
Tutorial

AIR-BAGEL: An Interactive Root cause-Based Anomaly Generator for Event Logs

(Link to tutorial video: <https://shorturl.at/wRY04>)

Contents

Before starting	1
Prerequisites	1
Running AIR-BAGEL	1
Virtual Image Appliance	2
Introduction	3
1: Data import and preprocessing	3
2.1: Root parameter setting	4
2.1(A): Parameter setting for “Resource” root	5
2.1(B): Parameter setting for “System” root	7
2.2: Fault injection	9
3: Output evaluation and data export	10
Appendix	13



Before starting

This section contains information regarding how to install AIR-BAGEL on your own machine. If instead you want to run the pre-configured AIR-BAGEL VM, please jump to the next section.

➤ Prerequisites

AIR-BAGEL requires Python 3, tk/tkinter, the Python Imaging Library (PIL), and PM4Py.

1. Python 3 (the tool has been tested with Python 3.6.10 in Windows and Linux/Ubuntu environments; it should be compatible with any more recent versions of Python)
 - a. You can download Python 3 here: <https://www.python.org/downloads/>
2. Install tk/tkinter
 - a. It comes pre-packaged in Anaconda3 and ActivePython 3.x Python installations
 - b. Alternatively:

```
pip install python-tk or
pip3 install python3-tk
```
3. Install Pillow
 - a. `pip install Pillow` or `pip3 install Pillow`
4. Install PM4Py
 - a. Detailed instructions at: <https://pm4py.fit.fraunhofer.de/install>
5. Clone (or copy) the AIR-BAGEL project source code from our Github repository:
 - a. HTTPS: `git clone https://github.com/jonghyeonk/AIR-BAGEL.git`
 - b. SSH: `git clone git@github.com:jonghyeonk/AIR-BAGEL.git`

In a Linux/Ubuntu system, steps 2 and 3 can be substituted by the following commands:

```
sudo apt-get install python3-tk
sudo apt-get install python3-pil
sudo apt-get install python3-pil.imagetk
```

➤ Running AIR-BAGEL

1. Copy your event logs (in csv format) into the `input` directory in the AIR-BAGEL project folder; Note that the project on GitHub already contains a few event logs in the `input` directory for your convenience;
2. In the command prompt, set your current directory to the AIR-BAGEL project folder
3. Run `PageOne.py` (ex: `python PageOne.py` or `python3 PageOne.py` from command prompt, assuming that `python` or `python3` is in your PATH)
4. The starting page of AIR-BAGEL will appear, you are ready to go!



➤ Virtual Image Appliance

A 64-bit Ubuntu 20.04-based virtual machine (VM) pre-configured to run AIR-BAGEL has been prepared. It has been created using the Open Virtualization Format 2.0 (extension .ova), which can be imported by popular hosted hypervisor applications. We have tested Oracle Virtualbox (<https://www.virtualbox.org/>). The instructions and screenshots below refer to importing the AIR-BAGEL VM in Virtualbox 6.0.

- Steps:

1. Download the `air-bagel.ova` image file here (size ~4GB);
: <https://shorturl.at/kmHN2>
2. Open Virtualbox, select "Import Appliance";
3. In the screen "Appliance to Import" select the `air-bagel.ova` file that you have downloaded at Step 1;
4. In the screen "Appliance settings", you can configure the default hardware settings, e.g., how much RAM memory to give to the VM (we suggest at least 4GB for optimal usage).
5. Click "Import".

The import process may take some time. At the end of it, "air-bagel" will appear as one option in the main menu of Virtualbox. Double-click on it to start the VM. No login is required. If needed, both the username and password are "airbagel".

Once you are logged in, you should see the following screen:



Click on the bagel-like icon on the left-hand panel. This will open AIR-BAGEL (and a terminal to see additional logging info). The working directory of the tool is `/home/airbagel/AIR-BAGEL`. Some event logs have been pre-loaded in `/home/airbagel/AIR-BAGEL/input`.

You are ready to go!

Introduction

AIR-BAGEL is a simulation tool to generate pseudo-real trace-level anomalies in event logs. The anomalies, such as deleting, replacing, or moving events in a trace, are caused by two possible root causes, i.e., resource behavior or system malfunctioning. Figure 1 shows the typical usage workflow of AIR-BAGEL: (i) data import and preprocessing, (ii) root parameter setting and fault injection, and (iii) Output evaluation. Next, every step depicted in Figure 1 is presented in detail.

