

Part II C3 Petrology - Practical 24: Crystallisation in the Icelandic Crust - ANSWERS

*Simon Matthews (sm905@cam.ac.uk), Lent 2023

In this practical you will apply some commonly used geobarometers to understanding how crystallisation happens in the Icelandic crust, and how that might be transferrable to understanding mid-ocean ridges in general.

1. Crustal Accretion at Mid-Ocean Ridges

The mode of crustal accretion at mid-ocean ridges can be constrained with geophysical and geochemical observations at active ridges and geological work at ophiolites. Two end-member models of crustal accretion are the gabbro-glacier models and the stacked-sills models.

In the **gabbro glacier**, melt is supplied directly from the mantle to a shallow magma chamber. Cooling of melt and crystallisation occurs in this magma chamber. The solid cumulate gabbros of the lower crust are then fed downwards by solid flow to generate the full thickness of oceanic crust. Importantly, in this model, all crystallisation takes place in the shallow melt lens and above.

In contrast, the **stacked-sills model**, favoured by workers such as Peter [Kelemen](#), involves crystallisation throughout the lower axial crust. In this model, mantle melts stall close to, or beneath, the Moho in small melt sills. Cooling and crystallisation occurs in these sills. Melt is then extracted from the sills, and may pond and crystallise in a series of sills in the lower crust. Therefore, in contrast to the gabbro-glacier model, the stacked-sills model involves crystallisation at a range of depths in the lower crust.



Figure 1: Schematic showing distribution of magma chambers, solid flow and melt flow in a) gabbro glacier and b) stacked sills crustal accretion models at typical fast-spreading ridges.

Arguments over oceanic crustal accretion mechanisms tend to focus on fast-spreading ridges. However, studying the Icelandic rift zones has made an important contribution to this subject. Geophysical studies have identified a shallow melt lens at 3-5 km depth under the Krafla caldera. This melt lens apparently sits above a predominantly solid lower crust that extends to ~20 km depth at Krafla. The arrangement of a shallow melt lens over a seismically fast lower crust inspired the development of a gabbro-glacier model of crustal accretion for Iceland ([Menke and Sparks, 1995](#)). In this practical, we will use the major element composition and petrography of Icelandic basalts to test this gabbro-glacier model.

2. Iceland: The Krafla and Theistareykir volcanic systems

The northern part of the Northern Rift Zone is a good location to examine crustal accretion mechanisms because it has been the focus of geophysical, geological and petrological observations since the 1975-84 Krafla fissure eruptions. Use Google Earth to take a look at the region and refer to geological maps to get a feeling for the area.



Figure 2: (a) Results of the FIRE seismic survey across the ridge axis at Krafla. Both the horizontal and vertical scales are in km; there is no vertical exaggeration. The zero position lies close to the centre of the Krafla caldera. P-wave isovelocity contours are marked with black lines and the velocity in km s^{-1} . The white fill shows the position of the melt lens. The black triangles mark the points where the seismic section crosses the caldera edge. (b) Tectonic map of Iceland showing positions of fissure swarms (light grey fill) and volcanic zones. WVZ, Western Volcanic Zone; EVZ, Eastern Volcanic Zone; NVZ, Northern Volcanic Zone. Glaciers are marked with white fill and the study area in the NVZ is outlined in black. (c) Tectonic and geological map of Krafla and Theistareykir. Extent of fissure swarms is filled with diagonal lines. The Krafla caldera is marked with ticks on the down-thrown side and the margins of the central volcanic areas are marked with dashed lines. The white circles show the positions of samples collected from Theistareykir and the black squares show the locations of the Krafla samples. The black solid line which cuts across the Krafla caldera is that of the FIRE seismic survey shown in (a). Lake Myvatn is filled white and labelled MY.

3. Composition of crystallising material

The major element trends of picrites and basalts from this region can be used to estimate the modal mineralogy of material that is crystallising within the Icelandic crust.

Q3.1: Take a look at some field photos and photos of thin sections of samples from the Krafla and Theistareykir segments, available in the lab and on Moodle. Note the range of macrocryst types present, their textures, and the modal mineralogy of cumulate xenoliths excavated by these mafic melts.

Your notes here...

3.1. Reading in and plotting up the Data

First we will import the packages we need for the first part of the practical:

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Next, we will load up and plot some whole-rock data from the Krafla and Theistareykir segments.

Q3.2: Write some code in the cell below to read in the file `maj_p24.csv` , which contains whole rock compositions, as a DataFrame:

```
In [5]: # Your code here...

wr = pd.read_csv('maj_p24.csv')
wr.head()
```

Out[5]:

	Unnamed: 0	Sample	Eruption	Volume.km3.	num_min_age.wrt2000AD.	num_max_age.wrt2000AD.	Reference
0	1	H16	Laxarhraun_yngra	2.50	2100.0	2100.0	Kokfelt2006
1	2	b1	Reykjaheidi	NaN	0.0	5000.0	Hemond1993
2	3	I9358	Storavitishraun	18.40	11300.0	11300.0	Slater2001
3	4	I9372	Arnahvammurhraun	0.01	8500.0	10300.0	Slater2001Stracke2003a
4	5	TH13elliott	Skessuhraun	0.01	8500.0	10300.0	Slater2001Elliott1991

Q3.3: Read in the file `min_p24.csv` , which contains mineral compositions, as a DataFrame:

```
In [6]: # Your code here...

mn = pd.read_csv('min_p24.csv')
mn
```

Out[6]:

	Mineral	SiO2	Al2O3	FeOt	MgO	CaO
0	Olivine	40.21	0.01	12.82	46.23	0.33
1	Clinopyroxene	51.84	3.75	5.91	17.05	19.56
2	Plagioclase	47.29	33.53	0.44	0.10	17.16

