Anomaly Detection Using CUSUM Charts

There are several functions that are sourced and called in this anomaly detection code. They are included here for reference:

detector_function

```
#this function takes a time series of length of at least 75
#and looks for anomalies using the statistical process control chart,
the CUSUM chart
#TEMPORARY: the input should not be much longer than 365
#FINAL VERSION: the input can be any length longer than 74


detector <- function(time_series){
      #temporary version
      require(qcc)
      require(ggplot2)
      n<-length(time_series)
      stopifnot(n>=75)
      source("training_data_finder_function.R")
      training_data_object <- training_data_finder(time_series)
      if (length(training_data_object$training_data) == 0) {
            print("This method cannot be applied to this particular
time series.")
      }
      else {

            training_data<-training_data_object$training_data
            #plotting the time_series and the training set by
            #creating a data frame that is to be plotted using ggplot2
            start_val<-training_data_object$segment_starting_point
            segment_length<-training_data_object$segment_length
            end_val<-segment_length + start_val-1
            df <- data.frame(
```

```
                    index = 1:n,
                    y = time_series,
                    col = c(rep("black", (start_val-1)),
                            rep("red", segment_length),
                            rep("black", (n-end_val)))
              )
              p<-ggplot(df, aes(x=index, y=y)) +
                    geom_line(aes(colour=col, group=1)) +
                    scale_colour_identity()
              #creating the CUSUM chart
              x_bar<-mean(training_data)
              sigma<-sd(training_data)
              k<-training_data_object$k
              H<-training_data_object$H
              c<-cusum(time_series,sizes=1, center=x_bar, std.dev=sigma,
                    decision.interval=H, se.shift=k,restore.par = TRUE)
              return(p)

      }
}
```

training_data_finder_function

```
#this function takes a vector of daily measurements
#of length of at least 75
#and finds a stretch of most normally distributed sub-segment

training_data_finder <- function(sequence){

      source("normality_finder_function.R")
      source("checker_function.R")
      source("parameter_finder_function.R")
      n <-length(sequence)
      stopifnot(n>=75)
      segment_found<-FALSE
```

```
      no_solution<-FALSE
      #TEMPORARY:creating a data frame that will keep track of the
statistics for each segment
      stats_df<-data.frame()

      row<-c(-1, rep(1000000,7)) #non-sensical dummy values

      segment_length <- n %/% 3

      while(((!segment_found ) & (!no_solution))){

            for (i in (1:(n-segment_length+1))){
                  x<-sequence[i: (i+segment_length-1)]
                  if (normality_finder(x)$p_value >= 0.05){
                        skew<-normality_finder(x)$skew
                        kurt<-normality_finder(x)$kurt
                        p_value<-normality_finder(x)$p_value
                        cusum_parameters<-parameter_finder(x)
                        repr <- abs(skew) + abs(3-kurt)+abs(1-p_value)
+cusum_parameters$H*100 + cusum_parameters$k*100
                        new_row<-c(i,segment_length,skew,kurt,p_value,
                           cusum_parameters$k, cusum_parameter$H,repr)
                        stats_df<-rbind(stats_df,new_row)
                        if (repr < row[8]) row<-new_row



                  }
            }
            if (checker(row)) {
                  segment_found<-TRUE
            }
            else if (segment_length > 25) segment_length<-
segment_length-1
            else {
                  print("Couldn't find a usable training set, cannot
proceed with searching for anomalies.")
                  no_solution<-TRUE
            }
```

```
      }
      names(stats_df)<-c("segment_starting_point", "segment_length",
"skewness","kurtosis", "Shapiro_test_p_value", "cusum_k",
"cusum_H","summarizing_value")
      if (no_solution) {
            training_data<-NULL
            k<-NULL
            H<-NULL
            segment_starting_point<-NULL
            segment_length<-NULL
            summarizing_value<-NULL
      }
      else {

            training_data<-sequence[row[1]:(row[1]+row[2]-1)]
            k<-row[6]
            H<-row[7]
            segment_starting_point<-row[1]
            segment_length<-row[2]
            summarizing_value<-row[8]
            start_val<-segment_starting_point
            end_val<-segment_length + start_val-1
            #two_color_plotter(sequence, segment_length, start_value,
end_value)

      }
      return(list(df=stats_df, training_data=training_data, k=k, H=H,
                  segment_starting_point= segment_starting_point,
segment_length=segment_length,
                  summarizing_value=summarizing_value))


}
```

