PyBet Tools User Guide

1. Introduction
   1.1 PyBetVH
   1.2 PyBetEF
   1.3 PyBetUnrest

2. Background Theory
   2.1 Bayesian Inference
   2.2 Monitoring

3. Local Installation Requirements
   3.1 Linux
   3.2 Mac OSX/Windows
   3.3 Requirements verification

4. Running the tools
   4.1 PyBetVH
   4.2 PyBetEF
   4.3 PyBetUnrest

5. Input preparation
   5.1 PyBetVH
   5.2 PyBetEF

Disclaimer

References

PyBet models are a set of probabilistic tools aiming to provide probabilistic estimation of volcanic unrest, of eruption, and of any hazardous phenomena linked to volcanic activity. For this reason, they can be used for probabilistic forecasting during crisis of unrest and/or more comprehensive probabilistic volcanic hazard assessments (PVHA). The background theory of these models is based on the definition of ad-hoc Bayesian event trees, where each node is treated through a Bayesian inference approach. This allows to merge together different kind of available information, as theoretical/empirical models, a-priori beliefs, past data (for short-term analysis).
current software implementations of these models, with the specific goal of providing a user-friendly interface to the final users. The softwares have been developed by using the Python programming language, since the broad compatibility of Python allows to install them on all the most common operating systems (Linux, Mac OSX, Windows) and also on the VHub cyber-infrastructure, where they can be freely run online or downloaded. In the next sub-sections they are briefly introduced.

More details on the BET's models background theory can be found in literature (among others see Marzocchi et al., 2004; Marzocchi et al., 2010) and are out of the scope of this document. However, some basic principles will be briefly introduced in order to properly approach the use of the corresponding tools.

1.1 PyBetVH

PyBetVH is a completely new, free, open-source and cross-platform software implementation of the Bayesian Event Tree for Volcanic Hazard (BET_VH). The purpose of the tool is to provide a graphical support to BET_VH model, which estimates the long-term probability of any magmatic hazardous phenomenon (i.e., lava flows, tephra fall, pyroclastic flows, lahars, etc.) occurring in a selected time frame, accounting for all the uncertainties. The BET_VH model represents a flexible tool to provide probabilities of any specific event at which we are interested in, by merging all the available information, such as theoretical models, a priori beliefs, and past eruptions. It is mainly based on the Bayesian inference and it deals with long-term forecasting only, therefore it can be useful in land use planning and in risk assessments.

PyBetVH calculates hazard curves, which describe the distribution of the exceedance probability as a function of a user-selected intensity measure (e.g., tephra load) on a grid of points covering the target area. The computed hazard curves are (i) absolute (accounting for the probability of eruption in a given time frame, and for all the possible vent locations and eruptive sizes) and (ii) Bayesian (computed at different percentiles, in order to quantify the epistemic uncertainty). Such curves allow representation of the full information resulting from PVHA, and are well suited to become a main input to quantitative risk analyses. PyBetVH allows for interactive visualization of both the computed hazard curves, and the corresponding Bayesian hazard/probability maps. PyBetVH is designed to minimize the efforts of end users, making PVHA results accessible to people who may be less experienced in probabilistic methodologies, e.g. decision makers.

More details on PyBetVH tool can be found in Tonini et al. (2015a), where the use of the tool is illustrated through the example of PVHA for tephra fallout from the Okataina Volcanic Centre (OVC), New Zealand, by highlight the range of outputs that the tool can generate.