Q3.4: Modify the code below so that it will plot from the dataframes you have just created:

```
In [7]: fig, ax = plt.subplots()

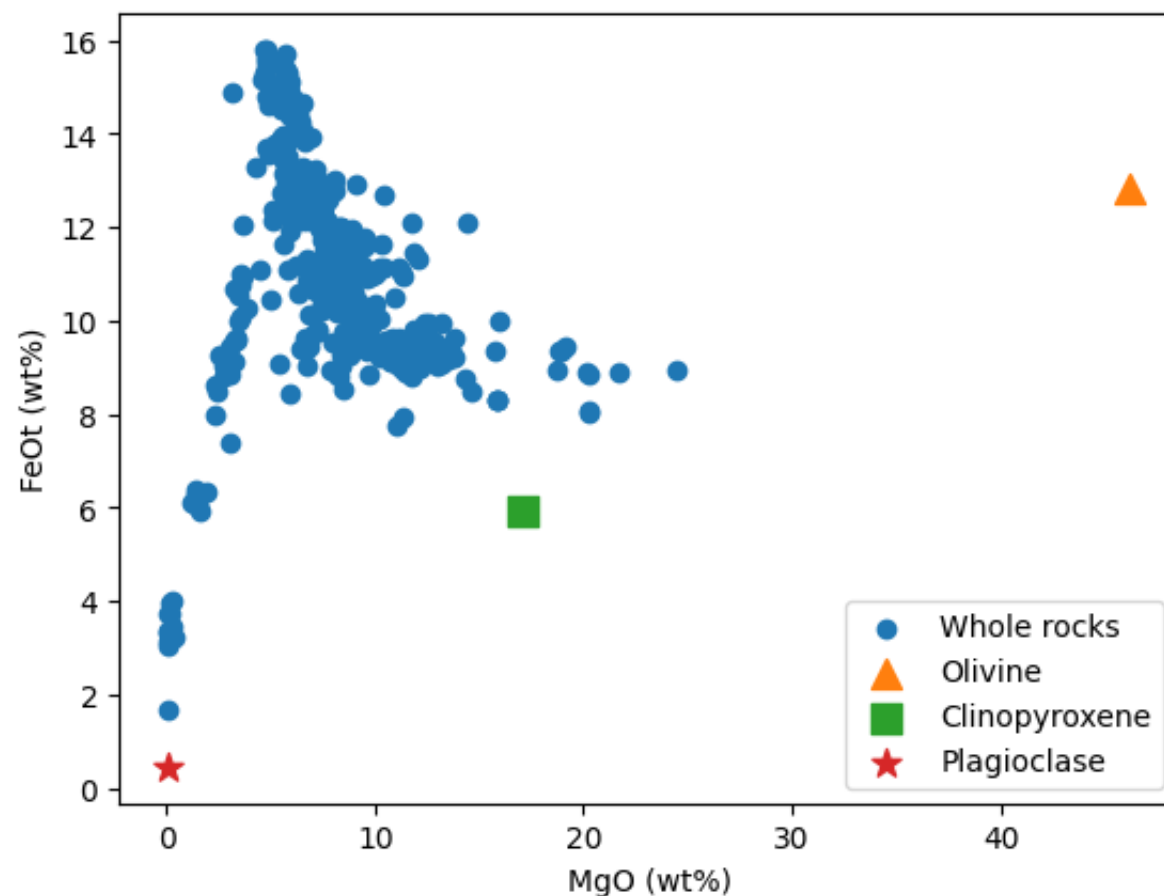
# Plot the whole rock data:
ax.scatter(wr.MgO, wr.FeOt, label='Whole rocks')

# Plot the minerals, each with a different symbol:
markers = ['^', 's', '*']
colors = ['C1', 'C2', 'C3']
for i, mineral in zip(range(3), mn.Mineral.unique()):
    ax.scatter(mn.MgO[mn.Mineral==mineral],
               mn.FeOt[mn.Mineral==mineral],
               marker=markers[i],
               c=colors[i],
               s=100,
               label=mineral)

ax.legend()

ax.set_xlabel('MgO (wt%)')
ax.set_ylabel('FeOt (wt%)')

plt.show()
```



FeOt means all iron, both ferric and ferrous, expressed as FeO. Usually about 15% of the Fe is Fe³⁺.

Q3.5: Now copy and modify this code to produce plots of MgO vs Al₂O₃ and MgO vs CaO:

```
In [9]: # Your code here...

fig, ax = plt.subplots()

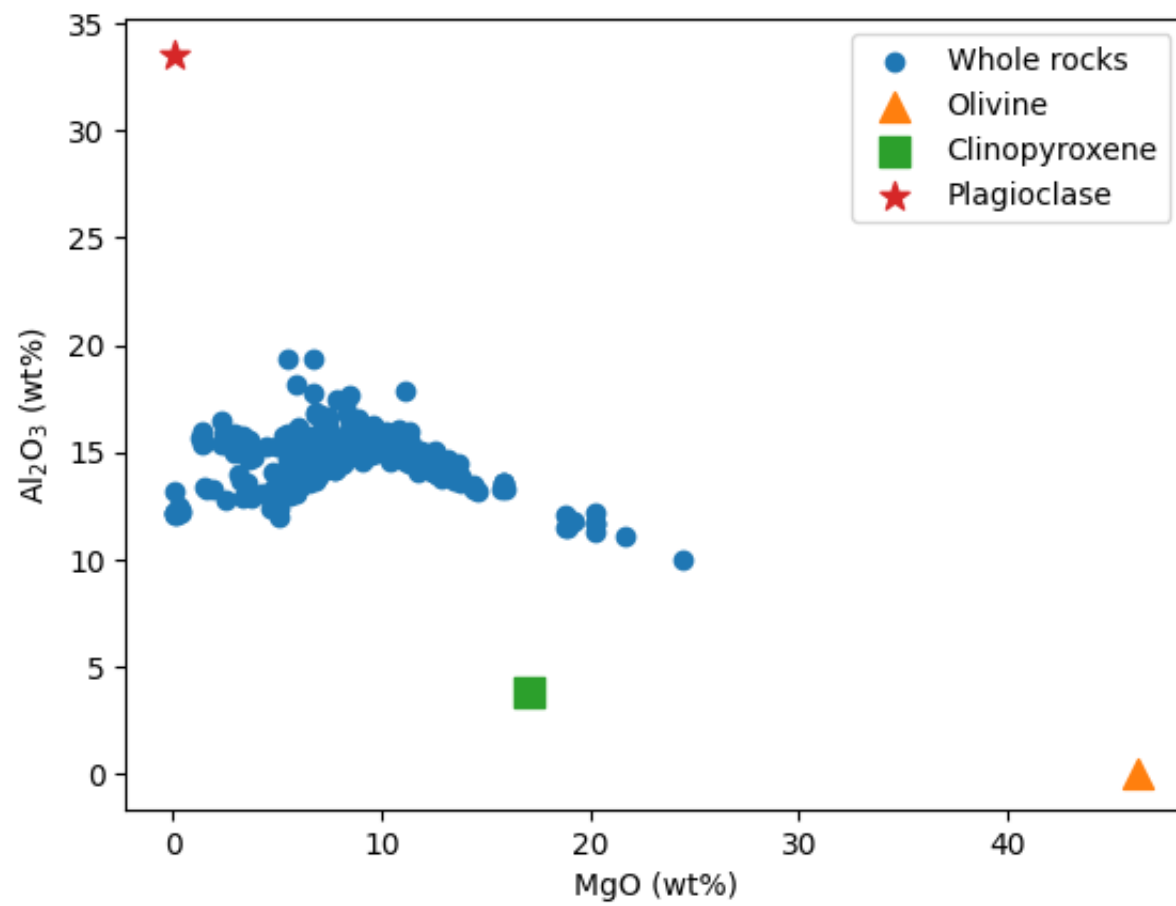
# Plot the whole rock data:
ax.scatter(wr.MgO, wr.Al2O3, label='Whole rocks')

# Plot the minerals, each with a different symbol:
markers = ['^', 's', '*']
colors = ['C1', 'C2', 'C3']
for i, mineral in zip(range(3), mn.Mineral.unique()):
    ax.scatter(mn.MgO[mn.Mineral==mineral],
               mn.Al2O3[mn.Mineral==mineral],
               marker=markers[i],
               c=colors[i],
               s=100,
               label=mineral)

ax.legend()

ax.set_xlabel('MgO (wt%)')
ax.set_ylabel('Al2O3 (wt%)')

plt.show()
```