normality_finder_function

```
##Given a vector x, this function finds its Skewness, Kurtosis
##and Shapiro-Wilk normality test p-value

normality_finder<-function(x){
      require(moments)
      return(list(skew=skewness(x), kurt=kurtosis(x),
p_value=shapiro.test(x)$p.value))
}
```

parameter_finder_function

```
#This function takes a time series - that has (hopefully) a nearly
normal distribution
#and selects the smallest parameters k and H for which the time series
is in statistical
#control
#TEMPORARY VERSION:For now the value of H will remain a constant, H=5,
# only the value of k will increase from its starting value of 3 until
the
#time_series is in statistical control
#FINAL VERSION: either allow H to change its value as well (if it is
needed)
#or remove it
parameter_finder <- function(time_series){
      require(qcc)
      xbar<-mean(time_series)
      sigma<-sd(time_series)
      k<-3
      H<-5
      parameters_found<-FALSE
      while (!parameters_found){
            object<-cusum(time_series,sizes=1, center=xbar,
std.dev=sigma,
                          decision.interval=H, se.shift=k, plot=FALSE)
```

```
            #print(object)
            lower_count<-length(object$violations$lower)
            upper_count<-length(object$violations$upper)
            if ((lower_count==0) & (upper_count==0)) {
                  parameters_found<-TRUE
            }
            else k<-k+1

      }
      return(list(k=k, H=H))
}
```

checker_function

```
checker <- function(x){
      if (x[1]==-1){
            dummy<-FALSE
            return(dummy)
      }
      else {
            dummy<-TRUE
            return(dummy)
      }
}
```

Using Anomaly Detection on the Interconnection Study Data

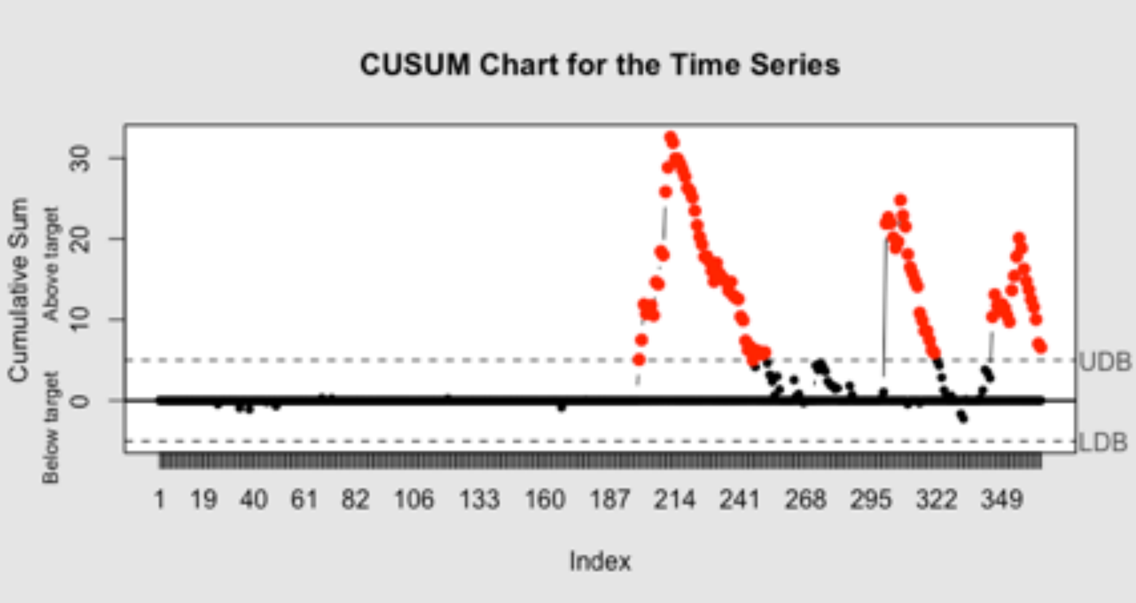
Preparing the Raw Data

```
#this function takes an Interconnection study .csv file name as an
input and
#returns the daily median throughput time-series created from the file

interconn_data_prep<-function(file_name){
      require(dplyr)
      orig_df <- read.csv(file_name)
      names(orig_df)<-c("log_time", "download_thruput")
      orig_df$date <-  as.POSIXlt(as.integer(orig_df$log_time),origin
= "1970-01-01", tz = "GMT")
      head(orig_df)
      orig_df$yearMoDay <- as.character(format(orig_df$date, "%Y %m
%d"))
      orig_df$date<-as.character(orig_df$date)
      daily_df <- group_by(orig_df, yearMoDay)
      daily_df_summary <- summarise(daily_df, count=n(),

median_thruput=median(download_thruput))
      test_data <- daily_df_summary$median_thruput
      return(test_data)
}
```

