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 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(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_colo
```

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/"

    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.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 [19]:
```
def plot_continuous(data, x_lable, title):
    fig = plt.figure(figsize=(12,8))
    ax = fig.add_axes([0,0,1,1])
    ax.set_xlabel(x_lable)
    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 [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.ticklabel_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.ticklabel_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=Fal
    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 [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 controlls 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.ticklabel_format(style='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 = 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 [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 = 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()
    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_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_catagorical_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_catagorical_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}')
        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/MikeDairsie/Hillforts-Primer/main/hillforts-atlas-source-data-cs
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 dt
ype 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','norhtern-ireland', 'isle-of-man
                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 uninqe reference identiriers 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()
```

Out[59]:

	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()
```

Out[60]:

	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 Irealnd', cluster_data['Clust
   )
north_ireland = cluster_data[cluster_data['Cluster'] == 'North Irealnd'].copy()

cluster_data['Cluster'] = np.where(
   (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , 'South Irealnd', cluster_data['Cluste
   )
south_ireland = cluster_data[cluster_data['Cluster'] == 'South Irealnd'].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_pack
```

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 seperate 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 carrage 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
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()
```

Out[73]:

	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	
1	None	Yes	No	No	No	No	
2	A number of cloudy blue flints, two burnt flin...	Yes	No	No	No	No	
3	Possible hut circles 12m-15m in diameter were ...	Yes	No	No	No	No	
4	At least 118 hut platforms have been identifie...	No	Yes	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()
```

Out[75]:

	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 Encoeable 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()
```

Out[79]:

	Interior_Water_None	Interior_Water_Spring	Interior_Water_Stream	Interior_Water_Pool	Interior_Water_Flush	Interior_Water_Well	Interic
0	Yes	No	No	No	No	No	
1	Yes	No	No	No	No	No	
2	Yes	No	No	No	No	No	
3	Yes	No	No	No	No	No	
4	No	Yes	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 revieled through excvation 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)
```

Out[80]:

	Interior_Water_None	Interior_Water_Spring	Interior_Water_Stream	Interior_Water_Pool	Interior_Water_Flush	Interior_Water_Well	Interic
0	Yes	No	No	No	No	No	
1	Yes	No	No	No	No	No	
2	Yes	No	No	No	No	No	
3	Yes	No	No	No	No	No	
4	No	Yes	No	No	No	No	
5	Yes	No	No	No	No	No	
6	No	No	Yes	Yes	No	No	

There a 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.

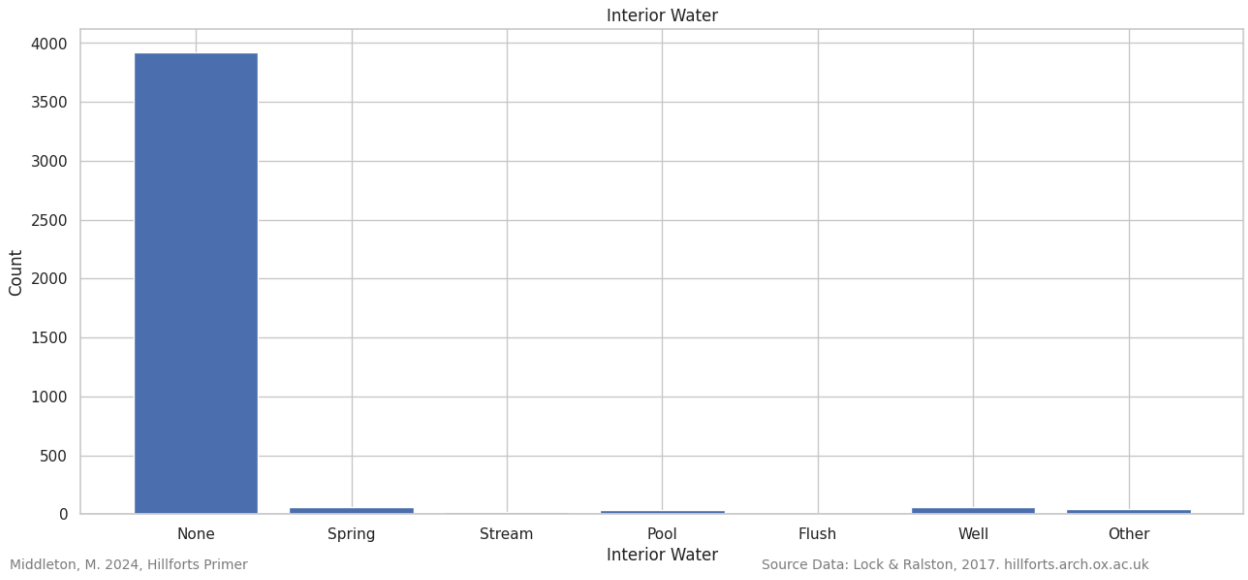
In [82]:
```
none_water = sum(water_data["Interior_Water_None"]== "Yes")
none_water
```

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()
```