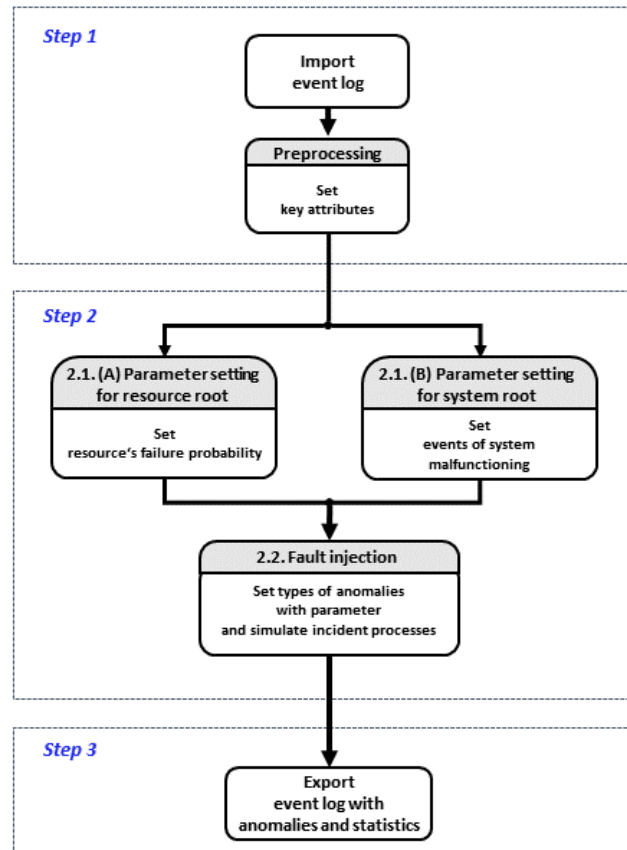


Figure 1 AIR-BAGEL usage workflow

1: Data import and preprocessing

The first step is to import the event log in which you want to inject anomalies. AIR-BAGEL can handle only the event log(s) contained in the directory `<project-folder>/input`. Under “Files” (see Fig.2), you will see a list of event logs currently in the directory `<project-folder>/input`. Select one event log and click **Load**.

Then, in “Environment” you can review the loaded event log. In particular, you have to assign the key attributes to columns in the event log: Case_ID, Event_ID (optional), Activity, and Timestamp. AIR-BAGEL tries to guess the matching, but this can be overridden if needed. For the format of Timestamp, you can select a correct timestamp format from a list, but if there is not any matching format in the list, you have to manually define the timestamp format, e.g., ‘%Y.%m.%d %H:%M:%S’ for the format ‘2020.02.01 10:30:00’. Note that a preview of the loaded event load is given in “Data preview” to facilitate this task.

Quick Guide (see Fig.2)

1. Select and **Load** a dataset (event log)
2. Review attributes in overview of event log in “Data Preview”
3. Complete “Set key attributes”
4. Click **Next**

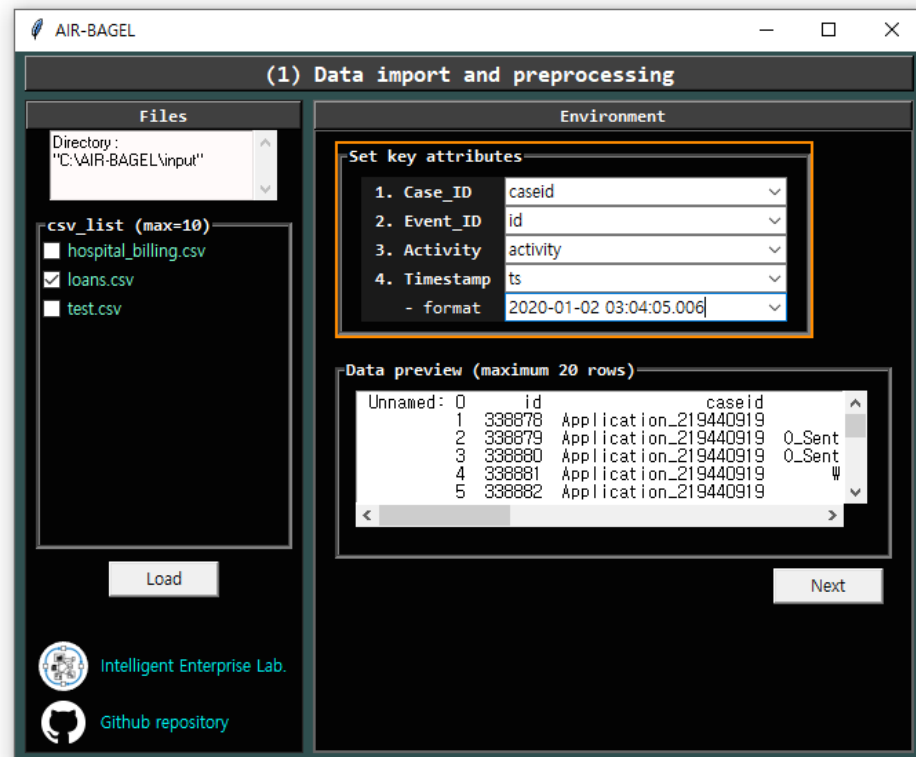


Figure 2 Data import and preprocessing

2.1: Root parameter setting

The second step is for choosing the type of root cause for anomalies to be injected. You can choose Resource or System (or both, but in that case you will have to configure each root cause individually, i.e., repeating the steps described next for each root cause).

