S8_CerroNegro_isobar_comparison

March 19, 2021

[1]:

```
import sys
sys.path.append('../../../..')

import VESIcal as v
import numpy as np
import scipy
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:

```
#Import the data
basalts = v.BatchFile("../../Datasets/cerro_negro.xlsx")

#Calculate the average composition of the entire dataset
columns = list(basalts.get_data())
avg_vals = []
for col in columns:
    try:
        avg_vals.append(basalts.data[col].mean())
    except:
        avg_vals.append("AVG")

avg_dict = dict(zip(columns, avg_vals))
avg_dict = v.get_oxides(avg_dict)
```

[3]:

```
#Calculate isobars for all samples at 3,000 bars
isobar_list = []
for index, row in basalts.get_data().iterrows():
    isobar_list.append(v.calculate_isobars_and_isopleths(sample=basalts.
 ↪get_sample_composition(samplename=row.name, asSampleClass=True),␣
 ↪temperature=1200, pressure_list=[3000], isopleth_list=[0.5],␣
 ↪print_status=True).result[0])
```

```
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
```

```
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
```

```
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
Calculating isobar at 3000 bars
 done.
Done!
```

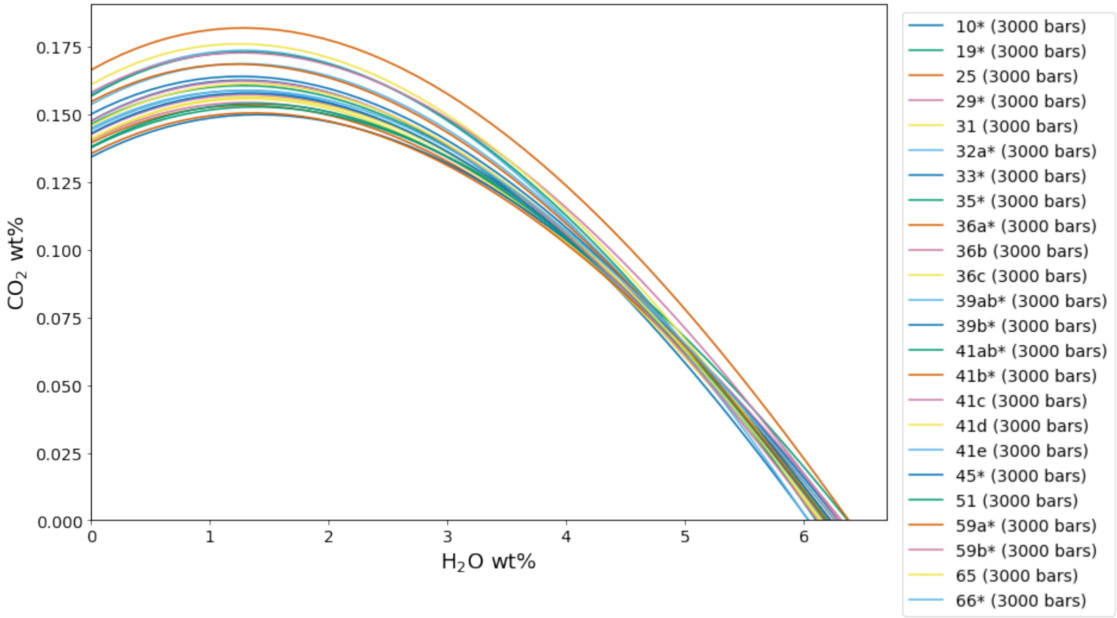
[5]:

```
#Calculate isobar at 3,000 bars for "Average Sample"
avg_isobar = v.calculate_isobars_and_isopleths(sample=v.Sample(avg_dict),␣
 ↪temperature=1200, pressure_list=[3000], isopleth_list=[0.5],␣
 ↪print_status=True).result[0]
```

```
Calculating isobar at 3000 bars
 done.
Done!
```

[6]:

```
#Plot all isobars from dataset
fig, ax = v.plot(isobars=[isobar for isobar in isobar_list], isobar_labels=[row.
 ↪name for index, row in basalts.get_data().iterrows()])
v.show()
```



[7]:

```
#calculate area under each curve for dataset and "Average Sample"
areas = []
samples = [row.name for index, row in basalts.get_data().iterrows()]
for isobar in isobar_list:
    x_vals = np.array([row["H2O_liq"] for index, row in isobar.iterrows()])
    y_vals = np.array([row["CO2_liq"] for index, row in isobar.iterrows()])
    area_under_the_curve = scipy.integrate.simps(y_vals, x_vals)
    areas.append(area_under_the_curve)

average_area = scipy.integrate.simps(avg_isobar['CO2_liq'],␣
 ↪avg_isobar['H2O_liq'])
```

[8]:

```
#Get maximum and minimum areas from dataset, with corresponding sample names
area_dict = dict(zip(samples, areas))
max_sample = max(area_dict, key=area_dict.get)
min_sample = min(area_dict, key=area_dict.get)
print("ISM values for entire dataset: \n" + str(area_dict) + "\n")
print("'Average Sample' ISM = " + str(average_area))
```

```
ISM values for entire dataset:
{'10*': 0.7022828487675895, '19*': 0.7099131142452471, '25': 0.6857308764836786,
'29*': 0.6959245973174225, '31': 0.6857679600090202, '32a*': 0.6981011318157468,
'33*': 0.6794449586415775, '35*': 0.6983522136970503, '36a*':
0.6737990590298144, '36b': 0.700302027451814, '36c': 0.6967336243144274,
'39ab*': 0.7498555237770486, '39b*': 0.7254223767902859, '41ab*':
0.7551957184715765, '41b*': 0.823496578531669, '41c': 0.7722064502996333, '41d':
0.7707799763517975, '41e': 0.7414493381948651, '45*': 0.6989410687176959, '51':
```

```
0.68721735630067, '59a*': 0.739335114894365, '59b*': 0.7091202270350897, '65':
0.7108222436239792, '66*': 0.7105778333531161}

'Average Sample' ISM = 0.7160086630830478
```

[9]:

```
#Now, calculate isobars for the max and min samples at multiple pressures
max_isobars, max_isopleths = v.calculate_isobars_and_isopleths(sample=basalts.
 ↪get_sample_composition(max_sample, asSampleClass=True), temperature=1200,␣
 ↪pressure_list=[500, 1000, 2000, 3000, 4000], isopleth_list=[0.5],␣
 ↪print_status=True).result
min_isobars, min_isopleths = v.calculate_isobars_and_isopleths(sample=basalts.
 ↪get_sample_composition(min_sample, asSampleClass=True), temperature=1200,␣
 ↪pressure_list=[500, 1000, 2000, 3000, 4000], isopleth_list=[0.5],␣
 ↪print_status=True).result

#Calculate isobars for the average composition
avg_isobars, avg_isopleths = v.calculate_isobars_and_isopleths(sample=v.
 ↪Sample(avg_dict), temperature=1200, pressure_list=[500, 1000, 2000, 3000,␣
 ↪4000], isopleth_list=[0.5], print_status=True).result
```

```
Calculating isobar at 500 bars
 done.
Calculating isobar at 1000 bars
 done.
Calculating isobar at 2000 bars
 done.
Calculating isobar at 3000 bars
 done.
Calculating isobar at 4000 bars
 done.
Done!
Calculating isobar at 500 bars
 done.
Calculating isobar at 1000 bars
 done.
Calculating isobar at 2000 bars
 done.
Calculating isobar at 3000 bars
 done.
Calculating isobar at 4000 bars
 done.
Done!
Calculating isobar at 500 bars
 done.
Calculating isobar at 1000 bars
 done.
Calculating isobar at 2000 bars
 done.
```

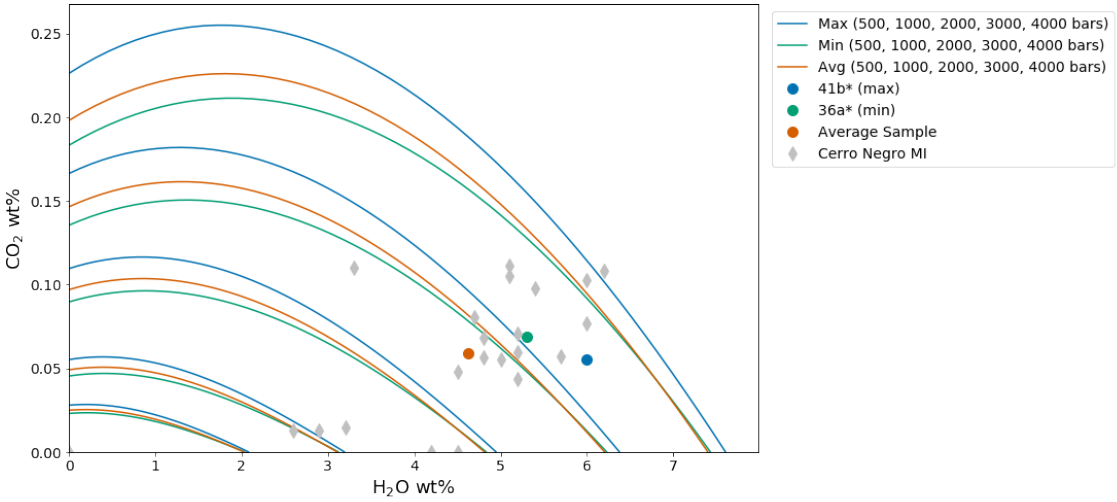
```
Calculating isobar at 3000 bars
 done.
Calculating isobar at 4000 bars
 done.
Done!
```

[10]:

```
#Make dataset with all data except for max and min values
other_data = basalts.get_data().drop([max_sample, min_sample])
```

[13]:

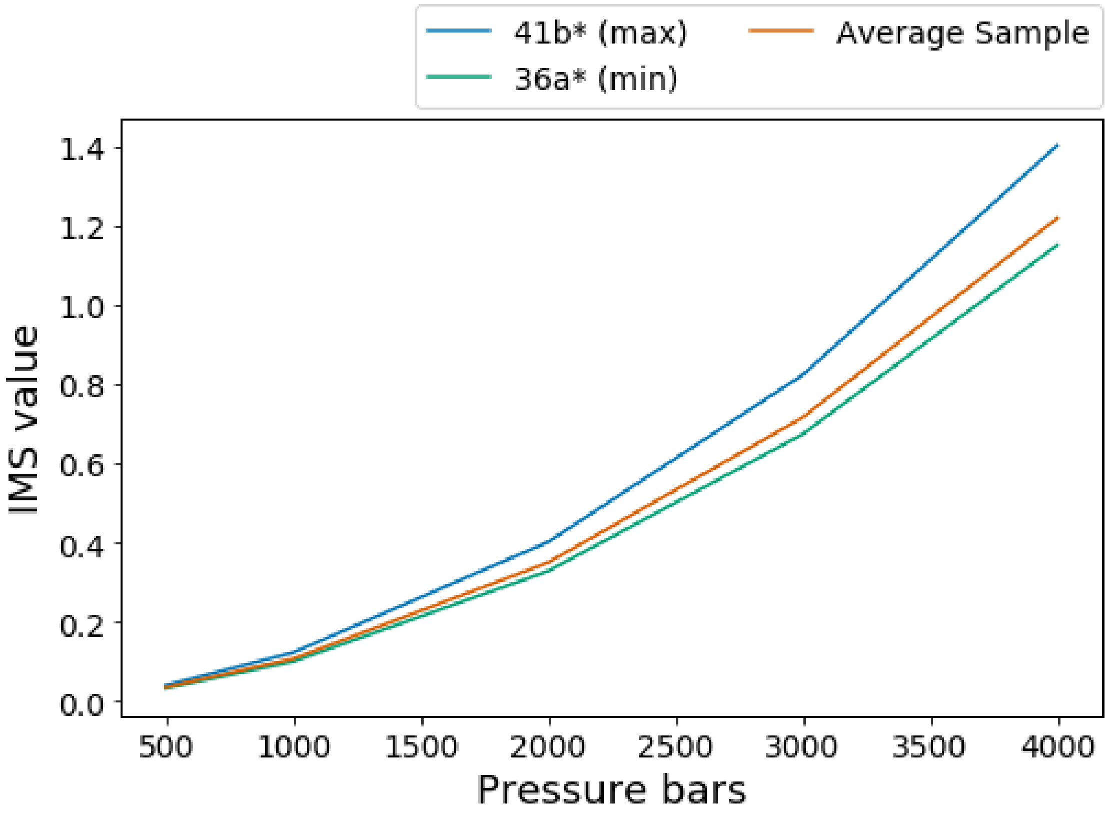
```
#set up what to pass to v.plot
isobars = [max_isobars,
           min_isobars,
           avg_isobars]

isobar_labels = ["Max",
                 "Min",
                 "Avg"]

custom_H2O=[basalts.get_sample_composition(max_sample)["H2O"],
            basalts.get_sample_composition(min_sample)["H2O"],
            avg_dict["H2O"],
            other_data["H2O"]]

custom_CO2=[basalts.get_sample_composition(max_sample)["CO2"],
            basalts.get_sample_composition(min_sample)["CO2"],
            avg_dict["CO2"],
            other_data["CO2"]]

custom_labels = [str(max_sample) + " (max)",
                 str(min_sample) + " (min)",
                 "Average Sample",
                 "Cerro Negro MI"]

custom_colors = [v.vplot.color_list[0],
                 v.vplot.color_list[1],
                 v.vplot.color_list[2],
                 'silver']

custom_symbols = ['o',
                  'o',
                  'o',
                  'd']

fig, ax = v.plot(isobars=isobars, isobar_labels=isobar_labels,
       custom_H2O=custom_H2O,
       custom_CO2=custom_CO2,
       custom_labels=custom_labels,
       custom_colors=custom_colors,
       custom_symbols=custom_symbols)

v.show()
```



[14]:

```
pressure_vals = [500, 1000, 2000, 3000, 4000]

max_IMS_dict = {}
min_IMS_dict = {}
avg_IMS_dict = {}

IMS_dicts = [max_IMS_dict,
             min_IMS_dict,
             avg_IMS_dict]

for i in range(len(isobars)):
    IMS_dicts[i].update({"Pressure": pressure_vals})
    IMS_list = []
    for pressure in pressure_vals:
        IMS_list.append(scipy.integrate.simps(isobars[i].
 ↪loc[isobars[i]['Pressure']==pressure]["CO2_liq"], isobars[i].
 ↪loc[isobars[i]['Pressure']==pressure]["H2O_liq"]))
        IMS_dicts[i].update({"IMS": IMS_list})


labels = ["Maximum, Minimum, Average"]

fig, ax = plt.subplots(1, figsize = (8,5))

for i in range(len(IMS_dicts)):
    ax.plot(IMS_dicts[i]["Pressure"], IMS_dicts[i]["IMS"],␣
 ↪label=custom_labels[i])
    ax.set_xlabel("Pressure bars")
    ax.set_ylabel("IMS value")

ax.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower right',
           ncol=2, borderaxespad=0.)

#fig.savefig('Cerro_Negro_img3.pdf')
```

[14]:
```
<matplotlib.legend.Legend at 0x7fb6a9bd9f50>
```



[15]:
```
max_IMS_dict
```

[15]:
```
{'Pressure': [500, 1000, 2000, 3000, 4000],
 'IMS': [0.03964674041589094,
  0.12230513394511863,
  0.40131268243443224,
  0.823496578531669,
  1.4033483467221493]}
```

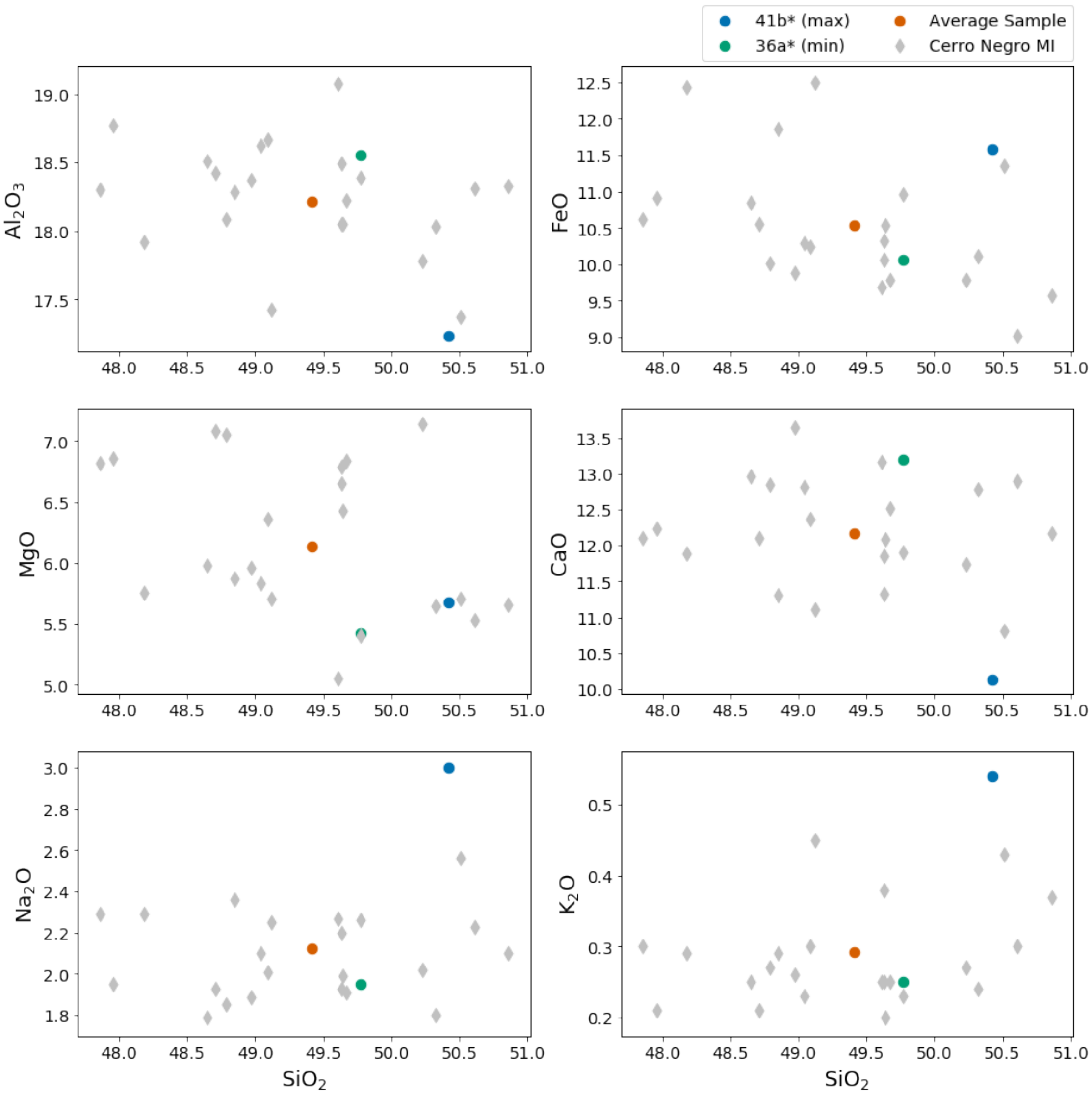
[16]:

```
other_oxides = ["Al2O3", "FeO", "MgO", "CaO", "Na2O", "K2O"]
my_samples = [basalts.get_sample_composition(max_sample),
              basalts.get_sample_composition(min_sample),
             avg_dict,
             other_data]

fig, axs = plt.subplots(3,2, figsize = (15,15))
print(len(axs))

for j in range(len(my_samples)):
    axs[0][0].scatter(my_samples[j]["SiO2"], my_samples[j]["Al2O3"],␣
 ↪marker=custom_symbols[j], s=70, color=custom_colors[j],␣
 ↪label=custom_labels[j])
    axs[0][0].set_ylabel("Al$_2$O$_3$")
    axs[0][1].scatter(my_samples[j]["SiO2"], my_samples[j]["FeO"],␣
 ↪marker=custom_symbols[j], s=70, color=custom_colors[j],␣
 ↪label=custom_labels[j])
    axs[0][1].set_ylabel("FeO")
    axs[1][0].scatter(my_samples[j]["SiO2"], my_samples[j]["MgO"],␣
 ↪marker=custom_symbols[j], s=70, color=custom_colors[j],␣
 ↪label=custom_labels[j])
    axs[1][0].set_ylabel("MgO")
    axs[1][1].scatter(my_samples[j]["SiO2"], my_samples[j]["CaO"],␣
 ↪marker=custom_symbols[j], s=70, color=custom_colors[j],␣
 ↪label=custom_labels[j])
    axs[1][1].set_ylabel("CaO")
    axs[2][0].scatter(my_samples[j]["SiO2"], my_samples[j]["Na2O"],␣
 ↪marker=custom_symbols[j], s=70, color=custom_colors[j],␣
 ↪label=custom_labels[j])
    axs[2][0].set_ylabel("Na$_2$O")
    axs[2][0].set_xlabel("SiO$_2$")
    axs[2][1].scatter(my_samples[j]["SiO2"], my_samples[j]["K2O"],␣
 ↪marker=custom_symbols[j], s=70, color=custom_colors[j],␣
 ↪label=custom_labels[j])
    axs[2][1].set_ylabel("K$_2$O")
    axs[2][1].set_xlabel("SiO$_2$")

axs[0][1].legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower right',
           ncol=2, borderaxespad=0.)

#fig.savefig('Cerro_Negro_img4.pdf')
```

```
3
```

[16]:

```
<matplotlib.legend.Legend at 0x7fb6aa2606d0>
```



1 Alternative plots

[18]:
```
#Calculate Saturation Pressure for all samples
other_file = v.BatchFile_from_DataFrame(dataframe=other_data)
satP_other = other_file.calculate_saturation_pressure(temperature=1200)
satP_max = v.calculate_saturation_pressure(sample=basalts.
 ↪get_sample_composition(max_sample, asSampleClass=True), temperature=1200,␣
 ↪verbose=True).result
satP_min = v.calculate_saturation_pressure(sample=basalts.
 ↪get_sample_composition(min_sample, asSampleClass=True), temperature=1200,␣
 ↪verbose=True).result
```

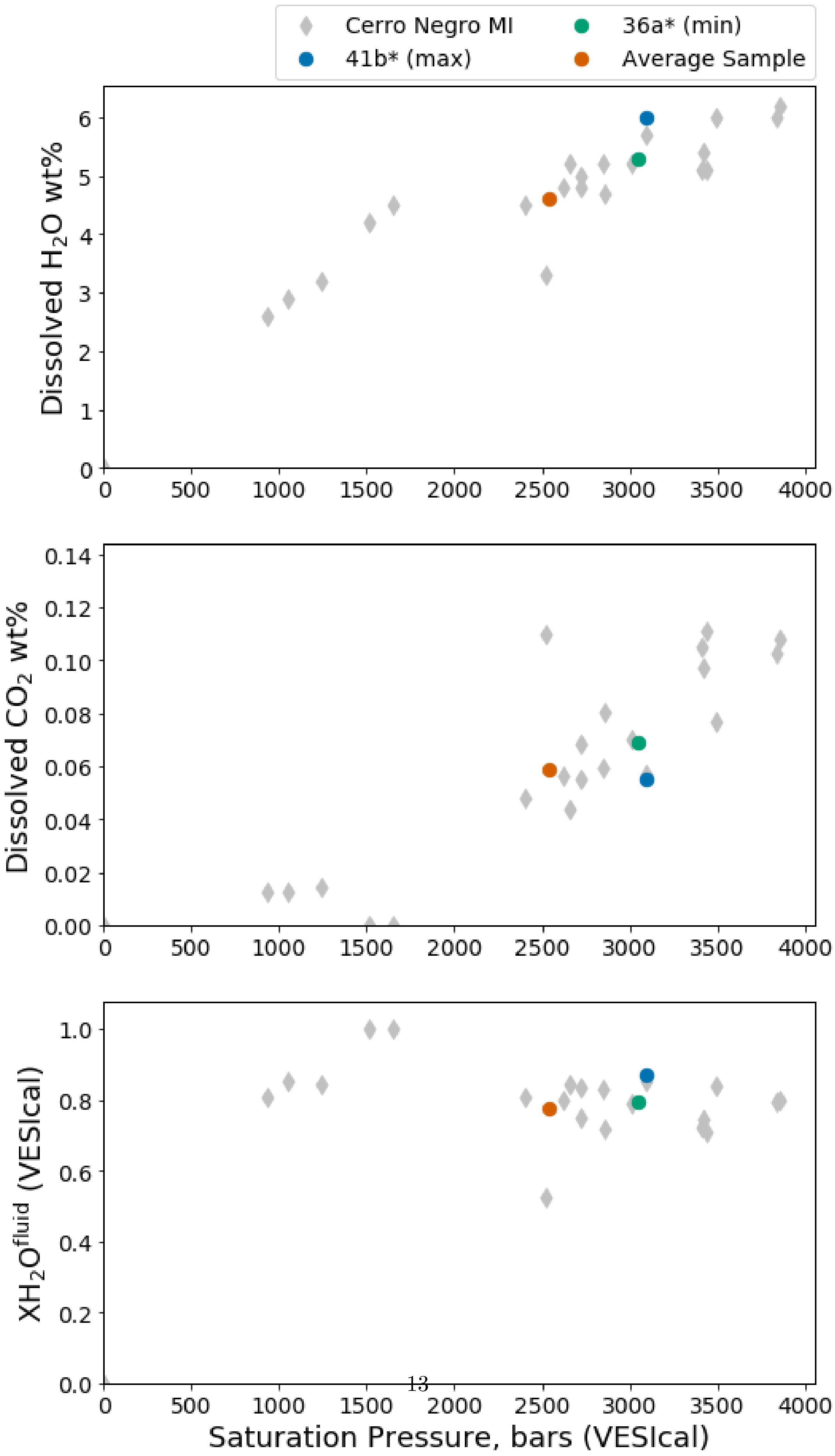
```
satP_avg = v.calculate_saturation_pressure(sample=v.Sample(avg_dict),␣
 ↪temperature=1200, verbose=True).result
```

```
[====================] 100%  Working on sample 66*
```

[20]:

```
#Create alternative plots using Matplotlib
single_data = [satP_max,
               satP_min,
               satP_avg]

single_samples = [basalts.get_sample_composition(max_sample),
                  basalts.get_sample_composition(min_sample),
                  avg_dict]

fig, axs = plt.subplots(3, figsize = (8,15))
axs[0].scatter(satP_other["SaturationP_bars_VESIcal"], satP_other["H2O"],␣
 ↪marker=custom_symbols[3], s=70, color=custom_colors[3],␣
 ↪label=custom_labels[3])
axs[1].scatter(satP_other["SaturationP_bars_VESIcal"], satP_other["CO2"],␣
 ↪marker=custom_symbols[3], s=70, color=custom_colors[3],␣
 ↪label=custom_labels[3])
axs[2].scatter(satP_other["SaturationP_bars_VESIcal"],␣
 ↪satP_other["XH2O_fl_VESIcal"], marker=custom_symbols[3], s=70,␣
 ↪color=custom_colors[3], label=custom_labels[3])

for j in range(len(single_data)):
    axs[0].scatter(single_data[j]["SaturationP_bars"],␣
 ↪single_samples[j]["H2O"], marker=custom_symbols[j], s=70,␣
 ↪color=custom_colors[j], label=custom_labels[j])
    axs[0].set_ylabel("Dissolved H$_2$O wt%")
    axs[0].set_ylim(0)
    axs[0].set_xlim(0)
    axs[1].scatter(single_data[j]["SaturationP_bars"],␣
 ↪single_samples[j]["CO2"], marker=custom_symbols[j], s=70,␣
 ↪color=custom_colors[j], label=custom_labels[j])
    axs[1].set_ylabel("Dissolved CO$_2$ wt%")
    axs[1].set_ylim(0)
    axs[1].set_xlim(0)
    axs[2].scatter(single_data[j]["SaturationP_bars"],␣
 ↪single_data[j]["XH2O_fl"], marker=custom_symbols[j], s=70,␣
 ↪color=custom_colors[j], label=custom_labels[j])
    axs[2].set_ylabel("XH$_2$O$^{fluid}$ (VESIcal)")
    axs[2].set_ylim(0)
    axs[2].set_xlabel("Saturation Pressure, bars (VESIcal)")
    axs[2].set_xlim(0)

axs[0].legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower right',
           ncol=2, borderaxespad=0.)

#fig.savefig('Cerro_Negro_img5.pdf')
```

```
[20]: <matplotlib.legend.Legend at 0x7fb6aa3bba10>
```



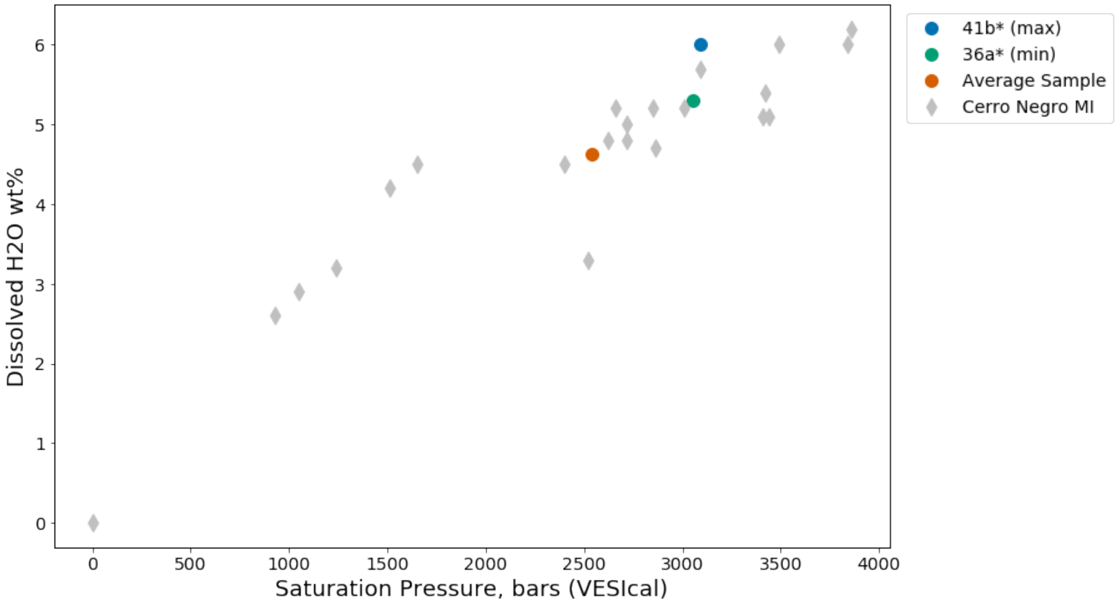
Cerro Negro MI
41b* (max)
36a* (min)
Average Sample
Dissolved H_2O wt%
0
1
2
3
4
5
6
Dissolved CO_2 wt%
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
XH_2O^{fluid} (VESIcal)
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
Saturation Pressure, bars (VESIcal)

[22]:

```
#Create alternative plots using VESIcal's scatterplot() function
single_samples = [basalts.get_sample_composition(max_sample),
                 basalts.get_sample_composition(min_sample),
                 avg_dict]

v.vplot.scatterplot(custom_x=[satP_max['SaturationP_bars'],
                        satP_min['SaturationP_bars'],
                        satP_avg['SaturationP_bars'],
                       satP_other['SaturationP_bars_VESIcal']],
              custom_y=[single_samples[0]['H2O'],
                        single_samples[1]['H2O'],
                        single_samples[2]['H2O'],
                       satP_other['H2O']],
             custom_symbols=custom_symbols,
             custom_colors=custom_colors,
             custom_labels=custom_labels,
             xlabel="Saturation Pressure, bars (VESIcal)",
             ylabel="Dissolved H2O wt%")
```

[22]:

```
(<Figure size 864x576 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7fb6aa3cae90>)
```



2 Calculate saturation pressures for each composition

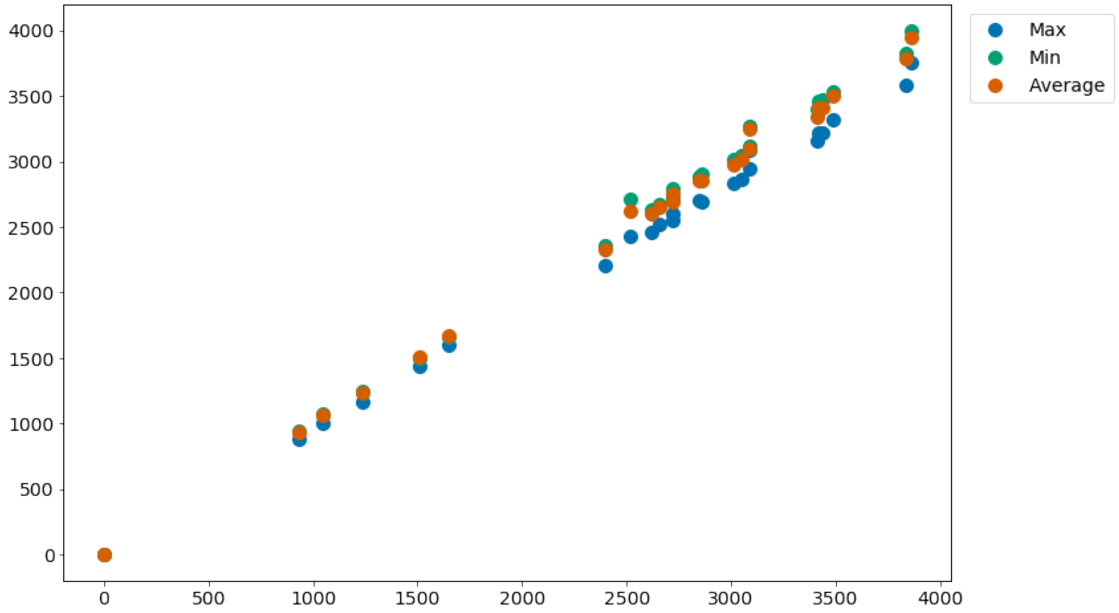
Here we calculate the saturation pressures of each melt inclusion using: a) the composition of the melt inclusion; b) the composition of the “minimum” melt inclusion (36a*); c) the composition of the “maximum” melt inclusion (41b*); and d) the composition of the “average” melt inclusion as calculated above.

[23]:
```
satP_data_orig = v.BatchFile('cerro_negro_satP_compare.xlsx')
satP_data_min = v.BatchFile('cerro_negro_satP_compare.xlsx', sheet_name='min')
satP_data_max = v.BatchFile('cerro_negro_satP_compare.xlsx', sheet_name='max')
satP_data_avg = v.BatchFile('cerro_negro_satP_compare.xlsx', sheet_name='avg')
```

[26]:
```
satP_orig = satP_data_orig.calculate_saturation_pressure(temperature=1200)
satP_min = satP_data_min.calculate_saturation_pressure(temperature=1200)
satP_max = satP_data_max.calculate_saturation_pressure(temperature=1200)
satP_avg = satP_data_avg.calculate_saturation_pressure(temperature=1200)
```

```
[====================] 100%  Working on sample 66*
[====================] 100%  Working on sample 66*
[====================] 100%  Working on sample 66*
[====================] 100%  Working on sample 66*
```

[27]:
```
fig, ax = v.vplot.scatterplot(custom_x=[satP_orig["SaturationP_bars_VESIcal"],␣
 ↪satP_orig["SaturationP_bars_VESIcal"],␣
 ↪satP_orig["SaturationP_bars_VESIcal"]],
            custom_y=[satP_max["SaturationP_bars_VESIcal"],␣
 ↪satP_min["SaturationP_bars_VESIcal"], satP_avg["SaturationP_bars_VESIcal"]],
            custom_labels=["Max", "Min", "Average"])
v.show()
```


Max
Min
Average