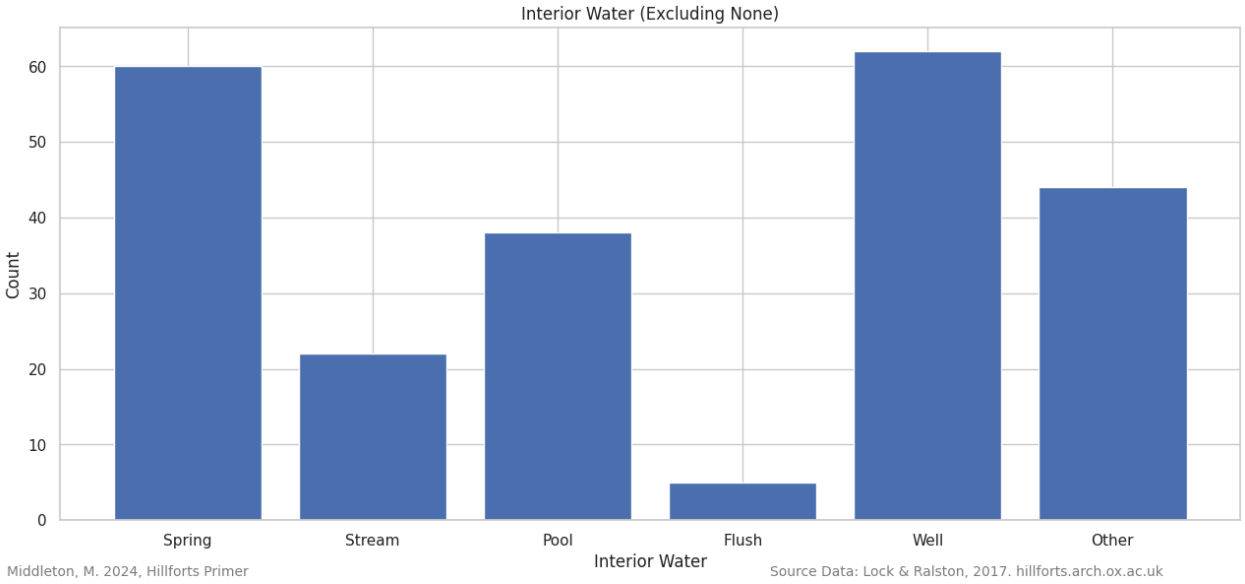
Out[85]:

	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

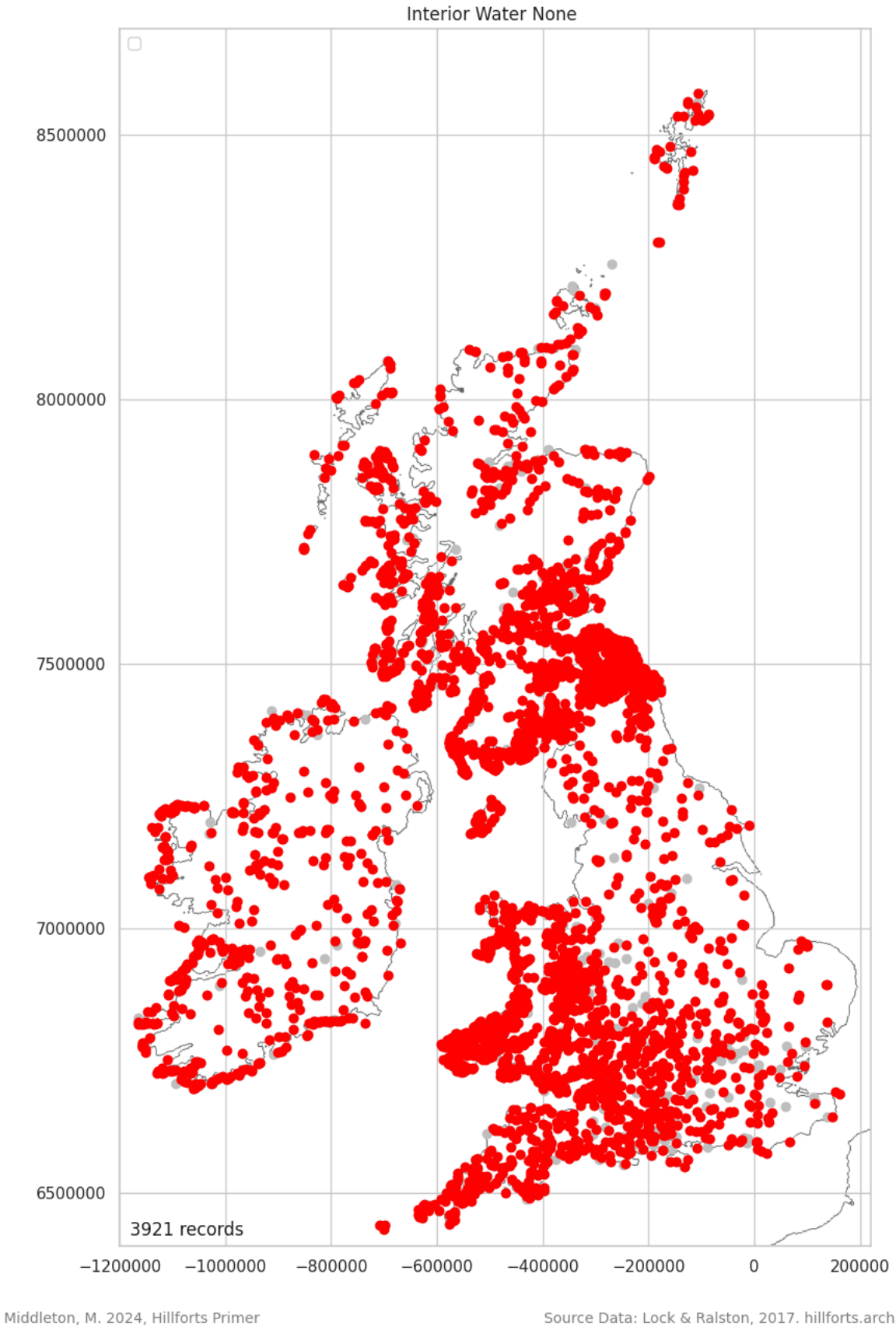
There are very few records relating to water features within hillforts. Most (94.55%) have no water features recorded.

