Appendix B

Download the Jupyter Notebook file called "LinearProgramming_FromCSV.ipynb" here: https://github.com/charraden/Mineralogy-by-Linear-Programming/tree/main

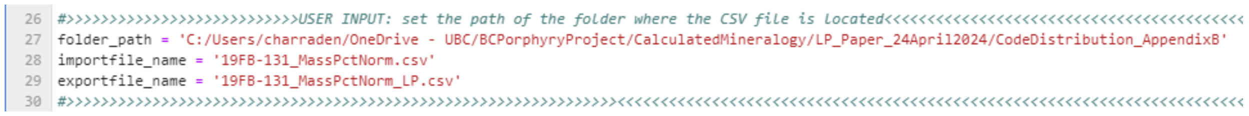
Python package prerequisites (user to install): pandas, numpy, pandarallel, matplotlib, seaborn, docplex, squarify

This Jupyter notebook contains Python code to run the linear programming method to estimate mineralogy from µXRF elemental data. The code requires the elemental data comes in as exported from the code in Appendix A, with the element column headers containing the element only. The code first reads in the compiled .csv table, then runs the linear programming model on each row (pixel) in the image, and appends the calculated proportion of each mineral active in the library to each of the pixels (rows) in the original elemental table. The algorithm also appends two additional columns: (1) the sum of all estimated minerals per row subtracted form 100%, indicating the unassigned mineralogy (termed "residuals"), and (2) the mineral reporting the highest proportion in each row (pixel). Finally, the algorithm outputs a number of visualizations and plots to show, at a glance, what the linear programming model produced. The visual outputs allow the user to identify the success of the linear programming model and mineral library as well as determine minerals that might be missing from the mineral library that could be added to improve the mineralogical interpretation. These include: (1) a box and whisker plot showing the elemental statistics for any rows (pixels) reporting a majority (i.e. greater than 50%) residuals/unassigned mineralogy, (2) a histogram of calculated modal mineralogy annotated with the percentage value for the entire sample (including residuals), (3) a Treemap plot showing a visual area representation of the modal mineralogy, and (4) a map showing each pixel's highest calculated proportion mineral or residuals. Looking at the residual distribution in space in addition to the chemical constituents of those high residuals was useful to determine which minerals might be missing, have incorrect chemical compositions in the library, or edge effects. Analyzing the chemistry and spatial distribution of the residuals (unassigned mineralogy), additional minerals were then added on a sample-by-sample basis and re-processed. For example, if a sample showed high residual pixels were elevated in P, Ca, and S, minerals calcite and apatite (both likely minerals in porphyry deposits) were added to the library. The code has been optimized to run in parallel, meaning that it will utilize a number of the processors available on the computer executing the code. This allows for much more rapid processing, especially for high resolution or pixel count µXRF maps. The tabular data can be viewed either in external software (e.g. QGIS, or ioGAS) or the user can add additional Python code to plot the tabular data as an image within the Python platform.

The instructions below are specific to having the input data as quantified (mass percent normalized) elements that have been compiled using the method in Appendix A. To use this Jupyter Notebook:

1) Save or copy the .csv file containing mass percent normalized elemental data into the same folder as the Jupyter Notebook.
2) Add the incoming file path (ensuring that any "\" are changed to "/" – this is a Python requirement) in line 27. This will also be the folder that the output file is saved into.

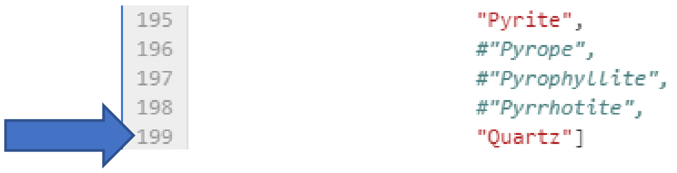
3) Add the input file name in line 28 and the output file name in line 29 (making sure to put names in quotes and adding the ".csv" file extension).



