**PCA Analysis Manual**

**Section 1: Software Setup and Data Acquisition (Before You Run)**

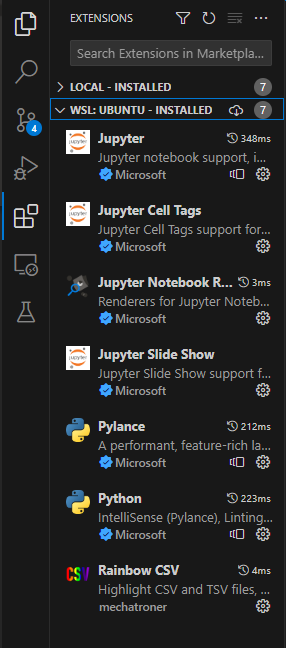
1. **Preliminaries** The PCA analysis software has been tested most recently using Python 3.8.10 on a Windows 10 machine with the following packages installed:

|  |  |  |
| --- | --- | --- |
| **Package Name** | | **Version Tested** |
| sklearn | | 1.3.0 |
| python-docx | | 0.8.11 |
| pandas | | 2.0.3 |
| numpy | | 1.24.4 |
| matplotlib | | 3.7.2 |
| customtkinter | | 5.2.0 |
| scipy | | 1.10.1 |
| (optional) ipykernel | 6.25.1 | |
| (optional) torch | 2.0.1 | |
| (optional) torchvision | 0.15.2 | |
| (optional) optuna | 3.3.0 | |
| (optional) tqdm | 4.66.1 | |
| (optional) seaborn | 0.12.2 | |

***Table 1***

(The optional packages are only required for some of the Jupyter notebooks in the repository, which are used to test future functionality.) Follow the instructions below to set up Python and the necessary packages on your PC:

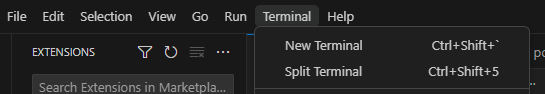
* Download and install Visual Studio Code here: <https://code.visualstudio.com/> (or use your preferred IDE)
* Open up Command Prompt or PowerShell and install WSL with Ubuntu as the chosen Linux distribution using the following commands:
  + wsl --install
  + wsl --install -d Ubuntu
  + wsl -l -v (to check version)
  + wsl --set-version Ubuntu [insert either 1 or 2; see below]
    - Note: There's a bit of a tricky situation with WSL versions 1 and 2. Version 1 won't allow you to run code in the PCA analysis software that shows user interface elements (main\_test.py), and version 2 can't connect to the internet over a VPN. If you're on a normal WiFi network or can temporarily disconnect from your VPN, you should be able to download Python packages and do version control without a problem, so just set the version to version 2. If you are required by your organization to use a VPN at all times, you will have to either only run code that uses the command-line interface (main.py) over WSL 1 or use a separate Git Bash terminal to clone the repository and pull changes on WSL 2.
* **Note:** To start and stop wsl, respectively, you can type the following commands in CMD or PowerShell:
  + wsl
  + wsl --terminate Ubuntu
* Install the WSL + Python + Jupyter extensions in VSCode using the extensions store:



* + (Make sure to install the extensions under the WSL: Ubuntu dropdown you see above and not Local)
* Connect to WSL in VSCode
  + Find the blue icon in the bottom left-hand corner of the screen:



* Then select "Connect to WSL" in the drop-down menu that appears (you can also access the command palette by using Ctrl+Shift+P)
* Now open a terminal in WSL by selecting Terminal -> New Terminal:



* In your terminal, install pip with the following commands:
  + sudo apt update
  + sudo apt upgrade
  + sudo apt install pip
* Now install Python (version 3.8.2 or later should be used):
  + pip install python
* Create a virtual environment:
  + How to create a virtual environment using virtualenv:

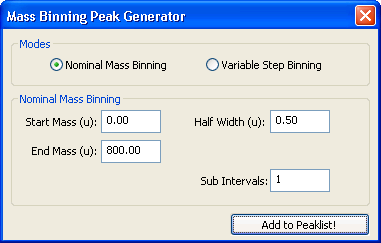
<https://learnpython.com/blog/how-to-use-virtualenv-python/>