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')
```

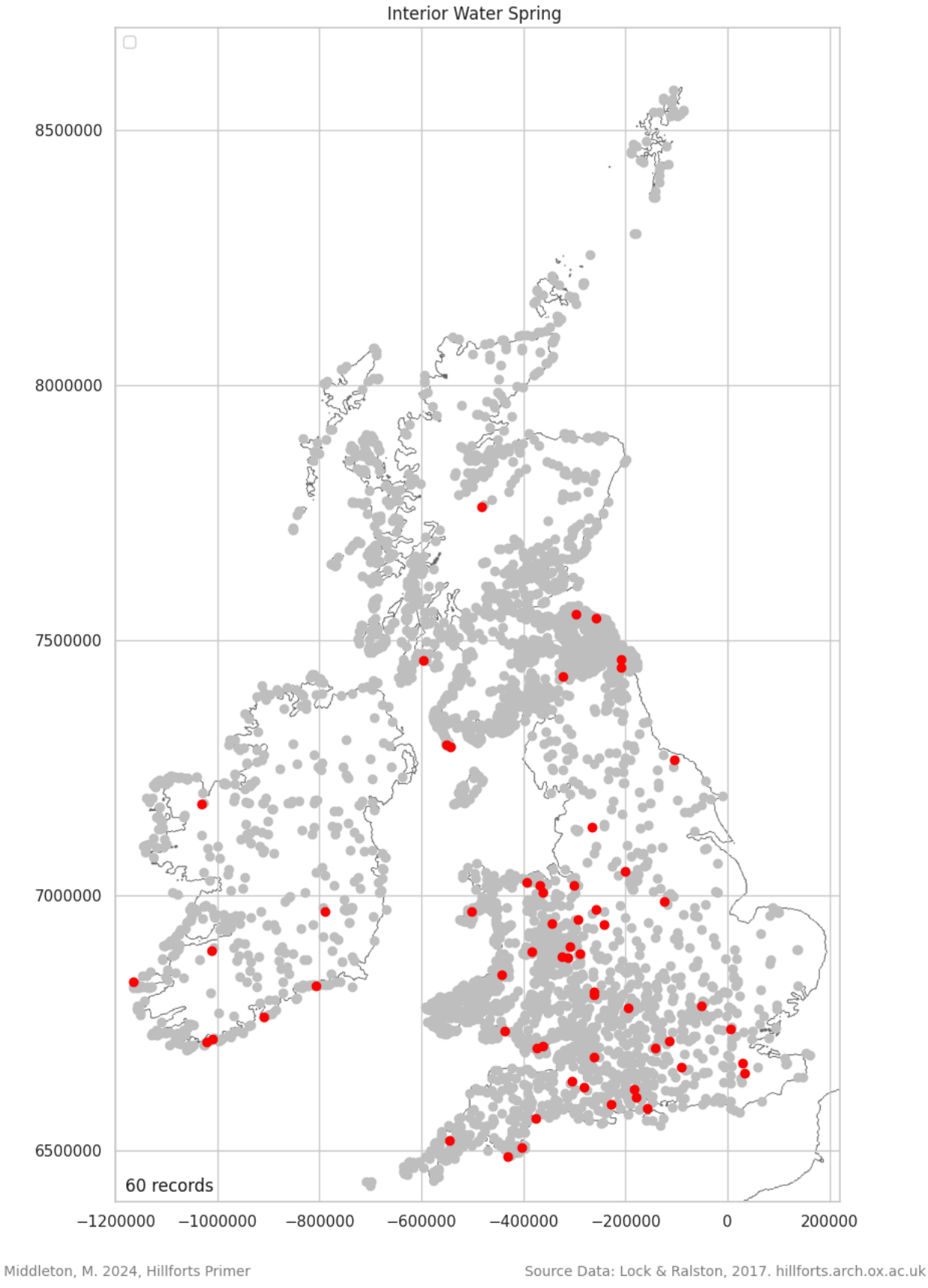


94.55%

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')
```

