DUG-Seis Documentation

S. Demir

August 29, 2019

Contents

1 Installation and Use

1.1 Installation Manual

- **1**: This software has been tested to work on both macOS and Windows. Login to gitlab.seismo.ethz.ch with your ETH credentials and let Dr. Katrin Plenkers know. Once you have done that, you'll be given acces to the DUGLab repository.

- **2**: Install GIT on your computer, if you don´t already have it. The software is found on the web page: "**https://git-scm.com/"**. Keep all default recomendation options during installation.

- **3**: Download the latest version of the DUG-Seis software to your computer using GIT. This is done by going to the project in "**gitlab.seismo.ethz.ch**", going into "**doetschj/DUG-Seis**", clicking on clone and copying the link behind HTTPS by using the copy button. Create an empty folder, with the following name: "**DUG-Seis**", in which the software will be put. Now open the anaconda prompt in the computer. use "**cd**" to go to the "**DUG-Seis**" folder where you wish to install DUG-Seis. When you're in the folder give the following command with the copied link in it: "**git clone "copied link**"". This should equal the following command: "**git clone https://gitlab.seismo.ethz.ch/doetschj/DUG-Seis.git**". Now a login window will appear. Use your ETH credentials for this.

- **4**: Install anaconda, if you don´t already have it. The software is found on the web page: "**https://www.anaconda.com/distribution/**". Keep all default recommendation options during installation.

- **5**: For commands in the anaconda prompt it is recommended to type these manually instead of copying from this document. Go to the anaconda prompt (to be found when searching for cmd in windows search) and give the following command: "**conda config --add channels conda-forge**"

- **6**: Create a new conda environment using the command: "**conda create -n dug-seis python=3.6 pip numpy redis celery**". It is important to use python version 3.6, as there are some incompatibilities between different packages for 3.7.6. Always press "**y**" followed by "enter" when asked to do so.

- **7**: Activate the DUG-Seis environment with the following command: "**conda activate dug-seis**"

- **8**: Install the obspy package with the following command: "**conda install obspy**"

- **9**: As a test, you can make sure that you have the latest version of obspy with the command: "**conda update obspy**"

- **10**: cd into the base folder of the DUG-Seis software. This is the newly created DUG-Seis folder.

- **11**: Install the DUG-Seis software using "**pip install -e .**" The "**-e**" in the install command makes sure that you can change the code and the changes will be directly effective, when you call DUG-Seis again. Once we are out of the immediate development stage, it makes sense to install using "**pip install .**". In this latter case you will have to re-install, when you changed the code or downloaded a new version.

1.2 Use Manual

- **1**: For running the software, create a new folder that will be filled with software output during operation and copy a YAML file (dug-seis.yaml) into the empty folder. The YAML file contains parameter preferences for the main purposes of the software: data acquisition and processing. It is located in the "**config**" folder in the DUG-Seis software. For the first time of running the software it is recommended to call this folder "**DUG_Seis_output**".

- **2**: In the YAML file, change the asdf_folder parameter to the path to the folder containing the data you wish to process. Change other YAML parameters according to the information giving in the YAML file chapter.

- **3**: Go to the anaconda prompt and use cd till you are in the same folder as where the YAML file is located: "**DUG_Seis_output**".

- **4**: Activate the DUG-Seis environment with the following command: "**conda activate dug-seis**"

- **5**: If you enter "**dug-seis**" in the commandwindow, it will give you the the following "help" output:

  Usage: dug-seis [OPTIONS] COMMAND [ARGS]...
  Run data acquisition and real-time processing of induced seismicity during rock-laboratory experiments
  Options:
  -v, –verbose Enables verbose mode
  –cfg <config file> Source config file. If not specified,the script tries ./dug-seis.yaml and config/dug-seis.yaml
  –mode <mode> Mode can be either "live" (default) or"post", for post processing mode)
  –log <path> Specify a log file
  –version Show the version and exit.
  –help Show this message and exit.
  Commands:
  acquisition Run data acquisition The output goes to ASDF files in the...
  dashboard Run dashboard to show recent events
  merge Merge short ASDF files The output goes to ASDF files in...
  processing Run event trigger on ASDF files
  show-parameters Show parameters

- **6**: The software can be used for 3 independent purposes that can be done on their own shown below in 6a, 6b, and 6c. In all 3 modes, the program will look for the YAML file "**dug-seis.yaml**" in the folder it is called, but you can use a YAML file with a different name using the "—**cfg**" option. To quit the running of each mode type: "**ctrl+c**" in the command prompt.

