PUMPKIN USER’S GUIDE

Contributors: Reginald Cushing

Pumpkin framework is the follow up of the Datafluo framework, having the same objective: Support researchers to run complex applications on distributed and federated computing resources.
# Table of Content

0. Download/Deploy and Run the Hello World example................................................................. 3

1. Sensitivity analysis and Pumpkin.................................................................................................. 4

2. Atomic service “SA_tools” ........................................................................................................... 5

3. Pumpkin framework ..................................................................................................................... 6
   3.1. Input data: .......................................................................................................................... 7
   3.2. Seeds...................................................................................................................................... 7
   3.3. Data Packet structure .......................................................................................................... 8
   3.4. Configure and initialize the data processing ....................................................................... 9

4. Frequently Asked Questions ...................................................................................................... 10
   4.1 How can I run the default sensitivity analysis with the default settings? ......................... 10
   4.2. How can I change the model of the sensitivity analysis with my own model? ................. 10
   4.3. How can I define a new workflow other than the sensitivity analysis workflow? .......... 10
   4.4. How can I create new seeds in Pumpkin? .......................................................................... 10
Pumpkin is a framework that helps you to process your data by simply describing which process(es) should be applied to transform your data from one state to another. For example a Hello World application in Pumpkin would typically include 3 tasks: a task that inputs or injects the data in this case the string *world*, a second task that processes the data, it greets world with a *hello* and the final task for extracting or output the data *hello world*.

![State Diagram](image)

The above can also be described as a data centric state transition diagram where the emphasis is on the description of the data. The figure below illustrates the state diagram for the above.

![State Diagram](image)

0. Download/Deploy and Run the Hello World example

- Download Pumpkin
  - git clone https://github.com/recap/pumpkin.git
- make sure these python packages are also installed
  - argparse, BaseHTTPServer, cmd, collections, ConfigParser, copy, cProfile, pstats, errno, fcntl, hashlib, imp, inspect, json, logging, networkx, pyinotify, Queue, re, shelve, struct, tarfile, tftp, uuid, zmq
- run the test example
  - ./bin/<pmk –taskdir>/examples/helloworld/

If all goes well you should get something like:

```plaintext
INFO:Node assigned UID: slimme-72ceeb4
INFO:Exec context: ff0f1929
INFO:Node bound to IP: 145.18.166.194
INFO:Starting thread: ZMQBroadcaster
INFO:Starting broadcast listener on port 7700
Starting FTP server
INFO:Initialised seed: extract
WARNING:Class "extract" called on_load but not implemented.
INFO:Initialised seed: greet
WARNING:Class "greet" called on_load but not implemented.
INFO:Initialised seed: inject
WARNING:Class "inject" called on_load but not implemented.
INFO:Discovered new peer: extract at inproc://slimme-72ceeb4f
INFO:Discovered new peer: inject at inproc://slimme-72ceeb4f
INFO:Discovered new peer: greet at inproc://slimme-72ceeb4f
Greeting: Hello World
```
Further info including how to inject data packets and distribute Pumpkin refer to https://github.com/recap/pumpkin/blob/master/README.md

1. Sensitivity analysis and Pumpkin

To explain how Pumpkin works we will use the sensitivity analysis applications developed in the VPH-share project. The sensitivity analysis is composed of a number of tasks with data dependencies (workflow). Some of these tasks are CPU intensive and will take a lot of time to process on a single computer. Hopefully, these tasks allow for data parallelism, which makes it easy to run parallel tasks and take advantage of available Cloud and Grid resources. Hopefully these tasks expose data parallelism, which make it easy to run parallel tasks and take advantage of available Cloud and Grid resources. With Pumpkin you can re-run the sensitivity analysis applications with default setting, but you can also replace the default sensitive analysis by your own model. In the guide we will explain to you step by step: In this guide we will explain how you can apply the sensitivity analysis on your own model of interest. More specifically:

- How you can annotate, and deploy data to make it ready to be used by the Pumpkin.
- How can you replace the default sensitive analysis by your own model

