# An unfinished Tutorial for


# L_SU,
# a graphical user interface
# for Seismic Unix (CSM),
# under Linux

**Louisiana State University**
**Dept. Geology and Geophysics**
**Baton Rouge**
**Juan M. Lorenzo**

**Version: 0.3.6 (January, 2019)**

# 1 General Information

## 1.1 Acknowledgements

This project is possible only because of the selfless work of others. I have shamelessly copied and modified notes extensively from the Colorado School of Mines website (Stockwell) for S*nix. Over the years, many students have also contributed to these notes: Class of 2008: Erin Walden, Kody Kramer, Erin Elliott, Andrew Harrison, Andrew Sampson, Ana Felix, JohnD'Aquin, Russell Crouch, Michael Massengale, and David Smolkin; Chang Liu (2013).

## 1.2 What is L_SU?

L_SU, a graphical user interface (GUI), serves to select and build sequences of Perl modules and their parameters. L_SU generates two versions of these instructions in text files. These text files contain a shell and a Perl script version that can be modified and also executed independently of this GUI and from the command line.

Seismic Unix (Stockwell, 1999) is a widely distributed free software package for processing seismic reflection and signal processing. In Seismic Unix, a sequence of independent programs receive modify and generate data files of streams of data that are displayed on the screen. The data file read in and the generated output data original are handled internally by stdin, stdout functions in C while the data exchanges between programs and the linux operating system are managed from the command line via pipes "|" and redirections "> or <" respectively. Traditionally, the instructions on the command line can be assembled and saved as re-usable bash scripts. L_SU assembles these same scripts for the operating system to run with the help of modules written in Perl. L_SU generates these scripts within the directory of the user and thes scripts can be run independently of L_SU running.

L_SU is written using Perl/Tk which is mature, well-documented Perl module that allows its users to construct graphical user interfaces.

In a classroom environment, shell scripting of SU modules engages students and helps focus on the theoretical limitations and strengths of signal processing. However, complex interactive processing stages, e.g., selection of optimal stacking velocities, killing bad data traces, or spectral analysis requires advanced flows beyond the scope of introductory classes. In a research setting, special functionality from other free seismic processing software such as SioSeis (UCSD-NSF) can be incorporated readily via an object-oriented style to programming.

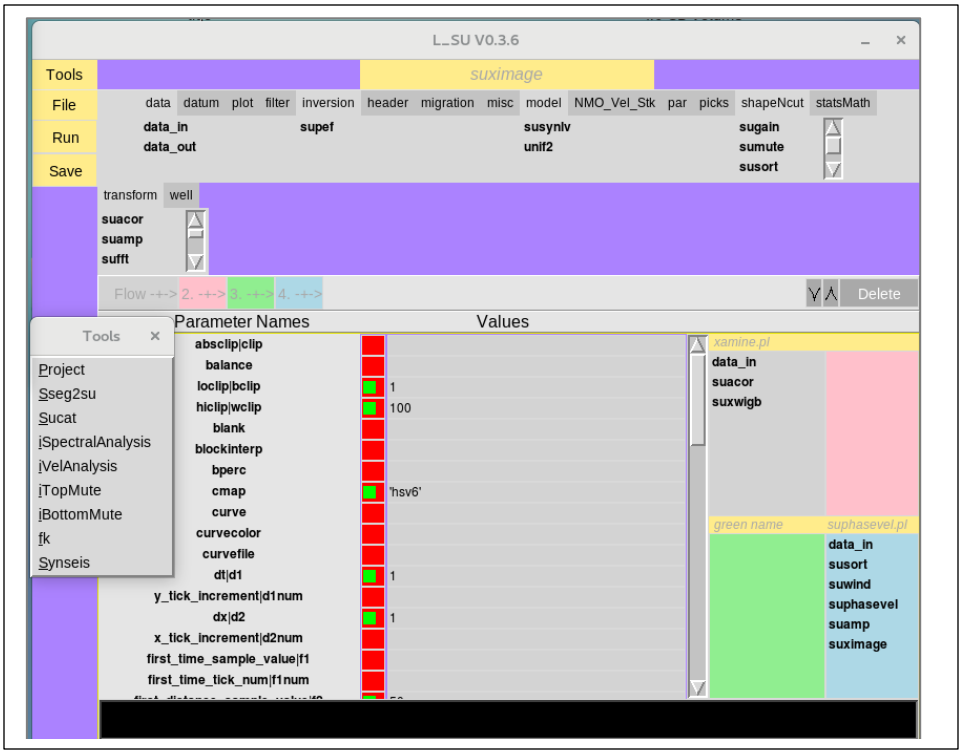
An object-oriented approach is a first step toward efficient extensible programming of multi-step processes, and a simple GUI simplifies parameter selection and decision making. Currently, in L_SU, Perl 5 packages wrap 19 of the most common SU modules that are used in teaching

