="red")
text(unique(data1$testSample1)[lof_1>1.5], labels = round(lof_1, 1)[lof_1>1.5], pos =
 3)
```




```
plot(unique(data1$testSample2), pch = ".", main = "LOF (k=3)")
points(unique(data1$testSample2), cex = (lof_2-1)*3, pch = 1, col="red")
text(unique(data1$testSample2)[lof_2>1.5], labels = round(lof_2, 1)[lof_2>1.5], pos =
 3)
```



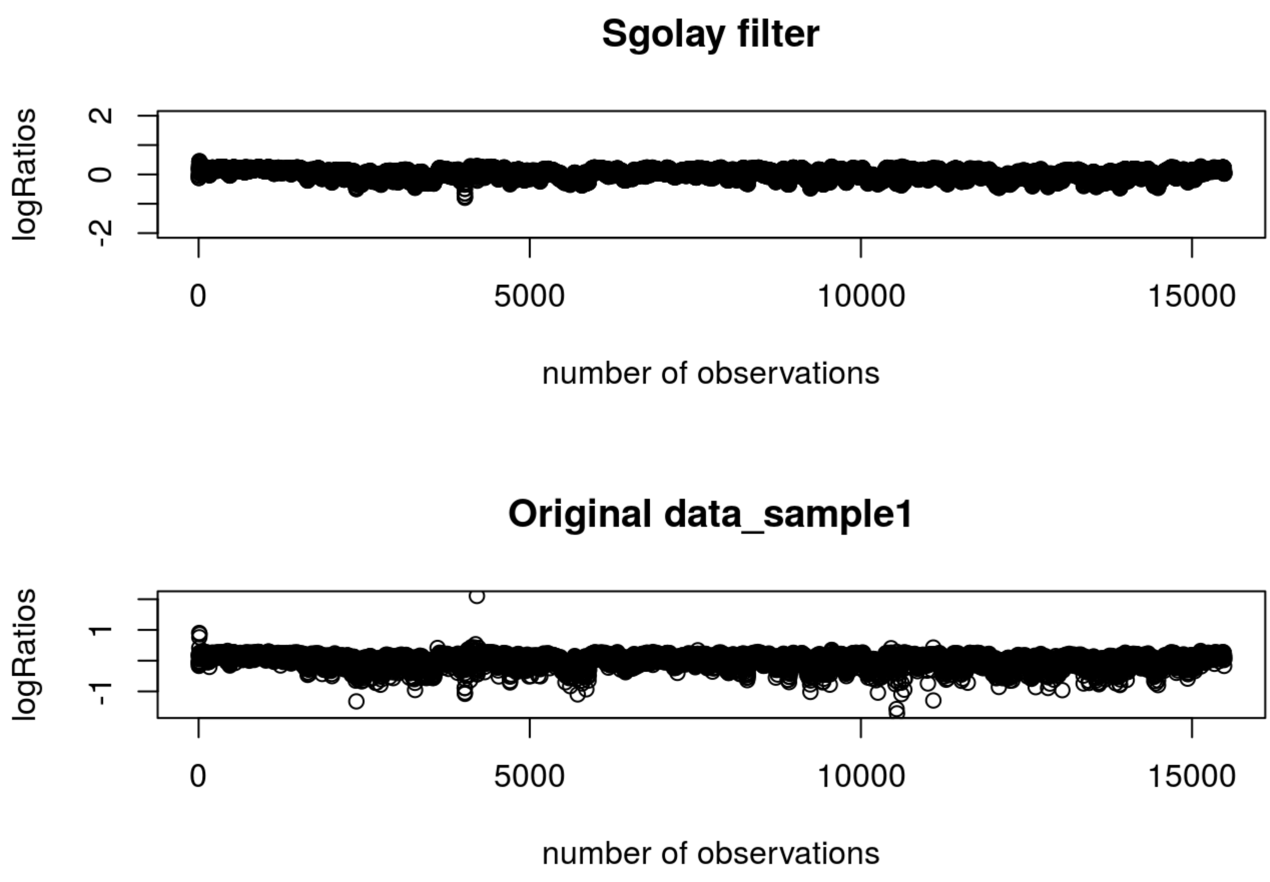