In [12]: *# Your code here...*

```
fig, ax = plt.subplots()

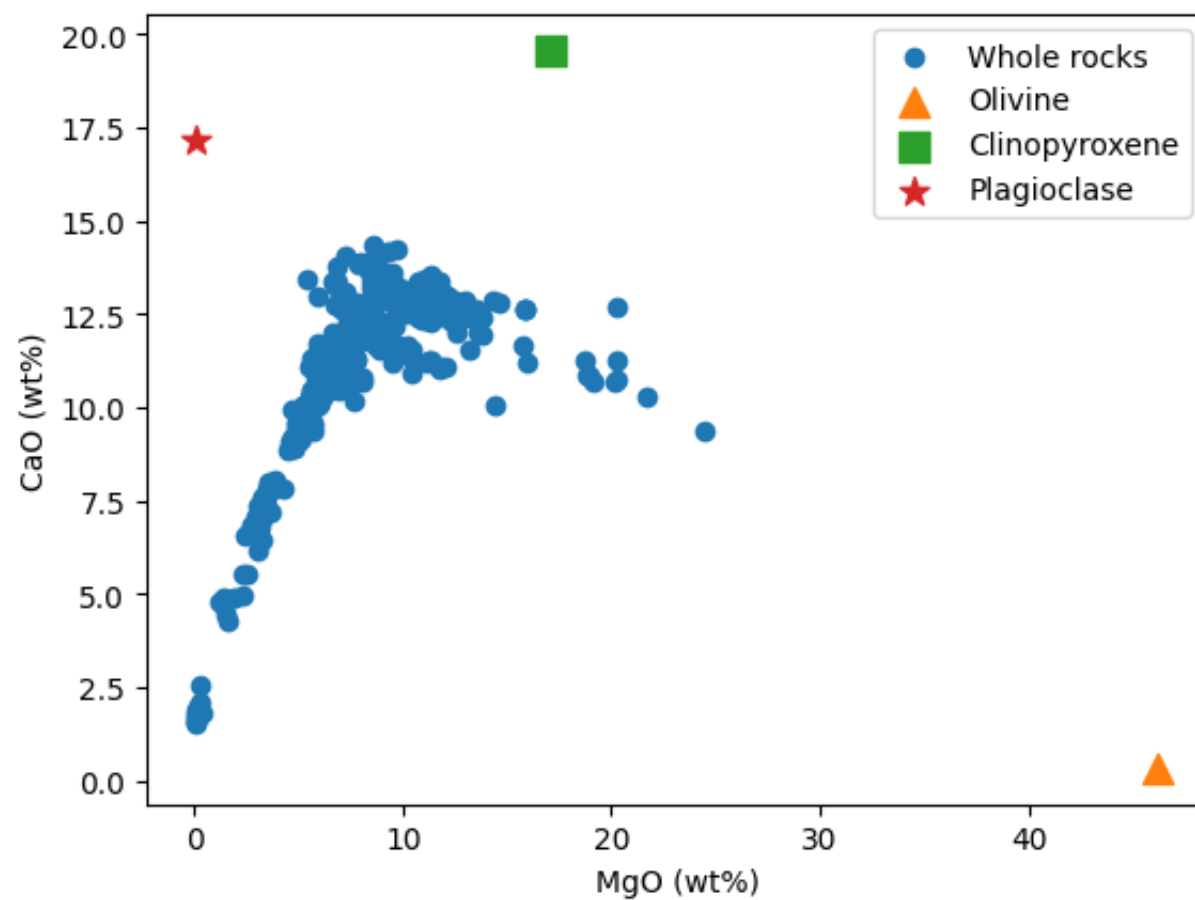
# Plot the whole rock data:
ax.scatter(wr.MgO, wr.CaO, label='Whole rocks')

# Plot the minerals, each with a different symbol:
markers = ['^', 's', '*']
colors = ['C1', 'C2', 'C3']
for i, mineral in zip(range(3), mn.Mineral.unique()):
    ax.scatter(mn.MgO[mn.Mineral==mineral],
               mn.CaO[mn.Mineral==mineral],
               marker=markers[i],
               c=colors[i],
               s=100,
               label=mineral)

ax.legend()

ax.set_xlabel('MgO (wt%)')
ax.set_ylabel('CaO (wt%)')

plt.show()
```




3.2 Interpreting the trends

It is possible to understand the kinks and trends in these plots using the concept of a *liquid line of descent*. In this way of thinking, a single liquid composition is supplied from the mantle and then evolves by cooling and fractional crystallisation in magma chambers. The pattern of whole-rock compositions can be broadly matched by such an approach for Krafla and Theistareykir.

Q3.6: The primary melt compositions for Iceland are not likely to contain more than about 15 wt% MgO. How is it then possible to have whole-rock lava samples that contain >20 wt% MgO?

Hint - look at the thin section images available in the lab and on Moodle.

 *Crystal accumulation of olivine*

You can also see a strong linear trend in samples with <5 wt% MgO. This trend is thought to result partly from fractional crystallisation of a cumulate material containing magnetite (which explains the drop in Fe with evolution) and also mixing of evolved basalts with rhyolites. You should also be able to identify two other dominant segments.

We will now quantify the proportions of crystals required to crystallise to generate these trends. One way of doing this (by hand) would be to plot the rock compositions on ternary graph paper and back project the trends onto tie-lines.

We are going to use a method similar to this, but using python rather than hand-plotting. We will draw a triangle defined by the three minerals on each of the major element plots above, then find a line that describes the liquid line of descent, and finally see where that intersects the triangle.

Here's some code to make these plots.

