Supplement for

Triggering Interventions for Influenza: The ALERT Algorithm

Nicholas G Reich, Derek A T Cummings, Stephen A Lauer, Martha Zorn,
Christine Robinson, Ann-Christine Nyquist, Connie S Price,
Michael Simberkoff, Lewis J Radonovich, Trish M Perl

December 7, 2015

Contents

1 Introduction to the ALERT algorithm

In this document we examine and describe the utility of the "Above Local Elevated Respiratory illness Threshold" (ALERT) algorithm in prospectively determining the start and end to a period of elevated influenza incidence in a community. This algorithm could provide a valuable tool to communities, schools, hospitals and other institutions looking for a simple method to objectively define a period when, for example, enhanced patient contact precautions, empiric therapy, or other prevention measures should be implemented. The ALERT algorithm is a simple metric that can be easily operationalized to predict the onset of influenza season. It is not currently designed to generate predictions of case counts.

The ALERT algorithm uses data from previous flu seasons to determine an ALERT threshold. The ALERT period begins when the reported number of laboratory-confirmed cases for a given week exceeds the established ALERT threshold. This serves as the flu season trigger, signaling larger than expected fluctuations. To account for reporting delays and possible delays in implementation of any policies, the user may specify a lag period: a number of days between the reporting date associated with the trigger and the date the ALERT period should be put into effect. The ALERT period ends when the reported number of cases falls below the same threshold, after a minimum passage of eight weeks. (This grace period is chosen by default to be eight weeks but its duration can be modified by the user.)

The ALERT algorithm can be implemented either via the ALERT R package (available on GitHub), the ALERT web applet, or an Excel spreadsheet. The software for the package and the web applet are open-source and made available under the GNU General Public License, version 2.

This document describes the ALERT algorithm in detail and provides example R code and output.

The ALERT algorithm performs two distinct tasks. It can

1. calculate the historical performance of possible ALERT thresholds, thereby providing the information needed to choose an appropriate threshold, and

2. validate the prospective performance of the ALERT algorithm under one or more threshold decision rules.

For most users of the ALERT algorithm, the first task that calculates performance of thresholds will be all that is run. We discuss each of the tasks in more detail in the subsequent sections.

2 Getting started with ALERT

2.1 An example

We demonstrate the use of the ALERT algorithm to evaluate these possible thresholds, as implemented in the ALERT R package. You must have R installed on your computer. You can install the latest version of the ALERT package by running the following commands:

```
install.packages("devtools")
devtools::install_github("nickreich/ALERT")
```

Then, to load the ALERT package and the sample dataset that comes with the package, run the following commands:

