PMU DATA PROCESSING AND VISUALIZATION TOOL OVERVIEW AND USER GUIDE

This documents provides a user guide for the software tool that was created by the University of Illinois team to automate the process of translating data saved from PowerWorld, or a similar power system simulation package, into PMU like data and visualizations of it. The software tool is meant to provide a foundation for future researchers to easily develop new visualizations without having to worry about data formatting or processing. The data input and output from the software tool is well documented and standardized, such that the "wheel" does not need to keep being reinvented for each new visualization idea. In addition to providing a tool for future research, the software program also showcases the work completed by the University of Illinois team, with regards to visualization ideas. The following sections describe how to set up and use our open source software tool. Note that this user guide will likely change as improvements are made to the software too.

I. COMPUTER AND SOFTWARE SETUP

In order for the software tool to work, several software packages and installations are required, including a data science package called Anaconda, which contains Python, a graphical user interface tool for Python called Kivy, and other packages for mapping and plotting in Python. All these files can be found at the links given in each step, or in my GitHub repository, in the folder 'packages'. The software tool was written in Python.

1. Install Anaconda3-2.3.0 from their archives, here.
    - Kivy only works with Python 3.4, which the above installation of Anaconda uses. If/when Kivy supports 3.5, you can use the latest Anaconda installation, here.
    - Anaconda contains all the Python modules needed for data science, such as numpy, pandas, scikit-learn, and more.
2. Open a command prompt (Windows) or terminal window (Linux and OS X), and check if Python 3.4 is the default Python version:

    ```
    python --version
    ```

    If you do not see something with Python 3.4 listed, like:

    ```
    python 3.4.5 :: Anaconda 2.3.0 (x86_64)
    ```

Try typing:

```
python3 --version
```

From here on, use either python or python3, based on which one returned the correct version, Python 3.4.x, in the previous step.

3. Install Kivy for Python 3, from here.
   - Instructions for Linux, OS X, and Windows installations are found at the above hyperlink.
   - If 'python3' was the only successful option, in Step 2, replace 'python' with 'python3' and 'pip' with 'pip3' in the instructions given at the hyperlink.
4. Install the basemap package, which will be used for plotting on geographic maps.

   On OS X:

   ```
   conda install -c anaconda basemap=1.0.7
   ```

   After the basemap installation, you may need to set your bash profile to use UTF-8. To do this, type in a terminal window:

   ```
   nano ~./bash_profile
   ```

   Then, add the following statements, as two separate lines, anywhere in the text:

   ```
   export LC_ALL=en_US.UTF-8
   export LANG=en_US.UTF-8
   ```

   On Windows:

   Download the correct wheel for your computer from here, and save in a folder of your choice. If you have a 32-bit processor use the wheel that contains 'win32' and if you have a 64-bit processor use the wheel that contains 'amd64'.

   eg. basemap-1.0.8-cp34-none-win32.whl
   basemap-1.0.8-cp34-none-win_amd64.whl

   Open an Anaconda Command Prompt window, and navigate to the folder where you have saved the basemap wheel, something like the following:

   ```
   cd C:\Users\Name\Downloads
   ```

   Within the folder from above, make sure the basemap wheel is listed when you type:

   ```
   ls
   ```

   If you do, install the basemap wheel you downloaded by typing:

`pip install basemap-1.0.8-cp34-none-win_amd64.whl`

Note: If your wheel name is different than the above, substitute your specific wheel name, after "install"

Verify that basemap has been installed by checking if it is returned when you type: `pip list`

5. Because we installed an older version of Anaconda, so that Kivy would work with it, several packages need to be updated within the Anaconda package, namely pandas. To make the updates, in a terminal window or command prompt type:
   `conda update all` or to just update pandas `conda update pandas`
   - If you have issues, consult: https://github.com/conda/conda/issues/1967

II. FILE STRUCTURE

With the software installed. The next step is setting up the computer to store the PowerWorld files and their corresponding data. To do this, a folder hierarchy, within which all data will be stored, needs to be created.

First, create a folder called "cases". This folder will hold the actual PowerWorld files that will be used to create simulations. Note that PowerWorld files are referred to as cases. Figure 1, shows an example with three PowerWorld cases saved. Each PowerWorld case file has an extension of ".pwd".

Figure 1: Example file structure for storing PowerWorld case files.

Next, create a folder called "data". Within the "data" folder, create a folder for each case created and saved in the first step. For the example of Figure 1, that means three folders titled "case_name1", "case_name2", and "case_name3" are needed. Then, within each case directory, create a folder for each simulation you have made or intend to make and a folder named "case_info". Each simulation folder will hold data generated when a simulation is performed and the "case_info" folder will contain data about the power system model being used (case). Figure 2 illustrates the required file structure.

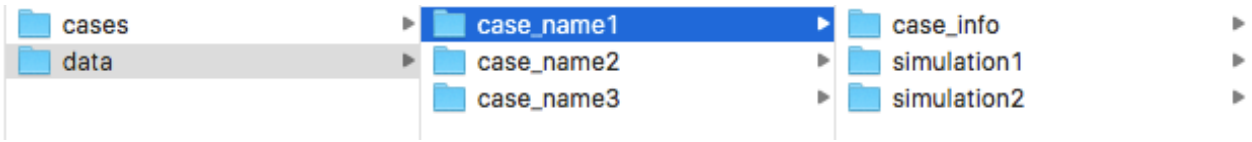

Figure 2: Example file structure for storing data, from each PowerWorld case file listed in figure 1.

Within each simulation folder and "case_info" folder, create a folder called "raw". You can also create a folder called "formatted" in each of those folders, as is shown in Figures 3 and 4, but it is not necessary, since the software tool will automatically create the "formatted" folder when it runs. Data from PowerWorld will later be saved directly into the "raw" folders.

Figure 3: Example file structure for storing case information from each PowerWorld case file listed in Figure 1.

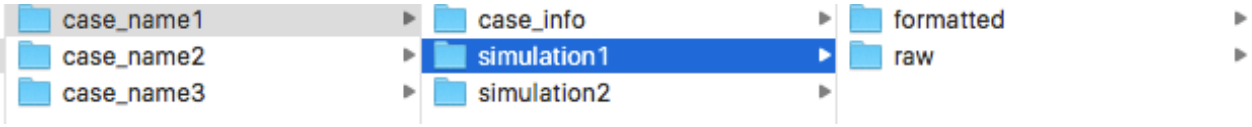

Figure 4: Example file structure for storing simulation data, for each simulation created in a PowerWorld case file.

III. USING POWERWORLD AND SAVING DATA FROM IT

It is assumed that at least one electric grid model has already been created. You can find an example case in my GitHub repository, in the folder "cases". The following instructions assume a basic working knowledge of PowerWorld. If you already have data saved in the format described in Section 4, you can skip this section.

3.1 Saving Case Data

Save information about your case (power system model), including data for buses, substations, and generators, to the folder "case_info/raw". This information is available by selecting "Model Explorer" from the "Case Information" tab in the ribbon, as shown in Figure 5.

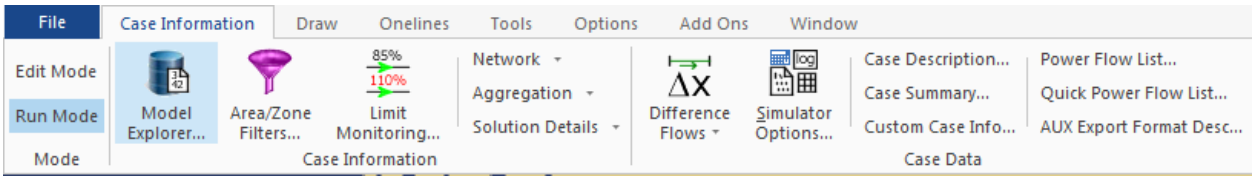

Figure 5: Screenshot of how to open the Model Explorer to access case (power system model) information.

Once the dialog box opens, select “Buses” from the list of options in the left pane, as shown in Figure 6. You will see lots of data in your version of PowerWorld, though I have whited out mine to ensure confidentiality. Make sure you have, at least, the columns shown in the selection box, towards the top of Figure 6. If not, right click within the data, and select “Display /Column Options…”. Select the missing column names from the dialog box that pops up. Then, to save the bus data select the icon shown in Figure 7, and select “Send All to Excel” from the drop down menu. In Excel, save the spreadsheet as a csv file, named “buses.csv” and specify the directory to save it to as “data/case_nameX/case_info/”, where you replace “case_nameX” with the name of your specific case. Repeat this process for “Generators” and “Substations”, where the column names required are “Bus Number”, “Area Name”, and “Gen MW” and “Sub Num”, “Area Name”, “Nominal kV”, “Latitude”, and “Longitude”, respectively. Name the data for generators “gens.csv” and for substations, “subs.csv”. When all case data has been saved, the folder layout will look similar to Figure 8.

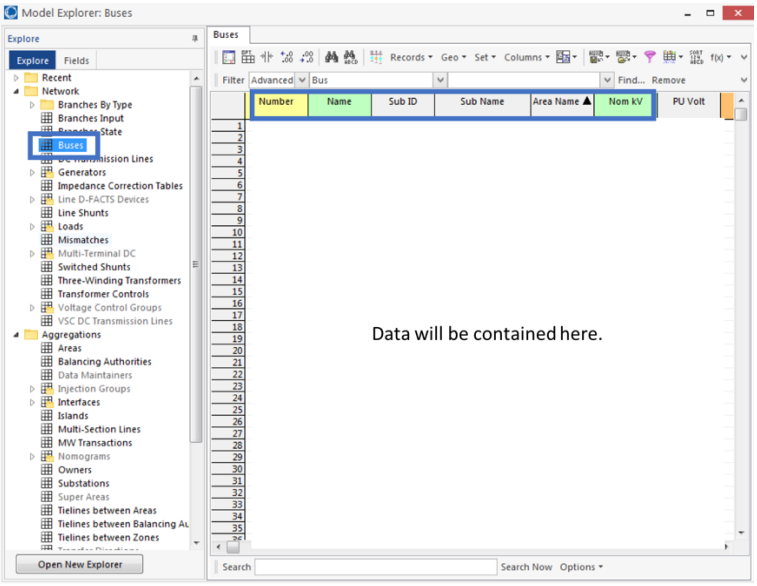

Figure 6: Bus data and the required columns.