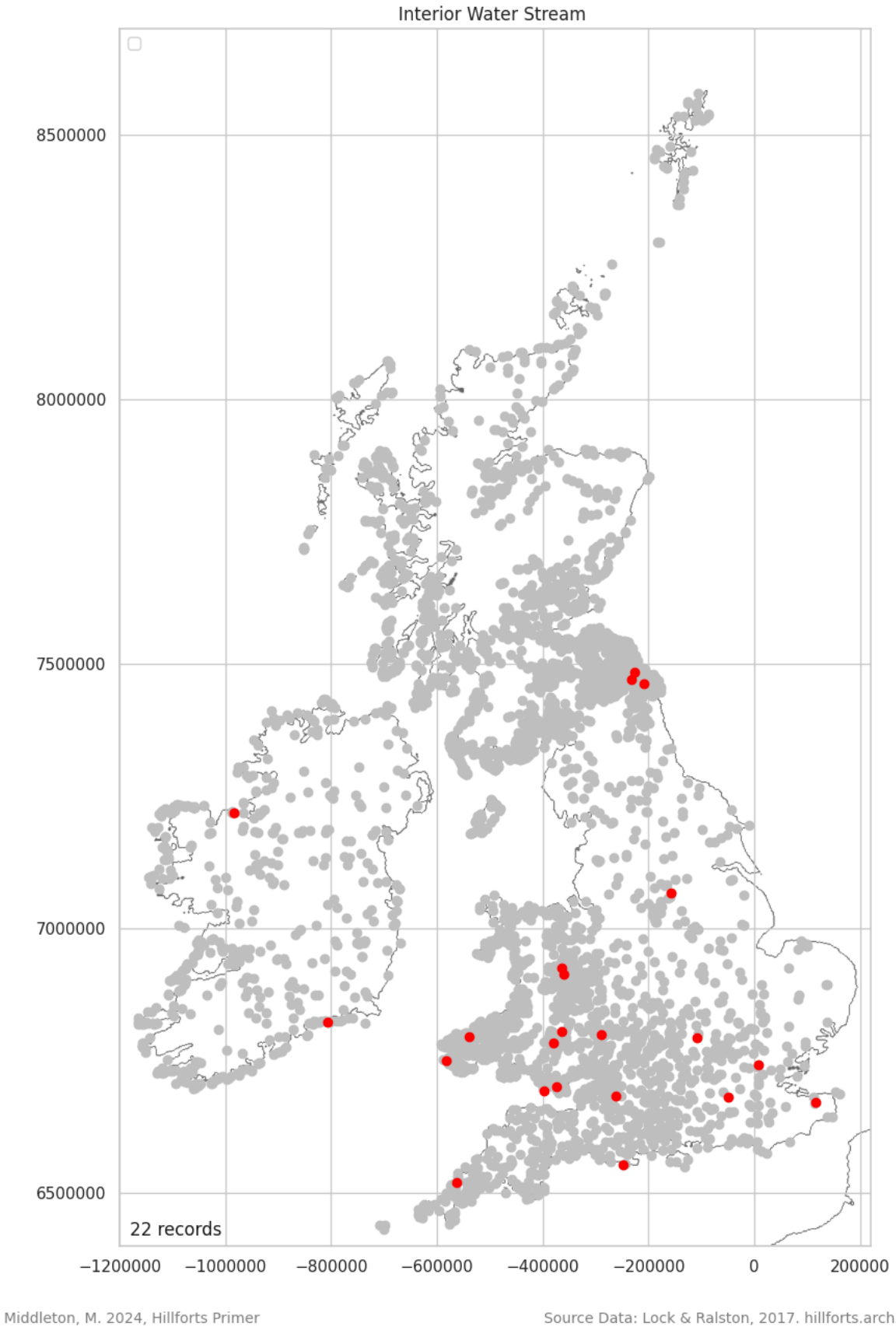


1.45%

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')
```

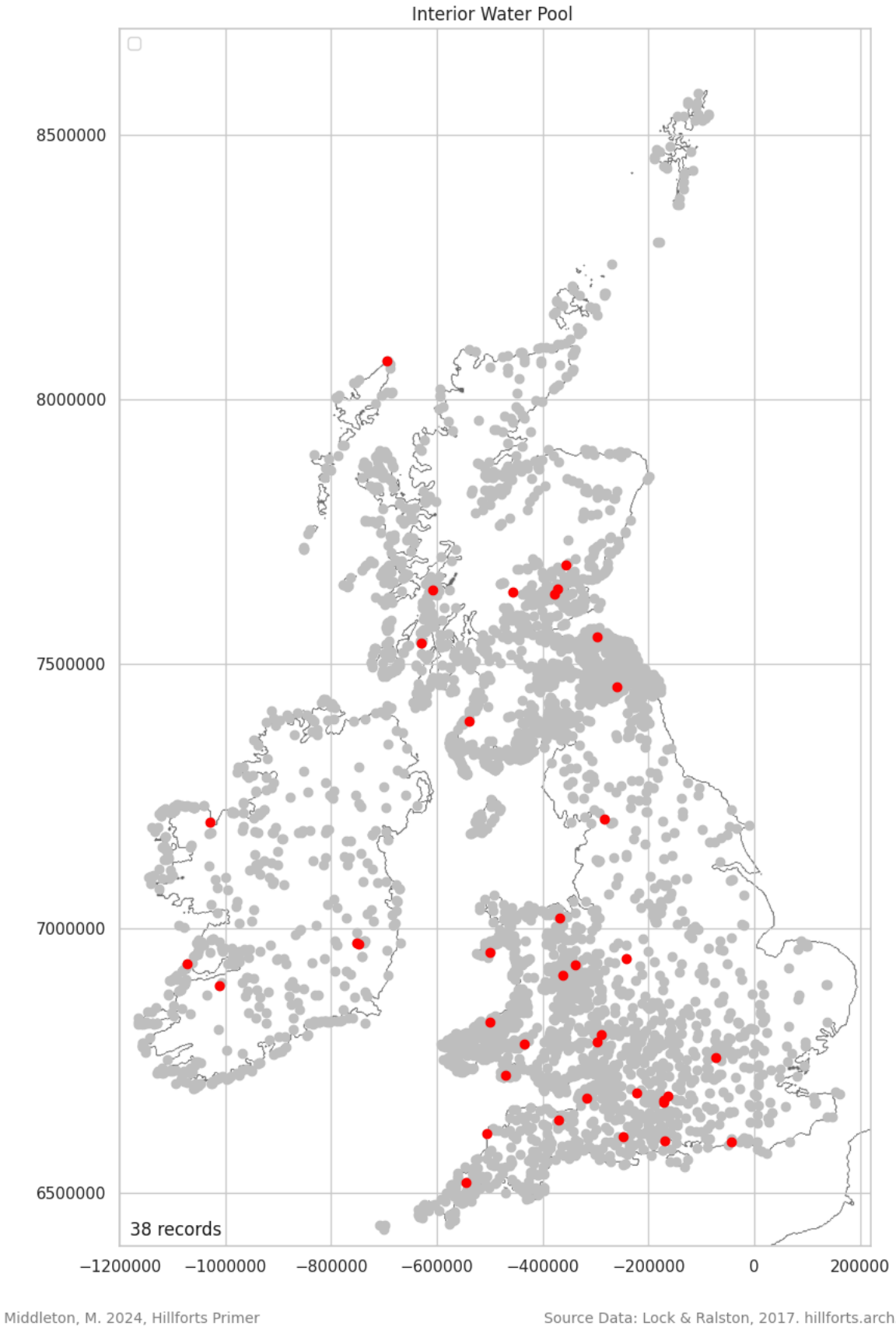


0.53%

Pool Mapped

Just 0.92% have a pool recorded withing the hillfort.

