Link: https://github.com/jonghyeonk/AIR-BAGEL

Tutorial

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

(Link to tutorial video: https://drive.google.com/file/d/1ooY1gTM9xjiSCpxnctxkvPBe6agvZSME/view?usp=sharing)

Contents

Link: https://github.com/jonghyeonk/AIR-BAGEL

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!

Link: https://github.com/jonghyeonk/AIR-BAGEL

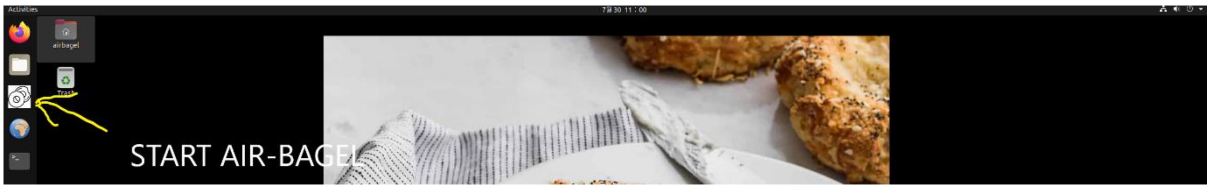
➢ Virtual Image Appliance

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://drive.google.com/file/d/1YU2Xv2_IQN42UDrV1V_5DrGqM4jJYufJ/view?usp=sharing
  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:



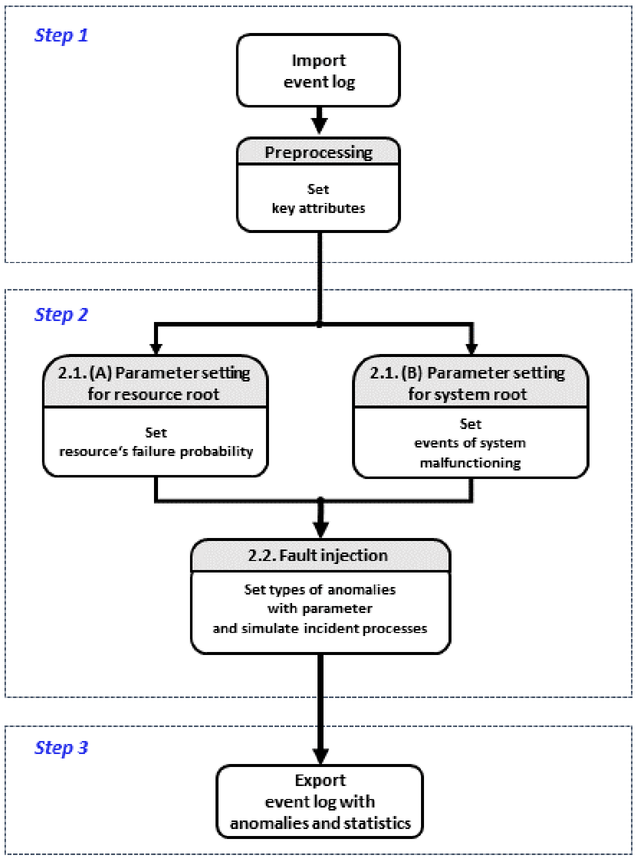
Click on the bagel-like icon on the left-hand panel. This will open AIR-BAGEL (and a terminal to see additional logging info). The working directory of the tool is `/home/airbagel/AIR-BAGEL`. Some event logs have been pre-loaded in `/home/airbagel/AIR-BAGEL/input`.

You are ready to go!

Link: https://github.com/jonghyeonk/AIR-BAGEL

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.



Step 1
Import event log
Preprocessing
Set key attributes
Step 2
2.1. (A) Parameter setting for resource root
Set resource's failure probability
2.1. (B) Parameter setting for system root
Set events of system malfunctioning
2.2. Fault injection
Set types of anomalies with parameter and simulate incident processes
Step 3
Export event log with anomalies and statistics