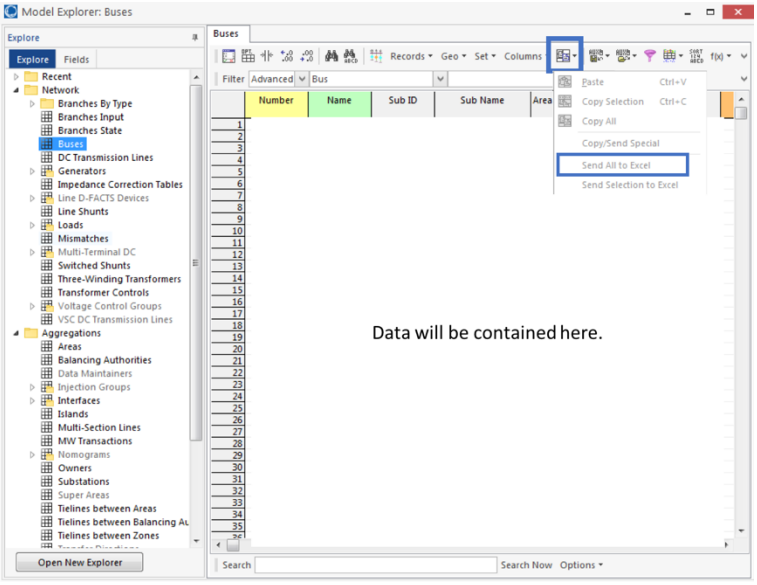

Figure 7: Screenshot of how to send bus data, for a case file, to Excel for saving.

Figure 8: Example of file structure for storing case (model) data for a PowerWorld case file.

3.2 Creating Transient Stability Simulations

This section describes how to create a transient stability simulation in PowerWorld. Within a PowerWorld case, open up a transient stability window. The transient stability tool is used to create the power system events for simulating earthquakes, physical equipment damage, etc. that will strain the power system model (case) and provide time stamped measurements at each bus.

To open the transient stability tool, select "Transient Stability" in the ribbon tab "Add Ons", as shown in Figure 9.

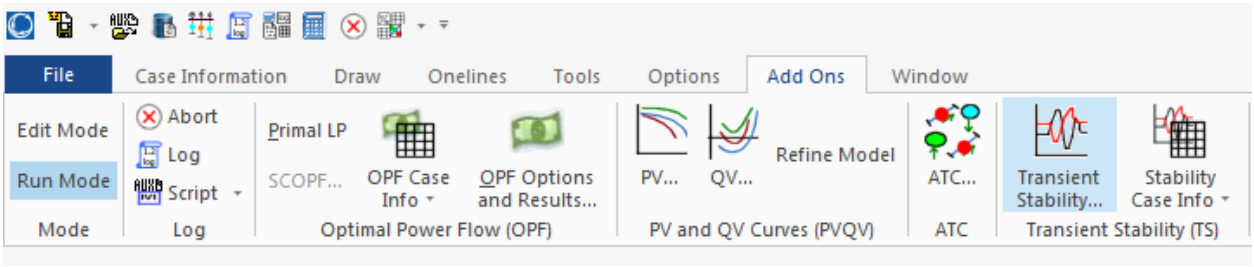

Figure 9: Screenshot of how to open the Transient Stability tool in PowerWorld.

A dialog box, like the one shown in Figure 10, will appear. In it, make a name for your new simulation, like “earthquake”, by selecting the button “Rename”, or if you already have a simulation and want to create another one, select “Add” and then “Rename”. Then, create a new transient stability definition by clicking on the button ‘Insert’.

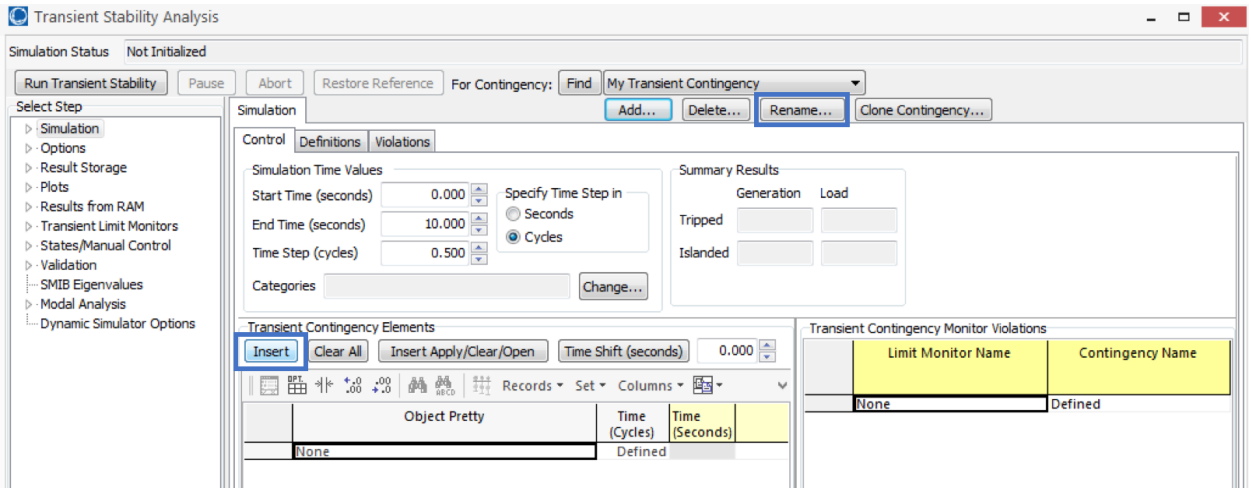

Figure 10: Screenshot of a transient stability home dialog box, with buttons highlighted for renaming or starting a new transient stability definition.

A new dialog box, like the one shown in Figure 11, will open. From this dialog box, you can select the type of equipment you want to affect in your simulation, in the far left pane, which exact element will be affected, in the central pane, and how it will be affected, in the bottom pane. For each piece of equipment you want to impact, repeat the process of pressing “Insert” and then completing the definition when the new dialog box opens. Note that a transient stability simulation will often have at least a few pieces of equipment that are being affected, though a simulation can be run without any equipment modifications. This can be especially helpful to get a baseline assessment of what is normal for the system.

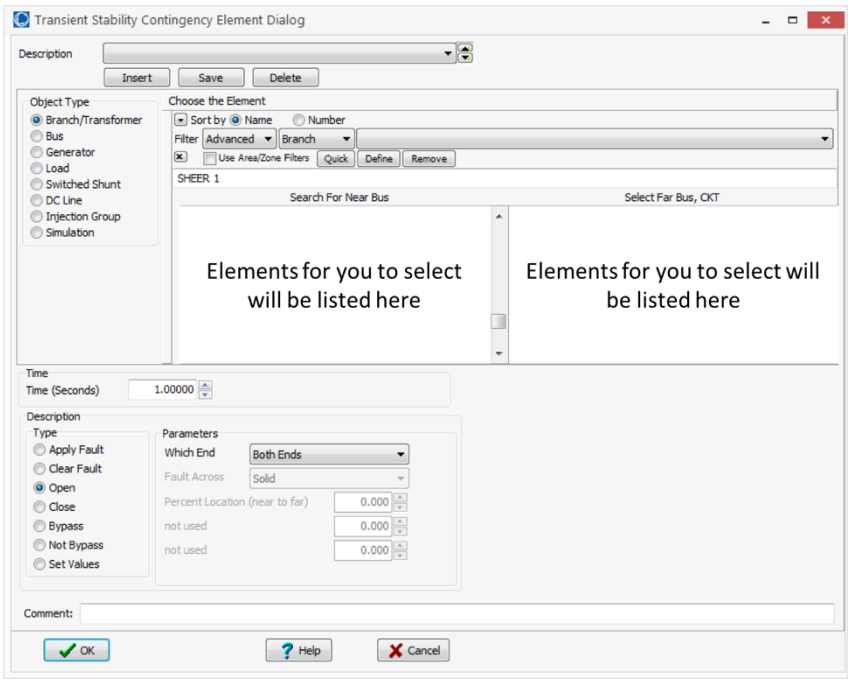

Figure 11: Screenshot of the dialog box used to create a transient stability event for a piece of equipment in the system.

When the transient stability simulation has been created, the next step requires configuring how the simulation should run and what data should be collected when it does. First, set up how long the simulation should run and what time step it should use. To do this, in the home dialog box of the Transient Stability tool, select "Simulation" → "Control", and then make your settings for the items outlined in blue of Figure 12. To collect data 30 times per second, as a PMU would, you can leave the time step at 0.5 seconds, for now.

Then, change how often data measurements are saved, by selecting "Results Storage" → "Store to Ram Options", and set the value for "Save Results Every n Time Steps". To mimic PMUs, set this value to 8, as shown in Figure 13.

Next, you need to make sure results are saved according to their bus number. To do this, select "Options" → "General" and then select the radio button "Number" on the far right, under "Identify Buses in Events and Results by", shown in Figure 14.