- **6a**: For using the software as a processing tool for seismic data, type the command "**dug-seis processing**" in the command window. Important note: In the processing mode, the software will output, among other things, csv files and continuously write into these while the software is running. Do not open the created csv files while running the software. This will cause an error since the software cannot write into a csv file when it is opened. The folder with the YAML file and the created software output will look like shown below.

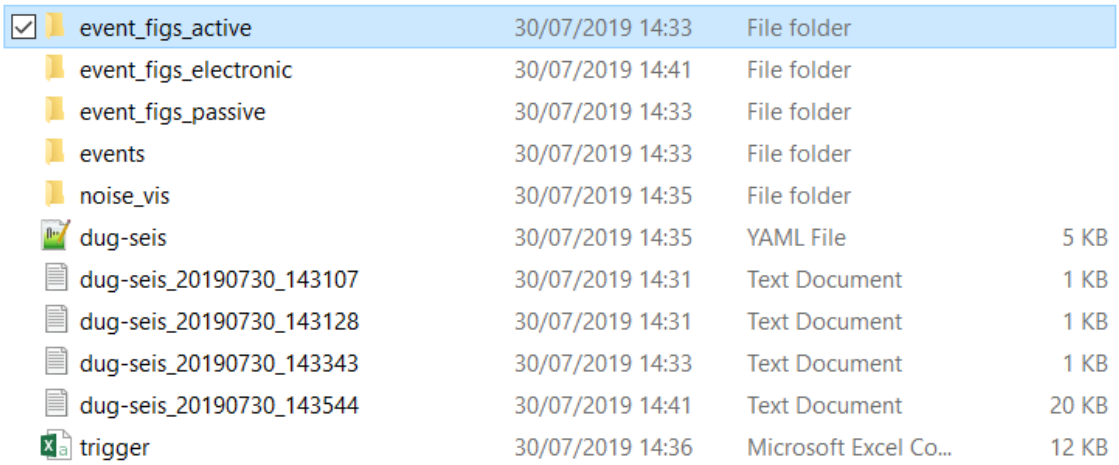

Figure 1: DUG-Seis processing output

- **6b**: In order to use the software as a seismic data acqusiition tool in the rocklab type "**dug-seis acquisition**" in the command window.
- **6c**: For showing the waveforms, event list and a visualization, use "**dug-seis dashboard**".

2 The DUG-Seis Software Structure Intodroduction

DUG-Seis is a software developed in Python using the open soruce "**obspy**" library. The structure of the software can be seen below.

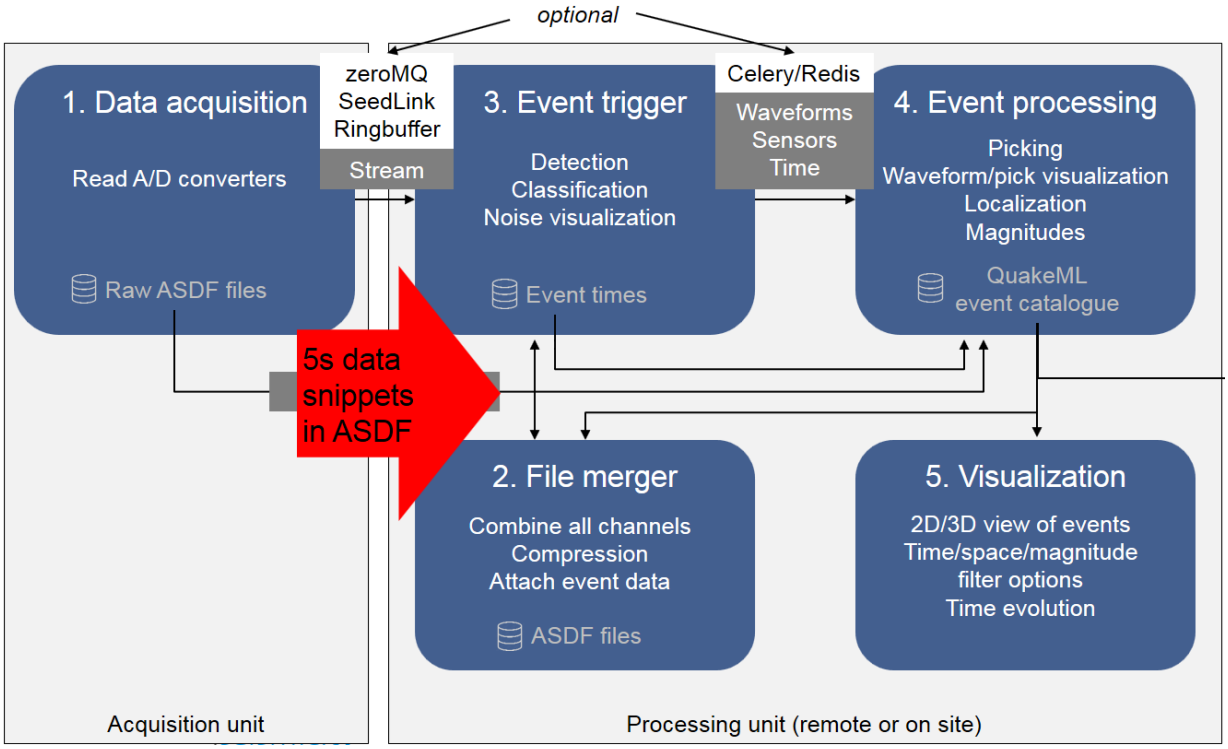

Figure 2: DUG-Seis Software General Structure

The purpose of the software is to be independently able to be used for both seismic data acquisition and seismic data processing. When using the software in the data acquisition mode, data snippets of a to be set time are being created and put in a to be set folder.

In the processing mode, the software then reads in these data snippets (normally set to 5s) and goes through a chain of operations/modules on the data. The first operation is the event trigger shown in block "**3**". In this operation a trigger is run over the data at specific traces that are set manually and when a event is detected this even is classified into "**active**", "**passive**" and "**electronic**". "**active**" means that the event is an active seismic source like a hammer shot. "**passive**" means that the event is a passive seismic event caused by micro fracturing for example. "**electronic**" means that the event is electronic noise. The output of this operation is the earliest time of each event (event times) and it's assigned classification. In the second operation shown in block "**4**", the software creates a snippet of set length, normally 20 ms, around the triggered event and now picks arrivals in all traces. Then these picked arrivals are used in a localization algorithm, after which the magnitude of the event is determined with a magnitude algorithm and finally the snippet (standard 20ms) us plotted showing all traces in this time window and the picks that have been done on these traces. All outputs of this operation except the waveform plot is stored in the QuakeML(xml) format. This means that for each event, a QuakeML(xml) file is created. Block "**5**" does not exist.

3 The YAML file

The YAML file is the python compatible parameter configuration file for the DUG-Seis software. The file is divided in different compartments that roughly equal the the different modules in the DUG-Seis software. This division in different compartments is shown by indention. The different compartments are: General, Acquisition, Trigger, Processing and Merge.

3.1 Relation Software Structure and YAML file

The Data acquisition (block "**1**") parameters are located in the Acquisition compartment in the YAML file. The Event trigger parameters (block "**3**") are located in the Trigger compartment in the YAML file. The Event processing (block "**4**") including waveform visualization parameters are located in the Processing compartment in the YAML file.

3.2 General

In the general compartment the general parameters of the rock lab experiment can be specified. Most important are the following parameters:

- **project_name**: The name of the project, names need need underscores at spaces.
- **project_name**: The location of the project, names need need underscores at spaces.
- **operator**: The operators of the project. This needs the following format: "**['J. Doetsch','S. Demir']**".
- **sensorcount**: number of sensors used and the coordinates of each of these.
- **sensorcoords**: the coordinates in x,y,z in the local coordinate system. Each line specifies the coordinates of a single sensor. There are always 32 lines necessary. When there are not 32 sensors being used these lines need to be filled in with zero's for the x, y and z coordinates.
- **activetriggerchannel**: the number of the sensor used as active trigger. This is normally set to 1. Then this emans that in the rocklab trace 1 needs to be conencted to an active source like a hammer.
- **originch1903east**: easting origin of local coordinate system.
- **originch1903north**: northing of local coordinate system.
- **originelev**: elevation of origin of local coordinate system. This is relative to sea level.

3.2.1 stats

- **network**: A 2 letter combination specific to data from a certain site. It is important when processing data with the processing module that the network parameter is is the same as the one used when the data was recorded using the acquisition module. In case of the Grimsel testdata set, the network parameter is "**GR**", for the data recorded in Mt. Terri it is "**XM**".

3.3 Acquisition

3.3.1 hardware settings

This is a sub compartment with parameters specific to the hardware setting for the data file acquisition.

- **samplingfrequency**: The frequency of sampling for data acquisition in Hz. Normally set to 200000.
- **gainselection**: The gain range selection for each channel. A single channel equals a single sensor. a lower value will result in a higher file size since the noise level will take more storage space. Normally set to 10000 for each channel.

3.3.2 asdf settings

This is a sub compartment with parameters specific to the settings for the data file acquisition.

- **use_float32**: Normally set this parameter to false, which means that the data is stored in the "**int32**" format.
- **datafolder**: The folder in which acquisition data files will be stored.

- **datafoldertmp**: The folder in which a single file is stored while it is still being formed. After this is done it is moved to the standard data folder.
- **compression**: Normally set this parameter to "**'gzip-3'**". The use of other compression modes results in a change in file sizes.
- **filelengthsec**: The recording time of each data file(snippet) in seconds. Normally set between 5 and 30 seconds.
- **station_naming**: with this parameter the order of the traces can be manipulated. It is best to not change it.

3.4 Trigger

This is the first compartment with parameters that are used for triggering and event classification.

- **asdf_folder**: The folder from which data files are read in for processing.
- **channels**: The channels that the trigger is applied up on. For example, if there are only 4 sensors in a borehole close to the point of injection and all the other sensors are in the main tunnel further away from the point of injection, then these 4 could be selected for the trigger. The active trigger channel should normally be included in the channels used for triggering since this channel that is conencted to an active source like a hammer will be used by the software to determine is an event is active or passive.
- **overlap**: The overlap used when loading in data snippets. The data snippets are load in for triggering with a certain overlap with the goal to not miss any events. The unit is in seconds.
- **gainrange**: The gainrange is used for scaling the loaded data. Normally set to 'YAML' which means that the gainrange is retrieved out of the YAML file, specifically the gainselection parameter in the hardware setting compartment. The other option is to retrieve this information fromt he data snipepts directly but this has not been implemented yet.
- **bandpass_f_min**: before triggering a bandpass is applied. This is the minimum frequency.
- **bandpass_f_max**: before triggering a bandpass is applied. This is the maximum frequency.
- **trigger_algorithm**: This is the used trigger algorithm. Normally set 'recstalta' which stands for recursive STA LTA. Another option is 'threshold'.
- **coincidence**: The coincidence used for triggering in the recursive STA LTA algorithm. If the coincidence is set to for example 3, this effectively tells the trigger to only go off and register an event if there are at least arrivals registered at 3 or more traces.
- **starttime**: Time subtracted of earliest trigger time that decides at what time the snippet for processing starts.
- **endtime**: Time added to earliest trigger time that decides at what time the snippet for processing ends.
- **noise_vis**: Choice between doing noise visualization, "True" or "False". Normally set to True.
- **noise_vis_interval**: The interval between noise visualizations being generated in seconds. Normally set to 300, which means that every 300 seconds a plot of all traces (filled with noise) is generated without any trigger having gone off.

- **time_between_events**: The empty time in seconds that has to exist in between 2 different events. For example set to 0.0006 seconds. The purpose of this parameter is to make sure that a single event is not recognized as 2 separate events. If the parameter is chosen too high 2 separate events may be recognized as a single event.
- **time**: Date and time from which to start processing uses the format 20190523_0505 for 23rd of may 2019 at 5.05 am. This means that if there are data files from before this date in the data folder that these will not be used for processing.

