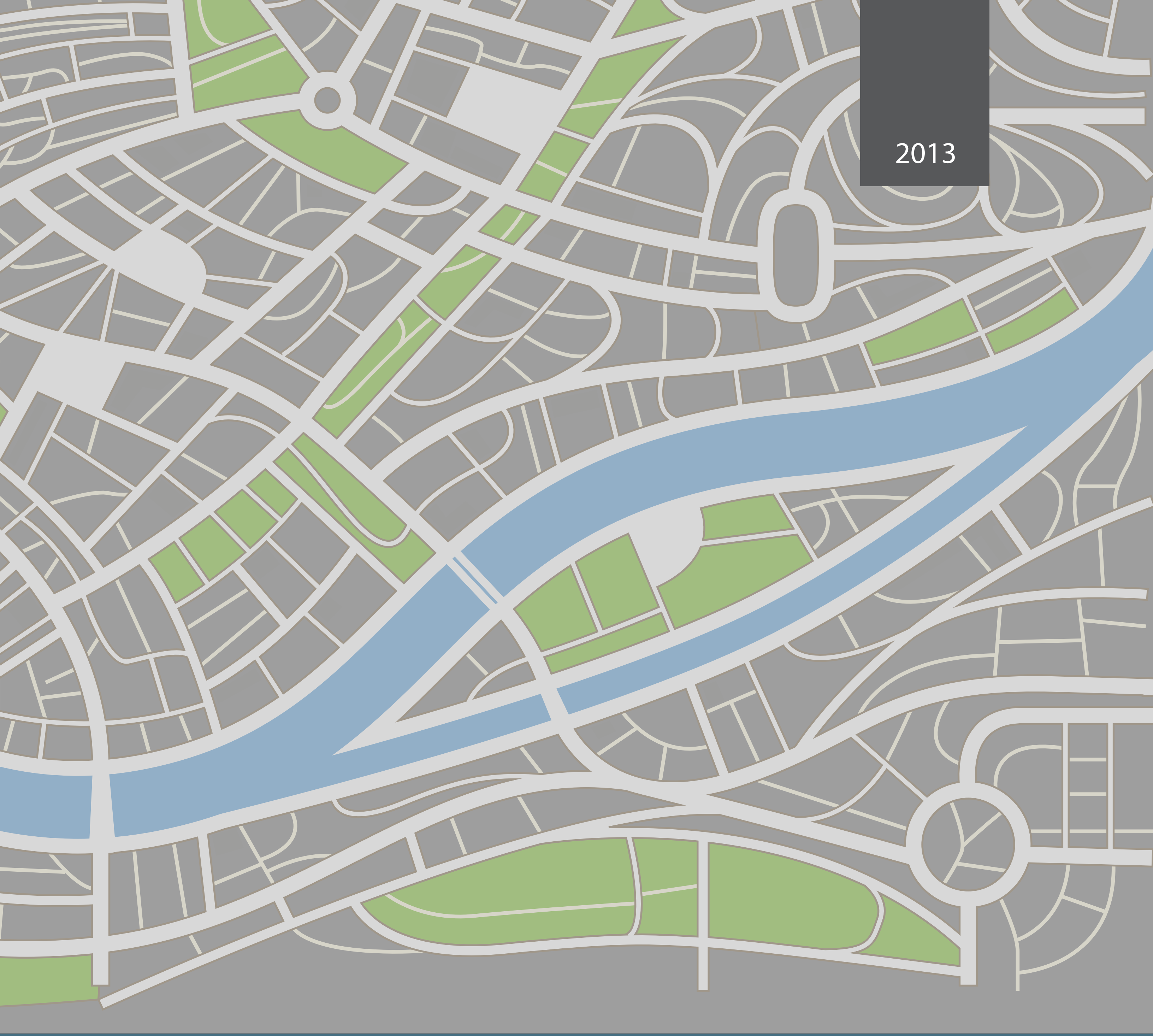


2013

InaSAFE: Indonesia Scenario Assesment for Emergencies

Training Manual for Software Developers

OpenDRI OPEN DATA FOR RESILIENCE INITIATIVE

GFDRR Global Facility for Disaster Reduction and Recovery

THE WORLD BANK Working for a World Free of Poverty

Published in Washington DC - February 2013
World Bank
www.worldbank.org/lcrdrm
Contact: Bradley Lyon (blyon@worldbank.org)
Bishwa Pandey (bpandey@worldbank.org)

University of West Indies - St. Augustine
www.sta.uwi.edu/eng/surveying
Contact: Earl Edwards (Earl.Edwards@sta.uwi.edu)

AUSTRALIA-INDONESIA
FACILITY FOR
DISASTER REDUCTION

InaSAFE Documentation

Release 1.0.1-final

Ole Moeller Nielsen, Tim Sutton

January 09, 2013

CONTENTS

1 InaSAFE Tutorial **1**
- 1.1 Overview . . . 1
- 1.2 Installation . . . 2
- 1.3 Using InaSAFE . . . 6
- 1.4 International Language . . . 30
- 1.5 Thank You . . . 31

2 InaSAFE user documentation **33**
- 2.1 Installing InaSAFE . . . 33
- 2.2 InaSAFE Screenshots . . . 35
- 2.3 Using the InaSAFE Plugin . . . 35
- 2.4 Keywords System . . . 45
- 2.5 InaSAFE Options . . . 51
- 2.6 Impact Functions . . . 53
- 2.7 Impact Functions Documentation . . . 54
- 2.8 Post-processors . . . 58
- 2.9 Demo projects . . . 60
- 2.10 Data Types . . . 60
- 2.11 Using QGIS . . . 60
- 2.12 Getting involved . . . 61
- 2.13 Frequently Asked Questions . . . 62
- 2.14 Known Issues . . . 64
- 2.15 Whats new? . . . 64

3 InaSAFE developer documentation **69**
- 3.1 Development under Gnu/Linux . . . 69
- 3.2 Development under MS Windows . . . 73
- 3.3 Coding Standards . . . 86
- 3.4 Logging . . . 90
- 3.5 Getting and using the named Logger instance . . . 90
- 3.6 Logging exceptions . . . 90
- 3.7 Logging in loops . . . 91
- 3.8 Remote logging . . . 91
- 3.9 QGIS Log Messages . . . 91
- 3.10 Logging with third party applications . . . 92
- 3.11 Profiling . . . 92
- 3.12 Internationalisation Support . . . 92
- 3.13 Version Control . . . 97
- 3.14 Maintaining the documentation . . . 100

3.15 Postprocessors . . . 103
3.16 Reporting bugs and getting help . . . 105
3.17 Preparing a release . . . 105
3.18 Continuous Integration Testing with Jenkins . . . 112
3.19 IRC Notes for InaSAFE . . . 118
3.20 Frequently Asked Questions . . . 119
3.21 Todo List . . . 120
3.22 InaSAFE Realtime . . . 121

4 InaSAFE's API documentation 129
4.1 Packages safe_qgis . . . 129
4.2 Unit Tests . . . 131
4.3 Packages safe . . . 133

5 Authors, Contributors and Funders 149

6 Indices and tables 151

Python Module Index 153

CHAPTER

ONE

INASAFE TUTORIAL

1.1 Overview

1.1.1 Description

InaSAFE is a plugin for QGIS software. It aims to produce realistic natural hazard impact scenarios for better planning, preparedness and response activities, using hazard and exposure geographic data. A user's manual developed by Ole Moeller Nielsen and Tim Sutton complements this short training material. During this training, you will explore the different components of InaSAFE plugin and their usage for an easy-to-use risk and impact scenarios assessments. A data sample will be provided with the present material to give you the ability to learn how to manipulate data in InaSAFE, what the requirements of the data are and what kind of results you can expect out of the data.

1.1.2 Prerequisites

This training is designed for those with a basic or intermediate knowledge of GIS, who wants to integrate scientific-based risk impact scenario in their decision-making. Since the tool is a QGIS plugin, some knowledge of the QGIS environment will help, but not required.

1.1.3 Goals

After the training, you will be able to:

- install a plugin in QGIS and understand its usage for spatial analysis
- understand what are hazard and exposure data and how they can be used to estimate impacts
- learn how to prepare the data to be able to use it in InaSAFE
- perform a risk scenario using flood and earthquake data
- analyze estimated impact of the example scenarios
- learn how to print and save the result of the simulation

1.1.4 Software Requirements

To complete the exercise, you need a standard computer with at least 4GB of RAM running Windows, Linux or Mac OS X. You also need to install the following on your computer:

- QGIS

- InaSAFE plugin.

1.1.5 Reference

"InaSAFE Documentation, Release 0.4.0-alpha", Ole Moeller Nielsen & Tim Sutton, June 2012. For more information, please visit www.inasafe.org.

1.2 Installation

1.2.1 Install QGIS

You need QGIS version 1.7 or newer.

You can download the software from http://download.qgis.org.

1.2.2 Install InaSAFE Plugin

Option 1: From QGIS Repository:

To install the InaSAFE, use the plugin manager in QGIS:

Plugins → *Fetch Python Plugins* Then search for 'InaSAFE', select it and click the install button. The plugin will now be added to your plugins menu.

Option 2: Manual Installation:

To install the InaSAFE plugin, go to https://github.com/AIFDR/inasafe/downloads.

The application package comes as a zip file. Please select the most updated version.

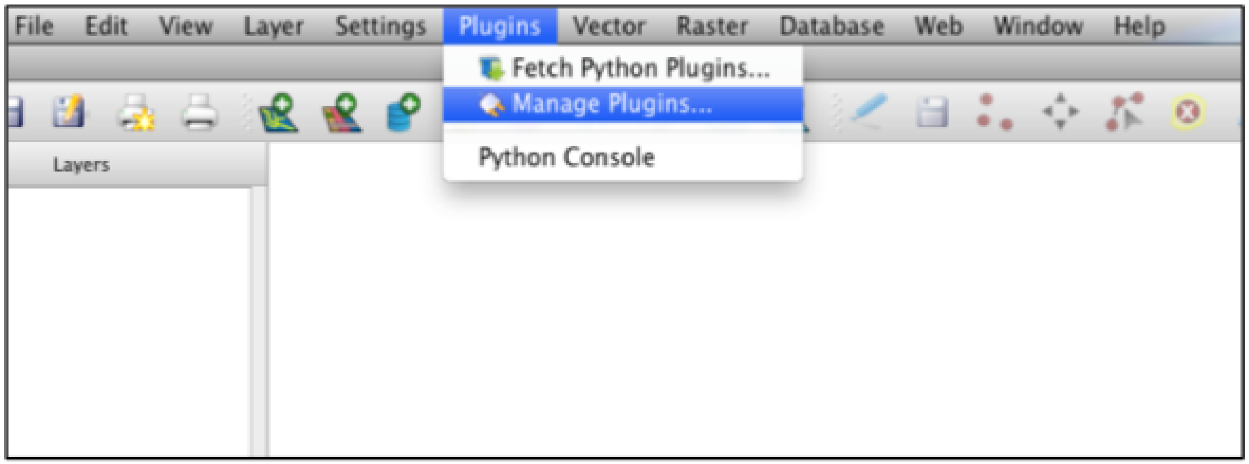
Extract the zip file into the QGIS directory `C:Users<your username>.qgispythonplugins`.

After extracting the plugin, it should be available as `C:Users<yourusername>.qgispythonpluginsinasafe`

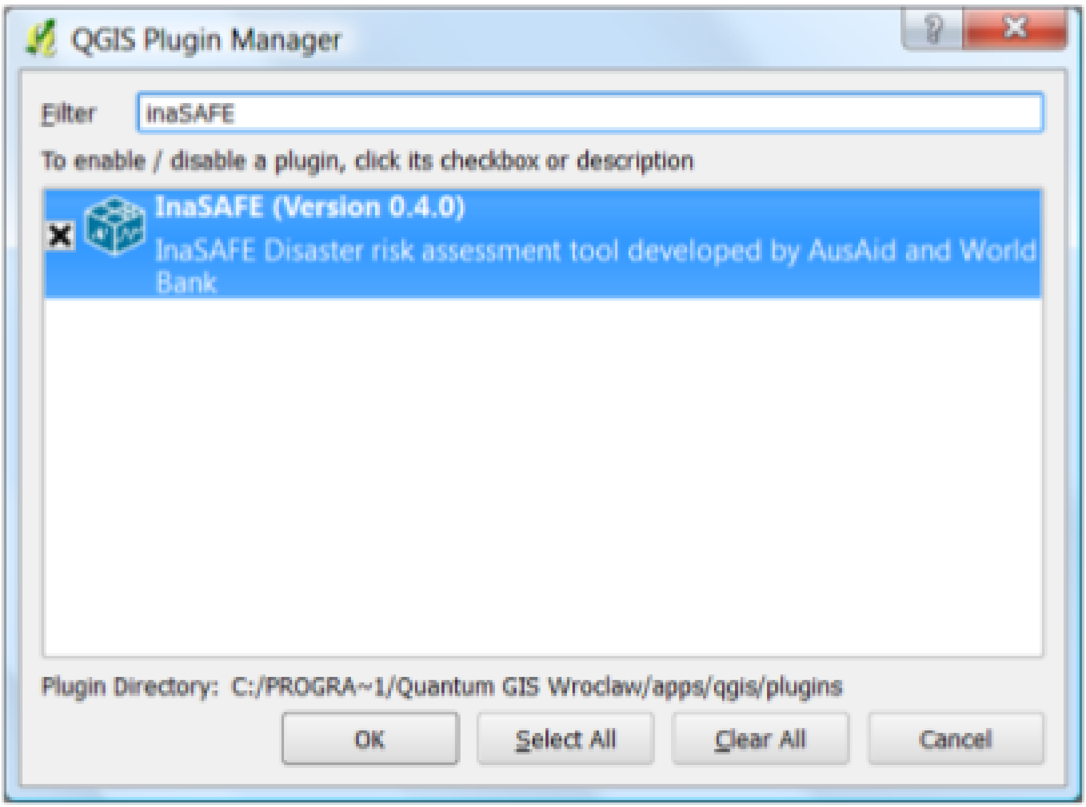
Mac and Linux users need to follow the same procedure but instead the plugin directory will be under your `$HOME` directory.

1.2.3 Enable InaSAFE Plugin in QGIS

Once the plugin is extracted in QGIS plugin directory, start QGIS and enable it from the plugin manager. To do this, open **Manage Plugins** from the **Plugins** menu on the menu toolbar.



A pop-up window that lists all available plugins in your current QGIS project will appear. Type **InaSAFE** in the filter box. You should see the InaSAFE plugin appear in the list. Now **tick the checkbox** next to it to enable the plugin.



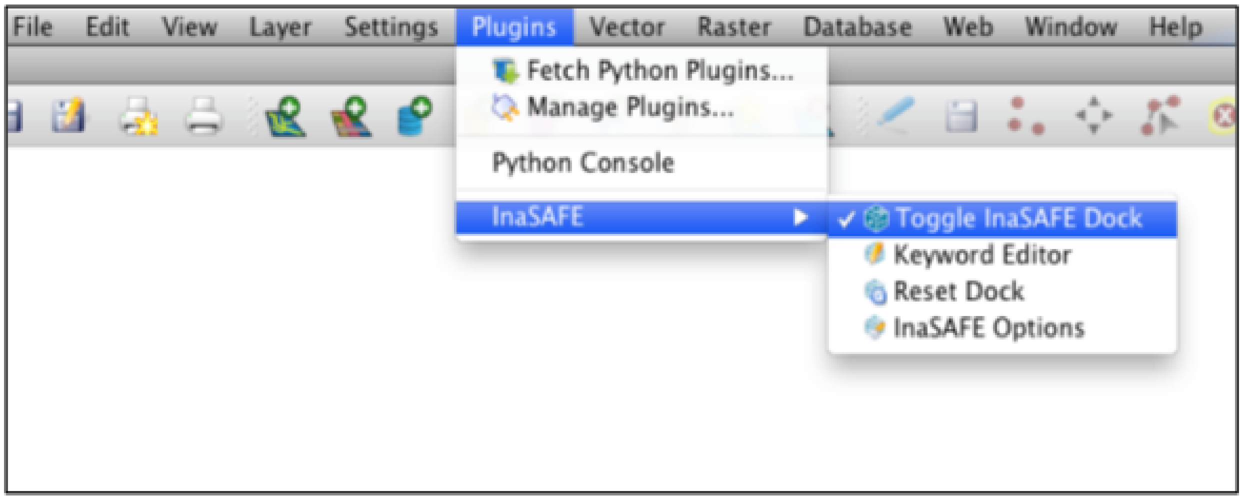
To enable / disable a plugin, click its checkbox or description

InaSAFE (Version 0.4.0)
InaSAFE Disaster risk assessment tool developed by AusAid and World Bank

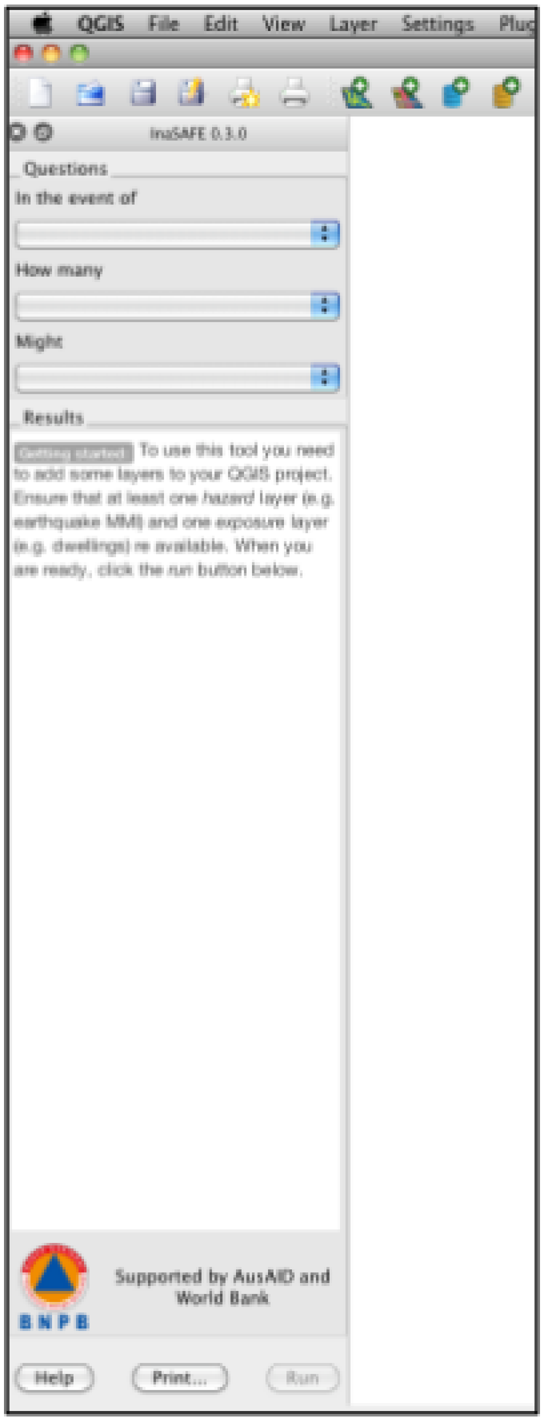
Plugin Directory: C:/PROGRA~1/Quantum GIS Wroclaw/apps/qgis/plugins

The plugin now will be added to your **Plugins** menu.

Now you will need to add the INASAFE panel on your QGIS interface. For that, select **Toggle Inasafe Dock** in the INASAFE plugin scroll list.



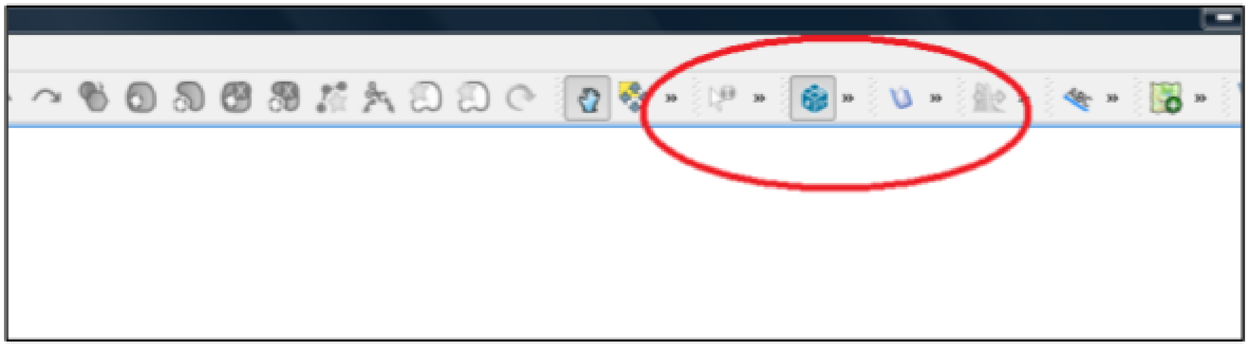
The InaSAFE dock panel will then appear on the left of your QGIS window.



Getting started To use this tool you need to add some layers to your QGIS project. Ensure that at least one *hazard* layer (e.g. earthquake MMI) and one exposure layer (e.g. dwellings) re available. When you are ready, click the *run* button below.

It is the main way to interact with the tools that are provided in InaSAFE.

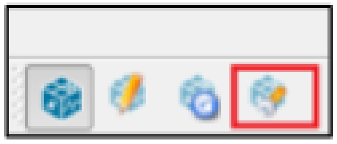
Also, an InaSAFE icon will appear on the QGIS toolbar.



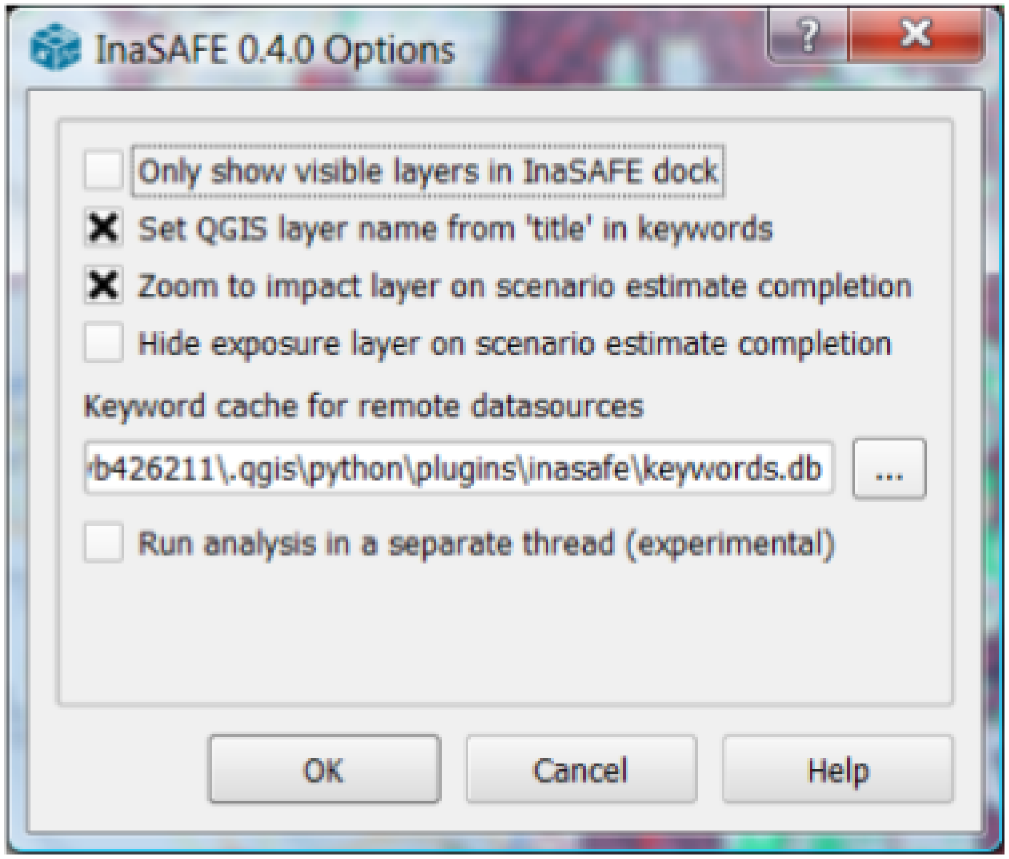
1.3 Using InaSAFE

1.3.1 InaSAFE Options

The InaSAFE plugin provides an options dialog which allows you to define various options relating to how InaSAFE will behave. The options dialog can be launched by clicking on the InaSAFE plugin toolbar's options icon (as shown below) or from QGIS *Plugins* → *InaSAFE* → *InaSAFE Options*.



Then the dialog will appear, looking something like this:


InaSAFE 0.4.0 Options

- [ ] Only show visible layers in InaSAFE dock
- [✓] Set QGIS layer name from 'title' in keywords
- [✓] Zoom to impact layer on scenario estimate completion
- [ ] Hide exposure layer on scenario estimate completion

Keyword cache for remote datasources

b426211\.qgis\python\plugins\inasafe\keywords.db

- [ ] Run analysis in a separate thread (experimental)

Note: You can click on the Help button at any time and it will open the help documentation browser to this page.

The following options are available on the Options Dialog:

- **Only show visible layers in the InaSAFE dock:** This option will determine whether (when unchecked) all hazard, exposure and impact layers should be listed in the InaSAFE dock's combo boxes; or (when checked) only visible layers.
- **Set QGIS layer name from 'title' in keywords:** This option will (when enabled) cause QGIS to name layers in the Layers tree, using the title keyword in the layer's keywords file. If the layer has no 'title' in its keywords, or it has no keywords at all, the normal QGIS behavior for naming layers will apply.
- **Zoom to impact layer on scenario estimate completion:** This option will cause the map view to zoom in/out in order to completely contain the InaSAFE impact scenario map output when an analysis is completed.
- **Hide exposure layer on scenario estimate completion:** This option will cause QGIS to turn off the exposure layer used when InaSAFE completes the current analysis. You can re-enable the layer visibility again by checking its checkbox in the legend.
- **Keyword cache for remote datasources:** This option is used to determine where keywords are stored for datasets where it is not possible to write them into a .keywords file. See Keywords System for more information on the keywords system.
- **Run analysis in separate thread (experimental):** This option cause the analysis to be run in its own thread.

WARNING!

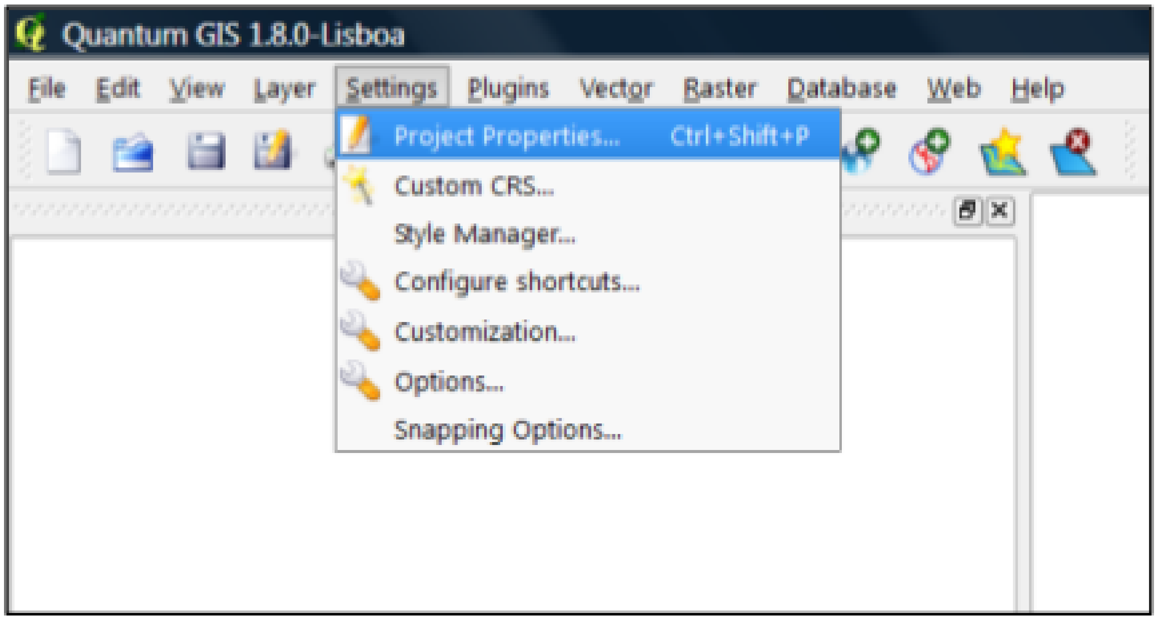
- It is not recommended to use the threaded implementation at this time. For this reason it is disabled by default.

- Pressing Cancel at any time will close the options dialog and any changes made will not be applied.
- Pressing Ok at any time will close the options dialog and any changes made will be applied immediately.
- The exact button order shown on this dialog may differ depending on your operating system or desktop environment.

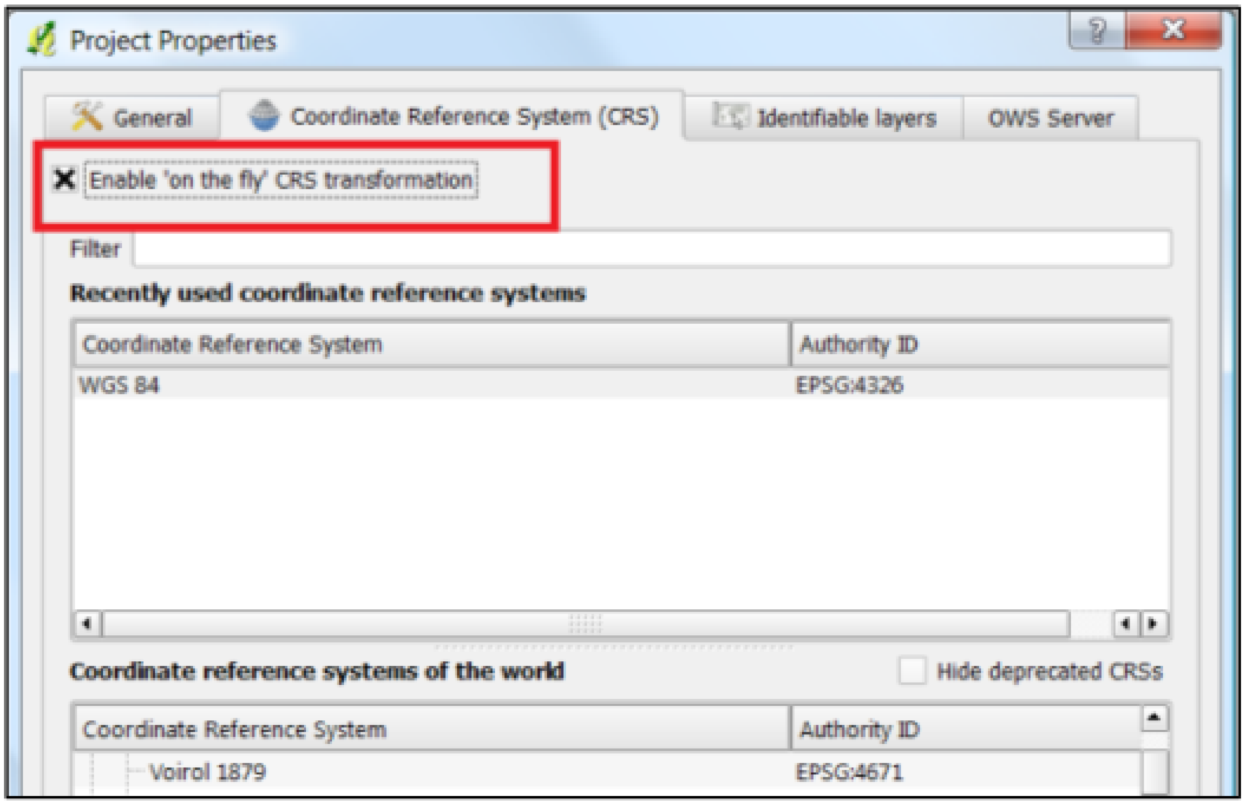
1.3.2 Adjust Projection

Before continuing we need to turn one more QGIS functionality on, to enable all data layers display in one projection (no matter what their projection).

For that, go to QGIS **Settings/Project Properties.**



Click on **Coordinate Reference System (CRS)** tab in the new dialog box. Tick the **Enable 'on the fly' CRS transformation** box. And then **OK.**



☒ Enable 'on the fly' CRS transformation

Recently used coordinate reference systems

Coordinate Reference System	Authority ID
WGS 84	EPSG:4326

Coordinate reference systems of the world

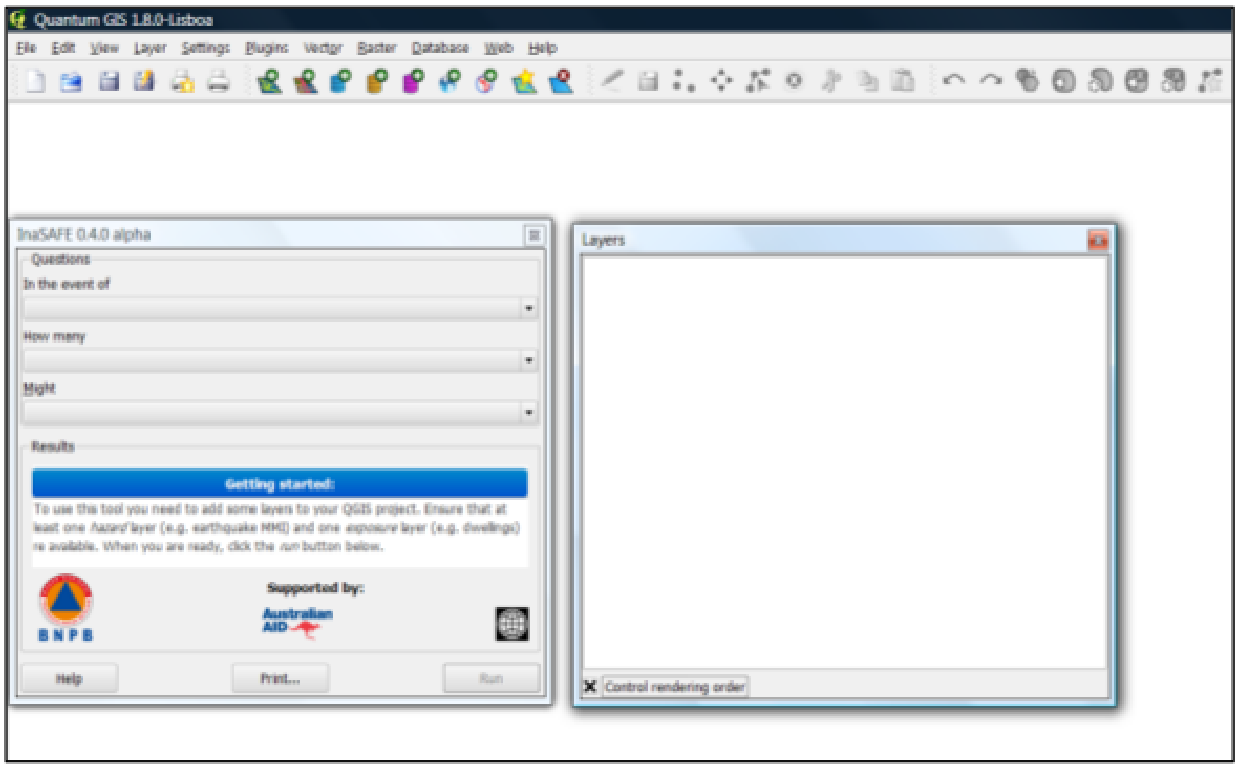
Coordinate Reference System	Authority ID
Voirol 1879	EPSG:4671

Now, any data layer that we will integrate into our project will be adjusted on the same coordinate.

1.3.3 Exploring InaSAFE Plugin

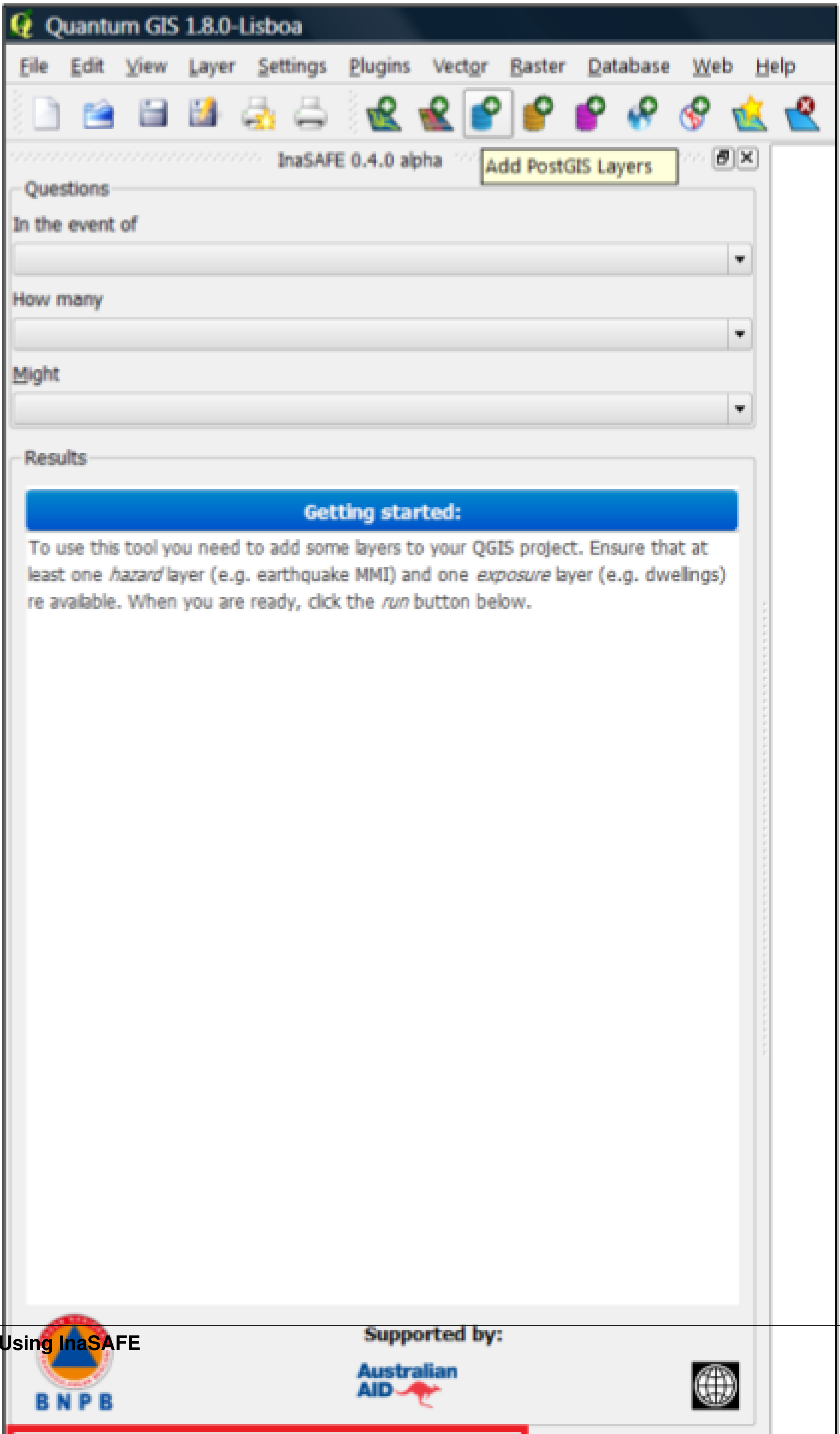
You can drag and drop the dock panel to reposition it in the user interface. For example, dragging the panel towards the right margin of the QGIS application will dock it to the right side of the screen.

Depending on your preference you could show the **Layer** and **InaSAFE** panel at the same time.



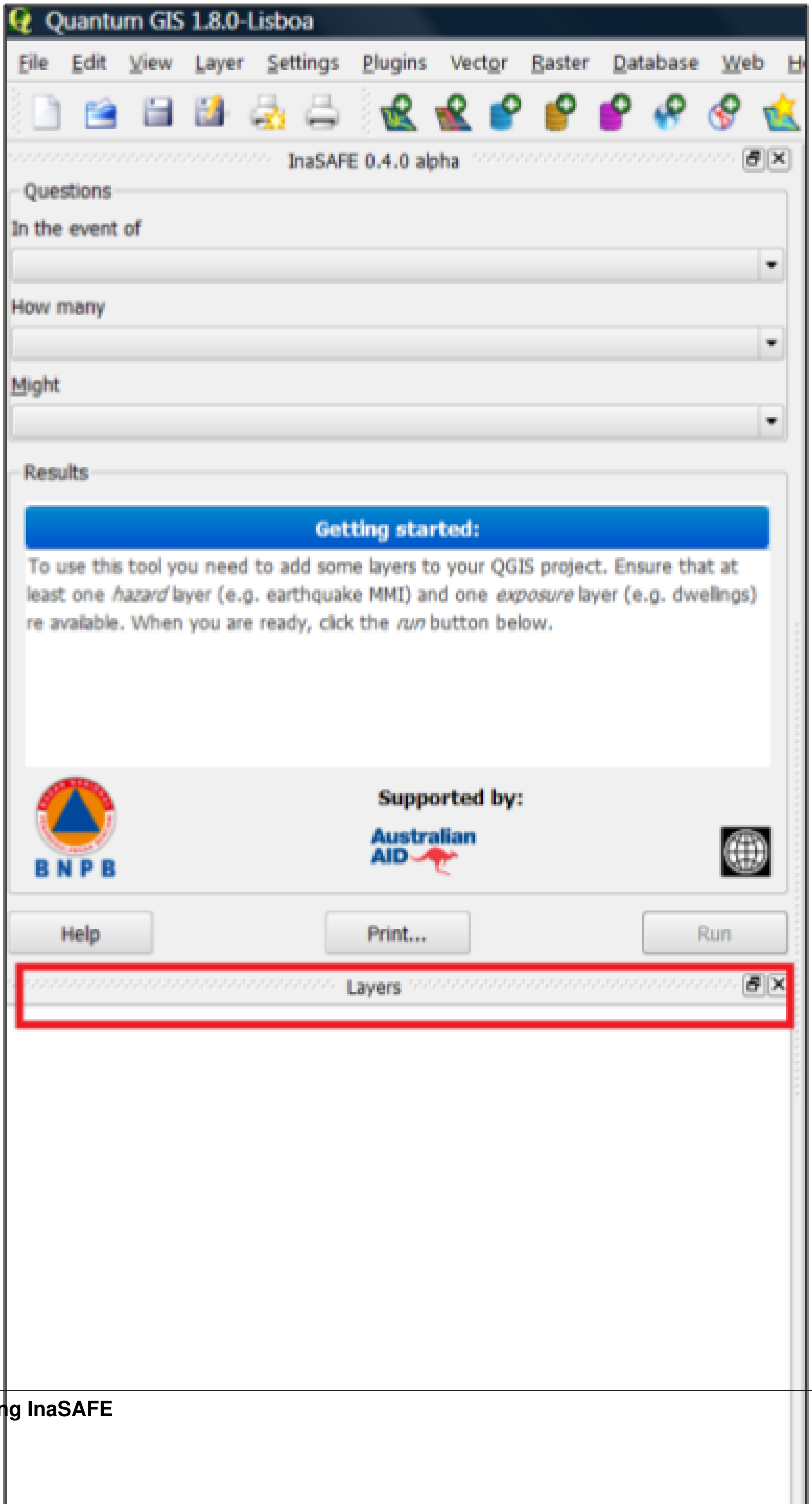
To use this tool you need to add some layers to your QGIS project. Ensure that at least one *hazard* layer (e.g. earthquake MMI) and one *exposure* layer (e.g. dwellings) re available. When you are ready, click the *run* button below.

Or have the **Layer** and **InaSAFE** panels in a tab systems.



To use this tool you need to add some layers to your QGIS project. Ensure that at least one *hazard* layer (e.g. earthquake MMI) and one *exposure* layer (e.g. dwellings) re available. When you are ready, click the *run* button below.

Or for more convenience, having them on top of each other.



To use this tool you need to add some layers to your QGIS project. Ensure that at least one *hazard* layer (e.g. earthquake MMI) and one *exposure* layer (e.g. dwellings) re available. When you are ready, click the *run* button below.

The INASAFE panel contains 3 sections: **Questions, Results** and **Buttons.** We will explore those sections one by one.

1.3.4 The Questions Section

The intention of InaSAFE is to make it really simple and easy to perform your impact analysis. The Questions area provides a simple way for you to formulate what it is you want to find out? All questions are formulated in the form:

In the event of **[hazard]** *how many* **[exposure]** *might* **[impact].**

For example: "In the event of a **flood** how many **buildings** might be **closed**?"

Let's practice this exercise.

In order to answer such question, InaSAFE developers have built a number of impact functions that cover risk scenarios such as flood, tsunami, volcanic ash fall, earthquake and so on. In our case, we will use the flood impact function.

To answer our question "In the event of a flood, how many buildings might be closed", we need to complete all the areas in the Questions section: hazard, exposure, impact.

1.3.5 Hazard

Hazard is the physical event that creates the risk.