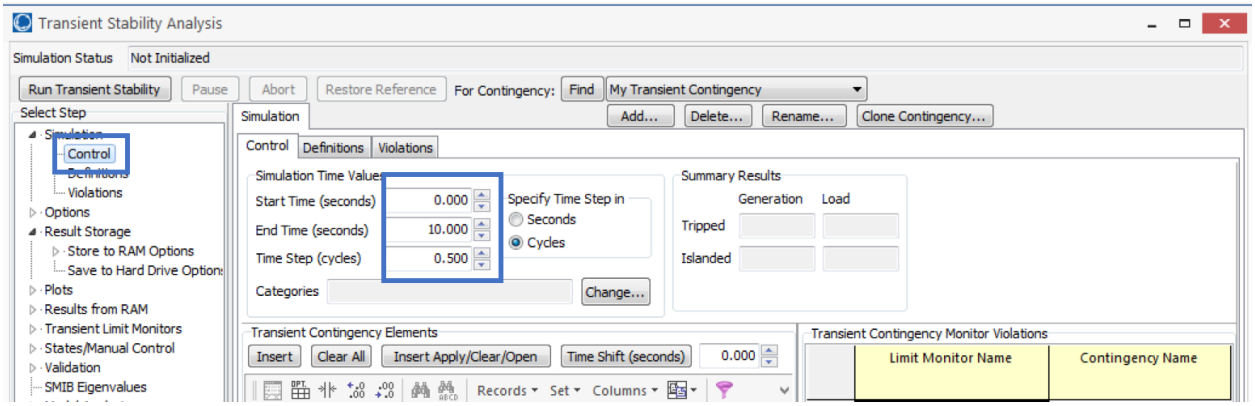

Figure 12: Screenshot of which buttons and text boxes to adjust to change transient stability simulation run-time and time step.

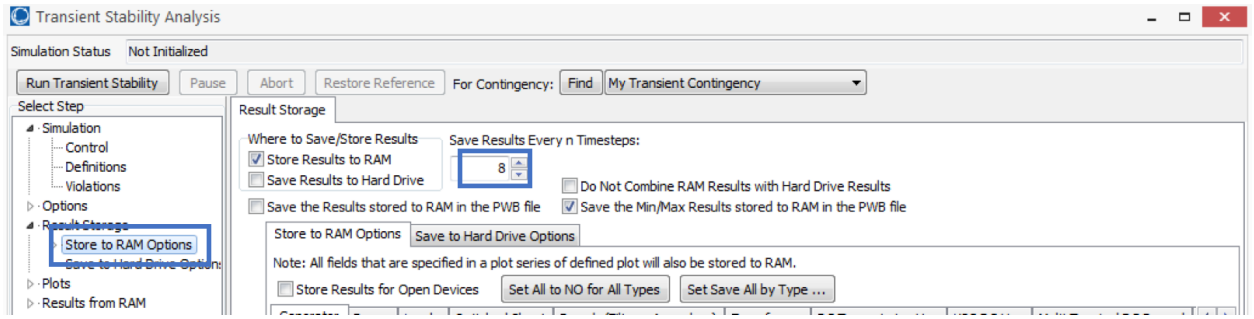

Figure 13: Screenshot of which buttons and text boxes to adjust to change often to record data measurements.

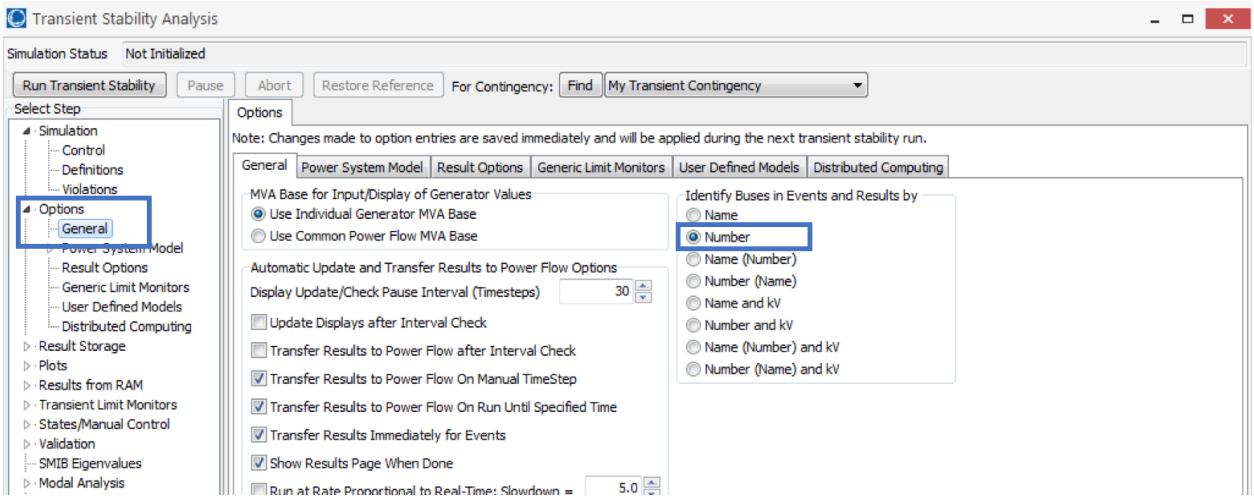

Figure 14: Screenshot of which buttons to press to make sure results are saved according to their bus number.

Lastly, you need to make sure that the correct measurements will be recorded. To replicate what PMUs would collect, we need measurements at buses for frequency, voltage angle, and voltage magnitude and in branches, the current magnitude. In the home transient stability dialog box, select "Results Storage" → "Store to Ram Options". First, select the "Bus" tab in the central

pane, shown in Figure 15. Then make sure all values in the columns “Save V pu”, Save V angle”, and “Save frequency” are set to “YES”. If not, right click somewhere within the column, and select “Set/Toggle/Columns” → “All YES”. Second, select the “Branch” tab in the central pane, shown in Figure 16, and make sure all values in the “Save Current To” column are set to “YES”. All other columns can be left as they are. Though, if any extras are set to “YES” the simulation will run more slowly and the computer memory will fill more quickly. Thus, it is best to set any unnecessary columns to all have values of “NO”.

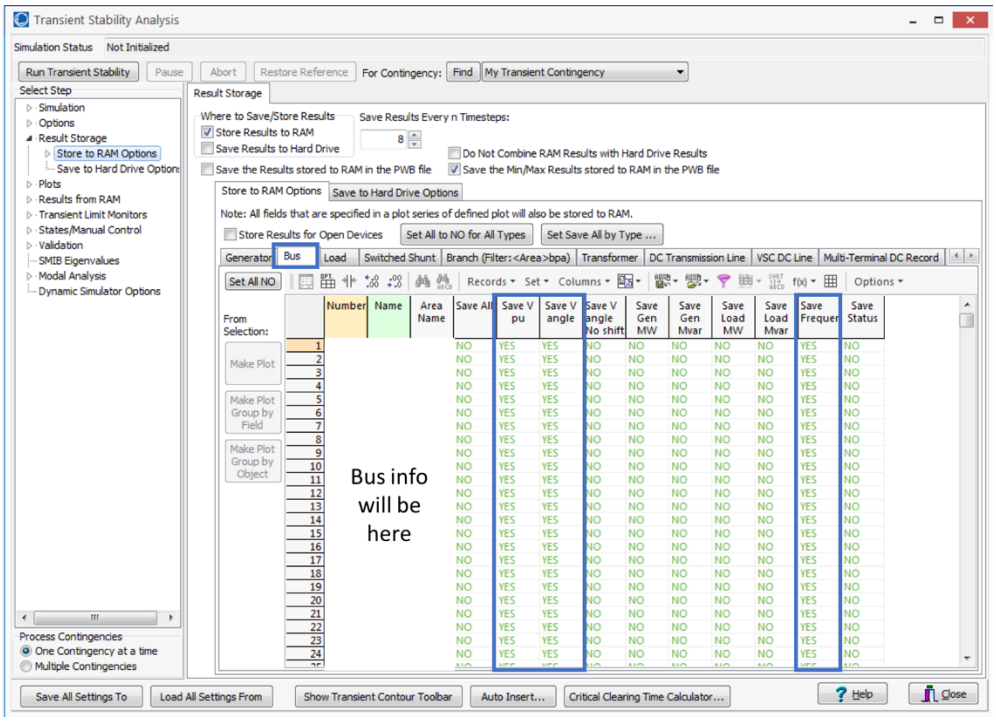

Figure 15: Screenshot of how to specify which measurements should be recorded for buses, during each simulation.

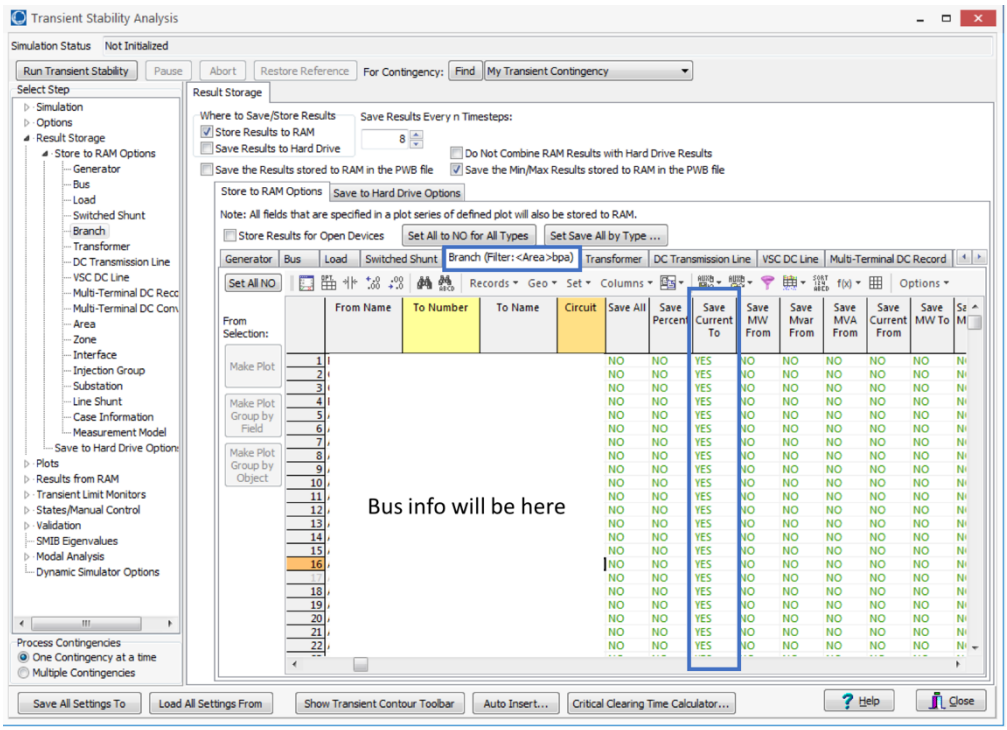

Figure 16: Specify which measurements should be recorded for every branch.