1.2 PyBetEF

PyBetEF is an open and cross-platform implementation of BET_EF (Bayesian Event Tree for Eruption Forecasting) equipped with a Graphical User Interface. BET_EF model aims to provide probabilities of unrest/eruption forecasting of magmatic events, by merging all the available information such as theoretical models, a priori beliefs, monitoring measures, and any kind of past data. BET_EF model is based on a Bayesian procedure and it relies on the fuzzy approach to manage monitoring data. The method deals with short- and long-term forecasting; therefore, it can be useful in many practical aspects such as land use planning and volcanic disaster management. BET_EF model has been applied to several volcanic systems (Marzocchi et al., 2008; Sandri et al., 2012; Selva et al., 2012b). NOTE: PyBetEF is not maintained anymore, since it has been included in the PyBetVH package.
PyBetUnrest is an open and cross-platform implementation of BET_UNREST model. BET_UNREST is an extension of BET_EF
monic unrest and its relating hazardous phenomena, by adding a specific branch to the event tree (Rouwet
ned and developed as one of the final products of the EU VUELCO project and tested during the last
ized in Dominica (West Indies) in the frame of VUELCO (Sandri et al., Under review).

More details on PyBetUnrest tool can be found in Tonini et al. (2016), where the use of the tool is illustrated through an application to
sets of the Kawah Ijen stratovolcano, Indonesia. In particular, the tool is set on the basis of monitoring
000–2010, and is then blindly applied to the test period 2010–2012, during which significant unrest

2. Background Theory

ET models are the definition of an event tree (ET, hereafter) and the use of the Bayesian inference. ET is
events in which individual branches are alternative steps from a general prior event, state, or condition
subsequent events (intermediate outcomes) to final outcomes (see Figure 1). In this way, an ET shows
s of volcanic unrest at progressively higher degrees of detail. The branches at each node represent
ough these need not be mutually exclusive or exhaustive. The Bayesian approach is used at the
retical models of the eruptive process, past (historical and geological) data, and, for short-term
ring of the volcano. Thus it allows to formally account for both aleatoric (due to the intrinsic randomness
(due to limited data or knowledge) uncertainties. Through the ET, one can calculate both absolute and
responds to consider the selected full path or just a single node in the ET itself.
2.1 Bayesian Inference

The Bayesian methodology allows to include a formal probabilistic treatment of the available data and their uncertainties and each node of the event tree is represented by a probability density function (pdf) of the probability at the node. Each node is assigned a probability by means of a Bayesian approach, meaning that a prior probability distribution (usually coming from theoretical models) and information from past data are statistically combined together to obtain a posterior probability. Quantities are estimated by considering all the available information (analytical/empirical models, expert beliefs, data) as described in the following formulation:

$$	heta \mid \text{data} = \frac{1}{Z} \cdot \text{pdf}(\theta \mid \text{data}) \cdot \text{pdf}(\text{data} \mid \theta)$$
In the BET model, both prior and posterior have a Beta distribution.

Figure 2: Bayesian inference: posterior distribution as result of prior model updated through available observed data.

The nature of the Bayesian analysis procedure can be summarized by the following practical concepts:

1. If the prior is uninformative, the posterior is very much determined by the data (the posterior is data-driven).
2. If the prior is informative, the posterior is a mixture of the prior and the data.
The more informative the prior, the more data you need to change your beliefs.

Monitoring data are accounted by identifying all the monitored parameters that can provide information about a degree of anomaly at each node of the ET. The identification of anomalies is where volcanologists introduce their conceptual models about the pre-eruptive process. The degrees of anomaly of each measure is computed in two ways (see Figure 3 here below): the parameter’s anomaly can exceed one threshold, as shown in the left panel of Figure 3, or through a fuzzy approach (by defining two thresholds), as shown in the right panel of Figure 3.

Once a number of monitoring parameters have been identified and the corresponding thresholds defined, it is possible to compute for each degree of anomaly any time new data become available. For the first node (unrest), the identification of an anomaly for one parameter is enough to set the probability of unrest to one. For nodes 2 (magmatic unrest) and 3 (eruption) the code translates the total degree of anomaly at the node into a conditional probability. The total degree of anomaly at the node is

\[ Z = \sum_{i} w_i \]

where \( z_i \) is the anomaly for parameter \( i \) and \( w_i \) the weight given at parameter \( i \).

