capstone_code

February 6, 2020

1 Plot the Likelihood of Finding Cobalt Deposits in British Columbia

By Brandon Loong, 02/06/2020

1.0.1 Load packages and data

[1]:
```
# Load packages using correct env
import geopandas as gpd
import shapely.geometry as shp
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import seaborn as sns
import numpy as np
import random
%matplotlib inline

# Import data
INPUT_FILE = "BedrockP.shp"
data = gpd.read_file(INPUT_FILE)
```

1.0.2 Preview data structure, info, and columns

[2]:
```
data.head(2)
```

[2]:
```
       gid      upid     area_m2 strat_unit       era      period  \
0  33649.0  9c192ffb   4368828.0     uKPCvl  Mesozoic  Cretaceous
1  26344.0  824df15e  34431394.0       lKGsv  Mesozoic  Cretaceous

          strat_age                          strat_name       gp_suite  \
0  Upper Cretaceous  Powell Creek Formation - upper unit           None
1  Lower Cretaceous                       Gambier Group  Gambier Group

               fm_lithodm  …  terrane basin basin_age              project  \
0  Powell Creek Formation  …  Overlap  None      None  Chilcotin-Bonaparte
1                    None  …  Overlap  None      None            Mid-coast
```

```
                                               src_url  \
0  http://www.em.gov.bc.ca/Mining/Geoscience/Publ…
1  http://www.em.gov.bc.ca/Mining/Geoscience/Publ…

                                             src_ref_s  \
0  Schiarizza et al., 1997, Chilcotin-Bonaparte, …
1  Bellefontaine et al., 1994, Mid-coast, BCGS, O…

                                 map_comp   edit_date  \
0  Schiarizza, 2017, Chicotin-Bonaparte  2018-04-05
1               Massey et al., 2005, BC  2018-04-05

                                pub_org  \
0  British Columbia Geological Survey
1  British Columbia Geological Survey

                                              geometry
0  POLYGON ((466212.8026486822 5669781.23969778, …
1  POLYGON ((541326.0118804163 5529535.058946755,…

[2 rows x 28 columns]
```

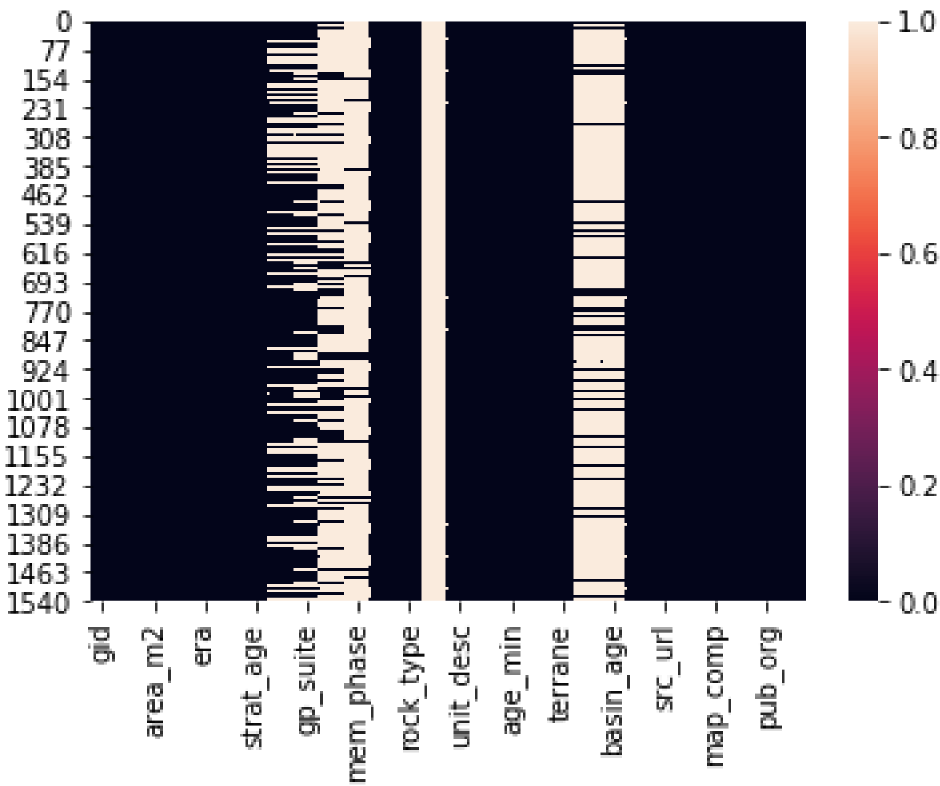
[3]:
```python
data.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1596 entries, 0 to 1595
Data columns (total 28 columns):
gid           1596 non-null float64
upid          1596 non-null object
area_m2       1596 non-null float64
strat_unit    1596 non-null object
era           1596 non-null object
period        1596 non-null object
strat_age     1596 non-null object
strat_name    1184 non-null object
gp_suite      1036 non-null object
fm_lithodm    443 non-null object
mem_phase     76 non-null object
rock_class    1596 non-null object
rock_type     1596 non-null object
rk_char       2 non-null object
unit_desc     1596 non-null object
age_max       1596 non-null object
age_min       1596 non-null object
belt          1596 non-null object
terrane       1596 non-null object
basin         240 non-null object
```

```
basin_age     240 non-null object
project       1596 non-null object
src_url       1596 non-null object
src_ref_s     1596 non-null object
map_comp      1596 non-null object
edit_date     1596 non-null object
pub_org       1596 non-null object
geometry      1596 non-null object
dtypes: float64(2), object(26)
memory usage: 349.2+ KB
```