Figure 1 depicts a high-level view of part of the Sensitivity analysis workflow which can be decomposed into three main tasks:

1. **HyperCubeSampler**: Generation of a set of different input samples by using Latin Hypercube Sampling. Each sample consists of a different numerical value for each model parameter, representing a variation that will be applied on the model parameter in step 2.
2. **Model**: Here the model of interest is run with the default values of the model parameters changed by using the variation given in the input sample. This step will be repeated for each input sample resulting in multiple realisations of the output of interest.
3. **SobolDecompositer**: In this step the multiple realisations of the output of interest are used together with the input samples to calculate the Sobol sensitivity indices. This step is thus the actual sensitivity analysis.

In Pumpkin you should not consider the sensitivity analysis as workflow neither as a set of interlinked processes but as a framework that regulates how data will be processed by one or more tasks composing the sensitivity analysis and how this data is transformed from an initial state to a new state. In this data centric view, data have to be annotated and contain all the information needed to bring it to its final state. Pumpkin introduced two concepts to enable you to prepare your data to be processed by various processes.

- **Data packets** which contain among other things the meta-data needed to process a data set
- **Seeds** which is a wrapper that helps to run a process on a given data set

Once you are able to think of a traditional workflow application in term of data and process to be performed on it, and you have learnt how to define/create data packets and seeds in Pumpkin, you
will be able to process various data sets with various processes beyond the sensitivity analysis tools. Here we will focus on the sensitivity analysis tools and precisely on the sub-workflow depicted in Figure 1.

![Image](image.png)

**Figure 1: high-level view of the workflow which can be decomposed into three main phases: HyperCubeSampler, Model, and SobolDecompositor**

Workflow tasks with respect to HyperCubeSampler and SobolDecompositor are executed using the Pumpkin framework. A dedicated VPH virtual image “Atomic Service” (in VPH-share terminology), hosts the Pumpkin framework and the tools necessary to execute the workflow tasks. Once the Atomic Service is running, the Pumpkin starts processing tasks (tasks are data driven and thus initiated when a new data file is dropped into the data directory of the Pumpkin framework).

Note for VPH-share users: Within VPH Virtual Machine management is handled by Atmosphere, which is responsible for starting and stopping VMs.

2. **Atomic service “SA_tools”**

The Atomic service SA_tools consist of two functions and a model. The model needs to be included by the user of the sensitivity tools.

- **The function GenInput (HyperCubeSampler)[matlab]:** Create the input samples for the model of interest. Each input sample consists of a numerical value for each model parameter that represents a variation (expressed in fractions) that will be applied to the model parameters in the model. Running the model for multiple input samples results in multiple realizations of the output of interest (see Deliverable: D5.2 Uncertainty Tools for more details). The function reads a file (e.g. 'input.txt') in which the user provides the names of the model parameters and their uncertainty domains, e.g.:

  Aortic length -10 10
  Aortic diameter -5 5

- After reading this file, the function creates a set of input samples consisting of a variation for each model parameter by using Latin Hypercube sample and of the samples in a .csv file, where each line represents an input sample and each column represents a parameter.

- **Model:** A deterministic model is created by the user himself, e.g. a pulse wave propagation model, 3d-model or an analytical function. To be able to integrate your own model within the framework you need to change your code at the position where you assign numerical values...
to your model parameters. Here you need to apply the variation given by the input sample \( k \) to each model parameter. In case of a Matlab model you can easily do this as follows:

\[
X1 = X1*(1+\text{beta}(r,1));
X2 = X2*(1+\text{beta}(r,2));
\text{etc.},
\]

In which \( \text{beta} \) is an array with all numerical values of the previously created csv-file, \( r \) the \( r^{th} \) run (input sample) and the column index refers to the column in \( \text{beta} \) corresponding to the specific model parameter. Moreover, the user needs to select the output of interest (f.e. mean pressure or flow) and store this automatically to an Xsim-output.csv file in which each row corresponds to the \( r^{th} \) run and each column to the output of interest defined. For example, in case of two output of interests \( y1 \) and \( y2 \) column 1 corresponds to \( y1 \) and column 2 to \( y2 \).