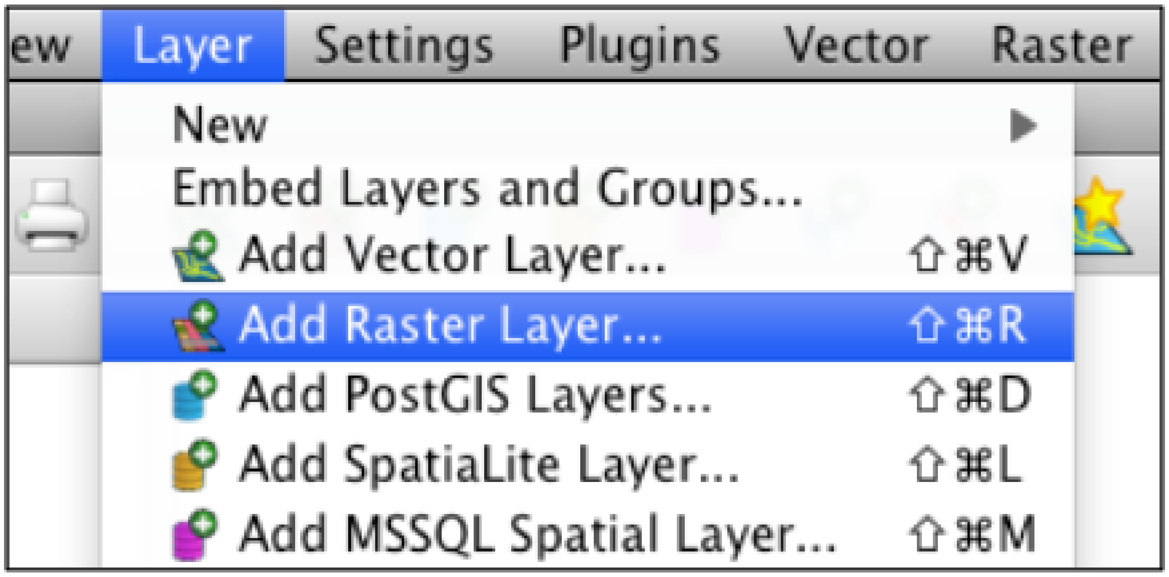
A hazard (in **the event of**) may be represented as a raster layer or as an area (polygon). For example:

- **Raster:** where each pixel in the raster represents the current flood depth following an inundation event.
- **Polygon:** where it has been identified that flood has existed in that area (this will not have depth related information)

For our exercise, we will use an example from Jakarta, Indonesia. Those data are already installed on your computer at `C:Users<your username>desktopinasafe_data`. If they are not, you can load the data on your desktop using the thumb drive distributed with this material.

The inasafe_data package contains various geographic data that we will use along the workshop.

Now, we will add the hazard layer in the INASAFE dock. For that, we need to add the hazard layer from QGIS first. The flood layer is in a raster format, so we will go to the QGIS menu, click on **Layer,** and select **Add Raster Layer.**


ew
Layer
Settings
Plugins
Vector
Raster
New
Embed Layers and Groups...
Add Vector Layer... ⇧⌘V
Add Raster Layer... ⇧⌘R
Add PostGIS Layers... ⇧⌘D
Add SpatiaLite Layer... ⇧⌘L
Add MSSQL Spatial Layer... ⇧⌘M

Once you click on that, a pop-up window will appear where you will have to fetch your flood data. Please select the **"Flood_Design_Depth_Jakarta_geographic.asc"** file from the hazard folder. This is a raster data (in ASCII format) that represents flooding depth in the Jakarta province. The display name is **Jakarta 2007 flood with dredging.**

You will notice that the layer filled automatically the "hazard" area in the InaSAFE dock panel. There are two important things to note when **uploading data** in InaSAFE.

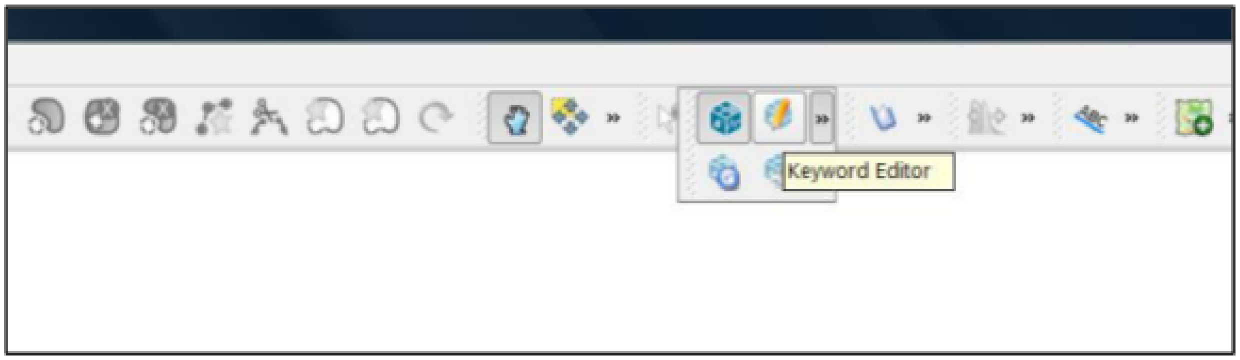
- Data should follow a keyword metadata system that allows InaSAFE to determine if the layer is a hazard or if it is an exposure.
- The area of analysis should overlap.

1.3.6 Adding keyword metadata

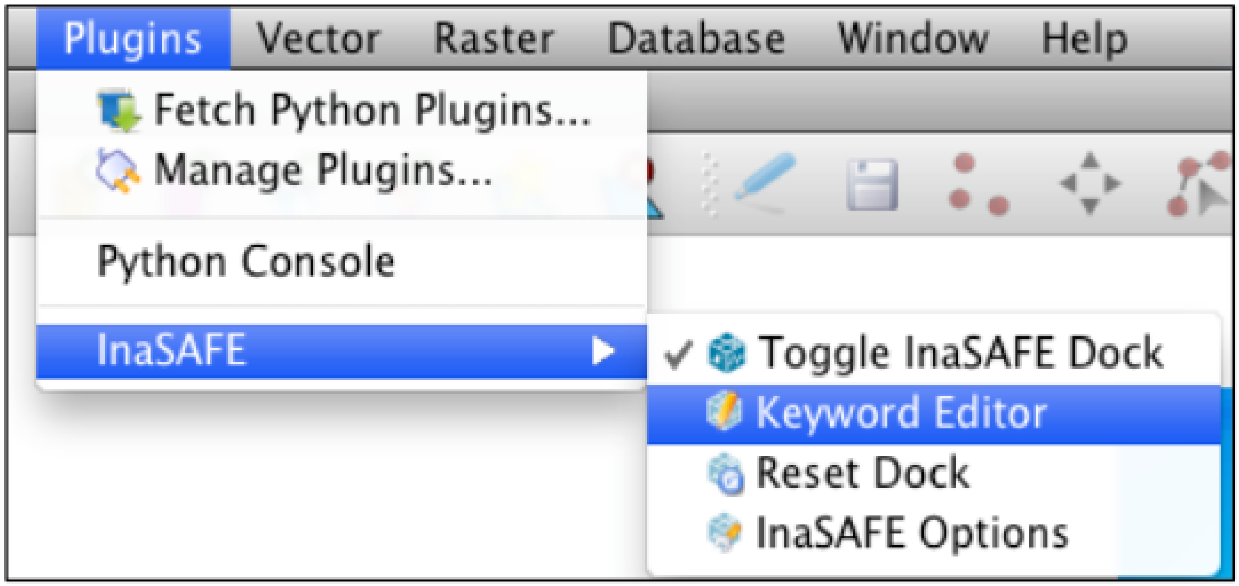
You may be wondering how the InaSAFE plugin determines whether a layer should be listed in the "In the event of" "How many" combo boxes? The plugin relies on simple keyword metadata to be associated with each layer (*The keyword system is described in detail in the user's manual under Keywords System*). Each layer that has a keyword allocating it's category to hazard will be listed in the "In the event of" combo. Similarly, a category of exposure in the keywords for a layer will result in it being listed under the "How many" combo. InaSAFE uses the combination of category, subcategory, units and datatype keywords to determine which impact functions will be listed in the "Might" combo.

In our exercise, the keywords were already created, so the data could fill automatically the "In the event of" "How many combo" boxes. If the keywords were not created in advance, then we will create them by following one of the two steps:

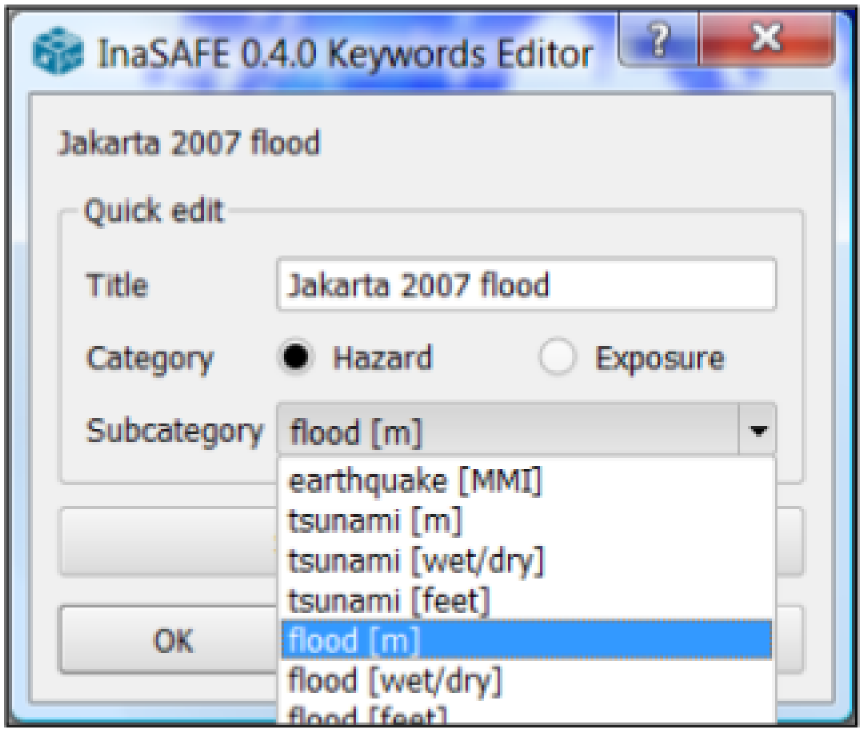
Go to the Inasafe tools on the toolbar, click on the **Keyword Editor** icon.



Or, open the **Plugin** menu on QGIS toolbar, click on **InaSAFE,** then click on the **Keyword Editor** in the scroll list.



Once you click on the Keyword Editor, a dialog box containing the flood data will be prompted. Since the flood data is a hazard layer, pinpoint the **Hazard** Category. In the Subcategory, we will choose flood [m] because our data represents depth of flood in Jakarta in meter unit.



Then click **OK.**

Now the data follow the keyword rule, and can be used in the InaSAFE function.

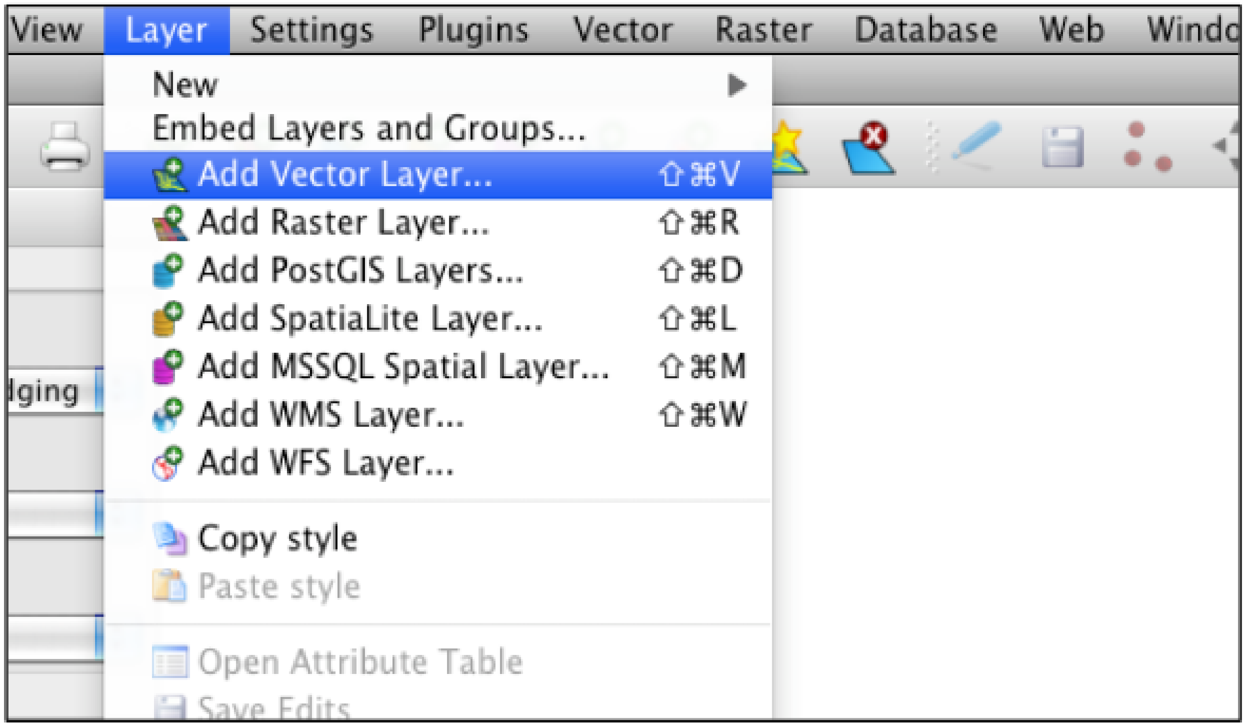
1.3.7 Exposure

Exposure is the sum of assets and population that are at risks.

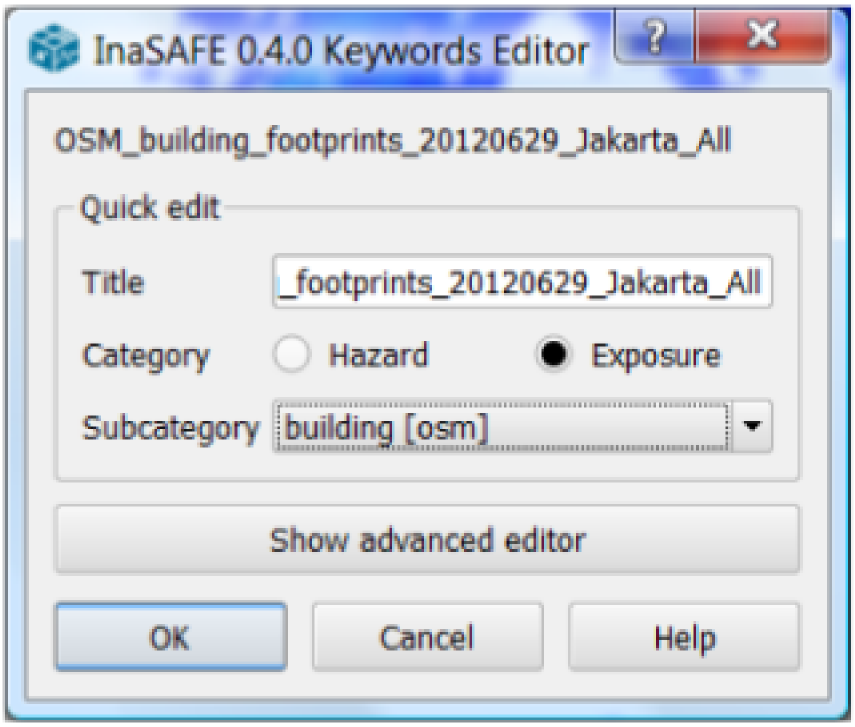
An exposure (How **many**) layer could be represented, for example, as vector polygon data representing building outlines, or a raster outline where each pixel represents the number of people resident in that cell.

Now, we will add the exposure layer in our InaSAFE project. For that, we need to add the exposure layer to QGIS first. For our exercise, we will use the OpenStreetMap (OSM) data that represents buildings in Jakarta Province.

The OSM building layer is in a vector format, so we will go to the QGIS menu toolbar, click on **Layer,** and select **Add Vector Layer.**



Once you click on that, a pop-up window will appear where you will have to fetch your OSM buildings data.



OSM_building_footprints_20120629_Jakarta_All

Quick edit

Title: _footprints_20120629_Jakarta_All

Category: Hazard [✓ Exposure]

Subcategory: building [osm]

Please select the "OSM_building_footprints_20120629_Jakarta_All.shp" file from the exposure folder.

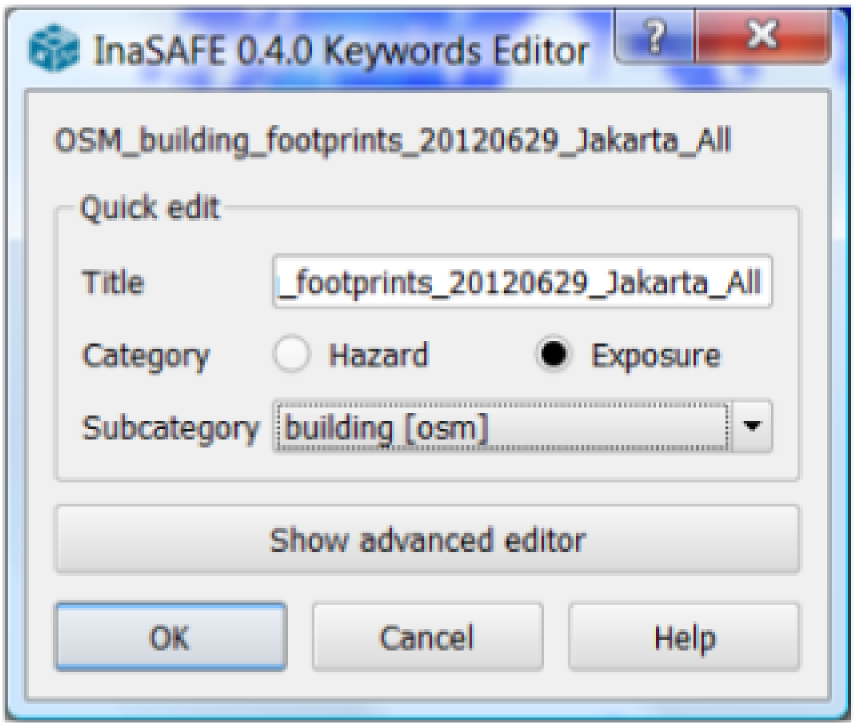
Click **Open.**

This is a vector data (in ESRI SHP format) that represents buildings data gathered by the Jakarta province community using the OSM participatory tools. The display name is "OSM **buildings** ".

Please note that the exposure data should follow the same **keyword system** that we explained earlier for the hazard data.

In our case, the keyword was already created. If the keyword was not created in advance, then we will create it by using the **Keyword Editor** in InaSAFE from the toolbar or from the **Plugins** menu.

Go to the **Plugin menu** on QGIS toolbar. Click on **InaSAFE.** Then, click on the **Keyword Editor** in the dialog box. Pinpoint the **Exposure** category. Choose **building [OSM]** in the **Subcategory** scroll box. Click **OK.**



OSM_building_footprints_20120629_Jakarta_All

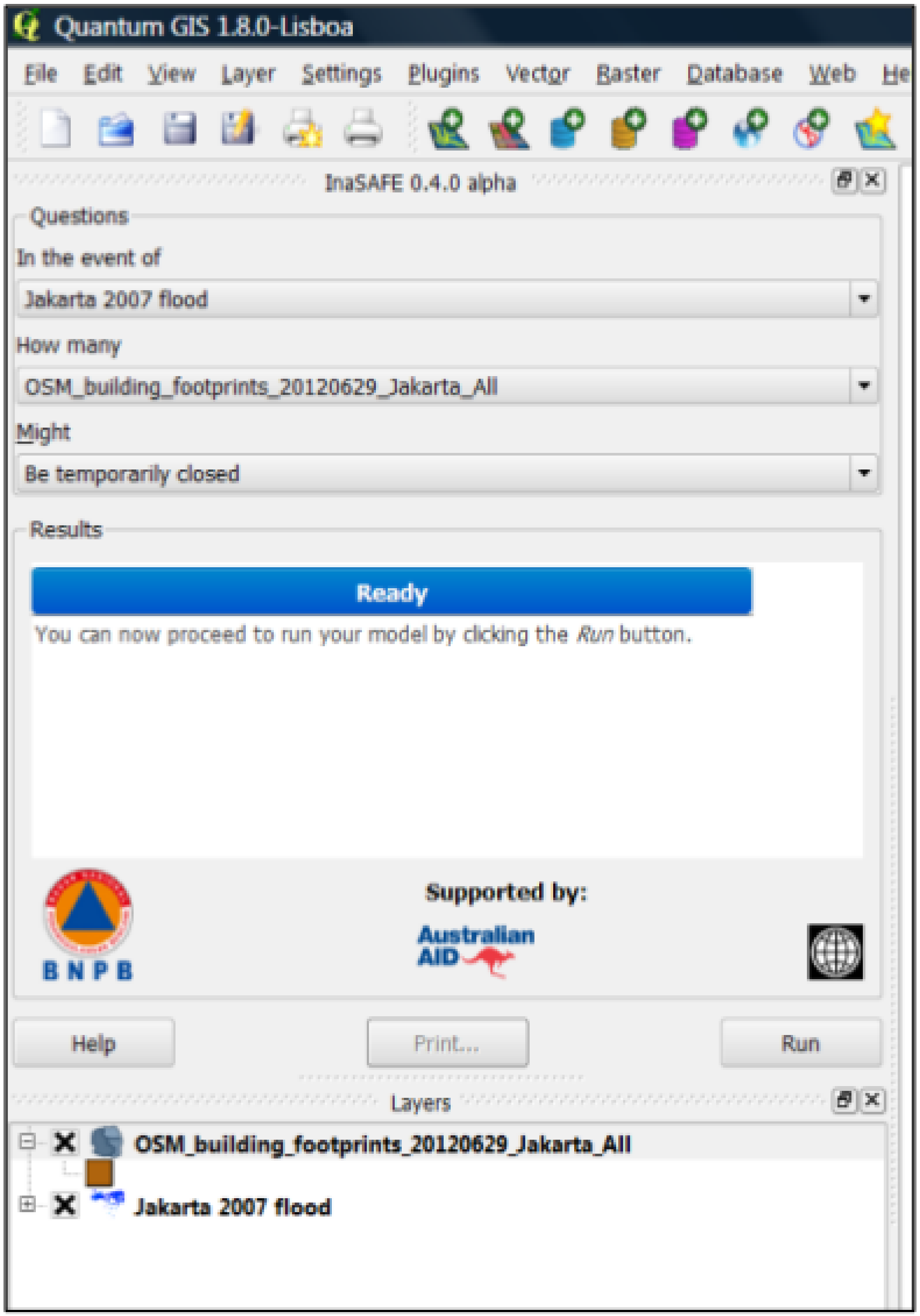
Quick edit

Title: _footprints_20120629_Jakarta_All

Category: Hazard [✓ Exposure]

Subcategory: building [osm]

Now our OSM building exposure data can be used in INASAFE and was automatically entered in the **How many box** of the INASAFE dock panel.



In the event of
Jakarta 2007 flood
How many
OSM_building_footprints_20120629_Jakarta_All
Might
Be temporarily closed

Ready

You can now proceed to run your model by clicking the *Run* button.

OSM_building_footprints_20120629_Jakarta_All
Jakarta 2007 flood

Also note that the 2 datasets are sitting on top of one another even though they are different projections.

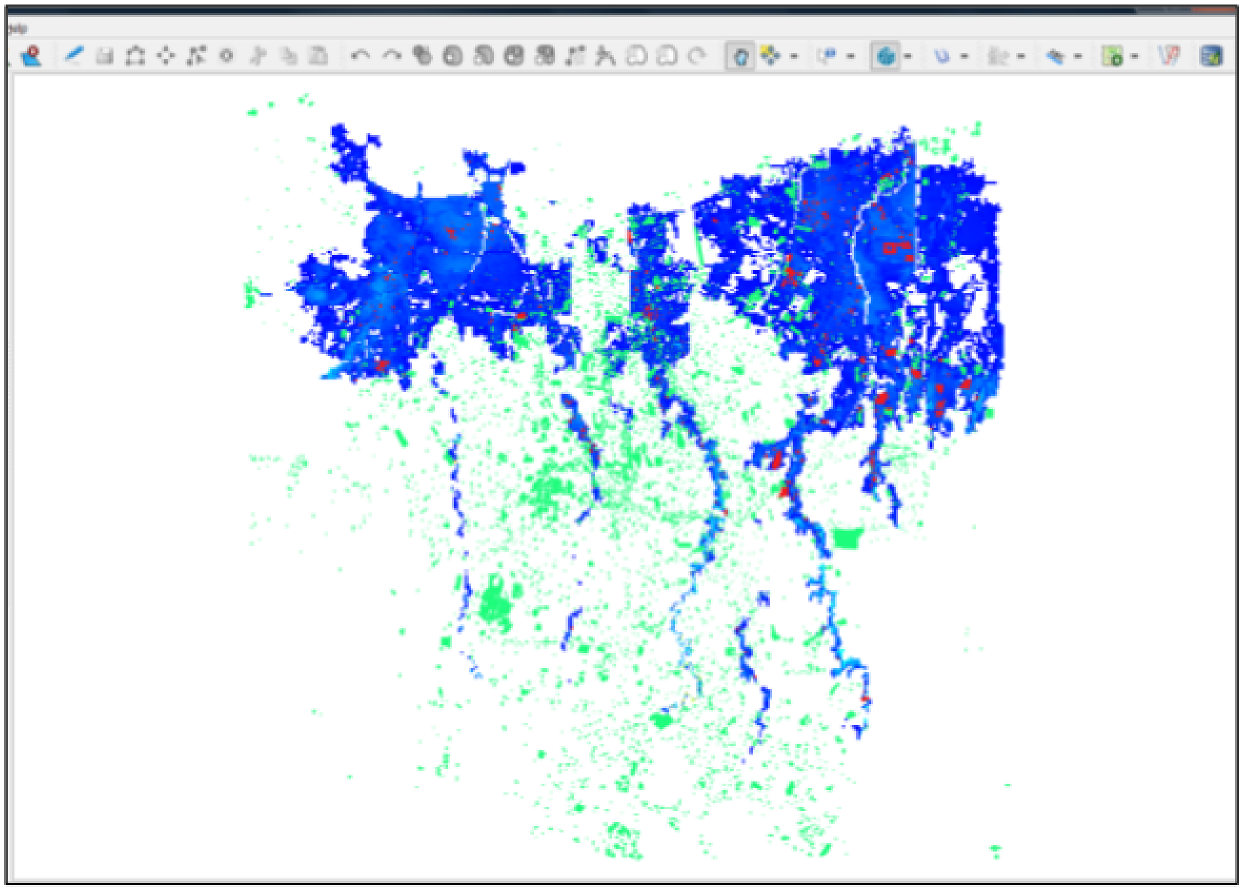
1.3.8 Impact Analysis

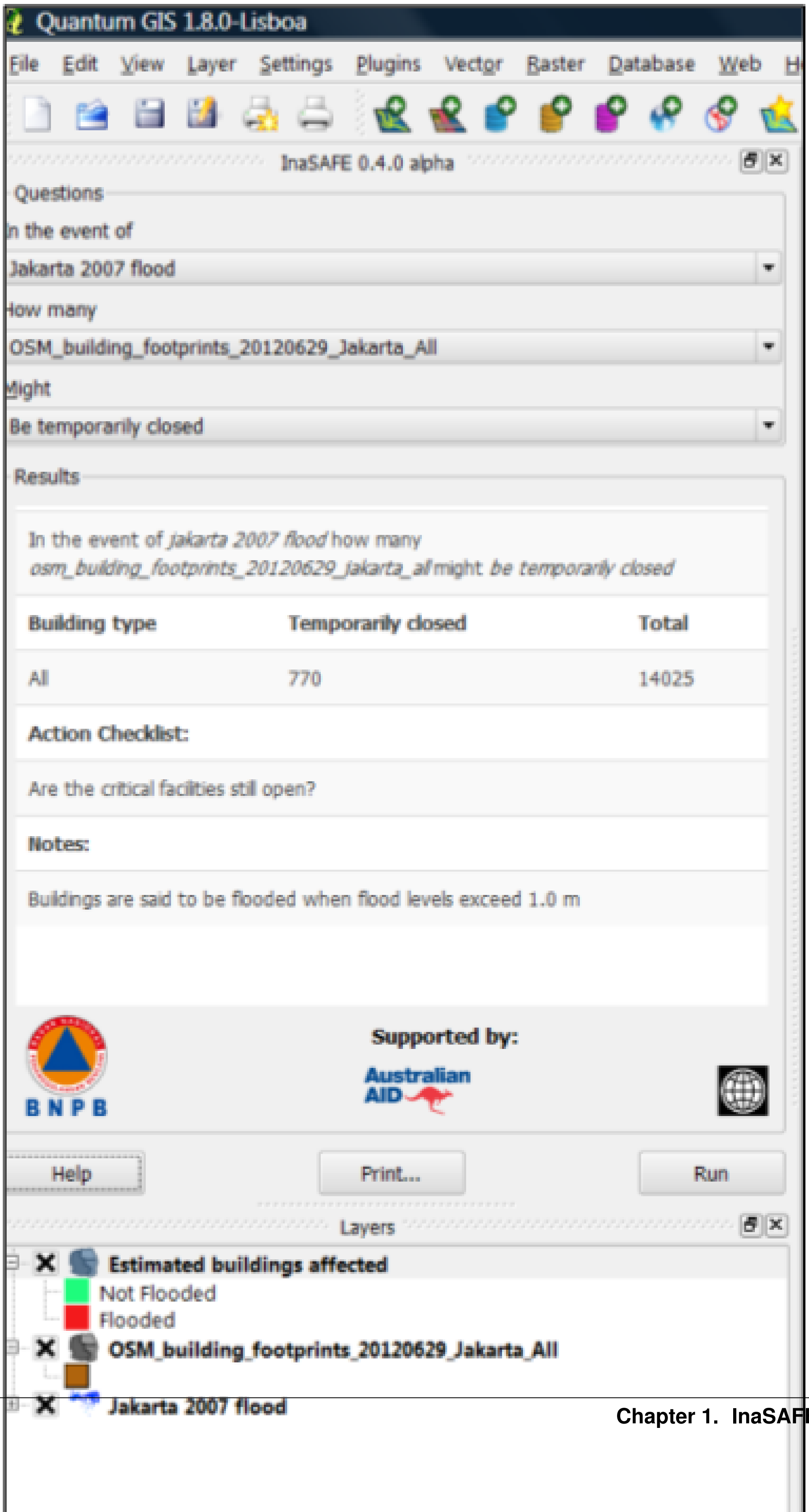
The impact function (**Might**) will spatially combine the hazard and exposure input layers in order to postulate what the impacts of the hazard will be on the exposure infrastructure or people. By selecting a combination from the "In the event of" and "How many" combo boxes, an appropriate set of impact functions will be listed in the "Might" combo box.

Impact scenarios are predefined depending on what the decision-maker is looking for. For our flood analysis in Jakarta, we only have on predefined impact function which asks: **In case of flood event, how many buildings might be temporarily closed?** As we see on the previous step, this is filled automatically by default in the InaSAFE panel dock as soon as the hazard [flood] and exposure [buildings] layers are entered correctly.

1.3.9 The Results section

Now that we have our two input layers and that we know what impacts we would like to assess, click on the **Run** button at the bottom to start the impact analysis. At the end of the process, figures will be shown in the **Results** section, a new layer will be added in the QGIS layer panel representing the result of the impact function, and the map will differentiate affected and non-affected building.





In the event of *jakarta 2007 flood* how many *osm_building_footprints_20120629_jakarta_all* might *be temporarily closed*

Building type	Temporarily closed	Total
All	770	14025

Action Checklist:

Are the critical facilities still open?

Notes:

Buildings are said to be flooded when flood levels exceed 1.0 m

The result shows **total number of buildings** and the **number of buildings that might be temporarily closed** in the event of a flood. Also, there is an **Action Checklist** where the question: *Are the critical facilities still open?* And a **Note** description explaining that buildings are said flooded when the flood level exceeds 1m.

1.3.10 Enhancing the Map Output

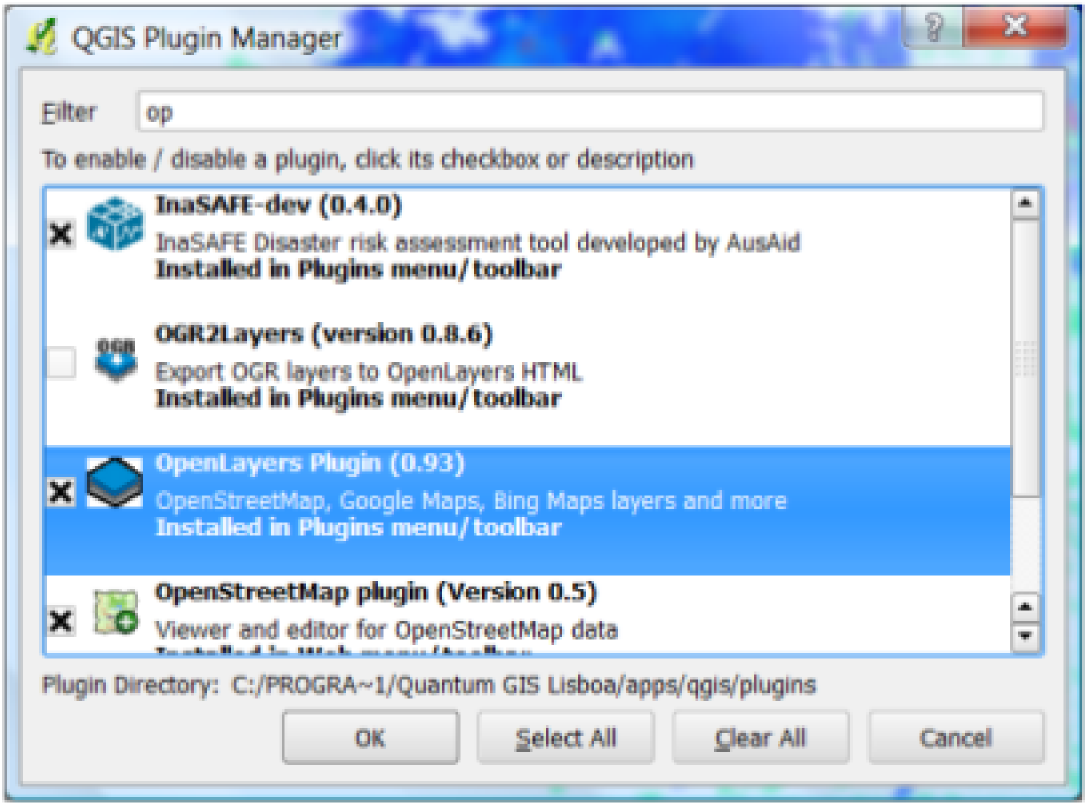
The final output map can be enhanced using cartographic functions in QGIS. Styles can be changed, background layer or other relevant layers can be added, layout can be changed using the **Print Composer** in QGIS.

If you would like to add openlayer background to your map, all you need to do is add a new plugin called **OpenLayer Plugin** in QGIS and follow the same steps as we did to acquire the InaSAFE plugin.

You can download the plugin from the website http://build.sourcepole.ch/qgis/plugins.xml, and put it in the QGIS plugin directory `C:Users<your username>.qgispythonplugins`.

To enable the OpenLayer plugin, go to the QGIS **Plugin Manager** and select **OpenLayers Plugin.**

Once installed you should be able to use background imagery and tiles from google, yahoo, bing and openstreetmap.



To enable / disable a plugin, click its checkbox or description

[✓] InaSAFE-dev (0.4.0)
InaSAFE Disaster risk assessment tool developed by AusAid
Installed in Plugins menu/toolbar

[ ] OGR2Layers (version 0.8.6)
Export OGR layers to OpenLayers HTML
Installed in Plugins menu/toolbar

[✓] OpenLayers Plugin (0.93)
OpenStreetMap, Google Maps, Bing Maps layers and more
Installed in Plugins menu/toolbar

[✓] OpenStreetMap plugin (Version 0.5)
Viewer and editor for OpenStreetMap data

Plugin Directory: C:/PROGRA~1/Quantum GIS Lisboa/apps/qgis/plugins

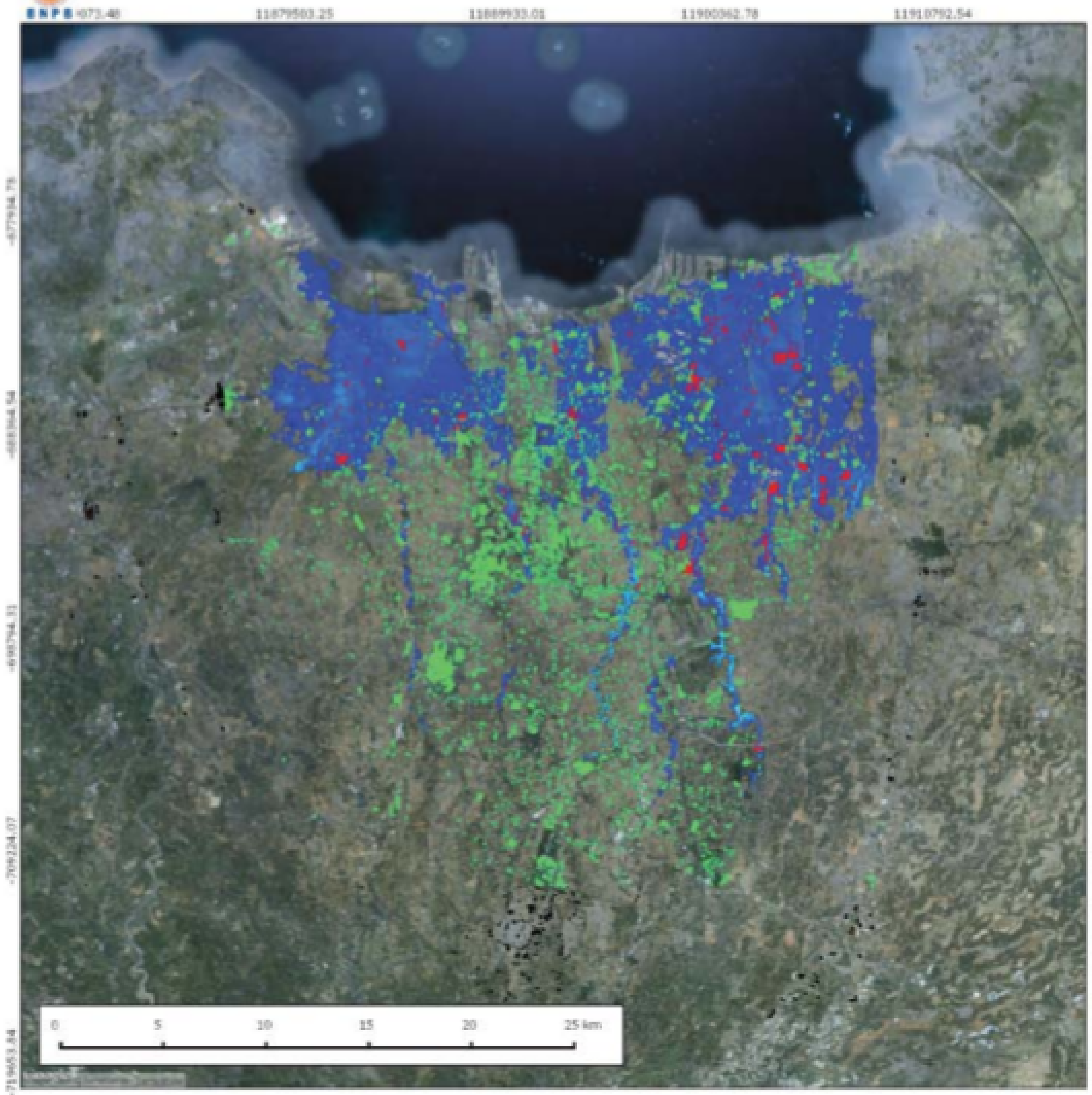
For our exercise, let's add the Google Satellite view to our map as a background. Make sure that the background is not on top of the other active layers.

1.3.11 Print Results

The data shown on the screen can be saved into a **PDF file** by clicking on **Print** at the bottom of the InaSAFE panel. The PDF file contains then the **legend** for the result of the impact assessment, the **map** created and a **table** summarizing the results from the impact function.

However, any change that you want to make into the final map document should be done **before** clicking on the **Print** button of the INASAFE dock panel. The print should be only use once the data is exactly as you want it to be displayed.

InaSAFE - Indonesian Scenario Assessment for Emergencies



Buildings inundated

Legend

🟩 Not Flooded
🟥 Flooded

In the event of jakarta 2007 flood how many osm_building_footprints_20120629_jakarta_all might be temporarily closed

Building type	Temporarily closed	Total
All	770	14025

Action Checklist:

Are the critical facilities still open?

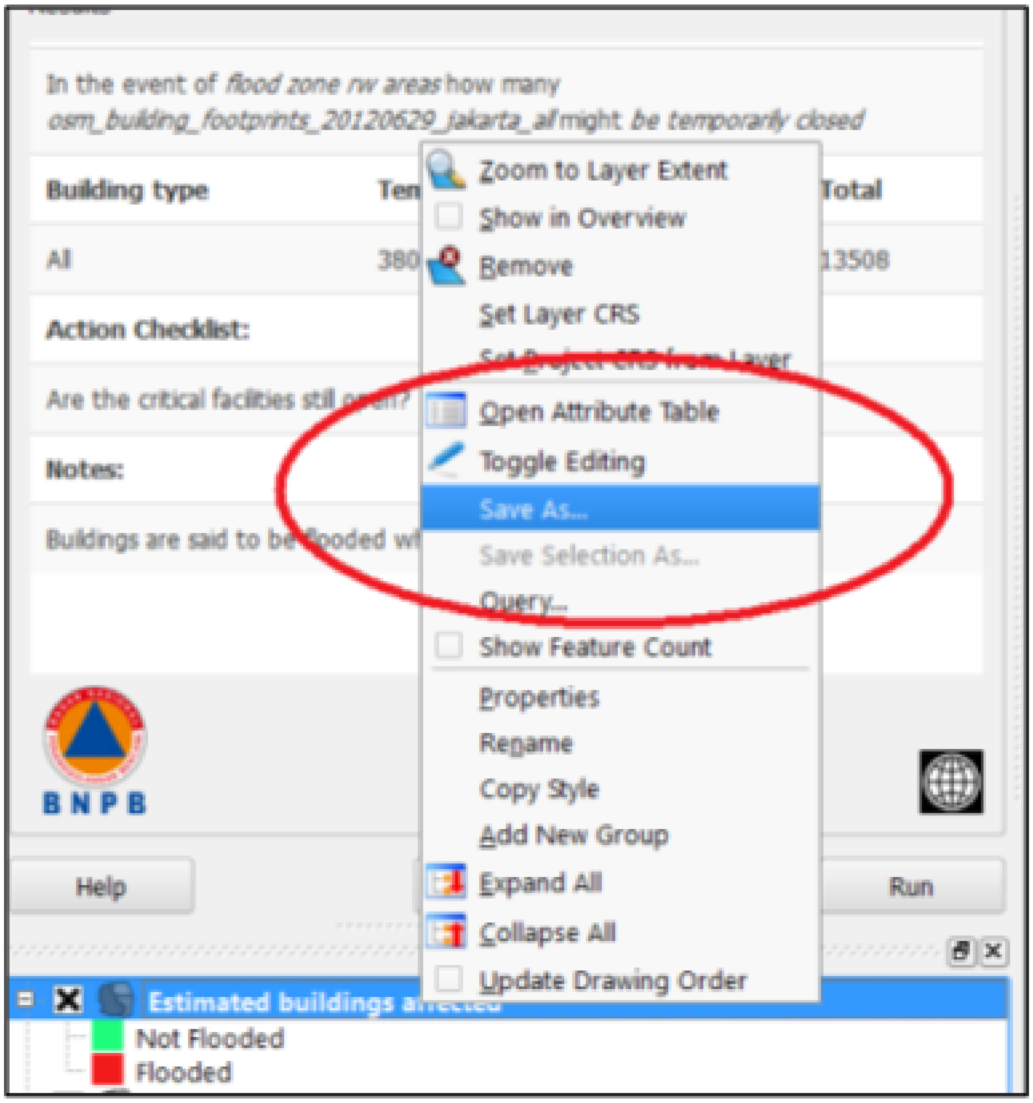
Notes:

Buildings are said to be flooded when flood levels exceed 1.0 m

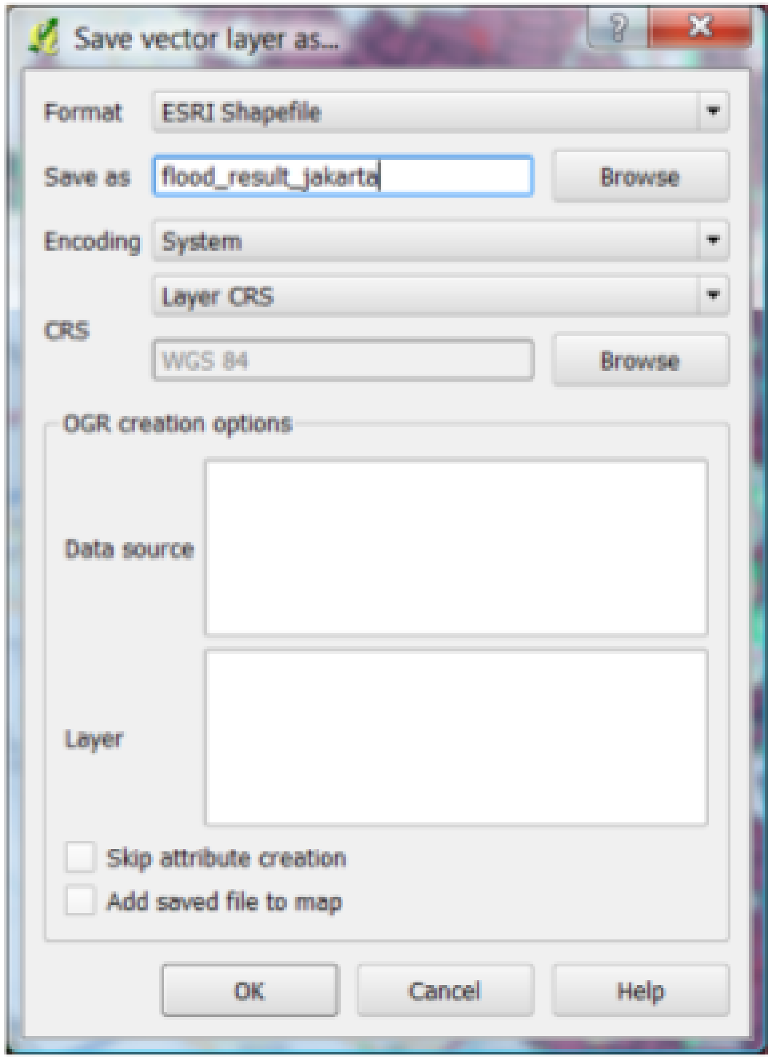
InaSAFE has been jointly developed by BNPB, AusAid & the World Bank

1.3.12 Save results and QGIS project

The output layer result of the assessment can be saved by right clicking on the layer.