Figure 1 AIR-BAGEL usage workflow

1: Data import and preprocessing

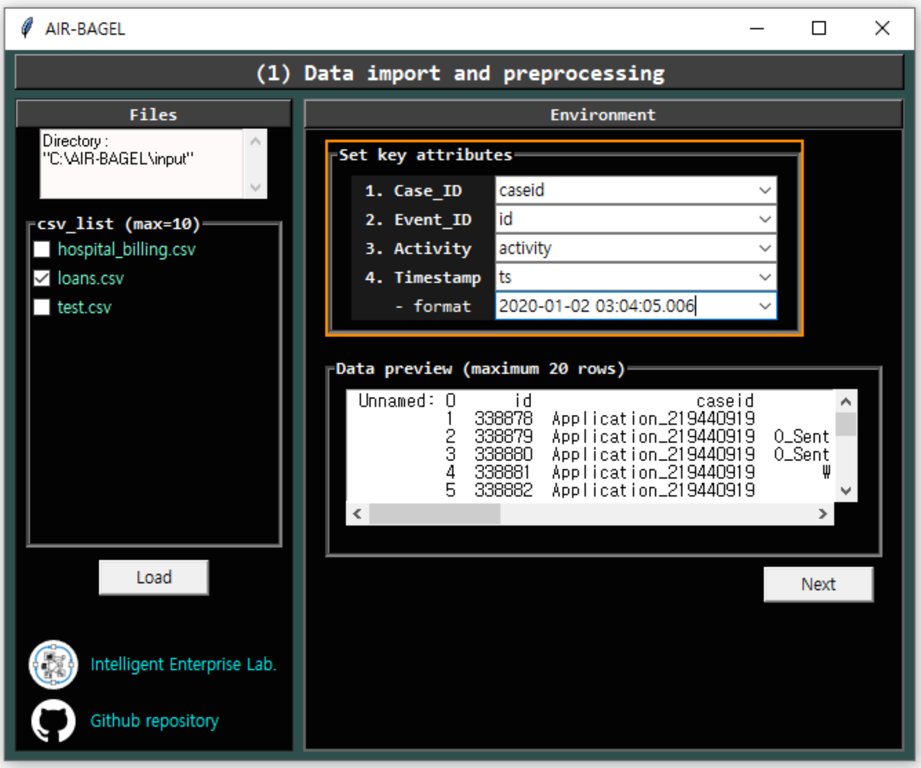
The first step is to import the event log in which you want to inject anomalies. AIR-BAGEL can handle only the event log(s) contained in the directory `<project-folder>/input`. Under "Files" (see Fig.2), you will see a list of event logs currently in the directory `<project-folder>/input`. Select one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

Link: https://github.com/jonghyeonk/AIR-BAGEL

Quick Guide (see Fig.2)

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


AIR-BAGEL
(1) Data import and preprocessing
Files
Environment
Load
Next
Intelligent Enterprise Lab.
Github repository

Directory : "C:\AIR-BAGEL\input"

csv_list (max=10)
- [ ] hospital_billing.csv
- [x] loans.csv
- [ ] test.csv

Set key attributes

1. Case_ID	caseid
2. Event_ID	id
3. Activity	activity
4. Timestamp	ts
- format	2020-01-02 03:04:05.006

Data preview (maximum 20 rows)

```
Unnamed: 0      id                    caseid
         1  338878  Application_219440919
         2  338879  Application_219440919  O_Sent
         3  338880  Application_219440919  O_Sent
         4  338881  Application_219440919       ₩
         5  338882  Application_219440919
```

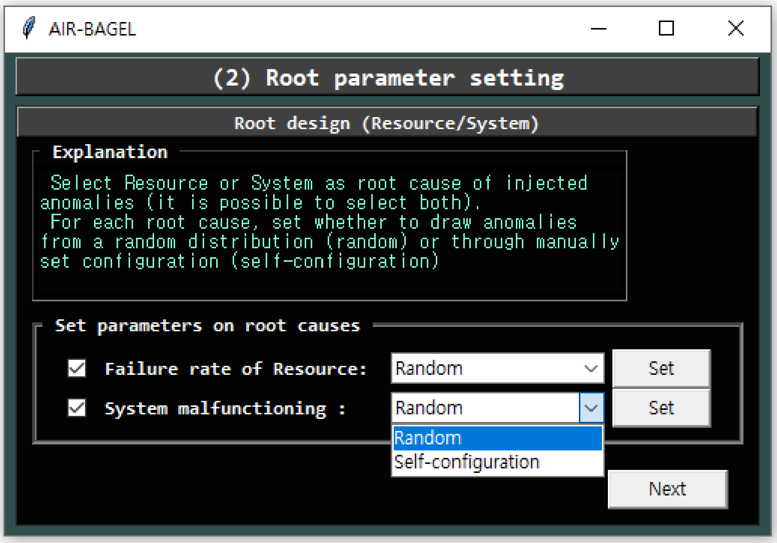
Figure 2 **Data import and preprocessing**