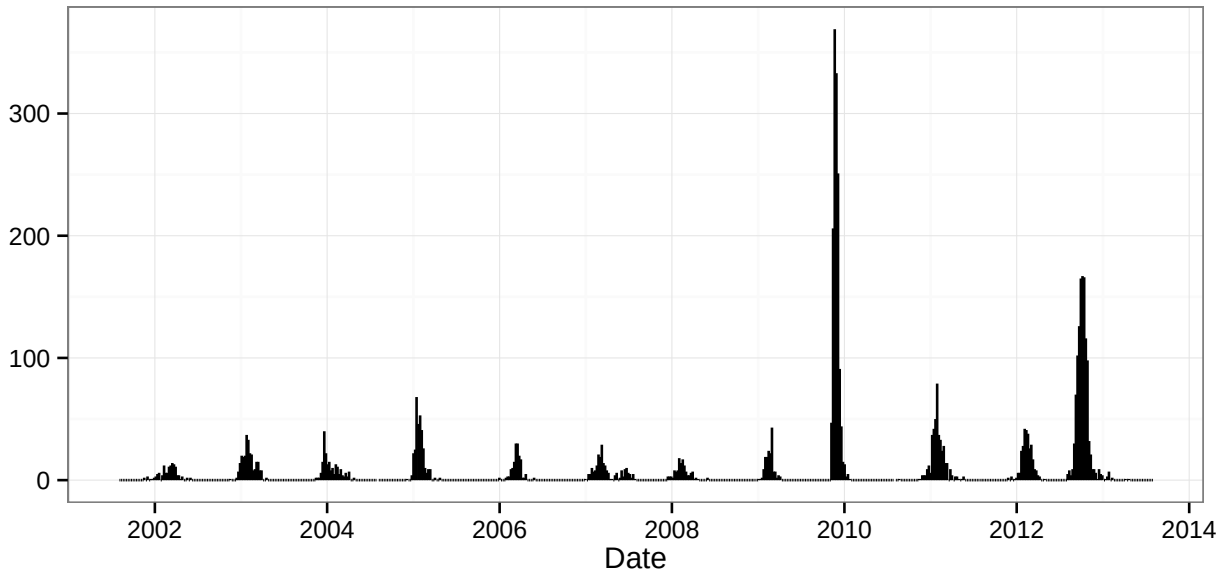
```
library(ALERT)
data(fluData)
```

The `fluData` dataset has a particular format that we assume all ALERT datasets will follow. In particular, it is a `data.frame` with a column named "Date" and a column named "Cases". These represent the number of cases observed in the week of time defined by the date given. This sample dataset is based on a real dataset, but the counts and dates have been modified from the original format based upon data sharing agreements with the owners of the data. This particular dataset has a total of 4849 cases reported across 624 weeks. The earliest observation is from 08/04/2001 and the last observation is from 07/27/2013. This is what the dataset looks like. (NOTE: lines that begin with ## indicate they are ouput from R that would be seen by the user who is having an interactive an R session.)

```
head(fluData)

##         Date Cases
## 1 2001-08-04     0
## 2 2001-08-11     0
## 3 2001-08-18     0
## 4 2001-08-25     0
## 5 2001-09-01     0
## 6 2001-09-08     0

library(ggplot2)
theme_set(theme_bw())
qplot(x=Date, ymin=0, ymax=Cases, data=fluData, geom="linerange")
```



2.2 Data formatting

To get your dataset to work with the ALERT package, you will need to ensure that you have a column named "Date" and another named "Cases". You can rename columns of a `data.frame` or `matrix` object using the following command:

Also, the column of dates must be in the formal 'Date' format of R. You can check the format of your date column by using the command

```
class(fluData$Date)

## [1] "Date"
```

If the class of your Date column is not "Date", then you will need to run a command to modify the data. For example, if your dates are character strings in mm/dd/yyyy format, then you could run the following command to convert the format

```
fluData$Date <- as.Date(fluData$Date, "%m/%d/%Y")
```

Additional details about how to specify the format can be found by typing the commands `?as.Date` and/or `?strptime`.

3 Evaluating possible ALERT thresholds

3.1 Methodological details

To define the ALERT period, we use past surveillance data to evaluate the retrospective performance of possible thresholds. The ALERT algorithm defaults to choosing potential thresholds as

the 10th, 20th, 30th, 40th, 50th, and 60th percentiles of all of the non-zero historical weekly case counts. In the R implementation of the ALERT algorithm, this can be specified as all integer thresholds between the 10th and 60th percentile.

For each threshold considered, the ALERT algorithm summarizes data from previous years as if that threshold had been applied. Say that we have historical data on N seasons. Let $X_{i,t}$ be the percentage of cases captured in the ALERT period for season i ($i = 1, ..., N$) and threshold t. Let $D_{i,t}$ be the duration of the ALERT period for season i and threshold t. For each threshold t considered, the ALERT algorithm calculates and reports the following metrics:

1. Across all seasons, the median percentage of all influenza cases contained within the ALERT period, $median(X_{i,t})$.
2. The minimum and maximum of $X_{i,t}$, the percentage of all influenza cases contained within the ALERT period.
3. The median ALERT period duration, $median(D_{i,t})$.
4. The fraction of seasons in which the ALERT period contained the peak week.
5. The fraction of seasons in which the ALERT period contained the peak week $+/-k$ weeks (k is specified by the user).
6. The mean number of weeks included in the ALERT period with counts less than the threshold.
7. The mean difference between, for each season, the duration of the ALERT period and the duration of the shortest period needed to capture P percent of cases for that season. (This metric requires a bit more computation time, and is only computed if the user specifies a P.)