* To summarize, the following commands create, activate, and deactivate a virtual environment named "virtenv":
  + - virtualenv -p python3 virtenv
    - source virtenv/bin/activate
    - deactivate
  + You just need to execute the first two for now
* In your virtual environment, install all necessary Python packages by applying the following command for each package selected from Table 1:
  + pip install [PACKAGE NAME]
  + If on the PNNL VPN, use the following command option to install packages over a PNNL proxy to resolve SSL issues:
    - pip install [PACKAGE NAME] --proxy=http://proxy01.pnl.gov:3128
* If you can’t install packages using the method above, you’ll have to install them manually. As an example, suppose you want to download python-docx. First,
  + Google the package name (it can be found at <https://pypi.org/project/python-docx/>)
  + Then manually install this package by typing python3 -m pip install ./downloads/python-docx-0.8.11.tar.gz in the Terminal (Use the correct path of the file)
* Now, you can clone (i.e., make a local copy of) the repository where the PCA software is stored by doing the following:
  + Download Git here: <https://git-scm.com/download/win>
  + Run git clone <https://gitlab.com/pacific-northwest-national-laboratory/ATOFSIMSCLASS/pca-analysis> in your Terminal or in Git Bash
* Finally, if you desire to run some of the test notebooks in the repository, you can set up Jupyter notebooks to work in a virtual environment:
  + After activating virtual environment, enter the following commands in your Terminal:
  + pip install ipykernel
  + ipython kernel install --user --name=[virtual environment name]

1. **Getting the Data from Surface Lab** Generate the spectra data in .txt format (we use IONTOF’s Surface Lab software as an example to show how to export the unit mass .txt spectral data. If other manufacturers’ instruments are used, please follow their instructions to generate the data.)
   1. In the “spectra” window of Surface Lab 6 or 7 software → open the set of interesting spectra (either positive or negative)
   2. For every spectrum, follow the following steps to calibrate it:

Double-click a spectrum to select it → press F3 (or Spectrum -> Mass Calibration) to enter the mass calibration menu → default calibration points for negative spectra should be H-, C-, C2-, C3- and for positive spectra should be H+, H2+, CH­3+, C­3H3+ → if H- is highlighted red, it is oversaturated and needs to be excluded (which is done automatically), so click “Recalibrate” in the lower right-hand corner → exit the mass calibration menu by pressing ESC

* Note for positive spectra calibration: The 4th default ion, C3H­3+, usually causes large deviations (>100ppm) on the calibration; we should be using CH3O+ instead. To fix this, on each spectra’s mass calibration menu, you should right click this ion 🡪 select “Delete” 🡪 click on the “Find” search bar on the top-middle section of the screen 🡪 type CH3O+ and press enter 🡪 click on the red arrow that pops up on the center panel 🡪 an ID for the species you selected should pop up on the mass calibration menu 🡪 if it is incorrect, tune your result on the zoomed-in spectrum panel below the main panel by clicking the red arrow on that panel 🡪 now CH­3O+ should show up in the mass calibration panel’s text box 🡪 select “Add” 🡪 select “Recalibrate” 🡪 your calibration should have smaller deviations now!
  1. Activate the 1st spectrum→ “Peak List” → “Create Nominal Mass Binning” → in the new small window, copy the parameters below and click “Add to Peaklist”



* 1. Don’t close the pop-up window → right-click the text box that says “Nominal Mass Binning” in the submenu found in the lower left-hand corner→ select “Add To” -> “All Spectra” → recheck a few sample spectra to make sure all new peak lists have been created → now close the pop-up window
  2. Now right click the dialog box next that says “Nominal Mass Binning” again → select “Replace All Existing Peak Lists Of -> All Spectra to delete the extraneous “List 1” for all other spectra → also make sure to delete “List 1” for the current spectra if it is still there → this list can cause problems later, so MAKE SURE to delete it now!
  3. Select all spectra → click the “Peak Statistics On Selected Spectra” button at the left side of the peak list sub-window → a new window should pop up after 1-2 minutes, showing the results → select “Area normalized by Total Ion Intensity” → save the .txt file

