# Appendix 2 - Enhanced Surveillance in R

The Enhanced Surveillance algorithm originally created in SAS was validated within an R Shiny application. Included here in Appendix 2 is the output of an RMarkdown showing (1) the aggregation tools used to convert child level data into monthly data and (2) the analysis tools used to perform the Enhanced Surveillance algorithm on the moving windows of the data. It is performed on entirely *simulated* data, simply to showcase the process and verify that it is reproducible.

### Application Background

The application code is packaged into a not-yet-released EnhancedSurveillance R package shown here with the dependency suggestions:

library(EnhancedSurveillance)  
#' shiny (>= 1.3.2),  
# lubridate (>= 1.7.4),  
# shinythemes (>= 1.1.2),  
# readxl (>= 1.3.1),  
# dplyr (>= 0.8.1),  
# ggplot2 (>= 3.2.0),  
# plotly (>= 4.9.0),  
# haven (>= 2.1.1)  
  
library(lubridate)   
library(dplyr)

### Child Level Aggregations

The child level aggregations to the proportion of tests above threshold are determined via this rule:

1. A single test per child per calendar year
2. If a child has multiple test within a calendar year
   1. The highest venous test is chosen
      1. If no venous test is available, the lowest non-venous is chosen.

To show a template version of the analysis, the following child level data is *simulated*.

100,000 child tests across 20,000 unique children across 9 years (2010-2019). Assume all children are < 6 years of age when tested. While we know there are seasonal and long-term trends in BLLs they are ignored to showcase the tool. Instead, the month of Dec 2019 is artifically raised to show a spike (60% of tests will be elevated instead of 30%). The first 10 rows of this *simulated* table are shown as well:

set.seed(4)  
childlevel <- data.frame(  
 childID = floor(runif(n = 100000,min = 1,max = 20000)),   
 SpecimenDate = {   
 temp. <- runif(n = 100000,  
 min = as.numeric(as.Date("2010-01-01")),  
 max = as.numeric(as.Date("2019-12-31"))  
 )  
 as.Date(temp., origin = "1970-01-01") # windows default origin   
 },   
 SampleType = sample(x = c("Venous","Capillary"),  
 size = 100000,  
 replace = TRUE))  
  
spike\_index <- childlevel$SpecimenDate > as.Date("2019-11-30")  
  
childlevel$TestResult <- unlist(lapply(  
 X = spike\_index, FUN = function(x){   
 if(x){   
 sample(x = c(0,6), size = 1, prob = c(.3,.7))  
 } else {   
 sample(x = c(0,6), size = 1, prob = c(.6,.4))  
 }  
 }))  
  
head(childlevel, n = 10)

## childID SpecimenDate SampleType TestResult  
## 1 11716 2011-02-13 Venous 6  
## 2 179 2018-12-05 Venous 0  
## 3 5875 2018-02-07 Capillary 0  
## 4 5548 2014-06-21 Capillary 6  
## 5 16271 2016-07-19 Capillary 0  
## 6 5209 2014-12-25 Venous 0  
## 7 14488 2013-07-04 Capillary 6  
## 8 18121 2010-03-06 Capillary 6  
## 9 18980 2012-09-05 Capillary 6  
## 10 1463 2018-01-21 Venous 6

Child level data is aggregated via:

1. Splitting the data into children with only a single test in a calendar year and children with multiple tests within a calendar year.
2. Making the 2nd dataset (multiple tests per child per year) compliant with our one test per child per year rule (highest venous if available, lowest non-venous if a venous is not available).
3. Recombining the datasets (no longer 100,000 rows, only 78,662)
4. Aggregating the data to proportion of tests above the threshold monthly. Note: The YYYY-MM-28 structure is used for aesthetic spacing on plots not included in this markdown, that is the day being 28 doesn’t have significance (i.e. Jan 31 data is aggregated).

potential\_duplicates <- duplicate\_split(x = childlevel,  
 childID\_column = "childID",  
 specimen\_date\_column = "SpecimenDate")  
  
childlevel\_compliant <- one\_test\_per\_child(x\_list = potential\_duplicates,  
 test\_result\_column = "TestResult",  
 sample\_type\_column = "SampleType",  
 venous\_code = "Venous")  
  
monthlevel <- monthly\_aggregator(x = childlevel\_compliant,  
 specimen\_date\_column = "SpecimenDate",  
 test\_result\_column = "TestResult",  
 threshold = 5)[,c(1,2)]  
  
# In the shiny application, labels are overwritten via a dropdown  
# menu to remove the need for templating. To fit into later function   
# defaults we'll change the column names   
  
colnames(monthlevel) <- c("Date","Proportion")  
  
  
head(monthlevel, n = 10)