3.2 Evaluating threshold calculations

Using the dataset shown in Section 2, we will evaluate a set of potential thresholds using the ALERT algorithm.

```
fluData_subset <- subset(fluData, Date<as.Date("2011-08-14"))
alert_summary <- createALERT(fluData_subset, allThresholds=TRUE,
                             k=2, firstMonth=8, target.pct=.85)
```

Full details of the usage of the `createALERT()` function can be obtained by running the command `?createALERT`. The options that we specified above are explained below:

- Specifying `allThresholds=TRUE` means that all integer thresholds between the 10th and 60th percentile of non-zero case counts are used. If `FALSE`, only the 10th, 20th, ..., 60th percentiles are used.

- Setting `k=2` specifies the number of weeks to compute ALERT coverage for around each season's peak week.
- The `firstMonth` option allows the user to specify in which month (specified by number) ALERT should start looking for an increase in flu activity each season.
- If the `target.pct` option is specified, then `createALERT()` computes the median difference between the duration of the ALERT period in a season and the duration of the shortest period needed to capture `target.pct` of cases for that season.

Table 1 shows slightly reformatted results from the `alert_summary$out` object, showing the historical performance of different thresholds. As one example, a user might choose the threshold of 6 because it is the highest threshold that has historically captured at least 85% of the cases half of the time.

Table 1: Printed table of the alert_summary$out object.

		% of cases captured			% captured			
threshold	med dur	median	min	max	peaks	peaks+/-k	low weeks	diff
2.0	14.5	96.9	16.4	98.1	90.0	80.0	1.7	3.2
3.0	12.5	92.9	14.1	97.2	90.0	70.0	1.5	1.4
4.0	12.5	86.5	69.1	97.2	100.0	80.0	1.2	0.9
5.0	11.5	85.7	67.4	96.2	100.0	70.0	1.3	-0.2
6.0	10.0	85.3	62.3	96.2	100.0	70.0	1.3	-1.0
7.0	8.5	83.9	60.2	96.2	100.0	70.0	1.5	-1.5
8.0	8.5	83.9	60.2	96.2	100.0	70.0	2.0	-1.6
9.0	8.0	76.4	55.0	96.2	90.0	60.0	2.4	-2.3

Looking at the historical performance metrics provides a useful snapshot of the performance of different thresholds. In many settings, this may provide enough information to choose a threshold for use in the future. However, Section 5 provides a more robust validation to determine whether the ALERT algorithm provides robust predictions about the future performance of a given threshold.

4 Applying an ALERT threshold to a single year of data

If we know what threshold we would like to use, we can apply it to a year of data to determine how it would have performed on that season of data using the `applyALERT()` function.

```
fluData_singleSeason <- subset(fluData,
                               Date>=as.Date("2011-08-14")&Date<=as.Date("2012-08-13"))
alert_applied <- applyALERT(data=fluData_singleSeason, threshold=3,
                            k=2, target.pct=0.85)
round(alert_applied, 2)
```

```
##         tot.cases          duration       ALERT.cases   ALERT.cases.pct
##            303.00             18.00            284.00              0.94
##     peak.captured peak.ext.captured    low.weeks.incl     duration.diff
##              1.00              1.00              4.00              8.00
```

The `applyALERT()` function shown here is the workhorse function that does most of the calculation for `createALERT()` and `robustALERT()` functions.

5 ALERT algorithm validation

Calculations from the `createALERT()` function that summarize the historical performance of different ALERT thresholds are useful. However taken on their own, it is not clear whether these results may be the used as good predictors of future performance. Therefore, to estimate the performance of the ALERT algorithm in prospective use, we designed and implemented a leave-one-season-out cross-validation analysis (see Section 5.2). This analysis evaluates whether the historical performance measures are adequate for projecting future performance for a given dataset.