[4]:
```python
# View any missing data
sns.heatmap(data.isna())
```

[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f953a2055c0>
```



These missing data seem irrelevant to the end goal of creating heat map, so ignore for now

[5]:
```python
# View some columns with granodioritic and ultramafic data
data.iloc[12:19,4:13]
```

```
[5]:                    era                       period  \
12  Paleozoic-Mesozoic  Carboniferous to Jurassic
13            Mesozoic                   Cretaceous
14            Cenozoic                    Paleogene
15            Cenozoic                    Paleogene
16  Paleozoic-Mesozoic  Carboniferous to Jurassic
17           Paleozoic                      Permian
18            Mesozoic     Jurassic to Cretaceous

                                   strat_age  \
12  Middle Mississippian to Middle Jurassic
13                         Lower Cretaceous
14                                   Eocene
15                                   Eocene
16  Middle Mississippian to Middle Jurassic
17                                  Permian
18        Late Jurassic to Early Cretaceous

                                            strat_name  \
12                               Bridge River Complex
13               Taylor Creek Group - Lizard Formation
14                                Mission Ridge suite
15                                Mission Ridge suite
16                               Bridge River Complex
17  Shulaps Ultramafic Complex - Serpentinite Mela…
18                                               None

                    gp_suite                 fm_lithodm mem_phase  \
12        Bridge River Complex                       None      None
13          Taylor Creek Group           Lizard Formation      None
14         Mission Ridge suite                       None      None
15         Mission Ridge suite                       None      None
16        Bridge River Complex                       None      None
17  Shulaps Ultramafic Complex  Serpentinite Melange unit      None
18                        None                       None      None

           rock_class                              rock_type
12  sedimentary rocks  marine sedimentary and volcanic rocks
13  sedimentary rocks            undivided sedimentary rocks
14    intrusive rocks         granodioritic intrusive rocks
15    intrusive rocks         granodioritic intrusive rocks
16  sedimentary rocks  marine sedimentary and volcanic rocks
17   ultramafic rocks                       ultramafic rocks
18    intrusive rocks        quartz dioritic intrusive rocks
```

View the number of rock classes and types

[6]:
```python
# Rock class counts
data['rock_class'].value_counts()
```

[6]:
```
sedimentary rocks    556
intrusive rocks      455
volcanic rocks       435
metamorphic rocks    104
ultramafic rocks      46
Name: rock_class, dtype: int64
```

[7]:
```python
# Unique rock types
nrocks = data['rock_type'].nunique()
print(f"Number of rock_types: {nrocks}")

# View most common rock types and where gran/serp/ultra line up
data['rock_type'].value_counts().head(12)
```

```
Number of rock_types: 43
```

[7]:
```
undivided sedimentary rocks                          246
undivided volcanic rocks                             149
granodioritic intrusive rocks                        147
marine sedimentary and volcanic rocks                146
basaltic volcanic rocks                              141
dioritic intrusive rocks                              96
coarse clastic sedimentary rocks                      79
quartz dioritic intrusive rocks                       72
ultramafic rocks                                      46
calc-alkaline volcanic rocks                          44
serpentinite ultramafic rocks                         42
high level quartz phyric, felsitic intrusive rocks    39
Name: rock_type, dtype: int64
```

1.0.3 Create labels for the two relevant rock_types (gran + serp/ultra)

[8]:
```python
# Label for 1 granodioritic rock_type
allrocks = list(data['rock_type'].unique())
rocks = [x for x in allrocks if 'grano' in x]