Q3.7: Add the array of whole rock compositions to these plots.

```
In [30]: line_Fe = {'m': 1,
                  'c': 1}

line_Al = {'m': 2,
           'c': 2}

line_Ca = {'m': 2,
           'c': 2}

fig, ax = plt.subplots(2, 2, figsize=[10,7])
ax = np.ravel(ax).tolist()

mn = mn # You can change the r.h.s. to your mineral dataframe

### PLOT WHOLE ROCKS HERE ###
ax[0].scatter(wr.MgO, wr.FeOt, label='Whole rocks')
ax[1].scatter(wr.MgO, wr.Al2O3, label='Whole rocks')
ax[2].scatter(wr.MgO, wr.CaO, label='Whole rocks')

markers = ['^', 's', '*']
colors = ['C1', 'C2', 'C3']
for i, mineral in zip(range(3), mn.Mineral.unique()):
    ax[0].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.FeOt[mn.Mineral==mineral],
                  marker=markers[i],
                  c=colors[i],
                  s=100,
                  label=mineral)

    ax[1].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.Al2O3[mn.Mineral==mineral],
                  marker=markers[i],
                  c=colors[i],
                  s=100,
                  label=mineral)

    ax[2].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.CaO[mn.Mineral==mineral],
                  marker=markers[i],
                  c=colors[i],
                  s=100,
                  label=mineral)

ax[0].plot([mn.MgO[mn.Mineral=='Olivine'],
```

```

        mn.MgO[mn.Mineral=='Clinopyroxene']],
        [mn.FeOt[mn.Mineral=='Olivine'],
         mn.FeOt[mn.Mineral=='Clinopyroxene']],
        c='k')
ax[0].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
            [mn.FeOt[mn.Mineral=='Olivine'],
             mn.FeOt[mn.Mineral=='Plagioclase']],
            c='k')
ax[0].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.FeOt[mn.Mineral=='Plagioclase'],
             mn.FeOt[mn.Mineral=='Clinopyroxene']],
            c='k')

ax[1].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.Al2O3[mn.Mineral=='Olivine'],
             mn.Al2O3[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[1].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
            [mn.Al2O3[mn.Mineral=='Olivine'],
             mn.Al2O3[mn.Mineral=='Plagioclase']],
            c='k')
ax[1].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.Al2O3[mn.Mineral=='Plagioclase'],
             mn.Al2O3[mn.Mineral=='Clinopyroxene']],
            c='k')

ax[2].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.CaO[mn.Mineral=='Olivine'],
             mn.CaO[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[2].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
            [mn.CaO[mn.Mineral=='Olivine'],
             mn.CaO[mn.Mineral=='Plagioclase']],
            c='k')
ax[2].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.CaO[mn.Mineral=='Plagioclase'],
             mn.CaO[mn.Mineral=='Clinopyroxene']],
            c='k')

ax[3].set_visible(False)

for i in range(3):
    ax[i].set_xlabel('MgO (wt%)')
    ax[i].set_xlim(ax[i].get_xlim())
    ax[i].set_ylim(ax[i].get_ylim())
ax[0].set_ylabel('FeOt (wt%)')
ax[1].set_ylabel('Al2O3 (wt%)')
ax[2].set_ylabel('CaO (wt%)')

ax[0].plot(ax[0].get_xlim(),
            line_Fe['m'] * np.array(ax[0].get_xlim()) + line_Fe['c'],
            c='red')

ax[1].plot(ax[1].get_xlim(),
            line_Al['m'] * np.array(ax[1].get_xlim()) + line_Al['c'],
            c='red')

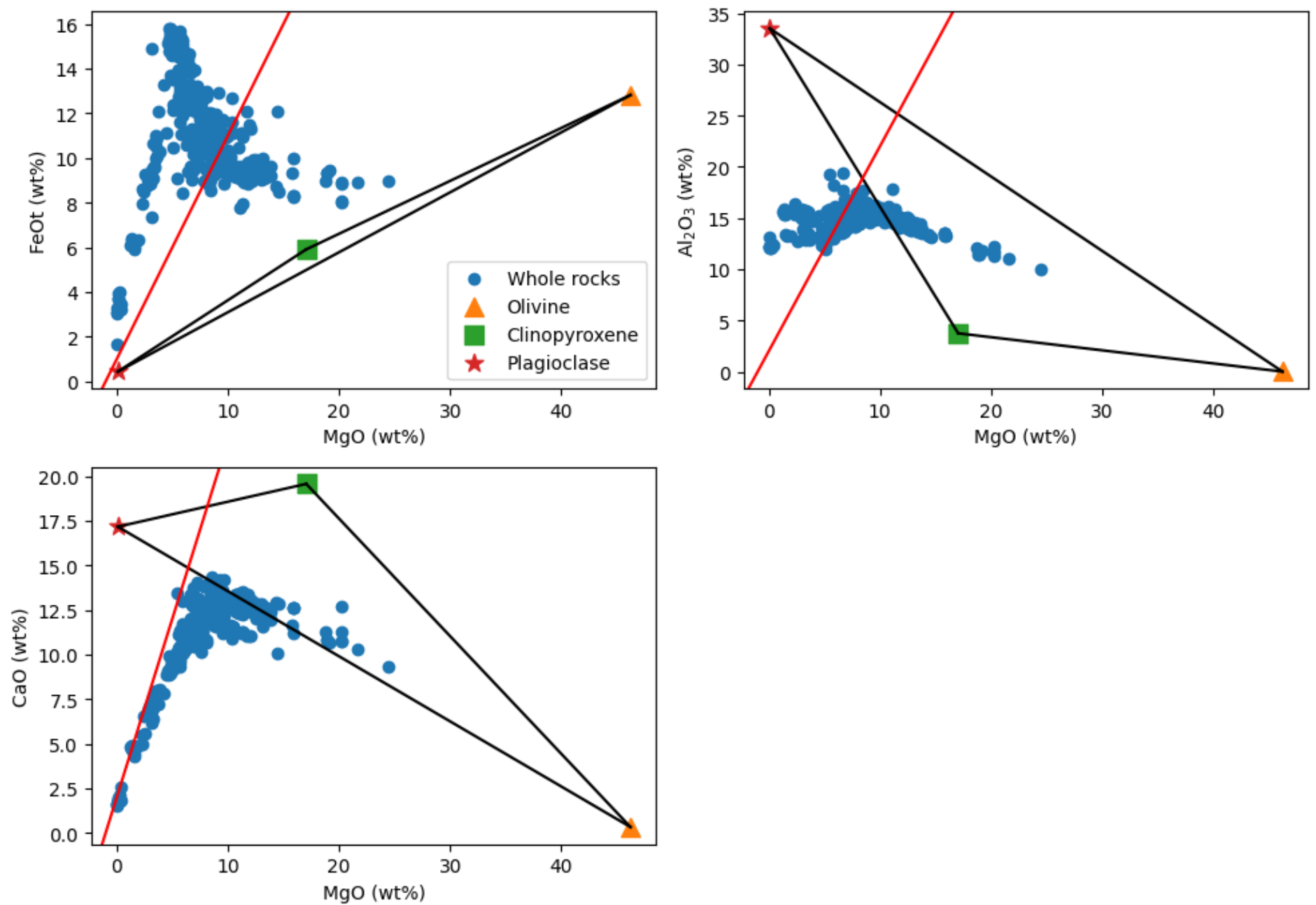
ax[2].plot(ax[2].get_xlim(),
            line_Ca['m'] * np.array(ax[2].get_xlim()) + line_Ca['c'],
            c='red')

ax[0].legend(loc='lower right')

fig.tight_layout()

plt.show()

```

Q3.8: Use these plots (and changing the `line_Fe`, `line_Al`, `line_Ca` parameters) to work out what cumulate proportions could give rise to the two high MgO segments of the liquid line of descent.

You might want to copy and paste the code for each segment. I also suggest decreasing the size of the symbols for the whole rocks.

```
In [73]: ### HIGH MG SEGMENT #####

line_Fe = {'m': -0.05,
           'c': 10}

line_Al = {'m': -0.35,
           'c': 19}

line_Ca = {'m': -0.27,
           'c': 16.0}

fig, ax = plt.subplots(2, 2, figsize=[10,7])
ax = np.ravel(ax).tolist()

mn = mn # You can change the r.h.s. to your mineral dataframe

### PLOT WHOLE ROCKS HERE ###
ax[0].scatter(wr.MgO, wr.FeOt, label='Whole rocks', s=10)
ax[1].scatter(wr.MgO, wr.Al2O3, label='Whole rocks', s=10)
ax[2].scatter(wr.MgO, wr.CaO, label='Whole rocks', s=10)

markers = ['^', 's', '*']
colors = ['C1', 'C2', 'C3']
for i, mineral in zip(range(3), mn.Mineral.unique()):
    ax[0].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.FeOt[mn.Mineral==mineral],
                  marker=markers[i],
                  c=colors[i],
                  s=100,
                  label=mineral)

    ax[1].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.Al2O3[mn.Mineral==mineral],
                  marker=markers[i],
```

```

        c=colors[i],
        s=100,
        label=mineral)

ax[2].scatter(mn.MgO[mn.Mineral==mineral],
              mn.CaO[mn.Mineral==mineral],
              marker=markers[i],
              c=colors[i],
              s=100,
              label=mineral)

props = np.linspace(0.1, 0.9, 8)
ax[0].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.FeOt[mn.Mineral=='Olivine'],
            mn.FeOt[mn.Mineral=='Clinopyroxene']],
           c='k')
ax[0].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.FeOt[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.FeOt[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[0].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
           [mn.FeOt[mn.Mineral=='Olivine'],
            mn.FeOt[mn.Mineral=='Plagioclase']],
           c='k')
ax[0].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0],
              props * mn.FeOt[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.FeOt[mn.Mineral=='Plagioclase'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[0].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.FeOt[mn.Mineral=='Plagioclase'],
            mn.FeOt[mn.Mineral=='Clinopyroxene']],
           c='k')
ax[0].scatter(props * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.FeOt[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.FeOt[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)

ax[1].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.Al2O3[mn.Mineral=='Olivine'],
            mn.Al2O3[mn.Mineral=='Clinopyroxene']],
           c='k')
ax[1].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.Al2O3[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.Al2O3[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[1].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
           [mn.Al2O3[mn.Mineral=='Olivine'],
            mn.Al2O3[mn.Mineral=='Plagioclase']],
           c='k')
ax[1].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0],
              props * mn.Al2O3[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.Al2O3[mn.Mineral=='Plagioclase'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[1].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.Al2O3[mn.Mineral=='Plagioclase'],
            mn.Al2O3[mn.Mineral=='Clinopyroxene']],
           c='k')
ax[1].scatter(props * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.Al2O3[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.Al2O3[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)

ax[2].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.CaO[mn.Mineral=='Olivine'],
            mn.CaO[mn.Mineral=='Clinopyroxene']],
           c='k')