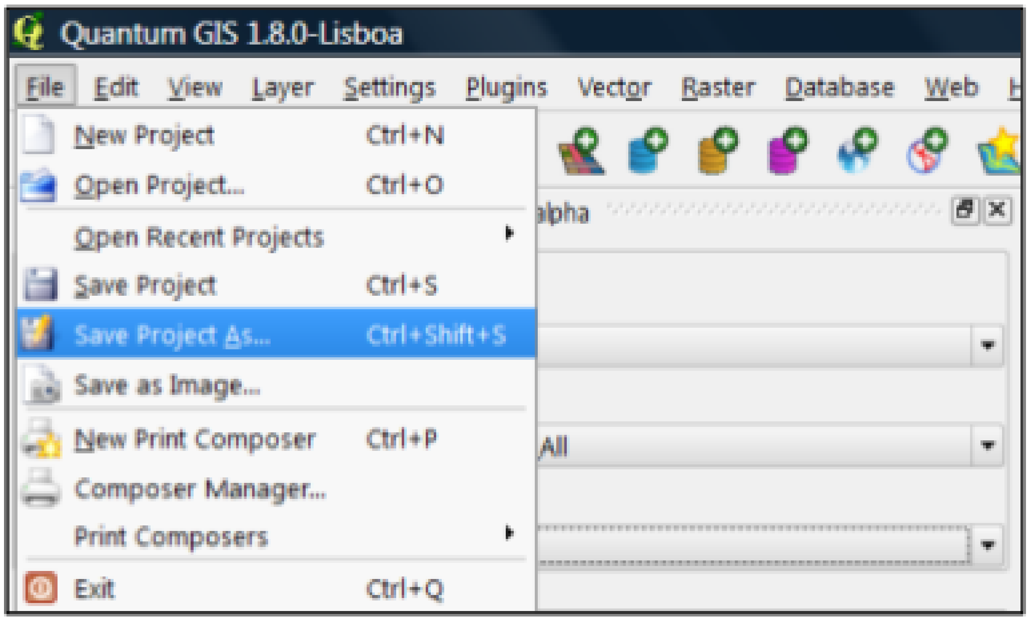


Then **Save As** a shapefile or a raster. However the keywords and statistics do not get saved.

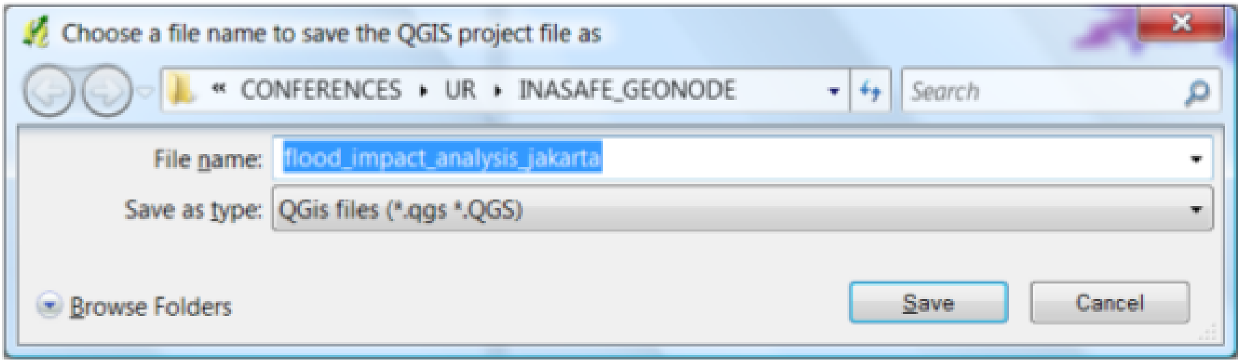


Format: ESRI Shapefile
Save as: flood_result_jakarta
Encoding: System
CRS: Layer CRS
WGS 84

You can also save the project under QGIS so that you can access your current window view anytime needed. For that, go to the **File** menu on QGIS toolbar. Click on **Save Project As.**



Give a name to the project and put it in the directory you want to store your work. Then click on **Save.**


Choose a file name to save the QGIS project file as
« CONFERENCES ▸ UR ▸ INASAFE_GEONODE
Search
Browse Folders
Save
Cancel

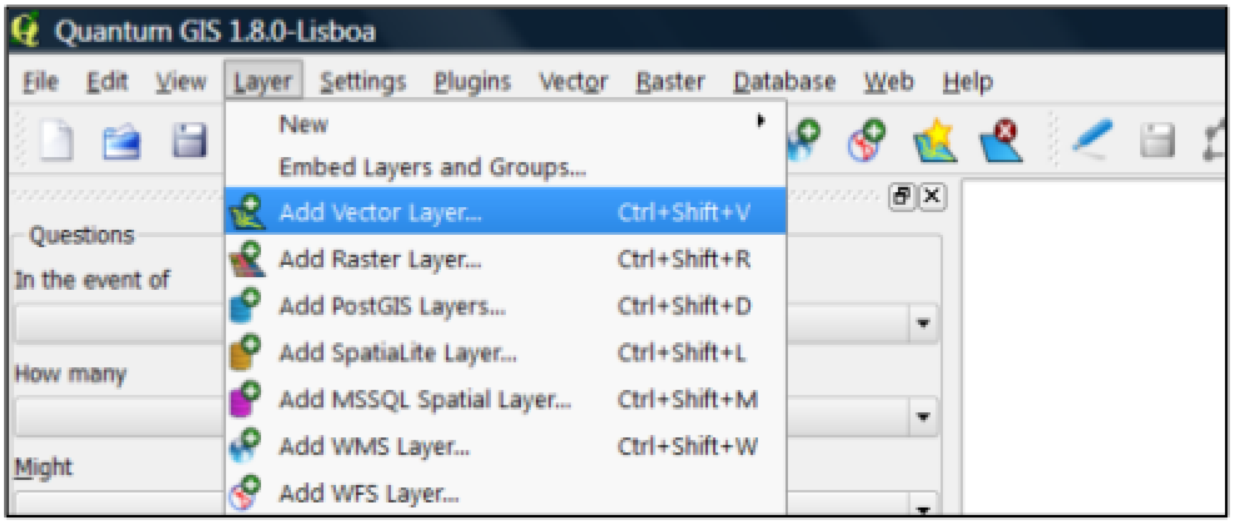
File name: flood_impact_analysis_jakarta

Save as type: QGis files (*.qgs *.QGS)

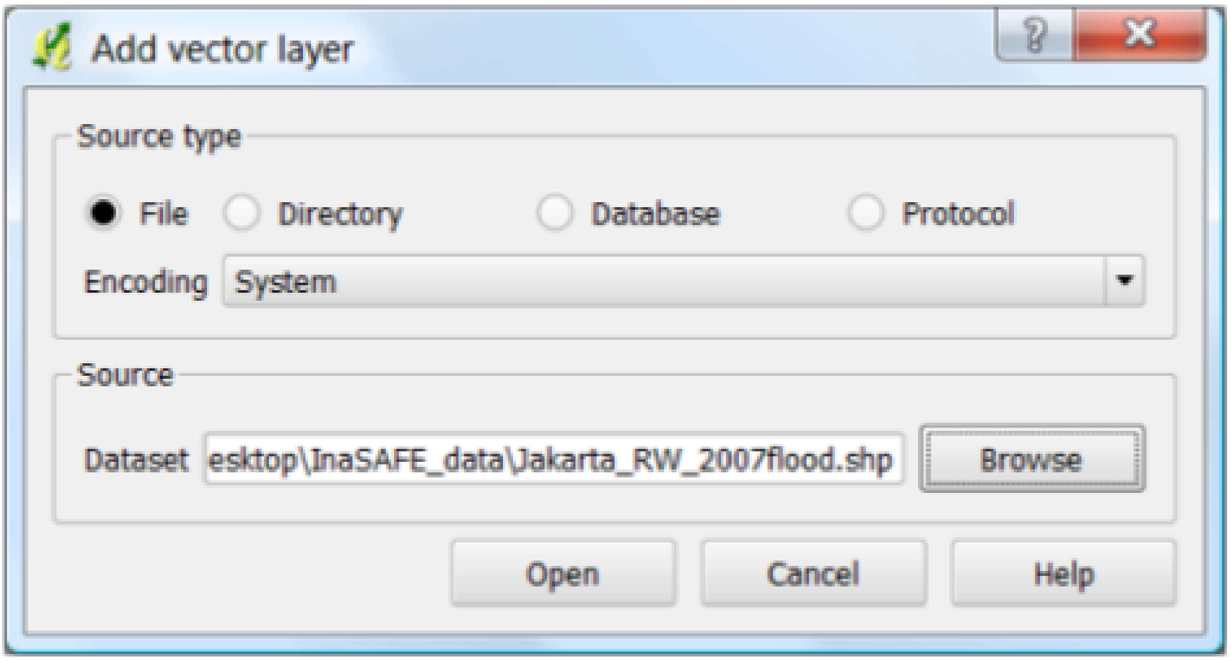
Now that the project is saved under QGIS, you can go back to your work anytime you need. However, the statistical data will be lost whenever the project is closed. To get the data back, you will need to redo the impact analysis process we described above from **Run.**

The impact assessment can also be replicated for flood zone areas; the only change is to add the layer as a vector and using keyword flood (wet/dry).

Go to QGIS **Layer/Add Vector Layer.**


Quantum GIS 1.8.0-Lisboa
File Edit View Layer Settings Plugins Vector Raster Database Web Help
New
Embed Layers and Groups...
Add Vector Layer... Ctrl+Shift+V
Add Raster Layer... Ctrl+Shift+R
Add PostGIS Layers... Ctrl+Shift+D
Add SpatiaLite Layer... Ctrl+Shift+L
Add MSSQL Spatial Layer... Ctrl+Shift+M
Add WMS Layer... Ctrl+Shift+W
Add WFS Layer...
Questions
In the event of
How many
Might

Add the **Jakarta_RW_2007flood.shp**

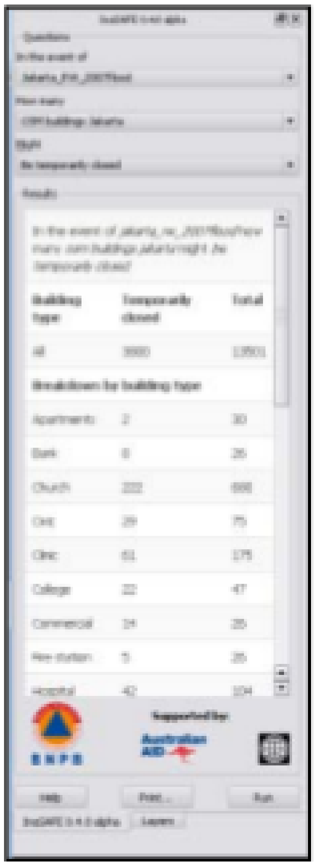


Source type: File

Encoding: System

Dataset: esktop\InaSAFE_data\Jakarta_RW_2007flood.shp

The impact functions can also be modified to pull out certain attributes from the input layers, in this case it can tease out the breakup of building types.



InaSAFE 0.4.0 alpha

Questions

In the event of

Jakarta_[illegible]_2007flood

How many

OSM buildings Jakarta

Might

Be temporarily closed

Results

In the event of jakarta_[illegible]_2007flood how many osm buildings jakarta might be temporarily closed

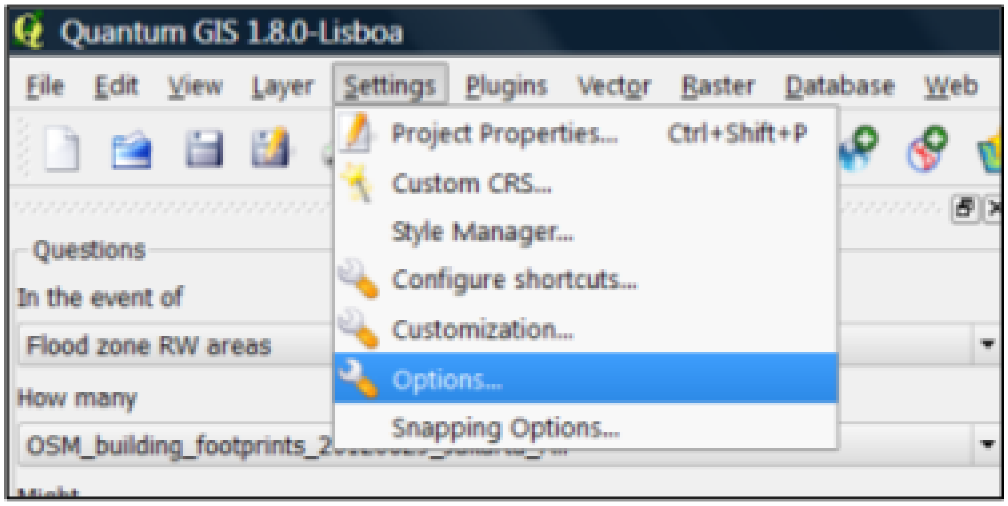
Building type	Temporarily closed	Total
All	[illegible]	[illegible]
Breakdown by building type		
Apartments	2	30
Bank	8	26
Church	222	[illegible]
Civic	29	75
Clinic	61	175
College	22	47
Commercial	14	26
Fire station	5	26
Hospital	42	104

Note: The flood zone areas were derived from sub-village administration boundaries and defined as areas that were flooded in the last large flood in Jakarta in 2007.

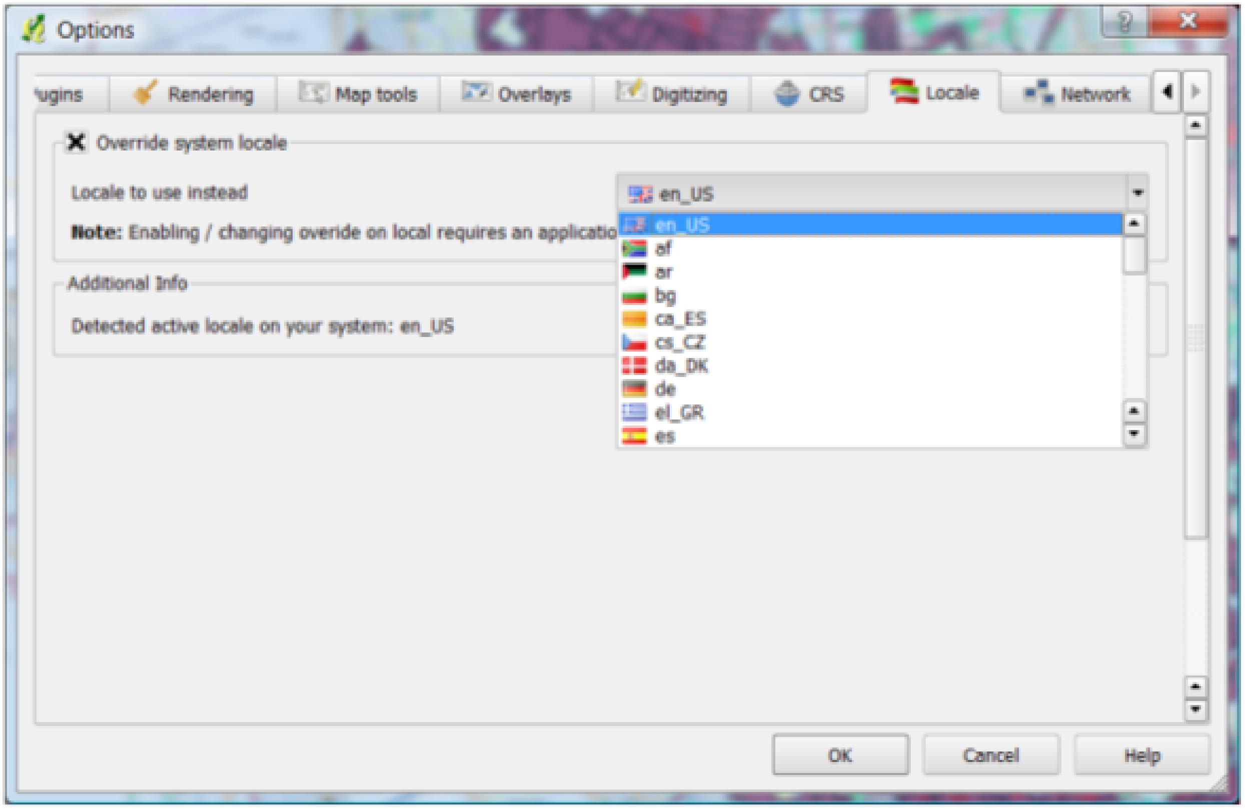
1.4 International Language

Both QGIS and InaSAFE come with multi-lingual support.

Go to QGIS Settings/options.



Go to the Locale tab, and click to **Override system locale.** You must close the QGIS project and reopen for the language to take hold.


Options
ugins Rendering Map tools Overlays Digitizing CRS Locale Network
Override system locale
Locale to use instead
en_US
Note: Enabling / changing overide on local requires an applicatio
en_US
af
ar
bg
ca_ES
cs_CZ
da_DK
de
el_GR
es
Additional Info
Detected active locale on your system: en_US
OK Cancel Help

At this point InaSAFE has only been translated into Indonesian, however there is a framework that can support all languages outlined in QGIS.

1.5 Thank You

THANK YOU FOR YOUR PARTICIPATION! :)

Let us know how you enjoyed the training and what you think about the tool.

THE INASAFE TEAM

- Indonesian Disaster Management Agency (BNPB)
- AusAID - Australia-Indonesia Facility for Disaster Reduction
- World Bank - Global Facility for Disaster Reduction and Recovery

CHAPTER

TWO

INASAFE USER DOCUMENTATION

This section of the documention describes how to use the InaSAFE system.

2.1 Installing InaSAFE

Note: InaSAFE is a plugin for Quantum GIS (QGIS), so QGIS must be installed first.

2.1.1 Via QGIS Python Plugin Repository

To install the InaSAFE, use the plugin manager in QGIS:

```
Plugins -> Fetch Python Plugins
```

Then search for 'InaSAFE', select it and click the install button. The plugin will now be added to your plugins menu.

2.1.2 From Zip Archive

Warning: This installation method is not recommended unless you have no internet access or wish to use a specific version of InaSAFE. Please install using the plugin repository described above rather.

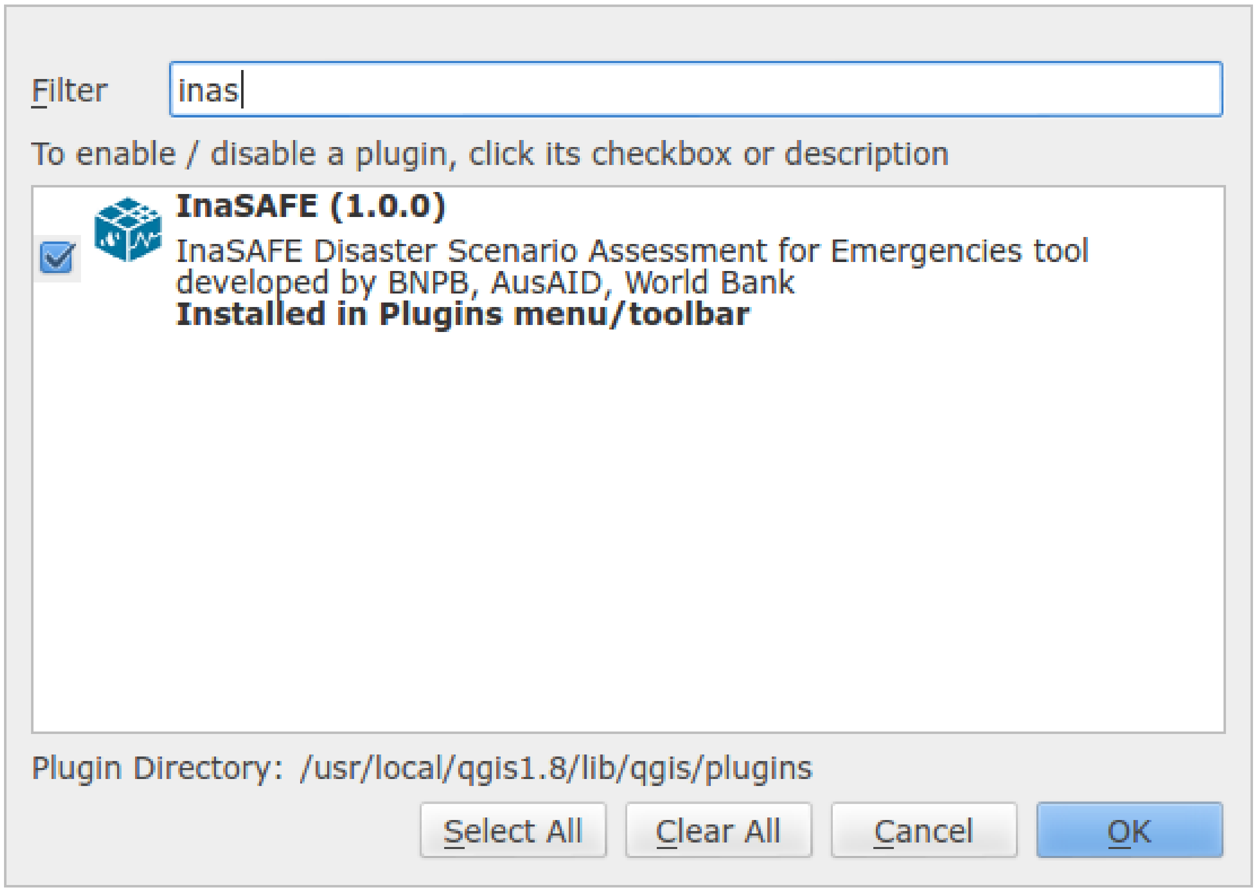
We make regular releases of the InaSAFE plugin and they are available at https://github.com/AIFDR/inasafe/downloads. Simply choose the most recent (i.e. the one with the largest version number) and save it to your hard disk.

Now extract the zip file into the QGIS plugins directory. Under windows the plugins directory is under `c:\Users\<your username>\.qgis\python\plugins`.

After extracting the plugin, it should be available as `c:\Users\<your username>\.qgis\python\plugins\inasafe\`.

Mac and Linux users need to follow the same procedure but instead the plugin directory will be under your $HOME directory.

Once the plugin is extracted, start QGIS and enable it from the plugin manager. To do this open the plugin manager (*Plugins* → *Manage plugins...*) and type `insafe` into the filter box. You should see the InaSAFE plugin appear in the list. Now tick the checkbox next to it to enable the plugin.



Filter inas

To enable / disable a plugin, click its checkbox or description

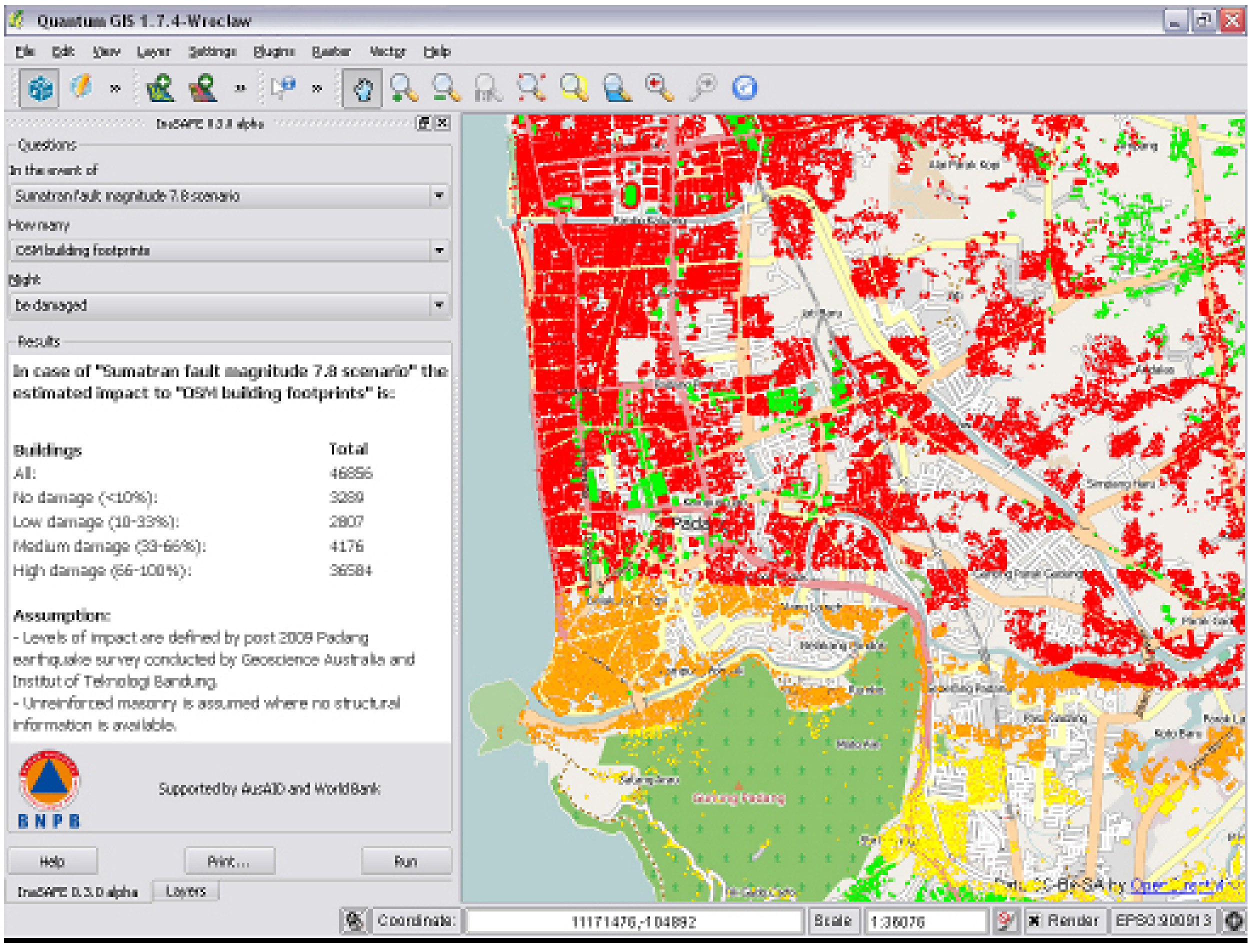
✓ **InaSAFE (1.0.0)**
InaSAFE Disaster Scenario Assessment for Emergencies tool developed by BNPB, AusAID, World Bank
Installed in Plugins menu/toolbar

Plugin Directory: /usr/local/qgis1.8/lib/qgis/plugins

2.1.3 System Requirements

- A standard PC with at least 4GB of RAM running Windows, Linux or Mac OS X
- The Open Source Geographic Information System QGIS (http://www.qgis.org). InaSAFE requires QGIS version 1.7 or newer.

2.2 InaSAFE Screenshots



In case of "Sumatran fault magnitude 7.8 scenario" the estimated impact to "OSM building footprints" is:

Buildings	Total
All:	46856
No damage (<10%):	3289
Low damage (10-33%):	2807
Medium damage (33-66%):	4176
High damage (66-100%):	36584

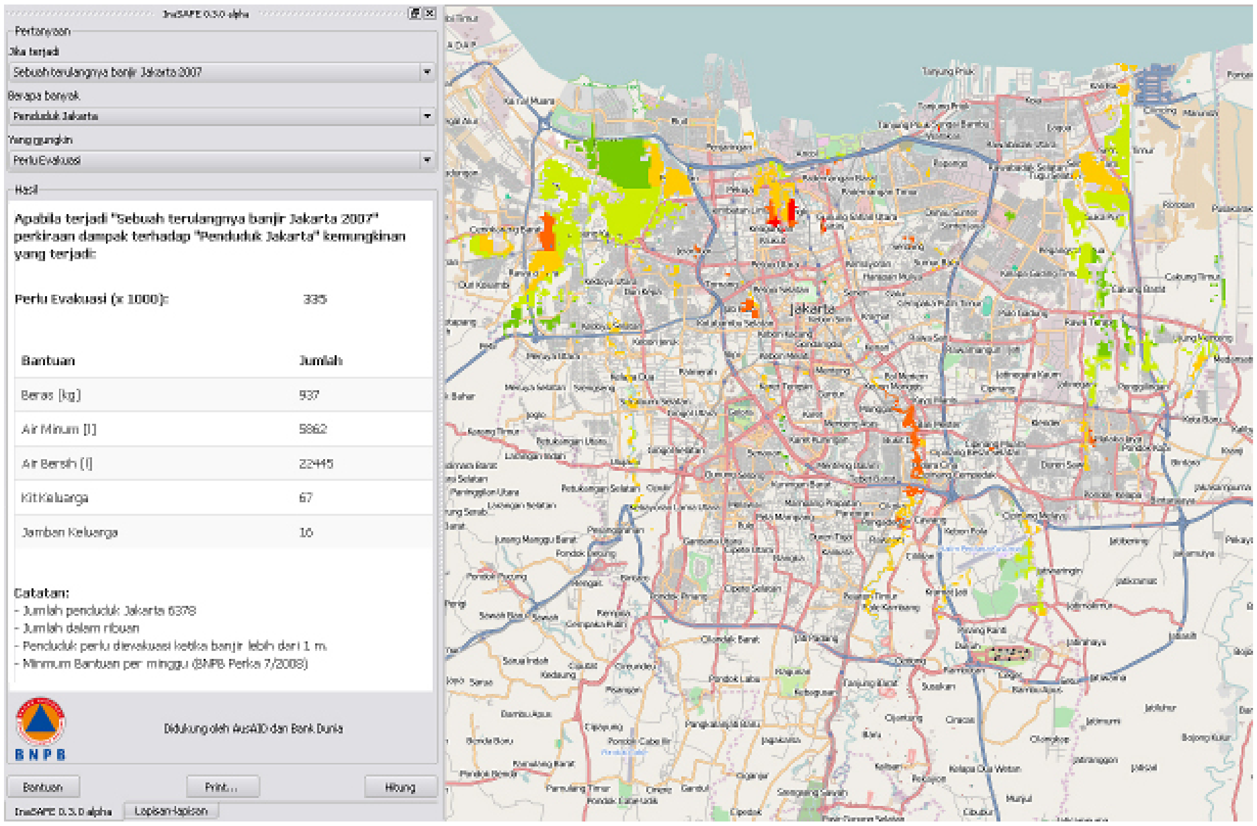
Assumption:
- Levels of impact are defined by post 2009 Padang earthquake survey conducted by Geoscience Australia and Institut of Teknologi Bandung.
- Unreinforced masonry is assumed where no structural information is available.

2.3 Using the InaSAFE Plugin

This document describes the useage of the InaSAFE 'dock panel' - which is the main interface for running risk scenarios within the Quantum GIS environment.

Note: In order to use the InaSAFE tool effectively, you should probably also read the *Keywords System* and *Impact Functions* documentation before you get started.

The InaSAFE Dock panel is the main way to interact with the tools that are provided in InaSAFE. After you have installed the InaSAFE plugin, the dock panel will automatically load in QGIS, appearing on the left hand side of the screen.



Apabila terjadi "Sebuah terulangnya banjir Jakarta 2007" perkiraan dampak terhadap "Penduduk Jakarta" kemungkinan yang terjadi:

Perlu Evakuasi (x 1000): 335

Bantuan	Jumlah
Beras [kg]	937
Air Minum [l]	5862
Air Bersih [l]	22445
Kit Keluarga	67
Jamban Keluarga	16

Catatan:
- Jumlah penduduk Jakarta 6378
- Jumlah dalam ribuan
- Penduduk perlu dievakuasi ketika banjir lebih dari 1 m.
- Minimum Bantuan per minggu (BNPB Perka 7/2008)

You can drag and drop the dock panel to reposition it in the user interface. For example, dragging the panel towards the right margin of the QGIS application will dock it to the right side of the screen.

There are 3 main areas to the panel:

- the *Questions* area
- the *Results* area
- the buttons area

At any time you can obtain help in InaSAFE by clicking on the *help* buttons provided on each dock and dialog.

2.3.1 The Questions Area

The intention of InaSAFE is to make it really simple and easy to perform your impact analysis. The question area provides a simple way for you to formulate what it is you want to find out? All questions are formulated in the form:

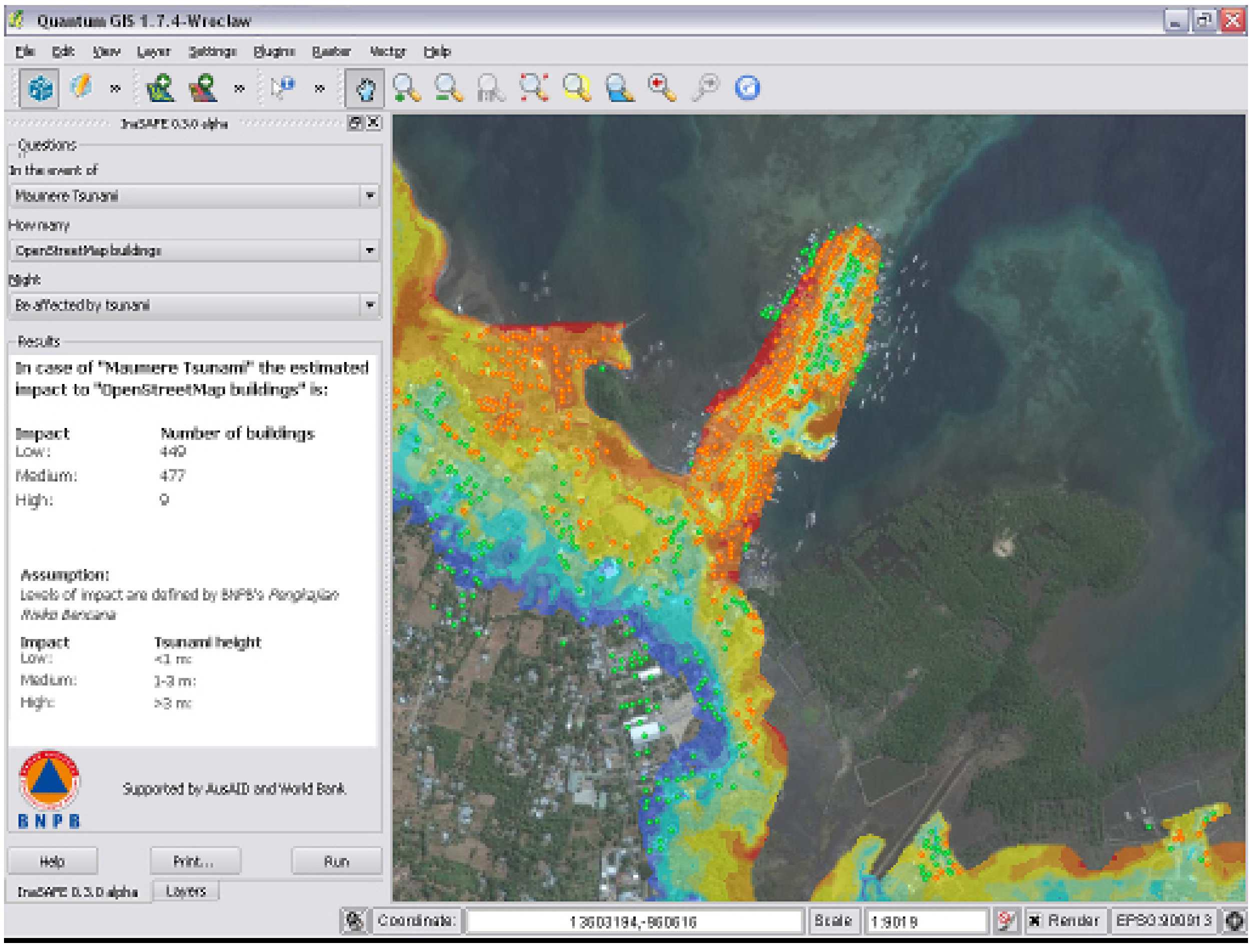
If [**hazard**] how many [**exposure**] might [**impact**].

For example:

If **there is a flood** how many **schools** might **be closed**.

In order to answer such questions, the InaSAFE developers have built a number of **impact functions** that cover scenarios such as flood, tsunami, volcanic ash fall, earthquake and so on. You can read our impact function documentation to find out more information about the various *Impact Functions* implemented.

The formulation of these questions if carried out by loading layers into QGIS that represent either **hazard** or **exposure** scenarious.



In the event of: Maumere Tsunami

How many: OpenStreetMap buildings

Might: Be affected by tsunami

In case of "Maumere Tsunami" the estimated impact to "OpenStreetMap buildings" is:

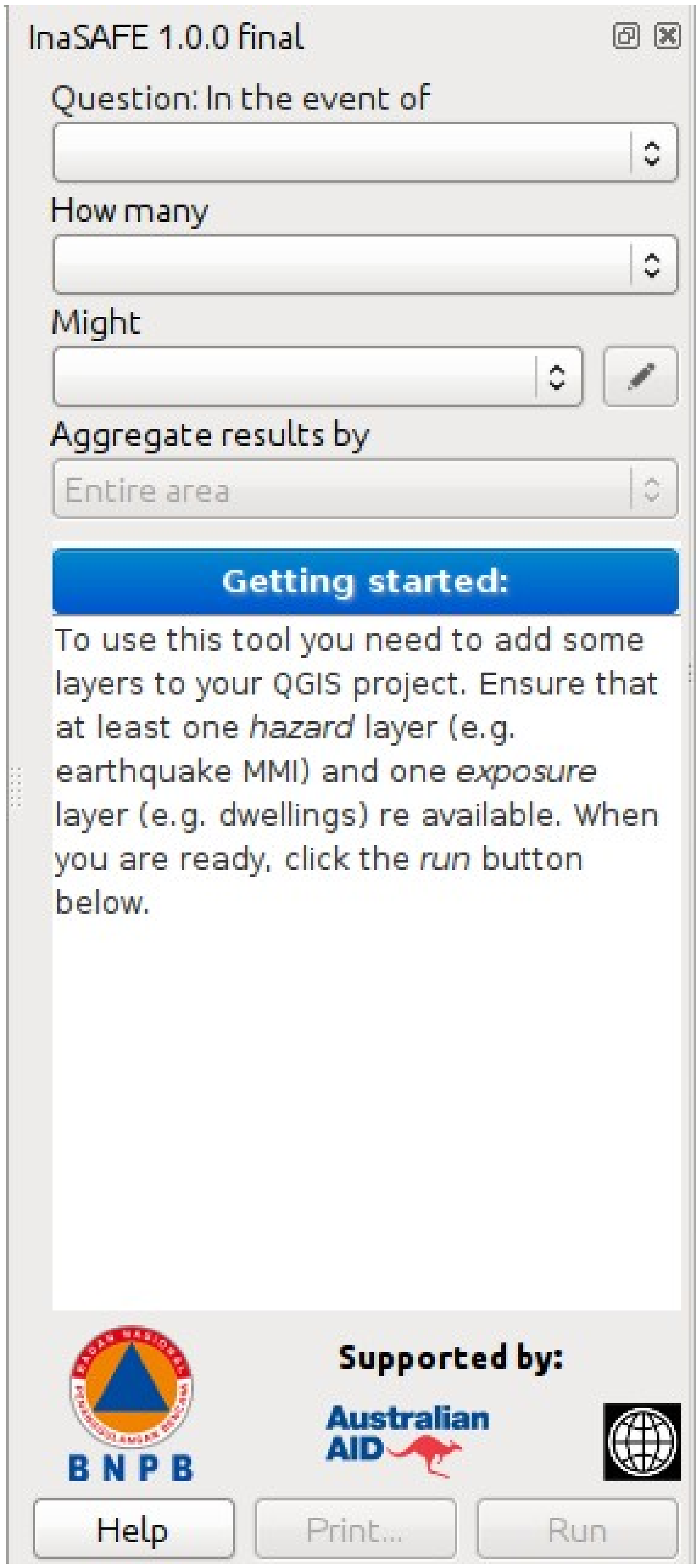
Impact	Number of buildings
Low:	449
Medium:	477
High:	9

Assumption:
Levels of impact are defined by BNPB's *Pengkajian Risiko Bencana*

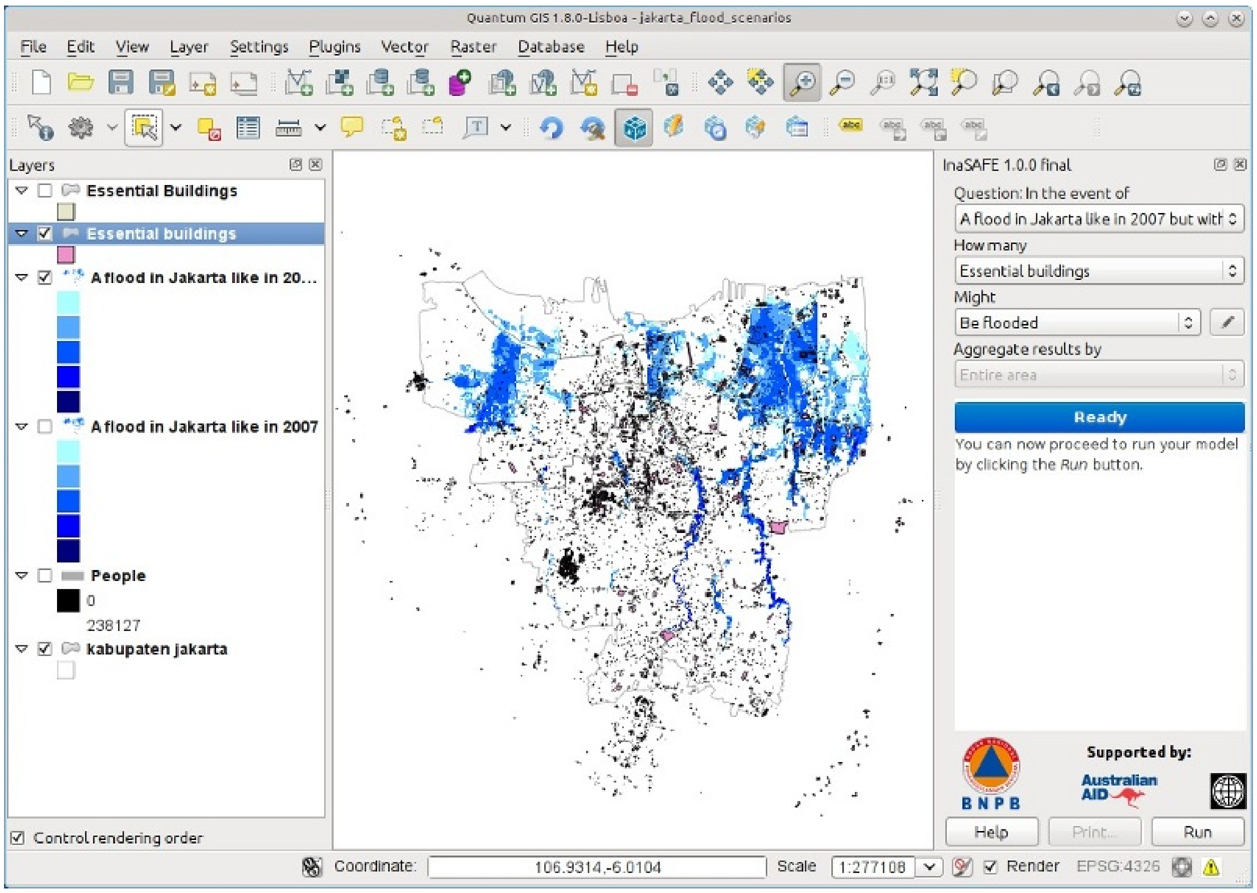
Impact	Tsunami height
Low:	<1 m:
Medium:	1-3 m:
High:	>3 m:

Supported by AusAID and World Bank

Coordinate: 13603194,-960616 Scale 1:9019 EPSG:900913



To use this tool you need to add some layers to your QGIS project. Ensure that at least one *hazard* layer (e.g. earthquake MMI) and one *exposure* layer (e.g. dwellings) re available. When you are ready, click the *run* button below.


Quantum GIS 1.8.0-Lisboa - jakarta_flood_scenarios
File Edit View Layer Settings Plugins Vector Raster Database Help
Layers
Essential Buildings
Essential buildings
A flood in Jakarta like in 20...
A flood in Jakarta like in 2007
People
0
238127
kabupaten jakarta
Control rendering order
InaSAFE 1.0.0 final
Help
Print...
Run
Supported by:
Australian AID
BNPB

Question: In the event of
A flood in Jakarta like in 2007 but with
How many
Essential buildings
Might
Be flooded
Aggregate results by
Entire area
Ready
You can now proceed to run your model by clicking the *Run* button.

Coordinate: 106.9314,-6.0104 Scale 1:277108 Render EPSG:4326

- A **hazard** (*In the event of*) may be represented as, for example, a raster layer in QGIS where each pixel in the raster represents the current flood depth following an inundation event.
- An **exposure** (*How many*) layer could be represented, for example, as vector polygon data representing building outlines, or a raster outline where each pixel represents the number of people resident in that cell.

The **impact function** (*Might*) will combine these two input layers in a mathematical model in order to postulate what the impacts of the hazard will be on the exposure infrastructure or people.