```
#Removing outliers
index_1=which(lof_1>=1.5)
index_2=which(lof_2>=1.5)
length(index_1)
```

```
## [1] 776
```

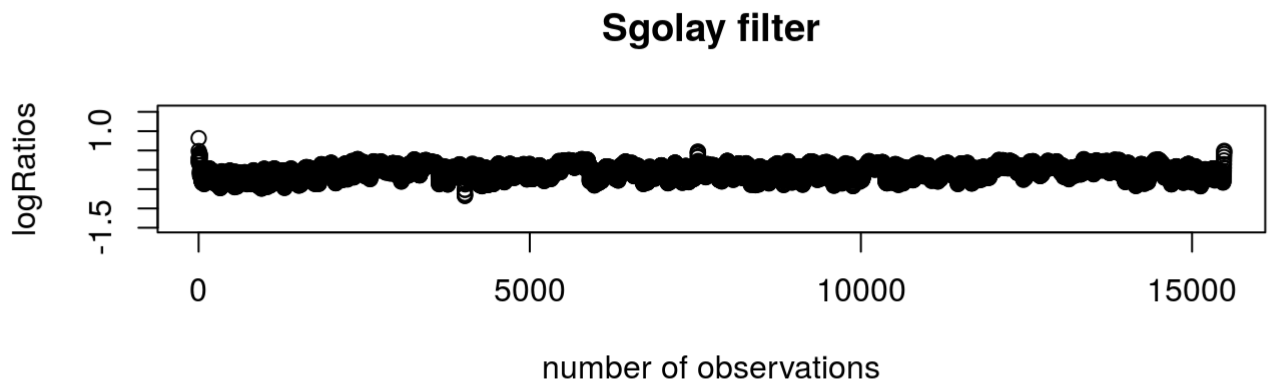
```
length(index_2)
```

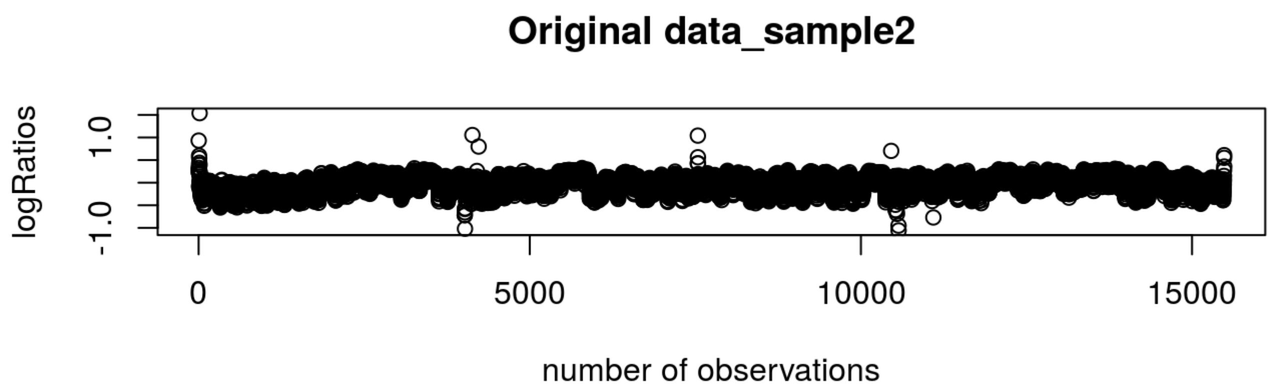
```
## [1] 779
```

```
#Around 776 and 779 points for sample1 and sample2 respectively are outliers
#Removing those points
new_test_sample1=data1$testSample1[-index_1]
new_test_sample2=data1$testSample2[-index_2]

#Final preprocessed signal for both samples are stored in new_test_sample1 and new_te
st_sample2
#This method using local outlier factor has to be further tuned to get the best signa
l to noise ratio. This is just a naive implementation