3.4.1 sta_lta

This is a sub compartment with parameters specific to the recursive STA LTA trigger algorithm. For more information: "**http://gfzpublic.gfz-potsdam.de/pubman/item/escidoc:4097:3/component/escidoc:4098/IS_8.**

- **threshold_on**: The treshold for starting of triggering (-).
- **threshold_off**: The treshold for quitting of triggering (-).
- **st_window**: Length of the short time window (samples).
- **lt_window**: Length of the long time window (samples).

3.4.2 classification

- **spread_int**: The maximum time between the first and the last moment of triggering at different stations for which this event is classified as electronic interference. in seconds. Normally set to 0.25e-2.

3.5 Processing

This compartment contains the parameters related to event picking, localization, magnitude estimation and visualization.

- **parallel_processing**: The choice for using parallel processing or not. True or False.
- **number_workers**: The number of workers used when parallel processing is set to true. Normally set to 5 which means 5 events will be processed at the same time
- **vp**: p-wave velocity in m/s used for localization or hypocenters.
- **vs**: s-wave velocity in m/s used for localization or hypocenters.
- **bandpass_f_min**: Before picking a bandpass is applied. This is the minimum frequency.
- **bandpass_f_max**: Before picking a bandpass is applied. This is the maximum frequency.

3.5.1 picking

- **picker_algorithm**: The used algorithm for picking. As of 14 june 2019, the STA LTA picker is most stable and configurable for different conditions. This algorithm is selected with: 'sta_lta'. Other options are the P-Phase picker from USGS "**https://www.usgs.gov/software/p-phase-picker**", set with "**pphase**", and the AICD, FB and KT picker from Austin Holland "**https://github.com/austinholland/PhasePApy**", set with "**aicd**", "**fb**" and "**kt**" respectively.

3.5.1.1 sta_lta

- **threshold_on**: The threshold for starting of triggering.
- **threshold_off**: The threshold for stopping of triggering.
- **st_window**: Short time window in samples.
- **lt_window**: Long time window in samples.

3.5.2 locate

- **localization_algorithm**: The algorithm used for hypocenter localization. standard set to 'hom_iso'. other options: hom_iso, hom_aniso.
- **min_picks**: The minimum number of picks at different stations that have to be done for one event for hypocenter localization to be done. Normally set to 6.

3.5.2.1 hom_aniso

- **vp_min**: Used p wave velocity in m/s.
- **epsilon**: Used Thomsen epsilon value.
- **delta**: Used Thomsen delta value.
- **azi**: Azimuth of anisotropy vector in degrees.
- **inc**: Inclination of anisotropy vector in degrees.

3.5.3 Magnitude

- **magnitude_algorithm**: Which algorithm to use for magnitude estimation of the event. As of 14 june 2019 one available: 'relative'.

4 Output

4.1 QuakeML (xml) Files

4.1.1 events

In the folder "events" that is created automatically if it doesnt exist, QuakeML (xml) files are created for each event. This file contains all processing information for each event such as picks, localization and magnitude.

4.2 Event Visualization (png) Files

4.2.1 Event figs active

In this folder the events that have been classified as active events are plotted. The naming follows the same structure as the "time" parameter in the trigger compartment in the YAML file.

4.2.2 Event figs passive

In this folder the events that have been classified as passive events are plotted. The naming follows the same structure as the "time" parameter in the trigger compartment in the YAML file.

4.2.3 Event figs electronic

In this folder the events that have been classified as electronic interference are plotted. The naming follows the same structure as the "time" parameter in the trigger compartment in the YAML file.

4.3 CSV Files

4.3.1 event-loc.csv

In this csv file the event id is given in column 1 and x,y and z coordinates are given in columns 3, 4 and 5 respectively.

4.3.2 trigger.csv

In this csv file the event id is given in column 1, the time of the first trigger is given in column 3 and the classification is given in column 4.

4.4 Log-File

4.4.1 dug-seis.log

In this file all log messages generated during the runtime of the processing are stored. This includes messages on which files is being worked, time information on triggered events and number of picks done as well as plotting information.