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 case individually, i.e., repeating the steps described next for each root cause).
For each type of root causes, you must **<u>Set</u>** 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 **<u>Next</u>** 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 **<u>Set</u>** to configure parameters of the selected root causes (this opens a new window)
4. (After having configured the parameters of selected root cause) click **<u>Next</u>**


AIR-BAGEL

(2) Root parameter setting

Root design (Resource/System)

Explanation

Select Resource or System as root cause of injected anomalies (it is possible to select both).
For each root cause, set whether to draw anomalies from a random distribution (random) or through manually set configuration (self-configuration)

Set parameters on root causes

✓ Failure rate of Resource: Random

✓ System malfunctioning : Random

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 numbers[1]

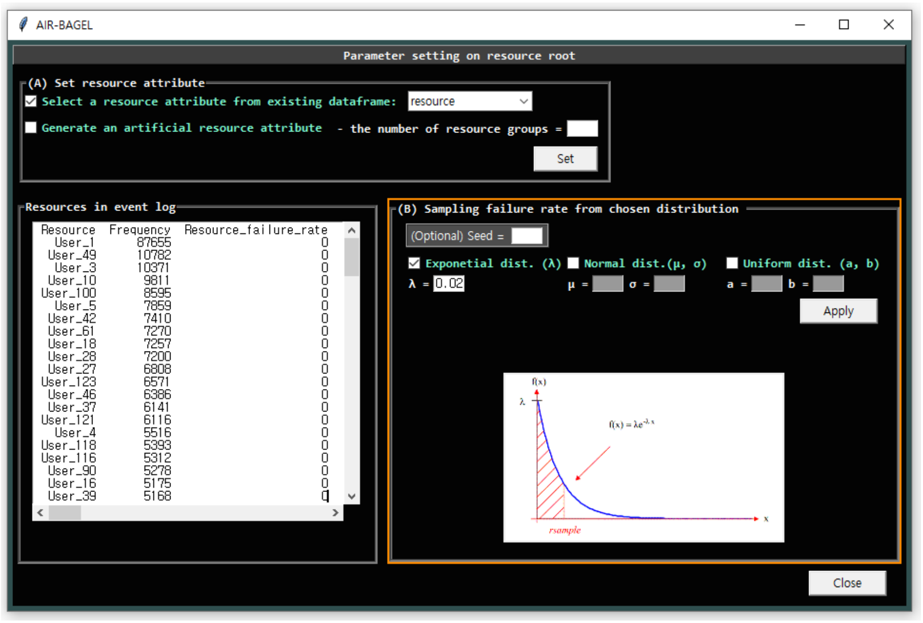
[1] Changing the seed is needed when you want to create multiple anomalous versions of the same log (e.g., 10 anomalous event logs for cross-validation of your anomaly detection method). If the seed is not changed, then all logs configured with the same anomaly injection parameters will contain exactly 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**



AIR-BAGEL

Parameter setting on resource root

(A) Set resource attribute

☑ Select a resource attribute from existing dataframe: resource

☐ Generate an artificial resource attribute - the number of resource groups =

Set

Resources in event log

```
Resource  Frequency  Resource_failure_rate
  User_1     87655                     0
 User_49     10782                     0
  User_3     10371                     0
 User_10      9811                     0
User_100      8595                     0
  User_5      7859                     0
 User_42      7410                     0
 User_61      7270                     0
 User_18      7257                     0
 User_28      7200                     0
 User_27      6808                     0
User_123      6571                     0
 User_46      6386                     0
 User_37      6141                     0
User_121      6116                     0
  User_4      5516                     0
User_118      5393                     0
User_116      5312                     0
 User_90      5278                     0
 User_16      5175                     0
 User_39      5168                     0
```