Reading in the Data and Applying the detector_function to it

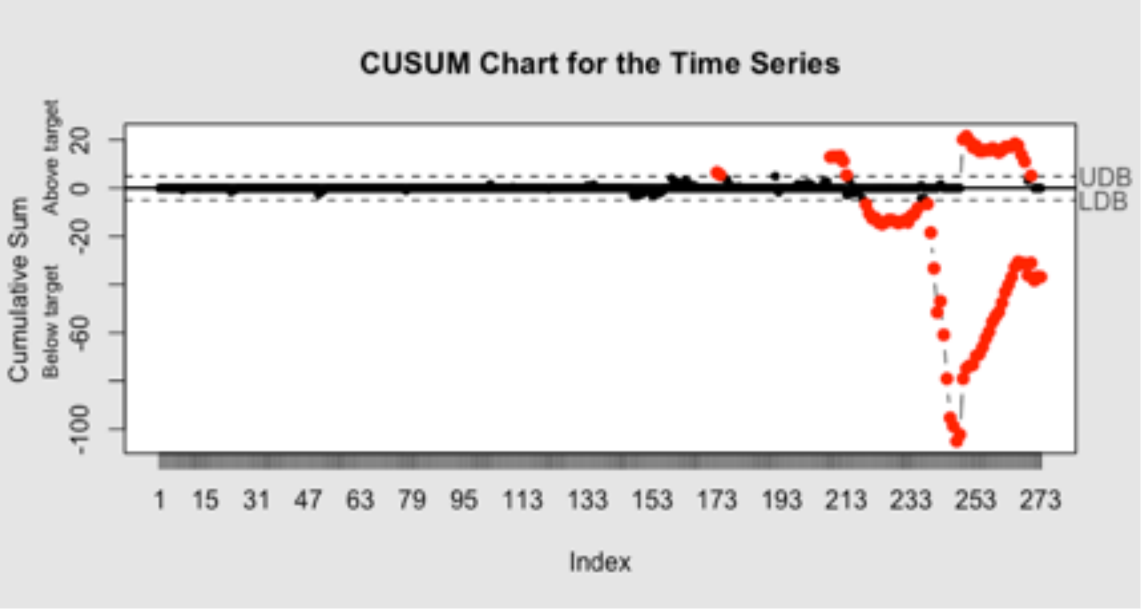
```
setwd("~/Dropbox/outreachy/anomalyDetection")
source("detector_function.R")
source("interconn_data_prep_function.R")
x<-
interconn_data_prep("2013-01-01-000000+365d_sea01_verizon_packet_retra
nsmit_rate-raw.csv")
```

```r
detector(x)
```