For each type of root causes, you must **Set** one option between ‘Random’ and ‘Self-configuration’ (see Fig.3) according to whether the failure probability or system malfunctioning time interval duration for resource and system, respectively, will be set by sampling from a statistical distribution or manually configured. After completing to set parameters on the selected root cause, you can click **Next** to go to the next step to set types of anomaly patterns to apply and start to simulate those. (Note that the details to set parameters on selected root cause and anomaly patterns will be explained in the next page.)

Quick Guide (see Fig.3)

1. Select root causes by ticking the corresponding box
2. For the selected root cause choose ‘Random’ or ‘Self-configuration’
3. Click **Set** to configure parameters of the selected root causes (this opens a new window)
4. (After having configured the parameters of selected root cause) click **Next**

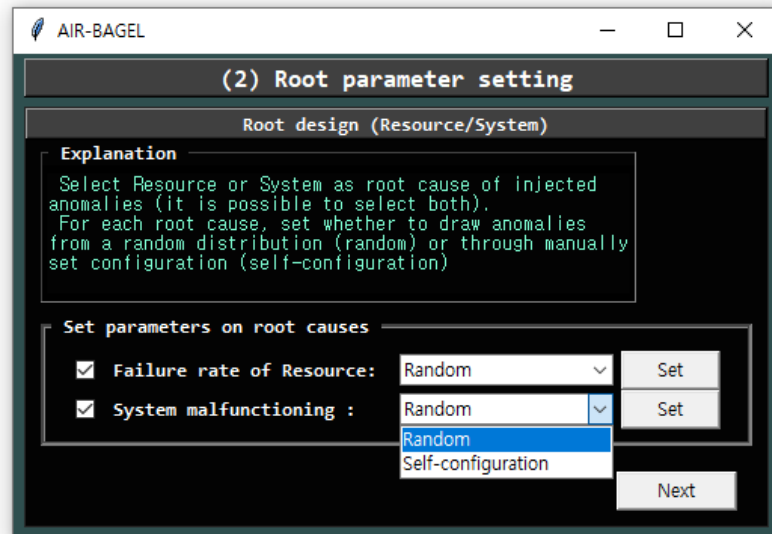


Figure 3 Setting root causes and anomaly patterns

2.1(A): Parameter setting for “Resource” root

Depending on your selection at step 2, you will be shown a different screen to select how to assign the failure probability to each resource. There are two options: randomly sampling from a statistical distribution (exponential, normal or uniform) or by manual configuration. Note that the failure probability of a resource R is the probability that an event executed by R will be injected with anomaly.

In either case, you need first to tell AIR-BAGEL which is the resource attribute in the event log (see Fig.4). If the resource attribute is missing, AIR-BAGEL can generate it for you: you can specify the number and size of resource groups. Resource groups will then be mapped randomly to event classes, i.e., activities (see Fig.5). One resource from each group will be assigned to events based on its class (i.e., activity label). You can then manually modify this configuration if needed. Before injecting anomalies, for each event belonging to certain class, AIR-BAGEL will assign one resource chosen randomly from the corresponding group.

Quick guide: “Random” option

1. Set a resource attribute
 - A. If there is a resource attribute in event log, select it and click **Set** (Fig.4)
 - B. If not, generate an artificial resource attribute
 - i. Set the number of resource groups and click **Set** (Fig.4)
 - ii. Set the size of each resource group (Fig.5)
 - iii. Map each activity (event class) to a resource group (Fig.5)
 - iv. Click **Apply** (Fig.5, you can modify the created mapping manually if needed)
2. Select and parametrize the random distribution
 - Select one distribution among exponential/normal/uniform dist. (Fig.4)
 - If needed, manually specify a new “seed” for sampling random numbers¹

¹ Changing the seed is needed when you want to create multiple anomalous versions of the same log (e.g., 10 anomalous event logs for cross-validation of your anomaly detection method). If the seed is not changed, then all logs configured with the same anomaly injection parameters will contain exactly the same anomalies. An option to change the seed is available for all randomly- generated parameters in AIR-BAGEL.