(B) Sampling failure rate from chosen distribution

(Optional) Seed =

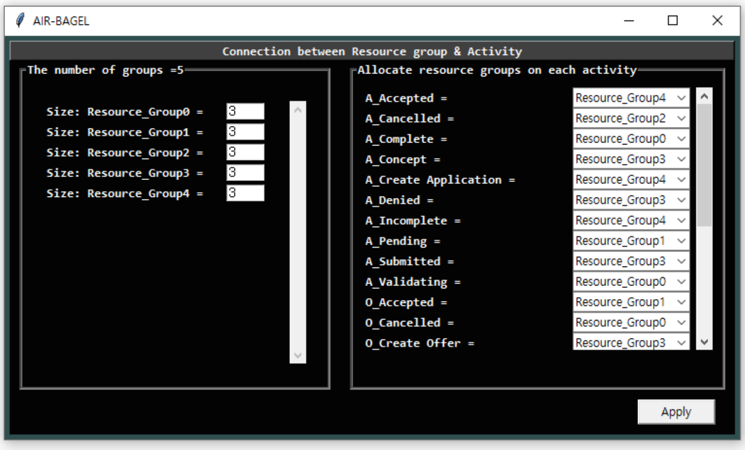
☑ Exponetial dist. (λ) ☐ Normal dist.(μ, σ) ☐ Uniform dist. (a, b)

λ = 0.02 μ = σ = a = b =

Apply

Close

Figure 4 Setting failure rate with 'Random' option



AIR-BAGEL

Connection between Resource group & Activity

The number of groups =5

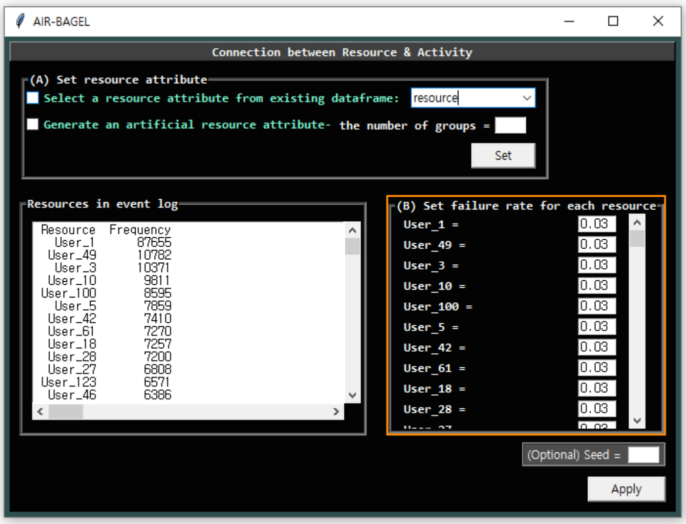
Size: Resource_Group0 =	3
Size: Resource_Group1 =	3
Size: Resource_Group2 =	3
Size: Resource_Group3 =	3
Size: Resource_Group4 =	3

Allocate resource groups on each activity

A_Accepted =	Resource_Group4
A_Cancelled =	Resource_Group2
A_Complete =	Resource_Group0
A_Concept =	Resource_Group3
A_Create Application =	Resource_Group4
A_Denied =	Resource_Group3
A_Incomplete =	Resource_Group4
A_Pending =	Resource_Group1
A_Submitted =	Resource_Group3
A_Validating =	Resource_Group0
O_Accepted =	Resource_Group1
O_Cancelled =	Resource_Group0
O_Create Offer =	Resource_Group3

Apply

Figure 5 Generating an artificial resource attribute


AIR-BAGEL

Connection between Resource & Activity

(A) Set resource attribute

☐ Select a resource attribute from existing dataframe: resource

☐ Generate an artificial resource attribute- the number of groups =

Set

Resources in event log

```
Resource  Frequency
  User_1     87655
 User_49     10782
  User_3     10371
 User_10      9811
User_100      8595
  User_5      7859
 User_42      7410
 User_61      7270
 User_18      7257
 User_28      7200
 User_27      6808
User_123      6571
 User_46      6386
```

(B) Set failure rate for each resource

User_1 =	0.03
User_49 =	0.03
User_3 =	0.03
User_10 =	0.03
User_100 =	0.03
User_5 =	0.03
User_42 =	0.03
User_61 =	0.03
User_18 =	0.03
User_28 =	0.03