With all the aforementioned steps complete, the simulation can be run by selecting "Run Transient Stability Simulation" from the top of the transient stability home dialog box, as shown in Figure 17.

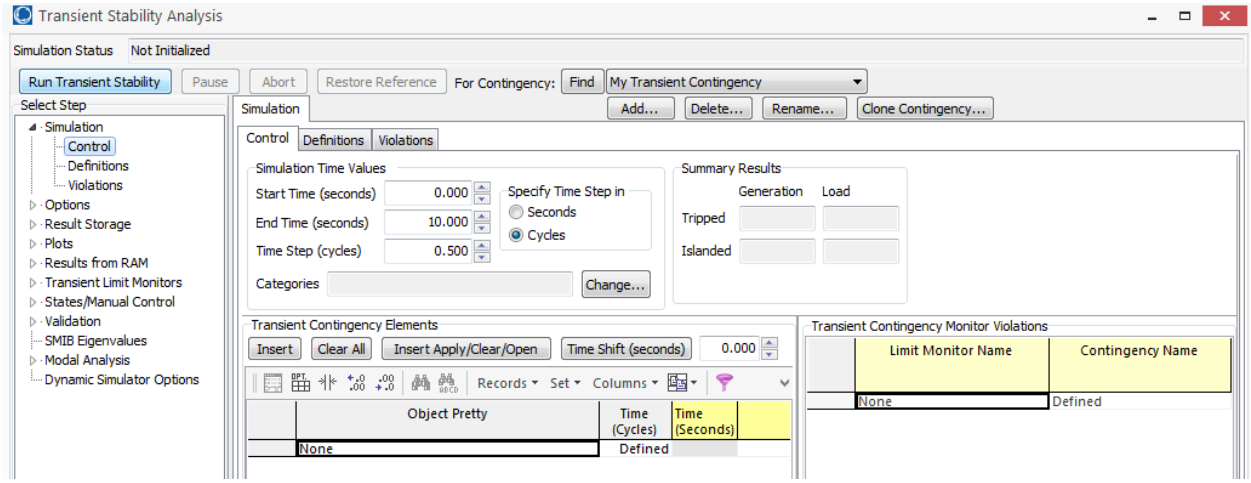

Figure 17: Screenshot of how to start a transient stability simulation.

3.3 Saving Transient Simulation Data

Once a simulation is complete, the data from it needs to be saved to csv files. To do this, within the home transient stability dialog box shown in Figure 10, on the left side, select "Results from

RAM" → "Time Values". Then, select the "Bus" tab. On the left side of the screen, select the measurement type you want to look at. First, select frequency. Then, click on the icon shown in Figure 18, and select "Send All to Excel" from the drop down menu. In Excel, save the file as "bus_freq.csv" in the folder "/data/case_nameX/simulationX/raw/". Repeat this for the measurement types "V angle" and "V pu", and save the files, in the same folder, as "bus_vang.csv" and "bus_vmag.csv", respectively. Note, if you want the actual kV value instead of the per-unit value, for voltage magnitude, simply select "V (kV)" instead of "V pu".

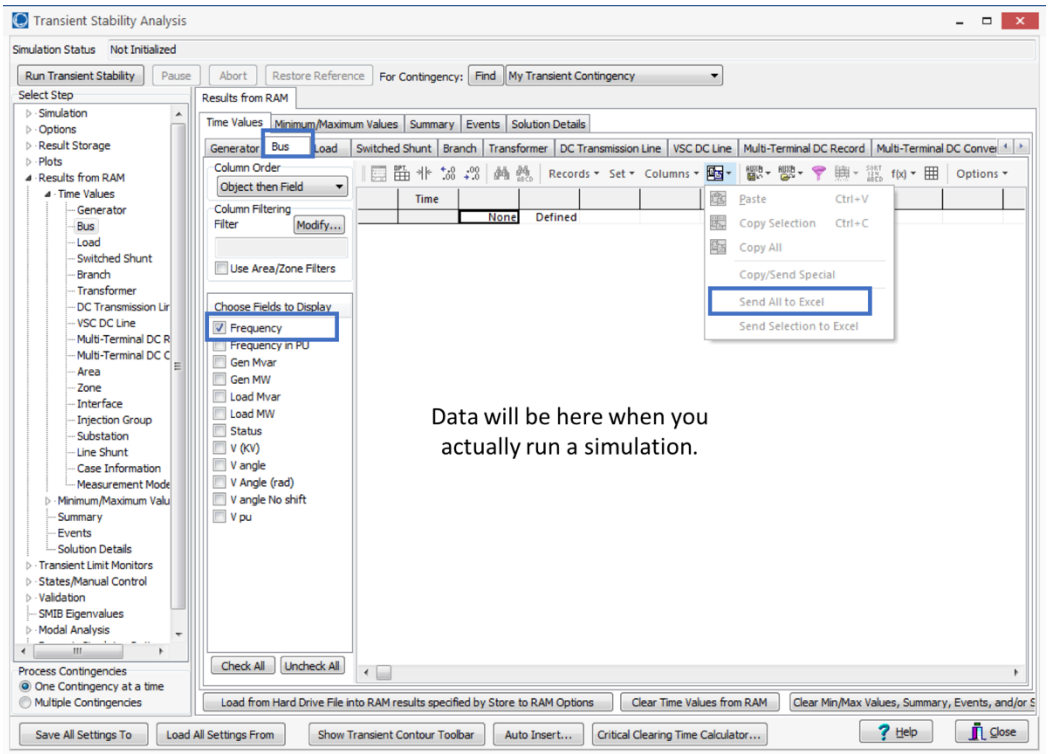

Figure 18: How to save frequency data from the transient stability simulation. Follow the same process to save voltage angle and voltage magnitude data, simply be selecting their corresponding checkbox, instead of "Frequency".

Then, in the same window, select the "Branch" tab. Select the measurement type of "Current To in PU". Then, click on the icon shown in Figure 19, and select "Send All to Excel" from the drop down menu. In Excel, save the file as "branch_cmag.csv" in the folder "/data/case_nameX/simulationX/raw/". Note, if you want the actual ampere value instead of the per-unit value, for current magnitude, simply select "Current To" instead of "Current To in PU". When all files have been saved, the data folder should look like the one shown in Figure 20.

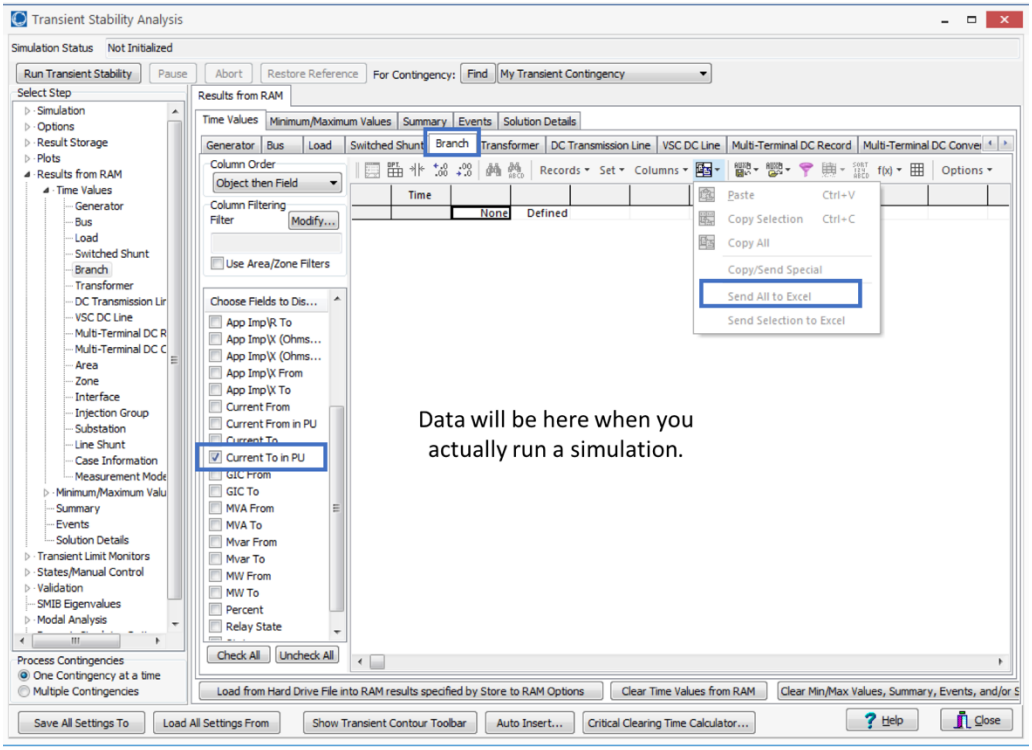

Figure 19: How to save current magnitude data from a transient stability simulation.

Figure 20: File structure for simulation data.

IV. DATA FORMATTING

This section provides an overview of the format for data that is input to the software tool, and data that is output from it. Note, if you have data already saved in the format required for input data, or can easily put it in that format, you may not need to use PowerWorld to generate any more data.

4.1 Input Data

This section describes the file format and location for data that is saved directly from PowerWorld and is the data that will be read in by the software tool. All data that is considered input data is stored in the "raw" folders, which are subdirectories of the "case_info" (Figure 8) and "simulationX" (Figure 20) folders. Input data is broken into two categories, Case Data and

Simulation Data, as is described in Table 1. A screenshot of the format of Case Data files is shown in Figure 21 and Simulation Data files in Figure 22. Note that no data is shown for any of the Case Data files, just the column headings, to ensure confidentiality.

Table 1: Data input to the software tool. Case data describes the PowerWorld file and the power system model contained within it, and simulation data contains the measurements recorded during a simulation.