# Labels for both ultramafic rock types
rocks.extend([x for x in allrocks if 'ultramafic' in x])
print(rocks)
```

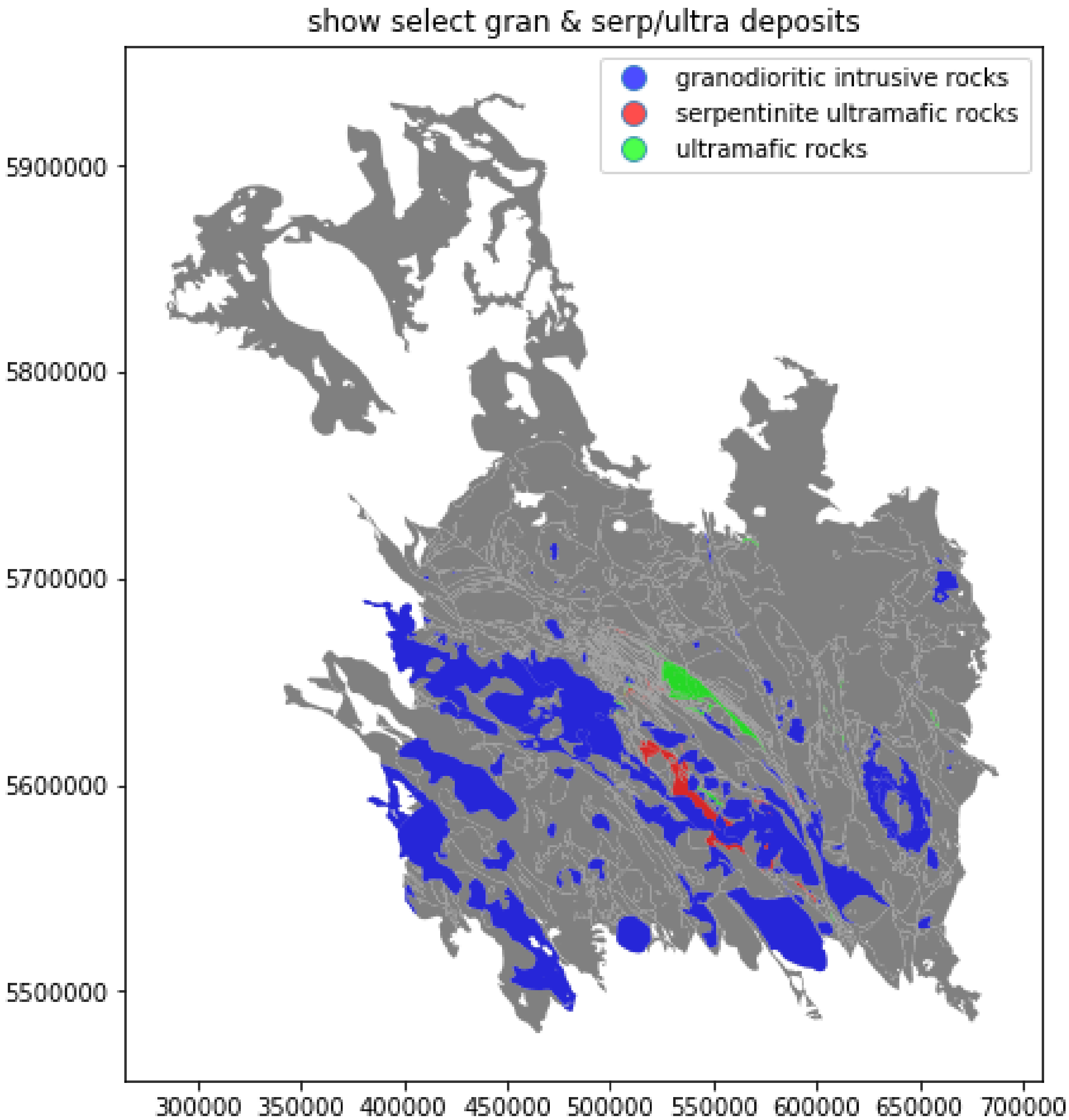
```
['granodioritic intrusive rocks', 'ultramafic rocks', 'serpentinite ultramafic
rocks']
```

[9]:
```
# Get indexes of 2 major rock type labels
gran_idx = data['rock_type'].apply(lambda x: x == rocks[0])
serp_idx = data['rock_type'].apply(lambda x: x in rocks[1:])
all_idx = (gran_idx | serp_idx)

# Verify that sum matches value_counts() above: 147 (gran), 88 (46 ultra + 42␣
↪serp)
print(f"Total gran deposits: {sum(gran_idx)}\nTotal serp deposits:␣
↪{sum(serp_idx)}")
```

```
Total gran deposits: 147
Total serp deposits: 88
```

1.0.4 Plot only deposits with relevant rocks

[10]:
```
fig, ax = plt.subplots(1, 1, figsize=(8,8))
data.plot(ax=ax, color='grey')
data[all_idx].plot(ax=ax, column="rock_type", legend=True, cmap='brg', alpha=0.
↪7)
ax.title.set_text('show select gran & serp/ultra deposits')
```



Although ultramafic and serpentinite labels are shown individually here, for the rest of the analysis, they will be combined (see instructions)

1.0.5 View coordinate reference system (crs) of data set

EPSG: 26910 (NAD83/UTM Zone 10N).

[11]:
```
print(data.crs)
```

```
{'init': 'epsg:26910'}
```

1.0.6 Calculate likelihood of finding a cobalt deposit for all regions

Find intersecting regions at a distance r from relevant deposits, then assign a likelihood of finding cobalt in that region. The instructions state that the likelihood should be maximum along where granodiorite and serpentite regions touch, then fall smoothly to zero at a distance $R = 10\ km$. A likelihood function that matches these goals can be described by the function

$$L(r) = 1 - r/R. \tag{1}$$

The max distance R determines when the likelihood should go to zero when $r \geq R$.

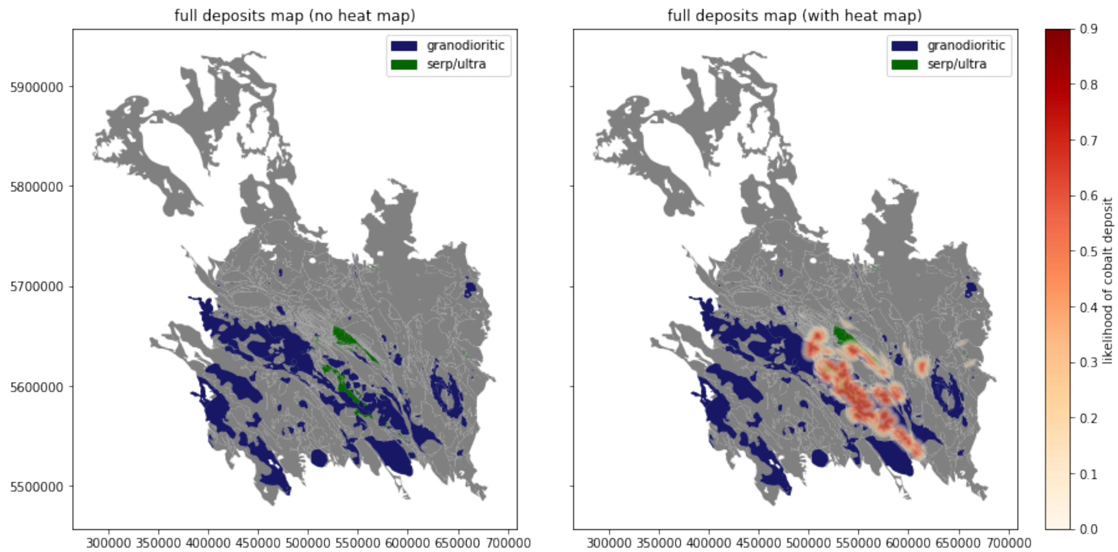
[12]:
```
# Set buffer resolution, buffer distances (in meters), & blank heatdf
buffer_num = 1+10
buffer_max = 10000
distances = np.linspace(0, buffer_max, buffer_num)
heatdf = gpd.GeoDataFrame()

# Loop over buffer regions to find intersections & set equal to same likelihood
for dist in distances:
    # Enlarge regions by a buffer distance
    granbuffer = data.loc[gran_idx, 'geometry'].buffer(dist)
    serpbuffer = data.loc[serp_idx, 'geometry'].buffer(dist)

    # Merge all buffers into MultiPolygon, then recast as GeoDataFrame
    grantotal = gpd.GeoDataFrame({'geometry':granbuffer.unary_union, 'rock':
↪'gran'})
    serptotal = gpd.GeoDataFrame({'geometry':serpbuffer.unary_union, 'rock':
↪'serp'})

    # Get all intersecting regions between gran/serp at this dist & assign␣
↪likelihood
    inter = gpd.overlay(grantotal, serptotal, how='intersection')
    inter['likelihood'] = 1 - dist/buffer_max

    # Store results in heatdf
    heatdf = heatdf.append(inter)

# Post-process heatdf to improve plotting & preserve crs
heatdf = heatdf.sort_values(by='likelihood', axis=0)
heatdf = heatdf.reset_index()
heatdf = heatdf.drop(labels=['index','rock_1','rock_2'], axis=1)
heatdf.crs = {'init': 'epsg:26910'}
```

[13]:
```
# Select a random point to show likelihood at in region (if turned on)
heatdf['coords'] = heatdf['geometry'].boundary.apply(lambda x: x.
↪representative_point().coords[:])
heatdf['coords'] = [random.choice(coords) for coords in heatdf['coords']]
```

1.0.7 Plot heat maps

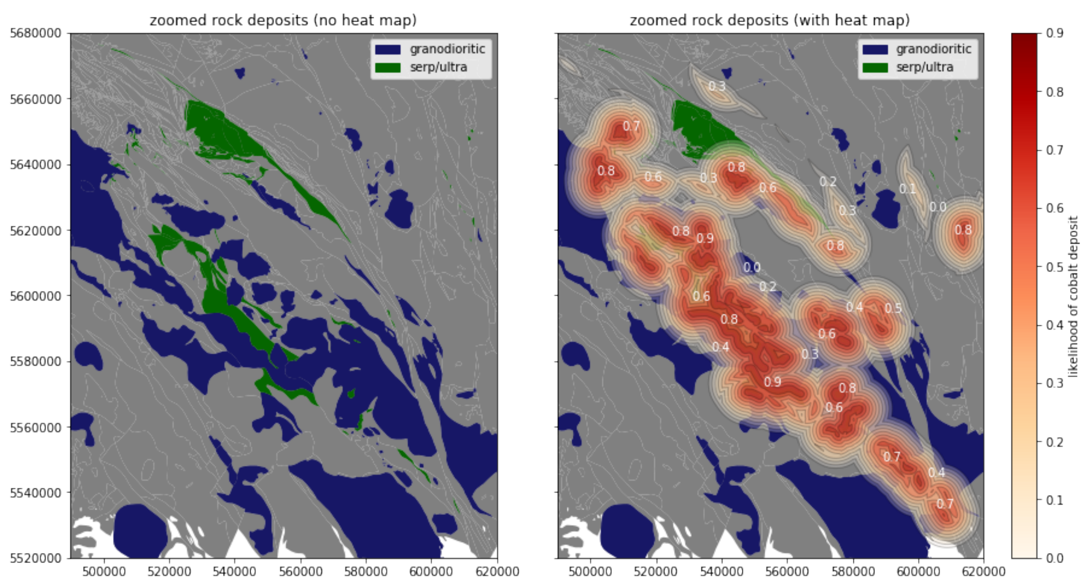
The full heat maps show that the majority of the interfacial regions between granodioritic and the serpentite/ultramafic rocks, hence, the likelihood, are in the lower portion of the plot. I will plot both the full heat map, as well as a zoomed in version to see more details.

[14]:
```python
# Create plot variables: zoom axis limits, RGB colors, legend handles/text
xlm = [490000, 620000]
ylm = [5520000, 5680000]
gran_color = (0.09, 0.09, 0.40)     # dark blue
serp_color = (0.02, 0.40, 0.00)     # dark green
gran_patch = mpatches.Patch(color=gran_color)
serp_patch = mpatches.Patch(color=serp_color)
handles = [gran_patch, serp_patch]
leg = ['granodioritic', 'serp/ultra']

# Full deposits plot (no colorbar)
fig, axs = plt.subplots(1, 2, sharex=True, sharey=True, figsize=(16,7.4))
data.plot(ax=axs[0], color='grey')
data[gran_idx].plot(ax=axs[0], color=gran_color)
data[serp_idx].plot(ax=axs[0], color=serp_color)
axs[0].title.set_text('full deposits map (no heat map)')
axs[0].legend(handles=handles, labels=leg, loc='upper right')

# Full deposits plot (with colorbar)
data.plot(ax=axs[1], color='grey')
data[gran_idx].plot(ax=axs[1], color=gran_color)
data[serp_idx].plot(ax=axs[1], color=serp_color)
heatdf.plot(ax=axs[1], column='likelihood', alpha=0.2, cmap='OrRd')
axs[1].set_title('full deposits map (with heat map)');
axs[1].legend(handles=handles, labels=leg, loc='upper right')
plt.subplots_adjust(wspace=0.0)

# Create custom colorbar
cim = plt.cm.ScalarMappable(cmap='OrRd',
                            norm=plt.Normalize(vmin=heatdf['likelihood'].min(),
                                               vmax=heatdf['likelihood'].max()))
cbar = plt.colorbar(cim, orientation='vertical')
cbar.set_label('likelihood of cobalt deposit')
plt.subplots_adjust(wspace=0.0)

# Save figure (keep commented until ready). figsizes (20,9), (16,7.4), (13,6)
# plt.savefig('full-small.png')
```



[15]:
```
# Zoomed deposits plot (no colorbar)
fig, axs = plt.subplots(1, 2, sharex=True, sharey=True, figsize=(16,8))
data.plot(ax=axs[0], color='grey')
data[gran_idx].plot(ax=axs[0], color=gran_color)
data[serp_idx].plot(ax=axs[0], color=serp_color)
axs[0].title.set_text('zoomed rock deposits (no heat map)')
axs[0].legend(handles=handles, labels=leg, loc='upper right')
axs[0].set_xlim(xlm);
axs[0].set_ylim(ylm);

# Zoomed deposits plot (with colorbar)
data.plot(ax=axs[1], color='grey')
data[gran_idx].plot(ax=axs[1], color=gran_color)
data[serp_idx].plot(ax=axs[1], color=serp_color)
heatdf.plot(ax=axs[1], column='likelihood', alpha=0.25, cmap='OrRd',␣
↪edgecolor='black')
axs[1].title.set_text('zoomed rock deposits (with heat map)')
axs[1].legend(handles=handles, labels=leg, loc='upper right', fancybox=True)
axs[1].set_xlim(xlm);
axs[1].set_ylim(ylm);

# Annotate plot with a subset of likelihoods spaced out from each other
ptlist = []
for idx, row in heatdf.iterrows():
    lhood = row['likelihood']
    cpt = shp.Point(row['coords'])
    check = sum([cpt.within(pt.buffer(7000)) for pt in ptlist])
```

```
    # Check if coord is not within specified buffer of any point in ptlist
    if check == 0:
        ptlist.append(cpt)
        label = round(lhood, 2)
        plt.annotate(s=label, xy=row['coords'], ha='center', color='white')

# Create custom colorbar
cim = plt.cm.ScalarMappable(cmap='OrRd',
                            norm=plt.Normalize(vmin=heatdf['likelihood'].min(),
                                               vmax=heatdf['likelihood'].max()))
cbar = plt.colorbar(cim, orientation='vertical')
cbar.set_label('likelihood of cobalt deposit')
plt.subplots_adjust(wspace=0.0)

# Save figure (comment until ready). figsizes large:(20,10), med:(16,8), small:
↪(13,6))
# plt.savefig('zoomed-med.png')
```



1.0.8 Create a function to query exact likelihoods at any coord

The map provides a good way to quickly see general areas of high likelihood, but displaying exact likelihoods at every location cluttered the map too much. So if an exact coordinate is provided, the function below will return the numerical likelihood at that location.

[16]:
```
def likelihood(coord):
    """ Given a coord tuple of (x, y), print the numerical likelihood of␣
 ↪finding cobalt
    coord (540000, 5660000) should yield l=0.3
    coord (600000, 5540000) should yield l=0.7
    coord (578000, 5560000) should yield l=0.9
    coord (520000, 5540000) should yield l=0.0"""

    cpt = shp.Point(coord)
    ptwithinlist = [cpt.within(shape) for shape in heatdf['geometry']]
    hits = sum(ptwithinlist)

    if hits == 0:
        maxhit = 0.0
    else:
        maxhit = max(heatdf['likelihood'][ptwithinlist])

    print(f"The likelihood of finding cobalt at {coord} = {round(maxhit, 2)}")
```

[17]:
```
# Test cases
likelihood((540000, 5660000))
likelihood((600000, 5540000))
likelihood((578000, 5560000))
likelihood((520000, 5540000))
```

```
The likelihood of finding cobalt at (540000, 5660000) = 0.3
The likelihood of finding cobalt at (600000, 5540000) = 0.7
The likelihood of finding cobalt at (578000, 5560000) = 0.9
The likelihood of finding cobalt at (520000, 5540000) = 0.0
```