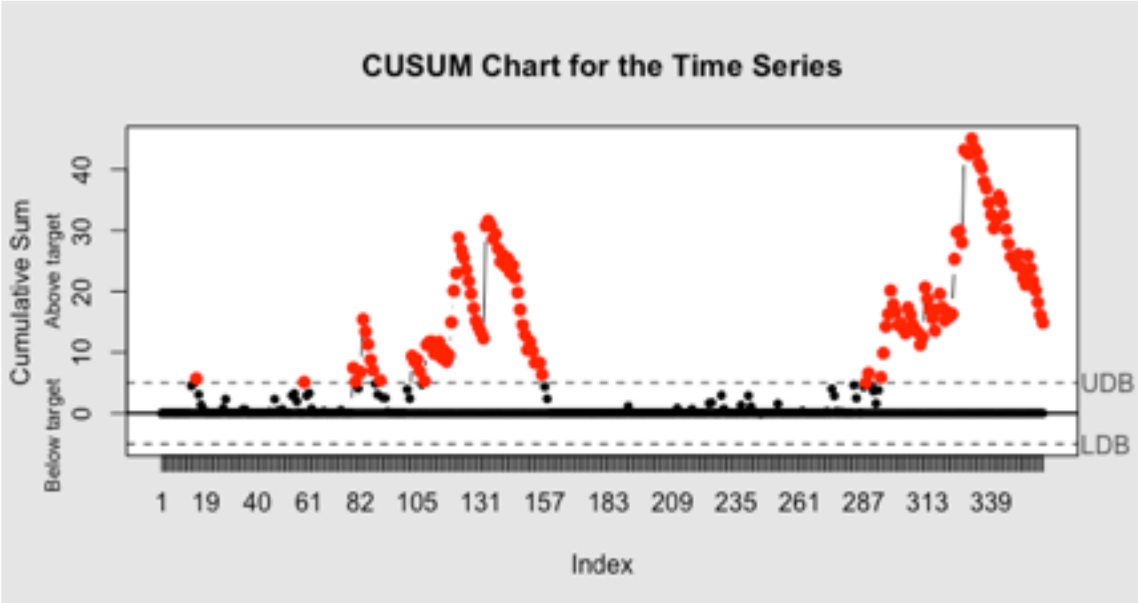

CUSUM Chart for the Time Series
Cumulative Sum
Above target
Below target
30
20
10
0
UDB
LDB
1 19 40 61 82 106 133 160 187 214 241 268 295 322 349
Index


Time Series Plot with Training Data in Red
y
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
index

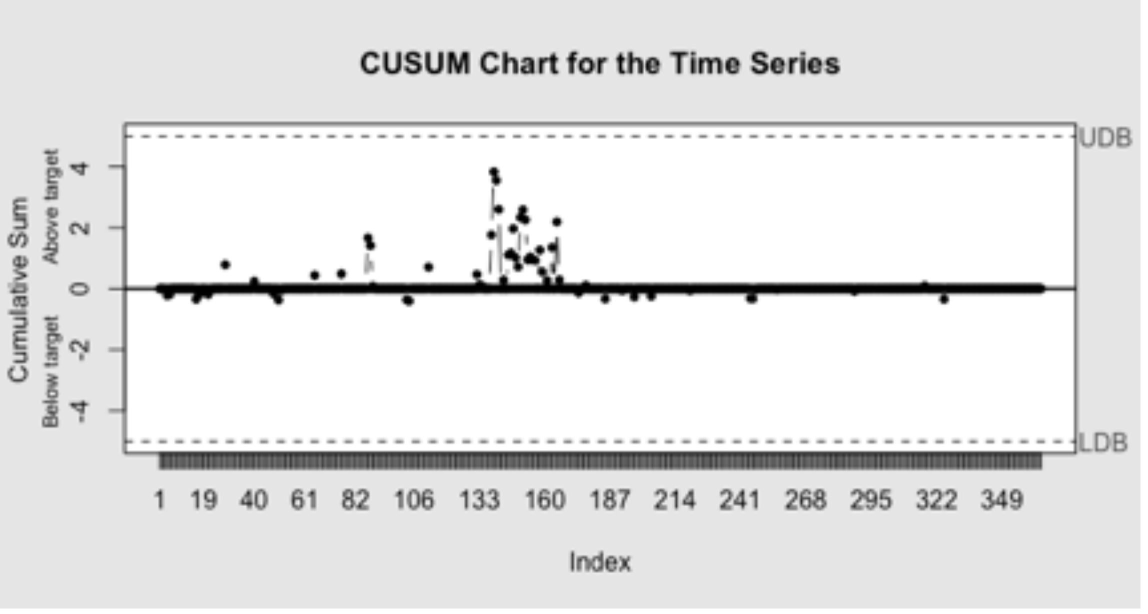
```
x<-
interconn_data_prep("2014-01-01-000000+273d_atl01_comcast_upload_throughput-raw.csv")
detector(x)
```



```
x<-
interconn_data_prep("2013-01-01-000000+365d_nuq02_verizon_upload_throu
ghput-raw.csv")
detector(x)
```



```
x<-
interconn_data_prep("2013-01-01-000000+365d_ord01_verizon_upload_throughput-raw.csv")
detector(x)
```



```
x<-
interconn_data_prep("2012-01-01-000000+366d_lax01_verizon_download_thr
oughput-raw.csv")
detector(x)
```