File Name (.csv)	Type	Description
buses	Case	Contains bus name and number, voltage level, and substation identifier, for every bus in the system.
subs	Case	Contains substation identifier, operating area name, and geographic coordinates, for every substation in the system.
gen	Case	Contains bus number, MW generating capacity, and substation identifier for every generator in the system.
real_pmus	Case	If known, contains a list of substations where PMUs are actually located in a real system, and their corresponding voltage levels
bus_freq	Simulation	Frequency measurements, in Hz, recorded at every bus in the PowerWorld model
bus_vang	Simulation	Voltage angle measurements, in degrees, recorded at every bus in the PowerWorld model
bus_vmag	Simulation	Voltage magnitude measurements, in per-unit, recorded at every bus in the PowerWorld model
branch_cmag	Simulation	Current magnitude measurements, in per-unit, recorded coming in to every bus in the PowerWorld model

CASE DATA

buses.csv

Bus				
Number	Name	Sub Name	Area Name	Nom kV

gens.csv

Gen				
Number of Bus	Name of Bus	Sub Name of Bus	Area Name of Bus	Gen MW

real_pmus.csv (if any PMU substations are known at the beginning)

Sub Name	Nom kV

subs.csv

Substation				
Sub Num	Sub Name	Area Name	Latitude	Longitude

Figure 21: Screenshot of what each case data csv file looks like. Note that the first row can be omitted, where it says “Bus”, “Gen”, or “Substations” and the software tool will still work. The column headings of the second row need to be the same.

SIMULATION DATA

bus_freq.csv

1	TSTimePointResult					
2	Time	Bus 10292 Frequency	Bus 10318 Frequency	Bus 10319 Frequency	Bus 10320 Frequency	Bus 10321 Frequency
3	0	60	60	60	60	60
4	0.033333	60	60	60	60	60
5	0.066667	60	60	60	60	60
6	0.1	60	60	60	60	60

bus_vang.csv

1	TSTimePointResult					
2	Time	Bus 10292 V angle	Bus 10318 V angle	Bus 10319 V angle	Bus 10320 V angle	Bus 10321 V angle
3	0	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789
4	0.033333	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789
5	0.066667	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789
6	0.1	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789

bus_vmag.csv

1	TSTimePointResult					
2	Time	Bus 10292 V pu	Bus 10318 V pu	Bus 10319 V pu	Bus 10320 V pu	Bus 10321 V pu
3	0	1.029	1.007	1.0081	1.0331	1.0366
4	0.033333	1.029	1.007	1.0081	1.0331	1.0366
5	0.066667	1.029	1.007	1.0081	1.0331	1.0366
6	0.1	1.029	1.007	1.0081	1.0331	1.0366

branch_cmag.csv

1	TSTimePointResult				
2	Time	Line 30005 TO 40687 CKT 99 Current To in PU	Line 30020 TO 45035 CKT 99 Current To in PU	Line 30020 TO 45063 CKT 99 Current To in PU	Line 30245 TO 45063 CKT 99 Current To in PU
3	0	11.5918	7.5895	0.0234	0.0902
4	0.033333	11.5918	7.5895	0.0234	0.0902
5	0.066667	11.5918	7.5895	0.0234	0.0902
6	0.1	11.5918	7.5895	0.0234	0.0902

Figure 22: Screenshot of what each simulation data csv file looks like. Note that the first row can be omitted, where it says "TSTimePointResult" and the software tool will still work. The column headings of the second row need to be of the same format, though the exact bus or circuit numbers will be different.

4.2 Output Data

This section describes the file format and location for data exported from the software tool. All data that is considered output data is stored in the "formatted" folders, which are subdirectories of the "case_info" (Figure 8) and "simulationX" (Figure 20) folders. If the "formatted" folder does not exist in either the "case_info" or "simulationX" folders, the software tool will automatically create it. Output data is broken into two categories, Case Data and Simulation, PMU Data, as is described in Figure 2. A screenshot of the format of Case Data files is shown in Figure 23 and

Simulation Data files in Figure 24. Note that no data is shown for any of the Case Data files, just the column headings, to ensure confidentiality.

Table 2: Data export from the software tool. Case data describes the PowerWorld file and the power system model contained within it, and Simulation, PMU data contains the measurements recorded during a simulation, but only for a select number of buses being monitored by real or fictitiously assigned PMUs.

File Name (.csv)	Type	Description
bus_info	Case	Contains bus name and number, voltage level, substation identifier, and geographic coordinates of the substation, for every bus in the system.
gens	Case	Contains list of all buses at substations with a generator, total MW generating capacity, substation identifier, for every generating substation in a system
pmu_info	Case	Contains bus name number, voltage level, substation identifier, operating area name, and geographic coordinates for every bus/substation being monitored by a real or fictitious PMU
pmu_freq	Simulation, PMU	Frequency measurements, in Hz, recorded at every bus being monitored by a PMU
pmu_vang	Simulation, PMU	Voltage angle measurements, in degrees, recorded at every bus being monitored by a PMU
pmu_vmag	Simulation, PMU	Voltage magnitude measurements, in per-unit, recorded at every bus being monitored by a PMU
pmu_cmag	Simulation, PMU	Current magnitude measurements, in per-unit, recorded coming in to every bus being monitored by a PMU

CASE DATA

bus_info.csv

Bus Number	Bus Name	Sub Name	Nom kV	Area Name	Latitude	Longitude

gens.csv

Sub Name	Area Name	Total Gen MW	Num Gens	Bus Numbers

pmu_info.csv

Bus Number	Bus Name	Sub Name	Nom kV	Area Name	Latitude	Longitude

Figure 23: Screenshot of what each case data csv file looks like, after being export from the software tool.

SIMULATION DATA

pmu_freq.csv

1	Time	40045	40714	40051	40718	40719	40716
2	0	60	60	60	60	60	60
3	0.033333	60	60	60	60	60	60
4	0.066667	60	60	60	60	60	60
5	0.1	60	60	60	60	60	60

pmu_vang.csv

1	Time	40045	40714	40051	40718	40719	40716
2	0	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965
3	0.033333	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965
4	0.066667	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965
5	0.1	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965

pmu_vmag.csv

1	Time	40045	40714	40051	40718	40719	40716
2	0	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909
3	0.033333	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909
4	0.066667	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909
5	0.1	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909

pmu_cmag.csv

1	Time	Line 40045 TO 40601 CKT 1	Line 40045 TO 40774 CKT 1	Line 40045 TO 40821 CKT 2	Series Cap 40051 TO 40714 CKT
2	0	6.5926	5.59	4.8535	9.5533
3	0.033333	6.5926	5.59	4.8535	9.5533
4	0.066667	6.5926	5.59	4.8535	9.5533
5	0.1	6.5926	5.59	4.8535	9.5533

Figure 24: Screenshot of what each Simulation, PMU data csv file looks like, after it has been exported from the software tool. Columns are distinguished by their bus number.

V. DESCRIPTION OF SOFTWARE TOOL AND HOW TO USE IT

The software tool is a means to automate the process of turning PowerWorld simulation data into PMU data visualizations. Key functions of the software tool are as follows:

- If PMU locations are not known for a system, the program can create a list of substations that would likely have a PMU installed
- PMU data for frequency, voltage angle, and voltage magnitude can be processed

- The program uses a well-defined, documented, and unchanging format for data input and output from the program. This standardization allows for new modules to be built without having to reinvent the wheel each time.
- The program is written in Python, is all open source, and is hosted on GitHub.
- User interaction with the tool is pretty intuitive. More advanced settings can be changed in the actual code, and areas that lend themselves to those changes are well commented.
- K-means and DBSCAN clustering is built into the tool.
- Several visualizations of the data can be created.

Once you have saved the PowerWorld data, you then need to run a script to format the data properly for visualization. To run that script, you first need to make some adjustments to the 'userdef.py' file, so that the variables definitions in that file fit your system. If your power system model contains multiple area names, and you only want data for certain areas, specify the area names of interest, in the list variable 'SYSTEM_AREA_NAMES' in the userdef.py file. Names should be strings that match the case and spacing of the area names given in the file 'buses.csv'. Next, you should specify the number of PMUs you want to have monitoring data in your system, by setting the variable 'NUM_SUBS_WITH_PMU', in the 'userdef.py' file. Lastly, set the voltage levels you want to record data from. These are usually the two highest voltages in the system. Once you have made the adjustments (similar to what is shown in Figure 25), save the 'userdef.py' file.

With those settings complete, you can then run the data formatting script, 'dataprocessing.py'. To do this, you may either use your choice of Integrated Development Environment (eg. PyCharm, Spyder, CodeRunner) or launch the script from the command line. Using an Integrated Development Environment, open the file "dataprocessing.py". Figure 26 shows how to do this using CodeRunner, by opening the "dataprocessing.py" file and then hitting the "Run" button. Figure 27 shows how to run the tool in PyCharm, where you need to open the file "dataprocessing.py", right click on the "dataprocessing.py" tab, and then select "run dataprocessing.py" from the drop down menu.

```
Functionality:
- This module contains user specifications for filtering data,
  and values that should remain constant to use the application.
  If no user definitions are made, the following defaults are used:

  NUM_SUBS_WITH_PMU = 10% of total number of buses
  SYSTEM_AREA_NAMES = all areas
  PMU_VOLT_LVLS = two highest voltage levels in case
  COLOR_PALLETTE = 'ggplot'
"""

### USER SPECIFICATIONS
'''
   Change these variables as you like.
   The program will try to make assumptions so it
   can run, just in case you enter crazy things.
'''

# List of area names that should be considered
SYSTEM_AREA_NAMES = ['TN']

# Number of substations that have a PMU
NUM_SUBS_WITH_PMU = 25

# Specify voltage levels where PMUs likely located
PMU_VOLT_LVLS = ['230', '500']
```

Figure 25: Screenshot of variables to adjust in the userdef.py file.