In [92]:
```
int_pool = plot_over_grey(location_water_data, 'Interior_Water_Pool', 'Yes')
```

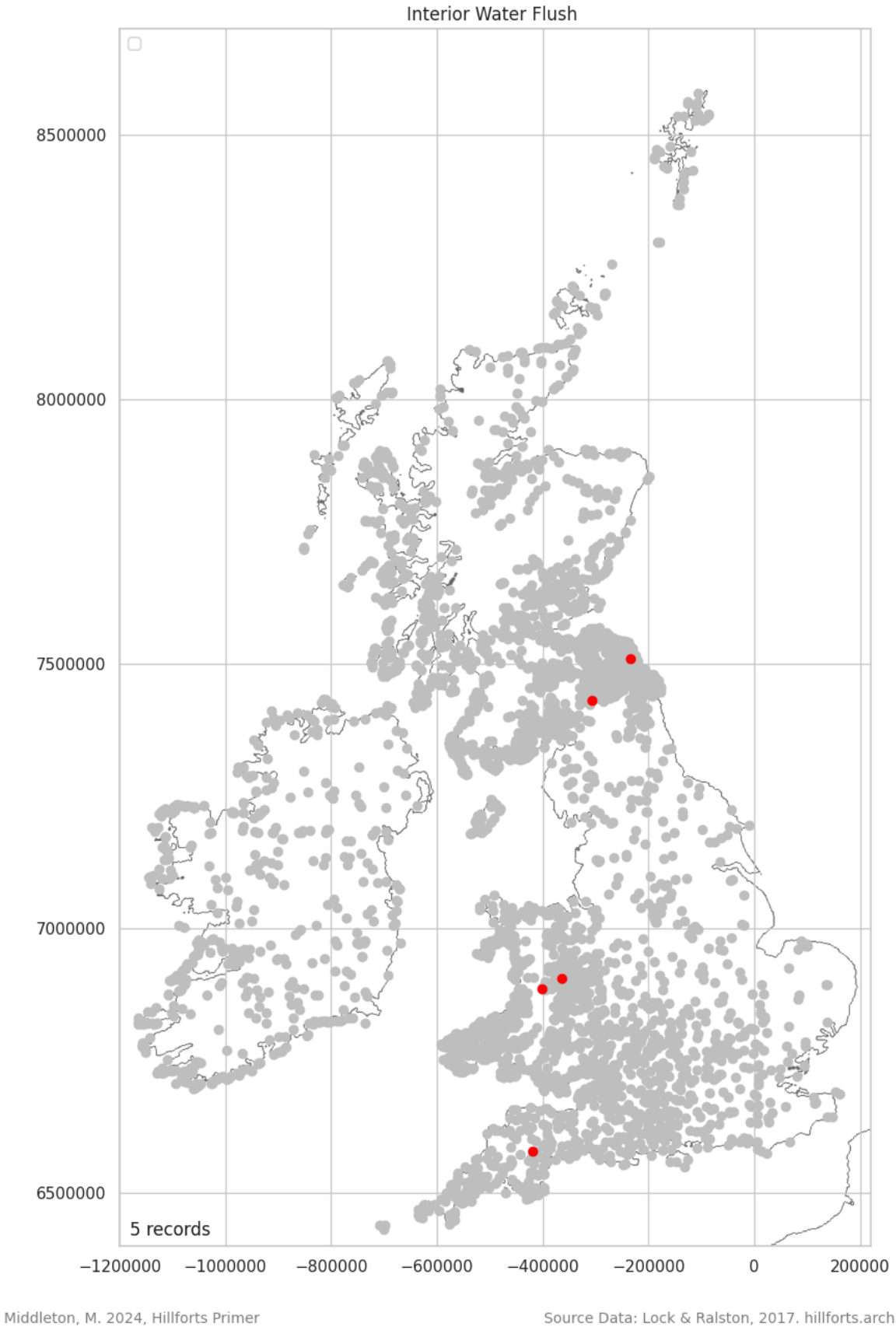


0.92%

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')
```