#METHOD 2
#A Savitzky–Golay filter is a digital filter that can be applied to a set of digital
 data points for the purpose of smoothing the data, that is,
#to increase the signal-to-noise ratio without greatly distorting the signal.
#The package signal needs to be installed
library(signal)
sg_1<-sgolay(p=3,n=13,m=0)
df_1<-filter(sg_1,data1$testSample1)
layout(matrix(c(1,2),2,1,byrow=TRUE))
plot(df_1,ylim=c(-2,2),xlab="number of observations",ylab="logRatios",main="Sgolay fi
lter")
plot(data1$testSample1,xlab="number of observations",ylab="logRatios",main="Original
 data_sample1")
```

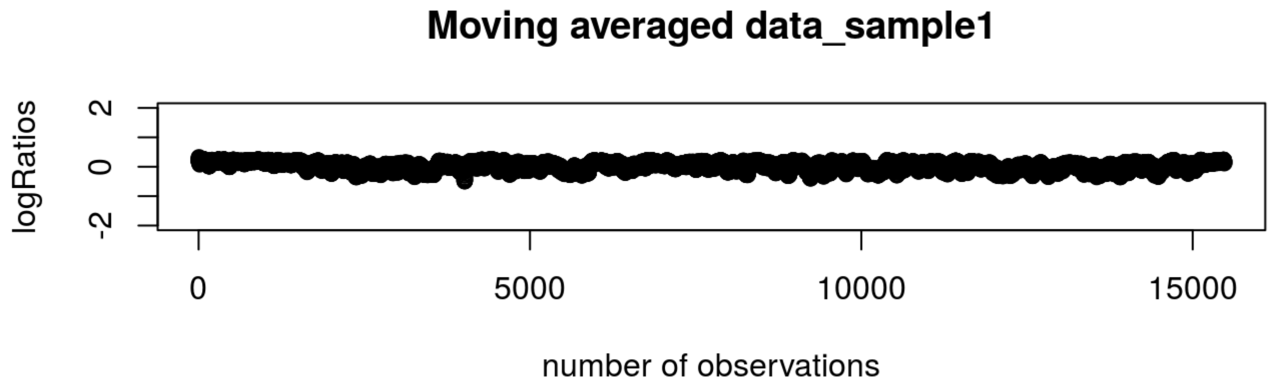


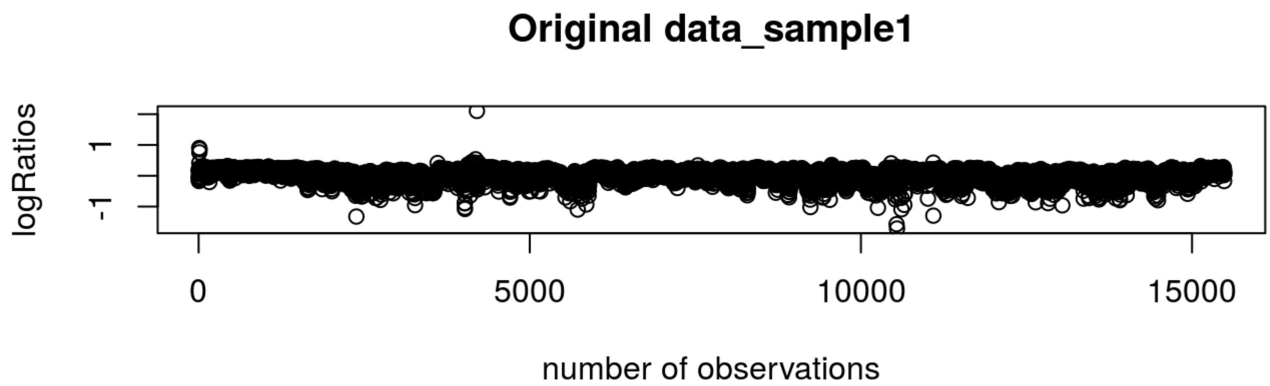
```
sg_2<-sgolay(p=3,n=13,m=0)
df_2<-filter(sg_2,data1$testSample2)
layout(matrix(c(1,2),2,1,byrow=TRUE))
plot(df_2,ylim=c(-1.5,1.54),xlab="number of observations",ylab="logRatios",main="Sgol
ay filter")
plot(data1$testSample2,xlab="number of observations",ylab="logRatios",main="Original
 data_sample2")