The relationship between the degree of anomaly \( Z \) and the entropy score \( H \) (defined by logarithm of \( 1-P \)) which is a measure of the predictability of the event of interest. In other words, when the degree of anomaly doubles, the predictability doubles. Specifically, the degree of anomaly is used to estimate the mean of a Beta distribution describing the conditional probability.
of magmatic unrest (node 2), or eruption (node 3) through:

\[ H = -\ln(1-P) = -(a? - bZ) \]

from which we obtain

\[ P = 1 - a \exp(-bZ) \]

The parameter \( a \) and \( b \) can be defined for any volcanic systems, since their meaning is strictly connected to specific configuration of
and the expert’s beliefs on the selected volcano.

The parameter \( a \) (intercept with the y axis) is set from \( 1-a \) that is the probability that the event occurs when we observe \( Z=0 \) (no
anomalies). The default for the parameter \( a \) is 0.9 (at node 3, this means that there is 0.10 probability to have an eruption in the next time window without
observing any anomaly at the node).

The parameter \( b \) (steepness of the curve) is set defining what is the value of \( Z \) for which there is 0.90 probability that the event
occurs. For instance \( b=1 \) corresponds to about 0.90 probability of occurrence if 2 parameters are recognized as anomalous. The
definition of this parameter also depends on the weighting scheme. We consider that the maximum weight for one parameter is 1.

The value of \( P \) calculated by equation (*) is the mean of a Beta distribution. The variance of the distribution is set by the parameter

\( \Lambda \) that is the equivalent number of data. The equivalent number of data mimics how much we believe to the
conceptual model that stands behind the choice of the anomalies. In other words, if we set \( \Lambda=5 \) we think that our model
we can get from a probability calculated from 5 data. Of course the larger \( \Lambda \), the higher the
3. Local Installation Requirements

All PyBet tools require Python and some Python third party modules/libraries which need to be installed in order to be able to run the tools if the Python interpreter is installed. By typing “python” in a terminal window followed by the Enter key, a similar response:

    Apr 20 2012, 22:39:59

> Python 2.7.3 (default, Apr 20 2012, 22:39:59)

Type “help”, “credits” or “license” for more information.

The required additional modules/libraries are:

1. wxPython (graphical user interface library)
2. Numpy (scientific computing library)
3. Matplotlib (2D plotting library)
4. Pillow (a fork of the former Python Image library)

All of them are open source and freely available and usable for all the most common operating systems (Linux, Mac OSX and Windows). Depending on the operating system and on the experience of the user in using its machine and/or Python, the procedure to install these libraries can be approached in different ways. **NOTE:** The current versions of all BET tools run on Python 2.7.x. The porting to Python 3.x.x is on the way.

### 3.1 Linux

In the majority of Linux distributions, Python libraries can be easily installed by means of their own package managers, which install, update, remove softwares in the system in a consistent manner, by checking for all the required dependencies and shared libraries. For Debian-like distributions, you can use the synaptic tool or simply the following command from the terminal:

```
sudo apt-get install python-numpy python-matplotlib python-wxgtk3.0 python-pil
```

The procedure for other distributions is very similar.
Both these operating systems do not have a native package manager to handle libraries and softwares in a smart way as Linux distribution. So the installation of Python libraries could not be so straightforward. One can download and install the single libraries from the respective web sites, being very careful to the Python Standard Library version installed on his/her own computer as a reference to choose the correct files. Moreover, Mac OSX users can try to use MacPorts (https://www.macports.org/) or Homebrew in their own expertise in using these tools, which should play as package managers similarly to the ones available for Linux systems. Alternatively, as easiest way, one can install the Enthought Canopy free version which will install a scientific Python environment in but more solid and it takes care of the compatibility among all the libraries. Once you have installed Canopy (no matter in your own operating system), one should set the Canopy Python environment as the default Python. Canopy is a very rich Python package distribution, which will install a scientific Python environment in your system. It is less flexible, but more solid and it takes care of the compatibility among all the libraries. Once you have installed Canopy (as any other software in your own operating system), one should set the Canopy Python environment as the default Python.