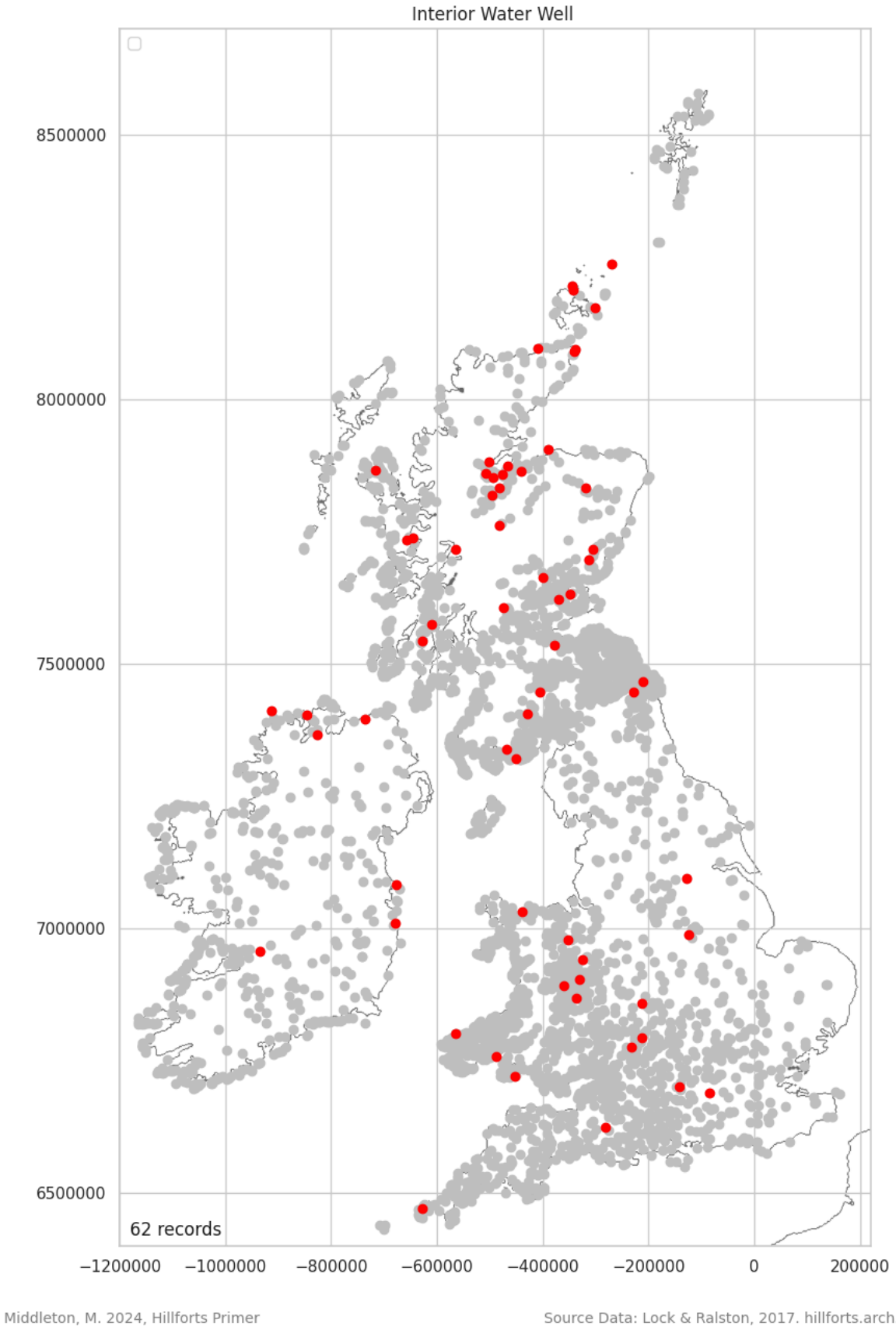


0.12%

Well Mapped

Wells are the most recorded water feature with 1.5% of hillorts recoded as having one.