```
26 #>>>>>>>>>>>>>>>>>>>>>>>>>>>>USER INPUT: set the path of the folder where the CSV file is located<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
27 folder_path = 'C:/Users/charraden/OneDrive - UBC/BCPorphyryProject/CalculatedMineralogy/LP_Paper_24April2024/CodeDistribution_AppendixB'
28 importfile_name = '19FB-131_MassPctNorm.csv'
29 exportfile_name = '19FB-131_MassPctNorm_LP.csv'
30 #>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

4) In lines 42-127 represent the mineral library used in the linear programming calculations. Check that minerals have the correct formulas for your application (especially solid solution minerals) and add any minerals you want included. New minerals can be added to the mineral_elements list (the order doesn't matter), but must also be added to the active_minerals list to be considered. Note that some elements like C are not included for minerals (even for minerals that contain C, like calcite) – this is because the µXRf does not measure C, but this can be modified by the user for specific applications.

5) This is the list of active minerals to be included in the linear programming library. To select the minerals you want included, comment out (add a "#" in front of) any minerals you want <u>excluded</u> from the mineral library and make sure the minerals you want <u>included</u> are not commented out (the text should be red). Also make sure there is a "]" bracket (and no comma) after the last active mineral in the list. For example, if quartz is the last mineral you want active, then the last line in the active minerals list would be: "Quartz"]

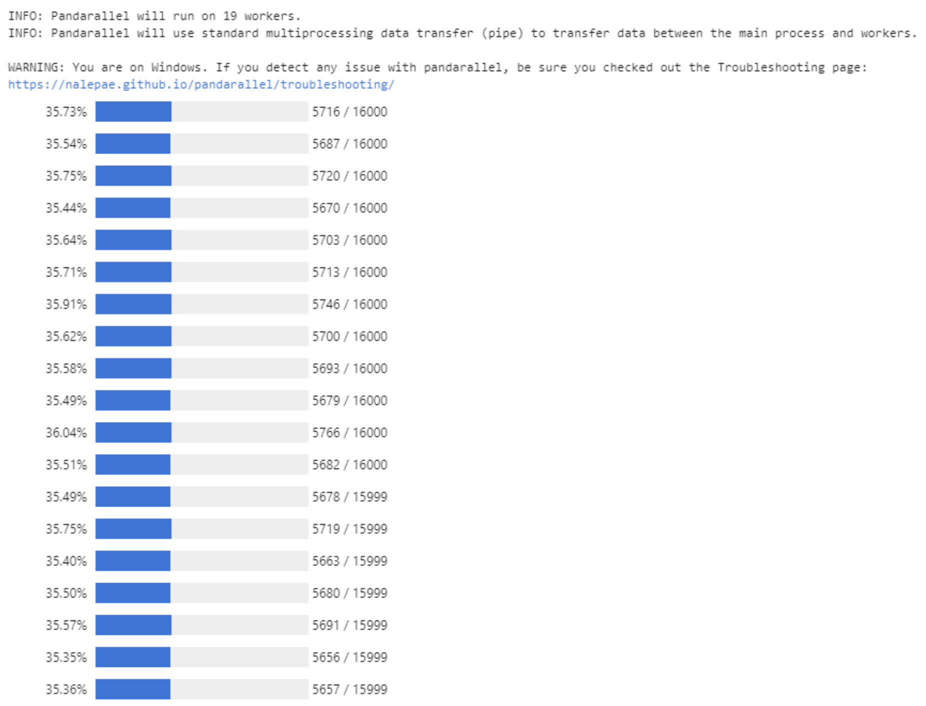


```
                        "Pyrite",
                        #"Pyrope",
                        #"Pyrophyllite",
                        #"Pyrrhotite",
                        "Quartz"]
```

NOTE: the code WILL NOT warn if an element is missing for a mineral - it will continue calculating with the other elements available, so, for example, if you have zircon active, but Zr is not included in the incoming elemental data, the algorithm will only use Si and, in this case, it could map zircon where quartz should be. It is important to think about the mineral selections and available chemistry as well as the mineralogical and geological context of the analysis.