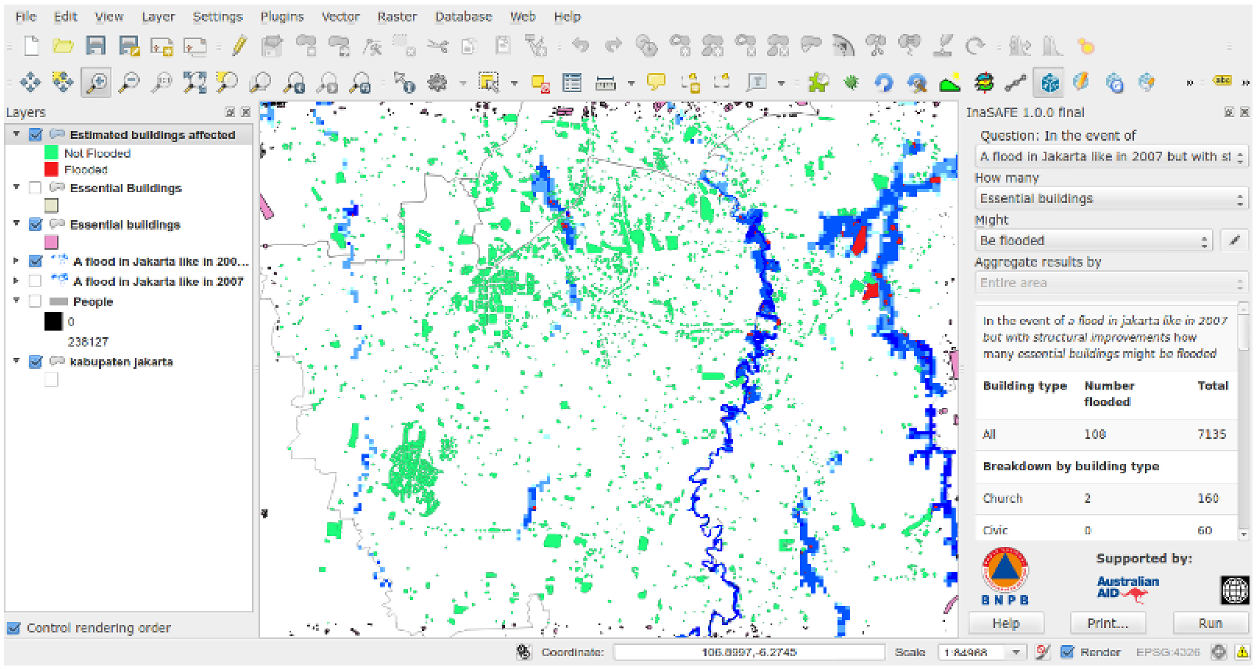
By selecting a combination from the *In the event of* and *How many* combo boxes, an appropriate set of impact functions will be listed in the *Might* combo box.

You may be wondering how the InaSAFE plugin determines whether a layer should be listed in the *In the event of* or *How many* combo boxes? The plugin relies on simple keyword metadata to be associated with each layer. The keyword system is described in detail in *Keywords System*. Each layer that has a keyword allocating it's **category** to **hazard** will be listed in the *In the event of* combo. Similarly, a **category** of **exposure** in the keywords for a layer will result in it being listed under the *How many* combo.

InaSAFE uses the combination of **category**, **subcategory**, **units** and **datatype** keywords to determine which **impact functions** will be listed in the *Might* combo.

2.3.2 The results area

The *Results* area is used to display various useful feedback items to the user. Once an impact scenario has been run (see next section below), a summary table will be shown.



Question: In the event of
A flood in Jakarta like in 2007 but with st
How many
Essential buildings
Might
Be flooded
Aggregate results by
Entire area

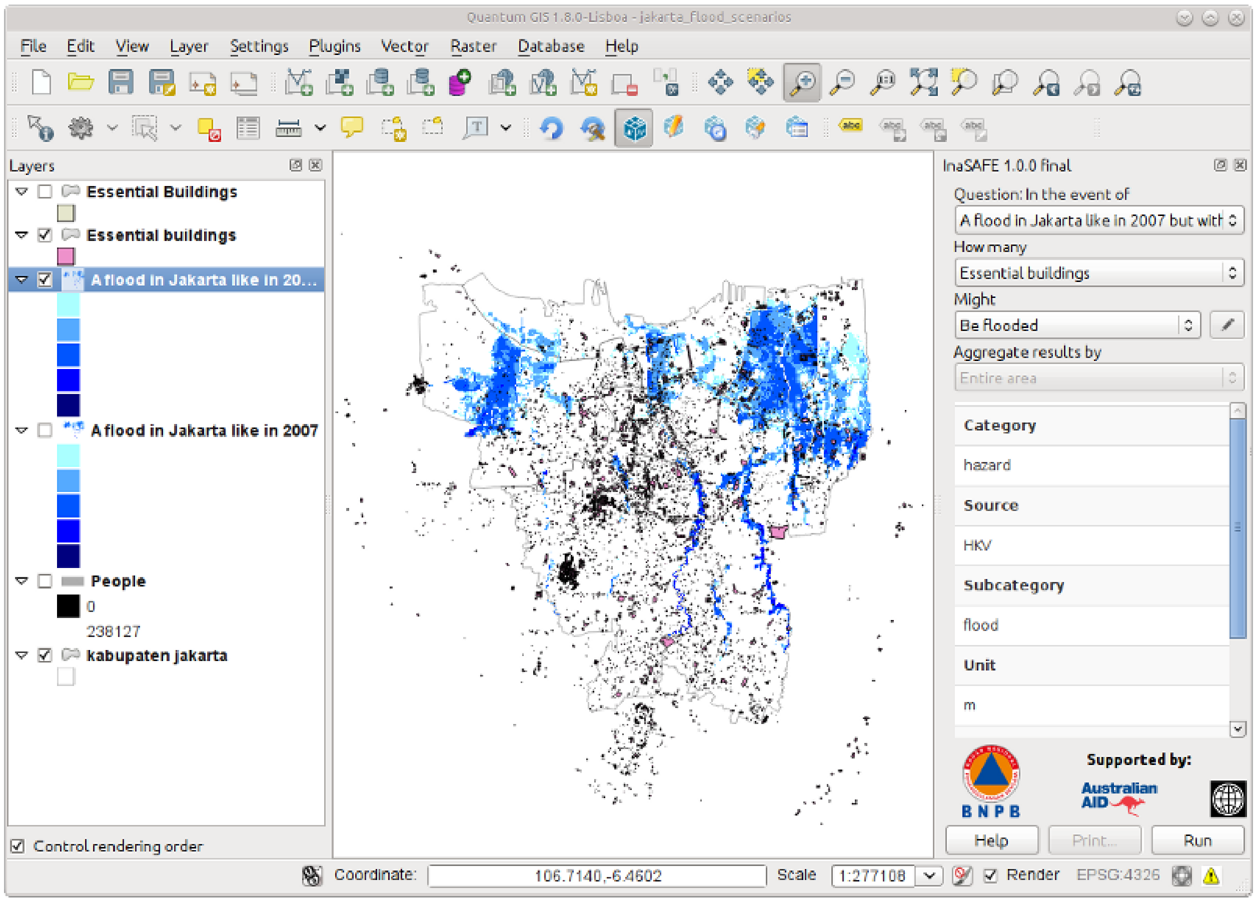
In the event of a flood in jakarta like in 2007 but with structural improvements how many essential buildings might be flooded

Building type	Number flooded	Total
All	108	7135
Breakdown by building type		
Church	2	160
Civic	0	60

If you select an **impact layer** (i.e. a layer that was produced using an InaSAFE impact function), in the QGIS layers list, this summary will also be displayed in the results area.

When you select a **hazard** or **exposure** layer in the QGIS layers list, the keywords for that layer will be shown in the *Results* area, making it easy to understand what metadata exists for that layer.

The *Results* area is also used to display status information. For example, when a suitable combination of **hazard** (*In the event of*), **exposure** (*How many*) and **impact function** (*In the event of*) are selected, the results area will be updated to indicate that you can proceed to run the impact scenario calculation.



Layers:

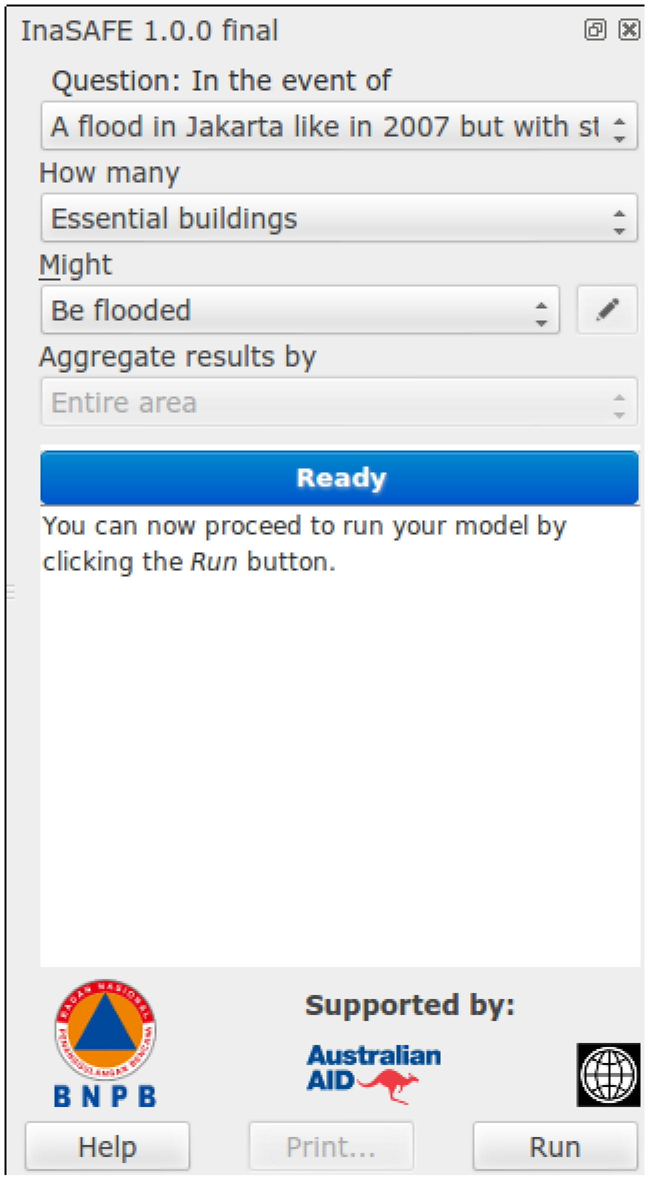
- Essential Buildings
- ✓ Essential buildings
- ✓ A flood in Jakarta like in 20...
- A flood in Jakarta like in 2007
- People
  - 0
  - 238127
- ✓ kabupaten jakarta

✓ Control rendering order

Question: In the event of
A flood in Jakarta like in 2007 but with
How many
Essential buildings
Might
Be flooded
Aggregate results by
Entire area

Category
hazard
Source
HKV
Subcategory
flood
Unit
m

Coordinate: 106.7140,-6.4602 Scale 1:277108 ✓ Render EPSG:4326


InaSAFE 1.0.0 final

Question: In the event of

A flood in Jakarta like in 2007 but with st

How many

Essential buildings

Might

Be flooded

Aggregate results by

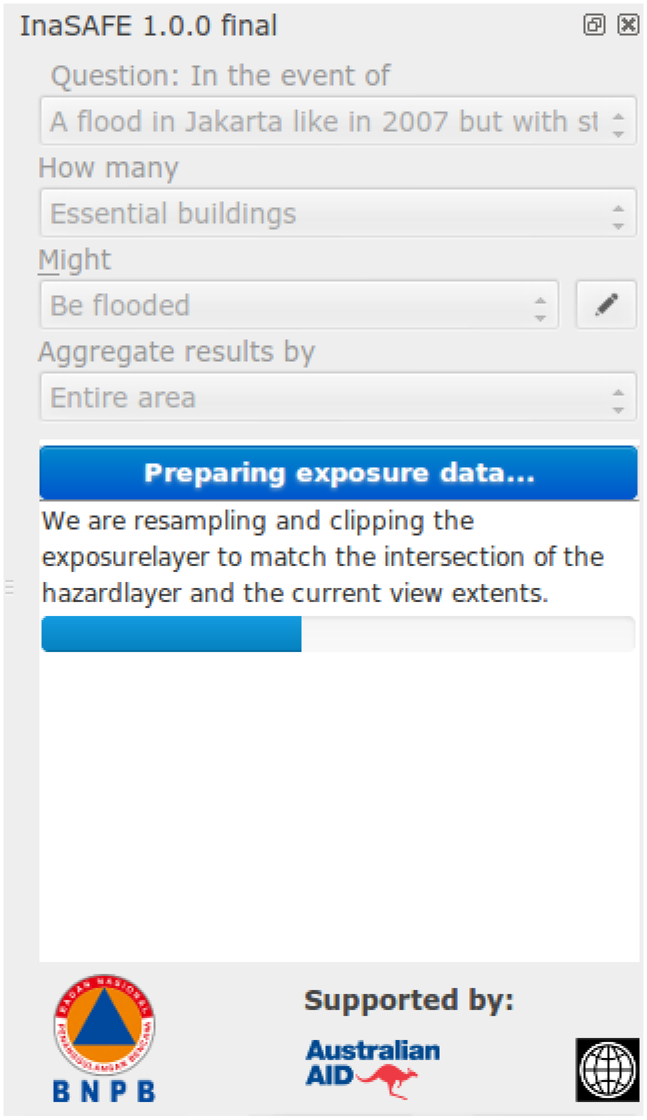
Entire area

Ready

You can now proceed to run your model by clicking the *Run* button.

Supported by:

While a scenario is being computed, the *Results* area displays the current progress of the analysis.


InaSAFE 1.0.0 final
Question: In the event of
A flood in Jakarta like in 2007 but with st
How many
Essential buildings
Might
Be flooded
Aggregate results by
Entire area
Preparing exposure data...
We are resampling and clipping the exposurelayer to match the intersection of the hazardlayer and the current view extents.

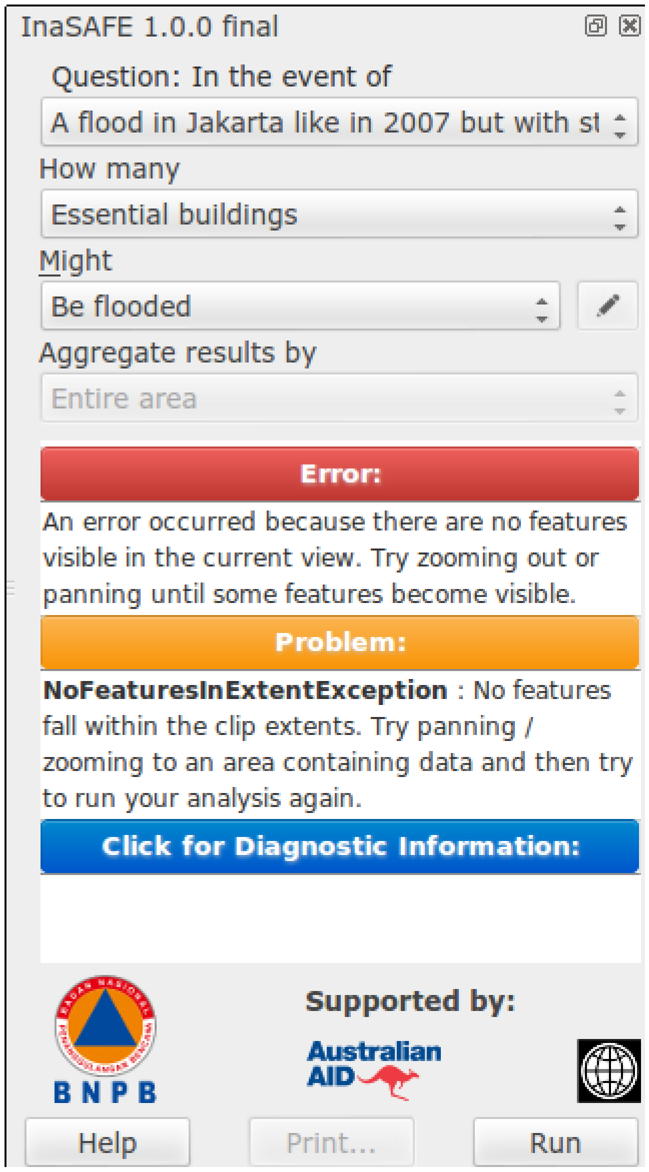
Finally, the *Results* area is also used to display any error messages so that the user is informed as to what went wrong and why.

Note: At the bottom of error display you may see button like this: .. image:: ../../toggle-traceback.png If you click on this button, it will display a box which will contain useful diagnostic information which can be submitted as part of a bug report if you think the error was incorrect.

2.3.3 The buttons area

The buttons area contains three buttons:

- *Help* - click on this if you need context help, such as the document you are reading right now!
- *Print...* - click on this if you wish to create a pdf of your impact scenarion project. An **impact layer** must be active before the *Print...* button will be enabled.



An error occurred because there are no features visible in the current view. Try zooming out or panning until some features become visible.

NoFeaturesInExtentException : No features fall within the clip extents. Try panning / zooming to an area containing data and then try to run your analysis again.

- *Run* - if the combination of options in the *Questions* area's combo boxes will allow you to run a scenario, this button is enabled.

2.3.4 Data conversions when running a scenario

When running a scenario, the data being used needs to be processed into a state where it is acceptible for use by the impact function. In particular it should be noted that:

- Remote datasets will be copied locally before processing.
- All datasets will be clipped to the intersection of the **hazard** layer, the **exposure** layer and the current view extents.
- All clipped datasets will be converted (reprojected) to Geographic (EPSG:4326) coordinate reference system before analysis.

2.4 Keywords System

This document describes the purpose and usage of the InaSAFE *keywords* system.

2.4.1 Purpose

The keywords system is used by the *Impact Functions* to determine the nature of the input layers that have been passed to them.

Each input GIS dataset used by InaSAFE needs to have an accompanying keywords file. The purpose of the keywords file is to provide additional metadata needed by the impact functions. For example, the keywords file will indicate whether a given dataset should be treated as a *hazard* or an *impact* layer. It is also used to indicate the context of the layer (e.g. "it's a *flood* layer, it's an *earthquake* layer).

By convention and expectation, the keywords file should be named with the same base name of the GIS datasource it accompanies. For example a flood dataset saved as:

```
c:\gisdata\flood.tif
```

Would need to have an accompanying keywords file saved as:

```
c:\gisdata\flood.keywords
```

Note: We recommend to **avoid using spaces** in your file names and file paths!

The InaSAFE QGIS plugin provides an editor for these keywords. The purpose of this document is to describe the keywords editor and to provide guidelines as to the use of keywords.

Note: Currently keywords are not validated by the library. This means if you for example misspell a keyword, use the wrong letter case (e.g. upper case instead of lower case) or provide the wrong keyword for the context (e.g. provide a subcategory of flood to an exposure category), the system will not be able to determine what to do with the file. For that reason you should follow the guidelines below carefully to ensure you have entered your keywords correctly.

2.4.2 Guidelines

In this section we lay out the guidelines for keyword usage.

Category

Every dataset should have a category assigned to it. Category should be written in lower case.

Key	Allowed Values
category	hazard
category	exposure

Example keywords file entry:

```
category: hazard
```

Subcategory

The selection of a subcategory value is dependent on the category:

Valid subcategories for category 'hazard':

Key	Allowed Values
subcategory	tsunami
subcategory	flood
subcategory	tephra

Where tephra is volcanic ashfall.

Valid subcategories for category 'exposure':

Key	Allowed Values
exposure	population
exposure	building
exposure	roads

Example keywords file entry:

```
category: hazard
subcategory: flood
```

Units

The units keyword is only valid in the context of 'hazard' layers, and is used to indicate the metric or imperial units represented by each data entity (a grid cell or a vector feature) in the hazard layer. Example keywords file entry:

```
category: hazard
subcategory: flood
units: m
```

In the above case there is a soft constraint to use a value for units of m, feet or wet/dry as defined by the table below because the subcategory is 'flood' . The following are the allowed units which are dependent on the subcategory defined.

Valid subcategories for subcategory 'tsunami' or subcategory 'flood':

Key	Allowed Values
units	m
units	wet/dry
units	feet

In the case where the units are m (meters) or feet, the input dataset should be a raster layer where each cell in the raster represents a depth in the units specified.

In the case of wet/dry, the input dataset needs to be a vector polygon layer. Any area that is inside a polygon is considered 'wet' and any area outside of a polygon is considered to be 'dry'.

Valid subcategories for subcategory 'volcano'

Key	Allowed Values
units	kg2/m2

In this case the dataset should be a raster layer where each cell represents the kilograms per meters squared of ash fall on that cell.

Note: 'units' and 'datatype' (described below) should be considered to be mutually exclusive - i.e. they should not both be present in any given keywords file.

Datatype

The datatype keyword is specific to exposure layers and represents the datatype of people, infrastructure etc. within a given area.

Valid densities for different subcategories

Subcategory	Key	Allowed Values
population	datatype	count
population	datatype	density
building	datatype	osm
building	datatype	sigab
building	datatype	other
roads	datatype	not used for roads

2.4.3 Assumptions

The following assumptions are made about keywords, which may or may not be programmatically enforced by the InaSAFE library and GUI:

- There should only be **one keyword for a given key** in the keywords file
- Switching from hazard to exposure will clear parts of the GUI since in general most keywords are category dependent. In particular, selecting **'hazard'** will remove the **'datatype'** key/value pair, and selecting **'exposure'** will remove the **'units'** key value pair.
- Keywords for **category** are **enforced** to be one of 'hazard' or 'exposure' by the GUI.
- All keywords should be in **lower case**, **without spaces** with the exception of 'Title' whose value may contain both spaces and mixed case letters.
- Values for keywords should generally be lower case, with the exception of **datatype values may be in upper case** (e.g. MMI)
- Keys and values should **not contain colons**. In the keyword editor, any colons will be replaced with a full stop character.

- All other Keywords and values that do not fit the above domain lists may be used but they may produce undesired results.

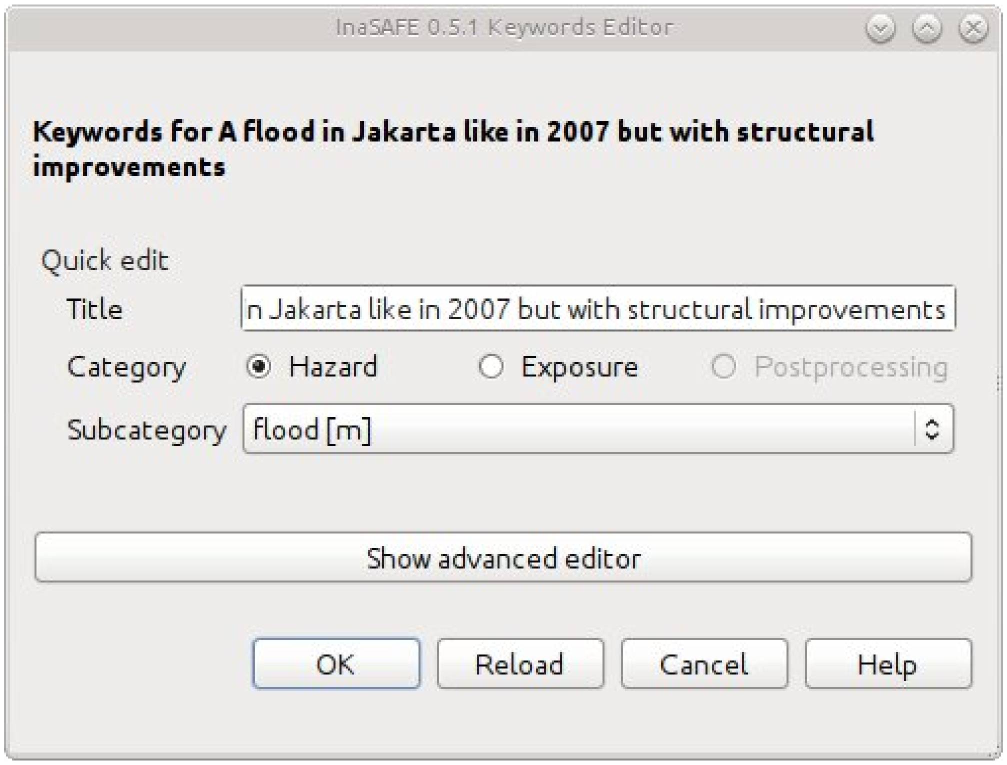
2.4.4 The keywords editor graphical user interface

The graphical user interface for keyword editing is divided into two parts:

1.) **Minimal mode**: In this mode, only following options are provided:

- **Title** - a 'friendly' name for the dataset which will be displayed in reports, the user interface and so on.
- **Category** - A mandatory choice between 'hazard' and 'exposure'.
- **Subcategory** - An amalgamated subcategory/units picklist (in the case of hazard) or amalgamated subcategory/datatype (in the case of exposure). In this case, the secondary characteristic (units or datatype) are shown in square brackets after the subcategory name e.g. `flood [m]` is used for subcategory 'flood', units 'm'.

An example of the keywords editor in minimal mode is shown below.



Keywords for A flood in Jakarta like in 2007 but with structural improvements

Quick edit

Title: n Jakarta like in 2007 but with structural improvements

Category: [✓ Hazard] Exposure Postprocessing

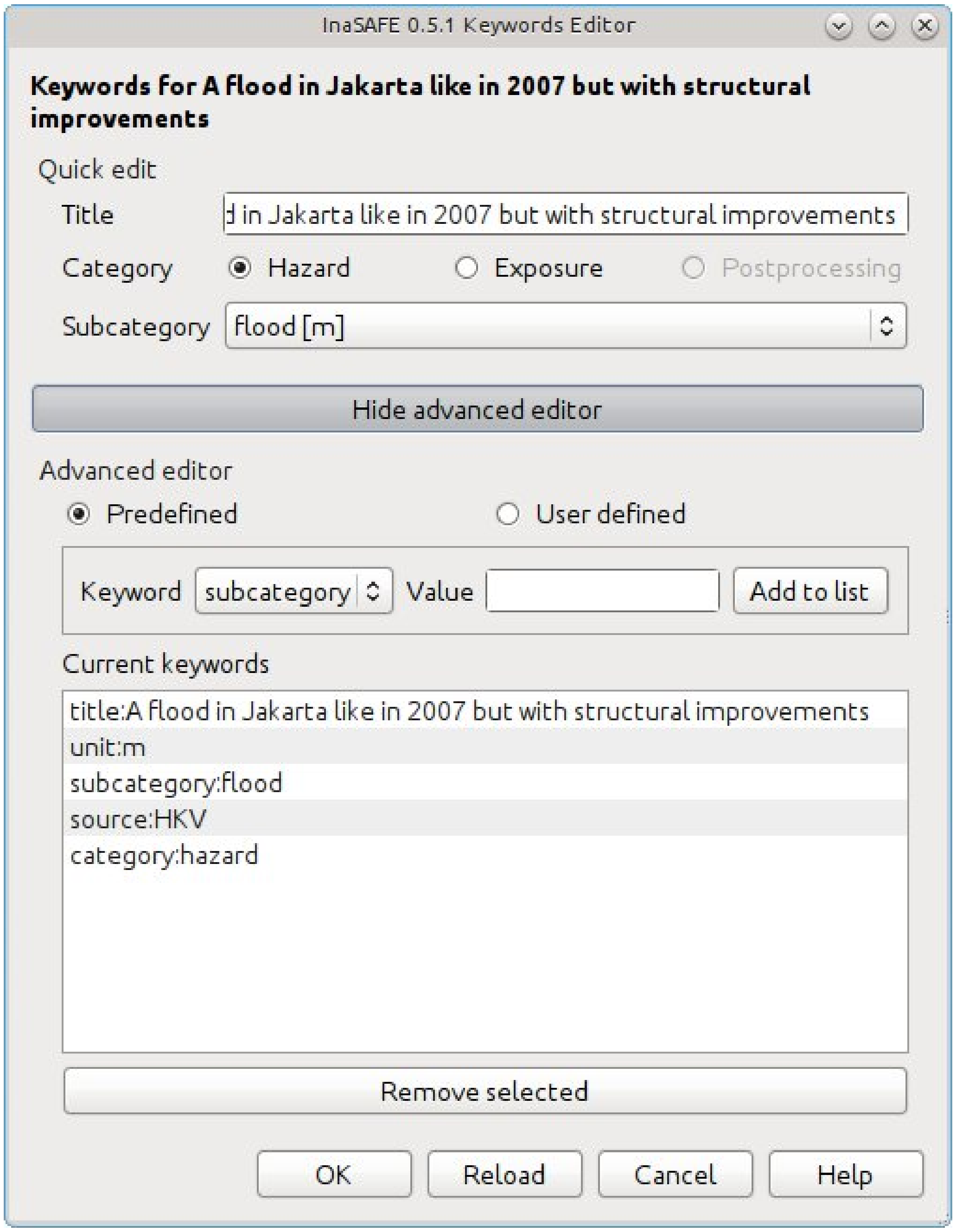
Subcategory: flood [m]

2. **Advanced mode**: In this mode several extra options are provided in addition to the minimal mode options. Unlike minimal mode, in advanced mode only basic validation is performed and the user is given more flexibility to manually define and remove key/value pairs. Three sections are provided for this:

- **Predefined** - In this section, the user selects from a constrained list of keywords, enters a free-form value and then adds the key/value pair to the keywords list (see below).
- **User defined** - In this section, there is no constraint on the keywords entered - any single lower case word will be accepted for both the key and the value components.
- **Current keywords** - In this area a complete list of all the keywords for the dataset are displayed. The keywords list here is updated when any changes are made in both the simple and advanced mode editors. It is also possible in this area to manually remove unwanted keywords using the 'remove selected' button. Multiple keywords can

be removed in a single operation by `Control-clicking` on multiple keyword entries in the current keyword list and then clicking *Remove selected*

An example of the keywords editor in advanced mode is shown below.



Title: d in Jakarta like in 2007 but with structural improvements

Category: [✓ Hazard] Exposure Postprocessing

Subcategory: flood [m]

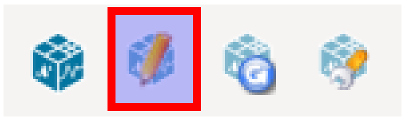
[✓ Predefined] User defined

Keyword: subcategory

title:A flood in Jakarta like in 2007 but with structural improvements
unit:m
subcategory:flood
source:HKV
category:hazard

2.4.5 Invoking the keywords editor

The keyword editor can easily be invoked by selecting any layer in the QGIS layers list, and then using the plugin menu to start the editor (*Plugins* → *InaSAFE* → *Keyword Editor*). Alternatively, you may use the keywords editor icon on the plugins toolbar as illustrated below.



2.4.6 Saving your edits

To save your keyword edits, simply press the *OK* button and the .keywords file will be written to disk.

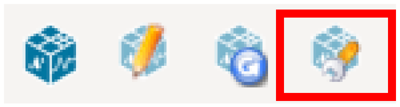
2.4.7 Cancelling your edits

You can cancel your changes at any time by pressing the *Cancel* button. No changes will be written to disk and your .keywords file will remain in its original state.

2.4.8 Keywords for remote and non-file based layers

If you are using a PostgreSQL, WFS, Spatialite or other non-file based resources, you can still create keywords. In these circumstances the keywords will be written to a sqlite database - by default this database is stored as `keywords.db` within the InaSAFE plugin directory root.

You may wish to use a different location for the `keywords.db` keywords database - you can configure this by using the InaSAFE options dialog. The options dialog can be launched by clicking on the InaSAFE plugin toolbar's options icon (as shown below) or by doing *Plugins* → *InaSAFE* → *InaSAFE Options*.



When the options dialog is opened, the keywords database path can be specified using the *keyword cache for remote datasources* option as shown below.

Note: (1) Support for remote and non-file based layers was added in InaSAFE version 0.3. (2) The database can be opened using a sqlite editor such as sqliteman, but the data in the keywords table is not intended to be human readable or edited. The table columns consist of an MD5 hash based on the URI for the datasource (typically the database connection details) and a blob which contains the keywords as a pickled python dictionary.

See the *InaSAFE Options* document for more information about the InaSAFE options dialog.

2.4.9 Sharing your keywords cache

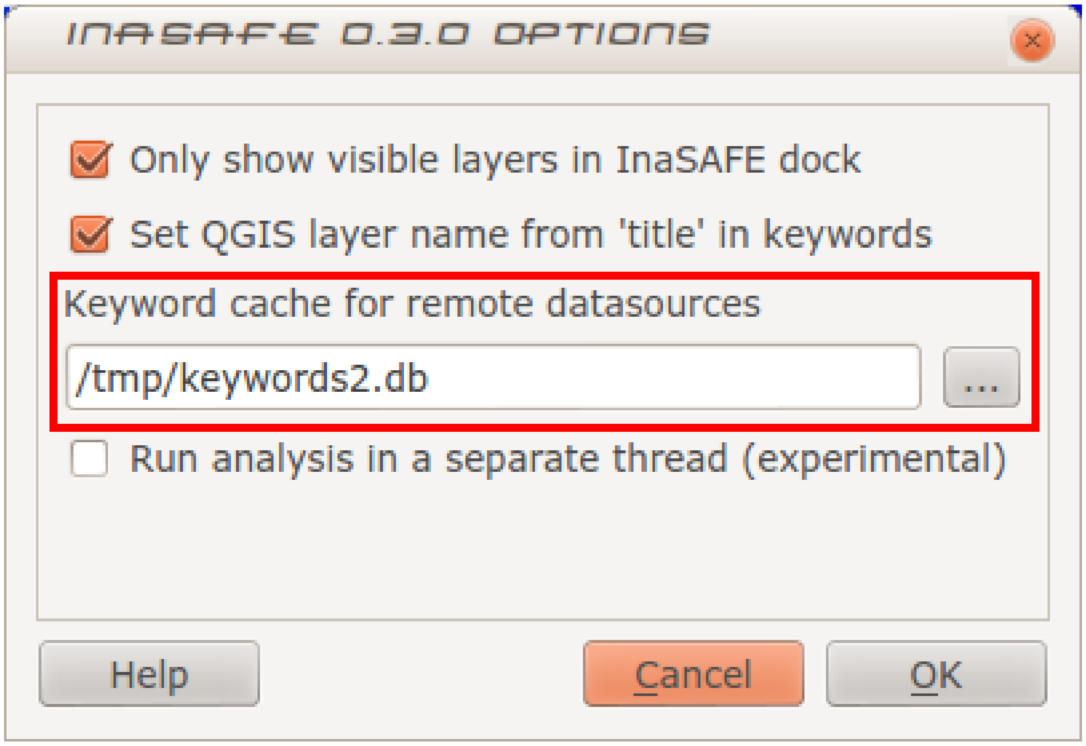
In theory you can place the keywords file on a network share and create a shared keyword repository in a multi-user environment, but you should note that the layer URI hashes need to be identical in order for a layer's keyword to be found. This means that, for (contrived), example:

```
connection=postgresql,user=joe,password=secret,resource=osm_buildings
```

would not be considered the same as:

```
connection=postgresql,user=anne,password=secret,resource=osm_buildings
```

since the user credentials differ, resulting in a different URI. To work around this you could create a common account so that every user will effectively use the same URI to load that layer e.g.:



☑ Only show visible layers in InaSAFE dock
☑ Set QGIS layer name from 'title' in keywords
Keyword cache for remote datasources
/tmp/keywords2.db
☐ Run analysis in a separate thread (experimental)

```
connection=postgresql,user=public,password=secret,resource=osm_buildings
```

For certain resources (e.g. ArcInfo coverages, Spatialite databases) where the keywords cache is also used, you should take care to use a common mount point or network share to access the data if you wish to successfull hit the cache with the layer's URI. For example you could have all users mount your data to the same place. Under Unix like operating systems this could look something like this:

```
/mnt/gisdata/jk.sqlite
```

Under Windows you could always the same drive letter and path the to share e.g.:

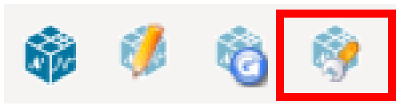
```
Z:\gisdata\jk.sqlite
```

2.4.10 Getting help

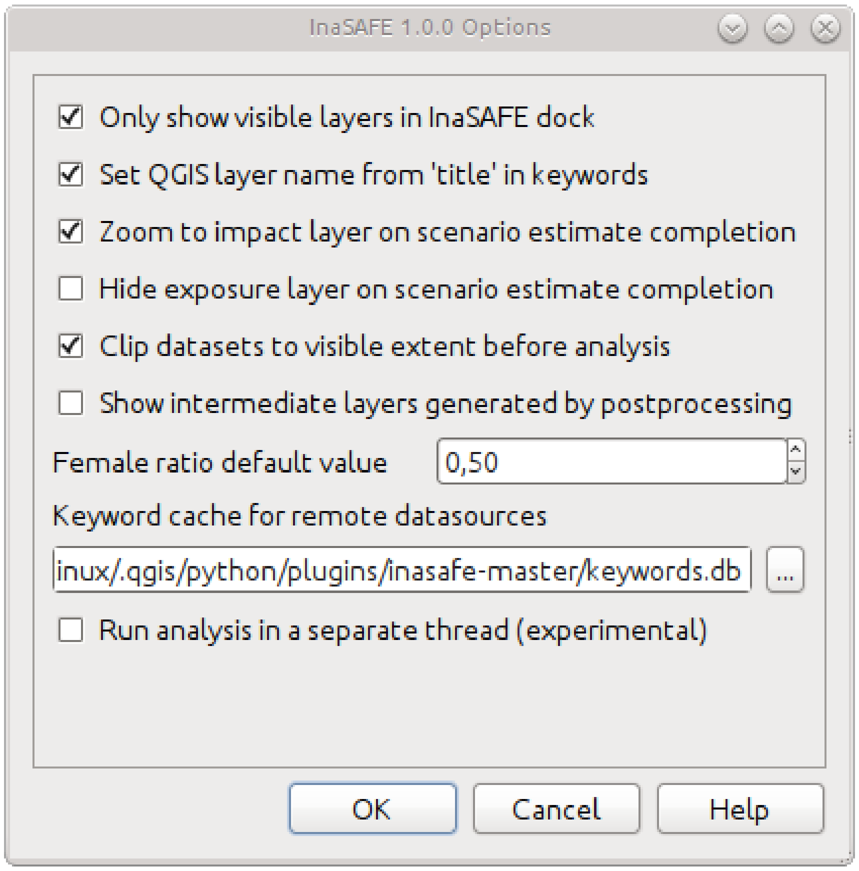
If you need help using the keywords editor, you can click on the *Help* button at the bottom of the dialog and this page will be displayed.

2.5 InaSAFE Options

The InaSAFE plugin provides an options dialog which allows you to define various options relating to how InaSAFE will behave. The options dialog can be launched by clicking on the InaSAFE plugin toolbar's options icon (as shown below) or by doing *Plugins* → *InaSAFE* → *InaSAFE Options*.



Then the dialog will appear, looking something like this:



☑ Only show visible layers in InaSAFE dock
☑ Set QGIS layer name from 'title' in keywords
☑ Zoom to impact layer on scenario estimate completion
☐ Hide exposure layer on scenario estimate completion
☑ Clip datasets to visible extent before analysis
☐ Show intermediate layers generated by postprocessing
Female ratio default value 0,50
Keyword cache for remote datasources
inux/.qgis/python/plugins/inasafe-master/keywords.db
☐ Run analysis in a separate thread (experimental)

Note: You can click on the *Help* button at any time and it will open the help documentation browser to this page.

The following options are available on the *Options Dialog*:

- *Only show visible layers in the InaSAFE dock* : This option will determine whether **all** (when unchecked) hazard and impact layers should be listed in the InaSAFE dock's combo boxes. or (when checked) only visible layers.
- *Set QGIS layer name from 'title' in keywords* : This option will (when enabled) cause QGIS to name layers in the *Layers tree* using the *title* keyword in the layer's keywords file. If the layer has no 'title' in its keywords, or it has no keywords at all, the normal QGIS behaviour for naming layers will apply.
- *Zoom to impact layer on scenario estimate completion* : This option will cause the map view to zoom in/out in order to completely contain the InaSAFE impact scenario map output when an analysis completes.
- *Hide exposure layer on scenario estimate completion* : This option will cause QGIS to turn off the exposure layer used when InaSAFE completes the current analysis. You can re-enable the layer visibility again by checking its checkbox in the legend.
- *Clip datasets to visible extent before analysis* : This option will cause QGIS to clip hazard and exposure layers to the currently visible extent on the map canvas.
- *When clipping, also clip features (e.g. will clip polygon smaller)* : This option will any polygons that extend beyond the analysis extents to be clipped so that they are contained by the analysis extents. For example if you have a flood hazard polygon layer that extends far beyond your area of interest, the flood polygons will be reduced so that only the part inside of your area of interested is retained. This can speed up processing time somewhat.

- *Help to improve InaSAFE by submitting errors to a remote server* : This option, when enabled, will submit diagnostic information back to an InaSAFE project server in the event of any error taking place that we are able to trap. The information provided is useful to the InaSAFE team to improve the robustness of the software we produce and to spot trends in issues people encounter. Please note the warning text below this option which is included here in full: "Note: The above setting requires a QGIS restart to disable / enable. Error messages and diagnostic information will be posted to http://sentry.linfiniti.com/inasafe-desktop/. Some institutions may not allow you to enable this feature - check with your network administrator if unsure. Although the data is submitted anonymously, the information contained in tracebacks may contain file system paths which reveal your identity or other information from your system."
- *Show intermediate layers generated by postprocessing* : This option will cause QGIS to show the intermediate files generated by the postprocessing steps in the map canvas.
- *Default female ratio* : This is the default female to total population ratio used por generating Gender breakdowns
- *Keyword cache for remote datasources* : This option is used to determine where keywords are stored for datasets where it is not possible to write them into a .keywords file. See *Keywords System* for more information on the keywords system.
- *Run analysis in separate thread (experimental)* : This option cause the analysis to be run in its own thread.

Warning: It is not recommended to use the threaded implementation at this time. For this reason it is disabled by default.

Pressing *Cancel* at any time will close the options dialog and any changes made will **not** be applied.

Pressing *Ok* at any time will close the options dialog and any changes made **will** be applied immediately.

Note: The exact button order shown on this dialog may differ depending on your operating system or desktop environment.

2.6 Impact Functions

This document explains the purpose of impact functions and lists the different available impact function and the requirements each has to be used effectively.

Note: This document is still a work in progress.

2.6.1 What is an impact function?

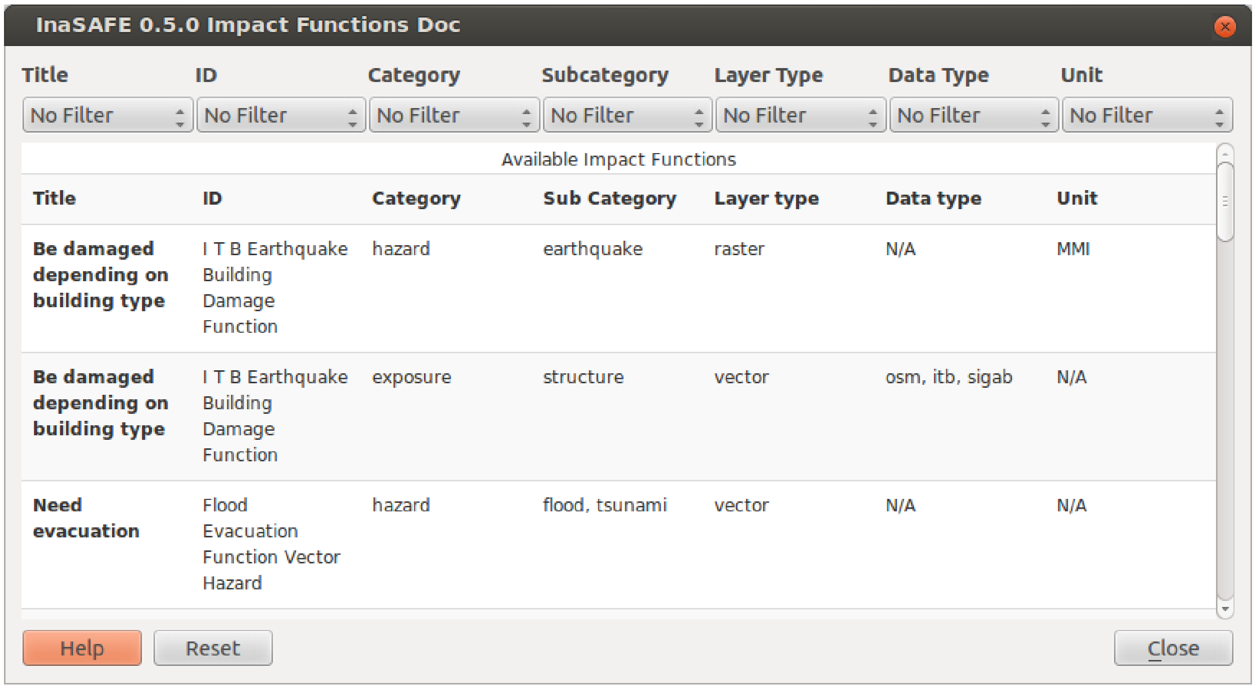
In impact function is a software programme that computes a risk assessment given a number of inputs. The risk assessment will typically have a spatial component (e.g. a GIS layer which can be incorporated into a map) and a non-spatial component (e.g. a list of actions you may want to consider carrying out, or a list of estimates of disaster risk reduction elements such as how many bags of rice to make available).

2.6.2 Selecting an impact function

Impact functions are bundled with the InaSAFE software. The graphical user interface (provided as a plugin for QGIS will offer a list of impact functions that can be used based on the layers you have loaded and their *keywords*.

2.6.3 Exploring impact functions

You can use the impact function table to explore all of the possible impact functions that can be used. They are listed in a table with a series of pick-lists (combo boxes) above which can be used to filter the functions based on different criteria as illustrated below.