(Optional) Seed =

Apply

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

2.1(B): Parameter setting for "System" root

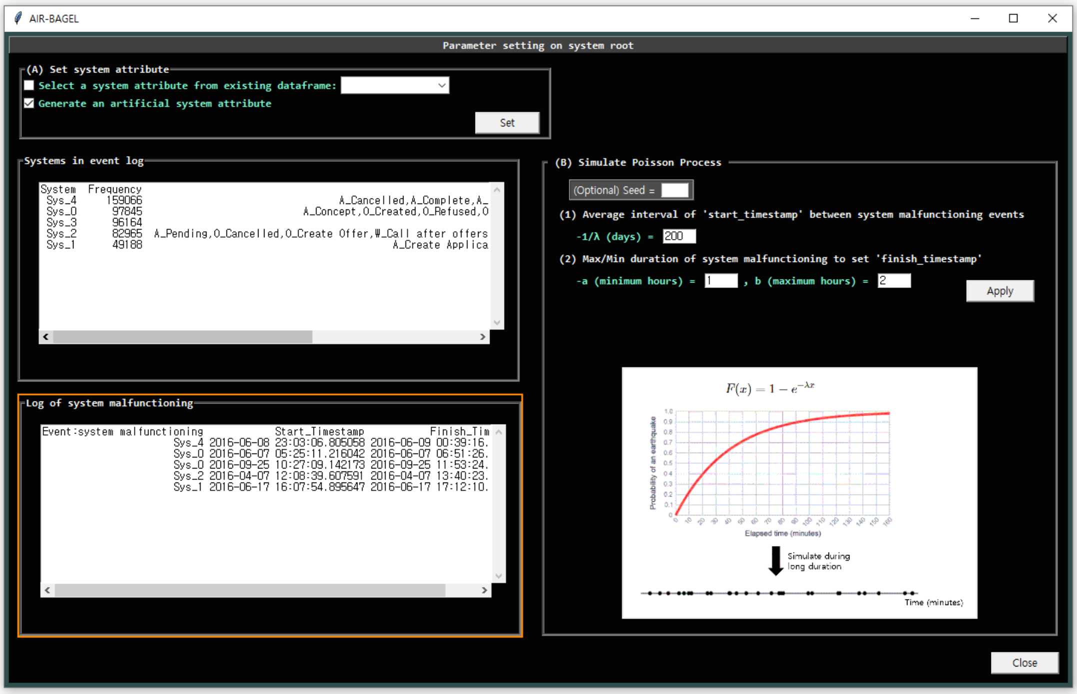
The objective of this step is to assign one or more intervals of system malfunctioning to the systems in an event log. Depending on your selection at step 2, you will be shown a different screen to select two possible options: 'Random' and 'Self-configuration'. The 'Random" option draws the system malfunctioning 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**


AIR-BAGEL
Parameter setting on system root
Set
Apply
Close
Simulate during long duration
Probability of an earthquake
Elapsed time (minutes)
Time (minutes)

(A) Set system attribute
- [ ] Select a system attribute from existing dataframe:
- [✓] Generate an artificial system attribute

Systems in event log

```
System  Frequency
 Sys_4     159066                                  A_Cancelled,A_Complete,A_
 Sys_0      97845                            A_Concept,O_Created,O_Refused,O
 Sys_3      96164
 Sys_2      82965  A_Pending,O_Cancelled,O_Create Offer,W_Call after offers
 Sys_1      49188                                             A_Create Applica
```

Log of system malfunctioning

```
Event:system malfunctioning              Start_Timestamp            Finish_Tim
                  Sys_4 2016-06-08 23:03:06.805058 2016-06-09 00:39:16.
                  Sys_0 2016-06-07 05:25:11.216042 2016-06-07 06:51:26.
                  Sys_0 2016-09-25 10:27:09.142173 2016-09-25 11:53:24.
                  Sys_2 2016-04-07 12:08:39.607591 2016-04-07 13:40:23.
                  Sys_1 2016-06-17 16:07:54.895647 2016-06-17 17:12:10.
```