In [94]:
```
int_well = plot_over_grey(location_water_data, 'Interior_Water_Well', 'Yes')
```

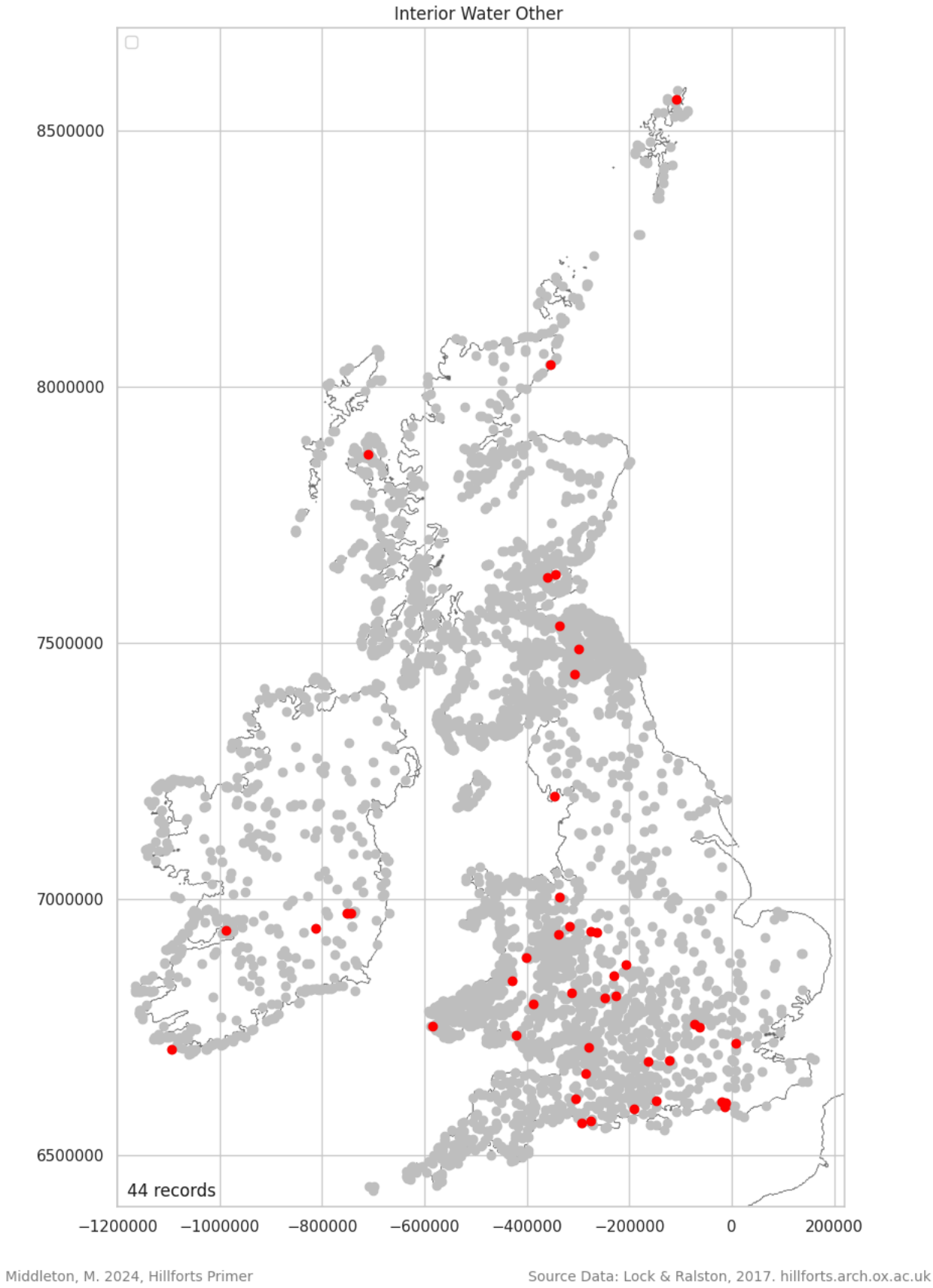


1.5%

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')
```



1.06%

Surface Data

This sections contains eight classes relating to internal fetures that are visible on the surface. The majority of hillforts (69.57%) have 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 addtion to these areas, rectangular structres 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 inerpretation 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	
1	Yes	No	No	No	No	
2	Yes	No	No	No	No	
3	No	No	No	No	Yes	
4	No	No	No	Yes	No	

There a 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