```



```

ax[2].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.CaO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.CaO[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[2].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
           [mn.CaO[mn.Mineral=='Olivine'],
            mn.CaO[mn.Mineral=='Plagioclase']],
           c='k')
ax[2].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0],
              props * mn.CaO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.CaO[mn.Mineral=='Plagioclase'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[2].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.CaO[mn.Mineral=='Plagioclase'],
            mn.CaO[mn.Mineral=='Clinopyroxene']],
           c='k')
ax[2].scatter(props * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.CaO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.CaO[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)

ax[3].set_visible(False)

for i in range(3):
    ax[i].set_xlabel('MgO (wt%)')
    ax[i].set_xlim(ax[i].get_xlim())
    ax[i].set_ylim(ax[i].get_ylim())
ax[0].set_ylabel('FeOt (wt%)')
ax[1].set_ylabel('Al2O3 (wt%)')
ax[2].set_ylabel('CaO (wt%)')

ax[0].plot(ax[0].get_xlim(),
           line_Fe['m'] * np.array(ax[0].get_xlim()) + line_Fe['c'],
           c='red')

ax[1].plot(ax[1].get_xlim(),
           line_Al['m'] * np.array(ax[1].get_xlim()) + line_Al['c'],
           c='red')

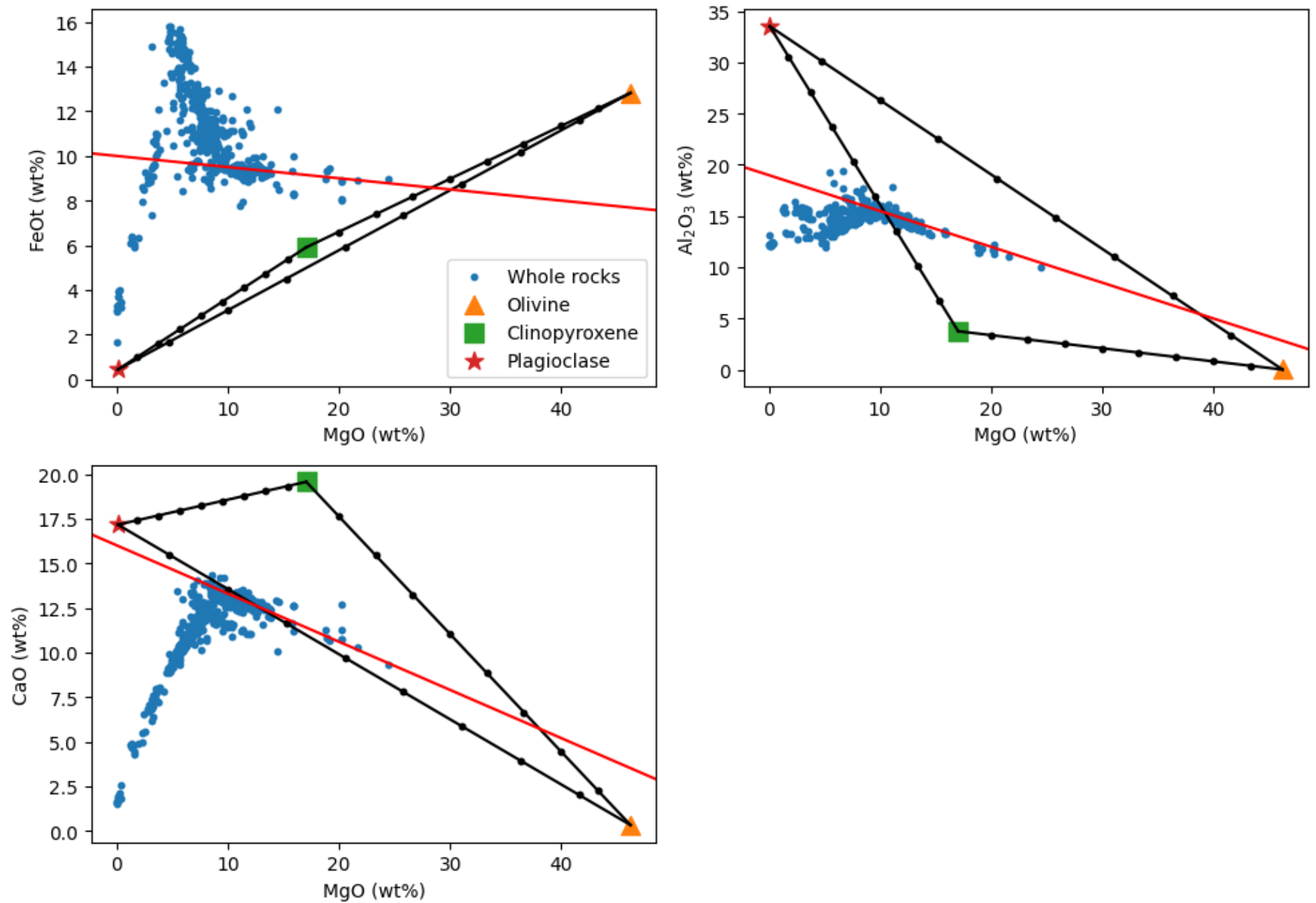
ax[2].plot(ax[2].get_xlim(),
           line_Ca['m'] * np.array(ax[2].get_xlim()) + line_Ca['c'],
           c='red')

ax[0].legend(loc='lower right')

fig.tight_layout()

plt.show()

```



```
In [78]: ### LOWER MG SEGMENT #####

line_Fe = {'m': -2,
           'c': 26}

line_Al = {'m': 0.7,
           'c': 10}

line_Ca = {'m': 0.2,
           'c': 10.5}

fig, ax = plt.subplots(2, 2, figsize=[10,7])
ax = np.ravel(ax).tolist()

mn = mn # You can change the r.h.s. to your mineral dataframe

### PLOT WHOLE ROCKS HERE ###
ax[0].scatter(wr.MgO, wr.FeOt, label='Whole rocks', s=10)
ax[1].scatter(wr.MgO, wr.Al2O3, label='Whole rocks', s=10)
ax[2].scatter(wr.MgO, wr.CaO, label='Whole rocks', s=10)

markers = ['^', 's', '*']
colors = ['C1', 'C2', 'C3']
for i, mineral in zip(range(3), mn.Mineral.unique()):
    ax[0].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.FeOt[mn.Mineral==mineral],
                  marker=markers[i],
                  c=colors[i],
                  s=100,
                  label=mineral)

    ax[1].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.Al2O3[mn.Mineral==mineral],
                  marker=markers[i],
                  c=colors[i],
                  s=100,
                  label=mineral)

    ax[2].scatter(mn.MgO[mn.Mineral==mineral],
                  mn.CaO[mn.Mineral==mineral],
```

```

        marker=markers[i],
        c=colors[i],
        s=100,
        label=mineral)

props = np.linspace(0.1, 0.9, 8)
ax[0].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.FeOt[mn.Mineral=='Olivine'],
            mn.FeOt[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[0].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.FeOt[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.FeOt[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[0].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
            [mn.FeOt[mn.Mineral=='Olivine'],
            mn.FeOt[mn.Mineral=='Plagioclase']],
            c='k')
ax[0].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0],
              props * mn.FeOt[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.FeOt[mn.Mineral=='Plagioclase'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[0].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.FeOt[mn.Mineral=='Plagioclase'],
            mn.FeOt[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[0].scatter(props * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.FeOt[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.FeOt[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)

ax[1].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.Al2O3[mn.Mineral=='Olivine'],
            mn.Al2O3[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[1].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.Al2O3[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.Al2O3[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[1].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Plagioclase']],
            [mn.Al2O3[mn.Mineral=='Olivine'],
            mn.Al2O3[mn.Mineral=='Plagioclase']],
            c='k')
ax[1].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0],
              props * mn.Al2O3[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.Al2O3[mn.Mineral=='Plagioclase'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[1].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.Al2O3[mn.Mineral=='Plagioclase'],
            mn.Al2O3[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[1].scatter(props * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.Al2O3[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.Al2O3[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)

ax[2].plot([mn.MgO[mn.Mineral=='Olivine'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
            [mn.CaO[mn.Mineral=='Olivine'],
            mn.CaO[mn.Mineral=='Clinopyroxene']],
            c='k')
ax[2].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.CaO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.CaO[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[2].plot([mn.MgO[mn.Mineral=='Olivine'],