1. **Editing Group Column Names of the Spectra File** Open the .txt data file using Excel → remove any irrelevant columns and rows (i.e., we probably don’t need the top- or bottom-most two rows or the stats columns at the end (mean, median, etc.) → make sure the columns are in order: Do this by highlighting all cells and going to “Data” -> “Sort” -> add “Sort by” “Row 1”, then select “Options” -> “Sort left to right” -> click OK → (1) make sure the 1st column’s name is “Mass (u)” 🡪 edit column headers; currently, they might be something like <No Sample Name> (01-N1) \*, <No Sample Name> (01-N2) \*…..; change them to 01-N1, 01-N2, 01-N3, 01-N4, 01-N5, 01-N6, 02-N1, 02-N2, ….. using CTRL+H to find and replace any extraneous character expressions → save the new file as “sims\_data.txt” → paste the new file into the pca data folder (<PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/sims-data/OriginalData)
2. Now open the file again in Notepad (preferably Notepad++ for RegEx editing purposes) → if there are any blank lines, remove them using either Notepad++’s features (select “Edit” -> “Line Operations -> “Remove Empty Lines”) or a Python script → also remove any trailing whitespace using Notepad++ by selecting “Edit” -> “Blank Operations” -> “Trim Trailing Space” → once you are certain that no pesky extraneous whitespaces are remaining, save the file and exit (if you have ANY formatting differences from what the code expects, you could get an error when doing PCA analysis later; usually, \*\*\*Error! Cannot Recognize Data!\*\*\*. Here’s a snapshot from a sample .txt data file immediately after it was generated by Surface Lab 7:

Table

Description automatically generated with medium confidence

and after being cleaned up:

Text, table

Description automatically generated

**Section 2: Running the Software**

1. **Checking Paths and File Names**:

There are several paths that must be changed before you run the software in order to ensure it is using the correct data and file locations. There are two options here: Either run the file main.py or run the file main\_test.py, which includes a user interface and buttons that allow you to adjust parameters and update your report without having to rerun the code. Do note that main\_test is being actively developed, so features and stability may be subject to change. Below, you can find instructions for running the more bare-bones command-line interface using main.py; main\_test will prompt you for these values every time it is run, so you DON’T need to worry about the following instructions if using this option.

* 1. PCA Directory Find main.py and the folder in which it exists—this should be the path of the location in which you downloaded the PCA analysis software followed by “pca-analysis/SIMS\_PCA/SIMS\_PCA/src” (for instance, your path will probably look something like /home/username/pca-analysis/SIMS\_PCA/SIMS\_PCA/src) 🡪 open main.py with VSCode, your chosen IDE, or Notepad 🡪 on line 16 of main.py, change the value of pcaDir to ”<PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA”
  2. Positive / Negative Ions On line 22 of main.py, you will see a variable called positive\_or\_negative\_ion that will have a value of either “positive” or “negative” and will need to be changed depending on the nature of your data. Here, type in the exact text, entirely in lowercase, of the option that corresponds to the sign of the ions in your raw data file.
  3. Raw Data File Name On line 25 of main.py, the variable f\_rawsims\_data will contain a string that says something like “sims-data/OriginalData/Hifh P Pasture\_Chris\_Positive.TXT”; change the final part of this path to the name of your raw sims data file, including the extension.
  4. Report Name On line 34 of main.py, the variable f\_report will contain a string that says something like “output\_sample/report.docx”; you can change report.docx to your preferred name for the report. Make sure to include the file extension.
  5. Group Numbers The file located at ”<PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/sims-data/OriginalData/\_groupnumbers.txt” specifies which group numbers you would like to analyze out of the groups found in your raw data file. You will need to make sure the numbers contained in this file are correct. (See Appendix 2 for detailed information on formatting.)
  6. Metadata File The file located at ”<PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/sims-data/OriginalData/\_metadata.txt” contains useful information about the collection of your data that will be included in the report, including the operator, sample group information, and more. You will need to make sure that the data here is formatted exactly as specified. (See Appendix 3 for detailed information on formatting.)