```



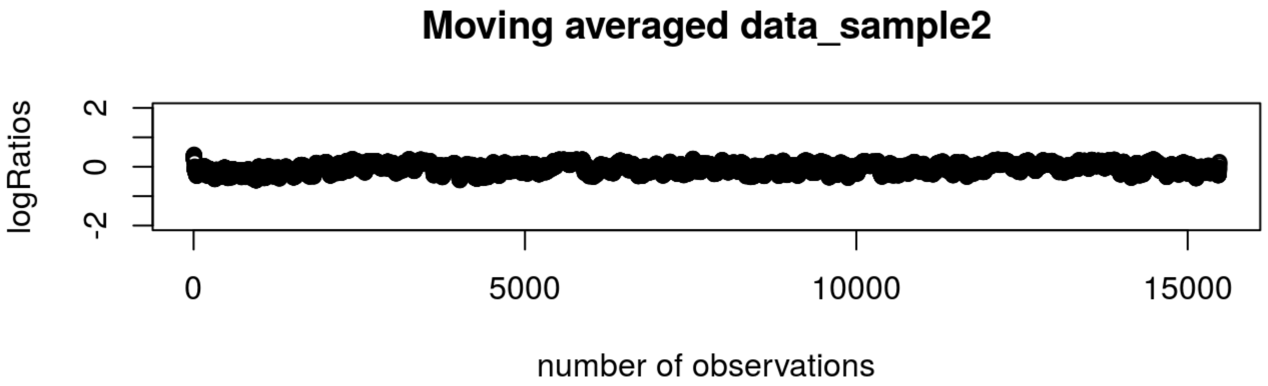


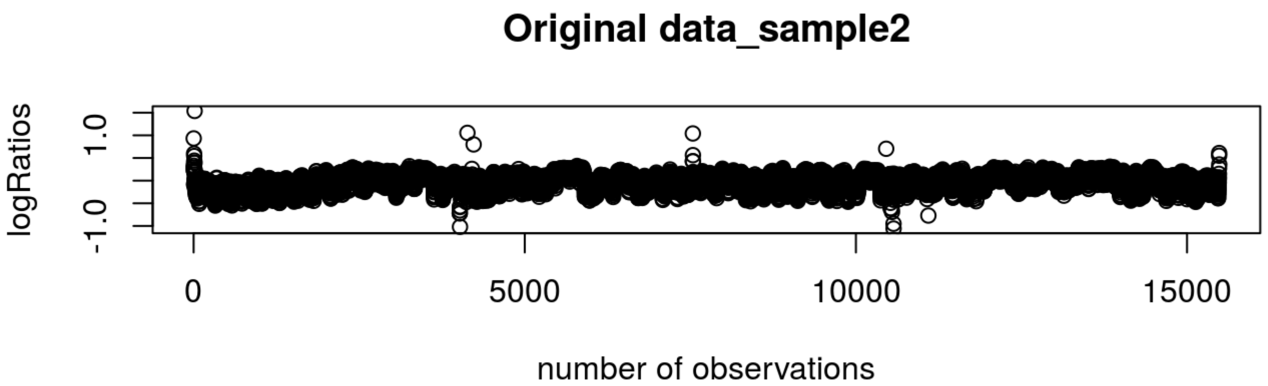

```
#Method 3
#Moving average method
#This method takes M samples of input at a time and take the average of those M-sampl
es and produces a single output point.
#Very handy for removal of unwanted noise from signal
mv_1=movav(data1$testSample1,w=11)
plot(mv_1,xlab="number of observations",ylab="logRatios",ylim=c(-2,2),main="Moving av
eraged data_sample1")
plot(data1$testSample1,xlab="number of observations",ylab="logRatios",main="Original
 data_sample1")
```






```
mv_2=movav(data1$testSample2,w=11)
plot(mv_2,xlab="number of observations",ylab="logRatios",ylim=c(-2,2),main="Moving av
eraged data_sample2")
plot(data1$testSample2,xlab="number of observations",ylab="logRatios",main="Original
 data_sample2")
```


Moving averaged data_sample2
logRatios
2
0
-2
0
5000
10000
15000
number of observations


Original data_sample2
logRatios
1.0
-1.0
0
5000
10000
15000
number of observations