Title	ID	Category	Subcategory	Layer Type	Data Type	Unit
No Filter	No Filter	No Filter	No Filter	No Filter	No Filter	No Filter

Available Impact Functions

Title	ID	Category	Sub Category	Layer type	Data type	Unit
Be damaged depending on building type	I T B Earthquake Building Damage Function	hazard	earthquake	raster	N/A	MMI
Be damaged depending on building type	I T B Earthquake Building Damage Function	exposure	structure	vector	osm, itb, sigab	N/A
Need evacuation	Flood Evacuation Function Vector Hazard	hazard	flood, tsunami	vector	N/A	N/A

When applying a filter set, the list of available functions that meet those criteria is updated as shown below.

The impact function table is simply a browser to help you to familiarise yourself with the functions available. For the actual usage of the functions you need to have layers available (i.e. loaded in QGIS) with the appropriate keywords for your target function.

2.6.4 Configurable Impact Functions

Some impact functions can be configured before use. For example if you have a raster flood hazard layer where each pixel represents flood depth, you can set depth thresholds (low / medium / high)

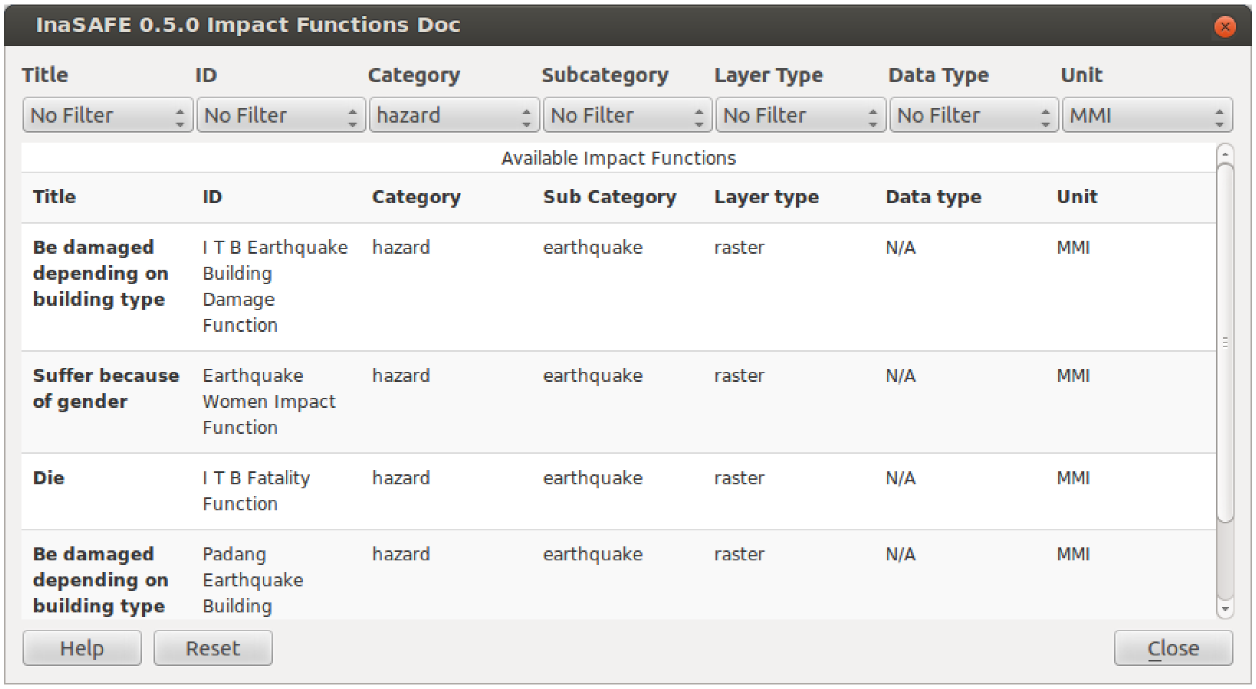
2.6.5 Creating impact functions

If you feel there is an important impact function which is missing, there are two avenues you can follow:

- If you have basic python programming skills, you could implement a new post- processor yourself,
- You can file a ticket on our issue tracking system, and if time and resources allow we will implement it for you.

2.7 Impact Functions Documentation

This document explains the purpose of impact functions and lists the different available impact function and the requirements each has to be used effectively.



Title	ID	Category	Subcategory	Layer Type	Data Type	Unit
No Filter	No Filter	hazard	No Filter	No Filter	No Filter	MMI

Available Impact Functions

Title	ID	Category	Sub Category	Layer type	Data type	Unit
Be damaged depending on building type	I T B Earthquake Building Damage Function	hazard	earthquake	raster	N/A	MMI
Suffer because of gender	Earthquake Women Impact Function	hazard	earthquake	raster	N/A	MMI
Die	I T B Fatality Function	hazard	earthquake	raster	N/A	MMI
Be damaged depending on building type	Padang Earthquake Building	hazard	earthquake	raster	N/A	MMI

2.7.1 Flood Road Impact Function Experimental

Overview

Unique Identifier: Flood Road Impact Function Experimental

Title: Be flooded

Details

No documentation found

2.7.2 Volcano Polygon Hazard Population

Overview

Unique Identifier: Volcano Polygon Hazard Population

Rating: 4

Title: Need evacuation

Author: AIFDR

Details

No documentation found

2.7.3 Flood Evacuation Function Vector Hazard

Overview

Unique Identifier: Flood Evacuation Function Vector Hazard

Rating: 4

Title: Need evacuation

Author: AIFDR

Details

No documentation found

2.7.4 Flood Evacuation Function

Overview

Unique Identifier: Flood Evacuation Function

Rating: 4

Title: Need evacuation

Author: AIFDR

Details

No documentation found

2.7.5 Flood Building Impact Function

Overview

Rating: 0

Limitation: Lorem ipsum limitation

Title: Be flooded

Unique Identifier: Flood Building Impact Function

Permissible Hazard Input: A raster layer where each cell represents flood depth, or a vector polygon layer where each polygon represents an inundated area. The following attributes are recognised (in order): Flooded (True or False), FLOODPRONE (Yes or No) and Affected (True or False).

Author: Ole Nielsen, Kristy van Putten

Actions: Provide details about where critical response areas are

Synopsis: To assess the impacts of inundation on building footprints originating from OpenStreetMap (OSM).

Citations:

- Hutchings, Field & Parks. Assessment of Flood impacts on buildings. Impact. Vol 66(2). 2012

Permissible Exposure Input: vector polygon layer extracted from OSM where each polygon represents the footprint of a building.

Details

This is an area for free form text where adetailed description of the methodology used is given.

2.7.6 Categorised Hazard Population Impact Function

Overview

Unique Identifier: Categorised Hazard Population Impact Function

Rating: 2

Title: Be impacted

Author: AIFDR

Details

No documentation found

2.7.7 I T B Fatality Function

Overview

Unique Identifier: I T B Fatality Function

Rating: 3

Title: Die or be displaced

Author: Hadi Ghasemi

Details

No documentation found

2.7.8 Volcano Building Impact

Overview

Unique Identifier: Volcano Building Impact

Rating: 4

Title: Be affected

Author: AIFDR

Details

No documentation found

2.7.9 Earthquake Building Impact Function

Overview

Unique Identifier: Earthquake Building Impact Function

Title: Be affected

Details

No documentation found

2.7.10 Categorised Hazard Building Impact Function

Overview

Unique Identifier: Categorised Hazard Building Impact Function

Rating: The Rating

Title: Be affected

Author: The Author

Details

No documentation found

2.7.11 Padang Earthquake Building Damage Function

Overview

Unique Identifier: Padang Earthquake Building Damage Function

Title: Be damaged depending on building type

Details

No documentation found

2.8 Post-processors

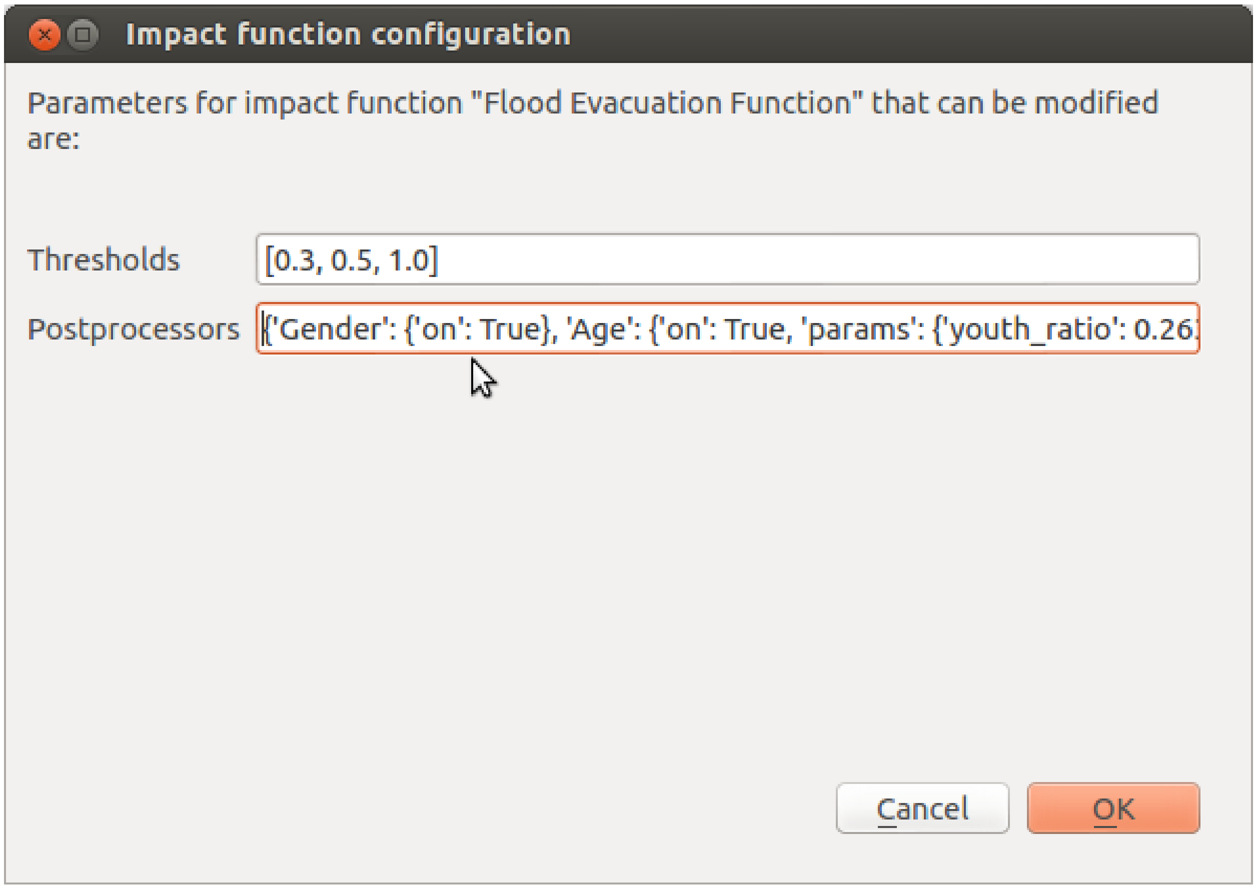
This document explains the purpose of post-processors and lists the different available post-processors and the requirements each has to be used effectively.

2.8.1 What is a post-processor?

A postprocessor is a function that takes the results from the impact function and calculates derivative indicators, for example if you have an affected population total, the **Gender** post-processor will calculate gender specific indicators such as additional nutritional requirements for pregnant women

2.8.2 Selecting a post-processor

Post-processors and their settings can be edited in the user configurable function parameters dialog. To disable a post-processor simply change the *Postprocessors* `"'on': True" to "'on': False"`. The same is valid for any other setting you might encounter there. If you don't see a post-processors field, it means that the impact function you are trying to use does not support any post processor



Parameters for impact function "Flood Evacuation Function" that can be modified are:

Thresholds [0.3, 0.5, 1.0]

Postprocessors {'Gender': {'on': True}, 'Age': {'on': True, 'params': {'youth_ratio': 0.26

Each activated post-processor will create an additional report in the dock and in the printout. If problems arise while post processing, the system will inform you and will skip post-processing.

2.8.3 Creating post-processors

If you feel there is an important post-processor which is missing, there are two avenues you can follow:

- You can develop it yourself or with the aid of a programmer who has a good understanding of the python programming language.
- You can file a ticket on our issue tracking system, and if time and resources allow we will implement it for you.

2.9 Demo projects

InaSAFE comes with bundled demonstration data and a collection of QGIS projects that demonstrate different use cases:

```
.. |project_name| replace:: InaSAFE
```

2.10 Data Types

The impact functions essentially combine spatial data in different formats through a common interpolation library. This essentially allows values from one data set to be assigned to another independent of their types. Given two layers H and E, say, the call:

```
I = assign_hazard_values_to_exposure_data(H, E, ...)
```

will produce a new layer with all values from H transferred to E in a manner appropriate for the data types of H and E. Generally, existing values in E will be also be carried over to I. Conceptually, the new layer I represents the values of H interpolated to E.

The function takes a number of optional keyword arguments that pertain to certain type combinations. They are

- layer_name: Optional name of returned layer. If None (default) the name of the exposure layer is used.
- attribute_name:
  - If hazard layer is of type raster, this is the name for new attribute in the result containing the hazard level. If None (default) the name of hazard is used
  - If hazard layer is of type vector, it is the name of the attribute to transfer from the hazard layer into the result. If None (default) all attributes are transferred.
- mode: Interpolation mode for raster to point interpolation only. Options are 'linear' (default) or 'constant'

The following table shows allowed combinations and what interpolation means in each case.

H	E	Methodology	Return value
Polygon	Point	Clip points to polygon and assign all attributes	Point
Polygon	Line		N/A
Polygon	Polygon		N/A
Polygon	Raster	Convert to points and use Polygon-Point algorithm	Point
Raster	Point	Bilinear or piecewise constant interpolation	Point
Raster	Line		N/A
Raster	Polygon	Convert to centroids and use Raster-Point algorithm	Polygon
Raster	Raster	Check that rasters are aligned and return E	Raster

2.11 Using QGIS

InaSAFE uses Quantum GIS (QGIS) - as the platform on which to provide its functionality. QGIS itself has good documentation and a very helpful user community. In this section we focus on those aspects of QGIS functionality that will be helpful to someone wanting to carry out impact scenario assessments using InaSAFE.

There are three good resources to get you up to speed with using GIS in general and QGIS in particular:

- The Gentle Introduction to GIS by Linfiniti Consulting - it is a free electronic book that introduces the concepts of GIS in a very easy to read manner.
- The QGIS User Manual - also an electronic book that serves as a reference guide to the QGIS Desktop Application.
- The QGIS Training Manual by Linfiniti Consulting that provides a detailed training guide on QGIS and PostGIS.

Things to cover (suggested by Trias Aditya)

- access to statistics data
- access to WFS (for exposure data)
- access to WMS (for background data)
- you have got an idea to access WCS (e.g. hazard, elevation data etc)
- map the projected impact into graphical charts (this will be my favourites then...)

2.12 Getting involved

If you like the work we are doing in InaSAFE and would like to contribute, we will welcome your involvement. We spend a lot of time and effort trying to make a robust, user friendly and useful software platform, but there is no substitute for having interested users participating and sharing their needs, problems and successes! Here are a few simple ways you can get involved:

Visit our web site! The web site will always represent the latest information on how to use InaSAFE. We encourage anyone who wants to get involved with the project to first read the content available on the site to familiarise themselves with the content. The website is available at:

http://inasafe.org

Join our mailing list! If you would like to discuss methodologies, using the software or any issues you encounter, you can join our google group (you do not need a google account to use this forum) by sending an email to:

inasafe-users+subscribe@googlegroups.com

You can unsubscribe again at any time by simply sending an email to:

inasafe-users+unsubscribe@googlegroups.com

Use our issue tracker! We maintain an issue tracker here:

http://github.com/AIFDR/inasafe/issues

On this page you can browse and search existing issues and create new issues. The issue tracker is a great place to let us know about specific bugs you encounter or tell us about new features you would like to see in the software.

Chat live to developers on IRC! Internet Relay Chat (IRC) is a chat room environment where you can talk (by typing messages) to other inasafe users and developers to discuss ideas and get help. You can use your own IRC client and join #inasafe on the irc.freenode.net network. Alternatively, you can use your web browser to join the chat room using the link below:

http://webchat.freenode.net/

On the form that appears, choose a user name, enter #inasafe in the :guilabel:‘ Channels:‘ box and complete the rest of the details in the form. After logging in wait a few moments and you will be take to the #inasafe channel.

Note: Other people in the room may not be actively watching the channel, so just ask your question and leave the chat window open and check back every now and then until you see other chat room members become active.

Submit your code! We would really like to emphasise that InaSAFE is Free and Open Source. That means anyone (or any organisation) can freely modify, adapt and improve the software. We will welcome any contributions to InaSAFE. The easiest way to do this is to fork the InaSAFE code base on GitHub and then send us a pull request.

We also welcome small improvements, translations or other fixes via the issue management system mentioned above.

Note: We have strict requirements that all code submitted to InaSAFE is compliant with high *Coding Standards* and is continually tested by a comprehensive regression testing system. We have this requirement in place to ensure a good experience for our users and to ensure that users can have confidence in the results produced by InaSAFE.

2.13 Frequently Asked Questions

2.13.1 I found a bug, how should I report it?

We manage the project issues using a GitHub issue tracker. The InaSAFE issue tracker is open to everyone, though you will first need to register a (free) account on GitHub to use it. You can find the GitHub self-registration page here.

2.13.2 Do I need to pay to use InaSAFE?

No, the software is completely Free and Open Source.

2.13.3 What license is InaSAFE published under?

InaSAFE is published under the GPL version 2 license, the full text of which is available at www.gnu.org/licenses/gpl-2.0.txt.

Under the terms of the license of you may freely copy, share and modify the software, as long as you make it available under the same license.

2.13.4 How is the project funded?

The project is being developed 'for the good of humanity' and has been jointly developed by BNPB, AusAid & the World Bank.

2.13.5 Could we request a new feature?

If you have a feature request, please use the issue tracker to let us know about it, using the same procedure as for bug reporting.

2.13.6 How do I rename a shape file and all the helper files?

Use the rename command. rename [-v] [-n] [-f] perlexpr [files]. For example:

```
rename -v "s/^building/OSM_building_polygons_20110905/" building.*
```

2.13.7 How do I reproject a spatial data file to WGS84 geographic coordinates

For raster data, use gdalwarp, for example:

```
gdalwarp -t_srs EPSG:4326 <source>.tif <target>.tif
```

For vector data use ogr2ogr. For example from TM-3 zone 48.2:

```
ogr2ogr -s_srs EPSG:23834 -t_srs EPSG:4326 <target>.shp <source>.shp
```

2.13.8 How do I get Open Street Map building data into InaSAFE?

For Indonesia, you can download latest collections at data.kompetisiosm.org. or you can add our Open Street Map building PostGIS mirror to InaSAFE:

- Add PostGIS layer with host=203.77.224.77, database=osm, username=aifdr, port 5432, SSL mode=disable
- Select view named vw_planet_osm_polygon
- Build query: upper(geometrytype("way")) IN ('POLYGON','MULTIPOLYGON') AND BUILDING != ''

Another way, you can export osm data from HOT Exports:

- Go to Preset JOSM.
- Find and download *Building* preset created by Kate Chapman and save it
- Go to HOT Exports website www.hot-export.geofabrik.de.
- Go to *New Job* menu in the upper right of the page
- Select region, currently only 3 regions are supported by HOT Export (Haiti, Indonesia, Africa)
- Fill *Job Name* and *Description*
- Select area that you want to export by zooming or panning the map
- You can choose smaller area by clicking *Select Smaller Area* and creating rectangle in the map or filling min/max longitude/latitude value for it
- Click *Save* if your map is ready
- Upload JOSM Preset File which have been downloaded before, and click *Save*
- Your job is created and you have to wait until it finish. It'll take some minutes if your map is big one
- When the job is finished, there will be a table contains files that can be downloaded
- Download the *ESRI Shapefile (zipped)*
- Extract it on your computer, and the data will be ready to use

2.13.9 How do I take screen capture e.g. for use in a presentation?

On Ubuntu, get the packages gtk-recordmydesktop and mencoder Record using recordmydesktop (start and stop icon in the top bar) Convert to other formats using mencoder, e.g:

```
mencoder -idx yogya_analysis-6.ogv -ovc lavc -oac lavc -lavcopts \
vcodec=mpeg4:vpass=1 -of lavf -o yogya_analysis.avi
```

or:

```
mencoder -idx yogya_analysis-6.ogv -ovc lavc -oac lavc -lavcopts \
vcodec=wmv2 -of lavf -o yogya_analysis.wmv
```

2.13.10 How do I convert a vector hazard layer to a raster layer?

For vector to raster conversion, use gdal_rasterize utility, for example:

```
gdal_rasterize -a <attribute_name> -l <source>.shp <destination>.tif
```

2.13.11 Why does the plugin not show up in my QGIS Plugin Manager?

One common issue is that if you upgraded from QGIS 1.7.x to 1.8 you may not get the new plugin repo added to your repo list. To fix this you can do:

- open QGIS
- Go Plugins -> Fetch Python Plugins
- click 'Repositories' tab
- click add
- *Name*: Official QGIS Repository
- *Url*: http://plugins.qgis.org/plugins/plugins.xml
- Save it and the plugin repo list should update. If it doesn't, close and open QGIS to force an update.
- In the python plugin manager main tab now you should find InaSAFE available

2.14 Known Issues

2.14.1 Population density grids must be in geographic coordinates

Currently, if population density grids are projected they won't be scaled correctly if resampled. See also ticket #123

2.14.2 Help documentation has no style under windows

Currently if you open the help documentation for InaSAFE, not styling is applied and images may not show, so it looks rather plain. We are working to fix this in a future release. See also ticket #163

2.15 Whats new?

2.15.1 Changelog for version 1.0.1

- Fix https://github.com/AIFDR/inasafe/issues/374
- Fix https://github.com/AIFDR/inasafe/issues/375

2.15.2 Changelog for version 1.0.0

- Added post processor infrastructure including gender and age specific indicators
- Added data source attribution
- Various GUI updates
- Added use of transparency in generated maps
- Added an earthquake impact function
- Documentation updates
- Many bugfixes and architectural improvements
- Better internationalisation support

2.15.3 Changelog for version 0.5.2

- This is a bugfix update to address some minor translation issues in the InaSAFE package.

2.15.4 Changelog for version 0.5.1

- This is a bugfix update to reduce the size of the InaSAFE package.

2.15.5 Changelog for version 0.5.0

- Better documentation. See http://inasafe.org/contents.html
- Time stamp and other metadata added to generated map PDF.
- Initial support for parameterisation of impact functions.
- Updated logging infrastructure including support for logging to the QGIS log panel.
- Fixed missing InaSAFE icon in QGIS plugin manager.
- Fixes for help system under windows.
- Multi-page support for generated report PDF (which is now created as a separate document).
- Ability to combine polygon hazard (such as flood prone areas) with population density.
- Option to use entire intersection of hazard and exposure instead of clipping to the somewhat arbitrary viewport (the training revealed that this was a bit confusing)
- Aggregation of raster impact layers by arbitrary polygon layers (such as kelurahan boundaries)
- Limited support for runtime configuration of impact functions (e.g. by changing thresholds). This is an interim measure while the team is working on a GUI to manipulate impact functions more generally.
- More DRR actions added to impact function reports (such as how will warnings be disseminated, how will we reach stranded people etc.)
- Volcanic (zonal hazard) impact assessments on building and population
- New function table view that lists all the available impact functions and allows them to be filtered by different criteria.
- Lots of small improvements to error reporting, GUI, translations and code quality.

Changelog for version 0.4.1

- This is a minor bugfix release with packaging and documentation related changes only so that InaSAFE can be deployed via the official QGIS repository.
- Added InaSAFE tutorial to sphinx documentation

Changelog for version 0.4.0

- Ability to automatically handle multipart vector data: https://github.com/AIFDR/inasafe/issues/160
- Better error reporting:
- https://github.com/AIFDR/inasafe/issues/170
- https://github.com/AIFDR/inasafe/issues/161
- https://github.com/AIFDR/inasafe/issues/157
- Bug fixing:
- https://github.com/AIFDR/inasafe/issues/159
- https://github.com/AIFDR/inasafe/issues/156
- https://github.com/AIFDR/inasafe/issues/173
- https://github.com/AIFDR/inasafe/issues/166
- https://github.com/AIFDR/inasafe/issues/162
- InaSAFE APIs better defined: https://github.com/AIFDR/inasafe/issues/134
- Release procedure developed: https://github.com/AIFDR/inasafe/issues/109
- Added estimate of displaced people to earthquake fatality model: https://github.com/AIFDR/inasafe/commit/04f0e1d
- Achieved 100% translation for Bahasa Indonesia
- Made bundled test and demo data public with associated license information
- Added AusAid and World Bank logos to dock
- Fixed bug with flood population evacuation reporting units

Changelog for version 0.3.0

- Documentation updates - extended guides for using the InaSAFE dock and keyword editors.
- Support for remote layers in keywords editor and scenario modelling
- Added options dialog
- Support for using all layers in hazard and exposure combos, not just visible ones (configurable in options dialog)
- Support for displaying keywords title in QGIS layer list (configurable in options dialog)
- When selecting a hazard or exposure layer, its keywords are now displayed in the results area.
- Performance improvements when toggling layer visibility and adding and removing layers.
- Support for QGIS 1.8 when it is released
- Numerous other ‘under the hood’ bug fixes and improvements
- Migrated code base from RIAB to InaSAFE and restructured the code base

- Added additional tests

Changelog for version 0.2.1:

- Correct translation of 'run' in indonesian. Closes #128
- Updated so that version number is shown in dock
- Removed generated file from polygon test
- Removed the -dev designation from branch releases
- Fix indent error causing noise to show in qgis plugin manager
- Fixed typo - BNPD to BNPB
- Fixed bug where close button does not dispose of the help dialog
- Fixed an issue that prevented the use of earthquake functions when using keywords with lowercase mmi. Closes #142
- Fix for mac clipping issues - the plugin should work on OSX now. Closes #141. Note that OSX users should upgrade to GDAL 1.9 available here: http://www.kyngchaos.com/software/qgis

Changelog for version 0.2.1:

- Map printing support
- Improved translation support and Indonesian translation updates
- Rebranded from Risk in a Box to InaSAFE
- Documentation updates and documented windows developer procedures
- Support for generating documentation and running tests under Windows
- Scripts for semi-automatic packaging of a release
- Improvements to Impact calculator algorithms

Changelog for version 0.1.0:

- First QGIS plugin implementation of InaSAFE.
- Migrated calculation engine from Risiko project.
- Implemented support for polygon hazard layers.
- Added dock widget for designing and executing a scenario model.
- Added the keyword editor for assigning metadata to input files.
- Added integrated context help tool.
- Removed django specific dependencies from the InaSAFE libs.
- removed dependency on SciPy
- Support for internationalisation.
- Comprehensive documentation system.

CHAPTER

THREE

INASAFE DEVELOPER DOCUMENTATION

Additional useful links for developers:

- GitHub site for InaSAFE : https://github.com/AIFDR/inasafe/
- Continuous Integration testbed for InaSAFE : http://jenkins.linfiniti.com/job/InaSAFE/
- QGIS Plugin Home Page : http://plugins.qgis.org/plugins/inasafe/
- InaSAFE Ohloh Project Stats : https://www.ohloh.net/p/inasafe

Developer Documentation Contents:

3.1 Development under Gnu/Linux

Risk-in-a-box is built in python and runs as a plugin in QGIS.

3.1.1 Quick Installation Guide - Linux (Debian based)

These instructions are for setting up a development version on a Debian based linux system such as Ubuntu or Mint.

1. Goto the area where you do development, e.g cd ~/sandbox
2. wget bit.ly/inasafe-install
3. source ./inasafe-install

To verify that the installation works you can run the test suite from the command line:

```
cd inasafe-dev
make test
```

This will run all the regression tests and also highlight any code issues. Note that first time the tests are run they will pull 250MB of test data from our git repository. See further notes on running tests below.

To run the plugin start QGIS and enable it from the *Plugins* → *Manage Plugins* menu.

If this doesn't work see section towards the end of this document about dependencies and try to do a manual install.

3.1.2 Manual Installation Guide - Linux (Debian based)

Dependencies

The Graphical User Interface components are built using PyQt4 and the QGIS plugin API (useful resources: the QGIS Python Cookbook and the QGIS C++ API documentation). As such it is helpful if you are familiar with these technologies (python, Qt4, PyQt4, QGIS). In addition, the following are needed on your machine in order to work effectively with the code base:

- git
- rsync
- pep8
- nosetests (with coverage plugin)
- python-numpy (for numerical computations)
- python-gdal (python bindings to underlying gis functionality)
- python-sphinx (compilation of documents)
- cloud-sptheme (sphinx theme)
- pyqt4-dev-tools (compiling ui and resources)
- qt4-doc (qt4 API documentation)
- pyflakes (test for bad coding style like unused imports / vars)
- python-nosexcover and python-coverage (code coverage reporting)

On an ubuntu system you can install these requirements using apt:

```
sudo apt-get install git rsync pep8 python-nose python-coverage \
python-gdal python-numpy python-sphinx pyqt4-dev-tools pyflakes

sudo pip install cloud-sptheme python-nosexcover
```

In some cases these dependencies may already be on your system via installation process you followed for QGIS.

Cloning the source code from git

To develop on the plugin, you first need to copy it to your local system. If you are a developer, the simplest way to do that is go to *~/.qgis/python/plugins* and clone inasafe from our GitHub repository page like this:

```
git clone git://github.com/AIFDR/inasafe.git  (for read only)
git clone git@github.com:AIFDR/inasafe.git    (to commit changes)
```

QGIS installed in a non-standard location

For running unit tests that need QGIS, you may need to adjust *PYTHONPATH* and *QGIS_PREFIX_PATH* if QGIS is running in a non standard location. For example with QGIS built from source into /usr/local (and python bindings global install option disabled), you could run these commands (or add them to your ~/.bashrc):

```
export QGIS_PREFIX_PATH=/usr/local
export PYTHONPATH=$PYTHONPATH:/usr/local/share/qgis/python/
```

Note: The above can be set within Eclipse's project properties if you are running your tests using the PyDev IDE environment.

Adding inasafe to your python path:

Lastly, you should add the inasafe plugin folder to your PYTHONPATH so that package and module paths can be resolved correctly. E.g:

```
export PYTHONPATH=$PYTHONPATH:${HOME}/.qgis/python/plugins/inasafe
```

Once again you could add this to your .bashrc or set it in Eclipse for convenience if needed.

Running tests

You can run all tests (which includes code coverage reports and other diagnostics) by doing this within the inasafe plugin folder:

```
make test
```

You can also run individual tests using nose. For example to run the riabclipper test you would do:

```
nosetests -v gui.test_riabclipper
```

Achievements

Note: This is optional and thus not hard coded into the makefile.

Optionally you can enable nose achievments which is a motivational tool that gives you little achievement awards based on your test results:

```
sudo pip install git+git://github.com/exogen/nose-achievements.git
```

Now create this file in the root of your inasafe git checkout `setup.cfg`:

```
[nosetests]
with-achievements=1
```

When you run tests occasionally achievements will be displayed to you at the end of the test run. See the achievements home page at http://exogen.github.com/nose-achievements/.

3.1.3 Developing using Eclipse (Linux)

Note: This is optional - you can use any environment you like for editing python, or even a simple text editor.

If you wish to use an IDE for development, please refer to this article for detailed information on how to get the basic Eclipse with PyDev setup.

Creating a project

The procedure for doing this is to do: *File* → *New* → *Project...* and then from the resulting dialog do *PyDev* → *PyDev Project*.

In the resulting project dialog, set the following details:

- *Project name:* : `inasafe`
- *Use default* : `uncheck`
- :guilabel (linux):*Directory* : `/home/<your user name/.qgis/python/plugins/inasafe/`
- :guilabel (windows):*Directory* : `/home/<your user name/.qgis/python/plugins/inasafe/`
- *Choose project type* : `Python`
- *Grammar Version* : `2.7`
- *Add project directory to PYTHONPATH?* : `check`

At this point you should should click the link entitled 'Please configure an interpreter in related preferences before continuing.' And on the resulting dialog do:

- *Python Interpreters: New...* : `click this button`

In the dialog that appears do:

- *Interpreter Name* : `System Python 2.7`
- *Interpreter Executable* : `/usr/bin/python`
- *OK Button* : `click this button`

Another dialog will appear. Tick the first entry in the list that points to your:

```
~/.eclipse/org.eclipse.platform_3.7.0_155965261/plugins/org.python.pydev_2.3.0.2011121518/
```

(or simply click the 'Select All' button)

- *OK Button* : `click this button`

You will be returned to the Python Interpreters list and should see an entry for System Python 2.7 listed there. Now do in the *Libraries* tab:

- *Finish* : `click this button`

Remote Debugging with Eclipse

For remote debugging, you should add pydevd to your PYTHONPATH before starting *QGIS* for example (you will need to adjust these paths to match your system):

```
export PYTHONPATH=$PYTHONPATH:/home/timlinux/.eclipse/org.eclipse.platform_3.7.0_155965261/plugins/or
```

Note: If you are running with remote debugging enabled, be sure to start the PyDev debug server first before launching the Risk-in-a-box QGIS plugin otherwise QGIS will likely crash when it can't find the debug server.

You will need to ensure that the PYTHONPATH containing your pydev package folder is set before you launch QGIS - for example by adding the above line to your ~/.bashrc or by making a small batch file containing the above export and then sourcing the file before launching QGIS e.g.:

```
source riab_paths.sh
/usr/local/bin/qgis
```

Running Unit tests from the IDE

Python has very good integrated support for unit testing. The first thing you should do after setting up the IDE project is to run the tests. You can run tests in the following ways:

- For the entire inasafe package
- For individual sub packages (e.g. engine, gui, storage, impact_functions)
- for an individual test module within a package
- for an class within a test module
- for an individual method within a test class

You can view these individual entities by browsing and expanding nodes in the project panel in the left of the IDE.

Note: If you run the test suite for the entire inasafe package, it will mistakenly treat the sphinx documentation conf.py (docs.source.conf) as a test and fail for that test. This is 'normal' and can be ignored.

3.2 Development under MS Windows

In this document we will walk you through the different activities you will need to do as a windows developer wishing to work on the InaSAFE codebase.

3.2.1 Installation of version control tools

Setup msysgit

To check out the code for development, you first need to install a git client. We cover msysgit here, but you can also use tortoisegit if you prefer (although the tortoise git procedure is not covered here).

To install msysgit (which is a command line git client), download the latest version of the software from the msysgit web site. There is no need to get the 'full install' - just fetching the latest 'preview' is good enough. For example at the time of writing I downloaded `Git-1.7.9-preview20120201.exe`. The download is around 14mb in size.

Once the file is downloaded, run it and respond to the installer prompts as illustrated below:

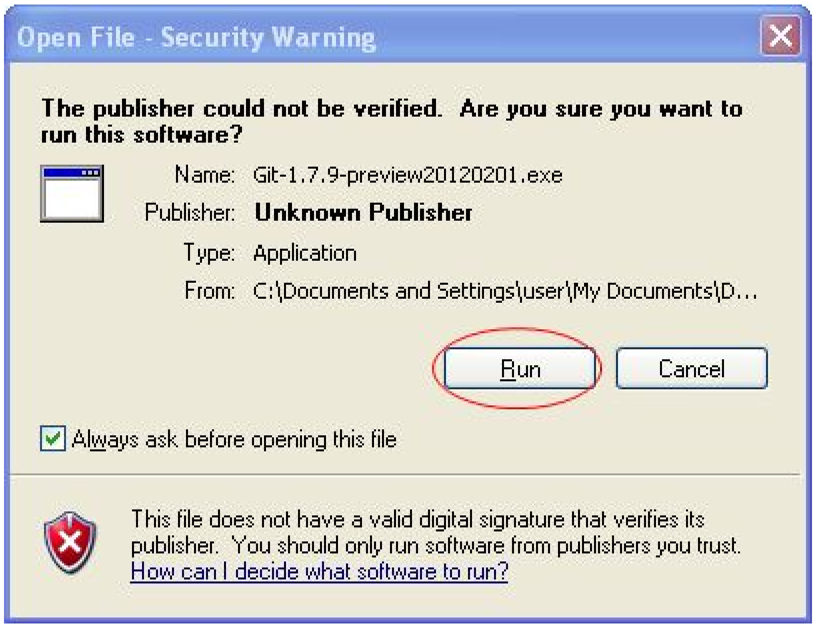
3.2.2 Check out the code and the test data

In this section we actually check out the source code and the test data using the tools we installed above.

Clone the code repository

First open a GIT bash prompt as illustrated below:

The repository can now be cloned by issuing the commands listed below.:



The publisher could not be verified. Are you sure you want to run this software?

Name: Git-1.7.9-preview20120201.exe

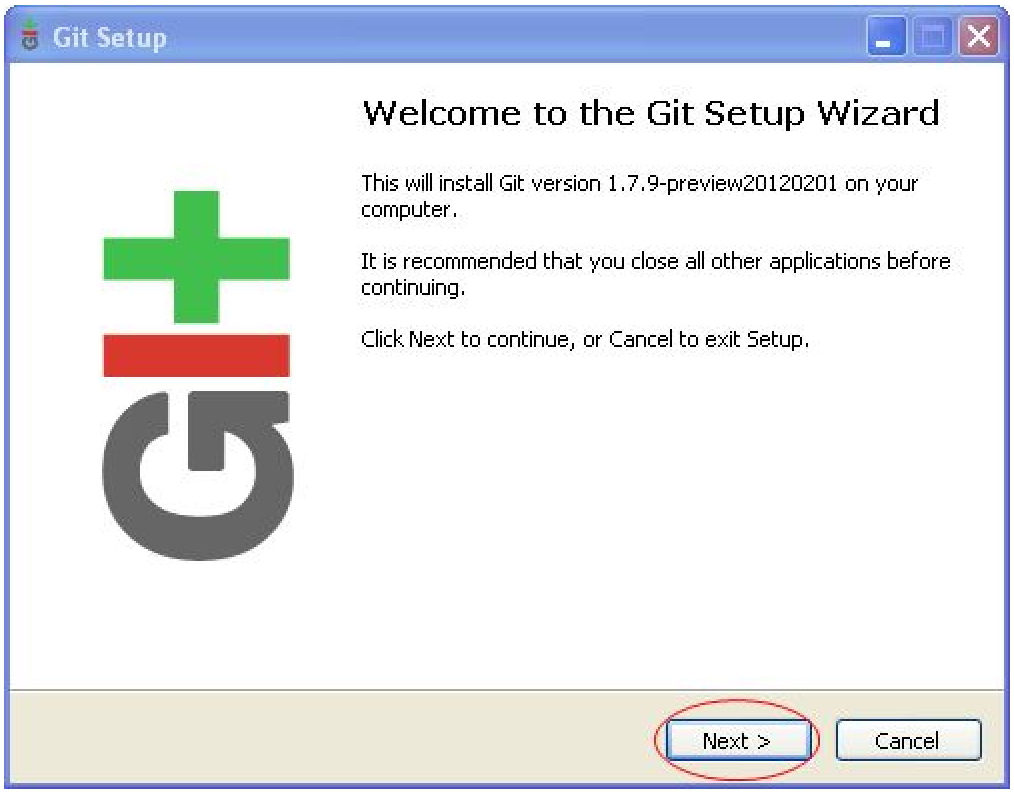
Publisher: Unknown Publisher

Type: Application

From: C:\Documents and Settings\user\My Documents\D...

[✓] Always ask before opening this file

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust. How can I decide what software to run?

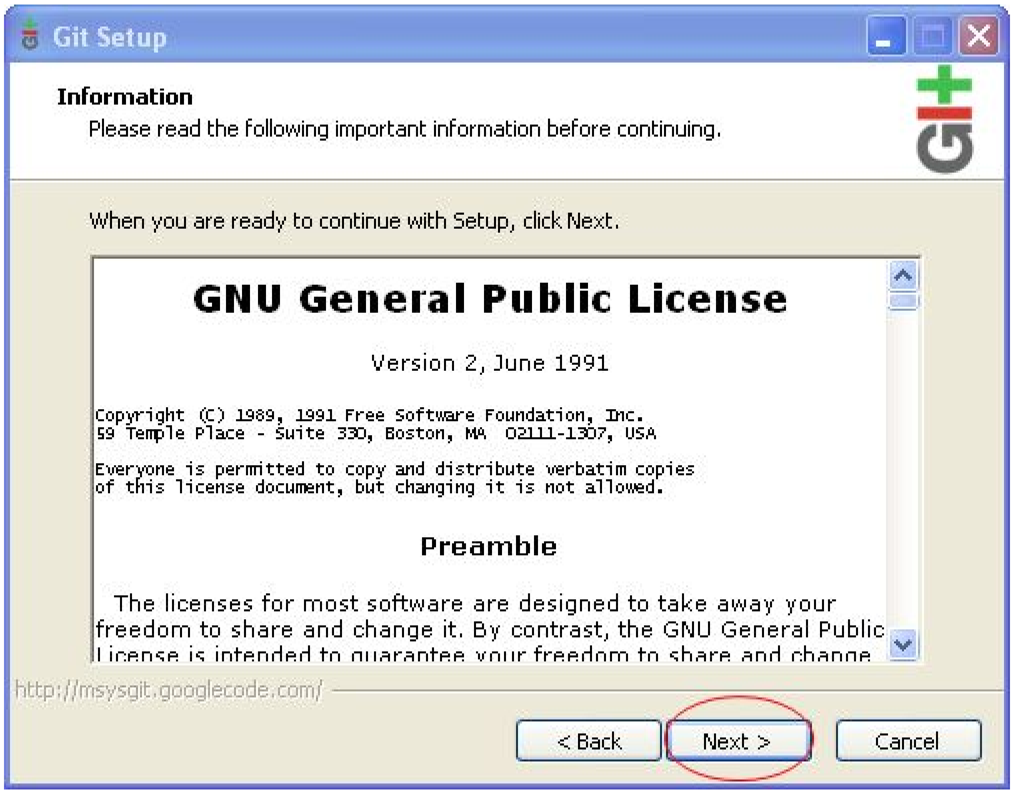


Welcome to the Git Setup Wizard

This will install Git version 1.7.9-preview20120201 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.



Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

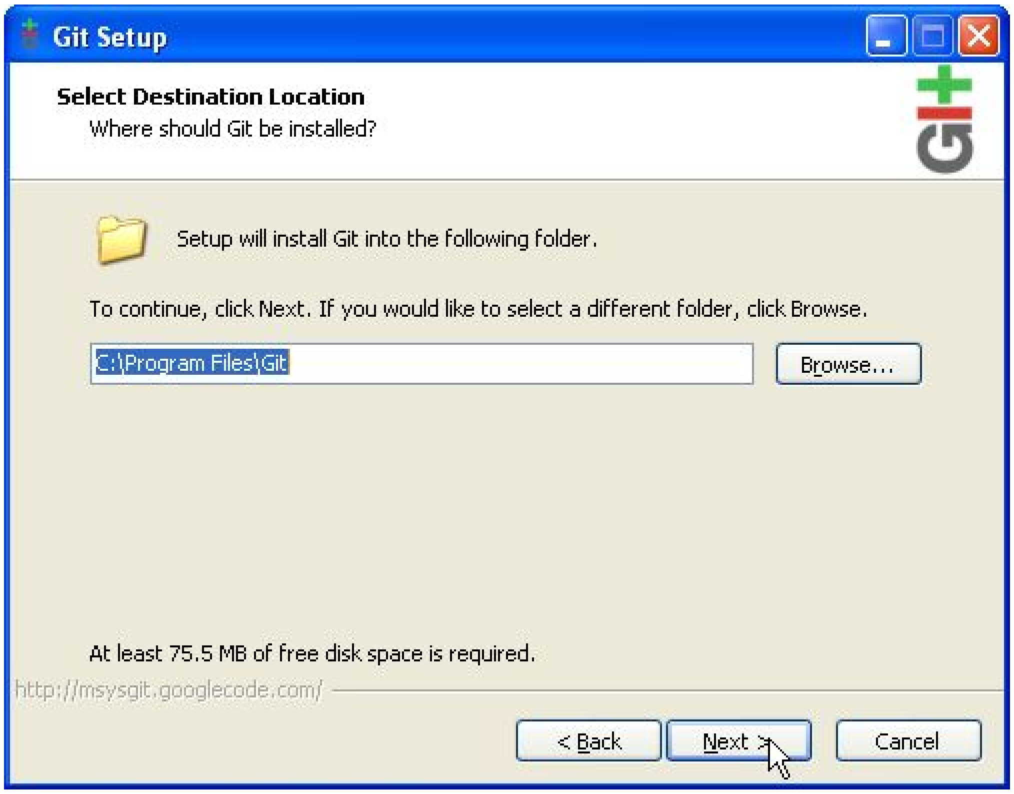
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

http://msysgit.googlecode.com/



Select Destination Location

Where should Git be installed?