6) Click inside the code box once, then hold down the SHIFT key, and hit ENTER on your keyboard to run the code.

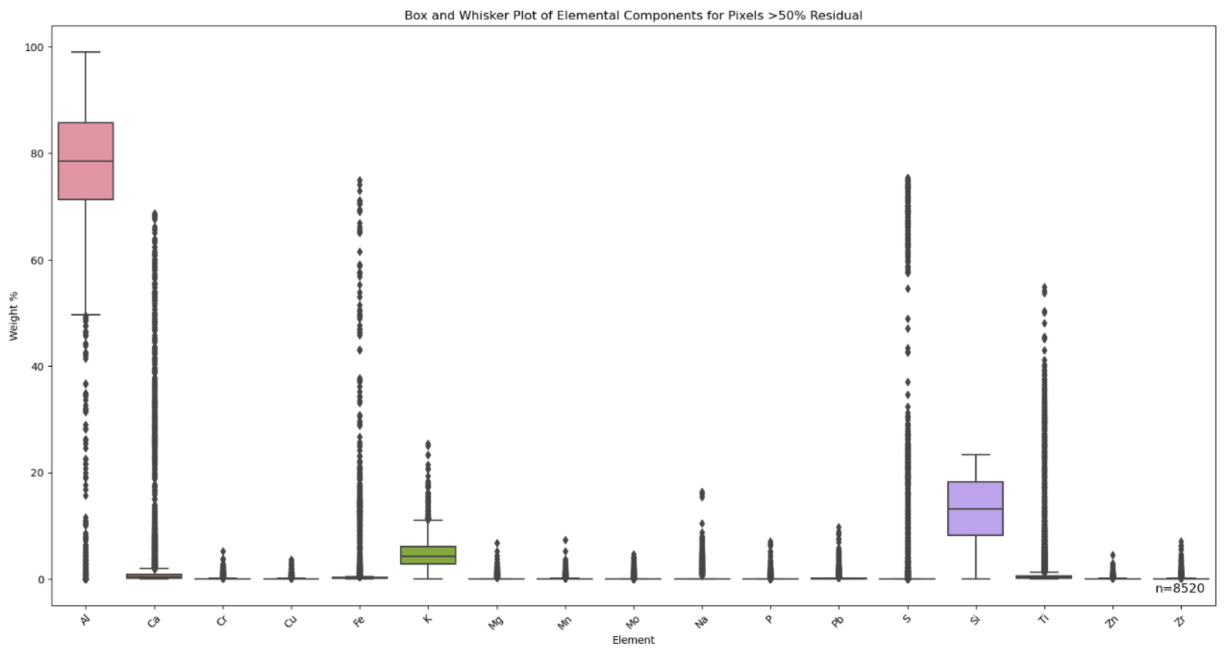
7) The parallel processing status will appear at the bottom so you can track the status of the processing. NOTE: this code is set up to use MOST of the processing capacity of your computer - if you are doing other tasks, running this will slow your computer/processing down significantly.