(B) Simulate Poisson Process

(Optional) Seed =

(1) Average interval of 'start_timestamp' between system malfunctioning events

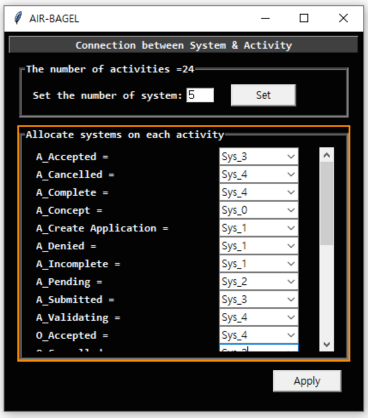
-1/λ (days) = 200

(2) Max/Min duration of system malfunctioning to set 'finish_timestamp'

-a (minimum hours) = 1 , b (maximum hours) = 2

$F(x) = 1 - e^{-\lambda x}$

Figure 7 Generating occurrences of system malfunctioning with 'Random' option



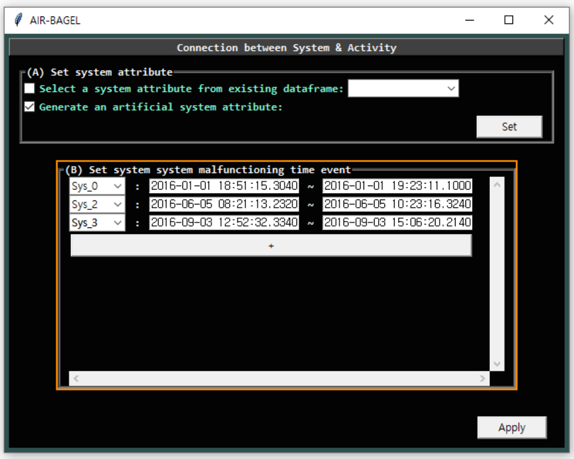
The number of activities =24

Set the number of system: 5

Allocate systems on each activity

A_Accepted =	Sys_3
A_Cancelled =	Sys_4
A_Complete =	Sys_4
A_Concept =	Sys_0
A_Create Application =	Sys_1
A_Denied =	Sys_1
A_Incomplete =	Sys_1
A_Pending =	Sys_2
A_Submitted =	Sys_3
A_Validating =	Sys_4
O_Accepted =	Sys_4

Figure 8 Generating an artificial system attribute


AIR-BAGEL

Connection between System & Activity

(A) Set system attribute
- [ ] Select a system attribute from existing dataframe:
- [x] Generate an artificial system attribute:

(B) Set system system malfunctioning time event

Sys_0	:	2016-01-01 18:51:15.3040	~	2016-01-01 19:23:11.1000
Sys_2	:	2016-06-05 08:21:13.2320	~	2016-06-05 10:23:16.3240
Sys_3	:	2016-09-03 12:52:32.3340	~	2016-09-03 15:06:20.2140