- Set probability parameters with number ranged from 0 and 1 on selected distribution (Fig.4)
(Note that you can see more details of the random distribution in Appendix)
- 2. Click **Apply**, you can check the resource list updated with failure probability in left-hand frame (Fig.4)
- 3. Click **Close** (Fig.4)

Quick guide: “Self-configuration” option (see Fig.6)

- 1. Set a resource attribute (same as “Random” option”)
- 2. Define failure rate for each resource– This rate is a probability, so it must be between 0 and 1. For instance, 0.5 means that an event executed by this resource has a 50% chance of becoming anomalous.
- 3. Click **Apply**

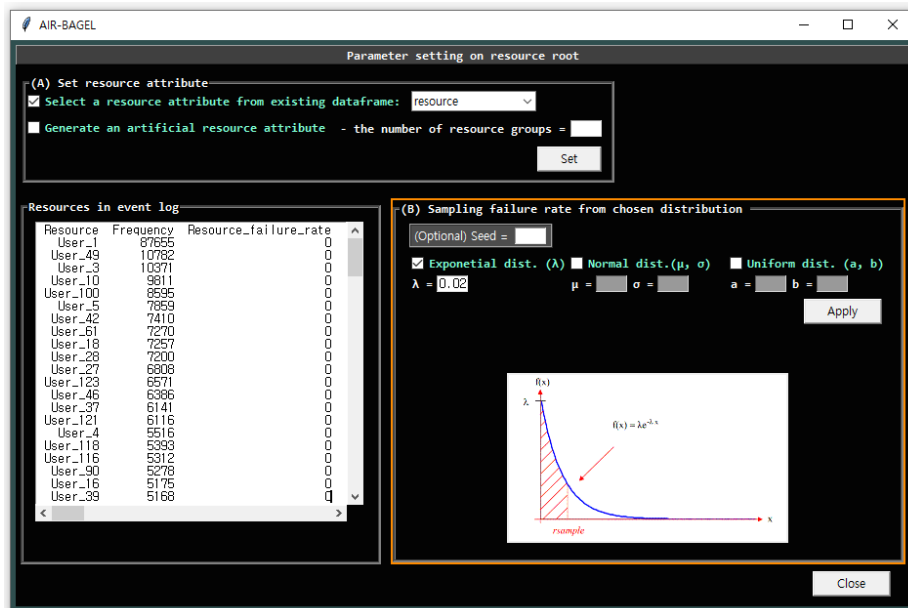


Figure 4 Setting failure rate with 'Random' option

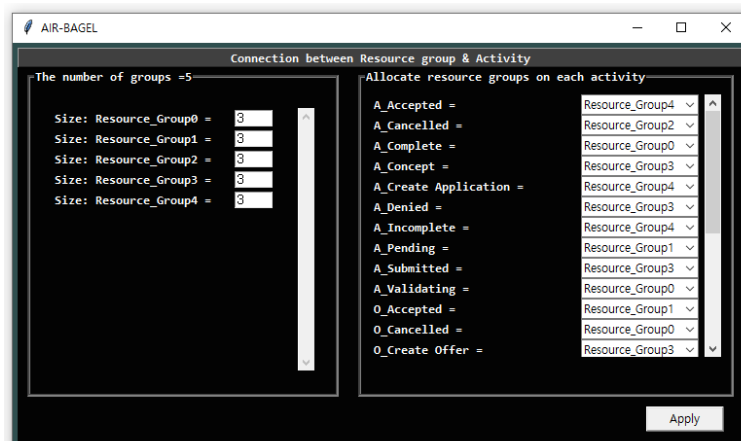


Figure 5 Generating an artificial resource attribute

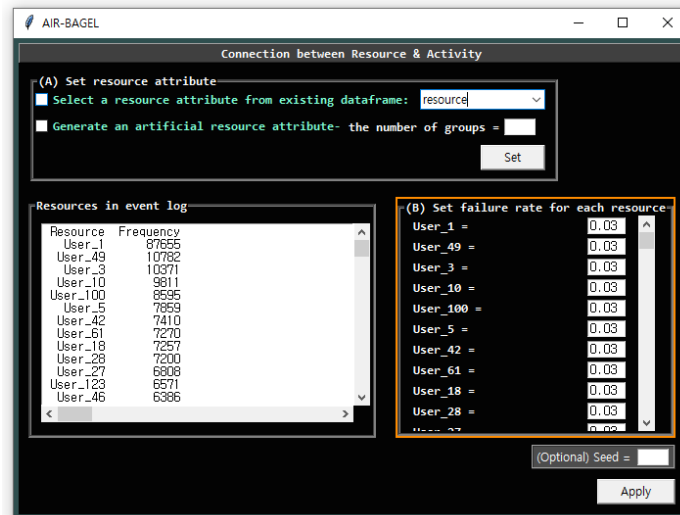


Figure 6 Setting failure rate with 'Self-configuration' option

2.1(B): Parameter setting for “System” root

The objective of this step is to assign one or more intervals of system malfunctioning to the systems in an event log. Depending on your selection at step 2, you will be shown a different screen to select two possible options: 'Random' and 'Self-configuration'. The 'Random' option draws the system malfunctioning intervals from a Poisson process, where the average time interval between the occurrence of system malfunctioning is controlled by a Poisson parameter ($1/\lambda$ days), and the duration of system malfunctioning is defined by two parameters of a Uniform distribution a (minimum time) and b (maximum time) (see Fig.7). In the “self-configuration” option the user can specify one or more system malfunctioning intervals for each system (see Fig.8).