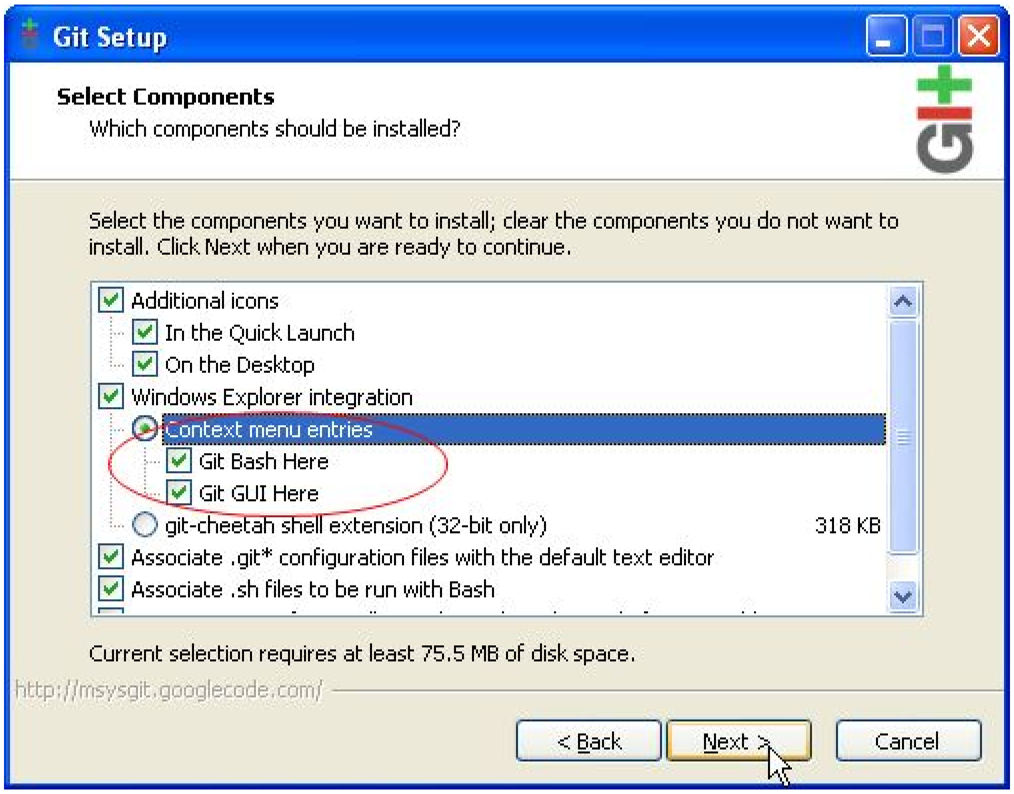
Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 75.5 MB of free disk space is required.

http://msysgit.googlecode.com/


Git Setup

Select Components

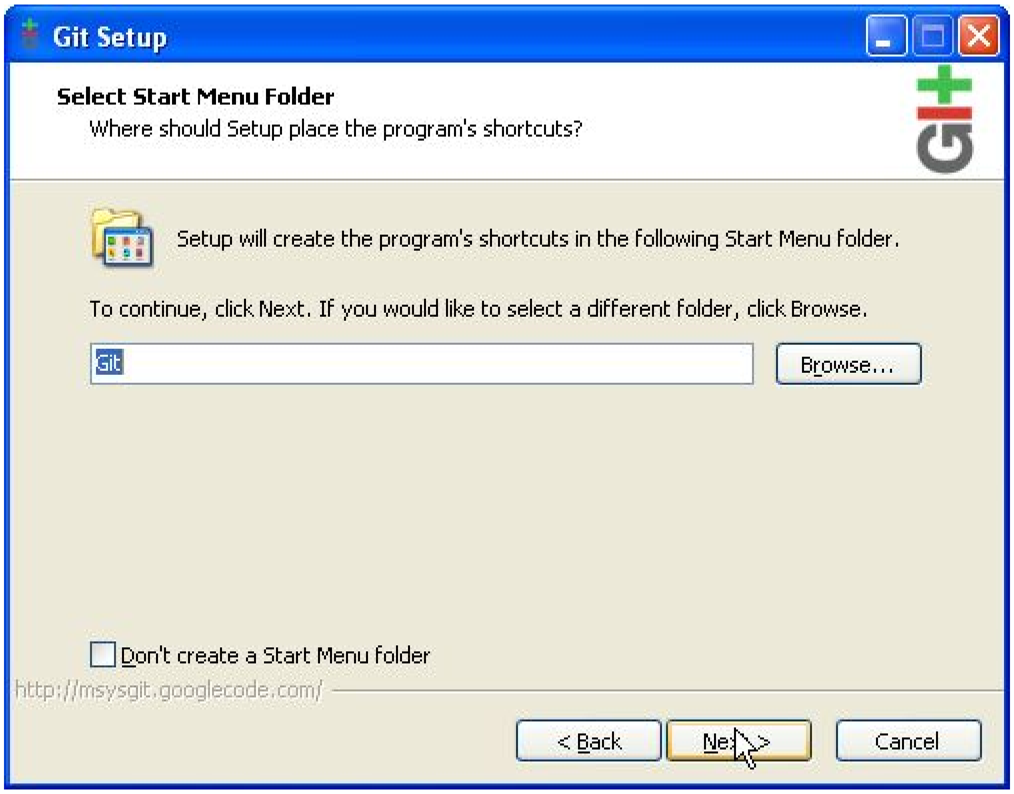
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ✓ Additional icons
  - ✓ In the Quick Launch
  - ✓ On the Desktop
- ✓ Windows Explorer integration
  - ● Context menu entries
    - ✓ Git Bash Here
    - ✓ Git GUI Here
  - ○ git-cheetah shell extension (32-bit only) 318 KB
- ✓ Associate .git* configuration files with the default text editor
- ✓ Associate .sh files to be run with Bash

Current selection requires at least 75.5 MB of disk space.

http://msysgit.googlecode.com/


Git Setup

Select Start Menu Folder

Where should Setup place the program's shortcuts?

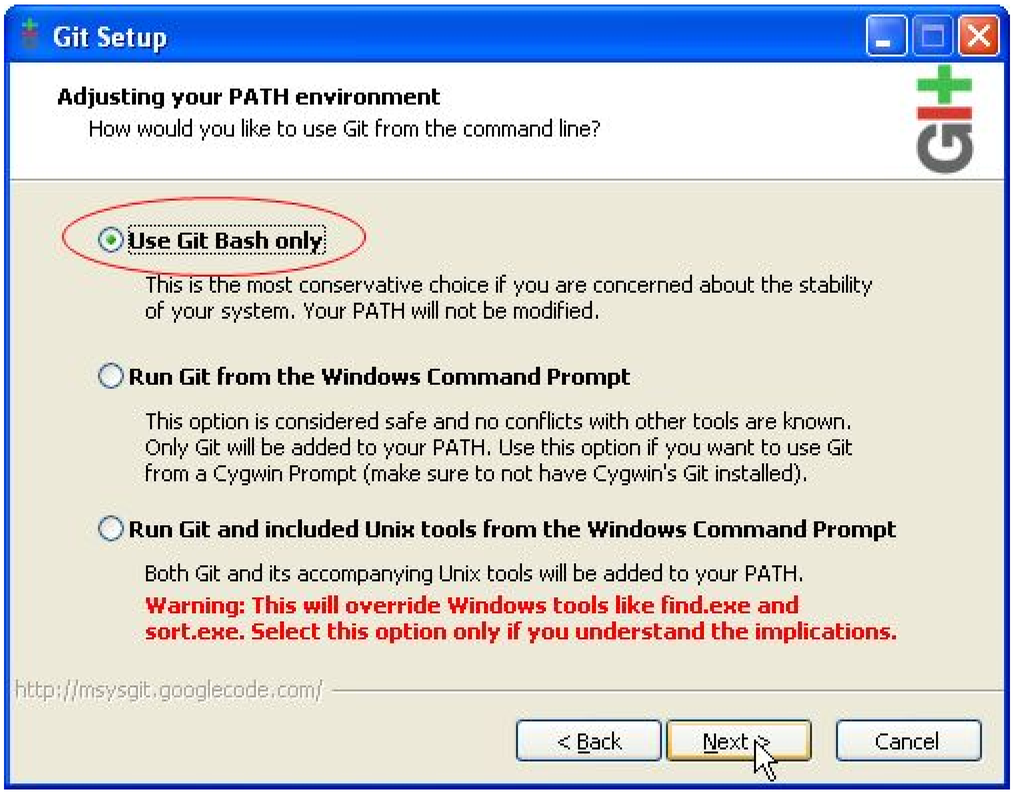
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Git

☐ Don't create a Start Menu folder

http://msysgit.googlecode.com/


Git Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

[✓ Use Git Bash only]

This is the most conservative choice if you are concerned about the stability of your system. Your PATH will not be modified.

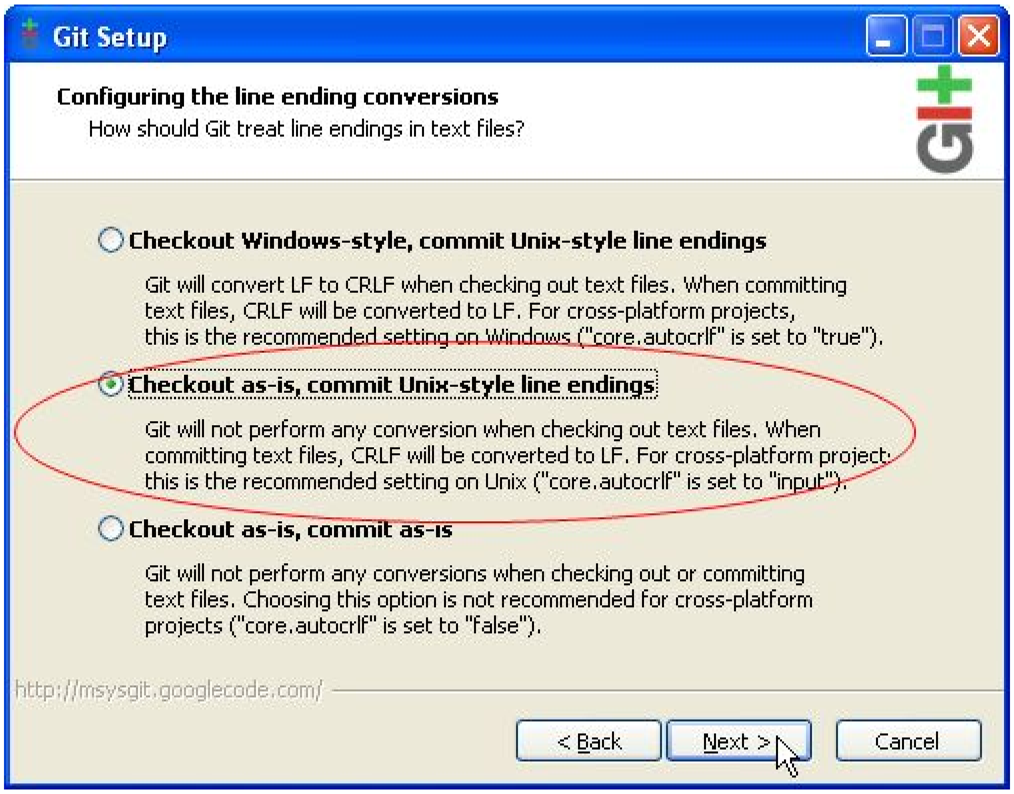
Run Git from the Windows Command Prompt

This option is considered safe and no conflicts with other tools are known. Only Git will be added to your PATH. Use this option if you want to use Git from a Cygwin Prompt (make sure to not have Cygwin's Git installed).

Run Git and included Unix tools from the Windows Command Prompt

Both Git and its accompanying Unix tools will be added to your PATH.

Warning: This will override Windows tools like find.exe and sort.exe. Select this option only if you understand the implications.


Git Setup

Configuring the line ending conversions

How should Git treat line endings in text files?

Checkout Windows-style, commit Unix-style line endings

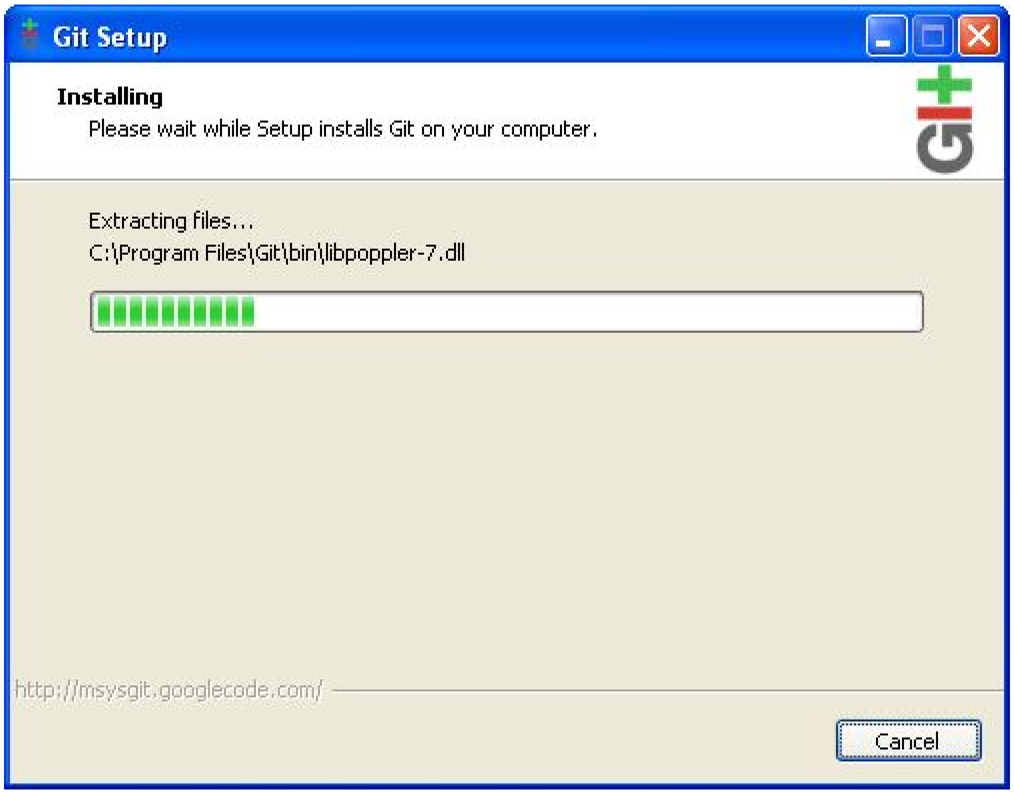
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

[✓ Checkout as-is, commit Unix-style line endings]

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

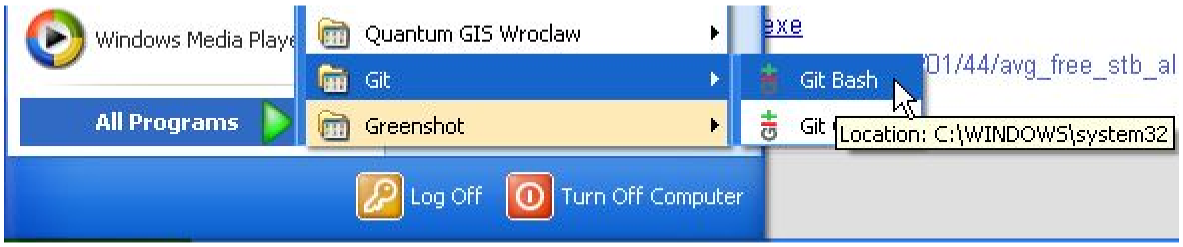
Checkout as-is, commit as-is

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").



Extracting files...
C:\Program Files\Git\bin\libpoppler-7.dll

http://msysgit.googlecode.com/



Location: C:\WINDOWS\system32

```
cd  /c/Documents\ and\ Settings/<your username>/

mkdir -p .qgis/python/plugins

cd .qgis/python/plugins/

git clone https://<your username>@github.com/AIFDR/inasafe.git inasafe-dev
```

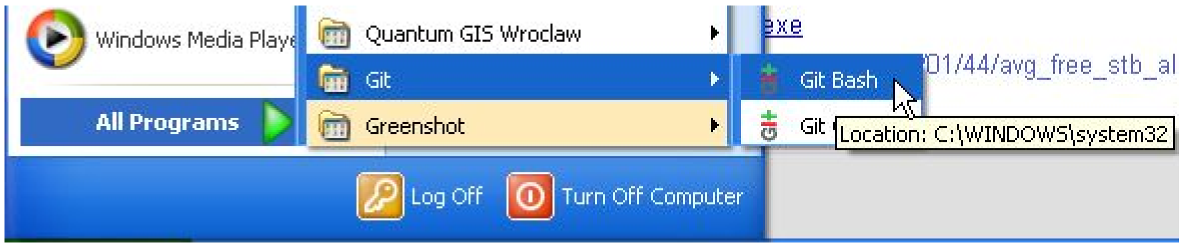
Note: The items in angle brackets above should be replaced with your personal details as required.

When the final command above runs, you should see something like this in the console when the clone process is completed:

```
$ git clone https://timlinux@github.com/AIFDR/inasafe.git inasafe-dev
Cloning into 'inasafe'...
remote: Counting objects: 5002, done.
remote: Compressing objects: 100% (1526/1526), done.
remote: Total 5002 (delta 3505), reused 4835 (delta 3338)
Receiving objects: 100% (5002/5002), 2.38 MiB | 7 KiB/s, done.
Resolving deltas: 100% (3505/3505), done.
```

Checkout the test data

To check out the test data from git, first open a GIT bash prompt as illustrated below:



The repository can now be cloned by issuing the commands listed below. (Already completed in previous step):

```
cd  /c/Documents\ and\ Settings/<your username>/.qgis/python/plugins/

git clone https://<your username>@github.com/AIFDR/inasafe_data.git inasafe_data
```

Note: The items in angle brackets above should be replaced with your personal details as required.

When the final command above runs, you should see something like this in the console when the clone process is completed:

```
$ git clone https://timlinux@github.com/AIFDR/inasafe_data.git inasafe_data
Cloning into 'inasafe_data'...
remote: Counting objects: 5002, done.
remote: Compressing objects: 100% (1526/1526), done.
remote: Total 5002 (delta 3505), reused 4835 (delta 3338)
Receiving objects: 100% (5002/5002), 2.38 MiB | 7 KiB/s, done.
Resolving deltas: 100% (3505/3505), done.
```

Install QGIS

Download the latest QGIS 'standalone' installer from http://download.qgis.org and install it by running the installation wizard and accepting the defaults throughout.

After opening QGIS (*Start* → *All Programs* → *Quantum GIS Lisboa* → *Quantum GIS Desktop (1.8.0)*) you need to enable the plugin from the plugin menu by doing *Plugins* → *Manage Plugins* and then search for the InaSAFE plugin in the list and enable it.

Windows Caveats

Our primary development platform is Linux (specifically Ubuntu Linux). Some features of the development environment - particularly the **Make** tools do not run on Windows. Some helper scripts have been written to substitute for make but they do not have feature parity with the make scripts.

3.2.3 Command line environment setup

Create a shell launcher

A command line environment is useful for running unit tests and for developing and testing standalone scripts written to use the InaSAFE libraries.

We will create a custom shell launcher that will give you a python shell environment using the python that comes bundled with QGIS, and that sets various paths and evironment variables so everything works as expected. Save the following listing in <QGIS Install Dir>/bin/python-shell.bat:

```
@echo off
SET OSGEO4W_ROOT=C:\PROGRA~1\QUANTU~1
call "%OSGEO4W_ROOT%"\bin\o4w_env.bat
call "%OSGEO4W_ROOT%"\apps\grass\grass-6.4.2\etc\env.bat
@echo off
SET GDAL_DRIVER_PATH=%OSGEO4W_ROOT%\bin\gdalplugins\1.9
path %PATH%;%OSGEO4W_ROOT%\apps\qgis\bin
path %PATH%;%OSGEO4W_ROOT%\apps\grass\grass-6.4.2\lib
path %PATH%;"%OSGEO4W_ROOT%\apps\Python27\Scripts\"

set PYTHONPATH=%PYTHONPATH%;%OSGEO4W_ROOT%\apps\qgis\python;
set PYTHONPATH=%PYTHONPATH%;%OSGEO4W_ROOT%\apps\Python27\Lib\site-packages
set QGIS_PREFIX_PATH=%OSGEO4W_ROOT%\apps\qgis
cd "%HOMEPATH%\.qgis\python\plugins\inasafe-dev"
start "Quantum GIS Shell" /B "cmd.exe" %*
```

Note: The QGIS_PREFIX_PATH environment variable should be unquoted!.

Note: You may need to replace PROGRA~1 above with PROGRA~2 if you are on 64bit windows.

Note: This script is for QGIS 1.8. You may need to do some adjustment if you are using another version of QGIS

For easy access to this shell launcher, right click on the qgis-shell.bat script and (without releasing your initial right click) drag with the file onto your start / windows button in the bottom left corner of the screen.

Verifying your system path

To verify your path, launch your python shell (by clicking the python-shell.bat) and then start a python shell. Now enter the follow simple script:

```
import sys
for item in sys.path:
    print item
```

Which should produce output like this:

```
C:\Users\inasafe\.qgis\python\plugins\inasafe-dev
C:\PROGRA~1\Quantum GIS Lisboa\apps\qgis\python
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\Lib\site-packages
C:\PROGRA~1\Quantum GIS Lisboa\bin\python27.zip
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\DLLs
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\plat-win
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\lib-tk
C:\PROGRA~1\Quantum GIS Lisboa\bin
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\PIL
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\win32
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\win32\lib
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\Pythonwin
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\wx-2.8-msw-unicode
```

It is particularly the second and third lines that you need to have in place so that the QGIS libs can found. Now dow a simple test to see if you can import the QGIS libs:

```
from qgis.core import *
exit()
```

Assuming you get no error messages, you have a functional python command line environment which you can use to test QGIS functionality with.

3.2.4 Nose testing tools

Installing pip

We need to install easy_install so that we can install pip so that we can install nosetests and other python tools. Under windows you need to run a little script to install easy_install and then use easy_install to install pypi. Download the script on this page called ez_setup.py and save it somewhere familiar e.g. `c:temp`.

Note: If you use windows 32bit, do not download the .exe file as said on the page, but just download the ez_setup.py

Next launch the shell (python-shell.bat as described in *Command line environment setup*) **as administrator** (by right clicking the file and choosing run as administrator). Then from the command line, launch **ez_setup.py** by typing this:

```
python c:\temp\ez_setup.py
```

Note: You will need to launch the shell as administrator whenever you need to install python packages by pypi.

Now in the same shell, use easy setup to install pip (make sure you have added the QGIS scripts dir to your shell launcher's - which should be the case if you have followed the notes in *Command line environment setup*):

```
easy_install pip
```

If the installation goes successfully, you should see output like this:

```
Searching for pip
Reading http://pypi.python.org/simple/pip/
Reading http://pip.openplans.org
Reading http://www.pip-installer.org
Best match: pip 1.1
Downloading http://pypi.python.org/packages/source/p/pip/pip-1.1.tar.gz#md5=62a9f08dd5dc69d76734568a
Processing pip-1.1.tar.gz
Running pip-1.1\setup.py -q bdist_egg --dist-dir c:\users\timsut~1\appdata\local
\temp\easy_install--zkw-t\pip-1.1\egg-dist-tmp-mgb9he
warning: no files found matching '*.html' under directory 'docs'
warning: no previously-included files matching '*.txt' found under directory 'docs\_build'
no previously-included directories found matching 'docs\_build\_sources'
Adding pip 1.1 to easy-install.pth file
Installing pip-script.py script to C:\PROGRA~2\QUANTU~1\apps\Python25\Scripts
Installing pip.exe script to C:\PROGRA~2\QUANTU~1\apps\Python25\Scripts
Installing pip.exe.manifest script to C:\PROGRA~2\QUANTU~1\apps\Python25\Scripts
Installing pip-2.5-script.py script to C:\PROGRA~2\QUANTU~1\apps\Python25\Scripts
Installing pip-2.5.exe script to C:\PROGRA~2\QUANTU~1\apps\Python25\Scripts
Installing pip-2.5.exe.manifest script to C:\PROGRA~2\QUANTU~1\apps\Python25\Scripts

Installed c:\progra~2\quantu~1\apps\python25\lib\site-packages\pip-1.1-py2.5.egg
Processing dependencies for pip
Finished processing dependencies for pip
```

Note: If your Windows is 64bit, you need to do a little trick to install pip. First you have to install python 32bit and add its path to PATH variable in environment variable (on Windows 7: *System Properties –> Advanced –> Environment Variables*). After that, run command prompt (as administrator if needed) and run *easy_install pip*

Installing nose

Nose is a tool for automation of running python unit tests. With nose you can run a whole batch of tests in one go. With the nosecover plugin you can also generate coverage reports which will indicate how many lines of your code actually have been tested.

To install these tools, launch your python prompt as administrator and then do:

```
pip install nose nose-cov
```

Running tests using nose

Once they are installed, you can run the nose tests from windows by going to the plugin directory/inasafe-dev folder (in your python-shell.bat shell session) and running:

```
runtests.bat
```

3.2.5 Building sphinx documentation

Sphinx is a tool for building documentation that has been written in the ReSTructured text markup language (a simple wiki like format). You can build the sphinx documentation under windows using a helper script provided in the docs directory of the InaSAFE source directory, but first you need to actually install sphinx.

Installing sphinx

Launch your QGIS python shell environment (see *Installing pip*) as administrator and then run the following command:

```
pip install sphinx
```

The cloud-sptheme package installs the sphinx theme we are using.

Building the documentation

To build the documentation, open a QGIS python shell (no need to be admin) and go into your inasafe-dev/docs directory. Now run the following command:

```
make.bat html
```

Note: Only the html make target has been tested. To use other make targets you may need to perform further system administrative tasks.

Viewing the documentation

The documentation can be viewed from withing QGIS by clicking the *help* button on the InaSAFE dock panel, or you can view it in your browser by opening a url similar to this one:

```
file:///C:/Users/Tim%20Sutton/.qgis/python/plugins/inasafe/docs/_build/html/index.html
```

3.2.6 Developing using Eclipse (Windows)

Note: This is optional - you can use any environment you like for editing python, or even a simple text editor.

If you wish to use an IDE for development, please refer to this article for detailed information on how to get the basic Eclipse with PyDev setup.

Installing Eclipse

You can download and install eclipse by getting the latest installer at eclipse.org. Just run the installer accepting all defaults.

Installing PyDev

With Eclipse running, click on *Help* → *Eclipse Marketplace* and from the resulting dialog that appears, type `PyDev` into the search box and then click *Go*. On the search results page, choose PyDev and click the *Install* button next to it. Agree to the license terms and accept the aptana certificate, then restart Eclipse as requested.

Custom Eclipse Launcher

You need to create a custom Eclipse launcher in order to use Eclipse PyDev. The process is similar to *Command line environment setup* in that you need to create a custom batch file that launches eclipse only after the osgeo4w environment has been imported. Here are the typical contexts of the file:

```
@echo off

SET OSGEO4W_ROOT=C:\PROGRA~2\QUANTU~1
call "%OSGEO4W_ROOT%"\bin\o4w_env.bat
call "%OSGEO4W_ROOT%"\apps\grass\grass-6.4.2\etc\env.bat
@echo off
SET GDAL_DRIVER_PATH=%OSGEO4W_ROOT%\bin\gdalplugins\1.8
path %PATH%;%OSGEO4W_ROOT%\apps\qgis\bin;%OSGEO4W_ROOT%\apps\grass\grass-6.4.2\lib
set PYTHONPATH=%PYTHONPATH%;%OSGEO4W_ROOT%\apps\qgis\python;
set PYTHONPATH=%PYTHONPATH%;%OSGEO4W_ROOT%\apps\Python27\Lib\site-packages
set QGIS_PREFIX_PATH=%OSGEO4W_ROOT%\apps\qgis
"C:\Progra~2\eclipse\eclipse.exe"
```

Note: Use the path where your eclipse was extracted. Also note that PROGRA~2 may be PROGRA~1 in 32bit windows.

Save this file under <QGIS Install Dir>/bin/python-shell.bat and then right-drag it from explorer to your Windows start button to create an easily accessible shortcut to eclipse.

Creating a project

The procedure for doing this is to do: *File* → *New* → *Project...* and then from the resulting dialog do *PyDev* → *PyDev Project*.

In the resulting project dialog, set the following details:

- *Project name:* : `inasafe`
- *Use default* : `uncheck`
- :guilabel (windows):*Directory* : `C:\Users\<user>\.qgis\python\plugins\inasafe\`
- *Choose project type* : `Python`
- *Grammar Version* : `2.7`
- *Add project directory to PYTHONPATH?* : `check`

Note: The python shipped with QGIS for windows is version 2.7 so you should avoid using any additions to the python spec introduced in later versions.

At this point you should should click the link entitled 'Please configure an interpreter in related preferences before continuing.' And on the resulting dialog do:

- *Python Interpreters: New...* : `click this button`

In the dialog that appears do:

- *Interpreter Name* : `QGIS Python 2.7`
- *Interpreter Executable* : `C:\Program Files (x86)\Quantum GIS Lisboa\bin\python.exe`
- *OK Button* : `click this button`

Another dialog will appear. Tick the first entry in the list that points to your:

```
C:\\users\\inasafe\\Downloads\\eclipse\\plugins\\org.python.pydev
_2.6.0.2012062818\\pysrc
```

The resulting list of python paths should look something like this:

```
C:\Program Files\eclipse\plugins\org.python.pydev_2.6.0.2012062818\pysrc
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\DLLs
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\plat-win
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\lib-tk
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\win32
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\win32\lib
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\Pythonwin
C:\PROGRA~1\Quantum GIS Lisboa\apps\Python27\lib\site-packages\wx-2.8-msw-unicode
```

Click on the *New folder* button and add the QGIS python dir:

```
C:\Program Files\Quantum GIS Lisboa\apps\qgis\python
```

- *OK Button* : `click this button`

You will be returned to the Python Interpreters list and should see an entry for **QGIS Python 2.7** listed there. Now do in the **Environment** tab:

New

In the dialog that appears

Name : `QGIS_PREFIX_PATH` *Value* : `C:\PROGRA~1\QUANTU~1\apps\qgis`

Then click ok to close the environment variable editor.

- *Ok* : `click this button`

Then click finsih to finish the new project dialog .

- *Finish* : `click this button`

Remote Debugging with Eclipse

For remote debugging, you should add pydevd to your PYTHONPATH before starting *QGIS*. Under Windows, the best way to do this is to add the following line to **qgis.bat** under C:Program Files (x86)Quantum GIS Wroclawbin:

```
SET PYTHONPATH=%PYTHONPATH%;C:\Progra~1\eclipse\plugins\org.python.pydev.debug_2.3.0.2011121518\pysrc
```

Note: (1) You need to add a settrace() line at the point in your code where you would like to initiate remote debugging. After that, you can insert eclipse debugger breakpoints as per normal.

(2) If you are running with remote debugging enabled, be sure to start the PyDev debug server first before launching the Risk-in-a-box QGIS plugin otherwise QGIS will likely crash when it can't find the debug server.

(3) Place the above PYTHONPATH command before the final line that launches QGIS!

(4) The exact path used will vary on your system - check in your eclipse plugins folder for "org.python.pydev.debug_<some date> to identify the correct path.

To initiate a remote debugging session, add the settrace() directive to your source file and then start the python remote debugging service from the PyDev debug perspective. Then launch QGIS (or your command line application) and use the application until the settrace line is encountered. QGIS will appear to freeze - this is normal. Now switch to Eclipse and you should see the settrace line has been highlighted in green and you can step through the code using standard Eclipse debugging tools (done most easily from the debugging perspective).

Note: Always remove or comment out settrace() when are done debugging!

Running Unit tests from the IDE

Using PyDev's build in test runner

Python has very good integrated support for unit testing. The first thing you should do after setting up the IDE project is to run the tests. You can run tests in the following ways:

- For the entire inasafe package
- For individual sub packages (e.g. engine, gui, storage, impact_functions)
- for an individual test module within a package
- for an class within a test module
- for an individual method within a test class

You can view these individual entities by browsing and expanding nodes in the project panel in the left of the IDE.

Note: If you run the test suite for the entire inasafe package, it will mistakenly treat the sphinx documentation conf.py (docs.source.conf) as a test and fail for that test. This is 'normal' and can be ignored.

Setting PyDev to use the Nose test runner

You can also configure Eclipse to run the tests using nose (which is recommended). To do this first do:

Window → Preferences → PyDev – PyUnit

Now set *TestRunner* to `Nosetests` and set the following options:

```
-v --with-id --with-coverage --cover-package=storage,engine,impact_functions,gui
```

As with using Pydev's built in test runner, you can also run any module, class etc. while using the nose test runner by right clicking on the item in the PyDev package explorer.

Note: Actually, we can run the test runner until this step. But, we got a problem, so you need to install python in your windows machine.

3.3 Coding Standards

3.3.1 Code Style

Please observe the following coding standards when working on the codebase:

- Docstrings quoted with `"""`
- Simple strings in source code should be quoted with `'`
- Coding must follow a style guide. In case of Python it is pep8 and using the command line tool pep8 (or `make pep8`) to enforce this. The pep8 checks E121-E128 have been disabled until pep8 version 1.3 becomes widely available.
- Python documentation guide
- Comments should be complete sentences. If a comment is a phrase or sentence, its first word should be capitalized, unless it is an identifier that begins with a lower case letter (never alter the case of identifiers!). Comments should start with a # and a single space.
- Adherence to regression/unit testing wherever possible (`make test`)
- Use of github for revision control, issue tracking and management
- Simple deployment procedure - all dependencies must be delivered with the plugin installer for QGIS or exist in standard QGIS installs.
- Develop in the spirit of XP/Agile, i.e. frequent releases, continuous integration and iterative development. The master branch should always be assumed to represent a working demo with all tests passing.
- All strings should be internationalisation enabled. Please see *Internationalisation Support* for details.
- Code must pass a pylint validation (http://www.logilab.org/card/pylint_manual#what-is-pylint). You can test this using the make target `make pylint`. In some cases you may wish to override a line or group of lines so that they are not validated by lint. You can do this by adding either:

  ```
  import foo  # pylint: disable=W1203
  ```

  or:

  ```
  # pylint: disable=W1234
  print 'hello'
  print 'goodbye'
  # pylint: enable=W1234
  ```

  The relevant id (W1234) is provided on the output of the above mentioned lint command's output. A complete list of codes is available at http://pylint-messages.wikidot.com/all-codes.

  Note: You can globally ignore messages by adding them to `pylintrc` in the `[MESSAGES CONTROL]` section.

  The following pylint messages have been thus globally excluded from the check. For a discussion of these see also github issue https://github.com/AIFDR/inasafe/issues/245.

  - **All type R: Refactor suggestions such as limiting the number of local** variables. We may bring some back later.
  - All type I: Information only
  - W0142: Allow the Python feature F(*args, **kwargs)
  - W0201: Allow definition of class attributes outside the constructor.
  - W0212: Allow access to protected members (e.g. _show_system_info)
  - W0231: Allow classes without constructors.
  - W0232: Un-instantiated classes is a feature used in this project.
  - W0403: Relative imports are OK for modules that live in the same dir

  - W0511: Appearance of TODO and FIXME is not a sign of poor quality
  - E1101: Disable check for missing attributes.
  - E1103: This one does not understand numpy variables.
  - C0103: Allow mathematical variables such as x0 or A.
  - C0111: Allow missing docstrings in some cases
  - C0302: No restriction on the number of lines per module

  It is of course possible to run all pylint checks on any part of the code if desired: E.g pylint safe/storage/raster.py

- Each source file should include a standard header containing copyright, authorship and version metadata as shown in the exampled below.

Example standard header:

```
# -*- coding: utf-8 -*-
"""**One line description.**

.. tip::
   Detailed multi-paragraph description...

"""

__author__ = 'Ole Nielsen <ole.moller.nielsen@gmail.com>'
__revision__ = '$Format:%H$'
__date__ = '01/11/2010'
__license__ = "GPL"
__copyright__ = 'Copyright 2012, Australia Indonesia Facility for '
__copyright__ += 'Disaster Reduction'
```

Note: Please see *How did you embed the git version SHA1 into each .py file?* for details on how the revision tag is replaced with the SHA1 for the file when the release packages are made.

Doc strings

All code should be self documenting. We used the following style for documenting functions and class methods:

```
def setKeywordDbPath(self, thePath):
    """Set the path for the keyword database (sqlite).

    The file will be used to search for keywords for non local datasets.

    Args:
        * thePath: a valid path to a sqlite database. The database does
              not need to exist already, but the user should be able to write
              to the path provided.
    Returns:
        None
    Raises:
        None
    """
    self.keywordDbPath = str(thePath)
```

- If you use a few lines of code in more than one place, refactor them into their own function.
- If you use a literal string or expression in more than one place, refactor it into a function or variable.

Various other sphinx markup elements may be used in the docstrings too. For more information see also: http://thomas-cokelaer.info/tutorials/sphinx/docstring_python.html

Human Interface Guidelines

For consistency of user experience, the user interfaces created in Risk in a Box should adhere to the QGIS Human Interface Guidelines (HIG) which are listed here for your convenience:

- Group related elements using group boxes: Try to identify elements that can be grouped together and then use group boxes with a label to identify the topic of that group. Avoid using group boxes with only a single widget / item inside.
- Capitalise first letter only in labels: Labels (and group box labels) should be written as a phrase with leading capital letter, and all remaing words written with lower case first letters
- Do not end labels for widgets or group boxes with a colon: Adding a colon causes visual noise and does not impart additional meaning, so don't use them. An exception to this rule is when you have two labels next to each other e.g.: Label1 [Plugin Path:] Label2 [/path/to/plugins]
- Keep harmful actions away from harmless ones: If you have actions for 'delete', 'remove' etc, try to impose adequate space between the harmful action and innocuous actions so that the users is less likely to inadvertantly click on the harmful action.
- Always use a QButtonBox for 'OK', 'Cancel' etc buttons: Using a button box will ensure that the order of 'OK' and 'Cancel' etc, buttons is consistent with the operating system / locale / desktop environment that the user is using.
- Tabs should not be nested. If you use tabs, follow the style of the tabs used in QgsVectorLayerProperties / QgsProjectProperties etc. i.e. tabs at top with icons at 22x22.
- Widget stacks should be avoided if at all possible. They cause problems with layouts and inexplicable (to the user) resizing of dialogs to accommodate widgets that are not visible.
- Try to avoid technical terms and rather use a laymans equivalent e.g. use the word 'Transparency' rather than 'Alpha Channel' (contrived example), 'Text' instead of 'String' and so on.
- Use consistent iconography. If you need an icon or icon elements, please contact Robert Szczepanek on the mailing list for assistance.
- Place long lists of widgets into scroll boxes. No dialog should exceed 580 pixels in height and 1000 pixels in width.
- Separate advanced options from basic ones. Novice users should be able to quickly access the items needed for basic activities without needing to concern themselves with complexity of advanced features. Advanced features should either be located below a dividing line, or placed onto a separate tab.
- Don't add options for the sake of having lots of options. Strive to keep the user interface minimalistic and use sensible defaults.
- If clicking a button will spawn a new dialog, an ellipsis (...) should be suffixed to the button text.

Code statistics

- https://www.ohloh.net/p/inasafe/analyses/latest
- https://github.com/AIFDR/inasafe/network
- https://github.com/AIFDR/inasafe/graphs

3.4 Logging

InaSAFE includes a logging subsystem that can be used to:

- record event messages in the QGIS Log Console
- record event messages to a file e.g. `/tmp/inasafe/10-10-2012/timlinux/logs/inasafe.log`
- record exception details to a Sentry installation available at http://sentry.linfiniti.com/inasafe/
- email a developer a message when a logging event occurs (currently disabled)

In this section we describe best practices and procedures for logging in InaSAFE.

3.5 Getting and using the named Logger instance

We use the 'InaSAFE' logger instance as standard. It is the responsibility of each client package (e.g. `safe_qgis`) to setup the logger - typically in the `__init__.py` for the package:

```
from utilities import setupLogger
setupLogger()
```

The utility method that sets up the logger will determine which logging backends are made available. In the `safe_qgis` package, a number of different backends are setup in the **:funct:'setupLogger'** function. The logger will typically be assigned to a module variable `LOGGER`.

To actually use the logger in your module you need to do something like this:

```
import logging
LOGGER = logging.getLogger('InaSAFE')

# And then in your class / method:
LOGGER.debug('Hello world')
```

3.6 Logging exceptions

It is recommended to log exceptions as per the following example:

```
try:
    1/0
except Exception:
    LOGGER.exception('Something went terribly wrong')
```

The exception log type will cause the full traceback, the exception message and the message provided to the LOGGER.exception call to all be logged e.g.:

```
2012-10-10 10:53:54,733 - InaSAFE - ERROR - Something went terribly wrong
Traceback (most recent call last):
  File "<input>", line 2, in <module>
ZeroDivisionError: integer division or modulo by zero
```

The above example was contrived in the QGIS python console. When the exception originates inside a module, the traceback will include the complete call tree.

3.7 Logging in loops

Warning: Please be considerate when logging into loops as this can slow execution a lot (we had a spatial analysis loop with two logged messages and it took 15sec for 1000 itterations, removing logging brought it to 5sec) See also *Profiling*

3.8 Remote logging

There is support for logging to a remote server. This currently intended for developer use only and will provide ongoing statistics about the number and nature of exceptions taking place in InaSAFE.

Note: For privacy / security reasons this is disabled by default and you need to jump through two hoops to make it work.

The remote server is available here: http://sentry.linfiniti.com/inasafe/

Remote logging is implemented using raven and sentry. Raven needs to be installed on the local client. On ubuntu you can install it by doing:

```
sudo pip install raven
```

To prevent user's unwittingly sending exception reports, it is required to first set an environment variable before starting QGIS / running tests:

```
export INASAFE_SENTRY=1
```

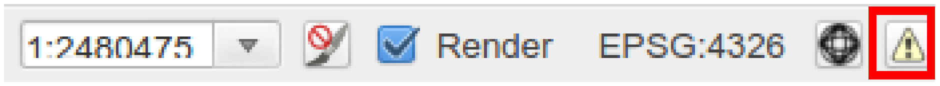
Note: The sentry logger is set to only log exceptions.

Here is an example session which will install raven, enable sentry and then launch QGIS:

```
sudo pip install raven
export INASAFE_SENTRY=1
/usr/local/bin/qgis
```

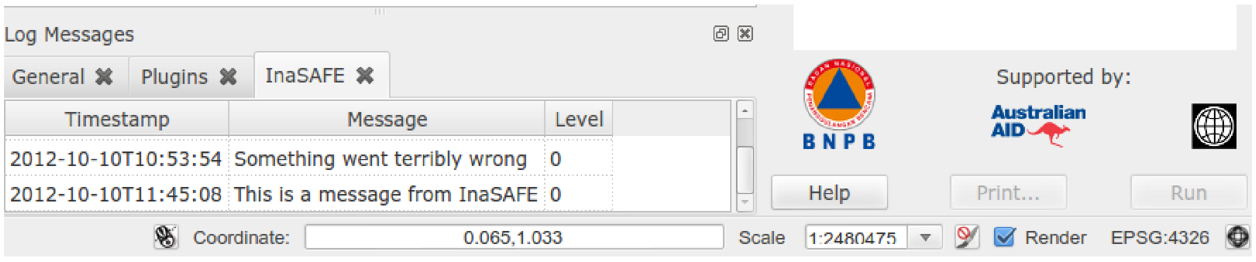
3.9 QGIS Log Messages

For the `safe_qgis` package, log messages will also be written to the QGIS log console under a tab labelled 'InaSAFE'. You can view these messages by clicking on the small triangular icon in the bottom right corner of the QGIS main window.



Clicking on the triangle indicated in red above will open the log dock window in QGIS from where you can browse log messages conveniently.

Note: QGIS 1.8 or greater is required for this functionality.



Timestamp	Message	Level
2012-10-10T10:53:54	Something went terribly wrong	0
2012-10-10T11:45:08	This is a message from InaSAFE	0

3.10 Logging with third party applications

If you have written your one SAFE library client, you should set up your own logger instance - just be sure that it is a named logger (called `InaSAFE`) and any log messages from the safe library will be written to your logger. For inspiration on how to do this, take a look at the

> `setupLogger()` function in `safe_qgis/utilities.py`.

3.11 Profiling

Profiling a script in python is as easy as calling: python -m cProfile myscript.py see also: http://docs.python.org/2/library/profile.html#module-cProfile

the problem is that sometimes the code you want to profile deep in ianSAFE. You can still get nice cProfiles by replacing the original call with a call to cProfile.runctx()

so:

```
self.preparePolygonLayerForAggr(theClippedHazardFilename, myHazardLayer)
```

would become:

```
cProfile.runctx('self.preparePolygonLayerForAggr(theClippedHazardFilename, myHazardLayer)', globals()
```

see also http://stackoverflow.com/questions/1031657/profiling-self-and-arguments-in-python

You can put a raise statement right after the runctx call so the execution is stopped and you can see your cProfile results in the console

3.12 Internationalisation Support

This document is divided into three parts:

- Quick reference - use this is you just want to quickly remember how to mark a word or phrase for translation
- QGIS Plugin - this is in-depth information on how the translation framework is set up for Qt based code components
- Library - this is in-depth information on how the translation framework is set up for the pure python library implementation.

3.12.1 Quick Reference

The translation system works differently for different parts of the project:

- Strings that appear in the InaSAFE graphical user interface (gui module). These include buttons, help strings, menu items etc
- Strings that appear in the Python code which constitutes the rest of InaSAFE including the impact functions
- Strings that appear in data and keywords used by InaSAFE

Flagging strings for translation is done differently for these three cases and are treated separately in the next three sections. However, making and deploying the translations is unified and takes care of all three and described in section "Make translations" below.

Translation of strings in the gui module

- Classes and anonymous functions that do not inherit from the class QObject should use the format `QCoreApplication.translate('Riab', 'Translations loaded')`
- Classes that inherit from QObject should use the form `self.tr('foo')`
- String replacement arguments should be provided using the QString `arg` method. Example: `self.tr('Error: %1').arg(message)`

Translation of strings *not* in the gui such as impact functions

- Import the gettext helper e.g:

  ```
  from safe.common.utilities import ugettext as _
  ```

  ```
  (Note that you must have the sequence "import ugettext" in the statement. It will not work as pa
  ```

- All strings should be wrapped using the _ helper e.g:

  ```
  tr('Are there enough shelters available for %i people?') % displaced
  ```

- The library will use at run-time the `LANG` environment variable which should be set to the iso code e.g. 'id' for 'Indonesia' of the Locale you wish to use. This is done automatically for you by the QGIS Plugin, but if you are using the riab library in another context, be sure to set it before using any libary functions if you want them to return translated strings e.g.:

  ```
  os.environ['LANG'] = 'id'
  # do stuff with riab lib
  ```

Translation of strings that appear at runtime

This applies e.g. to titles of layers or attribute names in data.