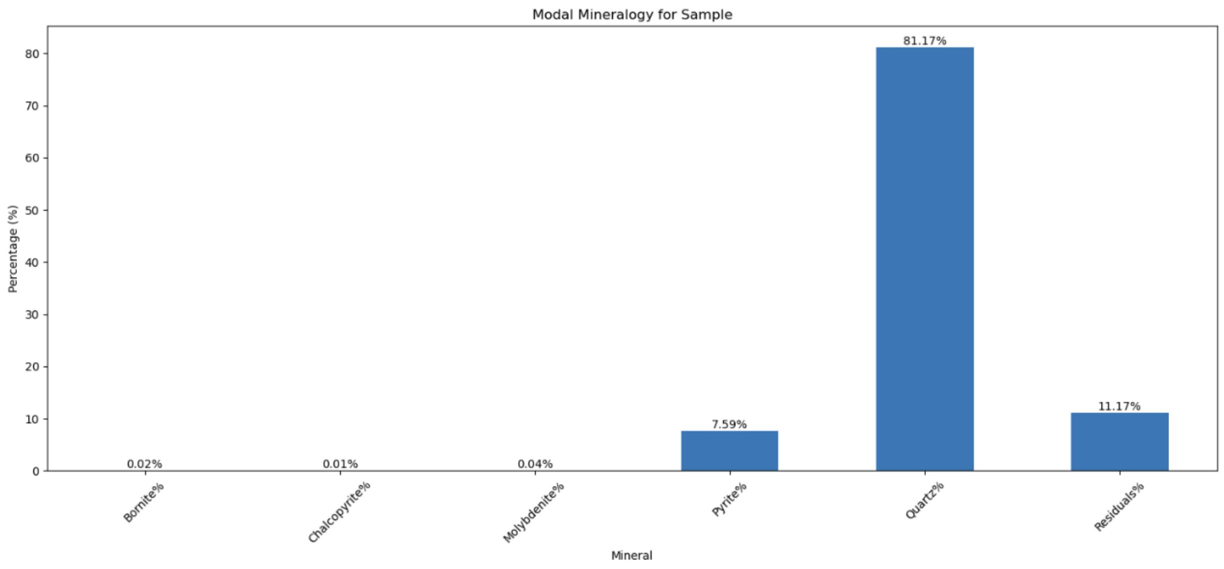
```
INFO: Pandarallel will run on 19 workers.
INFO: Pandarallel will use standard multiprocessing data transfer (pipe) to transfer data between the main process and workers.

WARNING: You are on Windows. If you detect any issue with pandarallel, be sure you checked out the Troubleshooting page:
https://nalepae.github.io/pandarallel/troubleshooting/
35.73%  5716 / 16000
35.54%  5687 / 16000
35.75%  5720 / 16000
35.44%  5670 / 16000
35.64%  5703 / 16000
35.71%  5713 / 16000
35.91%  5746 / 16000
35.62%  5700 / 16000
35.58%  5693 / 16000
35.49%  5679 / 16000
36.04%  5766 / 16000
35.51%  5682 / 16000
35.49%  5678 / 15999
35.75%  5719 / 15999
35.40%  5663 / 15999
35.50%  5680 / 15999
35.57%  5691 / 15999
35.35%  5656 / 15999
35.36%  5657 / 15999
```

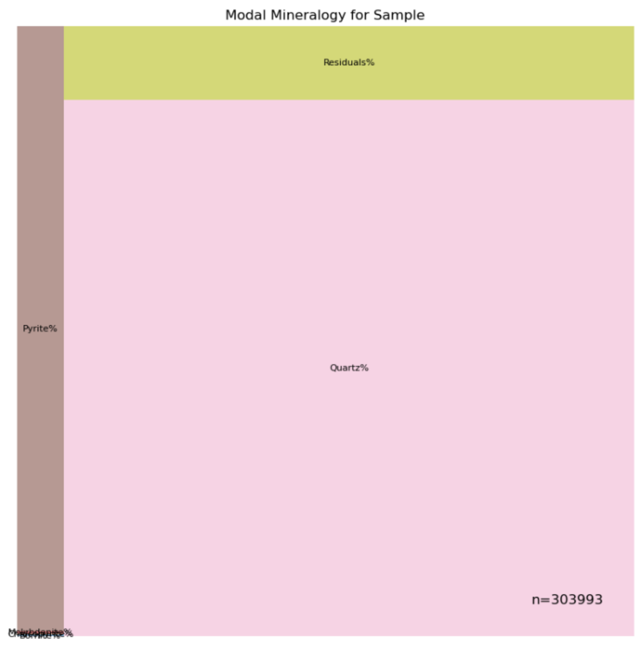
8) When the processing is complete, the status bars turn green, and the following plots appear:
   a) box and whisker plot representing the chemistry of rows/pixels >50% residuals



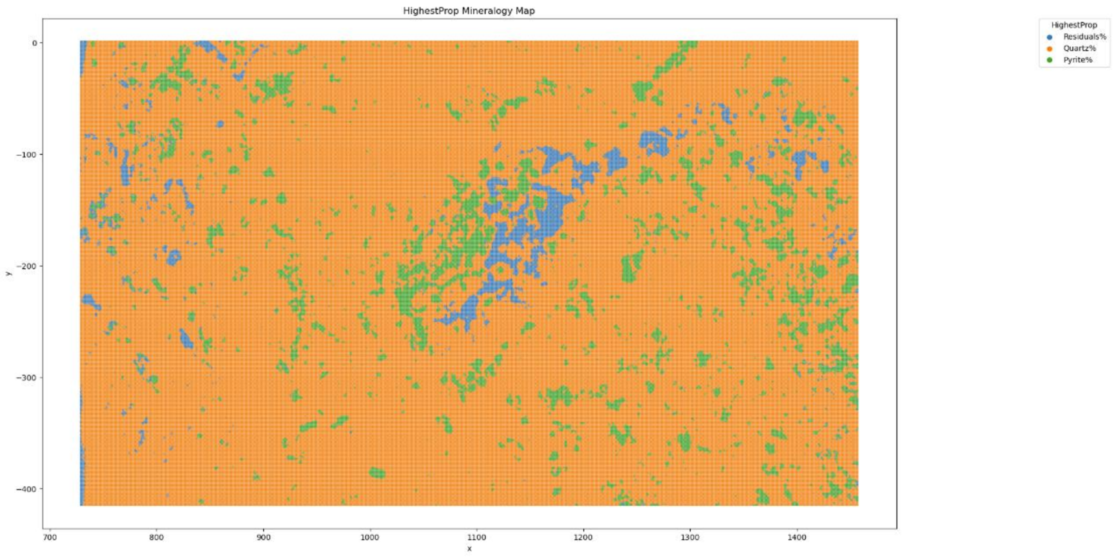
b) modal mineralogy histogram for the entire sample



c) Treemap showing modal mineralogy for the entire sample



d) map showing the highest proportion mineral for each row/pixel with spatial context