- **Sobol** *(SobolDecompositor)*: This function calculates the Sobol-indices by using sobol2002 function in R statistical (see Deliverable: D5.2 Uncertainty Tools for more details) and to additional files that were automatically generated by the function GenInput. The decomposer is a process that reads the model outputs file (e.g. Xsim-output.csv) and transforms it into a set of sensitivity indices. The sensitivity indices, saved in a new csv-file are what the users are interested in and can be considered the ultimate output of the SA tool.

### 3. Pumpkin framework

The Pumpkin framework is deployed (hosted) in the Atomic service **SA_tools**. Once **SA_tools** is running you can ssh to the Virtual Machine (VM) to customize Pumpkin to Sensitivity analysis with your own model. You need to provide:

- Data sets to be used and input for the Sensitivity analysis workflow.
  - The *input.txt* is the ultimate input of the SA tool, and its contents depend on the model the SA tool is applied to.
- Edit the Pumpkin configuration file
- Specify the name of the data file
- The number of iteration
- Model: The user should be able to provide the model. However, they must provide it in a way that it will take a set of arguments (i.e. a specific set of values for \( x1, x2 \) and \( x3 \)) and then return the set of numbers specific to that set of arguments (i.e. a specific set of values for \( y1 \) and \( y2 \)). The model will need to be run for all the sets of input sets that were generated using the sampler (i.e. for each line in the .csv file), yielding a file with the outputs (i.e. for each line in the input .csv a corresponding line in the output .csv file).

To let Pumpkin perform the SA workflow, you need to move the data files *input.txt* to /data/a/ directory in the pumpkin framework installation. When the Atomic service **SA_tools** is running Pumpkin installation is accessible through webdav. A test Atomics service is currently running on the VPH-share infrastructure and pumpkin is accessible through webdav:

```bash
webdav://149.156.10.132:58178/pmkda
```  
(Figure 2).
When the Atomic service SA_tools is running pumpkin installation is accessible through webdav, the screenshot shows the typical directory setup of Pumpkin.

### 3.1. Input data:

The data to be processed by Pumpkin has to be saved in the /data directory of the pumpkin framework.

For the sensitivity analysis workflow 'input.txt' is **the ultimate input of the SA tool, and its contents depend on the model the SA tool is applied to.** The 'input.txt' is the input of the Sampler GenInput (HyperCubeSampler) and provides the uncertainty ranges of each model. Note that this file will depend on the model the user wants to apply SA tools to because different models can have different numbers of input parameters and also different uncertainty ranges.

### 3.2. Seeds

Pumpkin framework refers to application processes (workflow tasks) as seeds. These application processes can be executed on a set of distributed computing resources (similar to Seed dispersal in nature). To allow Pumpkin to distributed the seeds on various computers, a seed has to be defined and stored in the /seeds/ directory. A seed is a wrapper of the application process (workflow task), it is written in Python and has a specific structure. A stub-model seed submodel.py is provided in the directory /seeds/ of Pumpkin framework. In the same directory you will find two seeds, which have developed for the two Sensitivity analyses workflow tasks the sobolDecompsitor (sobol.py) and hypercubeSampler (genInput.py).

You can edit and adapt the submodel.py to your own model. It is assumed that your own application is deployed in the Pumpkin framework. In the SA_tool Atomic service we have deployed all the code and libraries needed to run the sobolDecompsitor and the hypercubeSampler of the sensitivity application in the directory /sensitivity/.

The sub-model seed has the following structure
- A configuration section marked by `##`
- A class SubModel which three methods:
  - `__init__` method: initialize the environment for seeds to run on a given computer
  - `on_load` method: loads the seed code
  - `split` method: implement a data specific data split routine, it splits the data before the run function is executed (should be used only when data parallelism is possible)
**merge** method: implement a data specific merging routine. It can be use to reassemble the outcome processing concurrently data after a split or merge multiple input data to create a complete new set of data ready for further processing.