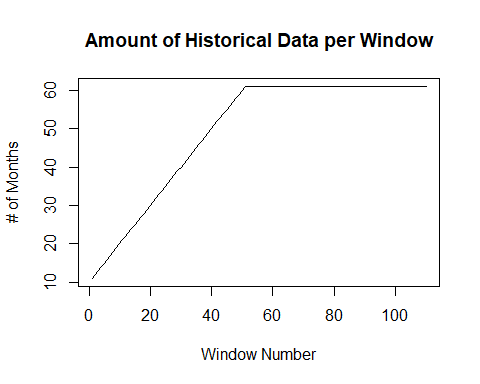
## # A tibble: 10 x 2  
## Date Proportion  
## <date> <dbl>  
## 1 2010-01-28 0.444  
## 2 2010-02-28 0.393  
## 3 2010-03-28 0.387  
## 4 2010-04-28 0.398  
## 5 2010-05-28 0.362  
## 6 2010-06-28 0.447  
## 7 2010-07-28 0.426  
## 8 2010-08-28 0.438  
## 9 2010-09-28 0.406  
## 10 2010-10-28 0.428

### Analysis on Monthly data

Once data is aggregated to the monthly level, the rest of the modeling is based around a univariate time series (with splines and knots as described in the primary paper.)

Where the first set size indicates the smallest amount of historical data a window can have to receive analysis (default is 11 months) and the window size indicates the largest amount of historical data a window can have. When the windows begin to have that much data, the windows move instead of expanding.

month\_windows <- datasplit(dataset = monthlevel,  
 first\_set\_size = 11,  
 window\_size = 60)  
  
plot(x = 1:length(month\_windows),   
 y = unlist(lapply(month\_windows,FUN = nrow)),   
 type = "l", ylab = "# of Months", xlab = "Window Number",  
 main = "Amount of Historical Data per Window"  
 )



February 2018 is analyzed with a history from Feb 2013 - Jan 2018 and March 2019 is analyzed with a history from March 2014 - Feb 2019. Customized arima, cusum, and shewhart irchart functions are applied to create an alert system that is plotted in the application.

Here the defaults for the cusum are shown: the residuals should be normally distributed around 0 (target mu), and we tolerate a 1 standard deviation range (k reference interval) before cumulatively adding the excess normalized residuals (S) to identify an alert.

Because the Feb 2018 results are not relevant to the March 2019 results (although Feb 2018’s data is included in the March 2019 history) the final row of each window is extracted to form our final dataset for visualization.

arima\_list <- lapply(X = month\_windows, FUN = CDC\_arima\_v2)  
  
cusum\_analyses <- S\_Loop(list\_of\_v\_tbls = arima\_list,  
 target\_mu\_ = 0,  
 kref = 1)  
  
shewhart\_analyses <- do.call(what = rbind,  
 args = lapply(X = arima\_list,   
 FUN = irchart))  
  
final\_table <- cbind(  
 cusum\_analyses[,c("Date","Proportion","predicted","S","residuals")],  
 shewhart\_analyses[,c("alert\_min","override\_cusum")]  
 )  
  
# for aesthetic display   
  
final\_table[,c("Proportion","predicted","S","residuals","alert\_min")] <-   
 round(final\_table[,c("Proportion","predicted",  
 "S","residuals","alert\_min")],  
 digits = 3)

### Analyzing the Final Output

tail(x = final\_table, n = 10)

Date Proportion predicted S residuals alert\_min override\_cusum  
2019-03-28 0.408 0.391 0.000 0.017 0.049 FALSE  
2019-04-28 0.391 0.390 0.000 0.001 0.049 FALSE  
2019-05-28 0.398 0.391 0.000 0.007 0.048 FALSE  
2019-06-28 0.424 0.388 1.079 0.036 0.049 FALSE  
2019-07-28 0.420 0.395 0.894 0.025 0.048 FALSE  
2019-08-28 0.418 0.401 0.183 0.016 0.049 FALSE  
2019-09-28 0.404 0.406 0.000 -0.002 0.049 FALSE  
2019-10-28 0.405 0.407 0.000 -0.002 0.049 FALSE  
2019-11-28 0.420 0.408 0.000 0.012 0.049 FALSE  
2019-12-28 0.718 0.409 16.269 0.308 0.066 TRUE

Looking at the last 10 rows of the final table: the increase from 42% (Proportion) of tests above 5 micrograms/deciLiter in November 2019 to 71.8% in December 2019 is far above the expected 40.9% (predicted). The cusum algorithm determines this to be 16.269 cumulative deviations (S) above expected.

Independently, because the 30.8% difference (residuals) between the actual and predicted proportion for December 2019 is above the threshold of 6.6% (alert\_min), the IRChart acts as an override for cusum (override\_cusum). This alert minimum plus the predicted value allows a warning line to be generated so that spikes in EBLLs can be flagged.

Ultimately, this final table was used to validate the SAS findings and create the R Shiny application (and R Package) used for ongoing analysis. Because the final plot is interactive it cannot be rendered in a word document, but an example from real data is included in the primary paper.