While the developed code contains a library of common minerals and generic compositions, the user is able to add minerals or modify the chemical compositions of those minerals easily to suit their needs. In solid solution scenarios (e.g. feldspars), the user can modify the chemical compositions to match end-member or mid-member chemistry, or can use externally validated chemical compositions measured from other methods like electron microprobe analysis (EMPA).

For the minerals reported in the paper, the elemental concentrations used in the linear programming mineral library are listed below:

Mineral	Al wt%	Ca wt%	Cu wt%	Fe wt%	K wt%	Mg wt%	Na wt%	P wt%	S wt%	Si wt%	Ti wt%	Zn wt%
Albite	10.77	0.76					8.30			31.50		
Andalusite	33.30									17.33		
Apatite		39.36						18.25				
Biotite	6.22			6.44	9.02	14.02				19.44		
Calcite		40.04										
Chalcopyrite			34.64	30.43					34.94			
Chlorite	10.27			32.81		2.26				10.73		
Corundum	52.93											
Hornblende	5.75	9.67		1.70		11.84				23.94		
Kaolinite	20.90									21.76		
Potassium Feldspar	9.69				14.05					30.27		
Magnetite				72.36								
Muscovite	20.30				9.81					21.13		
Plagioclase	9.96	7.40					4.25			31.12		
Pyrite				46.55					53.45			
Pyrophyllite	14.98									31.18		
Pyroxene	4.57	15.26		4.73		9.26				22.58	2.03	
Quartz										46.74		
Rutile											59.94	
Sphalerite				2.88					33.06			64.06
Titanite		19.25								14.20	18.16	