undergraduate and first-year graduate student classes (e.g., filtering, display, velocity analysis and stacking). Perl packages (classes) can advantageously add new functionality around each module and clarify parameter names for easier usage. For example, through the use of methods, packages can isolate the user from repetitive control structures, as well as replace the names of abbreviated parameters with self-describing names. Moose, an extension of the Perl 5 object system, greatly facilitates an object-oriented style. Perl wrappers are self-documenting via Perl programming document markup language.

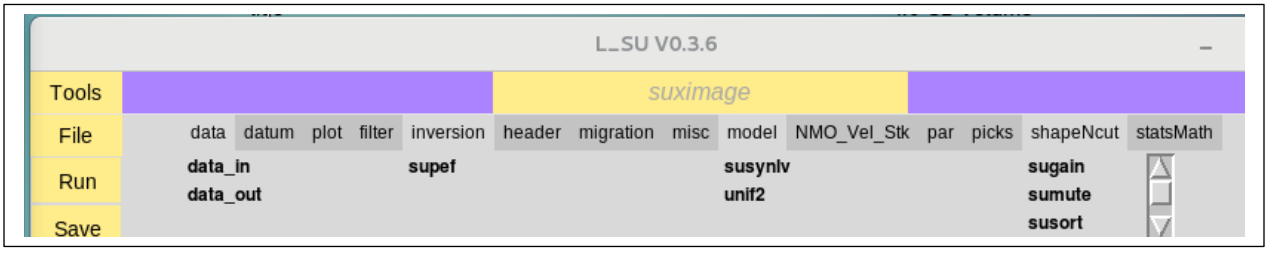
An automatic directory structure is created for the user in which data and programs are distributed according to a pre-defined hierarchy. All the directories and minimal files needed by L_SU are created whenever a new 'Project' is created within the 'Project Selector' tool. The user can also create new projects within main GUI of L_SU as well as selecting different projects. At all times the user can use linux commands to navigate freely through the directories. Sometimes the user may find it convenient to create new subdirectories within the existing file structure. L_SU will not be able to detect these folders and their contents.