1. Open your Terminal as administrator 🡪 Input “cd <PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/src” to change your directory to where the software is stored, then press Enter 🡪 Input the command python main.py (or python main\_test.py), then press Enter to run the code. It will ask you whether you would like to update the database and prompt you to enter either “y” for yes or “n” for no. The first time you run, you should enter “n” since you need to generate the report before making any updates. Then press enter, and the code should run to completion.
2. Go to the folder (<PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/sims-data/output\_sample to check PCA results
   1. 10 combinations of 2-d PCA scores plots of PC1-PC5 with confidence circles
   2. 10 combinations of 2-d PCA scores plots of PC1-PC5 without confidence circles
   3. PC1-PC5 single PC scores plots
   4. PC1-PC5 scores tables for all spectra
   5. PC1 to PC5 loadings plots
   6. PC1-PC5 top 20 loadings tables

**Note 1:** In the 10 combinations of 2-d PCA scores plots of PC1-PC5 with confidence circles**,** 10 combinations of 2-d PCA scores plots of PC1-PC5 without confidence circles, 1-d PC1 to PC5 scores plots, andPC1 to PC5 loadings plots**,** to change plot characteristics such as the **font type** and **size** along with the plot **label** and **marker** attributes**:** 🡪 go to the folder <PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/src/pca\_sims and open plotting.py🡪 find the following lines: line 18-28 🡪 change the parameters below 🡪 save it and run again:

'font.family':'serif',

    'font.serif':'Times New Roman',

    'axes.labelsize': '28',

    'xtick.labelsize':'28',

    'ytick.labelsize':'28',

    'lines.linewidth':3,

    'lines.markersize':10,

    'legend.fontsize': '25',

**Note 2:** The circles on the graphs are specified by a **Confidence Interval** (or “confidence limits”) of 90%. If other values are needed, open plotting.py 🡪 find “nstd = 1.645” on line 113 🡪 change “1.645” to the desired value based on the below table 🡪 save plotting.py and rerun the main python script main.py:

|  |  |
| --- | --- |
| ***Confidence Interval*** | ***Z*** |
| 80% | 1.282 |
| 85% | 1.440 |
| 90% | 1.645 |
| 95% | 1.960 |
| 99% | 2.576 |
| 99.5% | 2.807 |
| 99.9% | 3.291 |

**Note 3:** If more or fewer **PCs** are required 🡪 open main.py 🡪 find line 37 and change max\_pcacomp = 5 to max\_pcacomp=n, where n is the desired number of components. 🡪 save main.py and run it again.

**Note 4:** In the 10 combinations of 2-d PCA scores plots of PC1-PC5 with confidence circles, 10 combinations of 2-d PCA score plots of PC1-PC5 without confidence circles, and 1-dPC1 to PC5 score plots, to change **symbol sizes** or **colors** 🡪 open plotting.py 🡪 find lines 115-122 🡪 replace the current strings with the desired ones according to the matplotlib documentation 🡪 save plotting.py and run main.py again.

markern = [ ',','o','H','^','v','x','1','D','X','2','3','4','8','s','p','.',

                    'h','+','d','|','\_',0,1,2,3,4,5,6,7,8,9,10,11]

        colorn = ['purple','green','blue','brown','red',

                    'teal','orange','magenta','pink',

                    'gray','violet','turquoise','yellow',

                    'lavender','tan','cyan','aqua','yellowgreen','chocolate',

                    'coral','fuchsia','goldenrod','indigo',

                    'grey','darkorange','rosybrown','palegreen','deepskyblue']

**Note 5:** In PC1-PC5 loadings tables, to change **the** **numbers of +/- loadings**: open pca\_sims.py and find line 282 in addition to opening plotting.py and finding line 242 🡪 change fetchn\_more=20 in both places to the desired number of loadings 🡪 save both files and run main.py again.