```

```

        mn.MgO[mn.Mineral=='Plagioclase']],
        [mn.CaO[mn.Mineral=='Olivine'],
         mn.CaO[mn.Mineral=='Plagioclase']],
        c='k')
ax[2].scatter(props * mn.MgO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0],
              props * mn.CaO[mn.Mineral=='Olivine'].iloc[0] + (1 - props) * mn.CaO[mn.Mineral=='Plagioclase'].iloc[0],
              c='k',
              marker='o',
              s=10)
ax[2].plot([mn.MgO[mn.Mineral=='Plagioclase'],
            mn.MgO[mn.Mineral=='Clinopyroxene']],
           [mn.CaO[mn.Mineral=='Plagioclase'],
            mn.CaO[mn.Mineral=='Clinopyroxene']],
           c='k')
ax[2].scatter(props * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0],
              props * mn.CaO[mn.Mineral=='Plagioclase'].iloc[0] + (1 - props) * mn.CaO[mn.Mineral=='Clinopyroxene'].iloc[0],
              c='k',
              marker='o',
              s=10)

ax[3].set_visible(False)

for i in range(3):
    ax[i].set_xlabel('MgO (wt%)')
    ax[i].set_xlim(ax[i].get_xlim())
    ax[i].set_ylim(ax[i].get_ylim())
ax[0].set_ylabel('FeOt (wt%)')
ax[1].set_ylabel('Al2O3 (wt%)')
ax[2].set_ylabel('CaO (wt%)')

ax[0].plot(ax[0].get_xlim(),
           line_Fe['m'] * np.array(ax[0].get_xlim()) + line_Fe['c'],
           c='red')

ax[1].plot(ax[1].get_xlim(),
           line_Al['m'] * np.array(ax[1].get_xlim()) + line_Al['c'],
           c='red')

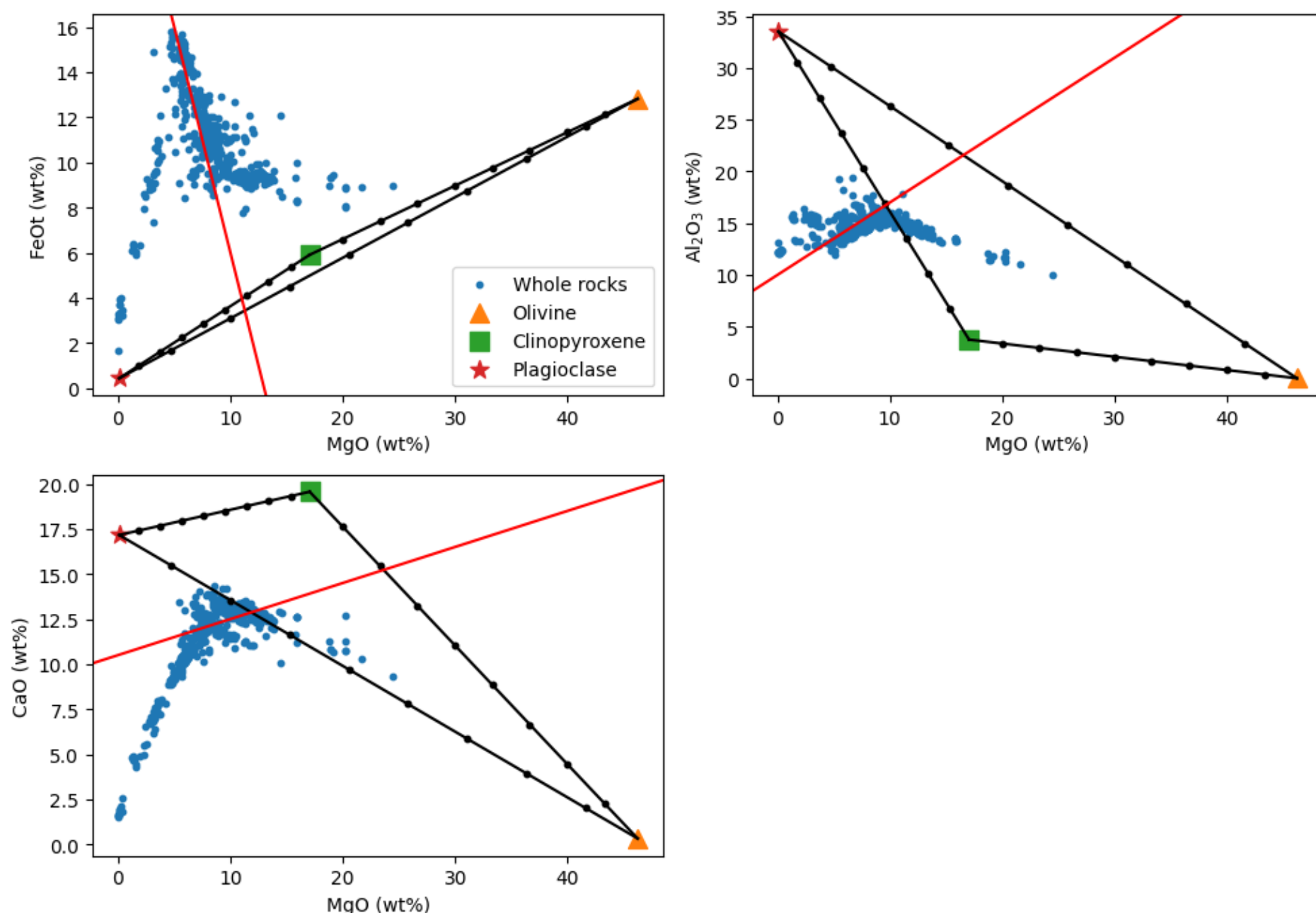
ax[2].plot(ax[2].get_xlim(),
           line_Ca['m'] * np.array(ax[2].get_xlim()) + line_Ca['c'],
           c='red')

ax[0].legend(loc='lower right')

fig.tight_layout()

plt.show()

```



● High MgO trend: approx. 70% olivine, 15% cpx, 15% plag (olivine gabbro)

However, it is likely that the olivine crystallising was more MgO rich (the Fo content of the olivine in the table is around 75%). A more MgO rich olivine would move the olivine point of the triangle so that there is no plagioclase component. This would then make a wehrlite.

● Lower MgO trend: approx. 10% olivine, 40% cpx, 50% plag (gabbro)

*Note that in the previous version of the practical the answer for the high MgO

Q3.9: Comment on the relationship between the composition of the crustal xenoliths observed in the Krafla and Theistareykir samples (see photos in lab or moodle) and the calculated cumulate compositions from Question 3.4.