# 1.3 GUI Sections

## 1.3.1 Overview




## 1.3.2 Top Menu



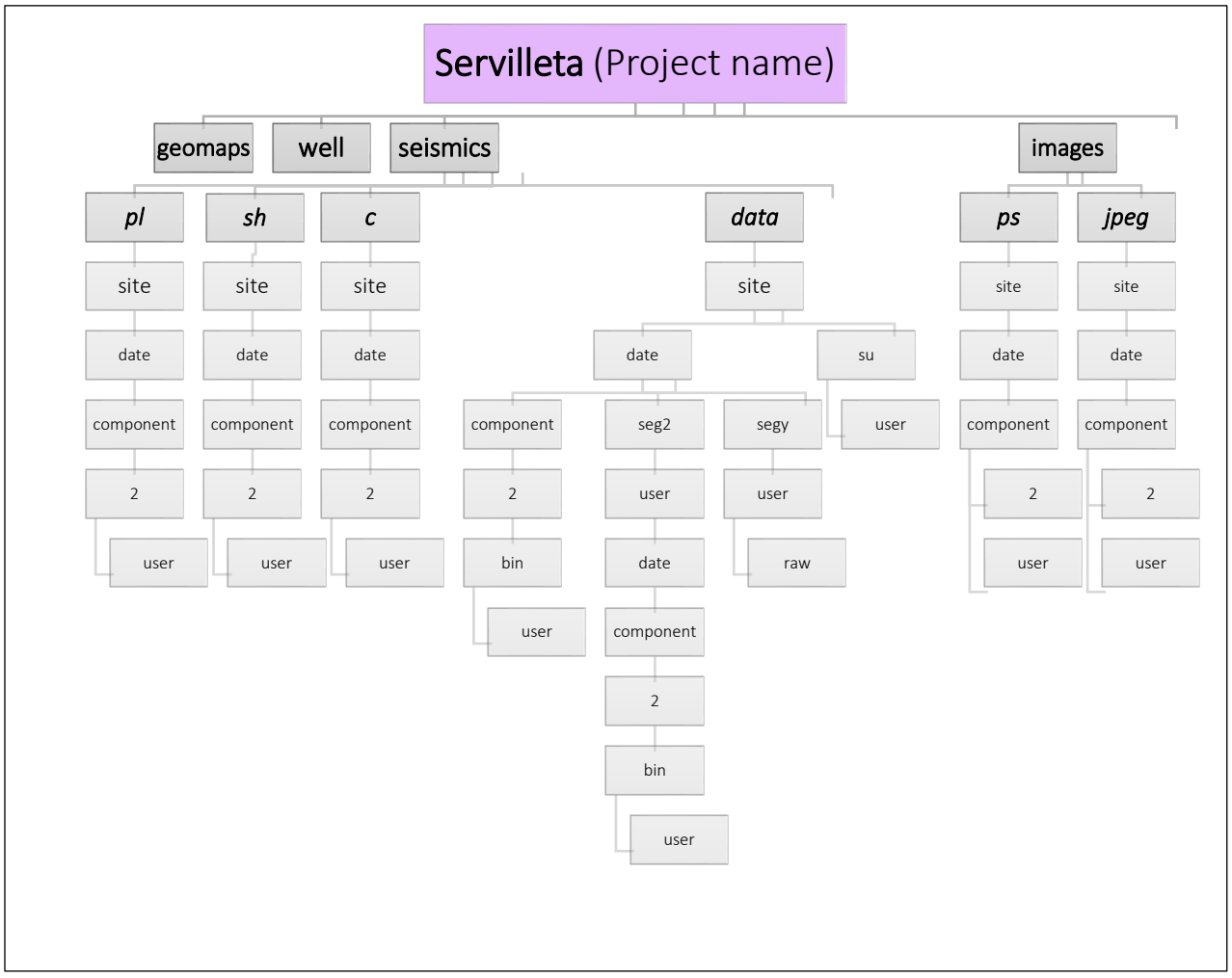
### 1.3.3 Side Menu

### 1.3.4 Parameter Names and input

### 1.3.5 Four Flow Boxes

### 1.3.6 Bottom Message Box

# 1.4 What is an example directory structure for a Project?




## 1.4.1 Where are my data sets stored?

Before starting a new project you should understand the file structure in which programs and data sets are stored. The main directories are as follows:

## 1.4.2 Copying data into the project directory structure from elsewhere in the system

If you want to copy seismic data already in su (seismic unix) format copy it with the following instruction, but first move yourself into the directory that receives the data.

Example 1:

```
% cd  PROJECT_HOME/seismics/data/site/component/line/username
```

Example 2:

```
% cd  /home/gom/seismics/data/Servilleta/H/1/gllore
```

Example 2:

```
% cp data /home/refseis18/Aug27_lab1/*.su   .
```

### 1.4.3 Where are my flows kept?

## 1.5 Text conventions in this tutorial and their meaning

Left Mouse click is abbreviated to <MB1> and is displayed with a blue background

Right Mouse click is abbreviated to <MB3> and is displayed with a blue background