In either case, you need first to tell AIR-BAGEL which is the system, attribute in the event log. If the system attribute is missing, AIR-BAGEL can generate it for you: you can specify the number of systems. System groups will then be mapped randomly to event classes, i.e., activities, so that events with the same activities are always associated with the same system. You can then manually modify this configuration if needed.

Quick Guide: “Random” option

1. Set a system attribute
 - A. If there is a system attribute in event log, select it and click **Set** (Fig.7)
 - B. If not, generate an artificial system attribute
 - i. Check the checkbox button “Generate an artificial system attribute” and click **Set** (Fig.7)
 - ii. Set the number of systems (Fig.8)
 - iii. Map each activity to a system (Fig.8)
 - iv. Click **Apply** (Fig.8, you can modify the created mapping manually if needed)
2. Define system malfunctioning intervals
 - A. Specify a parameter ($1/\lambda$ days) to set the distribution of occurrence of system malfunctioning. Technically, this parameter controls the Start_Timestamp of system malfunctioning (Fig.7)
 - B. Define two parameters a (minimum time) and b (maximum time) to control the duration of a system malfunctioning, which will be drawn uniformly between a and b. In other words, these parameters set the End_Timestamp of a system malfunctioning (Fig.7)
3. Click **Apply** (Fig.7)
4. Click **Close** (Fig.7)

Quick Guide: “Self-configuration” option (see Fig.9)

1. Set a system attribute (same as “Random” option)
2. Configure system malfunctioning events– Here you can manually add system malfunctioning intervals by specifying the system and the start and end timestamp of each malfunctioning.
3. Click **Apply**

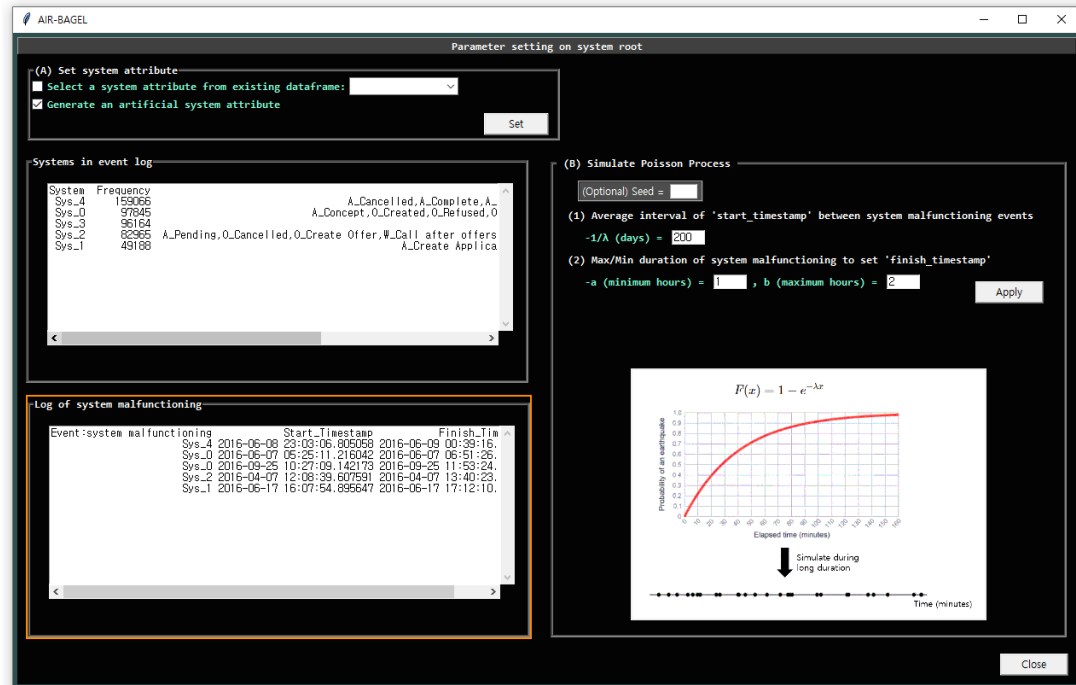


Figure 7 Generating occurrences of system malfunctioning with ‘Random’ option

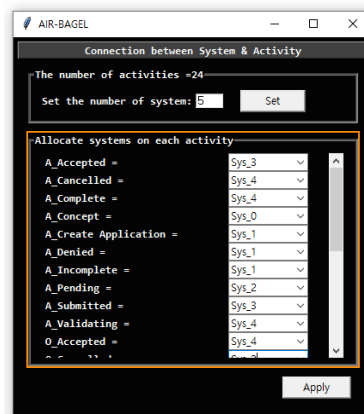


Figure 8 Generating an artificial system attribute

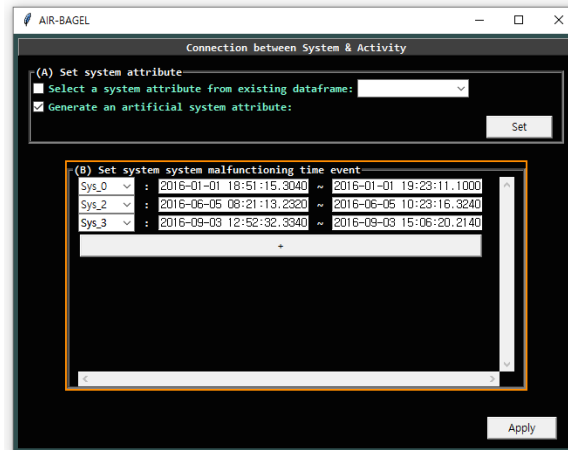


Figure 9 Generating occurrences of system malfunctioning with ‘Self-configuration’ option

2.2: Fault injection (select anomaly types and inject anomalies)

The objective of this step is to define what types of anomalies to apply (when a resource fails or a system malfunctions, as specified in the previous step). There are a total 11 anomaly patterns that can be applied, 8 for the resource root and 3 for the system root (see Fig.10). For each selected pattern, the user specifies a number of configuration parameters (if needed) and a parameter to set the “strength” of the pattern. The strength controls the relative frequency at which the pattern will be applied while injecting anomalies. For instance, if pattern skip has strength 5 and pattern switch has strength 3, then, when an event has been sampled to be anomalous, it is 5/3 times more likely to be skipped than switched. Note that the strength is not mandatory. The default value is 1. Therefore, if strength is not defined for all patterns selected, then every pattern is equally likely to be selected when injecting an anomaly. Once the parameter have been specified, click on Apply to inject the fault. After this process is finished, you will be able to review the results.

Quick Guide (see Fig.10)

1. Select anomaly type(s):
 - A. Set strength of each anomaly pattern to define which type of anomaly patterns will be applied more frequently
 - B. Set parameters on selected patterns:
 - <Resource> Skip: no parameter
 - <Resource> Switch: no parameter
 - <Resource> Replace: no parameter
 - <Resource> Incomplete: no parameter
 - <Resource> Rework: maximum number of times that the event will be replicated.
 - <Resource> Form-based: maximum number of consequent events from the target event that will be recorded at the same time (i.e., will have the same timestamp).
 - <Resource> Insert: maximum number of unrelated events that will be inserted after the target event.
 - <Resource> Moved: maximum absolute value of timestamp delay/anticipation (i.e., a certain amount of time will be added or subtracted from the timestamp of the target event).
 - <System> Skip: no parameter
 - <System> Form based: no parameter
 - <System> Cut: no parameter
2. Click **Apply**
3. Click **See result**

The screenshot shows the AIR-BAGEL application window titled "(3) Fault injection". It contains a "Set anomaly patterns" section with a table for resource patterns and a section for system patterns. Below these are "Apply" and "See result" buttons. At the bottom is a "Progress Status" window showing a log of operations.

Set anomaly patterns

(Optional) Seed =

Pattern (Resource)	Strength	Parameter
<input checked="" type="checkbox"/> skip	<input type="text" value="1"/>	
<input checked="" type="checkbox"/> switch	<input type="text" value="2"/>	
<input checked="" type="checkbox"/> replace	<input type="text" value="3"/>	
<input type="checkbox"/> incomplete	<input type="text"/>	
<input checked="" type="checkbox"/> rework	<input type="text" value="1"/>	Max_length = <input type="text" value="3"/>
<input type="checkbox"/> form based	<input type="text"/>	Max_length = <input type="text"/>
<input type="checkbox"/> insert	<input type="text"/>	Max_length = <input type="text"/>
<input checked="" type="checkbox"/> moved	<input type="text" value="2"/>	Max: Y = <input type="text"/> m = <input type="text"/> d = <input type="text"/> h = <input type="text" value="5"/> M = <input type="text"/>

Pattern (System)	Strength
<input checked="" type="checkbox"/> skip	<input type="text" value="1"/>
<input type="checkbox"/> form based	<input type="text"/>
<input checked="" type="checkbox"/> cut	<input type="text" value="2"/>

Apply See result

Progress Status

```

<Resource> Finished skip (5/5)
<Resource> Finished to inject anomaly patterns (running time=0:00:38.029258)
<System> Started preprocessing to set the input of anomaly patterns
<System> Finished preprocessing (running time=0:00:20.436277)
<System> Finished to inject anomaly patterns
<System> Started skip (1/2)
<System> Finished skip (1/2)
<System> Finished cut (2/2)
<System> Finished to inject anomaly patterns (running time=0:00:29.781398)
Concatenating dataset
  
```

Figure 10 Setting anomaly patterns and parameters

3: Output evaluation and data export

The last step shows a summary of the anomalies injected in the event log. In 'Anomaly frequency' (see Fig 11), the frequency and ratio of each anomaly patterns is reported. The right-hand side ('Root information') shows statistics about root causes, including mean/max/min failure probability of resources and/or mean/max/min system malfunctioning intervals.

Users can visualize or download the two process models (Petri nets) discovered using the inductive miner, one from the original event log and one from the event log injected with anomalies. These models may help giving a first visual cue of the effect of anomaly injections on the discovered process models.

Finally, the event log with injected anomalies can be exported (in csv format). After clicking Download, the event log with anomalies will be downloaded in the directory <project-folder>/output.

The event log with anomalies is augmented with additional attributes. These are presented in detail next.

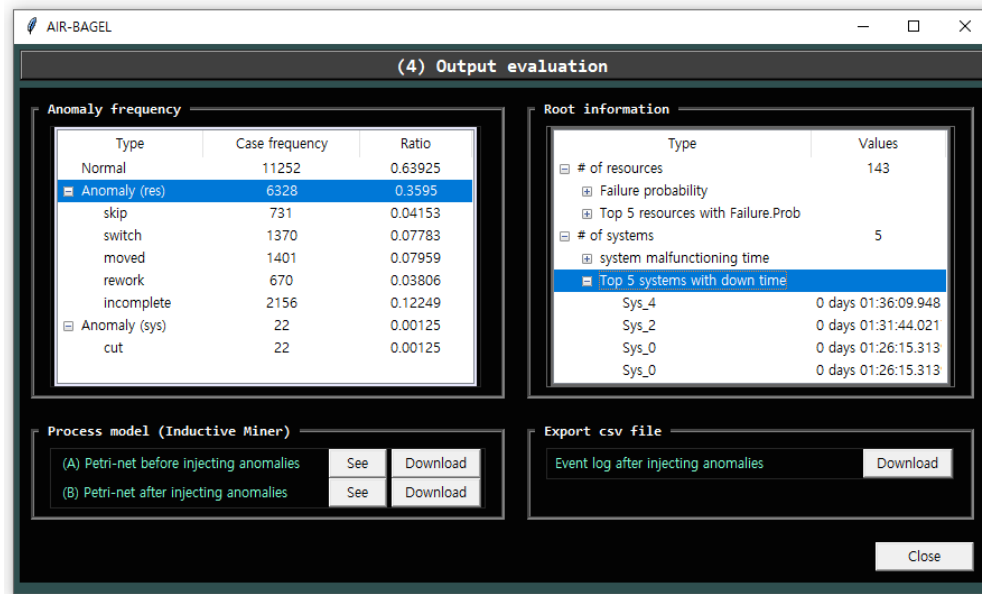


Figure 11 Output evaluation

➤ Attributes in exported data

(in *italic* the attributes added by AIR-BAGEL – note that Resource/System additional attributes are added only if resource or system root anomalies have been injected; case-level additional attributes are the ones assuming the same value for all events in a trace. An example is shown in Table 1)

- **Case**
- **Event**
- **Activity**
- **Timestamp**
- **order**: progressive order of this event in the corresponding trace
- **variant_num**: type of trace (variant)
- **Resource**
- **Resource_failure_rate (event-level)**: probability of introducing an anomaly for that resource
- **Resource_Pass/Fail (event-level)**: 0 (normal) or 1 (anomaly) value from a binomial distribution defined by the resource_failure_rate (that is, whether this event has been made anomalous or not as a result of resource failure rate)
- **resource_anomaly_type (case-level)**: anomaly pattern from resource root applied to the trace to which the event belongs, e.g., “skip” if an event in the trace was skipped
- **resource_parameter (case_level)**: details of applied anomaly pattern, e.g., applied timestamp delay/anticipation and event_id for ‘moved’ pattern, location and length of skipped event(s) for ‘skip’ pattern. This information may be useful for testing event log reconstruction, where the objective is to understand the type of anomaly;
- **is_trace_anomalous_resource (case-level)**: 1 or 0 label capturing whether the corresponding trace has become anomalous as a result of applying anomalies (note in fact that, in some cases, an anomaly injected at event level may not result in an anomalous trace, i.e., the order of events may remain the same!)
- **System**
- **malfunc_duration (event-level)**: duration of system malfunctioning
- **sys_anomaly_type (case_level)**: anomaly pattern injected from system root, e.g., “cut_to” if the case has been cut into 2 different cases as a result of applying the system root anomalies.
- **sys_parameter (case_level)**: details of applied anomaly pattern, e.g., start/finish/ duration of system malfunctioning for ‘skip’ or ‘form-based’ pattern;
- **is_trace_anomalous_system (case-level)**: 1 or 0 value depending whether the system root anomalies have made the trace to which this event belongs anomalous or not.
- **anomaly_type (case_level)**: comprehensive anomaly label from two root causes, e.g.,

“skip(res), cut_to(sys)” if event(s) in a trace have been skipped and the trace has been cut as a result of applying anomaly patterns.