The translation system works by scanning the Python code for strings marked as described above. However, it has know way of knowing about titles of layers or names that appear in datasets processed by InaSAFE. However, if such names are known a priori they can be made visible to the translation system as follows:

- Edit the file:

  ```
  common/dynamic_translations.py
  ```

  and add the name to the dictionary "names". E.g. `'college' = tr('college')`

- Update the translation strings as described in the section below
- Make impact functions refer to the dynamic translations e.g. as in this example:

```
from common.dynamic_translations import names as internationalised_values

if building_type in internationalised_values:
    building_type = internationalised_values[building_type]
```

3.12.2 Make translations

When new strings have been added as described above the procedure to translate them is (example is given for LANG=id):

- run `make update-translation-strings` to collect all strings marked for translation
- Using either an editor or the tool Qt Linguist provide translations in the files
  - safe/i18n/id/LC_MESSAGES/inasafe.po
  - safe_qgis/i18n/inasafe_id.ts
- run `make compile-translation-strings` to make the translations available to InaSAFE

3.12.3 QGIS Plugin

The QGIS Plugin uses QtLinguist. this free, open source application can be downloaded and used to translate the Qt translation files.

Preparing for a release

As developer, before a release you should do:

- run `make update-translation-strings` to update the translation files
- distribute the .ts files under `gui/i18n` to the translators
- instruct them to open the .ts file for their locale with QtLinguist
- commit the returned file from the translator when all strings have been translated
- run `make compile-translation-strings` to create binary loadable translations
- ensure the .qm files are distributed with the release (the .ts files do not need to be released)

Note: Translators should take heed - when refreshing the .ts file in QtLinguist, the file *must be closed* (*File* → *Close*) and then reopened. Simply loading doing (*File* → *Open*) and choosing the same file you already have in the workspace will not refresh the workspace with any new changes that appeared on disk.

Note: *make update-translation-strings* is non destructive. That is, you can safely run it as many times as you like, new strings will be added to it, deprecated strings will be left in place and already translated strings will remain translated.

Adding a new language

To add a new language, edit the `gui/riab.pro` file and append the new locale to the bottom of the file. For example, to add South African english as a new locale, change this:

```
TRANSLATIONS = i18n/riab_id.ts
```

to this:

```
TRANSLATIONS = i18n/riab_id.ts\
               i18n/riab_en_ZA.ts
```

Save and close the .pro file. Next run `make update-translation-strings` to generate the new .ts file under gui/i18n. Don't forget to `git add` the new file and place it under version control.

3.12.4 InaSAFE Library Translations

Low level gettext usage

Translation is done using gettext.

Create the initial .po file:

```
xgettext -d id -o i18n/id/LC_MESSAGES/riab.po i18ntest.py
```

After you create the initial .pot, you need to specify the characterset and encoding for that file (by editing it with a text editor). For example:

```
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
```

If you add strings to the file, update the .pot file by adding -j option:

```
xgettext -j -d id -o i18n/id/LC_MESSAGES/riab.po i18ntest.py
```

Next, you can make the .po files available to translators. Recent versions of QtLinguist support translations of .po files, so you can use a similar process to that described in the gui section above.

When the .po file has been updated, it should be committed to the git repository (e.g. via a pull request from the user's repository clone, or by emailing the .po file to a developer). After receiving an updated .po file, it should be compiled to a `.mo` file (which is a binary representation of the strings):

```
msgfmt -o i18n/id/LC_MESSAGES/riab.mo i18n/id/LC_MESSAGES/riab.po
```

The `msgfmt` command accepts one or more input files which can be merged into a single `.mo`.

Note: These functions are wrapped as make scripts so you should not need to use them on a day to day basis.

Preparing for a release

As developer, before a release you should do:

- run `make update-translation-strings` to update the translation files
- distribute the .po files under `i18n/<locale>/LC_MESSAGES/riab.po` to the translators
- instruct them to open the .po file for their locale with QtLinguist
- commit the returned file from the translator when all strings have been translated
- run `make compile-translation-strings` to create binary loadable translations (.mo files)
- ensure the .mo files are distributed with the release (the .po files do not need to be released)

Note: Translators should take heed - when refreshing the .po file in QtLinguist, the file *must be closed* (*File* → *Close*) and then reopened. Simply loading doing (*File –< Open*) and choosing the same file you already have in the workspace will not refresh the workspace with any new changes that appeared on disk.

Note: *make update-translation-strings* is non destructive. That is, you can safely run it as many times as you like, new strings will be added to it, deprecated strings will be left in place and already translated strings will remain translated.

Adding a new language

To add a new language, edit the `Makefile` file and append the new locale to the bottom of the file. For example, to add South African english as a new locale, change this section:

```
update-translation-strings: compile
```

copy one of the existing stanzas e.g.:

```
xgettext -j -d id -o i18n/id/LC_MESSAGES/riab.po \
   storage/test_io.py \
   impact_functions/flood/flood_building_impact.py
```

Save and close the Makefile file. Next you need to create the initial translation stringlist for that locale by creating a locale directory and running the command above without the `-j` (j is for 'join' which merges old content with new, avoiding destroying previous translated strings). So for example you would run from the command line:

```
mkdir -p i18n/en_ZA/LC_MESSAGES/
xgettext -d id -o i18n/en_ZA/LC_MESSAGES/riab.po \
   storage/test_io.py \
   impact_functions/flood/flood_building_impact.py
```

The above adding a hypothetical new translation for South African English. After the inital creation of your .po files using the above commands, you can update them anytime the strings in the library have been changed by doing:

```
make update-translation-strings`
```

to generate the updated .po file under i18n/en_ZA/LC_MESSAGES. Don't forget to `git add` the new directory and file and place them under version control.

To convert the .po file to a binary .mo file (which is used at runtime for the actual translation), follow the *Preparing for a release* section above.

Adding a new source file for translation

To add a new source file, edit the `Makefile` file and append the new sourcefile to the bottom of the file list in the `update-translation-strings` section. For example:

```
xgettext -j -d id -o i18n/id/LC_MESSAGES/riab.po \
   storage/test_io.py \
   impact_functions/flood/flood_building_impact.py
```

Would become:

```
xgettext -j -d id -o i18n/id/LC_MESSAGES/riab.po \
   storage/test_io.py \
   impact_functions/flood/flood_building_impact.py \
   impact_functions/flood/flood_population_fatality
```

The above adding the impact_function *flood_population_fatality* to the list of translatable source files. Now run:

```
make update-translation-strings`
```

to generate the updated .po file and make it available to translators. When the translated file is returned, convert the .po file to a binary .mo file (which is used at runtime for the actual translation), then follow the *Preparing for a release* section above.

3.12.5 Sphinx Translation

Generic documentation on how to translate sphinx documentation is available *here <http://sphinx.pocoo.org/latest/intl.html>*-.

Initial notes on the process:

```
cd docs
make gettext
```

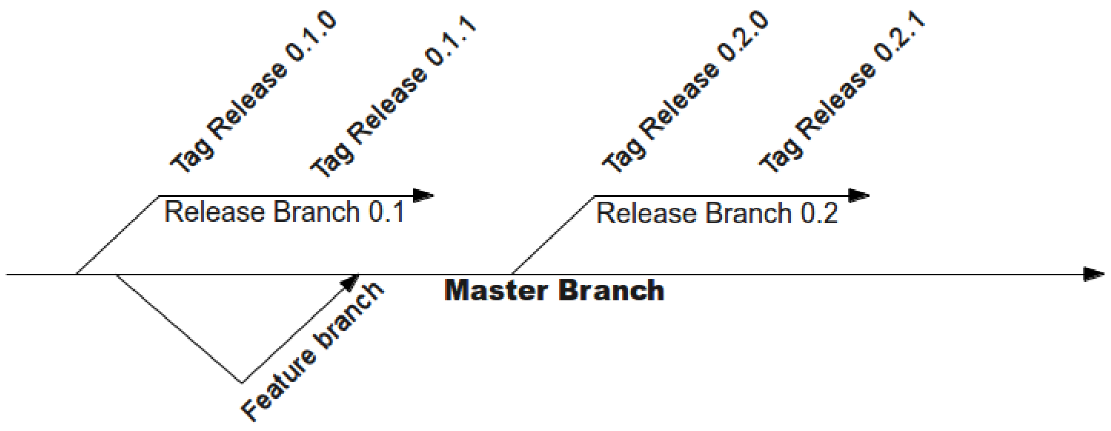
Which will create pot files which can be found under build/locale. Note that this make target needs to be tweaked so that it builds a unique directory for each supported locale.

3.13 Version Control

We are using git for version control. You can find all the latest source code here: https://github.com/AIFDR/inasafe

3.13.1 Branching guide

InaSAFE follows the following simple branching model:



New development takes place in *master*. Master should always be maintained in a usable state with tests passing and the code functional as far as possible such that we can create a new release from master at short notice.

Releases should take place in long lived branches named after the minor version number (we follow the semantic versioning scheme) so for example the first release would be version 0.1 and would be in a branch from master called *release_0-1*.

After the minor release branch is made, the *point releases (patch)* are created as tags off that branch. For example the release flow for version 0.1.0 would be:

- branch release_0.1 from master
- apply any final polishing the the relase_0-1 branch
- when we are ready to release, tag the branch as release_0-1-0
- create packages from a checkout of the tag

After the release, development should take place in master. Additional short lived branches can be made off master while new features are worked on, and then merged into master when they are ready.

Optionally, development can also be carried out in independent forks of the inasafe repository and then merged into master when they are ready via a pull request or patch.

Commits to master that constitute bug fixes to existing features should be backported to the current release branch using the `git cherry-pick` command. Alternatively, if a fix is made in the release branch, the changeset should be applied to master where appropriate in order to ensure that master includes all bug fixes from the release branches.

3.13.2 Process for developers adding a new feature

Create a feature branch

- git checkout -b <featurebranch> master

Write new code and tests ...

Publish (if unfinished)

- git commit -a -m "I did something wonderful"
- git push origin <featurebranch>

To keep branch up to date:

- git checkout <featurebranch>
- git merge origin master
- (possibly resolve conflict and verify test suite runs)
- git push origin <featurebranch>

When all tests pass, either merge into master

- git checkout master
- git merge –no-ff <featurebranch> (possibly resolve conflict and verify test suite runs)
- git push origin master

Or issue a pull request through github

To delete when branch is no longer needed (though it is preferable to do such work in a fork of the official repo).

- git push origin :<featurebranch>

3.13.3 Process for checking out the release branch and applying a fix:

Create a local tracking branch:

```
git fetch
git branch --track release-0_1 origin/release-0_1
git checkout release-0_1
```

Now apply your fix, test and commit:

```
git commit -m "Fix issue #22 - results do not display"
git push
```

To backport the fix to master do (you should test after cherry picking and before pushing though):

```
git checkout master
git cherry-pick 0fh12
git push
```

3.13.4 To checkout someone else's fork:

Example:

```
git remote add jeff git://githup.com/jj0hns0n/riab.git
git remote update
git checkout -b impact_map jeff/impact_map
```

3.13.5 Windows Specific Notes:

To Switch branches using TortioiseGIT:

- Navigate to the inasafe plugin folder
- Right click on any whitespace
- From the context menu choose TortoiseGIT->Switch/Checkout
- Tick 'Branch radio button and choose 'master' from the list
- Click ok

To update the master branch:

- Right click on the whitespace again
- TortoiseGIT->Pull from the context menu
- Tick the remote radio
- Set remote to origin
- Tick the ellipses button next to 'Remote Branch'
- Choose 'master' from the list
- Click OK

For subsequent pull requests on that branch you can just do TortoiseGIT->Pull from the context menu and press ok

3.14 Maintaining the documentation

The documentation for InaSAFE is written using ReSTructured text (.rst) and the Sphinx documenation builder. The best way to learn how to write .rst is to look at the source of existing documentation - the markup syntax is very simple. There are a number of useful tags that you can use to make your documentation clear and visually interesting, the more commonly used in this document are listed below. For a more detailed list, please visit the Spinx Inline Markup page

A complete list of supported .rst markup is also available here.

3.14.1 Common tags

Here are some common useful tags:

```
Heading
=======

SubHeading
----------

Sub subheading
..............

`web link<http://foo.org>`_

:doc:filename  (rst file without  extension)

*italics*

**bold**

.. note:: Note in a little call out box

.. todo:: Todo item in a call out box

.. table:: table title

============  ================
  Key         Allowed Values
============  ================
units         m
units         wet/dry
units         feet
============  ================

:menuselection:`Plugins --> Manage Plugins`

:kbd:`Control-x Control-f`

:guilabel:`Ok Button`
```

Creating API Documentation

Each class method and function in the code base must include a docstring explaining its usage and purpose as per the example listed below (taken from the riab.py setupI18n method):

```
"""Setup internationalisation for the plugin.

See if QGIS wants to override the system locale
and then see if we can get a valid translation file
for whatever locale is effectively being used.

Args:
   thePreferredLocale - optional parameter which if set
       will override any other way of determining locale.
Returns:
   None.
Raises:
   no exceptions explicitly raised.
"""
```

There should be a blank line between each paragraph and before the Args option.

Where multiple inputs or outputs are used, a ReSTructured text bullet list should be used to list them.

Note: You can use any ReSTructured text withing the docstring to deliver rich markup in the final API documentation outputs.

In order for a new module's documentation to appear in the API docs, the following steps are required:

- Create a new file in `docs/source/api-docs/<package_name>` named after the module. For example, for the gui/riab.py module we would create `docs/source/api-docs/gui/riab.rst` (note the .rst extension). See below for an example of its contents
- Add the new file to the API docs master index (`docs/source/api-docs/index.rst`). The .rst extension is not needed or desired when adding to the index list.
- Regenerate the documentation using the **make docs** command from the top level directory in the source tree.
- Add the new .rst file and generated html files to the revision control system.

Note: It is probably most expedient to simply copy and rename one of the existing API documentation files and replace the python module paths therein.

An example of the contents of a module's API .rst if provided below:

```
Module:  safe.common.polygon
============================

.. automodule:: safe.common.polygon
      :members:
```

This module forms part of the InaSAFE tool.

A couple of things should be noted here:

- Sphinx provides automodule and autoclass directives. We have opted to use **automodule** for all API documentation because autoclass requires that each class be enumerated and anonymous functions need to be explicitly listed.
- Automodule must point to a fully qualified python module path.
- The **members** directive instructs autodocs to enumerate all classes and functions in that module.

Once the new document has been added and the documentation generated, you should see it appear in the API section of the InaSAFE documentation.

Documenting new features

New features should be well documented and that documentation should be made available uder the `user-docs` subfolder of the sphinx sources tree.

For example, when the keywords editor dialog feature was introduced, we created a new sphinx document `docs/sources/user-docs/keywords.rst` which documents this new feature. Additionally, the help button is set to launch the help dialog in the context of the new help document e.g.:

```
def showHelp(self):
   """Load the help text for the keywords gui"""
   if not self.helpDialog:
      self.helpDialog = RiabHelp(self.iface.mainWindow(), 'keywords')
   self.helpDialog.show()
```

Where the 'keywords' parameter indicates the user-docs/*.rst document that should be opened when the help button is clicked. The general style and approach used in existing documentation should inform your documentation process so that all the documentation is constent.

Publishing the documentation to GitHub Pages

Initially we have used http://readthedocs.org to host our site (and the pages you are reading now). However they don't support internationalisation and there are various other issues with it, so we opted to move our content into gh-pages. To use this, the site is stored in a special branch.

3.14.2 Initial gh-pages setup

In order to set up the gh-pages branch this is the procedure followed.

Note: This is a once-off process you do not need to repeat it, it is here for reference purposes only.

Enable gh-pages in the gh project admin page. On your local system do something like this:

```
git clone file:///home/timlinux/dev/python/inasafe-dev \
    inasafe-github-pages
cd inasafe-github-pages
cp ../inasafe-dev/.git/config .git/config
git pull
git symbolic-ref HEAD refs/heads/gh-pages
rm .git/index
git clean -fdx
cp -r ../inasafe-dev/docs/build/html .
cd html/
touch .nojekyll
git add .
git commit -a -m "First commit of docs"
git push origin gh-pages
```

Now wait ten minutes or so and the pages should be visble here at http://aifdr.github.com/inasafe/

See also: http://help.github.com/articles/creating-project-pages-manually

Deployment of the site requires the following steps:

- Update the documentation as needed
- Commit/push to master

- Run scripts/update_website.sh
- Apidoc are build automatically, this might update/create/remove some files. If it is the case, the script will ask you if you wish to commit those changes to master. Normally you should.
- Wait approximately 10 minutes

After this the changes should be visible here http://aifdr.github.com/inasafe/ and http://inasafe.org.

Also see http://github.com/AIFDR/inasafe/issues/257 for further details of how the documentation publishing process works.

3.15 Postprocessors

This document explains the purpose of postprocessors and lists the different available postprocessor and the requirements each has to be used effectively.

Note: This document is still a work in progress.

3.15.1 What is a postprocessor?

A postprocessor is a function that takes the results from the impact function and calculates derivative indicators, for example if you have an affected population total, the Gender postprocessor will calculate gender specific indicators such as additional nutritional requirements for pregnant women

3.15.2 Creating postprocessors

Adding a new postprocessor is as simple as adding a new class called XxxxxxPostprocessor that inherits AbstractPostprocessor with one mandatory method (process), 2 optional ones and as many indicators as you need.

the minimal class could look like this:

```
class MySuperPostprocessor(AbstractPostprocessor):
    def __init__(self):
        AbstractPostprocessor.__init__(self)

    def setup(self, params):
        AbstractPostprocessor.setup(self, None)

    def process(self):
        AbstractPostprocessor.process(self)

    def clear(self):
        AbstractPostprocessor.clear(self)

    def _calculate_my_indicator(self):
        x = 5
        A = 0.5
        myResult = 10 * x / A
        self._append_result('My Indicator', myResult)
```

After that you need to import the new class into postprocessor_factory and update AVAILABLE_POSTPTOCESSORS to include the postprocessor prefix (e.g. MySuper if the class is called MySuperPostprocessor)

As last step you have to update or add the *parameters* variable to the impact functions that you want to use the new postprocessor. This will need to include a dicionary of the available postprocessors as shown below.

```
parameters = {
        'thresholds': [0.3, 0.5, 1.0],
        'postprocessors':
            {'Gender': {'on': True},
             'Age': {'on': True,
                     'params': {
                        'youth_ratio': defaults['YOUTH_RATIO'],
                        'adult_ratio': defaults['ADULT_RATIO'],
                        'elder_ratio': defaults['ELDER_RATIO']
                        }
                    },
              'MySuper': {'on': True}
             }
        }
```

or as a minimum:

```
parameters = {'postprocessors':{'MySuper': {'on': True}}}
```

for implementation examples see AgePostprocessor and GenderPostprocessor which both use mandatory and optional parameters

3.15.3 Output

Dock.postprocOutput will hold the result datastructure (shown below) of all the postprocessors. The structure is then parsed by Dock._postProcessingOutput() and stored in the impact layer's keywords

Data structure of results

```
{'Gender': [
    (QString(u'JAKARTA BARAT'), OrderedDict([(u'Total', {'value': 278349, 'metadata': {}}),
                                              (u'Females count', {'value': 144741, 'metadata': {}}),
                                              (u'Females weekly hygiene packs', {'value': 114881, 'met
    (QString(u'JAKARTA UTARA'), OrderedDict([(u'Total', {'value': 344655, 'metadata': {}}),
                                              (u'Females count', {'value': 179221, 'metadata': {}}),
                                              (u'Females weekly hygiene packs', {'value': 142247, 'met
 'Age': [
    (QString(u'JAKARTA BARAT'), OrderedDict([(u'Total', {'value': 278349, 'metadata': {}}),
                                              (u'Youth count', {'value': 73206, 'metadata': {}}),
                                              (u'Adult count', {'value': 183432, 'metadata': {}}),
                                              (u'Elderly count', {'value': 21990, 'metadata': {}})])),
    (QString(u'JAKARTA UTARA'), OrderedDict([(u'Total', {'value': 344655, 'metadata': {}}),
                                              (u'Youth count', {'value': 90644, 'metadata': {}}),
                                              (u'Adult count', {'value': 227128, 'metadata': {}}),
                                              (u'Elderly count', {'value': 27228, 'metadata': {}})]))
    ]
}
```

3.16 Reporting bugs and getting help

We are very open to ideas, suggestions and patches to improve this software - it is open source after all! If you experience an issue you can report it here:

Issue tracker: https://github.com/AIFDR/inasafe/issues

3.17 Preparing a release

This document outlines the steps that need to be carried out in order to issue a new release of the InaSAFE plugin. The steps can be outlined as follows and are described in detail below:

- Identify what version number the new release will be assigned.
- Close all issues marked as blockers for the release.
- If needed, create a release branch
- Update all source files to the new version number.
- Update all source files for PEP8 and PEP257 compliance.
- Ensure no assert statements are critical to code flow
- Ensure that Qt resources and user interface files have been compiled
- Ensure that user interface files meet HIG compliance
- Enure all unit tests complete successfully and that tests that are expected to fail are documented.
- Ensure that user acceptance testing has been carried out.
- Ensure that all translation string lists have been updated and that the translation process has been carried out.
- Ensure that all new and existing features are adequately documented.
- Ensure that the API documentation is up to date.
- Update the changelog.
- Ensure that the sphinx documentation is compiled.
- Generate python optimsed (.pyo) files for all sources.
- Update the plugin metadata to reflect current version.
- Generate a test package and validate in a clean room environment.
- Optionally branch the release in the revision control system.
- Tag the release in the revision control system.
- Upload the updated package zip file to old QGIS python plugin repository.
- Upload the updated package zip file to the new QGIS python plugin repository.
- Make announcements and press releases as needed.

3.17.1 Release numbering

InaSAFE will follow the semantic versioning system. Simply put, the following scheme should be applied to version numbers:

Version Increment Intention	
Major e.g 1.0.0	API incompatibility with the previous major release.
Minor e.g. 1.1.0	API compatibility and extension over previous minor release.
Point e.g. 1.1.1	API compatibility, bug fixes for previous point release.
Alpha e.g. 1.0.0a	Feature incomplete preview of a minor or major release
RC e.g. 1.1.0rc1	Feature complete preview of a minor or major release

To identify the next release number, the table above can simply be appled. Here are a couple of examples.

- You have fixed various bugs without adding new features or breaking the API, and you are ready to immediately publish your work. Result: **New point release.**
- You have implemented many new features, some of which required breaking API compatibility with the existing major release. Now you would like to make a public preview of your work before committing to a final release. Result: **New major release candidate.**

Outcome: A version number for the next release eg. 0.1.0.

3.17.2 Issue completion

Having determined the release number, you should use the GitHub *labels* capability to assign a label matching the release to each blocking ticket. There is no fixed rule on which tickets should be tagged for the release - the best judgment of developers and managers should be used based on severity of issues, available time to deadline, budget etc.

Outcome: At the end of this step all issues tagged for the release should be closed.

3.17.3 Branching

Branching is requred for Major and Minor releases. The process of branching is described in *Version Control* whose accompanying illustration is repeated below for your convenience:

The purpose of creating a branch is to isolate incompatible and possibly unstable changes that take place in the *master branch* from stable code that forms the basis of a release. You will note from the diagram above that branches are named after the minor version, and are tagged with the point version at the point of release.

Outcome: If needed, create a release branch which provides a 'known good' version that can be returned to at any point in time.

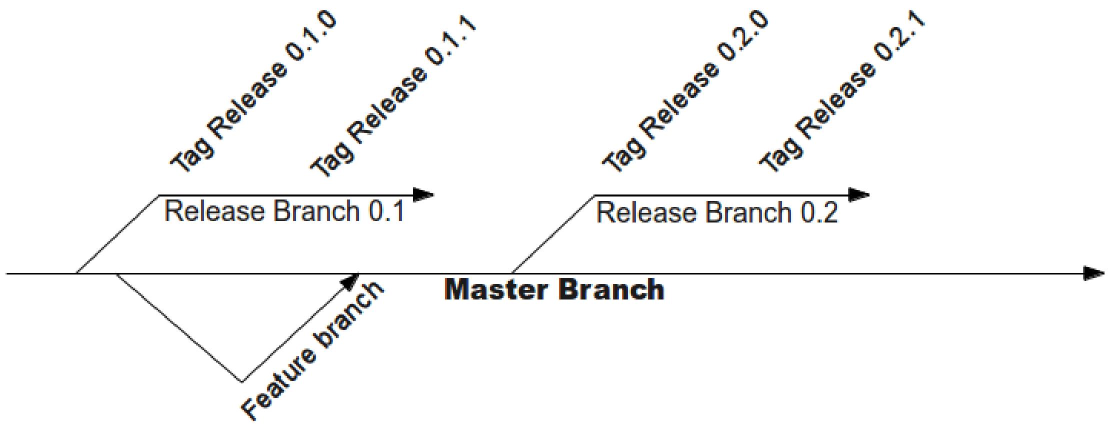
3.17.4 Updating the source version number

In the preample to any source file there should be a standard header as described in the *Coding Standards* document. Included in this header section is the version number e.g.:

```
__version__ = '0.1.0'
```

This number should be updated in every source file prior to release. Under linux this can easily be done using the **rpl** command (which can easily be installed (by doing for example **sudo apt-get install rpl**).

Using th example above, to update version numbers for minor release '0.1.1' you could issue the following command at the root of the plugin source tree:



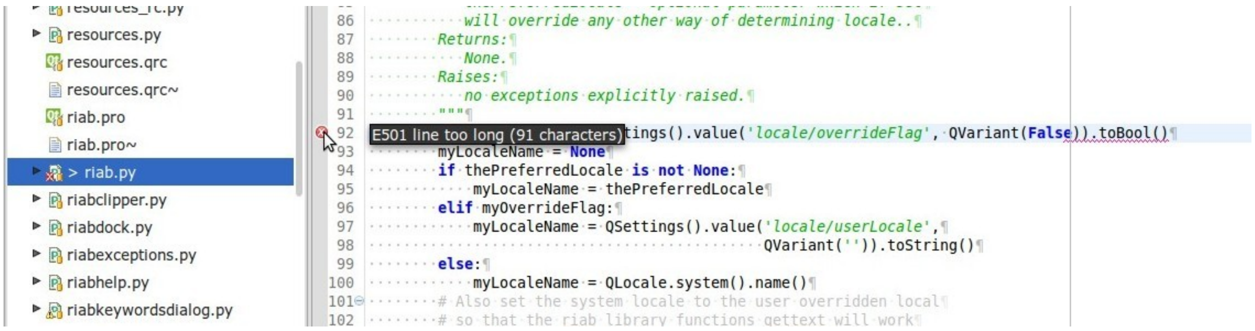
```
rpl "__version__ = '0.1.0'" "__version__ = '0.1.1'" *.py
```

Outcome:: Every source file should be updated to indicate the version number.

3.17.5 PEP8 and PEP257 compliance

These **Python Enhancement Proposals** (PEP) relate to the formatting of python source code. In particular they mandate spacing, layout, line lengths and so on. The outcome of PEP8 and PEP257 compliance is code that is consistently formatted accross the whole code base, regardless of authorship.

This consistency makes it easier to incorporate new members into the project team and to collaborate effectively within the team. A number of tools are available to help you to identify PEP8 and PEP257 transgressions, and there is a Makefile target (**make pep8** which will do a PEP8 test for you). Under the Eclipse/PyDev IDE, there is also on the fly checking support which can be enabled and that will notify you of any compliance issues as illutrated in the screenshot below.



```
            will override any other way of determining locale..
        Returns:
            None.
        Raises:
            no exceptions explicitly raised.
        """
E501 line too long (91 characters)tings().value('locale/overrideFlag', QVariant(False)).toBool()
        myLocaleName = None
        if thePreferredLocale is not None:
            myLocaleName = thePreferredLocale
        elif myOverrideFlag:
            myLocaleName = QSettings().value('locale/userLocale',
                                             QVariant('')).toString()
        else:
            myLocaleName = QLocale.system().name()
        # Also set the system locale to the user overridden local
        # so that the riab library functions gettext will work
```

Outcome: All source files for PEP8 and PEP257 compliance.

3.17.6 Check for assert statements

Using assert to raise exceptions in non test code can have bad side effects because if python is run in optimised mode e.g. python -O, these lines are ignored and the program logic will no longer work as expected. On some platforms the use of python optimised code is mandated and we are likely to get hard to investigate bug reports from end users at some unspecified point in the future.

Note: This is a 'soft' requirement - since the python code for the plugin will be executed by the QGIS python internals, we can be fairly certain that python code will be executed with out the -O optimisation option for the short term.

Outcome: No assert statements used to control logic flow.

3.17.7 Compile Qt resources and user interface files

The Qt4 resource and user interface definition files supplied with Risk in a Box need to be compiled before they can be deployed. There are two utility functions provided by Qt4 for this purpose:

- **pyuic4** - A tool to compile Qt4 user interface definition files (.ui) into python source code. The .ui files contain xml which describes the placement of widgets within a user interface file.
- **pyrcc4** - A tool to compile Qt4 resource files into python source code. Qt4 resources are 'in-code' representations of application resources needed at run time. These include images, icons, html, css etc. - whatever the application may need to use at runtime without resorting to retrieving assets from the filesystem.

The compilation of these resources if the default make target in the root and *gui* python package. To compile them simply do:

```
cd <inasafe source>
make
```

Outcome: Qt resources and user interface files have been compiled

3.17.8 HIG Compliance

The InaSAFE human interface guidelnes (HIG) are described in the *Human Interface Guidelines* document. User interface should strive to comply with these guidelines. As an over-arching principle, before any release, the user interface elements that comprise that release should be tested both for usability and to ensure that they are functional.

There is no automated test system for HIG. Before making a release of HIG compliance, each dialog should be manually tested and inspected.

Outcome: A consistent, user friendly and functional graphical user interface environment for the software that comprises the releases.

3.17.9 Unit Testing

During the development process, unit tests should be written (following the principles of test driven development). A good test suite allows the code to be shipped with confidence knowing it will behave as expected. At the time of release, all the tests in the test suite should either pass or have documented reasons as to why they fail, and that they are expected to fail.

In addition, tests should provide a code coverage of 80% or better of the shipped code base. More informationn on running unit tests is included in *Running tests*.

Outcome: All unit tests complete successfully, or when expected to fail are documented accordingly.

User Acceptance Testing

While unit testing provides a quantitative measure of the code's robustness, user acceptance testing provides a qualitative measure. The plugin should be made available to 'invested' users to test with real world data and in real world usage scenarios. Any issues with workflow, ease of use, quality of model outputs and reports etc. should be identified at this point and remedied.

Outcome: Software that works in real world usage.

Document new features

New features in the release should be well documented using the procedure described in *Documenting new features*.

Outcome: All new and existing features are adequately documented.

API Documentation

In addition to documenting new features, any new python modules introduced during the development work leading up to the release need to be included in the API documentation. This process is described in detail in the *Creating API Documentation* document.

Outcome: The API is completely documented with rich, relevant documentation.

Update the changelog

A changelog should be maintained (`docs/sources/user-docs/changelog.rst`) that lists the key new features and improvement made with each release. Use the *Whats new?* file to guide the style of any edits and additions made.

The changelog should not exhaustively list every commit that took place. Rather it should list the key features and bug fixes that were made during the release cycle.

Note: New release changesets should be introduced to this file **at the top** so that the newest release is alwas listed first.

Outcome: A succinct list of changes and improvements that were made during the release cycle.

3.17.10 Finalise translations

The InaSAFE plugin is built from the ground up for internationalization. In particular the following two languages are supported as part of this project:

- English
- Bahasa Indonesian

There are three components of the project that require translation:

- The Graphical User Interface - primarily the `gui` python package. Qt4 .ts files are used for these translations.
- The InaSAFE libraries - these components provide the underlying functionality of the scenario assessment. Python gettext is used for these translations.
- The sphinx documentation - this is translated using gettext.

The translation process for the first two items above is documented in detail in *Internationalisation Support*. The sphinx translation process is not yet well documented, although it will be similar to the gettext process.

The final strings should be made available to translators before the release, during which time a string freeze should be in effect on the release code tree.

Once the translation files have been updated, they should be converted to compiled string lists (.qm and .mo files for Qt4 and gettext respectively) and made available as part of the distribution.

Outcome: The released plugin will be multilingual supporting both indonesian and english.

Compile the sphinx documentation

Once documentation is completed, it should be compiled using **make docs** and the **git status** command should be used to ensure that all generated documentation is also under version control.

Outcome: Sphinx documentation is compiled providing complete documentation to be shipped with the plugin.

Update plugin metadata

QGIS uses specific metadata to register the plugin. At the time of writing the mechanism for registering this metadata is in transition from an in-source based system to an .ini file based system. In the interim, both should be maintained.

The in-source metadata is updated by editing the `__init__.py` file in the top level directory of the source tree:

```
def name():
   """A user friendly name for the plugin."""
   return '|project_name|'


def description():
    """A one line description for the plugin."""
    return 'InaSAFE Disaster risk assessment tool developed by AusAid and World Bank'


def version():
    """Version of the plugin."""
    return 'Version 0.1'


def icon():
    """Icon path for the plugin."""
    return 'icon.png'


def qgisMinimumVersion():
    """Minimum version of QGIS needed to run this plugin -
    currently set to 1.7."""
    return '1.7'
```

In general only the version function needs to be updated to reflect the new version of the InaSAFE plugin.

Note: The above will be deprecated with the release of QGIS 2.0, see below for the alternative method of describing the plugin.

For newer versions of QGIS (1.8+), the `metadata.txt` will be used to store descriptive information about the plugin. Simply edit this file with a text editor and update it as needed.

Outcome: The plugin metadata to reflects the current version of Risk in a Box.

Generate a test package

At this point a test package should be generated that can be used to test the plugin in a clean room environment. A clean room environment comprises a system that has a fresh operating system installation with the desired version of QGIS installed, and **no other software**. It is probably a good practice to use machine virtualisation for this purpose, for example with images of a windows and a linux system installed. Some virtualisation tools such as vmware provide the ability to create a system snapshot and roll back to it.

To generate a test package, use the `scripts/release.sh` bash script.

For exampled to create a test package for version 0.1.0 of the software, issue the following command:

```
scripts/release.sh 0.1.0
```

The generated package will be placed in the /tmp directory of your linux system.

Once the clean system is started, extract the package contents into the user's personal plugin directory. For example under Linux:

```
mkdir -p ~/.qgis/python/plugins
cd ~/.qgis/python/plugins
unzip inasafe.0.1.0.zip
```

Now start QGIS and enable the plugin in the QGIS plugin manager (*Plugins* → *Manage Plugins*).

Branch the release

This step is only done for minor and major releases, point releases are only tagged. The branch should be named after the major and minor version numbers only - for example: `version-1_0`. The following console log illustrates how to create a local branch, push it to the origin repository, remove the local branch and then track the repository version of the branch localy:

```
git branch version-0_1
git push origin version-0_1
git branch -D version-0_1
git fetch origin
git branch --track version-0_1 origin/version-0_1
git checkout version-0_1
```

Outcome: A branch on the remote repository named after the majon and minor version numbers.

Tag the release

Tagging the release provides a 'known good' state for the software which represents a point in time where all of the above items in this list have been checked. The tag should be named after the major, minor and point release for example `version-0_1_0`. If the release is a releas candidate or and alpha release the letters `rc` or `a` resepectively should be appended respectively, along with the related number. For example version 0.1.0 alpha 1 would be tagged as `version-0_1_0a1`. To tag the release simply do it in git as illustrated below.:

```
git tag version-0_1_0
git push --tags origin version-0_1_0
```

Note: Replace 'dot' separators with underscores for the version number.

Note: You can differentiate release **branches** from release **tags** by the fact that branch names have only the minor version number (e.g. version-0_4) whereas release tags are reserved for point releases (e.g. version-0_4_1).

Outcome: The release is tagged in GIT and can be checked out at any point in the future. The tagged source tree can easily be downloaded at any point by visiting https://github.com/AIFDR/inasafe/tags

Upload the package

QGIS provides an online plugin repository that centralizes the distribution and retrieval of plugins. It is the most efficient way to make your plugin available to the world at large.

- Upload the updated package zip file to old QGIS python plugin repository.
- Upload the updated package zip file to the new QGIS python plugin repository.

Press announcements

Once the release has been made, an announcement should be made to inform interested parties about the availability of the new software. A pro-forma announcement is provided below **(Trevor or Ole todo)**:

```
Dear |project_name| Users

We are pleased to announce the immediate availability of the newest
version of |project_name| (version X.X.X). This version includes numerous
bug fixes and improvements over the previous release::

----- changelog goes here -------------

We welcome any feedback you may have on this release. You can use our
issue tracker (requires free account) to notify us of any issues you may
have encountered whilst using the system. The tracker is available here:

https://github.com/AIFDR/inasafe/issues

This project is supported by the Australian Aid Agency and the World Bank.

Best regards

(Name of person)
```

A standard list of contacts should be compiled and the notification sent to all those listed.

Outcome: Interested parties are informed about the availability of the new release.

3.18 Continuous Integration Testing with Jenkins

Note: This documentation is Copyright Linfiniti Consulting CC. June 2012 and has been included here to provide provenance, and adapted for the InaSAFE project.

Jenkins is an automated build and test server.

Warning: These notes are a port of notes kept during setup of Jenkins for some django projects and were written after the fact. As such, they need to be proof read and validated on a new server instance.

3.18.1 Installation procedure - Jenkins server

The install procedure is described at http://pkg.jenkins-ci.org/debian/.

To start, do this:

```
wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -
```

Now add this to your `/etc/apt/sources.list`:

```
deb http://pkg.jenkins-ci.org/debian binary/
```

Now do:

```
sudo apt-get update
sudo apt-get install jenkins
```

DNS and Reverse Proxy

I set up the following subdomain http://jenkins.linfiniti.com

Now create a virtual host on the server's apache installation to reverse proxy by creating this file: `/etc/apache2/sites-available/3-jenkins.linfiniti.com` which should contain:

```
<VirtualHost *:80>
  ServerAdmin tim@linfiniti.com
  ServerName jenkins.linfiniti.com

  ProxyPass         / http://localhost:8080/
  ProxyPassReverse  / http://localhost:8080/
  ProxyRequests     Off

  # Local reverse proxy authorization override
  # Most unix distribution deny proxy by default
  # (ie /etc/apache2/mods-enabled/proxy.conf in Ubuntu)
  <Proxy http://localhost:8080/*>
    Order deny,allow
    Allow from all
  </Proxy>
  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  LogLevel warn

  ErrorLog /var/log/apache2/jenkins.linfiniti.error.log
  CustomLog /var/log/apache2/jenkins.linfiniti.access.log combined
  ServerSignature Off

</VirtualHost>
```

Now enable this vhost:

```
sudo a2ensite 3-jenkins.linfiniti.com
sudo /etc/init.d/apache reload
```

3.18.2 Configure the server

Go to http://jenkins.linfiniti.com and log in as an admin user.

Enable Plugins

Jenkins → *Manage Jenkins* → *Manage Plugins* or go to http://jenkins.linfiniti.com/pluginManager/.

Enable the following plugins:

- External Monitor Job Type Plugin
- LDAP Plugin
- pam-auth
- javadoc
- ant
- Jenkins Subversion Plug-in
- Jenkins GIT plugin
- Maven Integration plugin
- Jenkins SLOCCount Plug-in
- Jenkins Sounds plugin
- Jenkins Translation Assistance plugin
- ruby-runtime
- Jenkins CVS Plug-in
- Coverage/Complexity Scatter Plot PlugIn
- Status Monitor Plugin
- Git Parameter Plug-In
- github-api
- GitHub plugin
- Jenkins Violations plugin
- git-notes Plugin
- Twitter plugin
- Jenkins Cobertura Plugin
- Jenkins Gravatar plugin
- Jenkins SSH Slaves plugin
- xvfb

Lock down access

We are hosting private repositories on the server so we want to make everything private. To do that we do: *Jenkins* → *Manage Jenkins* → *Configure System* or go to http://jenkins.linfiniti.com/configure.

Under *Access control - Security Realm*, check *Jenkins's own user database*.

Under *Authorization* check *Project-based Matrix Authorization Strategy*, then add your admin users only here, giving them all permissions. We will add less priveledged users on a project by project basis.

Other Configuration options

- *Jenkins URL* set to `http://jenkins.linfiniti.com/`
- *Sender E-mail Address* set to `Linfiniti Jenkins <jenkins@linfiniti.com>`
- *GitHub Web Hook* set to `Manually manage hook URLs`

3.18.3 Creating the Jenkins Farminfo Job

Create a new Job (project):

Jenkins → *New Job* or go to http://jenkins.linfiniti.com/view/All/newJob.

Here is a log of the options we set for the Jenkins job:

- *Project Name* set to `InaSAFE` (don't use spaces for the project name!).
- Check *Enable project-based security* then add an entry for **Anonymous** with `Job: Read` and `Job Discover` permissions only. This will allow anonymous read-only access to the project.
- *Xvfb* set to ticked.
- *GitHub project* set to `https://github.com/AIFDR/inasafe/`
- *Source Code Management* check *Git* and set to `git@github.com:AIFDR/inasafe.git`
- *Branch Specifier (blank for default)* set to `master`
- *Repository browser* set to `auto`
- *Build Triggers* set to `Build when a change is pushed to GitHub` - this will cause the project to build every time a new commit is pushed to GitHub. We also need to setup a GitHub hook to do this which is described further down.

Build Actions

- *Build* add an `Execute Shell` step and set the script as follows:

```
export PYTHONPATH=/usr/local/qgis1.8/share/qgis/python/
export LD_LIBRARY_PATH=/usr/local/qgis1.8/lib
export QGIS_PREFIX_PATH=/usr/local/qgis1.8/

# Make sure data dir is current and synced it its git clone
scripts/update-test-data.sh

#Go on with metrics and tests
make jenkins-pyflakes
make jenkins-pep8
make jenkins-pylint
make jenkins-sloccount
make jenkins-test
```

Note: In this server instance we are using a hand built QGIS. The initial 3 export lines may not be needed if you are using a package build QGIS installation.

- *Add build step* set to `play a sound` and choose a sound which will play when the build completes (optional)

Post-build Actions

Add the following actions and settings:

- *Publish Cobertura Coverage Report* set *Cobertura xml report pattern* to `coverage.xml`. For the rest take the defaults or tweak them as you like.
- ***Publish JUnit test result report* set *Test report XMLs*** to `nosetests.xml`. I also disabled *Retain long standard output/error* (you can enable it temporarily while debugging server side build issues).
- *Publish SLOCCount analysis reports* set *SLOCcount reports* to `sloccount.sc`
- ***Report Violations* set the following:**
  - *pep8* to `pep8.log`
  - *pylint* to `pylint.log`
- *Email notification* set *Recipients* to `tim@linfiniti.com`. Add any additional recipients as needed (space delimited). Also I set *Send e-mail for every unstable build* checked and *Send separate e-mails to individuals who broke the build* checked.
- *Jenkins Sounds* set *Failure* to `argh(wave)`
- *Jenkins Sounds* set *Success* to `Yahoo.Yodel(wave)`
- *Publish Coverage / Complexity Scatter Plot* check *Locate the graph at the topmost of Jenkins project page*
- *Status Monitor* enable (this is optional)

Click *Save*

3.18.4 Server Configuration in the shell

We need to do the following on the server as the jenkins user:

Create an ssh key for Jenkins

This key will be used to enable Jenkins access to private repos on GitHub. You only need to do this once on the server for each project:

```
sudo su - jenkins
jenkins@maps:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa_inasafe):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa_inasafe.
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa_inasafe.pub.
The key fingerprint is:
29:7e:b6:f5:18:5e:cb:e7:f6:b7:84:e1:f5:66:31:41 jenkins@maps.linfiniti.com
The key's randomart image is:
+--[ RSA 2048]----+
|               E |
|              .  |
|               . |
|         .      .|
|      . S    . + |
|     . .    . + +|
|      . o o .o .+|
|       o + * .+o.|
|        . o ++.o+|
+++++++++++++++++++
```

Here's our key:

```
cat .ssh/id_rsa_inasafe.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC5f7il/lmIUQeIrcQ3f10krOOjLSwPhpJB5G6T
LG1Dtxoyb/o/XgxPOjNfVfoVNLhB7AVp6I/IsrK3KvO8wkuOVWQ5Q5p3ntPnT6eX62JTLAlPeOGo
MX0Iq1Vs6cOAhNK11uMfXwdUk//cht3zSlb6GLeg7mTNw/JGPAzJV4YkCcIK87H3e1ClYEU+7Kzb
lbGKBfXWxgUPIHVRfwVZJZwbCgt1hBSBs7nJFmLqC654rRxhpFAWi72/Go79AW+YEDWUOOSZEsGc
aT0kJ4mrks0nOMwAhcimcFqfdmmwOqIXgLPm/1jG78z08zGYhNiLOcaZDj1ULnQNmAtDx0/rCOuL
jenkins@maps.linfiniti.com
```

You need to create an ssh host alias so that when doing a git checkout, the correct keypair is used. Add this to the `~/.ssh/config` of the jenkins user:

```
Host InaSAFEGitHub
     IdentityFile /home/jenkins/.ssh/id_rsa_inasafe
     HostName github.com
```

Now go to https://github.com/AIFDR/inasafe/admin/keys and add the above key as a deploy key.

Next we add the github key to the Jenkins user's allowed hosts. The easiest way to do this is just to make a temporary clone of the repo. Again this only needs to be done once and then Jenkins will work for all github projects.

Back on the server:

```
jenkins@maps:~$ cd /tmp/
jenkins@maps:/tmp$ git clone git@InaSAFEGitHub:timlinux/inasafe.git
Cloning into inasafe...
The authenticity of host 'github.com (207.97.227.239)' can't be established.
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
```

Also you should set the Jenkins git user and email:

```
jenkins@maps:~$ git config --global user.email "jenkins@linfiniti.com"
jenkins@maps:~$ git config --global user.name "Jenkins Build Server @Linfiniti2"
```

Note: The source tree will be in `~/jobs/InaSAFE/workspace/`.