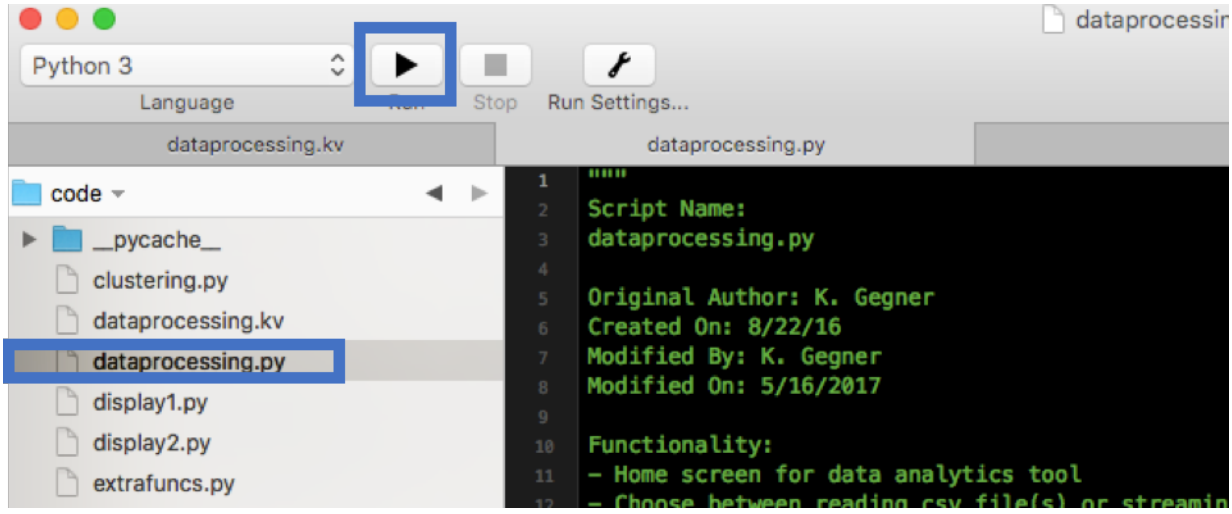

Figure 26: Screenshot of how to run the 'dataprocessing.py' tool using CodeRunner.

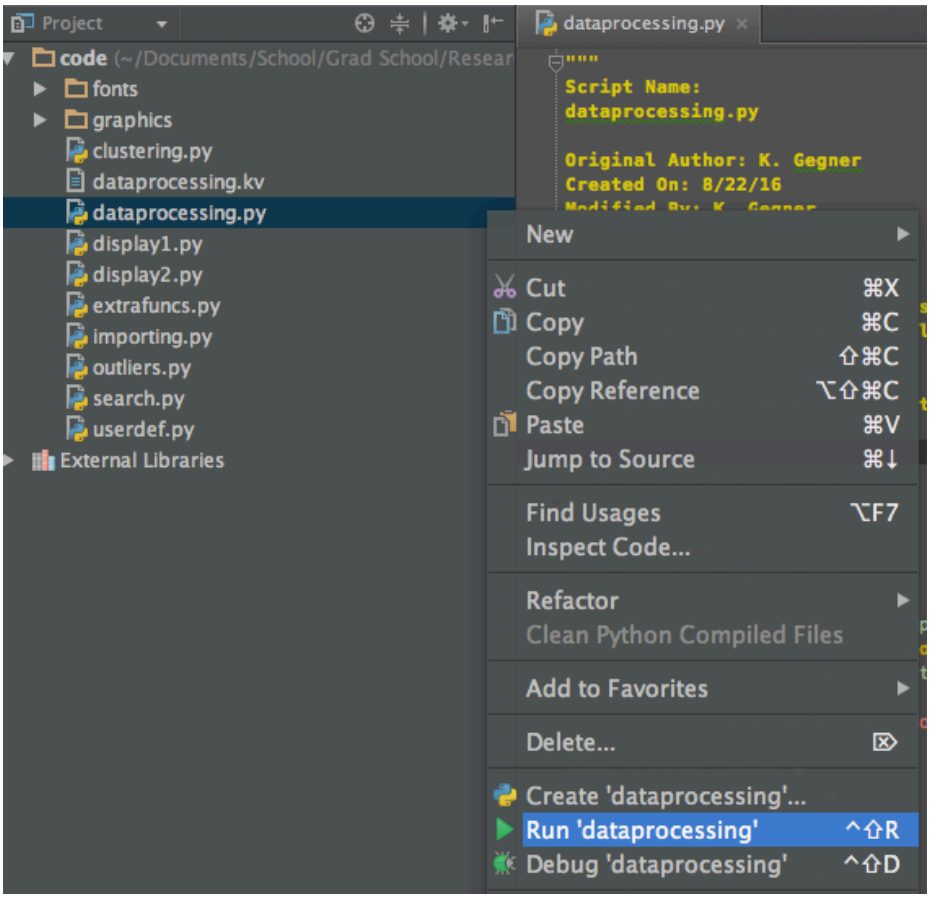

Figure 27: Screenshot of how to run the 'dataprocessing.py' file using PyCharm.

Alternatively, you can launch the software tool from a terminal window, by navigating to the folder where the file "dataprocessing.py" is saved. Using the file format described above, the file should be located in the folder hierarchy shown in Figure 28. Figure 29 summarizes the navigation process and how to launch the dataprocessing.py script, by typing:

```
python analytics.py
```

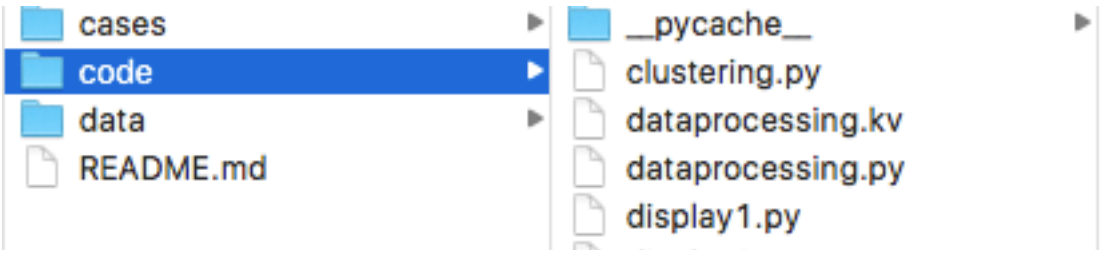

Figure 28: Folder hierarchy to where the 'dataprocessing.py' file is stored.

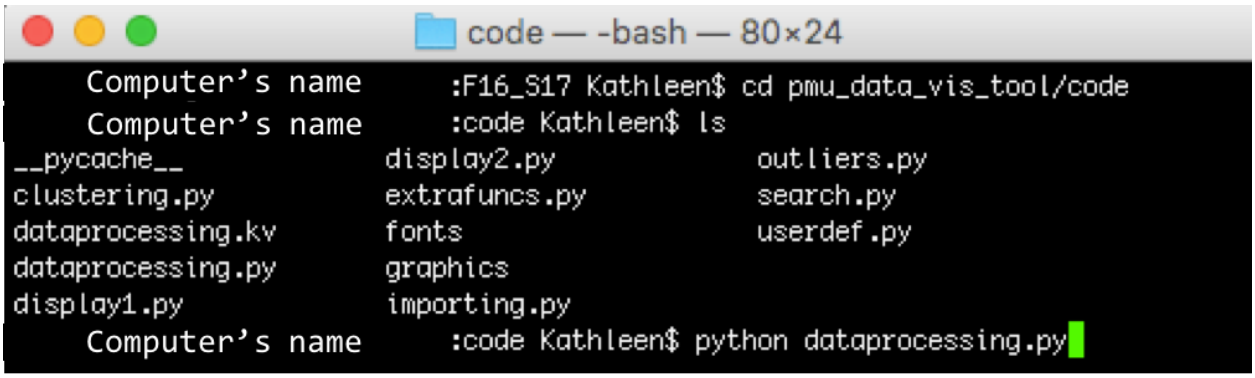

Figure 29: Screenshot of how to navigate to the 'dataprocessing.py' file folder, and how to run the script.

Once the software tool has been launched, a screen like Figure 30 will appear. Select "Load CSV File". A new screen will pop up, like that in Figure 31. Select the folder (one-click) that contains the simulation data you want to use. Sub-folders are accessed by selecting the '>' icon next to each folder name. Once you have selected the appropriate folder, like is shown in Figure 31, click the button 'Load'. Progress will be shown in the console, as shown in Figure 32. To begin the visualization process, copy the last couple lines from the console into the 'userdef.py' file, as shown in Figure 33.

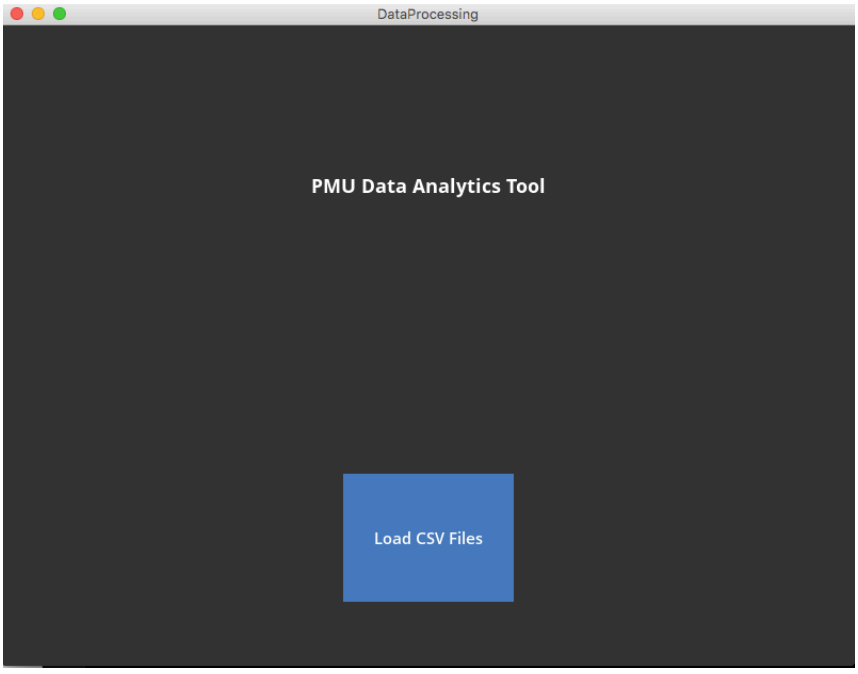

Figure 30: Analytics tool home screen.