For this purpose, one could export the following line to the .bash_profile file:

```bash
export PATH=/Users/$YOURUSERNAME/Library/Enthought/Canopy_64bit/User/bin:$PATH
```

When Canopy is set as your default Python environment, by running the python command from your terminal, you should get something similar:

```
macosx@macuser:~$ python
Enthought Canopy Python 2.7.9 | 64-bit | (default, Jun 30 2015, 19:41:21)
[Apple Inc. build 5658] (LLVM build 2335.6) on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

### 3.3 Requirements verification

Independently on the operating system you are using and on the method used to install these Python libraries, their correct installation can be verified by entering in the Python interactive shell and manually importing the libraries:

```
rob@robelix:~$ python
Python 2.7.3 (default, Apr 20 2012, 22:39:59)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> import numpy

>>> import matplotlib

>>> import wx

>>> from PIL import Image
```
If the previous requirements are satisfied, you can proceed with the installation of the tool(s) on your system.

4. Running the tools

4.1 PyBetVH

Download the source code from the PyBetVH main page on the VHub web site (Download does not require to be registered users). On the right, by clicking on download, it will be asked to save a compressed file (betvh-rxxx.tar.gz). The folder where the user has full permissions (reading, writing and executing) on files. This will be the installation folder of PyBetVH and should not be moved anymore. After having extracted the files from the tar.gz file, the following should appear:

```
ls -hot betvh-rxxx
```

```
Jan 8 22:04 bin
Jan 8 22:02 src
Nov 9 19:01 data
Nov 9 19:01 doc
Nov 9 19:01 examples
Nov 9 19:01 rappture
Nov 9 19:01 LICENSE.txt
Nov 8 16:56 COPYING
```

This is the structure required by VHub repository. After having opened a terminal window and moved into src/ folder, it must be executed the following command:

```
rob@robelix:betvh-rxxx$ make install all
```

This installs all the executable files in the bin/ directory. Open a terminal and move inside the bin/ folder:
You can now launch the tool with:

```
rob@robelix:~$ python betvh.py
```

Depending on your operating system, you can now set a link to the main executable, which is in /yourpath/betvh-rxxx/bin/betvh.py. For example, in Linux and Mac OSX platforms:

```
rob@robelix:~$ chmod u+x /yourpath/betvh-rxxx/bin/betvh.py
rob@robelix:~$ ln -s /yourpath/betvh-rxxx/bin/betvh.py /somewhereinyourPATH/PyBetVH
```

Finally, if all the operations above have been successfully concluded, the tool can be launched by simply doing, from any position in your file system, in the terminal window:

```
rob@robelix:~$ PyBetVH
```

4.2 PyBetEF

PyBetEF is not yet installed on VHub. You can download it from [here](#) together with a working example. Uncompress the .zip file and place the whole folder where you prefer. The folder contains the PyBetEF main program and all the modules needed to be properly executed. The user should not modify, remove or do any change to this folder, since the risk is to compromise the functionality of the tool. The tool is ready to be launched for a first trial. Open a terminal and move inside the folder containing the PyBetEF source files:

```
rob@robelix:~$ cd /your/path/to/PyBetEF/folder/
```

then launch the tool:

```
rob@robelix:~$ python PyBetEF.py
```

There are several ways to make this operation faster. In Linux and Mac OSX systems, as illustrative example, one can make the following operations:

```
rob@robelix:~$ chmod u+x /your/path/to/PyBetEF/folder/*
rob@robelix:~$ ln -s /your/path/to/PyBetEF/folder/* /somewhereinyourPATH/PyBetEF
```
The executable and then create a (symbolic) link to it in a folder belonging to its PATH environment