● Remarkably similar - ultramafic cumulates and gabbros

Q3.10: Use your answers to the previous question and the mineral compositions to estimate the MgO content of the cumulate material formed from the high MgO magmas, and the low MgO magmas:

```
In [123... # You might want to write some code here...

# High MgO
print(0.7 * mn.MgO[mn.Mineral=='Olivine'].iloc[0]
      + 0.15 * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0]
      + 0.15 * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0])

# Low MgO
print(0.1 * mn.MgO[mn.Mineral=='Olivine'].iloc[0]
      + 0.4 * mn.MgO[mn.Mineral=='Clinopyroxene'].iloc[0]
      + 0.5 * mn.MgO[mn.Mineral=='Plagioclase'].iloc[0])
```

34.933499999999995

11.493000000000002

● High MgO = 34.9 wt%

● Low MgO = 11.5 wt%

Q3.11: If the primary melt contains ~14 wt% MgO, then use mass balance to estimate the fraction of crystallisation required to form liquids with 10 wt% MgO and 7 wt% MgO.

In [133]: `# You might want to write some code here...`

```
f10 = (14 - 10) / (34.9 - 10)
print(f10)

f7 = f10 + (10 - 7) / (11.5 - 7)
print(f7)
```

0.1606425702811245
0.8273092369477911

- 10 wt% MgO: 16% crystallisation
- 7 wt% MgO: 83% crystallisation

3.3. Summary

You have just extracted information about the composition of the cumulate material by assuming that the variation in the compositions of lava samples can be matched by fractional crystallisation of the minerals observed as macrocryst phases in the samples themselves. This seems like a reasonable approach. Another approach involves specifying a primary melt composition and the physical conditions of crystallisation and using parameterisations of experimental data and thermodynamic logic to predict the sequence of crystallisation, the mineral proportions, mineral compositions and the liquid line of descent. Programs such as [PETROLOG](#) by Leonid Danyushevsky and [MELTS](#) by Ghiorso and Sack provide useful tools for exploring crystallisation paths. If you have the time and the inclination you could try to run an LLD model and see if you can match the Icelandic data. There are multiple ways of running MELTS, including via a python interface using cloud based Jupyter Notebooks on the [ENKI server](#). The ENKI server is free to use, but requires registering.

4. Depth of crystallisation: Clinopyroxene-melt barometry

Clinopyroxene phenocrysts are rare in MORB. This observation, combined with the fact that many MORB fractionation trends are gabbro-controlled (i.e. ~40% modal clinopyroxene) has been termed the ‘pyroxene paradox’. However, it is not paradoxical if one understands the effect of pressure on the pyroxene stability field.

Fortunately, some Icelandic basalt flows contain clinopyroxene phenocrysts. More primitive flows contain pyroxenes with a distinctive ‘Heineken’ green colour – chromian diopsides. More evolved flows, such as the Krafla fissure eruptions, contain black/purple Ti-bearing augites. The composition of pyroxene phenocrysts is influenced both by the composition of the melt that they grow from and by the physical conditions (i.e. P, T) at the time of growth. [Neave & Putirka](#) have recently parameterised the results of hundreds of experiments that involve cpx-basalt equilibrium, and have produced a thermobarometer for Iceland compositions, shown below:

$$P = -26.27 + 39.16 \frac{T}{10^4} \ln\left[\frac{X_{jd}}{X_{NaO0.5}X_{AlO1.5}X_{SiO2}^2}\right] - 4.22 \ln(X_{DiHd}) + 78.43X_{AlO1.5} + 393.81\left(X_{NaO0.5}X_{KO0.5}\right)^2$$

where X_{jd} and X_{DiHd} describe the Jadeite and Diopside-Hedenbergite components in the clinopyroxene and $X_{NaO0.5}$, $X_{AlO1.5}$, X_{SiO2} and $X_{KO0.5}$ give cation fractions of various elements in the equilibrium melt.

Q4.1: Inspect this equation. The key term is the second on the right hand side. Does the relationship between pyroxene composition and pressure agree with that which you would expect from metamorphic petrology?

- Yes, high jadeite content at high pressure.

The estimated pressure of crystallisation correlates with the jadeite content of the pyroxene. Application of this thermobarometer to real rocks relies upon the assumption that the pyroxene crystals and the liquid composition are in equilibrium. Unfortunately, many Icelandic samples have crystalline groundmasses, so the whole-rock composition has to be used as an estimate of the liquid composition – problematic if the rock is rich in phenocrysts. The table below shows liquid and clinopyroxene phenocryst compositions from Borgarhraun (a primitive early postglacial flow from Theistareykir) and from the 1984 Krafla fissure eruption. Note the high Al and Cr content of the Borgarhraun cpx – a good sign of high pressure crystallisation.

This geobarometer (and many others) has been coded up in the Python module [Thermobar](#), which we will use here

In [82]: `import Thermobar`

Here are the compositions of co-existing clinopyroxene and liquids (assuming the whole rock represents the liquid composition):


```
In [107... krafla_cpx = {'SiO2': 51.0,  
                  'TiO2': 0.68,  
                  'Al2O3': 2.80,  
                  'FeOt': 8.79,  
                  'MnO': 0.24,  
                  'MgO': 16.55,  
                  'CaO': 18.79,  
                  'Na2O': 0.28,  
                  'K2O': 0.0,  
                  'Cr2O3': 0.85}  
  
krafla_liq = {'SiO2': 49.45,  
             'TiO2': 1.66,  
             'Al2O3': 14.78,  
             'FeOt': 12.15,  
             'MnO': 0.21,  
             'MgO': 7.61,  
             'CaO': 11.64,  
             'Na2O': 2.06,  
             'K2O': 0.23,  
             'Cr2O3': 0.04}  
  
borg_cpx = {'SiO2': 51.28,  
            'TiO2': 0.26,  
            'Al2O3': 4.79,  
            'FeOt': 4.18,  
            'MnO': 0.14,  
            'MgO': 17.37,  
            'CaO': 20.09,  
            'Na2O': 0.22,  
            'K2O': 0.0,  
            'Cr2O3': 1.40}  
  
borg_liq = {'SiO2': 48.65,  
            'TiO2': 0.69,  
            'Al2O3': 14.52,  
            'FeOt': 9.28,  
            'MnO': 0.17,  
            'MgO': 12.01,  
            'CaO': 12.84,  
            'Na2O': 1.62,  
            'K2O': 0.05,  
            'Cr2O3': 0.12}
```

Let's see what Thermobar needs:

```
In [83]: Thermobar.calculate_cpx_liq_press?
```

Signature:

```

Thermobar.calculate_cpx_liq_press(
    *,
    equationP,
    cpx_comps=None,
    liq_comps=None,
    meltmatch=None,
    T=None,
    eq_tests=False,
    Fe3Fet_Liq=None,
    H2O_Liq=None,
    sigma=1,
    Kd_Err=0.03,
)

```

Docstring:

Clinopyroxene-Liquid barometer, calculates pressure in kbar
(and equilibrium tests as an option)

Parameters

cpx_comps: pandas.DataFrame

Clinopyroxene compositions with column headings SiO2_Cpx, MgO_Cpx etc.

liq_comps: pandas.DataFrame

Liquid compositions with column headings SiO2_Liq, MgO_Liq etc.