Project setup

Try to do an intial build. It will fail because we have to copy the InaSAFE test data into the job directory. Use the same procedure as above to add ad deploy key to the inasafe-data repo and then clone it (as jenkins user):

```
cd ~/jobs/InaSAFE/
git clone git@InaSAFEDataGitHub:timlinux/inasafe-data.git
```

GitHub Hook Setup

In GitHub hooks, enable Jenkins (GitHub plugin) and set the url to:

```
http://jenkins.linfiniti.com/github-webhook/
```

Testing on Jenkins server

Simply commit something and push it to the server. Turn the speakers on on your desktop with the Jenkins page (http://jenkins.linfiniti.com) open and you should hear a cheer or a groan depending on whether the test passes or fails.

3.18.5 Further Reading

Here are some useful resources we found while getting things set up.

- http://www.alexconrad.org/2011/10/jenkins-and-python.html
- http://hustoknow.blogspot.com/2011/02/setting-up-django-nose-on-hudson.html
- http://blog.jvc26.org/2011/06/13/jenkins-ci-and-django-howto
- https://sites.google.com/site/kmmbvnr/home/django-jenkins-tutorial (probably the most usful article I found)
- http://blog.mathieu-leplatre.info/django-et-jenkins-fr.html (in French)

3.19 IRC Notes for InaSAFE

The IRC channel is #inasafe on freenode.

The channel is logged, you can read the logs at http://irclogs.geoapt.com/inasafe/.

Tim (timlinux) has ops and has registered the channel:

```
/query chanserv
register #inasafe
```

To obtain ops after logging off do:

```
/query chanserv
op #inasafe timlinux
```

To give others op status in the channel do:

```
/op <user>
```

There is a github hook enabled 'IRC" https://github.com/AIFDR/inasafe/admin/hooks with the following settings:

- **server** : irc.freenode.net
- **port** : 6667
- **room** : inasafe
- **nick** : gh-inasafe
- **branch regexes** : (leave blank)
- **password** : (leave blank)
- **ssl** : no

- **message without join** : yes
- **no colors** : no
- **Long URL** : yes
- **Notice** : yes
- **Active** : yes

Also you need to do a mode change on the chanserv channel to enable outsider messages:

```
/query chanserv
set #inasafe mlock -n
```

3.20 Frequently Asked Questions

3.20.1 How does the documentation work?

The InaSAFE documentation files are written using the RST format (quickreference guide) and stored with the source code in github:

```
https://github.com/AIFDR/inasafe/tree/master/docs/source
```

The RST files are used for two products:

- HTML files generated using Sphinx (http://sphinx.pocoo.org) by running https://github.com/AIFDR/inasafe/blob/master/docs/Makefile. These files are accessible through both the file browser and the help button available in InaSAFE
- The web site http://readthedocs.org/docs/inasafe which automatically reads the RST files from github to update its content. The steps to achieve this are

1. Register the project on the dashboard at ReadTheDocs (http://readthedocs.org/dashboard/inasafe/edit). In particular, this form points to the github repository where the RST files reside.
2. Either manually build the project by clicking 'Build latest version' on http://readthedocs.org/dashboard/inasafe/ or by activating the service hook for ReadTheDocs at github: https://github.com/AIFDR/inasafe/admin/hooks

3.20.2 How do I replace a string across multiple files

To replace string layer_type, say, with layertype across all python files in project, do:

```
find . -name "*.py" -print | xargs sed -i 's/layer_type/layertype/g'
```

Alternative you can install the 'rpl' command line tool:

```
sudo apt-get install rpl
```

Using rpl is much simpler, just do:

```
rpl "oldstring" "newstring" *.py
```

For details on the find command see this article.

3.20.3 How did you embed the git version SHA1 into each .py file?

The format was derived using the git log format tag. It is stored in the source of each python as:

```
__revision__ = '$Format:%H$'
```

'%H' being the format tag for the SHA1. The __revision__ is **not** updated with each commit. Rather is is registered with git for replacement when using git-archive by doing this:

```
echo "*.py export-subst" > .gitattributes
git add .gitattributes
```

The above only needs to be done once and then all python files with format substitutions will be replaced when running git-archive. The actual substition takes place at the time that a git archive is generated (git archive creates a copy of the repo with all repository metadata stripped out). For example:

```
git archive version-0_3 | tar -x -C /tmp/inasafe-0.3.0
```

You can verify SHA1 replacement has been made by doing:

```
cat /tmp/inasafe/gui/is_plugin.py | grep revision
__revision__ = 'a515345e43b25d065e1ae0d73687c13531ea4c9c'
```

The deployment of version tagged files is automated by using the `scriptsrelease.sh` script.

3.20.4 How do you profile code?

sudo apt-get install python-profiler python -m cProfile -s time safe/common/test_polygon.py

and

sudo easy_install pycallgraph sudo apt-get install graphviz pycallgraph safe/common/test_polygon.py

See also http://stackoverflow.com/questions/582336/how-can-you-profile-a-python-script

3.20.5 Why do I got 1e+24 (or another big number) instead of NaN when I saved vector files?

InaSAFE needs to convert NaN to 1e+24 or more specifically 24 digits of 9 because NaN is not intepreted correclty on Windows. You can read more information about it on http://trac.osgeo.org/gdal/ticket/4799 or in Github issue #269.

But don't worry, when InaSAFE read vector file, it will convert the 1e+24 back to NaN. So, any computation inside InaSAFE will treat NaN as NaN in numpy.

Note: You'll need to convert the 1e+24 to NaN value manually if you want to use it without InaSAFE.

3.21 Todo List

Todo

Get this working...

Todo

Get this working...

(The *original entry* is located in /var/build/user_builds/inasafe/checkouts/latest/docs/source/developer-docs/todo.rst, line 4.)

3.22 InaSAFE Realtime

InaSAFE Realtime is a component of the InaSAFE project designed for deployment on a server and creation of Impact Maps a short interval after an event occurs.

Currently the realtime system supports Earthquake fatality impact assessments, though in future we envisage additional support for other disaster types being facilitated.

Note: This work was funded by the Australia-Indonesia Facility for Disaster Reduction, Geoscience Australia and the GFDRR. We thank you for your support.

3.22.1 Historical Note

The original prototype of the realtime system was implemented by Ole Nielsen (AusAID). The subsequent port of the realtime system to InaSAFE was implemented by Tim Sutton (Linfiniti Consulting CC., funded by The World Bank and the AIFDR).

3.22.2 Supported Platforms

Currently only Ubuntu 12.04 is supported. The software may work or can easily be made to work on other platforms but it is untested.

3.22.3 Generated Products

For every shake event, the tool produces a number of GIS products:

- **A raster layer interpolated from the original MMI point matrix and symbolized** according to the MMI scale colours.
- **A vector (shapefile) layer generated from the original MMI point matrix and** symbolized according to the MMI scale colours.
- **A vector (shapefile) layer depicting MMI isolines and symbolized according to** the MMI scale colours.
- **A cities layer (shapefile) which lists the affected cities along with key** data such as distance from and direction to epicenter, number of people resident in the city, mmi exposure etc.

In addition to the above mentioned GIS products, the following 3 files are created for each event:

- A PNG file containing a single page report as illustrated above.
- **A large PNG image which contains exactly the same content as the pdf but in** an image format.
- A thumbnail PNG image which contains a reduced size image of the report.

3.22.4 Architecture

InaSAFE Realtime is implemented by four main python modules:

- **ftp_client - A generic tool to fetch directory listings and** files from a remote server.
- **shake_data - A mostly generic tool to fetch shake files from** an ftp server. There is an expectation that the server layout follows a simple flat structure where files are named after the shake event and are in the format of shake data as provided by the USGS (XXXXXX TODO fact check XXXX). `ftp://118.97.83.243/20110413170148.inp.zip`
- **shake_event - A rather monolithic module that 'knows' how to** fetch, unpack, process and generate a report for a quake event. The module logic is based on the standard shake data packaging format supplied by the USGS. We have restricted out implementation to require only the `grid.xml` file contained in the inp.zip file in the downloaded zip file.
- **make_map - A simple python tool for running one or multiple shake** analyses.

InaSAFE has strong dependencies on QGIS (http://qgis.org) which is used for much of the data processing and reporting functionality.

Note: Currently version c064933677d48d90776bdea0426fdea1e3490d01 is the 'known good' SHA1.

Two of these dependencies is a template QGIS project and a map composition template. We have designed the realtime reporting engine to allow end users to customise the map report to their needs with little or no programming. The primary way to achieve this is by opening the custom template `realtime/fixtures/realtime-template.qpt` in QGIS and modifying its contents. You could also build a new template from scratch provided the item IDs listed in the section that follows are used.

3.22.5 Installation

The supported platform is currently Ubuntu 12.04 LTS. The instructions provided below are for that OS. First we are going to hand build QGIS. This may not be needed in future once 2.0 packages are available, but for now it is recommended.:

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt-get update
sudo apt-get build-dep qgis

cd ~
mkdir -p dev/cpp
mkdir -p dev/python
sudo mkdir /home/web
sudo chown <youruser>.<youruser> /home/web
cd ~/dev/cpp
sudo apt-get install git cmake-curses-gui
git clone git@github.com:qgis/Quantum-GIS.git
```

To run the realtime component of InaSAFE, you need to have QGIS 2.0 and the QGIS 2.0 python bindings installed. In addition the `pytz` module is needed:

```
sudo apt-get install python-tz
```

You also need to have the standard datasets needed for the cartography:

- population

- indonesia.sqlite (can be changed by adjusting the QGIS project).

3.22.6 QGIS Map Template Elements

This section describes the various elements that comprise the standard map template, and which you can modify directly in the template. These fall into three groups:

- **Static elements**.
- **Elements containing tokens for replacement**.
- **Elements that are directly updated by the renderer**.

Static Elements

These are e.g. logos which are not touched by the realtime map renderer at all. You can remove or replace them with your own elements as needed.

- **logo-left** - the logo element at the top left corner of the map layout.
- **right-logo** - the logo element at the top right corner of the map layout.
- **overview-map - a map overview showing the locality of the event. This** is the overview frame for map-0 (the main map in the layout). It is locked and limited to show the population layer only.
- **legend - a map legend, by default configured to show only the layer for** the population layer. It is locked and limited to the population layer.

Elements containing tokens for replacement

In this case the element name is not significant, only the token(s) it contains. At render time any of the tokens in these elements will be replaced with translated (if an alternative locale is in effect) content from the map renderer according to the keywords listed below in this document.

- **main-title - the main title at the top of the page. By default this** element contains the keyword: `[map-name]`.
- **intensity-date - the date and intensity of the event. By default this** label contains the following replacement tokens: `M[mmi] [date] [time]`
- **position-depth - the position (lon, lat) and depth of the event. By** default this label contains the following replacement tokens: `[longitude-name] [longitude-value] [latitude-name] [latitude-value] [depth-name] [depth-value] [depth-unit]`
- **location-description - the postion of the event described relative to** the nearest major populated place. By default this label contains the following replacement tokens: `[located-label] [distance] [distance-unit], [bearing-degrees] [bearing-compass] [direction-relation] [place-name]`
- **elapsed-time - the time elapsed between the event and when this report** was generated. By default this label contains the following replacement tokens: `[elapsed-time-label] [elapsed-time]`
- **scalebar - the scalebar which reflects the scale of the main map.** This is **Currently disabled**.
- **disclaimer - A block of text for displaying caveats, cautionary notes,** interpretive information and so on. This contains the following replacement tokens: `[limitations]`.
- **credits - A block of text for displaying credits on the map output.** This contains the following replacement tokens: `[credits]`.

Elements that are directly updated by the renderer

In this case any content that may be present in the element is completely replaced by the realtime map renderer, although certain styling options (e.g. graticule settings on the map) will remain in effect.

- **impacts-table - a table generated by ShakeEvent which will list the** number of modelled affected people in each of the MMI bands. This is an HTML element and output will fail if it is not present.
- **main-map - primary map used to display the event and neighbouring towns.** Developers can set a minimum number of neighbouring towns to display using the ShakeEvent api. This is a map element and output will fail if it is not present. This is an HTML element and output will fail if it is not present.
- **affected-cities - a table generated by ShakeEvent which will list the** closes N cities (configurable using the ShakeEvent api) listed in order of shake intensity then number of people likely to be affected.

3.22.7 Replaceable Keywords

This section describes tokenised keywords that are passed to the map template. To insert any of these keywords into the map template, simply enclose the key in [] (e.g. [place-name]) and it will be replaced by the text value (e.g. Tondano). The list includes static phrases which have been internationalised (and so will display in the language of the selected map local, defaulting to English where no translation if available. In cases where static definitions are used (e.g. [credits]) you can substitute your own definitions by creating your own template. More on that below in the next section.

- **map-name**: Estimated Earthquake Impact
- **exposure-table-name**: Estimated number of people exposed to each MMI level
- **city-table-name**: Places Affected
- **legend-name**: Population density
- **limitations**: This impact estimation is automatically generated and only takes into account the population and cities affected by different levels of ground shaking. The estimate is based on ground shaking data from BMKG, population density data from asiapop .org, place information from geonames.org and software developed by BNPB. Limitations in the estimates of ground shaking, population data and place names datasets may result in significant misrepresentation of the on-the-ground situation in the figures shown here. Consequently decisions should not be made solely on the information presented here and should always be verified by ground truthing and other reliable information sources.
- **credits**: Supported by the Australia-Indonesia Facility for Disaster Reduction and Geoscience Australia.
- **place-name**: Tondano
- **depth-name**: Depth
- **location-info**: M 5.0 26-7-2012 2:15:35 Latitude: 12 ‘36.00”S Longitude: 124‘27‘0.00”E Depth: 11.0km Located 2.50km SSW of Tondano
- **depth-unit**: km
- **bearing-compass**: SSW
- **distance-unit**: km
- **mmi**: 5.0
- **longitude-name**: Longitude
- **date**: 26-7-2012
- **time**: 2:15:35

- **formatted-date-time: 26-Jul-12 02:15:35
- **located-label**: Located
- **bearing-degrees**: -163.055923462
- **distance**: 2.50
- **direction-relation**: of
- **latitude-name**: Latitude
- **latitude-value**: 12‘36.00”S
- **longitude-value**: 12‘4‘27.00
- **depth-value**: 11.0
- **version**: Version: 1.0.1
- **bearing-text**: bearing
- **elapsed-time-name**: Elapsed time
- **elapsed-time**: 26-Jul-12 02:15:35
- **fatality-name**: Estimated Fatalities
- **fatality-count**: 55

3.22.8 Customising the template

You have a few options to customise the template - we have gone to great lengths to ensure that you can flexibly adjust the report composition **without doing any programming**. There are three primary ways you can achieve this:

- Moving replacement tags into different elements, or removing them completely.
- **Moving the template elements themselves around or adding / removing them** completely.
- **Creating your own template from scratch and pointing the realtime tool to** your preferred template.

The template is provided as `realtime/fixtures/realtime-template.qpt` and can be modified by opening the template using the QGIS map composer, making your changes and then overwriting the template. You should take care to test your template changes before deploying them to a live server, and after deploying them to a live server.

If you wish to use your own custom template, you need to specify the `INSAFE_REALTIME_TEMPLATE` environment variable, populating it with the path to your preferred template file.

3.22.9 QGIS Realtime Project

The cartography provided in the realtime maps is loaded from the `realtime/fixtures/realtime.qgs` QGIS project file. You can open this file using QGIS, change the layers and their symbology, and your changes will be reflected in the generated realtime shake report.

There are however some caveats to this:

- The overview map has locked layers
- The main map should always have a population layer with grayscale legend matching that provided in the original. If you do remove / change the population layer you should also remove / change the population layer legend.

If you wish to use your own custom project, you need to specify the `INSAFE_REALTIME_PROJECT` environment variable, populating it with the path to your preferred project file.

3.22.10 Configuration of population data

Population data is used as the 'exposure' dataset for shake reports. The following priority will be used to determine the path of the population raster dataset. # the class attribute **self.populationRasterPath**

will be checked and if not None it will be used.

# the environment variable `INASAFE_POPULATION_PATH` will be checked if set it will be used.

# A hard coded path of `/fixtures/exposure/population.tif` will be checked.

# A hard coded path of `/usr/local/share/inasafe/exposure/population.tif` will be used.

3.22.11 Running a shake event

To run a single event locally on a system with an X-Server you can use the provided script `scripts/make-shakemap.sh`. The script can be used with the following options:

- **–list: `scripts/make-shakemap.sh --list` - retrieve a list of** all known shake events on the server. Events are listed as their full ftp url e.g. `ftp://118.97.83.243/20121106084105.out.zip` and both *inp* and *out* files are listed.
- **[event id]: `scripts/make-shakemap.sh 20121106084105` - retrieve** and process a single shake event. A pdf, png and thumbnail will be produced.
- **–all: `scripts/make-shakemap.sh --all` - process all identified** events on the server in batch mode. **Note:** this is experimental and not production ready - we recommend to use the approach described in *Batch validation & running*.
- **no parameters: `scripts/make-shakemap.sh` - fetch and process** the latest existing shake dataset. This is typically what you would want to use as the target of a cron job.

Note: The `make_shakemap.sh` script is just a thin wrapper around the python `realtime.make_map` python module.

3.22.12 Unit tests

A complete set of unit tests is provided with the realtime package for InaSAFE. You can execute these tests like this:

```
nosetests -v --with-id --with-xcoverage --with-xunit --verbose --cover-package=realtime realtime
```

There are also a number of Jenkins tasks provided in the Makefile for InaSAFE to automate testing on our continuous integration server. You can view the current state of these tests by visiting this URL:

http://jenkins.linfiniti.com/job/InaSAFE-Realtime/

3.22.13 Batch validation & running

The `scripts/make-all-shakemaps.sh` provided in the InaSAFE source tree will automate the production of one shakemap report per event found on the shake ftp server. It contains a number of environment variable settings which can be used to control batch execution. First a complete script listing:

```
#!/bin/bash

export QGIS_DEBUG=0
export QGIS_LOG_FILE=/tmp/inasafe/realtime/logs/qgis.log
export QGIS_DEBUG_FILE=/tmp/inasafe/realtime/logs/qgis-debug.log
export QGIS_PREFIX_PATH=/usr/local/qgis-master/
export PYTHONPATH=/usr/local/qgis-master/share/qgis/python/:`pwd`
export LD_LIBRARY_PATH=/usr/local/qgis-master/lib
export INASAFE_WORK_DIR=/home/web/quake
export SAFE_POPULATION_PATH=/var/lib/jenkins/jobs/InaSAFE-Realtime/exposure/population.tif
for FILE in `xvfb-run -a --server-args="-screen 0, 1024x768x24" python realtime/make_map.py --list |
do
    FILE=`echo $FILE | sed 's/ftp:\/\/118.97.83.243\///g'`
    FILE=`echo $FILE | sed 's/.out.zip//g'`
    echo "Running: $FILE"
    xvfb-run -a --server-args="-screen 0, 1024x768x24" python realtime/make_map.py $FILE
done
exit
```

An example of the output produced from such a batch run is provided at:

http://quake.linfiniti.com/

3.22.14 Hosting the shakemaps

In this section we describe how to easily host the shakemaps on a public web site.

An apache configuration file and a set of resources are provided to make it easy to host the shakemap outputs. The resources provided can easily be modified to provide a pleasing, user friendly directory listing of shakemap reports.

Note: You should adapt the paths used below to match the configuration of your system.

First create a file (as root / sudo) with this content in your `/etc/apache2/sites-available/quake-apache.conf`. for example:

```
<VirtualHost *:80>
  ServerAdmin tim@linfiniti.com
  ServerName quake.linfiniti.com

  DocumentRoot /home/web/quake/public/
  <Directory /home/web/quake/public/>
    Options Indexes FollowSymLinks
    IndexOptions +FancyIndexing
    IndexOptions +FoldersFirst
    IndexOptions +XHTML
    IndexOptions +HTMLTable
    IndexOptions +SuppressRules
    HeaderName resource/header.html
    ReadmeName resource/footer.html
    IndexStyleSheet "resource/bootstrap.css"
    IndexIgnore .htaccess /resource
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>

  ErrorLog /var/log/apache2/quake.linfiniti.error.log
  CustomLog /var/log/apache2/quake.linfiniti.access.log combined
  ServerSignature Off

</VirtualHost>
```

Now make the `/home/web/quake/public` directory in which the outputs will be hosted:

```
mkdir -p /home/web/quake/public
```

Unpack the `realtime/fixtures/web/resource` directory into the above mentioned public directory. For example:

```
cd /home/web/quake/public
cp -r ~/dev/python/inasafe/realtime/fixtures/web/resource .
```

Next ensure that apache has read access to your hosting directory:

```
chmod +X /home/web/quake/public
chmod +X /home/web/quake/public/resource
```

You can customise the look and feel of the hosted site by editing the files in `/home/web/quake/public/resource` (assumes basic knowledge of HTML).

Lastly, you should regularly run a script to move generated pdf and png outputs into the public directory. An example of such a script is provided as `realtime/fixtures/web/make-public.sh`. To run this script regularly, you could add it to a cron job e.g.:

```
crontab -e
```

And then add a line like this to the cron file:

```
* * * * * /home/timlinux/dev/python/inasafe-realtime/realtime/fixtures/web/make-public.sh
```

Note: The resources used in the above examples are all available in the source code under `realtime/fixtures/web`.

---

http://paradox460.newsvine.com/_news/2008/04/05/1413490-how2-stylish-apache-directory-listings

CHAPTER

FOUR

INASAFE'S API DOCUMENTATION

This is the API documentation for the InaSAFE project. You can find out more about the InaSAFE project by visiting **'riskinabox.org<http://www.inasafe.org/>'_**.

4.1 Packages safe_qgis

4.1.1 Package::safe_qgis

Module: safe_qgis.plugin

This module forms part of the InaSAFE tool.

Module: safe_qgis.dock_base

This module forms part of the InaSAFE tool.

Module: safe_qgis.impact_calculator

This module forms part of the InaSAFE tool.

Module: safe_qgis.utilities

This module forms part of the InaSAFE tool.

Module: safe_qgis.keyword_io

This module forms part of the InaSAFE tool.

Module: safe_qgis.safe_interface

This module forms part of the InaSAFE tool.

Module: safe_qgis.function_options_dialog

This module forms part of the InaSAFE tool.

Module: safe_qgis.html_renderer

This module forms part of the InaSAFE tool.

Module: safe_qgis.resources_rc

This module forms part of the InaSAFE tool.

Module: safe_qgis.function_options_dialog_base

This module forms part of the InaSAFE tool.

Module: safe_qgis.keywords_dialog_base

This module forms part of the InaSAFE tool.

Module: safe_qgis.map_legend

This module forms part of the InaSAFE tool.

Module: safe_qgis.qgis_interface

This module forms part of the InaSAFE tool.

Module: safe_qgis.impact_calculator_thread

This module forms part of the InaSAFE tool.

Module: safe_qgis.map

This module forms part of the InaSAFE tool.

Module: safe_qgis.clipper

This module forms part of the InaSAFE tool.

Module: safe_qgis.help

This module forms part of the InaSAFE tool.

Module: safe_qgis.options_dialog

This module forms part of the InaSAFE tool.

Module: safe_qgis.impact_functions_doc

This module forms part of the InaSAFE tool.

Module: safe_qgis.exceptions

This module forms part of the InaSAFE tool.

Module: safe_qgis.utilities_test

This module forms part of the InaSAFE tool.

Module: safe_qgis.keywords_dialog

This module forms part of the InaSAFE tool.

Module: safe_qgis.resources

This module forms part of the InaSAFE tool.

Module: safe_qgis.impact_functions_doc_base

This module forms part of the InaSAFE tool.

Module: safe_qgis.dock

This module forms part of the InaSAFE tool.

Module: safe_qgis.options_dialog_base

This module forms part of the InaSAFE tool.

4.2 Unit Tests

4.2.1 Package::safe_qgis_tests

Module: safe_qgis.test_map_legend

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_function_options_dialog

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_clipper

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_keyword_io

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_dock

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_utilities

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_init

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_impact_functions_doc

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_help

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_safe_translations

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_safe_interface

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_plugin

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_html_renderer

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_map

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_impact_calculator

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_keywords_dialog

This module forms part of the InaSAFE tool.

Module: safe_qgis.test_qgis_environment

This module forms part of the InaSAFE tool.

4.3 Packages safe

4.3.1 Package::safe

Module: safe.api

This module forms part of the InaSAFE tool.

Module: safe.defaults

SAFE (Scenario Assessment For Emergencies) - API

The purpose of the module is to provide a well defined public API for the packages that constitute the SAFE engine. Modules using SAFE should only need to import functions from here.

Contact : ole.moller.nielsen@gmail.com

This module forms part of the InaSAFE tool.

Package::safe.engine

Module: safe.engine.test_engine

This module forms part of the InaSAFE tool.

Module: safe.engine.interpolation

This module forms part of the InaSAFE tool.

Module: safe.engine.utilities

Miscellaneous utility functions for Risk-in-a-Box (riab_core)

This module forms part of the InaSAFE tool.

Module: safe.engine.core

This module forms part of the InaSAFE tool.

Package::safe.engine.impact_functions_for_testing

Module: safe.engine.impact_functions_for_testing.flood_road_impact This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.itb_fatality_model_org This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.error_raising_functions This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.unspecific_building_impact_model This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.allen_fatality_model This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.NEXIS_building_impact_model This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.BNPB_earthquake_guidelines This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.general_ashload_impact This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.empirical_fatality_model This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.itb_fatality_model_configurable This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.HKV_flood_study This module forms part of the InaSAFE tool.

Module: safe.engine.impact_functions_for_testing.earthquake_impact_on_women This module forms part of the InaSAFE tool.

Package::safe.postprocessors

Module: safe.postprocessors.age_postprocessor

Postprocessors package.

class `safe.postprocessors.age_postprocessor.`**`AgePostprocessor`**

Postprocessor that calculates age related statistics. see the _calculate_* methods to see indicator specific documentation

see `safe.defaults` for default values information

`clear``()`

concrete implementation it takes care of the needed parameters being properly cleared

Args: None

Returns: None

Raises: None

`process``()`

concrete implementation it takes care of the needed parameters being available and performs all the indicators calculations

Args: None

Returns: None

Raises: None

`setup``(`*params*`)`

concrete implementation it takes care of the needed parameters being initialized

Args: params: dict of parameters to pass to the post processor

Returns: None

Raises: None

This module forms part of the InaSAFE tool.

Module: safe.postprocessors.gender_postprocessor

Postprocessors package.

class `safe.postprocessors.gender_postprocessor.`**`GenderPostprocessor`**

Postprocessor that calculates gender related statistics. see the _calculate_* methods to see indicator specific documentation

see `safe.defaults` for default values information

`clear``()`

concrete implementation it takes care of the needed parameters being properly cleared

Args: None

Returns: None

Raises: None

`process``()`

concrete implementation it takes care of the needed parameters being available and performs all the indicators calculations

Args: None

Returns: None

Raises: None

`setup` (*params*)
concrete implementation it takes care of the needed parameters being initialized

Args: params: Dict of parameters to pass to the post processor

Returns: None

Raises: None

This module forms part of the InaSAFE tool.

Module: safe.postprocessors.postprocessor_factory

Postprocessors package.

Tip: import like this from safe.postprocessors import get_post_processors and then call get_post_processors(requested_postprocessors)

`safe.postprocessors.postprocessor_factory.get_post_processors` (*requested_postprocessors*)
Creates a dictionary of applicable postprocessor instances

Args:

- **requested_postprocessors: dictionary of requested** postprocessors such as { 'Gender': {'on': True}, 'Age': {'on': True,

  'params': { 'youth_ratio': defaults['YOUTH_RATIO'], 'adult_ratio': defaults['ADULT_RATIO'], 'elder_ratio': defaults['ELDER_RATIO'] }

  }

  } with 'PostprocessorName': {'on': True} being the minimum needed to activate a postprocessor. If asked for unimplemented postprocessors, the factory will just skip it returning the valid ones

Returns:

dict of postprocessors instances e.g. {'Gender':GenderPostprocessors instance}

`safe.postprocessors.postprocessor_factory.get_postprocessor_human_name` (*postprocesor*)
Returns the human readable name of post processor

Args:

- postprocessor: Machine name of the postprocessor

Returns: str with the human readable name

This module forms part of the InaSAFE tool.

Module: safe.postprocessors.abstract_postprocessor

Abstract postprocessor class, do not instantiate directly

class `safe.postprocessors.abstract_postprocessor.`**`AbstractPostprocessor`**

Abstract postprocessor class, do not instantiate directly. but instantiate the PostprocessorFactory class which will take care of setting up many prostprocessors. Alternatively you can as well instantiate directly a sub class of AbstractPostprocessor.

Each subclass has to overload the process method and call its parent like this: AbstractPostprocessor.process(self) if a postprocessor needs parmeters, then it should override the setup and clear methods as well and call respectively AbstractPostprocessor.setup(self) and AbstractPostprocessor.clear(self).

for implementation examples see AgePostprocessor which uses mandatory and optional parameters

`clear``()`

Abstract method to be called from the concrete implementation with AbstractPostprocessor.process(self) it takes care of results being cleared

Args: None

Returns: None

Raises: None

`process``()`

Abstract method to be called from the concrete implementation with AbstractPostprocessor.process(self) it takes care of results being initialized

Args: None

Returns: None

Raises: None

`results``()`

Returns the postprocessors results

Args: None

Returns: Odict of results

Raises: None

`setup``(`*params*`)`

Abstract method to be called from the concrete implementation with AbstractPostprocessor.setup(self, None) it takes care of results being initialized

Args: params: dict of parameters to pass to the post processor

Returns: None

Raises: None

This module forms part of the InaSAFE tool.

Package::safe.impact_functions

Module: safe.impact_functions.test_real_impact_functions

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.test_plugin_core

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.test_mappings

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.utilities

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.mappings

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.test_plugins

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.core

This module forms part of the InaSAFE tool.

Module: safe.impact_functions.styles

This module forms part of the InaSAFE tool.

Package::safe.impact_functions.tephra

Package::safe.impact_functions.generic

Module: safe.impact_functions.generic.categorised_hazard_building_impact This module forms part of the InaSAFE tool.

Module: safe.impact_functions.generic.categorised_hazard_population This module forms part of the InaSAFE tool.

Package::safe.impact_functions.volcanic

Module: safe.impact_functions.volcanic.volcano_building_impact This module forms part of the InaSAFE tool.

Module: safe.impact_functions.volcanic.volcano_population_evacuation_polygon_hazard This module forms part of the InaSAFE tool.

Package::safe.impact_functions.earthquake

Module: safe.impact_functions.earthquake.padang_building_impact_model This module forms part of the InaSAFE tool.

Module: safe.impact_functions.earthquake.itb_earthquake_fatality_model This module forms part of the InaSAFE tool.

Module: safe.impact_functions.earthquake.earthquake_building_impact This module forms part of the InaSAFE tool.

Module: safe.impact_functions.earthquake.itb_building_impact_model This module forms part of the InaSAFE tool.

Package::safe.impact_functions.inundation

Module: safe.impact_functions.inundation.flood_road_impact_experimental This module forms part of the InaSAFE tool.

Module: safe.impact_functions.inundation.flood_OSM_building_impact This module forms part of the InaSAFE tool.

Module: safe.impact_functions.inundation.flood_population_evacuation_polygon_hazard This module forms part of the InaSAFE tool.

Module: safe.impact_functions.inundation.flood_population_evacuation This module forms part of the InaSAFE tool.

Package::safe.common

Module: safe.common.test_geodesy

This module forms part of the InaSAFE tool.

Module: safe.common.version

`safe.common.version.`**`get_git_changeset`**`()`

Returns a numeric identifier of the latest git changeset.

The result is the UTC timestamp of the changeset in YYYYMMDDHHMMSS format. This value isn't guaranteed to be unique, but collisions are very unlikely, so it's sufficient for generating the development version numbers.

`safe.common.version.`**`get_version`**`(`*version=None*`)`

Returns a PEP 386-compliant version number from VERSION.

This module forms part of the InaSAFE tool.

Module: safe.common.utilities

Utilities for InaSAFE

`safe.common.utilities.`**`temp_dir`**(*sub_dir='work'*)

Obtain the temporary working directory for the operating system.

An inasafe subdirectory will automatically be created under this and if specified, a user subdirectory under that.

Note: You can use this together with unique_filename to create a file in a temporary directory under the inasafe workspace. e.g.

tmpdir = temp_dir('testing') tmpfile = unique_filename(dir=tmpdir) print tmpfile /tmp/inasafe/23-08-2012/timlinux/testing/tmpMRpF_C

If you specify INASAFE_WORK_DIR as an environment var, it will be used in preference to the system temp directory.

Args:

sub_dir str - optional argument which will cause an additional subirectory to be created e.g. /tmp/inasafe/foo/

Returns: Path to the output clipped layer (placed in the system temp dir).

Raises: Any errors from the underlying system calls.

`safe.common.utilities.`**`ugettext`**(*s*)

Translation support

`safe.common.utilities.`**`unique_filename`**(***kwargs*)

Create new filename guaranteed not to exist previously

Use mkstemp to create the file, then remove it and return the name

If dir is specified, the tempfile will be created in the path specified otherwise the file will be created in a directory following this scheme:

:file:'/tmp/inasafe/<dd-mm-yyyy>/<user>/impacts'

See http://docs.python.org/library/tempfile.html for details.

Example usage:

tempdir = temp_dir(sub_dir='test') filename = unique_filename(suffix='.keywords', dir=tempdir) print filename /tmp/inasafe/23-08-2012/timlinux/test/tmpyeO5VR.keywords

Or with no preferred subdir, a default subdir of 'impacts' is used:

filename = unique_filename(suffix='.shp') print filename /tmp/inasafe/23-08-2012/timlinux/impacts/tmpoOAmOi.shp

`safe.common.utilities.`**`verify`**(*statement*, *message=None*)

Verification of logical statement similar to assertions Input

statement: expression message: error message in case statement evaluates as False

Output None

Raises VerificationError in case statement evaluates to False

`safe.common.utilities.`**`zip_shp`**(*shp_path*, *extra_ext=None*, *remove_file=False*)

Zip shape file and its gang (.shx, .dbf, .prj) and extra_file is a list of another ext related to shapefile, if exist The zip file will be put in the same directory

This module forms part of the InaSAFE tool.

Module: safe.common.numerics

This module forms part of the InaSAFE tool.

Module: safe.common.tables

tables.py - v0.04 2009-07-28 Philippe Lagadec

This module provides a few classes to easily generate HTML code such as tables and lists.

Project website: http://www.decalage.info/python/html

License: CeCILL (open-source GPL compatible), see source code for details. http://www.cecill.info

class `safe.common.tables.`**`List`**(*lines=None*, *ordered=False*, *start=None*, *attribs=None*)

a List object is used to create an ordered or unordered list in HTML. (UL/OL tag)

Attributes: - lines: list, tuple or any iterable, containing one string for each line - ordered: bool, choice between an ordered (OL) or unordered list (UL) - attribs: dict, additional attributes for the OL/UL tag

Reference: http://www.w3.org/tr/html4/struct/lists.html

class `safe.common.tables.`**`Table`**(*rows=None*, *border=None*, *style=None*, *width=None*, *cellspacing=None*, *cellpadding=None*, *attribs=None*, *header_row=None*, *table_class=None*, *col_width=None*, *col_align=None*, *col_valign=None*, *col_char=None*, *col_charoff=None*, *col_styles=None*, *caption=None*, *caption_at_bottom=False*)

a Table object is used to create a HTML table. (table tag)

Attributes: - rows: list, tuple or any iterable, containing one iterable or TableRow

object for each row

- header_row: list, tuple or any iterable, containing the header row (optional)
- class: str, CSS class to use. Defaults to DEFAULT_TABLE_CLASS
- caption: str, caption for the table
- border: str or int, border width
- style: str, table style in CSS syntax (thin black borders by default)
- width: str, width of the table on the page
- attribs: dict, additional attributes for the table tag
- col_width: list or tuple defining width for each column
- col_align: list or tuple defining horizontal alignment for each column
- col_char: list or tuple defining alignment character for each column
- col_charoff: list or tuple defining charoff attribute for each column
- col_valign: list or tuple defining vertical alignment for each column
- col_styles: list or tuple of HTML styles for each column

Reference: http://www.w3.org/tr/html4/struct/tables.html#h-11.2.1

`column`(*col*, *header=False*)

Return a list contains all element in col-th column Args:

- col = number columnn
- header = if False, doesn't include the header

Returns:

- list of string represent each element

Note: If there is not column number col in a row, it will be represent as empty string ''

toNewlineFreeString()

Return a string representation of the table which contains no newlines.

Note: any preformatted <pre> blocks will be adversely affected by this.

---

class safe.common.tables.**TableCell**(*text='', bgcolor=None, header=False, width=None, align=None, char=None, charoff=None, valign=None, style='', attribs=None, cell_class=None, row_span=None, col_span=None*)

a TableCell object is used to create a cell in a HTML table. (td or th)

Attributes: - text: text in the cell (may contain HTML tags). May be any object which

can be converted to a string using str().

- header: bool, false for a normal data cell (td), true for a header cell (th)
- bgcolor: str, background color
- width: str, width
- align: str, horizontal alignement (left, center, right, justify or char)
- char: str, alignment character, decimal point if not specified
- charoff: str, see HTML specs
- valign: str, vertical alignment (top|middle|bottom|baseline)
- style: str, CSS style
- attribs: dict, additional attributes for the td/th tag

Reference: http://www.w3.org/tr/html4/struct/tables.html#h-11.2.6

class safe.common.tables.**TableRow**(*cells=None, bgcolor=None, header=False, attribs=None, col_align=None, col_valign=None, col_char=None, col_charoff=None, col_styles=None*)

a TableRow object is used to create a row in a HTML table. (tr tag)

Attributes: - cells: list, tuple or any iterable, containing one string or TableCell

object for each cell

- header: bool, true for a header row (th), false for a normal data row (td)
- bgcolor: str, background color
- col_align, col_valign, col_char, col_charoff, col_styles: see Table class
- attribs: dict, additional attributes for the tr tag

Reference: http://www.w3.org/tr/html4/struct/tables.html#h-11.2.5

apply_properties (*cell*, *col*)
Apply properties to the row

column_count ()
Return the number of columns in this row

safe.common.tables.**htmllist** (**args*, ***kwargs*)
return HTML code for a list as a string. See List class for parameters.

safe.common.tables.**table** (**args*, ***kwargs*)
return HTML code for a table as a string. See Table class for parameters.

This module forms part of the InaSAFE tool.

Module: safe.common.test_interpolate

This module forms part of the InaSAFE tool.

Module: safe.common.testing

This module forms part of the InaSAFE tool.

Module: safe.common.test_numerics

This module forms part of the InaSAFE tool.

Module: safe.common.test_polygon

This module forms part of the InaSAFE tool.

Module: safe.common.dynamic_translations

Lookup table mapping layer titles to translatable strings

Layer titles are kept in the associated keywords files. However, these files are not seen by the internationalisation system and can therefore not be translated.

To achieve this for selected titles, we maintain a lookup table of the form

{string: translatable_string}

where string e.g. comes from the keywords file or attribute names/values in datasets and translatable_string is the string that will appear as the translated at runtime.

e.g (using a neutral symbol) {'title1': tr('Jakarta 2007 flood')}

or (using existing title) {'Schools': tr('Schools')}

or (attribute value) {'school': tr('school')}

With the underscore function, the specified string will be seen by the translation system and can appear in the supported languages as with other strings in SAFE.

Note, this module does *not* provide translations! Rather it provides mappings between strings expected at runtime to strings seen by the existing translation systems.

To use:

This module contains words and phrases that need to be translatable but would not normally be available in code for example, that originate from a dataset or external source.

Just put your translations here, and it will be available to the tr function but never import this module or the functions in it!

`safe.common.dynamic_translations.`**`dynamic_translations`**`()`
    These listed here so they get translated apriori to loading data.

This module forms part of the InaSAFE tool.

Module: safe.common.test_tables

InaSAFE Disaster risk assessment tool developed by AusAid - Table Tests implementation.

Contact : ole.moller.nielsen@gmail.com

class `safe.common.test_tables.`**`TablesTest`**`(`*methodName='runTest'*`)`
    Test the SAFE Table

`setUp``()`
    Fixture run before all tests

`tearDown``()`
    Fixture run after each test

`test_cell_header``()`
    Test we can make a cell as a <th> element

`test_cell_link``()`
    Test cell links work

`test_col_span``()`
    Testing column spanning

`test_column``()`
    Test to retrieve all element in a column.

`test_nested_table_in_cell``()`
    Test nested table in cell

`test_row_from_string``()`
    Test row from string - it should span to the table width too

`test_row_span``()`
    Testing row spanning

`test_simple_table``()`
    Test simple table creation

`test_table_by_rows``()`
    Test table from infividual rows

`test_table_caption``()`
    Test table caption

test_table_cells()
Test table from individual cells

test_table_from_string()
Test table from string - should be a single cell in a single row

test_table_with_colalign()
Table columns can be right justified

test_table_with_header()
Test html render of a table with header row(s).

tmpDir()
Helper to get os temp dir.

Args: None

Returns: str Absolute filesystem path to temp dir.

Raises: None

This module forms part of the InaSAFE tool.

Module: safe.common.exceptions

InaSAFE Disaster risk assessment tool developed by AusAid - **Exception Classes.**

Custom exception classes for the SAFE library

Contact : ole.moller.nielsen@gmail.com

exception safe.common.exceptions.**BoundingBoxError**
For errors relating to bboxes

exception safe.common.exceptions.**BoundsError**
For points falling outside interpolation grid

exception safe.common.exceptions.**GetDataError**
When layer data cannot be obtained

exception safe.common.exceptions.**InaSAFEError**
Base class for all user defined execptions

exception safe.common.exceptions.**PolygonInputError**
For invalid inputs to numeric polygon functions

exception safe.common.exceptions.**PostProcessorError**
Raised when requested import cannot be performed if QGIS is too old.

exception safe.common.exceptions.**ReadLayerError**
When a layer can't be read

exception safe.common.exceptions.**VerificationError**
Exception thrown by verify()

exception safe.common.exceptions.**WindowsError**
For windows specific errors.

exception `safe.common.exceptions.`**`WriteLayerError`**

When a layer can't be written

This module forms part of the InaSAFE tool.

Module: safe.common.interpolation2d

This module forms part of the InaSAFE tool.

Module: safe.common.interpolation1d

This module forms part of the InaSAFE tool.

Module: safe.common.polygon

This module forms part of the InaSAFE tool.

Module: safe.common.geodesy

This module forms part of the InaSAFE tool.

Package::safe.storage

Module: safe.storage.test_vector

This module forms part of the InaSAFE tool.

Module: safe.storage.utilities

This module forms part of the InaSAFE tool.

Module: safe.storage.test_clipping

This module forms part of the InaSAFE tool.

Module: safe.storage.test_io

This module forms part of the InaSAFE tool.

Module: safe.storage.test_utilities

This module forms part of the InaSAFE tool.

Module: safe.storage.geometry

class safe.storage.geometry.**Geometry**
Common class for geometries

class safe.storage.geometry.**Polygon**(*outer_ring*, *inner_rings=None*)
Polygon geometry

Contains outer ring and optionally list of inner rings

All rings are assumed to be Nx2 numpy arrays of vertex coordinates lon, lat

This module forms part of the InaSAFE tool.

Module: safe.storage.projection

This module forms part of the InaSAFE tool.

Module: safe.storage.vector

This module forms part of the InaSAFE tool.

Module: safe.storage.core

This module forms part of the InaSAFE tool.

Module: safe.storage.clipping

This module forms part of the InaSAFE tool.

Module: safe.storage.utilities_test

Utilities to to support test suite

safe.storage.utilities_test.**same_API**(*X*, *Y*, *exclude=None*)
Check that public methods of X and Y are the same.

Args:

- X, Y: Python objects
- exclude: List of names to exclude from comparison or None

This module forms part of the InaSAFE tool.

Module: safe.storage.raster

This module forms part of the InaSAFE tool.

Module: safe.storage.layer

This module forms part of the InaSAFE tool.

CHAPTER

FIVE

AUTHORS, CONTRIBUTORS AND FUNDERS

Authors ordered by date of first contribution:

- Ole Nielsen <ole.nielsen@aifdr.org>
- Ted Dunstone <ted@biometix.com>
- Ariel Nunez <ingenieroariel@gmail.com>
- Tim Sutton <tim@linfiniti.com>
- Ismail Sunni <ismailsunni@yahoo.co.id>
- Krisy Van Putten <Kristy.VanPutten@ausaid.gov.au>
- Marco Bernasocchi <marco@opengis.ch>
- Oscar Kurniawan <misugijunz@gmail.com>
- Trevor Dhu <Trevor.Dhu@ausaid.gov.au>
- Gigih Aji Ibrahim <bungcip@gmail.com>

Other contributors who have provided input, testing, translations etc.

- I Made Anombawa <anom1403@gmail.com>
- Giovanni Manghi <giovanni.manghi@faunalia.pt>
- Maning Sambale <emmanuel.sambale@gmail.com>

This work was kindly supported by AusAid and the World Bank.

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

S

safe.common.dynamic_translations, 143
safe.common.exceptions, 145
safe.common.tables, 141
safe.common.test_tables, 144
safe.common.utilities, 139
safe.common.version, 139
safe.defaults, 133
safe.engine.utilities, 133
safe.postprocessors.abstract_postprocessor, 136
safe.postprocessors.age_postprocessor, 135
safe.postprocessors.gender_postprocessor, 135
safe.postprocessors.postprocessor_factory, 136
safe.storage.geometry, 147
safe.storage.utilities_test, 147

InaSAFE: Indonesia Scenario Assesment for Emergencies
◣ Training Manual for Software Developers