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

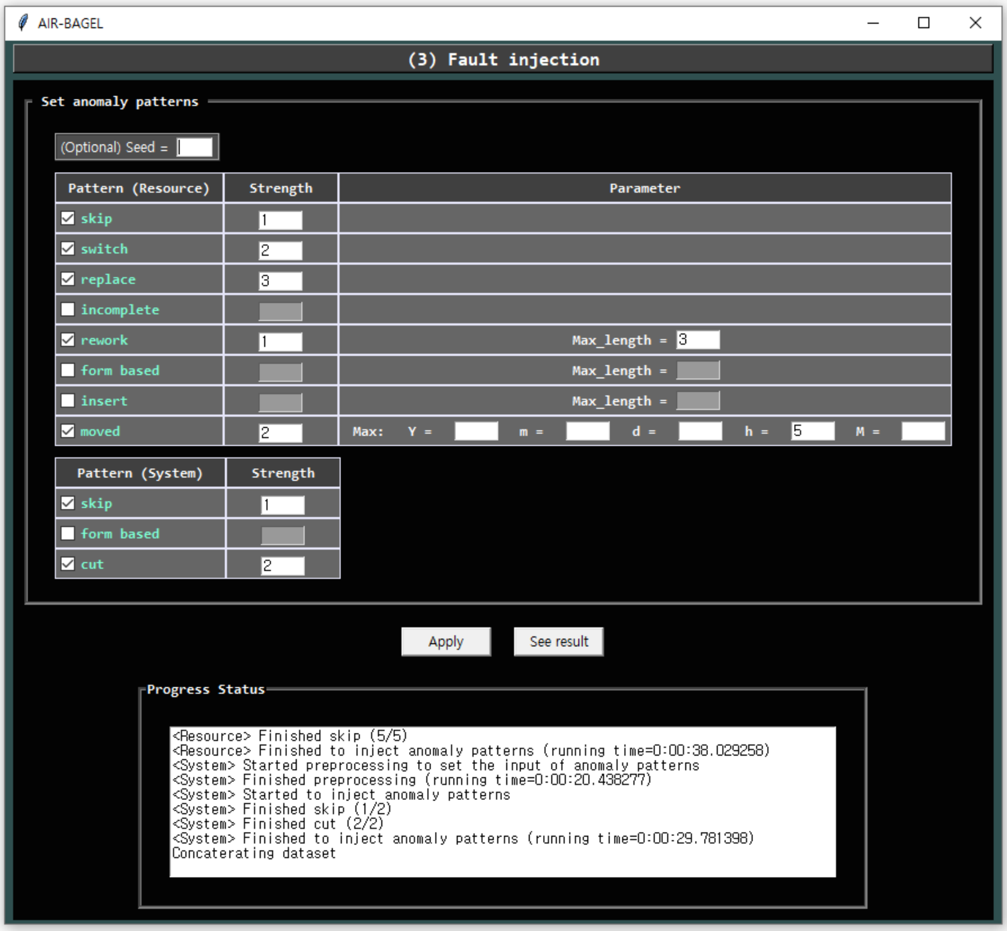
2.2: Fault injection (select anomaly types and inject anomalies)

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**



Set anomaly patterns

(Optional) Seed =

Pattern (Resource)	Strength	Parameter
✓ skip	1	
✓ switch	2	
✓ replace	3	
☐ incomplete		
✓ rework	1	Max_length = 3
☐ form based		Max_length =
☐ insert		Max_length =
✓ moved	2	Max: Y = m = d = h = 5 M =