**Variable names are shown in a large bold-style font**, and their meanings are highlighted with a green background

*% Command-line instructions are shown with pink background*

## 1.6 Glossary

| Term | Explanation and Example | Brief |
|---|---|---|
| **HOME** | Full linux directory path to the user's home directory, e.g. /home/xavier45 | home directory path |
| **PROJECT_HOME** | Located inside **HOME** directory -- can be a soft link | project directory path |
| **Projectname** | e.g., Servilleta -- a National Wildlife Refuge in New Mexico, U.S.A. | name of the project |

| | | |
|---|---|---|
| **spare_dir** | can be left empty | a bonus directory |
| **date** | 053018 | Of field work |
| **component** | Z stands for vertical and and H can be horizontal but any name is possible | Geophone particle displacement component |
| **line** | 1 | used to identify a profile |
| **user** | e.g., xavier45 | login name |
| **subUser** | must be set to **username** | Allows groups to share Project space |
| **flow** | Data_in, sugain, suximage | Sequence of programs to execute |
| **geomaps** | Directories will be created when working with-maps | Directories for third-party software (if installed and accessible) |
| **sqlite** | Databases | Directories for third-party software (if installed and accessible) |
| **gmt** | GMT | Directories for third-party software (if installed and accessible) |
| **grass** | GRASS GIS | Directories for third-party software (if installed and accessible) |

**Table 1: Definitions of terms used when creating working projects**

# 2 Quick start: New Projects and old Projects

**When either creating a new project or accessing a pre-existing project instances, always start by running the following instruction:**

*% L_SU*

## 2.1 Create your first Project

The following instruction starts the program:

***% L_SU***

If you do not have any projects created previously, then:

Left Mouse click <MB1> on Create New

Otherwise, go to next section: Old Projects)

After clicking on Create New, a default set of parameter names (e.g., **HOME**) and their values (e.g. **/home/gllore**) appears:

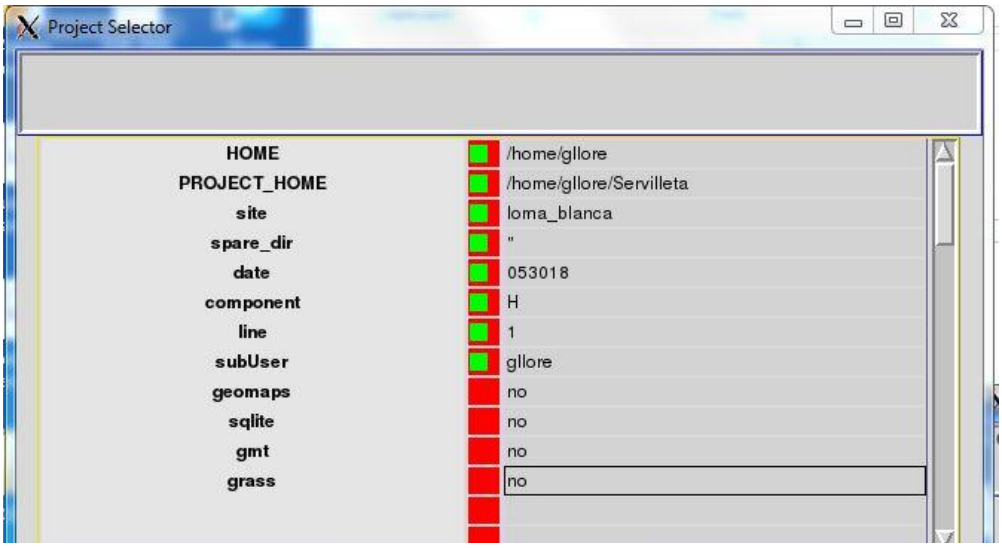



**Figure 1: Screen capture of Project Selector Pane with parameters and their values**

The Project Selector pane displays several default options that work with the test data set that is included for this tutorial. These options should be update with an actual user name, for example:

| Parameter name | Default values | User's new values |
|---|---|---|
| **HOME** | /home/**gllore** | home/**username** |
| **PROJECT_HOME** | /home/**gllore**/Servilleta | /home/**username**/Servilleta |
| **Site** | Servilleta | **loma-blanca** |
| **spare_dir** | "" | |
| **date** | 053018 | 053018 |
| **component** | Z | H |
| **line** | 2 | 1 |
| **subUser** | gllore | **username** |
| **geomaps** | no | no |
| **sqlite*** | no | no |
| **gmt*** | no | no |
| **grass*** | no | no |

