User Manual for the Repeating Event Sequence Alarm (RESA)

By: Gabrielle Tepp (Alaska Volcano Observatory/USGS)

Last updated: 3/20/2018

Warning: Documentation is still under development! Hopefully there’s currently enough to get you started.

**1. Introduction**

The repeating event sequence alarm (RESA) was designed to alert users to sequences of repeating events that sometimes occur during volcanic activity. These sequences may precede eruptions, so the RESA was created to provide automatic early detection and short-term warning of possible eruptions. However, the RESA is capable of detecting and alerting on any type of repeating activity, including that sometimes preceding landslides/avalanches and repetitive small explosions.

The idea behind the RESA is to use a standard detector (e.g., STA/LTA) to find events and then apply a correlation-matching algorithm to identify repeating event sequences. Once a sequence has been identified in progress on a minimum number of stations, an alert notification (email or text) is sent to specified recipients. This procedure can also be followed for a “level 2” alert that notifies of an increase in event rate. Notifications are also sent when a sequence event rate drops below the minimum requirements and is deemed to have ended. The alarm runs periodically on a chosen time interval.

Next is a brief overview of the alarm algorithm. For more information, you can read the paper listed below. Please cite the paper if you use this software. Thanks!

Tepp, G. (2018) A Repeating Event Sequence Alarm for Monitoring Volcanoes, in revision for *Seismological Research Letters*.

1.1 Method

The idea behind the RESA is to use a standard detector (e.g., STA/LTA) to find events and then apply a correlation-matching algorithm to identify repeating events. Once a sequence has been identified in progress on a minimum number of stations, an alert notification is sent to specified recipients.

1) Event Detection (STA/LTA)

The RESA starts by detecting events on the chosen time interval (i.e., how often it runs for real-time). This is currently done with a short-term average/long-term average detector, but it could be replaced by something better.

2) Correlation-Matching

Once events have been found, they go through a template-matching step. Since the RESA is initialized without any templates, the first event must become template 1. After that each event is correlated with all existing templates to find the best matching template. If the cross-correlation value exceeds a minimum threshold, then the event is stacked with the template and its onset time is added to the template's event list. If the threshold is not exceeded, the event becomes a new template. To reduce file sizes and keep the run-time low, a maximum number of templates is set. Once the maximum number is reached, the oldest template is removed whenever a new one is added.

After all events have been correlated, all templates are cross-correlated, and any correlating above the threshold are merged - the waveforms are stacked, event lists combined, and now-duplicate templates removed.