**Section 3: Peak Assignment Table in Report**

Once the PCA code is successfully run, a table of peak assignments can be found in the report. The peak assignments are taken from the ‘negative\_doc\_mass\_record.csv’ and ‘positive\_doc\_mass\_record.csv’ documents found in <PATH\_TO\_PCA>/SIMS\_PCA/SIMS\_PCA/sims-data. These are the default peak assignments, but the user can update these assignments from the report. To do so, complete the following steps:

1. Open the report file once it has been generated.
2. Find the columns labeled “Updated Peak Assignment (from Document Mass)” and “Updated Document Mass” in the report.
3. Enter the new peak assignment; for instance, this might look something like Cl- or C4H10NO+. Make sure to include the ion’s chemical formula and charge in each case. Note that you don’t have to use subscripts or superscripts, which the program will add for you in the update phase. You can also add a list of updated peak assignments, which MUST be separated by commas or new lines (achieved by pressing Enter).
4. Enter the new document masses as real numbers, usually to around 4 decimal places. You MUST enter the same number of these as you did peak assignments, or the code will give you an error.
5. (Optional) Find the column labeled “Measured Mass” and enter a real number found from a precise measurement of the ion’s mass-to-charge ratio made by your instrument. This can be used for comparison to the document mass to make sure your calibration checks out. The column “Peak Assignment (Qualified)” will highlight the peak assignment corresponding to your measured mass value in green if the document mass is less than 100ppm from the measured mass, yellow if it is between 100 and 200ppm away, and red if it is more than 200ppm away.
6. Save and close the report.
7. Find main.py and run it again. This time, when prompted on whether to update the database from user changes to the report, type “y” and press Enter. Once this is completed, the program will stop, and you should see that the values stored in measured\_masses.csv and positive\_doc\_mass\_record.csv / negative\_doc\_mass\_record.csv have been updated.
8. Now run main.py again, this time typing “n” and pressing Enter. Once the program has finished executing, open your report again. The Document Mass, Initial Peak Assignment, and Initial Probabilities columns should have been updated to reflect your changes!

**Note:** If using main\_test.py, you don’t have to rerun the code each time you would like to update the database or generate the report; just press the “Update document values from report” and “Generate PCA report from current database” buttons instead!

For example, in positive ion mode, the signals of C2H3+, Al+, and CNH+ all correspond to a unit mass of 27. The first few columns of the table should look something like the following before and after running the code:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Default Report Table** | | | → | **User-Updated Table** | | |
| Unit Mass | Document Mass | Initial Peak Assignment | Unit Mass | Document Mass | Initial Peak Assignment |
| 27 | 27.02 | C2H3+ | 27 | 27.02  26.98  27.01 | C2H3+  Al+  CNH+ |

If the mass record documents are updated using main.py, the user must rerun the python script to have the updated peak assignments appear in the exported report document.

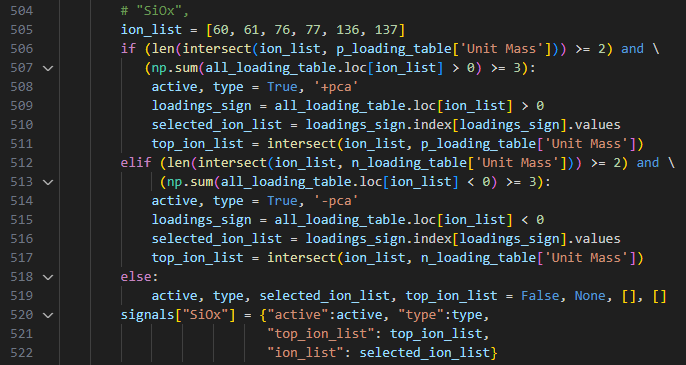
**Section 4: Adding Mass Combination Rules for Similar Chemical Species**

The following section provides guidance on how the user can incorporate rules to assign mass values to similar chemical species. For example, for SiOx-based signals, there can be a variety of mass values that are correspondent. Referring to the ‘negative\_doc\_mass\_record.csv’, values related to SiOx are tabulated.

|  |  |  |
| --- | --- | --- |
| Unit Mass | Assignment | Document Mass |
| 60 | SiO2- | 59.9673 |
| 61 | SiO2H- | 60.9751 |
| 76 | SiO3- | 75.9622 |
| 77 | SiO3H- | 76.97 |
| 136 | Si2O5- | 135.9289 |
| 137 | Si2O5H- | 136.9368 |

Once the mass documents are updated for their respective signals, the user can also update pca\_sims.py to account for chemically similar groups of ions by specifying the relevant unit masses.