**run** method: runs the seed

### 3.3. Data Packet structure

Data Packet structure is the meta-data that helps Pumpkin to execute the seeds on multiple computing resources. For the StubModel.py we provide the example of Data Packet structure in the /example_packets/ directory, the file DataPacket-GenInput.pkt contains the data packet structure that goes with stubModel, Table 1 gives a description of the various fields and the default values for the StubModel.

![Diagram of Data Packet structure](image)

**Figure 3:** The structure of a data packet (left) and the content of the StubModel File provided in the /example_packets/ directory (right). The latter shows the syntax in which data packet are defined. Each section in the data packet structure is defined within {} and contains a comma separated list of key-value pairs. In the StubModel example the automaton and the data route sections are empty.

In the future release the entire data packet will be automatically generated, for now the users still has to provide manually a the value of a couple of fields. Table 1 gives a description of the various fields and the default values for the StubModel.
Table 1: description of the fields that should be modified by the users.

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
<th>Default value in StubModel</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Container Section</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Last func</td>
<td>- Name of the function after which the data processing stop automatically</td>
<td>“none”</td>
</tr>
<tr>
<td>• Stop func</td>
<td></td>
<td>“StubModel”</td>
</tr>
<tr>
<td><strong>Automaton Section</strong></td>
<td>Should be left empty</td>
<td></td>
</tr>
<tr>
<td><strong>Data Provenance Section</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• data</td>
<td>Path to the input data set. Name of the function that can process the data.</td>
<td>“/data/c466f-98ea09ac.tar.gz”</td>
</tr>
<tr>
<td>• func</td>
<td></td>
<td>“StubModel”</td>
</tr>
<tr>
<td><strong>Route Section</strong></td>
<td>Should be left empty</td>
<td></td>
</tr>
</tbody>
</table>

3.4. Configure and initialize the data processing

Pumpkin configuration file DataPacket-GenInput.pkt is located in the /examples_packets/. Open this file in your favourite text editor:

- Search for the line: "data": "/data/testje3.txt,10",
- Edit this line such that it point to the your own input data file (new input of the GenInputSampler process)
- You can specify the number of iteration by changing the default value 10
- Save the DataPacket-GenInput.pkt and move it to rx/ directory
- The last step will **automatically initiate the sensitivity analysis workflow** (Figure 1).
- The results will be stored in /out_data/
4. Frequently Asked Questions

4.1 How can I run the default sensitivity analysis with the default settings?

1. Move the file DataPacket-GenInput.pkt from the /examples_packets/ directory to the rx/
2. The last step will automatically initiate the sensitivity analysis workflow on the data file testje3.txt sets stored in the /data/ directory
3. The results will be stored in /out_data/

4.2. How can I change the model of the sensitivity analysis with my own model?

1. Create the appropriate input file for your model ('input.txt' contains: the uncertainty ranges of each input parameter to the model)
2. Write a Matlab function that describe your model
3. Modify the stubModel to call your model
4. Modify the DataPacket-GenInput.pkt is located in the /examples_packets/ to point to your own input.txt

More details are can be found in Section 2 of the manual.

4.3. How can I define a new workflow other then the sensitivity analysis workflow?

- In Pumpkin the traditional workflow concept doesn’t exist, this question can be reformulated as “how can transform my data from state A to state B?” The Answer to this question is
  - You need all to deploy the processes that can be applied to your data
  - You to define the DataPacket
  - Write a seed for every process to be applied on your data

4.4. How can I create new seeds in Pumpkin?

1. Deploy all the code and the libraries in the needed to run your new seed in the directory setup of Pumpkin (figure 2). Create a separate directory with the name of your application
2. Create a seed by adapting the StubModel.py to your specific case (see Section 3.2).
3. Save the new seed in /seeds/ directory
4. Create a Data Packet structure by adapting the DataPacket-StubModel.pkt to your specific case (see Section 3.3).
5. Save the new DataPacket file to /rx/ directory to initiate the sensitivity analysis workflow
6. The results will be stored in /out_data/