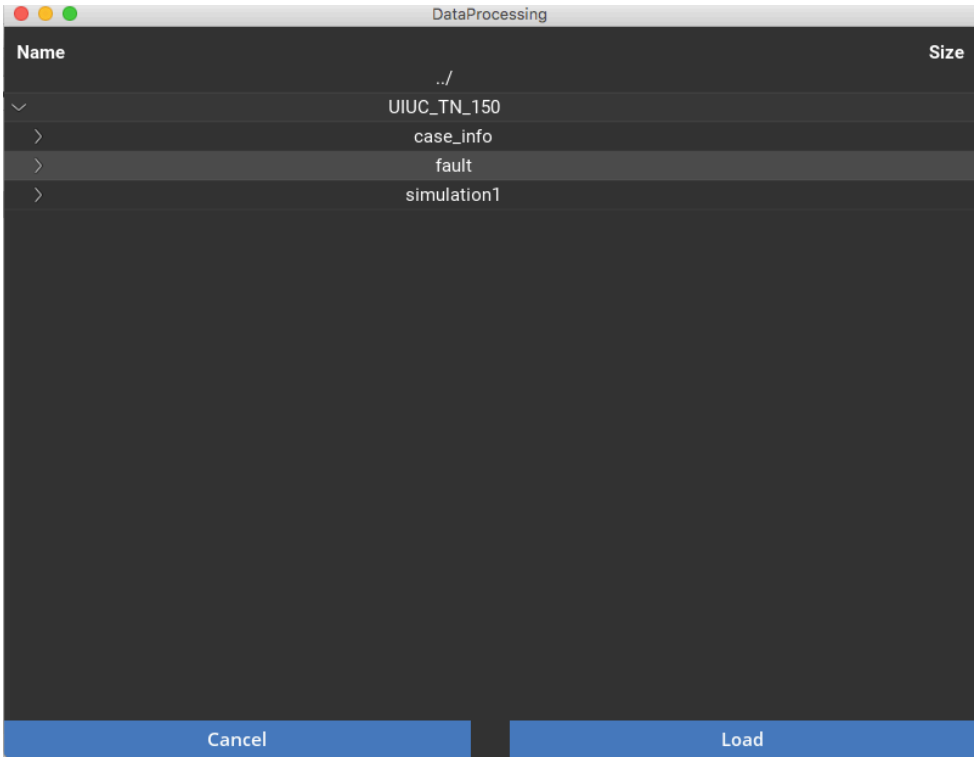

Figure 31: Data processing tool folder selection screen. To view sub-folders, click on the '>' icon next to a folder's name.

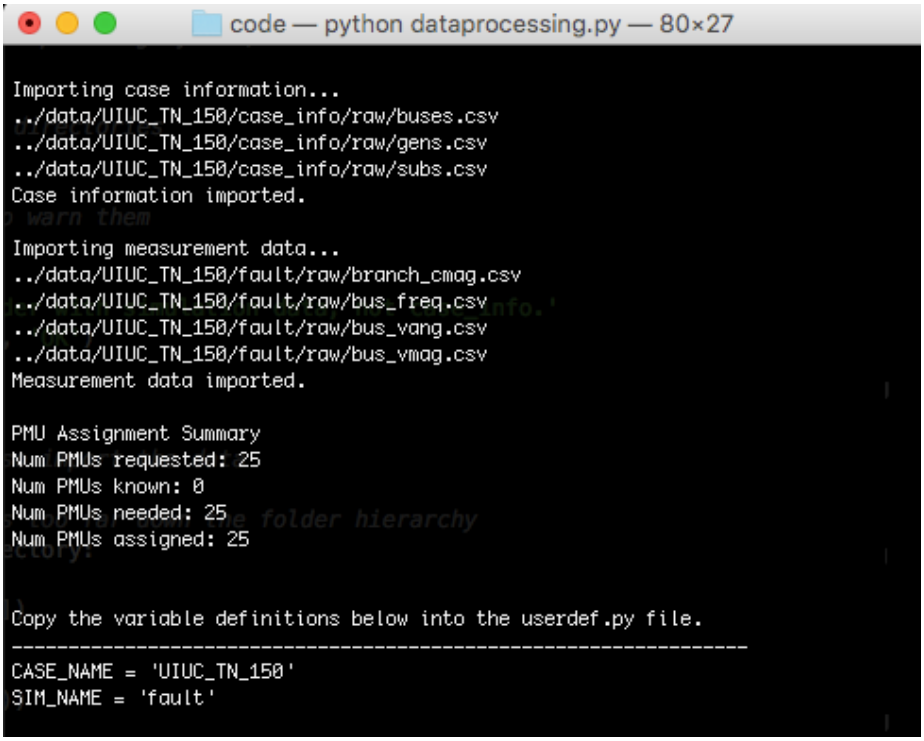

Figure 32: Data processing tool console output. Copy the last two lines into the file, userdef.py, so the newly data can be visualized.

With the data formatted, the next step is to visualize the data. As before, the userdef.py file needs to first be adjusted. The first two variables that need to be set are copied from the output on the console after running 'dataprocessing.py'. Those variables are 'CASE_NAME' and 'SIM_NAME'. The third variable is called MAP_REGION, which is a geographic region that can be chosen from a list of options, in MAPS, toward the bottom of the 'userdef.py' file. Alternatively, if the geographic region you need is not included, you can define your own region in the MAPS variable. Follow the format that is used, and specify the lower left latitude and longitude coordinate of the geographic area you want to map, and the upper right latitude and longitude corner. Figure 33 illustrates the above.

```
# Case name and simulation name, as copied from the console
# after running dataprocessing.py
CASE_NAME = 'UIUC_TN_150'
SIM_NAME = 'fault'

# Map region as selected from the MAPS dictionary below
# eg. MAP_REGION = 'tn'  # for the state of tennessee
MAP_REGION = 'tn'
```

```
# Define lat/long boundary coordinates for regions of north america and the u.s.
# na = north america, us = united states, bpa = bonneville power administration
MAPS = {'na': {'ll_lat': 25, 'll_long': -169, 'ur_lat': 71, 'ur_long': -55},        # north america
        'na_west': {'ll_lat': 30, 'll_long': -141, 'ur_lat': 60, 'ur_long': -102},  # western north america
        'na_mid': {'ll_lat': 25, 'll_long': -106, 'ur_lat': 60, 'ur_long': -86},    # midwest north america
        'na_east': {'ll_lat': 30, 'll_long': -88, 'ur_lat': 60, 'ur_long': -54},    # eastern north america
        'us': {'ll_lat': 25, 'll_long': -125, 'ur_lat': 50, 'ur_long': -66},        # continental united states
        'us_west': {'ll_lat': 31, 'll_long': -125, 'ur_lat': 50, 'ur_long': -102},  # western continental united states
        'us_mid': {'ll_lat': 25, 'll_long': -106, 'ur_lat': 50, 'ur_long': -89},    # midwest united states
        'us_east': {'ll_lat': 30, 'll_long': -92, 'ur_lat': 50, 'ur_long': -66},    # eastern united states
        'us_pnw': {'ll_lat': 40, 'll_long': -125, 'ur_lat': 50, 'ur_long': -103},   # united states pacific northwest
        'us_ne': {'ll_lat': 39, 'll_long': -81, 'ur_lat': 48, 'ur_long': -66},      # united states northeast
        'us_se': {'ll_lat': 25, 'll_long': -107, 'ur_lat': 40, 'ur_long': -75},     # united states southeast
        'bpa': {'ll_lat': 41, 'll_long': -125, 'ur_lat': 50, 'ur_long': -111},      # bonneville power admin. footprint
        'il': {'ll_lat': 36, 'll_long': -92, 'ur_lat': 43, 'ur_long': -87},         # state of illinois
        'tn': {'ll_lat': 34, 'll_long': -92, 'ur_lat': 38, 'ur_long': -80},}        # state of tennessee
```

Figure 33: Userdef.py variables to be adjusted to set up the program for creating visualizations.

Once the settings have been adjusted, in the 'userdef.py' file the visualizations can be created. The visualization scripts are called 'display1.py' and 'display2.py'. As before, you can launch these from an integrated development environment, see Figure 34-35 or from a terminal console. For the terminal console, you will need to navigate to the correct folder, as you did before to get to 'dataprocessing.py'. Figure 36 shows this.

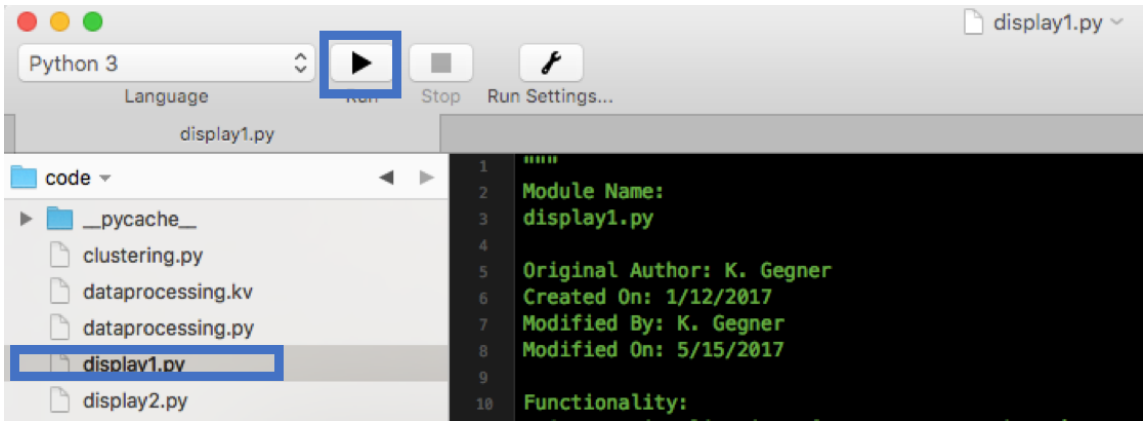

Figure 34: Screenshot of how to launch the 'display1.py' file using CodeRunner.