```
rob@robelix:~$ chmod u+x /yourpath/PyBetEF/PyBetEF.py
rob@robelix:~$ ln -s /yourpath/PyBetEF/PyBetEF.py /somewhereinyourPATH/PyBetEF
```

In this way one should be able to run the tool from any place of her/his filesystem by opening a terminal and simply doing:

```
rob@robelix:~$ PyBetEF
```

In Windows one can create a link (i.e., on the Desktop) by clicking with the mouse’s right button and selecting create a link. Then double click on the link should directly launch the tool.

You have to unzip the file and place the FakeVolcano folder wherever in your file system. The folder PyBetEF. Instructions on how to prepare the input files are given in Input preparation section. The next step is to load the Volcano button in the upper left panel and select the FakeVolcano folder. If everything went okay this example is loaded in the lower left panel and the user can explore it by clicking on the nodes. Once the user can press the COMPUTE button on the lower right panel. A new window frame will be opened, visualizing the corresponding output.

You have to unzip the file and place the FakeVolcano folder wherever in your file system. The folder PyBetEF. Instructions on how to prepare the input files are given in Input preparation section. The next step is to load the Volcano button in the upper left panel and select the FakeVolcano folder. If everything went okay this example is loaded in the lower left panel and the user can explore it by clicking on the nodes. Once the user can press the COMPUTE button on the lower right panel. A new window frame will be opened, visualizing the corresponding output.

4.3 PyBetUnrest

The procedure is the same shown in 3.1 for PyBetVH

5. Input preparation

Input to feed PyBet tools consists in a set of text files organized in a main folder representing the volcano in which the user is interested. The information is driven by a configuration file called pybet.cfg, which must always exist in order to run the code. The user should not change the names of the various blocks (identified by square brackets) and neither the field tags that are to the left of the sign ‘=’, but only fill in the spaces on its right (for instance, in the block “Main Settings”, the variable name and cannot be changed, while on the right the user can set the values for the desired case study). Details on input preparation to PyBetVH can be found in Tonini et al. (2015) main paper and in the relating supplementary material.
PyBetEF

Example of pybet.cfg

[Main Settings]
volcano name = Fake Volcano
time window = 1
time window = 1
sampling = 1000
background image = filename/None
map limits (m UTM) = 555000, 570000, 5108000, 5123000
monitoring = filename/None
anomaly function parameters = 0.1, 1.0, 1.0

[Node Unrest]
prior probability = 0.5
equivalent n. data (lambda) = 1
past data (successes) = 0
past data (total) = 0

[Node Magmatic]
prior probability = 0.5
equivalent n. data (lambda) = 1
past data (successes) = 0
past data (total) = 0

[Node Magmatic Eruption]
prior probability = 0.5
equivalent n. data (lambda) = 1
past data (successes) = 0
past data (total) = 0

[Node Magmatic Vent Location]
file name = node_vent.txt
file name monitoring = node_vent_monitoring.txt/None

node 4-5 dependence = False
Block "Main Settings"

1. "volcano name =": the user should add volcano name
2. "volcano center =": the user should add longitude and latitude in UTM coordinates (meters), which allows to select the geometry of the possible vent positions. Please add the keyword “Field” for a volcanic field or a caldera, corresponding to a rectangular grid of possible vent locations, or “Cone” for a central geometry characterized by a central vent position over the summit crater area, and lateral sectors

Geometry “Field” is selected in the previous line, the user should input 5 values (comma separated): the longitude length (in meters) of the field, the along-latitude length (in meters) of the field, the number of columns (Nc) to divide the field along the longitude, the number of rows (Nr) to divide the field along the latitude, and the strike (in decimal degrees) from North;

Geometry “Cone” is selected in the previous line, the user should input 3 values (comma separated): the radius of the central vent position above the summit crater area (in meters), the outer radius (to the apron) of the volcano edifice (in meters), the strike (in decimal degrees) from North to rotate clockwise the four 90-degrees outer sectors;

3. "utm zone =": the user should add the UTM zone together with the hemisphere, N=North or S=South (i.e, 33N or 18S);
4. "time window =": the user should add a number that will determine the exposure time. For example, if the user wants to perform an assessment for a 1 year exposure time, he must input the value “1” here, and keep in mind that time unit the user should add the number of samples that PyBetEF will draw at each node of the event tree to assess epistemic uncertainty. Acceptable values (compromise between accuracy and runtime) are between 400 and 2000, but the user could insert any value (integer);
5. "sampling =": the user should add the number of samples that PyBetEF will draw at each node of the event tree to assess epistemic uncertainty. Acceptable values (compromise between accuracy and runtime) are between 400 and 2000, but the user could insert any value (integer);
6. "background map =": the user should enter a name of file (in png format) containing the background image to be overlaid by contour hazard maps or probability map. If keyword “None” is provided, the tool will try to connect to the internet and load a map from Google Maps based on the limits provided in the next item. A good practice could be to let the tool downloading a map from Google Maps the first time, then inserting here the filename of the downloaded map for the further runs of the tool (the current PyBetEF version can load .png image format only);
7. "map limits =": the user should provide SW and NE corners of the background map in UTM coordinates (m);
8. "monitoring =": the user should provide file name where monitoring parameters are set.