* The python file to update is found here: <PATH\_TO\_PCA>/SIMS\_PCA/SIMS\_PCA/src/pca\_sims/pca\_sims.py
* The following snapshot contains the relevant block of code in the pca\_sims.py file (lines 504-522)
* The user should add unit masses for signals corresponding to the SiOx group on line 505:



**Appendix 1: Formatting the Raw Data File**

The following provides detailed guidance to the user on how to properly format the raw data .txt file. Careful consideration of the formatting is necessary for the code to execute.

**Note 1:** The first column name is ‘Mass (u)’ and the values in the column should be unit masses; i.e., whole integers.

**Note 2:** The remaining column names should be adjusted as follows:

* The general column name format should take the form ‘XXX-XX’ or ‘XXX\_XX’ (i.e., use either a hyphen or underscore separator between two groups of three and two characters, respectively)
* The first three digits are the sample group number
  + For the first sample of interest, XXX = 479
  + There can be up to 999 samples simultaneously analyzed
* The last two characters specify whether 1) the spectra are positive or negative ions and 2) the measurement location number, respectively
  + For example, if there are six locations analyzed for the first sample, and the data is from negative ion mode for the first location, then XX = N1
* Additional example: Column name for sample 479 at the fourth location analyzed in negative mode: 479-N4

The following screenshot shows an example of the raw data file to illustrate proper formatting:

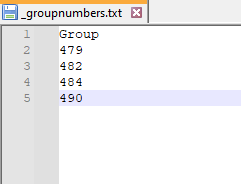
Text, table

Description automatically generated

**Note 3:** Save the formatted data as a text file in pca/SIMS\_PCA/SIMS\_PCA/sims-data/OriginalData

**Appendix 2: Formatting the Group Numbers File**

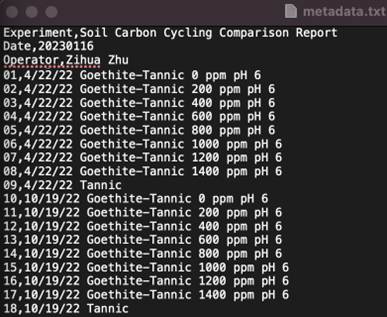
The file \_groupnumbers.txt, located in <PATH\_TO\_PCA>/pca-analysis/SIMS\_PCA/SIMS\_PCA/sims-data/OriginalData, must also be formatted properly before the software is run. The file should contain the word “Group” followed by a list of three-digit numbers, one on each line, as in the following example:

****

These represent the desired group numbers (from the top row in the raw data file) to be analyzed. If you would like to include all data in the PCA analysis, then make sure to write all sample numbers to this file; in this case, you would need to include 479, 480, 482, 483, 484, 489, 490, and 491—eight numbers in total. However, since only 4 numbers are given, only data from columns whose headers contain one of the sample numbers above will be included in the PCA analysis and subsequent component tables found in the report. Make sure that you include at least one three-digit number, and remember to double-check that each is actually one of your sample numbers. If you do happen to accidentally include a sample number that doesn’t exist, it will simply be ignored by the software.

**Appendix 3: Formatting the Metadata File**

To properly correspond the sample name to the number inputted into the txt file from Section 1, the metadata.txt file (found in <PATH\_TO\_PCA>/SIMS\_PCA/SIMS\_PCA/sims-data/OriginalData) should be appropriately formatted. The following image shows an example; a key consideration is to not have any spaces after the commas in the txt file:

****

The leftmost column should contain the sample group numbers, and the rightmost column should contain the sample descriptions. These will be used to label the samples on the PCA plots found in the report. If they are incorrectly numbered, your sample descriptions won’t match up with the samples in the actual raw data file.