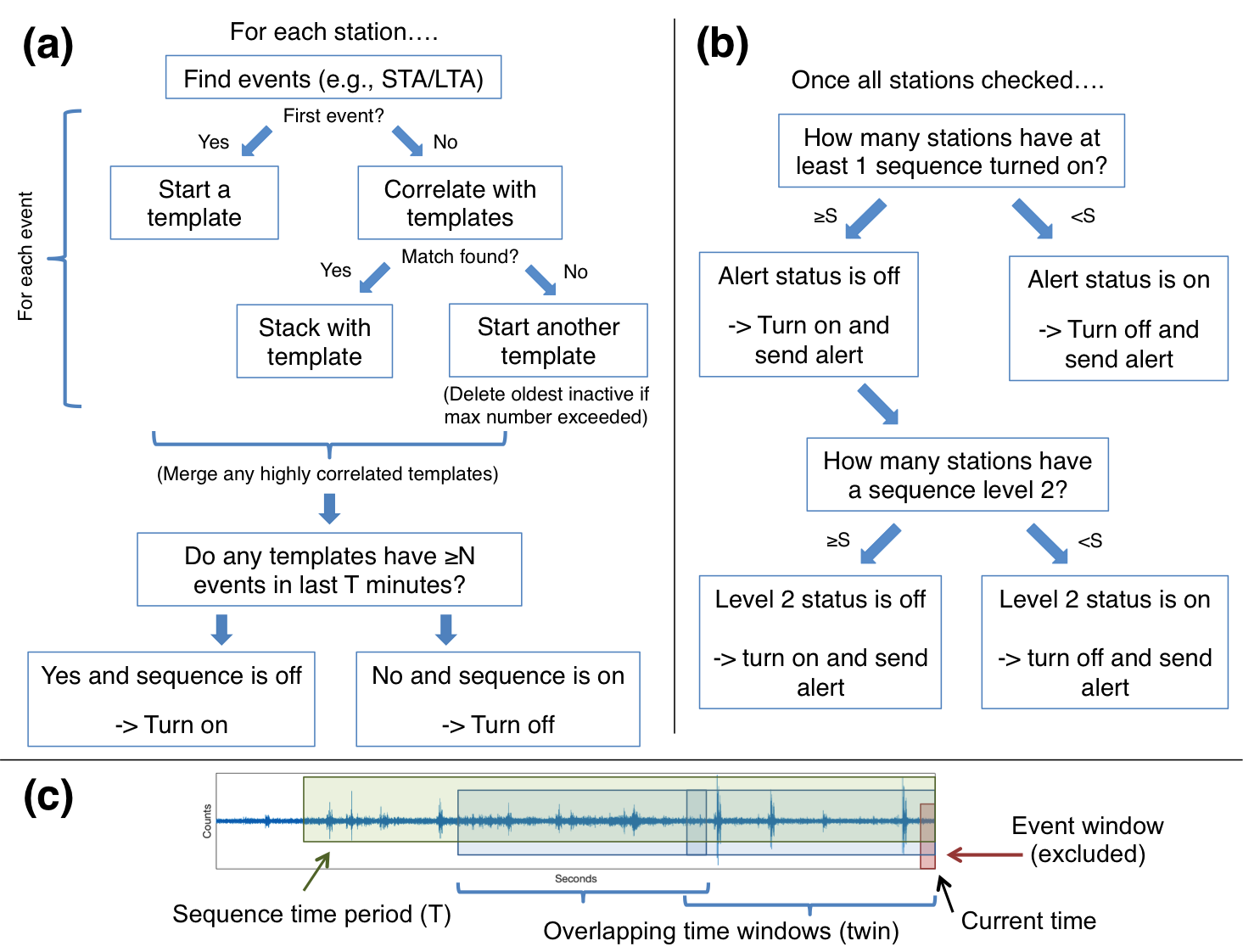


Figure 1.1: Diagram of the RESA method.

3) Sequence Determination

The next step is to determine whether any of the templates are a sequence. Sequences are defined as a number of events (N) in some time period (T). Two levels can be set up to send alerts for increases in event rates in addition to sequence starts/ends. The number of events in each template over the preceding T period are counted. If the number is greater than or equal to N, then the template is flagged as having a sequence. Steps 1-3 are performed on each station individually with a station being flagged if at least one of its templates is.

4) Send Notification (if applicable)

If the alarm is "off" and at least a minimum number stations have been flagged with a sequence, then the alarm is turned "on" and a text/email notification is sent. The same happens if the alarm is "on" and the number of stations drops below the threshold.

**2. How to Use**

2.1 Dependencies

The RESA uses the GISMO toolbox for MatLab (https://github.com/geoscience-community-codes/GISMO). Two of the GISMO files have been updated (see notes at top of RESA code) and can be downloaded from the main page.

If you want to pull data directly from the IRIS Data Management Center (DMC), make sure you download the irisFetch script. (Note that the RESA retro test gets data from IRIS.) The software can be downloaded here: https://ds.iris.edu/ds/nodes/dmc/software/downloads/irisFetch.m/

2.2 RESA Retro

This script can be run directly in Matlab and can use data from a Winston Waveserver, irisFetch, or files in SAC, CSS, or generic binary format (additional file formats could be added). The parameters are hard-wired at the top of the script.

2.3 RESA Cron

This script was designed to be run as a cron job. It reads in parameters from a text file and saves template waveforms and other necessary information to additional files.

Currently, I have the cron jobs running via a bash script in the following format. Note that lines which write out the times are optional as are the lines pertaining to error messaging.

#! /bin/bash

/Applications/MATLAB\_R2016b.app/bin/matlab -nodesktop -nodisplay -nosplash <<EOF

disp(['Current time: ' datestr(now) ' AKDT.']);

try;

paramfile = '/Users/username/Documents/Alarm\_folder/Parameters/volcano.txt';

RESA\_cron;

disp(['Current time: ' datestr(now) ' AKDT.']);

catch excep;

setpref('Internet','E\_mail','volc\_eventseq.alarm@host.ext');

disp('Script did not run successfully.');

disp(['Error: ' excep.message ' at line ' num2str(excep.stack(1).line)]);

disp(['Current time: ' datestr(now) ' AKDT.']);

sendtochk = {'email@host.ext'};

sendmail(sendtochk,'Volcano RESA Failure',['RESA failed to successfully run at ' datestr(now) ' AKDT.' 10 10 'Error: excep.message ' at line ' num2str(excep.stack(1).line)]);

end;

exit

EOF

**3. Testing**

The RESA Retro comes set up with parameters to draw data from the IRIS DMC via irisFetch. It can be run straight out of the download with these parameters as a test. More details and test results for comparison coming soon!