Pattern (System)	Strength
✓ skip	1
☐ form based	
✓ cut	2

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=0:00:20.438277)
<System> Started to inject anomaly patterns
<System> Finished skip (1/2)
<System> Finished cut (2/2)
<System> Finished to inject anomaly patterns (running time=0:00:29.781398)
Concaterating dataset
```

Figure 10 Setting anomaly patterns and parameters

3: Output evaluation and data export

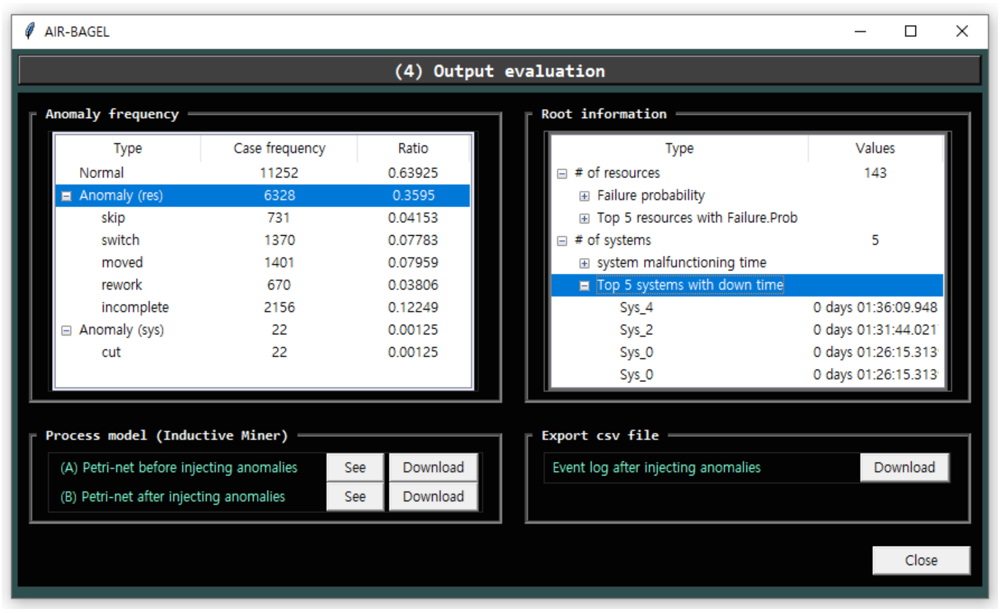
The last step shows a summary of the anomalies injected in the event log. In 'Anomaly frequency' (see Fig 11), the frequency and ratio of each anomaly patterns is reported. The right-hand side ('Root information') shows statistics about root causes, including mean/max/min failure probability of resources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

Link: https://github.com/jonghyeonk/AIR-BAGEL



AIR-BAGEL

(4) Output evaluation

Anomaly frequency

Type	Case frequency	Ratio
Normal	11252	0.63925
⊟ Anomaly (res)	6328	0.3595
skip	731	0.04153
switch	1370	0.07783
moved	1401	0.07959
rework	670	0.03806
incomplete	2156	0.12249
⊟ Anomaly (sys)	22	0.00125
cut	22	0.00125

Root information

Type	Values
⊟ # of resources	143
⊞ Failure probability	
⊞ Top 5 resources with Failure.Prob	
⊟ # of systems	5
⊞ system malfunctioning time	
⊟ Top 5 systems with down time	
Sys_4	0 days 01:36:09.948
Sys_2	0 days 01:31:44.021
Sys_0	0 days 01:26:15.313
Sys_0	0 days 01:26:15.313

Process model (Inductive Miner)

(A) Petri-net before injecting anomalies	See	Download
(B) Petri-net after injecting anomalies	See	Download

Export csv file

Event log after injecting anomalies	Download

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

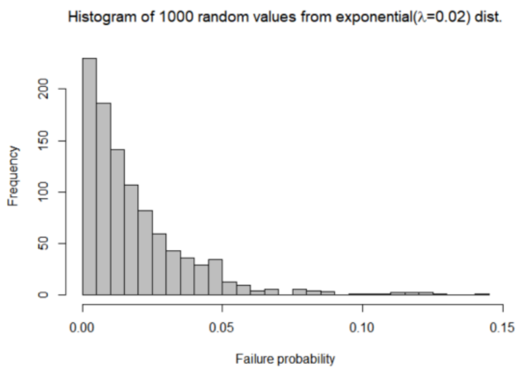
Case	Activity	*Order*	Resource	*resource_ anomaly_ type*	*resource_parameter*	*Is_trace_anomalous _resource*
Application_124569708	A_Create Application	1	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Create Application	2	User_38	skip	loc = 8, len = 1.0	1
Application_124569708	A_Concept	3	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

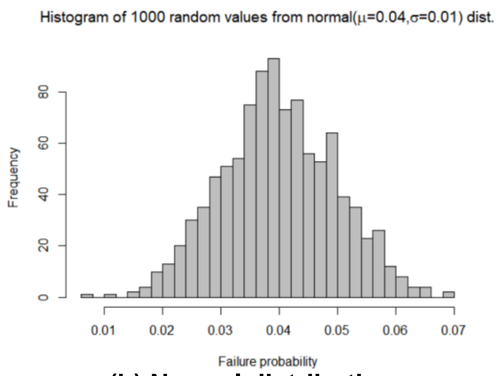
Appendix

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

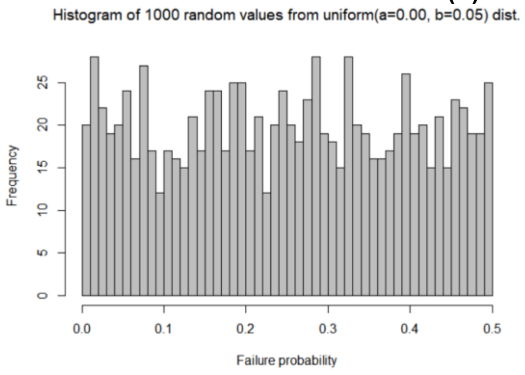
Here we give simple examples to support users in choosing the parameter values of statistical distribution in *step 2.1.(A).* Let us assume that there are 1,000 resources in an event log. If you select the exponential distribution and set the average failure probability to $\lambda = 0.02$, 1000 random 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.



(a) Exponential distribution



(b) Normal distribution



(c) Uniform distribution

Figure 132 Histograms of random numbers from statistical distribution