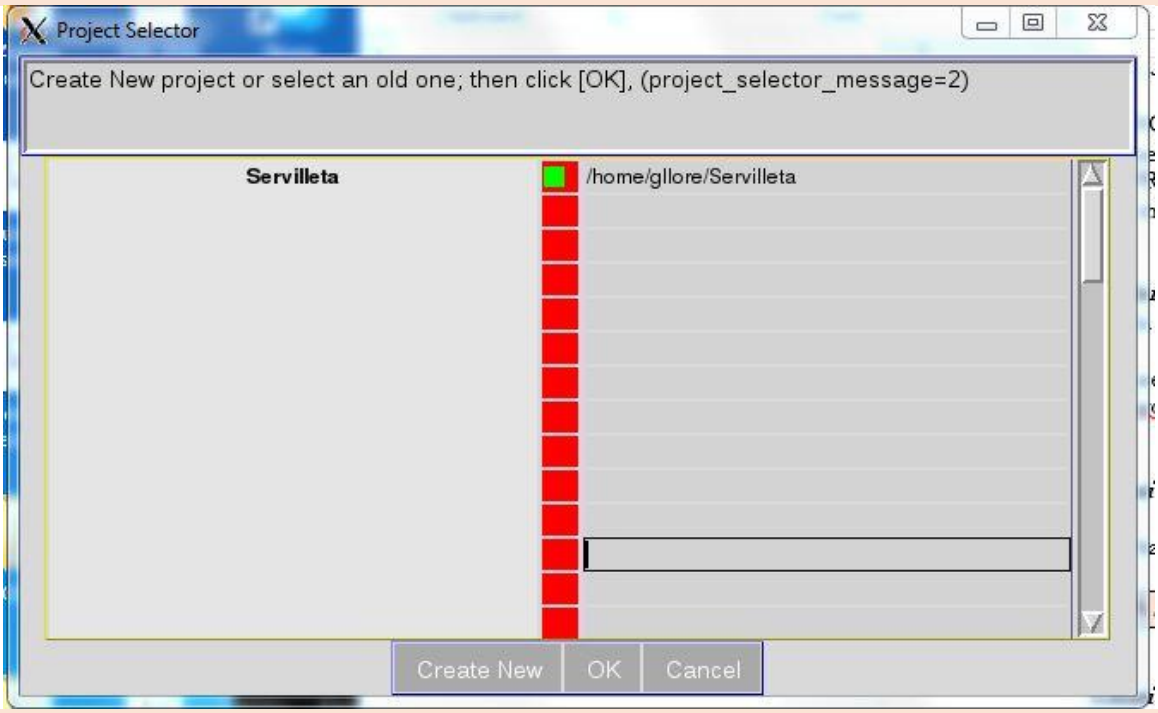
**Table 1: Suggested changes to parameter vlaues**

*** if set to 'yes' only the directories will come to be created although the accompanying programs are not yet available in this version ( Aug 27 2018)**

## 2.2 Select an existing Project

The following instruction starts the program*:

***% L_SU***




*% L_SU*

If the project of interest (in this case Servilleta) is selected (**button is green**) then:

Left Mouse click <MB1> on OK

The old variables are defaulted from prior projects and serve as an example to guide your input. Only the path to the home directory of the user is required to follow the standard linux file structure naming system.

## 2.3 Create your first flow

Assemble a sequence of modules to carry out a processing procedure. Choose one of four differently colored flow windows ( grey, pink, green and blue) in which to place your sequence. The colored window appear on the right-hand side of the main window.

A module, with a specific functionality, is selected by clicking on its name from within the list on the left-hand side of the main window.

The module name must be transferred to the list on the right by clicking one of the four different colored flow arrows, just to the right of the word “Flow”.

A final assembled flow must first be saved to a file before it is executed (**File->SaveAs**). Thereafter all executions require that the flow be first saved before running.

In a simple sequence of modules, data are usually read in first, the data is modified and the result is placed into another file or displayed using an imaging module (e.g., suximage, suxwigb)

1. Select the following named modules: ***data_in*, *sugain*,** and ***suxwigb***. Click on each names inside list on the left side of the window. When you do that, the words in the row immediately above will become activated. You will then be able to click on the words inside the grey box:

Flow-+->

You should be able to see the name of the program that you just selected move over to a colored box on the right-hand side of the window.

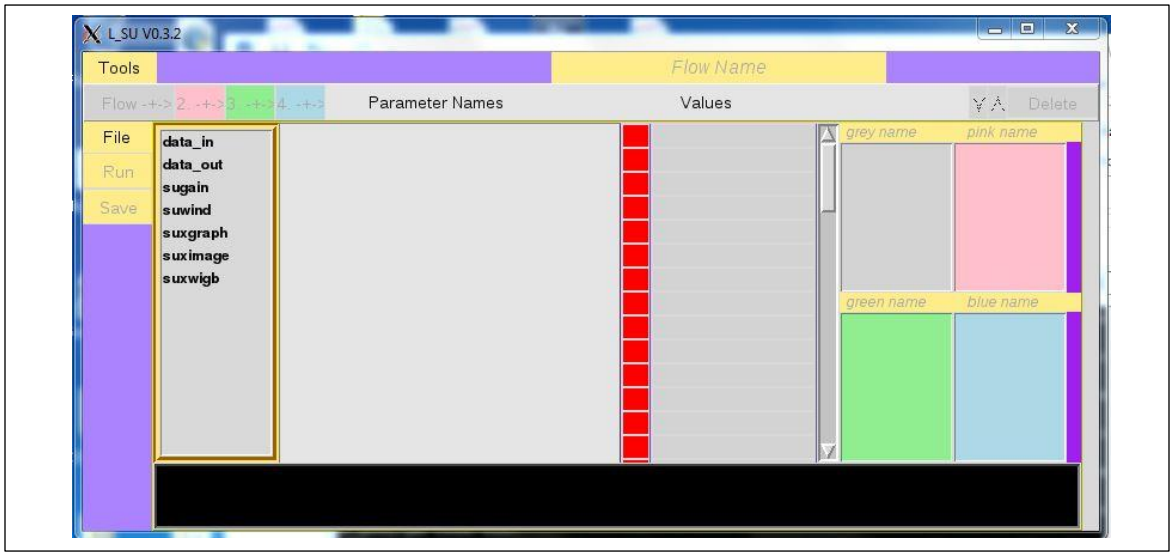
Select each of the three program names: ***data_in*, *sugain*,** and ***suxwigb***

2. You are required to select a **Value** for **base_file_name** (= "file name").

To do so, move your cursor into the corresponding row to the right of **base_file_name**.

A click of the right-mouse-button will automatically open a second window from which you can select a file, e.g. **"1001_clean.su"** Please make certain that you select a file with the words: "**clean**" in its name. These files are guaranteed to be fit for reading.

Before you can run the program you have built, it must be saved:




<MB 1> on **File/SaveAs**

Save the resultant file as, e.g.,

“ViewFile.pl”

Then, click on (***Run***)

You will need to run a sequence of 3 programs in L_SU:

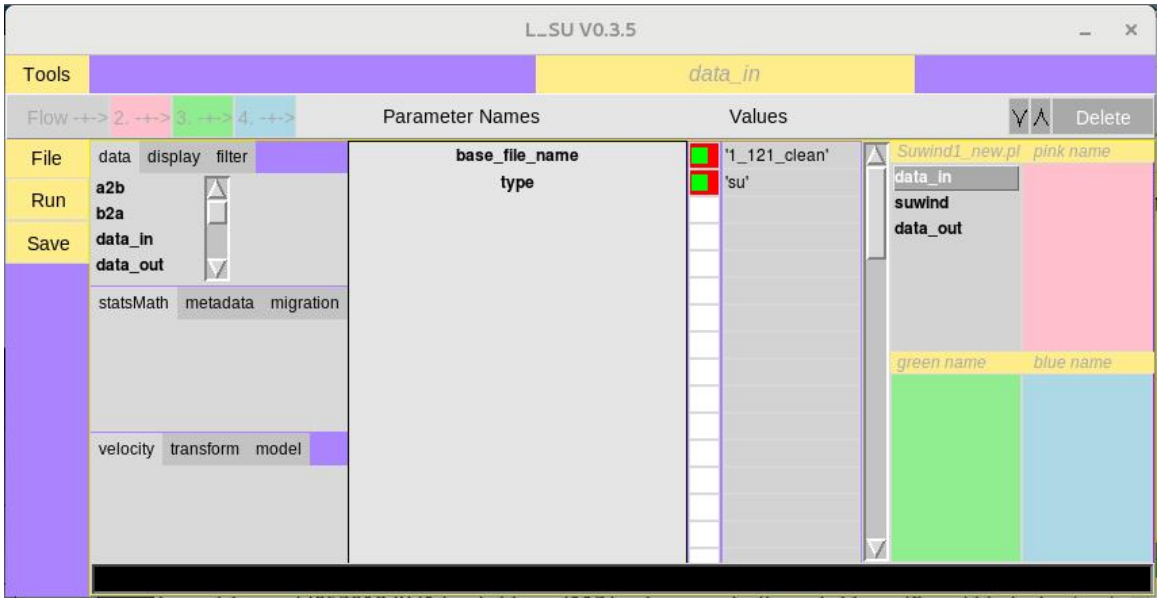
data_in, sugain, and suximage

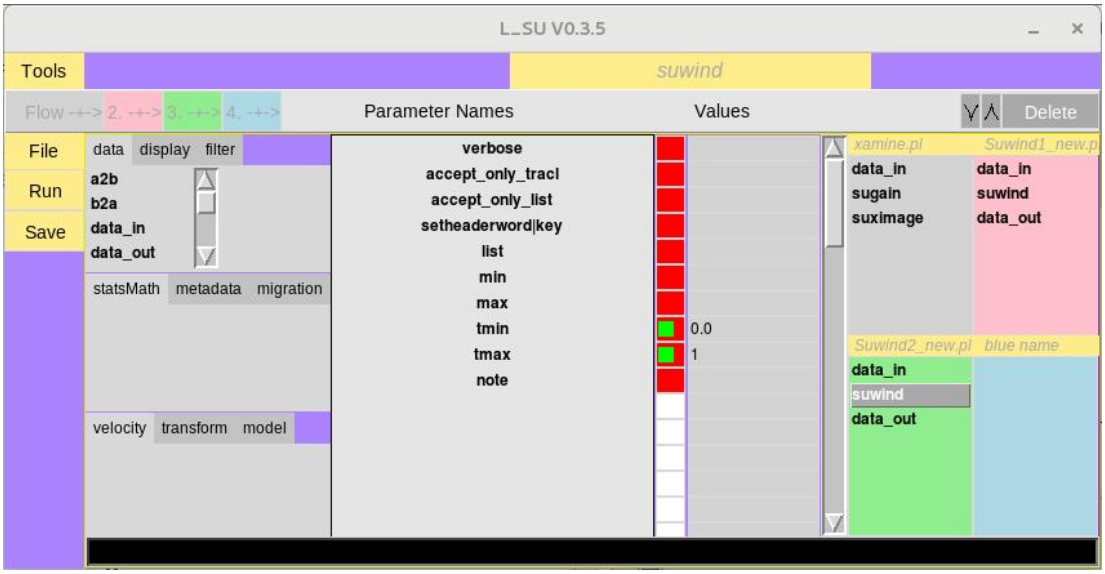
Save the sequence in its own file,

<MB1> File/SaveAs

Then, click on (***Run)*** on the left-hand side of the window.

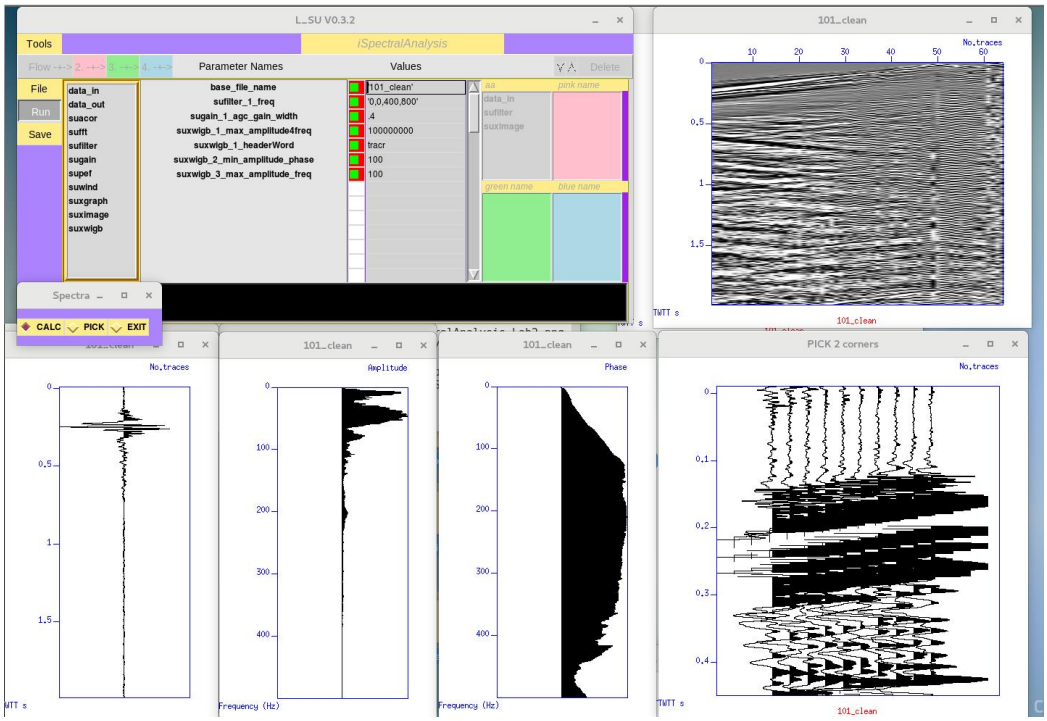
When you want to select and save only portions of the data set you can window out these segments by the value of the header or the range of times collected in the data






*tmin=0.001*

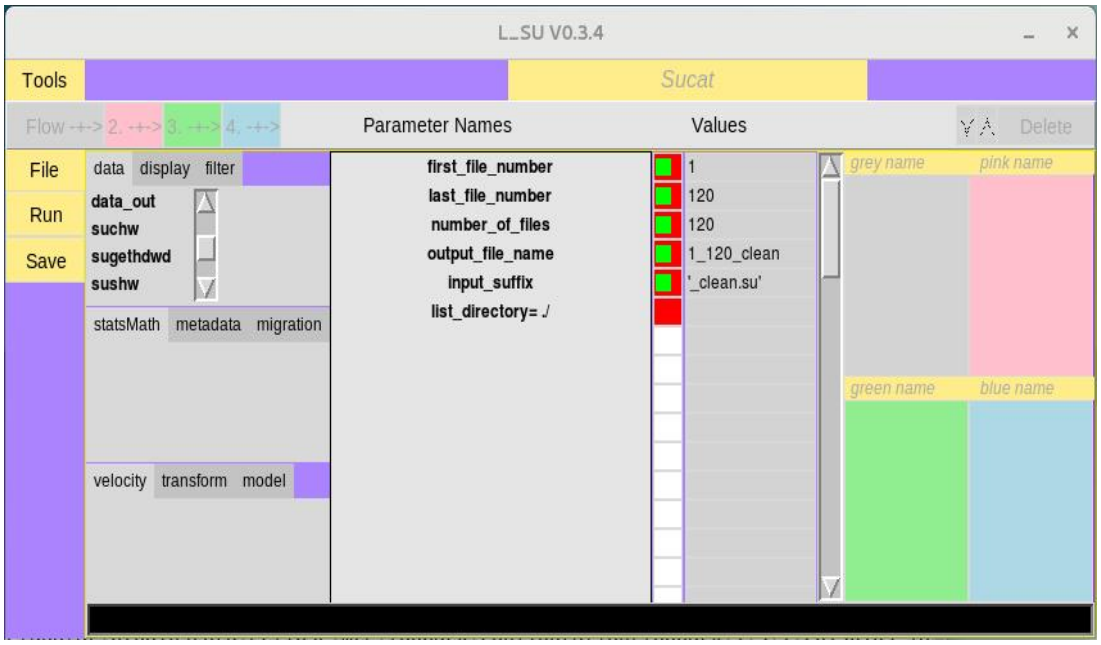
*tmax = 1*



# 3 L_SU Tools

### 3.1.1 How to paste a contiguous set of files into one larger file (Tool: Sucat)

### 3.1.2




### 3.1.9 How to slice your data ( suwind)

### 3.1.10 How to spike your dataset (suacor and supef)

### 3.1.11 How to examine the spectral characteristics of a data file interactively (Too: iSpectralAnalysis)