**4. Setting up Parameters**

The RESA has many input parameters. Most should be known to those who are familiar with seismology, STA/LTA algorithms, and/or cross-correlation. Below is a list of the parameters, what they are, and more information about how they might be set.

4.1 Network, Stations, and Channels

All lists should be comma separated with entries the same number of characters (blank spaces are ignored when ).

params.net.str - list of networks to be used; currently assumes only one network (could easily be changed)

params.sta.str - list of stations to be used

params.cha.str - list of channels for each station (in same order)

4.2 File Information

directory - directory to save files to

structfile - name of structures file; this is where the template data and other information is saved to

logfile - name of the log file

4.3 General Data Parameters

dbtype - database type (IRIS or Winston (WIN, default))

mySource = datasource('winston',IPaddress,Port); % data source object (Winston source ONLY)

Data file parameters - Retro ONLY

ftype - file type if using files (options: CSS, BIN (binary), SAC)

datadir - data file directory if using files

fileinfo - list of file information if using files

twin - data window to search (in minutes)

Note: twin includes the time window overlap which is automatically set = l\_lta + evwin to avoid duplicate events; For RESA Cron, the script should be set to run every twin-overlap.

4.4 Filter parameters

The RESA assumes a band-pass filter. However, this could be changed fairly easily.

poles - number of poles of the filter

lf - low frequency limit of filter band

hf - high frequency limit of filter band

4.5 STA-LTA Parameters

min\_sep - minimum separation between detections (in seconds)

min\_dur - minimum duration between “on” and “off” needed to make a detection (in seconds)

l\_sta - short-term average window length (in seconds; measured at the right end of the LTA window)

l\_lta - long-term average window length (in seconds)

threson - threshold for detection “on”

thresoff - threshold for detection “off”

4.6 Event Waveform & Cross-correlation Parameters

buff - time of window start before event onset (in seconds)

evwin - length of event waveform window (in seconds)

tempmax - maximum number of template events to hold in memory

mincc – minimum cross-correlation value needed for a match

maxlag - maximum allowable lag time between an event and a template needed for a match (note: may be good to set as 2\*l\_sta; in seconds)

xcst - cross-correlation start time relative to start of event/template window (in seconds)

xcend - cross-correlation end time relative to start of event/template window (must be <= evwin; in seconds)

4.7 Sequence Parameters

minev - minimum number of events required for sequence

seqT - time period within which sequence can be declared (in minutes)

minevoff - minimum number of events required for sequence to remain on (will turn off when number drops below this)

seqToff - time period to use for sequence off (in minutes)

l2chk - check for increasing (level 2) sequence (1 for on, 0 for off (default))

minevl2 - minimum number of events required for level 2 sequence

seqTl2 - time period within which level 2 sequence can be declared (in minutes)

minevoffl2 - minimum number of events required for level 2 sequence to remain on (will turn off when number drops below this)

seqToffl2 - time period to use for level 2 sequence off (in minutes)

minsta - minimum number of stations with sequence on needed to send alert

minreq - minimum number of stations required from preferred list to send alert

reqstastr - list of preferred/required stations

offwait - number of time windows with no sequences on before level 1 "off" alert is sent (set to 0 for none)

offwaitl2 -number of time windows with no sequences on before level 2 "off" alert is sent (set to 0 for none)

seqoffwin - number of extra time windows to wait before station-level sequence turns "off" (set to 0 for none)

holdt - amount of time required before alert status can change to "off" (in minutes)

holdtl2 - amount of time required before level 2 alert status can change to "off" (in minutes)

4.8 Email/Text Alerts

setpref('Internet','E\_mail','address@host.ext'); - this sets the “from” address

sendlist - list of email addresses or phone numbers to send sequence alert to; each row has the following format (separated by a semi-colon): {email, phone, lv 1 on, lv 1 off, lv 2 on, lv 2 off} (1=email, 2=phone, 0=none)

sendchk - send a check that code is running (1=yes, 0=no (default))

sendtochk - list of email addresses or phone numbers to send check alert to

**5. Hints and Tricks**

Coming soon!