Table 12: An example of case in which 8th event has been skipped

Case	Activity	Order	Resource	resource_anomaly_type	resource_parameter	Is_trace_anomalous_resource
Application_124569708	A_Create Application	1	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Create Application	2	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Concept	3	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Accepted	4	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	O_Created	5	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	O_Sent (mail and online)	6	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	O_Sent (mail and online)	7	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	W_Call after offers	8	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Complete	9	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Complete	10	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Complete	11	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Complete	12	User_38	skip	loc = 8, len = 1.0	1

Event at location 8 in this trace has been skipped as a result of anomaly injection

Appendix

➤ Sampling from statistical distribution (exponential/normal/ uniform distribution)

Here we give simple examples to support users in choosing the parameter values of statistical distribution in *step 2.1.(A)*. Let us assume that there are 1,000 resources in an event log. If you select the exponential distribution and set the average failure probability to $\lambda = 0.02$, 1000 random values of which the average value converges to $\lambda = 0.02$ will be sampled and one value will be mapped to each different resources. Fig 12(a) shows a histogram of one realization of this sampling.

If the normal distribution is chosen, you have to define the average (μ) and the standard deviation (σ) of failure probability, and, for the uniform distribution, the minimum value (a) and maximum value (b) are required. Similarly to what described for the exponential distribution, AIR-BAGEL will sample a number of values from these distribution equal to the number of resources in the log and map each value to a different resources. Fig 12(b) and (c) shows histogram of one realization of this sampling from the normal and uniform distribution, respectively.

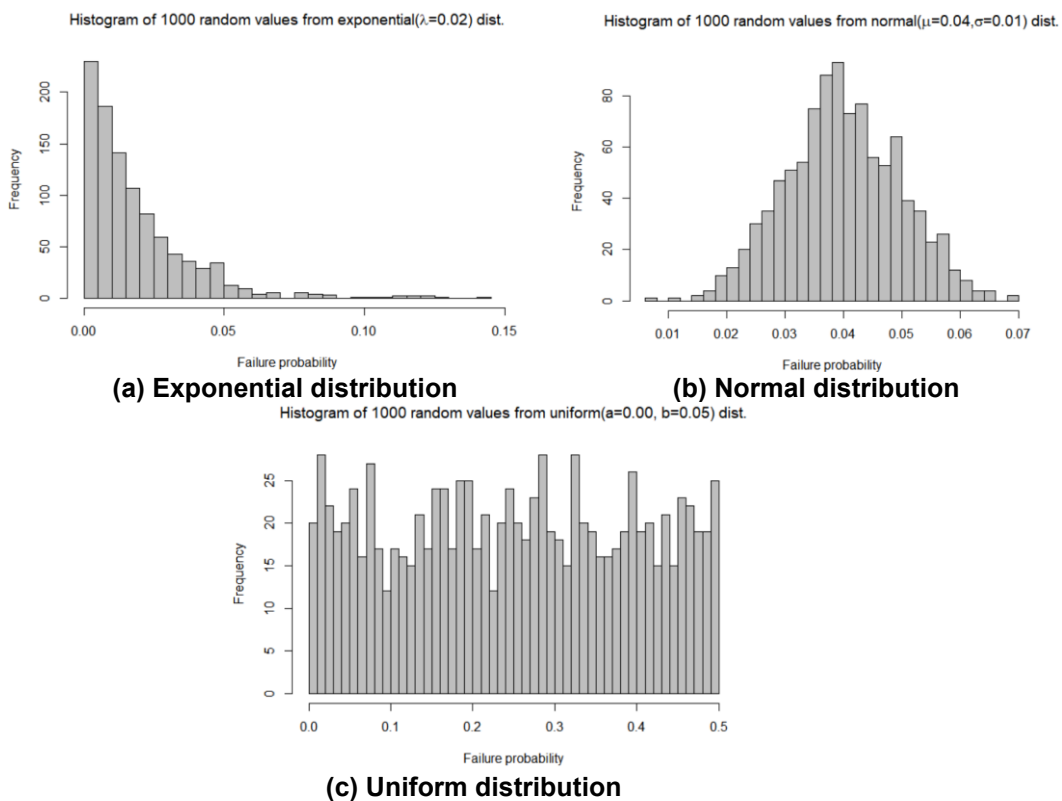


Figure 132 Histograms of random numbers from statistical distribution