EquationP: str

choose from:

```

| P_Put1996_eqP1 (T-dep, H2O-indep)
| P_Mas2013_eqPalk1 (T-dep, H2O-indep, alk adaption of P1)
| P_Put1996_eqP2 (T-dep, H2O-indep)
| P_Mas2013_eqPalk2 (T-dep, H2O-indep, alk adaption of P2)
| P_Put2003 ((T-dep, H2O-indep)
| P_Neave2017 (T-dep, H2O-indep)
| P_Put2008_eq30 (T-dep, H2O-dep)
| P_Put2008_eq31 (T-dep, H2O-dep)
| P_Put2008_eq32c (T-dep, H2O-dep)
| P_Mas2013_eqalk32c (T-dep, H2O-dep, alk adaption of 32c)
| P_Petrelli2020_Cpx_Liq (Returns voting)
| P_Jorgenson2022_Cpx_Liq (Returns voting)
| P_Petrelli2020_Cpx_Liq_onnx (Uses onnx, so consistent results, no voting)
| P_Wang2021_eq1

```

T: float, int, pandas.Series, str

Temperature in Kelvin to perform calculations at.

Only needed for T-sensitive barometers.

If enter T="Solve", returns a partial function

Else, enter an integer, float, or panda series

eq_tests: bool

If False, just returns pressure (default) as a panda series

If True, returns pressure, Values of Eq tests,

as well as user-entered cpx and liq comps and components

Returns

Pressure in kbar: pandas.Series

If eq_tests is False

Pressure in kbar + eq Tests + input compositions: pandas.DataFrame

If eq_tests is True

File: /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/Thermobar/clinopyroxene_thermobarometry.py

Type: function

So we need to give it the compositions as DataFrames with specific column headings. We can make these by manipulating the dictionaries defined above:

```
In [108... krafla_liq_renamed = {}
for ox in krafla_liq:
    krafla_liq_renamed[ox + '_Liq'] = krafla_liq[ox]

krafla_liq_df = pd.DataFrame([pd.Series(krafla_liq_renamed)])

krafla_cpx_renamed = {}
for ox in krafla_cpx:
    krafla_cpx_renamed[ox + '_Cpx'] = krafla_cpx[ox]

krafla_cpx_df = pd.DataFrame([pd.Series(krafla_cpx_renamed)])

borg_liq_renamed = {}
for ox in borg_liq:
    borg_liq_renamed[ox + '_Liq'] = borg_liq[ox]

borg_liq_df = pd.DataFrame([pd.Series(borg_liq_renamed)])

borg_cpx_renamed = {}
for ox in borg_cpx:
    borg_cpx_renamed[ox + '_Cpx'] = borg_cpx[ox]

borg_cpx_df = pd.DataFrame([pd.Series(borg_cpx_renamed)])
```

Now we will call a function which will sumultaneously solve for temperature and pressure from the cpx-liquid equilibrium. The temperature is calculated using a cpx-liquid thermometer from [Putirka \(2008\)](#). Note the extra information we have to give it:

```
In [116... Thermobar.calculate_cpx_liq_press_temp(equationP='P_Neave2017',
                                         equationT='T_Put2008_eq33',
                                         cpx_comps = krafla_cpx_df,
                                         liq_comps = krafla_liq_df,
                                         Fe3Fet_Liq=0.15,
                                         H2O_Liq=0.0)
```

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/Thermobar/core.py:1561: FutureWarning: In a future version, the Index constructor will not infer numeric dtypes when passed object-dtype sequences (matching Series behavior)

```
cpx_calc.loc[(AlVI_minus_Na<0), 'Jd']=cpx_calc['Al_VI_cat_6ox']
```

Out[116]:

	P_kbar_calc	T_K_calc	Delta_P_kbar_iter	Delta_T_K_iter
0	3.047989	1459.300665	0.0	0.0

Convert the temperature to °C:

```
In [117... 1459 - 273.15
```

Out[117]: 1185.85

Q4.2: Repeat this calculation for Borgarhraun, using the data supplied above.

```
In [118... # Your code here...

Thermobar.calculate_cpx_liq_press_temp(equationP='P_Neave2017',
                                         equationT='T_Put2008_eq33',
                                         cpx_comps = borg_cpx_df,
                                         liq_comps = borg_liq_df,
                                         Fe3Fet_Liq=0.15,
                                         H2O_Liq=0.0)
```

Out[118]:

	P_kbar_calc	T_K_calc	Delta_P_kbar_iter	Delta_T_K_iter
0	4.817255	1499.525488	0.0	0.0

```
In [119... # Your code here...

1499.5 - 273.15
```

Out[119]: 1226.35

Q4.3: Why do you think that the pyroxenes present in the samples in the thin sections show a distinctive anhedral shape?

HINT: You can look at [this paper](#) for a little bit of extra info.

● *Clinopyroxene grew at depth, but goes out of equilibrium with melting during decompression between storage at high pressure and eruption. Cpx phase field shrinks with decreasing pressure.*

5. Depth of crystallisation: Order of crystallisation

Crystallisation experiments on MORB-like compositions indicate that the order of appearance of phases on the liquidus (i.e. the order of crystallisation) is dependent on pressure. At pressures of >6 kbar, the crystallisation assemblages evolve from ol → ol+cpx → ol+cpx+plg. At lower pressures, plagioclase appears before clinopyroxene, giving, ol → ol+plg → ol+plg+cpx.

Recall from the thin section observations that the Borgarhraun eruption contains a number of cognate xenoliths, including gabbros, wehrlites (ol+cpx cumulates) and occasional troctolites (ol+ plag cumulates).

Q5.11: Use the information in Figure 3 to estimate the order of crystallisation and the pressure of crystallisation (higher or lower than 6 kbar?):



Figure 3: *Compositions of co-existing mineral pairs in Borgarhraun cognate xenoliths. The blue circles (at higher Fo) are wehrlite cumulates. The red squares are gabbroic cumulates.*

● *cpx first, more than 6 kbar*

6. Depth of crystallisation: Liquid composition

Experimental results have also shown that the composition of liquid in equilibrium with olivine, clinopyroxene and plagioclase varies as a function of pressure. A number of workers have parameterised this relationship. Projections of Krafla and Theistareykir sample compositions and predicted ol-cpx-plg saturated melts are shown in Figure 4.



Figure 4: *The compositions of whole-rock samples are projected from plagioclase onto the olivine-diopside-quartz face of the basalt tetrahedron. Circles are Theistareykir (including Borgarhraun) and squares Krafla. The predicted liquid compositions in equilibrium with ol-cpx-plag are shown as lines. Each line corresponds to a different pressure (here given in GPa).*

Q6.1: Are the pressure estimates of Borgarhraun crystallisation from Figure 4 in agreement with those from other barometric methods?

● *Not entirely. Crystallisation pressure estimates range from 5 kbar to about 10 kbar. Some of this represents systematic errors (disequilibrium), some uncertainty in the methods - this discrepancy will hopefully be resolved in near future.*

Q6.2: At what depth does crystallization of Borgarhraun melt take place? Is this above or below the Moho in northern Iceland?

● *Use density estimate and crustal thickness from seismic survey in figure to show - given uncertainty, likely to be close to Moho, within 5 km or so*

Q6.3: Does crystallization of Krafla melt take place in the shallow chamber that has been seismically imaged at 3-5 km depth under the caldera?

● *No, it appears to crystallise in mid-crust*

Q6.4: Use the geophysical, petrological constraints to draw a conceptual model of crystallization under Krafla and Theistareykir. You may use the diagrams in Figure 1 as a starting point. Does the gabbro-glacier or stacked-sills model provide a better representation for crystallization under Iceland?

● *Stacked sills - substantial crystallisation must take place in the lower crust. Krafla melt compositions have lost >50% of their original mass by crystallisation, and are stored in the mid-crust. Indicates that shallow melt lens is not dominant location of crystallisation and crustal formation*

In []: