

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Mike Middleton
<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

The following document is a pdf copy of live code notebooks that can be found here:

<https://github.com/MikeDairsie/Hillforts-Primer>



Cover images from top to bottom:

Dunsinane Hill, Perth and Kinross

Dunsinane Hill, Perth and Kinross

Tap O'Noth, Aberdeenshire

Barry Hill, Perth and Kinross

Eildon Hill, The Scottish Borders

All photos by Mike Middleton

Contents

Part 1	1
Introduction	1
Document Structure	2
Source Data	2
Mountains and Hills	23
Review Data	25
Bias	25
Missing Data	25
Rows and Features (Columns)	25
Name and Number	26
Admin Data	26
Location Data	51
Status Data	113
Part 2	126
Management Data	137
Landscape Data	165
Part 3	241
Boundary Data	253
Dating Data	265
Part 4	303
Investigations Data	315
Interior Data	316
Part 5	380
Entrance Data	401
Enclosing Data	445
Annex Data	668
Reference Data	673
Acknowledgements	676
Postscript	676
Appendix 1	677
Hypotheses Testing ≥ 21 Hectares line	699
Appendix 2	707
Classification Northwest	722

What follows is a PDF copy showing the code and outputs from a set of seven Jupyter notebooks. These contain links to the following resources which have not been converted to live links in the PDF copy. These links are:

Notebooks and all Hillforts Primer data and outputs can be found here:

<https://github.com/MikeDairsie/Hillforts-Primer>

Part 1: Name, Admin & Location Data

Colab Notebook- Live code (Must be logged into Google. Select Google Colaboratory, at the top of the screen, if page opens as raw code):

<https://colab.research.google.com/drive/1pcJkVos5ltkR1wMp7nudJBYLTcep9k9b?usp=sharing>

HTML: Read only

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_part_01.html

https://github.com/MikeDairsie/Hillforts-Primer/blob/main/hillforts-primer-html/hillforts_primer_part_01.html

Part 2: Management & Landscape

Colab Notebook: Live code

<https://colab.research.google.com/drive/1yRwVJAr6JljGVeMLE0SB7fTQ0pHdQPp?usp=sharing>

HTML: Read only

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_part_02.html

https://github.com/MikeDairsie/Hillforts-Primer/blob/main/hillforts-primer-html/hillforts_primer_part_02.html

Part 3: Boundary & Dating

Colab Notebook: Live code

https://colab.research.google.com/drive/1dMKByCmq33hjniGZImBjUYUj785_71CT?usp=sharing

HTML: Read only

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_part_03.html

https://github.com/MikeDairsie/Hillforts-Primer/blob/main/hillforts-primer-html/hillforts_primer_part_03.html

Part 4: Investigations & Interior

Colab Notebook: Live code

https://colab.research.google.com/drive/1rNXpURD4K5aglEFhve_lPHWLXOflej2I?usp=sharing

HTML: Read only

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_part_04.html

https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Fwww.dairsieonline.co.uk%2Fhillforts%2Fhillforts_primer_part_04.html

Part 5: Entrance, Enclosing & Annex

Colab Notebook: Live code

<https://colab.research.google.com/drive/1OTDROidFmUjr8bqZJld0tPyjWd-gdSMn?usp=sharing>

HTML: Read only

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_part_05.html

https://github.com/MikeDairsie/Hillforts-Primer/blob/main/hillforts-primer-html/hillforts_primer_part_05.html

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

Colab Notebook: Live code

https://colab.research.google.com/drive/1Fq4b-z95nEX-Xa2y2yLnVAAbjY_xoR8z?usp=drive_link

HTML: Read only

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_appendix_01.html

https://github.com/MikeDairsie/Hillforts-Primer/blob/main/hillforts-primer-html/hillforts_primer_part_appendix_01.html

Appendix 2: Classification Northwest

Colab Notebook: Live code

https://www.dairsieonline.co.uk/hillforts/hillforts_primer_appendix_02.html

<https://drive.google.com/file/d/1dHRox9nnWC3m8qpFu2X2mPFVfG0QntL/view?usp=sharing>

HTML: Read only

https://github.com/MikeDairsie/Hillforts-Primer/blob/main/hillforts-primer-html/hillforts_primer_part_appendix_02.html

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 1

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

- [Introduction](#)
- [Mountains & Hills](#)
- [Admin Data](#)
- [Location Data](#)
- [Status Data](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Introduction

This Hillforts Primer is a research tool that analyses, maps, plots, transforms and adds to the data published in The Atlas of Hillforts of Britain and Ireland (Lock & Ralston, 2017). The atlas contains 4147 records, with each record having 244 columns of associated information. The size and shape of the data can make it difficult to manipulate in traditional software so, the aim is to analyse and process the data programmatically, to facilitate interpretation, reuse and machine learning. The data will be:

- 1 **Analysed**: to explore how the data is structured, how complete the data is, what flaws there might be and how confident the user can be in using it.
- 2 **Plotted**: to explore the distribution and spread of the data and to visually identify where the plots highlight insights within the data.
- 3 **Mapped**: to show the extent, distribution and quantity of data.
- 4 **Transformed**: to transpose data into distributions more likely to highlight insights and to convert categorical data into numeric data so that machine learning tools can be applied.
- 5 **Added-to**: to create new columns of information by combining existing data.

Document Structure

The Hillforts Primer is produced using Google Collaboratory (more often called Google Colab). This is a free online tool that enables code to be run in-browser and shared. The data is processed using the Python language and no configuration is needed by the reader. By including the processing steps the reader can rerun or reuse the code, making the processing steps repeatable and transparent. The notebook format used by Google Colab enables hybrid documents to be created that contain both free text and live code. The text automatically creates a **Table of Contents** which can be used to quickly navigate the document (if not visible, click on the **three lines** in the **left menu**). The code output, from the last run, can be seen inline (within the document). To run the code again select **Runtime> Run All** from the top menu. See: [Introduction to Colab](#). Care should be taken to run the code in sequence. Notebooks allow code blocks to be run individually - outwith the expected run order - and this can cause errors. If this does happen, rerun the Runtime using Run All.

Some sections, of this document, contain code functions that will be used to reprocess or restructure the data. In an aim to make the document simple to read, some code sections have been minimised so they do not show by default. These sections can be identified by the black arrow to the left of the section heading. If the arrow points to the right, the section has been minimised. The section can be expanded by clicking on the arrow.

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>
Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer
Licence: <https://creativecommons.org/licenses/by-sa/4.0/>
Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>
Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>
Hillforts: Britain, Ireland and the Nearer Continent (Sample):
<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:
<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only,

download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [ ]: confirmed_only = False  
In [ ]: download_data = False  
In [ ]: save_images = False  
In [ ]: show_topography = False
```

Acknowledgments

I'd like to thank Dr Dave Cowley for his constant encouragement and support, Professor Jeremy Huggett for his assistance in developing abstracts for papers related to the Hillforts Primer and Emily Middleton for proofreading these notebooks.

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Mountains & Hills](#).

Python Modules and Code Setup

The Python imports enable the Hillforts Atlas data to be analysed and mapped within this document. The Python code can be run on demand, (see: [User Settings](#)). This means that as new research becomes available, the source for this document can be updated to a revised copy of the Atlas data and the impact of that research can be reviewed using the same code and graphic output. The Hillforts Atlas is a baseline and this document is a tool that can be used to assess the impact new research is making in this area.

```
In [ ]: import sys  
print(f'Python: {sys.version}')  
  
import sklearn  
print(f'Sci Kit-Learn: {sklearn.__version__}')  
  
import pandas as pd  
print(f'pandas: {pd.__version__}')  
  
import numpy as np  
print(f'numpy: {np.__version__}')  
  
%matplotlib inline  
import matplotlib  
print(f'matplotlib: {matplotlib.__version__}')  
import matplotlib.pyplot as plt  
import matplotlib.cm as cm  
import matplotlib.patches as mpatches  
import matplotlib.patches as patches  
from matplotlib.cbook import boxplot_stats  
from matplotlib.lines import Line2D  
import matplotlib.cm as cm  
  
import seaborn as sns  
print(f'seaborn: {sns.__version__}')  
sns.set(style="whitegrid")  
  
import scipy  
print(f'scipy: {scipy.__version__}')  
from scipy import stats  
from scipy.stats import gaussian_kde  
  
import os  
import collections
```

```

import math
import random
import PIL
import urllib
# A random seed is used to ensure that the random numbers created are the same
# for each run of this document.
random.seed(42)

from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive

```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]

Sci kit-Learn: 1.2.2

pandas: 1.5.3

numpy: 1.25.2

matplotlib: 3.7.1

seaborn: 0.13.1

scipy: 1.11.4

Ref: <https://www.python.org/>

Ref: <https://scikit-learn.org/stable/>

Ref: <https://pandas.pydata.org/docs/>

Ref: <https://numpy.org/doc/stable/>

Ref: <https://matplotlib.org/>

Ref: <https://seaborn.pydata.org/>

Ref: <https://docs.scipy.org/doc/scipy/index.html>

Ref: <https://pypi.org/project/python-slugify/>

Plot Figures and Maps functions

The following section contains code snippets and functions used in plotting and mapping the data. To facilitate reading this document this section, containing mostly code blocks, is minimised in Google Colaboratory.

```
In [ ]: def show_records(plt, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plt.annotate(str(len(plot_data))+' records', xy=(-1180000, 6420000), \
                xycoords='data', ha='left', color=text_colour)
```

```
In [ ]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
                      "hillforts-topo-northwest-monus.png",
                      "hillforts-topo-northeast.png",
                      "hillforts-topo-south.png",
                      "hillforts-topo-south-plus.png",
                      "hillforts-topo-ireland.png",
                      "hillforts-topo-ireland-north.png",
                      "hillforts-topo-ireland-south.png"]
    else:
        backgrounds = ["hillforts-outline-01.png",
                      "hillforts-outline-north.png",
                      "hillforts-outline-northwest-plus.png",
                      "hillforts-outline-northwest-monus.png",
                      "hillforts-outline-northeast.png",
                      "hillforts-outline-south.png",
                      "hillforts-outline-south-plus.png",
                      "hillforts-outline-ireland.png",
                      "hillforts-outline-ireland-north.png",
                      "hillforts-outline-ireland-south.png"]
    return backgrounds
```

```
In [ ]: def get_bounds():
    bounds = [[-1200000, 220000, 6400000, 8700000],
              [-1200000, 220000, 7000000, 8700000],
              [-1200000, -480000, 7000000, 8200000],
              [-900000, -480000, 7100000, 8200000],
              [-520000, 0, 7000000, 8700000],
              [-800000, 220000, 6400000, 7100000],
              [-1200000, 220000, 6400000, 7100000],
              [-1200000, -600000, 6650000, 7450000],
              [-1200000, -600000, 7050000, 7450000],
              [-1200000, -600000, 6650000, 7080000]]
    return bounds
```

4

```
In [ ]: def show_background(plt, ax, location=""):
backgrounds = get_backgrounds()
bounds = get_bounds()
# "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/
# main/hillforts-topo/"
folder = "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo/"

if location == "n":
    background = os.path.join(folder, backgrounds[1])
    bounds = bounds[1]
elif location == "nw+":
    background = os.path.join(folder, backgrounds[2])
    bounds = bounds[2]
elif location == "nw-":
    background = os.path.join(folder, backgrounds[3])
    bounds = bounds[3]
elif location == "ne":
    background = os.path.join(folder, backgrounds[4])
    bounds = bounds[4]
elif location == "s+":
    background = os.path.join(folder, backgrounds[5])
    bounds = bounds[5]
elif location == "i+":
    background = os.path.join(folder, backgrounds[6])
    bounds = bounds[6]
elif location == "in":
    background = os.path.join(folder, backgrounds[7])
    bounds = bounds[7]
elif location == "is":
    background = os.path.join(folder, backgrounds[8])
    bounds = bounds[8]
else:
    background = os.path.join(folder, backgrounds[0])
    bounds = bounds[0]

img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
ax.imshow(img, extent=bounds)
```

```
In [ ]: def get_counts(data):
data_counts = []
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    data_counts.append(count)
return data_counts
```

```
In [ ]: def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.01), \
                xycoords='figure fraction', horizontalalignment = 'right')
```

```
In [ ]: def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.035), \
                xycoords='figure fraction', horizontalalignment = 'right')
```

```
In [ ]: def plot_bar_chart(data, split_pos, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = data.columns
    x_data = [x.split("_")[split_pos:] for x in x_data]
    x_data_new = []
    for l in x_data :
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])
    y_data = get_counts(data)
    ax.bar(x_data_new, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, n_bins):
    new_data = data.copy()
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    data[x_label].plot(kind='hist', bins=n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def get_bins(data, bins_count):
    data_range = data.max() - data.min()
    print(bins_count)
    if bins_count != None:
        x_bins = [x for x in range(data.min(), data.max(), bins_count)]
        n_bins = len(x_bins)
    else:
        n_bins = int(data_range)
        if n_bins < 10:
            multi = 10
            while n_bins < 10:
                multi *= 10
                n_bins = int(data_range * multi)
        elif n_bins > 100:
            n_bins = int(data_range)/10
    return n_bins
```

```
In [ ]: def plot_histogram(data, x_label, title, bins_count = None):
    n_bins = get_bins(data, bins_count)
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    ax.set_ylabel('Count')
    plt.ticklabel_format(style='plain')
    plt.hist(data, bins=n_bins)
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [ ]: def plot_continuous(data, x_label, title):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    plt.plot(data, linewidth=4)
    plt.ticklabel_format(style='plain')
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [ ]: # box plot
def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.ticklabel_format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v", whis=[2.2, 97.8])
    else:
        sns.boxplot(data=data, orient="h", whis=[2.2, 97.8])
    save_fig(feature + " Range")
    plt.show()

bp = boxplot_stats(data, whis=[2.2, 97.8])
low = bp[0].get('whislo')
```

```

q1 = bp[0].get('q1')
median = bp[0].get('med')
q3 = bp[0].get('q3')
high = bp[0].get('whishi')

return [low, q1, median, q3, high]

```

In []:

```

def location_XY_plot():
    plt.title(label='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 8700000)
    add_annotation_l_xy(plt)

```

In []:

```

def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')

```

In []:

```

def plot_over_grey_numeric(merged_data, a_type, title, extra='', inner=False, \
                           fringe=False, oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

In []:

```

def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    fig, ax = plt.subplots(figsize=(9.47, 15.33))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(location_data['Location_X'], location_data['Location_Y'], \
               c='Silver')
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    show_records(plt, plot_data)
    plt.title(get_print_title('Boundary_Type: ' + boundary_type))
    save_fig('Boundary_Type_' + boundary_type)
    plt.show()

```

In []:

```

def plot_densiti_over_grey(data, data_type):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_densiti(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
               c=new_data['Density'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density')
    plt.title(get_print_title('Density - {data_type}'))
    save_fig('Density_{data_type}')
    plt.show()

```

In []:

```

def add_21Ha_line():
    x_values = [-367969, -344171, -263690, -194654, -130542, -119597, -162994, \
                -265052], #-304545]
    y_values = [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, \
               6663609], #6611780]
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, \
             label='21 Ha Line')
    add_to_legend = Line2D([0], [0], color='k', lw=15, alpha=0.25, \
                          label='21 Ha Line')

```

```

        label = '≥ 21 Ha Line')
    return add_to_legend

In [ ]: def add_21Ha_fringe():
    x_values = [-367969, -126771, 29679, -42657, -248650, -304545, -423647, \
                -584307, -367969]
    y_values = [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, \
                6752378, 7019842]
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    return add_to_legend

In [ ]: def add_oxford_swindon(oxford=False, swindon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_size = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_size, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swindon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_size, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches

In [ ]: def plot_over_grey(merged_data, a_type, yes_no, extra="", inner=False, \
                       fringe=False, oxford=False, swindon=False):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles= patches)
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()
    print(f' {round(((len(plot_data)/4147)*100), 2)}%')
    return plot_data

In [ ]: def plot_type_values(data, data_type, title):
    new_data = data.copy()
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
                c=new_data[data_type], cmap=cm.rainbow, s=25)
    plt.colorbar(label=data_type)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

In [ ]: def bespoke_plot(plt, title):
    add_annotation_plot(plt)
    plt.ticklabel_format(style='plain')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

In [ ]: def plot_line(point1, point2):
    x_values = [point1[0], point2[0]]
    y_values = [point1[1], point2[1]]
    plt.plot(x_values, y_values, 'r', ls='-', lw=3)

In [ ]: def add_cluster_split_lines(plt, ax, extra=None):
    x_min = -550000
    if extra == 'ireland':
        x_min = -1200000
    plt.vlines(x=[-500000], ymin=7070000, ymax=9000000, colors='r', ls='-', lw=3)

```

```

plt.hlines(y=[7070000], xmin=x_min, xmax=200000, colors='r', ls='-', lw=3)
ax.annotate("N/S split", color='k', xy=(50000, 7090000), xycoords='data')
ax.annotate("E/W split", color='k', xy=(-480000, 8660000), xycoords='data')
ax.annotate("Irish Sea split", color='k', xy=(-1150000, 7510000), \
           xycoords='data')
plot_line((-800000, 6400000), (-550000, 7070000))
plot_line((-550000, 7070000), (-666000, 7440000))
plot_line((-666000, 7440000), (-900000, 7500000))

```

```

In [ ]: def plot_england(england_and_data):
    plt.subplots(figsize=(7.0, 7.0))
    plt.scatter(england_and_data['Main_X'], england_and_data['Main_Y'])
    plt.xlim(0, 700000)
    plt.ylim(0, 700000)
    bespoke_plot(plt, 'England - Main_X/Main_Y')

```

```

In [ ]: def plot_wales(wales_data):
    plt.subplots(figsize=(7.5, 7.5))
    plt.scatter(wales_data['Main_X'], wales_data['Main_Y'])
    plt.xlim(150000, 400000)
    plt.ylim(150000, 400000)
    bespoke_plot(plt, 'Wales - Main_X/Main_Y')

```

```

In [ ]: def plot_scotland(scotland_data):
    plt.subplots(figsize=(10.0, 15.0))
    plt.scatter(scotland_data['Main_X'], scotland_data['Main_Y'])
    plt.xlim(0, 500000)
    plt.ylim(500000, 1250000)
    bespoke_plot(plt, 'Scotland_Main_X/Main_Y')

```

```

In [ ]: def plot_ni(ni_data):
    plt.subplots(figsize=(7.5, 7.5))
    plt.scatter(ni_data['Main_X'], ni_data['Main_Y'])
    plt.xlim(600000, 775000)
    plt.ylim(800000, 975000)
    bespoke_plot(plt, 'Northern_Ireland_Main_X/Main_Y')

```

```

In [ ]: def plot_roi(roi_data):
    plt.subplots(figsize=(7.0, 10.0))
    plt.scatter(roi_data['Main_X'], roi_data['Main_Y'])
    plt.xlim(400000, 750000)
    plt.ylim(500000, 1000000)
    bespoke_plot(plt, 'Republic_of_Ireland_Main_X/Main_Y')

```

```

In [ ]: def plot_iom(iom_data):
    plt.subplots(figsize=(10.0, 10.0))
    plt.scatter(iom_data['Main_X'], iom_data['Main_Y'])
    plt.xlim(210000, 260000)
    plt.ylim(460000, 510000)
    bespoke_plot(plt, 'Isle_of_Man_Main_X/Main_Y')

```

```

In [ ]: def plot_ireland(roi_and_ni_data):
    plt.subplots(figsize=(8.0, 10.0))
    plt.scatter(roi_and_ni_data['Main_X'], roi_and_ni_data['Main_Y'])
    plt.xlim(400000, 800000)
    plt.ylim(500000, 1000000)
    bespoke_plot(plt, \
                 'Republic_of_Ireland_and_Northern_Ireland_Combined - Main_X/Main_Y')

```

```

In [ ]: def plot_all_but_ireland(eng_wal_sco_man_data):
    plt.subplots(figsize=(7.0, 13))
    plt.scatter(eng_wal_sco_man_data['Main_X'], eng_wal_sco_man_data['Main_Y'])
    plt.xlim(0, 700000)
    plt.ylim(0, 1300000)
    bespoke_plot(plt, \
                 'England, Wales, Scotland and Man Combined - Main_X/Main_Y')

```

```

In [ ]: def plot_main_xy(data):
    plt.subplots(figsize=(8.0, 13))
    plt.scatter(data['Main_X'], data['Main_Y'])
    plt.xlim(0, 800000)
    plt.ylim(0, 1300000)
    bespoke_plot(plt, 'Main_X/Main_Y')

```

```

In [ ]: def plot_location_xy(data):
    scale_xy = 0.66
    fig, ax = plt.subplots(figsize=(14.2 * scale_xy, 23.0 * scale_xy))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(data['Location_X'], data['Location_Y'])
    title = 'Location_X/Location_Y'
    plt.title(get_print_title(title))

```

```
    save_fig(title)  
plt.show()
```

```
In [ ]: def plot_latLong(data):
    plt.subplots(figsize=(8.8, 7.66))
    plt.scatter(data['Location_Longitude'], data['Location_Latitude'])
    plt.xlim(-11, 2.2)
    plt.ylim(49.5, 61.0)
    plt.title('Location_Longitude/Location_Latitude')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def density_transformed(transformed_location_numeric_data_short):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], \
               transformed_location_numeric_data_short['Location_Y'], \
               c=transformed_location_numeric_data_short['Density_trans'], \
               cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density Transformed')
    title = 'Density Transformed'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def get_rectangles(x_data, y_data, w1, w2):
    x_lo, x_q1, x_median, x_q3, x_hi = x_data[0], x_data[1], x_data[2], \
    x_data[3], x_data[4]
    y_lo, y_q1, y_median, y_q3, y_hi = y_data[0], y_data[1], y_data[2], \
    y_data[3], y_data[4]

    rect = patches.Rectangle((x_lo, y_lo), x_hi-x_lo, y_hi-y_lo, linewidth=w2, \
                           edgecolor='k', facecolor='none')
    rect2 = patches.Rectangle((x_q1, y_q1), x_q3-x_q1, y_q3-y_q1, linewidth=w1, \
                           edgecolor='r', facecolor='none')

    return rect, rect2, [y_q1, x_median, y_q3], [x_q1, y_median, x_q3]
```

```
In [ ]: def plot_densisty_boxplot(transformed_location_numeric_data_short):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], \
               transformed_location_numeric_data_short['Location_Y'], \
               c=transformed_location_numeric_data_short['Density_trans'], \
               cmap=cm.rainbow, s=25)
    rect, rect2, vline, xline = get_rectangles(location_X_data, \
                                                location_Y_data, 3, 1.5)
    ax.add_patch(rect)
    ax.add_patch(rect2)
    plt.vlines(x=[vline[1]], ymi=vline[0], ymax=vline[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline[1]], xmi=xline[0], xmax=xline[2], colors='purple', \
               ls='--', lw=1.5)
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of Boxplot (All Data)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```

plt.plot(-177825, 7447736, 'ko')
ax.annotate('EN0580: Howick Hill Camp', xy=(-177825, 7447736), \
            xycoords='data', ha='left', xytext=(15, -35), \
            textcoords='offset points', \
            arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                           connectionstyle="arc3,rad=-0.2"))
plt.plot(-487909, 7558812, 'ko')
ax.annotate('SC1458: Quinloch Muir', xy=(-487909, 7558812), \
            xycoords='data', ha='left', xytext=(-275, -45), \
            textcoords='offset points', \
            arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                           connectionstyle="arc3,rad=-0.2"))
plt.plot(-276998, 7682350, 'ko')
ax.annotate('SC3102: Red Head', xy=(-276998, 7682350), xycoords='data', \
            ha='left', xytext=(35, 35), textcoords='offset points', \
            arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                           connectionstyle="arc3,rad=-0.2"))
add_annotation_plot=plt
plt.colorbar(label='Density Transformed')
title = 'Density Transformed (North)'
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_northern_density_transformed(show_eildon, north_top5):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticker_label_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(7000000, 8700000)
    plt.scatter(north_top5['Location_X'], north_top5['Location_Y'], \
                c=north_top5['Density_trans'], cmap=cm.rainbow, s=25)
    if show_eildon:
        plt.plot(-301491, 7476601, 'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), \
                    xycoords='data', ha='left', xytext=(35, 35), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                   connectionstyle="arc3,rad=-0.2"))
    add_annotation_plot=plt
    plt.colorbar(label='Density Transformed - top 5%')
    title = 'Density Transformed - top 5% (North)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_density_transformed_north_capped(show_eildon, north_clip):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticker_label_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(7000000, 8700000)
    plt.scatter(north_clip['Location_X'], north_clip['Location_Y'], \
                c=north_clip['Density_trans'], cmap=cm.rainbow, s=25)
    if show_eildon:
        plt.plot(-301491, 7476601, 'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), \
                    xycoords='data', ha='left', xytext=(35, 35), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                   connectionstyle="arc3,rad=-0.2"))
        plt.plot(-609918, 7575512, 'ko')
        ax.annotate('SC2466: Dunadd', xy=(-609918, 7575512), xycoords='data', \
                    ha='left', xytext=(-250, 35), textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                   connectionstyle="arc3,rad=-0.2"))
        plt.plot(-487909, 7558812, 'ko')
        ax.annotate('SC1458: Quinloch Muir', xy=(-487909, 7558812), \
                    xycoords='data', ha='left', xytext=(-375, -115), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                   connectionstyle="arc3,rad=-0.2"))
        plt.plot(-449967, 7321023, 'ko')
        ax.annotate('SC0244: Drummore Castle', xy=(-449967, 7321023), \
                    xycoords='data', ha='left', xytext=(-50, -85), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                   connectionstyle="arc3,rad=-0.2"))
        plt.plot(-276998, 7682350, 'ko')
        ax.annotate('SC3102: Red Head', xy=(-276998, 7682350), xycoords='data', \
                    ha='left', xytext=(35, 35), textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                   connectionstyle="arc3,rad=-0.2"))
        plt.plot(-177825, 7447736, 'ko')
        ax.annotate('EN0580: Howick Hill Camp', xy=(-177825, 7447736), \

```

```

    xycoords='data', ha='left', xytext=(15, -35), \
    textcoords='offset points', \
    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                    connectionstyle="arc3,rad=-0.2"))
add_annotation_plot(pt)
pt.colorbar(label='Density Transformed - Capped')
title = 'Density Transformed - Capped (North)'
pt.title(get_print_title(title))
save_fig(title)
pt.show()

```

```

In [ ]: def plot_density_transformed_north_boxplot(north, location_X_north_data, \
                                                    location_Y_north_data):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(pt, ax, 'n')
    pt.ticker_label_format(style='plain')
    pt.xlim(-1200000, 220000)
    pt.ylim(7000000, 8700000)
    pt.scatter(north['Location_X'], north['Location_Y'], \
               c=north['Density_trans'], cmap=cm.rainbow, s=25)
    rect, rect2, vline, xline = get_rectangles(location_X_north_data, \
                                                location_Y_north_data, 3, 1.5)
    ax.add_patch(rect)
    ax.add_patch(rect2)
    pt.vlines(x=[vline[1]], ymin=vline[0], ymax=vline[2], colors='purple', \
               ls='--', lw=1.5)
    pt.hlines(y=[xline[1]], xmin=xline[0], xmax=xline[2], colors='purple', \
               ls='--', lw=1.5)
    show_eildon = False
    if show_eildon:
        pt.plot(-301491, 7476601, 'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), \
                    xycoords='data', ha='left', xytext=(35, 35), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    add_annotation_plot(pt)
    pt.colorbar(label='Density Transformed')
    title = 'Density Transformed overlaid by the extent of Boxplot (N)'
    pt.title(get_print_title(title))
    save_fig(title)
    pt.show()

```

```

In [ ]: def plot_north_west_density(show_eildon, north_west, location_Y_north_w_data):
    fig, ax = plt.subplots(figsize=((4.75 * 1.5)+2.0, (7.92 * 1.5)))
    show_background(pt, ax, 'nw+')
    pt.ticker_label_format(style='plain')
    pt.xlim(-1200000, -480000)
    pt.ylim(7000000, 8200000)
    pt.scatter(north_west['Location_X'], north_west['Location_Y'], \
               c=north_west['Density_trans'], cmap=cm.rainbow, s=25)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_north_w_data, \
                                                   location_Y_north_w_data, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    pt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    pt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    if show_eildon:
        pt.plot(-609918, 7575512, 'ko')
        ax.annotate('SC2466: Dunadd', xy=(-609918, 7575512), xycoords='data', \
                    ha='left', xytext=(-240, -10), textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    add_annotation_plot(pt)
    pt.colorbar(label='Density Transformed')
    title = 'Density Transformed (Northwest)'
    pt.title(get_print_title(title))
    save_fig(title)
    pt.show()

```

```

In [ ]: def plot_north_east_density(show_eildon, north_east, location_Y_north_e_data):
    fig, ax = plt.subplots(figsize=((3.43*1.5)+2.0, (11.22*1.5)))
    show_background(pt, ax, 'ne')
    pt.ticker_label_format(style='plain')
    pt.xlim(-520000, 0)
    pt.ylim(7000000, 8700000)
    pt.scatter(north_east['Location_X'], north_east['Location_Y'], \
               c=north_east['Density_trans'], cmap=cm.rainbow, s=25)
    rect1, rect2, vline1, xline1 = get_rectangles(location_X_north_e_data, \
                                                   location_Y_north_e_data, \
                                                   3, 1.5)

```

```

ax.add_patch(rect1)
ax.add_patch(rect2)
plt.vlines(x=[vline1[1]], ymin=vline1[0], ymax=vline1[2], colors='purple', \
           ls='--', lw=1.5)
plt.hlines(y=[xline1[1]], xmin=xline1[0], xmax=xline1[2], colors='purple', \
           ls='--', lw=1.5)
if show_eldon:
    plt.plot(-301491, 7476601, 'ko') # SC3327: Eildon Hill North
    ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), \
                xycoords='data', ha='left', xytext=(0, 85), \
                textcoords='offset points', \
                arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                connectionstyle="arc3,rad=-0.2"))
    plt.plot(-247628, 7490081, 'ko')
    ax.annotate('EN4137: Cornhill', xy=(-247628, 7490081), xycoords='data', \
                ha='left', xytext=(20, 15), textcoords='offset points', \
                arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                connectionstyle="arc3,rad=-0.2"))
add_annotation_plot(plt)
plt.colorbar(label='Density Transformed')
title = 'Density Transformed (Northeast)'
plt.title(get_pprint_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_northern_density_boxplots(show_eldon, north, \
                                         location_X_cluster_north_west, \
                                         location_Y_cluster_north_west):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(7000000, 8700000)
    plt.scatter(north['Location_X'], north['Location_Y'], \
                c=north['Density_trans'], cmap=cm.rainbow, s=25)
    plt.axvline(x=-500000, color='r', linestyle='-', lw=3)
    ax.annotate("E/W split", color='k', xy=(-480000, 7010000), xycoords='data')
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_north_west, \
                                                   location_Y_cluster_north_west, \
                                                   3, 1.5)
    #rect3, rect4, vline2, xline2 = get_rectangles(location_X_north_w_data, \
    #                                               location_Y_north_w_data, 3, 1.5)
    rect5, rect6, vline3, xline3 = get_rectangles(location_X_north_e_data, \
                                                   location_Y_north_e_data, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect5)
    ax.add_patch(rect6)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', \
               ls='--', lw=1.5)
    show_eldon = False
    if show_eldon:
        plt.plot(-301491, 7476601, 'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), \
                    xycoords='data', ha='left', xytext=(35, 35), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed (Northeast & Norht West)'
    plt.title(get_pprint_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_clusters(cluster_north_ireland, cluster_south_ireland, cluster_south, \
                        cluster_north_east, cluster_north_west):
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(cluster_north_ireland['Location_X'], \
                cluster_north_ireland['Location_Y'], c='Red')
    plt.scatter(cluster_south_ireland['Location_X'], \
                cluster_south_ireland['Location_Y'], c='magenta')
    plt.scatter(cluster_south['Location_X'], cluster_south['Location_Y'], \
                c='blue')
    plt.scatter(cluster_north_east['Location_X'], \
                cluster_north_east['Location_Y'], c='medi umorchi d')

```

```

plt.scatter(cluster_north_west['Location_X'], \
            cluster_north_west['Location_Y'], c='yellowgreen')
add_cluster_split_lines(plt, ax, 'ireland')
title = 'Location Cluster Data Packages'
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_north_west_dnesity_minus(show_eildon, cluster_north_west, \
                                         location_X_cluster_north_west, \
                                         location_Y_cluster_north_west):
    fig, ax = plt.subplots(figsize=((2.772 * 1.75)+2.0, (7.26 * 1.75)))
    show_background(plt, ax, 'nw-')
    plt.ticker._format(style='plain')
    plt.xlim(-900000, -480000)
    plt.ylim(7100000, 8200000)
    plt.scatter(cluster_north_west['Location_X'], \
                cluster_north_west['Location_Y'], \
                c=cluster_north_west['Density_trans'], cmap=cm.rainbow, s=25)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_north_west, \
                                                   location_Y_cluster_north_west, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.plot(-609918, 7575512, 'ko')
    if show_eildon:
        ax.annotate('SC2466: Dunadd', xy=(-609918, 7575512), xycoords='data', \
                    ha='left', xytext=(-110, -135), textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Trans & Boxplot (NW minus Ireland)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_southern_density(show_gaer_fach, south):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticker._format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 7100000)
    plt.scatter(south['Location_X'], south['Location_Y'], c=south['Density'], \
                cmap=cm.rainbow, s=25)
    if show_gaer_fach:
        plt.plot(-383564, 6803609, 'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564, 6803609), \
                    xycoords='data', ha='left', xytext=(-290, -65), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_south_top_5(show_gaer_fach, south_top5):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticker._format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 7100000)
    plt.scatter(south_top5['Location_X'], south_top5['Location_Y'], \
                c=south_top5['Density'], cmap=cm.rainbow, s=25)
    if show_gaer_fach:
        plt.plot(-383564, 6803609, 'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564, 6803609), \
                    xycoords='data', ha='left', xytext=(-290, -65), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density - top 5%')
    add_annotation_plot(plt)
    title = 'Density - top 5% (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```
In [ ]: def plot_south_density_transformed(show_gaer_fach, south):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticker._format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 7100000)
    plt.scatter(south['Location_X'], south['Location_Y'], \
                c=south['Density_trans'], cmap=cm.rainbow, s=25)
    if show_gaer_fach:
        plt.plot(-383564, 6803609, 'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564, 6803609), \
                    xycoords='data', ha='left', xytext=(-290, -65), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
        plt.plot(-267643, 6699776, 'ko')
        ax.annotate('EN4166: Freezing Hill', xy=(-267643, 6699776), \
                    xycoords='data', ha='left', xytext=(-200, -55), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def plot_southern_density_boxplot(show_gaer_fach, south, location_X_south_data, \
                                         location_Y_south_data):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticker._format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 7100000)
    plt.scatter(south['Location_X'], south['Location_Y'], \
                c=south['Density_trans'], cmap=cm.rainbow, s=25)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_south_data, \
                                                   location_Y_south_data, 3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    if show_gaer_fach:
        plt.plot(-383564, 6803609, 'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564, 6803609), \
                    xycoords='data', ha='left', xytext=(-282, -100), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed and Boxplot (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def plot_south_density_transformed_boxplot(show_gaer_fach, cluster_south, \
                                                location_X_cluster_south, \
                                                location_Y_cluster_south):
    fig, ax = plt.subplots(figsize=((6.73*1.5)+2.0, (4.62*1.5)))
    show_background(plt, ax, 's')
    plt.ticker._format(style='plain')
    plt.xlim(-800000, 220000)
    plt.ylim(6400000, 7100000)
    plt.scatter(cluster_south['Location_X'], cluster_south['Location_Y'], \
                c=cluster_south['Density_trans'], cmap=cm.rainbow, s=25)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, \
                                                   location_Y_cluster_south, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    if show_gaer_fach:
        plt.plot(-370211, 6775546, 'ko')
        ax.annotate('WA1336: Nant Tarthwynni East', xy=(-370211, 6775546), \
                    xycoords='data', ha='left', xytext=(-235, -75), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
```

```

    connectionstyle="arc3, rad=-0.2"))
plt.plot(-510962, 6777200, 'ko')
ax.annotate('WA2240: Hafod Camp', xy=(-510962, 6777200), xycoords='data', \
            ha='left', xytext=(-155, 55), textcoords='offset points', \
            arrowprops=dict(arrowstyle="->", color='k', lw=1, \
            connectionstyle="arc3, rad=-0.2"))
plt.colorbar(label='Density Transformed')
add_annotation_plot(plt)
title = 'Density Transformed and Boxplot (South minus Ireland)'
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_ns_boxplots(transformed_location_numeric_data_short, \
                           location_X_south_data, location_Y_south_data, \
                           location_X_north_data, location_Y_north_data):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], \
                transformed_location_numeric_data_short['Location_Y'], \
                c=transformed_location_numeric_data_short['Density_trans'], \
                cmap=cm.rainbow, s=25)
    plt.axhline(y=7070000, color='r', linestyle='-', lw=3)
    ax.annotate("N/S split", color='k', xy=(50000, 7090000), xycoords='data')
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_south_data, \
                                                   location_Y_south_data, 3, 1.5)
    rect5, rect6, vline3, xline3 = get_rectangles(location_X_north_data, \
                                                   location_Y_north_data, 3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect5)
    ax.add_patch(rect6)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of Boxplots: (N & S)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_ne_nw_s_boxplots(transformed_location_numeric_data_short, \
                               location_X_cluster_north_west, \
                               location_Y_cluster_north_west, location_X_north_e_data, \
                               location_Y_north_e_data, location_X_cluster_south, \
                               location_Y_cluster_south):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], \
                transformed_location_numeric_data_short['Location_Y'], \
                c=transformed_location_numeric_data_short['Density_trans'], \
                cmap=cm.rainbow, s=25)
    add_cluster_split_lines(plt, ax)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, \
                                                   location_Y_cluster_south, 3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    rect7, rect8, vline2, xline2 = get_rectangles(location_X_cluster_north_west, \
                                                   location_Y_cluster_north_west, 3, 1.5)
    rect9, rect10, vline3, xline3 = get_rectangles(location_X_north_e_data, \
                                                   location_Y_north_e_data, 3, 1.5)
    ax.add_patch(rect7)
    ax.add_patch(rect8)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect9)
    ax.add_patch(rect10)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', \
               ls='--', lw=1.5)

```

```

plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', \
           ls='--', lw=1.5)
plt.colorbar(label='Density Transformed')
title = 'Density overlayed by the extent of Boxplots: NE, NW & South'
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_density_irland(show_dooncarton, ireland_density_data):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'i')
    plt.scatter(ireland_density_data['Location_X'], \
                ireland_density_data['Location_Y'], \
                c=ireland_density_data['Density'], cmap=cm.rainbow, s=25)
    plt.xlim(-1200000, -600000)
    plt.ylim(6650000, 7450000)
    if show_dooncarton:
        plt.plot(-1095290, 7223461, 'ko')
        ax.annotate('IR1821 Dooncarton', xy=(-1095290, 7223461), \
                    xycoords='data', ha='left', xytext=(-150, 20), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
        plt.plot(-1077483, 6894257, 'ko')
        ax.annotate('IR1279: Doon West', xy=(-1077483, 6894257), \
                    xycoords='data', ha='left', xytext=(-175, -35), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.ticker_label_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_density_irland_top_5(show_dooncarton, ireland_density_top5):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'i')
    plt.scatter(ireland_density_top5['Location_X'], \
                ireland_density_top5['Location_Y'], \
                c=ireland_density_top5['Density'], cmap=cm.rainbow, s=25)
    plt.xlim(-1200000, -600000)
    plt.ylim(6650000, 7450000)
    if show_dooncarton:
        plt.plot(-1095290, 7223461, 'ko')
        ax.annotate('IR1821 Dooncarton', xy=(-1095290, 7223461), \
                    xycoords='data', ha='left', xytext=(-150, 20), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
        plt.plot(-1077483, 6894257, 'ko')
        ax.annotate('IR1279: Doon West', xy=(-1077483, 6894257), \
                    xycoords='data', ha='left', xytext=(-175, -35), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.ticker_label_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland - top 5%'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_density_north_irland(show_dooncarton, north_cluster):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 5.0))
    show_background(plt, ax, 'in')
    plt.scatter(north_cluster['Location_X'], north_cluster['Location_Y'], \
                c=north_cluster['Density'], cmap=cm.rainbow, s=25)
    plt.xlim(-1200000, -600000)
    plt.ylim(7050000, 7450000)
    if show_dooncarton:
        plt.plot(-1095290, 7223461, 'ko')
        ax.annotate('IR1821 Dooncarton', xy=(-1095290, 7223461), \
                    xycoords='data', ha='left', xytext=(-150, 30), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.ticker_label_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland - North'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

plt.title(get_pprint_title(title))
save_fig(title)
plt.show()

In [ ]: def plot_densify_south_ireland(show_dooncarton, south_cluster):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 5.0))
    show_background(plt, ax, 'is')
    plt.scatter(south_cluster['Location_X'], south_cluster['Location_Y'], \
                c=south_cluster['Density'], cmap=cm.rainbow, s=25)
    plt.xlim(-1200000, -600000)
    plt.ylim(6650000, 7080000)
    if show_dooncarton:
        plt.plot(-1077483, 6894257, 'ko')
        ax.annotate('IR1279: Doon West', xy=(-1077483, 6894257), \
                    xycoords='data', ha='left', xytext=(-175, -15), \
                    textcoords='offset points', \
                    arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland - South'
    plt.title(get_pprint_title(title))
    save_fig(title)
    plt.show()

In [ ]: def plot_ireland_one_boxplot(ireland_densify_data, location_X_ireland_data, \
                                     location_Y_ireland_data):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'l')
    plt.scatter(ireland_densify_data['Location_X'], \
                ireland_densify_data['Location_Y'], \
                c=ireland_densify_data['Density'], cmap=cm.rainbow, s=25)
    rect1, rect2, vline, xline = get_rectangles(location_X_ireland_data, \
                                                location_Y_ireland_data, 3, 1.5)
    ax.add_patch(rect1)
    ax.add_patch(rect2)
    plt.vlines(x=[vline[1]], ymin=vline[0], ymax=vline[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline[1]], xmin=xline[0], xmax=xline[2], colors='purple', \
               ls='--', lw=1.5)
    plt.xlim(-1200000, -600000)
    plt.ylim(6650000, 7450000)
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density overlaid by the extent of Boxplot (Ireland)'
    plt.title(get_pprint_title(title))
    save_fig(title)
    plt.show()

In [ ]: def plot_ireland_boxplots(ireland_densify_data, location_X_south_data_ireland, \
                                 location_Y_south_data_ireland, \
                                 location_X_north_data_ireland, \
                                 location_Y_north_data_ireland):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'l')
    plt.scatter(ireland_densify_data['Location_X'], \
                ireland_densify_data['Location_Y'], \
                c=ireland_densify_data['Density'], cmap=cm.rainbow, s=25)
    plt.axhline(y=7060000, color='r', linestyle='-', lw=3)
    ax.annotate("N/S split", color='k', xy=(-680000, 7040000), xycoords='data')
    rect3, rect4, vline2, xline2 = \
        get_rectangles(location_X_south_data_ireland, location_Y_south_data_ireland, \
                      3, 1.5)
    rect5, rect6, vline3, xline3 = \
        get_rectangles(location_X_north_data_ireland, location_Y_north_data_ireland, \
                      3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect5)
    ax.add_patch(rect6)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.xlim(-1200000, -600000)
    plt.ylim(6650000, 7450000)
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')

```

```

add_annotation_plot(plt)
title = 'Density overlayed by the extent of Boxplots - Ireland (N & S)'
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_five_clusters(transformed_location_numeric_data_short, \
                           location_X_cluster_north_west, \
                           location_Y_cluster_north_west, location_X_north_e_data, \
                           location_Y_north_e_data, location_X_cluster_south, \
                           location_Y_cluster_south, location_X_south_data_irland, \
                           location_Y_south_data_irland, location_X_north_data_irland, \
                           location_Y_north_data_irland):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], \
                transformed_location_numeric_data_short['Location_Y'], \
                c=transformed_location_numeric_data_short['Density_trans'], \
                cmap=cm.rainbow, s=25)
    plt.axhline(y=7070000, color='r', linestyle='-', lw=3)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, \
                                                   location_Y_cluster_south, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    add_cluster_split_lines(plt, ax, 'ireland')
    rect7, rect8, vline2, xline2 = get_rectangles(location_X_cluster_north_west, \
                                                   location_Y_cluster_north_west, \
                                                   3, 1.5)
    rect9, rect10, vline3, xline3 = get_rectangles(location_X_north_e_data, \
                                                   location_Y_north_e_data, \
                                                   3, 1.5)
    ax.add_patch(rect7)
    ax.add_patch(rect8)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect9)
    ax.add_patch(rect10)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', \
               ls='--', lw=1.5)
    rect11, rect12, vline4, xline4 = get_rectangles(location_X_south_data_irland, \
                                                   location_Y_south_data_irland, \
                                                   3, 1.5)
    rect13, rect14, vline5, xline5 = get_rectangles(location_X_north_data_irland, \
                                                   location_Y_north_data_irland, \
                                                   3, 1.5)
    ax.add_patch(rect11)
    ax.add_patch(rect12)
    plt.vlines(x=[vline4[1]], ymin=vline4[0], ymax=vline4[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline4[1]], xmin=xline4[0], xmax=xline4[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect13)
    ax.add_patch(rect14)
    plt.vlines(x=[vline5[1]], ymin=vline5[0], ymax=vline5[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline5[1]], xmin=xline5[0], xmax=xline5[2], colors='purple', \
               ls='--', lw=1.5)
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of the five main cluster Boxplots'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_adjusted_density(irland_density_data, cluster_south, cluster_north_west, \
                               north_east, location_X_cluster_south, \
                               location_Y_cluster_south, location_X_cluster_north_west, \
                               location_Y_cluster_north_west, location_X_north_e_data, \
                               location_Y_north_e_data, location_X_south_data_irland, \
                               location_Y_south_data_irland, \
                               location_X_north_data_irland, \
                               location_Y_north_data_irland):
    fig, ax = plt.subplots(figsize=(14.2 * 0.66), 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(irland_density_data['Location_X'], 19

```

```

    i rel and_densi ty_data[' Location_Y' ], \
    c=i rel and_densi ty_data[' Densi ty' ], cmap=cm.raibow, s=25)
plt.scatter(cluster_south[' Location_X' ], cluster_south[' Location_Y' ], \
            c=cluster_south[' Densi ty_trans' ], cmap=cm.raibow, s=25)
plt.scatter(cluster_north_west[' Location_X' ], \
            cluster_north_west[' Location_Y' ], \
            c=cluster_north_west[' Densi ty_trans' ], cmap=cm.raibow, s=25)
plt.scatter(north_east[' Location_X' ], north_east[' Location_Y' ], \
            c=north_east[' Densi ty_trans' ], cmap=cm.raibow, s=25)
plt.axhline(y=707000, color='r', linestyle='-', lw=3)
if show_boxplots:
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, \
                                                   location_Y_cluster_south, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymi=n=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmi=n=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
add_cluster_split_lines(plt, ax, 'ireland')
if show_boxplots:
    rect7, rect8, vline2, xline2 = get_rectangles(location_X_cluster_north_west, \
                                                   location_Y_cluster_north_west, \
                                                   3, 1.5)
    rect9, rect10, vline3, xline3 = get_rectangles(location_X_north_e_data, \
                                                   location_Y_north_e_data, \
                                                   3, 1.5)
    ax.add_patch(rect7)
    ax.add_patch(rect8)
    plt.vlines(x=[vline2[1]], ymi=n=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmi=n=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect9)
    ax.add_patch(rect10)
    plt.vlines(x=[vline3[1]], ymi=n=vline3[0], ymax=vline3[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline3[1]], xmi=n=xline3[0], xmax=xline3[2], colors='purple', \
               ls='--', lw=1.5)
    rect11, rect12, vline4, xline4 = get_rectangles(location_X_south_data_ireland, \
                                                   location_Y_south_data_ireland, \
                                                   3, 1.5)
    rect13, rect14, vline5, xline5 = get_rectangles(location_X_north_data_ireland, \
                                                   location_Y_north_data_ireland, \
                                                   3, 1.5)
    ax.add_patch(rect11)
    ax.add_patch(rect12)
    plt.vlines(x=[vline4[1]], ymi=n=vline4[0], ymax=vline4[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline4[1]], xmi=n=xline4[0], xmax=xline4[2], colors='purple', \
               ls='--', lw=1.5)
    ax.add_patch(rect13)
    ax.add_patch(rect14)
    plt.vlines(x=[vline5[1]], ymi=n=vline5[0], ymax=vline5[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline5[1]], xmi=n=xline5[0], xmax=xline5[2], colors='purple', \
               ls='--', lw=1.5)
title = 'The five main density clusters (Density adjusted by region)'
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data. To facilitate reading this document this section, containing mostly code blocks, is minimised.

```

In [ ]: def test_numeric(data):
    temp_data = data.copy()
    columns = data.columns
    out_cols = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
    feat, ent, num, non, nul = [], [], [], [], []
    for col in columns:
        if temp_data[col].dtype == 'object':
            feat.append(col)
            temp_data[col + '_num'] = temp_data[col].str.isnumeric()
            entries = temp_data[col].notnull().sum()
            true_count = temp_data[col + '_num'][temp_data[col + '_num'] == \
                                                 True].sum()
            null_count = temp_data[col].isna().sum()
            ent.append(entries)
            num.append(true_count)

```

```

        non.append(entries_true_count)
        nul.append(null_count)
    else:
        print(f'{col} {temp_data[col].dtype}')
summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
summary.columns = out_cols
return summary

```

```
In [ ]: def find_duplicated(numeric_data, text_data, encodable_data):
d = False
all_columns = list(numeric_data.columns) + list(text_data.columns) + \
list(encodable_data.columns)
duplicate = [item for item, count in \
collections.Counter(all_columns).items() if count > 1]
if duplicate:
    print(f"There are duplicate features: {duplicate}")
    d = True
return d
```

```
In [ ]: def test_data_split(main_data, numeric_data, text_data, encodable_data):
m = False
split_features = list(numeric_data.columns) + list(text_data.columns) + \
list(encodable_data.columns)
missing = list(set(main_data)-set(split_features))
if missing:
    print(f"There are missing features: {missing}")
    m = True
return m
```

```
In [ ]: def review_data_split(main_data, numeric_data, text_data, encodable_data = \
pd.DataFrame()):
d = find_duplicated(numeric_data, text_data, encodable_data)
m = test_data_split(main_data, numeric_data, text_data, encodable_data)
if d != True and m != True:
    print("Data split good.")
```

```
In [ ]: def find_duplicates(data):
print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')
```

```
In [ ]: def count_yes(data):
total = 0
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    total += count
    print(f'{col}: {count}')
print(f'Total yes count: {total}')
```

Null Value Functions

The following functions will be used to update null (missing) values. To facilitate reading this document this section, containing mostly code blocks, is minimised.

```
In [ ]: def fill_nan_with_minus_one(data, feature):
new_data = data.copy()
new_data[feature] = data[feature].fillna(-1)
return new_data
```

```
In [ ]: def fill_nan_with_NA(data, feature):
new_data = data.copy()
new_data[feature] = data[feature].fillna("NA")
return new_data
```

```
In [ ]: def test_numeric_value_in_feature(feature, value):
test = feature.isin([-1]).sum()
return test
```

```
In [ ]: def test_catagorical_value_in_feature(dataframe, feature, value):
test = dataframe[feature][dataframe[feature] == value].count()
return test
```

```
In [ ]: def test_cat_list_for_NA(dataframe, cat_list):
for val in cat_list:
    print(val, test_catagorical_value_in_feature(dataframe, val, 'NA'))
```

```
In [ ]: def test_num_list_for_minus_one(dataframe, num_list):
for val in num_list:
    feature = dataframe[val]
    print(val, test_numeric_value_in_feature(feature, -1))
```

```
In [ ]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data
```

```
In [ ]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data
```

Reprocessing Functions

The following functions are used to reprocess the data with the aim of either, creating a new feature or enhancing an existing one. The ambition is to convert the data in such a way as to provide more clarity into potential insights. To facilitate reading this document this section, containing mostly code blocks, is minimised.

```
In [ ]: def renew_density(data):
    new_data = data.copy()
    try:
        new_data = new_data.drop(['Density'], axis=1)
    except:
        pass
    try:
        new_data = new_data.drop(['Density_trans'], axis=1)
    except:
        pass
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    new_data['Density_trans'] = stats.boxcox(new_data['Density'], 0.5)
    return new_data
```

```
In [ ]: def add_density(data, transform=True):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    if transform:
        new_data['Density_trans'] = stats.boxcox(new_data['Density'], 0.5)
    return new_data
```

```
In [ ]: def load_location(data):
    location_numeric_data_short_features = ['Location_X', 'Location_Y']
    location_numeric_data_short = data[location_numeric_data_short_features]
    location_numeric_data_short = add_density(location_numeric_data_short)
    location_numeric_data_short.head()
    location_data = location_numeric_data_short.copy()
    location_data.head()
    return location_data
```

Save Image Functions

These functions are used in saving the image to file. To facilitate reading this document this section, containing mostly code blocks, is minimised.

```
In [ ]: fig_no = 0
part = 'Part01'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"
```

```
In [ ]: def get_file_name(title):
    file_name = slugify(title)
    return file_name
```

```
In [ ]: def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(",", ";")
    return title
```

```
In [ ]: def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
```

```
fig_no = fig_no + '0'
fig_no = fig_no + str(no)
return fig_no

In [ ]: if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass

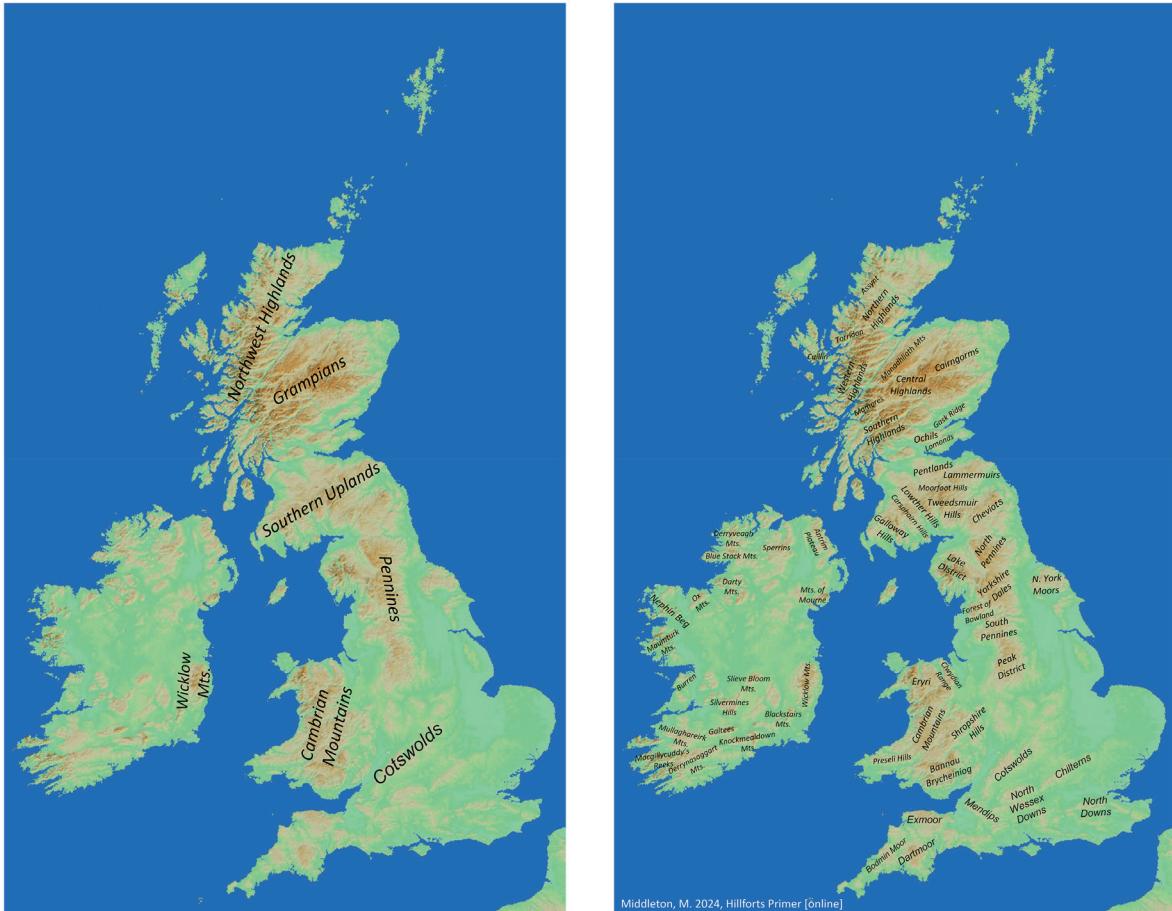
In [ ]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Part_01_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution, bbox_inches='tight')
    else:
        pass
```

Mountains & Hills

The maps below show the main mountain and hill ranges in the area covered by the Hillforts Atlas. The maps were created in QGIS 3.10.9 using the European Environment Agency's (EEA) Copernicus public digital elevation model web service, CopernicusDEM and the ESRI, World Hillshade web service.

CopernicusDEM: <https://image.discomap.eea.europa.eu/arcgis/rest/services/GioLandPublic/DEM/MapServer>

World Hillshade: https://services.arcgisonline.com/arcgis/rest/services/Elevation/World_Hillshade/MapServer/0



The Hills and Mountains of Britain and Ireland

Mike Middleton *CC BY-SA 3.0*

```
In [ ]: from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [ ]: # "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-atlas-source-data-csv/hillforts.csv"
# main/hillforts-atlas-source-data-csv/hillforts.csv"
hillforts_csv = r"https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-atlas-source-data-csv/hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)

<ipython-input-95-541076404a42>: 4: DtypeWarning: Columns (10, 12, 68, 83, 84, 85, 86, 165, 183) have mixed types. Specify dtype option on import or set low_memory=False.
    hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

	OBJECTID	Main_Atlas_Number	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Display_N
0	1	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Aconbury C Hereford (Aconbury Bea
1	2	2	EN	England	EN0002 Bach Camp, Herefordshire	Bach Camp	NaN	Bach C Hereford

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [ ]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
            'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.)
```

Using all 4147 record in the Hillforts Atlas.

Download Function

This functions is used in saving the reprocessed data to file.

```
In [ ]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', \
                'republic-of-ireland', 'northern-ireland', \
                'isle-of-man', 'roi-ni', 'eng-wal-sco-iom']:
                if pkg.shape[0] == hillforts_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
                else:
                    dl = data_list[0]
            dl = dl.replace('\r', ' ', regex=True)
            dl = dl.replace('\n', ' ', regex=True)
            fn = 'hillforts_primer_' + filename
            fn = get_file_name(fn)
            dl.to_csv(fn+'.csv', index=False)
            files.download(fn+'.csv')
    else:
        pass
```

Review Data

The Atlas contains a number of sections where the data has been grouped by what information the data conveys; Admin data, Location data, Land use data, etc. As these sections are reviewed, the features in each section will be separated into three categories:

1. Numeric data;
2. Text data (free text);
3. Text data that can be encoded numerically.

Data that can be encoded numerically will contain values such as yes/no or lists or repeated values, as might be found in a thesaurus. All columns will be reviewed for rows containing no information (null values) and, where found, these null values will be replaced, with an appropriate substitute, which enables null values to still be identified.

Bias

Bias is a term that is used often in this study. Bias does not mean that the data is wrong or of poor quality rather, bias means that there is something about the data that needs to be understood when considering any interpretations made using it. An example might be a regional recording bias, where data has been collected intensively in one area and not at all in another. It is important to be aware of this so that any suggestion of regional significance takes this into account.

Recognising bias in the data helps identify potential research opportunities. These may manifest as regions where research in a specific area might fill an under-recorded detail or it could reveal where standardising terminology across the Atlas might improve insights.

Missing Data

Missing values or empty features can cause machine learning tools to fail or produce misleading results. It is important to process the data so that missing values are resolved. One of the aims of this study is to produce data that can be used by machine learning tools to impute missing data (estimate based on similarities). This can be done by training machine learning models using data which contains no missing values and then assessing how sensitive the data is to missing data by incrementally removing values. If a model can be identified that is not sensitive to small quantities of missing data, imputation might be possible. It is important therefore, to fill null values in the data and for these values to remain identifiable so that the data can be filtered and imputation can be attempted.

This study will use two values to replace null data:

- In numeric data, -1;
- in text data, 'NA'.

Before these are added, it is important to ensure these values are not present within the existing data. If they are, an alternative replacement will be used.

Ref: <https://scikit-learn.org/stable/modules/impute.html>

Rows and Features (columns)

The 'info' and 'shape' functions confirm there are 4147 rows and 244 columns. Note that in Data Science columns are frequently referred to as **features** and that the index, mentioned above, is not included in the column count.

Ref: <https://developers.google.com/machine-learning/glossary>

```
In [ ]: hillforts_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 7.7+ MB
```

```
In [ ]: hillforts_data.shape
```

```
Out[ ]: (4147, 244)
```

Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-packages. Where needed, these data extracts will be combined with 'Name and Number' features to ensure the data can be understood and can, if needed, be concorded.

```
In [ ]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Admin Data

The 'admin_data' is a short data package containing only the admin features. This package contains the automated spatial data identifier, 'OBJECTID' as well as information on site name, country, county, Historic Environment Record (HER) IDs, National Monument Record (NMR) IDs, Scheduled Monument (SM) IDs and mapsheet.

This data package contains both numeric and text data types and these - like all future data packages - will be split into numeric data, text data and text data that can be encoded numerically.

Unique identifiers are useful in identifying sites and linking them with information held elsewhere but, they are unlikely to offer much in terms of understanding and interpretation. When it comes to applying data science tools, continuous unique ID number sequences are more likely to hinder than help. It is likely that when it comes to selecting features for further analysis, these features will be dropped.

```
In [ ]: admin_features = [
    'OBJECTID',
    'Main_Country_Code',
    'Main_Country',
    'Main_Tile_Name',
    'Main_Site_Name',
    'Main_Alert_Name',
    'Main_HER',
    'Main_HER_PRN',
    'Main_HER_ID',
    'Main_NMR_Mapsheet',
    'Main_NMR_ID',
    'Main_SM',
    'Main_Summary',
    'Main_Boundary']
```

```
admin_data = hillforts_data[admin_features].copy()
admin_data.head()
```

Out[]:	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_HER	Main_HER_PRN	Main_
0	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Herefordshire	MHE413	
1	2	EN	England	EN0002 Bach Camp, Herefordshire	Bach Camp	NaN	Herefordshire	MHE52	
2	3	EN	England	EN0003 Backbury Camp, Herefordshire	Backbury Camp	Ethelbert's Camp	Herefordshire	MHE411	
3	4	EN	England	EN0004 Brandon Camp, Herefordshire	Brandon Camp	NaN	Herefordshire	MHE818	
4	5	EN	England	EN0005 British Camp, Herefordshire	British Camp	Herefordshire Beacon	Herefordshire	MHE435	

In []: admin_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   OBJECTID        4147 non-null   int64  
 1   Main_Country_Code 4147 non-null   object  
 2   Main_Country     4147 non-null   object  
 3   Main_Title_Name  4147 non-null   object  
 4   Main_Site_Name   4147 non-null   object  
 5   Main_Alt_Name    1757 non-null   object  
 6   Main_HER          4140 non-null   object  
 7   Main_HER_PRN     4100 non-null   object  
 8   Main_HER_ID      205 non-null   object  
 9   Main_NMR_Mapsheet 4051 non-null   object  
 10  Main_NMR_ID      3465 non-null   object  
 11  Main_SM          2282 non-null   object  
 12  Main_Summary     4147 non-null   object  
 13  Main_Boundary    4147 non-null   object  
dtypes: int64(1), object(13)
memory usage: 453.7+ KB
```

The 'info' function shows there are a number of admin data features containing null values. These will be resolved in [Admin encodable Data - Replace Null Values](#) below.

The function also shows that the OBJECTID is an integer (numeric) while all the other fields are of type 'object'. We can see from the table above that some of these features look to contain numeric data. The task now is to assess if all the rows in these features contain numeric values and, if not, how simple it might be to convert these features into a numeric format.

In []: admin_data_numeric_review = test_numeric(admin_data)

admin_data_numeric_review

OBJECTID int64

Out[]:

	Feature	Entries	Numeric	Non-Numeric	Null
0	Main_Country_Code	4147	0	4147	0
1	Main_Country	4147	0	4147	0
2	Main_Title_Name	4147	0	4147	0
3	Main_Site_Name	4147	0	4147	0
4	Main_Alt_Name	1757	0	1757	2390
5	Main_HER	4140	0	4140	7
6	Main_HER_PRN	4100	2095	2005	47
7	Main_HER_ID	205	106	99	3942
8	Main_NMR_Mapsheet	4051	0	4051	96
9	Main_NMR_ID	3465	3410	55	682
10	Main_SM	2282	1785	497	1865
11	Main_Summary	4147	0	4147	0
12	Main_Boundary	4147	0	4147	0

Admin Numeric Data

OBJECTID

'OBJECTID' is a unique index automatically created by Esri software. 'OBJECTID' is a continuous sequence of numbers, starting at 1 and ending at 4147. There are no duplicate values.

Ref: <https://desktop.arcgis.com/en/arcmap/latest/manage-data/tables/fundamentals-of-objectid-fields.htm>

```
In [ ]: admin_numeric_features = [
    'OBJECTID']

admin_numeric_data = admin_data@admin_numeric_features].copy()
admin_numeric_data.head()
```

Out[]:

	OBJECTID
0	1
1	2
2	3
3	4
4	5

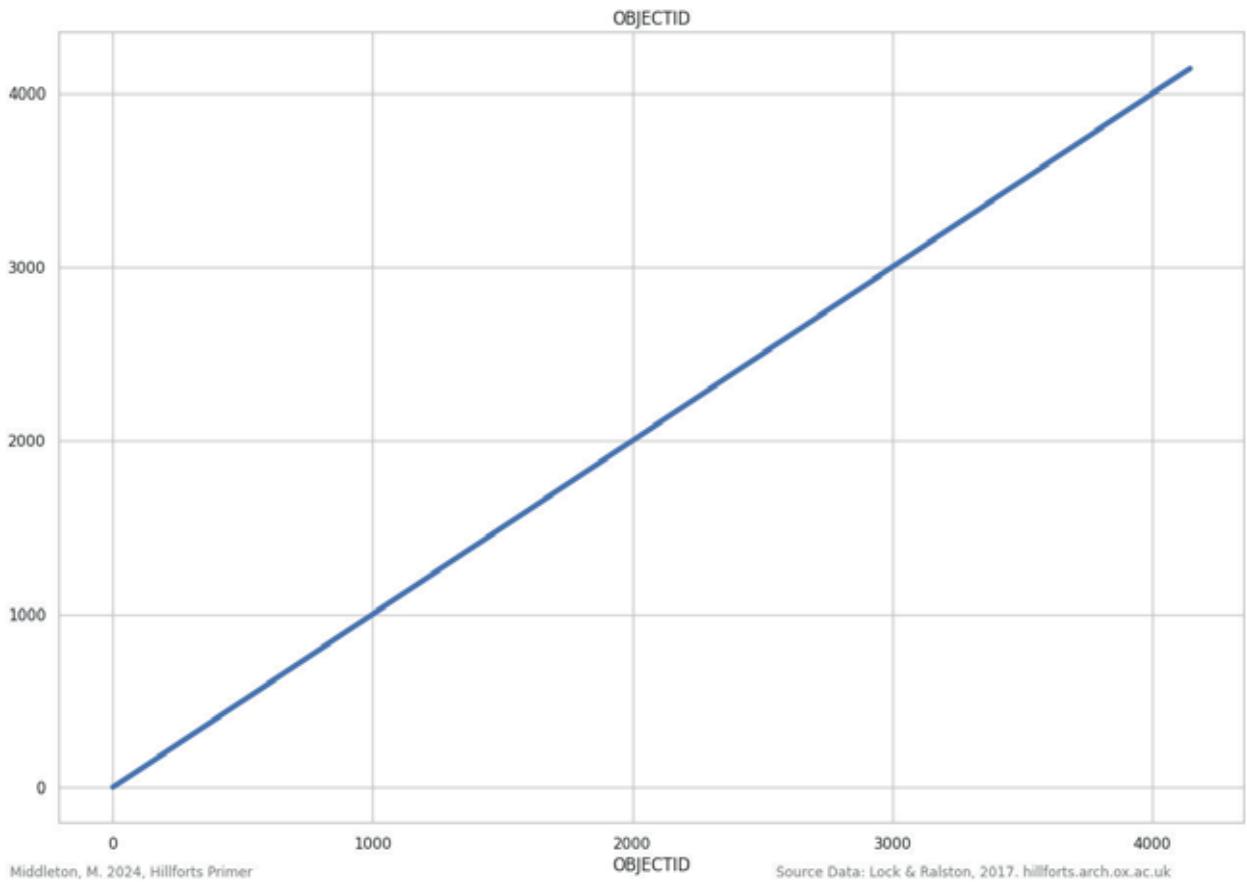
```
In [ ]: admin_numeric_data['OBJECTID'].describe()
```

```
Out[ ]: count    4147.000000
mean    2074.000000
std     1197.280112
min      1.000000
25%    1037.500000
50%    2074.000000
75%    3110.500000
max    4147.000000
Name: OBJECTID, dtype: float64
```

```
In [ ]: duplicates = find_duplicates(admin_numeric_data['OBJECTID'])
duplicates

0 duplicates.
```

```
In [ ]: plot_continuous(admin_numeric_data['OBJECTID'], 'OBJECTID', 'OBJECTID')
```



Admin Text Data

The text, in this context, means the data is free text. Data mining may be applied to these features in a future study.

```
In [ ]: admin_text_features = [
    'Main_Title_Name',
    'Main_Site_Name',
    'Main_Alt_Name',
    'Main_Summary']

admin_text_data = admin_data[admin_text_features].copy()
admin_text_data.head()
```

	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Summary
0	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Large, wooded, univallate, partial contour hil...
1	EN0002 Bach Camp, Herefordshire	Bach Camp		NaN Univallate, contour hillfort located on summit...
2	EN0003 Backbury Camp, Herefordshire	Backbury Camp	Ethelbert's Camp	Multivallate, contour hillfort with three bank...
3	EN0004 Brandon Camp, Herefordshire	Brandon Camp		NaN Univallate probable hillslope fort, re-used in...
4	EN0005 British Camp, Herefordshire	British Camp	Herefordshire Beacon	One of the finest and most spectacular contour...

```
In [ ]: admin_text_data_review = test_numeric(admin_text_data)
admin_text_data_review
```

	Feature	Entries	Numeric	Non-Numeric	Null
0	Main_Title_Name	4147	0	4147	0
1	Main_Site_Name	4147	0	4147	0
2	Main_Alt_Name	1757	0	1757	2390
3	Main_Summary	4147	0	4147	0

Main Title Name

The Main Title Name is a concatenation of multiple features. As the data is already present in the dataset and as the feature is another unique identifier, this feature can be dropped. There are no null values.

```
In [ ]: admin_text_data.head()
```

	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Summary
0	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Large, wooded, univallate, partial contour hil...
1	EN0002 Bach Camp, Herefordshire	Bach Camp	NaN	Univallate, contour hillfort located on summit...
2	EN0003 Backbury Camp, Herefordshire	Backbury Camp	Ethelbert's Camp	Multivallate, contour hillfort with three bank...
3	EN0004 Brandon Camp, Herefordshire	Brandon Camp	NaN	Univallate probable hillslope fort, re-used in...
4	EN0005 British Camp, Herefordshire	British Camp	Herefordshire Beacon	One of the finest and most spectacular contour...

Main Site Name

All the hillforts have a Main Site Name and there are therefore no null values.

```
In [ ]: Main_Site_Name_not_null = admin_text_data[~admin_text_data['Main_Site_Name'].isna()]
Main_Site_Name_not_null['Main_Site_Name'].describe()
```

```
Out[ ]: count      4147
unique     3975
top       Castle Hill
freq        10
Name: Main_Site_Name, dtype: object
```

```
In [ ]: admin_text_data[admin_text_data['Main_Site_Name']=='Castle Hill'].head()
```

	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Summary
245	SC0252 Castle Hill, Dumfries & Galloway	Castle Hill	NaN	This small fort crowns a hillock on the wester...
247	SC0254 Castle Hill, Dumfries & Galloway	Castle Hill	Cumstounend, Compstonend	This fort is situated on a low rocky ridge and...
270	SC0277 Castle Hill, Dumfries & Galloway	Castle Hill	Castlegower	The remains of this small fortification occupy...
297	SC0304 Castle Hill, Dumfries & Galloway	Castle Hill	Dalwhat	This fort occupies the summit of a spur on the...
332	SC0339 Castle Hill, Dumfries & Galloway	Castle Hill	NaN	This oval fort occupies a steep-sided hillock ...

Main Alt Name

Only 1757 hillforts have an alternative name. There are 2390 null values.

```
In [ ]: Main_Alt_Name_not_null = admin_text_data[~admin_text_data['Main_Alt_Name'].isna()]
Main_Alt_Name_not_null['Main_Alt_Name'].describe()
```

```
Out[ ]: count      1757
unique     1708
top       Castle Hill
freq        8
Name: Main_Alt_Name, dtype: object
```

```
In [ ]: admin_text_data[admin_text_data['Main_Alt_Name']=='Castle Hill'].head()
```

	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Summary
139	EN0142 Sinodun Hill Camp, Oxfordshire	Sinodun Hill Camp	Castle Hill	W of Little Wittenham, a contour fort on Castl...
237	SC0244 Drummore Castle, Dumfries & Galloway	Drummore Castle	Castle Hill	This fort is situated in woodland on the NE si...
326	SC0333 Castlehill, Dumfries & Galloway	Castlehill	Castle Hill	This heavily ploughed down fort is situated on...
489	EN0508 East Moneylaws Camp, Northumberland	East Moneylaws Camp	Castle Hill	At 189m OD, a multivallate hillfort visible as...
1663	SC1746 Westside, South Lanarkshire	Westside	Castle Hill	This small fortified enclosure, which has been...

Main Summary

The Main Summary is a free text summary of each hillfort. There are no null values.

```
In [ ]: sample_summary = admin_text_data['Main_Summary']\
[admin_text_data['Main_Title_Name']==EN0001 Aconbury Camp, Herefordshire].item()
sample_summary
```

```
Out[ ]: 'Large, wooded, univallate, partial contour hillfort located on Aconbury Hill, following the contours but sloping to the W, and on the interfluve above the Rivers Wye and Severn. Precipitous slopes to the W and N. Rampart surrounds camp, with a surviving main ditch to the S and E, elsewhere not visible, although it is possible that the steep slope is on the N and W precluded a ditch here. Internal area c. 7.1ha and footprint 9.3ha. The rampart reaches up to 4.5m in the S where the ditch is up to 1.5m deep and is impressive on the W overlooking steep slopes. Possible internal revetments within ramparts. Berm on N and W. Internal quarry scoops, especially on the S and N side. Slight investigations 1948-51 found occupation similar to Sutton Walls hillfort (Atlas No 0031) and Dinedor Camp (Atlas No 0013), with pottery similar to the former site. Two original entrances and four modern gaps. Occupied during the Civil War in 1642 and 1645. Mixed woodland since 19th century with internal tracks. Bracken, bramble, sapling and coppice re-growth. Visitor erosion of paths where cross rampart, with horse and quad bikes. Some quarrying. On 1st Ed. OS map (1888).'
```

Admin Text Data - Resolve Null Values

Test for the presence of 'NA' in 'Main_Alt_Name' ('Main_Alt_Name' is the only one of the text features to contain null values).

```
In [ ]: test_cat_list_for_NA(admin_text_data, ['Main_Alt_Name'])
Main_Alt_Name 0
```

Fill null values in 'Main_Alt_Name' with 'NA'.

```
In [ ]: admin_text_data = update_cat_list_for_NA(admin_text_data, ['Main_Alt_Name'])
admin_text_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Main_Title_Name  4147 non-null   object 
 1   Main_Site_Name   4147 non-null   object 
 2   Main_Alt_Name    4147 non-null   object 
 3   Main_Summary     4147 non-null   object 
dtypes: object(4)
memory usage: 129.7+ KB
```

Admin Encodable Data

Features from Admin data that has the potential to be encoded numerically.

```
In [ ]: admin_encodable_features = [
    'Main_Country_Code',
    'Main_Country',
    'Main_HER',
    'Main_HER_PRN',
    'Main_HER_ID',
    'Main_NMR_Mapsheet',
    'Main_NMR_ID',
    'Main_SM',
    'Main_Boundary']

admin_encodable_data = admin_data[admin_encodable_features].copy()
admin_encodable_data.head()
```

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Mapsheet	Main_NMR_ID	Main_SM	Main_Boundary
0	EN	England	Herefordshire	MHE413	910	SO 53 SW 1	110371	1001754	
1	EN	England	Herefordshire	MHE52	344	SO 56 SW 3	110884	1007316	
2	EN	England	Herefordshire	MHE411	908	SO 53 NE 2	110051	1003534	
3	EN	England	Herefordshire	MHE818	1639	SO 47 SW 2	108810	1011016	
4	EN	England	Herefordshire	MHE435	932	SO 74 SE 3	113786	1001792	

The following tests how many of the entries in a feature are numeric. This helps assess if each feature is a text based pick list or a numeric identifier.

```
In [ ]: admin_encodable_data_review = test_numeric(admin_encodable_data)
admin_encodable_data_review
```

	Feature	Entries	Numeric	Non-Numeric	Null
0	Main_Country_Code	4147	0	4147	0
1	Main_Country	4147	0	4147	0
2	Main_HER	4140	0	4140	7
3	Main_HER_PRN	4100	2095	2005	47
4	Main_HER_ID	205	106	99	3942
5	Main_NMR_Mapsheet	4051	0	4051	96
6	Main_NMR_ID	3465	3410	55	682
7	Main_SM	2282	1785	497	1865
8	Main_Boundary	4147	0	4147	0

Main Country Code

This feature contains six unique text values and no null values:

1. England - EN;
2. Wales - WA;
3. Scotland - SC;
4. Northern Ireland - NI;
5. The Republic of Ireland - IR;
6. The Isle of Man - IOM.

```
In [ ]: pd.unique(admin_encodeable_data['Main_Country_Code'])
Out[ ]: array(['EN', 'WA', 'SC', 'NI', 'IR', 'IOM'], dtype=object)
```

The country codes contain the same information as [Main Country](#), in an encoded form. Removing one of these features would result in no loss of information. See: [Admin Data - Drop Duplicate Admin Features](#).

Main Country Data Packages

The 'Main_Country_Code' is used here to create individual national data packages.

```
In [ ]: england_data = hillforts_data[hillforts_data['Main_Country_Code'] == 'EN'].copy()
wales_data = hillforts_data[hillforts_data['Main_Country_Code'] == 'WA'].copy()
scotland_data = hillforts_data[hillforts_data['Main_Country_Code'] == 'SC'].copy()
ni_data = hillforts_data[hillforts_data['Main_Country_Code'] == 'NI'].copy()
roi_data = hillforts_data[hillforts_data['Main_Country_Code'] == 'IR'].copy()
iom_data = hillforts_data[hillforts_data['Main_Country_Code'] == 'IOM'].copy()
```

England

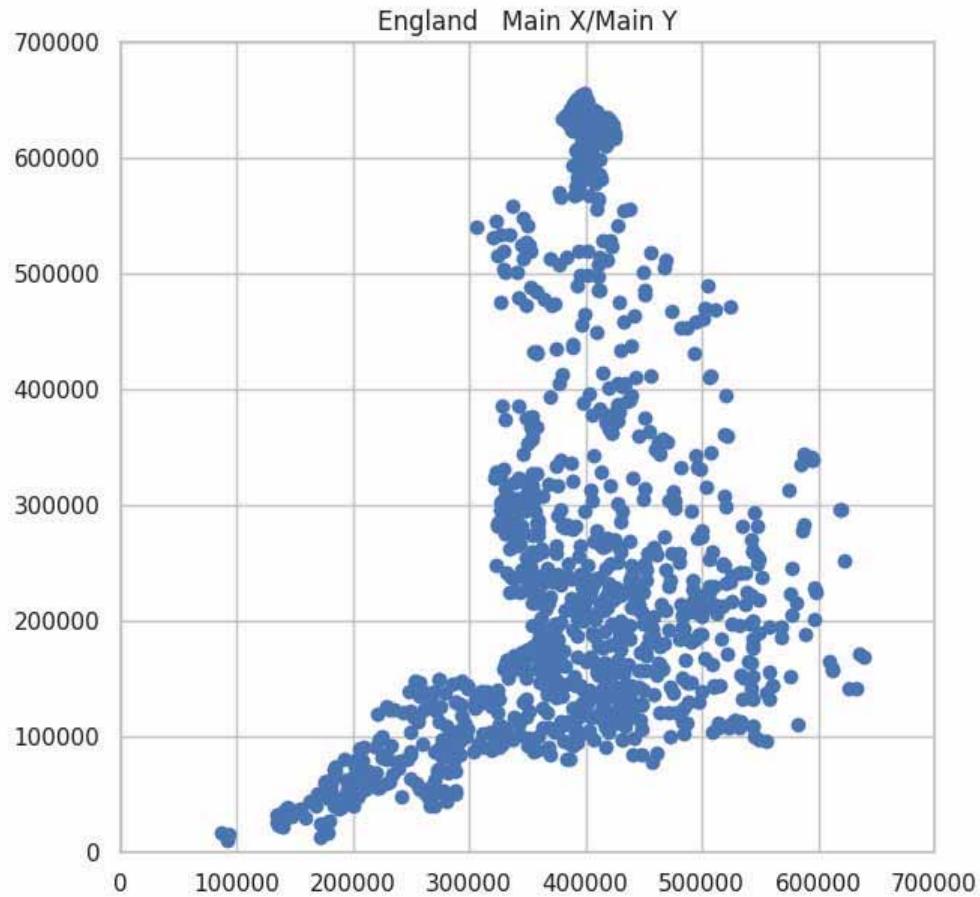
England contains 1225 records. The distribution of these records can be seen below, plotted against the Ordnance Survey National Grid.

See: [Main_X / Main_Y](#)

See: [Main_Coordinate_System](#)

```
In [ ]: england_data.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1225 entries, 0 to 4146
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 2.3+ MB
```

```
In [ ]: plot_england(england_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Notes on Downloading Data

The following download options (by nation) are commented out and will not run. To enable the download, remove the '#' from the beginning of the code block below, follow the instructions in [User Settings](#) and then select **Runtime>Run all**, in the main menu above. This will download an unedited, filtered selection, of the source data.

```
In [ ]: #download([england_data], 'england')
```

Wales

Wales contains 690 records. These are plotted against the Ordnance Survey National Grid.

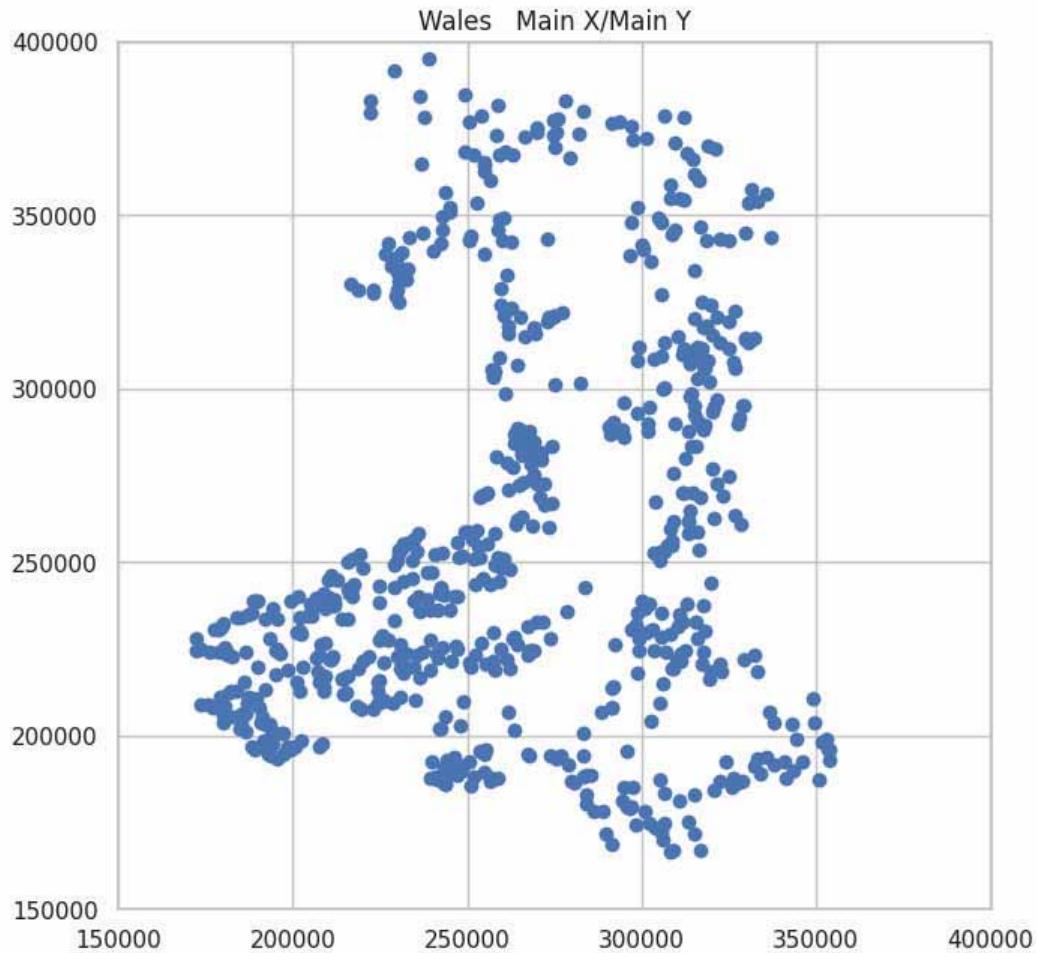
See: [Main_X / Main_Y](#)

See: [Main_Coordinate_System](#)

```
In [ ]: wales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 690 entries, 70 to 4127
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 1.3+ MB
```

```
In [ ]: plot_wales(wales_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download([wales_data], 'wales')
```

Scotland

Scotland contains 1695 records. These are plotted against the Ordnance Survey National Grid.

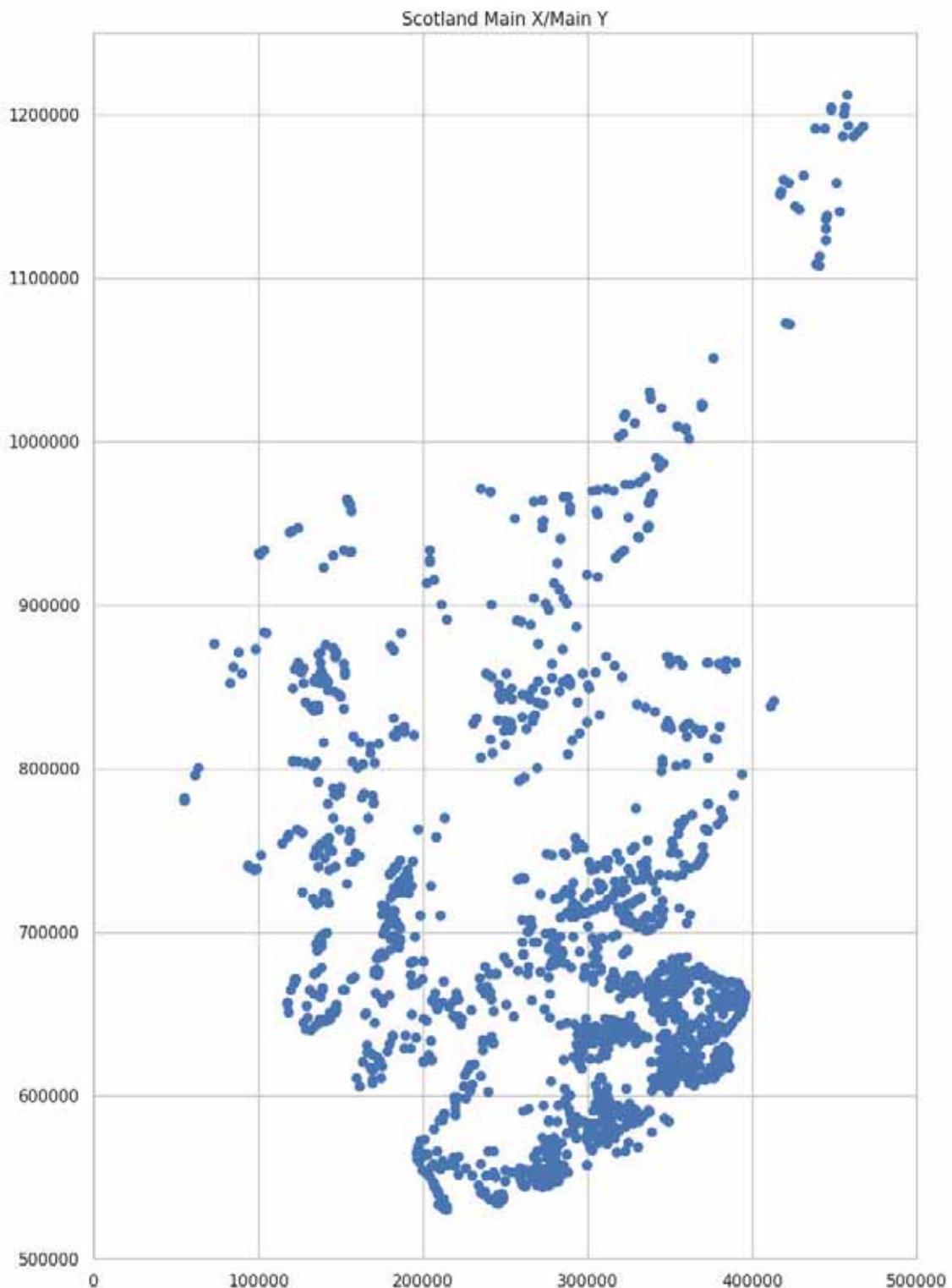
See: [Main_X / Main_Y](#)

See: [Main_Coordinate_System](#)

```
In [ ]: scotland_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1695 entries, 120 to 4145
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 3.2+ MB
```

```
In [ ]: plot_scotland(scotland_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download([scotland_data], 'scotland')
```

Northern Ireland

Northern Ireland contains 32 records. These are plotted against the, 'Irish Transverse Mercator' grid.

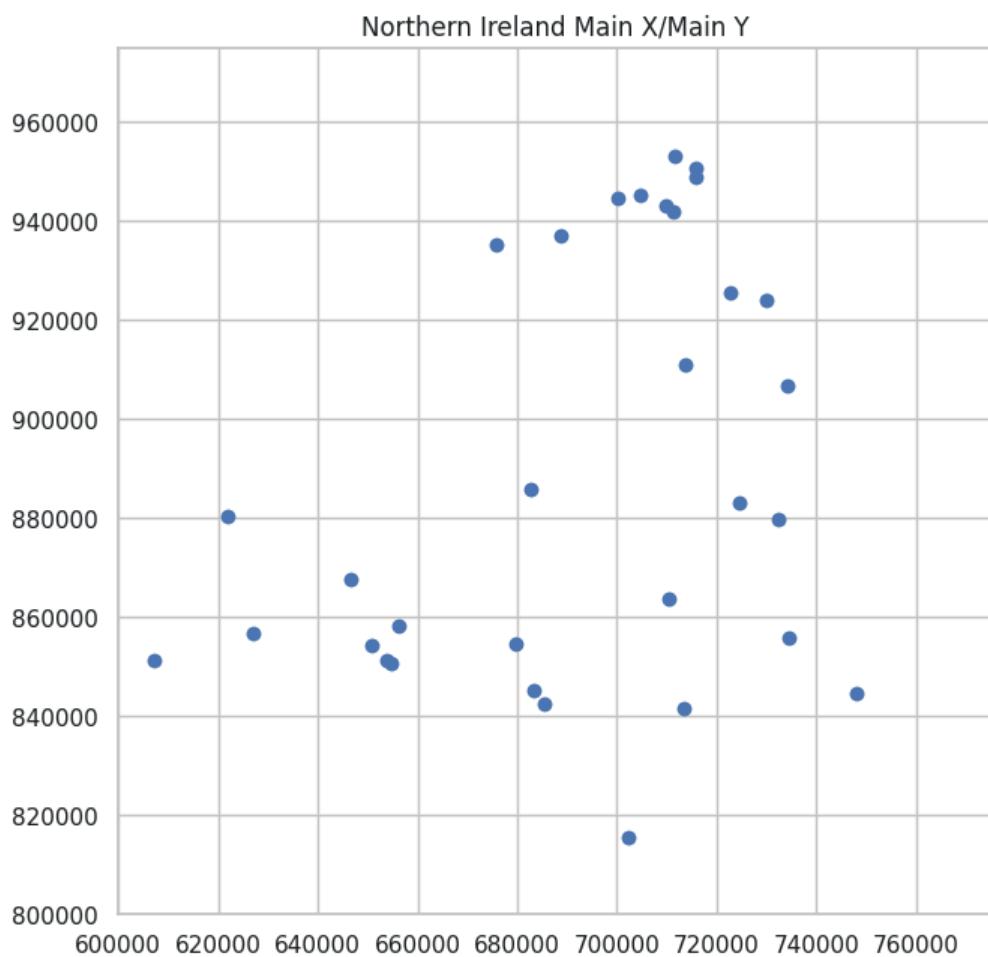
See: [Main_X / Main_Y](#)

See: [Main_Coordinate_System](#)

```
In [ ]: ni_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32 entries, 640 to 4136
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 61.2+ KB
```

```
In [ ]: plot_ni(ni_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download([ni_data], 'northern-ireland')
```

Republic of Ireland

The Republic of Ireland contains 475 records. These are plotted against the, 'Irish Transverse Mercator' grid.

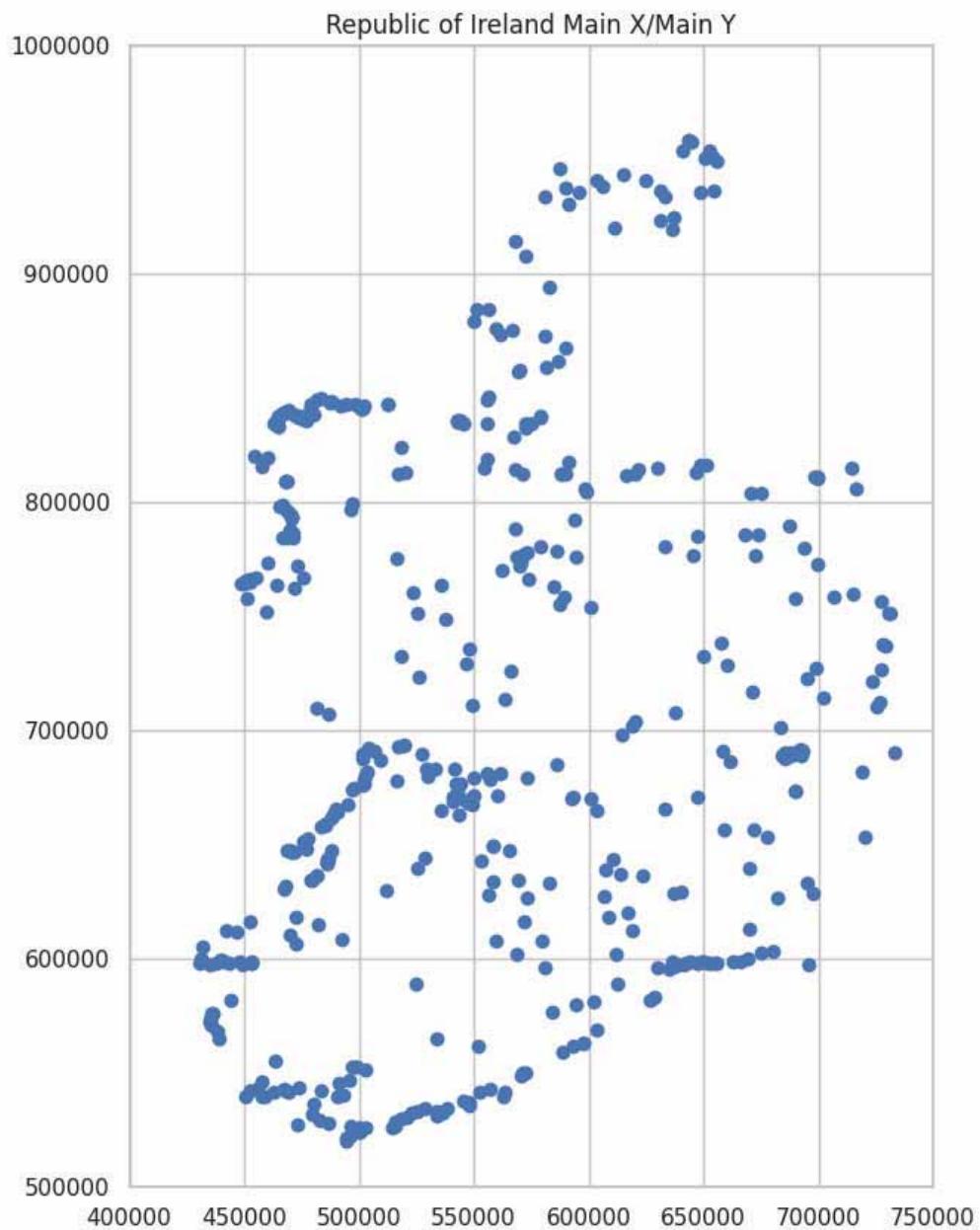
See: [Main_X / Main_Y](#)

See: [Main_Coordinate_System](#)

```
In [ ]: roi_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 475 entries, 641 to 4125
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 909.2+ KB
```

```
In [ ]: plot_roi(roi_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download([roi_data], 'republic-of-ireland')
```

Isle Of Man

The Isle of Man contains 30 records. These are plotted against the Ordnance Survey National Grid.

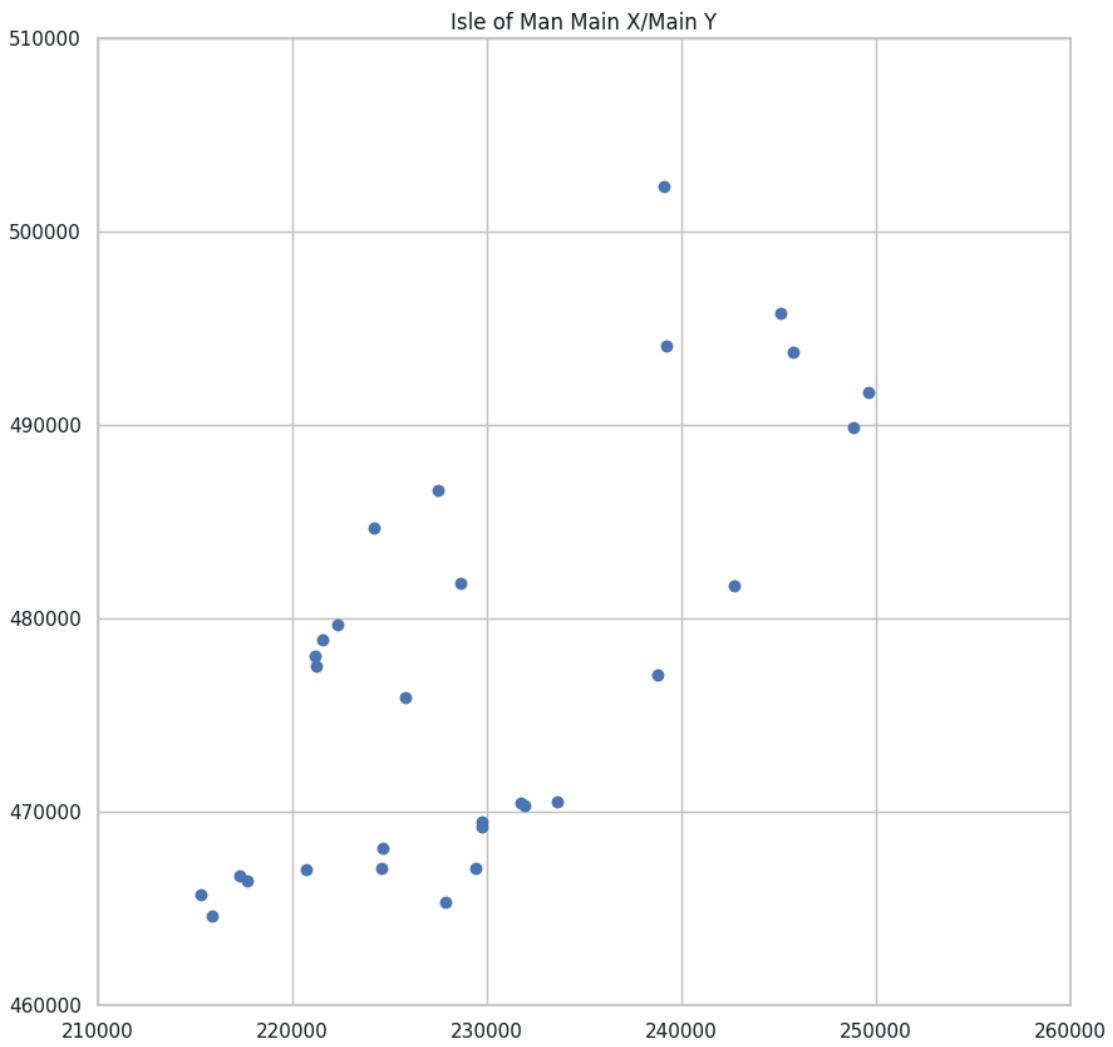
See: [Main_X / Main_Y](#)

See: [Main_Coordinate_System](#)

```
In [ ]: iom_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30 entries, 3034 to 4009
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 57.4+ KB
```

```
In [ ]: plot_iom(iom_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download(['iom_data'], 'isle-of-man')
```

Republic of Ireland and Northern Ireland Combined

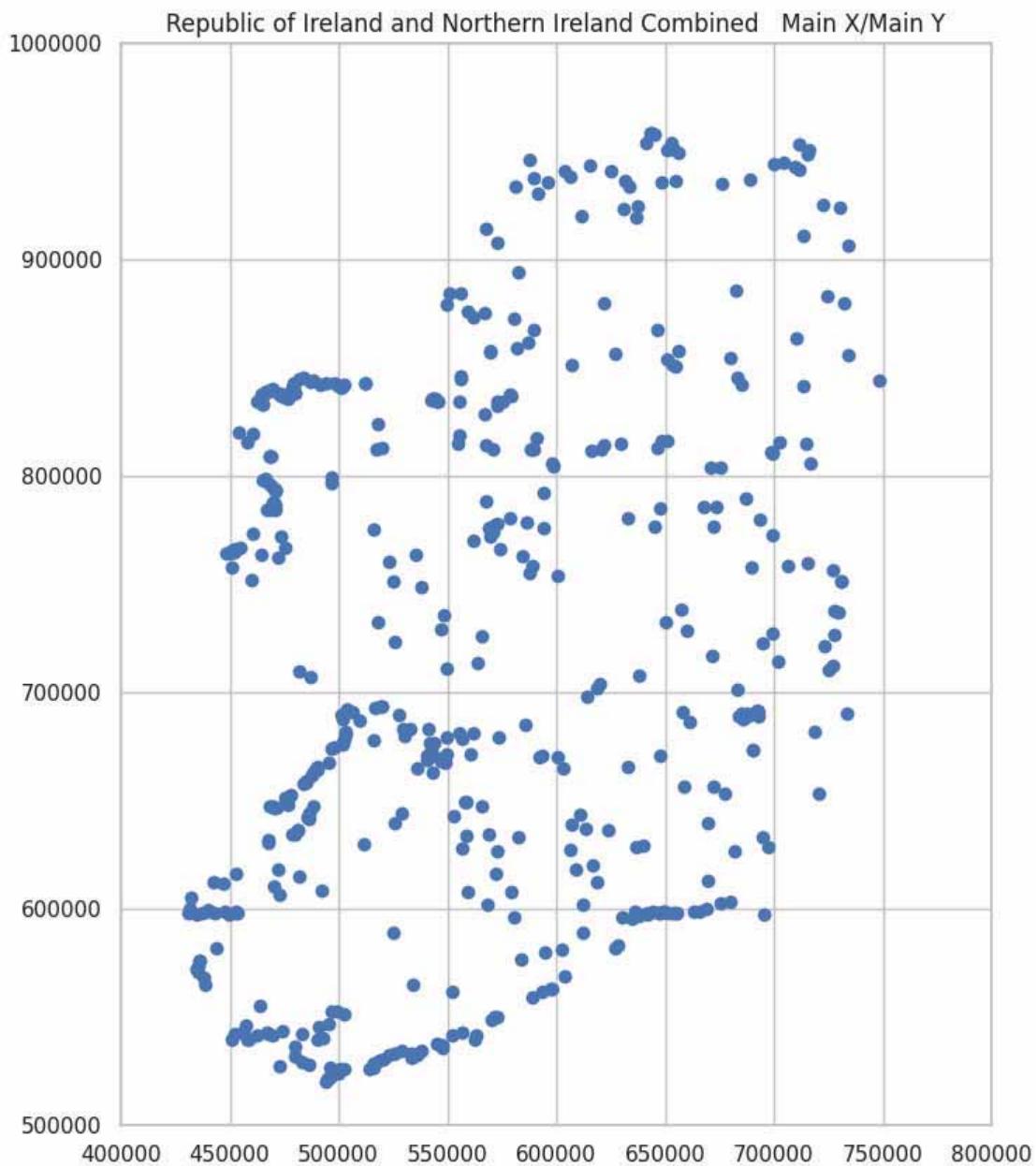
All 507 records on the island or Ireland.

```
In [ ]: roi_and_ni_data = \
hillforts_data[hillforts_data['Main_Country'].\
isin(['Republic of Ireland', 'Northern Ireland'])]
```

```
In [ ]: roi_and_ni_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 507 entries, 640 to 4136
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 970.4+ KB
```

```
In [ ]: plot_ireland(roi_and_ni_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download(['roi_and_ni_data'], 'roi-ni')
```

England, Wales, Scotland and Man Combined

All Atlas data excluding the island or Ireland. 3640 records.

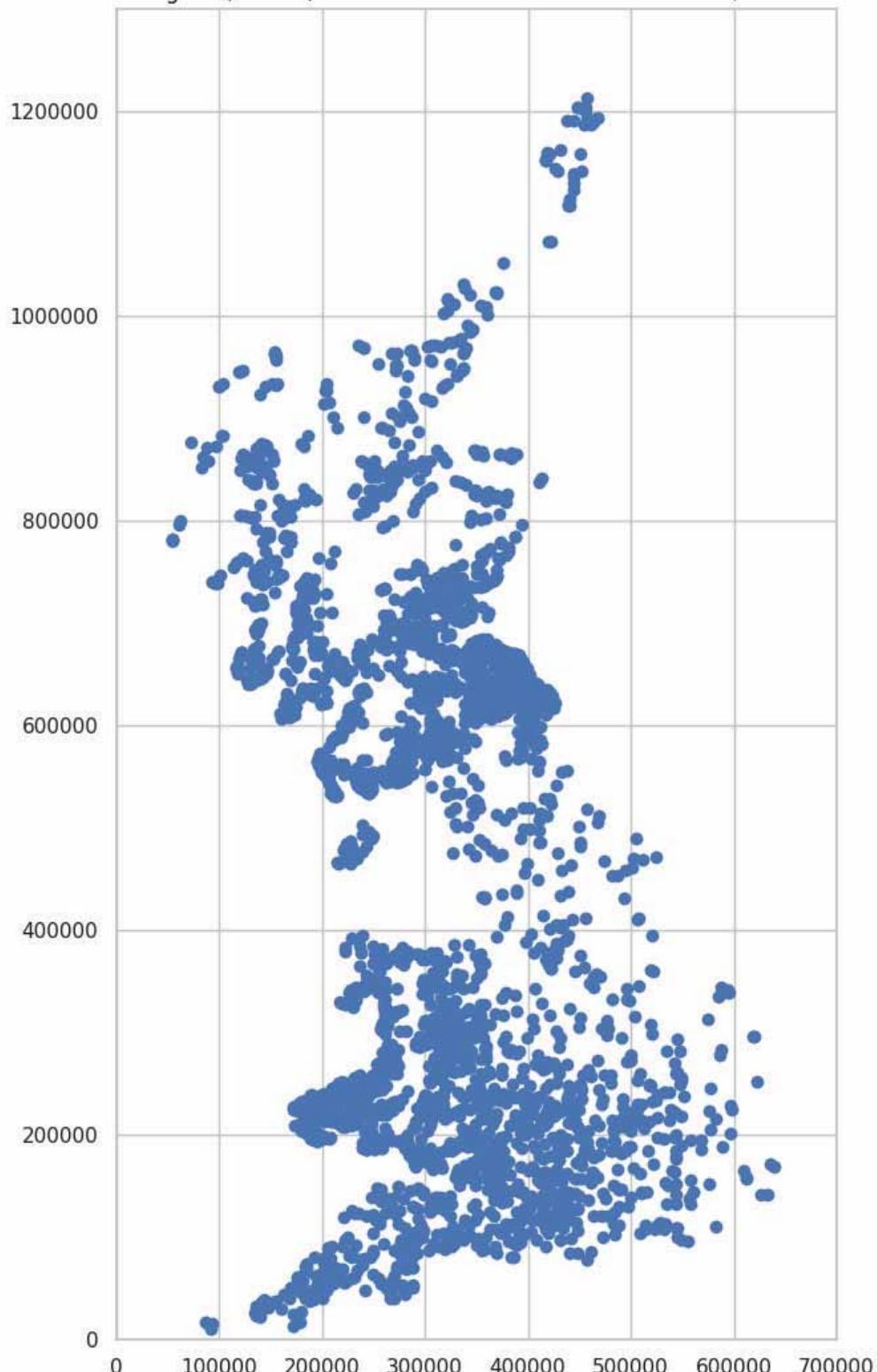
```
In [ ]: eng_wal_sco_man_data = \
hillforts_data[hillforts_data['Main_Country'].\
isin(['England', 'Wales', 'Scotland', 'Isle of Man'])]
```

```
In [ ]: eng_wal_sco_man_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3640 entries, 0 to 4146
Columns: 244 entries, OBJECTID to Record_URL
dtypes: float64(19), int64(6), object(219)
memory usage: 6.8+ MB
```

```
In [ ]: plot_all_but_ireland(eng_wal_sco_man_data)
```

England; Wales; Scotland and Man Combined Main X/Main Y



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Notes on Downloading Data](#)

```
In [ ]: #download([eng_wal_sco_man_data], 'eng-wal-sco-ion')
```

Main Country

[Main_Country_Code](#) and 'Main_Country' contain the same information. As these are duplicates, one of these features will be dropped: See: [Admin Data - Drop Duplicate Admin Features](#). There are no null values.

```
In [ ]: pd.unique(admin_n_encodeable_data['Main_Country'])
Out[ ]: array(['England', 'Wales', 'Scotland', 'Northern Ireland',
       'Republic of Ireland', 'Isle of Man'], dtype=object)
```

Main HER

The Main Historic Environment Record (HER) name. There is also a mix of numeric and categorical data. These will be resolved in [Admin Encodable Data - Resolve Null Values](#).

There are 88 unique values and seven null values (top 25 record counts by HER are shown below). To see more, change the value in the square brackets below and rerun the document as described in [User Settings](#).

Main_HER									
	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Mapsheet	Main_NMR_ID	Main_SM	Main
1141	SC	Scotland	NaN	No record found	NaN	NS 76 SE 2	45784	NaN	
1375	SC	Scotland	NaN	No record found	NaN	NS 57 NE 27	44425	1749	
1401	SC	Scotland	NaN	No record found	NaN	NS 67 NW 6	45194	1734	
1405	SC	Scotland	NaN	No record found	NaN	NS 67 SW 3	45262	NaN	
1420	SC	Scotland	NaN	No record found	NaN	NS 77 NW 28	45894	90008	
1423	SC	Scotland	NaN	No record found	NaN	NS 77 SE 3	45924	NaN	
4145	SC	Scotland	NaN	NaN	NaN	NH 45 SE 57	339438.0	NaN	

```
In [ ]: Main_HER_not_null = \
admin_n_encodeable_data[~admin_n_encodeable_data['Main_HER'].isna()].dropna()
Main_HER_not_null['Main_HER'].describe()
```

```
Out[ ]: count                4140
unique                  88
top      Archaeological Survey of Ireland SMR Database
freq                   475
Name: Main_HER, dtype: object
```

```
In [ ]: admin_n_encodeable_data['Main_HER'].value_counts()[:30]
```

Archaeological Survey of Ireland SMR Database	475
Scottish Borders	408
The West of Scotland Archaeology Service	326
Dyfed	302
Dumfries & Galloway	286
Northumberland	270
Highland HER	208
Clwyd Powys	184
Glamorgan Gwent	108
Gwynedd	96
East Lothian Council	89
Devon	86
Cornwall and Scilly	80
Perth and Kinross Heritage Trust	76
Shropshire	63
Hampshire	50
Wiltshire and Swindon	49
Glostershire	47
Fife Council	44
Aberdeenshire Historic Environment Record	42
Herefordshire	38
Somerset	38
Dorset	37
Oxfordshire	37
Angus SMR per Aberdeenshire Council	35
Shetland Amenity Trust	33
Northern Ireland Sites and Monuments Record	32
Isle of Man	30
Stirling	30
Comhairle nan Eilean Siar - Western Isles Sites and Monuments Record	30

Main HER PRN

This is assumed to stand for, Main Historic Environment Record Primary Reference Number. This is the unique ID within each Main_HER database. This feature contains a mix of data formats (categorical and numeric). It also contains null values, multiples and question marks. Null values will be resolved in [Admin Encodable Data - Resolve Null Values](#).

The 47 null 'Main_HER_PRN' entries relate to sites predominantly in Fife, with one each in Shetland and Edinburgh. There is a single NaN value (not a number = null). In addition to null values, this feature also contains the statement, 'No record found'.

```
In [ ]: null_Main_HER_PRN = \
admin_n_encodeable_data[admin_n_encodeable_data['Main_HER_PRN'].isna()]
```

```
In [ ]: null_Main_HER_PRN['Main_HER'].value_counts()
```

```
Out[ ]: Fife Council      44
City of Edinburgh        1
Shetland Amenity Trust   1
Name: Main_HER, dtype: int64
```

```
In [ ]: set([list(pd.unique(null_Main_HER_PRN['Main_HER']))])
```

```
Out[ ]: {'City of Edinburgh', 'Fife Council', 'Shetland Amenity Trust', nan}
```

```
In [ ]: Main_HER_not_null = \
admin_n_encodeable_data[~admin_n_encodeable_data['Main_HER_PRN'].isna()]
Main_HER_not_null['Main_HER_PRN'].describe()
```

```
Out[ ]: count          4100
unique         3863
top           No record found
freq            85
Name: Main_HER_PRN, dtype: object
```

Main HER ID

It is not clear what this is. It looks to be a secondary unique reference ID. There are 205 unique values which contain both categorical and numeric data. It should be noted that there is at least one duplicate. Null values will be resolved in [Admin Encodable Data - Resolve Null Values](#).

```
In [ ]: admin_n_encodeable_data[admin_n_encodeable_data['Main_HER_ID'].notnull().head()]
```

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_MapSheet	Main_NMR_ID	Main_SM	Main_
0	EN	England	Herefordshire	MHE413	910	SO 53 SW 1	110371	1001754	
1	EN	England	Herefordshire	MHE52	344	SO 56 SW 3	110884	1007316	
2	EN	England	Herefordshire	MHE411	908	SO 53 NE 2	110051	1003534	
3	EN	England	Herefordshire	MHE818	1639	SO 47 SW 2	108810	1011016	
4	EN	England	Herefordshire	MHE435	932	SO 74 SE 3	113786	1001792	

```
In [ ]: Main_HER_not_null = admin_n_encodeable_data[~admin_n_encodeable_data['Main_HER_ID'].isna()]
Main_HER_not_null['Main_HER_ID'].describe()
```

```
Out[ ]: count          205
unique         204
top           MNN11486
freq            2
Name: Main_HER_ID, dtype: object
```

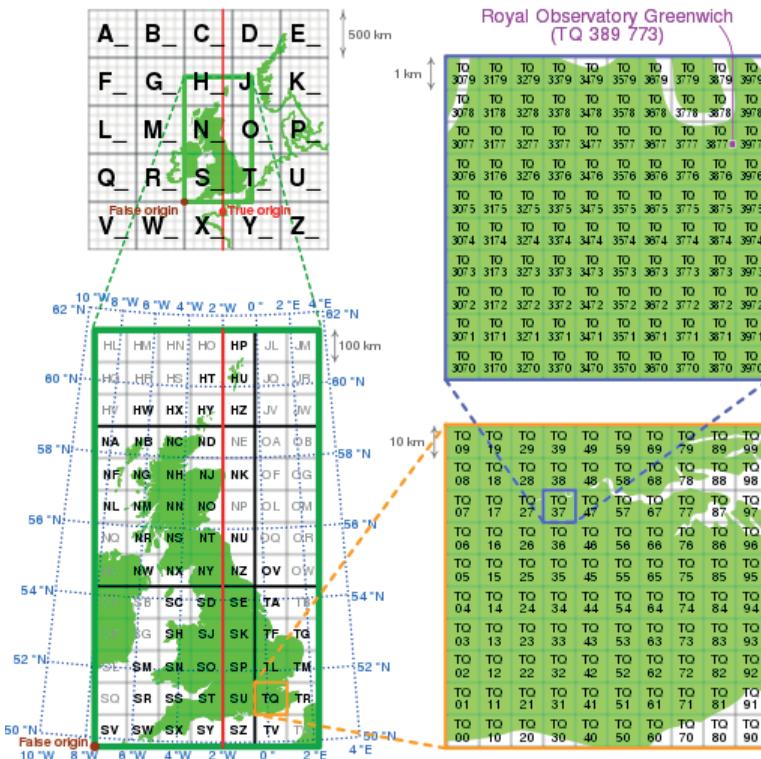
```
In [ ]: admin_n_data[admin_n_data['Main_HER_ID'] == 'MNN11486']
```

	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_HER	Main_HER_PRN
751	752	EN	England	EN0773 Borough Hill 1, Northamptonshire	Borough Hill 1	Borough Hill	Northamptonshire	631/4
752	753	EN	England	EN0774 Borough Hill 2, Northamptonshire	Borough Hill 2	Borough Hill, Northern	Northamptonshire	631/4

Main NMR Mapsheet

The National Monuments Record (NMR) Mapsheets contains concatenated information relating to the Ordnance Survey mapsheet and the NMR site number. It also contains null values. These will be resolved in [Admin Encodable Data - Resolve Null Values](#) and [Admin Encodable Data - Main NMR Mapsheet Replaced](#).

NMR Mapsheets are 5km recordsheets against which a sequence of sites is uniquely numbered. The first four digits relate to the 10km Ordnance Survey mapsheet. The fifth and sixth letters refer to one quarter of the mapsheet. The final numbers are the unique site number on that sheet. See [Location - Location_NGR](#).



The Ordnance Survey National Grid

cmglee, Strebe, MansLughter, Alexrk2 from naturalearthdata, Pethrus and nandhp, *CC BY-SA 3.0*, via Wikimedia Commons

The first record below is on the 10km Ordnance Survey sheet: SO 53;

The NMR 5km Recordsheet is: SO 53 SW (southwest);

The site number on sheet SO 53 SW is one.

```
In [ ]: admin_n_encodeable_data.head()
```

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Mapsheet	Main_NMR_ID	Main_SM	Main_
0	EN	England	Herefordshire	MHE413	910	SO 53 SW 1	110371	1001754	
1	EN	England	Herefordshire	MHE52	344	SO 56 SW 3	110884	1007316	
2	EN	England	Herefordshire	MHE411	908	SO 53 NE 2	110051	1003534	
3	EN	England	Herefordshire	MHE818	1639	SO 47 SW 2	108810	1011016	
4	EN	England	Herefordshire	MHE435	932	SO 74 SE 3	113786	1001792	

Main NMR ID

The 'Main_NMR_ID' is a unique identifier by national record. As there is information from multiple NMRs duplicate values are present. There are null values and there is a mix of data types. See: [Admin Encodable Data - Resolve Null Values](#).

```
In [ ]: temp_NMR_ID = admin_n_encodeable_data.copy()
pd.to_numeric(temp_NMR_ID['Main_NMR_ID'], errors='coerce')
temp_NMR_ID['Main_NMR_ID'].describe()
```

```
Out[ ]: count      3465
unique     3454
top       47004
freq        2
Name: Main_NMR_ID, dtype: object
```

```
In [ ]: admin_n_data[admin_n_data['Main_NMR_ID'] == '47004']
```

Out[]:	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_HER	Main_HER_PRN	Main...
	1483	1484	SC	Scotland	SC1559 The Tappoch, Torwood, Falkirk	The Tappoch, Torwood	NaN	Falkirk Community Trust, Falkirk Sites and Mon...	768
	3553	3554	EN	England	EN3750 Belle Tout, East Sussex	Belle Tout	NaN	East Sussex	MES3054

Main SM

The 'Main_SM' is a unique identifier by Scheduled Monument. As there is information from multiple nations, there are duplicate values. Duplication may also occur where Scheduled Monuments enclose an area that includes multiple, smaller, individually discrete, sites. There are null values and there is a mix of data types. See: [Admin Encodable Data - Resolve Null Values](#).

```
In [ ]: temp_Main_SM = admin_n_encodeable_data.copy()
pd.to_numeric(temp_Main_SM['Main_SM'], errors='coerce')
temp_Main_SM['Main_SM'].describe()
```

```
Out[ ]: count    2282
unique    2251
top     13032
freq      4
Name: Main_SM, dtype: object
```

```
In [ ]: admin_n_data[admin_n_data['Main_SM'] == '13032']
```

Out[]:	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_HER	Main_HER_PRN	Main...
	3522	3523	SC	Scotland	SC3716 Dunsapie, City of Edinburgh	Dunsapie	Edinburgh, Holyrood Park, Dunsapie; The King's...	City of Edinburgh	No record found
	3523	3524	SC	Scotland	SC3717 Salisbury Crags, Edinburgh, City of Edi...	Salisbury Crags, Edinburgh	Edinburgh, Holyrood Park, Salisbury Crags; Que...	City of Edinburgh	No record found
	3525	3526	SC	Scotland	SC3719 Arthur's Seat, City of Edinburgh	Arthur's Seat	Edinburgh, Holyrood Park, Arthur's Seat; Queen...	City of Edinburgh	No record found
	3526	3527	SC	Scotland	SC3720 Samson's Ribs, Arthur's Seat, City of E...	Samson's Ribs, Arthur's Seat	Edinburgh, Holyrood Park, Arthur's Seat, Samso...	City of Edinburgh	No record found

Main Boundary

This 'Yes/No' field indicates if the hillfort is on a boundary. Further detail on boundaries can be found in Part 3: Boundary & Dating https://colab.research.google.com/drive/1dMKByCmq33hjniGZImBjUYUj785_71CT?usp=sharing.

```
In [ ]: admin_n_encodeable_data['Main_Boundary'].describe()
```

```
Out[ ]: count    4147
unique      2
top        No
freq     3721
Name: Main_Boundary, dtype: object
```

```
In [ ]: pd.unique(admin_n_encodeable_data['Main_Boundary'])
```

```
Out[ ]: array(['No', 'Yes'], dtype=object)
```

```
In [ ]: admin_n_encodeable_data['Main_Boundary'].value_counts()
```

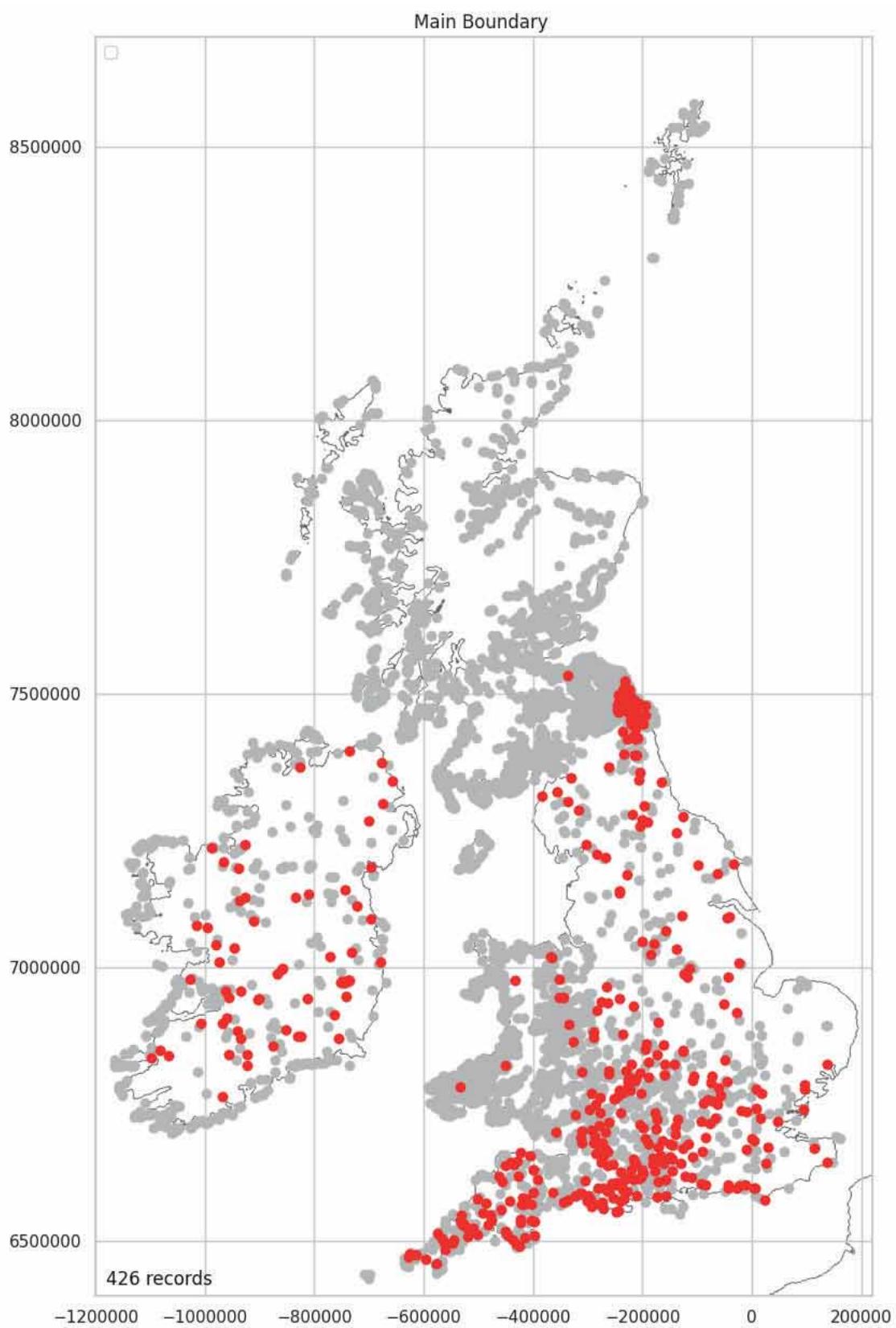
```
Out[ ]: No    3721
Yes   426
Name: Main_Boundary, dtype: int64
```

Main Boundary Mapped

The map below shows that there is a recording bias in this feature. There is only one record in Scotland and very few in Wales and Northern Ireland. The majority are in England and even here, the data is patchy as can be seen by the void in the data, south of the River Severn, toward the western end of the Mendips.

```
In [ ]: location_data = load_location(hillforts_data)
location_admin_data = pd.merge(location_data, \
                                admin_encodeable_data, left_index=True, \
                                right_index=True)
```

```
In [ ]: main_boundary_yes = plot_over_grey(location_admin_data, 'Main_Boundary', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

10.27%

Admin Encodable Data - Resolve Null Values

As mentioned in [Missing Data](#), 'NA' will be used to replace missing categorical values and -1 will be used to replace numeric. Before this can be done, the features are first reviewed to confirm they do not already contain these values. It is important that where null values have been replaced, they can still be identified and not be confused with existing data.

Three of the features are categorical and three are numeric. A different strategy will be required to resolve each.

```
In [ ]: cat_update_list = ['Main_HER', 'Main_HER_PRN', 'Main_NMR_Mapsheet']
num_update_list = ['Main_HER_ID', 'Main_NMR_ID', 'Main_SM']
```

```
In [ ]: test_cat_list_for_NA(admin_n_encodeable_data, cat_update_list)
```

```
Main_HER 0
Main_HER_PRN 0
Main_NMR_Mapsheet 0
```

The output above shows there are no records containing 'NA' in the selected categorical features.

```
In [ ]: test_num_list_for_minus_one(admin_n_encodeable_data, num_update_list)
```

```
Main_HER_ID 0
Main_NMR_ID 0
Main_SM 0
```

Again, the output shows there are no records with a value of -1 in the selected numeric features. The null values can now be replaced.

```
In [ ]: admin_n_encodeable_data = \
update_cat_list_for_NA(admin_n_encodeable_data, cat_update_list)
admin_n_encodeable_data = \
update_num_list_for_minus_one(admin_n_encodeable_data, num_update_list)
```

```
In [ ]: admin_n_encodeable_data_review = test_numeric(admin_n_encodeable_data)
admin_n_encodeable_data_review
```

	Feature	Entries	Numeric	Non-Numeric	Null
0	Main_Country_Code	4147	0	4147	0
1	Main_Country	4147	0	4147	0
2	Main_HER	4147	0	4147	0
3	Main_HER_PRN	4147	2095	2052	0
4	Main_HER_ID	4147	106	4041	0
5	Main_NMR_Mapsheet	4147	0	4147	0
6	Main_NMR_ID	4147	3410	737	0
7	Main_SM	4147	1785	2362	0
8	Main_Boundary	4147	0	4147	0

There are no longer any null values in the admin encodable data. The sample below shows an extract where the null values have been updated to 'NA' or -1.

```
In [ ]: admin_n_encodeable_data[admin_n_encodeable_data['Main_HER']=='NA']
```

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Mapsheet	Main_NMR_ID	Main_SM	Main_Boundary
1141	SC	Scotland	NA	No record found	-1	NS 76 SE 2	45784	-1	
1375	SC	Scotland	NA	No record found	-1	NS 57 NE 27	44425	1749	
1401	SC	Scotland	NA	No record found	-1	NS 67 NW 6	45194	1734	
1405	SC	Scotland	NA	No record found	-1	NS 67 SW 3	45262	-1	
1420	SC	Scotland	NA	No record found	-1	NS 77 NW 28	45894	90008	
1423	SC	Scotland	NA	No record found	-1	NS 77 SE 3	45924	-1	
4145	SC	Scotland	NA	NA	-1	NH 45 SE 57	339438.0	-1	

```
In [ ]:
```

Review Admin Data Split

The following function is used to confirm all features from the original Admin Data are present, and not duplicated, in the numeric, text and encodable data packages.

```
In [ ]: review_data_split(admin_data, admin_numeric_data, admin_text_data, admin_encodable_data)
```

Data split good.

Restructure Admin Encodable Features Containing Mixed Data Formats

Four features contain a mix of numeric and non-numeric data. To be encodable, these need to be restructured into a format that can be encoded:

1. Main_HER_PRN
2. Main_HER_ID
3. Main_NMR_ID
4. Main_SM

```
In [ ]: reformat_list = ['Main_HER_PRN', 'Main_HER_ID', 'Main_NMR_ID', 'Main_SM']
```

The data in these features is split into text and numeric components and saved into new features with suffixes '_num' and '_txt'.

```
In [ ]: def split_text_num(data, feature):  
    new_data = data.copy()  
    new_data[feature + '_num'] = new_data[feature].str.extract('(\d+)')  
    new_data[feature + '_txt'] = new_data[feature].str.extract('([a-zA-Z]+)')  
    return new_data
```

```
In [ ]: admin_encodeable_data_plus = admin_encodeable_data.copy()  
num_update_list = []  
txt_update_list = []  
for col in reformat_list:  
    admin_encodeable_data_plus = \  
        split_text_num(admin_encodeable_data_plus, col)  
    admin_encodeable_data_plus = \  
        update_cat_list_for_NA(admin_encodeable_data_plus, [col + '_txt'])  
    admin_encodeable_data_plus = \  
        update_num_list_for_mius_one(admin_encodeable_data_plus, [col + '_num'])
```

The four original features are deleted, as the information they contain is stored in the new features.

```
In [ ]: for col in reformat_list:  
    admin_encodeable_data_plus = admin_encodeable_data_plus.drop([col], axis=1)
```

```
In [ ]: admin_encodeable_data_plus.head(6)
```

```
Out[ ]:
```

	Main_Country_Code	Main_Country	Main_HER	Main_NMR_Mapsheet	Main_Boundary	Main_HER_PRN_num	Main_HER_PRN_txt	Main
0	EN	England	Herefordshire	SO 53 SW 1	No	413	MHE	
1	EN	England	Herefordshire	SO 56 SW 3	No	52	MHE	
2	EN	England	Herefordshire	SO 53 NE 2	No	411	MHE	
3	EN	England	Herefordshire	SO 47 SW 2	No	818	MHE	
4	EN	England	Herefordshire	SO 74 SE 3	Yes	435	MHE	
5	EN	England	Herefordshire	NA	No	21869	MHE	

```
In [ ]: admin_encodeable_data_plus.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Main_Country_Code 4147 non-null   object  
 1   Main_Country      4147 non-null   object  
 2   Main_HER          4147 non-null   object  
 3   Main_NMR_Mapsheet 4147 non-null   object  
 4   Main_Boundary     4147 non-null   object  
 5   Main_HER_PRN_num  4147 non-null   object  
 6   Main_HER_PRN_txt  4147 non-null   object  
 7   Main_HER_ID_num   4147 non-null   object  
 8   Main_HER_ID_txt   4147 non-null   object  
 9   Main_NMR_ID_num   4147 non-null   object  
 10  Main_NMR_ID_txt  4147 non-null   object  
 11  Main_SM_num       4147 non-null   object  
 12  Main_SM_txt       4147 non-null   object  
dtypes: object(13)
memory usage: 421.3+ KB

```

The strategy adopted here is simplistic in that it separates out the first numeric and categorical components into two new features. It assumes the data in the source feature follows a certain standard. In most cases it does but, where it does not the output will only be a sub-sample of the original. As these features relate to unique identifiers and will, most likely, be dropped before they are used by data science tools, it is not worth any further effort to improve the quality of the output.

An example of 'Main_HER_ID' records with an irregular data structure. In this case the records contain a forward slash.

```
In [ ]: pd.DataFrame(Main_HER_not_null[Main_HER_not_null['Main_HER_ID'].str.contains('/', regex=False)]['Main_HER_ID'].head())
```

```
Out[ ]: Main_HER_ID
497 10208/MNA124826 (NT)
3003 971/1
3004 1007/2
3005 1732/1/1-3
3006 970/1
```

Restructured data for record 497.

```
In [ ]: admn_n_encodeable_data_plus.iloc[[497]][['Main_HER_ID_num', 'Main_HER_ID_txt']]
```

```
Out[ ]: Main_HER_ID_num  Main_HER_ID_txt
497          10208          MNA
```

Record 497 shows how there has been some loss of information during the restructuring of this specific record. The restructured data contains only the first numeric and text components. In this case, the record does not have the same structure as seen throughout the majority of records in this feature, so some data has been lost. As mentioned above, unique identifiers are unlikely to provide insight into the interpretation of hillforts and no further effort will be made to reduce the small amount of data loss identified in restructuring these features.

A second example of 'Main_HER_ID' where the data does not follow the structure seen throughout most other records in this feature. In this case where multiple identifiers are separated by commas.

```
In [ ]: pd.DataFrame(Main_HER_not_null[Main_HER_not_null['Main_HER_ID'].str.contains(',', regex=False)]['Main_HER_ID'].head())
```

```
Out[ ]: Main_HER_ID
113 2303627, 2402293, 2303627, 2744876, 2303627
116 2303474, 2399905, 2303474, 2399906, 2303474
```

Output table showing the split data for index 113. Only the first number has been retained and all identifiers after the comma have been lost. It is important to be aware of the data loss so that an assessment can be made on how significant this loss is. In this case the loss is deemed to be insignificant as this feature is unlikely to be of use in machine learning. If, at a later stage, this information is thought to be important, the source data remains accessible and unchanged.

```
In [ ]: admn_n_encodeable_data_plus.iloc[[113]][['Main_HER_ID_num', 'Main_HER_ID_txt']]
```

```
Out[ ]: Main_HER_ID_num Main_HER_ID_txt
113      2303627      NA
```

Main NMR Mapsheet Replaced

The 'Main_NMR_Mapsheets' follows a formula with the first six characters identifying a 5km square Ordnance Survey mapsheet (know as a NMR Recordsheet) and the final numbers identifying the site number on that sheet. Not all records have a 'Main_NMR_Mapsheets'.

The 'Main_NMR_Mapsheets' data has been entered in a structured way with spaces separating the numeric and text components of the reference. The spaces can be used to split this data into new features: 'OS_MapSheet' and 'Site_no' (site number). 'NA' has already been added where there is no 'Main_NMR_Mapsheets' recorded and -1 is used for null 'Site_no' records.

```
In [ ]: admn_n_encodeable_data_plus[['Main_NMR_MapSheet']].head(6)
```

```
Out[ ]: Main_NMR_MapSheet
0      SO 53 SW 1
1      SO 56 SW 3
2      SO 53 NE 2
3      SO 47 SW 2
4      SO 74 SE 3
5          NA
```

```
In [ ]: admn_n_encodeable_data_plus[['pt1', 'pt2']] = \
admn_n_encodeable_data_plus[['Main_NMR_MapSheet']].str.split(" ", 1, expand=True)
admn_n_encodeable_data_plus[['pt2', 'pt3']] = \
admn_n_encodeable_data_plus[['pt2']].str.split(" ", 1, expand=True)
admn_n_encodeable_data_plus[['pt3', 'Site_no']] = \
admn_n_encodeable_data_plus[['pt3']].str.split(" ", 1, expand=True)
admn_n_encodeable_data_plus['OS_MapSheet'] = \
admn_n_encodeable_data_plus[['pt1']] + \
admn_n_encodeable_data_plus[['pt2']] + admn_n_encodeable_data_plus[['pt3']]
admn_n_encodeable_data_plus = \
admn_n_encodeable_data_plus.drop(['pt1', 'pt2', 'pt3', 'Main_NMR_MapSheet'], axis=1)
admn_n_encodeable_data_plus = \
update_cat_list_for_NA(admn_n_encodeable_data_plus, ['OS_MapSheet'])
admn_n_encodeable_data_plus = \
update_num_list_for_mi_nus_one(admn_n_encodeable_data_plus, ['Site_no'])
admn_n_encodeable_data_plus[['OS_MapSheet', 'Site_no']].head(6)
```

```
<ipython-input-187-dfc8dfeffef>:2: FutureWarning: In a future version of pandas all arguments of StringMethods.split except for the argument 'pat' will be keyword-only.
    admn_n_encodeable_data_plus[['Main_NMR_MapSheet']].str.split(" ", 1, expand=True)
<ipython-input-187-dfc8dfeffef>:4: FutureWarning: In a future version of pandas all arguments of StringMethods.split except for the argument 'pat' will be keyword-only.
    admn_n_encodeable_data_plus[['pt2']].str.split(" ", 1, expand=True)
<ipython-input-187-dfc8dfeffef>:6: FutureWarning: In a future version of pandas all arguments of StringMethods.split except for the argument 'pat' will be keyword-only.
    admn_n_encodeable_data_plus[['pt3']].str.split(" ", 1, expand=True)
```

```
Out[ ]: OS_MapSheet Site_no
0      SO53SW      1
1      SO56SW      3
2      SO53NE      2
3      SO47SW      2
4      SO74SE      3
5          NA     -1
```

Drop Duplicate Admin Features

'Main_Country_Code' and 'Main_Country' contain the same information in different formats. 'Main_Country' will be deleted.

Duplicating information means more features (also known as dimensions). Adding dimensions can lead to clusters in the data being more difficult to identify as the data gets pulled further apart. In the example below left, the data is 0.5 units apart in two dimensions. In the right image, the same x, y coordinates have had a z dimension added and now the data is 1.12 units apart.

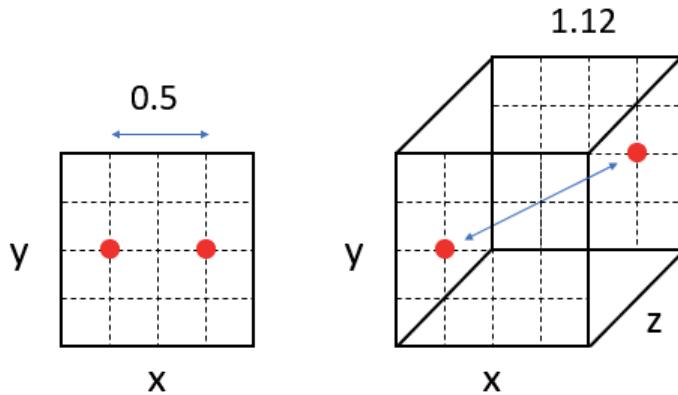


Figure showing how adding a dimension can increase the distance between data points and clusters
Mike Middleton *CC BY-SA 3.0*

```
In [ ]: admin_encodeable_data_plus = admin_encodeable_data_plus.drop(['Main_Country'], axis=1)
admin_encodeable_data_plus.head(6)
```

	Main_Country_Code	Main_HER	Main_Boundary	Main_HER_PRN_num	Main_HER_PRN_txt	Main_HER_ID_num	Main_HER_ID_txt	Main
0	EN	Herefordshire	No	413	MHE	910	NA	
1	EN	Herefordshire	No	52	MHE	344	NA	
2	EN	Herefordshire	No	411	MHE	908	NA	
3	EN	Herefordshire	No	818	MHE	1639	NA	
4	EN	Herefordshire	Yes	435	MHE	932	NA	
5	EN	Herefordshire	No	21869	MHE	-1	NA	

Admin Data Packages

A single list containing the pre-processed Admin data packages.

```
In [ ]: admin_data_list = \
[admin_numeric_data, admin_text_data, admin_encodeable_data_plus]
```

Download Admin Data

The download will only function if selected by the user. See: [User Settings](#).

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(admin_data_list, 'Admin_package')
```

Location Data

The location data contains the information needed to locate and plot the data. The section contains coordinates in multiple grid projections, information on the projections and information on parish and county. As with the Admin Data, the location data will be split into numeric, text and encodable data packages.

```
In [ ]: location_features = [
'Main_Coordinate_System',
'Main_X',
'Main_Y',
'Location_NGR',
'Location_X',
'Location_Y',
```

```
'Location_Longitude',
'Location_Latitude',
'Location_Current_County',
'Location_Historic_County',
'Location_Current_Parish']

location_data = hillforts_data[location_features]
location_data.head()
```

Out[]:

	Main_Coordinate_System	Main_X	Main_Y	Location_NGR	Location_X	Location_Y	Location_Longitude	Location_Latitude	Location_Current_County
0	OSGB36	350350	233050	SO 503330	-303295	6798973	-2.724548	51.993628	Henbury
1	OSGB36	354700	260200	SO 547602	-296646	6843289	-2.664819	52.238082	Henbury
2	OSGB36	358700	238900	SO 587389	-289837	6808611	-2.603651	52.046907	Henbury
3	OSGB36	340000	272400	SO 400724	-320850	6862993	-2.882242	52.346344	Henbury
4	OSGB36	376000	240000	SO 760400	-261765	6810587	-2.351472	52.057819	Henbury

Only 'Location_NGR' contains null values.

In []: location_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Main_Coordinate_System    4147 non-null   object 
 1   Main_X                  4147 non-null   int64  
 2   Main_Y                  4147 non-null   int64  
 3   Location_NGR             3650 non-null   object 
 4   Location_X                4147 non-null   int64  
 5   Location_Y                4147 non-null   int64  
 6   Location_Longitude        4147 non-null   float64 
 7   Location_Latitude         4147 non-null   float64 
 8   Location_Current_County  4147 non-null   object 
 9   Location_Historic_County 4147 non-null   object 
 10  Location_Current_Parish  4127 non-null   object 
dtypes: float64(2), int64(4), object(5)
memory usage: 356.5+ KB
```

Location Numeric Data

The numeric data comprises the easting and northing information for three grid projections. 'Location_X' and 'Location_Y' are the coordinates used throughout this study, when producing maps. The coastal outline, topographic background and distribution plots of the location data spread have only been produced for use with this projection.

The three packages of coordinates contain the same information transformed by three different grid projections. To reduce the number of dimensions and to remove duplicate information, only 'Location_X' and 'Location_Y' will be retained in the reprocessed data.

In []: location_numeric_features = [

```
'Main_X',
'Main_Y',
'Location_X',
'Location_Y',
'Location_Longitude',
'Location_Latitude']
```

```
location_numeric_data = location_data[location_numeric_features].copy()
location_numeric_data.head()
```

Out[]:

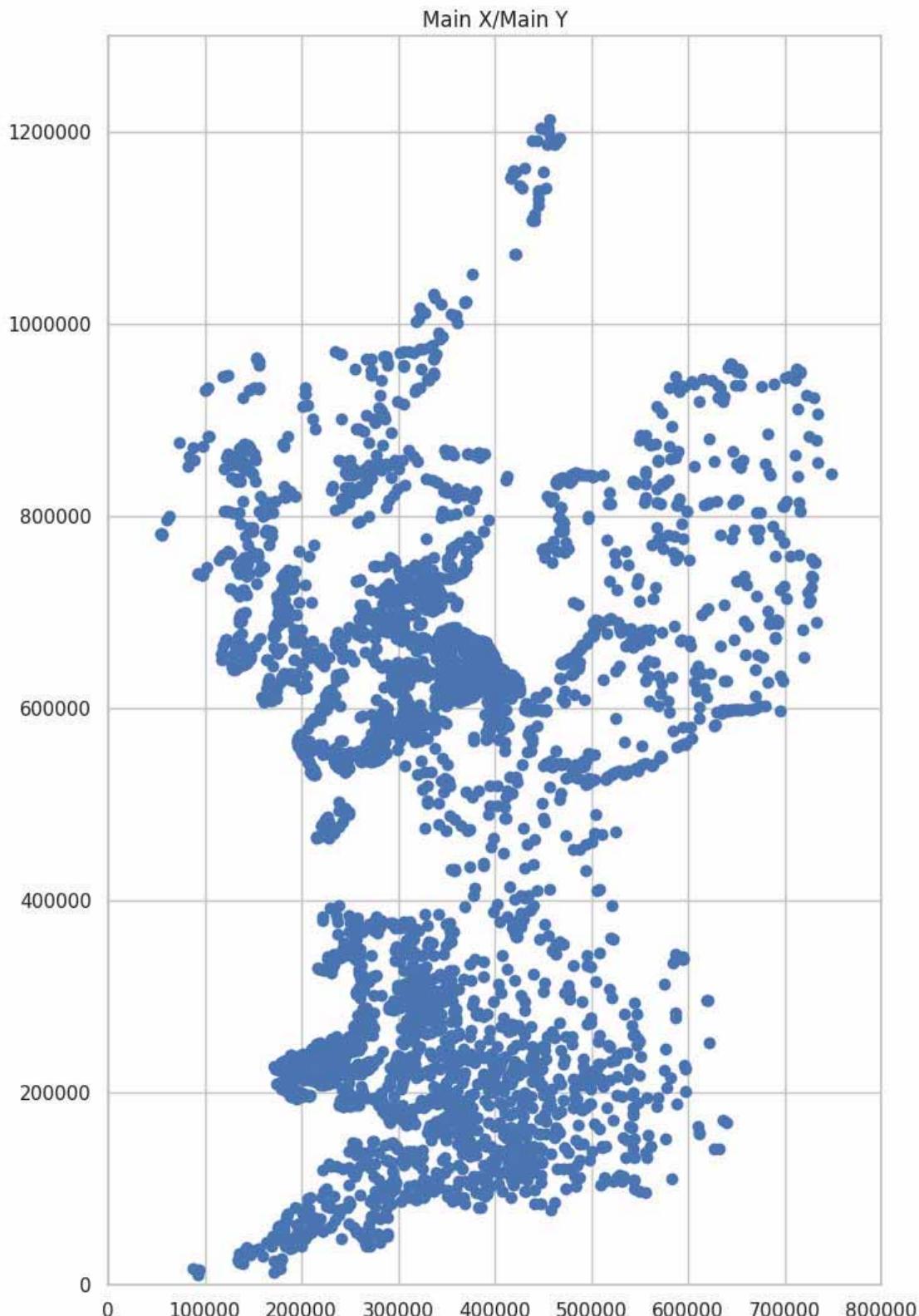
	Main_X	Main_Y	Location_X	Location_Y	Location_Longitude	Location_Latitude
0	350350	233050	-303295	6798973	-2.724548	51.993628
1	354700	260200	-296646	6843289	-2.664819	52.238082
2	358700	238900	-289837	6808611	-2.603651	52.046907
3	340000	272400	-320850	6862993	-2.882242	52.346344
4	376000	240000	-261765	6810587	-2.351472	52.057819

Main_X / Main_Y Mapped

'Main_X' and 'Main_Y' store the coordinates for each record based on the local grid. See: [Location - Main Coordinate System](#). This means that when plotting the atlas data, using 'Main_X' and 'Main_Y', the Irish and UK grids plot over one another.

The figure size is proportionate to the 'Main_X' / 'Main_Y' ranges i.e. on the x-axis the data range goes from 0 to 8,000,000 so the figure size chosen for the x-axis is 8. The same principle has been applied to the y-axis and this method has been used in the production of all future maps in this study.

```
In [ ]: plot_main_xy(location_numeric_data)
```



See: [Location - Main Coordinate System](#)

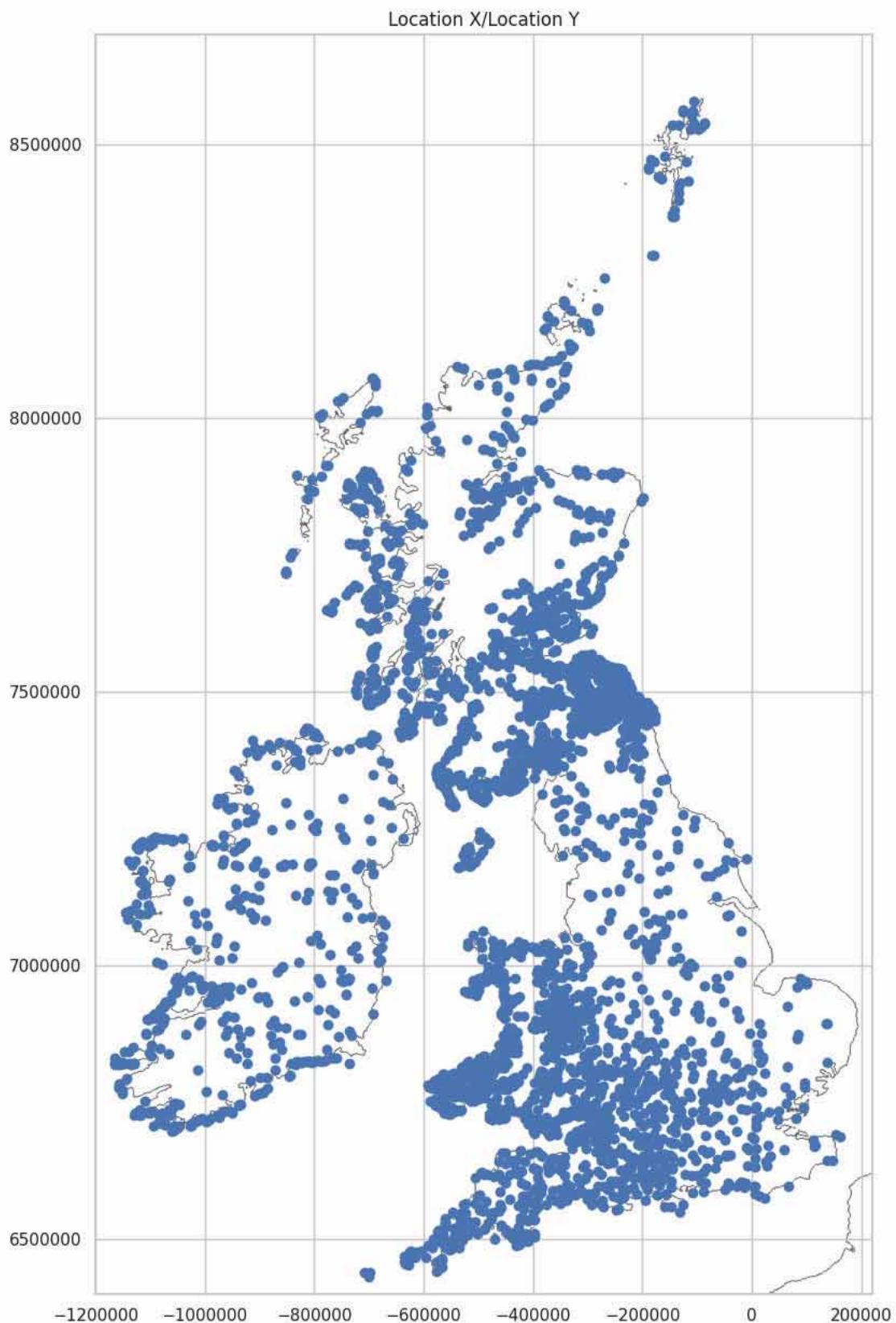
Location_X / Location_Y Mapped

'Location_X'/'Location_Y' stores the location data in the projection WGS84/ Pseudo-Mercator (also known as [EPSG:3857](#)).

As with the figure above, the figure size is proportionate to the 'Location_X' / 'Location_Y' data ranges.

The 'Location_X' & 'Location_Y' coordinates plot all the data in the hillforts atlas against the same grid meaning that Ireland now plots in its correct position relative the UK.

```
In [ ]: plot_location_xy(location_numeric_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

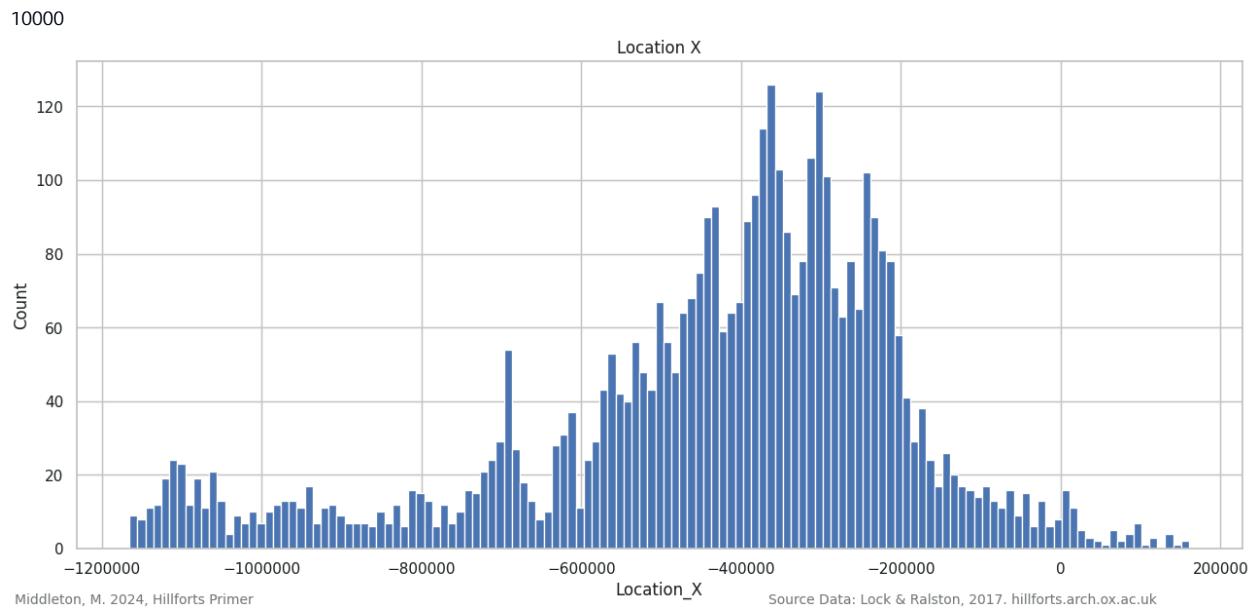
Location_X Data Plotted - Easting

Within the 'Location_X' plot there is a single large peak and a number of smaller peaks, rising out of the main peak. The data is plotted using 10km 'bins' (counts per 10km). These will be examined in more detail below.

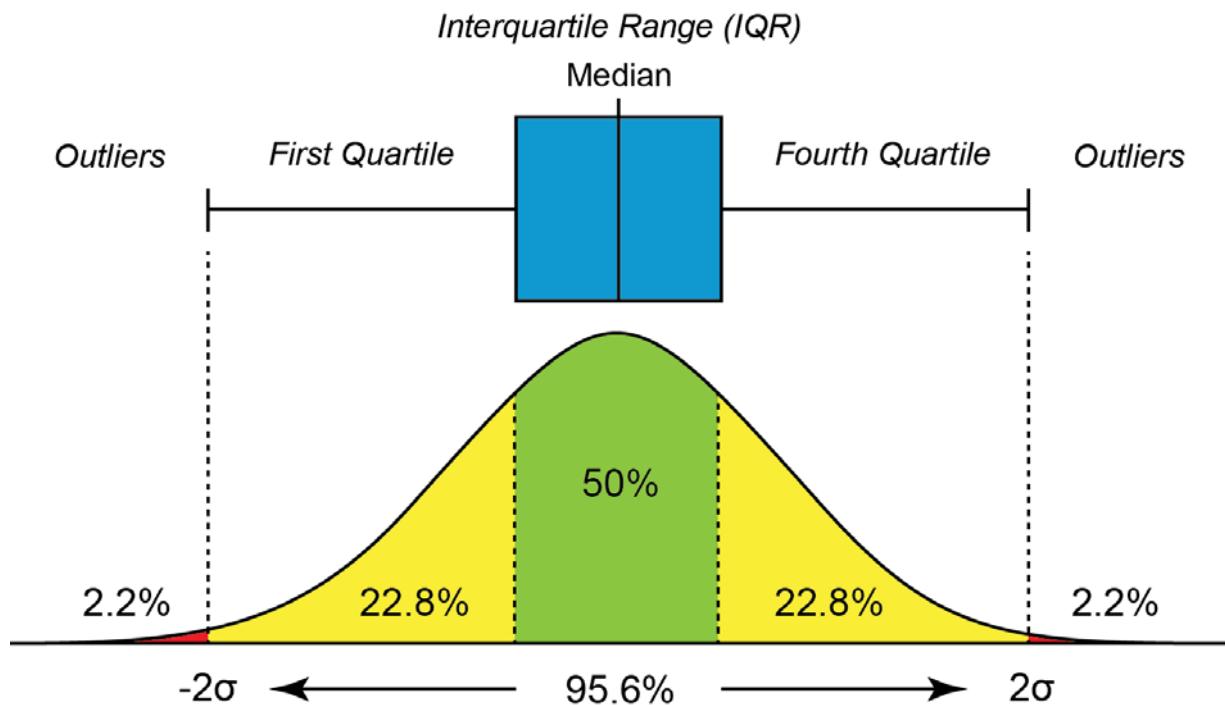
```
In [ ]: location_numeric_data['Location_X'].describe()
```

```
Out[ ]: count    4.147000e+03
         mean    -4.387439e+05
         std     2.493718e+05
         min    -1.165487e+06
         25%   -5.398565e+05
         50%   -3.839160e+05
         75%   -2.779725e+05
         max    1.606020e+05
Name: Location_X, dtype: float64
```

```
In [ ]: plot_histogram(location_numeric_data['Location_X'], \
                      'Location_X', 'Location_X', 10000)
```



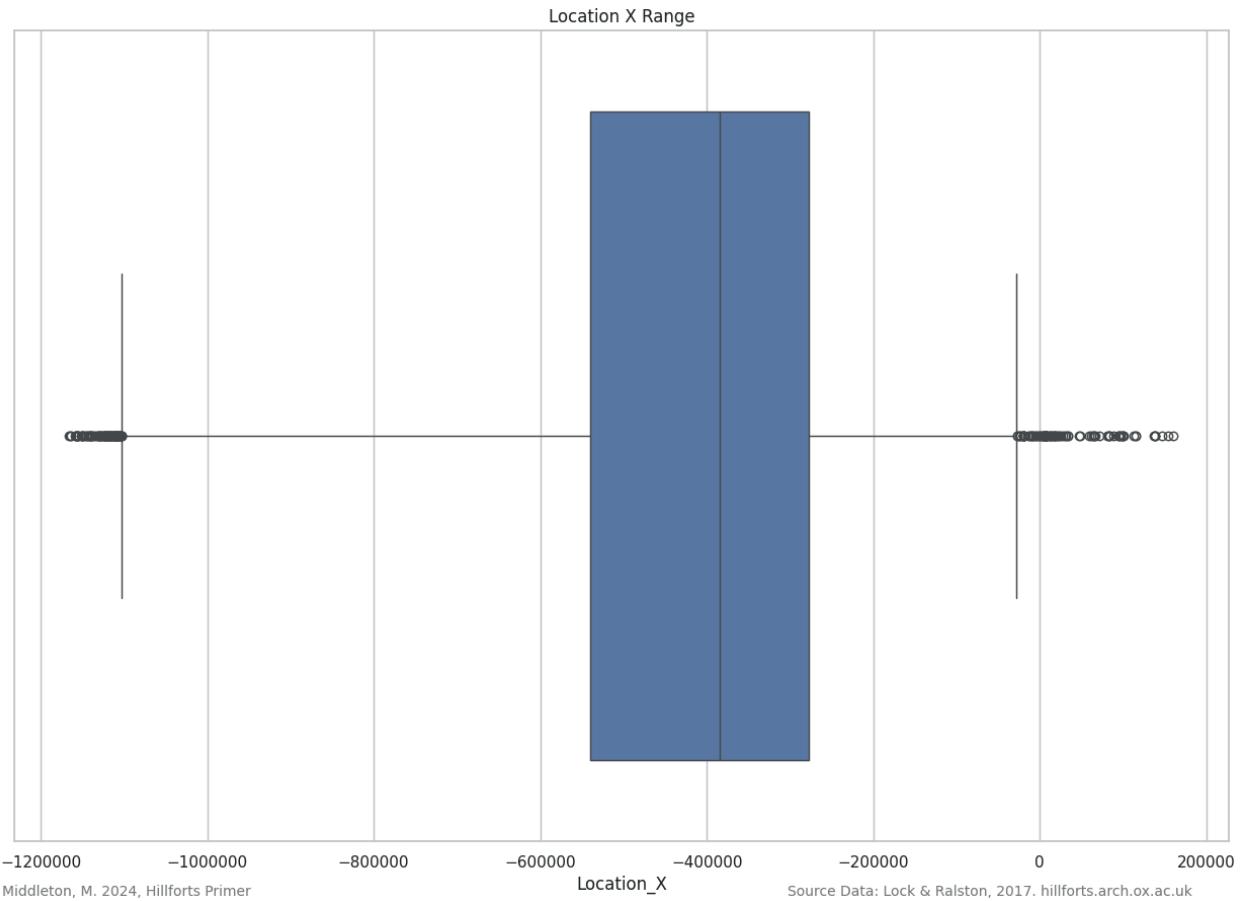
A boxplot shows the distribution of data. The central blue box shows the interquartile range (IQR), within which sits quarters two & three of the data (the mid 50%). The whiskers, ending in the black vertical lines, indicate the extent of all but 4.4% of the remaining data or the equivalent of two standard deviations. Black dots, beyond the whiskers, show outliers which sit in the remaining 4.4% of the data. The vertical black line in the blue box indicates the location of the median value.



*A boxplot showing its relationship to a normal distribution of data
Mike Middleton *CC BY-SA 3.0**

Unsurprisingly, along the 'Location_X' axis, most of the records are spread across the UK mainland, the boxplot shows there is far less data to the west. The number of hillforts in Ireland can be seen to be considerably lower than the main peak over the UK mainland. Outliers comprise Irish data in the west and records across the eastern lowlands of England. See: [Density Map showing Extent of Boxplot](#).

```
In [ ]: location_X_data = plot_data_range(location_numeric_data['Location_X'], \
                                         'Location_X', "h")
```



Location_Y Data Plotted - Northing

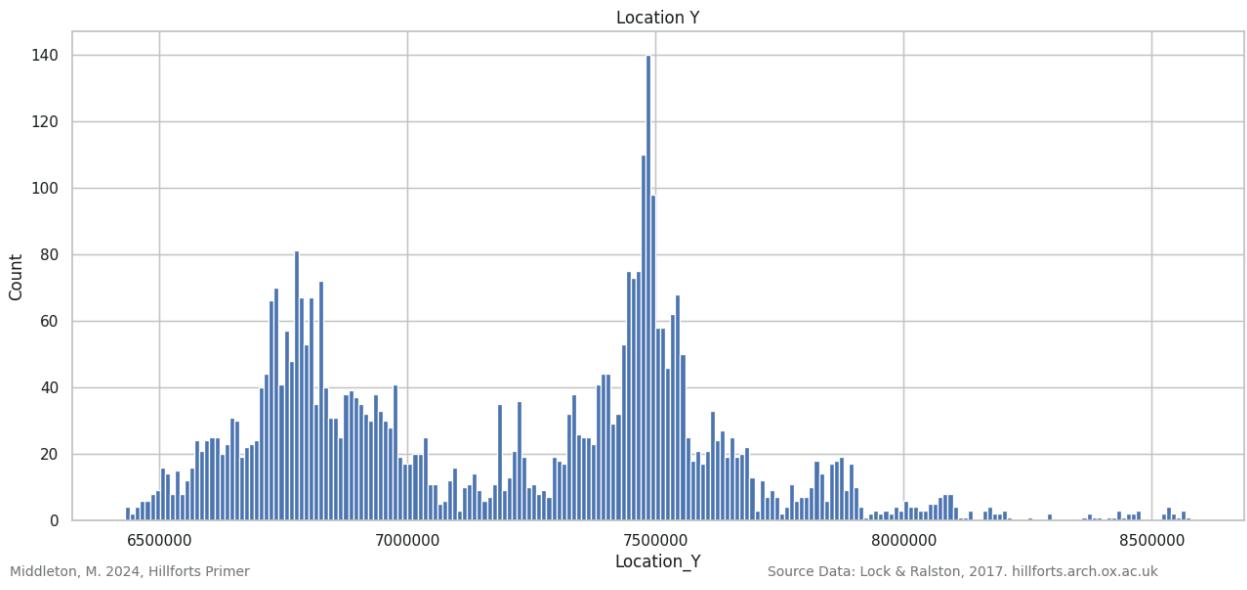
Within the 'Location_Y' data, there are two large peaks. These will be explored in more detail in [Location - Add Density Feature - North & South Density](#). Again, the data is plotted using 10km 'bins'.

```
In [ ]: location_numeric_data['Location_Y'].describe()
```

```
Out[ ]: count    4. 147000e+03
mean     7. 194890e+06
std      4. 145320e+05
min     6. 430971e+06
25%     6. 804457e+06
50%     7. 243520e+06
75%     7. 494886e+06
max     8. 578090e+06
Name: Location_Y, dtype: float64
```

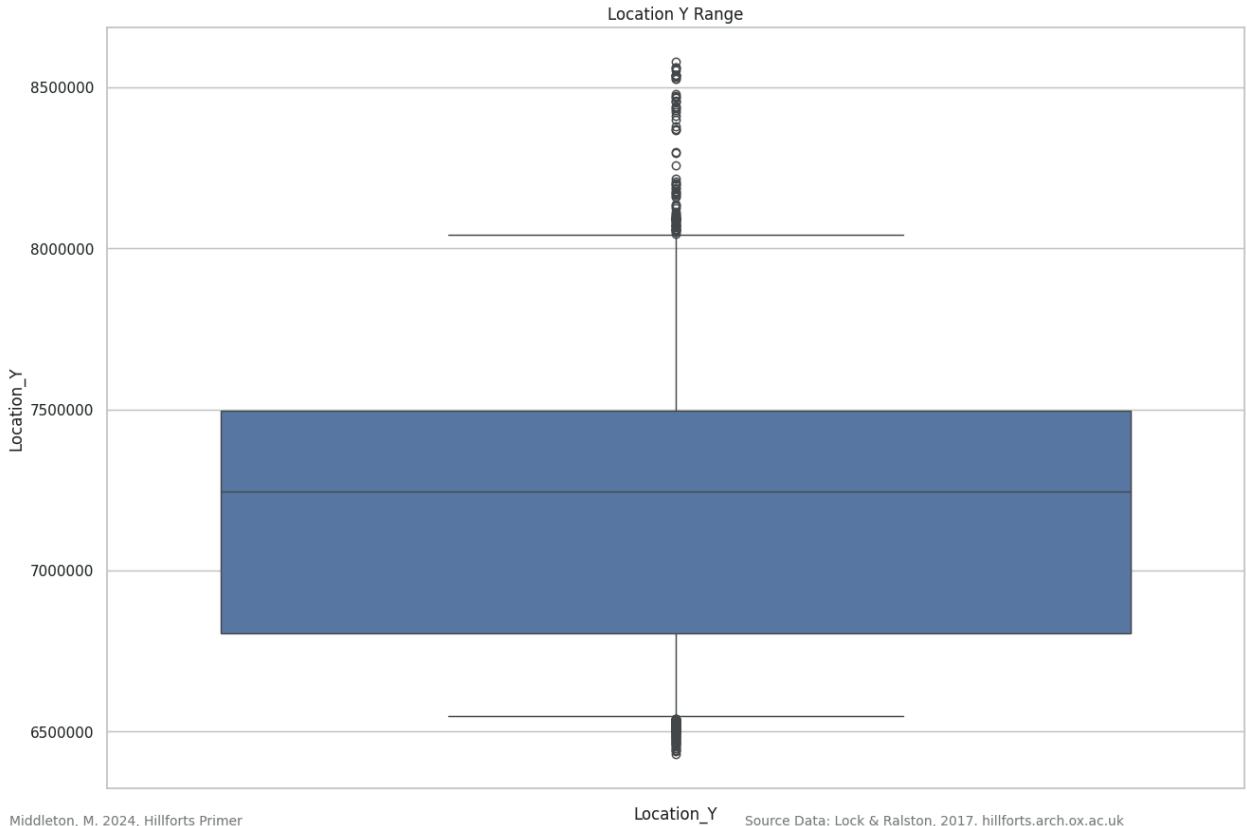
```
In [ ]: plot_histogram(location_numeric_data['Location_Y'], \
                      'Location_Y', 'Location_Y', 10000)
```

10000



The 'Location_Y' boxplot shows a concentration of data toward the centre of the data range with a large spread of outliers across the northern isles. Having seen the distribution spread across two peaks, in the plot above, it is clear the boxplot is not showing the subtlety of the distribution of the data. See: [Density Map showing Extent of Boxplot](#) and [North / South Density Split](#).

```
In [ ]: location_Y_data = \
plot_data_range(location_numeric_data['Location_Y'], 'Location_Y')
```

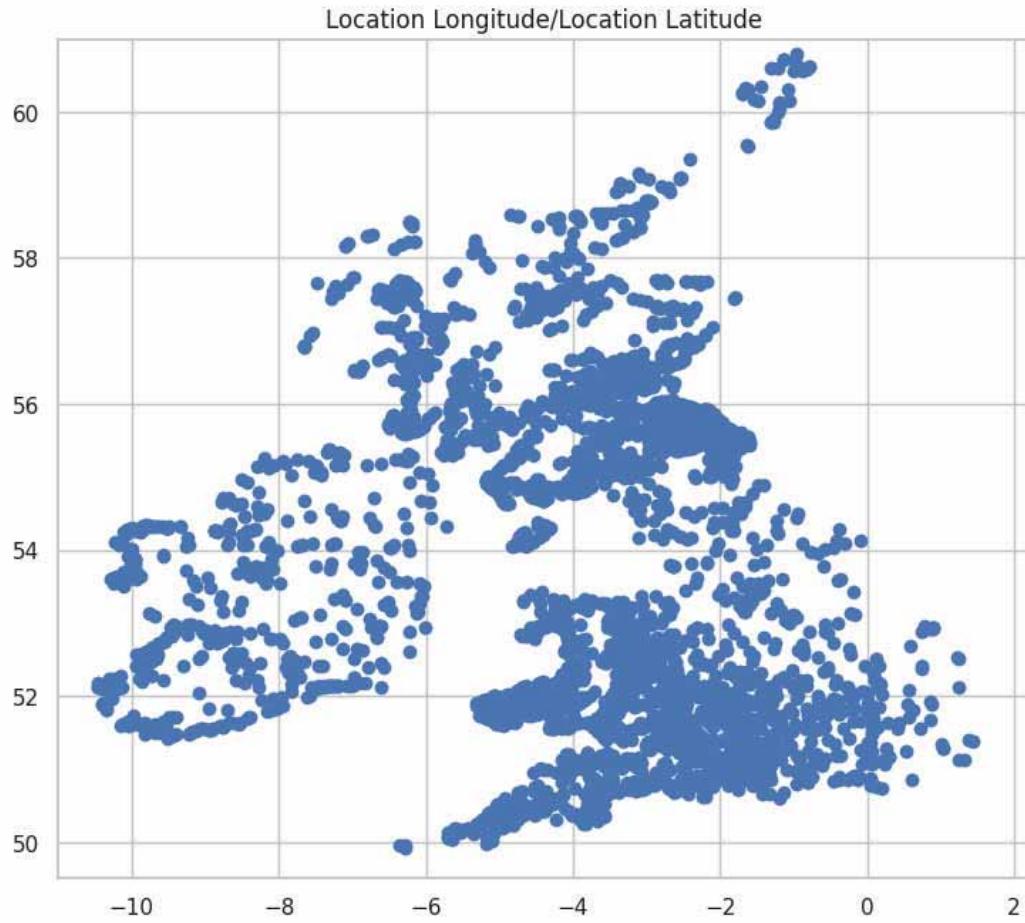


Longitude / Latitude Mapped

'Longitude' / 'Latitude' store the location data in the projection WGS84 (also known as [EPSG:4326](#)).

As above, the figure size is proportionate to the 'Longitude' / 'Latitude' data ranges.

```
In [ ]: plot_LatLong(location_numeric_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Location Text Data

There are no location text data features. All the remaining location features area encodable.

```
In [ ]: location_text_data = pd.DataFrame()
```

Location Encodable Data

There are five encodable location features, of which, two contain null values.

```
In [ ]: location_encodable_features = [
    'Main_Coordinate_System',
    'Location_NGR',
    'Location_Current_County',
    'Location_Historic_County',
    'Location_Current_Parish']

location_encodable_data = location_data[location_encodable_features].copy()
location_encodable_data.head()
```

	Main_Coordinate_System	Location_NGR	Location_Current_County	Location_Historic_County	Location_Current_Parish
0	OSGB36	SO 503330	Herefordshire	Herefordshire	Aconbury
1	OSGB36	SO 547602	Herefordshire	Herefordshire	Kimbolton
2	OSGB36	SO 587389	Herefordshire	Herefordshire	Dormington
3	OSGB36	SO 400724	Herefordshire	Herefordshire	Adforton
4	OSGB36	SO 760400	Herefordshire	Herefordshire	Colwall

```
In [ ]: location_encodable_data_review = test_numeric(location_encodable_data)
location_encodable_data_review
```

	Feature	Entries	Numeric	Non-Numeric	Null
0	Main_Coordinate_System	4147	0	4147	0
1	Location_NGR	3650	0	3650	497
2	Location_Current_County	4147	0	4147	0
3	Location_Historic_County	4147	0	4147	0
4	Location_Current_Parish	4127	0	4127	20

Main Coordinate System

The 'Main_X'/'Main_Y' features contain coordinates relating to one of two coordinate systems:

1. Ireland (including Northern Ireland) (IRENET95 also known as [EPSG:2157](#));
2. UK (excluding Northern Ireland) (OSGB36 known as [EPSG:27700](#)).

```
In [ ]: pd.unique(location_encodeable_data['Main_Coordinate_System'])
```

```
Out[ ]: array(['OSGB36', 'IRENET95'], dtype=object)
```

'Main_X' and 'Main_Y' store the grid coordinates based on the local grid for each nation. As seen in [Location - Main_X / Main_Y Mapped](#), plotting the data using the 'Main_X'/'Main_Y' coordinates will cause the two coordinate systems to plot over one another.

As the 'Main_X' and 'Main_Y' coordinates are not being used in this study, the 'Main_Coordinate_System' will be dropped from the processed location data package.

Location_NGR

The NGR (National Grid Reference) is an alternative way to refer to the coordinates in 'Main_X' and 'Main_Y' that are projected against the UK's Ordnance Survey National Grid. See [Admin Data - Main_NMR_Mapsheet](#)

For record 0, in the extract below, the first numbers of 'Main_X' and 'Main_Y' equate to the letters in 'Location_NGR':

300000, 200000 = SO

The second, third and fourth numbers from 'Main_X' and 'Main_Y' are then combined to create the six digit number in the 'Location_NGR':

503, 330 = 503330

As Ordnance survey grid references are 'six-figure', the grid location that can be derived from the 'Location_NGR' for SO 503330 is:

SO 503330 = 350300, 233000

As the derived coordinate end in two zeros on both axis the 'Location_NGR' is only accurate to within 100m.

```
In [ ]: location_data[['Main_X', 'Main_Y', 'Location_NGR']].head()
```

```
Out[ ]:   Main_X  Main_Y  Location_NGR
0    350350  233050    SO 503330
1    354700  260200    SO 547602
2    358700  238900    SO 587389
3    340000  272400    SO 400724
4    376000  240000    SO 760400
```

This feature duplicates coordinate information already held in 'Location_X' and 'Location_Y' and will be dropped from the location data package.

Location Current County

This feature contains a list of 176 modern day counties (top 25 counts are shown). There are no null values.

```
In [ ]: location_encodeable_data['Location_Current_County'].describe()
```

```
Out[ ]: count          4147  
unique         176  
top    Scottish Borders  
freq          408  
Name: Location_Current_County, dtype: object
```

To see more than 25 counties, change the value in the square brackets below and rerun the document. See: [User Settings](#).

```
In [ ]: location_encodeable_data['Location_Current_County'].\nvalue_counts()[:25]
```

```
Out[ ]: Scottish Borders      408  
Dumfries & Galloway     286  
Northumberland        271  
Highland             209  
Argyll & Bute          199  
Powys                 147  
Pembrokeshire        136  
Devon                  89  
East Lothian          89  
Ceredigion           86  
Carmarthenshire      80  
Cornwall              77  
Perth & Kinross       76  
Gwynedd                74  
Shropshire            63  
Cork                   61  
South Lanarkshire    55  
Mayo                   53  
Clare                  51  
Hampshire             50  
Wiltshire              50  
Somerset               48  
Gloucestershire       47  
Fife                   44  
Aberdeenshire         42  
Name: Location_Current_County, dtype: int64
```

Location Historic County

This feature contains a list of 120 historic counties (top 25 counts are shown). There are no null values.

```
In [ ]: location_encodeable_data['Location_Historic_County'].describe()
```

```
Out[ ]: count          4147  
unique         120  
top    Northumberland  
freq          271  
Name: Location_Historic_County, dtype: object
```

To see more than 25 counties, change the value in the square brackets below and rerun the document. See: [User Settings](#).

```
In [ ]: location_encodeable_data['Location_Historic_County'].\nvalue_counts()[:25]
```

```
Out[ ]: Northumberland      271  
Argyll                 201  
Roxburghshire          156  
Pembrokeshire          136  
Berwickshire           129  
Inverness-shire        127  
Dumfriesshire          125  
Peeblesshire           93  
Devon                  89  
East Lothian            89  
Kirkcudbrightshire    89  
Cardiganshire           86  
Perthshire              85  
Carmarthenshire        80  
Cornwall                80  
Glamorgan               80  
Somerset                 74  
Mayo                   73  
Wigtownshire           72  
Montgomeryshire        72  
Cork                   72  
Gloucestershire         71  
Shropshire              63  
Lanarkshire              57  
Caernarfonshire        54  
Name: Location_Historic_County, dtype: int64
```

Location Current Parish

This feature contains a list of 2259 current parishes (top 25 counts are shown).

```
In [ ]: location_encodeable_data['Location_Current_Parish'].describe()
```

```
Out[ ]: count      4127
unique     2259
top       Cavers
freq       21
Name: Location_Current_Parish, dtype: object
```

There are 20 null values.

```
In [ ]: location_encodeable_data['Location_Current_Parish'].isna().sum()
```

```
Out[ ]: 20
```

To see more than 25 parishes, change the value in the square brackets below and rerun the document. See: [User Settings](#).

```
In [ ]: location_encodeable_data['Location_Current_Parish'].value_counts()[:25]
```

```
Out[ ]: Cavers                21
Kildalton And Oa              20
Kirknewton                     19
Col di ngham                   19
Kilninian And Kilmore          18
Peebles                          18
Rerrick                          17
Chatton                           16
Lauder                            15
Ingram                            15
Garval d And Bara               15
Kirkmaiden                      14
Ford                               14
Small Isles                      13
Sni zort                          13
Jedburgh                         12
Kilmore And Kilbride             12
Horncliffe                       12
Hownam                           12
Broughton, Glenholm And Kilbicho 12
Campbel town                      11
Doddington                        11
Whithorn                          11
Farr                               11
Dui rinish                         11
Name: Location_Current_Parish, dtype: int64
```

Review Location Data Split

```
In [ ]: review_data_split(location_data, location_numeric_data, \
                           location_text_data, location_encodeable_data)
```

Data split good.

Drop Features From Location Numeric Data

Restructured Location Numeric Data

As mentioned in [Location Numeric Data](#), the coordinates are saved in multiple projections in the source data. 'Location_X' and 'Location_Y' will be used. All other projections will be deleted.

```
In [ ]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = \
location_numeric_data[location_numeric_data_short_features].copy()
location_numeric_data_short.head()
```

```
Out[ ]:   Location_X  Location_Y
0      -303295    6798973
1     -296646    6843289
2     -289837    6808611
3     -320850    6862993
4     -261765    6810587
```

Drop Features From Location Encodable Data

Restructured Location Encodable Data

As 'Location_NGR' is a duplicate of the data in 'Main_X' and 'Main_Y' and the study is using 'Location_X' and 'Location_Y', it will be deleted. See: [Location_NGR](#).

'Main_Coordinate_System' provides the projection information when using the coordinates, 'Main_X' and 'Main_Y'. As this study is using, 'Location_X' and 'Location_Y', this feature can be deleted. See: [Main Coordinate System](#).

```
In [ ]: location_encodable_features_short = [
    'Location_Current_County',
    'Location_Historic_County',
    'Location_Current_Parish']

location_encodable_data_short = \
location_encodable_data[location_encodable_features_short].copy()
location_encodable_data_short.head()
```

```
Out[ ]:   Location_Current_County  Location_Historic_County  Location_Current_Parish
0           Herefordshire          Herefordshire            Aconbury
1           Herefordshire          Herefordshire            Kimbolton
2           Herefordshire          Herefordshire            Dormington
3           Herefordshire          Herefordshire            Adferton
4           Herefordshire          Herefordshire            Colwall
```

Resolve Null Values in Location Current Parish

Test for 'NA' in 'Location_Current_Parish'.

```
In [ ]: test_cat_list_for_NA(location_encodable_data_short, ['Location_Current_Parish'])

Location_Current_Parish 0

'NA' was not present so the null values are now filled with 'NA'.
```

```
In [ ]: location_encodable_data_short = \
update_cat_list_for_NA(location_encodable_data_short, ['Location_Current_Parish'])
```

The reformatted data package is tested to confirm there are no null values.

```
In [ ]: location_encodable_data_short_review = test_numeric(location_encodable_data_short)
location_encodable_data_short_review
```

```
Out[ ]:      Feature  Entries  Numeric  Non-Numeric  Null
0  Location_Current_County    4147      0        4147      0
1  Location_Historic_County    4147      0        4147      0
2  Location_Current_Parish    4147      0        4147      0
```

Add Density Feature

Density can be calculated based on the distribution of the 'Location_X' and 'Location_Y' coordinates. The density is calculated using the `scipy.stats` tool, `gaussian_kde`.

Ref: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html

Ref: <https://matplotlib.org/stable/tutorials/colors/colormaps.html>

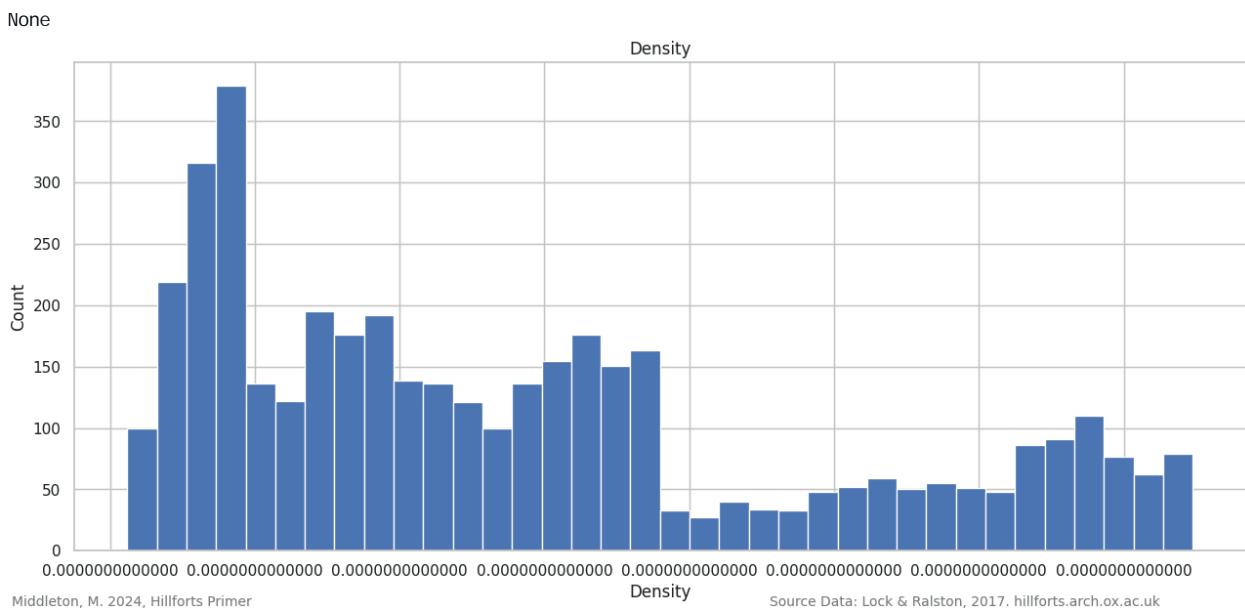
```
In [ ]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short, False)
location_numeric_data_short.head()
```

```
Out[ ]:   Location_X  Location_Y      Density
0     -303295    6798973  1.632859e-12
1     -296646    6843289  1.540172e-12
2     -289837    6808611  1.547729e-12
3     -320850    6862993  1.670548e-12
4     -261765    6810587  1.369981e-12
```

Density Data Plotted

The distribution of density values, seen in the density histogram, peaks toward the left and is low across the remainder of the chart. This is unlikely to give the best results when plotted. See: [Density Data Transformed](#).

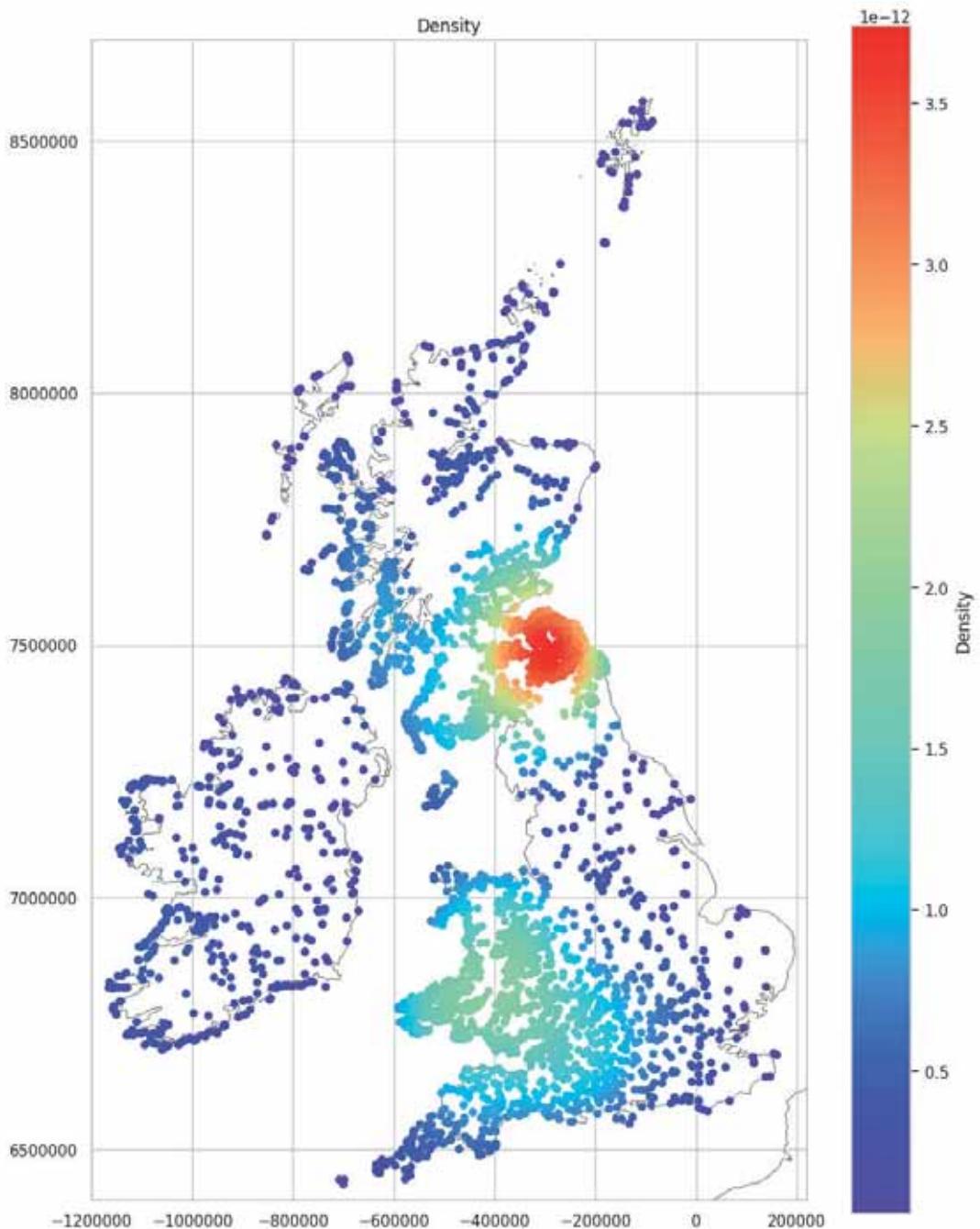
```
In [ ]: plot_histogram(location_numeric_data_short['Density'], 'Density', 'Density')
```



Density Data Mapped

The density plot shows two clusters in the data. The most intense cluster is toward the eastern end of the Southern Uplands, while another can be seen around the southern end of the Cambrian Mountains.

```
In [ ]: fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
show_background(plt, ax)
location_XY_plot()
plt.scatter(location_numeric_data_short['Location_X'],
            location_numeric_data_short['Location_Y'],
            c=location_numeric_data_short['Density'], cmap=cm.rainbow, s=25)
plt.colorbar(label='Density')
title = 'Density'
plt.title(get_print_title(title))
save_fig(title)
plt.show()
```



Middleton, M. 2024, Hillforts Primer

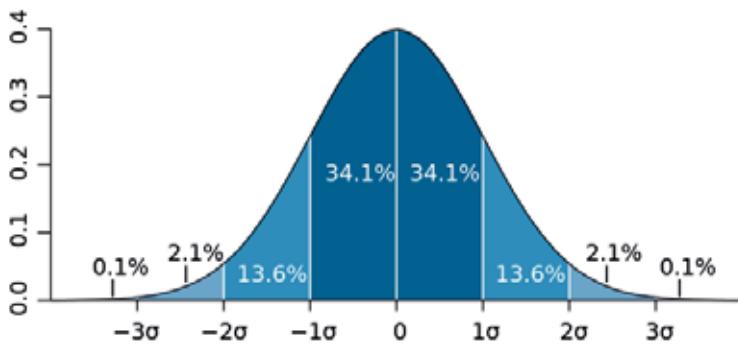
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Location Data: Density Data Transformed Mapped](#).

See: [Location Data: Ireland Density Data Mapped](#).

Density Data Transformed

In [Density Data Plotted](#) it was noted that the distribution of the density values was crushed to the left of the histogram and relatively flat across the remainder of the chart. Ideally, we want the data to be distributed as close as is possible to a normal curve, with most of the data toward the centre of the histogram.



A Normal Curve

M. W. Toews, *CC BY 2.5*, via Wikimedia Commons

The density values are transformed, to create a new feature, using the Boxcox power transformation. The Boxcox transformation searches across a range of transform formula and returns the distribution closest to a normal curve. This does not mean the resulting distribution is a normal curve, it is simply the closest possible from the options tested.

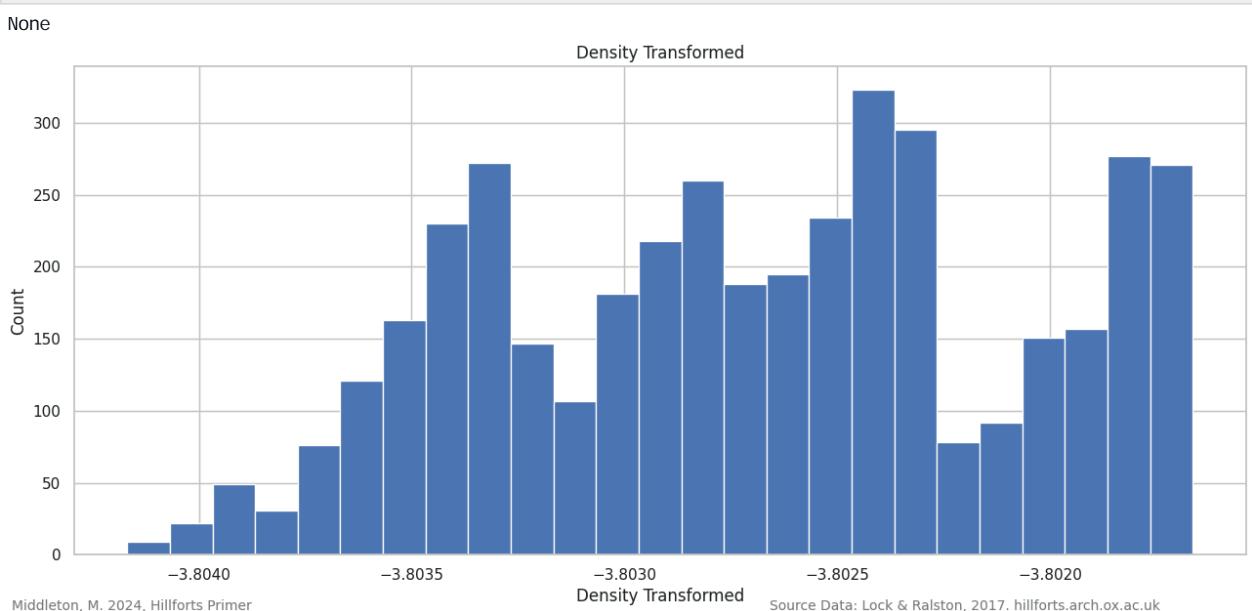
Ref: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.boxcox.html>

```
In [ ]: transformed_location_numeric_data_short = location_numeric_data_short.copy()
transformed_location_numeric_data_short['Density_trans'], best_lambda = \
stats.boxcox(transformed_location_numeric_data_short['Density'])
transformed_location_numeric_data_short.head()
```

	Location_X	Location_Y	Density	Density_trans
0	-303295	6798973	1.632859e-12	-3.802403
1	-296646	6843289	1.540172e-12	-3.802450
2	-289837	6808611	1.547729e-12	-3.802446
3	-320850	6862993	1.670548e-12	-3.802385
4	-261765	6810587	1.369981e-12	-3.802540

The resulting histogram is still loaded toward one end - this time the right - but the remainder of the data is spread across the centre of the histogram and is likely to lead to a more revealing density map.

```
In [ ]: plot_histogram(transformed_location_numeric_data_short['Density_trans'], \
'Density Transformed', 'Density Transformed')
```

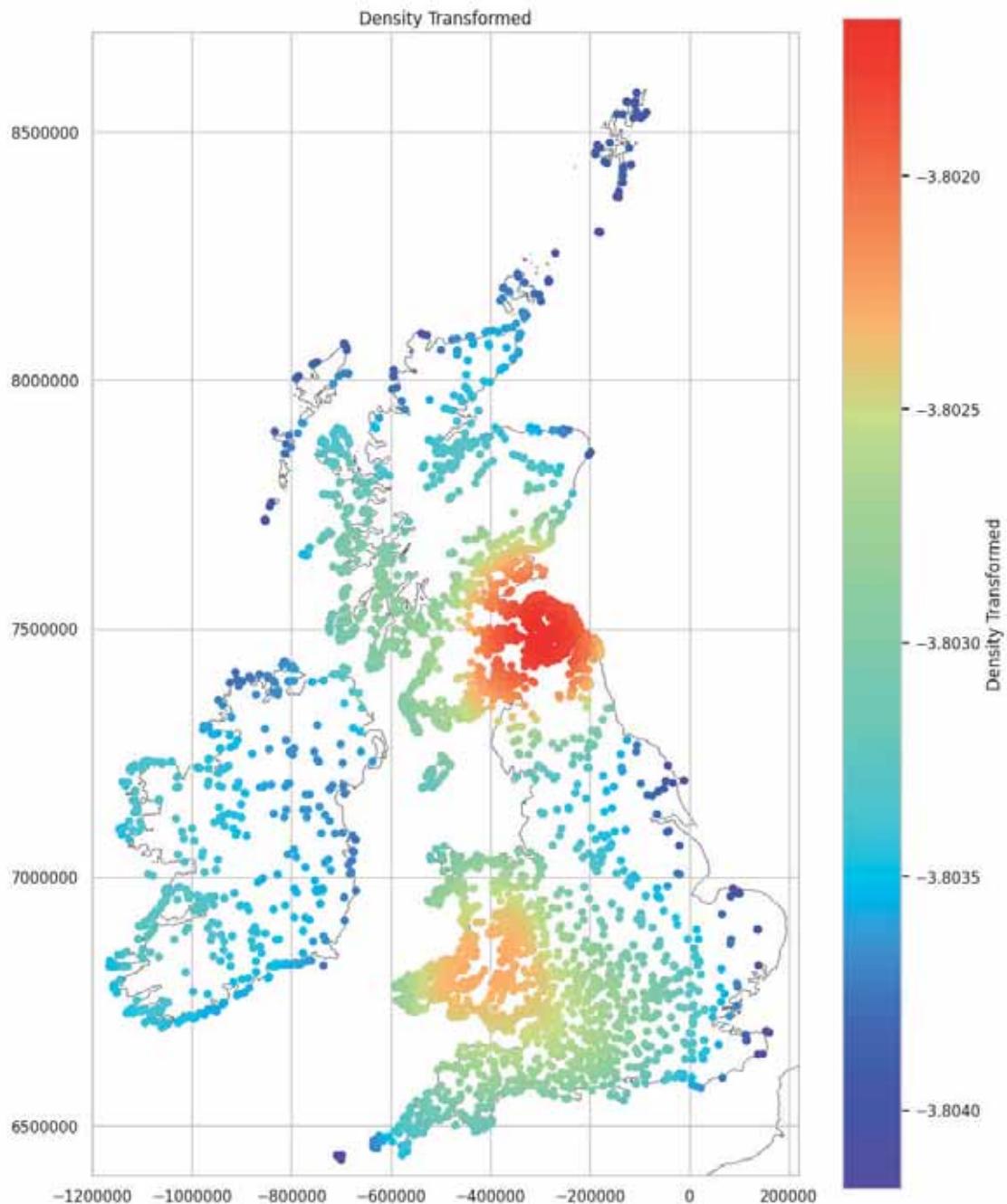


```
In [ ]: best_lambda
```

```
Out[ ]: 0.2627814121099985
```

The transformation results in a more subtle distribution with areas along the west coast of Scotland and the west coast of Ireland now beginning to highlight variations in distribution.

```
In [ ]: density_transformed(transformed_location_neruic_data_short)
```



See: [Location Data: Density Data Mapped](#).

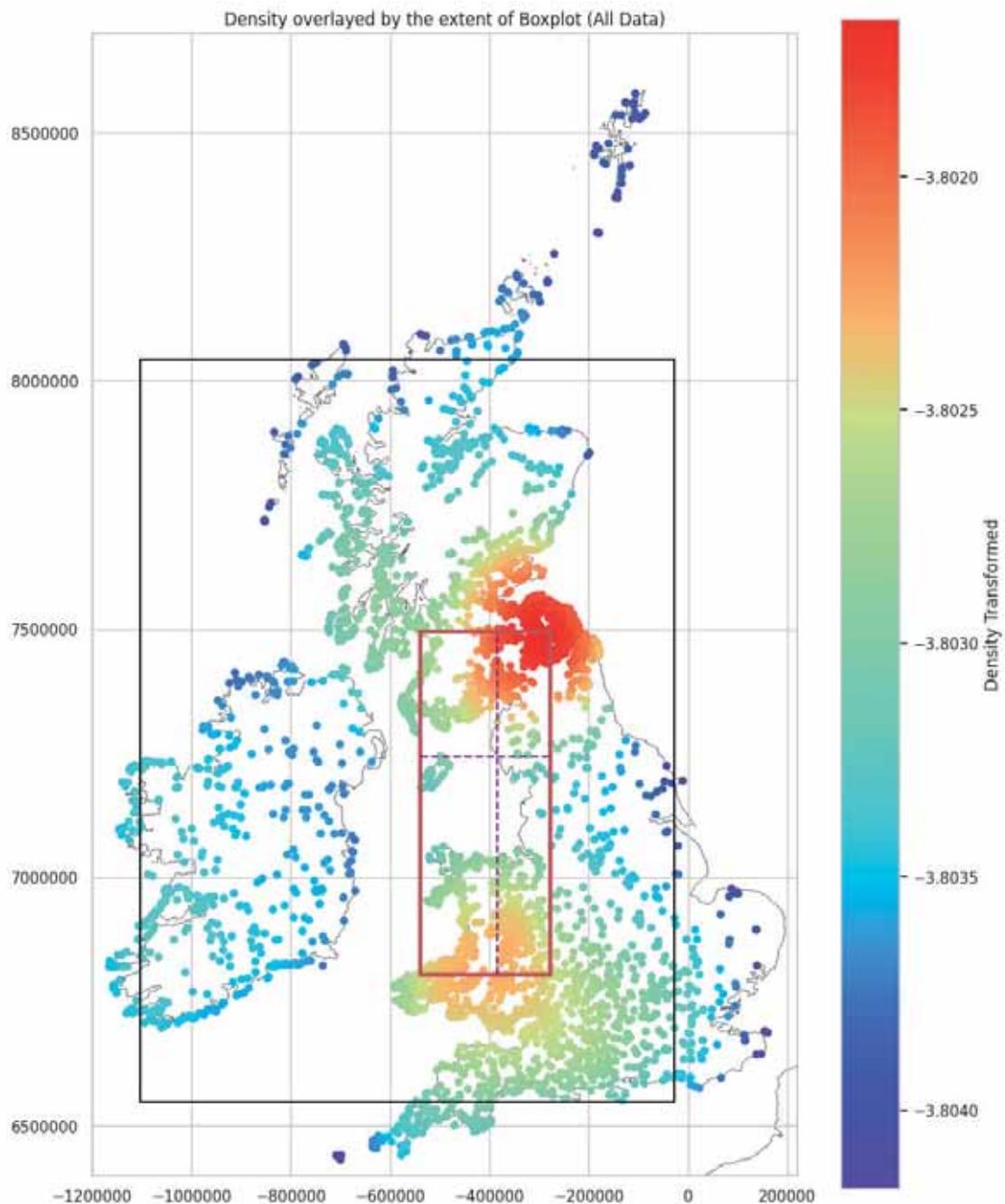
See: [Location Data: Ireland Density Data Mapped](#).

Density Map showing Extent of Boxplot

Mapped boxplots show the central 50% of the data as a red box - the IQR (interquartile range). The black box shows the extent of the whiskers, which delineate the extent of 95.6% of the data (two standard deviations). Data outside the black box are considered outliers. Within the central IQR the mean values are shown as a cross hair.

The density plot and the histogram in [Location_Y Data Plotted - Northing](#) show there are two large peaks in the 'Location_Y' data. The northern peak is more pronounced than the southern and it is therefore making it difficult to see variation in the density plot to the south. The boxplot confirms there are, at least, two peaks in the data as it is stretched between the two density clusters.

```
In [ ]: plot_density_boxplot(transformed_location_numeric_data_short)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

North / South Density Split

To improve the density plots for the southern data and to attempt to isolate discrete data packages for each cluster, the data is split at the lowest point between the two peaks ('Location_Y' = 7,070,000).

```
In [ ]: split_location = 7070000
south = \
transformed_location_numeric_data_short\[transformed_location_numeric_data_short['Location_Y'] < \
split_location].copy().reset_index(drop=True)
north = \
transformed_location_numeric_data_short\[transformed_location_numeric_data_short['Location_Y'] >= \
split_location].copy().reset_index(drop=True)
```

Northern Density

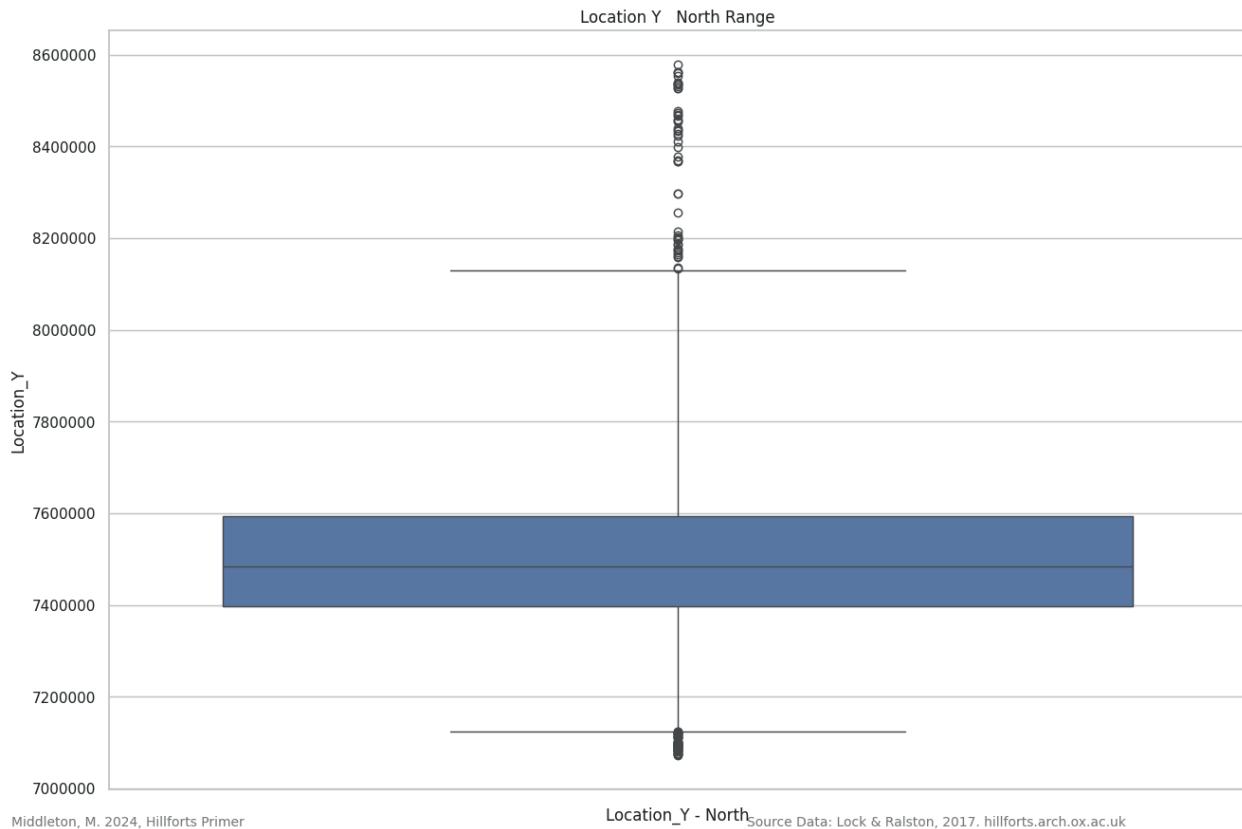
Northern Location Data Plotted

From northing 7,070,000 to 9,000,000, there are 2314 records. The data has a long tail of outliers to the north, due to the large area of land and high density of records in the Southern Uplands compared to the small area of land and tiny number of records in the Northern Isles.

```
In [ ]: north.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2314 entries, 0 to 2313
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Location_X    2314 non-null   int64  
 1   Location_Y    2314 non-null   int64  
 2   Density        2314 non-null   float64 
 3   Density_trans  2314 non-null   float64 
dtypes: float64(2), int64(2)
memory usage: 72.4 KB
```

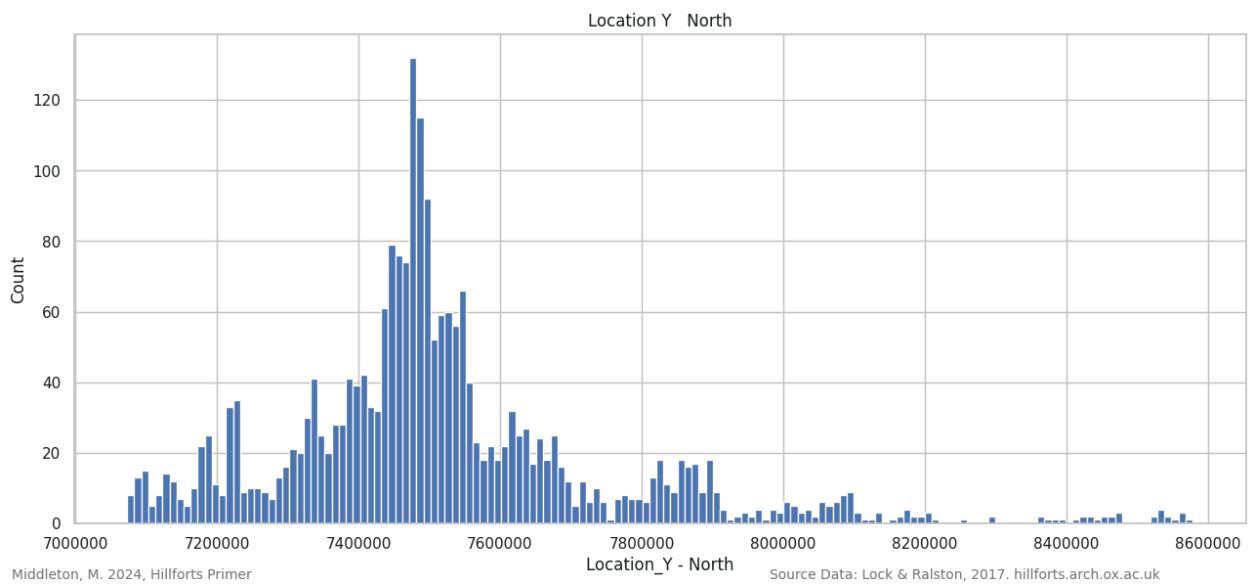
```
In [ ]: location_Y_north_data = plot_data_range(north['Location_Y'], 'Location_Y - North')
```



The histogram now shows a single main peak at around 7,500,000.

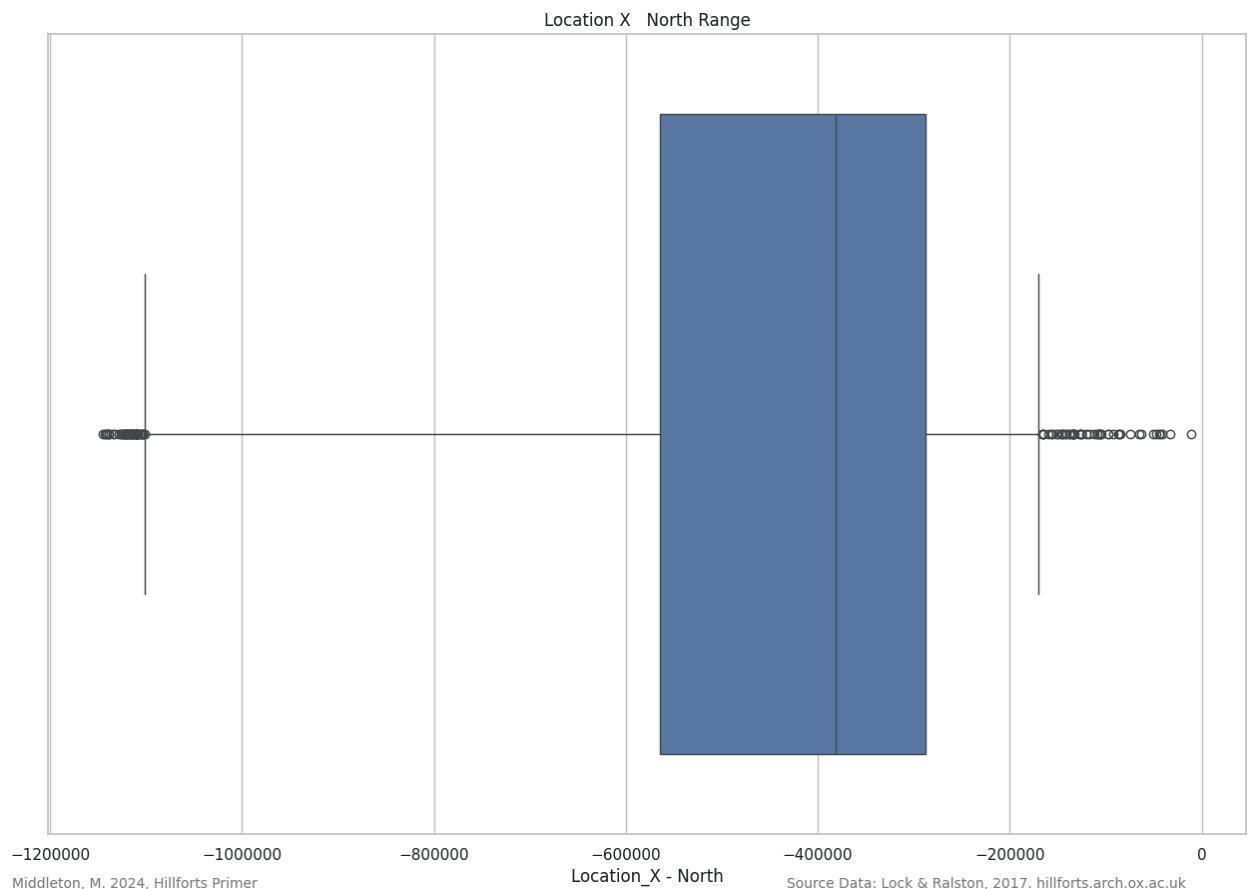
```
In [ ]: plot_histogram(north['Location_Y'], 'Location_Y - North', \
                      'Location_Y - North', 10000)
```

10000



Along the x-axis, the 'Location_X' boxplot shows the eastings to have an elongated whisker, toward the east, due to the data being spread, thinly, across Ireland.

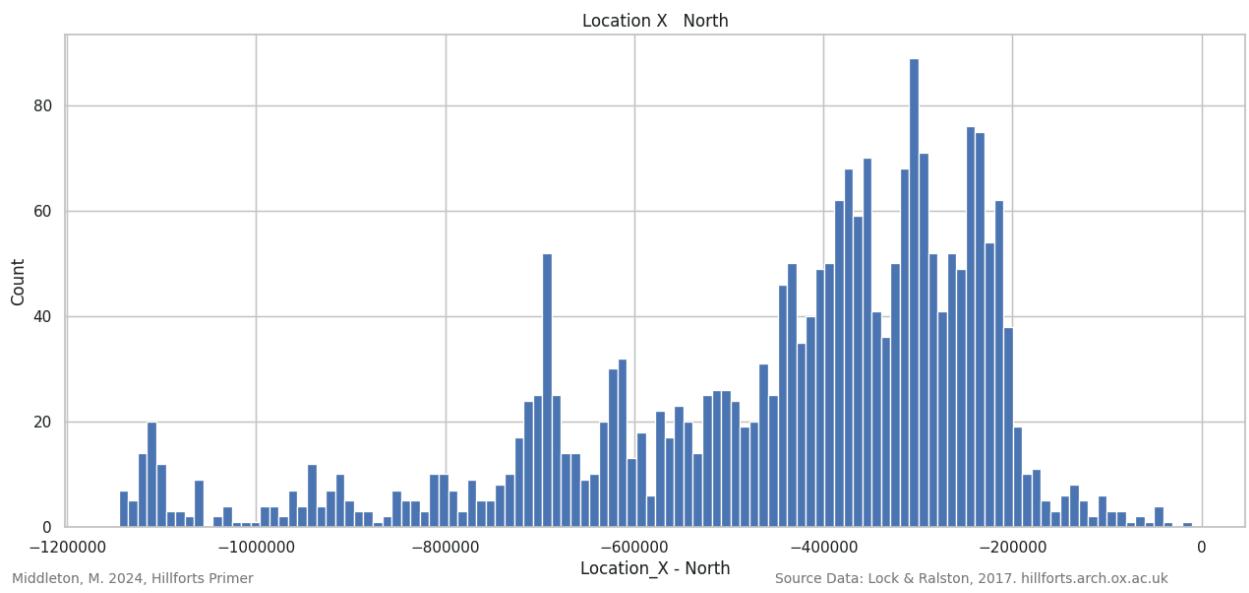
```
In [ ]: location_X_north_data = plot_data_range(north['Location_X'], \
                                              'Location_X - North', "h")
```



The 'Location_X' data contains a number of small peaks overlying the larger, broad, dominant peak. There is a trough in the data at around -590,000 while the midpoint between the two tallest peaks is around -500,000. See: [Northern Data Split East-West](#).

```
In [ ]: plot_histogram(north['Location_X'], 'Location_X - North', \
                      'Location_X - North', 10000)
```

10000



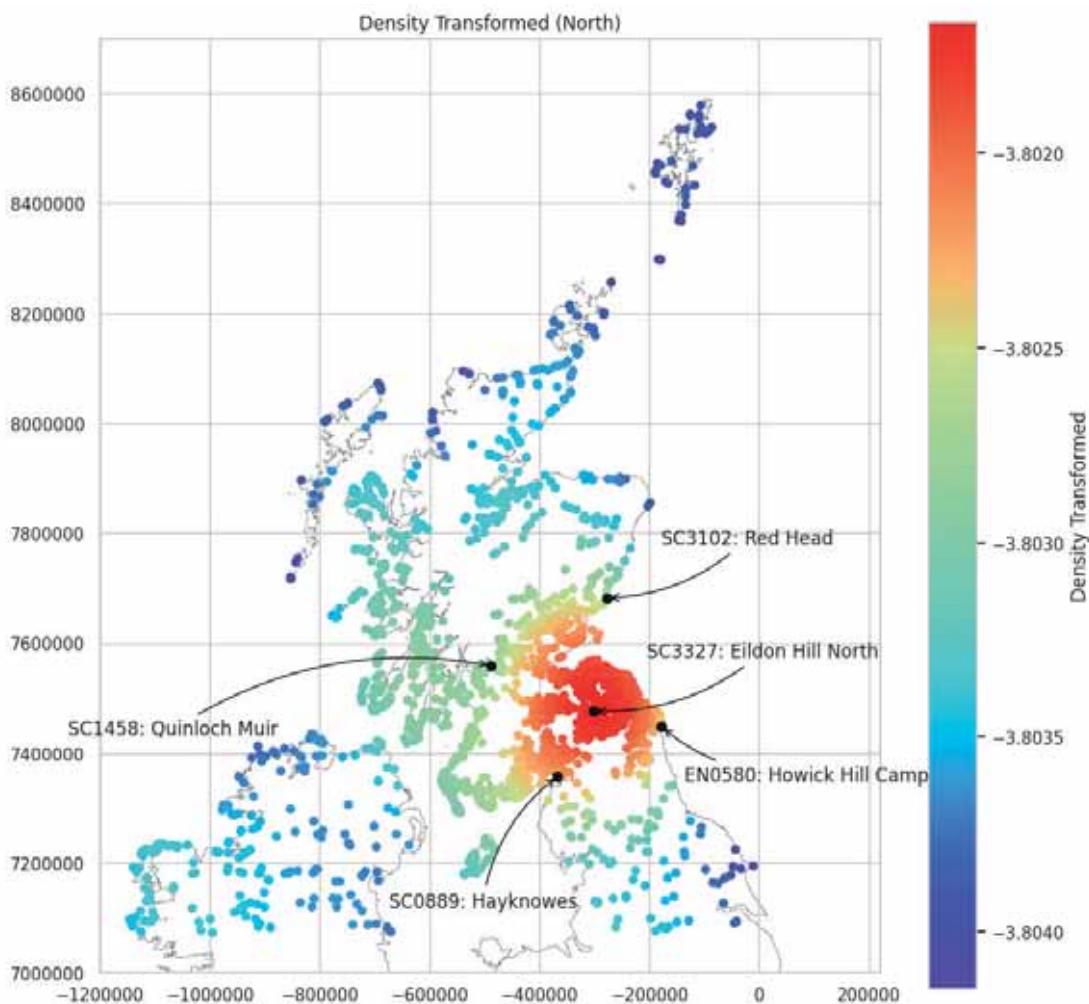
Northern Data Density Mapped (Transformed)

Data north of northing 7,070,000.

The density plot shows the main cluster centred near Scotland's largest hillfort, Eildon Hill. The southern edge of the most dense concentration of hillforts coincides with a line from the Solway Firth, around SC0889: Hayknowes to EN0580: Howick Hill Camp. The focus is very much to the east and does not extend into Dumfries and Galloway. There are secondary concentrations of hillforts along the line of the highland boundary fault, from SC1458: Quinloch Muir to the coast at SC3102: Red Head and another along the Atlantic coast from the Solway Firth to Skye.

```
In [ ]: show_eildon = True
```

```
In [ ]: plot_northern_density(show_eildon, north)
```



Middleton, M. 2024, Hillforts Primer

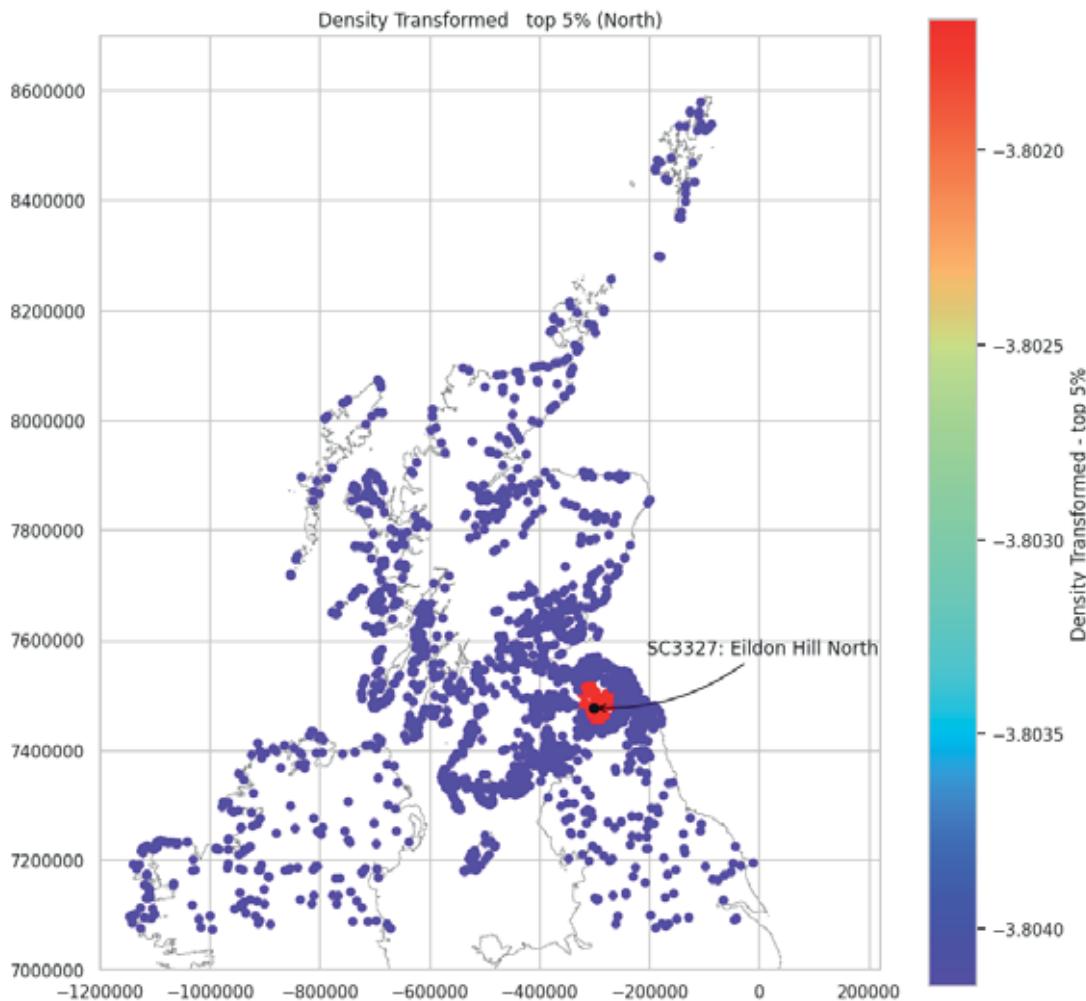
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Northern Data Density Mapped (top 5%)

The most dense concentration of forts, the top 5%, can be found between Hawick, Jedburgh and Selkirk.

```
In [ ]: north_top5 = north.copy()
north_top5['Density_trans'].where(north_top5['Density_trans'] > \
                                north_top5['Density_trans'].quantile(0.95), \
                                north_top5['Density_trans'].min(), inplace=True)
```

```
In [ ]: plot_northern_density_transformed(show_eildon, north_top5)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Northern Data Density Mapped (Capped)

Capping the density enables peripheral clusters of data to reveal themselves in a little more detail. Of the secondary clusters, the cluster along the Atlantic coast is most dense, along either side of the Clyde and extending up toward SC2466: Dunadd. A smaller cluster of hillforts can be seen, at the northern end of the Great Glen, and east, along the Northeastern edge of the Grampian mountains. The main cluster is now seen to be more 'greedy', extending from SC0244: Drummore Castle to SC1458: Quinloch Muir and then, along the full length of the highland boundary fault, to SC3102: Red Head.

```
In [ ]: north['Density_trans'].describe()
```

```
Out[ ]: count    2314.000000
mean      -3.802575
std       0.000704
min      -3.804147
25%      -3.803250
50%      -3.802482
75%      -3.801885
max      -3.801664
Name: Density_trans, dtype: float64
```

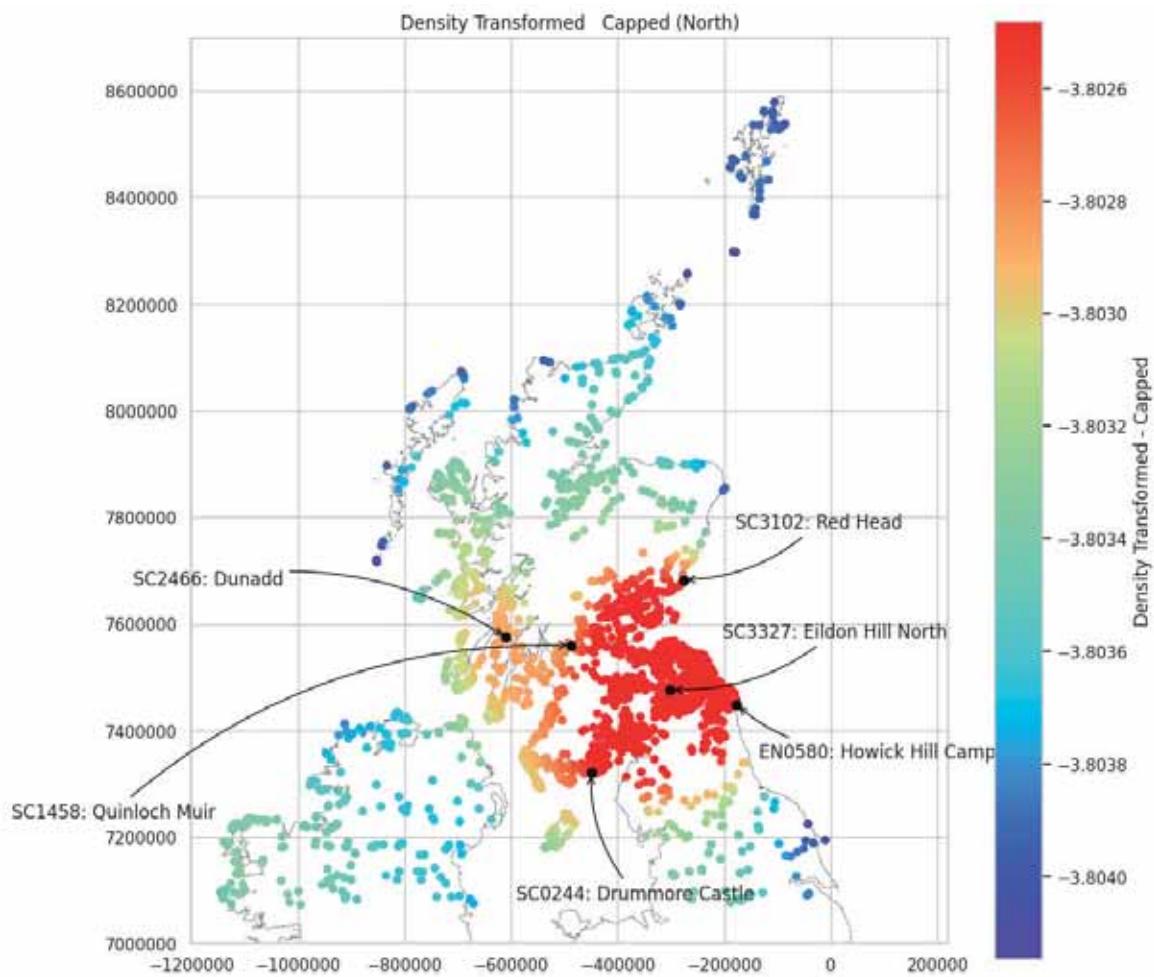
```
In [ ]: north['Density_trans'].quantile(0.5)
```

```
Out[ ]: -3.802482376914073
```

The data is capped using the transformed density value for the central quantile (the middle 50% of the data).

```
In [ ]: north_clip = north.copy()
north_clip['Density_trans'].where(north_clip['Density_trans'] < \
                                 north['Density_trans'].quantile(0.5), \
                                 north['Density_trans'].quantile(0.5), \
                                 inplace=True)
```

```
In [ ]: plot_density_transformed_north_capped(show_eildon, north_clip)
```



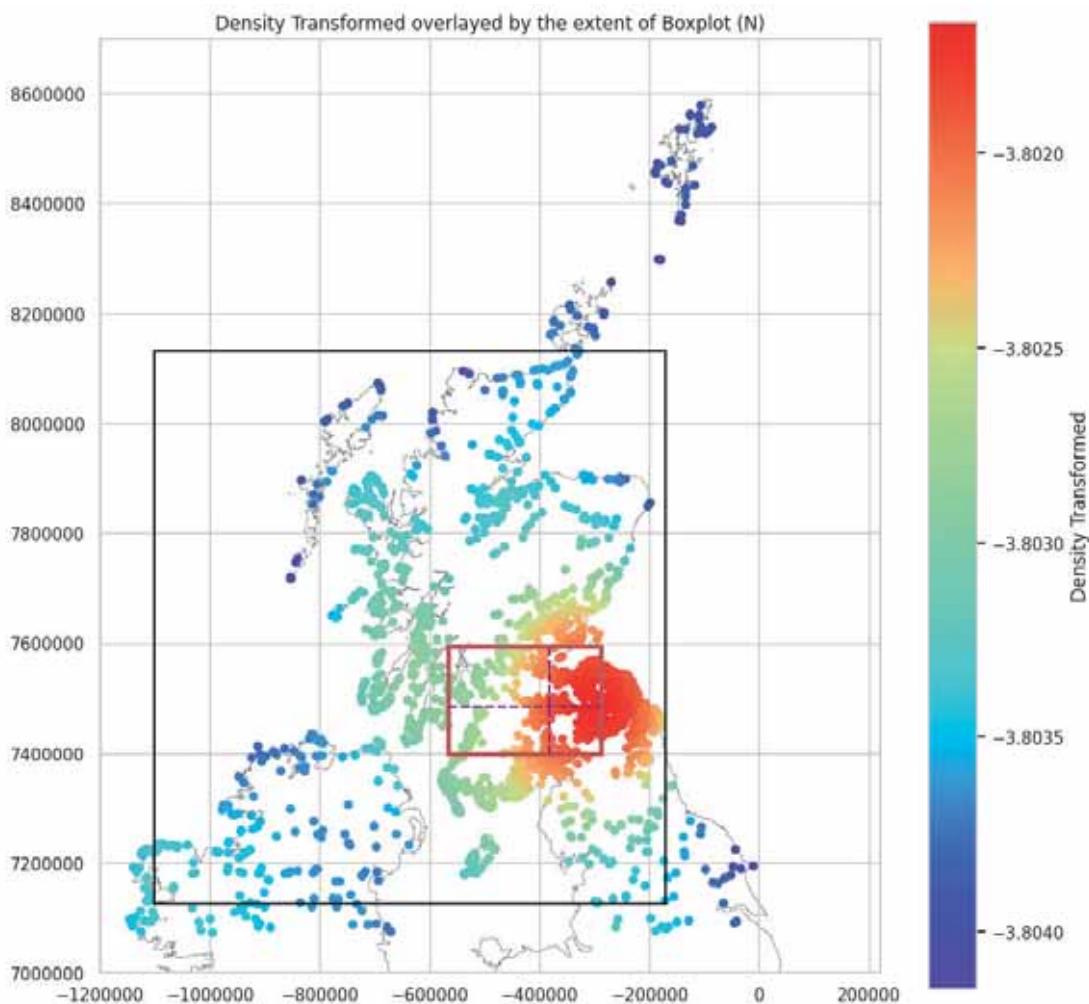
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Density Map showing Extent of Boxplots (North)

The northern boxplot does not align with the density cluster along the east-west axis. It is stretched to the west indicating there may be more than one cluster in the northern data package. This will be explored more in [Northern Data Split East-West](#).

```
In [ ]: plot_density_transformed_north_boxplot(north, location_X_north_data, location_Y_north_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Northern Data Split East-West

The data is split along the approximate midpoint between the two highest peaks in the data ('Location_X' = -500,000). See: '[Location_X - North](#)' in Northern Location Data Plotted.

```
In [ ]: split_location = -500000
north_west = north[north['Location_X'] < \
                  split_location].copy().reset_index(drop=True)
north_east = north[north['Location_X'] >= \
                  split_location].copy().reset_index(drop=True)
```

There are 716 records in the Northwestern data.

```
In [ ]: north_west.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 716 entries, 0 to 715
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   Location_X    716 non-null   int64  
 1   Location_Y    716 non-null   int64  
 2   Density       716 non-null   float64 
 3   Density_trans 716 non-null   float64 
dtypes: float64(2), int64(2)
memory usage: 22.5 KB
```

There are 1598 records in the Northeastern data.

```
In [ ]: north_east.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1598 entries, 0 to 1597
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Location_X    1598 non-null   int64  
 1   Location_Y    1598 non-null   int64  
 2   Density        1598 non-null   float64 
 3   Density_trans  1598 non-null   float64 
dtypes: float64(2), int64(2)
memory usage: 50.1 KB

```

Cluster Data Packages

This section contains code snippets and functions used in creating regional data packages. To facilitate reading this document this section, containing mostly code blocks, is minimised.

To enable further analysis and to aid the plotting of figures, the data is split into data packages based on the review of location clusters in this document. See: [Cluster Data Packages Mapped](#).

The cluster packages are identified in a new feature, 'Cluster'. This is given a default value of 'NA'.

```
In [ ]: cluster_data = hillforts_data[['Location_X', 'Location_Y', \
                                    'Main_Country_Code']].copy()
cluster_data['Cluster'] = 'NA'
```

Irish records are given a temporary identification of 'I'.

```
In [ ]: cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
                                    'NI', 'I', inplace=True)
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
                                    'IR', 'I', inplace=True)
```

The Irish records with a 'Location_Y' above 7,060,000 are identified as cluster 'North Ireland'. The remainder are identified as 'South Ireland'.

```
In [ ]: cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000) , \
    'North Ireland', cluster_data['Cluster']
)
cluster_north_irland = cluster_data[cluster_data['Cluster'] == \
                                    'North Ireland'].copy()
```

```
In [ ]: cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , \
    'South Ireland', cluster_data['Cluster']
)
cluster_south_irland = cluster_data[cluster_data['Cluster'] == \
                                    'South Ireland'].copy()
```

The remaining records are split into 'South', 'Northeast' and 'Northwest' based on the split locations detailed in the respective sections of this document

```
In [ ]: cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] < 7070000) , \
    'South', cluster_data['Cluster']
)
cluster_south = cluster_data[cluster_data['Cluster'] == 'South'].copy()
```

```
In [ ]: cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] >= -500000), 'Northeast', cluster_data['Cluster']
)
cluster_north_east = cluster_data[cluster_data['Cluster'] == 'Northeast'].copy()
```

```
In [ ]: cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] < -500000), 'Northwest', cluster_data['Cluster']
)
cluster_north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()
```

```
In [ ]: cluster_data = cluster_data.drop(['Main_Country_Code'], axis=1)
```

```
In [ ]: cluster_location_packages = [cluster_north_irland, cluster_south_irland, \
                                    cluster_south, cluster_north_east, \
                                    cluster_north_west]
```

Northwest Density

The density values are refreshed in the clipped Northwestern data.

```
In [ ]: north_west = renew_density(north_west)
```

There are 716 forts in the Northwest data package.

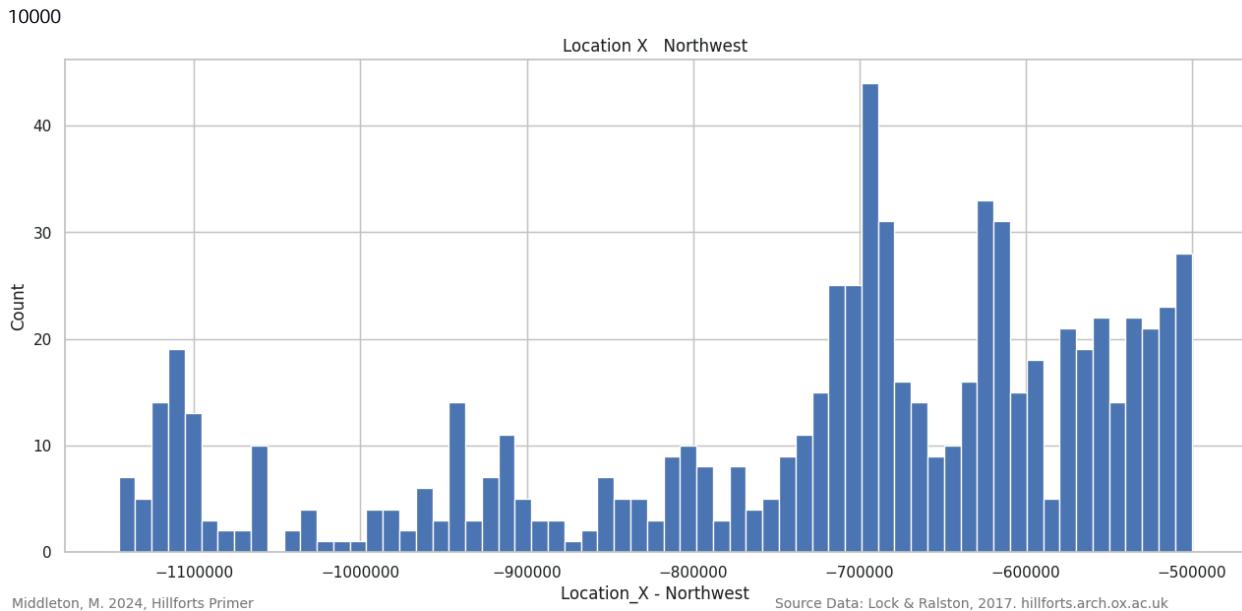
```
In [ ]: north_west.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 716 entries, 0 to 715
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Location_X    716 non-null   int64  
 1   Location_Y    716 non-null   int64  
 2   Density       716 non-null   float64
 3   Density_trans 716 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 22.5 KB
```

Northwest Data Plotted

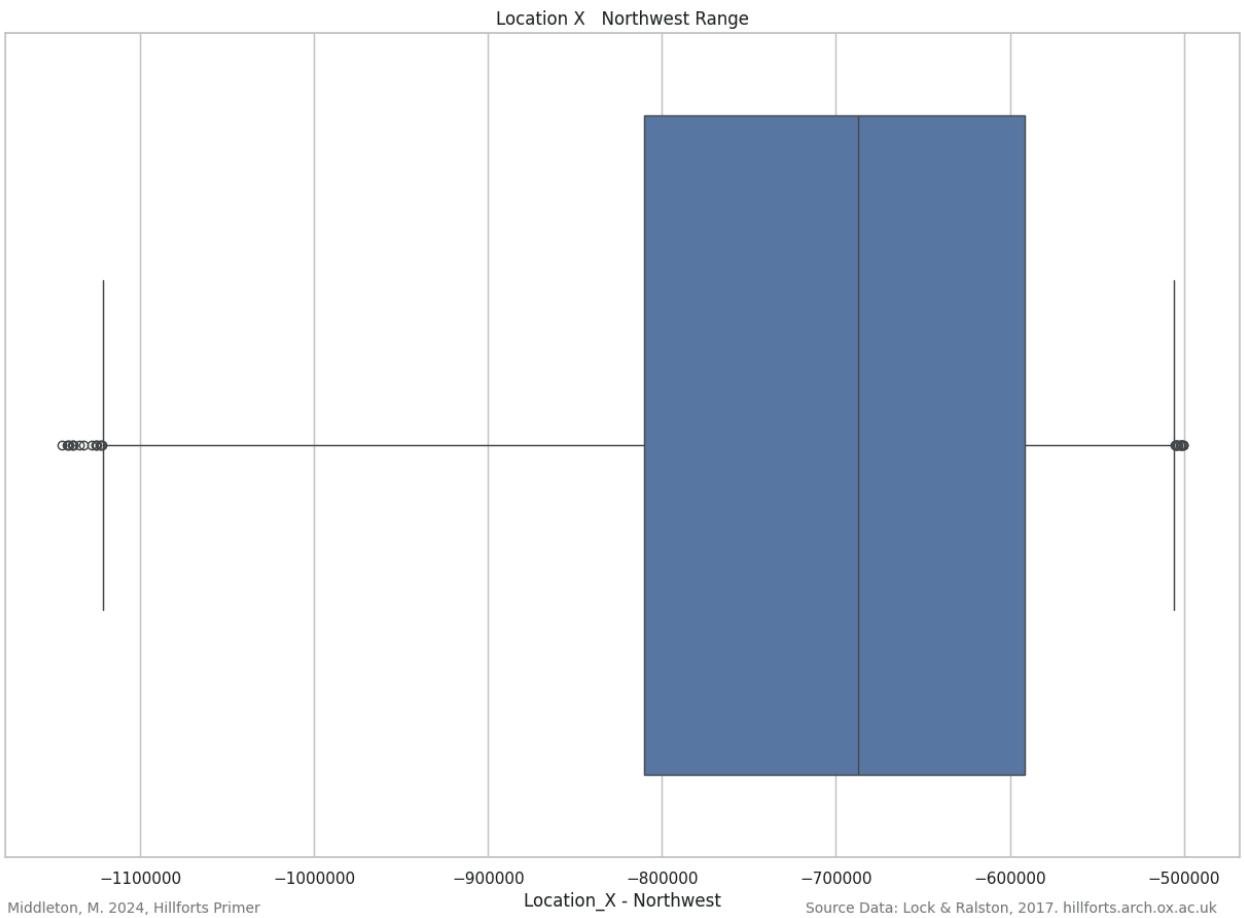
There is a higher concentration of records over Scotland, to the east. The concentration of records over Ireland, from around -750,000 to the far west, is low with the largest peak being beyond -1,100,000. The main peak in the Northwest data is at -700,000 with a secondary peak at -620,000 with a gradually increasing concentration of sites to the east. There is a notable trough in the data around -580,000.

```
In [ ]: plot_histogram(north_west['Location_X'], 'Location_X - Northwest', \
                      'Location_X - Northwest', 10000)
```



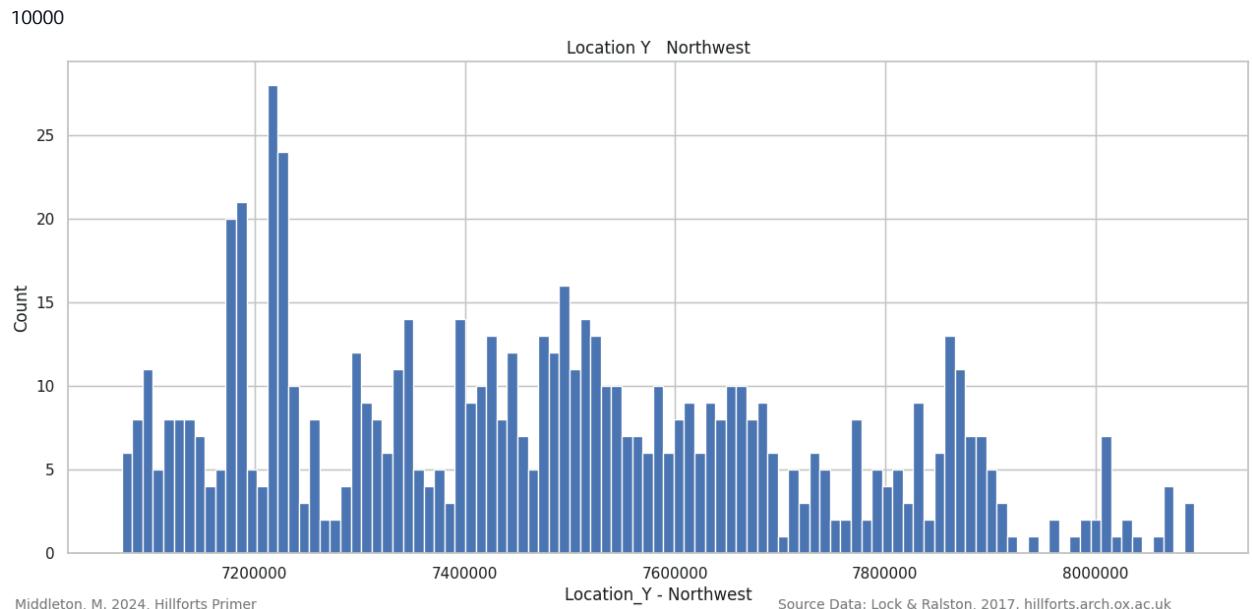
As with the boxplot in [Northern Location Data Plotted](#), the Northwest 'Location_X' boxplot has the elongated whisker caused by the sparse spread of Irish sites to the east. The interquartile range and the median align with the main peak in the data at -700,000.

```
In [ ]: location_X_north_w_data = plot_data_range(north_west['Location_X'], \
                                                    'Location_X - Northwest', "h")
```



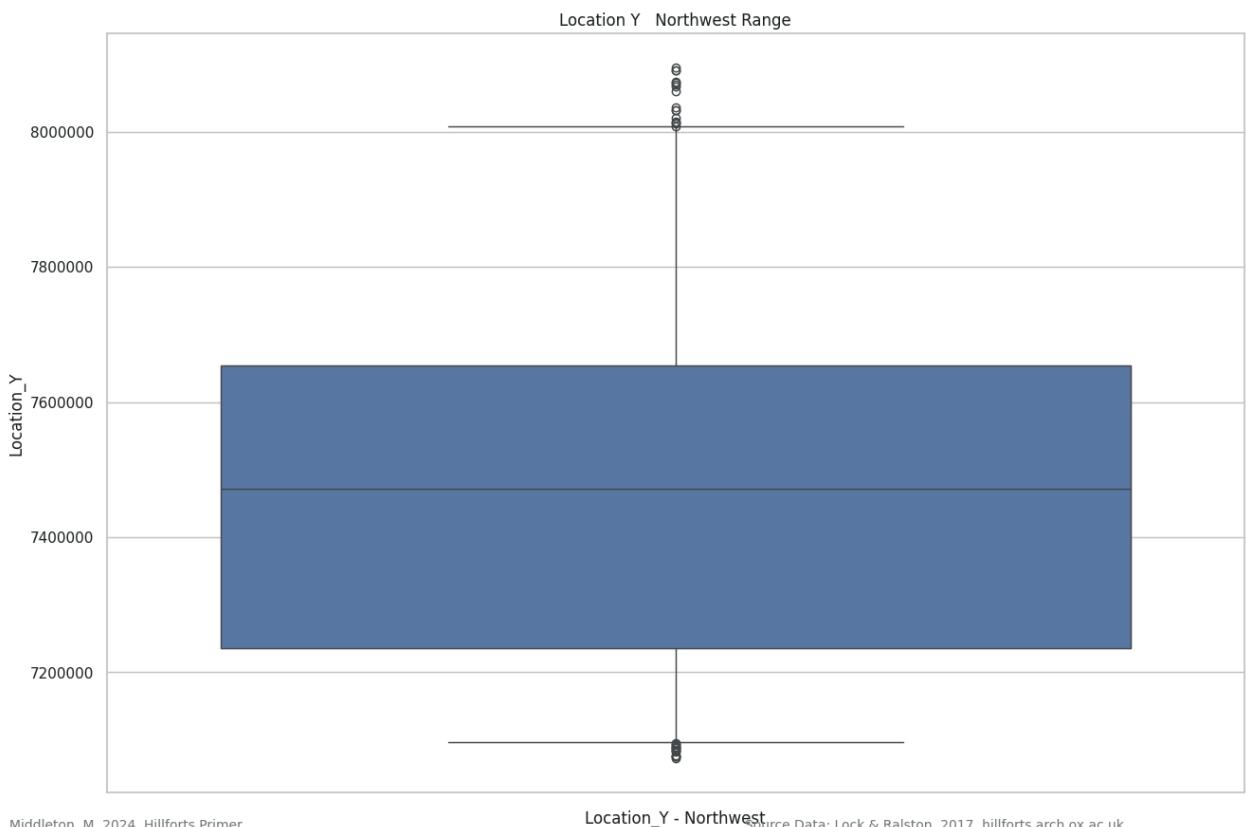
Along the 'Location_Y' axis, there is a roughly even spread of data between 7,300,000 to 7,900,000. Beyond this, to the north, the distribution of hillforts becomes very sparse. There are two pronounced, but narrow, peaks on either side of 7,200,000 which corresponds to the forts in the far west of Ireland.

```
In [ ]: plot_histogram(north_west['Location_Y'], 'Location_Y - Northwest', \
                      'Location_Y - Northwest', 10000)
```



The majority of the data is spread in a wide band on either side of 7,500,000 with most outliers concentrated toward the north.

```
In [ ]: location_Y_north_w_data = plot_data_range(north_west['Location_Y'], \
                                                 'Location_Y - Northwest')
```

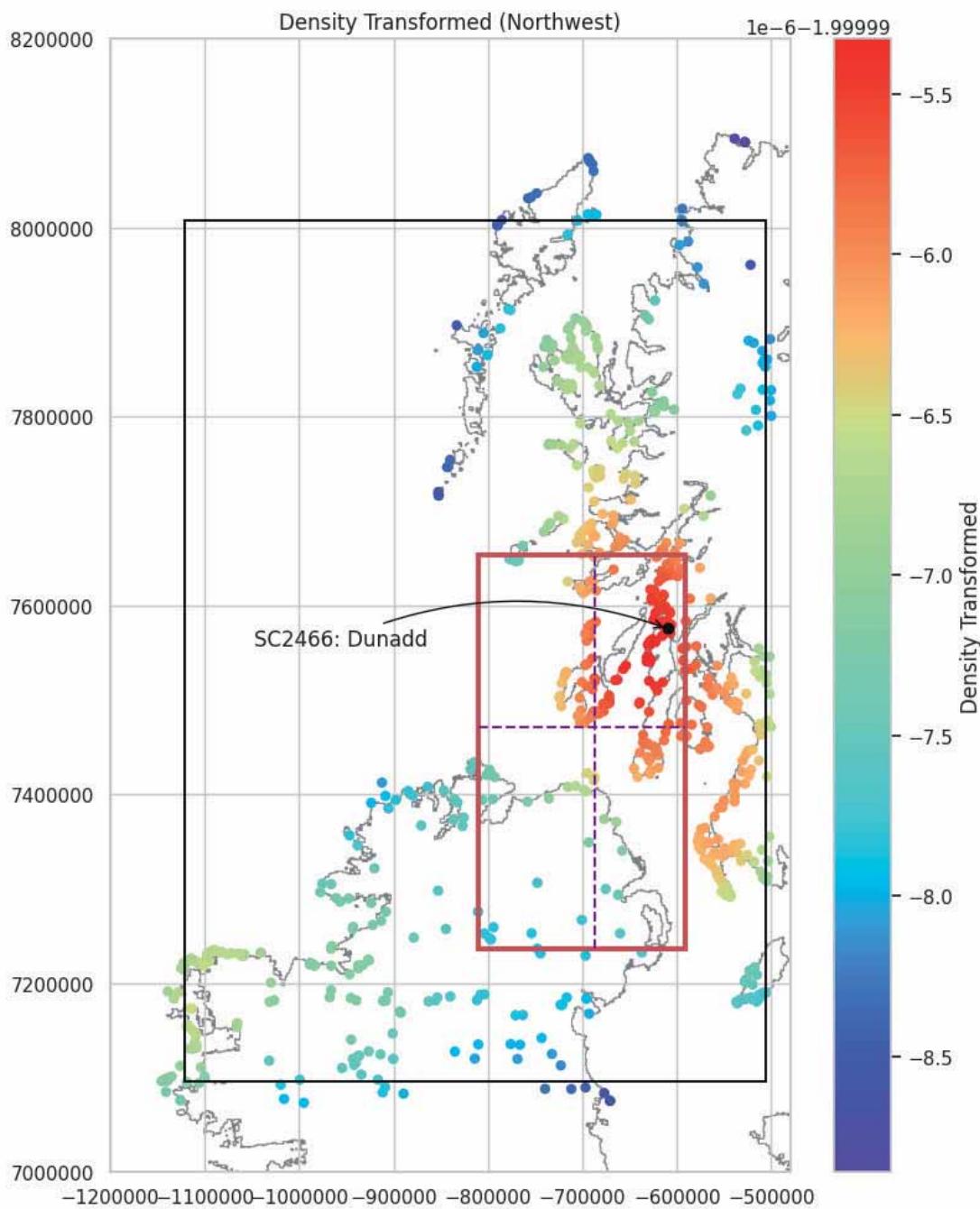


Northwest Data Mapped

The Northwestern data highlights the concentration of hillforts along the southern fringe of the west coast of Scotland. This is particularly pronounced around the Irish Sea with a focus near SC2466: Dunadd. The cluster is strung along the western seaboard and islands of Scotland from the Rhins of Galloway to Skye.

The elongated IQR along the 'Location_X' and 'Location_Y' axes suggests the Irish data is pulling the median value into the Irish Sea and this may indicate that the Northwest cluster should be assessed independently of the Irish data. See: [Northwestern Data Plotted Minus Ireland](#).

```
In [ ]: plot_north_west_densitiy(show_eildon, north_west, location_Y_north_w_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Density Map showing Extent of Northwest Boxplot](#)

See: [Northwestern Data Minus Ireland Mapped](#)

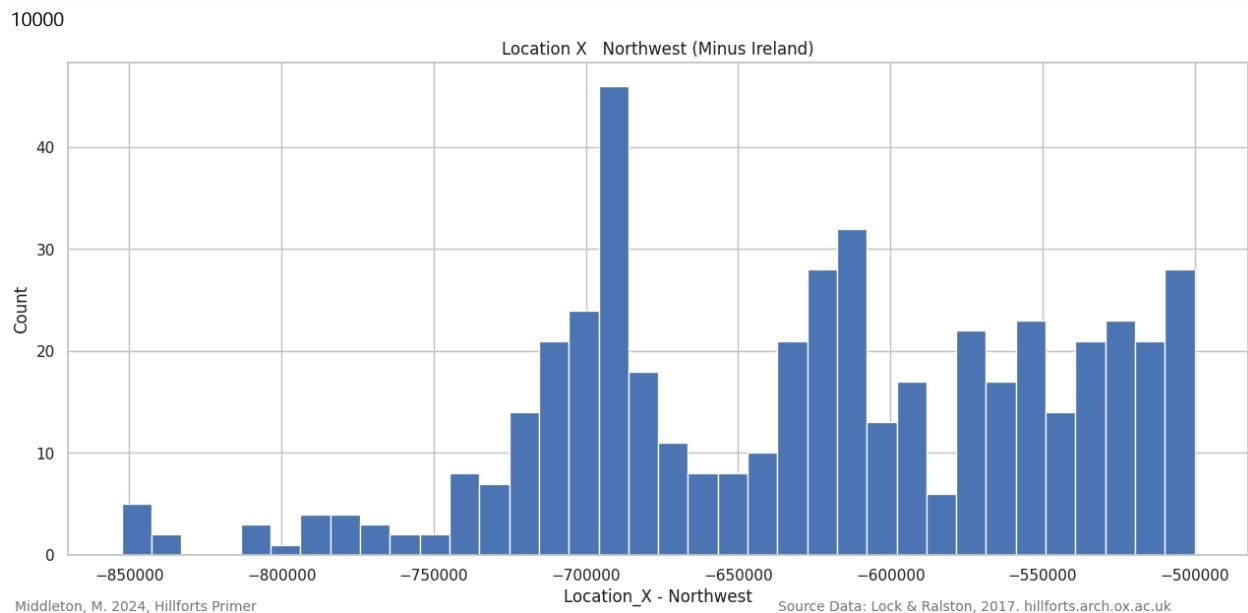
Northwest Data Minus Ireland Plotted

In []: `cluster_north_west.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 487 entries, 120 to 4145
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Location_X        487 non-null    int64  
 1   Location_Y        487 non-null    int64  
 2   Main_Country_Code 487 non-null    object  
 3   Cluster           487 non-null    object  
dtypes: int64(2), object(2)
memory usage: 19.0+ KB
```

The histogram now contains only the Northwestern data in Scotland. See: [Northwest Data Plotted](#).

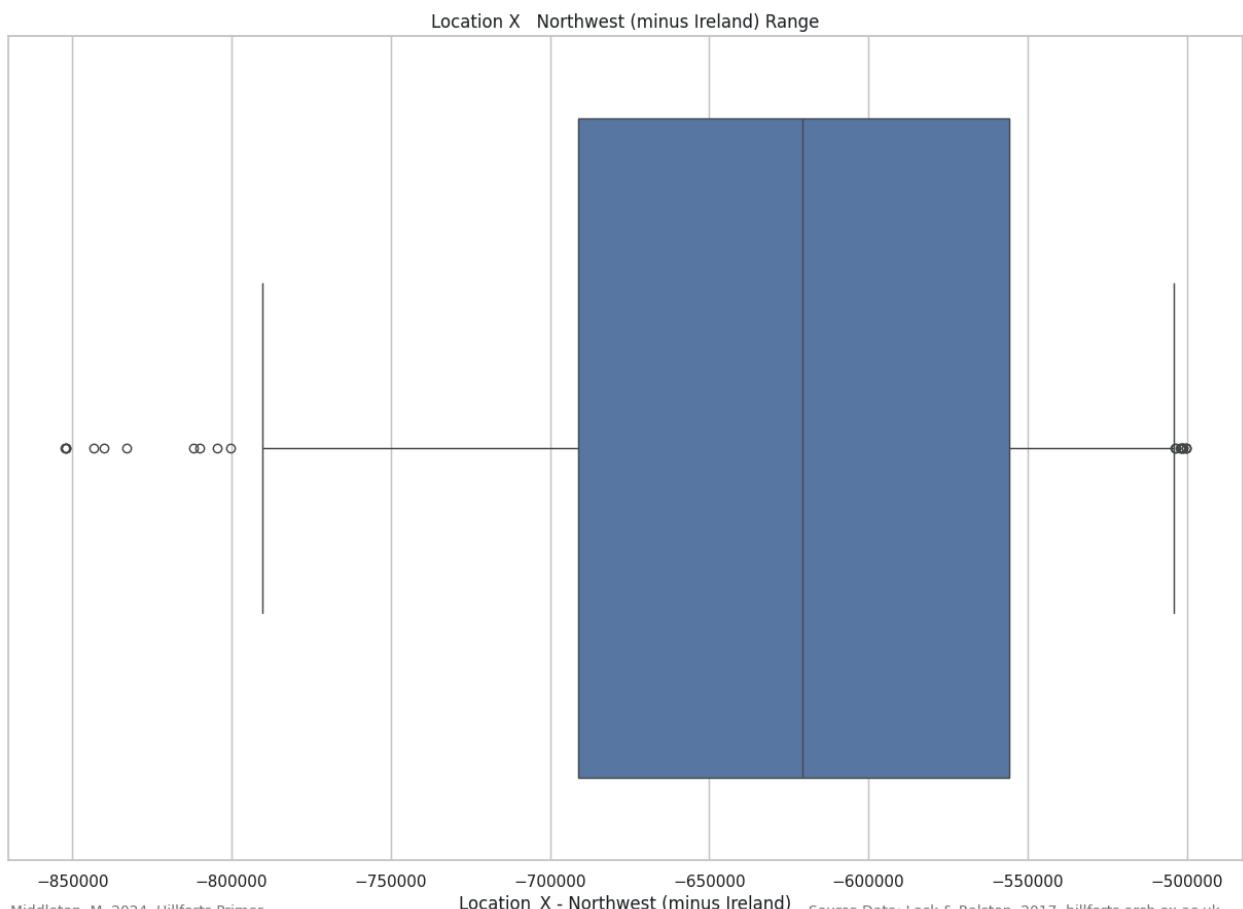
```
In [ ]: plot_histogram(cluster_north_west['Location_X'], 'Location_X - Northwest', \
'Location_X - Northwest (Minus Ireland)', 10000)
```



Having removed the Irish data, the interquartile range has moved east and the western whisker is shorter. The outliers to the west are all forts in the Western Isles.

```
In [ ]: cluster_north_west = cluster_north_west.copy().reset_index()
```

```
In [ ]: location_X_cluster_north_west = \
plot_data_range(cluster_north_west['Location_X'], \
'Location_X - Northwest (minus Ireland)', "h")
```



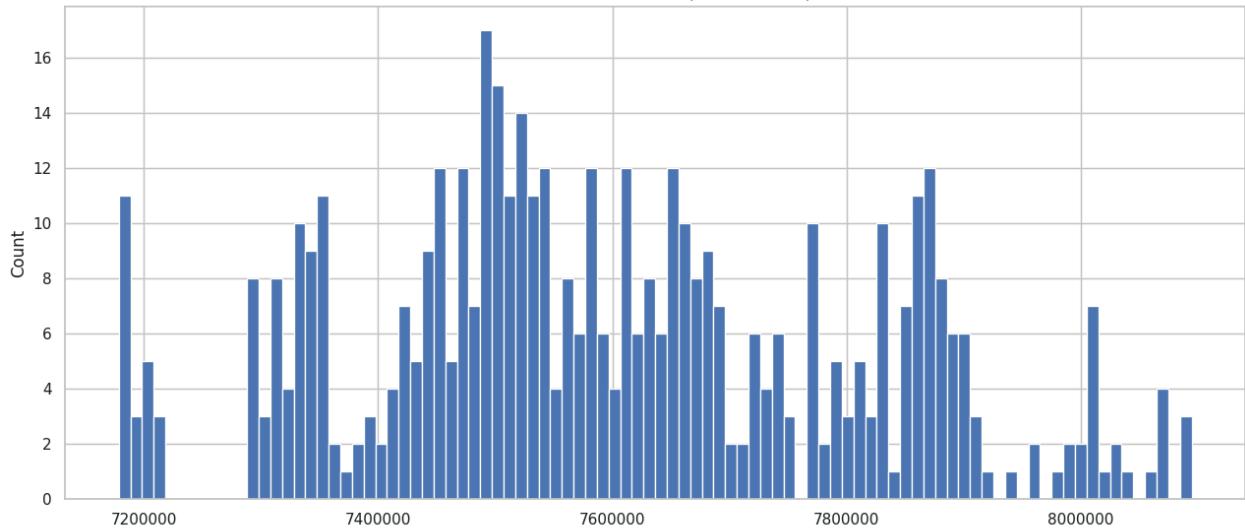
There are similar peaks in the data around 7,200,000, 7,300,000 and 7,900,000. The highest and broadest peak is around 7,500,000.

See: [Northwest Data Plotted](#).

```
In [ ]: plot_histogram(cluster_north_west['Location_Y'], \
'Location_Y - Northwest', \
'Location_Y - Northwest (minus Ireland)', 10000)
```

10000

Location Y Northwest (minus Ireland)

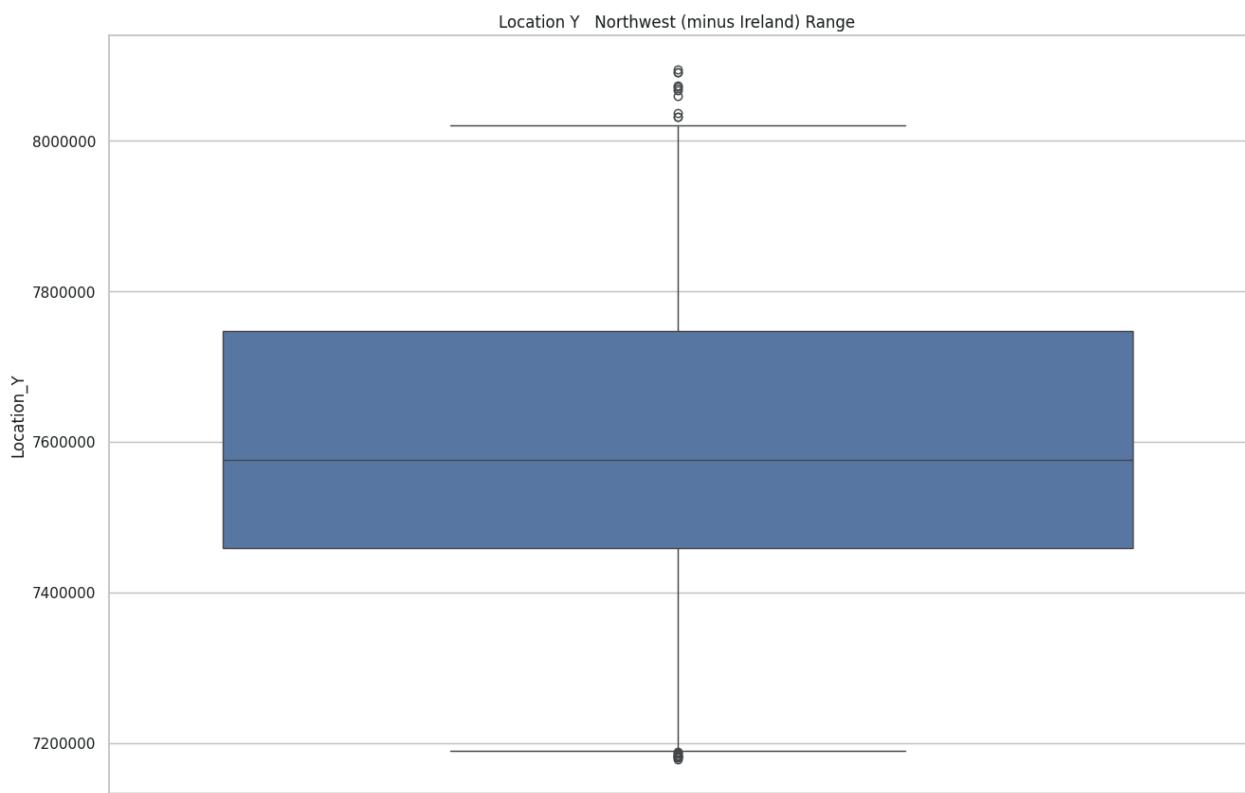


Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Without the Irish data, the interquartile range in the 'Location_Y' axis has contracted north.

```
In [ ]: location_Y_cluster_north_west = \
plot_data_range(cluster_north_west['Location_Y'], \
                 'Location_Y - Northwest (minus Ireland)')
```



Middleton, M. 2024, Hillforts Primer

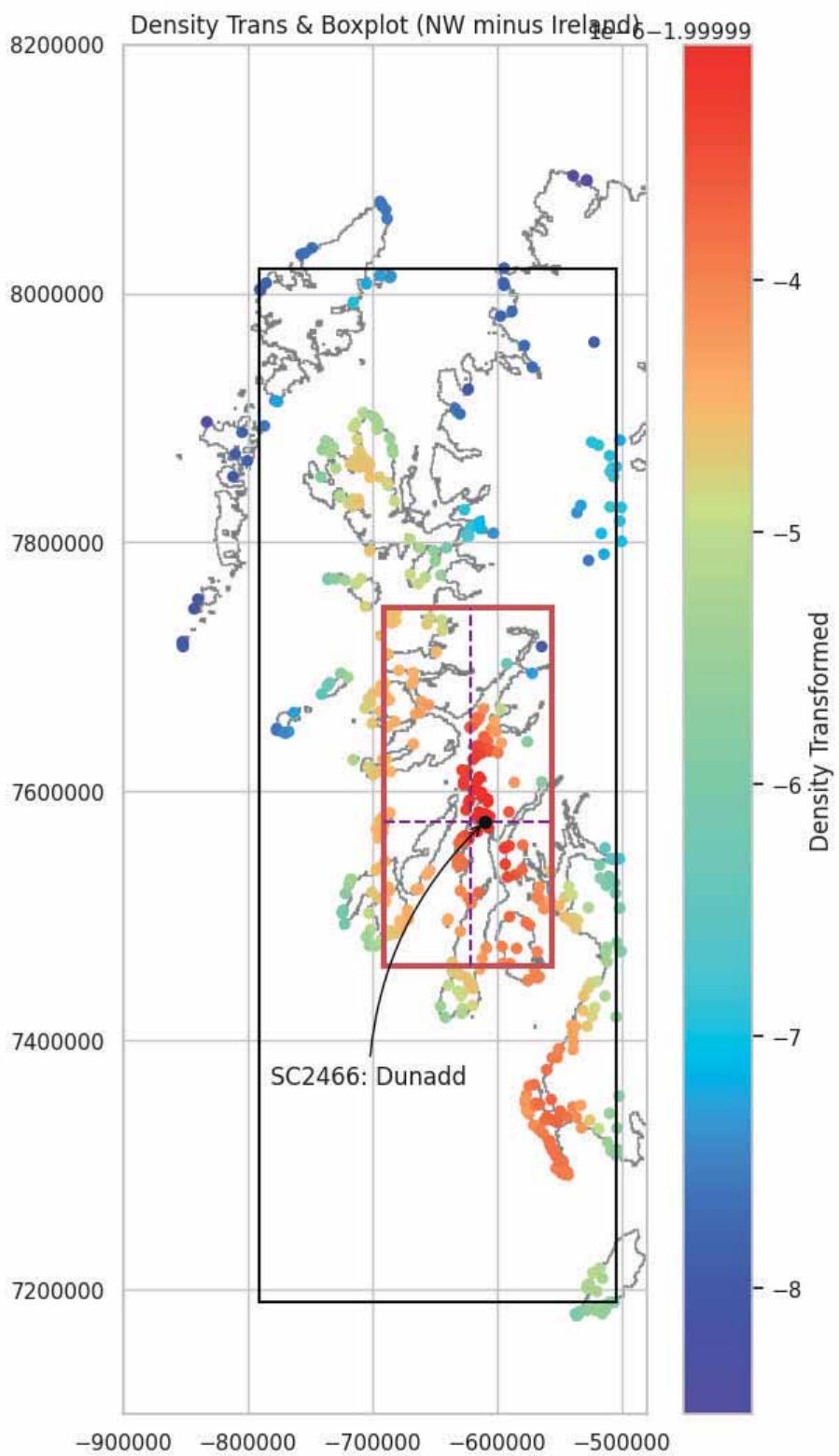
Location_Y - Northwest (minus Ireland). Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Northwestern Data Minus Ireland Mapped

Removing the Irish data focuses the boxplot, over the density cluster, near SC2466: Dunadd. This alignment of boxplot and cluster suggests the hillforts for this area of Northwest Scotland are not influenced by the distribution of hillforts in Ireland.

```
In [ ]: cluster_north_west = add_density(cluster_north_west)
cluster_north_west['Density_trans'] = \
stats.boxcox(cluster_north_west['Density'], 0.5)
```

```
In [ ]: plot_north_west_density_minus(show_elidon, cluster_north_west, \
                                     location_X_cluster_north_west, \
                                     location_Y_cluster_north_west)
```



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Northeast Density

The density values are refreshed in the clipped Northeastern data.

```
In [ ]: north_east = renew_densitiy(north_east)
```

There are 1598 forts in the Northeast data package.

```
In [ ]: north_east.info()
```

```

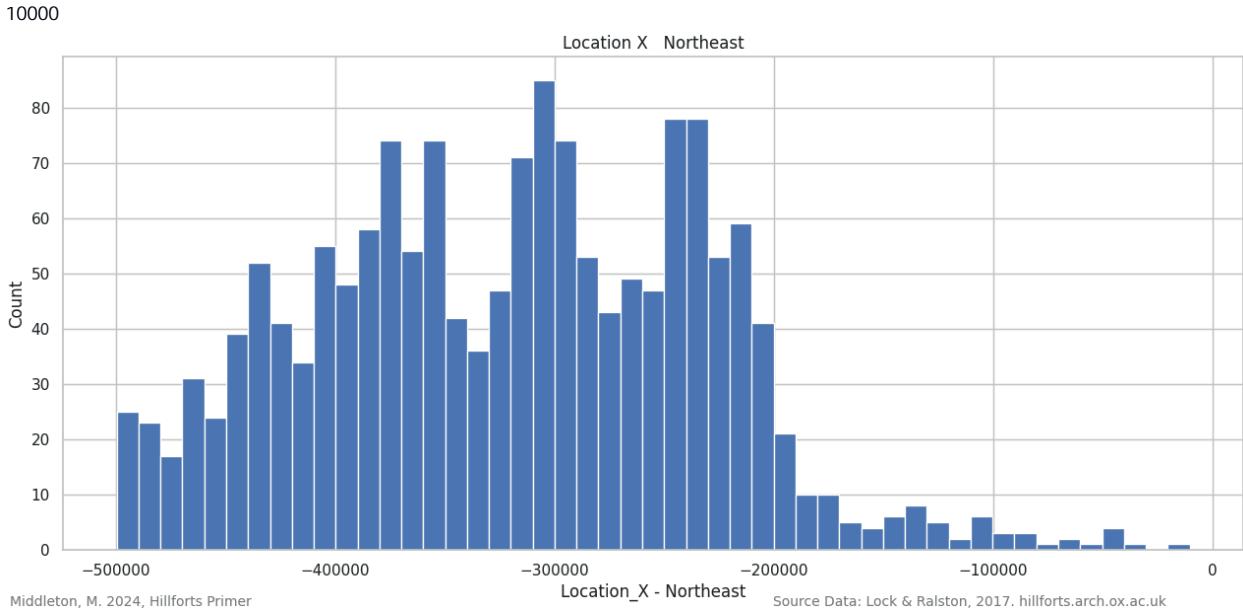
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1598 entries, 0 to 1597
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Location_X    1598 non-null   int64  
 1   Location_Y    1598 non-null   int64  
 2   Density        1598 non-null   float64
 3   Density_trans  1598 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 50.1 KB

```

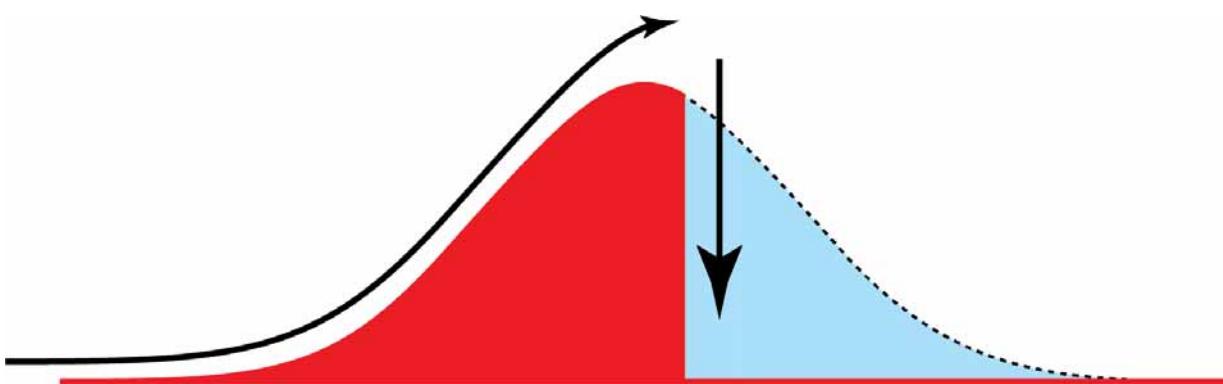
Northeast Data Plotted

There is an increasing concentration of hillforts from west to east up to the North Sea coast. At this point the distribution quite literally, 'falls off a cliff'. There is very little land to the east of -200,000.

```
In [ ]: plot_histogram(north_east['Location_X'], \
                      'Location_X - Northeast', 'Location_X - Northeast', 10000)
```



The density calculations assume the data will be spread across the entire area mapped. Where there is a peak in the data the density algorithm will expect the density to fall away on either side. The coast interrupts the expected spread, and the data is truncated immediately after the peak. This results in the boxplot being shifted west of where it would plot, if there were data to the east.

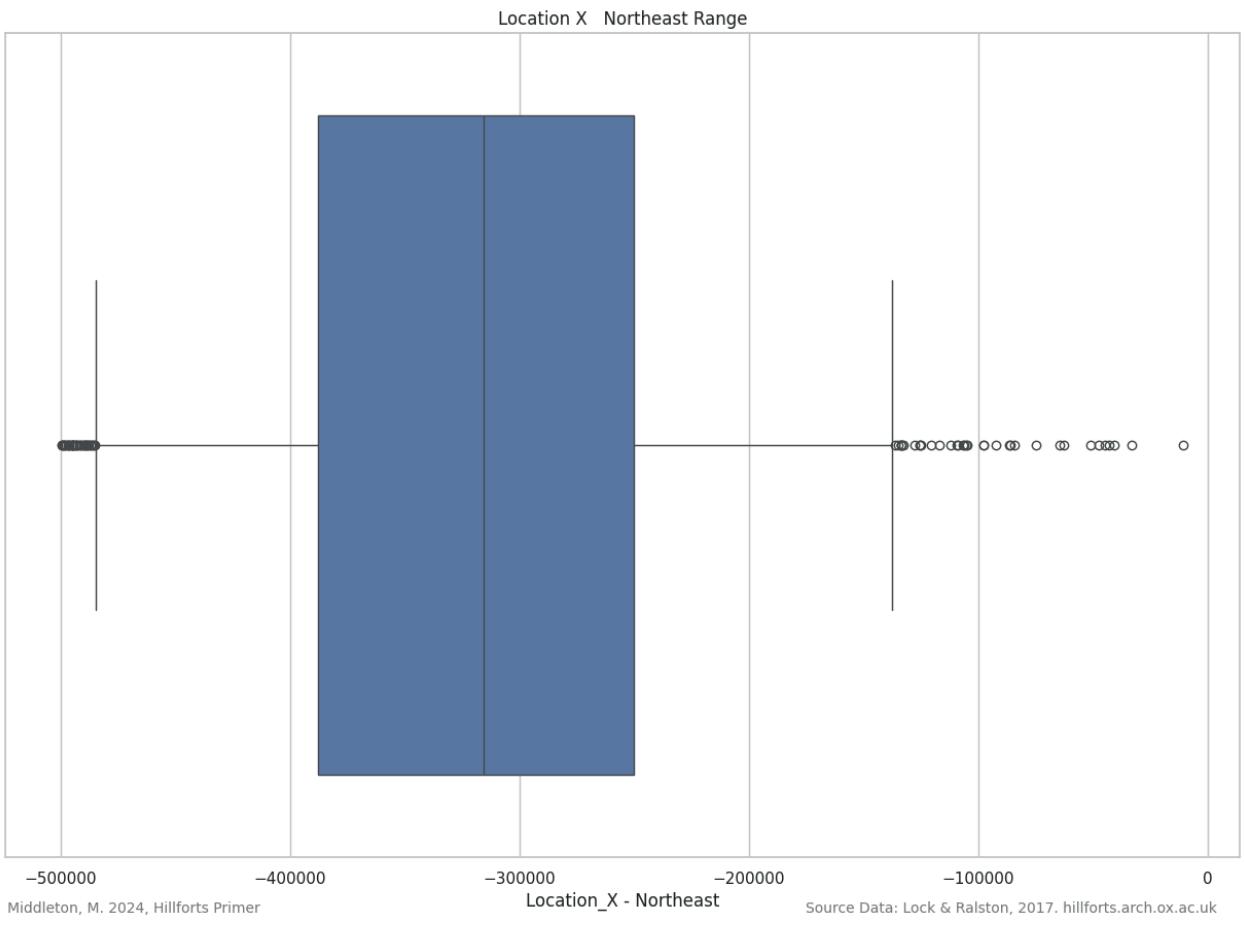


A data cliff. This figure shows how a peak in the data would be expected to plot in a normal distribution (blue) compared to an abstraction of how the 'Location_X' data actually plots in the Northeast, next to the coast (red) (see histogram above). As a boxplot is best used when the data has a normal distribution, in this case, the cliff causes the median value and the interquartile range to be offset to the left (west).

Mike Middleton *CC BY-SA 3.0*

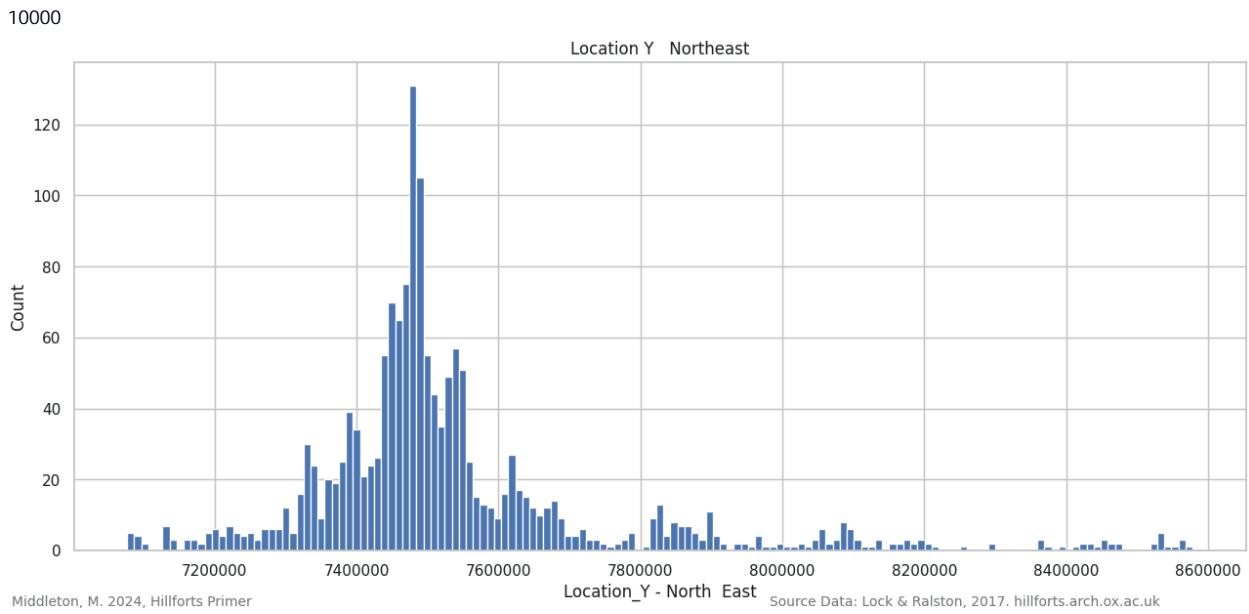
Because of the coast, the interquartile range is plotting further west than it should. The small number of forts in Northeastern England and the Northern Isles have created a long tail of outliers to the east.

```
In [ ]: location_X_north_e_data = plot_data_range(north_east['Location_X'], \
                                                 'Location_X - Northeast', "h")
```



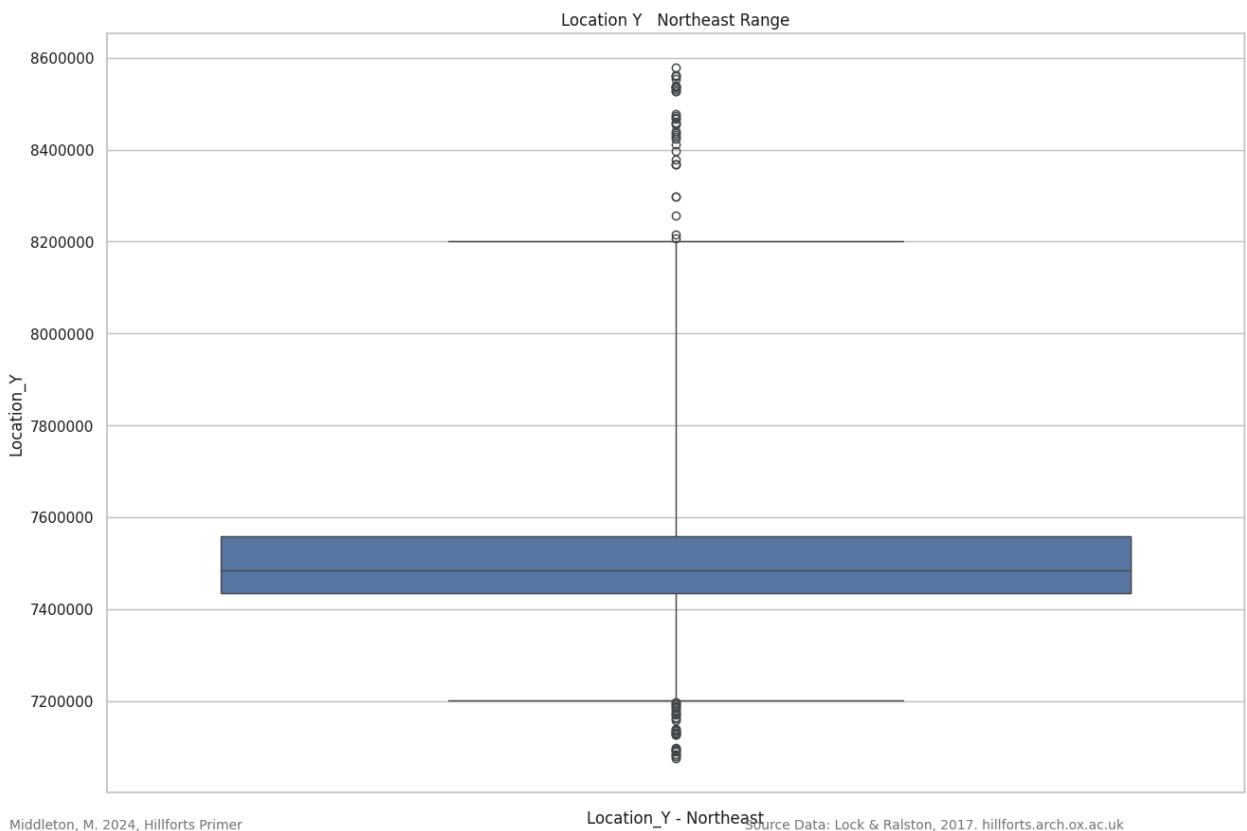
The peak around 7,500,000 dominates the Northeastern 'Location_Y' data and there is a very low density of hillforts north of 7,700,000.

```
In [ ]: plot_histogram(north_east['Location_Y'], 'Location_Y - North East', \
                      'Location_Y - Northeast', 10000)
```



The interquartile range in the 'Location_Y' data is concentrated in a very narrow band, between 7,300,000 and 7,500,000, over the eastern end of the Southern Uplands.

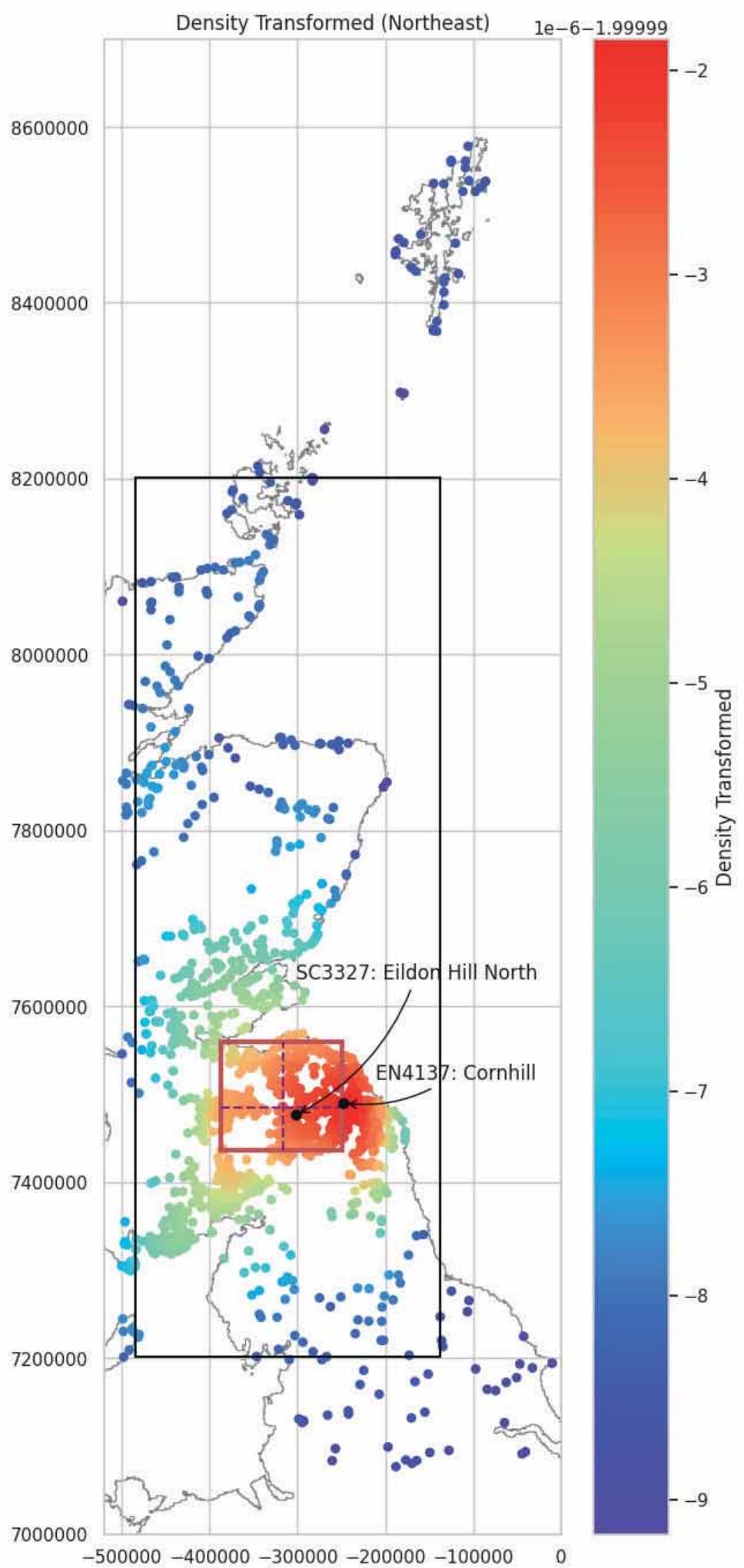
```
In [ ]: location_Y_north_e_data = \
plot_data_range(north_east['Location_Y'], 'Location_Y - Northeast')
```



Northeast Data Mapped

The boxplot is centred near SC3327 Eildon Hill North whereas the most dense concentration of forts is around EN4137 Cornhill and extends to include the entire Tweed Valley. As mentioned above, in Northeast Data Plotted, the coast is pushing the boxplot to the west. This being so, the boxplot and the cluster are relatively well aligned and there is no further distortion of the boxplot to indicate significant secondary clusters in this area.

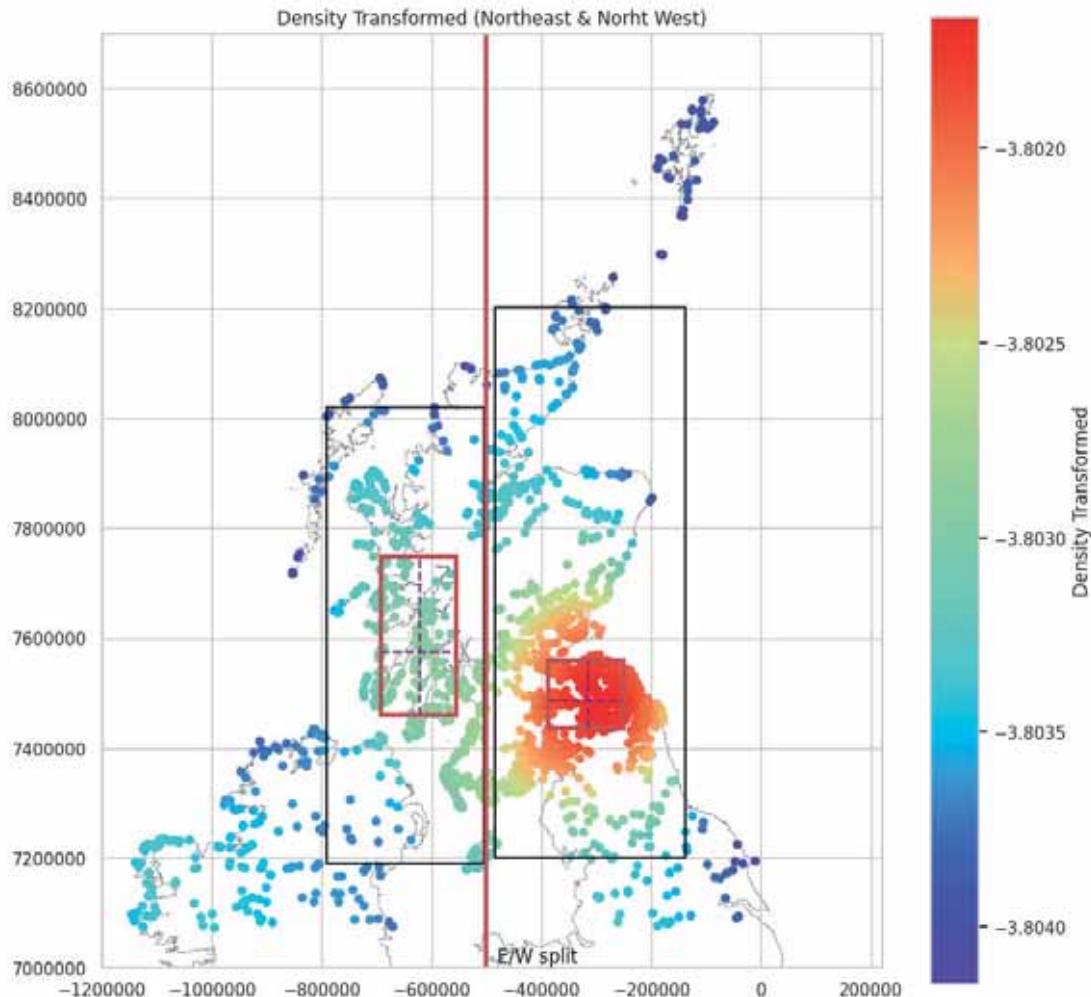
```
In [ ]: plot_north_east_density(show_eildon_north_east, location_Y_north_e_data)
```



Density Map showing Extent of the Northern Boxplots

The northern data contains two main clusters. The most intense is toward the east of the Southern Uplands, over the Tweed Valley. The second, when excluding the Irish forts, is focused on the west coast around SC2466: Dunadd.

```
In [ ]: plot_northern_densitiy_boxplots(show_eildon, north, \
                                         location_X_cluster_north_west, \
                                         location_Y_cluster_north_west)
```



See: [Northwestern Data Minus Ireland Mapped](#)

See: [Northeast Data Mapped](#)

Southern Density

Southern Location Data Plotted

There are 1833 records in the southern data package.

```
In [ ]: south.info()
```

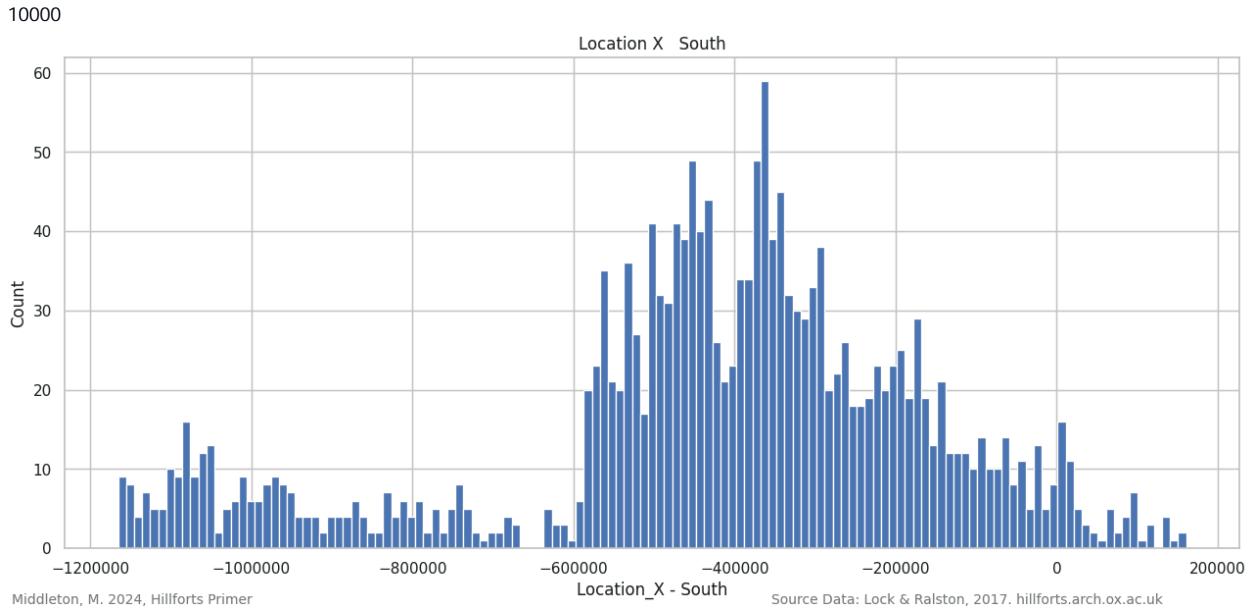
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1833 entries, 0 to 1832
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Location_X    1833 non-null   int64  
 1   Location_Y    1833 non-null   int64  
 2   Density        1833 non-null   float64 
 3   Density_trans  1833 non-null   float64 
dtypes: float64(2), int64(2)
memory usage: 57.4 KB

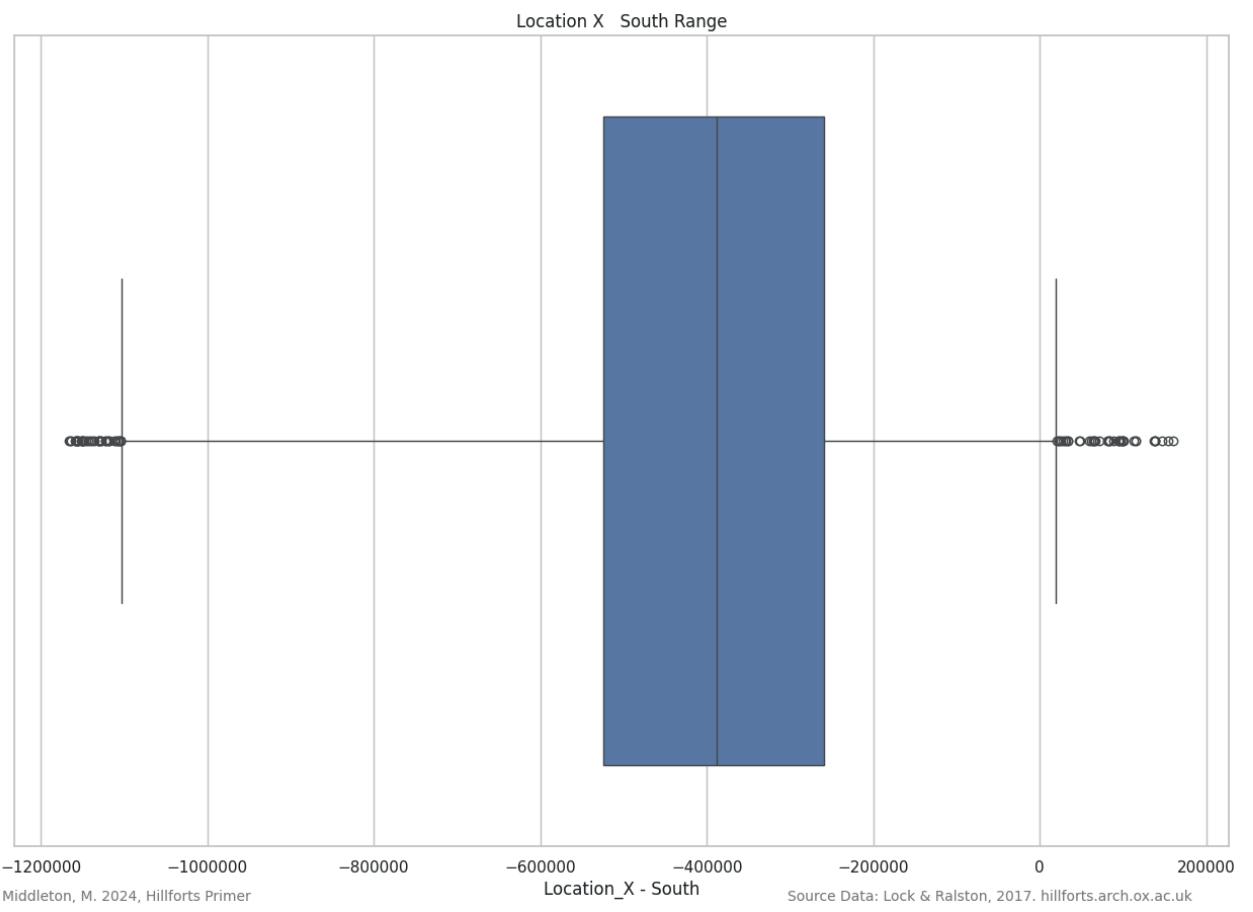
```

There are two distributions of data in the southern package. To the west, a low density of hillforts over Ireland and to the east, a more intense concentration of hillforts over Wales and into England. In the Welsh and English data there is a broad peak that increases in density from east to west. The highest peak is at -370,000 while a second peak can be seen at -450,000. There is a very low density of forts toward the east of England.

```
In [ ]: plot_histogram(south['Location_X'], 'Location_X - South', \
                      'Location_X - South', 10000)
```

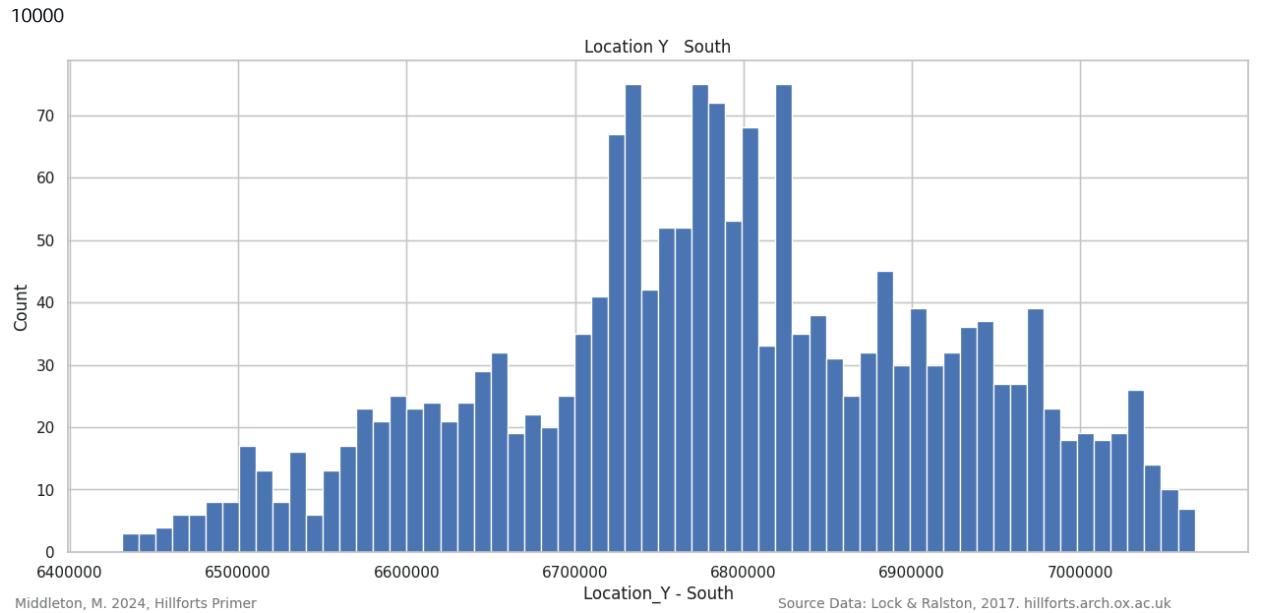


```
In [ ]: location_X_south_data = plot_data_range(south['Location_X'], \
                                              'Location_X - South', "h")
```

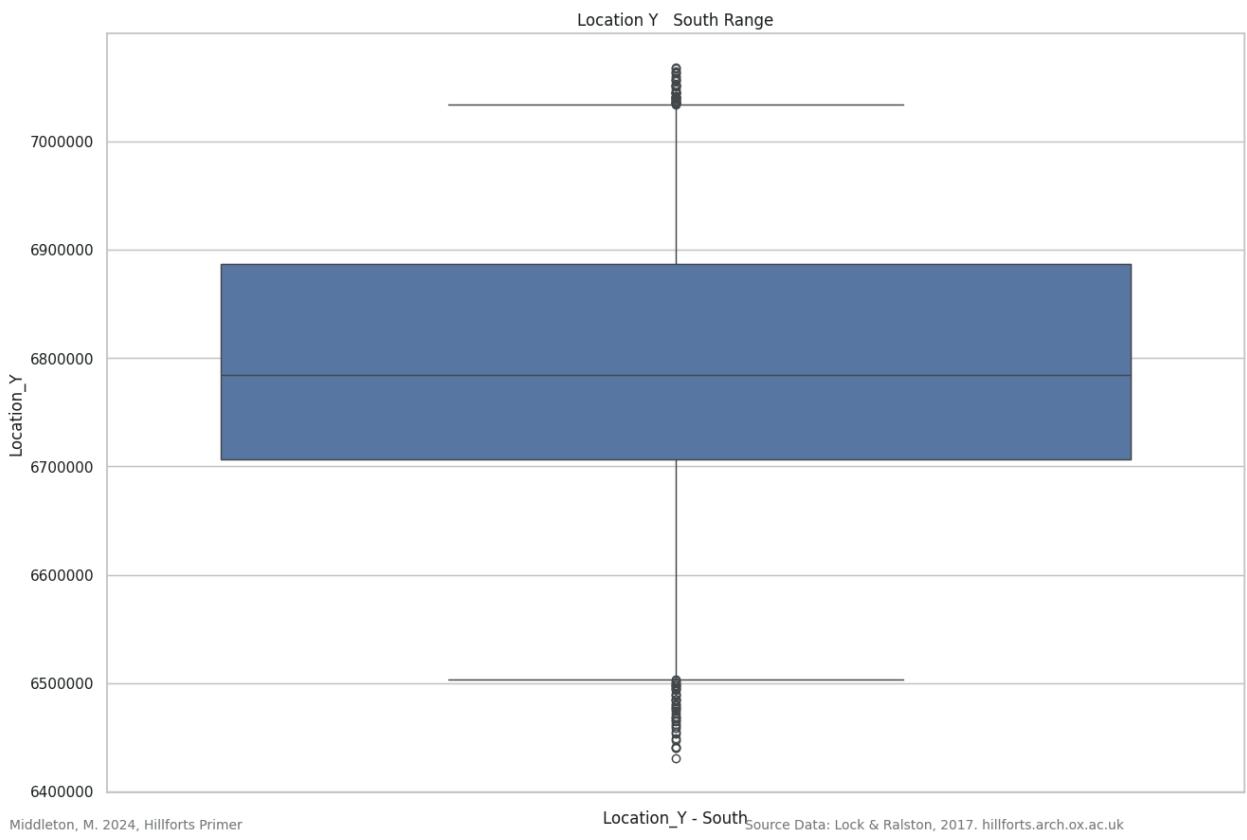


There is a broad peak in the 'Location_Y' data. Toward the centre of the broad peak, a narrow band of high density peaks can be seen between 6,720,000 to 6,820,000.

```
In [ ]: plot_histogram(south['Location_Y'], 'Location_Y - South', \
                      'Location_Y - South', 10000)
```



```
In [ ]: location_Y_south_data = plot_data_range(south['Location_Y'], \
                                               'Location_Y - South')
```

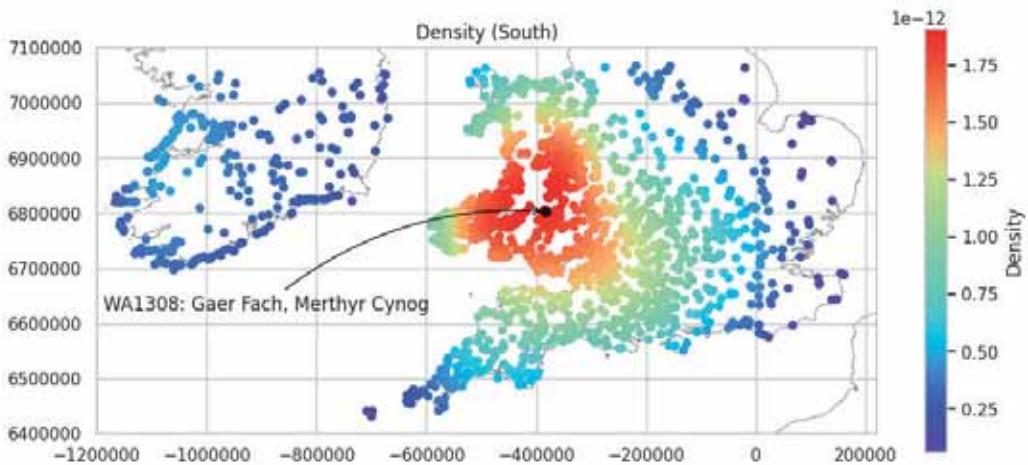


Southern Data Density Mapped

The southern cluster of hillforts is focussed near WA2273: Sugar Loaf, around the Abergwesyn Pass, which is located between the Brecon Beacons, to the south, and the Cambrian mountains, to the north.

```
In [ ]: show_gaer_fach = True
```

```
In [ ]: plot_southern_density(show_gaer_fach, south)
```



Southern Data Density Mapped (top 5%)

The most dense concentration of forts can be found between Llandeilo, Lampeter, Llandrindod and Brecon. The higher ground, along the north-south aligned Cambrian Mountains, are clear of hillforts as are the south facing slopes, along the Abergwesyn Pass between Llanwrtyd Wells and Llandrindod.

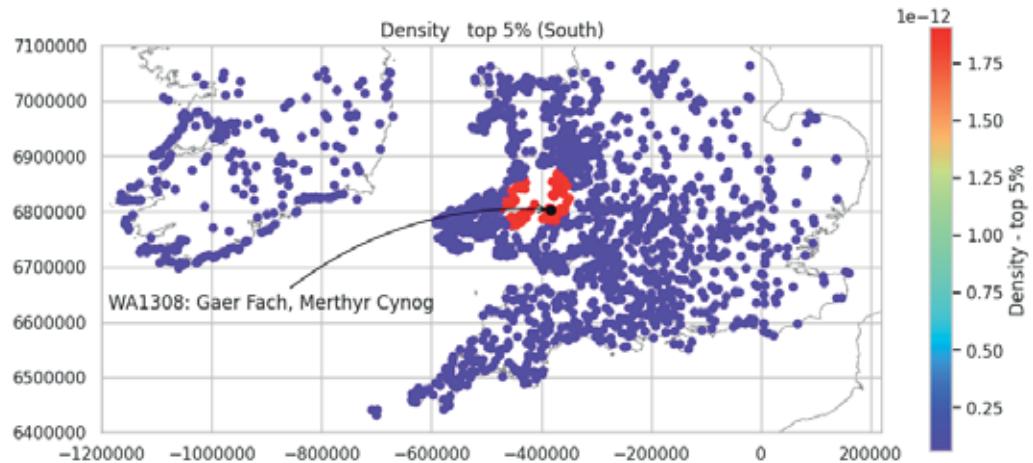
```
In [ ]: south_top5 = south.copy()
south_top5['Density'].where(south_top5['Density'] > \
                           south_top5['Density'].quantile(0.95), \
                           south_top5['Density'].min(), inplace=True)
south_top5['Density'].describe()
```

```

Out[ ]: count    1.833000e+03
         mean     1.507099e-13
         std      3.951188e-13
         min      5.991116e-14
         25%     5.991116e-14
         50%     5.991116e-14
         75%     5.991116e-14
         max      1.908704e-12
Name: Density, dtype: float64

```

```
In [ ]: plot_south_top_5(show_gaer_fach, south_top5)
```



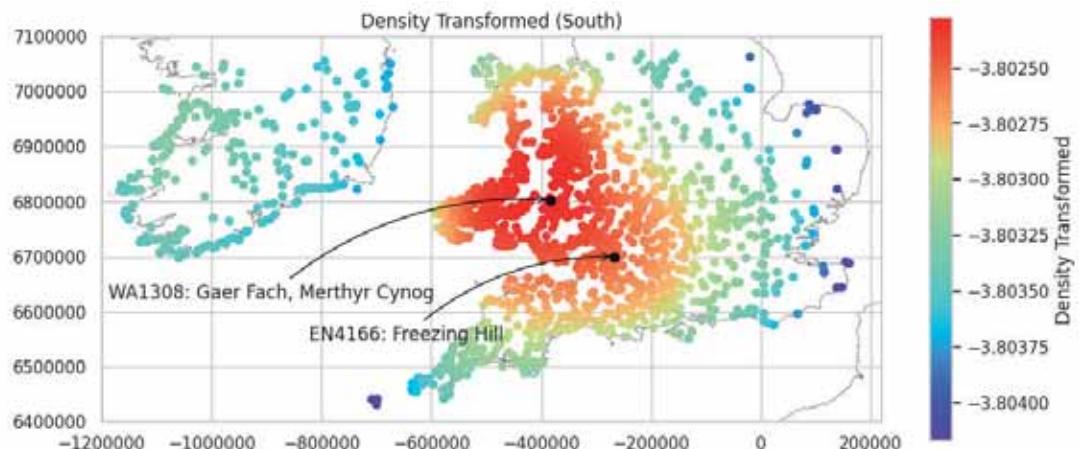
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Southern Data Density Mapped (Transformed)

The transformed density shows a secondary cluster of sites along the upper reaches of the Severn Estuary, centred in the region of EN4166: Freezing Hill.

```
In [ ]: plot_south_density_transformed(show_gaer_fach, south)
```



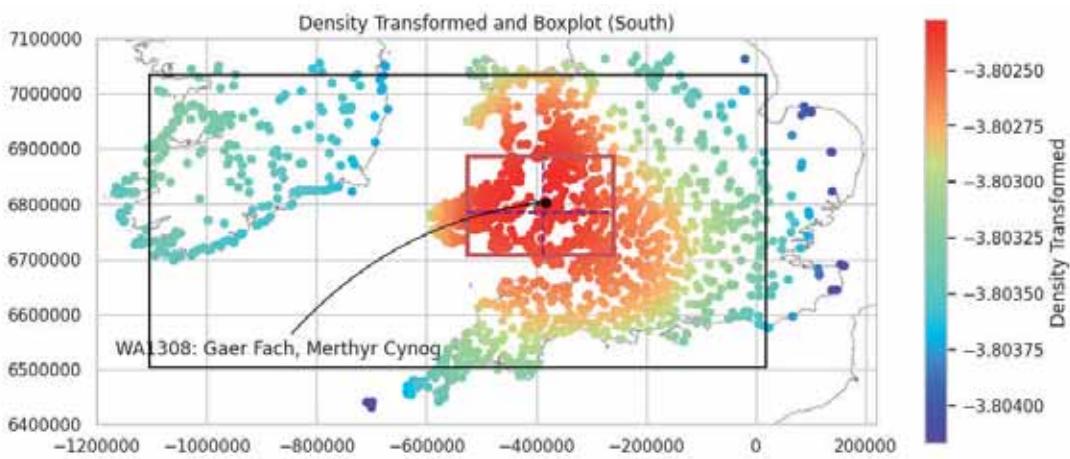
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Southern Density Boxplot

The southern boxplot aligns with the density plot in [Southern Data Density Mapped](#) and is focussed near WA1308: Gaer Fach.

```
In [ ]: plot_southern_density_boxplot(show_gaer_fach, \
                                     south, location_X_south_data, location_Y_south_data)
```



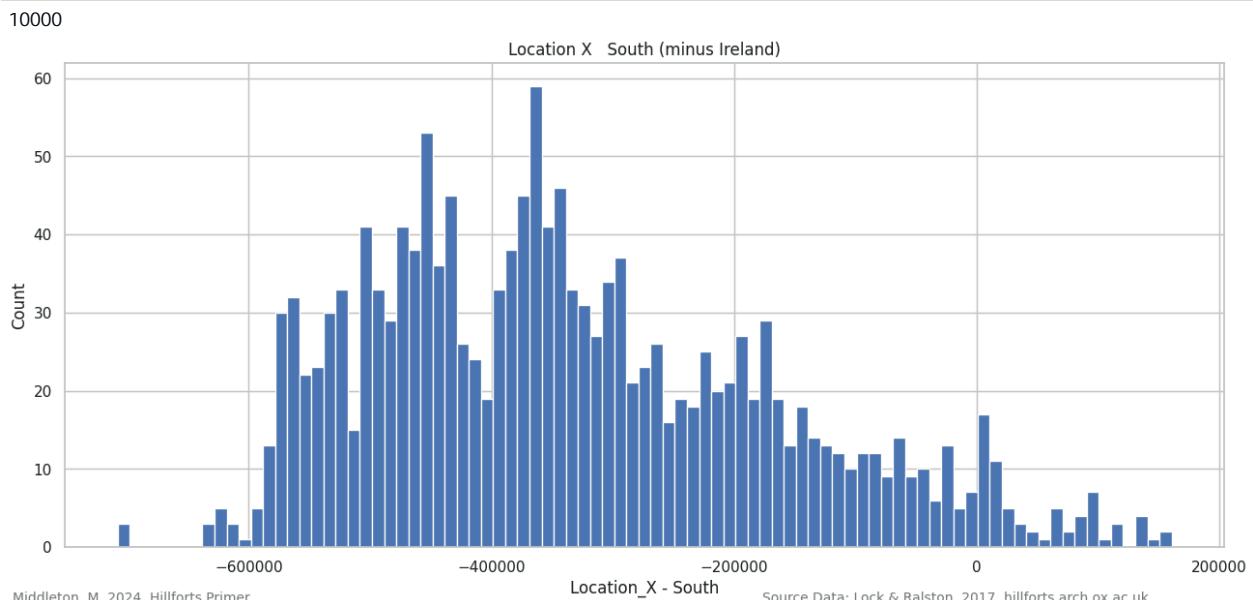
See: [Southern Location Data Plotted](#)

See: [Southern Density Boxplot Minus Ireland](#)

Southern Location Data Minus Ireland Plotted

The histogram now contains only the southern data, extending over Wales and England. See: [Southern Location Data Plotted](#)

```
In [ ]: plot_histogram(cluster_south['Location_X'], 'Location_X - South', \
                     'Location_X - South (minus Ireland)', 10000)
```

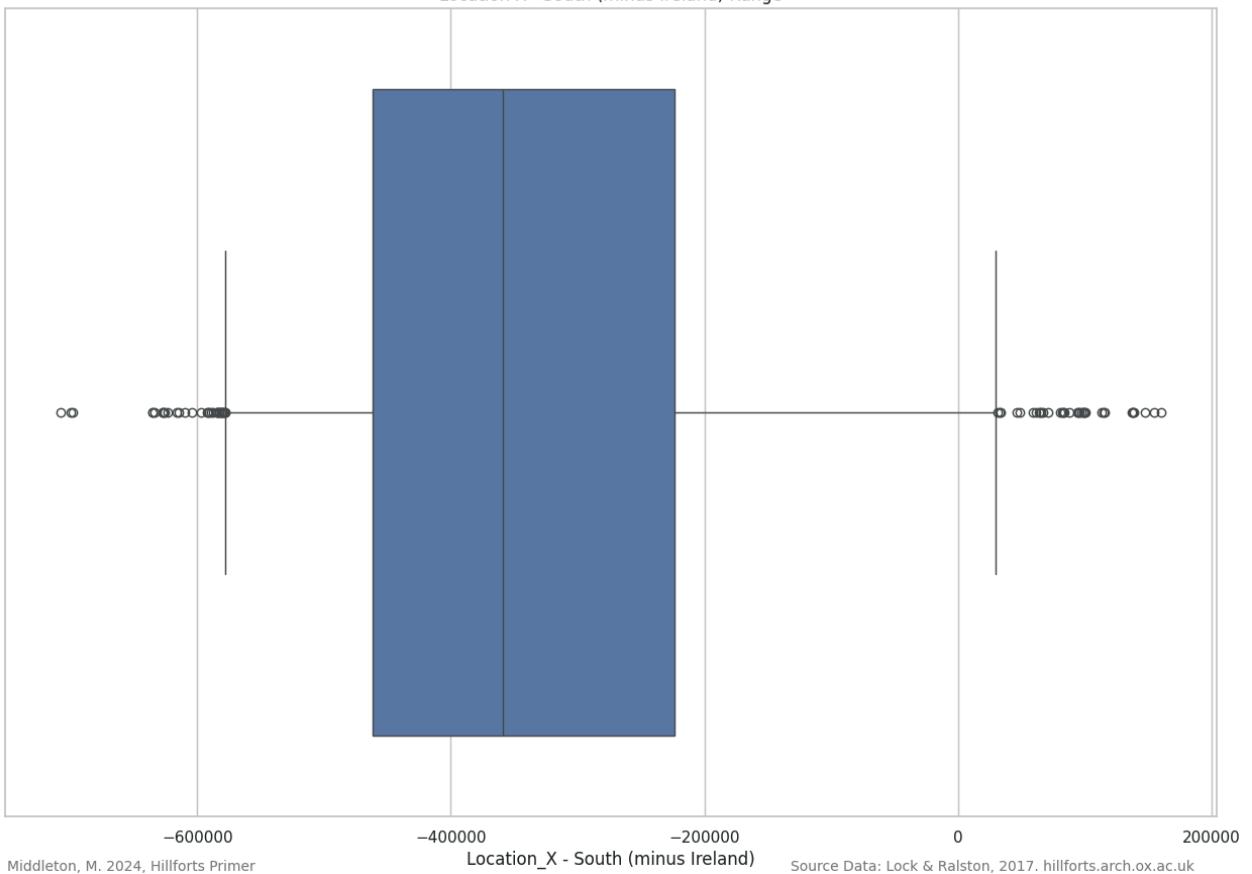


By removing the Irish data, the interquartile range has moved east. The outliers to the west are all forts at the extreme south of the South West peninsula and the Isles of Scilly. To the east, the outliers are the low density of forts along the eastern coast of England from The Wash to the Strait of Dover.

```
In [ ]: cluster_south = cluster_south.copy().reset_index()
```

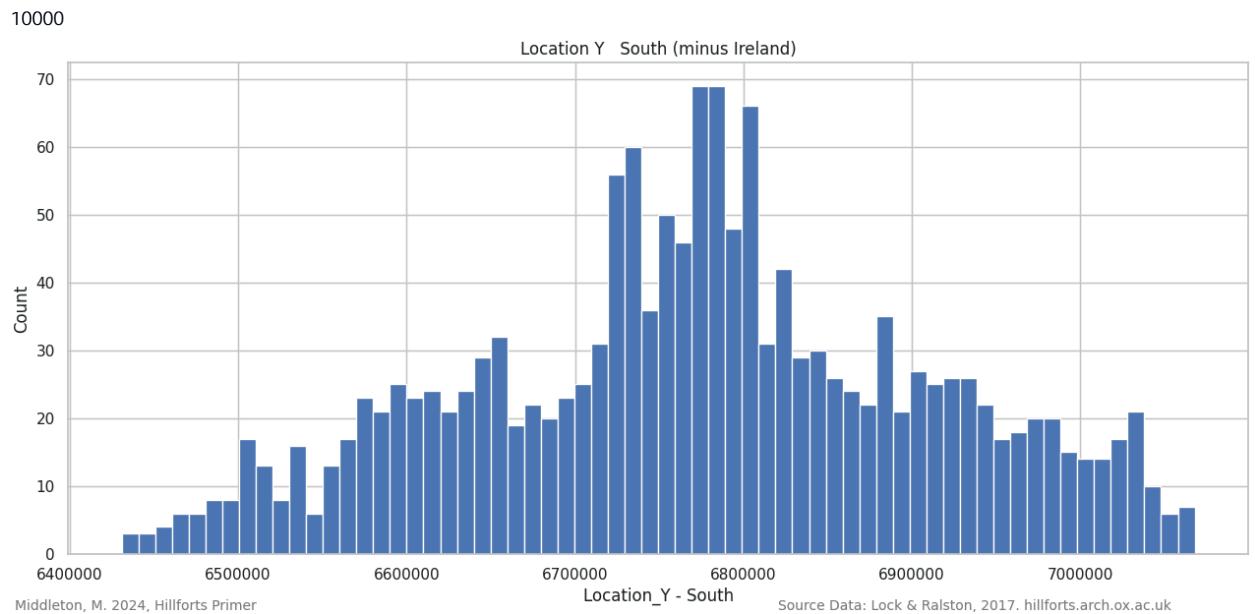
```
In [ ]: location_X_cluster_south = \
plot_data_range(cluster_south["Location_X"], \
                 'Location_X - South (minus Ireland)', "h")
```

Location X - South (minus Ireland) Range



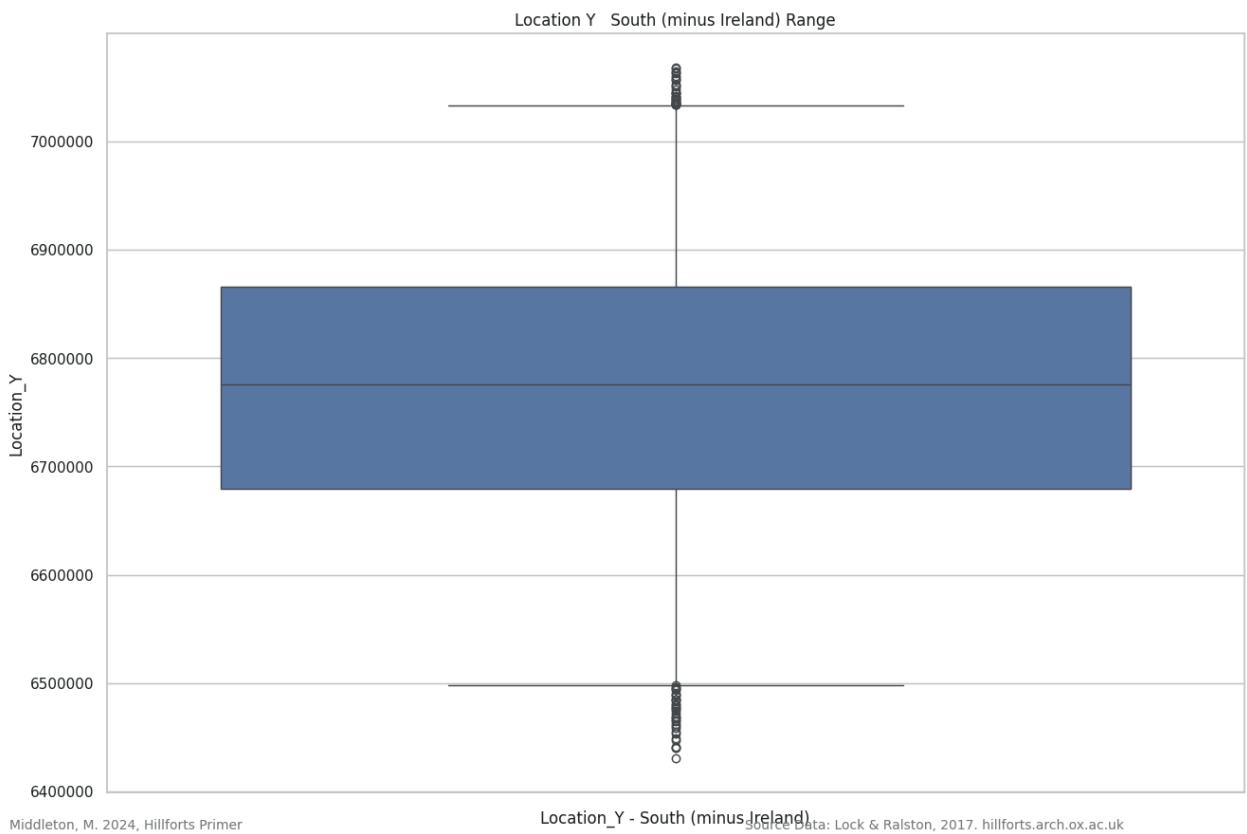
The broad, relatively smooth peak, covers the full range from north to south. The high peak at around 6,720,000 aligns with the southern coastline of Wales and the highest peak at 6,780,000 corresponds to one of the widest parts of the island.

```
In [ ]: plot_hi_stogram(cluster_south['Location_Y'], \
'Location_Y - South', 'Location_Y - South (minus Ireland)', 10000)
```



The 'Location_Y' interquartile range is almost unchanged after removing the Irish data. The outliers to the south are again the forts at the extreme south of the South West peninsula and the Isles of Scilly. To the north, the outliers are the low density of forts in the north of England and the forts along the north coast of Wales.

```
In [ ]: location_Y_cluster_south = \
plot_data_range(cluster_south['Location_Y'], \
'Location_Y - South (minus Ireland)')
```



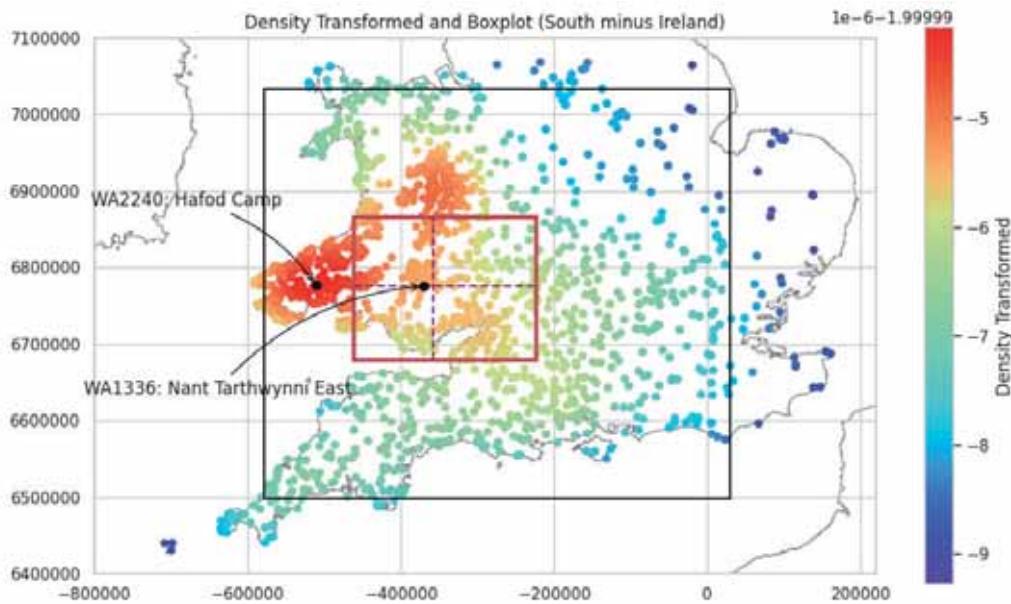
Southern Density Boxplot Minus Ireland

Removing the Irish data improves the definition of the southern data. There is an intense cluster along the Pembrokeshire peninsula, over the Preseli Hills. What could be a second cluster can be seen, to the east of the Cambrian Mountains, over the Shropshire Hills and, a possible third cluster spreads south east from the Brecon Beacons, to the North Wessex Downs. Because these clusters are on either side of the high ridges of the Cambrian mountains, it is unclear if these clusters are distinct or if they are manifestations of the same large cluster, split by topography.

As seen in [Northeast Data Plotted](#), the Pembrokeshire cluster is on the coast and this has the effect of pushing the boxplot to the east. Even so, the boxplot is further east than would be anticipated and this is likely due to the intensity of the cluster around EN4166: Freezing Hill discussed in [Southern Data Density Mapped \(Transformed\)](#).

```
In [ ]: cluster_south = add_density(cluster_south)
cluster_south['Density_trans'] = stats.boxcox(cluster_south['Density'], 0.5)
```

```
In [ ]: plot_south_density_transformed_boxplot(show_gaer_fach, \
cluster_south, location_X_cluster_south, \
location_Y_cluster_south)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Ireland Density

Ireland Location Data Plotted

There are 507 hillforts in the Irish data.

```
In [ ]: location_features_short = [
    'Location_X',
    'Location_Y']

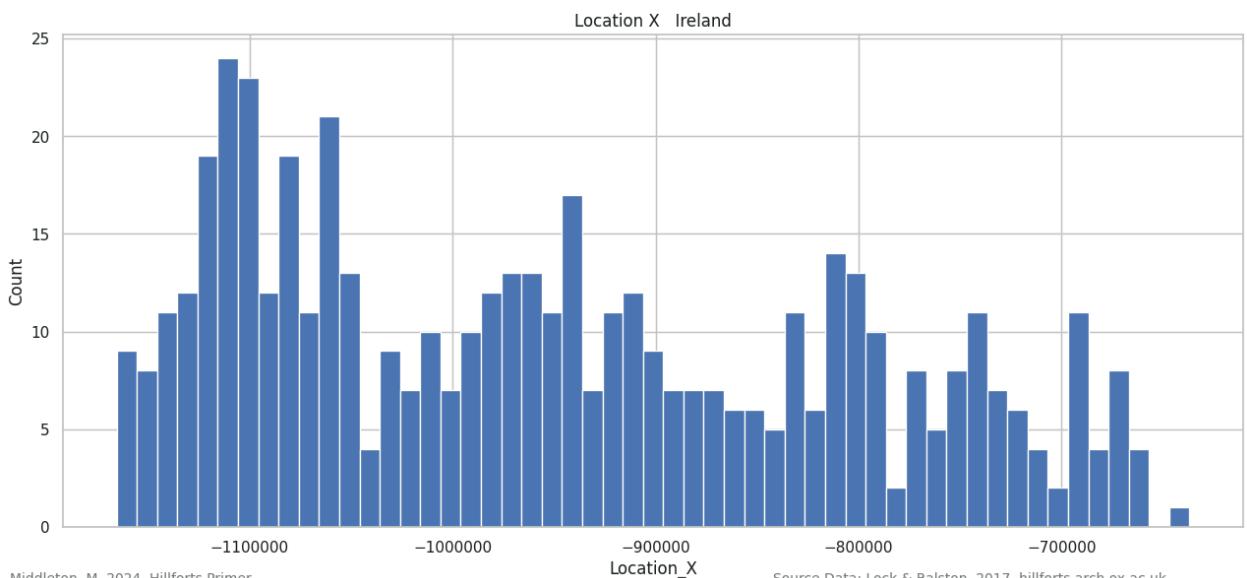
ireland_location_data = roi_and_ni_data[location_features_short].\
copy().reset_index(drop=True)
```

```
In [ ]: ireland_location_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 507 entries, 0 to 506
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   Location_X  507 non-null   int64  
 1   Location_Y  507 non-null   int64  
dtypes: int64(2)
memory usage: 8.0 KB
```

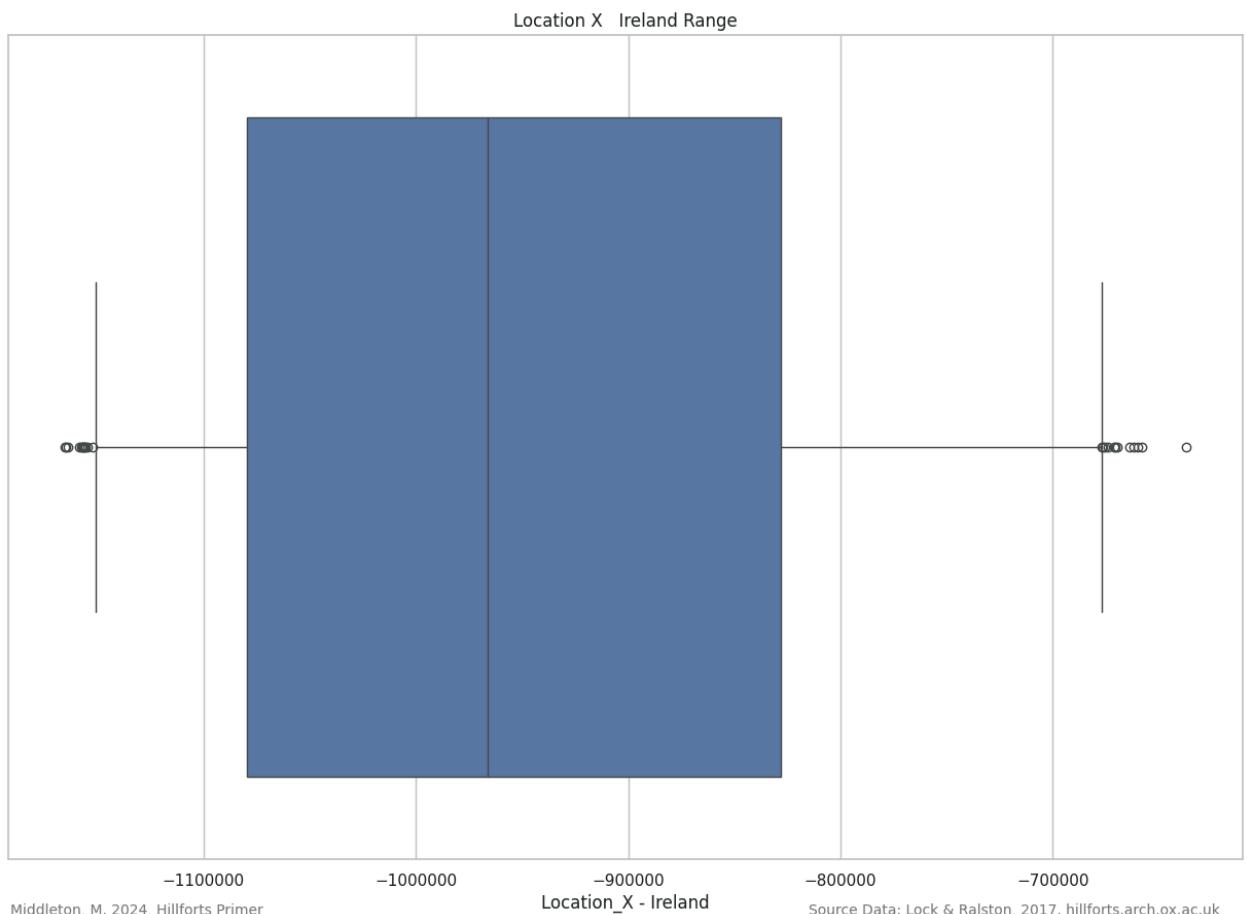
The 'Location_X' data shows a gradual increase in hillfort density to the west. The largest peak is at -11,000,000, to the far west.

```
In [ ]: plot_histogram(ireland_location_data['Location_X'], \
    'Location_X', 'Location_X - ireland', 10000)
```



The 'Location_X' boxplot shows a broad interquartile range set toward the west. The broad IQR indicates that the peaks are not that intense and that the forts are relatively evenly spread over a wide range.

```
In [ ]: location_X_ireland_and_data = \
    plot_data_range(ireland_and_location_data['Location_X'], \
        'Location_X - Ireland', "h")
```

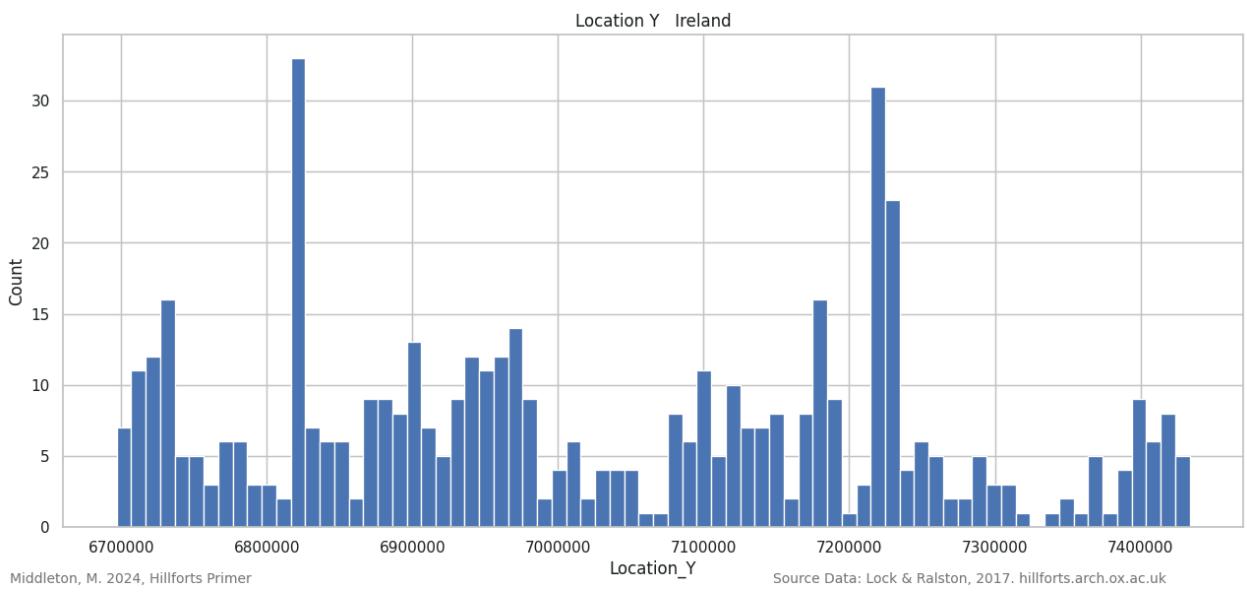


The 'Location_Y' histogram shows a number of shallow peaks and two, taller, narrow peaks. The peak at 6,820,000 corresponds to an unconnected grouping of forts along the south east coast of Ireland from Rosslare to Waterford and another group of forts along the Dingle Peninsula. This peak is a result of a coincidental topographic alignment.

The second peak at 7,220,000 corresponds to a true peak in the data along the northern coastline of the Mullet Peninsula. Again, this peak is exaggerated due to the coincidental alignment of the coast, perpendicular to the 'Location_Y' axis.

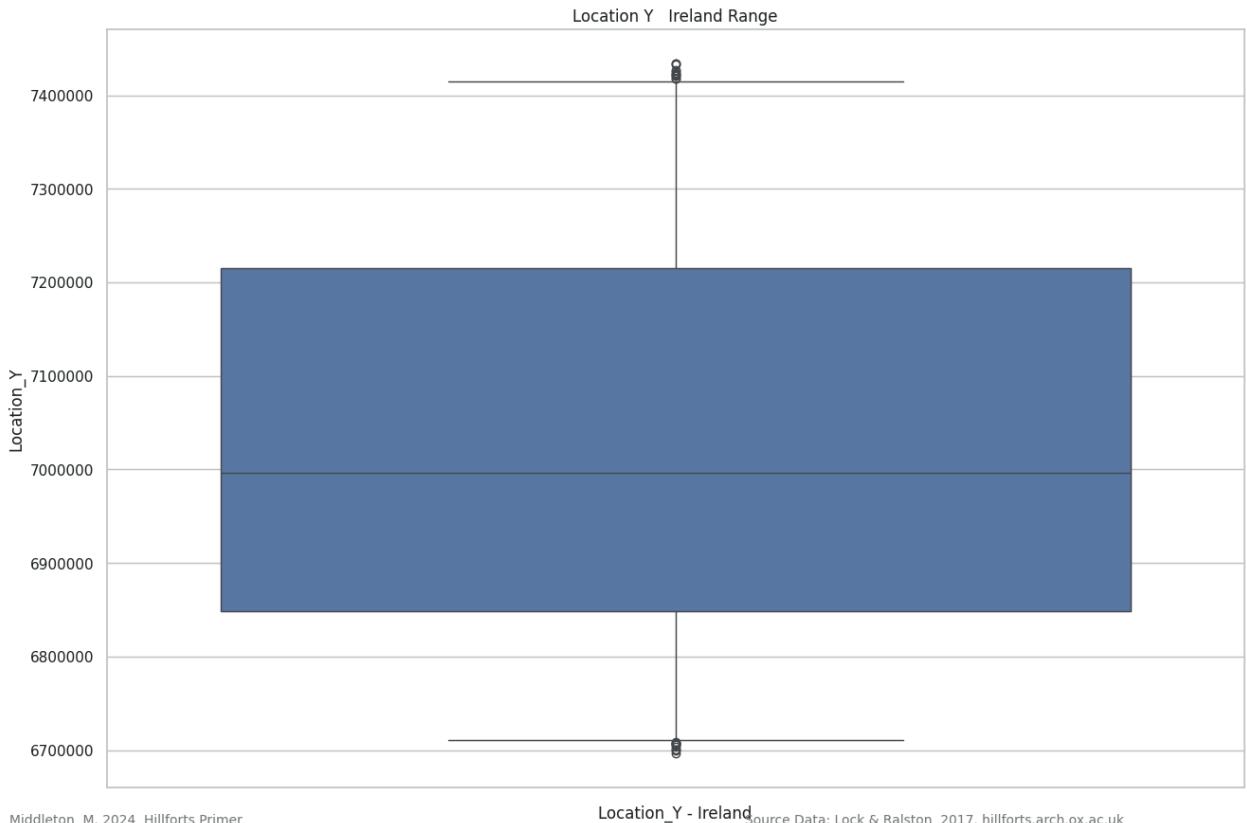
```
In [ ]: plot_histogram(ireland_and_location_data['Location_Y'], \
    'Location_Y', 'Location_Y - Ireland', 10000)
```

10000



As in the 'Location_X' boxplot above, the 'Location_Y' boxplot shows a broad interquartile range indicating that the peaks are not that intense and that the forts are relatively evenly spread over a whole range.

```
In [ ]: location_Y_ireland_data = \
    plot_data_range(ireland_and_location_data['Location_Y'], 'Location_Y - Ireland')
```



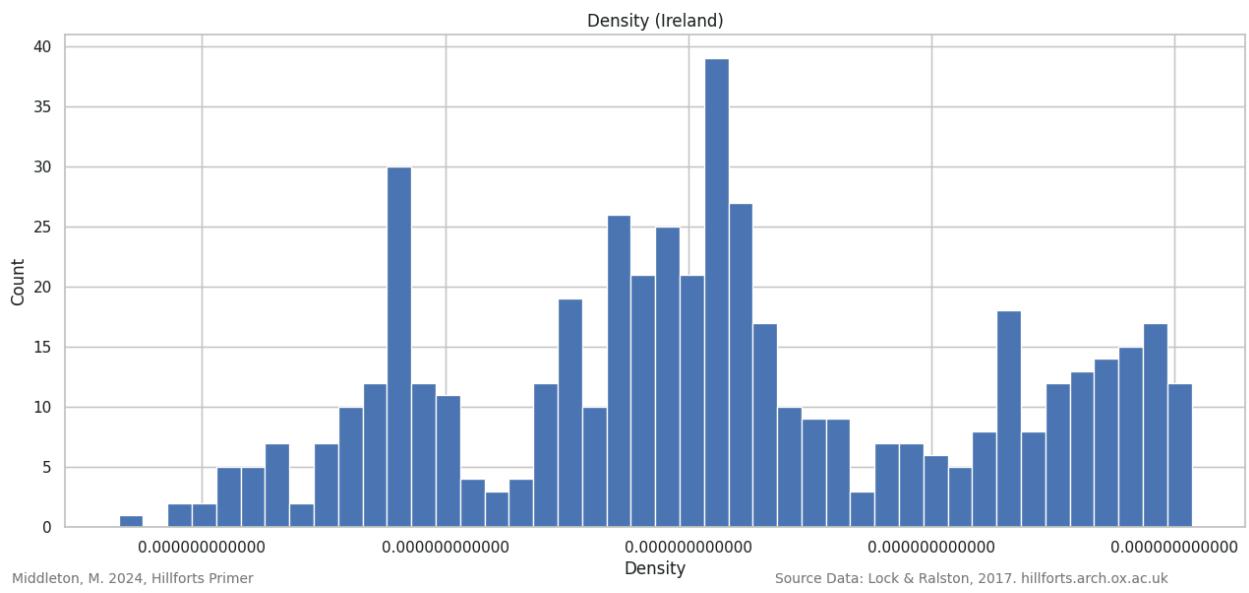
Ireland Density Data Plotted

The data is well distributed in the Irish density histogram. Most of the data is toward the centre of the plot and this should lead to a good mapped visualisation.

```
In [ ]: ireland_density_data = add_density(ireland_and_location_data).reset_index(drop=True)
```

```
In [ ]: plot_histogram(ireland_density_data['Density'], 'Density', 'Density (Ireland)')
```

None

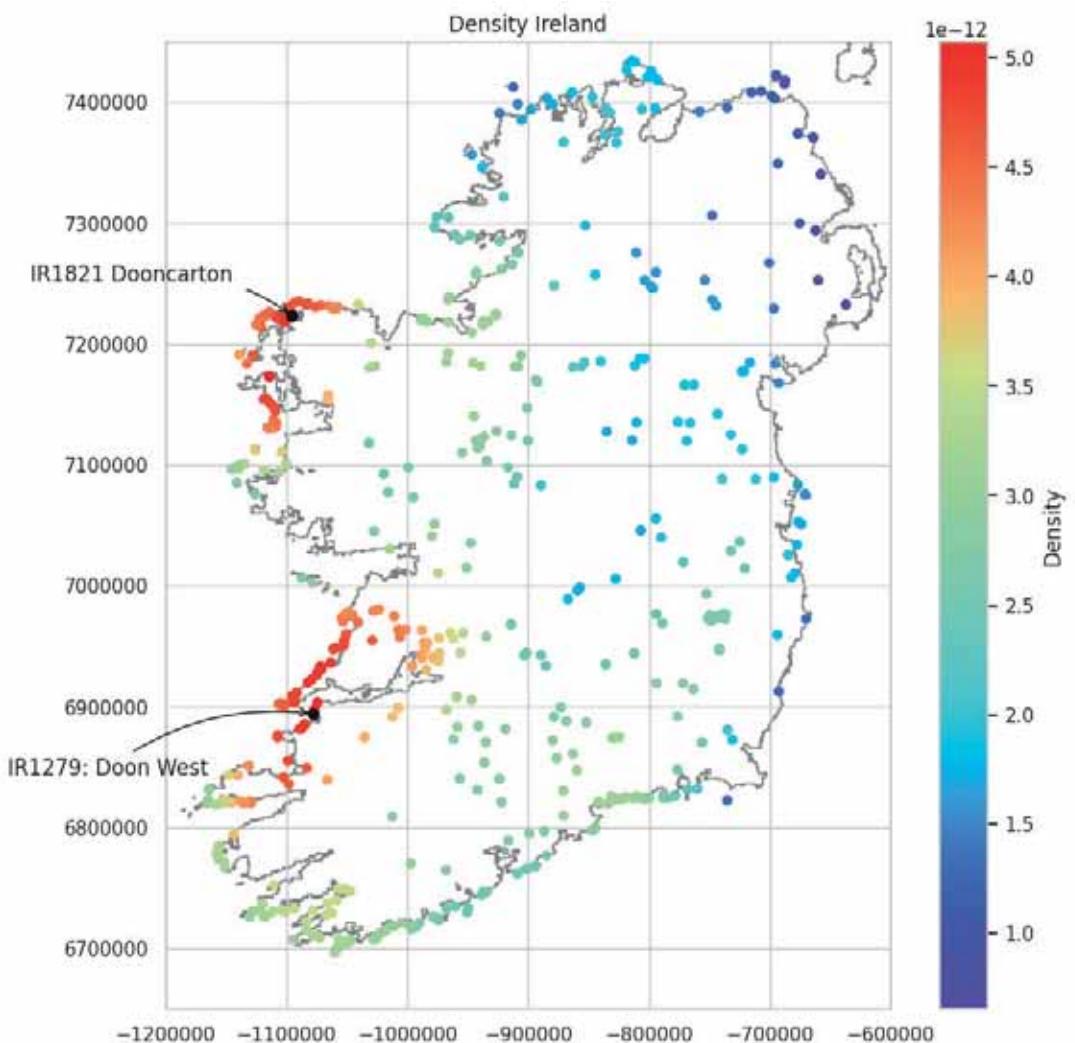


Ireland Density Data Mapped

There are two clusters on the west coast. To the north, around IR1821 Dooncarton and down along the Duvillaun, Achill and Inishkea islands. A second, toward the south is located around IR1279: Doon West. There is a general increase in hillfort concentration to the south and west, with the least dense concentration of forts being along the Northeast coast.

```
In [ ]: show_dooncarton = True
```

```
In [ ]: plot_density_ireland(show_dooncarton, ireland_density_data)
```



See: [Location Data: Density Data Mapped](#).

See: [Location Data: Density Data Transformed Mapped](#).

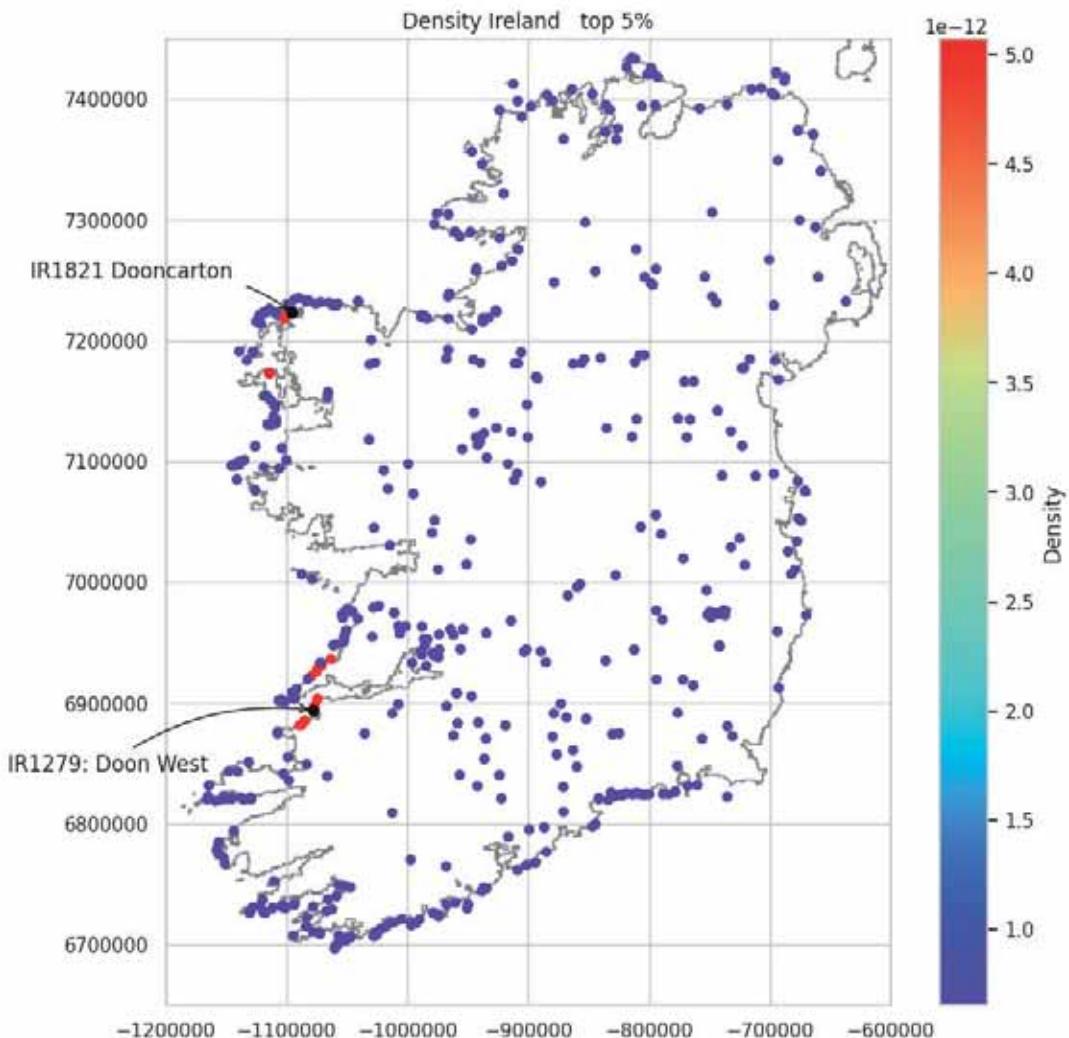
Ireland Data Density Mapped (top 5%)

The most dense concentration of forts, the top 5%, is split between the two west coast clusters.

```
In [ ]: ireland_density_top5 = ireland_density_data.copy()
ireland_density_top5['Density'].where(ireland_density_top5['Density'] > \
    ireland_density_top5['Density'].quantile(0.95), \
    ireland_density_top5['Density'].min(), \
    inplace=True)
ireland_density_top5['Density'].describe()
```

```
Out[ ]: count      5.070000e+02
mean       8.782268e-13
std        9.531839e-13
min       6.568557e-13
25%       6.568557e-13
50%       6.568557e-13
75%       6.568557e-13
max       5.072162e-12
Name: Density, dtype: float64
```

```
In [ ]: plot_density_ireland_top5(show_dooncarton, ireland_density_top5)
```



Ireland Data Split

Split Clusters in Irish Data

The 'Location_Y' data shows there is a trough, between the two main clusters, at around 7,060,000. See: [Ireland Location Data Plotted](#).

```
In [ ]: split_location = 7060000
south_cluster = ireland_density_data[ireland_density_data['Location_Y'] < \
                                    split_location].copy().reset_index(drop=True)
north_cluster = ireland_density_data[ireland_density_data['Location_Y'] >= \
                                    split_location].copy().reset_index(drop=True)
```

There are 278 records in the South Ireland data package.

```
In [ ]: south_cluster.info()
```

#	Column	Non-Null Count	Dtype
0	Location_X	278	int64
1	Location_Y	278	int64
2	Density	278	float64
3	Density_trans	278	float64

There are 229 records in the North Ireland data package.

```
In [ ]: north_cluster.info()
```

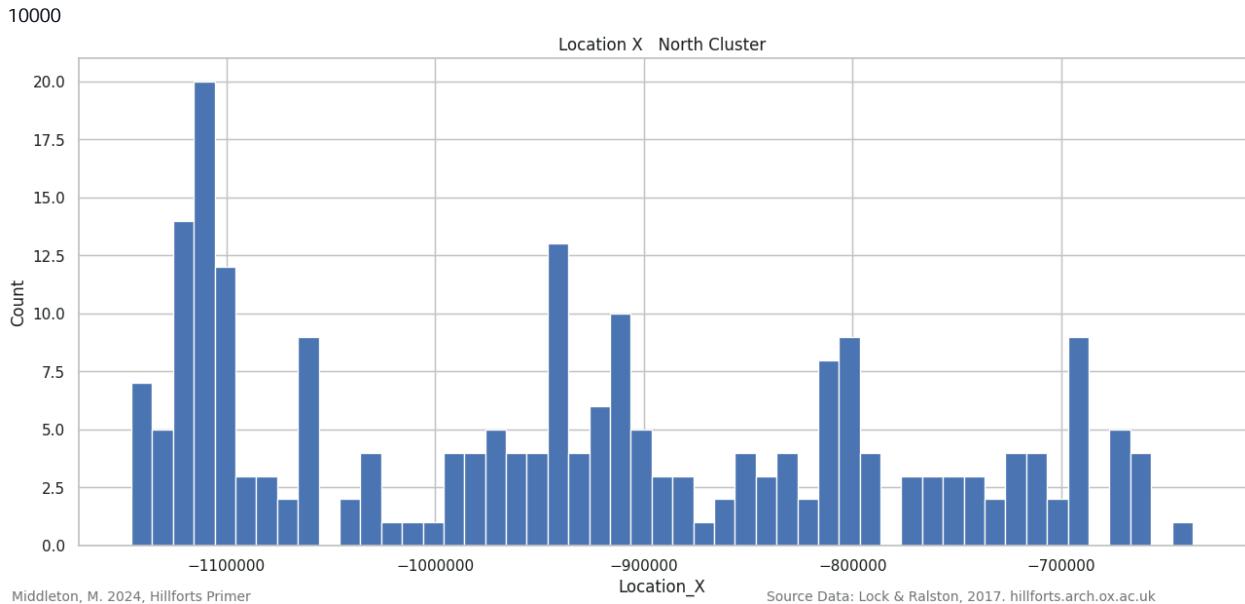
#	Column	Non-Null Count	Dtype
0	Location_X	229	int64
1	Location_Y	229	int64
2	Density	229	float64
3	Density_trans	229	float64

North Ireland Density

Nothern Location Data Plotted (Ireland)

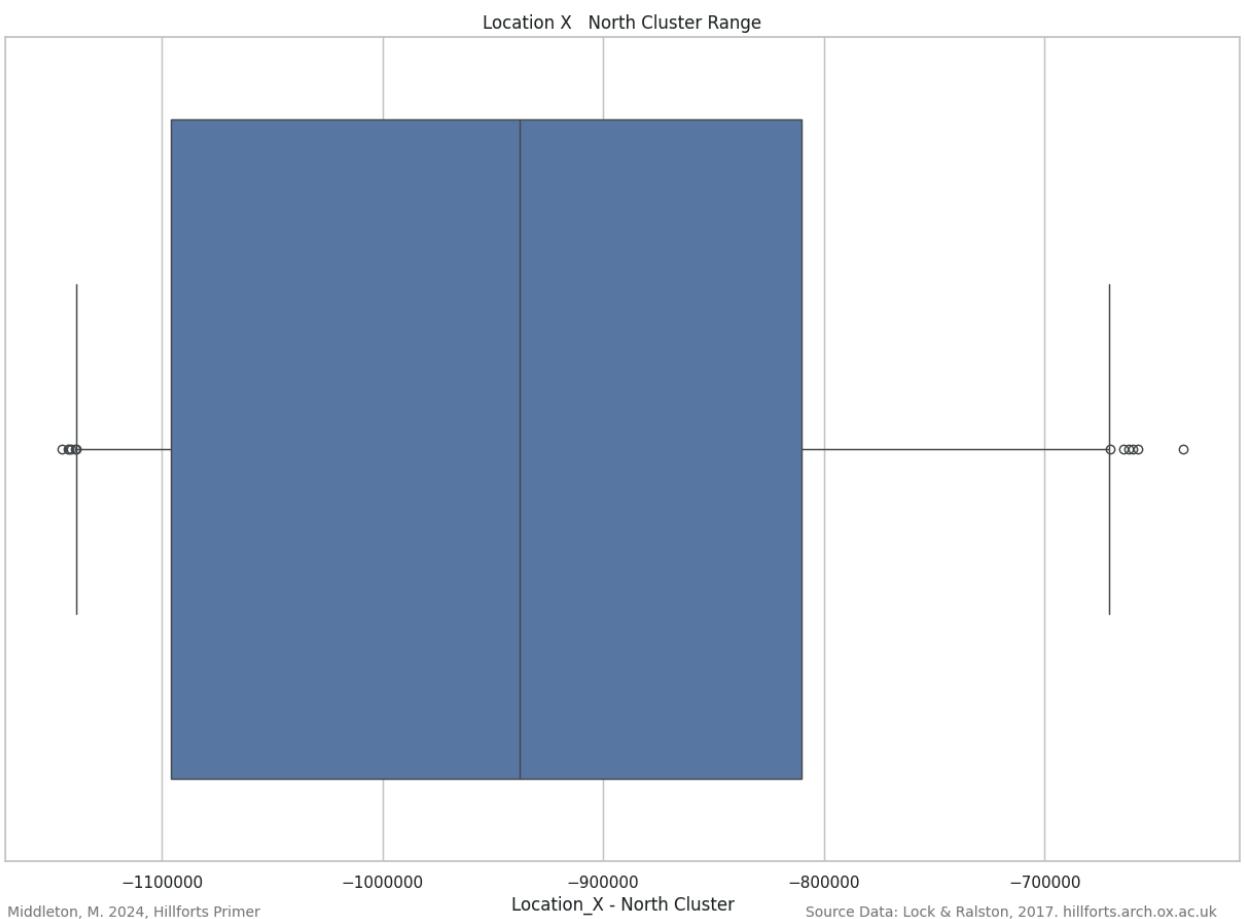
The distribution of forts in North Ireland is very similar to that seen in [Ireland Location Data Plotted](#).

```
In [ ]: plot_histogram(north_cluster['Location_X'], 'Location_X', \
                      'Location_X - North Cluster', 10000)
```



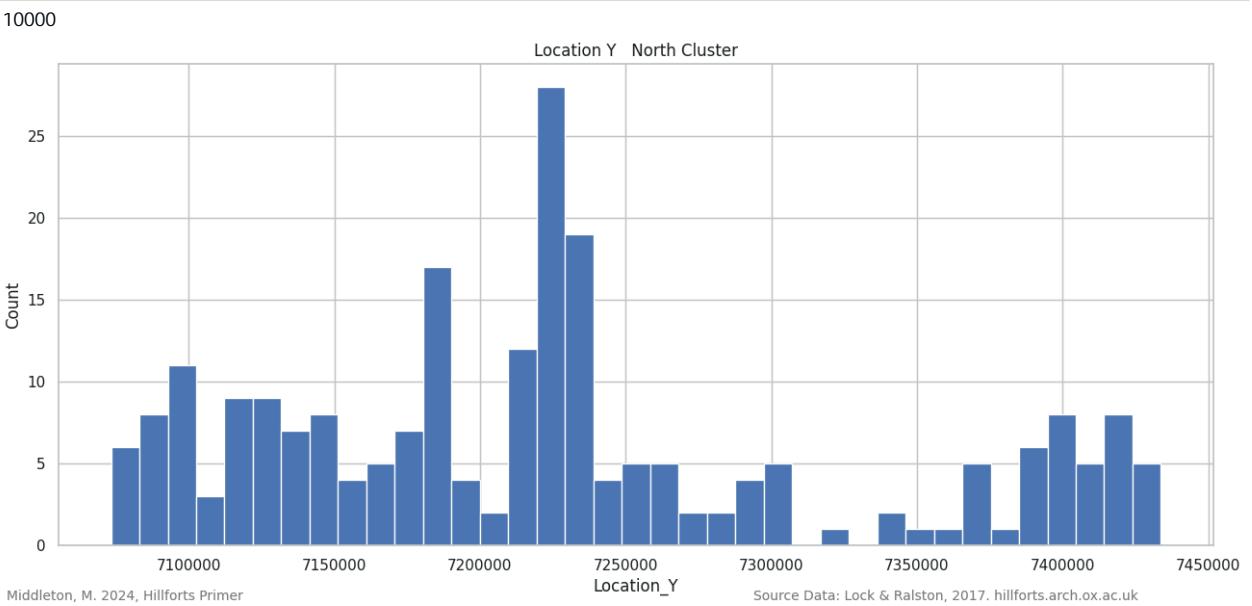
The interquartile range is offset toward the west and it is very broad. This suggests the forts are evenly spread over a wide range and the peaks are weak.

```
In [ ]: location_X_north_data_ireland = plot_data_range(north_cluster['Location_X'], \
                                                       'Location_X - North Cluster', "h")
```



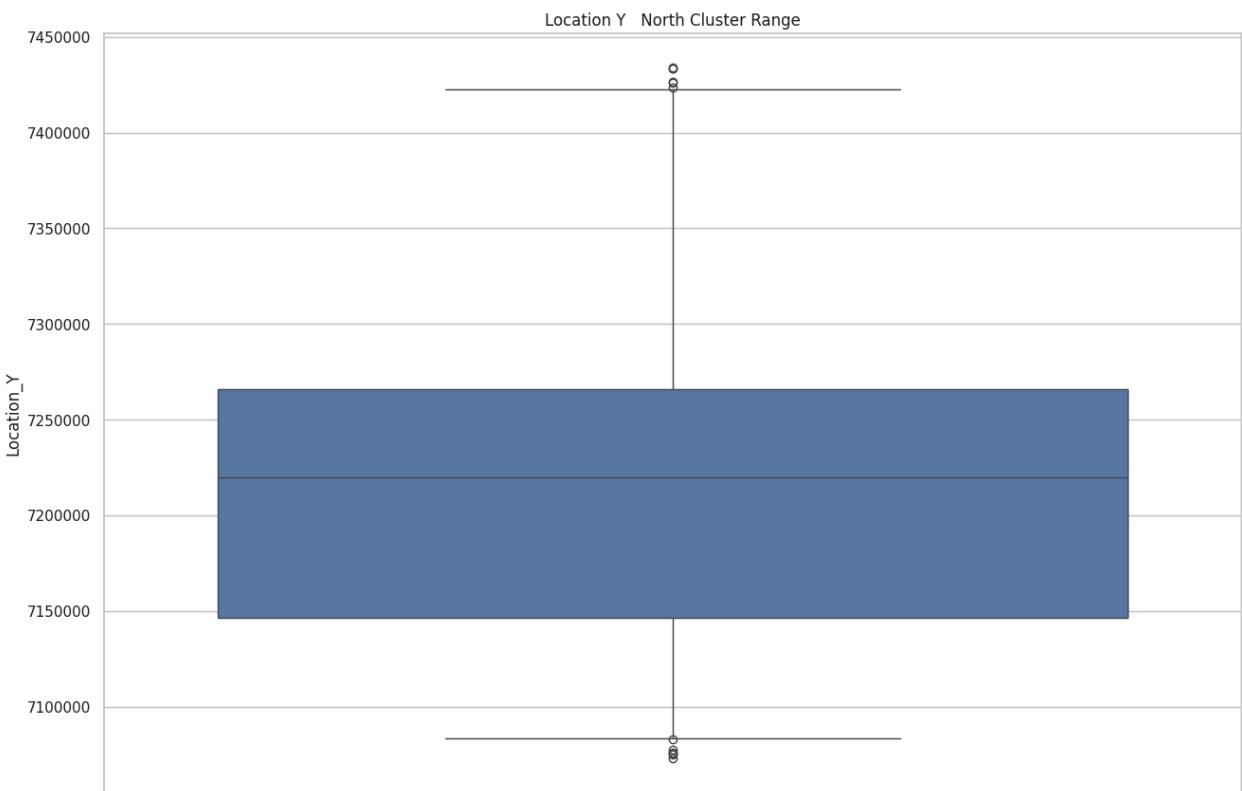
In the 'Location_Y' distribution, the main peak sits around 7,225,000. The distribution of forts to the north is very low while the distribution to the south is low but, gradually increasing toward the south.

```
In [ ]: plot_histogram(north_cluster['Location_Y'], 'Location_Y', \
                     'Location_Y - North Cluster', 10000)
```



The interquartile range is relatively narrow indicating the main peak in the 'Location_Y' distribution is quite strong.

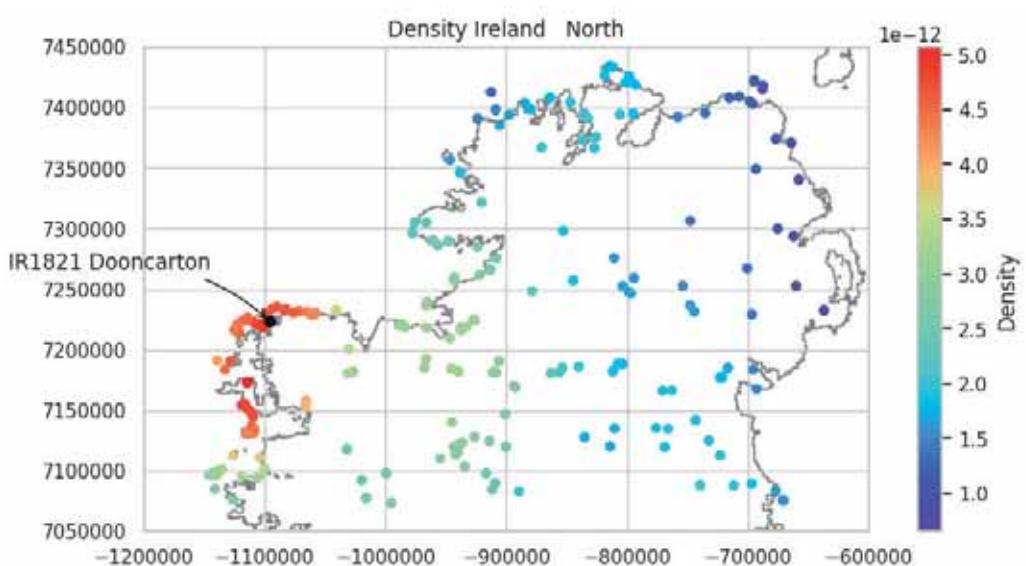
```
In [ ]: location_Y_north_data_iqr_and = \
plot_data_range(north_cluster['Location_Y'], 'Location_Y - North Cluster')
```



Nothern Cluster Location Data Mapped (Ireland)

The main cluster in the northern data is along the Mullet Peninsula around IR1821 Dooncarton and down along the Duvillaun, Achill and Inishkea islands.

```
In [ ]: plot_density_north_ireland(show_dooncarton, north_cluster)
```



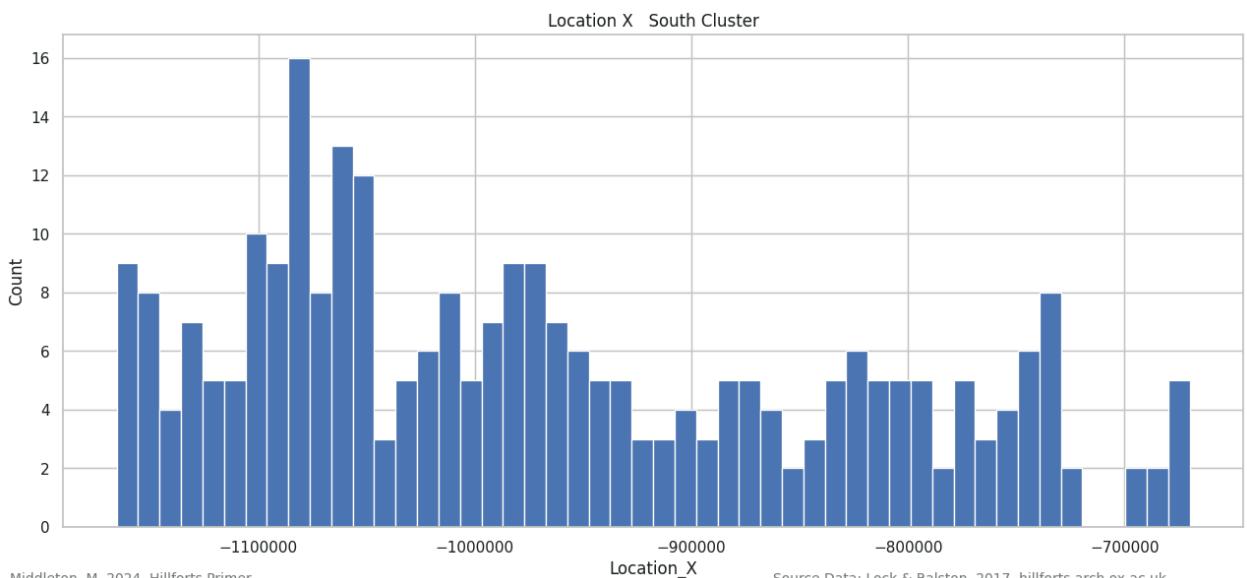
South Ireland Density

Southern Location Data Plotted (Ireland)

The distribution of forts along the 'Location_X' axis in South Ireland is a little more dense than is seen the northern data. The main peak is located around -1,080,000.

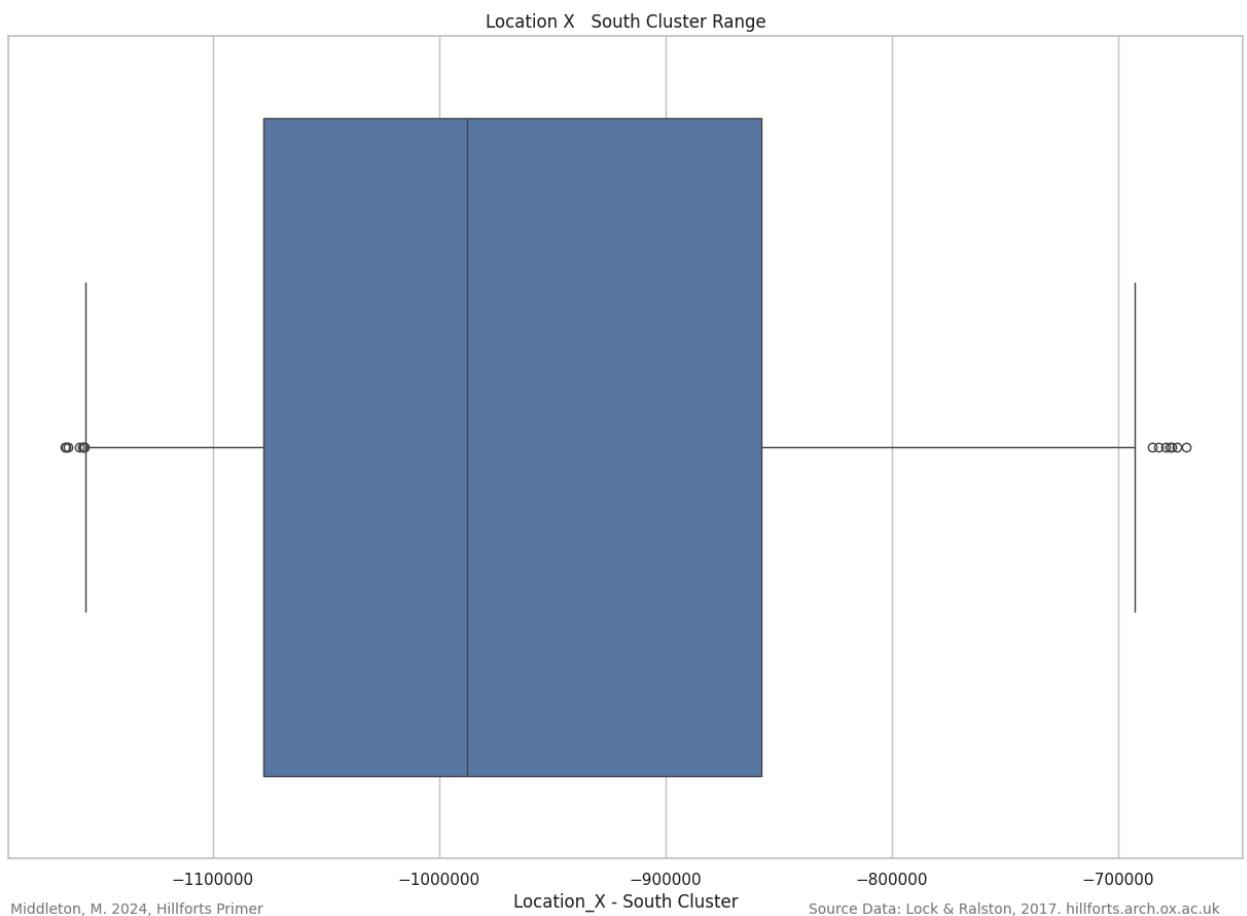
```
In [ ]: plot_histogram(south_cluster['Location_X'], 'Location_X', \
    'Location_X - South Cluster', 10000)
```

10000



The interquartile range is, like the north, offset toward the west and broad. The forts in the south are evenly spread across a wide range, the 'Location_X' axis and the peaks are weak.

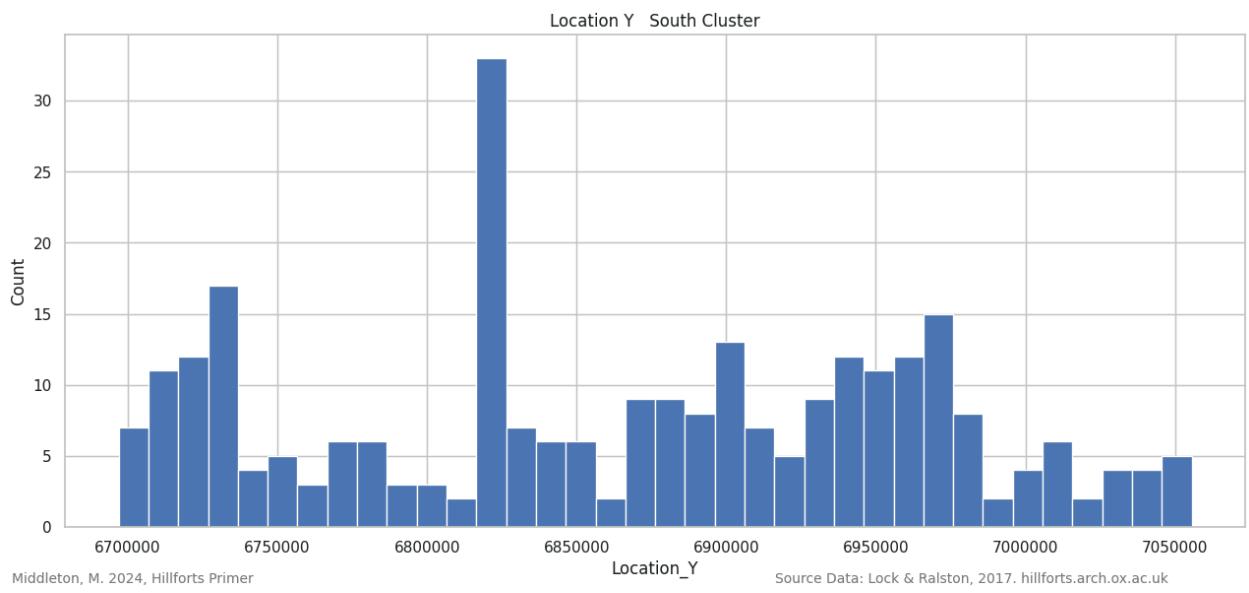
```
In [ ]: location_X_south_data_i_reland = \
plot_data_range(south_cluster['Location_X'], 'Location_X - South Cluster', "h")
```



As mentioned in [Ireland Location Data Plotted](#), the main peak is created by a coincidental topographic alignment. Otherwise, the general distribution, and the height of the peaks, are low.

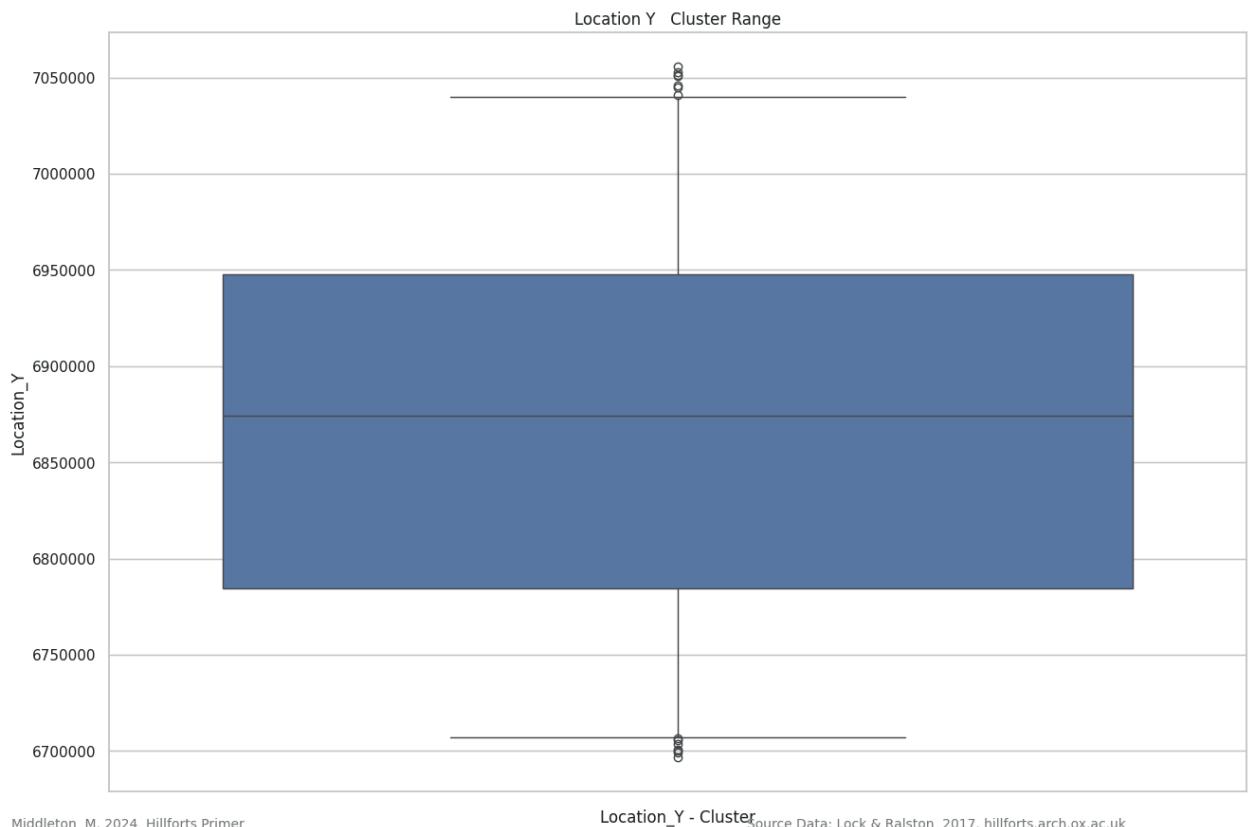
```
In [ ]: plot_histogram(south_cluster['Location_Y'], 'Location_Y', \
'Location_Y - South Cluster', 10000)
```

10000



The interquartile range in the 'Location_Y' data is broad, reflecting the even distribution seen in the histogram above.

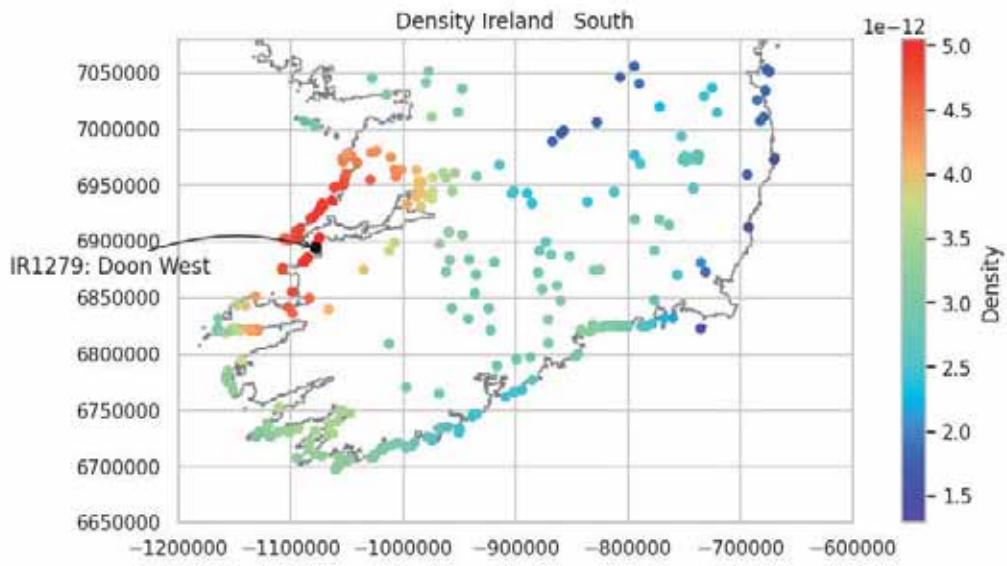
```
In [ ]: location_Y_south_data ireland = \
plot_data_range(south_cluster['Location_Y'], 'Location_Y - Cluster')
```



Southern Cluster Location Data Mapped (Ireland)

The cluster in the southern data is spread along the Galway Bay coast on either side of the Shannon Estuary.

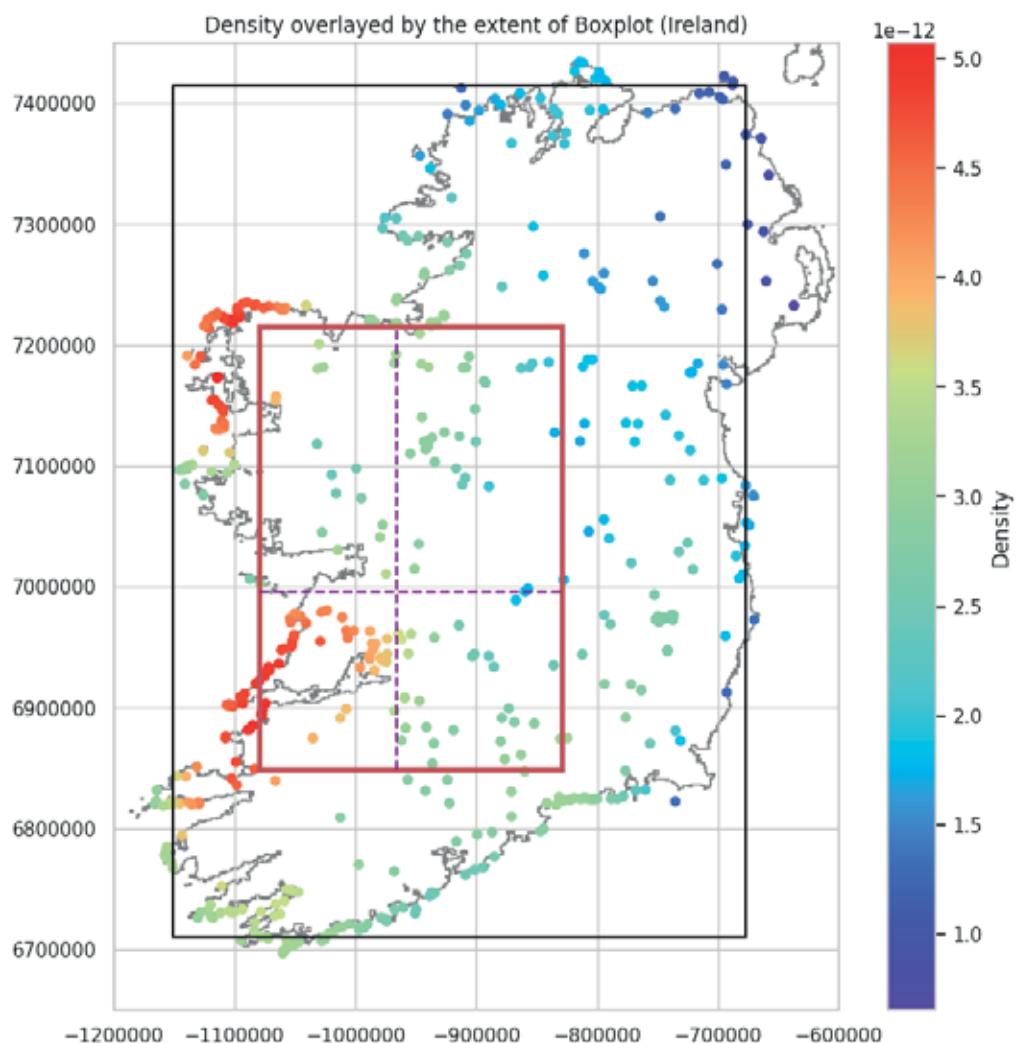
```
In [ ]: plot_density_south_ireland(show_dooncarton, south_cluster)
```



Ireland Boxplots

Density Map showing Extent of Boxplots (Ireland)

```
In [ ]: plot_ireland_one_boxplot(ireland_density_data, location_X_ireland_data, \
location_Y_ireland_data)
```

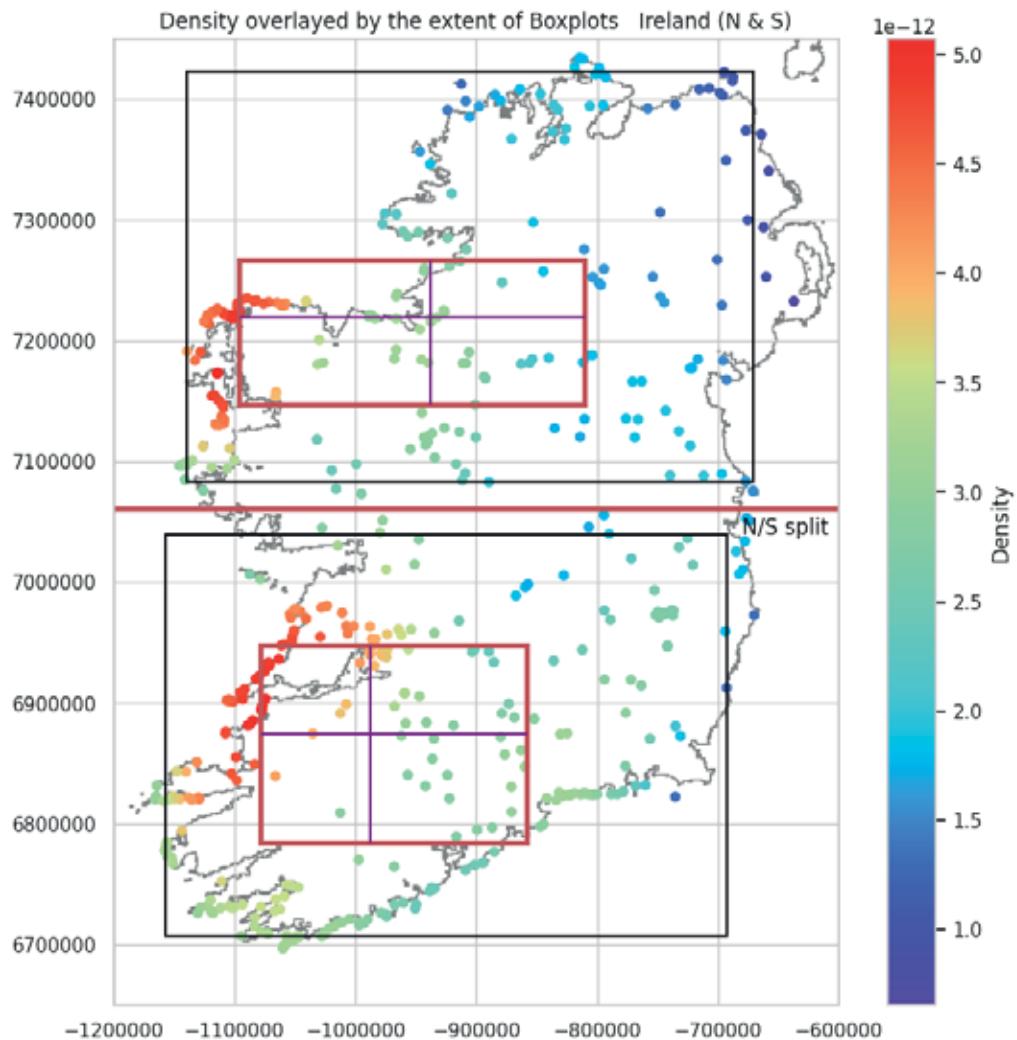


A boxplot over Ireland is stretched between two clusters in the 'Location_Y' axis. In the 'Location_X' axis the boxplot is broad because of the relatively even distribution of sites over most of the island.

Density Map showing Extent of Boxplots (N & S Ireland)

The northern and southern boxplots for Ireland align well along the y-axis for both clusters. By contrast, both boxplots are elongated along the x-axis. The two data packages are very small and small variations in the data may have a large influence over the outputs. Due to the small sample size and the uniform distributions noted above, no attempt will be made to identify secondary clusters within these areas.

```
In [ ]: plot_irland_boxplots(irland_density_data, \
    location_X_south_data_irland, \
    location_Y_south_data_irland, \
    location_X_north_data_irland, \
    location_Y_north_data_irland)
```



Middleton, M. 2024, Hillforts Primer

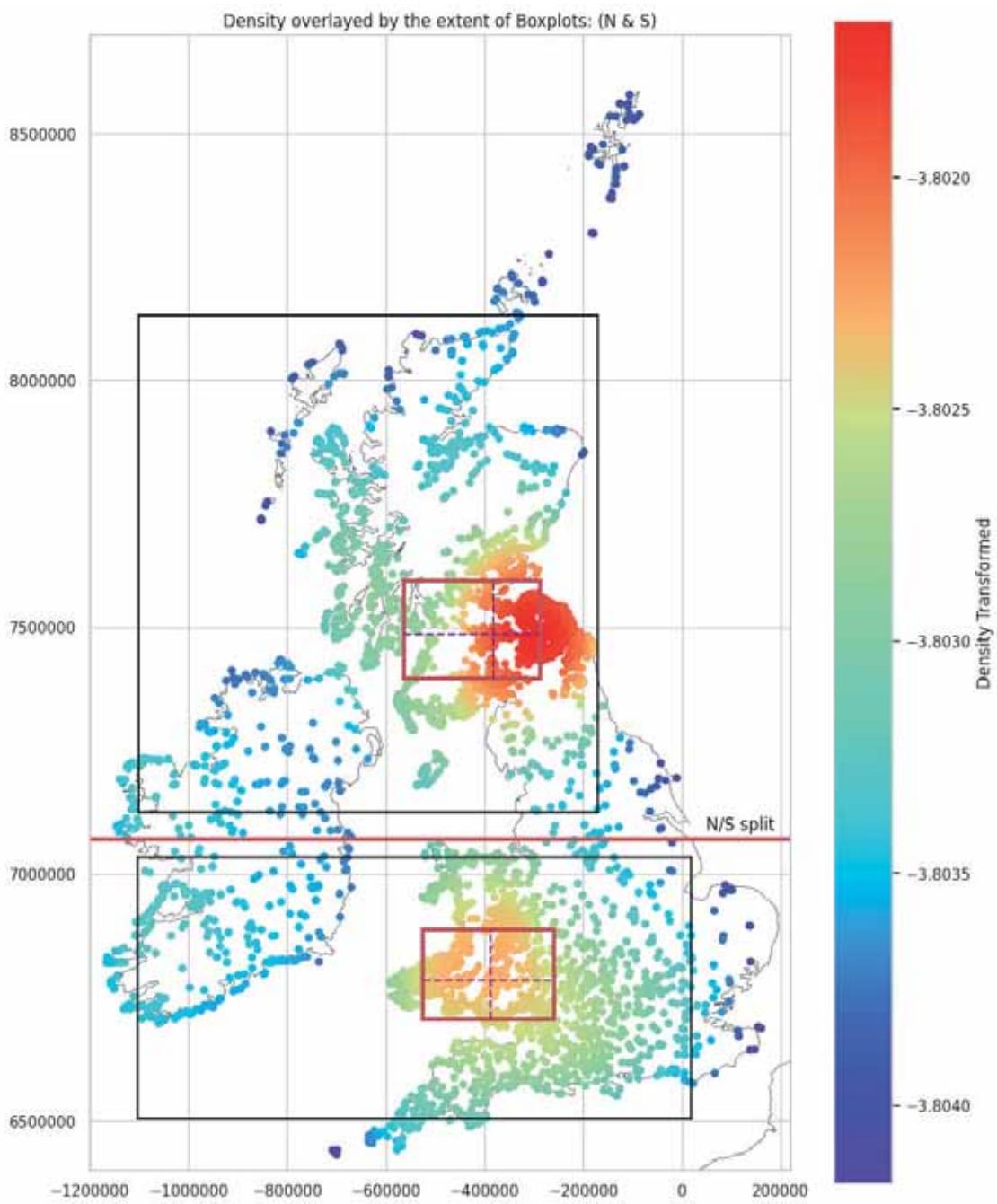
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Density Boxplots (All Data)

Density Map showing Extent of Boxplots (N & S)

To summarise, the Hillforts Atlas data was first split between the two most intense clusters in the north and south.

```
In [ ]: plot_ns_boxplots(transformed_location_numeric_data_short, \
    location_X_south_data, location_Y_south_data, \
    location_X_north_data, location_Y_north_data)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Density Map showing Extent of Boxplots \(North\)](#)

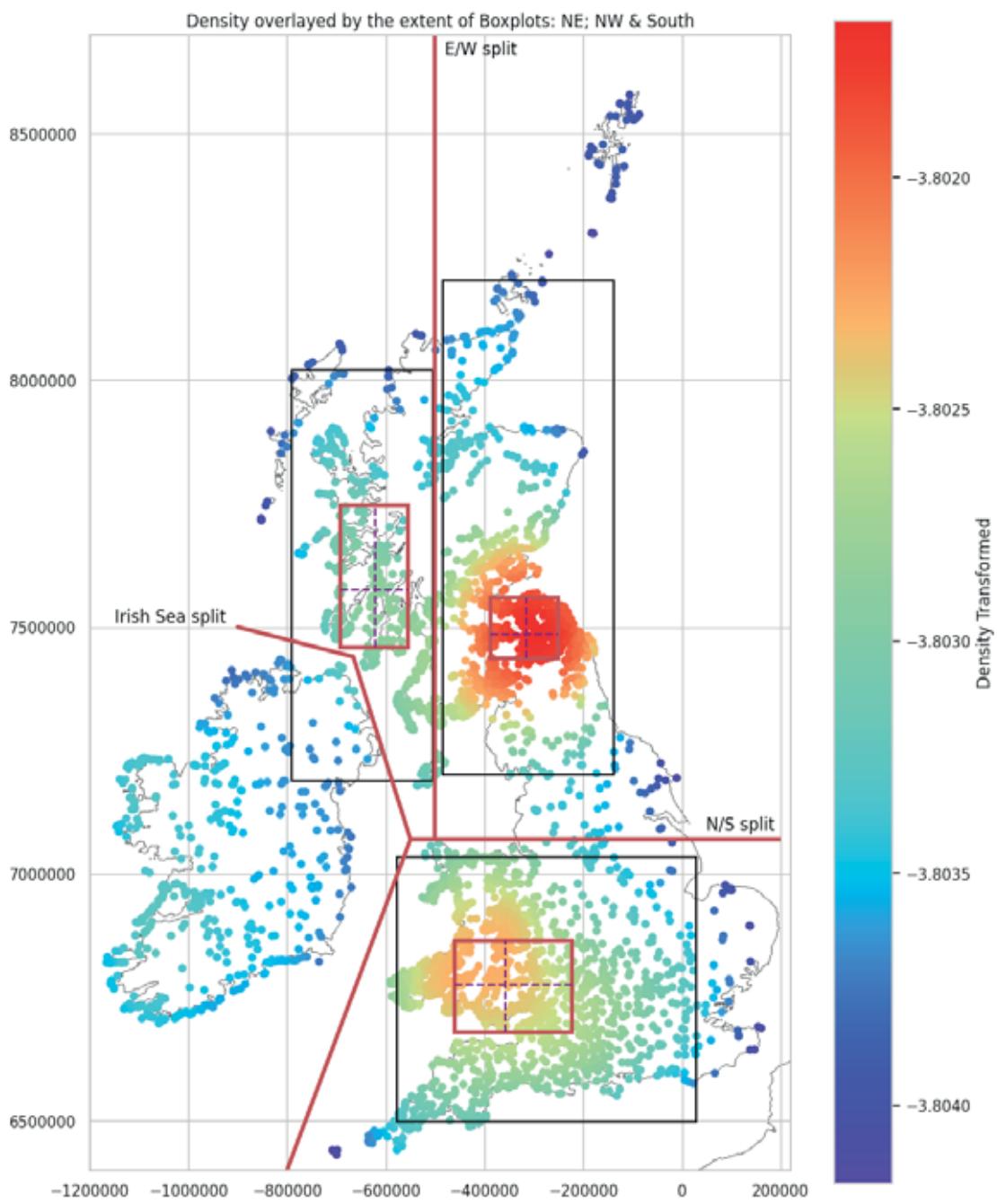
see: [Density Map showing Extent of Boxplots \(South\)](#)

Density Map showing Extent of Boxplots Northeast, Northwest and South

The northern boxplot in [Density Map showing Extent of Boxplots \(N & S\)](#) was stretched to the west indicating secondary clusters.

The northern data was then split to reveal a cluster focussed over the west coast.

```
In [ ]: plot_ne_nw_s_boxplots(transformed_location_numeric_data_short, \
location_X_cluster_north_west, \
location_Y_cluster_north_west, \
location_X_north_e_data, location_Y_north_e_data, \
location_X_cluster_south, location_Y_cluster_south)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Density Map showing Extent of Boxplots \(South\)](#)

See: [Northwestern Data Minus Ireland Mapped](#)

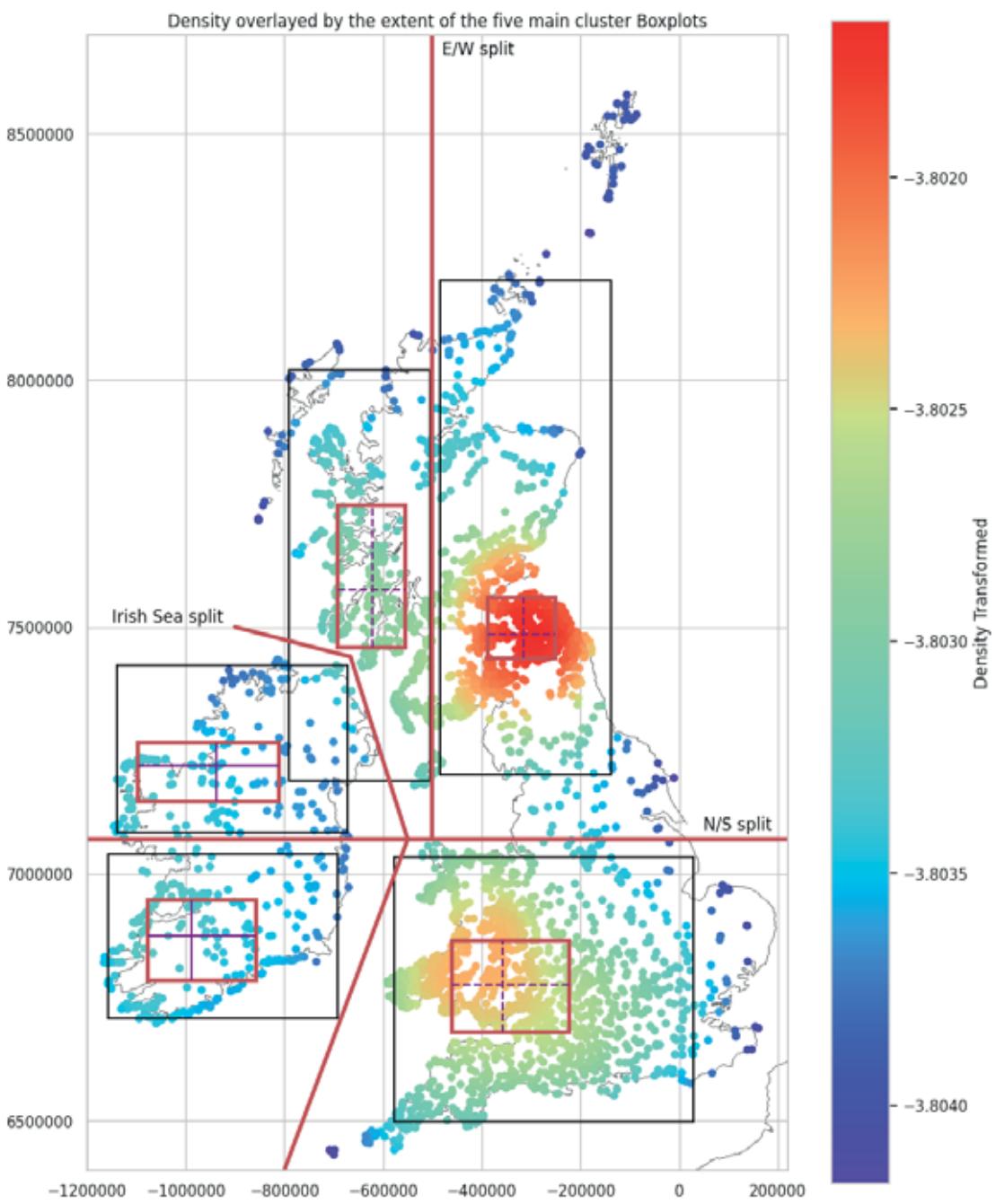
See: [Northeast Data Mapped](#)

See: [Density Map showing Extent of Boxplots identified in the Atlas Data](#)

Density Map showing Extent of Boxplots identified in the Atlas Data

Finally, the Irish data was split to reveal two clusters. Overall, five main clusters were identified.

```
In [ ]: plot_five_clusters(transformed_location_numeric_data_short, \
    location_X_cluster_north_west, \
    location_Y_cluster_north_west, \
    location_X_north_e_data, \
    location_Y_north_e_data, location_X_cluster_south, \
    location_Y_cluster_south, location_X_south_data_irland, \
    location_Y_south_data_irland, location_X_north_data_irland, \
    location_Y_north_data_irland)
```



Middleton, M. 2024, Hillforts Primer

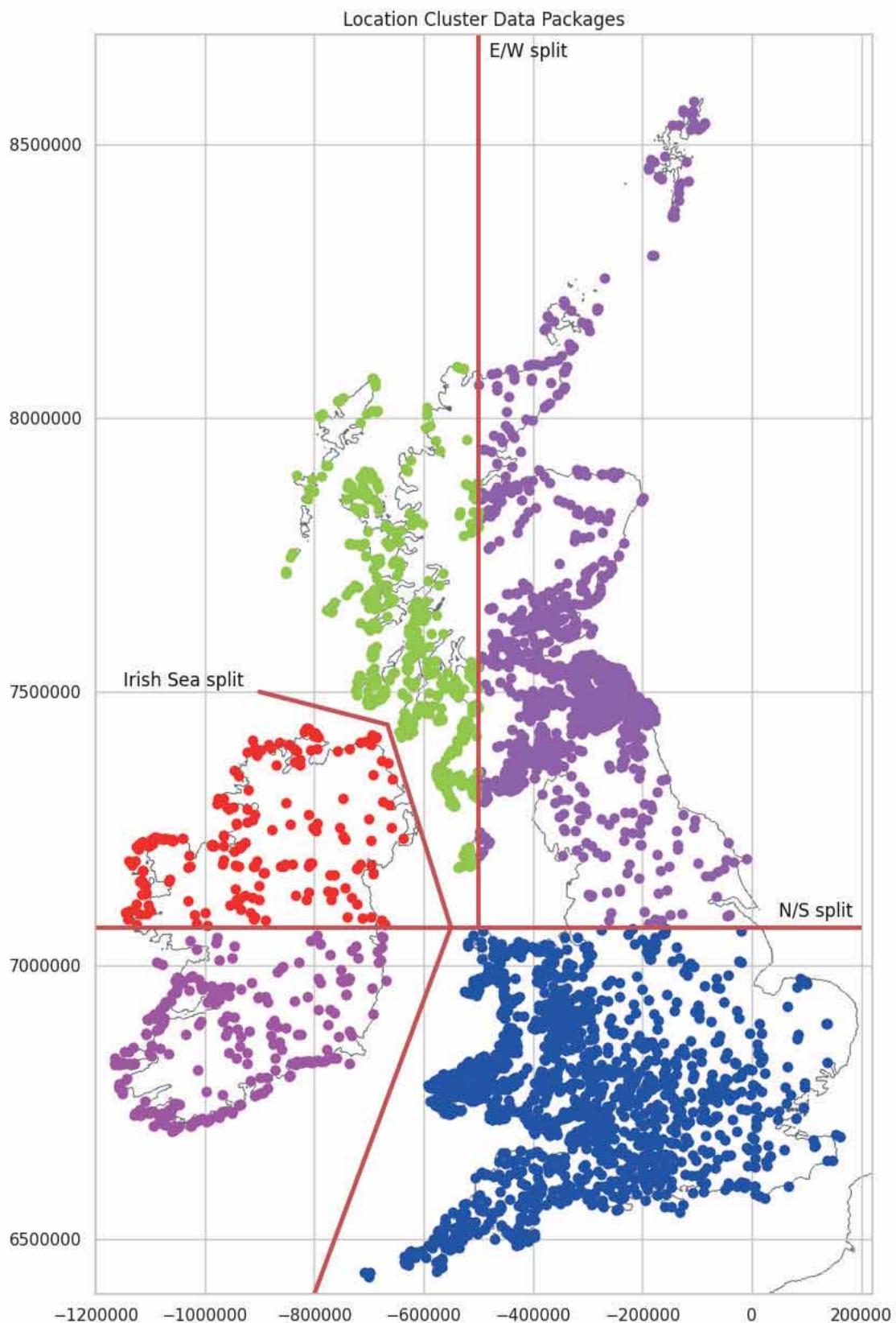
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: Southern Cluster Location Data Mapped (Ireland)
 See: Northern Cluster Location Data Mapped (Ireland)
 See: Density Map showing Extent of Boxplots (South)
 See: Northwestern Data Minus Ireland Mapped
 See: Northeast Data Mapped
 See: Density Map Showing Clusters Adjusted by Region

Cluster Data Packages Mapped

The data packages for each region were then exported so that the density and boxplots for each could be processed independently.

```
In [ ]: plot_clusters(cluster_north_ireland, cluster_south_ireland, \
cluster_south, cluster_north_east, cluster_north_west)
```



Middleton, M. 2024, Hillforts Primer

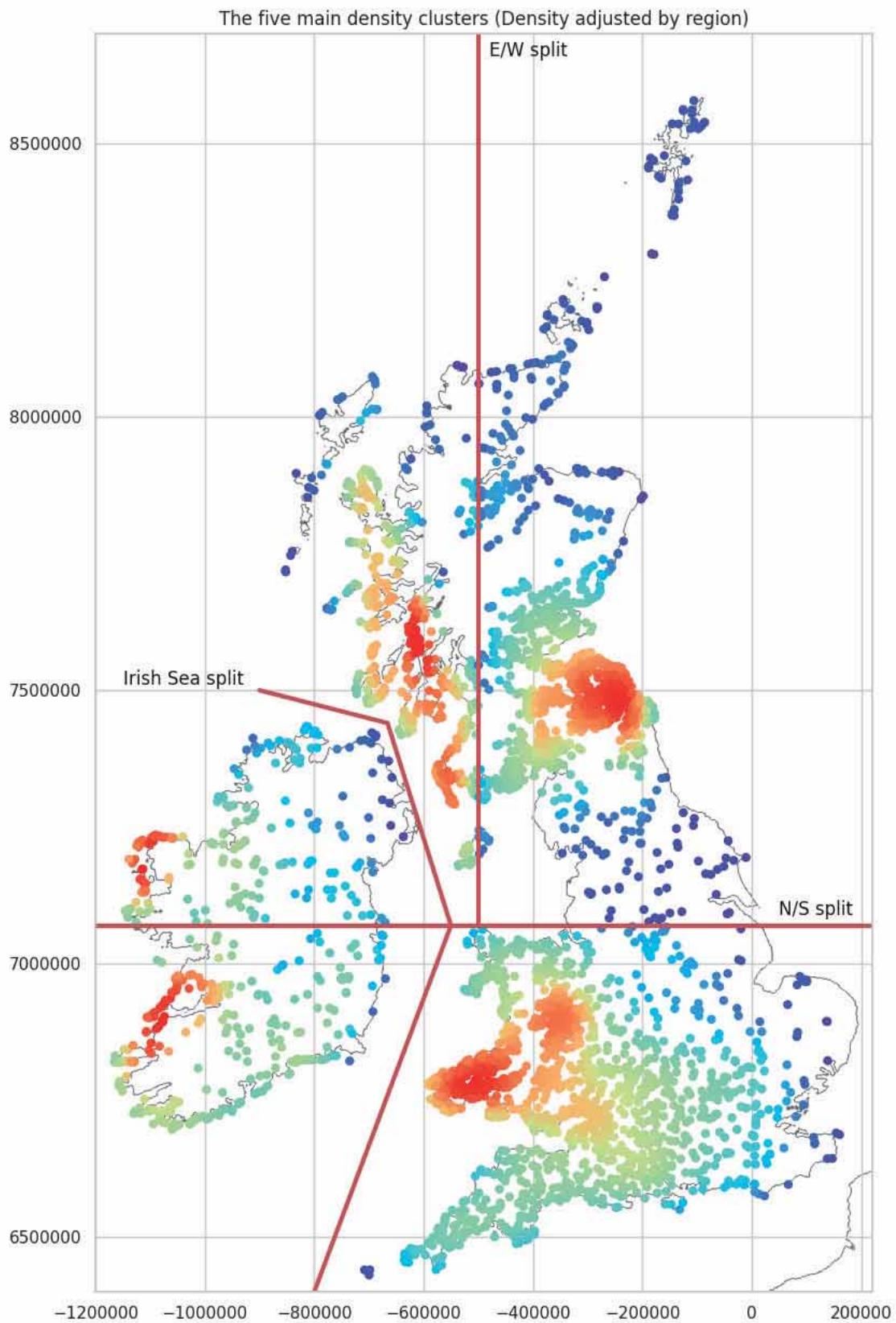
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Density Map Showing Clusters Adjusted by Region

This final density map shows the five main clusters identified based only on the data for each region. See [Density Map showing Extent of Boxplots \(South\)](#) for a discussion on the potential for the southern cluster to be further subdivided.

```
In [ ]: show_boxplots = False
```

```
In [ ]: plot_adjusted_density(i_rel_and_density_data, cluster_south, \
cluster_north_west, north_east, \
location_X_cluster_south, location_Y_cluster_south, \
location_X_cluster_north_west, location_Y_cluster_north_west, \
location_X_north_e_data, location_Y_north_e_data, \
location_X_south_data_i_rel_and, location_Y_south_data_i_rel_and, \
location_X_north_data_i_rel_and, location_Y_north_data_i_rel_and)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: [Southern Cluster Location Data Mapped \(Ireland\)](#)

See: [Northern Cluster Location Data Mapped \(Ireland\)](#)

See: Density Map showing Extent of Boxplots (South)
See: Northwestern Data Minus Ireland Mapped
See: Northeast Data Mapped
See: Density Map showing Extent of Boxplots identified in the Atlas Data

Location Data Packages

Pre-processed location data.

```
In [ ]: location_data_list = \
[transformed_location_numeric_data_short, \
location_text_data, location_encodeable_data_short]
download_location_data_list = \
[location_numeric_data, \
transformed_location_numeric_data_short[['Density', 'Density_trans']].copy(), \
location_text_data, location_encodeable_data_short]
```

Download Location Data

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:
<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(download_location_data_list, 'Location_Long')
download(location_data_list, 'Location_package')
```

Status Data

The status data contains information on the citizen science and the reliability of the data and interpretation.

```
In [ ]: status_features = [
    'Status_Citizen_Science',
    'Status_Citizen',
    'Status_Data_Reliability',
    'Status_Data_Comments',
    'Status_Interpretation_Reliability',
    'Status_Interpretation_Comments']

status_data = hillforts_data[status_features]
status_data.head()
```

```
Out[ ]:   Status_Citizen_Science  Status_Citizen  Status_Data_Reliability  Status_Data_Comments  Status_Interpretation_Reliability  Status_Interpretation_Comments
0           No          NaN        Confirmed          NaN            Confirmed
1           No          NaN        Confirmed          NaN            Confirmed
2           No          NaN        Confirmed          NaN            Confirmed
3           No          NaN        Confirmed          NaN            Confirmed
4           No          NaN        Confirmed          NaN            Confirmed
```

```
In [ ]: status_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Status_Citizen_Science  4147 non-null  object 
 1   Status_Citizen        246 non-null   object 
 2   Status_Data_Reliability 4147 non-null  object 
 3   Status_Data_Comments  333 non-null   object 
 4   Status_Interpretation_Reliability 4147 non-null  object 
 5   Status_Interpretation_Comments 1152 non-null  object 
dtypes: object(6)
memory usage: 194.5+ KB
```

Status Numeric Data

The Status data package contains no numeric data features.

```
In [ ]: status_numeric_data = pd.DataFrame()
```

Status Text Data

There are three free text Status features.

```
In [ ]: status_text_features = [
    'Status_Citizen',
    'Status_Data_Comments',
    'Status_Interpretation_Comments']

status_text_data = hilberts_data[status_text_features]
status_text_data.head(20)
```

Out[]:	Status_Citizen	Status_Data_Comments	Status_Interpretation_Comments
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN
5	NaN	The site is a cropmark and there are many irre...	The site is a cropmark and there are many irre...
6	NaN	NaN	NaN
7	NaN	NaN	NaN
8	NaN	NaN	NaN
9	NaN	NaN	NaN
10	NaN	NaN	NaN
11	NaN	NaN	NaN
12	NaN	NaN	NaN
13	NaN	Little is known about the site and waits the r...	At present the lack of data prohibits final in...
14	NaN	NaN	Issues remain as to its interpretation
15	NaN	NaN	NaN
16	NaN	NaN	NaN
17	NaN	NaN	NaN
18	NaN	NaN	NaN
19	Alistair Hodcroft (Glos Arch)	NaN	NaN

Status Text Data - Resolve Null Values

Test for the presence of 'NA' in the Management text features.

```
In [ ]: test_cat_list_for_NA(status_text_data, status_text_features)

Status_Citizen 0
Status_Data_Comments 0
Status_Interpretation_Comments 0
```

As 'NA' is not present, null values are filled with 'NA'.

```
In [ ]: status_text_data = update_cat_list_for_NA(status_text_data,
                                                status_text_features)
status_text_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Status_Citizen   4147 non-null   object 
 1   Status_Data_Comments 4147 non-null   object 
 2   Status_Interpretation_Comments 4147 non-null   object 
dtypes: object(3)
memory usage: 97.3+ KB
```

Status Encodable Data

There are three Status features that have the potential to be encoded.

```
In [ ]: status_encodeable_features = [
    'Status_Citizen_Science',
    'Status_Data_Reliability',
    'Status_Interpretation_Reliability']

status_encodeable_data = hillforts_data[status_encodeable_features]
status_encodeable_data.head()
```

```
Out[ ]:
```

	Status_Citizen_Science	Status_Data_Reliability	Status_Interpretation_Reliability
0	No	Confirmed	Confirmed
1	No	Confirmed	Confirmed
2	No	Confirmed	Confirmed
3	No	Confirmed	Confirmed
4	No	Confirmed	Confirmed

Status Data Mapped

The Status data is recombined with the 'Location' data so it can be mapped.

```
In [ ]: location_status_data = pd.merge(location_numeric_data_short,
                                         status_encodeable_data,
                                         left_index=True,
                                         right_index=True)
```

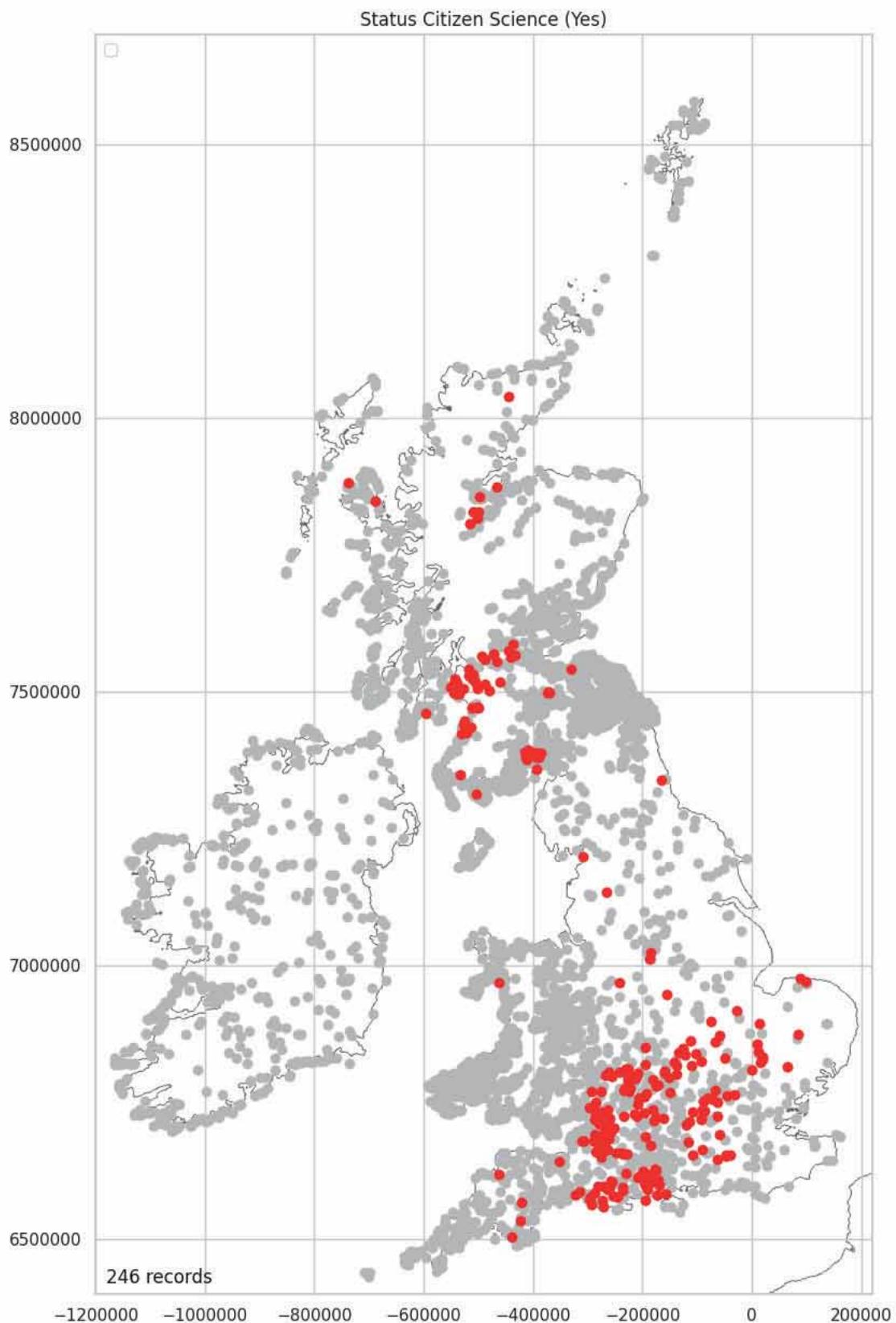
Citizen Science

Records citizen science activity.

```
In [ ]: status_data['Status_Citizen_Science'].value_counts()

Out[ ]: No      3901
        Yes     246
Name: Status_Citizen_Science, dtype: int64

In [ ]: status_citizen_science = plot_over_grey(location_status_data,
                                                'Status_Citizen_Science', 'Yes', '(Yes)')
```

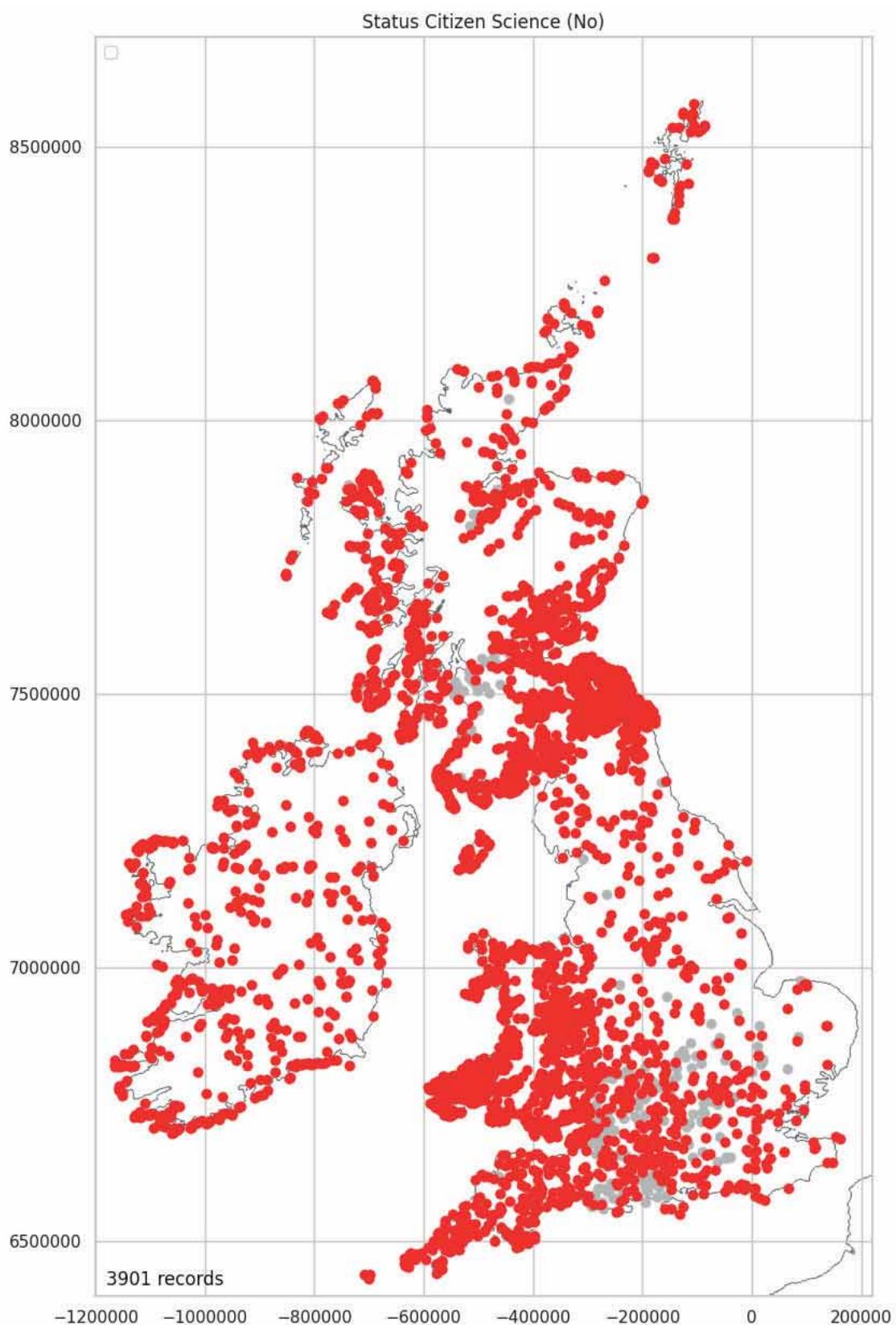


Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5. 93%

```
In [ ]: status_citizen_science_no = plot_over_grey(location_status_data,
    'Status_Citizen_Science', 'No', '(No)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

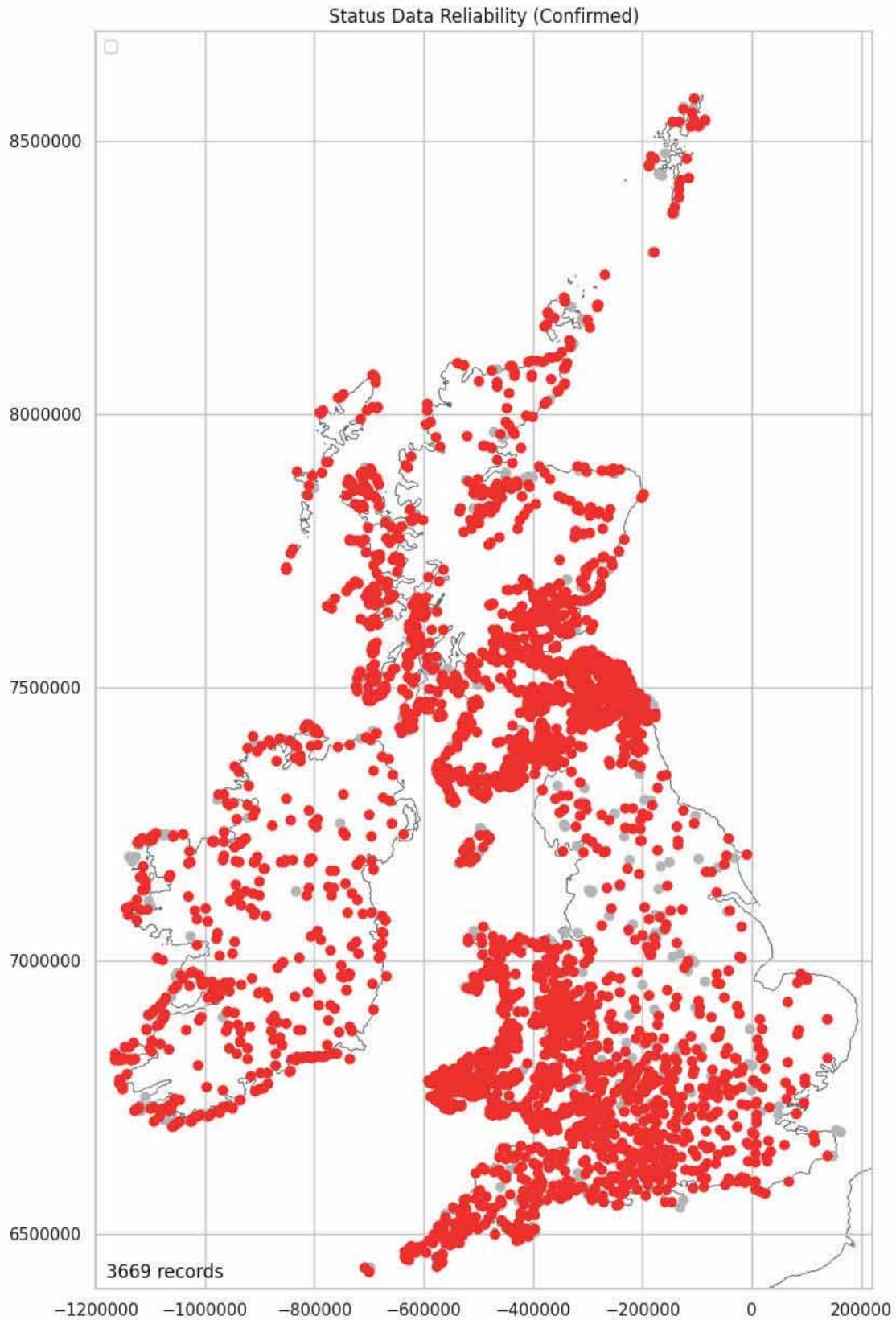
94.07%

Data Reliability

```
In [ ]: status_data['Status_Data_Reliability'].value_counts()
```

```
Out[ ]: Confirmed      3669
Unconfirmed      436
Irreconciled issues    42
Name: Status_Data_Reliability, dtype: int64
```

```
In [ ]: status_data_reliability_confirmed = \
plot_over_grey(location_status_data, \
'Status_Data_Reliability', 'Confirmed', '(Confirmed)')
```



Middleton, M. 2024, Hillforts Primer

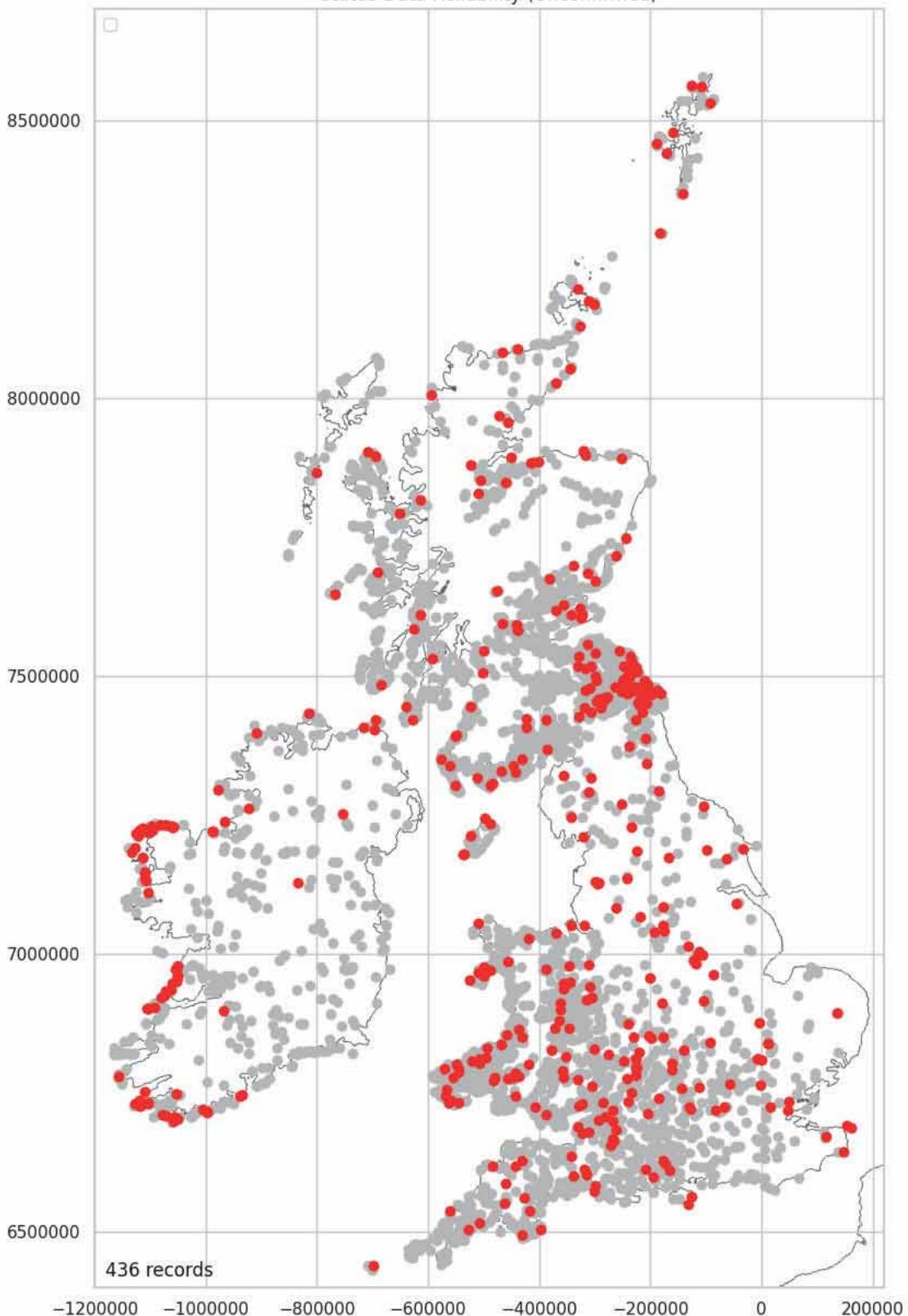
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

88.47%

Data Reliability - Unconfirmed

```
In [ ]: status_data_reliability_unconfirmed = \
plot_over_grey(location_status_data, \
'Status_Data_Reliability', 'Unconfirmed', '(Unconfirmed)')
```

Status Data Reliability (Unconfirmed)



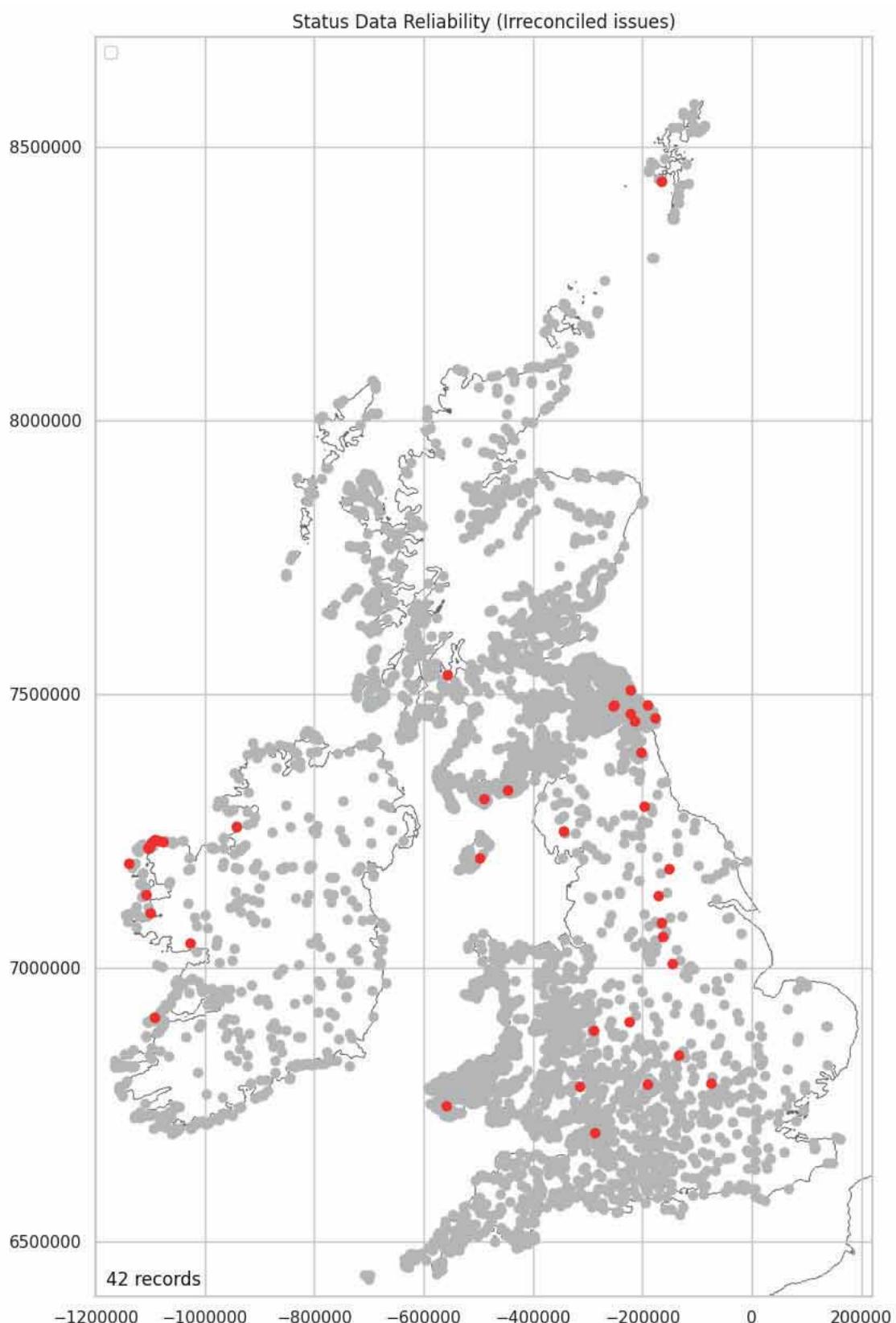
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

10.51%

Data Reliability - Irreconciled issues

```
In [ ]: status_data_reliability_irreconciled = \
plot_over_grey(location_status_data, \
    'Status_Data_Reliability', 'Irreconciled_issues', \
    '(Irreconciled_issues)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

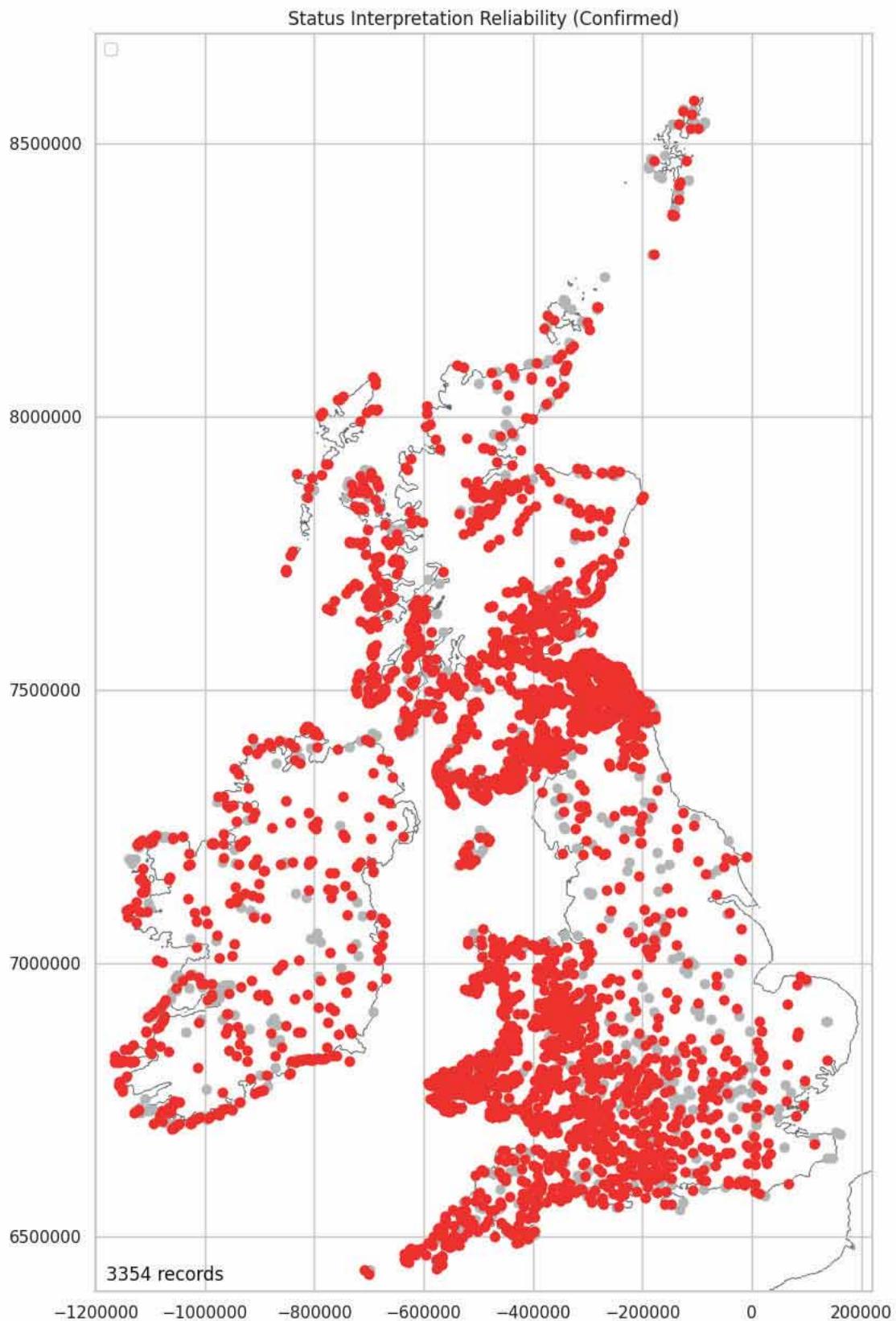
1.01%

Interpretation Reliability

```
In [ ]: status_data['Status_Interpretation_Reliability'].value_counts()
```

```
Out[ ]: Confirmed      3354
Unconfirmed     722
Irreconciled issues    71
Name: Status_Interpretation_Reliability, dtype: int64
```

```
In [ ]: status_interpretation_reliability_irreconciled = \
plot_over_grey(location_status_data, \
    'Status_Interpretation_Reliability', 'Confirmed', '(Confirmed')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

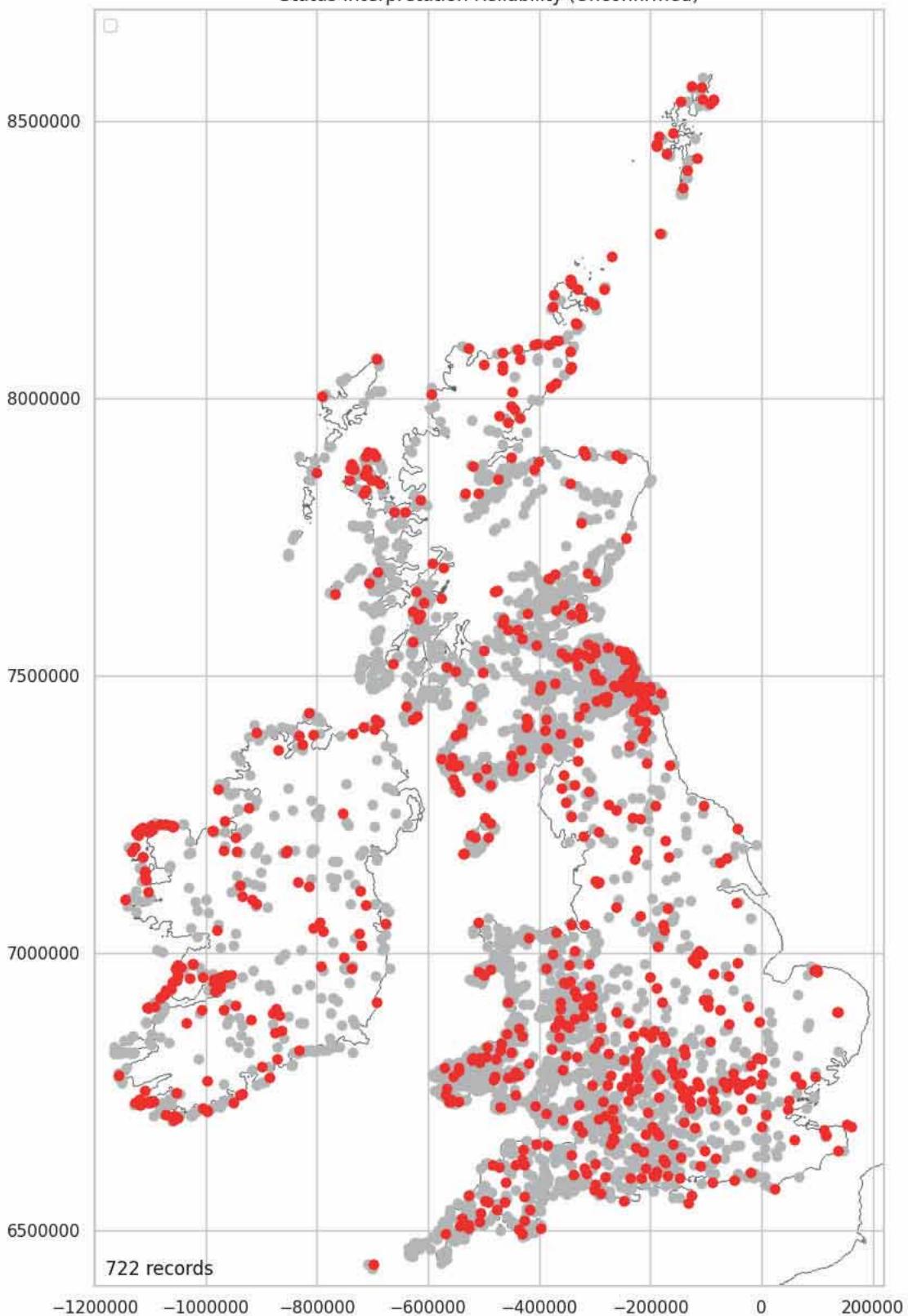
80.88%

Interpretation Reliability - Unconfirmed

```
In [ ]: status_interpretation_reliability_irreconciled = \
plot_over_grey(location_status_data, \
    'Status_Interpretation_Reliability', 'Unconfirmed', '(Unconfirmed')
```

```
'Status_Interpretation_Reliability', 'Unconfirmed', \
'(Unconfirmed')
```

Status Interpretation Reliability (Unconfirmed)



Middleton, M. 2024, Hillforts Primer

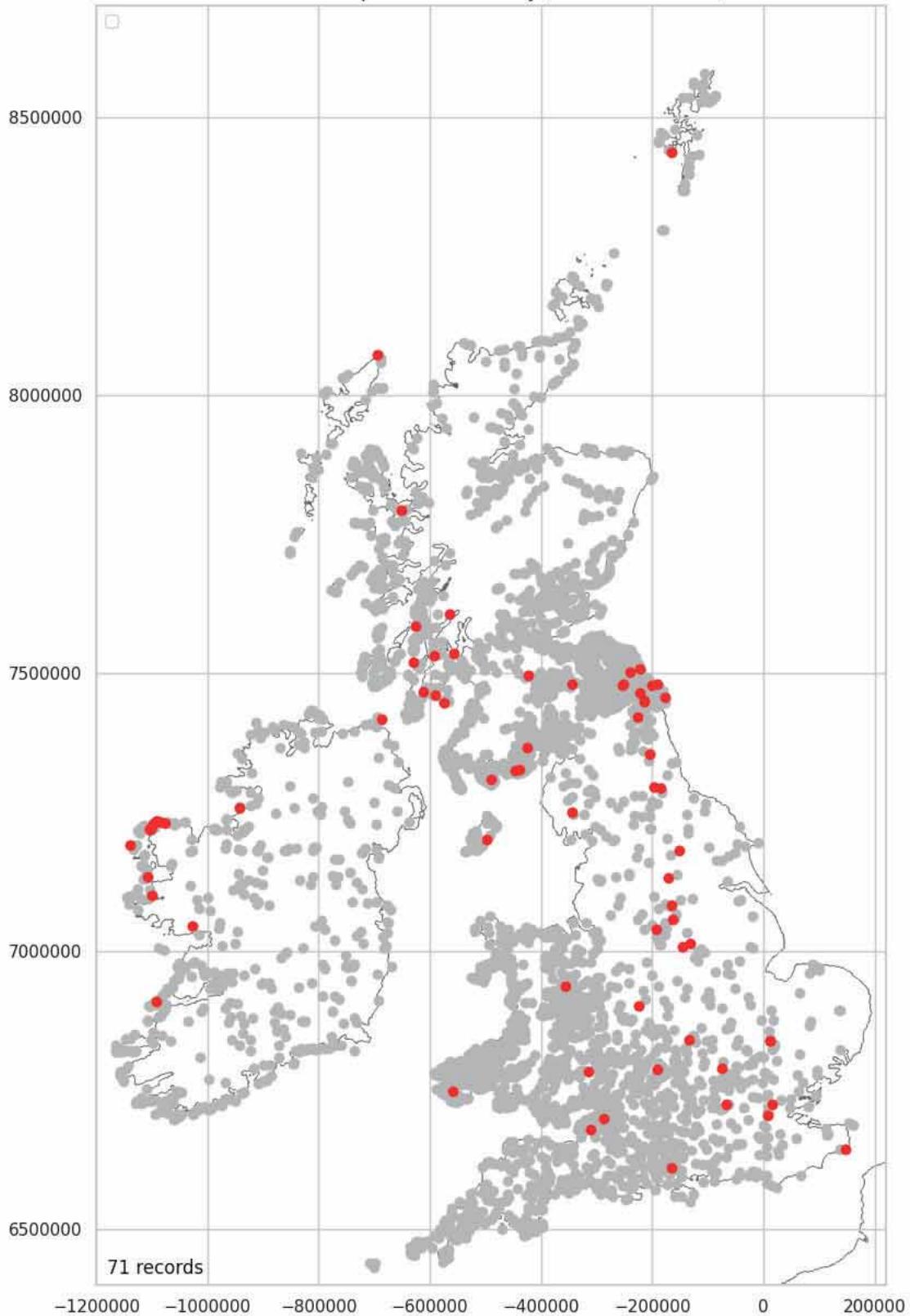
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

17.41%

Interpretation Reliability - Irreconciled issues

```
In [ ]: status_interpretation_reliability_irreconciled = \
plot_over_grey(location_status_data, \
'Status_Interpretation_Reliability', 'Irreconciled_issues', \
'(Irreconciled_issues)')
```

Status Interpretation Reliability (Irreconciled issues)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

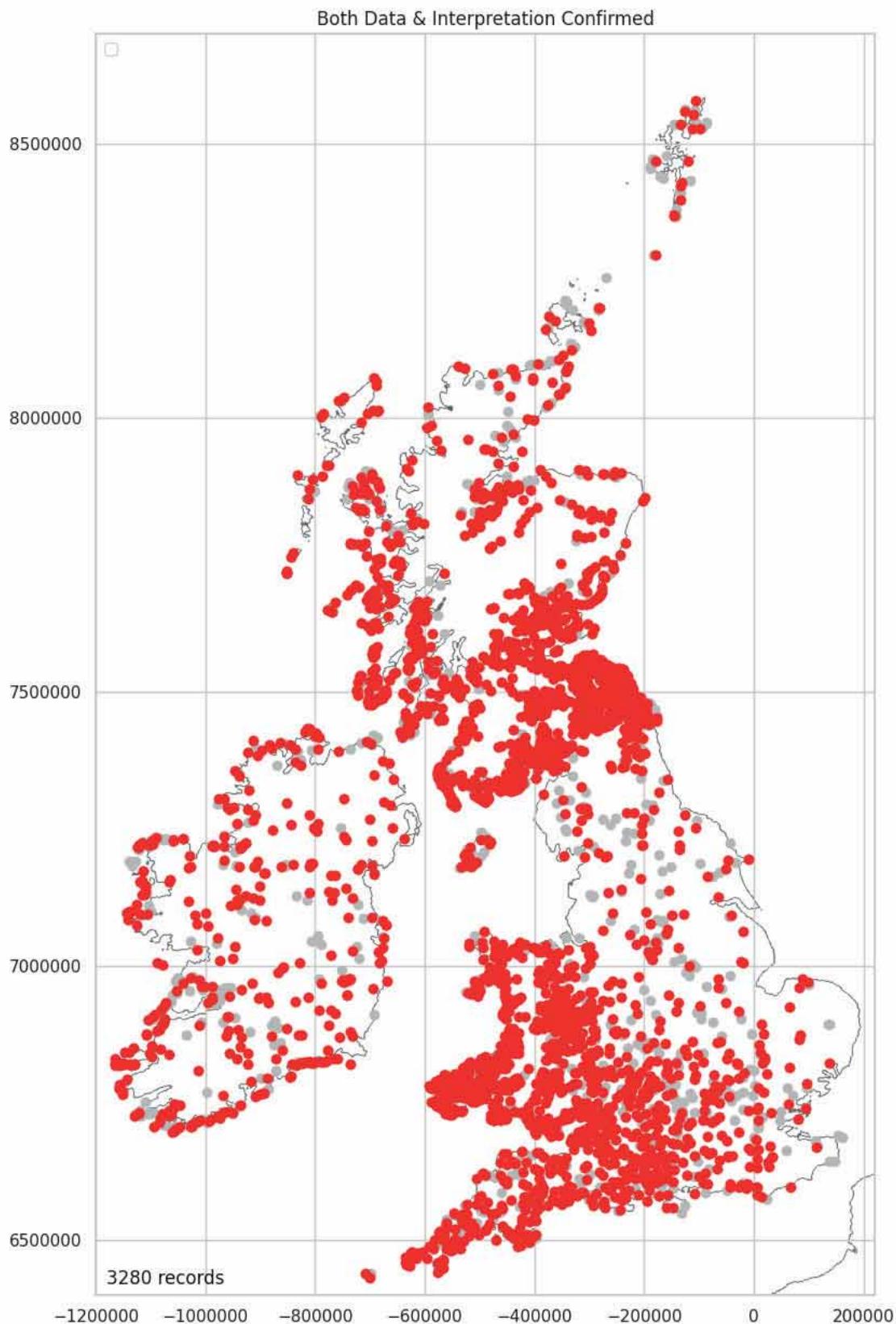
1. 71%

Data and Interpretation Reliability - Confirmed

```
In [ ]: condition = (location_status_data['Status_Interpretation_Reliability'] == \
                  'Confirmed') & (location_status_data['Status_Data_Reliability'] == \
                  'Confirmed')
both_confirmed = location_status_data[condition]
print(f"There are {len(both_confirmed)} forts where the data and interpretation reliability area confirmed.")
```

There are 3280 forts where the data and interpretation reliability area confirmed.

```
In [ ]: plot_over_grey_numeric(both_confirmed, "", "Both Data & Interpretation Confirmed")
```



Status Data Packages

Pre-processed status data.

```
In [ ]: status_data_list = [status_numeric_data, status_text_data, \
                           status_encodeable_data]
```

Download Status Data

The download will only function if selected by the user. See: [User Settings](#).

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(status_data_list, 'Status_package')
```

Save Figure List

```
In [ ]: if save_images:  
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")  
    fig_list.to_csv(path, index=False)
```

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 2

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

- [Management Data](#)
- [Landscape Data](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:
<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [ ]: confirmed_only = False  
In [ ]: download_data = False  
In [ ]: save_images = False  
In [ ]: show_topography = False
```

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Review Data Part 2](#).

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>
Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer
Licence: <https://creativecommons.org/licenses/by-sa/4.0/>
Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>
Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>
Hillforts: Britain, Ireland and the Nearer Continent (Sample):
<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Reload Data and Python Functions

This study is split over multiple documents. Each file needs to be configured and have the source data imported. As this section does not focus on the assessment of the data it is minimised to facilitate the documents readability.

Python Modules and Code Setup

The Python imports enable the Hillforts Atlas data to be analysed and mapped within this document. The Python code can be run on demand, (see: [User Settings](#)). This means that as new research becomes available, the source for this document can be updated to a revised copy of the Atlas data and the impact of that research can be reviewed using the same code and graphic output. The Hillforts Atlas is a baseline and this document is a tool that can be used to assess the impact new research is making in this area.

```
In [ ]: import sys
print(f'Python: {sys.version}')

import sklearn
print(f'Scikit-Learn: {sklearn.__version__}')

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

import os
import collections
import math
import random
import PIL
import urllib
# A random seed is used to ensure that the random numbers created are the
# same for each run of this document.
random.seed(42)

from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive
```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
SciKit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.25.2
matplotlib: 3.7.1
seaborn: 0.13.1
scipy: 1.11.4

Ref: <https://www.python.org/>
Ref: <https://scikit-learn.org/stable/>
Ref: <https://pandas.pydata.org/docs/>
Ref: <https://numpy.org/doc/stable/>
Ref: <https://matplotlib.org/>
Ref: <https://seaborn.pydata.org/>
Ref: <https://docs.scipy.org/doc/scipy/index.html>
Ref: <https://pypi.org/project/python-slugify/>

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

```
In [ ]: def show_records(plt, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plt.annotate(str(len(plot_data))+' records', xy=(-1180000, 6420000), \
                xycoords='data', ha='left', color=text_colour)
```

```
In [ ]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
```

```

        "hillforts-topo-northwest-minus.png",
        "hillforts-topo-northeast.png",
        "hillforts-topo-south.png",
        "hillforts-topo-south-plus.png",
        "hillforts-topo-ireland.png",
        "hillforts-topo-ireland-north.png",
        "hillforts-topo-ireland-south.png"]
else:
    backgrounds = ["hillforts-outline-01.png",
                   "hillforts-outline-north.png",
                   "hillforts-outline-northwest-plus.png",
                   "hillforts-outline-northwest-minus.png",
                   "hillforts-outline-northeast.png",
                   "hillforts-outline-south.png",
                   "hillforts-outline-south-plus.png",
                   "hillforts-outline-ireland.png",
                   "hillforts-outline-ireland-north.png",
                   "hillforts-outline-ireland-south.png"]
return backgrounds

```

```
In [ ]: def get_bounds():
    bounds = [[-1200000, 220000, 6400000, 8700000],
              [-1200000, 220000, 7000000, 8700000],
              [-1200000, -480000, 7000000, 8200000],
              [-900000, -480000, 7100000, 8200000],
              [-520000, 0, 7000000, 8700000],
              [-800000, 220000, 6400000, 7100000],
              [-1200000, 220000, 6400000, 7100000],
              [-1200000, -600000, 6650000, 7450000],
              [-1200000, -600000, 7050000, 7450000],
              [-1200000, -600000, 6650000, 7080000]]
    return bounds
```

```
In [ ]: def show_background(plt, ax, location=""):
    backgrounds = get_backgrounds()
    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo/"

    if location == "n":
        background = os.path.join(folder, backgrounds[1])
        bounds = bounds[1]
    elif location == "nw+":
        background = os.path.join(folder, backgrounds[2])
        bounds = bounds[2]
    elif location == "nw-":
        background = os.path.join(folder, backgrounds[3])
        bounds = bounds[3]
    elif location == "ne":
        background = os.path.join(folder, backgrounds[4])
        bounds = bounds[4]
    elif location == "s":
        background = os.path.join(folder, backgrounds[5])
        bounds = bounds[5]
    elif location == "s+":
        background = os.path.join(folder, backgrounds[6])
        bounds = bounds[6]
    elif location == "i":
        background = os.path.join(folder, backgrounds[7])
        bounds = bounds[7]
    elif location == "in":
        background = os.path.join(folder, backgrounds[8])
        bounds = bounds[8]
    elif location == "is":
        background = os.path.join(folder, backgrounds[9])
        bounds = bounds[9]
    else:
        background = os.path.join(folder, backgrounds[0])
        bounds = bounds[0]

    img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
    ax.imshow(img, extent=bounds)
```

```
In [ ]: def get_counts(data):
    data_counts = []
    for col in data.columns:
        count = len(data[data[col] == 'Yes'])
        data_counts.append(count)
    return data_counts
```

```
In [ ]: def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
                129
```

```

    horizontalalignment = 'left')
ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
            size='small', color='grey', xy=(0.99, 0.01), \
            xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [ ]: def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.035), \
                xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [ ]: def plot_bar_chart(data, split_pos, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = data.columns
    x_data = [x.split('_')[split_pos:] for x in x_data]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])
    y_data = get_counts(data)
    ax.bar(x_data_new, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_from_list(data, x_labels, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = x_labels
    y_data = [len(x) for x in data]
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, n_bins):
    new_data = data.copy()
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    data[x_label].plot(kind='hist', bins = n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def get_bins(data, bins_count):
    data_range = data.max() - data.min()
    print(bins_count)
    if bins_count != None:
        x_bins = [x for x in range(data.min(), data.max(), bins_count)]
        n_bins = len(x_bins)
    else:
        n_bins = int(data_range)
        if n_bins < 10:
            multi = 10
            while n_bins < 10:
                multi *= 10
                n_bins = int(data_range * multi)
        elif n_bins > 100:

```

```

    n_bins = int(data_range)/10
    return n_bins

```

```

In [ ]: def plot_histogram(data, x_label, title, bins_count = None):
    n_bins = get_bins(data, bins_count)
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    ax.set_ylabel('Count')
    plt.title(label_format(style='plain'))
    plt.hist(data, bins=n_bins)
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_continuous(data, x_label, title):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    plt.plot(data, linewidth=4)
    plt.title(label_format(style='plain'))
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [ ]: # box plot
from matplotlib.cbook import boxplot_stats
def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.title(label_format(style='plain'))
    if o == "v":
        sns.boxplot(data=data, orient="v")
    else:
        sns.boxplot(data=data, orient="h")
    save_fig(feature + " Range")
    plt.show()

    bp = boxplot_stats(data)

    low = bp[0].get('whislo')
    q1 = bp[0].get('q1')
    median = bp[0].get('med')
    q3 = bp[0].get('q3')
    high = bp[0].get('whishi')

    return [low, q1, median, q3, high]

```

```

In [ ]: def location_XY_plot():
    plt.title(label_format(style='plain'))
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 8700000)
    add_annotation_l_xy(plt)

```

```

In [ ]: def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')

```

```

In [ ]: def plot_over_grey_numeric(merged_data, a_type, title, extra="", inner=False, \
                           fringe=False, oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:

```

```

        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

In [ ]: def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    fig, ax = plt.subplots(figsize=(9.47, 15.33))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(location_data['Location_X'], location_data['Location_Y'], \
               c='Silver')
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    show_records(plt, plot_data)
    plt.title(get_print_title('Boundary_Type: ' + boundary_type))
    save_fig('Boundary_Type_' + boundary_type)
    plt.show()

In [ ]: def plot_densiti_over_grey(data, data_type):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_densiti(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
               c=new_data['Density'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density')
    plt.title(get_print_title(f'Density - {data_type}'))
    save_fig(f'Density_{data_type}')
    plt.show()

In [ ]: def add_21Ha_line():
    x_values = \
        [-367969, -344171, -263690, -194654, -130542, -119597, -162994, -265052] \
        #, -304545]
    y_values = \
        [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 6663609] \
        #, 6611780]
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, \
              label = '≥ 21 Ha Line')
    add_to_legend = Line2D([0], [0], color='k', lw=15, alpha=0.25, \
              label = '≥ 21 Ha Line')
    return add_to_legend

In [ ]: def add_21Ha_fringe():
    x_values = \
        [-367969, -126771, 29679, -42657, -248650, -304545, -423647, -584307, -367969]
    y_values = \
        [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, 6752378, 7019842]
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    return add_to_legend

In [ ]: def add_oxford_swindon(oxford=False, swindon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_size = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_size, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swindon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_size, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches

In [ ]: def plot_over_grey(merged_data, a_type, yes_no, extra="", inner=False, \
                       fringe=False, oxford=False, swindon=False):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]

```

```

fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
show_background(plt, ax)
location_XY_plot()
add_grey()
patches = add_oxford_swindon(oxford, swindon)
plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
if fringe:
    f_for_legend = add_21Ha_fringe()
    patches.append(f_for_legend)
if inner:
    i_for_legend = add_21Ha_line()
    patches.append(i_for_legend)
show_records(plt, plot_data)
plt.legend(loc='upper left', handles=patches)
plt.title(get_print_title(f'{a_type} {extra}'))
save_fig(f'{a_type}_{extra}')
plt.show()
print(f' {round(((len(plot_data)/4147)*100), 2)}%')
return plot_data

```

```

In [ ]: def plot_type_values(data, data_type, title):
new_data = data.copy()
fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
show_background(plt, ax)
location_XY_plot()
plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
            c=new_data[data_type], cmap=cm.rainbow, s=25)
plt.colorbar(label=data_type)
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def bespoke_plot(plt, title):
add_annotation_plot(plt)
plt.tickerlabel_format(style='plain')
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def spider_plot(counts, title):
df = pd.DataFrame(counts)
categories = list(df)[1:]
categories = [x.split('_')[2] for x in categories]
N = len(categories)
values=df.loc[0].drop('group').values.flatten().tolist()
values += values[:1]
angles = [n / float(N) * 2 * math.pi for n in range(N)]
angles += angles[:1]

fig = plt.figure(figsize=(8,8))
ax = plt.subplot(111, polar=True)
plt.xticks(angles[:-1], categories, color='black', size=12)
ax.set_rlabel_position(0)
plt.yticks([100, 200, 300], ["100", "200", "300"], color="grey", size=7)
plt.ylim(0, 300)

ax.plot(angles, values, linewidth=2, color="r", linestyle='solid')
ax.fill(angles, values, 'r', alpha=0.1)
add_annotation_plot(ax)
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

```

In [ ]: def test_numeric(data):
temp_data = data.copy()
columns = data.columns
out_cols = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
feat, ent, num, non, nul = [], [], [], [], []
for col in columns:
    if temp_data[col].dtype == 'object':
        feat.append(col)
        temp_data[col+'_num'] = temp_data[col].str.isnumeric()
        entries = temp_data[col].notnull().sum()
        true_count = temp_data[col+'_num'][temp_data[col+'_num'] == \
                                            True].sum()
        null_count = temp_data[col].isna().sum()
        ent.append(entries)
        num.append(true_count)
        non.append(null_count)
    else:
        nul.append(0)

```

```

        non.append(entries_true_count)
        nul.append(null_count)
    else:
        print(f'{col} {temp_data[col].dtype}')
summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
summary.columns = out_cols
return summary

```

```
In [ ]: def find_duplicated(numeric_data, text_data, encodeable_data):
d = False
all_columns = list(numeric_data.columns) + list(text_data.columns) + \
list(encodeable_data.columns)
duplicate = [item for item, count in \
collections.Counter(all_columns).items() if count > 1]
if duplicate:
    print(f"There are duplicate features: {duplicate}")
    d = True
return d
```

```
In [ ]: def test_data_split(main_data, numeric_data, text_data, encodeable_data):
m = False
split_features = list(numeric_data.columns) + list(text_data.columns) + \
list(encodeable_data.columns)
missing = list(set(main_data)-set(split_features))
if missing:
    print(f"There are missing features: {missing}")
    m = True
return m
```

```
In [ ]: def review_data_split(main_data, numeric_data, text_data, \
                           encodeable_data = pd.DataFrame()):
d = find_duplicated(numeric_data, text_data, encodeable_data)
m = test_data_split(main_data, numeric_data, text_data, encodeable_data)
if d != True and m != True:
    print("Data split good.")
```

```
In [ ]: def find_duplicates(data):
print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')
```

```
In [ ]: def count_yes(data):
total = 0
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    total += count
    print(f'{col}: {count}')
print(f'Total yes count: {total}')
```

Null Value Functions

The following functions will be used to update null values.

```
In [ ]: def fill_nan_with_minus_one(data, feature):
new_data = data.copy()
new_data[feature] = data[feature].fillna(-1)
return new_data
```

```
In [ ]: def fill_nan_with_NA(data, feature):
new_data = data.copy()
new_data[feature] = data[feature].fillna("NA")
return new_data
```

```
In [ ]: def test_numeric_value_in_feature(feature, value):
test = feature.isin([-1]).sum()
return test
```

```
In [ ]: def test_categorical_value_in_feature(dataframe, feature, value):
test = dataframe[feature][dataframe[feature] == value].count()
return test
```

```
In [ ]: def test_cat_list_for_NA(dataframe, cat_list):
for val in cat_list:
    print(val, test_categorical_value_in_feature(dataframe, val, 'NA'))
```

```
In [ ]: def test_num_list_for_minus_one(dataframe, num_list):
for val in num_list:
    feature = dataframe[val]
    print(val, test_numeric_value_in_feature(feature, -1))
```

```
In [ ]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data
```

```
In [ ]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data
```

Reprocessing Functions

```
In [ ]: def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data
```

Save Image Functions

```
In [ ]: fig_no = 0
part = 'Part02'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"
```

```
In [ ]: def get_file_name(title):
    file_name = slugify(title)
    return file_name
```

```
In [ ]: def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(",", ";")
    return title
```

```
In [ ]: def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no
```

```
In [ ]: if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass
```

```
In [ ]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Part_02_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution,
                    bbox_inches='tight')
    else:
        pass
```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [ ]: hillforts_csv = r"https://raw.githubusercontent.com/MikeDai/rise/Hillforts-Primer/main/hillforts-atlas-source-data.csv"
# "https://raw.githubusercontent.com/MikeDai/rise/Hillforts-Primer/main/hillforts-atlas-source-data-csv/hillforts.csv"
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)

<ipython-input-54-b2855eddaef9>: 4: DtypeWarning: Columns (10, 12, 68, 83, 84, 85, 86, 165, 183) have mixed types. Specify dt
ype option on import or set low_memory=False.
    hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

Out[]:

	OBJECTID	Main_Atlas_Number	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Display_N
0	1	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Aconbury C Hereford (Aconbury Bea
1	2	2	EN	England	EN0002 Bach Camp, Herefordshire	Bach Camp	NaN	Bach C Hereford

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [ ]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
            'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.')
```

Using all 4147 record in the Hillforts Atlas.

Download Function

```
In [ ]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', 'republic-of-ireland', \
                'northern-ireland', 'isle-of-man', 'roi-ni', \
                'eng-wal-sco-iom']:
                if pkg.shape[0] == hillforts_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
                else:
                    dl = data_list[0]
            dl = dl.replace('\r', ' ', regex=True)
            dl = dl.replace('\n', ' ', regex=True)
            fn = 'hillforts_primer_' + filename
            fn = get_file_name(fn)
            dl.to_csv(fn+'.csv', index=False)
            files.download(fn+'.csv')
    else:
        pass
```

Reload Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-data packages. Where needed, these data extracts will be combined with Name and Number features to ensure the data can be understood and can, if needed, be concorded.

```
In [ ]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Reload Location

```
In [ ]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short)
location_numeric_data_short.head()
location_data = location_numeric_data_short.copy()
location_data.head()
```

	Location_X	Location_Y	Density
0	-303295	6798973	1.632859e-12
1	-296646	6843289	1.540172e-12
2	-289837	6808611	1.547729e-12
3	-320850	6862993	1.670548e-12
4	-261765	6810587	1.369981e-12

Review Data Part 2

Management Data

The Management data is split into two subgroups:

- Condition
- Land Use

```
In [ ]: management_features = [
    'Management_Condition_Extant',
    'Management_Condition_Cropmark',
    'Management_Condition_Destroyed',
    'Management_Condition_Comments',
    'Management_Land_Use_Woodland',
    'Management_Land_Use_Plantation',
    'Management_Land_Use_Parkland',
    'Management_Land_Use_Pasture',
    'Management_Land_Use_Arabie',
    'Management_Land_Use_Scrub',
    'Management_Land_Use_Outcrop',
    'Management_Land_Use_Moorland',
    'Management_Land_Use_Heath',
    'Management_Land_Use_Urban',
    'Management_Land_Use_Coastal',
    'Management_Land_Use_Other',
    'Management_Land_Use_Comments']
```

```
management_data = hillforts_data[management_features]
management_data.head()
```

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed	Management_Condition_Comments
0	Yes	No	No	Main ditch gone on N and W sides. Visitor eros...
1	Yes	No	No	Natural and animal erosion with sheep scrapes...
2	Yes	No	No	Although the circuit continuous in part, the s...
3	Yes	No	No	Slumping has occurred around the SE corner. Ra...
4	Yes	No	No	Visitor numbers great, but erosion repair has ...

'Management_Condition_Comments' and 'Management_Land_Use_Comments' contain null values.

See: [Management Text Data - Resolve Null Values](#).

In []: `management_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Management_Condition_Extant    4147 non-null   object 
 1   Management_Condition_Cropmark  4147 non-null   object 
 2   Management_Condition_Destroyed 4147 non-null   object 
 3   Management_Condition_Comments  1849 non-null   object 
 4   Management_Land_Use_Woodland  4147 non-null   object 
 5   Management_Land_Use_Plantation 4147 non-null   object 
 6   Management_Land_Use_Parkland  4147 non-null   object 
 7   Management_Land_Use_Pasture   4147 non-null   object 
 8   Management_Land_Use_Arabic   4147 non-null   object 
 9   Management_Land_Use_Scrub    4147 non-null   object 
 10  Management_Land_Use_Outcrop  4147 non-null   object 
 11  Management_Land_Use_Moorland 4147 non-null   object 
 12  Management_Land_Use_Heath    4147 non-null   object 
 13  Management_Land_Use_Urban   4147 non-null   object 
 14  Management_Land_Use_Coastal  4147 non-null   object 
 15  Management_Land_Use_Other   4147 non-null   object 
 16  Management_Land_Use_Comments 1823 non-null   object 
dtypes: object(17)
memory usage: 550.9+ KB
```

Management Numeric Data

The Management data package contains no numeric data features.

In []: `management_numeric_data = pd.DataFrame()`

Management Text Data

There are two free text Management features.

```
In [ ]: management_text_features = [
    'Management_Condition_Comments',
    'Management_Land_Use_Comments']

management_text_data = hillforts_data[management_text_features].copy()
management_text_data.head()
```

	Management_Condition_Comments	Management_Land_Use_Comments
0	Main ditch gone on N and W sides. Visitor eros...	Mixed woodland since 19th century with interna...
1	Natural and animal erosion with sheep scrapes...	Potatoes once grown on the site, but vegetatio...
2	Although the circuit continuous in part, the s...	A wooded private site with access problems.
3	Slumping has occurred around the SE corner. Ra...	The interior is arable and pasture at present....
4	Visitor numbers great, but erosion repair has ...	Upland pasture, moorland and scrub. Medieval r...

Management Text Data - Resolve Null Values

Test for the presence of 'NA' in the Management text features.

```
In [ ]: test_cat_list_for_NA(management_text_data, management_text_features)
```

```
Management_Condition_Comments 0  
Management_Land_Use_Comments 0
```

As 'NA' is not present, null values are filled with 'NA'.

```
In [ ]: management_text_data = update_cat_list_for_NA(management_text_data, \  
                                                 management_text_features)  
management_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 4147 entries, 0 to 4146  
Data columns (total 2 columns):  
 #   Column           Non-Null Count  Dtype     
 ---  --  
 0   Management_Condition_Comments  4147 non-null    object    
 1   Management_Land_Use_Comments  4147 non-null    object    
 dtypes: object(2)  
memory usage: 64.9+ KB
```

Management Encodeable Data

Management features that have the potential to be encoded.

```
In [ ]: management_encodeable_features = [  
        'Management_Condition_Extant',  
        'Management_Condition_Cropmark',  
        'Management_Condition_Destroyed',  
        'Management_Land_Use_Woodland',  
        'Management_Land_Use_Plantation',  
        'Management_Land_Use_Parkland',  
        'Management_Land_Use_Pasture',  
        'Management_Land_Use_Arabl e',  
        'Management_Land_Use_Scrub',  
        'Management_Land_Use_Outcrop',  
        'Management_Land_Use_Moorl and',  
        'Management_Land_Use_Heath',  
        'Management_Land_Use_Urban',  
        'Management_Land_Use_Coastal',  
        'Management_Land_Use_Other' ]  
  
management_encodeable_data = hi l l f o r t s _ d a t a [ m a n a g e m e n t _ e n c o d e a b l e _ f e a t u r e s ] . c o p y ()  
management_encodeable_data.head()
```

```
Out[ ]:   Management_Condition_Extant  Management_Condition_Cropmark  Management_Condition_Destroyed  Management_Land_Use_Woodland  
0                 Yes                           No                         No                      Yes  
1                 Yes                           No                         No                      No  
2                 Yes                           No                         No                      Yes  
3                 Yes                           No                         No                      Yes  
4                 Yes                           No                         No                      No
```

Condition Data

The Management features are further subdivided into 'Condition' and 'Land-Use'. These will be reviewed independently.

```
In [ ]: condition_features = [  
        'Management_Condition_Extant',  
        'Management_Condition_Cropmark',  
        'Management_Condition_Destroyed']  
  
management_condition_data = management_encodeable_data[condition_features]  
management_condition_data.head()
```

```
Out[ ]: Management_Condition_Extant  Management_Condition_Cropmark  Management_Condition_Destroyed
```

0	Yes	No	No
1	Yes	No	No
2	Yes	No	No
3	Yes	No	No
4	Yes	No	No

Condition Data Plotted

Most recorded hillforts are extant. The total 'yes' count is greater than the total number of records, meaning there are hillforts which have a 'yes' in one or more of these features. It is possible for a hillfort to be extant, a cropmark and destroyed.

```
In [ ]: count_yes(management_condition_data)
```

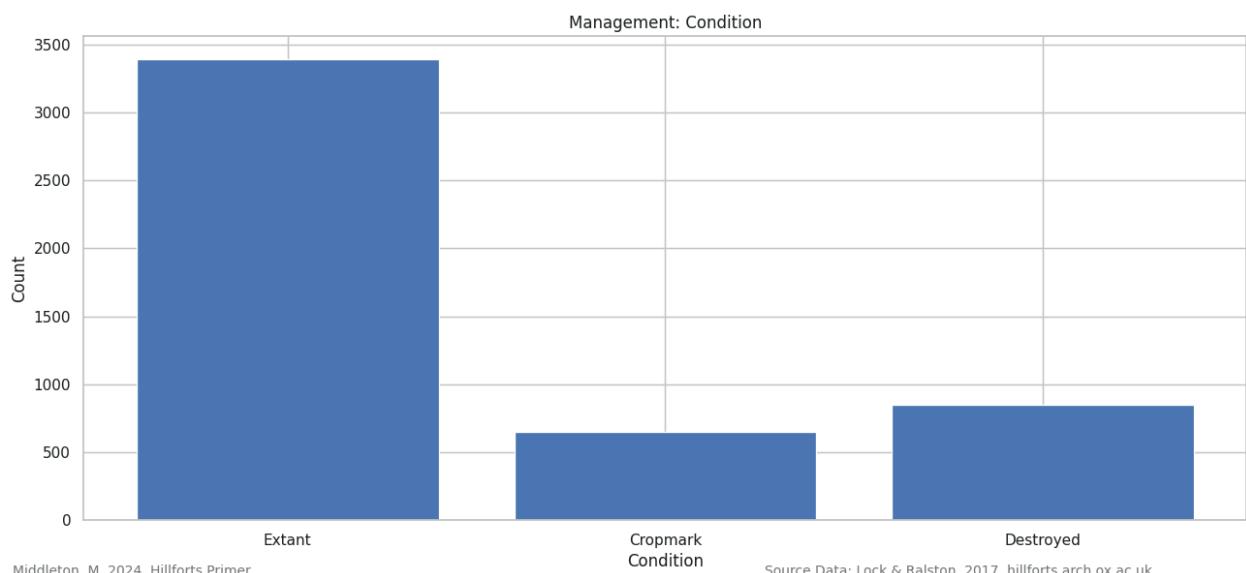
```
Management_Condition_Extant: 3391  
Management_Condition_Cropmark: 648  
Management_Condition_Destroyed: 852  
Total yes count: 4891
```

```
In [ ]: management_condition_data.loc[\n    (management_condition_data['Management_Condition_Cropmark'] == 'Yes') &\n    (management_condition_data['Management_Condition_Destroyed'] == 'Yes')].head()
```

```
Out[ ]: Management_Condition_Extant  Management_Condition_Cropmark  Management_Condition_Destroyed
```

26	Yes	Yes	Yes
158	No	Yes	Yes
340	No	Yes	Yes
375	Yes	Yes	Yes
410	Yes	Yes	Yes

```
In [ ]: plot_bar_chart(management_condition_data, 2, 'Condition', 'Count', 'Management: Condition')
```



Condition Data Mapped

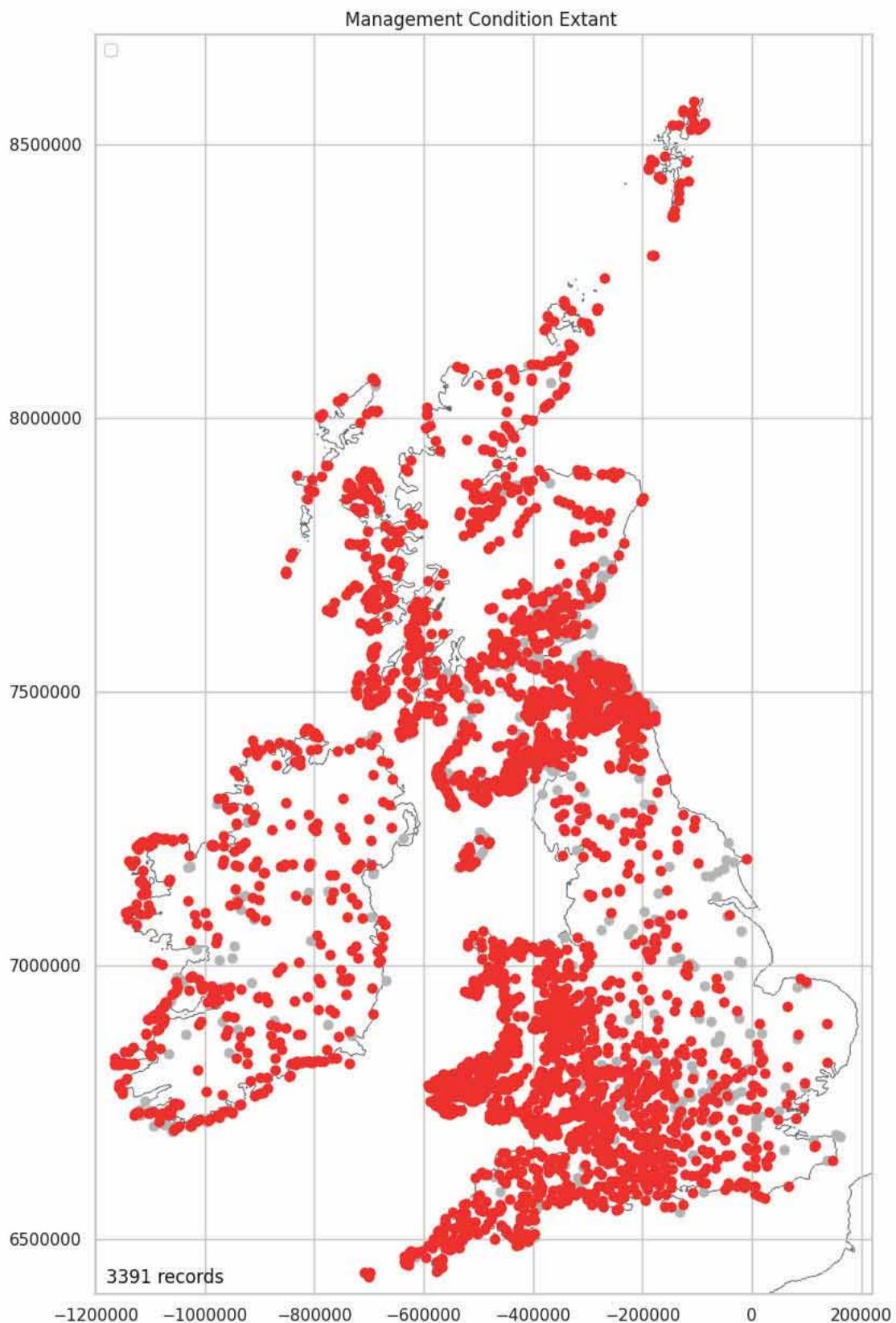
The Condition data is recombined with the 'Location' data so it can be mapped.

```
In [ ]: location_condition_data = pd.merge(location_numeric_data_short, \n                                             management_condition_data, \n                                             left_index=True, \n                                             right_index=True)
```

Extant Mapped

There are 3391 hillforts (81.77%) identified as extant. Of these, 2686 are fully extant.

```
In [ ]: man_extent = plot_over_grey(location_condition_data, \n                                    'Management_Condition_Extant', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

81.77%

```
In [ ]: extant_only = management_condition_data[
    (management_condition_data['Management_Condition_Extant'] == 'Yes') & \
    (management_condition_data['Management_Condition_Cropmark'] == 'No') & \
    (management_condition_data['Management_Condition_Destroyed'] == 'No')]
print(f'Extant only: {len(extant_only)})')

Extant only: 2686
```

Cropmark Mapped

There are 648 hillforts (15.63%), recorded as a cropmark. Of these 477 are known exclusively from the cropmark record.

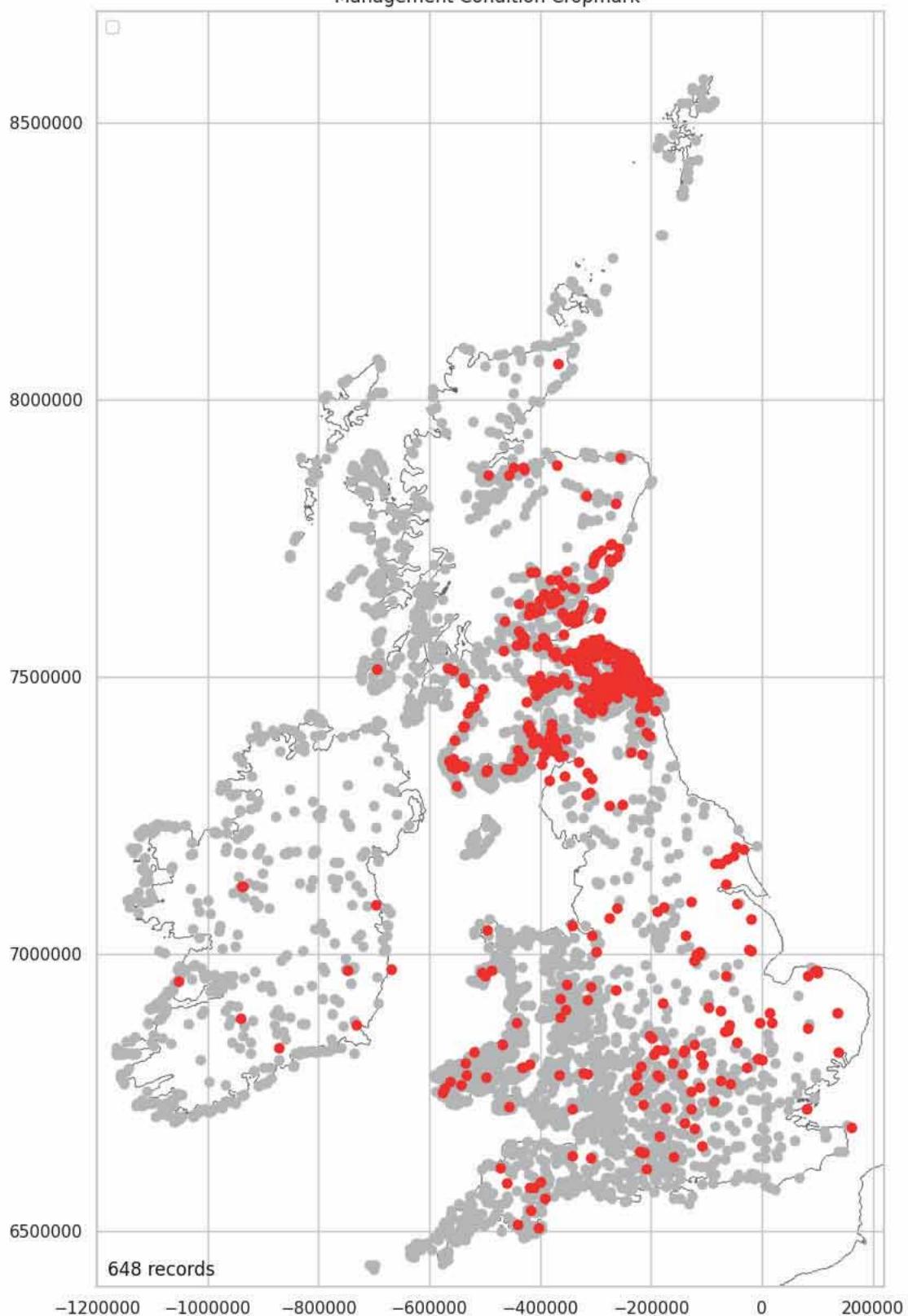
There are two forms of bias in this data. The first is a recording bias. There are 530 cropmark forts in the northern data compared to 118 in the South. There are only eight in Ireland. Even in the north, the map shows there is a concentration of records across the south-eastern lowlands of Scotland. This corresponds to the second area of bias, cropmark visibility bias toward areas of arable land. 480 of the 530 cropmark forts in the north (90.5%) are in areas of arable land. See: [Arable Mapped](#).

There are 1598 hillforts in the data package isolating the density cluster over the Southern Uplands (See [Part 1: Northeast Density](#)). 458 of the hillforts in this area (28.6%) are only known because they have been recorded as a cropmark. This high number of cropmark forts, compared to the very low numbers in other areas, means the analysis of clusters is not comparing like-for-like. The low density of cropmark forts outside the Northeast is depressing the intensity of the clusters in these areas.

It is not possible to tell from the data if the variation in distribution of cropmark hillforts relate to a difference in recording intensity between Scotland and the other nations or, if a more inclusive definition of hillfort was adopted in Scotland during the data gathering phase of the Hillforts Atlas.

```
In [ ]: man_cropmark = plot_over_grey(location_condition_data, \
    'Management_Condition_Cropmark', 'Yes')
```

Management Condition Cropmark

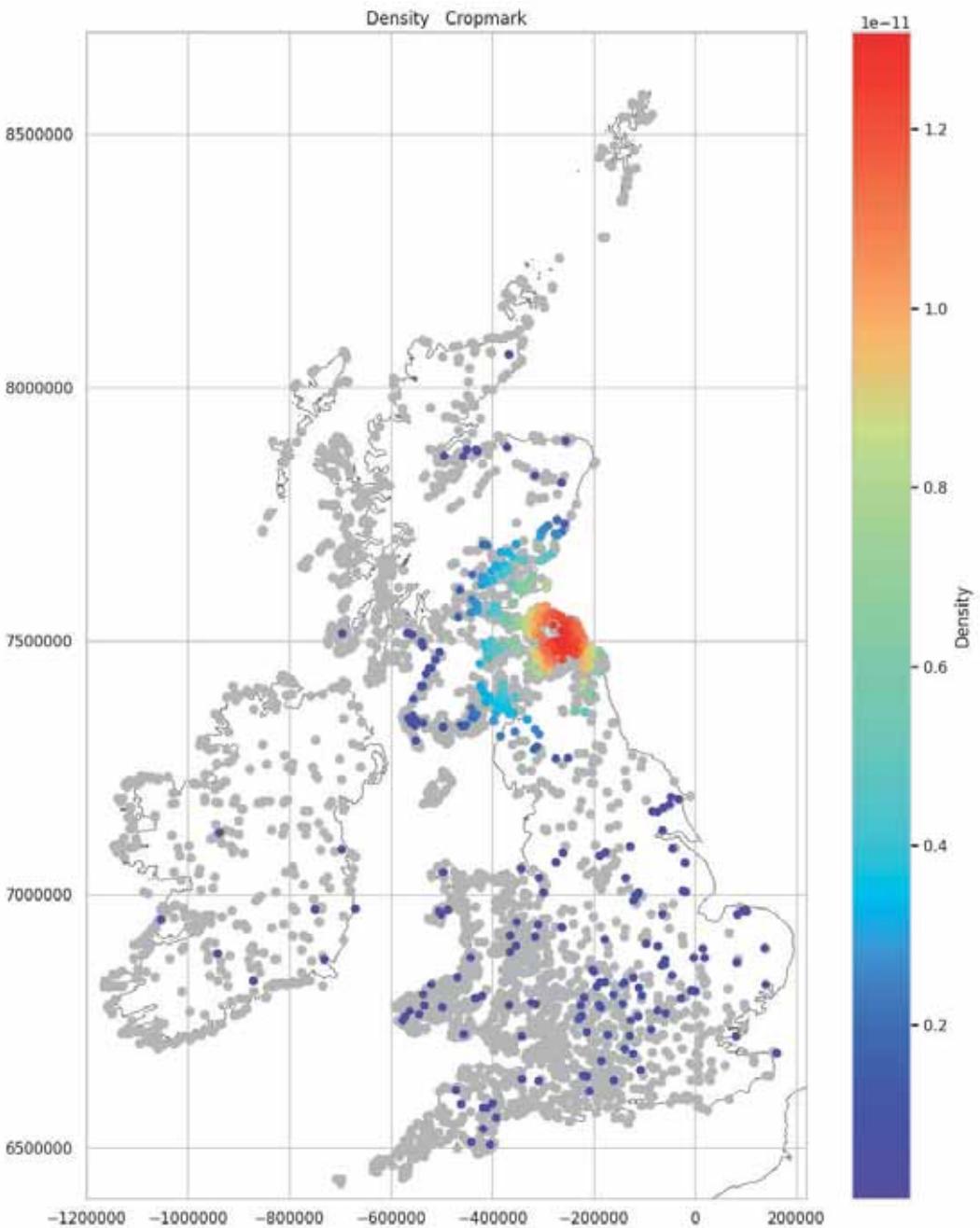


Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

15.63%

```
In [ ]: plot_density_over_grey(man_cropmark, 'Cropmark')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

```
In [ ]: cropmark_only = management_condition_data \
[(management_condition_data['Management_Condition_Cropmark']=='Yes') & \
(management_condition_data['Management_Condition_Extant']=='No') & \
(management_condition_data['Management_Condition_Destroyed']=='No')]
print(f'Cropmark only: {len(cropmark_only)}')
print(f'{round((len(cropmark_only)/4147)*100, 2)}% of the total number of hillforts in the Atlas are known exclusively from cropmarks.')
Cropmark only: 477
11.5% of the total number of hillforts in the Atlas are known exclusively from cropmarks.
```

```
In [ ]: split_location_y = 707000
split_location_x = -500000
location_management_encodeable_data = pd.merge(location_numeric_data_short, \
management_encodeable_data, \
left_index=True, right_index=True)
north_location_condition_data = \
location_management_encodeable_data[location_management_encodeable_data['Location_Y'] \
>= split_location_y].copy().reset_index(drop=True)
north_east_location_condition_data = \
north_location_condition_data[north_location_condition_data['Location_X'] >= \
split_location_x].copy().reset_index(drop=True)
cropmark_north_east_location_condition_data = \
north_east_location_condition_data[north_east_location_condition_data\
['Management_Condition_Cropmark']=='Yes']
print(f'There are {len(cropmark_north_east_location_condition_data)} cropmark forts in the Northeast.')
```

There are 503 cropmark forts in the Northeast.

```
In [ ]: cropmark_north_east_location_condition_data_arable = \
cropmark_north_east_location_condition_data\[cropmark_north_east_location_condition_data['Management_Land_Use_Arable']== 'Yes' ]
print(f'{len(cropmark_north_east_location_condition_data_arable)} of the forts in the Northeast are located in areas
print(f'{round((len(cropmark_north_east_location_condition_data_arable)/len(cropmark_north_east_location_condition_d
458 of the forts in the Northeast are located in areas of arable land use.
91.05%
```

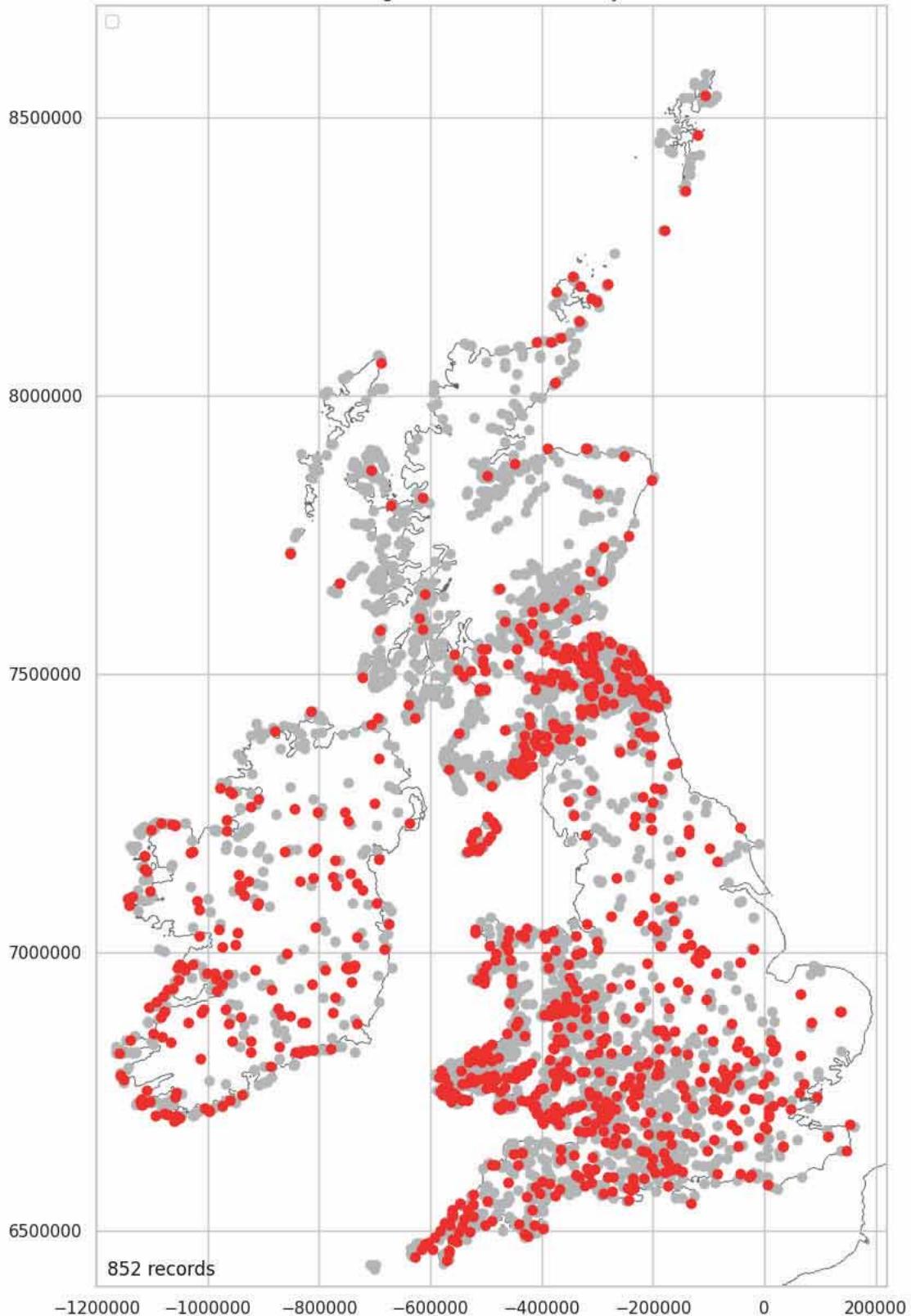
```
In [ ]: north_east_cropmark_only = \
cropmark_north_east_location_condition_data_arable\[cropmark_north_east_location_condition_data_arable\[ 'Management_Condition_Cropmark' ] == 'Yes' ) & \
(cropmark_north_east_location_condition_data_arable\[ 'Management_Condition_Extant' ] == 'No' ) & \
(cropmark_north_east_location_condition_data_arable\[ 'Management_Condition_Destroyed' ] == 'No' )
print(f'Of the 458 cropmark forts in the Northeast, {len(north_east_cropmark_only)} survive only as a cropmark.')
print(f'{round((len(north_east_cropmark_only)/len(cropmark_north_east_location_condition_data_arable))*100, 2)}%')
Of the 458 cropmark forts in the Northeast, 377 survive only as a cropmark.
82.31%
```

Destroyed Mapped

There are 852 hillforts (20.54%) identified as destroyed. Of these 212 (5.1%) have been entirely erased.

```
In [ ]: man_destroyed = \
plot_over_grey(location_condition_data, 'Management_Condition_Destroyed', 'Yes')
```

Management Condition Destroyed



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

20. 54%

```
In [ ]: destroyed_only = \
management_condition_data[(management_condition_data\
['Management_Condition_Destroyed']=='Yes') & \
(management_condition_data['Management_Condition_Extant']=='No') & \
(management_condition_data['Management_Condition_Cropmark']=='No')]
print(f'Destroyed only: {len(destroyed_only)})'
```

Destroyed only: 212

Condition Not Recorded

There are 24 records that are not recorded as being extant, a cropmark or destroyed.

```
In [ ]: all_no = \
management_conditon_data[(management_conditon_data\
['Management_Condition_Destroyed'] == 'No') & \
(management_conditon_data['Management_Condition_Extant'] == 'No') & \
(management_conditon_data['Management_Condition_Cropmark'] == 'No')]
print(f'Condition not recorded: {len(all_no)}')
```

Condition not recorded: 24

Five records are shown. To see the full list, update the 5, in the square brackets below, to 24 and rerun the document as described in [User Settings](#).

```
In [ ]: all_no_full = pd.merge(name_and_number, all_no, left_index=True, right_index=True)
all_no_full[:5]
```

	Main_Atlas_Number	Main_Display_Name	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_I
355	365	Harborough Hill, Churchill, Worcestershire	No	No	
1742	1830	Prae Wood, Hertfordshire (St. Albans; Verulami...)	No	No	
1743	1831	Wheathampstead, Hertfordshire	No	No	
1744	1832	Braughing, Hertfordshire (Gatesbury Camp)	No	No	
1747	1835	Béal Deirg Mór, Mayo	No	No	

Land Use Data

The Land Use data consists of 11 land use types and 'Other'. There is a bias in this data caused by there being a mix of habitat and land use classifications used across the Atlas.

```
In [ ]: land_use_features = [
'Management_Land_Use_Woodland',
'Management_Land_Use_Plantation',
'Management_Land_Use_Parkland',
'Management_Land_Use_Pasture',
'Management_Land_Use_Arabl e',
'Management_Land_Use_Scrub',
'Management_Land_Use_Outcrop',
'Management_Land_Use_Moorl and',
'Management_Land_Use_Heath',
'Management_Land_Use_Urban',
'Management_Land_Use_Coastal',
'Management_Land_Use_Other']

land_use_data = management_encodeable_data[land_use_features].copy()
land_use_data.head()
```

	Management_Land_Use_Woodland	Management_Land_Use_Plantation	Management_Land_Use_Parkland	Management_Land_Use_Pasture	Management_Land_Use_Other
0	Yes		No	No	No
1	No		No	No	Yes
2	Yes		No	No	No
3	Yes		No	No	Yes
4	No		No	No	Yes

Land Use Data Plotted

The 'yes' count shows that a hillfort can have multiple land use values. The feature, 'Management_Land_Use_Outcrop' contains only negative values and therefore has no predictive worth. It will be dropped. See: [Drop Land Management Features](#).

```
In [ ]: count_yes(land_use_data)
```

```

Management_Land_Use_Woodland: 880
Management_Land_Use_Plantation: 181
Management_Land_Use_Parkland: 101
Management_Land_Use_Pasture: 2314
Management_Land_Use_Arable: 712
Management_Land_Use_Scrub: 961
Management_Land_Use_Outcrop: 0
Management_Land_Use_Moorland: 875
Management_Land_Use_Heath: 27
Management_Land_Use_Urban: 255
Management_Land_Use_Coastal: 419
Management_Land_Use_Other: 386
Total yes count: 7111

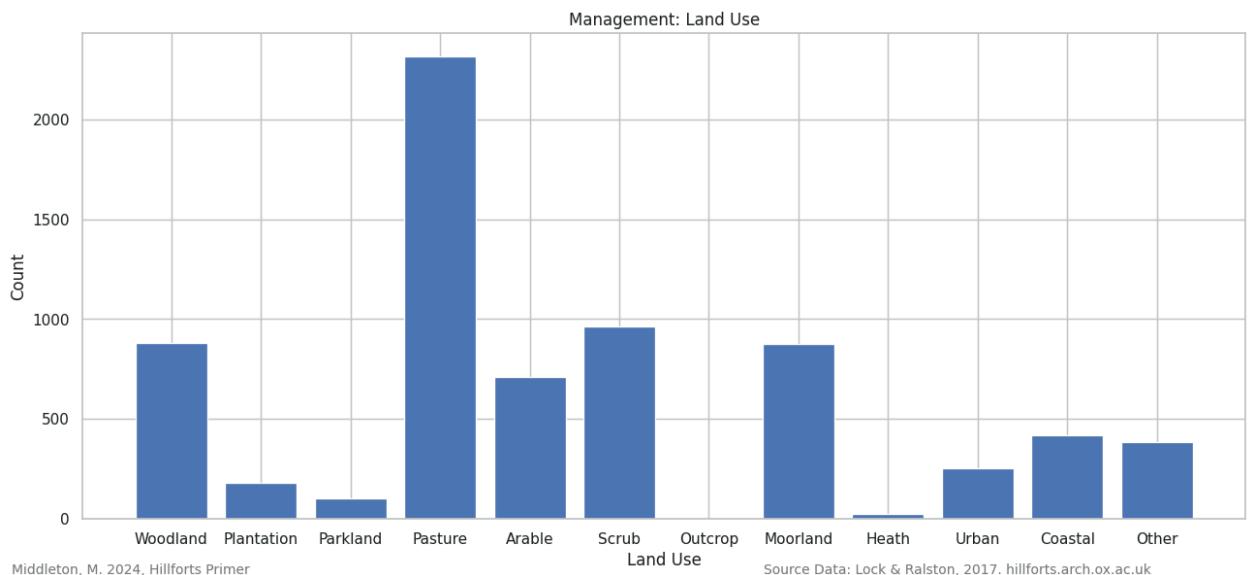
```

```
In [ ]: land_use_data.loc[(land_use_data['Management_Land_Use_Scrub'] == 'Yes') & \
(land_use_data['Management_Land_Use_Moorland'] == 'Yes')].head()
```

	Management_Land_Use_Woodland	Management_Land_Use_Plantation	Management_Land_Use_Parkland	Management_Land_Use_Pasture
4	No		No	Yes
88	Yes		No	No
89	No		No	No
236	No		No	No
265	No		No	Yes

The most frequently recorded land use is pasture which contains 2314 hillforts. Woodland, arable, scrub and moorland are roughly similar in proportion with between 700 to 900 hillforts with the remainder having around 400 or less. Heath has only 27 records and this highlights a terminology bias due to an inconsistency of terminology used atlas wide. Scrub and heath are habitat definitions and may better be classified by the land use classification, moorland. Using combined figures, 50.8% of hillforts are in pasture, 40.66% in moorland (including scrub and heath) and 25.58% are under trees.

```
In [ ]: plot_bar_chart(land_use_data, 3, 'Land Use', 'Count', 'Management: Land Use')
```



The percentages below are the number of hillforts recorded by each land use type compared to the total number of hillforts. It is important to remember that hillforts can contain multiple land use types and for this reason the combined percentage total is greater than 100%. This indicates that the average, across all forts is 1.7 land use types per fort. This in turn tells us that the land use, over most hillforts, is partitioned to some degree.

```

total = 0
for feature in land_use_features:
    feature_parts = feature.split('_')
    yes_count = land_use_data[land_use_data[feature]=='Yes']
    pcnt = round((len(yes_count)/len(land_use_data))*100, 2)
    total+=pcnt
    print(f'{feature_parts[-1]}: {pcnt}%')
print(f'Total: {total}%')

```

```
Woodl and: 21.22%
Pl antation: 4.36%
Parkland: 2.44%
Pasture: 55.8%
Arable: 17.17%
Scrub: 23.17%
Outcrop: 0.0%
Moorland: 21.1%
Heath: 0.65%
Urban: 6.15%
Coastal: 10.1%
Other: 9.31%
Total: 171.47%
```

Land Use Data Mapped

The 'Land Use' data is recombined with the 'Location' data so it can be mapped.

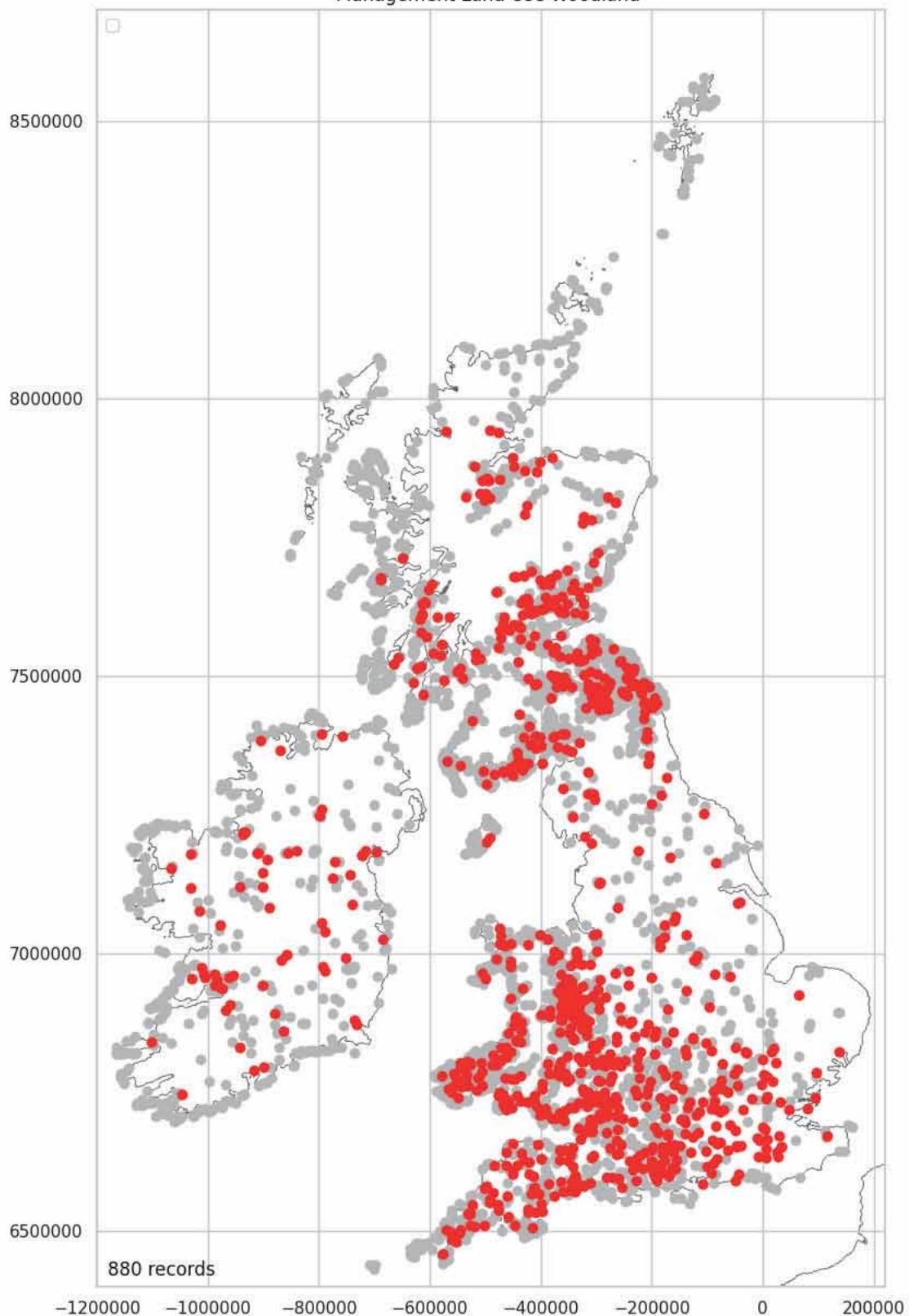
```
In [ ]: location_land_use_data = pd.merge(location_numeric_data_short, \
                                             land_use_data, left_index=True, \
                                             right_index=True)
```

Woodland Mapped

There are 880 hillforts (22.22%) in woodland.

```
In [ ]: lu_woodl and = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Woodl and', 'Yes')
```

Management Land Use Woodland



Middleton, M. 2024, Hillforts Primer

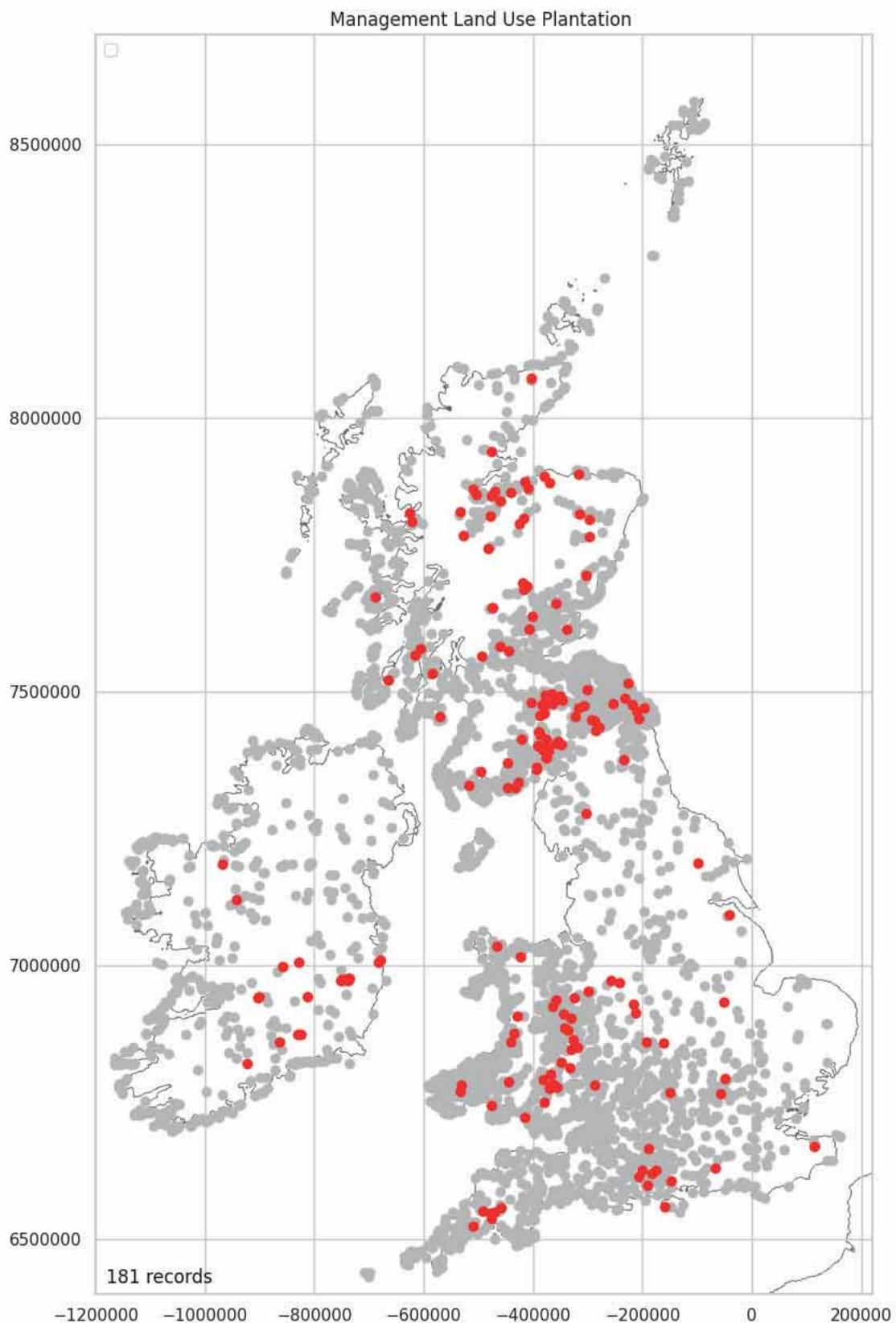
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

21.22%

Plantation Mapped

There are 181 hillforts (4.36%) under plantation.

```
In [ ]: lu_plantation = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Plantation', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

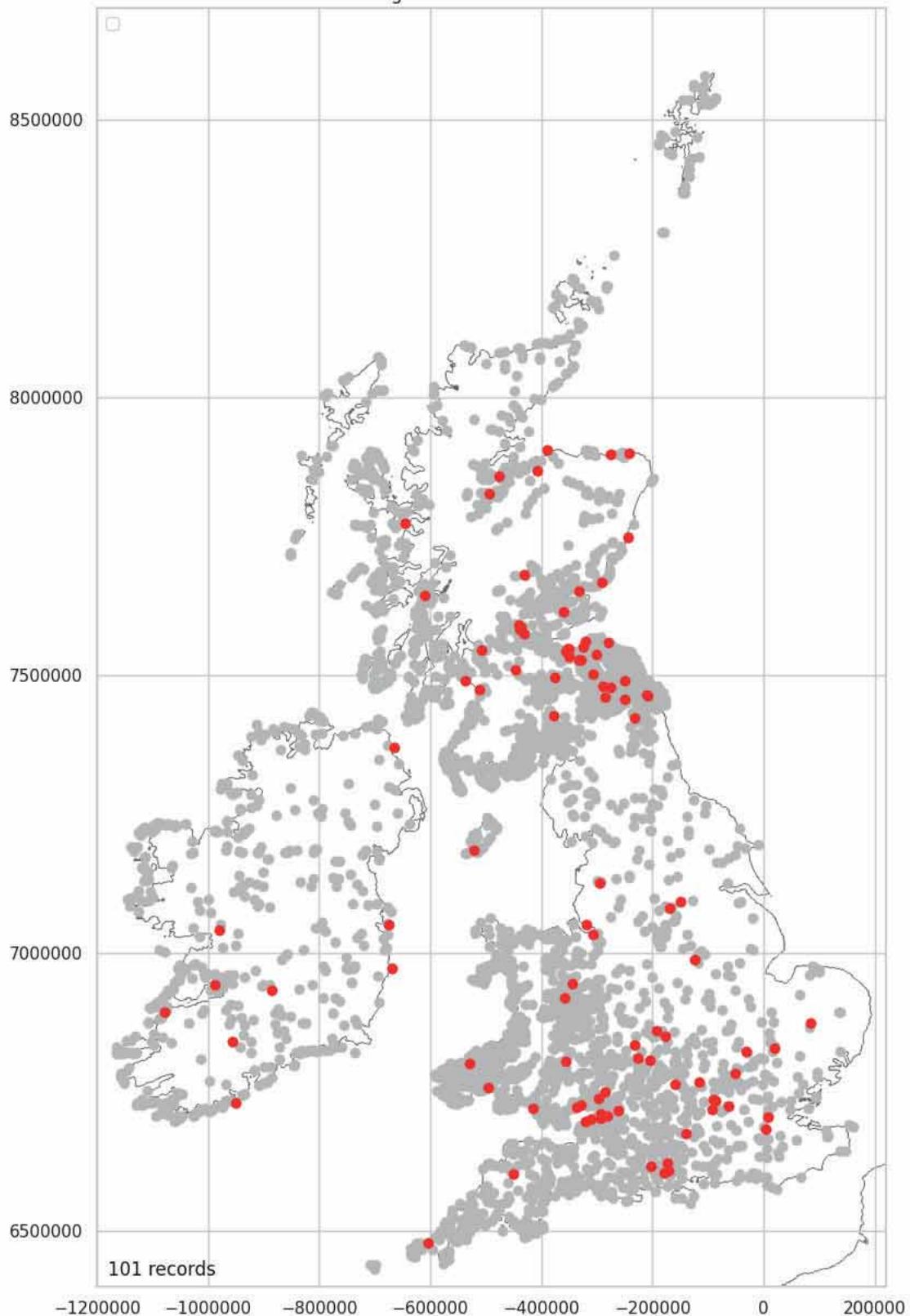
4. 36%

Parkland Mapped

There are 101 hillforts (2.44%) in parkland.

```
In [ ]: lu_parkland = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Parkland', 'Yes')
```

Management Land Use Parkland



Middleton, M. 2024, Hillforts Primer

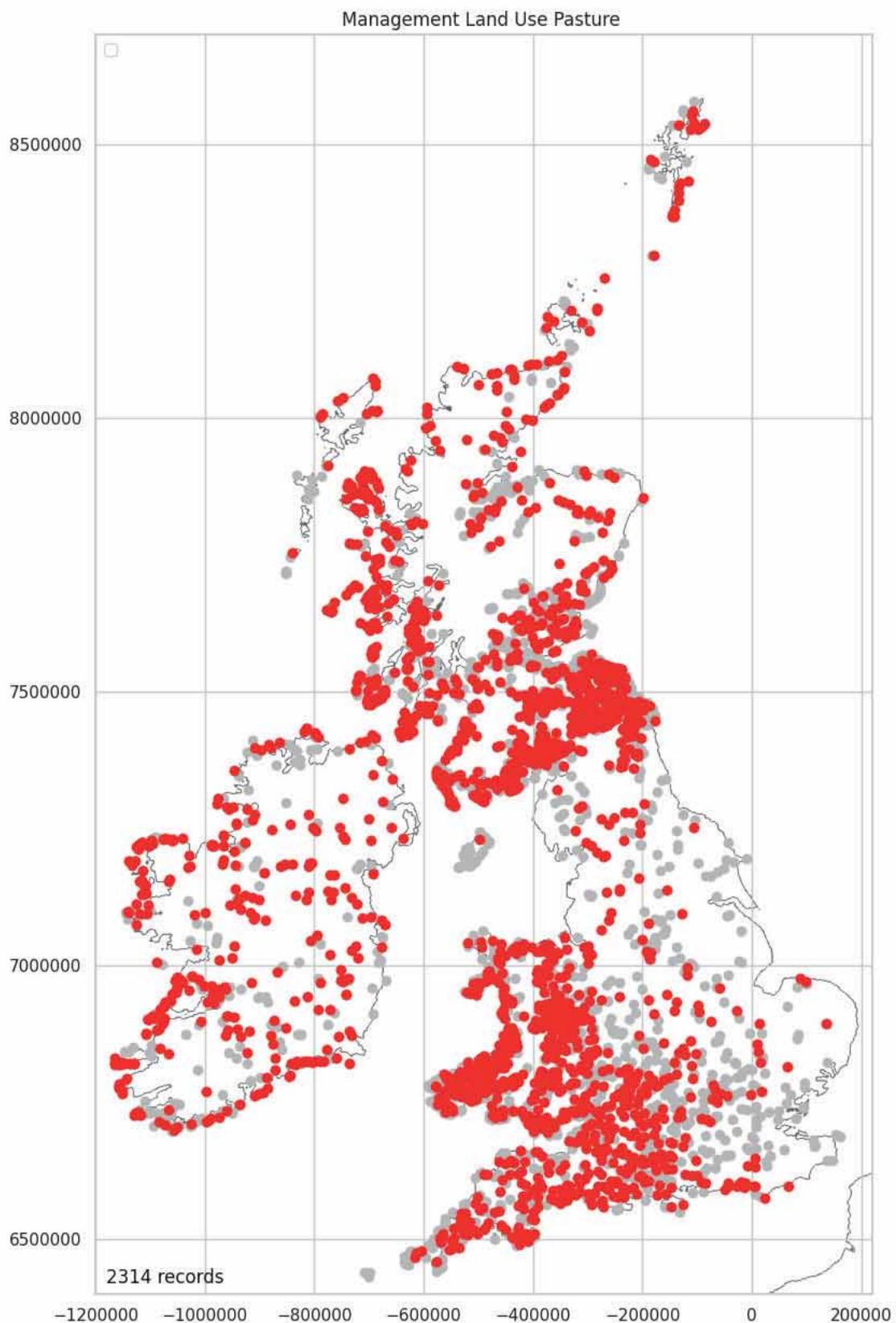
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2.44%

Pasture Mapped

There are 2114 hillforts (55.8%) in pasture.

```
In [ ]: lu_pasture = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Pasture', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

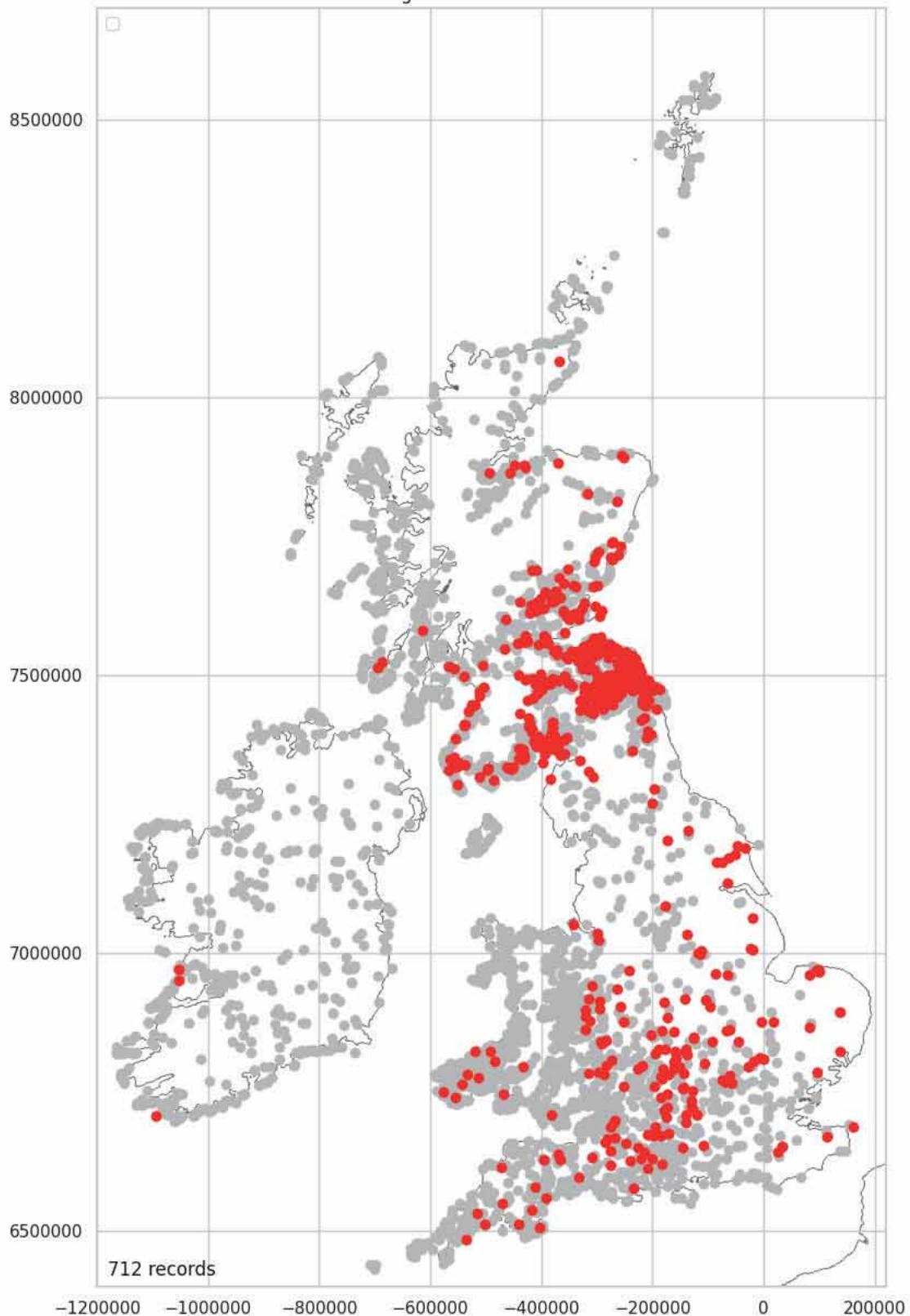
55.8%

Arable Mapped

There are 712 hillforts (17.17%) within arable farmland. 546 of these are all, or in part, cropmarks. See: [Cropmark Mapped](#).

```
In [ ]: lu_arable = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Arable', 'Yes')
```

Management Land Use Arable



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

17. 17%

```
In [ ]: arable_cm = \
len(management_data.loc[(management_data['Management_Land_Use_Arable']=='Yes') & \
(management_data['Management_Condition_Cropmark']=='Yes')])
```

```
Out[ ]: 546
```

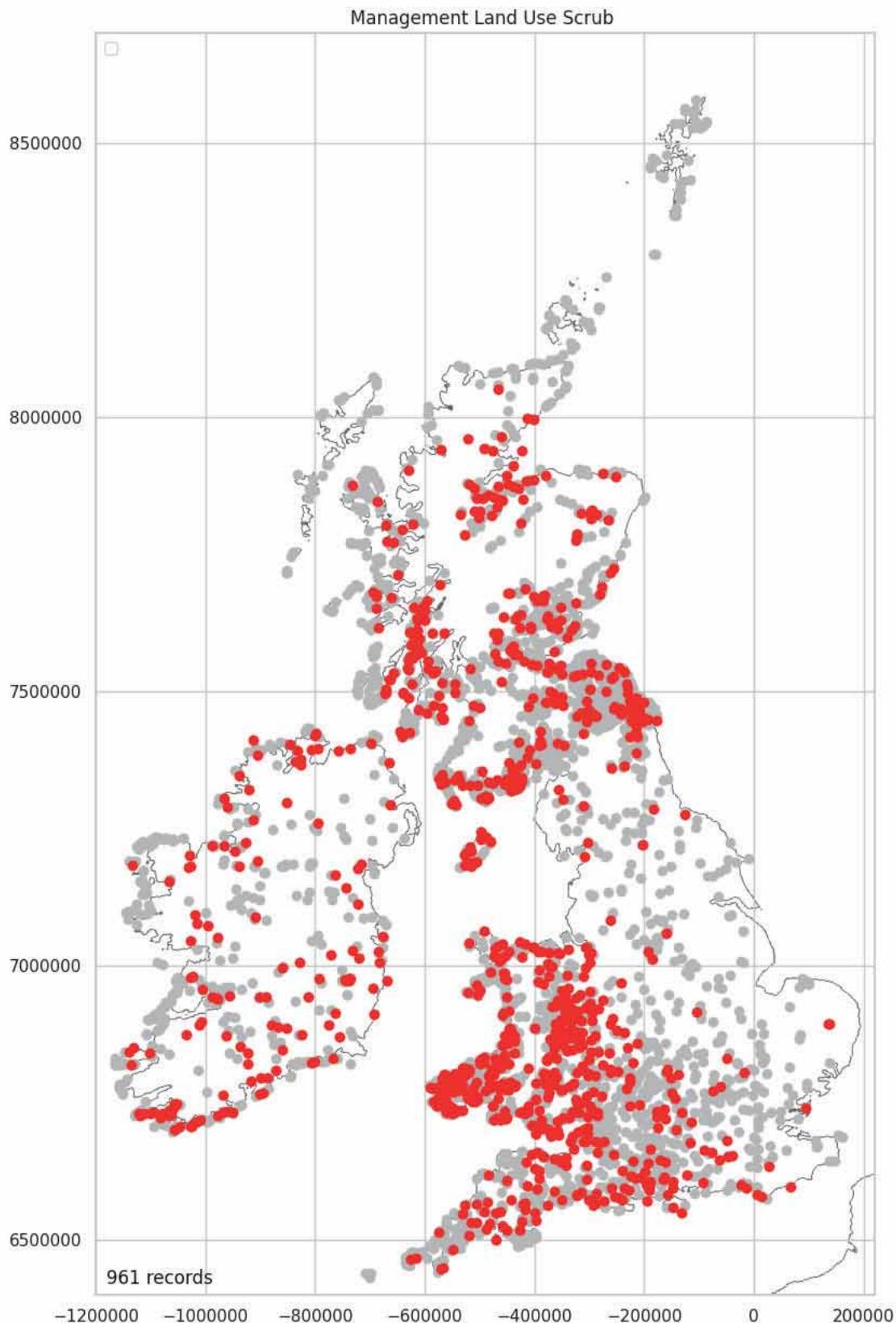
Scrub Mapped

961 hillforts (23.17%) are in scrub. This terminology is potentially confusing, as scrub is a habitat type, and is likely to be interchangeable with the terms moorland and heath.

See: [Mountain heath and montane scrub & UK BAP Priority Habitats](#).

See: [Moorland, Scrub and Heath Pooled and Mapped](#)

```
In [ ]: lu_scrub = \n    plot_over_grey(location_land_use_data, 'Management_Land_Use_Scrub', 'Yes')
```



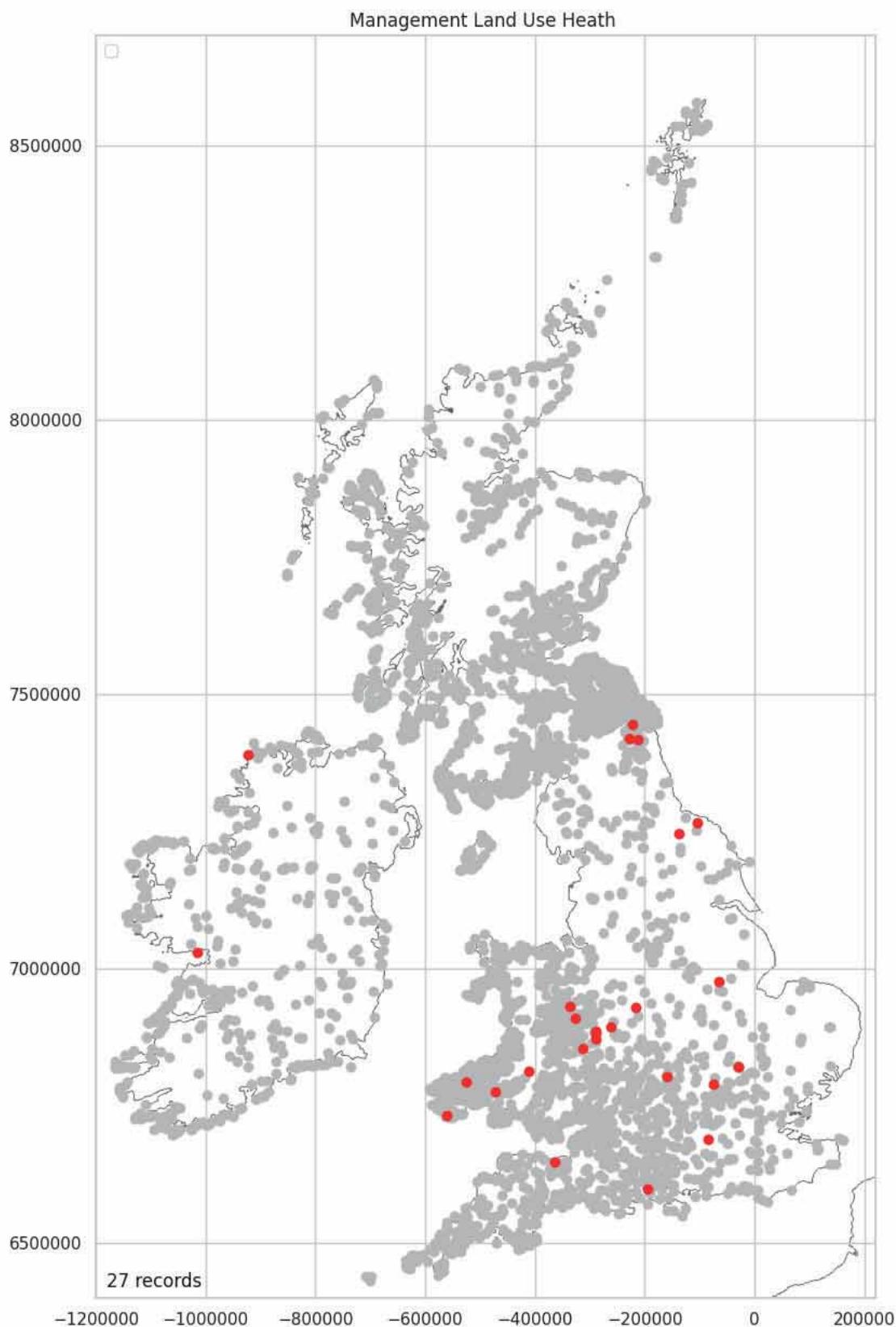
Heath Mapped

Only 27 hillforts (0.65%) are in heath. This terminology is potentially confusing, as heath is a habitat type, and is likely to be interchangeable with the terms moorland and scrub.

See: [Mountain heath and montane scrub & UK BAP Priority Habitats](#).

See: [Moorland, Scrub and Heath Pooled and Mapped](#)

```
In [ ]: lu_heath = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Heath', 'Yes')
```

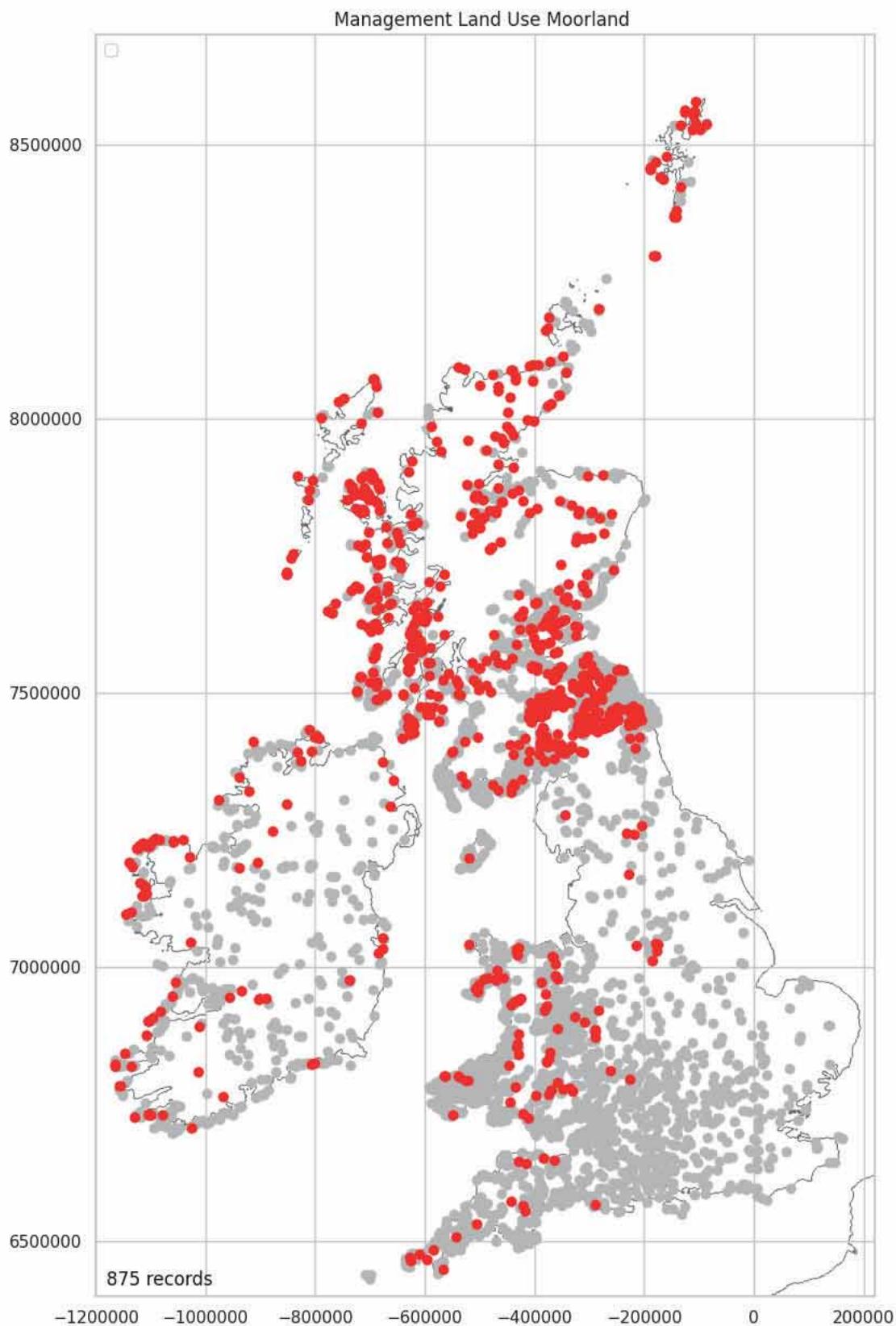


Moorland Mapped

875 hillforts (21.17%) are in moorland. This terminology is potentially confusing, and is likely to be interchangeable with the terms heath and scrub.

See: [Moorland, Scrub and Heath Pooled and Mapped](#)

```
In [ ]: lu_moorl and = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Moorl and', 'Yes')
```



Rough Grazing: Moorland, Scrub and Heath Combined and Mapped

Heath, moorland and scrub may better be referred to as the land use type Rough Grazing. Combined, moorland, scrub and heath gives a total 1686 hillforts (40.66%).

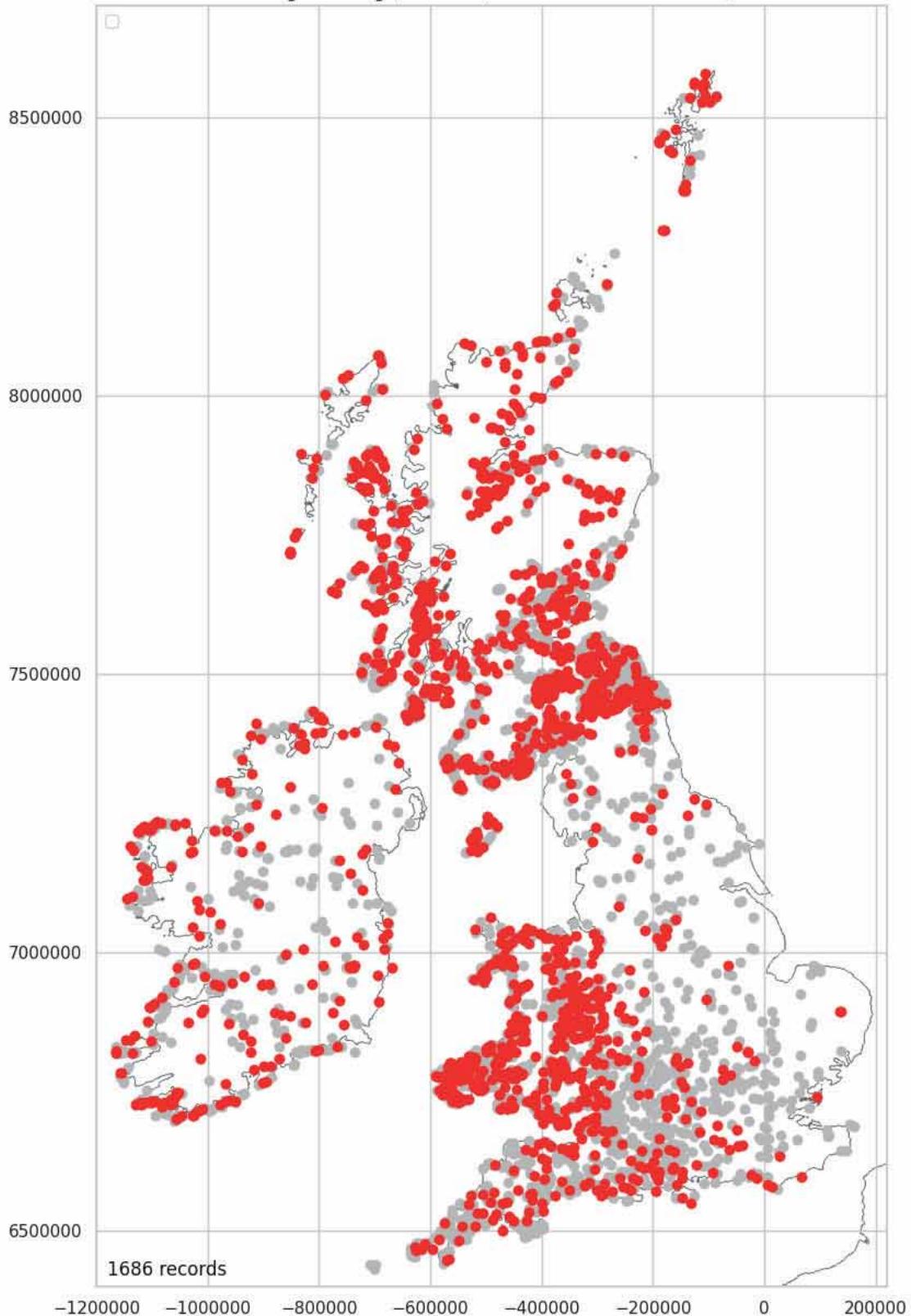
See: [Heath Mapped](#)

See: [Scrub Mapped](#)See: [Moorland Mapped](#)

```
In [ ]: temp_moorl and = location_land_use_data.copy()
temp_moorl and.loc[temp_moorl and['Management_Land_Use_Scrub'] == \
'Yes', 'Management_Land_Use_Moorl and'] = 'Yes'
temp_moorl and.loc[temp_moorl and['Management_Land_Use_Heath'] == \
'Yes', 'Management_Land_Use_Moorl and'] = 'Yes'
temp_moorl and['Rough_Grazing'] = temp_moorl and['Management_Land_Use_Moorl and']
```

```
In [ ]: lu_moorl and_temp = plot_over_grey(temp_moorl and, 'Rough_Grazing', \
'Yes', '(Moorl and, Heath & Scrub Combined)')
```

Rough Grazing (Moorland; Heath & Scrub Combined)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

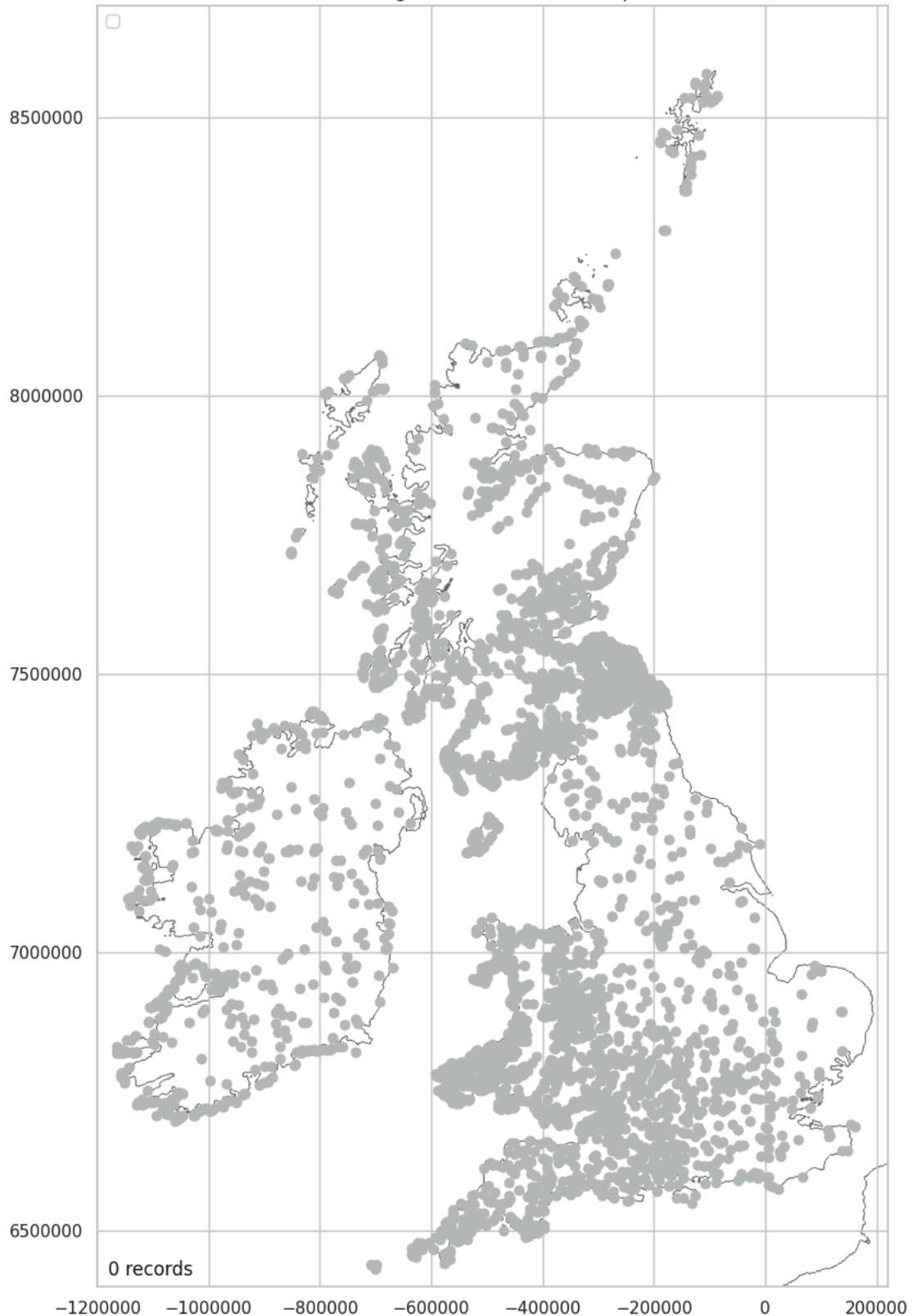
40. 66%

Outcrop Mapped

There are no hillforts with the classification 'Outcrop'. As all records contain the same information it has no predictive value. See: [Drop Land Management Features](#)

```
In [ ]: lu_outcrop = plot_over_grey(location_land_use_data, \
    'Management_Land_Use_Outcrop', 'Yes')
```

Management Land Use Outcrop



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

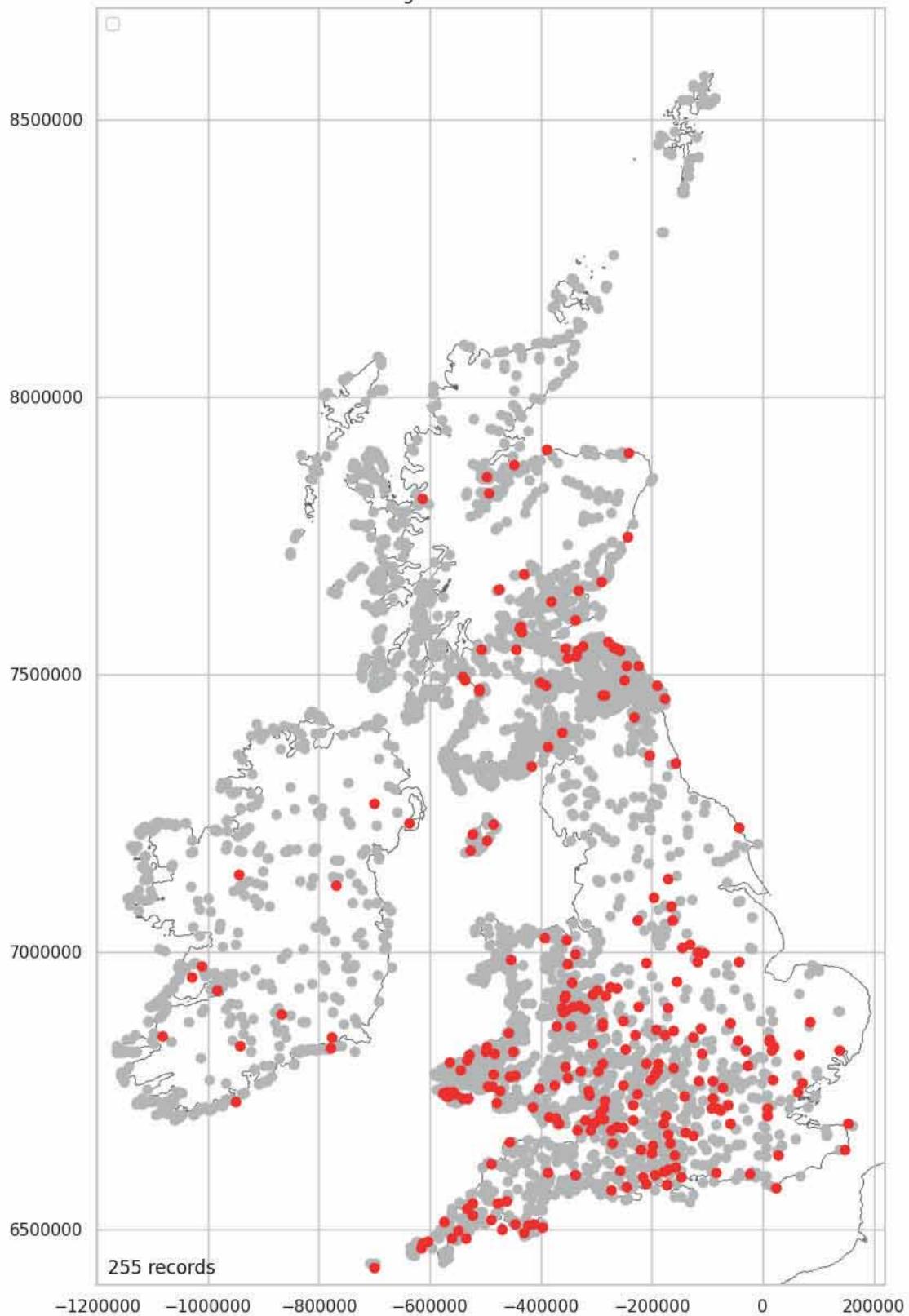
0.0%

Urban Mapped

255 hillforts (6.15%) are classified as having an urban land use.

```
In [ ]: lu_urban = plot_over_grey(location_land_use_data, \
    'Management_Land_Use_Urban', 'Yes')
```

Management Land Use Urban



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 15%

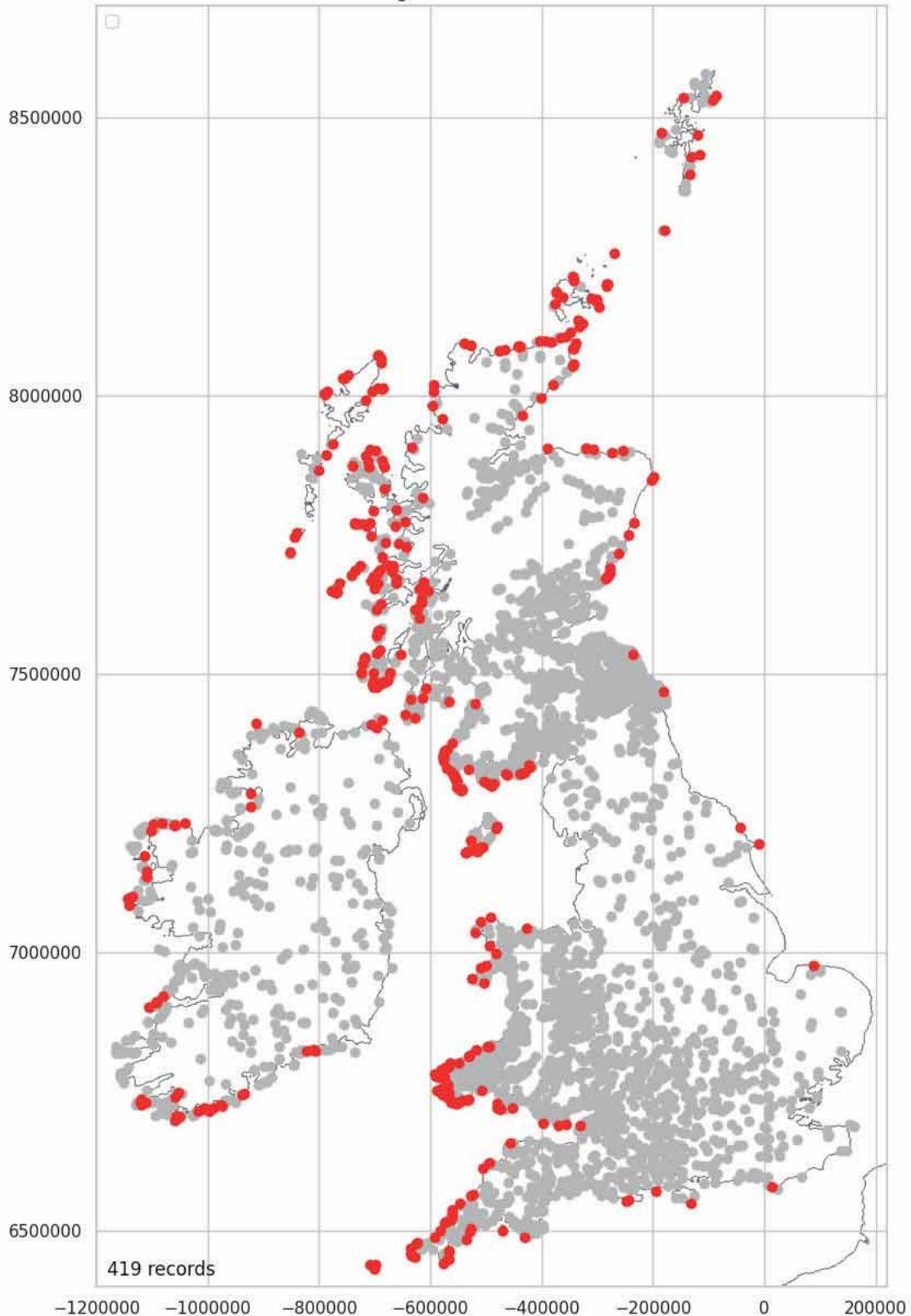
Coastal Mapped (Land Use)

419 hillforts (10.1%) are classified as having a coastal land use. Coastal is either a topographic location or a habitat type and does not convey a land use.

See: [Coastal Mapped \(Topo\)](#)

```
In [ ]: lu_coastal = plot_over_grey(location_land_use_data, \
    'Management_Land_Use_Coastal', 'Yes')
```

Management Land Use Coastal



Middleton, M. 2024, Hillforts Primer

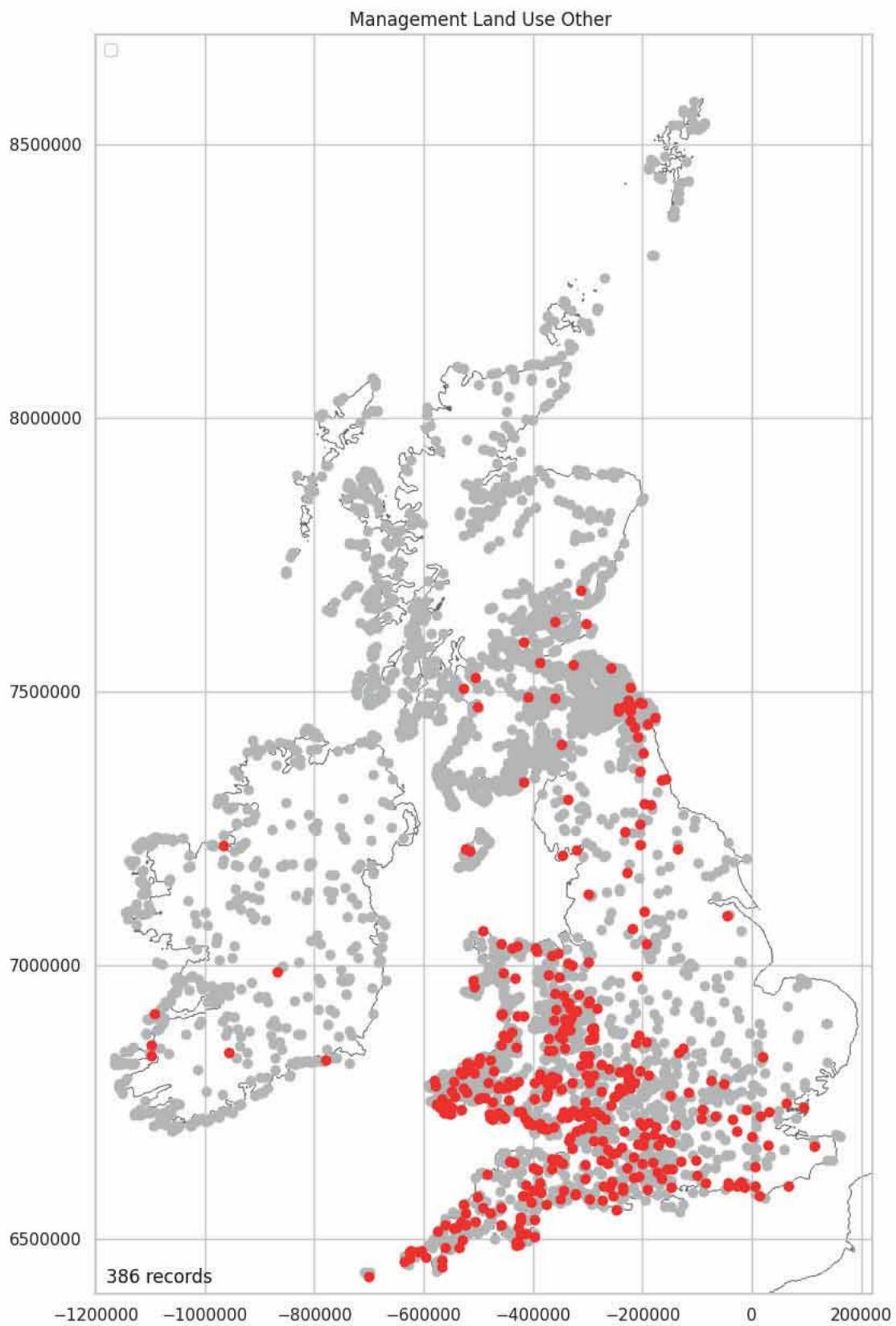
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

10.1%

Other Mapped

386 hillforts (9.31%) are classified as having an other land use.

```
In [ ]: lu_other = plot_over_grey(location_land_use_data, \
                                'Management_Land_Use_Other', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

9.31%

Land Use Not Recorded

There are 37 hillforts with no identified land use.

```
In [ ]: all_no_lu = land_use_data[(land_use_data['Management_Land_Use_Woodl and'] == 'No') &
                               (land_use_data['Management_Land_Use_Plantation'] == 'No') &
                               (land_use_data['Management_Land_Use_Parkl and'] == 'No') &
                               (land_use_data['Management_Land_Use_Pasture'] == 'No') &
                               (land_use_data['Management_Land_Use_Arabl e'] == 'No') &
                               (land_use_data['Management_Land_Use_Scrub'] == 'No') &
```

```

(I and_use_data['Management_Land_Use_Outcrop']=='No') &
(I and_use_data['Management_Land_Use_Moorl and']=='No') &
(I and_use_data['Management_Land_Use_Heath']=='No') &
(I and_use_data['Management_Land_Use_Urban']=='No') &
(I and_use_data['Management_Land_Use_Coastal']=='No') &
(I and_use_data['Management_Land_Use_Other']=='No'))
print(f'Land use not recorded: {len(all_no_lu)}')

```

Land use not recorded: 37

Five records are shown. To see the full list, update the 5, in the square brackets below, to 37 and rerun the document as described in [User Settings](#).

```
In [ ]: all_no_lu_full = pd.merge(name_and_number, all_no_lu, left_index=True, right_index=True)
all_no_lu_full [:5]
```

	Main_Atlas_Number	Main_Display_Name	Management_Land_Use_Woodland	Management_Land_Use_Plantation	Management_Land_Use
105	107	Castle Crag, Bampton, Cumbria	No		No
106	108	Castle Crag, Borrowdale, Cumbria	No		No
108	110	Castle Howe, Little Langdale, Cumbria	No		No
805	828	Baile Iaraghach Thuaidh (Ballyieragh North), Co..	No		No
841	864	Ballydivlin, Cork (Doonlea)	No		No

Review Management Data Split

```
In [ ]: review_data_split(management_data, management_numeric_data, management_text_data, management_encodeable_data)

Data split good.
```

Drop Management Features

The outcrop land use feature is dropped as it has no predictive value.

```
In [ ]: management_encodeable_data_plus = management_encodeable_data.copy()
management_encodeable_data_plus = \
management_encodeable_data_plus.drop(['Management_Land_Use_Outcrop'], axis=1)
management_encodeable_data_plus.head()
```

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed	Management_Land_Use_Woodland
0	Yes	No	No	Yes
1	Yes	No	No	No
2	Yes	No	No	Yes
3	Yes	No	No	Yes
4	Yes	No	No	No

Management Data Packages

```
In [ ]: management_data_list = \
[management_numeric_data, management_text_data, management_encodeable_data_plus]
```

Management Data Download Packages

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(management_data_list, 'Management_package')
```

Landscape Data

The Landscape features are subdivided into four subcategories:

- Type
- Topography
- Aspect
- Altitude

```
In [ ]: landscape_features = [
    'Landscape_Type_Contour',
    'Landscape_Type_Partial',
    'Landscape_Type_Promontory',
    'Landscape_Type_Hill_slope',
    'Landscape_Type_Level',
    'Landscape_Type_Marsh',
    'Landscape_Type_Multiple',
    'Landscape_Type_Comments',
    'Landscape_Topography_Hilltop',
    'Landscape_Topography_Coastal',
    'Landscape_Topography_Inland',
    'Landscape_Topography_Valley',
    'Landscape_Topography_Knoll',
    'Landscape_Topography_Ridge',
    'Landscape_Topography_Scarp',
    'Landscape_Topography_Hillslope',
    'Landscape_Topography_Lowland',
    'Landscape_Topography_Spur',
    'Landscape_Topography_Comments',
    'Landscape_Topography_Dominant',
    'Landscape_Aspect_N',
    'Landscape_Aspect_NE',
    'Landscape_Aspect_E',
    'Landscape_Aspect_SE',
    'Landscape_Aspect_S',
    'Landscape_Aspect_SW',
    'Landscape_Aspect_W',
    'Landscape_Aspect_NW',
    'Landscape_Aspect_Level',
    'Landscape_Altitude']
```

```
In [ ]: landscape_data = hillforts_data[landscape_features]
landscape_data.head()
```

Out[]:

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Landscape_Altitude
0	No	Yes	Yes	No	No	No
1	Yes	No	No	No	No	No
2	No	Yes	No	No	No	No
3	No	No	No	No	Yes	No
4	Yes	No	No	No	No	No

```
In [ ]: landscape_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 30 columns):
 #   Column           Non-Null Count Dtype  
 ---  -- 
 0   Landscape_Type_Condition 4147 non-null object  
 1   Landscape_Type_Partial     4147 non-null object  
 2   Landscape_Type_Promontory 4147 non-null object  
 3   Landscape_Type_Hill_slope 4147 non-null object  
 4   Landscape_Type_Level      4147 non-null object  
 5   Landscape_Type_Marsh      4147 non-null object  
 6   Landscape_Type_Multiple   4147 non-null object  
 7   Landscape_Type_Comments   2501 non-null object  
 8   Landscape_Topography_Hill_top 4147 non-null object  
 9   Landscape_Topography_Coastal 4147 non-null object  
 10  Landscape_Topography_Inland 4147 non-null object  
 11  Landscape_Topography_Valley 4147 non-null object  
 12  Landscape_Topography_Knoll 4147 non-null object  
 13  Landscape_Topography_Ridge 4147 non-null object  
 14  Landscape_Topography_Scarp 4147 non-null object  
 15  Landscape_Topography_Hill_slope 4147 non-null object  
 16  Landscape_Topography_Lowland 4147 non-null object  
 17  Landscape_Topography_Spur   4147 non-null object  
 18  Landscape_Topography_Comments 135 non-null object  
 19  Landscape_Topography_Dominant 2300 non-null object  
 20  Landscape_Aspect_N        4147 non-null object  
 21  Landscape_Aspect_NE       4147 non-null object  
 22  Landscape_Aspect_E        4147 non-null object  
 23  Landscape_Aspect_SE       4147 non-null object  
 24  Landscape_Aspect_S        4147 non-null object  
 25  Landscape_Aspect_SW       4147 non-null object  
 26  Landscape_Aspect_W        4147 non-null object  
 27  Landscape_Aspect_NW       4147 non-null object  
 28  Landscape_Aspect_Level    4147 non-null object  
 29  Landscape_Altitude        4115 non-null float64 
dtypes: float64(1), object(29)
memory usage: 972.1+ KB

```

Landscape Numeric Data

There is a single numeric feature in the Landscape data package.

```
In [ ]: landscape_numeric_features = [
    'Landscape_Altitude']

landscape_numeric_data = \
    hillsforts_data[landscape_numeric_features].copy()
landscape_numeric_data.head()
```

```
Out[ ]: Landscape_Altitude
0      276.0
1      150.0
2      225.0
3      150.0
4      338.0
```

```
In [ ]: landscape_numeric_data['Landscape_Altitude'].isna().sum()
```

```
Out[ ]: 32
```

There are 32 records containing no altitude information. The first 5 are listed. To see the full list, update the 5, in the square brackets below, to 32 and rerun the document as described in [User Settings](#).

```
In [ ]: nan_altitude_features = \
    name_and_number_features + location_numeric_data_short_features + \
    landscape_numeric_features
nan_altitude_data = hillsforts_data[nan_altitude_features]
nan_altitude_data[nan_altitude_data['Landscape_Altitude'].isna()[:5]]
```

Out[]:	Main_Atlas_Number	Main_Display_Name	Location_X	Location_Y	Landscape_Altitude	
	355	365	Harborough Hill, Churchill, Worcestershire	-240920	6874690	NaN
	446	464	Wadbury Camp, Somerset (Wadbury Hillfort)	-265052	6663609	NaN
	531	550	Norham Castle, Northumberland	-239382	7503047	NaN
	1376	1437	Gloster Camp, Flintshire	-369738	7037341	NaN
	1742	1830	Prae Wood, Hertfordshire (St. Albans; Verulamium)	-41506	6755099	NaN

Landscape Numeric Data - Resolve Null Values

Test to see if Landscape Altitude contains the value -1.

```
In [ ]: test_num_list_for_minus_one(landscape_numeric_data, \
['Landscape_Altitude'])
```

Landscape_Altitude 0

Update null values to -1.

```
In [ ]: landscape_numeric_data = \
update_num_list_for_minus_one(landscape_numeric_data, ['Landscape_Altitude'])
```

```
In [ ]: landscape_numeric_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 1 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Landscape_Altitude  4147 non-null   float64
dtypes: float64(1)
memory usage: 32.5 KB
```

Altitude Data

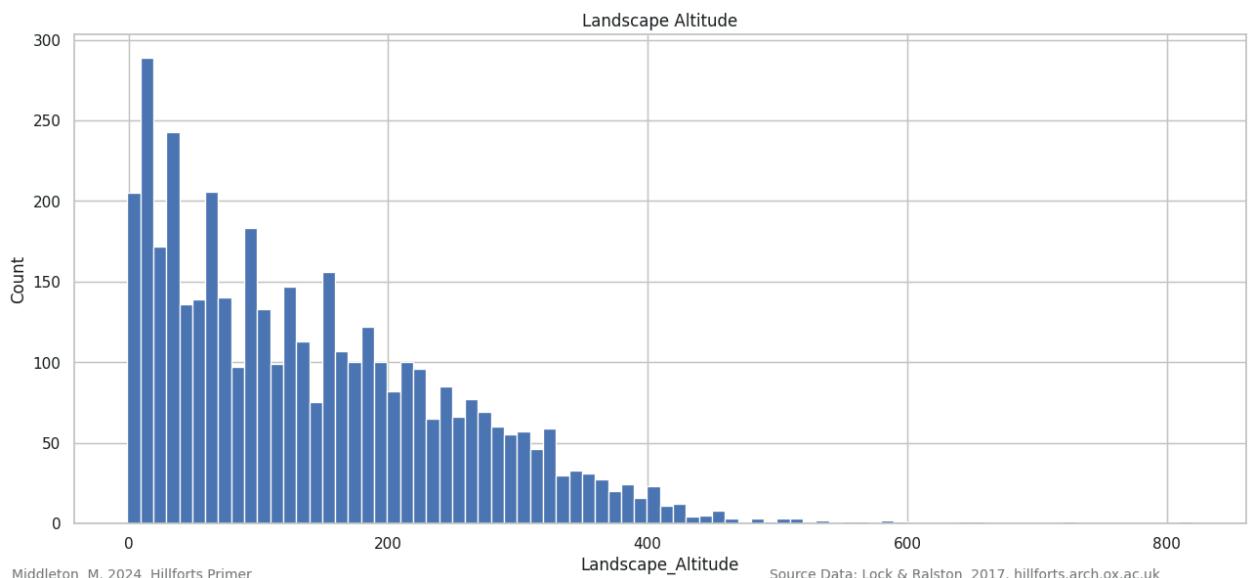
Most hillforts are located below 460m although there are outliers up to 820m. The data below 250m is grouped around peaks in the data at periodic intervals. This suggests the recording of altitude, in some cases, is not precise but, has often been generalised to the nearest 10, 25, 50 or 100m.

Altitude Data Plotted

```
In [ ]: landscape_numeric_data['Landscape_Altitude'].describe()
```

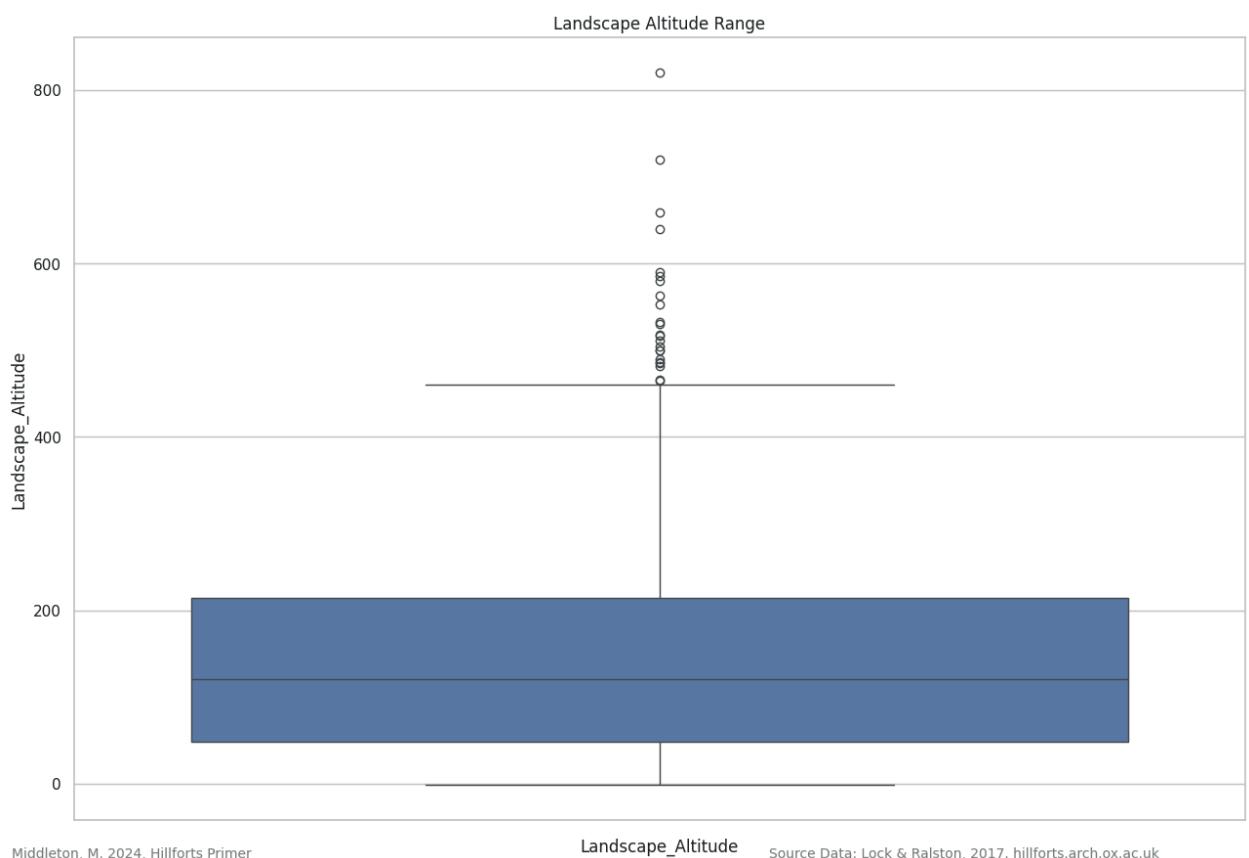
```
Out[ ]: count    4147.000000
mean     141.504558
std      111.052905
min     -1.000000
25%      48.500000
50%      120.000000
75%      214.000000
max      820.000000
Name: Landscape_Altitude, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(landscape_numeric_data, 1, \
'Landscape_Altitude', 'Count', 'Landscape_Altitude', 82)
```



95.6% of hillforts are located below 460m. The majority (the first three percentiles - 75% of the data) are located toward the lower end of the range, below 214m; 50% are below 120m and 25% are below 48.5m.

```
In [ ]: Landscape_Altitude_stats = \
plot_data_range(landscape_numeric_data['Landscape_Altitude'], \
'Landscape_Altitude')
```



```
In [ ]: Landscape_Altitude_stats
```

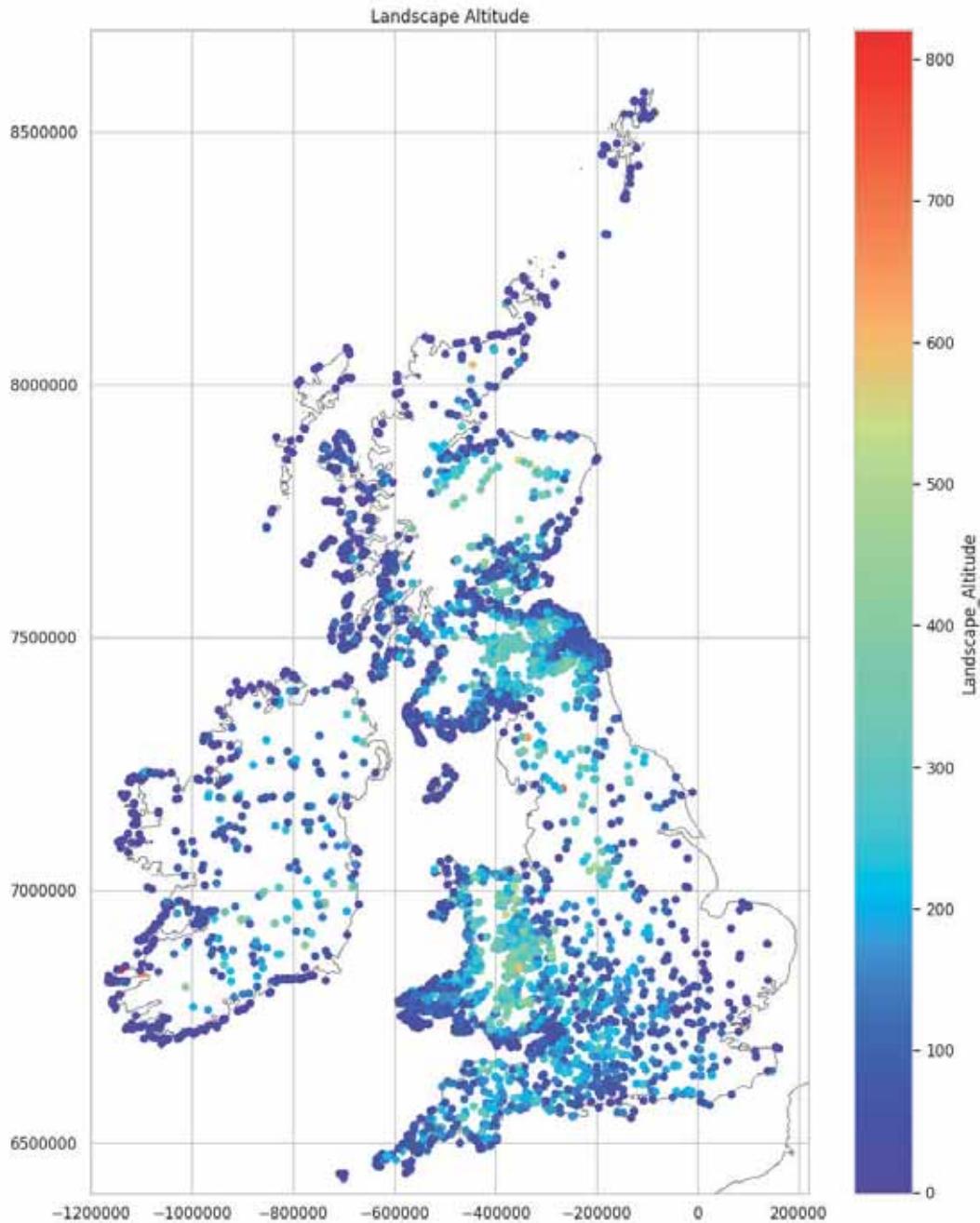
```
Out[ ]: [-1.0, 48.5, 120.0, 214.0, 460.0]
```

Altitude Mapped

With so many of the hillforts having an altitude toward the lower end of the range, the outliers cause the map of hillforts by altitude to be not particularly revealing.

```
In [ ]: location_Landscape_numeric_data = \
pd.merge(location_numeric_data_short, landscape_numeric_data, left_index=True, \
right_index=True)
```

```
In [ ]: plot_type_values(location_landscape_numeric_data, 'Landscape_Alitude', \
'Landscape_Alitude')
```



Middleton, M. 2024. Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

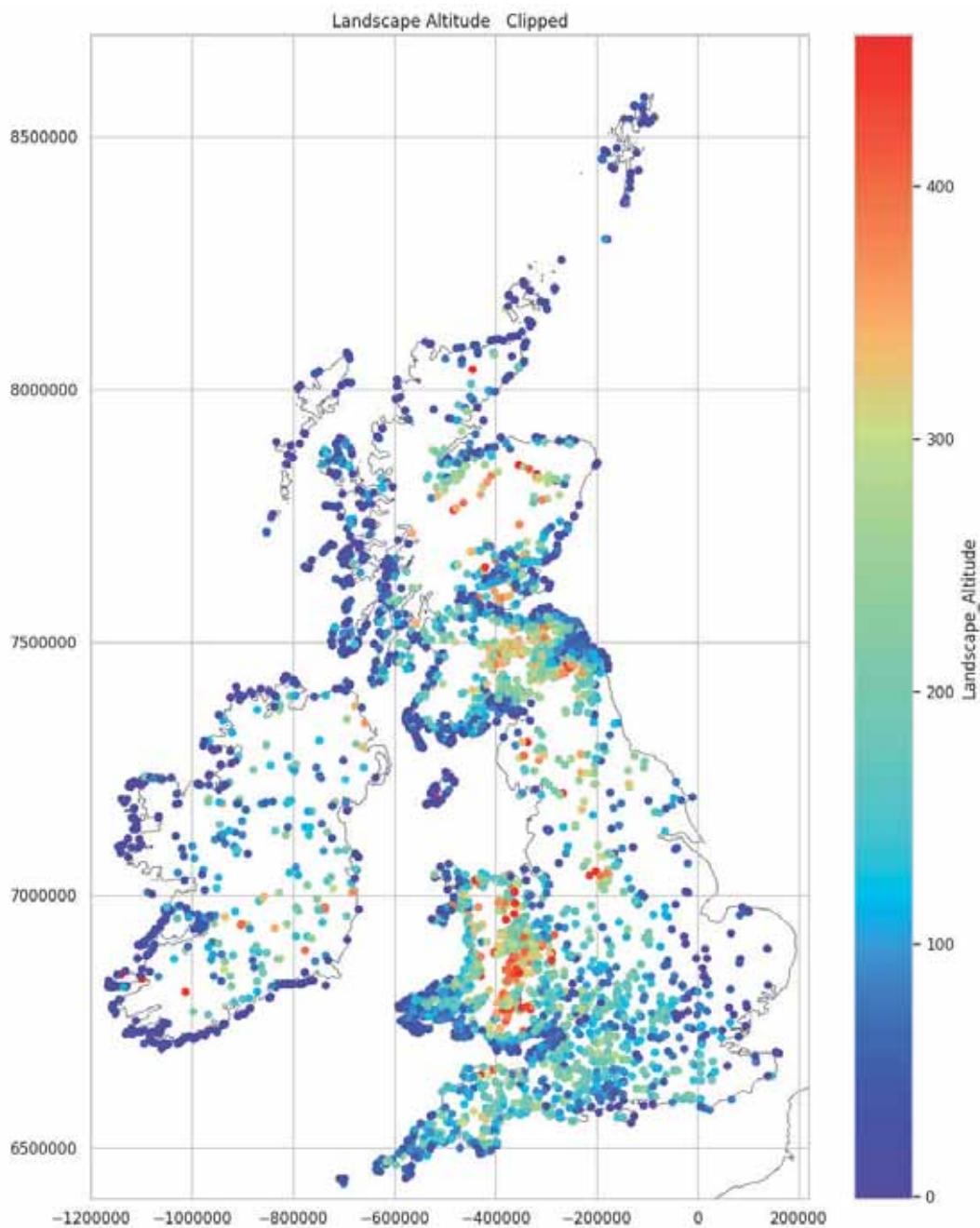
Altitude Capped Mapped

To improve the clarity of the distribution plot, the altitude values are capped at the top end of the fourth quartile, to 460m.

```
In [ ]: location_landscape_numeric_data_clip = location_landscape_numeric_data.copy()
location_landscape_numeric_data_clip['Landscape_Alitude'].\
where(location_landscape_numeric_data_clip['Landscape_Alitude'] < \
Landscape_Alitude_stats[4], Landscape_Alitude_stats[4], inplace=True)
location_landscape_numeric_data_clip['Landscape_Alitude'].describe()
```

```
Out[ ]: count    4147.000000
mean     140.993827
std      109.176613
min     -1.000000
25%      48.500000
50%     120.000000
75%     214.000000
max      460.000000
Name: Landscape_Alitude, dtype: float64
```

```
In [ ]: plot_type_values(location_landscape_numeric_data_clip, 'Landscape_Alitude', 'Landscape_Alitude - Clipped')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Altitude Over 460m Mapped (Outliers)

There are 23 hillforts (0.55%) over 460m. The highest is Faha in Kerry, Ireland at 820m. The ten highest are listed below.

```
In [ ]: over_460 = \
    location_Landscape_numeric_data[location_Landscape_numeric_data['Landscape_Altitude'] > 460].copy()
len(over_460)
```

```
Out[ ]: 23
```

To see the full list, update the 10, at the end of the second line below, to 23 and rerun the document as described in [User Settings](#).
(eg ... =False).head(23))

```
In [ ]: over_460_plus = \
    pd.merge(name_and_number, over_460, left_index=True, right_index=True)
over_460_plus[['Main_Display_Name', 'Landscape_Altitude']].\
    sort_values(by='Landscape_Altitude', ascending=False).head(10)
```

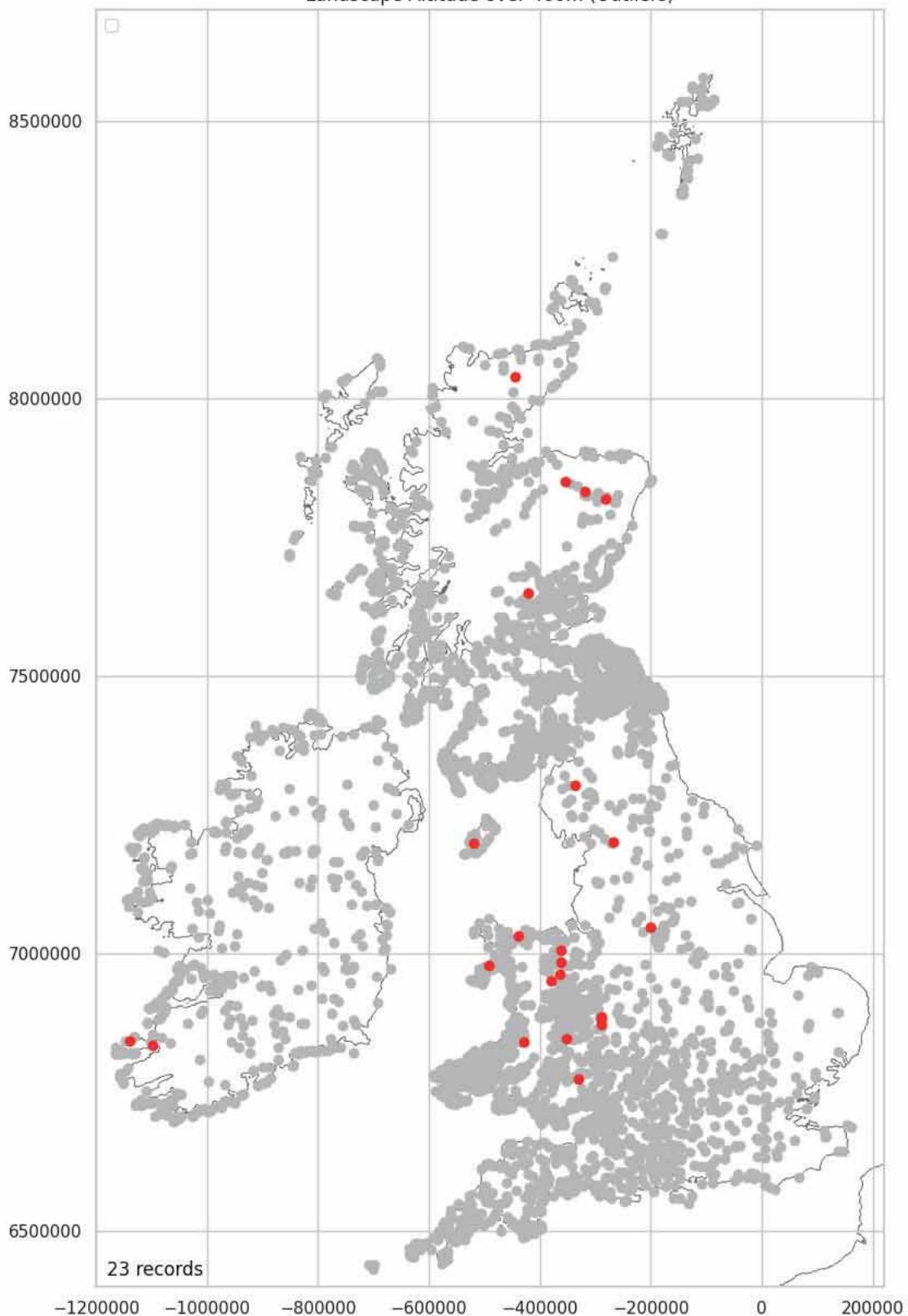
Out[]:

	Main_Display_Name	Landscape_Altitude
647	Faha, Kerry (Begagh; Binn na Port; An Fhaiche)	820.0
2388	Ingleborough, North Yorkshire	720.0
642	Caherconree, Kerry (Ballyarkane Oughter, Behee...)	659.0
104	Carrock Fell, Cumbria	640.0
2607	The Whimble, Powys	590.0
3035	South Barrule, Rushen	586.0
2619	Ben GRIAM Beg, Highland	580.0
2767	Tap o' Noth, Aberdeenshire (Hill of Noth)	563.0
2762	Little Conval, Moray	553.0
1355	Craig Rhiwarth, Powys	533.0

In []:

```
over_460['Landscape_Altitude'] = "Yes"
over_460_stats = \
plot_over_grey(over_460, 'Landscape_Altitude', 'Yes', 'over 460m (Outliers)')
```

Landscape Altitude over 460m (Outliers)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

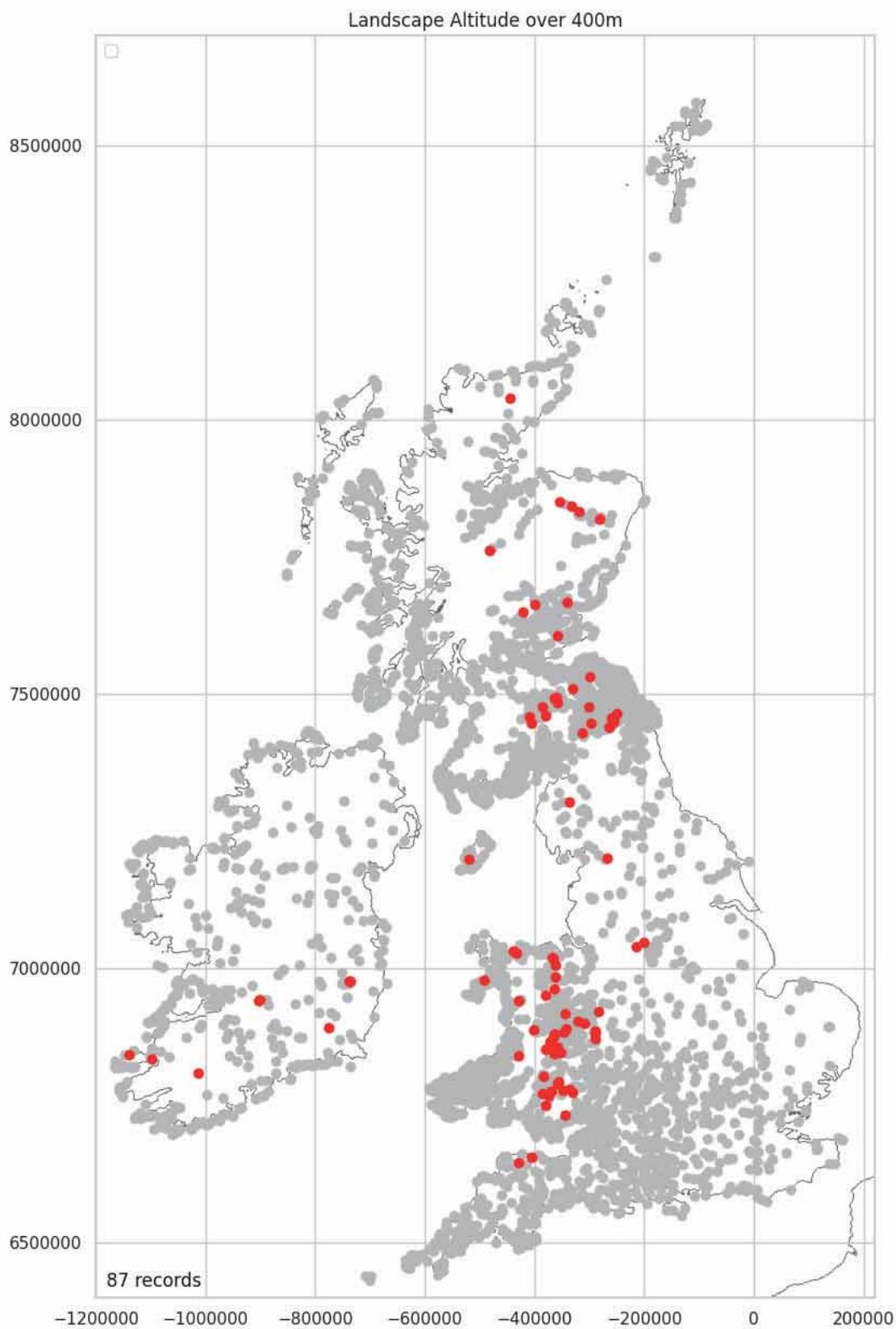
0.55%

Altitude Over 400m Mapped

87 hillforts (2.1%) are located above 400m. There is a notable pairing of hillforts (by proximity) across the central region of the southern uplands.

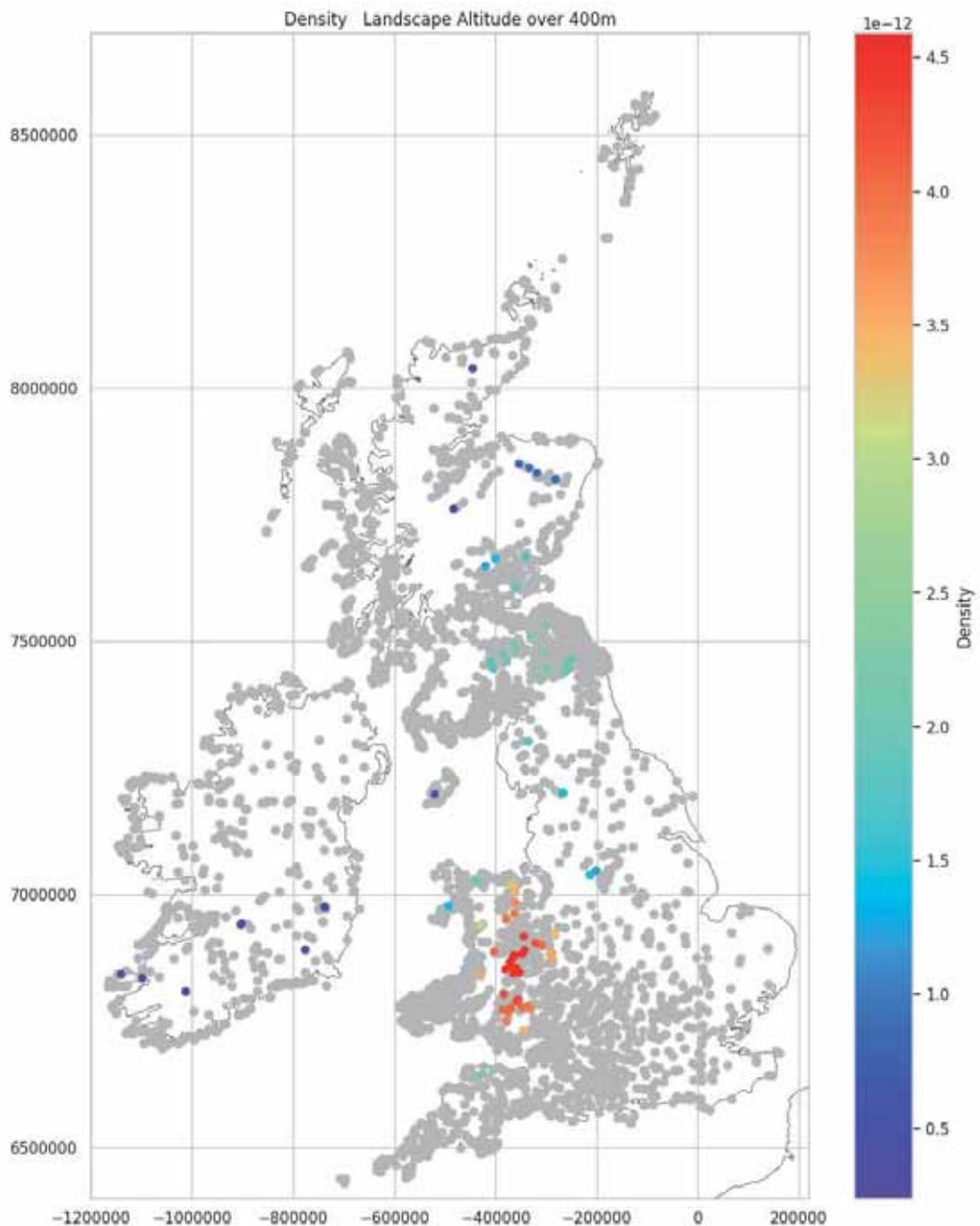
```
In [ ]: over_400 = \
    location_landscape_numeric_data_clip\ 
    [location_landscape_numeric_data_clip['Landscape_Altitude'] >= 400].copy()
over_400['Landscape_Altitude'] = "Yes"
```

```
In [ ]: over_400_stats = plot_over_grey(over_400, 'Landscape_Altitude', 'Yes', 'over 400m')
```



Most hillforts above 400m are in the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(over_400_stats, 'Landscape_Altitude_over_400m')
```



Middleton, M. 2024, Hillforts Primer

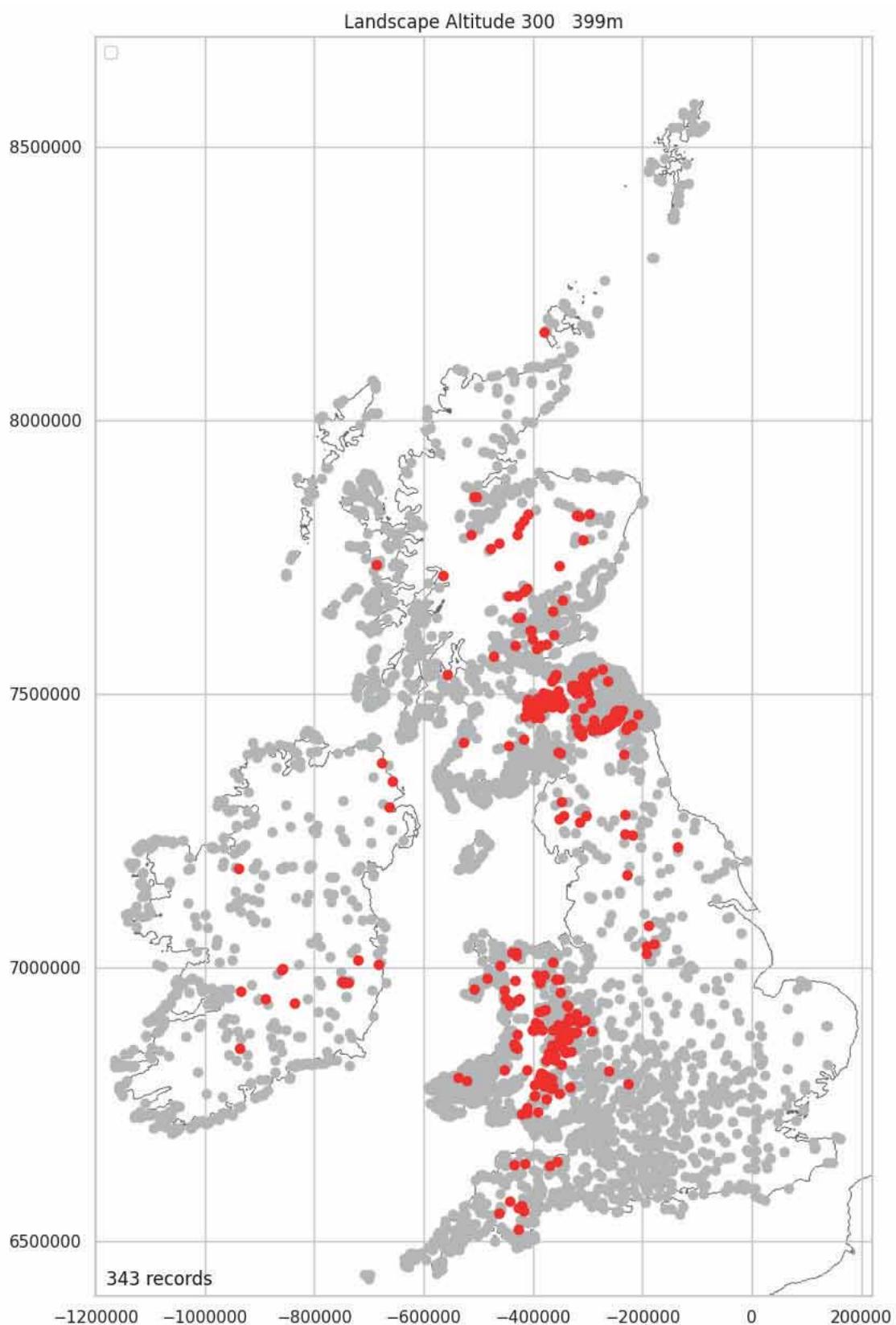
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Altitude 300 - 399m Mapped

There are two dominant groups of hillforts in the 300 to 399m data; over the Southern Uplands and along the Cambrian Mountains. There are a number of noticeable alignments along ridges in the southern cluster and the ring of hills around the Tweed basin is clearly distinguishable in the Northeast.

```
In [ ]: over_300 = \
    location_Landscape_numeric_data_clip[
        (location_Landscape_numeric_data_clip['Landscape_Altitude'] >= 300) & \
        (location_Landscape_numeric_data_clip['Landscape_Altitude'] < 400)].copy()
    over_300['Landscape_Altitude'] = "Yes"
```

```
In [ ]: over_300_stats = plot_over_grey(over_300, 'Landscape_Altitude', 'Yes', '300 - 399m')
```



Middleton, M. 2024, Hillforts Primer

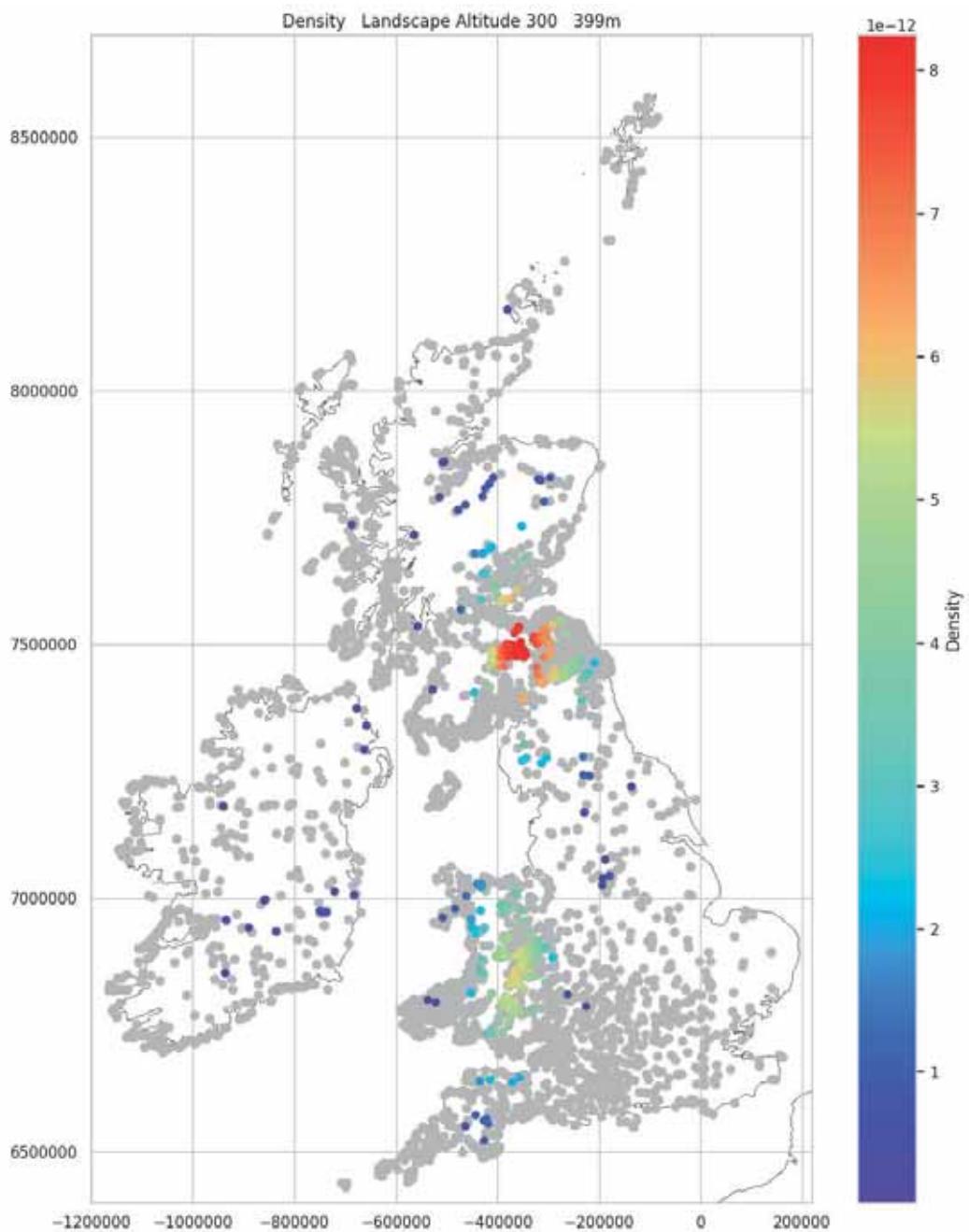
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

8.27%

Altitude Over 300 - 399m Density Mapped

Most hillforts between 300 and 399m cluster over the Moorfoot and Tweedsmuir Hills. There is a secondary cluster along the eastern flank of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(over_300_stats, 'Landscape_Altitude_300 - 399m')
```



Middleton, M. 2024, Hillforts Primer

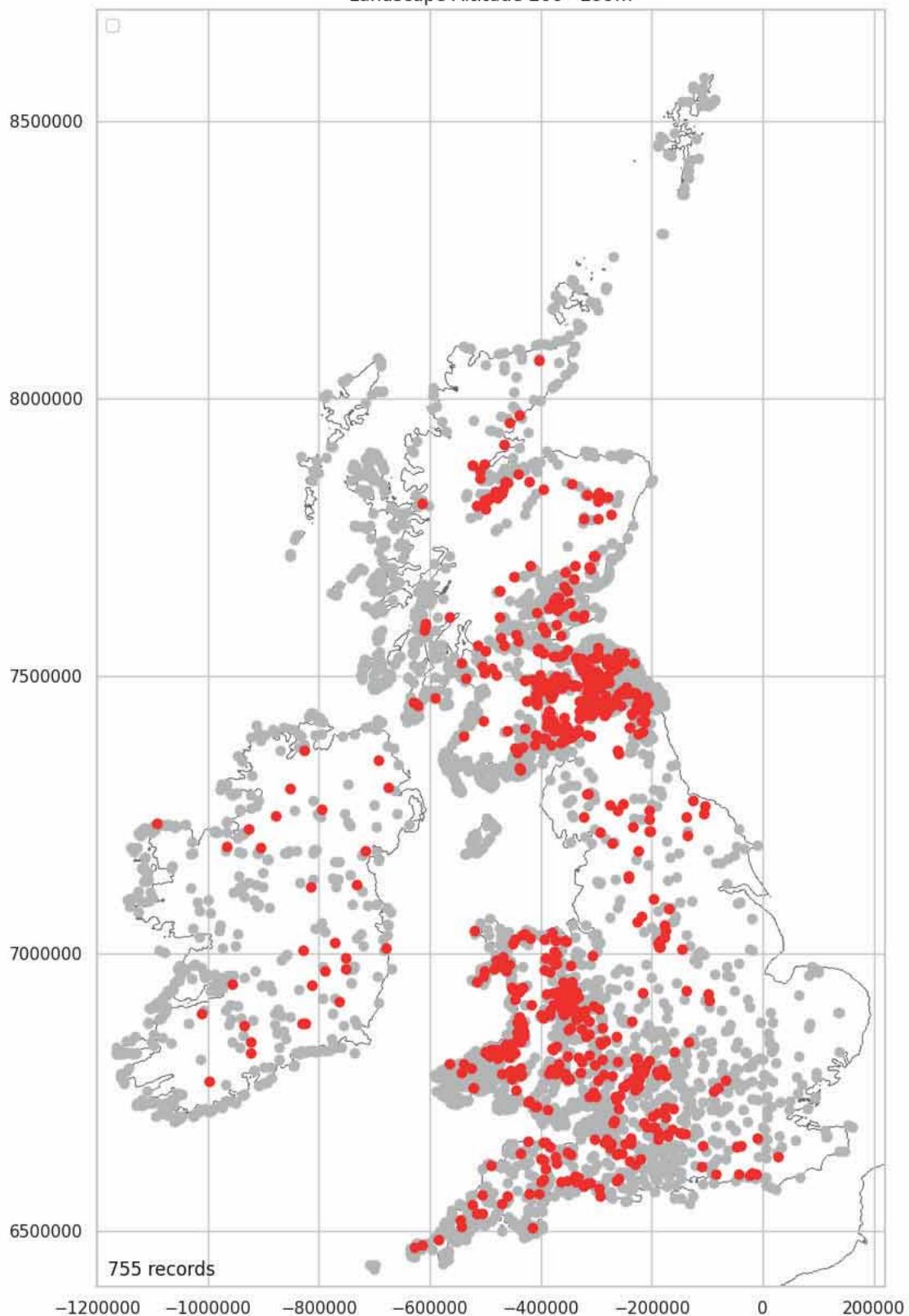
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Altitude 200 - 299m Mapped

Most hillforts between 200 and 299m are located in the Southern Uplands and the Cambrian Mountains. There are a good number of forts across the south and southeast of England as well as along the edges of the highlands, along the Highland Boundary fault, and up the east coast, to the north of the Grampian Mountains and into the Great Glen. There are a peppering of forts along the Pennines and across central and eastern Ireland.

```
In [ ]: over_200 = \
    location_Landscape_numeric_data_clip[
        (location_Landscape_numeric_data_clip['Landscape_Altitude'] >= 200) & \
        (location_Landscape_numeric_data_clip['Landscape_Altitude'] < 300)].copy()
over_200['Landscape_Altitude'] = "Yes"
```

```
In [ ]: over_200_stats = plot_over_grey(over_200, 'Landscape_Altitude', 'Yes', '200 - 299m')
```



Middleton, M. 2024, Hillforts Primer

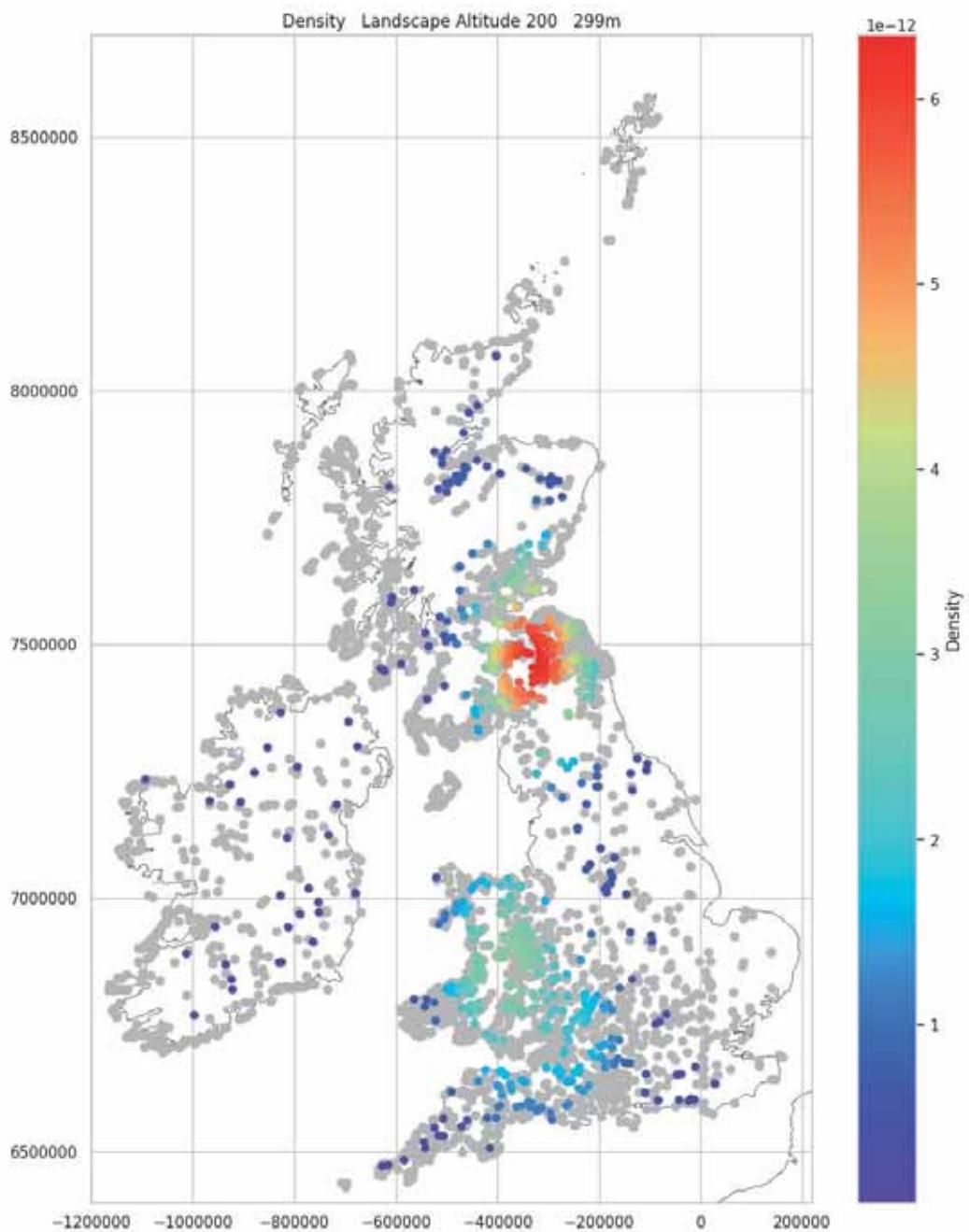
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

18. 21%

Altitude Over 200 - 299m Density Mapped

There is a main cluster in the Southern Uplands with a second cluster across the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(over_200_stats, 'Landscape_Altitude_200 - 299m')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

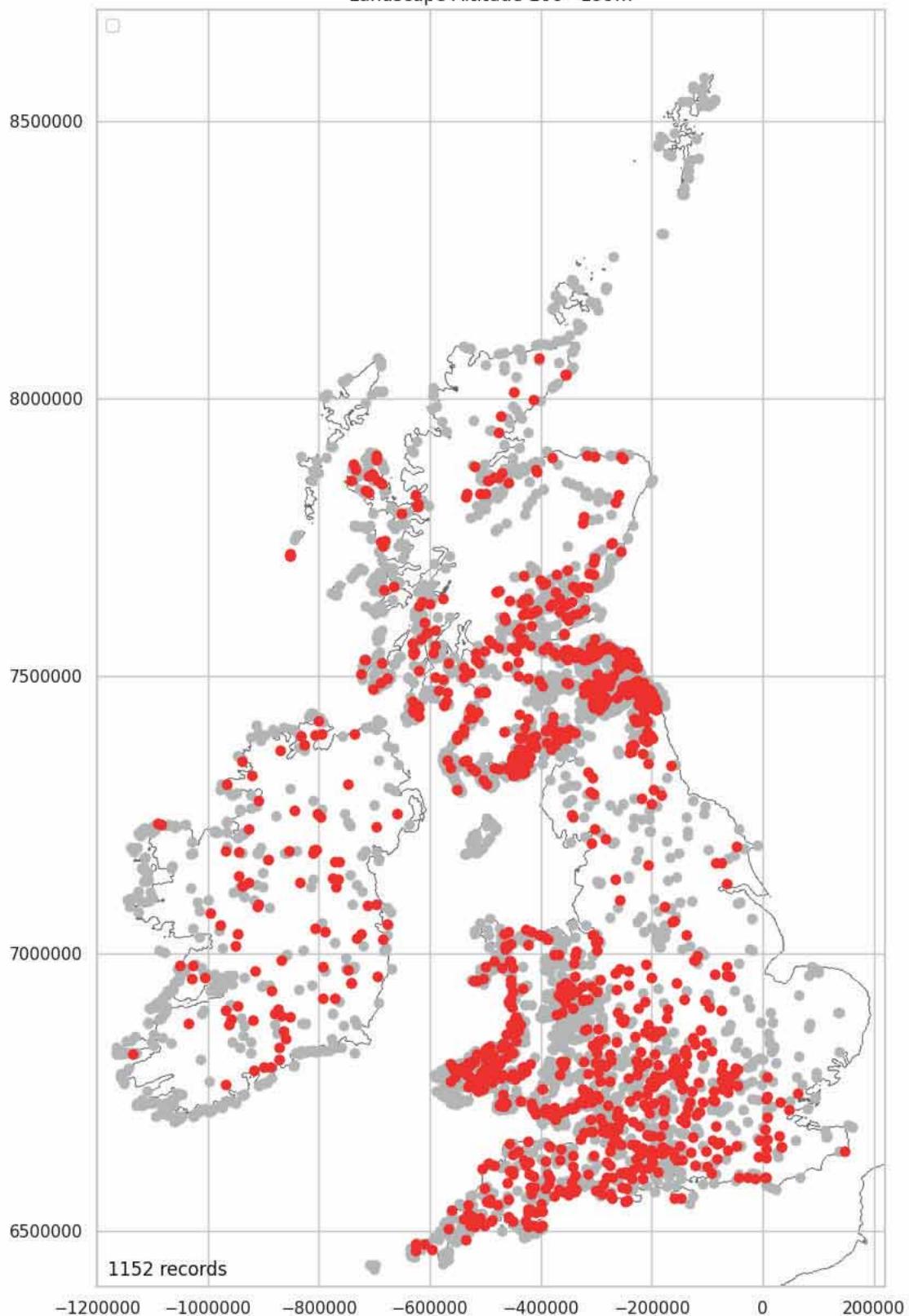
Altitude 100 - 199m Mapped

At lower altitudes we start to see hillforts spreading out over the South and Southeast, Pembrokeshire, the Northeast, the Highland Boundary Fault, either end of the Northwest hillforts density cluster, around the coast of the Moray Firth and peppered right across Ireland.

```
In [ ]: over_100 = \
    location_Landscape_numeric_data_clip[
        (location_Landscape_numeric_data_clip['Landscape_Altitude'] >= \
        100) & (location_Landscape_numeric_data_clip['Landscape_Altitude'] < \
        200)].copy()
over_100['Landscape_Altitude'] = "Yes"
```

```
In [ ]: over_100_stats = plot_over_grey(over_100, 'Landscape_Altitude', 'Yes', \
    '100 - 199m')
```

Landscape Altitude 100 199m



Middleton, M. 2024, Hillforts Primer

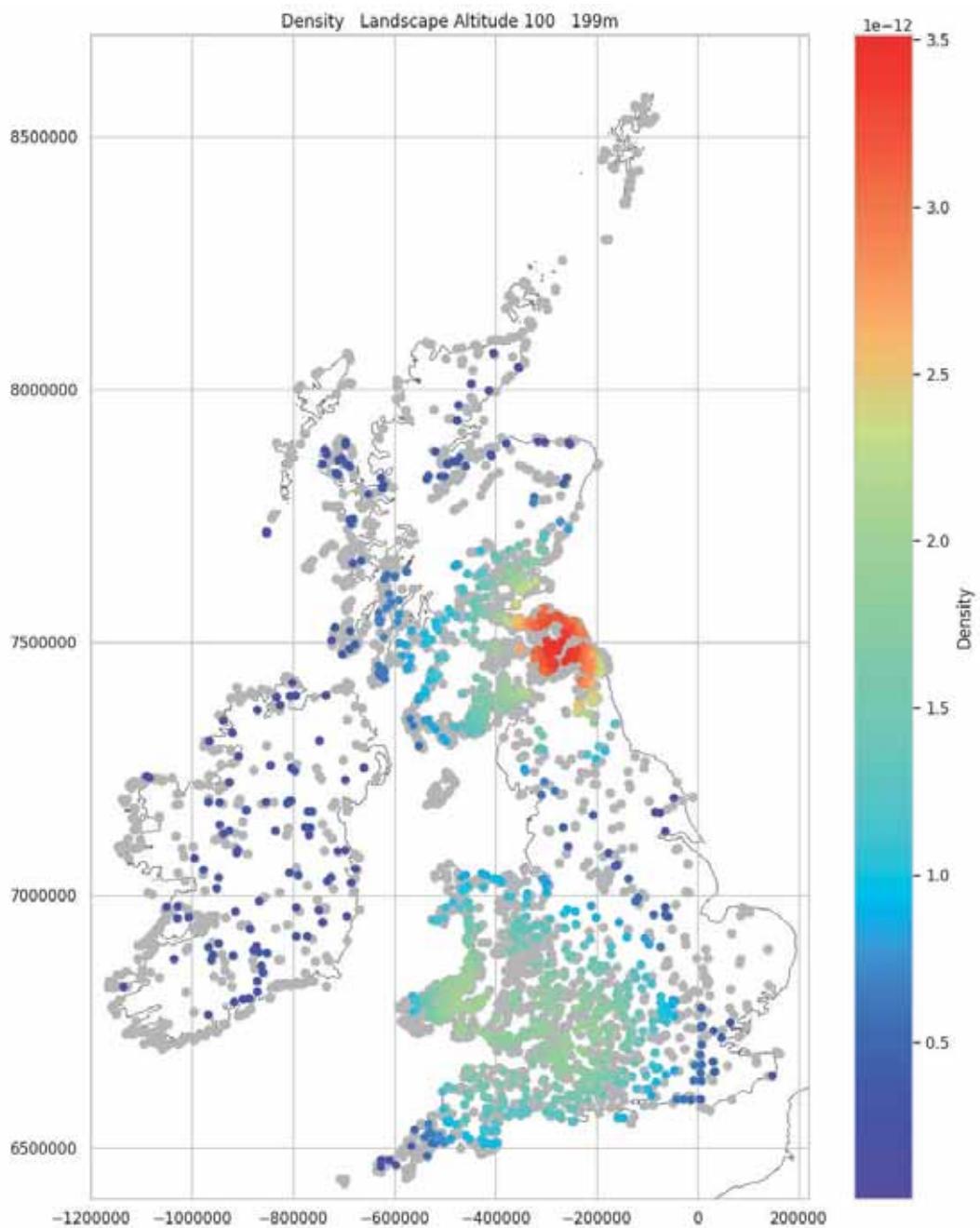
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

27.78%

Altitude Over 100 - 199m Density Mapped

The most intense cluster continues to be in the Northeast, spreading right across southern Scotland. A second cluster can be seen running from south Wales into south, central England.

```
In [ ]: plot_density_over_grey(over_100_stats, 'Landscape_Altitude_100 - 199m')
```



Middleton, M. 2024, Hillforts Primer

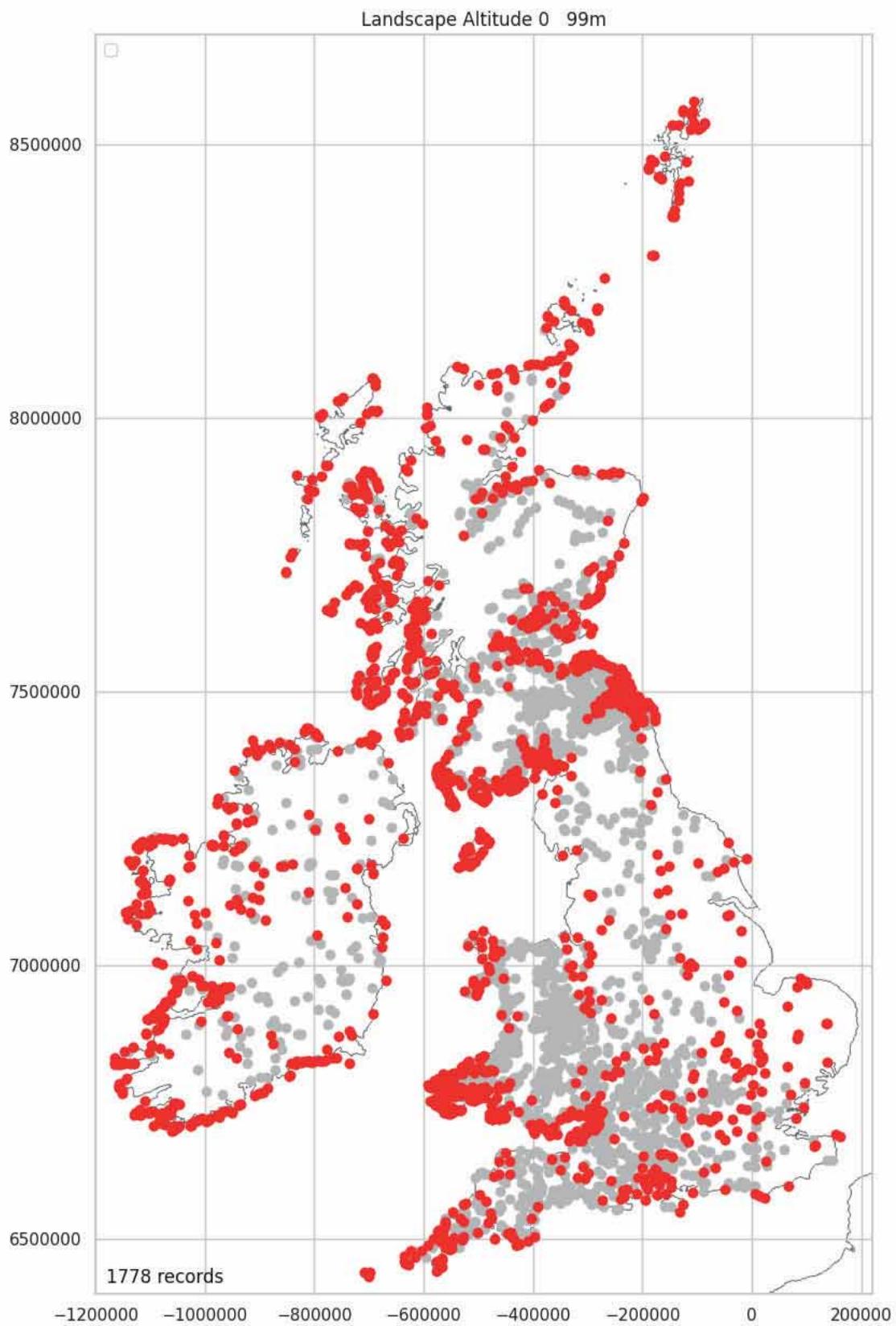
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Altitude 0 - 99m Mapped

The distribution of forts below 100m is noticeably more coastal. The majority of hillforts (42.87%) are located in this altitude range.

```
In [ ]: over_0 = \
    location_Landscape_numeric_data_clip[('Landscape_Altitude' >= 0) & \
    ('Landscape_numeric_data_clip['Landscape_Altitude' < 100]).copy()
over_0['Landscape_Altitude'] = "Yes"
```

```
In [ ]: over_0_stats = plot_over_grey(over_0, 'Landscape_Altitude', 'Yes', '0 - 99m')
```



Middleton, M. 2024, Hillforts Primer

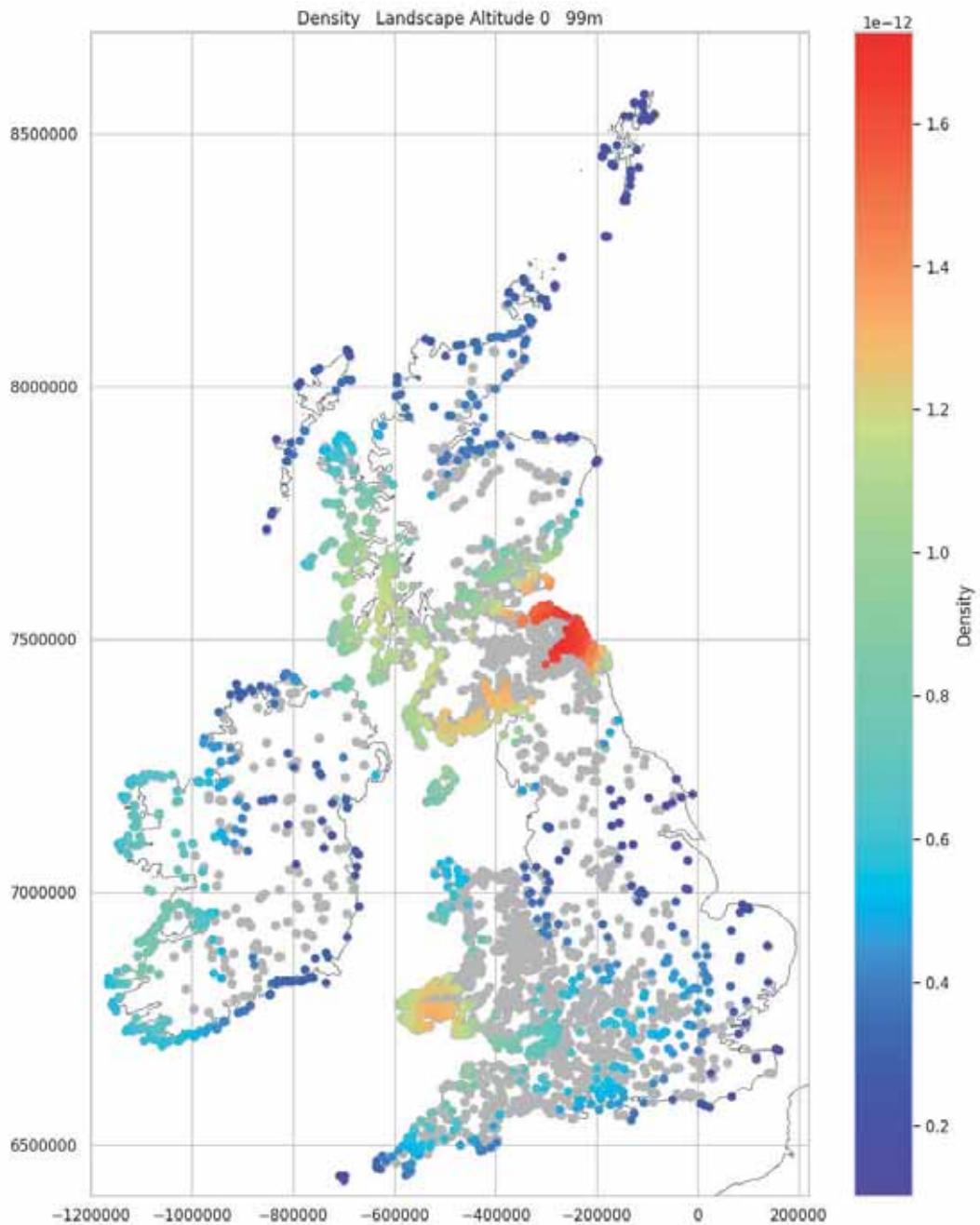
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

42.87%

Altitude Over 0 - 99m Density Mapped

The main cluster continues to be in the Southern Uplands, within the Tweed Basin, while Pembrokeshire, the Galloway coast, the Northwest and the west of Ireland all show as smaller clusters. All five main distribution clusters seen in, Part 1: Density Map Showing Clusters Adjusted by Region, can be seen in this altitude range.

```
In [ ]: plot_densitiy_over_grey(over_0_stats, 'Landscape_Altitude_0 - 99m')
```



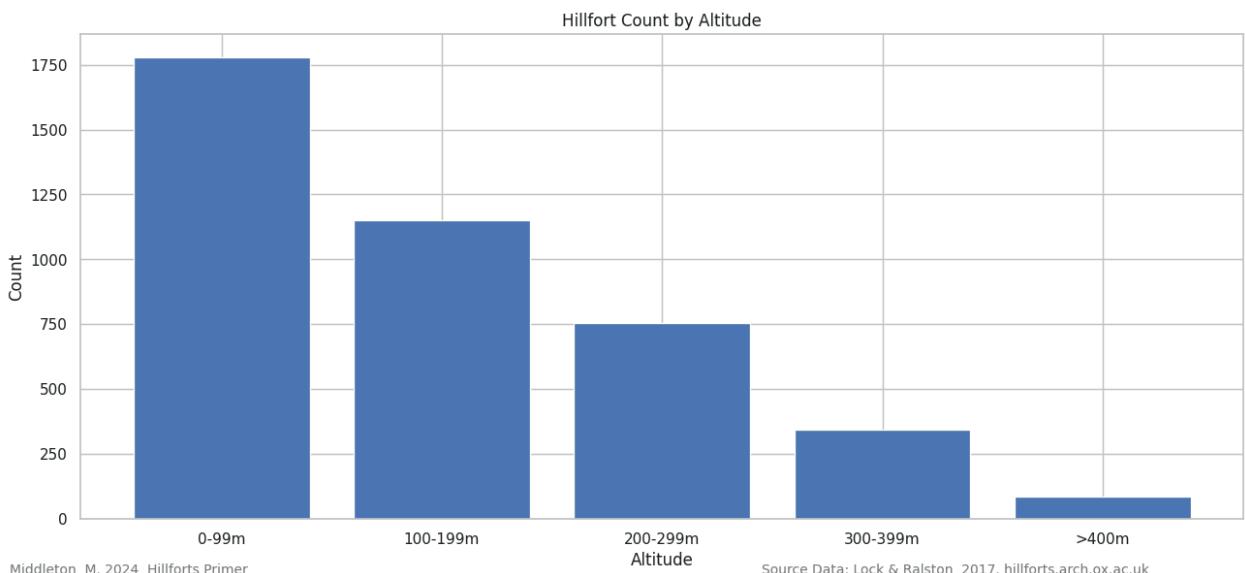
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Bracketed Altitude Plotted

42.87% of hillforts are below 100m. Within this bracket, most are in close to the coast. As we climb in altitude, hillforts become less common. 94.86% of hillforts are below 300m.

```
In [ ]: plot_bar_chart_from_list([over_0, over_100, over_200, over_300, over_400], \
["0-99m", "100-199m", "200-299m", "300-399m", ">400m"], "Altitude", \
"Count", "Hillfort Count by Altitude")
```



Landscape Text Data

There are three Landscape text features.

```
In [ ]: landscape_text_features = [
    'Landscape_Type_Comments',
    'Landscape_Topography_Comments',
    'Landscape_Topography_Dominant']

landscape_text_data = hillforts_data[landscape_text_features]
landscape_text_data.head()
```

```
Out[ ]:
```

	Landscape_Type_Comments	Landscape_Topography_Comments	Landscape_Topography_Dominant
0	Partial contour fort following the natural con...	NaN	Hill top, part promontory.
1	Univallate, contour hillfort located on summit...	NaN	Hill top spur.
2	Located part on slopes, part level ground. Sit...	NaN	Hilltop including the Adam's Rocks outcrop
3	The site is located on NW slopes just below th...	NaN	Flat-topped steep hill.
4	One of the finest examples of a contour fort i...	NaN	Malvern Hill crest.

Landscape Text Data - Resolve Null Values

Test for the presence of 'NA' in Landscape Text Data features.

```
In [ ]: test_cat_list_for_NA(landscape_text_data, landscape_text_features)
```

```
Landscape_Type_Comments 0
Landscape_Topography_Comments 0
Landscape_Topography_Dominant 0
```

Fill null values with 'NA'.

```
In [ ]: landscape_text_data = \
update_cat_list_for_NA(landscape_text_data, landscape_text_features)
landscape_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Landscape_Type_Comments  4147 non-null  object 
 1   Landscape_Topography_Comments 4147 non-null  object 
 2   Landscape_Topography_Dominant 4147 non-null  object 
dtypes: object(3)
memory usage: 97.3+ KB
```

Landscape Encodeable Data

There are 26 Landscape Encodeable Features grouped into three subcategories:

- Type

- Topography
- Aspect

```
In [ ]: landscape_encodeable_features = [
    'Landscape_Type_Contour',
    'Landscape_Type_Partial',
    'Landscape_Type_Promontory',
    'Landscape_Type_Hill_slope',
    'Landscape_Type_Level',
    'Landscape_Type_Marsh',
    'Landscape_Type_Multiple',
    'Landscape_Topo_Hill_top',
    'Landscape_Topo_Coastal',
    'Landscape_Topo_Inland',
    'Landscape_Topo_Valley',
    'Landscape_Topo_Knoll',
    'Landscape_Topo_Ridge',
    'Landscape_Topo_Scarp',
    'Landscape_Topo_Hill_slope',
    'Landscape_Topo_Lowland',
    'Landscape_Topo_Spur',
    'Landscape_Aspect_N',
    'Landscape_Aspect_NE',
    'Landscape_Aspect_E',
    'Landscape_Aspect_SE',
    'Landscape_Aspect_S',
    'Landscape_Aspect_SW',
    'Landscape_Aspect_W',
    'Landscape_Aspect_NW',
    'Landscape_Aspect_Level']
```

```
In [ ]: landscape_encodeable_data = hillforts_data[landscape_encodeable_features]
landscape_encodeable_data.head()
```

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Landsca
0	No	Yes		Yes	No	No
1	Yes	No		No	No	No
2	No	Yes		No	No	No
3	No	No		No	Yes	No
4	Yes	No		No	No	No

Landscape Type Data

There are seven landscape types. Partial is short for 'Partial Contour' and Multiple refers to multiple enclosures which enclose an area that can support occupation. See: [Data Structure](#).

```
In [ ]: landscape_type_features = [
    'Landscape_Type_Contour',
    'Landscape_Type_Partial',
    'Landscape_Type_Promontory',
    'Landscape_Type_Hill_slope',
    'Landscape_Type_Level',
    'Landscape_Type_Marsh',
    'Landscape_Type_Mul_tiple']
```

```
In [ ]: landscape_type_data = landscape_encodeable_data[landscape_type_features].copy()
landscape_type_data.head()
```

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Landsca
0	No	Yes		Yes	No	No
1	Yes	No		No	No	No
2	No	Yes		No	No	No
3	No	No		No	Yes	No
4	Yes	No		No	No	No

There are no null values in the Landscape Type data.

```
In [ ]: landscape_type_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   Landscape_Type_Contour    4147 non-null   object  
 1   Landscape_Type_Partial    4147 non-null   object  
 2   Landscape_Type_Promontory 4147 non-null   object  
 3   Landscape_Type_Hill_slope 4147 non-null   object  
 4   Landscape_Type_Level     4147 non-null   object  
 5   Landscape_Type_Marsh     4147 non-null   object  
 6   Landscape_Type_Multiple  4147 non-null   object  
dtypes: object(7)
memory usage: 226.9+ KB

```

A hillfort can be multiple landscape types.

```
In [ ]: landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='Yes') & (landscape_type_data['Landscape_Type_Promontory']=='Yes')].head(3)
```

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Lan
488	Yes	No	Yes	No	Yes	
2051	Yes	No	Yes	No	No	
2278	Yes	No	Yes	No	No	

There are 50 hillforts where a landscape type has not been recorded.

```
In [ ]: no_type = \
landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='No') \
&(landscape_type_data['Landscape_Type_Partial']=='No') \
&(landscape_type_data['Landscape_Type_Promontory']=='No') \
&(landscape_type_data['Landscape_Type_Hill_slope']=='No') \
&(landscape_type_data['Landscape_Type_Level']=='No') \
&(landscape_type_data['Landscape_Type_Marsh']=='No') \
&(landscape_type_data['Landscape_Type_Multiple']=='No')]

print(f'Landscape Type not recorded: {len(no_type)}.' )
```

Landscape Type not recorded: 50.

Five records are shown. To see the full list, update the 5, in the square brackets below, to 50 and rerun the document as described in [User Settings](#).

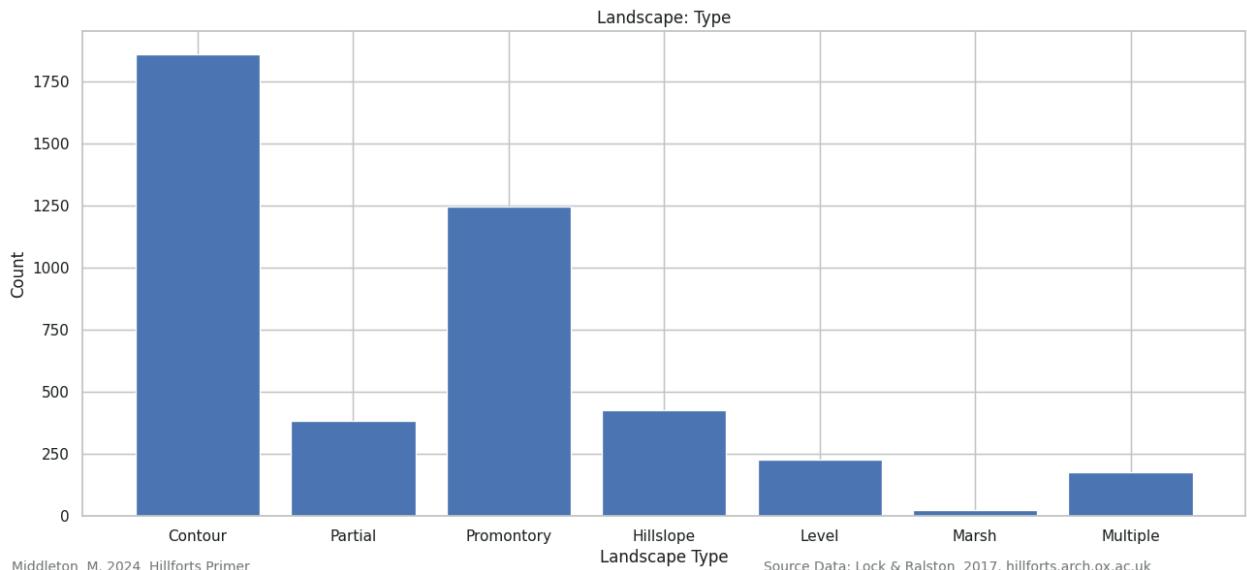
```
In [ ]: landscape_data_no_type = \
pd.merge(name_and_number, no_type, left_index=True, right_index=True)
landscape_data_no_type[['Main_Atlas_Number', 'Main_Display_Name']][5:]
```

	Main_Atlas_Number	Main_Display_Name
178	185	Dunbae Glen, Dumfries & Galloway
275	282	Kennan's Isle, Tongland Loch, Dumfries & Gallo...
531	550	Norham Castle, Northumberland
770	792	Carleton Hill, South Ayrshire
845	868	Broadgate, Dumfries & Galloway

Landscape Type Data Short Plotted

Contour hillforts (44.85%) and Partial Contour hillforts (9.26%) make up 54.11% of all forts. Promontory forts make up the next 30.09%. Hillslope forts account for 10.3%; Level forts 5.45% and Marsh forts make up just 0.55% of hillforts. 177 forts (4.27%) have been classified as multiple types. Hillforts can be more than one landscape type. For this reason, the total count is more than the total number of hillforts in the atlas.

```
In [ ]: plot_bar_chart(landscape_type_data, 2, 'Landscape_Type', 'Count', \
'Landscape_Type')
```



```
In [ ]: count_yes(landscape_type_data)
```

```
Landscape_Type_Contour: 1860
Landscape_Type_Partial : 384
Landscape_Type_Promontory: 1248
Landscape_Type_Hillslope: 427
Landscape_Type_Level: 226
Landscape_Type_Marsh: 23
Landscape_Type_Multiple: 177
Total yes count: 4345
```

There are two hillforts identified as both contour and partial contour.

```
In [ ]: contour_type = \
    landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='Yes') \
    &(landscape_type_data['Landscape_Type_Partial']=='Yes') \
    &(landscape_type_data['Landscape_Type_Promontory']=='No') \
    &(landscape_type_data['Landscape_Type_Hillslope']=='No') \
    &(landscape_type_data['Landscape_Type_Level']=='No') \
    &(landscape_type_data['Landscape_Type_Marsh']=='No') \
    &(landscape_type_data['Landscape_Type_Multiple']=='No')]
print(f'Landscape Type yes for both Contour and Partial Contour: {len(contour_type)}')
```

```
Landscape Type yes for both Contour and Partial Contour: 2
```

There are 13 hillforts classified as both contour and promontory.

```
In [ ]: contour_prom_type = \
    landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='Yes') \
    &(landscape_type_data['Landscape_Type_Partial']=='No') \
    &(landscape_type_data['Landscape_Type_Promontory']=='Yes') \
    &(landscape_type_data['Landscape_Type_Hillslope']=='No') \
    &(landscape_type_data['Landscape_Type_Level']=='No') \
    &(landscape_type_data['Landscape_Type_Marsh']=='No') \
    &(landscape_type_data['Landscape_Type_Multiple']=='No')]
print(f'Landscape Type yes for both Contour and Promontory: {len(contour_prom_type)}')
```

```
Landscape Type yes for both Contour and Promontory: 13
```

Landscape Type Data Mapped

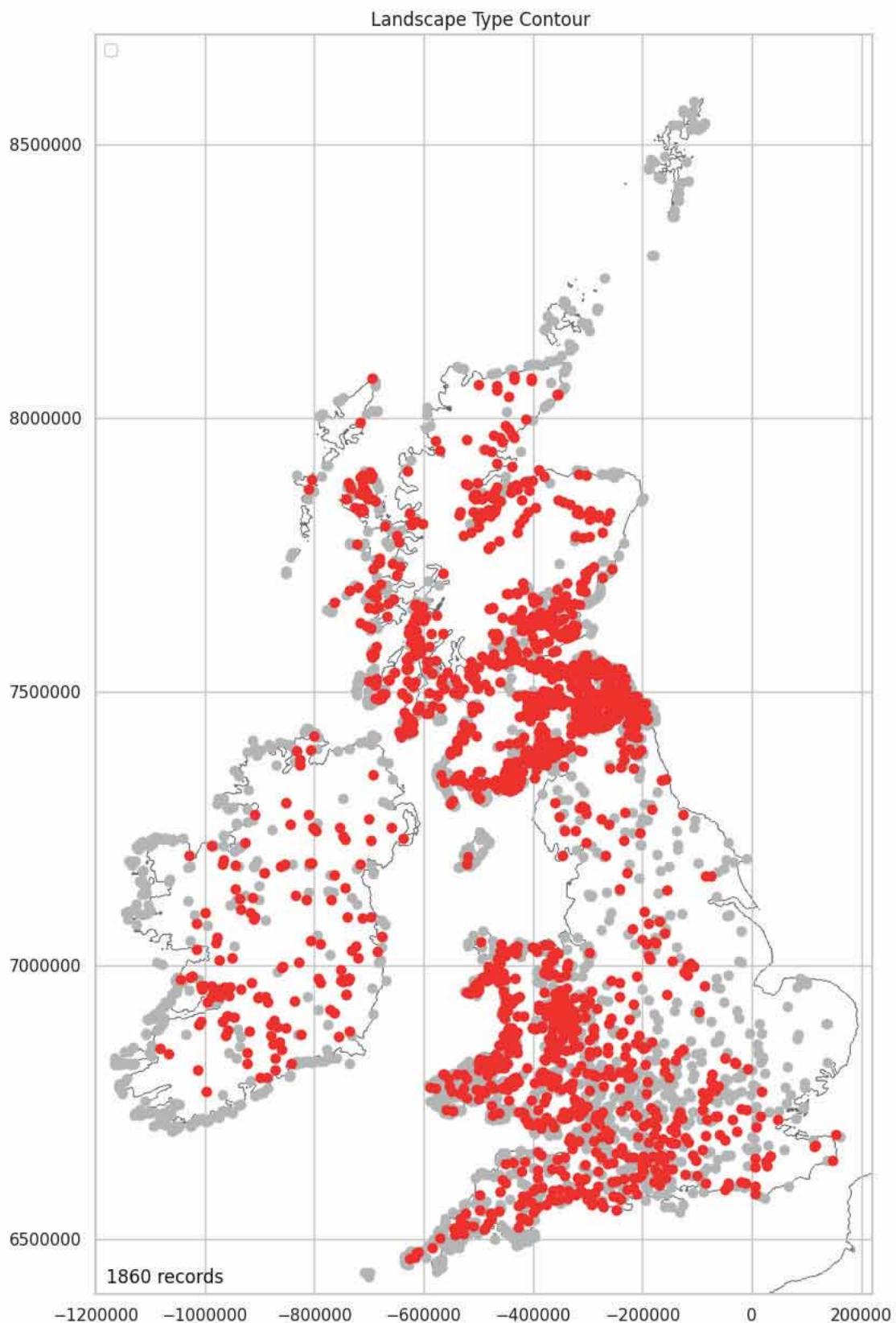
The Landscape Type data is recombined with the Location data so it can be mapped.

```
In [ ]: location_landscape_data = pd.merge(location_numeric_data_short, \
    landscape_type_data, left_index=True, \
    right_index=True)
```

Contour mapped

Contour forts are common across all of mainland Scotland, mainland Ireland, Wales and the South West of England. They are noticeably absent from the areas without significant hills, such as most coastlines, the northern Isles and the east of England.

```
In [ ]: tp_contour = plot_over_grey(location_landscape_data, \
    'Landscape_Type_Contour', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

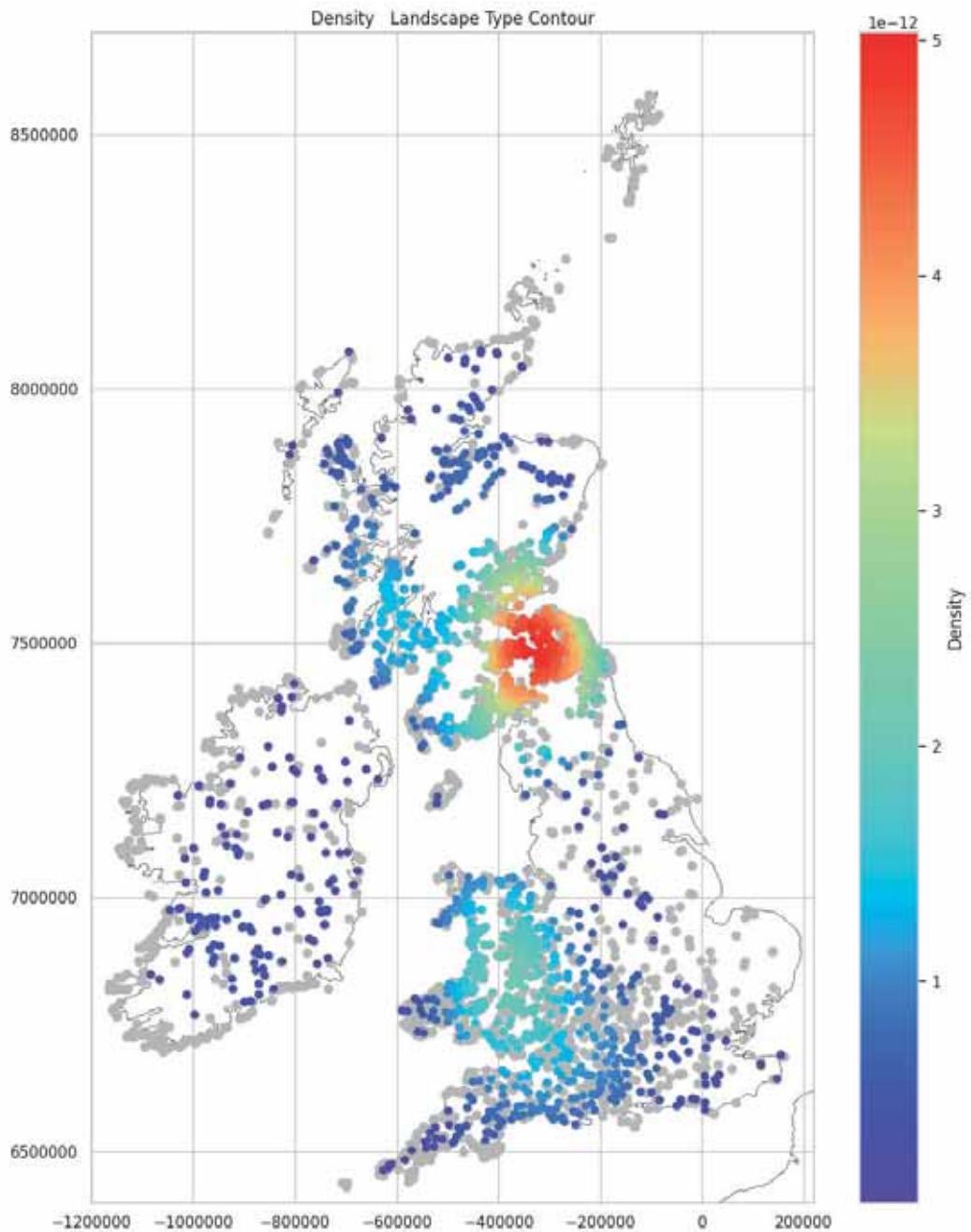
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

44.85%

Contour Density mapped

The density of contour hillforts matches the southern Cambrian Mountains and eastern Southern Uplands density clusters seen in Part1 (Density Map Showing Clusters Adjusted by Region). In contrast, the clusters seen in the Irish data are not replicated.

```
In [ ]: plot_density_over_grey(tp_contour, 'Landscape_Type_Contour')
```



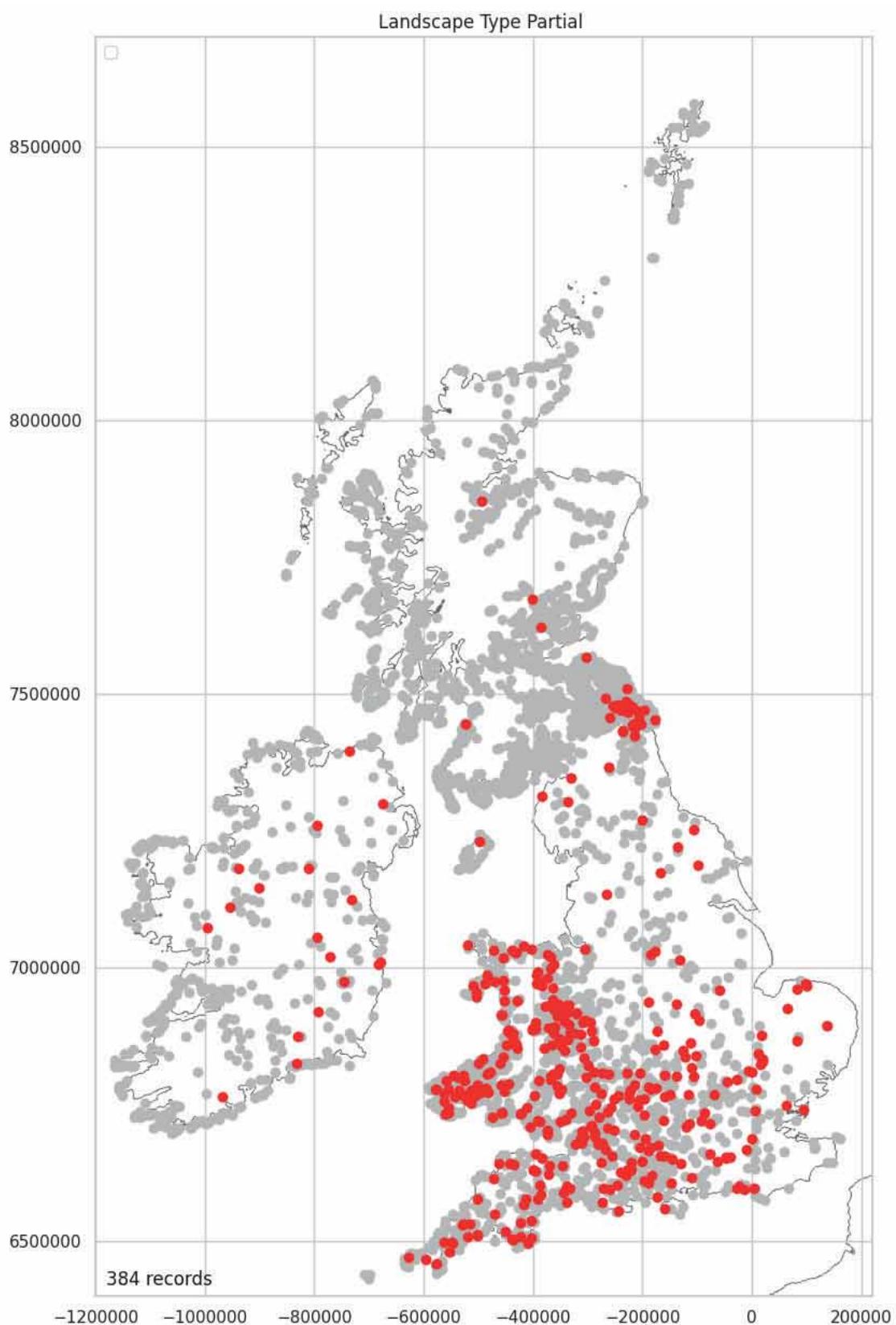
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Partial Mapped

There is a bias in that partial contour hillforts is a classification type use mostly in England and Wales. There are a couple of hillforts of this type recorded in Scotland and Ireland, but very few. Combining, 'Landscape_Type_Contour' and 'Landscape_Type_Partial' may minimise this bias.

```
In [ ]: tp_partial = plot_over_grey(location_landscape_data, \
'Landscape_Type_Partial', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

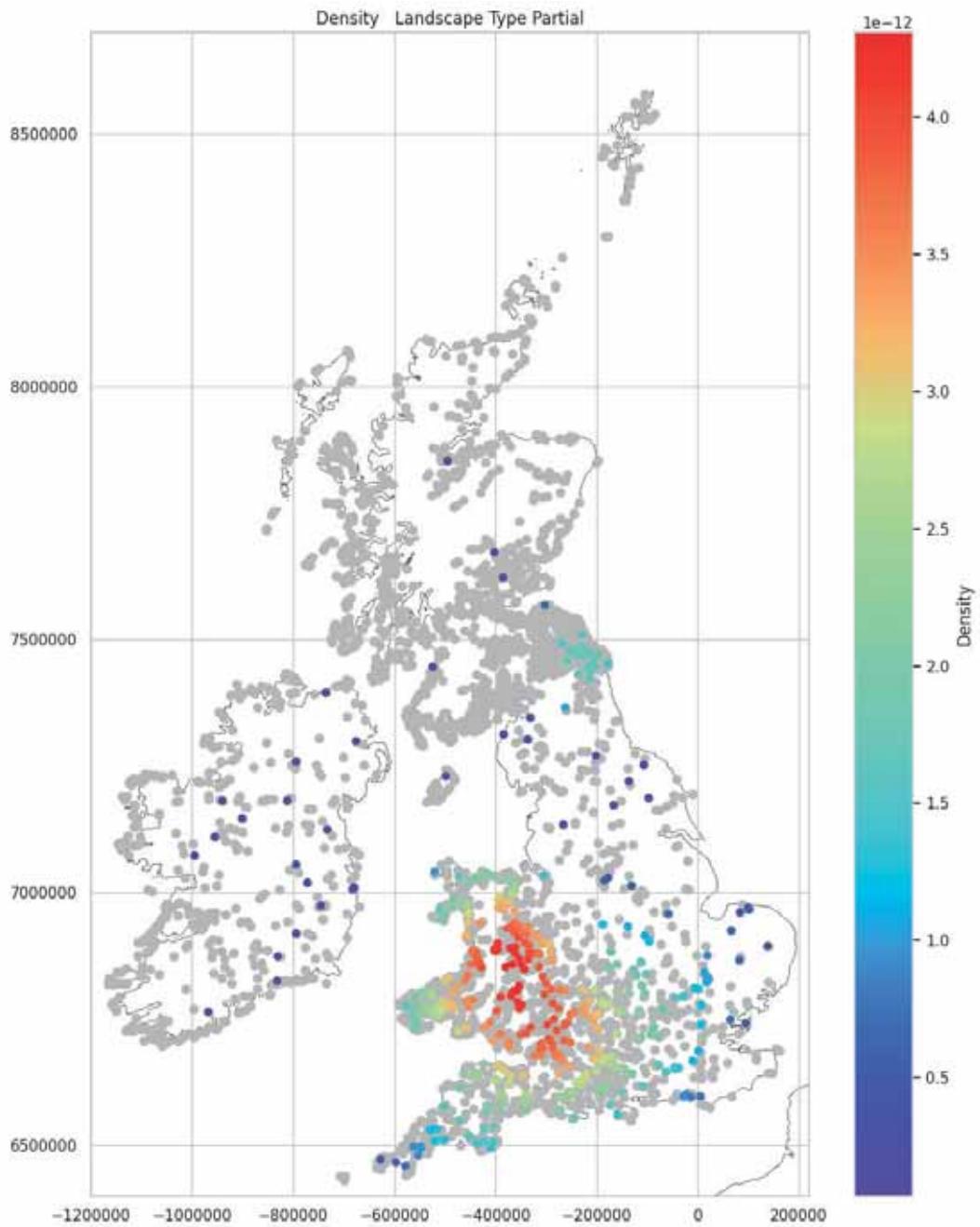
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

9.26%

Partial Density Mapped

Due to the bias in this data the density plot shows a single strong cluster in the south, over the Cambrian Mountains and into south central England. It is important to be aware of the bias in this data and to compare this density plot with, 'Landscape_Type_Contour', which shows a more meaningful, atlas wide distribution.

```
In [ ]: plot_density_over_grey(tp_partial, 'Landscape_Type_Partial')
```



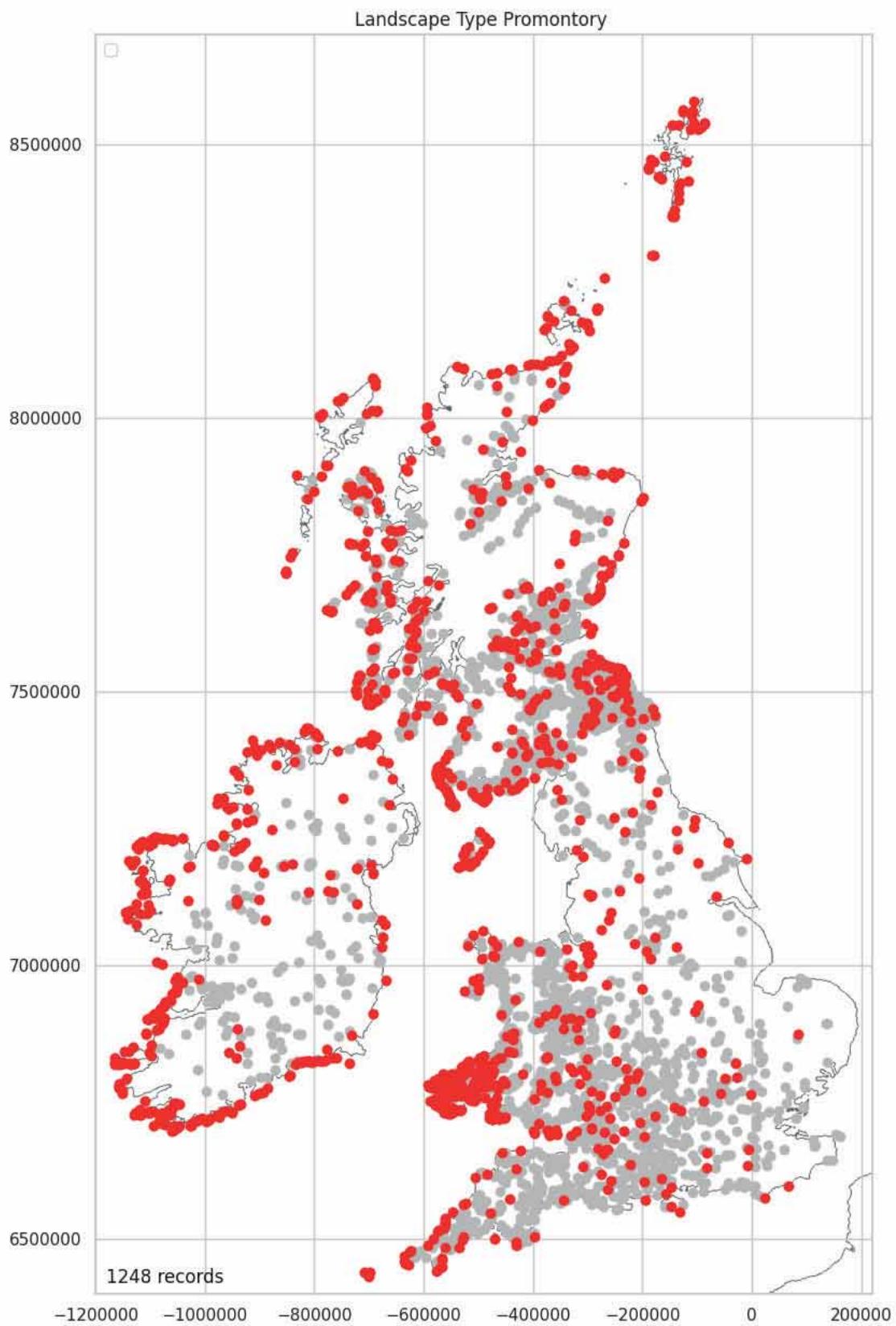
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Promontory Mapped

Promontory forts have a strong correlation to the coast, although promontory forts do also present inland.

```
In [ ]: tp_promontory = plot_over_grey(location_landscape_data, \
'Landscape_Type_Promontory', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

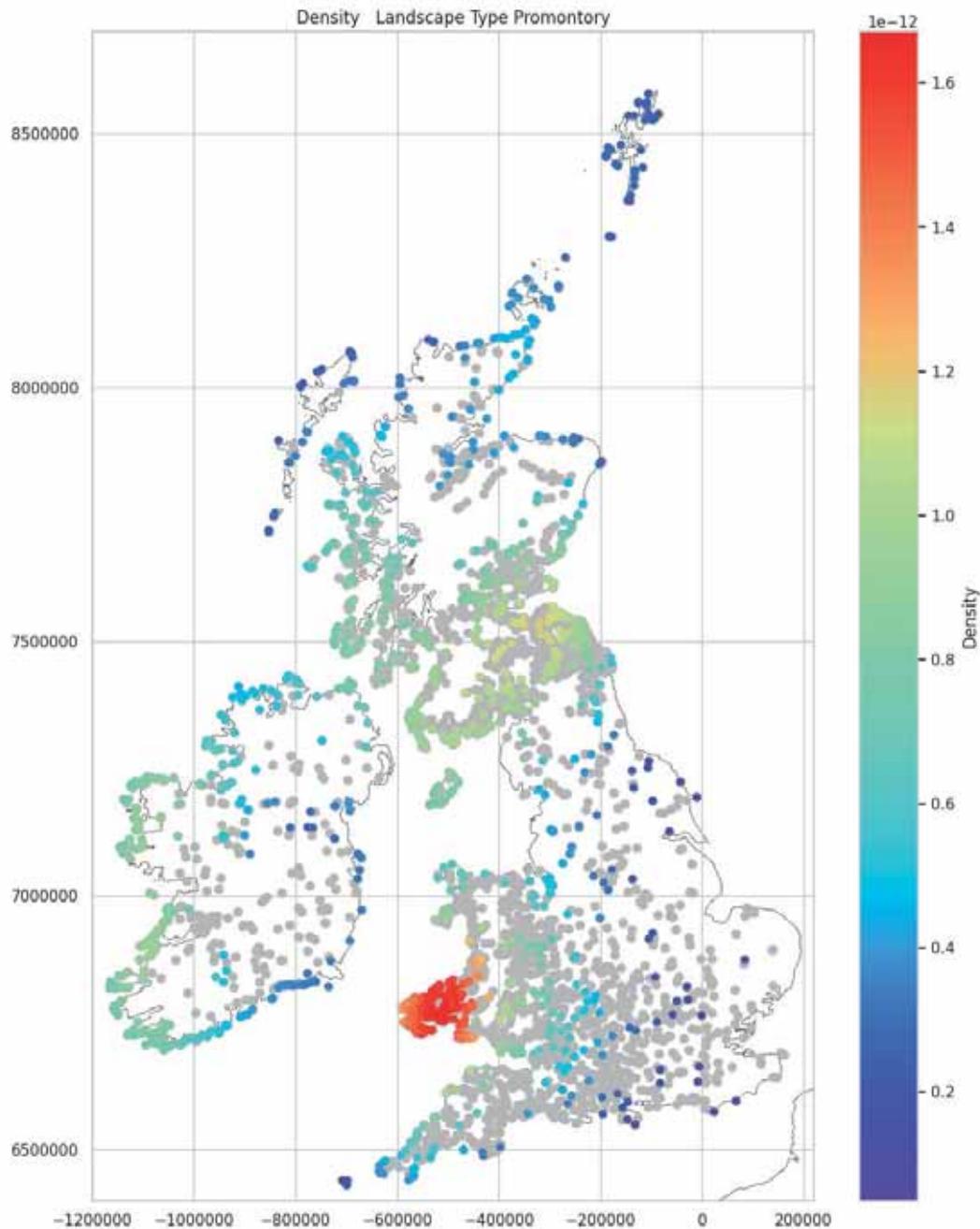
30.09%

Promontory Density Mapped

The highest concentration of promontory forts is along the Pembrokeshire peninsula. The clusters seen in the general density plots over the Irish west coast, also reveal themselves to be promontory forts. Similarly, there are concentrations of promontory forts along the west coast of Scotland and to the east of the Southern Uplands. See: [Part1](#) (Density Map Showing Clusters Adjusted by Region).

Ref: 3. Coastal Promontory Forts — Cliff Castles <https://intarch.ac.uk/journal/issue48/5/1.html#3>

```
In [ ]: plot_density_over_grey(tp_promontory, 'Landscape_Type_Promontory')
```



Middleton, M. 2024, Hillforts Primer

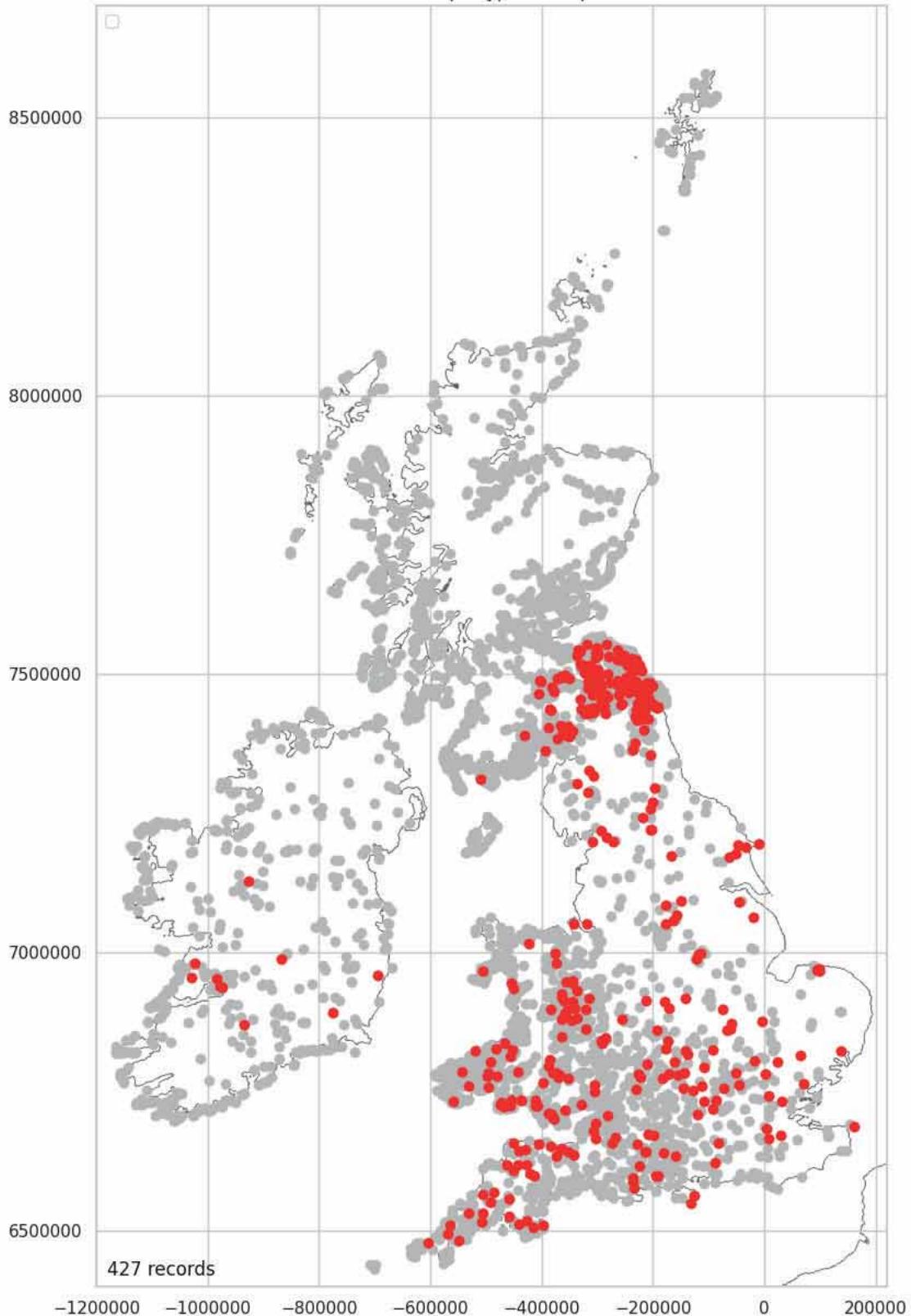
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Hillslope Mapped (Landscape)

It looks like there is a bias in this data. There are no hillslope hillforts in Scotland outside the Southern Uplands and there are very few in Ireland. This is either a recording bias across southern Scotland or a remarkably concentrated distribution of this type. This seems unlikely. A recording bias seems the most likely.

```
In [ ]: tp_hillslope = plot_over_grey(location_landscape_data, \
'Landscape_Type_Hillslope', 'Yes')
```

Landscape Type Hillslope



Middleton, M. 2024, Hillforts Primer

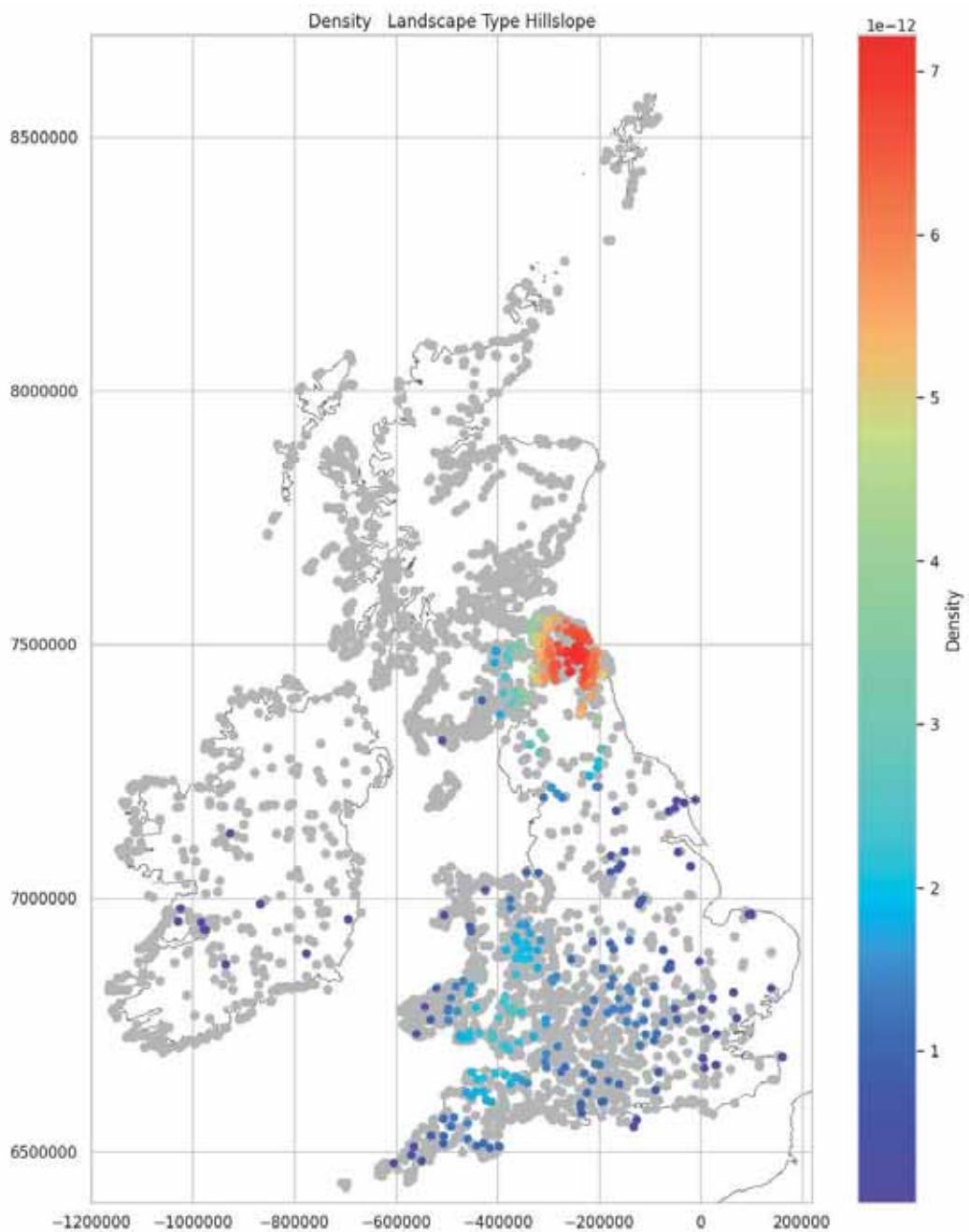
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

10.3%

Hillslope Density Mapped (Landscape)

Although caution should be taken with this data due to the probable bias, the hillslope data shows a strong concentration to the east of the Southern Uplands and another, weak concentration, over the Brecon Beacons and Exmoor.

```
In [ ]: plot_density_over_grey(tp_hillslope, 'Landscape_Type_Hillslope')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

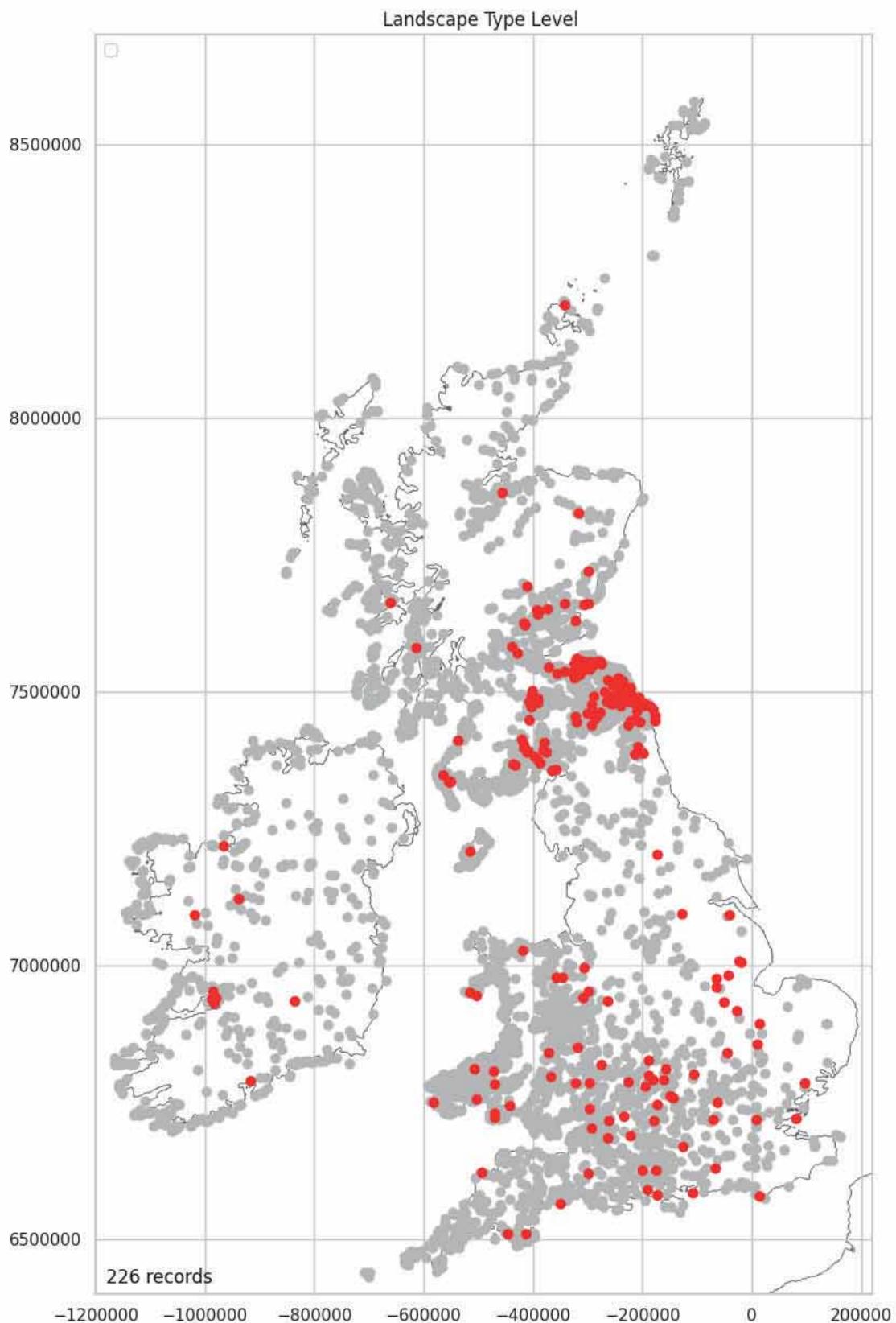
Level Mapped (Landscape)

There is a bias in that 74.83% of 'Level' hillforts, in the north, are known because of intensive cropmark survey carried out in southern Scotland. This can be seen in the distribution of 'Level' hillforts within the Tweed Valley and around the lowland fringes of the eastern Southern Uplands. There are a peppering of this type across the remainder of Scotland, Ireland and southern England.

See: [Cropmark Mapped](#)

See: [Aspect Level Mapped](#)

```
In [ ]: tp_Level = plot_over_grey(location_Landscape_data, \
'Landscape_Type_Level', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5.45%

```
In [ ]: split_location = 7070000
north = hillforts_data[hillforts_data['Location_Y'] >= split_location]
level_n = north[(north['Landscape_Type_Level'] == 'Yes')]
print(f'{len(level_n)} of the level hillforts are in the north.')
```

151 of the level hillforts are in the north.

```
In [ ]: level_cropmark_n = \
level_n[level_n['Management_Condition_Cropmark'] == 'Yes']
print(f'{len(level_cropmark_n)} ({round((len(level_cropmark_n)/len(level_n))*100, 2)}%) of the level hillforts in the
```

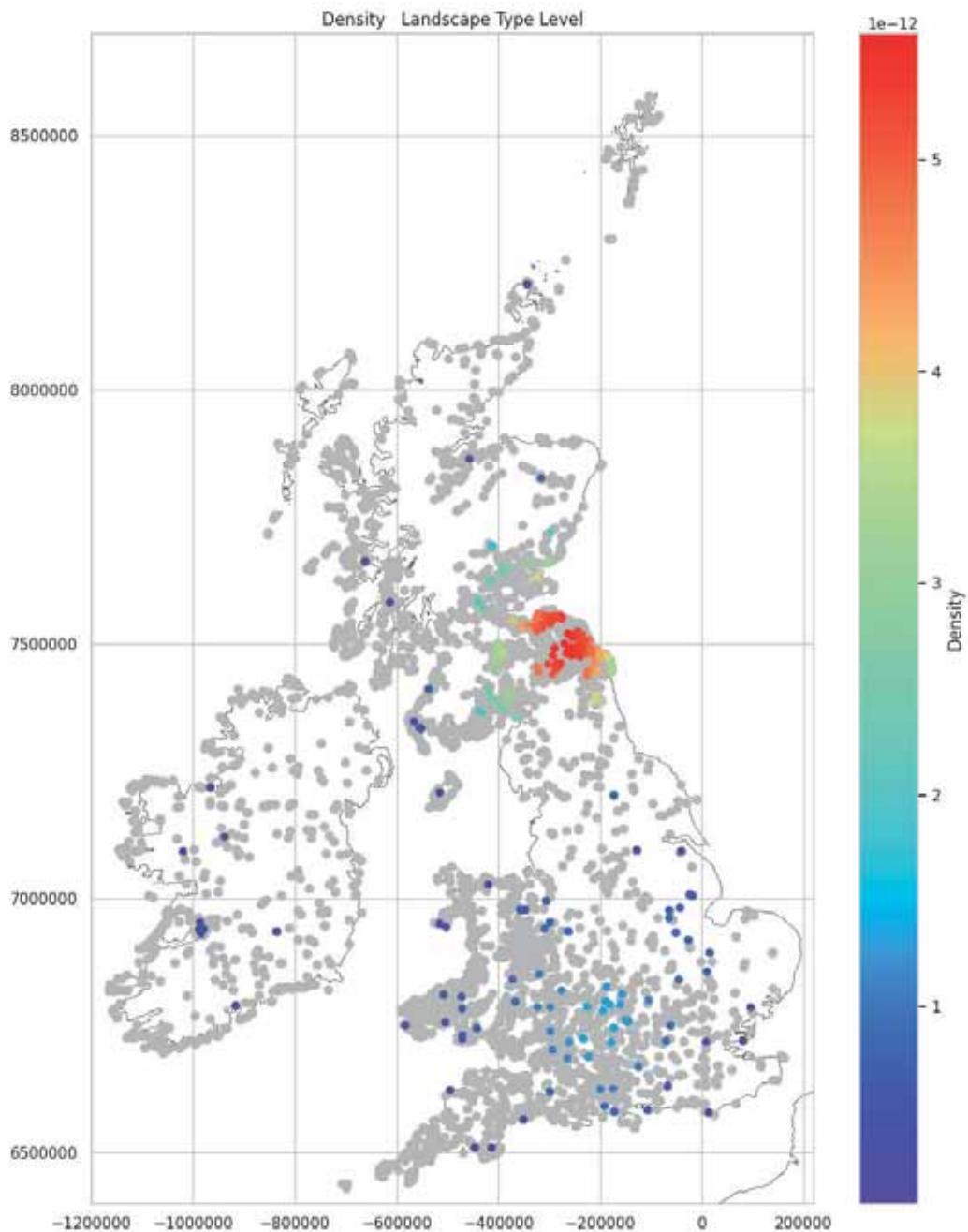
```
#print(f' {len(level_cropmark_n)} ({round((len(level_cropmark_n)/len(level_n))*#100, 2)}%) of the level hillforts in the north are cropmark sites.')
```

113 (74.83%) of the level hillforts in the north are cropmark sites.

Level Density Mapped

Because of the bias created by the intensive aerial survey work over the Tweed Basin, the northern cluster is exaggerated. There is a second, very slight concentration of level forts between the Cotswolds and the Shropshire Hills.

```
In [ ]: plot_density_over_grey(tp_level, 'Landscape_Type_Level')
```



Middleton, M. 2024, Hillforts Primer

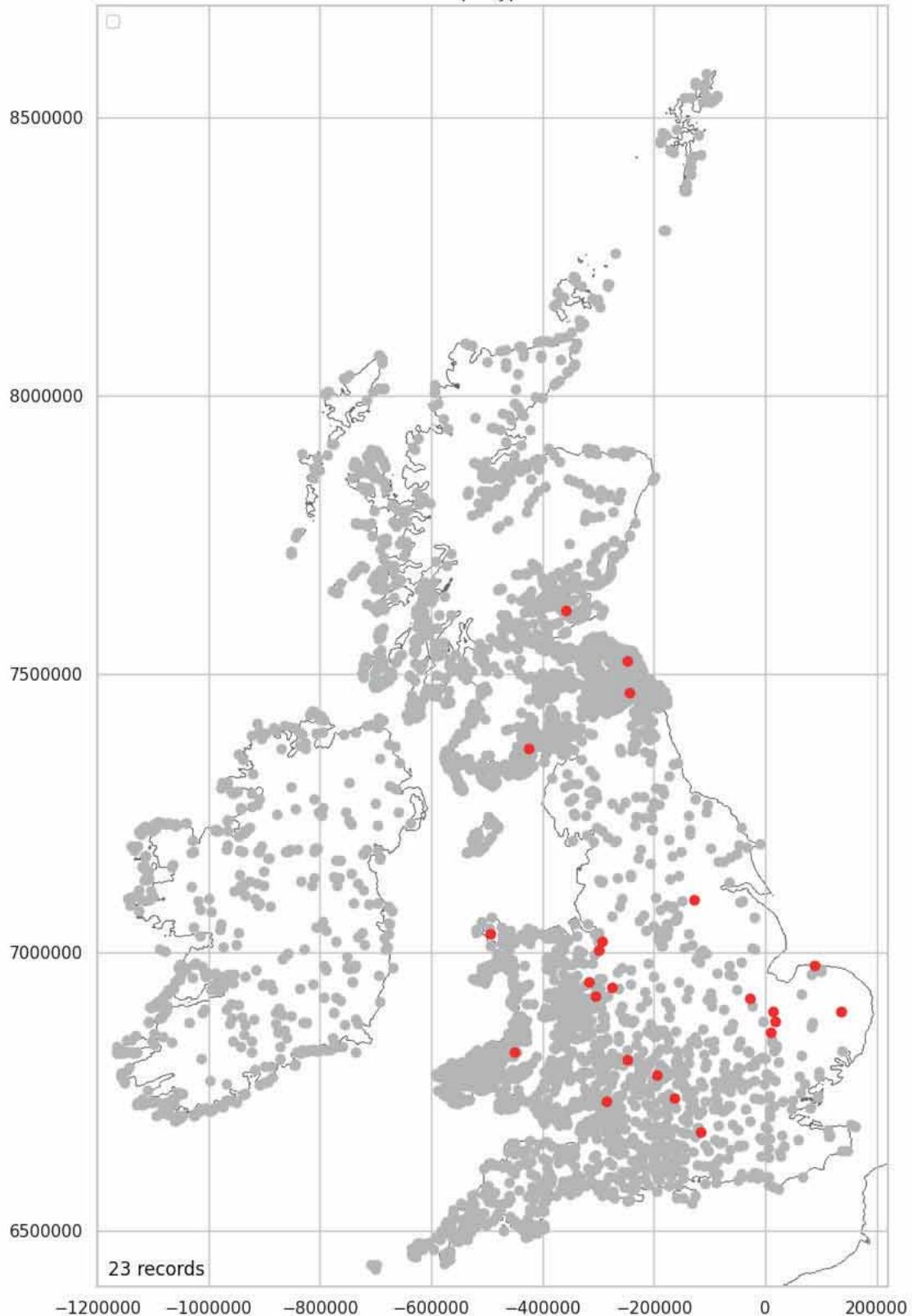
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Marsh Mapped

There are only 23 hillforts identified as type 'Marsh'. A density map has not been produced as a map based on such a small sample will be misleading.

```
In [ ]: tp_marsh = plot_over_grey(location_landscape_data, \
'Landscape_Type_Marsh', 'Yes')
```

Landscape Type Marsh



Middleton, M. 2024, Hillforts Primer

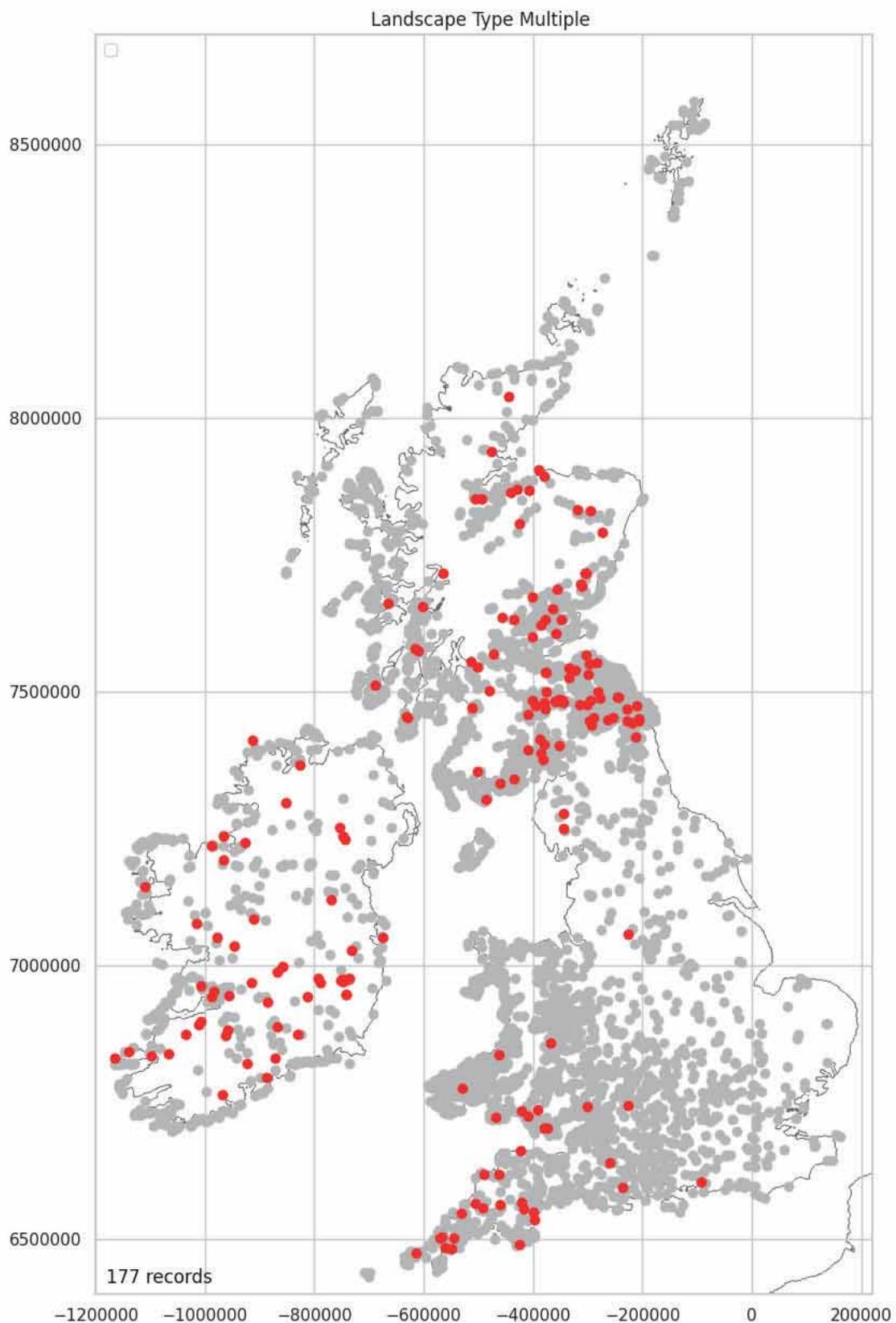
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.55%

Multiple Mapped

There is a recording bias in that the 'Multiple' Hillforts type has not been recorded uniformly. There are very few records in England and Wales, with almost none in the north, east and south England and the north of Wales.

```
In [ ]: tp_multiple = plot_over_grey(location_landscape_data, \
'Landscape_Type_Multiple', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

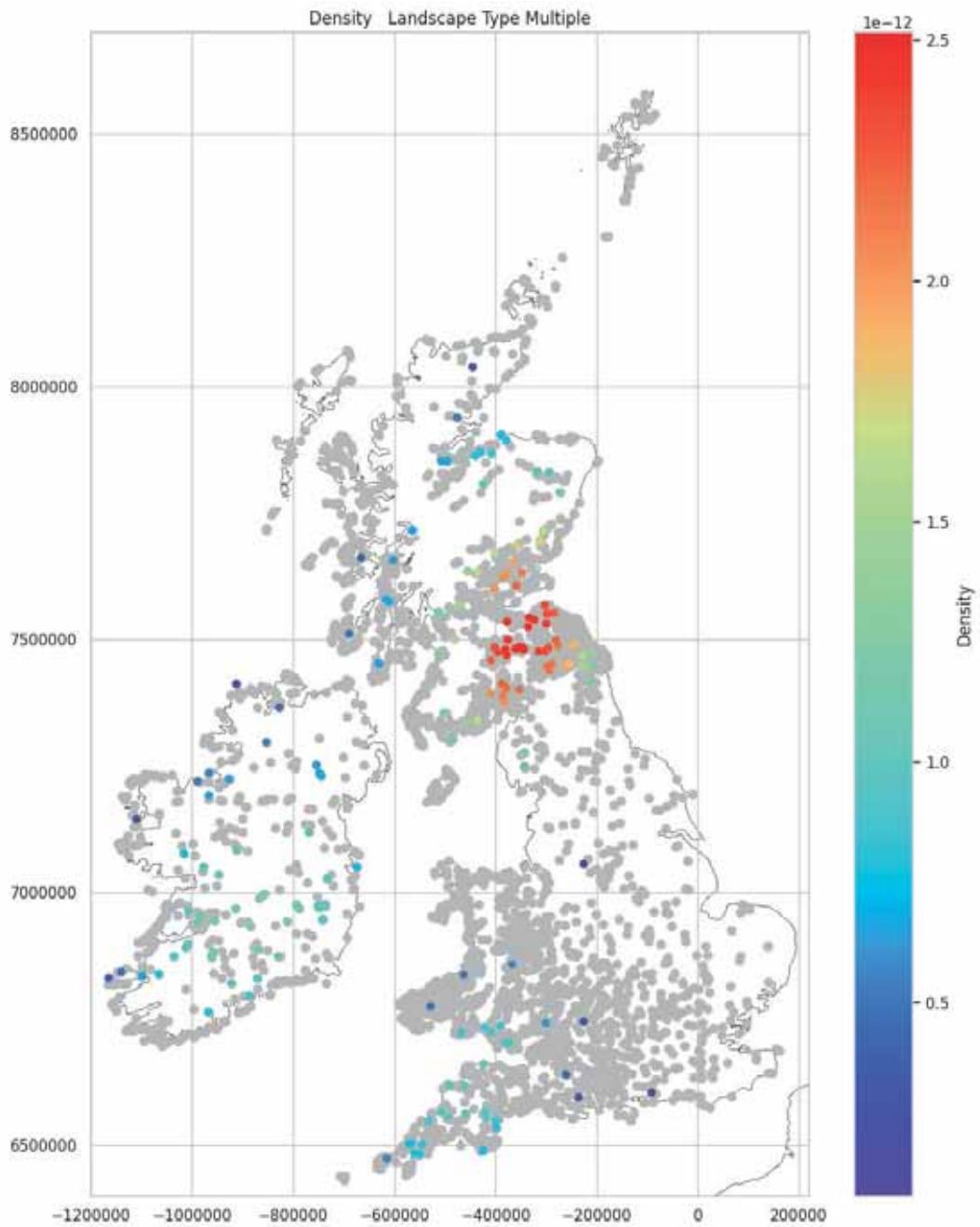
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 27%

Multiple Density Mapped

The bias in this data means the clusters in this density plot are not reliable. Where there is data, the eastern Southern Uplands is the main cluster and central Ireland shows as a slight, secondary concentration.

```
In [ ]: plot_density_over_grey(tp_multiple, 'Landscape_Type_Multiple')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Topography Data

The topography data consists of 10 topographic types. Multiple types can be assigned to a single hillfort.

```
In [ ]: landscape_topography_features = [
    'Landscape_Topography_Hilltop',
    'Landscape_Topography_Coastal',
    'Landscape_Topography_Inland',
    'Landscape_Topography_Valley',
    'Landscape_Topography_Knoll',
    'Landscape_Topography_Ridge',
    'Landscape_Topography_Scarp',
    'Landscape_Topography_Hillslope',
    'Landscape_Topography_Lowland',
    'Landscape_Topography_Spur']
```

```
In [ ]: landscape_topography_data = \
landscape_encodeable_data[landscape_topography_features].copy()
landscape_topography_data.head()
```

Out[]:	Landscape_Topography_Hilltop	Landscape_Topography_Coastal	Landscape_Topography_Inland	Landscape_Topography_Valley	Landscape_T
0	Yes	No	Yes	No	No
1	Yes	No	No	No	No
2	Yes	No	No	No	No
3	Yes	No	No	No	No
4	Yes	No	No	No	No

The topography data contains no null values.

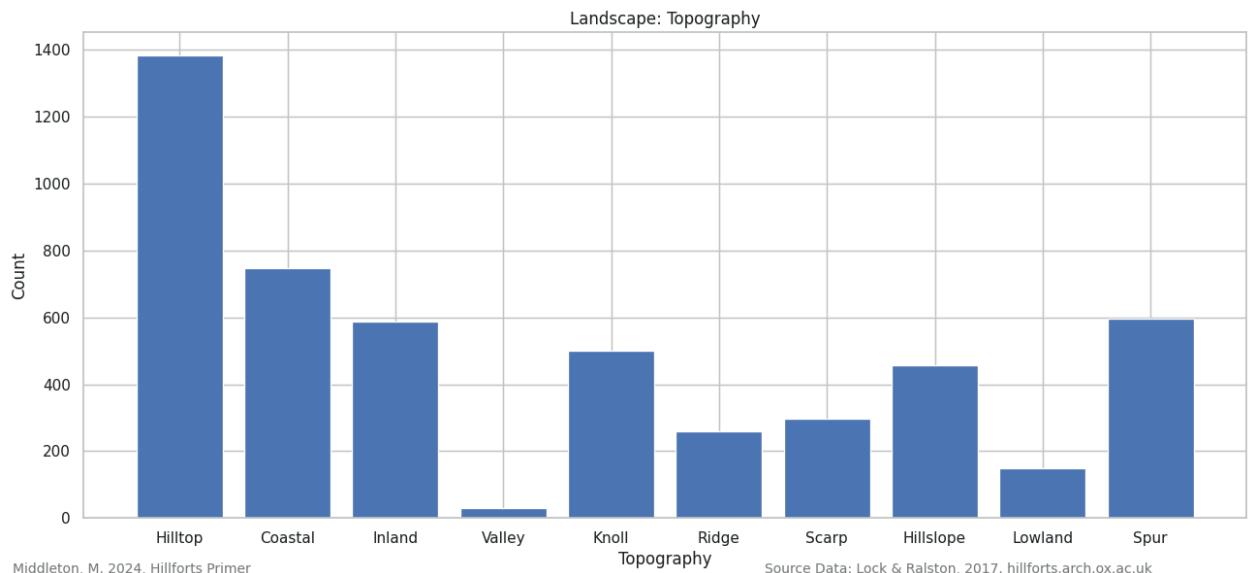
In []: `landscape_topography_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Landscape_Topography_Hilltop    4147 non-null   object  
 1   Landscape_Topography_Coastal   4147 non-null   object  
 2   Landscape_Topography_Inland   4147 non-null   object  
 3   Landscape_Topography_Valley  4147 non-null   object  
 4   Landscape_Topography_Knoll    4147 non-null   object  
 5   Landscape_Topography_Ridge   4147 non-null   object  
 6   Landscape_Topography_Scarp   4147 non-null   object  
 7   Landscape_Topography_Hillslope 4147 non-null   object  
 8   Landscape_Topography_Lowland  4147 non-null   object  
 9   Landscape_Topography_Spur    4147 non-null   object  
dtypes: object(10)
memory usage: 324.1+ KB
```

Topography Data Plotted

Unsurprisingly, most of the topographic types relate to upland features such as hill tops, ridges, spurs and knolls. Very few hillforts are identified as being in lowland areas or valleys. Coastal, inland and lowland types are notable as these are not topographic features but rather, areas which are defined with reference to the topography. A number of the terms look to have a regional bias.

In []: `plot_bar_chart(landscape_topography_data, 2, 'Topography', 'Count', 'Landscape: Topography')`



Topography Data Mapped

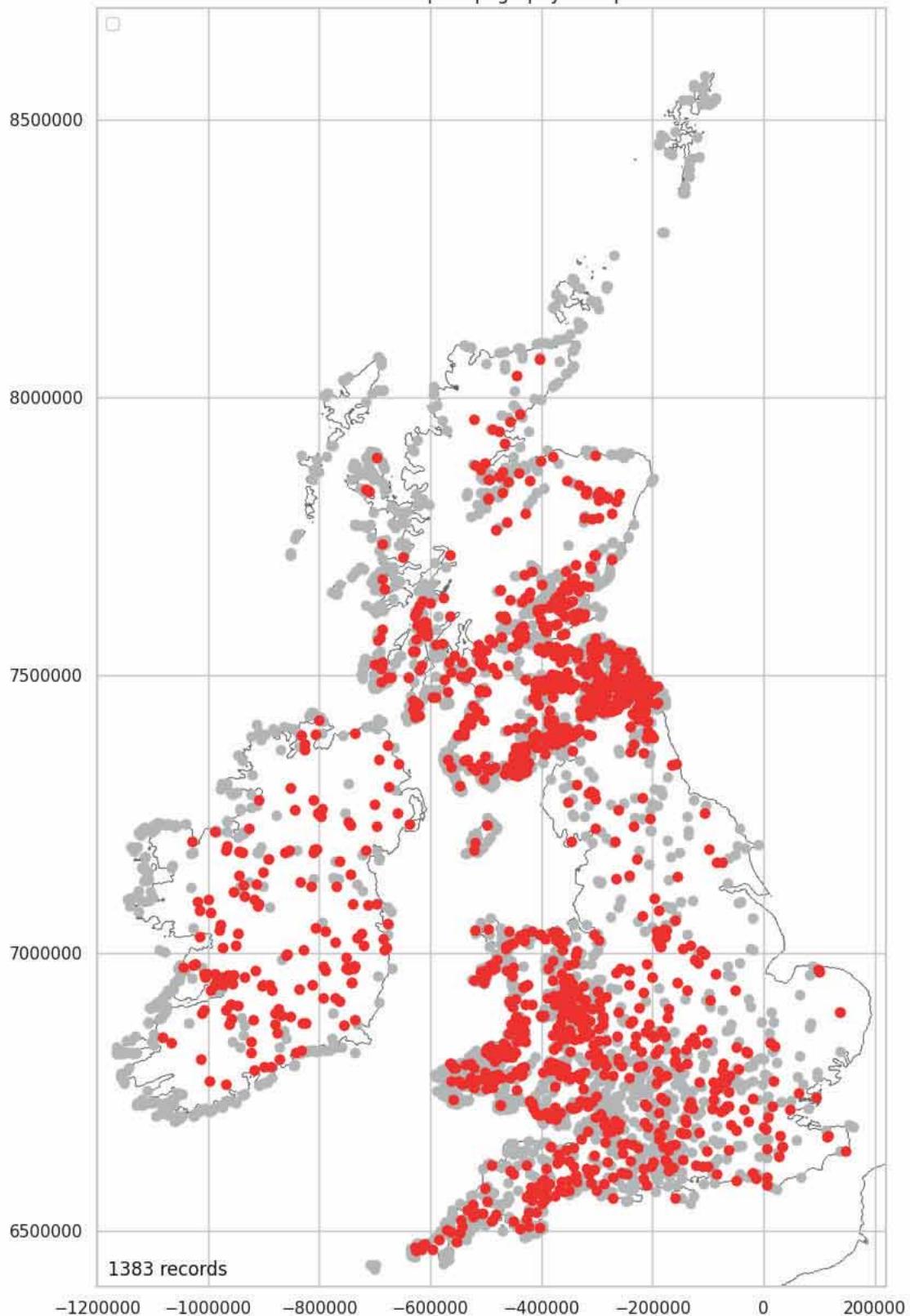
In []: `location_topo_data = pd.merge(location_numeric_data_short, landscape_topography_data, left_index=True, right_index=True)`

Hilltop Mapped

One third (1283) of all hillforts are recorded as being located on a hilltop.

In []: `topo_hilltop = plot_over_grey(location_topo_data, 'Landscape_Topography_Hilltop', 'Yes')`

Landscape Topography Hilltop



Middleton, M. 2024, Hillforts Primer

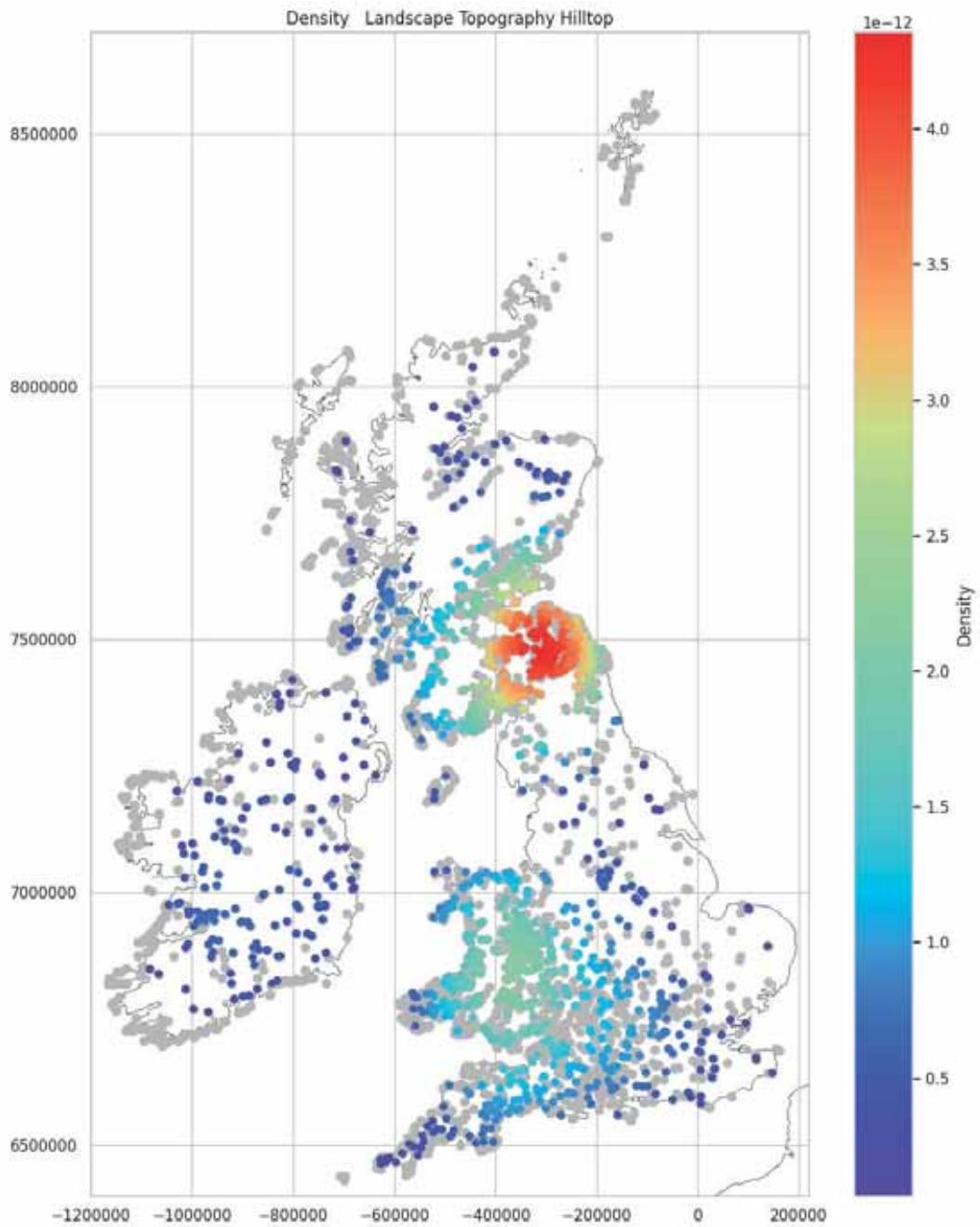
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

33. 35%

Hilltop Density Mapped

Hilltop densities have two main clusters which correspond with the main hillfort density concentrations seen over the Southern Uplands and the Cambrian Mountains. The clusters in the Northwest and the west of Ireland are not present. See Part 1: Density Map Showing Clusters Adjusted by Region.

```
In [ ]: plot_density_over_grey(topo_hilltop, 'Landscape_Topography_Hilltop')
```



Middleton, M. 2024, Hillforts Primer

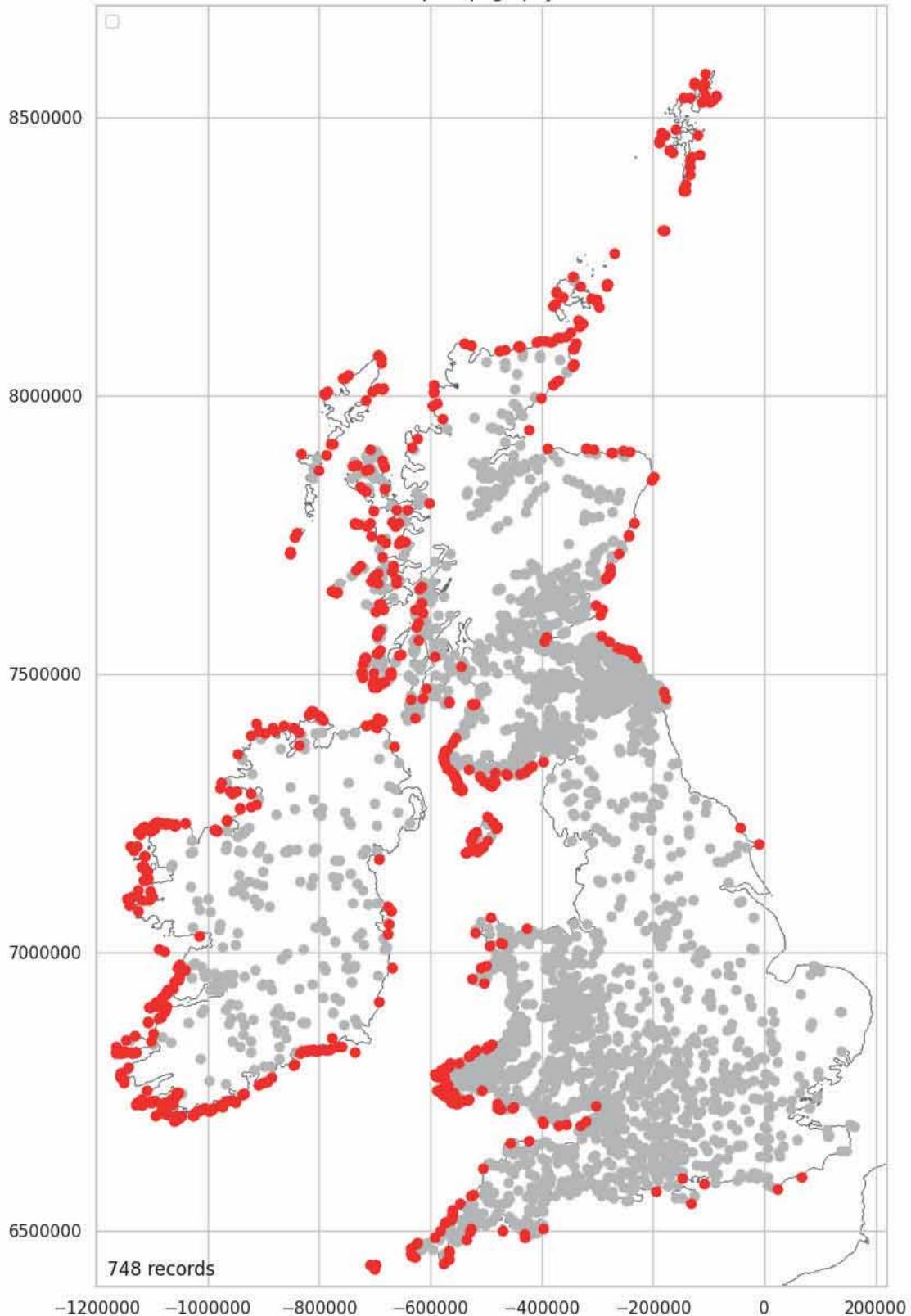
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Coastal Mapped (Topo)

748 hillforts (18.04%) are identified as coastal, with the west coasts of the mainland and Ireland being the focus for most. There are very few coastal forts along the east and south coasts of England. Coastal erosion, especially in the southeast, is likely to be at least partly responsible for this. There is a notable gap in the distribution of hillforts along the west coast of England between the Scottish and Welsh borders. See: [Coastal Mapped \(Land Use\)](#).

```
In [ ]: topo_coastal = plot_over_grey(location_topo_data, \
'Landscape_Topography_Coastal', 'Yes')
```

Landscape Topography Coastal



Middleton, M. 2024, Hillforts Primer

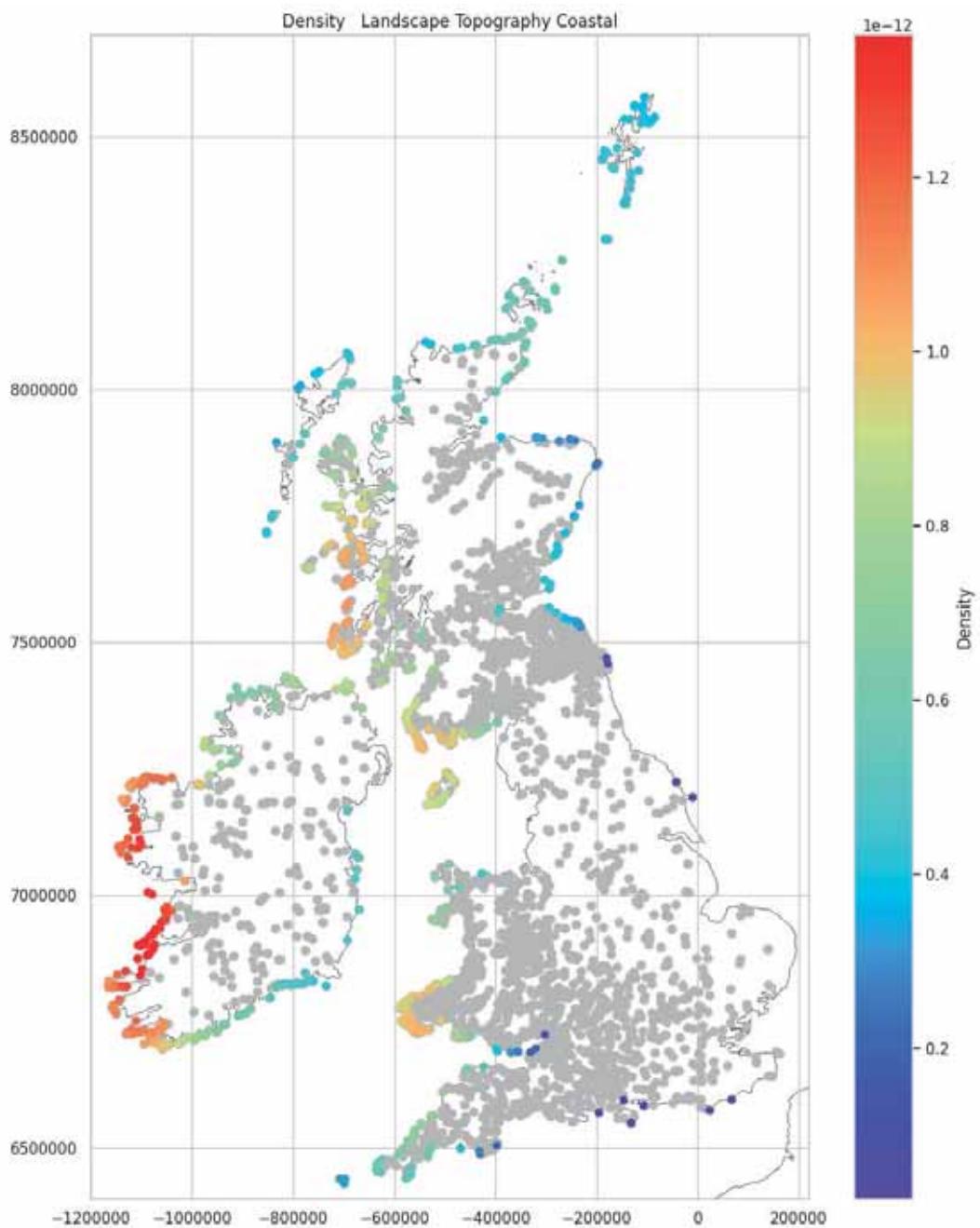
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

18.04%

Coastal Density Mapped (Topography)

The strongest concentration of coastal forts is along the west coast of Ireland. There are smaller concentrations on the Pembrokeshire coast; around Luce Bay (near Whithorn and including the Isle of Man) and along the line of islands from Islay to southern Skye. See: [Coastal Density Mapped \(Land Use\)](#).

```
In [ ]: plot_density_over_grey(topo_coastal, 'Landscape_Topography_Coastal')
```



Middleton, M. 2024, Hillforts Primer

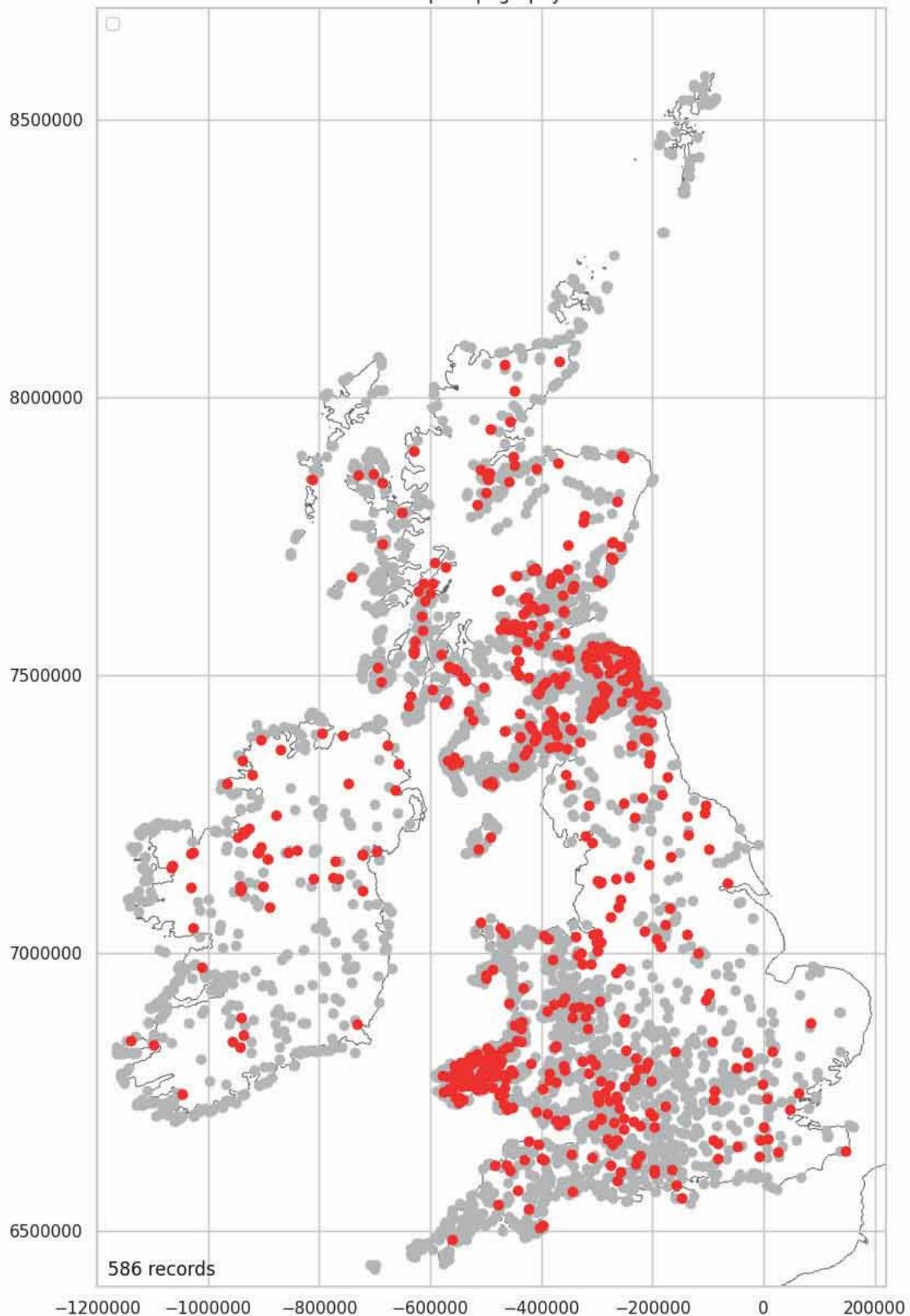
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Inland Mapped

It is unclear why some hillforts are classified as inland and why others are not.

```
In [ ]: topo_inland = plot_over_grey(location_topo_data, \
'Landscape_Topography_Inland', 'Yes')
```

Landscape Topography Inland



Middleton, M. 2024, Hillforts Primer

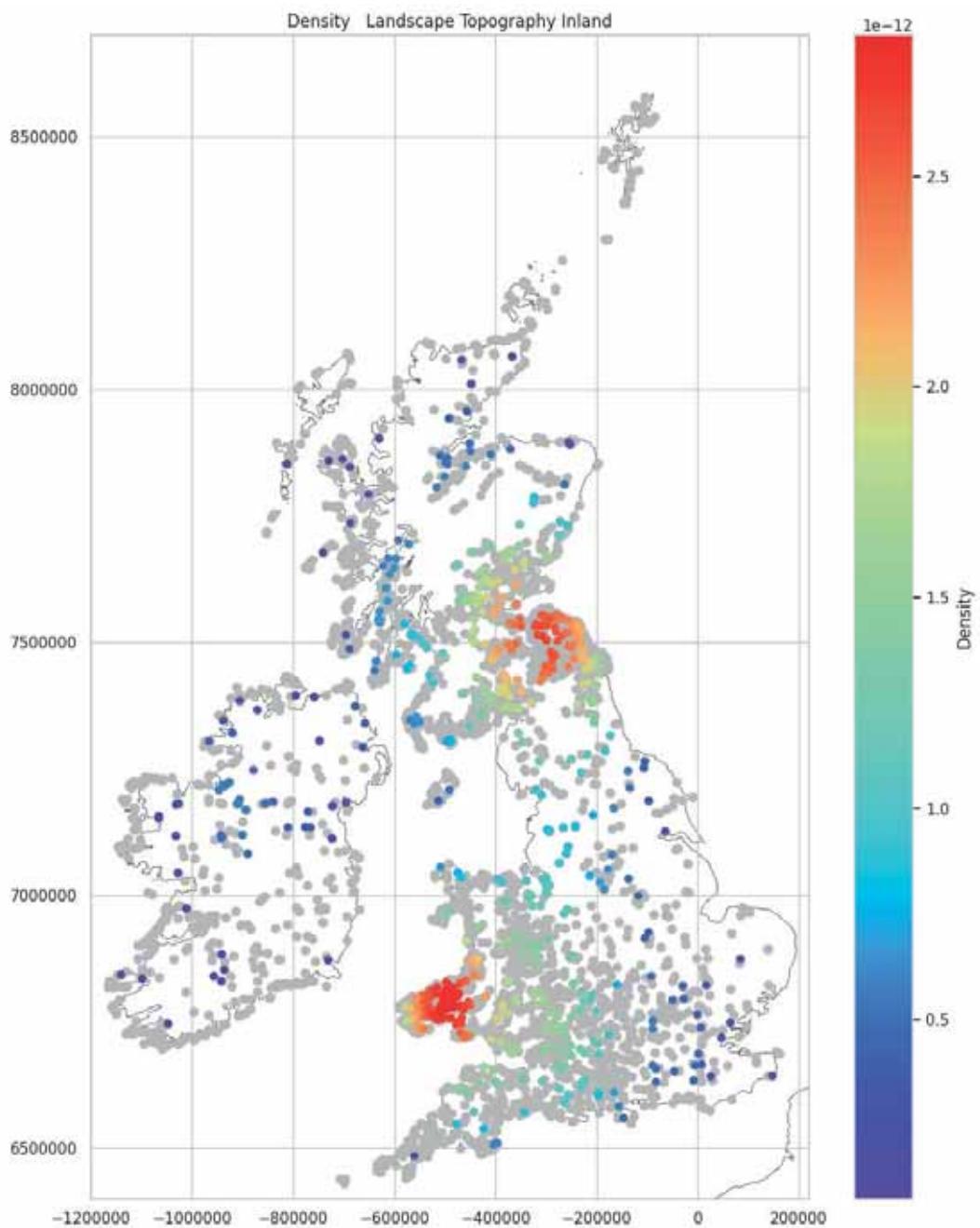
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

14. 13%

Inland Density Mapped

The inland density contains two main clusters. These mirror the two main clusters seen in the main atlas distribution (see: Part 1: Density Map Showing Clusters Adjusted by Region). The Welsh concentration is interestingly focussed slightly further west, with the cluster being located over the Preseli Hills and the Pembrokeshire peninsula.

```
In [ ]: plot_density_over_grey(topo_inland, 'Landscape_Topography_Inland')
```



Middleton, M. 2024, Hillforts Primer

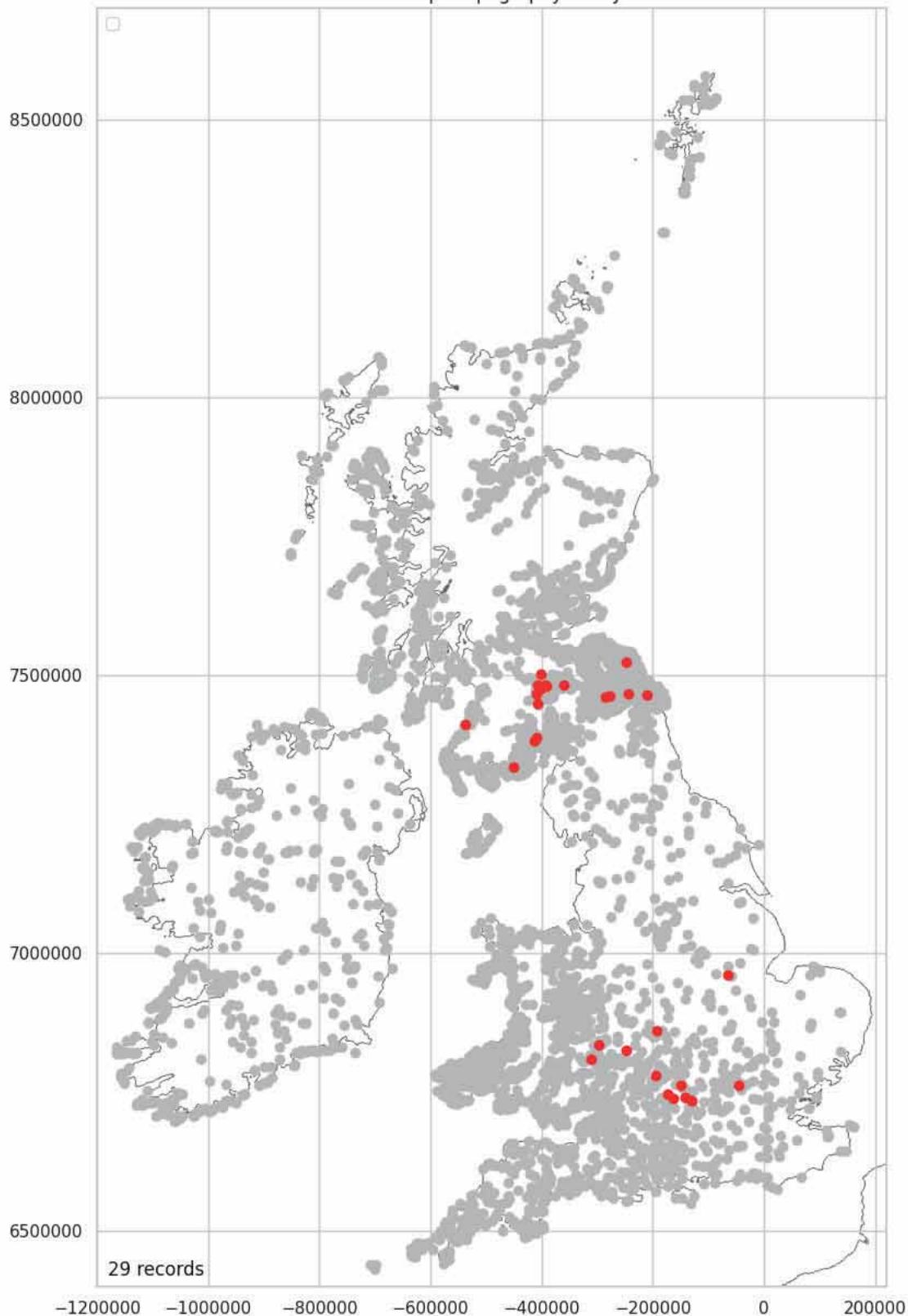
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Valley Mapped

Only 29 hillforts are identified as located in a 'valley'.

```
In [ ]: topo_valley = plot_over_grey(location_topo_data, \
'Landscape_Topography_Valley', 'Yes')
```

Landscape Topography Valley



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

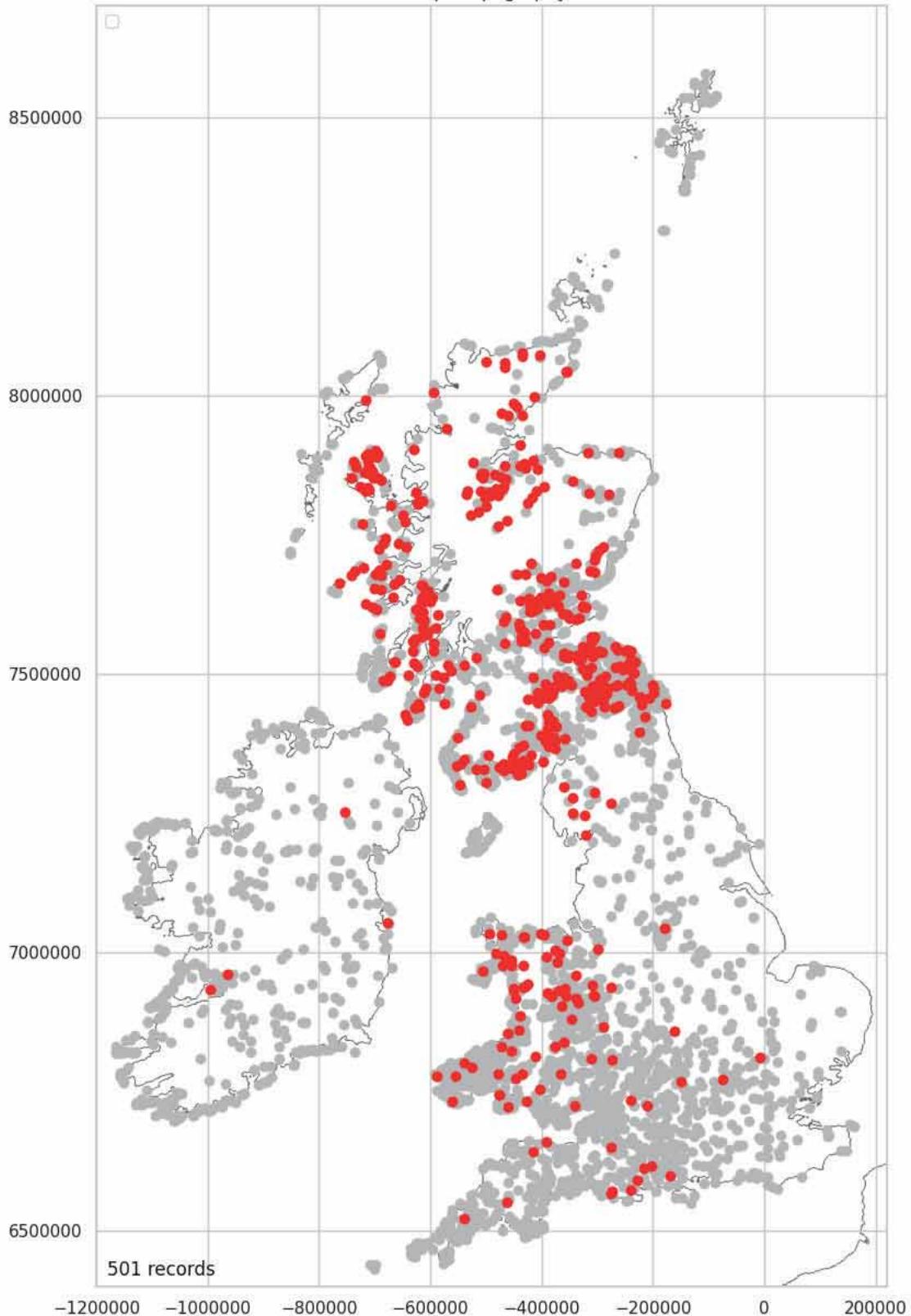
0.7%

Knoll Mapped

Hillforts identified as being located on a 'knoll' are almost exclusively in the upland areas of Scotland and Wales. The term is hardly used in Ireland and is far more common in Scotland. This may indicate a regional bias in where it is used.

```
In [ ]: topo_knoll = plot_over_grey(location_topo_data, \
'Landscape_Topography_Knoll', 'Yes')
```

Landscape Topography Knoll



Middleton, M. 2024, Hillforts Primer

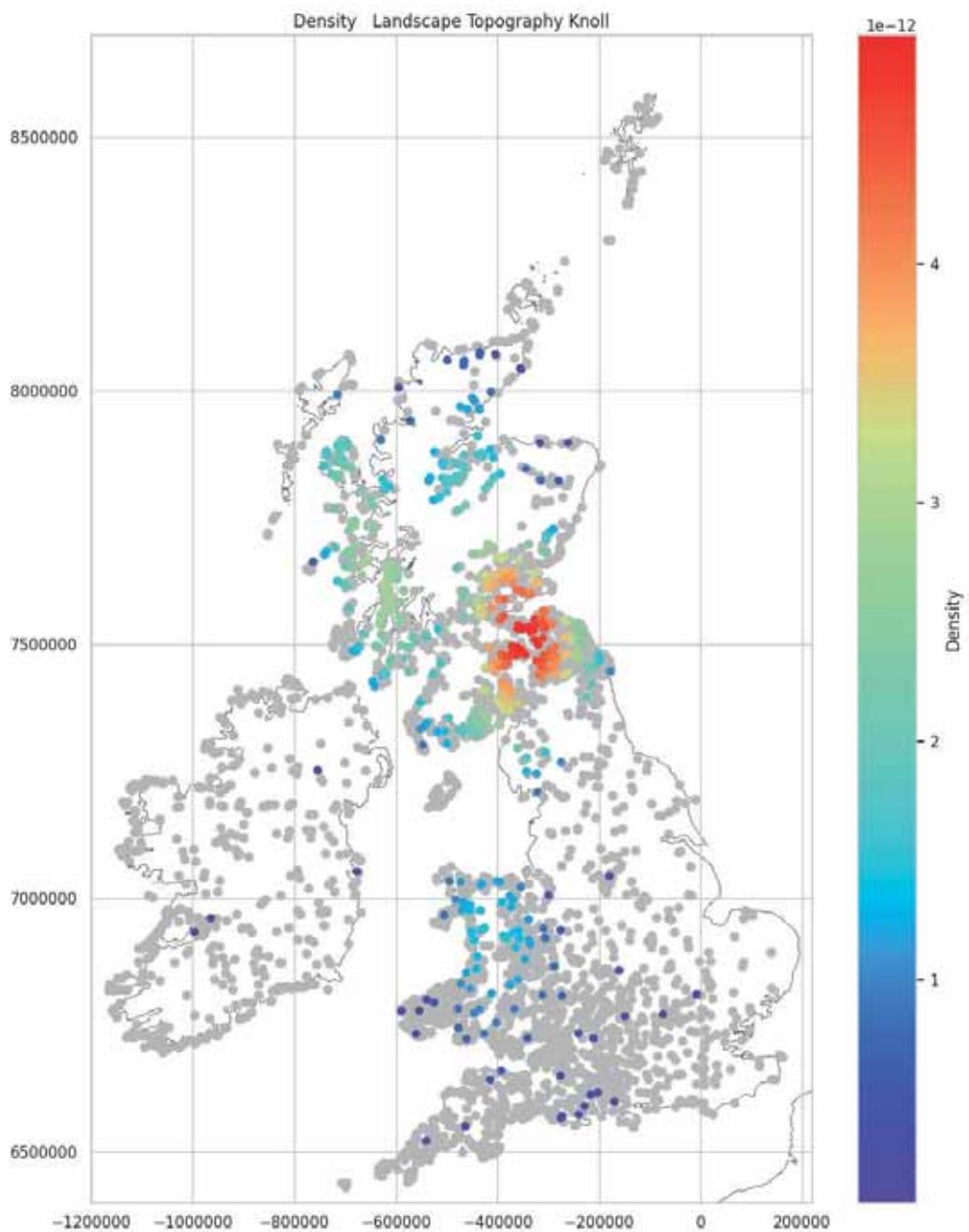
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

12.08%

Knoll Density Mapped

The main concentration of hillforts located on a 'knoll' is along the northern edge of the Southern Uplands, particularly across the Pentlands and Lammermuir hills, and along the west coast of Scotland, around the sea loch, Loch Linnhe. In Wales the focus is toward the northern end of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(topo_knoll, 'Landscape_Topography_Knoll')
```

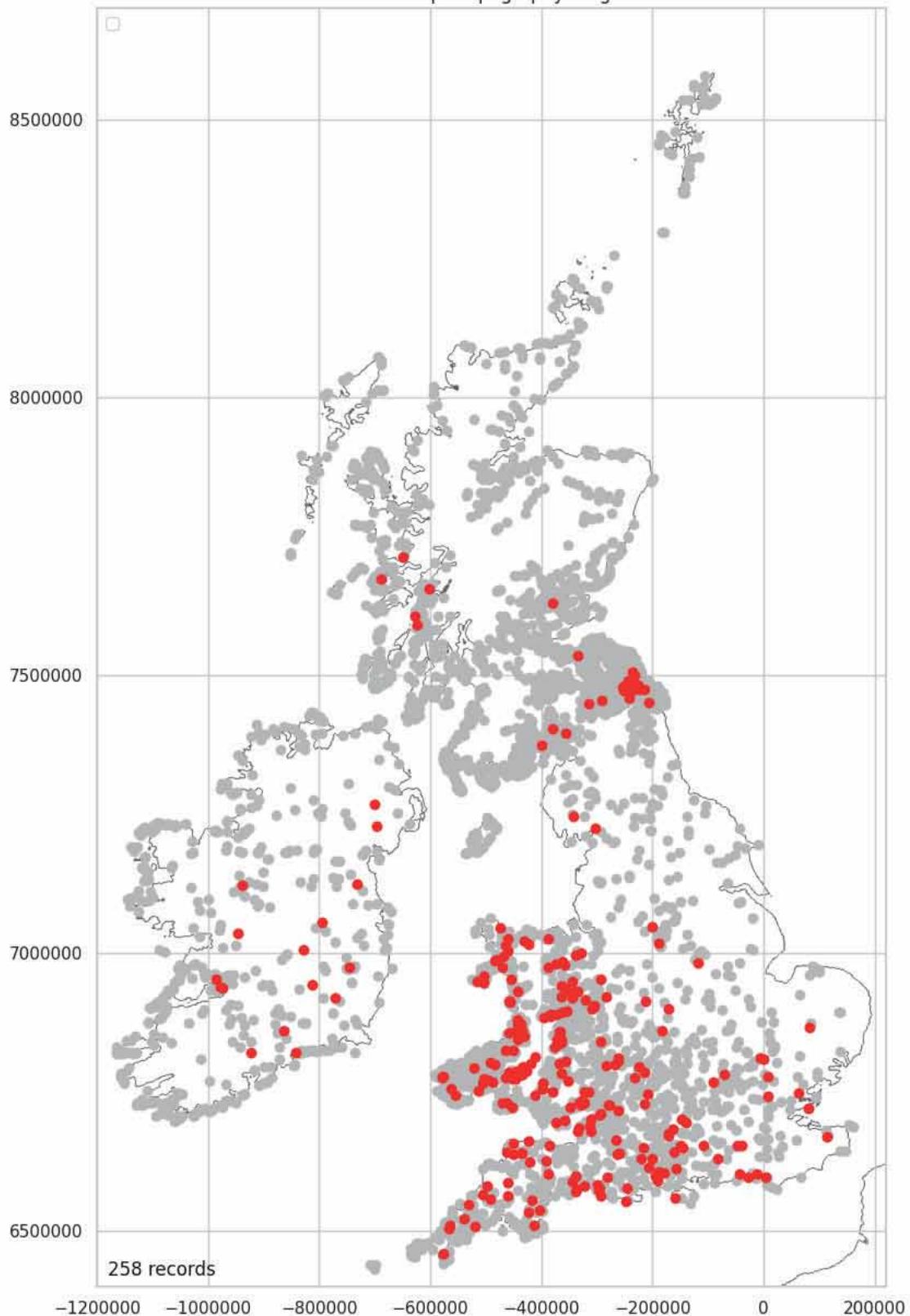


Ridge Mapped

The distribution of hillforts identified as on a 'ridge' is almost entirely focussed toward the South East and Wales. This may reflect a regional terminology bias.

```
In [ ]: topo_ridge = plot_over_grey(location_topo_data,
                                'Landscape_Topography_Ridge', 'Yes')
```

Landscape Topography Ridge



Middleton, M. 2024, Hillforts Primer

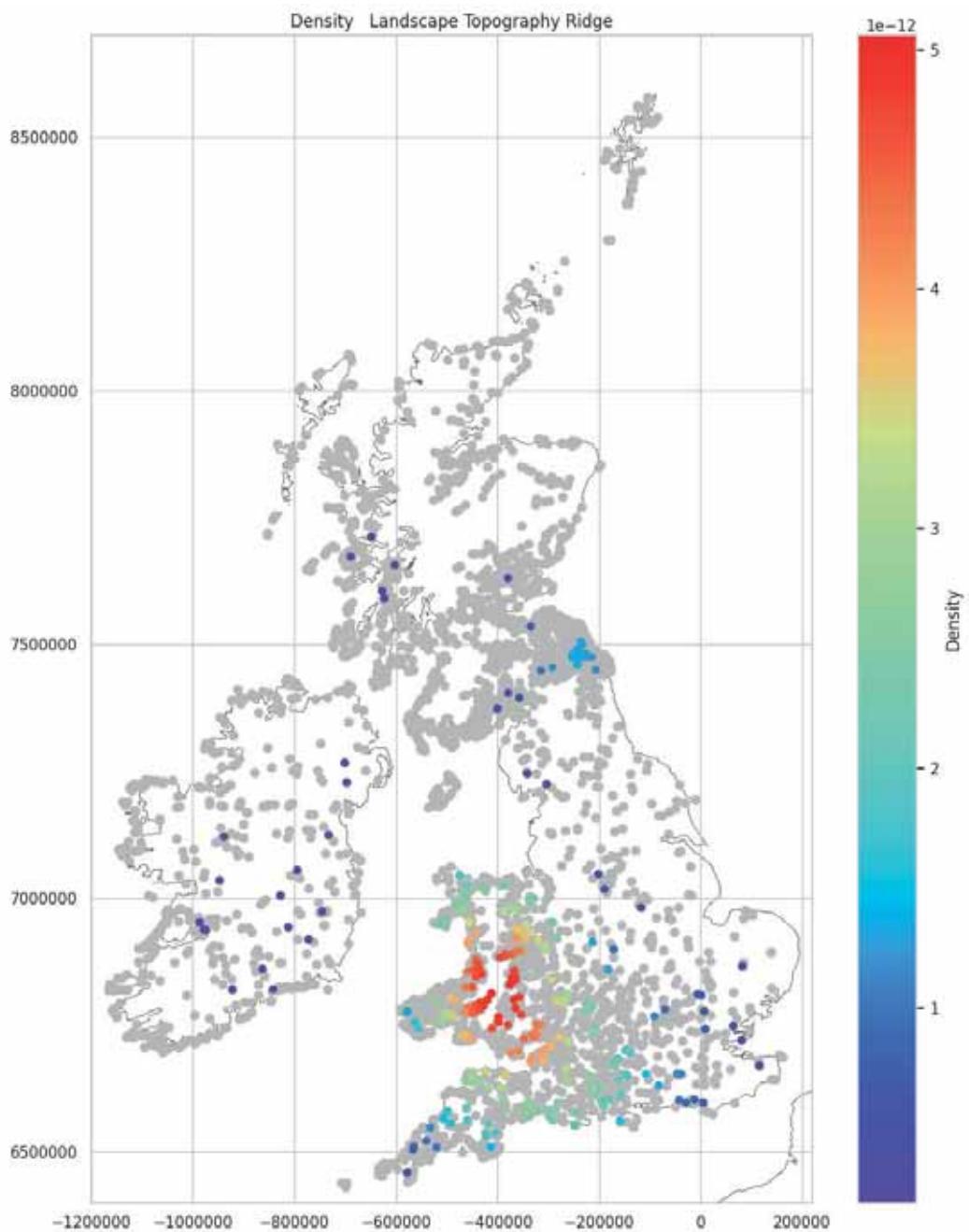
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 22%

Ridge Density Mapped

The southern cluster of hillforts on ridges coincides with the southern cluster seen in the full dataset. As mentioned above, there may be a terminology bias present in this distribution. See: Part 1: Density Map Showing Clusters Adjusted by Region.

```
In [ ]: plot_density_over_grey(topo_ridge, 'Landscape_Topography_Ridge')
```



Middleton, M. 2024, Hillforts Primer

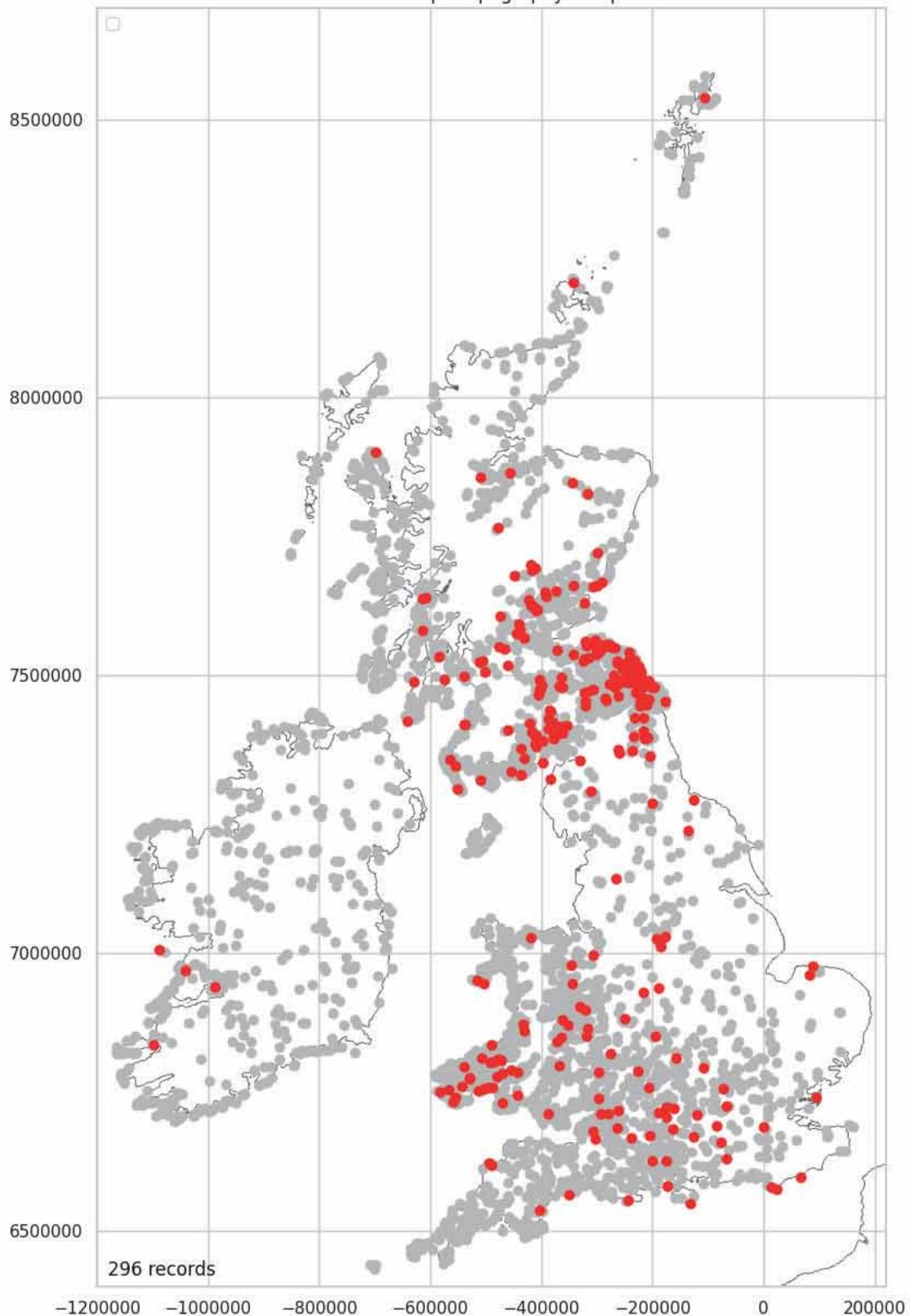
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Scarp Mapped

In Scotland, the use of the term 'Scarp' is concentrated mostly over the Southern Uplands and along the Highland Boundary fault. To the south, there is a concentration of hillforts in Pembrokeshire and a spread of hillforts across south, central England. The term is hardly used in Ireland and this may indicate a regional bias.

```
In [ ]: topo_scarp = plot_over_grey(location_topo_data, \
    'Landscape_Topography_Scarp', 'Yes')
```

Landscape Topography Scarp



Middleton, M. 2024, Hillforts Primer

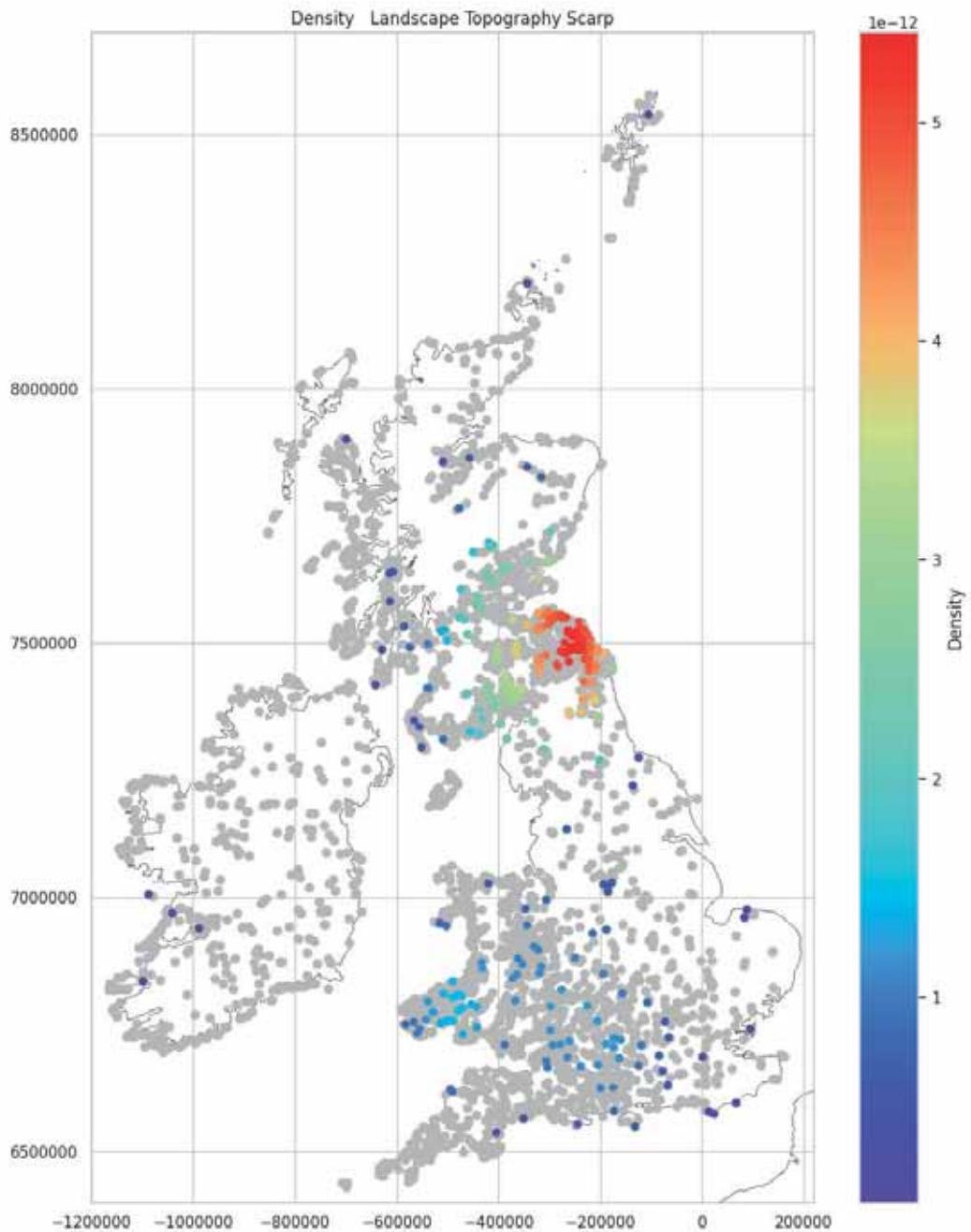
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

7.14%

Scarp Density Mapped

'Scarp' hillfort density over the Southern Uplands corresponds to the lower altitude forts in the Tweed Basin. (See: [Altitude Over 0 - 99m Density Mapped](#)). Just under half (48.34%) are known from the cropmark record. This distribution will contain a survey bias (intensive cropmark survey over Southern Scotland) and may also include a regional bias in the use of this term.

```
In [ ]: plot_densitiy_over_grey(topo_scarp, 'Landscape_Topography_Scarp')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

```
In [ ]: level_n = north[north['Landscape_Type_Level']=='Yes']
print(f'{len(level_n)} of the level hillforts are in the north.')
```

151 of the level hillforts are in the north.

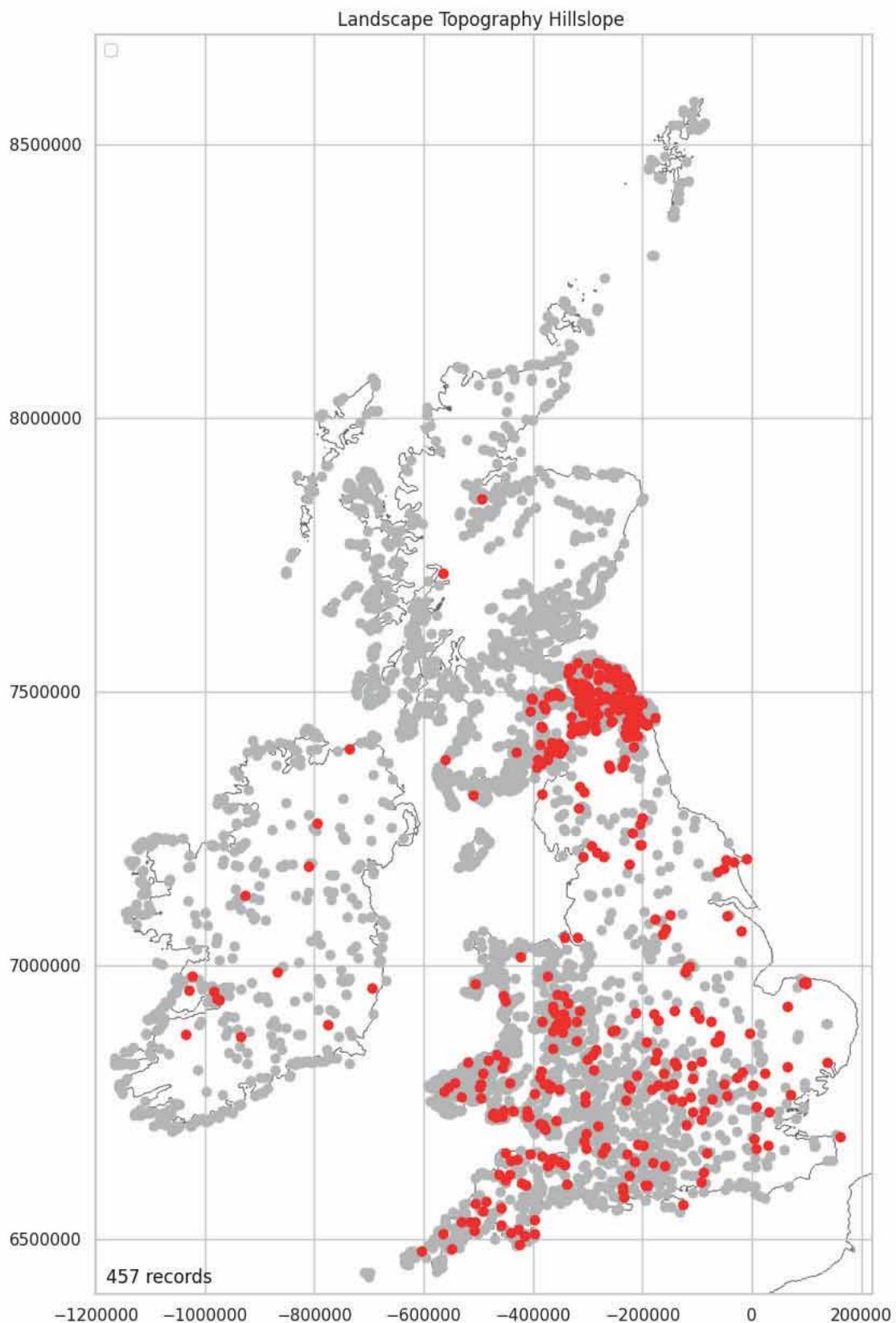
```
In [ ]: scarp_cropmark_n = level_n[level_n['Landscape_Topography_Scarp']=='Yes']
print(f'{len(scarp_cropmark_n)} ({round((len(scarp_cropmark_n)/len(level_n))*100, 2)}%) of the scarp hillforts in the
#print(f'{len(scarp_cropmark_n)} ({round((len(scarp_cropmark_n)/len(level_n))*100, 2)}%) of the scarp hillforts in the north are cropmark sites.')
```

73 (48.34%) of the scarp hillforts in the north are cropmark sites.

Hillslope Mapped (Topography)

The distribution of the term 'Hillslope' shows a bias toward the Southern Uplands. The term has been used widely in England and Wales although there is a notable lack of this type in northwest Wales. There are very few hillforts recorded as 'Hillslope' across the rest of Scotland and the distribution across Ireland is low. The bias seen here is mirrored in the bias seen in [Hillslope Mapped \(Landscape\)](#).

```
In [ ]: topo_hillslope = plot_over_grey(location_topo_data, \
'Landscape_Topography_Hillslope', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

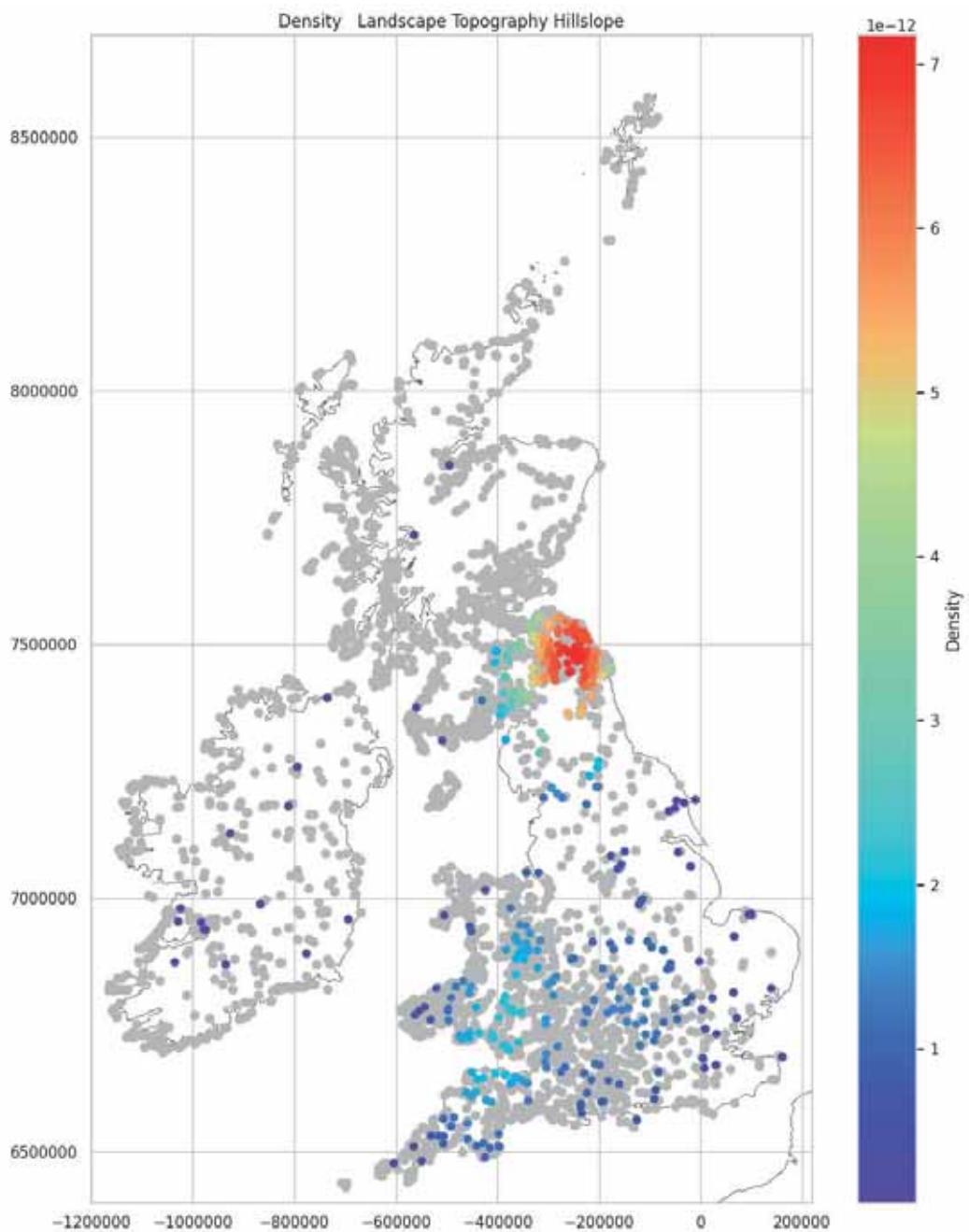
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

11.02%

Hillslope Density Mapped (Topography)

The 'Hillslope' density plot is very similar to that seen in [Hillslope Density Mapped \(Landscape\)](#).

```
In [ ]: plot_density_over_grey(topo_hillslope, 'Landscape_Topography_Hillslope')
```



Middleton, M. 2024, Hillforts Primer

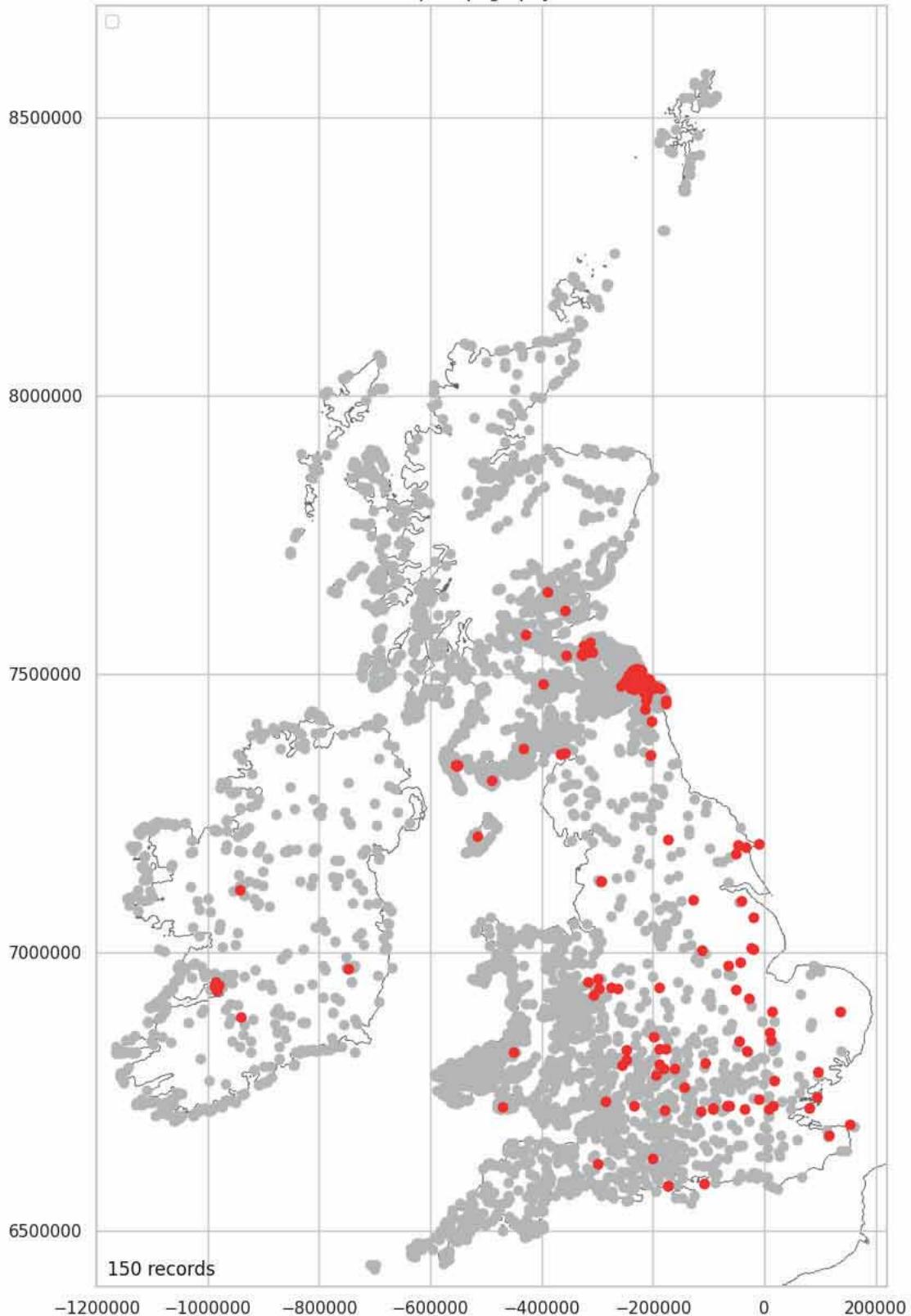
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Lowland Mapped

'Lowland' is a term that is relative to the topography rather than being a specific topographic type. Like similar relative terms used in this section, some 'lowland' hillforts have multiple topographic types recorded. There is a bias in the use of this term toward eastern England although the term has been used, in small numbers, across the whole of the atlas. Because of the bias, a density plot has not been produced.

```
In [ ]: topo_lowland = plot_over_grey(location_topo_data, \
'Landscape_Topography_Lowland', 'Yes')
```

Landscape Topography Lowland



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

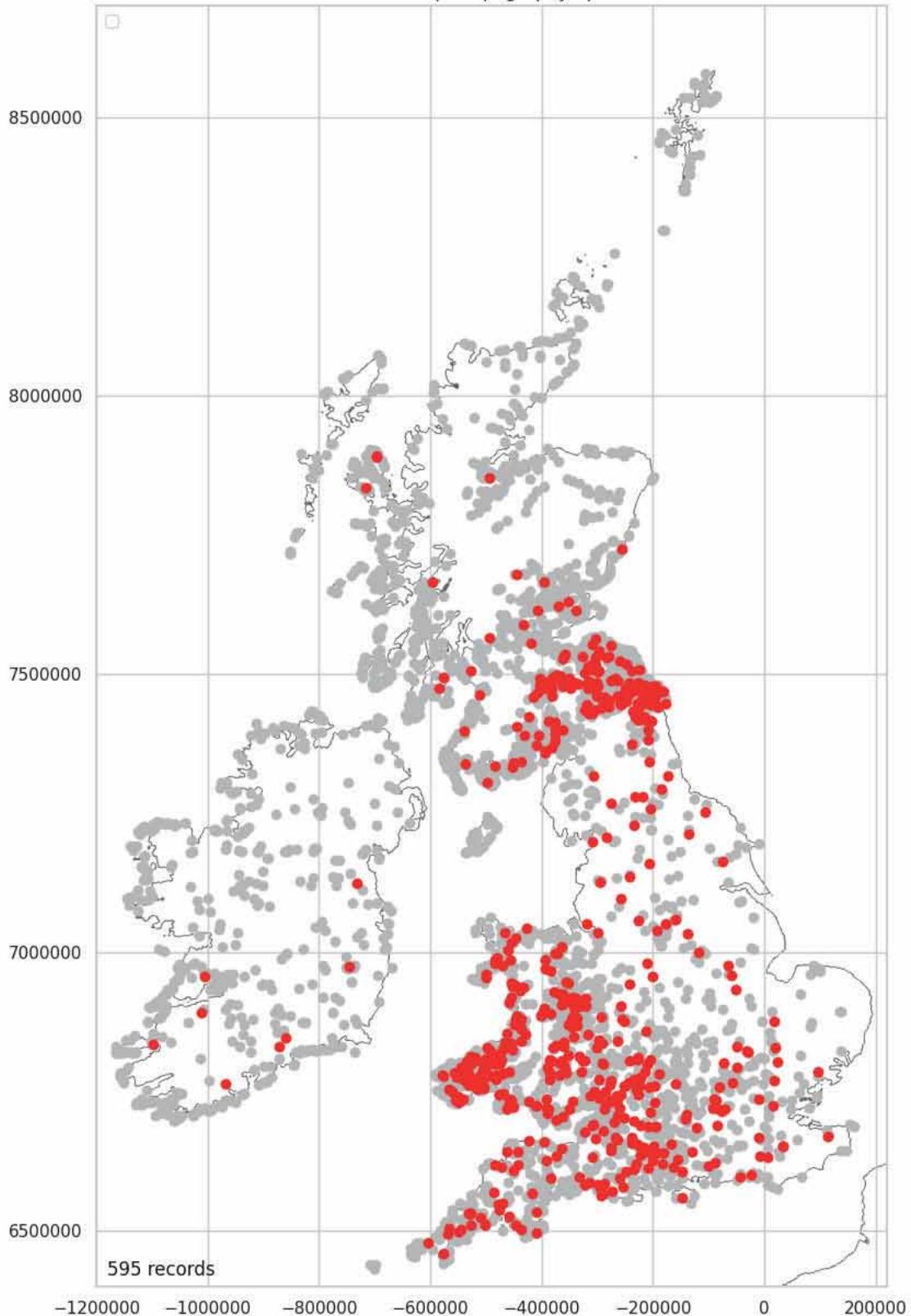
3. 62%

Spur Mapped

The distribution of the term 'Spur' is concentrated over the Southern Uplands, Wales and southeast England. The sparse distribution in Ireland and northern Scotland may hint at there being a recording bias.

```
In [ ]: topo_spur = plot_over_grey(location_topo_data,
    'Landscape_Topography_Spur', 'Yes')
```

Landscape Topography Spur



Middleton, M. 2024, Hillforts Primer

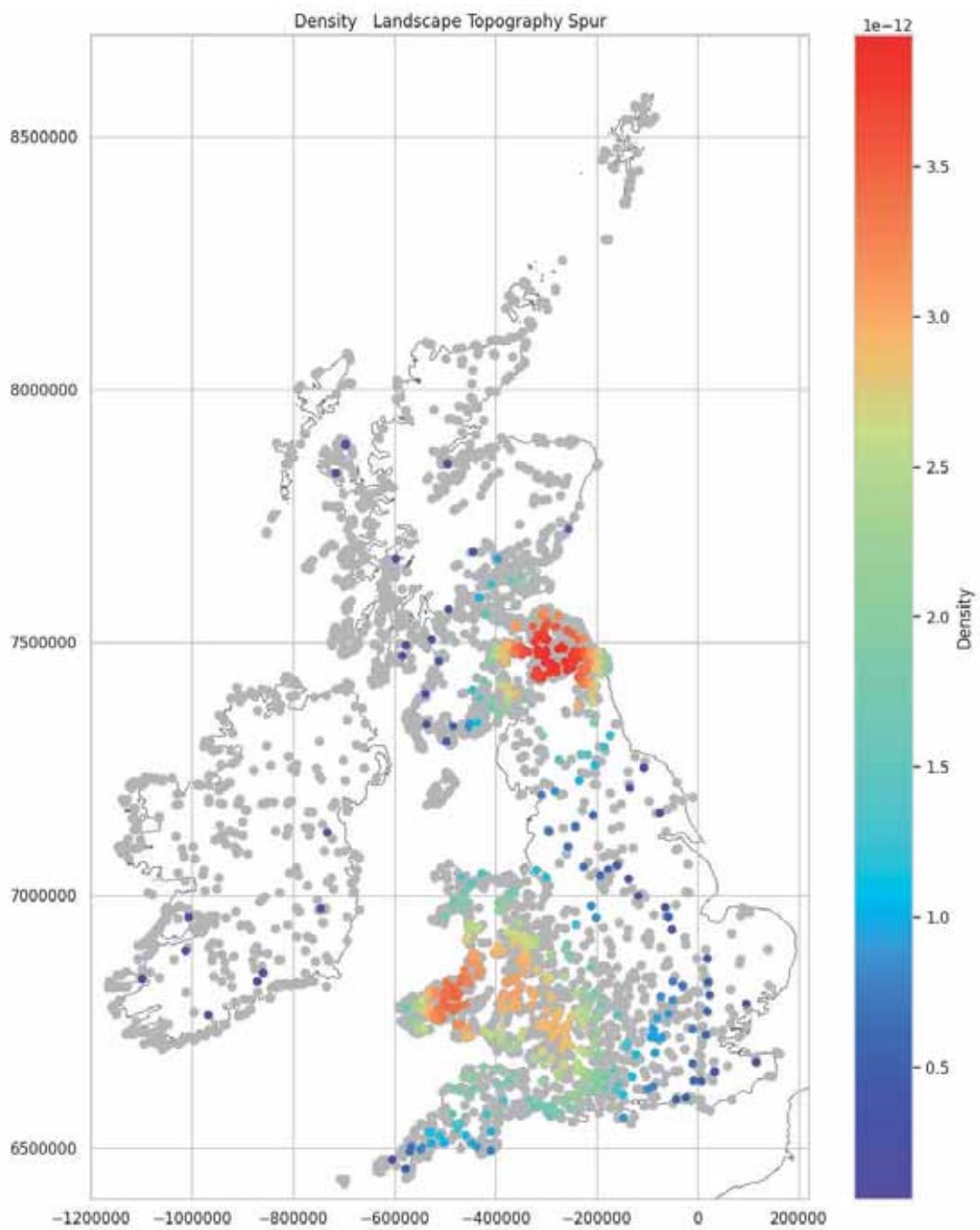
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

14. 35%

Spur Density Mapped

'Spur' density corresponds with the two main clusters in the full dataset (See Part 1: Density Map Showing Clusters Adjusted by Region). The clusters in the Northwest and the west of Ireland are not present.

```
In [ ]: plot_density_over_grey(topo_spur, 'Landscape_Topography_Spur')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect Data

This data records if a hillfort's aspect is level or toward one or more of the eight cardinal or intercardinal angles.

```
In [ ]: landscape_aspect_features = [
    'Landscape_Aspect_N',
    'Landscape_Aspect_NE',
    'Landscape_Aspect_E',
    'Landscape_Aspect_SE',
    'Landscape_Aspect_S',
    'Landscape_Aspect_SW',
    'Landscape_Aspect_W',
    'Landscape_Aspect_NW',
    'Landscape_Aspect_Level']

landscape_aspect_data = \
    landscape_data[landscape_aspect_features].copy()
landscape_aspect_data.head()
```

Out[]:	Landscape_Aspect_N	Landscape_Aspect_NE	Landscape_Aspect_E	Landscape_Aspect_SE	Landscape_Aspect_S	Landscape_Aspect_SW	Land:
0	No	No	No	No	No	No	No
1	No	No	No	No	No	No	No
2	No	No	No	No	Yes	Yes	No
3	No	No	No	No	No	No	No
4	No	No	No	No	No	No	No

There are no null values in the aspect data.

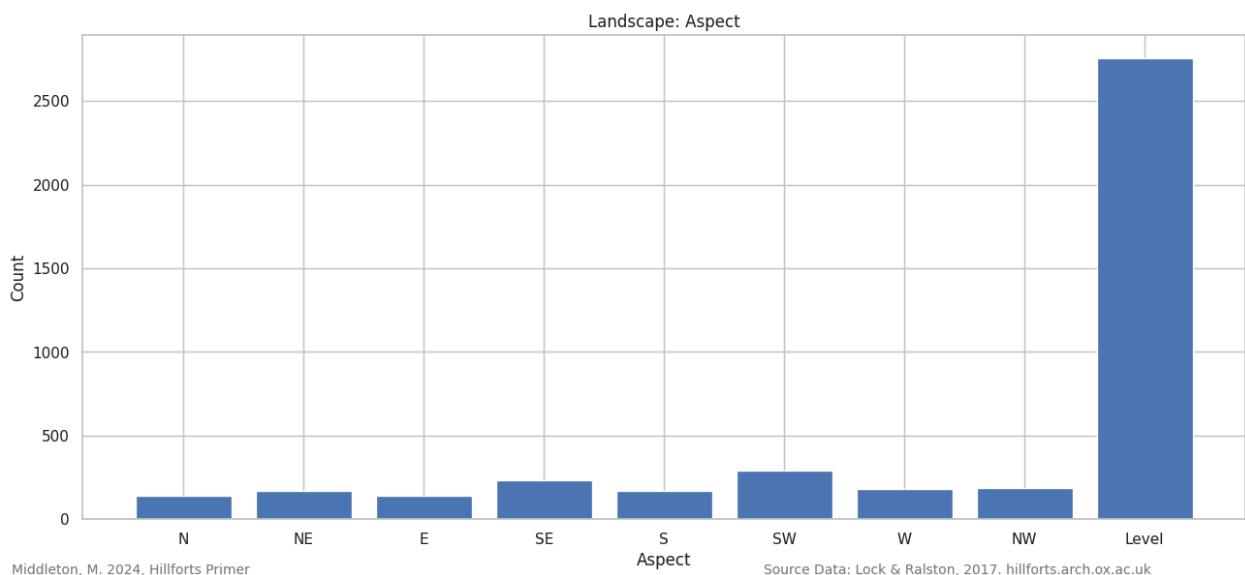
In []: `landscape_aspect_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Landscape_Aspect_N    4147 non-null   object 
 1   Landscape_Aspect_NE   4147 non-null   object 
 2   Landscape_Aspect_E    4147 non-null   object 
 3   Landscape_Aspect_SE   4147 non-null   object 
 4   Landscape_Aspect_S    4147 non-null   object 
 5   Landscape_Aspect_SW   4147 non-null   object 
 6   Landscape_Aspect_W    4147 non-null   object 
 7   Landscape_Aspect_NW   4147 non-null   object 
 8   Landscape_Aspect_Level 4147 non-null   object 
dtypes: object(9)
memory usage: 291.7+ KB
```

Aspect Data Plotted

2756 hillforts (66.46%) are recorded as being level. A small number of hillforts have alternative or additional aspect information. There is a recording bias in this data in that most sites with an aspect, other than level, are in England, Wales and Ireland. Aspects other than level have rarely been recorded in Scotland. 34 hillforts have no aspect recorded of which ten are identified as having been destroyed. The 24 extant forts without an aspect are all in England.

In []: `plot_bar_chart(landscape_aspect_data, 2, 'Aspect', 'Count', 'Landscape: Aspect')`



In []: `level_df = landscape_aspect_data[landscape_aspect_data['Landscape_Aspect_Level'] == "Yes"]
len(level_df)`

Out[]: 2756

In []: `no_aspect_data = hillforts_data.copy()
for feature in landscape_aspect_features:
 no_aspect_data = no_aspect_data[landscape_aspect_data[feature] == "No"]
len(no_aspect_data)`

```
<ipython-input-202-c24940298d19>:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.  
no_aspect_data = no_aspect_data[landscape_aspect_data[feature] == "No"]
```

```
Out[ ]: 34
```

```
In [ ]: not_destroyed_no_aspect = \
no_aspect_data[no_aspect_data['Management_Condition_Destroyed'] == "No"]
len(not_destroyed_no_aspect)
```

```
Out[ ]: 24
```

```
In [ ]: no_aspect_in_engl_and_not_destroyed =\
not_destroyed_no_aspect[not_destroyed_no_aspect['Main_Country'] == "England"]
len(no_aspect_in_engl_and_not_destroyed)
```

```
Out[ ]: 24
```

Aspect Data Spiderplot (Excluding Level)

Where an aspect other than level is recorded, there is a slight preference to the southwest. The mapped orientations are all based on very small subsets of the data. It is not clear if the orientations are a result of available local topography or if specific locations were chosen for a preferred aspect. The spider plot shows that the north, Northeast and east are the least favoured but it also shows that an aspect to the south has a similarly low count. It is likely that available local topography was the primary concern and that, if available, a south western aspect was desirable – remembering that level is, by far, the preferred choice.

```
In [ ]: landscape_aspect_data_mins = \
landscape_aspect_data.drop(['Landscape_Aspect_Level'], axis=1)
landscape_aspect_data_mins.head()
```

	Landscape_Aspect_N	Landscape_Aspect_NE	Landscape_Aspect_E	Landscape_Aspect_SE	Landscape_Aspect_S	Landscape_Aspect_SW	Landscape_Aspect_W
0	No	No	No	No	No	No	No
1	No	No	No	No	No	No	No
2	No	No	No	No	Yes	Yes	No
3	No	No	No	No	No	No	No
4	No	No	No	No	No	No	No

```
In [ ]: landscape_aspect_data_mins_reordered = landscape_aspect_data_mins[['Landscape_Aspect_E',
'Landscape_Aspect_NE',
'Landscape_Aspect_N',
'Landscape_Aspect_NW',
'Landscape_Aspect_W',
'Landscape_Aspect_SW',
'Landscape_Aspect_S',
'Landscape_Aspect_SE']].copy()
landscape_aspect_data_mins_reordered.head()
```

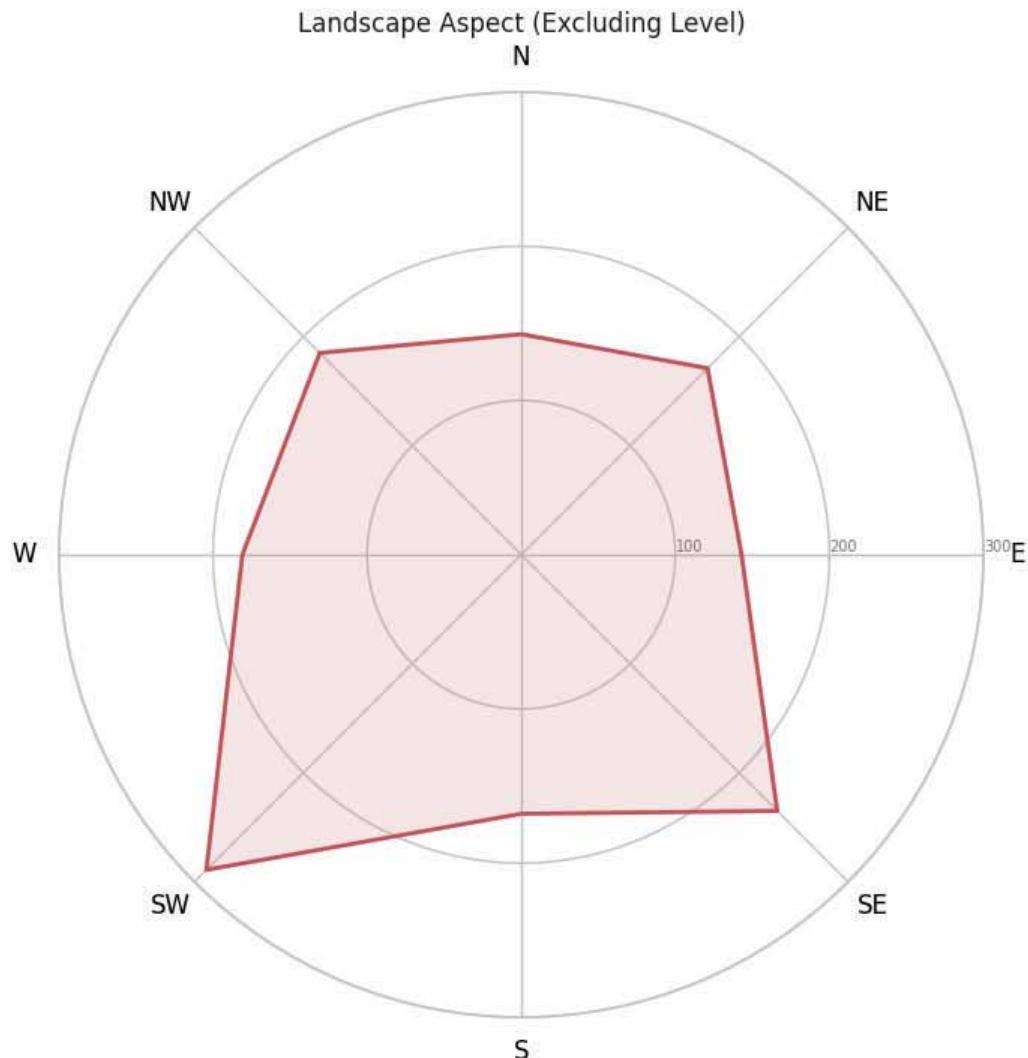
	Landscape_Aspect_E	Landscape_Aspect_NE	Landscape_Aspect_N	Landscape_Aspect_NW	Landscape_Aspect_W	Landscape_Aspect_SW	Landscape_Aspect_SE
0	No	No	No	No	No	Yes	No
1	No	No	No	No	No	No	No
2	No	No	No	No	No	No	No
3	No	No	No	No	Yes	No	No
4	No	No	No	No	No	No	No

```
In [ ]: landscape_aspect_counts = {}
landscape_aspect_counts['group'] = ['Total']
for col in landscape_aspect_data_mins_reordered.columns:
    count = \
len(landscape_aspect_data_mins_reordered[
    [landscape_aspect_data_mins_reordered[col] == 'Yes']])
    landscape_aspect_counts[col] = [count]

landscape_aspect_counts
```

```
Out[ ]: {'group': ['Total'],
 'Landscape_Aspect_E': [143],
 'Landscape_Aspect_NE': [171],
 'Landscape_Aspect_N': [143],
 'Landscape_Aspect_NW': [185],
 'Landscape_Aspect_W': [181],
 'Landscape_Aspect_SW': [289],
 'Landscape_Aspect_S': [168],
 'Landscape_Aspect_SE': [235]}
```

```
In [ ]: spider_plot(landscape_aspect_counts, 'Landscape_Aspect (Excluding Level)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Aspect Data Mapped

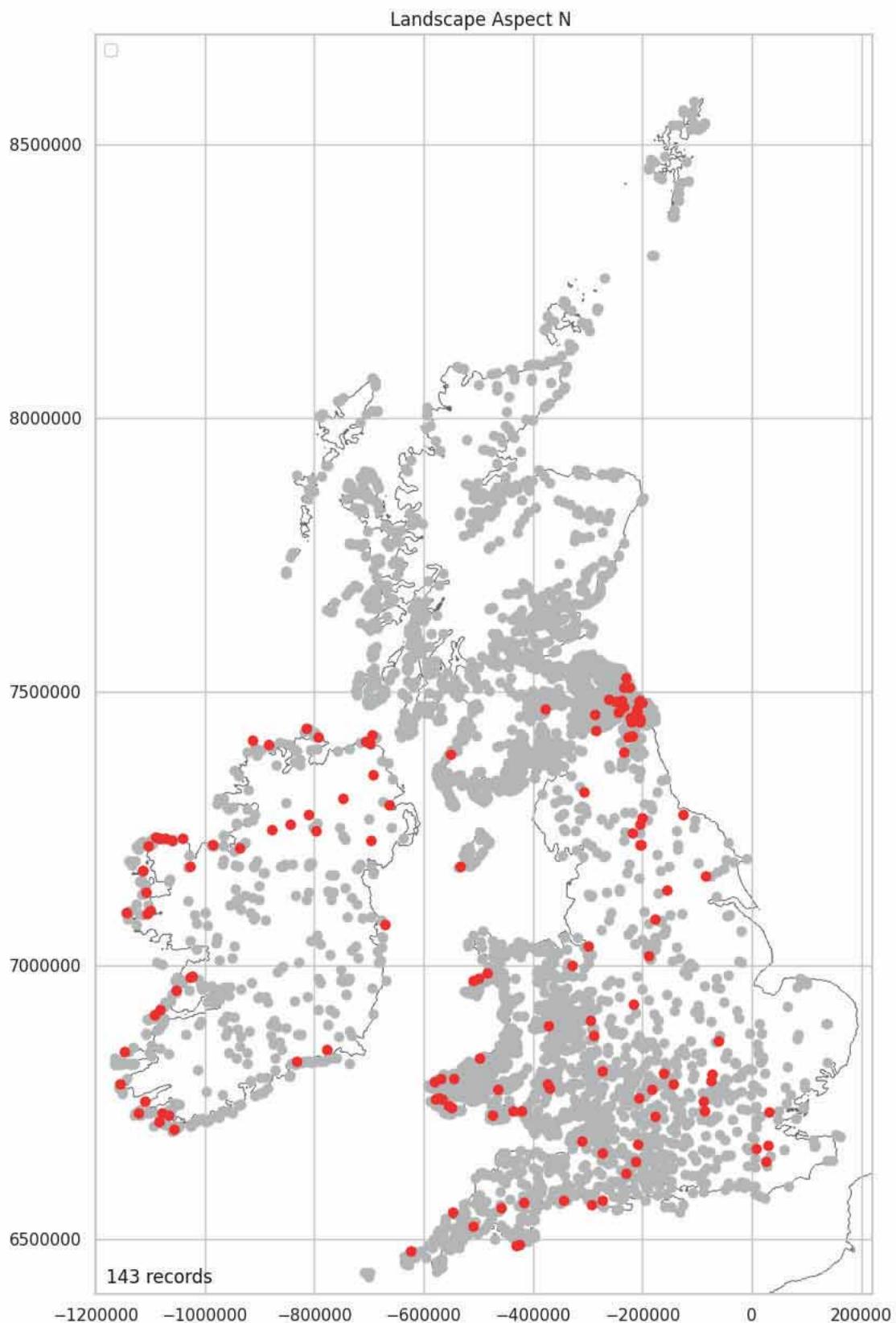
There is a recording bias in that aspects other than level have not been routinely recorded in Scotland.

```
In [ ]: location_aspect_data = \
pd.merge(location_numeric_data_short, landscape_aspect_data, \
left_index=True, right_index=True)
```

Aspect North Mapped

143 hillforts (3.45%) have an aspect facing north.

```
In [ ]: asp_n = plot_over_grey(location_aspect_data, 'Landscape_Aspect_N', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

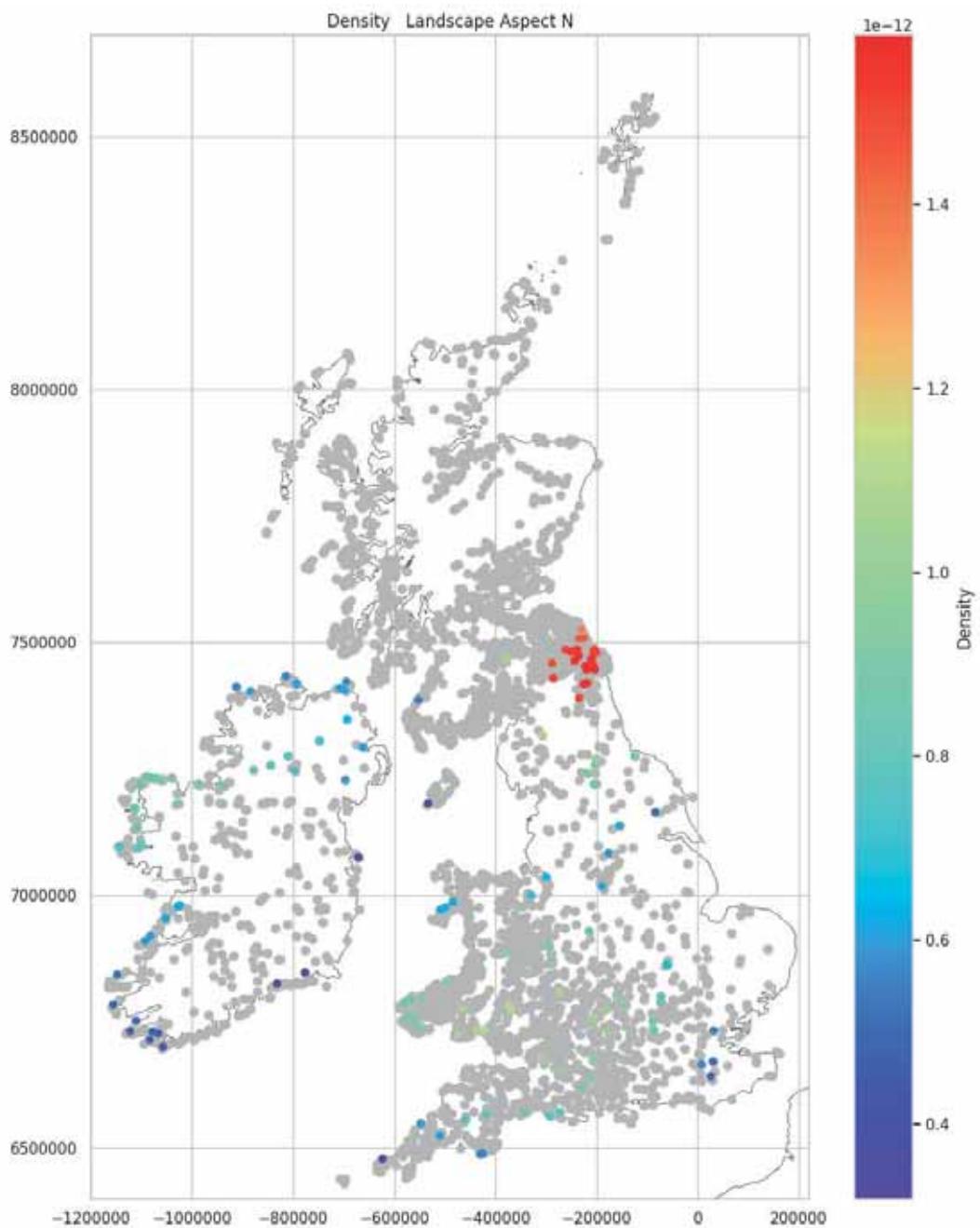
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3. 45%

Aspect Density North Mapped

The main cluster of hillforts, with a north facing aspect, are in the Southern Uplands. There is also a notable concentration of forts along the west coast of Ireland.

```
In [ ]: plot_density_over_grey(asp_n, 'Landscape_Aspect_N')
```



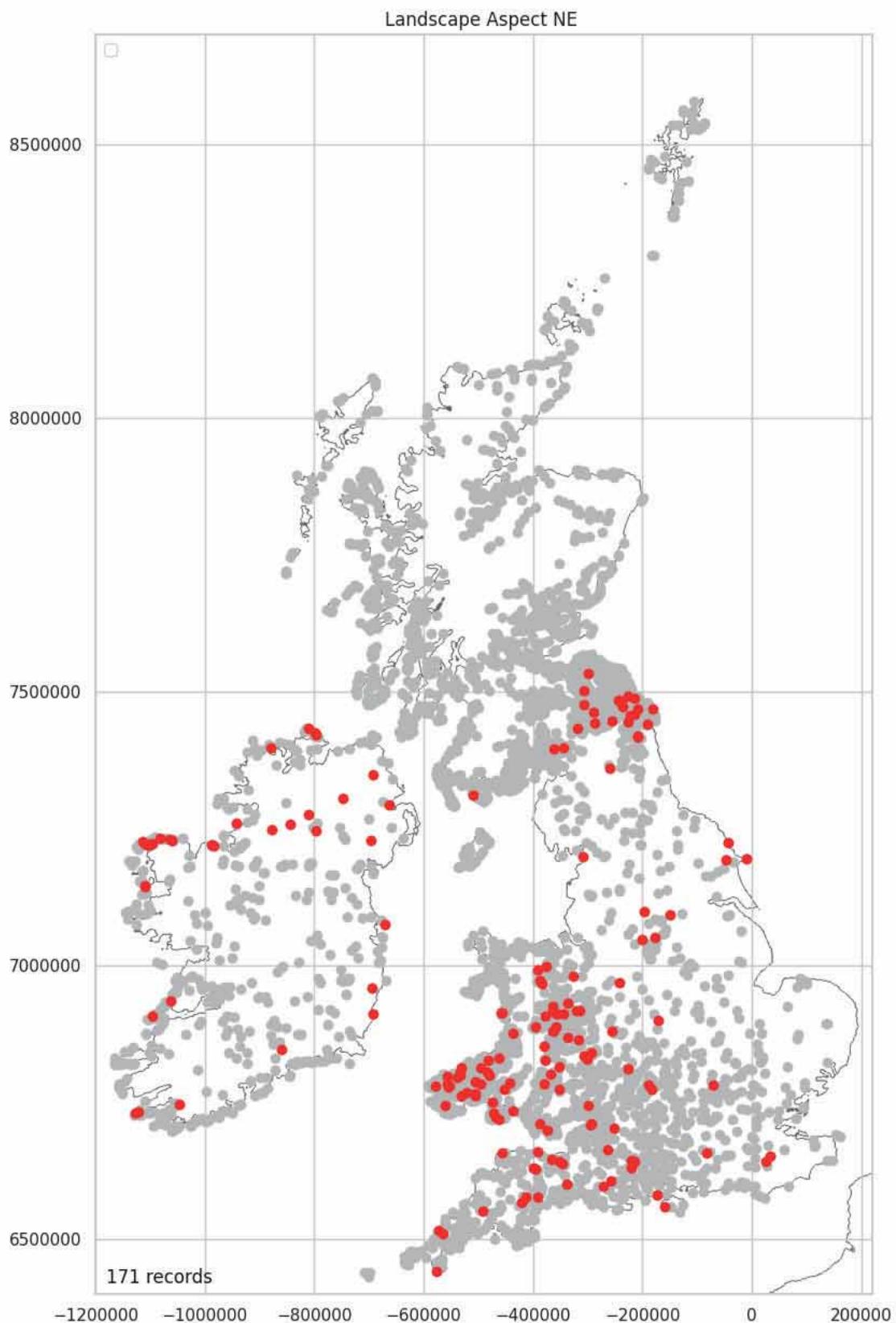
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect Northeast Mapped

171 hillforts (4.12%) have an aspect facing Northeast.

```
In [ ]: asp_ne = plot_over_grey(location_aspect_data, 'Landscape_Aspect_NE', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

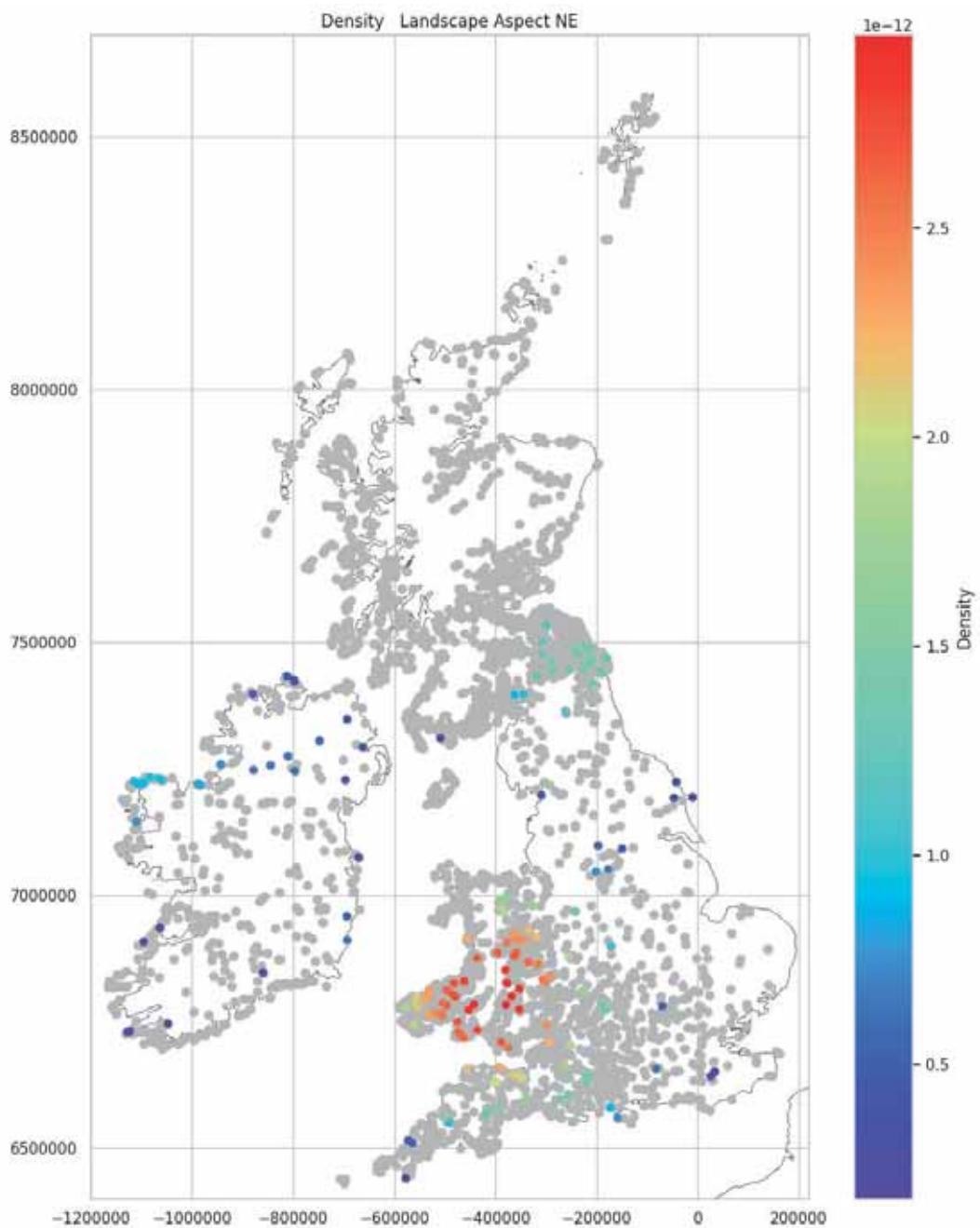
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 12%

Aspect Density Northeast Mapped

For hillforts with a Northeast aspect the focus is in the south, toward the southern end of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(asp_ne, 'Landscape_Aspect_NE')
```



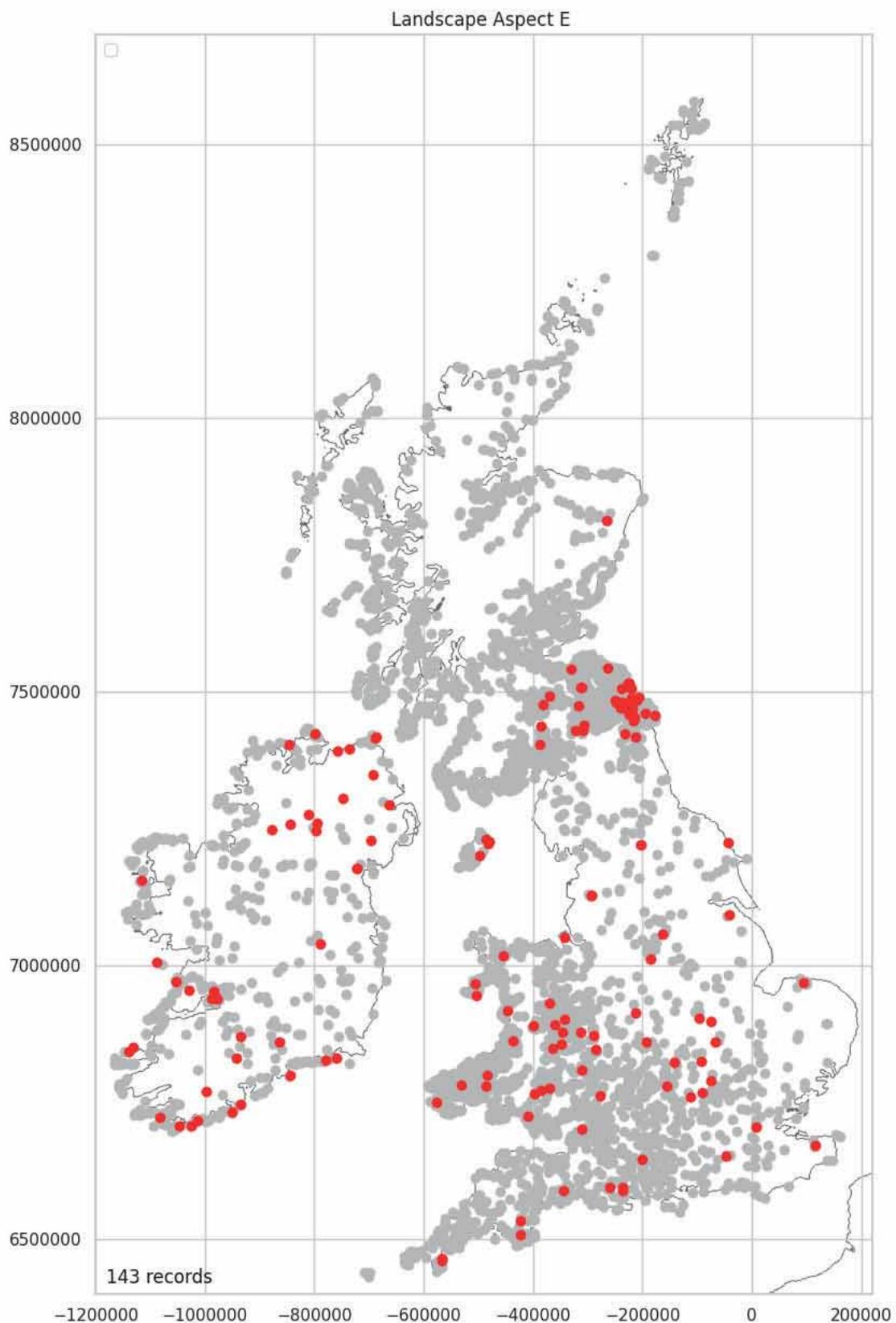
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect East Mapped

143 hillforts (3.45%) have an aspect facing east.

```
In [ ]: asp_e = \
plot_over_grey(location_aspect_data, 'Landscape_Aspect_E', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

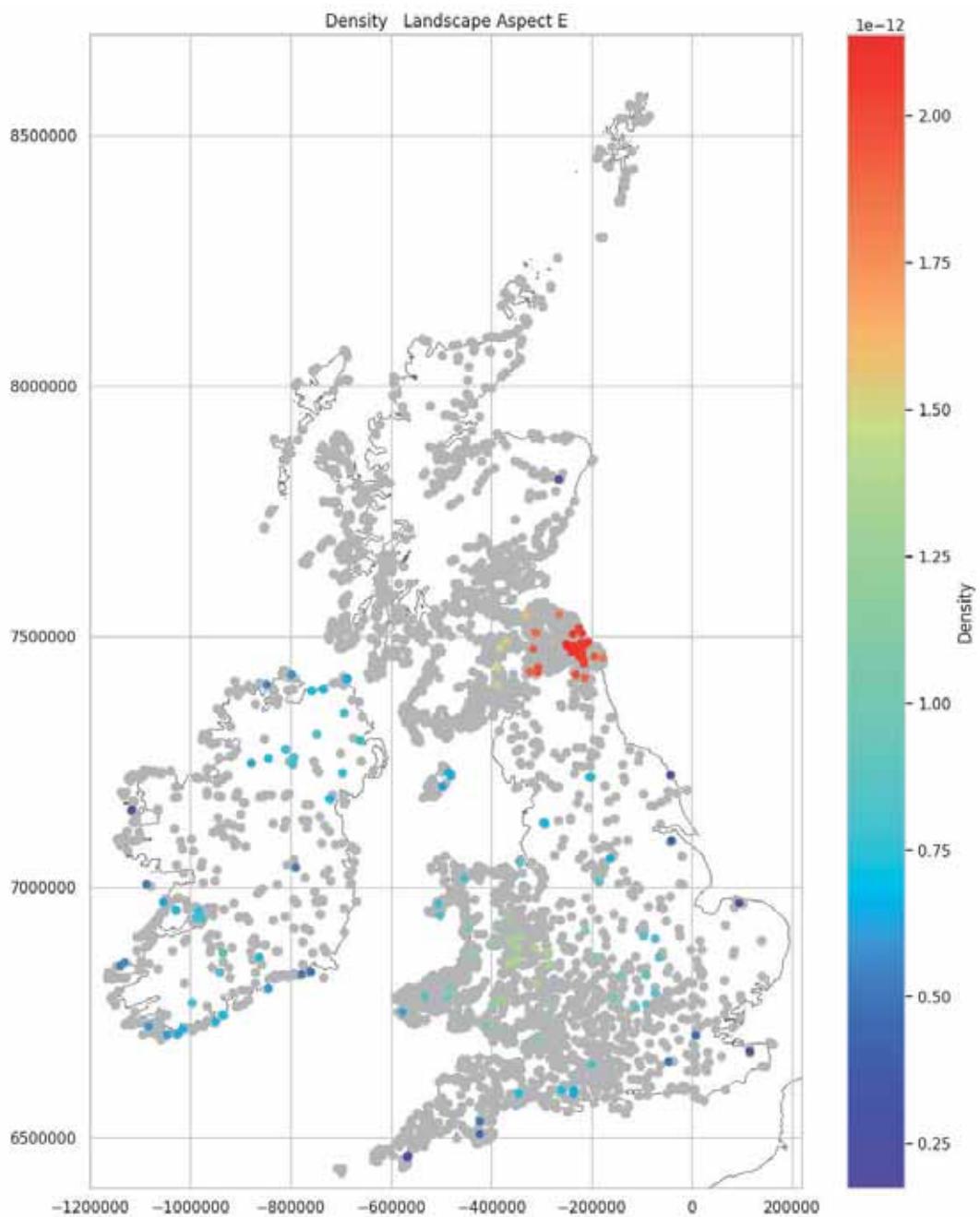
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3. 45%

Aspect Density East Mapped

For hillforts with an east facing aspect the focus is again over the Tweed Basin.

```
In [ ]: plot_density_over_grey(asp_e, 'Landscape_Aspect_E')
```



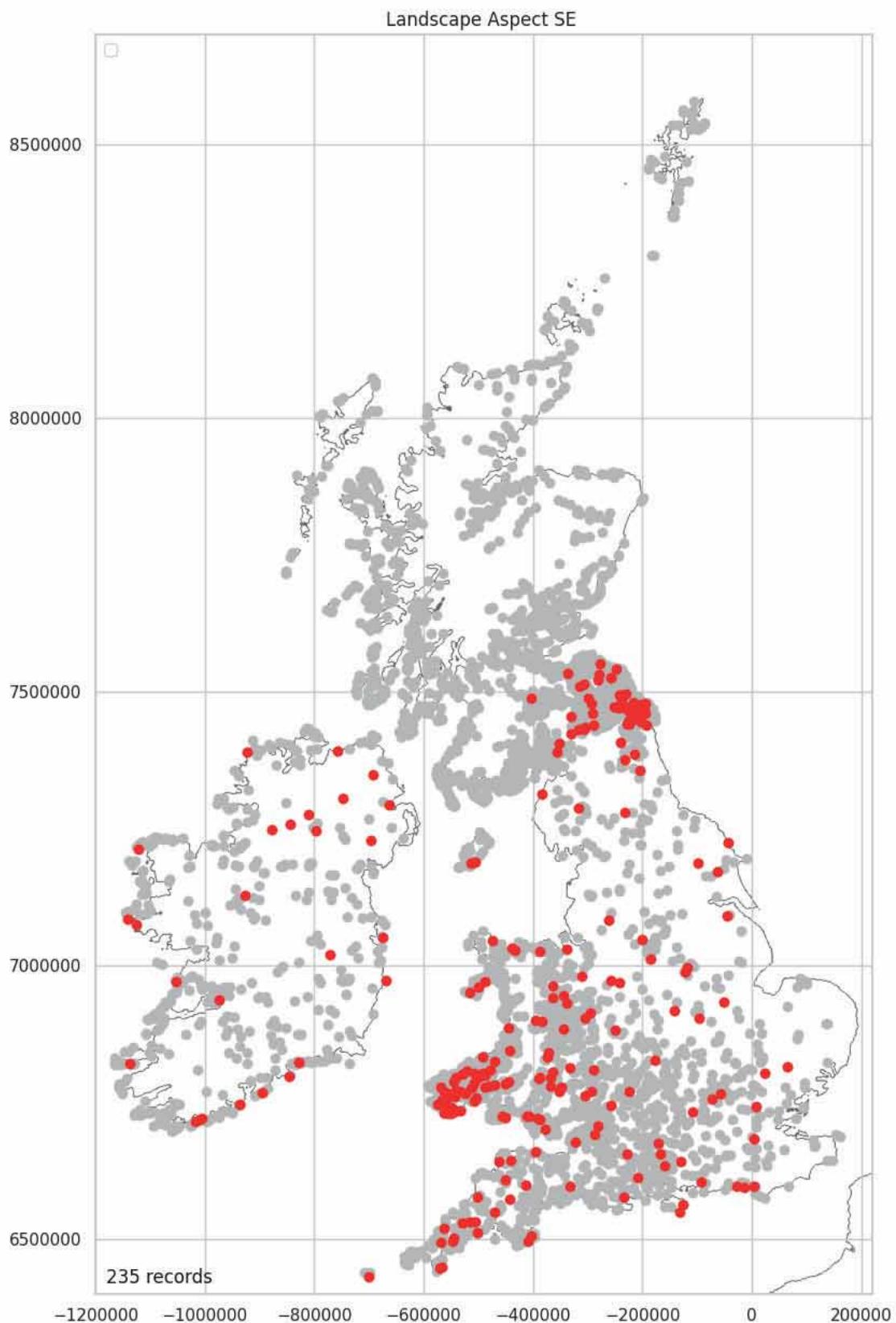
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect South East Mapped

235 hillforts (5.67%) have an aspect facing south east.

```
In [ ]: asp_se = plot_over_grey(location_aspect_data, 'Landscape_Aspect_SE', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

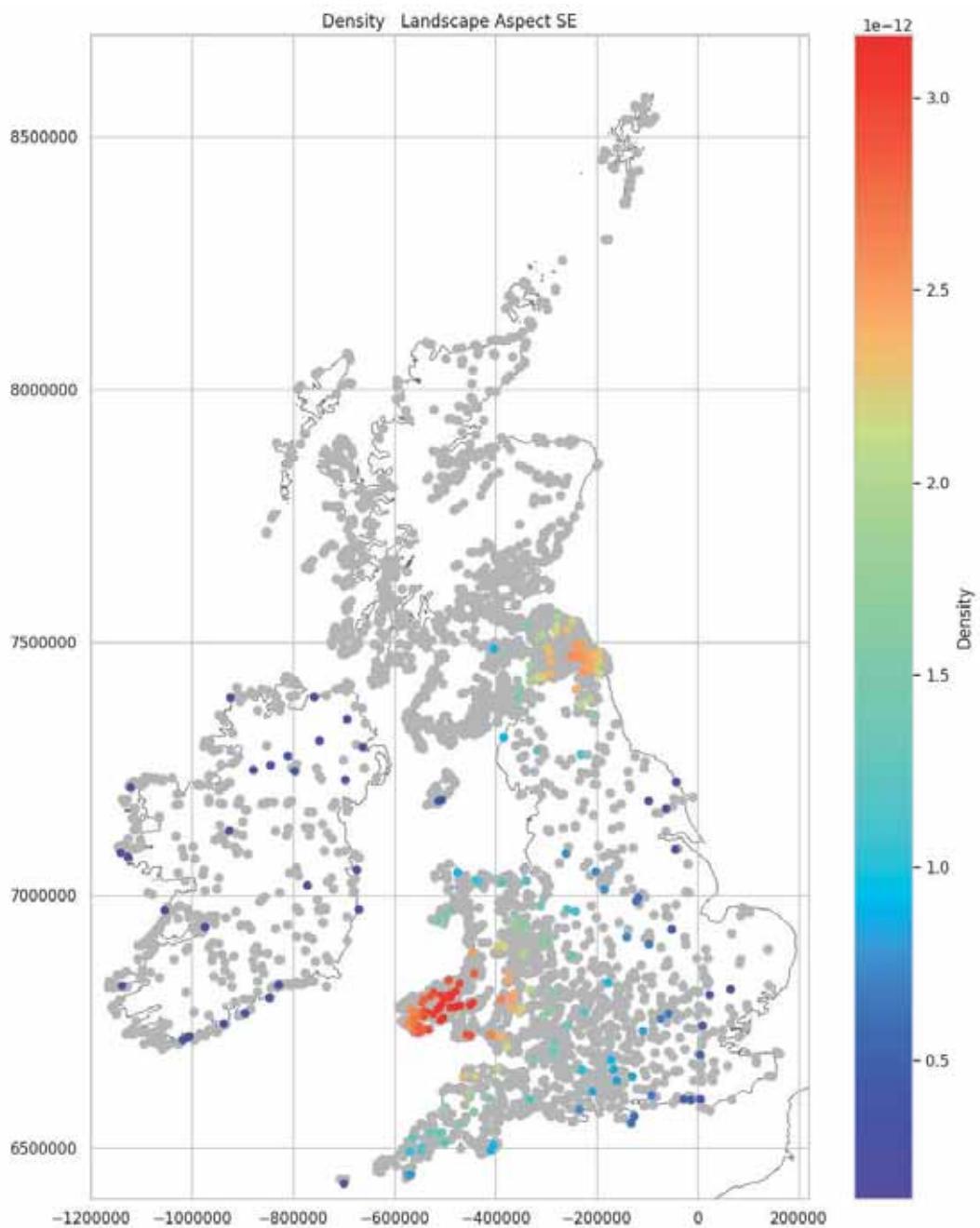
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

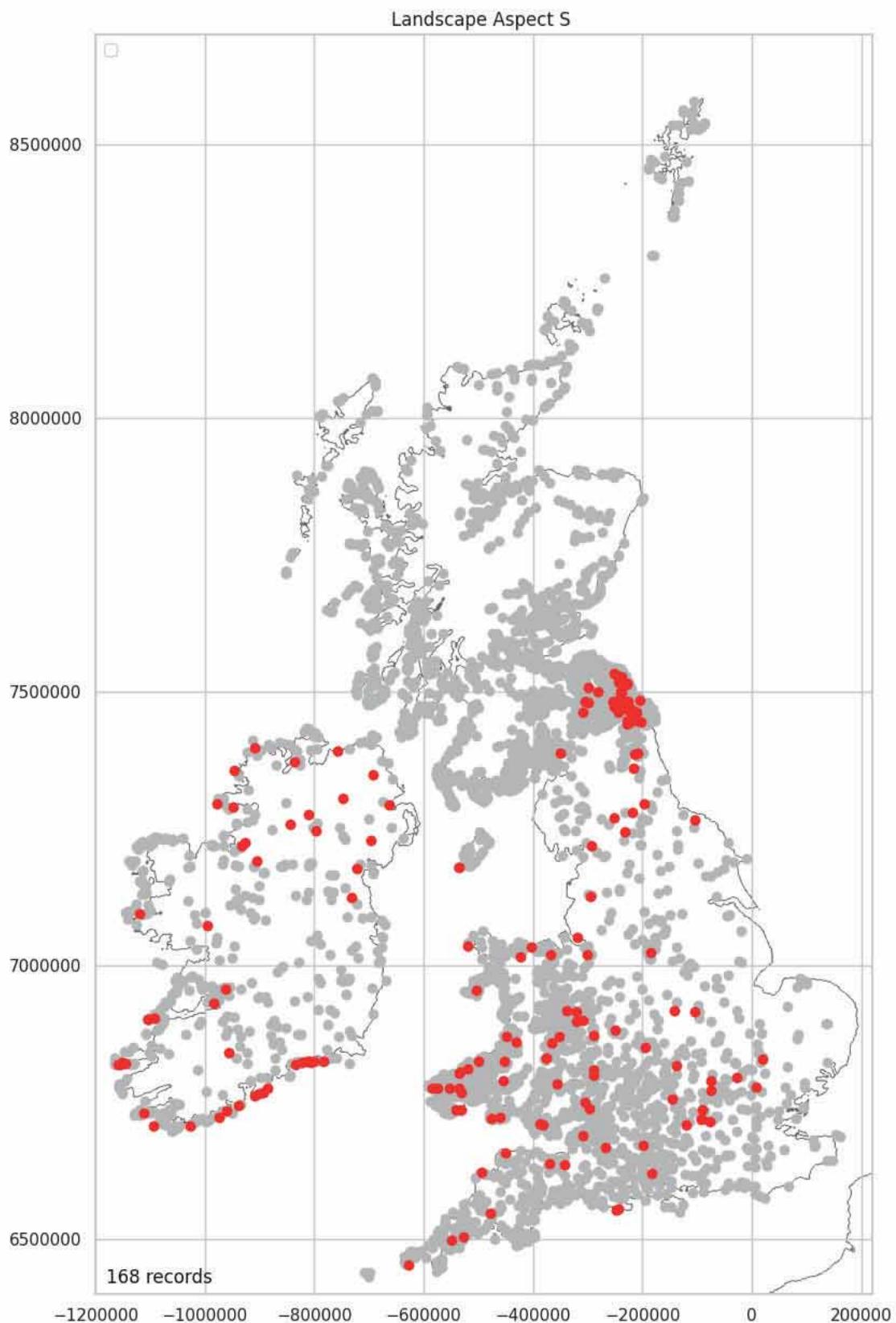
5.67%

Aspect Density South East Mapped

Hillforts with a south-eastern aspect are mostly clustered over the Pembrokeshire peninsula. There is also a moderate concentration over the Tweed Basin and the Cheviots.

```
In [ ]: plot_density_over_grey(asp_se, 'Landscape_Aspect_SE')
```





Middleton, M. 2024, Hillforts Primer

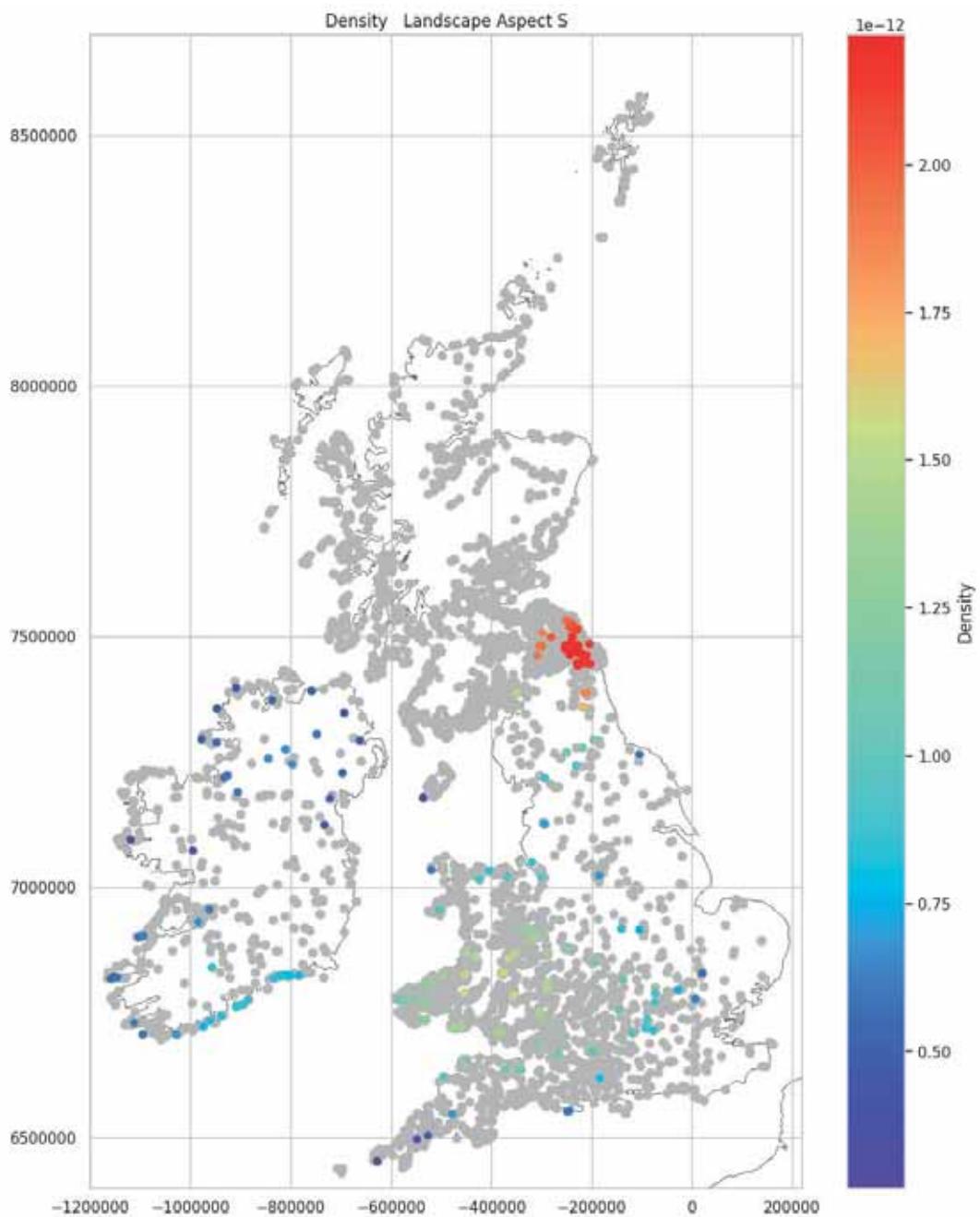
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4.05%

Aspect Density South Mapped

Hillforts, with a south facing aspect, are clustered over the Southern Uplands. There is an interesting cluster of these forts along the southern coast of Ireland suggesting local topography/geology is the main influence in this area.

```
In [ ]: plot_density_over_grey(asp_s, 'Landscape_Aspect_S')
```



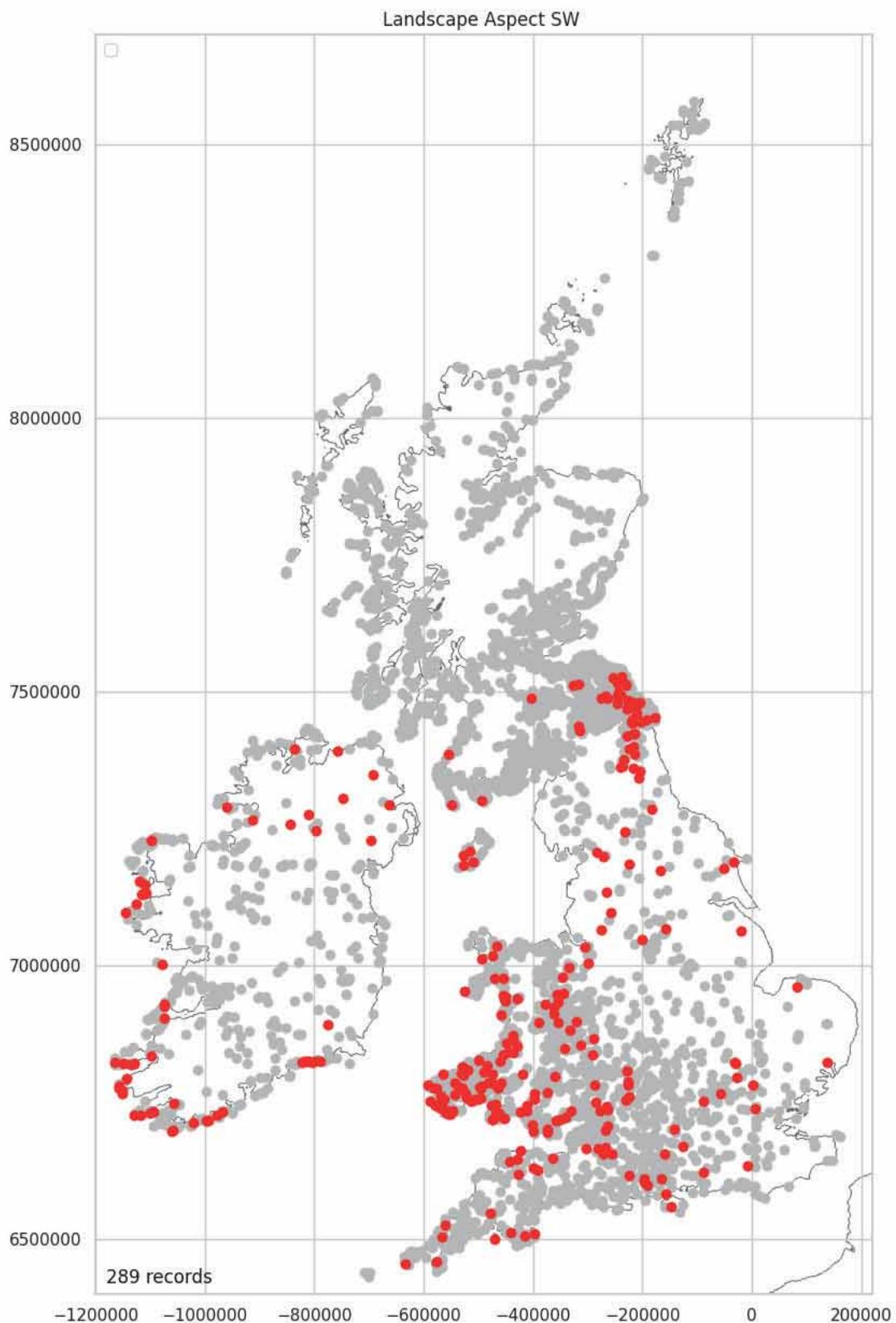
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect South West Mapped

289 hillforts (6.97%) have an aspect facing south west.

```
In [ ]: asp_sw = plot_over_grey(location_aspect_data, 'Landscape_Aspect_SW', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

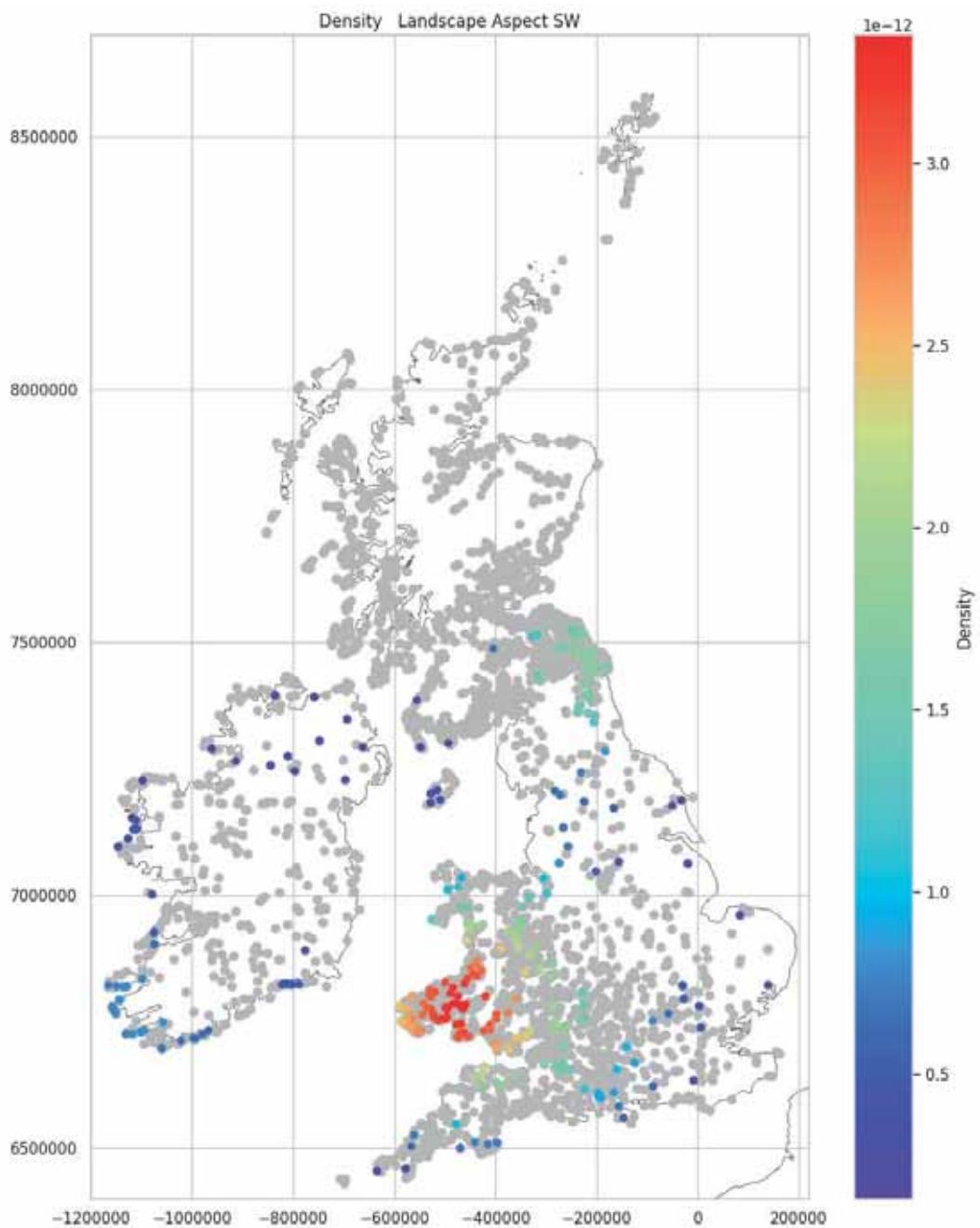
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 97%

Aspect Density South West Mapped

Hillforts with a south-western aspect are mainly clustered toward the southern end of the Cambrian Mountains and spread into the Pembrokeshire peninsula. There are small clusters over the Southern Uplands and the south-western coast of Ireland.

```
In [ ]: plot_density_over_grey(asp_sw, 'Landscape_Aspect_SW')
```



```
In [ ]: asp_w = plot_over_grey(location_aspect_data, 'Landscape_Aspect_W', 'Yes')
```

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

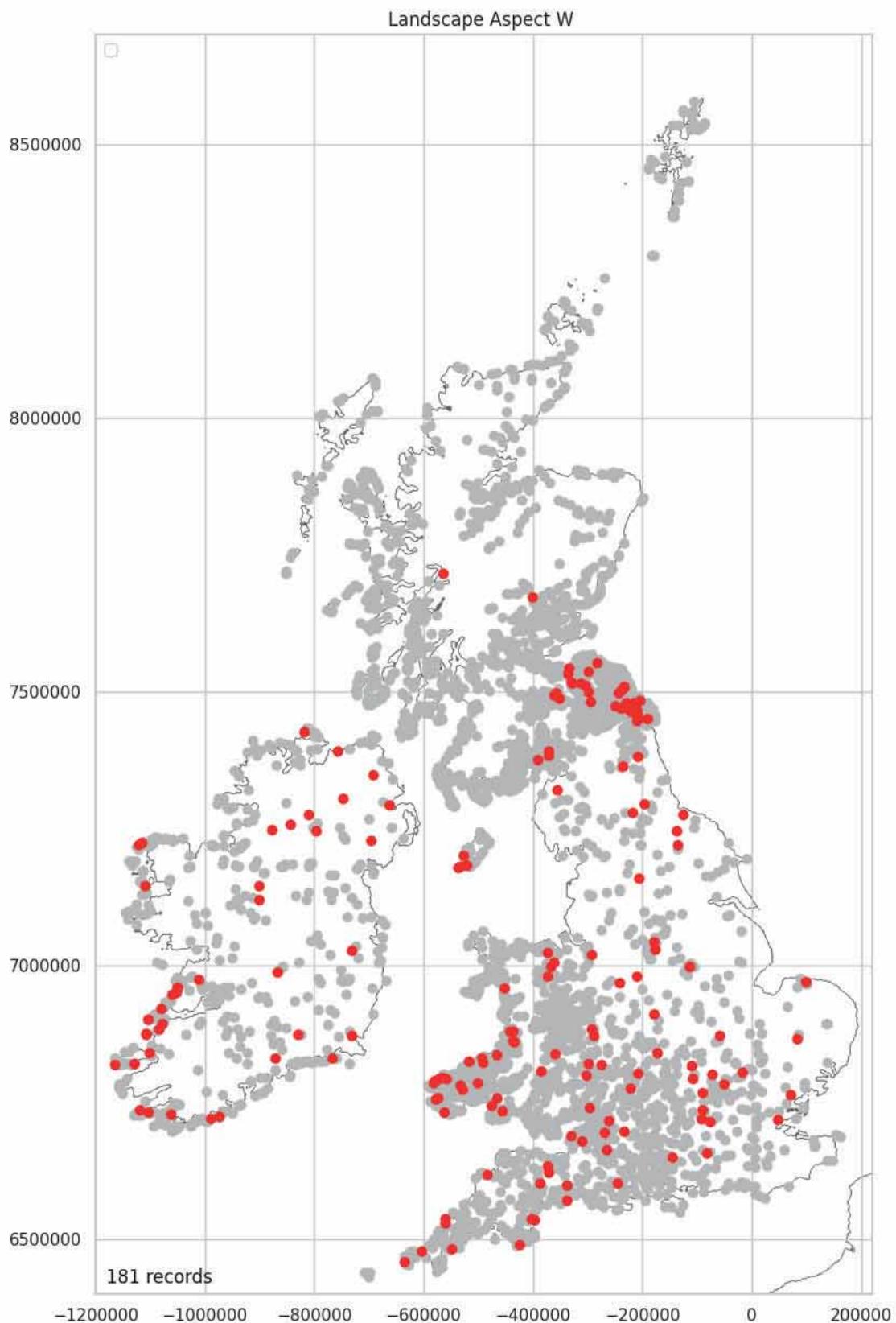
26

27

28

29

30



Middleton, M. 2024, Hillforts Primer

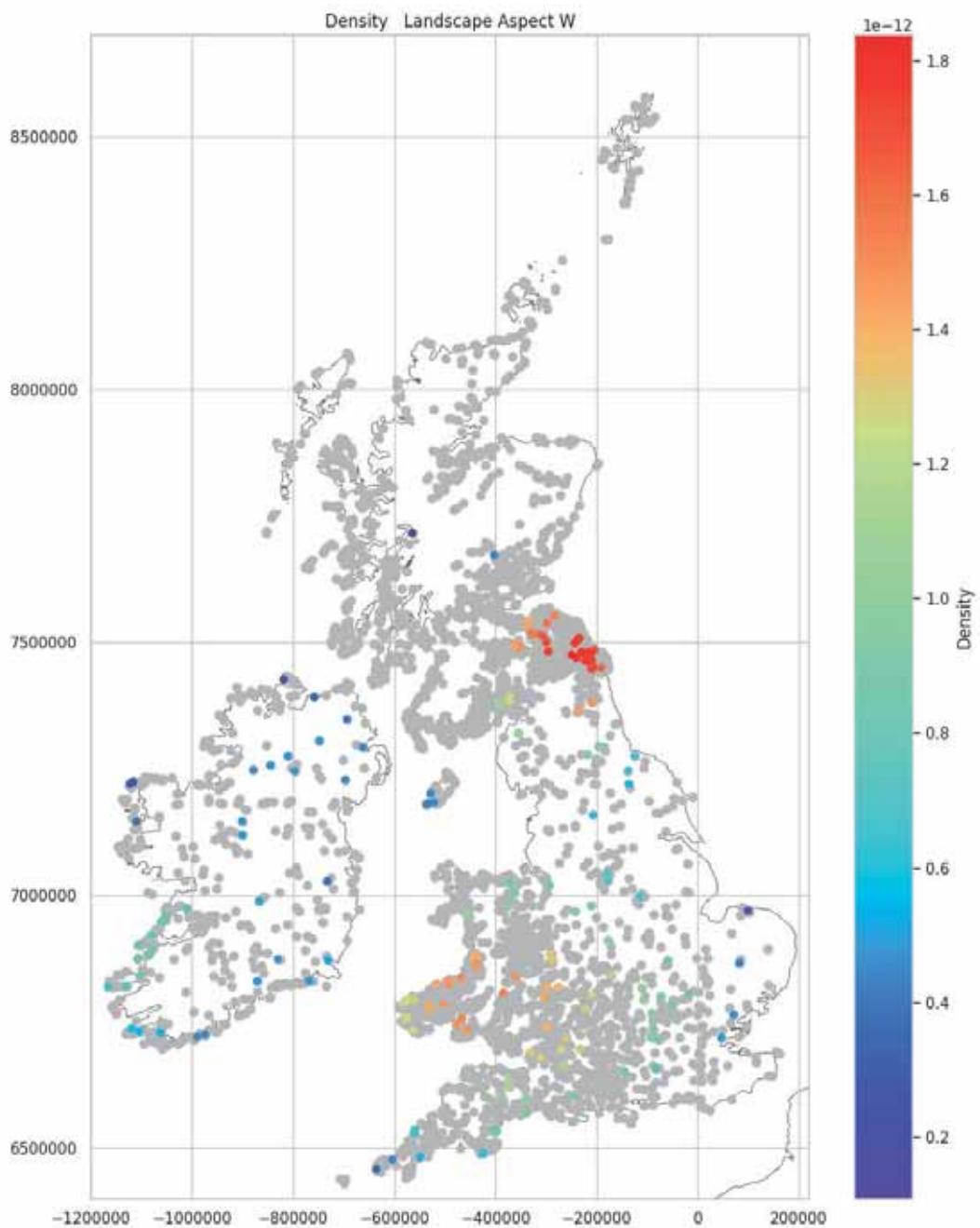
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 36%

Aspect Density West Mapped

Hillforts with a western aspect are clustered over the Tweed Basin and the Cheviots. There are also small concentrations over the southern end of the Cambrian Mountains and, along the coast, in west Ireland

```
In [ ]: plot_densitiy_over_grey(asp_w, 'Landscape_Aspect_W')
```



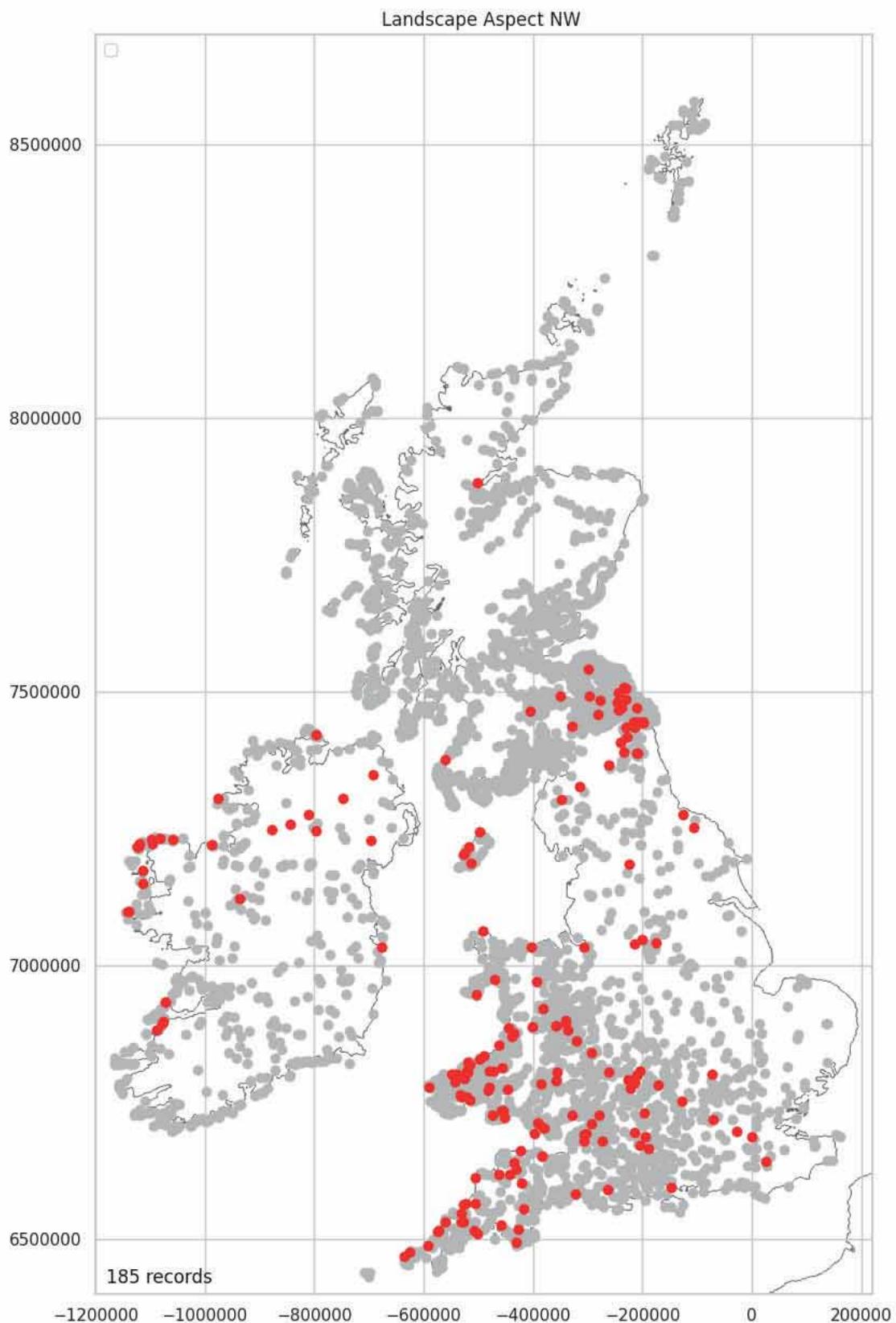
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect Northwest Mapped

185 hillforts (4.46%) have an aspect facing Northwest.

```
In [ ]: asp_nw = plot_over_grey(location_aspect_data, 'Landscape_Aspect_NW', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

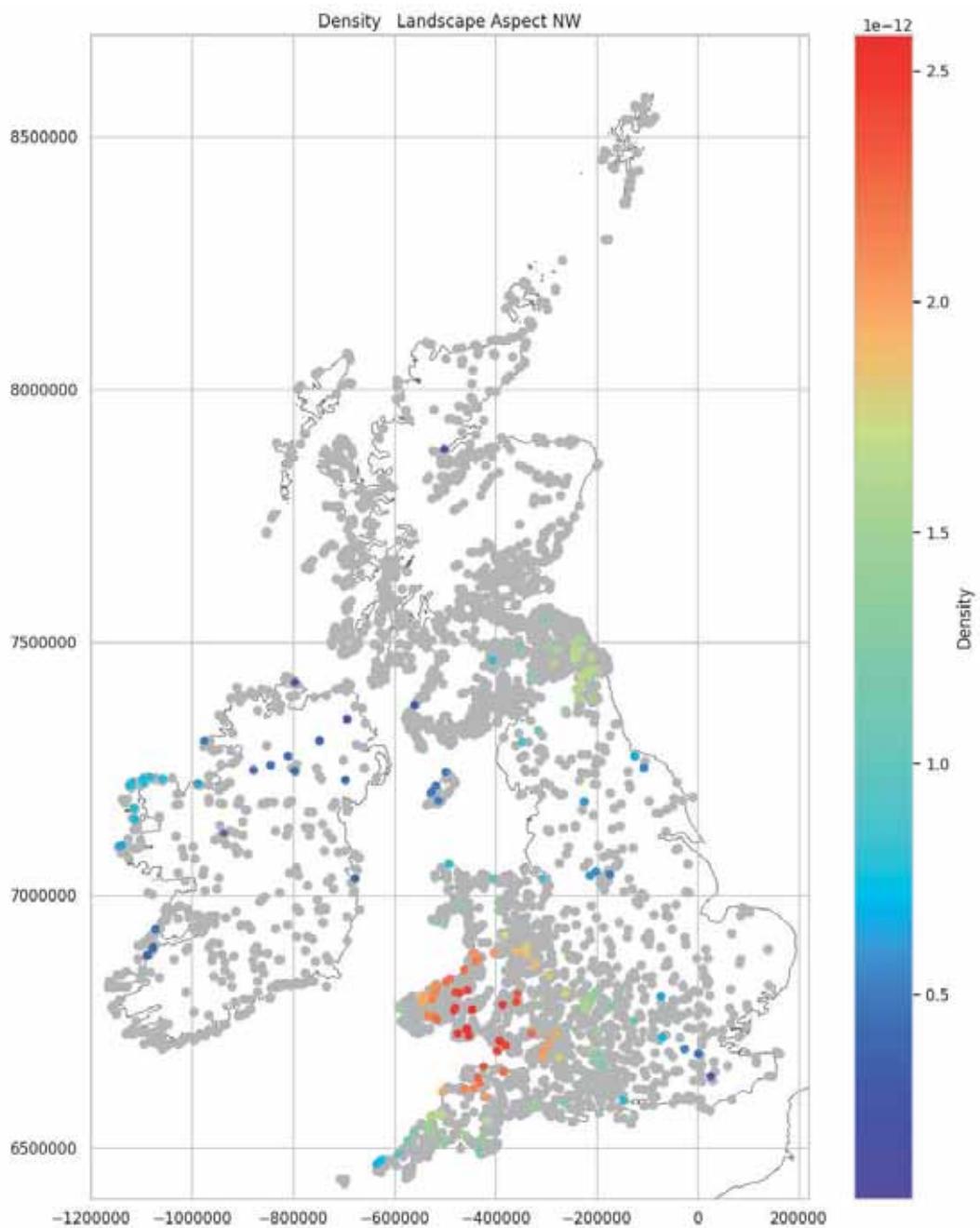
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4.46%

Aspect Density Northwest Mapped

Hillforts with a Northwestern aspect are mainly clustered over the southern end of the Cambrian Mountains. There are small clusters over the Tweed Basin and the Cheviots, and along the west coast of Ireland.

```
In [ ]: plot_densitiy_over_grey(asp_nw, 'Landscape_Aspect_NW')
```



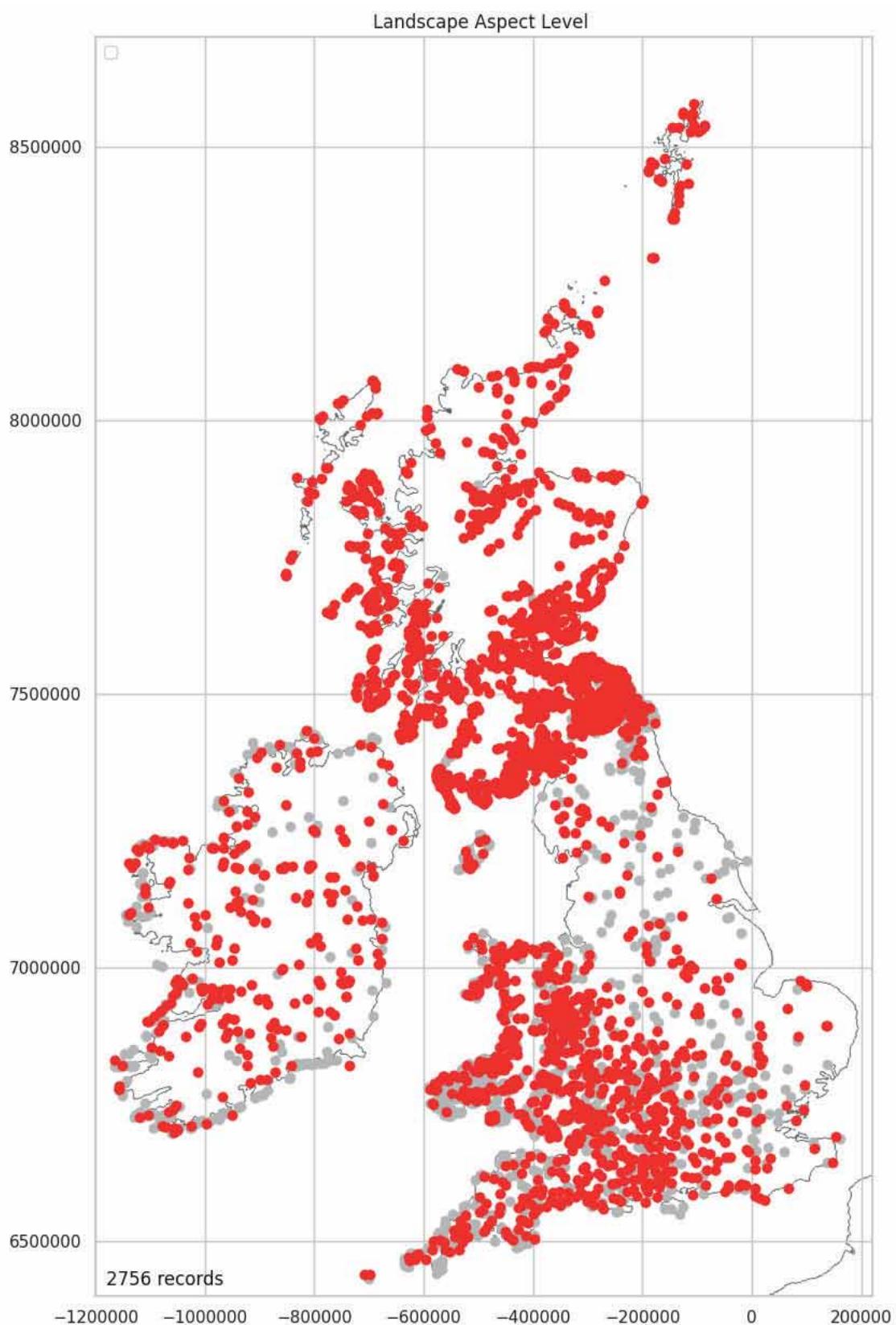
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Aspect Level Mapped

2756 hillforts (66.46%) are level. See: [Level Mapped \(Landscape\)](#).

```
In [ ]: asp_level = plot_over_grey(location_aspect_data, 'Landscape_Aspect_Level', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

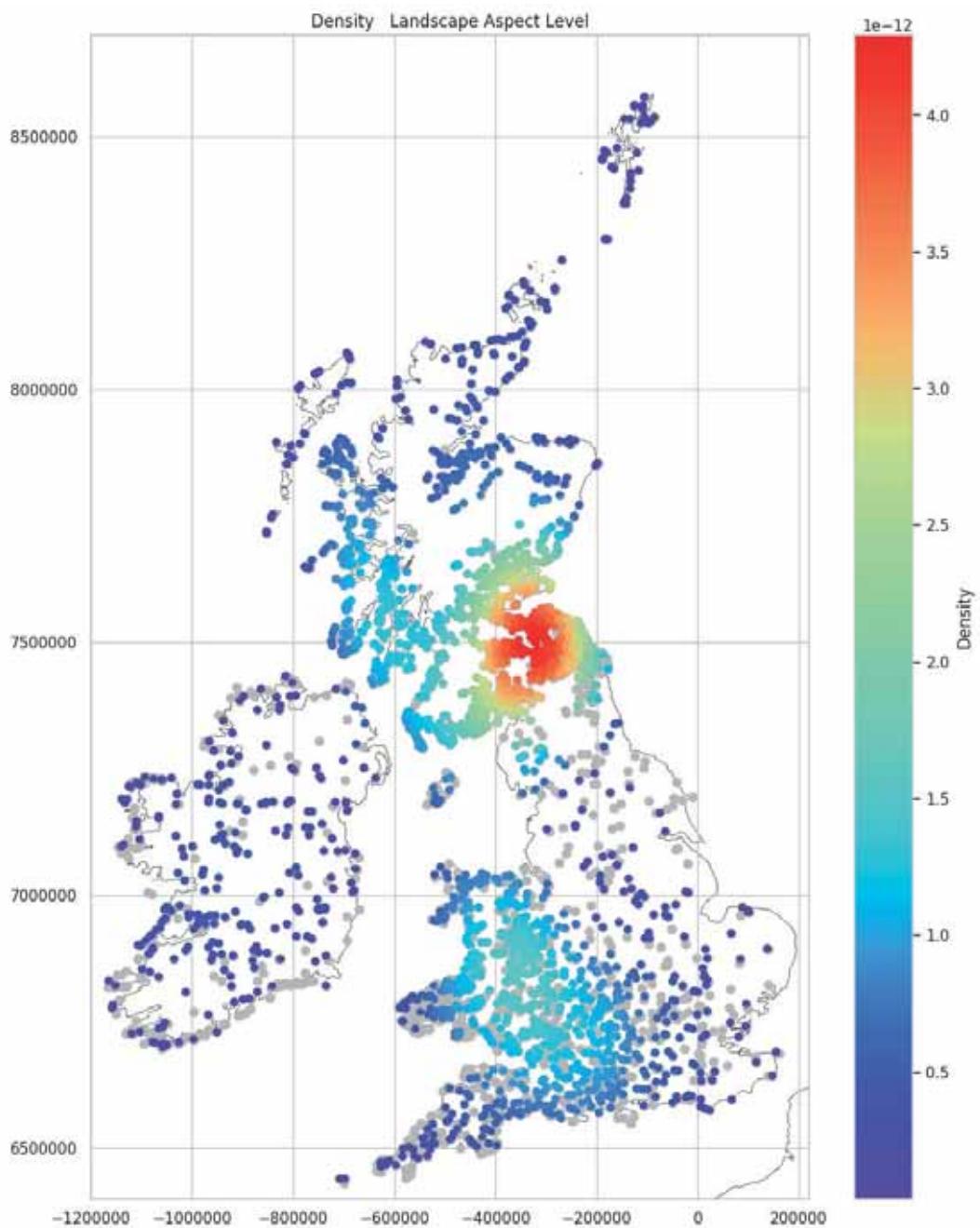
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

66.46%

Aspect Density Level Mapped

The three main clusters mirrors that seen in Part1 (Density Map Showing Clusters Adjusted by Region). The two smaller clusters in Ireland are not replicated.

```
In [ ]: plot_densitiy_over_grey(asp_level, 'Landscape_Aspect_Level')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Review Landscape Data Split

```
In [ ]: review_data_split(landscape_data, landscape_numeric_data, \
                           landscape_text_data, landscape_encodeable_data)

Data split good.
```

Landscape Data Package

```
In [ ]: landscape_data_list = [landscape_numeric_data, landscape_text_data, \
                           landscape_encodeable_data]
```

Landscape Data Download Packages

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(landscape_data_list, 'landscape_package')
```

Save Figure List

```
In [ ]: if save_images:  
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")  
    fig_list.to_csv(path, index=False)
```

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 3

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

- [Boundary Data](#)
- [Dating Data](#)

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [ ]: confirmed_only = False
```

```
In [ ]: download_data = False
```

```
In [ ]: save_images = False
```

```
In [ ]: show_topography = False
```

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Review Data Part 3](#).

Reload Data and Python Functions

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>

Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer

Licence: <https://creativecommons.org/licenses/by-sa/4.0/>

Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>

Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>

Hillforts: Britain, Ireland and the Nearer Continent (Sample):

<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Python Modules and Code Setup

```
In [ ]: import sys
print(f'Python: {sys.version}')

import sklearn
print(f'Sci Kit-Learn: {sklearn.__version__}')

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
```

```

print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

import os
import collections
import math
import random
import PIL
import urllib
random.seed(42) # A random seed is used to ensure that the random numbers
# created are the same for each run of this document.

from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive

```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
Sci kit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.25.2
matplotlib: 3.7.1
seaborn: 0.13.1
scipy: 1.11.4

Ref: <https://www.python.org/>
Ref: <https://scikit-learn.org/stable/>
Ref: <https://pandas.pydata.org/docs/>
Ref: <https://numpy.org/doc/stable/>
Ref: <https://matplotlib.org/>
Ref: <https://seaborn.pydata.org/>
Ref: <https://docs.scipy.org/doc/scipy/index.html>
Ref: <https://pypi.org/project/python-slugify/>

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

```
In [ ]: def show_records(plt, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plt.annotate(str(len(plot_data))+' records', xy=(-1180000, 6420000), \
                xycoords='data', ha='left', color=text_colour)
```

```
In [ ]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
                      "hillforts-topo-northwest-monus.png",
                      "hillforts-topo-northeast.png",
                      "hillforts-topo-south.png",
                      "hillforts-topo-south-plus.png",
                      "hillforts-topo-ireland.png",
                      "hillforts-topo-ireland-north.png",
                      "hillforts-topo-ireland-south.png"]
    else:
        backgrounds = ["hillforts-outline-01.png",
                      "hillforts-outline-north.png",
                      "hillforts-outline-northwest-plus.png",
                      "hillforts-outline-northwest-monus.png",
                      "hillforts-outline-northeast.png",
                      "hillforts-outline-south.png",
                      "hillforts-outline-south-plus.png",
                      "hillforts-outline-ireland.png"]
```

```

        "hillforts-outline-ireland-north.png",
        "hillforts-outline-ireland-south.png"]
    return backgrounds

```

```

In [ ]: def get_bounds():
    bounds = [[-1200000, 220000, 6400000, 8700000],
              [-1200000, 220000, 7000000, 8700000],
              [-1200000, -480000, 7000000, 8200000],
              [-900000, -480000, 7100000, 8200000],
              [-520000, 0, 7000000, 8700000],
              [-800000, 220000, 6400000, 7100000],
              [-1200000, 220000, 6400000, 7100000],
              [-1200000, -600000, 6650000, 7450000],
              [-1200000, -600000, 7050000, 7450000],
              [-1200000, -600000, 6650000, 7080000]]
    return bounds

```

```

In [ ]: def show_background(plt, ax, location=""):
    backgrounds = get_backgrounds()
    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDaiRSI/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDaiRSI/Hillforts-Primer/main/hillforts-topo/"

    if location == "n":
        background = os.path.join(folder, backgrounds[1])
        bounds = bounds[1]
    elif location == "nw+":
        background = os.path.join(folder, backgrounds[2])
        bounds = bounds[2]
    elif location == "nw-":
        background = os.path.join(folder, backgrounds[3])
        bounds = bounds[3]
    elif location == "ne":
        background = os.path.join(folder, backgrounds[4])
        bounds = bounds[4]
    elif location == "s":
        background = os.path.join(folder, backgrounds[5])
        bounds = bounds[5]
    elif location == "s+":
        background = os.path.join(folder, backgrounds[6])
        bounds = bounds[6]
    elif location == "i":
        background = os.path.join(folder, backgrounds[7])
        bounds = bounds[7]
    elif location == "in":
        background = os.path.join(folder, backgrounds[8])
        bounds = bounds[8]
    elif location == "is":
        background = os.path.join(folder, backgrounds[9])
        bounds = bounds[9]
    else:
        background = os.path.join(folder, backgrounds[0])
        bounds = bounds[0]

    img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
    ax.imshow(img, extent=bounds)

```

```

In [ ]: def get_counts(data):
    data_counts = []
    for col in data.columns:
        count = len(data[data[col] == 'Yes'])
        data_counts.append(count)
    return data_counts

```

```

In [ ]: def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.01), \
                xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [ ]: def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.035), \
                xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [ ]: def plot_bar_chart(data, split_pos, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))

```

```

ax = fig.add_axes([0, 0, 1, 1])
x_data = data.columns
x_data = [x.split("_")[split_pos:] for x in x_data]
x_data_new = []
for l in x_data :
    txt = ""
    for part in l:
        txt += "_" + part
    x_data_new.append(txt[1:])
y_data = get_counts(data)
ax.bar(x_data_new, y_data)
ax.set_xlabel(x_label)
ax.set_ylabel(y_label)
add_annotation_plot(ax)
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

```

In [ ]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, n_bins):
    new_data = data.copy()
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    data[x_label].plot(kind='hist', bins = n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_value_counts(data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    df = data.value_counts()
    x_data = df.index.values
    y_data = df.values
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def get_bins(data, bins_count):
    data_range = data.max() - data.min()
    print(bins_count)
    if bins_count != None:
        x_bins = [x for x in range(data.min(), data.max(), bins_count)]
        n_bins = len(x_bins)
    else:
        n_bins = int(data_range)
        if n_bins < 10:
            multi = 10
        while n_bins < 10:
            multi *= 10
            n_bins = int(data_range * multi)
    elif n_bins > 100:
        n_bins = int(data_range)/10

    return n_bins

```

```

In [ ]: def plot_histogram(data, x_label, title, bins_count = None):
    n_bins = get_bins(data, bins_count)
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    ax.set_ylabel('Count')
    plt.ticklabel_format(style='plain')
    plt.hist(data, bins=n_bins)
    plt.title(get_print_title(title))
    add_annotation_plot(ax)

```

```
save_fig(title)
plt.show()
```

```
In [ ]: def plot_continuous(data, x_label, title):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    plt.plot(data, linewidth=4)
    plt.title_format(style='plain')
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [ ]: # box plot
from matplotlib.cbook import boxplot_stats
def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.title_format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v")
    else:
        sns.boxplot(data=data, orient="h")
    save_fig(feature + " Range")
    plt.show()

bp = boxplot_stats(data)

low = bp[0].get('whislo')
q1 = bp[0].get('q1')
median = bp[0].get('med')
q3 = bp[0].get('q3')
high = bp[0].get('whishi')

return [low, q1, median, q3, high]
```

```
In [ ]: def location_XY_plot():
    plt.title_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 8700000)
    add_annotation_l_xy(plt)
```

```
In [ ]: def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')
```

```
In [ ]: def plot_over_grey_numeric(merged_data, a_type, title, extra="", inner=False, \
                           fringe=False, oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    fig, ax = plt.subplots(figsize=(9.47, 15.33))
```

```

show_background(plt, ax)
location_XY_plot()
add_grey(region=' ')
plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
show_records(plt, plot_data)
plt.title(get_print_title('Boundary_Type: ' + boundary_type))
save_fig('Boundary_Type_' + boundary_type)
plt.show()
print(f'{round((len(plot_data)/len(merged_data)*100), 2)}%')

```

```

In [ ]: def plot_density_over_grey(data, data_type):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_density(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
               c=new_data['Density'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density')
    plt.title(get_print_title(f'Density - {data_type}'))
    save_fig(f'Density_{data_type}')
    plt.show()

```

```

In [ ]: def add_21Ha_line():
    x_values = \
        [-367969, -344171, -263690, -194654, -130542, -119597, -162994, -265052], -304545]
    y_values = \
        [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 6663609], 6611780]
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, \
              label = '≥ 21 Ha Line')
    add_to_legend = \
        Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, label = '≥ 21 Ha Line')
    return add_to_legend

```

```

In [ ]: def add_21Ha_fringe():
    x_values = \
        [-367969, -126771, 29679, -42657, -248650, -304545, -423647, -584307, -367969]
    y_values = \
        [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, 6752378, 7019842]
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    add_to_legend = \
        Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, label = '≥ 21 Ha Fringe')
    return add_to_legend

```

```

In [ ]: def add_oxford_swindon(oxford=False, swindon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_size = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_size, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swindon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_size, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches

```

```

In [ ]: def plot_over_grey(merged_data, a_type, yes_no, extra="", \
                           inner=False, fringe=False, oxford=False, swindon=False):
    # plots selected data over the grey dots. yes_no controls filtering
    # the data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()

```

```
print(f' {round((len(plot_data)/len(merged_data)*100), 2)}%')
return plot_data
```

```
In [ ]: def plot_type_values(data, data_type, title):
    new_data = data.copy()
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plot, ax)
    location_XY_plot()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
                c=new_data[data_type], cmap=cm.rainbow, s=25)
    plt.colorbar(label=data_type)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def bespoke_plot(plot, title):
    add_annotation_plot(plot)
    plt.ticker.label_format(style='plain')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [ ]: def get_proportions(date_set):
    total = sum(date_set) - date_set[-1]
    newset = []
    for entry in date_set[:-1]:
        newset.append(round(entry/total, 2))
    return newset
```

```
In [ ]: def plot_dates_by_region(nw, ne, ni, si, s, features):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = nw[features].columns
    x_data = [x.split("_")[-2] for x in x_data][:-1]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])

    set1_name = 'NW'
    set2_name = 'NE'
    set3_name = 'N Ireland'
    set4_name = 'S Ireland'
    set5_name = 'South'
    set1 = get_proportions(get_counts(nw[features]))
    set2 = get_proportions(get_counts(ne[features]))
    set3 = get_proportions(get_counts(ni[features]))
    set4 = get_proportions(get_counts(si[features]))
    set5 = get_proportions(get_counts(s[features]))

    X_axis = np.arange(len(x_data_new))

    budge = 0.25

    plt.bar(X_axis - 0.55 + budge, set1, 0.3, label = set1_name)
    plt.bar(X_axis - 0.4 + budge, set2, 0.3, label = set2_name)
    plt.bar(X_axis - 0.25 + budge, set3, 0.3, label = set3_name)
    plt.bar(X_axis - 0.1 + budge, set4, 0.3, label = set4_name)
    plt.bar(X_axis + 0.05 + budge, set5, 0.3, label = set5_name)

    plt.xticks(X_axis, x_data_new)
    plt.xlabel('Dating')
    plt.ylabel('Proportion of Total Dated Hillforts in Region')
    title = 'Proportions of Dated Hillforts by Region'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

```
In [ ]: def test_numeric(data):
    temp_data = data.copy()
    columns = data.columns
    out_cols = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
    feat, ent, num, non, nul = [], [], [], [], []
    for col in columns:
```

```

    if temp_data[col].dtype == 'object':
        feat.append(col)
        temp_data[col+'_num'] = temp_data[col].str.isnumeric()
        entries = temp_data[col].notnull().sum()
        true_count = temp_data[col+'_num'][temp_data[col+'_num'] == True].sum()
        null_count = temp_data[col].isna().sum()
        ent.append(entries)
        num.append(true_count)
        non.append(entries-true_count)
        nul.append(null_count)
    else:
        print(f'{col} {temp_data[col].dtype}')
summary = pd.DataFrame([list(zip(feat, ent, num, non, nul))])
summary.columns = out_cols
return summary

```

```

In [ ]: def find_duplicated(numeric_data, text_data, encodeable_data):
d = False
all_columns = \
list(numeric_data.columns) + list(text_data.columns) + \
list(encodeable_data.columns)
duplicate = \
[item for item, count in collections.Counter(all_columns).items() \
if count > 1]
if duplicate:
    print(f"There are duplicate features: {duplicate}")
    d = True
return d

```

```

In [ ]: def test_data_split(main_data, numeric_data, text_data, encodeable_data):
m = False
split_features = \
list(numeric_data.columns) + list(text_data.columns) + \
list(encodeable_data.columns)
missing = list(set(main_data)-set(split_features))
if missing:
    print(f"There are missing features: {missing}")
    m = True
return m

```

```

In [ ]: def review_data_split(main_data, numeric_data, text_data, \
                           encodeable_data = pd.DataFrame()):
d = find_duplicated(numeric_data, text_data, encodeable_data)
m = test_data_split(main_data, numeric_data, text_data, encodeable_data)
if d != True and m != True:
    print("Data split good.")

```

```

In [ ]: def find_duplicates(data):
print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')

```

```

In [ ]: def count_yes(data):
total = 0
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    total += count
    print(f'{col}: {count}')
print(f'Total yes count: {total}')

```

Null Value Functions

The following functions will be used to update null values.

```

In [ ]: def fill_nan_with_minus_one(data, feature):
new_data = data.copy()
new_data[feature] = data[feature].fillna(-1)
return new_data

```

```

In [ ]: def fill_nan_with_NA(data, feature):
new_data = data.copy()
new_data[feature] = data[feature].fillna("NA")
return new_data

```

```

In [ ]: def test_numeric_value_in_feature(feature, value):
test = feature.isin([-1]).sum()
return test

```

```

In [ ]: def test_categorical_value_in_feature(dataframe, feature, value):
test = dataframe[feature][dataframe[feature] == value].count()
return test

```

```
In [ ]: def test_cat_list_for_NA(dataframe, cat_list):
    for val in cat_list:
        print(val, test_categorical_value_in_feature(dataframe, val, 'NA'))

In [ ]: def test_num_list_for_minus_one(dataframe, num_list):
    for val in num_list:
        feature = dataframe[val]
        print(val, test_numeric_value_in_feature(feature, -1))

In [ ]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data

In [ ]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data
```

Reprocessing Functions

```
In [ ]: def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data
```

Save Image Functions

```
In [ ]: fig_no = 0
part = 'Part03'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"
```

```
In [ ]: def get_file_name(title):
    file_name = slugify(title)
    return file_name
```

```
In [ ]: def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(".", ";")
    return title
```

```
In [ ]: def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no
```

```
In [ ]: if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass
```

```
In [ ]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Part_03_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution, bbox_inches='tight')
```

```

else:
    pass

```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [ ]: hillforts_csv = r"https://raw.githubusercontent.com/MikeDarsie/Hillforts-Primer/main/hillforts-atlas-source-data.csv"
#r"https://raw.githubusercontent.com/MikeDarsie/Hillforts-Primer/main/hillforts-atlas-source-data-csv/hillforts.csv"
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)

<ipython-input-55-03f8272c1daf>: 4: DtypeWarning: Columns (10, 12, 68, 83, 84, 85, 86, 165, 183) have mixed types. Specify dtype option on import or set low_memory=False.
    hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

```
Out[ ]:   OBJECTID Main_Atlas_Number Main_Country_Code Main_Country Main_Title_Name Main_Site_Name Main_Alt_Name Main_Display_N
          0           1                 1             EN       England      EN0001 Aconbury
                                         Camp, Herefordshire      Aconbury Camp      Aconbury Beacon      Aconbury C
                                         Hereford (Aconbury Bea
          1           2                 2             EN       England      EN0002 Bach
                                         Camp, Herefordshire      Bach Camp        NaN      Bach C
                                         Hereford
```

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [ ]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
                      'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.')

Using all 4147 record in the Hillforts Atlas.
```

Download Function

```
In [ ]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', \
                               'republic-of-ireland', 'northern-ireland', \
                               'isle-of-man', 'roi-ni', 'eng-wal-sco-iom']:
                if pkg.shape[0] == hillforts_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
                else:
                    dl = data_list[0]
            dl = dl.replace('\r', ' ', regex=True)
            dl = dl.replace('\n', ' ', regex=True)
            fn = 'hillforts_primer_' + filename
            fn = get_file_name(fn)
            dl.to_csv(fn+'.csv', index=False)
            files.download(fn+'.csv')
    else:
        pass
```

Reload Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-data

packages. Where needed, these data extracts will be combined with Name and Number features to ensure the data can be understood and can, if needed, be concorded.

```
In [ ]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Reload Location

```
In [ ]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short)
location_numeric_data_short.head()
location_data = location_numeric_data_short.copy()
location_data.head()
```

	Location_X	Location_Y	Density
0	-303295	6798973	1.632859e-12
1	-296646	6843289	1.540172e-12
2	-289837	6808611	1.547729e-12
3	-320850	6862993	1.670548e-12
4	-261765	6810587	1.369981e-12

Reload Location Cluster Data Packages

```
In [ ]: cluster_data = \
hillforts_data[['Location_X', 'Location_Y', 'Main_Country_Code']].copy()
cluster_data['Cluster'] = 'NA'
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
'NI', 'I', inplace=True)
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
'IR', 'I', inplace=True)

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000) , \
    'North_Ireland', cluster_data['Cluster'])
north_ireland = cluster_data[cluster_data['Cluster'] == 'North_Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , \
    'South_Ireland', cluster_data['Cluster'])
south_ireland = cluster_data[cluster_data['Cluster'] == 'South_Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] < 7070000) , \
    'South', cluster_data['Cluster'])
south = cluster_data[cluster_data['Cluster'] == 'South'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] >= -500000), 'Northeast', cluster_data['Cluster'])
north_east = cluster_data[cluster_data['Cluster'] == 'Northeast'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] < -500000), 'Northwest', cluster_data['Cluster'])
north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()

temp_cluster_location_packages = \
[north_ireland, south_ireland, south, north_east, north_west]

cluster_packages = []
for pkg in temp_cluster_location_packages:
    pkg = pkg.drop(['Main_Country_Code'], axis=1)
    cluster_packages.append(pkg)
```

```
north_i_rel_and, south_i_rel_and, south, north_east, north_west = \
cluster_packages[0], cluster_packages[1], cluster_packages[2], \
cluster_packages[3], cluster_packages[4]
```

Review Data Part 3

Boundary Data

The boundary data contains eight features.

```
In [ ]: boundary_features = [
    'Boundary_Boundary_Type',
    'Boundary_Boundary_Comments',
    'Boundary_Country_Code_2',
    'Boundary_HER_2',
    'Boundary_HER_PRN_2',
    'Boundary_Current_County_2',
    'Boundary_Historic_County_2',
    'Boundary_Current_Parish_2']

boundary_data = forts_data[boundary_features].copy()
boundary_data.head()
```

```
Out[ ]:   Boundary_Boundary_Type  Boundary_Boundary_Comments  Boundary_Country_Code_2  Boundary_HER_2  Boundary_HER_PRN_2  Boundary_Current_County_2  Boundary_Historic_County_2  Boundary_Current_Parish_2
0           NaN                   NaN                   NaN           NaN           NaN           NaN           NaN           NaN
1           NaN                   NaN                   NaN           NaN           NaN           NaN           NaN           NaN
2           NaN                   NaN                   NaN           NaN           NaN           NaN           NaN           NaN
3           NaN                   NaN                   NaN           NaN           NaN           NaN           NaN           NaN
4          County                 NaN                   NaN           NaN  Worcestershire           NaN           NaN           NaN
```

The boundary data is partial. Five of the features contain 20 records or less. There is so little data in these features that their distributions are not useful. These five features will be dropped from the reprocessed download.

```
In [ ]: boundary_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Boundary_Boundary_Type    428 non-null   object 
 1   Boundary_Boundary_Comments 259 non-null   object 
 2   Boundary_Country_Code_2    7 non-null    object 
 3   Boundary_HER_2            12 non-null   object 
 4   Boundary_HER_PRN_2        10 non-null   object 
 5   Boundary_Current_County_2 20 non-null   object 
 6   Boundary_Historic_County_2 17 non-null   object 
 7   Boundary_Current_Parish_2  360 non-null  object 
dtypes: object(8)
memory usage: 259.3+ KB
```

Boundary Numeric Data

There is no Boundary numeric data.

```
In [ ]: boundary_numeric_data = pd.DataFrame()
```

Boundary Text Data

There is a single boundary text feature.

```
In [ ]: boundary_text_features = [
    'Boundary_Boundary_Comments']

boundary_text_data = boundary_data[boundary_text_features].copy()
boundary_text_data[boundary_text_data['Boundary_Boundary_Comments'].notna()].head()
```

```
Out[ ]: Boundary_Boundary_Comments
```

- 9 Part of the site is located in Shropshire and ...
- 40 Part in Shropshire and part in Wales (Powys).
- 70 Part of site in Wales (Powys), part in England...
- 95 Formerly bisected by the historic counties of ...
- 96 Although situated entirely in Thatcham it lies...

Boundary Text Data - Resolve Null Values

Test for 'NA'.

```
In [ ]: test_cat_list_for_NA(boundary_text_data, boundary_text_features)
```

Boundary_Boundary_Comments 0

Fill null values with 'NA'.

```
In [ ]: boundary_text_data = \  
update_cat_list_for_NA(boundary_text_data, boundary_text_features)  
boundary_text_data.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 1 columns):
 # Column Non-Null Count Dtype
 --- --
 0 Boundary_Boundary_Comments 4147 non-null object
dtypes: object(1)
memory usage: 32.5+ KB

Boundary Encodable Data

There are seven Boundary encodable features. As mentioned above, five of these features contain so little data they will be dropped from the download. Before being dropped, the data will be plotted.

```
In [ ]: boundary_encodeable_features = [  
'Boundary_Boundary_Type',  
'Boundary_Country_Code_2',  
'Boundary_HER_2',  
'Boundary_HER_PRN_2',  
'Boundary_Current_County_2',  
'Boundary_Historic_County_2',  
'Boundary_Current_Parish_2']  
  
boundary_encodeable_data = boundary_data[boundary_encodeable_features].copy()  
boundary_encodeable_data.head()
```

```
Out[ ]: Boundary_Boundary_Type  Boundary_Country_Code_2  Boundary_HER_2  Boundary_HER_PRN_2  Boundary_Current_County_2  Boundary_Hi  
0  NaN  NaN  NaN  NaN  NaN  NaN  
1  NaN  NaN  NaN  NaN  NaN  NaN  
2  NaN  NaN  NaN  NaN  NaN  NaN  
3  NaN  NaN  NaN  NaN  NaN  NaN  
4  County  NaN  Worcestershire  WSM00932  Worcestershire
```

```
In [ ]: boundary_encodeable_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Boundary_Boundary_Type    428 non-null   object  
 1   Boundary_Country_Code_2   7 non-null    object  
 2   Boundary_HER_2            12 non-null   object  
 3   Boundary_HER_PRN_2        10 non-null   object  
 4   Boundary_Current_County_2 20 non-null   object  
 5   Boundary_Historic_County_2 17 non-null   object  
 6   Boundary_Current_Parish_2  360 non-null  object  
dtypes: object(7)
memory usage: 226.9+ KB

```

Boundary Encodeable Data - Resolve Null Values

All features in this dataset contain null values. 'NA' is not currently present in any feature and will be used to replace null values.

```
In [ ]: test_cat_list_for_NA(boundary_encodeable_data, boundary_encodeable_features)
```

```

Boundary_Boundary_Type 0
Boundary_Country_Code_2 0
Boundary_HER_2 0
Boundary_HER_PRN_2 0
Boundary_Current_County_2 0
Boundary_Historic_County_2 0
Boundary_Current_Parish_2 0

```

Null values updated to 'NA'.

```
In [ ]: boundary_encodeable_data = \
update_cat_list_for_NA(boundary_encodeable_data, boundary_encodeable_features)
```

```
In [ ]: boundary_encodeable_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Boundary_Boundary_Type    4147 non-null   object  
 1   Boundary_Country_Code_2   4147 non-null   object  
 2   Boundary_HER_2            4147 non-null   object  
 3   Boundary_HER_PRN_2        4147 non-null   object  
 4   Boundary_Current_County_2 4147 non-null   object  
 5   Boundary_Historic_County_2 4147 non-null   object  
 6   Boundary_Current_Parish_2  4147 non-null   object  
dtypes: object(7)
memory usage: 226.9+ KB

```

Boundary Type Plotted

The majority of hillforts (3719/89.68%) have no Boundary Type information.

```
In [ ]: boundary_encodeable_data['Boundary_Boundary_Type'].value_counts()
```

```

Out[ ]: NA                3719
Parish/Townland      391
County              22
Other                 8
National              7
Name: Boundary_Boundary_Type, dtype: int64

```

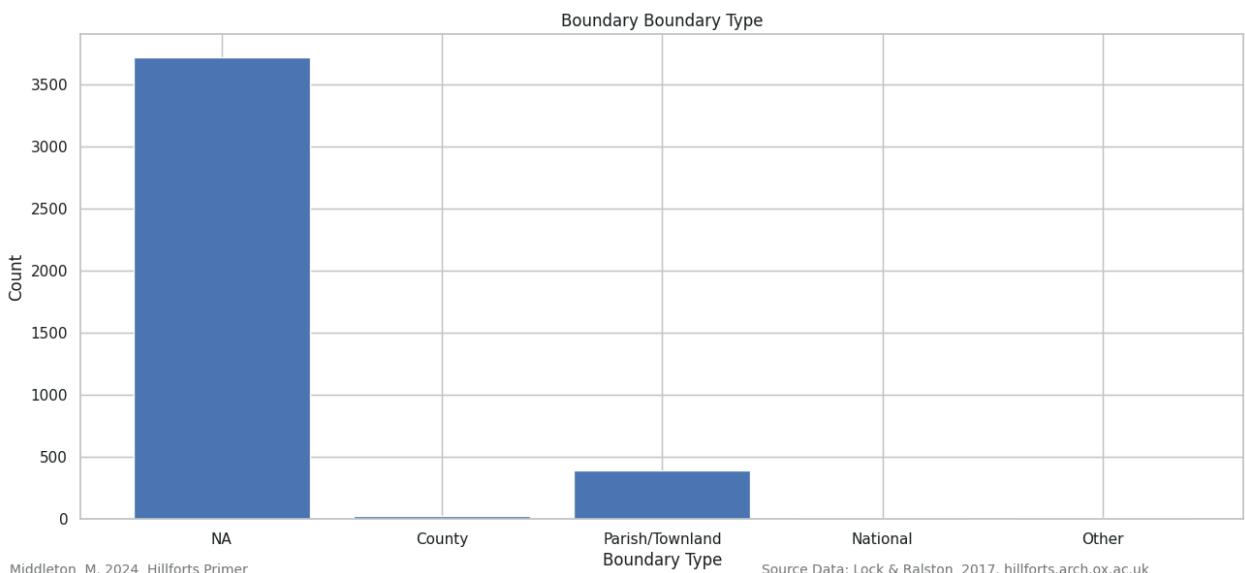
```

In [ ]: x_data = []
for bdry in list(pd.unique(boundary_encodeable_data['Boundary_Boundary_Type'])):
    x_data.append(bdry)

y_data = []
for entry in x_data:
    count = \
        len(boundary_encodeable_data[boundary_encodeable_data\
            ['Boundary_Boundary_Type'] == entry])
    y_data.append(count)

```

```
In [ ]: plot_bar_chart_using_two_tables(x_data, y_data, 'Boundary Type', 'Count', \
                                         'Boundary_Boundary_Type')
```

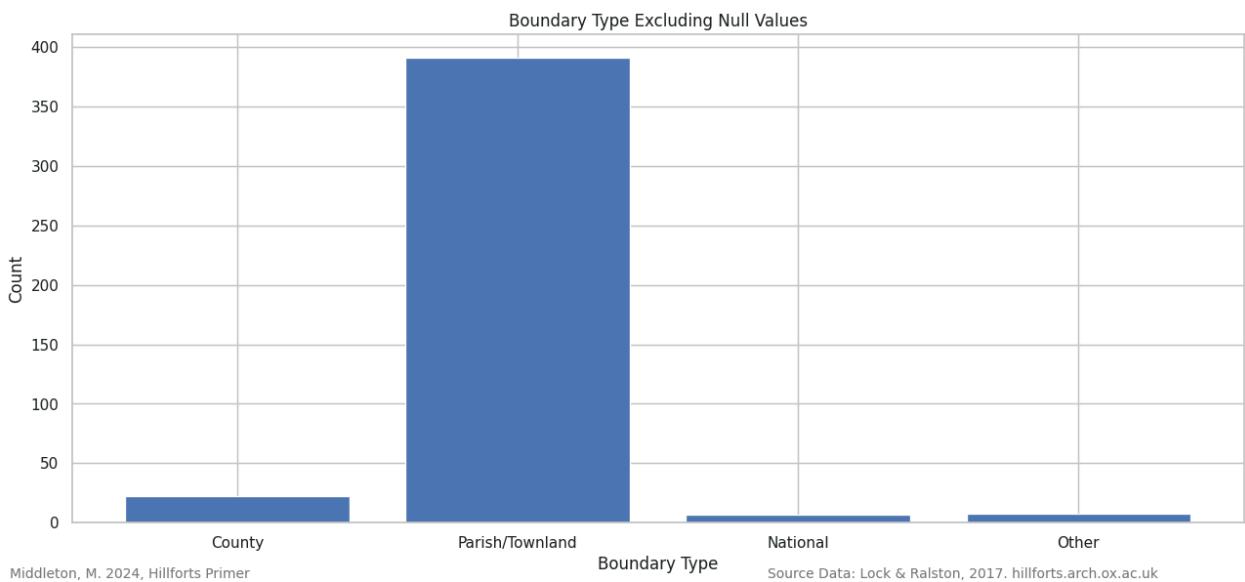


Boundary Data Plotted (Excluding No Boundary information)

Where a Boundary Type has been recorded, Parish/Townland is the most common (391). There are a small number of hillforts on a county boundary (22) and an even smaller number on a national border (7).

```
In [ ]: x_data_short = x_data[1:]
y_data_short = y_data[1:]
```

```
In [ ]: plot_bar_chart_using_two_tables(x_data_short, y_data_short, 'Boundary Type', \
                                      'Count', 'Boundary_Type_Excluding_Null_Values')
```



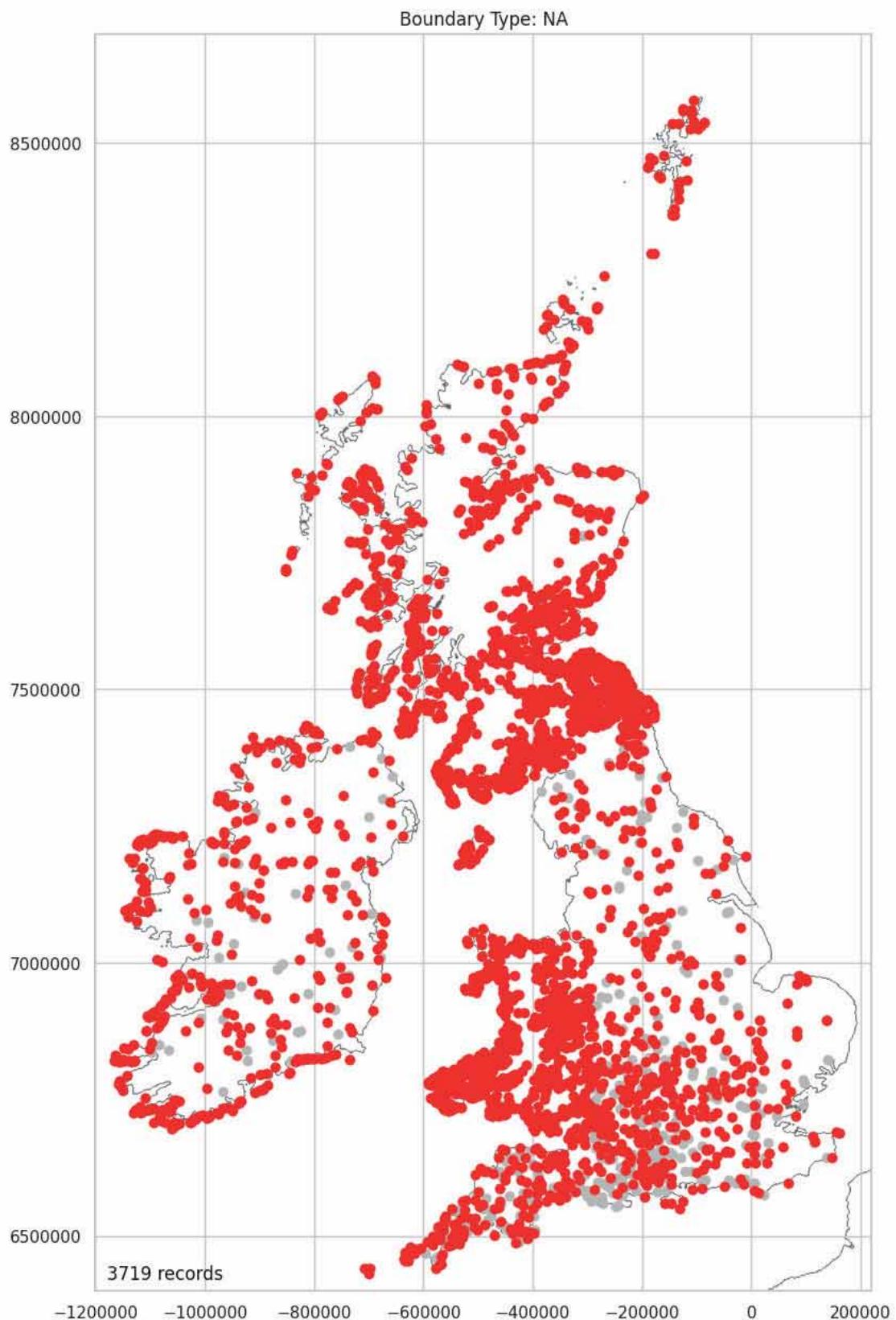
Boundary Data Mapped

```
In [ ]: location_boundary_data = \
pd.merge(location_numeric_data_short, boundary_encodeable_data, \
        left_index=True, right_index=True)
```

No Boundary Data Mapped

3719 hillforts (89.68%) have no associated boundary information.

```
In [ ]: plot_over_grey_boundary(location_boundary_data, 'Boundary_Boundary_Type', 'NA')
```



Middleton, M. 2024, Hillforts Primer

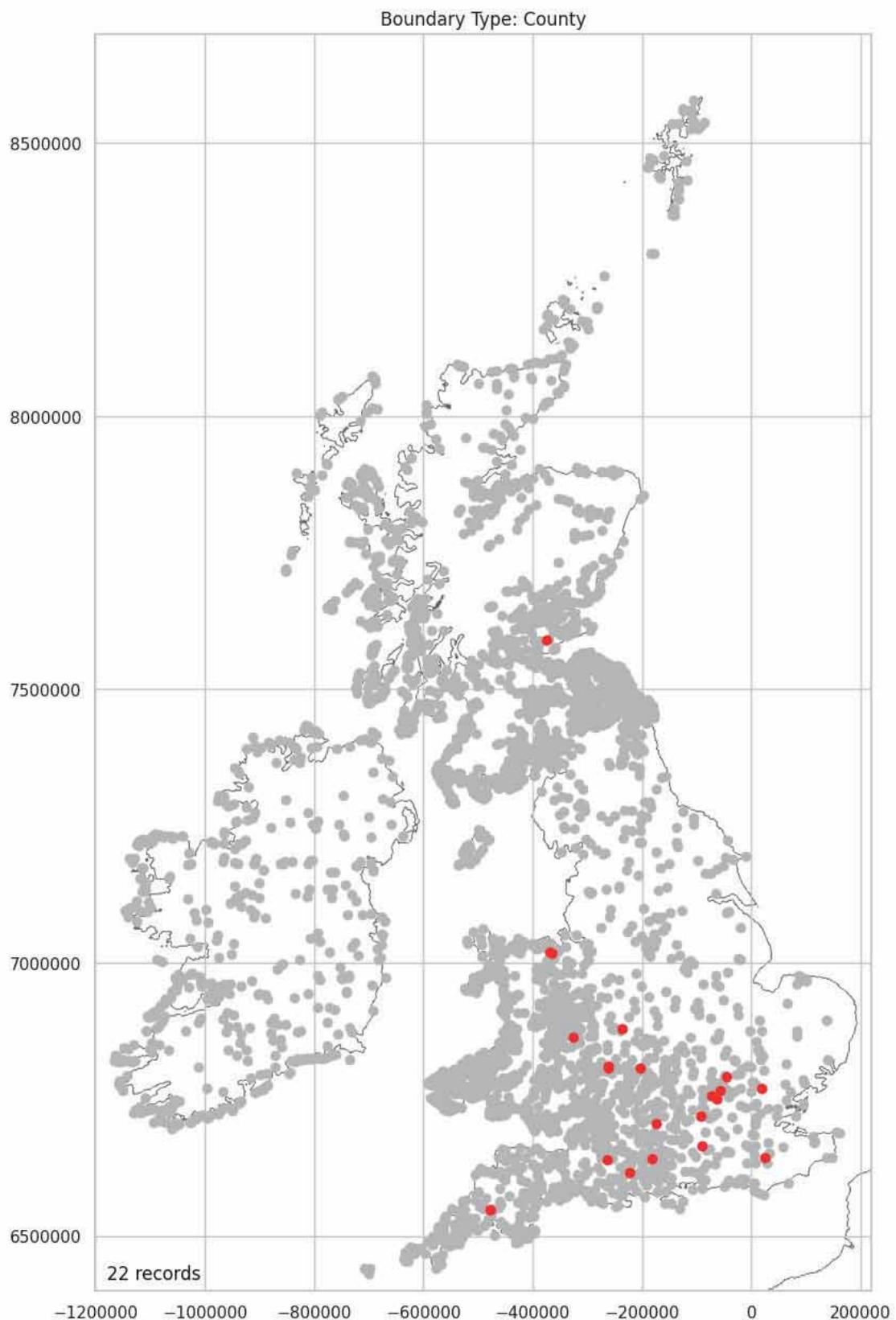
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

89.68%

County Boundary Mapped

Only 22 hillforts have a relationship to county boundaries.

```
In [ ]: plot_over_grey_boundary(location_boundary_data, 'Boundary_Boundary_Type', 'County')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

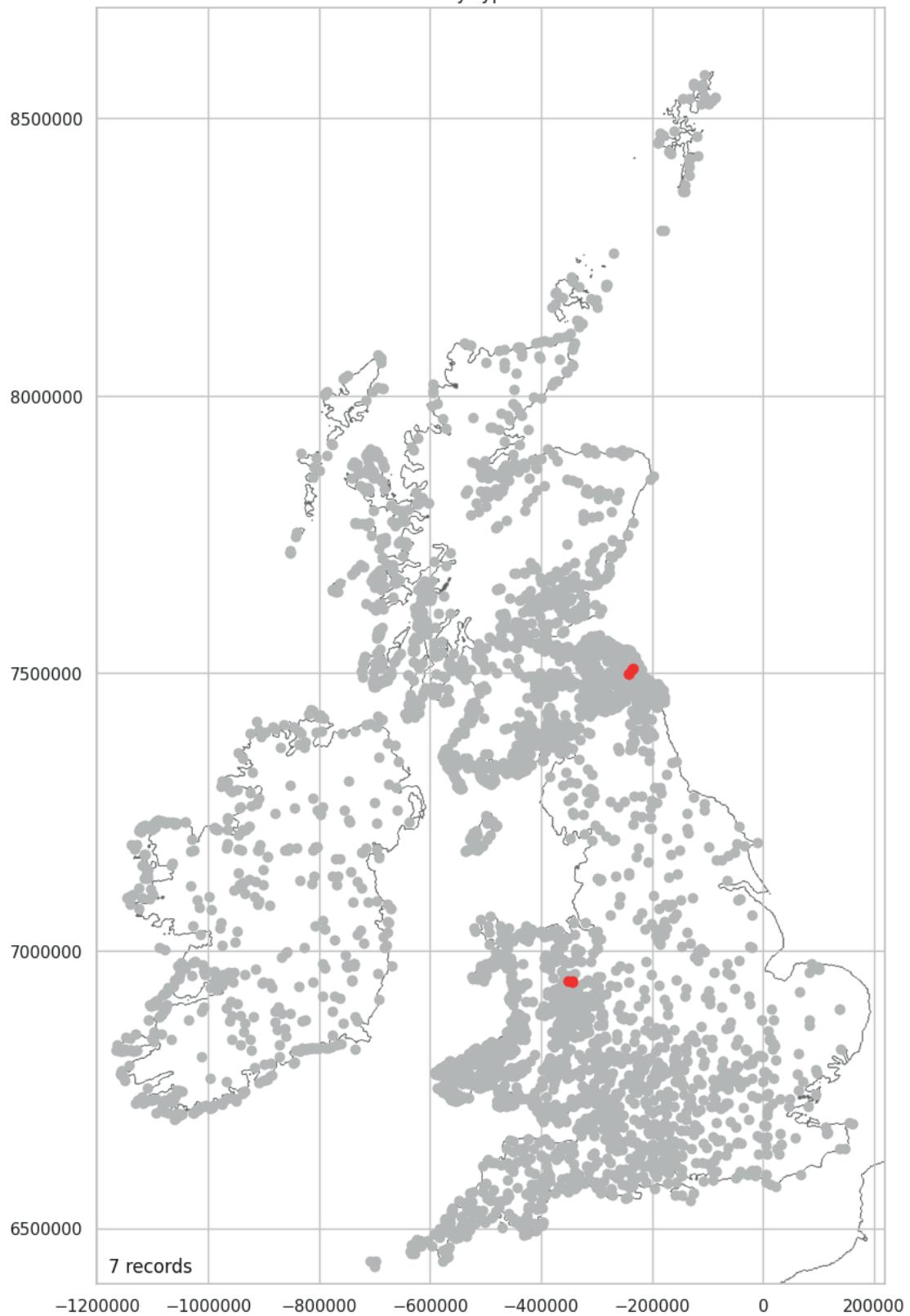
0.53%

National Boundary Mapped

Only seven hillforts have been recorded as built, on what is now, a national boundary.

```
In [ ]: plot_over_grey_boundary(location_boundary_data, \
                                'Boundary_Boundary_Type', 'National')
```

Boundary Type: National



Middleton, M. 2024, Hillforts Primer

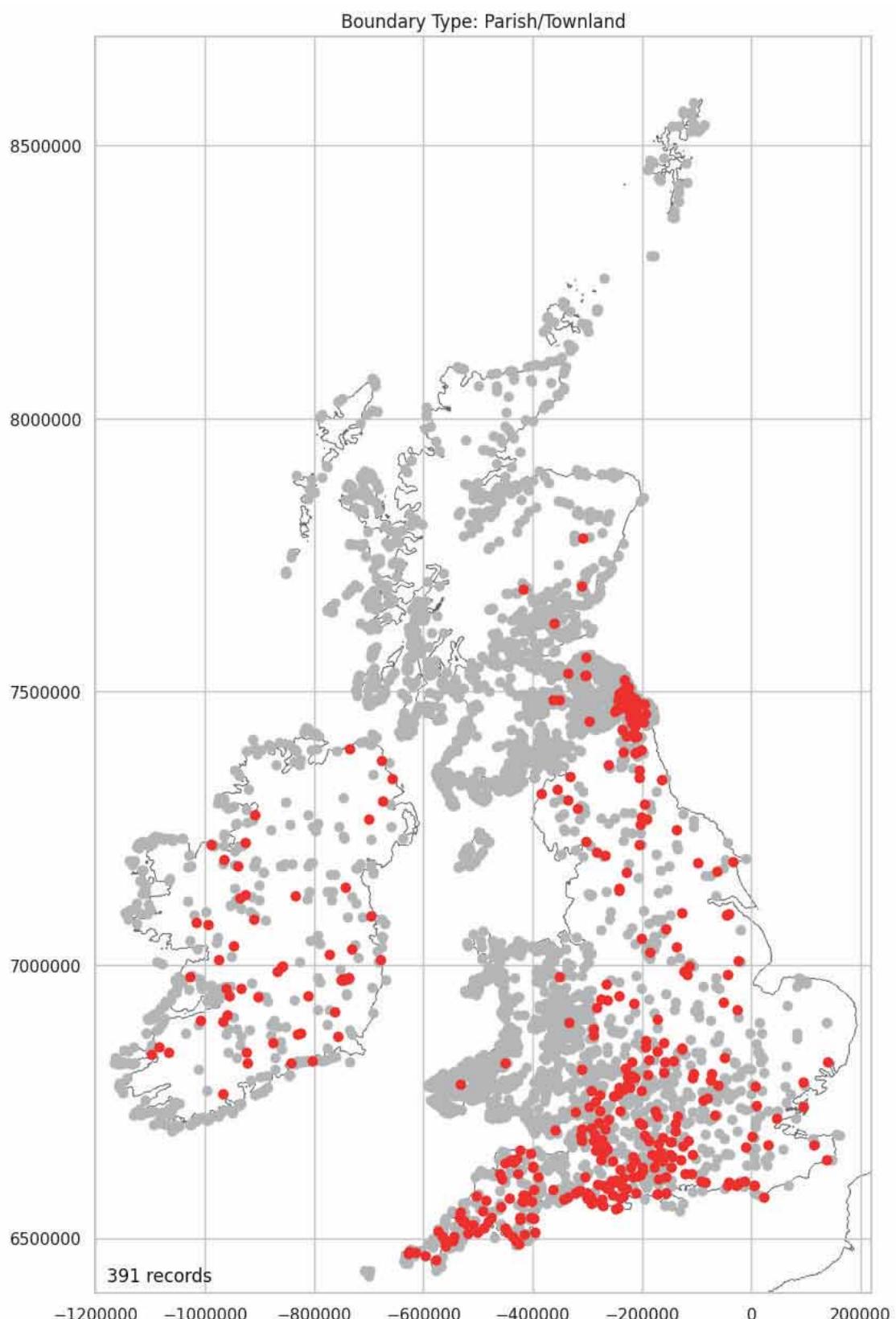
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.17%

Parish / Townland Boundary Mapped

This distribution contains a survey bias. The data for England and the Republic of Ireland looks to have some coherence but the data for Scotland, Wales and Northern Ireland is patchy. There also seems to be recording bias, caused by localised intense recording, around Berwick and various locations across the south of England. See: [Boundary Current Parish 2 Mapped](#).

```
In [ ]: plot_over_grey_boundary(location_boundary_data, 'Boundary_Boundary_Type', \
'Parish/Townland')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

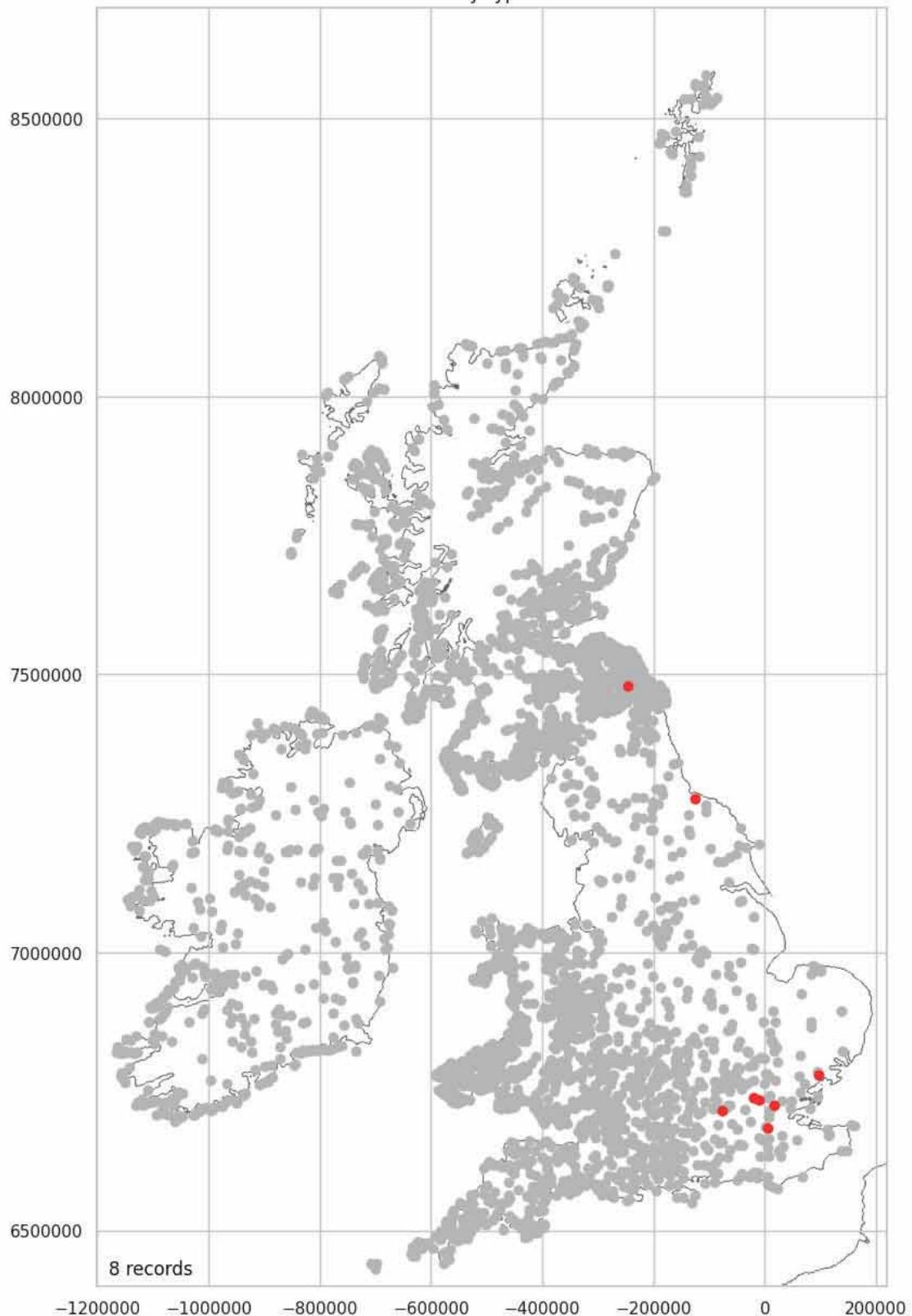
9.43%

Other Boundary Mapped

Eight hillforts coincide with boundaries classified as other.

```
In [ ]: plot_over_grey_boundary(location_boundary_data, 'Boundary_Boundary_Type', \
                                'Other')
```

Boundary Type: Other



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.19%

Boundary Country Code 2

Seven hillforts have a Boundary Code 2. There are three codes used which relate to England (EN), Scotland (SC) and Wales (WA).

```
In [ ]: boundary_encodeable_data['Boundary_Country_Code_2'].value_counts()
```

```
Out[ ]: NA      4140
         EN       4
         WA       2
         SC       1
Name: Boundary_Country_Code_2, dtype: int64
```

Boundary HER 2

There are 12 hillforts with a Boundary HER 2. There are nine values, of which most have a single entry. All are HER service names.

```
In [ ]: boundary_encodeable_data['Boundary_HER_2'].value_counts()
```

```
Out[ ]: NA          4135
         Worcestershire    2
         Shropshire        2
         Clwyd Powys       2
         West Berkshire     1
         Dudley            1
         Hampshire          1
         Fife Council       1
         East Sussex         1
         Wiltshire and Swindon 1
Name: Boundary_HER_2, dtype: int64
```

Boundary HER PRN 2

There are ten hillforts with a HER PRN 2 value. There are nine unique values, of which most have a single entry. All are HER PRN ID numbers.

```
In [ ]: boundary_encodeable_data['Boundary_HER_PRN_2'].value_counts()
```

```
Out[ ]: NA          4137
         WSM00932        2
         MSA868           1
         19259           1
         MSA828           1
         MWB3075          1
         7097             1
         3252             1
         28636            1
         MWI 17466         1
Name: Boundary_HER_PRN_2, dtype: int64
```

Boundary Current County 2

There are 20 hillforts with a Boundary Current County 2. There are 15 unique values, of which most have a single entry. All are county names.

```
In [ ]: boundary_encodeable_data['Boundary_Current_County_2'].value_counts()
```

```
Out[ ]: NA          4127
         Worcestershire    3
         Shropshire        2
         Powys            2
         Wiltshire          2
         West Berkshire     1
         West Midlands      1
         Somerset          1
         Flintshire        1
         Denbighshire       1
         Hampshire          1
         Fife              1
         Hertfordshire      1
         Windsor and Maidenhead 1
         Cornwall           1
         East Sussex         1
Name: Boundary_Current_County_2, dtype: int64
```

Boundary Historic County 2

There are 17 hillforts with a Boundary Historic County 2. There are 13 unique values, of which most have a single entry. All are historic county names.

```
In [ ]: boundary_encodeable_data['Boundary_Historic_County_2'].value_counts()
```

```

Out[ ]: NA          4130
         Worcestershire    3
         Shropshire        2
         Denbighshire       2
         Montgomeryshire   1
         Hampshire         1
         Berkshire          1
         Montgomeryshire   1
         Staffordshire     1
         Somerset          1
         Flintshire        1
         Selkirkshire       1
         Sussex             1
         Wiltshire          1
Name: Boundary_Historic_County_2, dtype: int64

```

Boundary Current Parish 2

There are 14 hillforts with a Boundary Current Parish 2. There are four unique values.

```

In [ ]: boundary_encodeable_data['Boundary_Current_Parish_2'].value_counts()[:5]

```

```

Out[ ]: NA      3787
         Branxton    4
         Chatton     4
         Doddington  3
         Lowick      3
Name: Boundary_Current_Parish_2, dtype: int64

```

Boundary Current Parish 2 Mapped

This feature is very similar to, and suffers the same survey bias as, that detailed in [Parish / Townland Boundary Mapped](#).

```

In [ ]: temp_boundary_data_plus = boundary_encodeable_data.copy()
temp_boundary_data_plus.where(temp_boundary_data_plus['Boundary_Current_Parish_2'] \
                           == 'NA', 'Yes', inplace=True)
temp_boundary_data_plus.head()

```

	Boundary_Boundary_Type	Boundary_Country_Code_2	Boundary_HER_2	Boundary_HER_PRN_2	Boundary_Current_County_2	Boundary_Hi
0	NA	NA	NA	NA	NA	NA
1	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA	NA
4	Yes	Yes	Yes	Yes	Yes	Yes

```

In [ ]: temp_location_boundary_data = pd.merge(location_numeric_data_short, temp_boundary_data_plus, left_index=True, right_

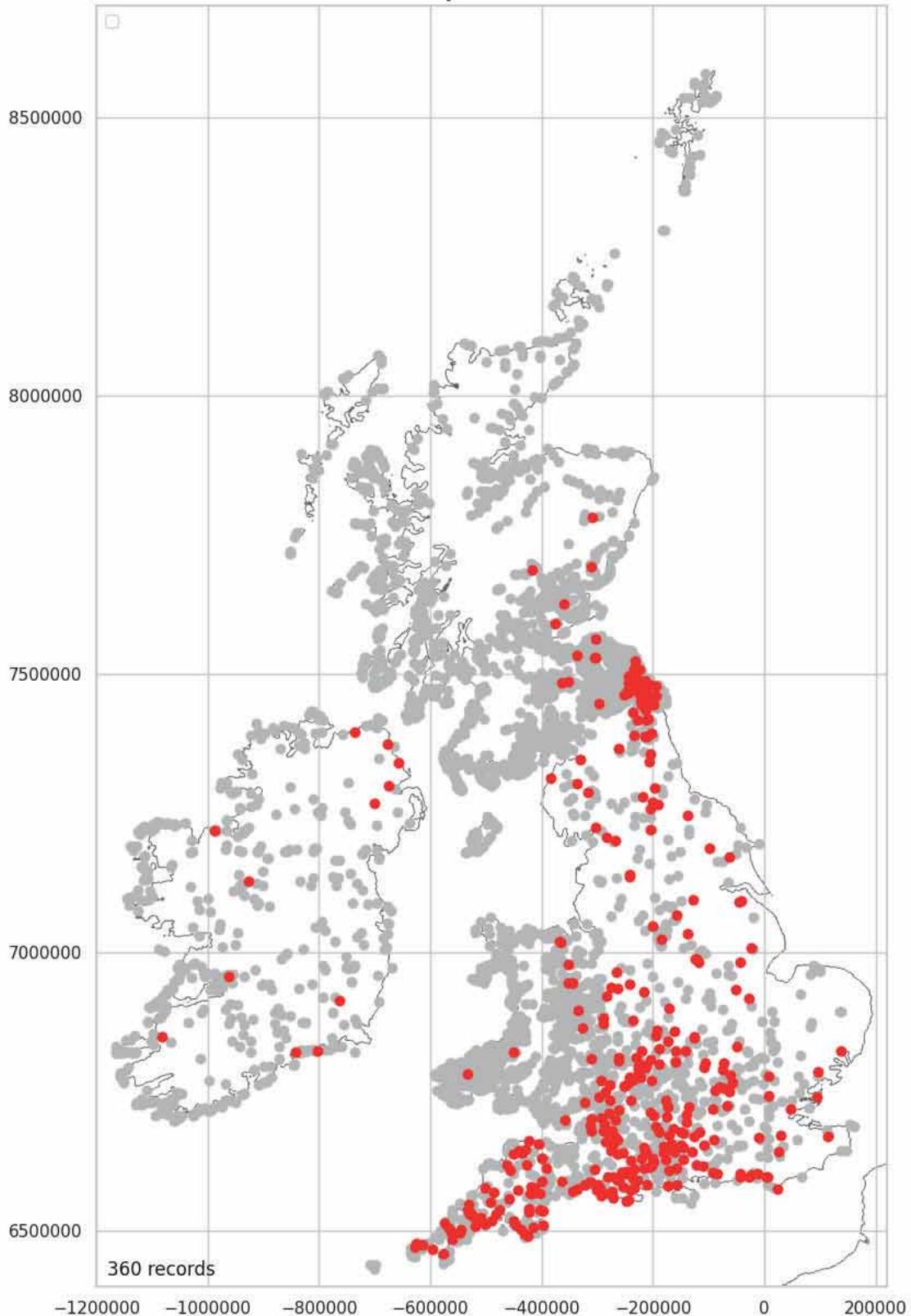
```

```

In [ ]: Boundary_Current_Parish_2_stats = \
plot_over_grey(temp_location_boundary_data, 'Boundary_Current_Parish_2', 'Yes')

```

Boundary Current Parish 2



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

8. 68%

Review Boundary Data Split

```
In [ ]: review_data_split(boundary_data, boundary_numeric_data, boundary_text_data, \
boundary_encodeable_data)
```

Data split good.

Drop Features From Boundary Encodeable Data

Only Boundary Type and Current Parish 2 will be retained in the download data package. The remaining boundary features do not contain sufficient data and would likely be misleading, if used in a machine learning model.

```
In [ ]: boundary_encodeable_features_short = [  
    'Boundary_Boundary_Type',  
    'Boundary_Current_Parish_2']
```

```
In [ ]: boundary_encodeable_short = \  
boundary_encodeable_data[boundary_encodeable_features_short].copy()  
boundary_encodeable_short.head()
```

```
Out[ ]:
```

	Boundary_Boundary_Type	Boundary_Current_Parish_2
0	NA	NA
1	NA	NA
2	NA	NA
3	NA	NA
4	County	Eastnor (Herefordshire); Little Malvern (Worce...

Boundary Data Package

```
In [ ]: boundary_data_list = [boundary_numeric_data, boundary_text_data, boundary_encodeable_short]
```

Boundary Data Download Package

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(boundary_data_list, 'Boundary_package')
```

Dating Data

Additional Dating information is held in a separate Dating Evidence Table. This can be downloaded from the Hillforts Atlas Rest Service API [here](#) or this project's data store [here](#). The Dating Evidence Table has not been analysed as part of the Hillforts Primer at this time.

There are 15 Dating features. The first eight record period date ranges. There are two features recording dating evidence of activity prior to construction and two features recording activity post abandon. The remaining three features record dating reliability, related dating evidence and general dating comments.

```
In [ ]: dating_features = [  
    'Dating_Date_Pre_1200BC',  
    'Dating_Date_1200BC_800BC',  
    'Dating_Date_800BC_400BC',  
    'Dating_Date_400BC_AD50',  
    'Dating_Date_AD50_AD400',  
    'Dating_Date_AD400_AD800',  
    'Dating_Date_Post_AD800',  
    'Dating_Date_Unknown',  
    'Dating_Date_Reliability',  
    'Dating_Date_Comments',  
    'Dating_Pre',  
    'Dating_Pre_Comments',  
    'Dating_Post',  
    'Dating_Post_Comments',  
    'Related_Dating_Evidence']  
  
dating_data = hillforts_data[dating_features].copy()  
dating_data.head()
```

```
Out[ ]:   Dating_Date_Pre_1200BC  Dating_Date_1200BC_800BC  Dating_Date_800BC_400BC  Dating_Date_400BC_AD50  Dating_Date_AD50_AD400
```

0	No	No	Yes	Yes	Yes
1	No	No	No	No	No
2	No	No	No	No	No
3	No	No	No	No	No
4	No	No	Yes	Yes	Yes

All the period features and the 'pre' and 'post' features contain yes/no responses. These contain no null entries. All the remaining features contain empty records. Reliability and Related dating evidence contain controlled vocabularies (data derived from a pick list).

```
In [ ]: dating_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
 ---  --  
 0   Dating_Date_Pre_1200BC    4147 non-null   object  
 1   Dating_Date_1200BC_800BC  4147 non-null   object  
 2   Dating_Date_800BC_400BC  4147 non-null   object  
 3   Dating_Date_400BC_AD50  4147 non-null   object  
 4   Dating_Date_AD50_AD400  4147 non-null   object  
 5   Dating_Date_AD400_AD800  4147 non-null   object  
 6   Dating_Date_Post_AD800  4147 non-null   object  
 7   Dating_Date_Unknown     4147 non-null   object  
 8   Dating_Date_Reliability 4134 non-null   object  
 9   Dating_Date_Comments   4116 non-null   object  
 10  Dating_Pre             4147 non-null   object  
 11  Dating_Pre_Comments   448 non-null   object  
 12  Dating_Post            4147 non-null   object  
 13  Dating_Post_Comments  1961 non-null   object  
 14  Related_Dating_Evidence 805 non-null   object  
dtypes: object(15)
memory usage: 486.1+ KB
```

Dating Numeric Data

There is no numeric Dating data.

```
In [ ]: dating_numeric_data = pd.DataFrame()
```

Dating Text Data

There are three Dating text features.

```
In [ ]: dating_text_features = [
'Dating_Date_Comments',
'Dating_Pre_Comments',
'Dating_Post_Comments']

dating_text_data = dating_data[dating_text_features].copy()
dating_text_data.head()
```

	Dating_Date_Comments	Dating_Pre_Comments	Dating_Post_Comments
0	The finding of Iron Age and Roman pottery sugg...	NaN	Evidence of Civil War occupation and possible ...
1	None	NaN	NaN
2	The chance finding of a number of cloudy blue ...	None	None
3	Iron Age to Roman and possible later enclosure...	Possible Bronze Age ring ditch could indicate ...	Possible later Roman or post-Roman enclosure.
4	The earlier enclosure of Phase I could be late...	There is no evidence of pre-hillfort activity,...	The ringwork is thought to be of medieval date.

Dating Text Data - Resolve Null Values

Test for 'NA'.

```
In [ ]: test_cat_list_for_NA(dating_text_data, dating_text_features)
```

```
Dating_Date_Comments 0  
Dating_Pre_Comments 0  
Dating_Post_Comments 0
```

Fill null values with 'NA'.

```
In [ ]: dating_text_data = update_cat_list_for_NA(dating_text_data, dating_text_features)  
dating_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 4147 entries, 0 to 4146  
Data columns (total 3 columns):  
 #   Column           Non-Null Count  Dtype     
 ---  --  
 0   Dating_Date_Comments  4147 non-null  object    
 1   Dating_Pre_Comments  4147 non-null  object    
 2   Dating_Post_Comments 4147 non-null  object    
 dtypes: object(3)  
memory usage: 97.3+ KB
```

Dating Encodable Data

```
In [ ]: dating_encodeable_features = [  
    'Dating_Date_Pre_1200BC',  
    'Dating_Date_1200BC_800BC',  
    'Dating_Date_800BC_400BC',  
    'Dating_Date_400BC_AD50',  
    'Dating_Date_AD50_AD400',  
    'Dating_Date_AD400_AD800',  
    'Dating_Date_Post_AD800',  
    'Dating_Date_Unknown',  
    'Dating_Date_Reliability',  
    'Dating_Pre',  
    'Dating_Post',  
    'Related_Dating_Evidence']
```

```
dating_encodeable_data = dating_data[dating_encodeable_features].copy()  
dating_encodeable_data.head()
```

```
Out[ ]:   Dating_Date_Pre_1200BC  Dating_Date_1200BC_800BC  Dating_Date_800BC_400BC  Dating_Date_400BC_AD50  Dating_Date_AD50_AD400  ...  
 0          No                No                Yes                Yes                Yes  
 1          No                No                No                 No                No  
 2          No                No                No                 No                No  
 3          No                No                No                 No                No  
 4          No                No                No                 Yes                Yes
```

Review Dating Data Split

```
In [ ]: review_data_split(dating_data, dating_numeric_data, \  
                           dating_text_data, dating_encodeable_data)
```

Data split good.

Period Data

The majority of hillforts have no date information. The maps below show the data contains a recording bias in that the majority of dating information comes from southern England. There is very little dating information outside this area.

```
In [ ]: date_features = [  
    'Dating_Date_Pre_1200BC',  
    'Dating_Date_1200BC_800BC',  
    'Dating_Date_800BC_400BC',  
    'Dating_Date_400BC_AD50',  
    'Dating_Date_AD50_AD400',  
    'Dating_Date_AD400_AD800',  
    'Dating_Date_Post_AD800',  
    'Dating_Date_Unknown']
```

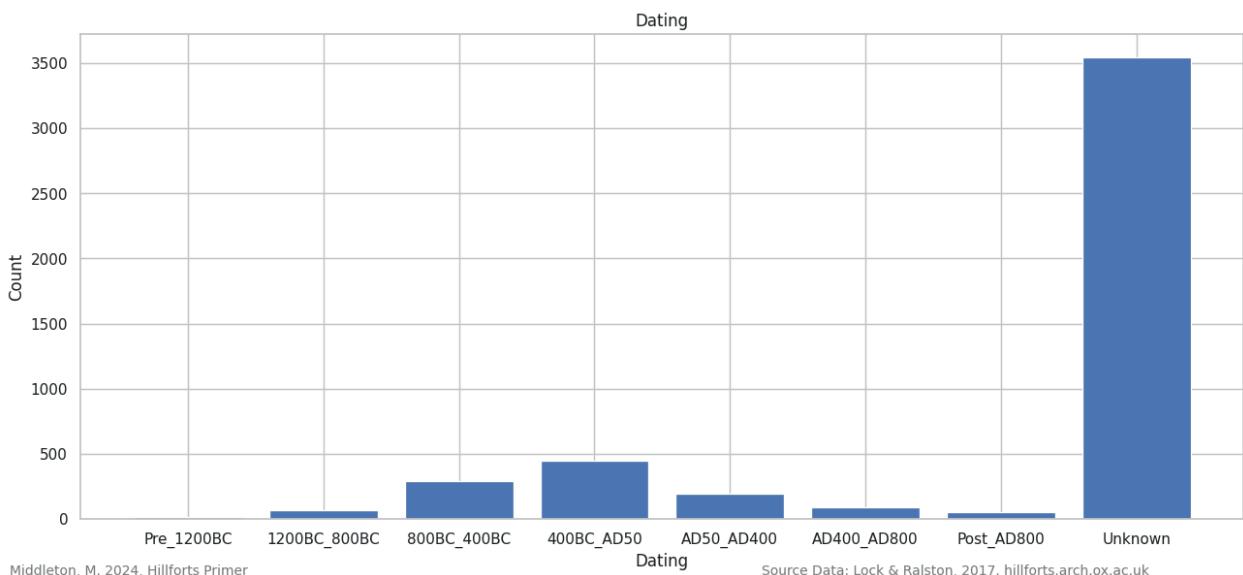
```
date_data = dating_encodeable_data[date_features]
date_data.head()
```

	Dating_Date_Pre_1200BC	Dating_Date_1200BC_800BC	Dating_Date_800BC_400BC	Dating_Date_400BC_AD50	Dating_Date_AD50_AD400	Dating_Date_Post_AD800
0	No	No	Yes	Yes	Yes	Yes
1	No	No	No	No	No	No
2	No	No	No	No	No	No
3	No	No	No	No	No	Yes
4	No	No	Yes	Yes	Yes	Yes

Period Data Plotted

The majority of hillforts are undated.

```
In [ ]: plot_bar_chart(date_data, 2, 'Dating', 'Count', 'Dating')
```



Date Data (Excluding No Dates) Plotted

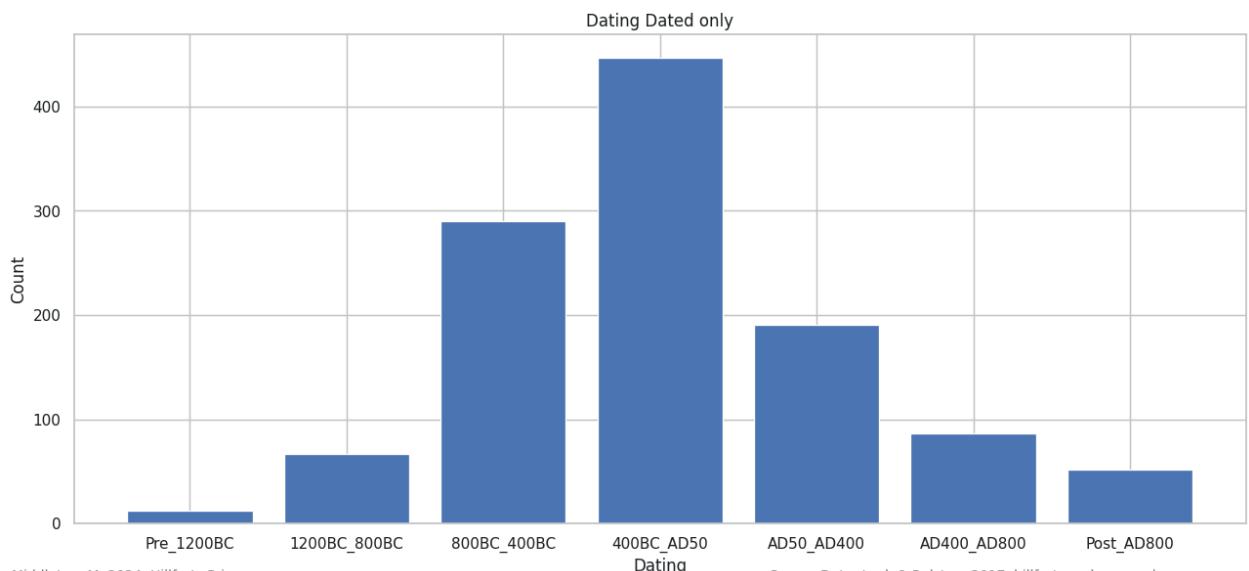
For the relatively few hillforts that have dating evidence, the majority are dated to the Iron Age (800BC to AD50) with the largest cluster being the late Iron Age (400BC to AD50). There are relatively few pre Iron Age dates. In contrast, there are a number early medieval dates (AD50 to AD800) with most falling at the lower end between AD50 and AD400.

```
In [ ]: date_features_mius = [
'Dating_Date_Pre_1200BC',
'Dating_Date_1200BC_800BC',
'Dating_Date_800BC_400BC',
'Dating_Date_400BC_AD50',
'Dating_Date_AD50_AD400',
'Dating_Date_AD400_AD800',
'Dating_Date_Post_AD800']
```

```
date_data_mius = hillforts_data[date_features_mius]
date_data_mius.head()
```

	Dating_Date_Pre_1200BC	Dating_Date_1200BC_800BC	Dating_Date_800BC_400BC	Dating_Date_400BC_AD50	Dating_Date_AD50_AD400	Dating_Date_Post_AD800
0	No	No	Yes	Yes	Yes	Yes
1	No	No	No	No	No	No
2	No	No	No	No	No	No
3	No	No	No	No	No	Yes
4	No	No	Yes	Yes	Yes	Yes

```
In [ ]: plot_bar_chart(date_data_mius, 2, 'Dating', 'Count', 'Dating_Dated_only')
```



Period Data Mapped

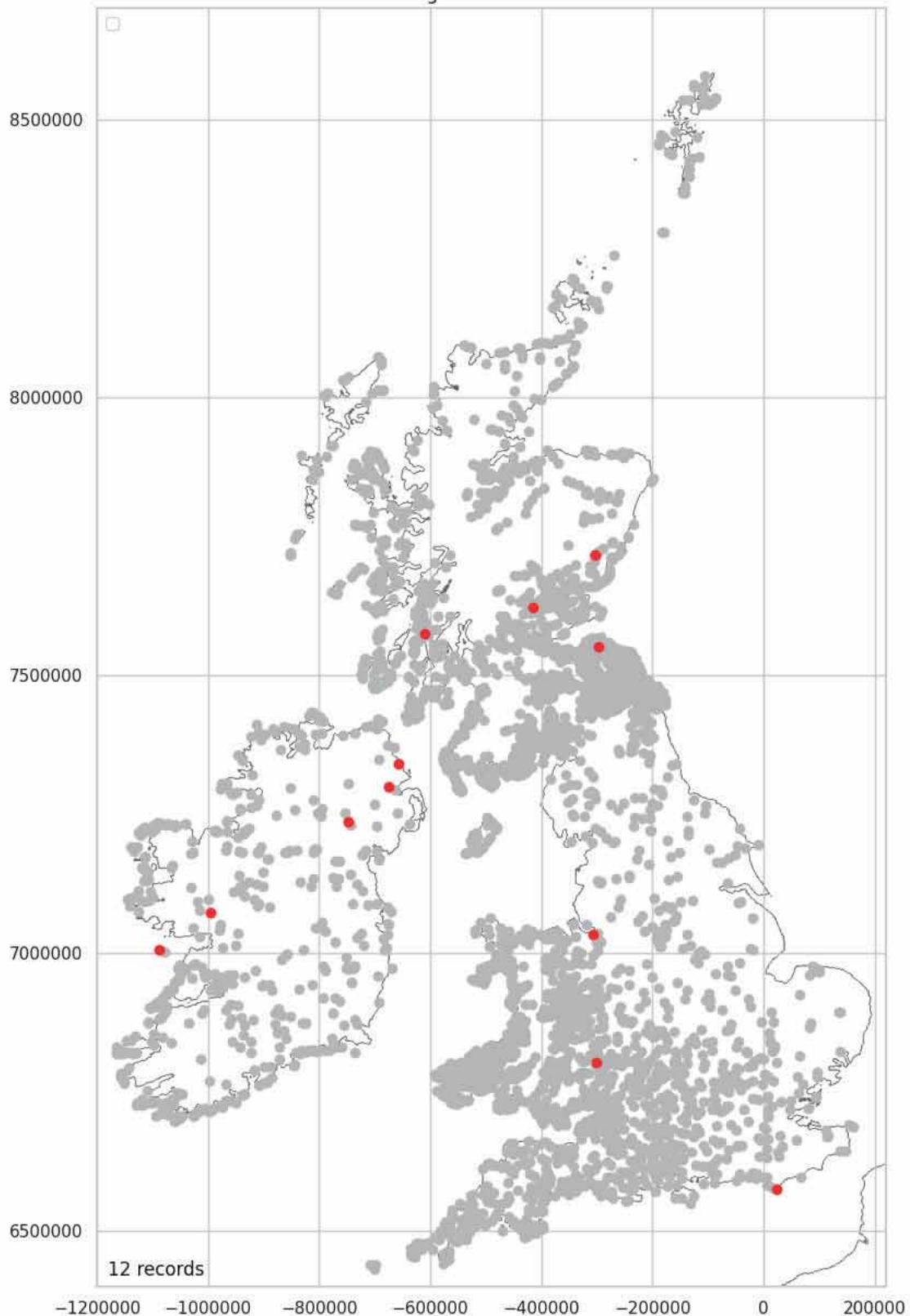
```
In [ ]: location_date_data = \
pd.merge(location_numeric_data_short, date_data, left_index=True, \
right_index=True)
```

Pre 1200 BC Mapped

There are only 12 hillforts which have produced dates pre 1200BC. A density plot has not been produced as there is insufficient data.

```
In [ ]: dt_1200 = plot_over_grey(location_date_data, 'Dating_Date_Pre_1200BC', 'Yes')
```

Dating Date Pre 1200BC



Middleton, M. 2024, Hillforts Primer

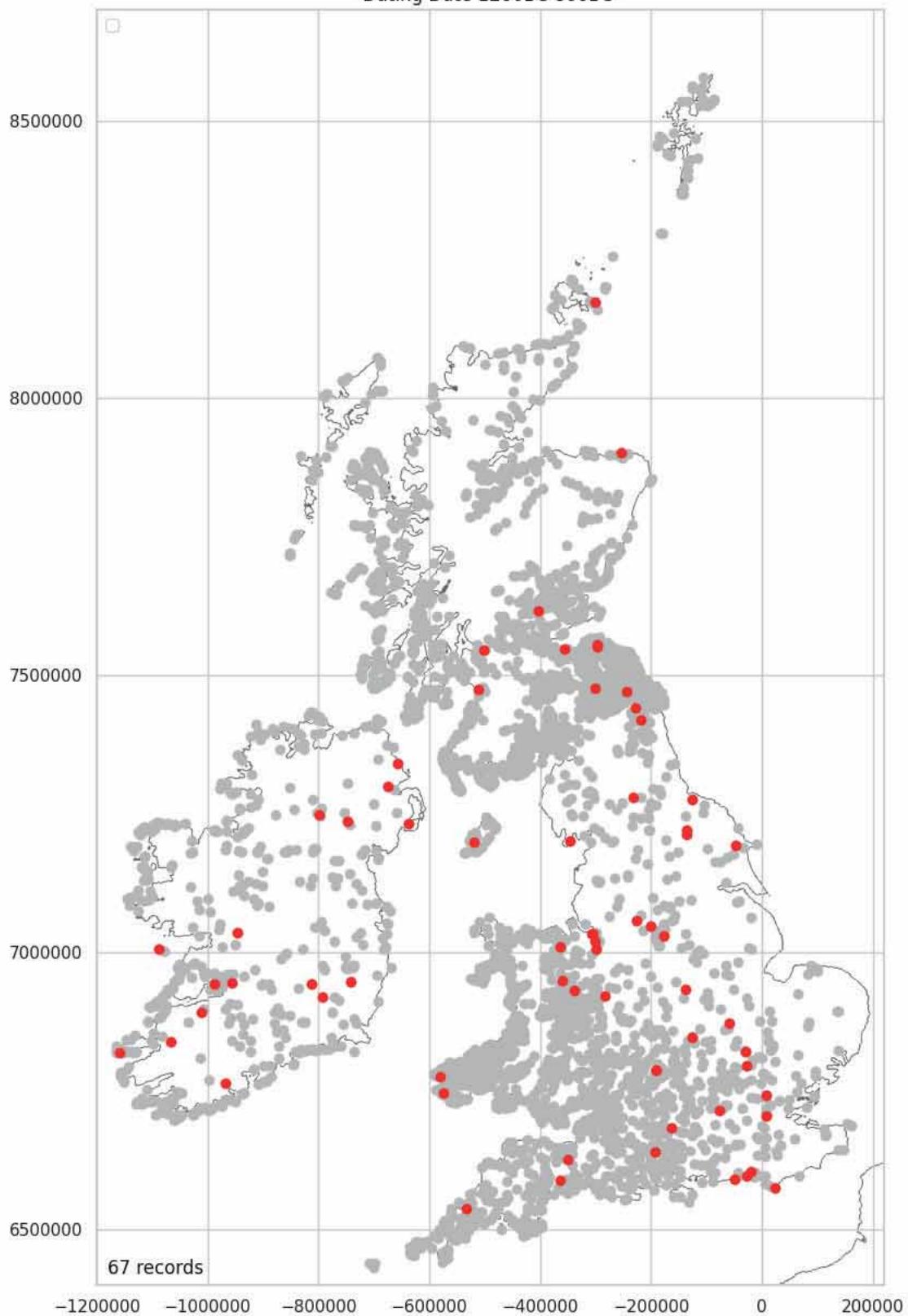
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.29%

1200BC - 800BC Mapped

Only 67 hillforts have a date between 1200 BC and 800 BC. A density plot has not been produced as there is insufficient data.

```
In [ ]: dt_1200_800 = \
plot_over_grey(location_date_data, 'Dating_Date_1200BC_800BC', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

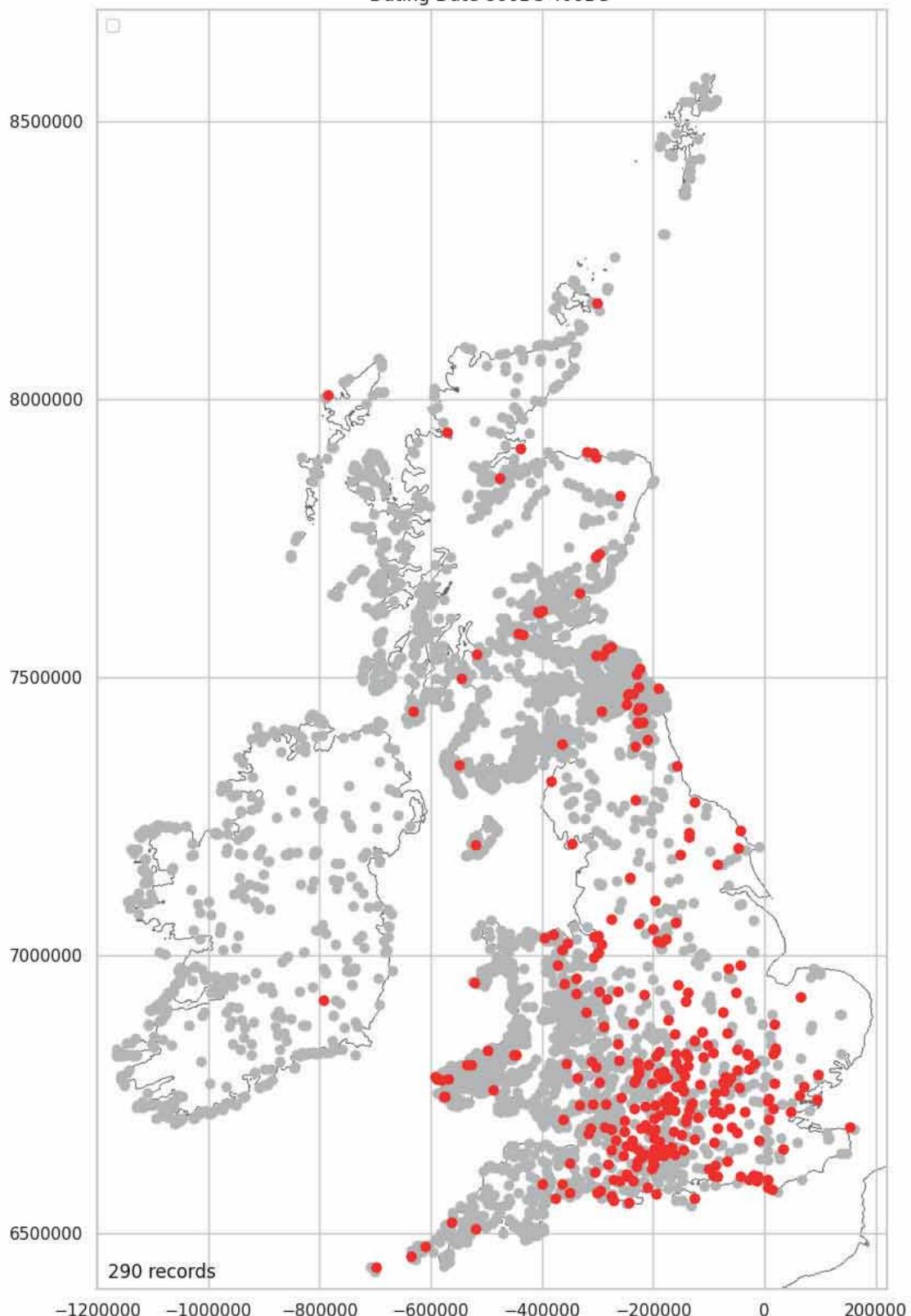
1. 62%

800BC - 400BC Mapped

Hillforts dated 800 BC to 400 BC are located predominantly in south central England. There is a clear bias in this distribution, resulting from a focus on excavation in the south, with there being only a single fort in Ireland and very few in Scotland, Wales and south-west England.

```
In [ ]: dt_800_400 = \
plot_over_grey(location_date_data, 'Dating_Date_800BC_400BC', 'Yes')
```

Dating Date 800BC 400BC



Middleton, M. 2024, Hillforts Primer

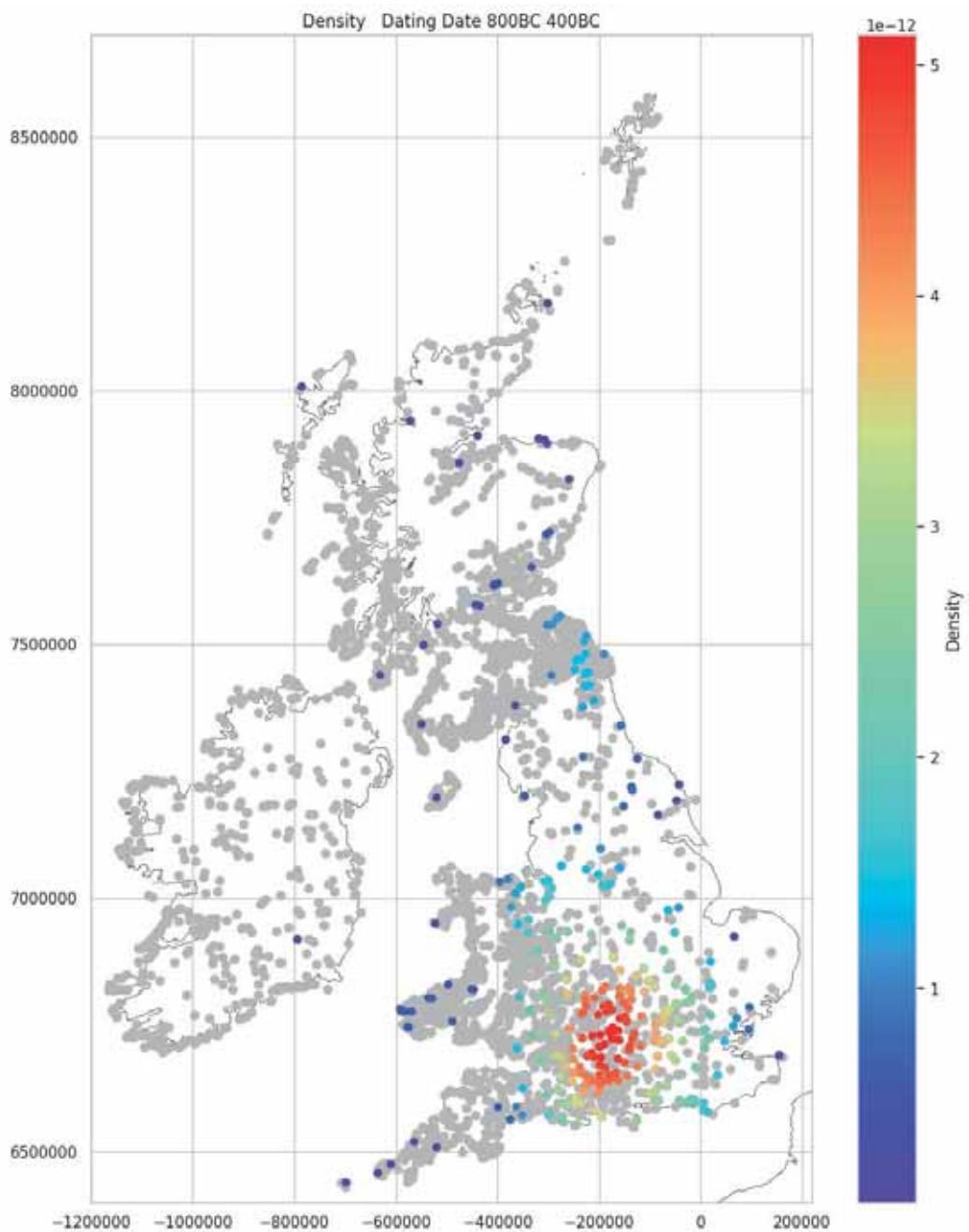
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 99%

800BC - 400BC Density Mapped

Although the density for this distribution has been produced, it looks to be a highly misleading cluster due to the excavation bias.

```
In [ ]: plot_density_over_grey(dt_800_400, 'Dating_Date_800BC_400BC')
```



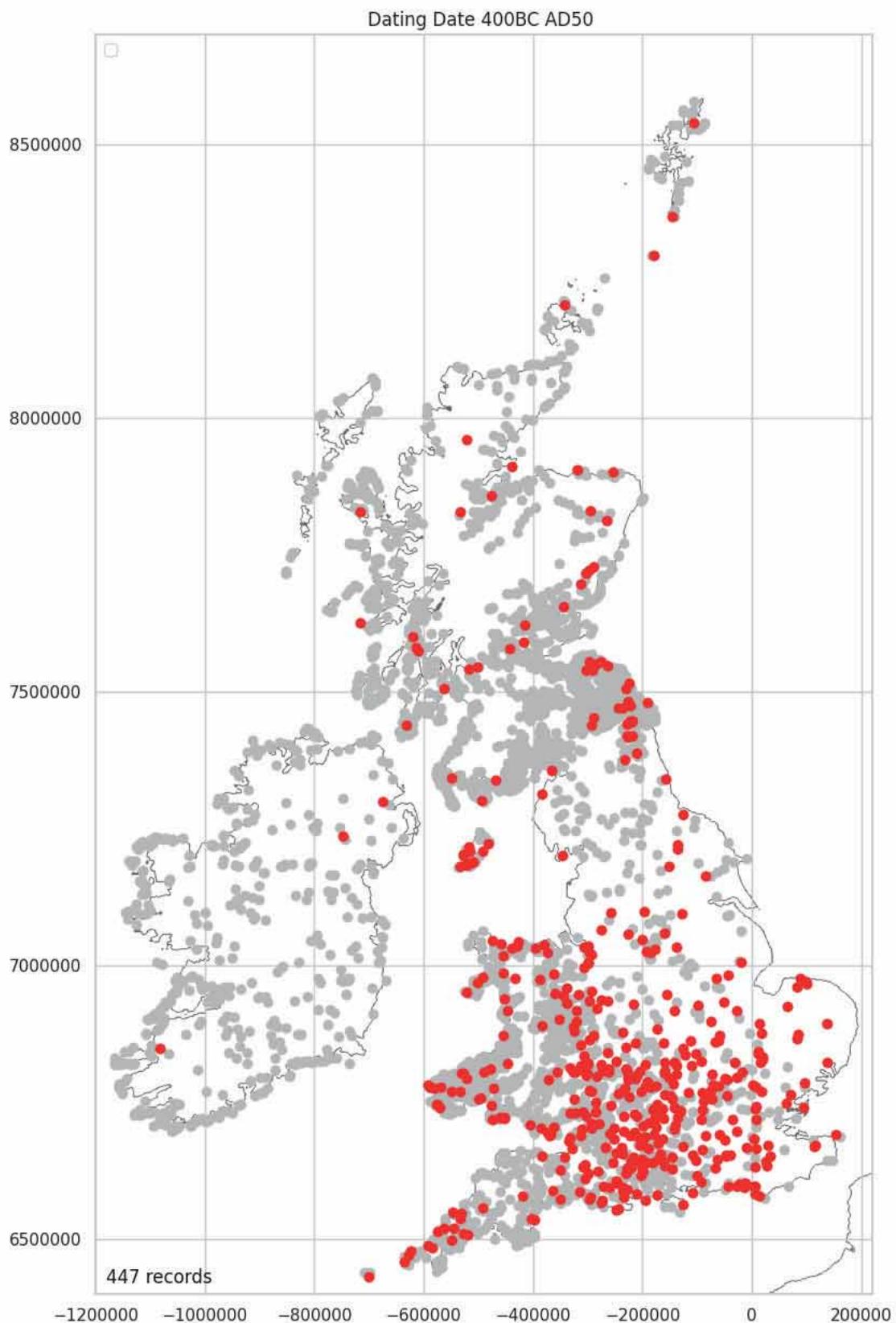
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

400BC - AD50 Mapped

The excavation bias seen in 800 BC to 400 BC is again visible in this cluster and similar voids in the record can be seen across the rest of the atlas.

```
In [ ]: dt_400_50 = plot_over_grey(location_date_data, 'Dating_Date_400BC_AD50', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

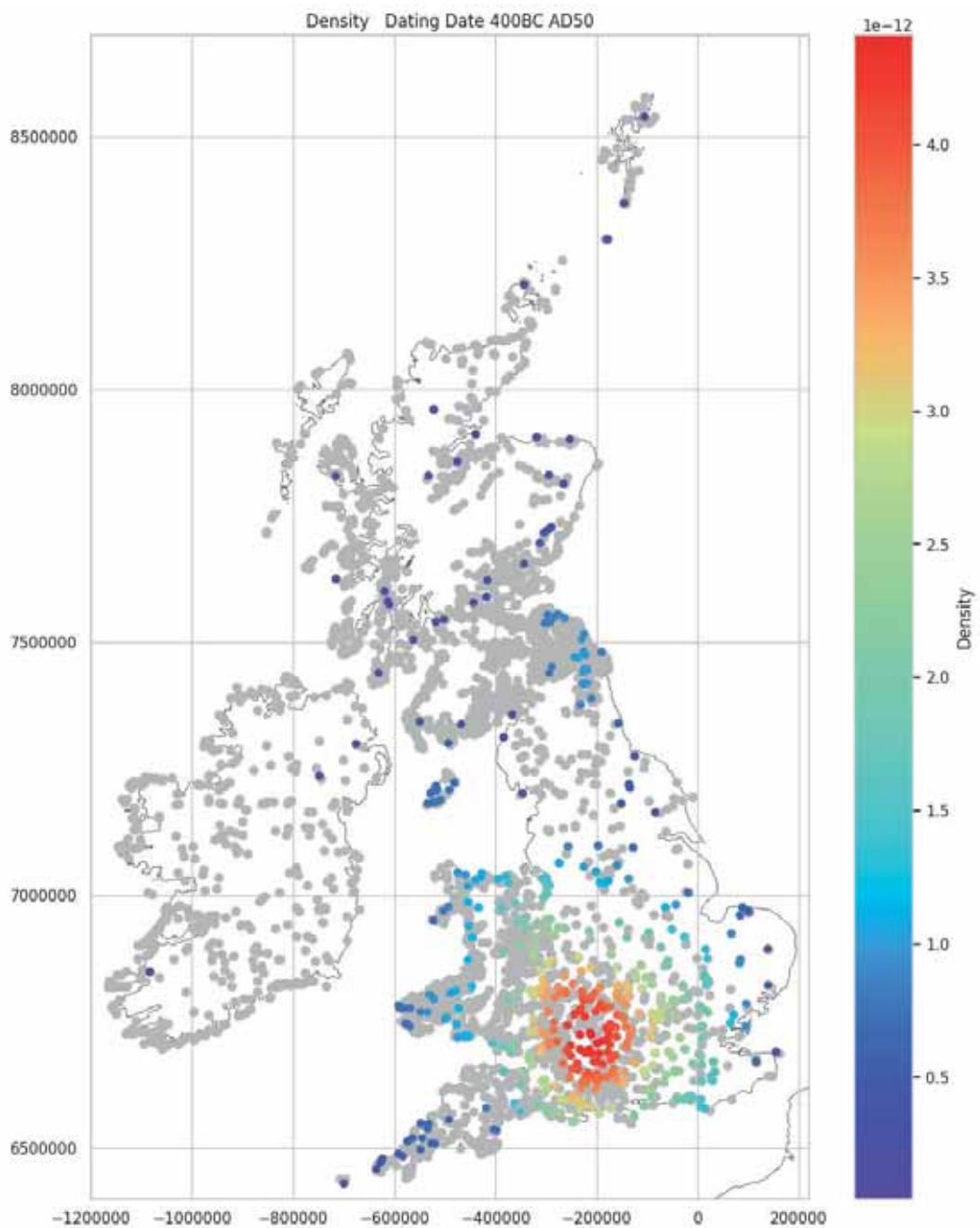
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

10. 78%

400BC - AD50 Density Mapped

The cluster focus is identical to that seen in 800 BC to 400 BC. Note the excavation bias mentioned above.

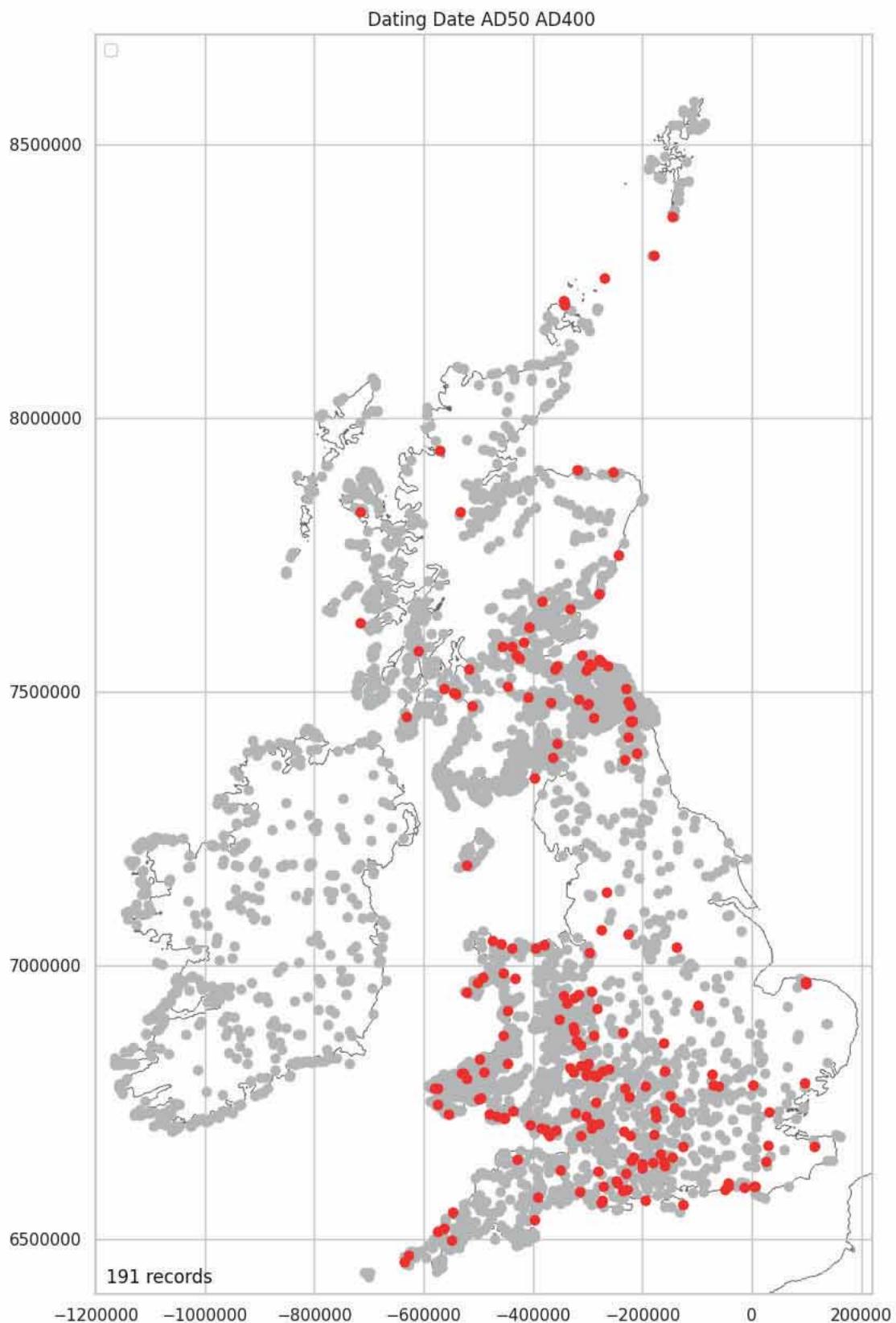
```
In [ ]: plot_density_over_grey(dt_400_50, 'Dating_Date_400BC_AD50')
```



AD50 - AD400 Mapped

The distribution of forts in the AD 50 to AD 400 range will also be biased because of the focus of excavation mentioned above. It is notable that there are no records recorded for this period in Ireland. Because of the excavation bias in the dating records, a distribution plot for this period has not been produced.

```
In [ ]: dt_50_400 = plot_over_grey(location_date_data, 'Dating_Date_AD50_AD400', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

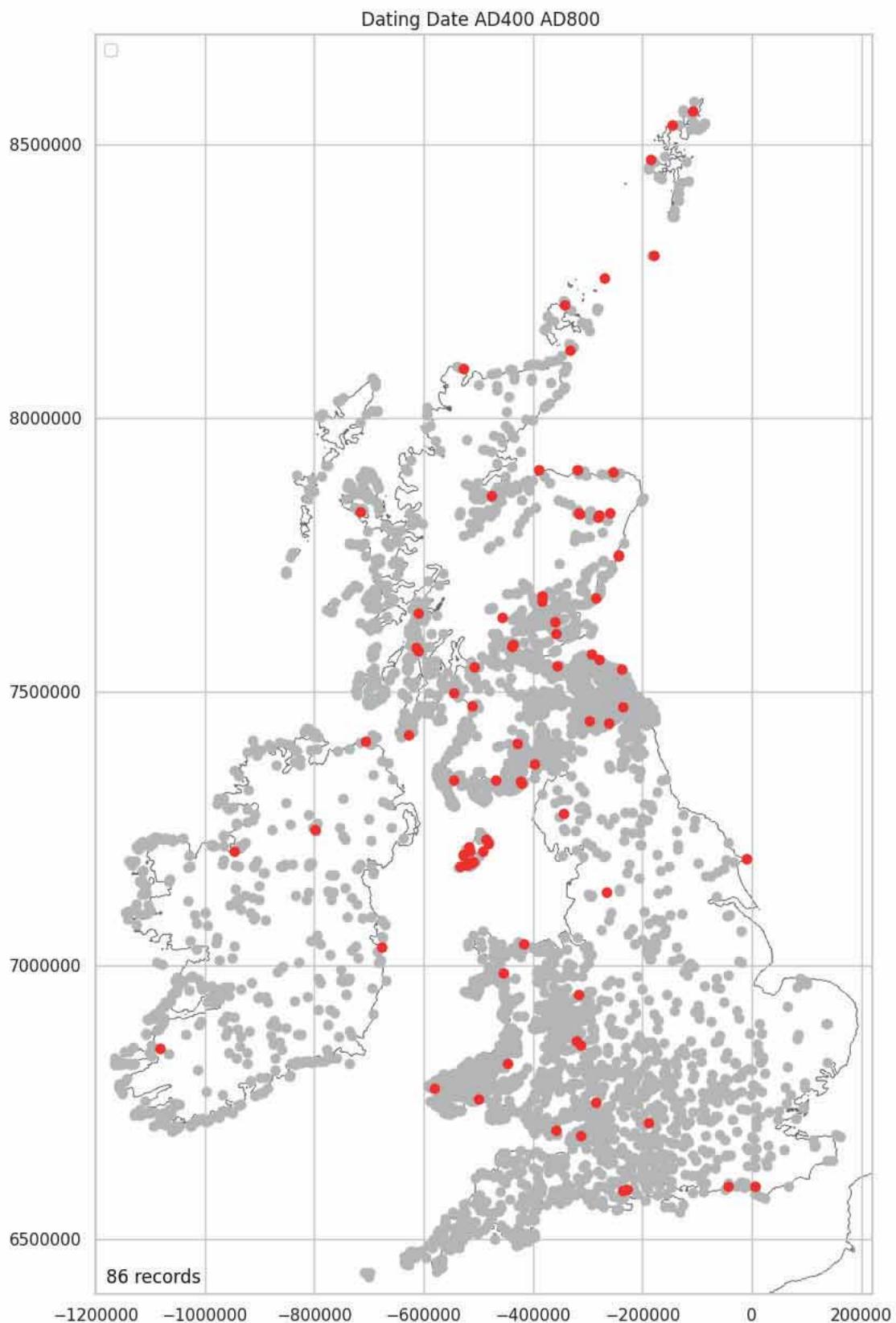
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4.61%

AD400 - AD800 Mapped

A small number of hillforts have dates in the AD 400 to AD 800 range. Interestingly most of these are outside the excavation bias seen in the figures above. This may be an observation that is meaningful for the south of England.

```
In [ ]: dt_400_800 = plot_over_grey(location_date_data, 'Dating_Date_AD400_AD800', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

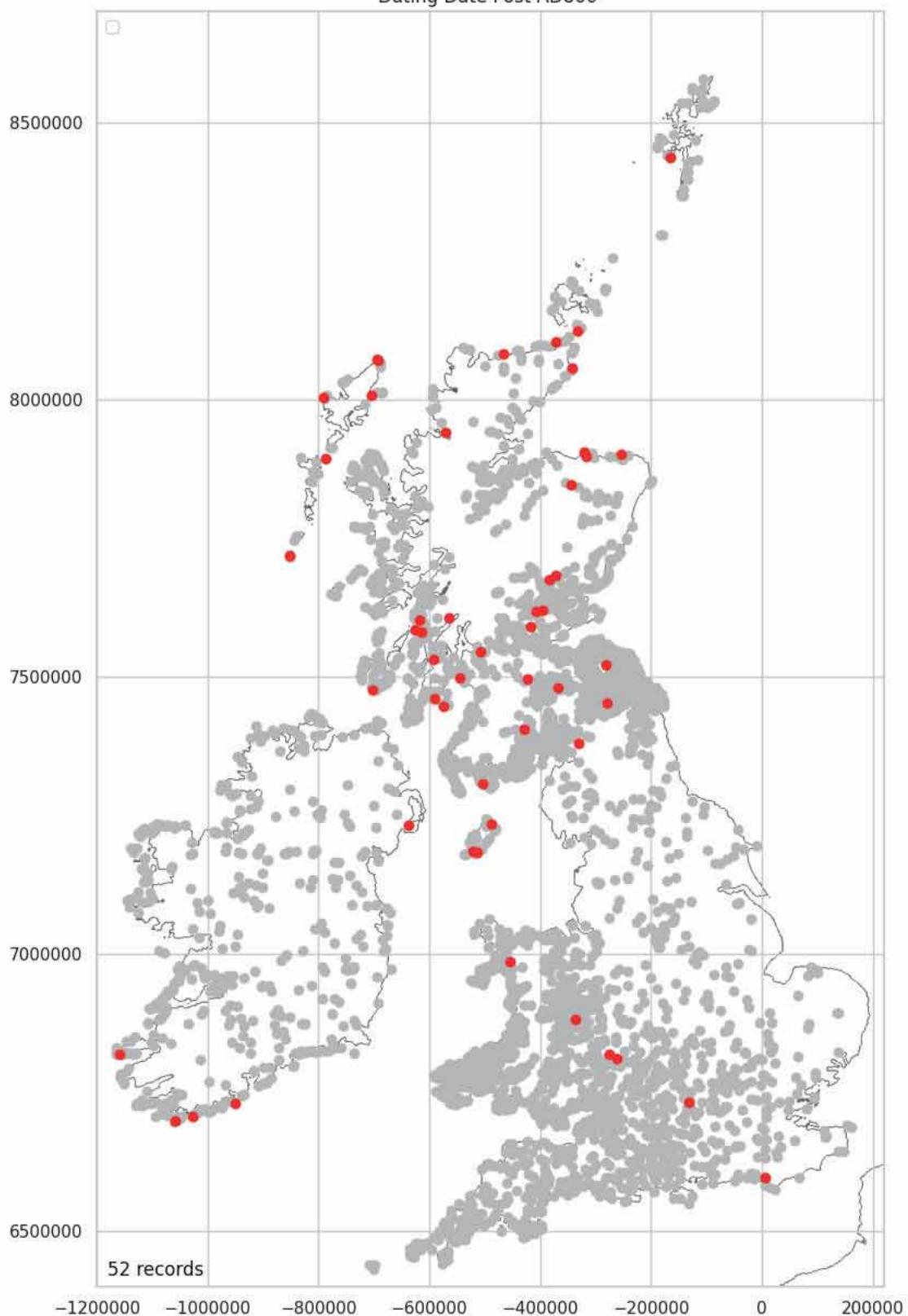
2.07%

Post AD800 Mapped

A small number of hillforts have dates post AD 800. Again, most are outside the area of excavation bias seen above. This may be an observation that is meaningful for the south of England.

```
In [ ]: dt_800 = plot_over_grey(location_date_data, 'Dating_Date_Post_AD800', 'Yes')
```

Dating Date Post AD800



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

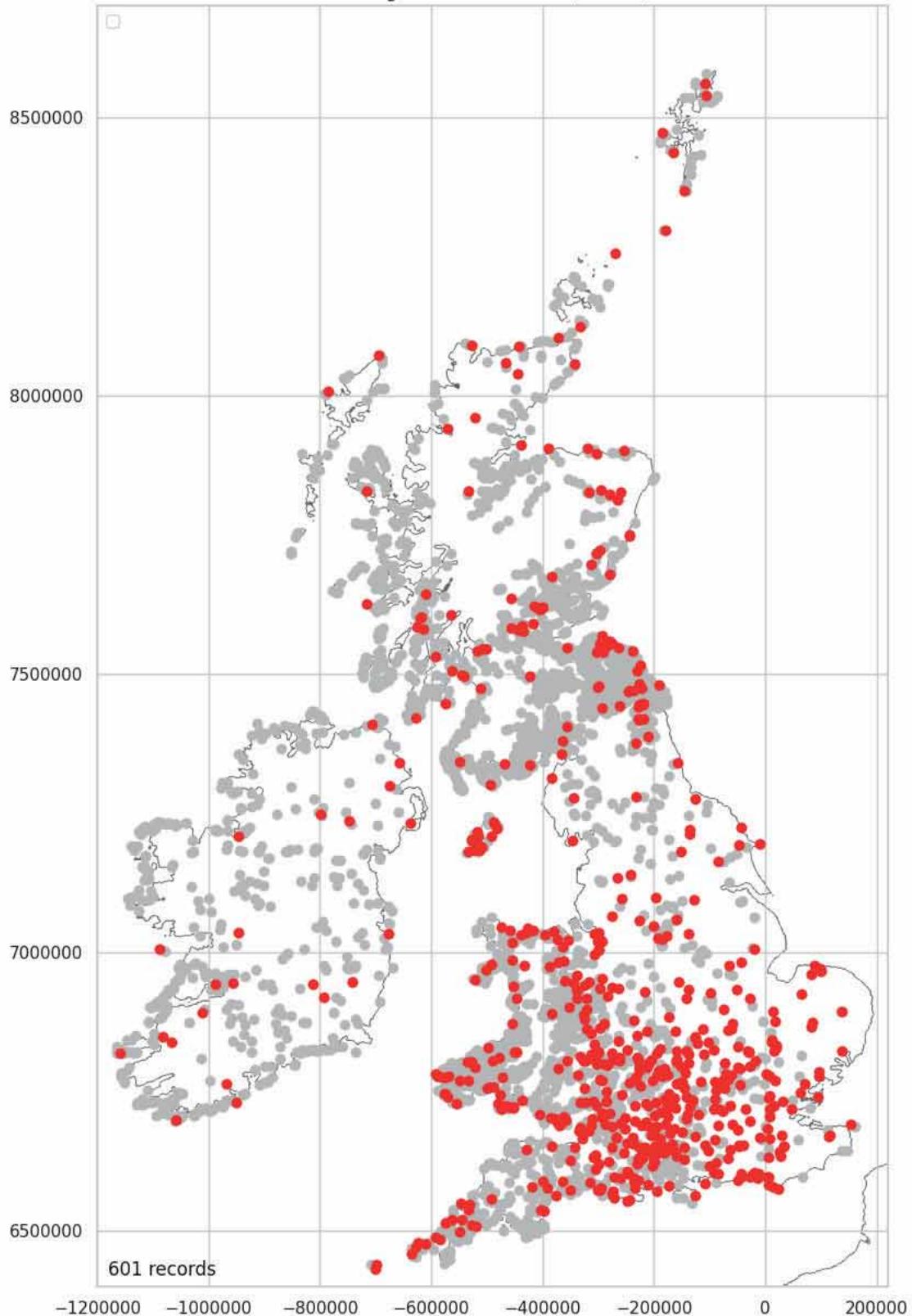
1. 25%

Date Known Mapped

Only 14.5% of hillforts have dating information and the majority of these are in the south of England.

```
In [ ]: dt_known = \
plot_over_grey(location_date_data, 'Dating_Date_Unknown', 'No', 'No (Known)', \
False, False, False, False)
```

Dating Date Unknown No (Known)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

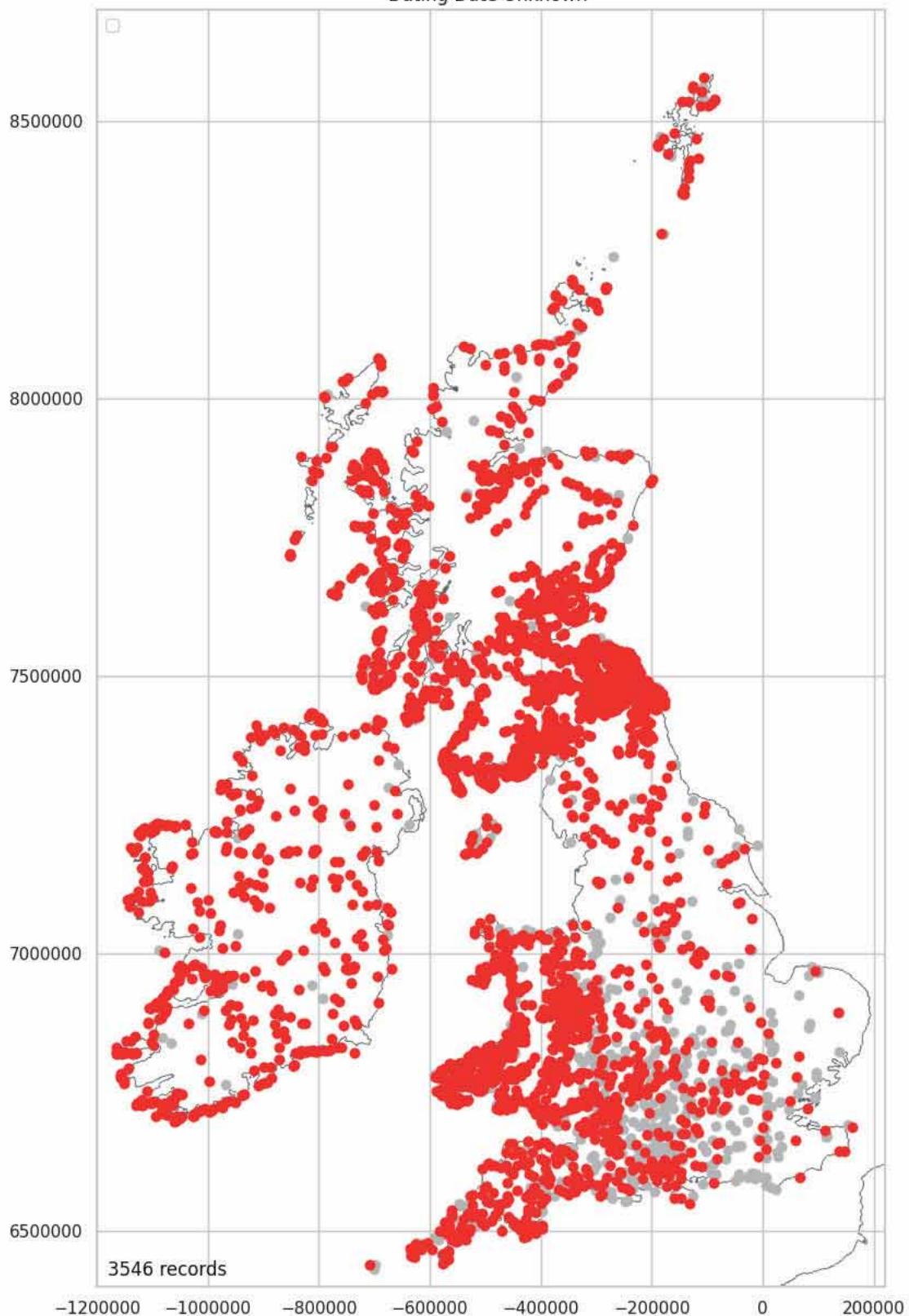
14.49%

Date Unknown Mapped

Most (85.5%) of hillforts have no dating evidence.

```
In [ ]: dt_unknown = plot_over_grey(location_date_data, 'Dating_Date_Unknown', 'Yes')
```

Dating Date Unknown



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

85. 51%

Dating Encodable Data - Resolve Null Values

There are two features containing null values

```
In [ ]: dating_encodeable_data[['Dating_Reliability', 'Related_Dating_Evidence']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Dating_Date_Reliability  4134 non-null   object 
 1   Related_Dating_Evidence  805 non-null   object 
dtypes: object(2)
memory usage: 64.9+ KB
```

Test for 'NA'.

```
In [ ]: test_cat_list_for_NA(dating_encodeable_data, \
['Dating_Date_Reliability', 'Related_Dating_Evidence'])
```

Dating_Date_Reliability 0
Related_Dating_Evidence 0

Update null values to 'NA'.

```
In [ ]: dating_encodeable_data = \
update_cat_list_for_NA(dating_encodeable_data, \
['Dating_Date_Reliability', 'Related_Dating_Evidence'])
```

```
In [ ]: dating_encodeable_data_review = \
test_numeric(dating_encodeable_data[['Dating_Date_Reliability', \
'Related_Dating_Evidence']])  
dating_encodeable_data_review
```

```
Out[ ]:
```

	Feature	Entries	Numeric	Non-Numeric	Null
0	Dating_Date_Reliability	4147	0	4147	0
1	Related_Dating_Evidence	4147	0	4147	0

Date Reliability

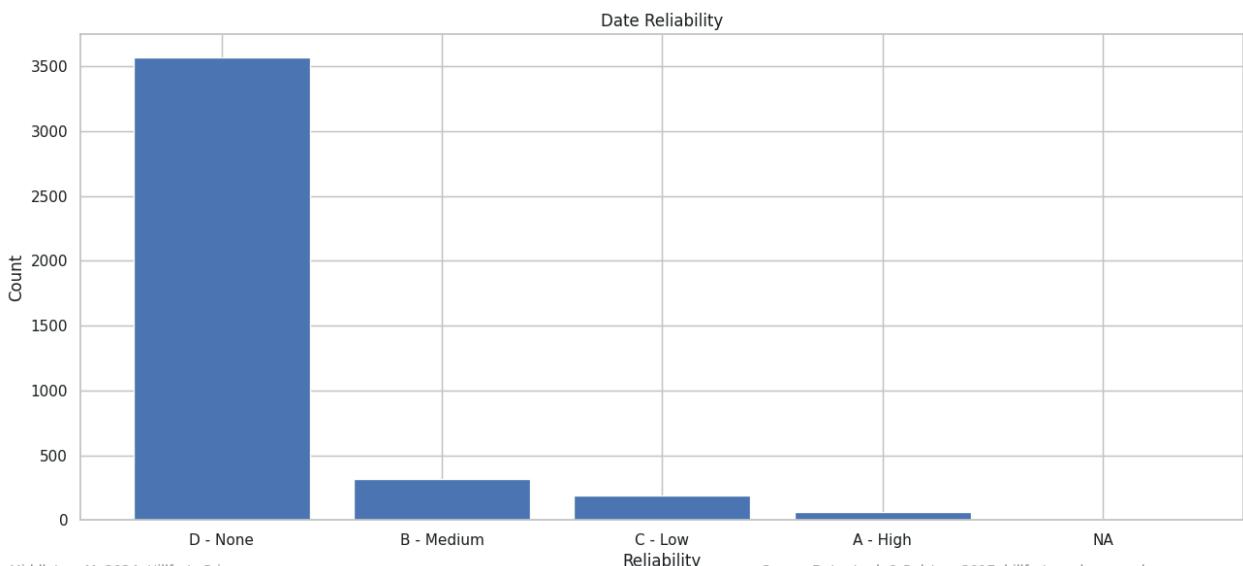
Data reliability contains five values.

```
In [ ]: pd.unique(dating_encodeable_data['Dating_Date_Reliability'])
```

```
Out[ ]: array(['C - Low', 'D - None', 'B - Medium', 'A - High', 'NA'],
              dtype=object)
```

The majority of hillforts have no dating reliability recorded as most hillforts do not have dating evidence. Of those that do have dating evidence, only 62 have dates that are classified as highly reliable.

```
In [ ]: plot_bar_chart_value_counts(dating_encodeable_data['Dating_Date_Reliability'], \
'Reliability', 'Count', 'Date_Reliability')
```



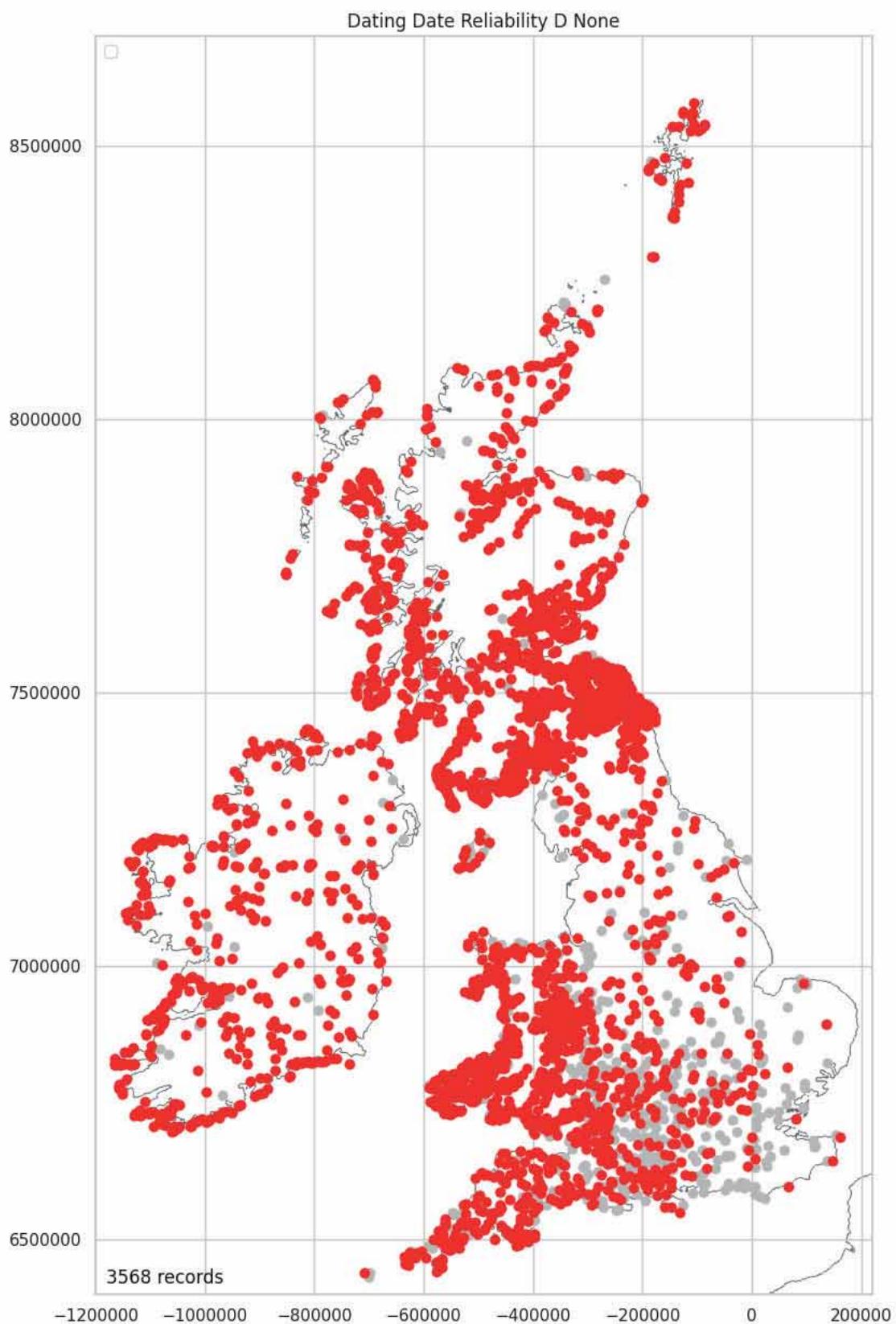
Date Reliability Mapped (D - None)

Most hillforts have no reliable dating evidence.

```
In [ ]: location_dating_encodeable_data = \
pd.merge(location_numeric_data_short, dating_encodeable_data, left_index=True, \
right_index=True)
```

```
right_index=True)
```

```
In [ ]: dating_reliability_none = \
location_dating_encodeable_data[location_dating_encodeable_data\
['Dating_Date_Reliability']=='D - None'].copy()
plot_over_grey_numeric(dating_reliability_none, 'Dating_Date_Reliability', \
'Dating_Date_Reliability_D-None')
```



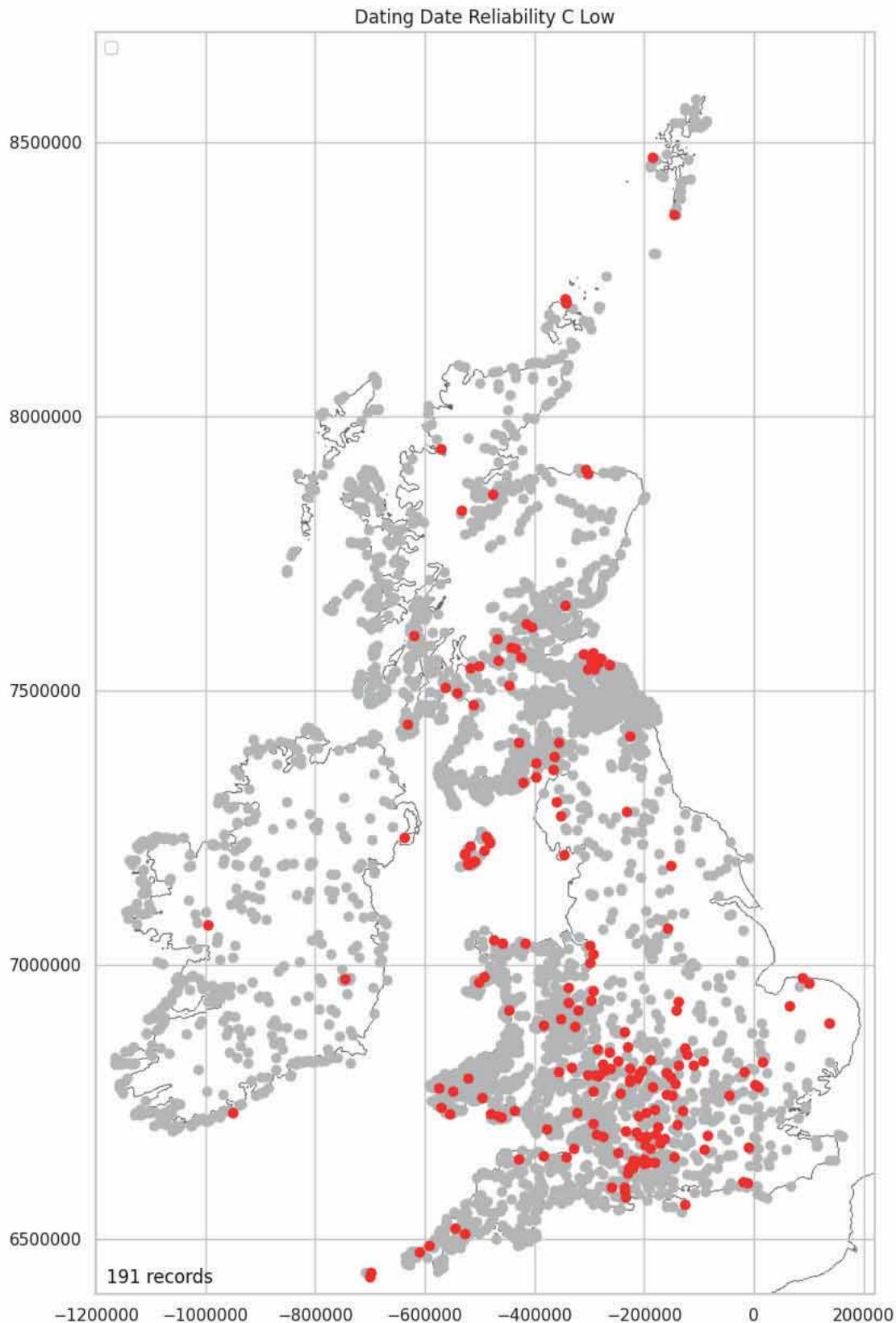
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Date Reliability Mapped (C - Low)

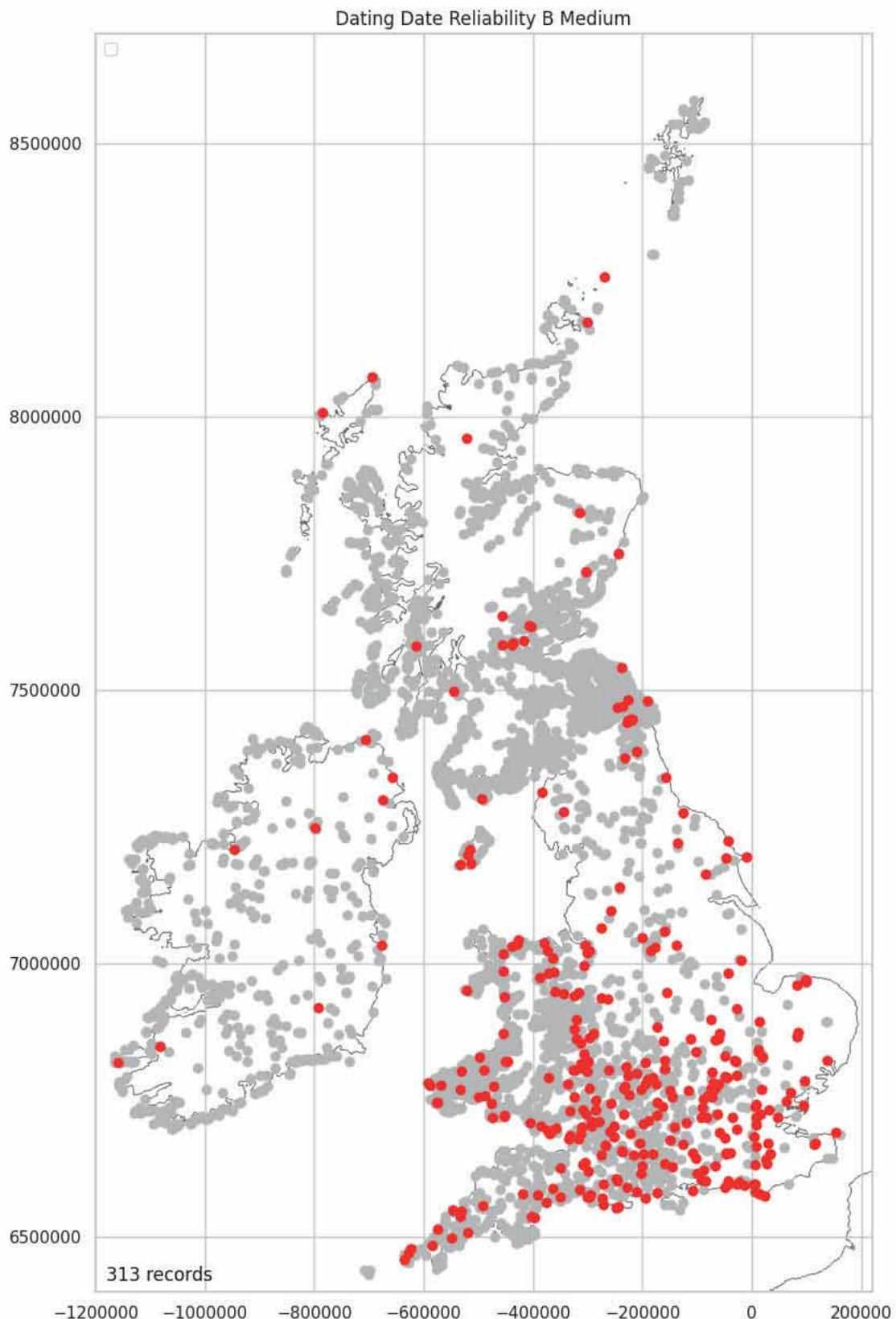
Of the hillforts that have dating evidence, only a relatively few have dating that is classified as low reliability. It should be noted that the distributions based on dating reliability suffer from the same excavation bias discussed above.

```
In [ ]: dating_reliability_low = \
location_dating_encodeable_data[location_dating_encodeable_data\
['Dating_Date_Reliability']=='C - Low'].copy()
plot_over_grey_numeric(dating_reliability_low, 'Dating_Date_Reliability', \
'Dating_Date_Reliability_C-Low')
```



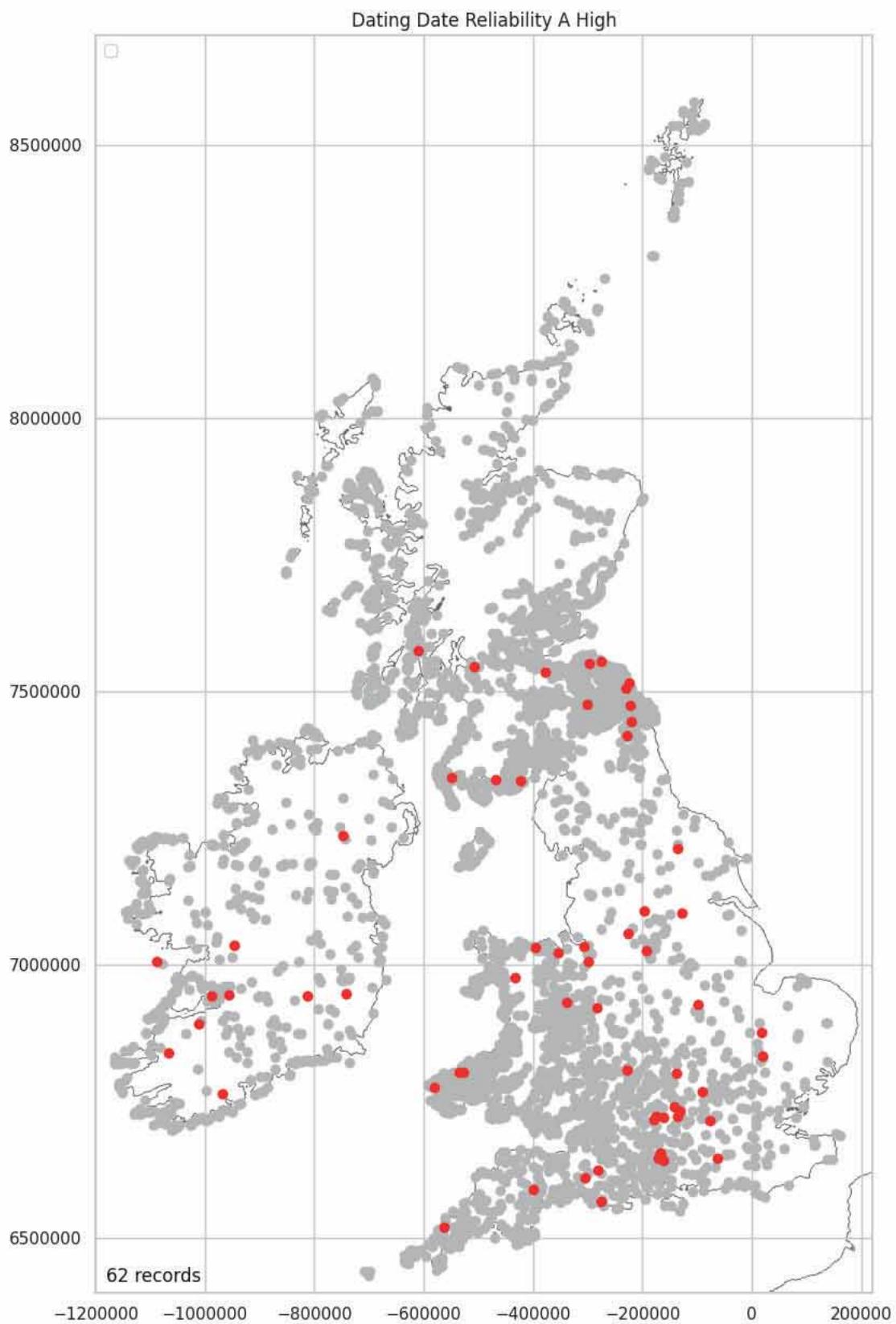
Date Reliability Mapped (B - Medium)

```
In [ ]: dating_reliability_med = \
location_dating_encodeable_data[location_dating_encodeable_data\
['Dating_Date_Reliability']=='B - Medium'].copy()
plot_over_grey_numeric(dating_reliability_med, 'Dating_Date_Reliability', \
'Dating_Date_Reliability_B-Medium')
```



Date Reliability Mapped (A - High)

```
In [ ]: dating_reliability_high = \
location_dating_encodeable_data[location_dating_encodeable_data\
['Dating_Date_Reliability']=='A - High'].copy()
plot_over_grey_numeric(dating_reliability_high, 'Dating_Date_Reliability', \
'Dating_Date_Reliability_A-High')
```



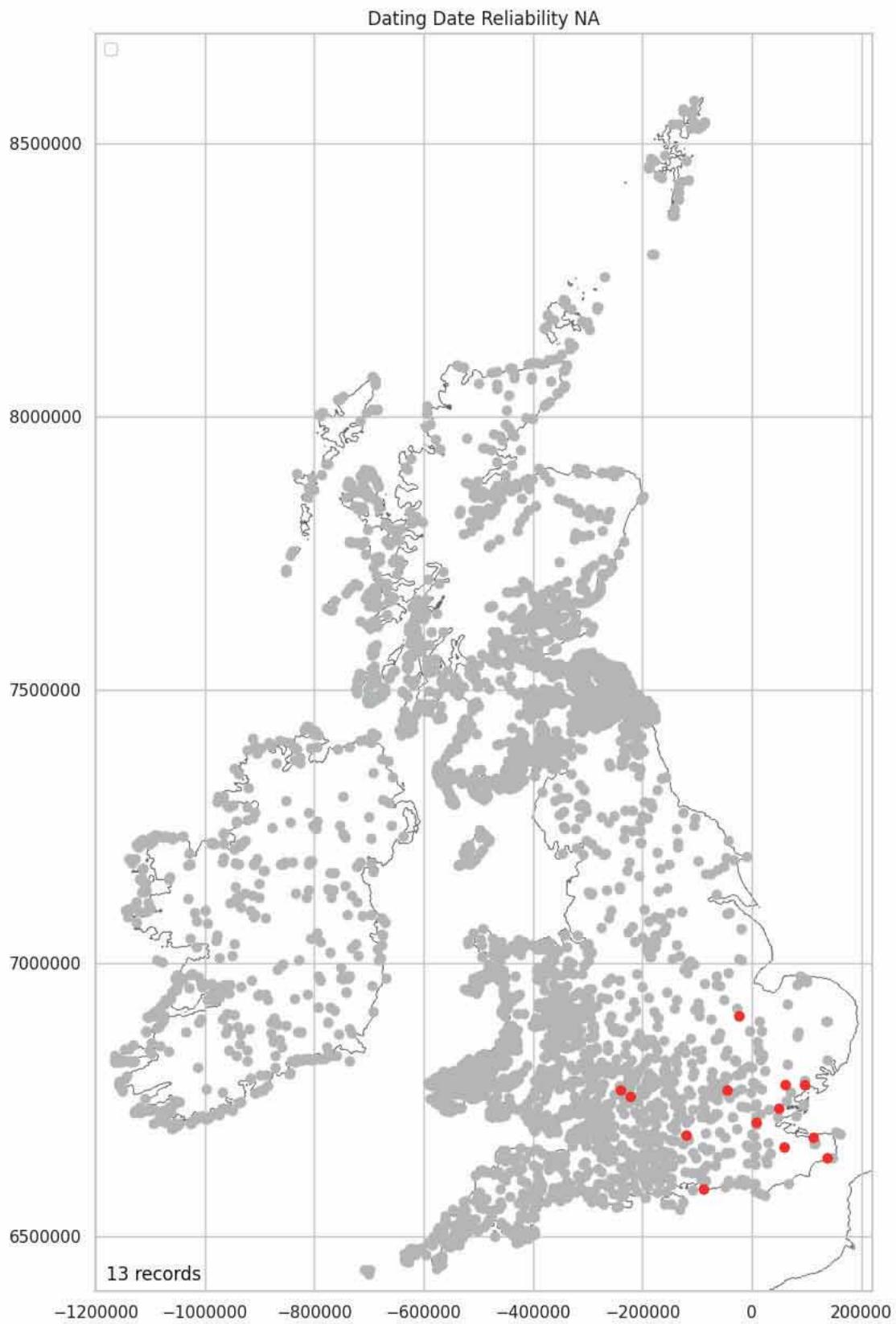
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Date Reliability Mapped (NA)

Thirteen records have no information recorded for date reliability.

```
In [ ]: dating_reliability_na = \
location_dating_encodeable_data[location_dating_encodeable_data\
['Dating_Date_Reliability']=='NA'].copy()
plot_over_grey_numeric(dating_reliability_na, 'Dating_Date_Reliability', \
'Dating_Date_Reliability_NA')
```



When this data is encoded the current values will be added as feature/ column headings. The current format of the data, with spaces and hyphen, could lead to problems so it is simplified.

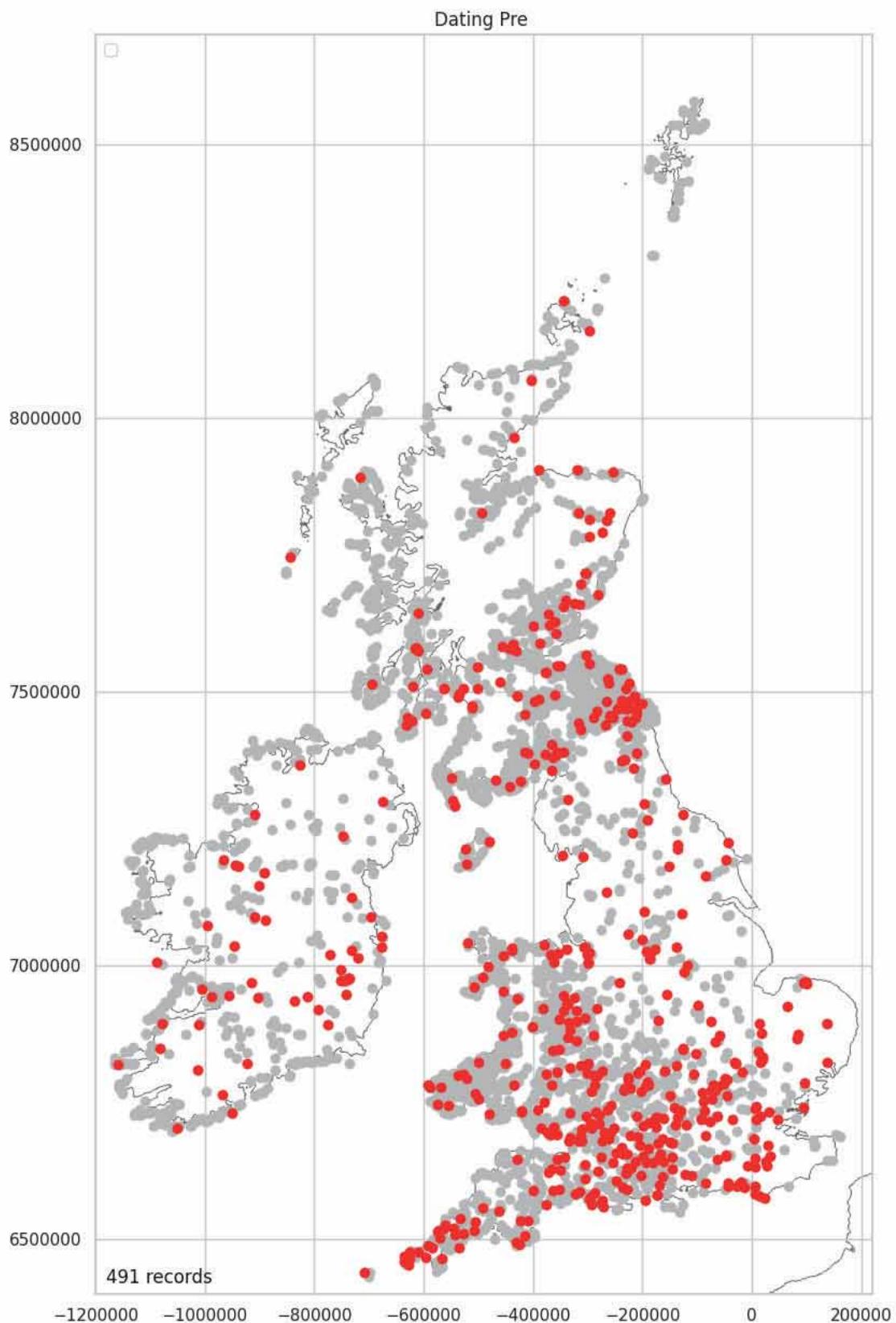
```
In [ ]: dating_encodeable_data['Dating_Date_Reliability'] = \
    np.where(dating_encodeable_data['Dating_Date_Reliability'] != 'NA', \
        dating_encodeable_data['Dating_Date_Reliability'].astype(str).str[0], \
        dating_encodeable_data['Dating_Date_Reliability'])
pd.unique(dating_encodeable_data['Dating_Date_Reliability'])

Out[ ]: array(['C', 'D', 'B', 'A', 'NA'], dtype=object)
```

Dating Pre

This feature records if there is activity on site prior to the construction of the hillfort.

```
In [ ]: dating_pre_stats = \
plot_over_grey(location_dating_encodeable_data, 'Dating_Pre', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

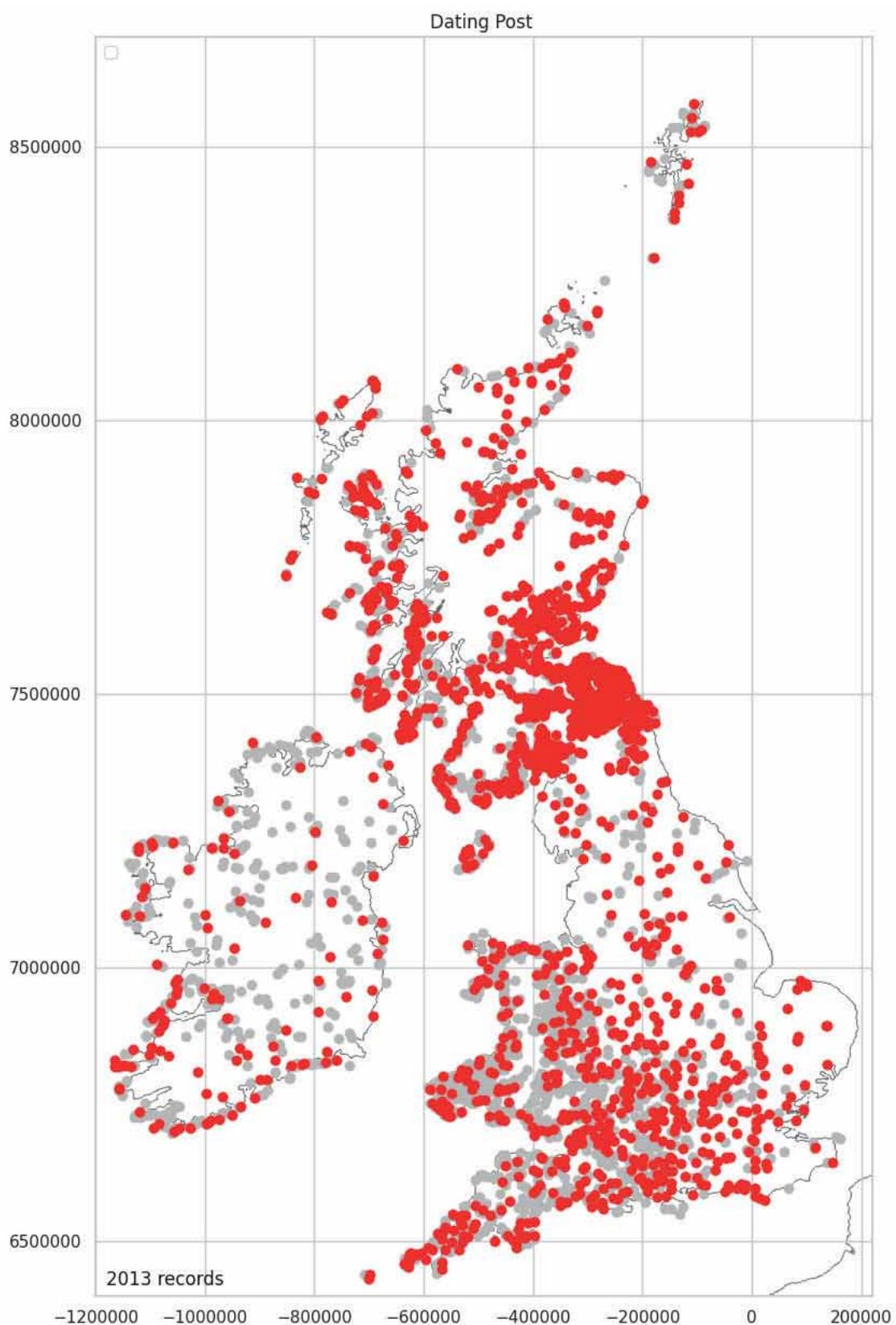
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

11.84%

Dating Post

This feature records if there is activity on site post the abandon of the hillfort.

```
In [ ]: dating_post_stats = plot_over_grey(location_dating_encodeable_data, \
'Dating_Post', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

48. 54%

Dating by Region

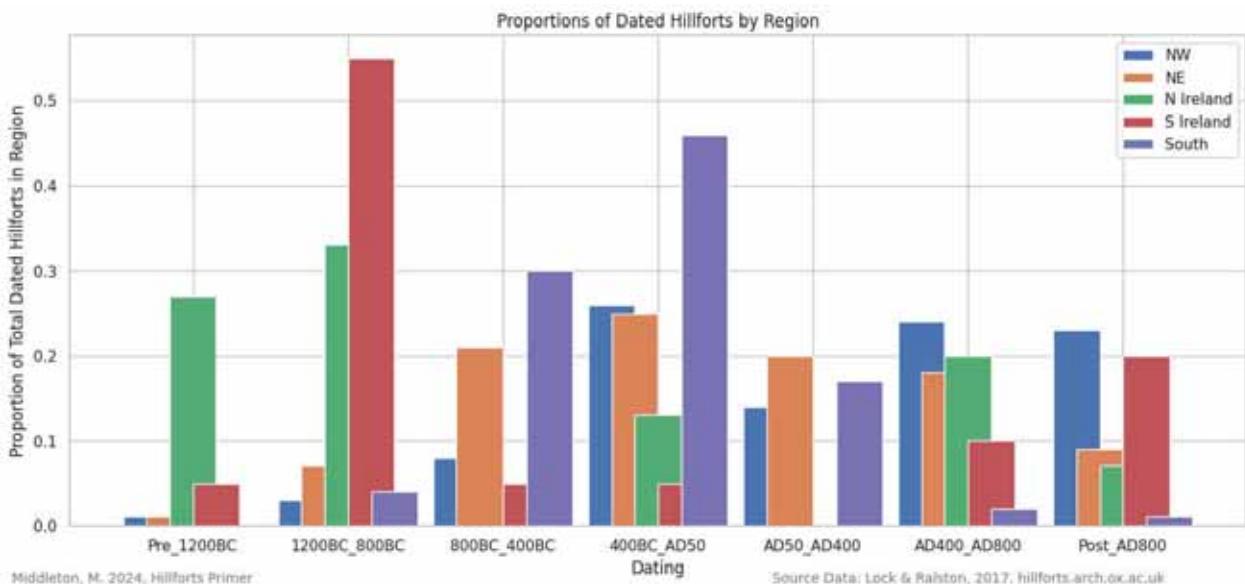
Only the south has sufficient data to produce a meaningful chart. The southern data shows peak between 400 BC and AD 50 with high concentrations of datable material on these sites running from 800 BC to AD 400. The dating in this region is predominantly derived from artifactual analysis. See: Map Related Dating Artefactual.

The Northeast and Northwest show a similar distribution of dates but differ in that they show signs of continued use beyond AD 800. Caution is needed in that this distribution of dates is based on a relatively small dataset.

The Irish data is quite different. There is a large peak from 1200 BC and 800 BC with the North of Ireland having a similarly large peak for dates pre 1200BC. There are almost no dates in the North of Ireland between 800 BC and 400 BC and only a small peak in South Ireland. There are very few dates in North or South Ireland between AD 50 and AD 400 and then increased activity, in terms of dating, from AD 400 on to post AD 800. There are very few dates for hillforts in Ireland. It is highly likely the distribution of dates just discussed could change radically as more dates become available. Extreme caution is needed when interpreting the distribution and spread of Irish dates.

```
In [ ]: location_enclosing_data_nw_dates = \
pd.merge(north_west, date_data, left_index=True, right_index=True)
location_enclosing_data_ne_dates = \
pd.merge(north_east, date_data, left_index=True, right_index=True)
location_enclosing_data_irland_n_dates = \
pd.merge(north_irland, date_data, left_index=True, right_index=True)
location_enclosing_data_irland_s_dates = \
pd.merge(south_irland, date_data, left_index=True, right_index=True)
location_enclosing_data_south_dates = \
pd.merge(south, date_data, left_index=True, right_index=True)
```

```
In [ ]: plot_dates_by_region(location_enclosing_data_nw_dates, \
location_enclosing_data_ne_dates, \
location_enclosing_data_irland_n_dates, \
location_enclosing_data_irland_s_dates, \
location_enclosing_data_south_dates, date_features)
```



Related Dating Evidence

```
In [ ]: list(pd.unique(dating_encodeable_data['Related_Dating_Evidence']))
```

```
Out[ ]: ['Artefactual',
 'NA',
 'Morphology/Earthwork/Typology',
 'Artefactual; C14; Morphology/Earthwork/Typology',
 'Artefactual; C14',
 'C14',
 'Artefactual; Morphology/Earthwork/Typology',
 'Morphology/Earthwork/Typology; Other',
 'C14; Morphology/Earthwork/Typology',
 'Other',
 'Artefactual; Other',
 'Artefactual; C14; Other',
 'C14; Morphology/Earthwork/Typology; Other',
 'C14; Other',
 'Artefactual; C14; Artefactual; C14',
 'Artefactual; Morphology/Earthwork/Typology; Other']
```

A gazetteer of five terms (classes) has been used - see below. These terms exclude 'NA' which was added by this study to replace null values. One hillfort may be associated with multiple dating classes. In the distributions that follow the same locations may occur in multiple dating plots.

```
In [ ]: related_dating_terms = \
['Artefactual', 'NA', 'Morphology/Earthwork/Typology', 'C14', 'Other']
```

Add new Related Dating Evidence Features

To enable the Related Dating Evidence to be interrogated more simply, five new, boolean, 'Yes/No' columns will be added to the encodeable data. Initially, they will be set to 'No' and then updated to 'Yes' if the term is found in the current 'Related_Dating_Evidence' feature. Note that the forward slash '/' will be removed from the column heading as this can cause problems.

```
In [ ]: dating_encodeable_data_plus = dating_encodeable_data.copy()
additional_related_dating_features = \
['Related_Dating_Artefactual', 'Related_Dating_NA', \
'Related_Dating_Morph_Earth_Typo', 'Related_Dating_C14', \
'Related_Dating_Other']
for feature in additional_related_dating_features:
    dating_encodeable_data_plus[feature] = 'No'
dating_encodeable_data_plus[additional_related_dating_features].head()
```

```
Out[ ]:   Related_Dating_Artefactual  Related_Dating_NA  Related_Dating_Morph_Earth_Typo  Related_Dating_C14  Related_Dating_Other
0           No          No                  No          No          No
1           No          No                  No          No          No
2           No          No                  No          No          No
3           No          No                  No          No          No
4           No          No                  No          No          No
```

Add Related Dating Artefactual

Populate the 'Artifactual' column.

```
In [ ]: dating_encodeable_data_plus['Related_Dating_Artefactual'].\
loc[date_encodeable_data_plus['Related_Dating_Evidence'].str.\n    contains('Artefactual', case=False)] = 'Yes'
date_encodeable_data_plus[['Related_Dating_Evidence', \
'Related_Dating_Artefactual']].\
loc[date_encodeable_data_plus['Related_Dating_Evidence'].str.\n    contains('Artefactual', case=False)][5:10]
```

```
Out[ ]:   Related_Dating_Evidence  Related_Dating_Artefactual
11  Artefactual; C14; Morphology/Earthwork/Typology      Yes
12          Artefactual                      Yes
14          Artefactual                      Yes
16  Artefactual; C14                      Yes
18          Artefactual                      Yes
```

Add Related Dating Morphology/Earthwork/Typology

Populate the 'Morphology/Earthwork/Typology' column.

```
In [ ]: date_encodeable_data_plus['Related_Dating_Morph_Earth_Typo'].\
loc[date_encodeable_data_plus['Related_Dating_Evidence'].str.\n    contains('Morphology/Earthwork/Typology', case=False)] = 'Yes'
date_encodeable_data_plus[['Related_Dating_Evidence', \
'Related_Dating_Morph_Earth_Typo']].\
loc[date_encodeable_data_plus['Related_Dating_Evidence'].str.contains\
('Morphology/Earthwork/Typology', case=False)].head()
```

```
Out[ ]:   Related_Dating_Evidence  Related_Dating_Morph_Earth_Typo
4        Morphology/Earthwork/Typology      Yes
11  Artefactual; C14; Morphology/Earthwork/Typology      Yes
39  Artefactual; Morphology/Earthwork/Typology          Yes
47  Morphology/Earthwork/Typology; Other                Yes
60      C14; Morphology/Earthwork/Typology                Yes
```

Add Related Dating C14

Populate the 'C14' column.

```
In [ ]: dating_encodeable_data_plus['Related_Dating_C14'].loc\[  
    [dating_encodeable_data_plus['Related_Dating_Evidence'].str.\  
        contains('C14', case=False)] = 'Yes'  
dating_encodeable_data_plus[['Related_Dating_Evidence', 'Related_Dating_C14']].\  
loc[dating_encodeable_data_plus['Related_Dating_Evidence'].str.\  
    contains('C14', case=False)].head()
```

```
Out[ ]:
```

	Related_Dating_Evidence	Related_Dating_C14
11	Artefactual; C14; Morphology/Earthwork/Typology	Yes
16	Artefactual; C14	Yes
21	Artefactual; C14	Yes
23	C14	Yes
60	C14; Morphology/Earthwork/Typology	Yes

Add Related Dating Other

Populate the 'Other' column.

```
In [ ]: dating_encodeable_data_plus['Related_Dating_Other'].loc\[  
    [dating_encodeable_data_plus['Related_Dating_Evidence'].str.\  
        contains('Other', case=False)] = 'Yes'  
dating_encodeable_data_plus[['Related_Dating_Evidence', 'Related_Dating_Other']].\  
loc[dating_encodeable_data_plus['Related_Dating_Evidence'].str.\  
    contains('Other', case=False)].head()
```

```
Out[ ]:
```

	Related_Dating_Evidence	Related_Dating_Other
47	Morphology/Earthwork/Typology; Other	Yes
65	Other	Yes
70	Other	Yes
75	Other	Yes
154	Artefactual; Other	Yes

Add Related Dating NA

Populate the 'NA' column.

```
In [ ]: dating_encodeable_data_plus['Related_Dating_NA'].loc\[  
    [dating_encodeable_data_plus['Related_Dating_Evidence'].str.\  
        contains('NA', case=False)] = 'Yes'  
dating_encodeable_data_plus[['Related_Dating_Evidence', 'Related_Dating_NA']].\  
loc[dating_encodeable_data_plus['Related_Dating_Evidence'].str.\  
    contains('NA', case=False)].head()
```

```
Out[ ]:
```

	Related_Dating_Evidence	Related_Dating_NA
1	NA	Yes
2	NA	Yes
5	NA	Yes
8	NA	Yes
9	NA	Yes

Review New Related Dating Evidence Features

Review a sample of the new columns to confirm the features are as expected.

```
In [ ]: dating_encodeable_data_plus[['Related_Dating_Evidence'] +\  
    additonal_related_dating_features][6:12]
```

Out[]:	Related_Dating_Evidence	Related_Dating_Artefactual	Related_Dating_NA	Related_Dating_Morph_Earth_Typo	Related_Dating_C14
6	Artefactual	Yes	No	No	No
7	Artefactual	Yes	No	No	No
8	NA	No	Yes	No	No
9	NA	No	Yes	No	No
10	Artefactual	Yes	No	No	No
11	Artefactual; C14; Morphology/Earthwork/Typology	Yes	No	Yes	Yes

Drop Related Dating Evidence

The information in 'Related_Dating_Evidence' has now been migrated to the new features so the original feature can now be deleted.

```
In [ ]: dating_encodeable_data_plus = \
dating_encodeable_data_plus.drop(['Related_Dating_Evidence'], axis=1)
dating_encodeable_data_plus.head()
```

Out[]:	Dating_Date_Pre_1200BC	Dating_Date_1200BC_800BC	Dating_Date_800BC_400BC	Dating_Date_400BC_AD50	Dating_Date_AD50_AD400	D
0	No	No	Yes	Yes	Yes	Yes
1	No	No	No	No	No	No
2	No	No	No	No	No	No
3	No	No	No	No	No	Yes
4	No	No	Yes	Yes	Yes	Yes

Map New Related Dating Evidence Features

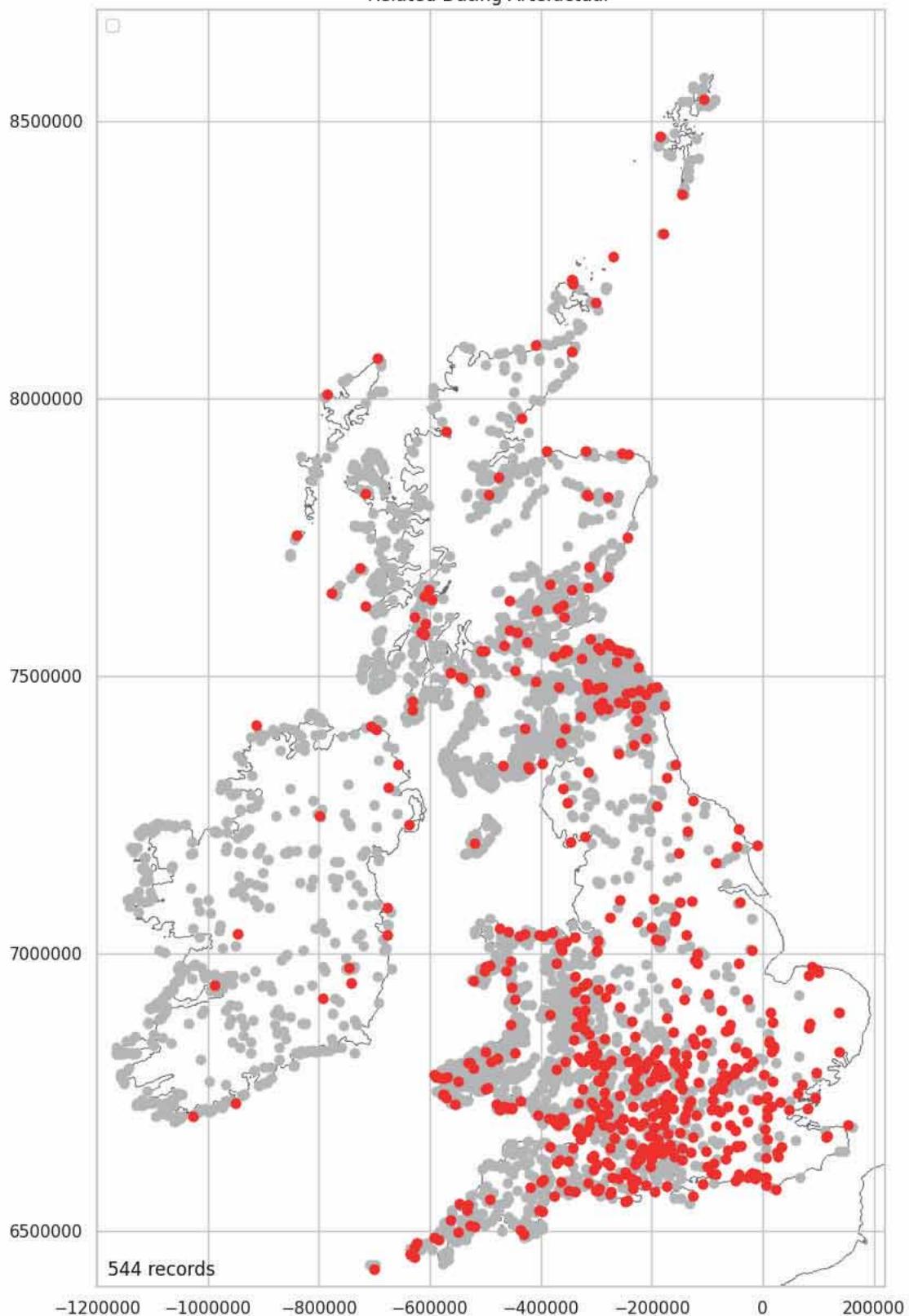
Map Related Dating Artefactual

Although most dated hillforts are dated using artefactual evidence, only 13.12% of hillforts have been dated in this way.

```
In [ ]: location_dating_encodeable_data = \
pd.merge(location_numeric_data_short, dating_encodeable_data_plus, \
left_index=True, right_index=True)

In [ ]: related_dating_artefactual_stats = \
plot_over_grey(location_dating_encodeable_data, \
'Related_Dating_Artefactual', 'Yes', '')
```

Related Dating Artefactual



Middleton, M. 2024, Hillforts Primer

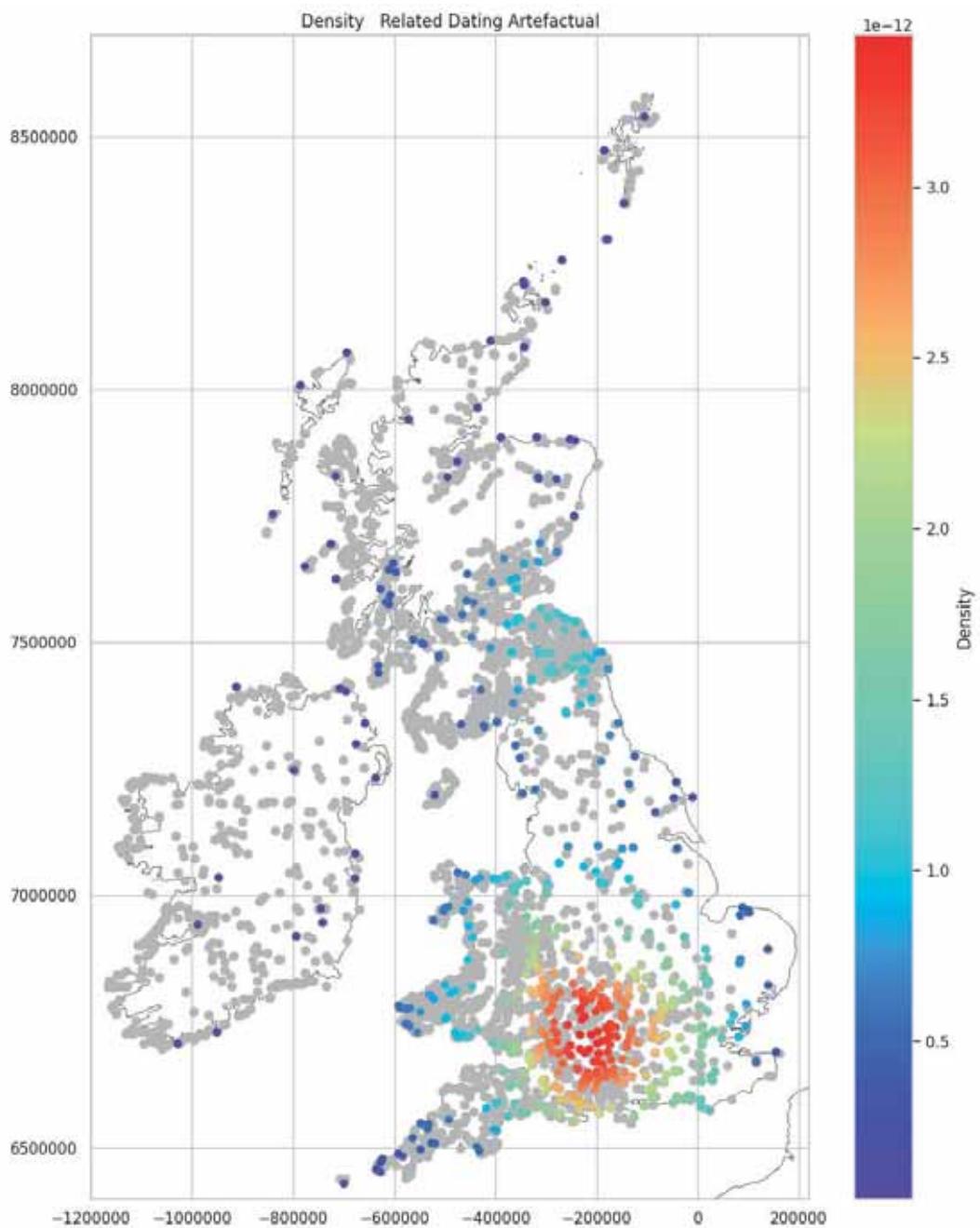
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

13. 12%

Map Related Dating Artefactual Density

There is a strong concentration of artifactual dates in south-central England and there is a significant reexcavation bias highlighted by this distribution

```
In [ ]: plot_density_over_grey(related_dating_artefactual_stats, \
                           'Related_Dating_Artefactual')
```



Middleton, M. 2024, Hillforts Primer

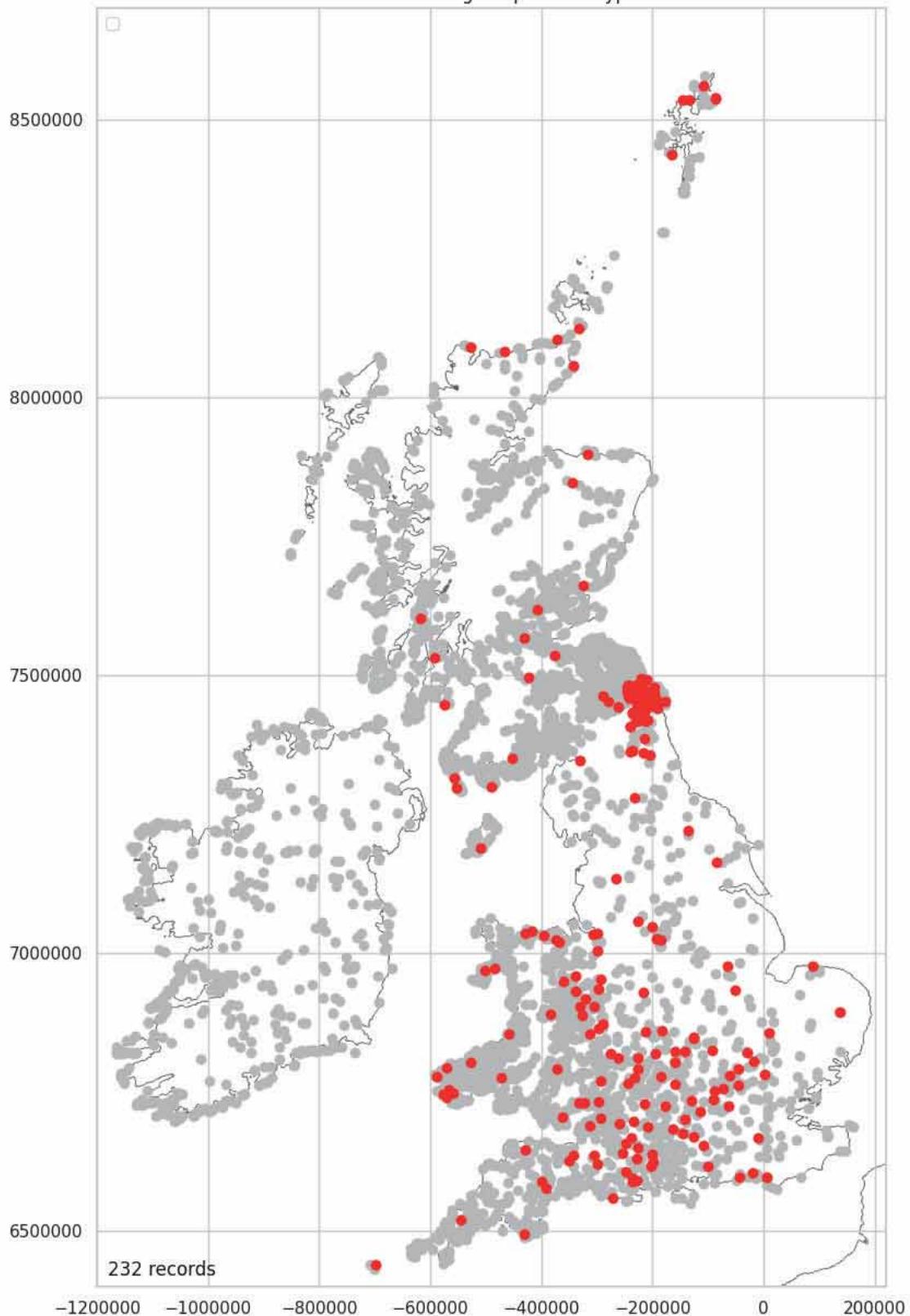
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Map Related Dating Morphology/Earthwork/Typology

Dating by means of morphology, earthwork and typology has a similar bias toward south central England and toward the northern border of Northumberland. Only 5.59% of hillforts have been dated in this way. Noteably, none are in Ireland.

```
In [ ]: met_stats = plot_over_grey(location_dating_encodeable_plus_data, \
                               'Related_Dating_Morph_Earth_Typo', 'Yes', '')
```

Related Dating Morph Earth Typo



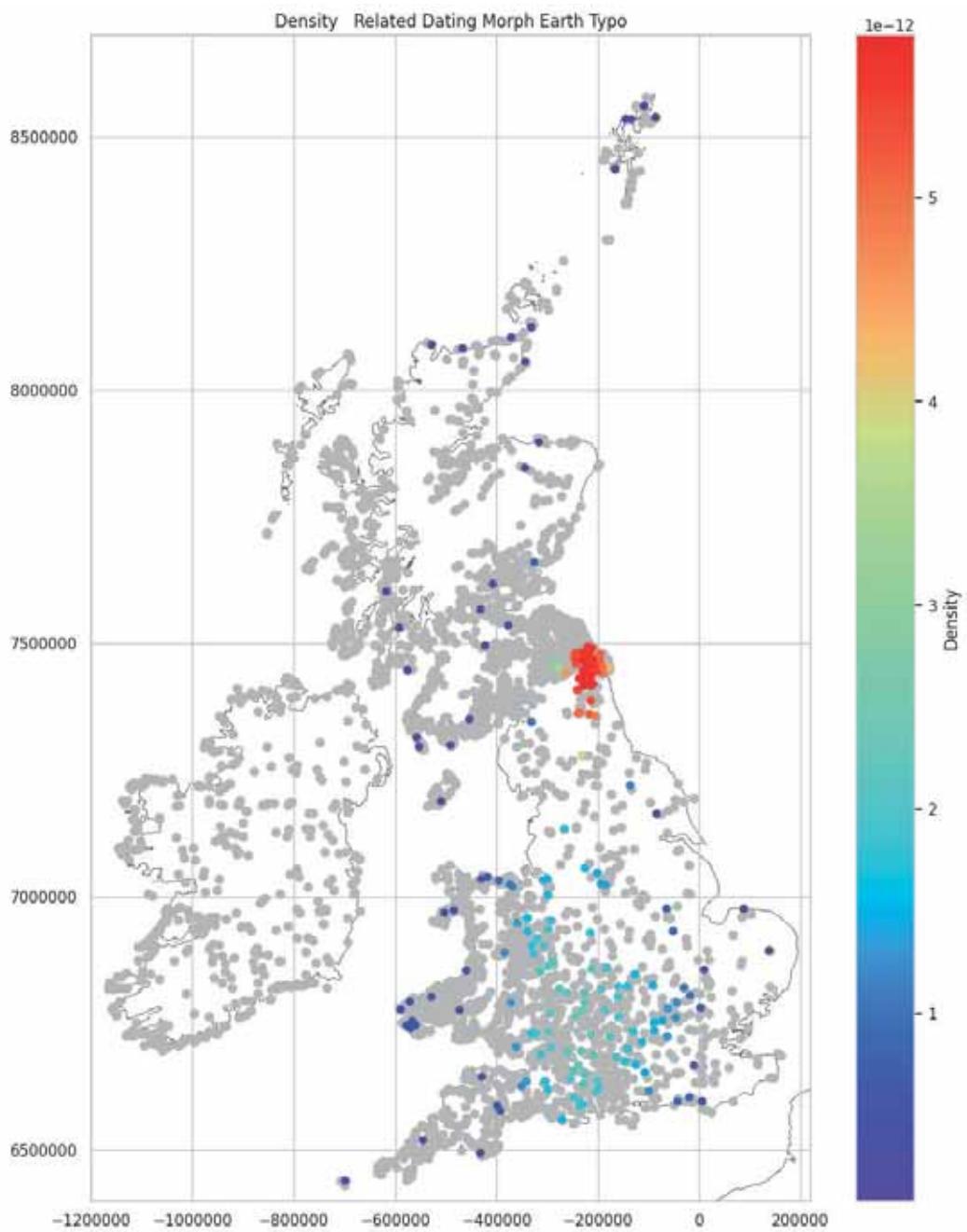
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5. 59%

Map Related Dating Morphology/Earthwork/Typology Density

```
In [ ]: plot_densitiy_over_grey(met_stats, 'Related_Dating_Morph_Earth_Typo')
```



Middleton, M. 2024, Hillforts Primer

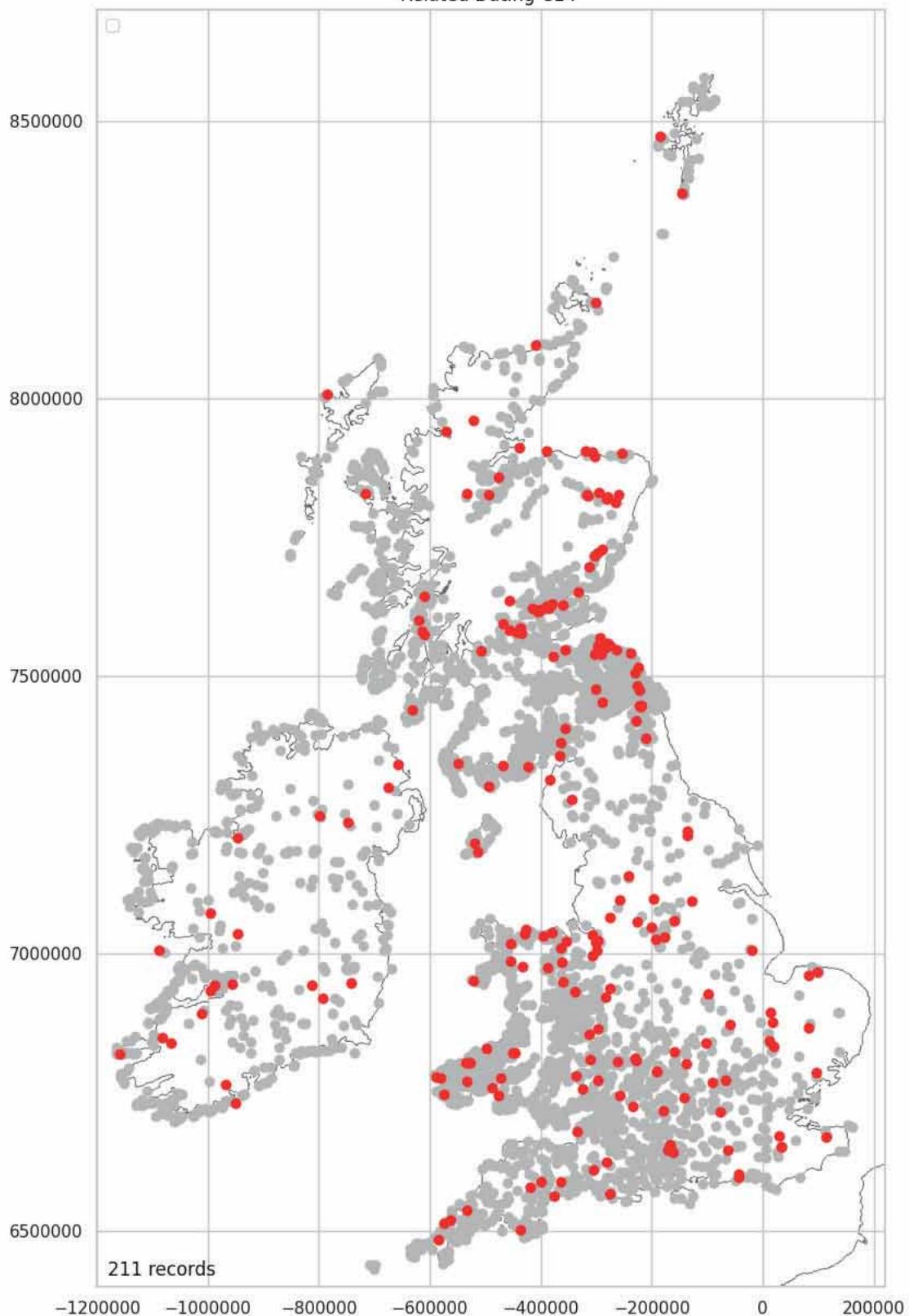
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Map Related Dating C14

Carbon 14 dating (C14) is the most scientifically rigorous of the dating techniques recorded in the atlas. Only 5.09% of hillforts have a C14 date but there does look to be a more even distribution of dates across the area of the atlas.

```
In [ ]: c14_stats = plot_over_grey(location_dating_encodeable_plots_data, \
    'Related_Dating_C14', 'Yes', '')
```

Related Dating C14



Middleton, M. 2024, Hillforts Primer

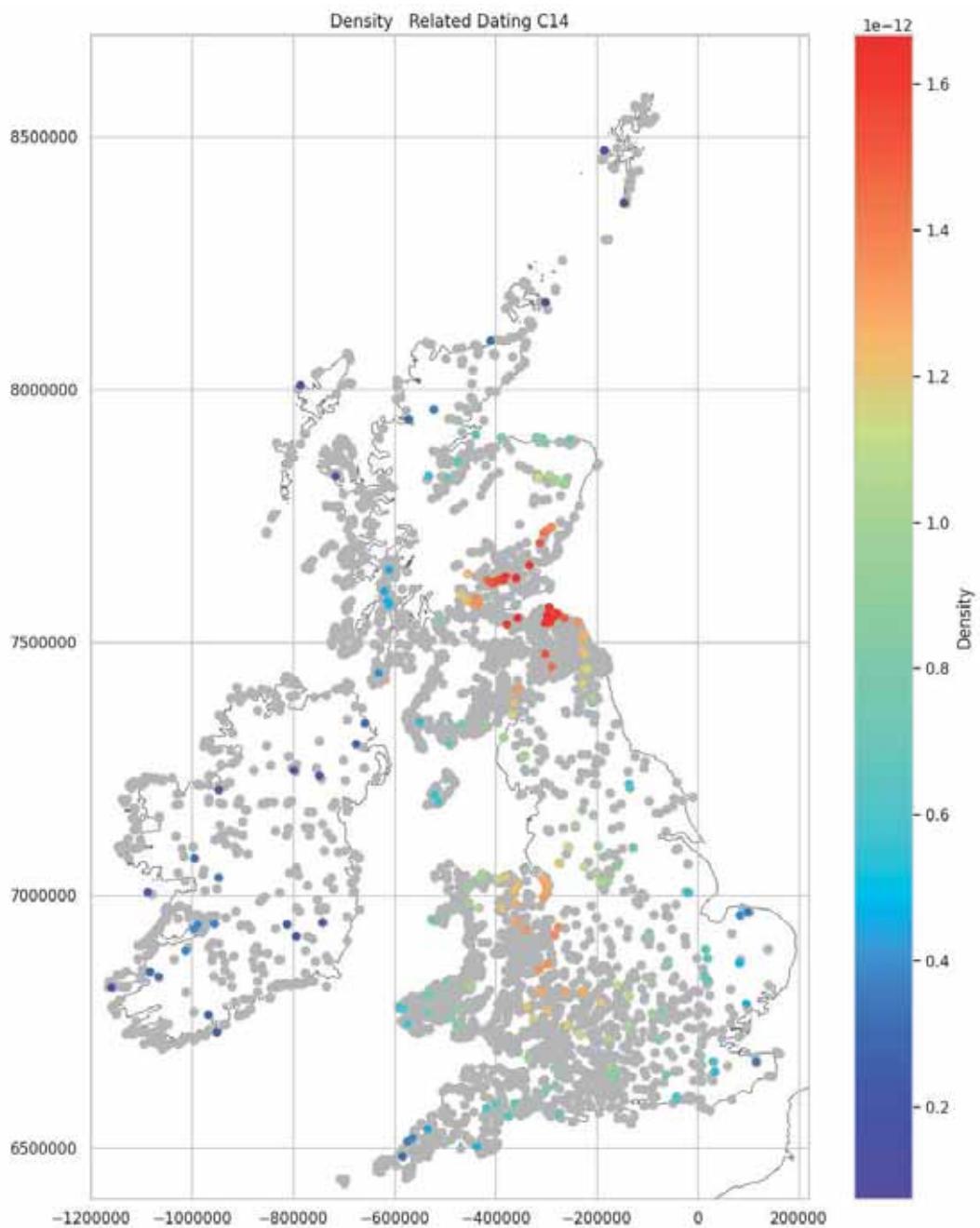
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5.09%

Map Related Dating C14 Density

There is a cluster of dates in the eastern Scottish lowlands, particularly around Traprain Law in East Lothian and along the line of the Gask Ridge. There is another, thin concentration, along the northern Welsh/English border, from the Shropshire Hills to the edge of Snowdonia.

```
In [ ]: plot_densiti_over_grey(c14_stats, 'Related_Dating_C14')
```



Middleton, M. 2024, Hillforts Primer

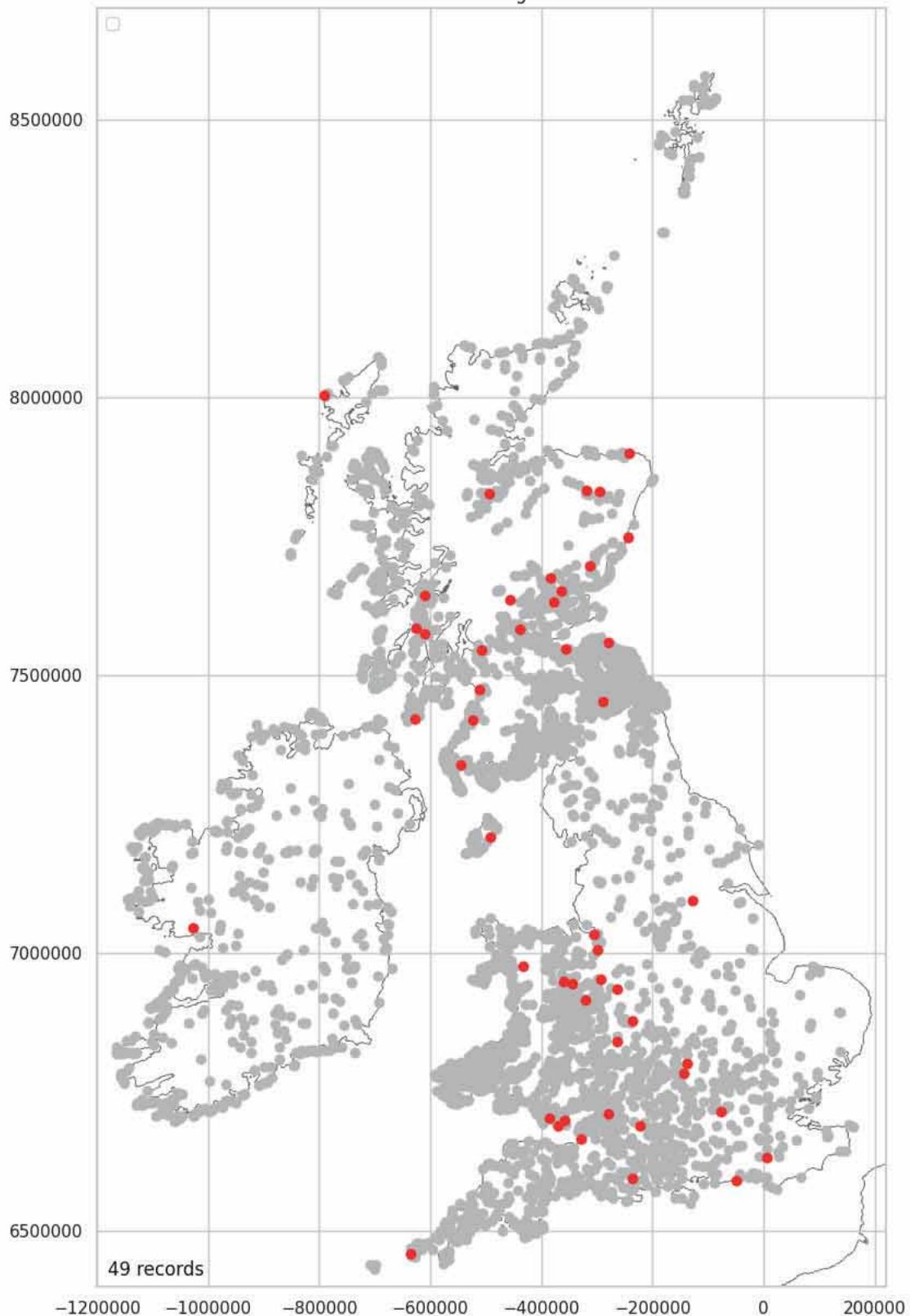
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Map Related Dating Other

Fourty nine hillforts (1.18%) of hillforts are identified as having 'other' dating evidence. No further information is available via the online data.

```
In [ ]: dating_other_stats = \
plot_over_grey(location_dating_encodeable_plus_data, 'Related_Dating_Other', \
    'Yes', '')
```

Related Dating Other



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

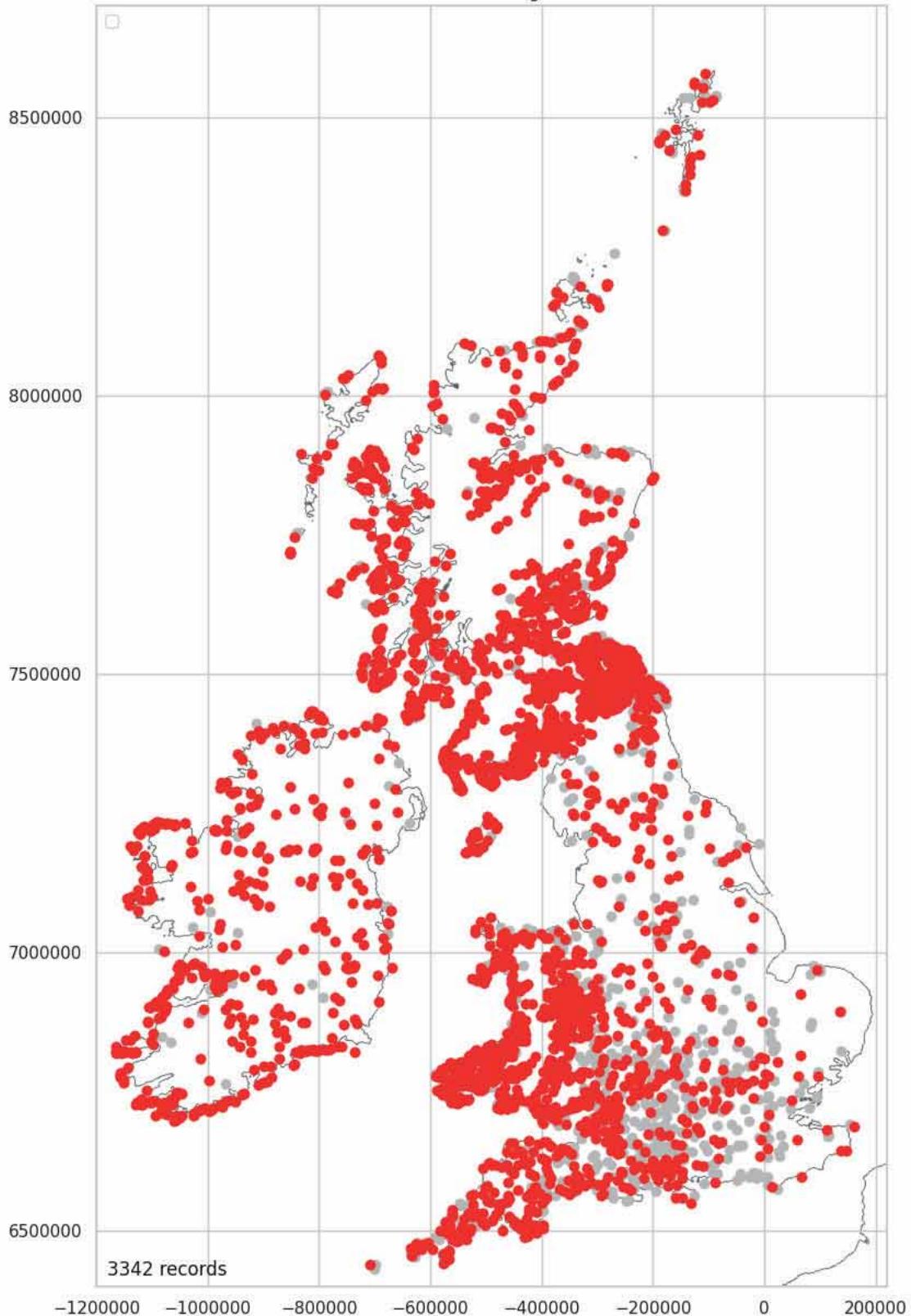
1. 18%

Map Related Dating NA

Most (80.59%) of hillforts have no dating evidence.

```
In [ ]: dating_na_stats = \
plot_over_grey(location_dating_encodeable_plus_data, 'Related_Dating_NA', \
'Yes', '')
```

Related Dating NA



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

80. 59%

Dating Data Package

```
In [ ]: dating_data_list = [dating_numeric_data, dating_text_data, \
                           dating_encodeable_data_plus]
```

Dating Data Download Package

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(dating_data_list, 'Dating_package')
```

Save Figure List

```
In [ ]: if save_images:  
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")  
    fig_list.to_csv(path, index=False)
```

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 4

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

- [Investigations Data](#)
- [Interior Data](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:
<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [1]: confirmed_only = False  
In [2]: download_data = False  
In [3]: save_images = False  
In [4]: show_topography = False
```

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Review Data Part 4](#).

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>
Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer
Licence: <https://creativecommons.org/licenses/by-sa/4.0/>
Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>
Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>
Hillforts: Britain, Ireland and the Nearer Continent (Sample):
<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Reload Data and Python Functions

This study is split over multiple documents. Each file needs to be configured and have the source data imported. As this section does not focus on the assessment of the data it is minimised to facilitate the documents readability.

Python Modules and Code Setup

The Python imports enable the Hillforts Atlas data to be analysed and mapped within this document. The Python code can be run on demand, (see: [User Settings](#)). This means that as new research becomes available, the source for this document can be updated to a revised copy of the Atlas data and the impact of that research can be reviewed using the same code and graphic output. The Hillforts Atlas is a baseline and this document is a tool that can be used to assess the impact new research is making in this area.

```
In [5]: import sys
print(f'Python: {sys.version}')

import sklearn
print(f'Scikit-Learn: {sklearn.__version__}')

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

import os
import collections
import math
import random
import PIL
import urllib
random.seed(42) # A random seed is used to ensure that the random numbers created are the same for each run of this
from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive
```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]

Scikit-Learn: 1.2.2

pandas: 1.5.3

numpy: 1.25.2

matplotlib: 3.7.1

seaborn: 0.13.1

scipy: 1.11.4

Ref: <https://www.python.org/>

Ref: <https://scikit-learn.org/stable/>

Ref: <https://pandas.pydata.org/docs/>

Ref: <https://numpy.org/doc/stable/>

Ref: <https://matplotlib.org/>

Ref: <https://seaborn.pydata.org/>

Ref: <https://docs.scipy.org/doc/scipy/index.html>

Ref: <https://pypi.org/project/python-slugify/>

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

```
In [6]: def show_records(plot, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plot.annotate(str(len(plot_data))+ ' records', xy=(-1180000, 6420000), xycoords='data', ha='left', color=text_colour)
```

```
In [7]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ['hillforts-topo-01.png',
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
                      "hillforts-topo-northwest-minus.png",
                      "hillforts-topo-northeast.png",
                      "hillforts-topo-south.png",
```

```

    "hillforts-topo-south-plus.png",
    "hillforts-topo-ireland.png",
    "hillforts-topo-ireland-north.png",
    "hillforts-topo-ireland-south.png"]
else:
    backgrounds = ["hillforts-outline-01.png",
                    "hillforts-outline-north.png",
                    "hillforts-outline-northwest-plus.png",
                    "hillforts-outline-northwest-minus.png",
                    "hillforts-outline-northeast.png",
                    "hillforts-outline-south.png",
                    "hillforts-outline-south-plus.png",
                    "hillforts-outline-ireland.png",
                    "hillforts-outline-ireland-north.png",
                    "hillforts-outline-ireland-south.png"]
return backgrounds

```

```
In [8]: def get_bounds():
bounds = [[-1200000, 220000, 6400000, 8700000],
[-1200000, 220000, 7000000, 8700000],
[-1200000, -480000, 7000000, 8200000],
[-900000, -480000, 7100000, 8200000],
[-520000, 0, 7000000, 8700000],
[-800000, 220000, 6400000, 7100000],
[-1200000, 220000, 6400000, 7100000],
[-1200000, -600000, 6650000, 7450000],
[-1200000, -600000, 7050000, 7450000],
[-1200000, -600000, 6650000, 7080000]]
return bounds
```

```
In [9]: def show_background(plt, ax, location=""):
backgrounds = get_backgrounds()
bounds = get_bounds()
folder = "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo/"

if location == "n":
    background = os.path.join(folder, backgrounds[1])
    bounds = bounds[1]
elif location == "nw+":
    background = os.path.join(folder, backgrounds[2])
    bounds = bounds[2]
elif location == "nw-":
    background = os.path.join(folder, backgrounds[3])
    bounds = bounds[3]
elif location == "ne":
    background = os.path.join(folder, backgrounds[4])
    bounds = bounds[4]
elif location == "s+":
    background = os.path.join(folder, backgrounds[5])
    bounds = bounds[5]
elif location == "s-":
    background = os.path.join(folder, backgrounds[6])
    bounds = bounds[6]
elif location == "i+":
    background = os.path.join(folder, backgrounds[7])
    bounds = bounds[7]
elif location == "in":
    background = os.path.join(folder, backgrounds[8])
    bounds = bounds[8]
elif location == "is":
    background = os.path.join(folder, backgrounds[9])
    bounds = bounds[9]
else:
    background = os.path.join(folder, backgrounds[0])
    bounds = bounds[0]

img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
ax.imshow(img, extent=bounds)
```

```
In [10]: def get_counts(data):
data_counts = []
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    data_counts.append(count)
return data_counts
```

```
In [11]: def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.01), \
                xycoords='figure fraction', horizontalalignment = 'right')
```

```
In [12]: def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.035), \
                xycoords='figure fraction', horizontalalignment = 'right')
```

```
In [13]: def plot_bar_chart(data, split_pos, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = data.columns
    x_data = [x.split("_")[split_pos:] for x in x_data]
    x_data_new = []
    for l in x_data :
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])
    y_data = get_counts(data)
    ax.bar(x_data_new, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [14]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [15]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, n_bins):
    new_data = data.copy()
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    data[x_label].plot(kind='hist', bins = n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [16]: def plot_bar_chart_value_counts(data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    df = data.value_counts()
    x_data = df.index.values
    y_data = df.values
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [17]: def get_bins(data, bins_count):
    data_range = data.max() - data.min()
    print(bins_count)
    if bins_count != None:
        x_bins = [x for x in range(data.min(), data.max(), bins_count)]
        n_bins = len(x_bins)
    else:
        n_bins = int(data_range)
        if n_bins < 10:
            multi = 10
            while n_bins < 10:
                multi *= 10
                n_bins = int(data_range * multi)
        elif n_bins > 100:
            n_bins = int(data_range)/10

    return n_bins
```

```
In [18]: def plot_histogram(data, x_label, title, bins_count = None):
    n_bins = get_bins(data, bins_count)
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    ax.set_ylabel('Count')
    plt.ticker_label_format(style='plain')
    plt.hist(data, bins=n_bins)
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [19]: def plot_continuous(data, x_label, title):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    plt.plot(data, linewidth=4)
    plt.ticker_label_format(style='plain')
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [20]: # box plot
from matplotlib.cbook import boxplot_stats
def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.ticker_label_format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v")
    else:
        sns.boxplot(data=data, orient="h")
    save_fig(feature + " Range")
    plt.show()

bp = boxplot_stats(data)

low = bp[0].get('whislo')
q1 = bp[0].get('q1')
median = bp[0].get('med')
q3 = bp[0].get('q3')
high = bp[0].get('whishi')

return [low, q1, median, q3, high]
```

```
In [21]: def location_XY_plot():
    plt.ticker_label_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 8700000)
    add_annotation_l_xy(plt)
```

```
In [22]: def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')
```

```
In [23]: def plot_over_grey_numeric(merged_data, a_type, title, extra="", inner=False, fringe=False, oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_l_legend = add_21Ha_fringe()
        patches.append(f_for_l_legend)
    if inner:
        i_for_l_legend = add_21Ha_line()
        patches.append(i_for_l_legend)
```

```

show_records(plt, plot_data)
plt.legend(loc='upper left', handles=patches)
plt.title(get_print_title(title))
save_fig(title)
plt.show()

```

In [24]:

```

def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    fig, ax = plt.subplots(figsize=(9.47, 15.33))
    show_background(plt, ax)
    location_XY_plot()
    add_grey(region='')
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    show_records(plt, plot_data)
    plt.title(get_print_title('Boundary_Type: ' + boundary_type))
    save_fig('Boundary_Type_' + boundary_type)
    plt.show()

```

In [25]:

```

def plot_density_over_grey(data, data_type):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_density(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], c=new_data['Density'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density')
    plt.title(get_print_title(f'Density - {data_type}'))
    save_fig(f'Density_{data_type}')
    plt.show()

```

In [26]:

```

def add_21Ha_line():
    x_values = [-367969, -344171, -263690, -194654, -130542, -119597, -162994, -265052], [-304545]
    y_values = [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 6663609], 6611780]
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, label = '≥ 21 Ha Line')
    add_to_legend = Line2D([0], [0], color='k', lw=15, alpha=0.25, label = '≥ 21 Ha Line')
    return add_to_legend

```

In [27]:

```

def add_21Ha_fringe():
    x_values = [-367969, -126771, 29679, -42657, -248650, -304545, -423647, -584307, -367969]
    y_values = [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, 6752378, 7019842]
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, label = '≥ 21 Ha Fringe')
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, label = '≥ 21 Ha Fringe')
    return add_to_legend

```

In [28]:

```

def add_oxford_swindon(oxford=False, swindon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_size = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_size, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swindon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_size, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches

```

In [29]:

```

def plot_over_grey(merged_data, a_type, yes_no, extra="", inner=False, fringe=False, oxford=False, swindon=False):
    # plots selected data over the grey dots. yes_no controls filtering the data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()
    print(f'round((len(plot_data)/len(merged_data)*100), 2) %')
    return plot_data

```

```
In [30]: def plot_type_values(data, data_type, title):
    new_data = data.copy()
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], c=new_data[data_type], cmap=cm.rainbow, s=25)
    plt.colorbar(label=data_type)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [31]: def bespoke_plot(plt, title):
    add_annotation_plot(plt)
    plt.title(label='plain')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [32]: def get_proportions(date_set):
    total = sum(date_set) - date_set[-1]
    newset = []
    for entry in date_set[:-1]:
        newset.append(round(entry/total, 2))
    return newset
```

```
In [33]: def plot_dates_by_region(nw, ne, ni, si, s, features):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = nw[features].columns
    x_data = [x.split('_')[2:] for x in x_data][:-1]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])

    set1_name = 'NW'
    set2_name = 'NE'
    set3_name = 'N Ireland'
    set4_name = 'S Ireland'
    set5_name = 'South'
    set1 = get_proportions(get_counts(nw[features]))
    set2 = get_proportions(get_counts(ne[features]))
    set3 = get_proportions(get_counts(ni[features]))
    set4 = get_proportions(get_counts(si[features]))
    set5 = get_proportions(get_counts(s[features]))

    X_axis_s = np.arange(len(x_data_new))

    budge = 0.25

    plt.bar(X_axis_s - 0.55 + budge, set1, 0.3, label=set1_name)
    plt.bar(X_axis_s - 0.4 + budge, set2, 0.3, label=set2_name)
    plt.bar(X_axis_s - 0.25 + budge, set3, 0.3, label=set3_name)
    plt.bar(X_axis_s - 0.1 + budge, set4, 0.3, label=set4_name)
    plt.bar(X_axis_s + 0.05 + budge, set5, 0.3, label=set5_name)

    plt.xticks(X_axis_s, x_data_new)
    plt.xlabel('Dating')
    plt.ylabel('Proportion of Total Dated Hillforts in Region')
    title = 'Proportions of Dated Hillforts by Region'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [34]: def plot_bar_chart_two(data_1, data_2, split_pos, x_label, y_label, title, proportion=False):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = data_1.columns
    x_data = [x.split('_')[split_pos:][0] for x in x_data]

    x_name = data_1.columns[0].split('_')[1]
    y_name = data_2.columns[0].split('_')[1]
    set1 = get_counts(data_1)
    set2 = get_counts(data_2)
    if proportion:
        set1_total = sum(set1)
        set2_total = sum(set2)
        set1_prop = [round((x/set1_total) * 100, 2) for x in set1]
        set2_prop = [round((x/set2_total) * 100, 2) for x in set2]
```

```

set1 = set1_prop[:]
set2 = set2_prop[:]

X_axis_s = np.arange(len(x_data))

plt.bar(X_axis - 0.2, set1, 0.4, label = x_name)
plt.bar(X_axis + 0.2, set2, 0.4, label = y_name)

plt.xticks(X_axis, x_data)
plt.xlabel(x_label)
plt.ylabel(y_label)
plt.title(title)
plt.legend()
add_annotation_plot(ax)
save_fig(title)
plt.show()

```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

```
In [35]: def test_numeric(data):
    temp_data = data.copy()
    col umns = data.col umns
    out_col s = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
    feat, ent, num, non, nul = [], [], [], [], []
    for col in col umns:
        if temp_data[col].dtype == 'object':
            feat.append(col)
            temp_data[col+'_num'] = temp_data[col].str.isnumeric()
            entries = temp_data[col].notnull().sum()
            true_count = temp_data[col+'_num'][temp_data[col+'_num'] == True].sum()
            null_count = temp_data[col].isna().sum()
            ent.append(entries)
            num.append(true_count)
            non.append(entries-true_count)
            nul.append(null_count)
        else:
            print(f'{col} {temp_data[col].dtype}')
    summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
    summary.col umns = out_col s
    return summary
```

```
In [36]: def find_duplicated(numeric_c_data, text_data, encodeable_data):
    d = False
    all_col umns = list(numeric_c_data.col umns) + list(text_data.col umns) + list(encodeable_data.col umns)
    duplicate = [item for item, count in collections.Counter(all_col umns).items() if count > 1]
    if duplicate:
        print(f"There are duplicate features: {duplicate}")
        d = True
    return d
```

```
In [37]: def test_data_split(main_data, numeric_c_data, text_data, encodeable_data):
    m = False
    split_features = list(numeric_c_data.col umns) + list(text_data.col umns) + list(encodeable_data.col umns)
    missing = list(set(main_data)-set(split_features))
    if missing:
        print(f"There are missing features: {missing}")
        m = True
    return m
```

```
In [38]: def review_data_split(main_data, numeric_c_data, text_data, encodeable_data = pd.DataFrame()):
    d = find_duplicated(numeric_c_data, text_data, encodeable_data)
    m = test_data_split(main_data, numeric_c_data, text_data, encodeable_data)
    if d != True and m != True:
        print("Data split good.")
```

```
In [39]: def find_duplicates(data):
    print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')
```

```
In [40]: def count_yes(data):
    total = 0
    for col in data.col umns:
        count = len(data[data[col] == 'Yes'])
        total+= count
        print(f'{col}: {count}')
    print(f'Total yes count: {total}')
```

Null Value Functions

The following functions will be used to update null values.

```
In [41]: def fill_nan_with_minus_one(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna(-1)
    return new_data

In [42]: def fill_nan_with_NA(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna("NA")
    return new_data

In [43]: def test_numeric_value_in_feature(feature, value):
    test = feature.isin([-1]).sum()
    return test

In [44]: def test_categorical_value_in_feature(dataframe, feature, value):
    test = dataframe[feature][dataframe[feature] == value].count()
    return test

In [45]: def test_cat_list_for_NA(dataframe, cat_list):
    for val in cat_list:
        print(val, test_categorical_value_in_feature(dataframe, val, 'NA'))

In [46]: def test_num_list_for_minus_one(dataframe, num_list):
    for val in num_list:
        feature = dataframe[val]
        print(val, test_numeric_value_in_feature(feature, -1))

In [47]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data

In [48]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data
```

Reprocessing Functions

```
In [49]: def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data
```

Save Image Functions

```
In [50]: fig_no = 0
part = 'Part04'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"
```

```
In [51]: def get_file_name(title):
    file_name = slugify(title)
    return file_name
```

```
In [52]: def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(",", "; ")
    return title
```

```
In [53]: def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no
```

```
In [54]: if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass

In [55]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Part_04_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}.{fig_extension}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution, bbox_inches='tight')
    else:
        pass
```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [56]: hillforts_csv = r"https://raw.githubusercontent.com/MikeDarsie/Hillforts-Primer/main/hillforts-atlas-source-data.csv"
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)

<ipython-input-56-2b53084ab660>:2: DtypeWarning: Columns (10, 12, 68, 83, 84, 85, 86, 165, 183) have mixed types. Specify dtype option on import or set low_memory=False.
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

	OBJECTID	Main_Atlas_Number	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Name	Main_Display_N
0	1	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Aconbury C Hereford (Aconbury Bea
1	2	2	EN	England	EN0002 Bach Camp, Herefordshire	Bach Camp	NaN	Bach C Hereford

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [57]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
                      'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.')

Using all 4147 record in the Hillforts Atlas.
```

Download Function

```
In [58]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', 'republic-of-ireland', 'northern-ireland', 'isle-of-man']:
                if pkg.shape[0] == hf_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
```

```

    else:
        dl = data_list[0]
        dl = dl.replace('\r', ' ', regex=True)
        dl = dl.replace('\n', ' ', regex=True)
        fn = 'hillforts_primer_' + filename
        fn = get_file_name(fn)
        dl.to_csv(fn+'.csv', index=False)
        files.download(fn+'.csv')
    else:
        pass

```

Reload Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-data packages. Where needed, these data extracts will be combined with Name and Number features to ensure the data can be understood and can, if needed, be concorded.

```
In [59]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Reload Location

```
In [60]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short)
location_numeric_data_short.head()
location_data = location_numeric_data_short.copy()
location_data.head()
```

	Location_X	Location_Y	Density
0	-303295	6798973	1.632859e-12
1	-296646	6843289	1.540172e-12
2	-289837	6808611	1.547729e-12
3	-320850	6862993	1.670548e-12
4	-261765	6810587	1.369981e-12

Reload Location Cluster Data Packages

```
In [61]: cluster_data = hillforts_data[['Location_X', 'Location_Y', 'Main_Country_Code']].copy()
cluster_data['Cluster'] = 'NA'
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != 'NI', 'I', inplace=True)
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != 'IR', 'I', inplace=True)

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000), 'North_Ireland', cluster_data['Cluster'])
north_ireland = cluster_data[cluster_data['Cluster'] == 'North_Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000), 'South_Ireland', cluster_data['Cluster'])
south_ireland = cluster_data[cluster_data['Cluster'] == 'South_Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] < 7070000), 'South', cluster_data['Cluster'])
south = cluster_data[cluster_data['Cluster'] == 'South'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & (cluster_data['Location_X'] >= -500))
```

```

north_east = cluster_data[cluster_data['Cluster'] == 'Northeast'].copy()
cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & (cluster_data['Location_X'] < -5000
)
north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()
temp_cluster_location_packages = [north_ireland, south_ireland, south, north_east, north_west]

cluster_packages = []
for pkg in temp_cluster_location_packages:
    pkg = pkg.drop(['Main_Country_Code'], axis=1)
    cluster_packages.append(pkg)

north_ireland, south_ireland, south, north_east, north_west = cluster_packages[0], cluster_packages[1], cluster_packages[2]

```

Review Data Part 4

Investigations Data

The Investigations Data comprises two lists of publication references. Interventions may be anything from mapping events to aerial photography to field observations. The detail for each publication reference is held in a separate Interventions Table. This can be downloaded from the Hillforts Atlas Rest Service API [here](#) or from this project's data store [here](#). The Interventions Table has not been analysed as part of the Hillforts Primer at this time.

```
In [62]: investigations_features = ['Investigations_Summary', 'Related_Investigations']
investigations_data = hillforts_data[investigations_features]
investigations_data.head()
```

```
Out[62]:
```

	Investigations_Summary	Related_Investigations
0	In Aubrey's Monumenta Britannica (1665-1693). ...	1st Identified Map Depiction (1888); Other (19...)
1	On 1st Ed. OS map (1888). Herefordshire Aerial...	1st Identified Map Depiction (1888); Other (20...)
2	On 1st Ed. OS map (1888). Herefordshire Counci...	1st Identified Map Depiction (1888); Other (2012)
3	In Aubrey's Monumenta Britannica (1665-1693). ...	1st Identified Map Depiction (1888); Other (20...)
4	In Aubrey's Monumenta Britannica (1665-1693). ...	Excavation (1879); Other (1879); 1st Identifie...

```
In [63]: investigations_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Investigations_Summary  3614 non-null   object 
 1   Related_Investigations  3986 non-null   object 
dtypes: object(2)
memory usage: 64.9+ KB
```

The interventions data contains null values.

Investigations Numeric Data

There is no numeric Investigations Data.

```
In [64]: investigations_numeric_data = pd.DataFrame()
```

Investigations Text Data

Both interventions features are text fields.

```
In [65]: investigations_text_data = investigations_data.copy()
```

Investigations Text Data - Resolve Null Values

Test for 'NA'.

```
In [66]: test_cat_list_for_NA(investigations_text_data, investigations_features)
```

```
Investigations_Summary 0  
Related_Investigations 0
```

Fill null values with 'NA'.

```
In [67]: investigations_text_data = update_cat_list_for_NA(investigations_text_data, investigations_features)  
investigations_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 4147 entries, 0 to 4146  
Data columns (total 2 columns):  
 #   Column           Non-Null Count  Dtype     
 ---  --     
 0   Investigations_Summary  4147 non-null  object    
 1   Related_Investigations 4147 non-null  object    
 dtypes: object(2)  
 memory usage: 64.9+ KB
```

Remove hidden characters for new line '\n' and carriage return 'r'.

```
In [68]: investigations_text_data = investigations_text_data.replace('\r', ' ', regex=True)  
investigations_text_data = investigations_text_data.replace('\n', ' ', regex=True)
```

A investigations sample record.

```
In [69]: record_no = 39  
s_summary = investigations_text_data['Investigations_Summary'][record_no]  
sample_summary = investigations_text_data['Related_Investigations'][record_no]  
  
print('Investigations_Summary' + ' record: ' + str(record_no))  
for pt in s_summary.split('.'):   
    if (pt.strip != ""):  
        print("\t" + part.strip())  
  
print('Related_Investigations' + ' record: ' + str(record_no))  
for pt in sample_summary.split(';'):   
    print("\t" + pt.strip())
```

Investigations_Summary record: 39
Part04
Part04
Part04
Part04
Part04
Part04
Part04
Part04
Related_Investigations record: 39
Other (1974)
Other (1981)
Other (1985)
Other (2012)
1st Identified Map Depiction (1885-1900)
Other (1993-2000)

Investigations Encodable Data

There is no encodeable Investigations Data.

```
In [70]: investigations_encodeable_data = pd.DataFrame()
```

Investigations Data Package

```
In [71]: investigations_data_list = [investigations_numeric_data, investigations_text_data, investigations_encodeable_data]
```

Investigations Data Download Package

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [72]: download(investigations_data_list, 'Investigations_package')
```

Interior Data

There are 37 Interior Data features which are subgrouped into:

- Water
- Surface
- Excavation
- Geophysics

```
In [73]: interior_features = [
    'Interior_Summary',
    'Interior_Water_None',
    'Interior_Water_Spring',
    'Interior_Water_Stream',
    'Interior_Water_Pool',
    'Interior_Water_Flush',
    'Interior_Water_Well',
    'Interior_Water_Other',
    'Interior_Water_Comments',
    'Interior_Surface_None',
    'Interior_Surface_Round',
    'Interior_Surface_Rectangular',
    'Interior_Surface_Curvilinear',
    'Interior_Surface_Roundhouse',
    'Interior_Surface_Pit',
    'Interior_Surface_Quarry',
    'Interior_Surface_Other',
    'Interior_Surface_Comments',
    'Interior_Excavation_None',
    'Interior_Excavation_Pit',
    'Interior_Excavation_Posthole',
    'Interior_Excavation_Roundhouse',
    'Interior_Excavation_Rectangular',
    'Interior_Excavation_Road',
    'Interior_Excavation_Quarry',
    'Interior_Excavation_Other',
    'Interior_Excavation_Nothing',
    'Interior_Excavation_Comments',
    'Interior_Geophysics_None',
    'Interior_Geophysics_Pit',
    'Interior_Geophysics_Roundhouse',
    'Interior_Geophysics_Rectangular',
    'Interior_Geophysics_Road',
    'Interior_Geophysics_Quarry',
    'Interior_Geophysics_Other',
    'Interior_Geophysics_Nothing',
    'Interior_Geophysics_Comments'
]

interior_data = hillforts_data[interior_features].copy()
interior_data.head()
```

	Interior_Summary	Interior_Water_None	Interior_Water_Spring	Interior_Water_Stream	Interior_Water_Pool	Interior_Water_Flush	Interior_
0	Little information about interior was gleaned ...	Yes	No	No	No	No	No
1	None	Yes	No	No	No	No	No
2	A number of cloudy blue flints, two burnt flin...	Yes	No	No	No	No	No
3	Possible hut circles 12m-15m in diameter were ...	Yes	No	No	No	No	No
4	At least 118 hut platforms have been identifie...	No	Yes	No	No	No	No

Interior Numeric Data

There is no numeric Investigations Data.

```
In [74]: interior_numeric_data = pd.DataFrame()
```

Interior Text Data

There are five text features which comprise a summary of the interior and four comments features; one relating to each subgroup listed above.

```
In [75]: interior_text_features = [
    'Interior_Summary',
    'Interior_Water_Comments',
    'Interior_Surface_Comments',
    'Interior_Excavation_Comments',
    'Interior_Geophysics_Comments']
```

```
interior_text_data = interior_data[interior_text_features].copy()
interior_text_data.head()
```

	Interior_Summary	Interior_Water_Comments	Interior_Surface_Comments	Interior_Excavation_Comments	Interior_Geophysics_Comments
0	Little information about interior was gleaned ...	Spring 0.3km located outside the hillfort	Little information is available from surface e...	NaN	NaN
1	None	Stream 0.1km located outside hillfort	NaN	NaN	NaN
2	A number of cloudy blue flints, two burnt flin...	Stream 0.7km located outside the hillfort.	NaN	NaN	NaN
3	Possible hut circles 12m-15m in diameter were ...	Stream 0.1km located outside the hillfort	Possible hut circles 12m-15m in diameter were ...	Roman occupation of Neronian date with militar...	NaN
4	At least 118 hut platforms have been identifie...	Possible spring within the bottom of the first...	At least 118 hut platforms have been identifie...	Excavation in 1879. Possibly hut platforms.	NaN

```
In [76]: interior_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Interior_Summary  4139 non-null   object 
 1   Interior_Water_Comments 980 non-null   object 
 2   Interior_Surface_Comments 1113 non-null   object 
 3   Interior_Excavation_Comments 498 non-null   object 
 4   Interior_Geophysics_Comments 233 non-null   object 
dtypes: object(5)
memory usage: 162.1+ KB
```

Interior Text Data - Resolve Null Values

Test for 'NA'.

```
In [77]: test_cat_list_for_NA(interior_text_data, interior_text_features)
```

```
Interior_Summary 0
Interior_Water_Comments 0
Interior_Surface_Comments 0
Interior_Excavation_Comments 0
Interior_Geophysics_Comments 0
```

Fill null values with 'NA'.

```
In [78]: interior_text_data = update_cat_list_for_NA(interior_text_data, interior_text_features)
interior_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Interior_Summary  4147 non-null   object 
 1   Interior_Water_Comments 4147 non-null   object 
 2   Interior_Surface_Comments 4147 non-null   object 
 3   Interior_Excavation_Comments 4147 non-null   object 
 4   Interior_Geophysics_Comments 4147 non-null   object 
dtypes: object(5)
memory usage: 162.1+ KB
```

Interior Encodable Data

Thirty two of the Internal Data features are encodeable. All are yes/no booleans.

```
In [79]: interior_encodeable_features = [
    'Interior_Water_None',
    'Interior_Water_Spring',
    'Interior_Water_Stream',
    'Interior_Water_Pool',
    'Interior_Water_Flush',
    'Interior_Water_Well',
    'Interior_Water_Other',
    'Interior_Surface_None',
    'Interior_Surface_Round',
    'Interior_Surface_Rectangular',
    'Interior_Surface_Curvilinear',
    'Interior_Surface_Roundhouse',
    'Interior_Surface_Pit',
    'Interior_Surface_Quarry',
    'Interior_Surface_Other',
    'Interior_Excavation_None',
    'Interior_Excavation_Pit',
    'Interior_Excavation_Posthole',
    'Interior_Excavation_Roundhouse',
    'Interior_Excavation_Rectangular',
    'Interior_Excavation_Road',
    'Interior_Excavation_Quarry',
    'Interior_Excavation_Other',
    'Interior_Excavation_Nothing',
    'Interior_Geophysics_None',
    'Interior_Geophysics_Pit',
    'Interior_Geophysics_Roundhouse',
    'Interior_Geophysics_Rectangular',
    'Interior_Geophysics_Road',
    'Interior_Geophysics_Quarry',
    'Interior_Geophysics_Other',
    'Interior_Geophysics_Nothing'
]
```

```
interior_encodeable_data = interior_data[interior_encodeable_features].copy()
interior_encodeable_data.head()
```

	Interior_Water_None	Interior_Water_Spring	Interior_Water_Stream	Interior_Water_Pool	Interior_Water_Flush	Interior_Water_Well	Interior_Water_Other
0	Yes	No	No	No	No	No	No
1	Yes	No	No	No	No	No	No
2	Yes	No	No	No	No	No	No
3	Yes	No	No	No	No	No	No
4	No	Yes	No	No	No	No	No

Water Data

The Interior Water features comprise seven classes. A hillfort may contain multiple classes. 95.44% of hillforts have no recorded water feature. Only very small numbers of each water feature class have been recorded. It is possible that these figures indicate that water features inside hillforts are a rarity but it is more likely that this data is biased in that there has been a systematic under recording of water features or that water features are, most often, not visible unless reviewed through excavation or remote sensing.

```
In [80]: water_features = [
    'Interior_Water_None',
    'Interior_Water_Spring',
    'Interior_Water_Stream',
    'Interior_Water_Pool',
    'Interior_Water_Flush',
    'Interior_Water_Well',
    'Interior_Water_Other']

water_data = interior_encodeable_data[water_features].copy()
water_data.head(7)
```

	Interior_Water_None	Interior_Water_Spring	Interior_Water_Stream	Interior_Water_Pool	Interior_Water_Flush	Interior_Water_Well	Interior_Water_Other
0	Yes	No	No	No	No	No	No
1	Yes	No	No	No	No	No	No
2	Yes	No	No	No	No	No	No
3	Yes	No	No	No	No	No	No
4	No	Yes	No	No	No	No	No
5	Yes	No	No	No	No	No	No
6	No	No	Yes	Yes	Yes	No	No

There are no null values.

In [81]: `water_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Interior_Water_None    4147 non-null   object 
 1   Interior_Water_Spring   4147 non-null   object 
 2   Interior_Water_Stream   4147 non-null   object 
 3   Interior_Water_Pool    4147 non-null   object 
 4   Interior_Water_Flush   4147 non-null   object 
 5   Interior_Water_Well    4147 non-null   object 
 6   Interior_Water_Other   4147 non-null   object 
dtypes: object(7)
memory usage: 226.9+ KB
```

Water Data Plotted

Most hillforts (94.55%) have no recorded water features.

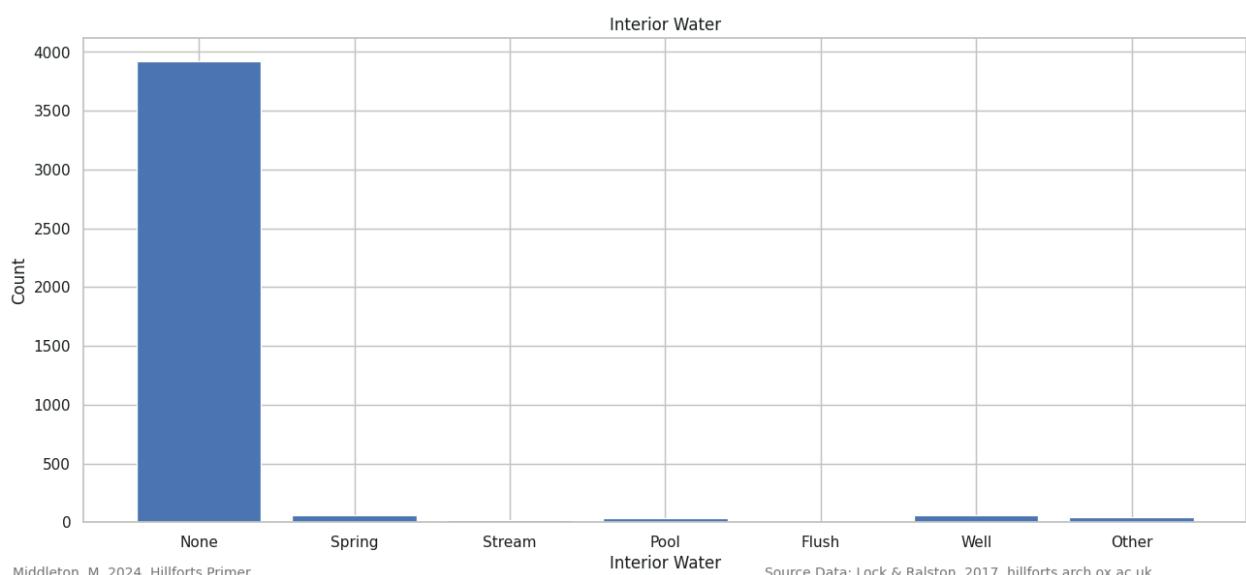
In [82]: `none_water = sum(water_data["Interior_Water_None"] == "Yes")
none_water`

Out[82]: 3921

In [83]: `pcnt_none = round((none_water/4147)*100, 2)
pcnt_none`

Out[83]: 94.55

In [84]: `plot_bar_chart(water_data, 2, 'Interior Water', 'Count', 'Interior Water')`



Water Data Plotted (Excluding None)

The number of hillforts with recorded internal water features is very low. Only 62 are recorded as containing a well, 60 as containing the source of a spring, 38 as having a pool, 22 a stream and just 5 as have a flush.

```
In [85]: water_data_minus = water_data.drop(['Interior_Water_None'], axis=1)
water_data_minus.head()
```

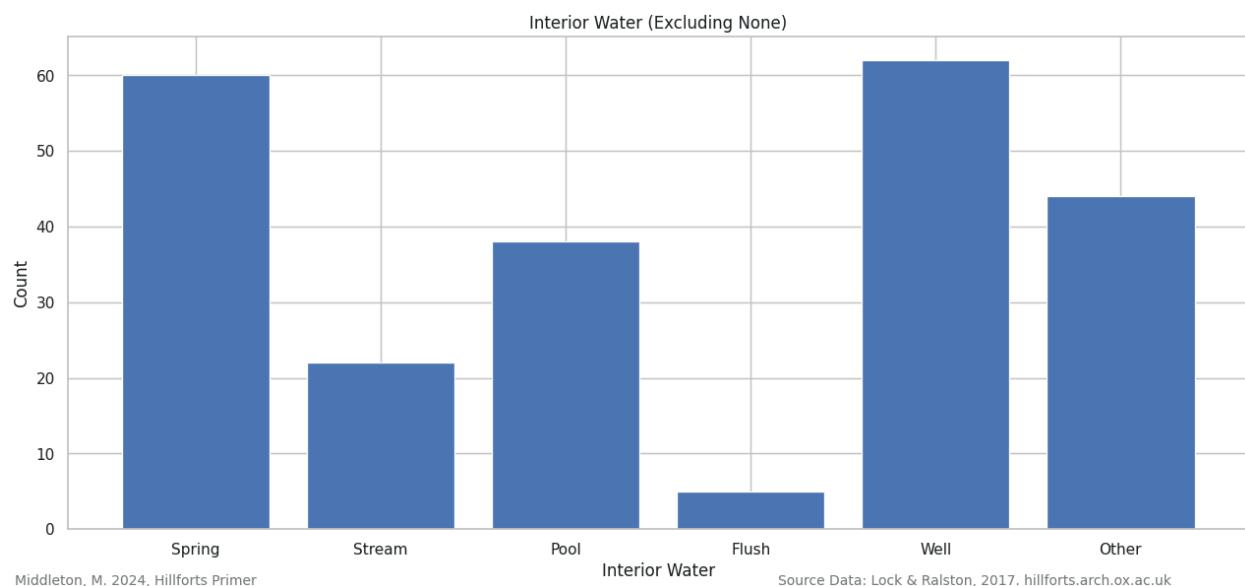
```
Out[85]:
```

	Interior_Water_Spring	Interior_Water_Stream	Interior_Water_Pool	Interior_Water_Flush	Interior_Water_Well	Interior_Water_Other
0	No	No	No	No	No	No
1	No	No	No	No	No	No
2	No	No	No	No	No	No
3	No	No	No	No	No	No
4	Yes	No	No	No	No	No

```
In [86]: for feature in water_features[1:]:
    interior_water_well = sum(water_data_minus[feature] == "Yes")
    print(feature + ": " + str(interior_water_well))
```

```
Interior_Water_Spring: 60
Interior_Water_Stream: 22
Interior_Water_Pool: 38
Interior_Water_Flush: 5
Interior_Water_Well: 62
Interior_Water_Other: 44
```

```
In [87]: plot_bar_chart(water_data_minus, 2, 'Interior Water', 'Count', 'Interior Water (Excluding None)')
```



Water Data Mapped

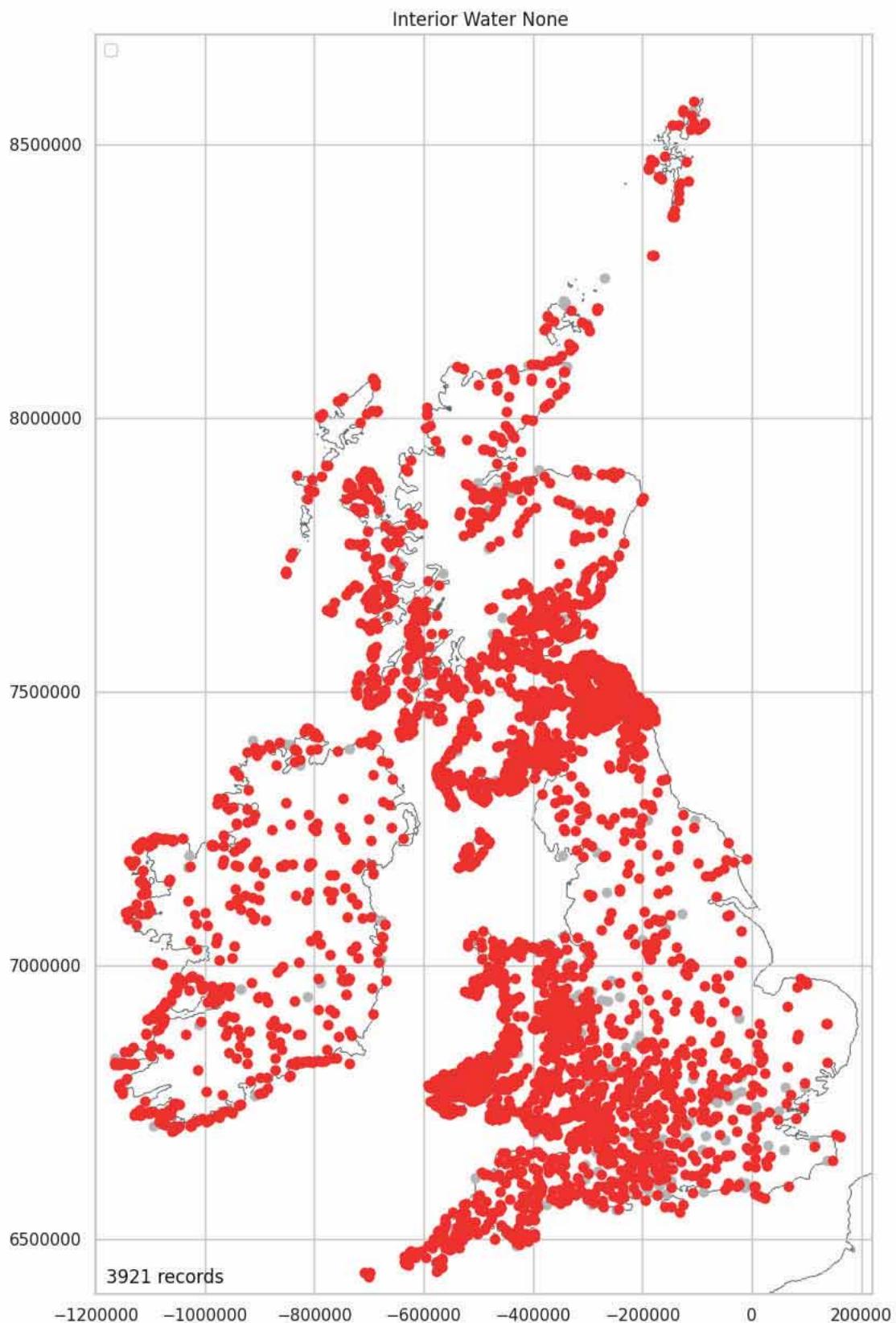
There are very few records relating to water features within hillforts. Most (94.55%) have no water features recorded.

```
In [88]: location_water_data = pd.merge(location_numeric_data_short, water_data, left_index=True, right_index=True)
```

No Water Mapped

Most hillforts have no water features.

```
In [89]: int_no_water = plot_over_grey(location_water_data, 'Interior_Water_None', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

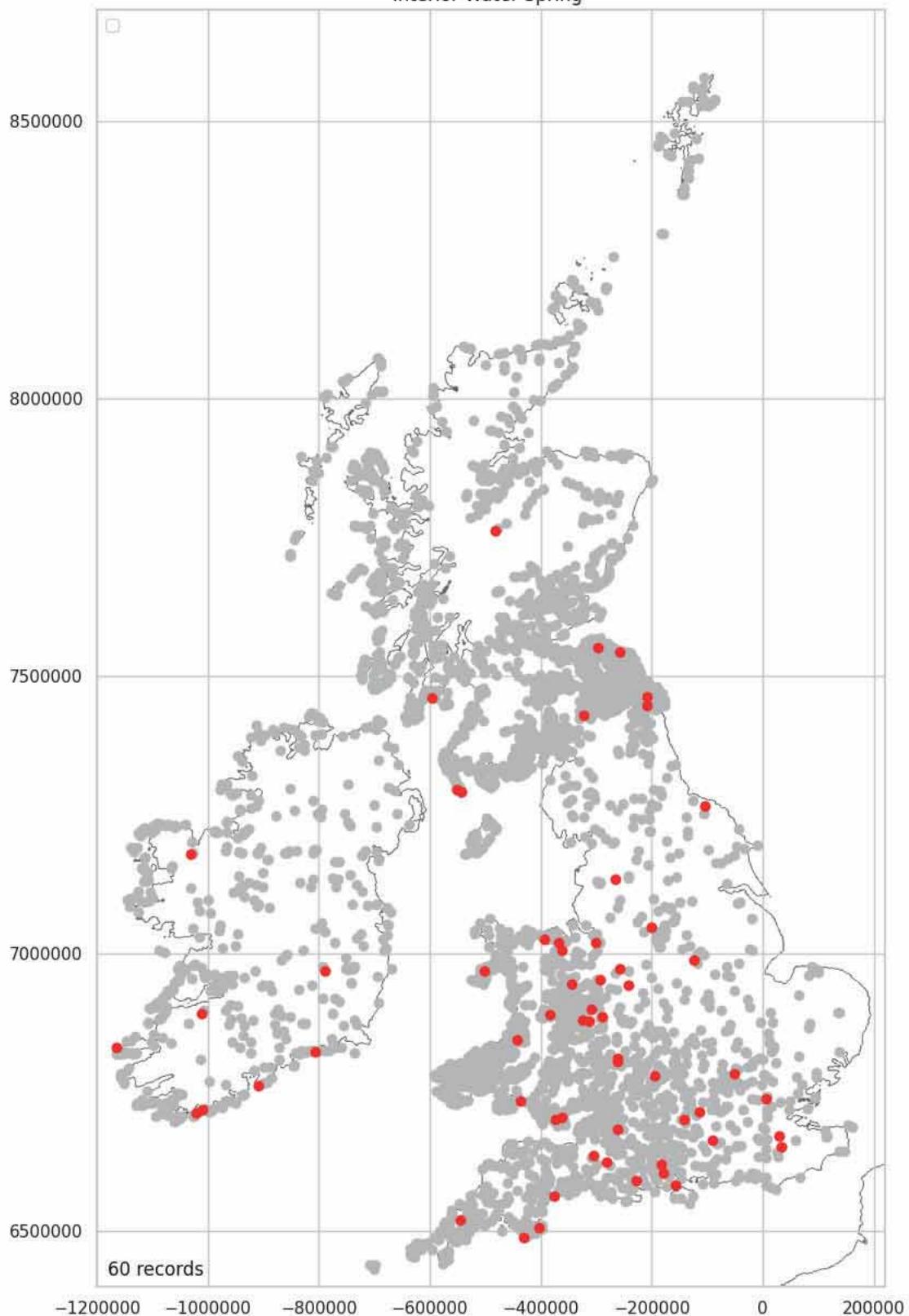
94.55%

Spring Mapped

Only 1.45% have a spring within the hillfort.

```
In [90]: int_spring = plot_over_grey(location_water_data, 'Interior_Water_Spring', 'Yes')
```

Interior Water Spring



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

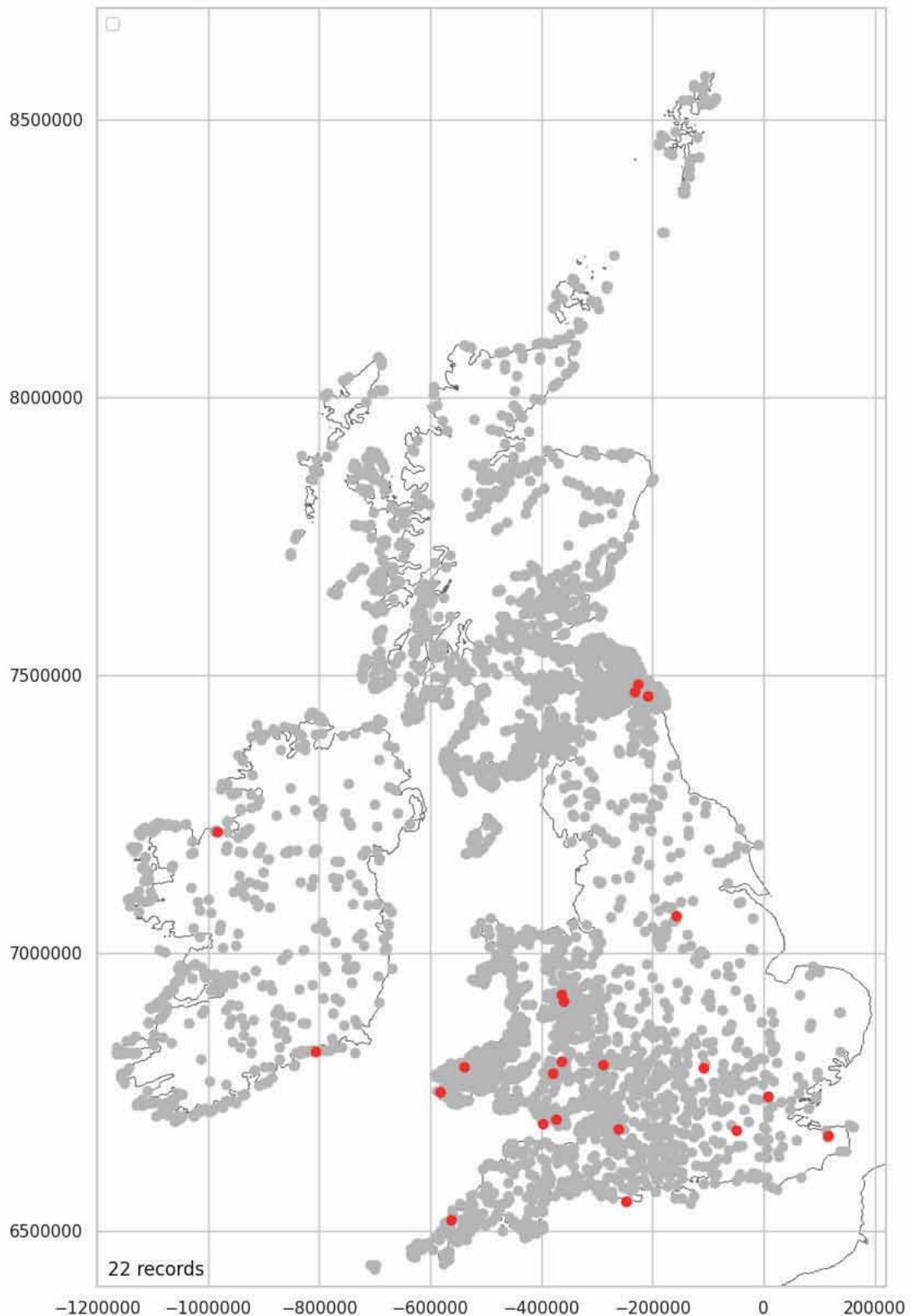
1. 45%

Stream Mapped

Only 0.53% have a stream within the hillfort.

```
In [91]: int_stream = plot_over_grey(location_water_data, 'Interior_Water_Stream', 'Yes')
```

Interior Water Stream



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

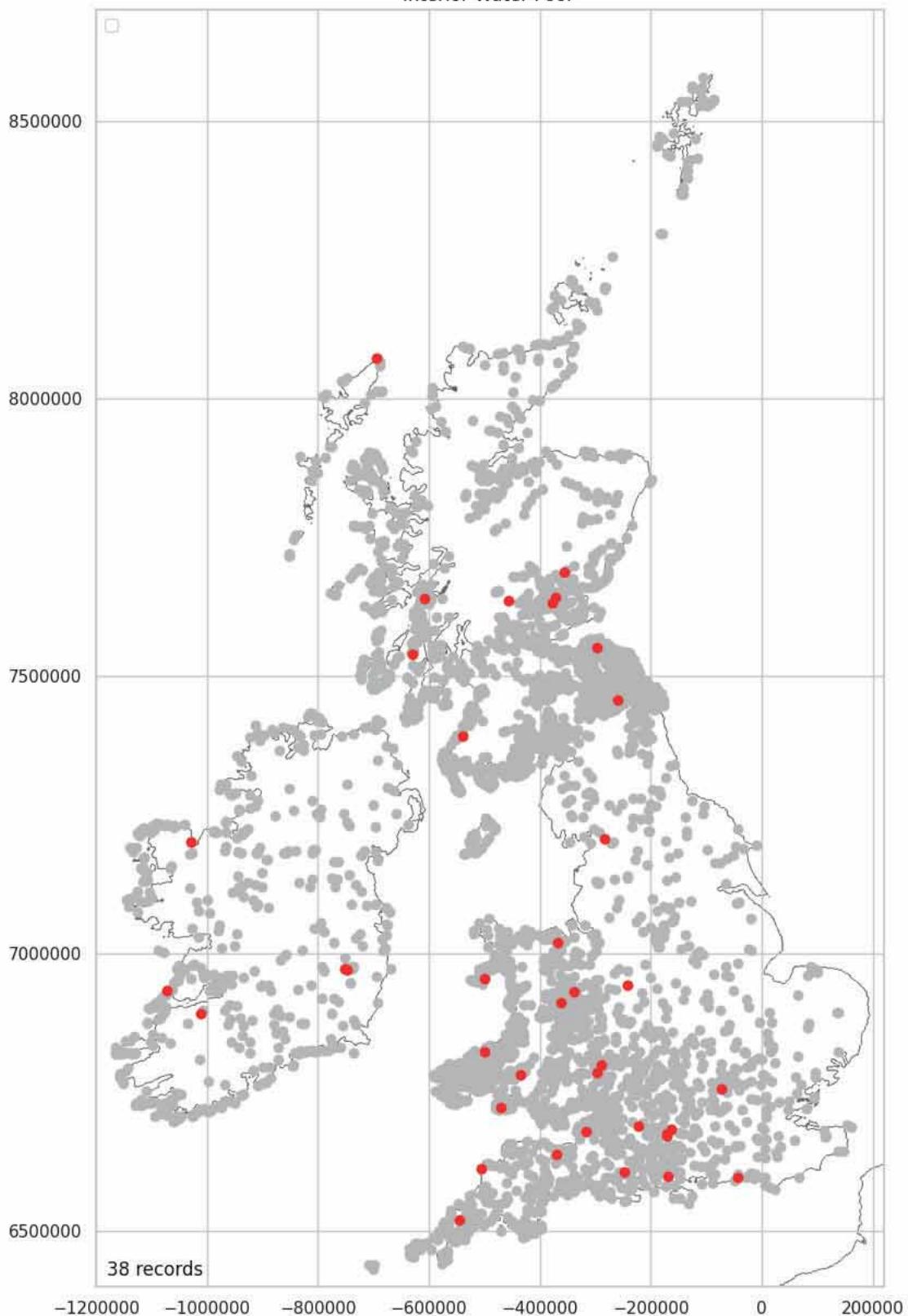
0.53%

Pool Mapped

Just 0.92% have a pool recorded within the hillfort.

```
In [92]: int_pool = plot_over_grey(location_water_data, 'Interior_Water_Pool', 'Yes')
```

Interior Water Pool



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

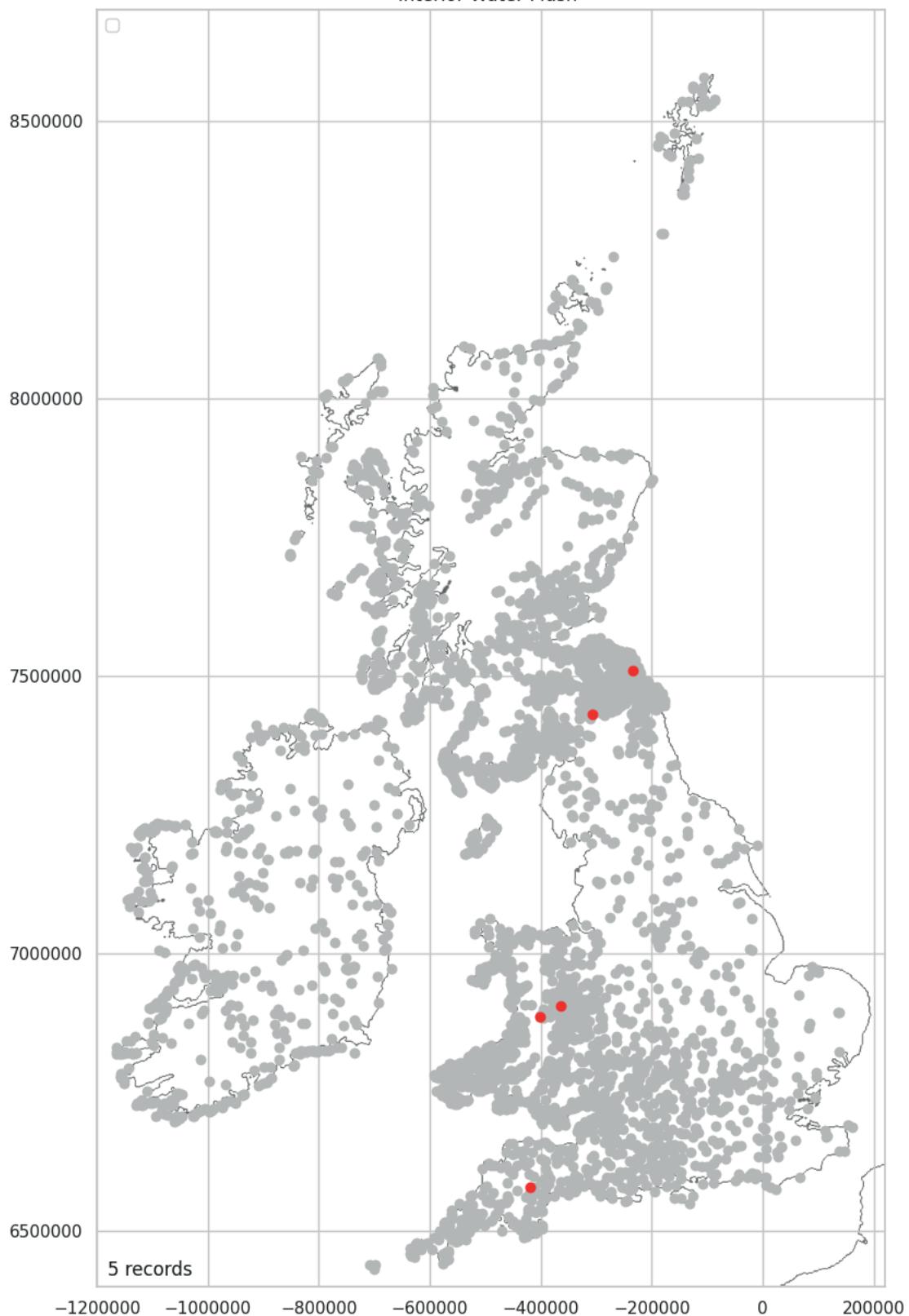
0. 92%

Flush Mapped

There are just five hillforts recorded as having a flush.

```
In [93]: int_flush = plot_over_grey(location_water_data, 'Interior_Water_Flush', 'Yes')
```

Interior Water Flush



Middleton, M. 2024, Hillforts Primer

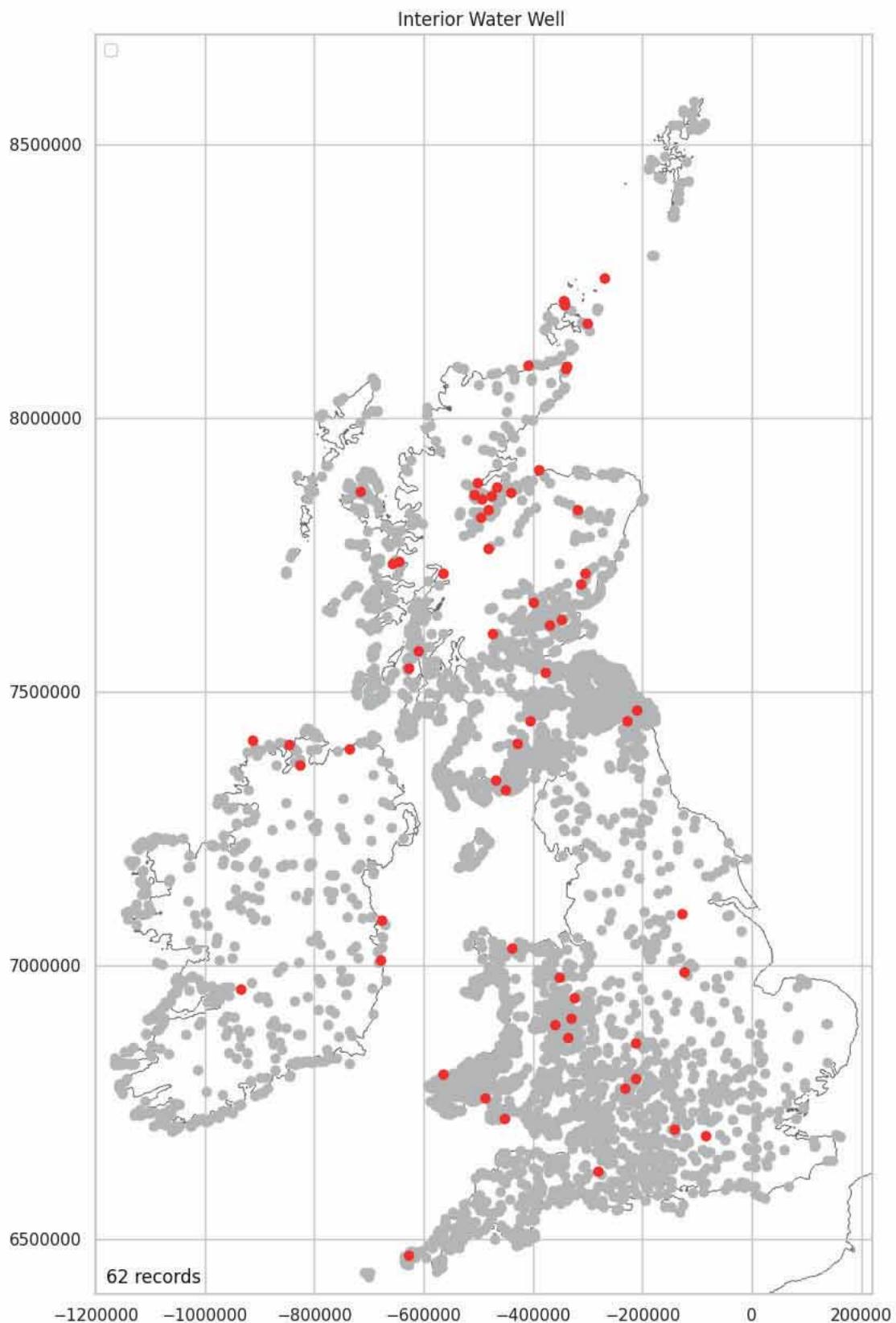
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.12%

Well Mapped

Wells are the most recorded water feature with 1.5% of hillforts recoded as having one.

```
In [94]: int_well = plot_over_grey(location_water_data, 'Interior_Water_Well', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

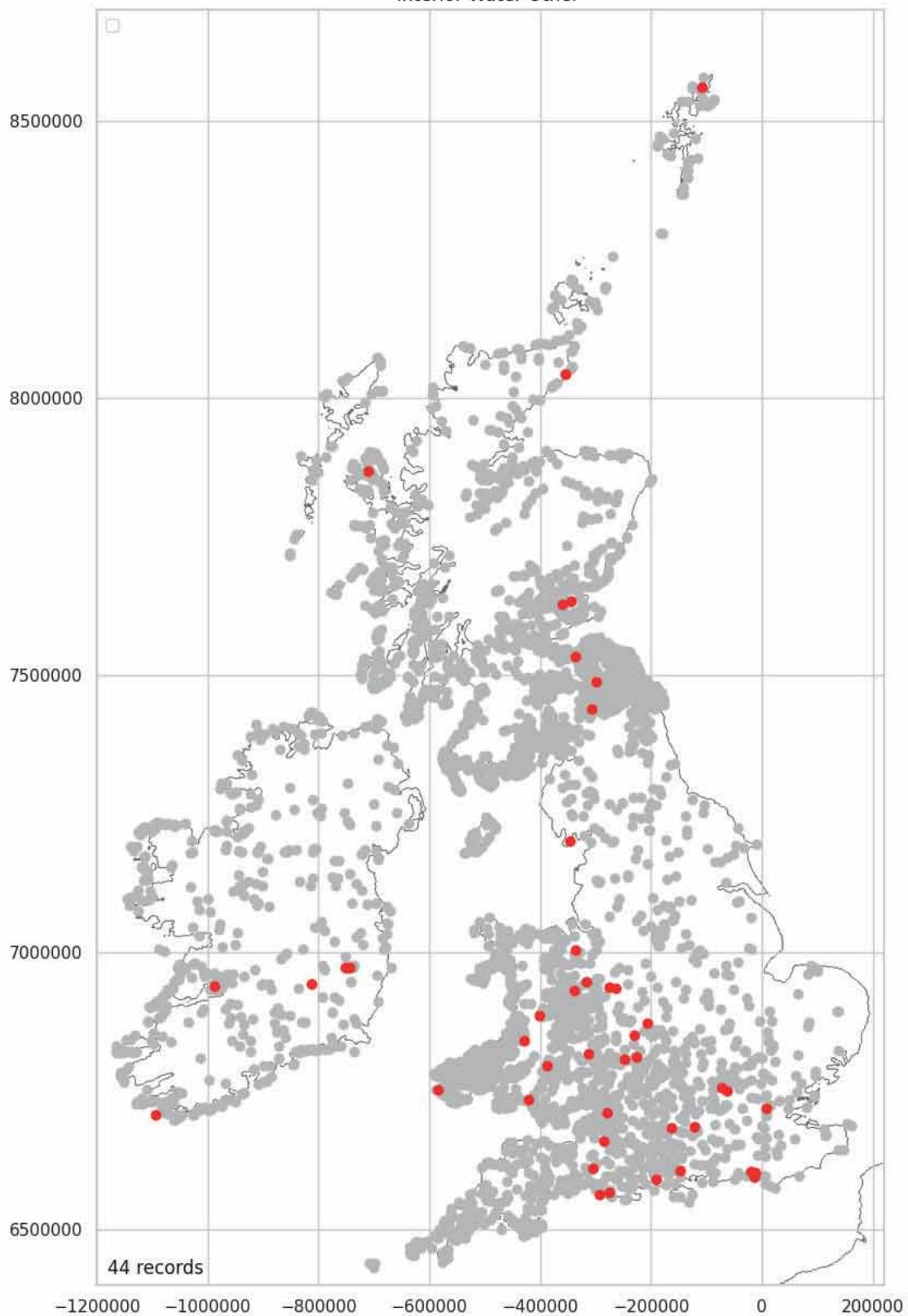
1. 5%

Other Water Mapped

Other water features are recorded at 1.06% of hillforts.

```
In [95]: int_water_other = plot_over_grey(location_water_data, 'Interior_Water_Other', 'Yes')
```

Interior Water Other



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 06%

Surface Data

This section contains eight classes relating to internal features that are visible on the surface. The majority of hillforts (69.57%) have no visible internal features recorded. Where they are, most are found in the two areas of highest hillfort density, the eastern Southern Uplands and the Cambrian Mountains. In addition to these areas, rectangular structures also cluster in the Northwest. Overall, there is a variable survey bias and it is highly probable that there is also a terminology bias with curvilinear being used by

some while others have used round and rectangular. Caution should be used when using this data for interpretation. Any interpretation based on these distributions should qualified.

```
In [96]: surface_features = [
    'Interior_Surface_None',
    'Interior_Surface_Round',
    'Interior_Surface_Rectangular',
    'Interior_Surface_Curvilinear',
    'Interior_Surface_Roundhouse',
    'Interior_Surface_Pit',
    'Interior_Surface_Quarry',
    'Interior_Surface_Other']

surface_data = interior_encodeable_data[surface_features].copy()
surface_data.head()
```

```
Out[96]:
```

	Interior_Surface_None	Interior_Surface_Round	Interior_Surface_Rectangular	Interior_Surface_Curvilinear	Interior_Surface_Roundhouse	In
0	No	No	No	No	No	No
1	Yes	No	No	No	No	No
2	Yes	No	No	No	No	No
3	No	No	No	No	No	Yes
4	No	No	No	No	Yes	No

There are no null values.

```
In [97]: surface_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Interior_Surface_None      4147 non-null   object 
 1   Interior_Surface_Round     4147 non-null   object 
 2   Interior_Surface_Rectangular 4147 non-null   object 
 3   Interior_Surface_Curvilinear 4147 non-null   object 
 4   Interior_Surface_Roundhouse 4147 non-null   object 
 5   Interior_Surface_Pit       4147 non-null   object 
 6   Interior_Surface_Quarry    4147 non-null   object 
 7   Interior_Surface_Other     4147 non-null   object 
dtypes: object(8)
memory usage: 259.3+ KB
```

Surface Data Plotted

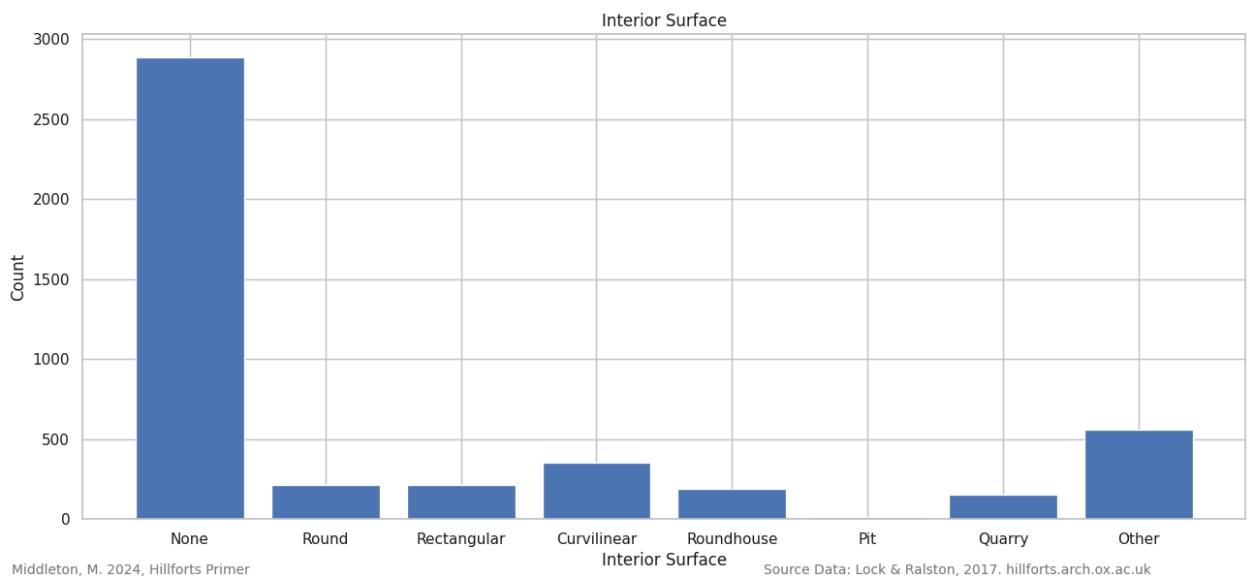
69.59% of Hillforts have no visible internal features recorded.

See: [Geophysics & Excavation Data Plotted \(Excluding None\)](#)

```
In [98]: for feature in surface_features:
    count = sum(interior_encodeable_data[feature] == "Yes")
    print(feature + ": " + str(count))

Interior_Surface_None: 2886
Interior_Surface_Round: 216
Interior_Surface_Rectangular: 211
Interior_Surface_Curvilinear: 350
Interior_Surface_Roundhouse: 192
Interior_Surface_Pit: 15
Interior_Surface_Quarry: 155
Interior_Surface_Other: 557
```

```
In [99]: plot_bar_chart(surface_data, 2, 'Interior Surface', 'Count', 'Interior Surface')
```



Surface Data Plotted (Excluding None)

Where internal features have been recorded, there is a relatively even distribution, across the classes, with 204 (± 12) forts with recorded examples of each, except for pits where there are only 15 and curvilinear features where there are 350.

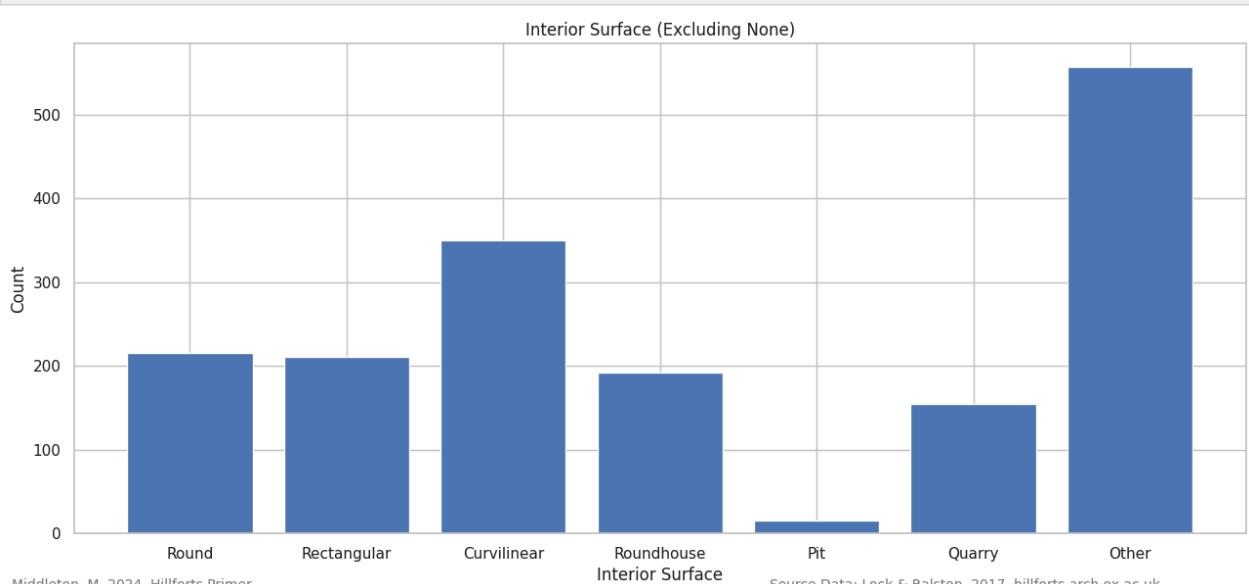
See: [Geophysics & Excavation Data Plotted \(Excluding None\)](#)

```
In [100]: surface_data_minus = surface_data.drop(['Interior_Surface_None'], axis=1)
surface_data_minus.head()
```

```
Out[100]:
```

	Interior_Surface_Round	Interior_Surface_Rectangular	Interior_Surface_Curvilinear	Interior_Surface_Roundhouse	Interior_Surface_Pit	Interior_Surface_Quarry
0	No	No	No	No	No	No
1	No	No	No	No	No	No
2	No	No	No	No	No	No
3	No	No	No	No	Yes	No
4	No	No	No	Yes	No	No

```
In [101]: plot_bar_chart(surface_data_minus, 2, 'Interior Surface', 'Count', 'Interior Surface (Excluding None)')
```



Surface Data Mapped

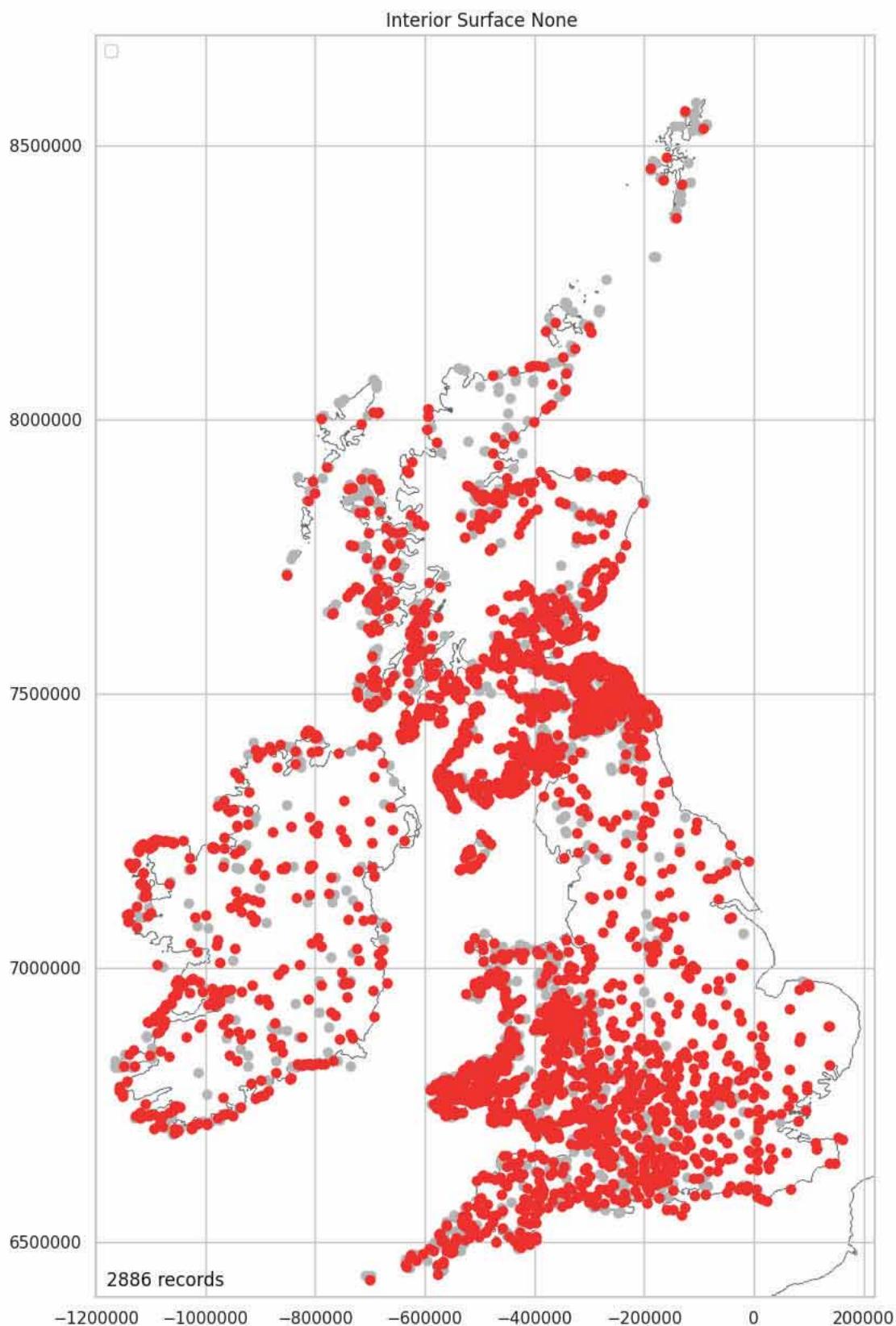
The distribution of recorded surface features is very low and all the following plots are likely to suffer from survey and recording bias.

```
In [102]: location_surface_data = pd.merge(location_numeric_data_short, surface_data, left_index=True, right_index=True)
```

Interior Surface None

Most (69.59%) of Hillforts have no visible internal features recorded.

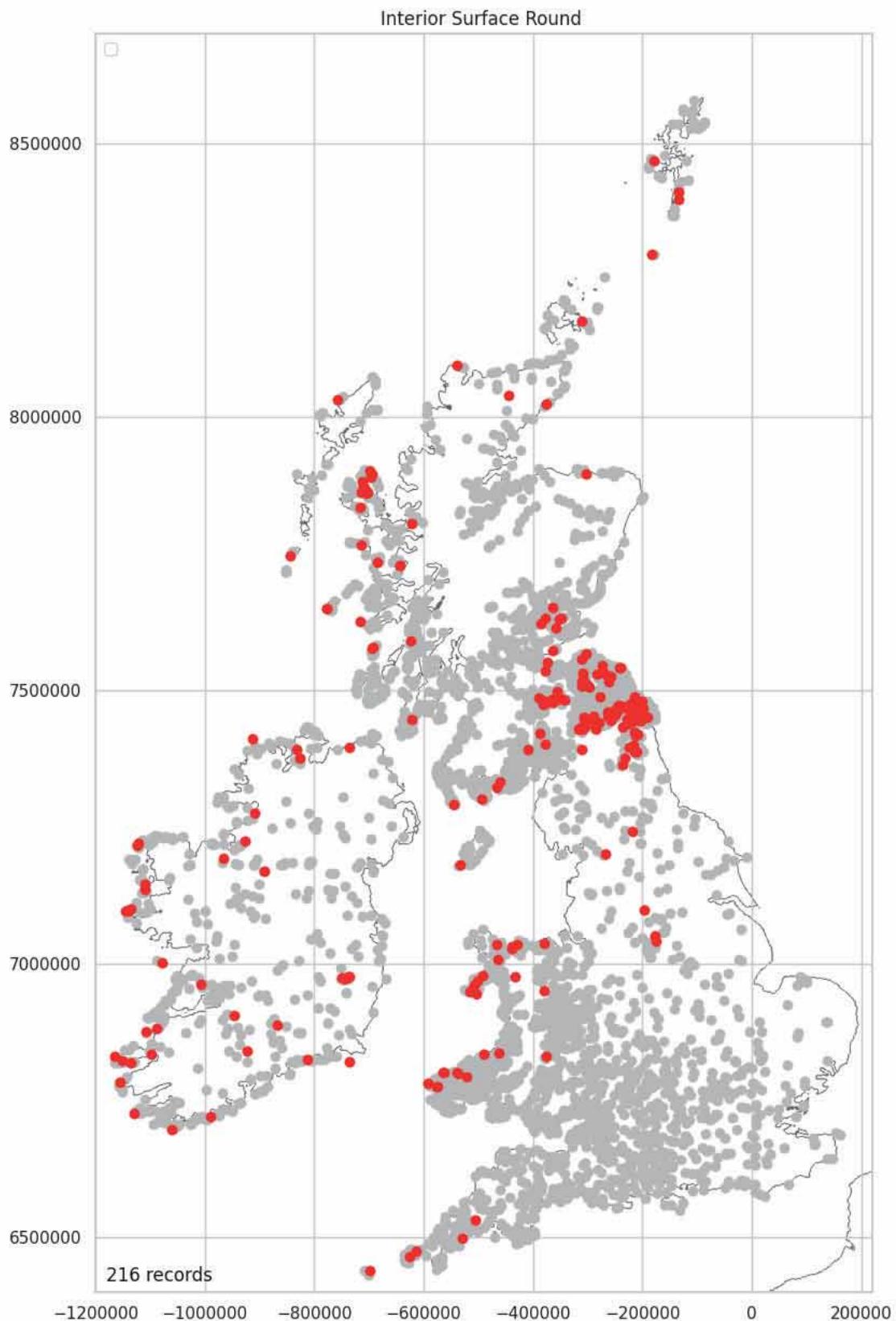
```
In [103]: su_none = plot_over_grey(location_surface_data, 'Interior_Surface_None', 'Yes')
```



5.21% of hillforts are recorded as having circular internal features visible at the surface. There is likely to be survey bias in this data, particularly toward the concentration of data toward the eastern end of the Southern Uplands. It is notable how few circular internal features have been recorded in England.

In [104...]

```
su_round = plot_over_grey(location_surface_data, 'Interior_Surface_Round', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

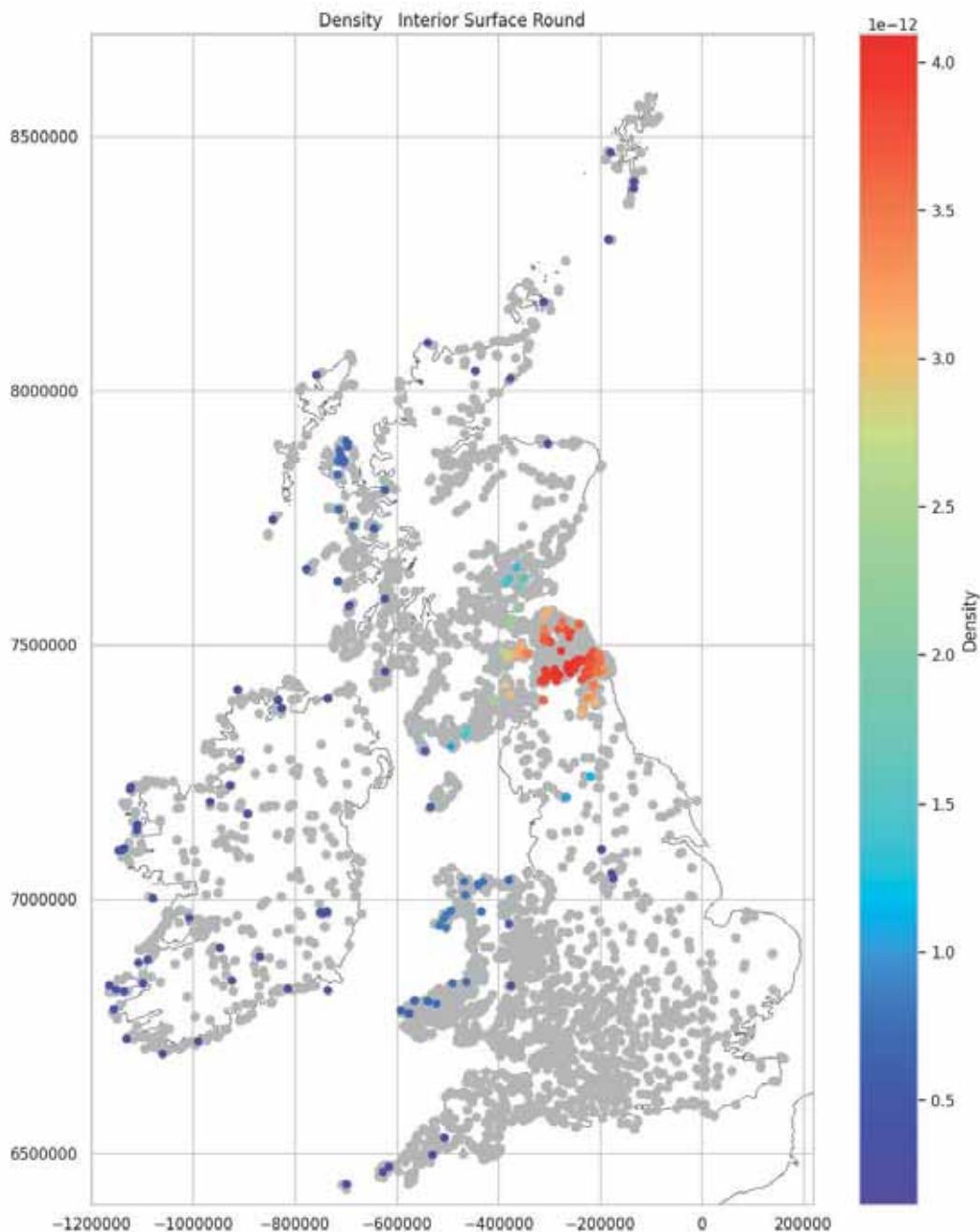
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

5.21%

Round Density Data Mapped

The density plot for round interior surface features most likely highlights a survey bias toward the eastern Southern Uplands rather than a meaningful distribution. This bias is amplified by the increased density of hillforts in this area.

```
In [105]: plot_density_over_grey(su_round, 'Interior_Surface_Round')
```



Middleton, M. 2024, Hillforts Primer

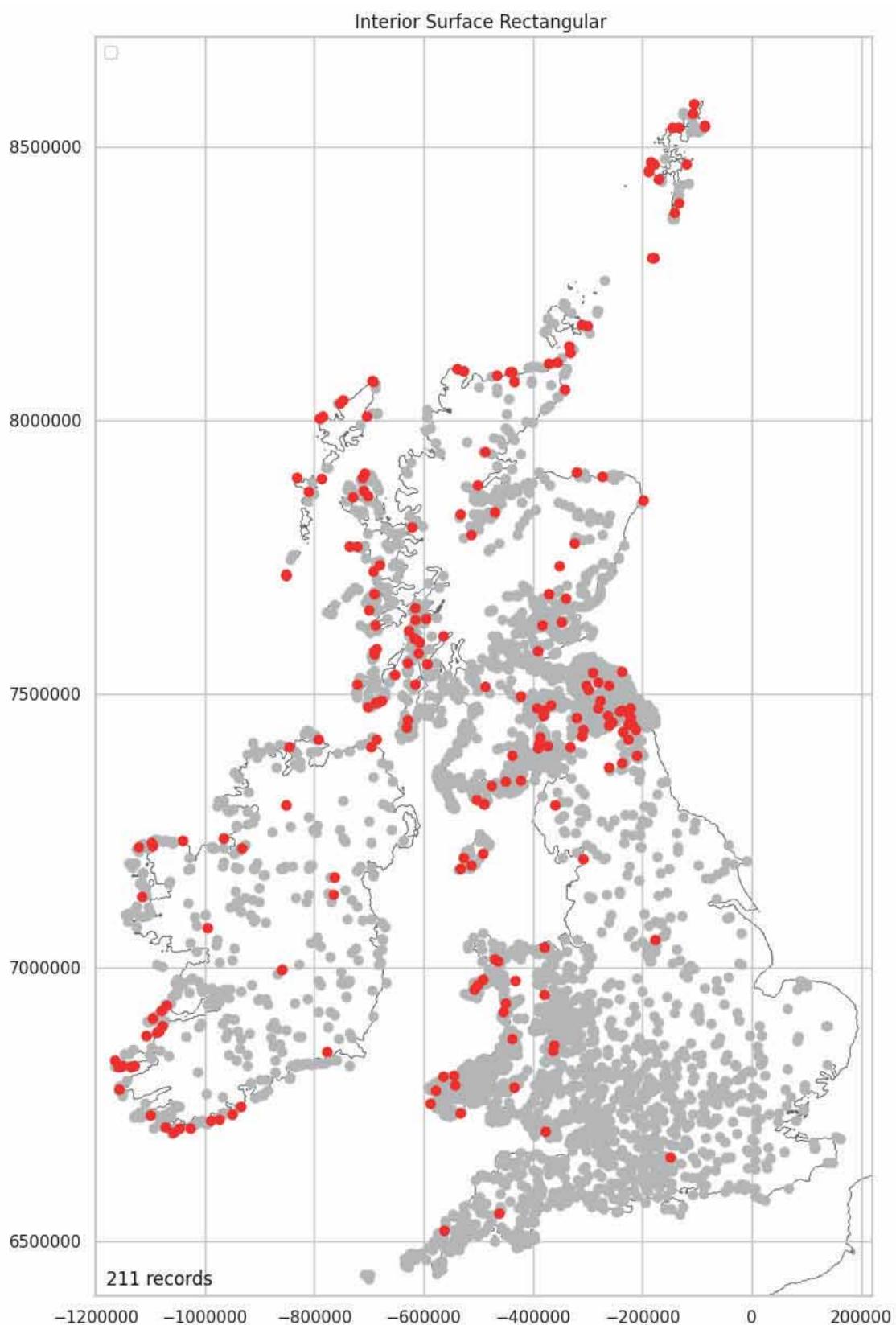
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Rectangular Data Mapped

5.09% of hillforts are recorded as having rectangular internal features. Like the round features above, this data looks to be suffering from a survey bias. The lack of records in England may indicate a lack of recording of these features or perhaps a different land management regime within these forts leading to features not showing at the surface.

There is a noticeable difference in the Northwest between the round and rectangular features. There would seem to be a larger number of rectangular structures recorded but the probable survey bias issues in this data mean caution must be taken in not over interpreting these results.

```
In [106]: su_rect = plot_over_grey(location_surface_data, 'Interior_Surface_Rectangular', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

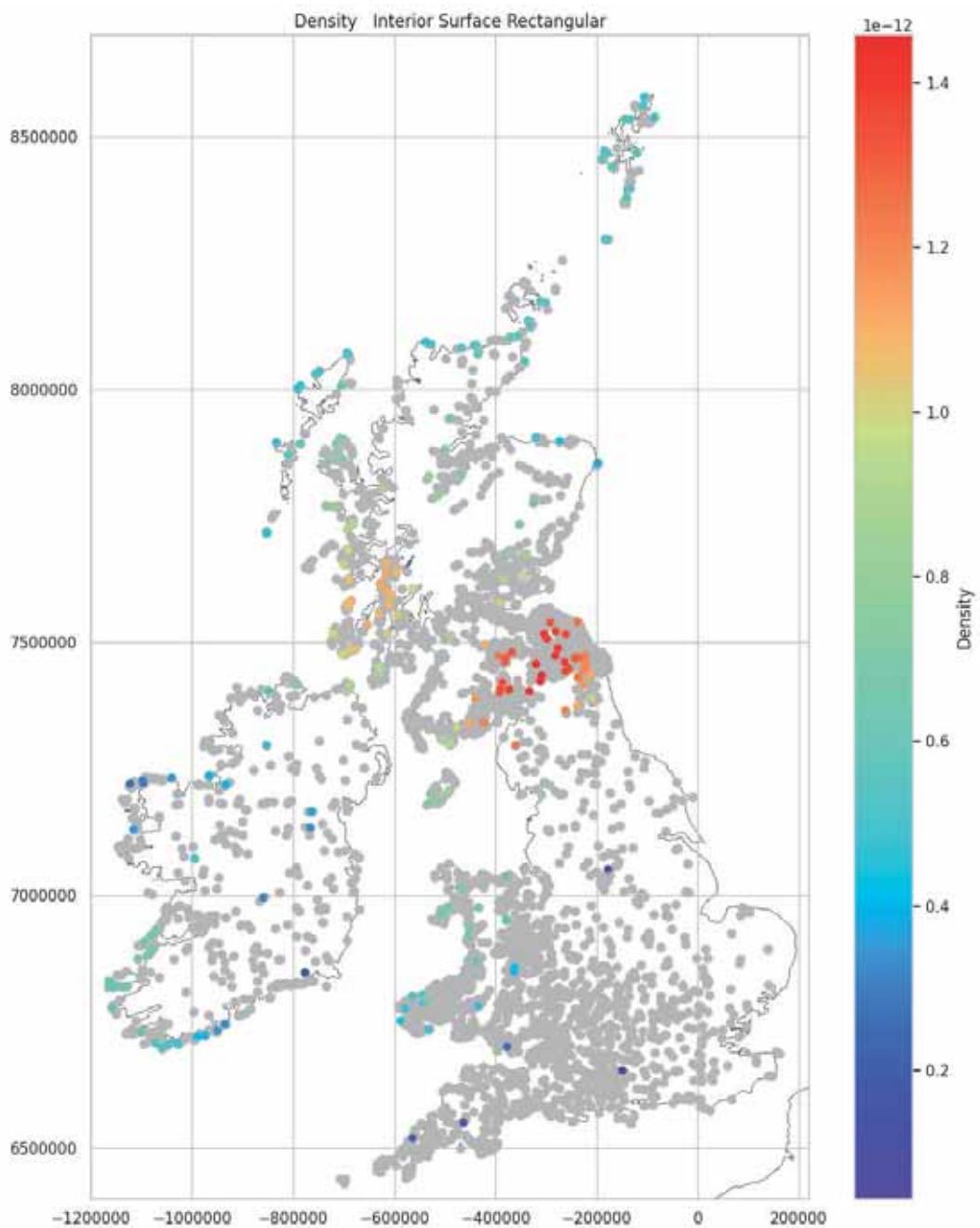
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5.09%

Rectangular Density Data Mapped

The high concentration of hillforts in the southern uplands and the probable survey bias toward this area show as the strongest cluster in this plot. The Northwest, around Dunnad, is notable as a secondary cluster.

```
In [107]: plot_density_over_grey(su_rect, 'Interior_Surface_Rectangular')
```



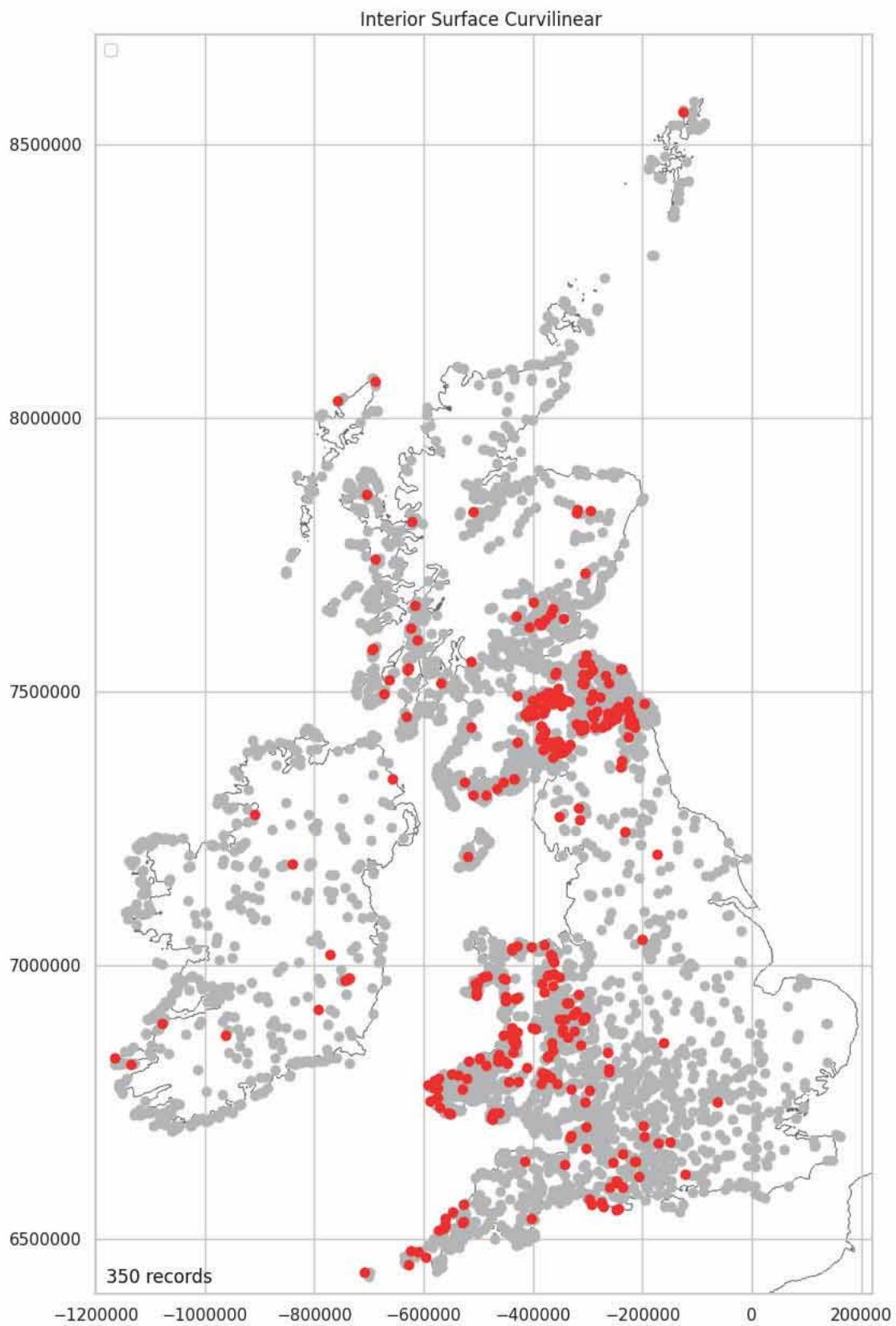
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Curvilinear Data Mapped

8.44% of hillforts are recorded as having curvilinear structures and these are mostly clustered across the two main areas of hillfort distribution - the eastern Southern Uplands and the Cambrian Mountains. Outwith these areas, the distribution of curvilinear structures is very low. The clustering looks to be influenced by survey bias and possible terminology bias - there being a possible preference for using curvilinear over round or rectangular in these areas.

```
In [108]: su_curvi = plot_over_grey(location_surface_data, 'Interior_Surface_Curvilinear', 'Yes')
```



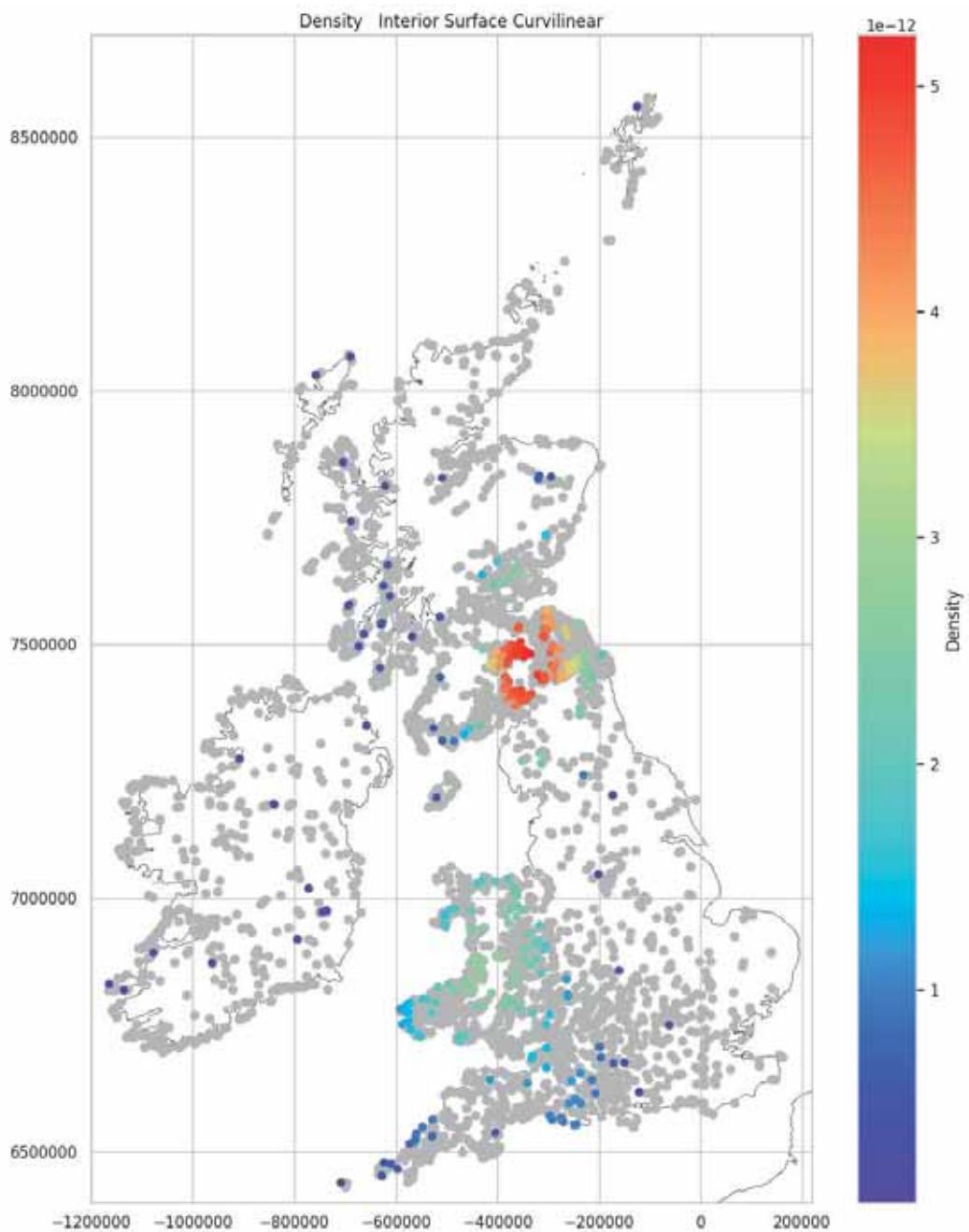
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

8. 44%

Curvilinear Density Data Mapped

There are significant numbers of curvilinear structures recorded on hillforts in the two main areas of hillforts desity - See: Part 1, Density Data Mapped. The cluster over the Southern Uplands is not focussed on the same location as that seen in the Part 1: Northeast Data Mapped. The focus is shifted west and is likely to be a response to a local area survey focus rather than being a meaningful focus of distribution. Outwith these areas there are very few curvilinear sturctures recorded.



Middleton, M. 2024, Hillforts Primer

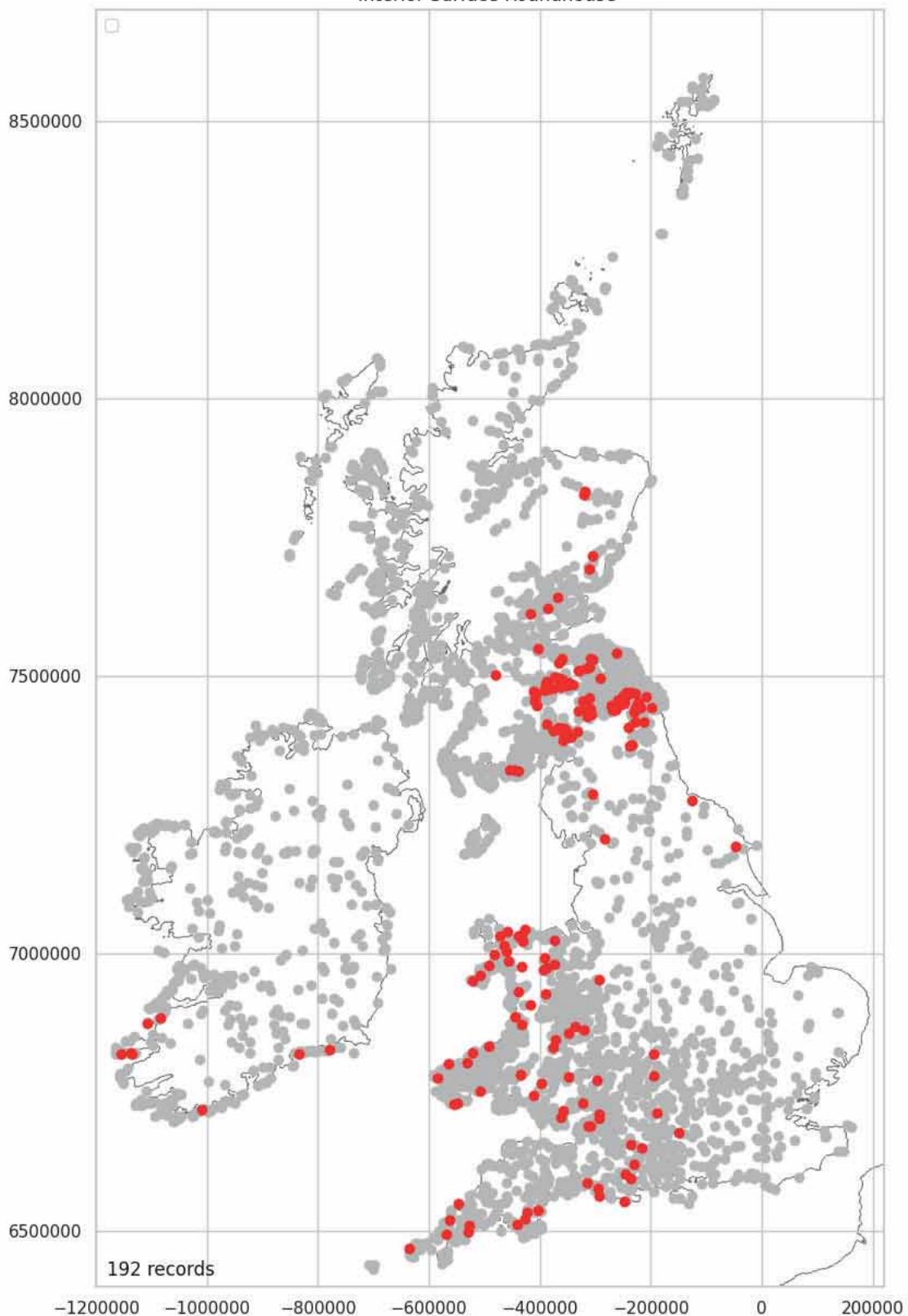
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Roundhouse Data Mapped

4.63% of hillforts have roundhouses recorded in their interior. Like curvilinear structures, the distribution is focussed over the two main areas of hillforts density - the eastern Southern Uplands and the Cambrian Mountains.

```
In [110]: su_roundhouse = plot_over_grey(location_surface_data, 'Interior_Surface_Roundhouse', 'Yes')
```

Interior Surface Roundhouse



Middleton, M. 2024, Hillforts Primer

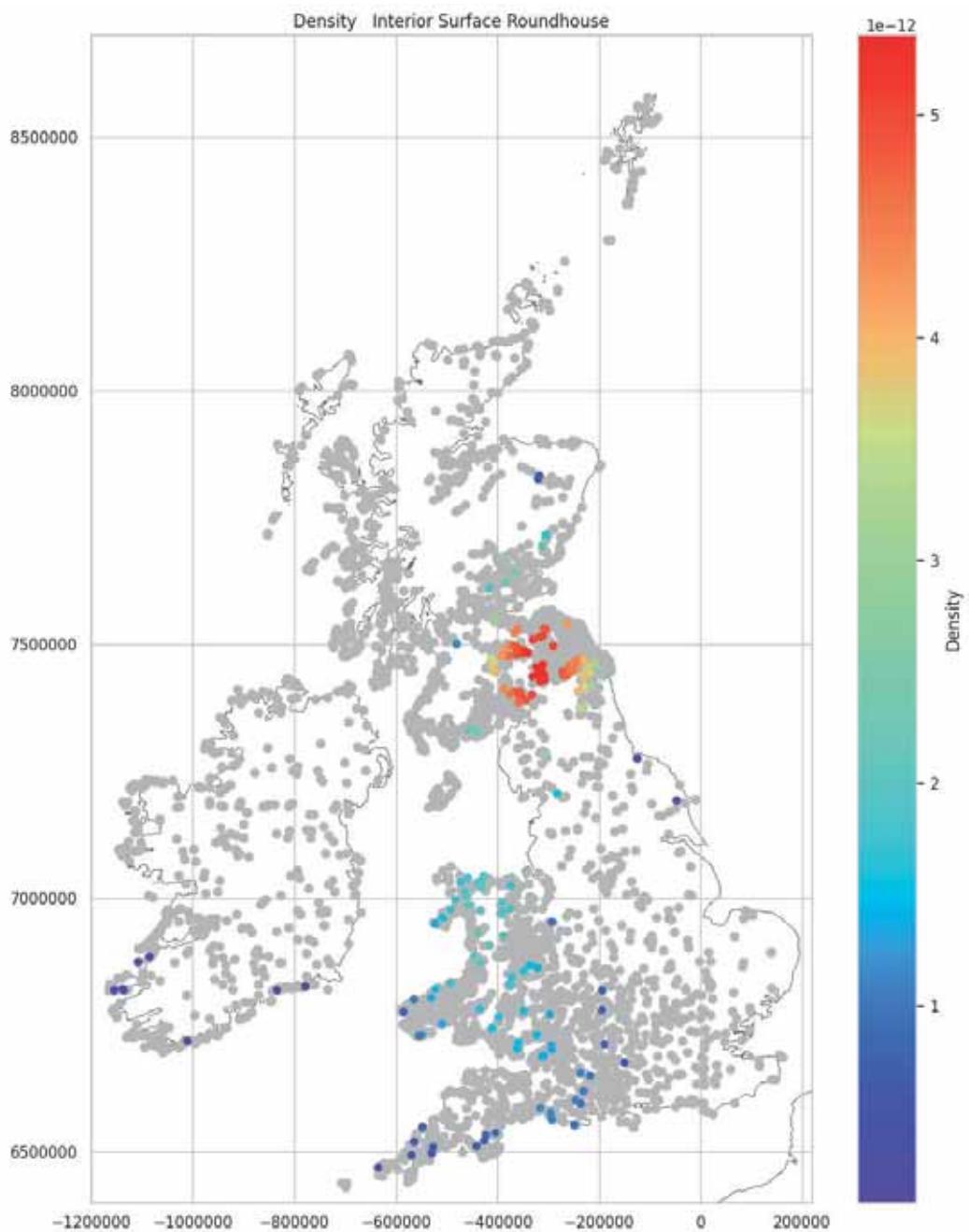
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 63%

Roundhouse Density Data Mapped

The distribution of roundhouses is biased. See discussion in [Curvilinear Density Data Mapped](#).

```
In [111]: plot_density_over_grey(su_roundhouse, 'Interior_Surface_Roundhouse')
```



Middleton, M. 2024, Hillforts Primer

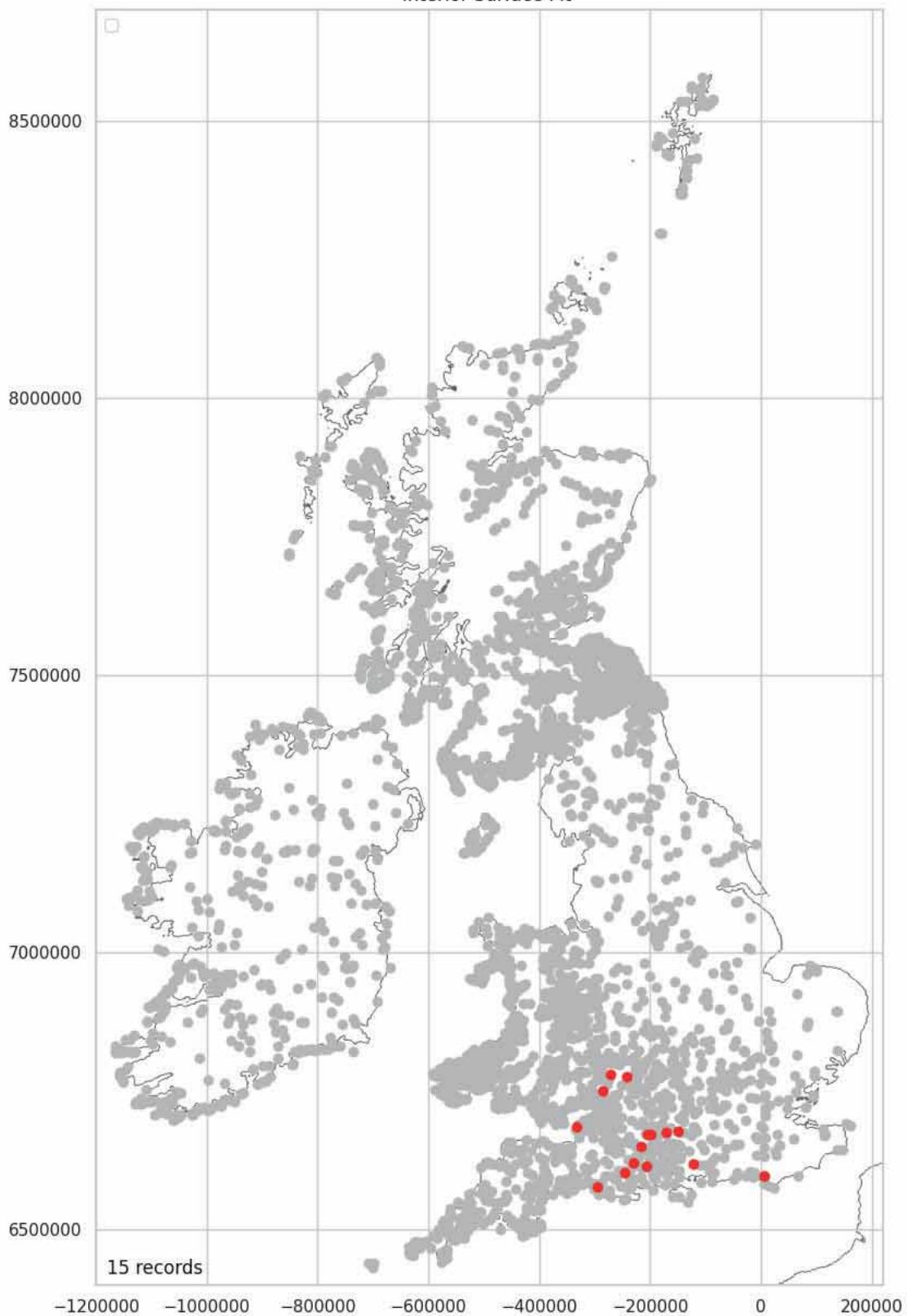
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Pit Data Mapped

Only 15 pits are recorded in hillforts. All are in the south of England. Their distribution is highly likely to be biased and is probably the result of survey focus rather than being a meaningful distribution.

```
In [112]: su_pit = plot_over_grey(location_surface_data, 'Interior_Surface_Pit', 'Yes')
```

Interior Surface Pit



Middleton, M. 2024, Hillforts Primer

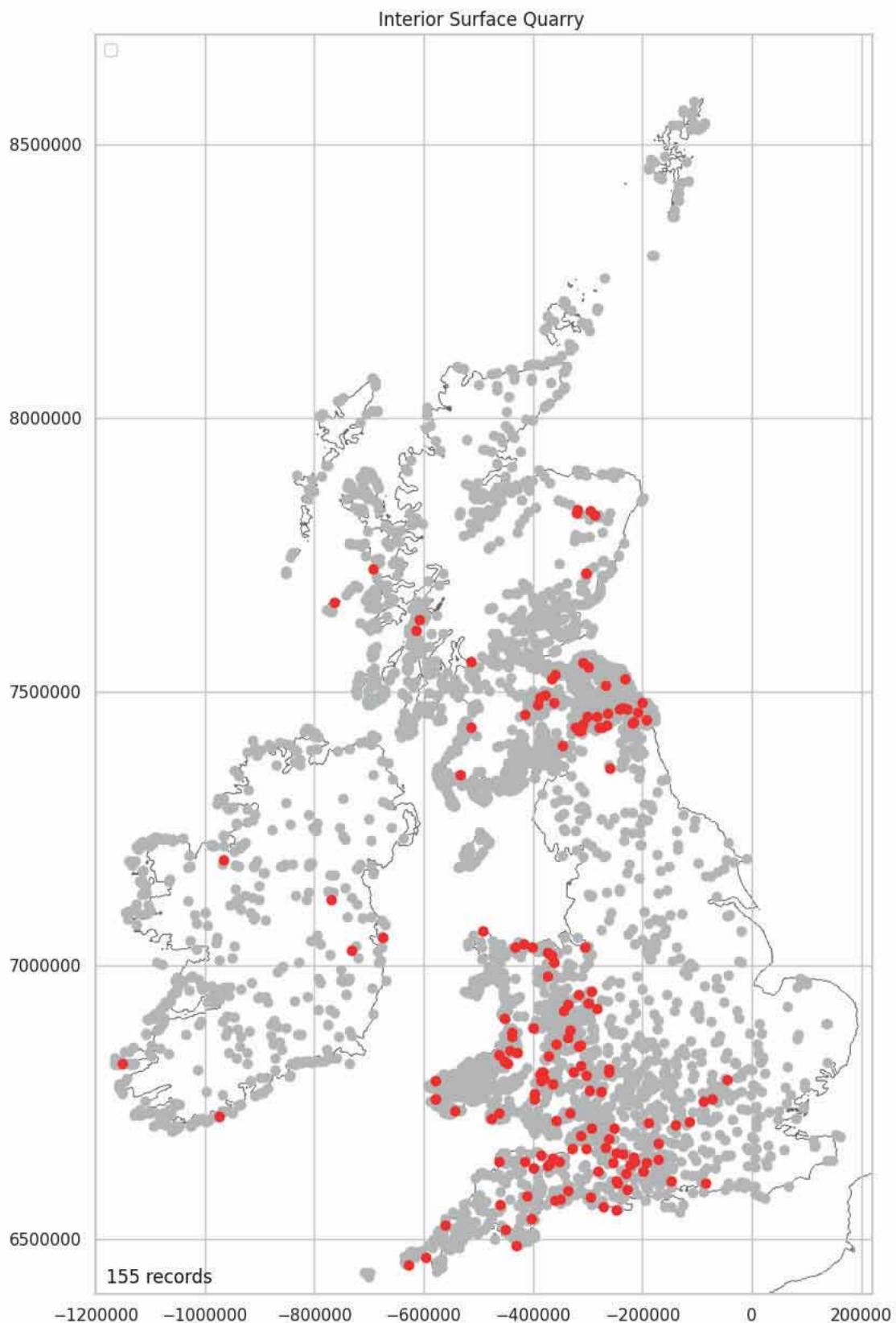
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.36%

Quarry Data Mapped

3.74% of hillforts have a quarry recorded in their interior. Like all the classes in this section, there is a bias in the distribution of these records. Over the Southern Uplands there is a recording bias with more hillforts to the south of the Scottish border having quarries than those in Scotland. There is a much more even distribution across south central England and up along the Welsh border. Generally, there is a survey variability bias across the whole atlas.

```
In [113]: su_quarry = plot_over_grey(location_surface_data, 'Interior_Surface_Quarry', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

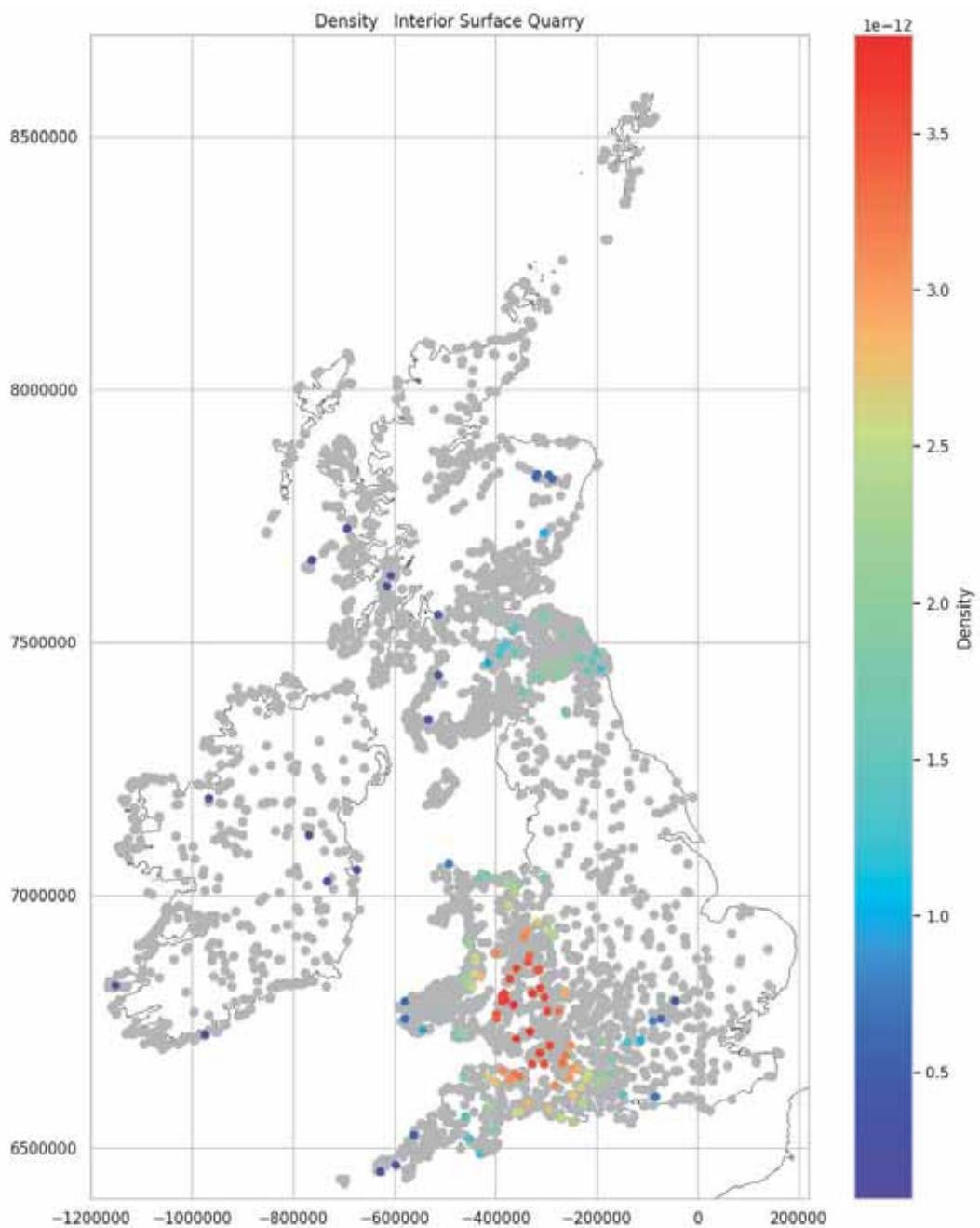
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3.74%

Quarry Density Data Mapped

Where quarries have been recorded the focus is along the Welsh border. This distribution is most likely to be biased by survey area focus and erratic survey outwith these areas.

```
In [114]: plot_density_over_grey(su_quarry, 'Interior_Surface_Quarry')
```



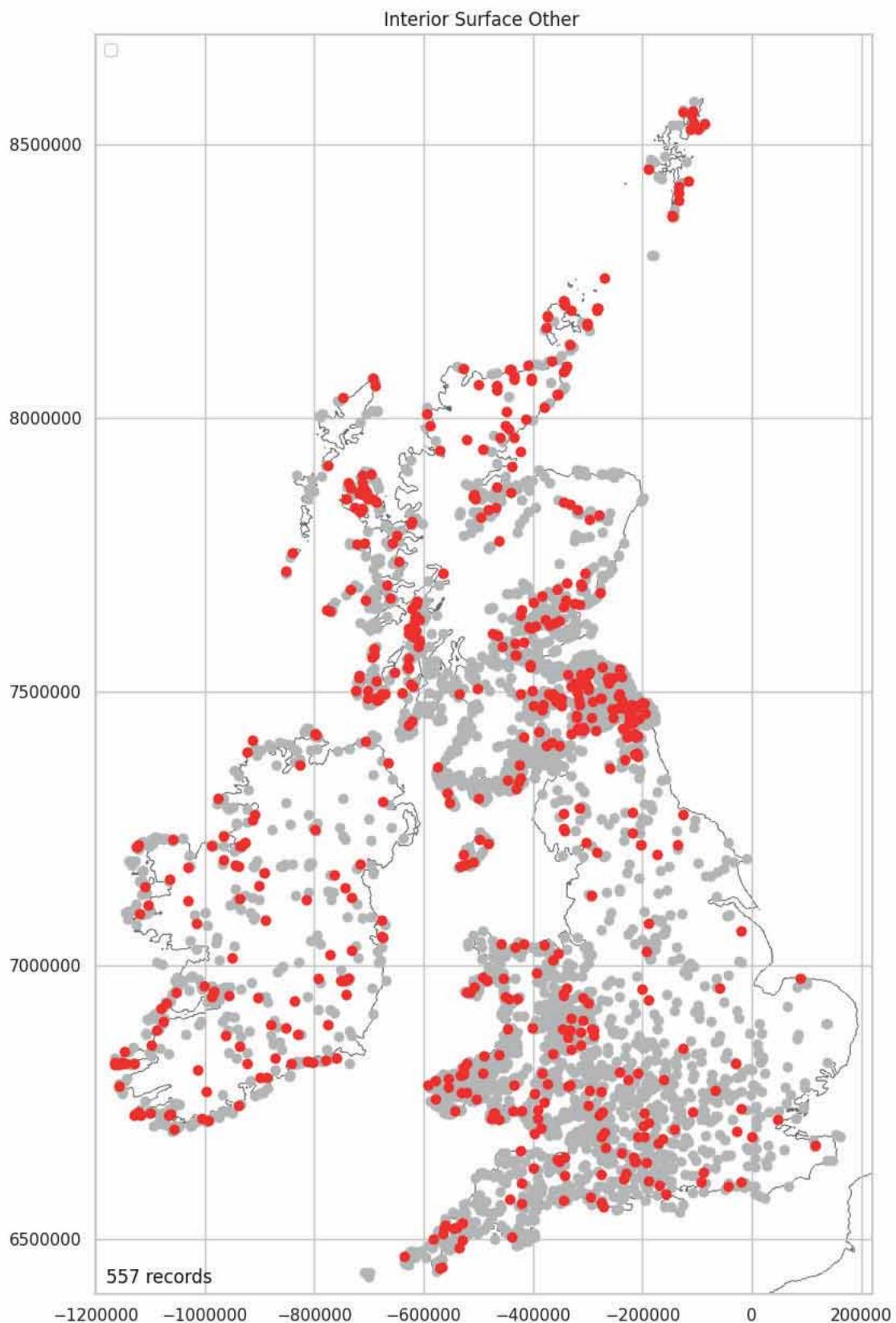
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Other Surface Data Mapped

13.43% of hillforts have 'other' surface features recorded.

```
In [115]: su_other = plot_over_grey(location_surface_data, 'Interior_Surface_Other', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

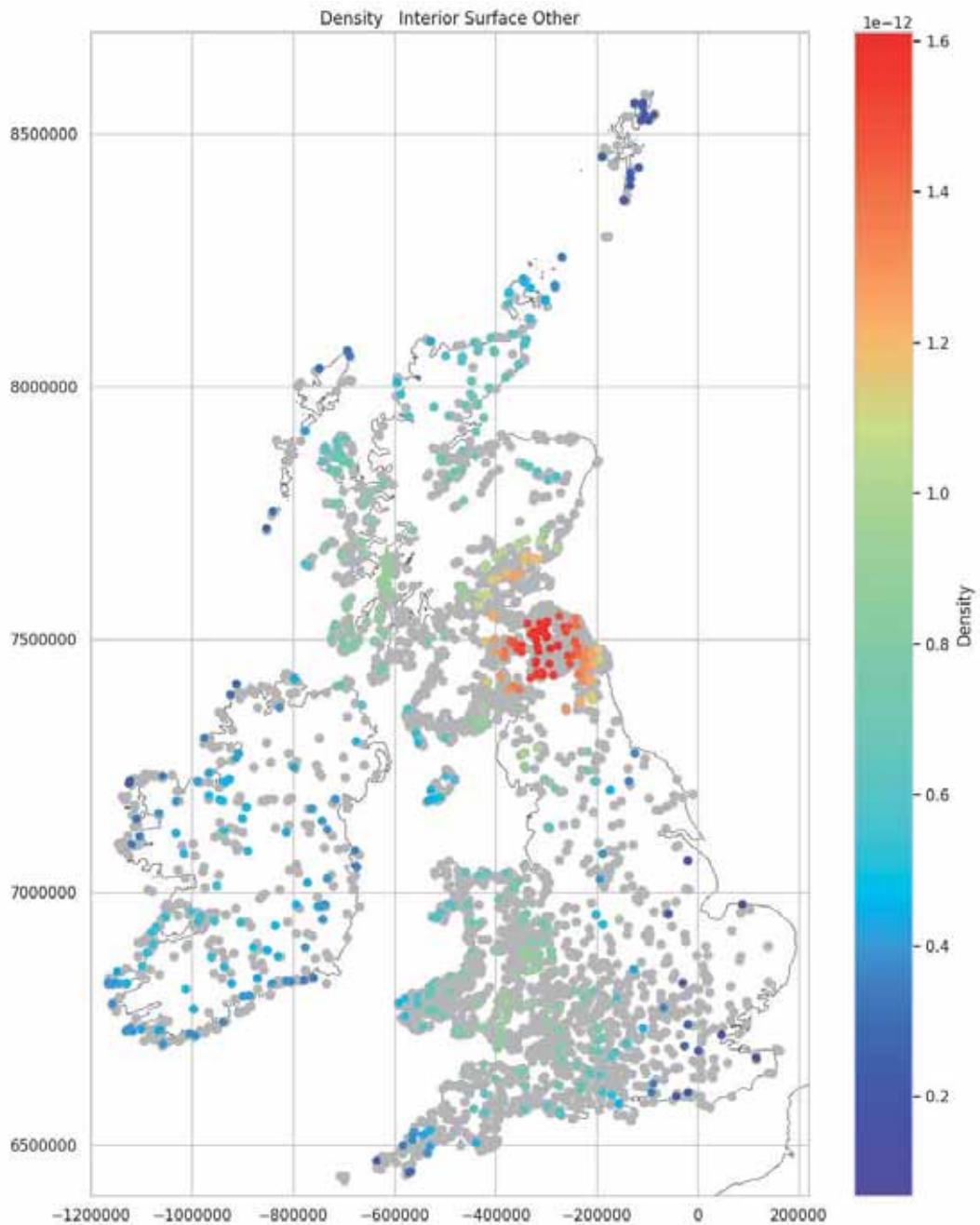
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

13.43%

Other Surface Density Data Mapped

The distribution is in line with the general transformed density plot seen in Part 1. [Density Data Transformed Mapped](#). The Northwest cluster is quite pronounced. The Southern Uplands cluster is as would be expected while the cluster of the Cambrian Mountains is off set to the east.

```
In [116]: plot_density_over_grey(su_other, 'Interior_Surface_Other')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavation Data

The Excavation Data contains nine classes. Most (84.01%) of hillforts have no excavation evidence. Eight of the classes describe the types of structures found within hillforts. The distribution of this data contains a dominant survey bias around south central England. See: [Excavation: None Density Mapped \(Excavated\)](#).

```
In [117...]: excavation_features = [
    'Interior_Excavation_None',
    'Interior_Excavation_Pit',
    'Interior_Excavation_Posthole',
    'Interior_Excavation_Roundhouse',
    'Interior_Excavation_Rectangular',
    'Interior_Excavation_Road',
    'Interior_Excavation_Quarry',
    'Interior_Excavation_Other',
    'Interior_Excavation_Nothing'
]

excavation_data = interior_encodeable_data[excavation_features].copy()
excavation_data.head()
```

	Interior_Excavation_None	Interior_Excavation_Pit	Interior_Excavation_Posthole	Interior_Excavation_Roundhouse	Interior_Excavation_Rectangular
0	Yes	No	No	No	No
1	Yes	No	No	No	No
2	Yes	No	No	No	No
3	No	Yes	No	Yes	Yes
4	No	No	No	No	No

There are no null values.

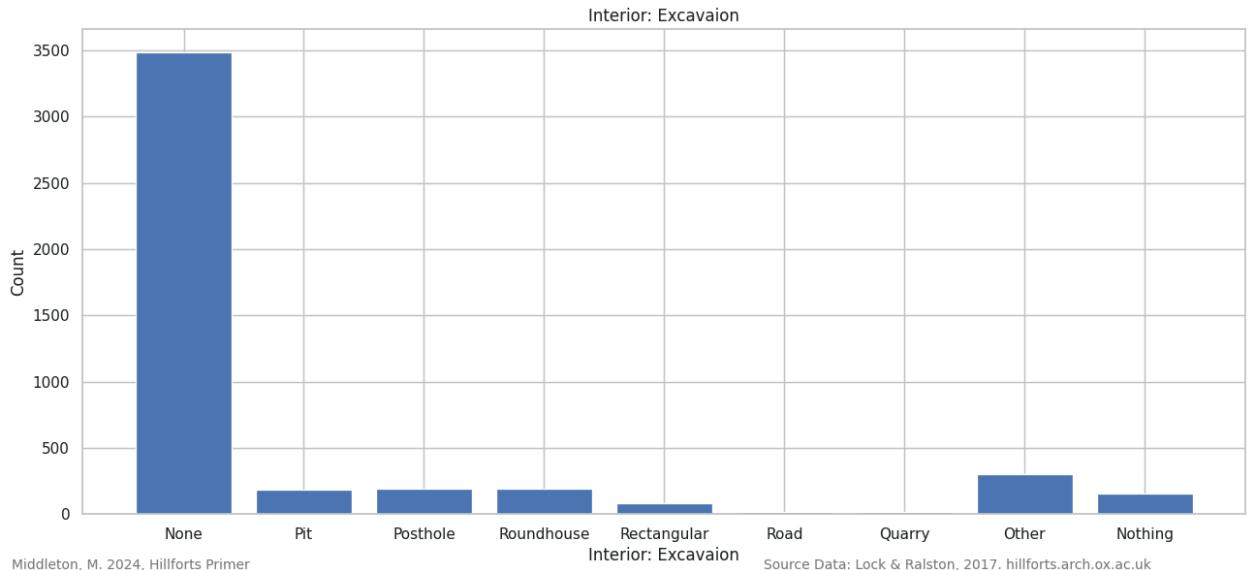
In [118]: `excavation_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Interior_Excavation_None    4147 non-null   object 
 1   Interior_Excavation_Pit     4147 non-null   object 
 2   Interior_Excavation_Posthole 4147 non-null   object 
 3   Interior_Excavation_Roundhouse 4147 non-null   object 
 4   Interior_Excavation_Rectangular 4147 non-null   object 
 5   Interior_Excavation_Road     4147 non-null   object 
 6   Interior_Excavation_Quarry   4147 non-null   object 
 7   Interior_Excavation_Other    4147 non-null   object 
 8   Interior_Excavation_Nothing  4147 non-null   object 
dtypes: object(9)
memory usage: 291.7+ KB
```

Excavation Data Plotted

None (no excavation data) dominates the plot and is excluded, to facilitate interpretation of the remaining classes, in the following plot.

In [119]: `plot_bar_chart(excavation_data, 2, 'Interior_Excavation', 'Count', 'Interior_Excavation')`



Excavation Data Plotted (Excluding None)

663 hillforts have been excavated. Of these, 153 (23.08%) have no recorded internal structures. Where there are structures, pits, postholes and roundhouses are evenly represented in around 188 (± 5) forts. Rectangular structures are present at at only 85 hillforts. Roads and quarries have been recorded at 19 sites. Just under half the excavated forts (45.55%) have other internal features.

In [120]: `excavated_forts = 4147 - sum(excavation_data['Interior_Excavation_None'] == "Yes")
excavated_forts`

Out[120]: 663

In [121]: `excavation_nothing = sum(excavation_data['Interior_Excavation_Nothing'] == "Yes")
excavation_nothing`

```
Out[121]: 153
```

```
In [122... excavation_nothing_pcnt = round((excavation_nothing / excavated_forts) * 100, 2)
excavation_nothing_pcnt
```

```
Out[122]: 23.08
```

```
In [123... for feature in excavation_features[1:-1]:
    print(feature + ": " + str(sum(excavation_data[feature]=="Yes")))
```

```
Interior_Excavation_Pit: 184
Interior_Excavation_Posthole: 189
Interior_Excavation_Roundhouse: 193
Interior_Excavation_Rectangular: 84
Interior_Excavation_Road: 19
Interior_Excavation_Quarry: 19
Interior_Excavation_Other: 302
```

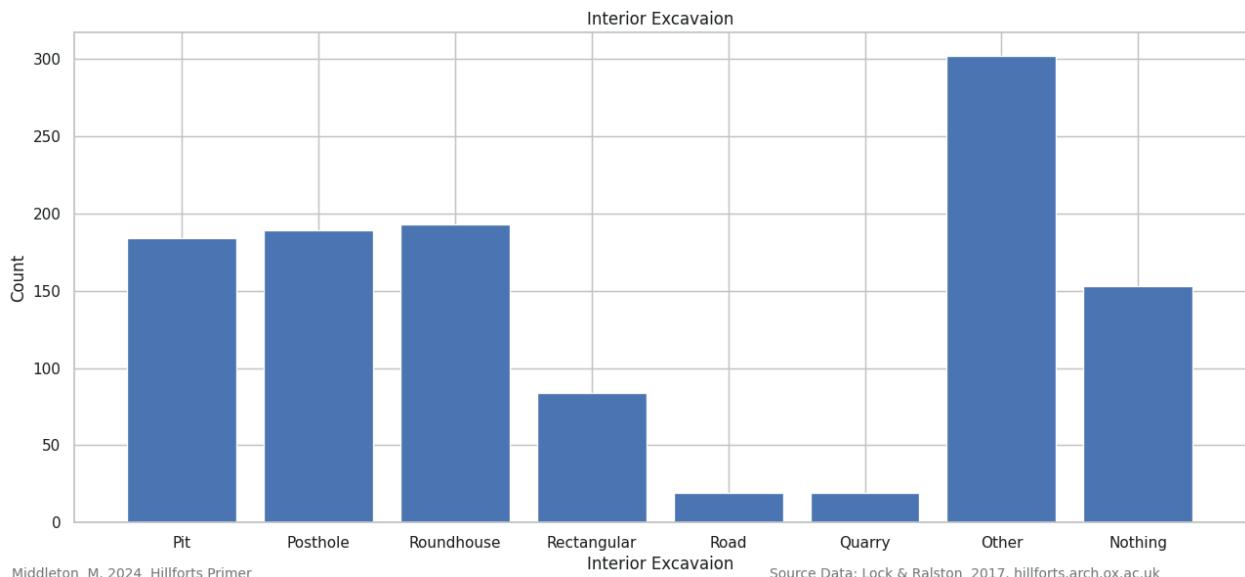
```
In [124... excavation_other_pcnt = round((sum(excavation_data['Interior_Excavation_Other']=="Yes") / excavated_forts) * 100, 2)
excavation_other_pcnt
```

```
Out[124]: 45.55
```

```
In [125... excavation_data_minus = excavation_data.drop(['Interior_Excavation_None'], axis=1)
excavation_data_minus.head()
```

```
Out[125]:   Interior_Excavation_Pit  Interior_Excavation_Posthole  Interior_Excavation_Roundhouse  Interior_Excavation_Rectangular  Interior_Excavation_Other
0            No                  No                  No                  No                  No
1            No                  No                  No                  No                  No
2            No                  No                  No                  No                  No
3           Yes                 Yes                 Yes                 Yes                 Yes
4            No                  No                  No                  No                  No
```

```
In [126... plot_bar_chart(excavation_data_minus, 2, 'Interior Excavation', 'Count', 'Interior Excavation')
```



Excavation Data Mapped

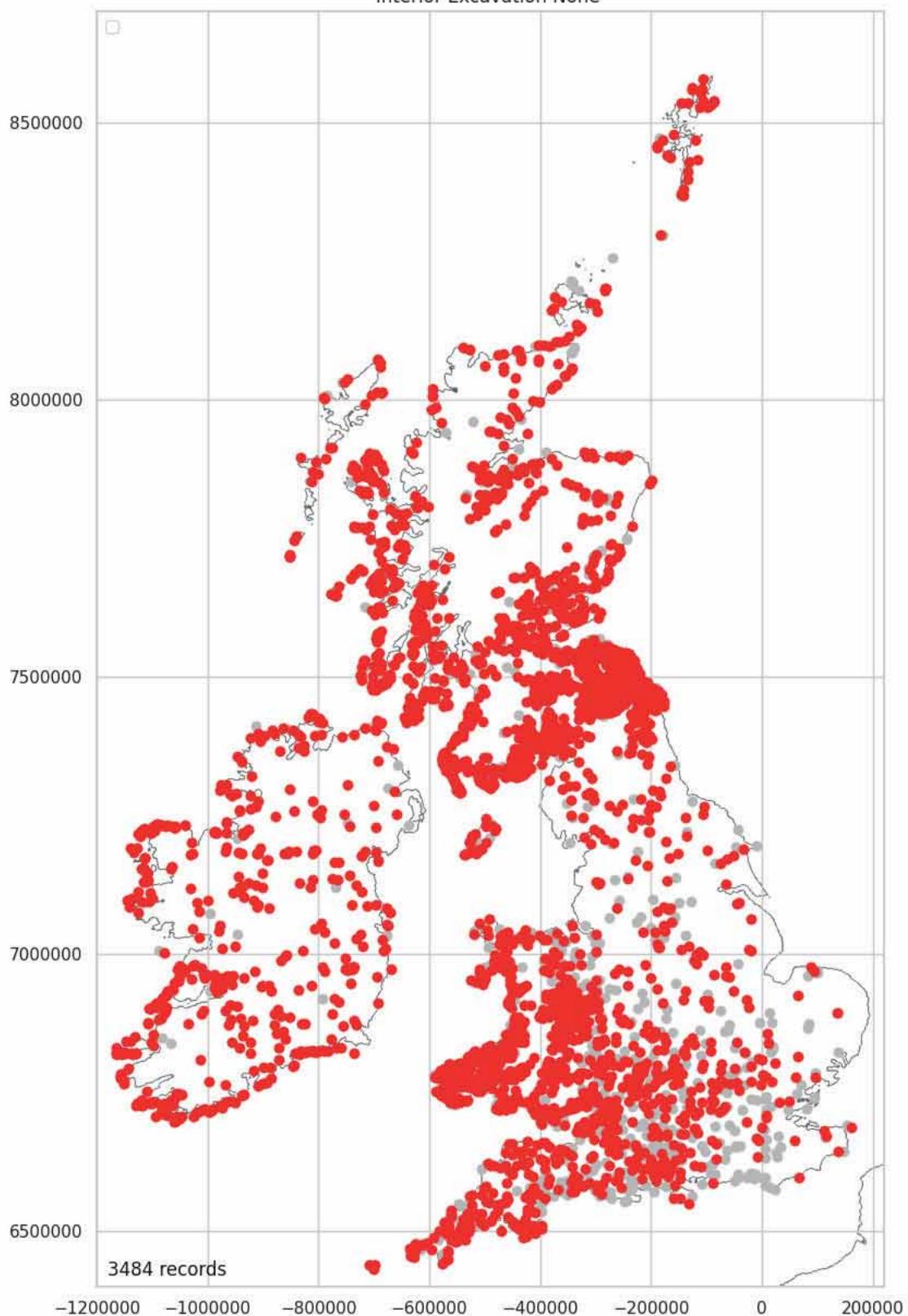
```
In [127... location_excavation_data = pd.merge(location_numeric_data_short, excavation_data, left_index=True, right_index=True)
```

Excavation: None Mapped (Not Excavated)

84.01% of hillforts have not been excavated.

```
In [128... int_ex_none = plot_over_grey(location_excavation_data, 'Interior_Excavation_None', 'Yes')
```

Interior Excavation None



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

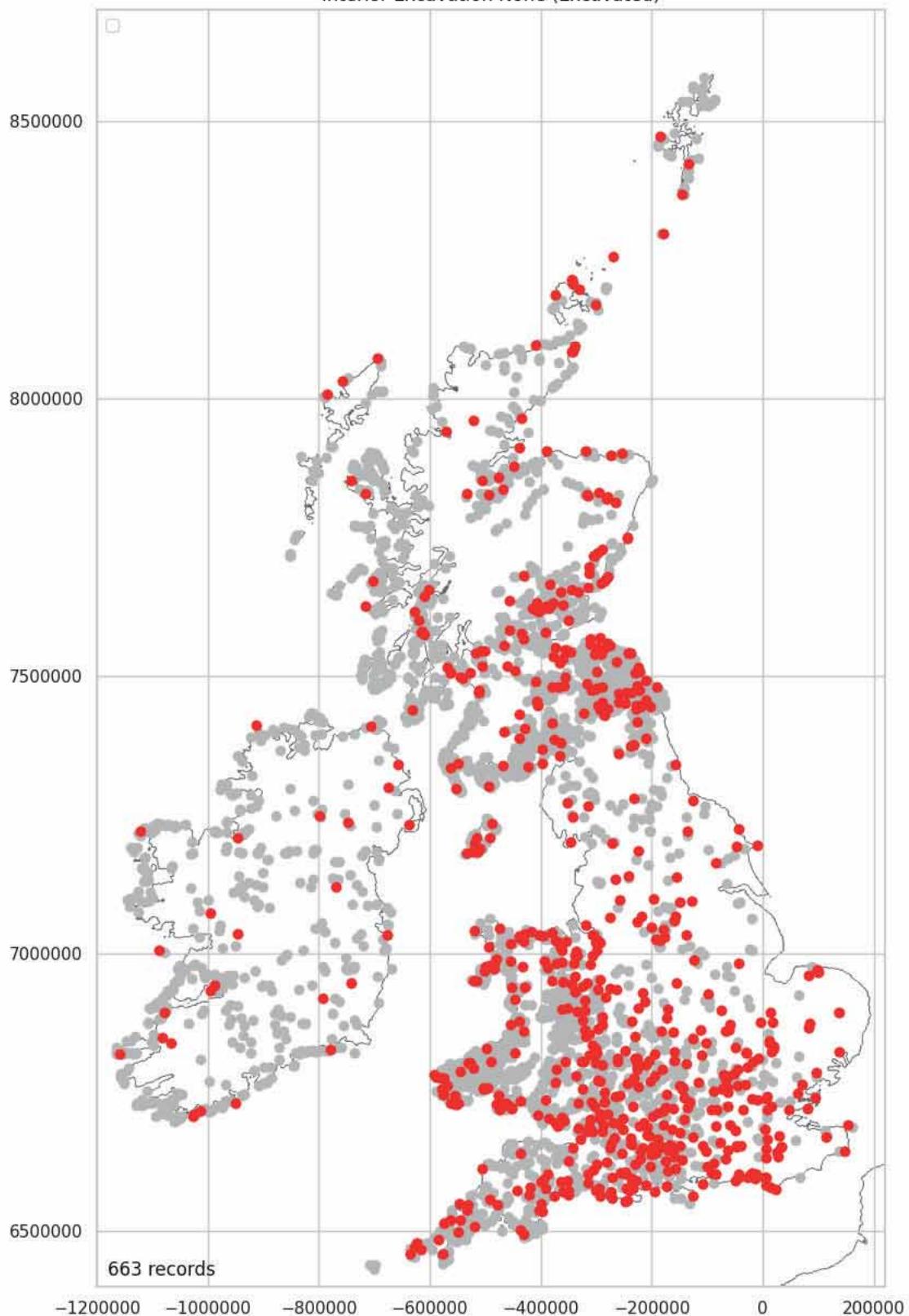
84.01%

Excavation: None Mapped (Excavated)

633 (15.99%) of hillforts have been excavated in part.

```
In [129]: int_ex = plot_over_grey(location_excavation_data, 'Interior_Excavation_None', 'No', "(Excavated)")
```

Interior Excavation None (Excavated)



Middleton, M. 2024, Hillforts Primer

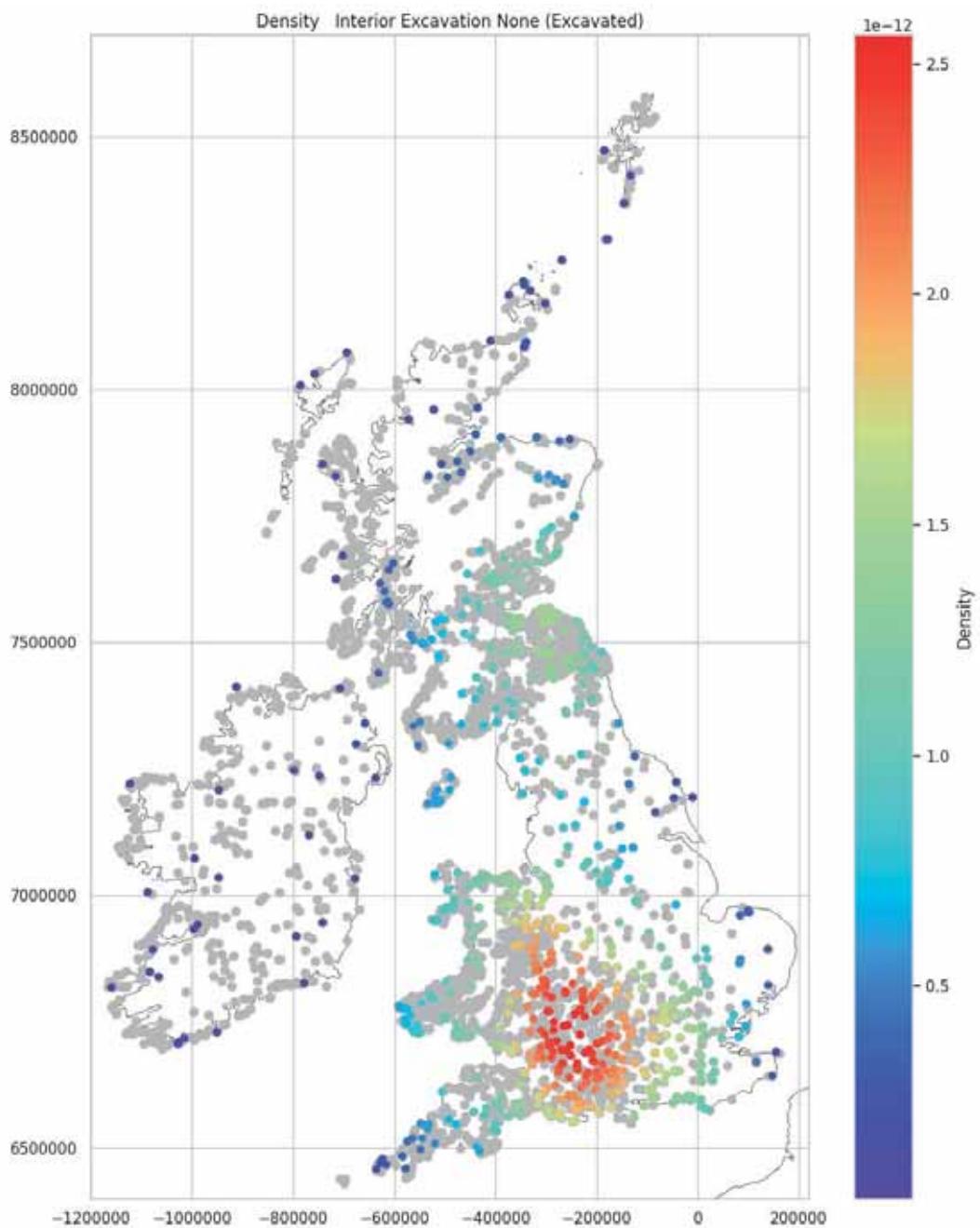
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

15. 99%

Excavation: None Density Mapped (Excavated)

The densest cluster of excavated hillforts is in south central England and up along the southern Welsh border. A secondary cluster can be seen to the eastern end of the Southern Uplands.

```
In [130]: plot_density_over_grey(int_ex, 'Interior_Excavation_None_(Excavated)')
```



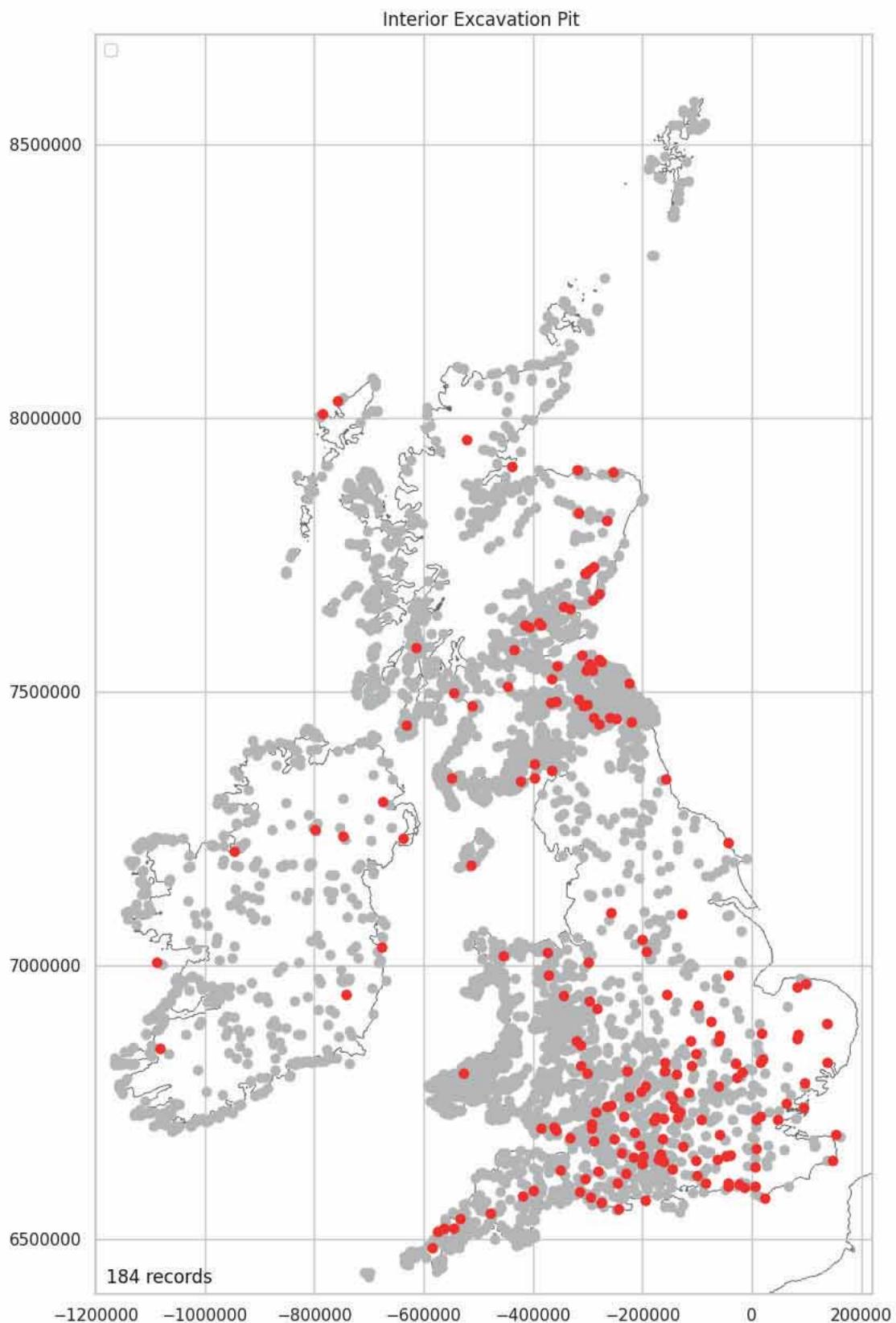
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavation: Pit Mapped

Pits are recorded at many of the southern hillforts and a good number of the northern forts. It is noticeable how few excavated forts in Wales have pits and there are also fewer recorded in the Northwest and across Ireland.

```
In [131]: int_ex_pit = plot_over_grey(location_excavation_data, 'Interior_Excavation_Pit', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 44%

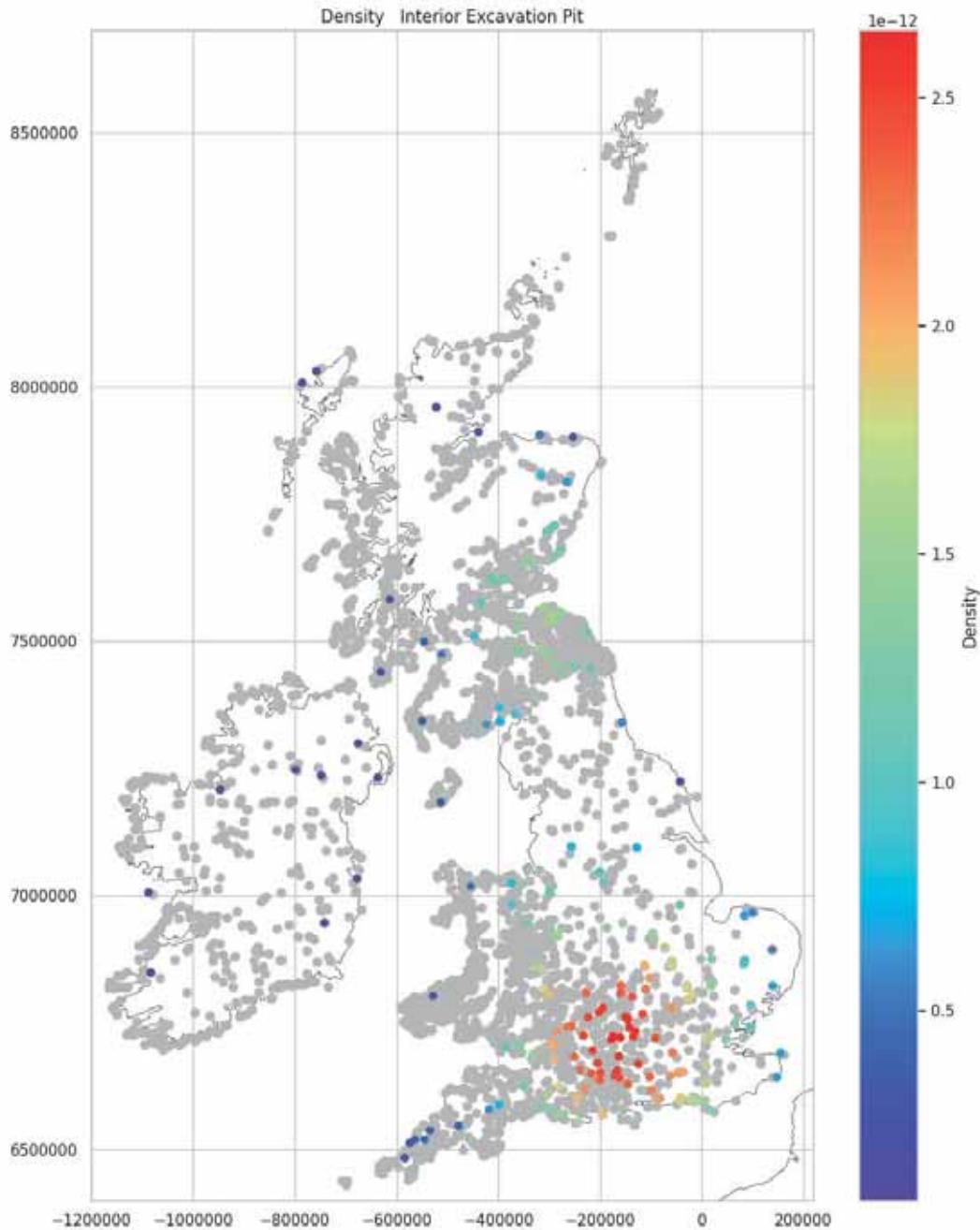
In [131...]

Excavation: Pit Density Mapped

The pit density cluster reflects the bias seen in the excavation sites data. This was focussed over south central England - See: [Excavation: None Density Mapped \(Excavated\)](#). Within that area, the excavation data clusters toward south, central England. In this pit cluster, the focus is further east and does not include the sites to the west and along the welsh border. There would suggest

that there is a meaningful distribution of pits in this limited area; This distribution being, less pits in the west and more in the east. It is probable that this is a result of the softer geology of South East England. See: [BGS Geology Viewer: S England](#).

```
In [132]: plot_density_over_grey(int_ex_pit, 'Interior_Excavation_Pit')
```



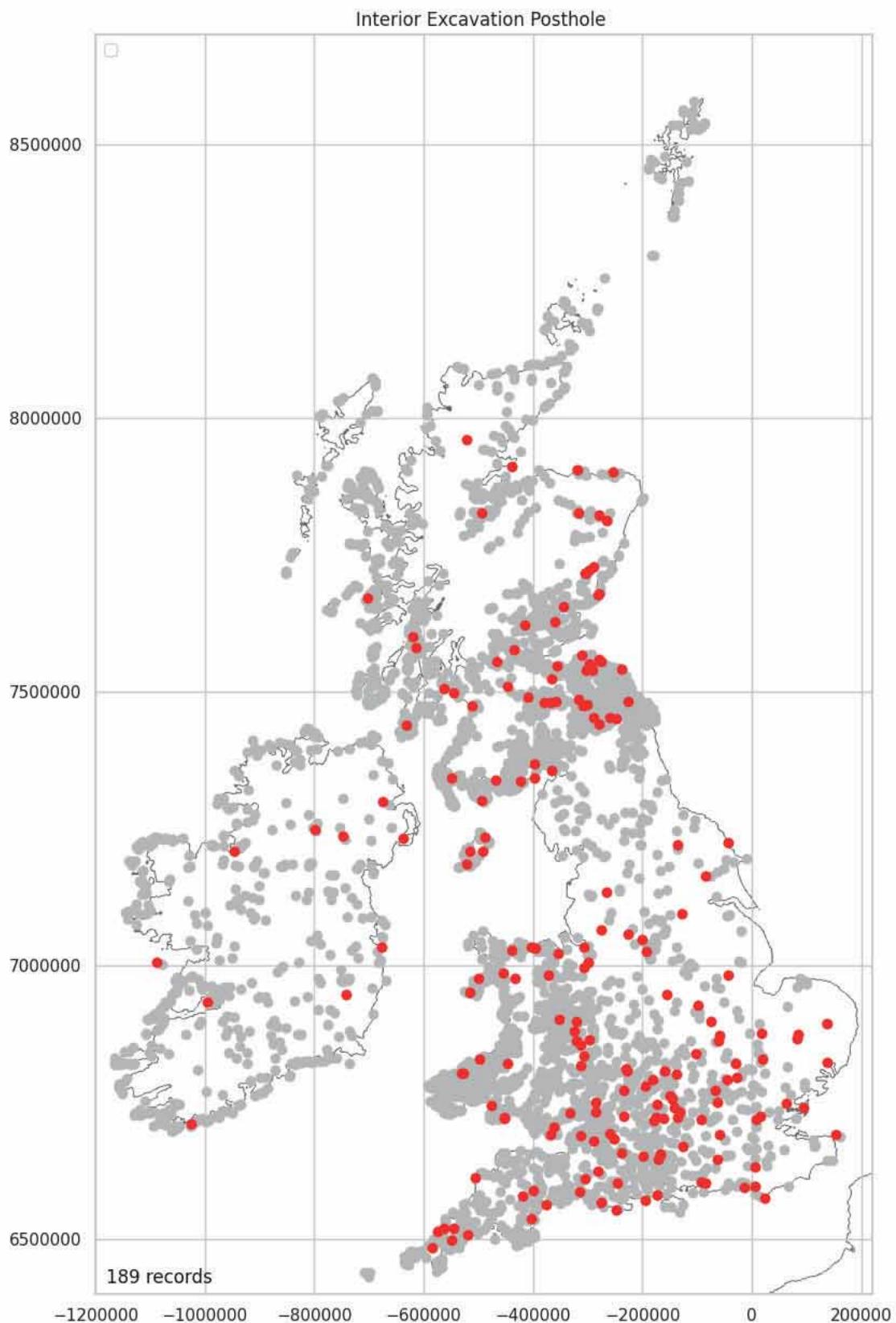
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavation: Posthole Mapped

The distribution of posthole features reflects the same bias discussed above for pit structures.

```
In [133]: int_ex_ph = plot_over_grey(location_excavation_data, 'Interior_Excavation_Posthole', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

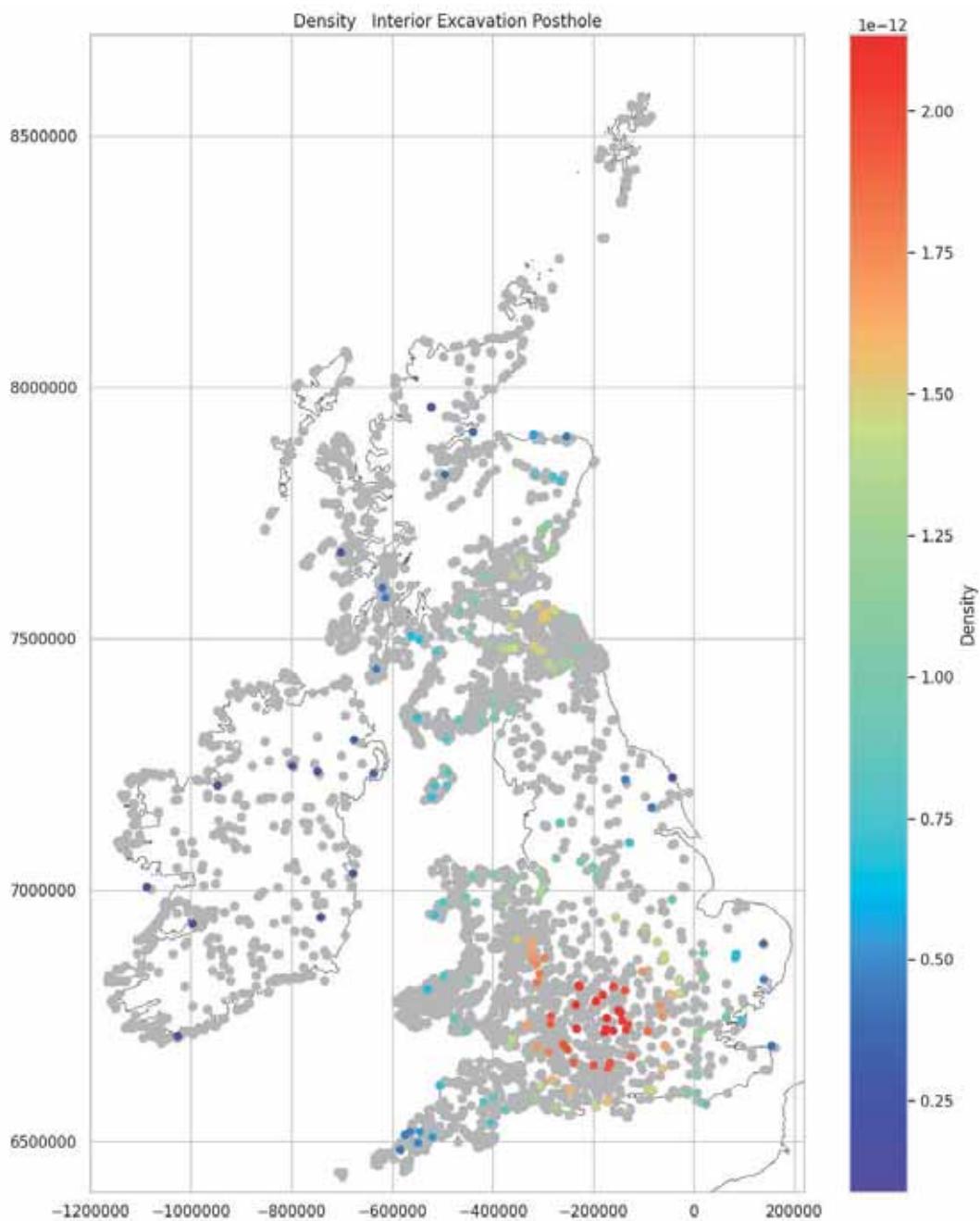
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 56%

Excavation: Posthole Density Mapped

Again the density of posthole features reflects the same bias discussed above for pit structures.

```
In [134]: plot_density_over_grey(int_ex_ph, 'Interior_Excavation_Posthole')
```



Middleton, M. 2024, Hillforts Primer

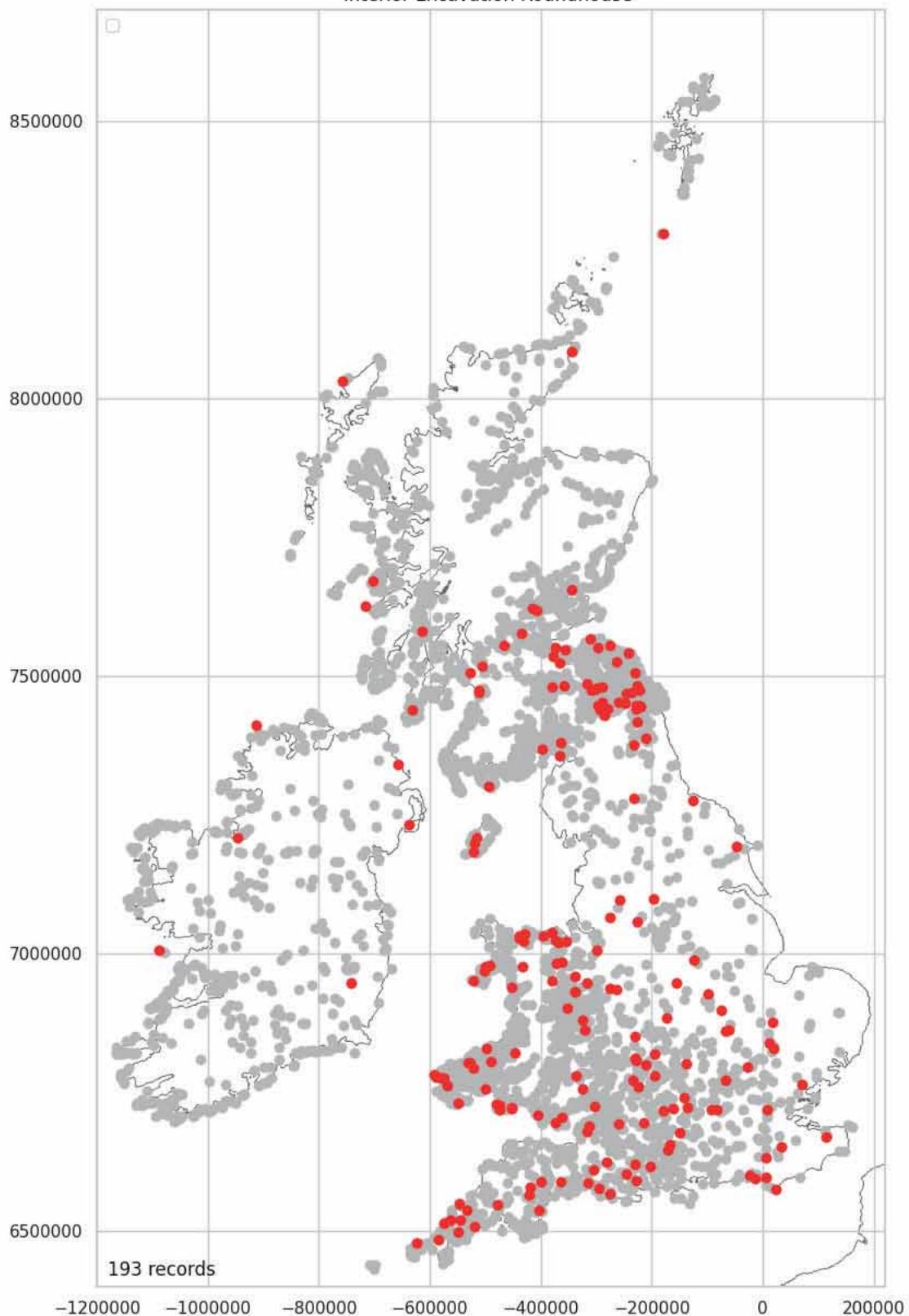
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavaion: Roundhouse Mapped

Roundhouses have been recorded widely across the excavation record. It is notable how few have been recorded in northern and western Scotland but it is possible that as roundhouses include a timber post ring, they have been recorded as posthole structures, and not roundhouses, in these areas.

```
In [135]: int_ex_rh = plot_over_grey(location_excavation_data, 'Interior_Excavation_Roundhouse', 'Yes')
```

Interior Excavation Roundhouse



Middleton, M. 2024, Hillforts Primer

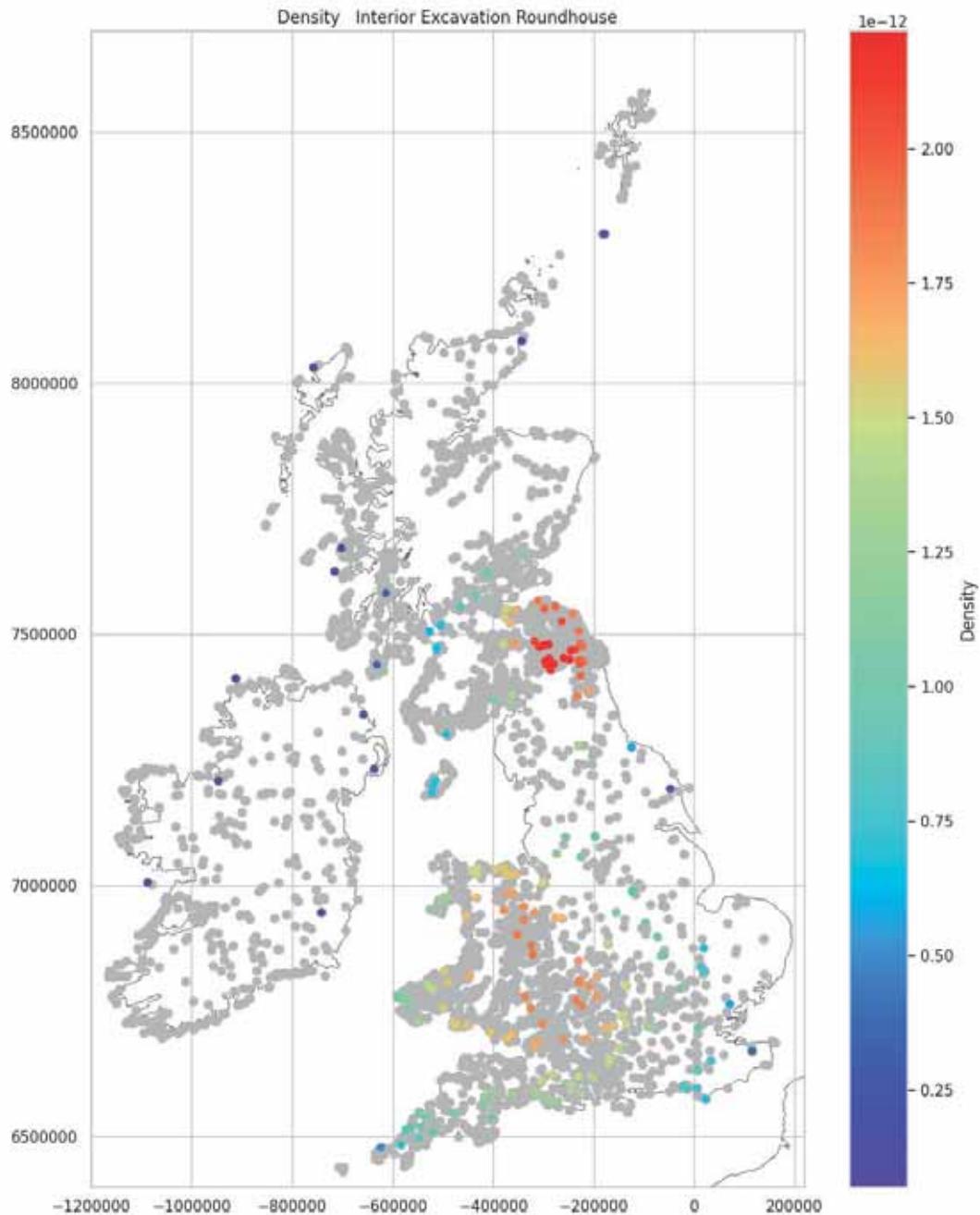
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 65%

Excavaion: Roundhouse Density

Considering the intensity of the excavation cluster over south central England and seen in [Excavation: None Density Mapped \(Excavated\)](#), it is suprising to see the most intense roundhouse cluster focussing over the eastern Southern Uplands. A secondary cluster runs up along the Welsh border. This suggests either that roundhouses are less common in the southern excavations or that the terminology used in these areas is not consistant and that roundhouses have been lumped into the posthole structures class in some areas.

```
In [136]: plot_density_over_grey(int_ex_rh, 'Interior_Excavation_Roundhouse')
```



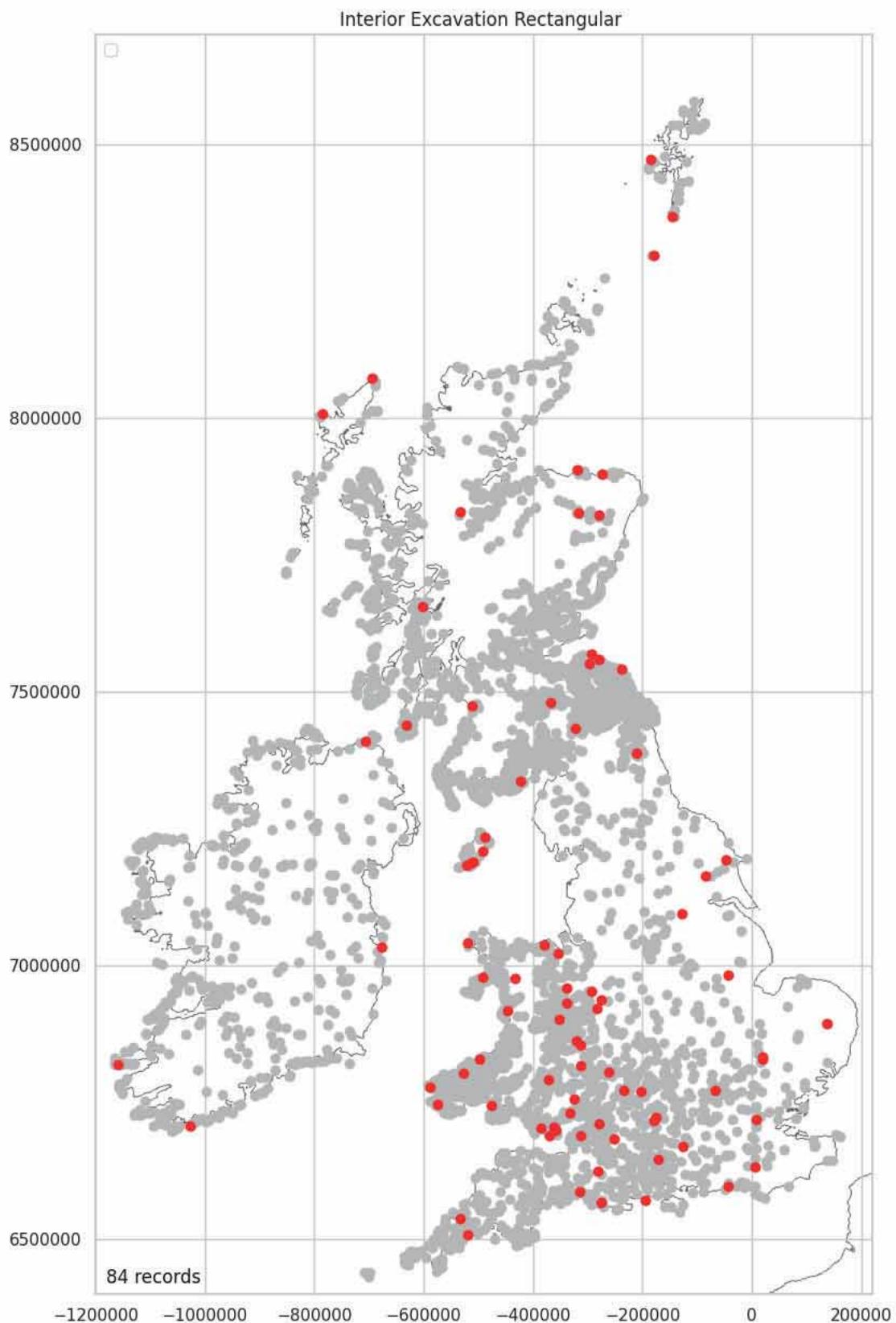
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavaion: Rectangular Mapped

There area far fewer excavated rectangular structures and most are in the south.

```
In [137]: int_ex_rect = plot_over_grey(location_excavation_data, 'Interior_Excavation_Rectangular', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

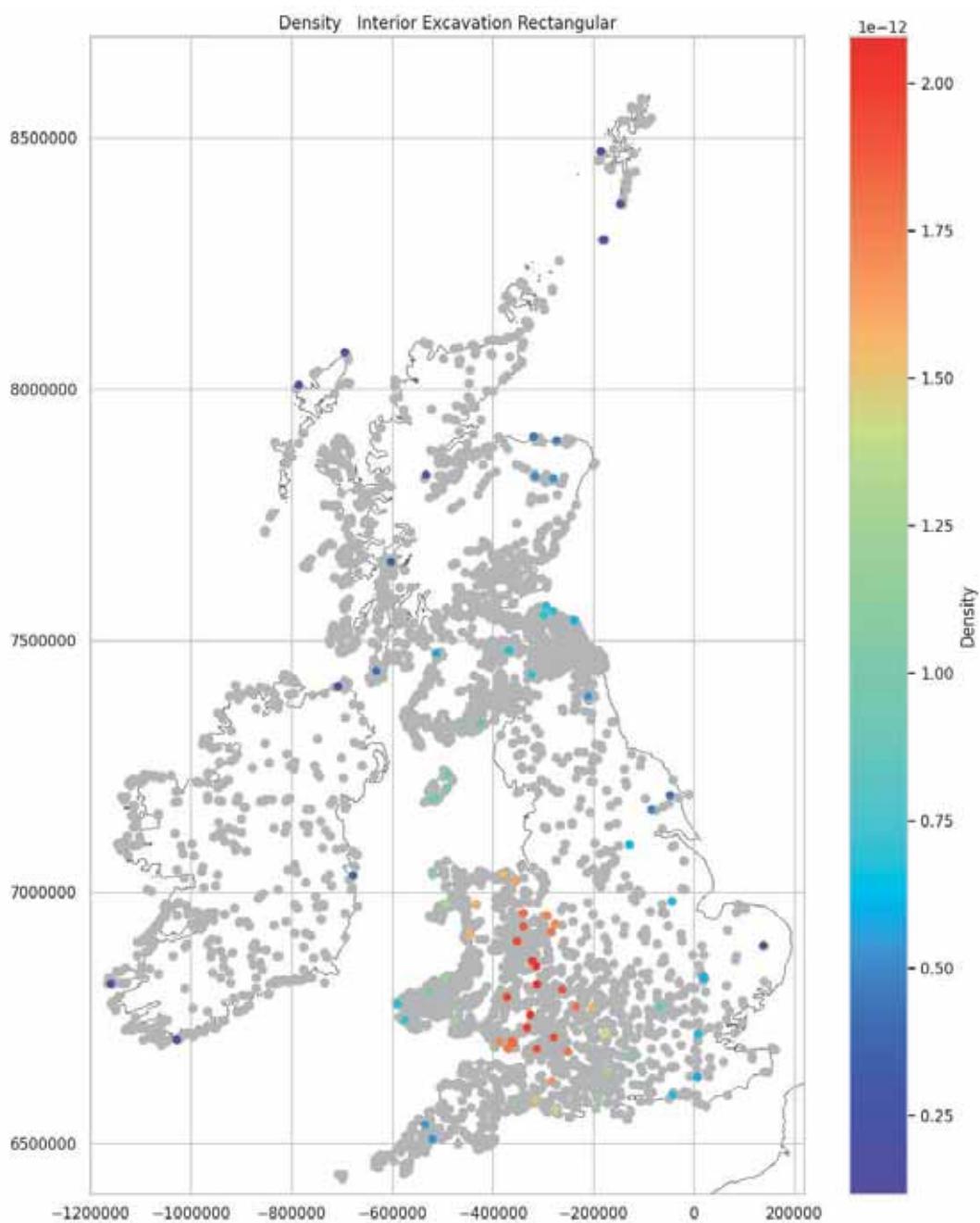
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2.03%

Excavaion: Rectangular Density Mapped

Out of the 663 excavated hillforts only 84 have revealed rectangular structures. Although these look to be clustering along the Welsh border this is also very close to the central focus of [Excavation: None Density Mapped \(Excavated\)](#) meaning the rectangular density cluster is likely to be a the result of the bias in the Excavation data. It is therefore unreliable.

```
In [138]: plot_density_over_grey(int_ex_rect, 'Interior_Excavation_Rectangular')
```



Middleton, M. 2024, Hillforts Primer

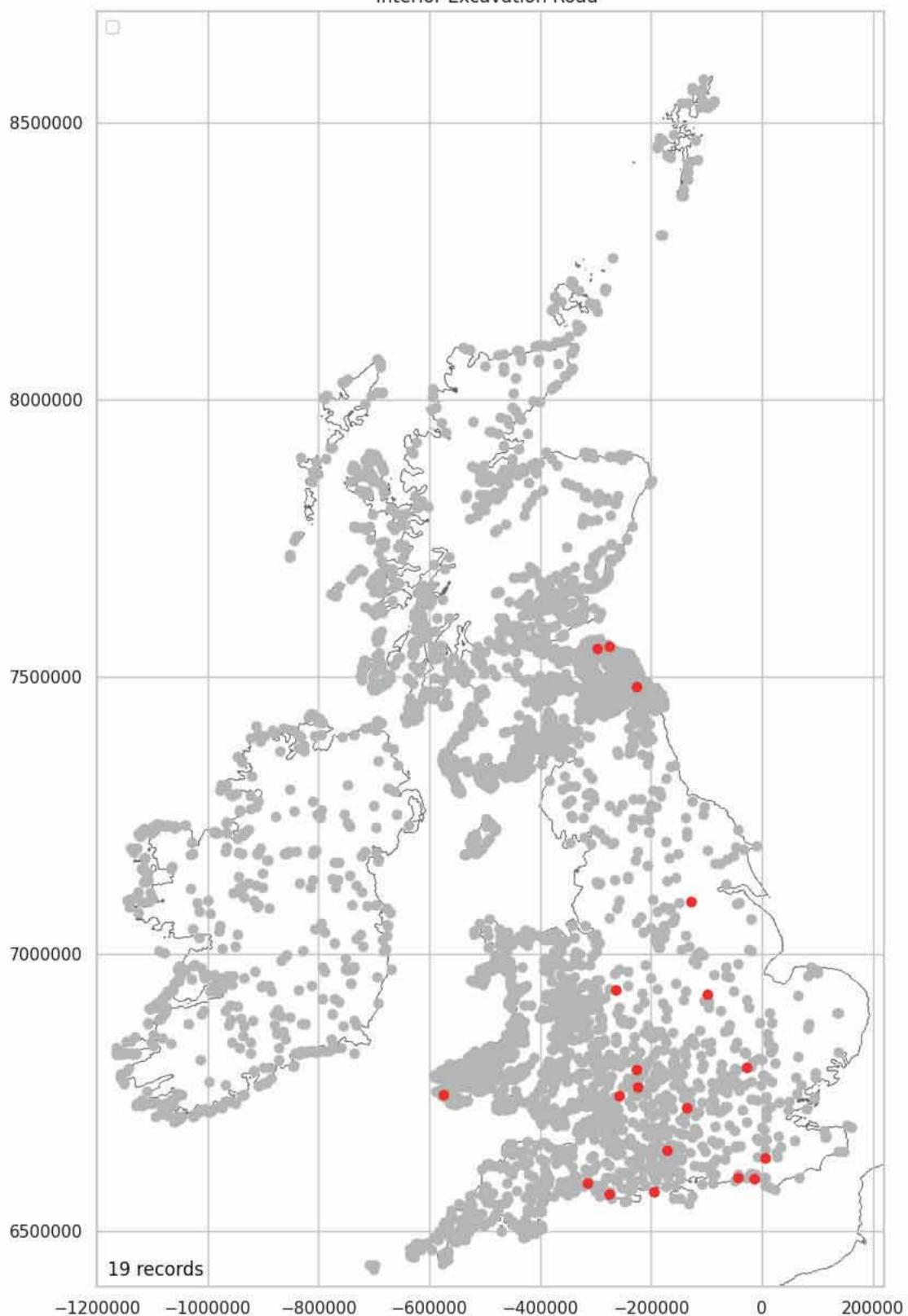
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavaion: Road Mapped

Excavated examples of roads have been identified at 19 hillforts.

```
In [139]: int_ex_road = plot_over_grey(location_excavation_data, 'Interior_Excavation_Road', 'Yes')
```

Interior Excavation Road



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

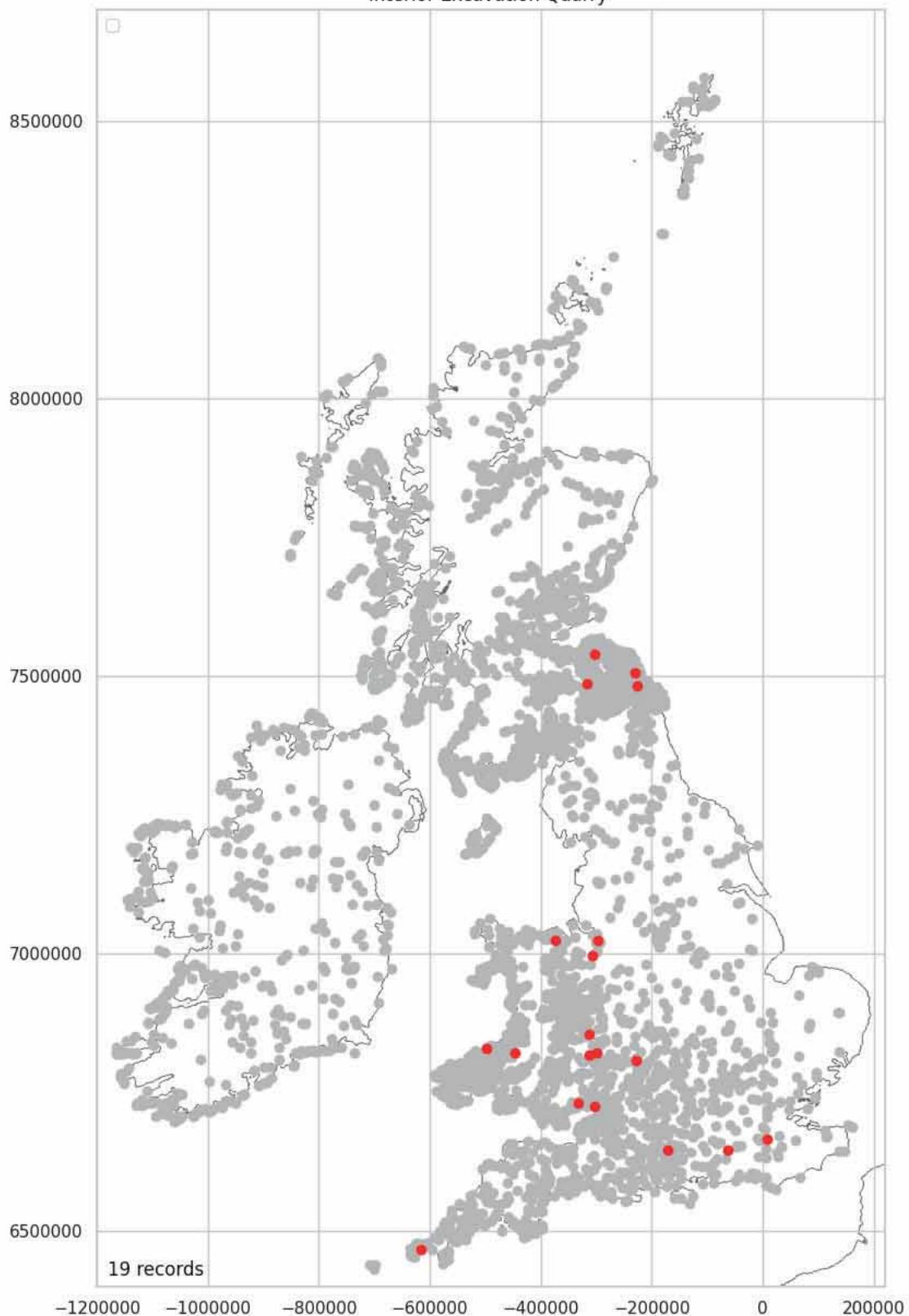
0.46%

Excavation: Quarry Mapped

Excavated examples of quarries have been identified at 19 hillforts.

```
In [140]: int_ex_quarry = plot_over_grey(location_excavation_data, 'Interior_Excavation_Quarry', 'Yes')
```

Interior Excavation Quarry



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

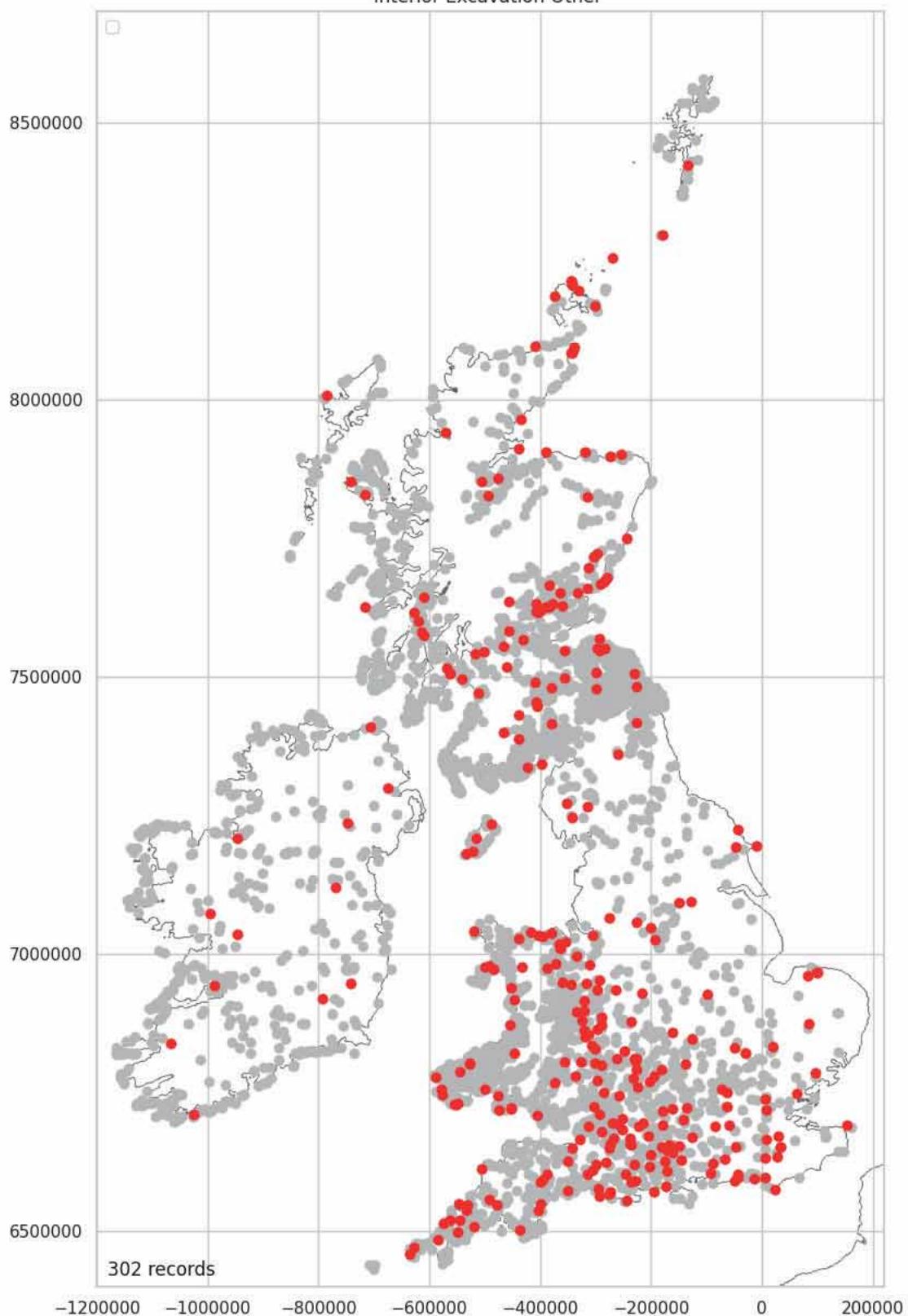
0.46%

Excavaion: Other Mapped

There are 302 hillforts where 'other' structures have been excavated. No further detail is given.

```
In [141]: int_ex_other = plot_over_grey(location_excavation_data, 'Interior_Excavation_Other', 'Yes')
```

Interior Excavation Other



Middleton, M. 2024, Hillforts Primer

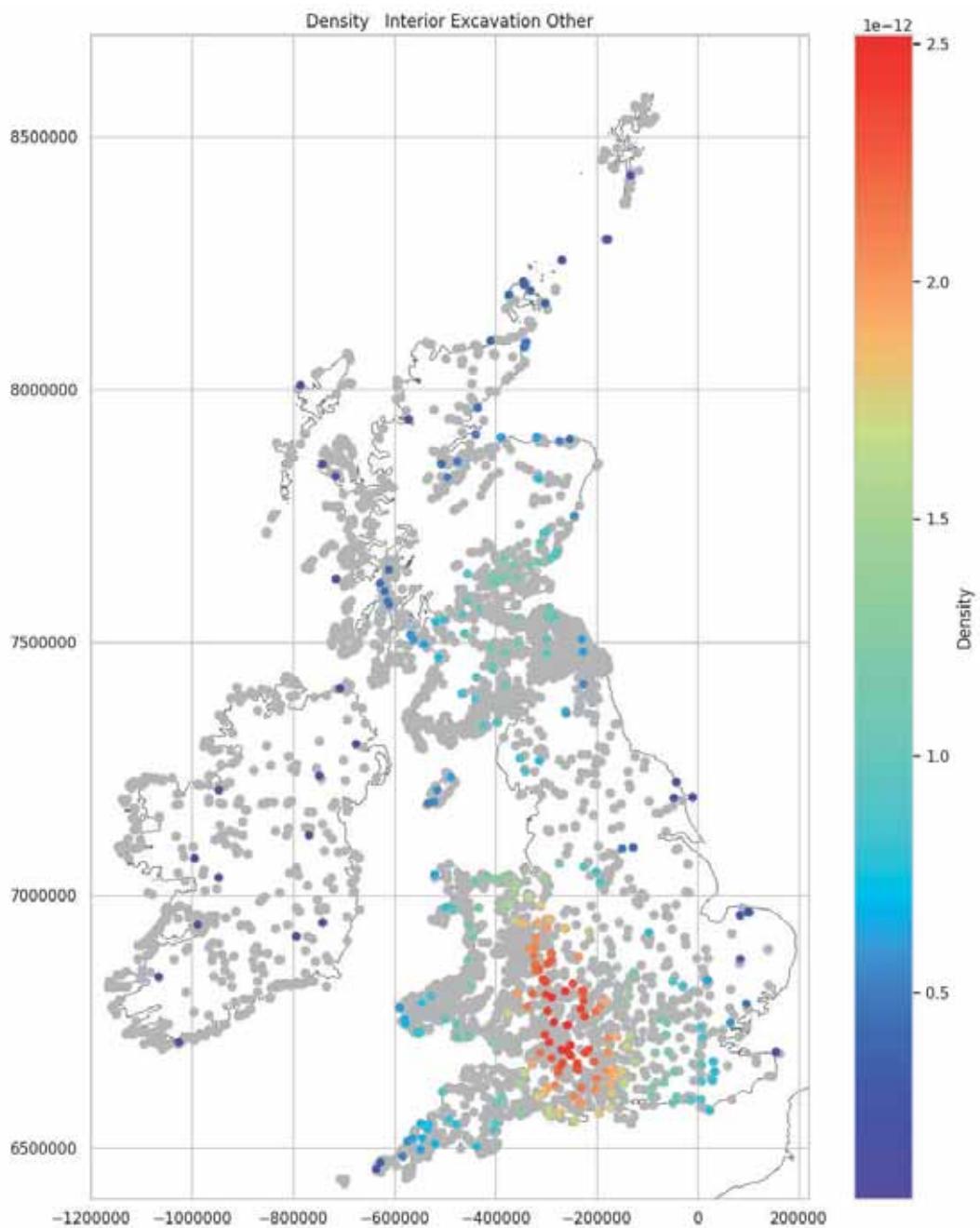
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

7.28%

Excavaion: Other Density Mapped

The clustering of 'other' structures mirrors that seen and discussed in [Excavation: None Density Mapped \(Excavated\)](#).

```
In [142]: plot_density_over_grey(int_ex_other, 'Interior_Excavation_Other')
```



Middleton, M. 2024, Hillforts Primer

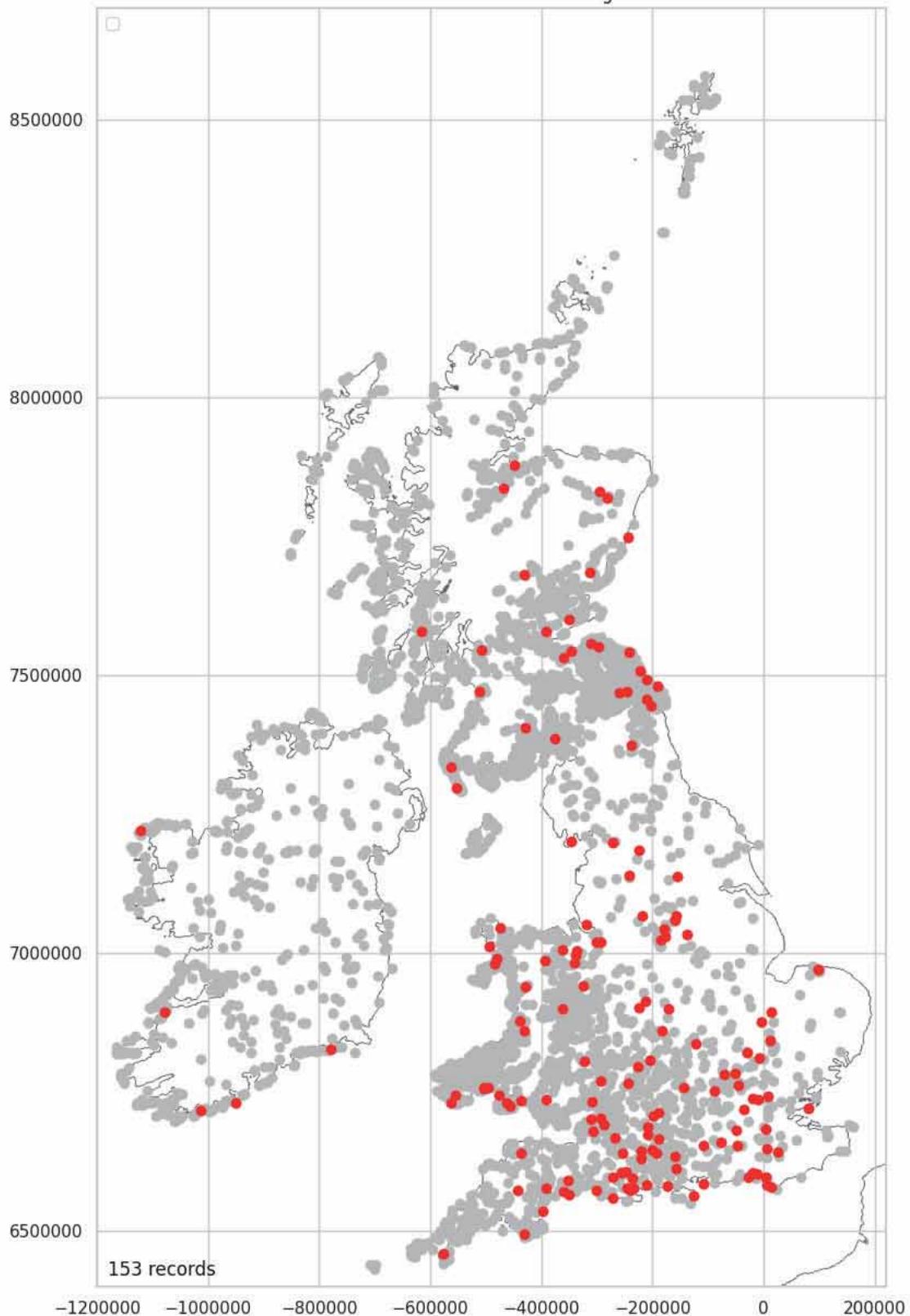
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Excavation: Nothing Mapped

3.69% of excavated hillforts identified no internal structures. It is not clear if this is because the excavations were focussed on the ramparts or if these are excavations in the interior of forts where no structures were identified.

```
In [143]: int_ex_nothing = plot_over_grey(location_excavation_data, 'Interior_Excavation_Nothing', 'Yes')
```

Interior Excavation Nothing



Middleton, M. 2024, Hillforts Primer

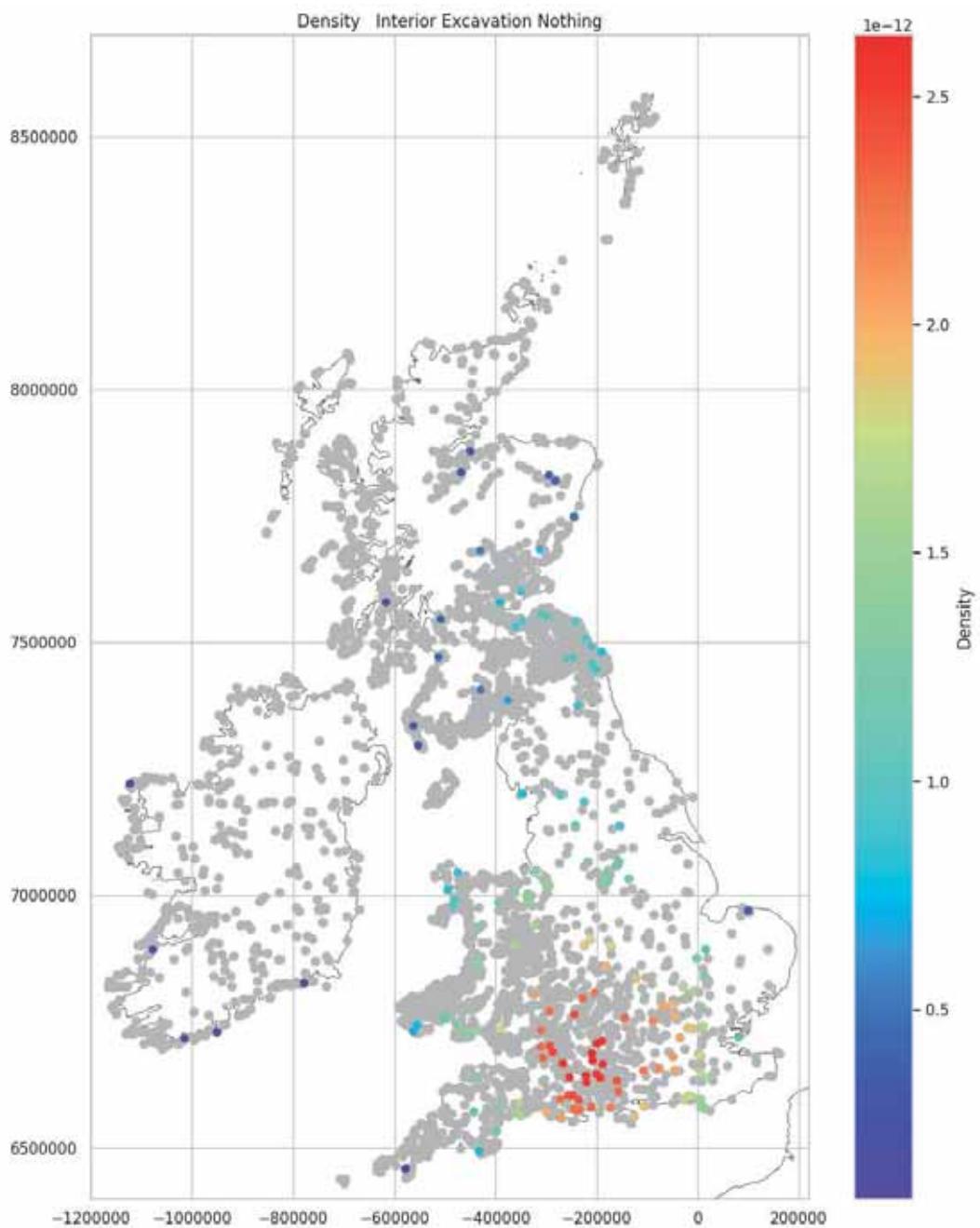
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3. 69%

Excavaion: Nothing Density Mapped

The dominenat cluster for this data mirrors that seen in [Excavation: None Density Mapped \(Excavated\)](#).

```
In [144]: plot_density_over_grey(int_ex_nothing, 'Interior_Excavation_Nothing')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Geophysics Data

```
In [145...]: geophysics_features = [
    'Interior_Geophysics_None',
    'Interior_Geophysics_Pit',
    'Interior_Geophysics_Roundhouse',
    'Interior_Geophysics_Rectangular',
    'Interior_Geophysics_Road',
    'Interior_Geophysics_Quarry',
    'Interior_Geophysics_Other',
    'Interior_Geophysics_Nothing']

geophysics_data = interior_encodeable_data[geophysics_features]
geophysics_data.head()
```

	Interior_Geophysics_None	Interior_Geophysics_Pit	Interior_Geophysics_Roundhouse	Interior_Geophysics_Rectangular	Interior_Geophysics_
0	Yes	No	No	No	No
1	Yes	No	No	No	No
2	Yes	No	No	No	No
3	Yes	No	No	No	No
4	Yes	No	No	No	No

There are no null values

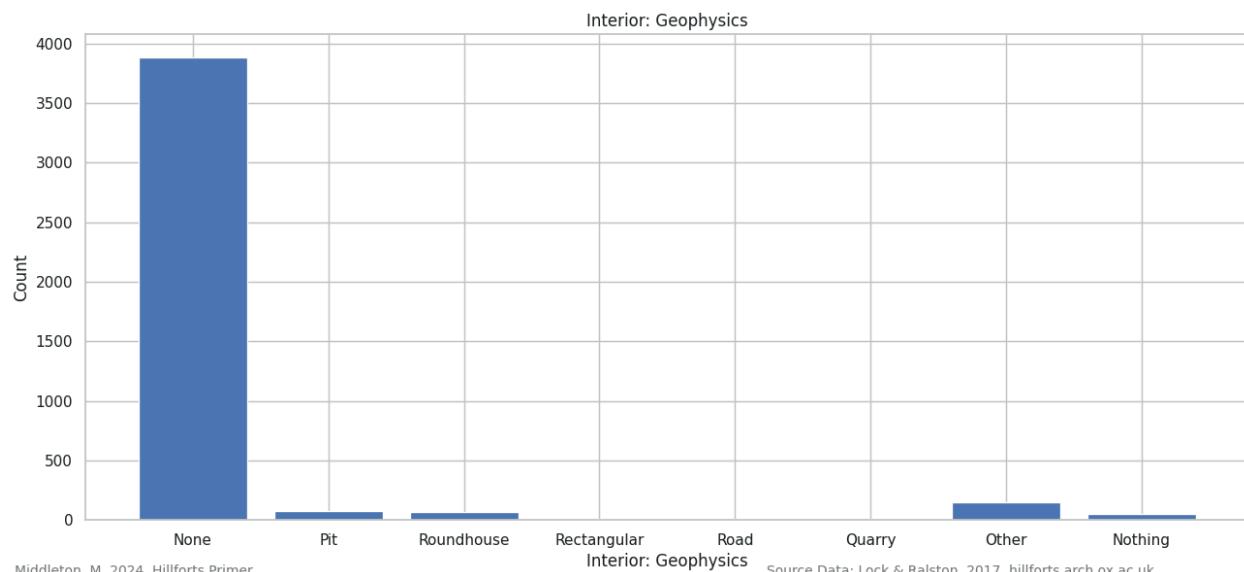
In [146...]: `geophysics_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Interior_Geophysics_None    4147 non-null   object 
 1   Interior_Geophysics_Pit     4147 non-null   object 
 2   Interior_Geophysics_Roundhouse 4147 non-null   object 
 3   Interior_Geophysics_Rectangular 4147 non-null   object 
 4   Interior_Geophysics_Road      4147 non-null   object 
 5   Interior_Geophysics_Quarry    4147 non-null   object 
 6   Interior_Geophysics_Other     4147 non-null   object 
 7   Interior_Geophysics_Nothing   4147 non-null   object 
dtypes: object(8)
memory usage: 259.3+ KB
```

Geophysics Data Plotted

No geophysics ('none') dominates the geophysics plot and will be removed to facilitate reading the other results.

In [147...]: `plot_bar_chart(geophysics_data, 2, 'Interior: Geophysics', 'Count', 'Interior: Geophysics')`



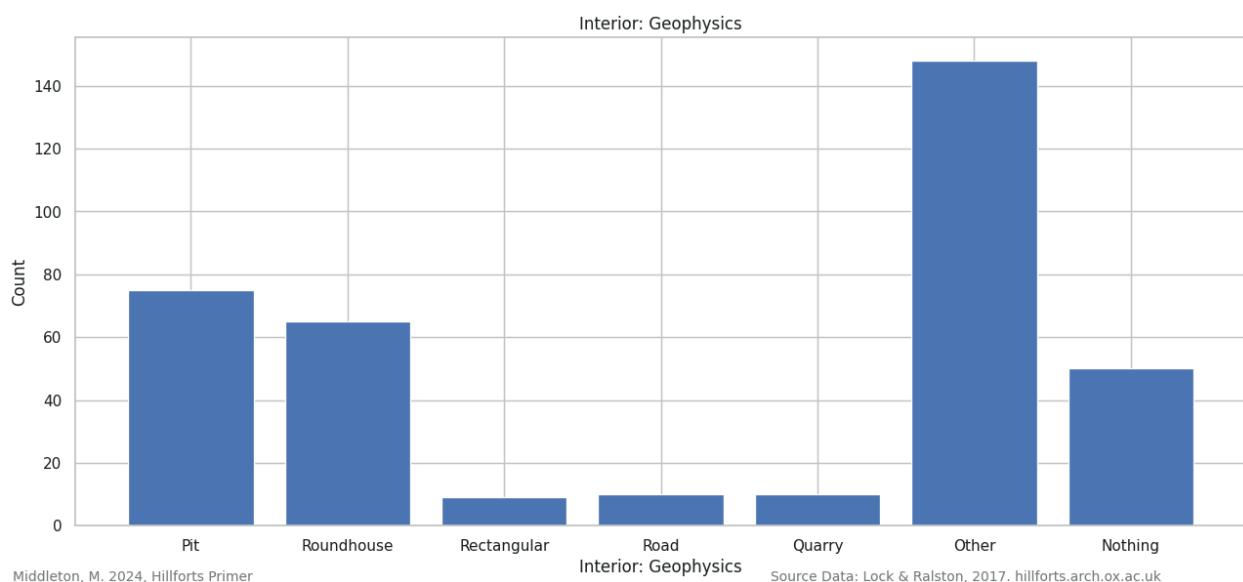
Geophysics Data Plotted (Excluding None)

Pits, roundhouses, other and nothing are the dominant classes in the geophysics data.

In [148...]: `geophysics_data_mius = geophysics_data.drop(['Interior_Geophysics_None'], axis=1)`
`geophysics_data_mius.head()`

	Interior_Geophysics_Pit	Interior_Geophysics_Roundhouse	Interior_Geophysics_Rectangular	Interior_Geophysics_Road	Interior_Geophysics_
0	No	No	No	No	No
1	No	No	No	No	No
2	No	No	No	No	No
3	No	No	No	No	No
4	No	No	No	No	No

```
In [149]: plot_bar_chart(geophysics_data_minus, 2, 'Interior: Geophysics', 'Count', 'Interior: Geophysics')
```



Geophysics & Excavation Data Plotted (Excluding None)

An posthole feature has been temporarily added to the geophysics data so the data can be plotted against the excavation data.

See: [Surface Data Plotted \(Excluding None\)](#)

```
In [150]: temp_geophysics = geophysics_data_minus.copy()
temp_geophysics['Interior_Geophysics_Posthole'] = 'No'
temp_geophysics.head()
```

	Interior_Geophysics_Pit	Interior_Geophysics_Roundhouse	Interior_Geophysics_Rectangular	Interior_Geophysics_Road	Interior_Geophysics_
0	No	No	No	No	No
1	No	No	No	No	No
2	No	No	No	No	No
3	No	No	No	No	No
4	No	No	No	No	No

The data is reordered to match the excavation data structure.

```
In [151]: temp_geophysics = temp_geophysics[
    ['Interior_Geophysics_Pit',
     'Interior_Geophysics_Posthole',
     'Interior_Geophysics_Roundhouse',
     'Interior_Geophysics_Rectangular',
     'Interior_Geophysics_Road',
     'Interior_Geophysics_Quarry',
     'Interior_Geophysics_Other',
     'Interior_Geophysics_Nothing']]
```

265 hillforts have had geophysics surveys carried out within them.

```
In [152]: geophys_forts = 4147 - sum(geophysics_data['Interior_Geophysics_None']=="Yes")
geophys_forts
```

Out[152]: 265

50 hillforts (18.87% of those surveyed) revealed no internal features.

```
In [153]: geophys_nothing = sum(geophysics_data['Interior_Geophysics_Nothing']=='Yes')  
geophys_nothing
```

Out[153]: 50

```
In [154]: geophys_nothing_pcnt = round((geophys_nothing / geophys_forts) * 100, 2)  
geophys_nothing_pcnt
```

Out[154]: 18.87

Pits and roundhouses are the dominant named structure recorded. Unnamed other structures are by far the most dominant.

```
In [155]: for feature in geophysics_features[1:-1]:  
    print(feature + ": " + str(sum(geophysics_data[feature]=='Yes')))
```

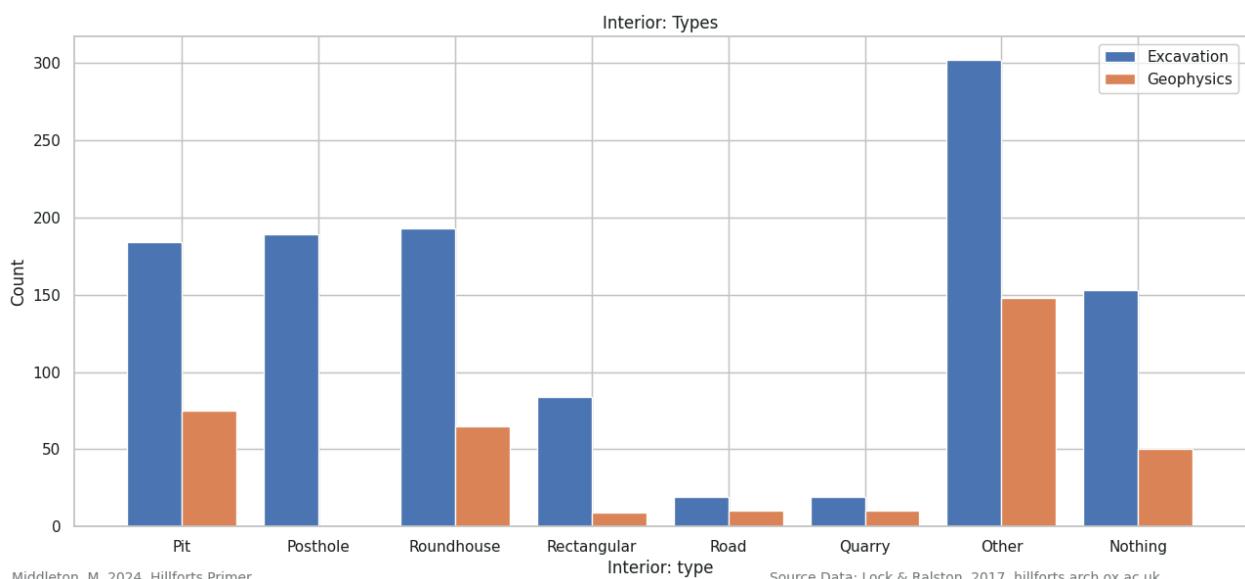
```
Interior_Geophysics_Pit: 75  
Interior_Geophysics_Roundhouse: 65  
Interior_Geophysics_Rectangular: 9  
Interior_Geophysics_Road: 10  
Interior_Geophysics_Quarry: 10  
Interior_Geophysics_Other: 148
```

```
In [156]: geophys_other_pcnt = round((sum(geophysics_data['Interior_Geophysics_Other']=='Yes') / geophys_forts) * 100, 2)  
geophys_other_pcnt
```

Out[156]: 55.85

Excavations have found more of each structure because there have been more excavations.

```
In [157]: plot_bar_chart_two(excavation_data_minus, temp_geophysics, 2, 'Interior_type', 'Count', 'Interior_Types')
```

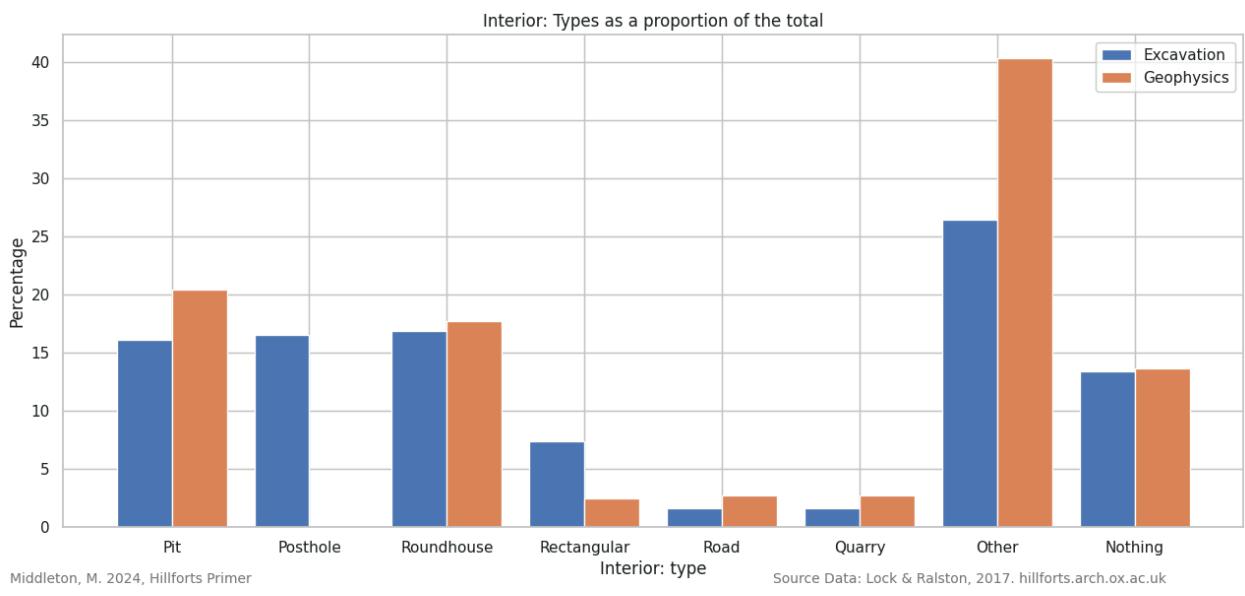


Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Proportionally, excavation and geophysics are finding roughly the same quantity of each structure except for pits, posthole and rectangular structures. Rectangular structures are being found but posthole structures have not to be specifically identified as a class in the geophysics data. Interestingly, geophysics is proportionally identifying more 'other' features than excavation and this difference is similar to the proportion of posthole structure identified in excavation. It is likely that geophysics is recording posthole structures within the 'other' category. If this is the case, excavation and geophysics are identifying very similar proportions of features within hillforts. The difference in pits may possibly be accounted for by geophysics cataloguing naturally occurring caustic features as pits which would be dismissed under excavation.

```
In [158]: plot_bar_chart_two(excavation_data_minus, temp_geophysics, 2, 'Interior_type', 'Percentage', 'Interior_Types as a
```



Geophysics Data Mapped

Only 265 (6.39%) of hillforts have been surveyed using geophysics and the majority of surveys cluster around Oxford University and the head office of Historic England in Swindon. Within this small area pits seem to follow a similar distribution to those seen in excavations but roundhouses and hillforts containing no structures show quite different distributions. Because of the survey bias and the small numbers of hillforts in each category, it is important to not over interpret these differences.

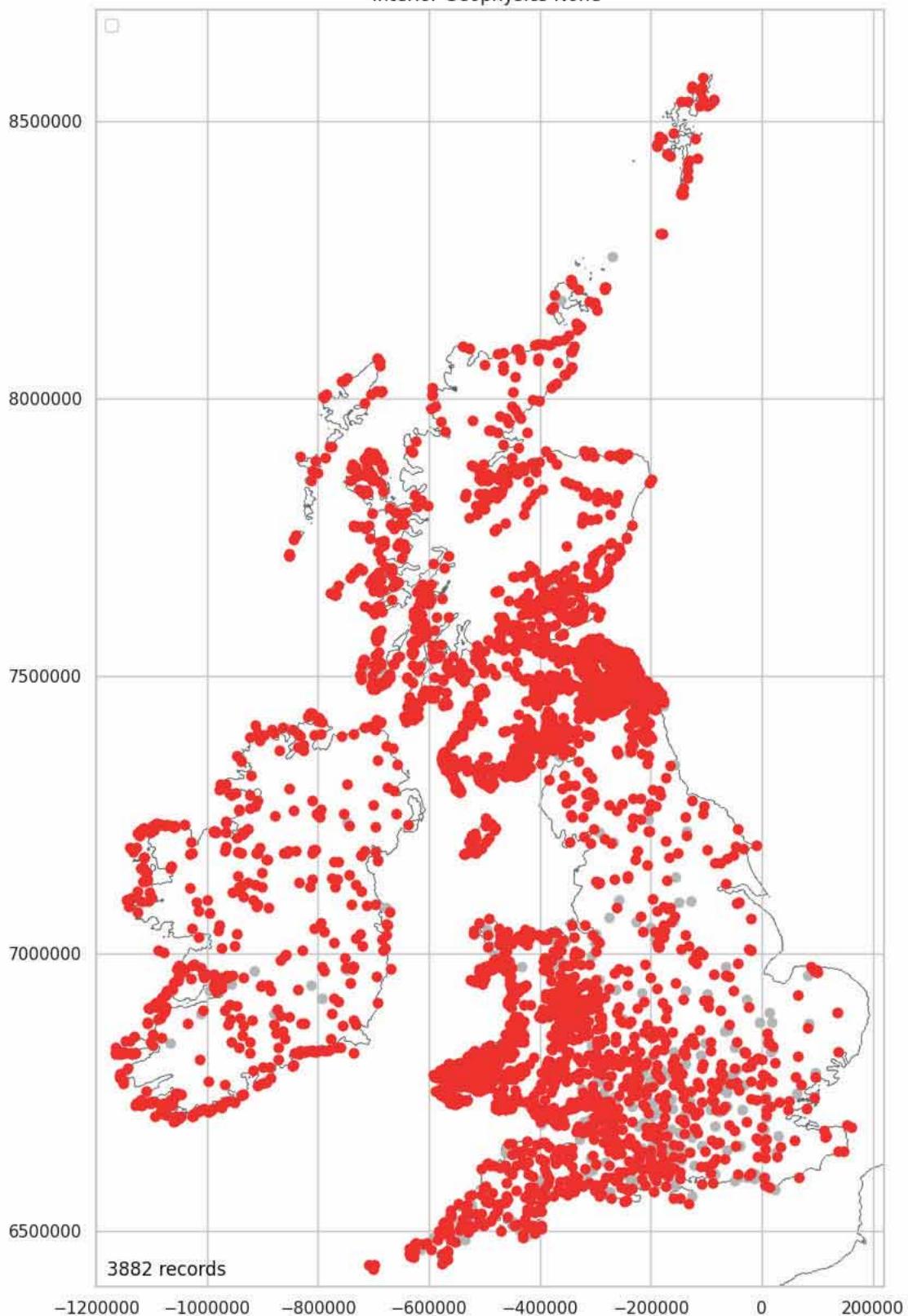
```
In [159]: location_geophysics_data = pd.merge(location_numeric_data_short, geophysics_data, left_index=True, right_index=True)
```

Geophysics: None Mapped (Not Surveyed)

Most (93.61%) hillforts have not been surveyed using geophysics equipment.

```
In [160]: int_geo_none = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_None', 'Yes')
```

Interior Geophysics None



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

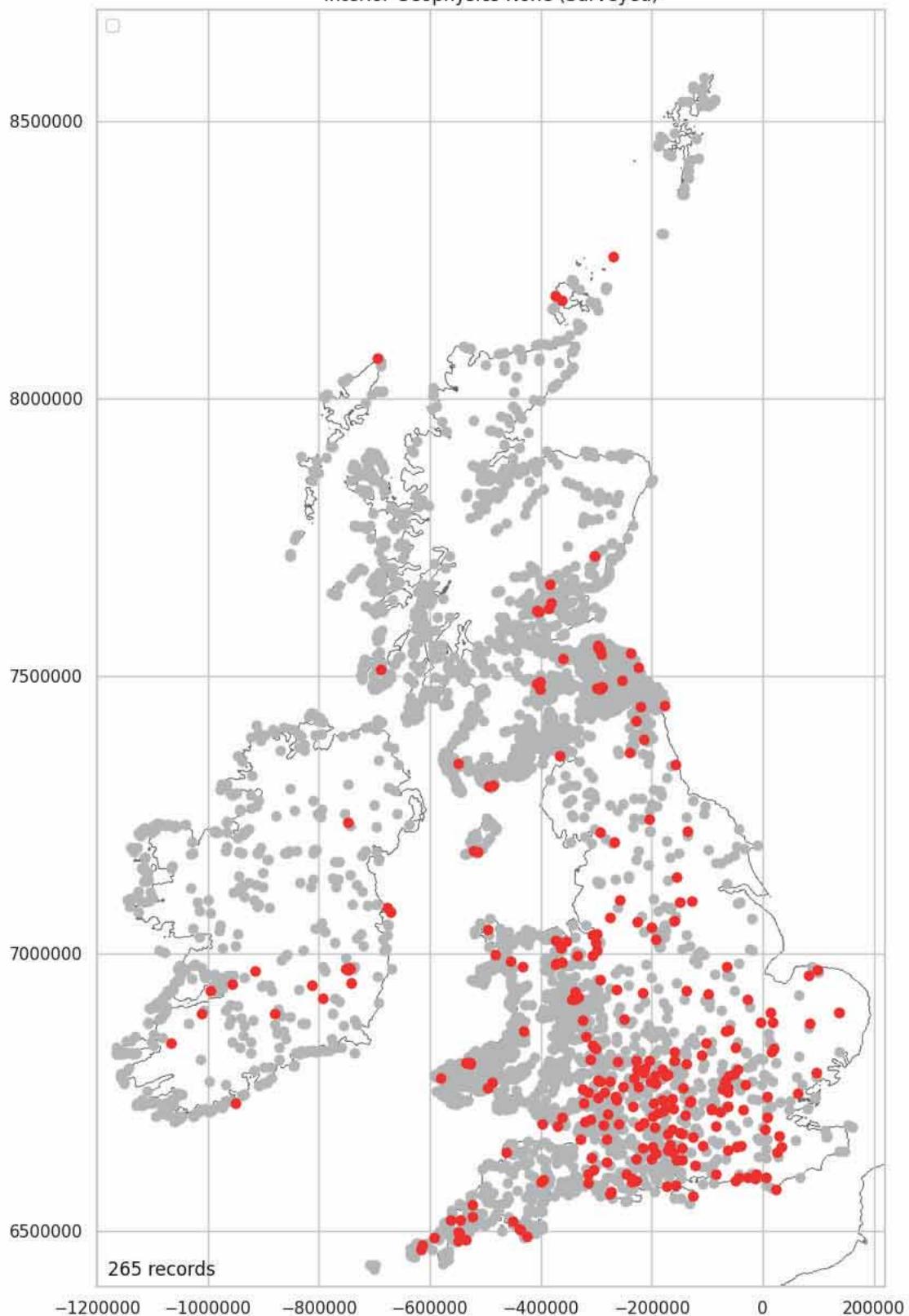
93.61%

Geophysics: None Mapped (Surveyed)

Similar to excavations, the majority of geophysics surveys have been carried out in south central England.

```
In [161]: int_geo_none = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_None', 'No', "(Surveyed)")
```

Interior Geophysics None (Surveyed)



Middleton, M. 2024, Hillforts Primer

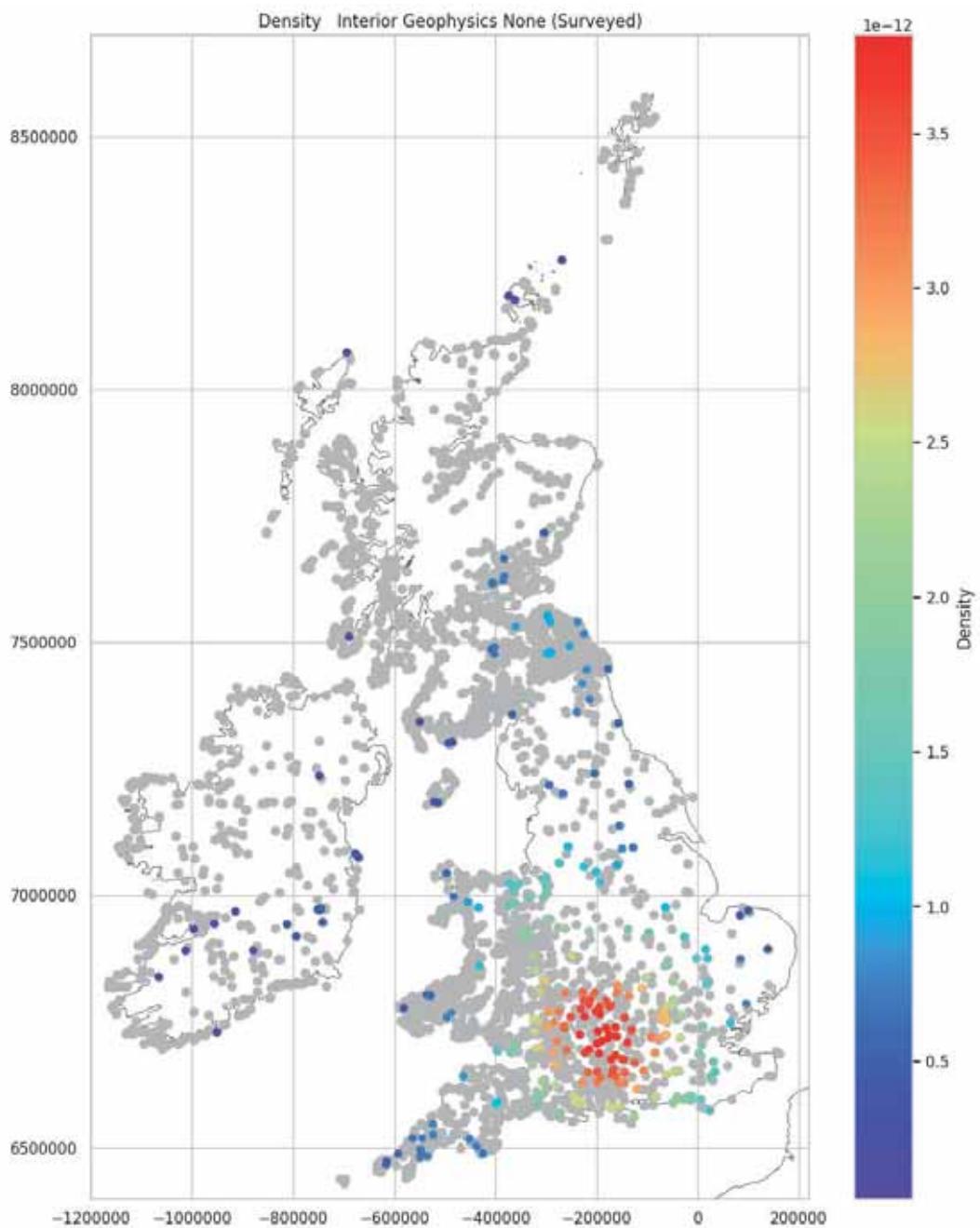
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6.39%

Geophysics: None Density Mapped (Surveyed)

The cluster is similar in location to that seen in [Excavation: None Density Mapped \(Excavated\)](#) but it is focussed more to the east.

```
In [162]: plot_density_over_grey(int_geo_none, 'Interior_Geophysics_None_(Surveyed)')
```



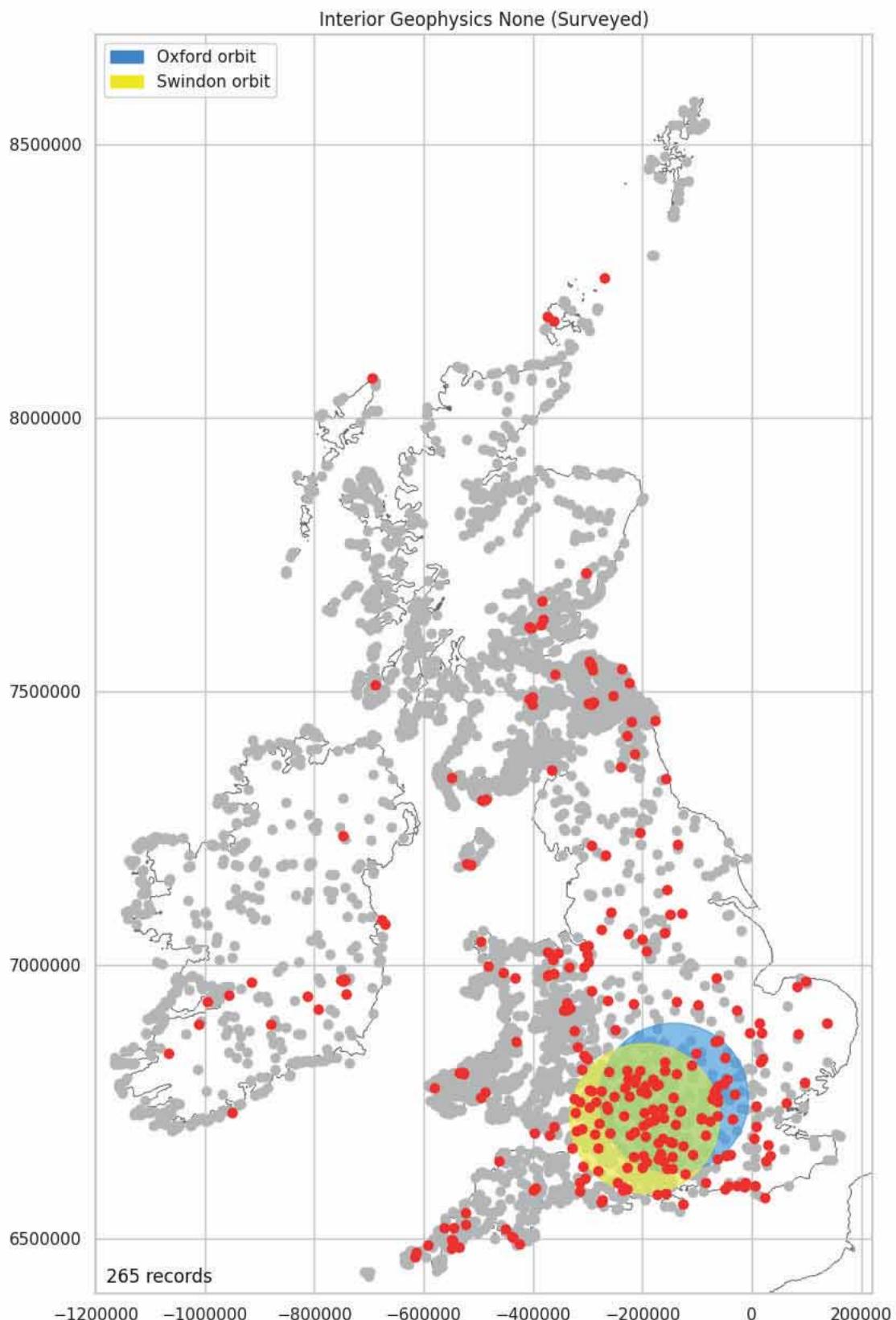
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Geophysics: None Mapped (Surveyed) Plus Oxford and Swindon Orbits

There is a significant survey bias. The most dense concentration of surveyed hillforts coincides with the overlapping orbits of Oxford University and the Historic England head office in Swindon.

```
In [163]: geophys_none = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_None', 'No', "(Surveyed)", False, False)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

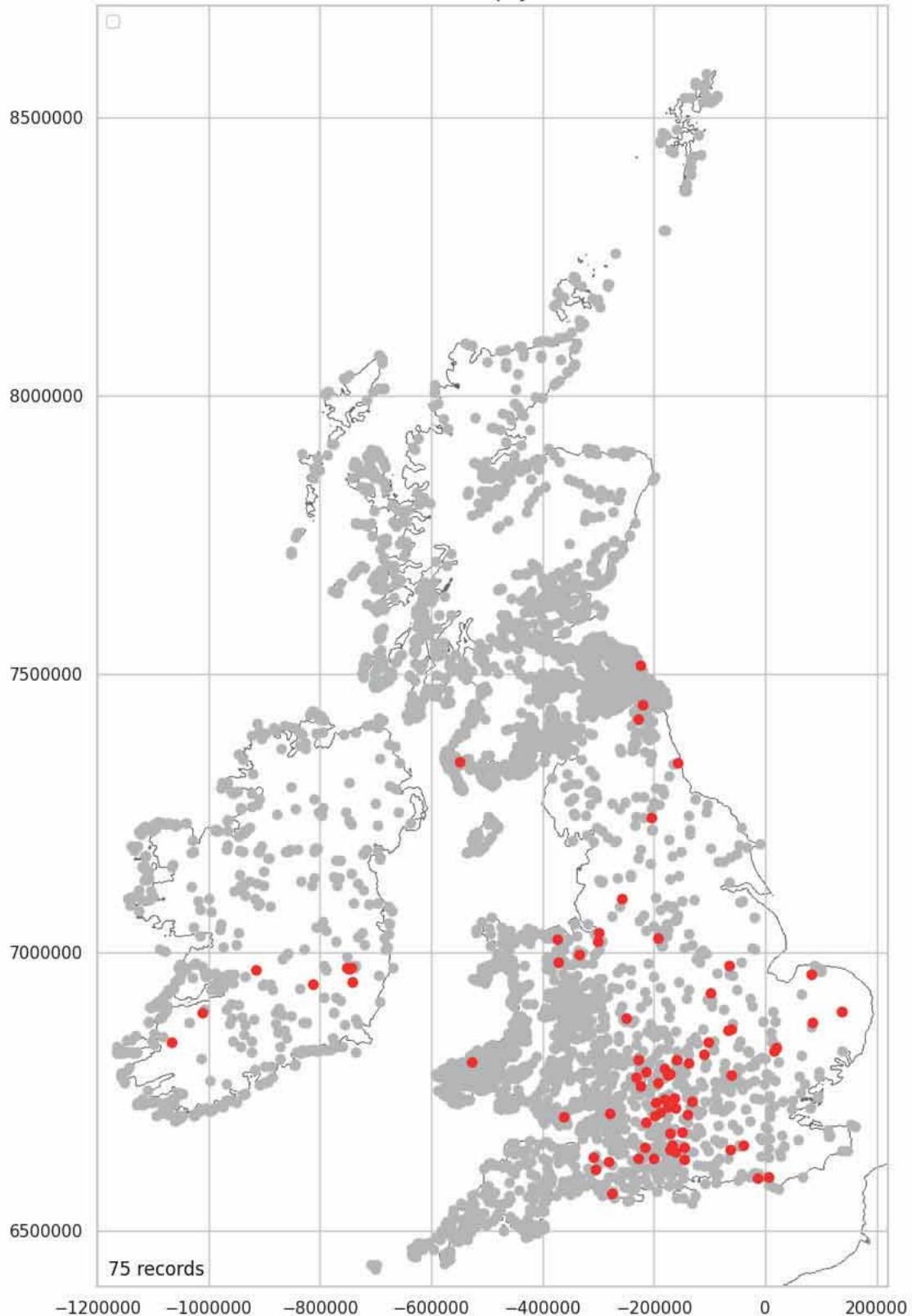
6.39%

Geophysics: Pit Mapped

Pits show the same survey bias as discussed in [Geophysics: None Mapped \(Surveyed\) Plus Oxford and Swindon Orbits](#) and they show a similar distribution, within this small area, to the excavated pits discussed in [Excavation: Pit Density Mapped](#).

```
In [164]: int_geo_pit = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Pit', 'Yes')
```

Interior Geophysics Pit



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

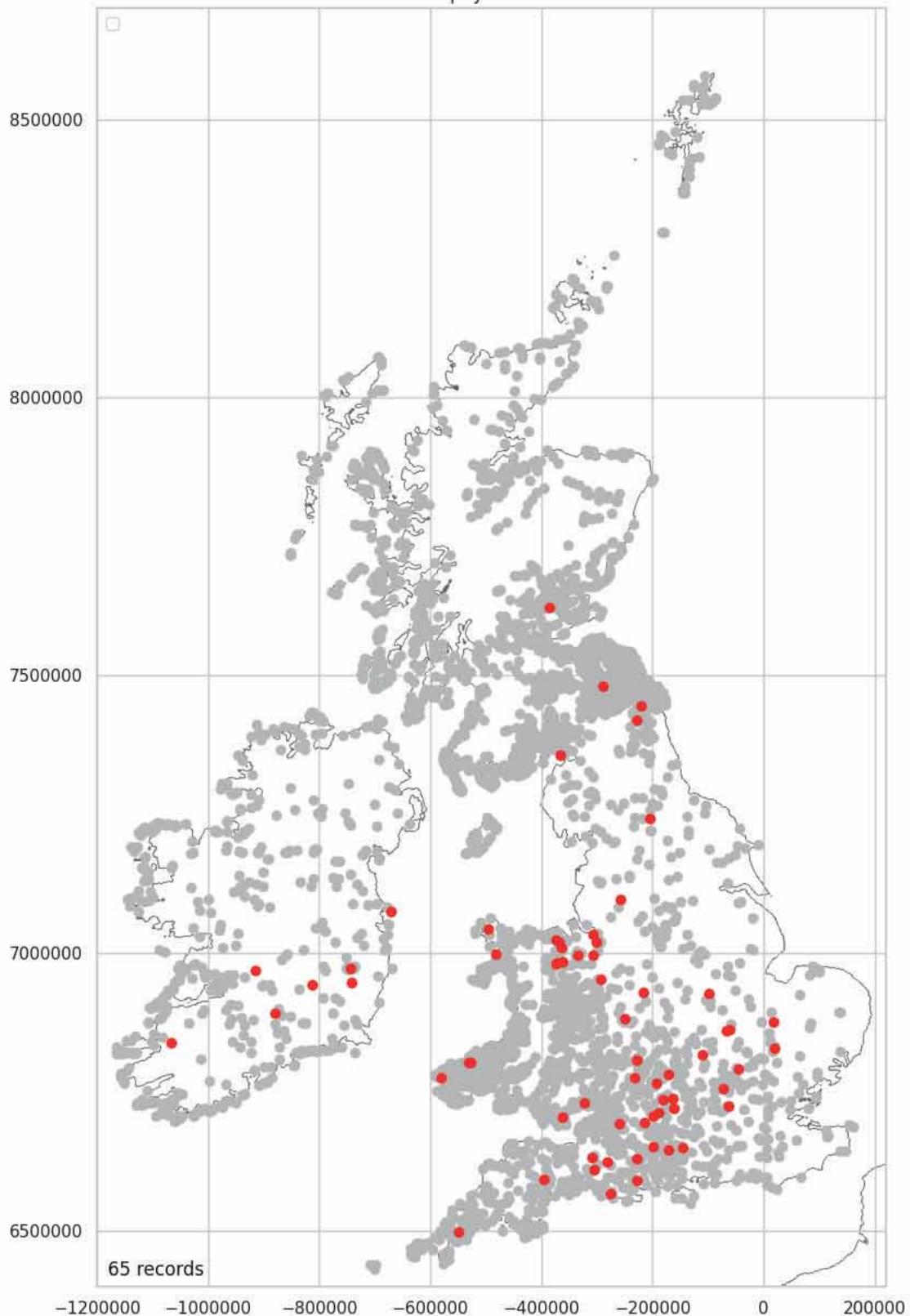
1.81%

Geophysics: Roundhouse Mapped

Roundhouses show the same bias as discussed in [Geophysics: None Mapped \(Surveyed\) Plus Oxford and Swindon Orbits](#). It is notable how different the distribution of roundhouses is in this small area to that discussed in [Excavation: Roundhouse Mapped](#).

```
In [165]: int_geo_rh = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Roundhouse', 'Yes')
```

Interior Geophysics Roundhouse



Middleton, M. 2024, Hillforts Primer

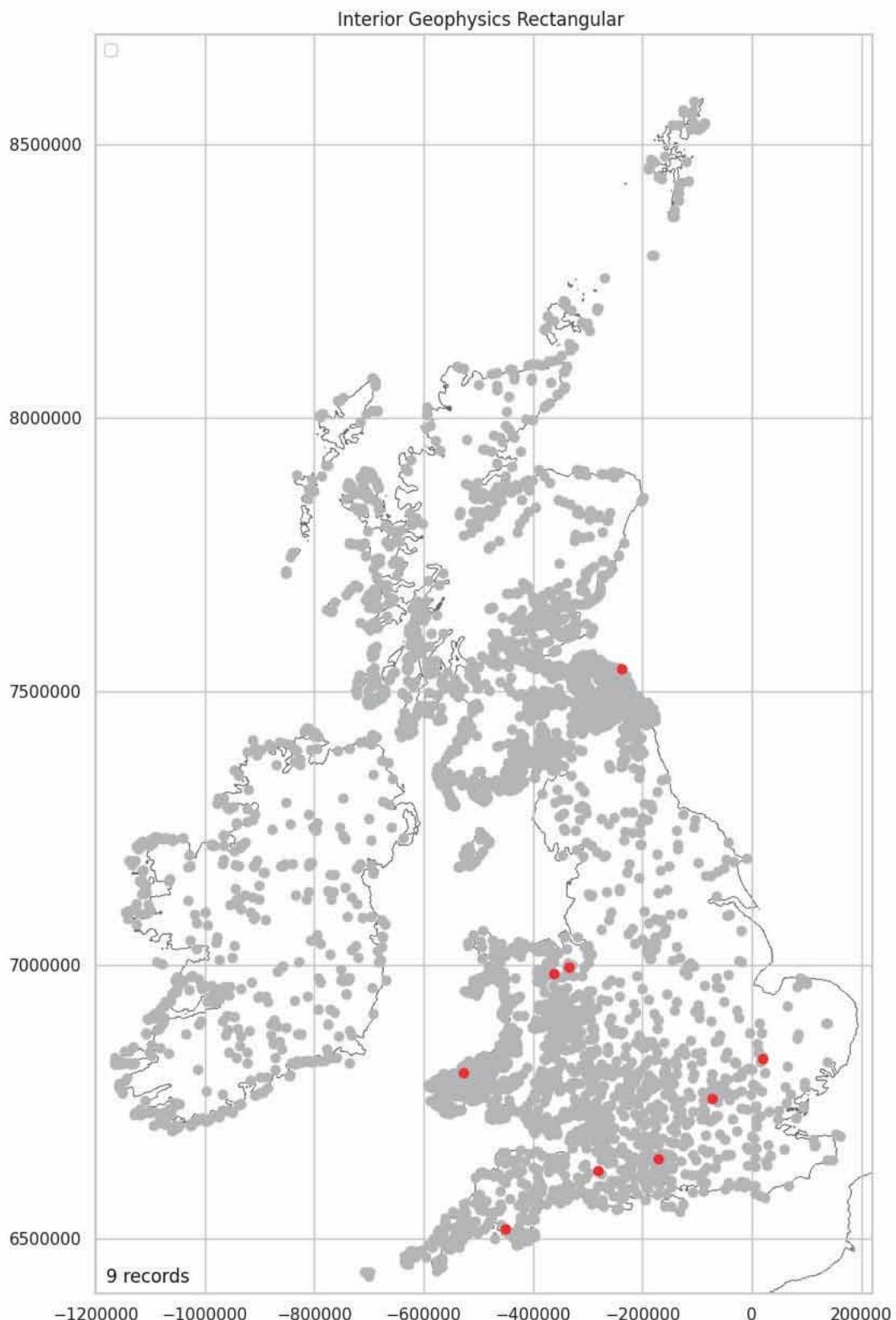
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 57%

Geophysics: Rectangular Mapped

Geophysics surveys have only identified rectangular structures in nine hillforts.

```
In [166]: int_geo_rect = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Rectangular', 'Yes')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

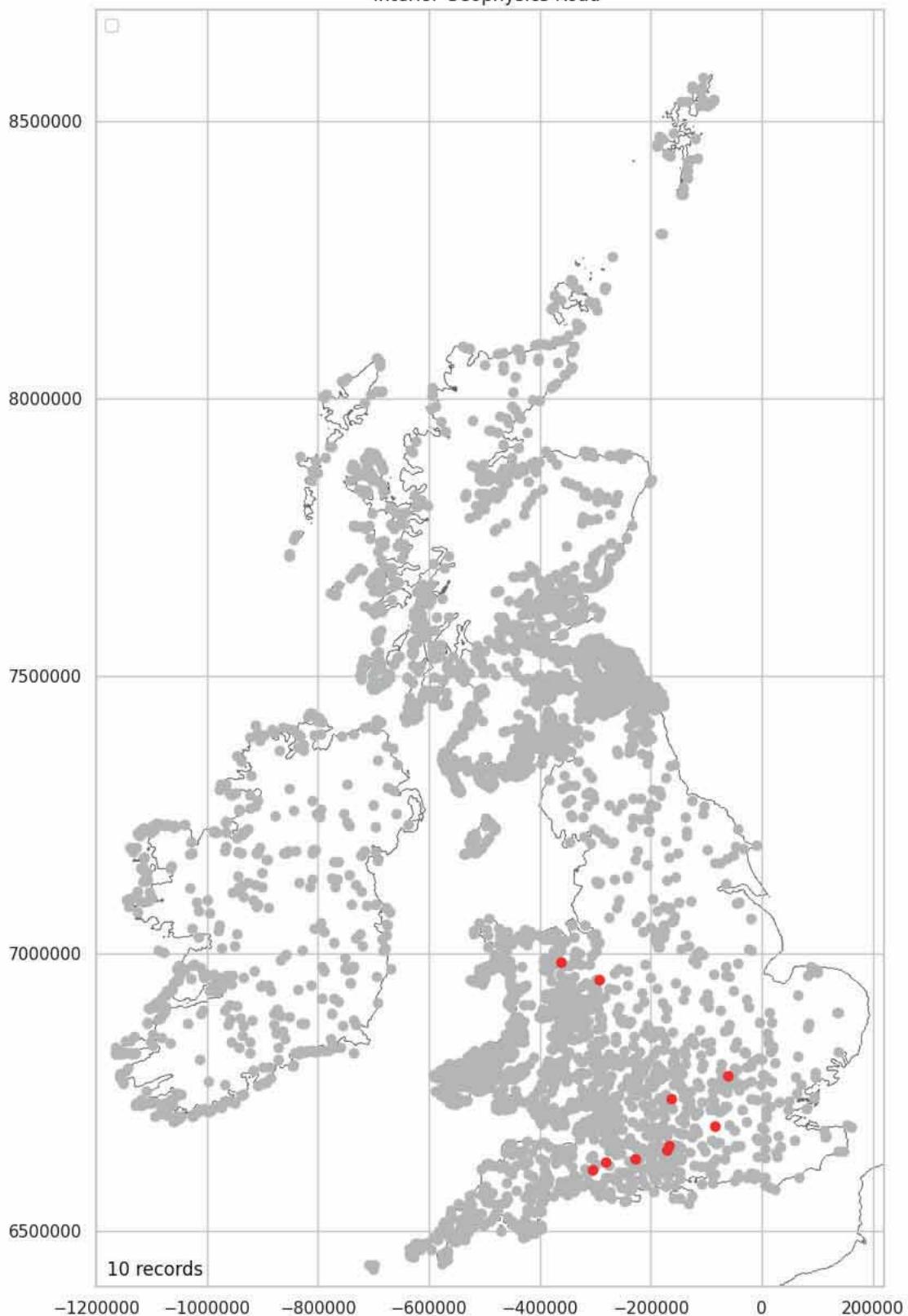
0.22%

Geophysics: Road Mapped

Geophysics surveys have only identified roads in ten hillforts.

```
In [167]: int_geo_road = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Road', 'Yes')
```

Interior Geophysics Road



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

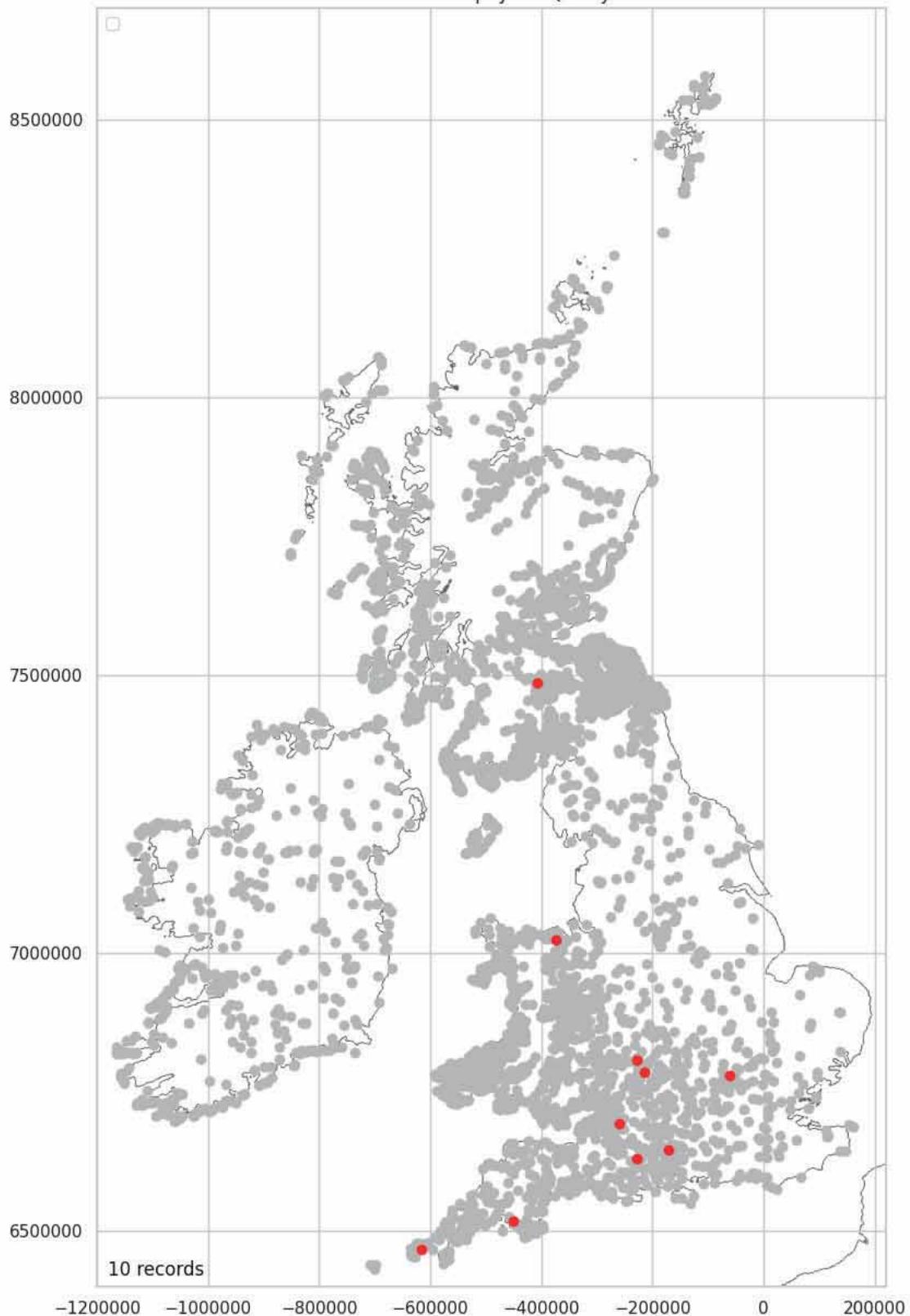
0.24%

Geophysics: Quarry Mapped

Geophysics surveys have only identified quarries in ten hillforts.

```
In [168]: int_geo_quarry = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Quarry', 'Yes')
```

Interior Geophysics Quarry



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

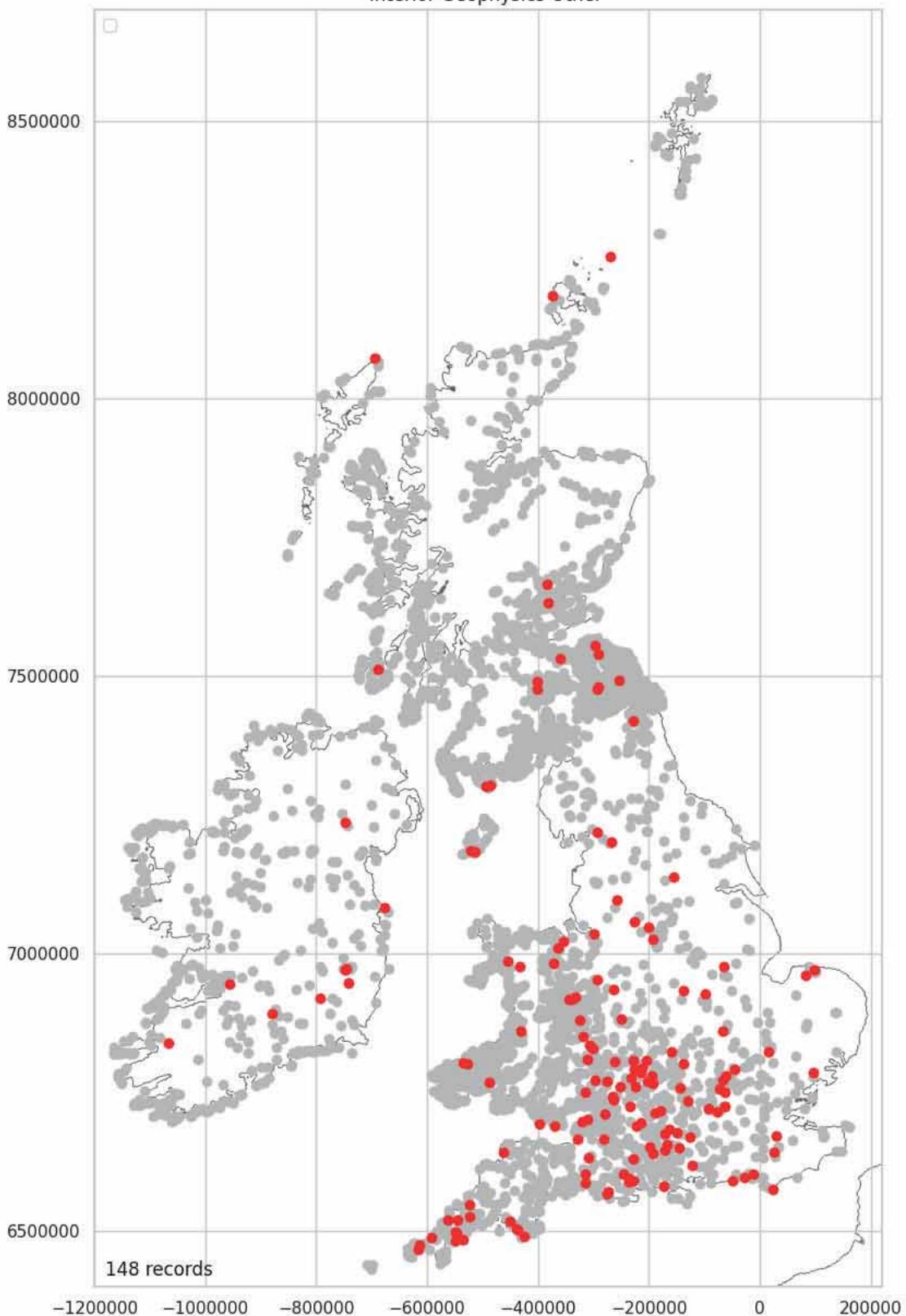
0.24%

Geophysics: Other Mapped

Other structures, identified in geophysical surveys, show the same bias as discussed in [Geophysics: None Mapped \(Surveyed\) Plus Oxford and Swindon Orbits](#).

```
In [169]: int_geo_other = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Other', 'Yes')
```

Interior Geophysics Other



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

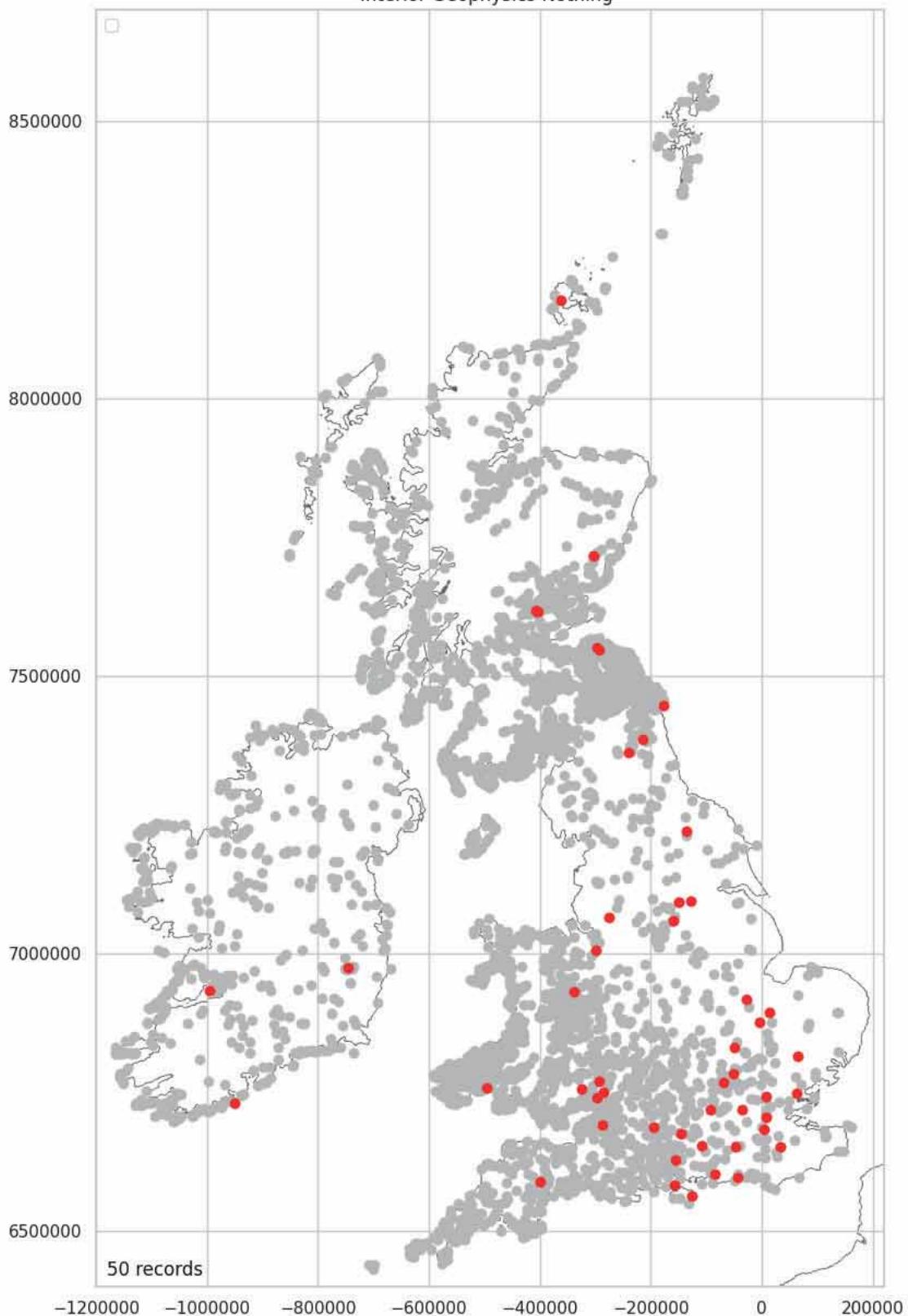
3.57%

Geophysics: Nothing Mapped

The distribution of hillforts, where nothing was recorded in geophysics surveys, is interesting in that most of the hillforts are located in the south east. This is interesting as it goes against what would be expected considering the bias discussed in [Geophysics: None Mapped \(Surveyed\) Plus Oxford and Swindon Orbits](#).

```
In [170]: int_geo_nothing = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Nothing', 'Yes')
```

Interior Geophysics Nothing



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 21%

Review Interior Data Split

```
In [171]: review_data_split(interior_data, interior_numeric_data, interior_text_data, interior_encodeable_data)  
Data split good.
```

Interior Data Package

Pre-processed interior data.

```
In [172...]: interior_data_list = [interior_numeric_data, interior_text_data, interior_encodeable_data]
```

Interior Data Download Package

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [173...]: download(interior_data_list, 'Interior_package')
```

Save Figure List

```
In [174...]: if save_images:
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")
    fig_list.to_csv(path, index=False)
```

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 5

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

- [Entrance Data](#)
- [Enclosing Data](#)
- [Annex Data](#)
- [Reference Data](#)
- [Acknowledgements](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:
<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [ ]: confirmed_only = False  
In [ ]: download_data = False  
In [ ]: save_images = False  
In [ ]: show_topography = False
```

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Review Data Part 5](#).

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>
Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer
Licence: <https://creativecommons.org/licenses/by-sa/4.0/>
Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>
Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>
Hillforts: Britain, Ireland and the Nearer Continent (Sample):
<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Reload Data and Python Functions

This study is split over multiple documents. Each file needs to be configured and have the source data imported. As this section does not focus on the assessment of the data it is minimised to facilitate the documents readability.

Python Modules and Code Setup

The Python imports enable the Hillforts Atlas data to be analysed and mapped within this document. The Python code can be run on demand, (see: [User Settings](#)). This means that as new research becomes available, the source for this document can be updated

to a revised copy of the Atlas data and the impact of that research can be reviewed using the same code and graphic output. The Hillforts Atlas is a baseline and this document is a tool that can be used to assess the impact new research is making in this area.

```
In [ ]: import sys
print(f'Python: {sys.version}')

import sklearn
print(f'Scikit-Learn: {sklearn.__version__}')

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

import os
import collections
import math
import random
import PIL
import urllib
random.seed(42)
# A random seed is used to ensure that the random numbers created are the
# same for each run of this document.

from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive
```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
SciKit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.25.2
matplotlib: 3.7.1
seaborn: 0.13.1
scipy: 1.11.4

Ref: <https://www.python.org/>
Ref: <https://scikit-learn.org/stable/>
Ref: <https://pandas.pydata.org/docs/>
Ref: <https://numpy.org/doc/stable/>
Ref: <https://matplotlib.org/>
Ref: <https://seaborn.pydata.org/>
Ref: <https://docs.scipy.org/doc/scipy/index.html>
Ref: <https://pypi.org/project/python-slugify/>

```
In [ ]: # # Ensure Python is ≥3.7
# import sys
# assert sys.version_info >= (3, 7)
# print(f'Python: {sys.version}')

# # Ensure Scikit-Learn is ≥1.0.2
# import sklearn
# assert sklearn.__version__ >= "1.0.2"
# print(f'Scikit-Learn: {sklearn.__version__}')

# # Ensure Pandas is ≥1.3.5
# import pandas as pd
# assert pd.__version__ >= "1.3.5"
# print(f'pandas: {pd.__version__}')
```

```

# # Ensure Numpy is ≥1.21.6
# import numpy as np
# assert np.__version__ >= "1.21.6"
# print(f'numpy: {np.__version__}')

# # Ensure matplotlib is ≥3.2.2
# %matplotlib inline
# import matplotlib
# assert matplotlib.__version__ >= "3.2.2"
# print(f'matplotlib: {matplotlib.__version__}')
# import matplotlib.pyplot as plt
# import matplotlib.cm as cm
# import matplotlib.patches as mpatches
# from matplotlib.cbook import boxplot_stats
# from matplotlib.lines import Line2D

# # Ensure Seaborn is ≥0.11.2
# import seaborn as sns
# assert sns.__version__ >= "0.11.2"
# print(f'seaborn: {sns.__version__}')
# sns.set(style="whitegrid")

# # Ensure Scipy is ≥1.4.1
# import scipy
# assert scipy.__version__ >= "1.4.1"
# print(f'scipy: {scipy.__version__}')
# from scipy import stats
# from scipy.stats import gaussian_kde

# # Import Python Libraries
# import os
# import collections
# from slugify import slugify

# # Import Google colab tools to access Drive
# from google.colab import drive

```

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

```
In [ ]: def show_records(plt, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plt.annotate(str(len(plot_data))+' records', xy=(-1180000, 6420000), \
                xycoords='data', ha='left', color=text_colour)
```

```
In [ ]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
                      "hillforts-topo-northwest-minus.png",
                      "hillforts-topo-northeast.png",
                      "hillforts-topo-south.png",
                      "hillforts-topo-south-plus.png",
                      "hillforts-topo-ireland.png",
                      "hillforts-topo-ireland-north.png",
                      "hillforts-topo-ireland-south.png"]
    else:
        backgrounds = ["hillforts-outline-01.png",
                      "hillforts-outline-north.png",
                      "hillforts-outline-northwest-plus.png",
                      "hillforts-outline-northwest-minus.png",
                      "hillforts-outline-northeast.png",
                      "hillforts-outline-south.png",
                      "hillforts-outline-south-plus.png",
                      "hillforts-outline-ireland.png",
                      "hillforts-outline-ireland-north.png",
                      "hillforts-outline-ireland-south.png"]
    return backgrounds
```

```
In [ ]: def get_bounds():
    bounds = [[-1200000, 220000, 6400000, 8700000],
              [-1200000, 220000, 7000000, 8700000],
              [-1200000, -480000, 7000000, 8200000],
              [-900000, -480000, 7100000, 8200000],
              [-520000, 0, 7000000, 8700000],
              [-800000, 220000, 6400000, 7100000],
              [-1200000, 220000, 6400000, 7100000],
              [-1200000, -600000, 6650000, 7450000],
              [-1200000, -600000, 7050000, 7450000],
```

```

[-1200000, -600000, 6650000, 7080000]]
return bounds

In [ ]: def show_background(plt, ax, location=""):
backgrounds = get_backgrounds()
bounds = get_bounds()
folder = "https://raw.githubusercontent.com/MikeDai/rise/Hillforts-Primer/main/hillforts-topo/"
#"https://raw.githubusercontent.com/MikeDai/rise/Hillforts-Primer/main/
#hillforts-topo/"

if location == "n":
    background = os.path.join(folder, backgrounds[1])
    bounds = bounds[1]
elif location == "nw+":
    background = os.path.join(folder, backgrounds[2])
    bounds = bounds[2]
elif location == "nw-":
    background = os.path.join(folder, backgrounds[3])
    bounds = bounds[3]
elif location == "ne":
    background = os.path.join(folder, backgrounds[4])
    bounds = bounds[4]
elif location == "s":
    background = os.path.join(folder, backgrounds[5])
    bounds = bounds[5]
elif location == "s+":
    background = os.path.join(folder, backgrounds[6])
    bounds = bounds[6]
elif location == "i":
    background = os.path.join(folder, backgrounds[7])
    bounds = bounds[7]
elif location == "in":
    background = os.path.join(folder, backgrounds[8])
    bounds = bounds[8]
elif location == "is":
    background = os.path.join(folder, backgrounds[9])
    bounds = bounds[9]
else:
    background = os.path.join(folder, backgrounds[0])
    bounds = bounds[0]

img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
ax.imshow(img, extent=bounds)

```

```

In [ ]: def get_counts(data):
data_counts = []
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    data_counts.append(count)
return data_counts

```

```

In [ ]: def add_annotation_plot(ax):
ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
            color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
            horizontalalignment = 'left')
ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
            size='small', color='grey', xy=(0.99, 0.01), \
            xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [ ]: def add_annotation_l_xy(ax):
ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
            color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
            horizontalalignment = 'left')
ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
            size='small', color='grey', xy=(0.99, 0.035), \
            xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [ ]: def plot_bar_chart(data, split_pos, x_label, y_label, title, clip=False):
fig = plt.figure(figsize=(12,5))
ax = fig.add_axes([0,0,1,1])
x_data = data.columns
x_data = [x.split("_")[split_pos:] for x in x_data]
x_data_new = []
for l in x_data:
    txt = ""
    for part in l:
        txt += "_" + part
    x_data_new.append(txt[1:])
if clip:
    x_data_new = x_data_new[:-1]
    new_data = data.copy()
    data = new_data.drop(['Dating_Date_Unknown'], axis=1)
y_data = get_counts(data)
ax.bar(x_data_new, y_data)

```

```

    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, \
                                n_bins, extra=''):
    new_data = data.copy()
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    data[x_label].plot(kind='hist', bins=n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    title = f'{title} {extra}'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def get_bins(data, bins_count):
    data_range = data.max() - data.min()
    print(bins_count)
    if bins_count != None:
        x_bins = [x for x in range(data.min(), data.max(), bins_count)]
        n_bins = len(x_bins)
    else:
        n_bins = int(data_range)
        if n_bins < 10:
            multi = 10
            while n_bins < 10:
                multi *= 10
                n_bins = int(data_range * multi)
        elif n_bins > 100:
            n_bins = int(data_range)/10
    return n_bins

```

```

In [ ]: def plot_histogram(data, x_label, title, bins_count = None):
    n_bins = get_bins(data, bins_count)
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    ax.set_ylabel('Count')
    plt.ticker._format(style='plain')
    plt.hist(data, bins=n_bins)
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_continuous(data, x_label, title):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    plt.plot(data, linewidth=4)
    plt.ticker._format(style='plain')
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.ticker._format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v", whis=[2.2, 97.8])

```

```

else:
    sns.boxplot(data=data, orient="h", whis=[2.2, 97.8])
    save_fig(feature + " Range")
    plt.show()

bp = boxplot_stats(data, whis=[2.2, 97.8])

low = bp[0].get('whislo')
q1 = bp[0].get('q1')
median = bp[0].get('med')
q3 = bp[0].get('q3')
high = bp[0].get('whishi')

return [low, q1, median, q3, high]

```

```

In [ ]: def plot_data_range_plus(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range (Outlier Steps)"))
    plt.ticklabel_format(style='plain')

    if o == "v":
        sns.boxplot(data=data, orient="v", whis=[2.2, 97.8])
    else:
        sns.boxplot(data=data, orient="h", whis=[2.2, 97.8])

    # Add annotation lines
    x = [24, 24, 54, 54]
    y = [-0.05, -0.075, -0.075, -0.05]
    x1 = [54, 54, 84, 84]
    y1 = [-0.1, -0.125, -0.125, -0.1]
    x2 = [84, 84, 114, 114]
    y2 = [-0.05, -0.075, -0.075, -0.05]

    line_1 = plt.plot(x, y)
    line_2 = plt.plot(x1, y1)
    line_3 = plt.plot(x2, y2)

    # Add annotation text
    text_kwargs = dict(ha='center', va='center', fontsize=16, color='k')
    plt.text(39, -0.1, '30 Ha', **text_kwargs)
    plt.text(69, -0.1, '30 Ha', **text_kwargs)
    plt.text(99, -0.1, '30 Ha', **text_kwargs)

    save_fig(feature + " Range")
    plt.show()

    return

```

```

In [ ]: def location_XY_plot():
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 8700000)
    add_annotation_l_xy(plt)

```

```

In [ ]: def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')

```

```

In [ ]: def plot_over_grey_numeric(merged_data, a_type, title, extra="", \
                                inner=False, fringe=False, \
                                oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()

```

```

        patches.append(i_for_legend)
show_records(plt, plot_data)
plt.legend(loc='upper left', handles=patches)
plt.title(get_print_title(title))
save_fig(title)
plt.show()
print(f' {round(((len(plot_data)/4147)*100), 2)}%')

```

```

In [ ]: def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    fig, ax = plt.subplots(figsize=(9.47, 15.33))
    show_background(plt, ax)
    location_XY_plot()
    add_grey(region='')
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    show_records(plt, plot_data)
    plt.title(get_print_title('Boundary_Type: ' + boundary_type))
    save_fig('Boundary_Type_' + boundary_type)
    plt.show()

```

```

In [ ]: def add_21Ha_line():
    x_values = \
        [-367969, -344171, -263690, -194654, -130542, -119597, -162994, -265052], -304545]
    y_values = \
        [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 6663609], 6611780]
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, label =
        '\u2265 21 Ha Line')
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.25, label =
        '\u2265 21 Ha Line')
    return add_to_legend

```

```

In [ ]: def add_21Ha_fringe():
    x_values = \
        [-367969, -126771, 29679, -42657, -248650, -304545, -423647, -584307, -367969]
    y_values = \
        [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, 6752378, 7019842]
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, label =
        '\u2265 21 Ha Fringe')
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, \
        label = '\u2265 21 Ha Fringe')
    return add_to_legend

```

```

In [ ]: def plot_density_over_grey(data, data_type, extra='', inner=False, fringe=False):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_density(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
        c=new_data['Density'], cmap=cm.rainbow, s=25)
    if fringe:
        add_21Ha_fringe()
    if inner:
        add_21Ha_line()
        plt.legend(loc='lower left')
    plt.colorbar(label='Density')
    title = f'Density - {data_type} {extra}'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_density_over_grey_three(data_low, data_iqr, data_high, title, \
                                         extra='', inner=False, fringe=False):
    new_data_low = data_low.copy()
    new_data_low = new_data_low.drop(['Density'], axis=1)
    new_data_low = add_density(new_data_low)

    new_data_iqr = data_iqr.copy()
    new_data_iqr = new_data_iqr.drop(['Density'], axis=1)
    new_data_iqr = add_density(new_data_iqr)

    new_data_high = data_high.copy()
    new_data_high = new_data_high.drop(['Density'], axis=1)
    new_data_high = add_density(new_data_high)

    fig, ax = plt.subplots(1, 3)
    fig.set_figheight(7)
    fig.set_figwidth(15)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDai/rsie/Hillforts-Primer/main/hillforts-topo/"
    #"https://raw.githubusercontent.com/MikeDai/rsie/Hillforts-Primer/main/

```

```

#hillforts-topo/
background = os.path.join(fol der, "hillforts-bw-02.png")
bounds = bounds[0]
img = np.array(PIL.Image.open(url lib.request.urlopen(background)))
ax[0].imshow(img, extent=bounds)
ax[1].imshow(img, extent=bounds)
ax[2].imshow(img, extent=bounds)

ax[0].scatter(new_data_low['Location_X'], new_data_low['Location_Y'], \
              c=new_data_low['Density'], cmap=cm.rainbow, s=25)
ax[1].scatter(new_data_iqr['Location_X'], new_data_iqr['Location_Y'], \
              c=new_data_iqr['Density'], cmap=cm.rainbow, s=25)
ax[2].scatter(new_data_high['Location_X'], new_data_high['Location_Y'], \
              c=new_data_high['Density'], cmap=cm.rainbow, s=25)

ax[0].get_yaxis().set_visible(False)
ax[1].get_yaxis().set_visible(False)
ax[2].get_yaxis().set_visible(False)

ax[0].get_xaxis().set_visible(False)
ax[1].get_xaxis().set_visible(False)
ax[2].get_xaxis().set_visible(False)

ax[0].set_title("1st Quarter (Tiny Hillforts)")
ax[1].set_title("IQR (Small to Medium Hillforts)")
ax[2].set_title("4th Quarter (Large Hillforts)")

fig.suptitle(get_print_title(title), y=1.08)
ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
               color='grey', xy=(0, -0.1), xycoords='axes fraction', \
               horizontalalignment = 'left')
ax[2].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(1, -0.1), \
               xycoords='axes fraction', horizontalalignment = 'right')
save_fig(title)
plt.show()

```

```

In [ ]: def plot_densiti_over_grey_four(data_1, data_2, data_3, data_4, \
                                         title, extra='', inner=False, fringe=False):
    new_data_1 = data_1.copy()
    new_data_1 = new_data_1.drop(['Density'], axis=1)
    new_data_1 = add_densiti(new_data_1)

    new_data_2 = data_2.copy()
    new_data_2 = new_data_2.drop(['Density'], axis=1)
    new_data_2 = add_densiti(new_data_2)

    new_data_3 = data_3.copy()
    new_data_3 = new_data_3.drop(['Density'], axis=1)
    new_data_3 = add_densiti(new_data_3)

    new_data_4 = data_4.copy()
    new_data_4 = new_data_4.drop(['Density'], axis=1)
    new_data_4 = add_densiti(new_data_4)

    fig, ax = plt.subplots(1, 4)
    fig.set_fiheight(7)
    fig.set_fiwidth(20)

    bounds = get_bounds()
    fol der = "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo/" \
    "#https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/" \
    "#hillforts-topo/"
    background = os.path.join(fol der, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = np.array(PIL.Image.open(url lib.request.urlopen(background)))
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)
    ax[3].imshow(img, extent=bounds)

    ax[0].scatter(new_data_1['Location_X'], new_data_1['Location_Y'], \
                  c=new_data_1['Density'], cmap=cm.rainbow, s=25)
    ax[1].scatter(new_data_2['Location_X'], new_data_2['Location_Y'], \
                  c=new_data_2['Density'], cmap=cm.rainbow, s=25)
    ax[2].scatter(new_data_3['Location_X'], new_data_3['Location_Y'], \
                  c=new_data_3['Density'], cmap=cm.rainbow, s=25)
    ax[3].scatter(new_data_4['Location_X'], new_data_4['Location_Y'], \
                  c=new_data_4['Density'], cmap=cm.rainbow, s=25)

    ax[0].get_yaxis().set_visible(False)
    ax[1].get_yaxis().set_visible(False)
    ax[2].get_yaxis().set_visible(False)
    ax[3].get_yaxis().set_visible(False)

```

```

ax[0].get_xaxis().set_visible(False)
ax[1].get_xaxis().set_visible(False)
ax[2].get_xaxis().set_visible(False)
ax[3].get_xaxis().set_visible(False)

ax[0].set_title("NE")
ax[1].set_title("SE")
ax[2].set_title("SW")
ax[3].set_title("NW")

fig.suptitle(get_pint_title(title), y=1.08)
ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
    color='grey', xy=(0, -0.1), xycoords='axes fraction', \
    horizontalalignment = 'left')
ax[3].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
    size='small', color='grey', xy=(1, -0.1), \
    xycoords='axes fraction', horizontalalignment = 'right')
save_fig(title)
plt.show()

```

```

In [ ]: def plot_density_over_grey_five(data_1, data_2, data_3, data_4, data_5, title, \
    extra='', inner=False, fringe=False):
    new_data_1 = data_1.copy()
    new_data_1 = new_data_1.drop(['Density'], axis=1)
    new_data_1 = add_density(new_data_1)

    new_data_2 = data_2.copy()
    new_data_2 = new_data_2.drop(['Density'], axis=1)
    new_data_2 = add_density(new_data_2)

    new_data_3 = data_3.copy()
    new_data_3 = new_data_3.drop(['Density'], axis=1)
    new_data_3 = add_density(new_data_3)

    new_data_4 = data_4.copy()
    new_data_4 = new_data_4.drop(['Density'], axis=1)
    new_data_4 = add_density(new_data_4)

    new_data_5 = data_5.copy()
    new_data_5 = new_data_5.drop(['Density'], axis=1)
    new_data_5 = add_density(new_data_5)

    fig, ax = plt.subplots(1, 5)
    fig.set_figheight(7)
    fig.set_figwidth(24)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-topo"
    background = os.path.join(folder, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)
    ax[3].imshow(img, extent=bounds)
    ax[4].imshow(img, extent=bounds)

    ax[0].scatter(new_data_1['Location_X'], new_data_1['Location_Y'], \
        c=new_data_1['Density'], cmap=cm.rainbow, s=25)
    ax[1].scatter(new_data_2['Location_X'], new_data_2['Location_Y'], \
        c=new_data_2['Density'], cmap=cm.rainbow, s=25)
    ax[2].scatter(new_data_3['Location_X'], new_data_3['Location_Y'], \
        c=new_data_3['Density'], cmap=cm.rainbow, s=25)
    ax[3].scatter(new_data_4['Location_X'], new_data_4['Location_Y'], \
        c=new_data_4['Density'], cmap=cm.rainbow, s=25)
    ax[4].scatter(new_data_5['Location_X'], new_data_5['Location_Y'], \
        c=new_data_5['Density'], cmap=cm.rainbow, s=25)

    ax[0].get_yaxis().set_visible(False)
    ax[1].get_yaxis().set_visible(False)
    ax[2].get_yaxis().set_visible(False)
    ax[3].get_yaxis().set_visible(False)
    ax[4].get_yaxis().set_visible(False)

    ax[0].get_xaxis().set_visible(False)
    ax[1].get_xaxis().set_visible(False)
    ax[2].get_xaxis().set_visible(False)
    ax[3].get_xaxis().set_visible(False)
    ax[4].get_xaxis().set_visible(False)

    ax[0].set_title("0")
    ax[1].set_title("1")
    ax[2].set_title("2")

```

```

ax[3].set_title("3")
ax[4].set_title("4")

fig.suptitle(get_print_title(title), y=1.08)
ax[0].annotate("Middleton, M. 2024. Hillforts Primer", size='small', \
               color='grey', xy=(0, -0.1), xycoords='axes fraction', \
               horizontalalignment = 'left')
ax[4].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(1, -0.1), \
               xycoords='axes fraction', horizontalalignment = 'right')
save_fig(title)
plt.show()

```

```

In [ ]: def plot_densiti_over_grey_six(data_1, data_2, data_3, data_4, data_5, data_6, \
                                      title, extra='', inner=False, fringe=False):
    new_data_1 = data_1.copy()
    new_data_1 = new_data_1.drop(['Density'], axis=1)
    new_data_1 = add_density(new_data_1)

    new_data_2 = data_2.copy()
    new_data_2 = new_data_2.drop(['Density'], axis=1)
    new_data_2 = add_density(new_data_2)

    new_data_3 = data_3.copy()
    new_data_3 = new_data_3.drop(['Density'], axis=1)
    new_data_3 = add_density(new_data_3)

    new_data_4 = data_4.copy()
    new_data_4 = new_data_4.drop(['Density'], axis=1)
    new_data_4 = add_density(new_data_4)

    new_data_5 = data_5.copy()
    new_data_5 = new_data_5.drop(['Density'], axis=1)
    new_data_5 = add_density(new_data_5)

    new_data_6 = data_6.copy()
    new_data_6 = new_data_6.drop(['Density'], axis=1)
    new_data_6 = add_density(new_data_6)

    fig, ax = plt.subplots(1, 6)
    fig.set_figheight(6)
    fig.set_figwidth(24)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDaiRsi/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDaiRsi/Hillforts-Primer/main/hillforts-topo/"
    background = os.path.join(folder, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)
    ax[3].imshow(img, extent=bounds)
    ax[4].imshow(img, extent=bounds)
    ax[5].imshow(img, extent=bounds)

    ax[0].scatter(new_data_1['Location_X'], new_data_1['Location_Y'], \
                  c=new_data_1['Density'], cmap=cm.rainbow, s=25)
    ax[1].scatter(new_data_2['Location_X'], new_data_2['Location_Y'], \
                  c=new_data_2['Density'], cmap=cm.rainbow, s=25)
    ax[2].scatter(new_data_3['Location_X'], new_data_3['Location_Y'], \
                  c=new_data_3['Density'], cmap=cm.rainbow, s=25)
    ax[3].scatter(new_data_4['Location_X'], new_data_4['Location_Y'], \
                  c=new_data_4['Density'], cmap=cm.rainbow, s=25)
    ax[4].scatter(new_data_5['Location_X'], new_data_5['Location_Y'], \
                  c=new_data_5['Density'], cmap=cm.rainbow, s=25)
    ax[5].scatter(new_data_5['Location_X'], new_data_5['Location_Y'], \
                  c=new_data_5['Density'], cmap=cm.rainbow, s=25)

    ax[0].get_yaxis().set_visible(False)
    ax[1].get_yaxis().set_visible(False)
    ax[2].get_yaxis().set_visible(False)
    ax[3].get_yaxis().set_visible(False)
    ax[4].get_yaxis().set_visible(False)
    ax[5].get_yaxis().set_visible(False)

    ax[0].get_xaxis().set_visible(False)
    ax[1].get_xaxis().set_visible(False)
    ax[2].get_xaxis().set_visible(False)
    ax[3].get_xaxis().set_visible(False)
    ax[4].get_xaxis().set_visible(False)
    ax[5].get_xaxis().set_visible(False)

    ax[0].set_title("Part Univalate")

```

```

ax[1].set_title("Univallate")
ax[2].set_title("Part Bivallate")
ax[3].set_title("Bivallate")
ax[4].set_title("Part Multivallate")
ax[5].set_title("Multivallate")

fig.suptitle(get_print_title(title), y=1.08)
ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
    color='grey', xy=(0, -0.1), xycoords='axes fraction', \
    horizontalalignment = 'left')
ax[5].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
    size='small', color='grey', xy=(1, -0.1), \
    xycoords='axes fraction', horizontalalignment = 'right')
save_fig(title)
plt.show()

```

```

In [ ]: def add_oxford_swindon(oxford=False, swindon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_size = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_size, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swindon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_size, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches

```

```

In [ ]: def plot_over_grey(merged_data, a_type, yes_no, extra="", inner=False, \
    fringe=False, oxford=False, swindon=False, topo=False):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()
    print(f'{round(((len(plot_data)/4147)*100), 2)}%')
    return plot_data

```

```

In [ ]: def plot_type_values(data, data_type, title, extra=''):
    new_data = data.copy()
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
        c=new_data[data_type], cmap=cm.rainbow, s=25)
    plt.colorbar(label=data_type)
    title = f'{data_type} {extra}'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def bespoke_plot(plt, title):
    add_annotation_plot(plt)
    plt.ticker.label_format(style='plain')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [ ]: def add_cluster_split_lines(plt, ax, extra=None):
    x_min = -550000
    if extra == 'ireland':
        x_min = -1200000
    plt.vlines(x=[-500000], ymin=7070000, ymax=9000000, colors='r', \
        ls='-', lw=3)
    plt.hlines(y=[7070000], xmin=x_min, xmax=200000, colors='r', \
        ls='-', lw=3)

```

```

    ax.annotate("N/S split", color='k', xy=(50000, 7090000), xycoords='data')
    ax.annotate("E/W split", color='k', xy=(-480000, 8660000), xycoords='data')
    ax.annotate("Irish Sea split", color='k', xy=(-1150000, 7510000), \
                xycoords='data')
    plot_line((-800000, 6400000), (-550000, 7070000))
    plot_line((-550000, 7070000), (-666000, 7440000))
    plot_line((-666000, 7440000), (-900000, 7500000))

```

```

In [ ]: def plot_values(data, feature, title, extra=''):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(data['Location_X'], data['Location_Y'], c=data[feature], \
                cmap=cm.rainbow, s=25)
    plt.colorbar(label=feature)
    title = f'{title} {extra}'
    plt.title(title)
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_line(point1, point2):
    x_values = [point1[0], point2[0]]
    y_values = [point1[1], point2[1]]
    plt.plot(x_values, y_values, 'r', ls='0', lw=2, alpha=1)

```

```

In [ ]: def density_scatter_lines(location_data, scatter_data, plot_title, \
                                inner=False, fringe=False):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(location_data['Location_X'], location_data['Location_Y'], \
                c=location_data['Density_trans'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density Transformed')
    if inner:
        add_21Ha_line()
    if fringe:
        add_21Ha_fringe()
    plt.scatter(scatter_data['Location_X'], scatter_data['Location_Y'], c='Red')
    plt.legend(loc='lower left')
    plt.title(get_print_title(plot_title))
    save_fig(plot_title)
    plt.show()

```

```

In [ ]: def south_density_scatter_lines(location_data, scatter_data, plot_title, \
                                       inner=False, fringe=False):
    fig, ax = plt.subplots(figsize=((6.73*1.5)+2.0, (4.62*1.5)))
    show_background(plt, ax, 's')
    plt.ticklabel_format(style='plain')
    plt.xlim(-800000, 220000)
    plt.ylim(6400000, 7100000)
    plt.scatter(location_data['Location_X'], location_data['Location_Y'], \
                c=location_data['Density_trans'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density Transformed')
    if inner:
        add_21Ha_line()
    if fringe:
        add_21Ha_fringe()
    plt.scatter(scatter_data['Location_X'], scatter_data['Location_Y'], \
                c='red', s=60)
    add_annotation_plot(plt)
    plt.legend(loc='lower right')
    plt.title(get_print_title(plot_title))
    save_fig(plot_title)
    plt.show()

```

```

In [ ]: def add_linear_south():
    x_values = [-115637, -286900]
    y_values = [6678188, 6585812]
    xx_values = [-244249, -363049]
    yy_values = [6555133, 6589612]
    xxx_values = [-392213, -363146]
    yyy_values = [6577365, 6647256]
    x4_values = [-169664, -207084]
    y4_values = [6599254, 6615290]
    x5_values = [-238560, -200891]
    y5_values = [6668083, 6637826]

    plt.plot(x_values, y_values, 'g', ls='-', lw=8, alpha=0.6, label = \
              'Poss. correlation to linear routes?')
    plt.plot(xx_values, yy_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(xxx_values, yyy_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(x4_values, y4_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(x5_values, y5_values, 'g', ls='-', lw=8, alpha=0.6)

```

```
In [ ]: def plot_over_grey_south(merged_data, a_type, yes_no, extra=""):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(1, figsize=((6.73*1.5), (4.62*1.5)))
    show_background(plt, ax, 's')
    plt.ticker._format(style='plain')
    plt.xlim(-800000, 220000)
    plt.ylim(6400000, 7100000)
    add_annotation_l_xy(plt)
    add_grey('s')
    add_l_iner_south()
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    plt.legend(loc='lower right')
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()
    return plot_data
```

```
In [ ]: def plot_over_grey_north(merged_data, a_type, yes_no, extra="", anno=False):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(1, figsize=((5.28*2), (5.28*2)))
    show_background(plt, ax, 'n')
    plt.ticker._format(style='plain')
    plt.xlim(-800000, 0)
    plt.ylim(7200000, 8000000)
    if anno == 'Stirling':
        plt.annotate('SC1514: Gillies Hill', xy=(-443187, 7578896), \
                     xycoords='data', ha='left', xytext=(-135, 110), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
        plt.annotate('SC3420: Morebattle Hill', xy=(-263209, 7461125), \
                     xycoords='data', ha='left', xytext=(-90, -100), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
        plt.annotate('EN4374: Pike House Camp', xy=(-209365, 7418590), \
                     xycoords='data', ha='left', xytext=(-10, 80), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
        plt.annotate('SC3900: Kilmurdie', xy=(-305133, 7566913), \
                     xycoords='data', ha='left', xytext=(20, 50), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
    elif anno == 'Traprain':
        plt.annotate('SC3932: Traprain Law', xy=(-297708, 7551155), \
                     xycoords='data', ha='left', xytext=(35, 80), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
        plt.annotate('SC3037: Law Hill', xy=(-372530, 7642082), \
                     xycoords='data', ha='left', xytext=(-150, 12), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
        plt.annotate("SC3571: Kerr's Knowe", xy=(-367362, 7485652), \
                     xycoords='data', ha='left', xytext=(-200, 0), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
        plt.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), \
                     xycoords='data', ha='left', xytext=(35, 35), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
    elif anno == 'Kerr':
        plt.annotate("SC3571: Kerr's Knowe", xy=(-367362, 7485652), \
                     xycoords='data', ha='left', xytext=(-200, 0), \
                     textcoords='offset points', \
                     arrowprops=dict(arrowstyle="->", color='k', lw=1, \
                                     connectionstyle="arc3,rad=-0.2"))
    add_annotation_l_xy(plt)
    add_grey('ne')
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()
    return plot_data
```

```
In [ ]: def get_proportions(date_set):
    total = sum(date_set) - date_set[-1]
    newset = []
    for entry in date_set[:-1]:
        newset.append(round(entry/total, 2))
    return newset

In [ ]: def plot_dates_by_region(nw, ne, ni, si, s, features):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = nw[features].columns
    x_data = [x.split("_")[-2] for x in x_data[:-1]]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])

    set1_name = 'NW'
    set2_name = 'NE'
    set3_name = 'N Ireland'
    set4_name = 'S Ireland'
    set5_name = 'South'
    set1 = get_proportions(get_counts(nw[features]))
    set2 = get_proportions(get_counts(ne[features]))
    set3 = get_proportions(get_counts(ni[features]))
    set4 = get_proportions(get_counts(si[features]))
    set5 = get_proportions(get_counts(s[features]))

    X_axis = np.arange(len(x_data_new))

    budge = 0.25

    plt.bar(X_axis - 0.55 + budge, set1, 0.3, label = set1_name)
    plt.bar(X_axis - 0.4 + budge, set2, 0.3, label = set2_name)
    plt.bar(X_axis - 0.25 + budge, set3, 0.3, label = set3_name)
    plt.bar(X_axis - 0.1 + budge, set4, 0.3, label = set4_name)
    plt.bar(X_axis + 0.05 + budge, set5, 0.3, label = set5_name)

    plt.xticks(X_axis, x_data_new)
    plt.xlabel('Dating')
    plt.ylabel('Proportion of Total Dated Hillforts in Region')
    title = 'Proportions of Dated Hillforts by Region'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [ ]: def get_pcent_list(old_list):
    pcnt_list = []
    total = sum(old_list)
    for item in old_list:
        pcnt_list.append(round(item/total, 2))
    return pcnt_list
```

```
In [ ]: def order_set(set_list, x_data, pcnt=False):
    new_list = []
    set_values = set_list.index.tolist()
    for val in x_data:
        if val in set_values:
            new_list.append(set_list.loc[[val]].values[0])
        else:
            new_list.append(0)
    if pcnt:
        new_list = get_pcent_list(new_list)
    return new_list
```

```
In [ ]: def plot_feature_by_region(nw, ne, ni, si, s, feature, title, clip):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    max_val = \
        int(max([nw[feature].max(), ne[feature].max(), \
                 ni[feature].max(), si[feature].max(), s[feature].max()])) + 2

    x_data = [x-1 for x in range(max_val+2)]

    set0_name = 'NW'
    set1_name = 'NE'
    set2_name = 'N Ireland'
    set3_name = 'S Ireland'
    set4_name = 'S'
```

```

set0 = nw[feature].value_counts()
set1 = ne[feature].value_counts()
set2 = ni[feature].value_counts()
set3 = si[feature].value_counts()
set4 = s[feature].value_counts()

set0 = order_set(set0, x_data, True)[:clip]
set1 = order_set(set1, x_data, True)[:clip]
set2 = order_set(set2, x_data, True)[:clip]
set3 = order_set(set3, x_data, True)[:clip]
set4 = order_set(set4, x_data, True)[:clip]

X_axis_s = np.arange(len(x_data[:clip]))

budge = 0.2

plt.bar(X_axis - 0.6 + budge, set0, 0.3, label = set0_name)
plt.bar(X_axis - 0.45 + budge, set1, 0.3, label = set1_name)
plt.bar(X_axis - 0.3 + budge, set2, 0.3, label = set2_name)
plt.bar(X_axis - 0.15 + budge, set3, 0.3, label = set3_name)
plt.bar(X_axis + 0 + budge, set4, 0.3, label = set4_name)

plt.xticks(X_axis, x_data)
plt.xlabel('Number')
plt.ylabel('Percentage of regional total')
plt.title(title)
plt.legend()
add_annotation_plot(ax)
save_fig(title)
plt.show()

```

```

In [ ]: def plot_quadrants(ramparts, ditches, ne, se, sw, nw):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    #x_data = [x for x in range(11)]
    x_data = [x for x in range(8)]

    set0_name = 'Ramparts'
    set00_name = 'Ditches'
    set1_name = 'NE'
    set2_name = 'SE'
    set3_name = 'SW'
    set4_name = 'NW'

    set0 = ramparts['Enclosing_Max_Ramparts'].value_counts()
    set0 = order_set(set0, x_data)[:8]
    set00 = ditches['Enclosing_Ditches_Number'].value_counts()
    set00 = order_set(set00, x_data)[:8]
    set1 = ne['Enclosing_NE_Quadrant'].value_counts()
    set1 = order_set(set1, x_data)[:8]
    set2 = se['Enclosing_SE_Quadrant'].value_counts()
    set2 = order_set(set2, x_data)[:8]
    set3 = sw['Enclosing_SW_Quadrant'].value_counts()
    set3 = order_set(set3, x_data)[:8]
    set4 = nw['Enclosing_NW_Quadrant'].value_counts()
    set4 = order_set(set4, x_data)[:8]

    X_axis_s = np.arange(len(x_data[:8]))

    budge = 0.2

    plt.bar(X_axis - 0.6 + budge, set0, 0.2, label = set0_name)
    plt.bar(X_axis - 0.46 + budge, set00, 0.2, label = set00_name)
    plt.bar(X_axis - 0.32 + budge, set1, 0.2, label = set1_name)
    plt.bar(X_axis - 0.18 + budge, set2, 0.2, label = set2_name)
    plt.bar(X_axis - 0.04 + budge, set3, 0.2, label = set3_name)
    plt.bar(X_axis + 0.1 + budge, set4, 0.2, label = set4_name)

    plt.xticks(X_axis, x_data)
    plt.xlabel('Number')
    plt.ylabel('Count')
    title = 'Ditches, Ramparts and Quadrant by Number'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [ ]: def plot_regions(nw, ne, ni, si, s, features, xlabel, title, split_pos, \
                      yes_no, pcent=False):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])

    x_data = features
    x_data = [x.split('_')[split_pos:] for x in x_data]

```

```

x_data_new = []
for l in x_data:
    txt = ""
    for part in l:
        txt += "_" + part
    x_data_new.append(txt[1:])

set0_name = 'NW'
set1_name = 'NE'
set2_name = 'N Ireland'
set3_name = 'S Ireland'
set4_name = 'S'

set0_list = []
set1_list = []
set2_list = []
set3_list = []
set4_list = []

for feature in features:
    set0_list.append((nw[feature].values == yes_no).sum())
    set1_list.append((ne[feature].values == yes_no).sum())
    set2_list.append((ni[feature].values == yes_no).sum())
    set3_list.append((si[feature].values == yes_no).sum())
    set4_list.append((s[feature].values == yes_no).sum())

set0 = set0_list
set1 = set1_list
set2 = set2_list
set3 = set3_list
set4 = set4_list

if pcent:
    set0 = get_pcent_list(set0)
    set1 = get_pcent_list(set1)
    set2 = get_pcent_list(set2)
    set3 = get_pcent_list(set3)
    set4 = get_pcent_list(set4)

X_axis = np.arange(len(x_data))

budge = 0.3

plt.bar(X_axis - 0.6 + budge, set0, 0.13, label = set0_name)
plt.bar(X_axis - 0.45 + budge, set1, 0.13, label = set1_name)
plt.bar(X_axis - 0.3 + budge, set2, 0.13, label = set2_name)
plt.bar(X_axis - 0.15 + budge, set3, 0.13, label = set3_name)
plt.bar(X_axis + 0 + budge, set4, 0.13, label = set4_name)

plt.xticks(X_axis, x_data_new)
plt.xlabel('Xlabel')
if pcent:
    plt.ylabel('Percentage of Regional Total')
else:
    plt.ylabel('Count')
plt.title(get_print_title(f'{title}'))
plt.legend()
add_annotation_plot(ax)
save_fig(title)
plt.show()

```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

```

In [ ]: def test_numeric(data):
    temp_data = data.copy()
    columns = data.columns
    out_cols = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
    feat, ent, num, non, nul = [], [], [], [], []
    for col in columns:
        if temp_data[col].dtype == 'object':
            feat.append(col)
            temp_data[col + '_num'] = temp_data[col].str.isnumeric()
            entries = temp_data[col].notnull().sum()
            true_count = temp_data[col + '_num'][temp_data[col + '_num'] == \
                                                True].sum()
            null_count = temp_data[col].isna().sum()
            ent.append(entries)
            num.append(true_count)
            non.append(entries - true_count)
            nul.append(null_count)
        else:
            print(f'{col} {temp_data[col].dtype}') 396

```

```

summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
summary.columns = out_cols
return summary

In [ ]: def find_duplicated(numeric_data, text_data, encodeable_data):
    d = False
    all_columns = \
        list(numeric_data.columns) + list(text_data.columns) + \
        list(encodeable_data.columns)
    duplicate = \
        [item for item, count in collections.Counter(all_columns).items() \
         if count > 1]
    if duplicate:
        print(f"There are duplicate features: {duplicate}")
        d = True
    return d

In [ ]: def test_data_split(main_data, numeric_data, text_data, encodeable_data):
    m = False
    split_features = \
        list(numeric_data.columns) + list(text_data.columns) + \
        list(encodeable_data.columns)
    missing = list(set(main_data)-set(split_features))
    if missing:
        print(f"There are missing features: {missing}")
        m = True
    return m

In [ ]: def review_data_split(main_data, numeric_data, text_data, \
                           encodeable_data = pd.DataFrame()):
    d = find_duplicated(numeric_data, text_data, encodeable_data)
    m = test_data_split(main_data, numeric_data, text_data, encodeable_data)
    if d != True and m != True:
        print("Data split good.")

In [ ]: def find_duplicates(data):
    print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')

```

```

In [ ]: def count_yes(data):
    total = 0
    for col in data.columns:
        count = len(data[data[col] == 'Yes'])
        total += count
        print(f'{col}: {count}')
    print(f'Total yes count: {total}')

```

Null Value Functions

The following functions will be used to update null values.

```

In [ ]: def fill_nan_with_minus_one(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna(-1)
    return new_data

In [ ]: def fill_nan_with_NA(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna("NA")
    return new_data

In [ ]: def test_numeric_value_in_feature(feature, value):
    test = feature.isin([-1]).sum()
    return test

In [ ]: def test_categorical_value_in_feature(dataframe, feature, value):
    test = dataframe[feature][dataframe[feature] == value].count()
    return test

In [ ]: def test_cat_list_for_NA(dataframe, cat_list):
    for val in cat_list:
        print(val, test_categorical_value_in_feature(dataframe, val, 'NA'))

In [ ]: def test_num_list_for_minus_one(dataframe, num_list):
    for val in num_list:
        feature = dataframe[val]
        print(val, test_numeric_value_in_feature(feature, -1))

In [ ]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()

```

```

    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data

```

```

In [ ]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data

```

Reprocessing Functions

```

In [ ]: def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data

```

Save Image Functions

```

In [ ]: # Set-up figure numbering
fig_no = 0
part = 'Part05'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"

```

```

In [ ]: # Remove unicode characters from file names
def get_file_name(title):
    file_name = slugify(title)
    return file_name

```

```

In [ ]: # Remove underscore from figure titles
def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(";", " ")
    return title

```

```

In [ ]: # Format figure numbers to have three digits
def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no

```

```

In [ ]: # Mount Google Drive if figures to be saved
if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass

```

```

In [ ]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Part_05_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution, \
                   bbox_inches='tight')
    else:
        pass

```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [ ]: hillforts_csv = r"https://raw.githubusercontent.com/MikeDarsie/Hillforts-Primer/main/hillforts-atlas-source-data.csv"
#"https://raw.githubusercontent.com/MikeDarsie/Hillforts-Primer/main/hillforts-atlas-source-data-csv/hillforts.csv"
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)

<ipython-input-73-b2855eddaef9>: 4: DtypeWarning: Columns (10, 12, 68, 83, 84, 85, 86, 165, 183) have mixed types. Specify dt
ype option on import or set low_memory=False.
    hillforts_data = pd.read_csv(hillforts_csv, index_col=False)

Out[ ]:   OBJECTID  Main_Atlas_Number  Main_Country_Code  Main_Country  Main_Title_Name  Main_Site_Name  Main_Alt_Name  Main_Display_N
          0             1                  1                EN        England      EN0001 Aconbury
                                         Camp,           Aconbury Camp       Aconbury Beacon  Aconbury C
                                         Herefordshire           (Aconbury Bea
                                         1             2                  2                EN        England      EN0002 Bach
                                         Camp,           Bach Camp           NaN           Bach C
                                         Herefordshire           Hereford           Hereford
```

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [ ]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
            'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.')

Using all 4147 record in the Hillforts Atlas.
```

Download Function

```
In [ ]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = \
            hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', \
                'republic-of-ireland', 'northern-ireland', \
                'isle-of-man', 'roi-ni', 'eng-wal-sco-iom']:
                if pkg.shape[0] == hillforts_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
                else:
                    dl = data_list[0]
                    dl = dl.replace('\r', ' ', regex=True)
                    dl = dl.replace('\n', ' ', regex=True)
                    fn = 'hillforts_primer_' + filename
                    fn = get_file_name(fn)
                    dl.to_csv(fn+'.csv', index=False)
                    files.download(fn+'.csv')
            else:
                pass
```

Reload Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-data packages. Where needed, these data extracts will be combined with Name and Number features to ensure the data can be understood and can, if needed, be concorded.

```
In [ ]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Reload Location

```
In [ ]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short)
location_numeric_data_short.head()
location_data = location_numeric_data_short.copy()
location_data.head()
```

	Location_X	Location_Y	Density
0	-303295	6798973	1.632859e-12
1	-296646	6843289	1.540172e-12
2	-289837	6808611	1.547729e-12
3	-320850	6862993	1.670548e-12
4	-261765	6810587	1.369981e-12

Reload Dating

```
In [ ]: date_features = [
'Dating_Date_Pre_1200BC',
'Dating_Date_1200BC_800BC',
'Dating_Date_800BC_400BC',
'Dating_Date_400BC_AD50',
'Dating_Date_AD50_AD400',
'Dating_Date_AD400_AD800',
'Dating_Date_Post_AD800',
'Dating_Date_Unknown']

date_data = hillforts_data[date_features].copy()
date_data.head()
```

	Dating_Date_Pre_1200BC	Dating_Date_1200BC_800BC	Dating_Date_800BC_400BC	Dating_Date_400BC_AD50	Dating_Date_AD50_AD400	Dating_Date_Post_AD800	Dating_Date_Unknown
0	No	No	Yes	Yes	Yes	Yes	Yes
1	No	No	No	No	No	No	No
2	No	No	No	No	No	No	No
3	No	No	No	No	No	No	Yes
4	No	No	No	Yes	Yes	Yes	Yes

Reload Regional Data Packages

See Cluster Data Packages in Part 1: Name, Admin & Location Data

<https://colab.research.google.com/drive/1C7HcuLuGGhG8o4EGciS-XTAhxVs3MhX3?usp=sharing>

```
In [ ]: cluster_data = \
hillforts_data[['Location_X', 'Location_Y', 'Main_Country_Code']].copy()
cluster_data['Cluster'] = 'NA'
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
'NI', 'I', inplace=True)
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
'IR', 'I', inplace=True)

cluster_data['Cluster'] = np.where(
(cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000), \
400
```

```

    'North Ireland', cluster_data['Cluster']
)
north_ireland = cluster_data[cluster_data['Cluster'] == 'North Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , \
    'South Ireland', cluster_data['Cluster']
)
south_ireland = cluster_data[cluster_data['Cluster'] == 'South Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] < 7070000) , \
    'South', cluster_data['Cluster']
)
south = cluster_data[cluster_data['Cluster'] == 'South'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] >= -500000), 'Northeast', cluster_data['Cluster']
)
north_east = cluster_data[cluster_data['Cluster'] == 'Northeast'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] < -500000), 'Northwest', cluster_data['Cluster']
)
north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()

temp_cluster_location_packages = [north_ireland, south_ireland, south, \
                                   north_east, north_west]

cluster_packages = []
for pkg in temp_cluster_location_packages:
    pkg = pkg.drop(['Main_Country_Code'], axis=1)
    cluster_packages.append(pkg)

north_ireland, south_ireland, south, north_east, north_west = \
cluster_packages[0], cluster_packages[1], cluster_packages[2], \
cluster_packages[3], cluster_packages[4]

```

Review Data Part 5

Entrance Data

Additional information relating to entrances is contained in an Entrances Table. This can be downloaded from the Hillforts Atlas Rest Service API [here](#) or this project's data store [here](#). The Entrances Table has not been analysed as part of the Hillforts Primer at this time.

```
In [ ]: entrance_features = [
    'Entrances_Breaks',
    'Entrances_Breaks_Comments',
    'Entrances_Original',
    'Entrances_Original_Comments',
    'Entrances_Guard_Chambers',
    'Entrances_Chevaux',
    'Entrances_Chevaux_Comments',
    'Entrances_Summary',
    'Related_Entrances']

entrance_data = hillforts_data[entrance_features].copy()
entrance_data.head()
```

		Entrances_Breaks	Entrances_Breaks_Comments	Entrances_Original	Entrances_Original_Comments	Entrances_Guard_Chambers	Entrances_C...
0	6.0	Two original and four modern gaps.		2.0	Two original inturned entrances at SE and SW c...		No
1	3.0	N entrance damaged by wagon access and possibl...		2.0	S entrance original, that on the NW possibly ...		No
2	2.0	Entrances intact		2.0	Interesting inturn to N entrance		No
3	3.0	Modern gap to the S.		2.0	Off-set entrance on the E. Possibly another to...		No
4	6.0	Probable modern breaks not recorded.		6.0	Two entrances are from Phase I and four from P...		No

There are null values in all but two entrance features.

In []: `entrance_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Entrances_Breaks    3830 non-null   float64
 1   Entrances_Breaks_Comments 1193 non-null   object 
 2   Entrances_Original     3941 non-null   float64
 3   Entrances_Original_Comments 1126 non-null   object 
 4   Entrances_Guard_Chambers 4147 non-null   object 
 5   Entrances_Chevaux      4147 non-null   object 
 6   Entrances_Chevaux_Comments 77 non-null   object 
 7   Entrances_Summary      4132 non-null   object 
 8   Related_Entrances     2749 non-null   object 
dtypes: float64(2), object(7)
memory usage: 291.7+ KB
```

Entrance Numeric Data

There are two numeric features. Both contain null values that will be resolved below.

In []: `entrance_numeric_features = [
 'Entrances_Breaks',
 'Entrances_Original']`

```
entrance_numeric_data = entrance_data[entrance_numeric_features].copy()  
entrance_numeric_data.head()
```

	Entrances_Breaks	Entrances_Original
0	6.0	2.0
1	3.0	2.0
2	2.0	2.0
3	3.0	2.0
4	6.0	6.0

Entrance Numeric Data - Resolve Null Values

Test for -1.

In []: `test_num_list_for_minus_one(entrance_numeric_data, entrance_numeric_features)`

```
Entrances_Breaks 0  
Entrances_Original 0
```

Replace null with -1.

In []: `entrance_numeric_data = \
update_num_list_for_minus_one(entrance_numeric_data, entrance_numeric_features)
entrance_numeric_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Entrances_Breaks    4147 non-null   float64
 1   Entrances_Original  4147 non-null   float64
dtypes: float64(2)
memory usage: 64.9 KB
```

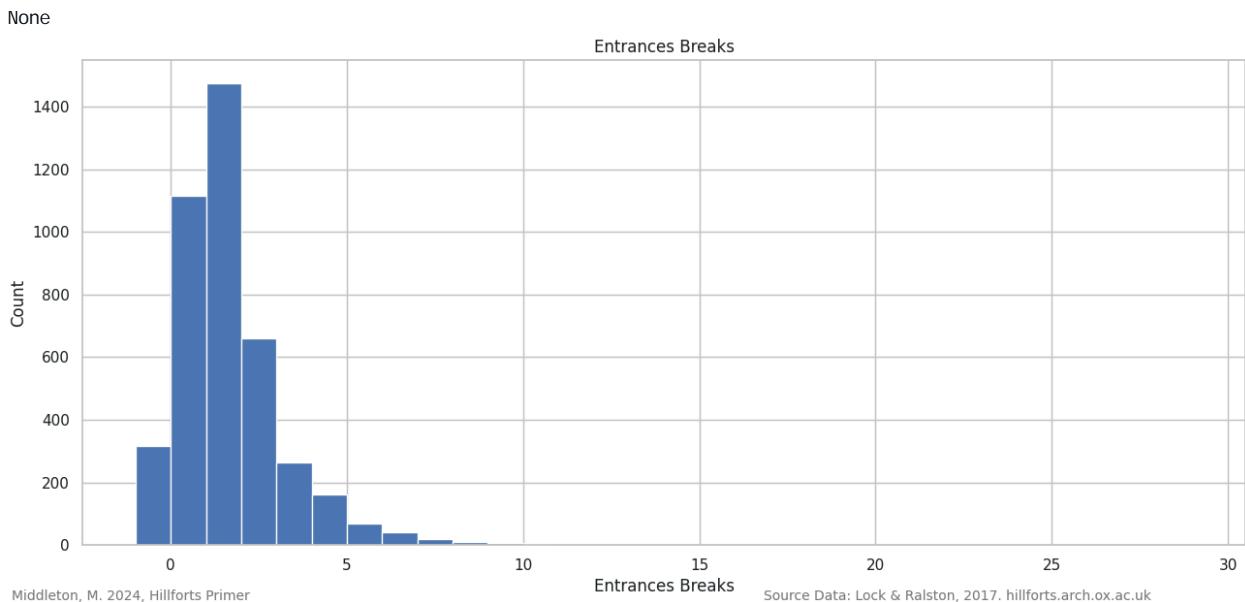
Entrances Breaks Data Plotted

Entrance breaks has a long tail of outliers. Most hillforts (90.26%) have five entrances or less. 78.3% have two entrances or less.

```
In [ ]: entrance_numeric_data['Entrances_Breaks'].value_counts().sort_index()
```

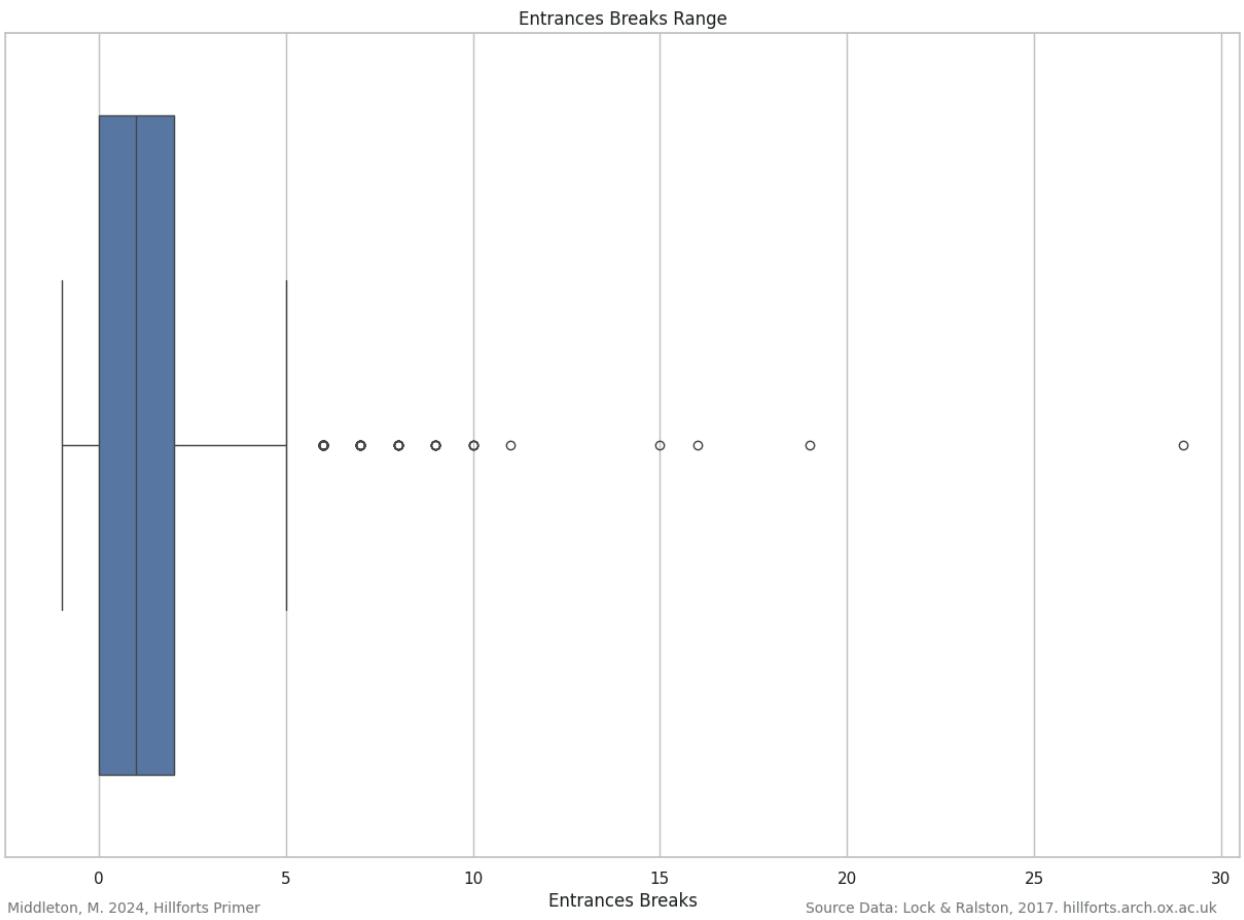
```
Out[ ]: -1.0      317
         0.0     1114
         1.0     1474
         2.0      661
         3.0      264
         4.0      162
         5.0      68
         6.0      42
         7.0      19
         8.0      10
         9.0      7
        10.0      4
        11.0      1
        15.0      1
        16.0      1
        19.0      1
       29.0      1
Name: Entrances_Breaks, dtype: int64
```

```
In [ ]: plot_histogram(entrance_numeric_data['Entrances_Breaks'], 'Entrances Breaks', \
                      'Entrances Breaks')
```



Outliers range from 6 to 29 entrances.

```
In [ ]: entrances_breaks_data = \
plot_data_range(entrance_numeric_data['Entrances_Breaks'], \
                 'Entrances Breaks', "h")
```



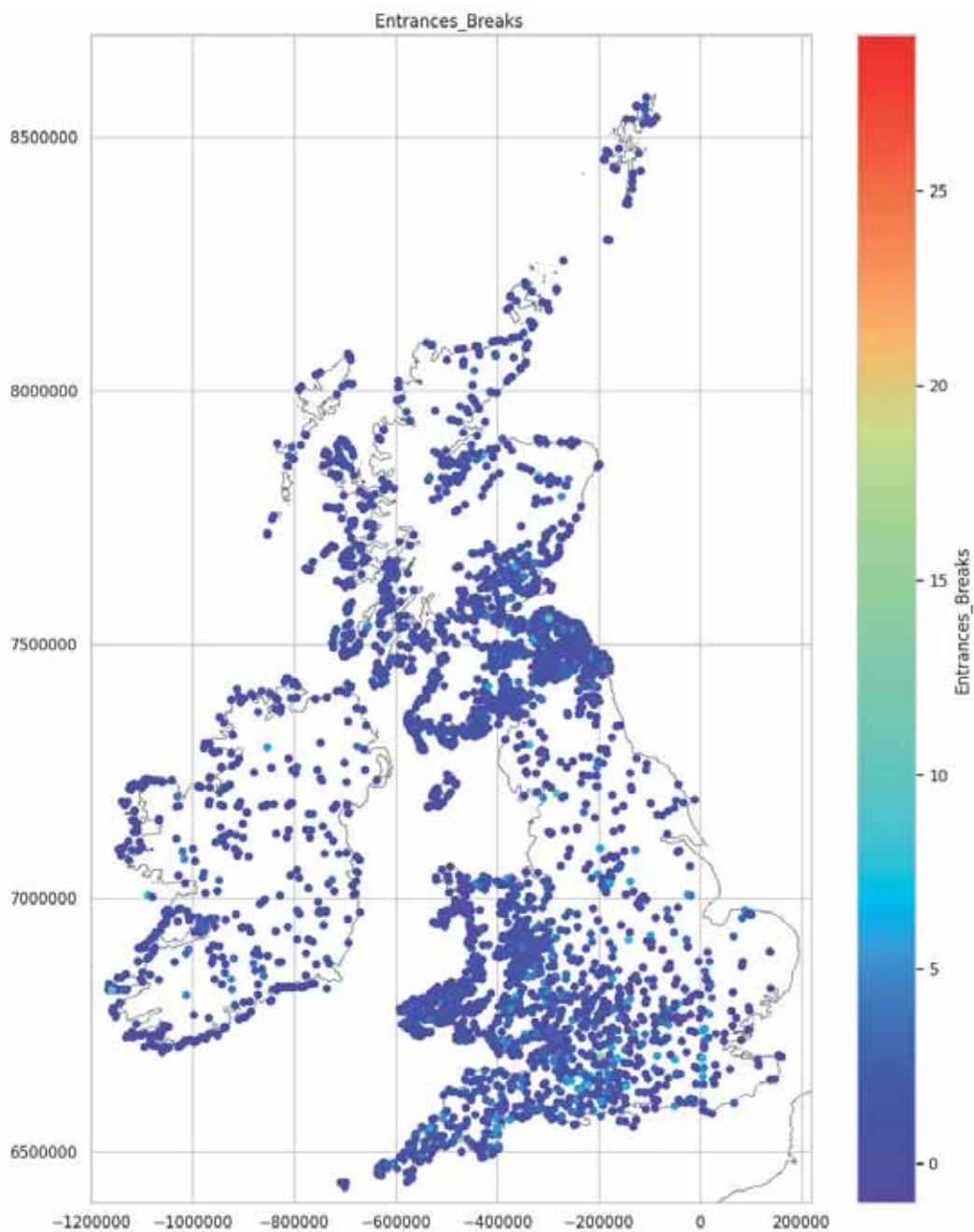
```
In [ ]: entrances_breaks_data
Out[ ]: [-1.0, 0.0, 1.0, 2.0, 5.0]
```

Entrance Breaks Mapped

The high concentration of hillforts with five entrances or less, and the long tail up to 29 entrances, causes the plot of entrance breaks to lack clarity. The options are to reproject the data using a boxcox projection or to split the data into ranges. In this case, splitting the data using quartile ranges, will plot meaningful groupings while reducing the plot range of each figure. This will improve the clarity of each map.

```
In [ ]: location_entrance_data = \
pd.merge(location_numeric_data_short, entrance_numeric_data, \
left_index=True, right_index=True)

In [ ]: plot_values(location_entrance_data, 'Entrances_Breaks', 'Entrances_Breaks')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

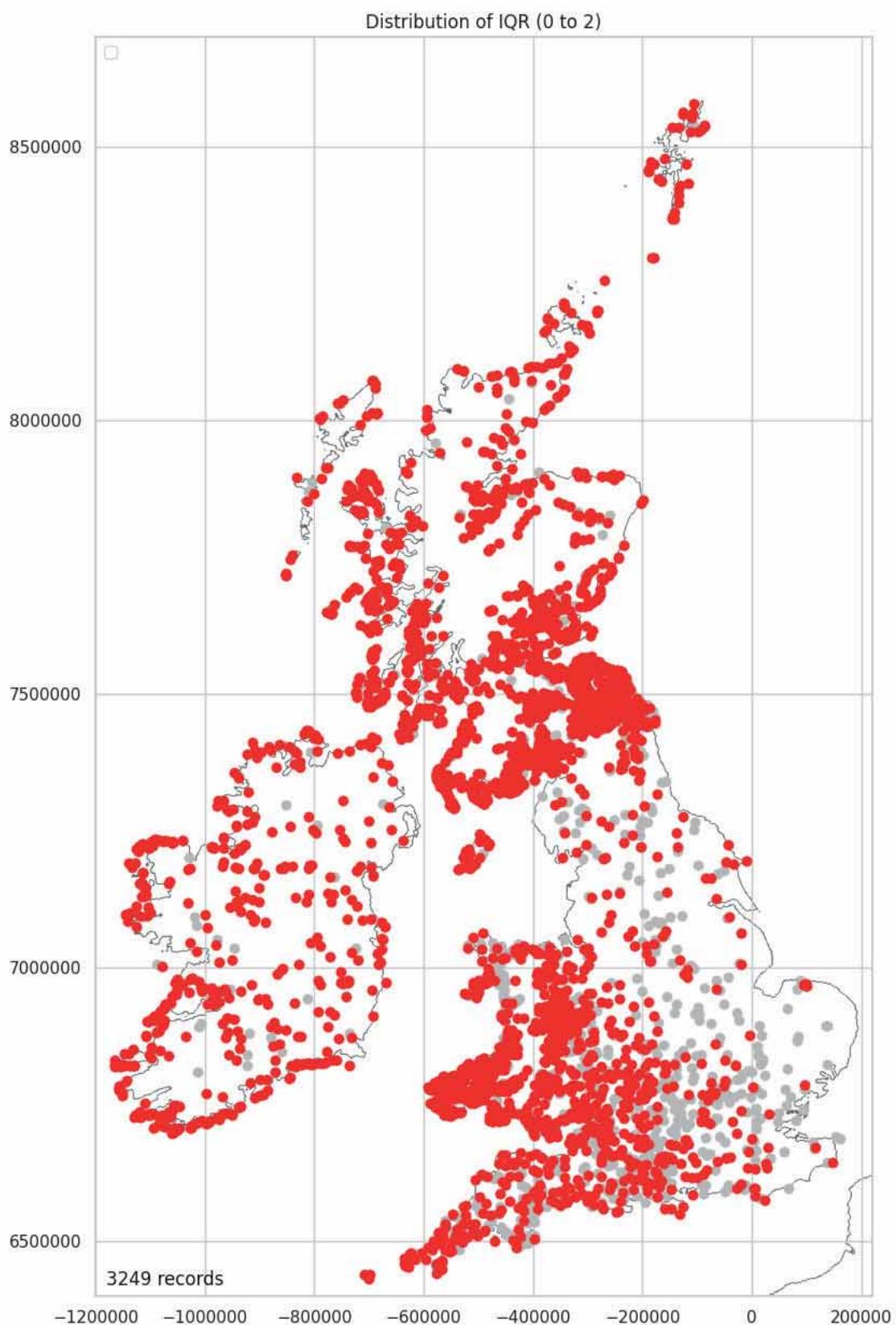
Entrance Breaks Interquartile Range (Mid 50%) Distribution Mapped

Most hillforts have zero to two entrances (78.3%). All coastal forts in Ireland, most in the west and north of Scotland and most in the Welsh uplands fall in this range. The Northeast and the South both have a large number of hillforts that fall within this range.

```
In [ ]: eb_iqr = \
location_entrance_data[location_entrance_data['Entrances_Breaks'].between(0, 2)].copy()
eb_upper_q = \
location_entrance_data[location_entrance_data['Entrances_Breaks'].between(2, 5)].copy()
eb_out = \
location_entrance_data[location_entrance_data['Entrances_Breaks']>5].copy()
```

```
In [ ]: print(f'{round(len(eb_iqr)/len(location_entrance_data)*100, 2)}% of hillforts have two entrances or less (IQR).')
78.35% of hillforts have two entrances or less (IQR).
```

```
In [ ]: plot_over_grey_numeric(eb_iqr, 'Entrances_Breaks', 'Distribution of IQR (0 to 2)')
```



Middleton, M. 2024, Hillforts Primer

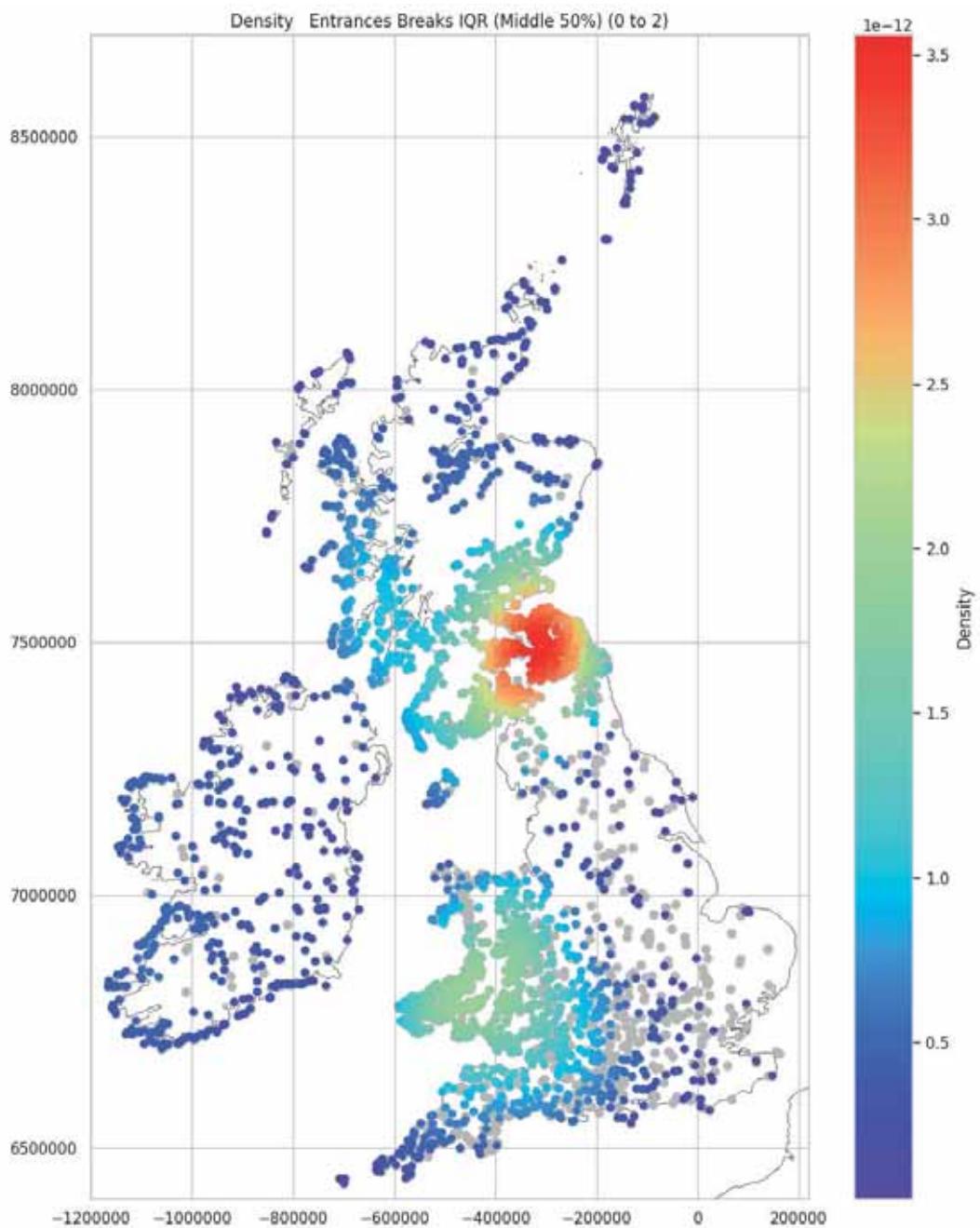
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

78.35%

Entrance Breaks Interquartile Range (Mid 50%) Density Mapped

This range has a high representation of hillforts from all regions meaning the distribution clusters, seen when plotting the location data in, Part 1: Density Data Transformed Mapped, are replicated in this subset of the Entrance Breaks data.

```
In [ ]: plot_density_over_grey(eb_iqr, 'Entrances_Breaks', 'IQR (Middle 50%) (0 to 2)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

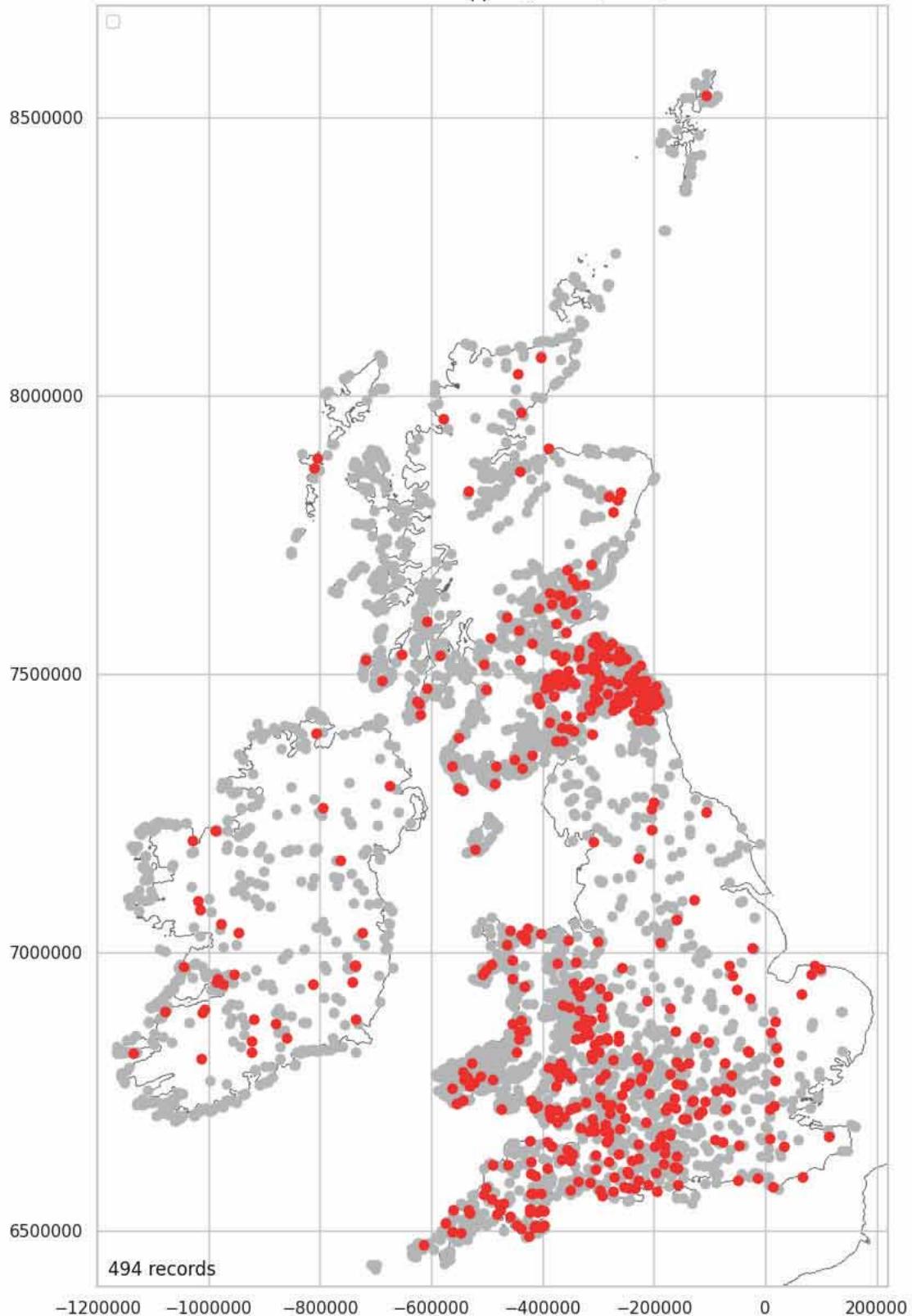
Entrance Breaks Upper Quartile Distribution Mapped

Only 11.9% of hillforts have three to five entrances.

```
In [ ]: print(f' {round(len(eb_upper_q)/len(location_entrance_data)*100, 2)} } have three to five entrances (Upper quartile.)')
11.91 have three to five entrances (Upper quartile).
```

```
In [ ]: plot_over_grey_numeric(eb_upper_q, 'Entrances_Breaks',
'Distribution of Upper Quartile (3 to 5)')
```

Distribution of Upper Quartile (3 to 5)



Middleton, M. 2024, Hillforts Primer

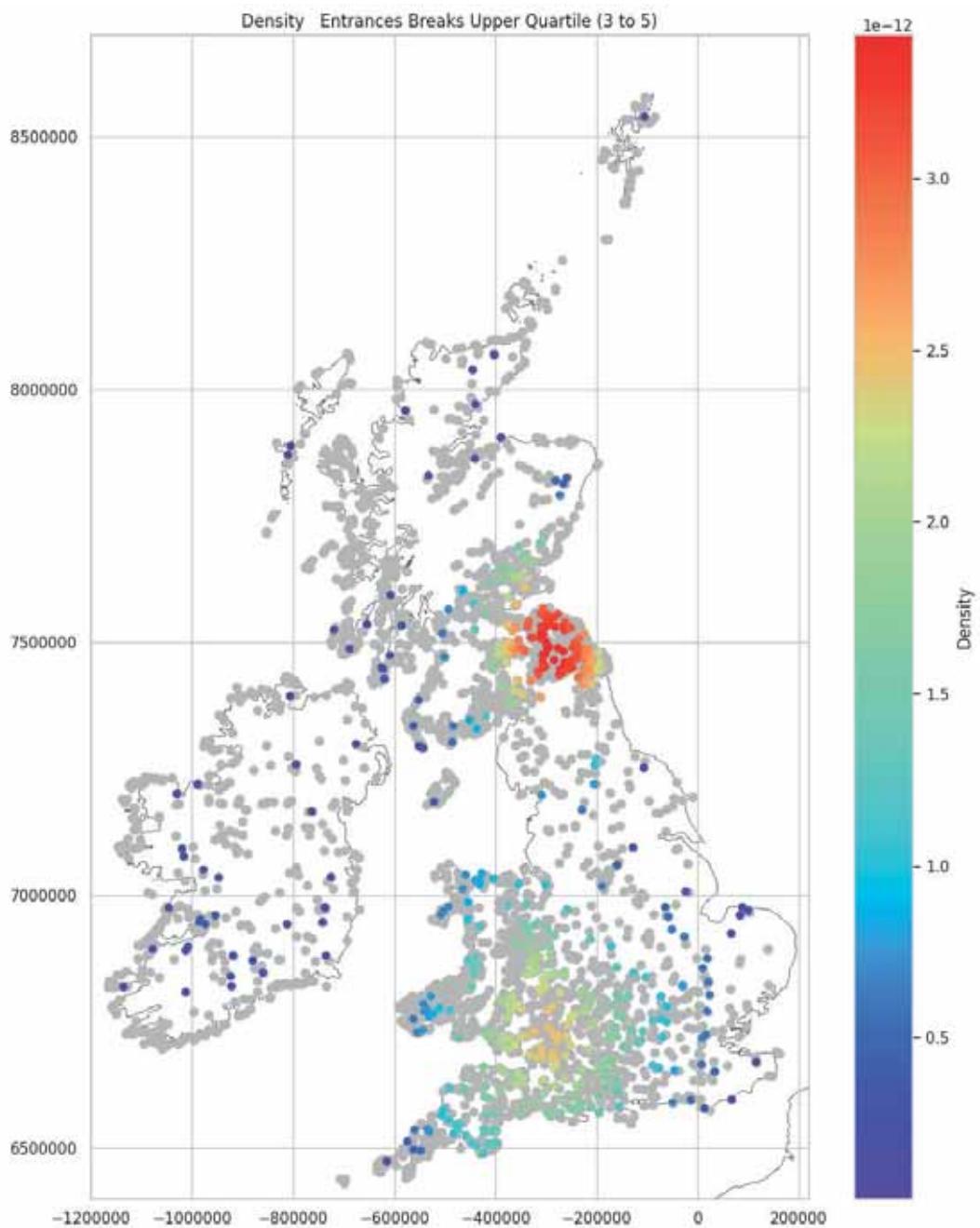
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

11. 91%

Entrance Breaks Upper Quartile Density Mapped

Hillforts with three to five entrances are cluster in the Northeast and south, central England. There are very few in all other regions. In Ireland, most of this group are in the south.

```
In [ ]: plot_density_over_grey(eb_upper_q, 'Entrances_Breaks', 'Upper Quartile (3 to 5)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

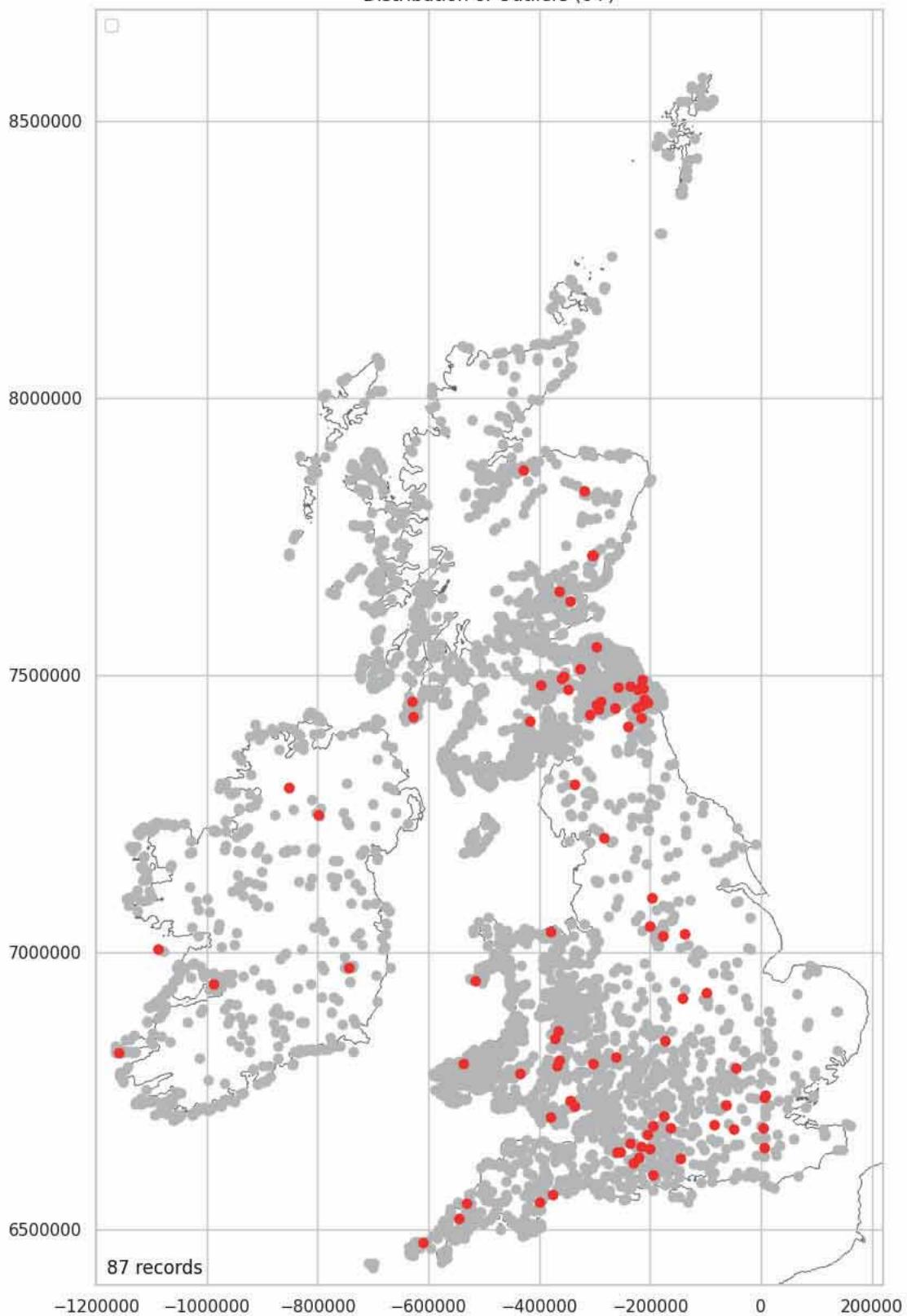
Entrance Breaks Outlier Distribution Mapped

There is a small concentration of hillforts with six or more entrances in the south of England, near the ridge way, and a similar small concentration in the Northeast. Most are peppered over western England, Wales and eastern Scotland. There is a notable survey bias visible in the Northeastern data, as can be seen by the increased density of these hillforts to the south of the Scottish border, in Northumberland. There is a similar recording cluster around Oxford.

```
In [ ]: print(f'{round(len(eb_out)/len(location_entrance_data)*100, 2)}% of hillforts have six or more entrances (Outliers).')
2.1% of hillforts have six or more entrances (Outliers).
```

```
In [ ]: plot_over_grey_numeric(eb_out, 'Entrances_Breaks', 'Distribution of Outliers (6+)')
```

Distribution of Outliers (6+)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2.1%

No Entrance Breaks Mapped

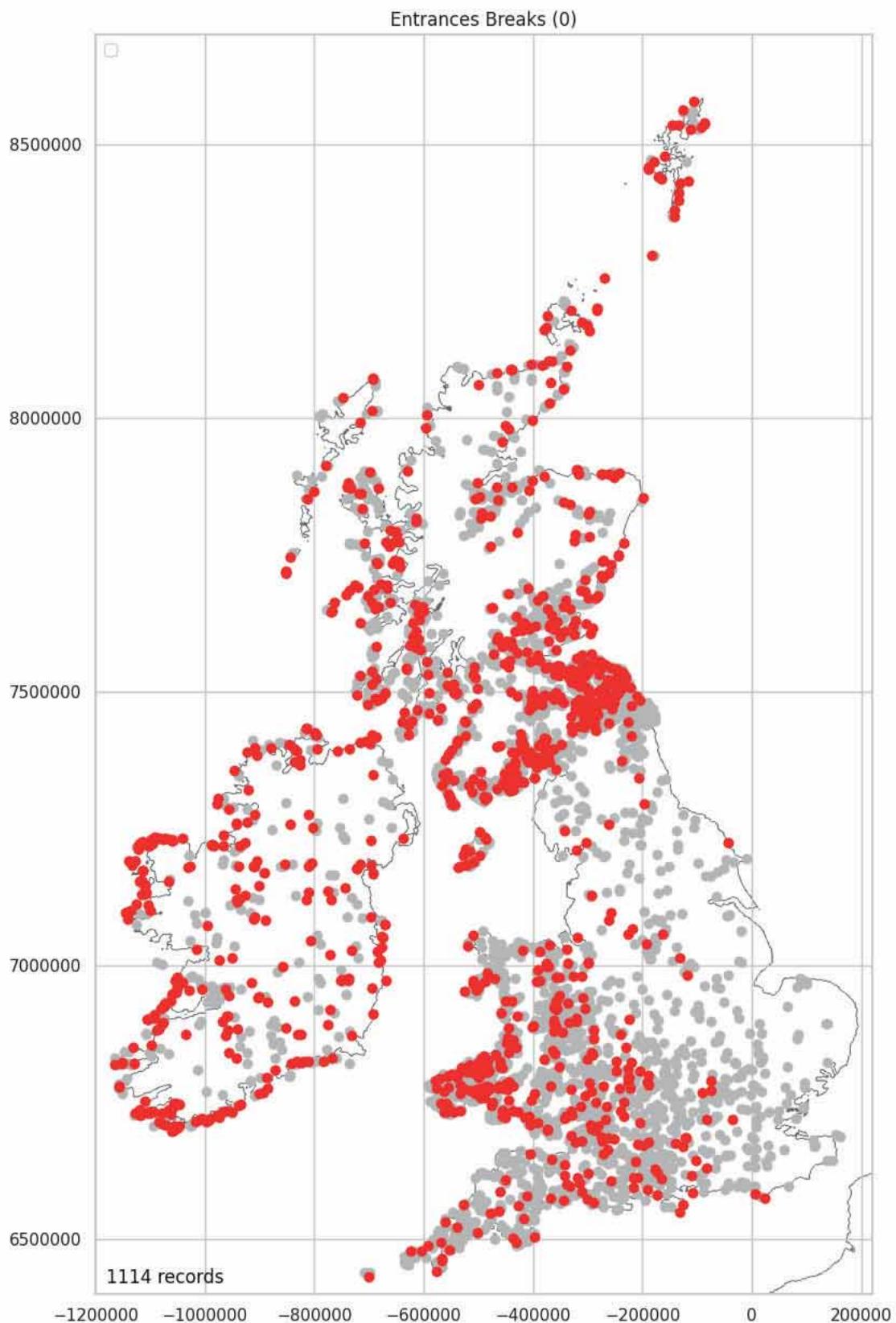
Just over a quarter of hillforts (26.86%) have no recorded entrance. These forts are most common at the northern end of the Northeastern cluster, in Pembrokeshire, up the southern end of the west coast of Scotland, over most of Ireland and peppered across the south west of England. Caution should be taken with regards the data in the Northeastern cluster in that, the southern boundary, between the intense concentration and no hillforts, is close to the England/Scotland border and it is likely that this reflects a recording bias in the data. If this is a recording bias, it does not replicate the bias, seen in other subsets of the data such as, Part 1: Main Boundary Mapped, where the modern border is clearly distinguishable. The fact that this line does not highlight the

modern border and it does not mirror the distribution seen in Part1: Northeast Data Mapped, may indicate that this is a meaningful distribution yet, it is still more likely to be the result of a recording bias. Hillforts with no recorded entrance may indicate that this information has not been recorded or there is no evidence of an entrance.

```
In [ ]: zero_enteances = \
location_entrance_data[location_entrance_data['Entrances_Breaks'] == 0].copy()
zero_enteances['Entrances_Breaks'] = "Yes"

In [ ]: print(f'{round(len(zero_enteances)/len(location_entrance_data)*100, 2)}% of hillforts have no recorded entrance.')
26.86% of hillforts have no recorded entrance.

In [ ]: zero_enteances_stats = plot_over_grey(zero_enteances, 'Entrances_Breaks', \
'Yes', '(0)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

26.86%

No Entrance Breaks Mapped (Northeast)

This figure shows the boundary between the high concentration of forts with no entrance breaks over the Southern Uplands and the abrupt line where this concentration stops, along the south side of this cluster.

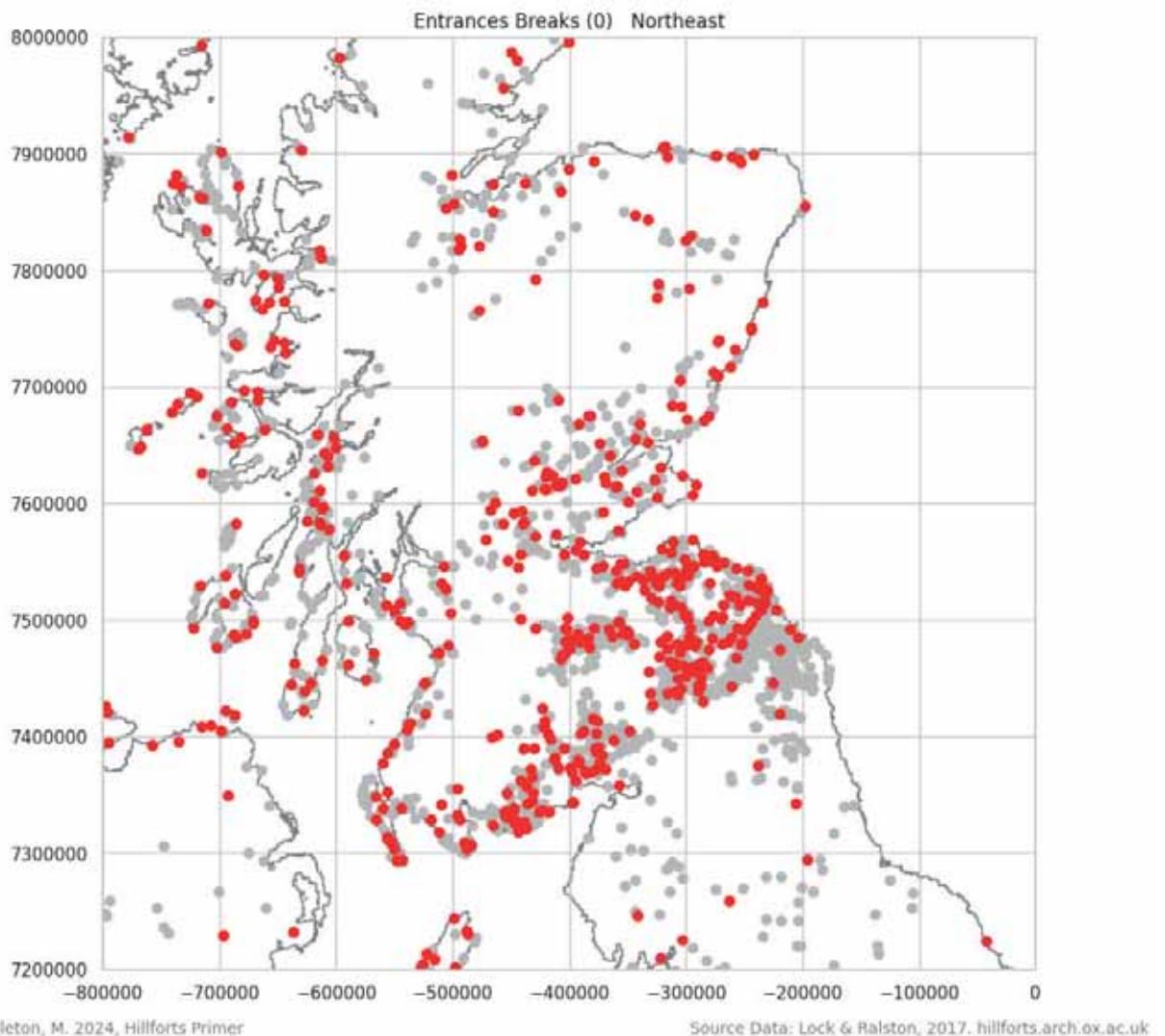
```
In [ ]: location_entrance_data_ne = \
location_entrance_data[location_entrance_data['Location_Y'] > 7070000].copy()
location_entrance_data_ne = \
location_entrance_data_ne[location_entrance_data_ne['Location_X'] > -800000].copy()
```

```

no_entances_ne = \
location_entrance_data_ne[location_entrance_data_ne['Entrances_Breaks'] == 0].copy()
no_entances_ne['Entrances_Breaks'] = "Yes"

In [ ]: no_entances_stats_ne = plot_over_grey_north(no_entances_ne, \
                                                 'Entrances_Breaks', 'Yes', \
                                                 '(0) - Northeast')

```



```

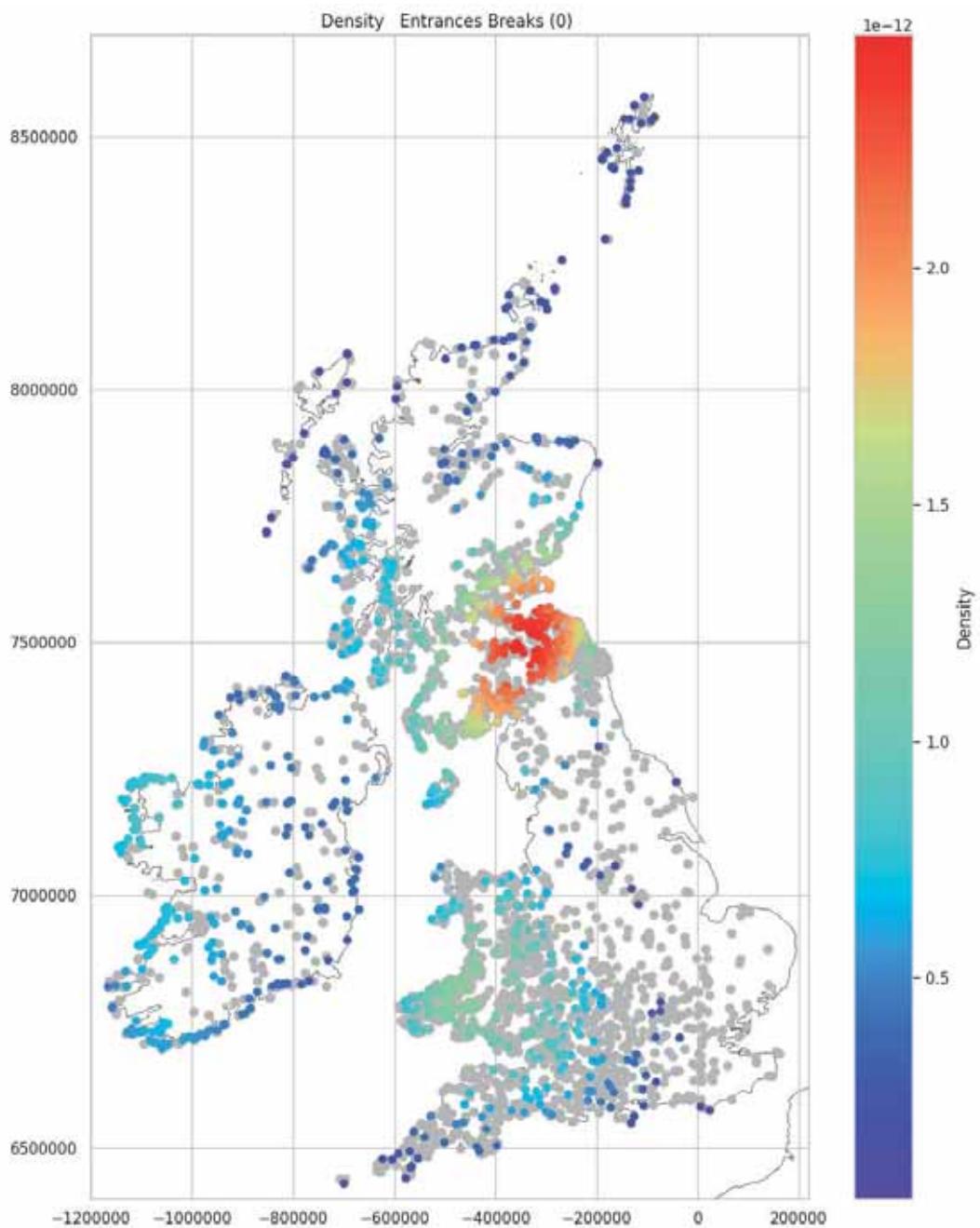
In [ ]: # This code can be used to get details of hillforts within certain x and y coordinate ranges
# To use this code, first run the document using Runtime > Run all, then remove the '#' from the lines
# starting temp below. Once removed press the Run cell button, on this cell, to the left.
# Update the 'Location_X' & 'Location_Y' values as required.
# temp = pd.merge(name_and_number, no_entances_ne, left_index=True, right_index=True)
# temp = temp[temp['Location_X'].between(-250000, -200000)]
# temp = temp[temp['Location_Y'].between(750000, 7510000)]
# temp.sort_values(by=['Location_X'], ascending=False)

```

No Entrance Breaks Density Mapped

All five clusters identified in, Part 1: Density Map showing Extent of Boxplots identified in the Atlas Data, can be seen in this subset of the data.

```
In [ ]: plot_density_over_grey(zero_entances_stats, 'Entrances_Breaks (0)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

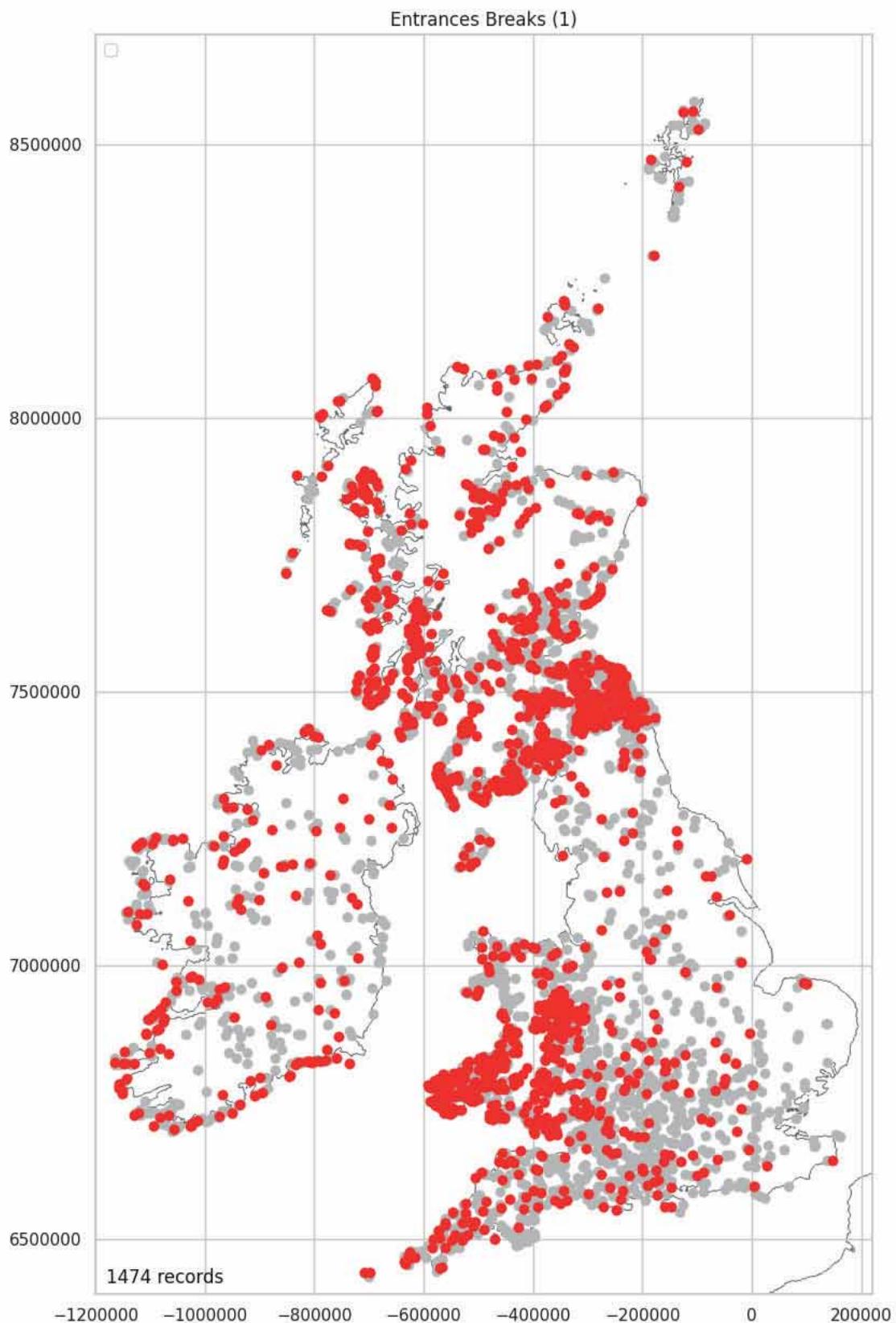
One Entrance Break Distribution Mapped

35.54% of hillforts have one recorded entrance. It is noticeable how few there are in south central England and northern Wales compared to how many there are over the Shropshire hills and southern Wales. The distinct difference between these areas may indicate a survey bias.

```
In [ ]: one_entrance = \
    location_entrance_data[location_entrance_data['Entrances_Breaks'] == 1].copy()
    one_entrance['Entrances_Breaks'] = "Yes"
```

```
In [ ]: print(f'{round(len(one_entrance)/len(location_entrance_data)*100, 2)}% of hillforts have one recorded entrance.')
35.54% of hillforts have one recorded entrance.
```

```
In [ ]: one_entrance_stats = \
    plot_over_grey(one_entrance, 'Entrances_Breaks', 'Yes', '(1)')
```



Middleton, M. 2024, Hillforts Primer

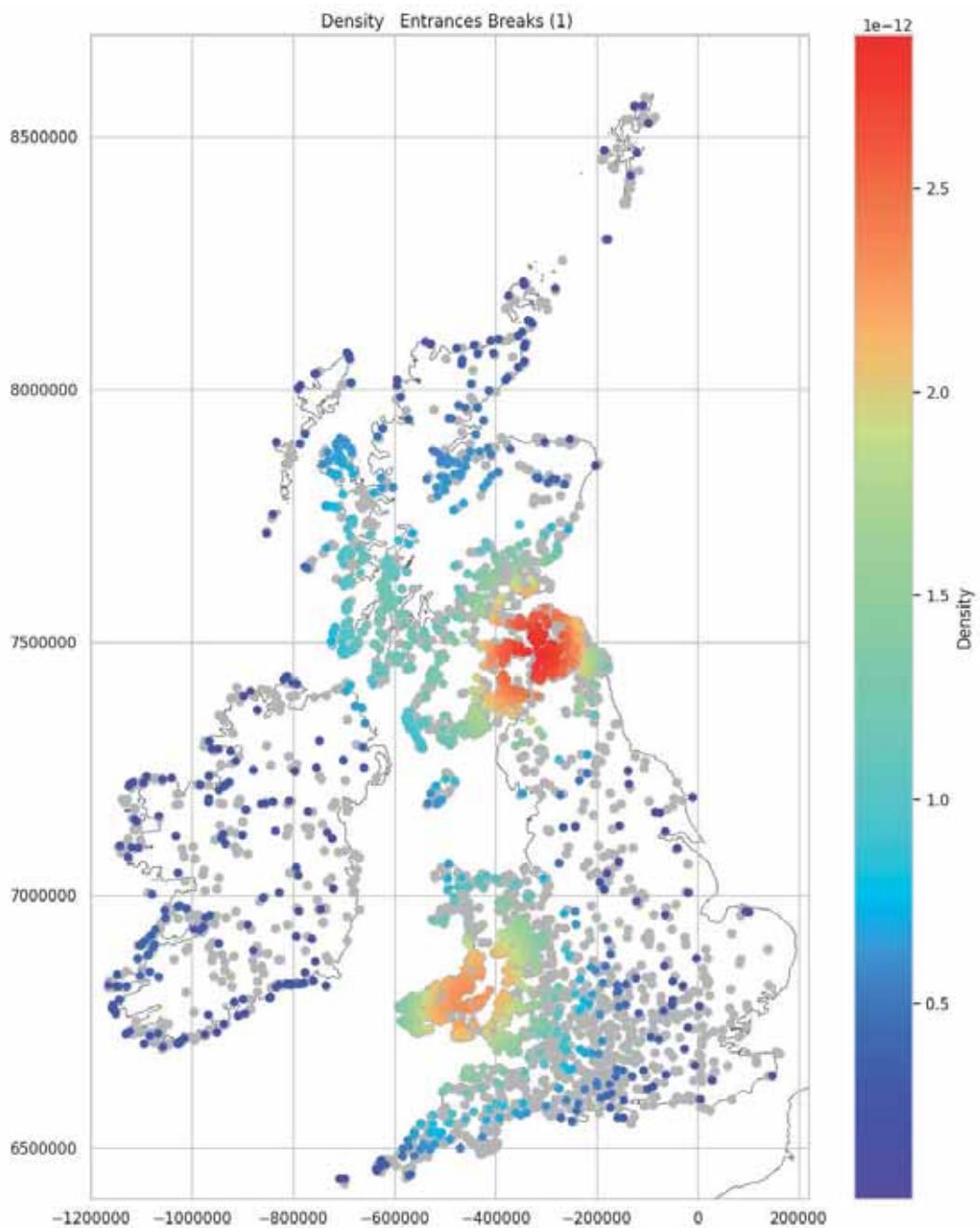
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

35. 54%

One Entrance Break Density Mapped

Single entrance hillforts are concentrated over the Southern Uplands, the southern Welsh uplands and along the south-western seaboard of Scotland. There is also a notable spread of these forts along the south-west of England and across central and western Ireland as well as clustering along the coasts of northern Scotland and south-western Ireland.

```
In [ ]: plot_densitiy_over_grey(one_entrance_stats, 'Entrances_Breaks (1)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Two Entrance Breaks Distribution Mapped

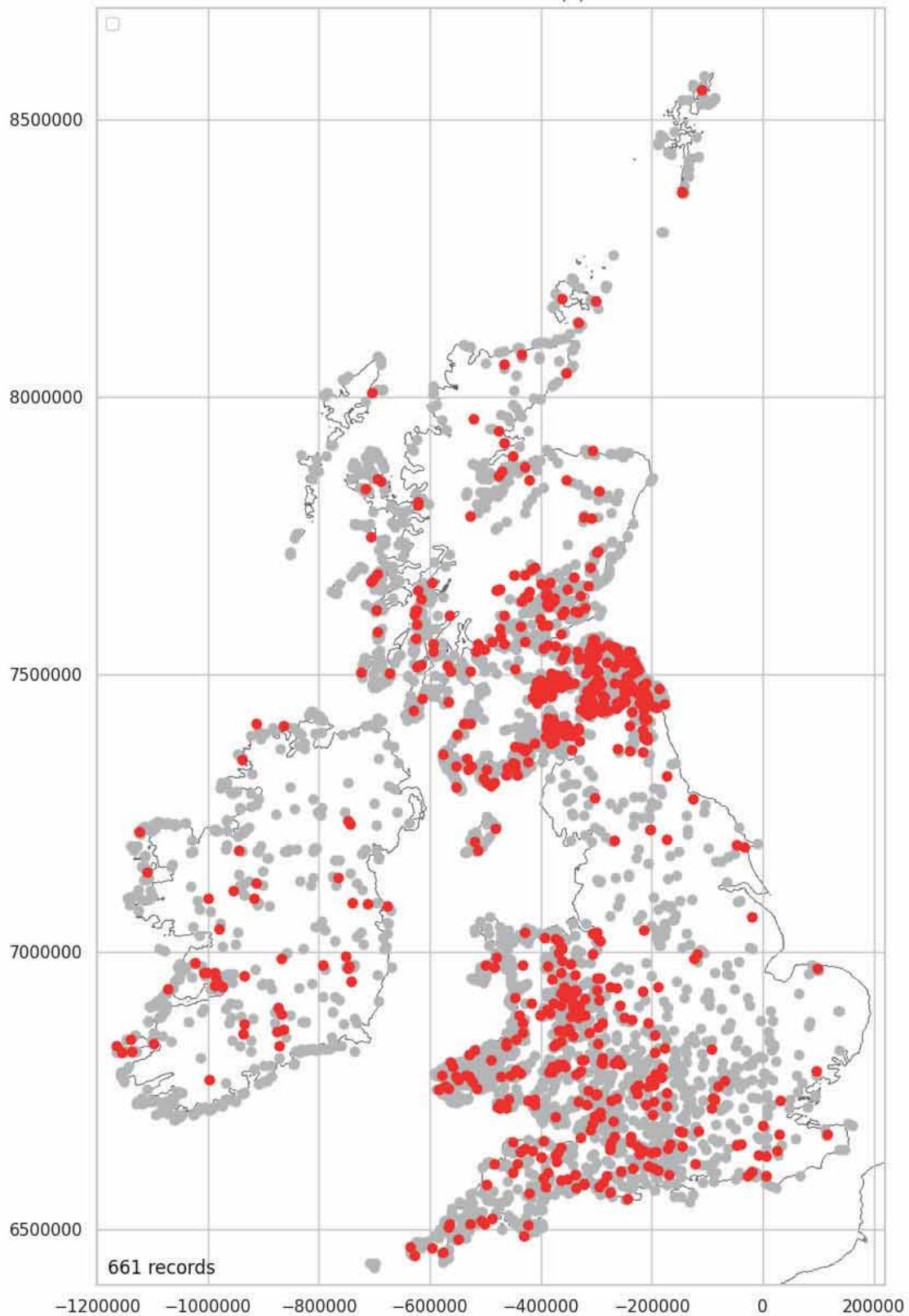
The distribution of two-entrance hillforts is more discreetly concentrated over the eastern Southern Uplands and to the east of the Cambrian Mountains. Interestingly, possible linear alignments of hillforts can be seen in the south of England, with the [Ridgeway](#) being the most prominent, running from the [Chiltern Hills](#) to [Lyme Bay](#).

```
In [ ]: two_entrances = \
location_entrance_data[location_entrance_data['Entrances_Breaks'] == 2].copy()
two_entrances['Entrances_Breaks'] = "Yes"
```

```
In [ ]: print(f'{round(len(two_entrances)/len(location_entrance_data)*100, 2)}% of hillforts have two entrances.')
15.94% of hillforts have two entrances.
```

```
In [ ]: two_entrances_stats = plot_over_grey(two_entrances, 'Entrances_Breaks', \
'Yes', '(2)')
```

Entrances Breaks (2)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

15. 94%

Two Entrance Breaks (South) & Possible Corolation to Linear Routes Mapped

An enlarged extract over the south showing some of the possible linear alignments of hillforts which may be highlighting routes and paths in this area.

```
In [ ]: location_entrance_data_s = \
location_entrance_data[location_entrance_data['Location_Y'] < 7070000].copy()
location_entrance_data_s = \
location_entrance_data_s[location_entrance_data_s['Location_X'] > -700000].copy()
```

```

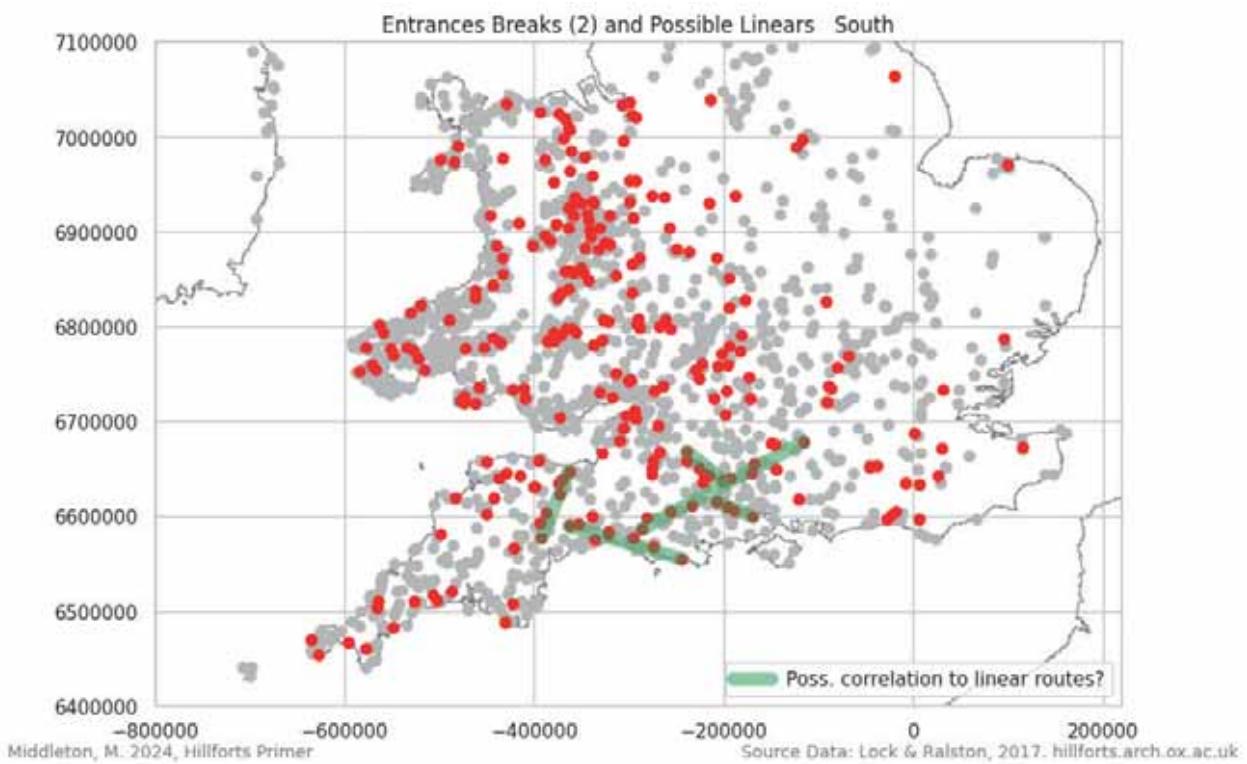
two_entrances_south = \
location_entrance_data_s[location_entrance_data_s['Entrances_Breaks'] == 2].copy()
two_entrances_south['Entrances_Breaks'] = "Yes"

```

```

In [ ]: two_entrances_stats_s = \
plot_over_grey_south(two_entrances_south, 'Entrances_Breaks', 'Yes', \
'(2) and Possible Linears - South')

```



```

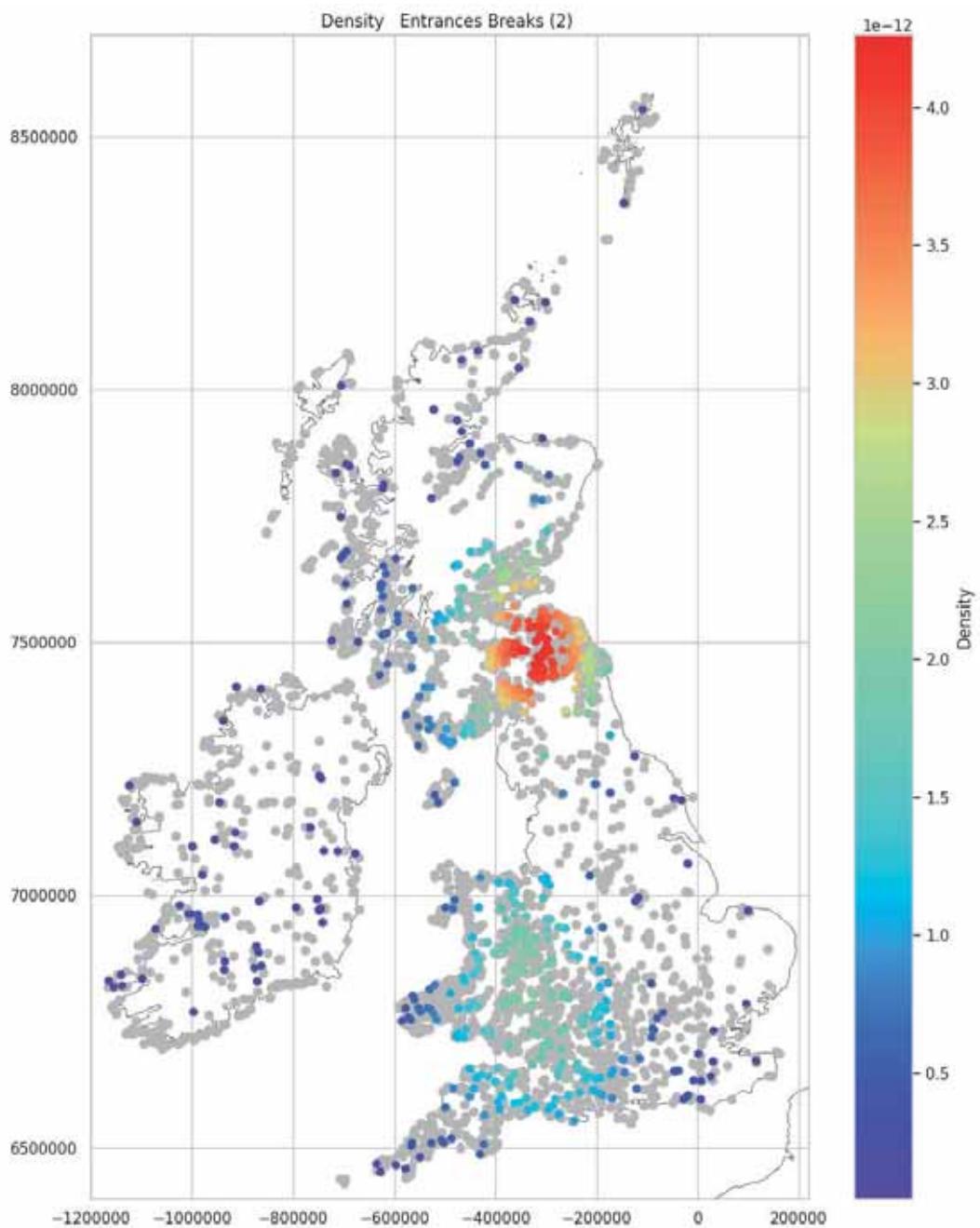
In [ ]: # This code can be used to get details of hillforts within certain x and y coordinate ranges
# To use this code, first run the document using Runtime > Run all, then remove the '#' from the lines
# starting temp below. Once removed press the Run cell button, on this cell, to the left.
# Update the 'Location_X' & 'Location_Y' values as required.
# temp = pd.merge(name_and_number, two_entrances_south, left_index=True, right_index=True)
# temp = temp[temp['Location_X'].between(-210000, -200000)]
# temp = temp[temp['Location_Y'].between(6620000, 6640000)]
# temp

```

Two Entrance Breaks Density Mapped

The focus of two entrance forts is in the Northeast and from the north end of the Cambrian Mountains, then along the eastern fringes of the Cambrian Mountains, down to the western end of south, central England. It is notable that there are almost none of this type around the Irish coast.

```
In [ ]: plot_density_over_grey(two_entrances_stats, 'Entrances_Breaks (2)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Three Entrance Breaks Distribution Mapped

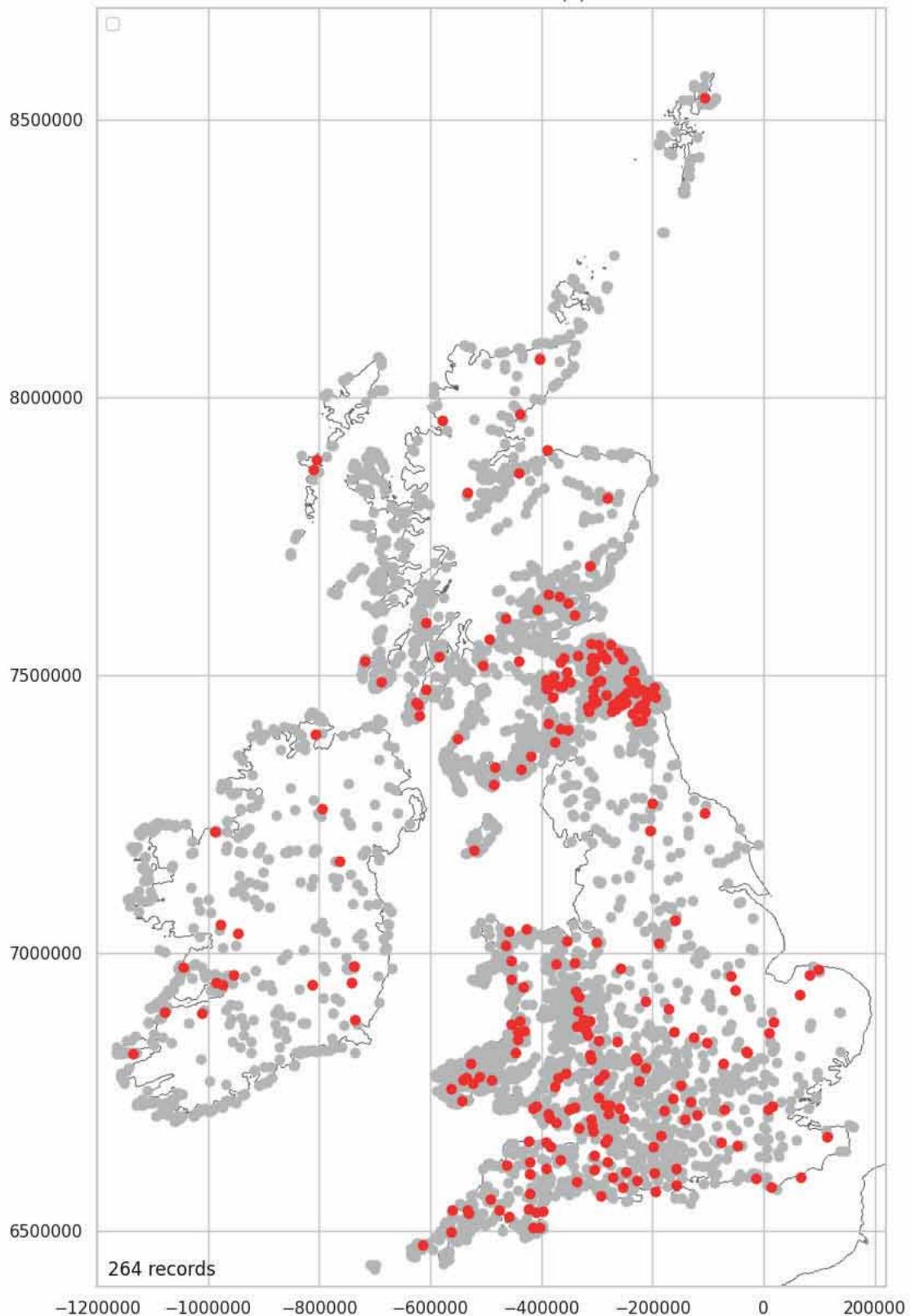
Three entrance forts show a similar distribution to two entrance forts except the focus, in Wales, is now toward the eastern side of the Brecon Beacons. What appears to be a hole at the centre of the Northeast data cluster reflects the local topography with the highlighted forts sitting on the higher ground and the void being the lowland of the Tweed Basin.

```
In [ ]: three_entrances = \
location_entrance_data[location_entrance_data['Entrances_Breaks'] == 3].copy()
three_entrances['Entrances_Breaks'] = "Yes"
```

```
In [ ]: print(f'{round(len(three_entrances)/len(location_entrance_data)*100, 2)}% of hillforts have three entrances.')
6.37% of hillforts have three entrances.
```

```
In [ ]: three_entrances_stats = \
plot_over_grey(three_entrances, 'Entrances_Breaks', 'Yes', '(3)')
```

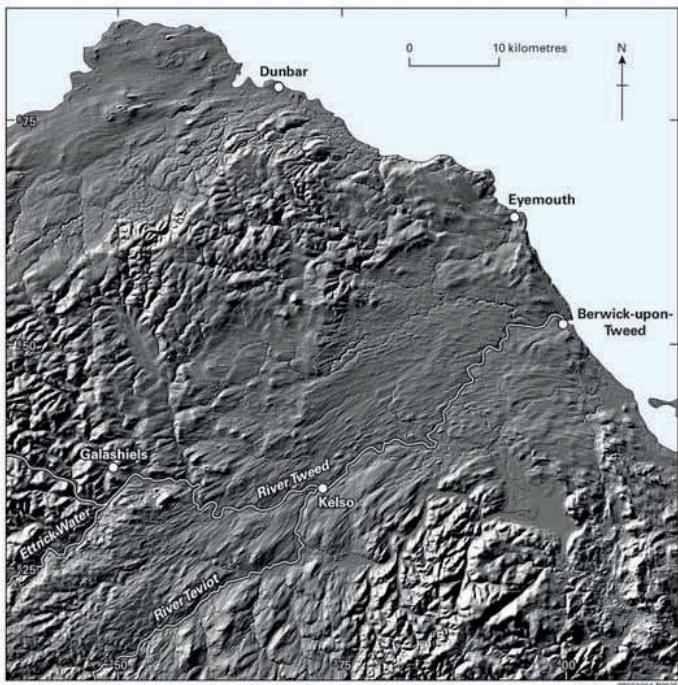
Entrances Breaks (3)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 37%



The Tweed Basin

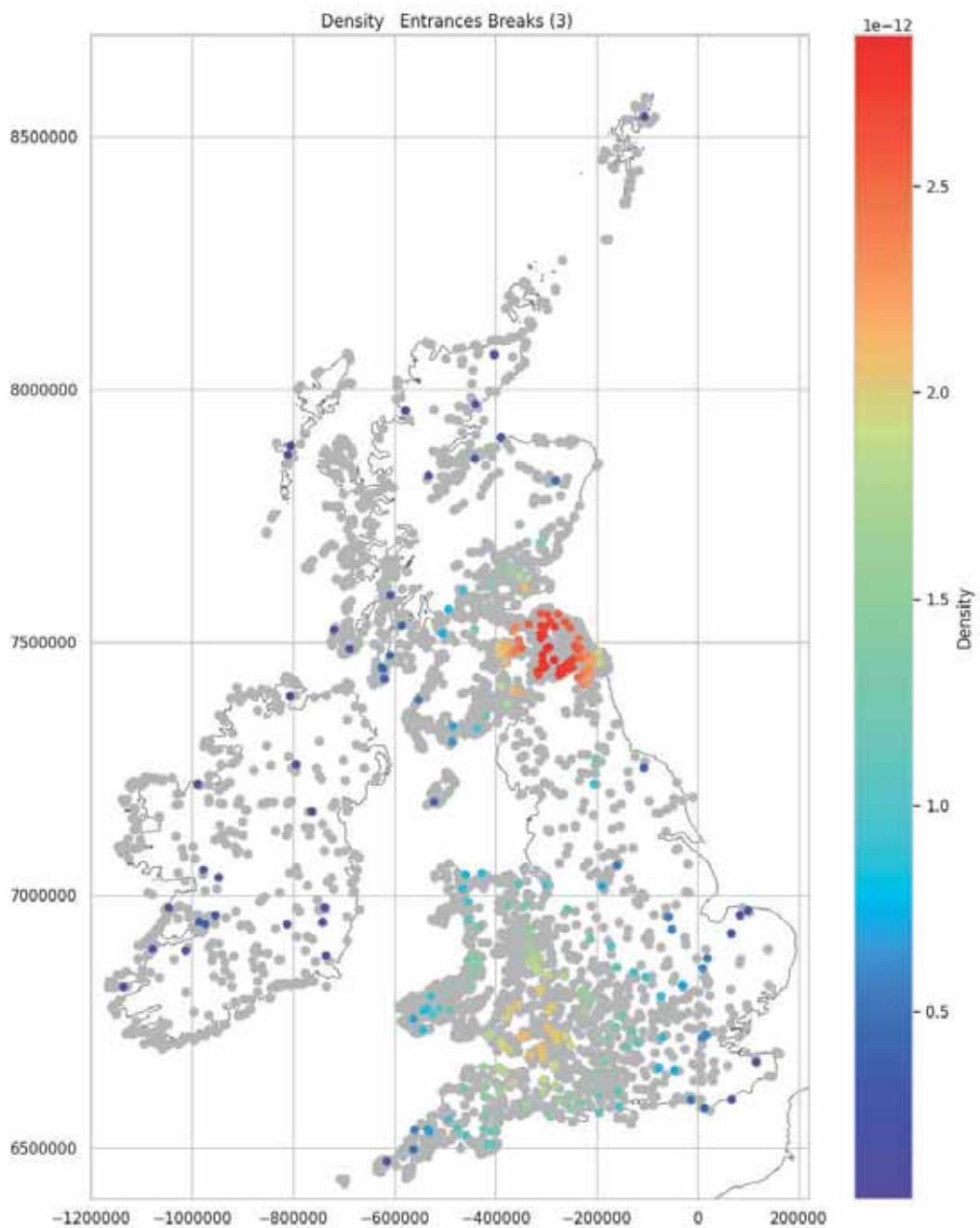
Copyright: British Geological Survey (P912371)

(For use in private study or research for a non-commercial purpose)

Three Entrance Breaks Density Mapped

The density of three entrance hillforts shows a focus over the Northeast. In the south, the distribution is sparse and here the focus of the cluster is toward the River Severn. There are very few of this type out with these two clusters.

```
In [ ]: plot_density_over_grey(three_entrances_stats, 'Entrances_Breaks (3)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

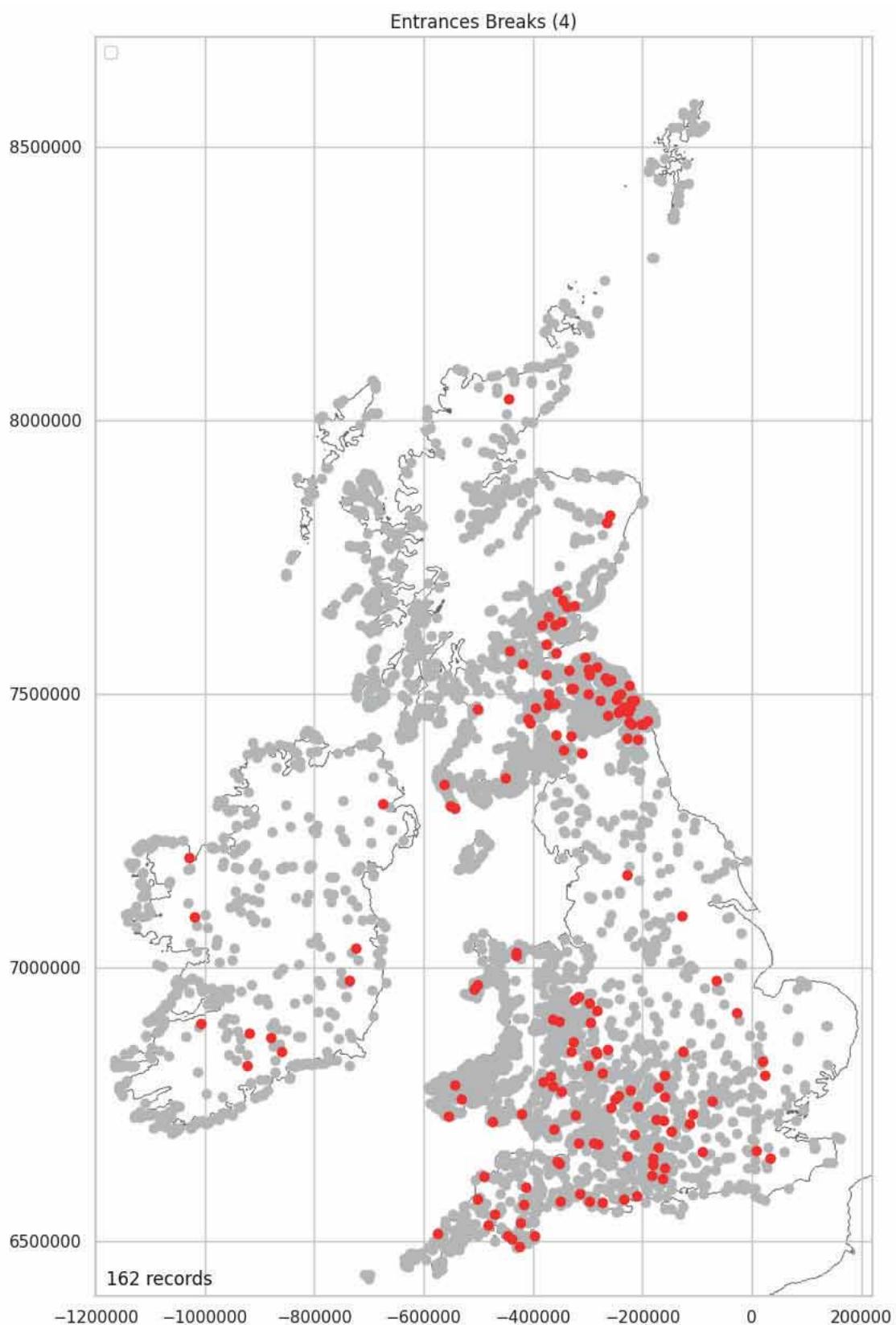
Four Entrance Breaks Distribution Mapped

Four entrance forts are almost exclusively located in the Northeast and south central England.

```
In [ ]: four_entrances = \
location_entrance_data[location_entrance_data['Entrances_Breaks'] == 4].copy()
four_entrances['Entrances_Breaks'] = "Yes"
```

```
In [ ]: print(f'{round(len(four_entrances)/len(location_entrance_data)*100, 2)}% of hillforts have four entrances.')
3.91% of hillforts have four entrances.
```

```
In [ ]: four_entrances_stats = \
plot_over_grey(four_entrances, 'Entrances_Breaks', 'Yes', '(4)')
```



Middleton, M. 2024, Hillforts Primer

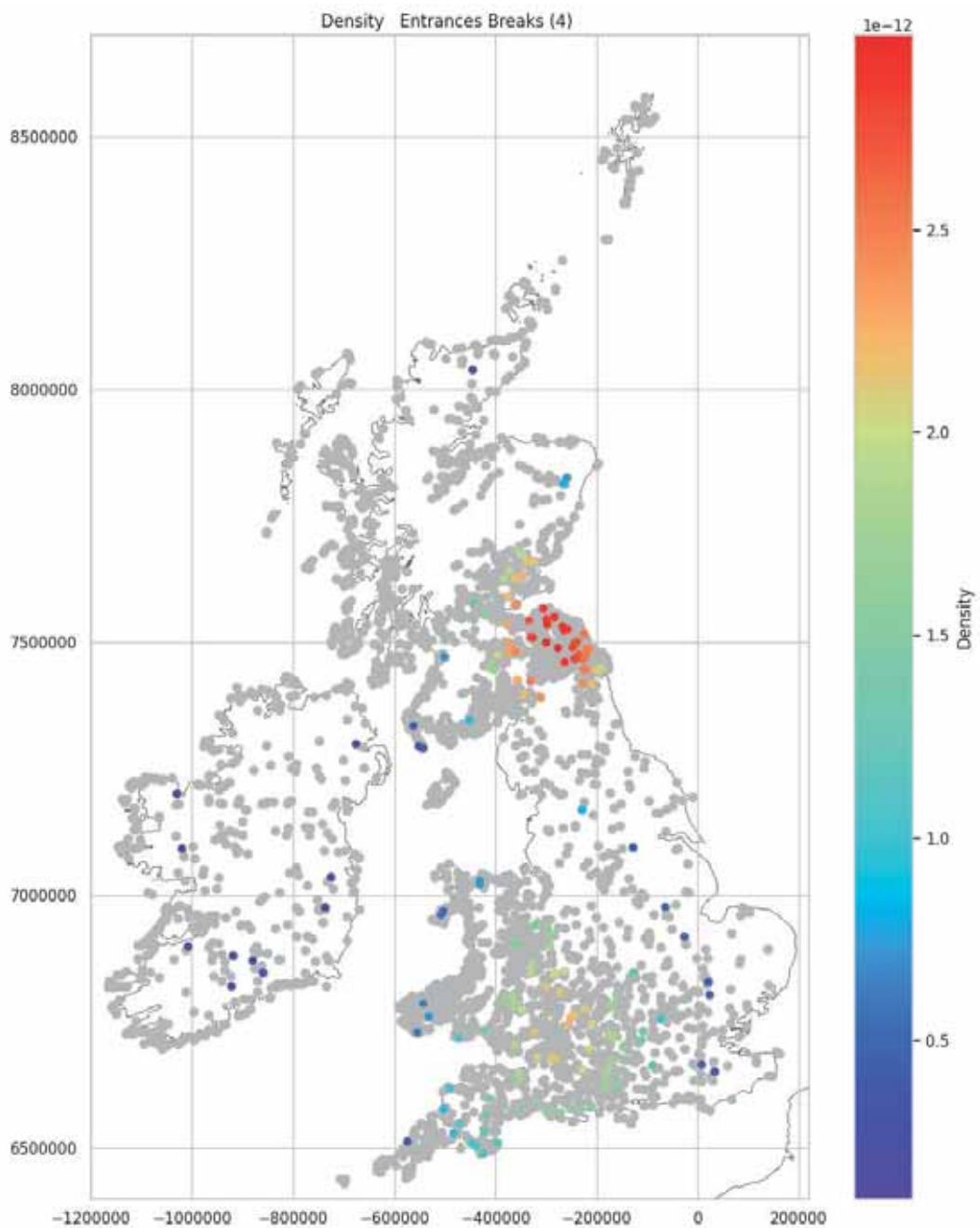
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3.91%

Four Entrance Breaks Density Mapped

The Northeast is the primary focus for four entrance forts. In the South, there is a slight cluster around the River Severn. See [Three Entrance Breaks Density Mapped](#).

```
In [ ]: plot_density_over_grey(four_entrances_stats, 'Entrances_Breaks (4)')
```



Middleton, M. 2024, Hillforts Primer

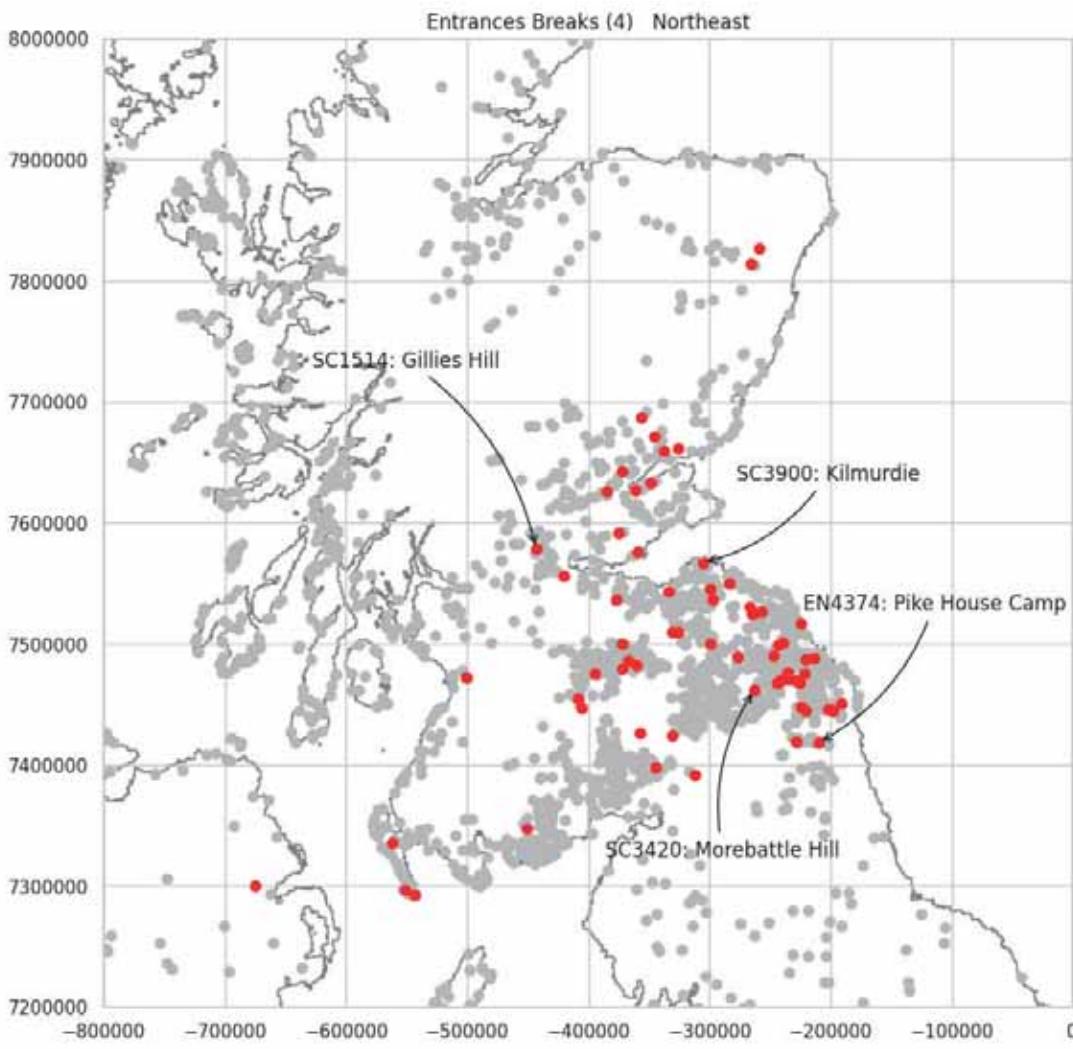
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Four Entrance Breaks Distribution Mapped (Northeast)

In the Northeast, four entrance forts show hints of alignment. One such alignment seems to run from Gillies Hill to Morebattle Hill at roughly 30 to 40 km intervals. Another, less defined alignment, looks to run from Pike House Camp, up toward Kilmurdie. There is also a notable cluster around the mouth of the Tay, just north and south of Perth.

```
In [ ]: four_entrances_ne = \
location_entrance_data_ne[location_entrance_data_ne['Entrances_Breaks'] == 4].copy()
four_entrances_ne['Entrances_Breaks'] = "Yes"
```

```
In [ ]: four_entrances_stats_ne = \
plot_over_grey_north(four_entrances_ne, 'Entrances_Breaks', 'Yes', \
'(4) - Northeast', 'Stirling')
```



```
In [ ]: # This code can be used to get details of hillforts within certain x and y coordinate ranges
# To use this code, first run the document using Runtime > Run all, then remove the '#' from the lines
# starting temp below. Once removed press the Run cell button, on this cell, to the left.
# Update the 'Location_X' & 'Location_Y' values as required.
# temp = pd.merge(name_and_number, four_entrances_stats_ne, left_index=True, right_index=True)
# temp = temp[temp['Location_X'].between(-400000, -300000)]
# temp = temp[temp['Location_Y'].between(760000, 770000)]
# temp
```

```
In [ ]: dist = int(np.sqrt((-443187 - -419817)**2 + (7578896 - 7556141)**2))
dist
```

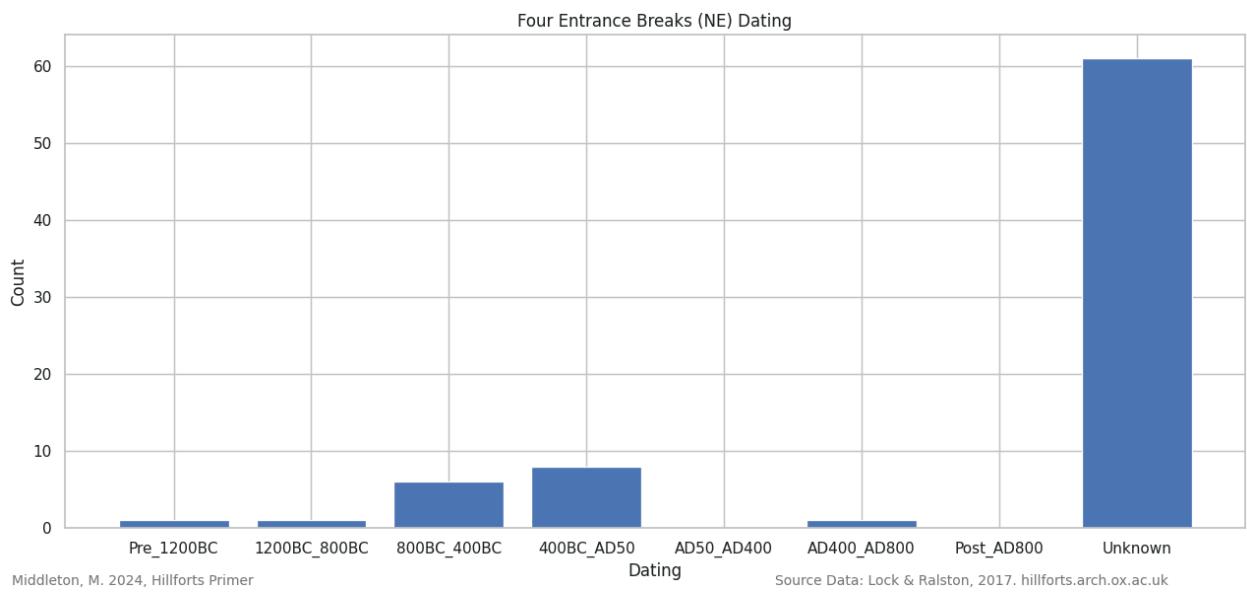
```
Out[ ]: 32618
```

Four Entrance Breaks Dating (Northeast)

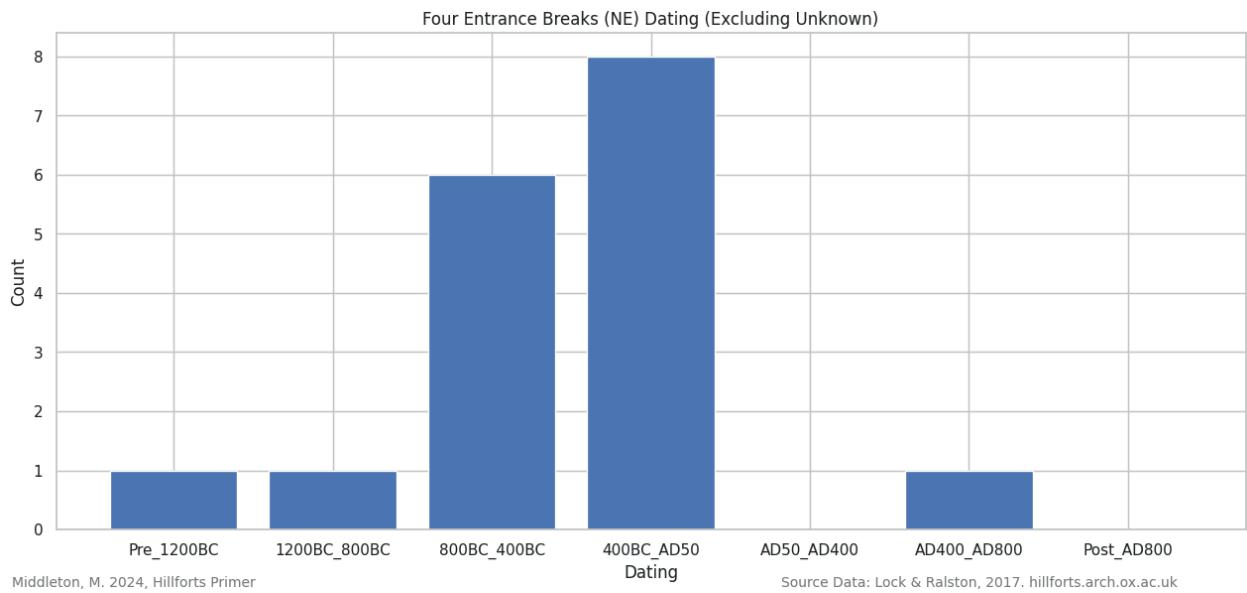
It was considered that the alignment of four entrance forts, and there being a possible relationship between them, might hint at these forts having a different period of construction. In terms of dating the majority of the four entrance breaks hillforts in the Northeast are undated. Of those that are, almost all have dates ranging between 800BC to AD50. There is an interesting lack of dates in the range AD50 to AD400 although it is important to note that the total count of dates is very low and the general distribution of dates is in line with those seen for all hillforts. There is no dating evidence to suggest these forts are related to a different period of construction or reuse.

```
In [ ]: four_entrances_ne_dates = \
pd.merge(four_entrances_ne, date_data, left_index=True, right_index=True)
```

```
In [ ]: plot_bar_chart(four_entrances_ne_dates[date_features], 2, 'Dating', 'Count', \
'Four Entrance Breaks (NE) Dating')
```



```
In [ ]: plot_bar_chart(four_entrances_ne_dates[date_features], 2, 'Dating', 'Count', \
    'Four Entrance Breaks (NE) Dating (Excluding Unknown)', True)
```



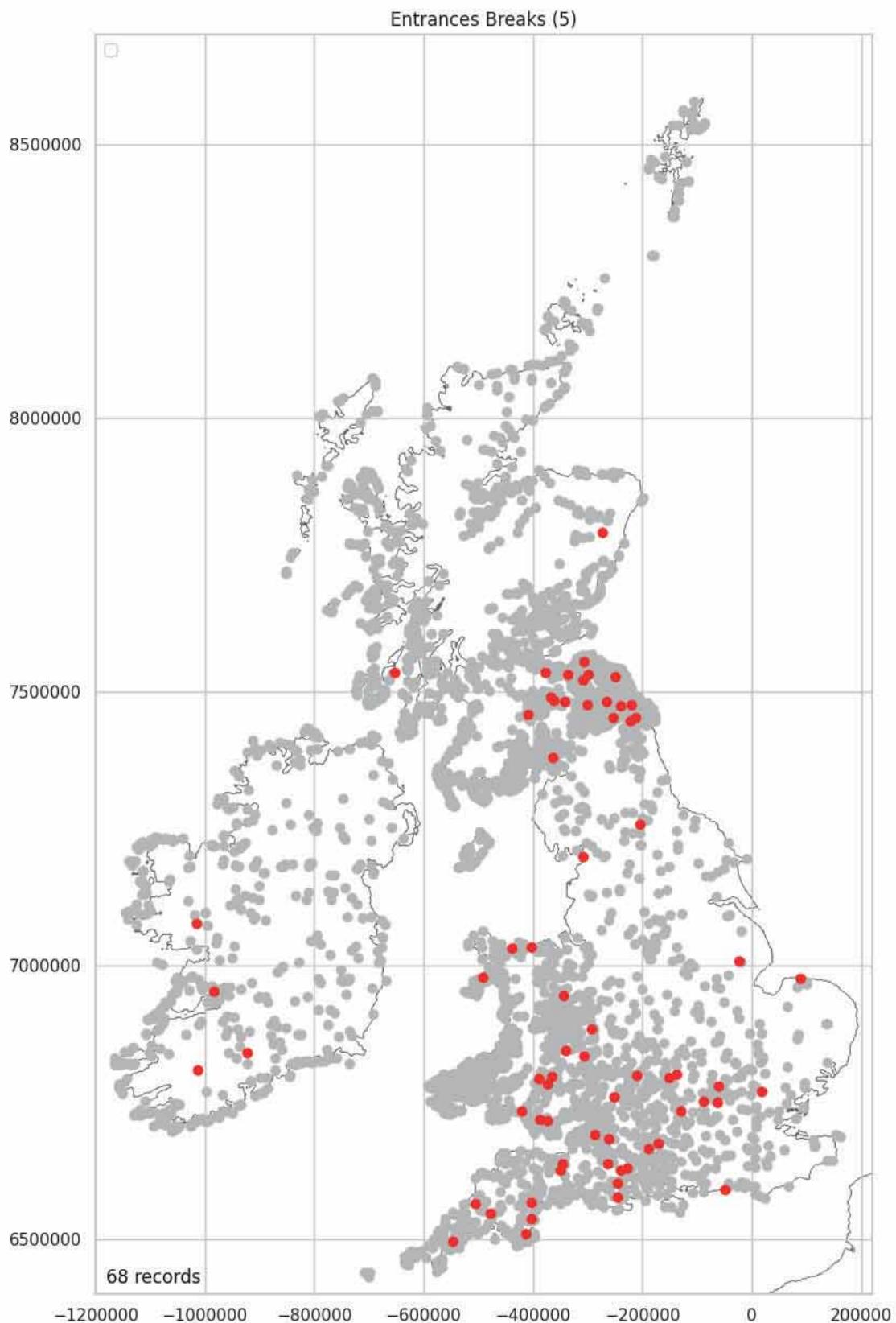
Five Entrance Breaks Distribution Mapped

As with three and four entrances above, the Northeast and south central to south west England are the main areas where hillforts with five entrances cluster.

```
In [ ]: five_entrances = \
location_entrance_data[location_entrance_data['Entrances_Breaks'] == 5].copy()
five_entrances['Entrances_Breaks'] = "Yes"
```

```
In [ ]: print(f'{round(len(five_entrances)/len(location_entrance_data)*100, 2)}% of hillforts have five entrances.')
1.64% of hillforts have five entrances.
```

```
In [ ]: five_entrances_stats = \
plot_over_grey(five_entrances, 'Entrances_Breaks', 'Yes', '(5)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 64%

Entrance Breaks Not Recorded Distribution Mapped

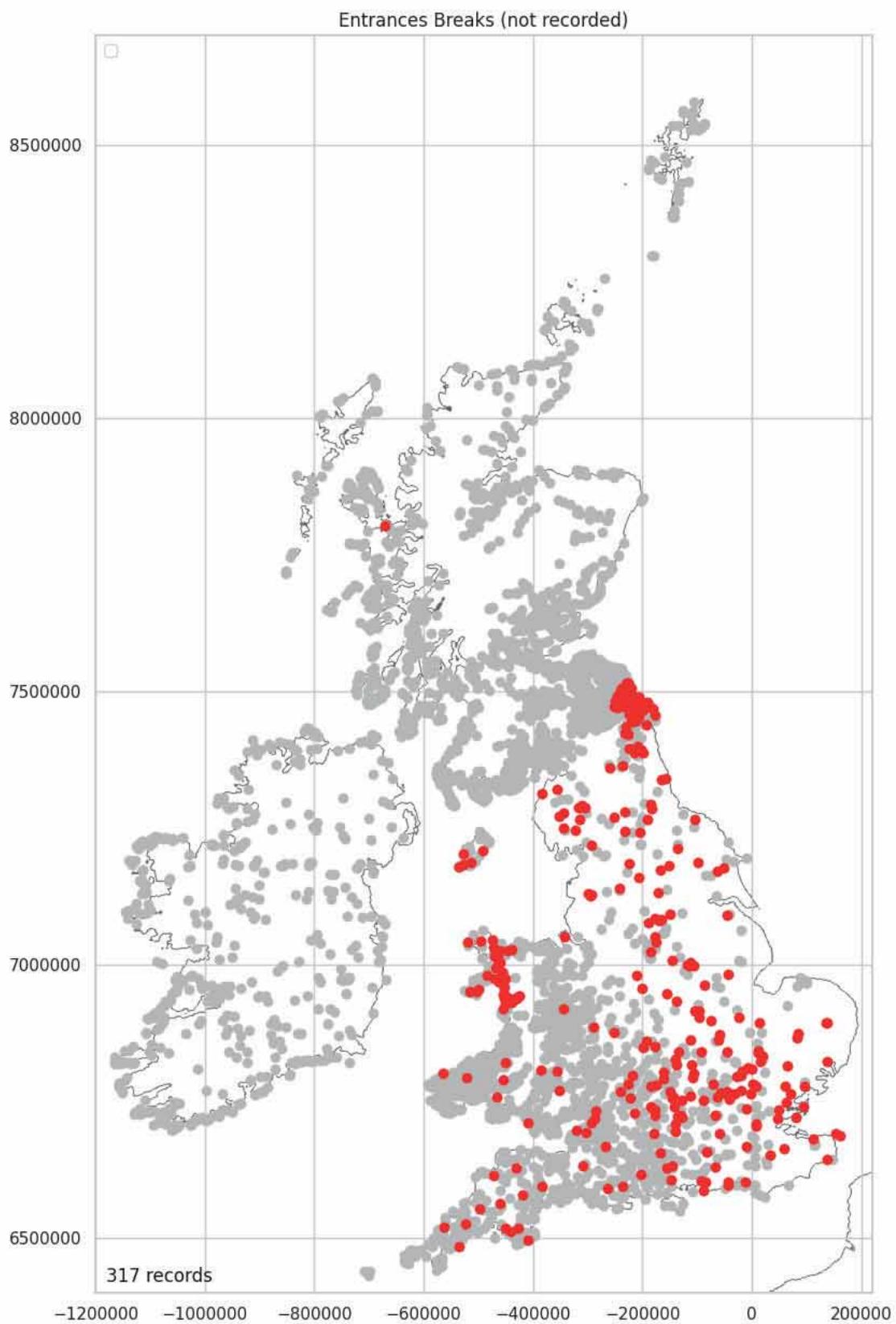
All but one fort in the Northwest and a couple in the Isle of Man, are in England and Wales.

```
In [ ]: mlnus_one_entrances = \
location_entrance_data[location_entrance_data['Entrances_Breaks'] == -1].copy()
mlnus_one_entrances['Entrances_Breaks'] = "Yes"
```

```
In [ ]: print(f'{round(len(mlnus_one_entrances)/len(location_entrance_data)*100, 2)}% of hillforts have no information record')
```

7.64% of hillforts have no information recorded regarding entrances.

```
In [ ]: minus_one_entances_stats = \
plot_over_grey(minus_one_entances, 'Entrances_Breaks', 'Yes', '(not recorded)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

7.64%

Entrances Original Data Plotted

Entrance Original has a long tail of outliers. 95.44% of hillforts have two original entrances or less. 80.23% have one or less. Only 1.69% of hillforts have four entrances or more.

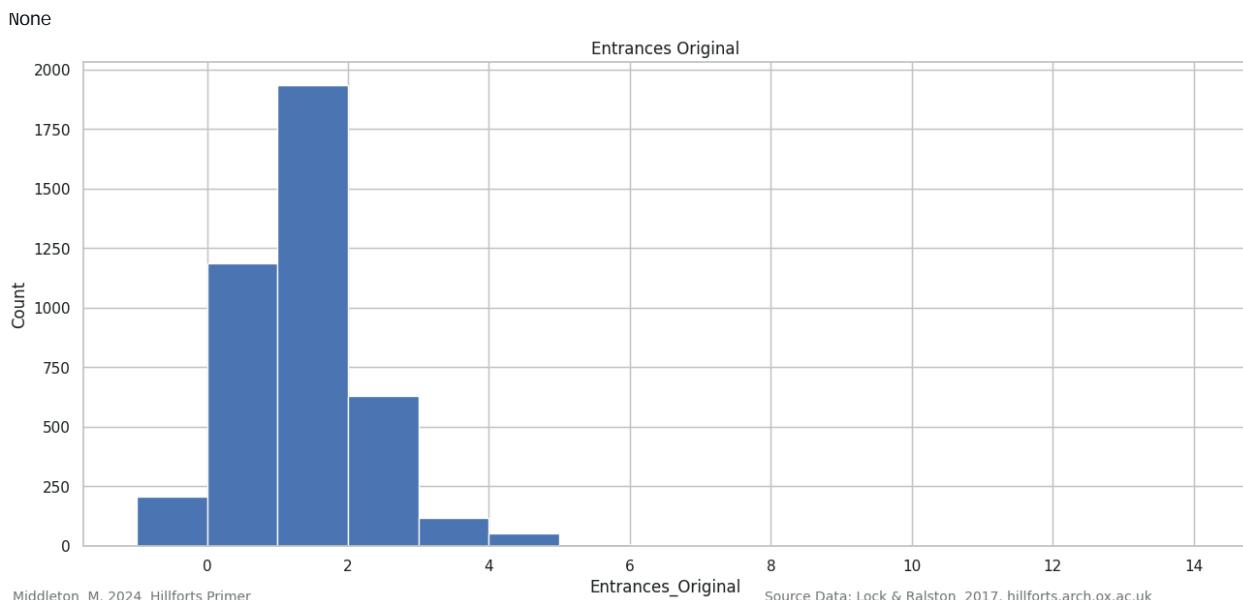
```
In [ ]: entrance_numeric_data['Entrances_Original'].value_counts().sort_index()
```

```
Out[ ]: -1.0    206
         0.0   1186
         1.0   1935
         2.0    631
         3.0    119
         4.0     53
         5.0      8
         6.0      2
         7.0      2
         8.0      1
         9.0      2
        12.0      1
        14.0      1
Name: Entrances_Original, dtype: int64
```

```
In [ ]: one_orig_ent = entrance_numeric_data[entrance_numeric_data['Entrances_Original']==1]
two_orig_ent_or_less = entrance_numeric_data[entrance_numeric_data['Entrances_Original']<=2] - 206
three_orig_ent_or_less = entrance_numeric_data[entrance_numeric_data['Entrances_Original']<=3] - 206
outlier_orig_ent = entrance_numeric_data[entrance_numeric_data['Entrances_Original']>3] - 206
print(f'{round(len(one_orig_ent)/len(location_entrance_data)*100,2)}% of hillforts have one original entrance.')
print(f'{round(len(two_orig_ent_or_less)/len(location_entrance_data)*100,2)}% of hillforts have two original entrances')
print(f'{round(len(three_orig_ent_or_less)/len(location_entrance_data)*100,2)}% of hillforts have three original entrances')
print(f'{round(len(outlier_orig_ent)/len(location_entrance_data)*100,2)}% of hillforts have four or more original entrances')

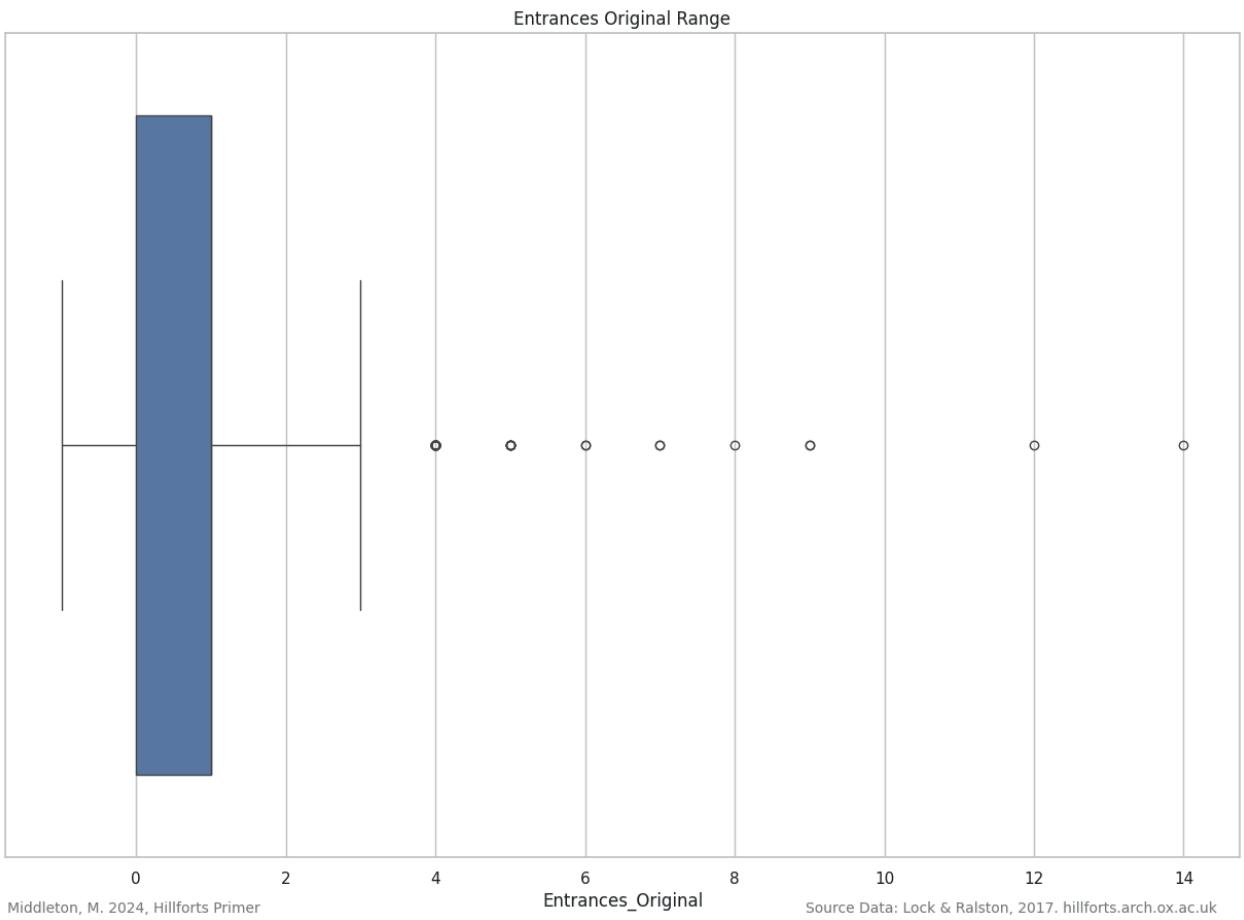
46.66% of hillforts have one original entrance.
95.44% of hillforts have two original entrances or less.
98.31% of hillforts have three original entrances or less.
1.69% of hillforts have four or more original entrances (Outliers).
```

```
In [ ]: plot_histogram(entrance_numeric_data['Entrances_Original'], \
                      'Entrances_Original', 'Entrances_Original')
```



Outliers range from four to 14 original entrances. The interquartile range is between zero and one original entrances.

```
In [ ]: entrances_original_data = \
plot_data_range(entrance_numeric_data['Entrances_Original'], \
                 'Entrances_Original', "h")
```



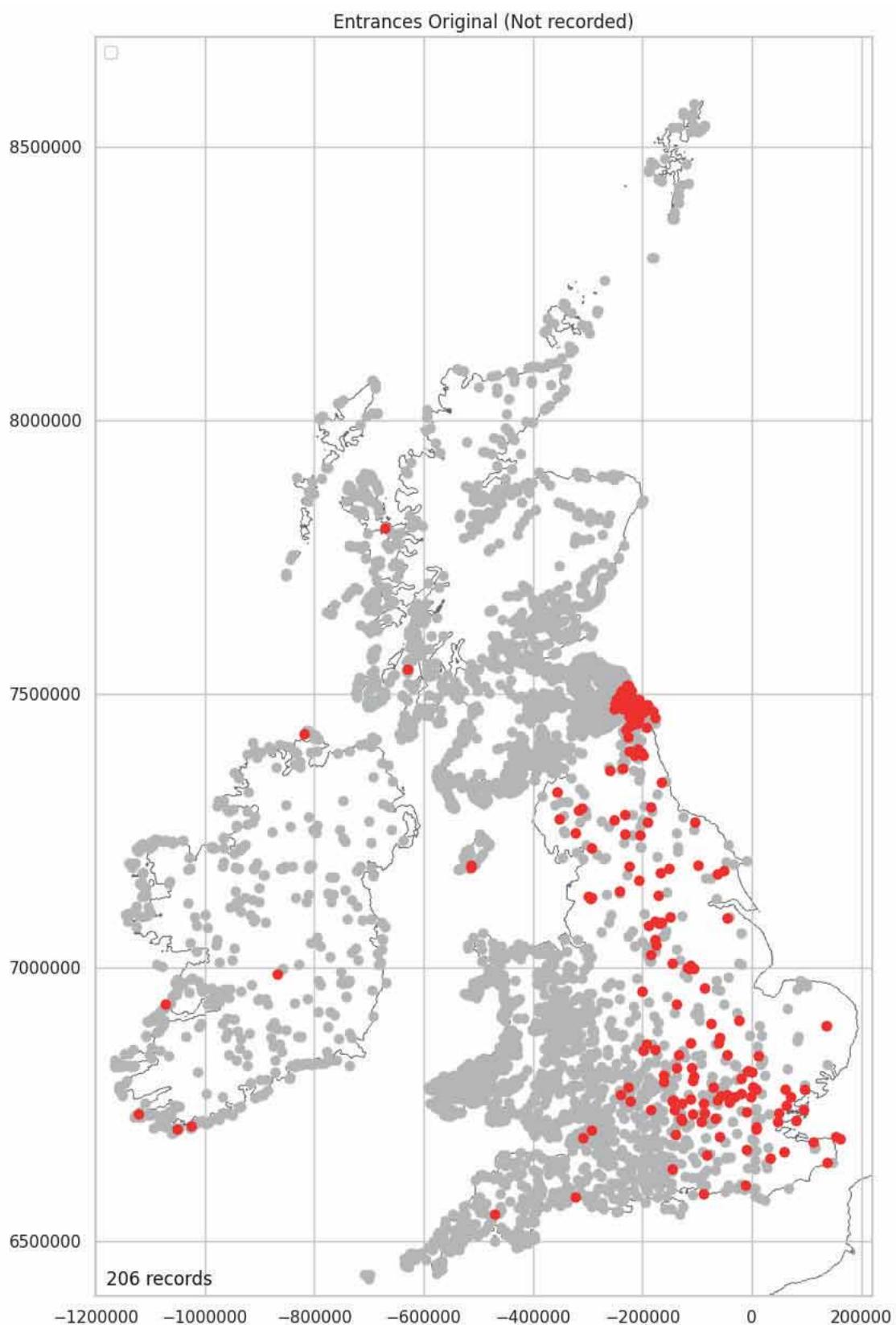
```
In [ ]: entrances_original_data
```

```
Out[ ]: [-1.0, 0.0, 1.0, 1.0, 3.0]
```

Entrance Original Not Recorded Distribution Mapped

There is a recording bias, in the original entrances data, across England. In England, the focus for recording this data has been in the west and south-west. Only 206 records have no information regarding original entrances and almost all are in the east. All hillforts in Wales and most in Scotland and Ireland have a recorded number of original entrances.

```
In [ ]: nan_orig_entrance = \
location_entrance_data[location_entrance_data['Entrances_Original'] == -1].copy()
nan_orig_entrance['Entrances_Original'] = "Yes"
nan_orig_entrances_stats = plot_over_grey(nan_orig_entrance, \
    'Entrances_Original', 'Yes', \
    '(Not recorded)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

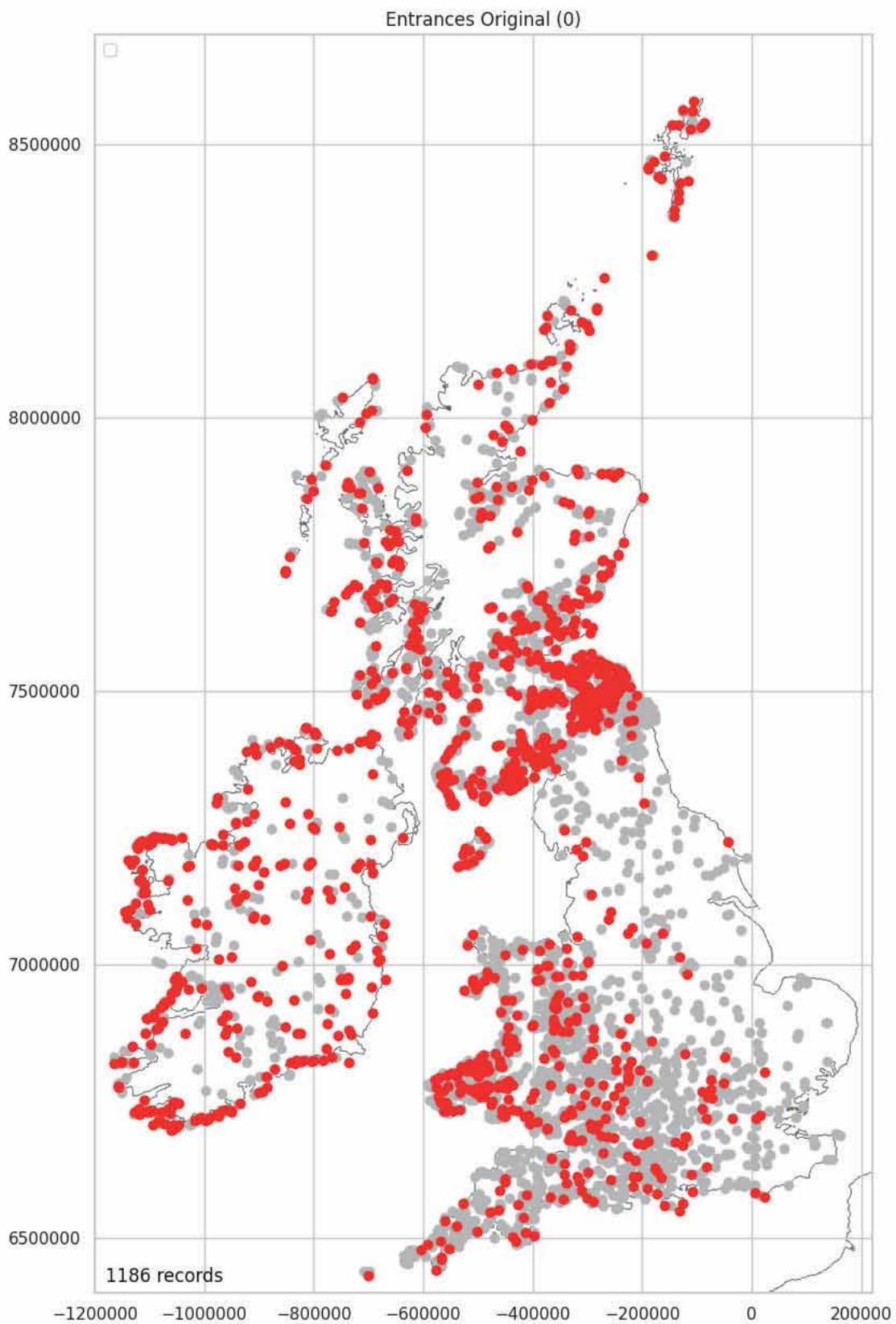
4. 97%

Zero Entrances Original Distribution Mapped

28.6% of hillforts are recorded as not having an original entrance. There is a noticeable lack of records in the east of England which is most likely the result of original entrances not being recorded. See: [Entrance Original Not Recorded Distribution Mapped](#).

```
In [ ]: zero_orig_entrance = \
location_entrance_data[location_entrance_data['Entrances_Original']==0].copy()
zero_orig_entrance['Entrances_Original'] = "Yes"
```

```
zero_orig_entrances_stats = \
plot_over_grey(zero_orig_entrance, 'Entrances_Original', 'Yes', '(0)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

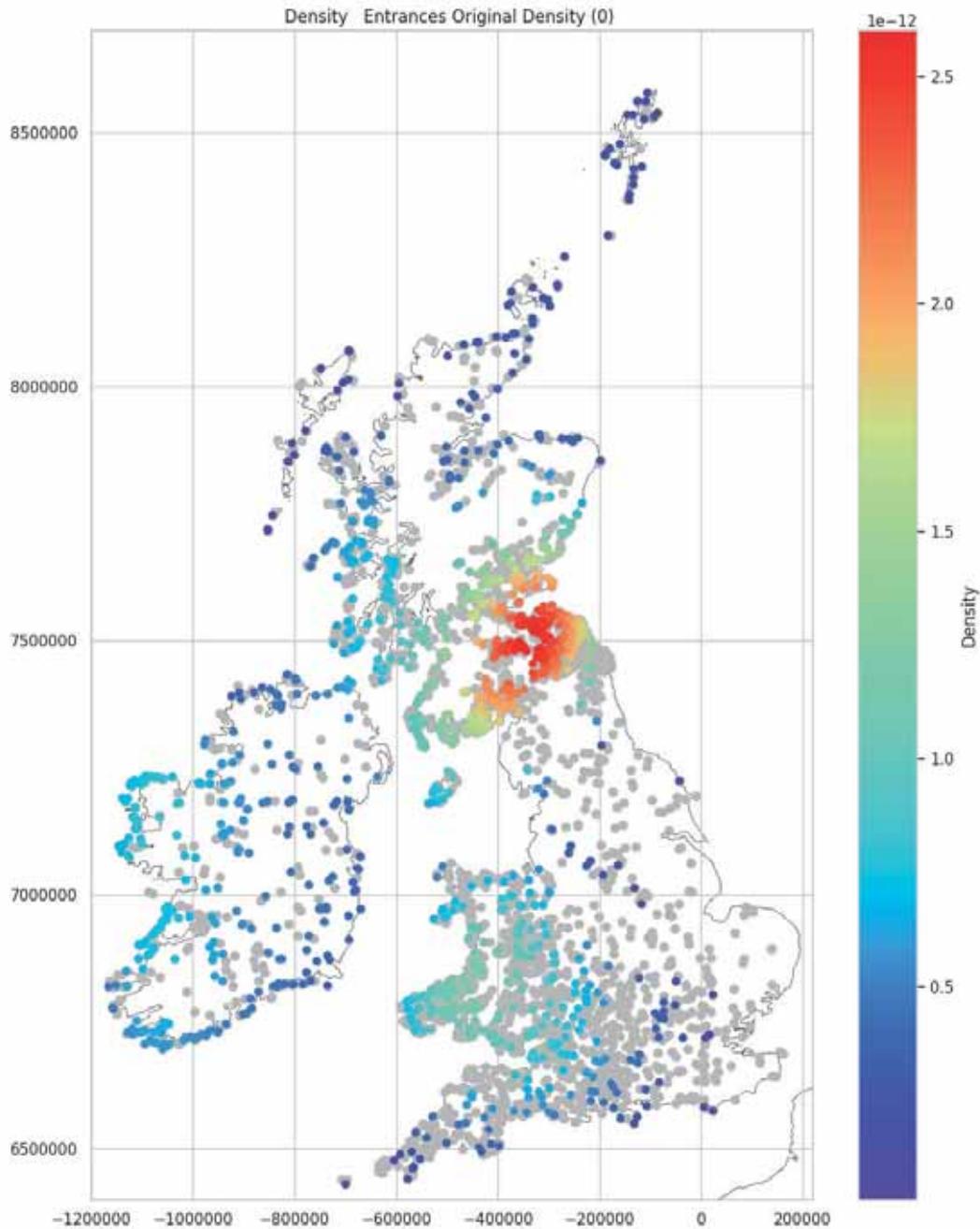
28.6%

Zero Entrance Original Density Mapped

There is a high degree of similarity between the distribution of hillforts with zero entrances and the distribution clusters seen when plotting the location data in, Part 1: Density Data Transformed Mapped. This may suggest that recording a hillfort as having no original entrances has been used as a shorthand to indicate that this information has not been recorded. If not, it suggests that

there is a uniform pattern, across the entire atlas, where original entrances leave no evidence of their existence - perhaps modified by later reuse or an entrance style that leaves no discernible trace.

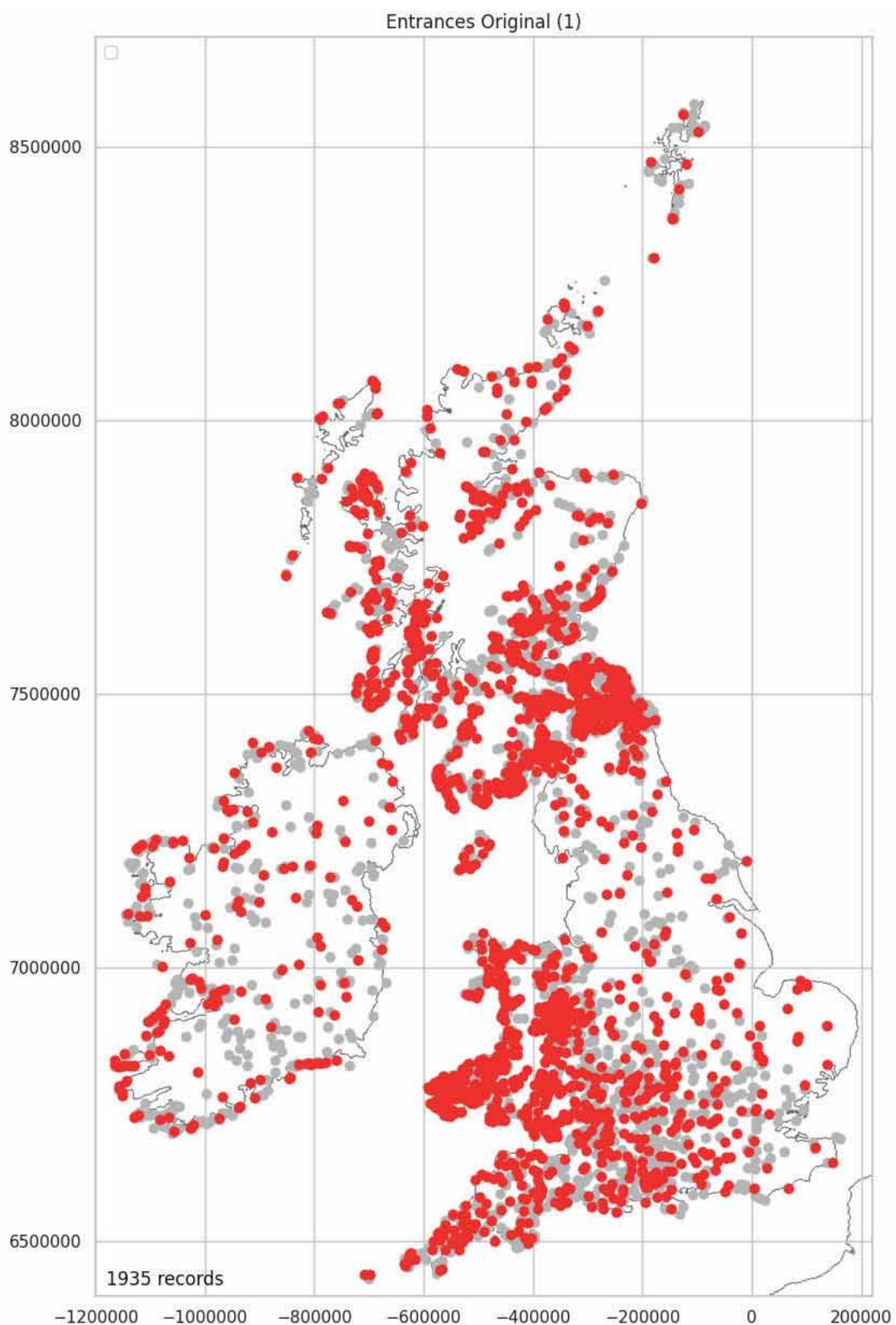
```
In [ ]: plot_densitiy_over_grey(zero_orig_entances_stats, 'Entrances_Original Density (0)')
```



One Entrance Original Distribution Mapped

Just under half (46.66%) of all hillforts have a single original entrance.

```
In [ ]: one_orig_entrance = \
location_entrance_data[location_entrance_data['Entrances_Original']==1].copy()
one_orig_entrance['Entrances_Original'] = "Yes"
one_orig_entances_stats = \
plot_over_grey(one_orig_entrance, 'Entrances_Original', 'Yes', '(1)')
```



Middleton, M. 2024, Hillforts Primer

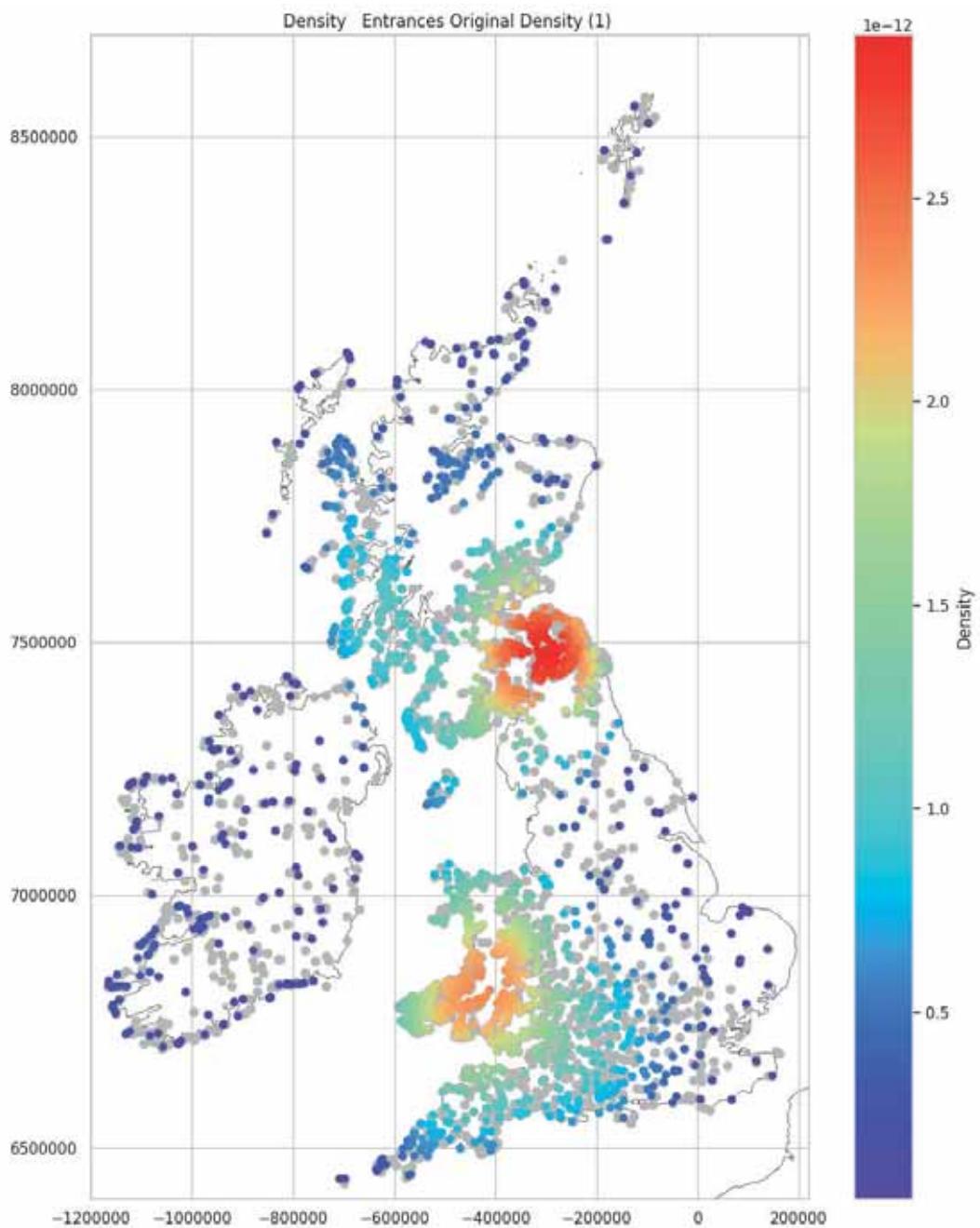
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

46.66%

One Entrance Original Density Mapped

Here the distributions in the Northeast, Northwest, and South match the main distributions seen in, Part 1: Density Data Transformed Mapped. In Ireland there is a sparse spread across the entire country but there is no obvious correlation with the two main clusters of forts, seen on the Density Data Transformed plot.

```
In [ ]: plot_density_over_grey(one_orig_entrances_stats, 'Entrances_Original_Density (1)')
```



Middleton, M. 2024, Hillforts Primer

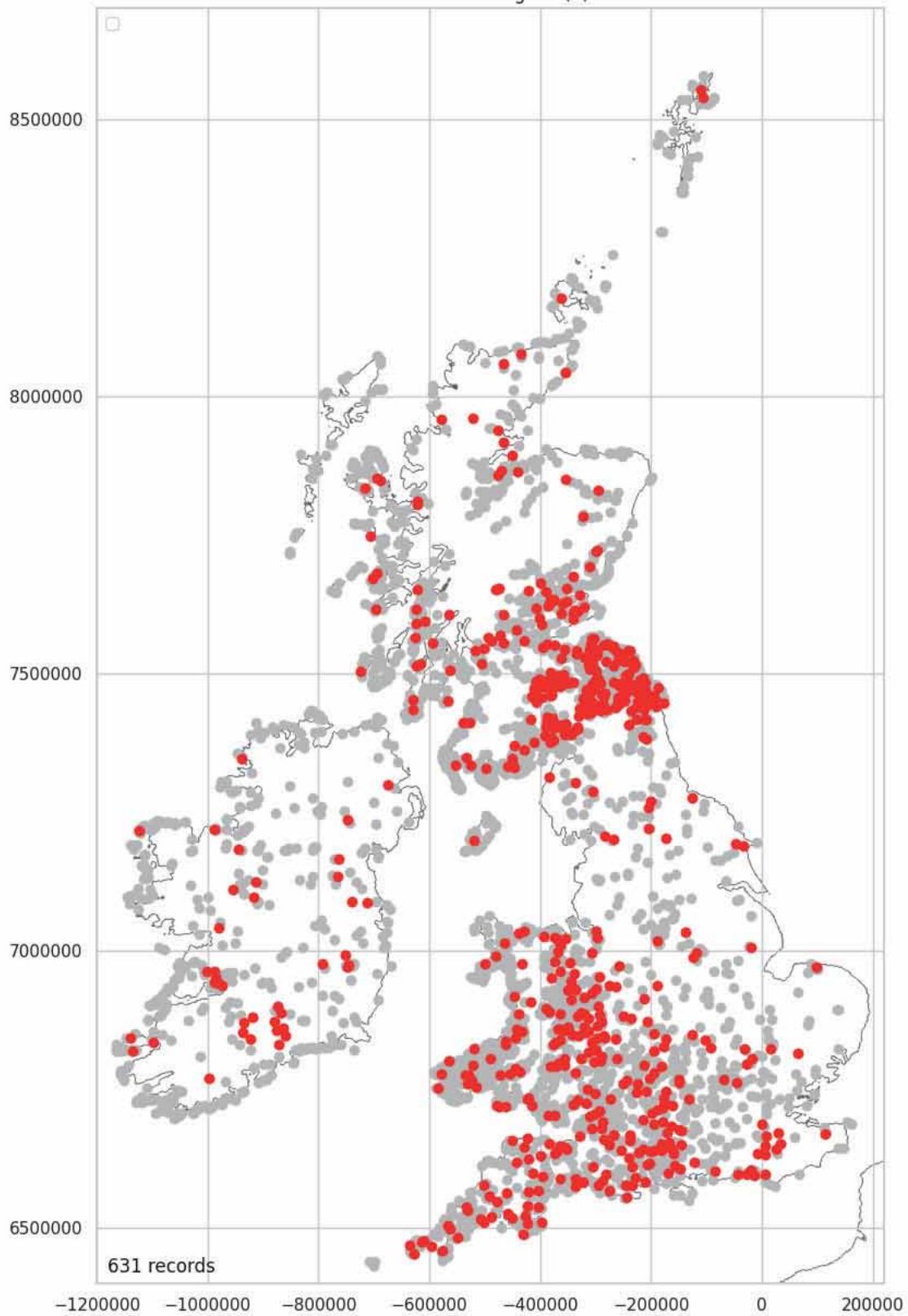
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Two Entrances Original Distribution Mapped

Just 15.22% of hillforts have two original entrances.

```
In [ ]: two_orig_entrance = \
location_entrance_data[location_entrance_data['Entrances_Ori ginal']==2].copy()
two_orig_entrance['Entrances_Ori ginal'] = "Yes"
two_orig_entrances_stats = \
plot_over_grey(two_orig_entrance, 'Entrances_Ori ginal', 'Yes', '(2)')
```

Entrances Original (2)



Middleton, M. 2024, Hillforts Primer

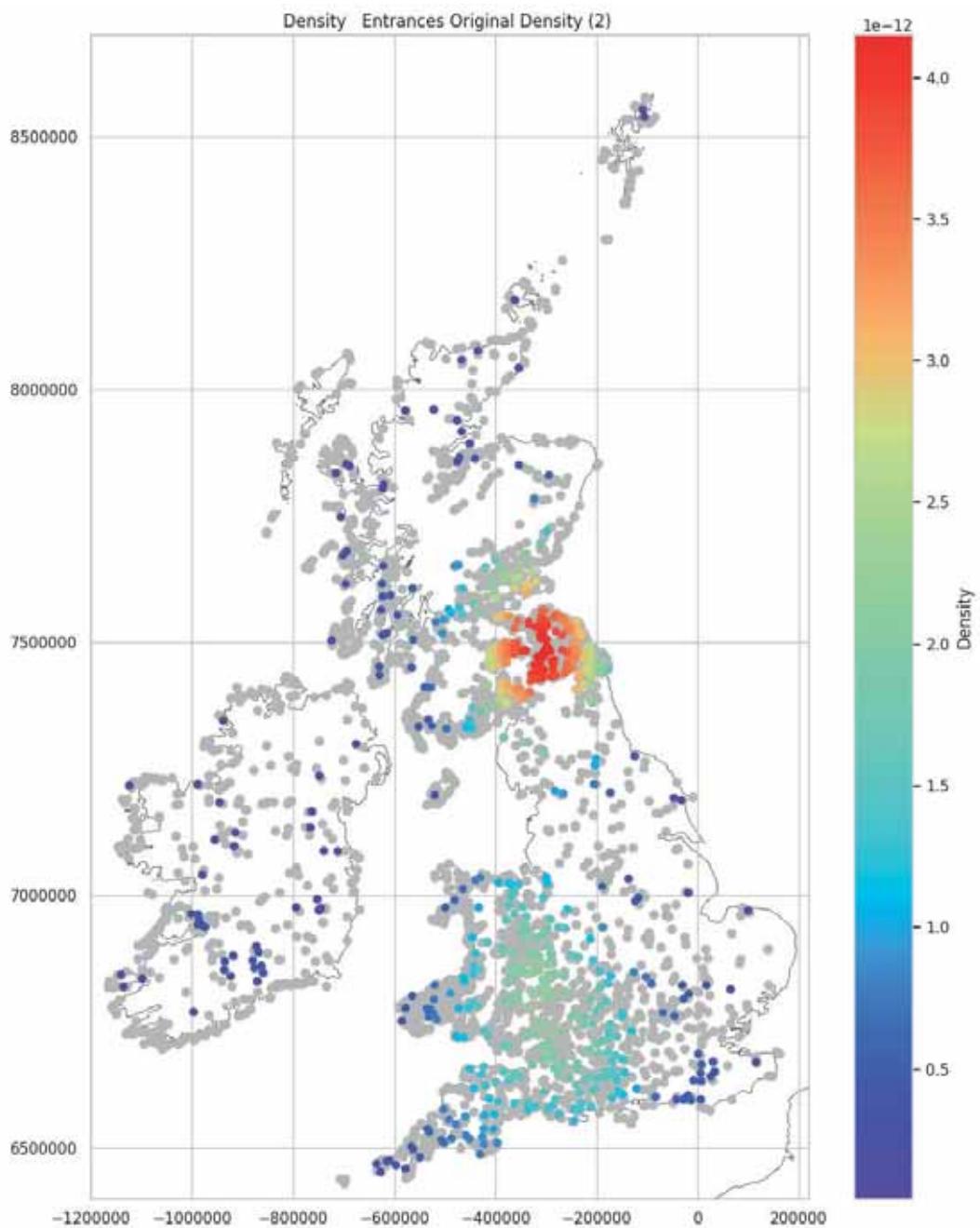
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

15. 22%

Two Entrances Original Density Mapped

There are two main clusters. One in the Northeast and the second to the east of the Cambrian Mountains. The contrast in the intensity and focus of the southern cluster is striking when compared with [One Entrance Original Density Mapped](#), with the two entrance cluster being more diffuse and focussed over the eastern slopes of the Cambrian mountains, into south, central England and down into the South-west.

```
In [ ]: plot_density_over_grey(two_orig_entrances_stats, 'Entrances_Original_Density_(2)')
```



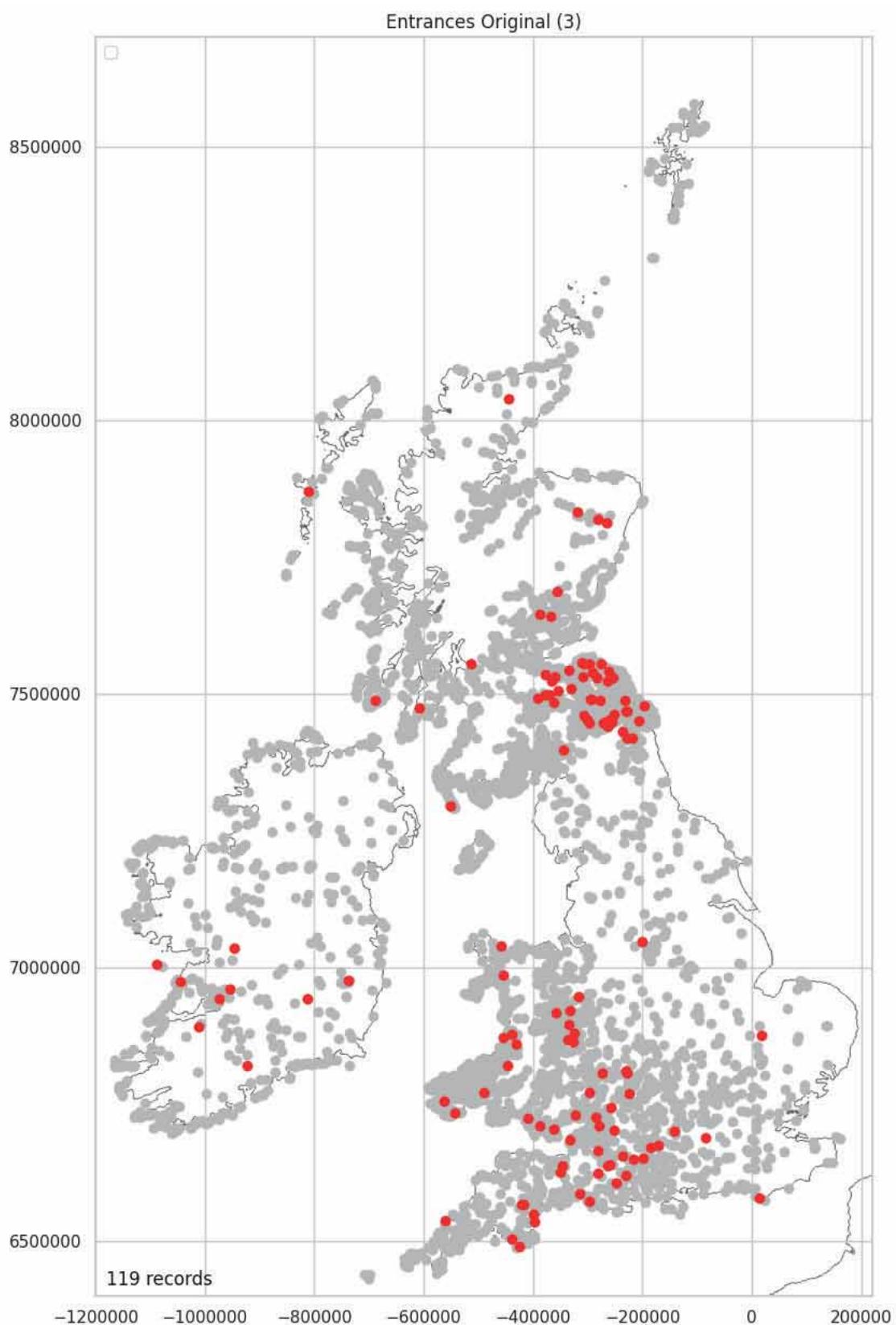
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Three Entrances Original Distribution Mapped

Just 2.87% of hillforts have three original entrances.

```
In [ ]: three_orig_entrance = \
location_entrance_data[location_entrance_data['Entrances_Original']==3].copy()
three_orig_entrance['Entrances_Original'] = "Yes"
three_orig_entrances_stats = \
plot_over_grey(three_orig_entrance, 'Entrances_Original', 'Yes', '(3)')
```



Middleton, M. 2024, Hillforts Primer

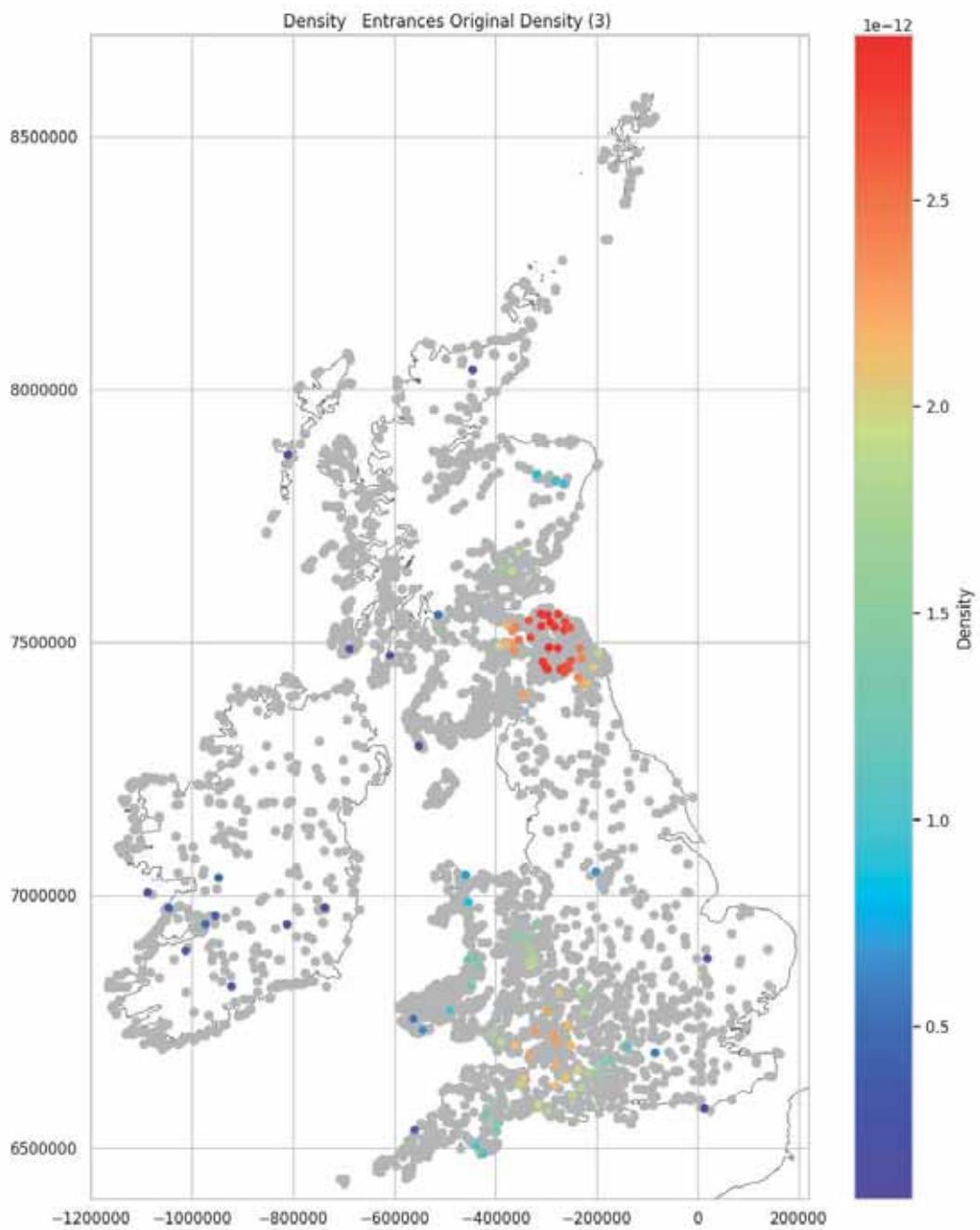
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2.87%

Three Entrances Original Density Mapped

Most of these forts are in the Northeast.

```
In [ ]: plot_density_over_grey(three_orig_entrances_stats, 'Entrances_Original_Density_(3)')
```



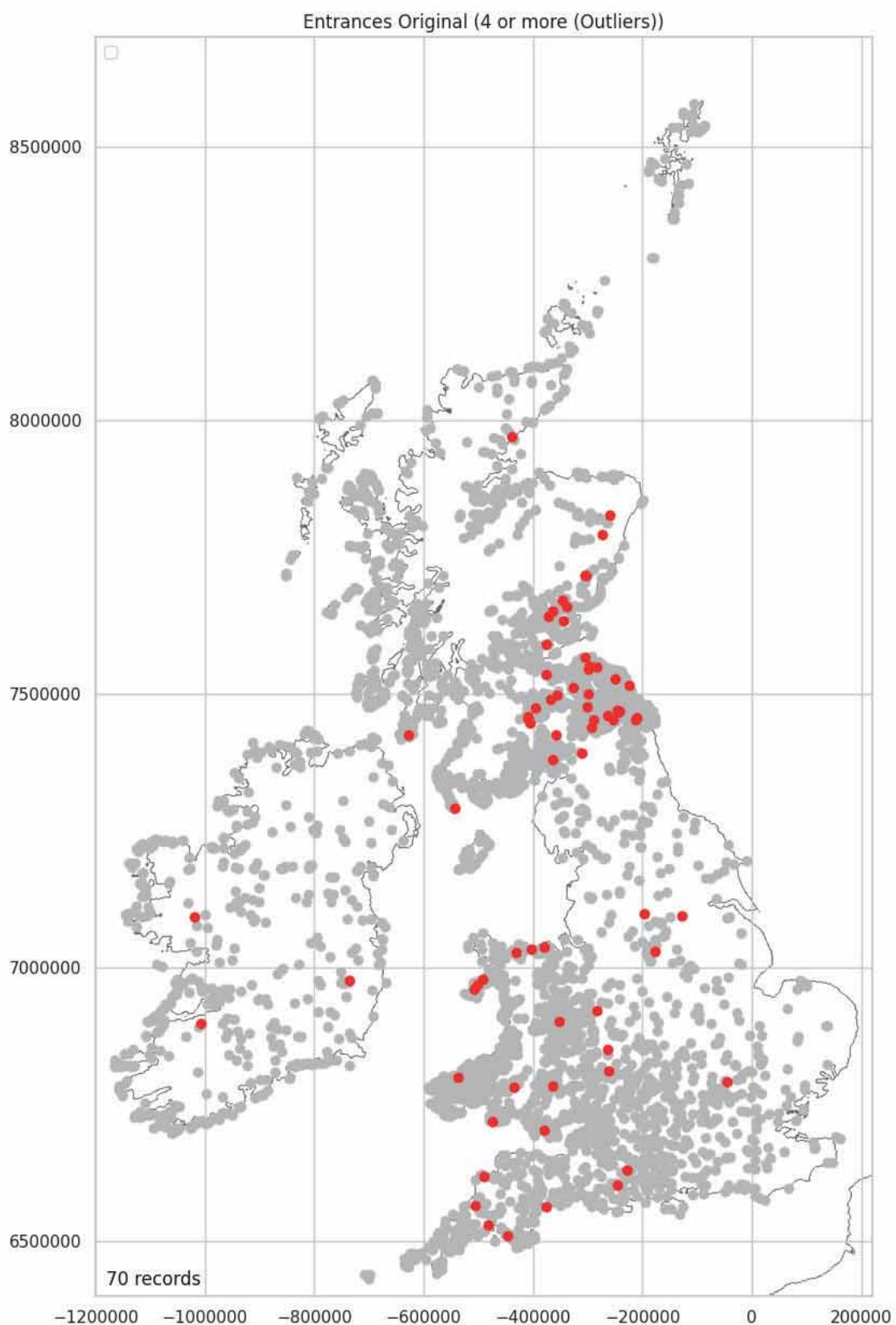
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Four Entrances Original Distribution Mapped

Just 70 hillforts (1.69%) have four original entrances.

```
In [ ]: four_plus_orig_entrance = \
location_entrance_data[location_entrance_data['Entrances_Ori ginal']>3].copy()
four_plus_orig_entrance['Entrances_Ori ginal'] = "Yes"
four_plus_orig_entrances_stats = \
plot_over_grey(four_plus_orig_entrance, 'Entrances_Ori ginal', \
'Yes', '(4 or more (Outliers))')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 69%

Entrance Text Data

There are five text features relating to entrances. All contain null values.

```
In [ ]: entrance_text_features = [
    'Entrances_Breaks_Comments',
    'Entrances_Original_Comments',
    'Entrances_Chevaux_Comments',
    'Entrances_Summary',
```

```
'Related_Entrances']

entrance_text_data = entrance_data[entrance_text_features].copy()
entrance_text_data.head()
```

	Entrances_Breaks_Comments	Entrances_Original_Comments	Entrances_Chevaux_Comments	Entrances_Summary	Related_Entrances
0	Two original and four modern gaps.	Two original inturned entrances at SE and SW c...		NaN	Two original entrances; the SE inturned. The S...
1	N entrance damaged by wagon access and possibl...	S entrance original, that on the NW possibly ...		NaN	Entrances difficult to unravel. The S entrance...
2	Entrances intact	Interesting inturn to N entrance		NaN	The curving N entrance is complex with inturn ...
3	Modern gap to the S.	Off-set entrance on the E. Possibly another to...		NaN	The gaps to the NE and E have been widened and...
4	Probable modern breaks not recorded.	Two entrances are from Phase I and four from P...		NaN	Both Phase I entrances, NE and SW, are out-tur...

```
In [ ]: entrance_text_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Entrances_Breaks_Comments    1193 non-null   object 
 1   Entrances_Original_Comments 1126 non-null   object 
 2   Entrances_Chevaux_Comments  77 non-null   object  
 3   Entrances_Summary          4132 non-null   object  
 4   Related_Entrances         2749 non-null   object  
dtypes: object(5)
memory usage: 162.1+ KB
```

Entrance Text Data - Resolve Null Values

Test for 'NA'.

```
In [ ]: test_cat_list_for_NA(entrance_text_data, entrance_text_features)

Entrances_Breaks_Comments 0
Entrances_Original_Comments 0
Entrances_Chevaux_Comments 0
Entrances_Summary 0
Related_Entrances 0
```

Fill null values with 'NA'.

```
In [ ]: entrance_text_data = \
update_cat_list_for_NA(entrance_text_data, entrance_text_features)
entrance_text_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Entrances_Breaks_Comments    4147 non-null   object 
 1   Entrances_Original_Comments 4147 non-null   object 
 2   Entrances_Chevaux_Comments  4147 non-null   object  
 3   Entrances_Summary          4147 non-null   object  
 4   Related_Entrances         4147 non-null   object  
dtypes: object(5)
memory usage: 162.1+ KB
```

Entrance Encodable Data

There are just two encodeable features. Neither contains null values.

```
In [ ]: entrance_encodeable_features = [
'Entrances_Guard_Chambers',
'Entrances_Chevaux']

entrance_encodeable_data = entrance_data[entrance_encodeable_features].copy()
entrance_encodeable_data.head()
```

Out[]:	Entrances_Guard_Chambers	Entrances_Chevaux
0	No	No
1	No	No
2	No	No
3	No	No
4	No	No

```
In [ ]: entrance_encodeable_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Entrances_Guard_Chambers  4147 non-null   object 
 1   Entrances_Chevaux        4147 non-null   object 
dtypes: object(2)
memory usage: 64.9+ KB
```

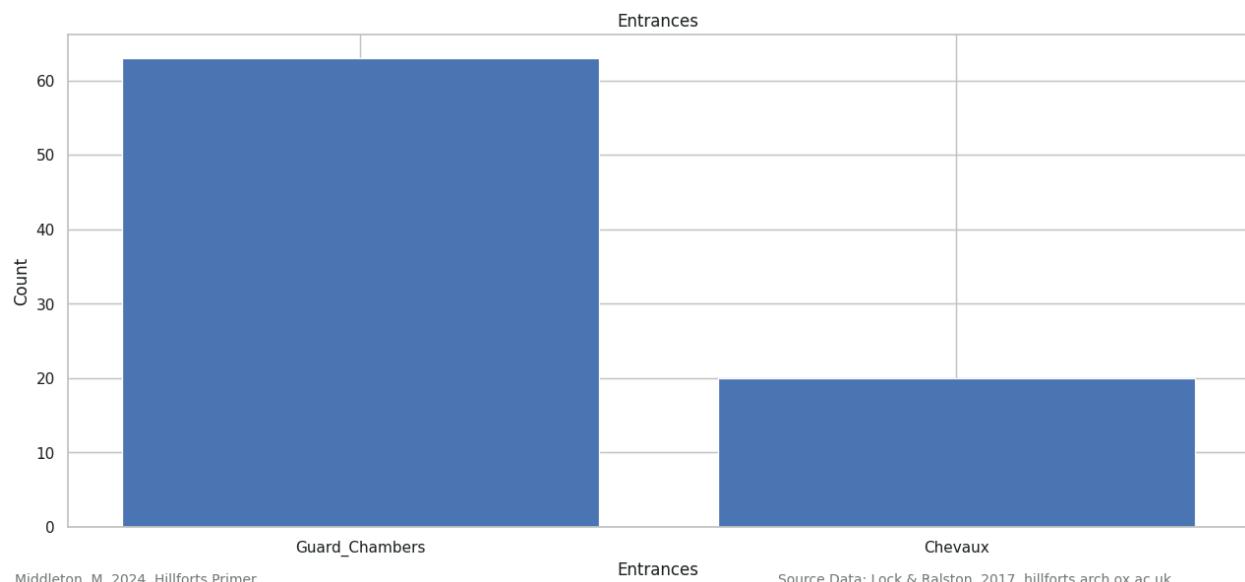
Entrance Encodable Data Plotted

Guard chambers are recorded at 63 hillforts. All but two are in England. Twenty hillforts have a Cheveaux de frise. It is likely that both have a significant survey bias.

```
In [ ]: for feature in entrance_encodeable_features:
    print(feature + ": " + str(sum(entrance_encodeable_data[feature]=="Yes")))

Entrances_Guard_Chambers: 63
Entrances_Chevaux: 20
```

```
In [ ]: plot_bar_chart(entrance_encodeable_data[['Entrances_Guard_Chambers', \
    'Entrances_Chevaux']], 1, 'Entrances', \
    'Count', 'Entrances')
```

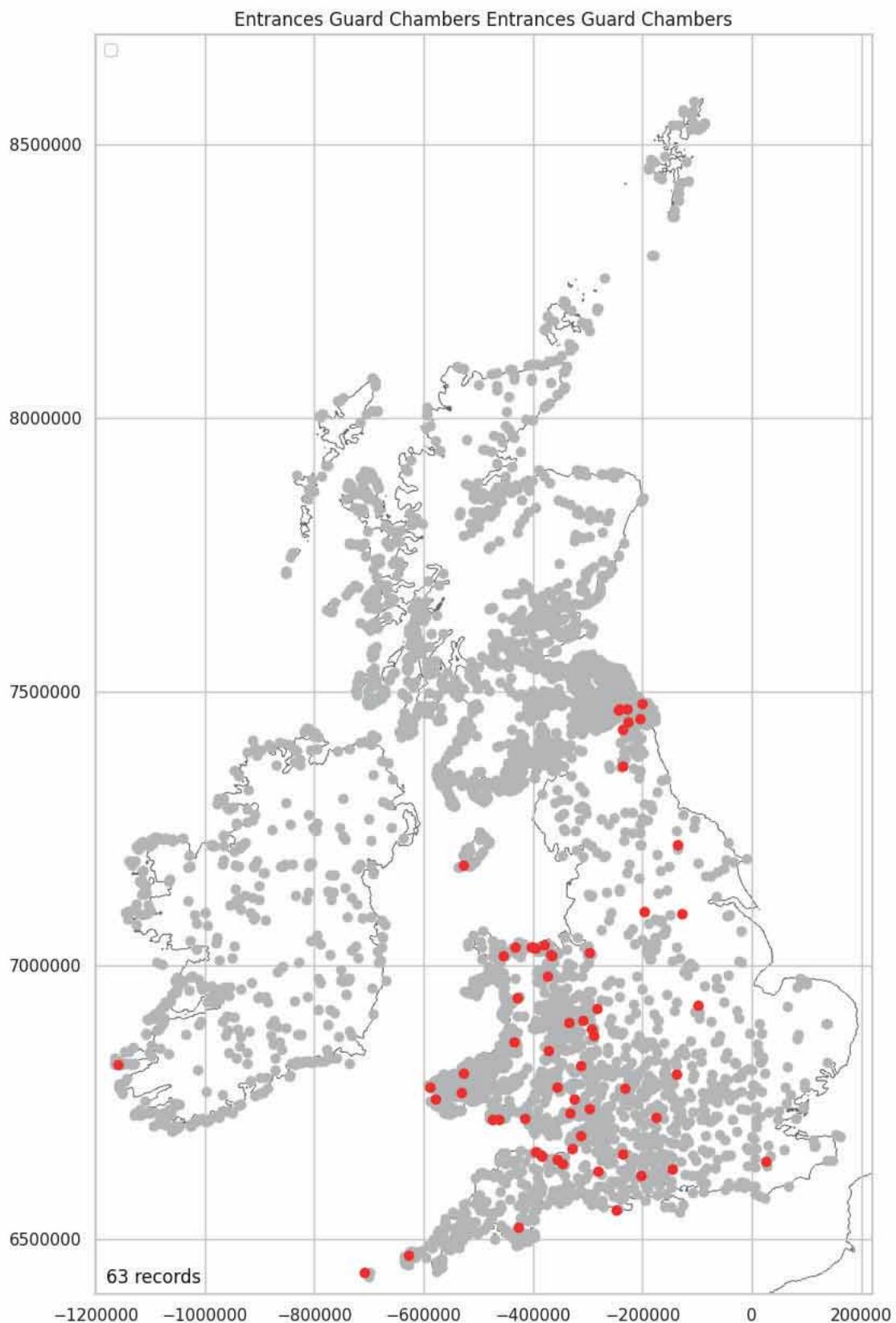


Guard Chambers Mapped

There is a recording bias with all but two of the hillforts recorded being in England and Wales.

```
In [ ]: location_entrance_encodeable_data = \
pd.merge(location_numeric_data_short, entrance_encodeable_data, \
left_index=True, right_index=True)
```

```
In [ ]: entrances_guard_chambers_stats = \
plot_over_grey(location_entrance_encodeable_data, 'Entrances_Guard_Chambers', \
    'Yes', 'Entrances_Guard_Chambers')
```



Middleton, M. 2024, Hillforts Primer

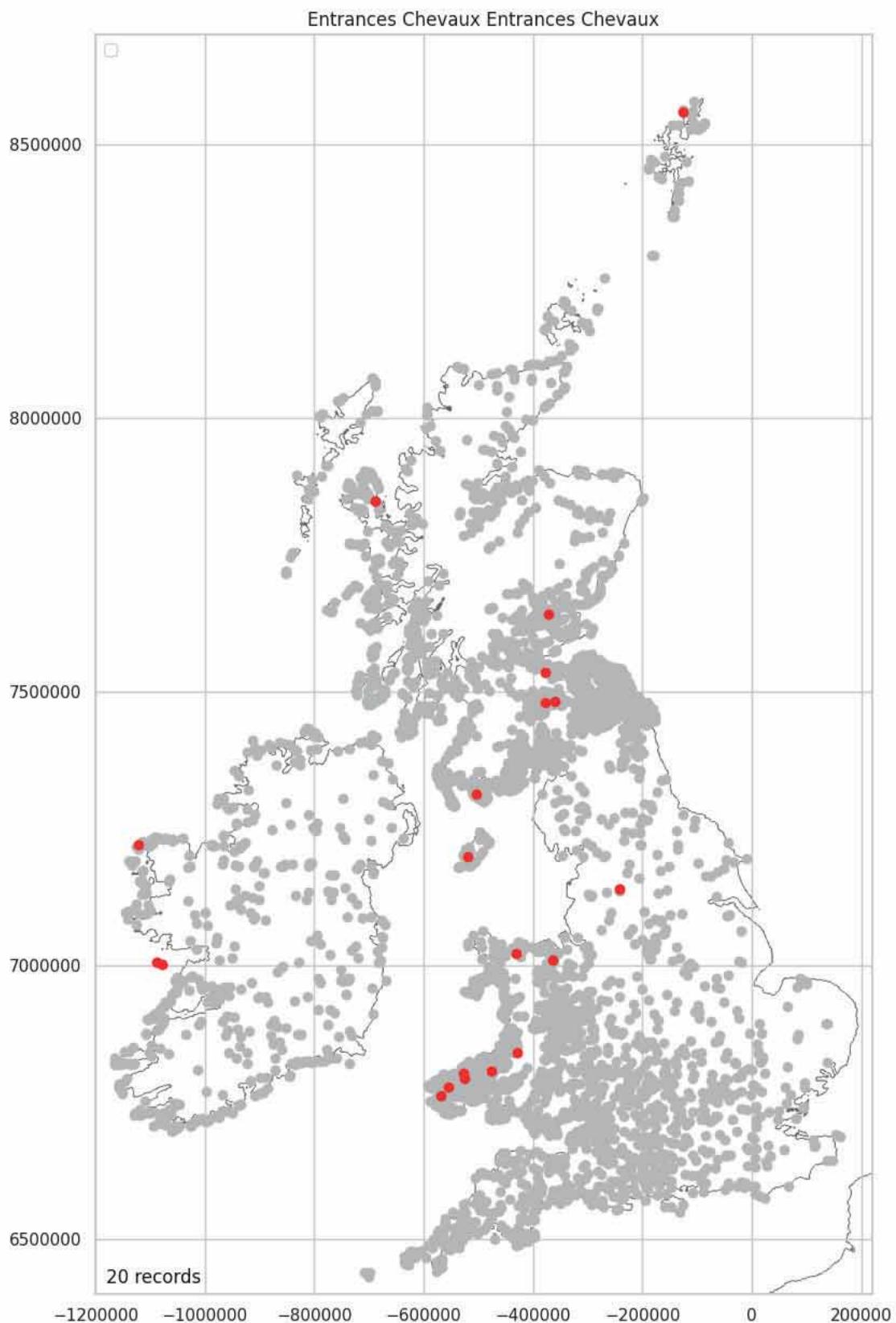
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 52%

Chevaux-de-frise Mapped

At just 20 examples it is not possible to say anything meaningful about the distribution of Chevaux de frise other than that they are rare and that most have been recorded in Wales and Scotland.

```
In [ ]: entrances_chevaux_stats = \
plot_over_grey(location_entrance_encodeable_data, 'Entrances_Chevaux', 'Yes', \
'Entrances_Chevaux')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.48%

Review Entrance Data Split

```
In [ ]: review_data_split(entrance_data, entrance_numeric_data, entrance_text_data, entrance_encodeable_data)  
Data split good.
```

Entrance Data Package

```
In [ ]: entrance_data_list = [entrance_numeric_data, entrance_text_data, entrance_encodeable_data]
```

Entrance Data Download Packages

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:
<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(entrance_data_list, 'entrance_package')
```

Enclosing Data

There are 64 Enclosing Data features which are subgrouped into:

- Area
- Multiperiod
- Circuit
- Ramparts
- Quadrants
- Current (enclosing form)
- Period (enclosing form)
- Surface (enclosing form)
- Excavation
- Gang Working
- Ditches

```
In [ ]: enclosing_features = [  
    'Enclosing_Summary',  
    'Enclosing_Area_1',  
    'Enclosing_Area_2',  
    'Enclosing_Area_3',  
    'Enclosing_Area_4',  
    'Enclosing_Enclosed_Area',  
    'Enclosing_Area',  
    'Enclosing_Multiperiod',  
    'Enclosing_Multiperiod_Comments',  
    'Enclosing_Circuit',  
    'Enclosing_Circuit_Comments',  
    'Enclosing_Max_Ramparts',  
    'Enclosing_NE_Quadrant',  
    'Enclosing_SE_Quadrant',  
    'Enclosing_SW_Quadrant',  
    'Enclosing_NW_Quadrant',  
    'Enclosing_Quadrant_Comments',  
    'Enclosing_Current_Part_Uni',  
    'Enclosing_Current_Uni',  
    'Enclosing_Current_Part_Bi',  
    'Enclosing_Current_Bi',  
    'Enclosing_Current_Part_Multi',  
    'Enclosing_Current_Multi',  
    'Enclosing_Current_Unknown',  
    'Enclosing_Period_Part_Uni',  
    'Enclosing_Period_Uni',  
    'Enclosing_Period_Part_Bi',  
    'Enclosing_Period_Bi',  
    'Enclosing_Period_Part_Multi',  
    'Enclosing_Period_Multi',  
    'Enclosing_Surface_None',  
    'Enclosing_Surface_Bank',  
    'Enclosing_Surface_Wall',  
    'Enclosing_Surface_Rubble',  
    'Enclosing_Surface_Walk',  
    'Enclosing_Surface_Timber',  
    'Enclosing_Surface_Vitrification',  
    'Enclosing_Surface_Burning',  
    'Enclosing_Surface_Paisade',  
    'Enclosing_Surface_Counter_Scarp',  
    'Enclosing_Surface_Berm',  
    'Enclosing_Surface_Unfinished',  
    'Enclosing_Surface_Other',  
    'Enclosing_Surface_Comments',  
    'Enclosing_Excavation_Nothing',  
    'Enclosing_Excavation_Bank',  
    'Enclosing_Excavation_Wall',  
    'Enclosing_Excavation_Murus',
```

```
'Encl osi ng_Exca vati on_Ti mber_Framed',
'Encl osi ng_Exca vati on_Ti mber_Laced',
'Encl osi ng_Exca vati on_Vi tri fi cation',
'Encl osi ng_Exca vati on_Burni ng',
'Encl osi ng_Exca vati on_Pali sade',
'Encl osi ng_Exca vati on_Counter_Scarp',
'Encl osi ng_Exca vati on_Berm',
'Encl osi ng_Exca vati on_Unfini shed',
'Encl osi ng_Exca vati on_No_Known',
'Encl osi ng_Exca vati on_Other',
'Encl osi ng_Exca vati on_Comments',
'Encl osi ng_Gang_Worki ng',
'Encl osi ng_Gang_Worki ng_Comments',
'Encl osi ng_Dit ches',
'Encl osi ng_Dit ches_Number',
'Encl osi ng_Dit ches_Comments']
```

```
encl osi ng_data = hill forts_data[encl osi ng_features].copy()
encl osi ng_data.head()
```

Out[]:	Enclosing_Summary	Enclosing_Area_1	Enclosing_Area_2	Enclosing_Area_3	Enclosing_Area_4	Enclosing_Enclosed_Area	Enclosing_Area_E
0	Univallate hillfort with complete circuit, but...	7.1	NaN	NaN	NaN	7.1	9.3
1	Defined differentially by single rampart to 5...	4.1	NaN	NaN	NaN	4.1	NaN
2	Three ramparts and ditches on the N. Although ...	2.8	NaN	NaN	NaN	2.8	NaN
3	Steep natural scarp artificially scarped with ...	4.8	NaN	NaN	NaN	4.8	NaN
4	In Phase I, c. 3ha were enclosed by a slight b...	3.0	14.7	NaN	NaN	14.7	NaN

Enclosing Numeric Data

There are 12 numeric Enclosing features. All contain null values.

```
In [ ]: encl osi ng_nume ri c_featu res = [
'Encl osi ng_Area_1',
'Encl osi ng_Area_2',
'Encl osi ng_Area_3',
'Encl osi ng_Area_4',
'Encl osi ng_Encl osed_Area',
'Encl osi ng_Area',
'Encl osi ng_Max_Ramparts',
'Encl osi ng_NE_Quadrant',
'Encl osi ng_SE_Quadrant',
'Encl osi ng_SW_Quadrant',
'Encl osi ng_NW_Quadrant',
'Encl osi ng_Dit ches_Number']
```

```
encl osi ng_nume ri c_data = encl osi ng_data[encl osi ng_nume ri c_featu res].copy()
encl osi ng_nume ri c_data.head()
```

Out[]:	Enclosing_Area_1	Enclosing_Area_2	Enclosing_Area_3	Enclosing_Area_4	Enclosing_Enclosed_Area	Enclosing_Area	Enclosing_Max_Rampar
0	7.1	NaN	NaN	NaN	7.1	9.3	1
1	4.1	NaN	NaN	NaN	4.1	NaN	1
2	2.8	NaN	NaN	NaN	2.8	NaN	2
3	4.8	NaN	NaN	NaN	4.8	NaN	1
4	3.0	14.7	NaN	NaN	14.7	NaN	2

```
In [ ]: encl osi ng_nume ri c_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Enclosing_Area_1    3807 non-null   float64
 1   Enclosing_Area_2    335  non-null   float64
 2   Enclosing_Area_3    68   non-null   float64
 3   Enclosing_Area_4    11   non-null   float64
 4   Enclosing_Enclosed_Area 3807 non-null   float64
 5   Enclosing_Area      1263 non-null   float64
 6   Enclosing_Max_Ramparts 3999 non-null   float64
 7   Enclosing_NE_Quadrant 3927 non-null   float64
 8   Enclosing_SE_Quadrant 3899 non-null   float64
 9   Enclosing_SW_Quadrant 3896 non-null   float64
 10  Enclosing_NW_Quadrant 3899 non-null   float64
 11  Enclosing_Ditches_Number 3279 non-null   float64
dtypes: float64(12)
memory usage: 388.9 KB

```

Enclosing Numeric Data - Resolve Null Values

Test for -1.

```
In [ ]: test_num_list_for_minus_one(enclosing_numeric_data, enclosing_numeric_features)
```

```

Enclosing_Area_1 0
Enclosing_Area_2 0
Enclosing_Area_3 0
Enclosing_Area_4 0
Enclosing_Enclosed_Area 0
Enclosing_Area 0
Enclosing_Max_Ramparts 0
Enclosing_NE_Quadrant 0
Enclosing_SE_Quadrant 0
Enclosing_SW_Quadrant 0
Enclosing_NW_Quadrant 0
Enclosing_Ditches_Number 0

```

Replace null with -1.

```
In [ ]: enclosing_numeric_data = \
update_num_list_for_minus_one(enclosing_numeric_data, enclosing_numeric_features)
enclosing_numeric_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Enclosing_Area_1    4147 non-null   float64
 1   Enclosing_Area_2    4147 non-null   float64
 2   Enclosing_Area_3    4147 non-null   float64
 3   Enclosing_Area_4    4147 non-null   float64
 4   Enclosing_Enclosed_Area 4147 non-null   float64
 5   Enclosing_Area      4147 non-null   float64
 6   Enclosing_Max_Ramparts 4147 non-null   float64
 7   Enclosing_NE_Quadrant 4147 non-null   float64
 8   Enclosing_SE_Quadrant 4147 non-null   float64
 9   Enclosing_SW_Quadrant 4147 non-null   float64
 10  Enclosing_NW_Quadrant 4147 non-null   float64
 11  Enclosing_Ditches_Number 4147 non-null   float64
dtypes: float64(12)
memory usage: 388.9 KB

```

Enclosing Area 1 Plotted

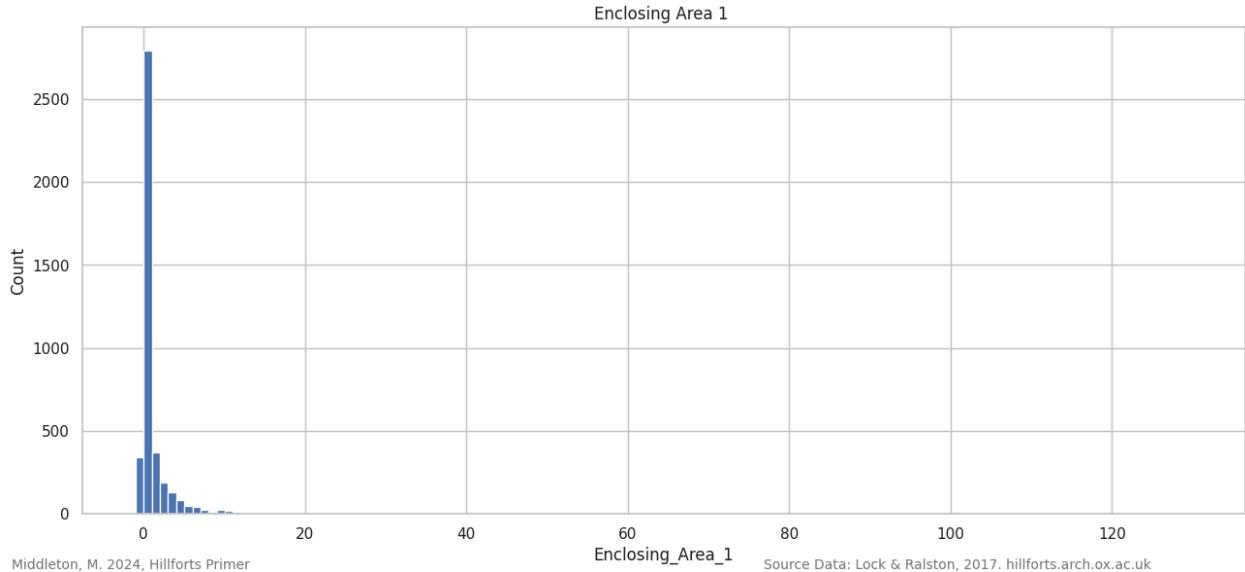
With 3807 entries, Area 1 is the most populated of the Area features and refers to the, "Enclosed area ... within the inner rampart/bank/wall". ([Data Structure](#))

Most forts are less than one hectare in size with outliers up to 130 hectares. The data has a very long tail.

```
In [ ]: enclosing_numeric_data['Enclosing_Area_1'].describe()
```

```
Out[ ]: count    4147.000000
         mean     1.427997
         std      5.192075
         min     -1.000000
         25%     0.130000
         50%     0.340000
         75%     1.000000
         max     130.000000
Name: Enclosing_Area_1, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(enclosing_numeric_data, 1, \
                               'Enclosing_Area_1', 'Count', 'Enclosing_Area_1', \
                               int(enclosing_numeric_data['Enclosing_Area_1'].max()))
```



Enclosing Area 1 Clipped Plotted

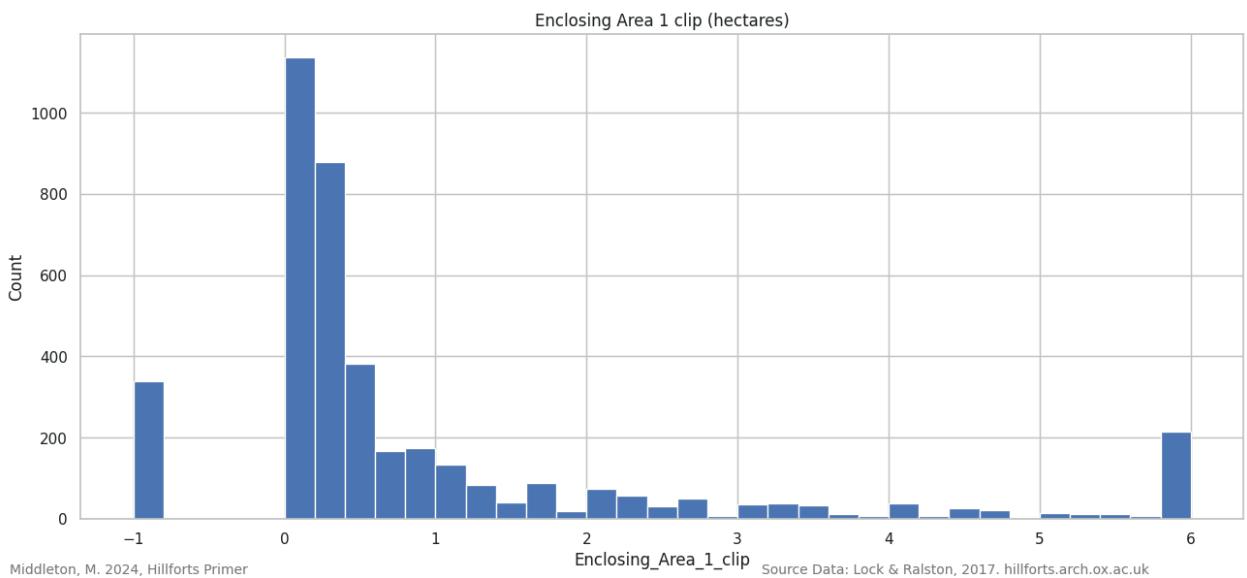
The outliers make it difficult to see the detail in the data at the lower end. To improve the clarity of the plot, the data is capped at 6 Ha. Note that the histogram includes the null values (-1). All outliers above 6 Ha are collected into the capped value.

Most forts are below 0.5 Ha in size. The majority (95.6%) of forts are below 10.5 Ha in size.

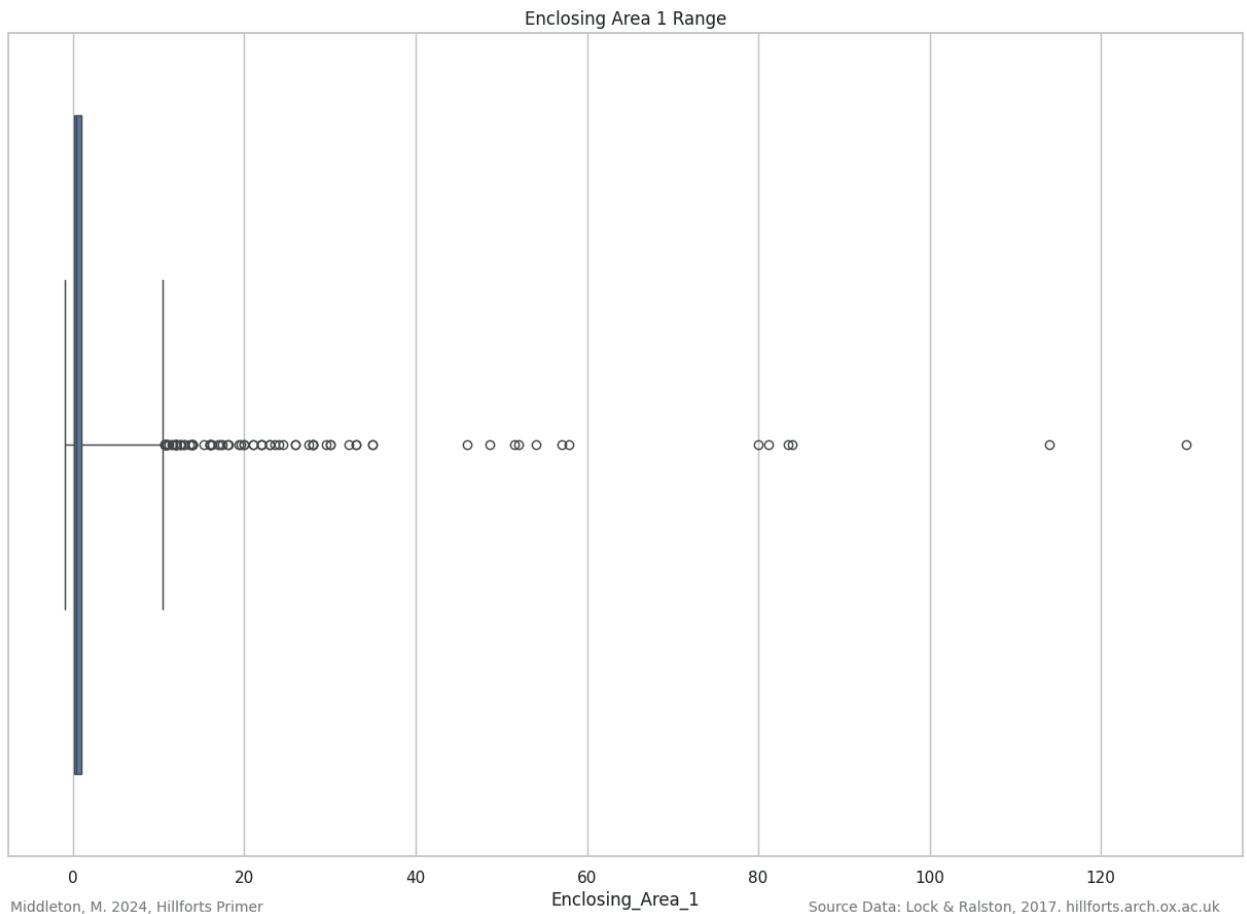
```
In [ ]: enclosing_area_1_data_clip = enclosing_numeric_data.copy()
enclosing_area_1_data_clip['Enclosing_Area_1_clip'] = \
enclosing_area_1_data_clip['Enclosing_Area_1'].clip(\
clip(enclosing_area_1_data_clip['Enclosing_Area_1'], 6, axis=0)
enclosing_area_1_data_clip['Enclosing_Area_1_clip'].describe()
```

```
Out[ ]: count    4147.000000
         mean     0.944245
         std      1.655712
         min     -1.000000
         25%     0.130000
         50%     0.340000
         75%     1.000000
         max     6.000000
Name: Enclosing_Area_1_clip, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(enclosing_area_1_data_clip, 1, 'Enclosing_Area_1_clip', \
                               'Count', 'Enclosing_Area_1_clip', 35, '(hectares)')
```



```
In [ ]: encl osing_area_1_data = \
plot_data_range(encl osing_nume ric_data['Encl osing_Area_1'], \
'Encl osing_Area_1', "h")
```



```
In [ ]: encl osing_area_1_data
```

```
Out[ ]: [-1.0, 0.13, 0.34, 1.0, 10.5]
```

The test below was carried out to review if using -1 for null values might influence the output in terms of the quartile ranges. The question was, does it alter the positive quartile ranges between the minimum value at the start of quarter 1, (-1) to the current maximum value at the top end of quarter 4, (10.5 Ha). The impact of using -1 was tested by changing -1 to -0.01. This was found to make no difference to the positive quartile values. As it had no impact, -1 was retained.

To activate this code, and to confirm the observations above, remove the '#' symbols and re-run the notebook using the menu **Runtime>Run all**.

```
In [ ]: # """Select area features"""
# area_features = [
#   'Encl osing_Area_1',
```

```
# 'Enclosing_Area_2',
# 'Enclosing_Area_3',
# 'Enclosing_Area_4',
# 'Enclosing_Enclosed_Area',
# 'Enclosing_Area']
```

```
In [ ]: # """Change -1 to -0.01"""
# for feature in area_features:
#     enclosing_numeric_data[feature] = enclosing_numeric_data[feature].replace(-1, -0.01)
# enclosing_numeric_data.head()
```

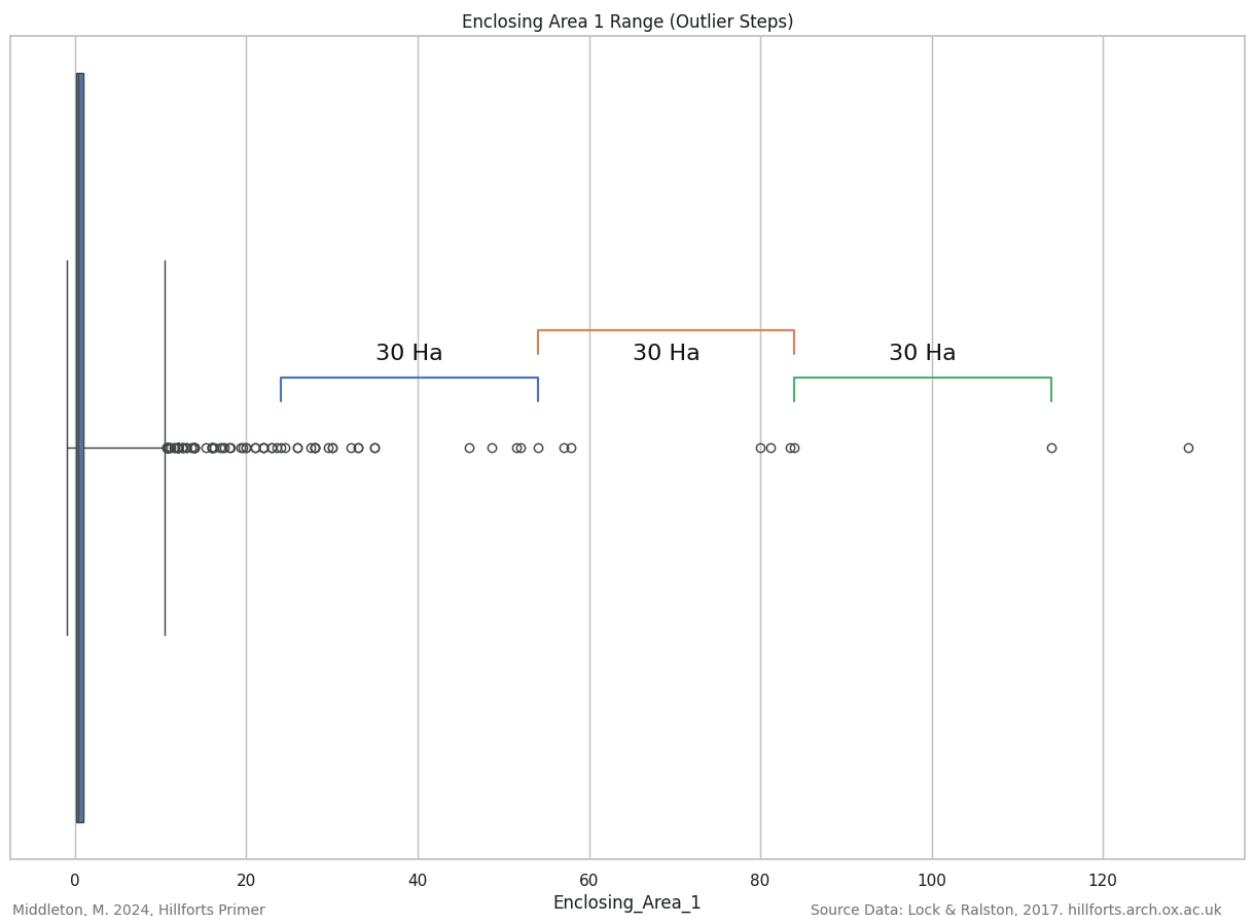
```
In [ ]: # """Plot new boxplot"""
# Enclosing_Area_1_data_updated = plot_data_range(enclosing_numeric_data['Enclosing_Area_1'], 'Enclosing_Area_1_Updated')
```

```
In [ ]: # """Review new boxplot values"""
# Enclosing_Area_1_data_updated
```

Enclosing Area 1 - Outlier Distribution

The outliers are grouped into four small clusters. The first continues out from the main range; There is then a gap to the next cluster at around 50 Ha; another gap to a small cluster at 80 Ha and then, a final gap, to a pair of sites which are over 110 Ha. One observation is that there is a similarity in the step sizes between these clusters of around 30 Ha. It is important to note that the numbers of sites in these clusters are very small. See: [Enclosing Area 1: Regional Boxplots](#)

```
In [ ]: enclosing_area_1_data = \
plot_data_range_plot(enclosing_numeric_data['Enclosing_Area_1'], \
'Enclosing_Area_1', "h")
```

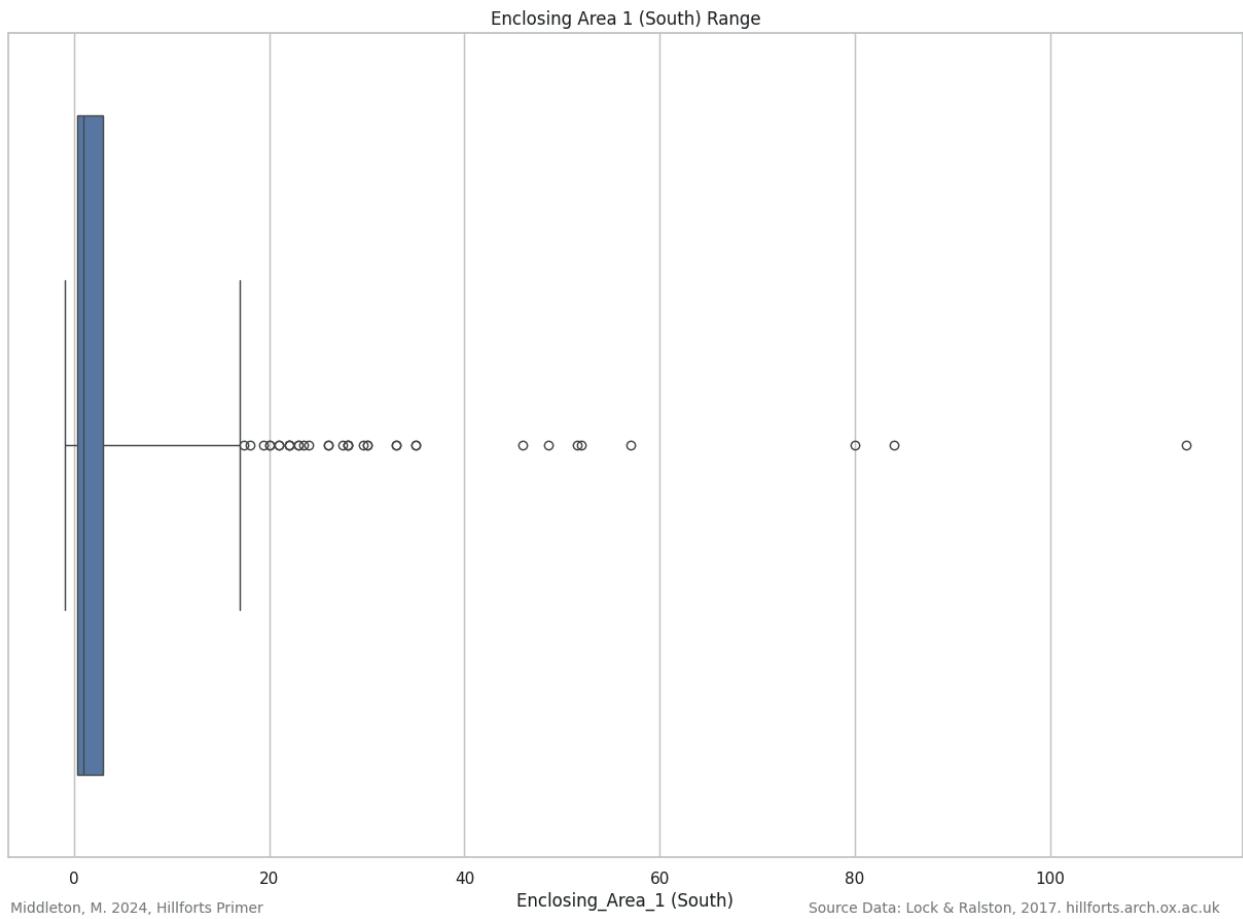


Enclosing Area 1: South Plotted

In the southern data package, 50% of the hillforts sit in a range between 0.3 and 3 hectares and 95.6% of the forts are less than 17 hectares. Most outliers are clustered near the main range, up to the high 30s. There is a small cluster between 40 and 60 hectares, two forts in the 80s and a single fort of 130 Ha. The median is 0.9 hectares and the bar chart shows the majority of forts are at the lower end of the range.

```
In [ ]: south['uid'] = south.index
location_enclosing_data_south = \
pd.merge(south, enclosing_numeric_data, left_on='uid', right_index=True)
```

```
In [ ]: encl osi ng_area_south_data = \
plot_data_range(location_encl osi ng_data_south['Encl osi ng_Area_1'], \
'Encl osi ng_Area_1 (South)', "h")
```



Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

```
In [ ]: encl osi ng_area_south_data
```

```
Out[ ]: [-1.0, 0.3, 0.9, 3.0, 17.0]
```

```
In [ ]: location_encl osi ng_data_south['Encl osi ng_Area_1'].describe()
```

```
Out[ ]: count    1555.000000
mean      2.637119
std       6.431279
min     -1.000000
25%      0.300000
50%      0.900000
75%      3.000000
max     114.000000
Name: Encl osi ng_Area_1, dtype: float64
```

Note how the mean and the median are quite different. The median, 0.9 Ha (the central value in a sorted list of values). Here the mean (2.63 Ha) is larger because of the huge variation in enclosing area. The small number of very large hillforts have an unduly large influence over the mean because the majority of hillforts are very small. A more realistic mean can be achieved by trimming the data to exclude a percentage of the data from the extremes.

```
In [ ]: trim_pcnt = 0.1 # 10%
location_encl osi ng_data_trim_mean = \
stats.trim_mean(location_encl osi ng_data_south['Encl osi ng_Area_1'], trim_pcnt)
location_encl osi ng_data_trim_mean
```

```
Out[ ]: 1.5289799196787148
```

```
In [ ]: test_cat_list_for_NA
```

```
Out[ ]: test_cat_list_for_NA
def test_cat_list_for_NA(dataframe, cat_list)
```

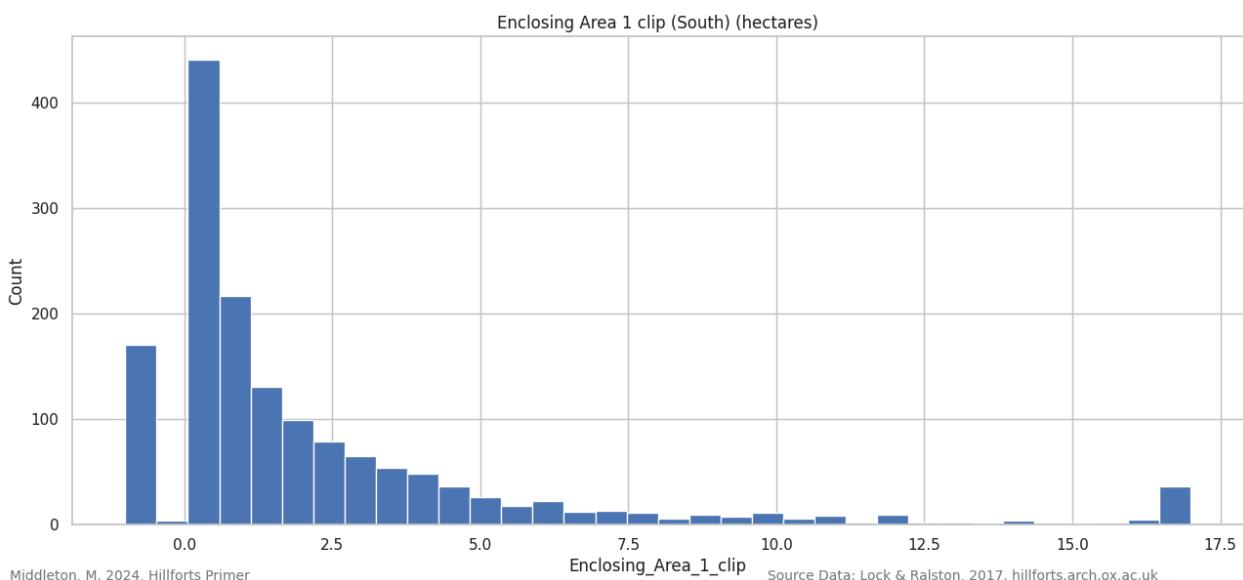
```
<no docstring>
```

To facilitate the reading of the plot, the data is clipped at the 75th percentile (17 Ha). Any data above 17 Ha is grouped at this value.

```
In [ ]: south_encl osi ng_area_1_data_clip = location_encl osi ng_data_south.copy()
south_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'] = \
south_encl osi ng_area_1_data_clip['Encl osi ng_Area_1'].\
clip(south_encl osi ng_area_1_data_clip['Encl osi ng_Area_1'], \
encl osi ng_area_south_data[4], axis=0)
south_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'].describe()
```

```
Out[ ]: count    1555.000000
mean     2.236154
std      3.570042
min     -1.000000
25%      0.300000
50%      0.900000
75%      3.000000
max     17.000000
Name: Encl osi ng_Area_1_clip, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(south_encl osi ng_area_1_data_clip, 1, \
'Encl osi ng_Area_1_clip', 'Count', \
'Encl osi ng_Area_1_clip (South)', \
int(encl osi ng_area_south_data[4]*2), '(hectares)')
```



Enclosing Area 1: Northeast Plotted

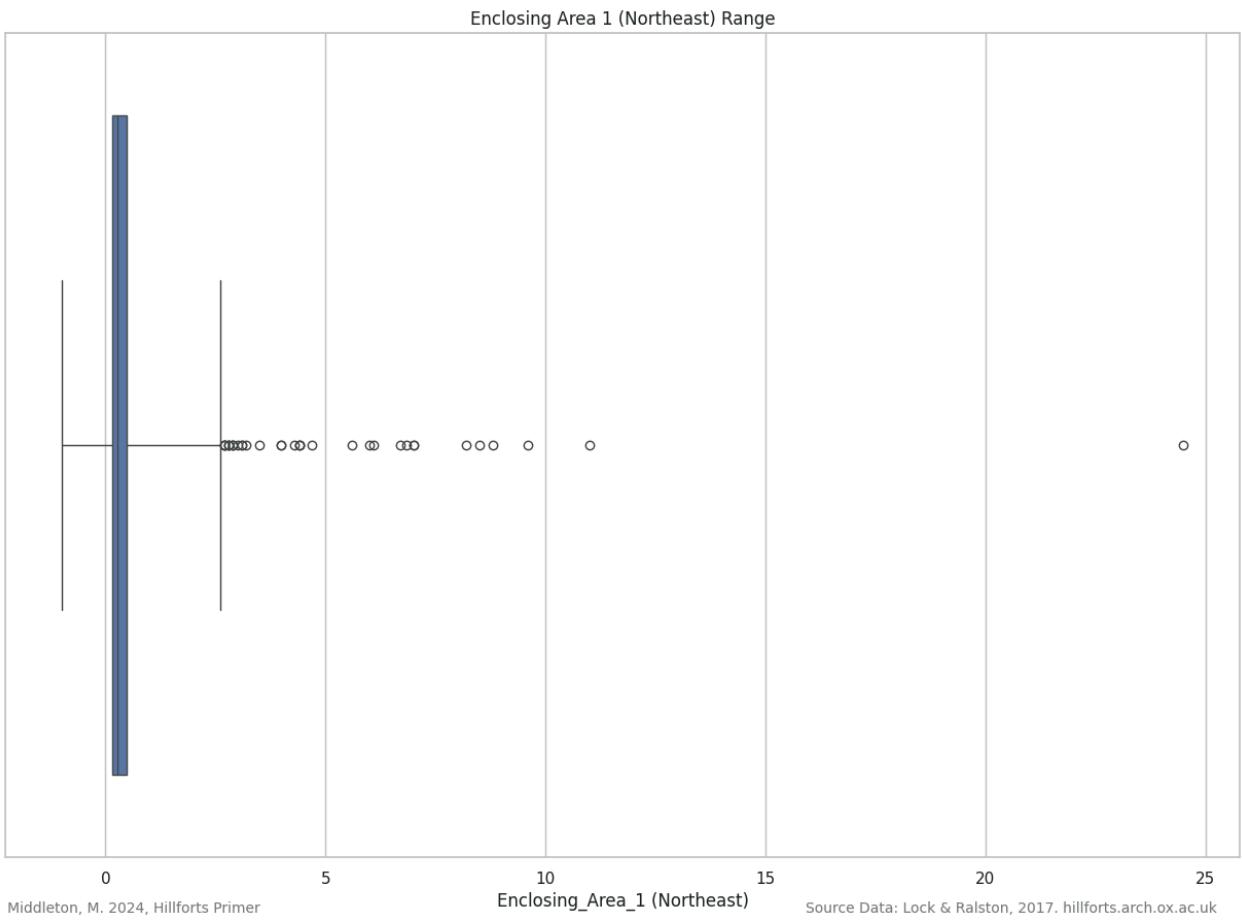
In the Northeast, 50% of sites sit within a narrow band between 0.15 and 0.48 Ha and 95.6% of sites are less than 2.6 Ha. Most outliers stand out from the top end of the main range up to 10 Ha. There is a single outlier at 24 Ha (1504: Roulston Scar, North Yorkshire) which is located right at the southern edge of the NE data package and may indicate that this fort has characteristics more in line with the southern data. See: [Enclosing Area 1: Regional Boxplots](#).

```
In [ ]: north_east['uid'] = north_east.index
location_encl osi ng_data_ne = \
pd.merge(north_east.reset_index(), encl osi ng_numeric_data, left_on='uid', \
right_index=True)
location_encl osi ng_data_ne = pd.merge(name_and_number, \
location_encl osi ng_data_ne, \
left_index=True, right_on='uid')
```

```
In [ ]: location_encl osi ng_data_ne[location_encl osi ng_data_ne['Encl osi ng_Area_1'] > 20]
```

	Main_Atlas_Number	Main_Display_Name	index	Location_X	Location_Y	Cluster	uid	Enclosing_Area_1	Enclosing_Area_2	Enclosi
404	1504	Roulston Scar, North Yorkshire (Sutton Bank; C...	1437	-134884	7213231	Northeast	1437	24.5		-1.0

```
In [ ]: encl osi ng_area_ne_data = plot_data_range(location_encl osi ng_data_ne['Encl osi ng_Area_1'], 'Encl osi ng_Area_1 (Northeas
```



```
In [ ]: encl osi ng_area_ne_data
```

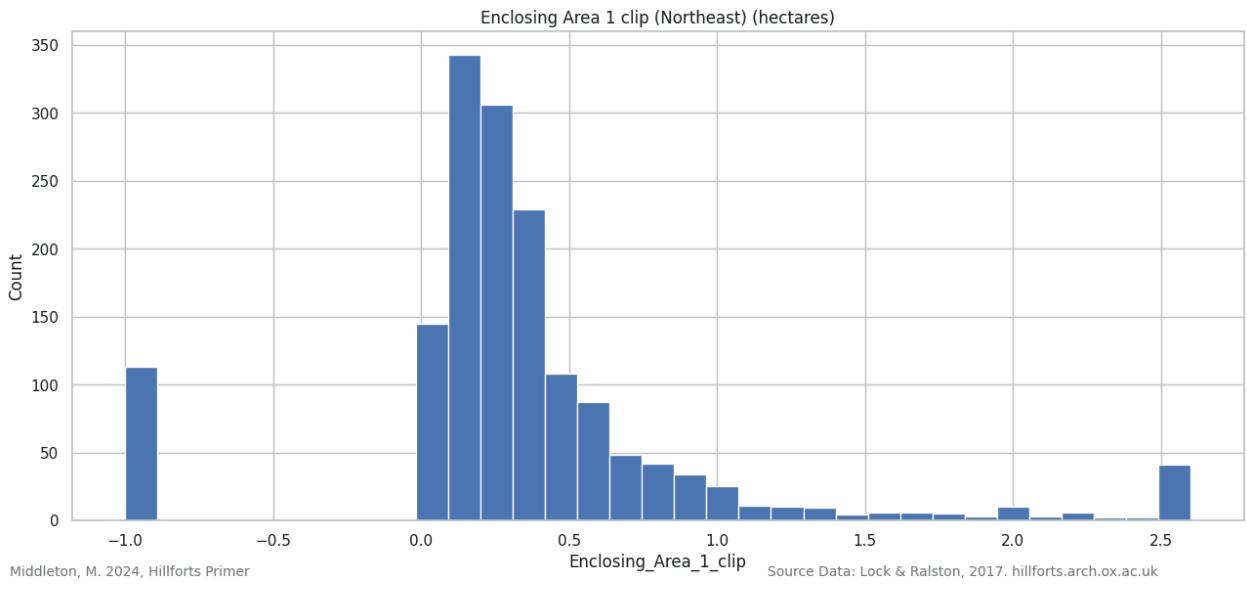
```
Out[ ]: [-1.0,  0.15,  0.27,  0.48,  2.6]
```

```
In [ ]: ne_encl osi ng_area_1_data_clip = location_encl osi ng_data_ne.copy()
ne_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'] = \
ne_encl osi ng_area_1_data_clip['Encl osi ng_Area_1']. \
clip(ne_encl osi ng_area_1_data_clip['Encl osi ng_Area_1'], \
      encl osi ng_area_ne_data[4], axis=0)
ne_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'].describe()
```

```
Out[ ]: count    1598.000000
mean      0.354887
std       0.624009
min     -1.000000
25%      0.150000
50%      0.270000
75%      0.480000
max      2.600000
Name: Encl osi ng_Area_1_clip, dtype: float64
```

The data is clipped at the 75th percentile (2.6 Ha). Any data above 2.6 Ha is grouped at this value.

```
In [ ]: plot_bar_chart_numeric(ne_encl osi ng_area_1_data_clip, 1, \
                           'Encl osi ng_Area_1_clip', 'Count', \
                           'Encl osi ng_Area_1_clip (Northeast)', \
                           int(encl osi ng_area_ne_data[4]*13), '(hectares)')
```



Enclosing Area 1: Northwest Plotted

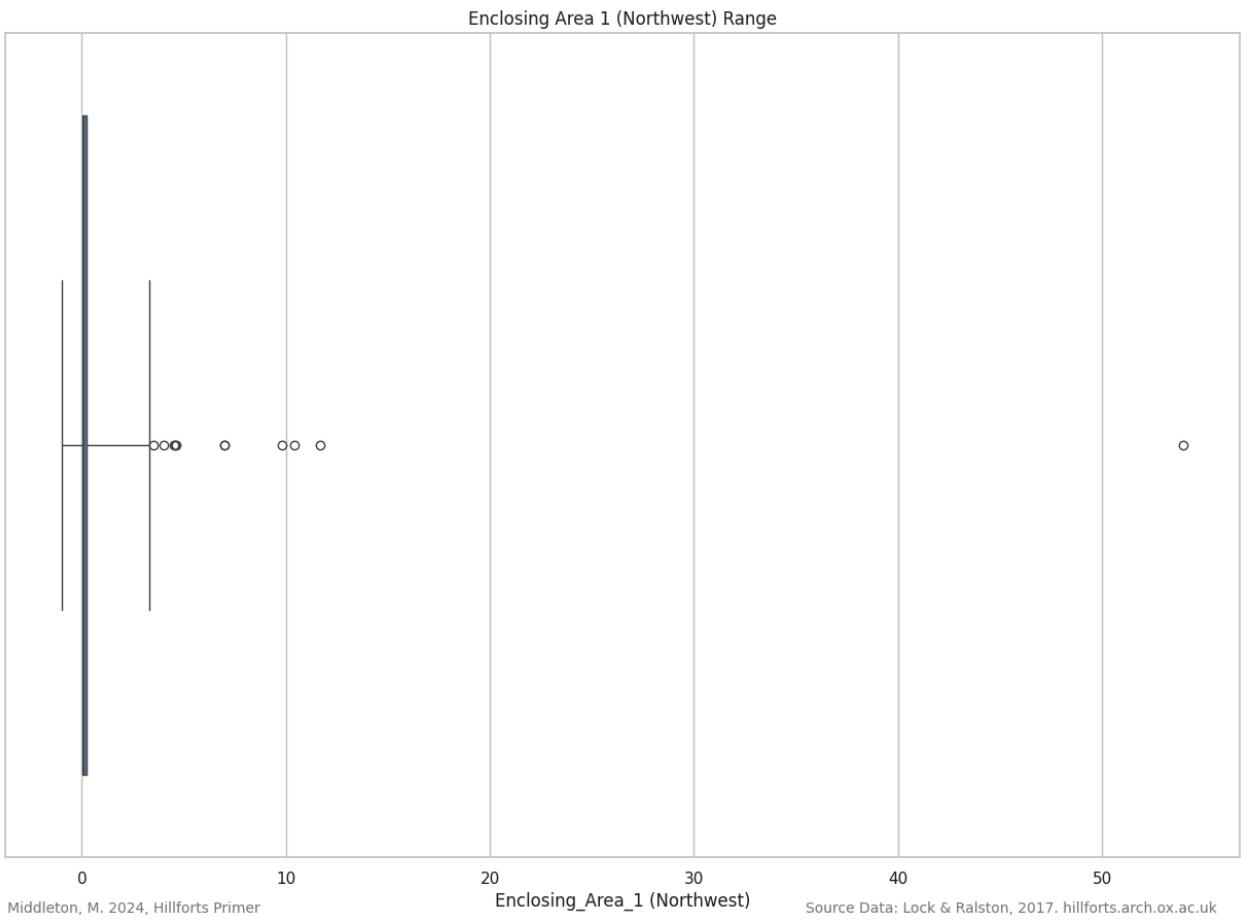
This area is notable for the small size of most forts. The central 50% of sites are contained in a very narrow band between 0.05 and 0.22 Ha and 95.6% being less than 3.3 Ha. There are a small number of outliers up to 11.7 Ha and one single, exceptionally large, fort at 54 Ha (201: Mull of Galloway). See: [Enclosing Area 1: Regional Boxplots](#).

```
In [ ]: north_west['uid'] = north_west.index
location_enclaving_data_nw = pd.merge(north_west.reset_index(), enclaving_numeric_data, left_on='uid', right_index=True)
location_enclaving_data_nw = pd.merge(name_and_number, location_enclaving_data_nw, left_index=True, right_on='uid')

In [ ]: location_enclaving_data_nw[location_enclaving_data_nw['Enclaving_Area_1'] > 50]

Out[ ]:   Main_Atlas_Number  Main_Display_Name  index  Location_X  Location_Y  Cluster  uid  Enclosing_Area_1  Enclosing_Area_2  Enclosing_
          36                  Mull of Galloway,  194     -542988    7291829  Northwest  194      54.0           -1.0

In [ ]: enclaving_area_nw_data = \
plot_data_range(location_enclaving_data_nw['Enclaving_Area_1'], \
'Enclaving_Area_1 (Northwest)', "h")
```



```
In [ ]: encl osi ng_area_nw_data
```

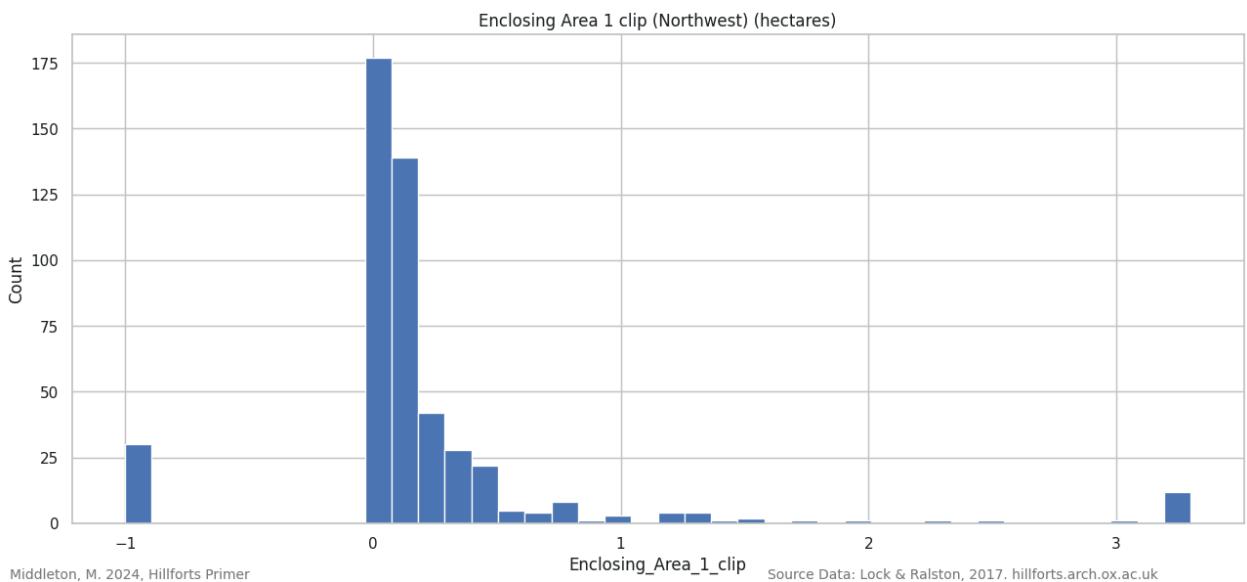
```
Out[ ]: [-1.0,  0.05,  0.09,  0.22,  3.3]
```

```
In [ ]: nw_encl osi ng_area_1_data_clip = location_encl osi ng_data_nw.copy()
nw_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'] = \
nw_encl osi ng_area_1_data_clip['Encl osi ng_Area_1']. \
clip(nw_encl osi ng_area_1_data_clip['Encl osi ng_Area_1'], \
      encl osi ng_area_nw_data[4], axis=0)
nw_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'].describe()
```

```
Out[ ]: count    487.000000
mean     0.213747
std      0.656978
min     -1.000000
25%      0.050000
50%      0.090000
75%      0.220000
max      3.300000
Name: Encl osi ng_Area_1_clip, dtype: float64
```

The data is clipped at the 75th percentile (3.3 Ha). Any data above 3.3 Ha is grouped at this value.

```
In [ ]: plot_bar_chart_numeric(nw_encl osi ng_area_1_data_clip, 1, \
                           'Encl osi ng_Area_1_clip', 'Count', \
                           'Encl osi ng_Area_1_clip (Northwest)', \
                           int(encl osi ng_area_south_data[4]*2.4), '(hectares)')
```



Enclosing Area 1: North Ireland Plotted

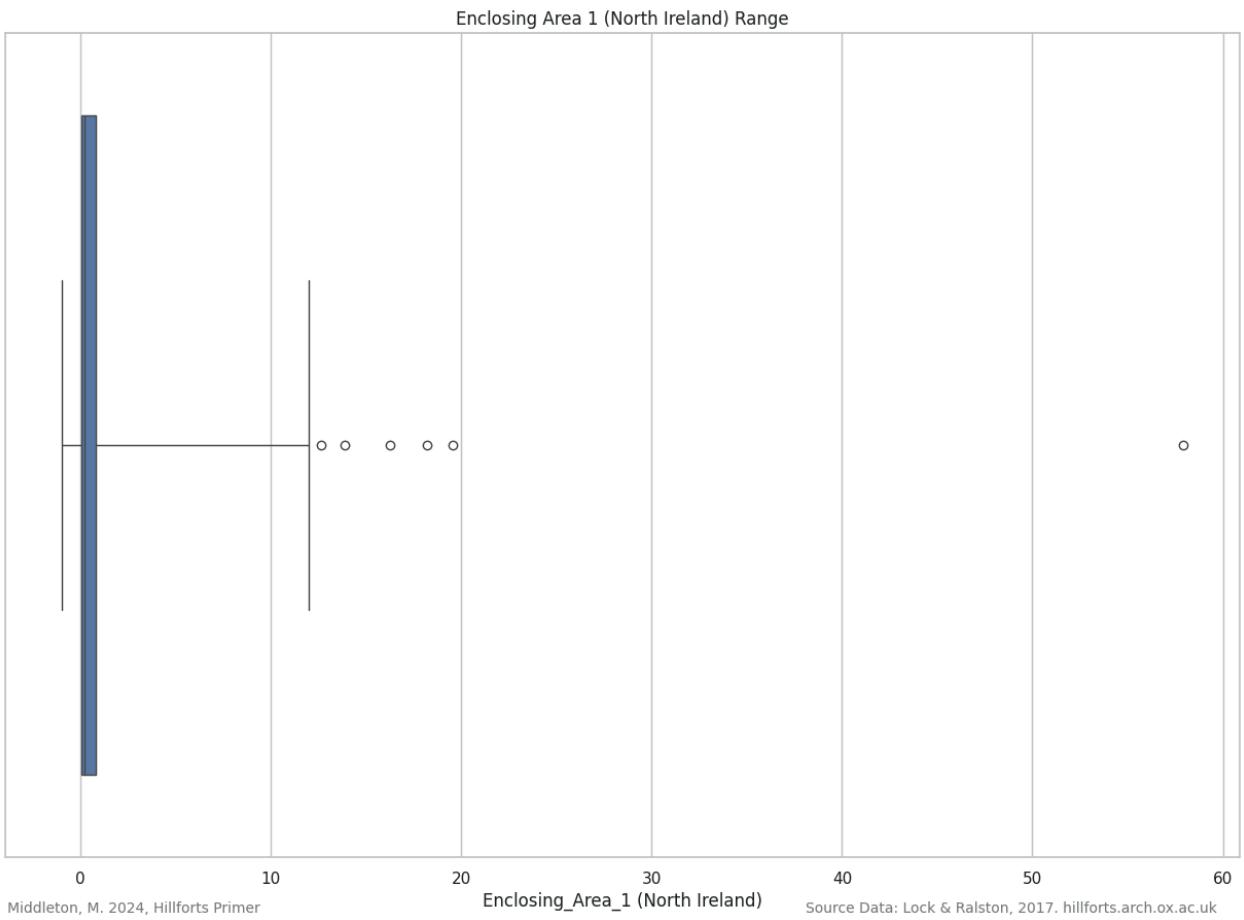
The central 50% of sites are between 0.07 and 0.79 Ha and 95.6% are less than 11.97 Ha. It is notable that the upper whisker is very long due to the concentration of forts at the lower end of the range and there being a large variance in size among the larger forts up to 11.7 Ha. See the bar chart below and [Enclosing Area 1 Distribution of Data by Region](#). There is one single, atypically large, fort at 57.94 Ha (1104: Inishark (Inis Airc)). See: [Enclosing Area 1: Regional Boxplots](#).

```
In [ ]: north_i_rel_and['uid'] = north_i_rel_and.index
location_enclising_data_i_rel_and_n = \
pd.merge(north_i_rel_and.reset_index(), enclising_numeric_data, \
left_on='uid', right_index=True)
location_enclising_data_i_rel_and_n = pd.merge(name_and_number, \
location_enclising_data_i_rel_and_n, \
left_index=True, right_on='uid')
```

```
In [ ]: location_enclising_data_i_rel_and_n \
[location_enclising_data_i_rel_and_n['Enclising_Area_1'] > 50]
```

```
Out[ ]:   Main_Atlas_Number  Main_Display_Name  index  Location_X  Location_Y  Cluster  uid  Enclosing_Area_1  Enclosing_Area_2  Enclosing_
96          1104    Inishark (Inis Airc),  1076     -1145432    7096391  North  1076      57.94           -1.0
```

```
In [ ]: enclising_area_i_rel_and_n_data = \
plot_data_range(location_enclising_data_i_rel_and_n['Enclising_Area_1'], \
'Enclising_Area_1 (North Ireland)', "h")
```



```
In [ ]: enclosi ng_area_i rel and_n_data
```

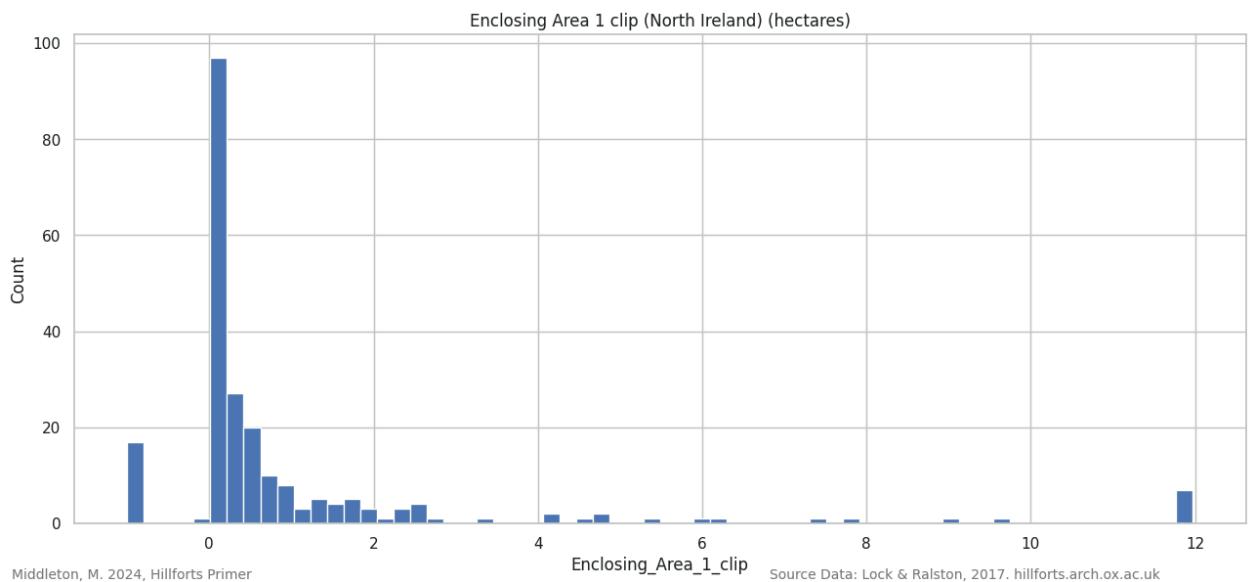
```
Out[ ]: [-1.0,  0.07,  0.21,  0.79,  11.97]
```

```
In [ ]: n_ie_encl osi ng_area_1_data_c l i p = location_encl osi ng_data_i rel and_n. copy()
n_ie_encl osi ng_area_1_data_c l i p['Encl osi ng_Area_1_c l i p'] = \
n_ie_encl osi ng_area_1_data_c l i p['Encl osi ng_Area_1']. \
clip(n_ie_encl osi ng_area_1_data_c l i p['Encl osi ng_Area_1'], \
enclosi ng_area_i rel and_n_data[4], axis=0)
n_ie_encl osi ng_area_1_data_c l i p['Encl osi ng_Area_1_c l i p']. descri be()
```

```
Out[ ]: count    229.000000
mean      1.046594
std       2.479355
mi n     -1.000000
25%      0.070000
50%      0.210000
75%      0.790000
max      11.970000
Name: Encl osi ng_Area_1_c l i p, dtype: float64
```

The data is clipped at the 75th percentile (11.97 Ha). Any data above 11.97 Ha is grouped at this value.

```
In [ ]: pl ot_bar_chart_nume ric(n_ie_encl osi ng_area_1_data_c l i p, 1, \
'Encl osi ng_Area_1_c l i p', 'Count', \
'Encl osi ng_Area_1_c l i p (North Ireland)', \
int(enclosi ng_area_south_data[4]*3.8), '(hectares)')
```



Enclosing Area 1: South Ireland Plotted

The boxplot for South Ireland is compressed due to the huge scale of the outliers in this region. For more clarity see [Enclosing Area 1: Regional Boxplots](#). The central 50% of sites are between 0.11 and 1.3 Ha and 95.6% are less than 12.01 Ha. Like North Ireland, the upper whisker is long due to the concentration of forts below 0.2 Ha and the variance in size among the larger forts up to 12.01 Ha. See the bar chart below. There are three forts over 80 Ha, of which one, at 130 Ha, is enormous (727: Spinans Hill 2). This is the largest fort, by area, recorded in the atlas.

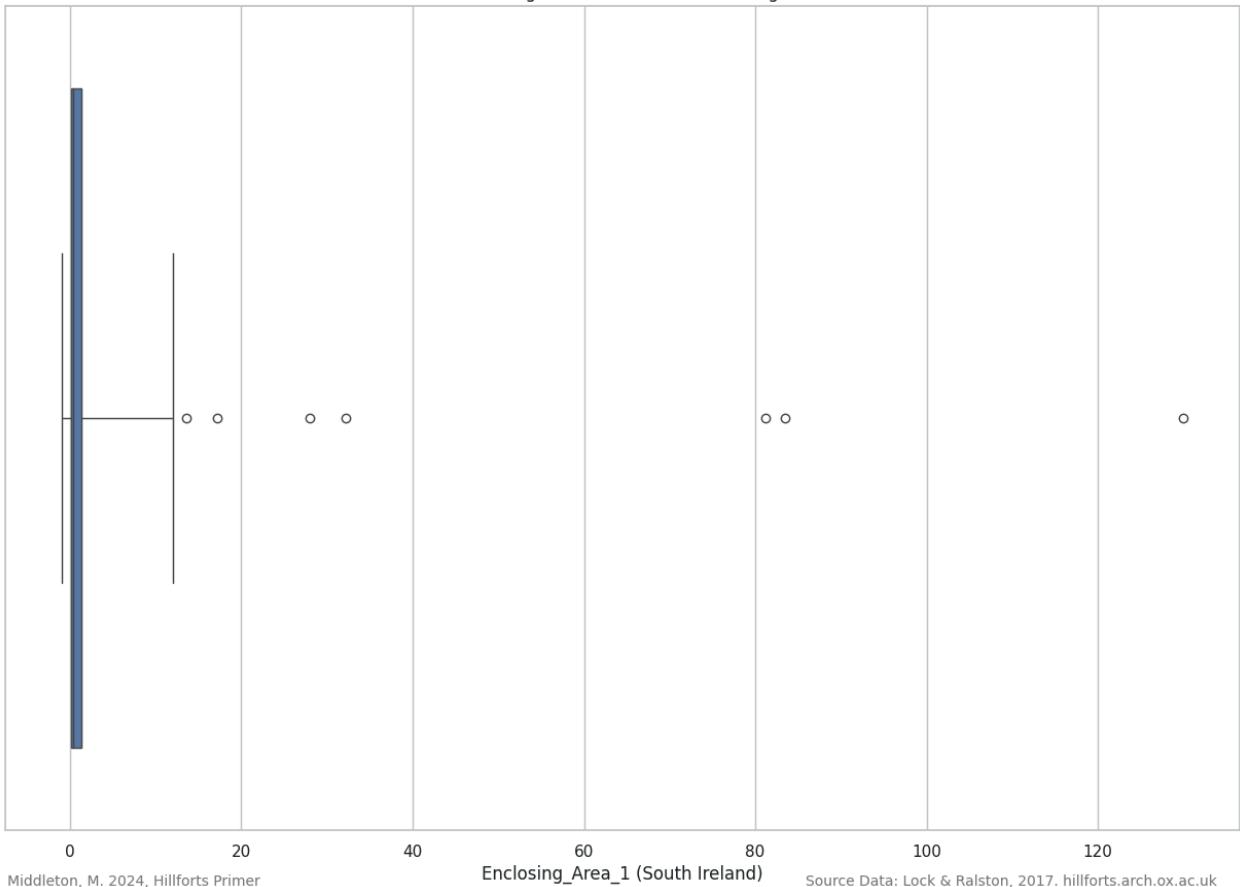
```
In [ ]: south_i_rel and['uid'] = south_i_rel and.i_index
location_encl osing_data_i_rel_and_s = \
pd.merge(south_i_rel_and.reset_index(), encl osing_numeric_c_data, left_on='uid', \
         right_index=True)
location_encl osing_data_i_rel_and_s = \
pd.merge(name_and_number, location_encl osing_data_i_rel_and_s, left_index=True, \
         right_on='uid')
```

```
In [ ]: location_encl osing_data_i_rel_and_s[
    location_encl osing_data_i_rel_and_s['Encl osing_Area_1'] > 80]. \
    sort_values(by='Encl osing_Area_1', ascending=False)
```

	Main_Atlas_Number	Main_Display_Name	index	Location_X	Location_Y	Cluster	uid	Enclosing_Area_1	Enclosing_Area_2	Enclosing
48	727	Spinans Hill 2, Wicklow (Brusselstown, Spinans...)	705	-737471	6976335	South Ireland	705	130.0		-1.0
264	1970	Ballynacarriga, Cork	1864	-1130188	6726204	South Ireland	1864	83.5		-1.0
121	901	Downmacpatrick (Old Head), Cork	878	-950310	6730002	South Ireland	878	81.2		-1.0

```
In [ ]: encl osing_area_i_rel_and_s_data = \
plot_data_range(location_encl osing_data_i_rel_and_s\ 
['Encl osing_Area_1'], 'Encl osing_Area_1 (South Ireland)', "h")
```

Enclosing Area 1 (South Ireland) Range



```
In [ ]: encl osi ng_area_i rel and_s_data
```

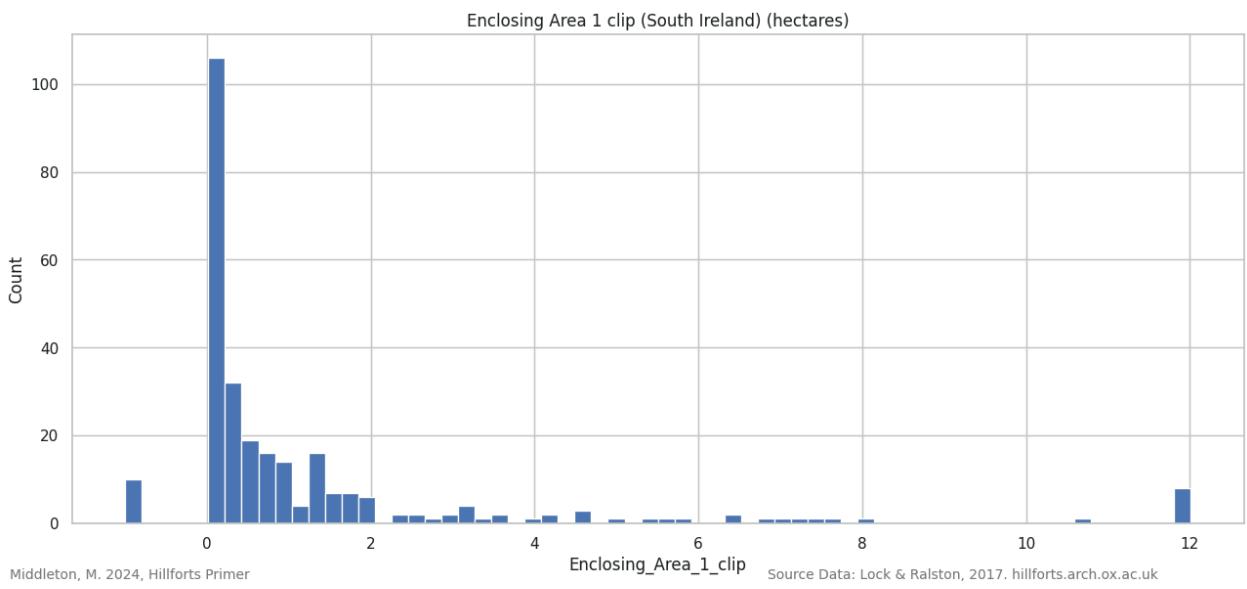
```
Out[ ]: [-1.0,  0.11,  0.32,  1.2975,  12.01]
```

```
In [ ]: s_ie_encl osi ng_area_1_data_clip = location_encl osi ng_data_i rel and_s.copy()
s_ie_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'] = \
s_ie_encl osi ng_area_1_data_clip['Encl osi ng_Area_1']. \
clip(s_ie_encl osi ng_area_1_data_clip['Encl osi ng_Area_1'], \
      enclosi ng_area_i rel and_s_data[4], axis=0)
s_ie_encl osi ng_area_1_data_clip['Encl osi ng_Area_1_clip'].describe()
```

```
Out[ ]: count    278.000000
mean     1.289640
std      2.483341
min     -1.000000
25%     0.110000
50%     0.320000
75%     1.297500
max     12.010000
Name: Encl osi ng_Area_1_clip, dtype: float64
```

The data is clipped at the 75th percentile (12.01 Ha). Any data above 12.01 Ha is grouped at this value.

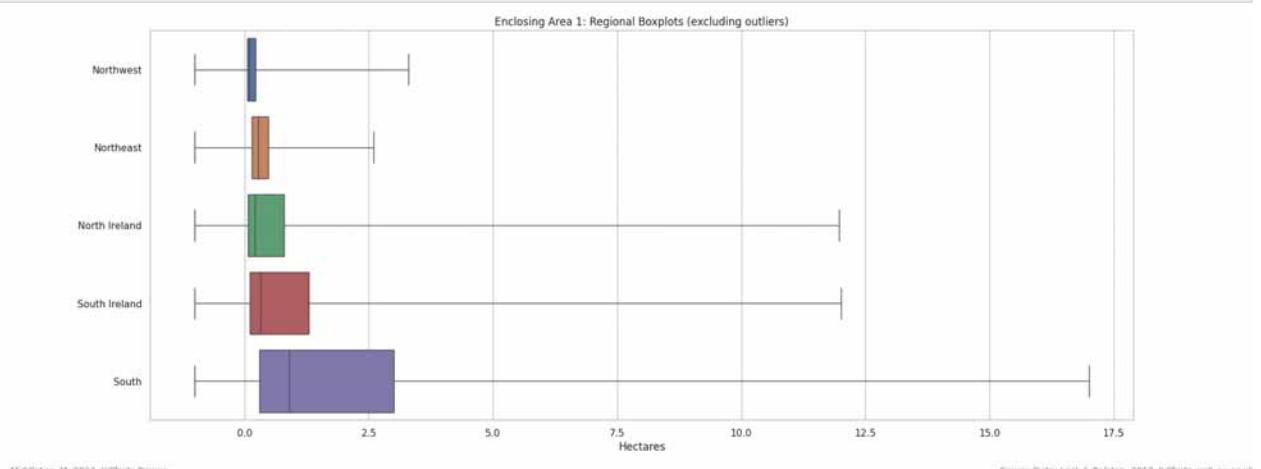
```
In [ ]: plot_bar_chart_numeric(s_ie_encl osi ng_area_1_data_clip, 1, \
                           'Encl osi ng_Area_1_clip', 'Count', \
                           'Encl osi ng_Area_1_clip (South Ireland)', \
                           int(encl osi ng_area_south_data[4]*3.8), '(hectares)')
```



Enclosing Area 1: Regional Boxplots

Removing outliers makes it easier to see the detail in the boxplots. They show a clear difference between the North (Scotland and N. England) and South (Wales and S. England). The North is dominated by small forts. The Northwest is notable for its tiny forts, of which the majority sit within a narrow range up to 0.22 Ha. In contrast, the South has a much larger range of fort areas, with an interquartile range between 0.3 and 3 Ha. North Ireland and South Ireland are similar with South Ireland differing in having slightly larger forts overall. The median size of forts in the Northeast and in Ireland are roughly similar ranging from 0.21 to 0.32 Ha. The Northwest are noticeably smaller, with a median of 0.09 Ha and the South are considerably larger, with a median of 0.9 Ha.

```
In [ ]: regional_dict = \
{'Northwest': location_enclising_data_nw['Enclosing_Area_1'], \
'Northeast': location_enclising_data_ne['Enclosing_Area_1'], \
'North Ireland': location_enclising_data_irland_n['Enclosing_Area_1'], \
'South Ireland': location_enclising_data_irland_s['Enclosing_Area_1'], \
'South': location_enclising_data_south['Enclosing_Area_1']}
plot_data = pd.DataFrame.from_dict(regional_dict)
plt.figure(figsize=(20,8))
ax = sns.boxplot(data=plot_data, orient="h", whis=[2.2, 97.8], showfliers=False);
add_annotation_plot(ax)
ax.set_xlabel('Hectares')
title = 'Enclosing_Area_1: Regional Boxplots (excluding outliers)'
plt.title(get_print_title(title))
save_fig(title)
plt.show()
```



Enclosing Area 1: Outliers

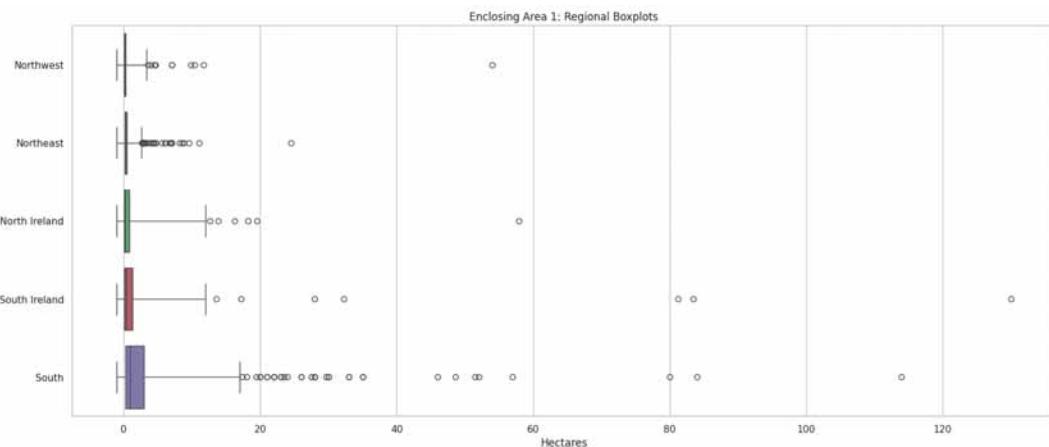
All the regions have outliers, and all have a small number of atypical, large forts. In general, these are very few in number and where outliers are present, they mostly cluster just above the main data range. The steps between outliers, noted in [Enclosing Area 1 - Outlier Distribution](#), are only visible in the south data package.

```
In [ ]: plt.figure(figsize=(20,8))
ax = sns.boxplot(data=plot_data, orient="h", whis=[2.2, 97.8], showfliers=True);
add_annotation_plot(ax)
ax.set_xlabel('Hectares')
```

```

title = 'Enclosing_Area_1: Regional Boxplots'
plt.title(get_pprint_title(title))
save_fig(title)
plt.show()

```



Enclosing Area 1 Mapped by Size

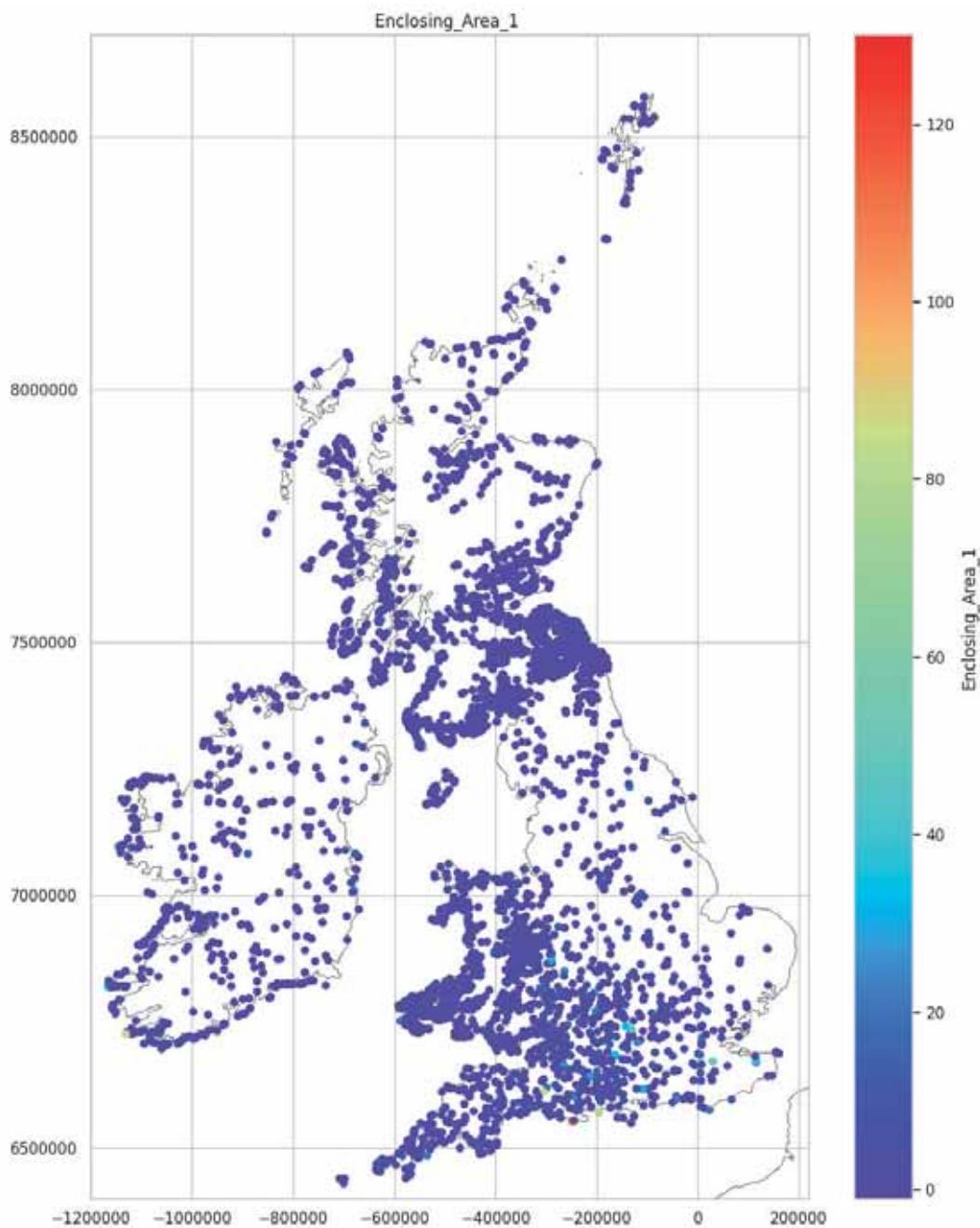
With most hillforts being less than 1 Ha and 'Enclosing Area 1' having a range up to 130 Ha, the resulting map, based on area, lacks clarity.

```

In [ ]: location_enclosing_data = \
pd.merge(location_numeric_data_short, enclosing_numeric_data, left_index=True, \
         right_index=True)

In [ ]: plot_valuses(location_enclosing_data, 'Enclosing_Area_1', 'Enclosing_Area_1')

```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

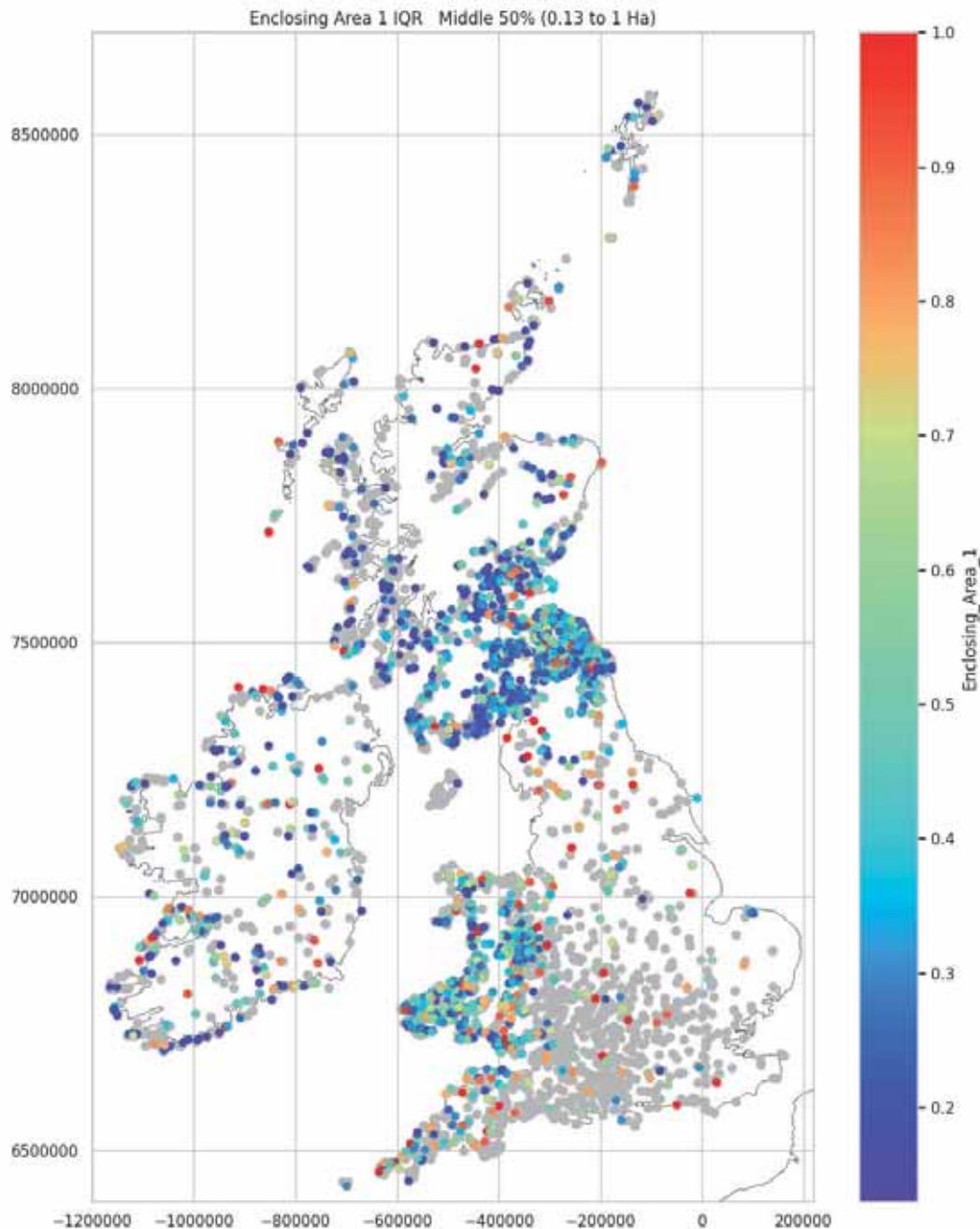
Enclosing Area 1 Interquartile Range (mid 50%) Mapped

Plotting the 50% of forts, from the mid-range of the boxplot (the IQR), shows a distribution across the Scottish Borders, the Welsh uplands, the southern end of the west coast of Scotland, the north coast of the South-west Peninsula, coastal sites around the south of Ireland and a peppering of other sites across central Ireland and NE Scotland. What is noticeable is the rarity of forts, in this range, across England. Those, that do fall in England, tend to be at the upper end of the area range. The hillforts in the interquartile range are located predominantly on the eastern Southern Uplands and the Cambrian Mountains.

```
In [ ]: encl osi ng_area_1_013_1 = location_encl osi ng_data.copy()
encl osi ng_area_1_013_1 = \
encl osi ng_area_1_013_1[encl osi ng_area_1_013_1['Encl osi ng_Area_1'].between(0.13, 1)]
encl osi ng_area_1_013_1['Encl osi ng_Area_1'].describe()
```

```
Out[ ]: count    2100.000000
mean      0.408538
std       0.231068
min       0.130000
25%       0.230000
50%       0.340000
75%       0.550000
max       1.000000
Name: Encl osi ng_Area_1, dtype: float64
```

```
In [ ]: plot_type_values(enclosing_area_1_013_1, 'Enclosing_Area_1', 'Enclosing_Area_1', \
extra='IQR - Middle 50% (0.13 to 1 Ha)')
```

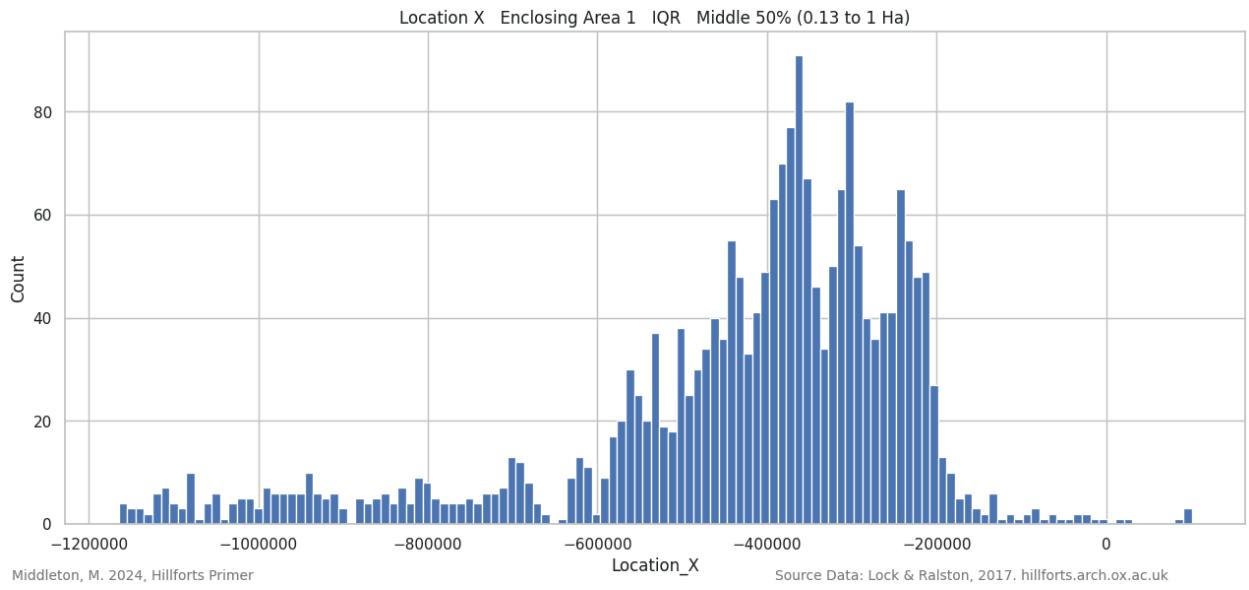


Enclosing Area 1 Interquartile Range (mid 50%) Location_X Plotted

The density peaks towards the east.

```
In [ ]: plot_histogram(enclosing_area_1_013_1['Location_X'], 'Location_X', \
'Location_X - Enclosing_Area_1 - IQR - Middle 50% (0.13 to 1 Ha)', \
10000)
```

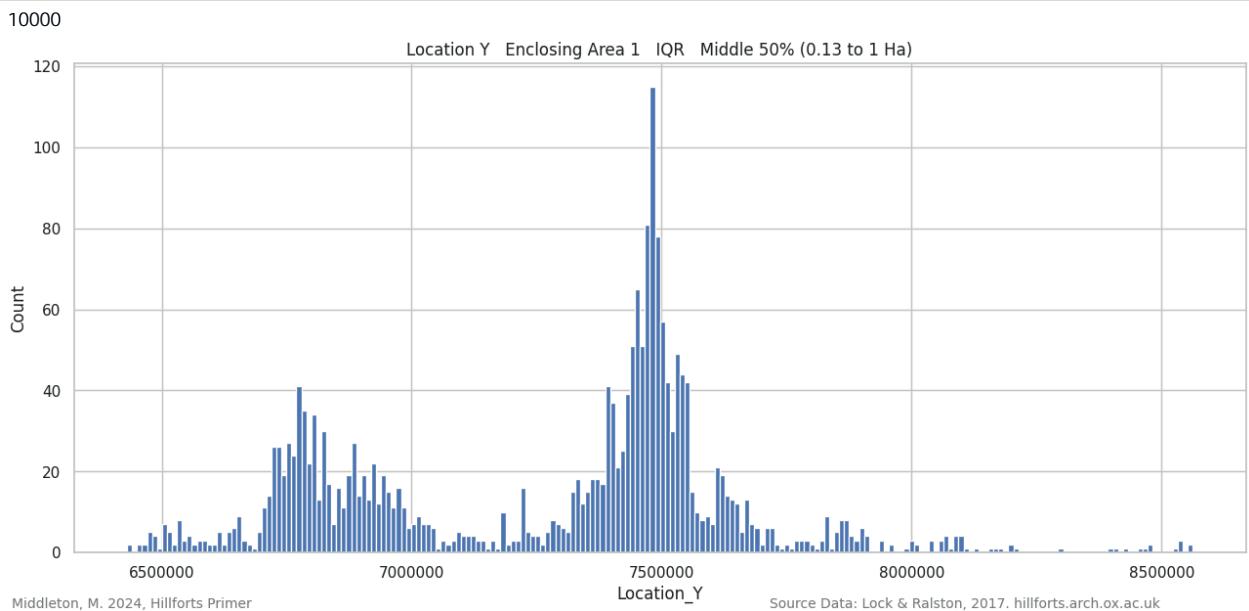
10000



Enclosing Area 1 Interquartile Range (mid 50%) Location_Y Plotted

Plotting the distribution against the Location_Y axis (the northing) shows the peak to the North to be nearly three times that in the South.

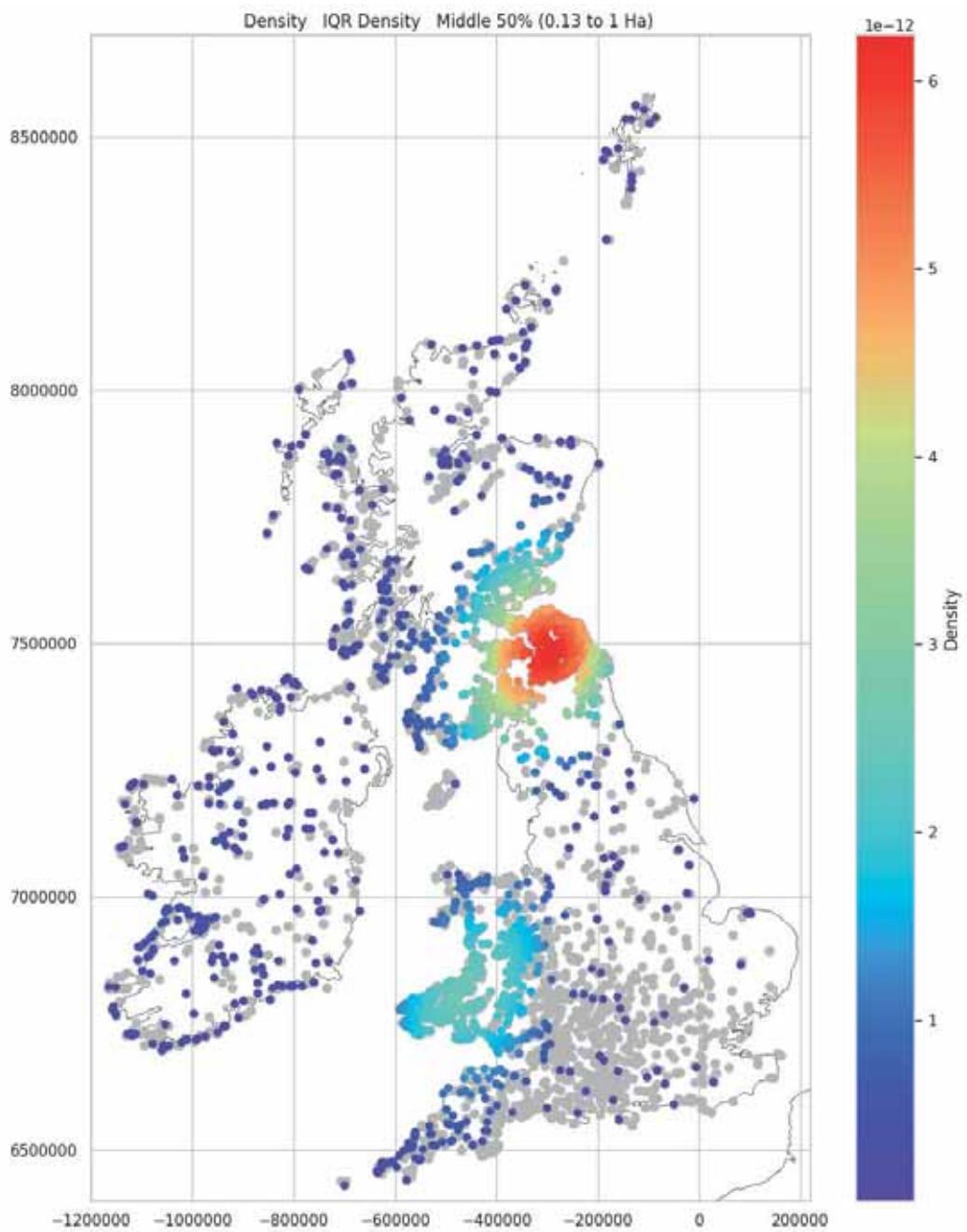
```
In [ ]: plot_histogram(enclosing_area_1_013_1['Location_Y'], 'Location_Y', \
                      'Location_Y - Enclosing_Area_1 - IQR - Middle 50% (0.13 to 1 Ha)', 10000)
```



Enclosing Area 1 Interquartile Range (mid 50%) Density Mapped

The density plot of the interquartile range shows a very intense cluster in the Northeast and a secondary cluster over the southern end of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(enclosing_area_1_013_1, \
                           'IQR Density - Middle 50% (0.13 to 1 Ha)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Area 1 First (lower) and Forth (Upper) Quarters (excluding outliers) Mapped

Mapping the first quarter of sites by area (the blues in the figure below), shows a distribution along the west coast of Scotland and the south, west and north coasts of Ireland. Additionally, there are forts along the Great Glen and forts along the south coast of Fife, and up into Perthshire and Angus. In Wales, there are small clusters of forts at a couple of locations along the west coast and along the Brecon Beacons. There are very few, from this range, within the mainland of Ireland, England or eastern and northern Wales.

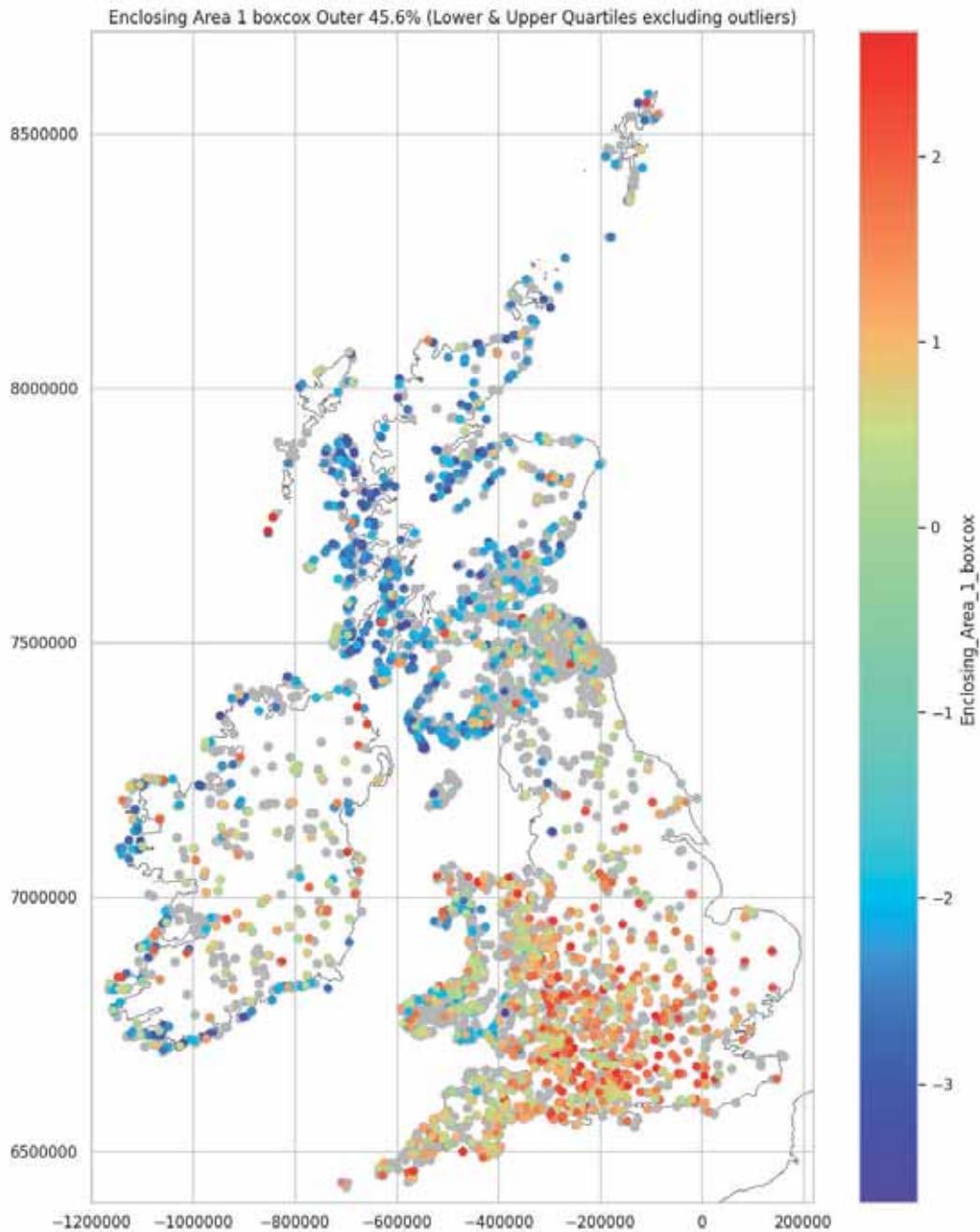
Mapping the fourth quarter, (the greens to red), shows a distribution of sites across eastern Wales, south-central England and into the Southwest Peninsula. There are a sprinkling of sites across the uplands of northern England and eastern Scotland, with a similar concentration across central and southern Ireland. There are noticeably few across western and northern Scotland.

```
In [ ]: from scipy import stats
```

```
In [ ]: enclising_area_1_temp = location_enclising_data.copy()
enclising_area_1_low = \
enclising_area_1_temp[(enclising_area_1_temp['Enclising_Area_1'] >= 0) & \
(enclising_area_1_temp['Enclising_Area_1'] < 0.13)]
enclising_area_1_high = \
enclising_area_1_temp[(enclising_area_1_temp['Enclising_Area_1'] > 1) & \
(enclising_area_1_temp['Enclising_Area_1'] <= 10.5)]
```

```
encl osi ng_area_1_high_low = pd.concat([encl osi ng_area_1_low, \
                                         encl osi ng_area_1_high])
encl osi ng_area_1_hi gh_low['Encl osi ng_Area_1_boxcox'] = \
stats.boxcox(encl osi ng_area_1_hi gh_low['Encl osi ng_Area_1'])[0]
```

```
In [ ]: plot_type_val ues(encl osi ng_area_1_hi gh_low, 'Encl osi ng_Area_1_boxcox', \
                           'Encl osi ng_Area_1 (Boxcox)', \
                           extra='Outer 45.6% (Lower & Upper Quartiles excluding outliers)')
```



Middleton, M. 2024, Hillforts Primer

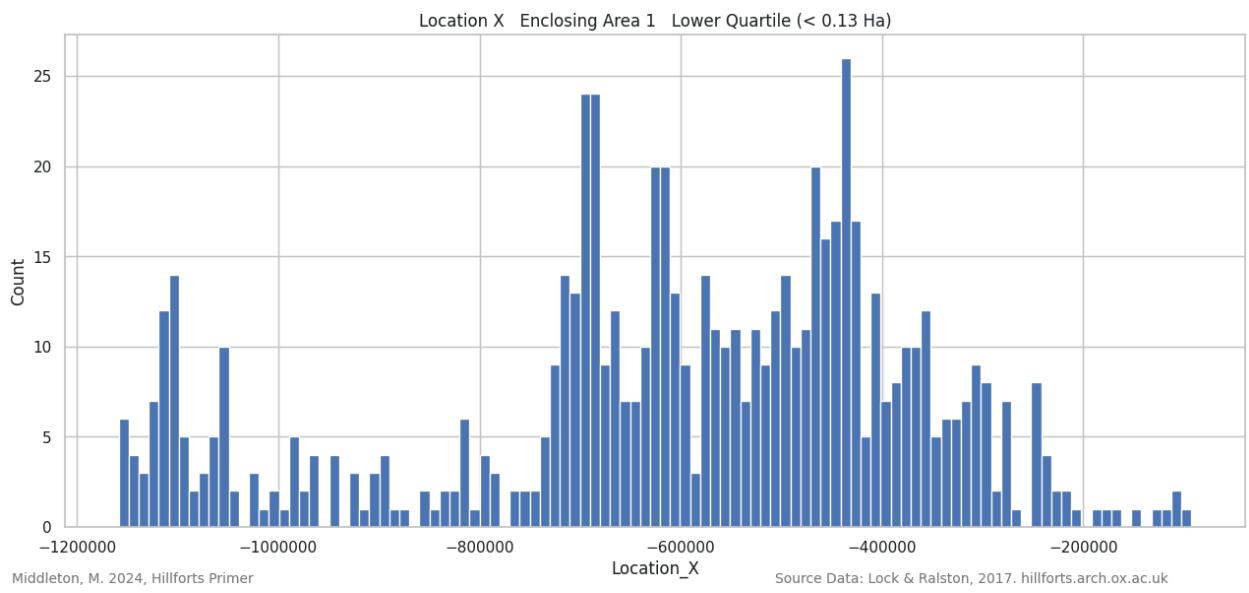
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Area 1 Lower Quartile (excluding outliers) Location_X Plotted

The Location_X data shows peaks in the data to the west and south of Ireland then three peaks across Scotland. Two over the western seaboard and one around the Great Glen.

```
In [ ]: plot_hi stogram(encl osi ng_area_1_low['Locati on_X'], 'Locati on_X', \
                           'Locati on_X - Encl osi ng_Area_1 - Lower Quartile (< 0.13 Ha)', \
                           10000)
```

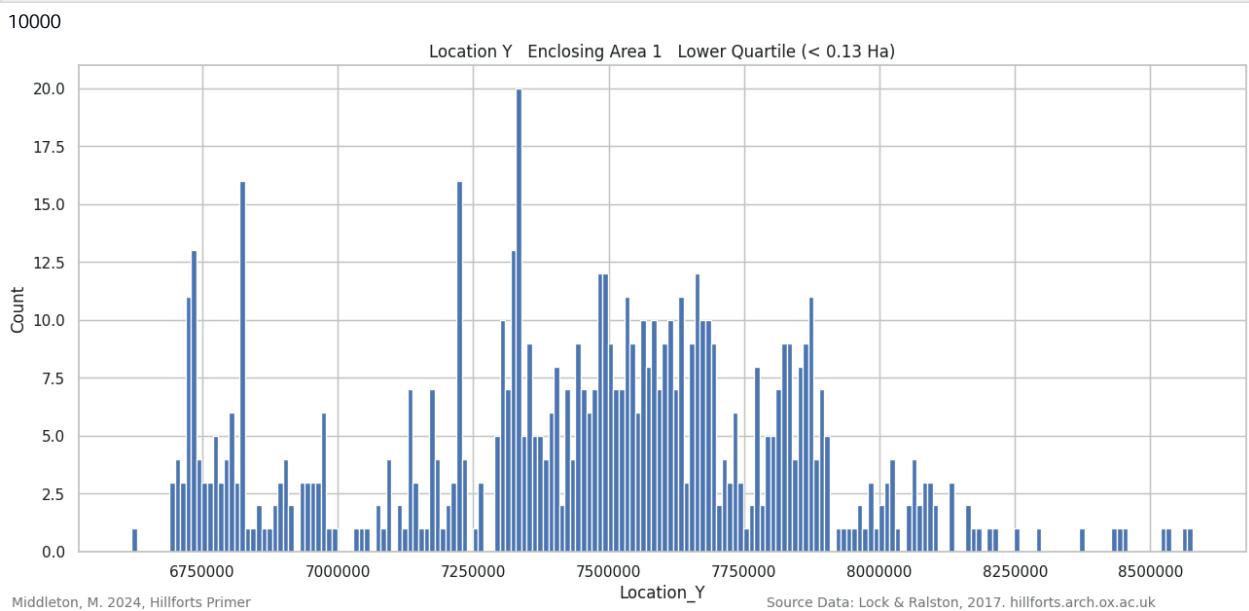
10000



Enclosing Area 1 Lower Quartile (excluding outliers) Location_Y Plotted

In the Location_Y axis there are peaks over the south coast of Ireland and a high peak aligning with the south coast of Galloway. This high peak projects from a broad, lower peak, running the length of the west coast of Scotland, up to Skye.

```
In [ ]: plot_hi_stogram(enclosing_area_1_low['Location_Y'], \
                      'Location_Y - Enclosing_Area_1 - Lower Quartile (< 0.13 Ha)', \
                      10000)
```

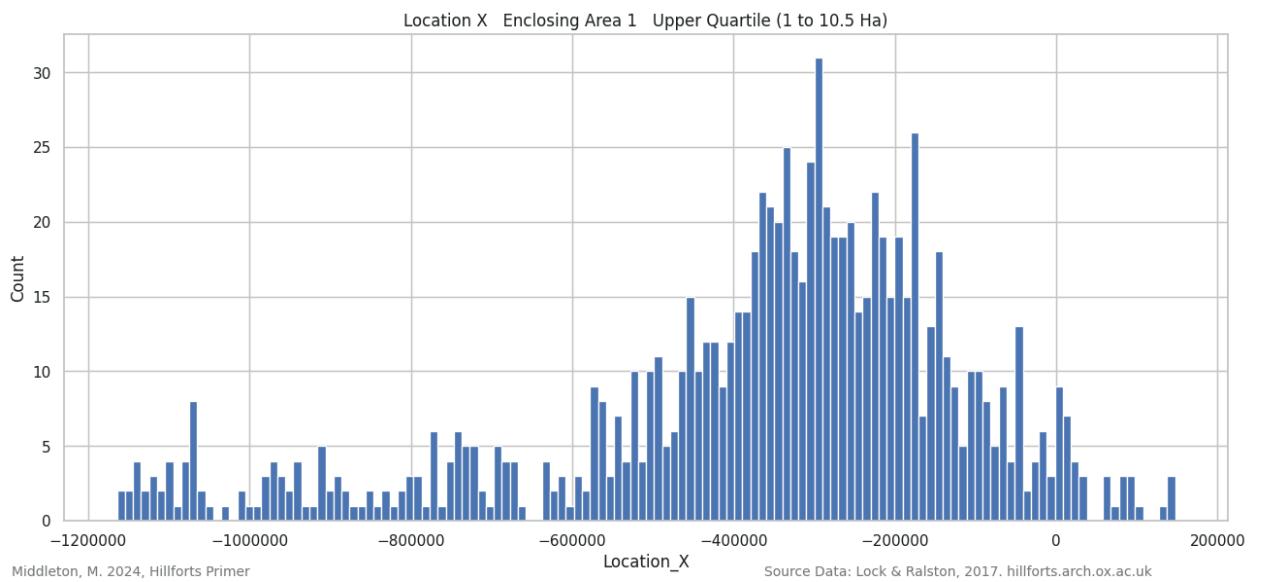


Enclosing Area 1 Upper Quartile (excluding outliers) Location_X Plotted

The fourth quartile Location_X data shows the cluster in southern England to have a broad peak.

```
In [ ]: plot_hi_stogram(enclosing_area_1_high['Location_X'], \
                      'Location_X', \
                      'Location_X - Enclosing_Area_1 - Upper Quartile (1 to 10.5 Ha)', \
                      10000)
```

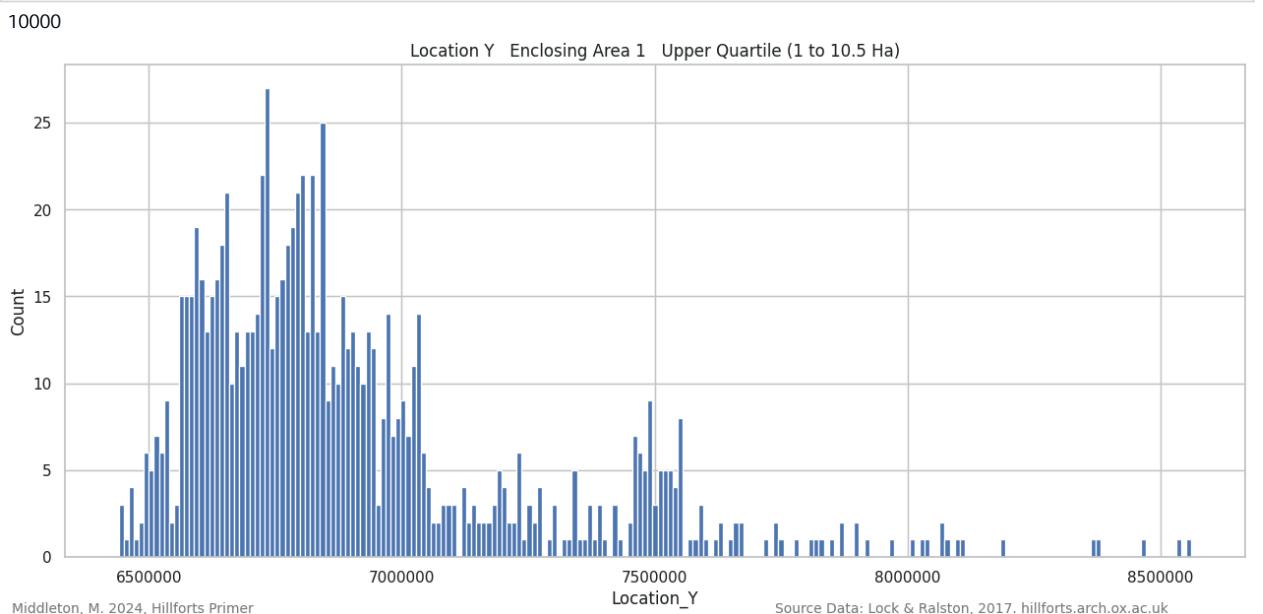
10000



Enclosing Area 1 Upper Quartile (excluding outliers) Location_Y Plotted

In the Location_Y data, although there is a small cluster over the Southern Uplands, the main peak, in the South, is broad and tall.

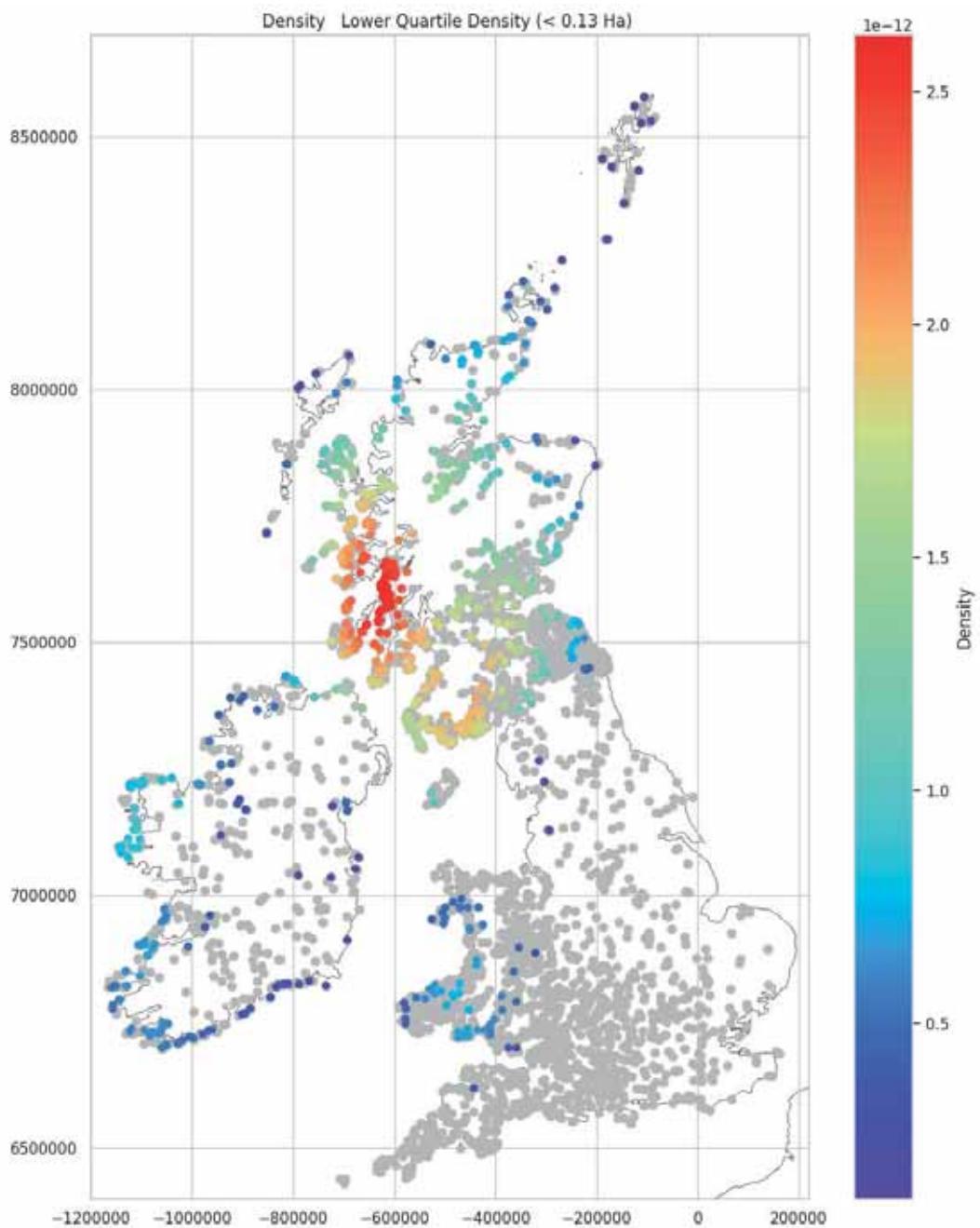
```
In [ ]: plot_hi_stogram(encl osing_area_1_hi gh['Locati on_Y'], \
'Locati on_Y', \
'Locati on_Y - Encl osing_Area_1 - Upper Quartile (1 to 10.5 Ha)', \
10000)
```



Enclosing Area 1 Lower Quartile Density Mapped (excluding outliers)

The clusters in the first (lower) quartile are striking. The plot is dominated by the cluster over the western seaboard of Scotland with an unmistakable focus around SC2466: Dunadd. There is a secondary concentration over Galloway and up into the Carsphairn and Lowther hills and a notable cluster toward the eastern end of the Great Glen. In Ireland there is a small cluster on the west coast and there is a small cluster in southern Wales, but it is sparse. Apart from the clusters the other notable feature of this distribution are the areas across England, north Wales and the Southwest where there are almost no forts of this class. Similarly, in Ireland the distribution is very much concentrated around the south and west coast with only a sparse peppering of hillforts inland.

```
In [ ]: plot_densi ty_over_grey(encl osing_area_1_low, \
'Lower Quartile Densi ty (< 0.13 Ha)')
```



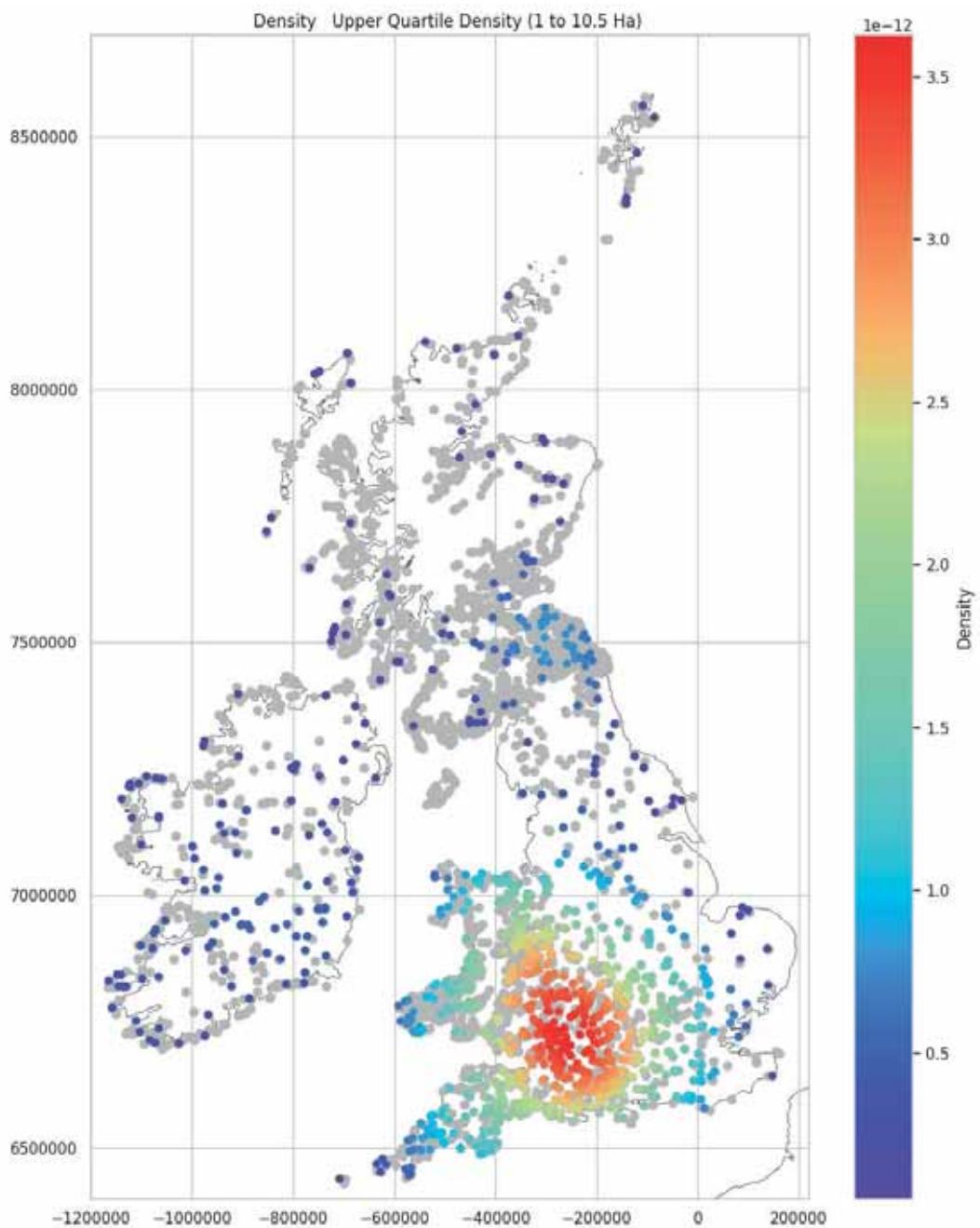
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Area 1 Upper Quartile Density Mapped (excluding outliers)

The fourth (upper) quartile is equally striking with a wide cluster focussed over south central England and running into Wales and the Southwest.

```
In [ ]: plot_density_over_grey(enclosing_area_1_high, \
    'Upper Quartile Density (1 to 10.5 Ha)')
```



Middleton, M. 2024, Hillforts Primer

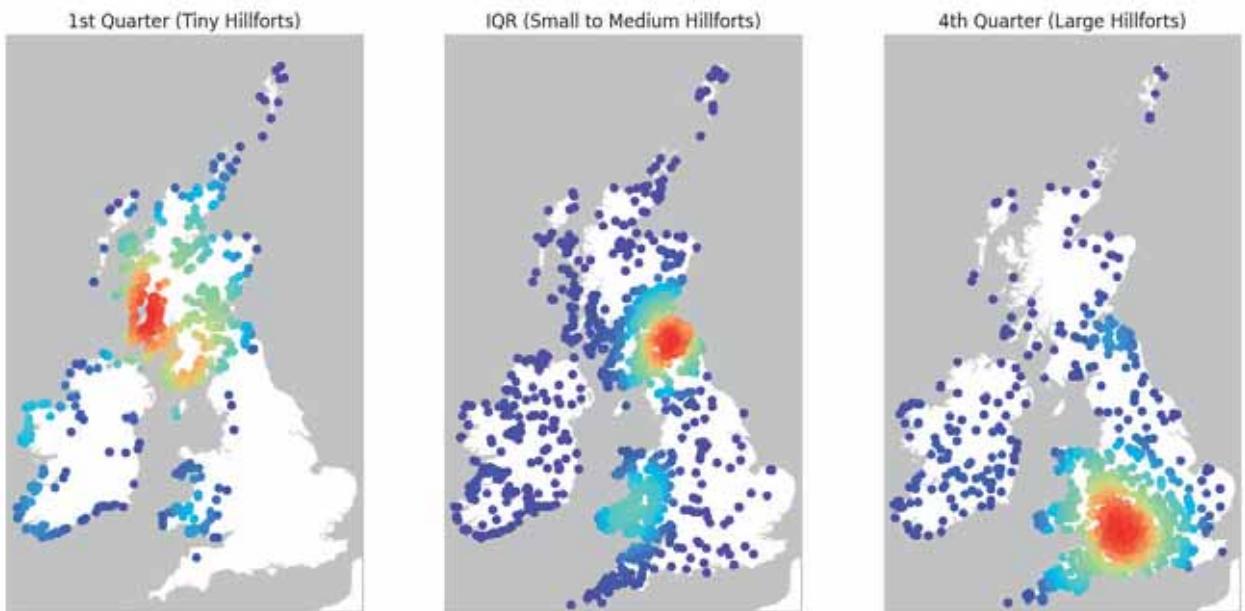
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Area 1 Density Summary

The analysis of the Enclosing Area 1 data highlights four, possibly five different clusters. In the 1st quarter, mapping the density of tiny hillforts, there is one intense cluster in the Northwest and a smaller, almost indistinguishable cluster, in the west of Ireland, along the Duvillaun, Achill and Inishkea islands. In the central interquartile range (IQR), of small to medium sized hillforts, there are two more clusters. Here, the most intense cluster is in the Northeast and the smaller, secondary cluster, is in southern Wales. In the 4th quarter, mapping large hillforts, there is one large cluster over south central England. Equally notable are the areas where there are large gaps in the distribution. In the 1st quarter, England, north Wales and the Southwest have almost no recorded tiny hillforts while, less surprisingly, the Highlands and the west coast of Scotland have very few large hillforts.

```
In [ ]: plot_density_over_grey_three(enclosing_area_1_low, \
enclosing_area_1_013_1, enclosing_area_1_high, \
'Enclosing Area 1 Density')
```

Enclosing Area 1 Density



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Enclosing Area 1 Outlier Distribution Mapped (Over 10.5 Ha)

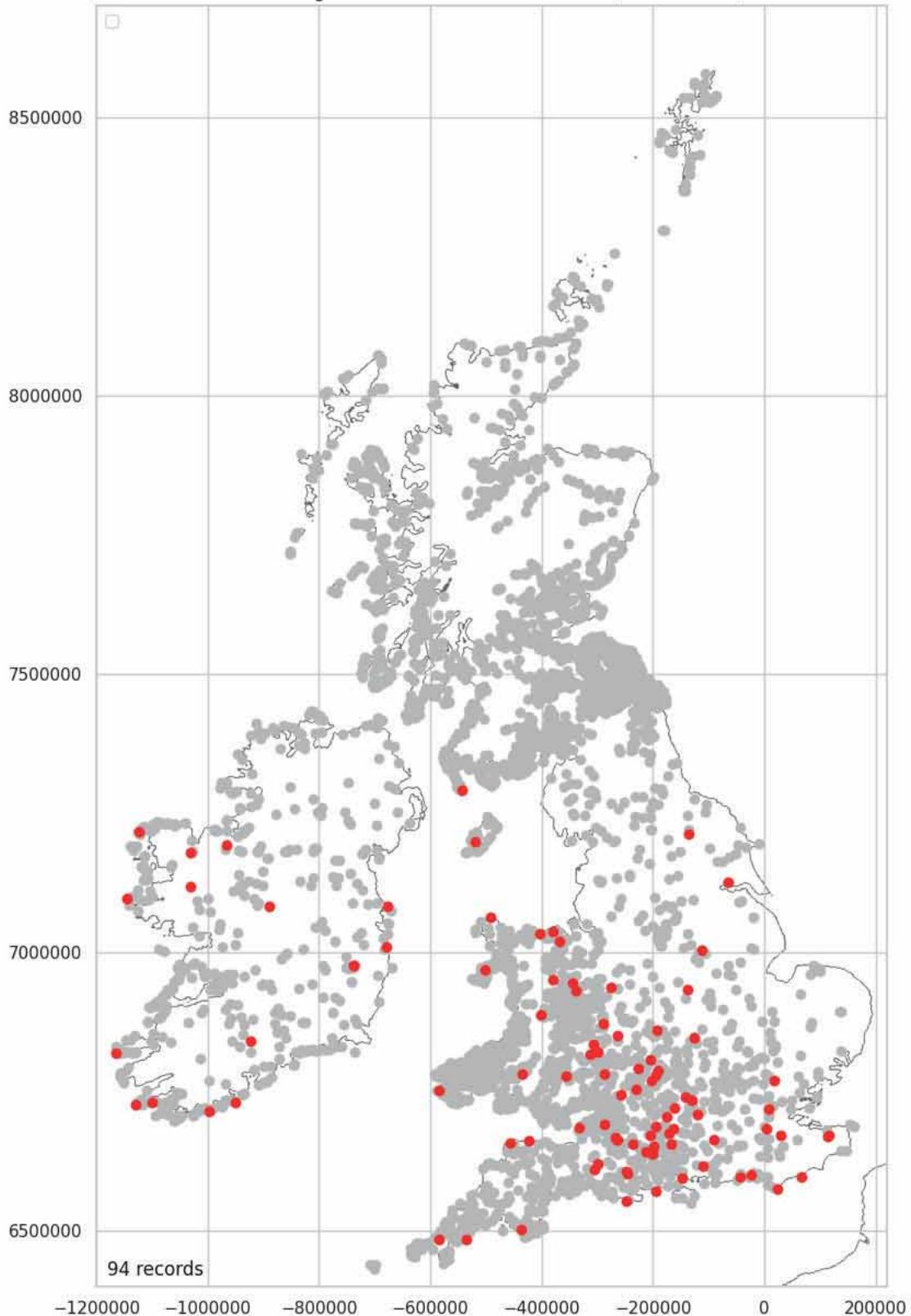
There are 94 outliers that range in size from 10.5 to 130 Ha. Most are located in south central England and 16 in south, central Ireland; There is one in Galloway and one on the Isle of Man.

```
In [ ]: enclosing_area_1_105 = location_enveloping_data.copy()
enclosing_area_1_105 = \
enclosing_area_1_105[enclosing_area_1_105['Enveloping_Area_1']>=10.5]
enclosing_area_1_105['Enveloping_Area_1'].describe()
```

```
Out[ ]: count    94.000000
mean     25.221489
std      22.195230
min     10.500000
25%     12.000000
50%     16.635000
75%     28.000000
max     130.000000
Name: Enveloping_Area_1, dtype: float64
```

```
In [ ]: plot_over_grey_numeric(enclosing_area_1_105, 'Enveloping_Area_1', \
                           'Enveloping_Area_1 Distribution All Outliers (over 10.5 Ha)')
```

Enclosing Area 1 Distribution All Outliers (over 10.5 Ha)



Middleton, M. 2024, Hillforts Primer

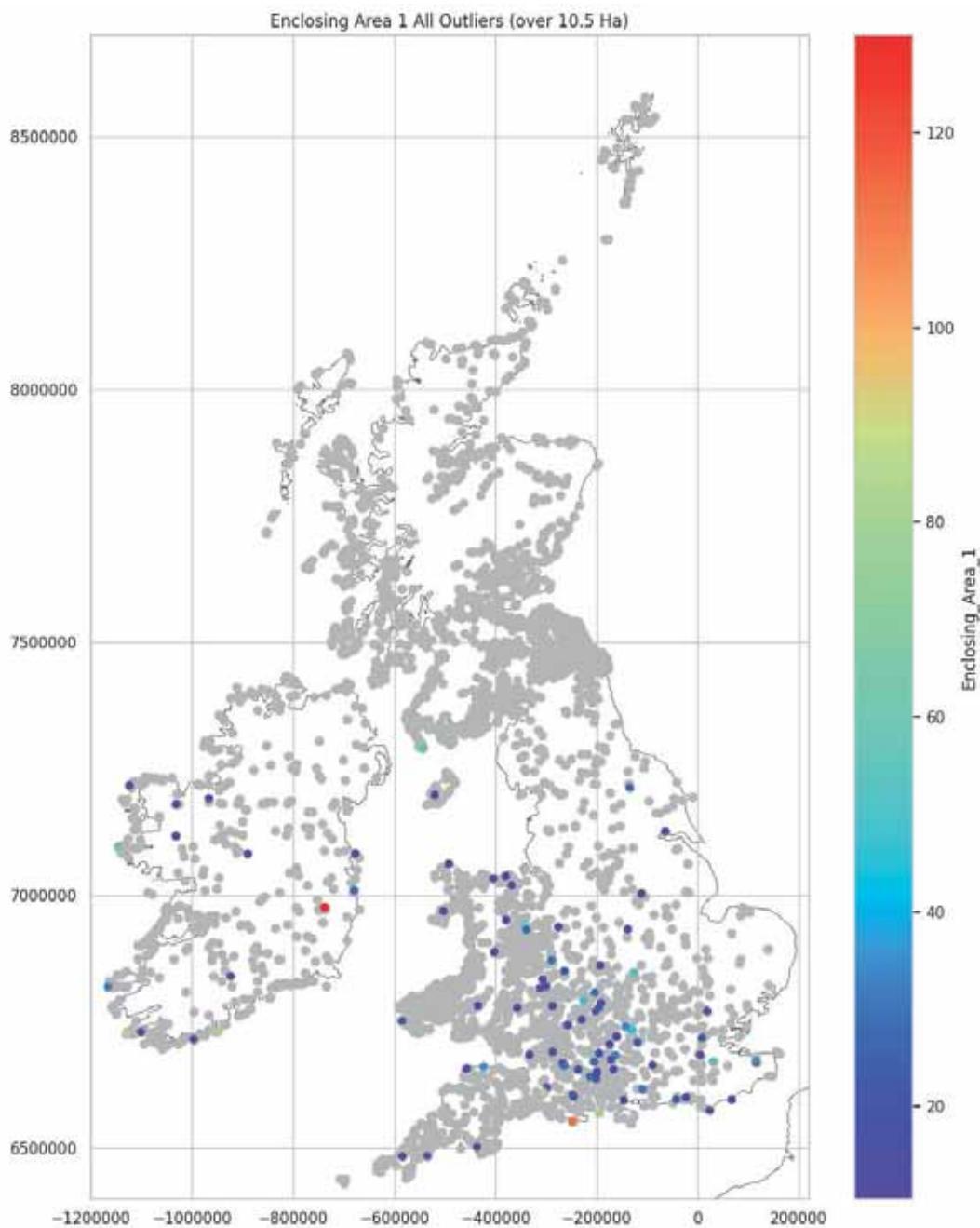
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2.27%

Enclosing Area 1 Outliers Mapped by Size (Over 10.5 Ha)

Within the outliers over 10.5 Ha, there are two very large hillforts over 100 Ha. Otherwise, most are around 20 Ha or less. In the mid-range the plot highlights an alignment of forts, over 40 Ha running from the Thames up toward north Wales (light blue).

```
In [ ]: plot_type_values(enclosing_area_1_105, 'Enclousing_Area_1', \
    'Enclousing_Area_1', extra='All Outliers (over 10.5 Ha)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Area 1: Distribution of Outliers Over 21 Ha Mapped

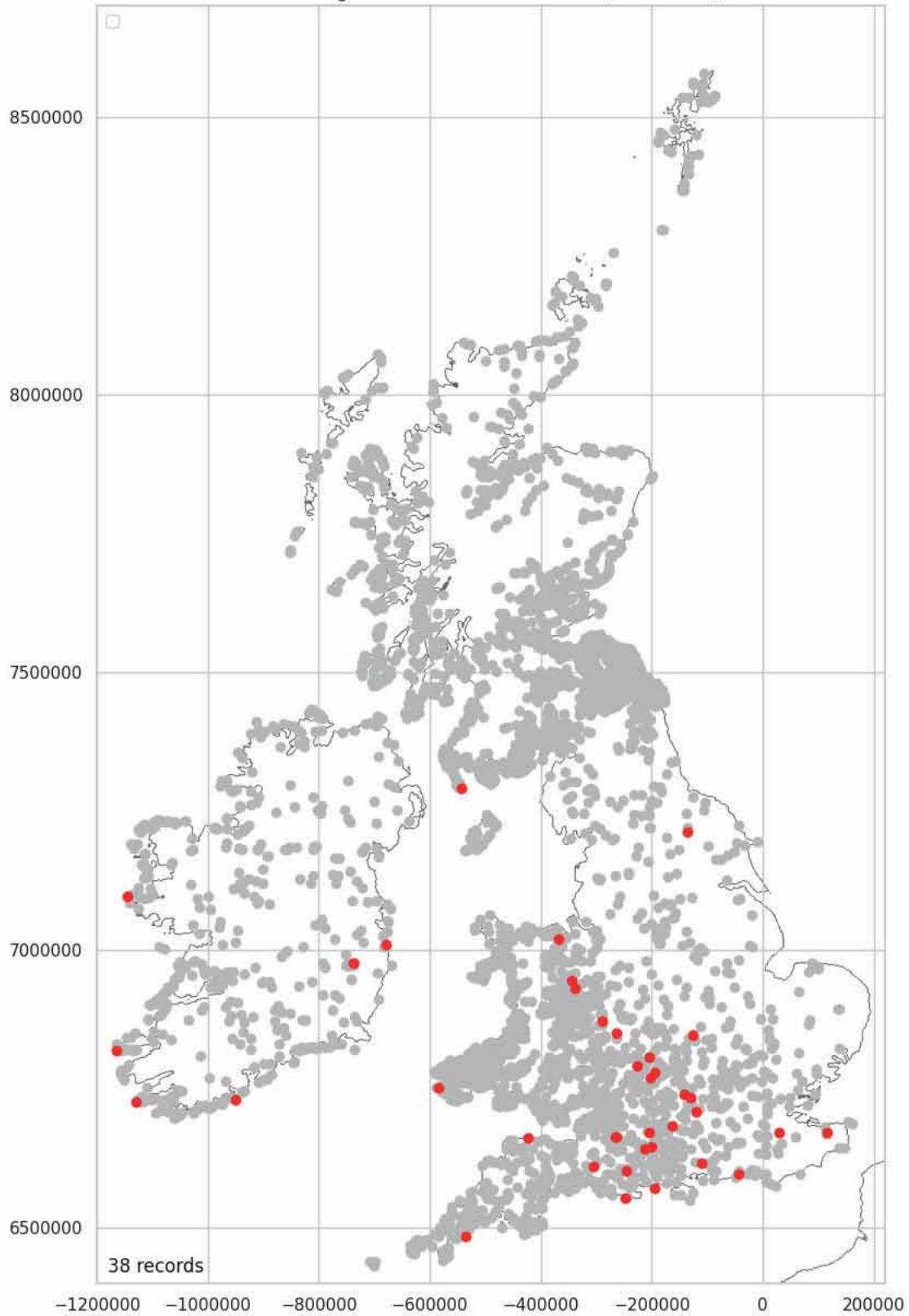
After multiple tests to filter the data for forts over various sizes, a possible alignment of hillforts was isolated for forts over 21 Ha. See [Appendix 1](#) where the straight section of the alignment from (1155) Penycloddiau, Denbighshire (Pen y Cloddiau) to (139) Bozedown Camp, Oxfordshire (Binditch) is hypothesis tested and show this alignment as likely to be meaningful.

```
In [ ]: enclosing_area_1_21 = location_enveloping_data.copy()
enclosing_area_1_21 =
enclosing_area_1_21[enclosing_area_1_21['Enveloping_Area_1'] >= 21]
enclosing_area_1_21['Enveloping_Area_1'].describe()
```

```
Out[ ]: count    38.000000
mean     42.501053
std      26.680691
min     21.000000
25%     24.875000
50%     30.000000
75%     51.875000
max     130.000000
Name: Enveloping_Area_1, dtype: float64
```

```
In [ ]: plot_over_grey_numeric(enclosing_area_1_21, 'Enveloping_Area_1', \
                           'Enveloping_Area_1 Distribution Outliers (over 21 Ha)', '')
```

Enclosing Area 1 Distribution Outliers (over 21 Ha)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0. 92%

Enclosing Area 1 Hillforts Over 21 Ha Mapped by Size

Clipping the maximum area at 80 Ha allows variation in the smaller sites to be visible. What can be seen is that most hillforts in the band from the Thames to north Wales are at the lower end of the area range. These are interspersed with forts in the mid-range (blue green). The largest forts are on or near the south coast or in Ireland.

```
In [ ]: enclosing_area_1_21_clip = enclosing_area_1_21.copy()
enclosing_area_1_21_clip['Enclosing_Area_1_clip'] = \
enclosing_area_1_21_clip['Enclosing_Area_1'].\
```

```

clip(enclosing_area_1_21_clip['Enclosing_Area_1'], 80, axis=0)
enclosing_area_1_21_clip['Enclosing_Area_1_clip'].describe()

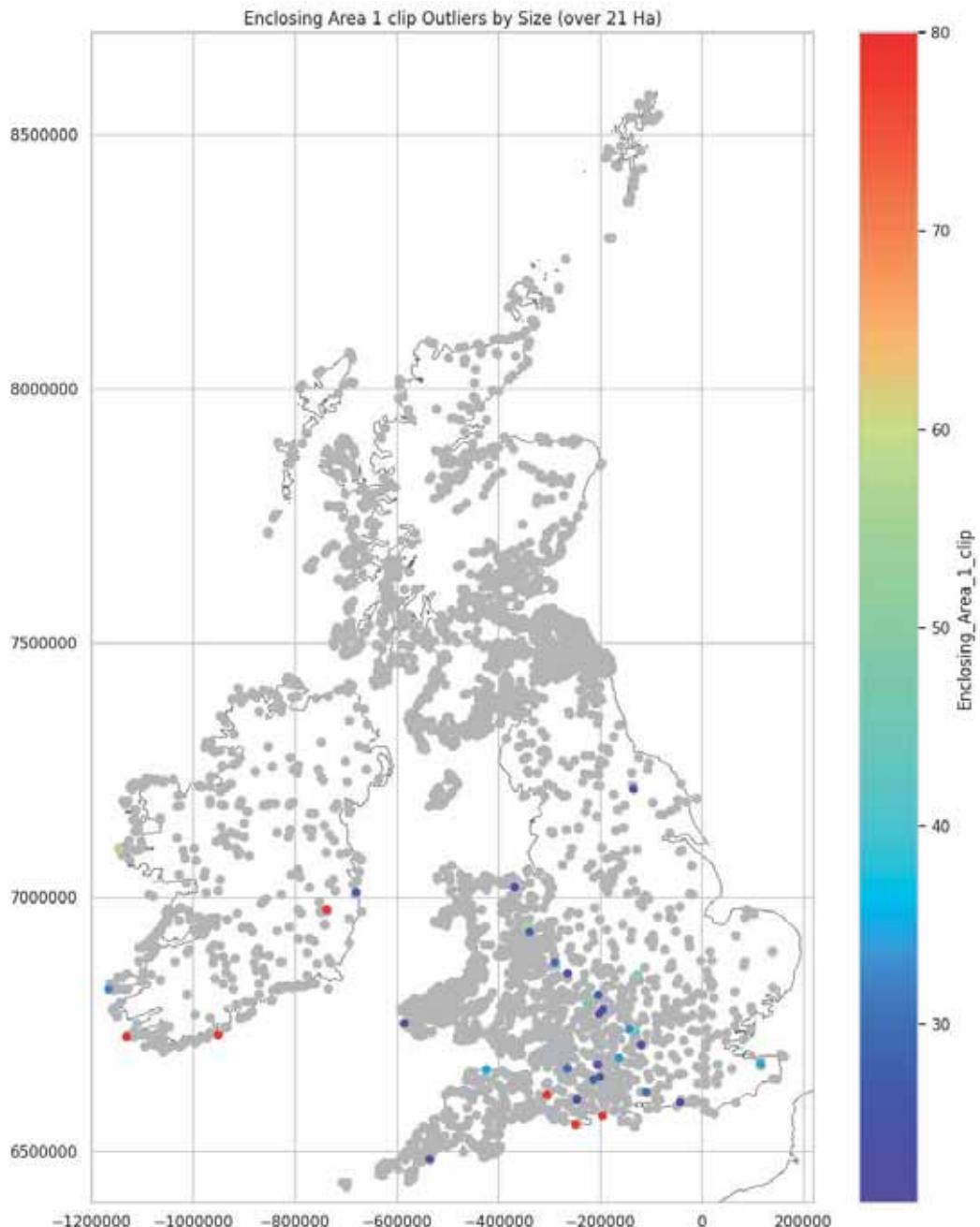
```

Out[]:

count	38.000000
mean	40.061579
std	20.450771
min	21.000000
25%	24.875000
50%	30.000000
75%	51.875000
max	80.000000

Name: Enclosing_Area_1_clip, dtype: float64

```
In [ ]: plot_type_values(enclosing_area_1_21_clip, 'Enclosing_Area_1_clip', \
                           'Enclosing_Area_1_clip', extra='Outliers by Size (over 21 Ha)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

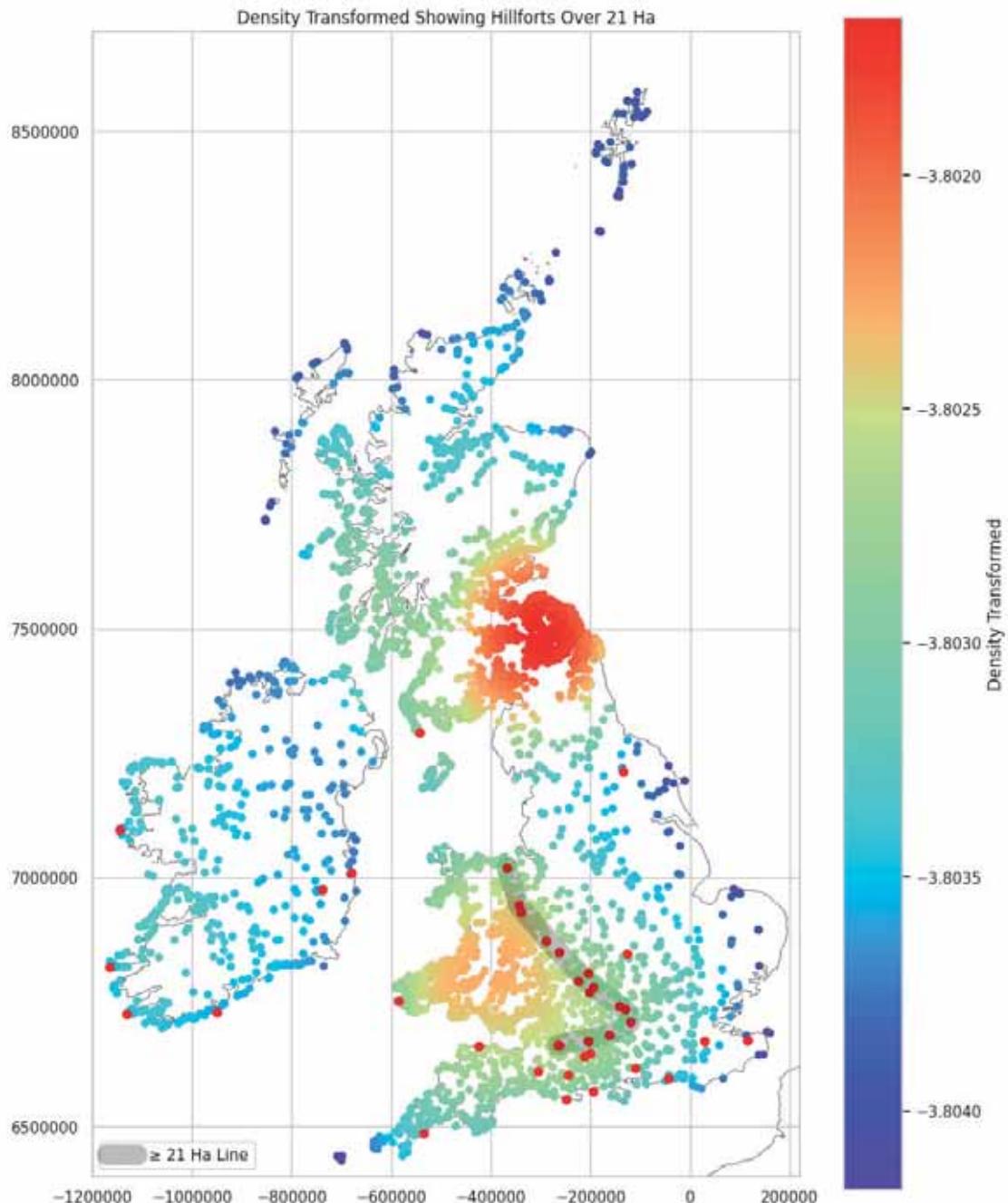
Enclosing Area 1: Hillfort Density Transformed Overlayed by Hillforts Over 21 Ha

Plotting the hillforts over 21 Ha against the hillfort density shows the alignment of forts sit along the eastern fringe of the southern density cluster. The forts are located roughly along the transition from the orange to green on the density map (-3.8025). This line has been annotated the, ' ≥ 21 Ha Line'.

```
In [ ]: transformed_location_numeric_data_short = location_data.copy()
transformed_location_numeric_data_short['Density_trans'] = best_lambda = \
475
```

```
stats.boxcox(transformed_location_numeric_data_short['Density'])
```

```
In [ ]: density_scatter_lines(transformed_location_numeric_data_short, \
    enclosing_area_1_21, \
    'Density Transformed Showing Hillforts Over 21 Ha', True)
```



Middleton, M. 2024, Hillforts Primer

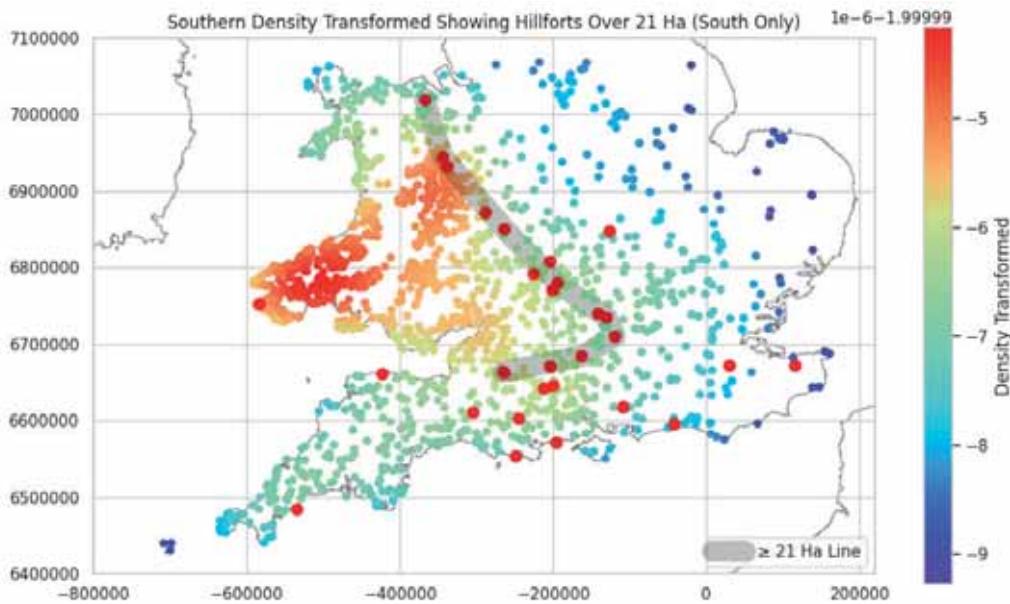
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Enclosing Area 1: Southern Hillfort Density Transformed overlayed by hillforts over 21 Ha

The same map showing only the southern data.

```
In [ ]: cluster_south = south.copy()
enclosing_area_1_21_s = \
enclosing_area_1_21[enclosing_area_1_21['Location_X'] > -600000]
cluster_south = add_density(cluster_south)
cluster_south['Density_trans'] = stats.boxcox(cluster_south['Density'], 0.5)
```

```
In [ ]: south_density_scatter_lines(cluster_south, enclosing_area_1_21_s, \
    'Southern Density Transformed Showing Hillforts Over 21 Ha (South Only)', \
    True, False)
```



Middleton, M. 2024, Hillforts Primer

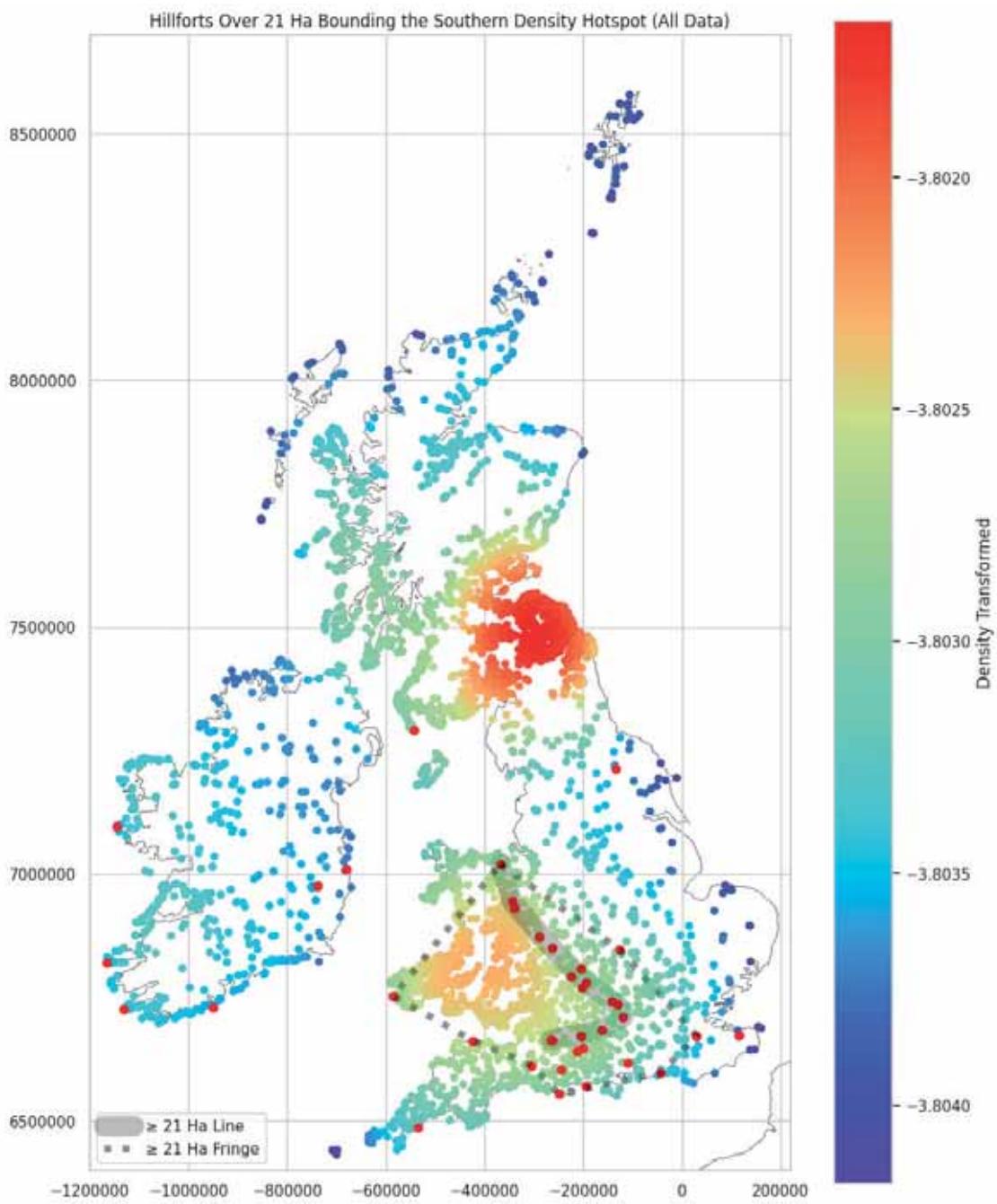
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Enclosing Area 1: Hillforts over 21 Ha Bounding the Southern Density Cluster

Most of the remaining outliers over 21 Ha in the South are located on the fringe of the southern density cluster. These have been annotated as the, ' ≥ 21 Ha Fringe'.

Both lines are based on a very small number of hillforts and are therefore highly speculative. There are only 38 hillforts greater than or equal to 21 Ha. This equates to 0.92% of all hillforts. These are not just the outliers (which are classified as lying in the outer 4.4% of a distribution), these are the outliers within the outliers. They are the most unusual hillforts in terms of Enclosing Area 1. For these hillforts to be distributed in such a uniform alignment is highly unlikely and can be shown not to be a random distribution in Appendix 1. For this class of forts to align with the edge of the most intense concentration of hillforts, seen in the southern density cluster, supports the idea that this alignment is not a coincidence. These hillforts seem to be positioned for a purpose. Could these be forts on a frontier between two cultural groups or perhaps these are forts focussed on trade, capable of hosting large gatherings of people and animals? It is hoped these observations will encourage a more detailed analysis.

```
In [ ]: density_scatter_lines(transformed_location_numeric_data_short, \
                           enclosing_area_1_21, \
                           'Hillforts Over 21 Ha Bounding the Southern Density Hotspot (All Data)', \
                           True, True)
```



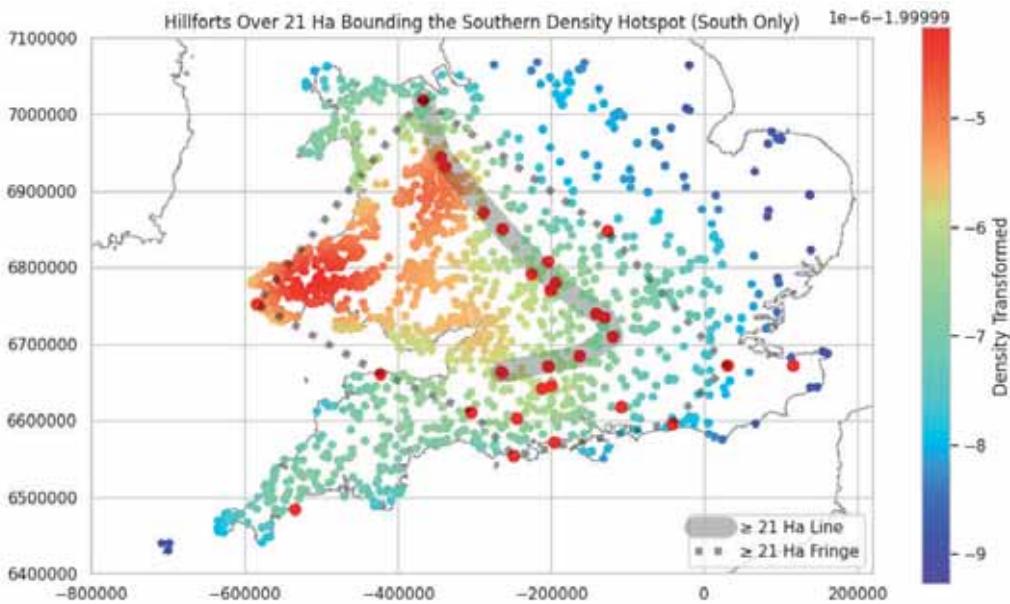
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Enclosing Area 1: Hillforts over 21 Ha Bounding the Southern Density Cluster (South Only)

The same plot showing only the southern data.

```
In [ ]: south_density_scatter_lines(cluster_south, enclosing_area_1_21_s, \
    'Hillforts Over 21 Ha Bounding the Southern Density Hotspot (South Only)', \
    True, True)
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

A full list of hillforts, of 21 hectares and over, in the southern data package.

```
In [ ]: greater_than_21ha_south = pd.merge(name_and_number, enclosing_area_1_21_s, \
                                     left_index=True, right_index=True)
greater_than_21ha_south[['Main_Atlas_Number', 'Main_Display_Name', 'Enclosing_Area_1', \
                        'Location_X', 'Location_Y']].sort_values(by='Enclosing_Area_1').\\
style.hide_index()

<ipython-input-248-398796f1c553>:6: FutureWarning: this method is deprecated in favour of `Styler.hide(axis="index")`  

style.hide_index()
```

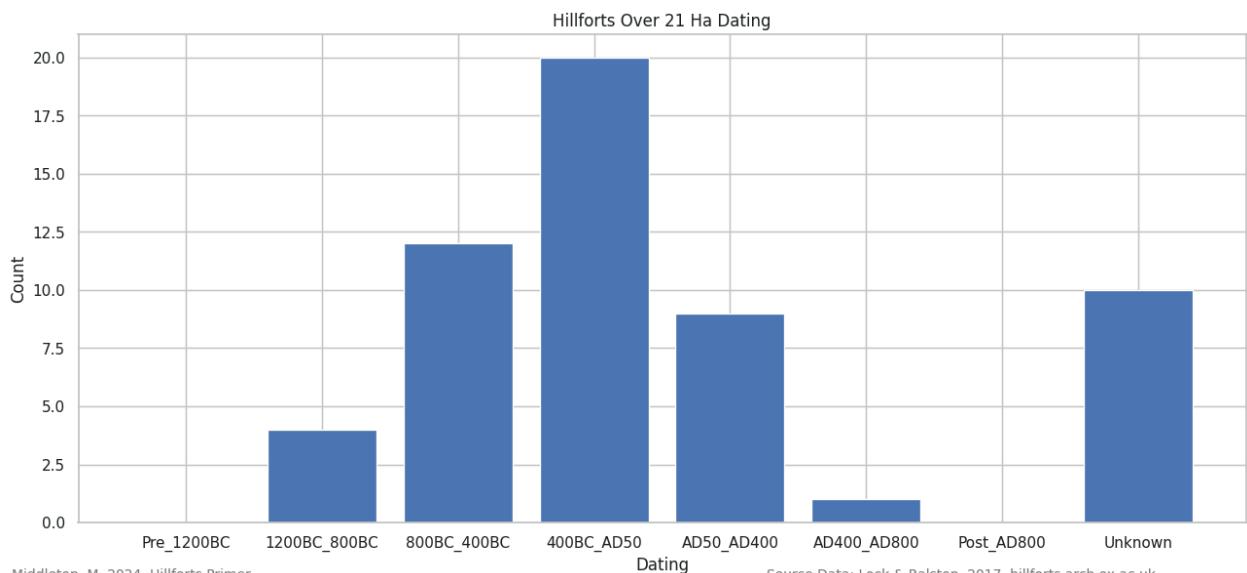
Main_Atlas_Number	Main_Display_Name	Enclosing_Area_1	Location_X	Location_Y
1155	Penycloddiau, Denbighshire (Pen y Cloddiau)	21.000000	-367969	7019842
643	Dodman Castle, Cornwall (Dodman Point; The Dodman)	21.000000	-534770	6485285
753	Norbury Camp, Northleach, Gloucestershire	22.000000	-202278	6770873
1997	Wooltack Point, Pembrokeshire (Deer Park Fort)	22.000000	-584307	6752378
3595	Hod Hill, Dorset	22.000000	-245476	6602754
757	Salmonsbury Camp, Gloucestershire	23.000000	-194654	6779602
367	Woodbury, Great Witley, Worcestershire (Woodbury Hill)	23.000000	-263690	6850593
139	Bozedown Camp, Oxfordshire (Binditch)	23.500000	-119597	6710127
3749	Cissbury Ring, West Sussex (Cissbury Camp)	24.000000	-42657	6596650
1504	Roulston Scar, North Yorkshire (Sutton Bank; Casten Dike)	24.500000	-134884	7213231
404	Ogbury Camp, Wiltshire	26.000000	-20007	6646748
461	Tedbury Camp, Somerset	26.000000	-263616	6663457
389	Casterley Camp, Wiltshire (Catterley Banks)	27.500000	-204306	6671153
756	Willersey Camp, Gloucestershire (Willersey Hill Camp)	28.000000	-203808	6807893
427	Ebsbury Hill, Wiltshire (Grovely Earthworks)	28.000000	-212997	6642126
1276	y Breiddin, Powys (Breddin Hillfort; The Breiddin Hillfort; Breiddin Hill Camp)	28.000000	-338884	6931868
91	Titterstone Clee, Shropshire	29.600000	-289034	6872454
3795	Butser Hill, Hampshire	30.000000	-109375	6617356
464	Wadbury Camp, Somerset (Wadbury Hillfort)	30.000000	-265052	6663609
173	Abingdon, The Vineyard, Oxfordshire	33.000000	-142388	6740992
97	Walbury Camp, West Berkshire	33.000000	-162994	6684152
3459	Countisbury Castle, Devon (Shousbury; Wind Hill)	35.000000	-423647	6662041
3823	Homestall Wood, Kent	35.000000	115292	6672762
172	Dyke Hills, Oxfordshire (Dike Hills)	46.000000	-130542	6735058
760	Nottingham Hill Camp, Gloucestershire	48.600000	-225551	6791821
3774	Oldbury Camp, Kent	51.500000	29679	6671658
773	Borough Hill 1, Northamptonshire (Borough Hill)	52.000000	-126771	6847138
201	Mull of Galloway, Dumfries & Galloway	54.000000	-542988	7291829
71	Llanymynech Hill, Powys	57.000000	-344171	6944572
3594	Hengistbury Head, Dorset	80.000000	-195572	6571275
448	Ham Hill, Somerset (Hamdon Hill Camp)	84.000000	-304545	6611780
3582	Bindon Hill, Dorset	114.000000	-248650	6554366

Enclosing Area 1: Southern Hillforts Over 21 Ha Dates

Most of the hillforts over 21 Ha, in the southern data, have dating evidence and the plot is consistent to that seen in Part 3: Date Data Plotted (Excluding No Dates) and Part3: Dating by Region, where the forts have dates from the late Bronze Age through to the Early Medieval with the highest peak being in the late Iron Age.

```
In [ ]: greater_than_21ha_south_dates = \
pd.merge(greater_than_21ha_south, date_data, left_index=True, right_index=True)
```

```
In [ ]: plot_bar_chart(greater_than_21ha_south_dates[date_features], 2, \
'Dating', 'Count', 'Hillforts Over 21 Ha Dating')
```



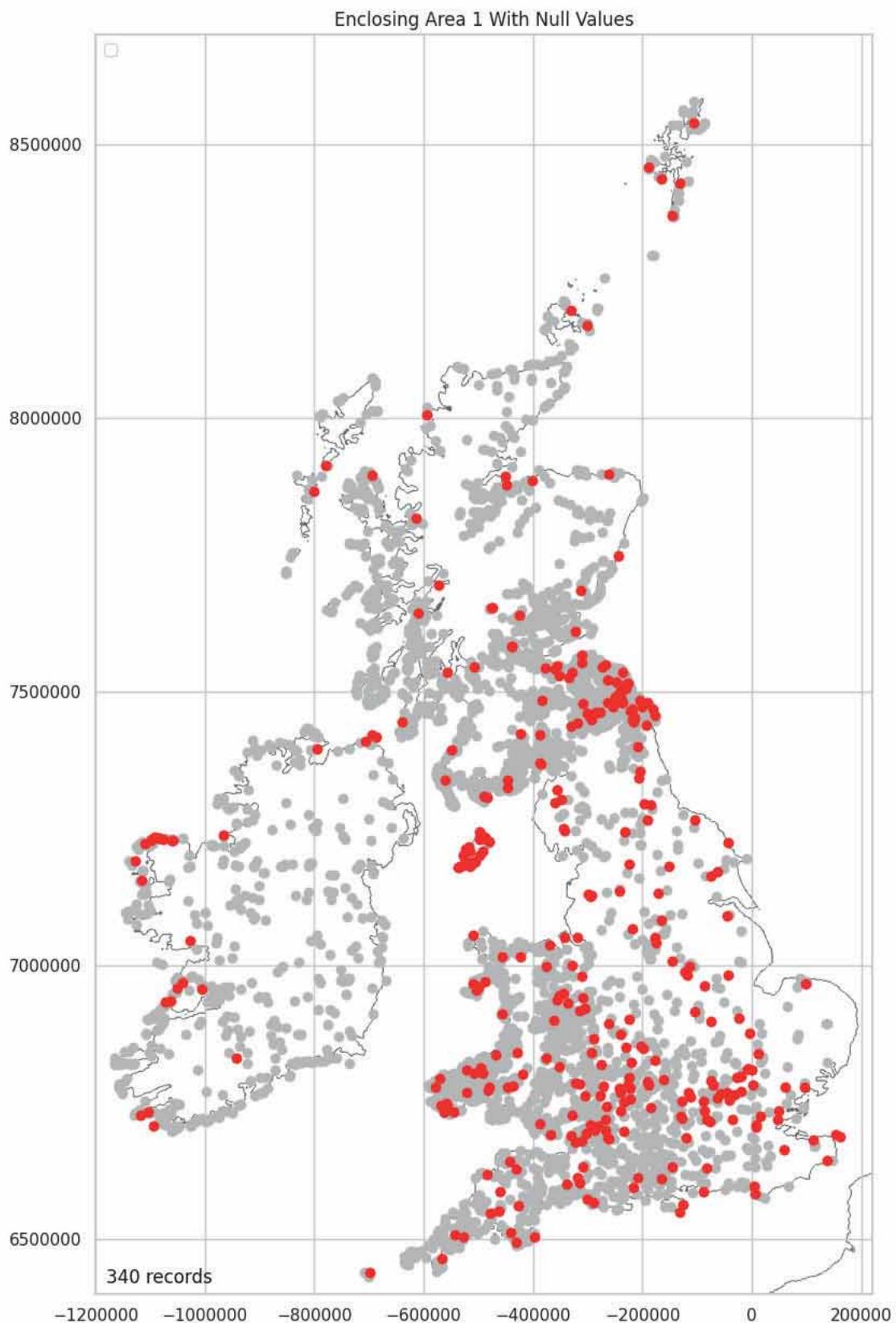
Enclosing Area 1 Null Values Mapped

There are 340 records where no 'Enclosing_Area_1' area is recorded.

```
In [ ]: encl osi ng_area_1_mi nus1 = location_encl osi ng_data.copy()
encl osi ng_area_1_mi nus1 = \
encl osi ng_area_1_mi nus1[encl osi ng_area_1_mi nus1['Encl osi ng_Area_1'] <0]
encl osi ng_area_1_mi nus1['Encl osi ng_Area_1'].describe()
```

```
Out[ ]: count    340.0
mean     -1.0
std      0.0
min     -1.0
25%     -1.0
50%     -1.0
75%     -1.0
max     -1.0
Name: Encl osi ng_Area_1, dtype: float64
```

```
In [ ]: plot_over_grey_numeric(encl osi ng_area_1_mi nus1, 'Encl osi ng_Area_1', \
'Encl osi ng_Area_1 Wi th Nul l Val ues')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

8.2%

Enclosing Area 2, 3 & 4

There are only 335 hillforts with an Enclosing_Area_2, 68 with an Enclosing_Area_3 and 11 with an Enclosing_Area_4. These additional area features have been used to capture the increased areas of hillforts when including, "outer enclosing works". ([Data Structure](#))

```
In [ ]: hillforts_data[['Enclosing_Area_2',
 'Enclosing_Area_3',
 'Enclosing_Area_4']].info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Enclosing_Area_2    335 non-null   float64
 1   Enclosing_Area_3    68 non-null   float64
 2   Enclosing_Area_4    11 non-null   float64
dtypes: float64(3)
memory usage: 97.3 KB

```

```

In [ ]: enclosing_area_2_short = \
location_enclaving_data[location_enclaving_data['Enclosing_Area_2'] >= 0]
enclosing_area_3_short = \
location_enclaving_data[location_enclaving_data['Enclosing_Area_3'] >= 0]
enclosing_area_4_short = \
location_enclaving_data[location_enclaving_data['Enclosing_Area_4'] >= 0]

```

Enclosing Area 2 Plotted

Like Enclosing_Area_1, the area of most hillforts, with an Enclosing_Area_2, are small. The spread of 95.6% of the data is quite wide, running from 0.12 to 14.46 Ha but, the interquartile range (the middle 50% of the data) only ranges from 0.4 to 2.21 Ha.

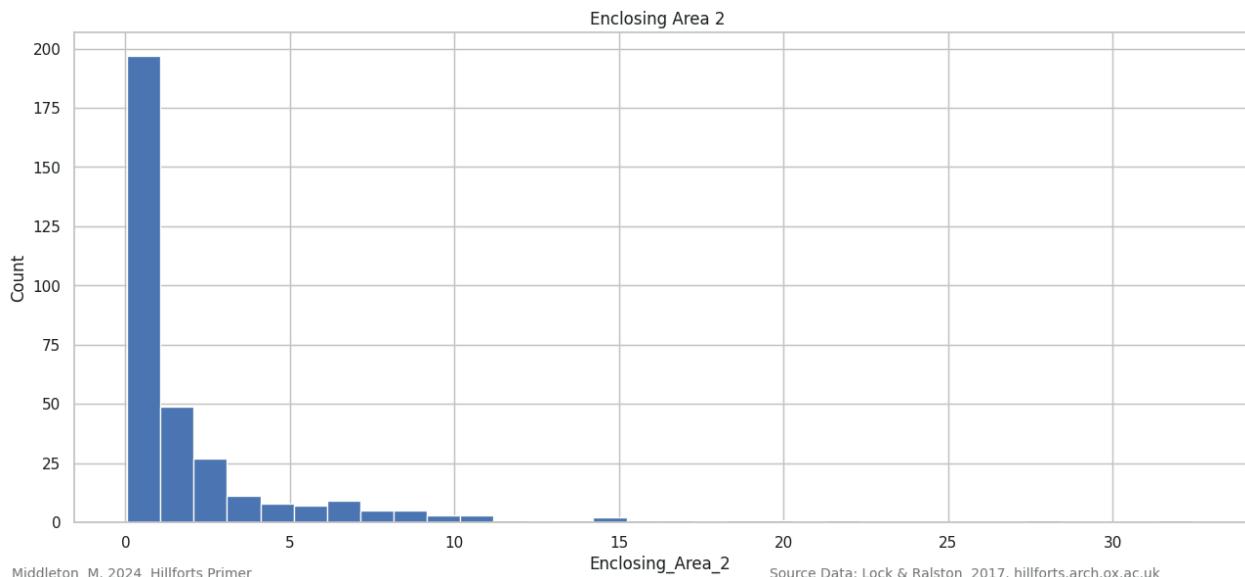
```
In [ ]: enclosing_area_2_short['Enclosing_Area_2'].describe()
```

```

Out[ ]: count    335.000000
mean     2.277761
std      3.924060
min     0.050000
25%     0.400000
50%     0.800000
75%     2.210000
max     32.430000
Name: Enclosing_Area_2, dtype: float64

```

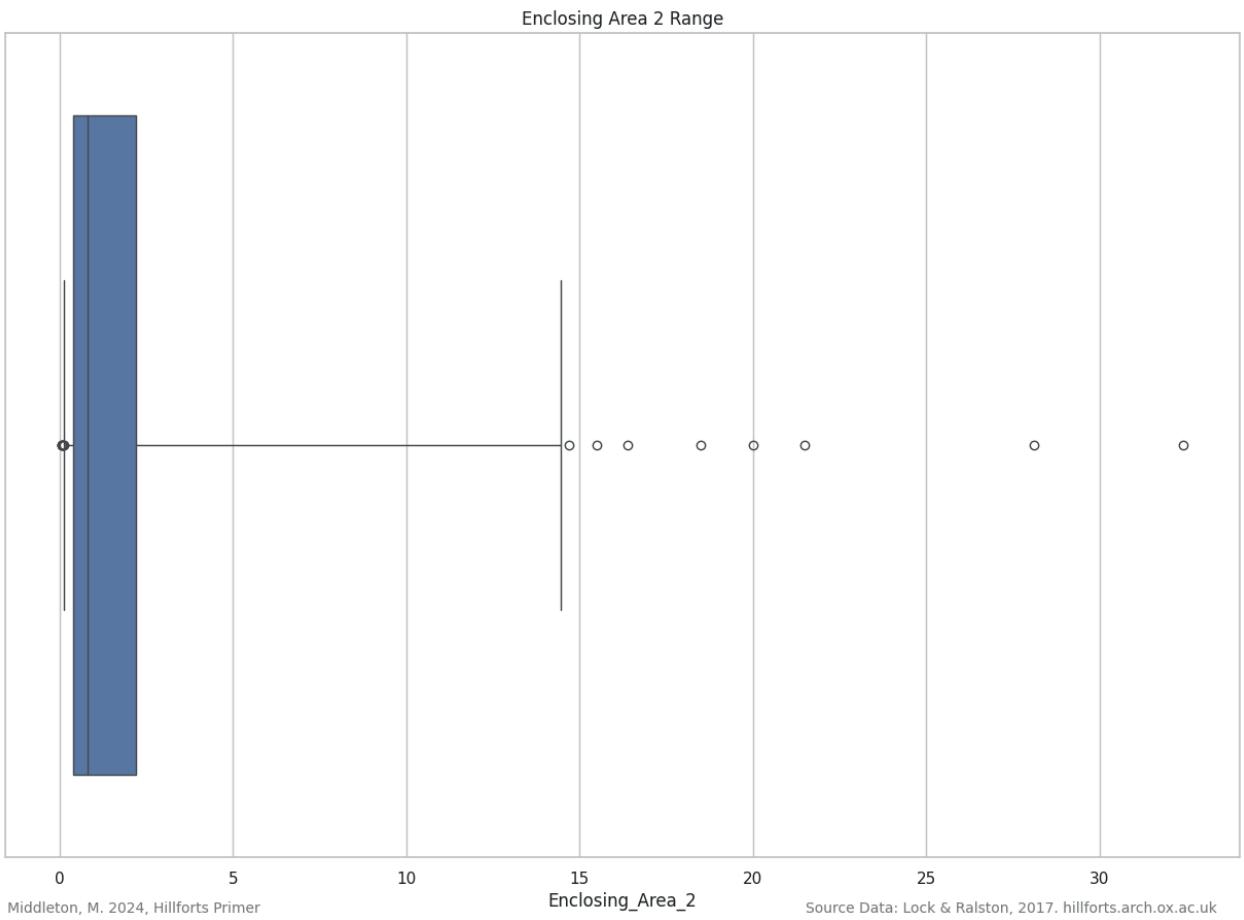
```
In [ ]: plot_bar_chart_numeric(enclosing_area_2_short, 1, 'Enclosing_Area_2', \
                             'Count', 'Enclosing_Area_2', \
                             int(enclosing_numeric_data['Enclosing_Area_2'].max()))
```



```

In [ ]: enclosing_area_2_data = \
plot_data_range(enclosing_area_2_short['Enclosing_Area_2'], \
                reset_index(drop = True), 'Enclosing_Area_2', "h")

```



```
In [ ]: enclosing_area_2_data
```

```
Out[ ]: [0.12, 0.4, 0.8, 2.21, 14.46]
```

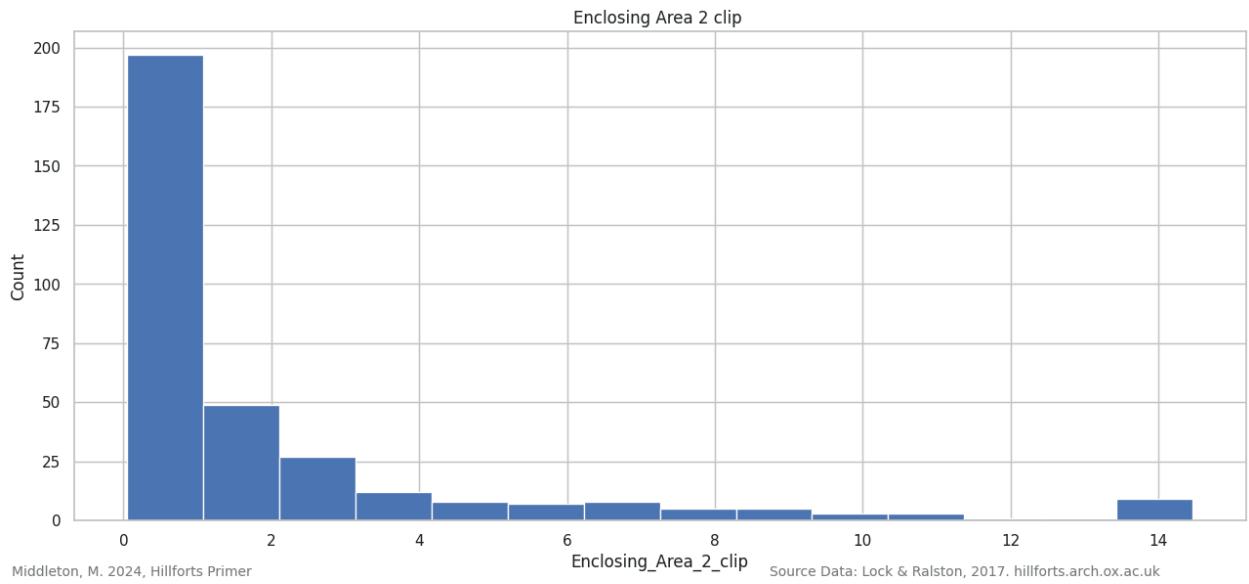
Enclosing Area 2 Clipped Plotted

To help visualise the data, outliers have been clipped. All values beyond 14.46 HA have been pooled into this value.

```
In [ ]: enclosing_area_2_data_clip = enclosing_area_2_short.copy()
enclosing_area_2_data_clip['Enclosing_Area_2_clip'] = \
enclosing_area_2_data_clip['Enclosing_Area_2'].clip(\
    enclosing_area_2_data_clip['Enclosing_Area_2'], \
    enclosing_area_2_data[-1], axis=0)
enclosing_area_2_data_clip['Enclosing_Area_2_clip'].describe()
```

```
Out[ ]: count    335.000000
mean     2.124119
std      3.128999
min     0.050000
25%     0.400000
50%     0.800000
75%     2.210000
max    14.460000
Name: Enclosing_Area_2_clip, dtype: float64
```

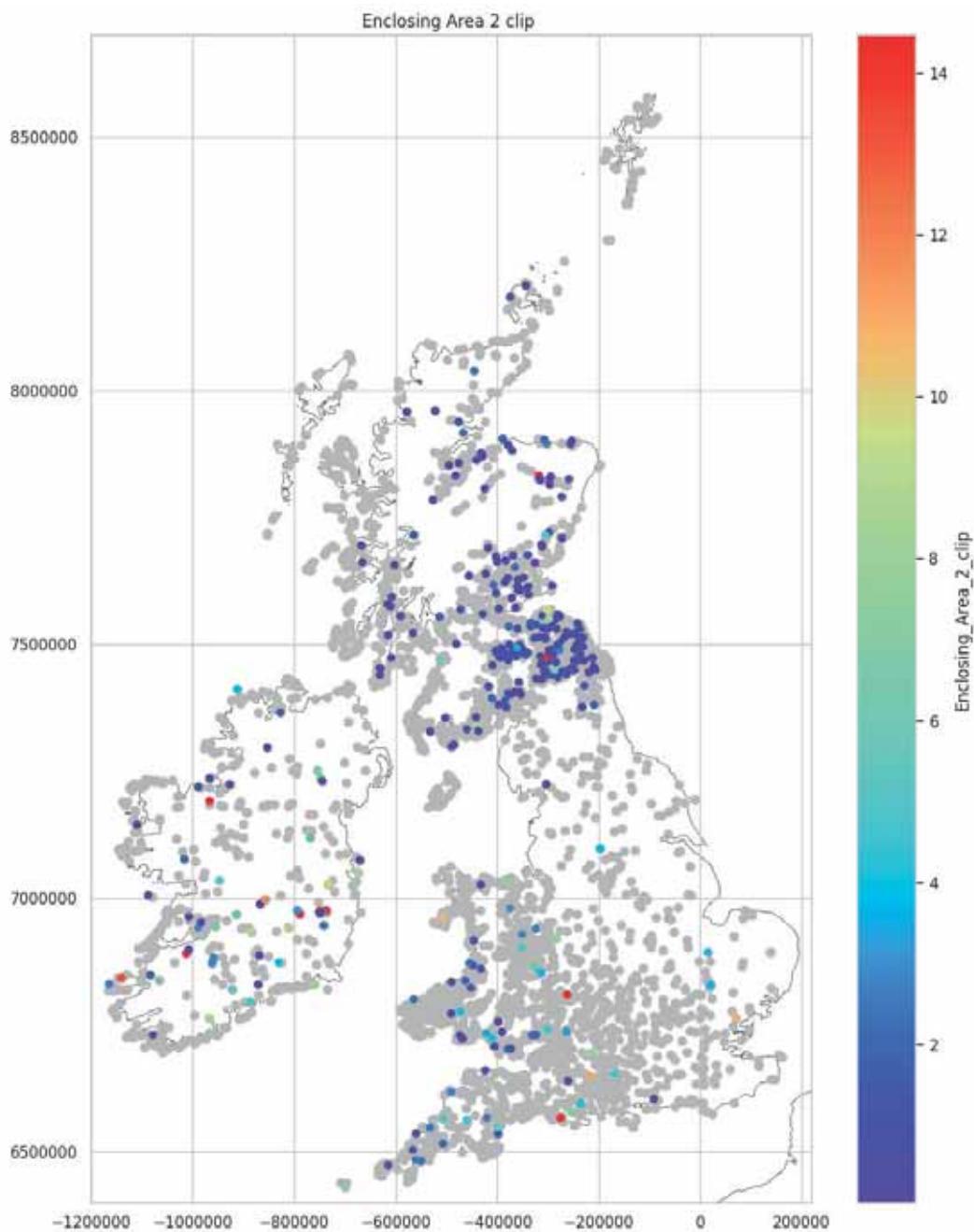
```
In [ ]: plot_bar_chart_numeric(enclosing_area_2_data_clip, 1, \
                           'Enclosing_Area_2_clip', 'Count', \
                           'Enclosing_Area_2_clip', \
                           int(enclosing_area_2_data_clip['Enclosing_Area_2_clip'].max()))
```



Enclosing Area 2 Clipped Mapped

The distribution of this data suggests there is a survey bias and that many hillforts with outer works have not had an Enclosing_Area_2 recorded. Of those that have, most are in the Northeast.

```
In [ ]: plot_type_values(enclosing_area_2_data_clip, 'Enclosing_Area_2_clip', \
                           'Enclosing_Area_2_clip')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

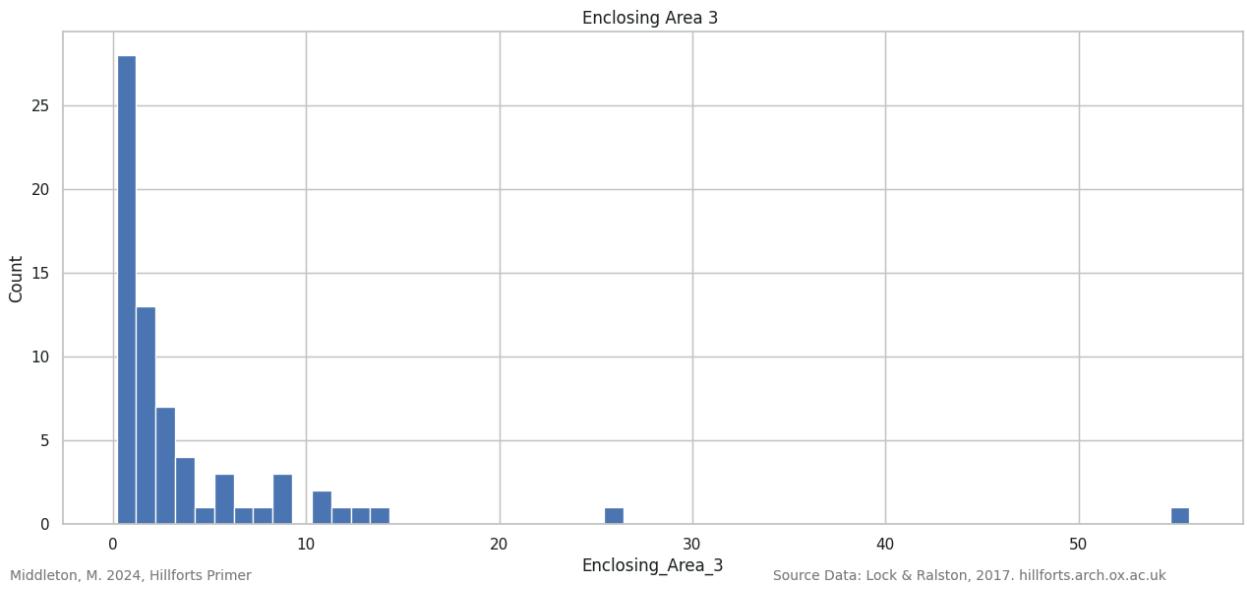
Enclosing Area 3 Plotted

Only 68 hillforts have an Enclosing_Area_3. They follow the same pattern as seen in Enclosing_Area_2, with most being quite small and most located in the North. As with Enclosing_Area_2, it is likely that this data contains a survey bias.

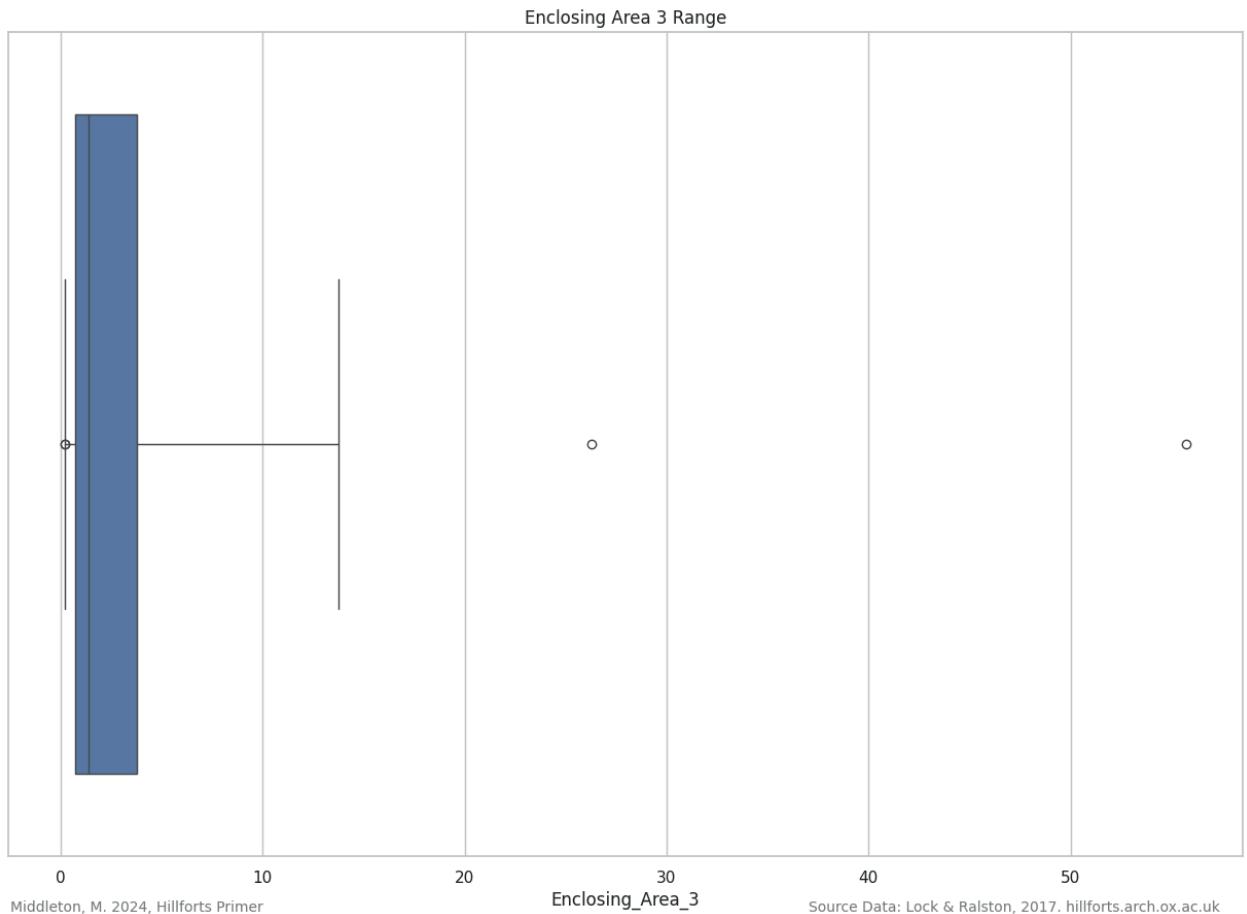
```
In [ ]: enclosing_area_3_short['Enclosing_Area_3'].describe()
```

```
Out[ ]: count    68.000000
mean     4.037647
std      7.742350
min     0.190000
25%     0.747500
50%     1.400000
75%     3.775000
max    55.740000
Name: Enclosing_Area_3, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(enclosing_area_3_short, 1, 'Enclosing_Area_3', \
                           'Count', 'Enclosing_Area_3', \
                           int(enclosing_numeric_data['Enclosing_Area_3'].max()))
```



```
In [ ]: encl osi ng_area_3_data = \
plot_data_range(encl osi ng_area_3_short['Encl osi ng_Area_3'].\
reset_index(drop = True), 'Encl osi ng_Area_3', "h")
```



```
In [ ]: encl osi ng_area_3_data
```

```
Out[ ]: [0.23, 0.7475, 1.4, 3.7750000000000004, 13.75]
```

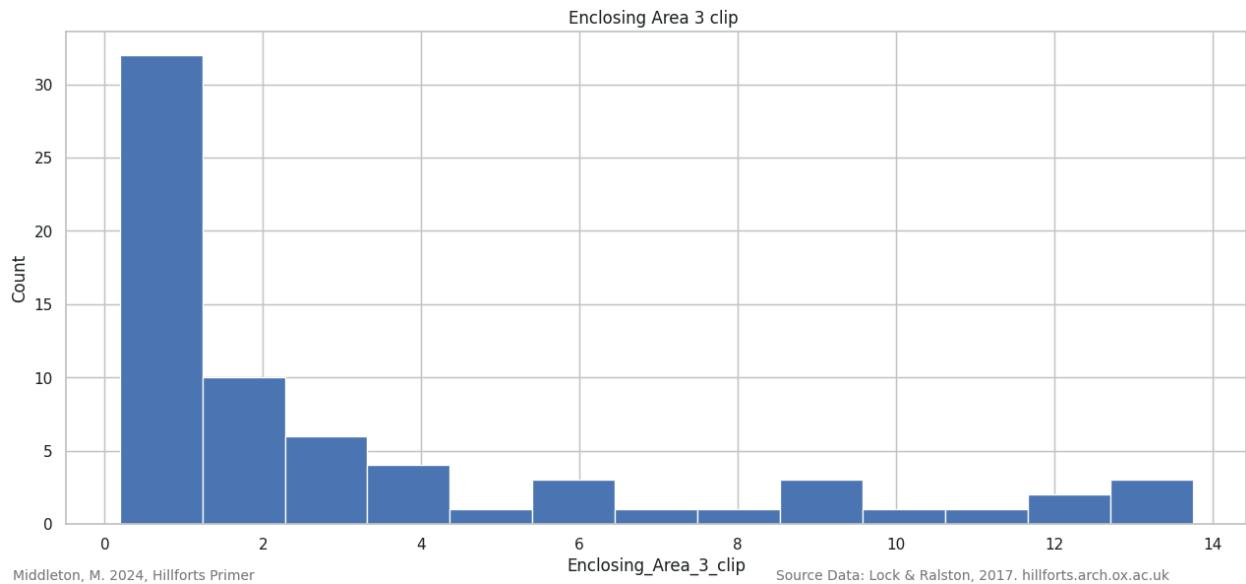
Enclosing Area 3 Clipped Plotted

To help visualise the data, outliers have been clipped. All values beyond 13.75 HA have been pooled into this value.

```
In [ ]: encl osi ng_area_3_data_cl i p = encl osi ng_area_3_short.copy()
encl osi ng_area_3_data_cl i p['Encl osi ng_Area_3_cl i p'] = \
encl osi ng_area_3_data_cl i p['Encl osi ng_Area_3'].\
clip(encl osi ng_area_3_data_cl i p['Encl osi ng_Area_3'], \
      encl osi ng_area_3_data[-1], axis=0)
encl osi ng_area_3_data_cl i p['Encl osi ng_Area_3_cl i p'].describe()
```

```
Out[ ]: count    68.000000
         mean     3.236176
         std      3.852233
         min      0.190000
         25%     0.747500
         50%     1.400000
         75%     3.775000
         max     13.750000
Name: Enclosing_Area_3_clip, dtype: float64
```

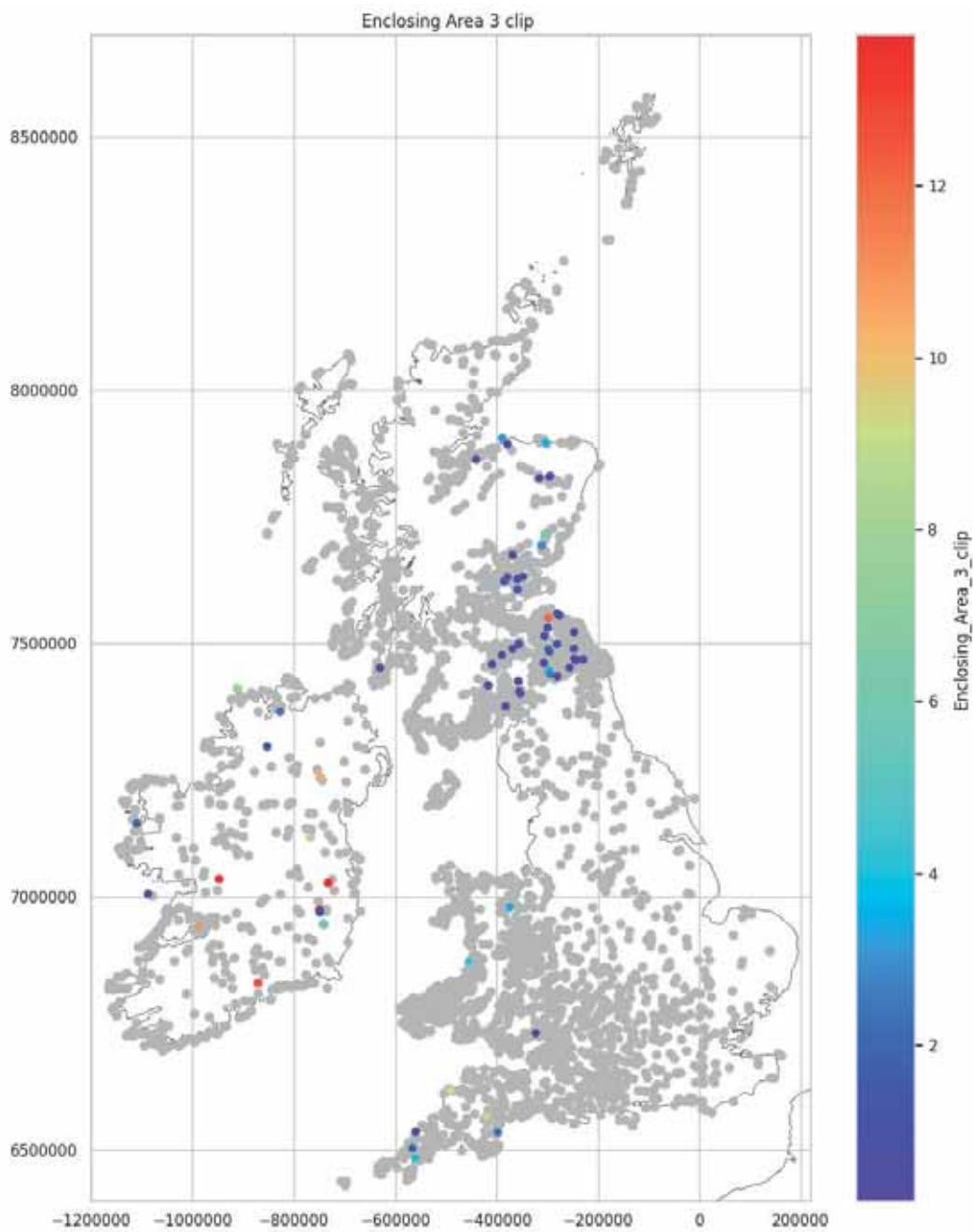
```
In [ ]: plot_bar_chart_numeric(enclosing_area_3_data_clip, 1, \
                               'Enclosing_Area_3_clip', 'Count', \
                               'Enclosing_Area_3_clip', \
                               int(enclosing_area_3_data_clip['Enclosing_Area_3_clip'].max()))
```



Enclosing Area 3 Clipped Mapped

The forts in this class are mostly located in the Northeast. This, and the low number of records in this class, suggests that this data has a survey bias toward this area.

```
In [ ]: plot_type_values(enclosing_area_3_data_clip, 'Enclosing_Area_3_clip', \
                           'Enclosing_Area_3_clip')
```



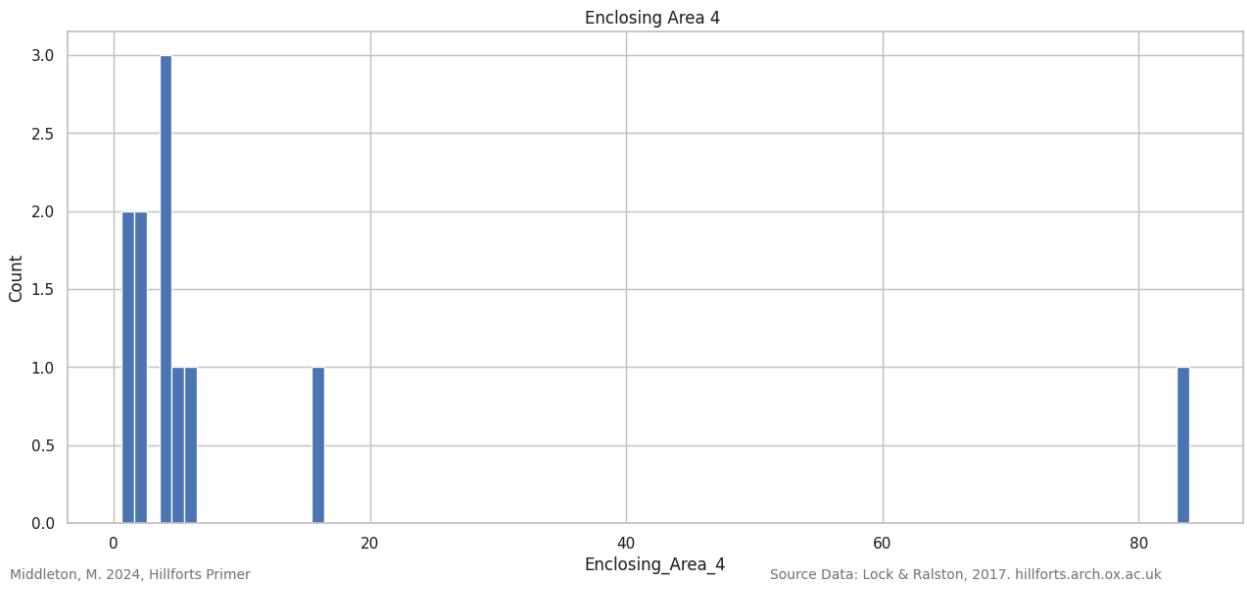
Enclosing Area 4 Plotted

Only 11 hillforts have a fourth outer work recorded.

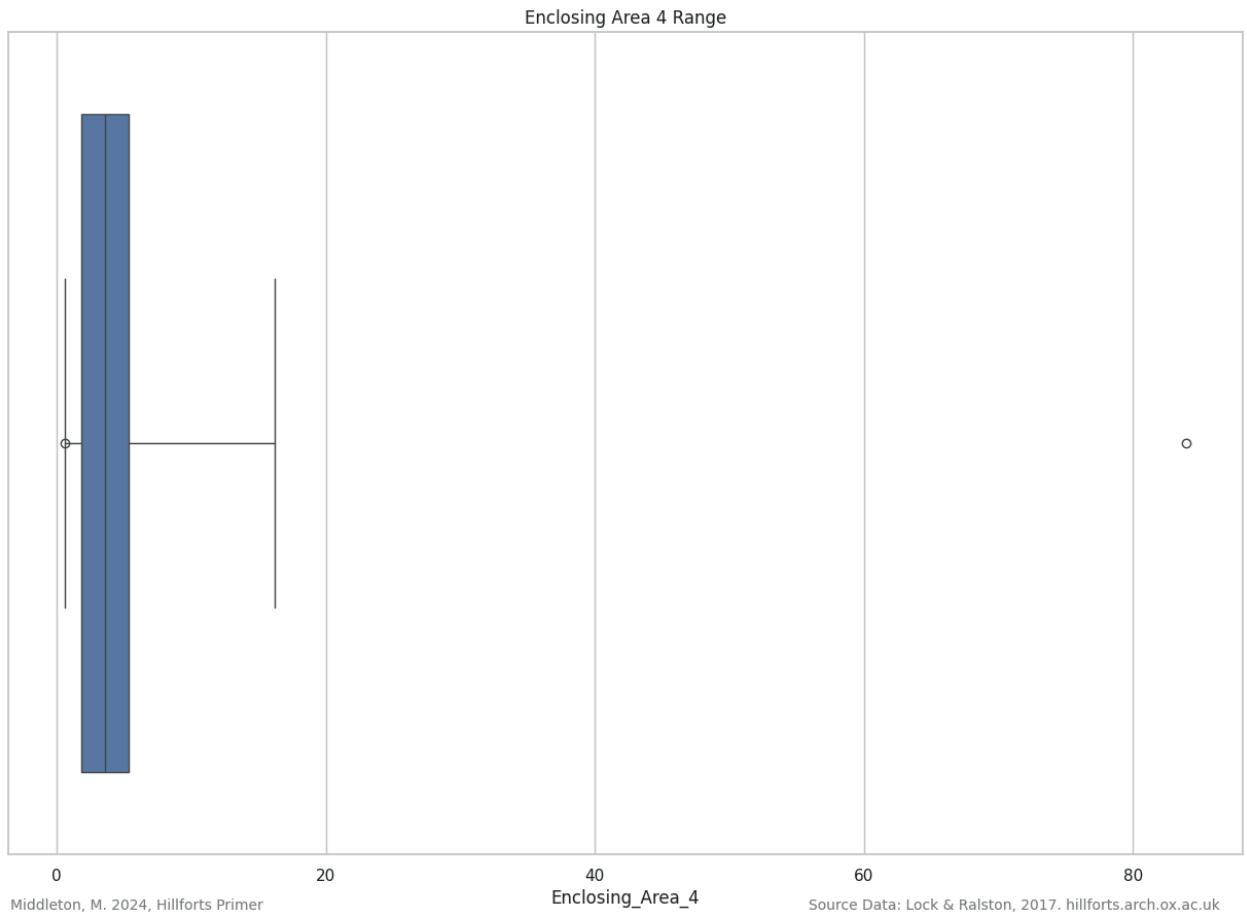
```
In [ ]: enclosing_area_4_short['Enclosing_Area_4'].describe()
```

```
Out[ ]:
count    11.000000
mean     11.568182
std      24.404416
min      0.560000
25%     1.830000
50%     3.600000
75%     5.350000
max     84.000000
Name: Enclosing_Area_4, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(enclosing_area_4_short, 1, \
                           'Enclosing_Area_4', 'Count', 'Enclosing_Area_4', \
                           int(enclosing_numeric_data['Enclosing_Area_4'].max()))
```



```
In [ ]: enclosing_area_4_data = \
plot_data_range(enclosing_area_4_short['Enclosing_Area_4'].\
reset_index(drop = True), 'Enclosing_Area_4', "h")
```



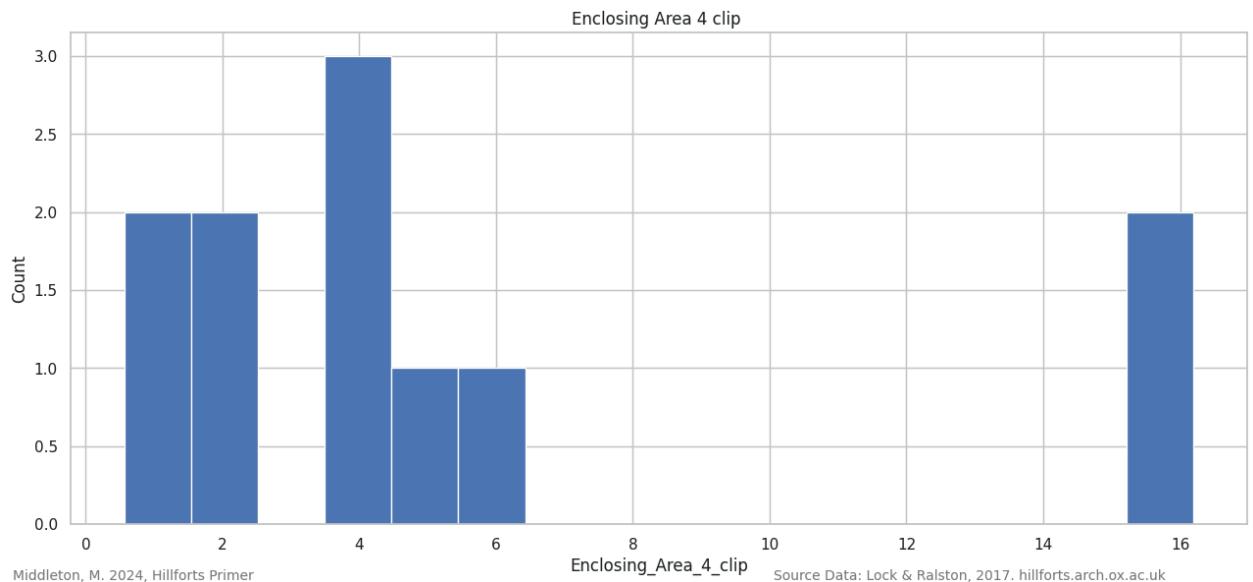
Enclosing Area 4 Clipped Plotted

This group contains forts in a range up to 16.2 Ha and a single huge hillfort at 84 Ha. To aid in visualising this data this outlier has been pooled to 16.2 Ha.

```
In [ ]: enclosing_area_4_data_clip = enclosing_area_4_short.copy()
enclosing_area_4_data_clip['Enclosing_Area_4_clip'] = \
enclosing_area_4_data_clip['Enclosing_Area_4'].\
clip(enclosing_area_4_data_clip['Enclosing_Area_4'], enclosing_area_4_data[-1], \
axis=0)
enclosing_area_4_data_clip['Enclosing_Area_4_clip'].describe()
```

```
Out[ ]: count    11.000000
         mean     5.404545
         std      5.594157
         min     0.560000
         25%    1.830000
         50%    3.600000
         75%    5.350000
         max    16.200000
Name: Enclosing_Area_4_clip, dtype: float64
```

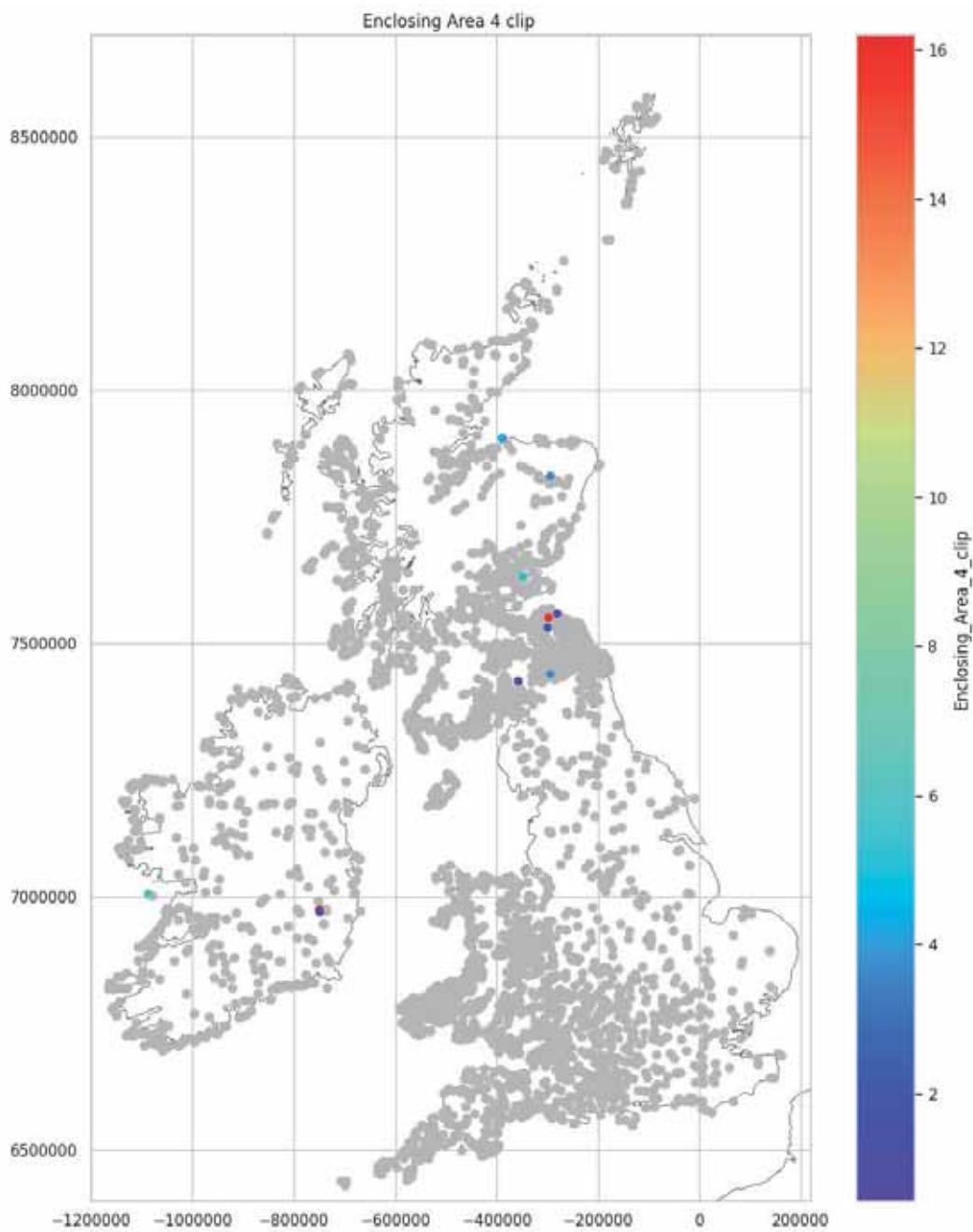
```
In [ ]: plot_bar_chart_numeric(enclosing_area_4_data_clip, 1, \
                               'Enclosing_Area_4_clip', 'Count', 'Enclosing_Area_4_clip', \
                               int(enclosing_area_4_data_clip['Enclosing_Area_4_clip'].max()))
```



Enclosing Area 4 Clipped Mapped

The forts in this class are mostly located in the Northeast. This, and the low number of records, suggest this data has a survey bias.

```
In [ ]: plot_type_values(enclosing_area_4_data_clip, 'Enclosing_Area_4_clip', \
                           'Enclosing_Area_4_clip')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Enclosed Area : Difference between Enclosing Enclosed Area and Enclosing Area 1 Plotted

Note: Enclosing_Enclosed_Area should not be confused with [Enclosing_Area](#).

There are 3807 hillfort records that have both 'Enclosing_Enclosed_Area' and 'Enclosing_Area_1' recorded. 313 of these hillforts have an 'Enclosing_Enclosed_Area' that is larger than 'Enclosing_Area_1'. Of these, the majority are between, 0.27 and 1.96 Ha larger. The largest difference is 79.33 Ha.

```
In [ ]: #Hillforts with an 'Enclosing_Enclosed_Area'
encl osi ng_encl osed_area = location_encl osi ng_data.copy()
encl osi ng_encl osed_area['encl osi ng_encl osed_area'] = \
encl osi ng_encl osed_area[encl osi ng_encl osed_area['Encl osi ng_Enclosed_Area'] >= 0]
encl osi ng_encl osed_area['Encl osi ng_Enclosed_Area'].describe()
```

```
Out[ ]: count    3807.000000
mean      1.823261
std       5.652301
min       0.010000
25%      0.200000
50%      0.410000
75%      1.400000
max     130.000000
Name: Encl osi ng_Enclosed_Area, dtype: float64
```

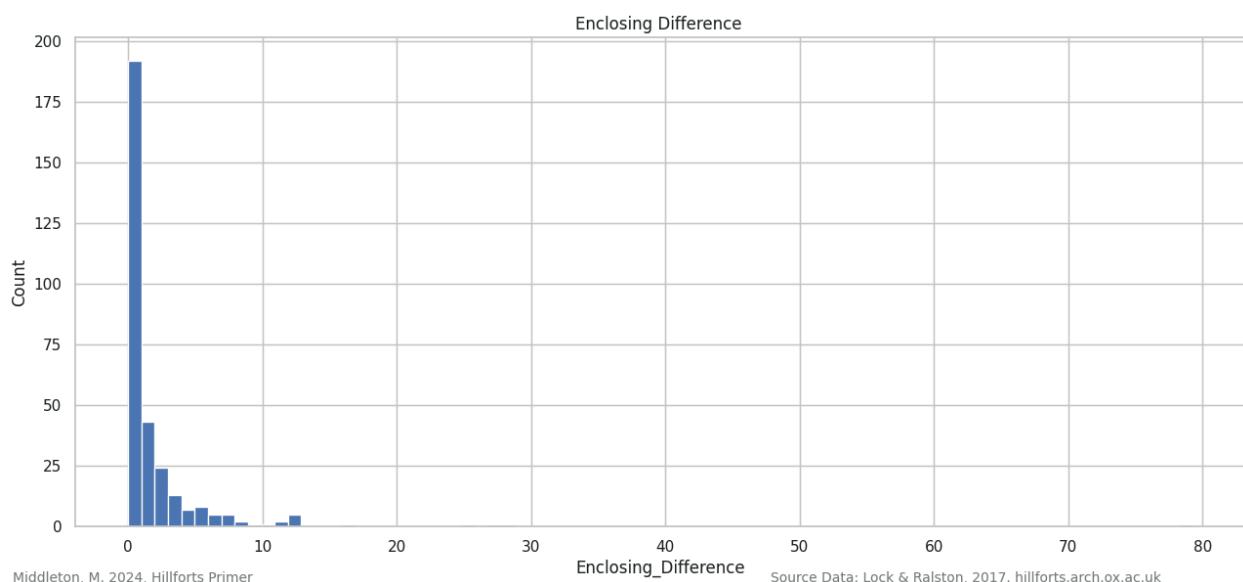
```
In [ ]: #The difference in area between 'Enclosing Enclosed Area' and 'Enclosing Area 1'
encl osing_encl osed_area['Encl osing_Difference'] = \
encl osing_encl osed_area['Encl osing_Enclosed_Area'] - \
encl osing_encl osed_area['Encl osing_Area_1']
encl osing_encl osed_area['Encl osing_Difference'].describe()
encl osing_difference = \
encl osing_encl osed_area[encl osing_encl osed_area['Encl osing_Difference'] > 0]
encl osing_difference['Encl osing_Difference'].describe()
```

```
Out[ ]:
count    313.000000
mean      2.170128
std       5.629909
min      0.010000
25%     0.270000
50%     0.600000
75%     1.970000
max     79.330000
Name: Encl osing_Difference, dtype: float64
```

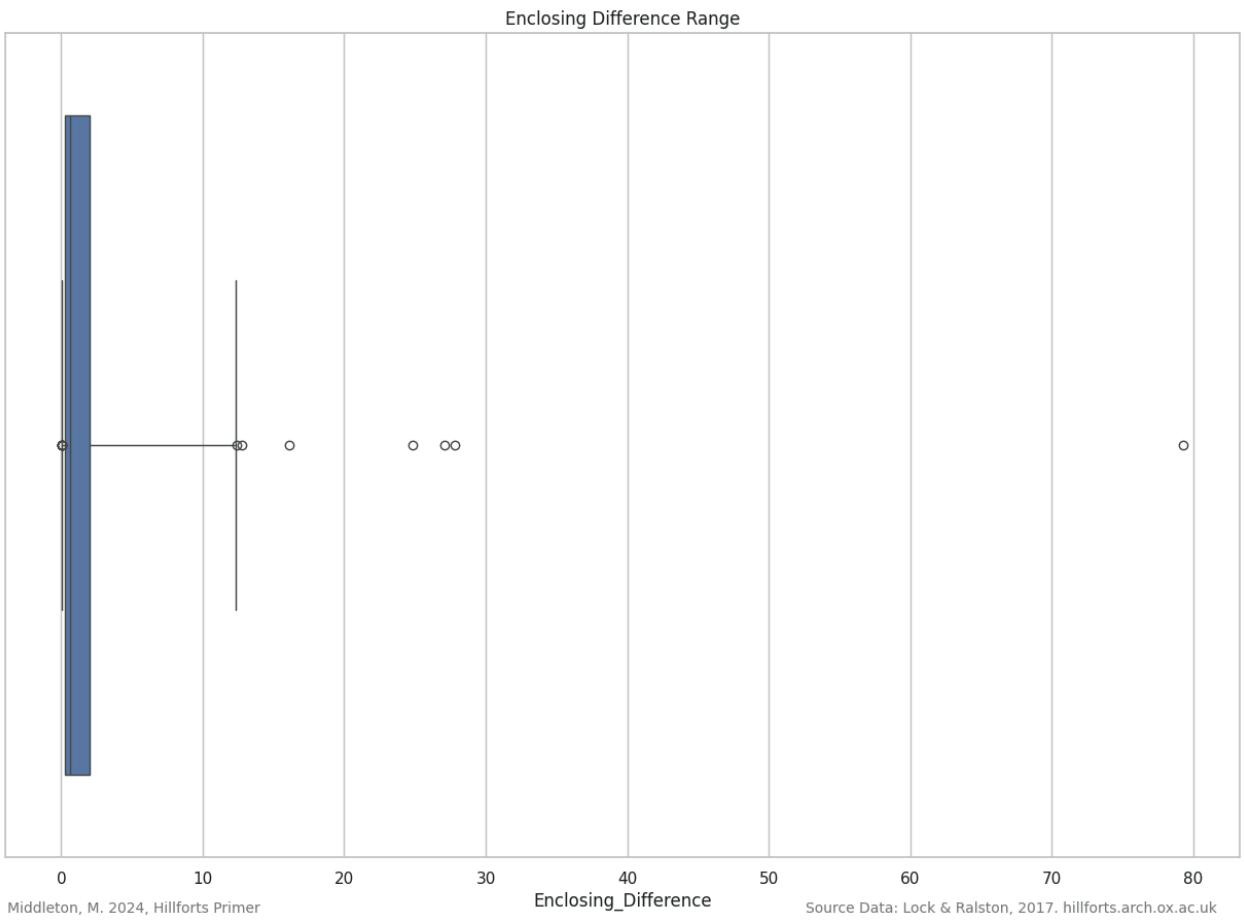
```
In [ ]: #Number of 'Enclosing Enclosed Area' records where there is no 'Enclosing Area 1'
ea_but_no_ea1 = \
encl osing_encl osed_area[encl osing_encl osed_area['Encl osing_Area_1'] == -1]
len(ea_but_no_ea1['Encl osing_Difference'])
```

```
Out[ ]: 0
```

```
In [ ]: plot_bar_chart_numeric(encl osing_difference, 1, 'Encl osing_Difference', \
'Count', 'Encl osing_Difference', 80)
```



```
In [ ]: encl osing_difference_data = \
plot_data_range(encl osing_difference['Encl osing_Difference'], \
reset_index(drop = True), 'Encl osing_Difference', "h")
```



```
In [ ]: encl osi ng_di fference_data
```

```
Out[ ]: [0.0499999999999999, 0.27, 0.6, 1.9699999999999989, 12.3]
```

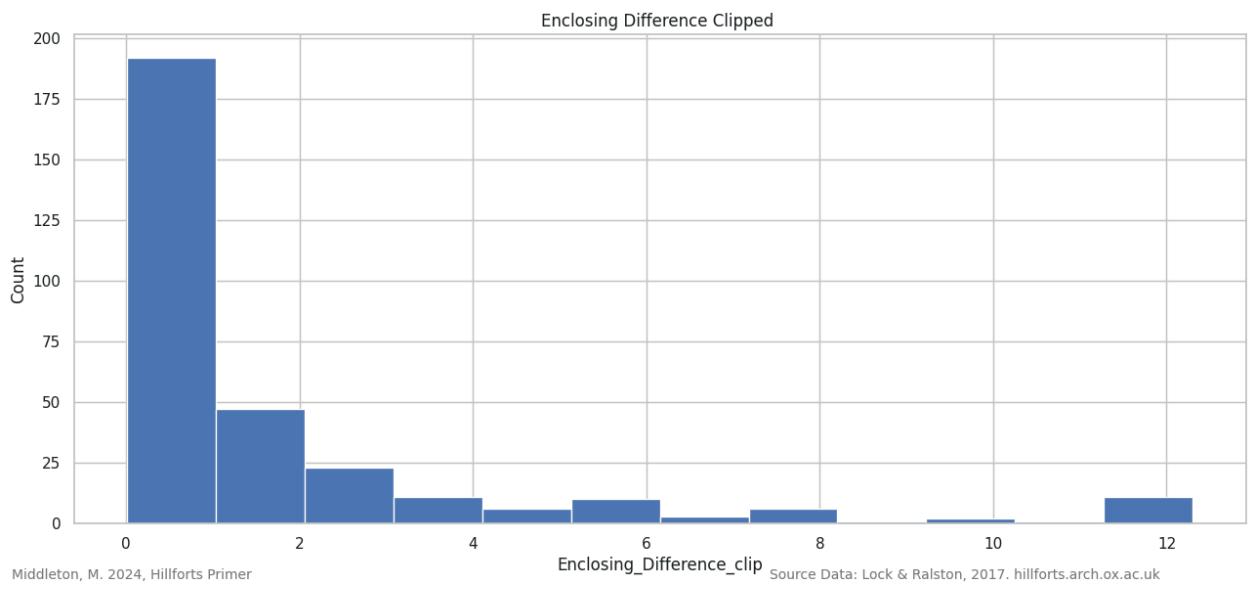
Enclosing Enclosed Area: Difference between Enclosing Enclosed Area and Enclosing Area 1 Clipped Plotted

To facilitate plotting the data has been clipped to 16 Ha. All values beyond this have been pooled into this value.

```
In [ ]: encl osi ng_di fference_data_cl ip = encl osi ng_di fference.copy()
encl osi ng_di fference_data_cl ip['Encl osi ng_Di fference_cl ip'] = \
encl osi ng_di fference_data_cl ip['Encl osi ng_Di fference'], \
clip(encl osi ng_di fference_data_cl ip['Encl osi ng_Di fference'], \
     encl osi ng_di fference_data[-1], axis=0)
encl osi ng_di fference_data_cl ip['Encl osi ng_Di fference_cl ip'].descri be()
```

```
Out[ ]: count    313.000000
mean      1.805176
std       2.779261
min       0.010000
25%      0.270000
50%      0.600000
75%      1.970000
max      12.300000
Name: Encl osi ng_Di fference_cl ip, dtype: float64
```

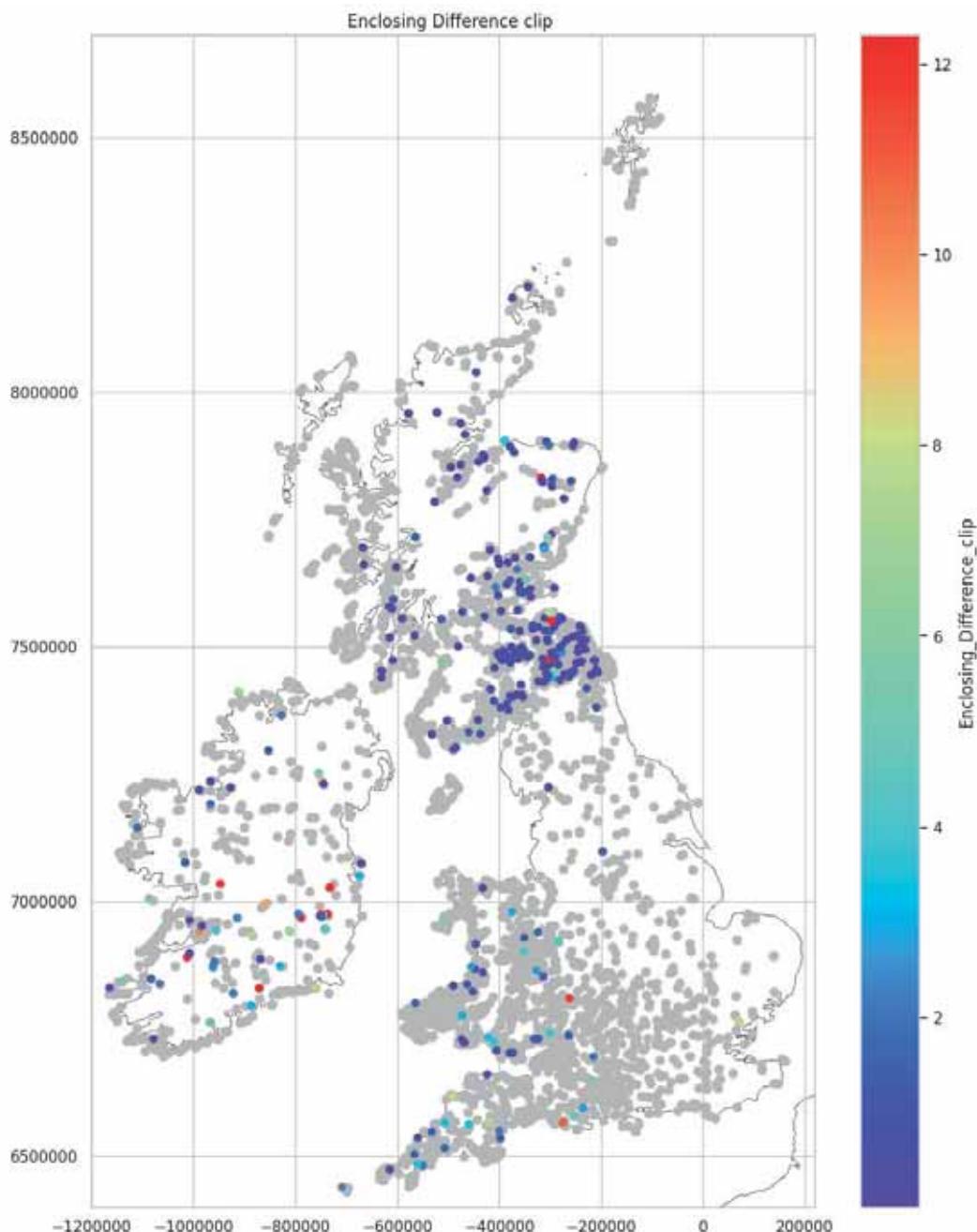
```
In [ ]: plot_bar_chart_numeric(encl osi ng_di fference_data_cl ip, 1, \
                           'Encl osi ng_Di fference_cl ip', 'Count', \
                           'Encl osi ng_Di fference Clipped', \
                           int(encl osi ng_di fference_data_cl ip['Encl osi ng_Di fference_cl ip'].max()))
```



Enclosing Enclosed Area: Difference between Enclosing Enclosed Area and Enclosing Area 1 Clipped Mapped

Most of the hillforts with an Enclosing_Enclosed_Area are located in the Northeast. This suggests there is a recording bias.

```
In [ ]: plot_type_values(enclosing_difference_data_clip, \
    'Enclosing_Difference_clip', 'Enclosing_Difference_Clip')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Area Plotted

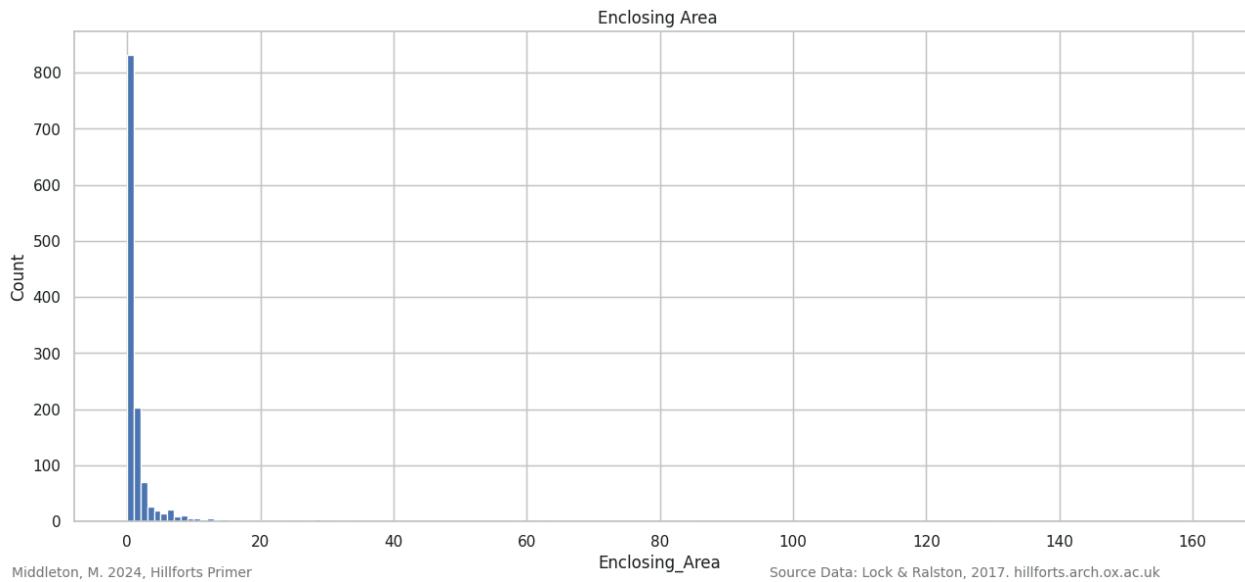
Note: Not to be confused with [Enclosing_Enclosed_Area](#).

There are only 1259 hillforts with a recorded Enclosing_Area, the area "within the inner rampart/bank/wall where measurable", compared to 3807 hillforts that have an Enclosing_Area_1 recorded ([Data Structure](#)). The areas range from 0.02 Ha to 160 Ha. 95.6% range between 0.06 Ha and 16 Ha. The interquartile range is 0.34, to 1.435 Ha.

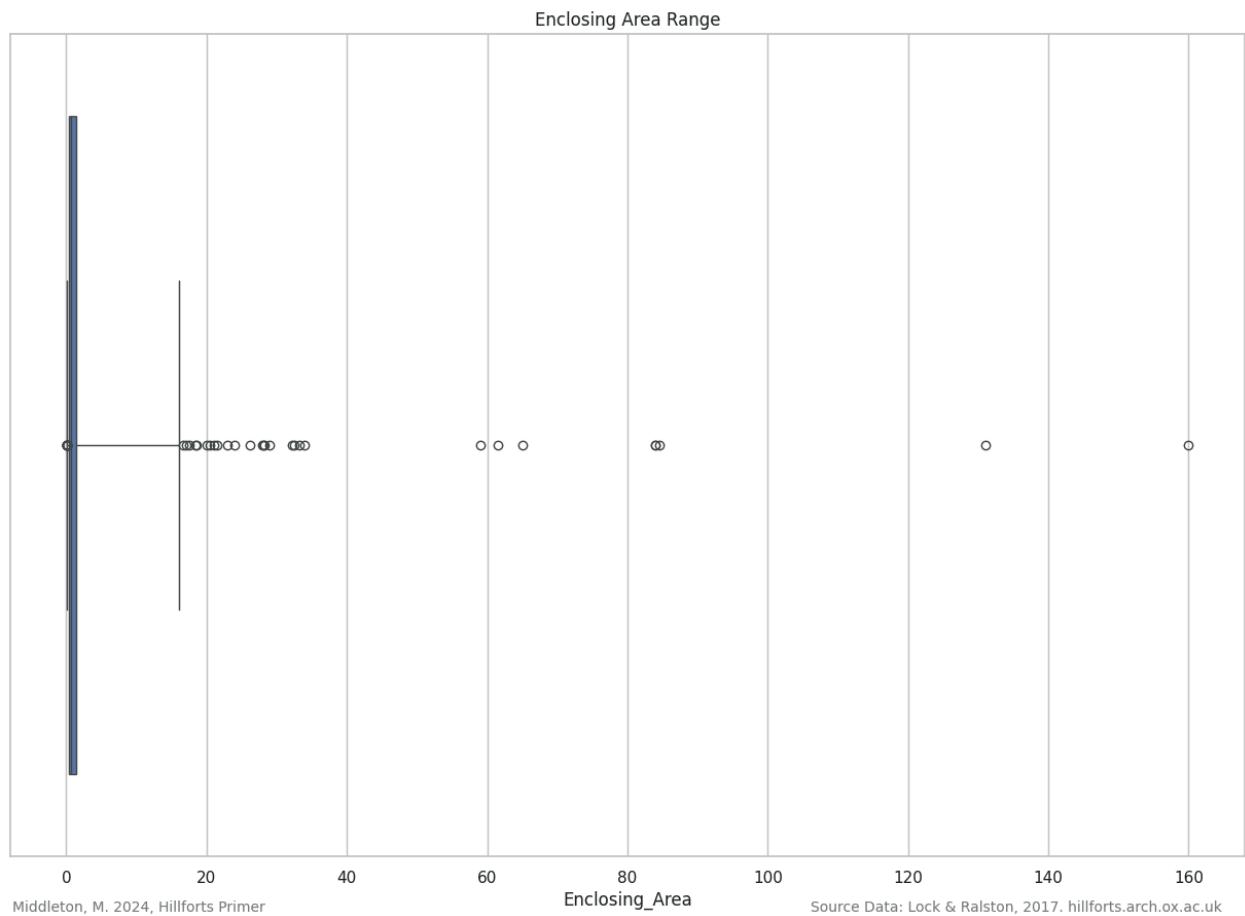
```
In [ ]: enclosing_area = location_enclosing_data.copy()
enclosing_area = enclosing_area[enclosing_area['Enclosing_Area'] > 0]
enclosing_area['Enclosing_Area'].describe()
```

```
Out[ ]: count    1259.000000
mean      2.325655
std       8.436951
min      0.020000
25%      0.340000
50%      0.700000
75%      1.450000
max     160.000000
Name: Enclosing_Area, dtype: float64
```

```
In [ ]: plot_bar_chart_numeric(enclosing_area, 1, 'Enclosing_Area', 'Count', \
                           'Enclosing_Area', \
                           int(enclosing_numeric_data['Enclosing_Area'].max()))
```



```
In [ ]: enclosing_area_data = \
plot_data_range(enclosing_area['Enclosing_Area'].reset_index(drop = True), \
                'Enclosing_Area', "h")
```



```
In [ ]: enclosing_area_data
```

```
Out[ ]: [0.06, 0.34, 0.7, 1.45, 16.0]
```

Enclosing Area Clipped Plotted

To facilitate plotting the data is clipped to 16 Ha. All values beyond will be pooled into this value.

```
In [ ]: enclosing_area_clip = enclosing_area.copy()
enclosing_area_clip['Enclosing_Area_Clip'] = \
enclosing_area_clip['Enclosing_Area'].clip(enclosing_area_clip['Enclosing_Area'], \
```

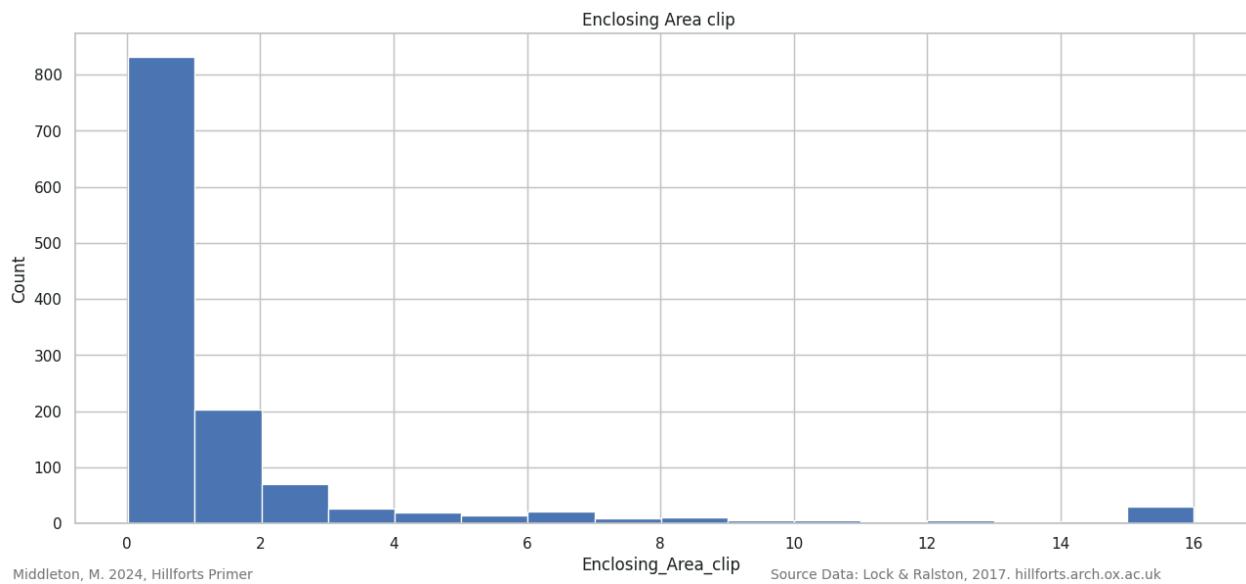
```
enclosing_area_clip['Enclosing_Area_clip'].describe()
```

```
Out[ ]:
```

	count	mean	std	min	25%	50%	75%	max
count	1259.000000	1.713487	3.047517	0.020000	0.340000	0.700000	1.450000	16.000000

Name: Enclosing_Area_clip, dtype: float64

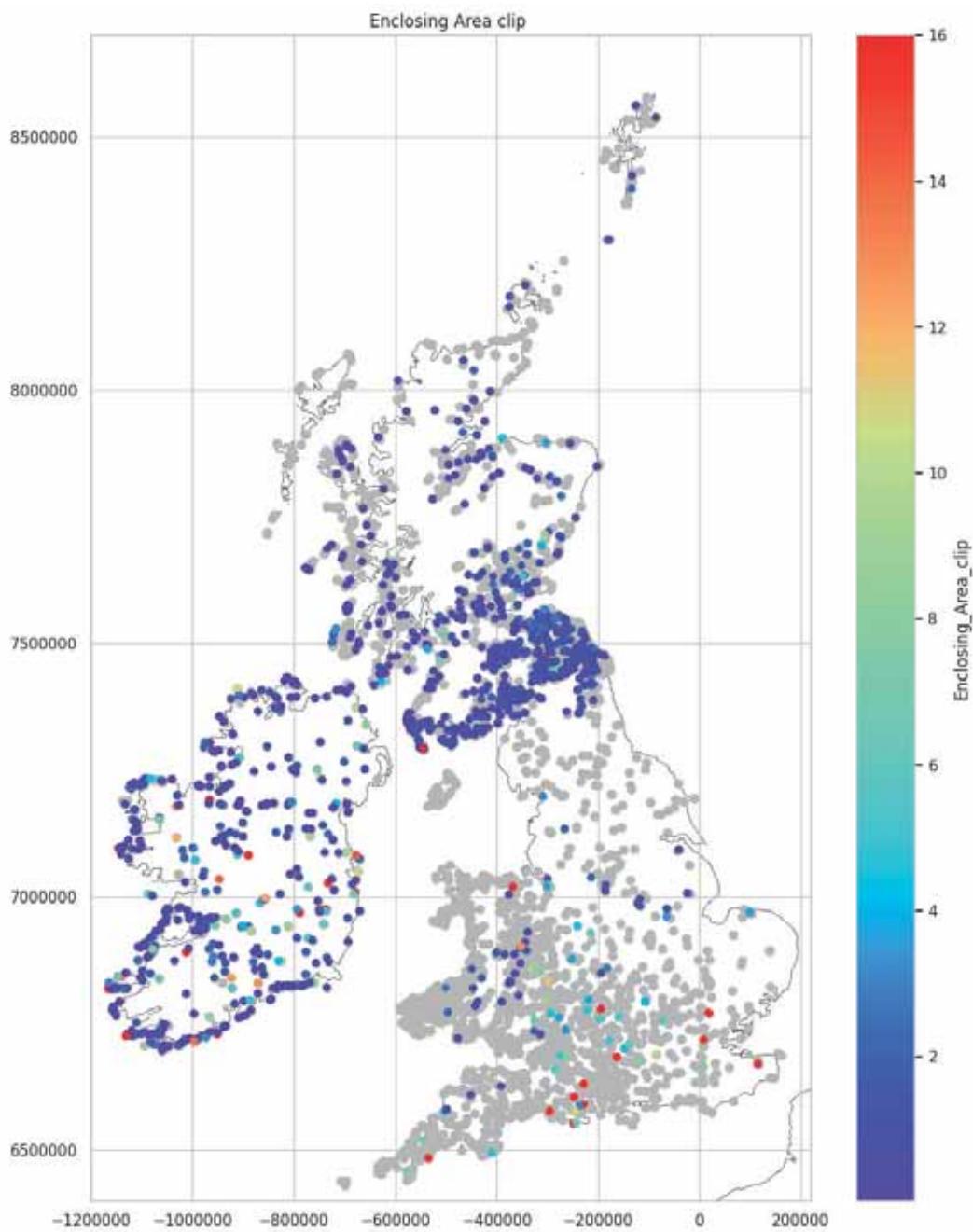
```
In [ ]: plot_bar_chart_numeric(enclosing_area_clip, 1, 'Enclosing_Area_clip', \
                                'Count', 'Enclosing_Area_clip', \
                                int(enclosing_area_clip['Enclosing_Area_clip'].max()))
```



Enclosing Area Clipped Mapped

There is a recording bias toward Ireland and Scotland. What data there is in England and Wales seems to follow the pattern observed in Enclosed_Area_1, where larger hillforts are located in the South. Similarly, in the Irish and Scottish data the larger forts are located in south central Ireland.

```
In [ ]: plot_type_values(enclosing_area_clip, 'Enclosing_Area_clip', \
                           'Enclosing_Area_Clipped')
```



Ramparts Plotted

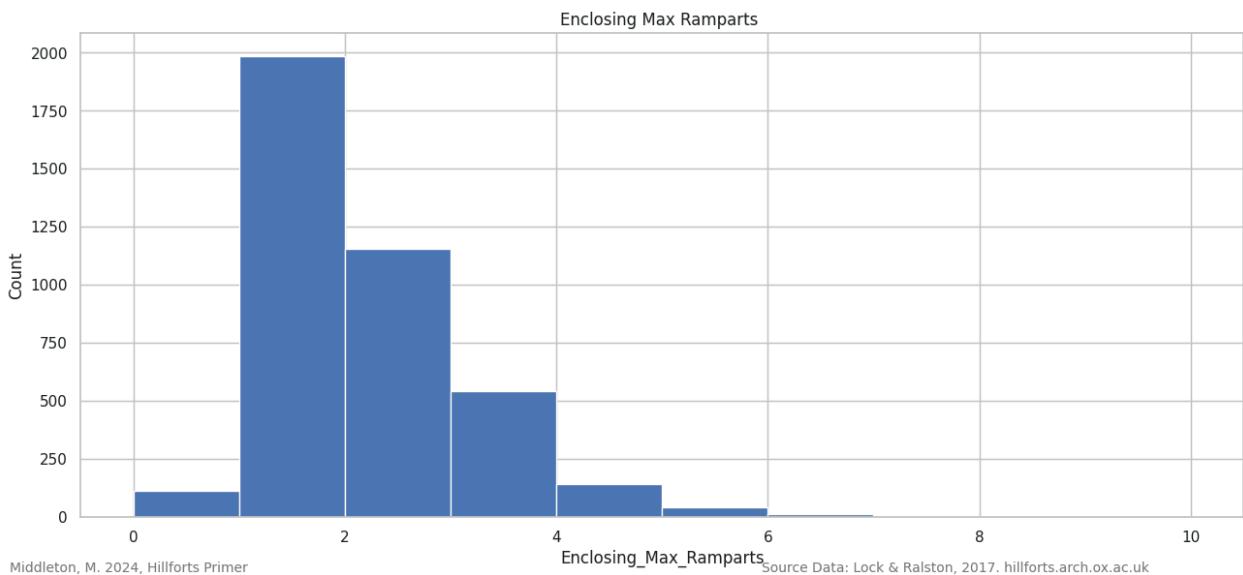
Most hillforts (75.77%) have one or two ramparts. 95.6% have four or less. 113 hillforts have no ramparts while, with 10 ramparts, West-Town, Waterford, in Ireland, is the fort with the most.

```
In [ ]: ramparts_location_enc_data = \
location_enclosing_data[location_enclosing_data['Enclousing_Max_Ramparts'] >= 0]
ramparts_location_enc_data['Enclousing_Max_Ramparts'].value_counts().sort_index()
```

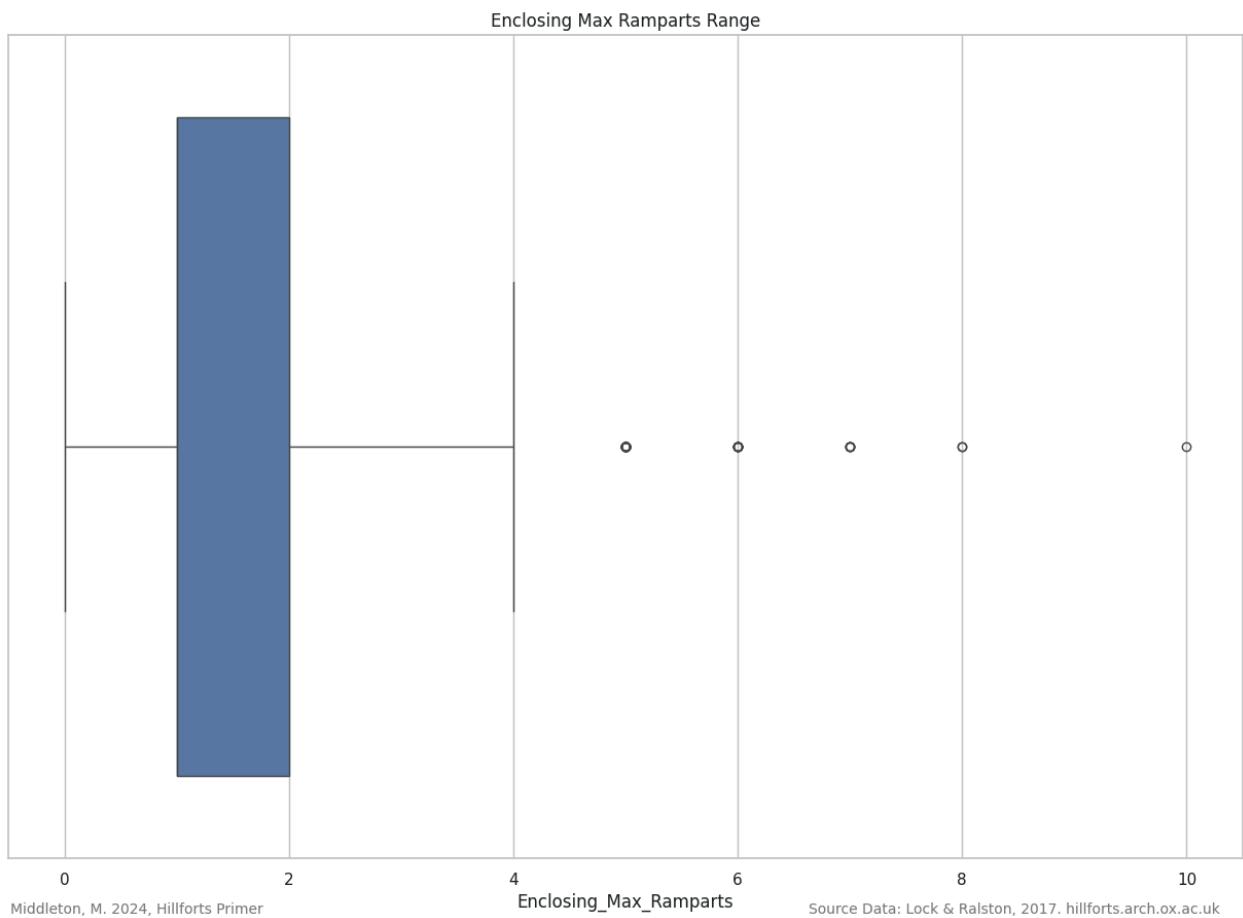
```
Out[ ]: 0.0      113
1.0     1985
2.0     1156
3.0      542
4.0      141
5.0      43
6.0      11
7.0       5
8.0       2
10.0      1
Name: Enclousing_Max_Ramparts, dtype: int64
```

```
In [ ]: plot_bar_chart_numeric(ramparts_location_enc_data, 1, \
                           'Enclousing_Max_Ramparts', 'Count', \
```

```
'Encl osi ng_Max_Ramparts', \
int(ramparts_location_enc_data['Encl osi ng_Max_Ramparts'].max()))
```



```
In [ ]: ramparts_data = \
plot_data_range(ramparts_location_enc_data['Encl osi ng_Max_Ramparts'].\
reset_index(drop = True), 'Encl osi ng_Max_Ramparts', "h")
```



```
In [ ]: ramparts_data
```

```
Out[ ]: [0.0, 1.0, 1.0, 2.0, 4.0]
```

Ramparts Clipped Mapped

To aid visualising the ramparts data, outliers are clipped to four ramparts. Any fort with more than four ramparts is pooled into this value. The clipped plot is still difficult to interpret so individual values will be reviewed below.

```
In [ ]: ramparts_clip = ramparts_location_enc_data.copy()
ramparts_clip['Encl osi ng_Max_Ramparts_clip'] = \
ramparts_clip['Encl osi ng_Max_Ramparts'].\
```

```

clip(ramparts_clip['Enclosing_Max_Ramparts'], ramparts_data[-1], axis=0)
ramparts_clip['Enclosing_Max_Ramparts'].value_counts().sort_index()

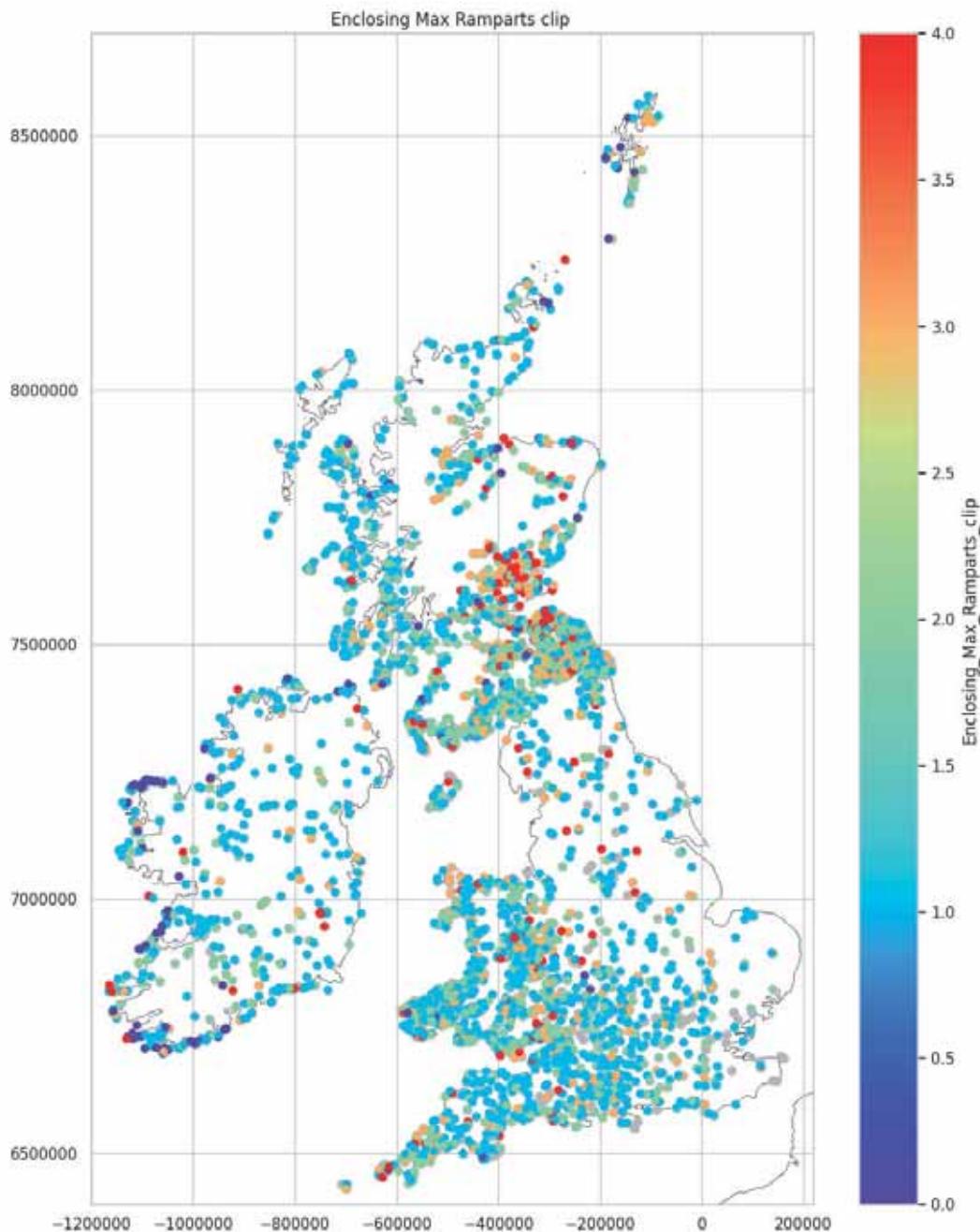
Out[ ]:
0.0    113
1.0   1985
2.0   1156
3.0    542
4.0    203
Name: Enclosing_Max_Ramparts_clip, dtype: int64

```

```

In [ ]: plot_type_values(ramparts_clip, 'Enclosing_Max_Ramparts_clip', \
                           'Enclosing_Max_Ramparts_clip')

```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ramparts Mapped (Not Recorded)

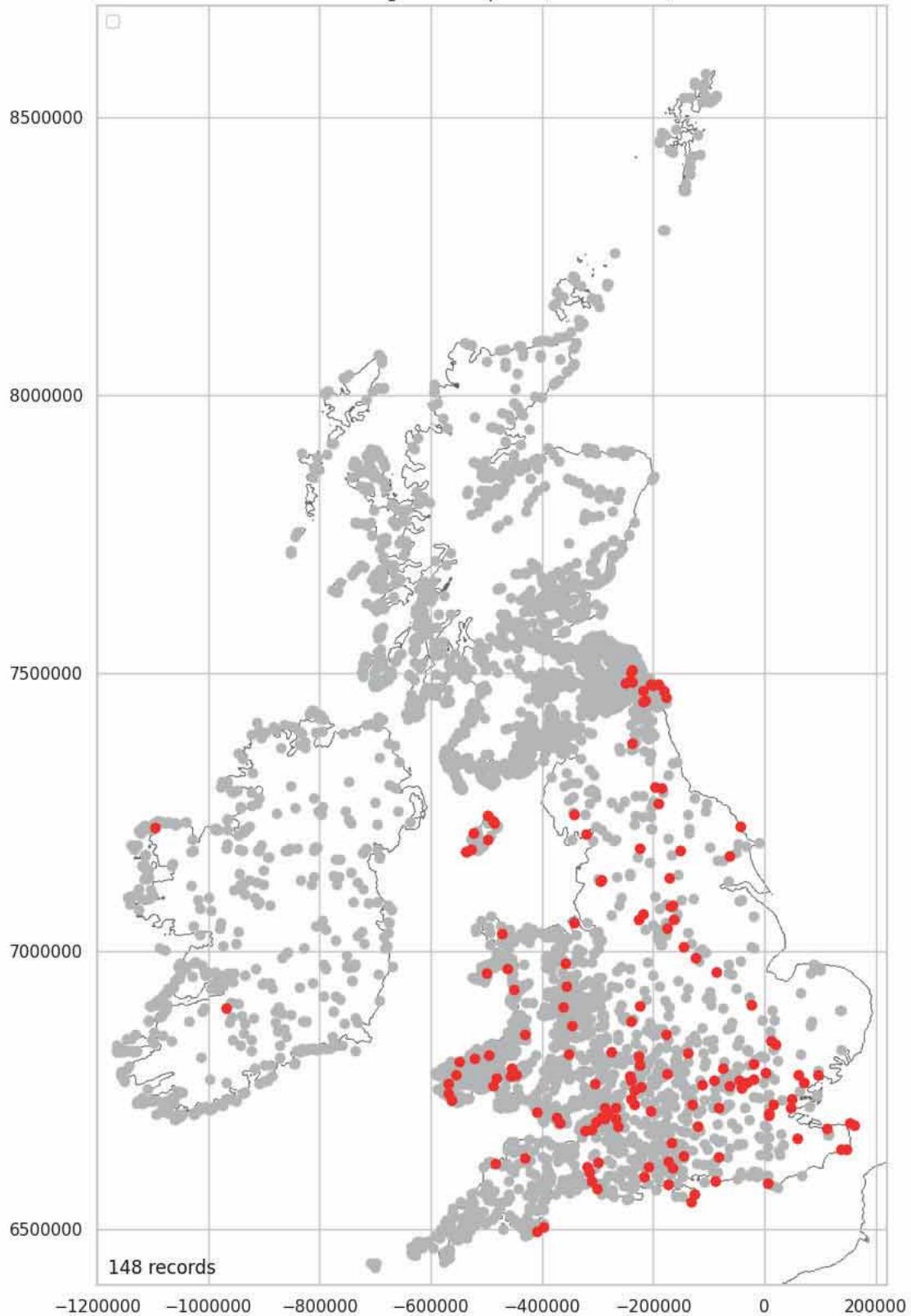
Just 148 (3.57%) of hillforts have not had the presence of ramparts recorded. Almost all are in England, Wales and the Isle of Man.

```

In [ ]: nan_ramparts = \
location_enclosing_data[location_enclosing_data['Enclosing_Max_Ramparts']==-1].copy()
nan_ramparts['Enclosing_Max_Ramparts'] = "Yes"
nan_ramparts_stats = \
plot_over_grey(nan_ramparts, 'Enclosing_Max_Ramparts', 'Yes', '(Not recorded)')

```

Enclosing Max Ramparts (Not recorded)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

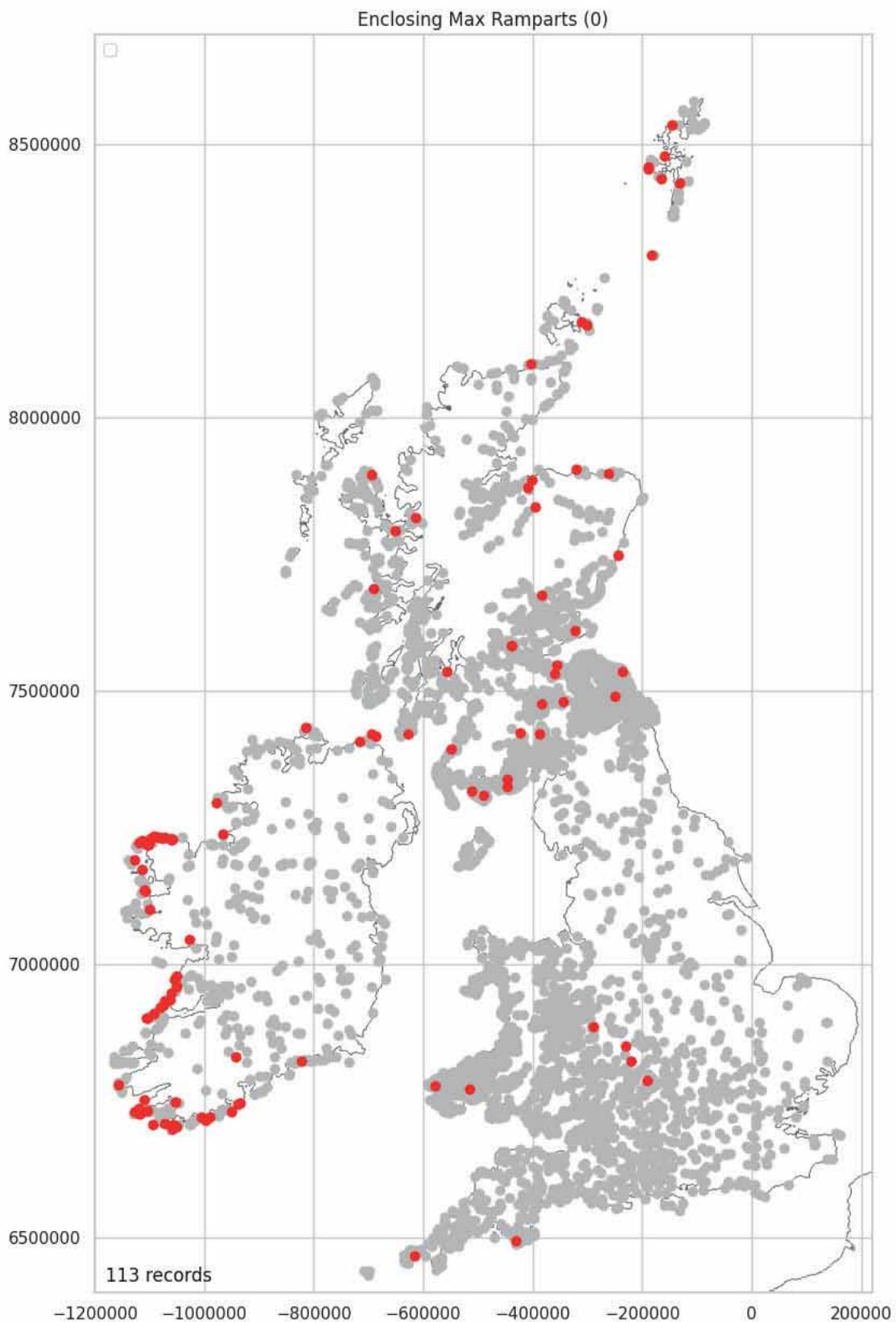
3. 57%

Ramparts Mapped (0)

Hillforts without ramparts are dominated by the coastal forts of Ireland. There is a peppering across Scotland with many again located on the coast. There are very few in England and Wales.

```
In [ ]: zero_ramparts = \
location_enclosing_data[location_enclosing_data['Enclosing_Max_Ramparts']==0].copy()
zero_ramparts['Enclosing_Max_Ramparts'] = "Yes"
```

```
zero_ramparts_stats = plot_over_grey(zero_ramparts, \
    'Enclosing_Max_Ramparts', 'Yes', '(0)')
```



Middleton, M. 2024, Hillforts Primer

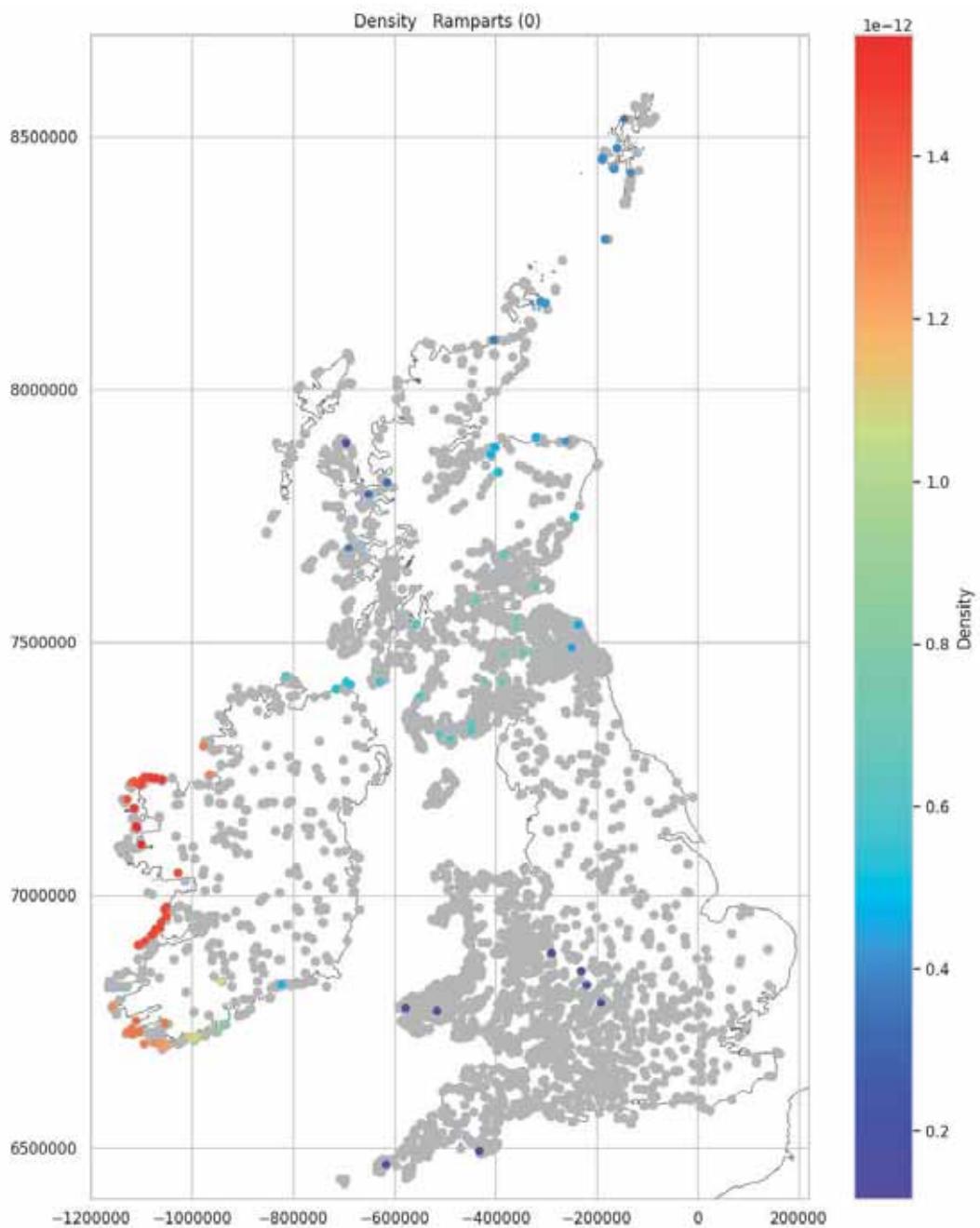
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2. 72%

Ramparts Density Mapped (0)

The west and south coast of Ireland is the focus of hillforts with no ramparts.

```
In [ ]: plot_densitiy_over_grey(zero_ramparts_stats, 'Ramparts (0)')
```



Middleton, M. 2024, Hillforts Primer

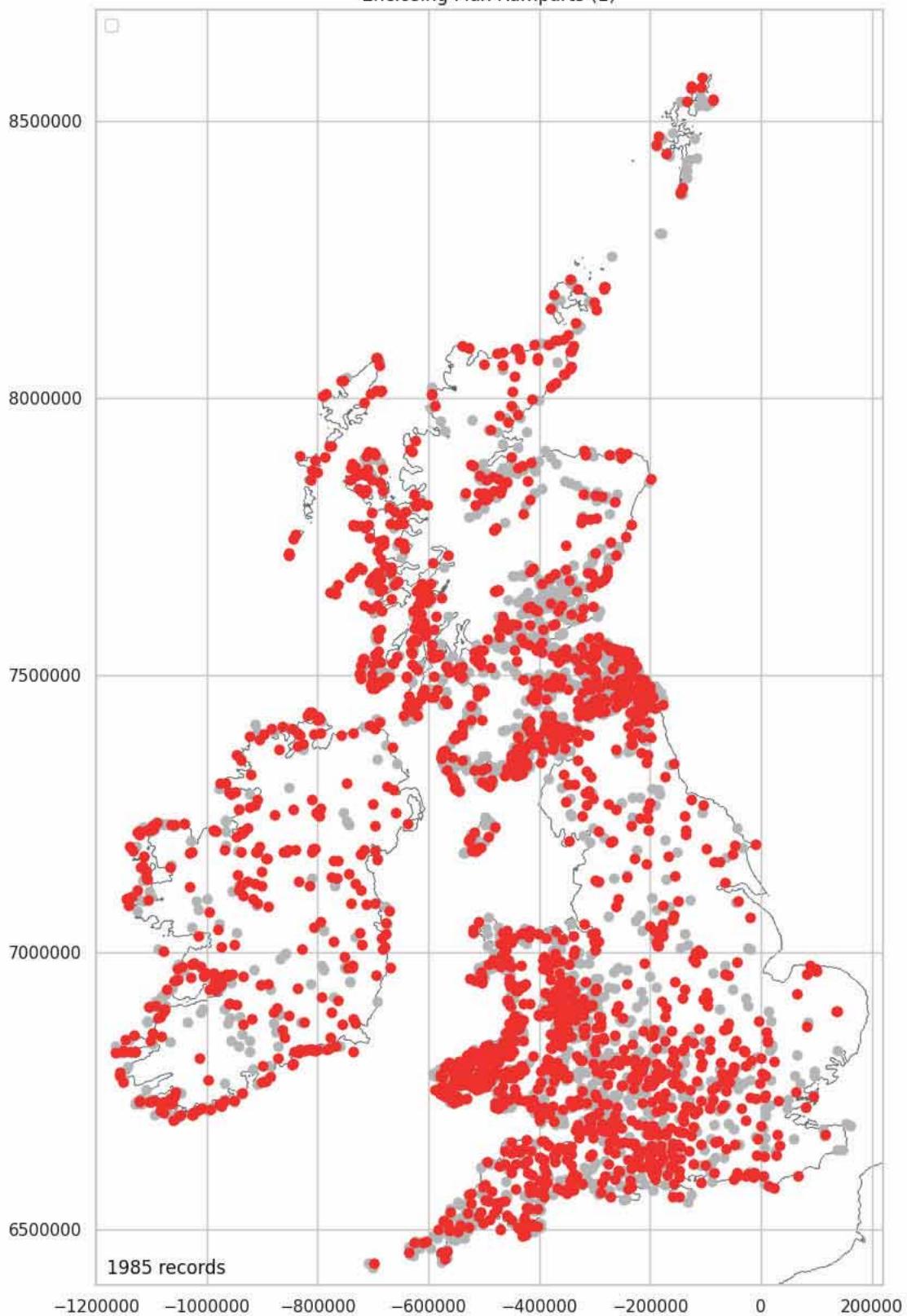
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ramparts Mapped (1)

Hillforts with a single rampart occur right across the Atlas. At 1985 examples (47.87%), a single rampart is the most common rampart layout.

```
In [ ]: one_rampart = \
location_enclising_data[location_enclising_data['Enclising_Max_Ramparts']==1].copy()
one_rampart['Enclising_Max_Ramparts'] = "Yes"
one_rampart_stats = plot_over_grey(one_rampart, \
'Enclising_Max_Ramparts', 'Yes', '(1)')
```

Enclosing Max Ramparts (1)



Middleton, M. 2024, Hillforts Primer

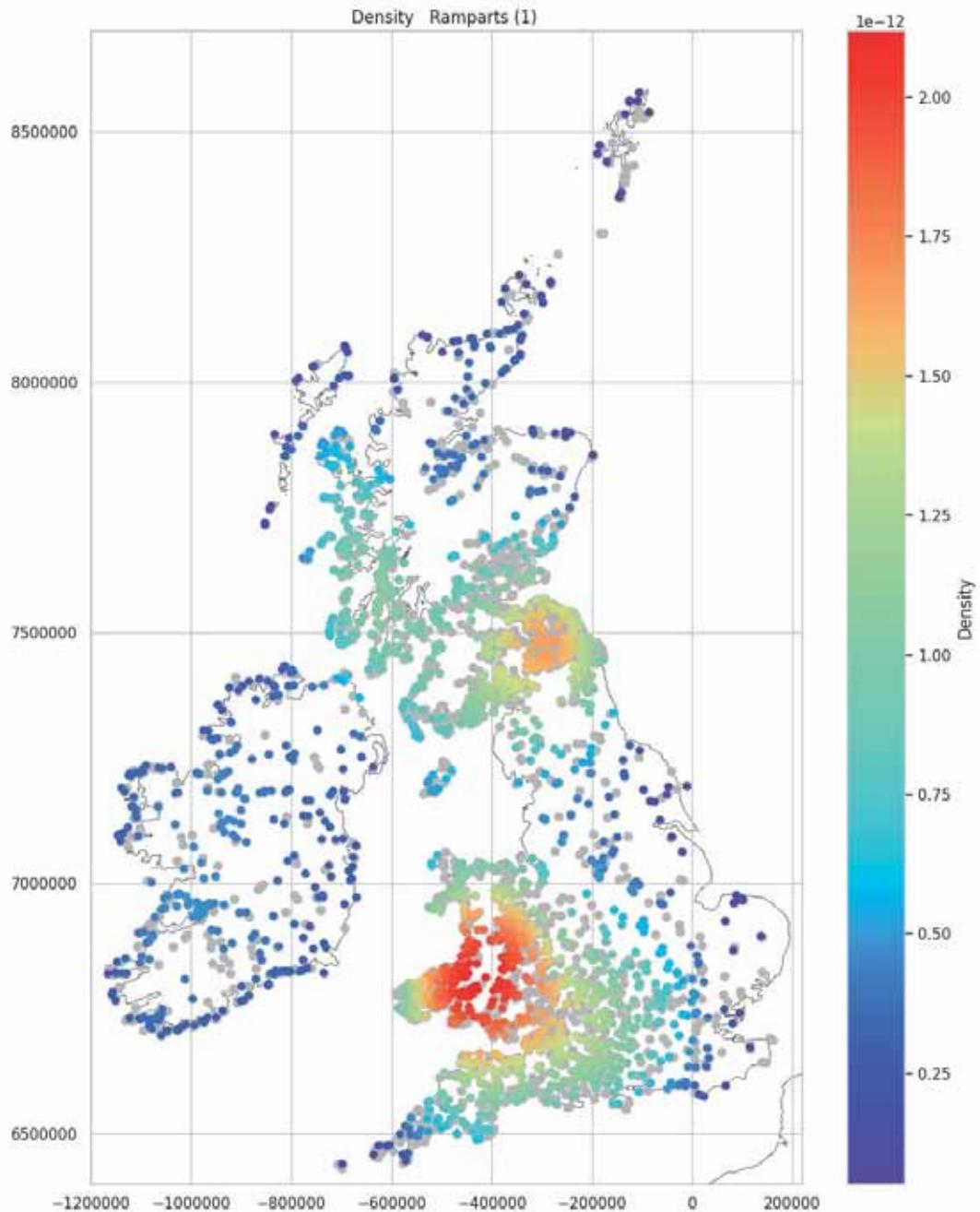
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

47.87%

Ramparts Density Mapped (1)

The density of single rampart forts is most intense in the South, over the southern end of the Cambrian Mountains. This contrasts to the general distribution seen in Part 1: Density Data Mapped where the most intense cluster was in the Northeast. The Northeast does show a cluster, but this is far less intense than that seen in the South. A third cluster can be seen in the Northwest. The distribution across Ireland is very uniform, and there are no significant concentrations.

```
In [ ]: plot_density_over_grey(one_rampart_stats, 'Ramparts (1)')
```



Middleton, M. 2024, Hillforts Primer

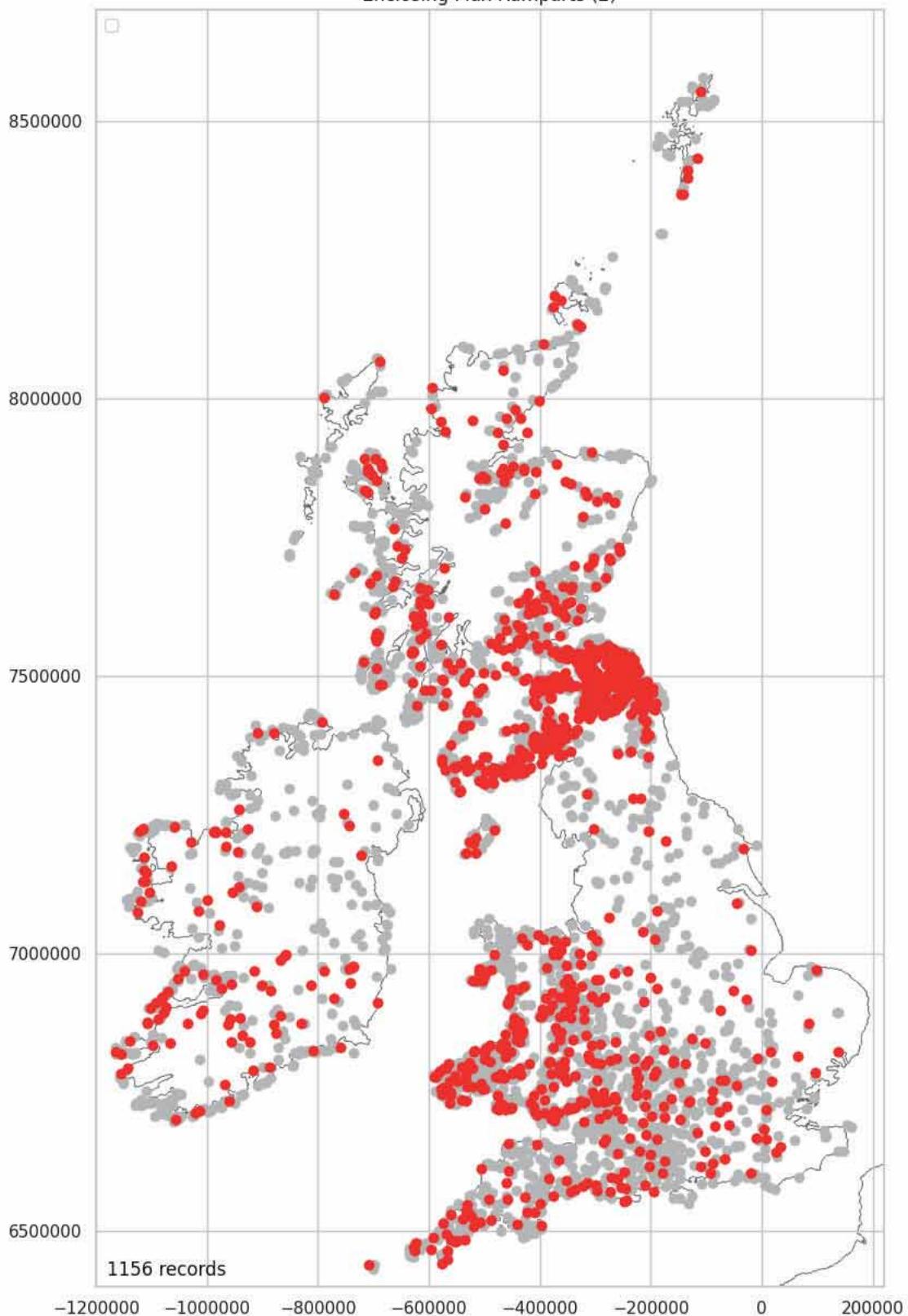
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ramparts Mapped (2)

There are 1356 (27.88%) hillforts with two ramparts. They are distributed mostly in the South, North and across southern Ireland.

```
In [ ]: two_ramparts = \
location_enclusing_data[location_enclusing_data['Enclusing_Max_Ramparts']==2].copy()
two_ramparts['Enclusing_Max_Ramparts'] = "Yes"
two_ramparts_stats = \
plot_over_grey(two_ramparts, 'Enclusing_Max_Ramparts', 'Yes', '(2)')
```

Enclosing Max Ramparts (2)



Middleton, M. 2024, Hillforts Primer

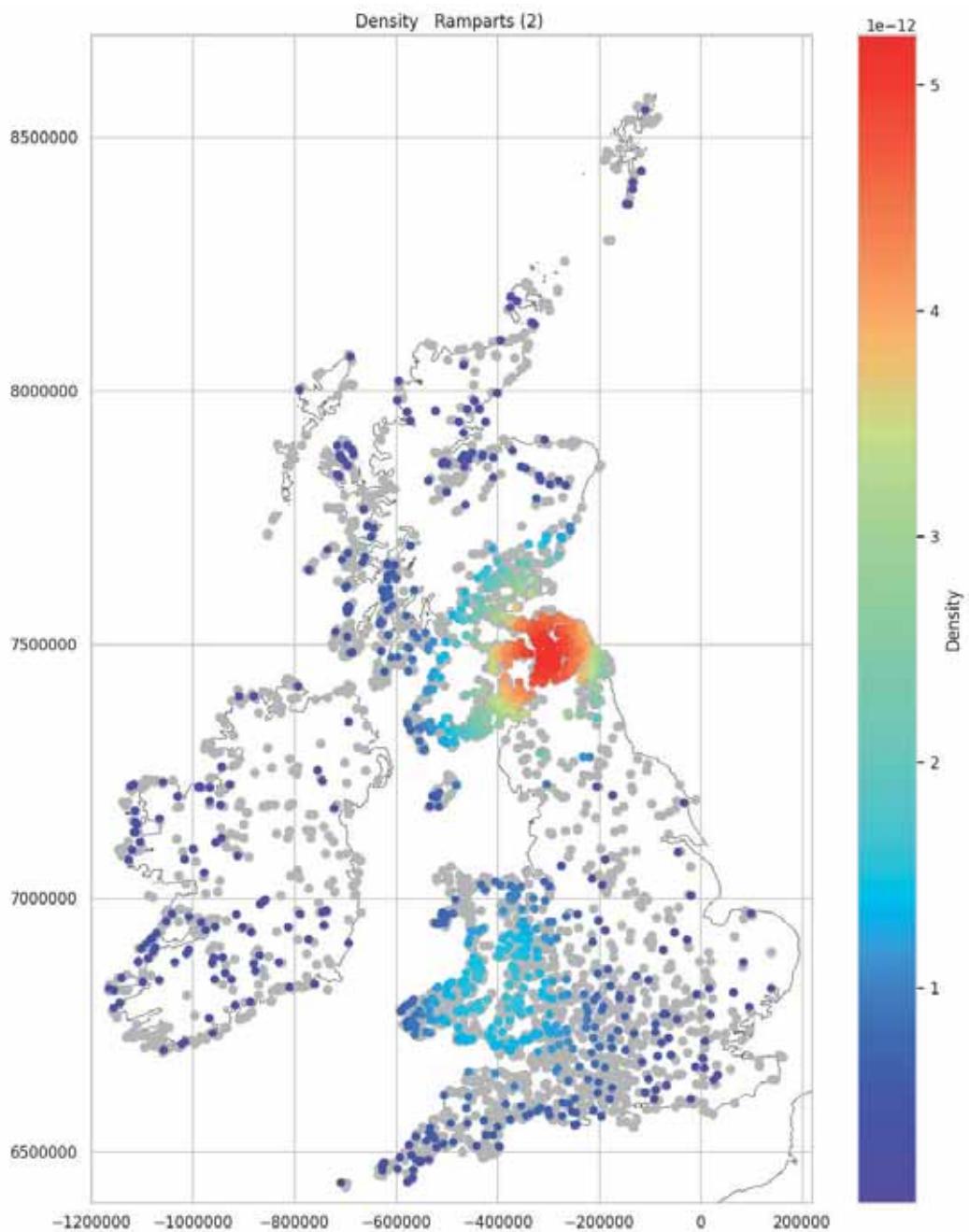
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

27.88%

Ramparts Density Mapped (2)

Hillforts with two ramparts cluster, most intensely, in the Northeast. There is a secondary, weak cluster, at the southern end of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(two_ramparts_stats, 'Ramparts (2)')
```



Middleton, M. 2024, Hillforts Primer

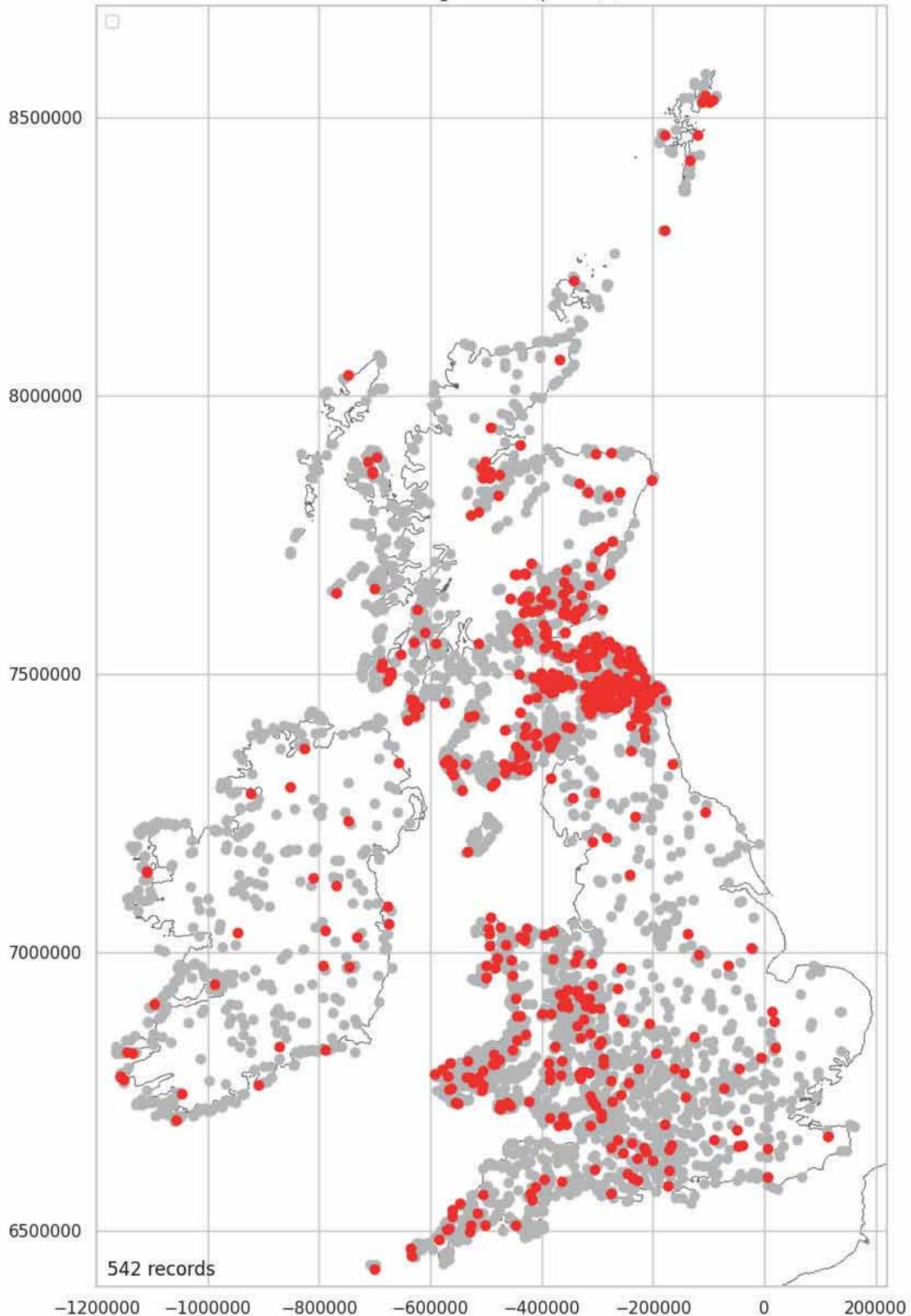
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ramparts Mapped (3)

542 hillforts (13.07%) are recorded as having three ramparts. These cluster in the Northeast and South and they are peppered lightly across Ireland.

```
In [ ]: three_ramparts = \
location_enclising_data[location_enclising_data['Enclising_Max_Ramparts']==3].copy()
three_ramparts['Enclising_Max_Ramparts'] = "Yes"
three_ramparts_stats = plot_over_grey(three_ramparts, \
'Enclising_Max_Ramparts', 'Yes', '(3)')
```

Enclosing Max Ramparts (3)



Middleton, M. 2024, Hillforts Primer

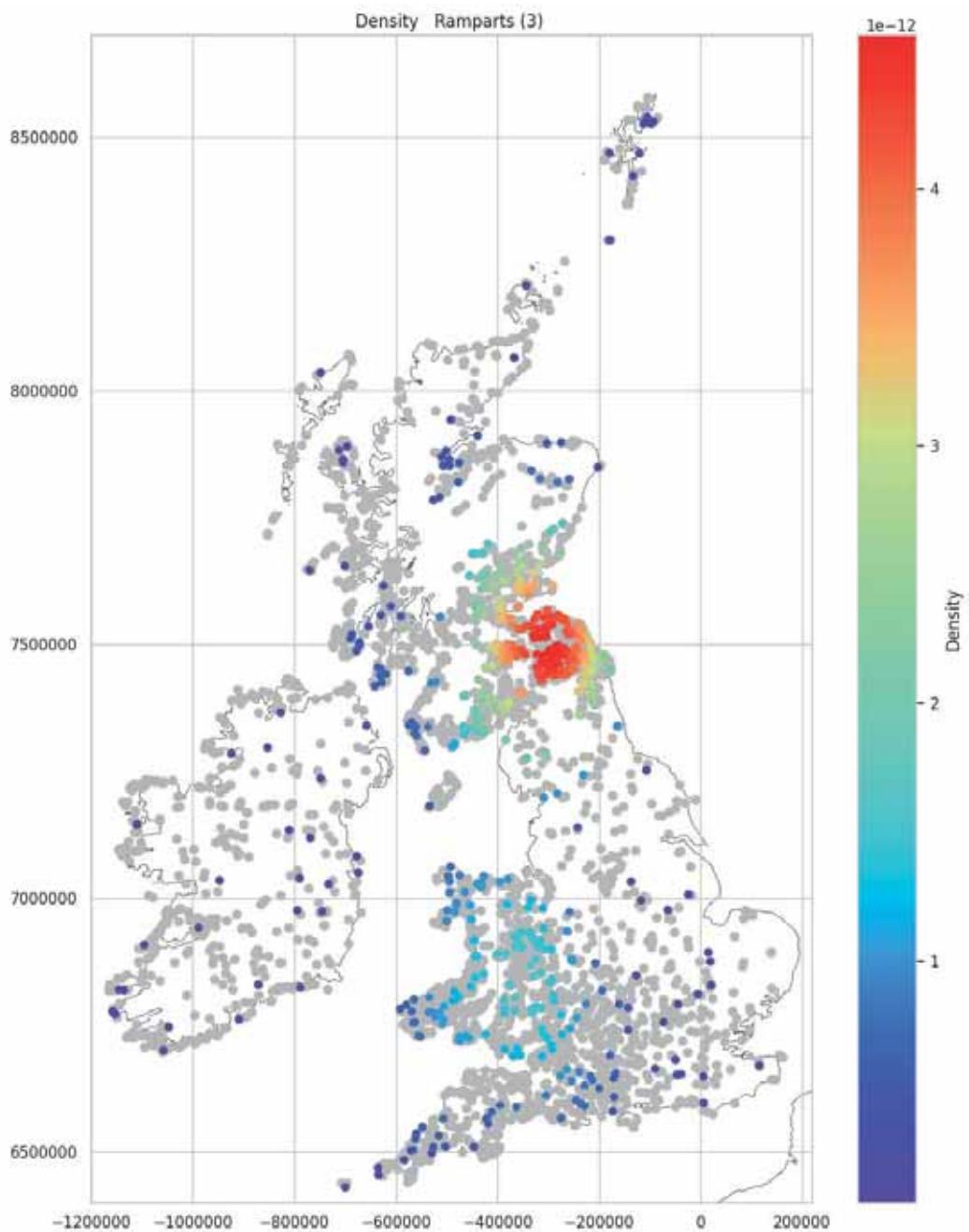
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

13.07%

Ramparts Density Mapped (3)

The main focus of hillforts with three ramparts is in the Northeast. There is a weak clustering along the eastern fringe of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(three_ramparts_stats, 'Ramparts (3)')
```



Middleton, M. 2024, Hillforts Primer

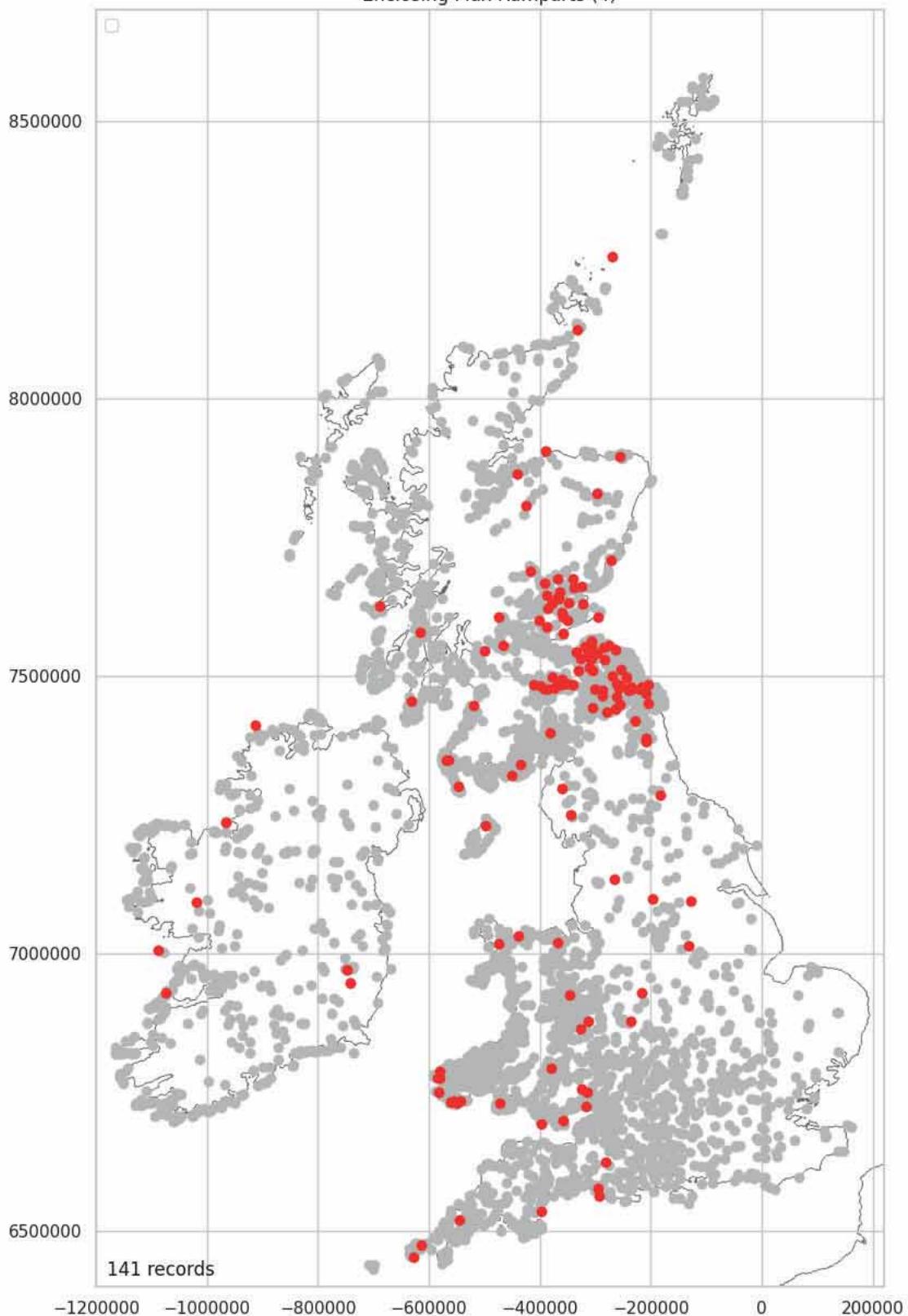
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ramparts Mapped (4)

141 hillforts (3.4%) have four ramparts. The distribution of these is noticeably concentrated in the Northeast and up into Fife, Perthshire and Angus.

```
In [ ]: four_ramparts = \
location_enclusing_data[location_enclusing_data['Enclusing_Max_Ramparts']==4].copy()
four_ramparts['Enclusing_Max_Ramparts'] = "Yes"
four_ramparts_stats = plot_over_grey(four_ramparts, \
'Enclusing_Max_Ramparts', 'Yes', '(4)')
```

Enclosing Max Ramparts (4)



Middleton, M. 2024, Hillforts Primer

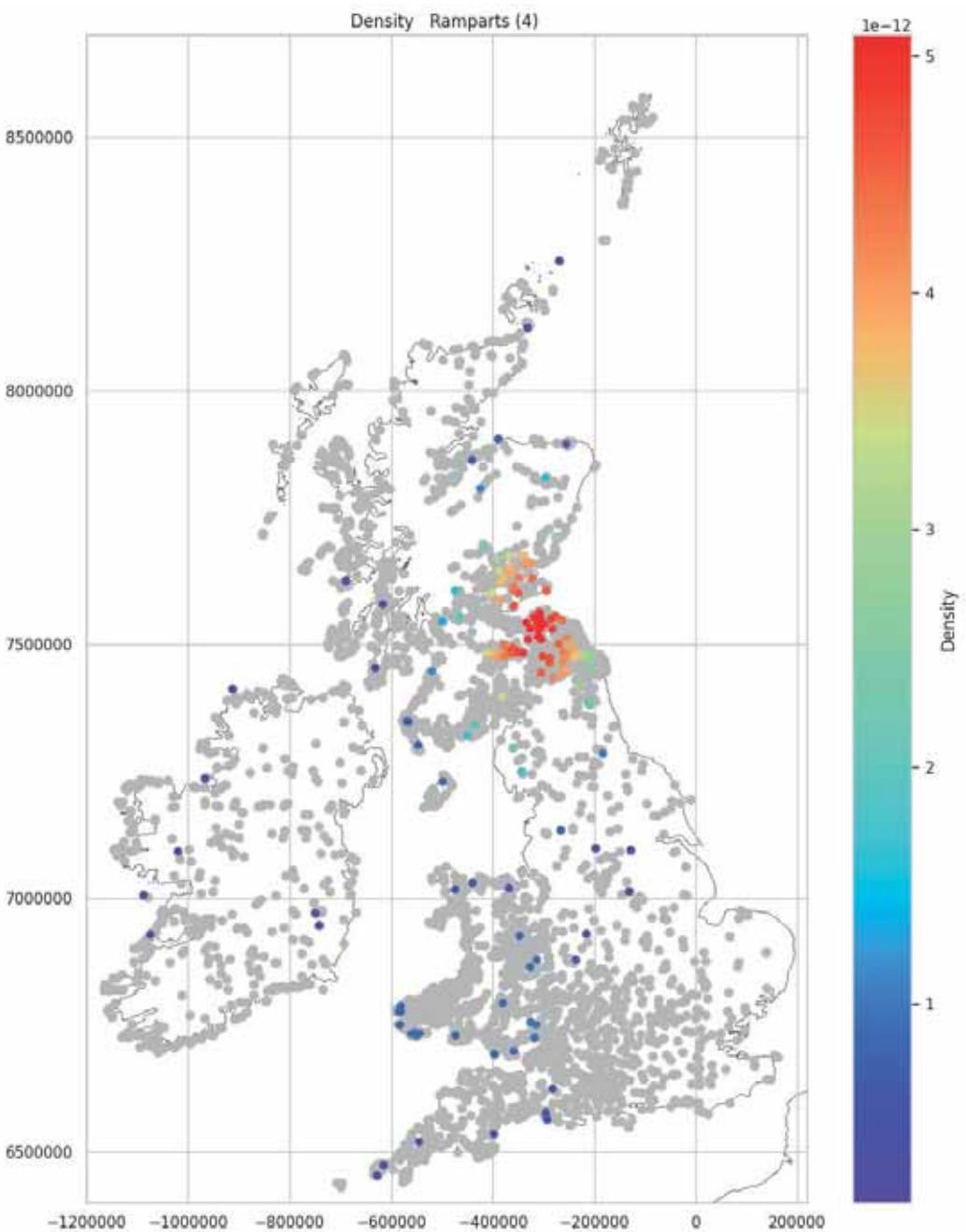
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3. 4%

Ramparts Density Mapped (4)

The focus for four rampart hillforts is in the Northeast over East Lothian.

```
In [ ]: plot_density_over_grey(four_ramparts_stats, 'Ramparts (4)')
```



Middleton, M. 2024, Hillforts Primer

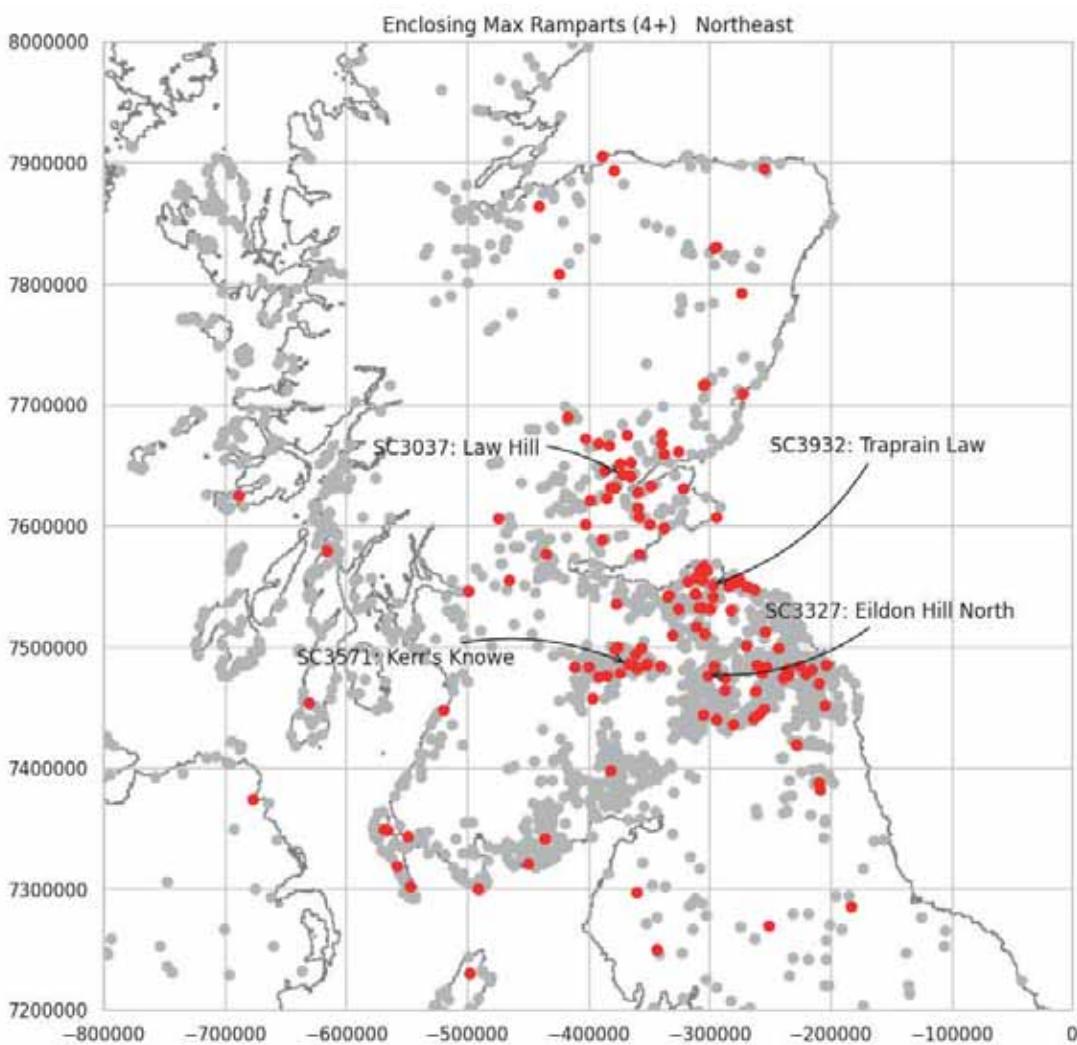
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ramparts Mapped (4+ NE)

In the Northeast, hillforts with four or more ramparts cluster along the eastern fringe of the Southern Uplands, up and across western Fife and on into Perthshire, around Law Hill. There is also a cluster in the vicinity of Kerr's Hill, in south central Scotland. An interesting observation is that this class includes two of the more significant hillforts in southern Scotland, Traprain Law and Eildon Hill North. In East Lothian, the focus of the main cluster is around Traprain Law. It is important to note that this is an area which has undergone intensive aerial survey (See: Part 2: Cropmark Mapped) and there is thus a significant survey bias in this area.

```
In [ ]: location_enclising_data_ne = \
location_enclising_data[location_enclising_data['Location_Y'] > 7070000].copy()
location_enclising_data_ne = \
[location_enclising_data_ne['Location_X'] > -800000].copy()
outlier_ramparts_ne = \
location_enclising_data_ne[
    ['Enclising_Max_Ramparts'] > 3].copy()
outlier_ramparts_ne['Enclising_Max_Ramparts'] = "Yes"
```

```
In [ ]: outlier_ramparts_stats_ne = \
plot_over_grey_north(outlier_ramparts_ne, 'Enclising_Max_Ramparts', 'Yes', \
'(4+) - Northeast', 'Traprain')
```



See: [Ditches Mapped \(4+ NE\)](#)

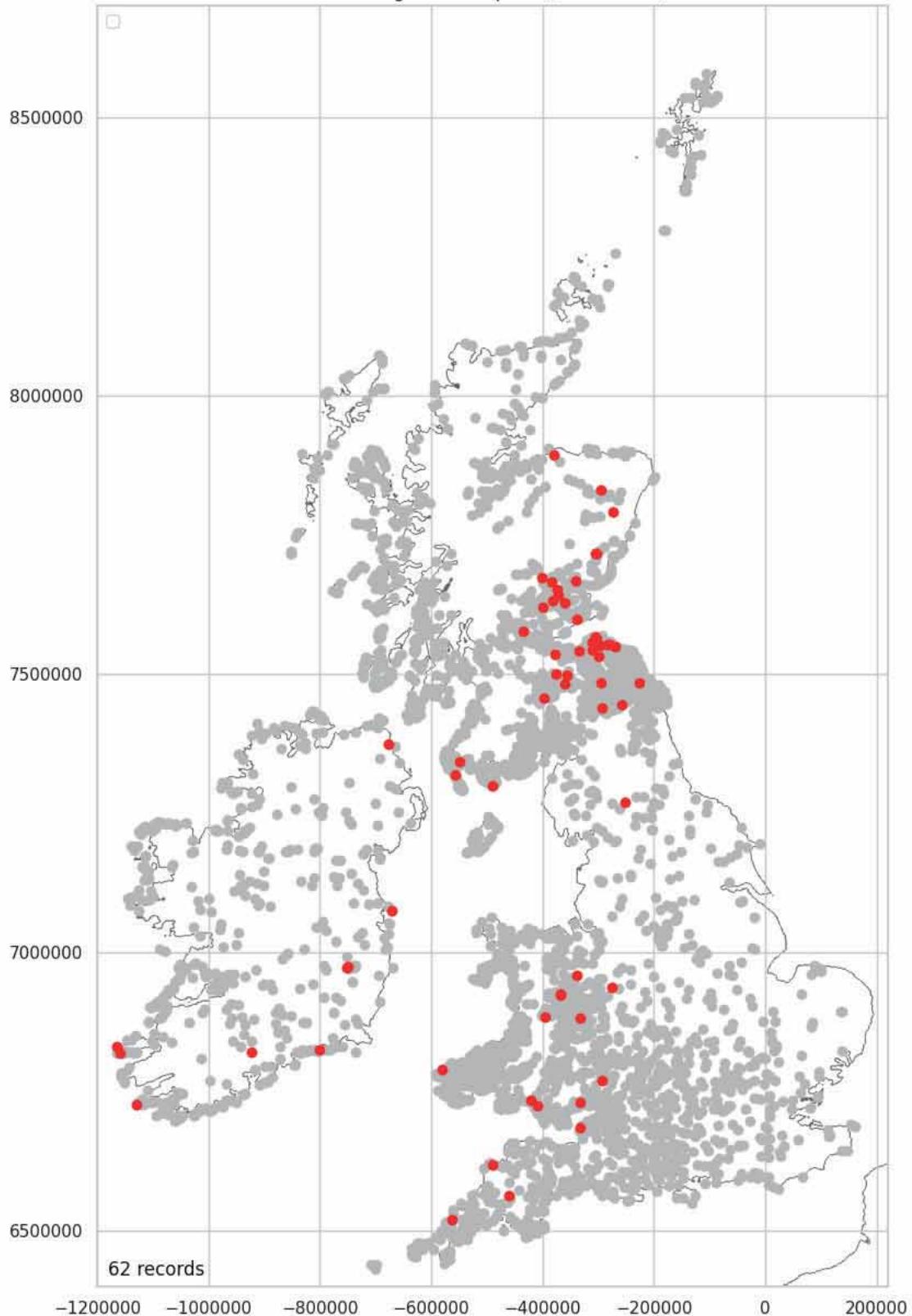
```
In [ ]: # This code can be used to get details of hillforts within certain x and y coordinate ranges
# To use this code, first run the document using Runtime > Run all, then remove the '#' from the lines
# starting temp below. Once removed press the Run cell button, on this cell, to the left.
# Update the 'Location_X' & 'Location_Y' values as required.
# temp = pd.merge(name_and_number, outlier_ramparts_ne, left_index=True, right_index=True)
# temp = temp[temp['Location_X'].between(-300000, -200000)]
# temp = temp[temp['Location_Y'].between(7700000, 7800000)]
# temp
```

Ramparts Mapped (5+ Outliers)

West-Town, Waterford, in southeast Ireland, is the only Hillfort recorded as having 10 ramparts. Only 62 hillforts are recorded as having five or more ramparts and most are in the Northeast.

```
In [ ]: outlier_ramparts = \
location_enclosing_data[location_enclosing_data['Enclousing_Max_Ramparts']>4].copy()
outlier_ramparts['Enclousing_Max_Ramparts'] = "Yes"
outlier_ramparts_stats = \
plot_over_grey(outlier_ramparts, 'Enclousing_Max_Ramparts', 'Yes', '(5+ Outliers)')
```

Enclosing Max Ramparts (5+ Outliers)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 5%

```
In [ ]: most_ramparts = \
location_enclosing_data[location_enclosing_data['Enclosing_Max_Ramparts'] == 10].copy()
most_ramparts = \
pd.merge(name_and_number, most_ramparts, left_index=True, right_index=True)
most_ramparts[['Main_Atlas_Number', 'Main_Display_Name', 'Enclosing_Area_1', \
'Enclosing_Max_Ramparts', 'Enclosing_Ditches_Number']]
```

Out[]:	Main_Atlas_Number	Main_Display_Name	Enclosing_Area_1	Enclosing_Max_Ramparts	Enclosing_Ditches_Number
	1015	1043 West-Town, Waterford (Great Island)		0.71	10.0

Ramparts by Region

Most hillforts have one to two ramparts. In the north of Ireland this is most likely to be one and, in the Northeast, it is most likely to range between one to three.

```
In [ ]: location_enclosing_data_ne = \
pd.merge(north_east.reset_index(), enclosing_numeric_data, left_on='uid', \
         right_index=True)
location_enclosing_data_ne = \
pd.merge(name_and_number, location_enclosing_data_ne, left_index=True, \
         right_on='uid')
```

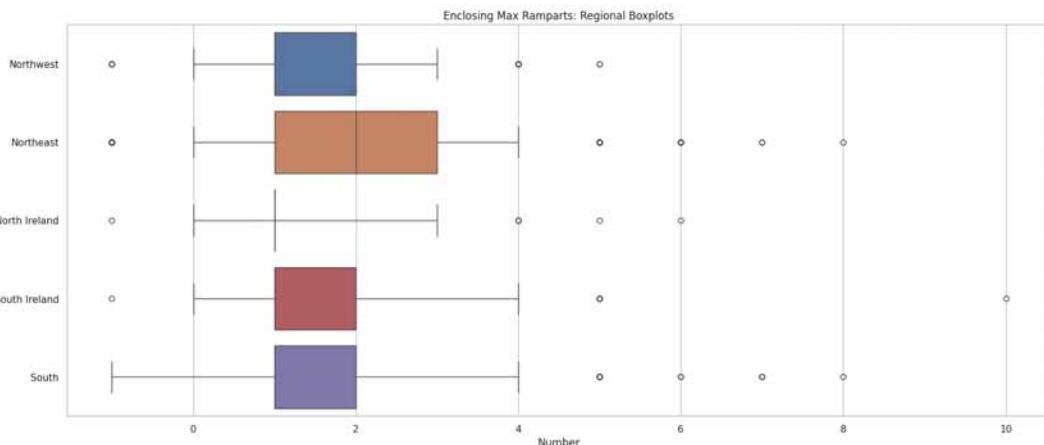
```
In [ ]: location_enclosing_data_nw = \
pd.merge(north_west.reset_index(), enclosing_numeric_data, left_on='uid', \
         right_index=True)
location_enclosing_data_nw = \
pd.merge(name_and_number, location_enclosing_data_nw, left_index=True, \
         right_on='uid')
```

```
In [ ]: location_enclosing_data_ireland_n = \
pd.merge(north_ireland.reset_index(), enclosing_numeric_data, left_on='uid', \
         right_index=True)
location_enclosing_data_iraland_n = \
pd.merge(name_and_number, location_enclosing_data_iraland_n, left_index=True, \
         right_on='uid')
```

```
In [ ]: location_enclosing_data_iraland_s = \
pd.merge(south_ireland.reset_index(), enclosing_numeric_data, left_on='uid', \
         right_index=True)
location_enclosing_data_iraland_s = \
pd.merge(name_and_number, location_enclosing_data_iraland_s, left_index=True, \
         right_on='uid')
```

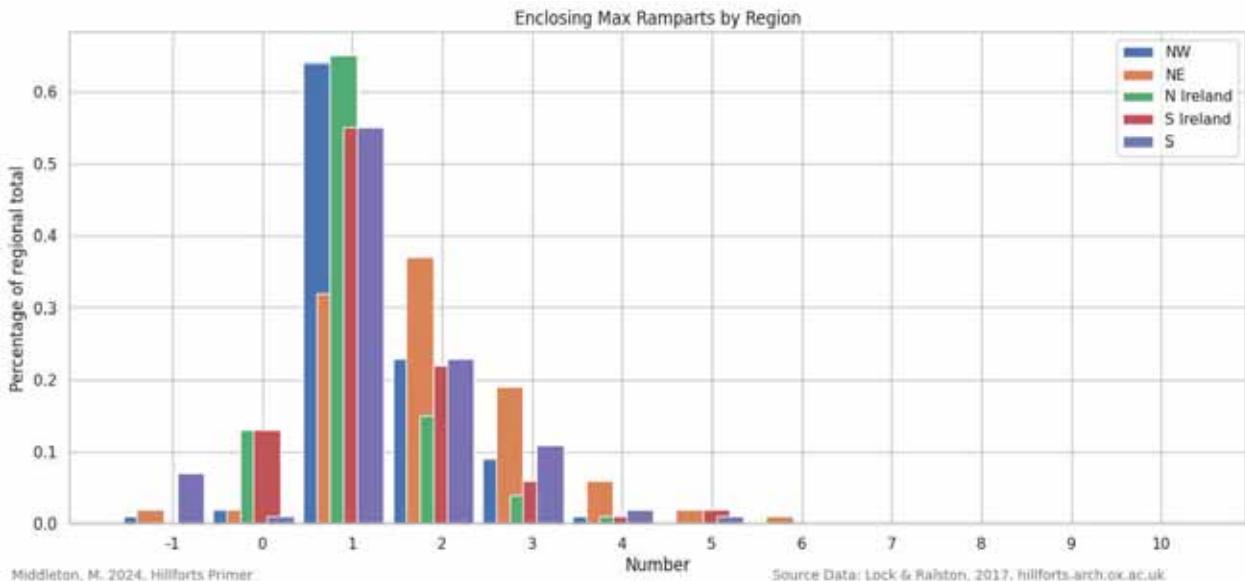
```
In [ ]: location_enclosing_data_south = \
pd.merge(south, enclosing_numeric_data, left_on='uid', right_index=True)
location_enclosing_data_south = \
pd.merge(name_and_number, location_enclosing_data_south, left_index=True, \
         right_on='uid')
```

```
In [ ]: regional_dict = \
{'Northwest': location_enclosing_data_nw\ 
['Enclosing_Max_Ramparts'], 'Northeast': location_enclosing_data_ne\ 
['Enclosing_Max_Ramparts'], 'North_Ireland': location_enclosing_data_iraland_n\ 
['Enclosing_Max_Ramparts'], 'South_Ireland': location_enclosing_data_iraland_s\ 
['Enclosing_Max_Ramparts'], 'South': location_enclosing_data_south\ 
['Enclosing_Max_Ramparts']}
plot_data = pd.DataFrame.from_dict(regional_dict)
plt.figure(figsize=(20,8))
ax = sns.boxplot(data=plot_data, orient="h", whis=[2.2, 97.8], showfliers=True);
add_annotation_plot(ax)
ax.set_xlabel('Number')
title = 'Enclosing_Max_Ramparts: Regional Boxplots'
plt.title(get_print_title(title))
save_fig(title)
plt.show()
```



The Northeast is noticeable in that hillforts with a single rampart are proportionally far less than in other areas. Similarly, the Northeast is more likely to have forts with two, three or four ramparts than other regions. Hillforts in the remaining regions have quite similar proportions of ramparts apart from forts with no ramparts, which are more common in Ireland.

```
In [ ]: plot_feature_by_region(location_enclosing_data_nw,
                               location_enclosing_data_ne,
                               location_enclosing_data_ireland_n,
                               location_enclosing_data_ireland_s,
                               location_enclosing_data_south,
                               'Enclosing_Max_Ramparts',
                               'Enclosing_Max_Ramparts_by_Region', 12)
```

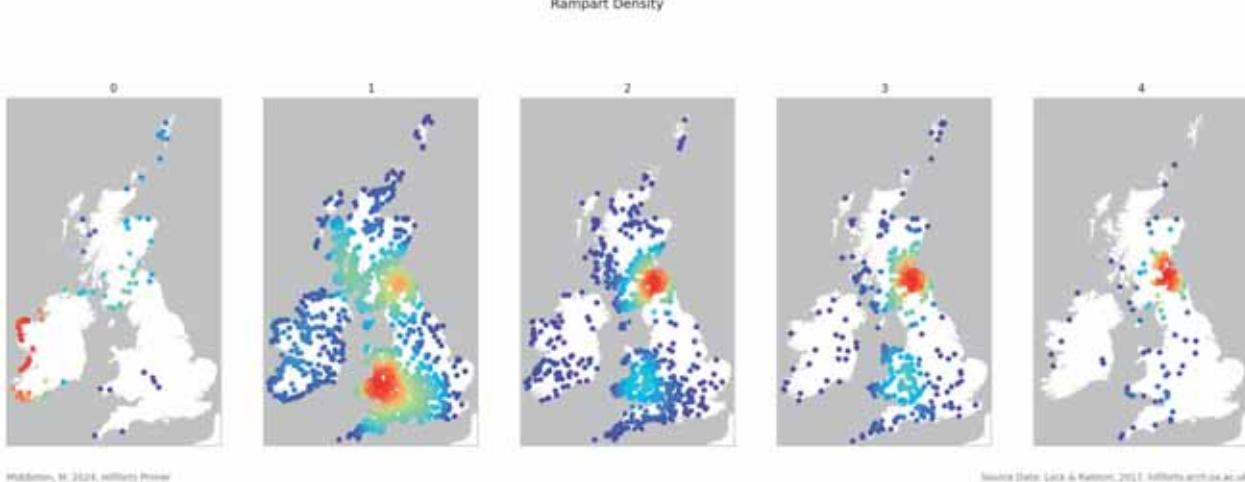


Ramparts Summary

Ramparts show four distinct clusters:

- Hillforts without ramparts along the west coast of Ireland
- Hillforts with one rampart in the Northwest
- Hillforts with mostly one rampart in southern Wales and south-central England (occasionally two or three)
- Hillforts with one or more ramparts in the Northeast

```
In [ ]: plot_density_over_grey_five(zero_ramparts, one_rampart, two_ramparts, \
                                    three_ramparts, four_ramparts, 'Rampart Density')
```



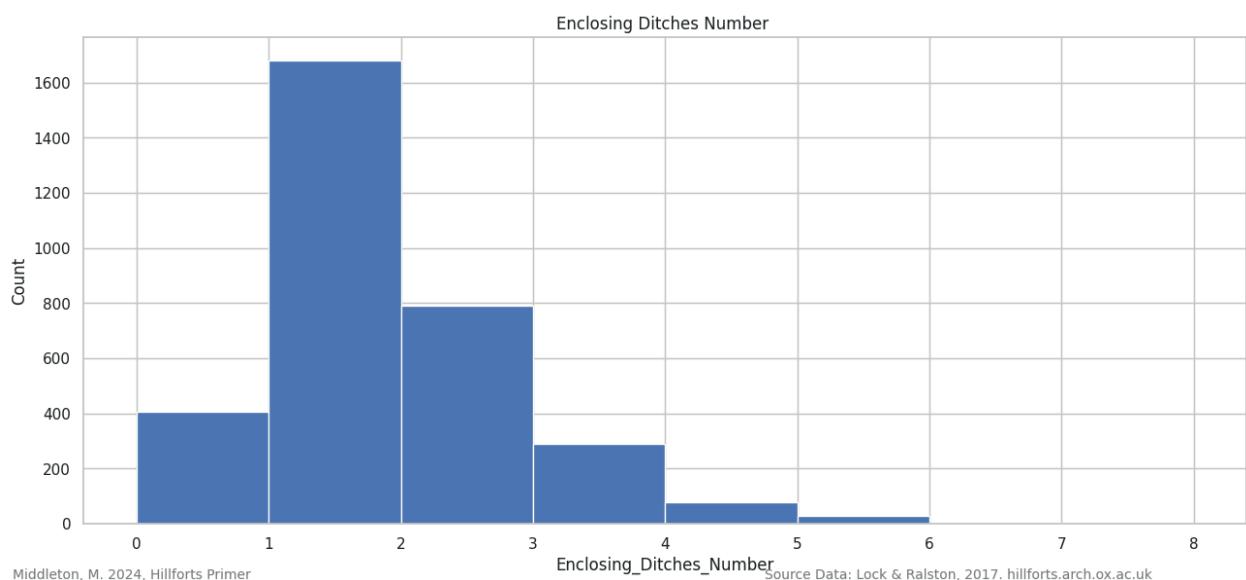
Ditches Plotted

868 hillforts (20.93%) have no information regarding ditches. These are mostly in Scotland. Of those that are recorded, 1681 (40.54%) have one ditch; 789 (19.3%) have two ditches and 406 (9.79%) no ditches. Because of the lack of recording in Scotland and what looks like a survey bias in the forts with no recorded ditches, caution should be taken when interpreting these distributions. The fort with the most ditches is Trevelgue Head in Cornwall which has eight.

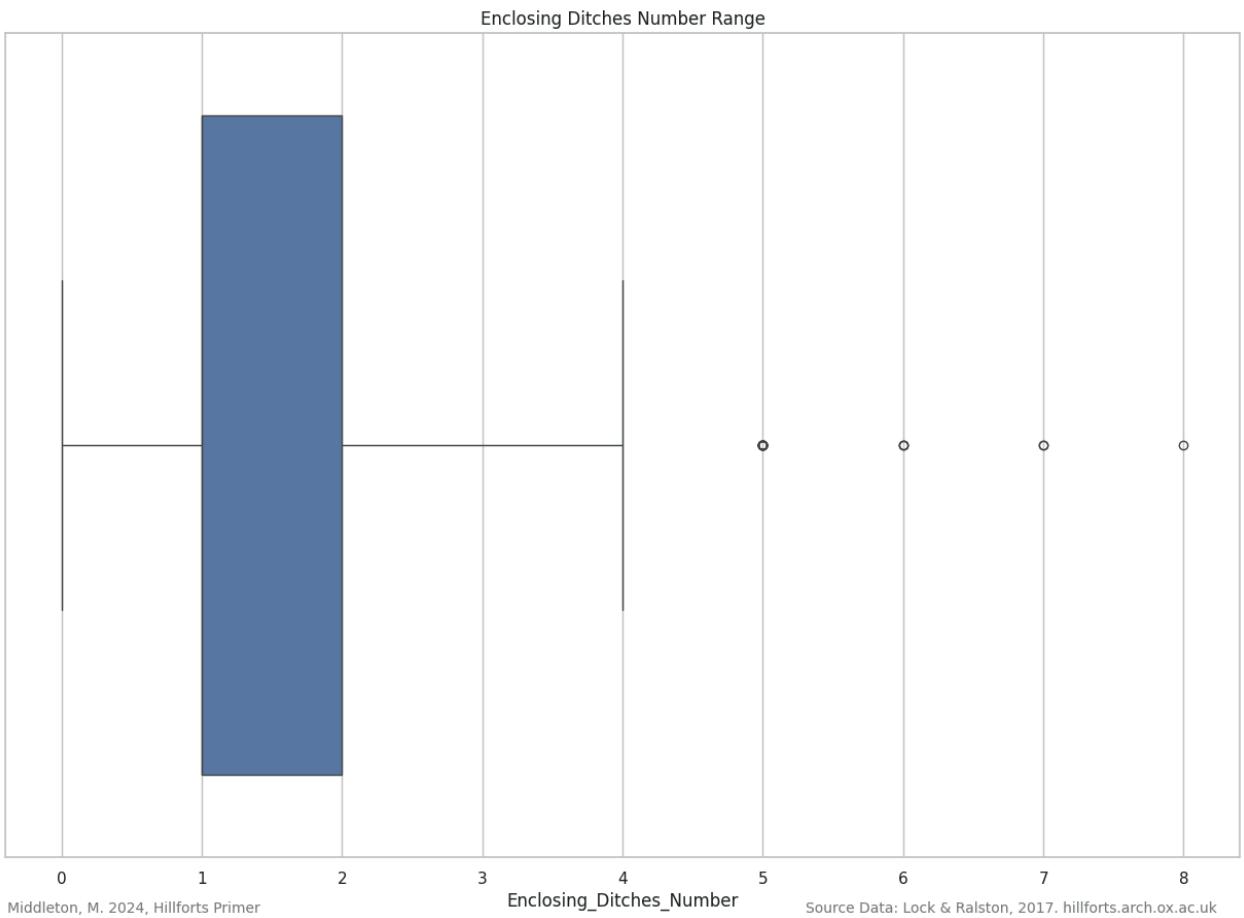
```
In [ ]: ditches_location_on_enc_data = \
location_enclusing_data[location_enclusing_data['Enclusing_Ditches_Number'] >= 0]
ditches_location_on_enc_data['Enclusing_Ditches_Number'].value_counts().sort_index()
```

```
Out[ ]: 0.0    406
1.0    1681
2.0    789
3.0    289
4.0     79
5.0     29
6.0      3
7.0      2
8.0      1
Name: Enclusing_Ditches_Number, dtype: int64
```

```
In [ ]: plot_bar_chart_numeric(ditches_location_on_enc_data, 1, \
'Enclusing_Ditches_Number', 'Count', \
'Enclusing_Ditches_Number', \
int(ditches_location_on_enc_data\
['Enclusing_Ditches_Number'].max()))
```



```
In [ ]: ditches_data = \
plot_data_range(ditches_location_on_enc_data['Enclusing_Ditches_Number'], \
reset_index(drop = True), 'Enclusing_Ditches_Number', "h")
```



```
In [ ]: ditches_data
```

```
Out[ ]: [0.0, 1.0, 1.0, 2.0, 4.0]
```

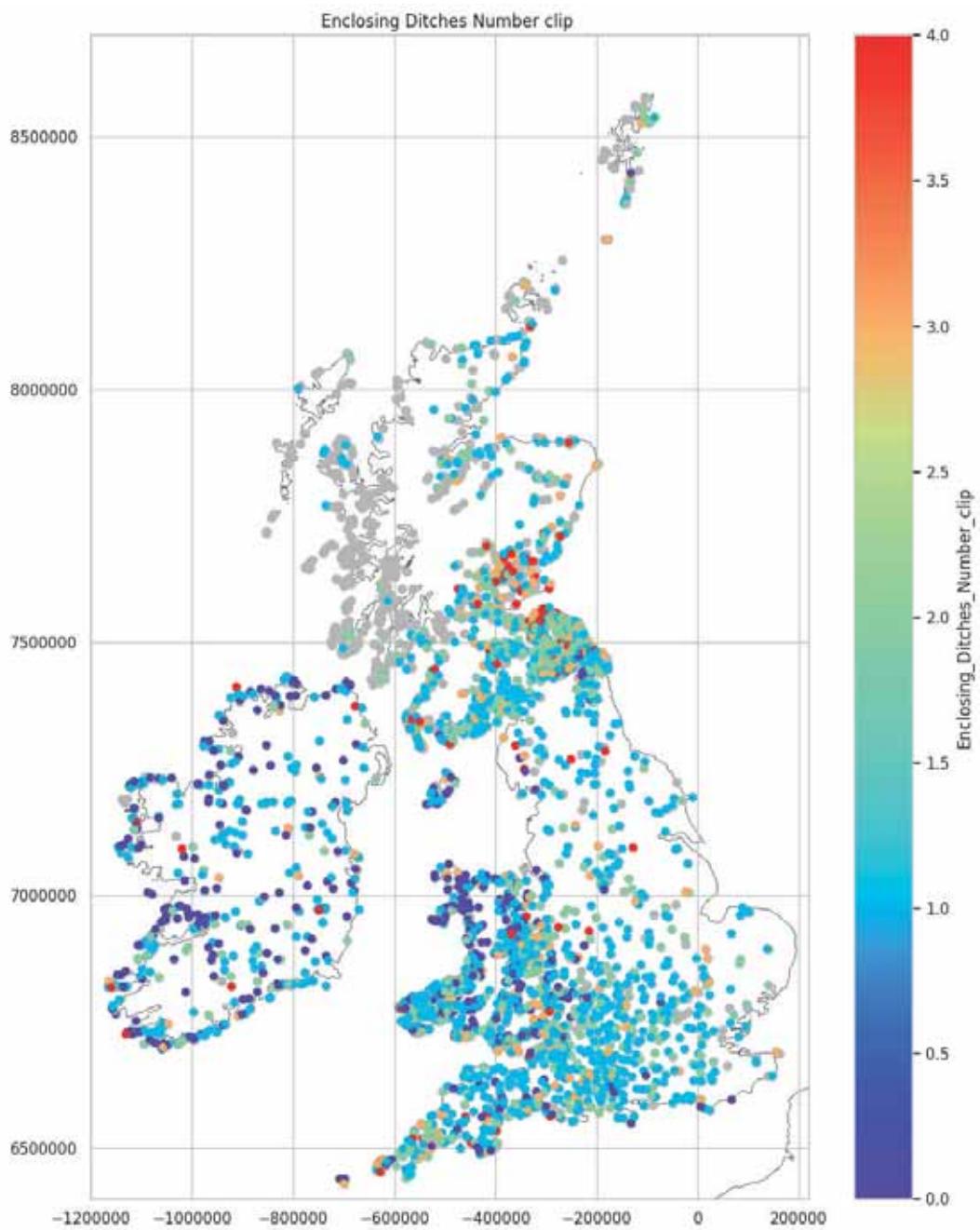
Ditches Clipped Mapped

As with ramparts, the combined plot is difficult to read so each value will be reviewed individually. There is a noticeable survey bias in the data across Scotland. There are very few records in the Northwest.

```
In [ ]: ditches_clip = ditches_location_enc_data.copy()
ditches_clip['Enclosing_Ditches_Number_clip'] = \
ditches_clip['Enclosing_Ditches_Number'].\
clip(ditches_clip['Enclosing_Ditches_Number'], ditches_data[-1], axis=0)
ditches_clip['Enclosing_Ditches_Number_clip'].value_counts().sort_index()
```

```
Out[ ]: 0.0    406
1.0    1681
2.0    789
3.0    289
4.0    114
Name: Enclosing_Ditches_Number_clip, dtype: int64
```

```
In [ ]: plot_type_values(ditches_clip, \
                           'Enclosing_Ditches_Number_clip', \
                           'Enclosing_Ditches_Number_clip')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

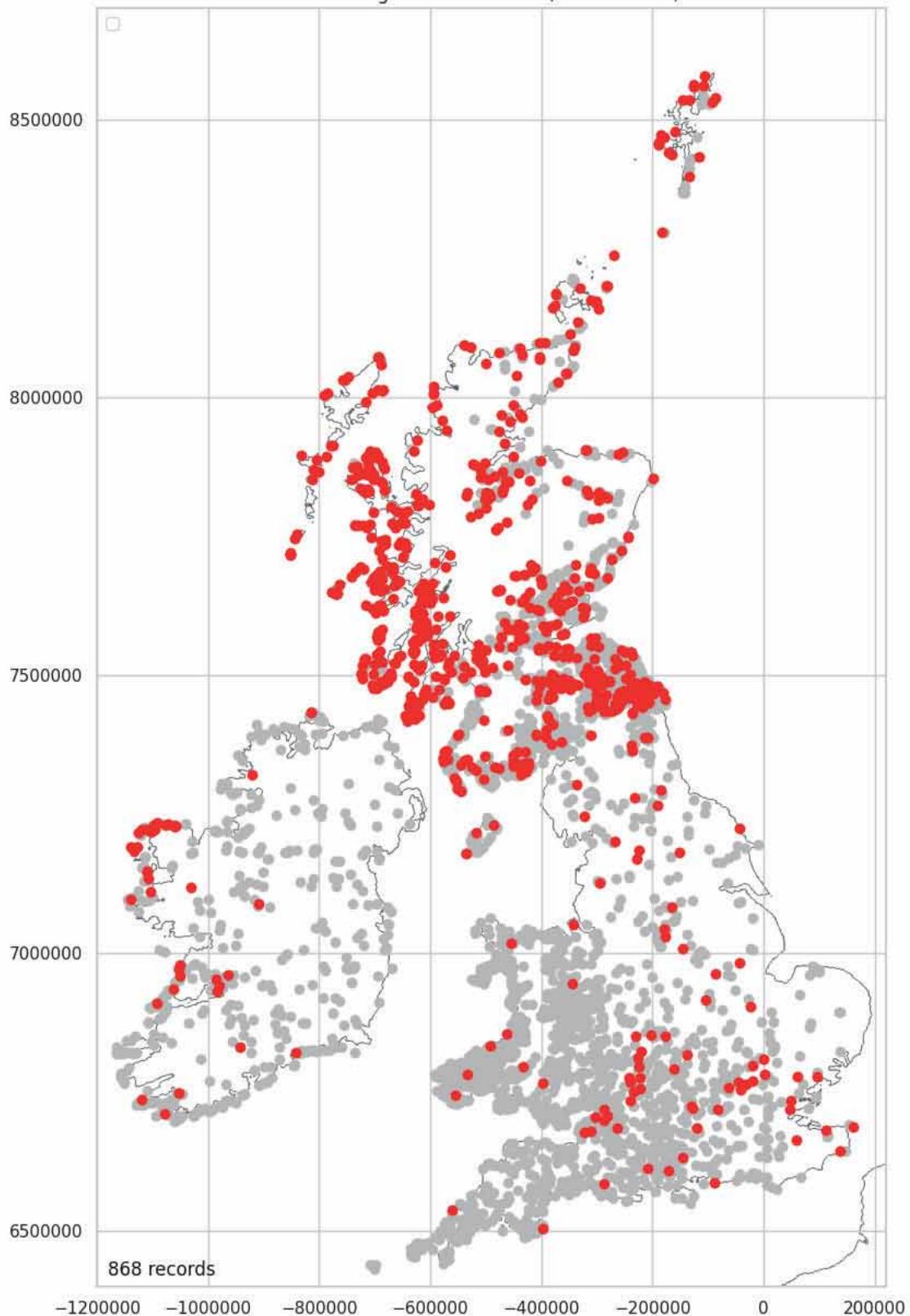
See: [Enclosing Ditches Density Mapped](#)

Ditches Mapped (Not Recorded)

A remarkable 868 (20.93%) of hillforts have no record of the presence of a ditch. Most of these are in Scotland. This is a survey bias in the data. It is likely that this is partly due to a practice of not recording ditches being used as a shorthand for there not being ditches. The hard geology, of the north, and surveyors thinking it is obvious combining to create ambiguous records. This would be an obvious area where a study could rapidly improve this section of the atlas.

```
In [ ]: nan_ditches = \
location_enclosing_data[location_enclosing_data['Enclosing_Ditches_Number']==-1].copy()
nan_ditches['Enclosing_Ditches_Number'] = "Yes"
nan_ditches_stats = plot_over_grey(nan_ditches, 'Enclosing_Ditches_Number', \
'Yes', '(Not recorded)')
```

Enclosing Ditches Number (Not recorded)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

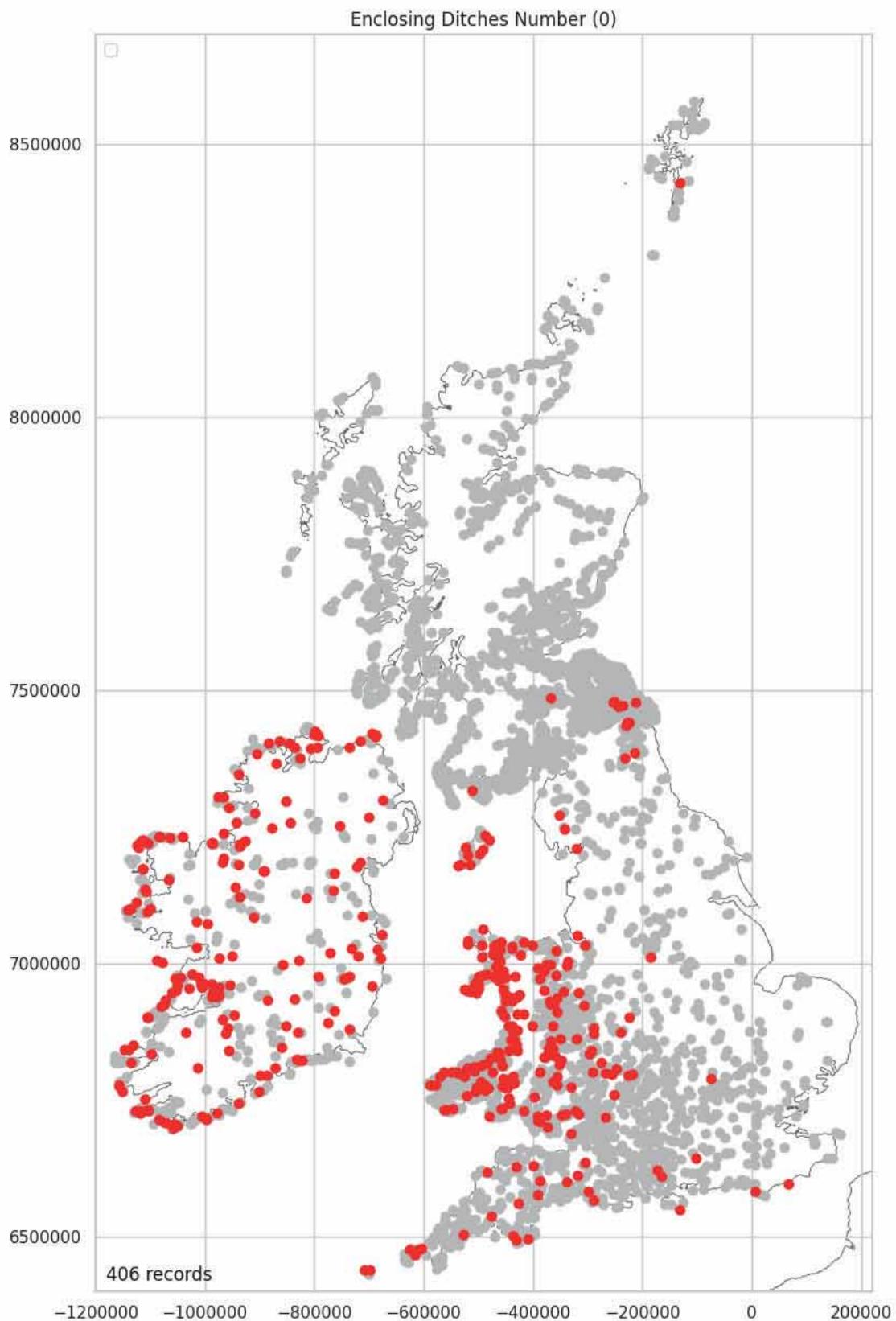
20. 93%

Ditches Mapped (0)

With the survey bias across Scotland in mind, see: [Ditches Mapped \(Not Recorded\)](#), the distribution of forts with no ditches is very much over Wales and Ireland.

```
In [ ]: zero_ditches = \
location_enclosing_data[location_enclosing_data['Enclosing_Ditches_Number']==0].copy()
zero_ditches['Enclosing_Ditches_Number'] = "Yes"
```

```
zero_ditches_stats = plot_over_grey(zero_ditches, 'Enclosing_Ditches_Number', \
'Yes', '(0)')
```



Middleton, M. 2024, Hillforts Primer

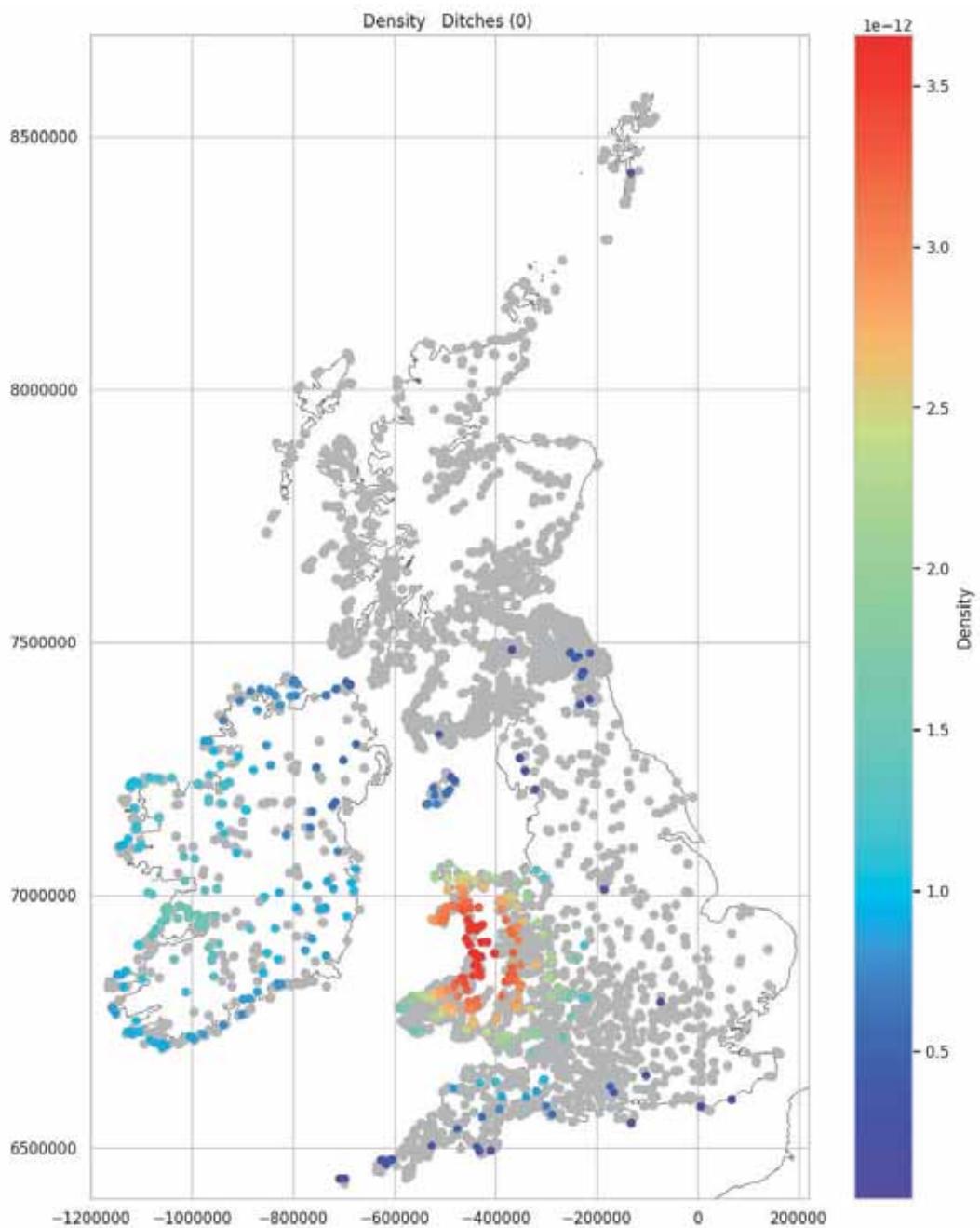
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

9.79%

Ditches Density Mapped (0)

The most intense cluster, of hillforts with no ditches, is over the western fringe of the Cambrian Mountains.

```
In [ ]: plot_densitiy_over_grey(zero_ditches_stats, 'Ditches (0)')
```



Middleton, M. 2024, Hillforts Primer

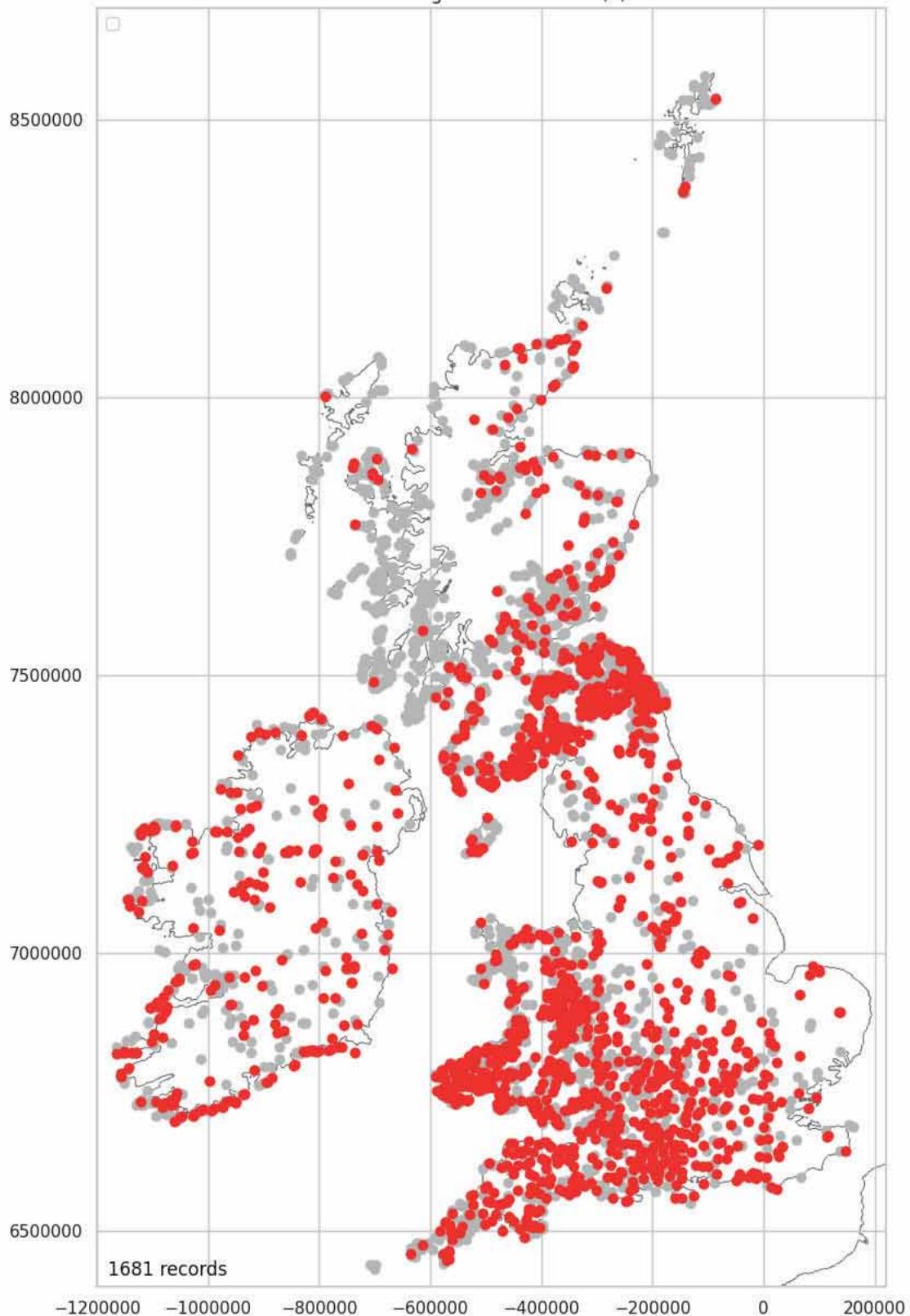
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ditches Mapped (1)

The distribution of single ditch hillforts is much more uniform. 1681 (40.84%) of hillforts fall into this class. Again, see [Ditches Mapped \(Not Recorded\)](#).

```
In [ ]: one_ditches = \
location_enclising_data[location_enclising_data['Enclising_Ditches_Number']==1].copy()
one_ditches['Enclising_Ditches_Number'] = "Yes"
one_ditches_stats = plot_over_grey(one_ditches, 'Enclising_Ditches_Number', \
'Yes', '(1)')
```

Enclosing Ditches Number (1)



Middleton, M. 2024, Hillforts Primer

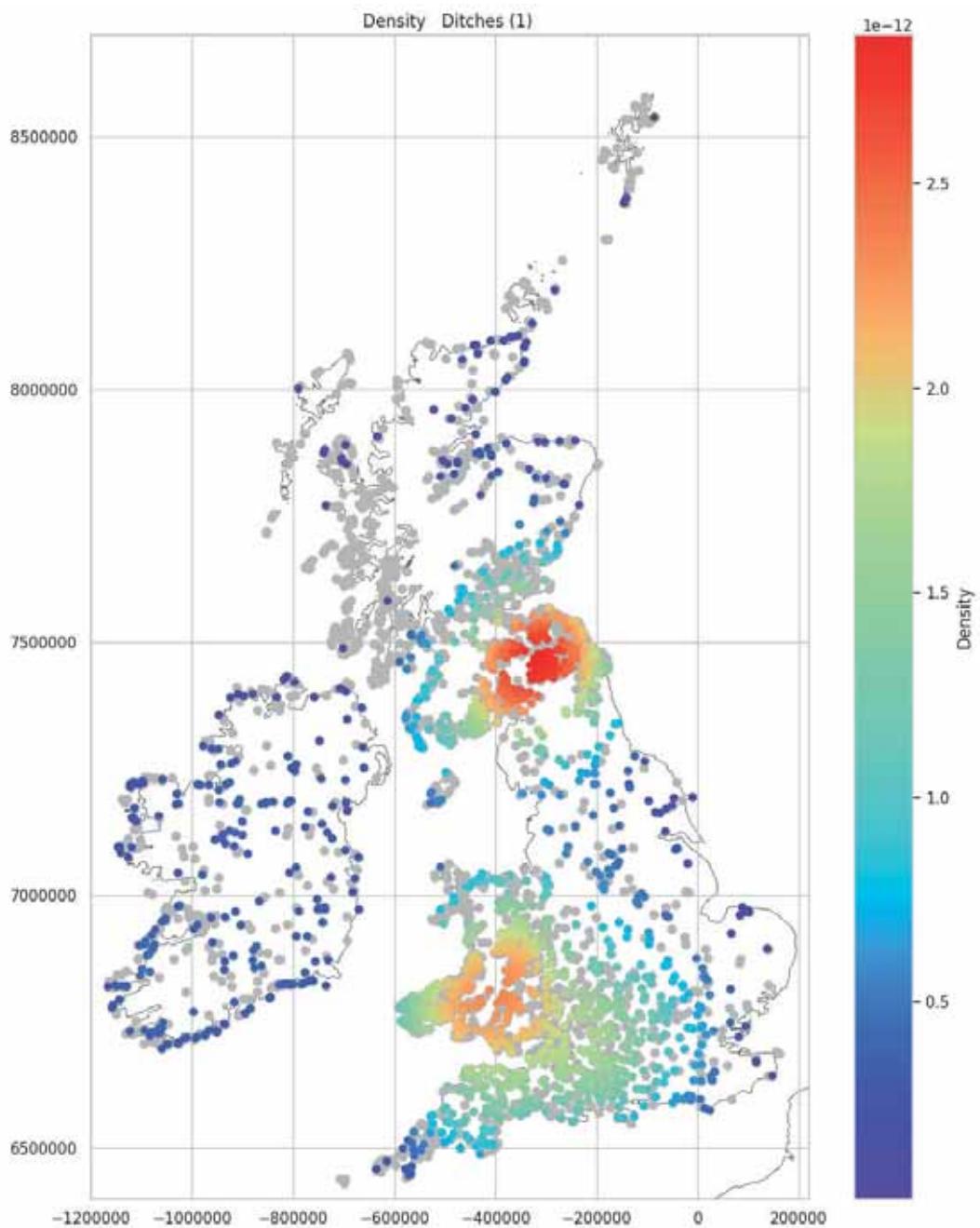
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

40. 54%

Ditches Density Mapped (1)

The density of hillforts with one ditch is split between two clusters. The most intense is over the Northeast with the other focussed over the southern end of the Cambrian Mountains and into south, central England. It is interesting to compare this with [Ramparts Density Mapped \(1\)](#) where the main focus was far more intense over Wales and far less intense over the Northeast.

```
In [ ]: plot_densitiy_over_grey(one_ditches_stats, 'Ditches (1)')
```



Middleton, M. 2024, Hillforts Primer

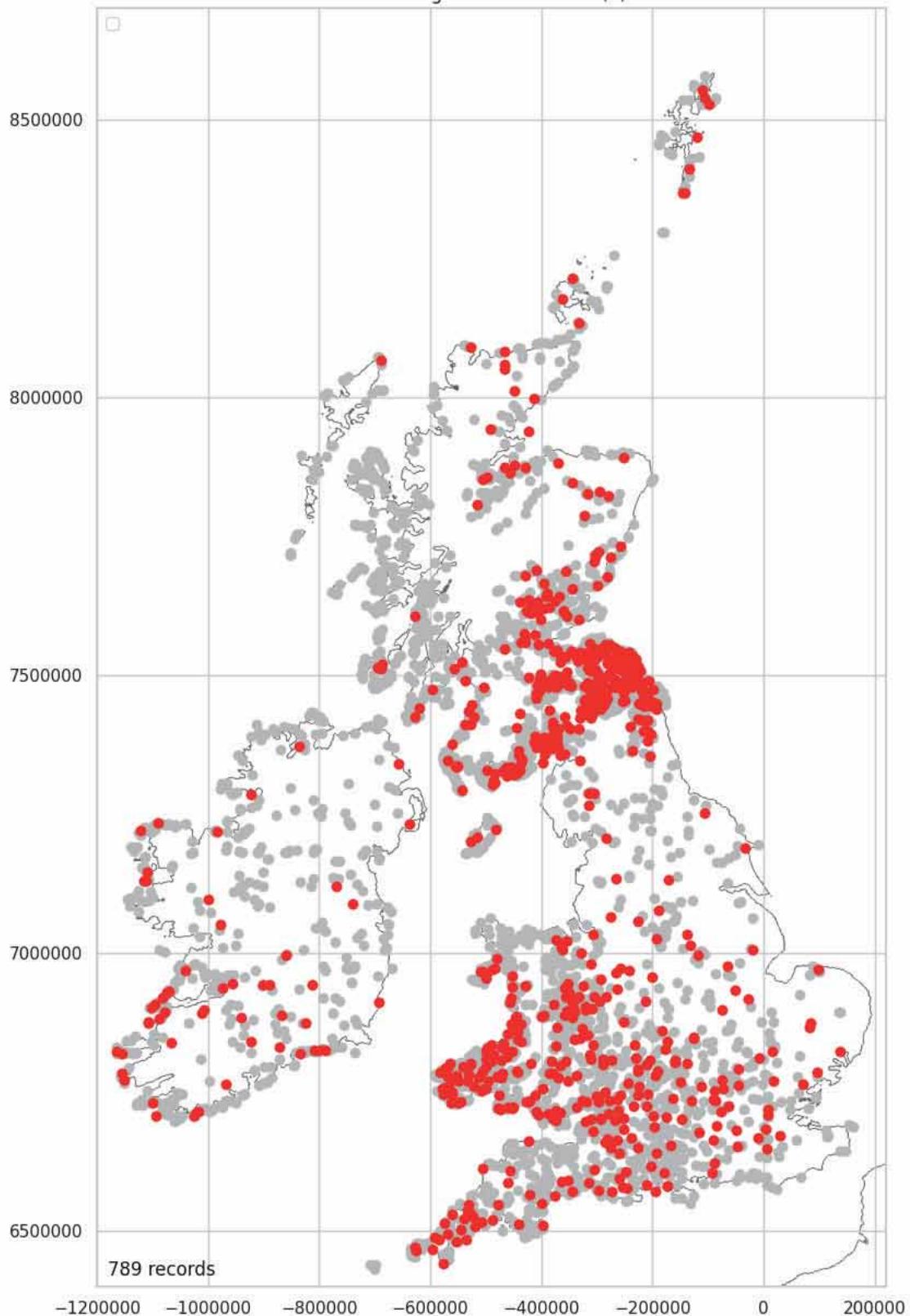
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ditches Mapped (2)

789 (19.03%) of hillforts are recorded as having two ditches. These are mostly distributed over the South, Northeast and southern Ireland. Note [Ditches Mapped \(Not Recorded\)](#).

```
In [ ]: two_ditches = \
location_enclising_data[location_enclising_data['Enclising_Ditches_Number']==2].copy()
two_ditches['Enclising_Ditches_Number'] = "Yes"
two_ditches_stats = plot_over_grey(two_ditches, 'Enclising_Ditches_Number', \
'Yes', '(2)')
```

Enclosing Ditches Number (2)



Middleton, M. 2024, Hillforts Primer

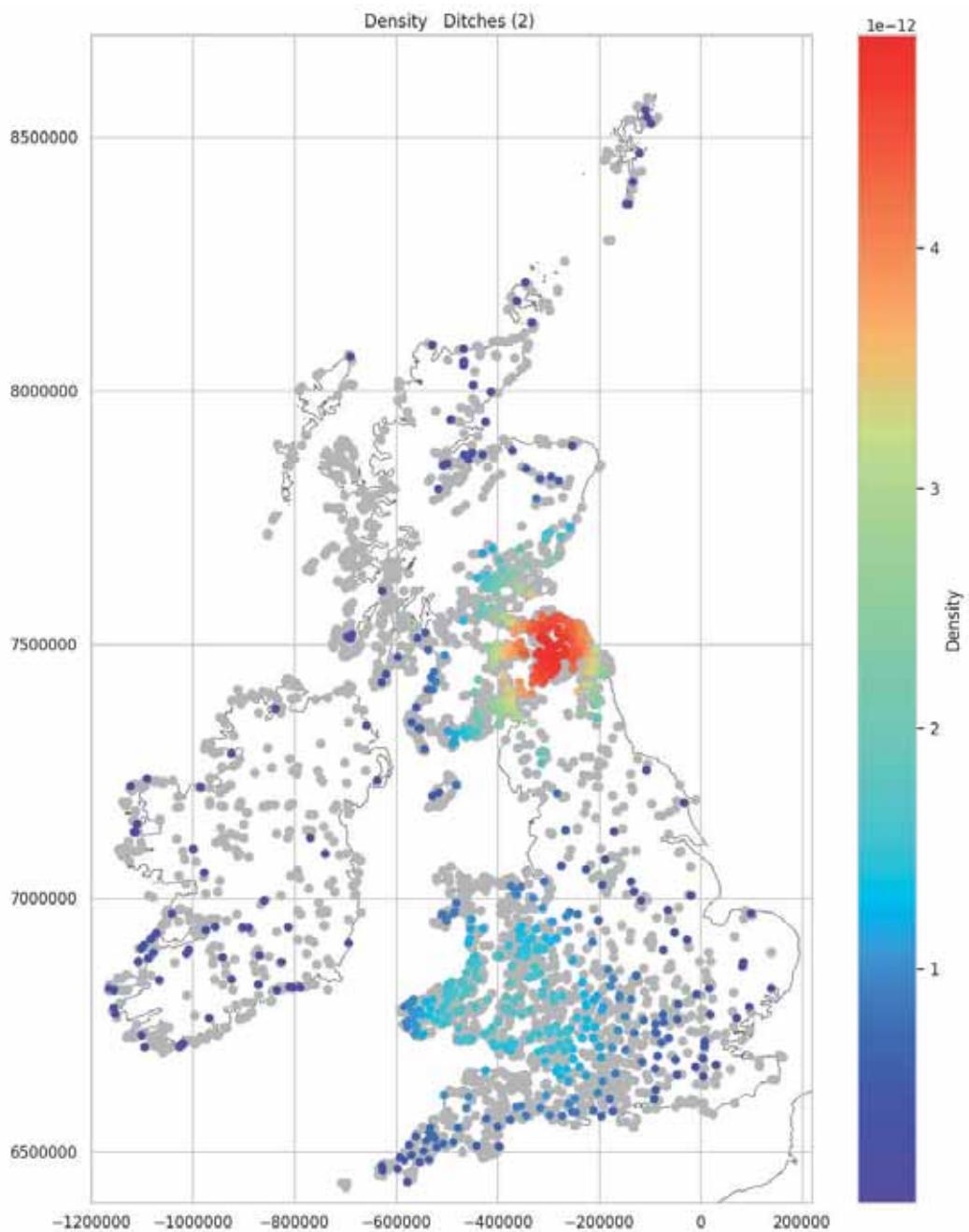
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

19.03%

Ditches Density Mapped (2)

As was seen with ramparts with more than one rampart, the main cluster of forts with more than one ditch is in the Northeast. A secondary cluster can be seen over the southern Cambrian Mountains and there is a peppering of these forts over southern and western Ireland.

```
In [ ]: plot_densiti_over_grey(two_ditches_stats, 'Ditches (2)')
```



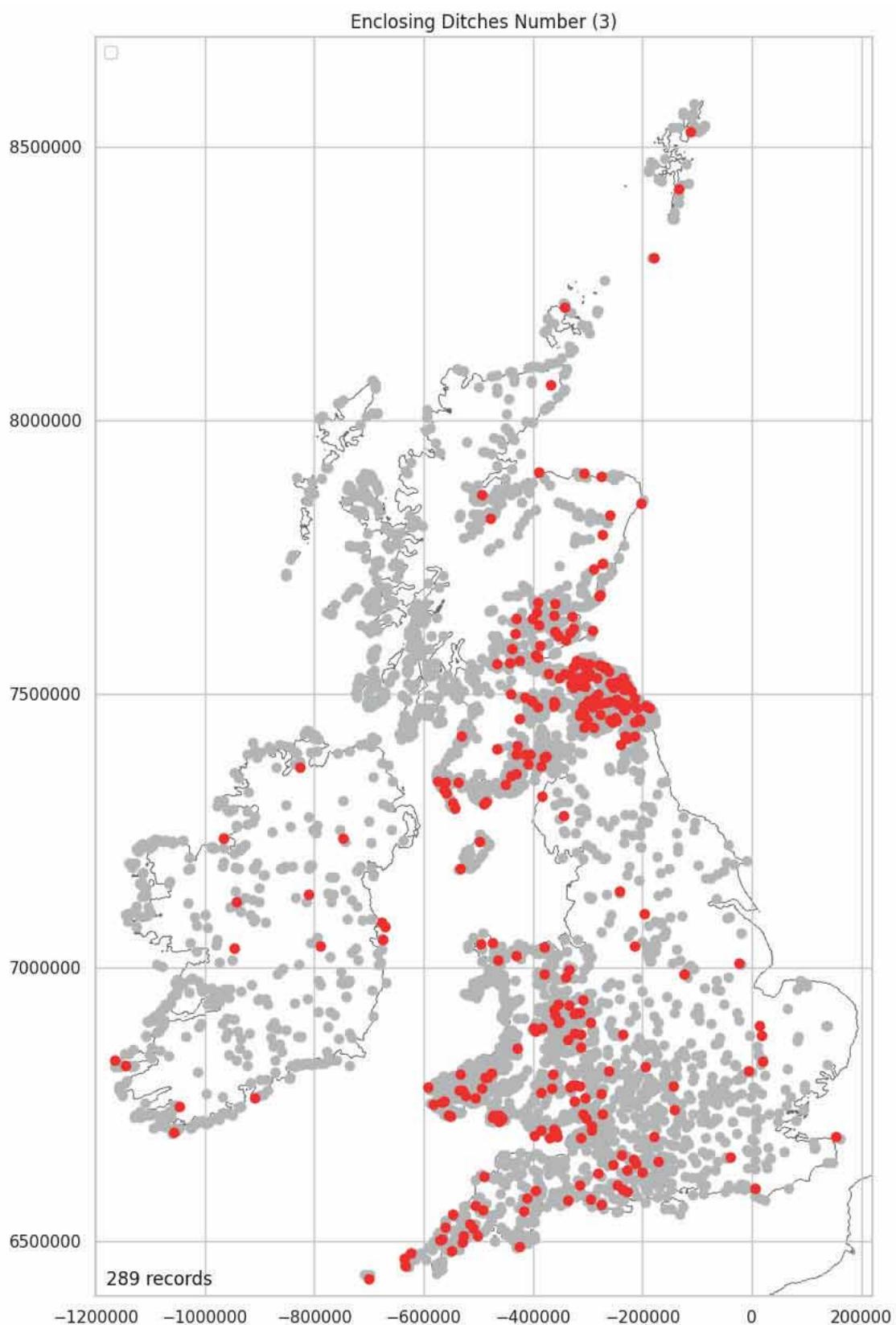
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ditches Mapped (3)

The distribution of hillforts with three ditches is focussed in the Northeast. Note [Ditches Mapped \(Not Recorded\)](#).

```
In [ ]: three_ditches = \
location_enclusing_data[location_enclusing_data['Enclusing_Ditches_Number']==3].copy()
three_ditches['Enclusing_Ditches_Number'] = "Yes"
three_ditches_stats = plot_over_grey(three_ditches, 'Enclusing_Ditches_Number', \
'Yes', '(3)')
```



Middleton, M. 2024, Hillforts Primer

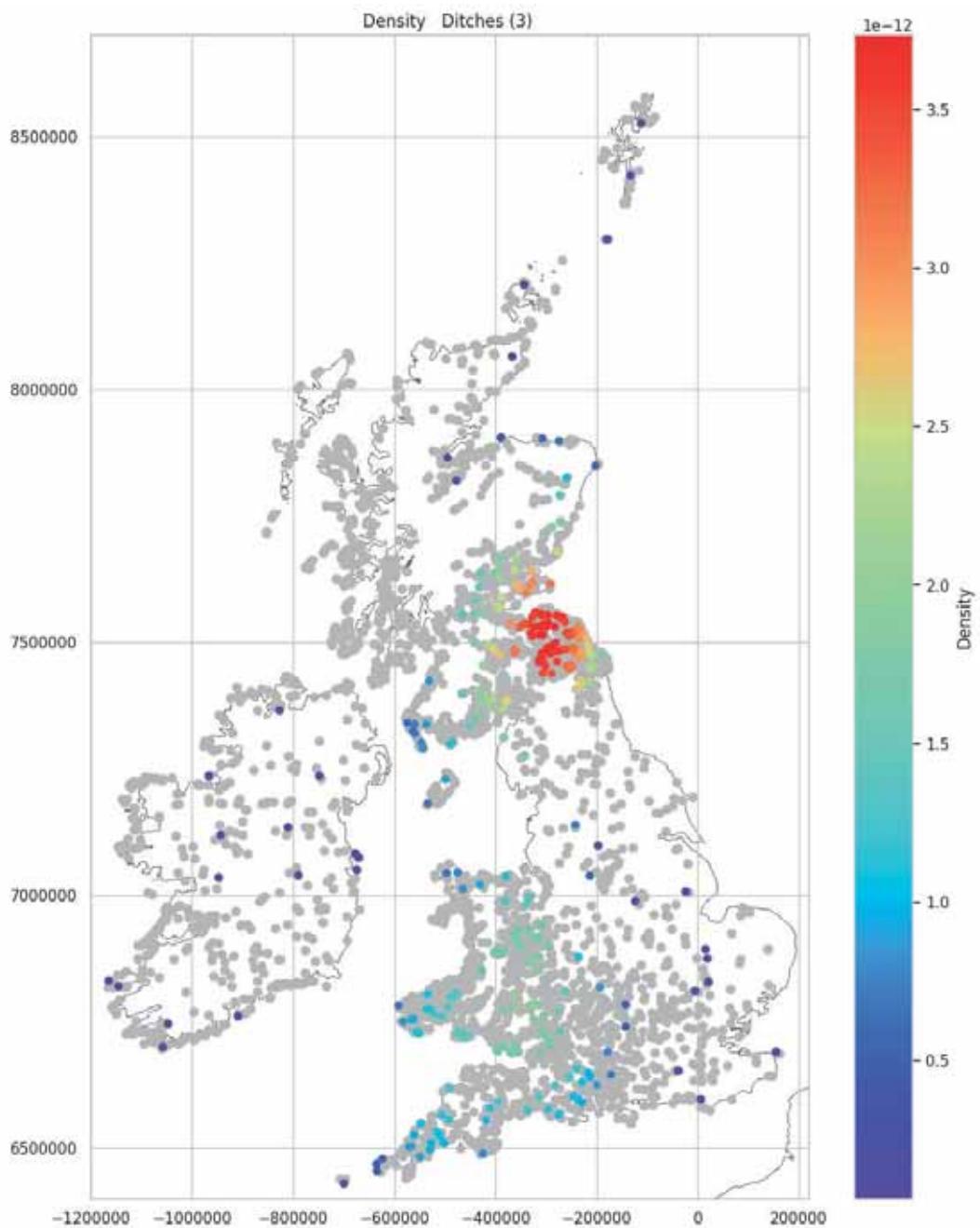
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 97%

Ditches Density Mapped (3)

The main cluster of three ditch hillforts is over the Northeast. A secondary cluster runs down the eastern fringe of the Cambrian Mountains.

```
In [ ]: plot_densitiy_over_grey(three_ditches_stats, 'Ditches (3)')
```



Middleton, M. 2024, Hillforts Primer

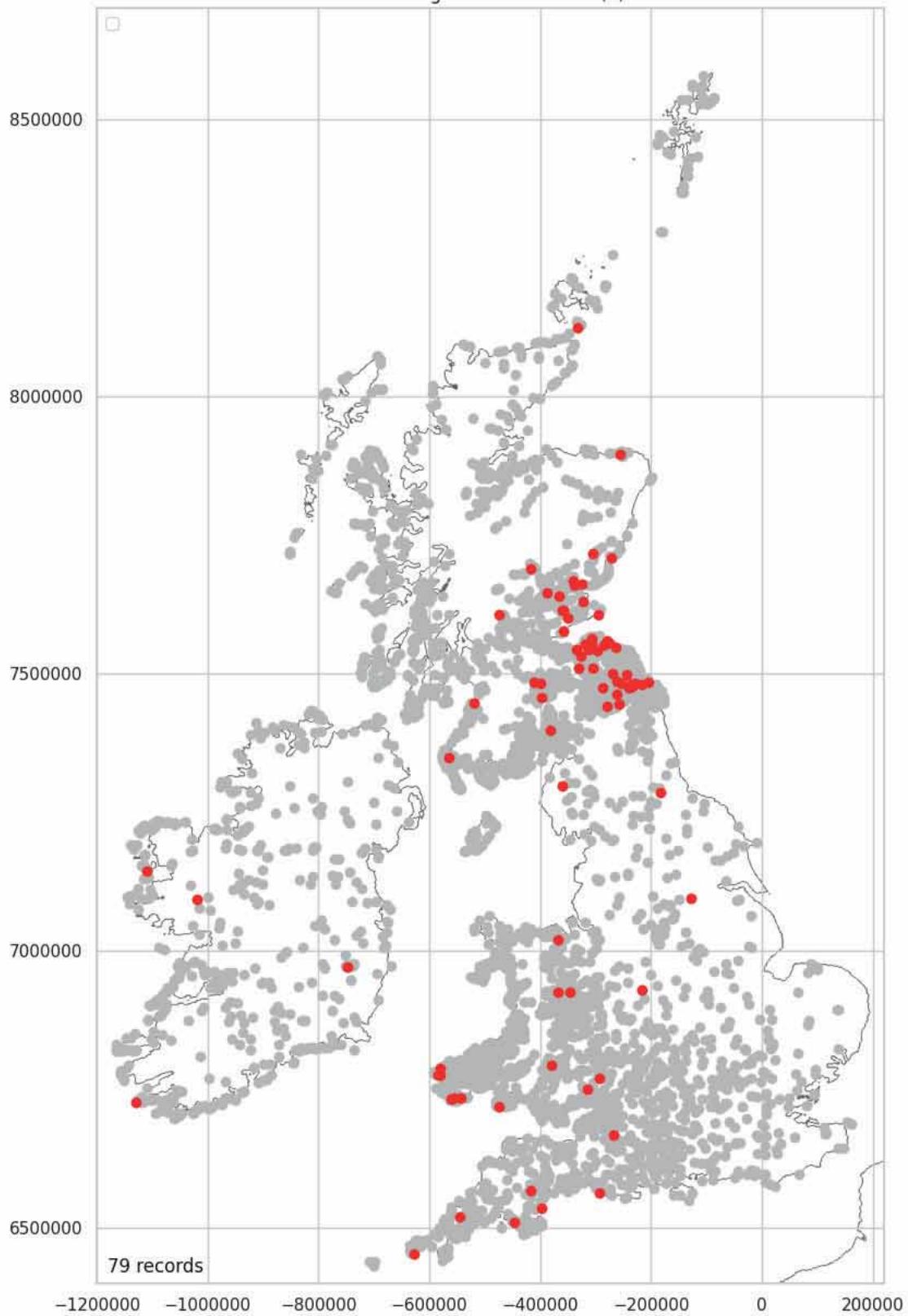
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Ditches Mapped (4)

The focus for four ditch hillforts is in the Northeast over East Lothian. This is in line with what was seen for ramparts. See [Ramparts Mapped \(4\)](#). Note [Ditches Mapped \(Not Recorded\)](#).

```
In [ ]: four_ditches = \
location_enclosing_data[location_enclosing_data['Enclosing_Ditches_Number']==4].copy()
four_ditches['Enclosing_Ditches_Number'] = "Yes"
four_ditches_stats = plot_over_grey(four_ditches, 'Enclosing_Ditches_Number', \
'Yes', '(4)')
```

Enclosing Ditches Number (4)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

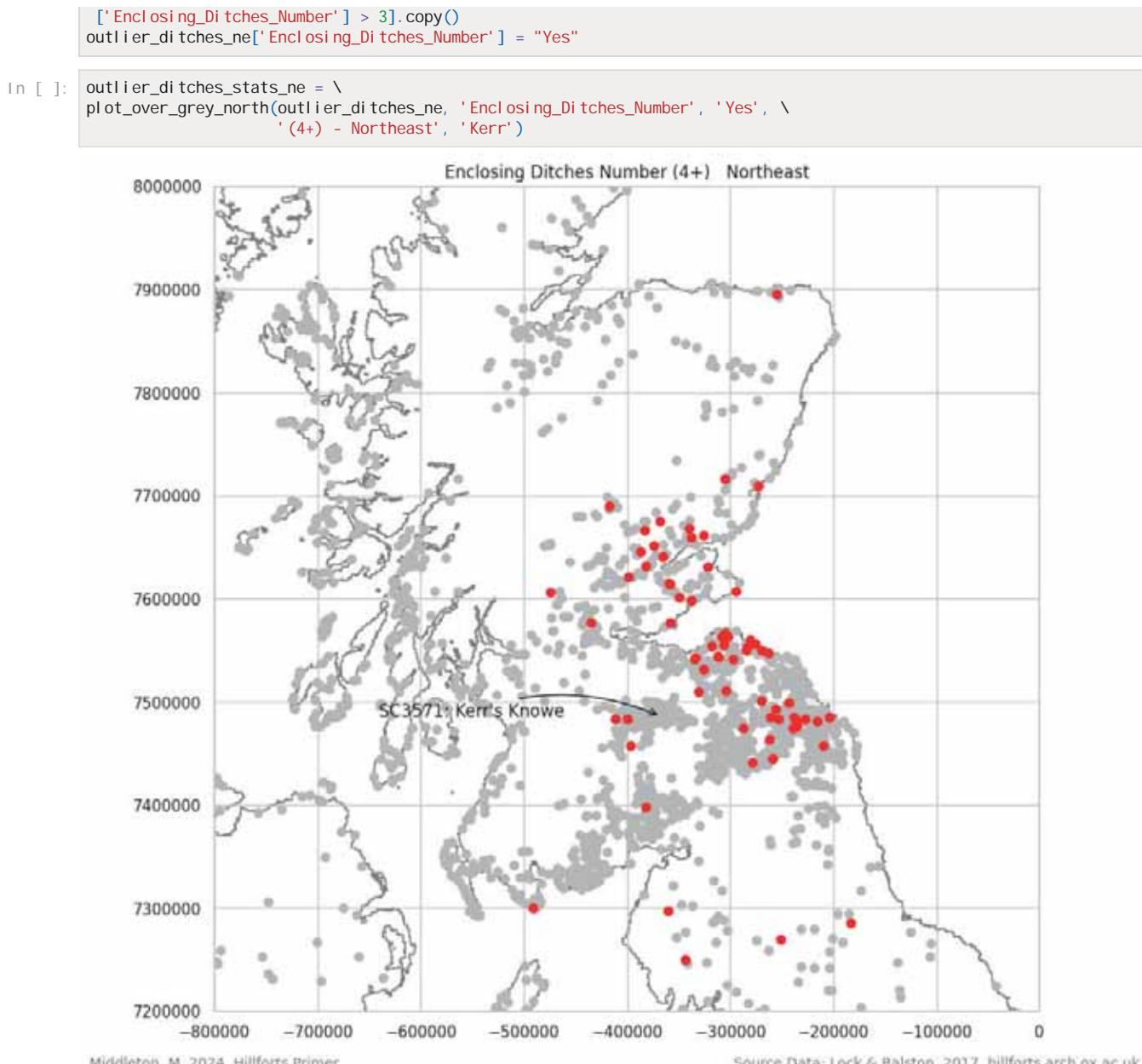
1. 9%

Ditches Mapped (4+ NE)

The concentration of hillforts, in the Northeast, with four or more ditches has a similar cluster to [Ramparts Mapped \(4+ NE\)](#), running along the eastern fringe of the Southern Uplands, up and across Fife and on into Perthshire and Angus. It does not include the cluster seen in the ramparts data around Kerr's Hill and neither Traprain Law or Eildon Hill North have four or more ditches.

```
In [ ]: outlier_ditches_ne = \
location_enclosing_data_ne[location_enclosing_data_ne <
```

529

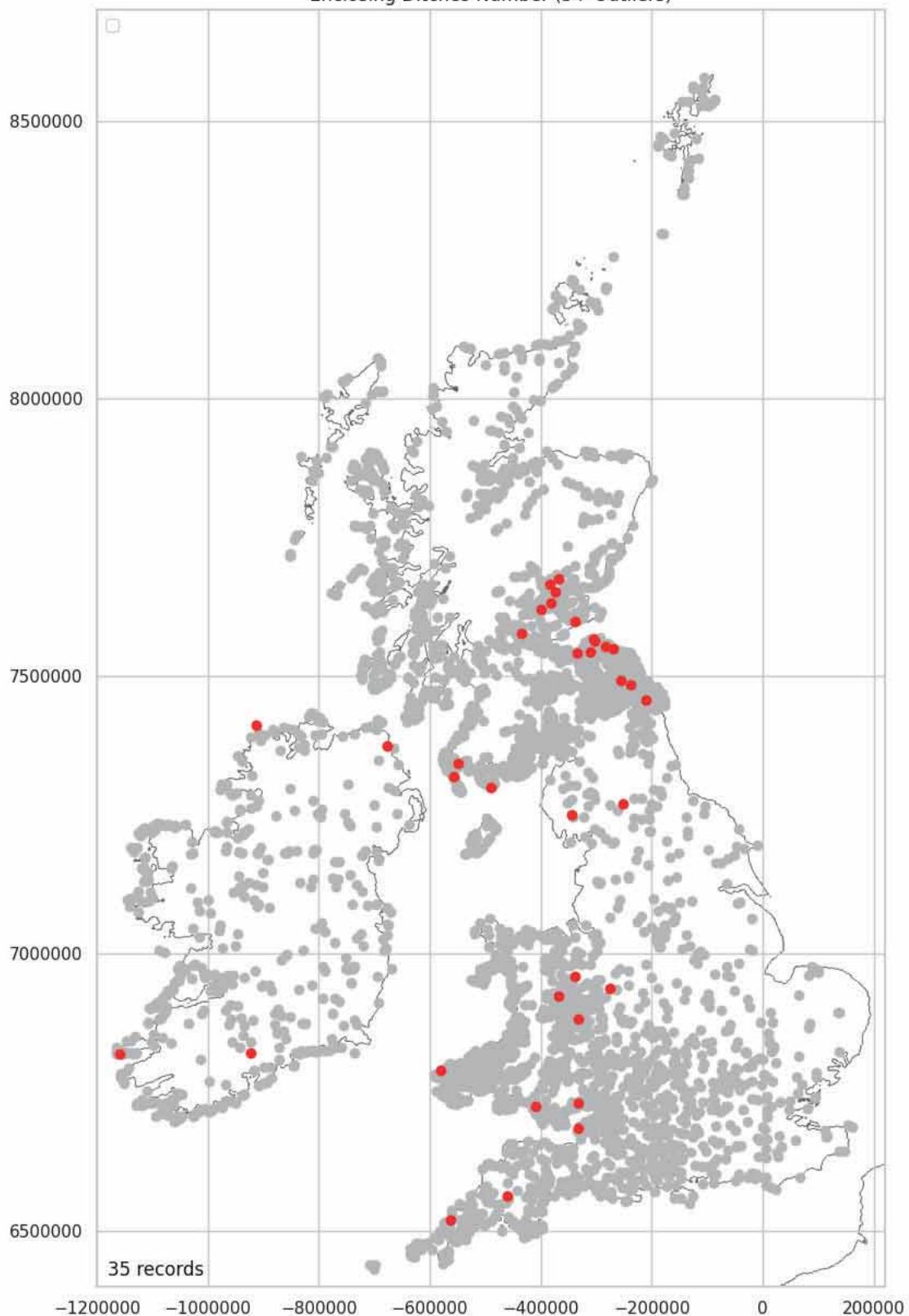


Ditches Mapped (5+ Outliers)

There are 35 hillforts with five or more ditches. Note [Ditches Mapped \(Not Recorded\)](#).

```
In [ ]: outlier_ditches = \
location_enclaving_data[ \
[location_enclaving_data['Enclosing_Ditches_Number'] > 4].copy()]
outlier_ditches['Enclosing_Ditches_Number'] = "Yes"
outlier_ditches_stats = plot_over_grey(outlier_ditches, \
'Enclosing_Ditches_Number', 'Yes', \
'(5+ Outliers)')
```

Enclosing Ditches Number (5+ Outliers)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.84%

```
In [ ]: most_ditches = \
    location_enclosing_data[location_enclosing_data['Enclosing_Ditches_Number'] == 8].copy()
most_ditches = \
    pd.merge(name_and_number, most_ditches, left_index=True, right_index=True)
most_ditches[['Main_Area_Number', 'Main_Display_Name', 'Enclosing_Area_1', \
    'Enclosing_Max_Ramparts', 'Enclosing_Ditches_Number']]
```

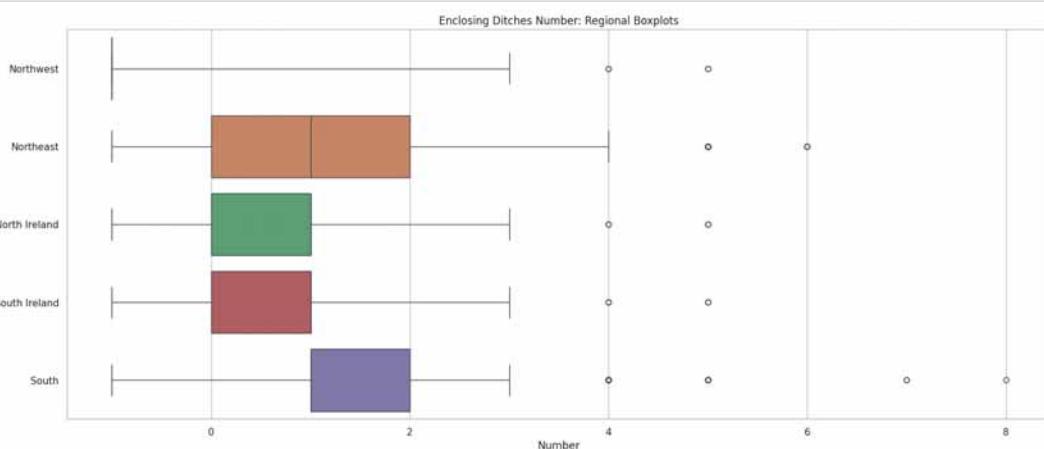
Out[]:	Main_Atlas_Number	Main_Display_Name	Enclosing_Area_1	Enclosing_Max_Ramparts	Enclosing_Ditches_Number
634	656	Trevelgue Head, Cornwall		3.2	8.0

Ditches by Region

Hillforts in Ireland are most likely to have zero or one ditch; In the Northeast, zero to two ditches and in the South, one to two ditches. It is not possible to say anything about the Northwest because of the survey bias seen in [Ditches Mapped \(Not Recorded\)](#).

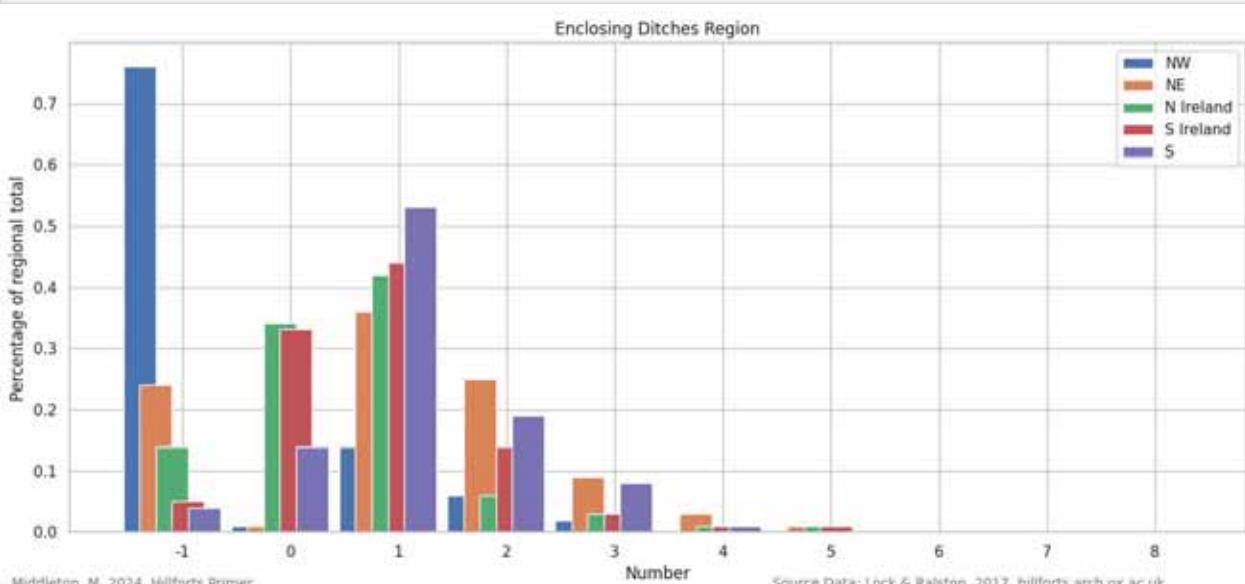
```
In [ ]: regional_dict = \
{'Northwest': location_enclosing_data_nw['Enclosing_Ditches_Number'], \
'Northeast': location_enclosing_data_ne['Enclosing_Ditches_Number'], \
'North Ireland': location_enclosing_data_ireland_n['Enclosing_Ditches_Number'], \
'South Ireland': location_enclosing_data_ireland_s['Enclosing_Ditches_Number'], \
'South': location_enclosing_data_south['Enclosing_Ditches_Number']}

plot_data = pd.DataFrame.from_dict(regional_dict)
plt.figure(figsize=(20,8))
ax = sns.boxplot(data=plot_data, orient="h", whis=[2.2, 97.8], showfliers=True)
add_annotation_plot(ax)
ax.set_xlabel('Number')
title = 'Enclosing_Ditches_Number: Regional Boxplots'
plt.title(get_print_title(title))
save_fig(title)
plt.show()
```



Overall, hillforts are most likely to have a single ditch. The proportions are roughly similar across all areas apart from Ireland, where forts are more likely to have no ditch. The large number of hillforts where ditches have not been recorded (-1) shows the data from the Northwest, Northeast and North Ireland is particularly susceptible to the survey bias noted in [Ditches Mapped \(Not Recorded\)](#).

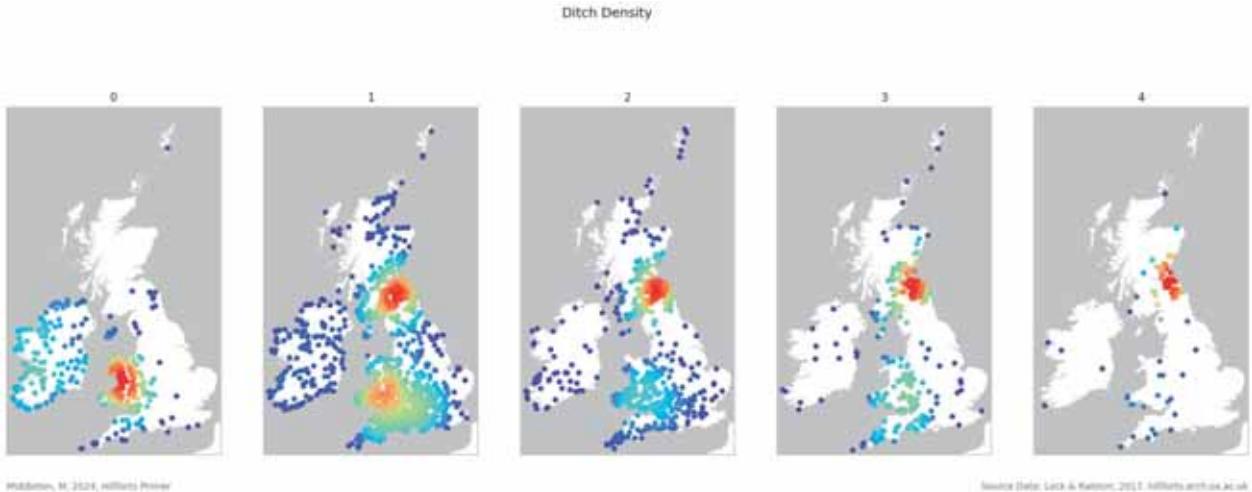
```
In [ ]: plot_feature_by_region(location_enclosing_data_nw,
location_enclosing_data_ne,
location_enclosing_data_ireland_n,
location_enclosing_data_ireland_s,
location_enclosing_data_south,
'Enclosing_Ditches_Number',
'Enclosing_Ditches_Region', 10)
```



Ditches Summary

The focus for hillforts without ditches is the upland areas of Wales. There is a smaller concentration of these forts in Ireland. It is important to note the bias in recording ditches seen in [Ditches Mapped \(Not Recorded\)](#). An absence of recording may indicate an absence of ditches in this area, and this may suggest there is a third cluster in the Northwest. Work needs to be done to confirm this either way. Hillforts with a single ditch cluster into two groups. One in the Southern Uplands and the second over the southern Cambrian Mountains and into south, central England. Hillforts with two or more ditches tend to cluster to the east of the Southern Uplands although there are also small clusters of these in Wales.

```
In [ ]: plot_densiti_over_grey_figure(zero_ditches, one_ditches, two_ditches, \
three_ditches, four_ditches, 'Ditch Density')
```



Quadrant Data

The commentary for the quadrant data will be summarised at the top of each class. Individual commentary will not be provided for each orientation.

Quadrant Data Mapped (Not Recorded)

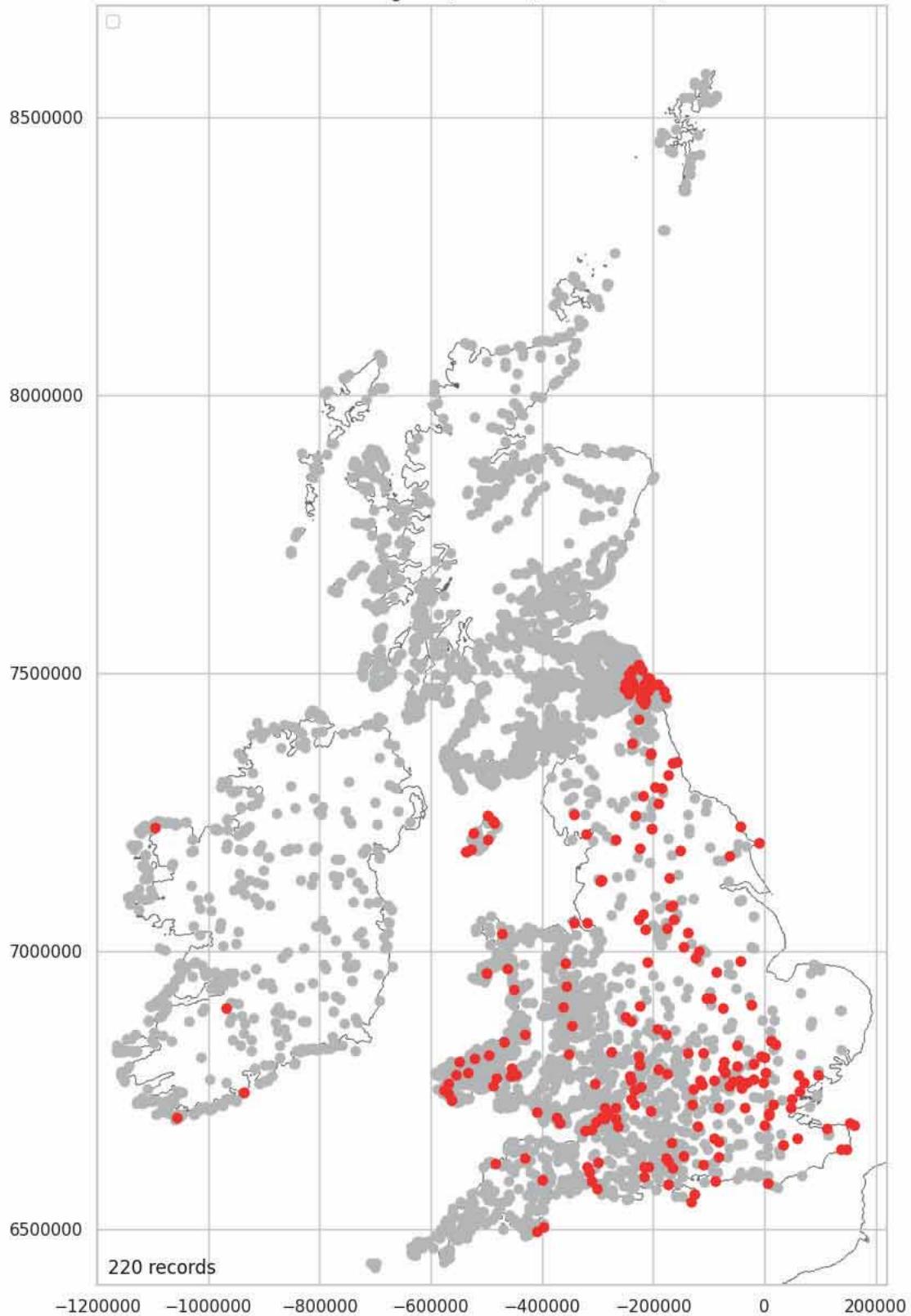
Only between 220 (NE) to 251 (SW) hillforts have not had quadrant data recorded. Almost all of these are in England and Wales.

NE Quadrant Data Mapped (Not Recorded)

```
In [ ]: all_ramparts = \
location_enclising_data[location_enclising_data['Enclising_Max_Ramparts'] >-1]
all_ditches = \
location_enclising_data[location_enclising_data['Enclising_Ditches_Number'] >-1]
ne_quadrant_data = \
location_enclising_data[location_enclising_data['Enclising_NE_Quadrant'] >-1]
se_quadrant_data = \
location_enclising_data[location_enclising_data['Enclising_SE_Quadrant'] >-1]
sw_quadrant_data = \
location_enclising_data[location_enclising_data['Enclising_SW_Quadrant'] >-1]
nw_quadrant_data = \
location_enclising_data[location_enclising_data['Enclising_NW_Quadrant'] >-1]
```

```
In [ ]: nan_ne = \
location_enclising_data\
[location_enclising_data['Enclising_NE_Quadrant'] == -1].copy()
nan_ne['Enclising_NE_Quadrant'] = "Yes"
nan_ne_stats = plot_over_grey(nan_ne, 'Enclising_NE_Quadrant', \
'Yes', '(Not Recorded)')
```

Enclosing NE Quadrant (Not Recorded)



Middleton, M. 2024, Hillforts Primer

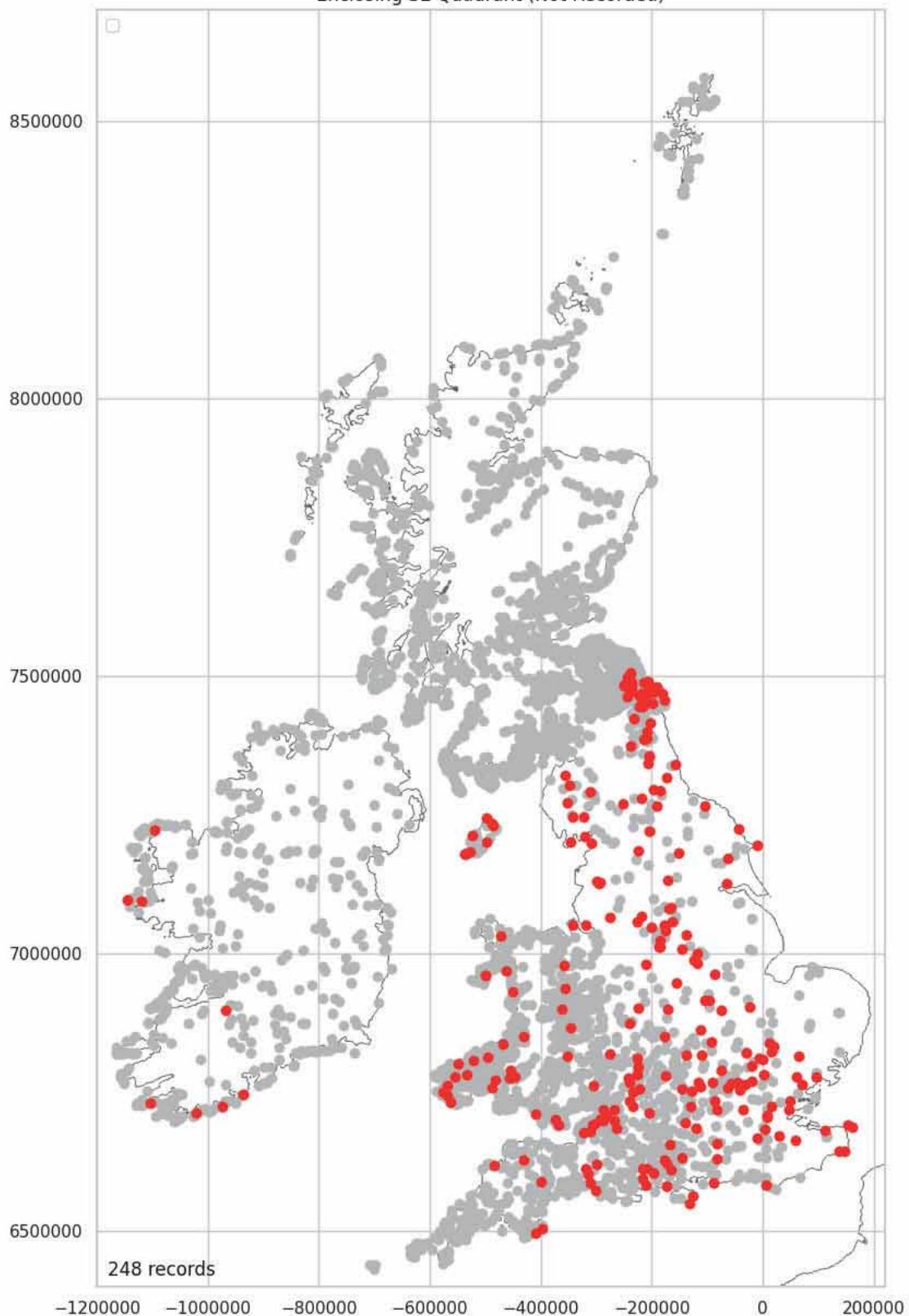
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5.31%

SE Quadrant Data Mapped (Not Recorded)

```
In [ ]: nan_se = \
location_enclising_data[location_enclising_data['Enclising_SE_Quadrant']==1].copy()
nan_se['Enclising_SE_Quadrant'] = "Yes"
nan_se_stats = plot_over_grey(nan_se, 'Enclising_SE_Quadrant', 'Yes', \
'Not Recorded')
```

Enclosing SE Quadrant (Not Recorded)



Middleton, M. 2024, Hillforts Primer

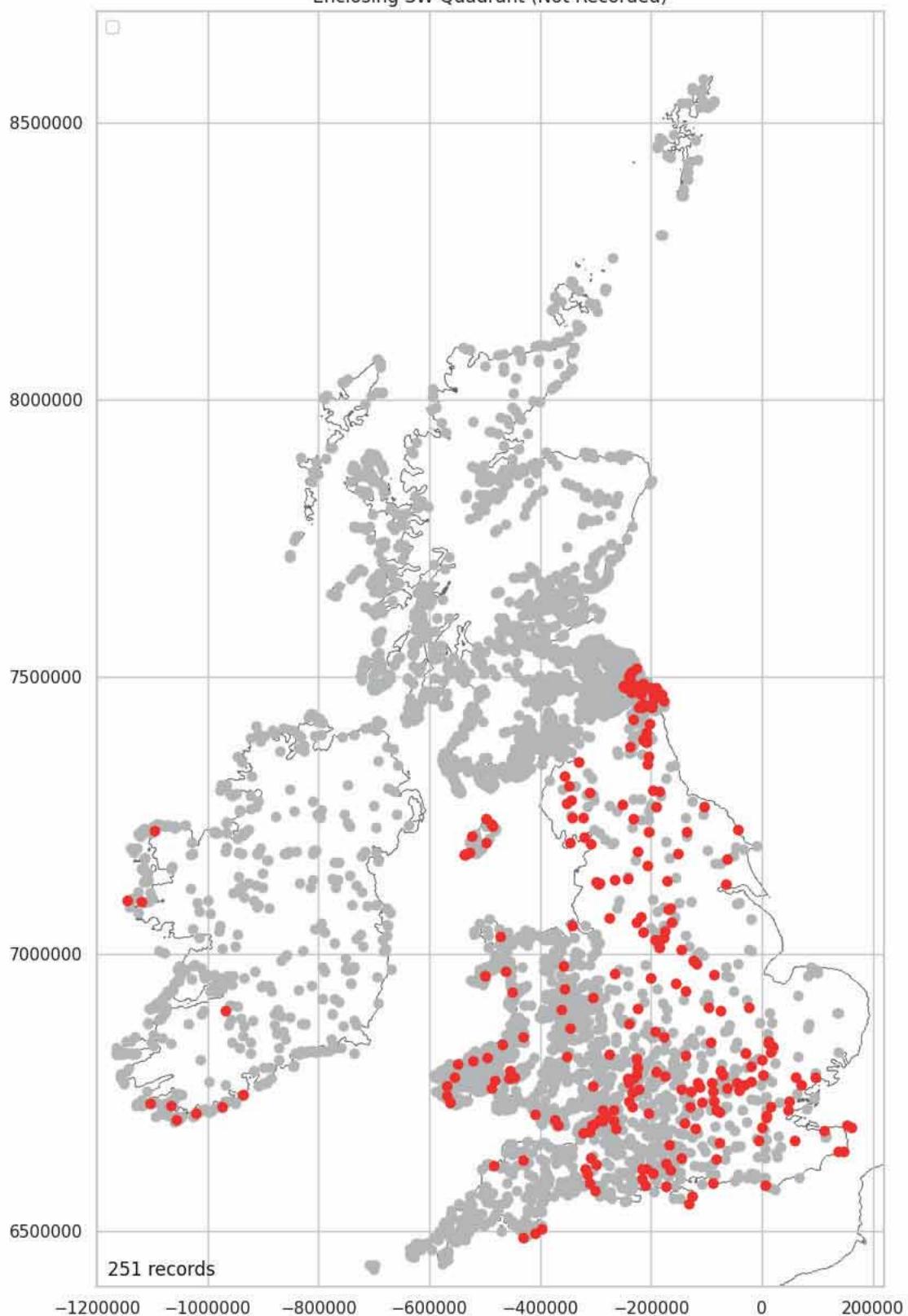
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5. 98%

SW Quadrant Data Mapped (Not Recorded)

```
In [ ]: nan_sw = \
location_enclising_data[location_enclising_data['Enclosing_SW_Quadrant']==1].copy()
nan_sw['Enclosing_SW_Quadrant'] = "Yes"
nan_sw_stats = plot_over_grey(nan_sw, 'Enclosing_SW_Quadrant', 'Yes', \
'Not Recorded')
```

Enclosing SW Quadrant (Not Recorded)



Middleton, M. 2024, Hillforts Primer

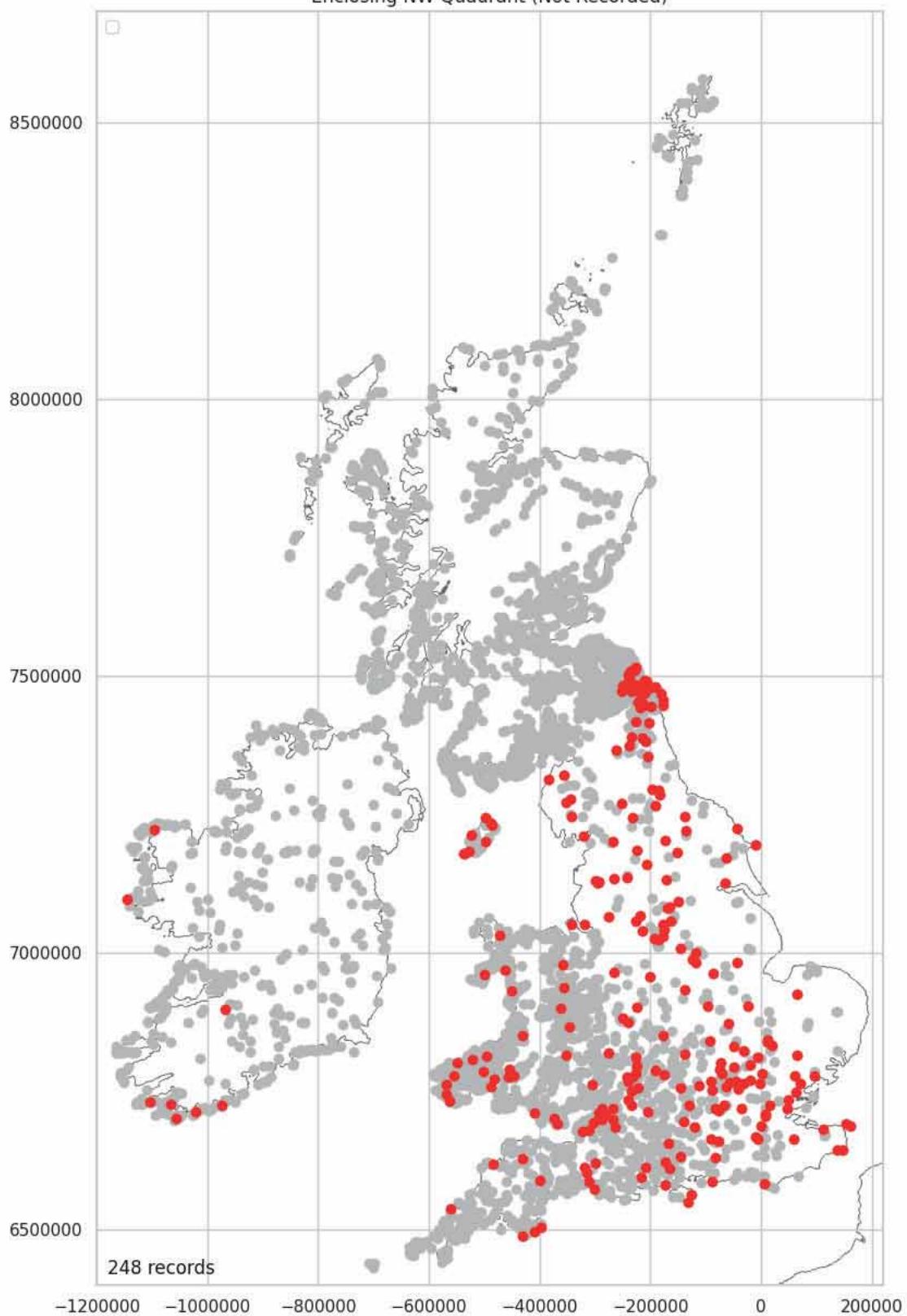
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6.05%

NW Quadrant Data Mapped (Not Recorded)

```
In [ ]: nan_nw = \
    location_enclosing_data\
    [location_enclosing_data['Enclosing_NW_Quadrant']==-1].copy()
nan_nw['Enclosing_NW_Quadrant'] = "Yes"
nan_nw_stats = plot_over_grey(nan_nw, 'Enclosing_NW_Quadrant', 'Yes', \
    '(Not Recorded)')
```

Enclosing NW Quadrant (Not Recorded)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

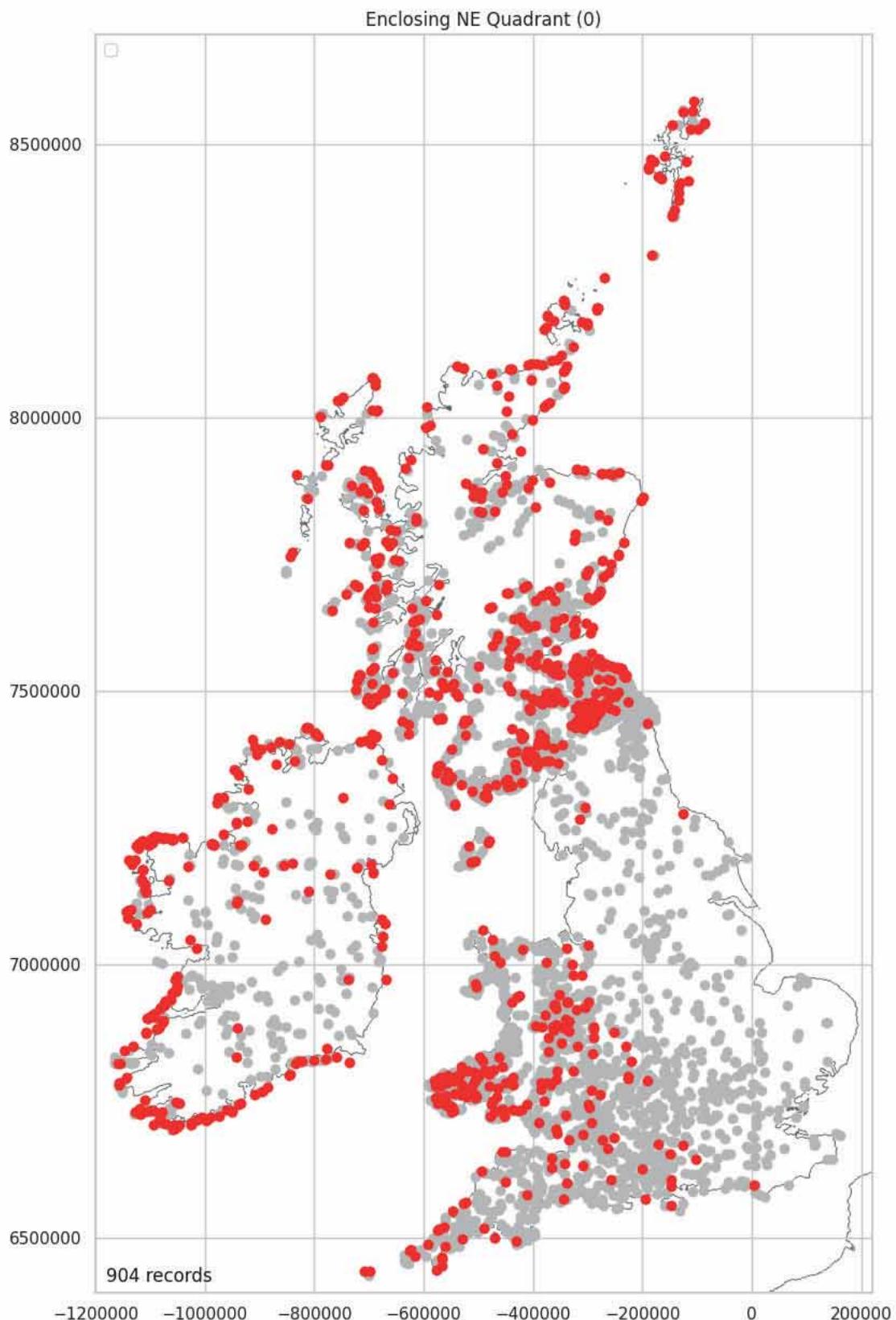
5. 98%

Quadrant Data Mapped (0)

Where there are no ramparts, these show an influence from the local topography. Irish coastal forts on the west coast show a cluster of forts with no ramparts facing southwest and northwest - toward the sea. Similarly, forts on the Pembrokeshire peninsula show more intense clusters of forts with no ramparts to the southwest and northwest. Again, facing the sea. As this data is most likely to reflect the macro topographic situation of each fort, large scale regional analysis of this data is likely to provide limited insight. Commentary over the remainder of this section will be brief.

NE Quadrant Data Mapped (0)

```
In [ ]: zero_ne = ne_quadrant_data[ne_quadrant_data['Encl osing_NE_Quadrant'] == 0].copy()
zero_ne['Encl osing_NE_Quadrant'] = "Yes"
zero_ne_stats = plot_over_grey(zero_ne, 'Encl osing_NE_Quadrant', 'Yes', '(0)')
```



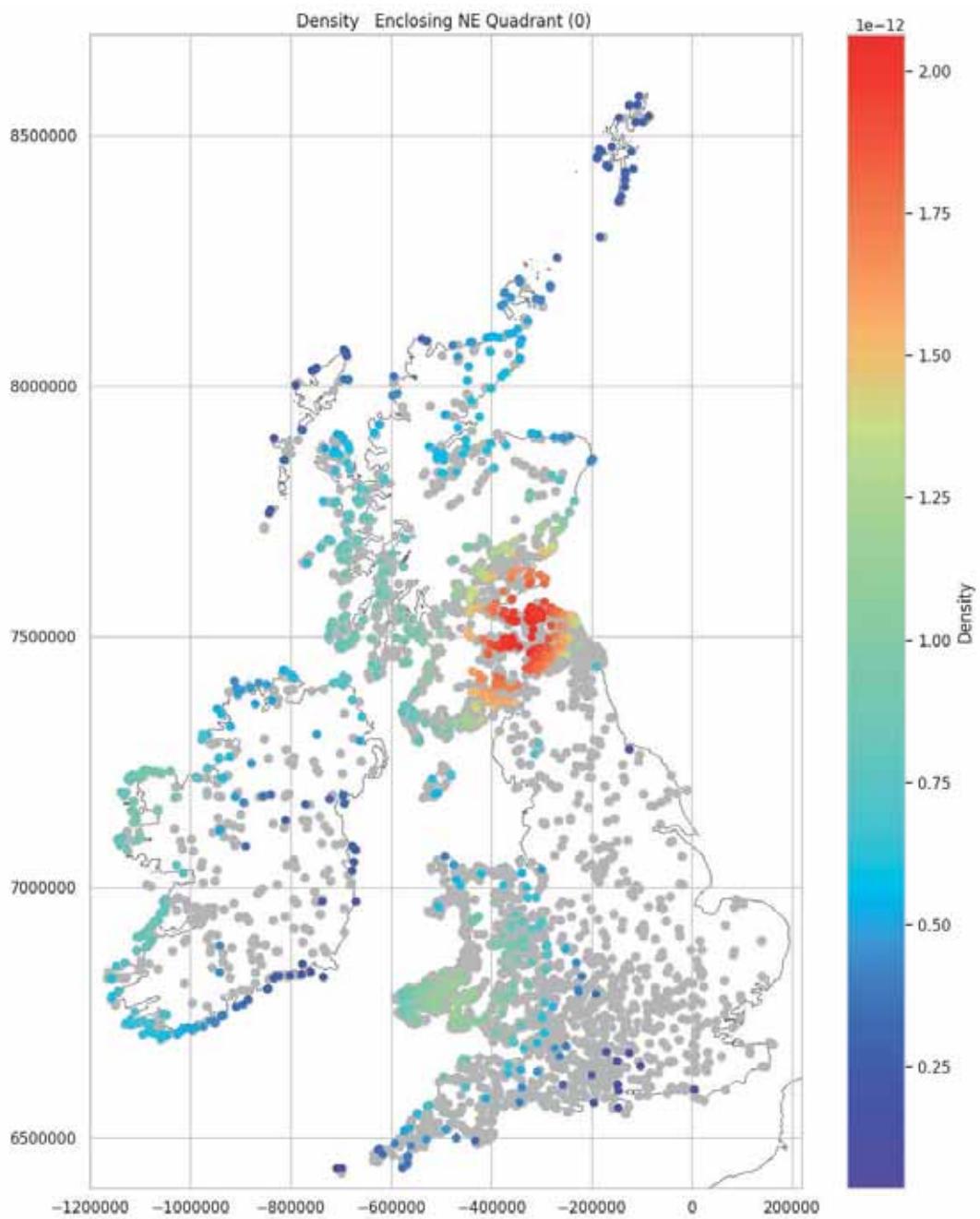
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

21.8%

NE Quadrant Data Density Mapped (0)

```
In [ ]: plot_densitiy_over_grey(zero_ne_stats, 'Encl osing_NE_Quadrant (0)')
```



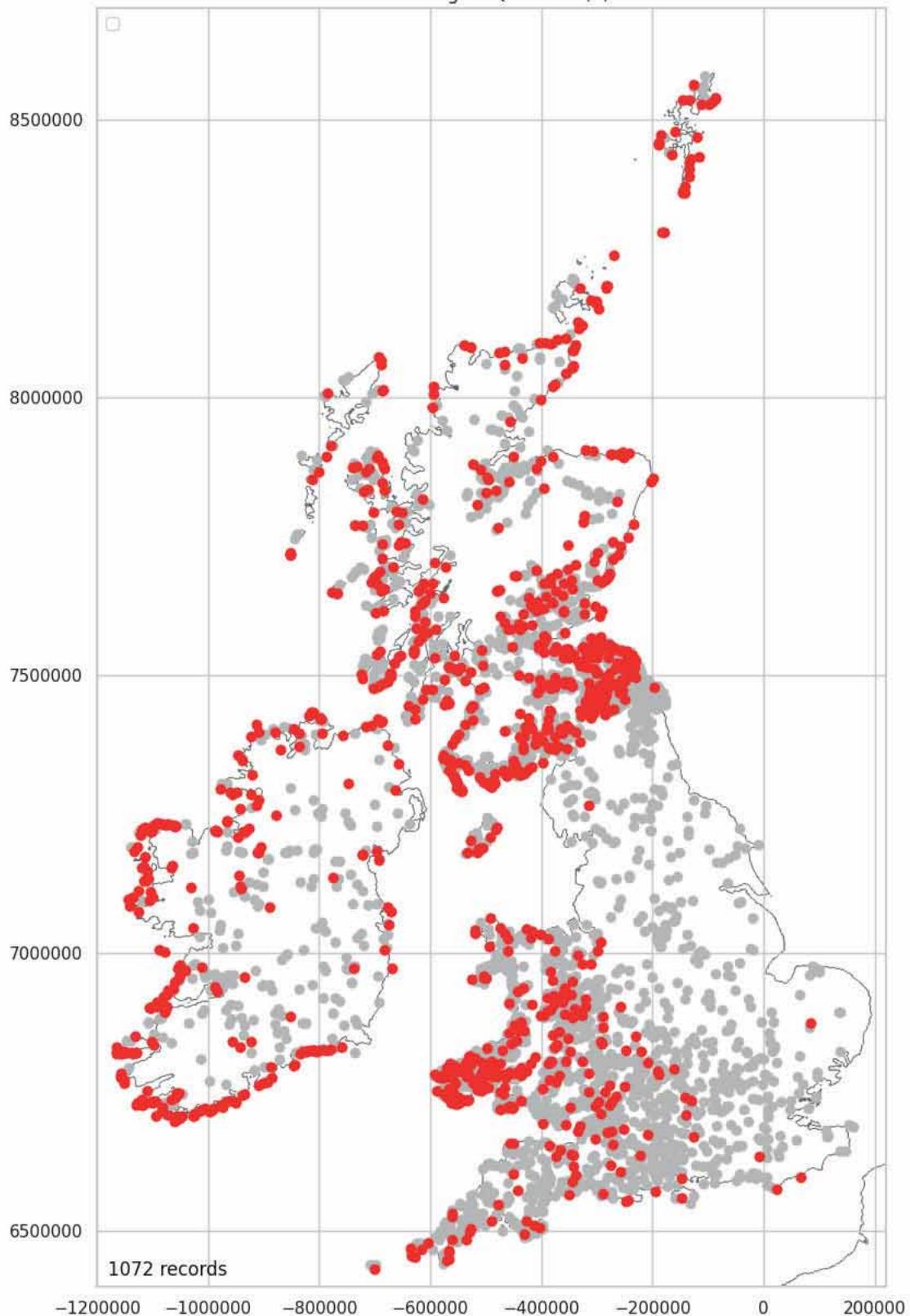
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SE Quadrant Data Mapped (0)

```
In [ ]: zero_se = se_quadrant_data[se_quadrant_data['Encl osing_SE_Quadrant']==0].copy()
zero_se['Encl osing_SE_Quadrant'] = "Yes"
zero_se_stats = plot_over_grey(zero_se, 'Encl osing_SE_Quadrant', 'Yes', '(0)')
```

Enclosing SE Quadrant (0)



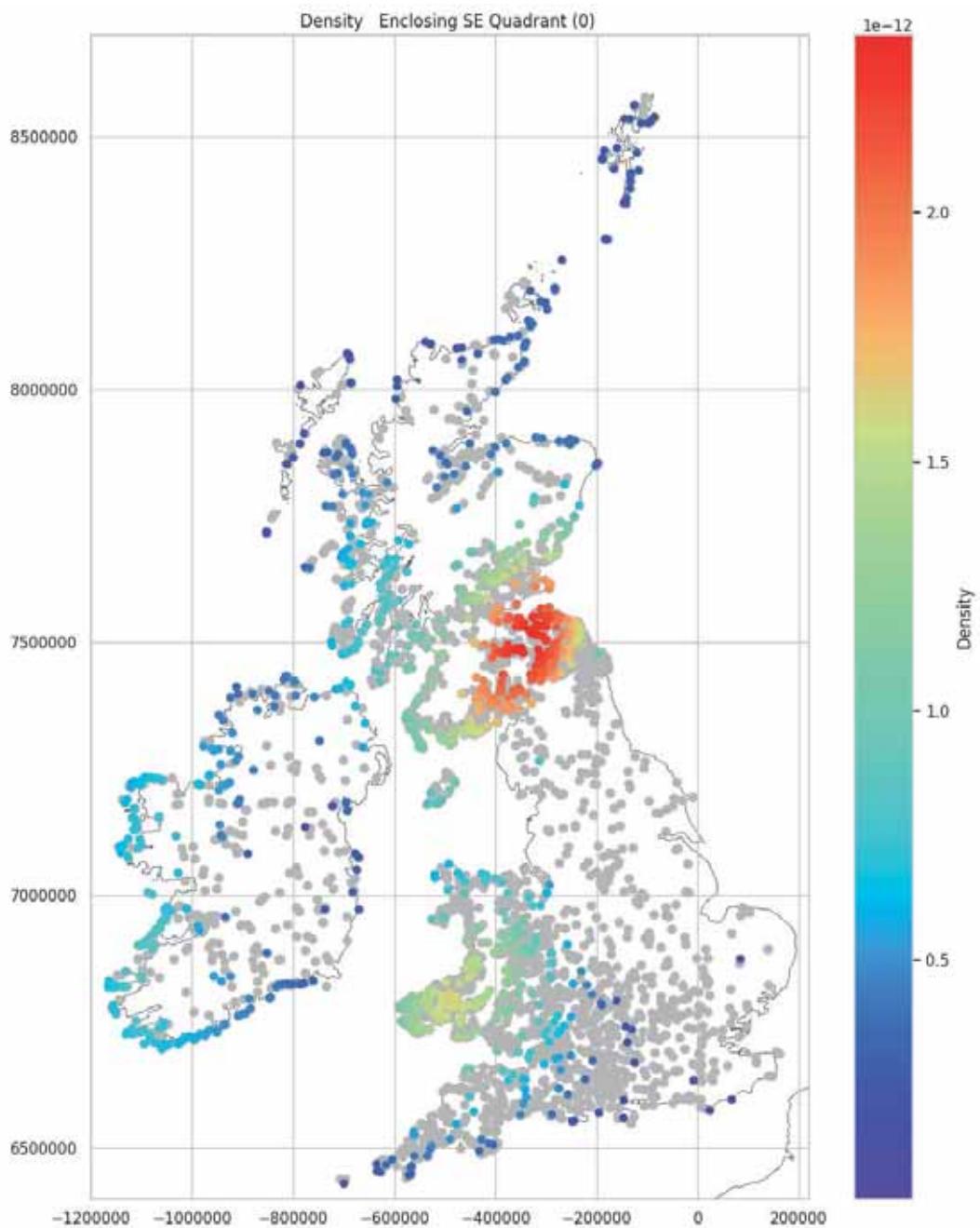
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

25. 85%

NE Quadrant Data Density Mapped (0)

```
In [ ]: plot_density_over_grey(zero_se_stats, 'Enclosing_SE_Quadrant (0)')
```



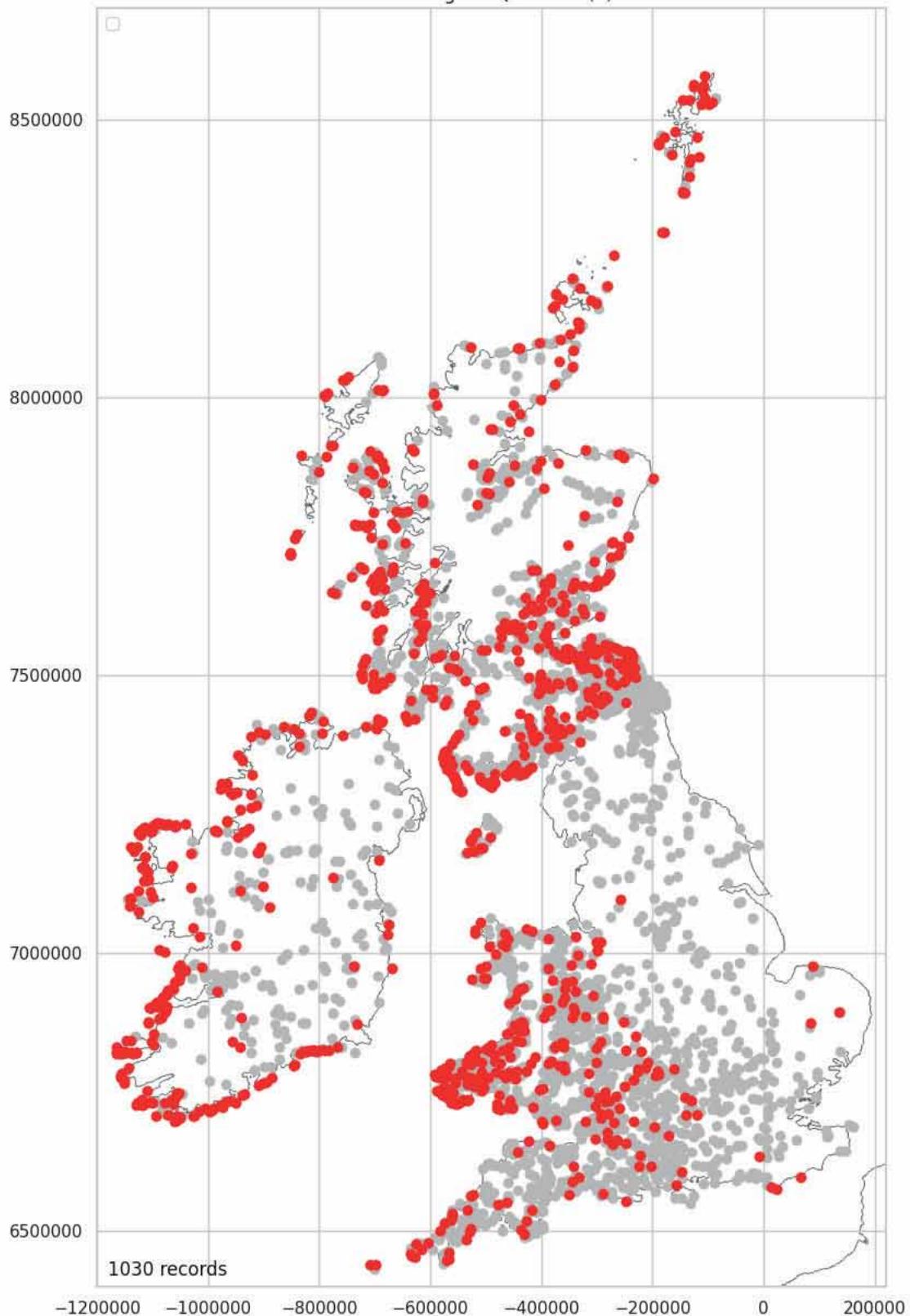
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SW Quadrant Data Mapped (0)

```
In [ ]: zero_sw = sw_quadrant_data[sw_quadrant_data['Encl osi ng_SW_Quadrant']==0].copy()
zero_sw['Encl osi ng_SW_Quadrant'] = "Yes"
zero_sw_stats = plot_over_grey(zero_sw, 'Encl osi ng_SW_Quadrant', 'Yes', '(0)')
```

Enclosing SW Quadrant (0)



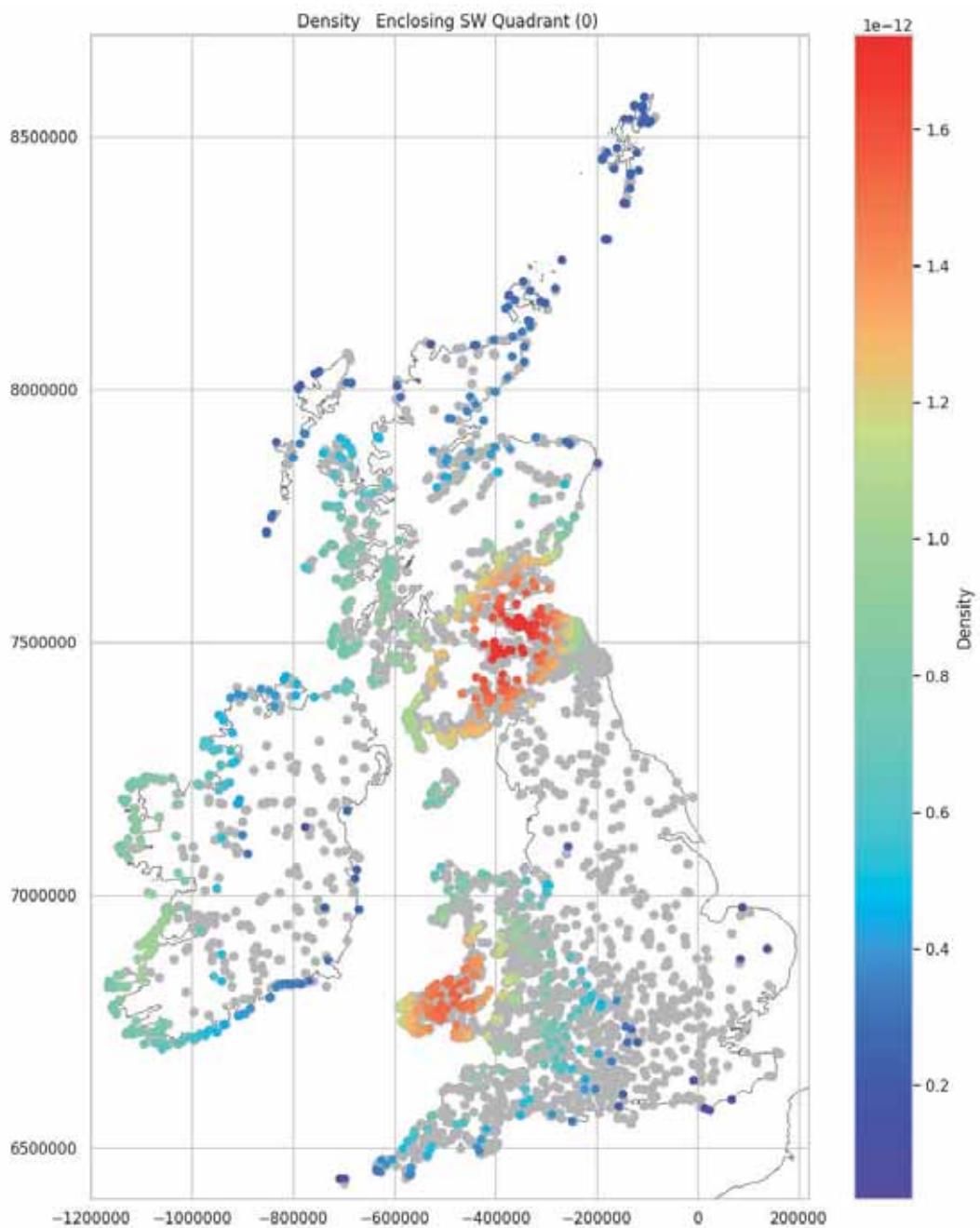
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

24.84%

SW Quadrant Data Density Mapped (0)

```
In [ ]: plot_density_over_grey(zero_sw_stats, 'Enclosing_SW_Quadrant (0)')
```



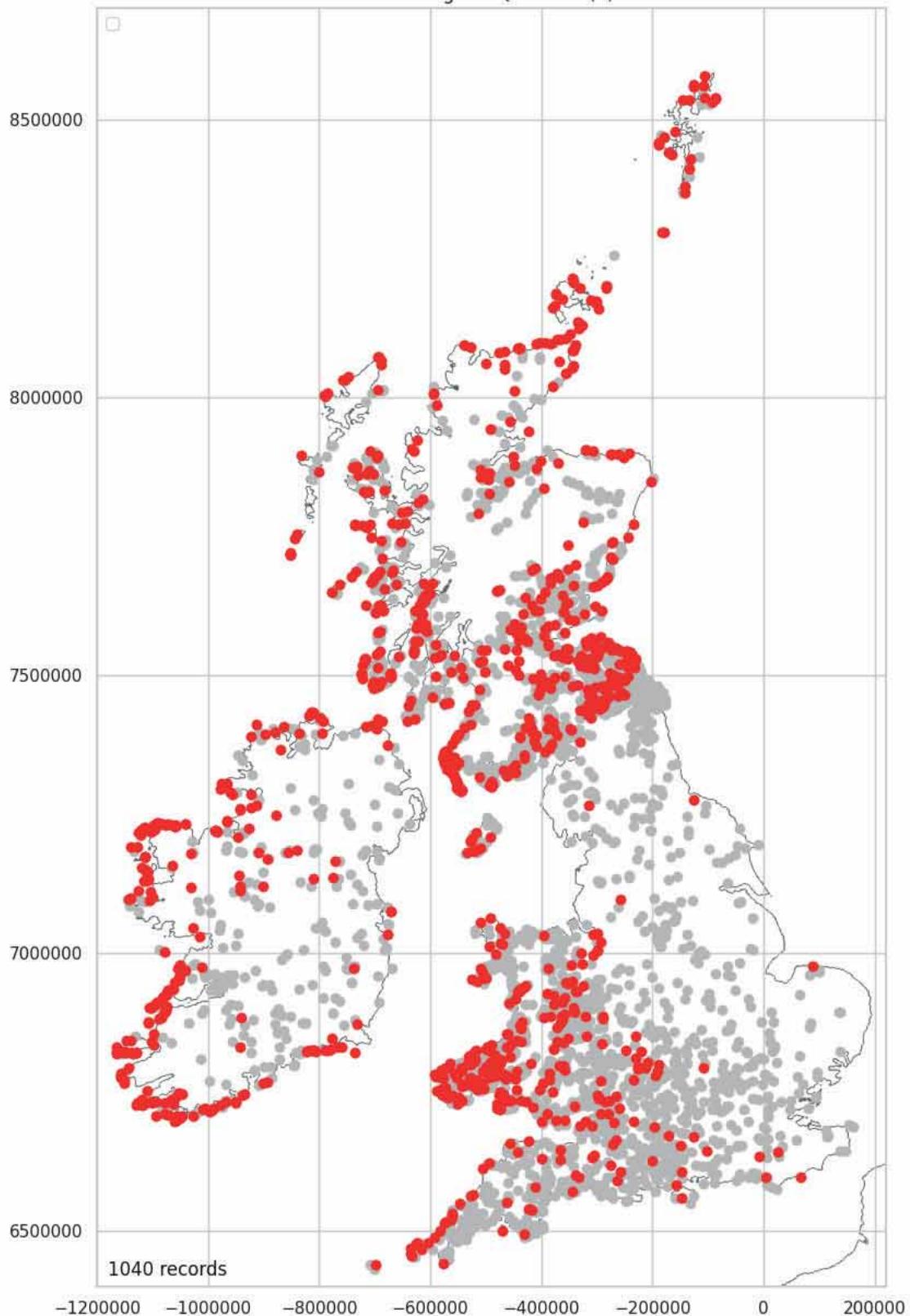
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

NW Quadrant Data Mapped (0)

```
In [ ]: zero_nw = nw_quadrant_data[nw_quadrant_data['Enclosing_NW_Quadrant']==0].copy()
zero_nw['Enclosing_NW_Quadrant'] = "Yes"
zero_nw_stats = plot_over_grey(zero_nw, 'Enclosing_NW_Quadrant', 'Yes', '(0)')
```

Enclosing NW Quadrant (0)



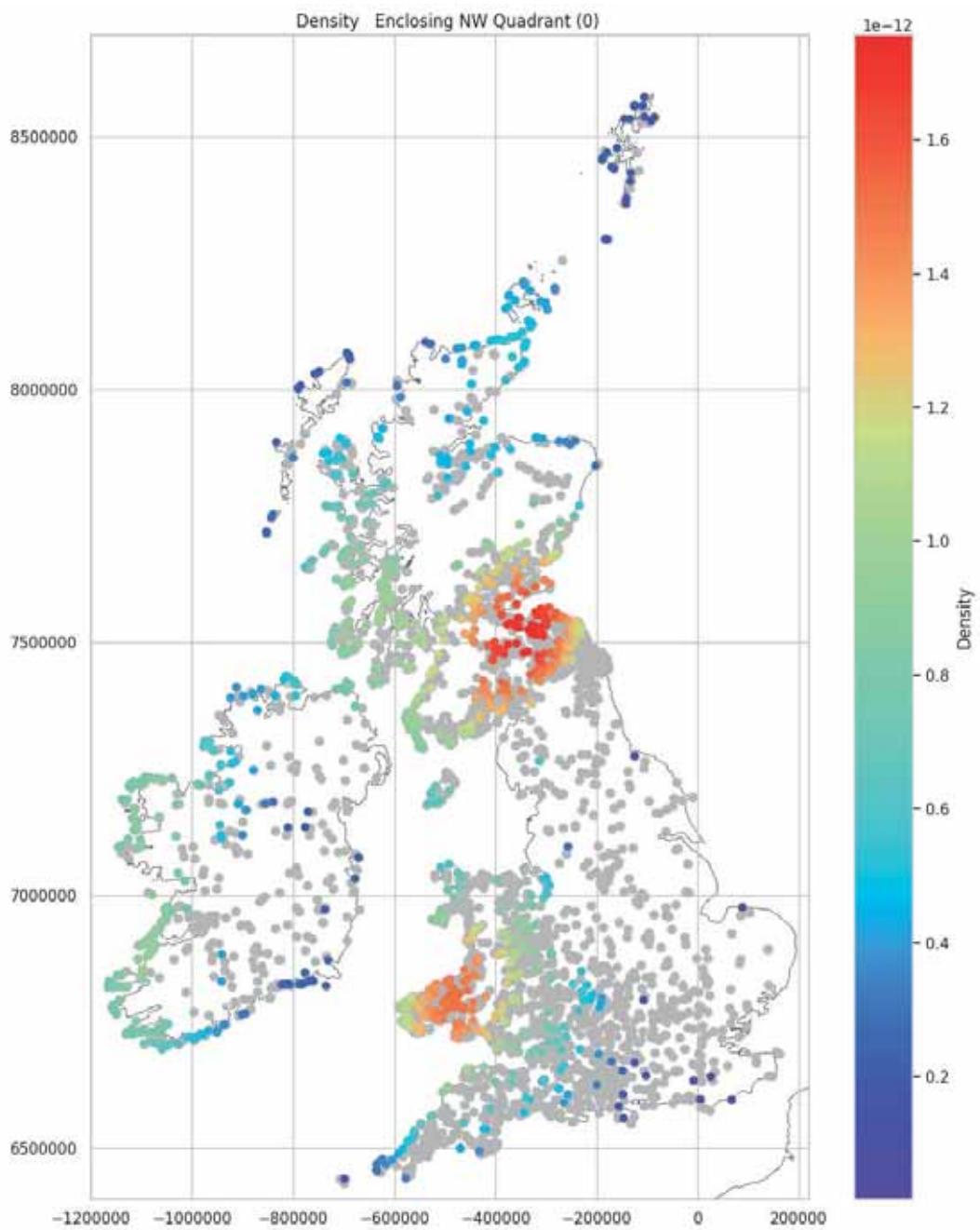
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

25.08%

NW Quadrant Data Density Mapped (0)

```
In [ ]: plot_density_over_grey(zero_nw_stats, 'Enclosing_NW_Quadrant (0)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

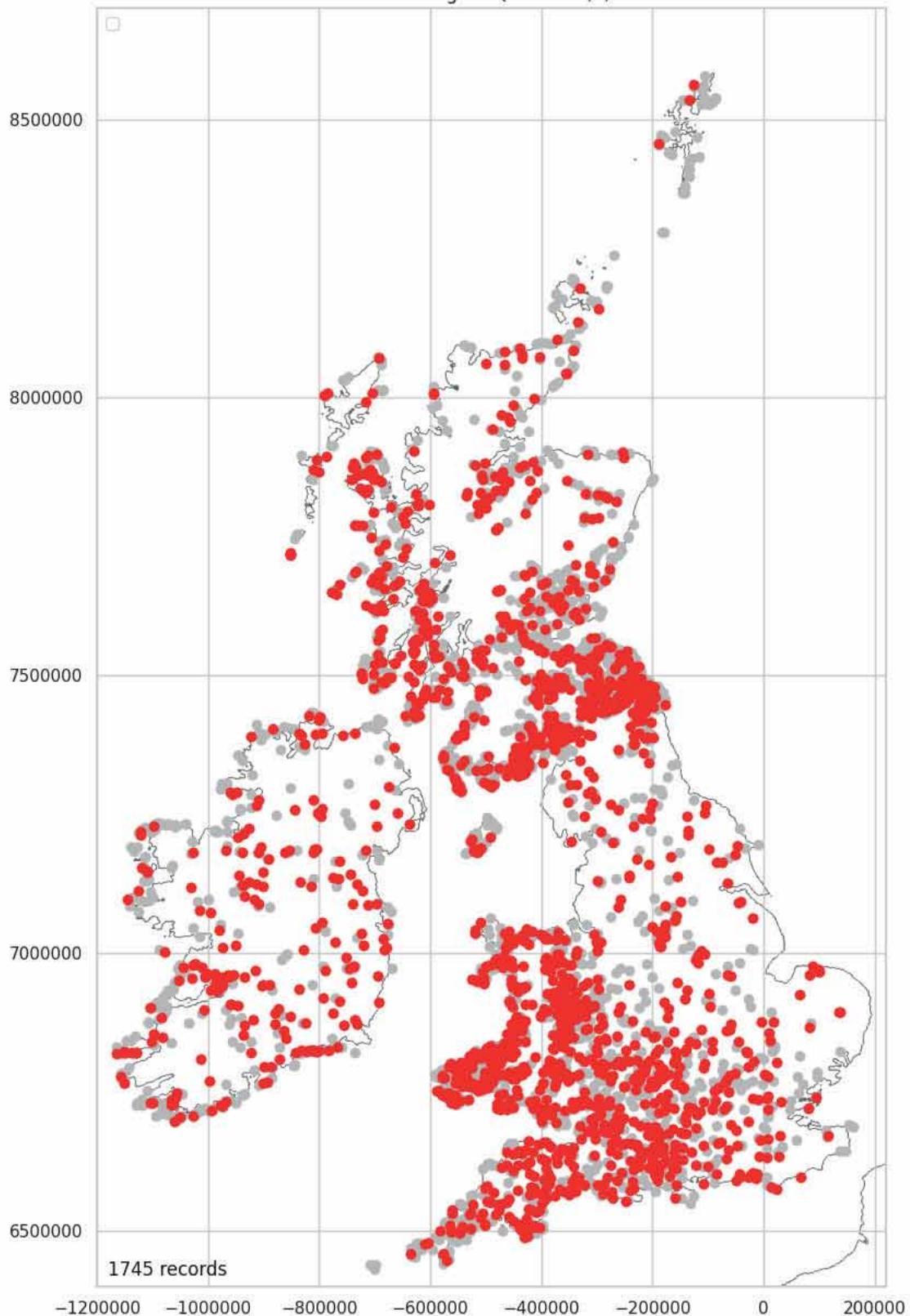
Quadrant Data Mapped (1)

The wide spread of forts with a single rampart reflects the distributions and clusters discussed in the ramparts section above. The general intensity of the clusters is as would be anticipated except for along the eastern fringe of the Cambrian Mountains where there is a slight reduction in the concentration of forts with ramparts facing northwest.

NE Quadrant Data Mapped (1)

```
In [ ]: one_ne = ne_quadrant_data[ne_quadrant_data['Encl osing_NE_Quadrant'] == 1].copy()
one_ne['Encl osing_NE_Quadrant'] = "Yes"
one_ne_stats = plot_over_grey(one_ne, 'Encl osing_NE_Quadrant', 'Yes', '(1)')
```

Enclosing NE Quadrant (1)



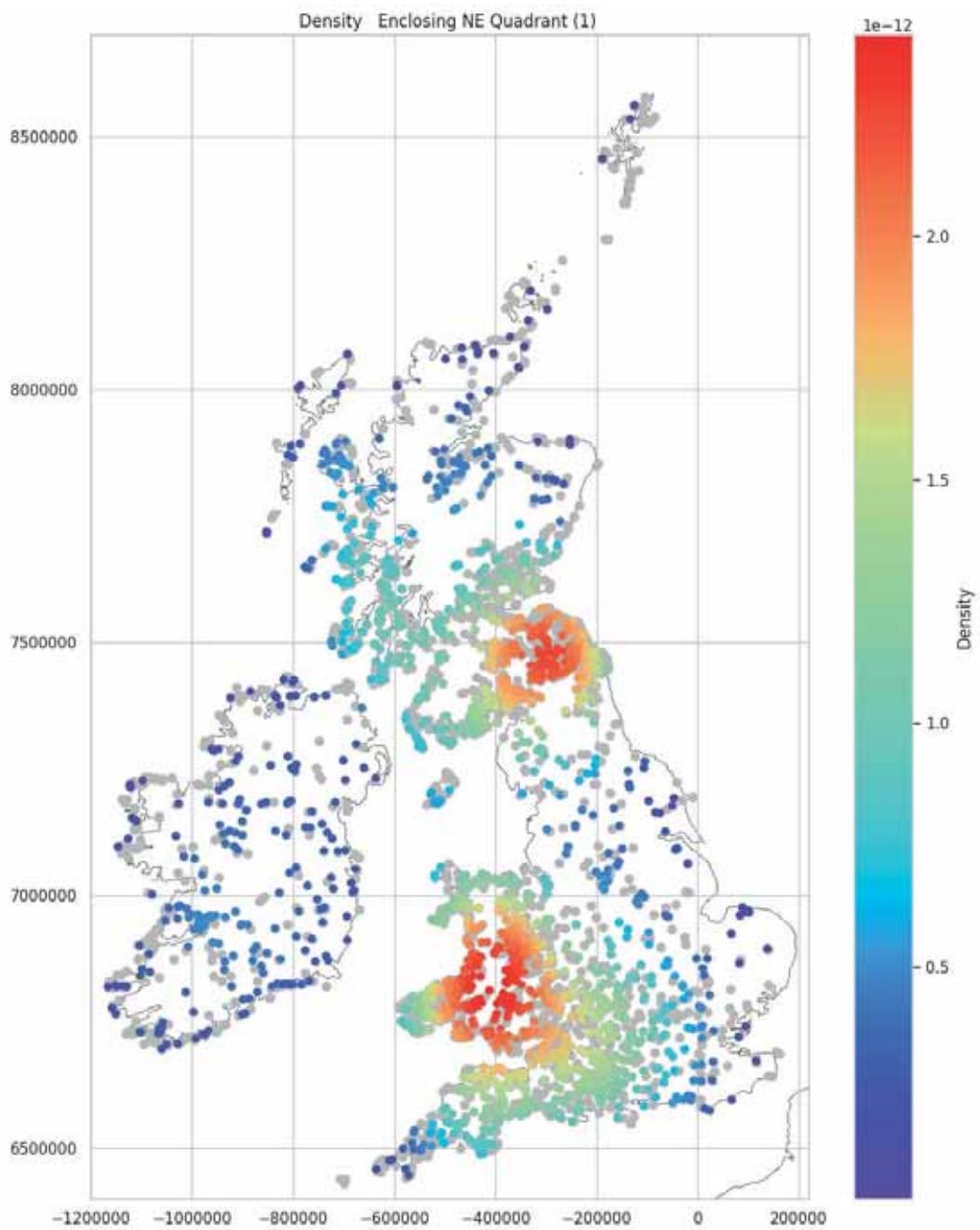
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

42.08%

NE Quadrant Data Density Mapped (1)

```
In [ ]: plot_density_over_grey(one_ne_stats, 'Enclosing_NE_Quadrant (1)')
```



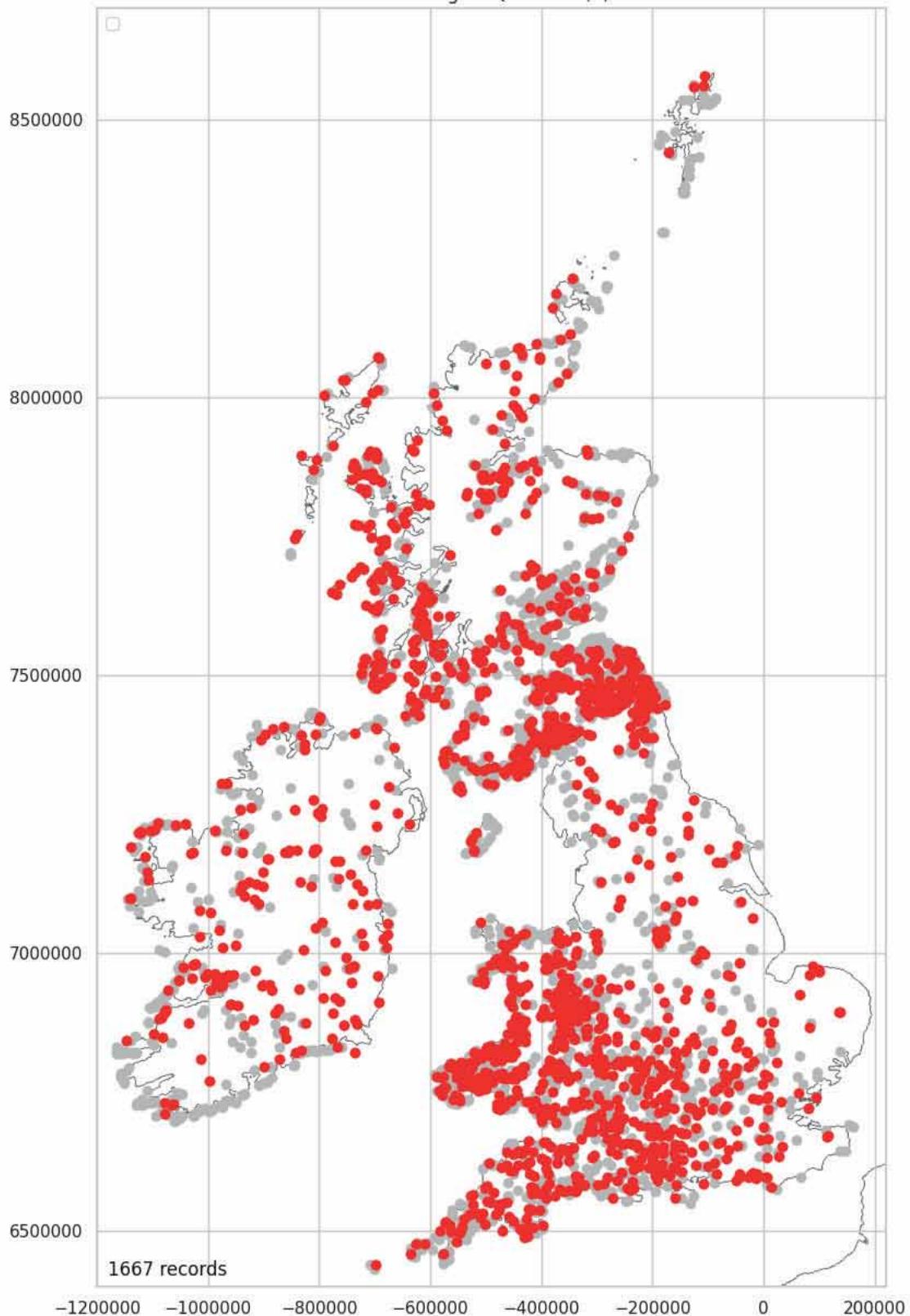
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SE Quadrant Data Mapped (1)

```
In [ ]: one_se = se_quadrant_data[se_quadrant_data['Enclosing_SE_Quadrant'] == 1].copy()
one_se['Enclosing_SE_Quadrant'] = "Yes"
one_se_stats = plot_over_grey(one_se, 'Enclosing_SE_Quadrant', 'Yes', '(1)')
```

Enclosing SE Quadrant (1)



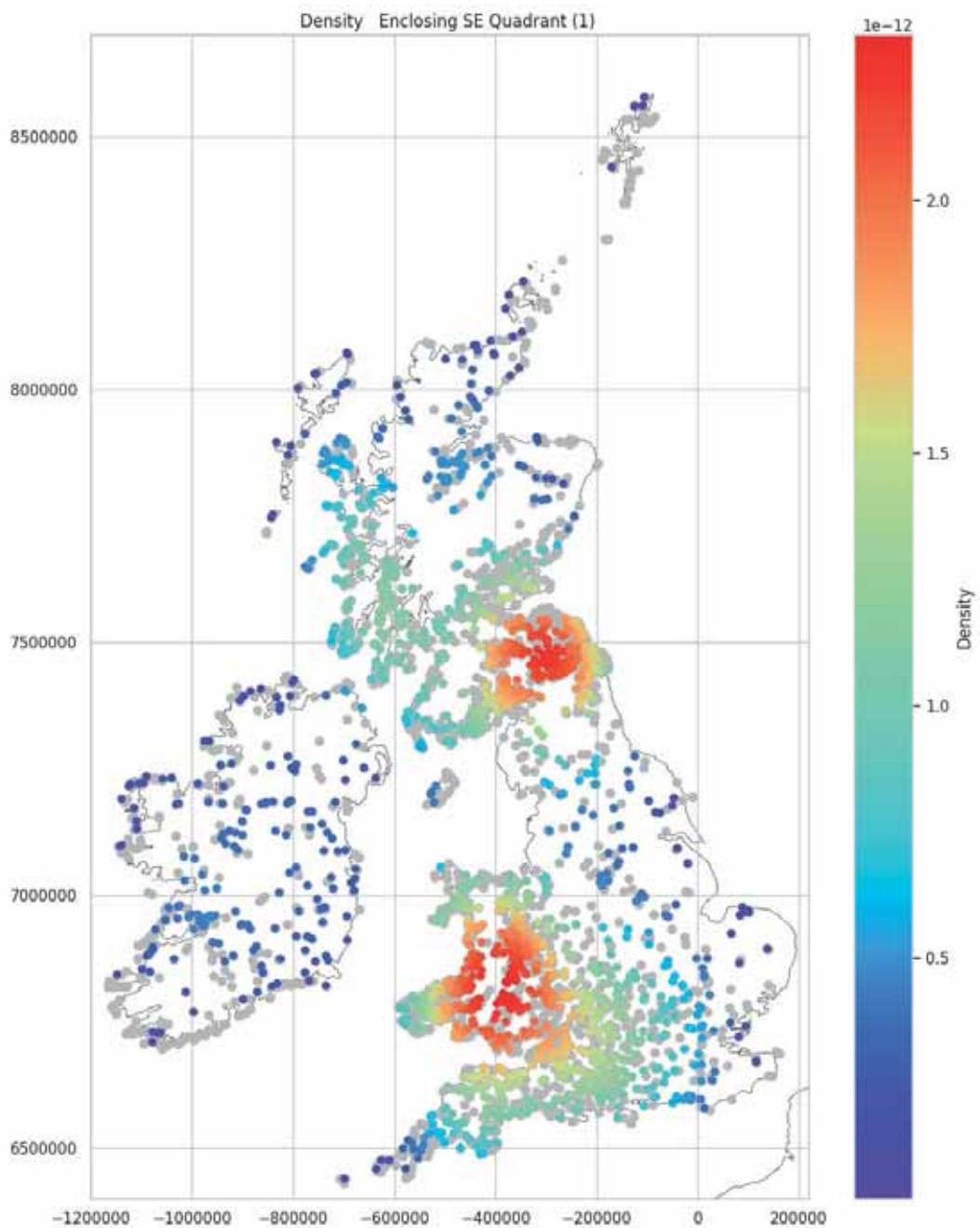
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

40.2%

SE Quadrant Data Density Mapped (1)

```
In [ ]: plot_density_over_grey(one_se_stats, 'Enclosing_SE_Quadrant (1)')
```



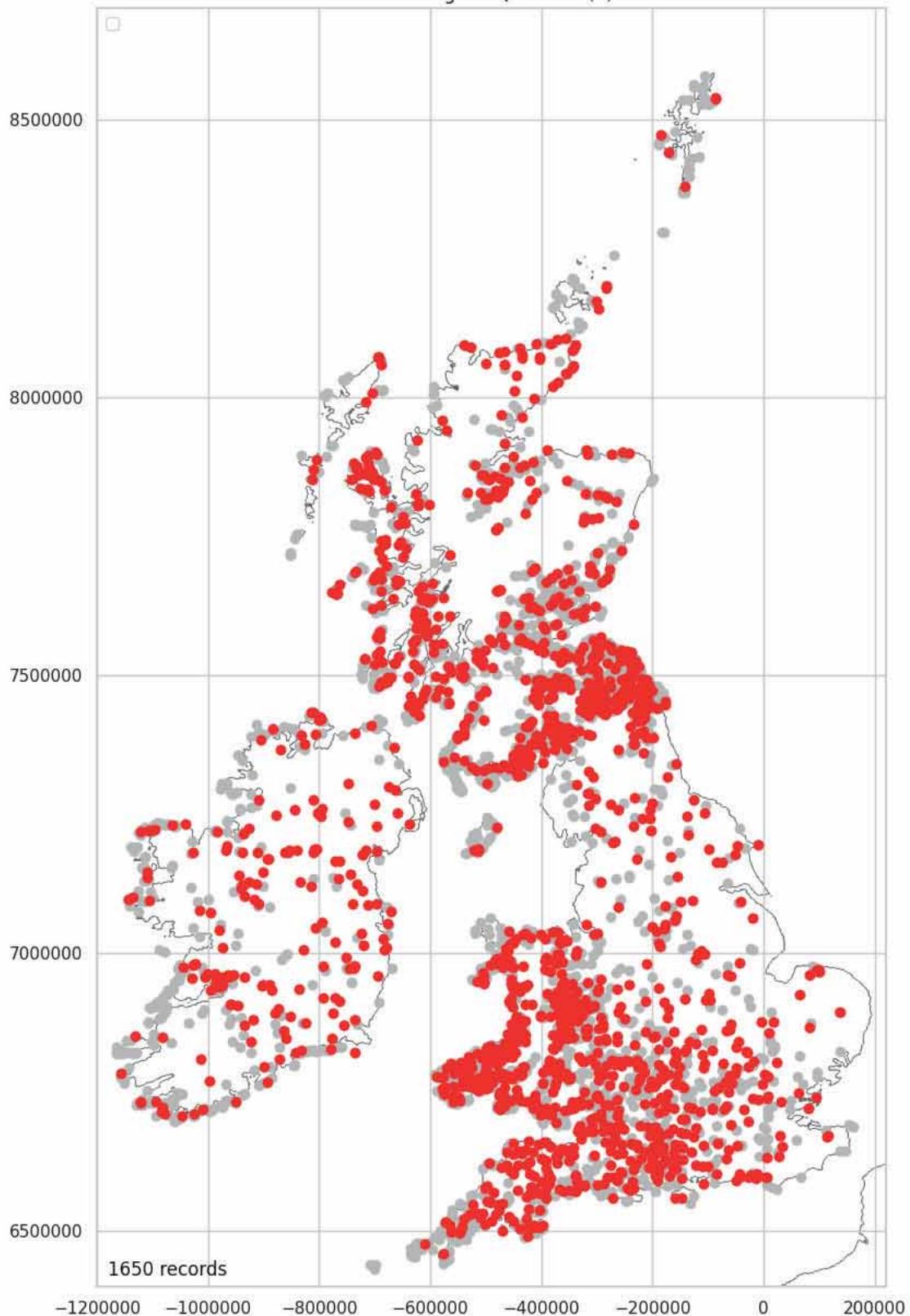
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SW Quadrant Data Mapped (1)

```
In [ ]: one_sw = sw_quadrant_data[sw_quadrant_data['Encl osi ng_SW_Quadrant'] == 1].copy()
one_sw['Encl osi ng_SW_Quadrant'] = "Yes"
one_sw_stats = plot_over_grey(one_sw, 'Encl osi ng_SW_Quadrant', 'Yes', '(1)')
```

Enclosing SW Quadrant (1)



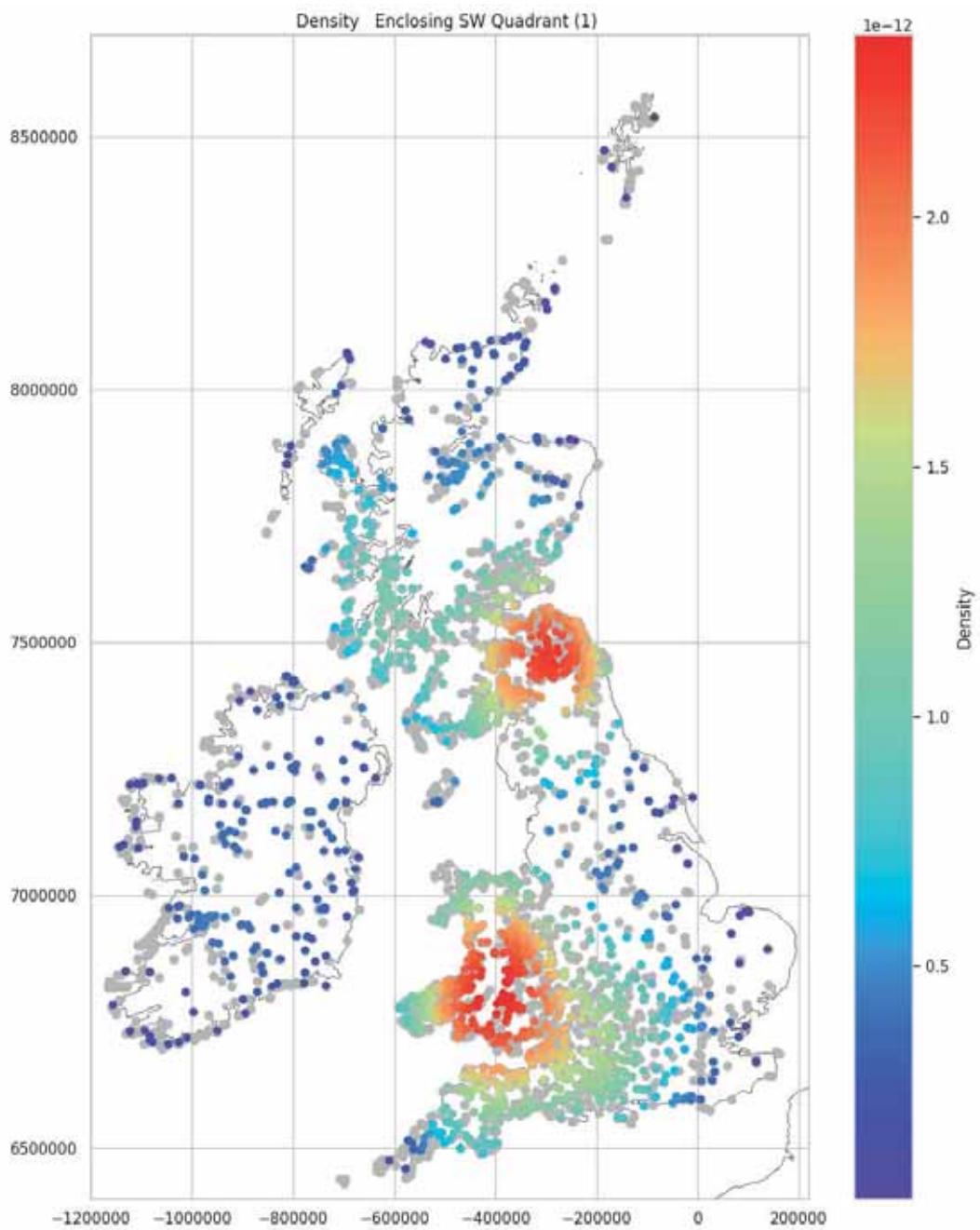
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

39.79%

SW Quadrant Data Density Mapped (1)

```
In [ ]: plot_density_over_grey(one_sw_stats, 'Enclosing_SW_Quadrant (1)')
```



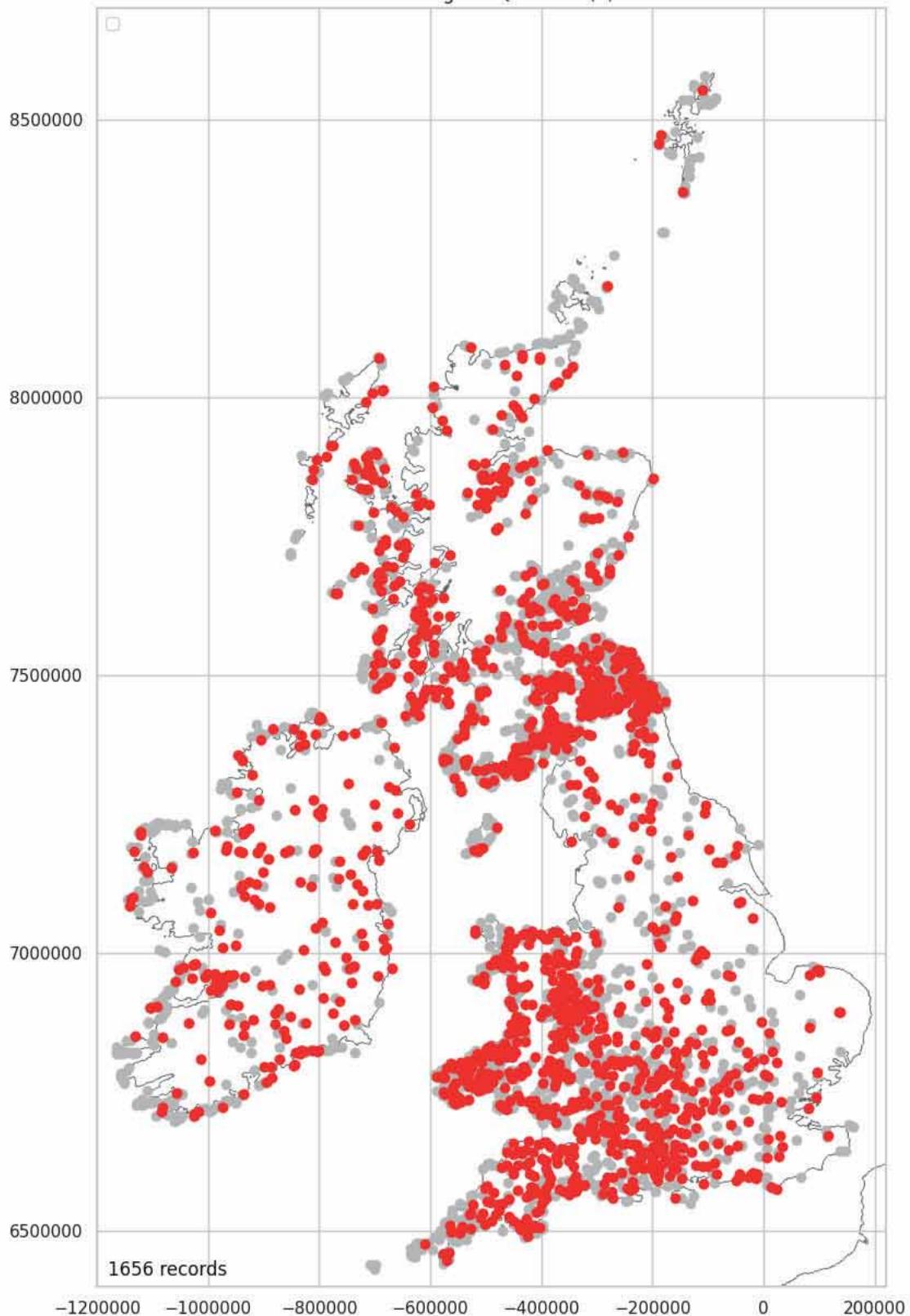
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

NW Quadrant Data Mapped (1)

```
In [ ]: one_nw = nw_quadrant_data[nw_quadrant_data['Enclosing_NW_Quadrant']==1].copy()
one_nw['Enclosing_NW_Quadrant'] = "Yes"
one_nw_stats = plot_over_grey(one_nw, 'Enclosing_NW_Quadrant', 'Yes', '(1)')
```

Enclosing NW Quadrant (1)



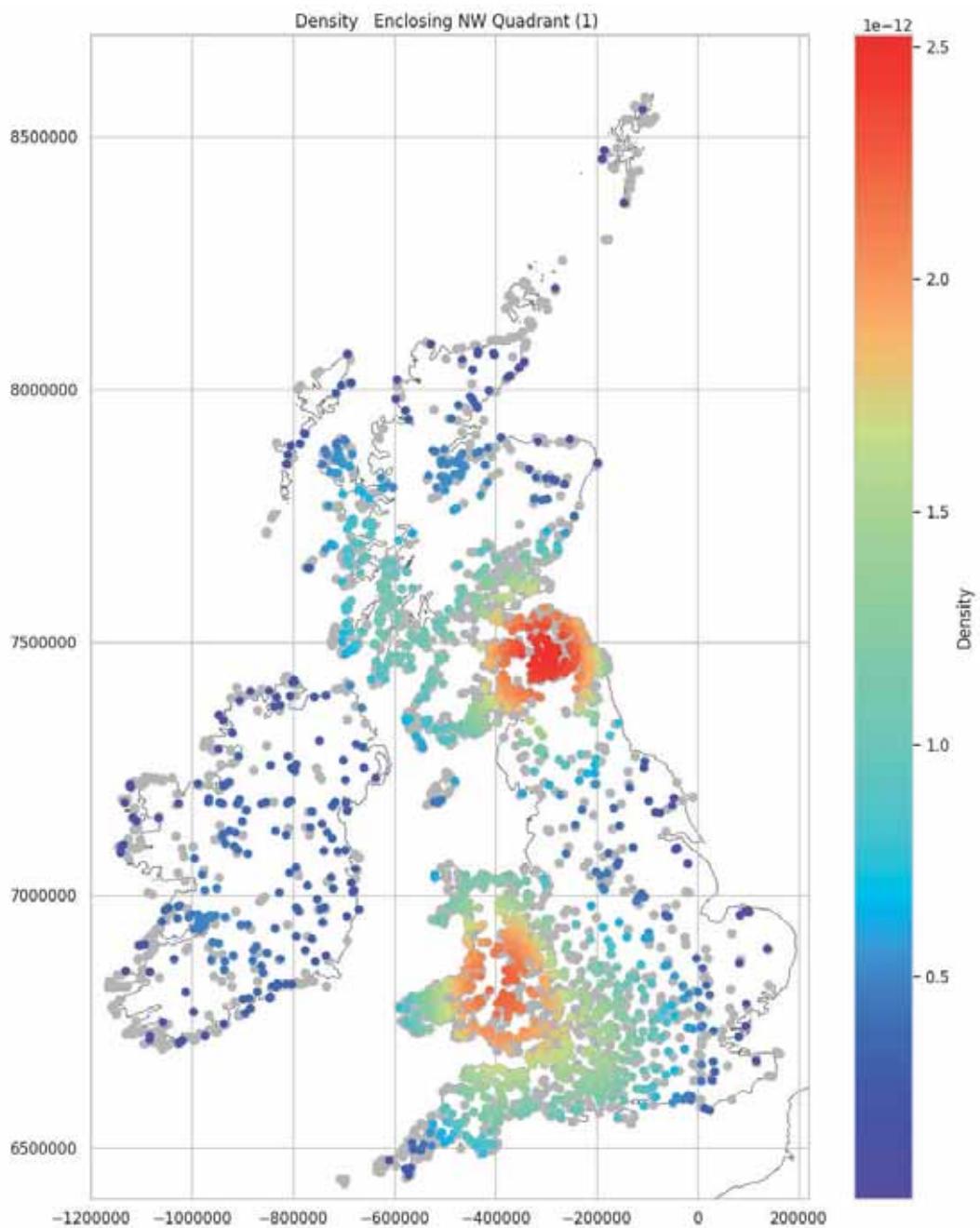
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

39. 93%

NW Quadrant Data Density Mapped (1)

```
In [ ]: plot_density_over_grey(one_nw_stats, 'Enclosing_NW_Quadrant (1)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

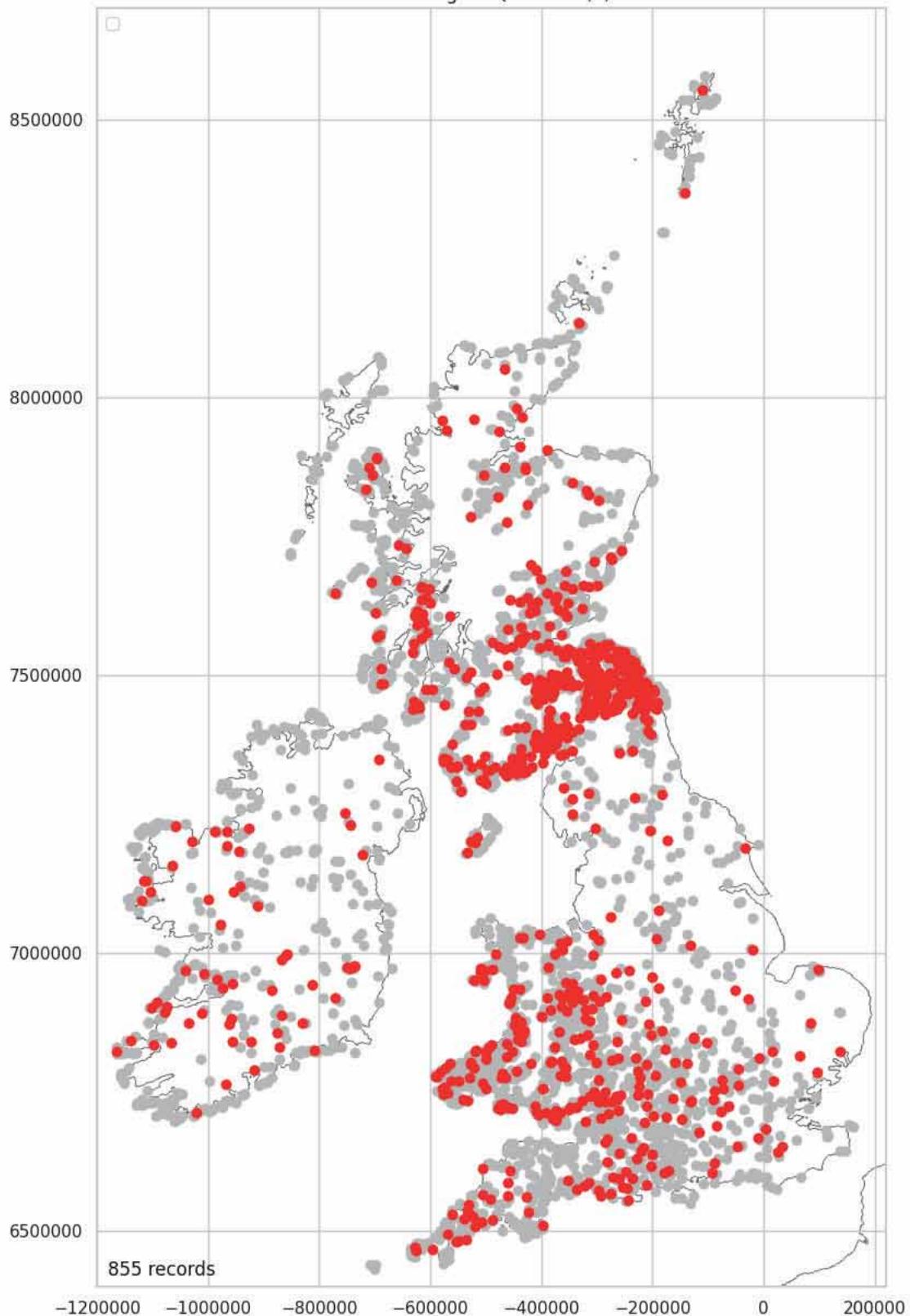
Quadrant Data Mapped (2)

There is little to be said about the quadrant data for two ramparts. Unsurprisingly, it is focussed over the Northeast. See [Ramparts Mapped \(2\)](#).

NE Quadrant Data Mapped (2)

```
In [ ]: two_ne = ne_quadrant_data[ne_quadrant_data['Encl osing_NE_Quadrant']==2].copy()
two_ne['Encl osing_NE_Quadrant'] = "Yes"
two_ne_stats = plot_over_grey(two_ne, 'Encl osing_NE_Quadrant', 'Yes', '(2)')
```

Enclosing NE Quadrant (2)



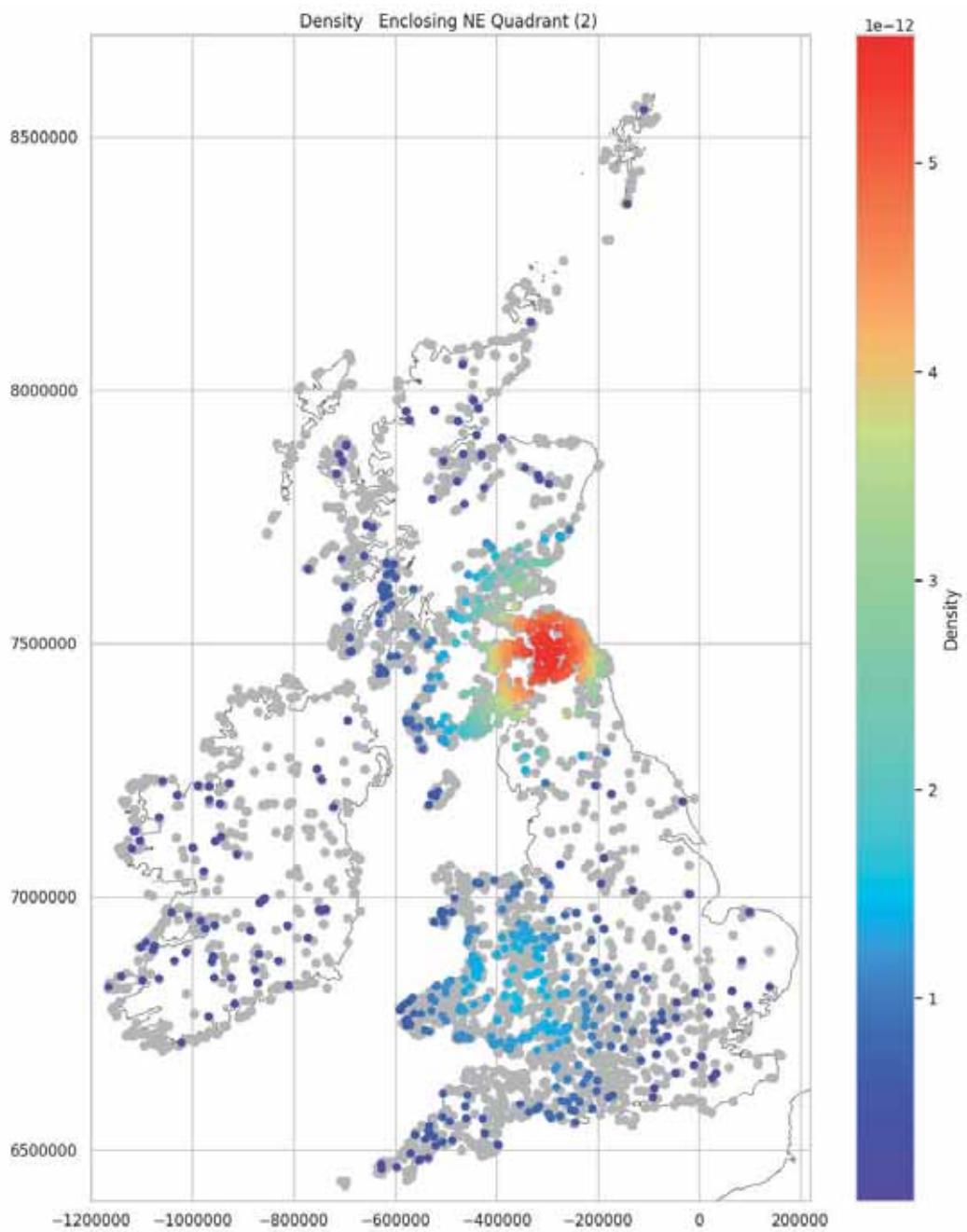
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

20.62%

NE Quadrant Data Density Mapped (2)

```
In [ ]: plot_density_over_grey(two_ne_stats, 'Enclosing_NE_Quadrant (2)')
```



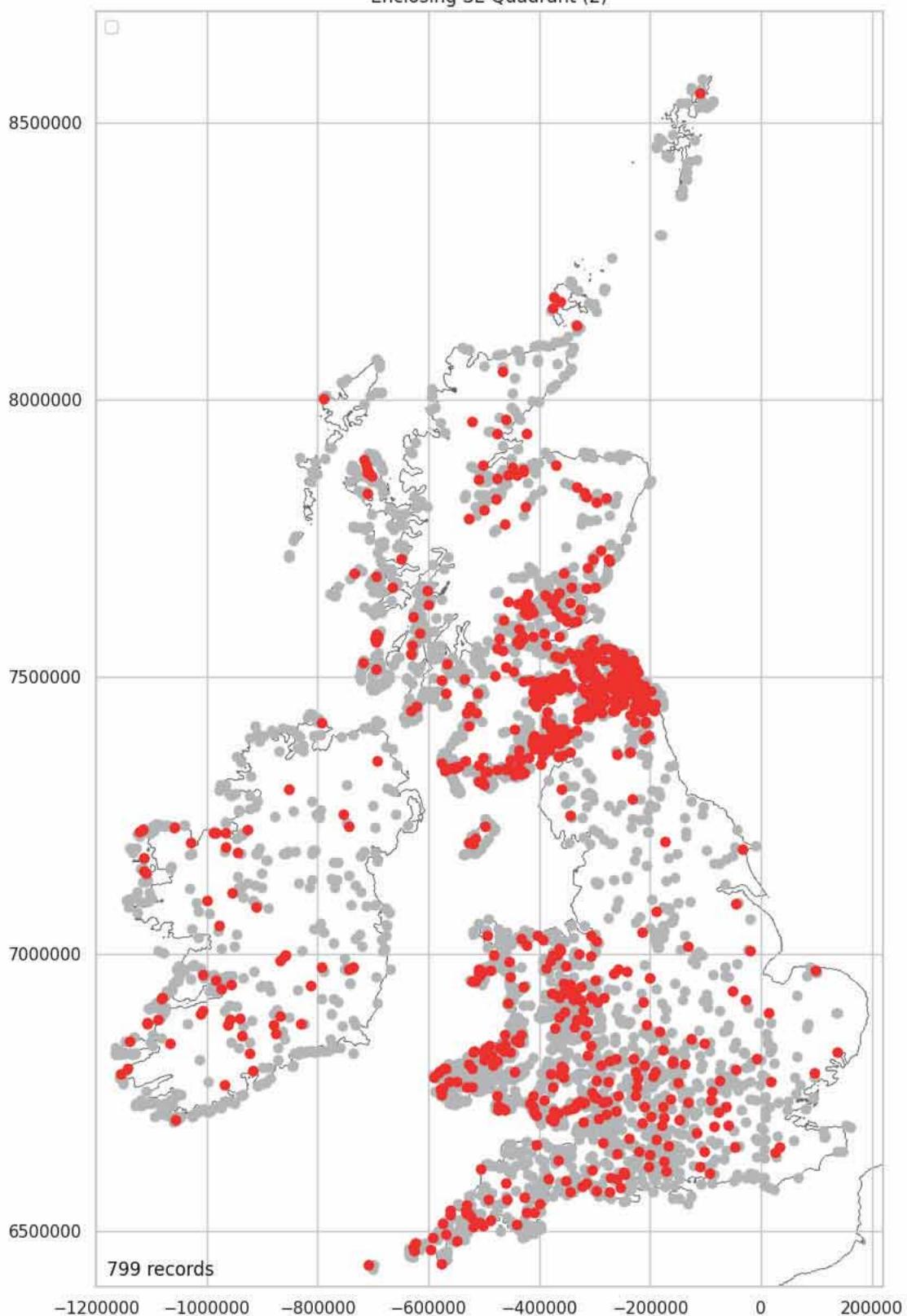
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SE Quadrant Data Mapped (2)

```
In [ ]: two_se = se_quadrant_data[se_quadrant_data['Enclosing_SE_Quadrant']==2].copy()
two_se['Enclosing_SE_Quadrant'] = "Yes"
two_se_stats = plot_over_grey(two_se, 'Enclosing_SE_Quadrant', 'Yes', '(2)')
```

Enclosing SE Quadrant (2)



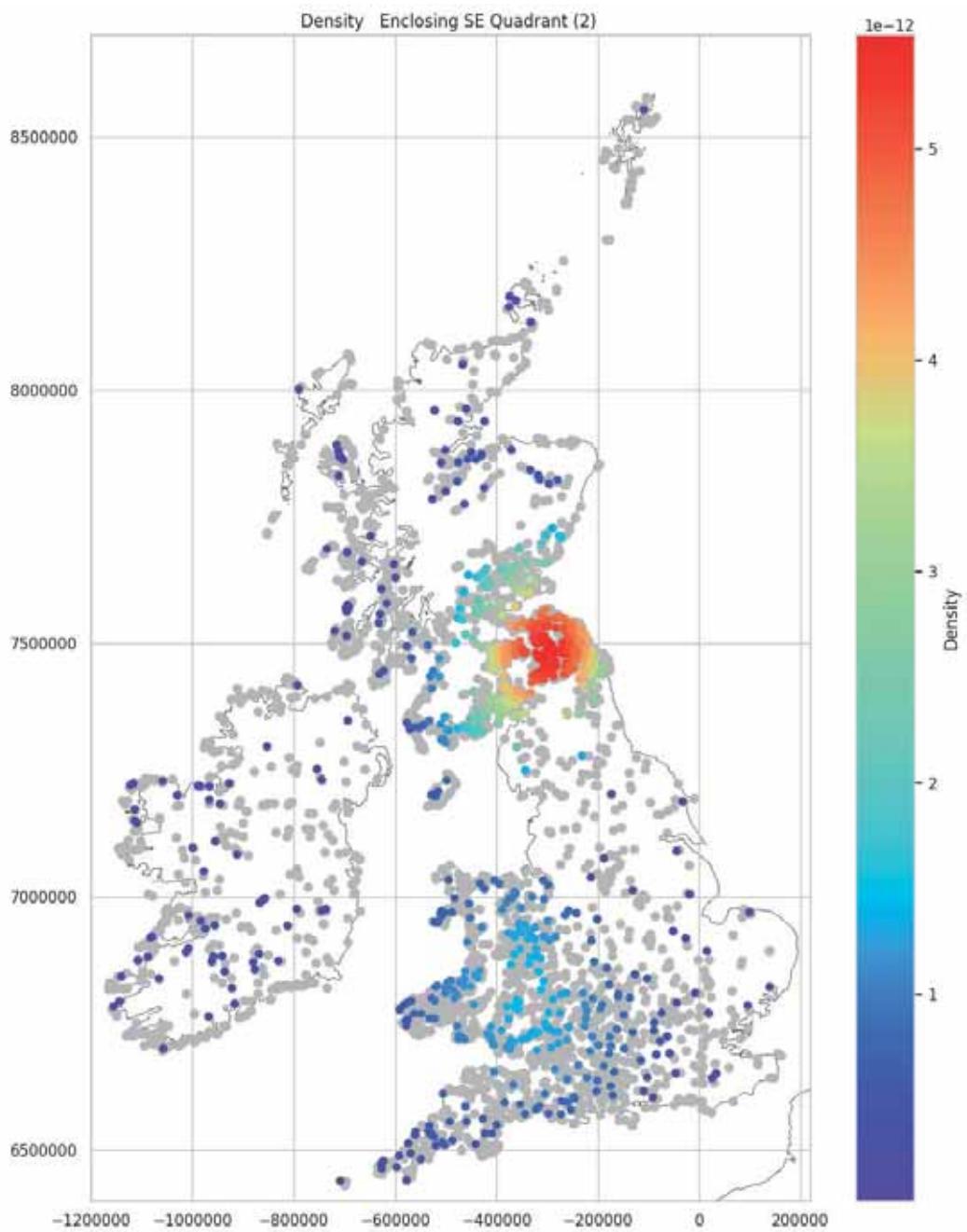
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

19.27%

SE Quadrant Data Density Mapped (2)

```
In [ ]: plot_density_over_grey(two_se_stats, 'Enclosing_SE_Quadrant (2)')
```



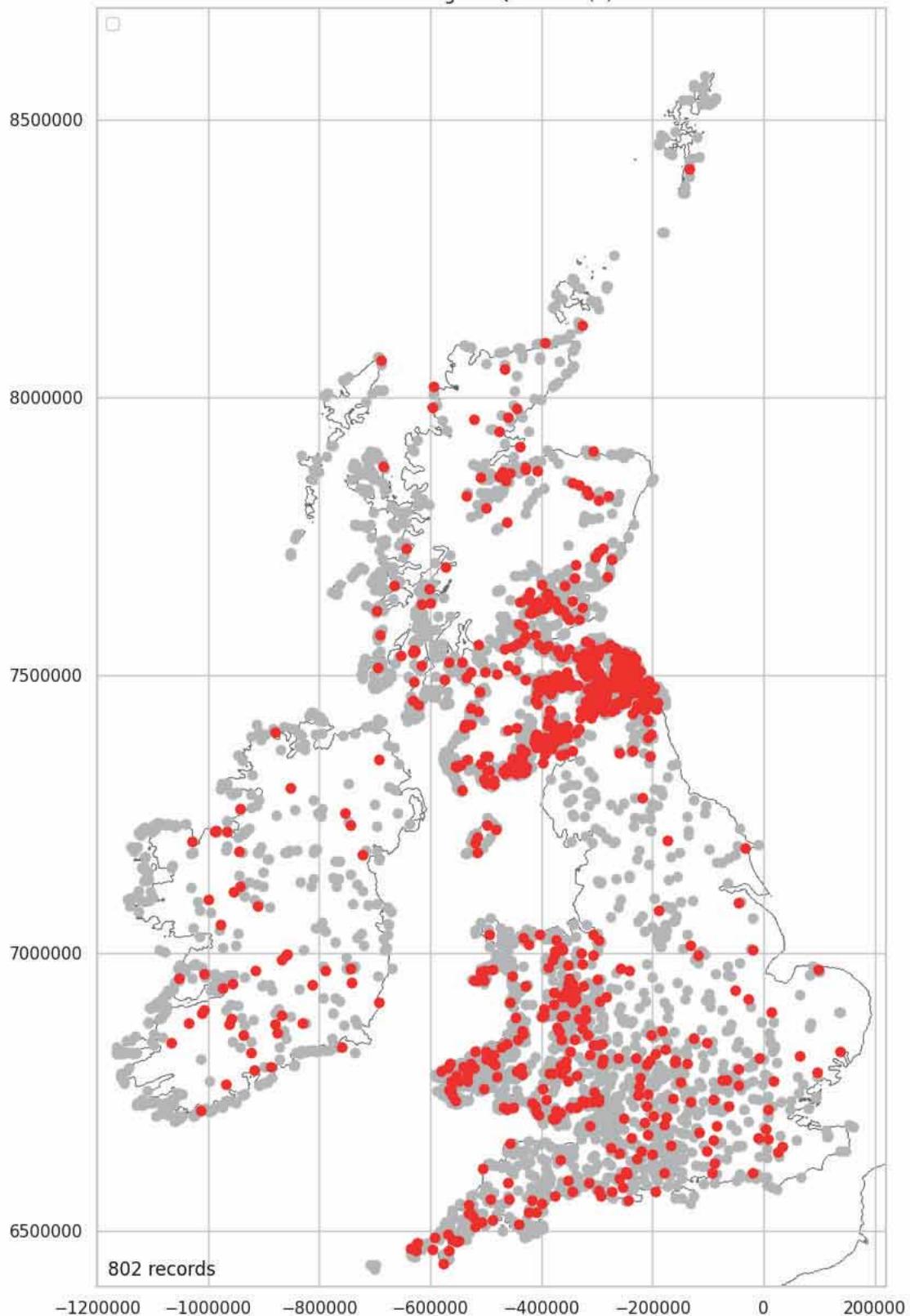
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SW Quadrant Data Mapped (2)

```
In [ ]: two_sw = sw_quadrant_data[sw_quadrant_data['Enclosing_SW_Quadrant']==2].copy()
two_sw['Enclosing_SW_Quadrant'] = "Yes"
two_sw_stats = plot_over_grey(two_sw, 'Enclosing_SW_Quadrant', 'Yes', '(2)')
```

Enclosing SW Quadrant (2)



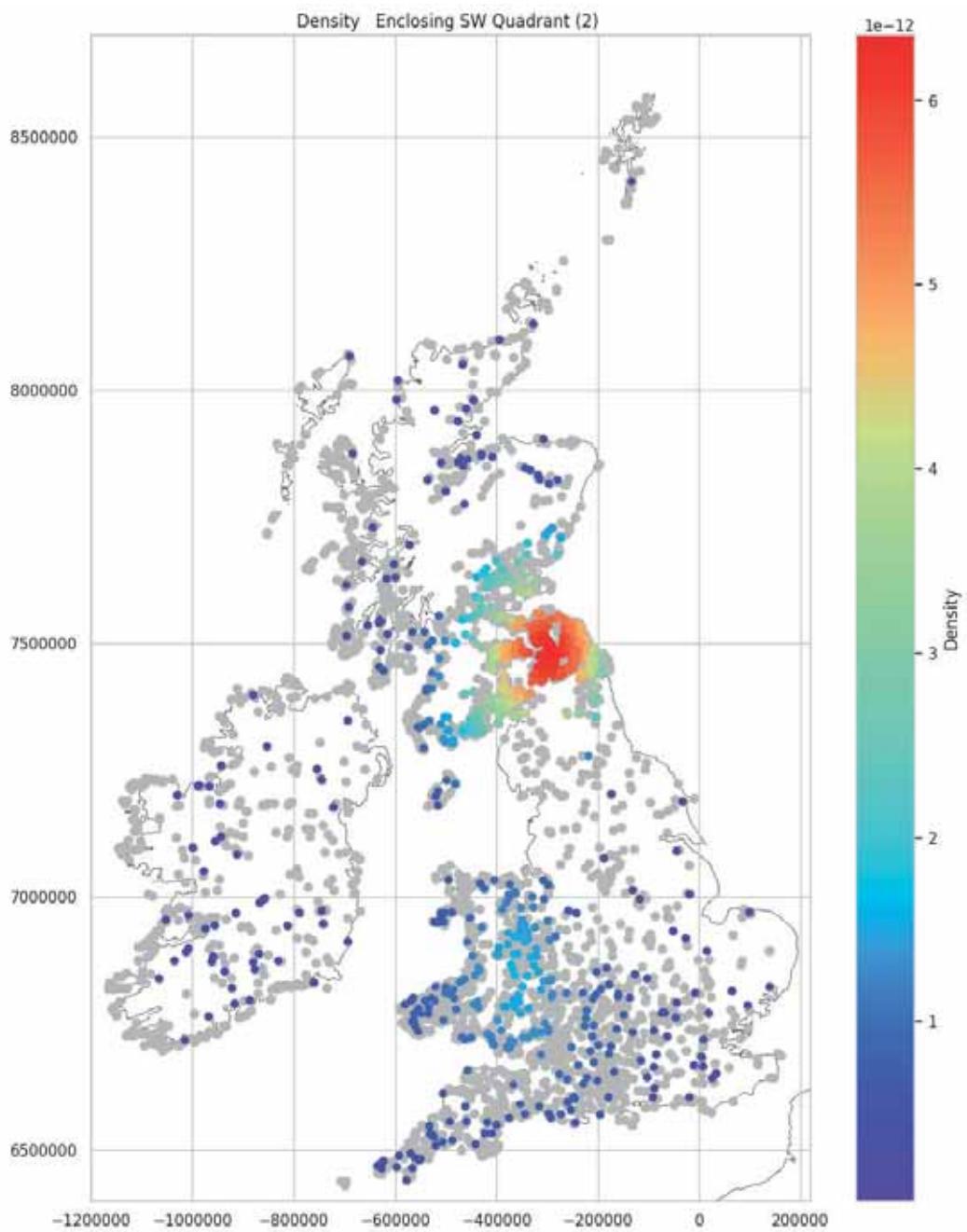
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

19.34%

SW Quadrant Data Density Mapped (2)

```
In [ ]: plot_density_over_grey(two_sw_stats, 'Enclosing_SW_Quadrant (2)')
```



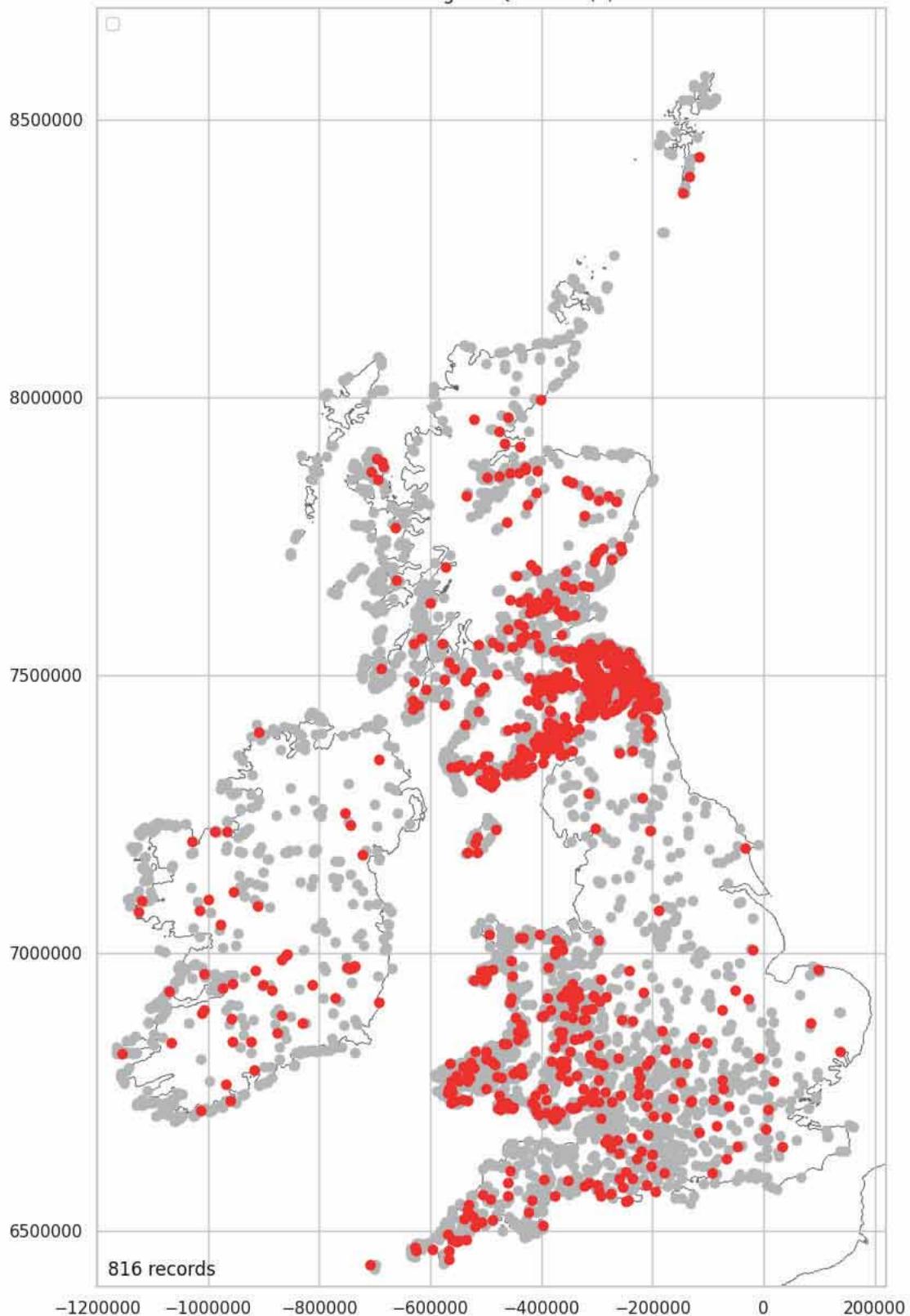
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

NW Quadrant Data Mapped (2)

```
In [ ]: two_nw = nw_quadrant_data[nw_quadrant_data['Enclosing_NW_Quadrant']==2].copy()
two_nw['Enclosing_NW_Quadrant'] = "Yes"
two_nw_stats = plot_over_grey(two_nw, 'Enclosing_NW_Quadrant', 'Yes', '(2)')
```

Enclosing NW Quadrant (2)



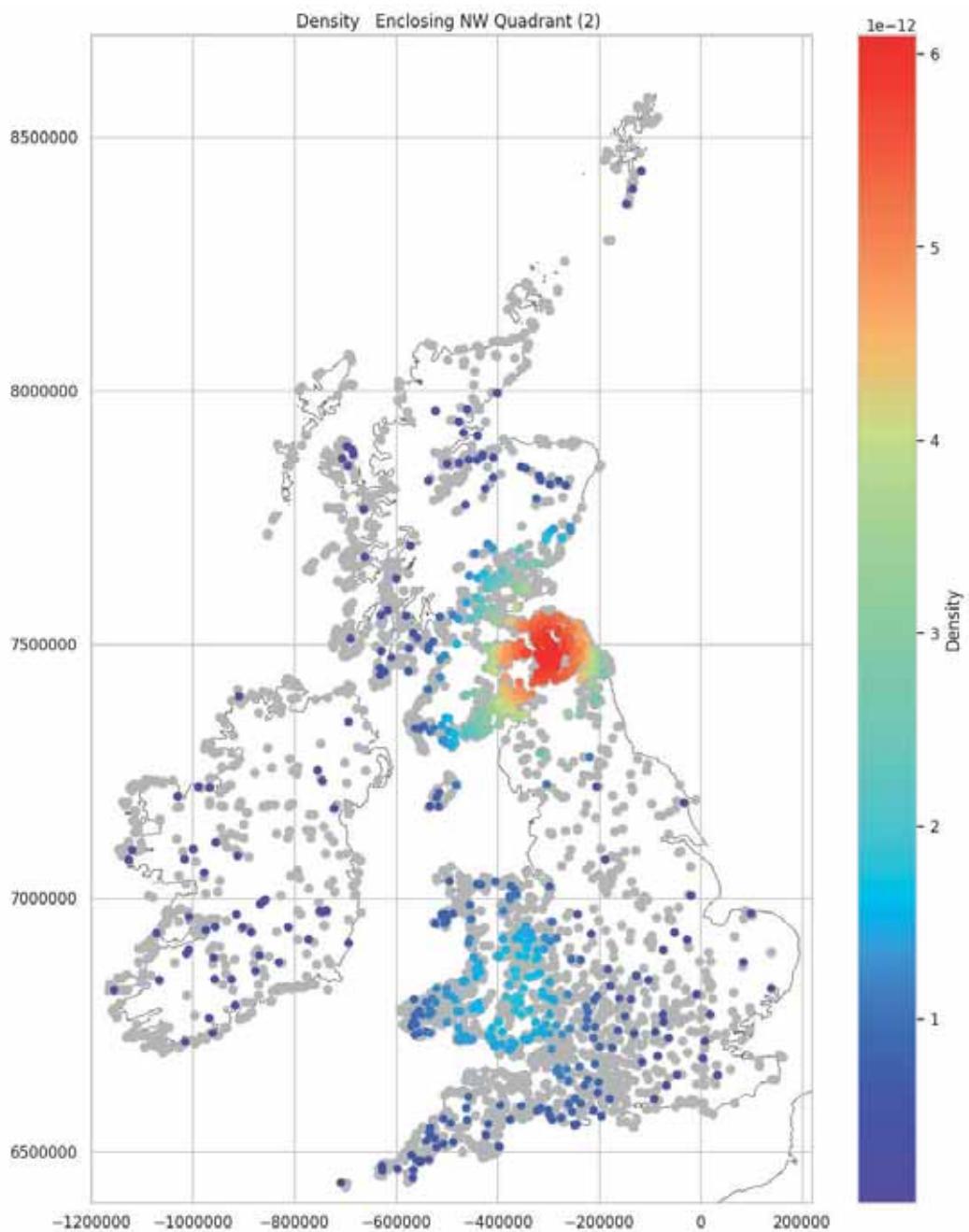
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

19.68%

NW Quadrant Data Density Mapped (2)

```
In [ ]: plot_density_over_grey(two_nw_stats, 'Enclosing_NW_Quadrant (2)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

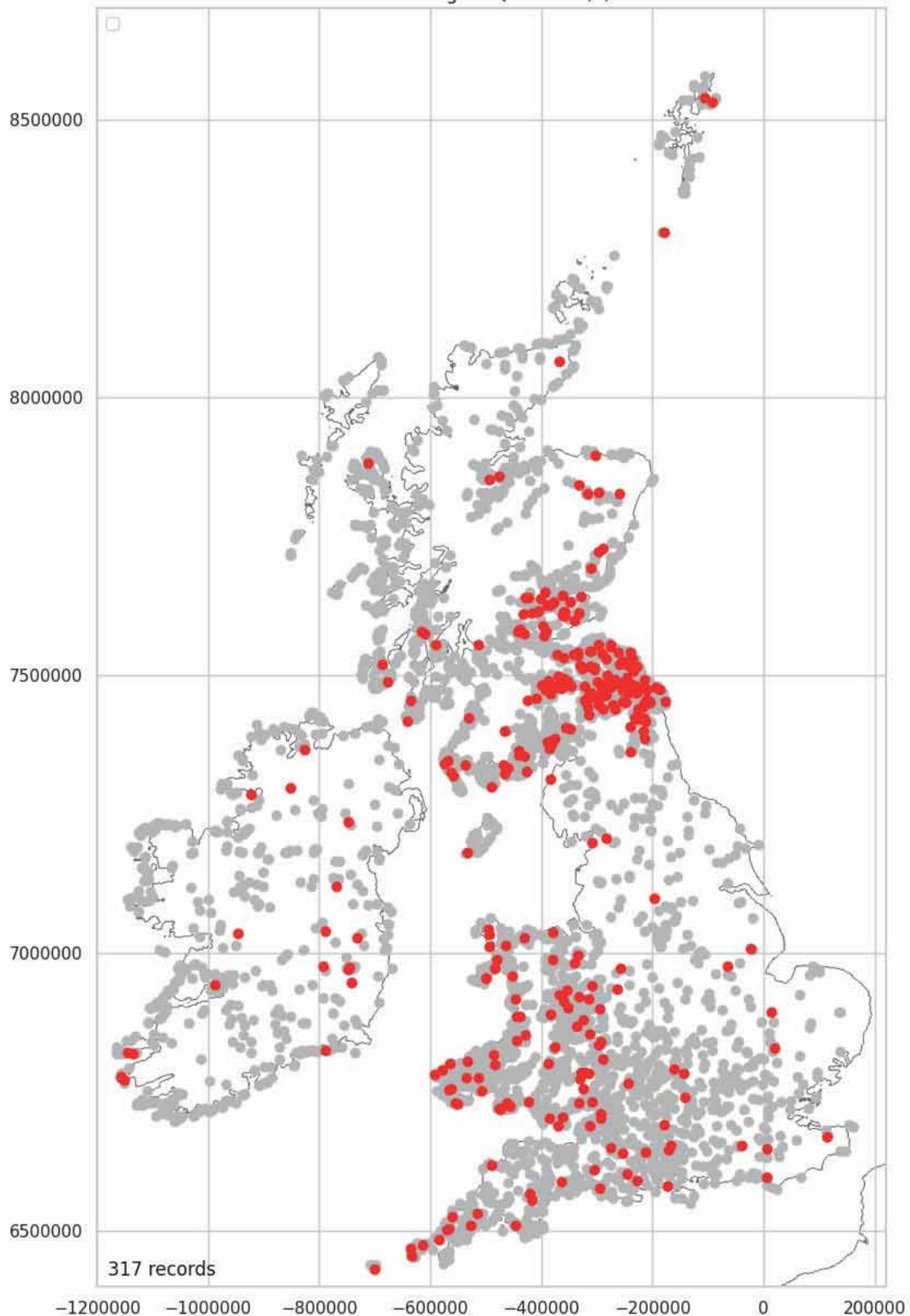
Quadrant Data Mapped (3)

There are no surprises in the quadrant data mapping three ramparts. As expected, it is focussed on the Northeast.

NE Quadrant Data Mapped (3)

```
In [ ]: three_ne = ne_quadrant_data[ne_quadrant_data['Enclosing_NE_Quadrant']==3].copy()
three_ne['Enclosing_NE_Quadrant'] = "Yes"
three_ne_stats = plot_over_grey(three_ne, 'Enclosing_NE_Quadrant', 'Yes', '(3)')
```

Enclosing NE Quadrant (3)



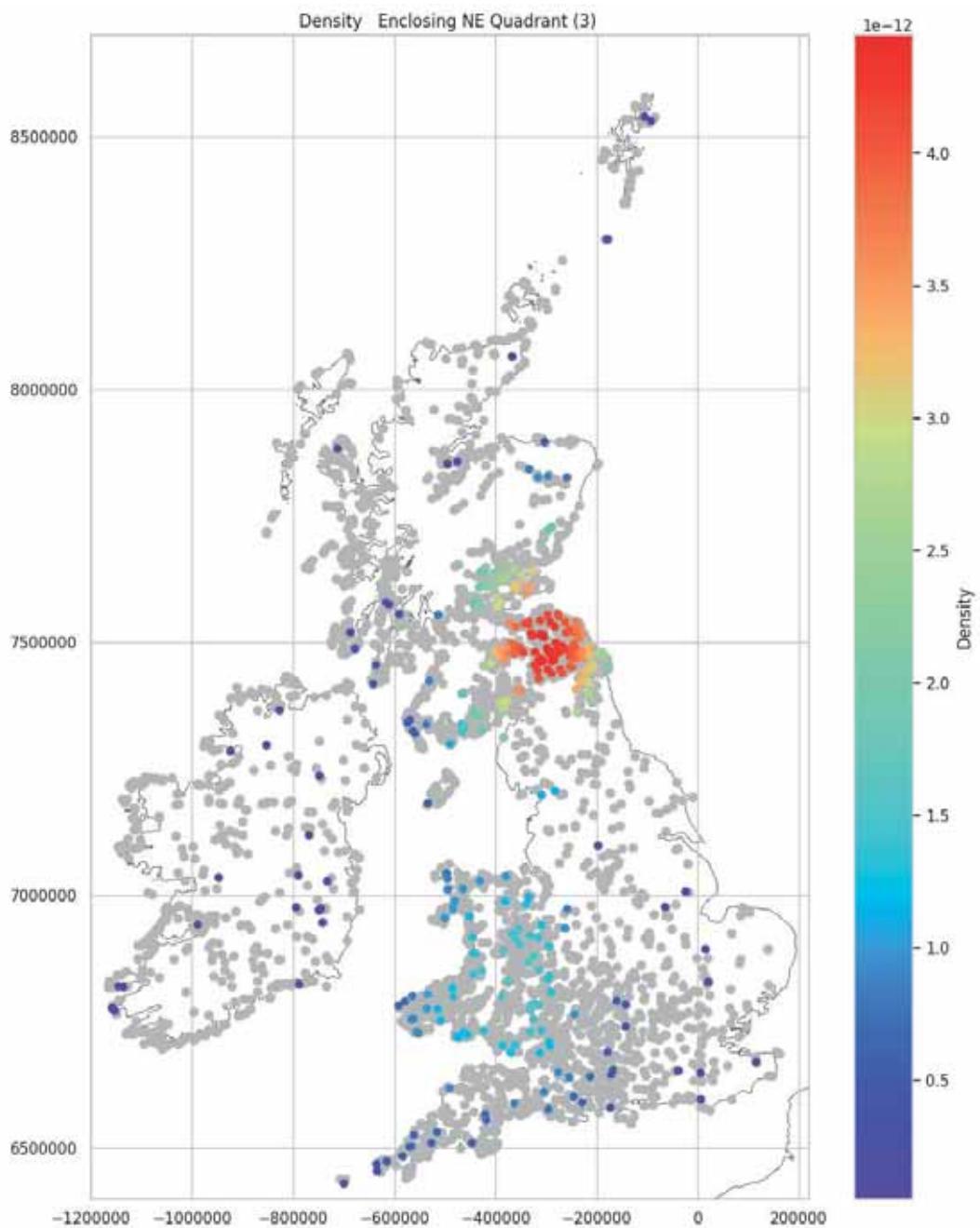
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

7.64%

NE Quadrant Data Density Mapped (3)

```
In [ ]: plot_density_over_grey(three_ne_stats, 'Enclosing_NE_Quadrant (3)')
```



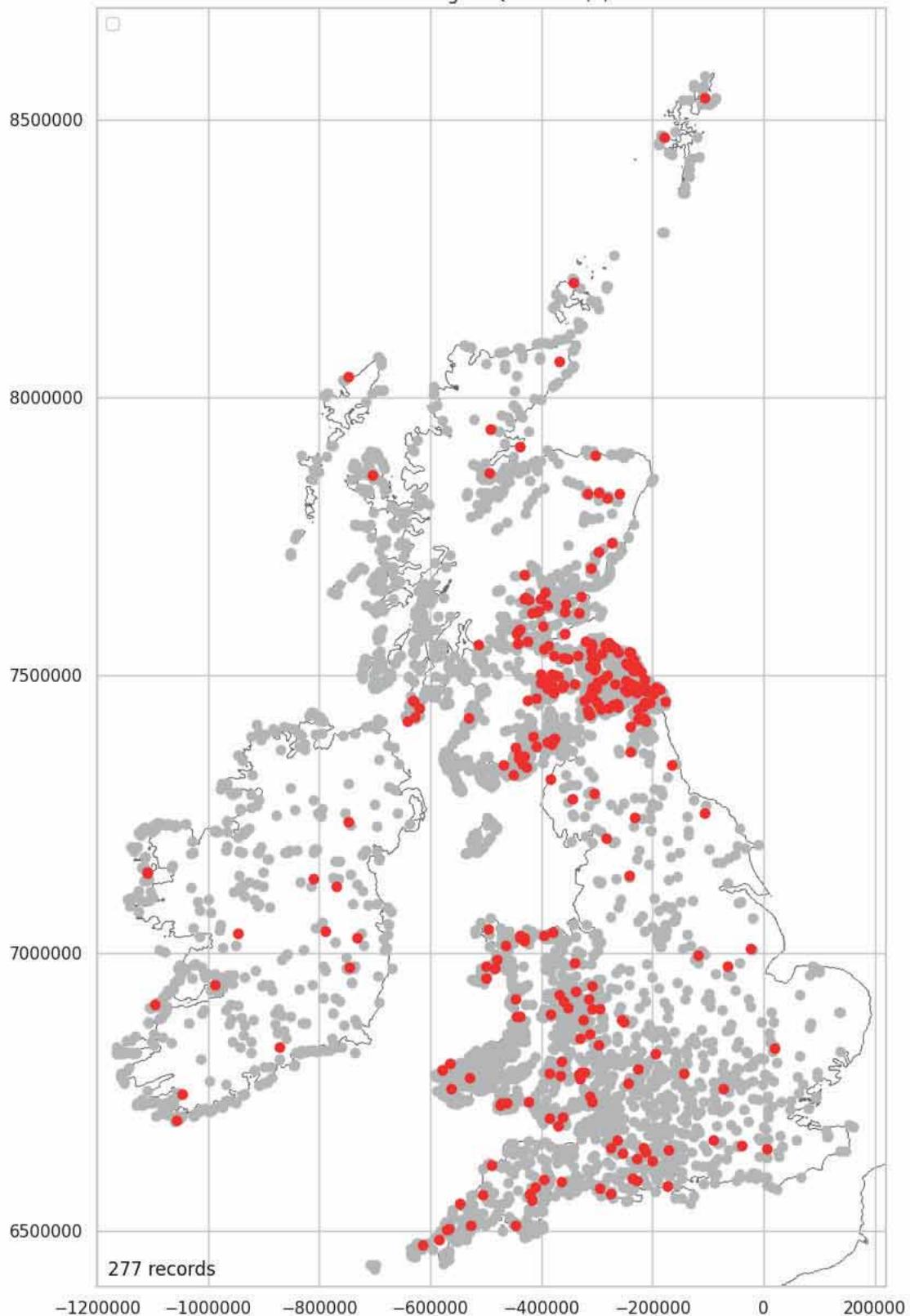
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SE Quadrant Data Mapped (3)

```
In [ ]: three_se = se_quadrant_data[se_quadrant_data['Encl osing_SE_Quadrant']==3].copy()
three_se['Encl osing_SE_Quadrant'] = "Yes"
three_se_stats = plot_over_grey(three_se, 'Encl osing_SE_Quadrant', 'Yes', '(3)')
```

Enclosing SE Quadrant (3)



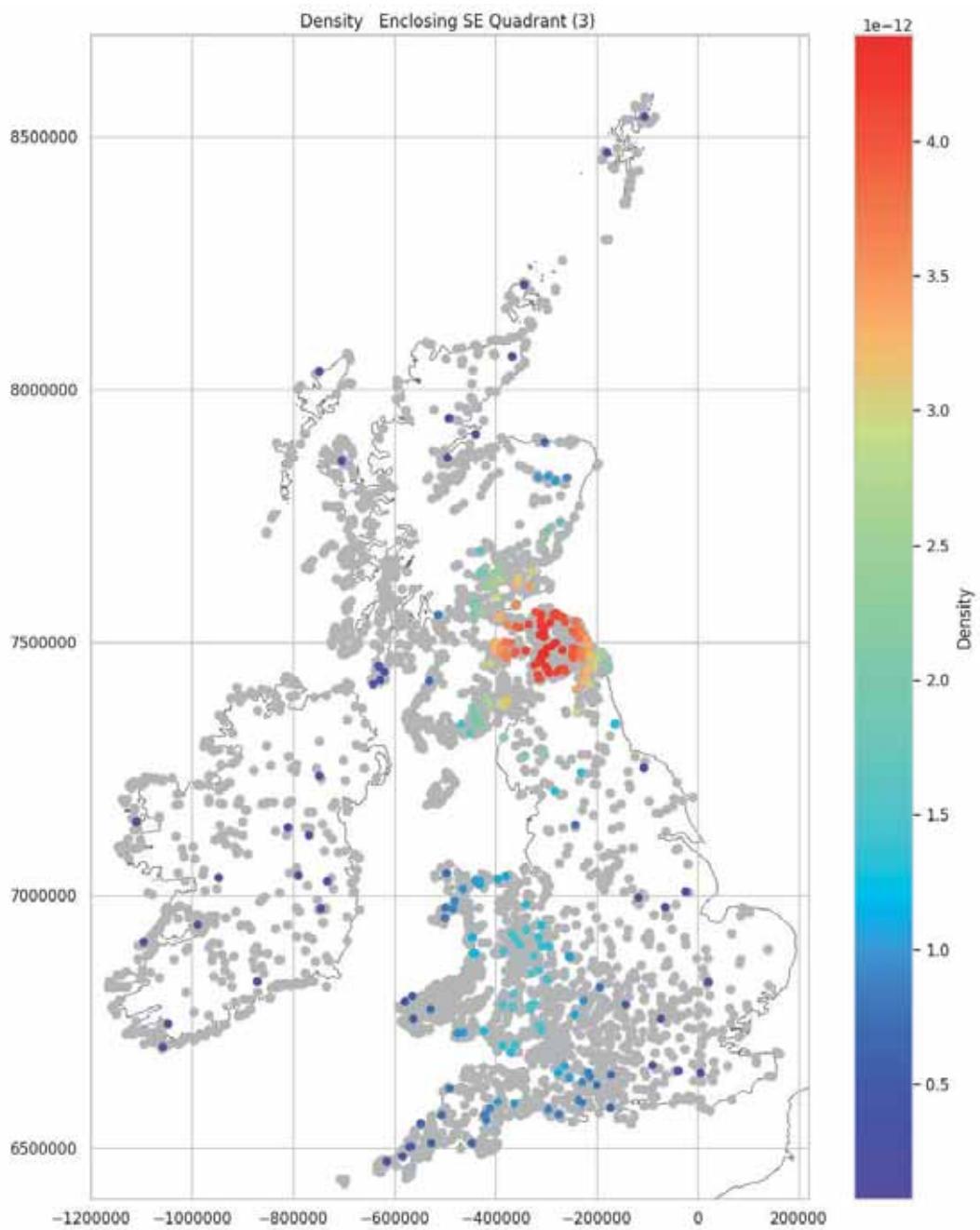
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 68%

SE Quadrant Data Density Mapped (3)

```
In [ ]: plot_density_over_grey(three_se_stats, 'Enclosing_SE_Quadrant (3)')
```



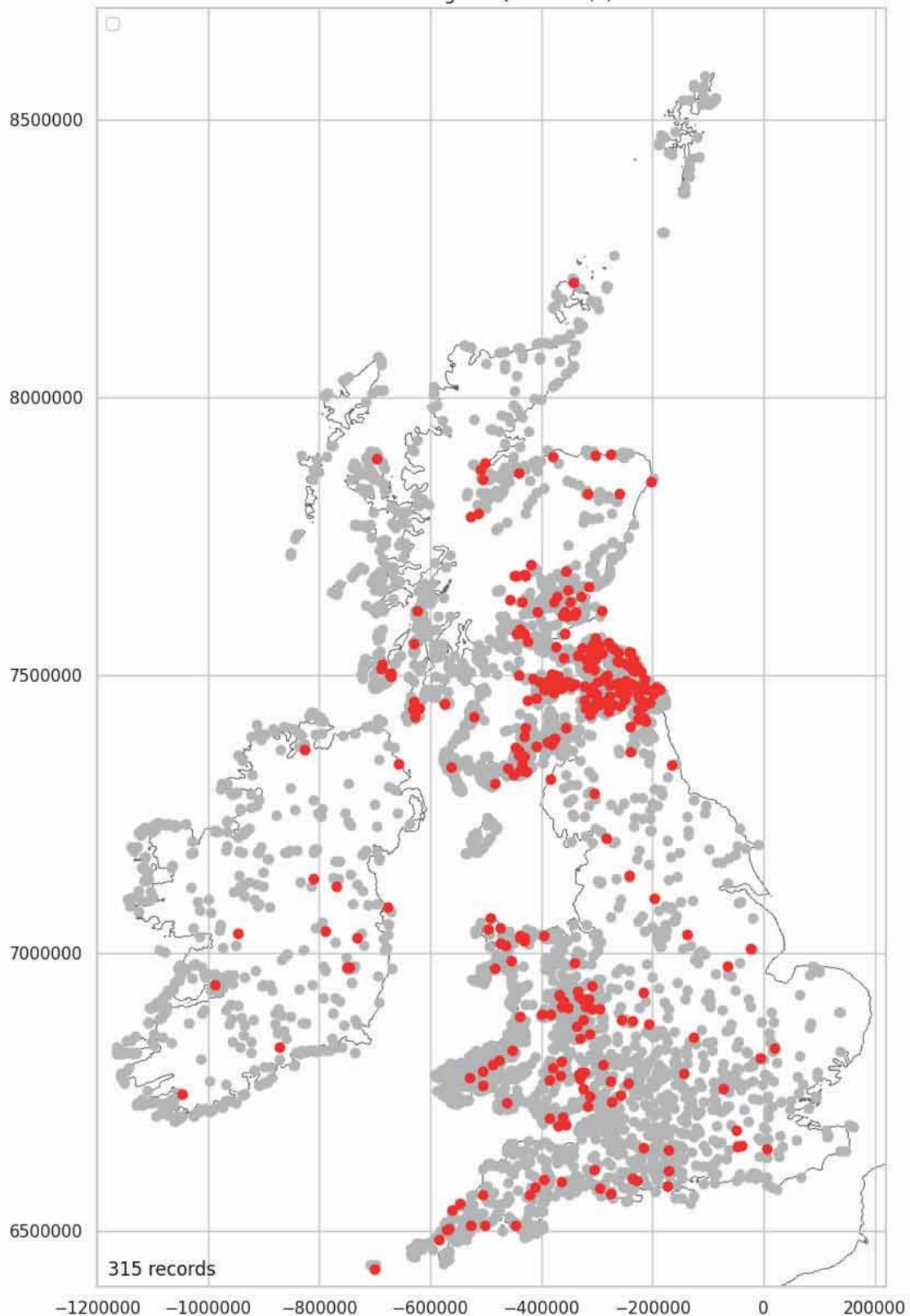
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

SW Quadrant Data Mapped (3)

```
In [ ]: three_sw = sw_quadrant_data[sw_quadrant_data['Encl osing_SW_Quadrant']==3].copy()
three_sw['Encl osing_SW_Quadrant'] = "Yes"
three_sw_stats = plot_over_grey(three_sw, 'Encl osing_SW_Quadrant', 'Yes', '(3)')
```

Enclosing SW Quadrant (3)



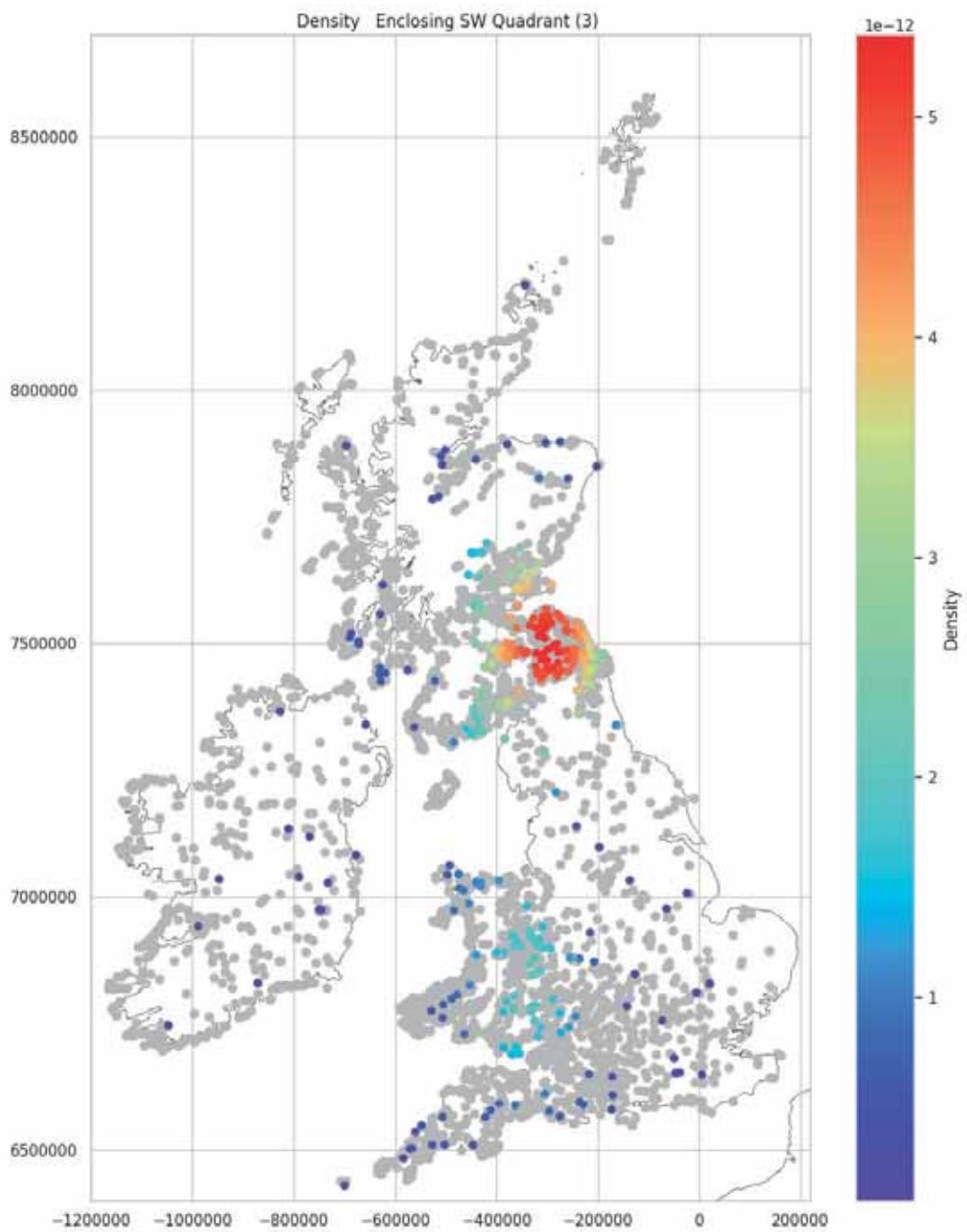
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

7.6%

SW Quadrant Data Density Mapped (3)

```
In [ ]: plot_density_over_grey(three_sw_stats, 'Enclosing_SW_Quadrant (3)')
```



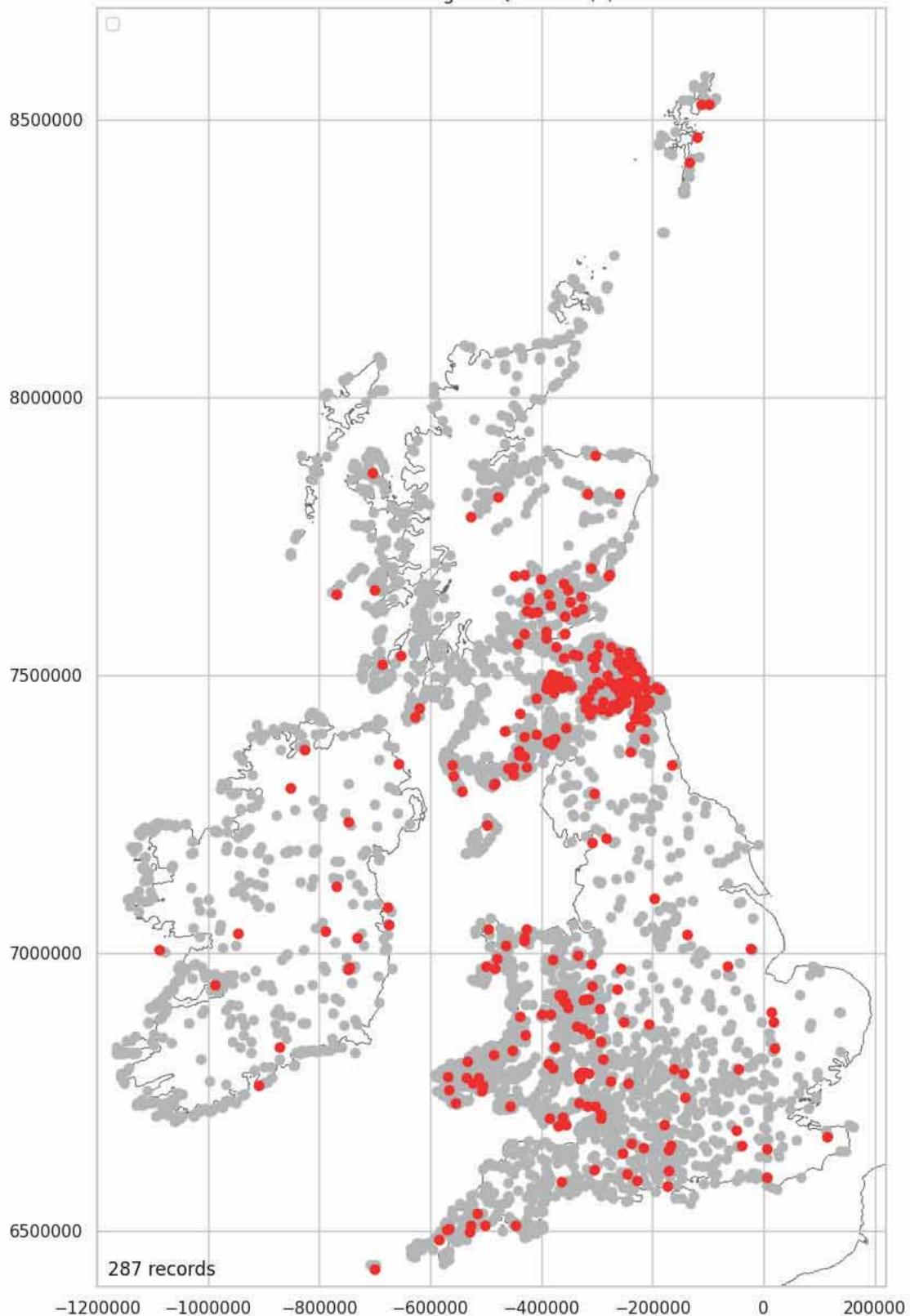
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

NW Quadrant Data Mapped (3)

```
In [ ]: three_nw = nw_quadrant_data[nw_quadrant_data['Encl osing_NW_Quadrant']==3].copy()
three_nw['Encl osing_NW_Quadrant'] = "Yes"
three_nw_stats = plot_over_grey(three_nw, 'Encl osing_NW_Quadrant', 'Yes', '(3)')
```

Enclosing NW Quadrant (3)



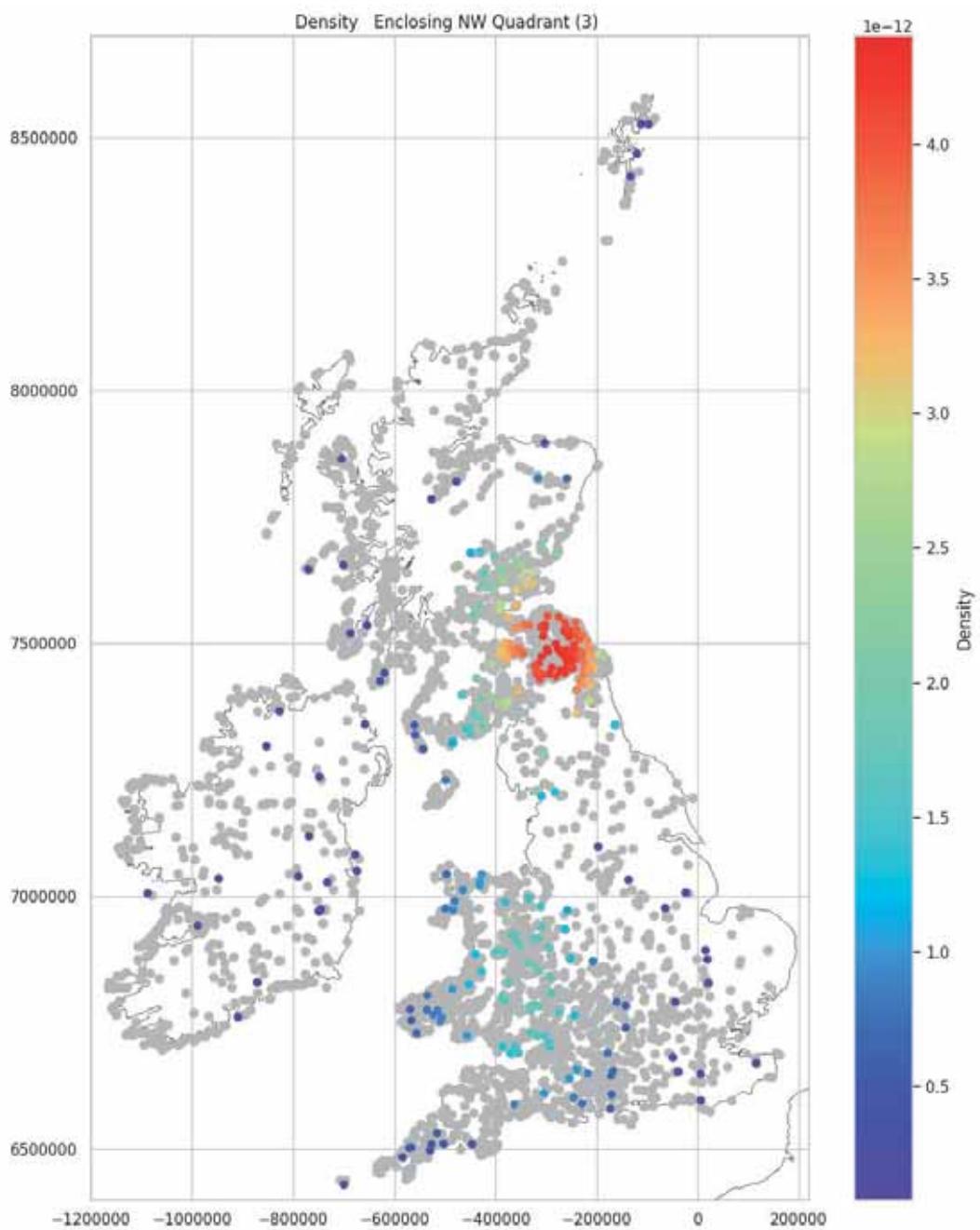
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 92%

NW Quadrant Data Density Mapped (3)

```
In [ ]: plot_density_over_grey(three_nw_stats, 'Enclosing_NW_Quadrant (3)')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

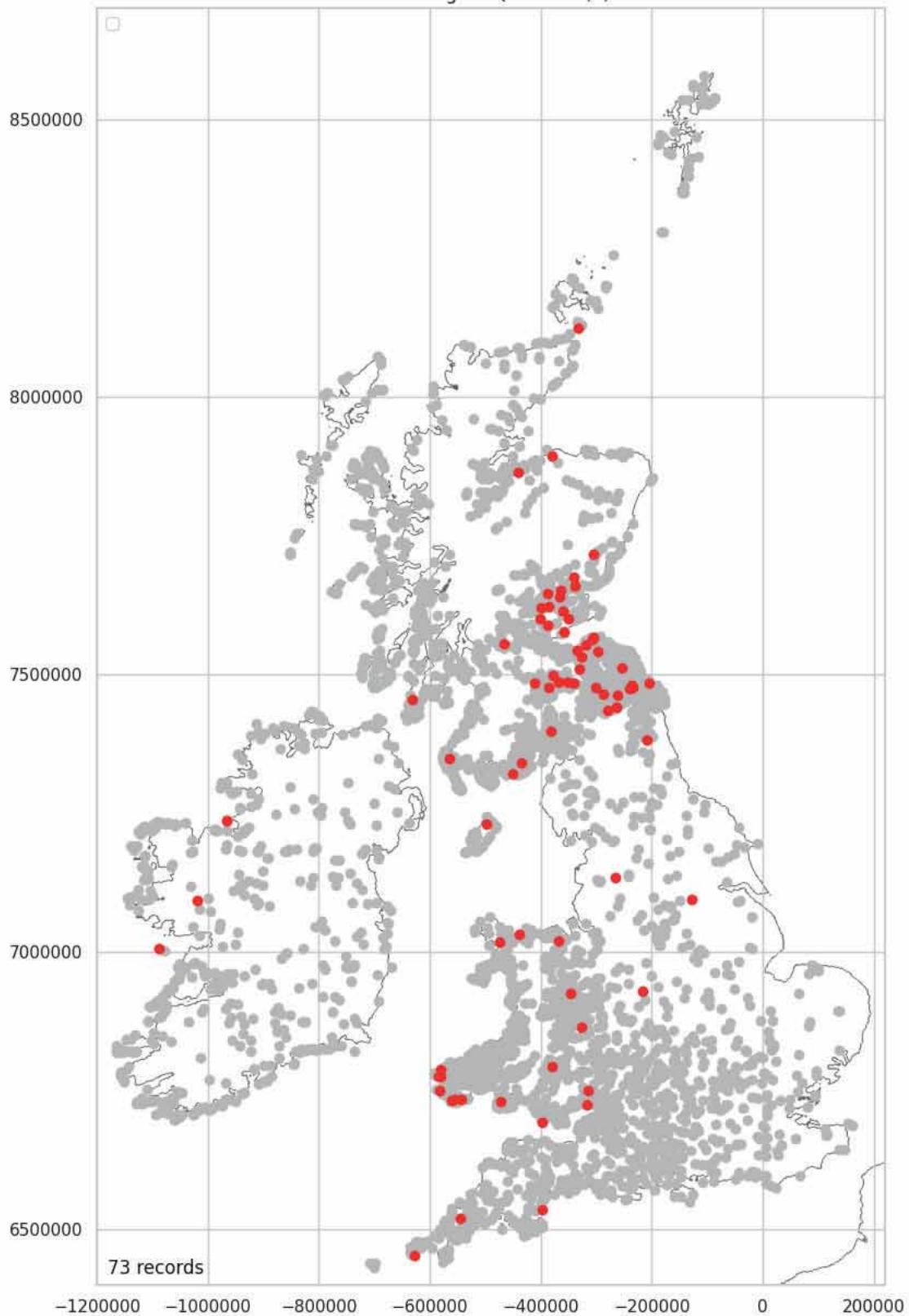
Quadrant Data Mapped (4)

As expected, the quadrant data mapping four ramparts is concentrated in the Northeast.

NE Quadrant Data Mapped (4)

```
In [ ]: four_ne = ne_quadrant_data[ne_quadrant_data['Encl osing_NE_Quadrant']==4].copy()
four_ne['Encl osing_NE_Quadrant'] = "Yes"
four_ne_stats = plot_over_grey(four_ne, 'Encl osing_NE_Quadrant', 'Yes', '(4)')
```

Enclosing NE Quadrant (4)



Middleton, M. 2024, Hillforts Primer

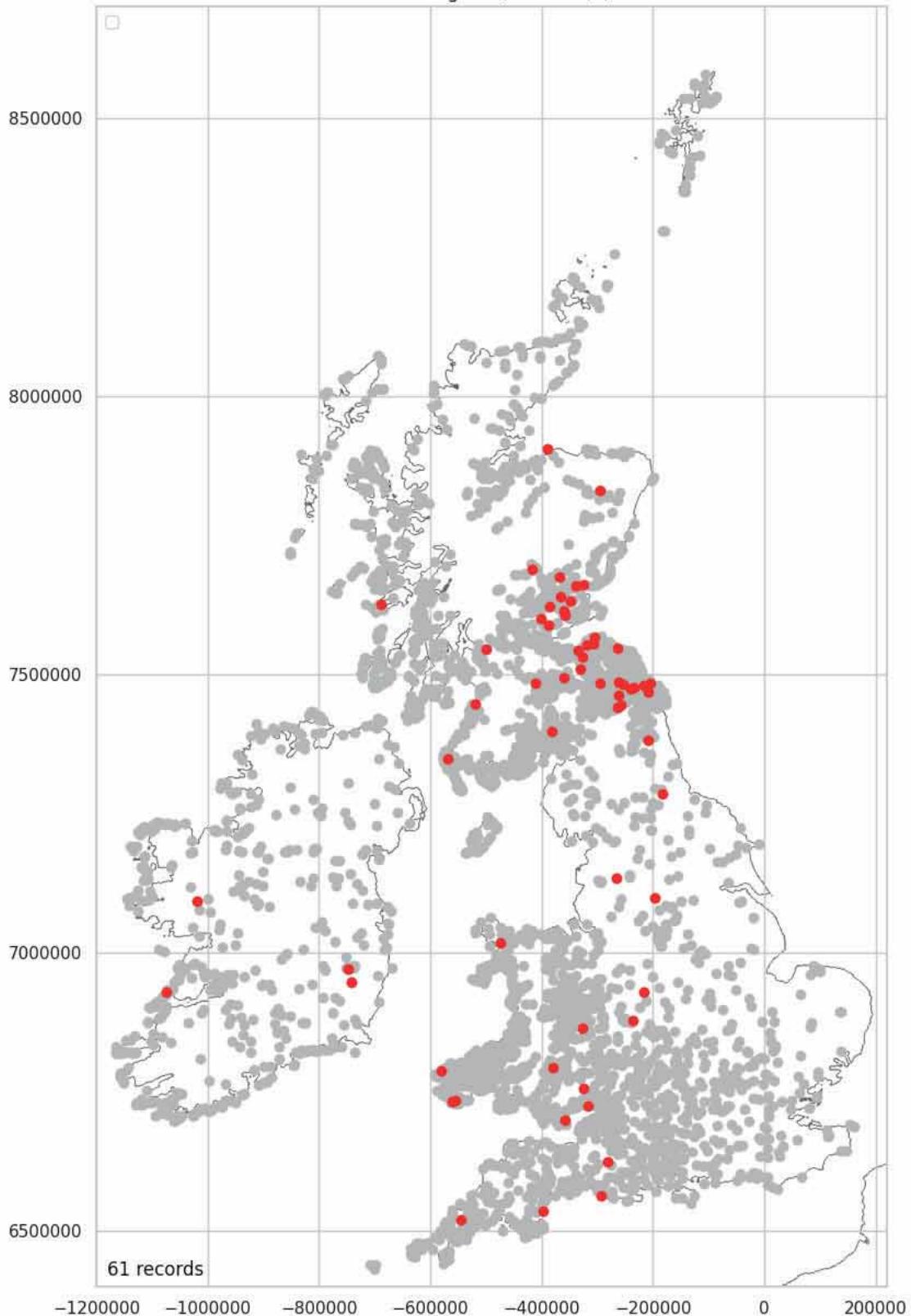
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 76%

SE Quadrant Data Mapped (4)

```
In [ ]: four_se = se_quadrant_data[se_quadrant_data['Encl osing_SE_Quadrant']==4].copy()
four_se['Encl osing_SE_Quadrant'] = "Yes"
four_se_stats = plot_over_grey(four_se, 'Encl osing_SE_Quadrant', 'Yes', '(4)')
```

Enclosing SE Quadrant (4)



Middleton, M. 2024, Hillforts Primer

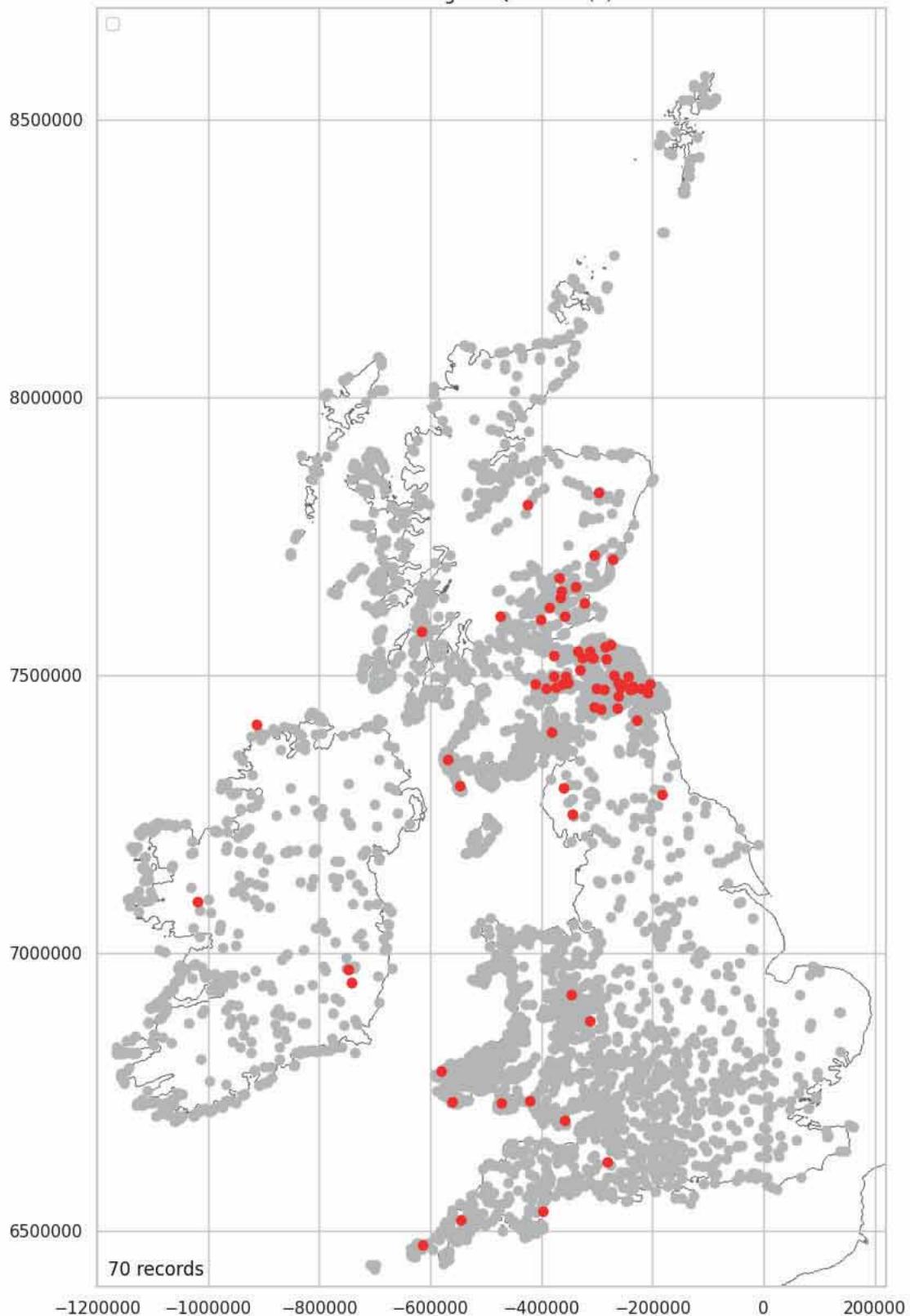
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 47%

SW Quadrant Data Mapped (4)

```
In [ ]: four_sw = sw_quadrant_data[sw_quadrant_data['Encl osing_SW_Quadrant']==4].copy()
four_sw['Encl osing_SW_Quadrant'] = "Yes"
four_sw_stats = plot_over_grey(four_sw, 'Encl osing_SW_Quadrant', 'Yes', '(4)')
```

Enclosing SW Quadrant (4)



Middleton, M. 2024, Hillforts Primer

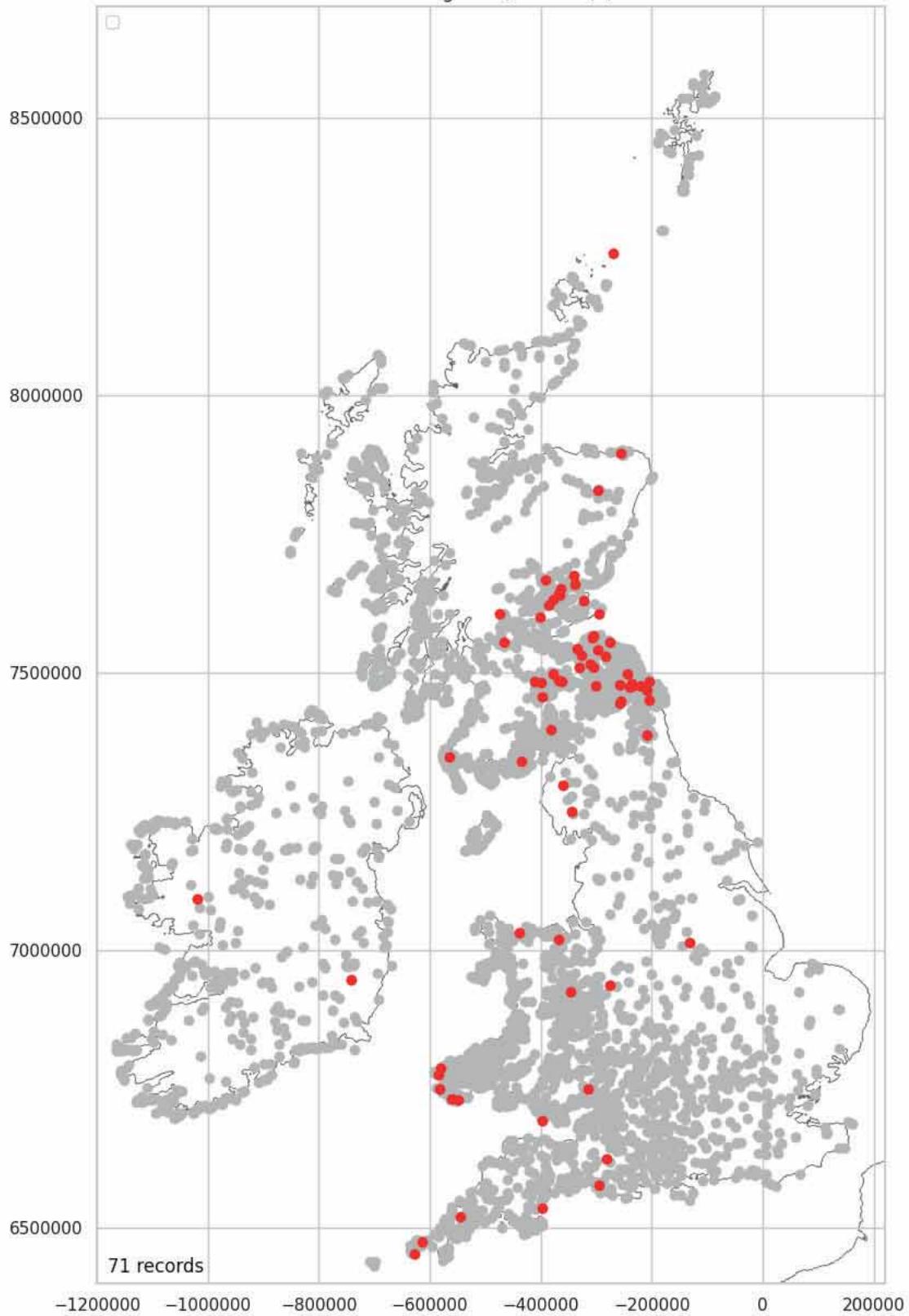
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 69%

NW Quadrant Data Mapped (4)

```
In [ ]: four_nw = nw_quadrant_data[nw_quadrant_data['Encl osing_NW_Quadrant']==4].copy()
four_nw['Encl osing_NW_Quadrant'] = "Yes"
four_nw_stats = plot_over_grey(four_nw, 'Encl osing_NW_Quadrant', 'Yes', '(4)')
```

Enclosing NW Quadrant (4)



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 71%

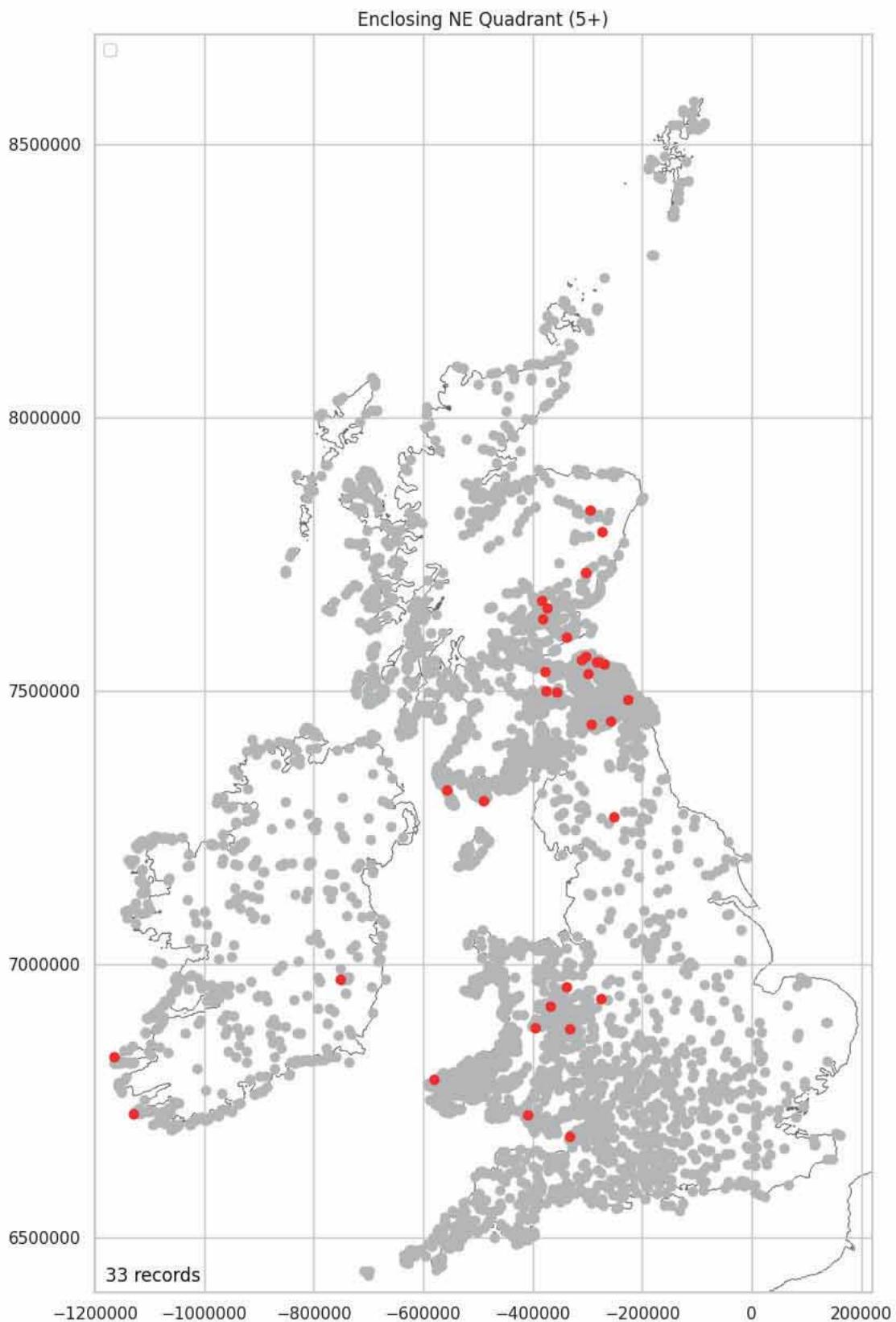
Quadrant Data Mapped (5+)

As expected, the quadrant data mapping five plus ramparts is concentrated in the Northeast.

NE Quadrant Data Mapped (5+)

```
In [ ]: outliers_ne = \
ne_quadrant_data[ne_quadrant_data['Enclosing_NE_Quadrant'] > 4].copy()
outliers_ne['Enclosing_NE_Quadrant'] = "Yes"
```

```
outliers_ne_stats = plot_over_grey(outliers_ne, 'Enclosing_NE_Quadrant', \
'Yes', '(5+)')
```



Middleton, M. 2024, Hillforts Primer

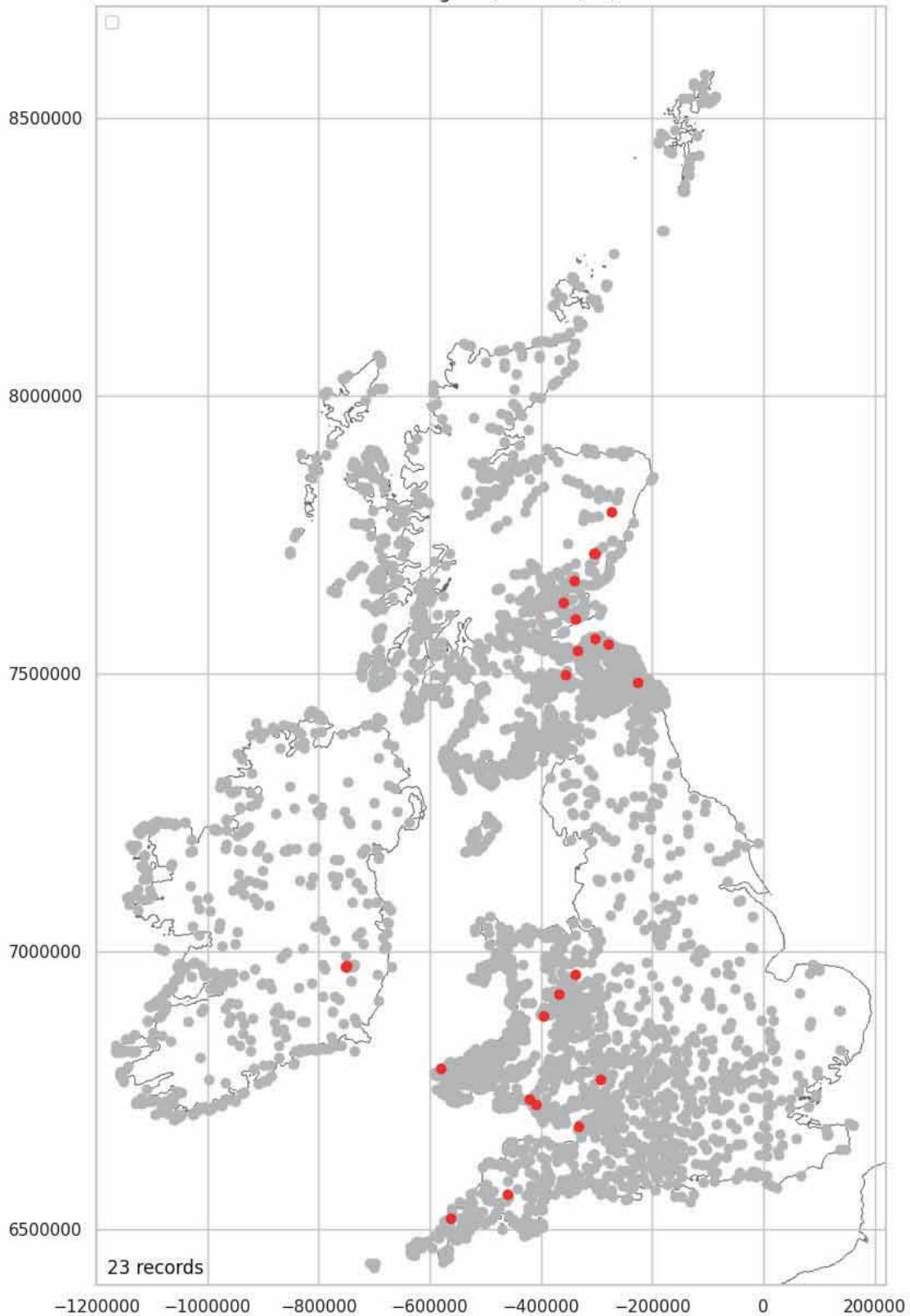
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.8%

SE Quadrant Data Mapped (5+)

```
In [ ]: outliers_se = \
se_quadrant_data[se_quadrant_data['Enclosing_SE_Quadrant']>4].copy()
outliers_se['Enclosing_SE_Quadrant'] = "Yes"
outliers_se_stats = plot_over_grey(outliers_se, 'Enclosing_SE_Quadrant', \
'Yes', '(5+)')
```

Enclosing SE Quadrant (5+)



Middleton, M. 2024, Hillforts Primer

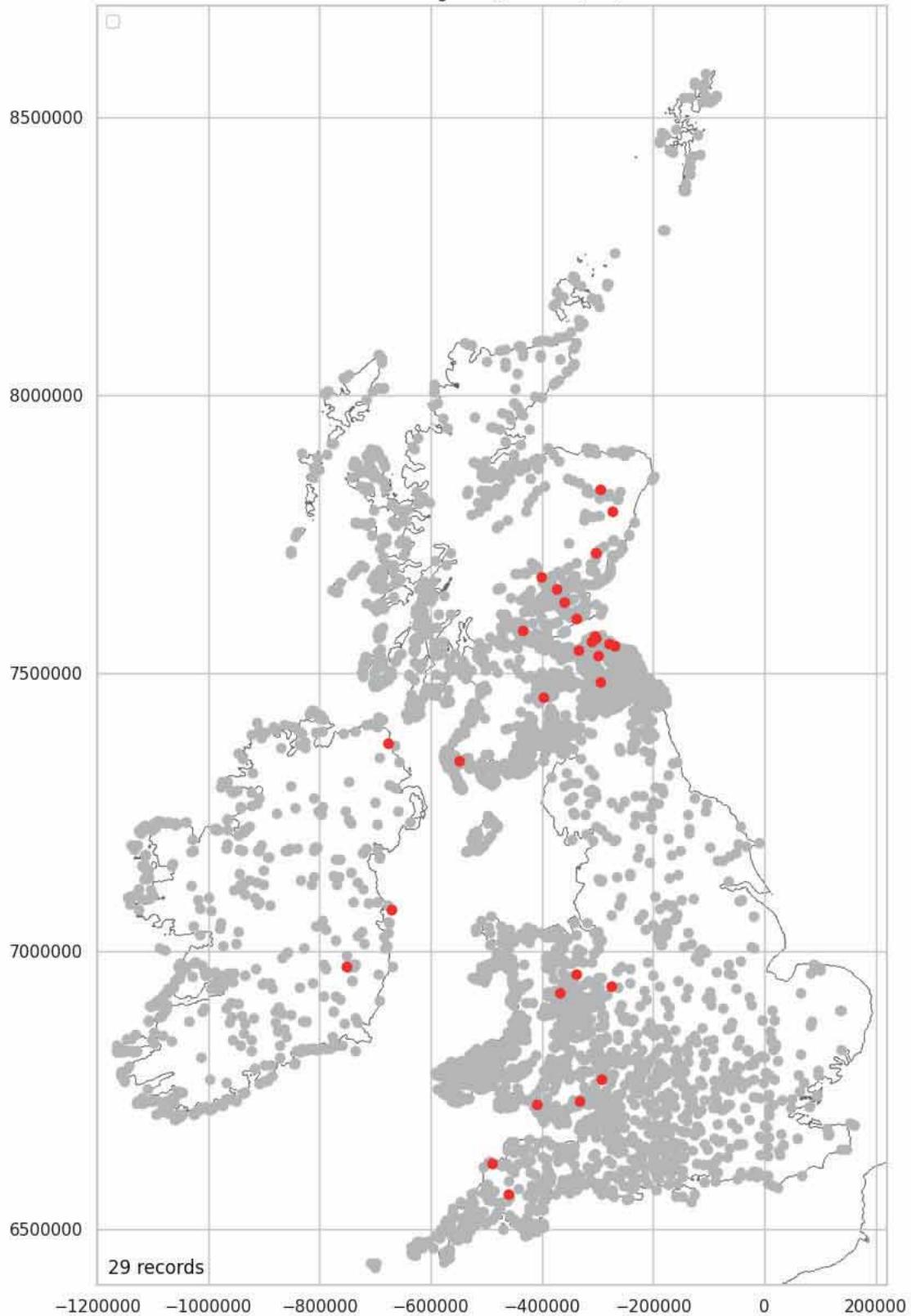
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.55%

SW Quadrant Data Mapped (5+)

```
In [ ]: outliers_sw = \
sw_quadrant_data[sw_quadrant_data['Encl osing_SW_Quadrant']>4].copy()
outliers_sw['Encl osing_SW_Quadrant'] = "Yes"
outliers_sw_stats = plot_over_grey(outliers_sw, 'Encl osing_SW_Quadrant', \
'Yes', '(5+)')
```

Enclosing SW Quadrant (5+)



Middleton, M. 2024, Hillforts Primer

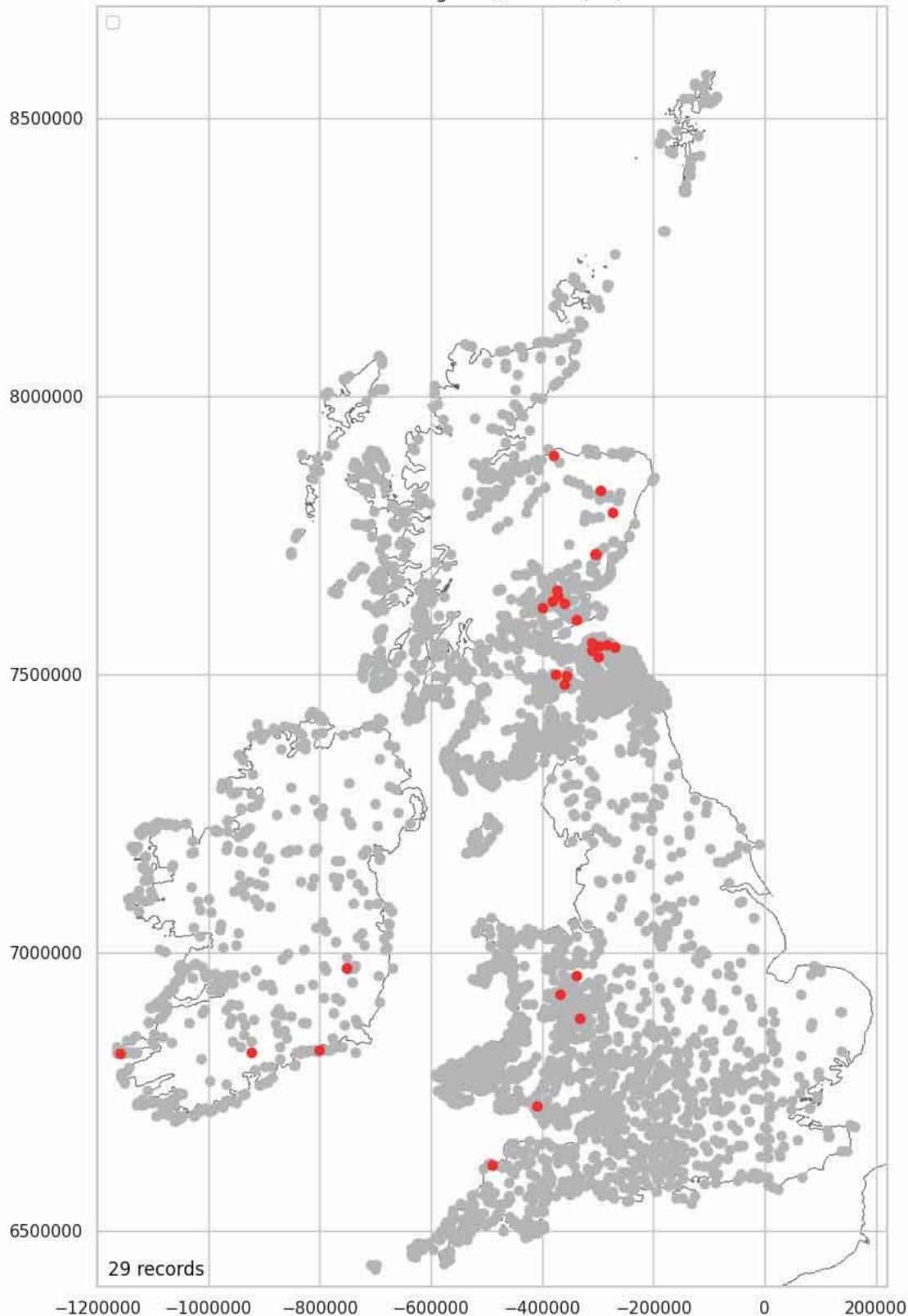
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.7%

NW Quadrant Data Mapped (5+)

```
In [ ]: outliers_nw = \
nw_quadrant_data[nw_quadrant_data['Encl osing_NW_Quadrant']>4].copy()
outliers_nw['Encl osing_NW_Quadrant'] = "Yes"
outliers_nw_stats = plot_over_grey(outliers_nw, 'Encl osing_NW_Quadrant', \
'Yes', '(5+)')
```

Enclosing NW Quadrant (5+)



Middleton, M. 2024, Hillforts Primer

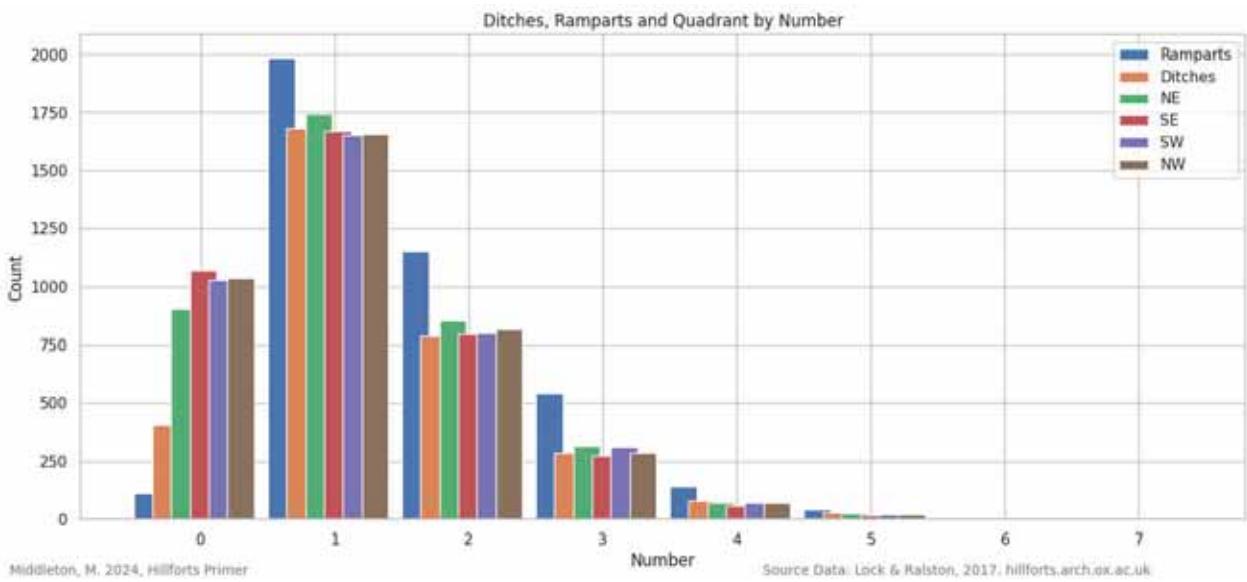
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.7%

Quadrant Data Plotted Against Ditches and Ramparts

As would be expected, the number of ramparts by quadrant roughly follows the distributions seen in the ramparts and ditches sections above.

```
In [ ]: plot_quadrants(all_ramparts, all_ditches, ne_quadrant_data,\n                     se_quadrant_data, sw_quadrant_data, nw_quadrant_data)
```



For the specific plots relating to ramparts and ditches see:

- Ramparts Plotted
- Ditches Plotted

```
In [ ]: ne_quadrant_data['Encl osing_NE_Quadrant'].value_counts().sort_index()
```

```
Out[ ]:
0.0    904
1.0   1745
2.0    855
3.0    317
4.0     73
5.0     28
6.0      3
7.0      1
8.0      1
Name: Encl osing_NE_Quadrant, dtype: int64
```

```
In [ ]: se_quadrant_data['Encl osing_SE_Quadrant'].value_counts().sort_index()
```

```
Out[ ]:
0.0   1072
1.0   1667
2.0    799
3.0    277
4.0     61
5.0     17
6.0     3
7.0     2
8.0     1
Name: Encl osing_SE_Quadrant, dtype: int64
```

```
In [ ]: sw_quadrant_data['Encl osing_SW_Quadrant'].value_counts().sort_index()
```

```
Out[ ]:
0.0   1030
1.0   1650
2.0    802
3.0    315
4.0     70
5.0     20
6.0     5
7.0     4
Name: Encl osing_SW_Quadrant, dtype: int64
```

```
In [ ]: nw_quadrant_data['Encl osing_NW_Quadrant'].value_counts().sort_index()
```

```
Out[ ]:
0.0   1040
1.0   1656
2.0    816
3.0    287
4.0     71
5.0     23
6.0     4
7.0     1
10.0    1
Name: Encl osing_NW_Quadrant, dtype: int64
```

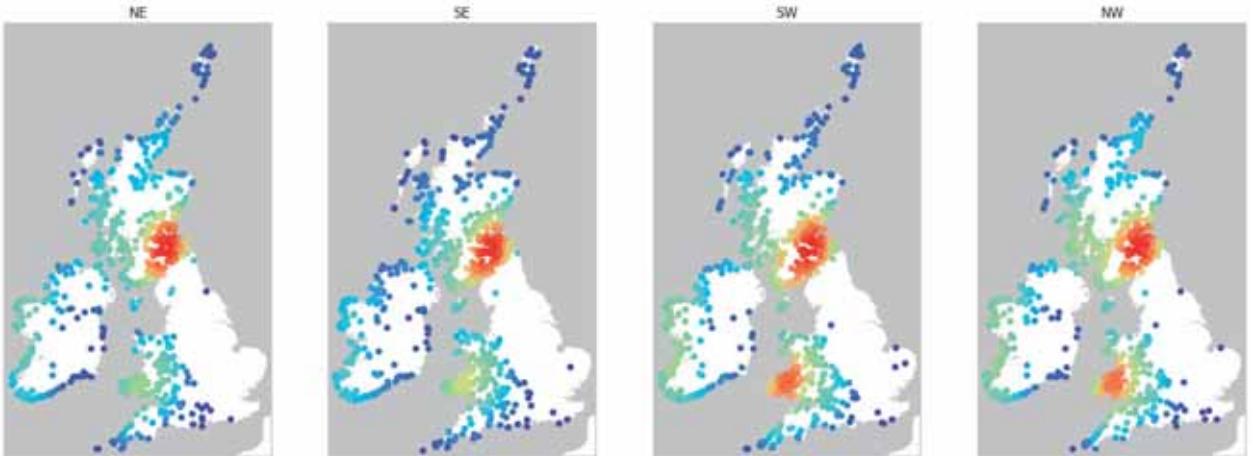
Quadrant Summary

Quadrant data is most influenced by local topography. This can be seen in Irish coastal forts, forts on the Pembrokeshire peninsula and the Northwestern hillforts all having less ramparts on their western, coastal sides. Other than this, large scale regional analysis

provides little additional insight beyond that already discussed for ramparts and ditches above.

```
In [ ]: plot_density_over_grey_four(zero_ne_stats, zero_se_stats, zero_sw_stats, \
zero_nw_stats, 'Quadrant 0')
```

Quadrant 0

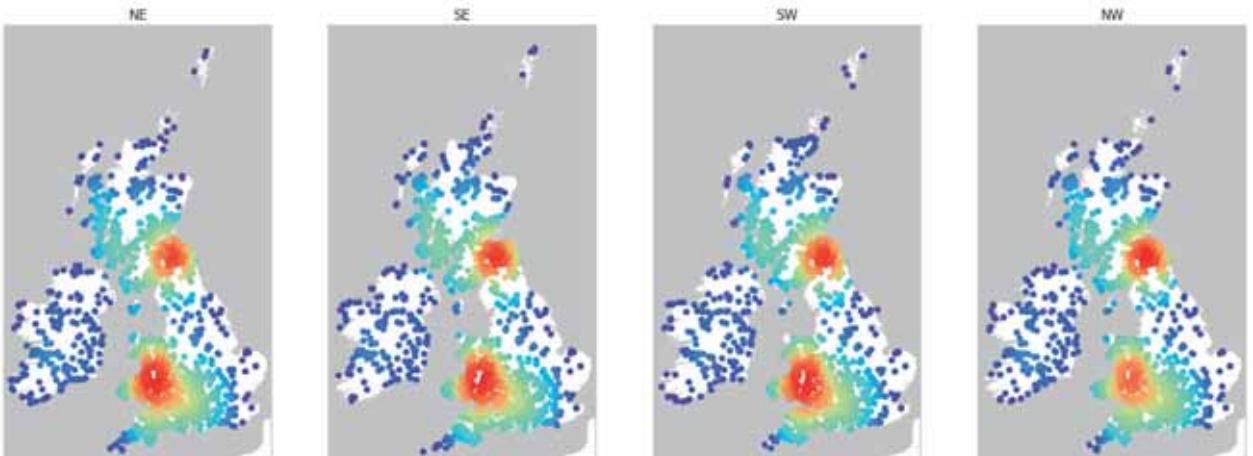


Huddleston, M. 2024, Hillforts Primrose

Source Data: Lock & Ralston, 2013, hillforts.arch.ox.ac.uk

```
In [ ]: plot_density_over_grey_four(one_ne_stats, one_se_stats, one_sw_stats, \
one_nw_stats, 'Quadrant 1')
```

Quadrant 1

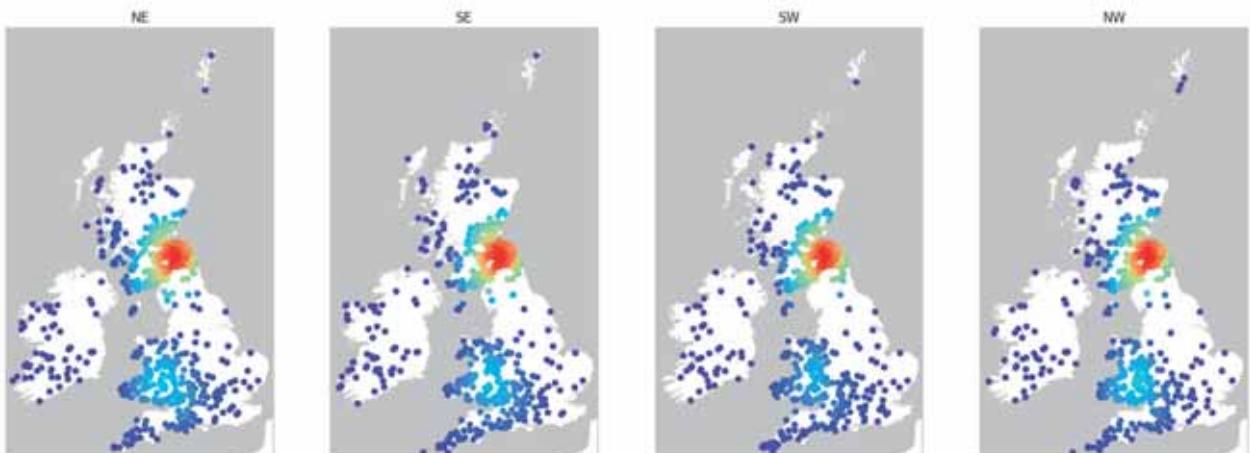


Huddleston, M. 2024, Hillforts Primrose

Source Data: Lock & Ralston, 2013, hillforts.arch.ox.ac.uk

```
In [ ]: plot_density_over_grey_four(two_ne_stats, two_se_stats, two_sw_stats, \
two_nw_stats, 'Quadrant 2')
```

Quadrant 2

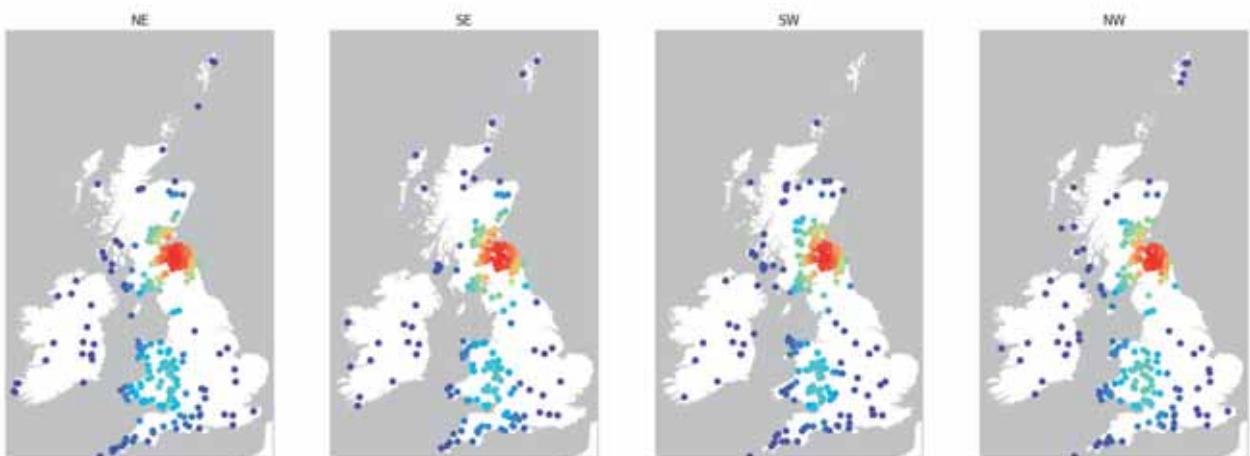


Huddleton, H. 2024, Hillforts Prinsen

Source Data: Lock & Ralestan, 2017; hillforts.arch.ox.ac.uk

```
In [ ]: plot_densitiy_over_grey_four(three_ne_stats, three_se_stats, three_sw_stats, \
three_nw_stats, 'Quadrant 3')
```

Quadrant 3



Huddleton, H. 2024, Hillforts Prinsen

Source Data: Lock & Ralestan, 2017; hillforts.arch.ox.ac.uk

Enclosing Text Data

There are eight Enclosing text features. All contain null values.

```
In [ ]: enclosing_text_features = [
    'Enclosing_Summary',
    'Enclosing_Multiperiod_Comments',
    'Enclosing_Circumt_Comments',
    'Enclosing_Quadrant_Comments',
    'Enclosing_Surface_Comments',
    'Enclosing_Exca_vation_Comments',
    'Enclosing_Gang_Working_Comments',
    'Enclosing_Ditches_Comments']
```

```
enclosing_text_data = enclosing_data[enclosing_text_features].copy()
enclosing_text_data.head()
```

	Enclosing_Summary	Enclosing_Multiperiod_Comments	Enclosing_Circuit_Comments	Enclosing_Quadrant_Comments	Enclosing_Surface_Co
0	Univallate hillfort with complete circuit, but...	Univallate hillfort with complete circuit.	Single rampart continues around circuit.	NaN	Little surface evi features and
1	Defined differentially by single rampart to 5....	NaN	The ramparts are irregular which makes assessm...	NaN	Bank possibly Counterscarp bank
2	Three ramparts and ditches on the N. Although ...	NaN	Ramparts damaged and discontinuous. On the W t...	NaN	
3	Steep natural scarp artificially scarped with ...	Area not exact.	The ramparts are slight but complete the circuit.	NaN	Possible earthen bar berm to 7.
4	In Phase I, c. 3ha were enclosed by a slight b...	The site is very long and sinuous. Phased cons...	The ramparts of the Phase II overall enclosure...	NaN	Surface evidence of an bank and

In []: `encl osi ng_text_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Enclosing_Summary    4138 non-null  object  
 1   Enclosing_Multiperiod_Comments 1016 non-null  object  
 2   Enclosing_Circuit_Comments 1201 non-null  object  
 3   Enclosing_Quadrant_Comments 58 non-null  object  
 4   Enclosing_Surface_Comments 1236 non-null  object  
 5   Enclosing_Excavation_Comments 526 non-null  object  
 6   Enclosing_Gang_Working_Comments 48 non-null  object  
 7   Enclosing_Ditches_Comments 1499 non-null  object  
dtypes: object(8)
memory usage: 259.3+ KB
```

Entrance Text Data - Resolve Null Values

Test for 'NA'.

In []: `test_cat_list_for_NA(encl osi ng_text_data, encl osi ng_text_features)`

```
Enclosing_Summary 0
Enclosing_Multiperiod_Comments 0
Enclosing_Circuit_Comments 0
Enclosing_Quadrant_Comments 0
Enclosing_Surface_Comments 0
Enclosing_Excavation_Comments 0
Enclosing_Gang_Working_Comments 0
Enclosing_Ditches_Comments 0
```

Fill null values with 'NA'.

In []: `encl osi ng_text_data = \
update_cat_list_for_NA(encl osi ng_text_data, encl osi ng_text_features)
encl osi ng_text_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Enclosing_Summary    4147 non-null  object  
 1   Enclosing_Multiperiod_Comments 4147 non-null  object  
 2   Enclosing_Circuit_Comments 4147 non-null  object  
 3   Enclosing_Quadrant_Comments 4147 non-null  object  
 4   Enclosing_Surface_Comments 4147 non-null  object  
 5   Enclosing_Excavation_Comments 4147 non-null  object  
 6   Enclosing_Gang_Working_Comments 4147 non-null  object  
 7   Enclosing_Ditches_Comments 4147 non-null  object  
dtypes: object(8)
memory usage: 259.3+ KB
```

Enclosing Encodable Data

There are 44 Enclosing encodable features. Non contain null values.

```
In [ ]: encl osi ng_encodeable_features = [
    'Encl osi ng_Mul ti peri od',
    'Encl osi ng_Circui t',
    'Encl osi ng_Current_Part_Uni',
    'Encl osi ng_Current_Uni',
    'Encl osi ng_Current_Part_Bi',
    'Encl osi ng_Current_Bi',
    'Encl osi ng_Current_Part_Mul ti',
    'Encl osi ng_Current_Mul ti',
    'Encl osi ng_Current_Uncertain',
    'Encl osi ng_Period_Part_Uni',
    'Encl osi ng_Period_Uni',
    'Encl osi ng_Period_Part_Bi',
    'Encl osi ng_Period_Bi',
    'Encl osi ng_Period_Part_Mul ti',
    'Encl osi ng_Period_Mul ti',
    'Encl osi ng_Surface_None',
    'Encl osi ng_Surface_Bank',
    'Encl osi ng_Surface_Wall',
    'Encl osi ng_Surface_Rubble',
    'Encl osi ng_Surface_Walk',
    'Encl osi ng_Surface_Timber',
    'Encl osi ng_Surface_Vitrification',
    'Encl osi ng_Surface_Burning',
    'Encl osi ng_Surface_Pal isade',
    'Encl osi ng_Surface_Counter_Scarp',
    'Encl osi ng_Surface_Berm',
    'Encl osi ng_Surface_Unfinished',
    'Encl osi ng_Surface_Other',
    'Encl osi ng_Excavation_Nothing',
    'Encl osi ng_Excavation_Bank',
    'Encl osi ng_Excavation_Wall',
    'Encl osi ng_Excavation_Murus',
    'Encl osi ng_Excavation_Timber_Framed',
    'Encl osi ng_Excavation_Timber_Laced',
    'Encl osi ng_Excavation_Vitrification',
    'Encl osi ng_Excavation_Burning',
    'Encl osi ng_Excavation_Pal isade',
    'Encl osi ng_Excavation_Counter_Scarp',
    'Encl osi ng_Excavation_Berm',
    'Encl osi ng_Excavation_Unfinished',
    'Encl osi ng_Excavation_No_Known',
    'Encl osi ng_Excavation_Other',
    'Encl osi ng_Gang_Working',
    'Encl osi ng_Ditches' ]
```

```
encl osi ng_encodeable_data = encl osi ng_data[encl osi ng_encodeable_features].copy()
encl osi ng_encodeable_data.head()
```

	Enclosing_Multiperiod	Enclosing_Circuit	Enclosing_Current_Part_Uni	Enclosing_Current_Uni	Enclosing_Current_Part_Bi	Enclosing_Current_Uncertain	
0	No	Yes		No	Yes	No	N
1	No	Yes		No	Yes	No	N
2	No	No		Yes	No	Yes	N
3	No	Yes		No	Yes	No	N
4	Yes	Yes		No	No	No	Y

```
In [ ]: encl osi ng_encodeable_data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 44 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Enclosing_Multiperiod    4147 non-null object 
 1   Enclosing_Circuit        4147 non-null object 
 2   Enclosing_Current_Part_Uni 4147 non-null object 
 3   Enclosing_Current_Uni    4147 non-null object 
 4   Enclosing_Current_Part_Bi 4147 non-null object 
 5   Enclosing_Current_Bi     4147 non-null object 
 6   Enclosing_Current_Part_Multi 4147 non-null object 
 7   Enclosing_Current_Multi   4147 non-null object 
 8   Enclosing_Current_Uncertain 4147 non-null object 
 9   Enclosing_Period_Part_Uni 4147 non-null object 
 10  Enclosing_Period_Uni     4147 non-null object 
 11  Enclosing_Period_Part_Bi 4147 non-null object 
 12  Enclosing_Period_Bi      4147 non-null object 
 13  Enclosing_Period_Part_Multi 4147 non-null object 
 14  Enclosing_Period_Multi   4147 non-null object 
 15  Enclosing_Surface_None   4147 non-null object 
 16  Enclosing_Surface_Bank   4147 non-null object 
 17  Enclosing_Surface_Wall   4147 non-null object 
 18  Enclosing_Surface_Rubble 4147 non-null object 
 19  Enclosing_Surface_Walk   4147 non-null object 
 20  Enclosing_Surface_Timber 4147 non-null object 
 21  Enclosing_Surface_Vitification 4147 non-null object 
 22  Enclosing_Surface_Burning 4147 non-null object 
 23  Enclosing_Surface_Paisade 4147 non-null object 
 24  Enclosing_Surface_Counter_Scarp 4147 non-null object 
 25  Enclosing_Surface_Berm    4147 non-null object 
 26  Enclosing_Surface_Unfinished 4147 non-null object 
 27  Enclosing_Surface_Other   4147 non-null object 
 28  Enclosing_Excavation_Nothing 4147 non-null object 
 29  Enclosing_Excavation_Bank 4147 non-null object 
 30  Enclosing_Excavation_Wall 4147 non-null object 
 31  Enclosing_Excavation_Murus 4147 non-null object 
 32  Enclosing_Excavation_Timber_Framed 4147 non-null object 
 33  Enclosing_Excavation_Timber_Laced 4147 non-null object 
 34  Enclosing_Excavation_Vitification 4147 non-null object 
 35  Enclosing_Excavation_Burning 4147 non-null object 
 36  Enclosing_Excavation_Paisade 4147 non-null object 
 37  Enclosing_Excavation_Counter_Scarp 4147 non-null object 
 38  Enclosing_Excavation_Berm    4147 non-null object 
 39  Enclosing_Excavation_Unfinished 4147 non-null object 
 40  Enclosing_Excavation_No_Known 4147 non-null object 
 41  Enclosing_Excavation_Other   4147 non-null object 
 42  Enclosing_Gang_Working    4147 non-null object 
 43  Enclosing_Ditches         4147 non-null object 

dtypes: object(44)
memory usage: 1.4+ MB

```

Enclosing Multiperiod

528 hillforts (12.73%) are recorded as being multiperiod.

```
In [ ]: multiperiod_counts = \
enclosiing_encodeable_data['Enclosing_Multiperiod'].value_counts()
multiperiod_counts
```

```
Out[ ]: No      3619
Yes      528
Name: Enclosing_Multiperiod, dtype: int64
```

```
In [ ]: round(multiperiod_counts[1]/len(enclosiing_encodeable_data)*100, 2)
```

```
Out[ ]: 12.73
```

```
In [ ]: location_enclosiing_encodeable_data = \
pd.merge(location_numeric_data_short, enclosiing_encodeable_data, \
left_index=True, right_index=True)
```

```
In [ ]: location_enclosiing_encodeable_data_ne = \
pd.merge(north_east.reset_index(), enclosiing_encodeable_data, \
left_on='uid', right_index=True)
location_enclosiing_encodeable_data_ne = \
pd.merge(name_and_number, location_enclosiing_encodeable_data_ne, \
left_index=True, right_on='uid')
```

```
In [ ]: location_enclosiing_encodeable_data_nw = \
pd.merge(north_west.reset_index(), enclosiing_encodeable_data, \
left_on='uid', right_index=True)
location_enclosiing_encodeable_data_nw = \
```

```

pd.merge(name_and_number, location_enclosing_encodeable_data_nw, \
         left_index=True, right_on='uid')

In [ ]: location_enclosing_encodeable_data_ireland_n = \
pd.merge(north_irland.reset_index(), enclosing_encodeable_data, \
         left_on='uid', right_index=True)
location_enclosing_encodeable_data_ireland_n = \
pd.merge(name_and_number, location_enclosing_encodeable_data_ireland_n, \
         left_index=True, right_on='uid')

In [ ]: location_enclosing_encodeable_data_ireland_s = \
pd.merge(south_irland.reset_index(), enclosing_encodeable_data, \
         left_on='uid', right_index=True)
location_enclosing_encodeable_data_ireland_s = \
pd.merge(name_and_number, location_enclosing_encodeable_data_ireland_s, \
         left_index=True, right_on='uid')

In [ ]: location_enclosing_encodeable_data_south = \
pd.merge(south, enclosing_encodeable_data, left_on='uid', right_index=True)
location_enclosing_encodeable_data_south = \
pd.merge(name_and_number, location_enclosing_encodeable_data_south, \
         left_index=True, right_on='uid')

```

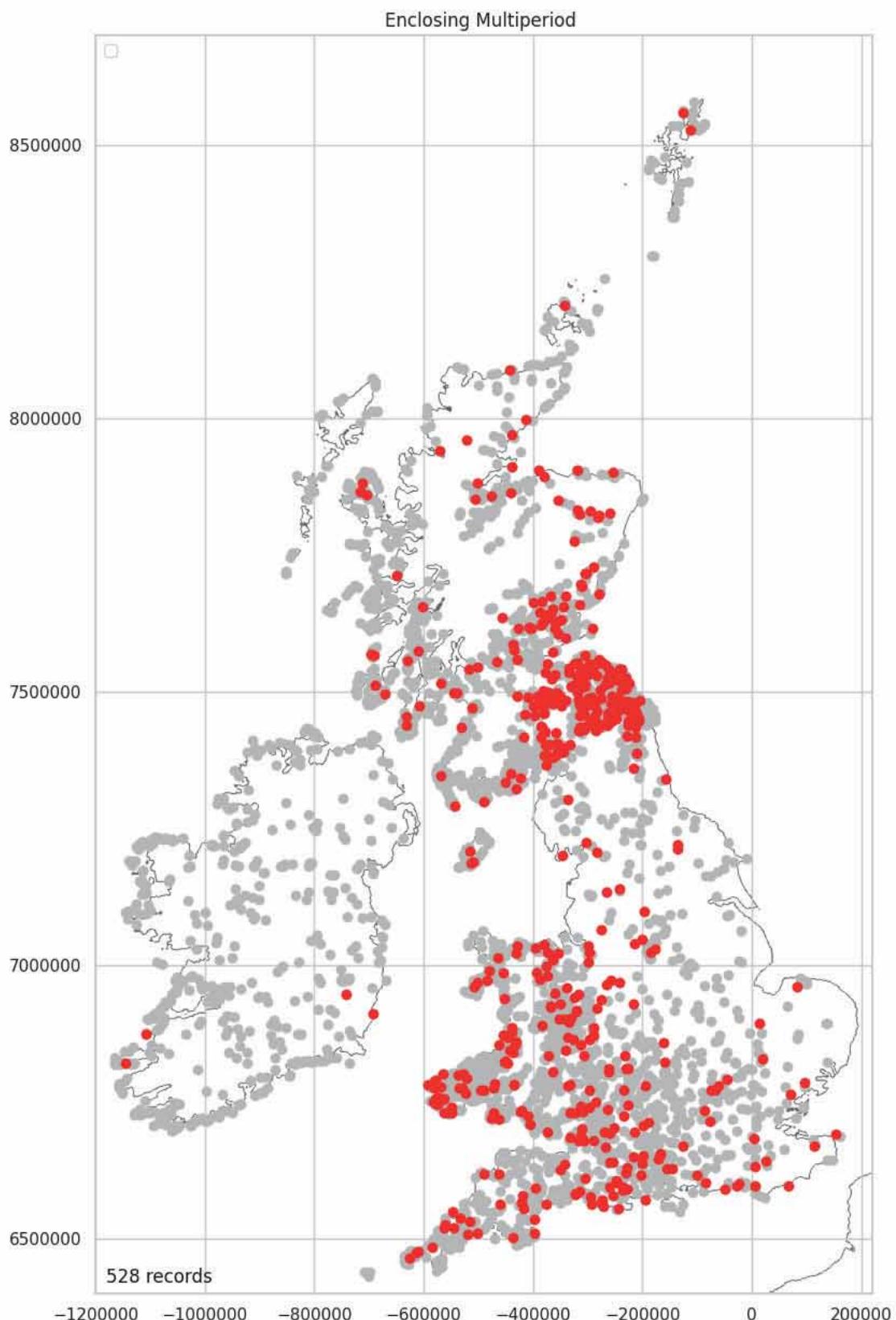
Enclosing Multiperiod Mapped

There is an obvious recording bias in this data over Ireland. Having seen the spread of dating information – with the main phases of occupation being from 800 BC to AD400 (See: Part 3: Dating Data) – it is unlikely that only 12.73% of all forts have multiperiod occupation so, there is most likely, a recording bias across this entire class.

```

multiperiod_data = \
plot_over_grey(location_enclosing_encodeable_data, 'Enclosing_Multiperiod', \
               'Yes', '')

```



Middleton, M. 2024, Hillforts Primer

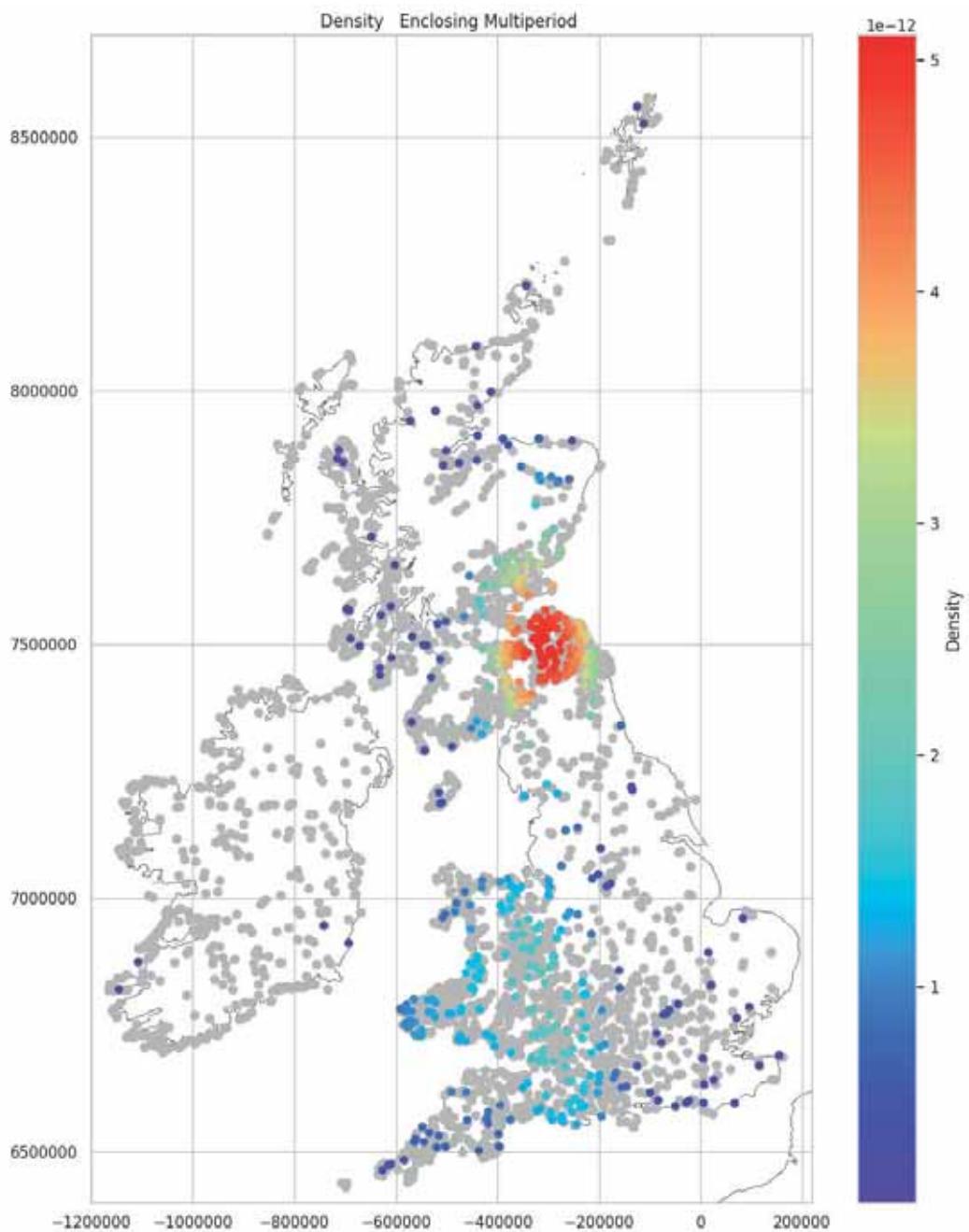
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

12.73%

Enclosing Multiperiod Density Mapped

Hillforts recorded as multiperiod cluster most intensely in the Northeast. There is a secondary cluster to the east of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(multiperiod_data, 'Enclosing_Multiperiod')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Circuit Mapped

There are 1891 (45.6%) of hillforts identified as having an Enclosing Circuit. It is assumed that Enclosing Circuit refers to hillforts having ramparts that form a completely enclosed ring.

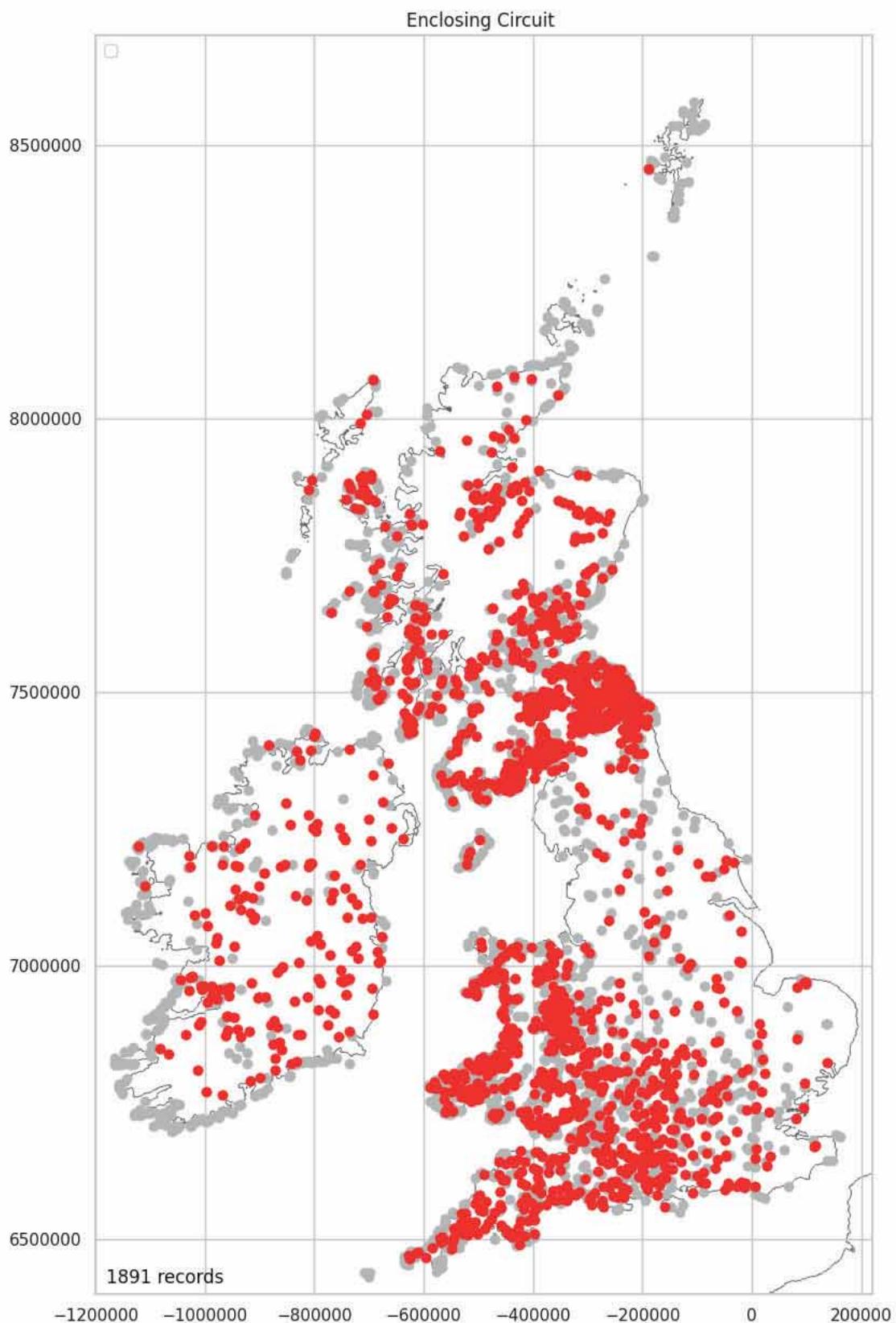
```
In [ ]: circui_t_counts = enclosing_encodeable_data['Enclosing_Circuit'].value_counts()
```

```
Out[ ]: No      226
Yes     1891
Name: Enclosing_Circuit, dtype: int64
```

```
In [ ]: print(f'{round(circui_t_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
```

45.6%

```
In [ ]: circui_t_data_yes = \
plot_over_grey(location_enclosing_encodeable_data, 'Enclosing_Circuit', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

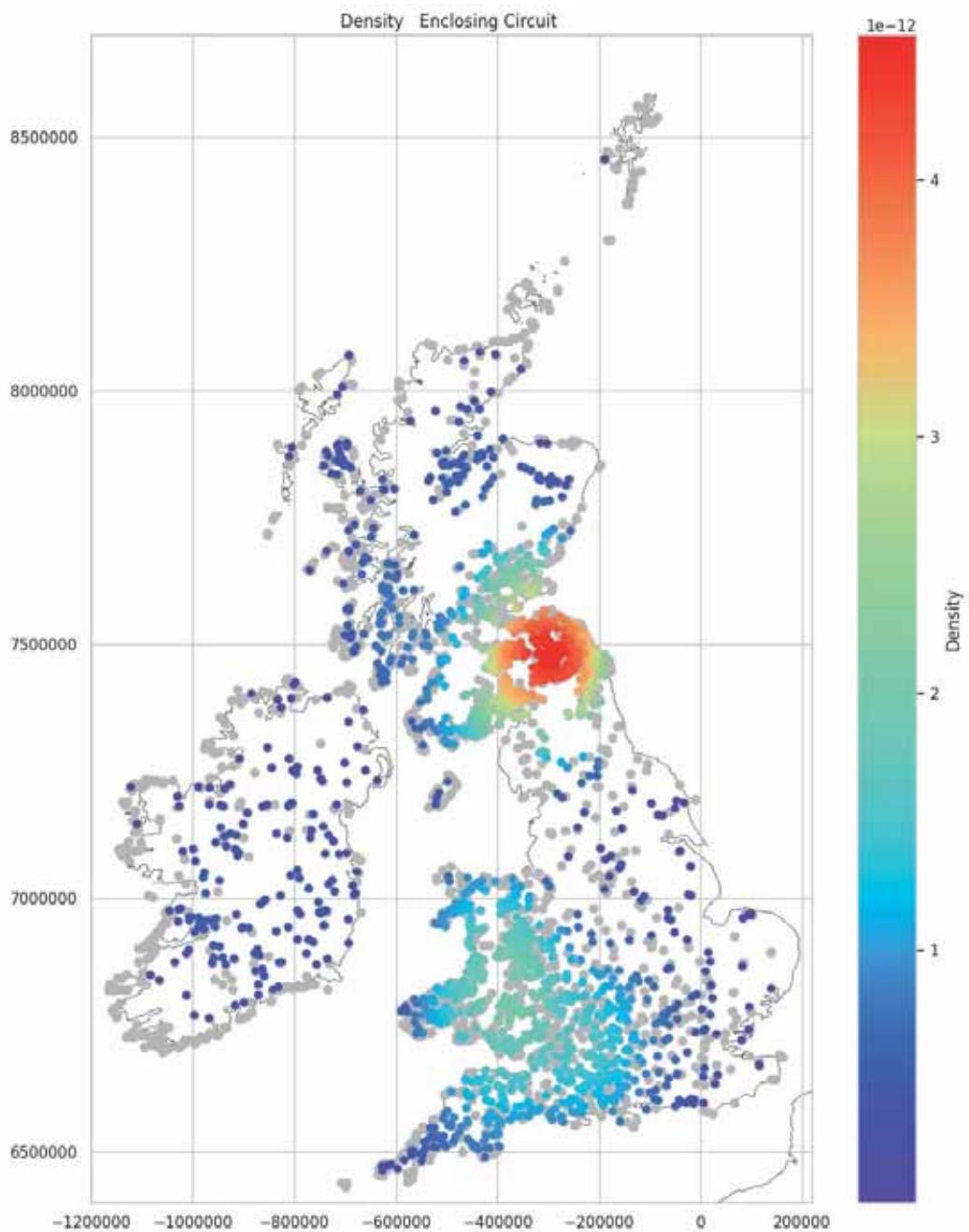
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

45.6%

Enclosing Circuit Density Mapped

The distribution is noticeably concentrated over the inland forts. There are two main concentrations, a strong cluster over the Northeast and a more subtle cluster to the east of the Cambrian Mountains. Unsurprisingly, coastal forts are less likely to have a fully enclosed rampart as they tend to incorporate naturally defensive features, such as cliffs, into their layout.

```
In [ ]: plot_density_over_grey(circum_data_yes, 'Enclosing_Circuit')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Current Part Univallate Mapped

1628 (39.26%) of hillforts are identified as Current Part Univallate. The distribution is relatively even across the atlas.

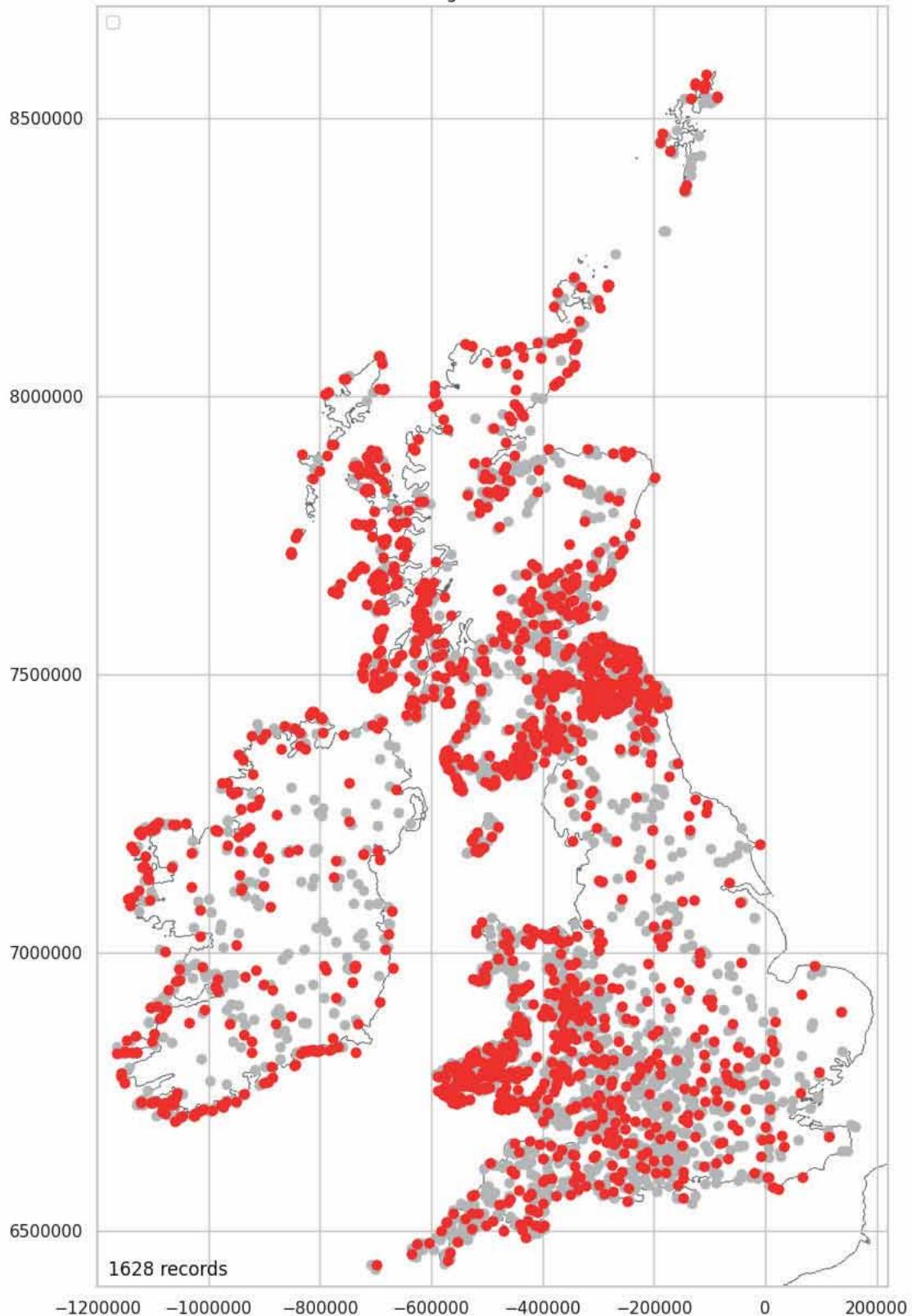
```
In [ ]: current_part_uni_counts = \
enclosi ng_encodeable_data['Encl osing_Current_Part_Uni'].value_counts()
current_part_uni_counts
```

```
Out[ ]: No      2519
Yes     1628
Name: Encl osing_Current_Part_Uni, dtype: int64
```

```
In [ ]: print(f'{round(current_part_uni_counts[1]/len(enclosi ng_encodeable_data)*100, 2)}%')
```

```
In [ ]: current_part_uni_data_yes = \
plot_over_grey(locate_encl osing_encodeable_data, 'Encl osing_Current_Part_Uni', \
'Yes', '')
```

Enclosing Current Part Uni



Middleton, M. 2024, Hillforts Primer

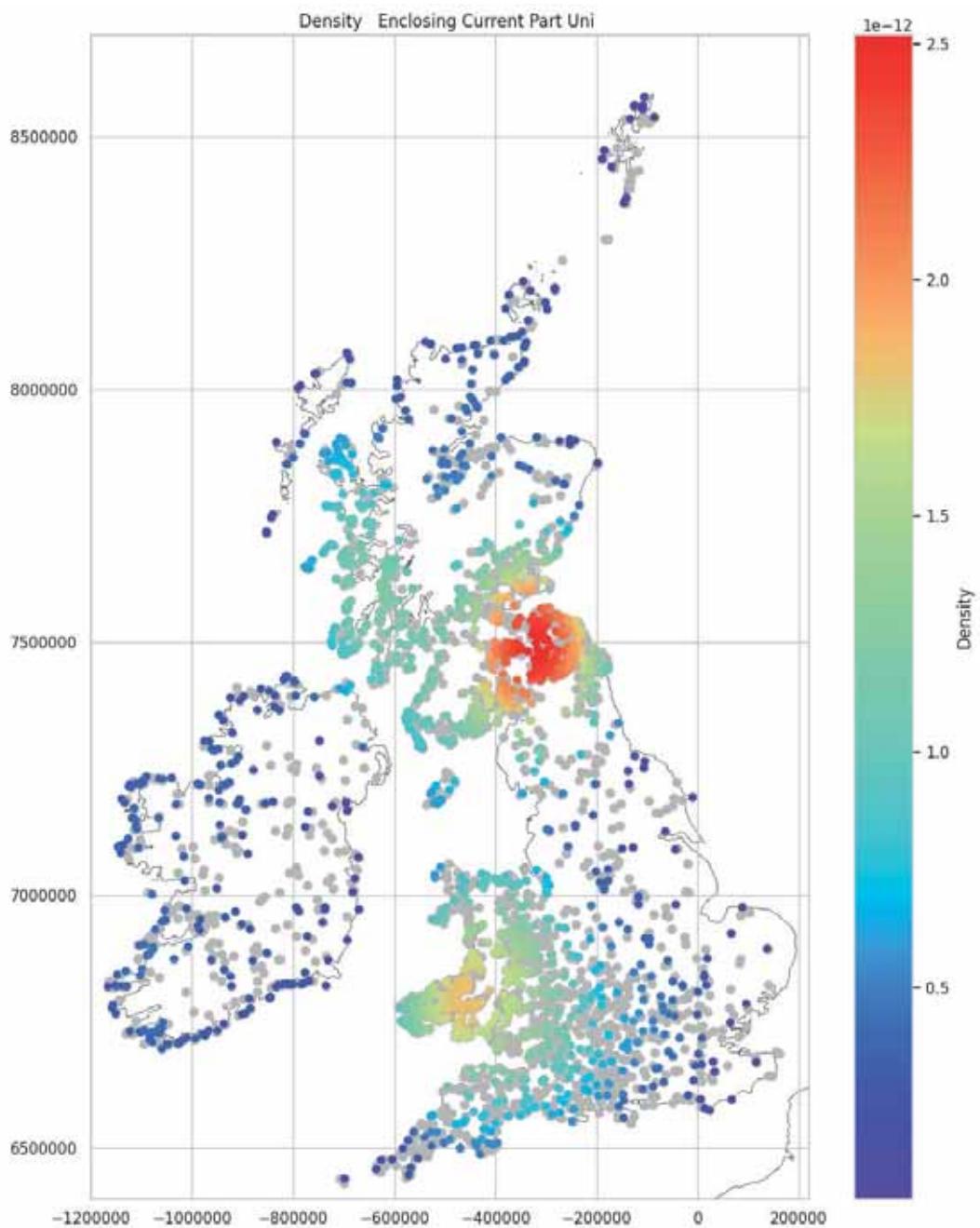
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

39.26%

Enclosing Current Part Univariate Density Mapped

The focus for this class is most intense over the Southern Uplands, southwest Wales and the Northwest.

```
In [ ]: plot_density_over_grey(current_part_uni_data_yes, 'Enclosing_Current_Part_Uni')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Current Univallate Mapped

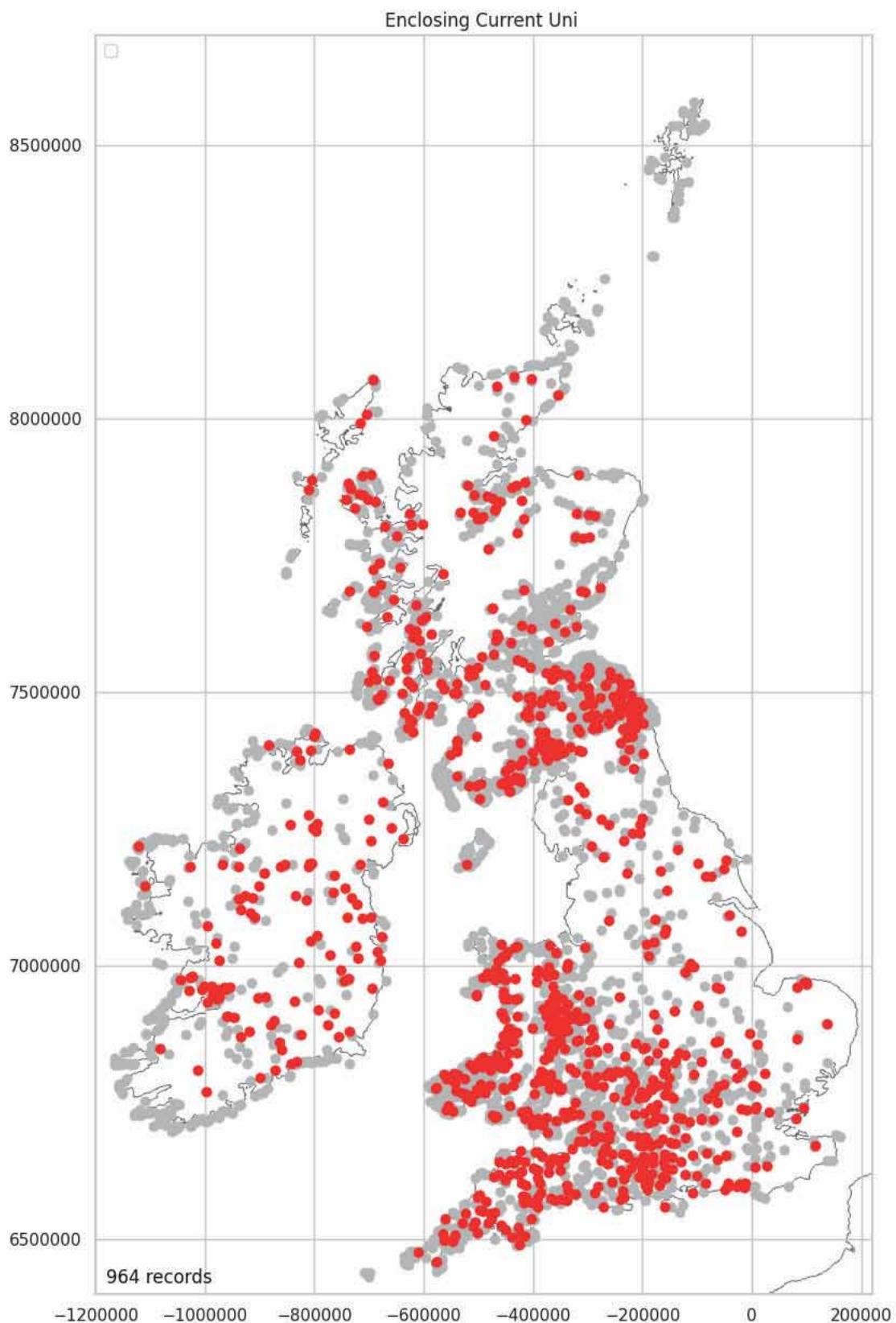
964 (23.25%) of hillforts are identified as Current Univallate. They are distributed right across the atlas.

```
In [ ]: current_uni_counts = \
enclosing_encodeable_data['Enclosing_Current_Uni'].value_counts()
current_uni_counts
```

```
Out[ ]: No      3183
Yes     964
Name: Enclosing_Current_Uni, dtype: int64
```

```
In [ ]: print(f'{round(current_uni_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
```

```
In [ ]: current_uni_data_yes = \
plot_over_grey(location_enclosing_encodeable_data, \
'Enclosing_Current_Uni', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

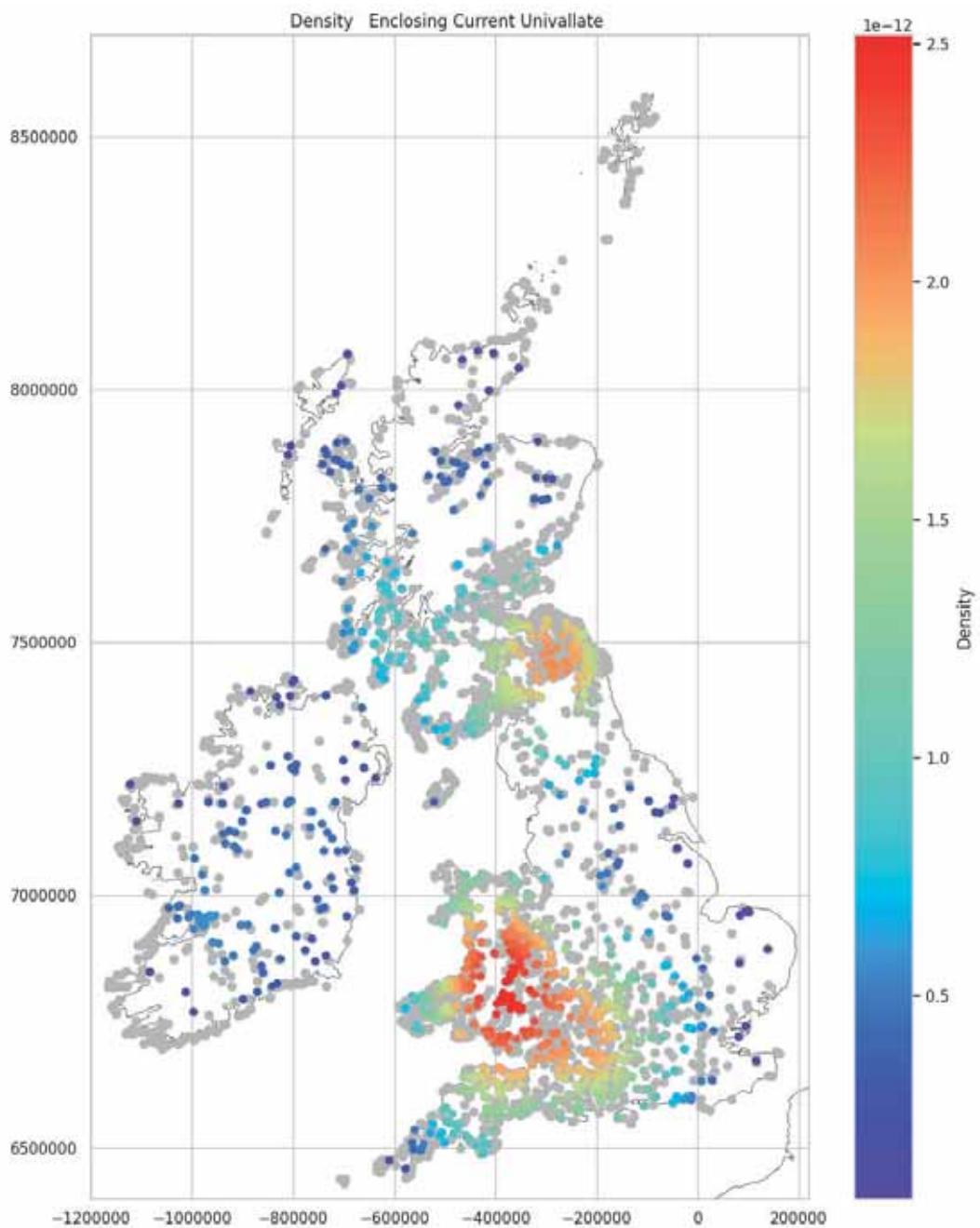
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

23.25%

Enclosing Current Univallate Density Mapped

Univallate hillforts cluster most in the South. The focus is noticeably east of the Cambrian Mountains and into South Central England. There is a secondary cluster in the Northeast and a third, much smaller cluster, in the Northwest.

```
In [ ]: plot_density_over_grey(current_uni_data_yes, 'Enclosing_Current_Univallate')
```



Enclosing Current Part Bivallate Mapped

1058 (25.51%) of hillforts are identified as Current Part Bivallate. They are distributed right across the atlas. They are noticeably sparse across northeast Ireland.

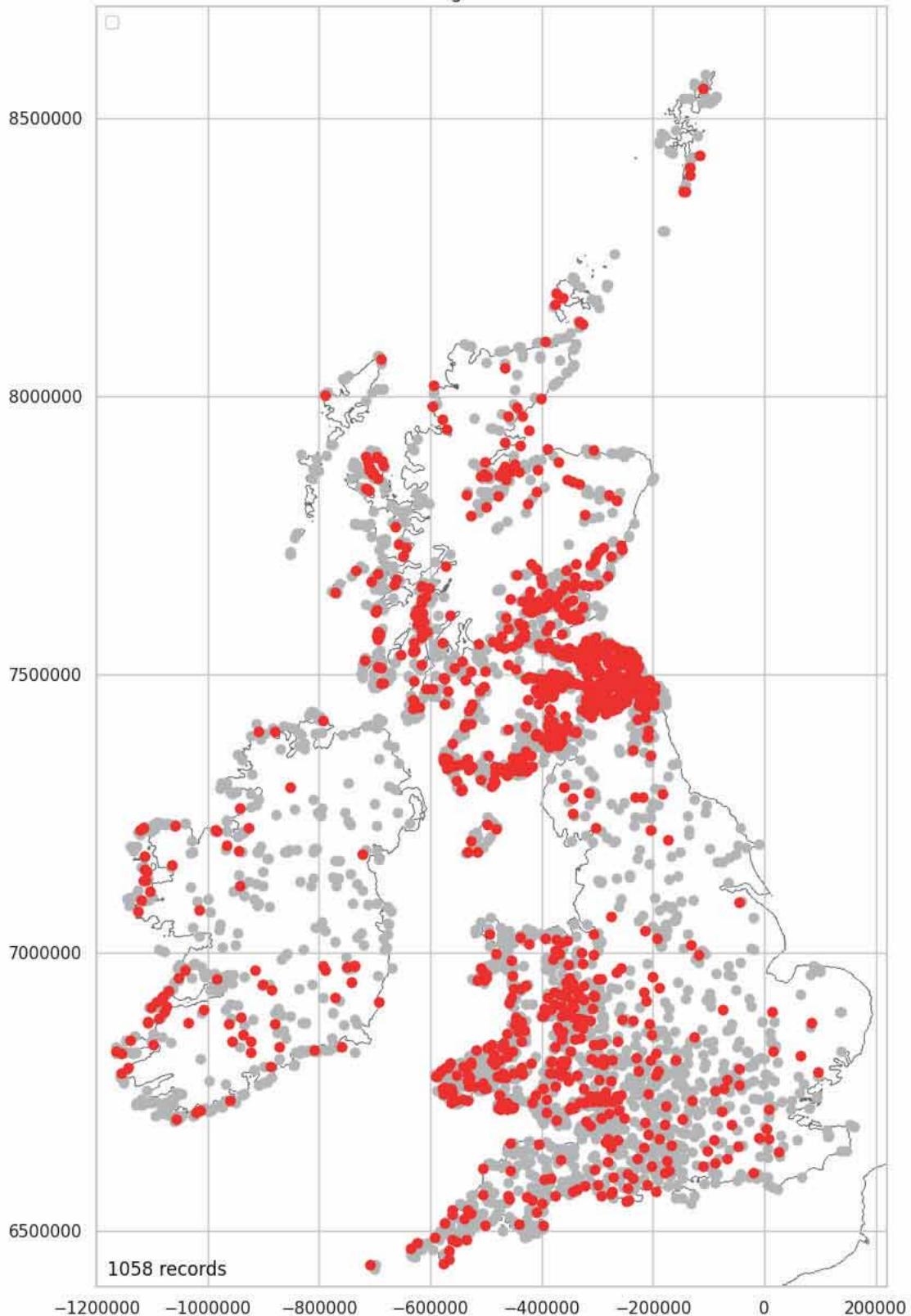
```
In [ ]: current_part_bi_counts = \
enclosing_encodeable_data['Enclosing_Current_Part_Bi'].value_counts()
current_part_bi_counts
```

```
Out[ ]: No      3089
Yes     1058
Name: Enclosing_Current_Part_Bi, dtype: int64
```

```
In [ ]: print(f'{round(current_part_bi_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
25.51%
```

```
In [ ]: current_part_bi_data_yes = \
plot_over_grey(location_enclosing_encodeable_data, 'Enclosing_Current_Part_Bi', \
'Yes', '')
```

Enclosing Current Part Bi



Middleton, M. 2024, Hillforts Primer

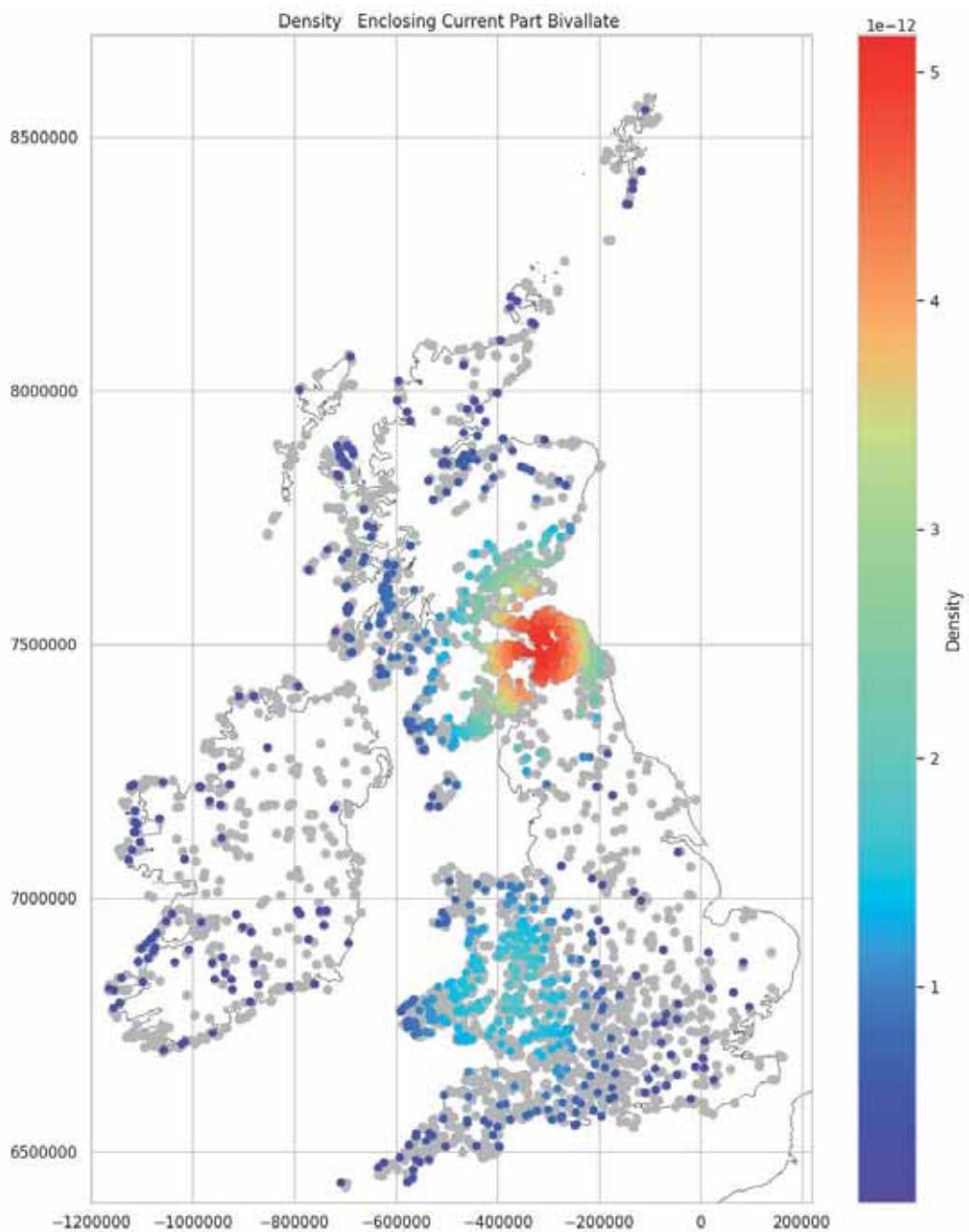
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

25. 51%

Enclosing Current Part Bivallate Density Mapped

Current Part Bivallate forts cluster most intensively in the Northeast. There is a secondary, much more sparse cluster, over the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(current_part_bi_data_yes, 'Enclosing_Current_Part_Bivallate')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Current Bivallate Mapped

395 (8.44%) of hillforts fall into the Current Bivallate class. They are noticeably more concentrated over the Northeast and away from the coasts.

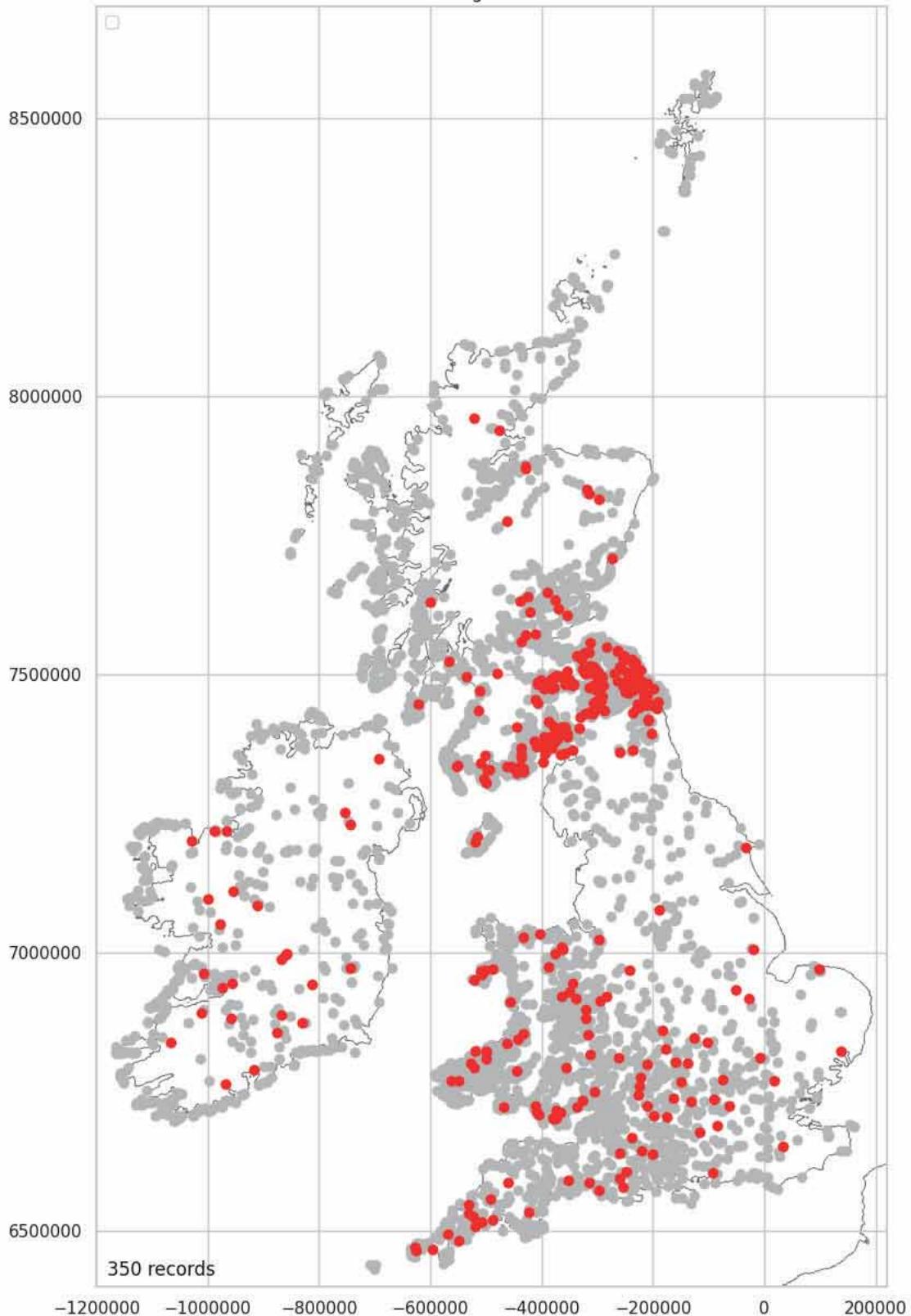
```
In [ ]: current_bi_counts = \
enclosi ng_encodeable_data['Encl osing_Current_Bi'].value_counts()
current_bi_counts
```

```
Out[ ]: No      3797
Yes     350
Name: Encl osing_Current_Bi, dtype: int64
```

```
In [ ]: print(f'{round(current_bi_counts[1]/len(enclosi ng_encodeabl e_data)*100, 2)}%')
8.44%
```

```
In [ ]: current_bi_data_yes = \
plot_over_grey(location_enclosi ng_encodeabl e_data, 'Encl osing_Current_Bi', \
'Yes', '')
```

Enclosing Current Bi



Middleton, M. 2024, Hillforts Primer

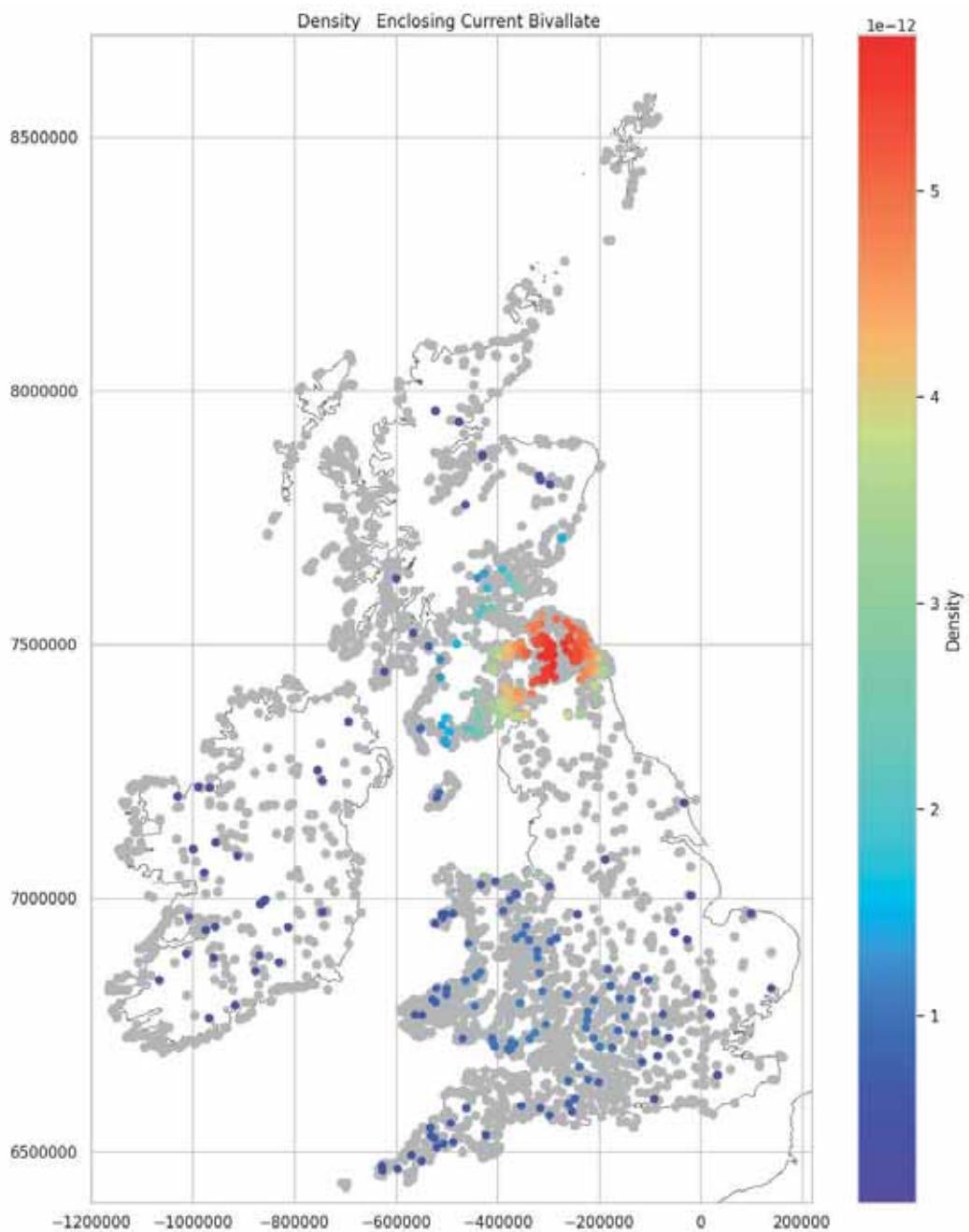
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

8.44%

Enclosing Current Bivallate Density Mapped

There is a single main cluster of Current Bivallate hillforts over the Northeast.

```
In [ ]: plot_density_over_grey(current_bi_data_yes, 'Enclosing_Current_Bivallate')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Current Part Multivallate Mapped

596 (14.37%) of hillforts are classified as Current Part Multivallate.

```
In [ ]: current_part_multi_counts = \
encl osing_encodeable_data['Encl osing_Current_Part_Multi'].value_counts()
current_part_multi_counts
```

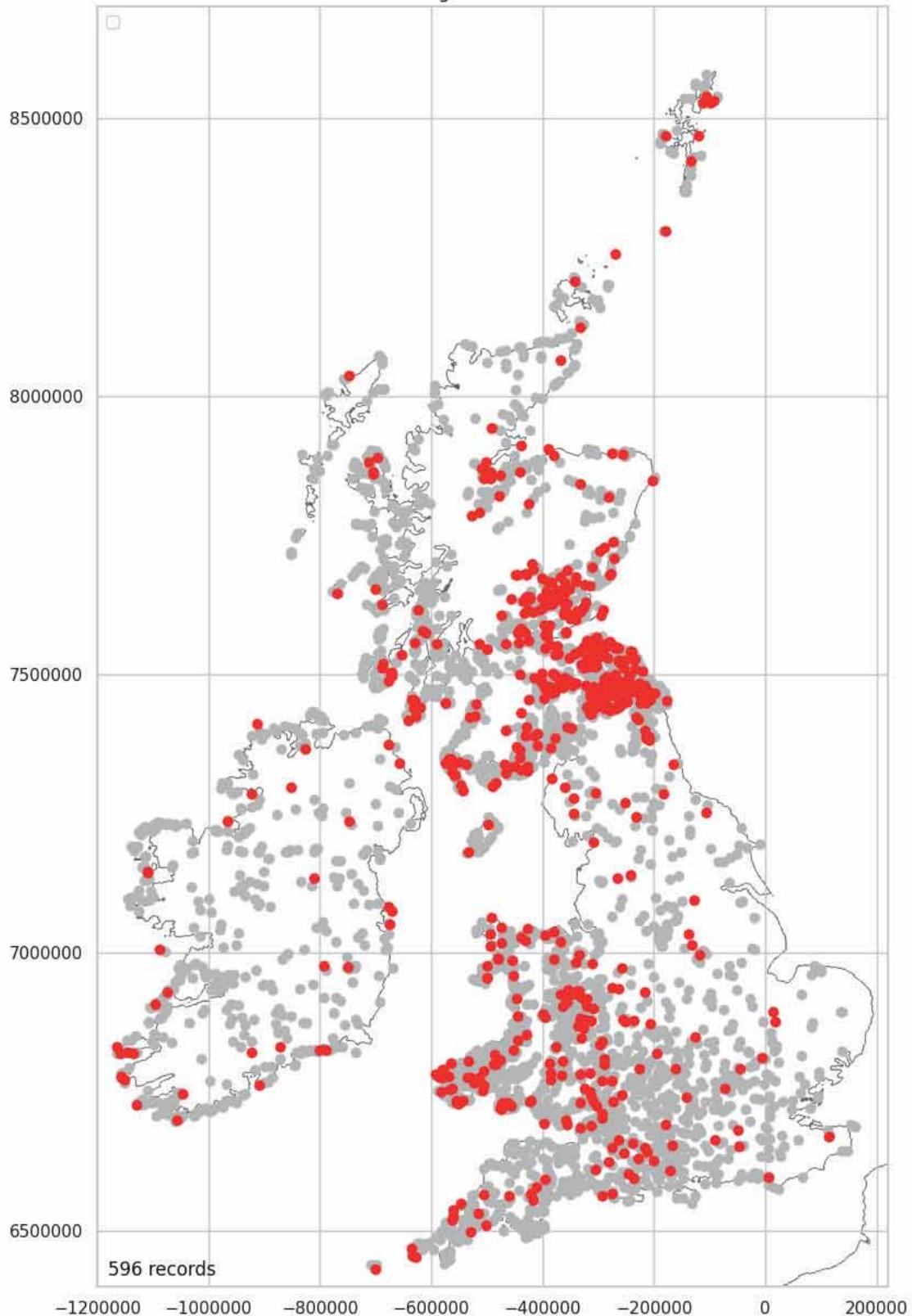
```
Out[ ]: No      3551
Yes     596
Name: Encl osing_Current_Part_Multi, dtype: int64
```

```
In [ ]: print(f'{round(current_part_multi_counts[1]/len(encl osing_encodeable_data)*100, 2)}%')
```

```
14.37%
```

```
In [ ]: current_part_multi_data_yes = \
plot_over_grey(locate_encl osing_encodeable_data, \
'Encl osing_Current_Part_Multi', 'Yes', '')
```

Enclosing Current Part Multi



Middleton, M. 2024, Hillforts Primer

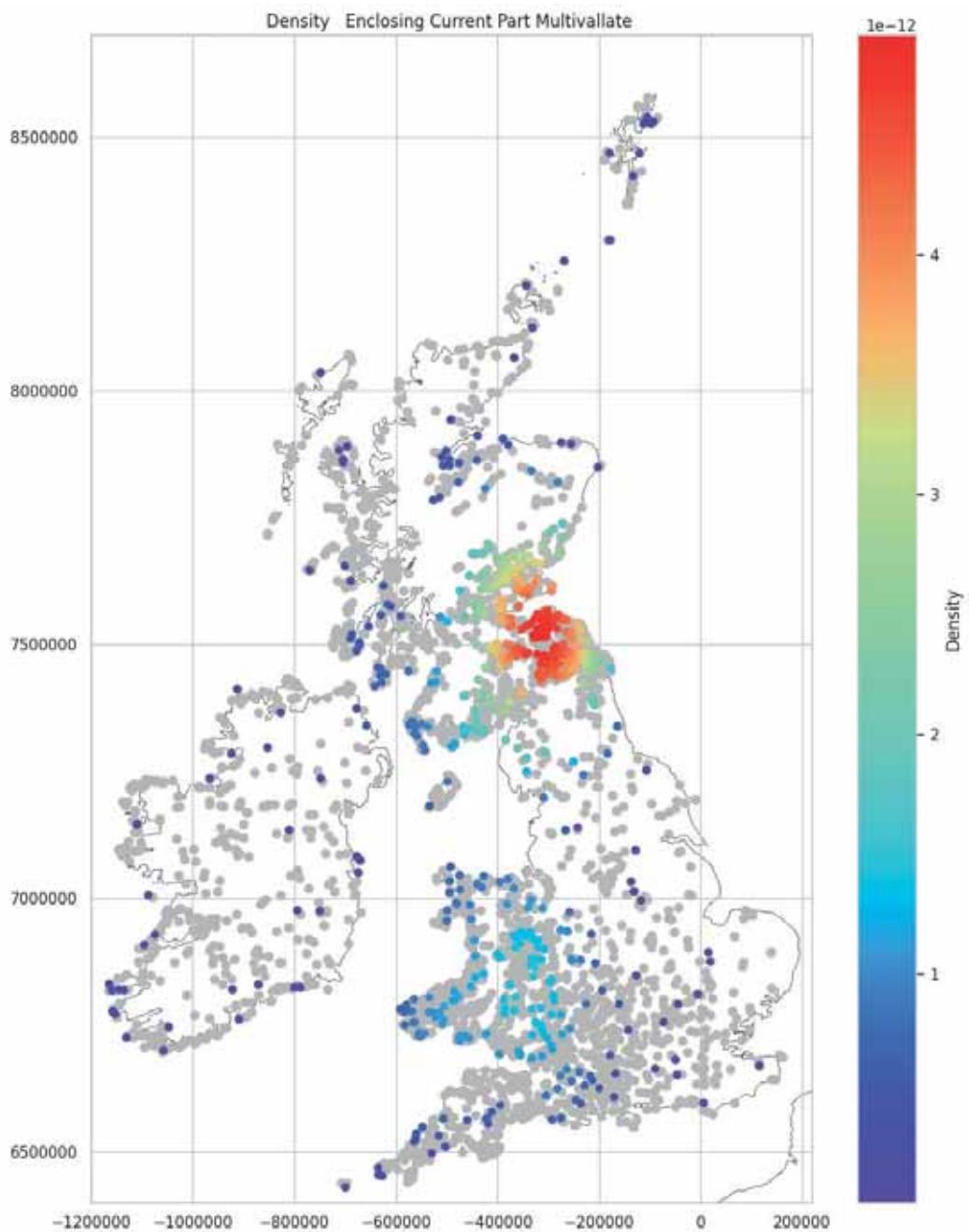
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

14.37%

Enclosing Current Part Multivallate Density Mapped

The main cluster of Current Part Multivallate hillforts is in the Northeast. There is a very sparse cluster in the South, east of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(current_part_multi_data_yes, \
    'Enclosing_Current_Part_Multivallate')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Current Multivallate Mapped

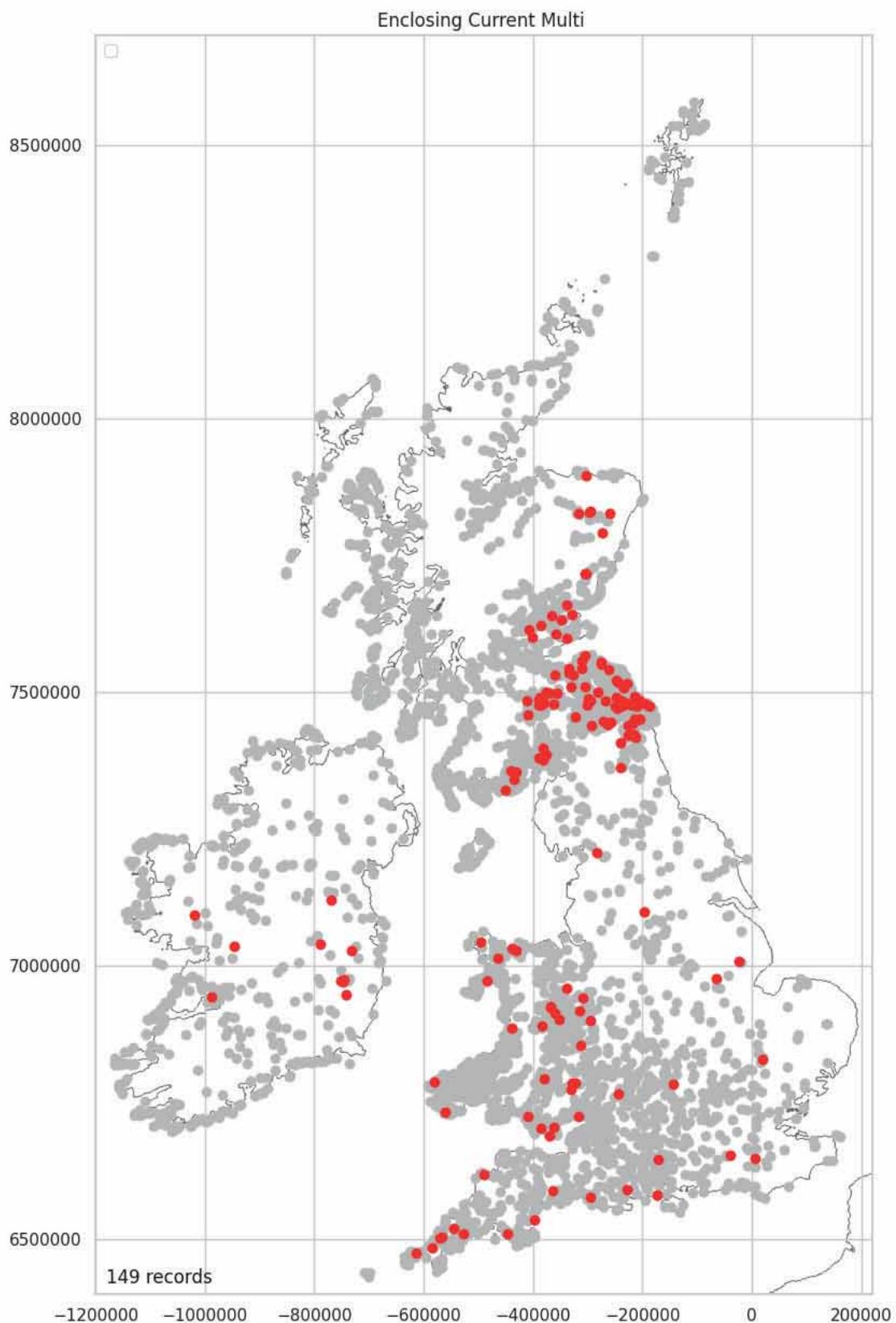
Just 149 (3.59%) of hillforts are identified as being current Multivallate.

```
In [ ]: current_mult_i_counts = \
encl osing_encodeabl e_data['Encl osing_Current_Mul ti'].val ue_counts()
current_mult_i_counts
```

```
Out[ ]: No      3998
Yes     149
Name: Encl osing_Current_Mul ti, dtype: int64
```

```
In [ ]: print(f'{round(current_mult_i_counts[1]/len(encl osing_encodeabl e_data)*100, 2)}%')
3.59%
```

```
In [ ]: current_mult_i_data_yes = \
plot_over_grey(locate_encl osing_encodeabl e_data, 'Encl osing_Current_Mul ti', \
'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

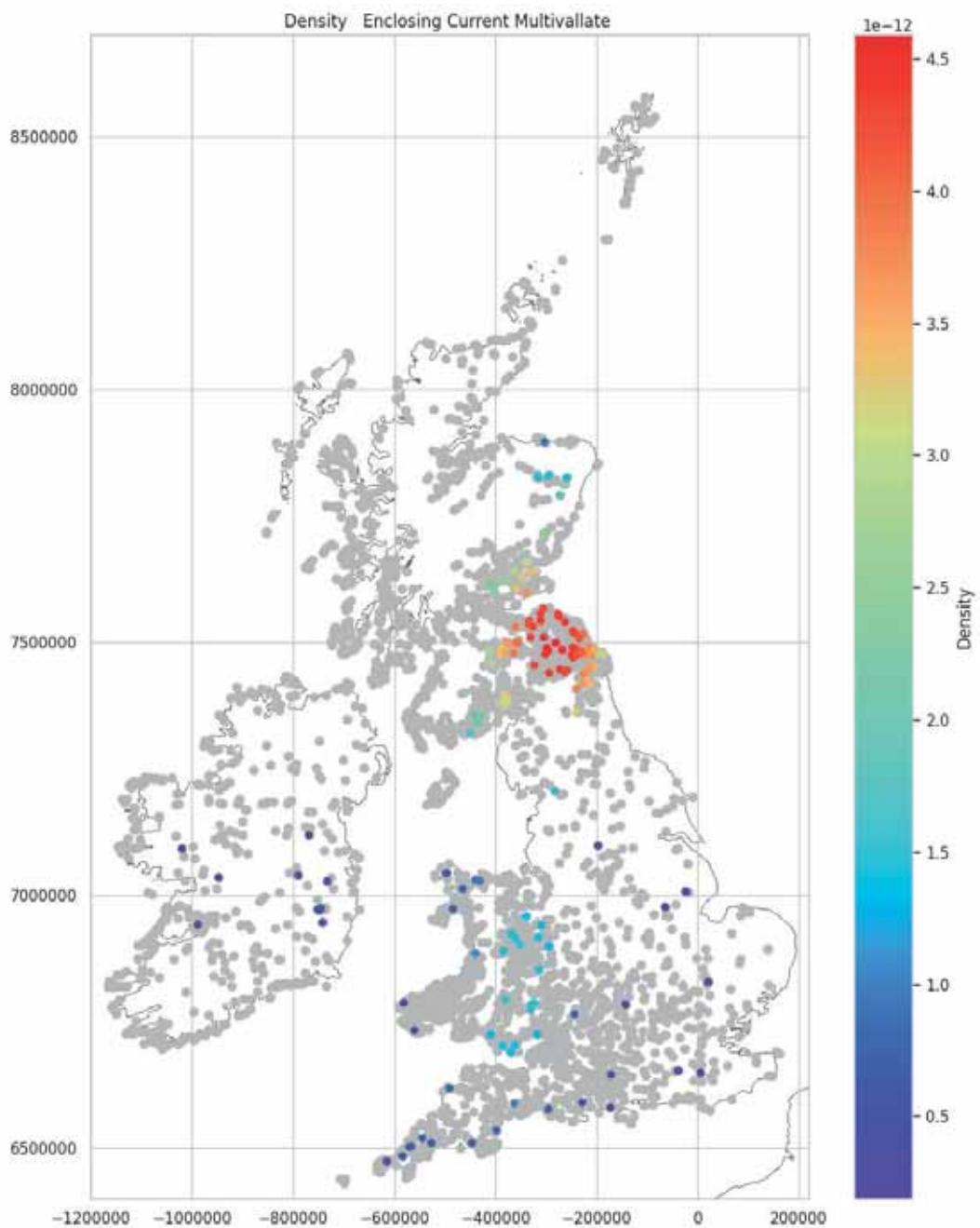
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3.59%

Enclosing Current Multivallate Density Mapped

There is a main cluster of Current Multivallate hillforts in the Northeast and a very sparse second cluster along the eastern fringe of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(current_multi_data_yes, 'Enclosing Current Multivallate')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Current Unknown Mapped

256 (6.34%) of hillforts are identified as having an unknown current enclosing circuit.

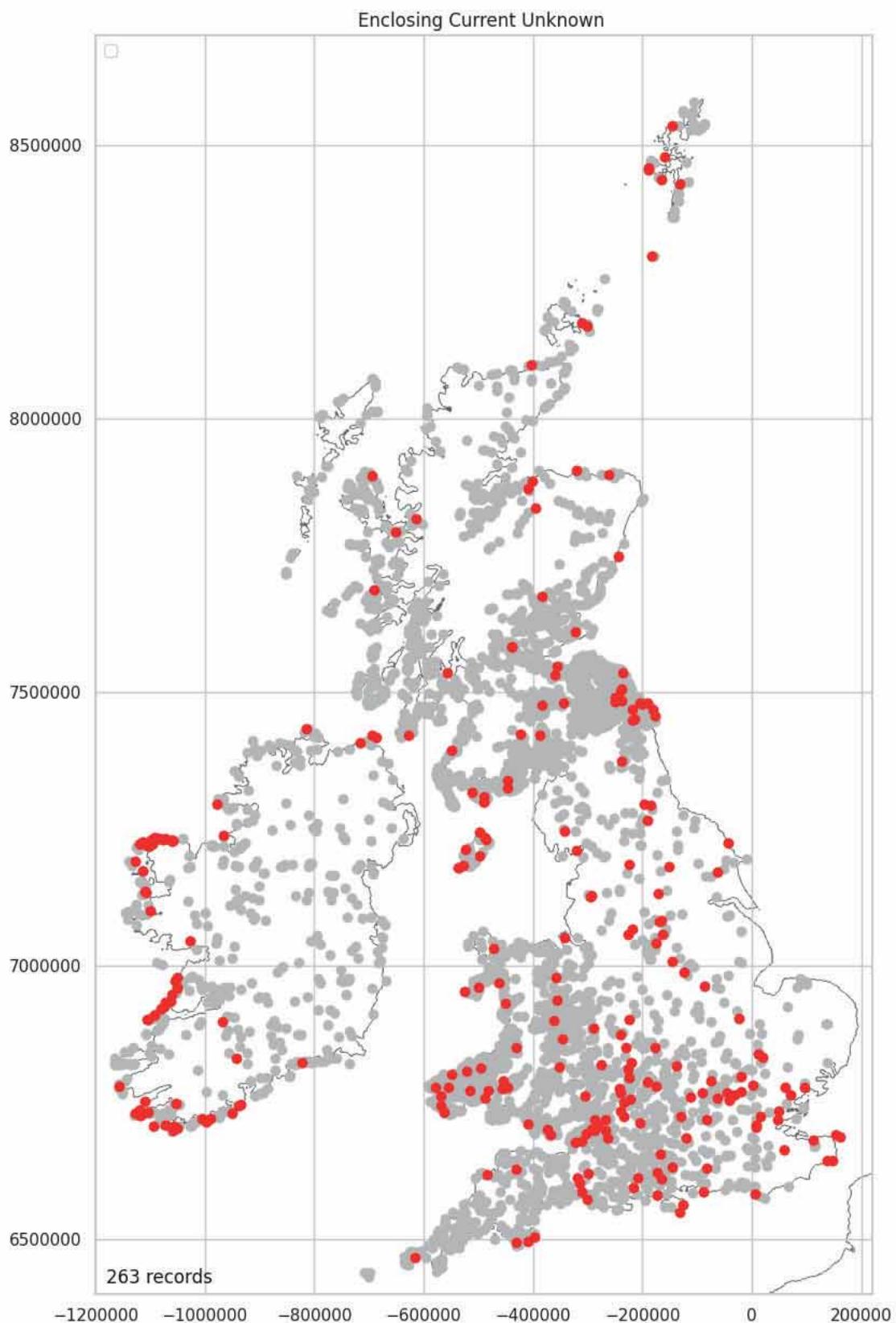
```
In [ ]: current_uk_counts = \
encl osing_encodeable_data['Encl osing_Current_Unknown'].val ue_counts()
current_uk_counts
```

```
Out[ ]: No      3884
Yes     263
Name: Encl osing_Current_Unknown, dtype: int64
```

```
In [ ]: print(f'{round(current_uk_counts[1]/len(encl osing_encodeabl e_data)*100, 2)}%')
```

6.34%

```
In [ ]: current_uk_data_yes = \
plot_over_grey(location_encl osing_encodeable_data, 'Encl osing_Current_Unknown', \
'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

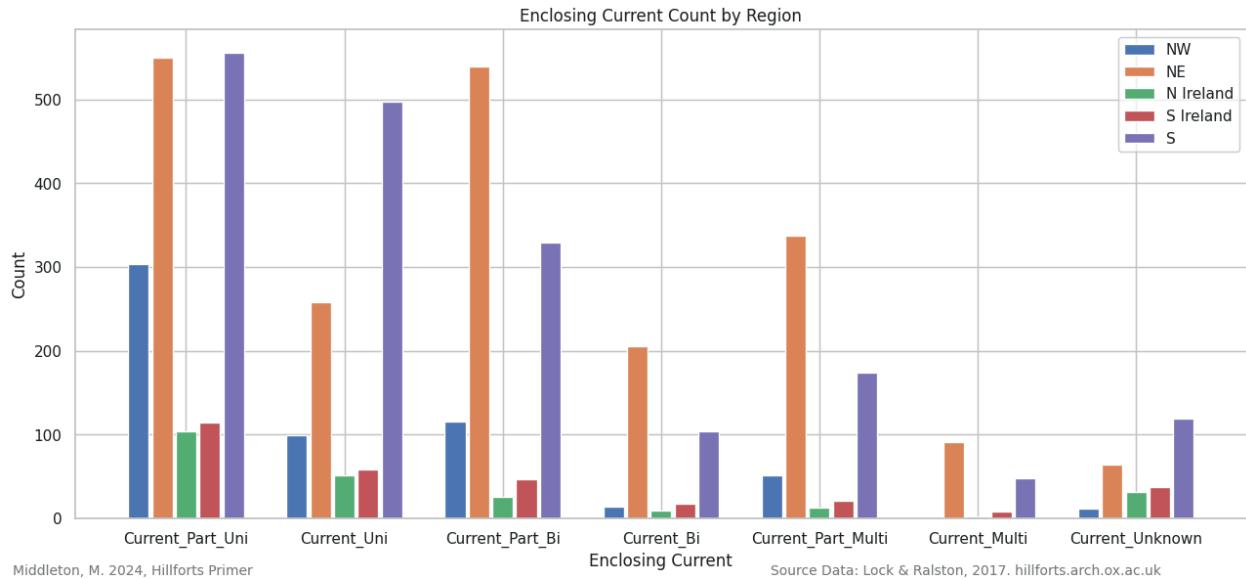
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 34%

Enclosing Current Plotted by Region (Count)

It is difficult to read the plot showing the count by current enclosing class as there are so many hillforts in the Northeast and South. It is simpler to look at all the proportions for an area and then compare these to the proportions across other areas. For instance, the South show strong returns at the bottom end of the range with high counts in Part Univallate, Univallate and Part Bivallate. In comparison, the Northeast has its three highest counts in Part Univallate, Part Bivallate and Part Multivallate.

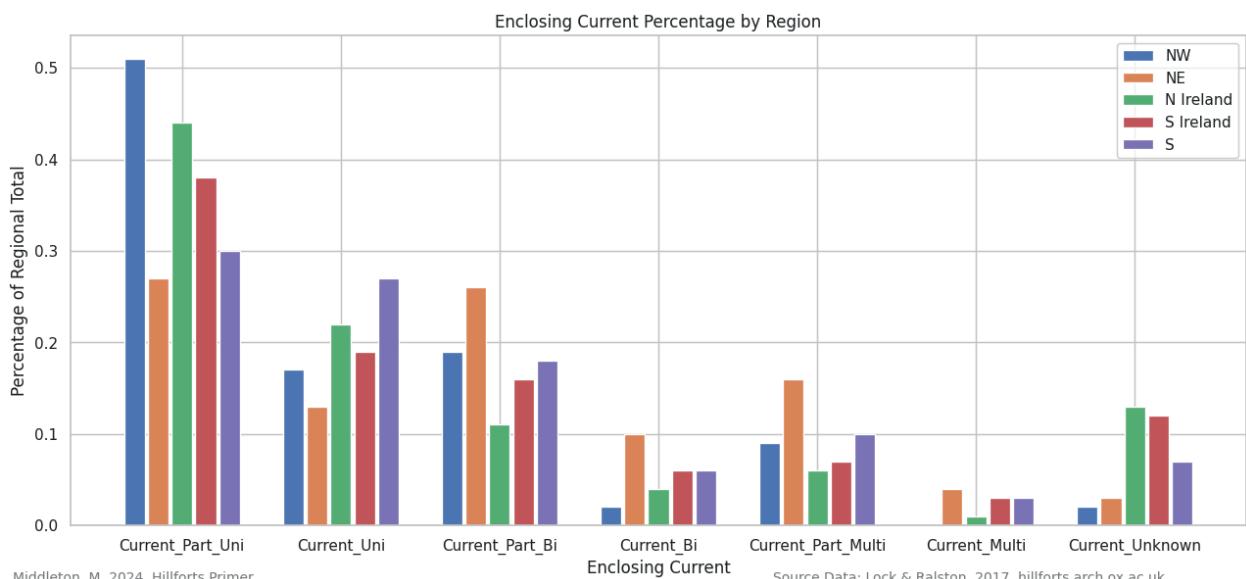
```
In [ ]: plot_regions(location_enclosing_encodeable_data_nw,
location_enclosing_encodeable_data_ne,
location_enclosing_encodeable_data_ireland_n,
location_enclosing_encodeable_data_ireland_s,
location_enclosing_encodeable_data_south,
['Enclosing_Current_Part_Uni', 'Enclosing_Current_Uni', \
'Enclosing_Current_Part_Bi', \
'Enclosing_Current_Bi', 'Enclosing_Current_Part_Multi', \
'Enclosing_Current_Multi', \
'Enclosing_Current_Unknown'], \
'Enclosing_Current', \
'Enclosing_Current_Count_by_Region', 1, 'Yes')
```



Enclosing Current Plotted by Region (Percentage)

It is revealing to look at this data proportionally. Looking at the data in this way, all the regions are relatively similar. All have a predominance of Part Univallate hillforts and secondary and tertiary clusters of Univallate and Part Bivallate. The Northeast is the outlier in that it is more likely to have Bivallate and Part Multivallate hillforts. The unknown are dominated by hillforts in Ireland.

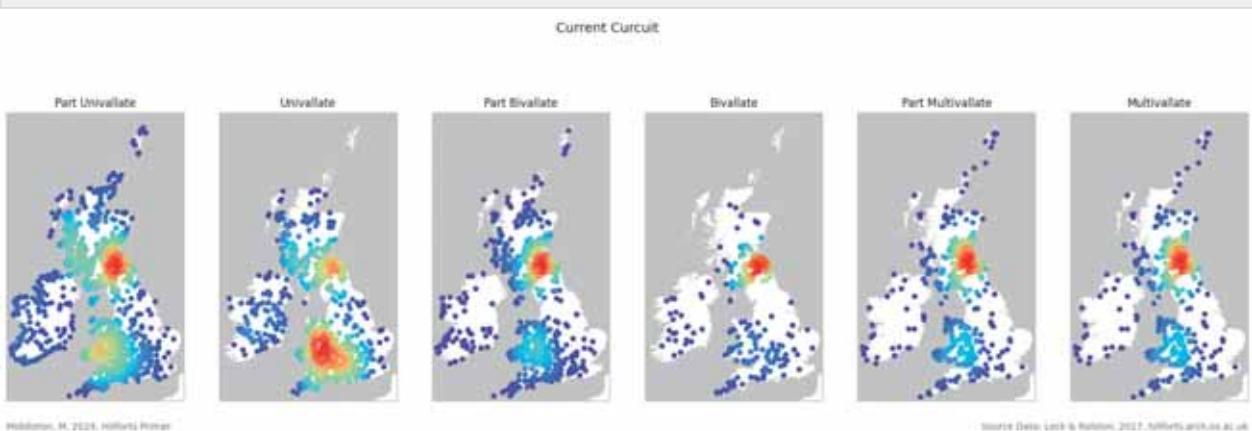
```
In [ ]: plot_regions(location_enclosing_encodeable_data_nw,
location_enclosing_encodeable_data_ne,
location_enclosing_encodeable_data_ireland_n,
location_enclosing_encodeable_data_ireland_s,
location_enclosing_encodeable_data_south,
['Enclosing_Current_Part_Uni', 'Enclosing_Current_Uni', \
'Enclosing_Current_Part_Bi', \
'Enclosing_Current_Bi', 'Enclosing_Current_Part_Multi', \
'Enclosing_Current_Multi', \
'Enclosing_Current_Unknown'], \
'Enclosing_Current', \
'Enclosing_Current_Percentage_by_Region', 1, 'Yes', True)
```



Current Circuit Summary

All areas have a high proportion of Part Univallate hillforts. The main clusters are in the Northeast, south Wales and the Northwest. As a proportion by region, they are most common in the Northwest. Univallate forts cluster most densely in the South with smaller clusters in the Northeast and Northwest. Although the southern cluster is the most intense, within their own region, Univallate hillforts are almost half as common as Part Univallate forts. In all the remaining classes the Northeast has the most intense cluster with the South showing secondary, much less intense clusters.

```
In [ ]: plot_density_over_grey_hex(current_part_uni_data_yes, current_uni_data_yes, \
                                    current_part_bi_data_yes, current_bi_data_yes, \
                                    current_part_multi_data_yes, current_multi_data_yes, \
                                    'Current Circuit')
```



Enclosing Period

It is assumed that Enclosing Period refers to the morphology of the enclosing works at the time of construction. Very few hillforts have a Period Enclosing recorded. There is insufficient data to show any meaningful distributions. The majority of records are in the Northeast and this may indicate there is a recording bias toward area.

Enclosing Period Part Univallate Mapped

There are just 36 hillforts where Period Part Univallate has been recorded. The majority are in the Northeast.

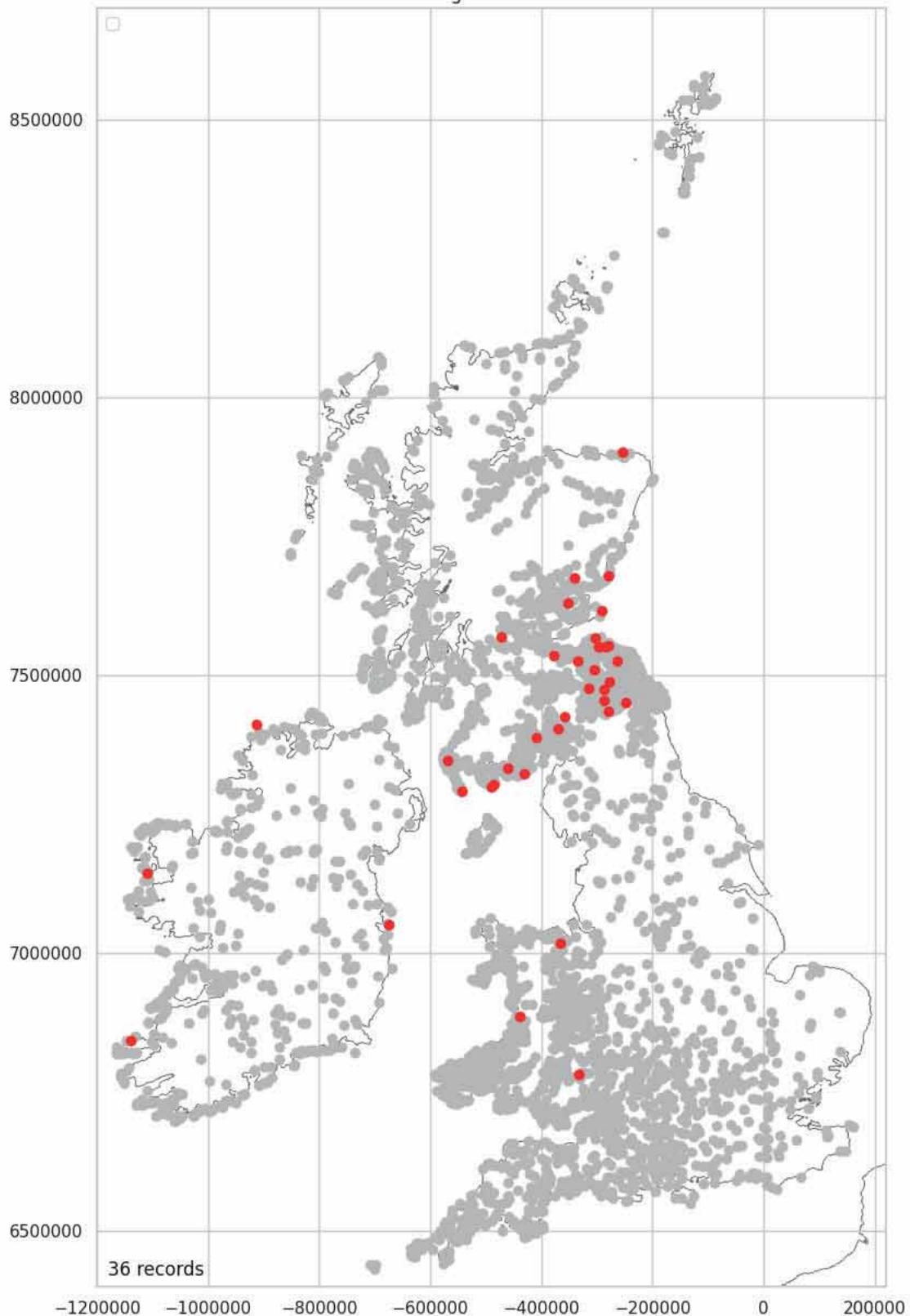
```
In [ ]: period_part_uni_counts = \
enclosing_encodeable_data\
['Enclosing_Period_Part_Uni'].value_counts()
period_part_uni_counts
```

```
Out[ ]: No      4111
Yes      36
Name: Enclosing_Period_Part_Uni, dtype: int64
```

```
In [ ]: print(f'{round(period_part_uni_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
0.87%
```

```
In [ ]: period_part_uni_data_yes = \
plot_over_hex(location_enclosing_encodeable_data, 'Enclosing_Period_Part_Uni', \
              'Yes', '')
```

Enclosing Period Part Uni



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.87%

Enclosing Period Univallate Mapped

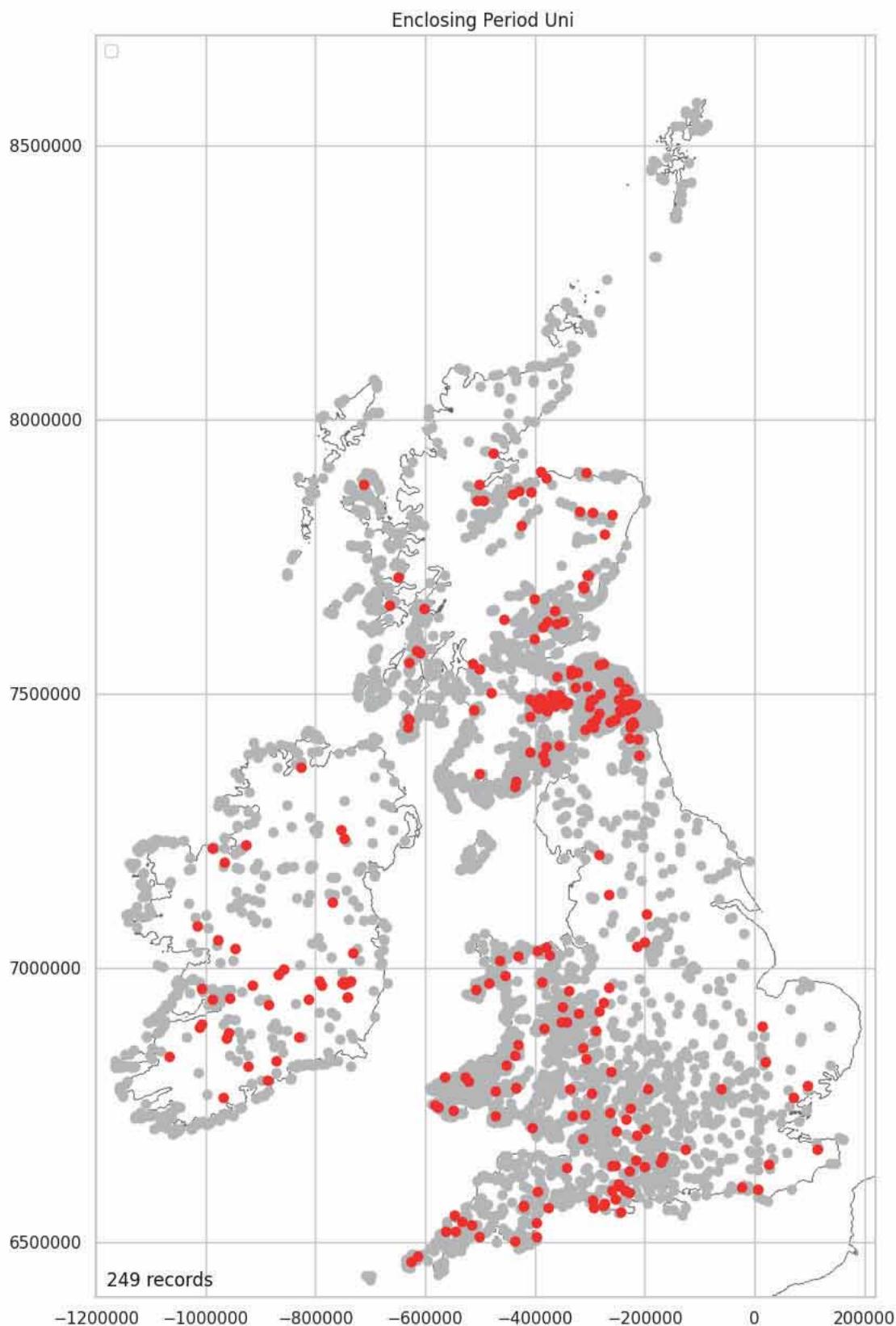
249 (6%) of hillforts have a Period Univallate classification.

```
In [ ]: period_uni_counts = \
enclosi ng_encodeable_data['Enclosi ng_Period_U ni'].value_counts()
period_uni_counts
```

```
Out[ ]: No      3898  
         Yes     249  
         Name: Enclosing_Period_Uni , dtype: int64
```

```
In [ ]: print(f' {round(period_uni_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')  
6.0%
```

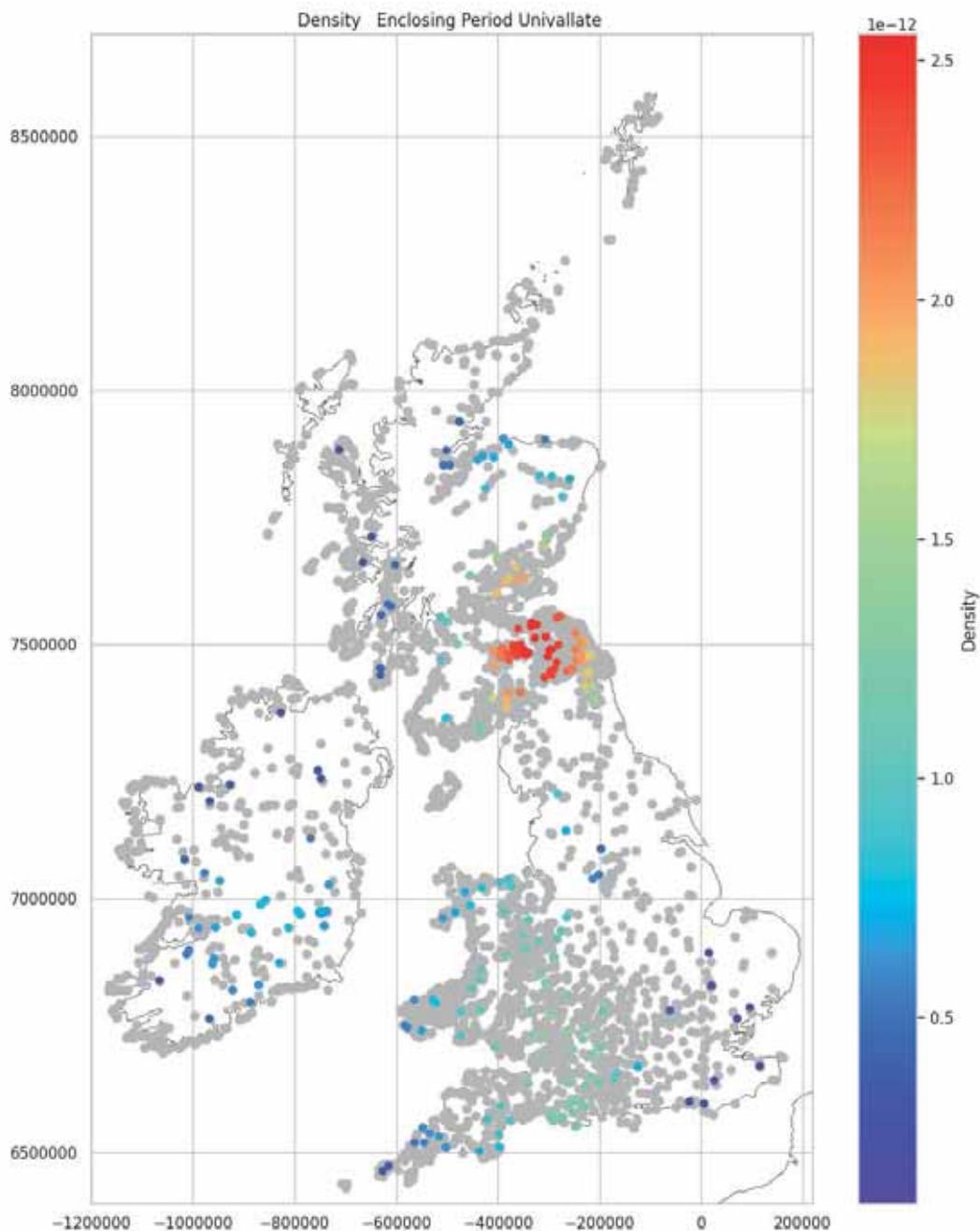
```
In [ ]: period_uni_data_yes = plot_over_grey(location_enclosing_encodeable_data, \  
                                         'Enclosing_Period_Uni', 'Yes', '')
```



Enclosing Period Univallate Density Mapped

The main cluster of Period Univallate forts is in the Northeast. There is a second, more diffuse cluster, over south-central England.

```
In [ ]: plot_density_over_grey(period_uni_data_yes, 'Enclosing Period Univallate')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Period Part Bivallate Mapped

There are 35 (0.84%) Period Part Bivallate forts. Again, these are mostly in the Northeast.

```
In [ ]: period_part_bi_counts = \
enclising_encodeable_data['Enclosing_Period_Part_Bi'].value_counts()
period_part_bi_counts
```

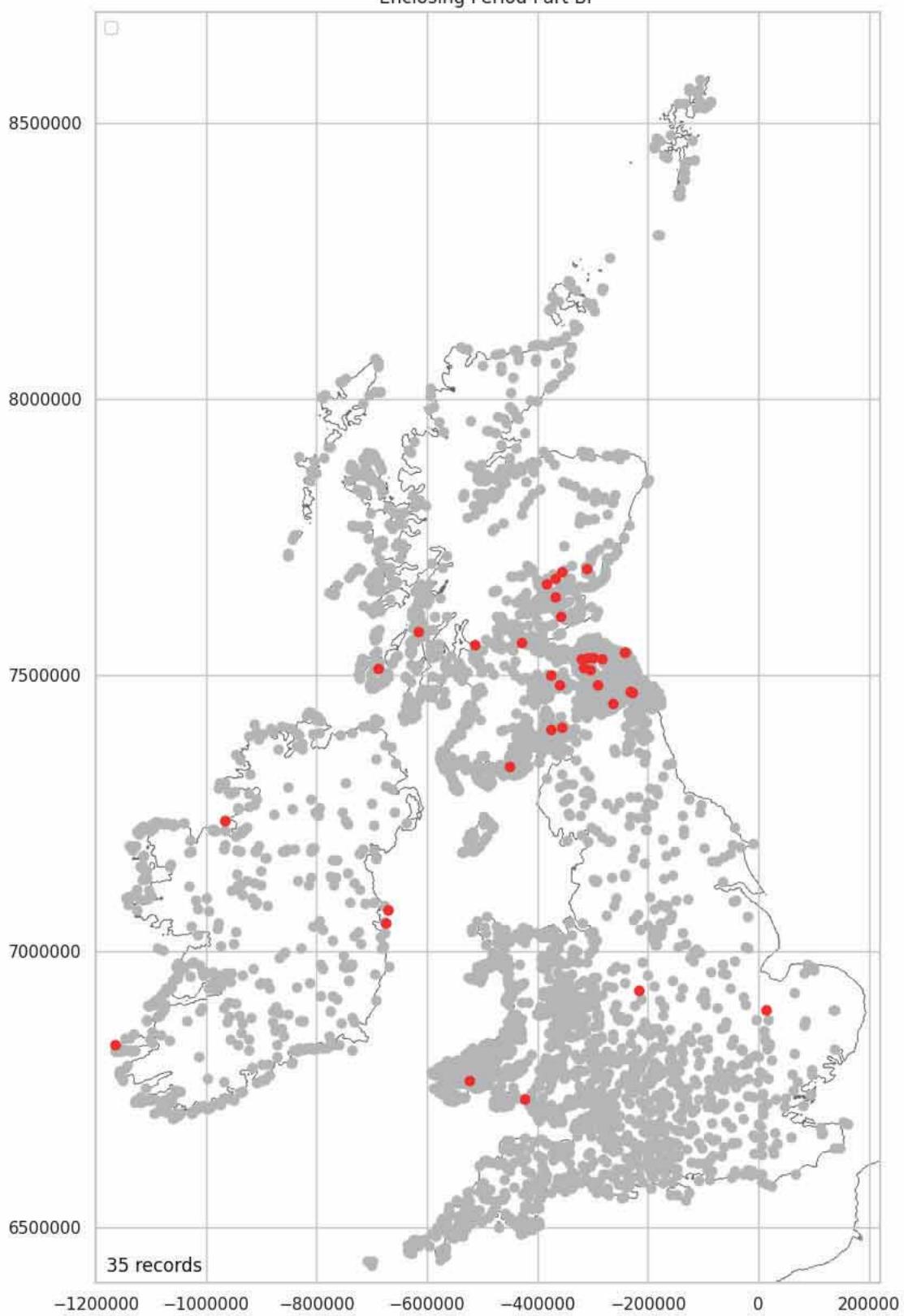
```
Out[ ]: No      4112
Yes      35
Name: Enclosing_Period_Part_Bi, dtype: int64
```

```
In [ ]: print(f'{round(period_part_bi_counts[1]/len(enclising_encodeable_data)*100, 2)}%')
0.84%
```

```
In [ ]: period_part_bi_data_yes = \
plot_over_grey(location_enclising_encodeable_data, 'Enclosing_Period_Part_Bi', \
606
```

'Yes', '')

Enclosing Period Part Bi



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.84%

Enclosing Period Bivallate Mapped

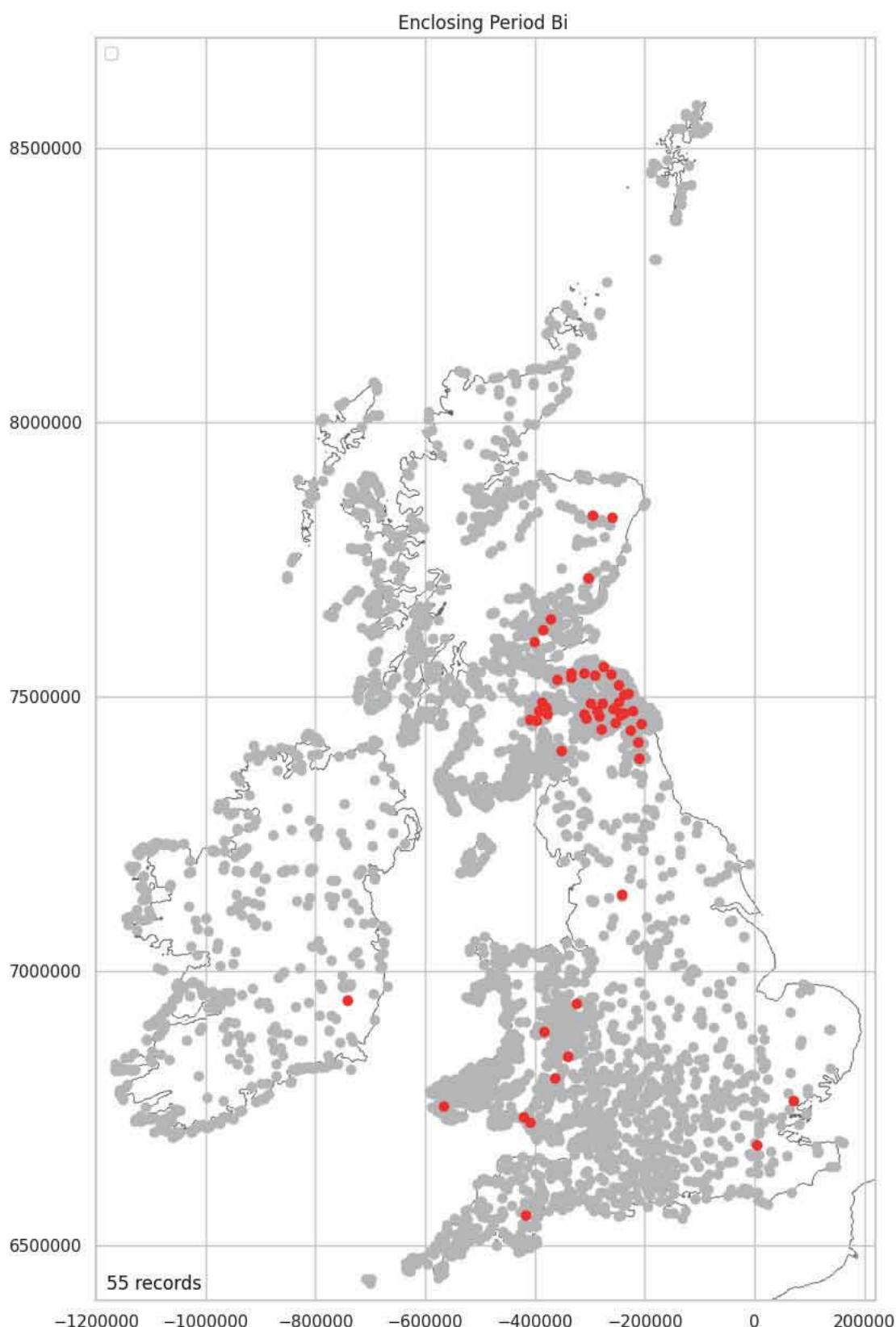
There are 55 (1.33%) Period Bivallate forts, also in the Northeast.

```
In [ ]: period_bi_counts = \
enclising_encodeable_data['Enclising_Period_Bi'].value_counts()
period_bi_counts
```

```
Out[ ]: No      4092  
         Yes     55  
         Name: Enclosing_Period_Bi , dtype: int64
```

```
In [ ]: print(f' {round(period_bi_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')  
1.33%
```

```
In [ ]: period_bi_data_yes = \  
plot_over_grey(location_enclosing_encodeable_data, 'Enclosing_Period_Bi', \  
'Yes', '')
```



Enclosing Period Part Multivallate Mapped

There are six (0.14%) Period Part Multivallate forts.

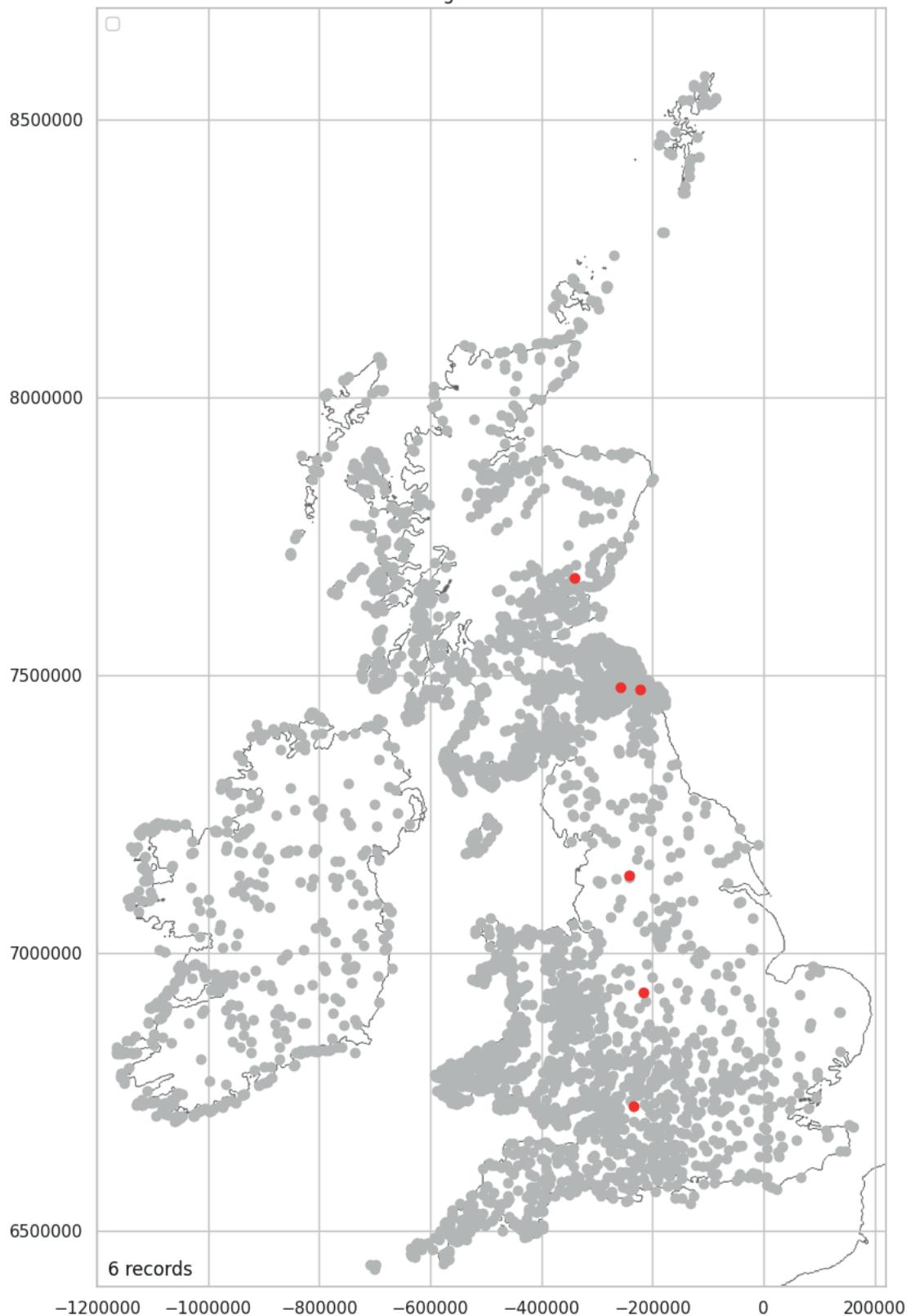
```
In [ ]: period_part_multi_counts = \
encl osing_encodeable_data['Encl osing_Period_Part_Mul ti'].value_counts()
period_part_multi_counts
```

```
Out[ ]: No      4141
Yes       6
Name: Encl osing_Period_Part_Mul ti, dtype: int64
```

```
In [ ]: print(f' {round(period_part_multi_counts[1]/len(encl osing_encodeable_data)*100, 2)}%')
0.14%
```

```
In [ ]: period_part_multi_data_yes = \
plot_over_grey(locate_encl osing_encodeable_data,
'Encl osing_Period_Part_Mul ti', 'Yes', '')
```

Enclosing Period Part Multi



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.14%

Enclosing Period Multivallate Mapped

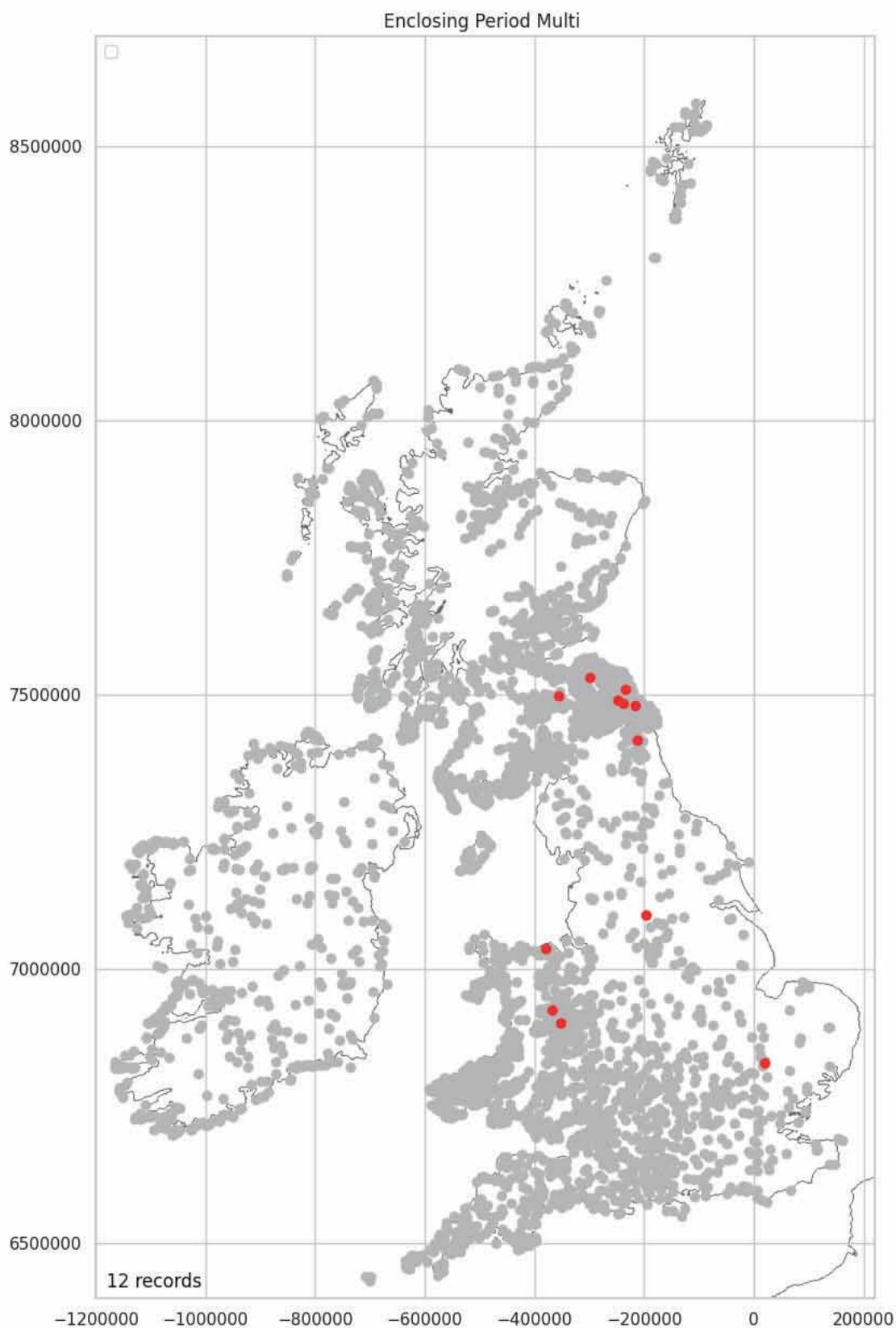
There are 12 (0.29%) Period Multivallate forts.

```
In [ ]: period_multi_counts = \
enclosing_encodeable_data['Enclosing_Period_Multi'].value_counts()
period_multi_counts
```

```
Out[ ]: No      4135  
         Yes     12  
         Name: Enclosing_Period_Multi, dtype: int64
```

```
In [ ]: print(f' {round(period_multi_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')  
0.29%
```

```
In [ ]: period_multi_data_yes = \  
plot_over_grey(location_enclosing_encodeable_data, \  
'Enclosing_Period_Multi', 'Yes', '')
```



Enclosing Surface

Enclosing Surface relates to the character of the enclosing circuit.

Enclosing Surface None Mapped

702 (16.93%) of hillforts have no information regarding the character of the enclosing circuit.

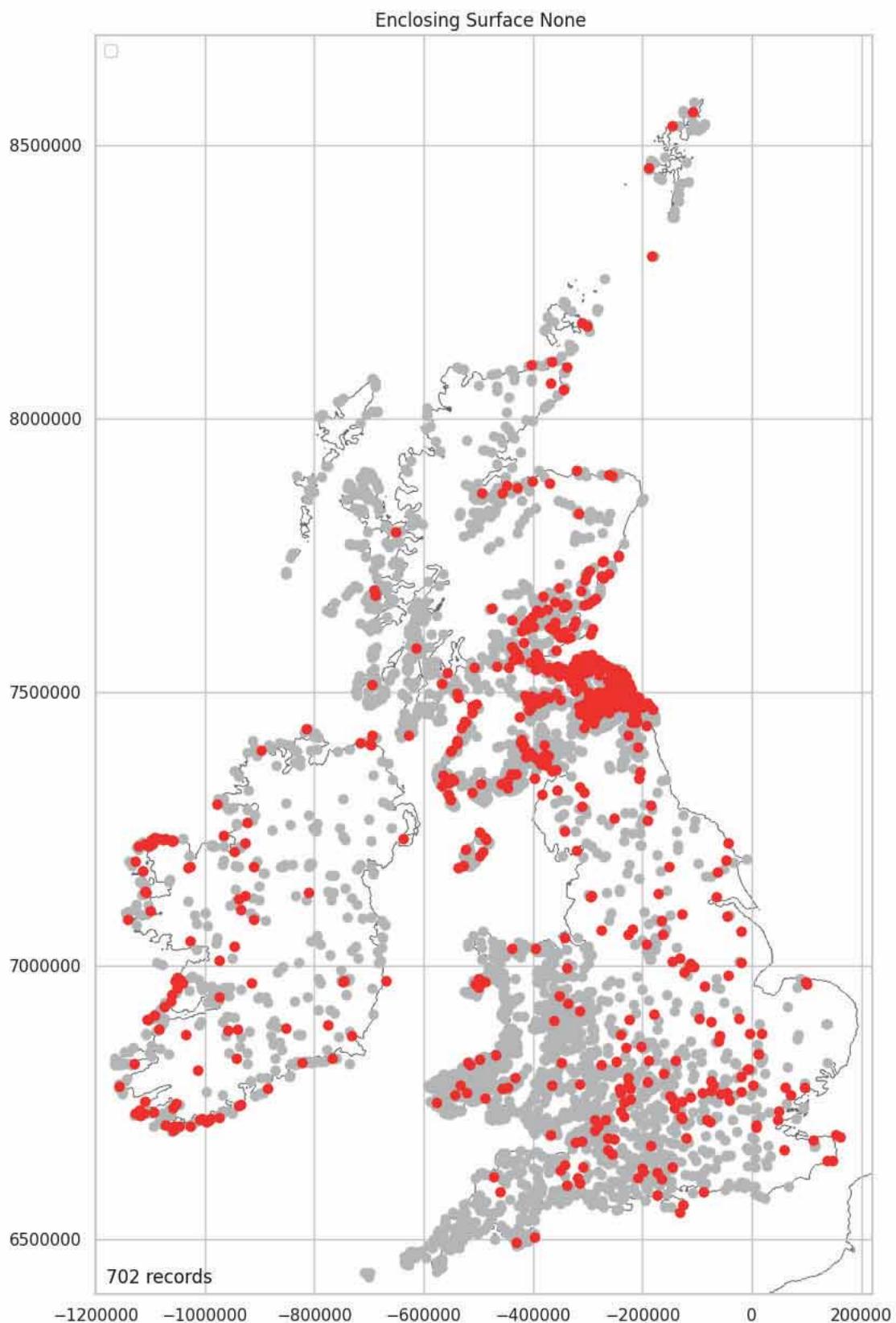
```
In [ ]: surface_none_counts = \
encl osi ng_encodeabl e_data['Encl osi ng_Surface_None'].val ue_counts()
```

```
Out[ ]: No      3445
Yes     702
Name: Encl osi ng_Surface_None, dtype: int64
```

```
In [ ]: print(f'{round(surface_none_counts[1]/len(encl osi ng_encodeabl e_data)*100, 2)}%')
```

16. 93%

```
In [ ]: surface_none_data_yes = \
plot_over_grey(location_encl osi ng_encodeabl e_data, 'Encl osi ng_Surface_None', \
'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

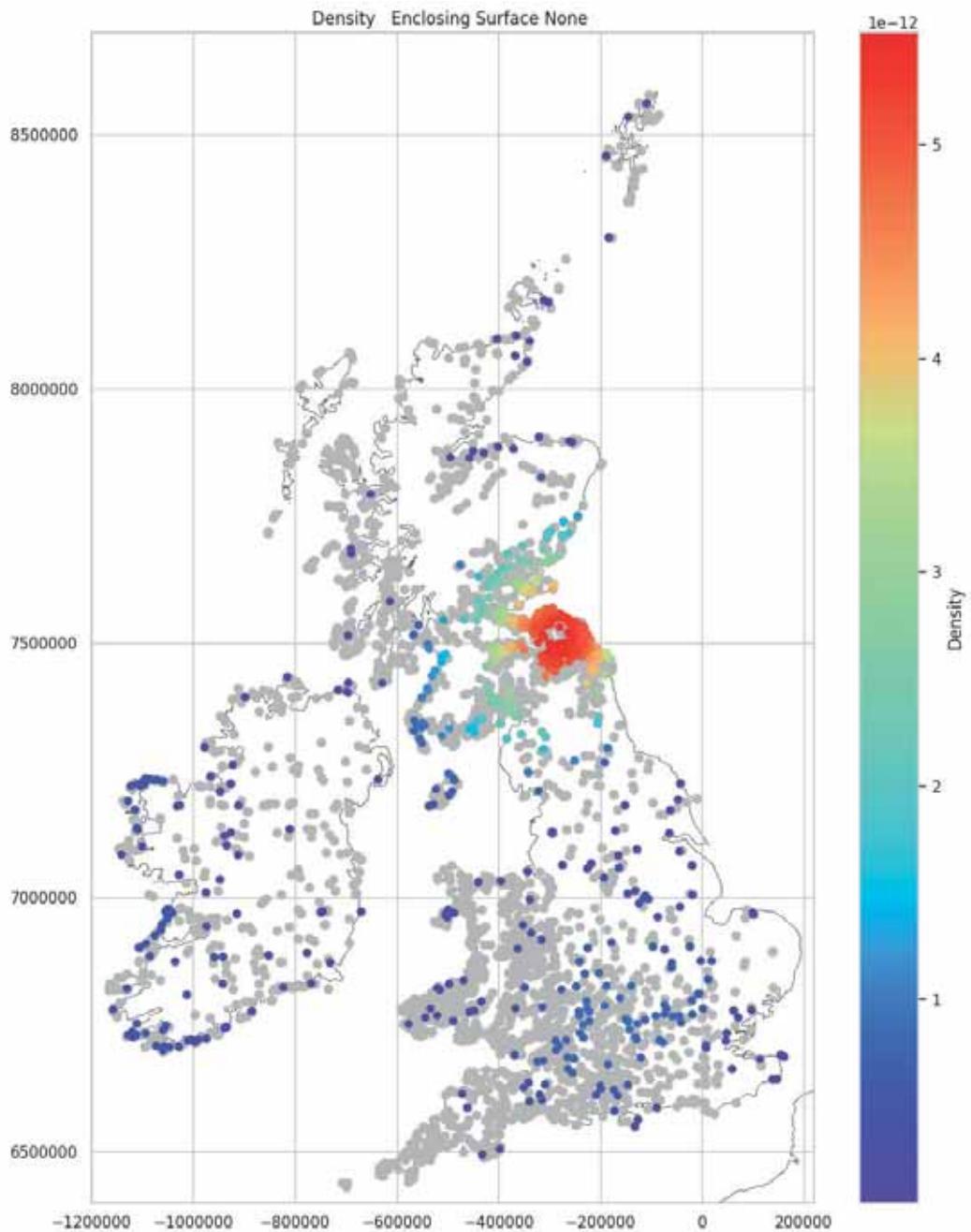
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

16. 93%

Enclosing Surface None Density Mapped

Most hillforts, which have no information regarding the enclosing circuit, are in the Northeast.

```
In [ ]: plot_density_over_grey(surface_none_data_yes, 'Enclosing_Surface_None')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Bank Mapped

1782 (42.97%) of hillforts have an enclosing bank. What is most noticeable from this is how few forts, north and west of the Highland Boundary Fault, fall into this class.

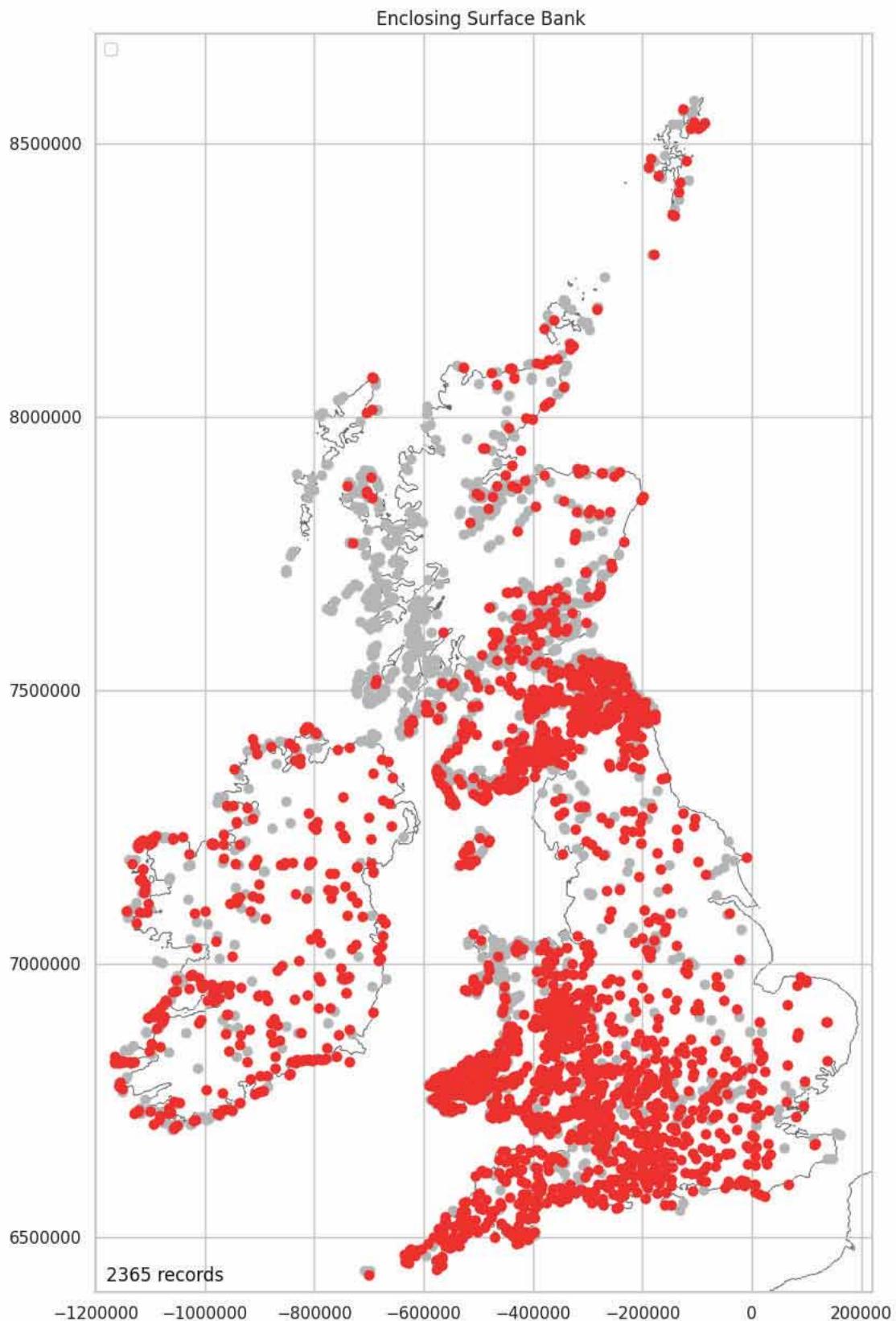
```
In [ ]: surface_bank_counts = \
enclosi ng_encodeabl e_data['Encl osing_Surface_Bank'].val ue_counts()
surface_bank_counts
```

```
Out[ ]: Yes    2365
No     1782
Name: Encl osing_Surface_Bank, dtype: int64
```

```
In [ ]: print(f' {round(surface_bank_counts[1]/len(enclosi ng_encodeabl e_data)*100, 2)}%')
```

42.97%

```
In [ ]: surface_bank_data_yes = \
plot_over_grey(locate n_enclosi ng_encodeabl e_data, \
'Encl osing_Surface_Bank', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

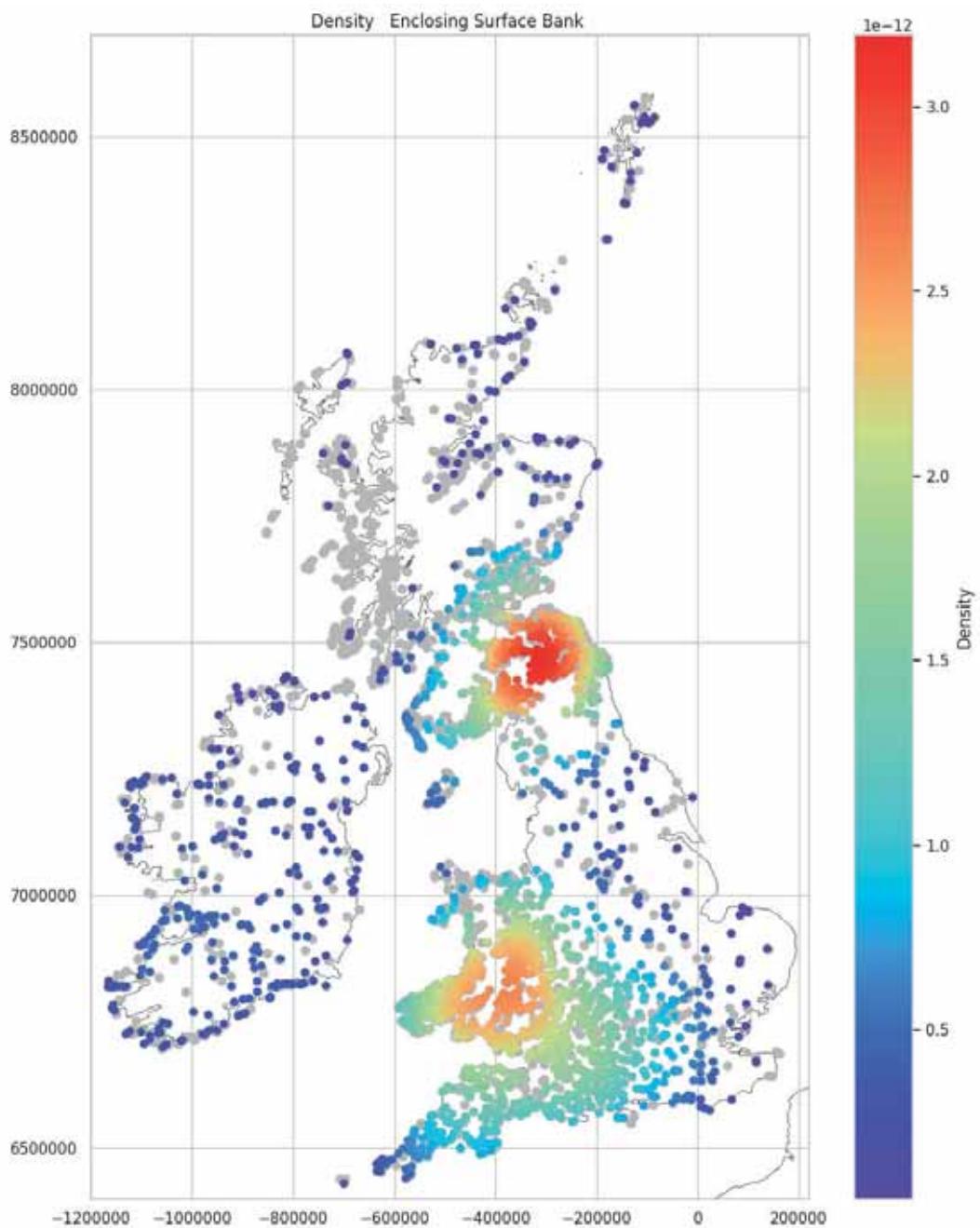
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

57.03%

Enclosing Surface Bank Density Mapped

There are two main clusters in this class. The most intense is in the Northeast while the second is to the southern end of the Cambrian Mountains. There looks to be a relatively even distribution of this class across the whole of Ireland.

```
In [ ]: plot_density_over_grey(surface_bank_data_yes, 'Enclosing_Surface_Bank')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Wall Mapped

987 (23.8%) of hillforts have an enclosing wall. Unsurprisingly, these are located predominantly in the areas of hard, exposed geology.

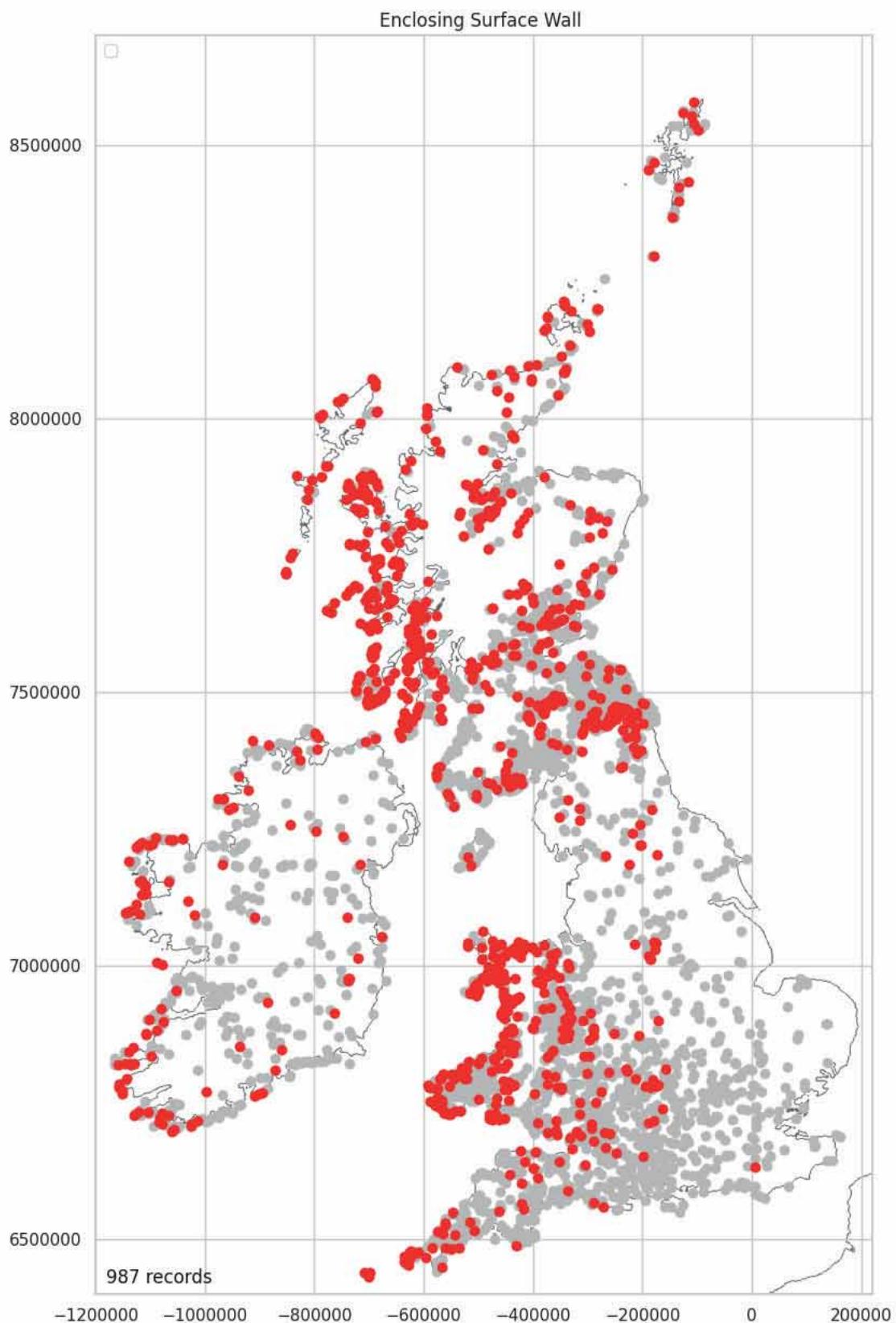
```
In [ ]: surface_wall_counts = enclosing_encodeable_data['Enclosing_Surface_Wall'].value_counts()
```

```
Out[ ]: No      3160
Yes     987
Name: Enclosing_Surface_Wall, dtype: int64
```

```
In [ ]: print(f'{round(surface_wall_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
```

23.8%

```
In [ ]: surface_wall_data_yes = \
plot_over_grey(location_enclosing_encodeable_data, \
'Enclosing_Surface_Wall', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

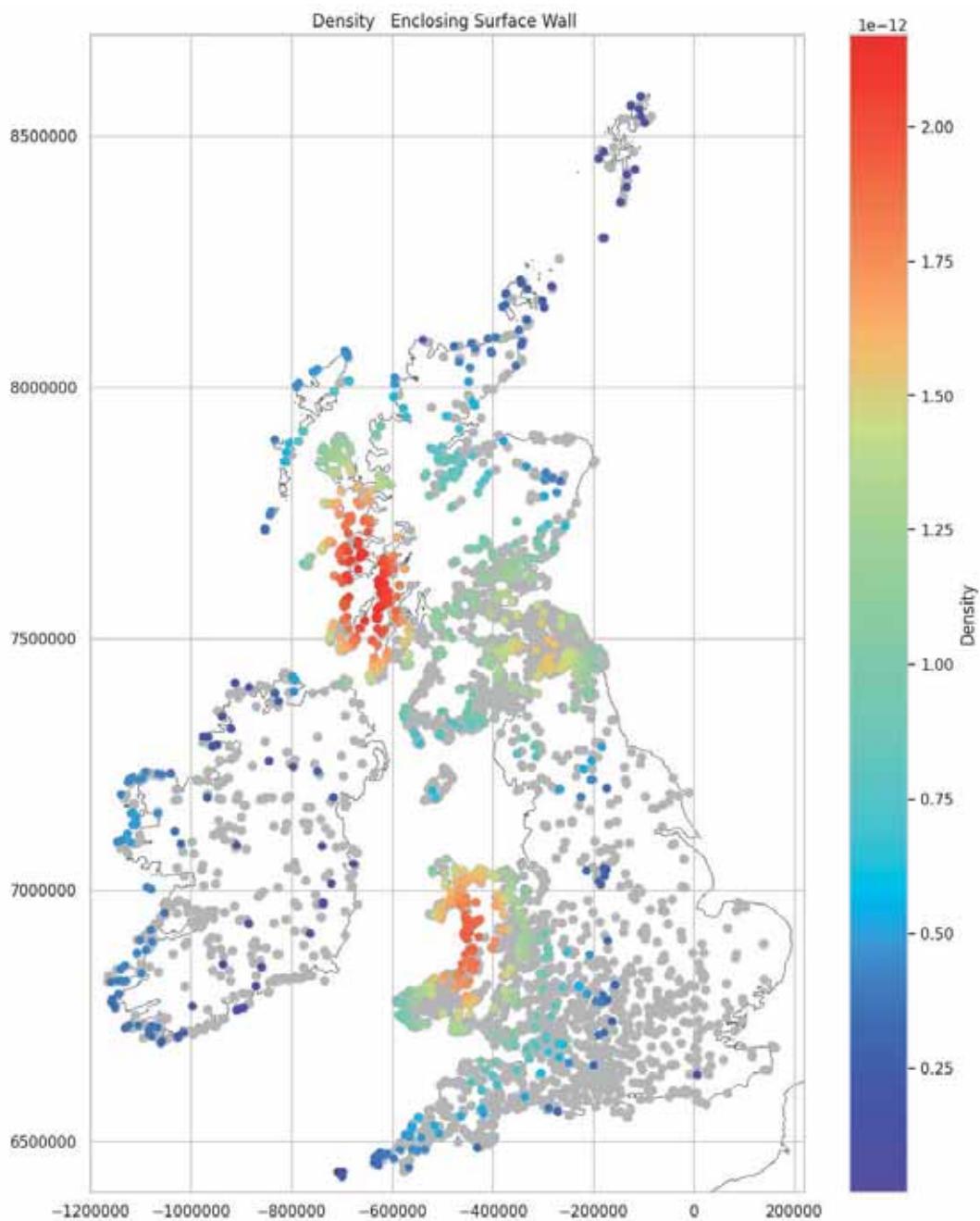
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

23.8%

Enclosing Surface Wall Density Mapped

Walls are focussed, most intensely, in the Northwest and in northwest Wales. There is a small cluster in the Northeast. In Ireland, coastal forts dominate the local distribution.

```
In [ ]: plot_density_over_grey(surface_wall_data_yes, 'Enclosing_Surface_Wall')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Rubble Mapped

659 (15.89%) of hillforts have a rubble enclosing circuit.

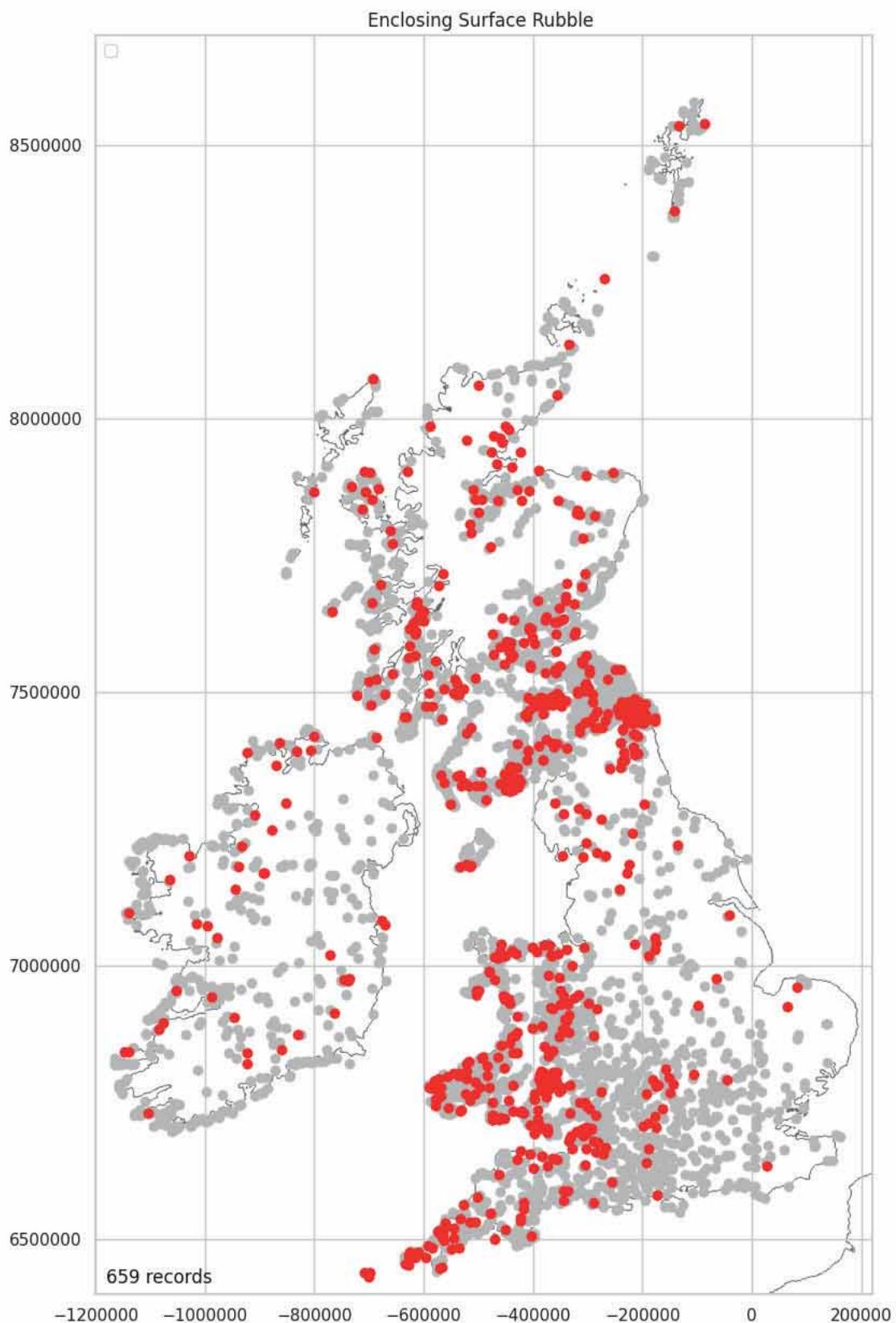
```
In [ ]: surface_rubble_counts = \
enclosi ng_encodeable_data['Encl osing_Surfac e_Rubbl e'].value_counts()
surface_rubble_counts
```

```
Out[ ]: No      3488
Yes     659
Name: Encl osing_Surfac e_Rubbl e, dtype: int64
```

```
In [ ]: print(f'{round(surface_rubble_counts[1]/len(enclosi ng_encodeable_data)*100, 2)}%')
```

15.89%

```
In [ ]: surface_rubble_data_yes = \
plot_over_grey(locate n_encl osing_encodeable_data, \
'Encl osing_Surfac e_Rubbl e', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

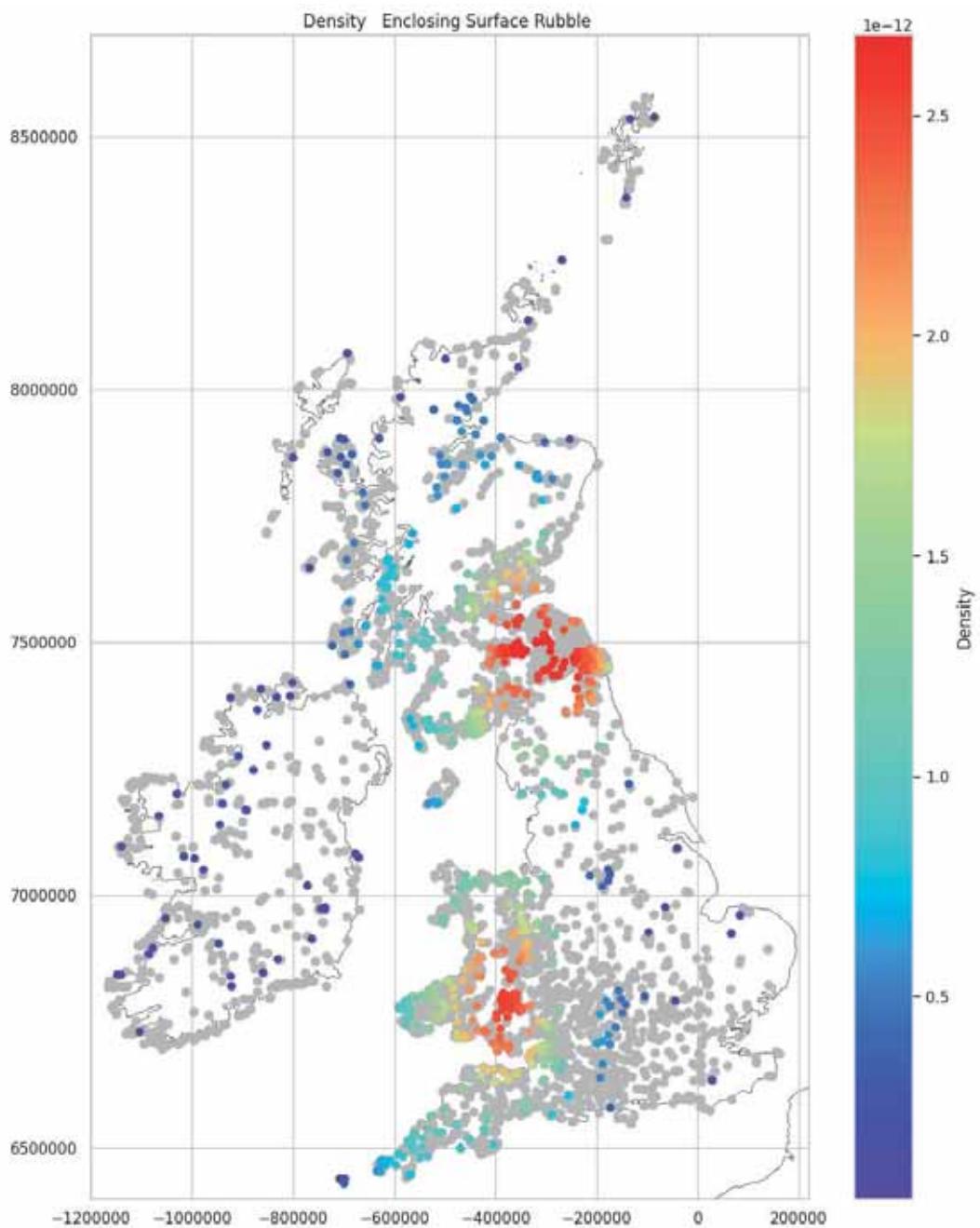
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

15.89%

Enclosing Surface Rubble Density Mapped

This class has two main clusters. The first in the Northeast and a second focussed over the Brecon Beacons, in the South.

```
In [ ]: plot_density_over_grey(surface_rubble_data_yes, 'Enclosing_Surface_Rubble')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Walk Mapped

Just 15 (0.36%) hillforts have evidence for a Surface Walk.

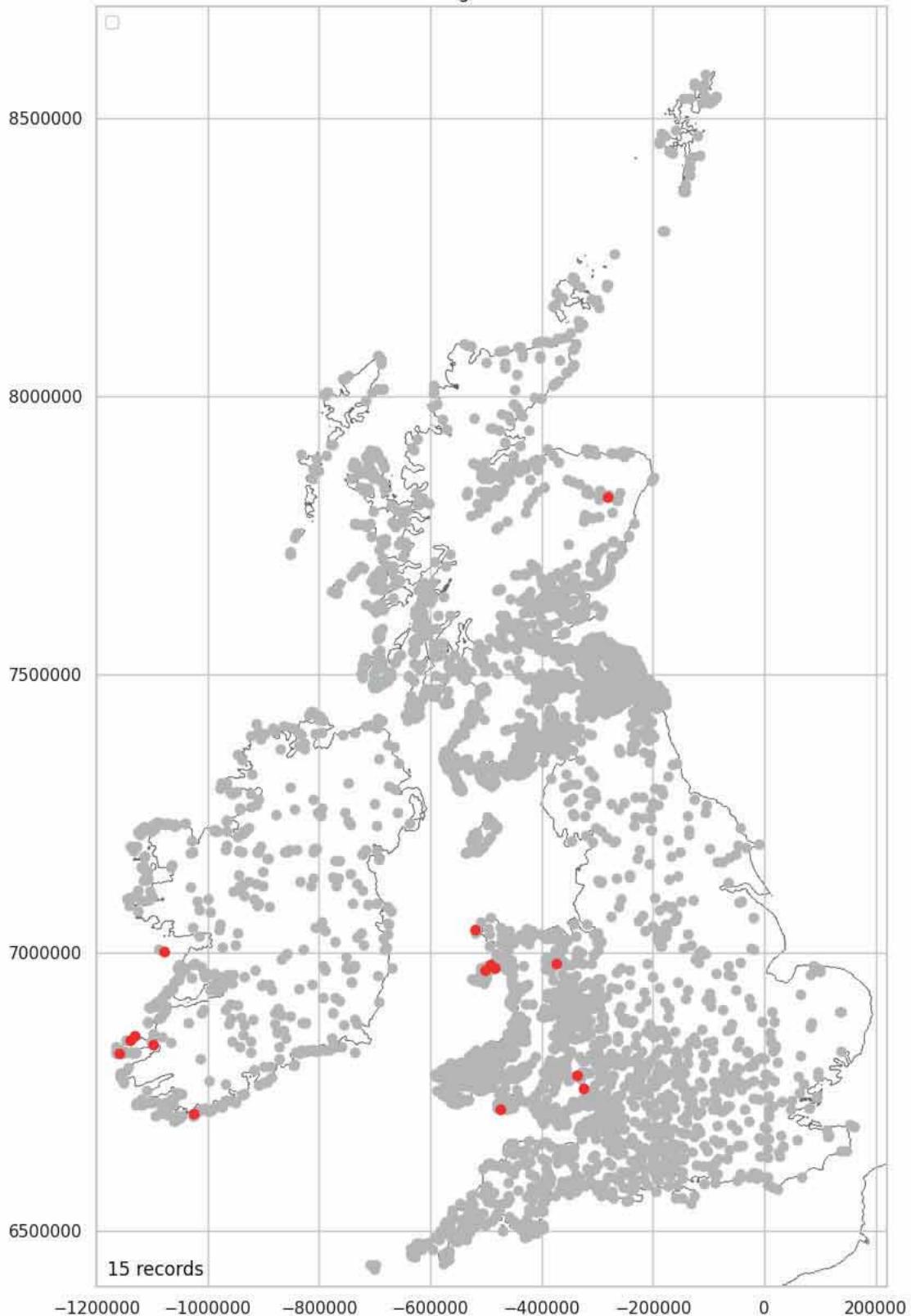
```
In [ ]: surface_walk_counts = \
enclosi ng_encodeable_data['Encl osing_Surfac e_Walk'].value_counts()
surface_walk_counts
```

```
Out[ ]: No      4132
Yes      15
Name: Encl osing_Surfac e_Walk, dtype: int64
```

```
In [ ]: print(f'{round(surface_walk_counts[1]/len(enclosi ng_encodeable_data)*100, 2)}%')
0.36%
```

```
In [ ]: surface_walk_data_yes = \
plot_over_grey(locate n_encl osing_encodeable_data, \
'Encl osing_Surfac e_Walk', 'Yes', '')
```

Enclosing Surface Walk



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.36%

Enclosing Surface Timber Mapped

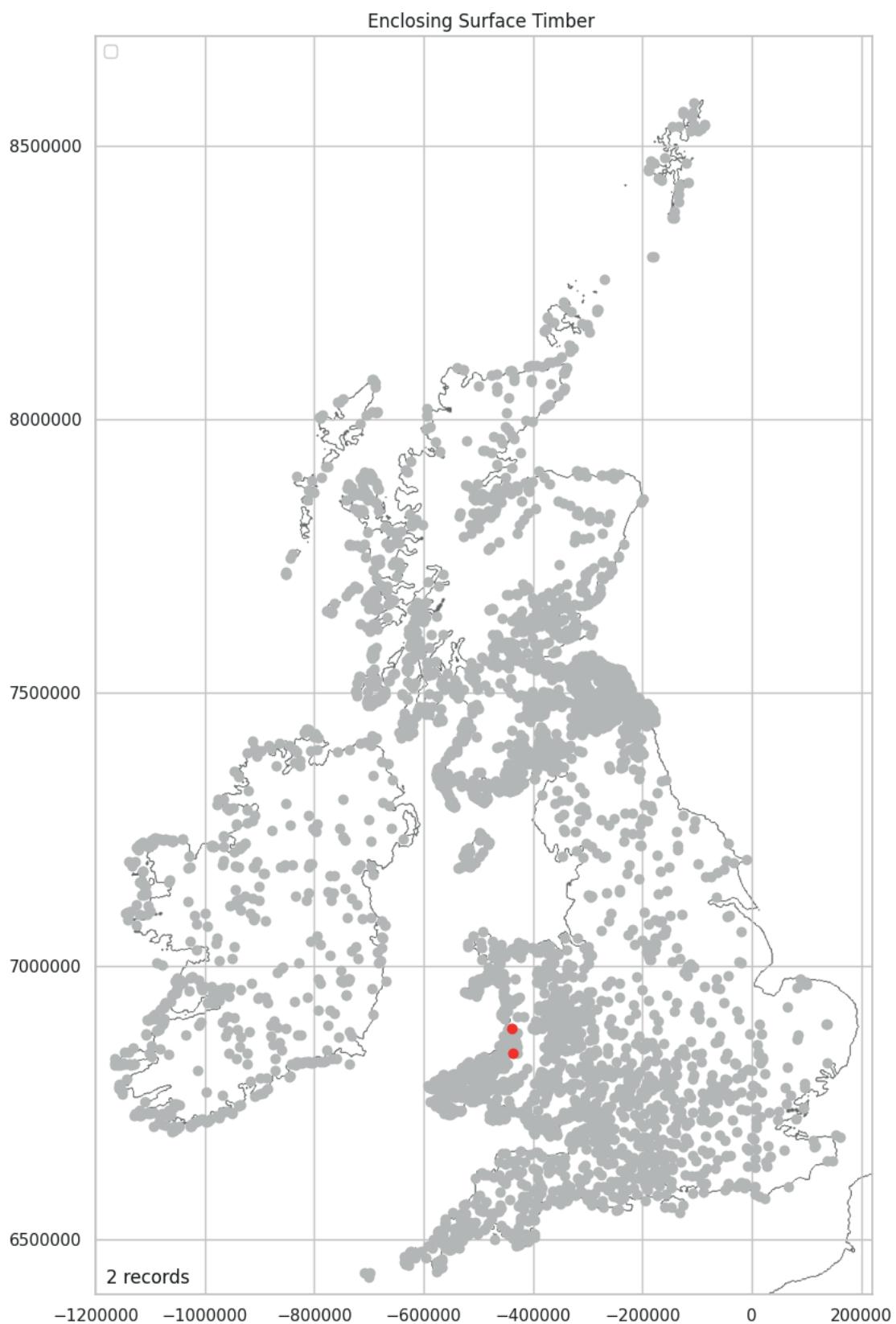
Only 2 hillforts have evidence for Surface Timber.

```
In [ ]: surface_timber_counts = \
enclosi ng_encodeabl e_data['Encl osing_Surface_Ti mber'].value_counts()
surface_timber_counts
```

```
Out[ ]: No      4145  
         Yes      2  
         Name: Enclosing_Surface_Timber, dtype: int64
```

```
In [ ]: print(f' {round(surface_timber_counts[1]/len(encl osing_encodeable_data)*100, 2)}%')  
0.05%
```

```
In [ ]: surface_timber_data_yes = \  
plot_over_grey(location_encl osing_encodeable_data, \  
'Enclosing_Surface_Ti mber', 'Yes', '')
```



Enclosing Surface Vitrification Mapped

88 (2.12%) hillforts show signs of vitrification. These are almost entirely in the North. See: [Enclosing Excavation Vitrification Mapped](#)

```
In [ ]: surface_vitrification_counts = \
enclosi ng_encodeabl e_data['Encl osi ng_Surface_Vi tri fi cati on'].value_counts()
```

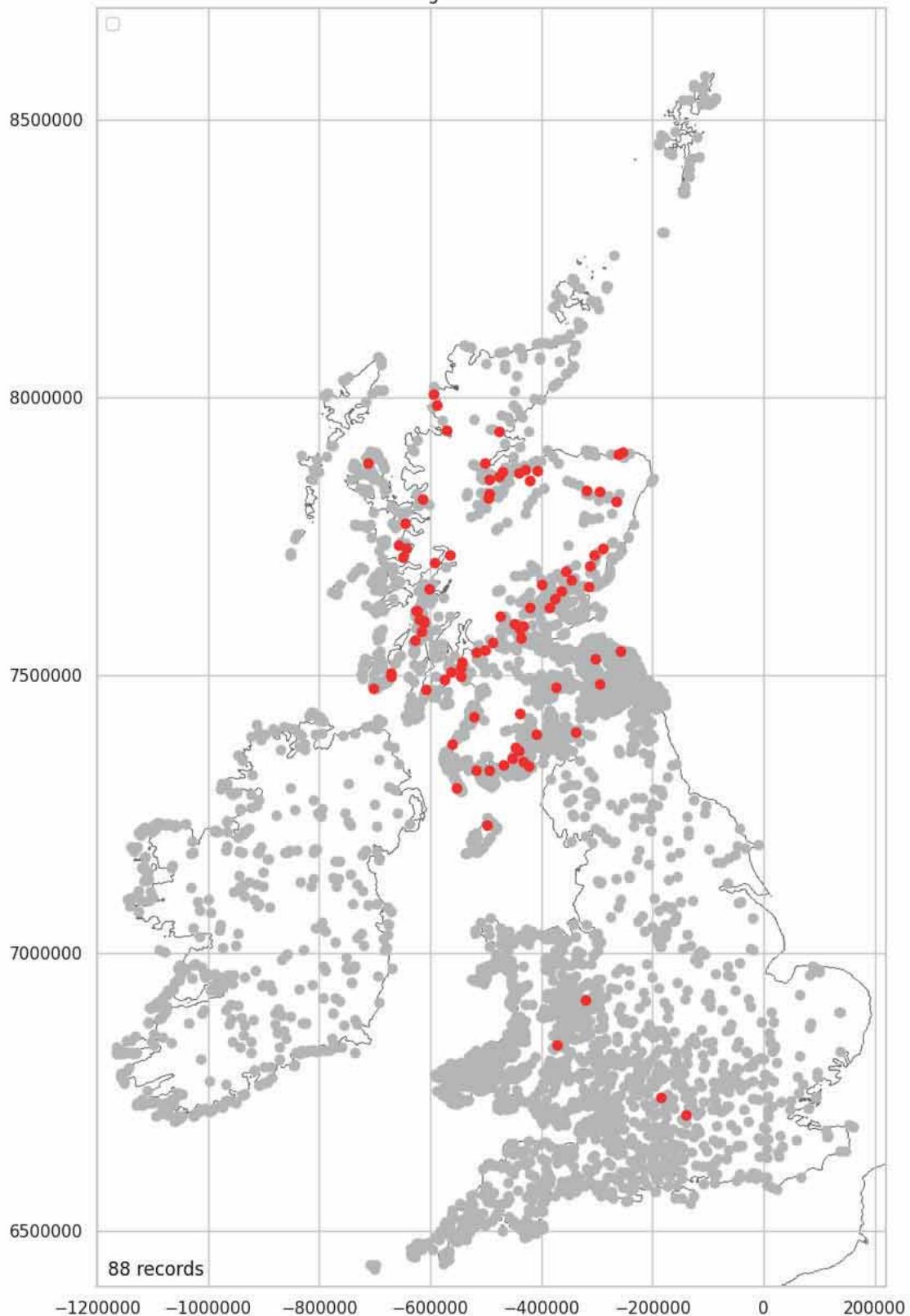
```
Out[ ]: No      4059
Yes      88
Name: Encl osi ng_Surface_Vi tri fi cati on, dtype: int64
```

```
In [ ]: print(f' {round(surface_vitrification_counts[1]/len(enclosi ng_encodeabl e_data)*100, 2)}%')
```

2.12%

```
In [ ]: surface_vitrification_data_yes = \
plot_over_grey(location_enclosi ng_encodeabl e_data, \
'Encl osi ng_Surface_Vi tri fi cati on', 'Yes', '')
```

Enclosing Surface Vitrification



Middleton, M. 2024, Hillforts Primer

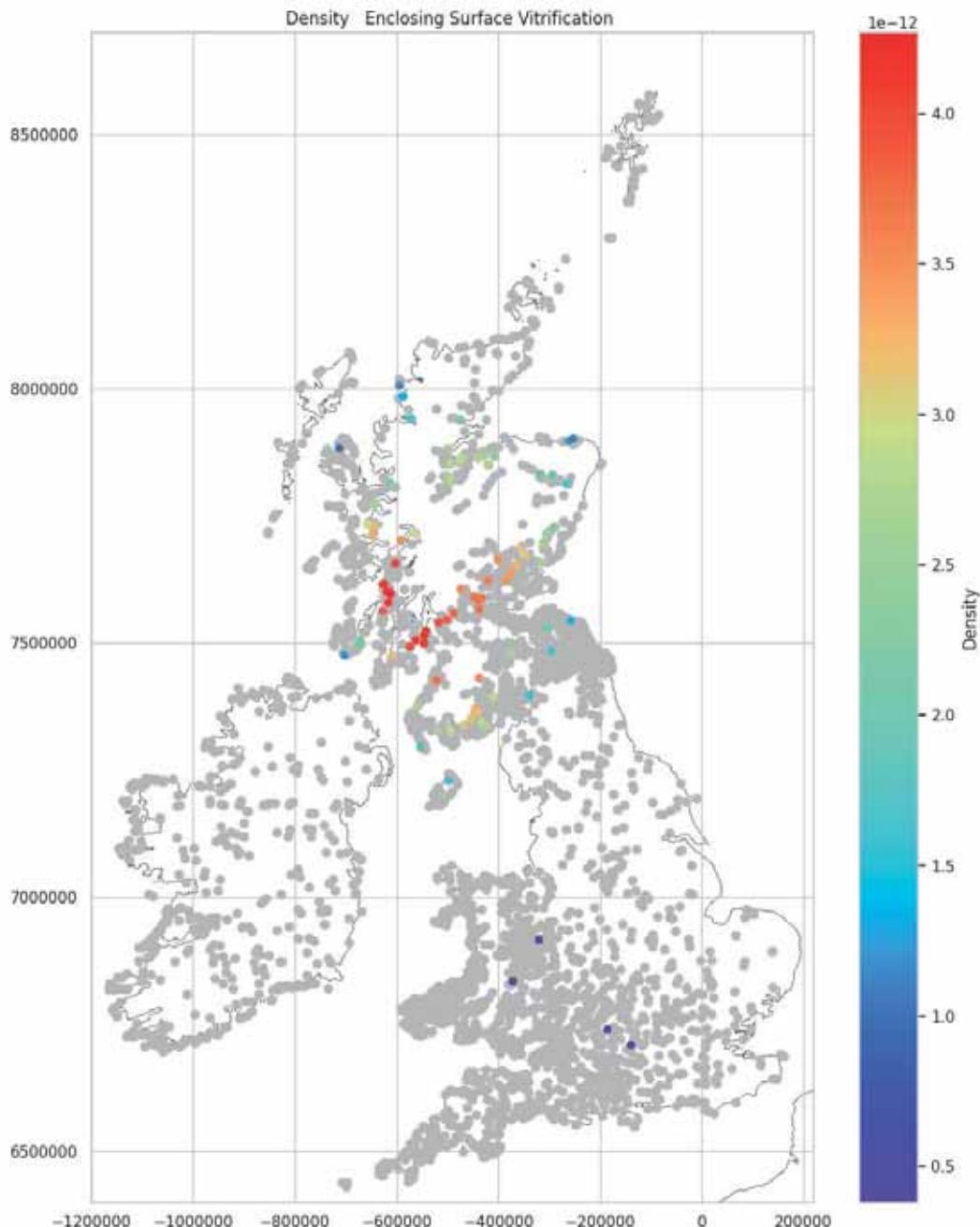
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

2. 12%

Enclosing Surface Vitrification Density Mapped

The main concentration of vitrified hillforts is in the vicinity of Dunnad, along the Clyde Valley and up the Highland Boundary Fault. This density plot has been produced using very few records and extra caution should be taken in not over interpreting these results. This class is also likely to have a recording bias in that vitrification is notorious for being misidentified. See: [Enclosing Excavation Vitrification Mapped](#).

```
In [ ]: plot_density_over_grey(surface_vitrification_data_yes, \
    'Enclosing_Surface_Vitrification')
```



Middleton, M. 2024. Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Burning Mapped

Only eight (0.19%) hillforts have signs of 'Other Burning'.

```
In [ ]: surface_burning_counts = \
    enclosing_encodeable_data['Enclosing_Surface_Burning'].value_counts()
surface_burning_counts
```

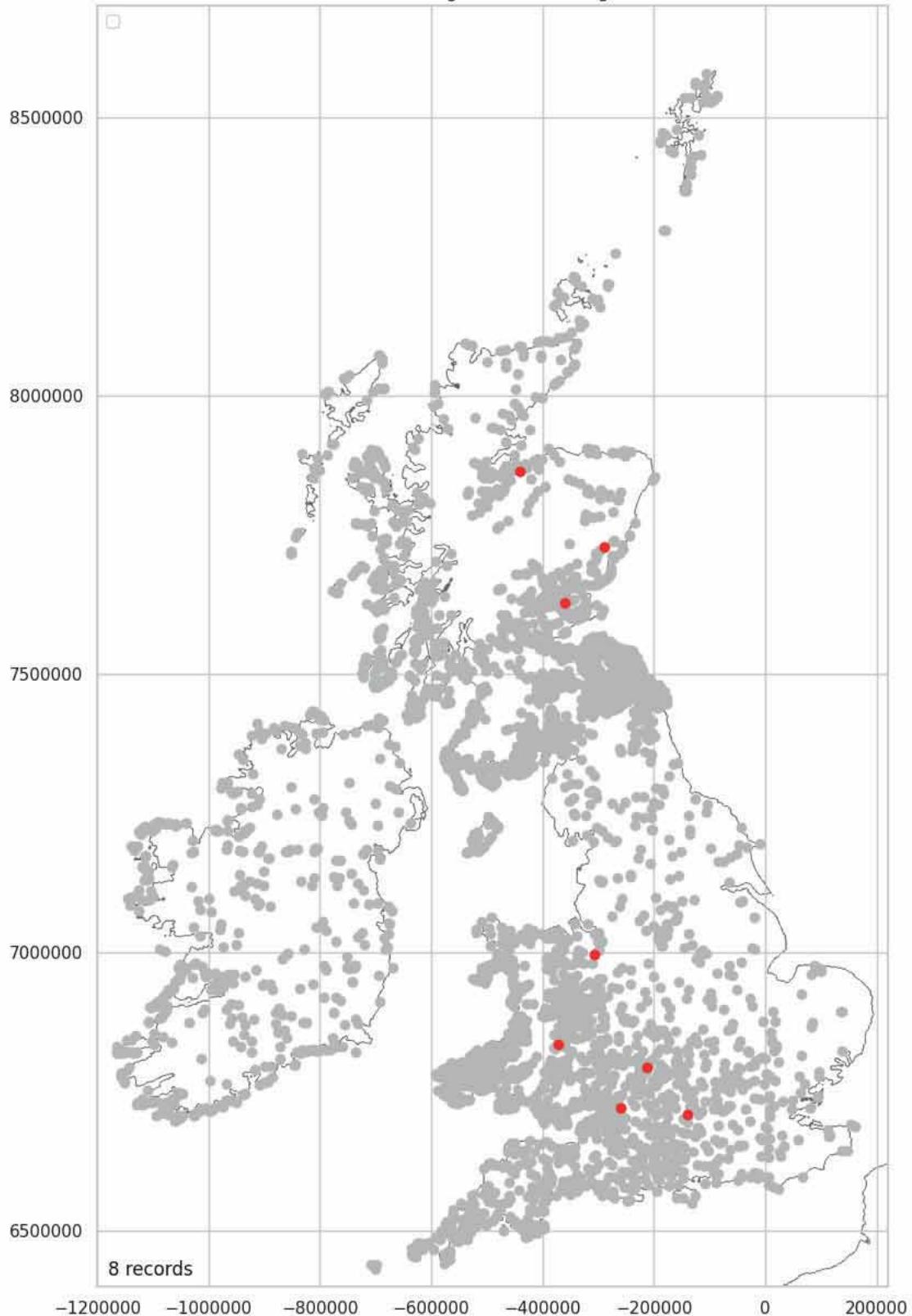
```
Out[ ]: No      4139
Yes       8
Name: Enclosing_Surface_Burning, dtype: int64
```

```
In [ ]: print(f'{round(surface_burning_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
```

0.19%

```
In [ ]: surface_burning_data_yes = \
    plot_over_grey(LOCATION_ENCLOSING_ENCODEABLE_DATA, \
        'Enclosing_Surface_Burning', 'Yes', '')
```

Enclosing Surface Burning



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.19%

Enclosing Surface Palisade Mapped

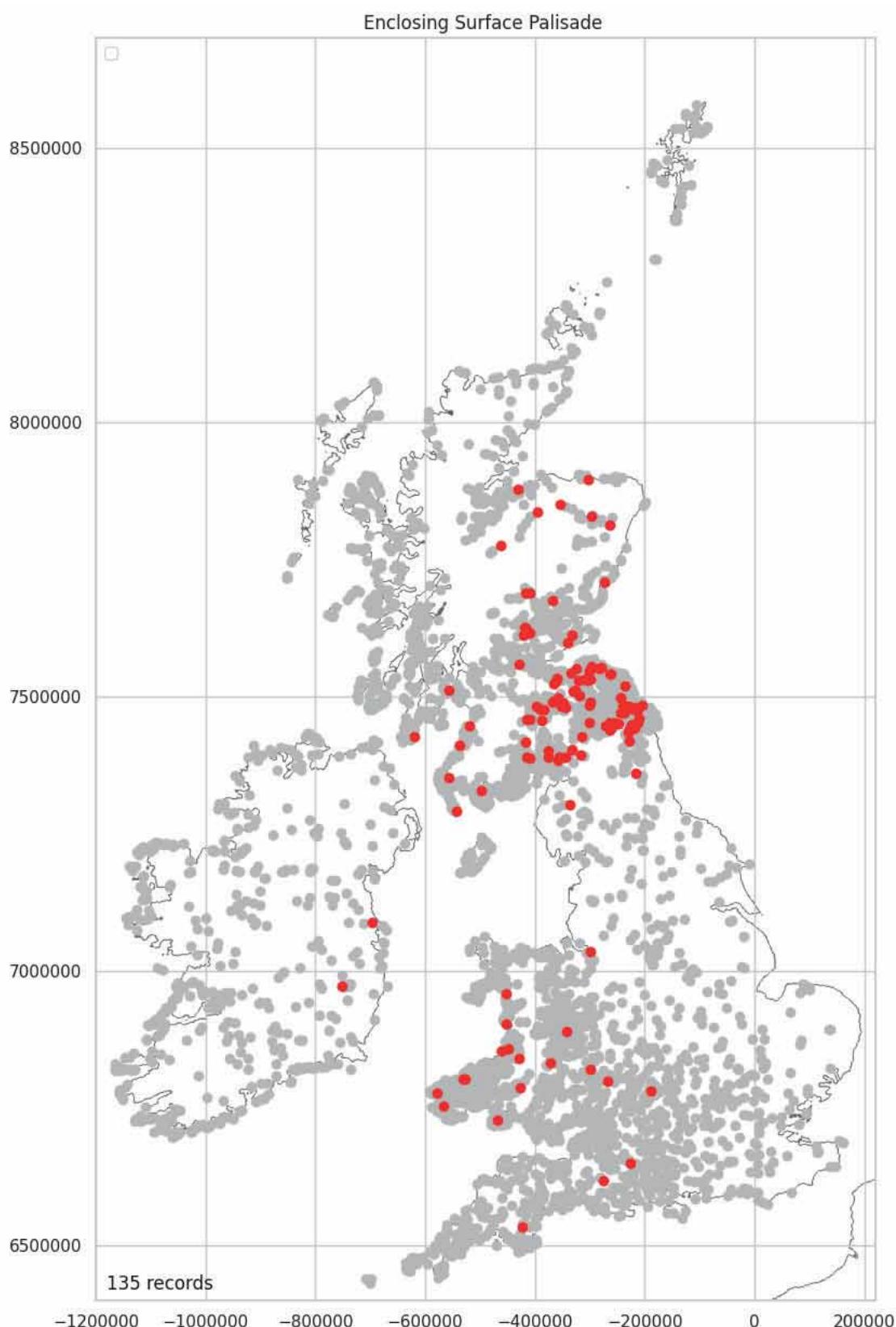
135 (3.26%) of hillforts have recorded evidence for a palisade.

```
In [ ]: surface_palisade_counts = \
enclosi ng_encodeable_data['Encl osing_Surface_Pal i sade'].value_counts()
surface_palisade_counts
```

```
Out[ ]: No      4012  
         Yes     135  
         Name: Enclosing_Surface_Palisade, dtype: int64
```

```
In [ ]: print(f' {round(surface_palisade_counts[1]/len(enclosing_encodeable_data)*100, 2)}% )  
3.26%
```

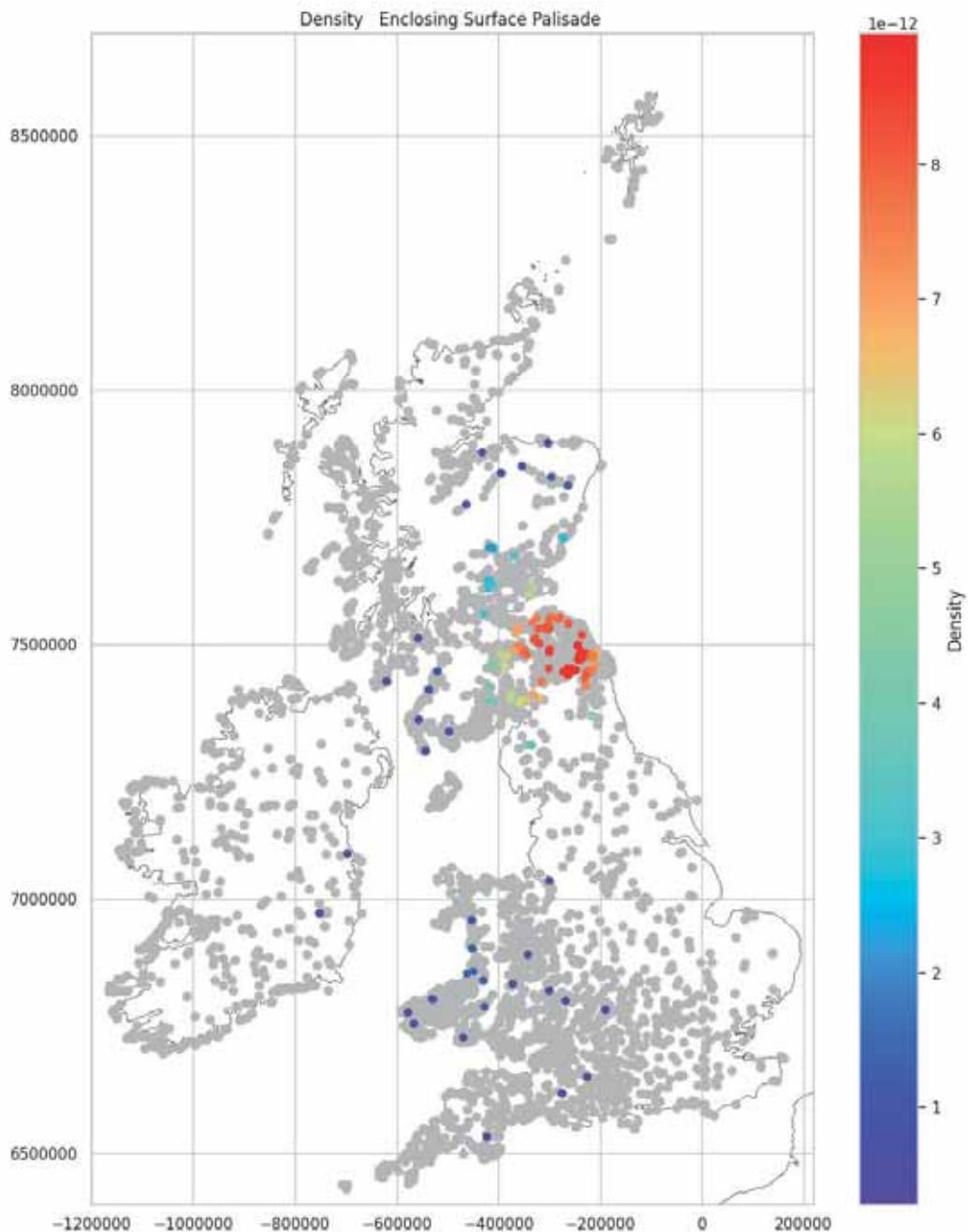
```
In [ ]: surface_palisade_data_yes = \  
plot_over_grey(location_enclosing_encodeable_data, \  
'Enclosing_Surface_Palisade', 'Yes', '')
```



Enclosing Surface Palisade Density Mapped

The main cluster for palisades is in the Northeast. Due to the ephemeral nature of these features this class is likely to have a recording bias toward areas where surveyors have been trained to identify these features.

```
In [ ]: plot_density_over_grey(surface_palisade_data_yes, 'Enclosing_Surface_Palisade')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Counter Scarp Mapped

561 (13.53%) of hillforts have a counterscarp. It is assumed that a counterscarp requires the presence of a ditch although there are ten hillforts where this is not the case.

```
In [ ]: surface_scarp_counts = \
enclosi ng_encodeabl e_data['Encl osing_Surface_Counter_Scarp'].value_counts()
surface_scarp_counts
```

```
Out[ ]: No      3586
Yes     561
Name: Encl osing_Surface_Counter_Scarp, dtype: int64
```

```
In [ ]: print(f' {round(surface_scarp_counts[1]/len(enclosi ng_encodeabl e_data)*100, 2)}%')
```

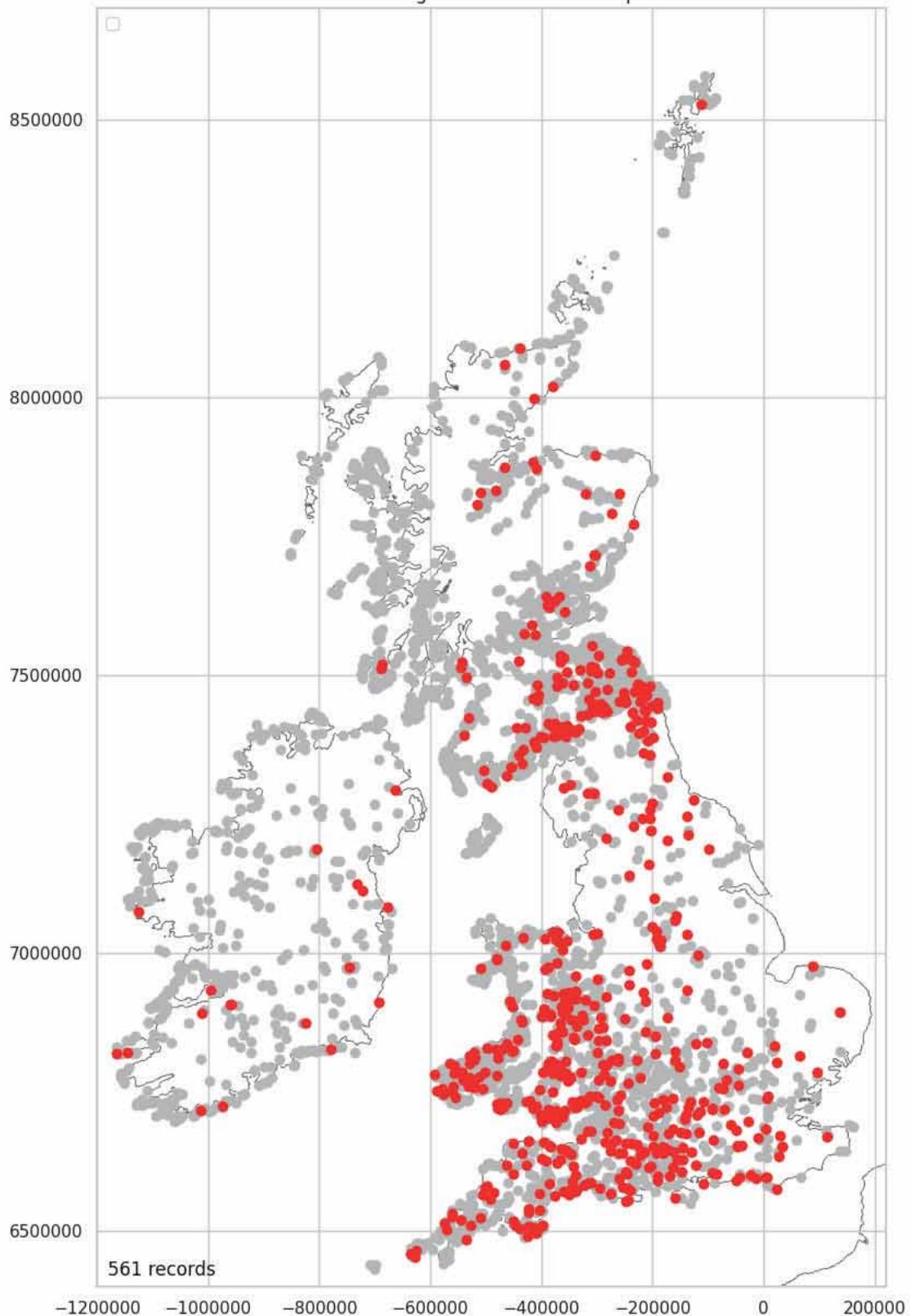
13. 53%

```
In [ ]: counterscarp_ditch = \
len(encl osi ng_data[(encl osi ng_data['Encl osi ng_Surface_Counter_Scarp'] == "Yes") & \
(encl osi ng_data['Encl osi ng_Di tches_Number'] > 0)])
counterscarp_ditch
```

Out[]: 551

```
In [ ]: surface_scarp_data_yes = \
plot_over_grey(location_encl osi ng_encodeable_data, \
'Encl osi ng_Surface_Counter_Scarp', 'Yes', '')
```

Enclosing Surface Counter Scarp



Middleton, M. 2024, Hillforts Primer

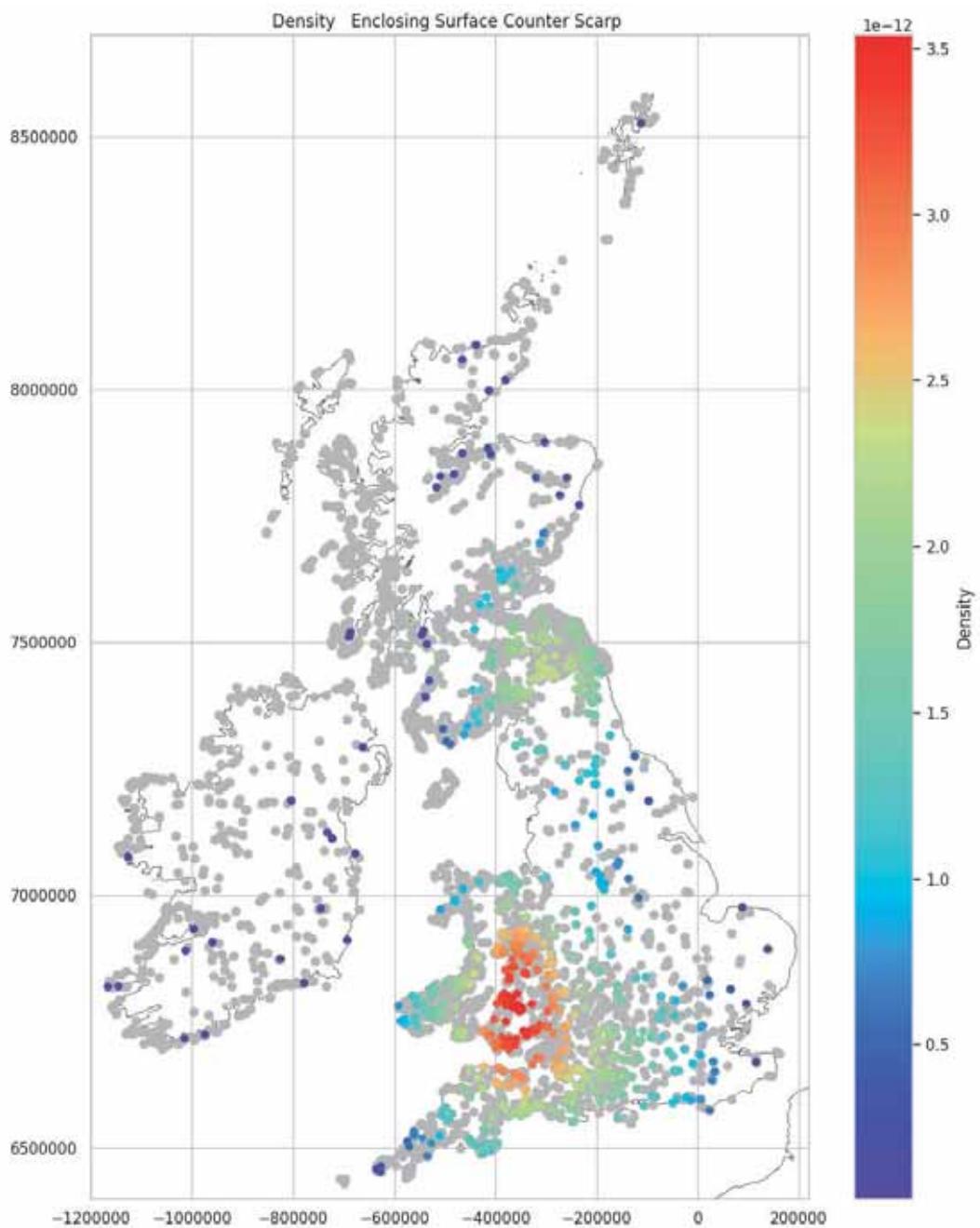
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

13.53%

Enclosing Surface Counter Scarp Density Mapped

The main cluster of hillforts with a counterscarp is over the Brecon Beacons and up along the eastern fringe of the Cambrian Mountains.

```
In [ ]: plot_density_over_grey(surface_scarp_data_yes, \
    'Enclosing_Surface_Counter_Scarp')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Berm Mapped

There are 136 (3.28%) hillforts where a berm has been recorded. The distribution is unusual and is likely the result of a recording bias across the south of England and up along the Welsh border.

```
In [ ]: surface_burm_counts = \
enclosi ng_encodeabl e_data['Encl osing_Surface_Berm'].val ue_counts()
surface_burm_counts
```

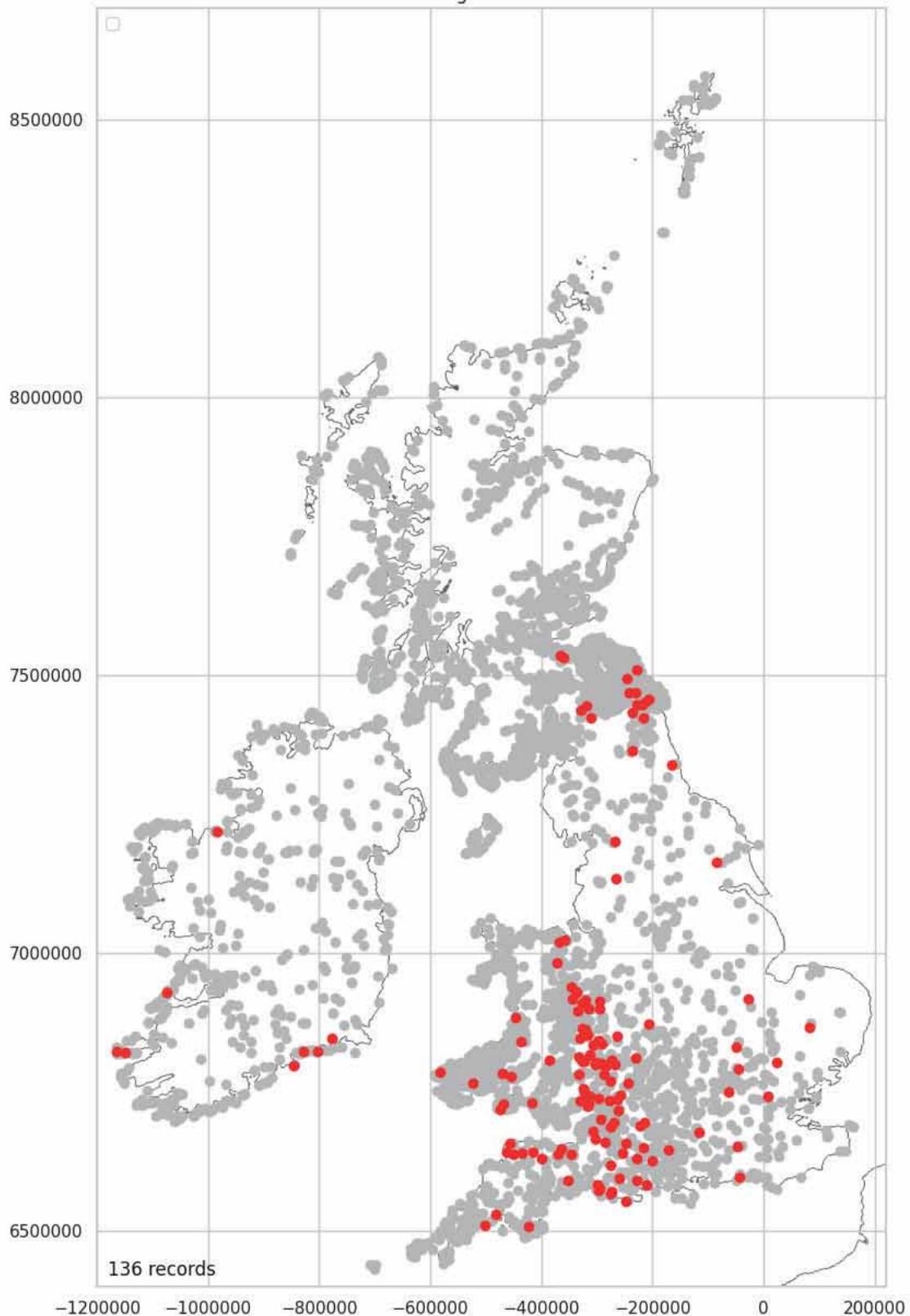
```
Out[ ]: No      4011
Yes     136
Name: Encl osing_Surface_Berm, dtype: int64
```

```
In [ ]: print(f'{round(surface_burm_counts[1]/len(enclosi ng_encodeabl e_data)*100, 2)}%')
```

3.28%

```
In [ ]: surface_burm_data_yes = \
plot_over_grey(location_enclosi ng_encodeabl e_data, \
'Encl osing_Surface_Berm', 'Yes', '')
```

Enclosing Surface Berm



Middleton, M. 2024, Hillforts Primer

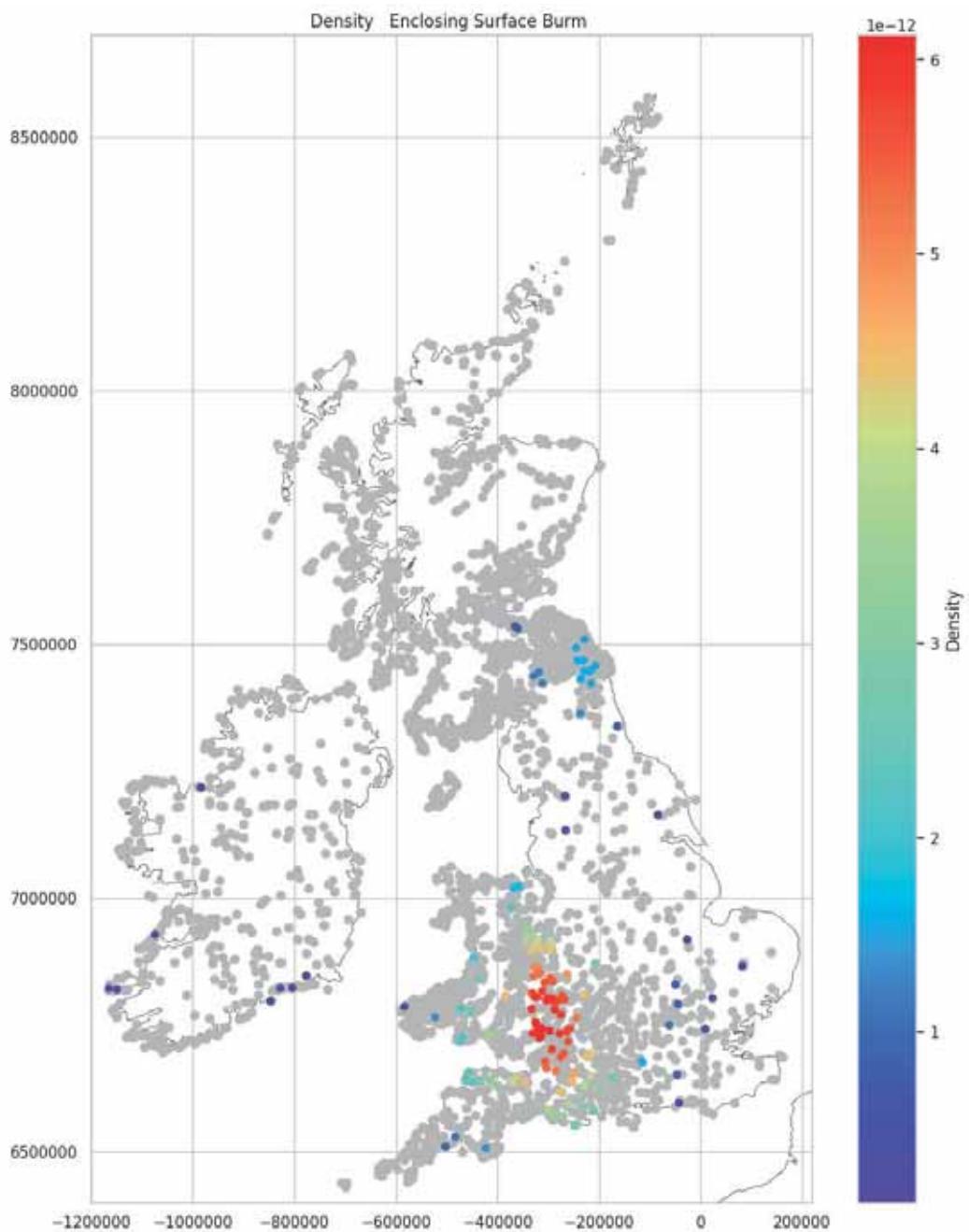
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

3. 28%

Enclosing Surface Berm Density Mapped

This cluster is likely to be highly biased and should only be used with caution.

```
In [ ]: plot_density_over_grey(surface_berm_data_yes, 'Enclosing_Surface_Berm')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Unfinished Mapped

175 (4.22%) of hillforts have an enclosing surface that has been recorded as unfinished.

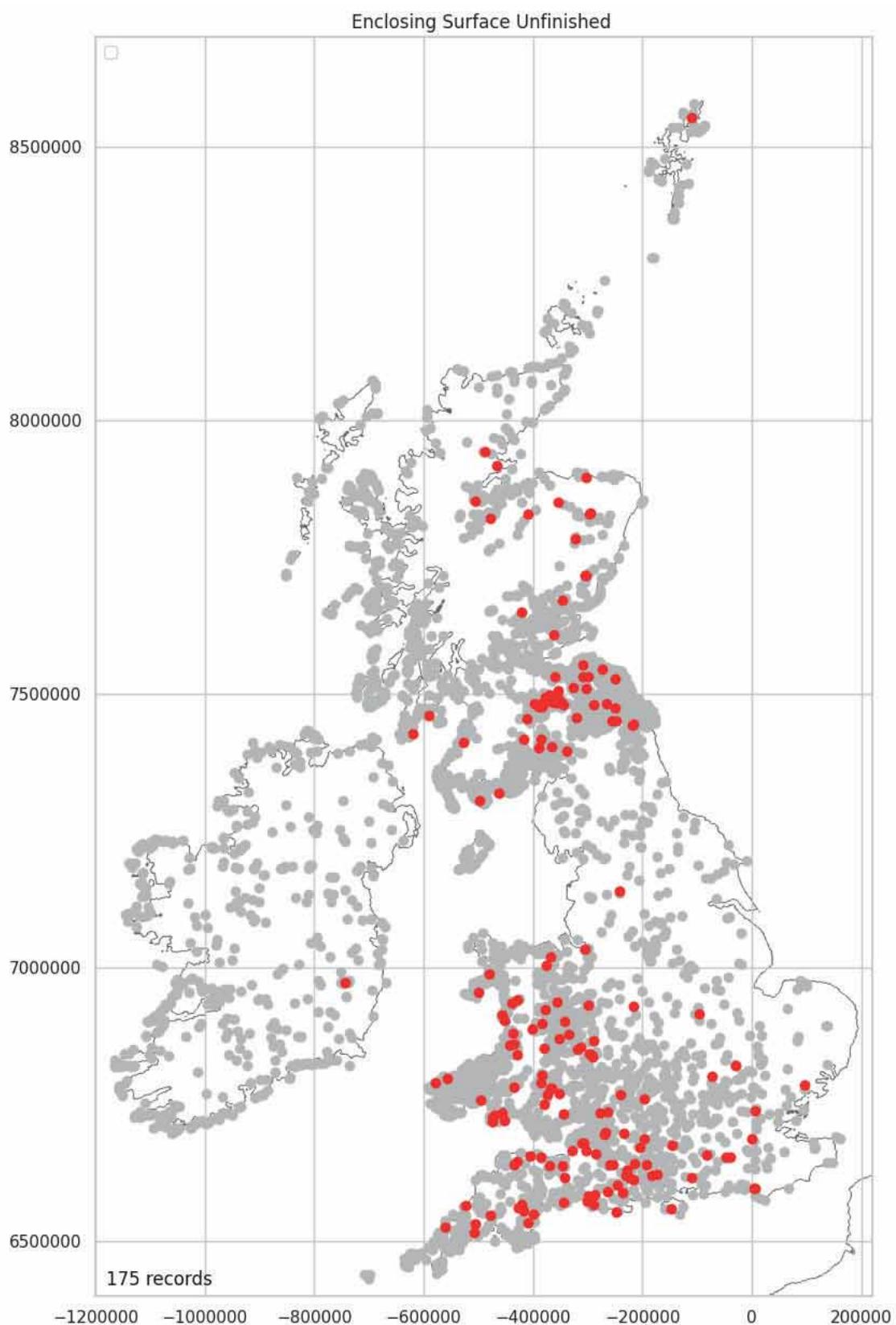
```
In [ ]: surface_unfinished_counts = \
enclosi ng_encodeable_data['Encl osing_Surface_Unfi ni shed'].value_counts()
surface_unfinished_counts
```

```
Out[ ]: No      3972
Yes     175
Name: Encl osing_Surface_Unfi ni shed, dtype: int64
```

```
In [ ]: print(f'{round(surface_unfinished_counts[1]/len(enclosi ng_encodeable_data)*100, 2)}%')
```

4.22%

```
In [ ]: surface_unfinished_data_yes = \
plot_over_grey(locate n_encl osing_encodeable_data, \
'Encl osing_Surface_Unfi ni shed', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

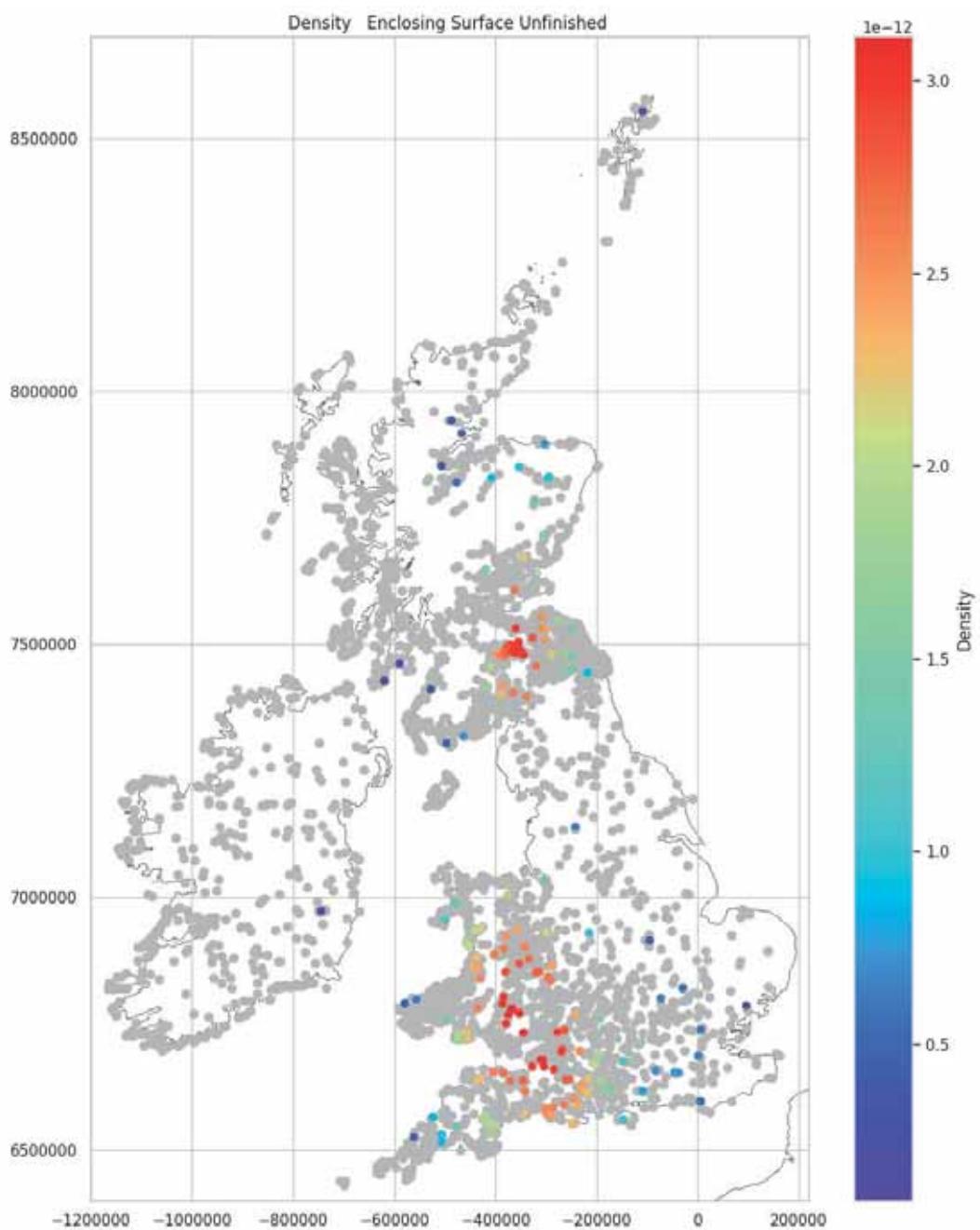
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 22%

Enclosing Surface Unfinished Density Mapped

There are two clusters. A diffuse cluster in the South and a small cluster in the North. Due to the small number of records used to create these clusters, caution should be taken to not over interpret these results

```
In [ ]: plot_density_over_grey(surface_unfinished_data_yes, \
    'Enclosing_Surface_Unfinished')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Surface Other Mapped

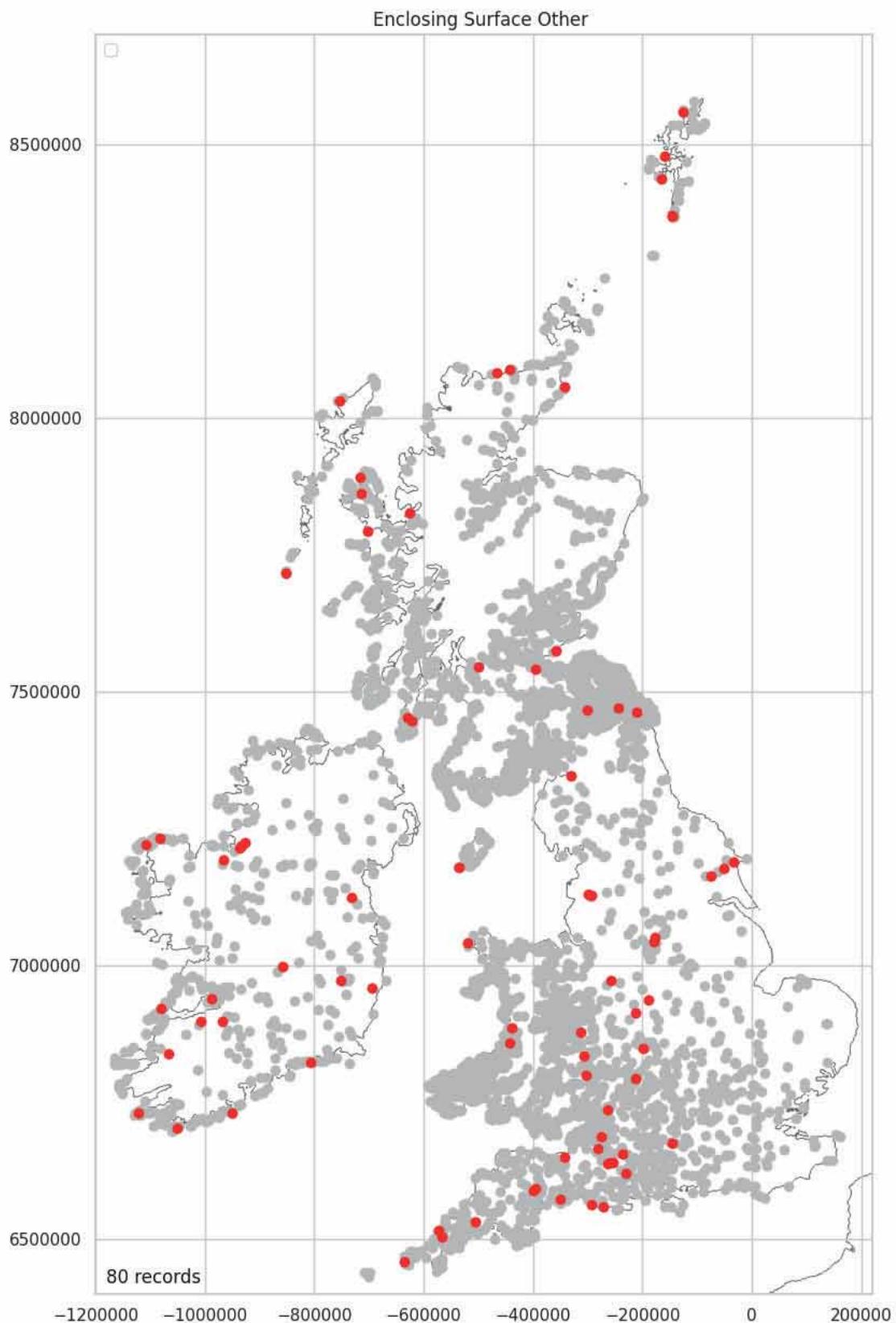
80 (4.22%) of hillforts have an unclassified enclosing surface.

```
In [ ]: surface_other_counts = \
encl osing_encodeable_data['Encl osing_Surface_Other'].value_counts()
surface_other_counts
```

```
Out[ ]: No      4067
Yes      80
Name: Encl osing_Surface_Other, dtype: int64
```

```
In [ ]: print(f'{round(surface_unfinished_counts[1]/len(encl osing_encodeable_data)*100, 2)}%')
4.22%
```

```
In [ ]: surface_other_data_yes = \
plot_over_grey(locate_encl osing_encodeable_data, \
'Encl osing_Surface_Other', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 93%

Enclosing Surface by Region (Count)

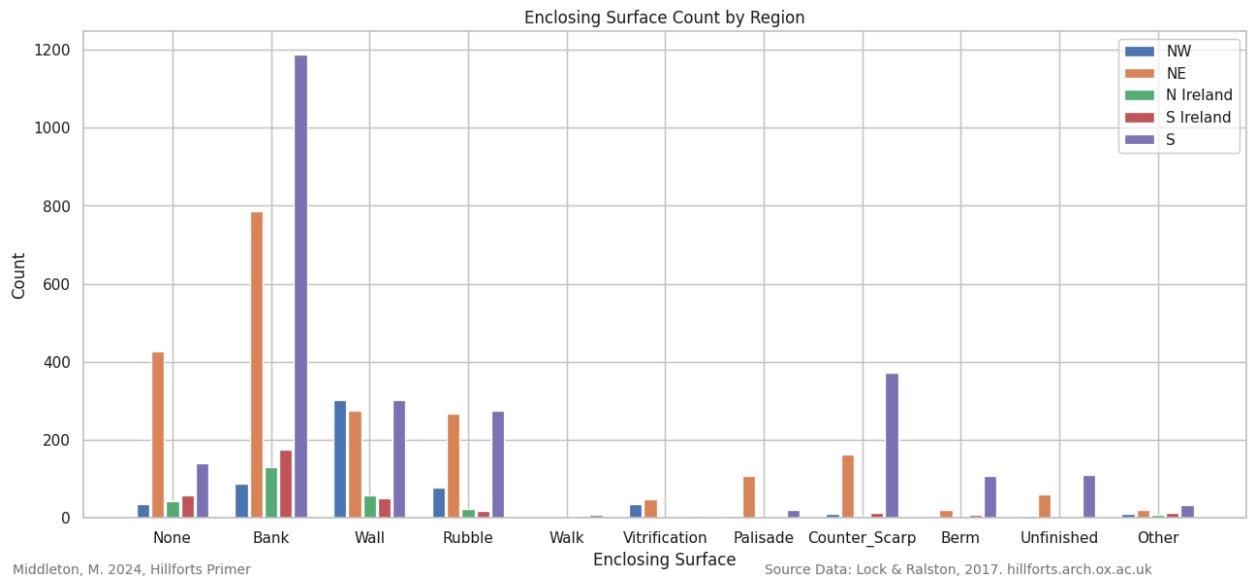
The counts for both 'Burning' and 'Timber' are in single figures and have not been included in the following plots. As was seen earlier, counts can be difficult to interpret. See below for the same data presented by proportion.

```
In [ ]: plot_regions(location_enclosing_encodeable_data_nw,
                     location_enclosing_encodeable_data_ne,
                     location_enclosing_encodeable_data_irland_n,
```

```

location_enclosing_encodeable_data_rel_and_s,
location_enclosing_encodeable_data_south,
['Enclosing_Surface_None',
'Enclosing_Surface_Bank',
'Enclosing_Surface_Wall',
'Enclosing_Surface_Rubble',
'Enclosing_Surface_Walk',
#'Enclosing_Surface_Timber',
'Enclosing_Surface_Vitrification',
#'Enclosing_Surface_Burning',
'Enclosing_Surface_Palisade',
'Enclosing_Surface_Counter_Scarp',
'Enclosing_Surface_Berm',
'Enclosing_Surface_Unfinished',
'Enclosing_Surface_Other'],
'Enclosing_Surface',
'Enclosing_Surface_Count_by_Region', 2, 'Yes')

```



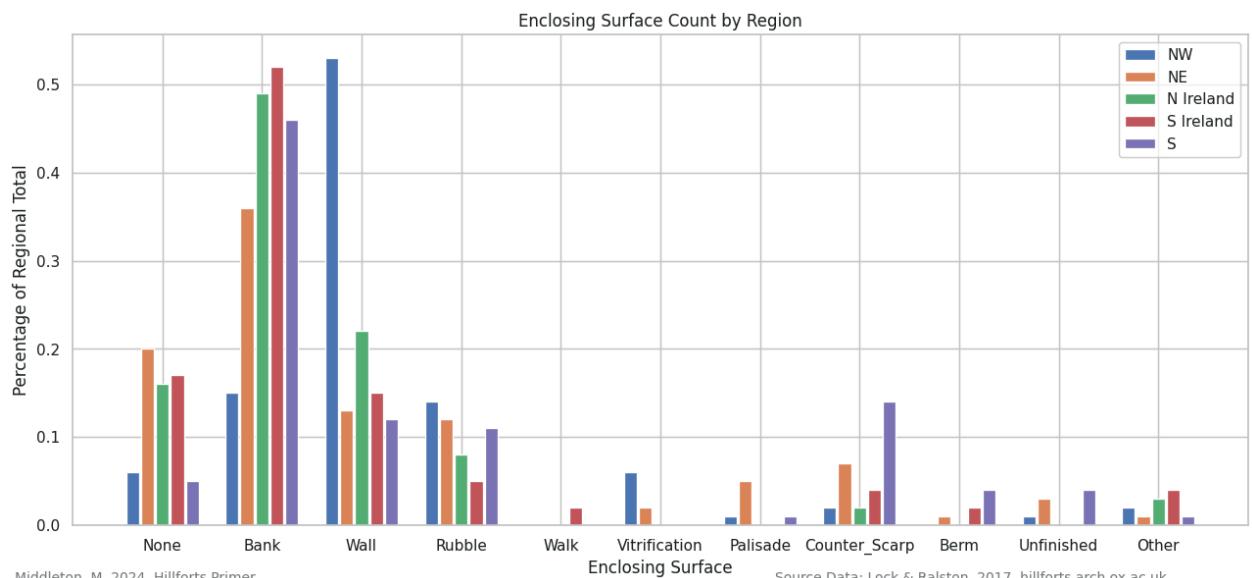
Enclosing Surface by Region (Percentage)

When plotted as a proportion of the data by region, banks dominate in the South, Northeast and across Ireland. In the Northwest walls are dominant. Walls, banks and rubble are the predominant enclosing structural forms.

```

In [ ]: plot_regions(location_enclosing_encodeable_data_nw,
location_enclosing_encodeable_data_ne,
location_enclosing_encodeable_data_ireland_n,
location_enclosing_encodeable_data_ireland_s,
location_enclosing_encodeable_data_south,
['Enclosing_Surface_None',
'Enclosing_Surface_Bank',
'Enclosing_Surface_Wall',
'Enclosing_Surface_Rubble',
'Enclosing_Surface_Walk',
#'Enclosing_Surface_Timber',
'Enclosing_Surface_Vitrification',
#'Enclosing_Surface_Burning',
'Enclosing_Surface_Palisade',
'Enclosing_Surface_Counter_Scarp',
'Enclosing_Surface_Berm',
'Enclosing_Surface_Unfinished',
'Enclosing_Surface_Other'],
'Enclosing_Surface',
'Enclosing_Surface_Count_by_Region', 2, 'Yes', True)

```



Enclosing Excavation Nothing Mapped

194 (4.22%) of hillforts have had an excavation where no enclosing circuit was identified.

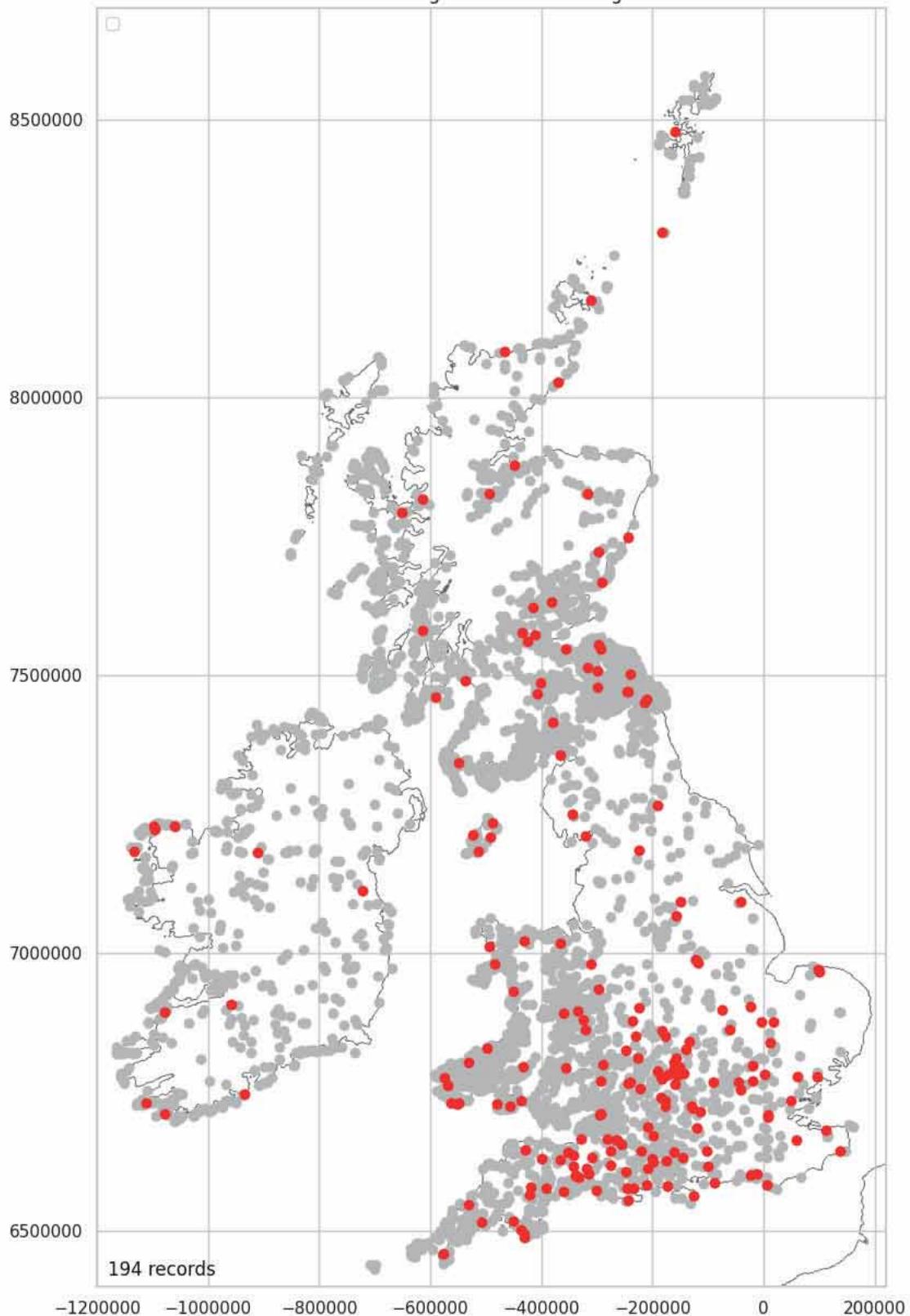
```
In [ ]: excavation_nothing_counts = \
enclosi ng_encodeable_data['Encl osing_Excavation_Nothing'].value_counts()
excavati on_nothing_counts
```

```
Out[ ]: No      3953
Yes     194
Name: Encl osing_Excavation_Nothing, dtype: int64
```

```
In [ ]: print(f'{round(surface_unfilled_counts[1]/len(enclosi ng_encodeable_data)*100, 2)}%')
4.22%
```

```
In [ ]: excavation_nothing_data_yes = \
plot_over_grey(location_encl osing_encodeable_data, \
    'Encl osing_Excavation_Nothing', 'Yes', '', \
    False, False, False, False)
```

Enclosing Excavation Nothing



Middleton, M. 2024, Hillforts Primer

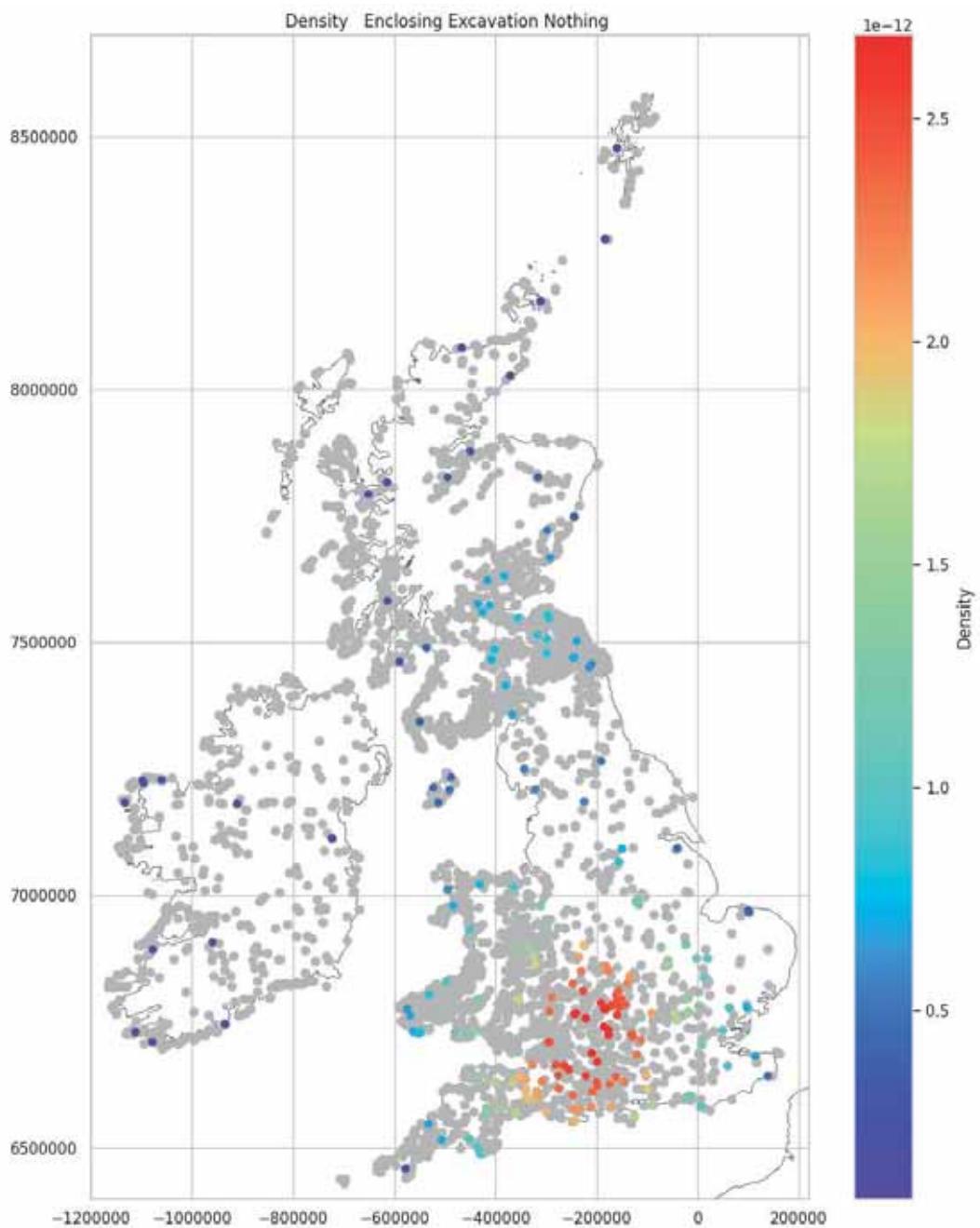
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

4. 68%

Enclosing Excavation Nothing Density Mapped

This cluster is biased and likely reflects the focus of excavation rather than anything more meaningful.

```
In [ ]: plot_density_over_grey(excavation_nothing_data_yes, \
    'Enclosing_Excavation_Nothing', '')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Excavation Bank Mapped

351 (8.4%) of hillforts have a bank exposed during excavation.

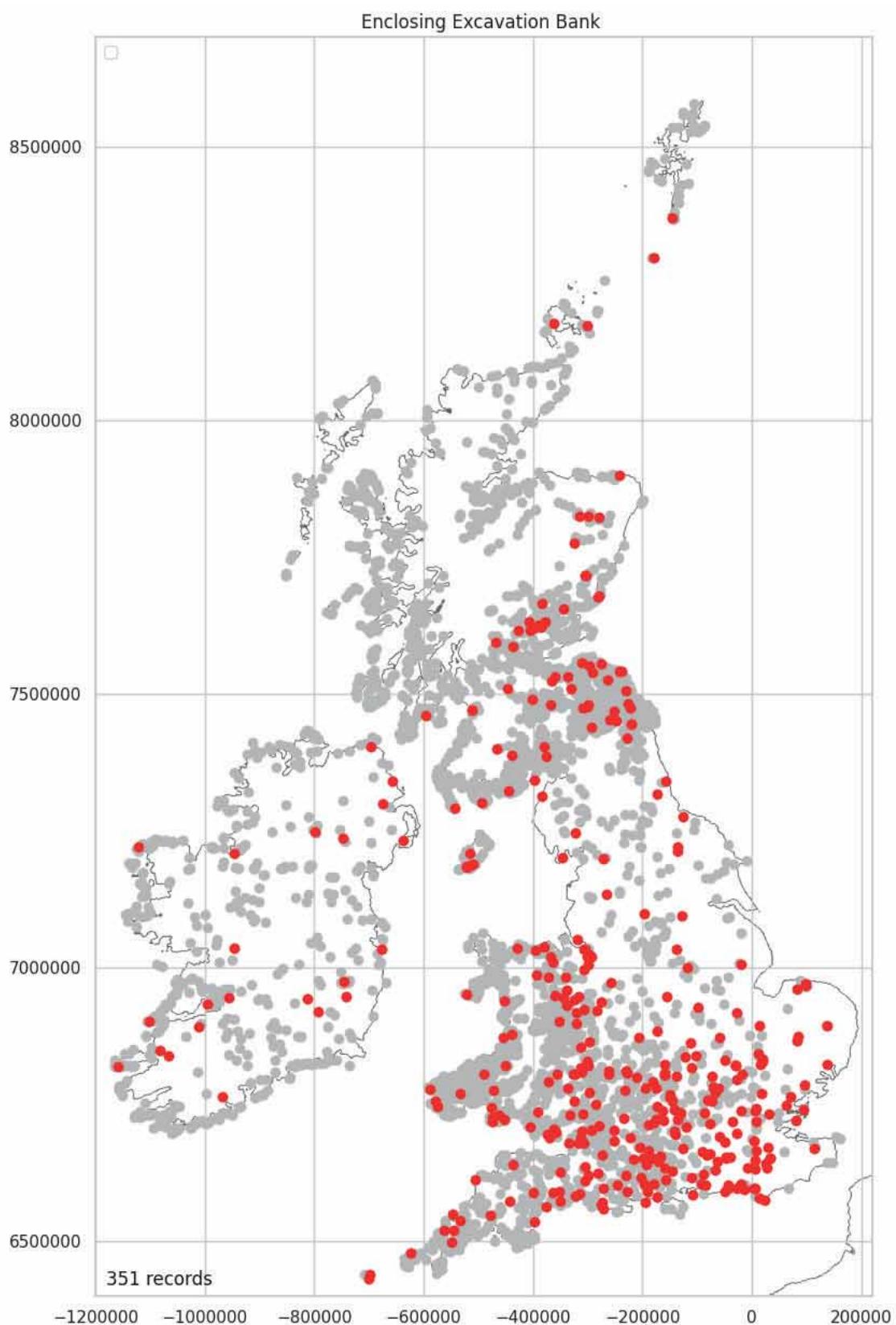
```
In [ ]: excavation_bank_counts = \
enclosi ng_encodeabl e_data['Encl osing_Exca vation_Bank'].value_counts()
excavation_bank_counts
```

```
Out[ ]: No      3796
Yes     351
Name: Encl osing_Exca vation_Bank, dtype: int64
```

```
In [ ]: print(f'{round(excavation_bank_counts[1]/len(enclosi ng_encodeabl e_data)*100, 2)}%')
```

8.46%

```
In [ ]: excavation_bank_data_yes = \
plot_over_grey(locate n_encl osing_encodeabl e_data, 'Encl osing_Exca vation_Bank', \
'Yes', '', False, False, False)
```



Middleton, M. 2024, Hillforts Primer

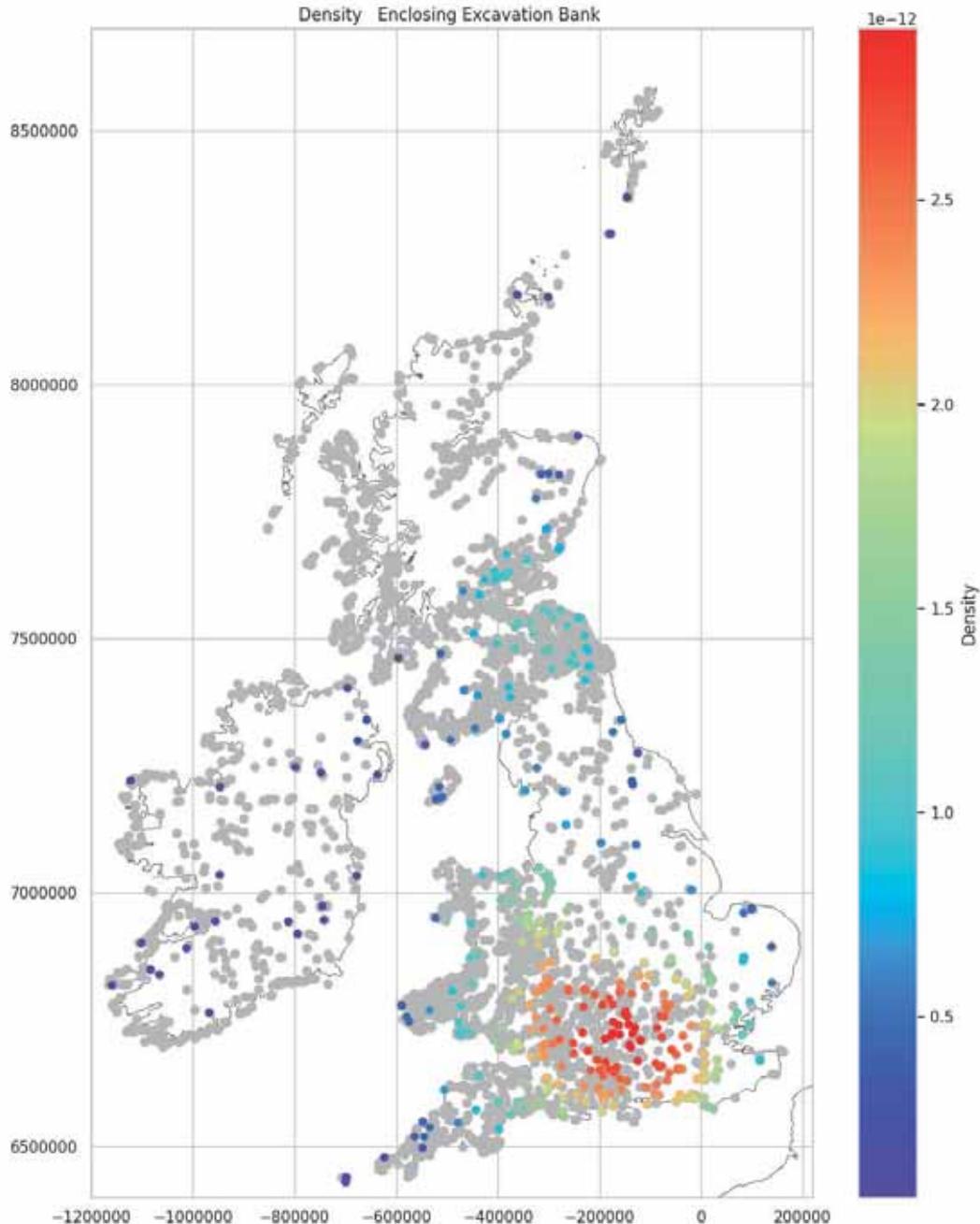
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

8. 46%

Enclosing Excavation Bank Density Mapped

The main cluster for this is in the South but, compared with the cluster seen in, Part 1: Southern Data Density Mapped (Transformed), the focus is considerably further east. Compared to the clusters seen in [Enclosing Surface Bank Density Mapped](#), where the main focus of banked enclosing circuits was in Wales and the Northeast, this distribution is misleading. The distribution is likely to reflect an excavation bias rather than being meaningful.

```
In [ ]: plot_densitiy_over_grey(excavation_bank_data_yes, 'Encl osing_Excavati on_Bank')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Excavation Wall Mapped

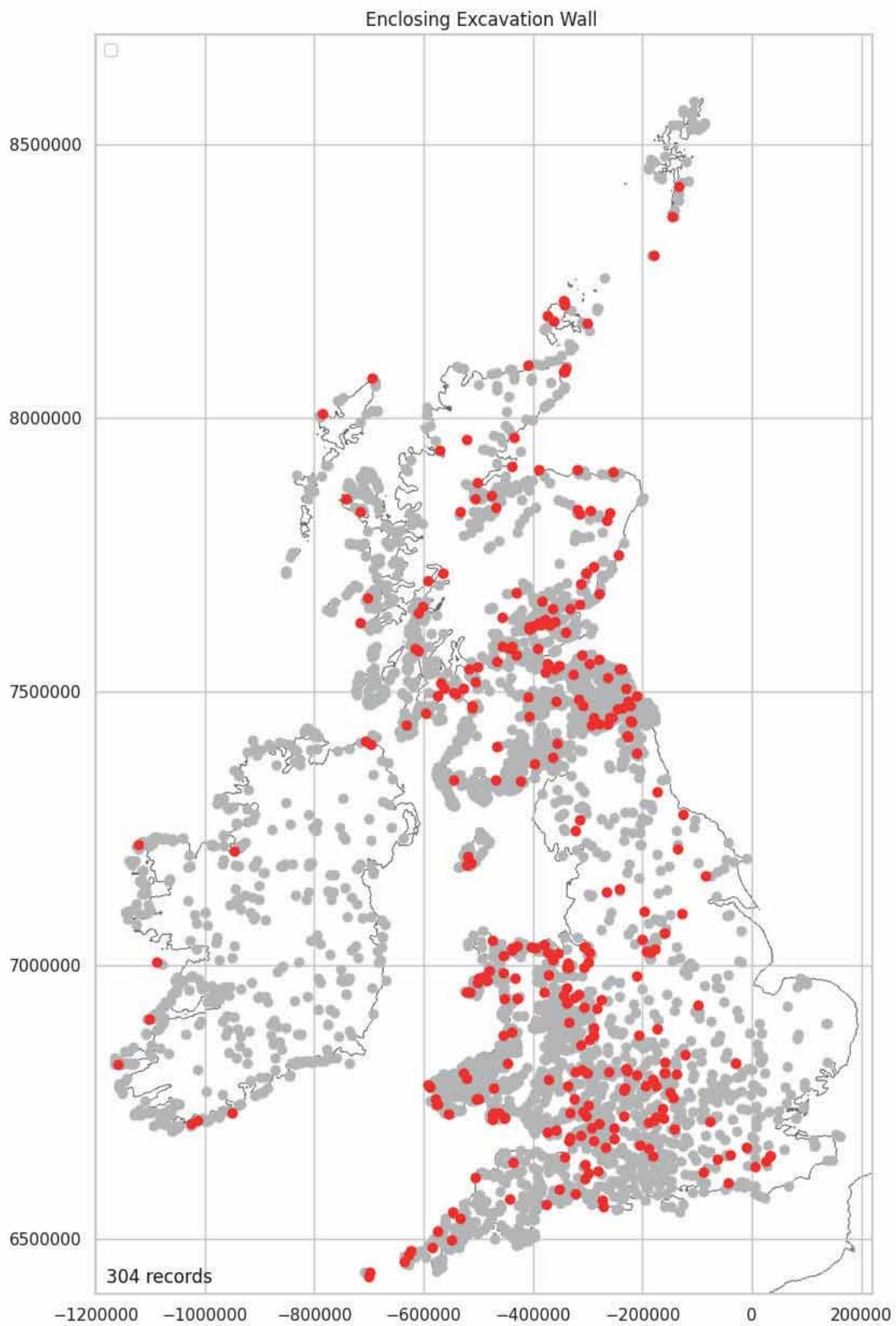
304 (7.33%) of hillforts excavated have revealed an enclosing wall.

```
In [ ]: excavation_wall_counts = \
encl osing_encodeable_data['Encl osing_Excavati on_Wall'].value_counts()
excavation_wall_counts
```

```
Out[ ]: No      3843
Yes     304
Name: Encl osing_Excavati on_Wall, dtype: int64
```

```
In [ ]: print(f'{round(excavation_wall_counts[1]/len(encl osing_encodeable_data)*100, 2)}%')
7.33%
```

```
In [ ]: excavation_wall_data_yes = \
plot_over_grey(location_encl osing_encodeable_data, 'Encl osing_Excavati on_Wall', \
'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

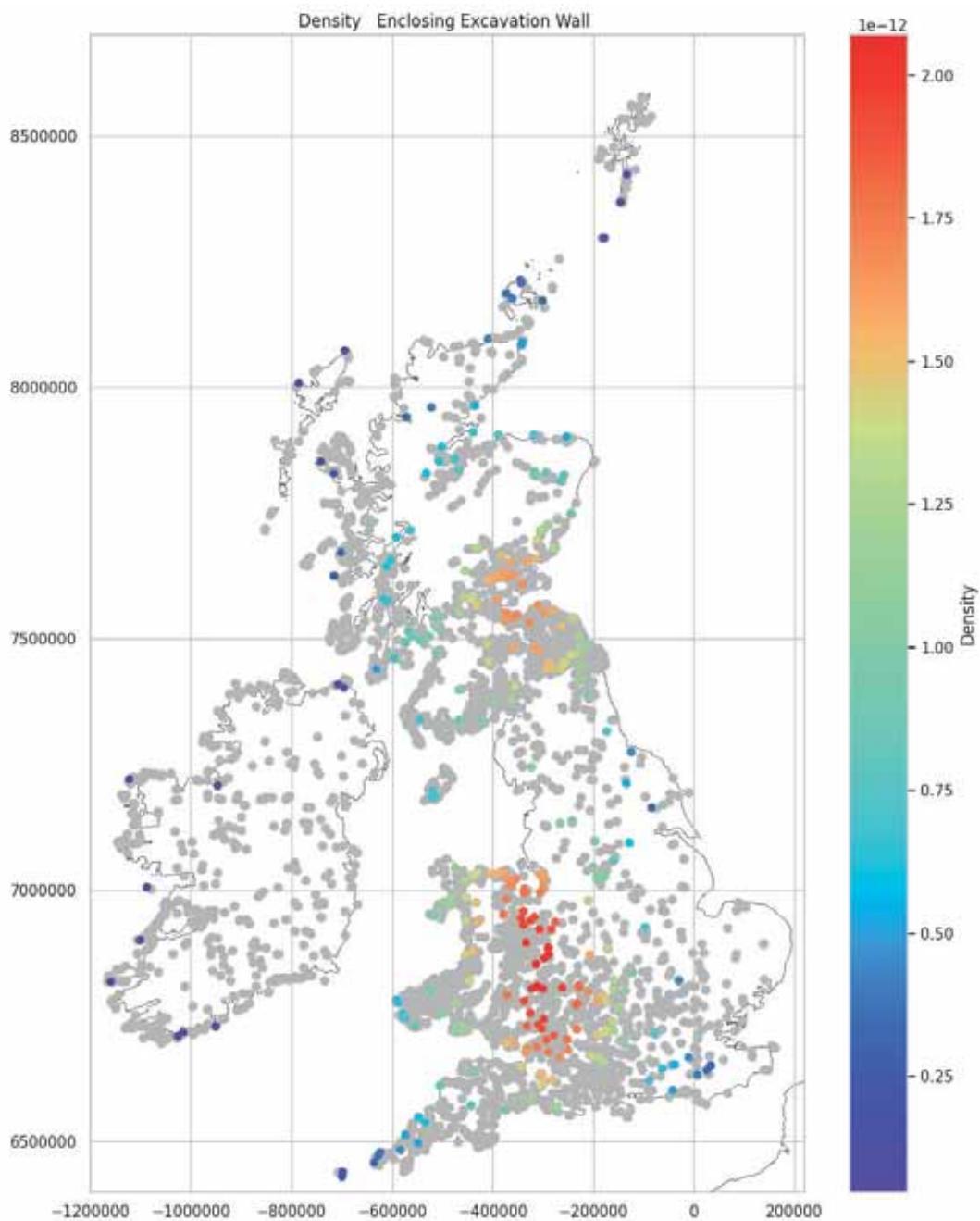
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

7.33%

Enclosing Excavation Wall Density

The main clusters of hillforts with walls seen in [Enclosing Surface Wall Density Mapped](#) was focussed in the Northwest and in west Wales. The main cluster here is to the east of the Cambrian Mountains and a smaller, secondary cluster can be seen in the Northeast. As with previous classes in the Enclosing Excavation section, this distribution suffers from survey bias.

```
In [ ]: plot_density_over_grey(excavation_wall_data_yes, \
    'Enclosing_Excavation_Wall'))
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Excavation Murus Gallicus Mapped

Just two hillforts have revealed Murus Gallicus recorded in excavation.

```
In [ ]: excavation_murus_counts = \
enclosi ng_encodeable_data['Encl osing_Excavati on_Murus'].value_counts()
excavation_murus_counts
```

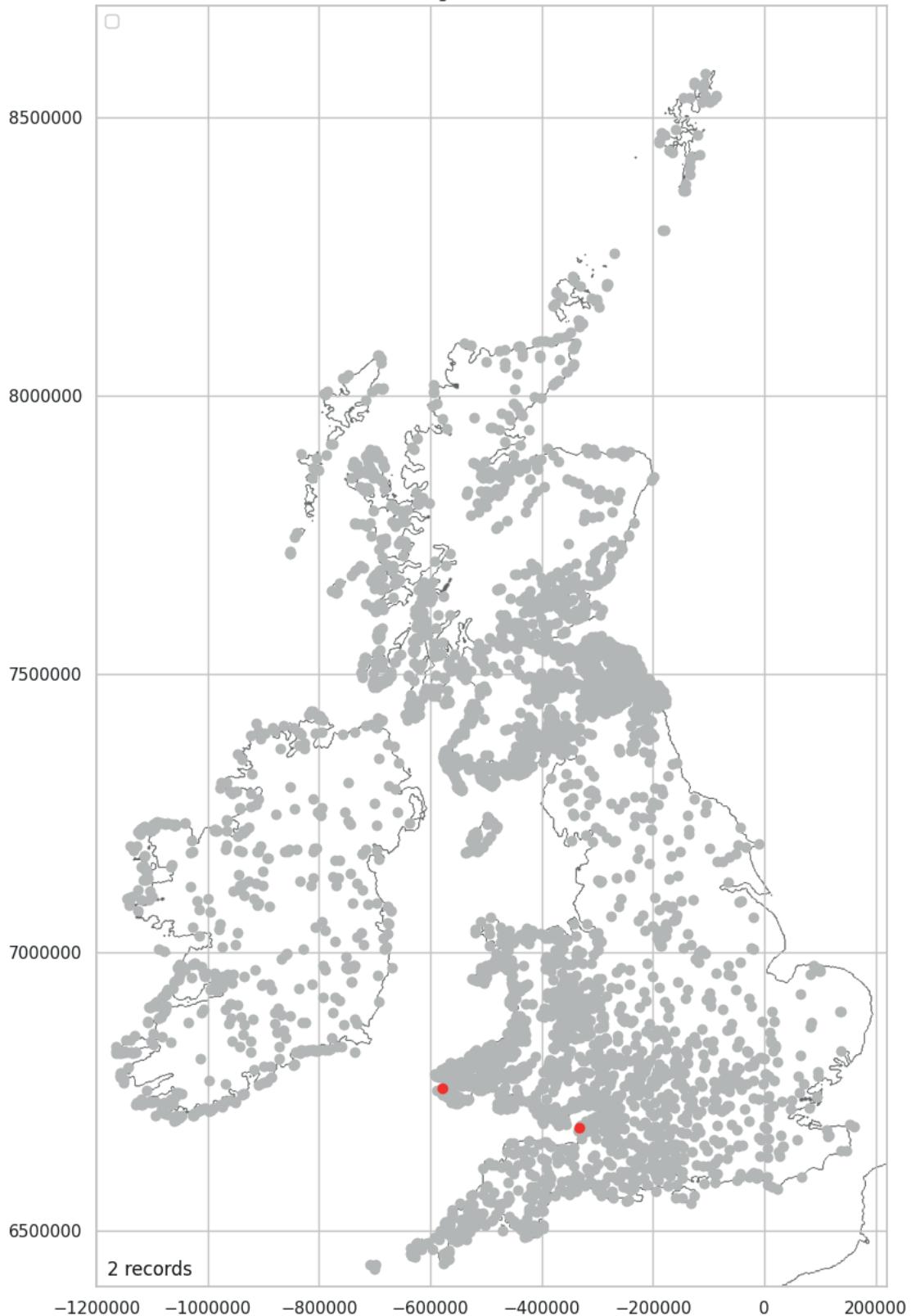
```
Out[ ]: No      4145
Yes       2
Name: Encl osing_Excavati on_Murus, dtype: int64
```

```
In [ ]: print(f'{round(excavation_murus_counts[1]/len(enclosi ng_encodeable_data)*100, 2)}%')
```

0.05%

```
In [ ]: excavation_murus_data_yes = \
plot_over_grey(locate_encl osing_encodeable_data, \
'Encl osing_Excavati on_Murus', 'Yes', '')
```

Enclosing Excavation Murus



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.05%

Enclosing Excavation Timber Framed Mapped

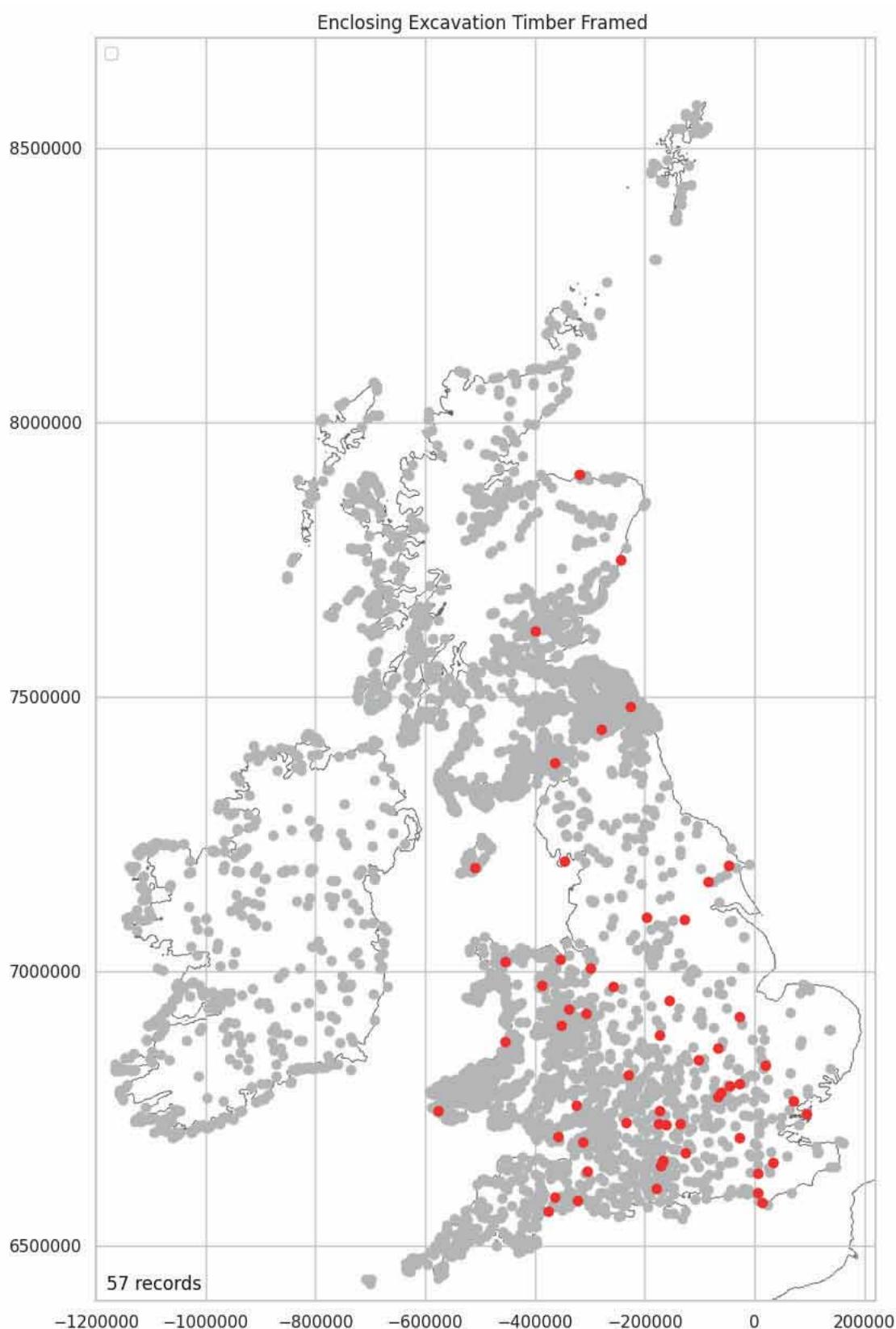
57 (1.37%) of hillforts have had a Timber Frame revealed during excavation.

```
In [ ]: excavation_tf_counts = \
enclosi ng_encodeabl e_data['Encl osing_Excavati on_Ti mber_Framed'].value_counts()
excavati on_tf_counts
```

```
Out[ ]: No      4090  
         Yes     57  
         Name: Enclosing_Excavation_Timber_Framed, dtype: int64
```

```
In [ ]: print(f' {round(excavation_tf_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')  
1.37%
```

```
In [ ]: excavation_tf_data_yes = plot_over_grey(location_enclosing_encodeable_data, \  
                                              'Enclosing_Excavation_Timber_Framed', \  
                                              'Yes', '')
```



Enclosing Excavation Timber Laced Mapped

46 (1.11%) of hillforts have had a Timber Lacing revealed during excavation.

```
In [ ]: excavation_tl_counts = \
encl osing_encodeable_data['Encl osing_Excavati on_Timber_Laced'].value_counts()
excavation_tl_counts
```

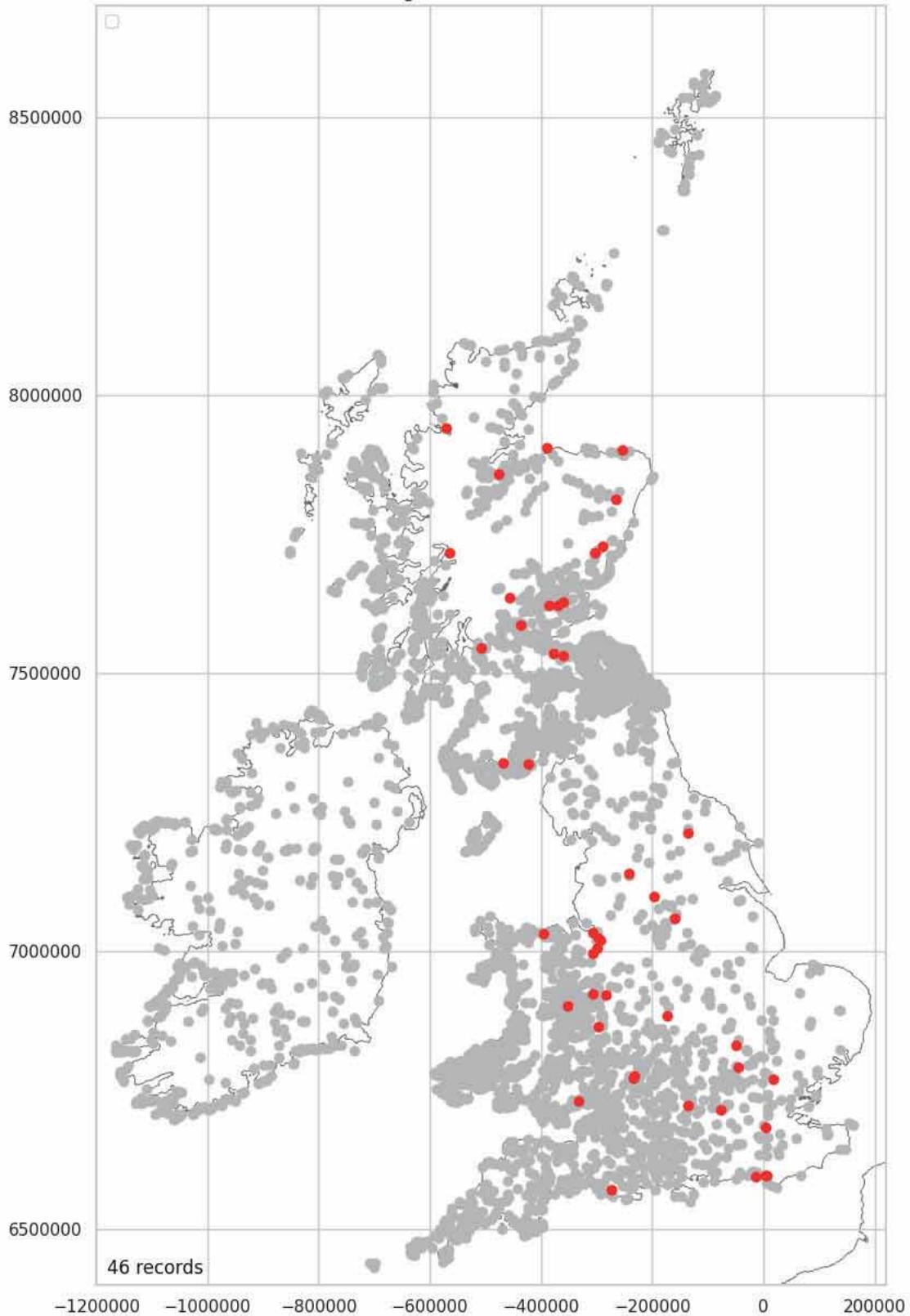
```
Out[ ]: No      4101
Yes      46
Name: Encl osing_Excavati on_Timber_Laced, dtype: int64
```

```
In [ ]: print(f'{round(excavation_tl_counts[1]/len(encl osing_encodeable_data)*100, 2)}%')
```

1.11%

```
In [ ]: excavation_tl_data_yes = \
plot_over_grey(location_encl osing_encodeable_data, \
'Encl osing_Excavati on_Timber_Laced', 'Yes', '')
```

Enclosing Excavation Timber Laced



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1.11%

Enclosing Excavation Vitrification Mapped

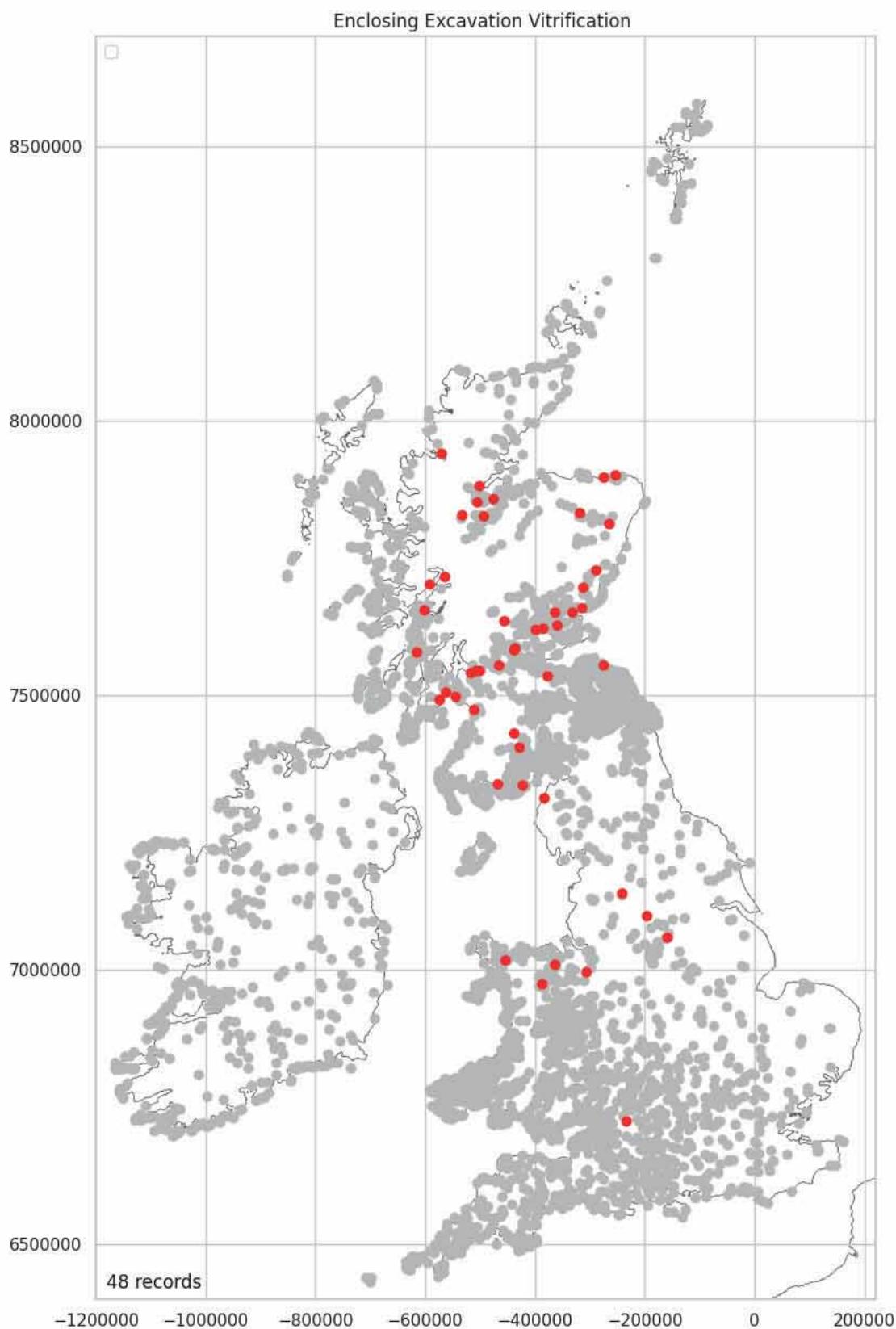
48 (1.16%) of hillforts have had Vitrification identified during excavation. See: [Enclosing Surface Vitrification Mapped](#)

```
In [ ]: excavation_vitrification_counts = \
enclosi ng_encodeabl e_data['Encl osing_Excavati on_Vi tri fi cati on'].value_counts()
excavati on_vitri fication_counts
```

```
Out[ ]: No      4099  
         Yes     48  
         Name: Enclosing_Excavation_Vitrification, dtype: int64
```

```
In [ ]: print(f' {round(excavation_vitrification_counts[1]/len(enclosing_encodeable_data)*100, 2)}% )  
1.16%
```

```
In [ ]: excavation_vitrification_data_yes = \  
plot_over_grey(location_enclosing_encodeable_data, \  
'Enclosing_Excavation_Vitrification', 'Yes', '')
```



Enclosing Excavation Burning Mapped

46 (1.11%) of hillforts have had burning, associated with the enclosing structure, identified during excavation.

```
In [ ]: excavation_burni ng_counts = \
encl osing_encodeabl e_data['Encl osing_Exca vati on_Burni ng'].val ue_counts()
excavati on_burni ng_counts
```

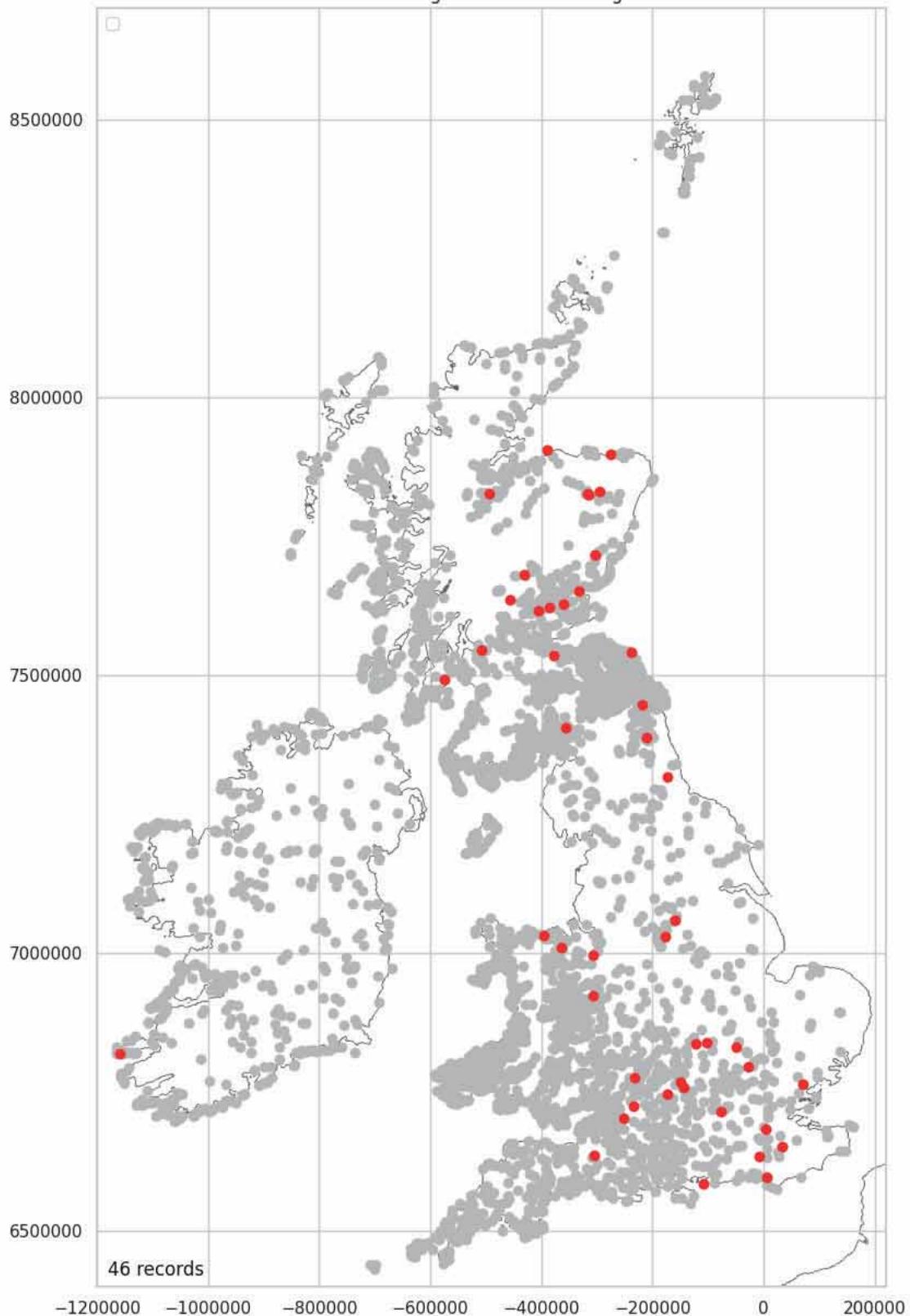
```
Out[ ]: No      4101
Yes      46
Name: Encl osing_Exca vati on_Burni ng, dtype: int64
```

```
In [ ]: print(f'{round(excavati on_burni ng_counts[1]/len(encl osing_encodeabl e_data)*100, 2)}%')
```

1.11%

```
In [ ]: excavation_burni ng_data_yes = \
plot_over_grey(location_encl osing_encodeabl e_data, \
'Encl osing_Exca vati on_Burni ng', 'Yes', '')
```

Enclosing Excavation Burning



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1.11%

Enclosing Excavation Palisade Mapped

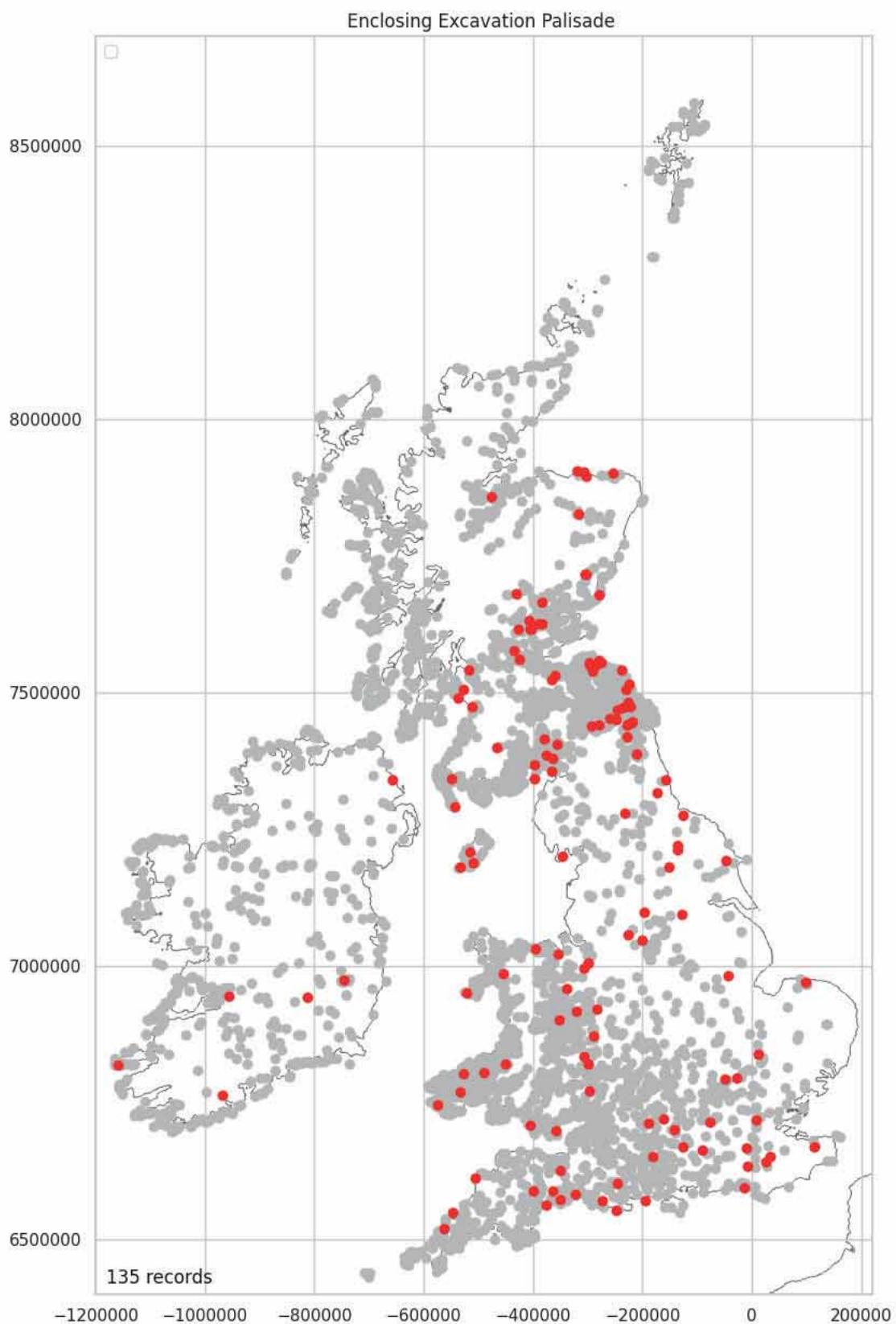
135 (3.26%) of hillforts have had a palisade revealed during excavation.

```
In [ ]: excavation_palisade_counts = \
enclosi ng_encodeable_data['Encl osing_Exca vati on_Pal isade'].value_counts()
excavati on_palisade_counts
```

```
Out[ ]: No      4012  
         Yes     135  
         Name: Enclosing_Excavation_Palisade, dtype: int64
```

```
In [ ]: print(f' {round(excavation_palisade_counts[1]/len(enclosing_encodeable_data)*100, 2)}% )  
3.26%
```

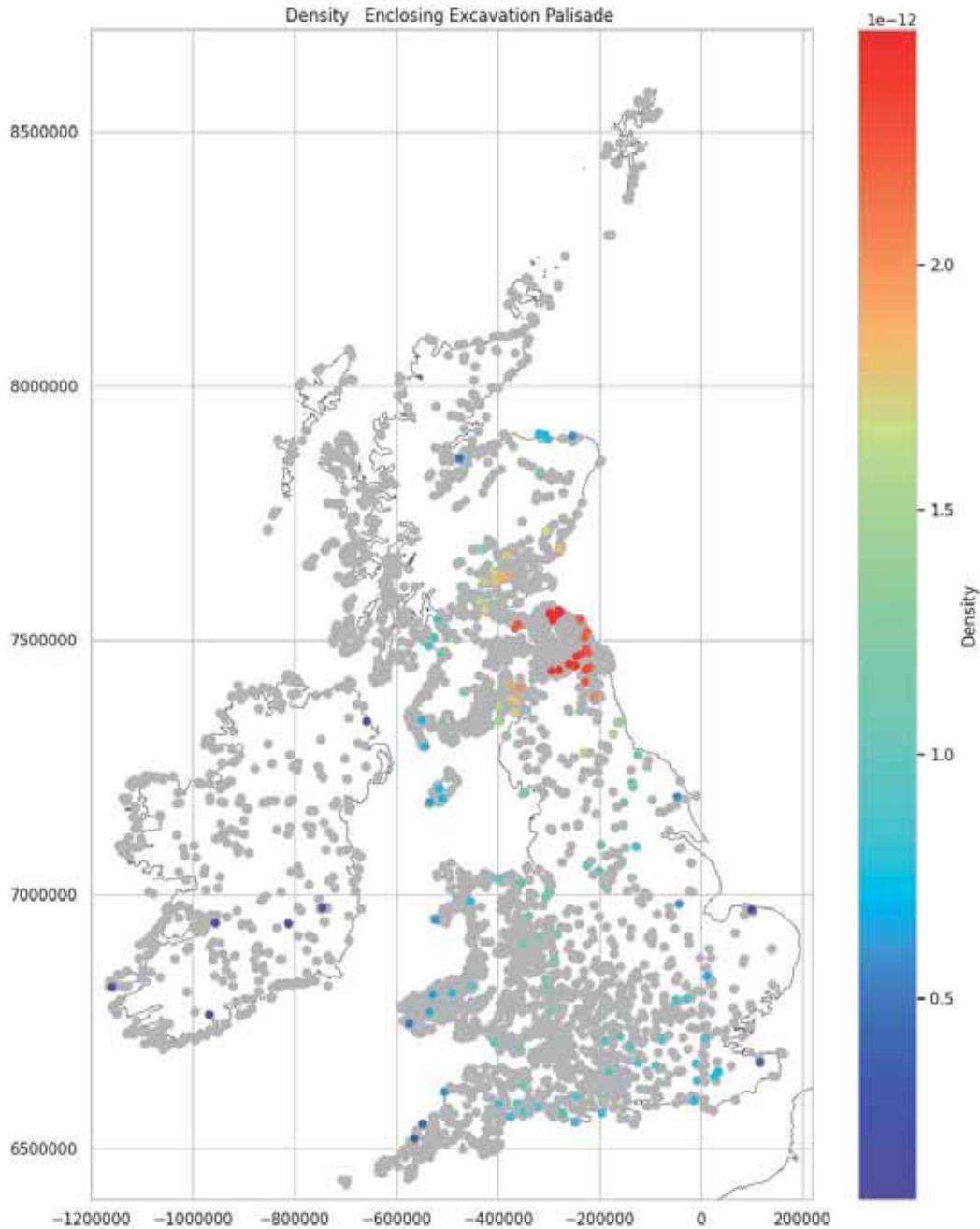
```
In [ ]: excavation_palisade_data_yes = \  
plot_over_grey(location_enclosing_encodeable_data, \  
'Enclosing_Excavation_Palisade', 'Yes', '')
```



Enclosing Excavation Palisade Density Mapped

The main cluster for excavated palisades is in the Northeast. This distribution mirrors that seen in [Enclosing Surface Palisade Density Mapped](#).

```
In [ ]: plot_density_over_grey(excavation_palisade_data_yes, \
    'Enclosing_Excavation_Palisade')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Excavation Counter Scarp Mapped

64 (1.54%) of hillforts have had a counterscarp exposed during excavation. See: [Enclosing Surface Counter Scarp Density Mapped](#).

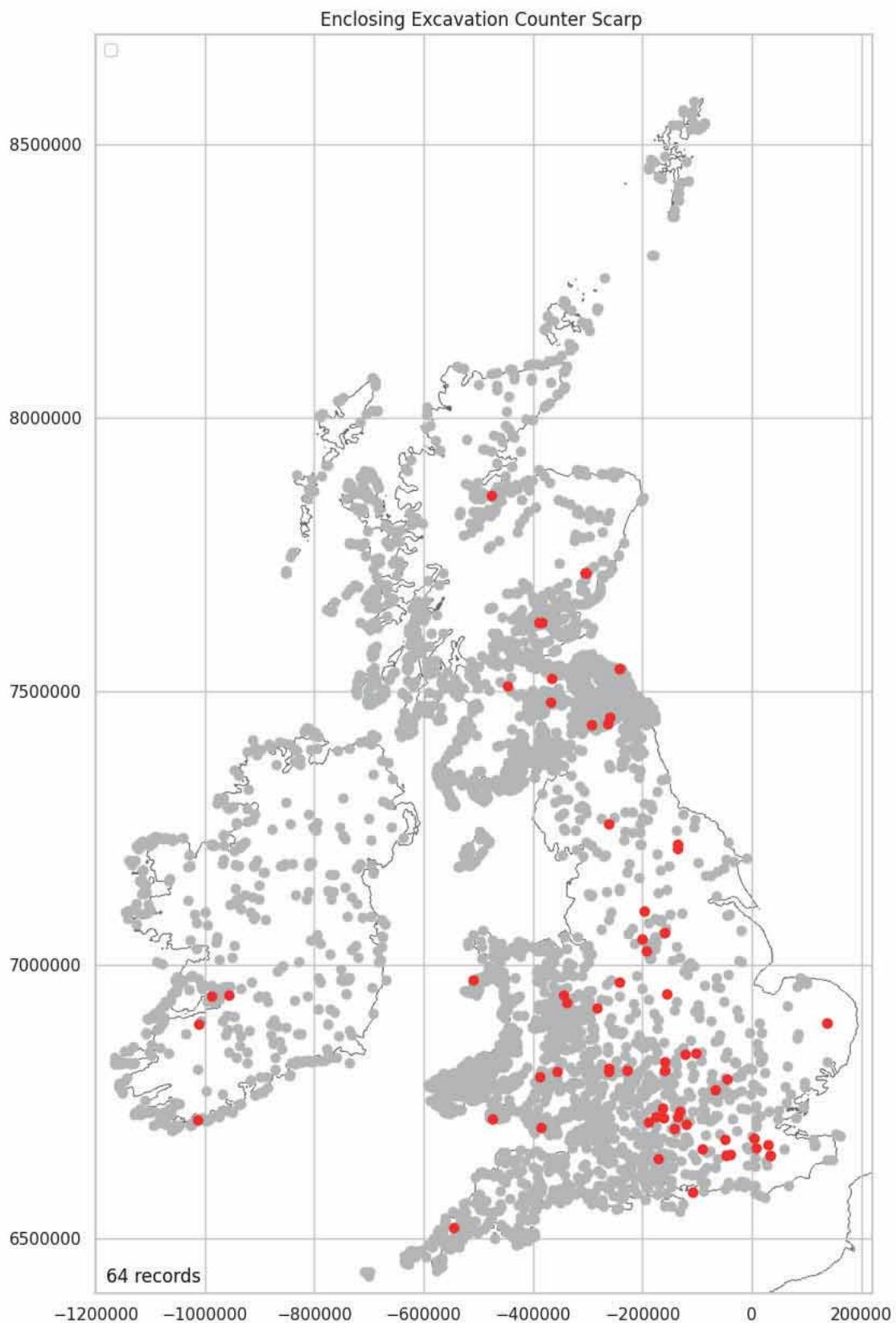
```
In [ ]: excavation_cs_counts = \
enclosiing_encodeable_data['Enclosing_Excavation_Counter_Scarp'].value_counts()
excavation_cs_counts
```

```
Out[ ]: No      4083
Yes      64
Name: Enclosing_Excavation_Counter_Scarp, dtype: int64
```

```
In [ ]: print(f'{round(excavation_cs_counts[1]/len(enclosiing_encodeable_data)*100,2)}%')
```

1. 54%

```
In [ ]: excavation_cs_data_yes = \
plot_over_grey(location_enclosing_encodeable_data, \
'Enclosing_Excavation_Counter_Scarp', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1. 54%

Enclosing Excavation Berm Mapped

24 (0.58%) of hillforts have had a berm revealed during excavation.

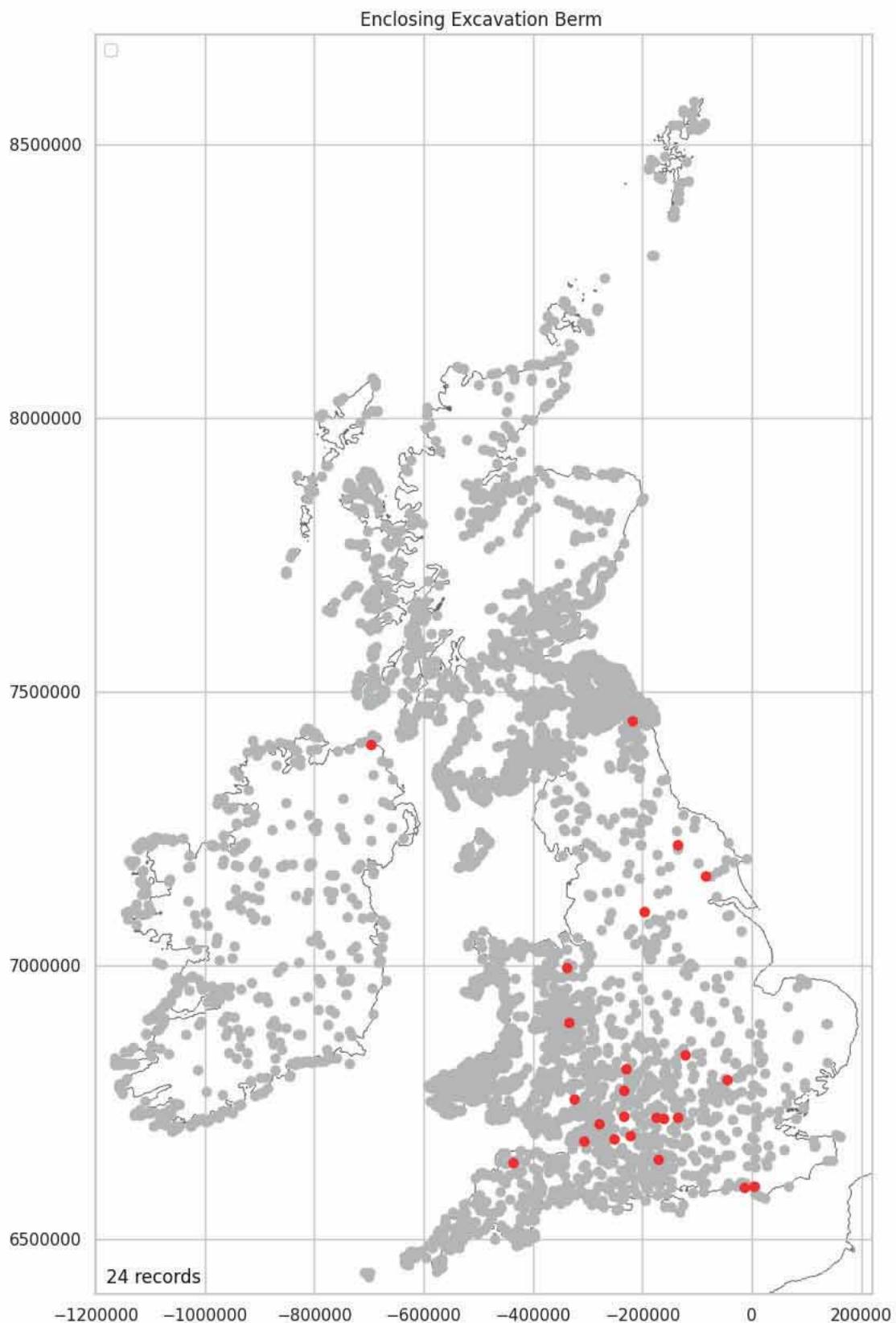
```
In [ ]: excavation_berm_counts = \
encl osing_encodeabl e_data['Encl osing_Excavati on_Berm'].val ue_counts()
excavati on_berm_counts
```

```
Out[ ]: No      4123
Yes      24
Name: Encl osing_Excavation_Berm, dtype: int64
```

```
In [ ]: print(f' {round(excavati on_berm_counts[1]/len(encl osing_encodeabl e_data)*100, 2)}%')
```

```
0. 58%
```

```
In [ ]: excavation_berm_data_yes = \
plot_over_grey(location_encl osing_encodeabl e_data, \
'Encl osing_Excavati on_Berm', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

0.58%

Enclosing Excavation Unfinished Mapped

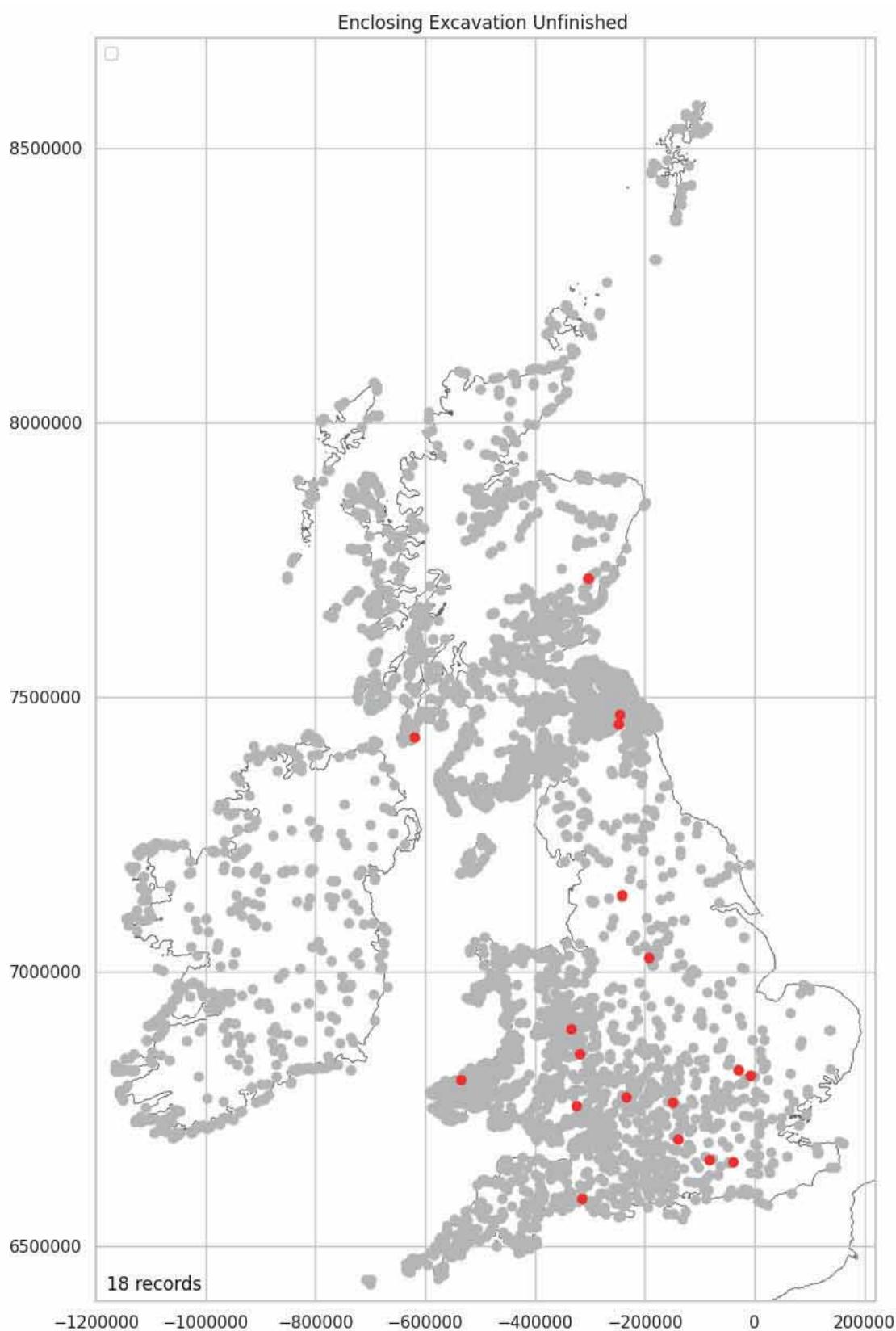
18 (0.43%) of hillforts have unfinished enclosing works revealed during excavation.