5.1 Thresholds are chosen based on rules

To perform this ALERT algorithm evaluation and validation, a user must state clearly what criteria they will use to choose a rule. Here are some examples:

- *We are interested in the highest threshold that has historically captured over 85% of cases.*
  Using Table 1, this would suggest a threshold of 6 cases.
- *We want the lowest threshold that has had a median duration of no more than 12 weeks.*
  Using Table 1, this would suggest a threshold of 5 cases.
- *We would like the highest threshold that has historically captured the peak and the two weeks on either side at least 80% of the time.*
  Using Table 1, this would suggest a threshold of 4 cases, although this metric appears to be unstable at low thresholds with our data.

5.2 Single rule evaluation

To evaluate a particular rule, the ALERT algorithm uses the `evalALERT()` function, which conducts the following steps for each season i:

- Create a dataset that includes all seasons except season i. We will refer to the included seasons as the "training seasons".
- Run the `createALERT()` function to calculate the performance of potential thresholds across the 'training seasons.'

- Choose the best threshold t based on the rule provided.
- Run the `applyALERT()` function using season i and threshold t.
- Save the ALERT performance metrics for season i.

5.3 Multiple rule evaluation

To evaluate many rules at a time, one can use the `robustALERT()` function which streamlines the evaluation of several rules in one function. Note that the following command may take a long time (several minutes, for us) to run. (For the compilation of this vignette, the alert_eval datafile was loaded from the package directly using `data(alert_eval)` rather than relying on the time-consuming computation each time we compiled the document.)

```
alert_eval <- robustALERT(data=fluData, allThresholds=TRUE,
                          k=2, firstMonth=8, lag=7, minWeeks=8,
                          minPercent=c(.8, .85, .9),
                          maxDuration=c(12, 13, 14))
```

Table 2 shows slightly reformatted results from the `alert_eval` object, showing the mean cross-validated results of different rules.

Table 2: Printed table of the alert_eval object.

rule	thresh	dur	ALERT cases	%	peaks	peaks+/-k	low weeks	diff
minPercent = 0.8	9	8.5	161.5	82.5	91.7	58.3	2.2	-0.3
minPercent = 0.85	6	11	196	88.2	100.0	75.0	1.3	0.2
minPercent = 0.9	3.5	13.5	211	90.2	100.0	83.3	1.4	0.8
maxDuration = 12	5	12	209.5	88.2	100.0	75.0	1.3	
maxDuration = 13	3.5	14.5	219	93.5	100.0	83.3	1.5	
maxDuration = 14	3.5	14.5	219	93.5	100.0	83.3	1.5	

5.4 Comparing validation with historical performance

We compared the performane of the raw ALERT threshold calculations with the cross-validated metric calculations. These results are shown in Table 3 for the "minPercent = 0.85" and "maxDuration = 12" rules. The close similarity in computed metrics between the pairs of raw and validated rows indicate that the metrics calculated on the raw historical data may serve as a robust measure on which to choose ALERT thresholds. If we had observed large differences between the raw and validated measures, then we would have concerns over the validity and robustness of the metrics computed only on historical data. However, these concerns are not supported by the data shown below. (Note that the duration difference metric can only be calculated when a target percentage is specified, which is not done for the maximum duration rule.)

Table 3: Comparison of ALERT output with validated metrics.

rule	type	thresh	dur	ALERT cases pct	peak	peak +/- k	low weeks	diff
minPercent=0.85	raw	6	10	85.3	100.0	70.0	1.3	-1
	validated	6	11	88.2	100.0	75.0	1.3	0.2
maxDuration=12	raw	5	11.5	85.7	100.0	70.0	1.3	
	validated	5	12	88.2	100.0	75.0	1.3	

6 Calibrating the sensitivity of the ALERT algorithm

Due to the variation in patterns and duration of seasonal influenza incidence, the ALERT algorithm will inevitably trigger false alarms and/or miss seasons where observed data are far from the historical norm. One way to insure against triggering too early and missing a season is to specify, a minimum ALERT duration time (e.g. 8 weeks). This prevents an ALERT period from being prematurely terminated by early season fluctuations around the trigger threshold. However, it also means that we are relying on the minimum duration time to help improve the performance of the algorithm. Eight weeks is a justifiable minimum time to allow a season to start to take off. However, defining a shorter minimum duration would lower the average duration while increasing the chances of a premature termination of the ALERT period.