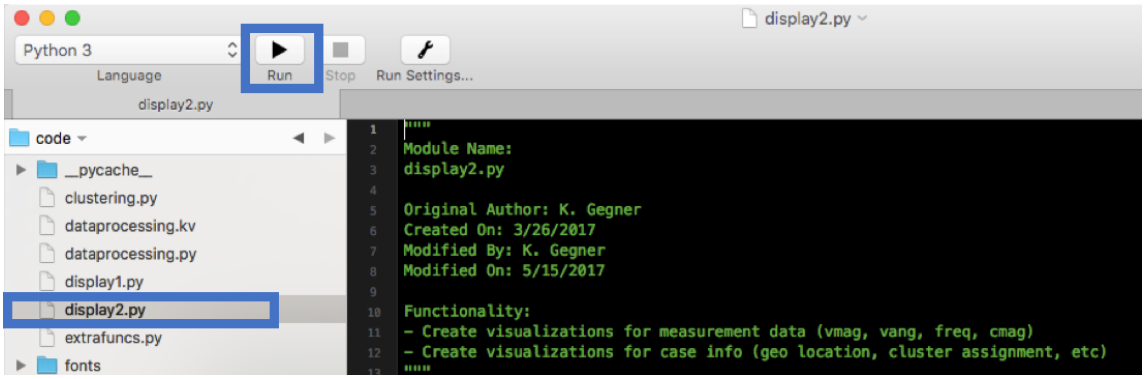

Figure 35: Screenshot of how to launch the 'display2.py' file using CodeRunner.

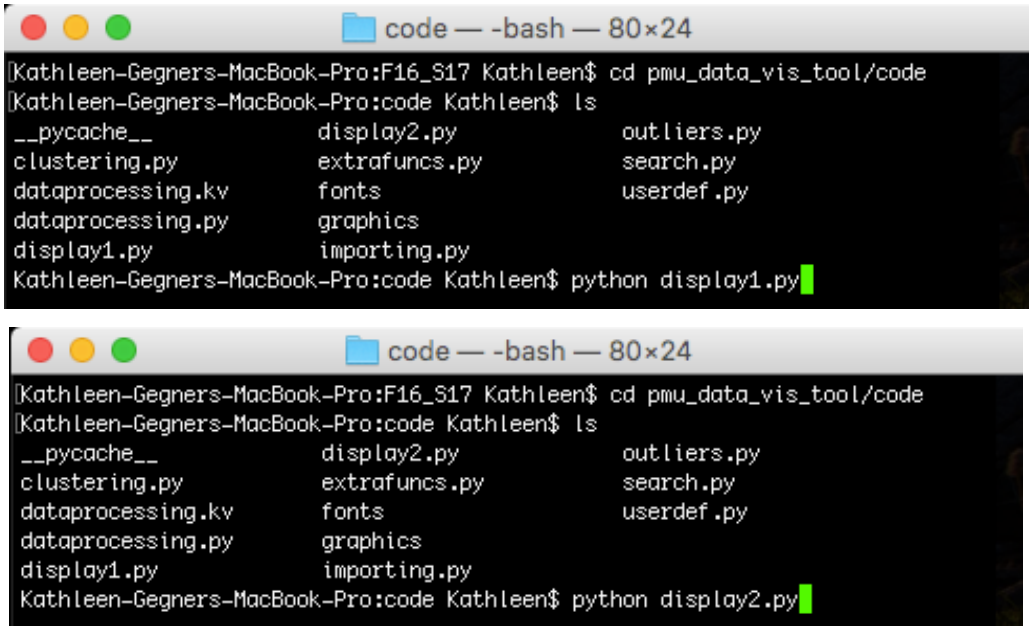

Figure 36: Screenshot of how to run the 'display1.py' and 'display2.py' files using a terminal console.

The interactive visualization output from the ‘display1.py’ file is shown in Figure 37. Within the interactive visualization window, the user can draw a rectangle over an area in the geographic map to see specific line plots for the buses selected by the user. To draw a rectangle, click and hold the left mouse button and drag it to the final point you want and release. A rectangle will not be shown, but know that the program is working. Within a second or so, the line plots will appear in the last row of the visualization.

The interactive visualization output from ‘display2.py’ is shown in Figure 38. Within the interactive window, the user can select which bin of a histogram they want to investigate by clicking anywhere on the histogram plot, as long as it is within the x values of the bin. When they do so, the line plots and geographic plots will be updated to show the data for only the buses that are contained in the user selected bin.

To quit the program, simply press the red “X” button, at the top right corner of the application window.

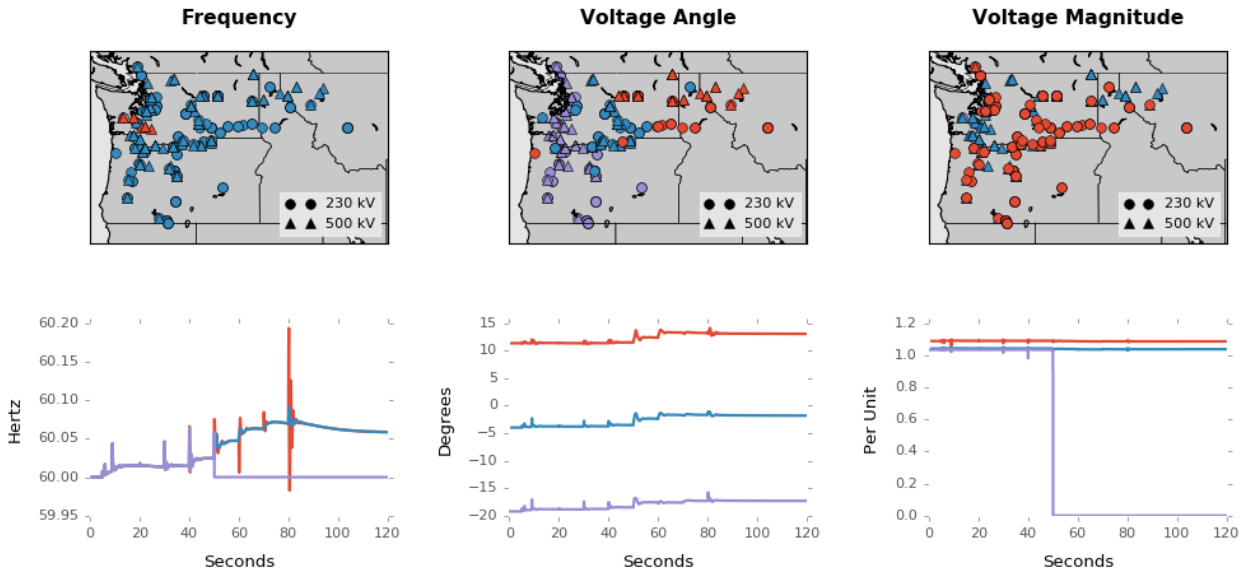

Figure 37: Analytics tool interactive visualization pane for geographic searching.

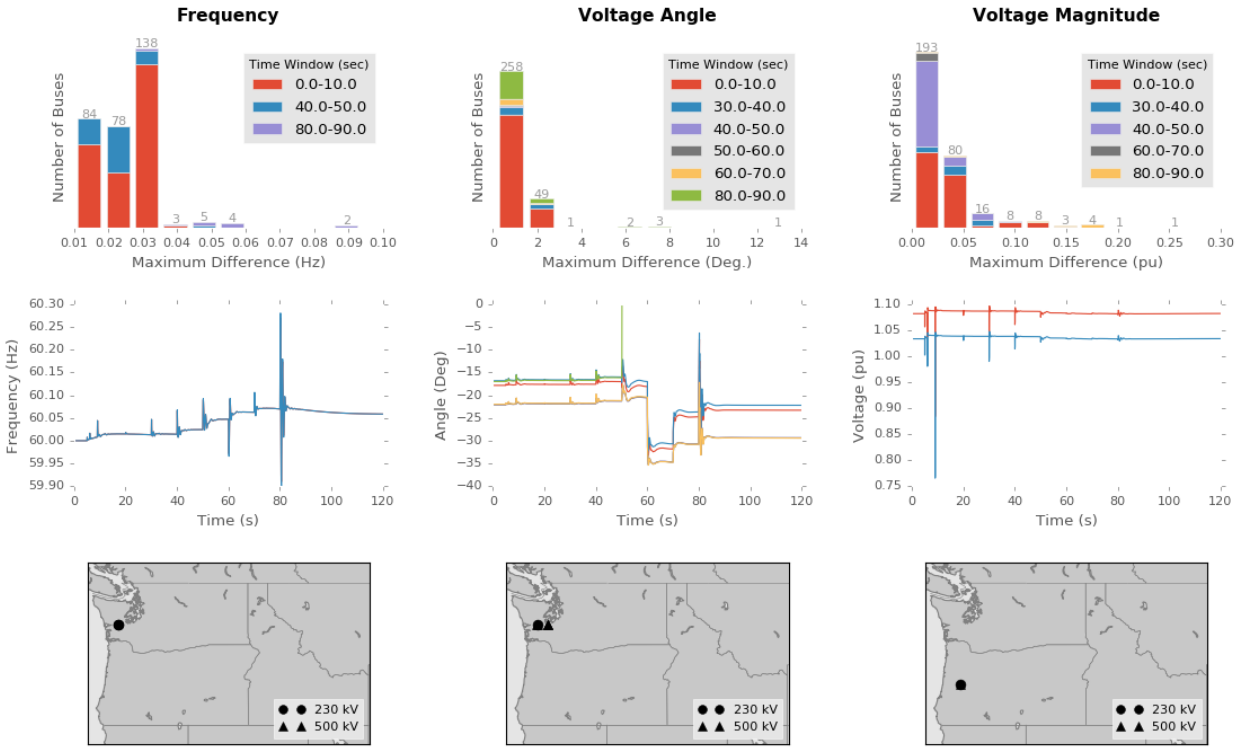

Figure 38: Analytics tool interactive visualization pane for searching by greatest measurement variation.