```
In [ ]: excavation_unfinished_counts = \
enclosi ng_encodeabl e_data['Enclosi ng_Exca vati on_Unfi ni shed'].value_counts()
excavati on_unfi ni shed_counts
```

```
Out[ ]: No      4129  
         Yes     18  
         Name: Enclosing_Excavation_Unfinished, dtype: int64
```

```
In [ ]: print(f' {round(excavation_unfinished_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')  
0.43%
```

```
In [ ]: excavation_unfinished_data_yes = \  
plot_over_grey(location_enclosing_encodeable_data, \  
'Enclosing_Excavation_Unfinished', 'Yes', '')
```



Enclosing Excavation Other Mapped

230 (5.55%) of hillforts have an enclosing circuit class, other than those listed above, recorded during excavation.

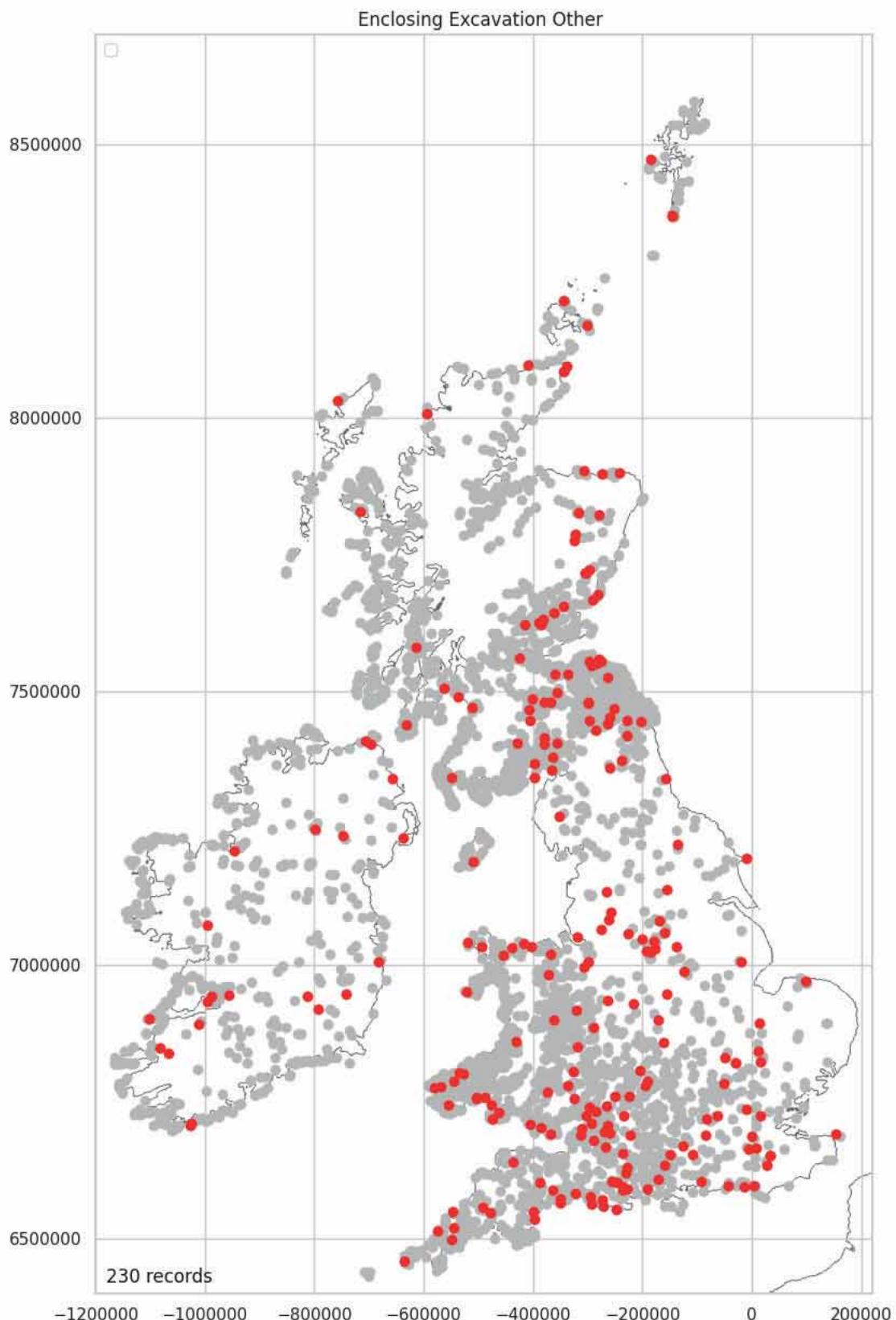
```
In [ ]: excavation_other_counts = \
enclosing_encodeable_data['Encl osing_Excavation_Other'].value_counts()
excavation_other_counts
```

```
Out[ ]: No      3917
Yes     230
Name: Encl osing_Excavation_Other, dtype: int64
```

```
In [ ]: print(f'{round(excavation_other_counts[1]/len(enclosing_encodeable_data)*100, 2)}%')
```

5.55%

```
In [ ]: excavation_other_data_yes = \
plot_over_grey(locations_encl osing_encodeable_data, 'Encl osing_Excavation_Other', \
'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

5.55%

Enclosing Excavation No Known Mapped

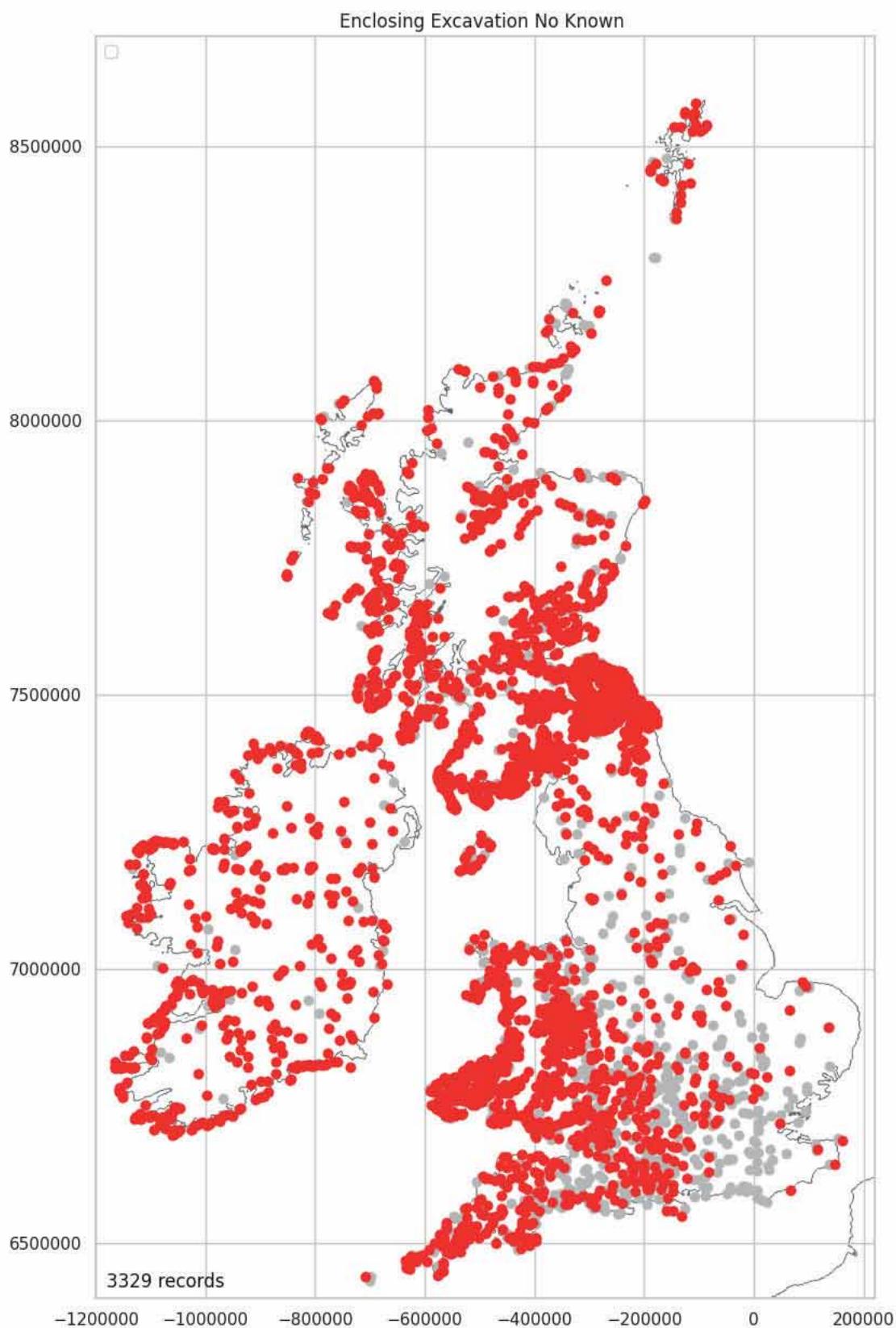
3329 (80.27%) of hillforts have had no known excavation on their enclosing circuit.

```
In [ ]: excavation_no_known_counts = \
enclosi ng_encodeabl e_data['Encl osing_Excavati on_No_Known'].value_counts()
excavati on_no_known_counts
```

```
Out[ ]: Yes    3329  
No     818  
Name: Enclosing_Excavation_No_Known, dtype: int64
```

```
In [ ]: print(f' {round(excavation_no_known_counts[0]/len(encl osing_encodeable_data)*100, 2)}% )')  
80.27%
```

```
In [ ]: excavation_no_known_data_yes = \  
plot_over_grey(location_encl osing_encodeable_data, \  
'Encl osing_Excavation_No_Known', 'Yes', '')
```



Enclosing Gang Working Mapped

44 (1.06%) of hillforts have signs of gang working recorded.

```
In [ ]: encl osi ng_gang_counts = \
encl osi ng_encodeabl e_data['Encl osi ng_Gang_Worki ng'].val ue_counts()
```

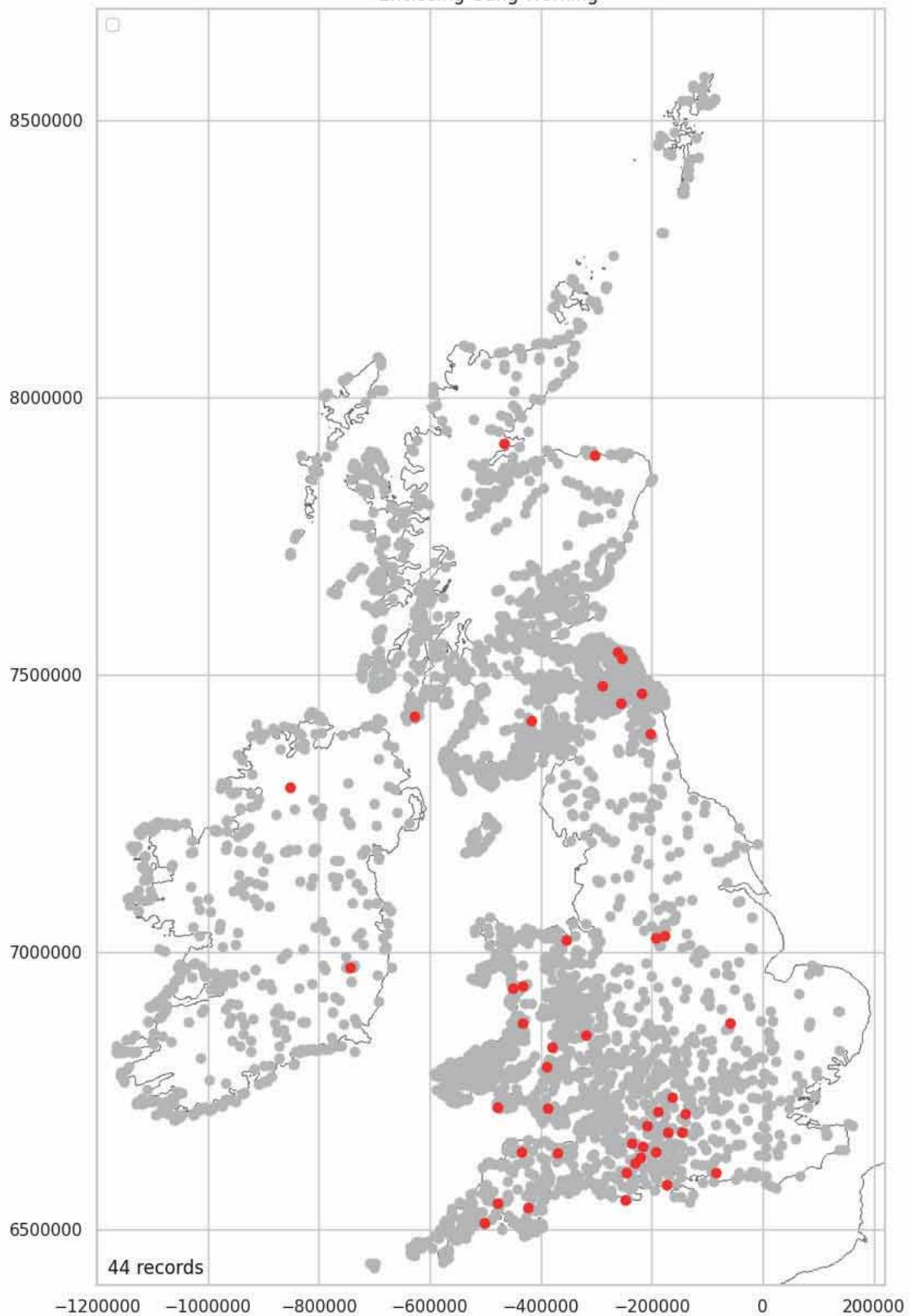
```
Out[ ]: No      4103
Yes     44
Name: Encl osi ng_Gang_Worki ng, dtype: int64
```

```
In [ ]: print(f' {round(encl osi ng_gang_counts[1]/len(encl osi ng_encodeabl e_data)*100, 2)}%')
```

1.06%

```
In [ ]: encl osi ng_gang_data_yes = \
plot_over_grey(locate_encl osi ng_encodeabl e_data, 'Encl osi ng_Gang_Worki ng', \
'Yes', '')
```

Enclosing Gang Working



Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

1.06%

Enclosing Ditches Mapped

2864 (69.06%) of hillforts are recorded as having ditches. This is nine less than the 2873 recorded in [Ditches Plotted](#). It is assumed that these nine do not form part of the enclosing circuit. With 91.89% of ditches, recorded in the ditches section above, also found here in the enclosing section, it can be said that ditches are predominantly an enclosing feature.

```
In [ ]: encl osing_ditches_counts = \
encl osing_encodeable_data['Encl osing_Di tches'].val ue_counts()
encl osing_ditches_counts
```

662

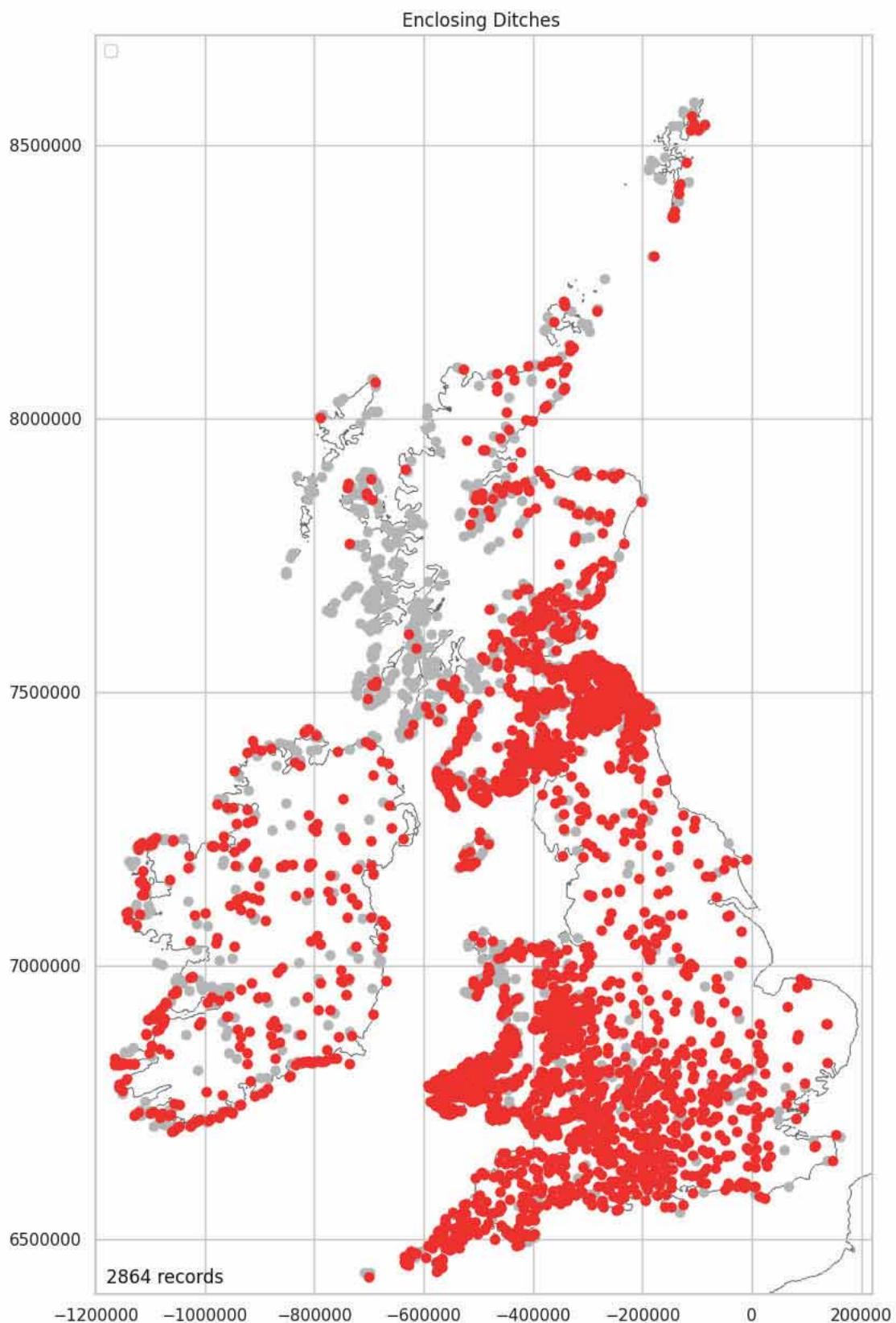
```
Out[ ]: Yes    2864
         No     1283
         Name: Enclosing_Di tches, dtype: int64
```

```
In [ ]: print(f' {round(enclosing_ditches_counts[0]/len(enclosing_encodeable_data)*100, 2)}% ' )
69.06%
```

```
In [ ]: enclosing_ditches_number_count = \
len(ditches_location_encode_data[ditches_location_encode_data\
['Enclosing_Di tches_Number'] > 0])
enclosing_ditches_number_count
```

```
Out[ ]: 2873
```

```
In [ ]: enclosing_ditches_data_yes = \
plot_over_grey(location_enclosing_encodeable_data, \
'Enclosing_Di tches', 'Yes', '')
```



Middleton, M. 2024, Hillforts Primer

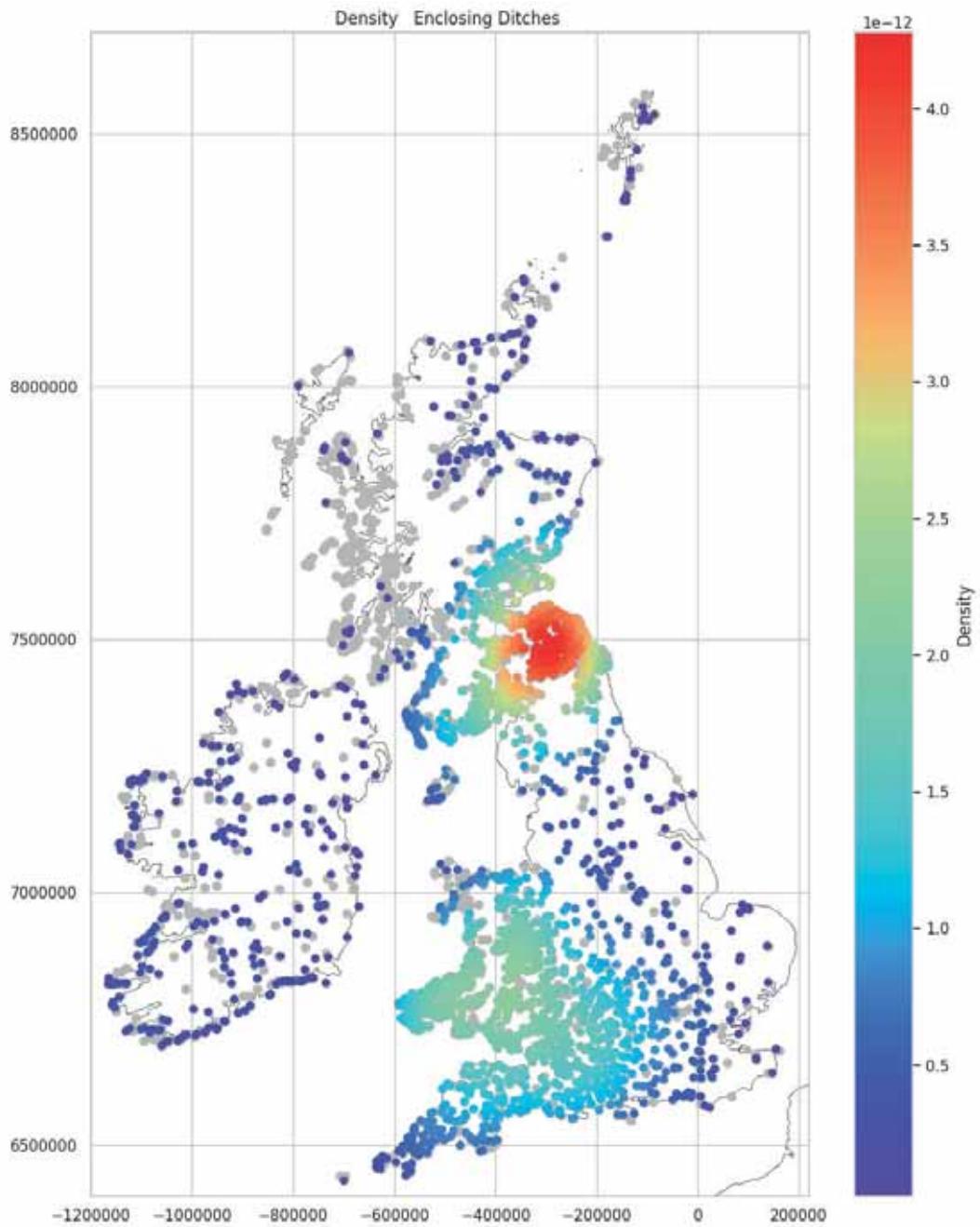
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

69.06%

Enclosing Ditches Density Mapped

As there is a 91.89% correlation between the ditches recorded above and the ditches in the enclosing section, it is unsurprising that the distribution of enclosure ditches matches that seen in [Ditches Clipped Mapped](#). The recording bias, specifically over the Northwest, and discussed in [Ditches Mapped \(Not Recorded\)](#), can be seen.

```
In [ ]: plot_density_over_grey(enclosing_ditches_data_yes, 'Enclosing_Ditches')
```



Middleton, M. 2024, Hillforts Primer

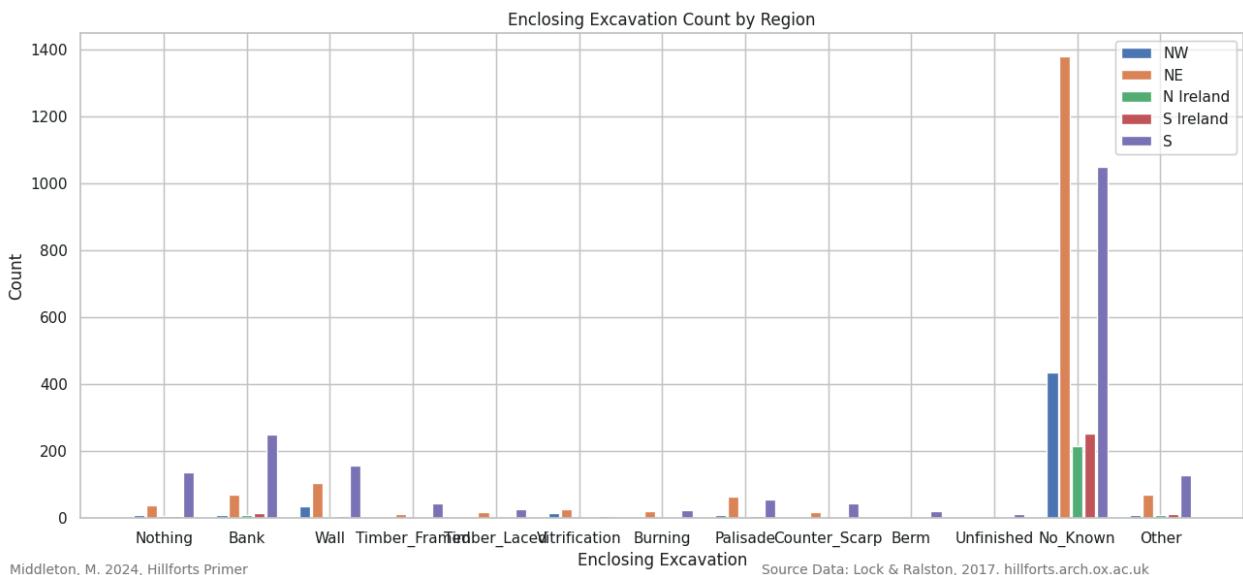
Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Enclosing Excavation by Region (Count)

No known excavation dominates this plot and will be removed in the following figure to improve the figure's legibility.

```
In [ ]: plot_regions(location_enclisings_encodeable_data_nw,
                   location_enclisings_encodeable_data_ne,
                   location_enclisings_encodeable_data_ireland_n,
                   location_enclisings_encodeable_data_ireland_s,
                   location_enclisings_encodeable_data_south,
                   ['Enclisings_Excavation_Nothing',
                    'Enclisings_Excavation_Bank',
                    'Enclisings_Excavation_Wall',
                    #'Enclisings_Excavation_Murus',
                    'Enclisings_Excavation_Timber_Framed',
                    'Enclisings_Excavation_Timber_Laced',
                    'Enclisings_Excavation_Vitriification',
                    'Enclisings_Excavation_Burning',
                    'Enclisings_Excavation_Palisade',
                    'Enclisings_Excavation_Counter_Scarp',
                    'Enclisings_Excavation_Berm',
                    'Enclisings_Excavation_Unfilled',
                    'Enclisings_Excavation_No_Known',
                    'Enclisings_Excavation_Other'],
```

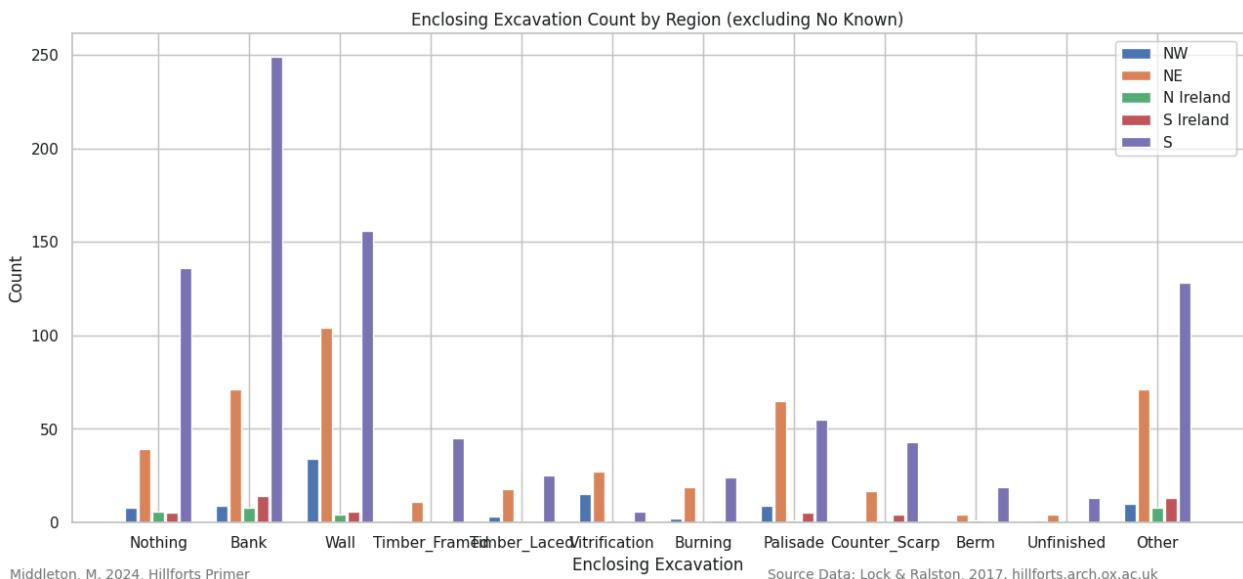
'Encl osing Excavation',
 'Encl osing_Excavation Count by Region', 2, 'Yes')



Enclosing Excavation by Region (Count) (Excluding No Known)

As was seen in [Enclosing Surface by Region \(Count\)](#), raw counts are difficult to read. See the figures plotted by proportion below.

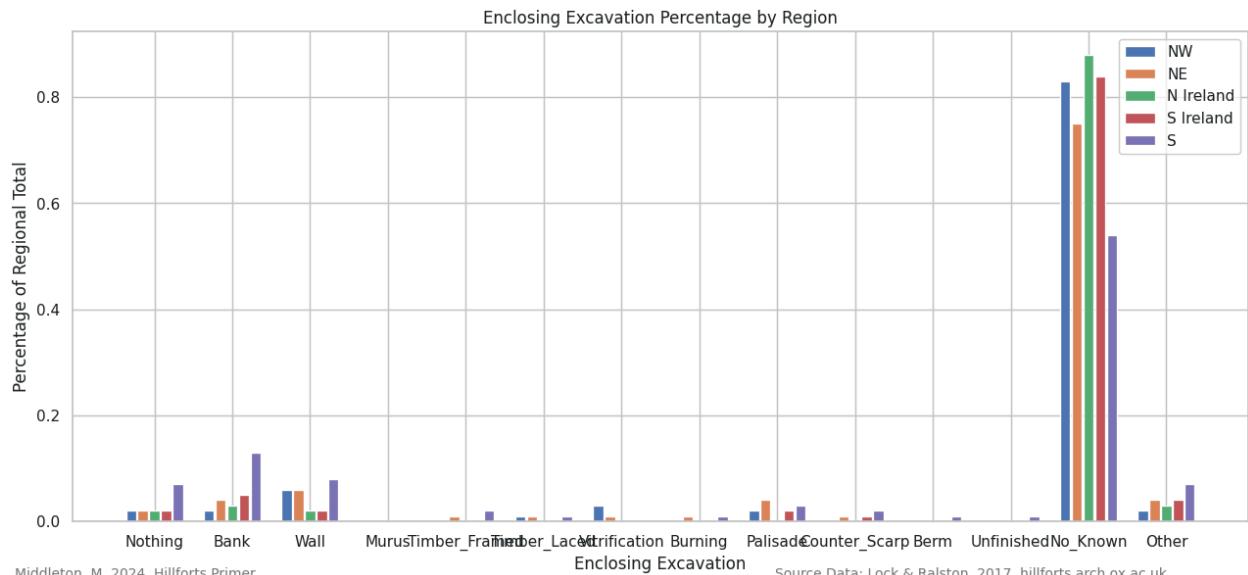
```
In [ ]: plot_regions(location_encl_osi_ng_encodeable_data_nw,
location_encl_osi_ng_encodeable_data_ne,
location_encl_osi_ng_encodeable_data_ireland_n,
location_encl_osi_ng_encodeable_data_ireland_s,
location_encl_osi_ng_encodeable_data_south,
['Encl osing_Excavation_Nothing',
'Encl osing_Excavation_Bank',
'Encl osing_Excavation_Wall',
#'Encl osing_Excavation_Murus',
'Encl osing_Excavation_Timber_Framed',
'Encl osing_Excavation_Timber_Laced',
'Encl osing_Excavation_Vitrifaction',
'Encl osing_Excavation_Burni ng',
'Encl osing_Excavation_Palisade',
'Encl osing_Excavation_Counter_Scarp',
'Encl osing_Excavation_Berm',
'Encl osing_Excavation_Unfi ni shed',
#'Encl osing_Excavati on_No_Known',
'Encl osing_Excavation_Other'],
'Encl osing_Excavation',
'Encl osing_Excavation Count by Region (excluding No Known)', 2, 'Yes')
```



Enclosing Excavation by Region (Percentage)

This chart shows that most hillforts, in all regions, have not been excavated. The low bar for South, under 'No Known', shows that more hillforts have been excavated in the South than elsewhere. This translates to more features, by class, having been found in the South than across the other regions.

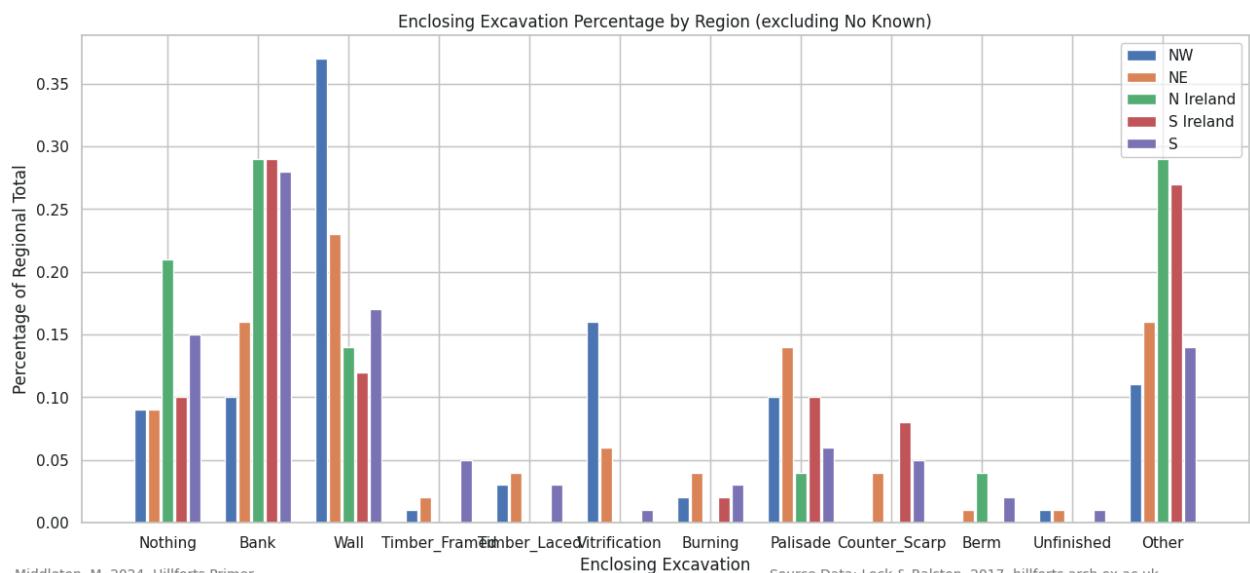
```
In [ ]: plot_regions(LOCATION_ENCODEABLE_DATA_NW,
                    LOCATION_ENCODEABLE_DATA_NE,
                    LOCATION_ENCODEABLE_DATA_RELAND_N,
                    LOCATION_ENCODEABLE_DATA_RELAND_S,
                    LOCATION_ENCODEABLE_DATA_SOUTH,
                    ['Encl osing_Excavation_Nothing',
                     'Encl osing_Excavation_Bank',
                     'Encl osing_Excavation_Wall',
                     'Encl osing_Excavation_Murus',
                     'Encl osing_Excavation_Timber_Framed',
                     'Encl osing_Excavation_Timber_Laced',
                     'Encl osing_Excavation_Vitrification',
                     'Encl osing_Excavation_Burning',
                     'Encl osing_Palisade',
                     'Encl osing_Counter_Scarp',
                     'Encl osing_Berm',
                     'Encl osing_Excavation_Unfinished',
                     'Encl osing_Excavation_No_Known',
                     'Encl osing_Excavation_Other'],
                     'Encl osing Excavation Percentage by Region', 2, 'Yes', True)
```



Enclosing Excavation by Region (Percentage) (Excluding No Known)

By excluding the 'No Known' excavation data, the remaining data can be plotted as a proportion of the total recorded classes by area. This reduces the dominance of the South data and enable the remaining plots to be comparable, proportionally, across the regions. This shows that in excavation, walls dominate the Northwest data while banks dominate in the South and across Ireland. In the Northeast, walls are proportionally the most common, but banks and palisades are also common. Of the remainder, vitrification is most common in the Northwest but is also found in the Northeast.

```
In [ ]: plot_regions(LOCATION_ENCODEABLE_DATA_NW,
                    LOCATION_ENCODEABLE_DATA_NE,
                    LOCATION_ENCODEABLE_DATA_RELAND_N,
                    LOCATION_ENCODEABLE_DATA_RELAND_S,
                    LOCATION_ENCODEABLE_DATA_SOUTH,
                    ['Encl osing_Excavation_Nothing',
                     'Encl osing_Excavation_Bank',
                     'Encl osing_Excavation_Wall',
                     #'Encl osing_Excavation_Murus',
                     'Encl osing_Excavation_Timber_Framed',
                     'Encl osing_Excavation_Timber_Laced',
                     'Encl osing_Excavation_Vitrification',
                     'Encl osing_Excavation_Burning',
                     'Encl osing_Palisade',
                     'Encl osing_Counter_Scarp',
                     'Encl osing_Berm',
                     'Encl osing_Excavation_Unfinished',
                     #'Encl osing_Excavation_No_Known',
                     'Encl osing_Excavation_Other'],
                     'Encl osing Excavation Percentage by Region (excluding No Known)', 2, 'Yes', True)
```



Review Enclosing Data Split

```
In [ ]: review_data_split(enclosi ng_data, enclosi ng_numeric_data, \
                           enclosi ng_text_data, enclosi ng_encodeable_data)
```

Data split good.

Enclosing Data Package

```
In [ ]: enclosi ng_data_list = \
          [enclosi ng_numeric_data, enclosi ng_text_data, enclosi ng_encodeable_data]
```

Enclosing Data Download Packages

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(enclosi ng_data_list, 'enclosi ng_package')
```

Annex Data

There are just two annex features.

```
In [ ]: annex_features = [
          'Annex',
          'Annex_Summary']

annex_data = hillforts_data[annex_features]
annex_data.head()
```

```
Out[ ]: Annex Annex_Summary
0 No NaN
1 No NaN
2 No NaN
3 No NaN
4 No NaN
```

Annex Numeric Data

There is no annex numeric data.

```
In [ ]: annex_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Annex       4147 non-null   object  
 1   Annex_Summary 533 non-null   object  
dtypes: object(2)
memory usage: 64.9+ KB
```

```
In [ ]: annex_numeric_data = pd.DataFrame()
```

Annex Text Data

There is a single annex text feature and it contains null values.

```
In [ ]: annex_text_data = pd.DataFrame(annex_data['Annex_Summary'].copy())
annex_text_data.head()
```

```
Out[ ]: Annex_Summary
```

	Annex_Summary
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN

Annex Text Data - Resolve Null Values

Test for 'NA'.

```
In [ ]: test_cat_list_for_NA(annex_text_data, ['Annex_Summary'])
```

```
Annex_Summary 0
```

Fill null values with 'NA'.

```
In [ ]: annex_text_data = update_cat_list_for_NA(annex_text_data, ['Annex_Summary'])
annex_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Annex_Summary 4147 non-null   object  
dtypes: object(1)
memory usage: 32.5+ KB
```

Annex Encodable Data

There is a single encodable annex feature. It does not contain null values.

```
In [ ]: annex_encodeable_data = pd.DataFrame(annex_data['Annex'].copy())
annex_encodeable_data.head()
```

```
Out[ ]: Annex
```

	Annex
0	No
1	No
2	No
3	No
4	No

```
In [ ]: location_annex_encodeable_data = \
pd.merge(location_numeric_data_short, annex_encodeable_data, \
         left_index=True, right_index=True)
```

```
In [ ]: location_annex_encodeable_data_ne = \
pd.merge(north_east.reset_index(), annex_encodeable_data, \
         left_on='uid', right_index=True)
location_annex_encodeable_data_ne = \
```

```

pd.merge(name_and_number, location_annex_encodeable_data_ne, \
         left_index=True, right_on='uid')

In [ ]: location_annex_encodeable_data_nw = \
pd.merge(north_west.reset_index(), annex_encodeable_data, \
         left_on='uid', right_index=True)
location_annex_encodeable_data_nw = \
pd.merge(name_and_number, location_annex_encodeable_data_nw, \
         left_index=True, right_on='uid')

In [ ]: location_annex_encodeable_data_i_rel_and_n = \
pd.merge(north_i_rel_and.reset_index(), annex_encodeable_data, \
         left_on='uid', right_index=True)
location_annex_encodeable_data_i_rel_and_n = \
pd.merge(name_and_number, location_annex_encodeable_data_i_rel_and_n, \
         left_index=True, right_on='uid')

In [ ]: location_annex_encodeable_data_i_rel_and_s = \
pd.merge(south_i_rel_and.reset_index(), annex_encodeable_data, \
         left_on='uid', right_index=True)
location_annex_encodeable_data_i_rel_and_s = \
pd.merge(name_and_number, location_annex_encodeable_data_i_rel_and_s, \
         left_index=True, right_on='uid')

In [ ]: location_annex_encodeable_data_south = \
pd.merge(south, annex_encodeable_data, left_on='uid', right_index=True)
location_annex_encodeable_data_south = \
pd.merge(name_and_number, location_annex_encodeable_data_south, \
         left_index=True, right_on='uid')

```

Annex Mapped

271 (6.53%) of hillforts have an annex recorded.

```

In [ ]: annex_counts = annex_encodeable_data[['Annex']].value_counts()
annex_counts

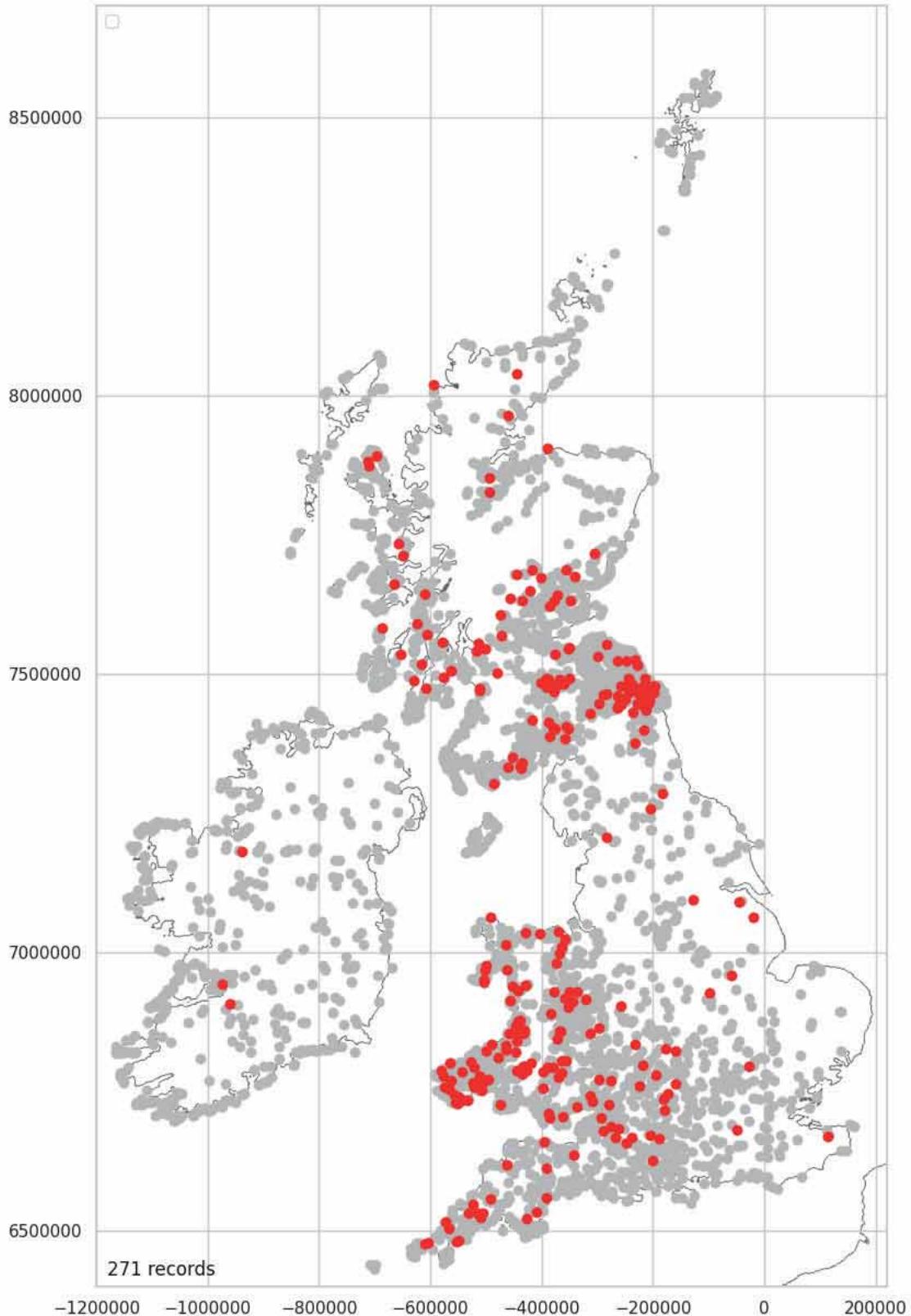
Out[ ]: Annex
No      3876
Yes     271
dtype: int64

In [ ]: print(f'{round(annex_counts[1]/len(annex_encodeable_data)*100, 2)}%')
6.53%

In [ ]: annex_data_yes = \
plot_over_grey(location_annex_encodeable_data, 'Annex', 'Yes', '')

```

Annex



Middleton, M. 2024, Hillforts Primer

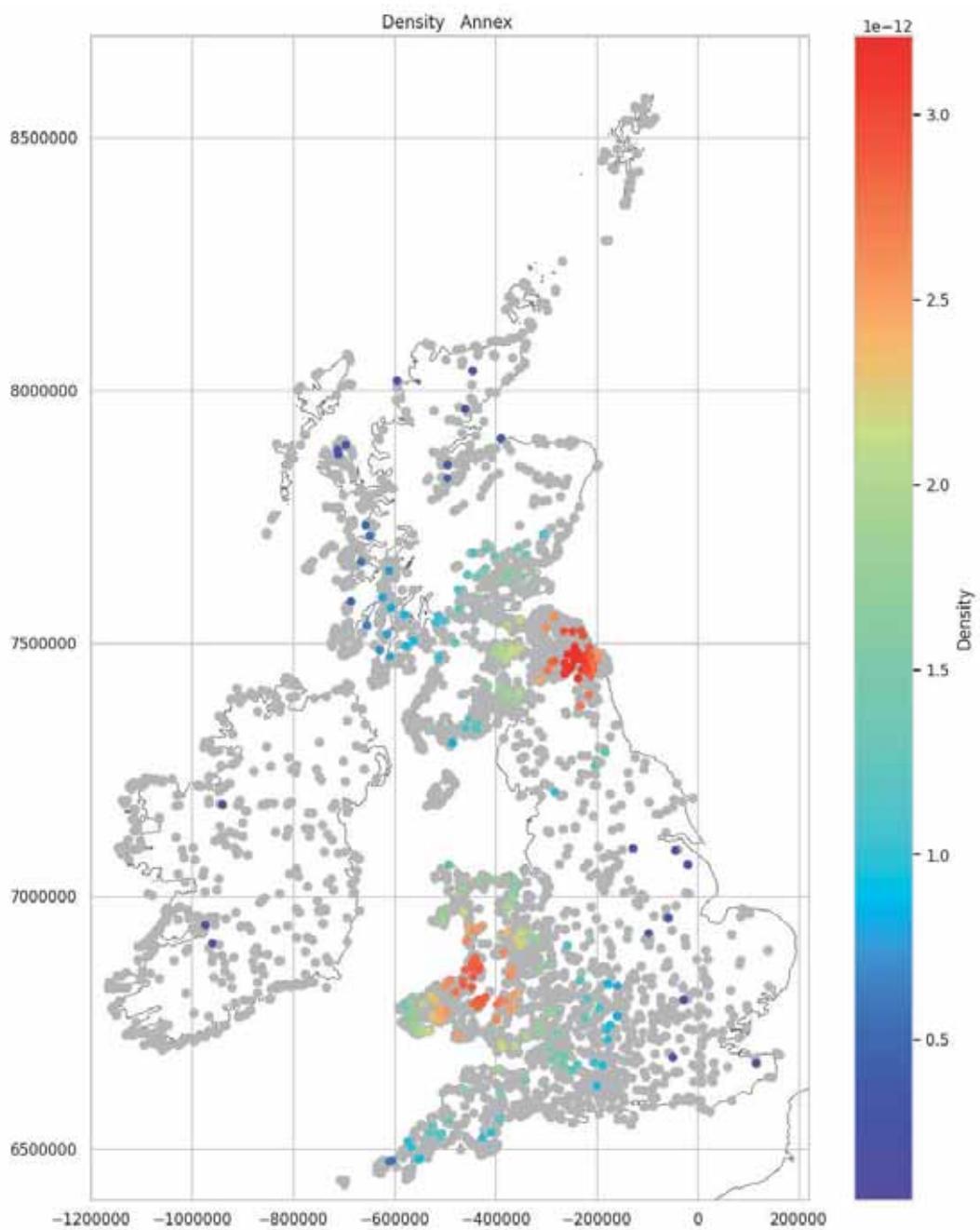
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

6. 53%

Annex Density Mapped

The two main annex clusters coincide with clusters seen in the general density distribution. See: Part 1: Density Data Transformed Mapped. There is a cluster in the Northeast and another over the southern end of the Cambrian mountains. There are very few annexes out with these areas, and this may indicate there is a recording bias.

```
In [ ]: plot_density_over_grey(annex_data_yes, 'Annex')
```



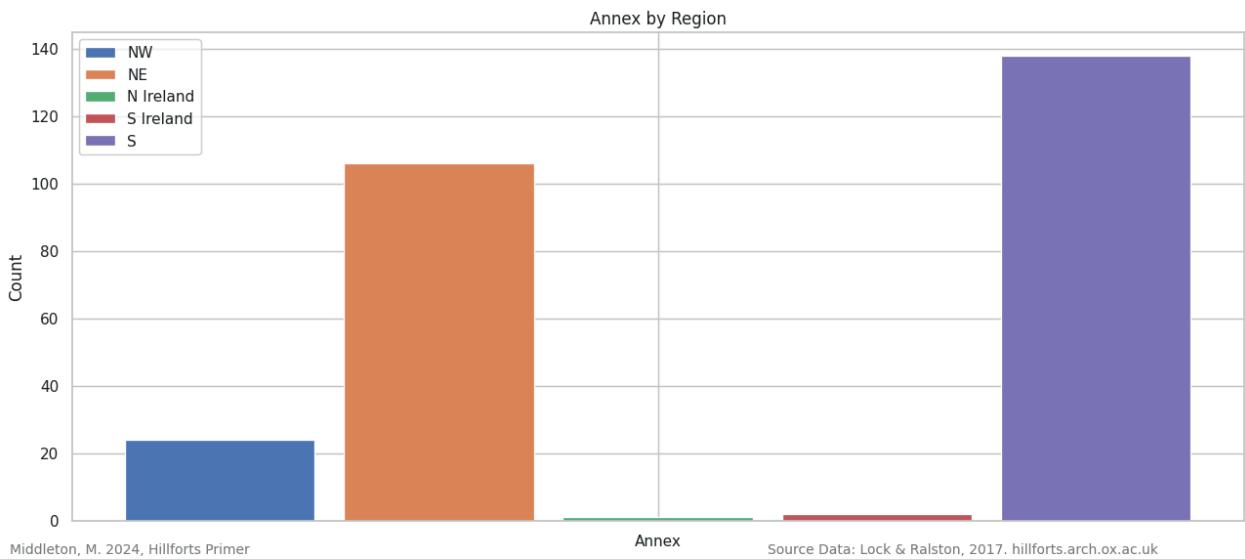
Middleton, M. 2024, Hillforts Primer

Source Data: Lock & Ralston, 2017, hillforts.arch.ox.ac.uk

Annex by Region (Count)

By count, most annexes are in the South and the Northeast. Annexes are rare in Ireland.

```
In [ ]: plot_regions(location_annex_encodeable_data_nw,
                     location_annex_encodeable_data_ne,
                     location_annex_encodeable_data_ireland_n,
                     location_annex_encodeable_data_ireland_s,
                     location_annex_encodeable_data_south,
                     ['Annex'],
                     '',
                     'Annex by Region', 0, 'Yes')
```



Review Annex Data Split

```
In [ ]: review_data_split(annex_data, annex_numeric_data, annex_text_data, annex_encodeable_data)
Data split good.
```

Annex Data Package

```
In [ ]: annex_data_list = [annex_numeric_data, annex_text_data, annex_encodeable_data]
```

Annex Data Download Packages

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(annex_data_list, 'annex_package')
```

Reference Data

Additional information relating to references is contained in a References Table. This can be downloaded from the Hillforts Atlas Rest Service API [here](#) or this project's data store [here](#). The References Table has not been analysed as part of the Hillforts Primer at this time.

There are eight reference data features. Three have no null values and two contain no data.

```
In [ ]: reference_features = [
'References',
'URL_Atlas',
'URL_Wiki',
'URL_NMR_Resource',
'NMR_URL',
'URL_HER_Resource',
'URL_HER',
'Record_URL']

reference_data = hillforts_data[reference_features]
reference_data.head()
```

Out[]:	References	URL_Atlas	URL_Wiki	URL_NMR_Resource	NMR_URL	URL_HER_Resource
0	Dorling, P. and Wigley, A. 2012. Assessment of...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31113987		NaN	NaN
1	Dorling, P. and Wigley, A. 2012. Assessment of...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31113996		NaN	NaN
2	Cooke, W.H. 1882. Collections towards the hist...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31114017		NaN	NaN
3	Dorling, P. and Wigley, A. 2012. Assessment of...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31114037		NaN	NaN
4	Bowden, M. 2000. British Camp or Herefordshire...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31114060		NaN	NaN

In []: reference_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   References      3643 non-null    object  
 1   URL_Atlas       4147 non-null    object  
 2   URL_Wiki         4147 non-null    object  
 3   URL_NMR_Resource 1695 non-null    object  
 4   NMR_URL          1690 non-null    object  
 5   URL_HER_Resource 0 non-null      float64 
 6   URL_HER          0 non-null      float64 
 7   Record_URL       4147 non-null    object  
dtypes: float64(2), object(6)
memory usage: 259.3+ KB
```

URL_HER_Resource and URL_HER contain no data. All other features in this class are object features. The names of these two features suggest they hold urls, thus object features. It looks like this data has been lost from the online Atlas because of a feature type mismatch. Certainly, urls cannot be stored as numeric float64. As they contain no data they will be dropped.

Reference Numeric Data

Because of the issue with the URL_HER_Resource and URL_HER feartures, mentioned above, reference numeric data contains no infromation.

In []: reference_numeric_data = pd.DataFrame()

Reference Text Data

Six of the reference features are text

```
In [ ]: reference_text_features = [
    'References',
    'URL_Atlas',
    'URL_Wiki',
    'URL_NMR_Resource',
    'NMR_URL',
    'Record_URL']

reference_text_data = pd.DataFrame(reference_data[reference_text_features].copy())
reference_text_data.head()
```

Out[]:	References	URL_Atlas	URL_Wiki	URL_NMR_Resource	NMR_URL
0	Dorling, P. and Wigley, A. 2012. Assessment of...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31113987	NaN	NaN http://hillforts.arch.ox.
1	Dorling, P. and Wigley, A. 2012. Assessment of...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31113996	NaN	NaN http://hillforts.arch.ox.
2	Cooke, W.H. 1882. Collections towards the hist...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31114017	NaN	NaN http://hillforts.arch.ox.
3	Dorling, P. and Wigley, A. 2012. Assessment of...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31114037	NaN	NaN http://hillforts.arch.ox.
4	Bowden, M. 2000. British Camp or Herefordshire...	https://hillforts.arch.ox.ac.uk/?query=Atlas_o...	http://www.wikidata.org/entity/Q31114060	NaN	NaN http://hillforts.arch.ox.

Reference Text Data - Resolve Null Values

Test for 'NA'.

```
In [ ]: test_cat_list_for_NA(reference_text_data, reference_text_features)

References 0
URL_Atlas 0
URL_Wiki 0
URL_NMR_Resource 0
NMR_URL 0
Record_URL 0
```

Fill null values with 'NA'.

```
In [ ]: reference_text_data = update_cat_list_for_NA(reference_text_data, reference_text_features)
reference_text_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   References      4147 non-null   object  
 1   URL_Atlas       4147 non-null   object  
 2   URL_Wiki         4147 non-null   object  
 3   URL_NMR_Resource 4147 non-null   object  
 4   NMR_URL          4147 non-null   object  
 5   Record_URL       4147 non-null   object  
dtypes: object(6)
memory usage: 194.5+ KB
```

Reference Encodable Data

There is no reference encodable data.

```
In [ ]: reference_encodeable_data = pd.DataFrame()
```

Review Reference Data Split

```
In [ ]: review_data_split(reference_data, reference_numeric_data, \
                         reference_text_data, reference_encodeable_data)
```

There are missing features: ['URL_HER', 'URL_HER_Resource']

Reference Data Package

```
In [ ]: reference_data_list = \
[reference_numeric_data, reference_text_data, reference_encodeable_data]
```

Reference Data Download Packages

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [ ]: download(reference_data_list, 'reference_package')
```

Save Figure List

```
In [ ]: if save_images:
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")
    fig_list.to_csv(path, index=False)
```

Acknowledgements

I would like to thank Emily Middleton for editing; Dr Dave Cowley for his encouragement, support and regular chats throughout this project; Strat Halliday for access to his expert knowledge and his thoughts on the data collection phase of the Hillforts Atlas, and to Professor Jeremy Huggett for his advice on how to summarise this work into abstracts for forthcoming publications.

Postscript

The work in the Hillforts Primer is the first phase in analysing the Hillforts Atlas data. This has been the data review. In reading these documents it is hoped that reader will have a solid grounding and understanding of the data's scope, limitations, areas of opportunity and where new research can complement what is already known. Throughout this document the data has been split into three groups; numeric, encodeable and text. The encoding has not been done in this phase as there is more to do. The data packages output, from this project, remain human readable. The next phase will look at correlations in the data as well as finally encoding and scaling the data. The next phase will render the data difficult to read for a human but it will make it much more likely to be useful for machine learning. Links to the next phase documents will be added once they become available. Thanks for reading.

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Appendix 1

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image

to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [1]: confirmed_only = False  
In [2]: download_data = False  
In [3]: save_images = False  
In [4]: show_topography = False
```

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Appendix 1](#).

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>
Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer
Licence: <https://creativecommons.org/licenses/by-sa/4.0/>
Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>
Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>
Hillforts: Britain, Ireland and the Nearer Continent (Sample):
<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Reload Data and Python Functions

This study is split over multiple documents. Each file needs to be configured and have the source data imported. As this section does not focus on the assessment of the data it is minimised to facilitate the documents readability.

Python Modules and Code Setup

The Python imports enable the Hillforts Atlas data to be analysed and mapped within this document. The Python code can be run on demand, (see: [User Settings](#)). This means that as new research becomes available, the source for this document can be updated to a revised copy of the Atlas data and the impact of that research can be reviewed using the same code and graphic output. The Hillforts Atlas is a baseline and this document is a tool that can be used to assess the impact new research is making in this area.

```
In [5]: import sys  
print(f'Python: {sys.version}')  
  
import sklearn  
print(f'Scikit-Learn: {sklearn.__version__}')
```

```

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

import os
import collections
import math
import random
import PIL
import urllib
random.seed(42)
# A random seed is used to ensure that the random numbers created
# are the same for each run of this document.

from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive

```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]

Scikit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.25.2
matplotlib: 3.7.1
seaborn: 0.13.1
scipy: 1.11.4

Ref: <https://www.python.org/>
Ref: <https://scikit-learn.org/stable/>
Ref: <https://pandas.pydata.org/docs/>
Ref: <https://numpy.org/doc/stable/>
Ref: <https://matplotlib.org/>
Ref: <https://seaborn.pydata.org/>
Ref: <https://docs.scipy.org/doc/scipy/index.html>
Ref: <https://pypi.org/project/python-slugify/>

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

```
In [6]: def show_records(plt, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plt.annotate(str(len(plot_data))+' records', xy=(-1180000, 6420000), \
                xycoords='data', ha='left', color=text_colour)
```

```
In [7]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
                      "hillforts-topo-northwest-minus.png",
                      "hillforts-topo-northeast.png",
                      "hillforts-topo-south.png",
                      "hillforts-topo-south-plus.png",
                      "hillforts-topo-ireland.png",
                      "hillforts-topo-ireland-north.png",
                      "hillforts-topo-ireland-south.png"]
```

```

else:
    backgrounds = ["hillforts-outline-01.png",
                   "hillforts-outline-north.png",
                   "hillforts-outline-northwest-plus.png",
                   "hillforts-outline-northwest-minus.png",
                   "hillforts-outline-northeast.png",
                   "hillforts-outline-south.png",
                   "hillforts-outline-south-plus.png",
                   "hillforts-outline-ireland.png",
                   "hillforts-outline-ireland-north.png",
                   "hillforts-outline-ireland-south.png"]
return backgrounds

```

```

In [8]: def get_bounds():
bounds = [[-1200000, 220000, 6400000, 8700000],
          [-1200000, 220000, 7000000, 8700000],
          [-1200000, -480000, 7000000, 8200000],
          [-900000, -480000, 7100000, 8200000],
          [-520000, 0, 7000000, 8700000],
          [-800000, 220000, 6400000, 7100000],
          [-1200000, 220000, 6400000, 7100000],
          [-1200000, -600000, 6650000, 7450000],
          [-1200000, -600000, 7050000, 7450000],
          [-1200000, -600000, 6650000, 7080000]]
return bounds

```

```

In [9]: def show_background(plt, ax, location=""):
backgrounds = get_backgrounds()
bounds = get_bounds()
folder = "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-topo/"
#"https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/
#hillforts-topo/"

if location == "n":
    background = os.path.join(folder, backgrounds[1])
    bounds = bounds[1]
elif location == "nw+":
    background = os.path.join(folder, backgrounds[2])
    bounds = bounds[2]
elif location == "nw-":
    background = os.path.join(folder, backgrounds[3])
    bounds = bounds[3]
elif location == "ne":
    background = os.path.join(folder, backgrounds[4])
    bounds = bounds[4]
elif location == "s":
    background = os.path.join(folder, backgrounds[5])
    bounds = bounds[5]
elif location == "s+":
    background = os.path.join(folder, backgrounds[6])
    bounds = bounds[6]
elif location == "i":
    background = os.path.join(folder, backgrounds[7])
    bounds = bounds[7]
elif location == "in":
    background = os.path.join(folder, backgrounds[8])
    bounds = bounds[8]
elif location == "is":
    background = os.path.join(folder, backgrounds[9])
    bounds = bounds[9]
else:
    background = os.path.join(folder, backgrounds[0])
    bounds = bounds[0]

img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
ax.imshow(img, extent=bounds)

```

```

In [10]: def get_counts(data):
data_counts = []
for col in data.columns:
    count = len(data[data[col] == 'Yes'])
    data_counts.append(count)
return data_counts

```

```

In [11]: def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
               color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
               horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(0.99, 0.01), \
               xycoords='figure fraction', horizontalalignment = 'right')

```

```

In [12]: def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \

```

```

        color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
horizontalalignment = 'left')
ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
size='small', color='grey', xy=(0.99, 0.035), \
xycoords='figure fraction', horizontalalignment = 'right')

```

```
In [13]: def plot_bar_chart(data, split_pos, x_label, y_label, title, clip=False):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    x_data = data.columns
    x_data = [x.split("_")[split_pos:] for x in x_data]
    x_data_new = []
    for l in x_data :
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])
    if clip:
        x_data_new = x_data_new[:-1]
        new_data = data.copy()
        data = new_data.drop(['Dating_Date_Unknown'], axis=1)
    y_data = get_counts(data)
    ax.bar(x_data_new,y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [14]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    ax.bar(x_data,y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [15]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, n_bins, \
extra=''):
    new_data = data.copy()
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    data[x_label].plot(kind='hist', bins = n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    title = f'{title} {extra}'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [16]: def plot_continuous(data, x_label, title):
    fig = plt.figure(figsize=(12,8))
    ax = fig.add_axes([0,0,1,1])
    ax.set_xlabel(x_label)
    plt.plot(data, linewidth=4)
    plt.ticklabel_format(style='plain')
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()
```

```
In [17]: def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12,2))
    ax = fig.add_axes([0,0,1,1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.ticklabel_format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v", whis=[2.2, 97.8])
    else:
        sns.boxplot(data=data, orient="h", whis=[2.2, 97.8])
    save_fig(feature + " Range")
    plt.show()

    bp = boxplot_stats(data, whis=[2.2, 97.8])

    low = bp[0].get('whislo')
    q1 = bp[0].get('q1')
```

```

    median = bp[0].get('med')
    q3 = bp[0].get('q3')
    high = bp[0].get('whishi')

    return [low, q1, median, q3, high]

```

```

In [18]: def plot_data_range_plus(data, feature, o="v"):
    fig = plt.figure(figsize=(12,8))
    ax = fig.add_axes([0,0,1,1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range (Outlier Steps)"))
    plt.ticklabel_format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v", whis=[2.2, 97.8])
    else:
        sns.boxplot(data=data, orient="h", whis=[2.2, 97.8])

    # Add annotation Lines
    x = [24, 24, 54, 54]
    y = [-0.05, -0.075, -0.075, -0.05]
    x1 = [54, 54, 84, 84]
    y1 = [-0.1, -0.125, -0.125, -0.1]
    x2 = [84, 84, 114, 114]
    y2 = [-0.05, -0.075, -0.075, -0.05]

    line_1 = plt.plot(x,y)
    line_2 = plt.plot(x1,y1)
    line_3 = plt.plot(x2,y2)

    # Add annotation text
    text_kwarg = dict(ha='center', va='center', fontsize=16, color='k')
    plt.text(39, -0.1, '30 Ha', **text_kwarg)
    plt.text(69, -0.1, '30 Ha', **text_kwarg)
    plt.text(99, -0.1, '30 Ha', **text_kwarg)

    save_fig(feature + " Range")
    plt.show()

    return

```

```

In [19]: def location_XY_plot():
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(6400000,8700000)
    add_annotation_l_xy=plt

```

```

In [20]: def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')

```

```

In [21]: def plot_over_grey_numeric(merged_data, a_type, title, extra="", inner=False, \
                           fringe=False, oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background=plt, ax
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records=plt, plot_data
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
    print(f'round(((len(plot_data)/4147)*100), 2)%')

```

```

In [22]: def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    682

```

```

fig, ax = plt.subplots(figsize=(9.47, 15.33))
show_background=plt, ax)
location_XY_plot()
add_grey(region='')
plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
show_records=plt, plot_data)
plt.title(get_print_title('Boundary_Type: ' + boundary_type))
save_fig('Boundary_Type_' + boundary_type)
plt.show()

```

In [23]:

```

def add_21Ha_line():
    x_values = \
        [-367969, -344171, -263690, -194654, -130542, -119597, -162994, -265052], -304545]
    y_values = \
        [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 6663609], 6611780]
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, \
              label = '≥ 21 Ha Line')
    add_to_legend = Line2D([0], [0], color='k', lw=15, alpha=0.25, \
              label = '≥ 21 Ha Line')
    return add_to_legend

```

In [24]:

```

def add_21Ha_fringe():
    x_values = \
        [-367969, -126771, 29679, -42657, -248650, -304545, -423647, -584307, -367969]
    y_values = \
        [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, 6752378, 7019842]
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    return add_to_legend

```

In [25]:

```

def plot_density_over_grey(data, data_type, extra='', inner=False, fringe=False):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_density(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background=plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
               c=new_data['Density'], cmap=cm.rainbow, s=25)
    if fringe:
        add_21Ha_fringe()
    if inner:
        add_21Ha_line()
        plt.legend(loc='lower left')
    plt.colorbar(label='Density')
    title = f'Density - {data_type} {extra}'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

In [26]:

```

def plot_density_over_grey_three(data_low, data_iqr, data_high, title, \
                                  extra='', inner=False, fringe=False):
    new_data_low = data_low.copy()
    new_data_low = new_data_low.drop(['Density'], axis=1)
    new_data_low = add_density(new_data_low)

    new_data_iqr = data_iqr.copy()
    new_data_iqr = new_data_iqr.drop(['Density'], axis=1)
    new_data_iqr = add_density(new_data_iqr)

    new_data_high = data_high.copy()
    new_data_high = new_data_high.drop(['Density'], axis=1)
    new_data_high = add_density(new_data_high)

    fig, ax = plt.subplots(1, 3)
    fig.set_figheight(7)
    fig.set_figwidth(15)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/
    # hillforts-topo"
    background = os.path.join(folder, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = plt.imread(background)
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)

    ax[0].scatter(new_data_low['Location_X'], new_data_low['Location_Y'], \
                  c=new_data_low['Density'], cmap=cm.rainbow, s=25)

```

```

ax[1].scatter(new_data_iqr['Location_X'], new_data_iqr['Location_Y'], \
            c=new_data_iqr['Density'], cmap=cm.rainbow, s=25)
ax[2].scatter(new_data_high['Location_X'], new_data_high['Location_Y'], \
            c=new_data_high['Density'], cmap=cm.rainbow, s=25)

ax[0].get_yaxis().set_visible(False)
ax[1].get_yaxis().set_visible(False)
ax[2].get_yaxis().set_visible(False)

ax[0].get_xaxis().set_visible(False)
ax[1].get_xaxis().set_visible(False)
ax[2].get_xaxis().set_visible(False)

ax[0].set_title("1st Quarter (Tiny Hillforts)")
ax[1].set_title("IQR (Small to Medium Hillforts)")
ax[2].set_title("4th Quarter (Large Hillforts)")

fig.suptitle(get_print_title(title), y=1.08)
ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
               color='grey', xy=(0, -0.1), xycoords='axes fraction', \
               horizontalalignment = 'left')
ax[2].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(1, -0.1), \
               xycoords='axes fraction', horizontalalignment = 'right')
save_fig(title)
plt.show()

```

```

In [27]: def plot_density_over_grey_four(data_1, data_2, data_3, data_4, title, \
                                      extra=',', inner=False, fringe=False):
    new_data_1 = data_1.copy()
    new_data_1 = new_data_1.drop(['Density'], axis=1)
    new_data_1 = add_density(new_data_1)

    new_data_2 = data_2.copy()
    new_data_2 = new_data_2.drop(['Density'], axis=1)
    new_data_2 = add_density(new_data_2)

    new_data_3 = data_3.copy()
    new_data_3 = new_data_3.drop(['Density'], axis=1)
    new_data_3 = add_density(new_data_3)

    new_data_4 = data_4.copy()
    new_data_4 = new_data_4.drop(['Density'], axis=1)
    new_data_4 = add_density(new_data_4)

    fig, ax = plt.subplots(1, 4)
    fig.set_figheight(7)
    fig.set_figwidth(20)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/
    # hillforts-topo/"
    background = os.path.join(folder, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = plt.imread(background)
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)
    ax[3].imshow(img, extent=bounds)

    ax[0].scatter(new_data_1['Location_X'], new_data_1['Location_Y'], \
                  c=new_data_1['Density'], cmap=cm.rainbow, s=25)
    ax[1].scatter(new_data_2['Location_X'], new_data_2['Location_Y'], \
                  c=new_data_2['Density'], cmap=cm.rainbow, s=25)
    ax[2].scatter(new_data_3['Location_X'], new_data_3['Location_Y'], \
                  c=new_data_3['Density'], cmap=cm.rainbow, s=25)
    ax[3].scatter(new_data_4['Location_X'], new_data_4['Location_Y'], \
                  c=new_data_4['Density'], cmap=cm.rainbow, s=25)

    ax[0].get_yaxis().set_visible(False)
    ax[1].get_yaxis().set_visible(False)
    ax[2].get_yaxis().set_visible(False)
    ax[3].get_yaxis().set_visible(False)

    ax[0].get_xaxis().set_visible(False)
    ax[1].get_xaxis().set_visible(False)
    ax[2].get_xaxis().set_visible(False)
    ax[3].get_xaxis().set_visible(False)

    ax[0].set_title("NE")
    ax[1].set_title("SE")
    ax[2].set_title("SW")
    ax[3].set_title("NW")

```

```

fig.suptitle(get_print_title(title), y=1.08)
ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
               color='grey', xy=(0, -0.1), xycoords='axes fraction', \
               horizontalalignment = 'left')
ax[3].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(1, -0.1), \
               xycoords='axes fraction', horizontalalignment = 'right')
save_fig(title)
plt.show()

```

```

In [28]: def plot_density_over_grey_five(data_1, data_2, data_3, data_4, data_5, title, \
                                      extra='', inner=False, fringe=False):
    new_data_1 = data_1.copy()
    new_data_1 = new_data_1.drop(['Density'], axis=1)
    new_data_1 = add_density(new_data_1)

    new_data_2 = data_2.copy()
    new_data_2 = new_data_2.drop(['Density'], axis=1)
    new_data_2 = add_density(new_data_2)

    new_data_3 = data_3.copy()
    new_data_3 = new_data_3.drop(['Density'], axis=1)
    new_data_3 = add_density(new_data_3)

    new_data_4 = data_4.copy()
    new_data_4 = new_data_4.drop(['Density'], axis=1)
    new_data_4 = add_density(new_data_4)

    new_data_5 = data_5.copy()
    new_data_5 = new_data_5.drop(['Density'], axis=1)
    new_data_5 = add_density(new_data_5)

    fig, ax = plt.subplots(1, 5)
    fig.set_fignheight(7)
    fig.set_fignwidth(24)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/
    # hillforts-topo/"
    background = os.path.join(folder, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = plt.imread(background)
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)
    ax[3].imshow(img, extent=bounds)
    ax[4].imshow(img, extent=bounds)

    ax[0].scatter(new_data_1['Location_X'], new_data_1['Location_Y'], \
                  c=new_data_1['Density'], cmap=cm.rainbow, s=25)
    ax[1].scatter(new_data_2['Location_X'], new_data_2['Location_Y'], \
                  c=new_data_2['Density'], cmap=cm.rainbow, s=25)
    ax[2].scatter(new_data_3['Location_X'], new_data_3['Location_Y'], \
                  c=new_data_3['Density'], cmap=cm.rainbow, s=25)
    ax[3].scatter(new_data_4['Location_X'], new_data_4['Location_Y'], \
                  c=new_data_4['Density'], cmap=cm.rainbow, s=25)
    ax[4].scatter(new_data_5['Location_X'], new_data_5['Location_Y'], \
                  c=new_data_5['Density'], cmap=cm.rainbow, s=25)

    ax[0].get_yaxis().set_visible(False)
    ax[1].get_yaxis().set_visible(False)
    ax[2].get_yaxis().set_visible(False)
    ax[3].get_yaxis().set_visible(False)
    ax[4].get_yaxis().set_visible(False)

    ax[0].get_xaxis().set_visible(False)
    ax[1].get_xaxis().set_visible(False)
    ax[2].get_xaxis().set_visible(False)
    ax[3].get_xaxis().set_visible(False)
    ax[4].get_xaxis().set_visible(False)

    ax[0].set_title("0")
    ax[1].set_title("1")
    ax[2].set_title("2")
    ax[3].set_title("3")
    ax[4].set_title("4")

    fig.suptitle(get_print_title(title), y=1.08)
    ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                   color='grey', xy=(0, -0.1), xycoords='axes fraction', \
                   horizontalalignment = 'left')
    ax[4].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                   size='small', color='grey', xy=(1, -0.1), \
                   xycoords='axes fraction', horizontalalignment = 'right')

```

```

save_fig(title)
plt.show()

In [29]: def plot_density_over_grey_six(data_1, data_2, data_3, data_4, data_5, \
                                 data_6, title, extra='', inner=False, fringe=False):
    new_data_1 = data_1.copy()
    new_data_1 = new_data_1.drop(['Density'], axis=1)
    new_data_1 = add_density(new_data_1)

    new_data_2 = data_2.copy()
    new_data_2 = new_data_2.drop(['Density'], axis=1)
    new_data_2 = add_density(new_data_2)

    new_data_3 = data_3.copy()
    new_data_3 = new_data_3.drop(['Density'], axis=1)
    new_data_3 = add_density(new_data_3)

    new_data_4 = data_4.copy()
    new_data_4 = new_data_4.drop(['Density'], axis=1)
    new_data_4 = add_density(new_data_4)

    new_data_5 = data_5.copy()
    new_data_5 = new_data_5.drop(['Density'], axis=1)
    new_data_5 = add_density(new_data_5)

    new_data_6 = data_6.copy()
    new_data_6 = new_data_6.drop(['Density'], axis=1)
    new_data_6 = add_density(new_data_6)

    fig, ax = plt.subplots(1, 6)
    fig.set_figheight(6)
    fig.set_figwidth(24)

    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-topo"
    background = os.path.join(folder, "hillforts-bw-02.png")
    bounds = bounds[0]
    img = plt.imread(background)
    ax[0].imshow(img, extent=bounds)
    ax[1].imshow(img, extent=bounds)
    ax[2].imshow(img, extent=bounds)
    ax[3].imshow(img, extent=bounds)
    ax[4].imshow(img, extent=bounds)
    ax[5].imshow(img, extent=bounds)

    ax[0].scatter(new_data_1['Location_X'], new_data_1['Location_Y'], \
                  c=new_data_1['Density'], cmap=cm.rainbow, s=25)
    ax[1].scatter(new_data_2['Location_X'], new_data_2['Location_Y'], \
                  c=new_data_2['Density'], cmap=cm.rainbow, s=25)
    ax[2].scatter(new_data_3['Location_X'], new_data_3['Location_Y'], \
                  c=new_data_3['Density'], cmap=cm.rainbow, s=25)
    ax[3].scatter(new_data_4['Location_X'], new_data_4['Location_Y'], \
                  c=new_data_4['Density'], cmap=cm.rainbow, s=25)
    ax[4].scatter(new_data_5['Location_X'], new_data_5['Location_Y'], \
                  c=new_data_5['Density'], cmap=cm.rainbow, s=25)
    ax[5].scatter(new_data_5['Location_X'], new_data_5['Location_Y'], \
                  c=new_data_5['Density'], cmap=cm.rainbow, s=25)

    ax[0].get_yaxis().set_visible(False)
    ax[1].get_yaxis().set_visible(False)
    ax[2].get_yaxis().set_visible(False)
    ax[3].get_yaxis().set_visible(False)
    ax[4].get_yaxis().set_visible(False)
    ax[5].get_yaxis().set_visible(False)

    ax[0].get_xaxis().set_visible(False)
    ax[1].get_xaxis().set_visible(False)
    ax[2].get_xaxis().set_visible(False)
    ax[3].get_xaxis().set_visible(False)
    ax[4].get_xaxis().set_visible(False)
    ax[5].get_xaxis().set_visible(False)

    ax[0].set_title("Part Univallate")
    ax[1].set_title("Univallate")
    ax[2].set_title("Part Bivallate")
    ax[3].set_title("Bivallate")
    ax[4].set_title("Part Multivallate")
    ax[5].set_title("Multivallate")

    fig.suptitle(get_print_title(title), y=1.08)
    ax[0].annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                  color='grey', xy=(0, -0.1), xycoords='axes fraction', \
                  horizontalalignment = 'left')

```

```

        ax[5].annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                       size='small', color='grey', xy=(1, -0.1), \
                       xycoords='axes fraction', horizontalalignment = 'right')
    save_fig(title)
    plt.show()

```

In [30]:

```

def add_oxford_swindon(oxford=False, swindon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_size = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_size, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swindon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_size, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches

```

In [31]:

```

def plot_over_grey(merged_data, a_type, yes_no, extra="", inner=False, \
                   fringe=False, oxford=False, swindon=False, topo=False):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background=plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records=plt, plot_data)
    plt.legend(loc='upper left', handles= patches)
    plt.title(get_print_title(f'{a_type} {extra}'))
    save_fig(f'{a_type}_{extra}')
    plt.show()
    print(f'{round(((len(plot_data)/4147)*100), 2)}%')
    return plot_data

```

In [32]:

```

def plot_type_values(data, data_type, title, extra=''):
    new_data = data.copy()
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background=plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
               c=new_data[data_type], cmap=cm.rainbow, s=25)
    plt.colorbar(label=data_type)
    title = f'{data_type} {extra}'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

In [33]:

```

def bespoke_plot(plt, title):
    add_annotation_plot(plt)
    plt.ticklabel_format(style='plain')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

In [34]:

```

def add_cluster_split_lines(plt, ax, extra=None):
    x_min = -550000
    if extra == 'ireland':
        x_min = -1200000
    plt.vlines(x=[-500000], ymin=7070000, ymax=9000000, colors='r', ls='-', lw=3)
    plt.hlines(y=[7070000], xmin=x_min, xmax=200000, colors='r', ls='-', lw=3)
    ax.annotate("N/S split", color='k', xy=(50000, 7090000), xycoords='data')
    ax.annotate("E/W split", color='k', xy=(-480000, 8660000), xycoords='data')
    ax.annotate("Irish Sea split", color='k', xy=(-1150000, 7510000), \
                xycoords='data')
    plot_line((-800000, 6400000), (-550000, 7070000))
    plot_line((-550000, 7070000), (-666000, 7440000))
    plot_line((-666000, 7440000), (-900000, 7500000))

```

In [35]:

```

def plot_values(data, feature, title, extra=''):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))

```

```

show_background(plt, ax)
location_XY_plot()
plt.scatter(data['Location_X'], data['Location_Y'], c=data[feature], \
            cmap=cm.rainbow, s=25)
plt.colorbar(label=feature)
title = f'{title} {extra}'
plt.title(title)
save_fig(title)
plt.show()

```

```
In [36]: def plot_line(point1, point2):
    x_values = [point1[0], point2[0]]
    y_values = [point1[1], point2[1]]
    plt.plot(x_values, y_values, 'r', ls='0', lw=2, alpha=1)
```

```
In [37]: def density_scatter_lines(location_data, scatter_data, plot_title, inner=False, \
                                fringe=False):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(location_data['Location_X'], location_data['Location_Y'], \
                c=location_data['Density_trans'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density Transformed')
    if inner:
        add_21Ha_line()
    if fringe:
        add_21Ha_fringe()
    plt.scatter(scatter_data['Location_X'], scatter_data['Location_Y'], c='Red')
    plt.legend(loc='lower left')
    plt.title(get_print_title(plot_title))
    save_fig(plot_title)
    plt.show()
```

```
In [38]: def add_limits():
    x_values = [-584307, 115292]
    y_values = [7019842, 7019842]
    xx_values = [115292, 115292]
    yy_values = [7019842, 6485285]
    xxx_values = [-584307, 115292]
    yyy_values = [6485285, 6485285]
    x4_values = [-584307, -584307]
    y4_values = [7019842, 6485285]

    plt.plot(x_values, y_values, 'g', ls='-', lw=8, alpha=0.6, \
              label = 'Bounds of test')
    plt.plot(xx_values, yy_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(xxx_values, yyy_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(x4_values, y4_values, 'g', ls='-', lw=8, alpha=0.6)
```

```
In [39]: def add_linear_south():
    x_values = [-115637, -286900]
    y_values = [6678188, 6585812]
    xx_values = [-244249, -363049]
    yy_values = [6555133, 6589612]
    xxx_values = [-392213, -363146]
    yyy_values = [6577365, 6647256]
    x4_values = [-169664, -207084]
    y4_values = [6599254, 6615290]
    x5_values = [-238560, -200891]
    y5_values = [6668083, 6637826]

    plt.plot(x_values, y_values, 'g', ls='-', lw=8, alpha=0.6, \
              label = 'Poss. correlation to linear routes?')
    plt.plot(xx_values, yy_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(xxx_values, yyy_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(x4_values, y4_values, 'g', ls='-', lw=8, alpha=0.6)
    plt.plot(x5_values, y5_values, 'g', ls='-', lw=8, alpha=0.6)
```

```
In [40]: def plot_over_grey_south(merged_data, a_type, yes_no, extra=""):
    # plots selected data over the grey dots. yes_no controls filtering the
    # data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(1, figsize=((6.73*1.5), (4.62*1.5)))
    show_background(plt, ax, 's')
    plt.ticklabel_format(style='plain')
    plt.xlim(-800000, 220000)
    plt.ylim(6400000, 7100000)
    add_annotation_l_xy=plt
    add_grey('s')
    add_grey('s')
    add_linear_south()
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    plt.legend(loc='lower right')
```

```

plt.title(get_print_title(f'{a_type} {extra}'))
save_fig(f'{a_type}_{extra}')
plt.show()
return plot_data

```

```

In [41]: def get_proportions(date_set):
    total = sum(date_set) - date_set[-1]
    newset = []
    for entry in date_set[:-1]:
        newset.append(round(entry/total,2))
    return newset

```

```

In [42]: def plot_dates_by_region(nw,ne,ni,si,s, features):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    x_data = nw[features].columns
    x_data = [x.split("_")[2:] for x in x_data][:-1]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])

    set1_name = 'NW'
    set2_name = 'NE'
    set3_name = 'N Ireland'
    set4_name = 'S Ireland'
    set5_name = 'South'
    set1 = get_proportions(get_counts(nw[features]))
    set2 = get_proportions(get_counts(ne[features]))
    set3 = get_proportions(get_counts(ni[features]))
    set4 = get_proportions(get_counts(si[features]))
    set5 = get_proportions(get_counts(s[features]))

    X_axis = np.arange(len(x_data_new))

    budge = 0.25

    plt.bar(X_axis - 0.55 + budge, set1, 0.3, label = set1_name)
    plt.bar(X_axis - 0.4 + budge, set2, 0.3, label = set2_name)
    plt.bar(X_axis - 0.25 + budge, set3, 0.3, label = set3_name)
    plt.bar(X_axis - 0.1 + budge, set4, 0.3, label = set4_name)
    plt.bar(X_axis + 0.05 + budge, set5, 0.3, label = set5_name)

    plt.xticks(X_axis, x_data_new)
    plt.xlabel('Dating')
    plt.ylabel('Proportion of Total Dated Hillforts in Region')
    title = 'Proportions of Dated Hillforts by Region'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [43]: def get_pcent_list(old_list):
    pcnt_list = []
    total = sum(old_list)
    for item in old_list:
        pcnt_list.append(round(item/total,2))
    return pcnt_list

```

```

In [44]: def order_set(set_list, x_data, pcnt=False):
    new_list = []
    set_values = set_list.index.tolist()
    for val in x_data:
        if val in set_values:
            new_list.append(set_list.loc[[val]].values[0])
        else:
            new_list.append(0)
    if pcnt:
        new_list = get_pcent_list(new_list)
    return new_list

```

```

In [45]: def plot_feature_by_region(nw,ne,ni,si,s, feature, title, clip):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    max_val = int(max([nw[feature].max(),ne[feature].max(),\
                      ni[feature].max(),si[feature].max(),s[feature].max()]))+2

    x_data = [x-1 for x in range(max_val+2)]

    set0_name = 'NW'
    set1_name = 'NE'

```

```

set2_name = 'N Ireland'
set3_name = 'S Ireland'
set4_name = 'S'

set0 = nw[feature].value_counts()
set1 = ne[feature].value_counts()
set2 = ni[feature].value_counts()
set3 = si[feature].value_counts()
set4 = s[feature].value_counts()

set0 = order_set(set0,x_data, True)[:clip]
set1 = order_set(set1,x_data, True)[:clip]
set2 = order_set(set2,x_data, True)[:clip]
set3 = order_set(set3,x_data, True)[:clip]
set4 = order_set(set4,x_data, True)[:clip]

X_axis = np.arange(len(x_data[:clip]))

budge = 0.2

plt.bar(X_axis - 0.6 + budge, set0, 0.3, label = set0_name)
plt.bar(X_axis - 0.45 + budge, set1, 0.3, label = set1_name)
plt.bar(X_axis - 0.3 + budge, set2, 0.3, label = set2_name)
plt.bar(X_axis - 0.15 + budge, set3, 0.3, label = set3_name)
plt.bar(X_axis + 0 + budge, set4, 0.3, label = set4_name)

plt.xticks(X_axis, x_data)
plt.xlabel('Number')
plt.ylabel('Percentage of regional total')
plt.title(title)
plt.legend()
add_annotation_plot(ax)
save_fig(title)
plt.show()

```

```

In [46]: def plot_quadrants(ramparts,ditches,ne,se,sw,nw):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    x_data = [x for x in range(11)]

    set0_name = 'Ramparts'
    set00_name = 'Ditches'
    set1_name = 'NE'
    set2_name = 'SE'
    set3_name = 'SW'
    set4_name = 'NW'
    set0 = ramparts['Enclosing_Max_Ramparts'].value_counts()
    set0 = order_set(set0,x_data)[:8]
    set00 = ditches['Enclosing_Ditches_Number'].value_counts()
    set00 = order_set(set00,x_data)[:8]
    set1 = ne['Enclosing_NE_Quadrant'].value_counts()
    set1 = order_set(set1,x_data)[:8]
    set2 = se['Enclosing_SE_Quadrant'].value_counts()
    set2 = order_set(set2,x_data)[:8]
    set3 = sw['Enclosing_SW_Quadrant'].value_counts()
    set3 = order_set(set3,x_data)[:8]
    set4 = nw['Enclosing_NW_Quadrant'].value_counts()
    set4 = order_set(set4,x_data)[:8]

    X_axis = np.arange(len(x_data[:8]))

    budge = 0.2

    plt.bar(X_axis - 0.6 + budge, set0, 0.2, label = set0_name)
    plt.bar(X_axis - 0.46 + budge, set00, 0.2, label = set00_name)
    plt.bar(X_axis - 0.32 + budge, set1, 0.2, label = set1_name)
    plt.bar(X_axis - 0.18 + budge, set2, 0.2, label = set2_name)
    plt.bar(X_axis - 0.04 + budge, set3, 0.2, label = set3_name)
    plt.bar(X_axis + 0.1 + budge, set4, 0.2, label = set4_name)

    plt.xticks(X_axis, x_data)
    plt.xlabel('Number')
    plt.ylabel('Count')
    title = 'Ditches, Ramparts and Quadrant by Number'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [47]: def plot_regions(nw,ne,ni,si,s, features, xlabel, title, split_pos, \
                     yes_no, pcent=False):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])

```

```

x_data = features
x_data = [x.split("_")[split_pos:] for x in x_data]
x_data_new = []
for l in x_data:
    txt = ""
    for part in l:
        txt += "_" + part
    x_data_new.append(txt[1:])

set0_name = 'NW'
set1_name = 'NE'
set2_name = 'N Ireland'
set3_name = 'S Ireland'
set4_name = 'S'

set0_list = []
set1_list = []
set2_list = []
set3_list = []
set4_list = []

for feature in features:
    set0_list.append((nw[feature].values == yes_no).sum())
    set1_list.append((ne[feature].values == yes_no).sum())
    set2_list.append((ni[feature].values == yes_no).sum())
    set3_list.append((si[feature].values == yes_no).sum())
    set4_list.append((s[feature].values == yes_no).sum())

set0 = set0_list
set1 = set1_list
set2 = set2_list
set3 = set3_list
set4 = set4_list

if pcent:
    set0 = get_pcent_list(set0)
    set1 = get_pcent_list(set1)
    set2 = get_pcent_list(set2)
    set3 = get_pcent_list(set3)
    set4 = get_pcent_list(set4)

X_axis = np.arange(len(x_data))

budge = 0.3

plt.bar(X_axis - 0.6 + budge, set0, 0.13, label = set0_name)
plt.bar(X_axis - 0.45 + budge, set1, 0.13, label = set1_name)
plt.bar(X_axis - 0.3 + budge, set2, 0.13, label = set2_name)
plt.bar(X_axis - 0.15 + budge, set3, 0.13, label = set3_name)
plt.bar(X_axis + 0 + budge, set4, 0.13, label = set4_name)

plt.xticks(X_axis, x_data_new)
plt.xlabel(xlabel)
if pcent:
    plt.ylabel('Percentage of Regional Total')
else:
    plt.ylabel('Count')
plt.title(get_print_title(f'{title}'))
plt.legend()
add_annotation_plot(ax)
save_fig(title)
plt.show()

```

In [48]:

```

def plot_data_range_95(data, feature, title, o="v"):
    fig = plt.figure(figsize=(12,2))
    ax = fig.add_axes([0,0,1,1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(title)
    plt.ticklabel_format(style='plain')
    plt.axvline(x=mse_h1, color='r', lw=4)
    if o == "v":
        sns.boxplot(data=data, orient="v", whis=[2.5, 97.5])
    else:
        sns.boxplot(data=data, orient="h", whis=[2.5, 97.5])

    plot_title = get_print_title(title)
    save_fig(plot_title)
    plt.show()

    bp = boxplot_stats(data, whis=[2.5, 97.5])

    low = bp[0].get('whislo')
    q1 = bp[0].get('q1')
    median = bp[0].get('med')

```

```

q3 = bp[0].get('q3')
high = bp[0].get('whishi')

return [low, q1, median, q3, high]

```

```

In [49]: def plot_mse_histogram(mse_list, mse_h1):
    fig = plt.figure(figsize=(12,5))
    ax = fig.add_axes([0,0,1,1])
    ax.set_xlabel('MSE')
    ax.set_ylabel('Count')
    plt.ticklabel_format(style='plain')
    plt.hist(mse_list, 100)
    plot_title = \
        get_print_title('Mean Squared Error for hillforts offset from test lines - H1 MSE shown in red.')
    #'Mean Squared Error for hillforts offset from test Lines -
    # H1 MSE shown in red.'
    plt.title(plot_title)
    plt.axvline(x=mse_h1, color='r', lw=4)
    add_annotation_plot(ax)
    save_fig(plot_title)
    plt.show()

```

```

In [50]: def rotate_around_point(xy, radians, rotation_origin=(0, 0)):
    x, y = xy
    offset_x, offset_y = rotation_origin
    adjusted_x = (x - offset_x)
    adjusted_y = (y - offset_y)
    cos_rad = math.cos(radians)
    sin_rad = math.sin(radians)
    qx = offset_x + cos_rad * adjusted_x + sin_rad * adjusted_y
    qy = offset_y + -sin_rad * adjusted_x + cos_rad * adjusted_y

    return qx, qy

```

```

In [51]: def get_random_line(dist):
    test_bounds_x = [-584307, 115292]
    test_bounds_y = [6485285, 7019842]
    rand_point = (random.randrange(test_bounds_x[0],test_bounds_x[1]), \
                  random.randrange(test_bounds_y[0],test_bounds_y[1]))
    new_second_point = (-1000000002, 1000000000)

    count = 0
    while (new_second_point[0] <= test_bounds_x[0]) or \
           (new_second_point[0] >= test_bounds_x[1]) or \
           (new_second_point[1] <= test_bounds_y[0]) or \
           (new_second_point[1] >= test_bounds_y[1]):
        rand_second_point = (rand_point[0]+dist, rand_point[1])
        rand_angle_rad = random.uniform(0.0,3.14)
        new_second_point = rotate_around_point(rand_second_point, \
                                               rand_angle_rad, rand_point)
        count+=1
    if count == 10:
        rand_point = (random.randrange(test_bounds_x[0],test_bounds_x[1]), \
                      random.randrange(test_bounds_y[0],test_bounds_y[1]))
        count = 0
    return [rand_point, new_second_point, math.degrees(rand_angle_rad)]

```

```

In [52]: def line_intersection(line1, line2):
    x = 1000000
    y = 1000000
    xdiff = (line1[0][0] - line1[1][0], line2[0][0] - line2[1][0])
    ydiff = (line1[0][1] - line1[1][1], line2[0][1] - line2[1][1])

    def det(a, b):
        return a[0] * b[1] - a[1] * b[0]

    div = det(xdiff, ydiff)
    if div == 0:
        pass
    else:
        d = (det(*line1), det(*line2))
        x = det(d, xdiff) / div
        y = det(d, ydiff) / div

    return x, y

```

```

In [53]: def show_perpendicular_lines(points, x_values, y_values, angle, scatter_data, \
                                show_perpendicular):
    length_divider = 4.85
    distance_list = []

    x_bounds = [-584307,115292]
    y_bounds = [6485285,7019842]

```

```

max_len = \
math.sqrt( (x_bounds[0] - x_bounds[1])**2 + (y_bounds[0] - y_bounds[1])**2 )
x0 = points[0][0]
y0 = points[0][1]
x1 = points[1][0]
y1 = points[1][1]
line_dist = math.sqrt( (x0 - x1)**2 + (y0 - y1)**2 )
for index, row in scatter_data.iterrows():
    x2 = row['Location_X']
    y2 = row['Location_Y']
    if x2 >= x_bounds[0] and x2 <= x_bounds[1] and \
    y2 >= y_bounds[0] and y2 <= y_bounds[1]:
        p1 = (x2,y2)
        p2 = (x2+max_len,y2)
        new_p2 = p2
        value = ((x1 - x0)*(y2 - y0)) - ((x2 - x0)*(y1 - y0))
        if value < 0:
            pass
        new_p2 = rotate_around_point(p2, math.radians((angle-90)), p1)
    elif value > 0:
        new_p2 = rotate_around_point(p2, math.radians((angle+90)), p1)
    else:
        # on Line
        new_p2 = p1
    line1 = ((x_values[0], y_values[0]), (x_values[1], y_values[1]))
    line2 = ((x2, y2), new_p2)
    x3, y3 = line_intersection(line1, line2)

    if x3 == 1000000:
        x3 = x2
        y3 = y2

    px_values = [x2, x3]
    py_values = [y2, y3]

    #set east/west
    west = x0
    east = x1
    if east < west:
        west = x1
        east = x0

    #set north/south
    north = y0
    south = y1
    if east < west:
        north = y1
        greater_than_21ha_south = y0

    if x3 >= west and x3 <= east and y3 >= south and y3 <= north:
        dist = \
        abs(math.sqrt( (px_values[1] - px_values[0])**2 + \
        (py_values[1] - py_values[0])**2 ))
        if dist < line_dist/length_divider:
            #print(row['Main_Atlas_Number'])
            distance_list.append(dist)
            if show_perpendicular:
                plt.plot(px_values, py_values, 'magenta', ls='-', \
                lw=3, alpha=0.6)
return distance_list

```

```
In [54]: def angle_between(p1, p2):
    ang1 = np.arctan2(*p1[::-1])
    ang2 = np.arctan2(*p2[::-1])
    return np.rad2deg((ang1 - ang2) % (2 * np.pi))
```

```
In [55]: def add_h1_line(scatter_data, show_perpendicular_from_main):
    penycloddiau_m = (312888, 367647)
    bozedown_m = (464350, 178250)
    x_values = [-376969, -119597]
    y_values = [7019842, 6710127]
    angle_rad = \
    math.atan((bozedown_m[1] - penycloddiau_m[1])/(bozedown_m[0] - \
    penycloddiau_m[0]))
    angle = \
    angle_between((x_values[0], y_values[0]), (x_values[1], y_values[1]))+45
    plt.plot(x_values, y_values, 'b', ls='-', lw=8, alpha=0.6, label = 'H1')

    points = [(x_values[0], y_values[0]), (x_values[1], y_values[1]), angle]
    distance_list = []

    if show_perpendicular_from_main:
        distance_list = show_perpendicular_lines(points, x_values, y_values, \
695
```

```

angle, scatter_data, \
show_perpendicular=True)

return distance_list

```

```
In [56]: def add_random_lines(scatter_data, test_lines, dist, show_perpendicular):
    add_lines = test_lines

    distance_lists = []

    for i in range(add_lines):
        points = get_random_line(dist)
        x_values = [points[0][0], points[1][0]]
        y_values = [points[0][1], points[1][1]]
        angle = points[2]
        plt.plot(x_values, y_values, 'darkmagenta', ls='-', lw=3, alpha=0.6)
        distance_list = []

        distance_list = show_perpendicular_lines(points, x_values, \
                                                    y_values, angle, scatter_data, \
                                                    show_perpendicular)
        distance_lists.append(distance_list)
    return distance_lists
```

```
In [57]: def south_density_scatter_lines(location_data, scatter_data, plot_title, \
                                      inner=False, fringe=False, h1=False, \
                                      limits=False, random_lines=False, test_lines=0, \
                                      show_perpendicular=False, \
                                      show_perpendicular_from_main=False):
    penycloddiau = (-367969, 7019842)
    bozedown = (-119597, 6710127)
    dist = math.sqrt( (bozedown[0] - penycloddiau[0])**2 + (bozedown[1] - \
                                                               penycloddiau[1])**2 )

    h1_distance_list = []
    distance_lists = []
    fig, ax = plt.subplots(figsize=((6.73*1.5)+2.0, (4.62*1.5)))
    show_background(plt, ax, 's')
    plt.ticklabel_format(style='plain')
    plt.xlim(-800000, 2200000)
    plt.ylim(6400000, 7100000)
    plt.scatter(location_data['Location_X'], location_data['Location_Y'], \
               c=location_data['Density_trans'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density Transformed')
    if inner:
        add_21Ha_line()
    if fringe:
        add_21Ha_fringe()
    if h1:
        h1_distance_list = add_h1_line(scatter_data, show_perpendicular_from_main)
    if limits:
        add_limits()
    if random_lines:
        distance_lists = add_random_lines(scatter_data, test_lines, dist, \
                                           show_perpendicular)
    plt.scatter(scatter_data['Location_X'], scatter_data['Location_Y'], \
               c='red', s=60)
    add_annotation_plot(plt)
    plt.legend(loc='lower right')
    plt.title(get_print_title(plot_title))
    save_fig(plot_title)
    plt.show()

    return h1_distance_list, distance_lists
```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

```
In [58]: def test_numeric(data):
    temp_data = data.copy()
    columns = data.columns
    out_cols = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
    feat, ent, num, non, nul = [], [], [], [], []
    for col in columns:
        if temp_data[col].dtype == 'object':
            feat.append(col)
            temp_data[col+'_num'] = temp_data[col].str.isnumeric()
            entries = temp_data[col].notnull().sum()
            true_count = \
                temp_data[col+'_num'][temp_data[col+'_num'] == True].sum()
            null_count = temp_data[col].isna().sum()
            ent.append(entries)
            num.append(true_count)
            non.append(entries-true_count)
```

```

        nul.append(null_count)
    else:
        print(f'{col} {temp_data[col].dtype}')
summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
summary.columns = out_cols
return summary

```

```

In [59]: def find_duplicated(numeric_data, text_data, encodeable_data):
    d = False
    all_columns = list(numeric_data.columns) + list(text_data.columns) + \
    list(encodeable_data.columns)
    duplicate = \
        [item for item, count in collections.Counter(all_columns).items() if \
        count > 1]
    if duplicate :
        print(f"There are duplicate features: {duplicate}")
        d = True
    return d

```

```

In [60]: def test_data_split(main_data, numeric_data, text_data, encodeable_data):
    m = False
    split_features = \
        list(numeric_data.columns) + list(text_data.columns) + \
        list(encodeable_data.columns)
    missing = list(set(main_data)-set(split_features))
    if missing:
        print(f"There are missing features: {missing}")
        m = True
    return m

```

```

In [61]: def review_data_split(main_data, numeric_data, text_data, \
                           encodeable_data = pd.DataFrame()):
    d = find_duplicated(numeric_data, text_data, encodeable_data)
    m = test_data_split(main_data, numeric_data, text_data, encodeable_data)
    if d != True and m != True:
        print("Data split good.")

```

```

In [62]: def find_duplicates(data):
    print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')

```

```

In [63]: def count_yes(data):
    total = 0
    for col in data.columns:
        count = len(data[data[col] == 'Yes'])
        total+= count
        print(f'{col}: {count}')
    print(f'Total yes count: {total}')

```

Null Value Functions

The following functions will be used to update null values.

```

In [64]: def fill_nan_with_minus_one(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna(-1)
    return new_data

```

```

In [65]: def fill_nan_with_NA(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna("NA")
    return new_data

```

```

In [66]: def test_numeric_value_in_feature(feature, value):
    test = feature.isin([-1]).sum()
    return test

```

```

In [67]: def test_catagorical_value_in_feature(dataframe, feature, value):
    test = dataframe[feature][dataframe[feature] == value].count()
    return test

```

```

In [68]: def test_cat_list_for_NA(dataframe, cat_list):
    for val in cat_list:
        print(val, test_catagorical_value_in_feature(dataframe, val,'NA'))

```

```

In [69]: def test_num_list_for_minus_one(dataframe, num_list):
    for val in num_list:
        feature = dataframe[val]
        print(val, test_numeric_value_in_feature(feature, -1))

```

```
In [70]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data
```

```
In [71]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data
```

Reprocessing Functions

```
In [72]: def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data
```

In [72]:

Save Image Functions

```
In [73]: # Set-up figure numbering
fig_no = 0
part = 'Appendix01'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"
```

```
In [74]: # Remove unicode characters from file names
def get_file_name(title):
    file_name = slugify(title)
    return file_name
```

```
In [75]: # Remove underscore from figure titles
def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(", ", ";")
    return title
```

```
In [76]: # Format figure numbers to have three digits
def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no
```

```
In [77]: # Mount Google Drive if figures to be saved
if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass
```

```
In [78]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Appendix_01_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution,
                    bbox_inches='tight')
    else:
        pass
```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [79]: hillforts_csv = r"https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/main/hillforts-atlas-source-data-csv"
# "https://raw.githubusercontent.com/MikeDairsie/Hillforts-Primer/
#main/hillforts-atlas-source-data-csv/hillforts.csv"
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)
```

```
<ipython-input-79-a029b1de176b>:4: DtypeWarning: Columns (10,12,68,83,84,85,86,165,183) have mixed types. Specify dt
ype option on import or set low_memory=False.
    hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

```
Out[79]: OBJECTID Main_Atlas_Number Main_Country_Code Main_Country Main_Title_Name Main_Site_Name Main_Alt_Name Main_Display_N
```

0	1	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconbury Beacon	Aconbury C Hereford (Aconbury Bea
1	2	2	EN	England	EN0002 Bach Camp, Herefordshire	Bach Camp	NaN	Bach C Hereford

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [80]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
                      'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.)
```

Using all 4147 record in the Hillforts Atlas.

Load Date Data

```
In [81]: date_features = [
    'Dating_Date_Pre_1200BC',
    'Dating_Date_1200BC_800BC',
    'Dating_Date_800BC_400BC',
    'Dating_Date_400BC_AD50',
    'Dating_Date_AD50_AD400',
    'Dating_Date_AD400_AD800',
    'Dating_Date_Post_AD800',
    'Dating_Date_Unknown']
```

```
date_data = hillforts_data[date_features]
```

Download Function

```
In [82]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', \
                                'republic-of-ireland', 'northern-ireland', \
                                'isle-of-man', 'roi-ni', 'eng-wal-sco-iom', \
                                'twenty-one-ha-and-over-south']:
                if pkg.shape[0] == hillforts_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
            else:
                dl = data_list[0]
        dl = dl.replace('\r', ' ', regex=True)
```

```

        d1 = d1.replace('\n',' ', regex=True)
        fn = 'hillforts_primer_` + filename
        fn = get_file_name(fn)
        d1.to_csv(fn+'.csv', index=False)
        files.download(fn+'.csv')
    else:
        pass

```

Reload Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-data packages. Where needed, these data extracts will be combined with Name and Number features to ensure the data can be understood and can, if needed, be concorded.

```
In [83]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Reload Location

```
In [84]: location_numeric_data_short_features = \
['Location_X', 'Location_Y', 'Main_X', 'Main_Y']
location_numeric_data_short = \
hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short)
location_numeric_data_short.head()
location_data = location_numeric_data_short.copy()
location_data.head()
```

	Location_X	Location_Y	Main_X	Main_Y	Density
0	-303295	6798973	350350	233050	1.632859e-12
1	-296646	6843289	354700	260200	1.540172e-12
2	-289837	6808611	358700	238900	1.547729e-12
3	-320850	6862993	340000	272400	1.670548e-12
4	-261765	6810587	376000	240000	1.369981e-12

Reload Regional Data Packages

See Cluster Data Packages in Part 1: Name, Admin & Location Data

<https://colab.research.google.com/drive/1C7HcuLuGGhG8o4EGciS-XTAhxVs3MhX3?usp=sharing>

```
In [85]: cluster_data = \
hillforts_data[['Location_X', 'Location_Y', 'Main_Country_Code']].copy()
cluster_data['Cluster'] = 'NA'
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != 'NI', 'I', \
inplace=True)
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != 'IR', 'I', \
inplace=True)

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000) , \
    'North Ireland', cluster_data['Cluster']
)
north_ireland = cluster_data[cluster_data['Cluster'] == 'North Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , \
    'South Ireland', cluster_data['Cluster']
)
south_ireland = cluster_data[cluster_data['Cluster'] == 'South Ireland'].copy()
```

```

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] < 7070000) , \
    'South', cluster_data['Cluster']
)
south = cluster_data[cluster_data['Cluster'] == 'South'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] >= -500000), 'Northeast', cluster_data['Cluster']
)
north_east = cluster_data[cluster_data['Cluster'] == 'Northeast'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] < -500000), 'Northwest', cluster_data['Cluster']
)
north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()

temp_cluster_location_packages = [north_ireland, south_ireland, south, \
                                   north_east, north_west]

cluster_packages = []
for pkg in temp_cluster_location_packages:
    pkg = pkg.drop(['Main_Country_Code'], axis=1)
    cluster_packages.append(pkg)

north_ireland, south_ireland, south, north_east, north_west = \
cluster_packages[0], cluster_packages[1], cluster_packages[2], \
cluster_packages[3], cluster_packages[4]

```

Reload Enclosing Area 1

```
In [86]: enclosing_data = hillforts_data.copy()
enclosing_numeric_features = ['Enclosing_Area_1']
enclosing_numeric_data = enclosing_data[enclosing_numeric_features].copy()
```

Enclosing Numeric Data - Resolve Null Values

Test for -1.

```
In [87]: test_num_list_for_minus_one(enclosing_numeric_data, enclosing_numeric_features)
Enclosing_Area_1 0

Replace null with -1.
```

```
In [88]: enclosing_numeric_data = \
update_num_list_for_minus_one(enclosing_numeric_data, enclosing_numeric_features)
enclosing_numeric_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 1 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Enclosing_Area_1  4147 non-null   float64 
dtypes: float64(1)
memory usage: 32.5 KB
```

Appendix 1

Enclosing Area 1: Distribution of Outliers Over 21 Ha Mapped

After multiple tests to filter the data for forts over various sizes, a possible alignment of hillforts was isolated for forts over 21 Ha.

```
In [89]: enclosing_numeric_data = \
update_num_list_for_minus_one(enclosing_numeric_data, enclosing_numeric_features)
location_enclosing_data = \
pd.merge(location_numeric_data_short, enclosing_numeric_data, left_index=True, \
        right_index=True)
enclosing_area_1_21 = location_enclosing_data.copy()
enclosing_area_1_21 = \
enclosing_area_1_21[enclosing_area_1_21['Enclosing_Area_1']>=21]
enclosing_area_1_21['Enclosing_Area_1'].describe()
```

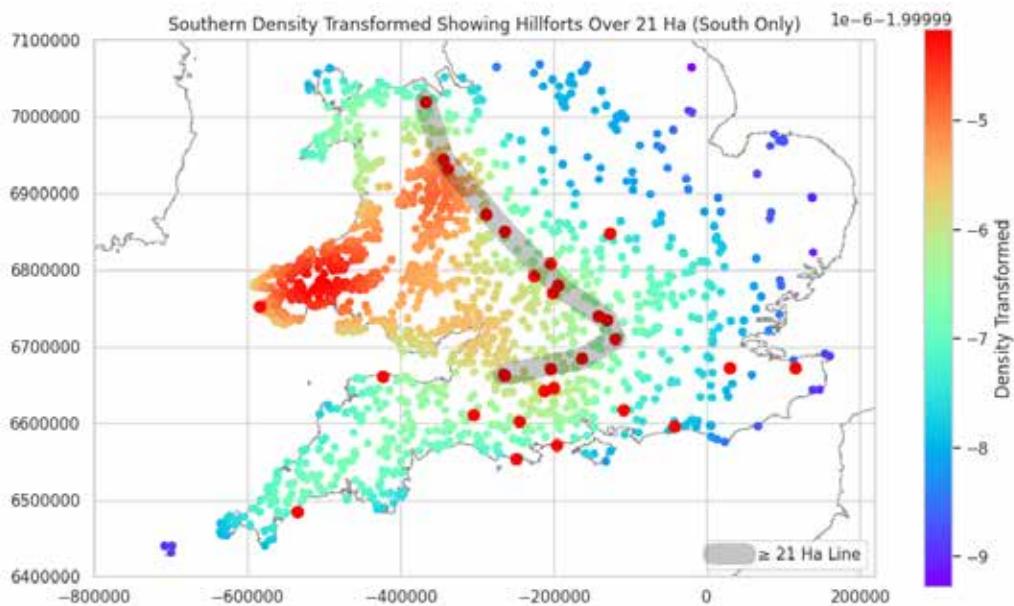
```
Out[89]: count    38.000000
mean      42.501053
std       26.680691
min       21.000000
25%      24.875000
50%      30.000000
75%      51.875000
max      130.000000
Name: Enclosing_Area_1, dtype: float64
```

Enclosing Area 1: Southern Hillfort Density Transformed overlayed by hillforts over 21 Ha

Map showing possible alignment of hillforts which are 21 hectares or larger.

```
In [90]: cluster_south = south.copy()
enclosing_area_1_21_s = \
enclosing_area_1_21[enclosing_area_1_21['Location_X'] > -600000]
cluster_south = add_density(cluster_south)
cluster_south['Density_trans'] = stats.boxcox(cluster_south['Density'], 0.5)
```

```
In [91]: _, _ = \
south_density_scatter_lines(cluster_south, enclosing_area_1_21_s, \
                             'Southern Density Transformed Showing Hillforts Over 21 Ha (South Only)', \
                             True, False)
# 'Southern Density Transformed Showing Hillforts Over 21 Ha (South Only)'
```



Middleton, M. 2024. Hillforts Primer

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Hillforts 21 Ha and over in the South

A full list of hillforts over 21 hectares in the southern data package.

```
In [92]: greater_than_21ha_south = \
pd.merge(name_and_number, enclosing_area_1_21_s, left_index=True, \
         right_index=True)
twenty_one_ha_and_over_south = \
greater_than_21ha_south\
[['Main_Atlas_Number', 'Main_Display_Name', 'Enclosing_Area_1', 'Location_X', \
'Location_Y', 'Main_X', 'Main_Y']].\
copy().sort_values(by='Enclosing_Area_1').style.hide_index()
twenty_one_ha_and_over_south = twenty_one_ha_and_over_south.data
twenty_one_ha_and_over_south
```

```
<ipython-input-92-2737903187fa>:8: FutureWarning: this method is deprecated in favour of `Styler.hide(axis="index")`  
copy().sort_values(by='Enclosing_Area_1').style.hide_index()
```

Out[92]:	Main_Atlas_Number	Main_Display_Name	Enclosing_Area_1	Location_X	Location_Y	Main_X	Main_Y
1127	1155	Penycloddiau, Denbighshire (Pen y Cloddiau)		21.0	-367969	7019842	312888 367647
621	643	Dodman Castle, Cornwall (Dodman Point; The Dod...		21.0	-534770	6485285	200100 39850
731	753	Norbury Camp, Northleach, Gloucestershire		22.0	-202278	6770873	412700 215500
1889	1997	Wooltack Point, Pembrokeshire (Deer Park Fort)		22.0	-584307	6752378	175770 209050
3403	3595	Hod Hill, Dorset		22.0	-245476	6602754	385670 110648
735	757	Salmonsbury Camp, Gloucestershire		23.0	-194654	6779602	417400 220900
357	367	Woodbury, Great Witley, Worcestershire (Woodbu...		23.0	-263690	6850593	374939 264523
136	139	Bozedown Camp, Oxfordshire (Binditch)		23.5	-119597	6710127	464350 178250
3552	3749	Cissbury Ring, West Sussex (Cissbury Camp)		24.0	-42657	6596650	513886 108026
1437	1504	Roulston Scar, North Yorkshire (Sutton Bank; C...		24.5	-134884	7213231	451490 481520
388	404	Ogbury Camp, Wiltshire		26.0	-200007	6646748	414320 138294
443	461	Tedbury Camp, Somerset		26.0	-263616	6663457	374400 148800
373	389	Casterley Camp, Wiltshire (Catterley Banks)		27.5	-204306	6671153	411584 153560
734	756	Willersey Camp, Gloucestershire (Willersey Hil...		28.0	-203808	6807893	411700 238300
410	427	Ebsbury Hill, Wiltshire (Grovely Earthworks)		28.0	-212997	6642126	406160 135380
1234	1276	y Breiddin, Powys (Breiddin Hillfort; The Breid...		28.0	-338884	6931868	329570 314408
89	91	Titterstone Clee, Shropshire		29.6	-289034	6872454	359516 277973
3598	3795	Butser Hill, Hampshire		30.0	-109375	6617356	471527 120313
446	464	Wadbury Camp, Somerset (Wadbury Hillfort)		30.0	-265052	6663609	373500 148900
166	173	Abingdon, The Vineyard, Oxfordshire		33.0	-142388	6740992	449950 197250
95	97	Walbury Camp, West Berkshire		33.0	-162994	6684152	437408 161800
3271	3459	Countisbury Castle, Devon (Shoulsbury; Wind Hill)		35.0	-423647	6662041	274024 149399
3625	3823	Homestall Wood, Kent		35.0	115292	6672762	611760 158930
165	172	Dyke Hills, Oxfordshire (Dike Hills)		46.0	-130542	6735058	457350 193650
738	760	Nottingham Hill Camp, Gloucestershire		48.6	-225551	6791821	398300 228400
3577	3774	Oldbury Camp, Kent		51.5	29679	6671658	558165 156303
751	773	Borough Hill 1, Northamptonshire (Borough Hill)		52.0	-126771	6847138	458877 262696
194	201	Mull of Galloway, Dumfries & Galloway		54.0	-542988	7291829	214381 530735
70	71	Llanymynech Hill, Powys		57.0	-344171	6944572	326479 322138
3402	3594	Hengistbury Head, Dorset		80.0	-195572	6571275	417263 90784
430	448	Ham Hill, Somerset (Hamdon Hill Camp)		84.0	-304545	6611780	348409 116570
3390	3582	Bindon Hill, Dorset		114.0	-248650	6554366	383568 80077

In [93]: `download(['twenty_one_ha_and_over_south'], 'twenty_one_ha_and_over_south')`

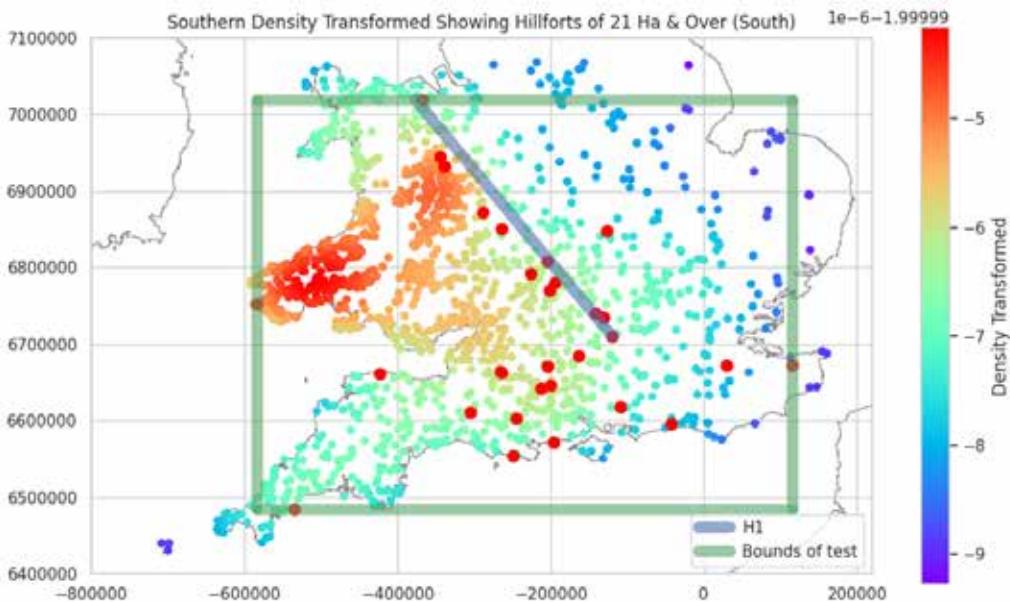
Hypothesis testing

It is beyond the skills of the author to test the hypothesis of the curved line shown in the image above. This being so, the longer straight section from (1155) Penycloddiau, Denbighshire (Pen y Cloddiau) and (139) Bozedown Camp, Oxfordshire (Binditch) will be tested.

Hypothesis (H1): Hillforts of 21 ha and over are aligned between (1155) Penycloddiau, Denbighshire (Pen y Cloddiau) and (139) Bozedown Camp, Oxfordshire (Binditch).

Null Hypothesis (H0): Hillforts of 21 ha and over, between (1155) Penycloddiau, Denbighshire (Pen y Cloddiau) and (139) Bozedown Camp, Oxfordshire (Binditch), are not aligned.

In [94]: `_ , _ = \
south_density_scatter_lines(cluster_south, enclosing_area_1_21_s, \
 'Southern Density Transformed Showing Hillforts of 21 Ha & Over (South)', \
 False, False, True, True)
#'Southern Density Transformed Showing Hillforts of 21 Ha & Over (South)'`



Middleton, M. 2024. Hillforts Primer.

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

The line is 242,511 m (242.5 Km) long.

```
In [95]: penycloddiau = (-367969, 7019842)
bozedown = (-119597, 6710127)
dist = math.sqrt( (bozedown[0] - penycloddiau[0])**2 + (bozedown[1] - \
penycloddiau[1])**2 )
print(f"round(dist)} units")
```

397004 units

```
In [96]: penycloddiau_m = (312888, 367647)
bozedown_m = (464350, 178250)
dist_m = math.sqrt( (bozedown_m[0] - penycloddiau_m[0])**2 + (bozedown_m[1] - \
penycloddiau_m[1])**2 )
print(f"round(dist_m)} m")
```

242512 m

```
In [97]: print(f"Line offset from line {round(dist_m/4.85)} m")
```

Line offset from line 50002 m

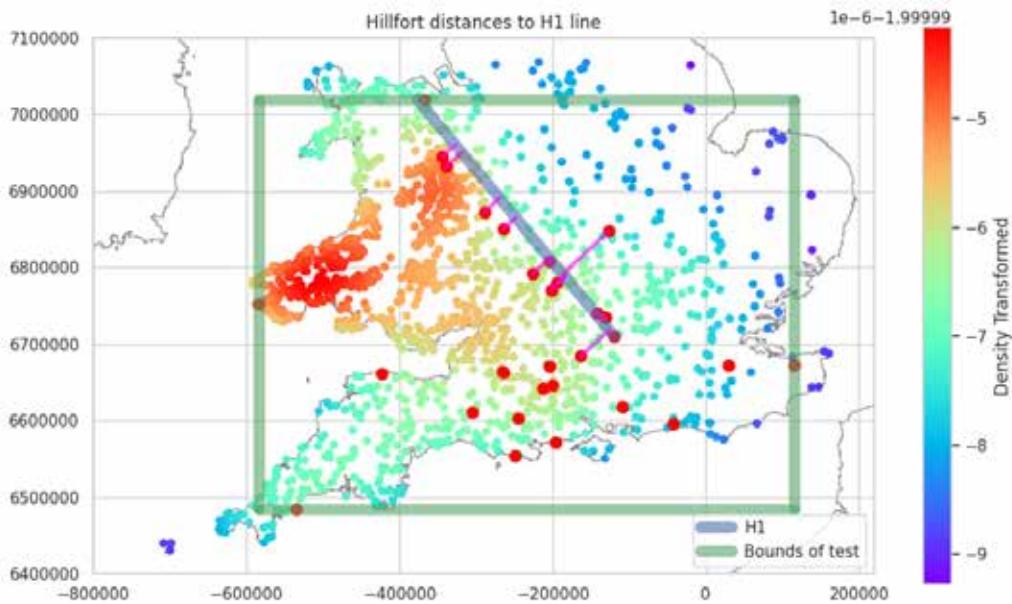
Random Alignments

To test the hypothesis, 1500 random alignments will be created within the bounds of the forts in the South. The test will measure the perpendicular distance from each hillfort to the line, within a buffer of 50 Km. These distances will be used to calculate the Mean Squared Error (MSE) for all forts falling within the limits of each test alignment. The alignments will all be the same length as the hypothesis line (H1). The perpendicular distance from the line will be classified as the error. The error, for each fort of 21 Ha or above, will be squared and the sum of the squared values, for each line, will be divided by the number of forts that fall within the length of that line. 1500 random alignments will be created but only those with five or more hillforts, within the length of the line will be retained to ensure that the result of the MSE is based on a group of hillforts and not just a couple of individuals. 1500 was chosen to ensure that, at least, 1000 alignments remain after the lines with less than five hillforts have been removed. A single example of a random line is shown below with the perpendicular distances to the line shown.

The resulting histogram for all 1000+ alignments will be a normal curve with the most common MSE toward the centre and the least toward the edges. For the null hypothesis to be rejected, the test alignment (H1 - the blue line) would be expected to be to the far left of the curve with a probability of less than 5%.

Shown below are the hillforts perpendicular to the H1 line within an offset distance of 50 km.

```
In [98]: h1_distance_list, _ = \
south_density_scatter_lines(cluster_south, greater_than_21ha_south, \
'Hillfort distances to H1 line', False, False, \
True, True, True, 0, False, True)
```



Middleton, M. 2024. Hillforts Primer.

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

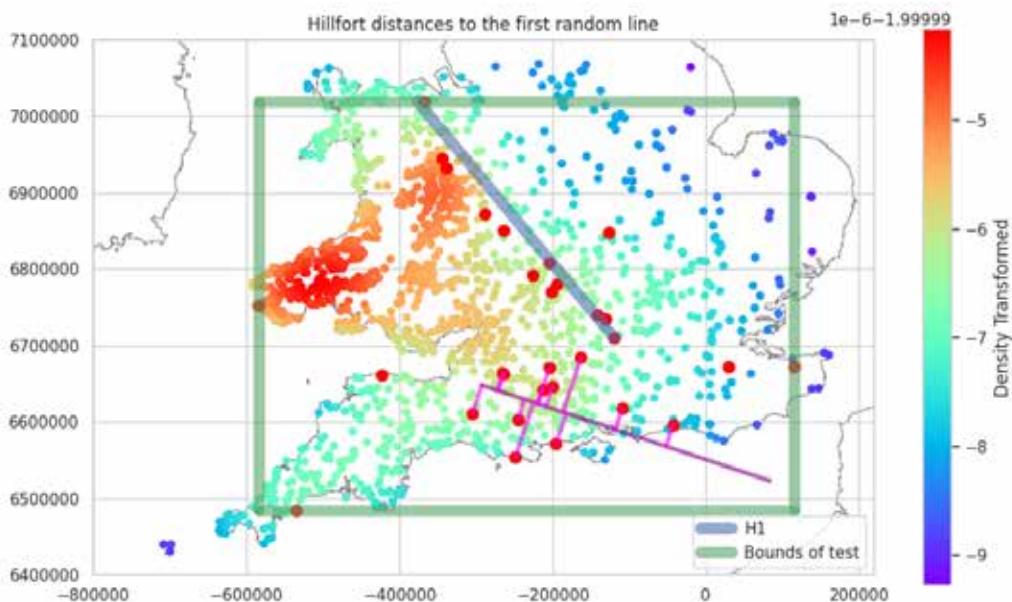
The H1 line has 14 hillforts, withing 50 km, along its length.

```
In [99]: len(h1_distance_list)
```

```
Out[99]: 14
```

A random seed is used to ensure that the random numbers created are the same for each run of this document. Below, hillforts perpendicular to the first random alignment are shown.

```
In [100...], _ = \
south_density_scatter_lines(cluster_south, enclosing_area_1_21_s, \
    'Hillfort distances to the first random line', \
    False, False, True, True, True, 1, True)
```

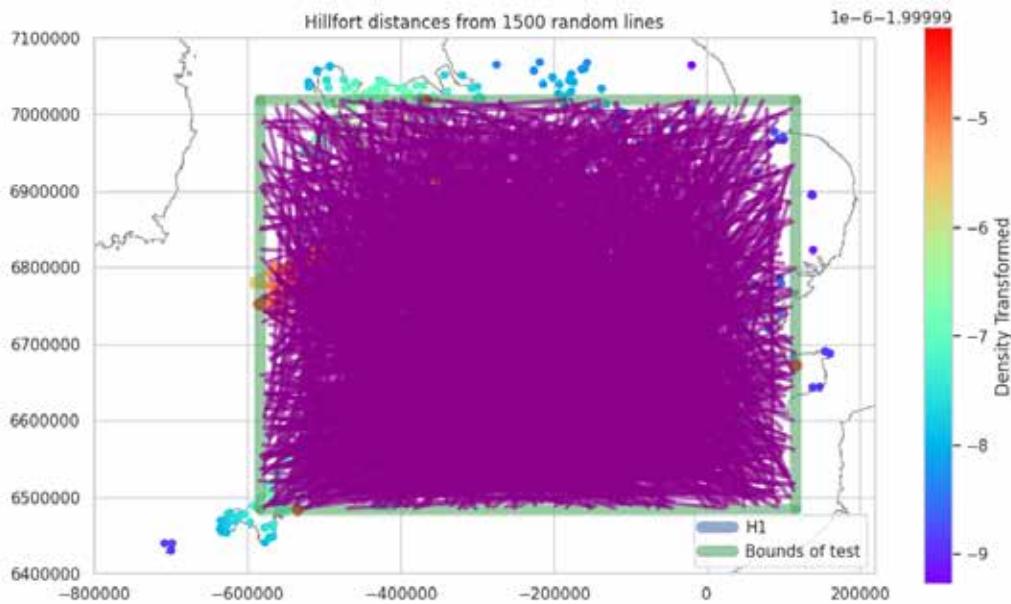


Middleton, M. 2024. Hillforts Primer.

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Next, 1500 random alignments are created within the bounds. The distance to hillforts perpendicular to each line are returned for analysis below.

```
In [101...], distance_lists = \
south_density_scatter_lines(cluster_south, enclosing_area_1_21_s, \
    'Hillfort distances from 1500 random lines', \
    False, False, True, True, True, 1500)
```



Middleton, M. 2024. Hillforts Primer.

Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

A list of the number of hillforts in each list is created.

```
In [102]: length_list = pd.DataFrame([len(x) for x in distance_lists], columns=["count"])
length_list.head()
```

```
Out[102]: count
0    6
1    9
2   15
3   13
4    9
```

There H1 alignment has 14 hillforts along its length. The 1,000 alignments have alignments with between 0 and 19 hillforts along their lengths. For this analysis we will exclude alignments with less than 5 hillforts along their length to reduce the potential for low hillfort counts skewing the results. We want the mean values to be the mean of groups of hillforts not just isolated individuals. For instance, a single hillfort laying close to the line will give a very low MSE while an alternative alignment with a single hillfort far from the line would give a very high MSE. The influence of individual values is reduced, the larger the sample.

```
In [103]: length_list.value_counts().sort_index()
```

```
Out[103]: count
0      22
1      62
2      75
3     101
4      91
5     100
6     104
7      85
8      92
9     107
10     116
11     114
12     139
13     132
14      69
15      45
16      28
17      10
18       4
19       4
dtype: int64
```

The filtered list contains 1149 alignments where there are at least 5 or more hillforts perpendicular to the line along the length of the line.

```
In [104]: filtered_length_list = [x for x in distance_lists if len(x) >= 5]
len(filtered_length_list)
```

```
Out[104]: 1149
```

First we get the MSE for the H1 line.

```
In [105... def get_mean_sum_of_squares(lst):
    sum_list = 0
    for i in lst:
        sum_list += i**2
    mse = sum_list/len(lst)
    return mse
```

```
In [106... mse_h1 = int(get_mean_sum_of_squares(h1_distance_list))
mse_h1
```

```
Out[106]: 959477730
```

Then we get the MSE for the filtered lines.

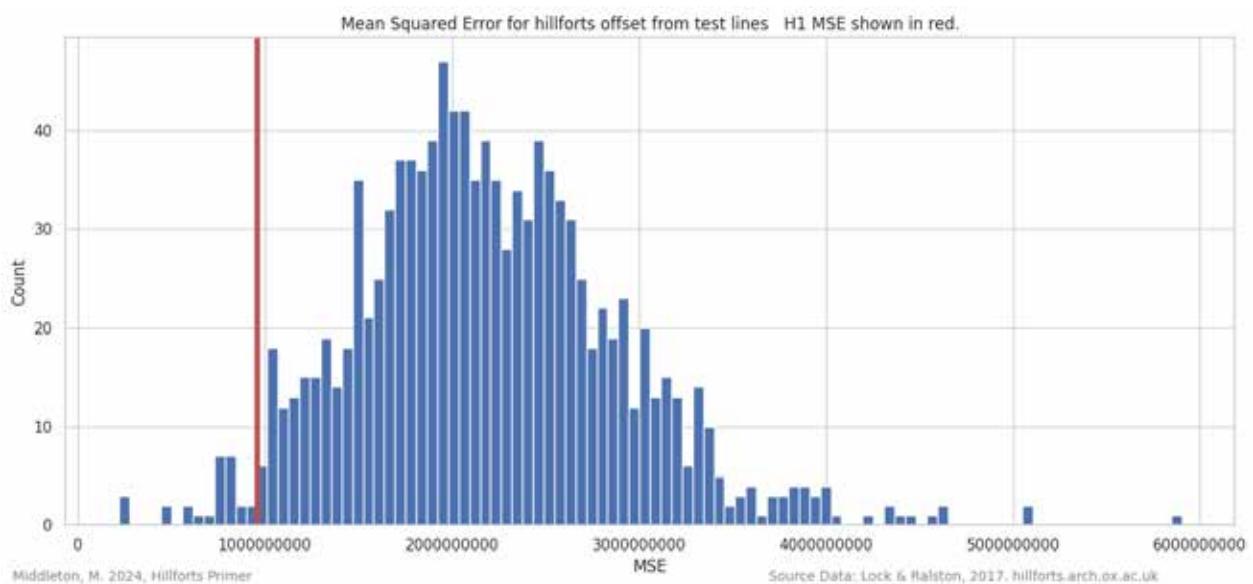
```
In [107... mse_list = [int(get_mean_sum_of_squares(x)) for x in filtered_length_list]
mse_df = pd.DataFrame({'mse':mse_list})
mse_df.head()
```

```
Out[107]:      mse
0   1329023715
1   1832084194
2   3014799598
3   2769225228
4   1299407983
```

Mean Squared Error (offset)

With these results we can plot a histogram of the MSE data. As anticipated the random alignments have produced (more-or-less) a normal curve. The H1 alignment is toward the far left of the curve.

```
In [108... plot_mse_histogram(mse_list, mse_h1)
```

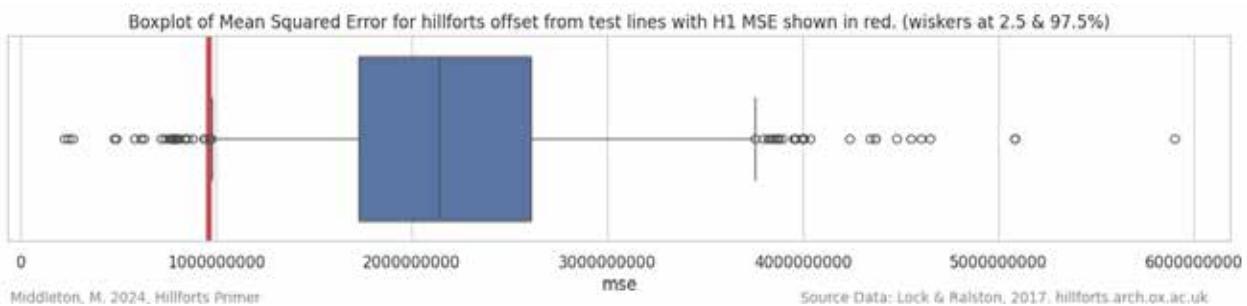


```
In [109... mse_h1
```

```
Out[109]: 959477730
```

A boxplot of the same data shows the interquartile range (IQR) (the blue box) and the first and fourth quartiles bounded by the whiskers. The whiskers, on either side of the IQR are located at 2.5% and 97.5%.

```
In [110... offset_range =
plot_data_range_95(mse_list, "mse", "Boxplot of Mean Squared Error for hillforts offset from test lines with H1 MSE
#"mse", "BoxPlot of Mean Squared Error for hillforts offset from test lines with H1 MSE shown in red. (wiskers at 2.
```



```
In [111]: print(f"H1 MSE = {round(mse_h1/(offset_range[0]/2.5),2)}%")
H1 MSE = 2.46%
```

The H1 line (red) has a probability of 2.46%, just under 2.5%. Below 2.5% or over 97.5% are routinely considered the values above and below which the null hypothesis can be rejected.

Hillforts of 21 Ha and over, falling within 50 km of a straight line between (1155) Penycloddiau, Denbighshire (Pen y Cloddiau) and (139) Bozedown Camp, Oxfordshire (Binditch), are aligned.

Save Figure List

```
In [112]: if save_images:
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")
    fig_list.to_csv(path, index=False)
```

Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Appendix 2

Mike Middleton

<https://orcid.org/0000-0001-5813-6347>

Version 1.0, March 2024.

This research was begun in March 2022.

Part 1: Name, Admin & Location Data

[Colab Notebook: Live code](#) (Must be logged into Google. Select [Google Colaboratory](#), at the top of the screen, if page opens as raw code)

[HTML: Read only](#)

Part 2: Management & Landscape

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 3: Boundary & Dating

[Colab Notebook: Live code](#)

[HTML: Read only](#)

- Boundary Data
- Dating Data

Part 4: Investigations & Interior

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Part 5: Entrance, Enclosing & Annex

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 1: Hypotheses Testing the Alignment of Hillforts with an Area of 21 Hectares or More

[Colab Notebook: Live code](#)

[HTML: Read only](#)

Appendix 2: Classification Northwest

[Colab Notebook: Live code](#)

[HTML: Read only](#)

User Settings

Pre-processed data and images are available for download (without the need to run the code in these files) here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

```
In [1]: confirmed_only = False
In [2]: download_data = False
In [3]: save_images = False
In [4]: show_topography = False
In [5]: # Percentage above which Appendix 02 plots will display
# minimum is 10%
show_percentage = 20
```

Bypass Code Setup

The initial sections of all the Hillforts Primer documents set up the coding environment and define functions used to plot, reprocess and save the data. If you would like to bypass the setup, please use the following link:

Go to [Review Data Appendix 2](#).

Reload Data and Python Functions

Source Data

The Atlas of Hillforts of Britain and Ireland data is made available under the licence, Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). This allows for redistribution, sharing and transformation of the data, as long as the results are credited and made available under the same licence conditions.

The data was downloaded from The Atlas of Hillforts of Britain and Ireland website as a csv file (comma separated values) and saved onto the author's GitHub repository thus enabling the data to be used by this document.

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: <https://hillforts.arch.ox.ac.uk>

Rest services: https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer

Licence: <https://creativecommons.org/licenses/by-sa/4.0/>

Help: <https://hillforts.arch.ox.ac.uk/assets/help.pdf>

Data Structure: <https://maps.arch.ox.ac.uk/assets/data.html>

Hillforts: Britain, Ireland and the Nearer Continent (Sample):

<https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C69/9781789692266-sample.pdf>

Map outlines made with Natural Earth. Free vector and raster map data @ naturalearthdata.com.

Python Modules and Code Setup

```
In [6]: import sys
print(f'Python: {sys.version}')

import sklearn
print(f'Sci kit-Learn: {sklearn.__version__}')

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
```

```

print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

import os
import collections
import math
import random
import PIL
import urllib
random.seed(42) # A random seed is used to ensure that the random numbers
# created are the same for each run of this document.

from slugify import slugify

# Import Google colab tools to access Drive
from google.colab import drive

```

Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
Sci kit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.25.2
matplotlib: 3.7.1
seaborn: 0.13.1
scipy: 1.11.4

Ref: <https://www.python.org/>
Ref: <https://scikit-learn.org/stable/>
Ref: <https://pandas.pydata.org/docs/>
Ref: <https://numpy.org/doc/stable/>
Ref: <https://matplotlib.org/>
Ref: <https://seaborn.pydata.org/>
Ref: <https://docs.scipy.org/doc/scipy/index.html>
Ref: <https://pypi.org/project/python-slugify/>

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

```
In [7]: def show_records(plt, plot_data):
    text_colour = 'k'
    if show_topography == True:
        text_colour = 'w'
    plt.annotate(str(len(plot_data))+' records', xy=(-1180000, 6420000), \
                xycoords='data', ha='left', color=text_colour)
```

```
In [8]: def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                      "hillforts-topo-north.png",
                      "hillforts-topo-northwest-plus.png",
                      "hillforts-topo-northwest-minus.png",
                      "hillforts-topo-northeast.png",
                      "hillforts-topo-south.png",
                      "hillforts-topo-south-plus.png",
                      "hillforts-topo-ireland.png",
                      "hillforts-topo-ireland-north.png",
                      "hillforts-topo-ireland-south.png"]
    else:
        backgrounds = ["hillforts-outline-01.png",
                      "hillforts-outline-north.png",
                      "hillforts-outline-northwest-plus.png",
                      "hillforts-outline-northwest-minus.png"]
```

```

    "hillforts-outline-northeast.png",
    "hillforts-outline-south.png",
    "hillforts-outline-south-plus.png",
    "hillforts-outline-ireland.png",
    "hillforts-outline-ireland-north.png",
    "hillforts-outline-ireland-south.png"]
return backgrounds

```

```
In [9]: def get_bounds():
    bounds = [[-1200000, 220000, 6400000, 8700000],
              [-1200000, 220000, 7000000, 8700000],
              [-1200000, -480000, 7000000, 8200000],
              [-900000, -480000, 7100000, 8200000],
              [-520000, 0, 7000000, 8700000],
              [-800000, 220000, 6400000, 7100000],
              [-1200000, 220000, 6400000, 7100000],
              [-1200000, -600000, 6650000, 7450000],
              [-1200000, -600000, 7050000, 7450000],
              [-1200000, -600000, 6650000, 7080000]]
    return bounds
```

```
In [10]: def show_background(plt, ax, location=""):
    backgrounds = get_backgrounds()
    bounds = get_bounds()
    folder = "https://raw.githubusercontent.com/MikeDaiRSI/Hillforts-Primer/main/hillforts-topo/"
    # "https://raw.githubusercontent.com/MikeDaiRSI/Hillforts-Primer/main/hillforts-topo/"

    if location == "n":
        background = os.path.join(folder, backgrounds[1])
        bounds = bounds[1]
    elif location == "nw+":
        background = os.path.join(folder, backgrounds[2])
        bounds = bounds[2]
    elif location == "nw-":
        background = os.path.join(folder, backgrounds[3])
        bounds = bounds[3]
    elif location == "ne":
        background = os.path.join(folder, backgrounds[4])
        bounds = bounds[4]
    elif location == "s":
        background = os.path.join(folder, backgrounds[5])
        bounds = bounds[5]
    elif location == "s+":
        background = os.path.join(folder, backgrounds[6])
        bounds = bounds[6]
    elif location == "i":
        background = os.path.join(folder, backgrounds[7])
        bounds = bounds[7]
    elif location == "in":
        background = os.path.join(folder, backgrounds[8])
        bounds = bounds[8]
    elif location == "is":
        background = os.path.join(folder, backgrounds[9])
        bounds = bounds[9]
    else:
        background = os.path.join(folder, backgrounds[0])
        bounds = bounds[0]

    img = np.array(PIL.Image.open(urllib.request.urlopen(background)))
    ax.imshow(img, extent=bounds)
```

```
In [11]: def get_counts(data):
    data_counts = []
    for col in data.columns:
        count = len(data[data[col] == 'Yes'])
        data_counts.append(count)
    return data_counts
```

```
In [12]: def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
               color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
               horizontalalignment='left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(0.99, 0.01), \
               xycoords='figure fraction', horizontalalignment='right')
```

```
In [13]: def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
               color='grey', xy=(0.01, 0.035), xycoords='figure fraction', \
               horizontalalignment='left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
               size='small', color='grey', xy=(0.99, 0.035), \
               xycoords='figure fraction', horizontalalignment='right')
```

```
In [14]: def plot_bar_chart(data, split_pos, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = data.columns
    x_data = [x.split('_')[split_pos:] for x in x_data]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])
    y_data = get_counts(data)
    ax.bar(x_data_new, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [15]: def plot_bar_chart_using_two_tables(x_data, y_data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [16]: def plot_bar_chart_numeric(data, split_pos, x_label, y_label, title, n_bins):
    new_data = data.copy()
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    data[x_label].plot(kind='hist', bins=n_bins)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [17]: def plot_bar_chart_value_counts(data, x_label, y_label, title):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    df = data.value_counts()
    x_data = df.index.values
    y_data = df.values
    ax.bar(x_data, y_data)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [18]: def get_bins(data, bins_count):
    data_range = data.max() - data.min()
    print(bins_count)
    if bins_count != None:
        x_bins = [x for x in range(data.min(), data.max(), bins_count)]
        n_bins = len(x_bins)
    else:
        n_bins = int(data_range)
        if n_bins < 10:
            multi = 10
            while n_bins < 10:
                multi *= 10
                n_bins = int(data_range * multi)
        elif n_bins > 100:
            n_bins = int(data_range)/10
    return n_bins
```

```
In [19]: def plot_histogram(data, x_label, title, bins_count = None):
    n_bins = get_bins(data, bins_count)
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(x_label)
    ax.set_ylabel('Count')
    plt.title(label_format(style='plain'))
```

```

plt.hist(data, bins=n_bins)
plt.title(get_print_title(title))
add_annotation_plot(ax)
save_fig(title)
plt.show()

```

```

In [20]: def plot_continuous(data, xlabel, title):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(xlabel)
    plt.plot(data, linewidth=4)
    plt.title_format(style='plain')
    plt.title(get_print_title(title))
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

```

In [21]: # box plot
from matplotlib import cbook
def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_axes([0, 0, 1, 1])
    ax.set_xlabel(feature)
    add_annotation_plot(ax)
    plt.title(get_print_title(feature + " Range"))
    plt.title_format(style='plain')
    if o == "v":
        sns.boxplot(data=data, orient="v")
    else:
        sns.boxplot(data=data, orient="h")
    save_fig(feature + " Range")
    plt.show()

    bp = boxplot_stats(data)

    low = bp[0].get('whislo')
    q1 = bp[0].get('q1')
    median = bp[0].get('med')
    q3 = bp[0].get('q3')
    high = bp[0].get('whishi')

    return [low, q1, median, q3, high]

```

```

In [22]: def location_XY_plot():
    plt.title_format(style='plain')
    plt.xlim(-1200000, 220000)
    plt.ylim(6400000, 8700000)
    add_annotation_l_xy(plt)

```

```

In [23]: def add_grey(region=''):
    if show_topography == False:
        # plots all the hillforts as a grey background
        loc = location_data.copy()
        if region == 's':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -710000].copy()
        elif region == 'ne':
            loc = loc[loc['Location_Y'] < 8000000].copy()
            loc = loc[loc['Location_X'] > -800000].copy()

    plt.scatter(loc['Location_X'], loc['Location_Y'], c='Silver')

```

```

In [24]: def plot_over_grey_numeric(merged_data, a_type, title, extra="", inner=False, \
                           fringe=False, oxford=False, swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford, swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_lLegend = add_21Ha_fringe()
        patches.append(f_for_lLegend)
    if inner:
        i_for_lLegend = add_21Ha_line()
        patches.append(i_for_lLegend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles=patches)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```
In [25]: def plot_over_grey_boundary(merged_data, a_type, boundary_type):
    plot_data = merged_data[merged_data[a_type] == boundary_type]
    fig, ax = plt.subplots(figsize=(9.47, 15.33))
    show_background(plt, ax)
    location_XY_plot()
    add_grey(region=' ')
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    show_records(plt, plot_data)
    plt.title(get_print_title('Boundary_Type: ' + boundary_type))
    save_fig('Boundary_Type_' + boundary_type)
    plt.show()
    print(f' {round((len(plot_data)/len(merged_data)*100), 2)}%')
```

```
In [26]: def plot_densiti_over_grey(data, data_type):
    new_data = data.copy()
    new_data = new_data.drop(['Density'], axis=1)
    new_data = add_densiti(new_data)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
               c=new_data['Density'], cmap=cm.rainbow, s=25)
    plt.colorbar(label='Density')
    plt.title(get_print_title(f'Density - {data_type}'))
    save_fig(f'Density_{data_type}')
    plt.show()
```

```
In [27]: def add_21Ha_line():
    x_val ues = \
        [-367969, -344171, -263690, -194654, -130542, -119597, -162994, -265052], \
        [-304545]
    y_val ues = \
        [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 6663609], \
        [6611780]
    plt.plot(x_val ues, y_val ues, 'k', ls='-', lw=15, alpha=0.25, \
              label = '≥ 21 Ha Line')
    add_to_l egend = \
        Line2D([0], [0], color='k', lw=15, alpha=0.25, label = '≥ 21 Ha Line')
    return add_to_l egend
```

```
In [28]: def add_21Ha_fringe():
    x_val ues = \
        [-367969, -126771, 29679, -42657, -248650, -304545, -423647, -584307, -367969]
    y_val ues = \
        [7019842, 6847138, 6671658, 6596650, 6554366, 6611780, 6662041, 6752378, 7019842]
    plt.plot(x_val ues, y_val ues, 'k', ls=':', lw=5, alpha=0.45, \
              label = '≥ 21 Ha Fringe')
    add_to_l egend = \
        Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, label = '≥ 21 Ha Fringe')
    return add_to_l egend
```

```
In [29]: def add_oxford_swi_ndon(oxford=False, swi_ndon=False):
    # plots a circle over Swindon & Oxford
    radius = 50
    marker_si ze = (2*radius)**2
    patches = []
    if oxford:
        plt.scatter(-144362, 6758380, c='dodgerblue', s=marker_si ze, alpha=0.50)
        b_patch = mpatches.Patch(color='dodgerblue', label='Oxford orbit')
        patches.append(b_patch)
    if swi_ndon:
        plt.scatter(-197416, 6721977, c='yellow', s=marker_si ze, alpha=0.50)
        y_patch = mpatches.Patch(color='yellow', label='Swindon orbit')
        patches.append(y_patch)
    return patches
```

```
In [30]: def plot_over_grey(merged_data, a_type, yes_no, extra="", \
                     inner=False, fringe=False, oxford=False, swi_ndon=False):
    # plots selected data over the grey dots. yes_no controls filtering
    # the data for a positive or negative values.
    plot_data = merged_data[merged_data[a_type] == yes_no]
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swi_ndon(oxford, swi_ndon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
        f_for_l egend = add_21Ha_fringe()
        patches.append(f_for_l egend)
    if inner:
        i_for_l egend = add_21Ha_line()
        patches.append(i_for_l egend)
    show_records(plt, plot_data)
```

```

plot.legend(loc='upper left', handles= patches)
plot.title(get_print_title(f'{a_type} {extra}'))
save_fig(f'{a_type}_{extra}')
plot.show()
print(f' {round((len(plot_data)/len(merged_data)*100), 2)}%')
return plot_data

```

```

In [31]: def plot_type_values(data, data_type, title):
    new_data = data.copy()
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.4, 23.0 * 0.66))
    show_background(plot, ax)
    location_XY_plot()
    plt.scatter(new_data['Location_X'], new_data['Location_Y'], \
                c=new_data[data_type], cmap=cm.rainbow, s=25)
    plt.colorbar(label=data_type)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [32]: def bespoke_plot(plot, title):
    add_annotation_plot(plot)
    plt.title_label_format(style='plain')
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()

```

```

In [33]: def get_proportions(date_set):
    total = sum(date_set) - date_set[-1]
    newset = []
    for entry in date_set[:-1]:
        newset.append(round(entry/total, 2))
    return newset

```

```

In [34]: def plot_dates_by_region(nw, ne, ni, si, s, features):
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_axes([0, 0, 1, 1])
    x_data = nw[features].columns
    x_data = [x.split('_')[2:] for x in x_data][:-1]
    x_data_new = []
    for l in x_data:
        txt = ""
        for part in l:
            txt += "_" + part
        x_data_new.append(txt[1:])

    set1_name = 'NW'
    set2_name = 'NE'
    set3_name = 'N Ireland'
    set4_name = 'S Ireland'
    set5_name = 'South'
    set1 = get_proportions(get_counts(nw[features]))
    set2 = get_proportions(get_counts(ne[features]))
    set3 = get_proportions(get_counts(ni[features]))
    set4 = get_proportions(get_counts(si[features]))
    set5 = get_proportions(get_counts(s[features]))

    X_axis = np.arange(len(x_data_new))

    budge = 0.25

    plt.bar(X_axis - 0.55 + budge, set1, 0.3, label = set1_name)
    plt.bar(X_axis - 0.4 + budge, set2, 0.3, label = set2_name)
    plt.bar(X_axis - 0.25 + budge, set3, 0.3, label = set3_name)
    plt.bar(X_axis - 0.1 + budge, set4, 0.3, label = set4_name)
    plt.bar(X_axis + 0.05 + budge, set5, 0.3, label = set5_name)

    plt.xticks(X_axis, x_data_new)
    plt.xlabel('Dating')
    plt.ylabel('Proportion of Total Dated Hillforts in Region')
    title = 'Proportions of Dated Hillforts by Region'
    plt.title(title)
    plt.legend()
    add_annotation_plot(ax)
    save_fig(title)
    plt.show()

```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

```

In [35]: def test_numeric(data):
    temp_data = data.copy()

```

```

columns = data.columns
out_cols = ['Feature', 'Entries', 'Numeric', 'Non-Numeric', 'Null']
feat, ent, num, non, nul = [], [], [], [], []
for col in columns:
    if temp_data[col].dtype == 'object':
        feat.append(col)
        temp_data[col + '_num'] = temp_data[col].str.isnumeric()
        entries = temp_data[col].notnull().sum()
        true_count = temp_data[col + '_num'][temp_data[col + '_num'] == True].sum()
        null_count = temp_data[col].isna().sum()
        ent.append(entries)
        num.append(true_count)
        non.append(entries - true_count)
        nul.append(null_count)
    else:
        print(f'{col} {temp_data[col].dtype}')
summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
summary.columns = out_cols
return summary

```

```
In [36]: def find_duplicated(numeric_data, text_data, encodeable_data):
    d = False
    all_columns = \
        list(numeric_data.columns) + list(text_data.columns) + \
        list(encodeable_data.columns)
    duplicate = \
        [item for item, count in collections.Counter(all_columns).items() \
         if count > 1]
    if duplicate:
        print(f"There are duplicate features: {duplicate}")
        d = True
    return d
```

```
In [37]: def test_data_split(main_data, numeric_data, text_data, encodeable_data):
    m = False
    split_features = \
        list(numeric_data.columns) + list(text_data.columns) + \
        list(encodeable_data.columns)
    missing = list(set(main_data) - set(split_features))
    if missing:
        print(f"There are missing features: {missing}")
        m = True
    return m
```

```
In [38]: def review_data_split(main_data, numeric_data, text_data, \
                           encodeable_data = pd.DataFrame()):
    d = find_duplicated(numeric_data, text_data, encodeable_data)
    m = test_data_split(main_data, numeric_data, text_data, encodeable_data)
    if d != True and m != True:
        print("Data split good.")
```

```
In [39]: def find_duplicates(data):
    print(f'{data.count() - data.duplicated(keep=False).count()} duplicates.')
```

```
In [40]: def count_yes(data):
    total = 0
    for col in data.columns:
        count = len(data[data[col] == 'Yes'])
        total += count
        print(f'{col}: {count}')
    print(f'Total yes count: {total}')
```

Null Value Functions

The following functions will be used to update null values.

```
In [41]: def fill_nan_with_mins_one(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna(-1)
    return new_data
```

```
In [42]: def fill_nan_with_NA(data, feature):
    new_data = data.copy()
    new_data[feature] = data[feature].fillna("NA")
    return new_data
```

```
In [43]: def test_numeric_value_in_feature(feature, value):
    test = feature.isin([-1]).sum()
    return test
```

```
In [44]: def test_categorical_value_in_feature(dataframe, feature, value):
    test = dataframe[feature][dataframe[feature] == value].count()
    return test

In [45]: def test_cat_list_for_NA(dataframe, cat_list):
    for val in cat_list:
        print(val, test_categorical_value_in_feature(dataframe, val, 'NA'))

In [46]: def test_num_list_for_minus_one(dataframe, num_list):
    for val in num_list:
        feature = dataframe[val]
        print(val, test_numeric_value_in_feature(feature, -1))

In [47]: def update_cat_list_for_NA(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_NA(new_data, val)
    return new_data

In [48]: def update_num_list_for_minus_one(dataframe, cat_list):
    new_data = dataframe.copy()
    for val in cat_list:
        new_data = fill_nan_with_minus_one(new_data, val)
    return new_data
```

Reprocessing Functions

```
In [49]: def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data
```

Save Image Functions

```
In [50]: fig_no = 0
part = 'Appendix02'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"

In [51]: def get_file_name(title):
    file_name = slugify(title)
    return file_name

In [52]: def get_print_title(title):
    title = title.replace("_", " ")
    title = title.replace("-", " ")
    title = title.replace(",", ";")
    return title

In [53]: def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no

In [54]: if save_images == True:
    drive.mount('/content/drive')
    os.getcwd()
else:
    pass
```

```
In [55]: def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Appendix_02_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}.{fig_extension}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
```

```

        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution, bbox_inches='tight')
    else:
        pass

```

Load Data

The source csv file is loaded and the first two rows are displayed to confirm the load was successful. Note that, to the left, an index has been added automatically. This index will be used frequently when splitting and remerging data extracts.

```
In [56]: hillforts_csv = r"https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-atlas-source-data.csv"
#r"https://raw.githubusercontent.com/MikeDairis/Hillforts-Primer/main/hillforts-atlas-source-data-csv/hillforts.csv"
hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
pd.set_option('display.max_columns', None, 'display.max_rows', None)
hillforts_data.head(2)

<ipython-input-56-03f8272c1daf>: 4: DtypeWarning: Columns (10, 12, 68, 83, 84, 85, 86, 165, 183) have mixed types. Specify dtype option on import or set low_memory=False.
    hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

```
Out[56]:   OBJECTID  Main_Atlas_Number  Main_Country_Code  Main_Country  Main_Title_Name  Main_Site_Name  Main_Alt_Name  Main_Display_N
0          1                 1             EN      England  EN0001 Aconbury Camp, Herefordshire  Aconbury Camp  Aconbury Beacon  Aconbury C Hereford (Aconbury Bea
1          2                 2             EN      England  EN0002 Bach Camp, Herefordshire  Bach Camp       NaN  Bach C Hereford
```

Filter confirmed (if selected)

If confirmed_only is set to True in User Settings above, this will filter the source data so that it contains only confirmed forts.

```
In [57]: if confirmed_only == True:
    hillforts_data = \
        hillforts_data[hillforts_data['Status_Interpretation_Reliability'] == \
                      'Confirmed']
    print(f'Data filtered to contain only {len(hillforts_data)} confirmed hillforts.')
else:
    print(f'Using all {len(hillforts_data)} record in the Hillforts Atlas.')
```

Using all 4147 record in the Hillforts Atlas.

Download Function

```
In [58]: from google.colab import files
def download(data_list, filename, hf_data=hillforts_data):
    if download_data == True:
        name_and_number = hf_data[['Main_Atlas_Number', 'Main_Display_Name']].copy()
        dl = name_and_number.copy()
        for pkg in data_list:
            if filename not in ['england', 'wales', 'scotland', \
                               'republic-of-ireland', 'northern-ireland', \
                               'isle-of-man', 'roi-ni', 'eng-wal-sco-iom']:
                if pkg.shape[0] == hillforts_data.shape[0]:
                    dl = pd.merge(dl, pkg, left_index=True, right_index=True)
                else:
                    dl = data_list[0]
            dl = dl.replace('\r', ' ', regex=True)
            dl = dl.replace('\n', ' ', regex=True)
            fn = 'hillforts_primer_' + filename
            fn = get_file_name(fn)
            dl.to_csv(fn+'.csv', index=False)
            files.download(fn+'.csv')
    else:
        pass
```

Reload Name and Number

The Main Atlas Number and the Main Display Name are the primary unique reference identifiers in the data. With these, users can identify any record numerically and by name. Throughout this document, the data will be clipped into a number of sub-data packages. Where needed, these data extracts will be combined with Name and Number features to ensure the data can be understood and can, if needed, be concorded.

```
In [59]: name_and_number_features = ['Main_Atlas_Number', 'Main_Display_Name']
name_and_number = hillforts_data[name_and_number_features].copy()
name_and_number.head()
```

	Main_Atlas_Number	Main_Display_Name
0	1	Aconbury Camp, Herefordshire (Aconbury Beacon)
1	2	Bach Camp, Herefordshire
2	3	Backbury Camp, Herefordshire (Ethelbert's Camp)
3	4	Brandon Camp, Herefordshire
4	5	British Camp, Herefordshire (Herefordshire Bea...

Reload Location

```
In [60]: location_numeric_data_short_features = ['Location_X', 'Location_Y']
location_numeric_data_short = hillforts_data[location_numeric_data_short_features]
location_numeric_data_short = add_density(location_numeric_data_short)
location_numeric_data_short.head()
location_data = location_numeric_data_short.copy()
location_data.head()
```

	Location_X	Location_Y	Density
0	-303295	6798973	1.632859e-12
1	-296646	6843289	1.540172e-12
2	-289837	6808611	1.547729e-12
3	-320850	6862993	1.670548e-12
4	-261765	6810587	1.369981e-12

Reload Location Cluster Data Packages

```
In [61]: cluster_data = \
hillforts_data[['Location_X', 'Location_Y', 'Main_Country_Code']].copy()
cluster_data['Cluster'] = 'NA'
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
'NI', 'I', inplace=True)
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != \
'IR', 'I', inplace=True)

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000) , \
    'North_Ireland', cluster_data['Cluster'])
north_irland = cluster_data[cluster_data['Cluster'] == 'North_Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , \
    'South_Ireland', cluster_data['Cluster'])
south_irland = cluster_data[cluster_data['Cluster'] == 'South_Ireland'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] < 7070000) , \
    'South', cluster_data['Cluster'])
south = cluster_data[cluster_data['Cluster'] == 'South'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] >= -500000), 'Northeast', cluster_data['Cluster'])
north_east = cluster_data[cluster_data['Cluster'] == 'Northeast'].copy()

cluster_data['Cluster'] = np.where(
    (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & \
    (cluster_data['Location_X'] < -500000), 'Northwest', cluster_data['Cluster'])
north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()

temp_cluster_location_packages = \
[north_irland, south_irland, south, north_east, north_west]

cluster_packages = []
```

```

for pkg in temp_cluster_location_packages:
    pkg = pkg.drop(['Main_Country_Code'], axis=1)
    cluster_packages.append(pkg)

north ireland, south ireland, south, north east, north west = \
cluster_packages[0], cluster_packages[1], cluster_packages[2], \
cluster_packages[3], cluster_packages[4]

```

Classification Northwest

Reload density function

```
In [62]: def renew_density(data):
    new_data = data.copy()
    try:
        new_data = new_data.drop(['Density'], axis=1)
    except:
        pass
    try:
        new_data = new_data.drop(['Density_trans'], axis=1)
    except:
        pass
    xy = np.vstack([new_data['Location_X'], new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    new_data['Density_trans'] = stats.boxcox(new_data['Density'], 0.5)
    return new_data
```

Refresh density values

```
In [63]: north_west = renew_density(north_west)
```

Northwest plot functions

```
In [64]: def plot_north_west_density_minus(show_elidon, cluster_north_west, \
                                         location_X_cluster_north_west, \
                                         location_Y_cluster_north_west):
    fig, ax = plt.subplots(figsize=((2.772 * 1.75)+2.0, (7.26 * 1.75)))
    show_background(plt, ax, 'nw-')
    plt.title_format(style='plain')
    plt.xlim(-900000, -480000)
    plt.ylim(7100000, 8200000)
    plt.scatter(cluster_north_west['Location_X'], \
                cluster_north_west['Location_Y'], \
                c=cluster_north_west['Density_trans'], cmap=cm.rainbow, s=25)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_north_west, \
                                                   location_Y_cluster_north_west, \
                                                   3, 1.5)
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', \
               ls='--', lw=1.5)
    plt.plot(-609918, 7575512, 'ko')
    if show_elidon:
        ax.annotate('SC2466: Dunadd', xy=(-609918, 7575512), xycoords='data', \
                    ha='left', xytext=(-110, -135), textcoords='offset points', \
                    arrowprops=dict(arrowstyle=">", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Trans & Boxplot (NW minus Ireland)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

```
In [65]: def plot_data_range_plus(data, feature, o=None):
    if o == None:
        pass
    else:
        fig = plt.figure(figsize=(12, 8))
        ax = fig.add_axes([0, 0, 1, 1])
        ax.set_xlabel(feature)
        add_annotation_plot(ax)
        plt.title(get_print_title(feature + " Range"))
        plt.title_format(style='plain')
```

```

if o == "v":
    sns.boxplot(data=data, orient="v")
elif o == None:
    pass
else:
    sns.boxplot(data=data, orient="h")

if o == None:
    pass
else:
    save_fig(feature + " Range")
plt.show()

bp = boxplot_stats(data)

low = bp[0].get('whislo')
q1 = bp[0].get('q1')
median = bp[0].get('med')
q3 = bp[0].get('q3')
high = bp[0].get('whishi')

return [low, q1, median, q3, high]

```

```

In [66]: def get_rectangles(x_data, y_data, w1, w2):
    x_lo, x_q1, x_median, x_q3, x_hi = x_data[0], x_data[1], x_data[2], \
    x_data[3], x_data[4]
    y_lo, y_q1, y_median, y_q3, y_hi = y_data[0], y_data[1], y_data[2], \
    y_data[3], y_data[4]

    rect = patches.Rectangle((x_lo, y_lo), x_hi-x_lo, y_hi-y_lo, linewidth=w2, \
                            edgecolor='k', facecolor='none')
    rect2 = patches.Rectangle((x_q1, y_q1), x_q3-x_q1, y_q3-y_q1, linewidth=w1, \
                            edgecolor='r', facecolor='none')

    return rect, rect2, [y_q1, x_median, y_q3], [x_q1, y_median, x_q3]

```

```

In [67]: def plot_north_west_densisty_for_column(show_eildon, regional_data, cluster, \
                                             column, location_X_cluster, \
                                             location_Y_cluster, show_boxes = False):
    fig, ax = plt.subplots(figsize=((2.772 * 1.75)+2.0, (7.26 * 1.75)))
    show_background(plt, ax, 'nw-')
    plt.title('Transformed Density')
    plt.xlim(-900000, -480000)
    plt.ylim(7100000, 8200000)
    plt.scatter(cluster['Location_X'], \
                cluster['Location_Y'], \
                c=cluster['Density_trans'], cmap=cm.rainbow, s=25)
    if show_boxes:
        rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster, \
                                                       location_Y_cluster, \
                                                       3, 1.5)
        ax.add_patch(rect3)
        ax.add_patch(rect4)
        plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], \
                   colors='purple', ls='--', lw=1.5)
        plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], \
                   colors='purple', ls='--', lw=1.5)
        plt.plot(-609918, 7575512, 'ko')
    if show_eildon:
        ax.annotate('SC2466: Dunadd', xy=(-609918, 7575512), xycoords='data', \
                    ha='left', xytext=(-110, -135), textcoords='offset points', \
                    arrowprops=dict(arrowstyle=">", color='k', lw=1, \
                                    connectionstyle="arc3,rad=-0.2"))
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = column
    text_colour = 'k'
    plt.annotate(str(round((len(cluster) / len(regional_data)) * 100, 1))+ "%", \
                 xy=(-880000, 7150000), xycoords='data', ha='left', \
                 color=text_colour)
    plt.annotate(str(len(cluster))+' / '+str(len(regional_data)), \
                 xy=(-880000, 7120000), xycoords='data', ha='left', \
                 color=text_colour)
    plt.title(get_print_title(title))
    save_fig("NW_" + title)
    plt.show()

```

```

In [68]: def plot_encodeable(hillforts_data, north_west, encodeable_features, show_percentage):
    encodeable_data = hillforts_data[encodeable_features].copy()
    location_data = pd.merge(north_west, encodeable_data, \
                             left_index=True, right_index=True)
    print(len(location_data))
    for column in encodeable_features:
        # Filter data

```

```

cluster = location_data[location_data[column] == "Yes"]
show_eldon = False
if show_percentage < 10:
    show_percentage = 10

if (len(cluster) / len(location_data)) * 100 > show_percentage:
    # refresh density
    cluster = renew_density(cluster)
    location_X_cluster = \
        plot_data_range_plus(cluster['Location_X'], '', None)
    location_Y_cluster = \
        plot_data_range_plus(cluster['Location_Y'], '', None)
    #print(len(cluster))
    plot_north_west_density_for_column(show_eldon, location_data, \
        cluster, column, location_X_cluster, \
        location_Y_cluster, False)

```

```

In [69]: def plot_altitude(hillforts_data, north_west, single_data_set, single_column, altitude, show_percentage):
    # Filter data
    location_data = pd.merge(north_west, single_data_set, \
        left_index=True, right_index=True)
    print(len(location_data))
    new_col_name = "Landscape_Altitude" + "_" + altitude
    location_data.columns = ["Location_X", "Location_Y", "Cluster", "Density", "Density_trans", "Location_X_y", "Location_Y_y"]
    cluster = location_data[location_data[new_col_name] == "Yes"]
    show_eldon = False
    if show_percentage < 10:
        show_percentage = 10

    if (len(cluster) / len(north_west)) * 100 > show_percentage:
        # refresh density
        cluster = renew_density(cluster)
        location_X_cluster = \
            plot_data_range_plus(cluster['Location_X'], '', None)
        location_Y_cluster = \
            plot_data_range_plus(cluster['Location_Y'], '', None)
        #print(len(cluster))
        plot_north_west_density_for_column(show_eldon, north_west, \
            cluster, new_col_name, location_X_cluster, \
            location_Y_cluster, False)

```

```

In [70]: def plot_numeric(hillforts_data, north_west, numeric_features, show_percentage):
    numeric_data = hillforts_data[numeric_features].copy()
    location_data = pd.merge(north_west, numeric_data, \
        left_index=True, right_index=True)
    print(len(location_data))
    for column in numeric_features:
        unique_values = sorted(location_data[column].dropna().unique())
        print(unique_values)
        for val in unique_values:
            # Filter data
            cluster = location_data[location_data[column] == val]
            new_col_name = column + "_" + str(val)
            cluster = cluster.rename(columns={column: new_col_name})
            show_eldon = False
            if show_percentage < 10:
                show_percentage = 10

            if (len(cluster) / len(location_data)) * 100 > show_percentage:
                # refresh density
                cluster = renew_density(cluster)
                location_X_cluster = \
                    plot_data_range_plus(cluster['Location_X'], '', None)
                location_Y_cluster = \
                    plot_data_range_plus(cluster['Location_Y'], '', None)
                #print(len(cluster))
                plot_north_west_density_for_column(show_eldon, location_data, \
                    cluster, new_col_name, location_X_cluster, \
                    location_Y_cluster, False)

```

```

In [71]: cluster_north_west = add_density(north_west)
cluster_north_west['Density_trans'] = \
stats.boxcox(cluster_north_west['Density'], 0.5)

```

```

In [72]: location_X_cluster_north_west = \
plot_data_range_plus(north_west['Location_X'], \
    'Location_X - Northwest (minus Ireland)', None)

```

```

In [73]: location_Y_cluster_north_west = \
plot_data_range_plus(north_west['Location_Y'], \
    'Location_Y - Northwest (minus Ireland)', None)

```

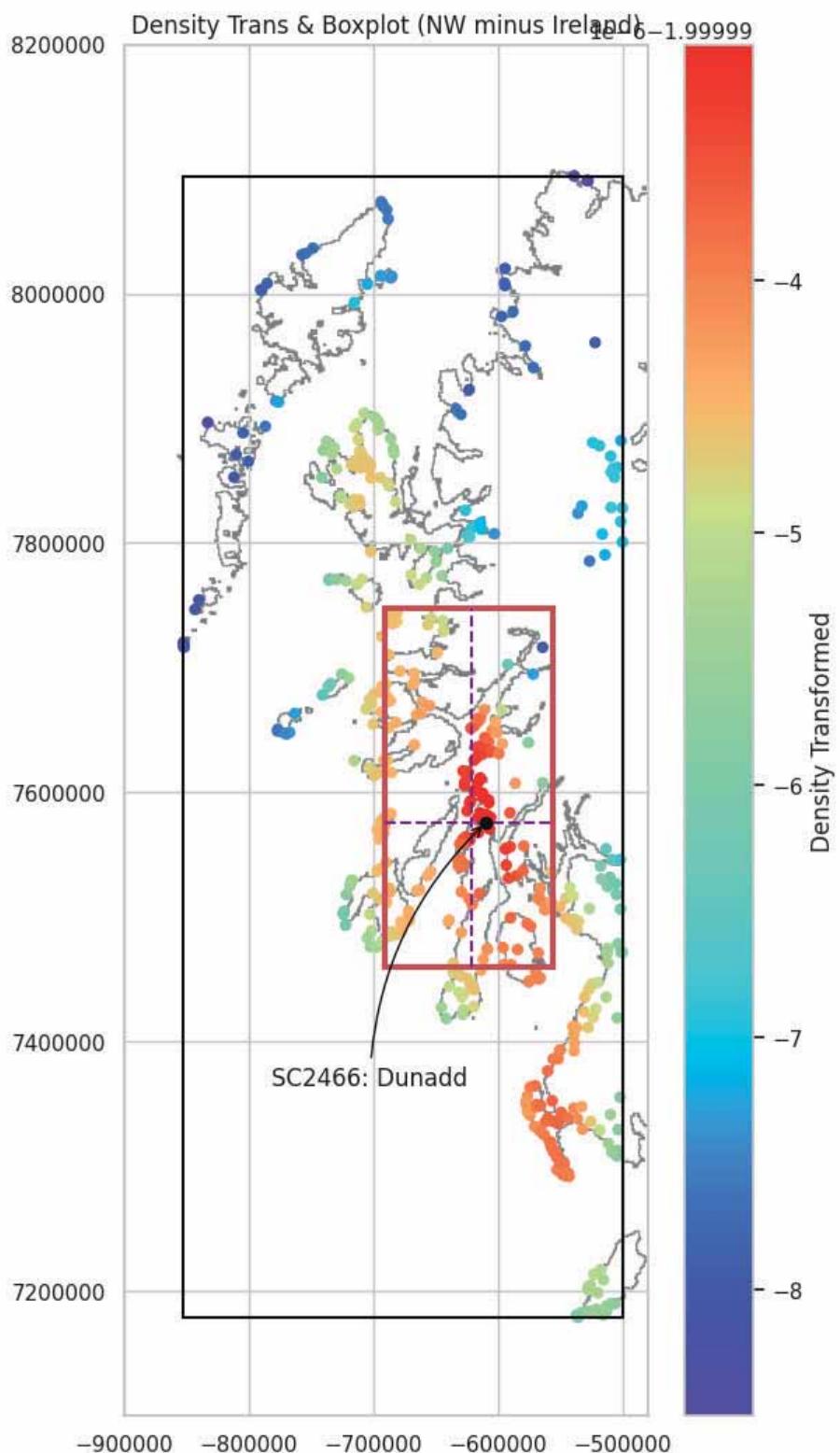
```
In [74]: show_ei_l don = True
```

Northwest classification density plots

Images will only be plotted below if the resulting filter of the data contains more sites, as a percentage of all sites in the area, than the show_percentage value set in User Settings above.

Plot Northwest location data

```
In [75]: plot_north_west_density_minus(show_ei_l don, north_west, \
                                      location_X_cluster_north_west, \
                                      location_Y_cluster_north_west)
```



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

```
In [76]: print(str(len(north_west)) + " Records")
487 Records
```

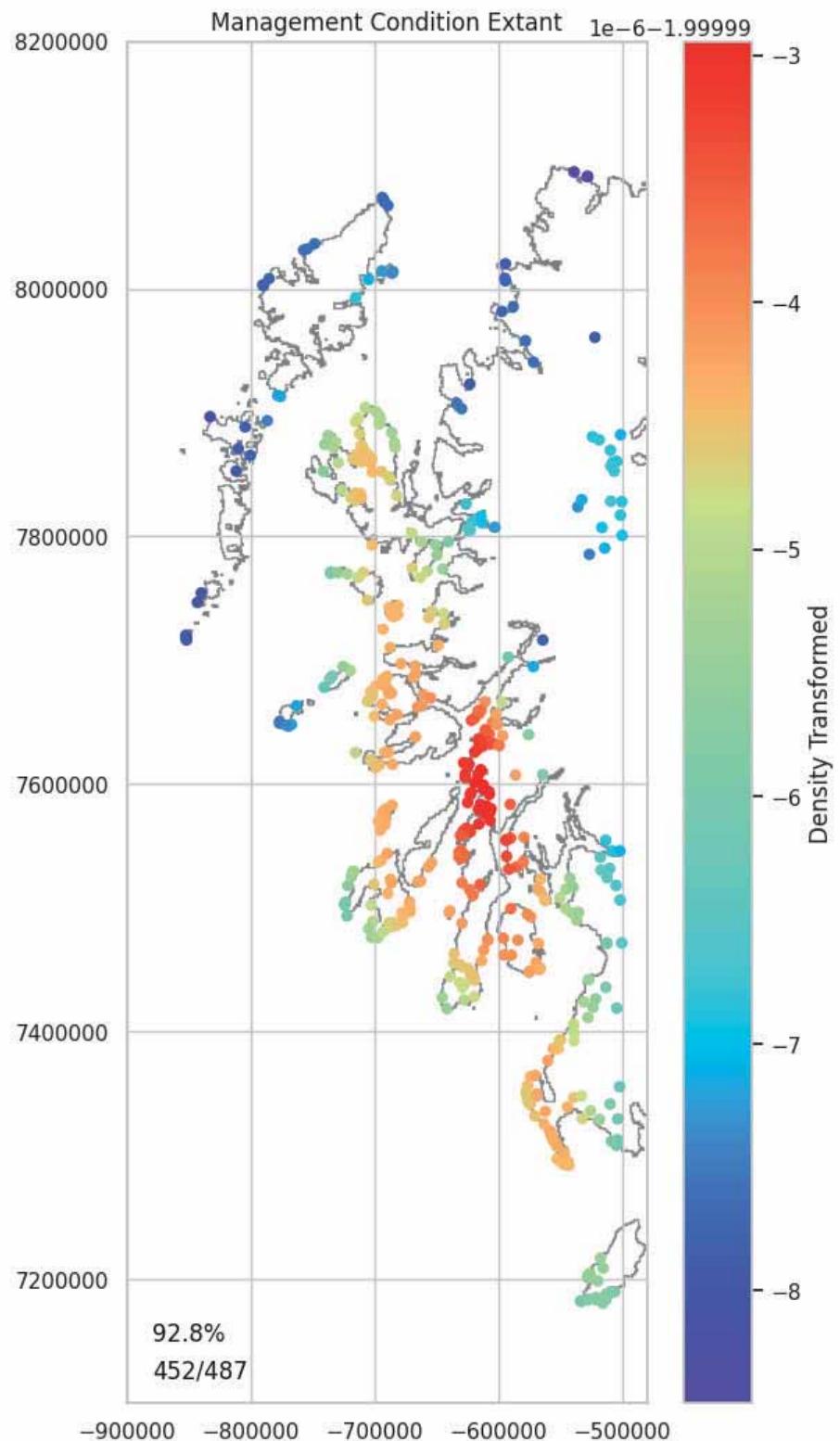
Management Encodable Data

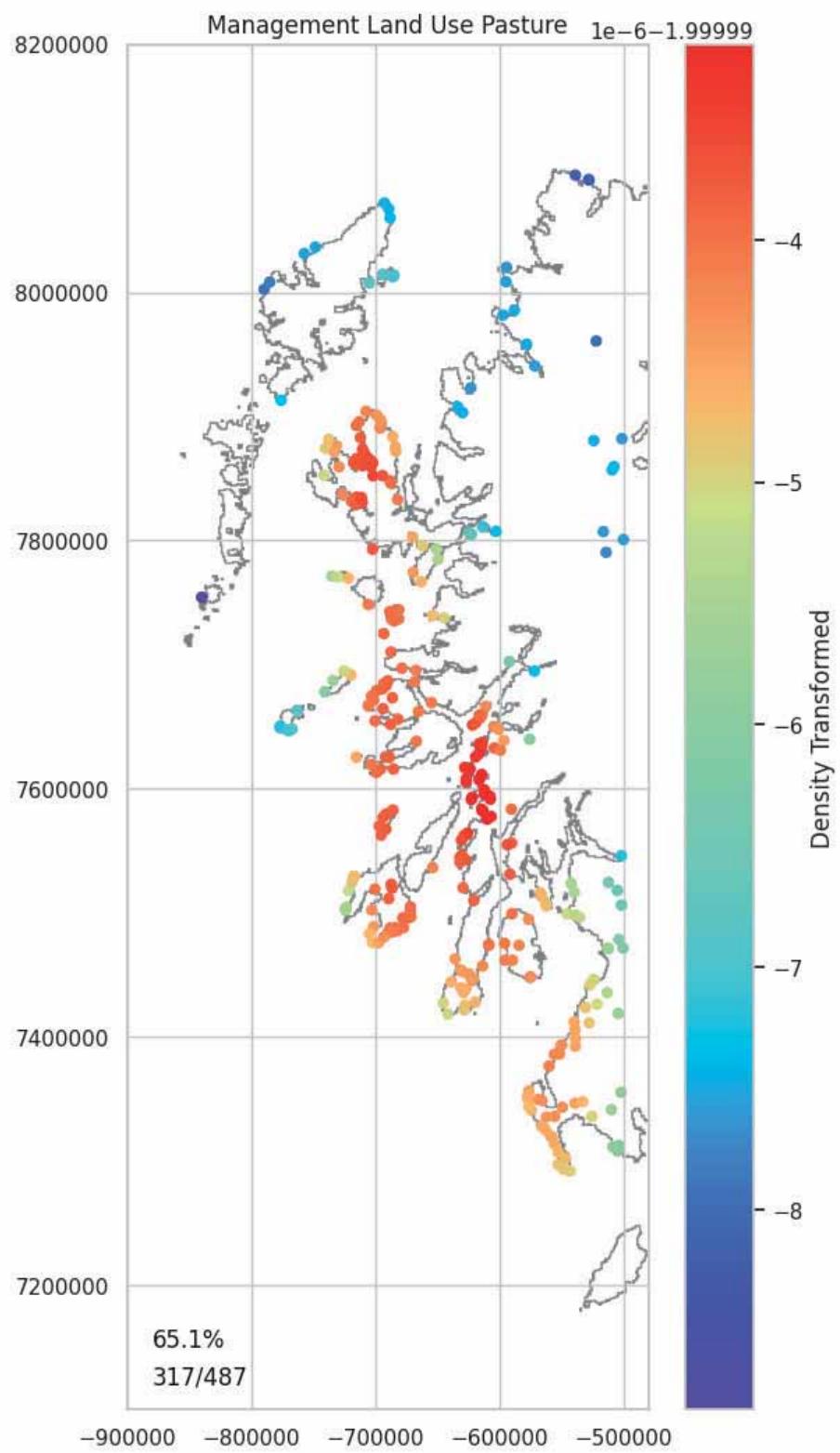
```
In [77]: management_encodeable_features = [
    'Management_Condition_Extant',
    'Management_Condition_Cropmark',
    'Management_Condition_Destroyed',
    'Management_Land_Use_Woodland',
    'Management_Land_Use_Plantation',
    'Management_Land_Use_Parkland',
```

```
'Management_Land_Use_Pasture',
'Management_Land_Use_Arabic',
'Management_Land_Use_Scrub',
'Management_Land_Use_Outcrop',
'Management_Land_Use_Moorland',
'Management_Land_Use_Heath',
'Management_Land_Use_Urban',
'Management_Land_Use_Coastal',
'Management_Land_Use_Other']
```

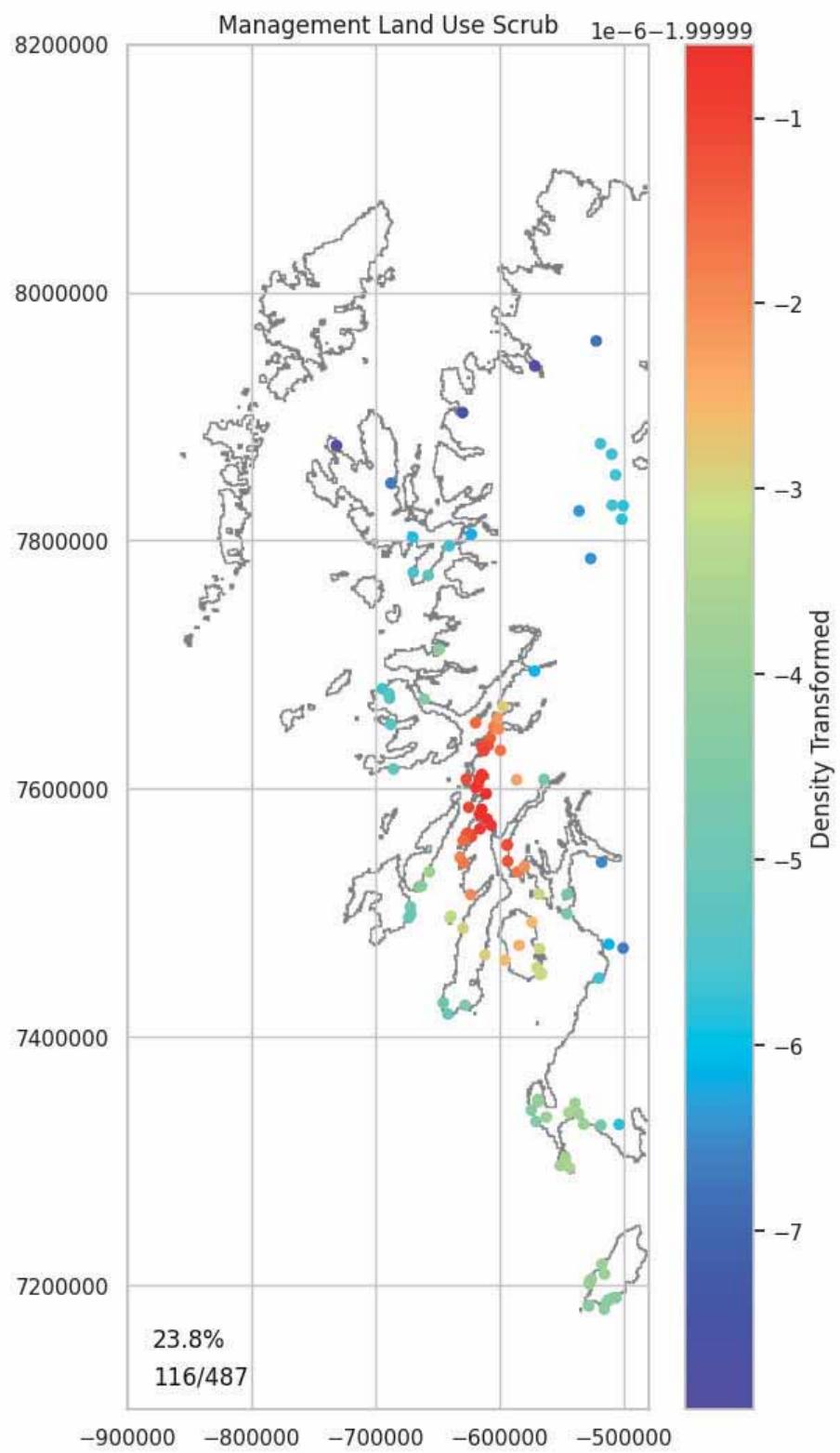
```
In [78]: plot_encodeable(hillforts_data, north_west, management_encodeable_features, show_percentage)
```

487

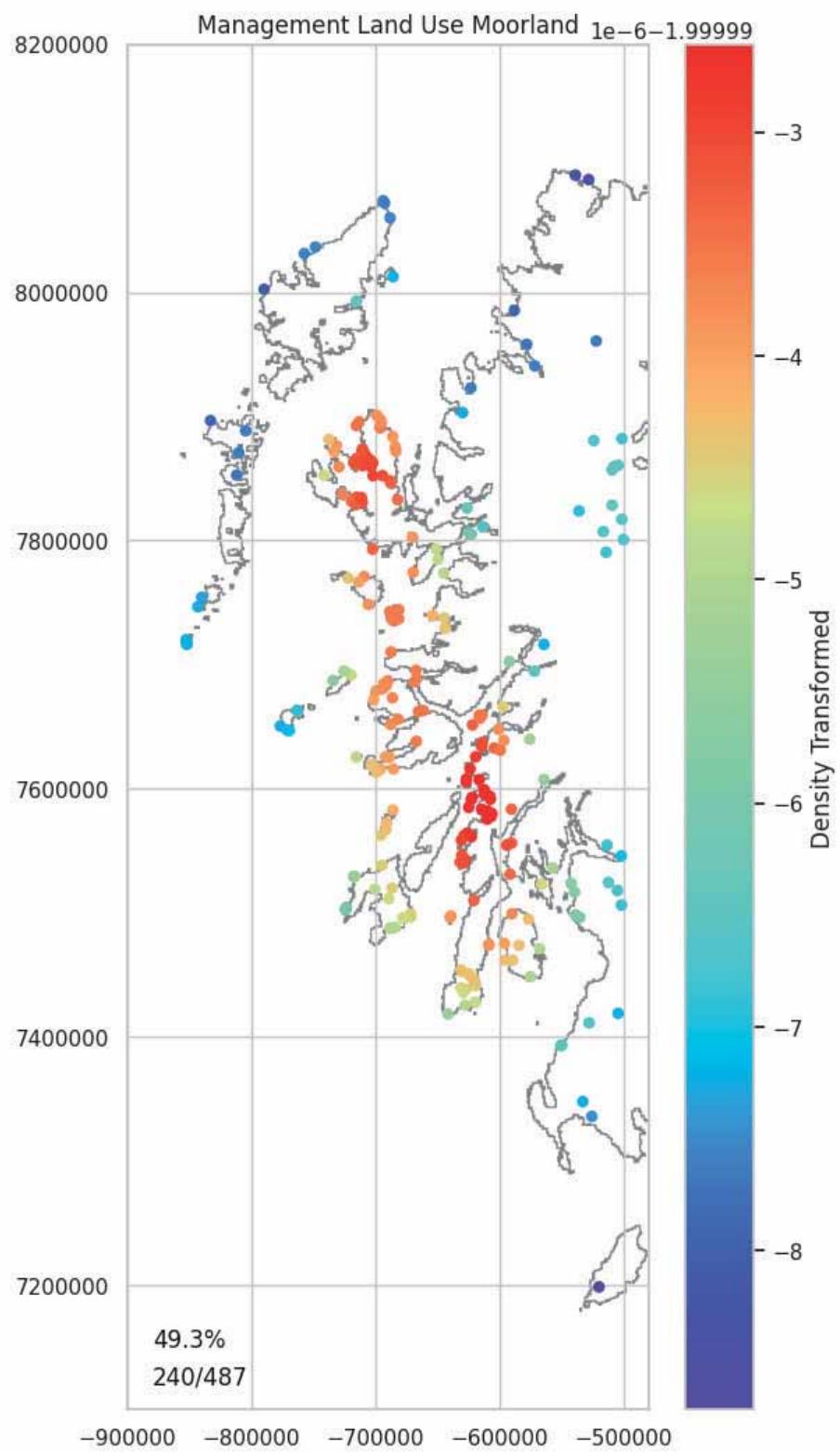




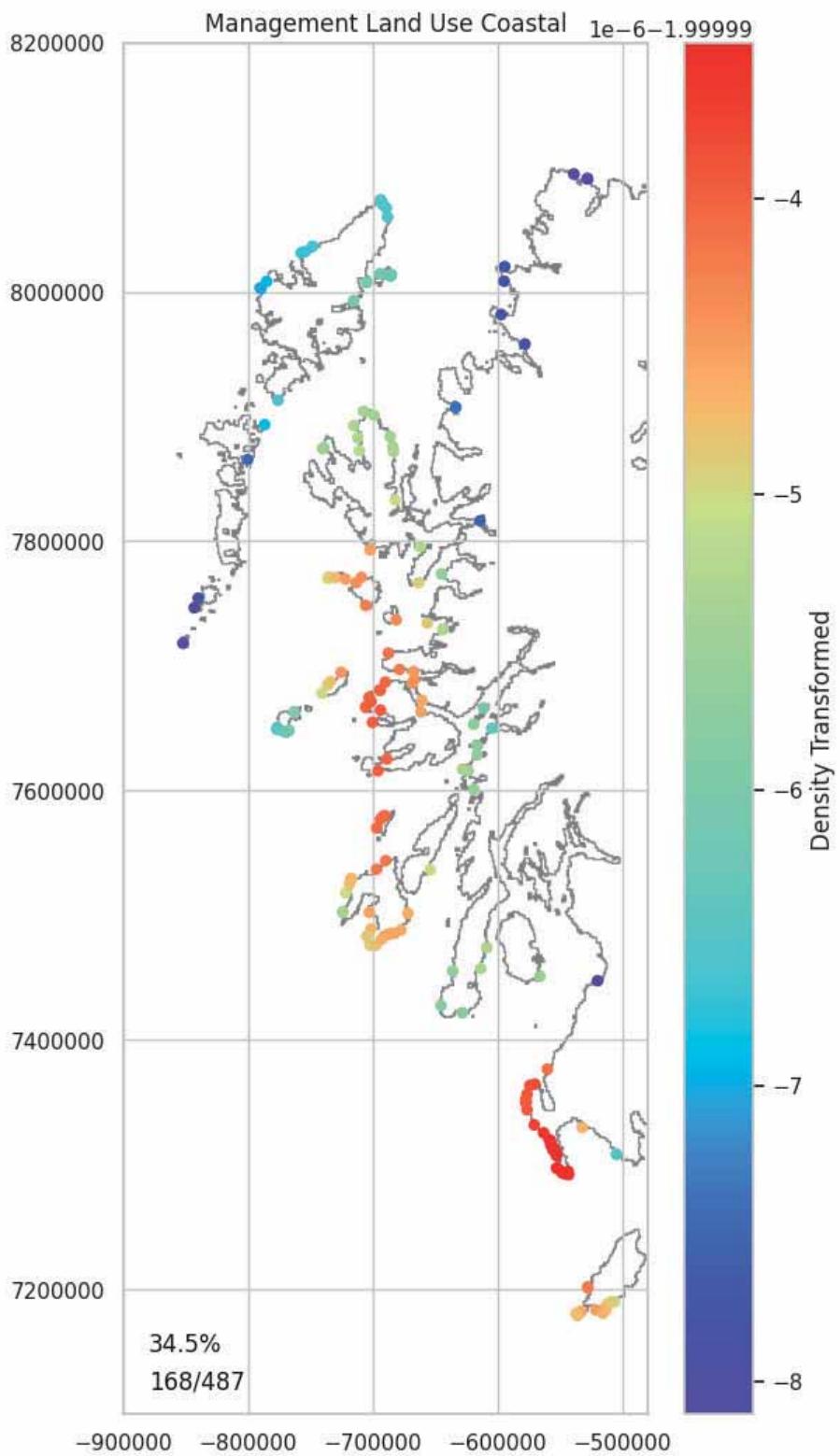
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Landscape Numeric Data

```
In [79]: landscape_numeric_features = [
    'Landscape_Alitude'
```

```
In [80]: landscape_numeric_data = \
hillforts_data[landscape_numeric_features].copy()
landscape_numeric_data.head()
```

```
Out[80]: Landscape_Altitude
```

	Landscape_Altitude
0	276.0
1	150.0
2	225.0
3	150.0
4	338.0

```
In [81]: location_Landscape_numeric_data = \
pd.merge(location_numeric_data_short, landscape_numeric_data, left_index=True, \
         right_index=True)
```

```
In [82]: over_300 = \
location_Landscape_numeric_data\[ \
    (location_Landscape_numeric_data['Landscape_Altitude'] >= 300) & \
    (location_Landscape_numeric_data['Landscape_Altitude'] < 400)\].copy()
over_300['Landscape_Altitude'] = "Yes"
```

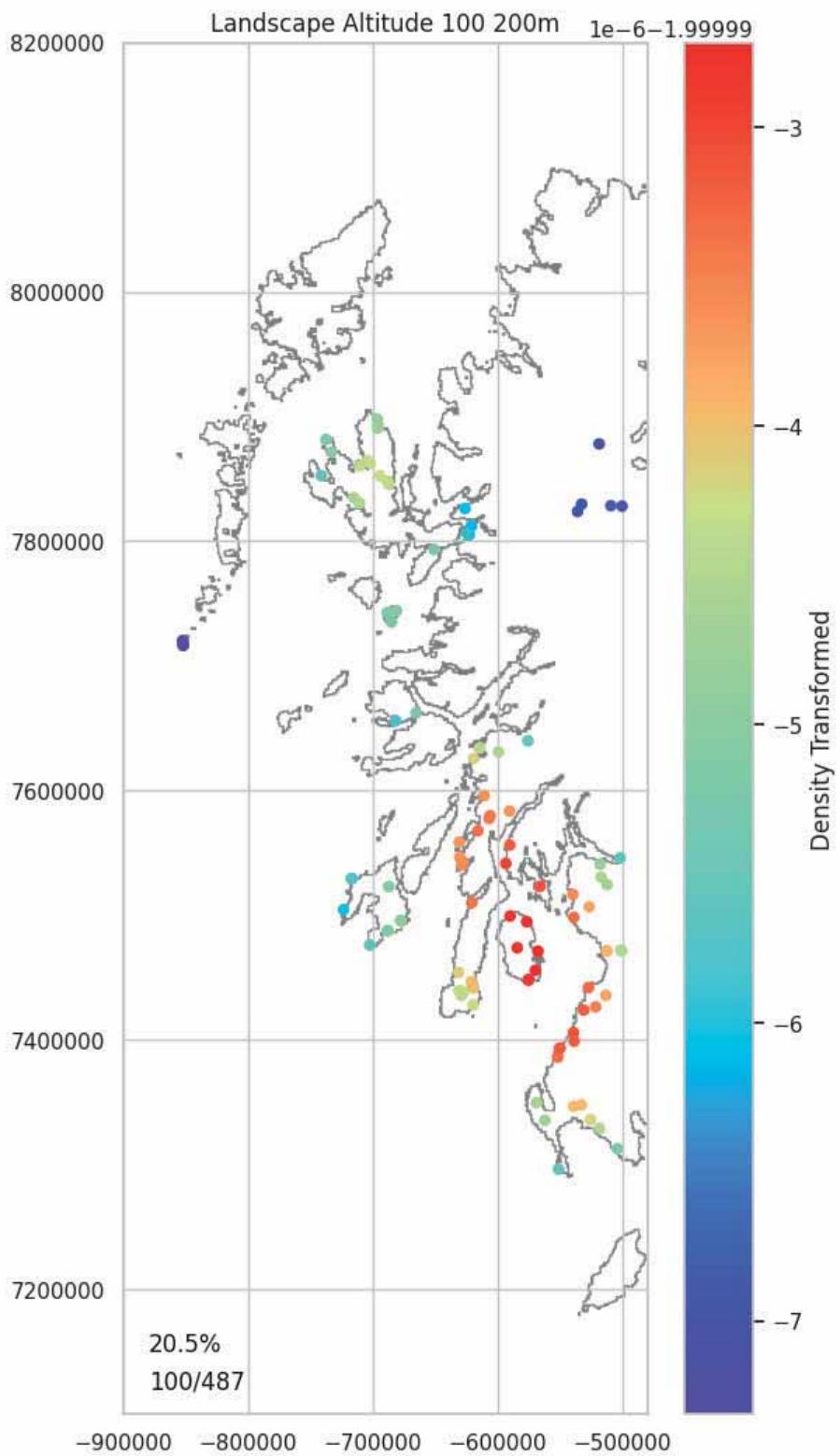
```
In [83]: plot_altitude(hills_data, north_west, over_300, 'Landscape_Altitude', "300-400m", show_percentage)
7
```

```
In [84]: over_200 = \
location_Landscape_numeric_data\[ \
    (location_Landscape_numeric_data['Landscape_Altitude'] >= 200) & \
    (location_Landscape_numeric_data['Landscape_Altitude'] < 300)\].copy()
over_200['Landscape_Altitude'] = "Yes"
```

```
In [85]: plot_altitude(hills_data, north_west, over_200, 'Landscape_Altitude', "200-300m", show_percentage)
23
```

```
In [86]: over_100 = \
location_Landscape_numeric_data\[ \
    (location_Landscape_numeric_data['Landscape_Altitude'] >= 100) & \
    (location_Landscape_numeric_data['Landscape_Altitude'] < 200)\].copy()
over_100['Landscape_Altitude'] = "Yes"
```

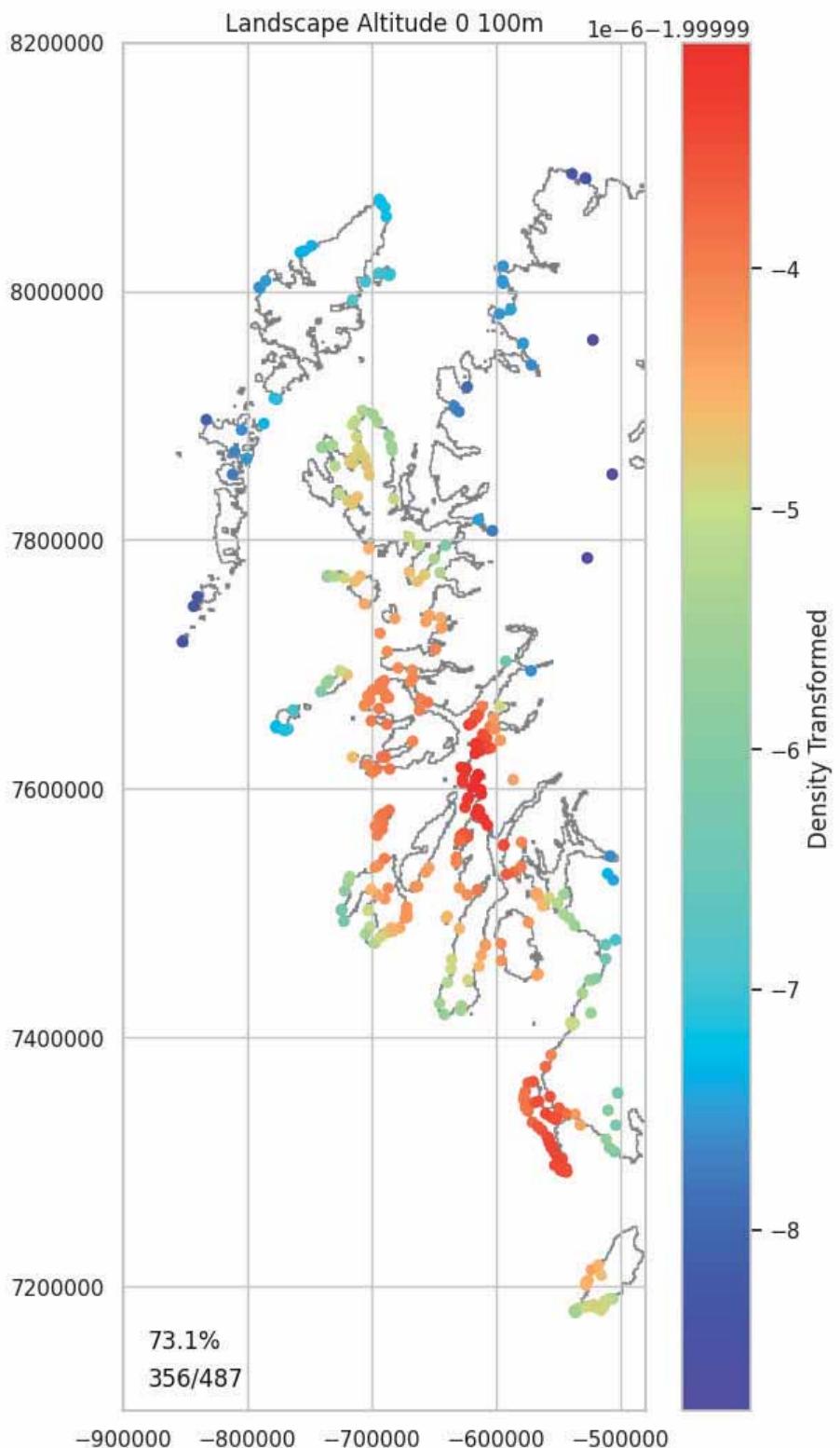
```
In [87]: plot_altitude(hills_data, north_west, over_100, 'Landscape_Altitude', "100-200m", show_percentage)
100
```



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

```
In [88]: over_000 = \
    location_landscape_numeric_data[
        (location_landscape_numeric_data['Landscape_Altitude'] >= 0) & \
        (location_landscape_numeric_data['Landscape_Altitude'] < 100)].copy()
    over_000['Landscape_Altitude'] = "Yes"
```

```
In [89]: plot_altitude(hillforts_data, north_west, over_000, 'Landscape_Altitude', '0-100m', show_percentage)
```



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

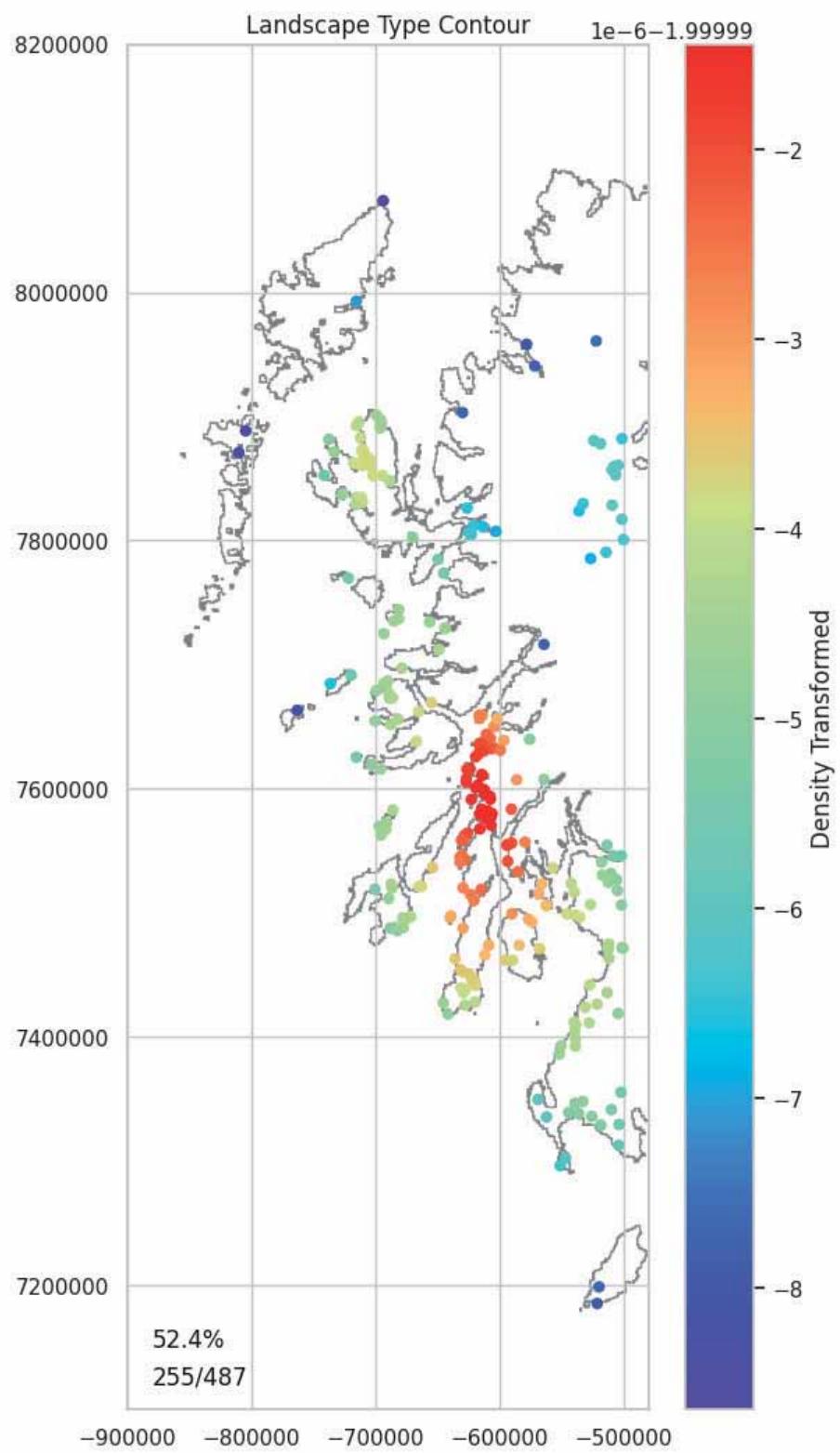
Landscape Encodeable Data

```
In [90]: landscape_encodeable_features = [
    'Landscape_Type_Contour',
    'Landscape_Type_Partial',
    'Landscape_Type_Promontory',
    'Landscape_Type_Hill_slope',
    'Landscape_Type_Level',
    'Landscape_Type_Marsh',
    'Landscape_Type_Multiple',
    'Landscape_Topography_Hill_top',
    'Landscape_Topography_Coastal']
```

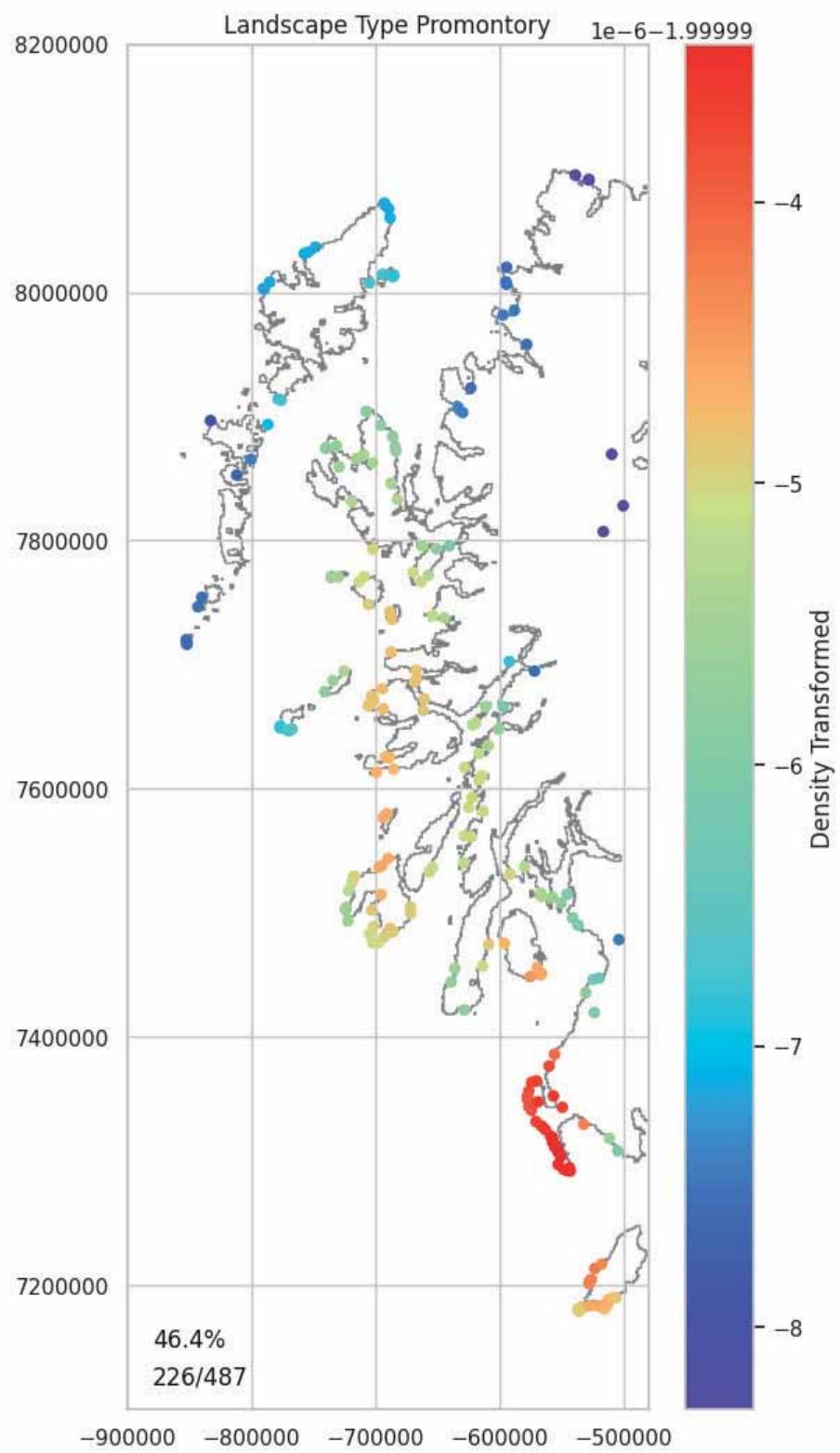
```
'Landscape_Topography_Inland',
'Landscape_Topography_Valley',
'Landscape_Topography_Knoll',
'Landscape_Topography_Ridge',
'Landscape_Topography_Scarp',
'Landscape_Topography_Hillside',
'Landscape_Topography_Lowland',
'Landscape_Topography_Spur',
'Landscape_Aspect_N',
'Landscape_Aspect_NE',
'Landscape_Aspect_E',
'Landscape_Aspect_SE',
'Landscape_Aspect_S',
'Landscape_Aspect_SW',
'Landscape_Aspect_W',
'Landscape_Aspect_NW',
'Landscape_Aspect_Level']
```

```
In [91]: plot_encodeable(hillsorts_data, north_west, landscape_encodeable_features, show_percentage)
```

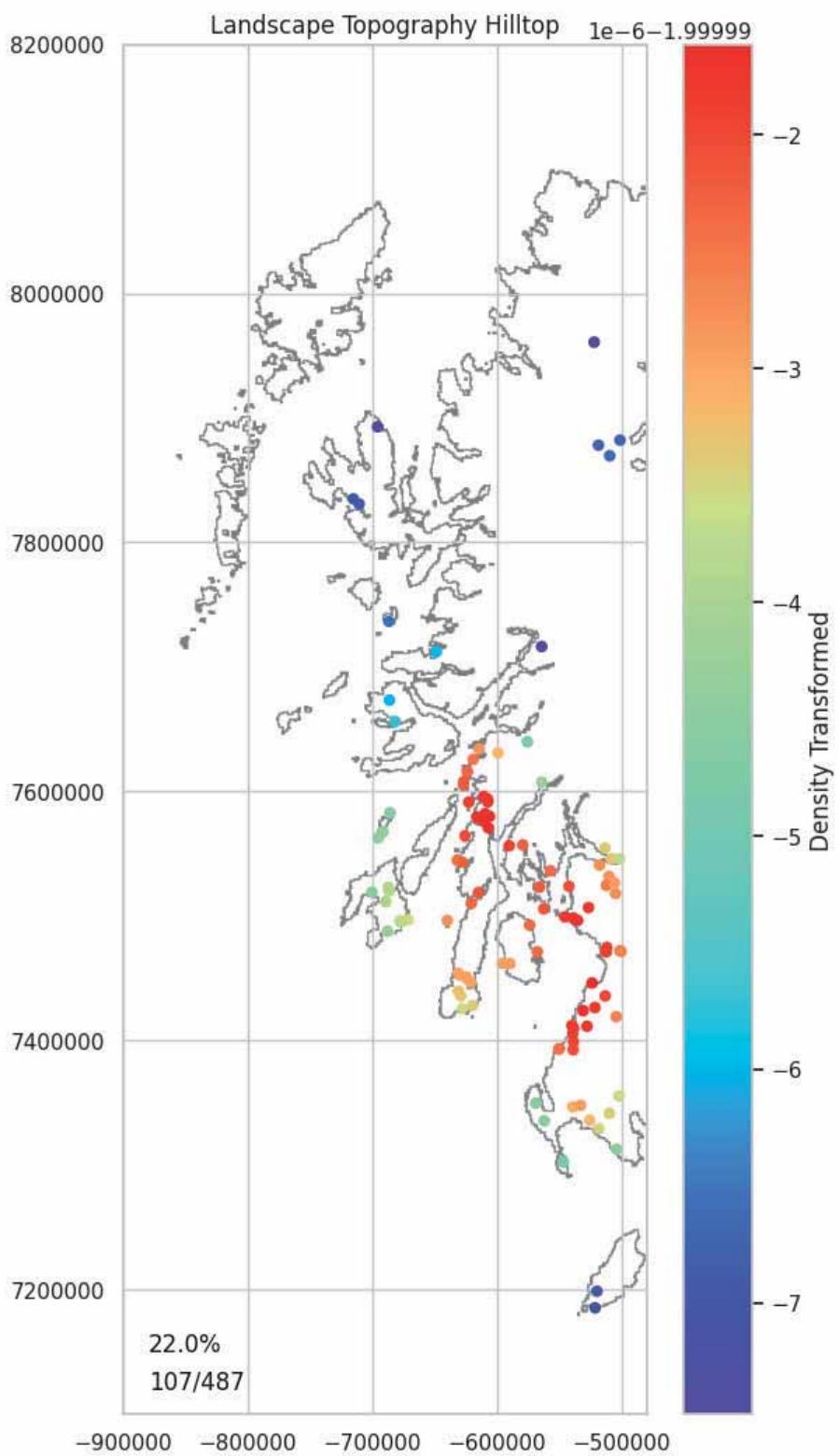
487



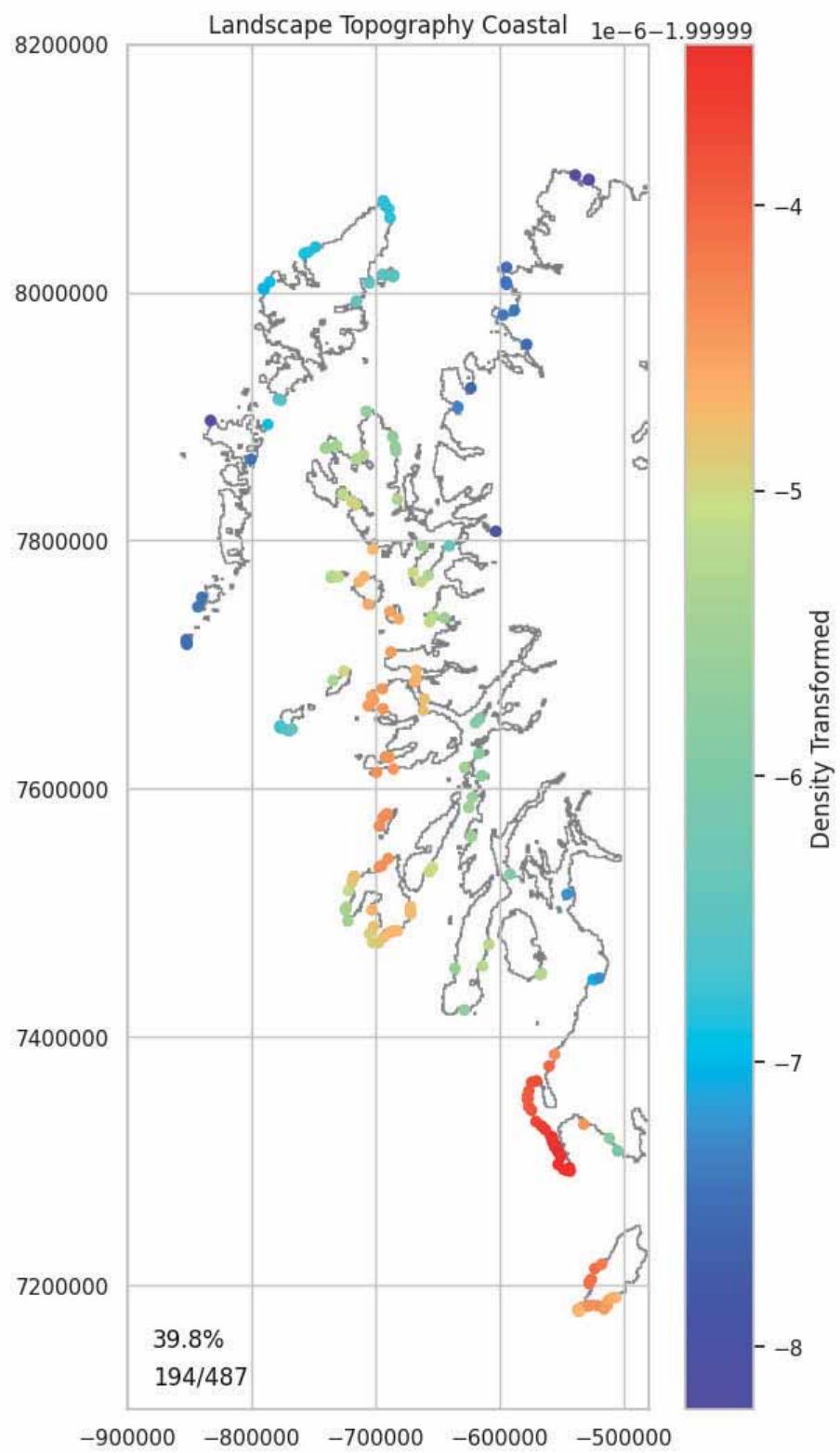
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



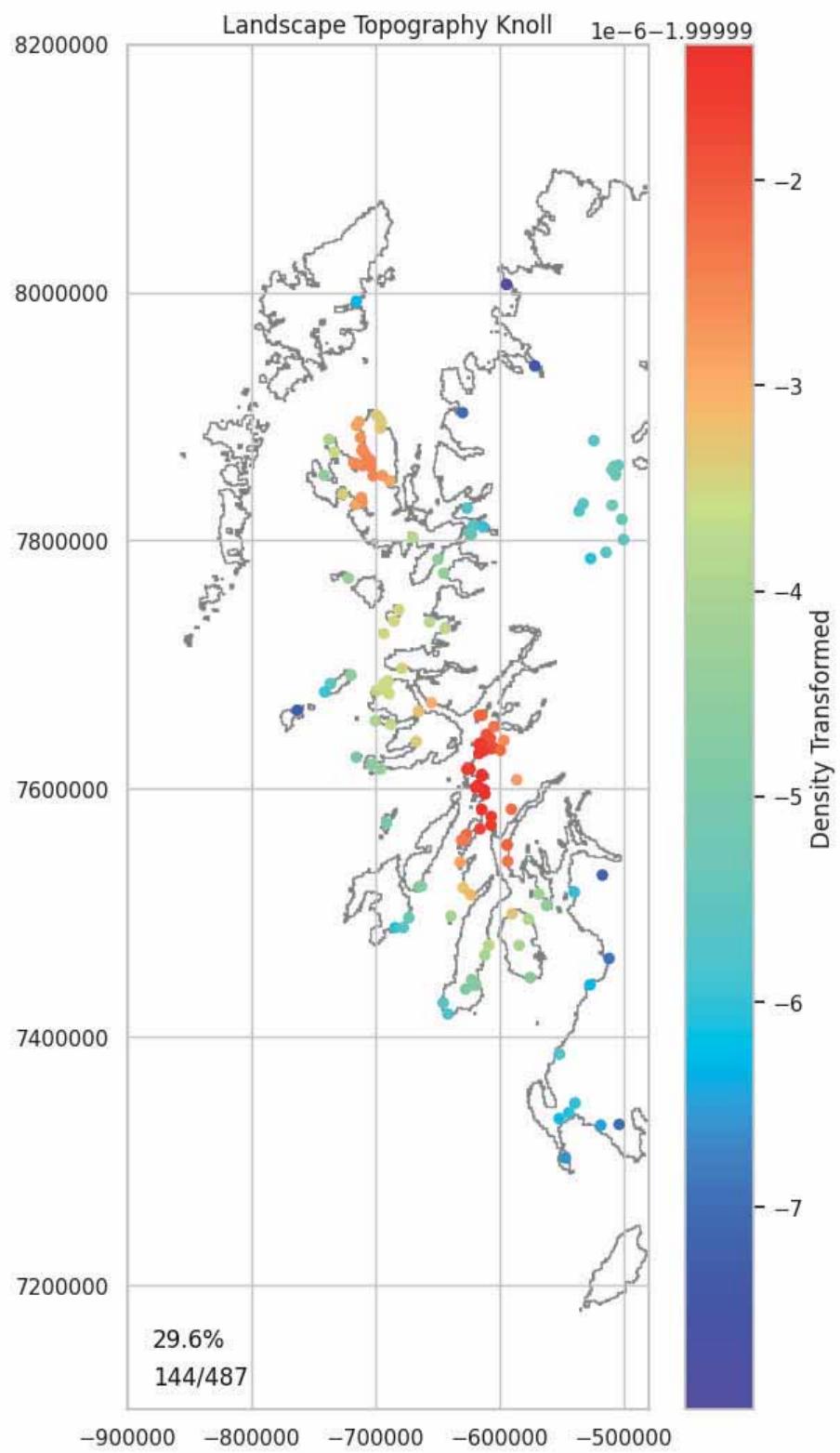
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



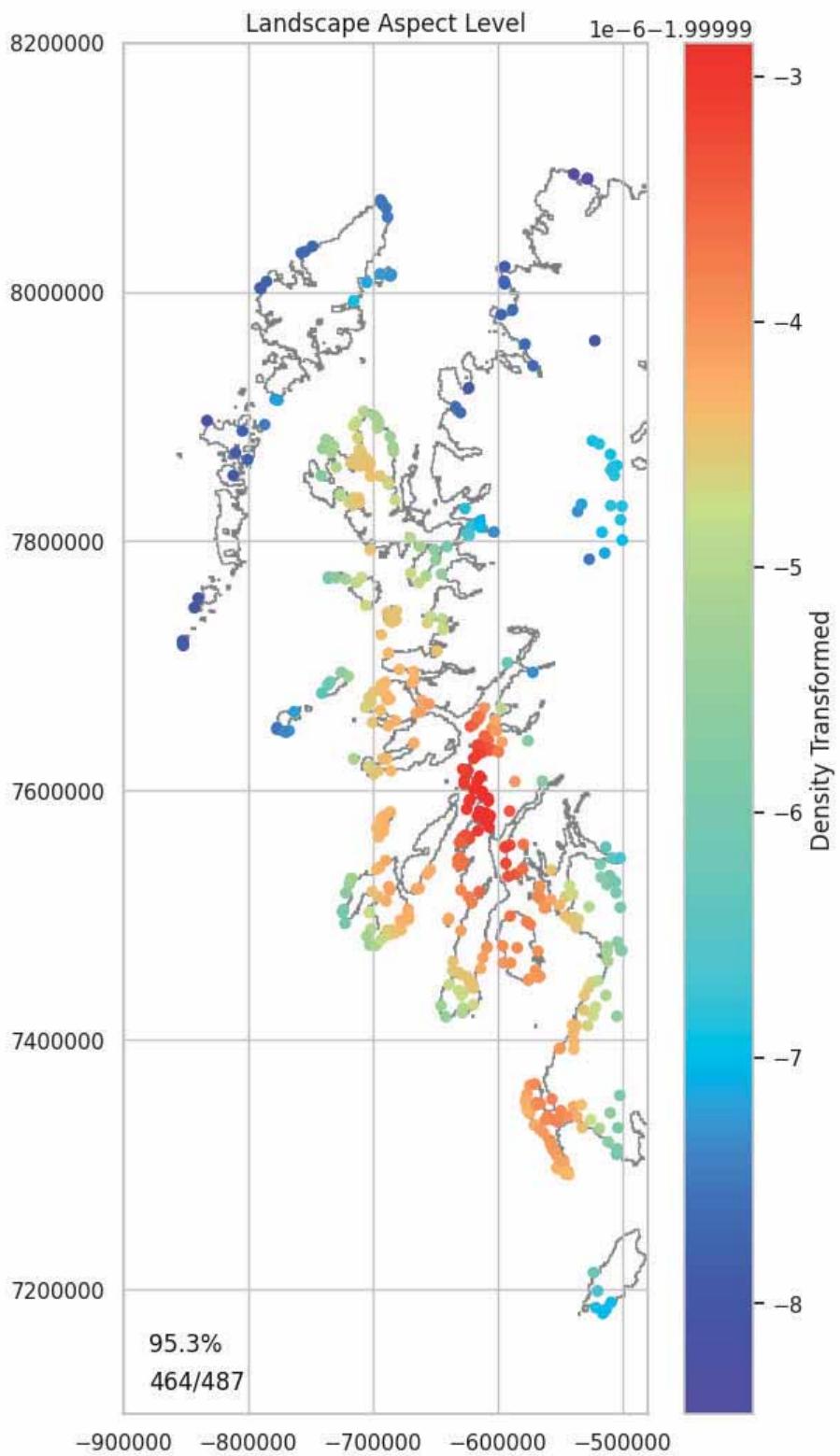
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



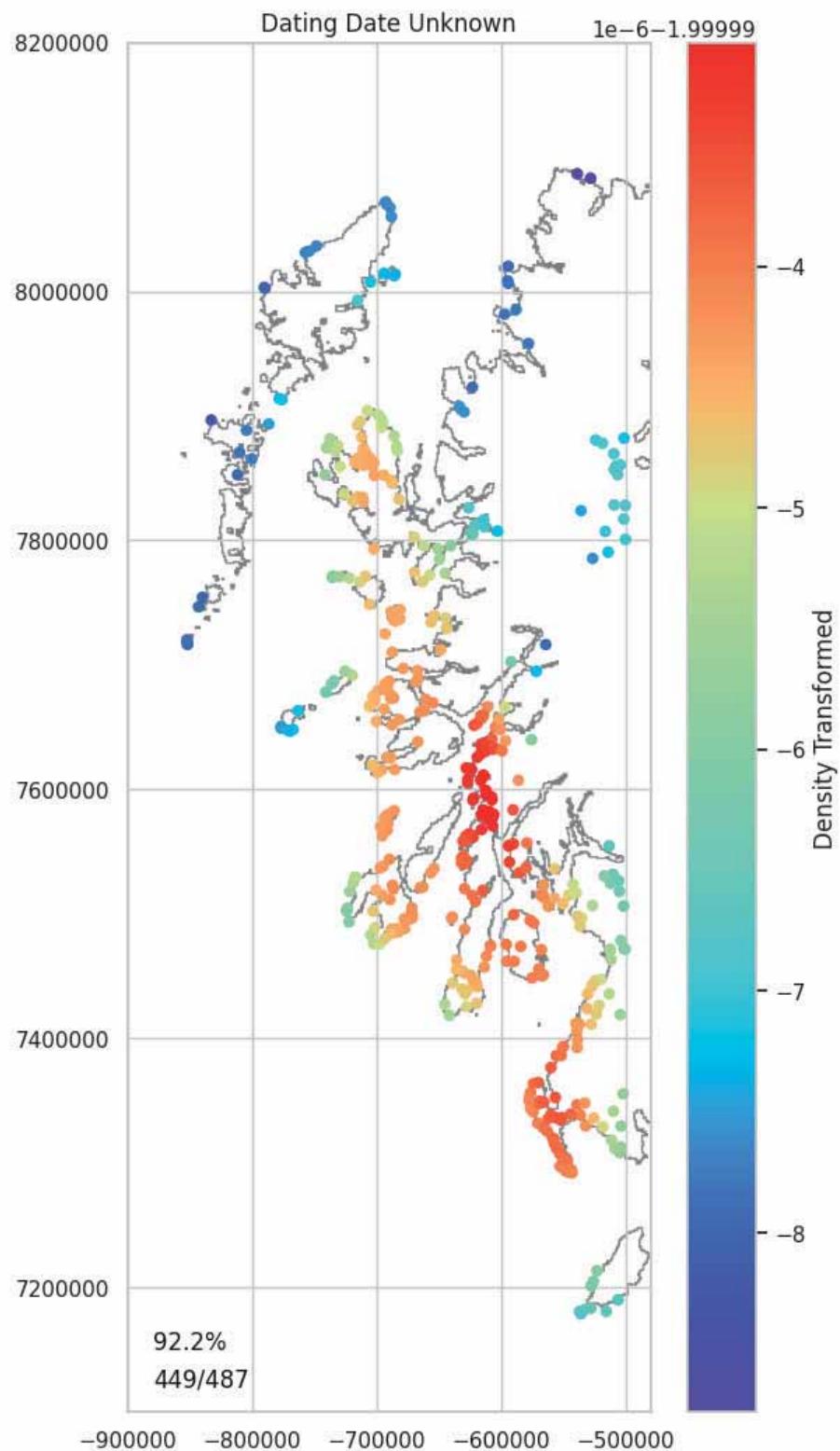
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Dating Encodable Data

```
In [92]: dating_encodeable_features = [
    'Dating_Date_Pre_1200BC',
    'Dating_Date_1200BC_800BC',
    'Dating_Date_800BC_400BC',
    'Dating_Date_400BC_AD50',
    'Dating_Date_AD50_AD400',
    'Dating_Date_AD400_AD800',
    'Dating_Date_Post_AD800',
    'Dating_Date_Unknown']
```

```
In [93]: plot_encodeable(hillforts_data, north_west, dating_encodeable_features, show_percentage)
```

487



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

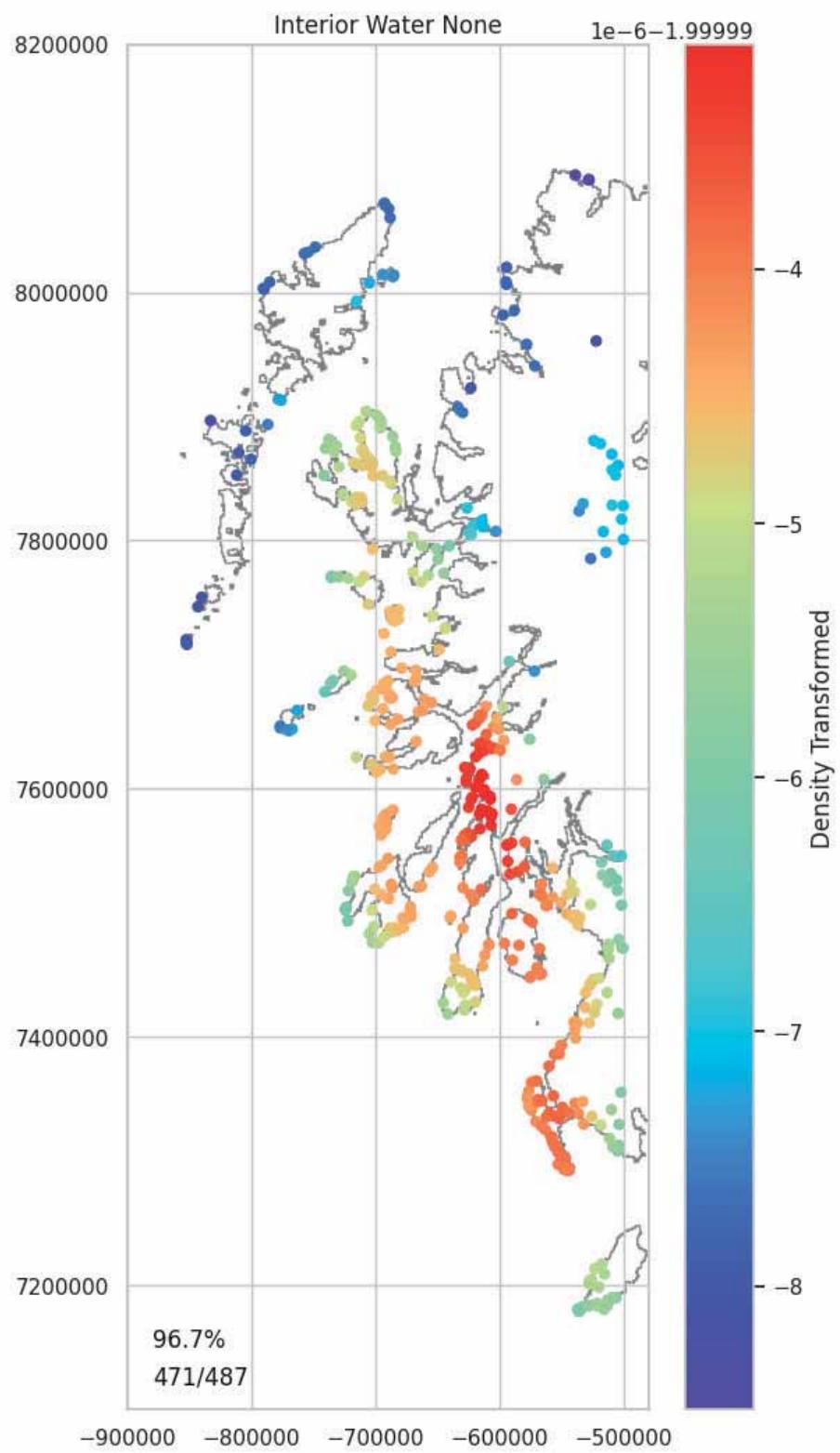
Interior Encodable Data

```
In [94]: interior_encodeable_features = [  
    'Interior_Water_None',  
    'Interior_Water_Spring',  
    'Interior_Water_Stream',  
    'Interior_Water_Pool',  
    'Interior_Water_Flush',  
    'Interior_Water_Well',
```

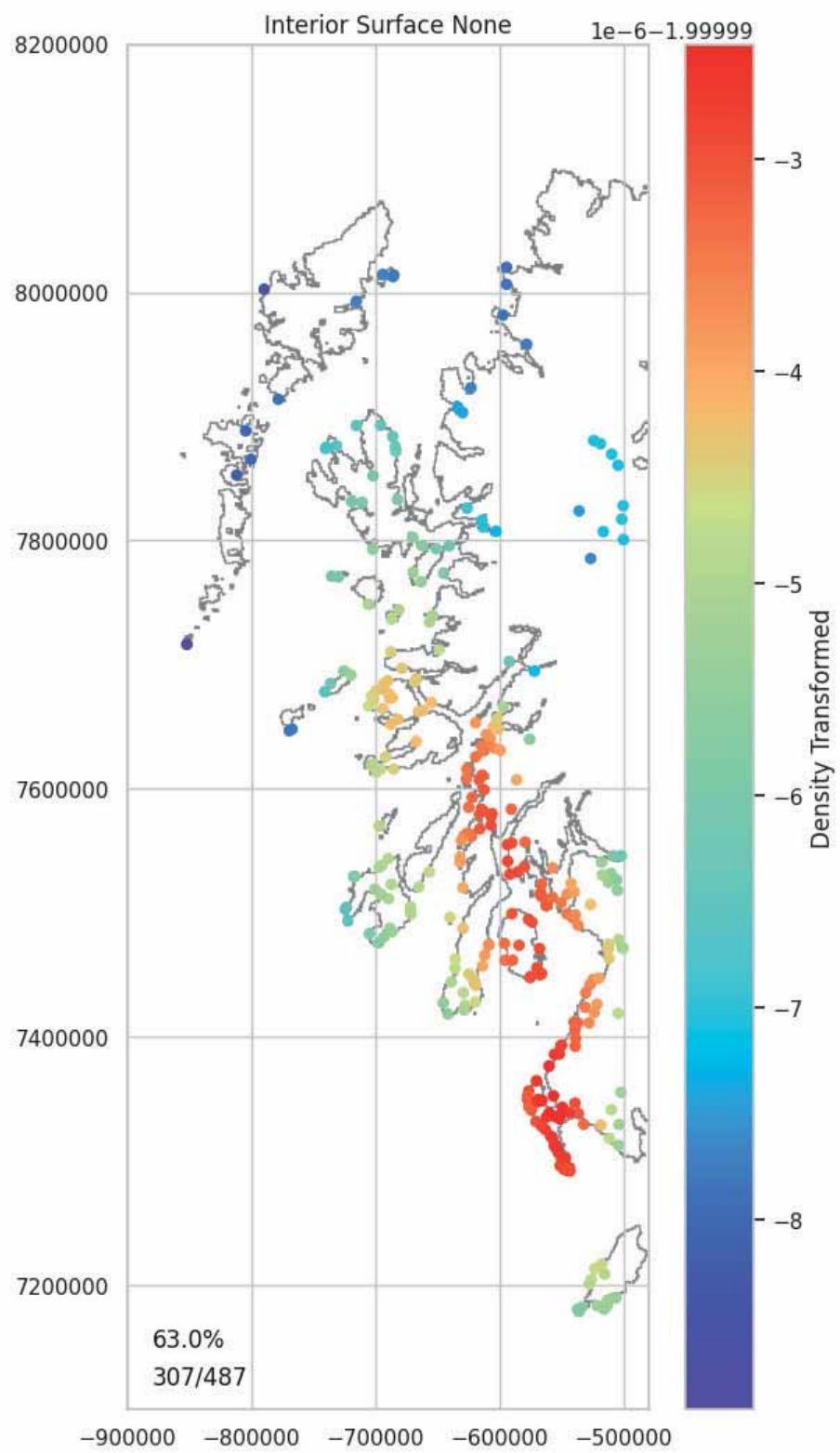
```
'Interior_Water_Other',
'Interior_Surface_None',
'Interior_Surface_Round',
'Interior_Surface_Rectangular',
'Interior_Surface_Curvilinear',
'Interior_Surface_Roundhouse',
'Interior_Surface_Pit',
'Interior_Surface_Quarry',
'Interior_Surface_Other',
'Interior_Excavation_None',
'Interior_Excavation_Pit',
'Interior_Excavation_Posthole',
'Interior_Excavation_Roundhouse',
'Interior_Excavation_Rectangular',
'Interior_Excavation_Road',
'Interior_Excavation_Quarry',
'Interior_Excavation_Other',
'Interior_Excavation_Nothing',
'Interior_Geophysics_None',
'Interior_Geophysics_Pit',
'Interior_Geophysics_Roundhouse',
'Interior_Geophysics_Rectangular',
'Interior_Geophysics_Road',
'Interior_Geophysics_Quarry',
'Interior_Geophysics_Other',
'Interior_Geophysics_Nothing']
```

```
In [95]: plot_encodeable(hills_forts_data, north_west, interior_encodeable_features, show_percentage)
```

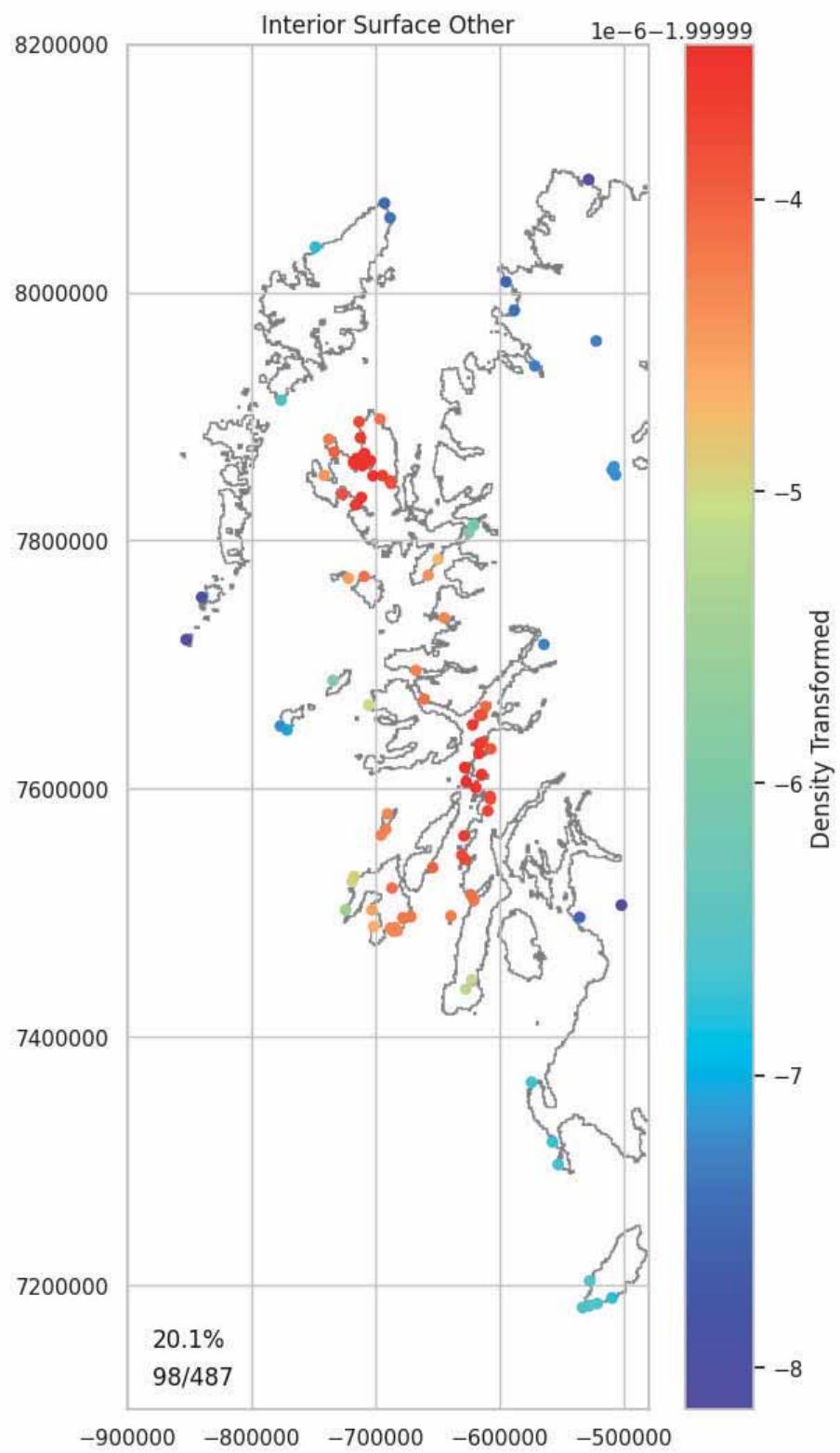
487



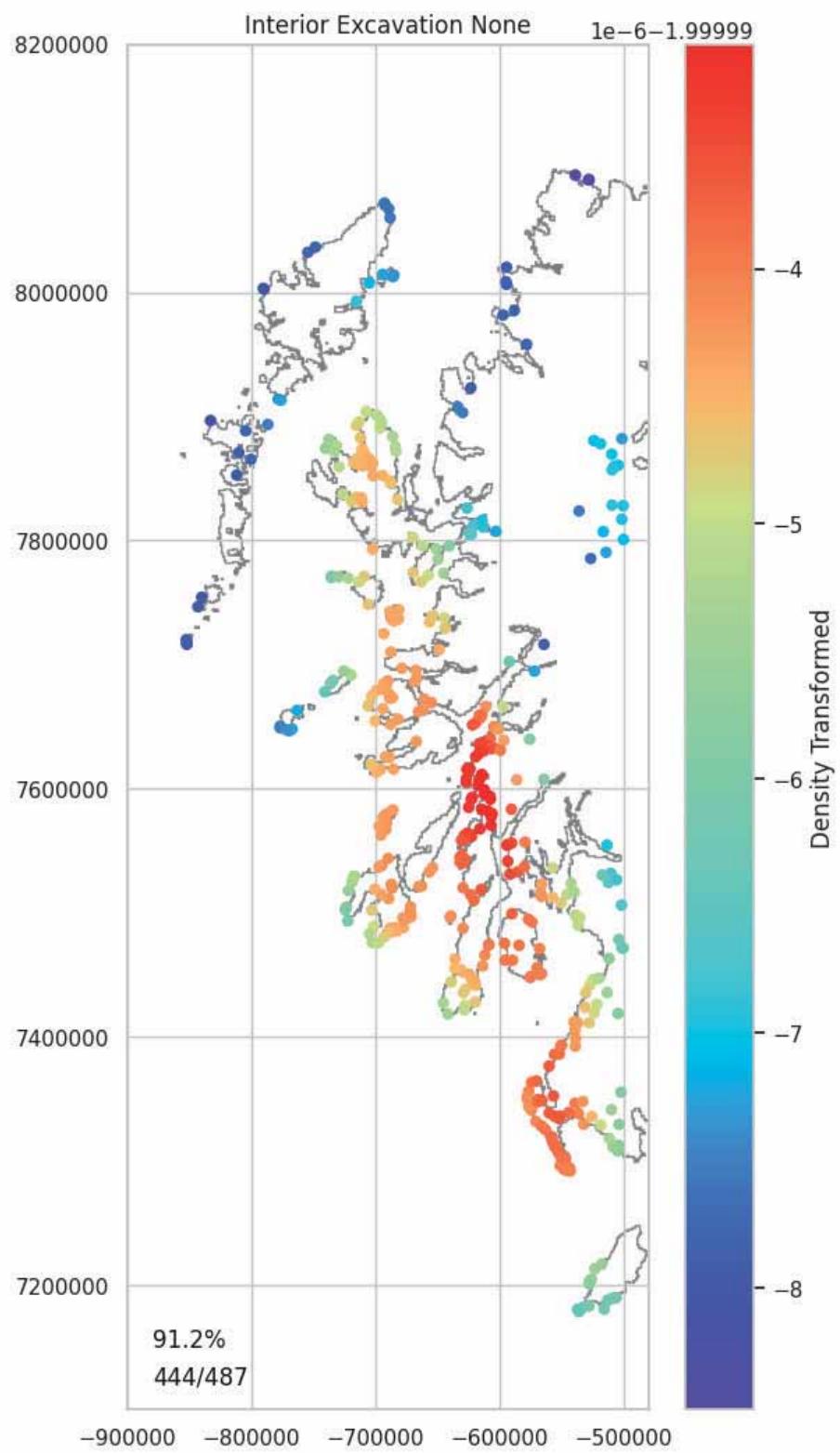
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



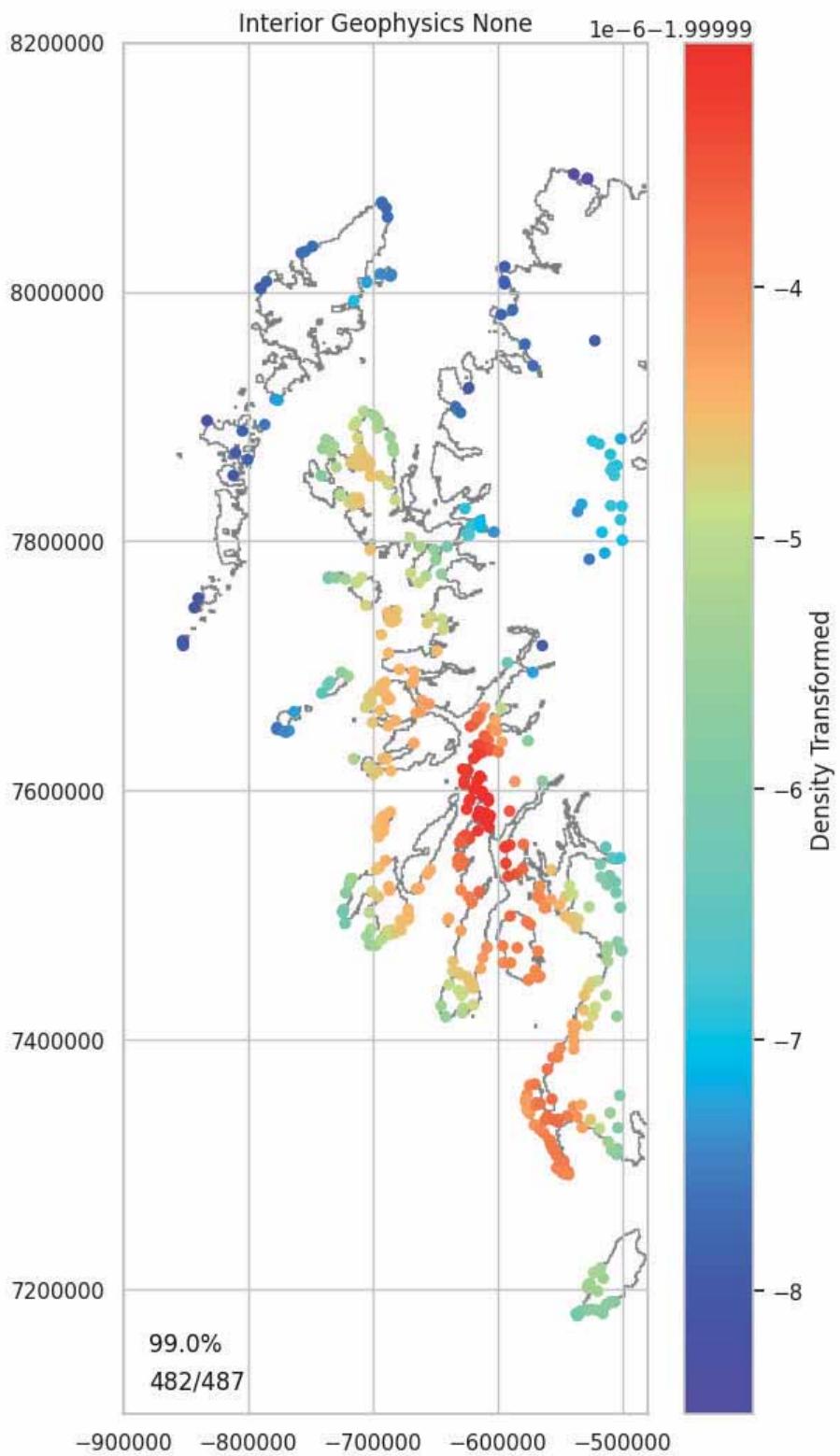
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

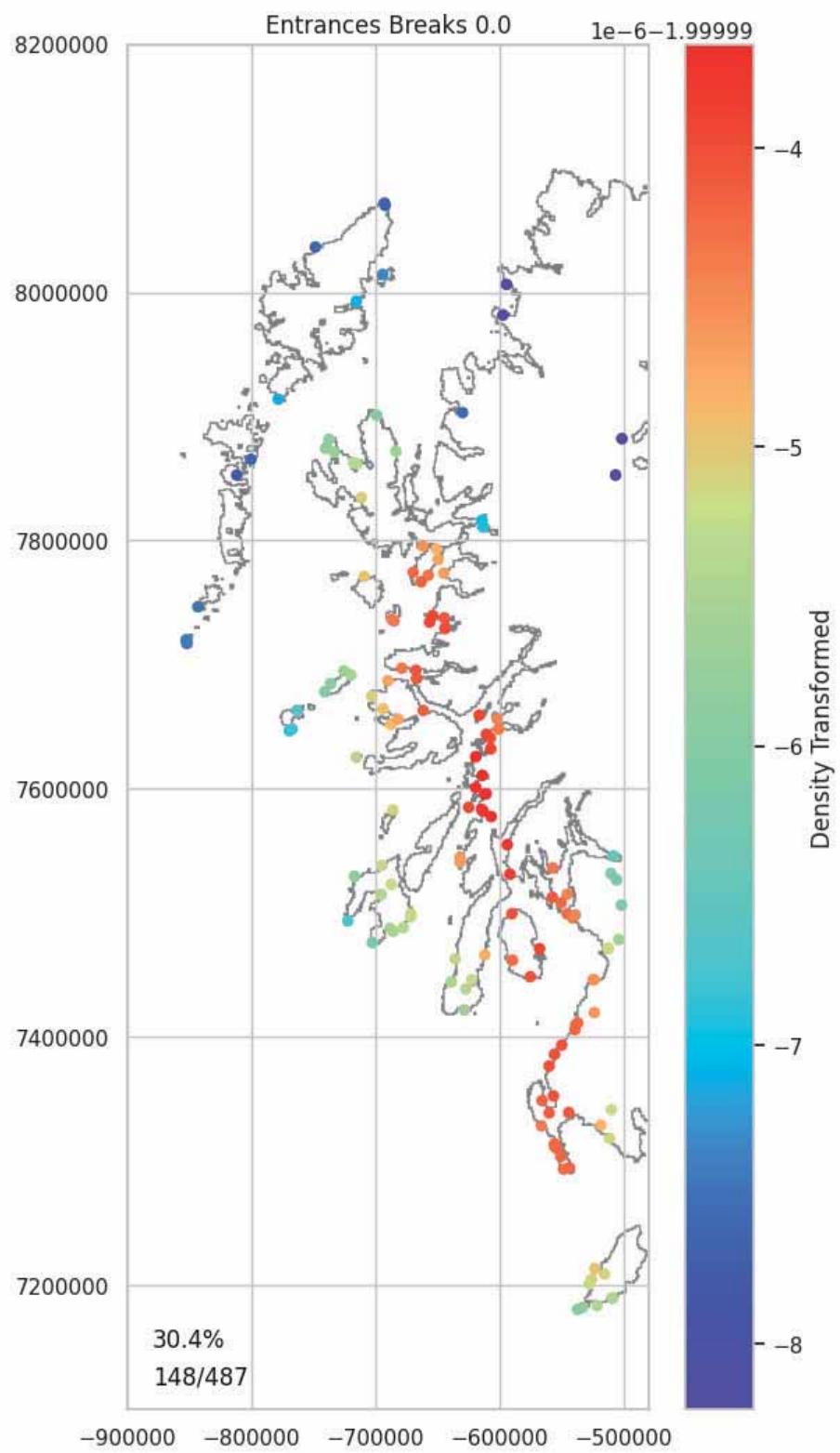


Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

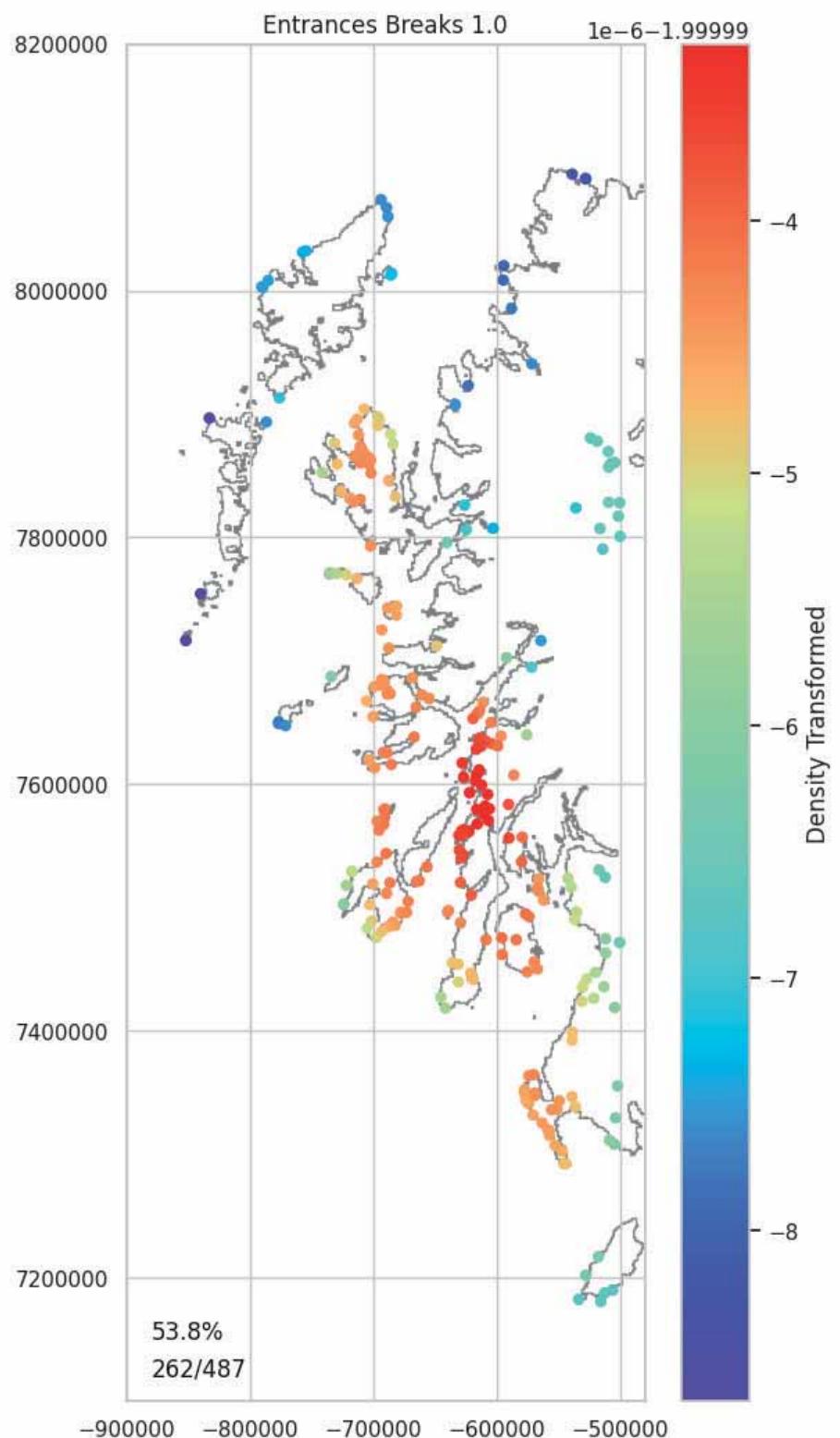
Entrance Numeric Data

```
In [96]: entrance_numeric_features = [
    'Entrances_Breaks',
    'Entrances_Original']
```

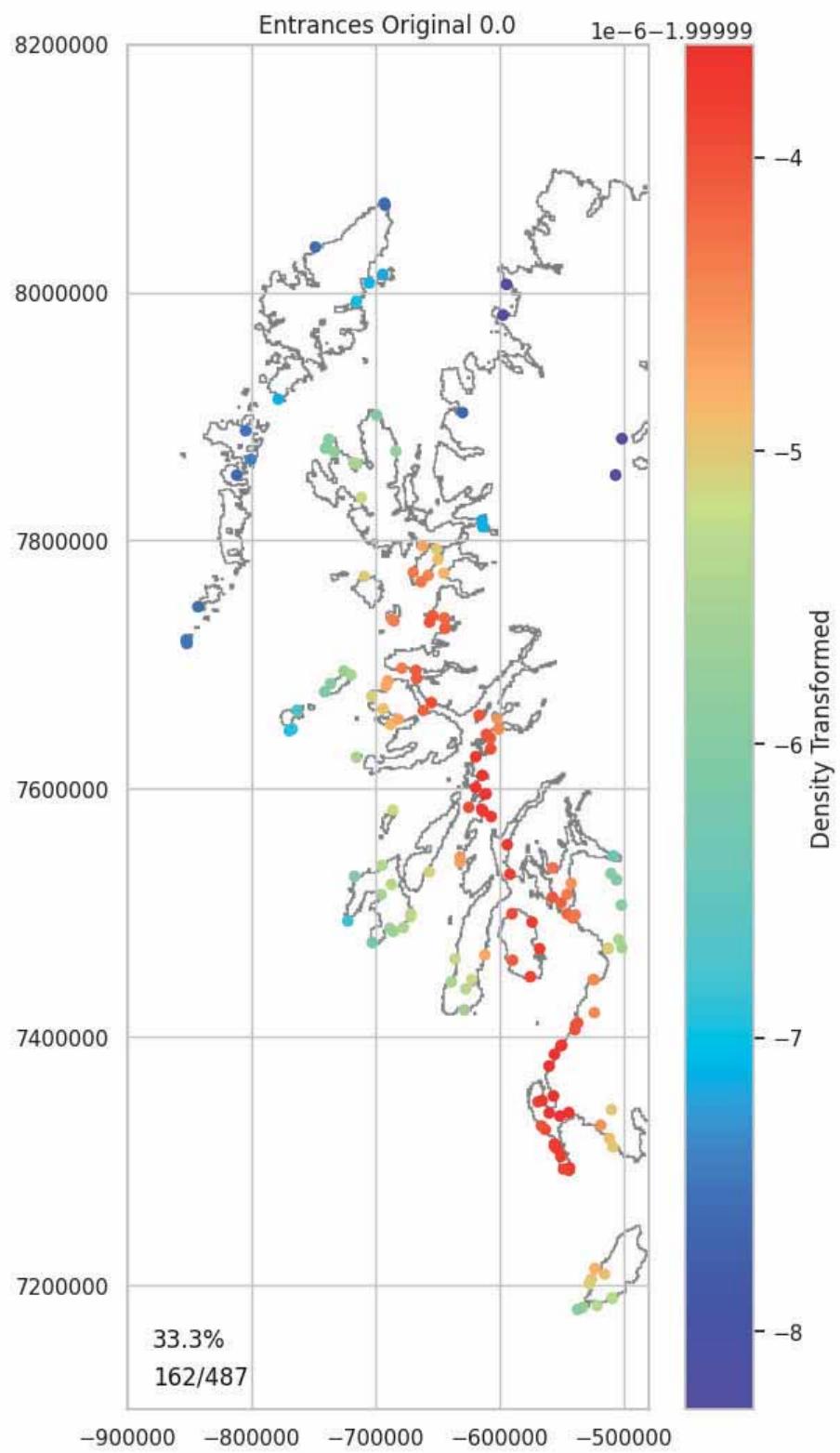
```
In [97]: plot_numeric(hillforts_data, north_west, entrance_numeric_features, show_percentage)
487
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 9.0]
```



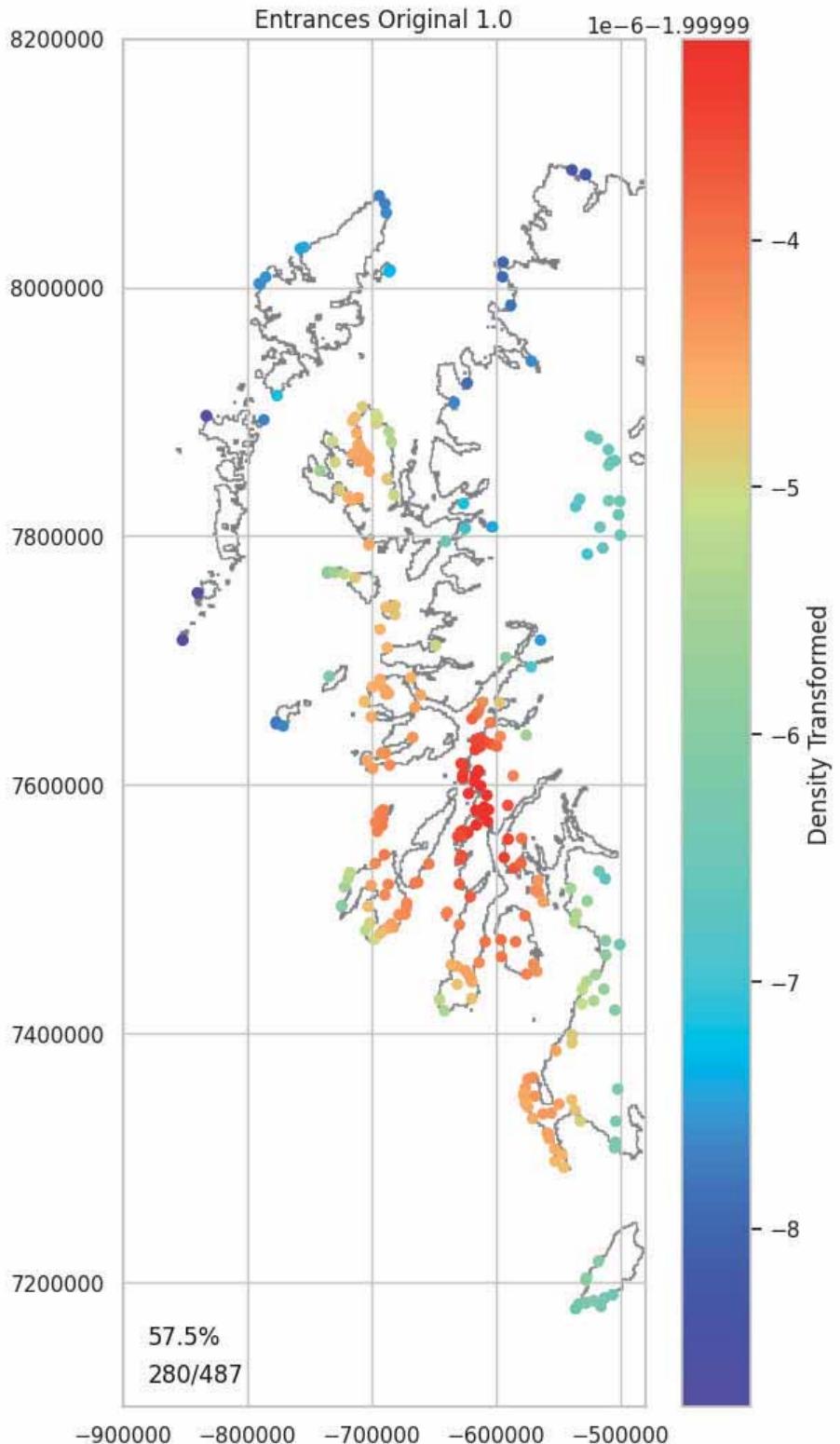
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk
[0.0, 1.0, 2.0, 3.0, 4.0]



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Entrance Encodable Data

```
In [98]: entrance_encodeable_features = [
    'Entrances_Guard_Chambers',
    'Entrances_Chevaux']
```

```
In [99]: plot_encodeable(hillforts_data, north_west, entrance_encodeable_features, show_percentage)
```

487

Enclosing Numeric Data

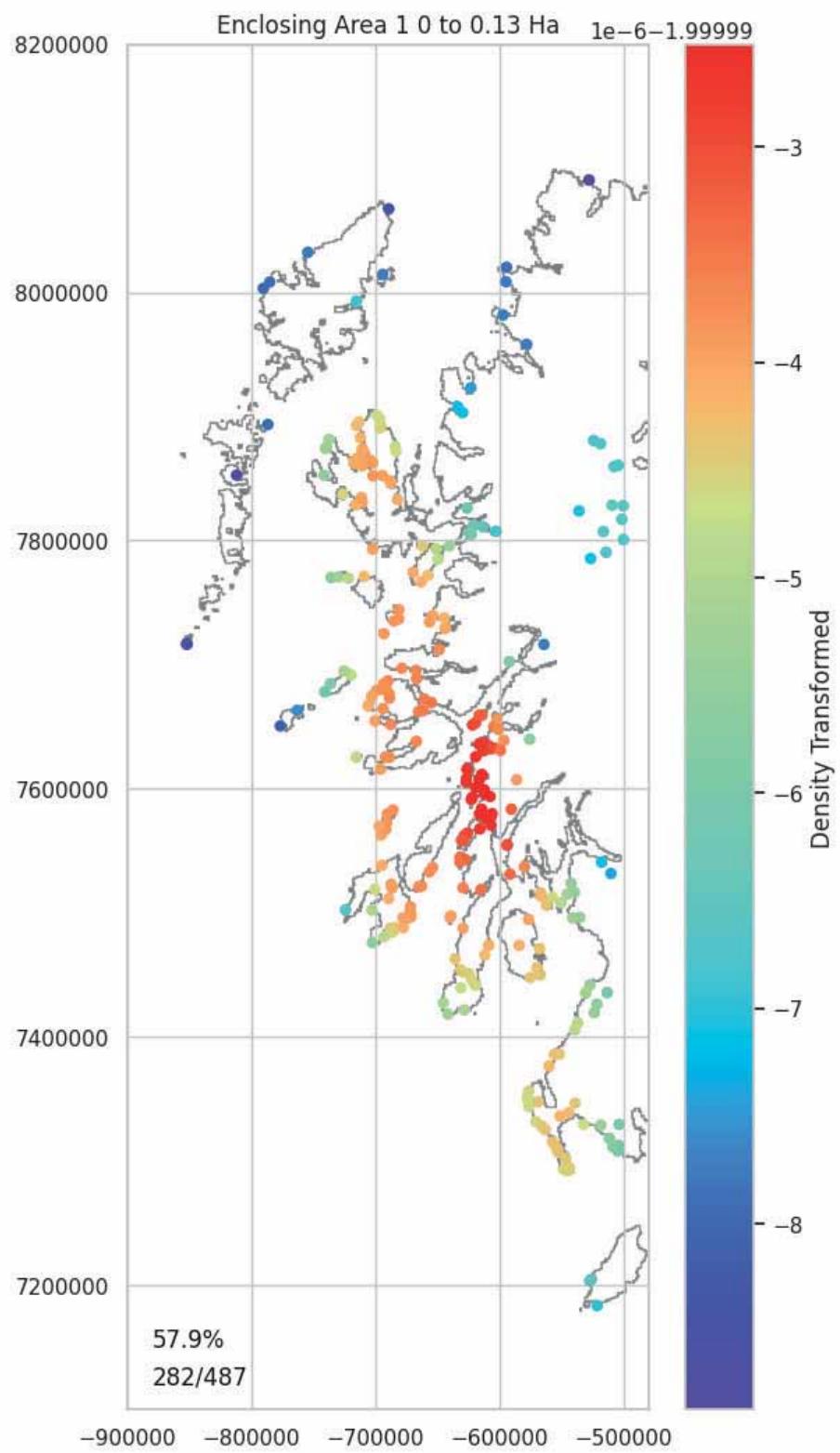
```
In [100...]  
def plot_enclosing_area_1(hillforts_data, north_west, numeric_features, show_percentage):  
    numeric_data = hillforts_data[numeric_features].copy()  
    location_data = pd.merge(north_west, numeric_data, \br/>                             left_index=True, right_index=True)  
    print(len(location_data))  
    ranges = [[0, 0.13], [0.13, 1], [1, 50]]  
    for rng in ranges:  
        #Filter data  
        cluster = location_data[location_data['Encl osing_Area_1'].between(rng[0], rng[1])]  
        new_col_name = 'Encl osing_Area_1' + "_" + str(rng[0]) + "_to_" + str(rng[1]) + "_Ha"  
        cluster = cluster.rename(columns={'Encl osing_Area_1': new_col_name})  
        show_el don = False  
        if show_percentage < 10:  
            show_percentage = 10  
  
        if (len(cluster) / len(location_data)) * 100 > show_percentage:  
            # refresh density  
            cluster = renew_densi ty(cluster)  
            location_X_cluster = \  
            plot_data_range_plus(cluster['Location_X'], '', None)  
            location_Y_cluster = \  
            plot_data_range_plus(cluster['Location_Y'], '', None)  
            #print(len(cluster))  
            plot_north_west_densi ty_for_col umn(show_el don, location_data, \  
                                         cluster, new_col_name, location_X_cluster, \  
                                         location_Y_cluster, False)
```

Only Enclosing Area 1 will be used. See Hillforts Primer Part 5.

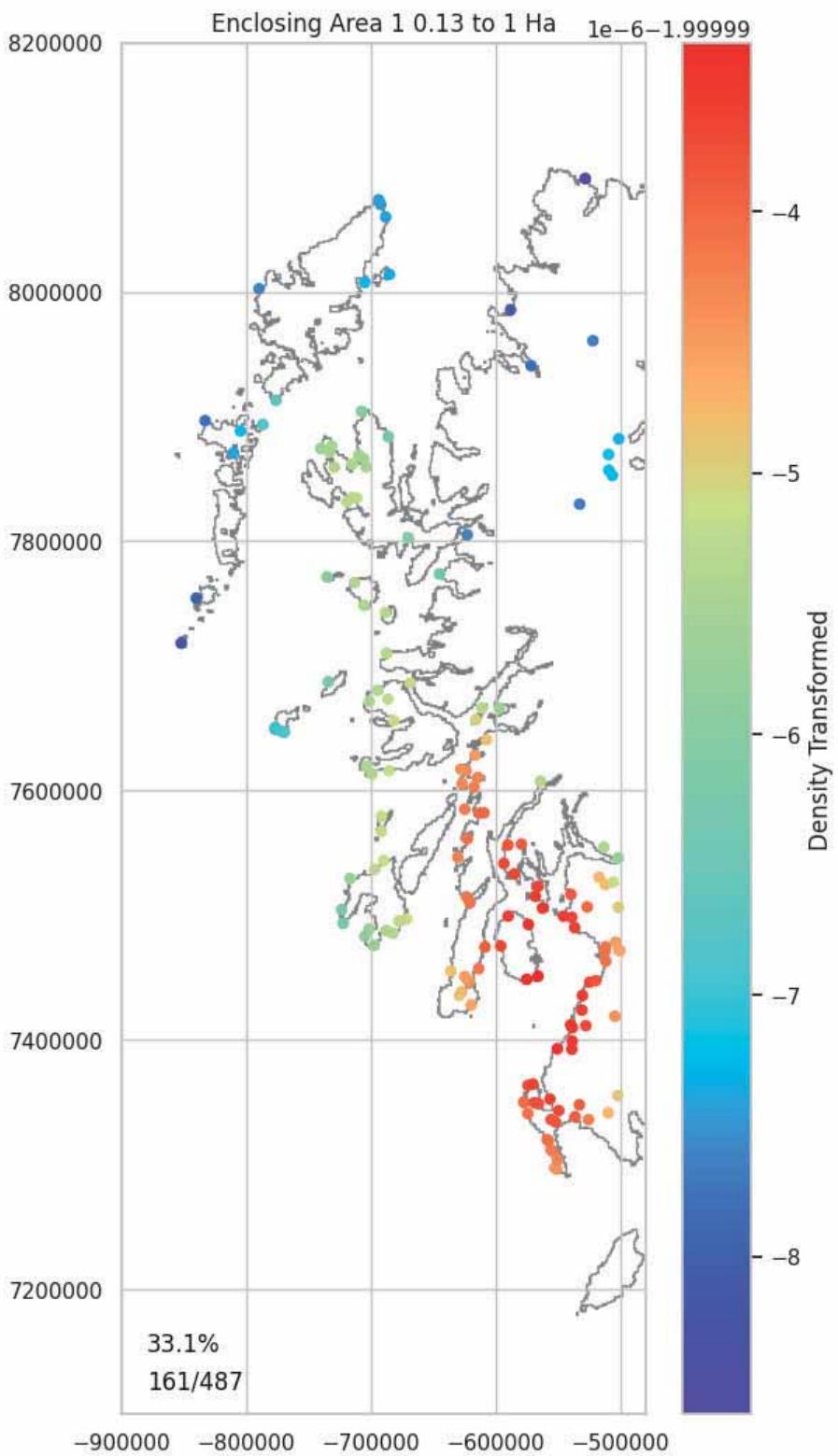
```
In [101...]  
enclosing_numeric_features_part1 = [  
    'Encl osing_Area_1',  
    #'Encl osing_Area_2',  
    #'Encl osing_Area_3',  
    #'Encl osing_Area_4',  
    #'Encl osing_Enclosed_Area',  
    #'Encl osing_Area',  
]
```

```
In [102...]  
plot_enclosing_area_1(hillforts_data, north_west, enclosing_numeric_features_part1, show_percentage)
```

487



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

```
In [103...]: print(enclosing_numeric_features_part1)
```

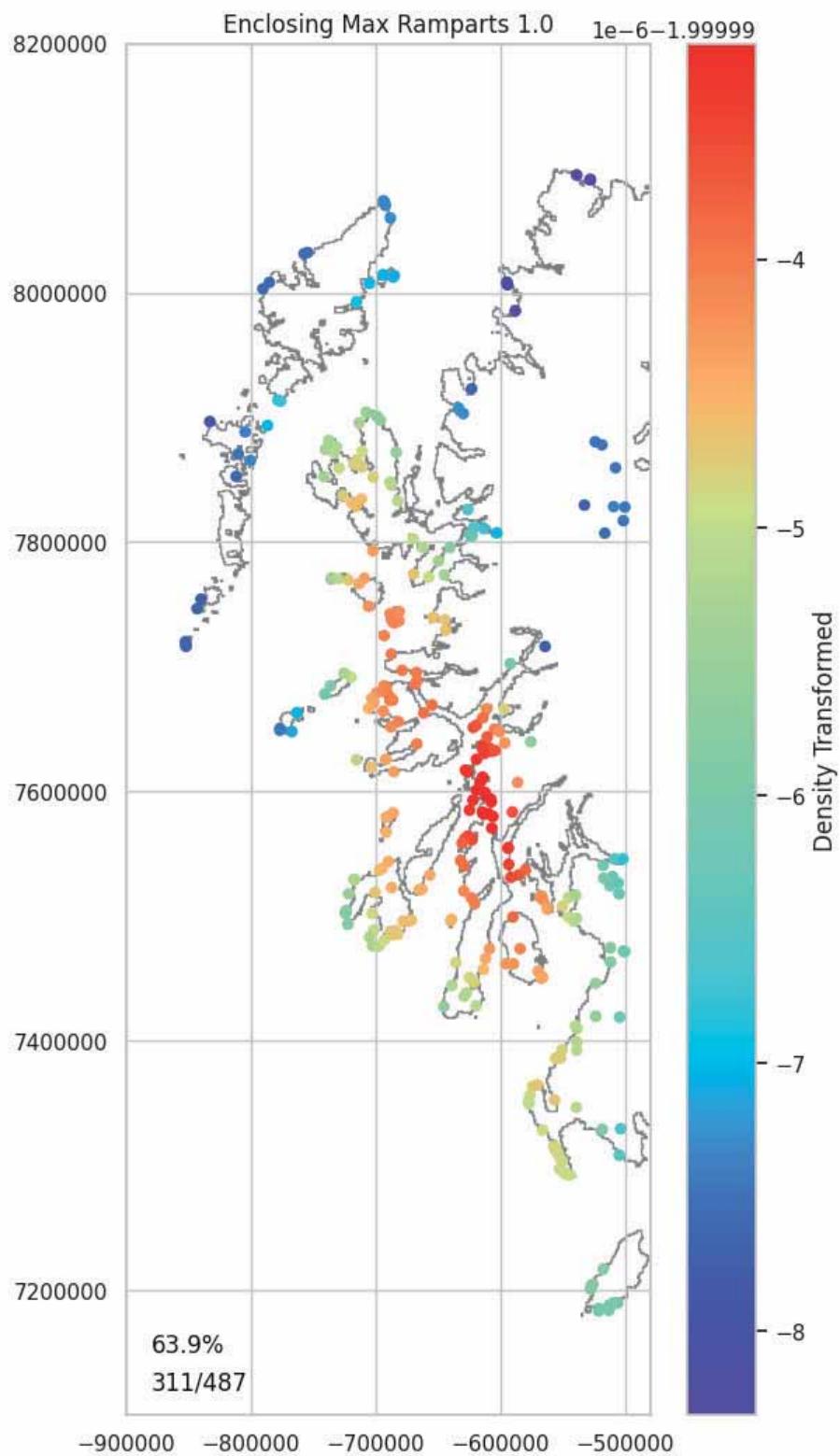
```
[ 'Enclosing_Area_1' ]
```

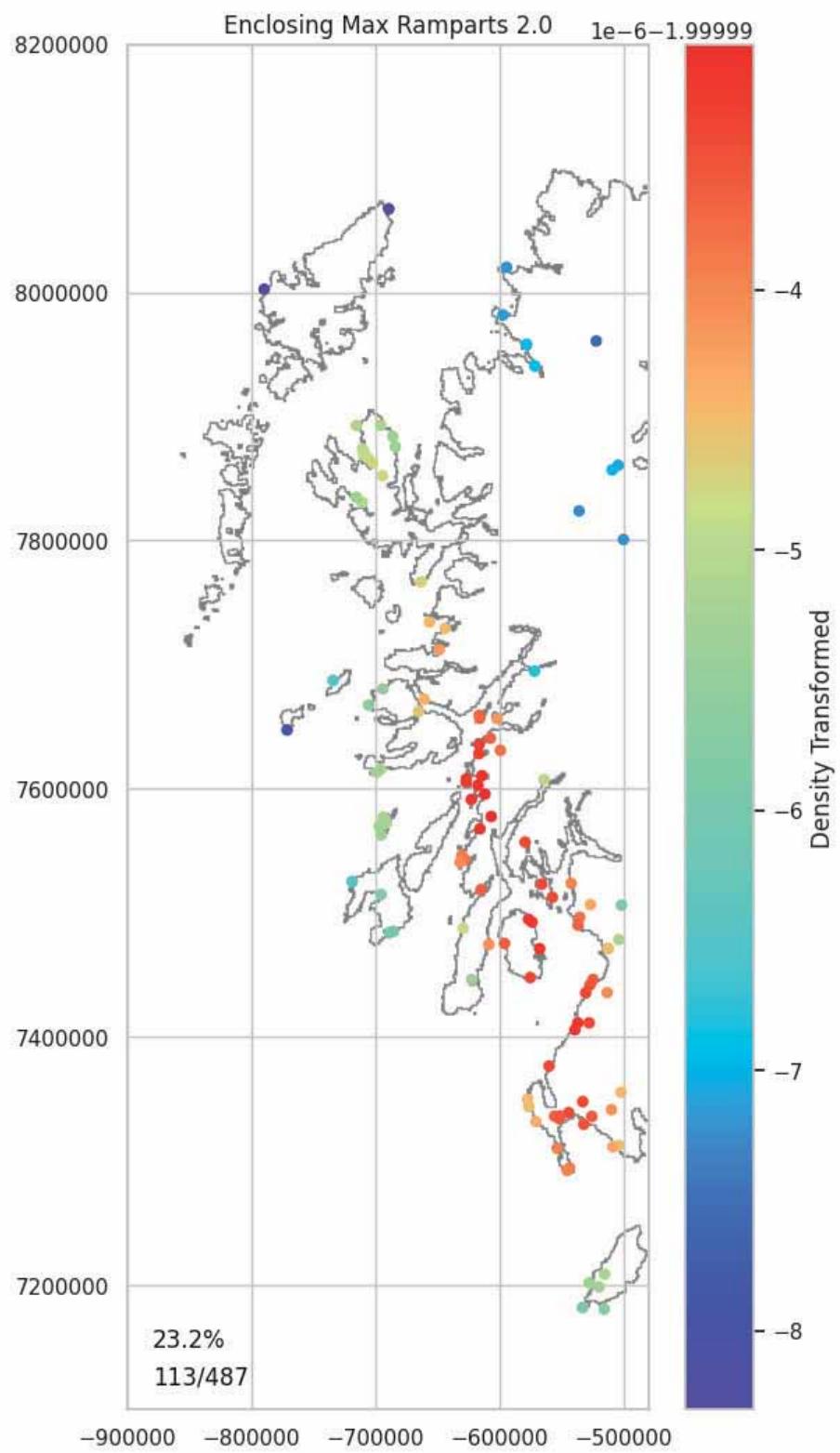
```
In [104...]: enclosing_numeric_features_part2 = [
```

```
'Enclosing_Max_Ramparts',
'Enclosing_NE_Quadrant',
'Enclosing_SE_Quadrant',
'Enclosing_SW_Quadrant',
'Enclosing_NW_Quadrant',
'Enclosing_Ditches_Number']
```

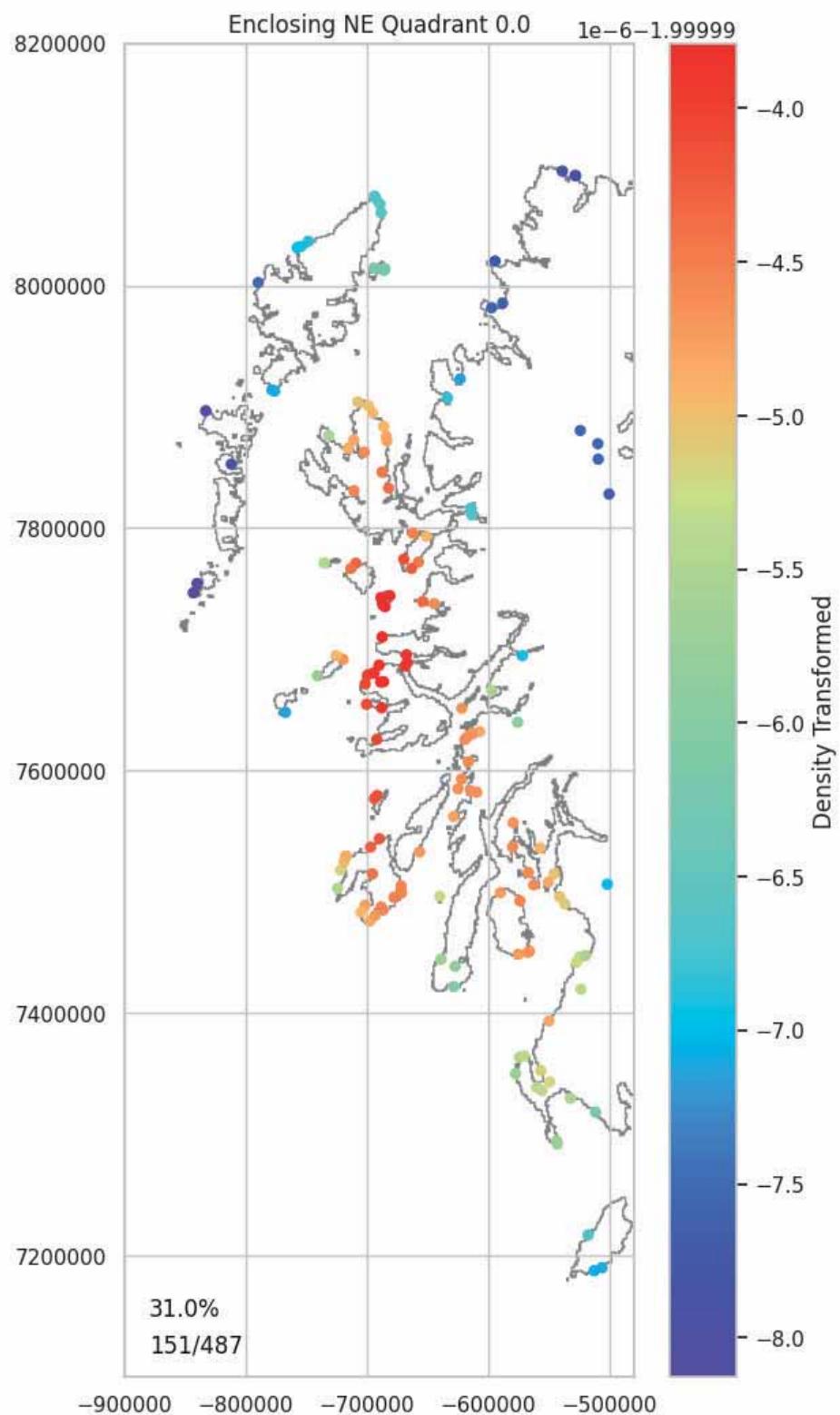
```
In [105...]: plot_numeric(hillforts_data, north_west, enclosing_numeric_features_part2, show_percentage)
```

487
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0]

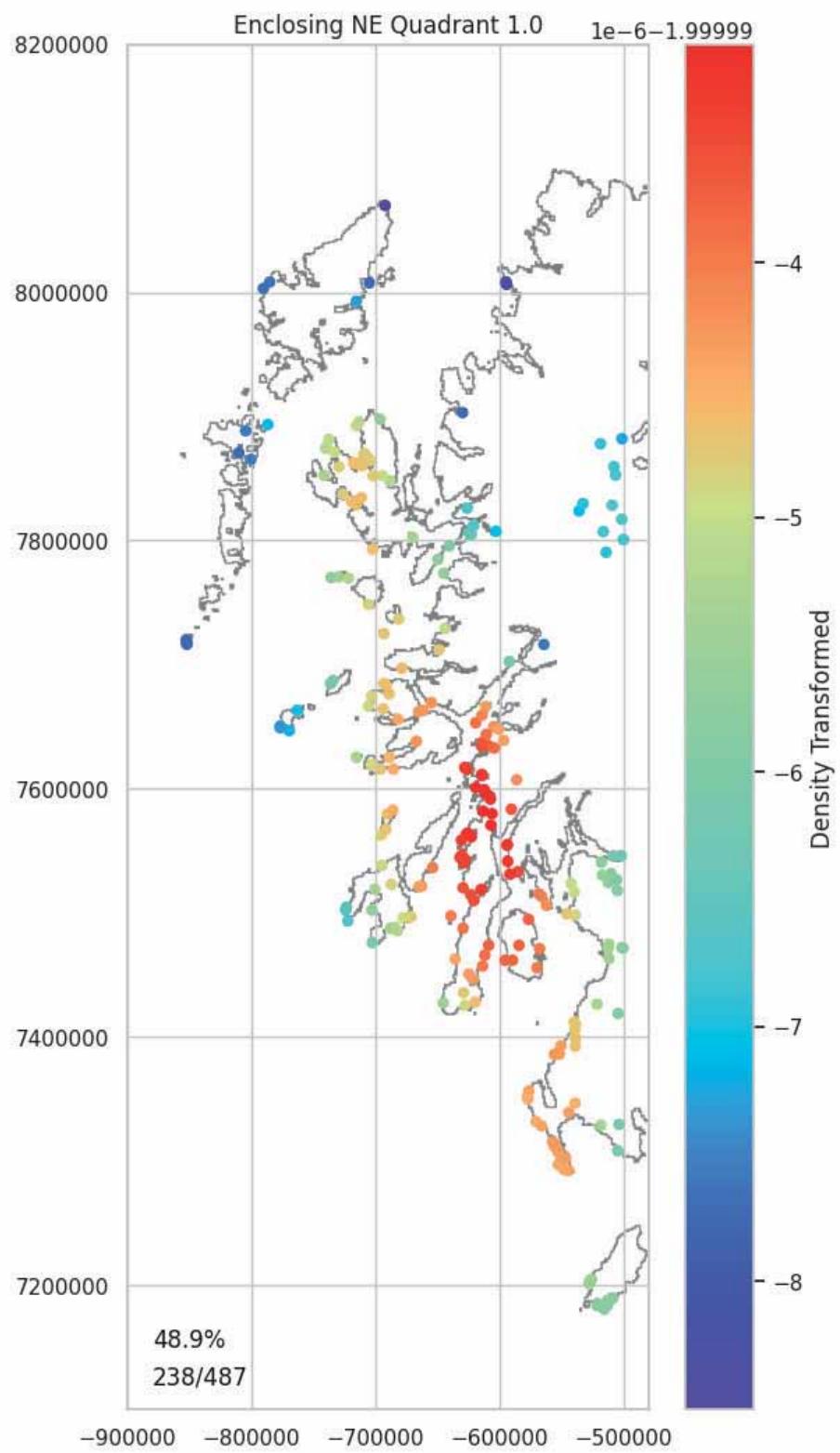




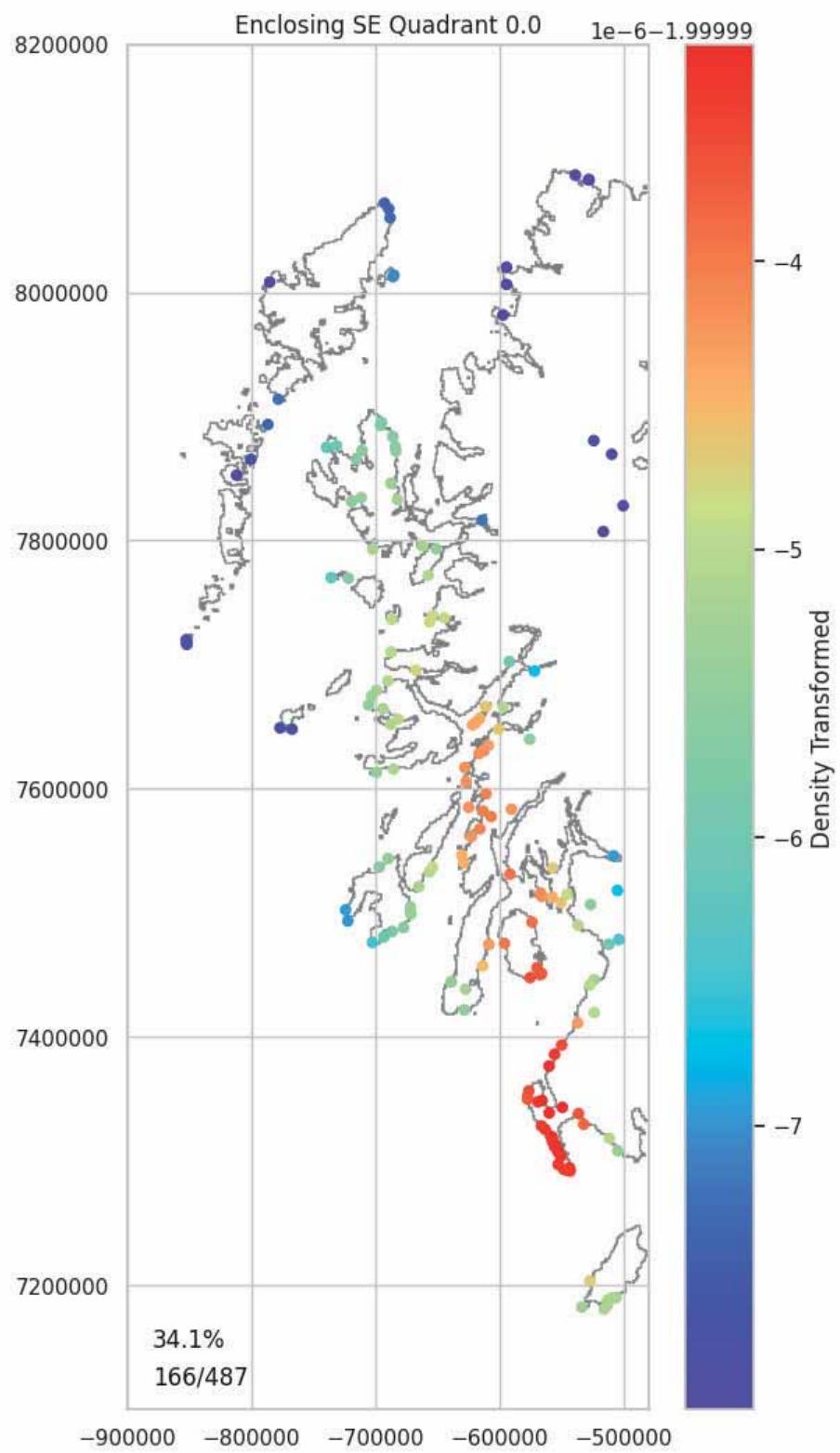
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0]



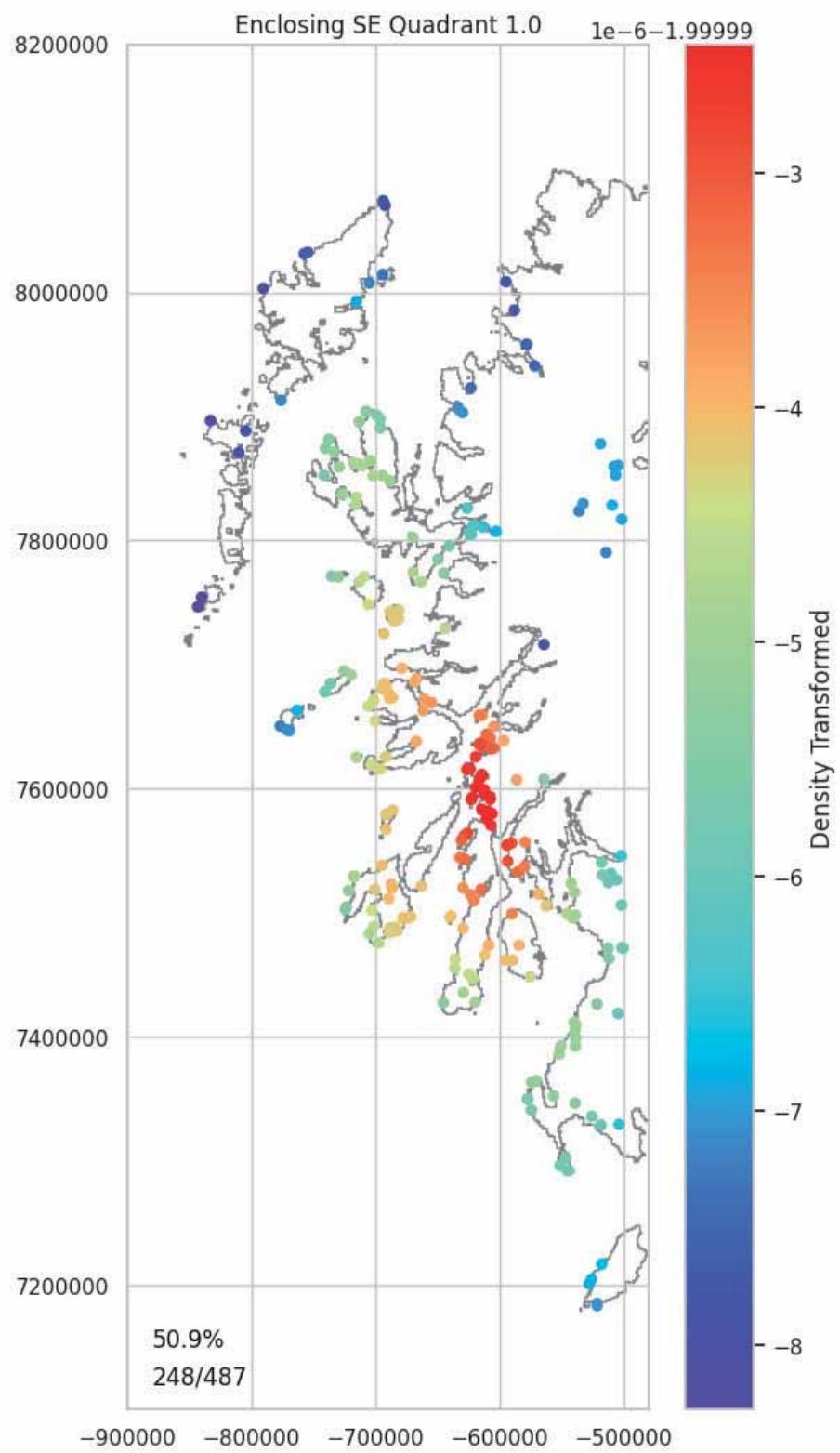
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



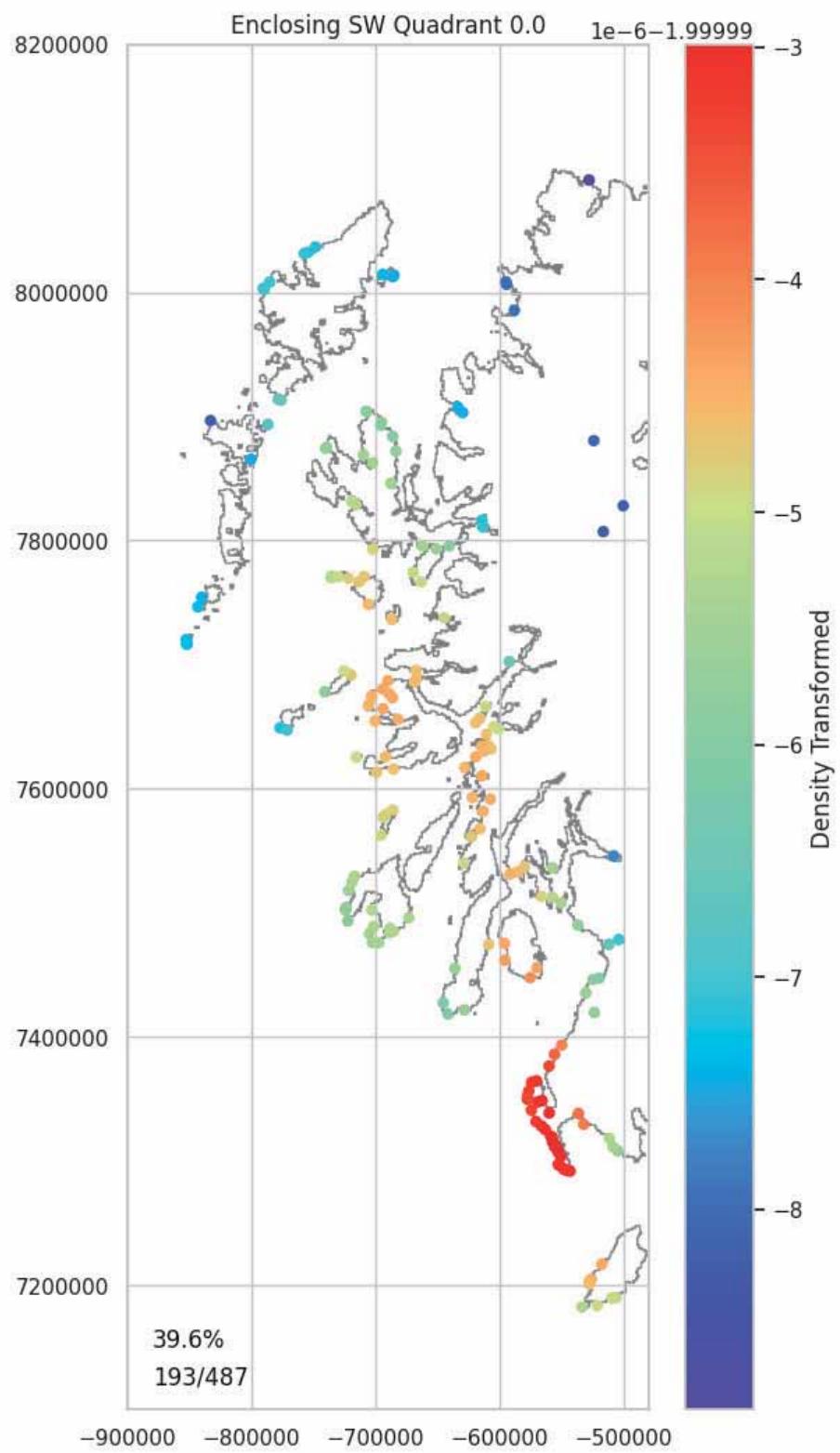
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk
[0.0, 1.0, 2.0, 3.0, 4.0]

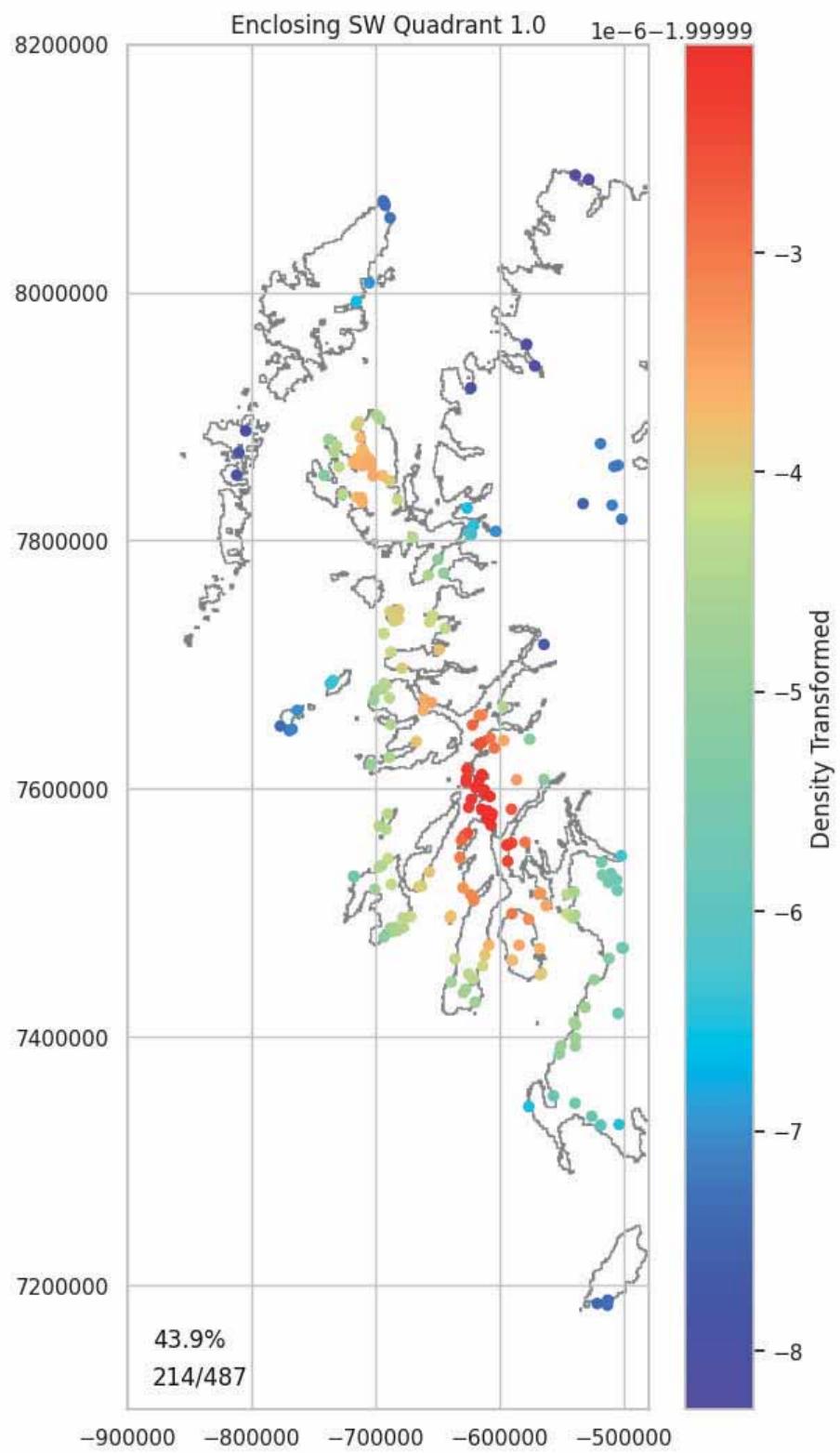


Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

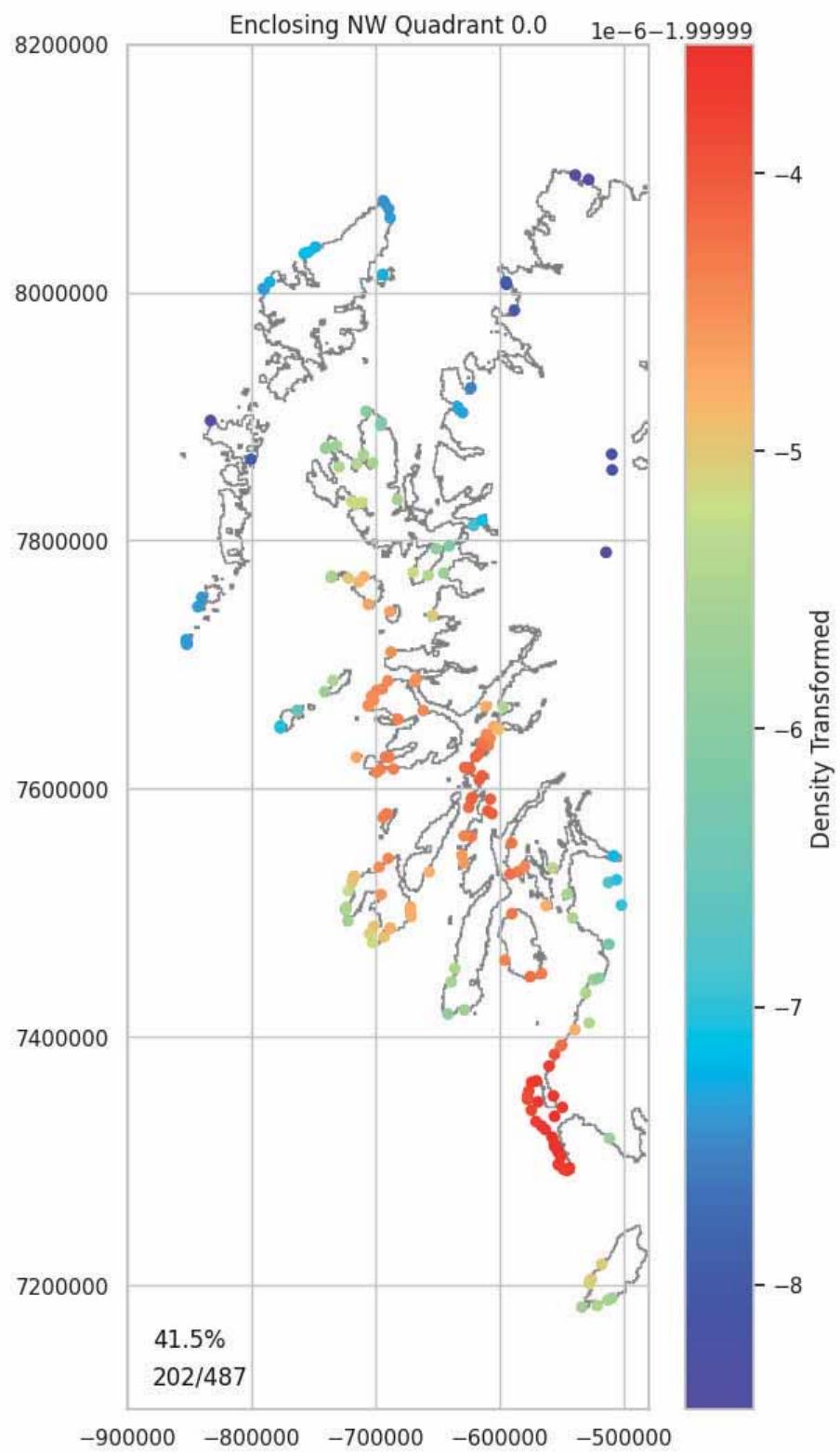


Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0]

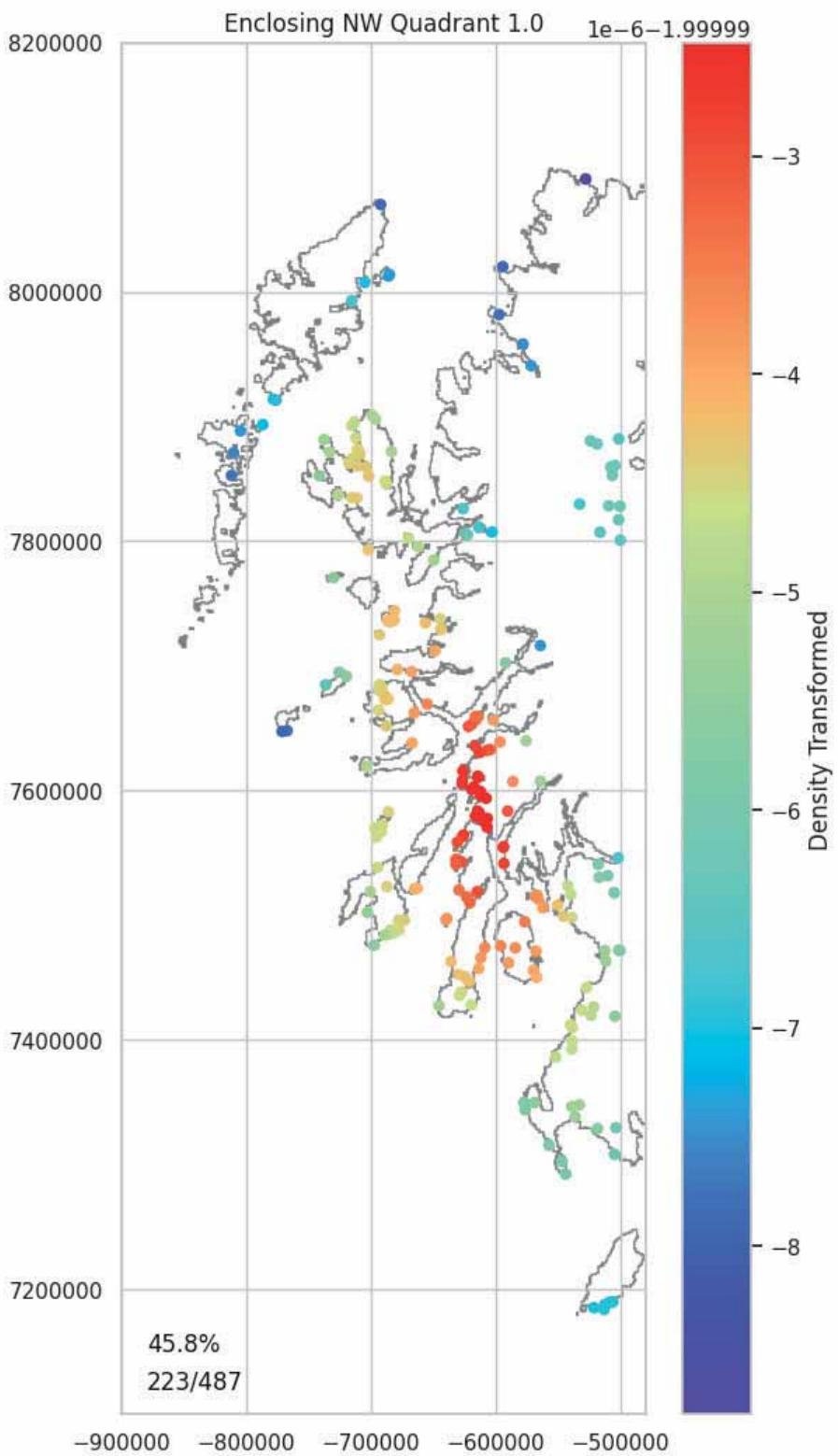




Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk
[0.0, 1.0, 2.0, 3.0, 4.0]



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk
 [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]

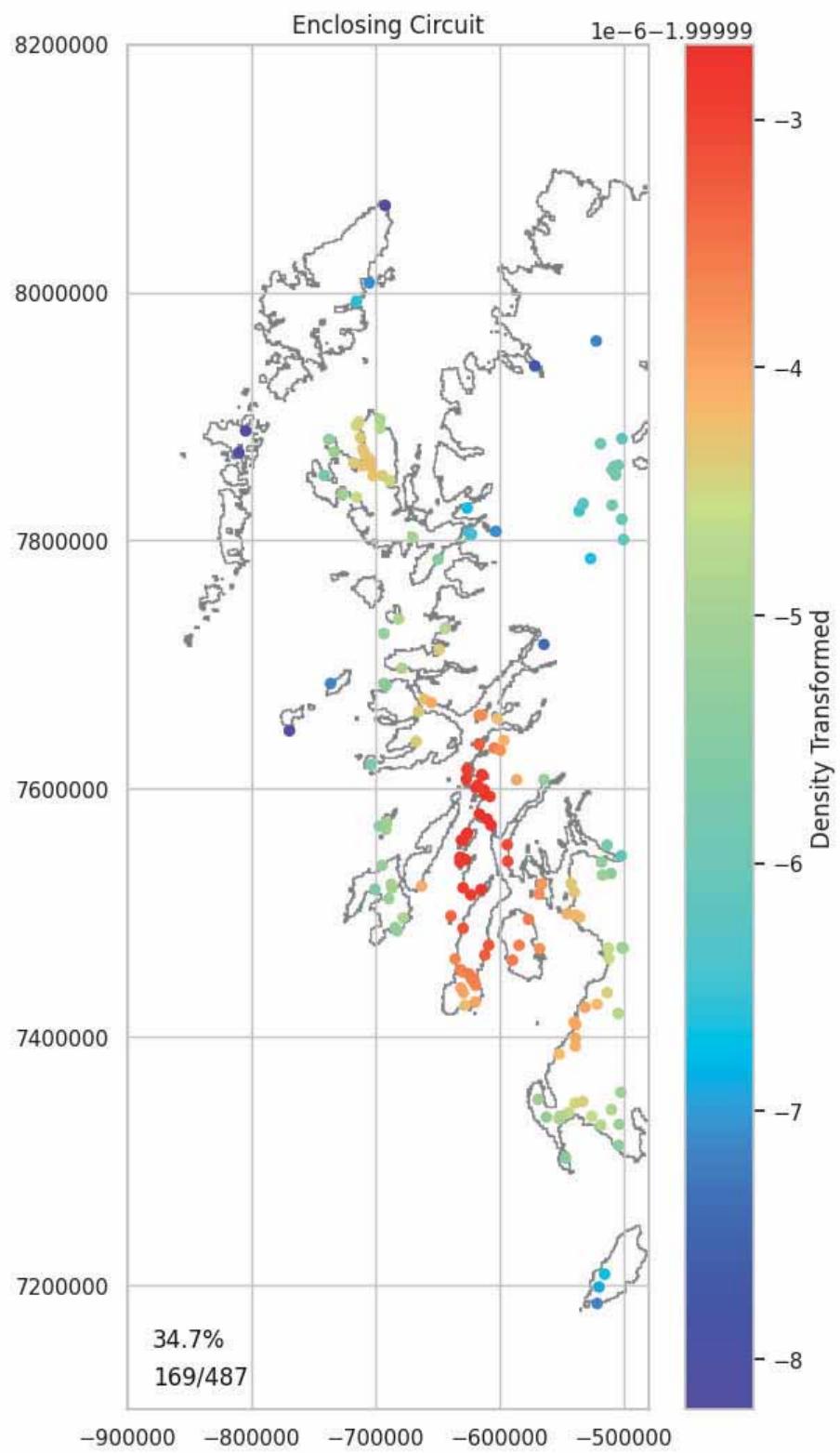
Enclosing Encodable Data

```
In [106]: encl osi ng_encodeable_features = [
  'Encl osi ng_Multi_peri od',
  'Encl osi ng_Circui t',
  'Encl osi ng_Current_Part_Uni',
  'Encl osi ng_Current_Uni',
  'Encl osi ng_Current_Part_Bi',
  'Encl osi ng_Current_Bi',
  'Encl osi ng_Current_Part_Multi',
  'Encl osi ng_Current_Multi']
```

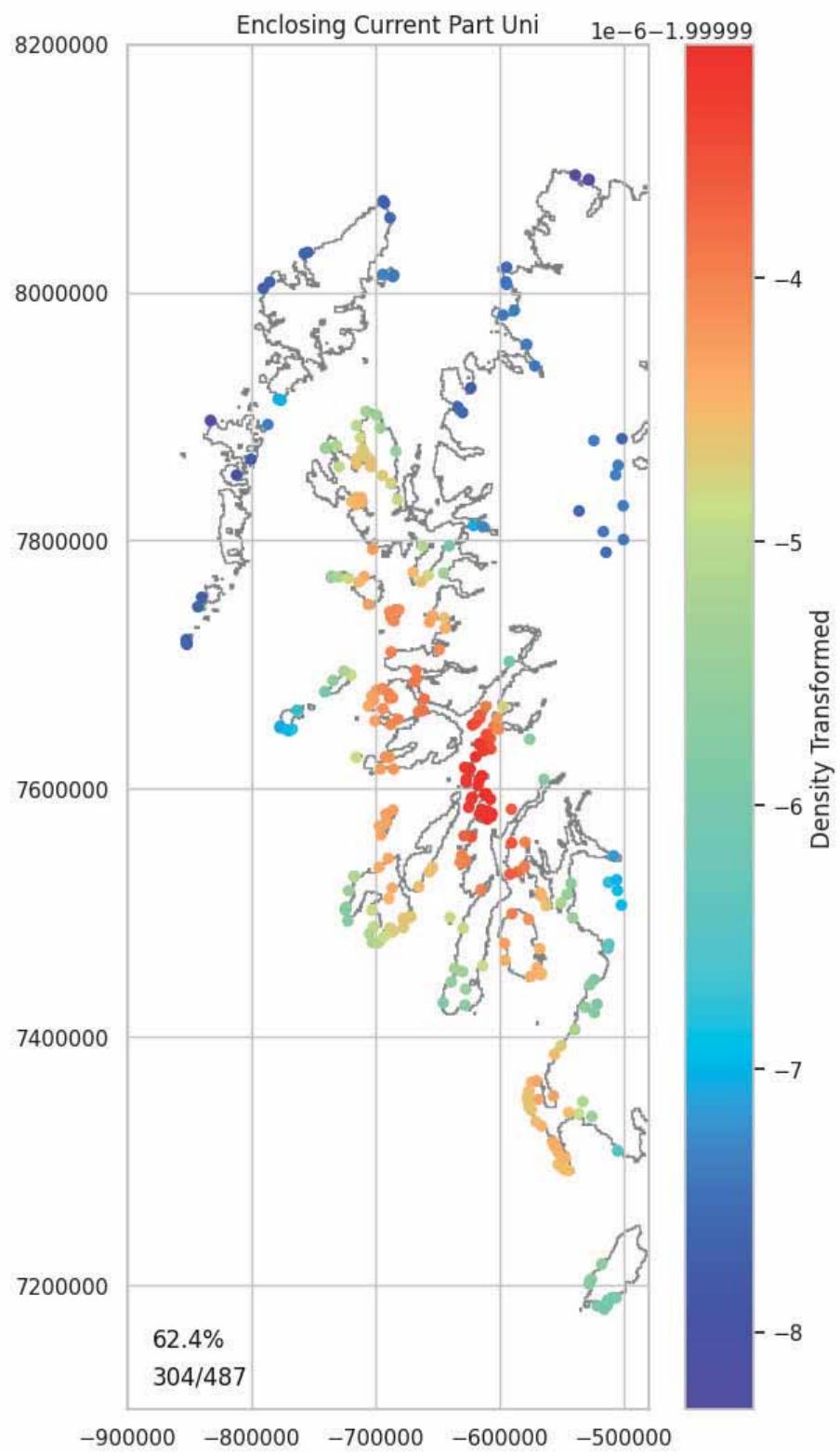
```
'Encl osi ng_Current_Unknown',
'Encl osi ng_Period_Part_Uni',
'Encl osi ng_Period_Uni',
'Encl osi ng_Period_Part_Bi',
'Encl osi ng_Period_Bi',
'Encl osi ng_Period_Part_Multi',
'Encl osi ng_Period_Multi',
'Encl osi ng_Surface_None',
'Encl osi ng_Surface_Bank',
'Encl osi ng_Surface_Wall',
'Encl osi ng_Surface_Rubble',
'Encl osi ng_Surface_Walk',
'Encl osi ng_Surface_Timber',
'Encl osi ng_Surface_Vitrification',
'Encl osi ng_Surface_Burning',
'Encl osi ng_Surface_Palisade',
'Encl osi ng_Surface_Counter_Scarp',
'Encl osi ng_Surface_Berm',
'Encl osi ng_Surface_Unfinished',
'Encl osi ng_Surface_Other',
'Encl osi ng_Excavation_Nothing',
'Encl osi ng_Excavation_Bank',
'Encl osi ng_Excavation_Wall',
'Encl osi ng_Excavation_Murus',
'Encl osi ng_Excavation_Timber_Framed',
'Encl osi ng_Excavation_Timber_Laced',
'Encl osi ng_Excavation_Vitrification',
'Encl osi ng_Excavation_Burning',
'Encl osi ng_Excavation_Palisade',
'Encl osi ng_Excavation_Counter_Scarp',
'Encl osi ng_Excavation_Berm',
'Encl osi ng_Excavation_Unfinished',
'Encl osi ng_Excavation_No_Known',
'Encl osi ng_Excavation_Other',
'Encl osi ng_Gang_Working',
'Encl osi ng_Ditches']
```

```
In [107]: plot_encodeable(hills_forts_data, north_west, enclosing_encodeable_features, show_percentage)
```

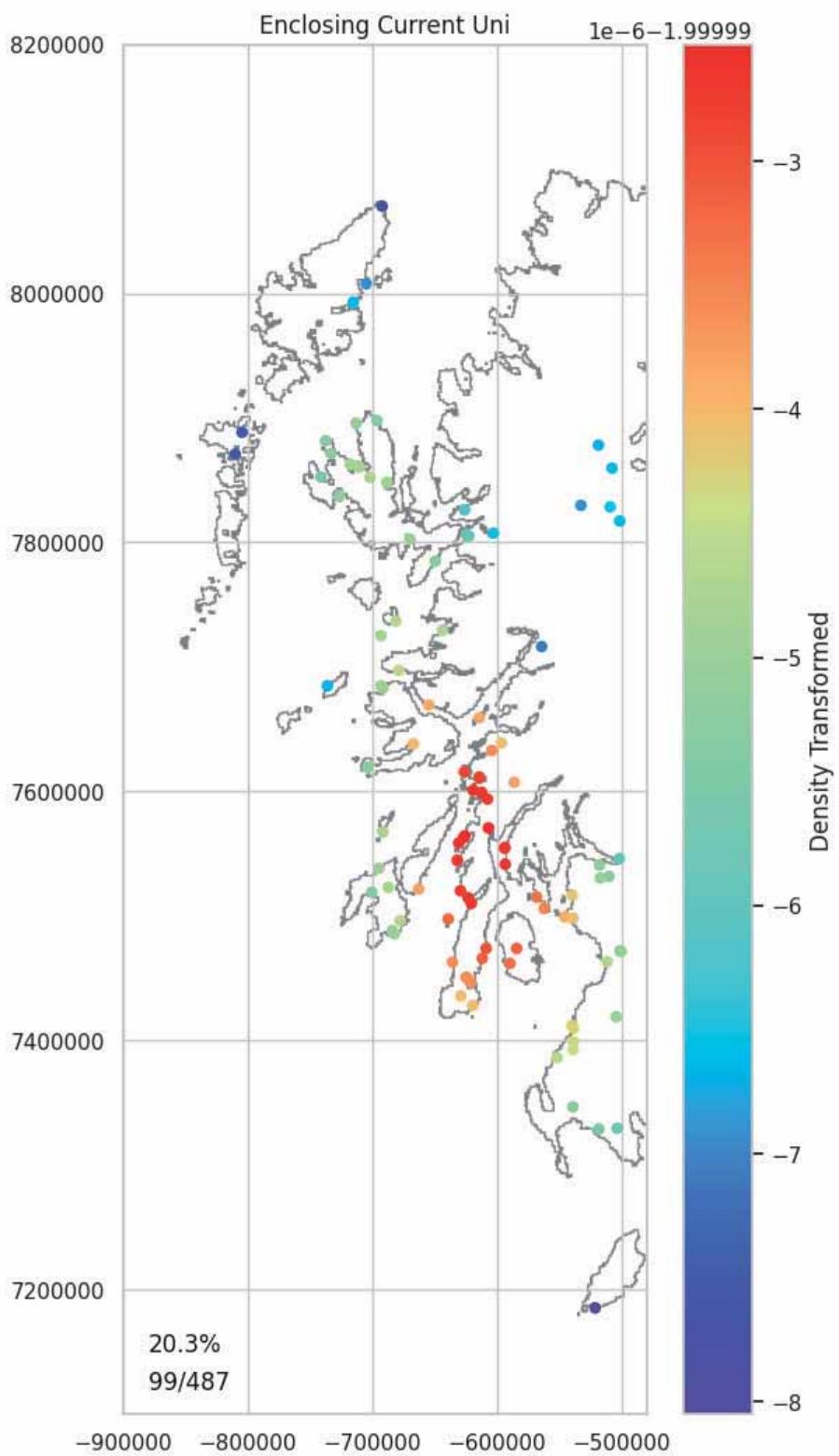
487



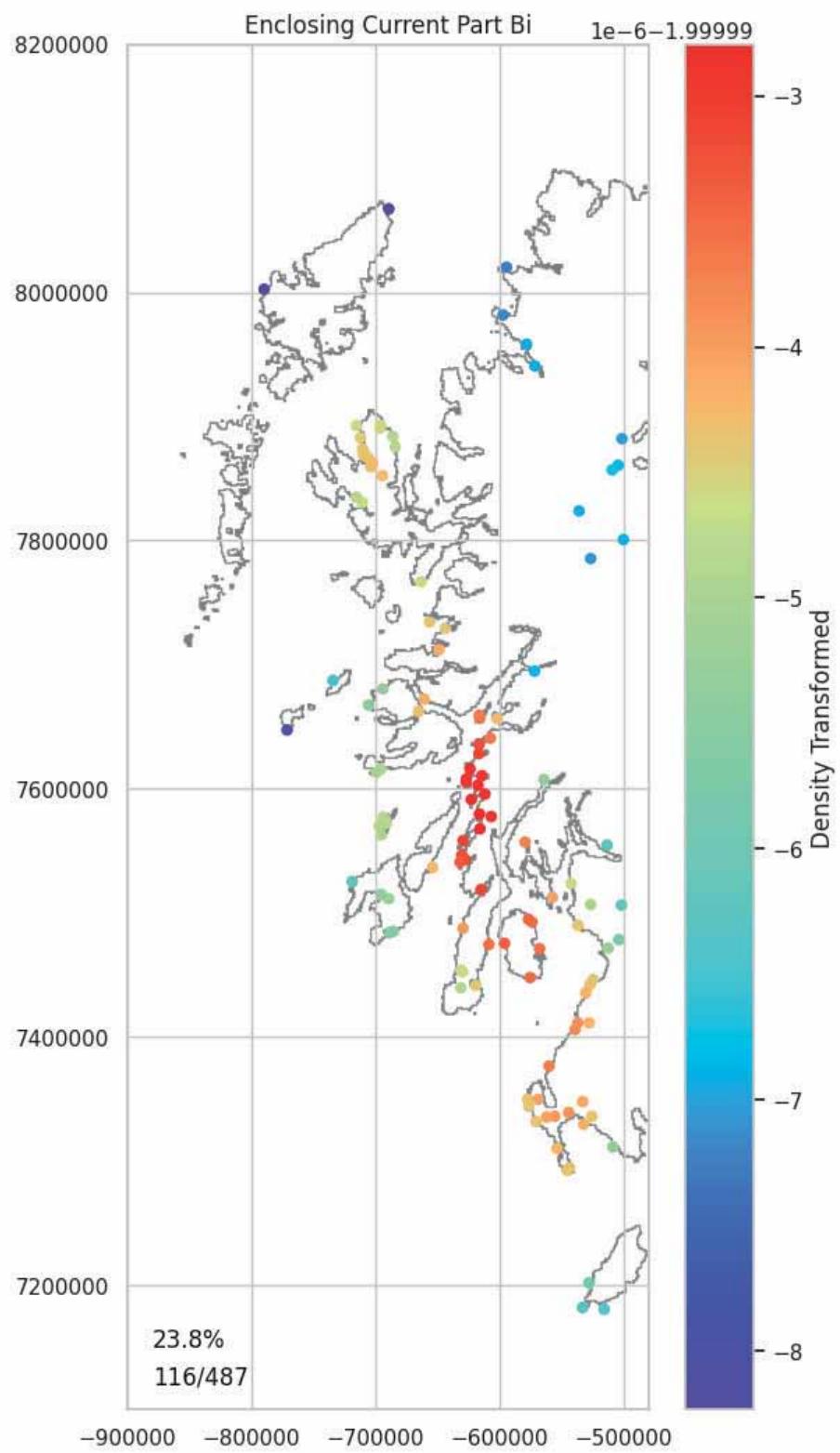
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



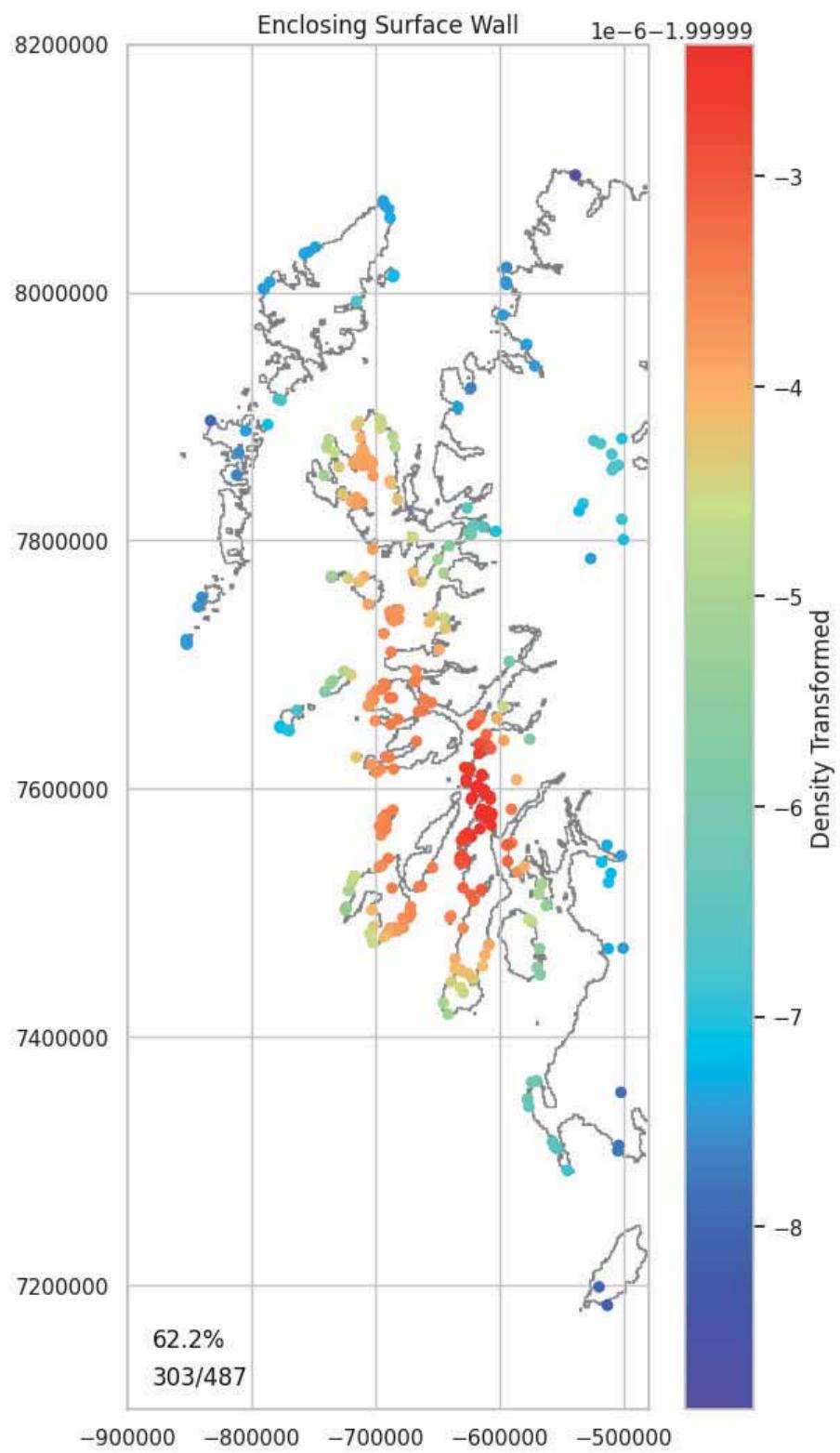
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



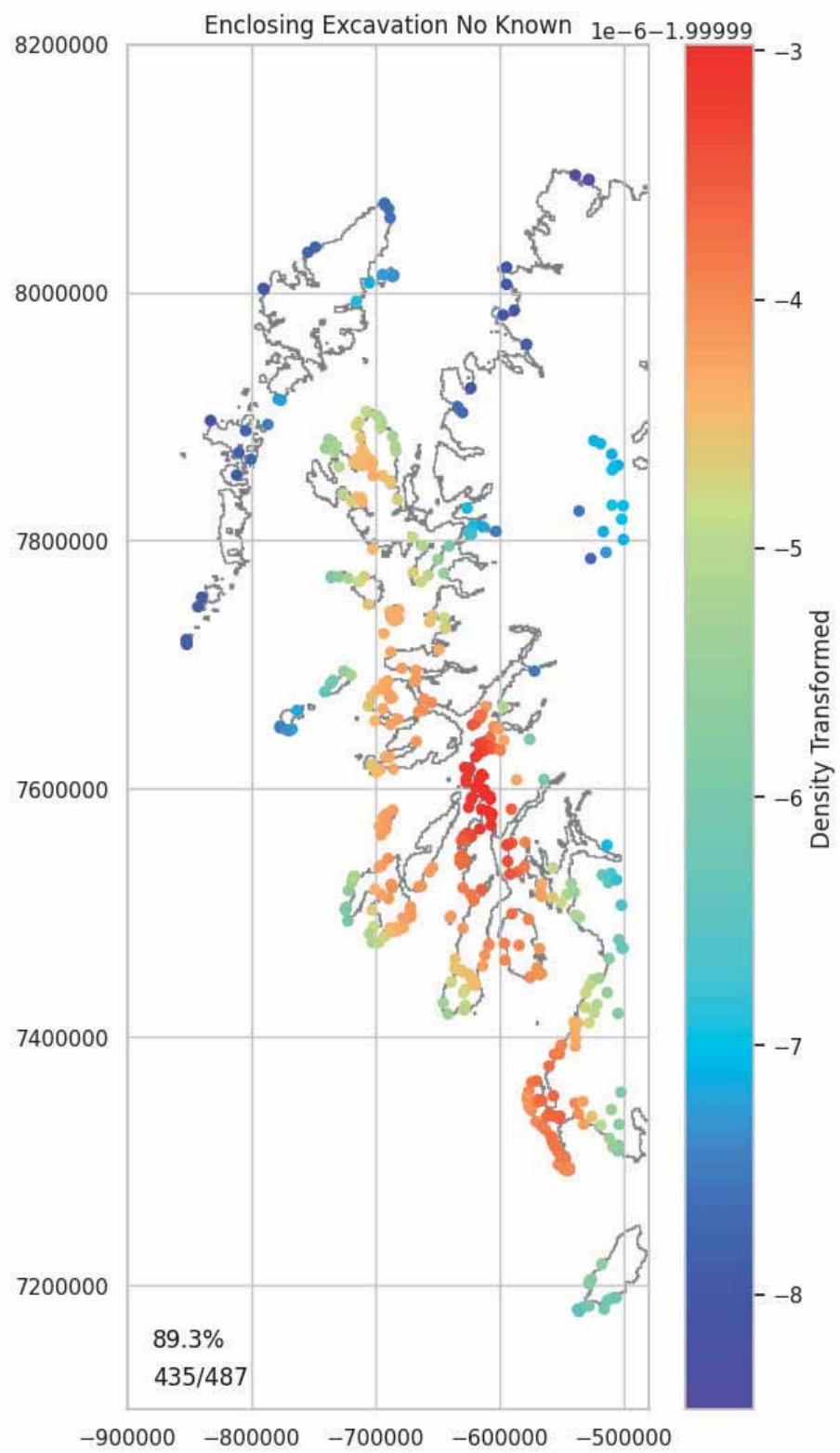
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



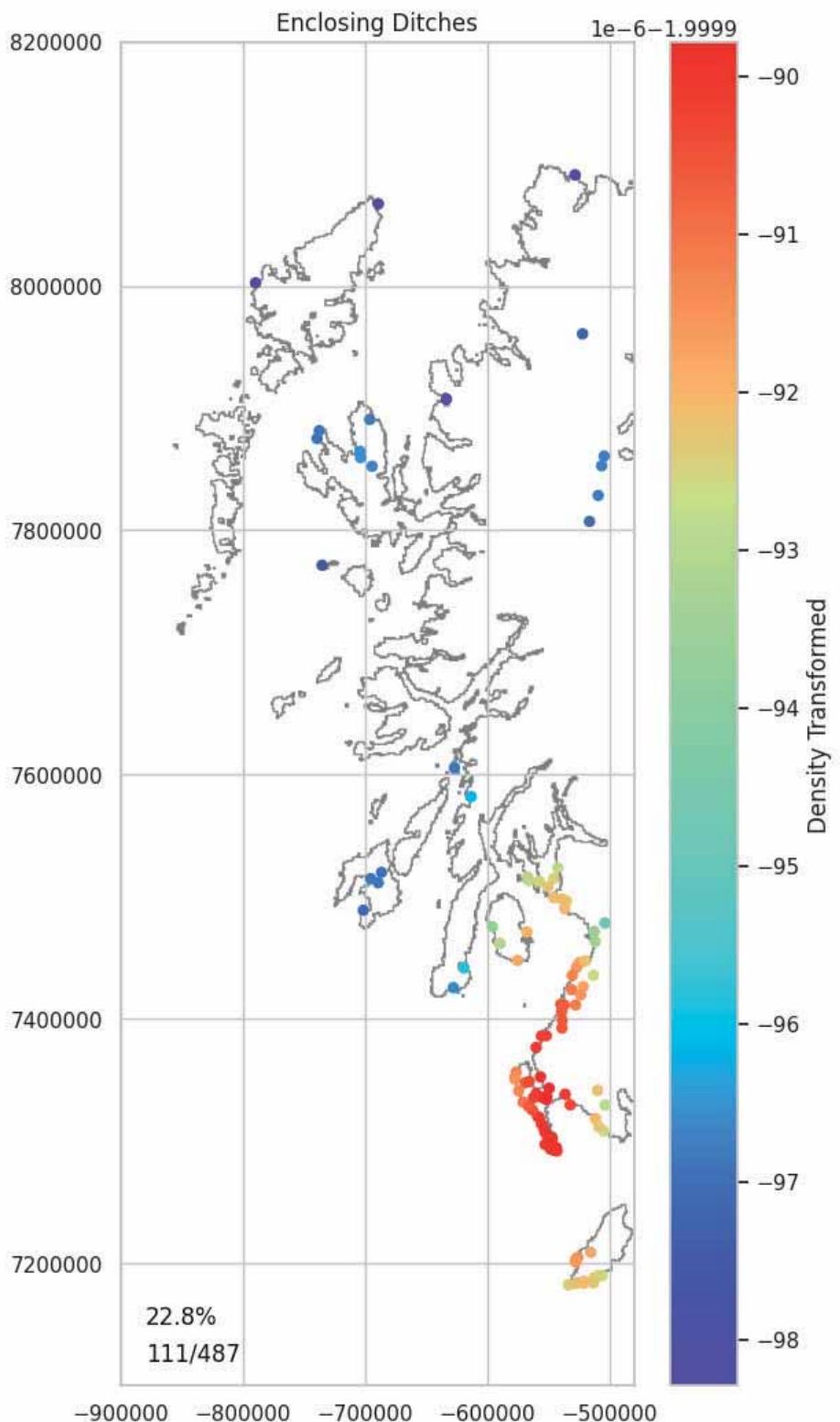
Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk



Middleton, M. 2024, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Annex Encodable Data

```
In [108]: annex_encodeable_features = ['Annex']
```

```
In [109]: plot_encodeable(hillforts_data, north_west, annex_encodeable_features, show_percentage)
```

487

Dating Data Download Package

If you do not wish to download the data using this document, all the processed data packages, notebooks and images are available here:

<https://github.com/MikeDairsie/Hillforts-Primer>.

```
In [110]: #download(dating_data_list, 'Dating_package')
```

Save Figure List

```
In [111]: if save_images:  
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")  
    fig_list.to_csv(path, index=False)
```