9. "anomaly function parameters =": a, b, lambda

Block "Node Unrest" (Node 1)

The user should input the prior probability value (a number between 0 and 1) for the volcano entering eruption. The parameter "equivalent n. data (lambda) =": the user should input a number expressing how much he is confident on the prior probability. This number represents a fictitious number of data he would like to collect before changing significantly his mind on that prior value. Possible values range from 1 (very low confidence on the prior probability value) to (theoretically) infinite; however, a value of 50 implies already a very high confidence on the prior probability value. Please note that the combination “prior probability = 0.5” and “equivalent n. data (lambda) = 1” expresses the maximum ignorance probability distribution (uniform distribution on the domain [0,1])
3. "past data (successes) =": the user should input the number of observed unrest episodes in the available record (that must be complete, meaning that it is reasonable to assume that no unrest has been lost over that time period covered by the record).

4. "past data (total) =": the user should input the number of observed time windows, in the time unit defined in the block “Main Settings” (in the above example, the number of time windows of 1 year), beginning in a state of NO UNREST. The user should discard the time windows where there was already unrest from the beginning.

Block “Node Magmatic” (Node 2)

1. "prior probability =": the user should input the conditional prior probability value (a number between 0 and 1) of the unrest being due to active magma, given unrest (see Marzocchi et al 2008; 2010 for more details).

2. "equivalent n. data (lambda) =": as above but related to the prior probability of magmatic unrest.

3. "past data (successes) =": the user should input the number of unrest episodes of known magmatic origin in the complete record.

4. "past data (total) =": the user should input the number of unrest episodes of known origin (magmatic, hydrothermal, tectonic) in the complete record.

Block “Node Magmatic Eruption” (Node 3)

1. "prior probability =": the user should input the conditional prior probability value (a number between 0 and 1) of the magmatic unrest leading to eruption, given magmatic unrest (see Marzocchi et al 2008; 2010 for more details).

2. "equivalent n. data (lambda) =": as above but related to the prior probability of magmatic unrest leading to eruption.

3. "past data (successes) =": the user should input the number of magmatic unrest episodes having led to an eruption in the complete record.

4. "past data (total) =": the user should input the number of magmatic unrest episodes in the complete record.

Block “Node Magmatic Vent Location” (Node 4)

1. "file name =": the user should input the name of the text file in which the prior spatial probability of vent opening, and past occurrences at each vent position, are stored (e.g., “nodevent.txt”, see below for its structure).

2. "equivalent n. data (lambda) =": as above, but related to the prior spatial probability of vent opening.

Block “Node Magmatic Style” (Node 5)

1. "node 4-5 dependence =": the user should input “False” (if there is no reason to postulate a different probability distribution of the sizes depending on the position for volcanic structures that are partially submerged by the sea, implying a different explosivity depending on the depth under water) or “True” on the contrary (for example for volcanic structures that are partially submerged by the sea, implying a different explosivity depending on the position of the vent or its depth under water).

2. "n. sizes =": the user should input the number of different size classes (Ns) characterising magmatic eruptions.

3. "file name =": the user should input the name of the text file in which the prior probability of the different size classes, and the past occurrences of eruptions of different size classes, are stored (e.g., “node_style.txt”, see below for its structure).

Other files:

- node_vent.txt: in this file, there must be Nv lines, each corresponding to one possible vent position. The number Nv of possible vent positions is implicitly declared in the “Main Settings” block of pybet.cfg file, by defining the geometry. If geometry is “Field”, Nv is equal to Nr*Nc (see above in the block “Main Settings”); if it is “Cone”, Nv is equal to 5 (Central vent position and 4 radial sectors). In each line of the file, there must be 2 numbers: the first one is the prior probability of that vent position, the second one is the number of past eruptions (in a complete dataset) occurred from that vent position. The sum of the prior probabilities over all the Nv vent positions must be 1 (due to the assumption that the Nv vent positions represent a complete and mutually exclusive set).
If the user has selected no dependence of node 5 on node 4 (“node 4-5 dependence = False” in the “Node Style” block of pybet.cfg), in this file there must be 1 line containing 2*Ns+1 numbers:
1. the first Ns numbers are the prior probabilities of the Ns size classes (and the sum must be 1)
2. then there must be the equivalent number of data of such prior probability distribution
3. finally, other Ns numbers corresponding to the number of past eruptions (in a complete dataset) of each size class

If the user has selected a dependence of node 5 on node 4 (“node 4-5 dependence = True” in the “Node Style” block of pybet.cfg), in this file there must be Nv lines, each containing 2*Ns+1 numbers.

Monitoring parameters are organized in a separate file, each one providing the current status of the selected monitored quantities. The path to the this file is given by setting the parameter “monitoring” in the “Main Setting” block of pybet.cfg. Any time the user decides to update the status of monitoring parameters, he/she can create a new file by simply coping the old one and modifying the new values recorded, in order to keep track of the evolution of the forecast.

Example of how a monitoring file has to be formatted:

```
[Unrest - Parameter 1]
name = Number of VT (M > 0.8) [ev/day]
value = 0
threshold 1 = 5
threshold 2 = 20
relation = >
weight = 1

[Unrest - Parameter 2]
name = Maximum magnitude (last month)
value = 0
threshold 1 = 2.5
threshold 2 = 3.0
relation = >
weight = 1

[Magmatic Eruption - Parameter #N]
name = Tremor (last month)
value = 0
threshold 1 = 1
threshold 2 = None
relation = >
weight = 1
```

...
Each block is defined by the node name (i.e., Unrest, Magmatic, …) and the ordinal number of the parameter (i.e., Parameter 1, Parameter 2, …) for each node. Then, it follows a series of rows that define the “fields” that describe the parameter. The following fields are present:

1. "name" is a description of the parameter;
2. "value" is the measure at the time of the assessment (t = t0);
3. "threshold 1" is the smallest value of the interval defining thresholds;
4. "threshold 2" is the largest value of the interval defining thresholds;
5. "relation" defines where anomalous values are, with respect to the interval of thresholds;
6. "weight" defines the information weight of each parameter.

Boolean parameters are considered by setting "threshold 2 = None".

Disclaimer
Even though any volcanic system could theoretically be analysed with a BET tool, the critical role played by hazard assessments in civil protection issues prevents us from leaving the user free to use all the applications. By using or downloading one of the BET tools, the user accepts that INGV and authors are not responsible for any wrong application by the user who takes his/her own responsibility.

References