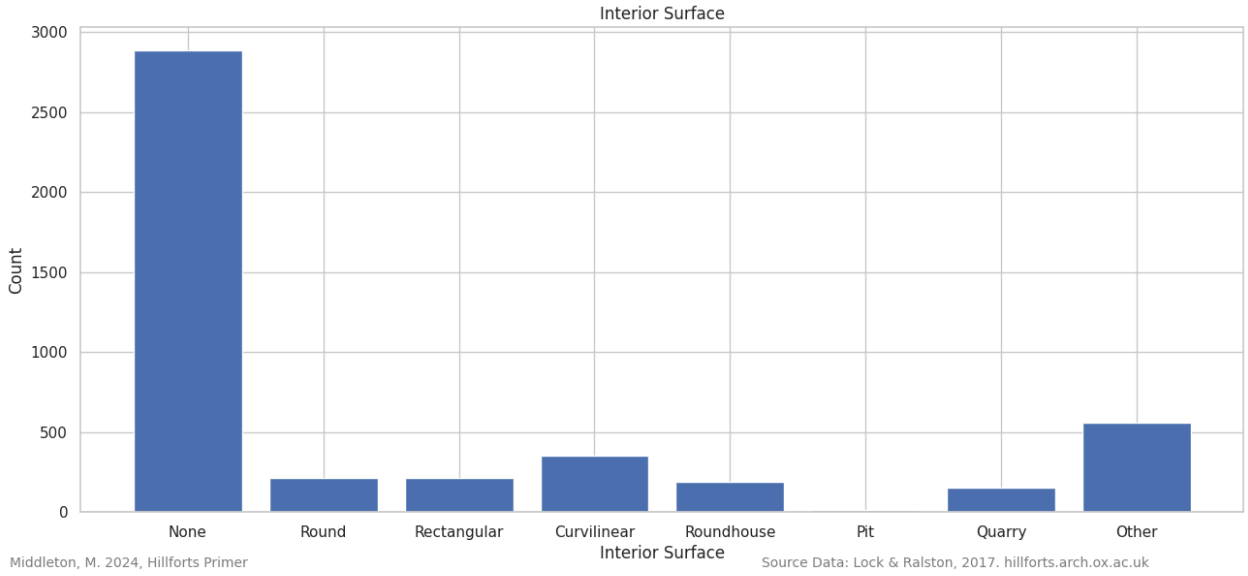
69.59% of Hillforts have no visible internal features recorded.

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)

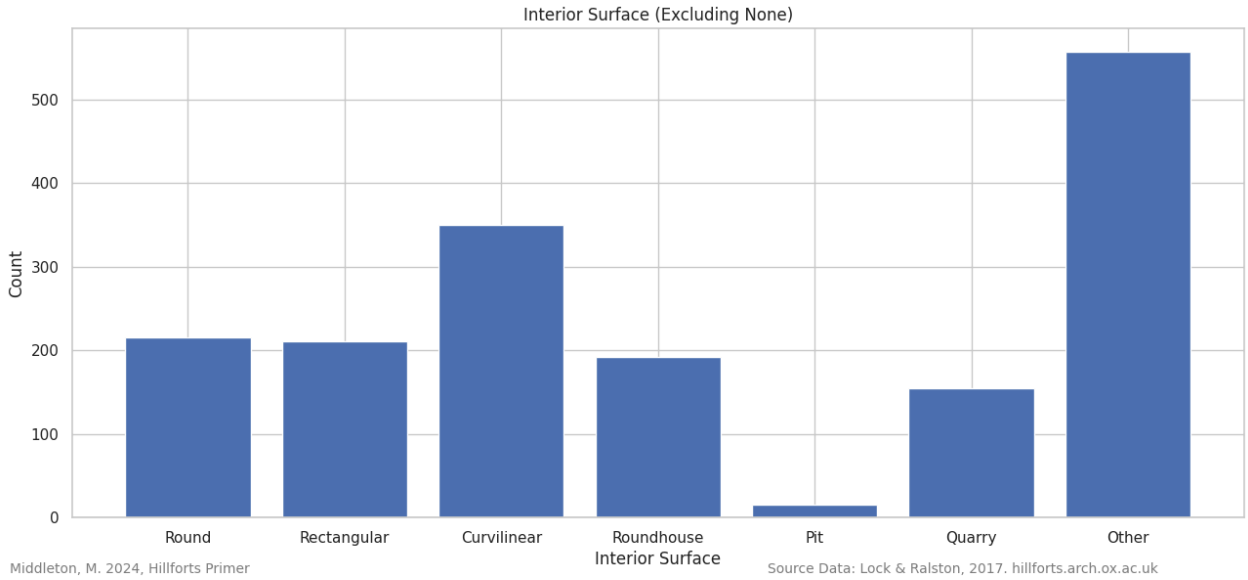
Where internal features have been recorded, there is a relitivly even distribuiton, accross the classes, with 204 (±12) forts with recorded examples of each, except for pits where there are only 15 and curvilinear features where there are 350.
See: Geophysics & Excavation Data Plotted (Excluding None)

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	Inter
0	No	No	No	No	No	
1	No	No	No	No	No	
2	No	No	No	No	No	
3	No	No	No	Yes	No	
4	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.

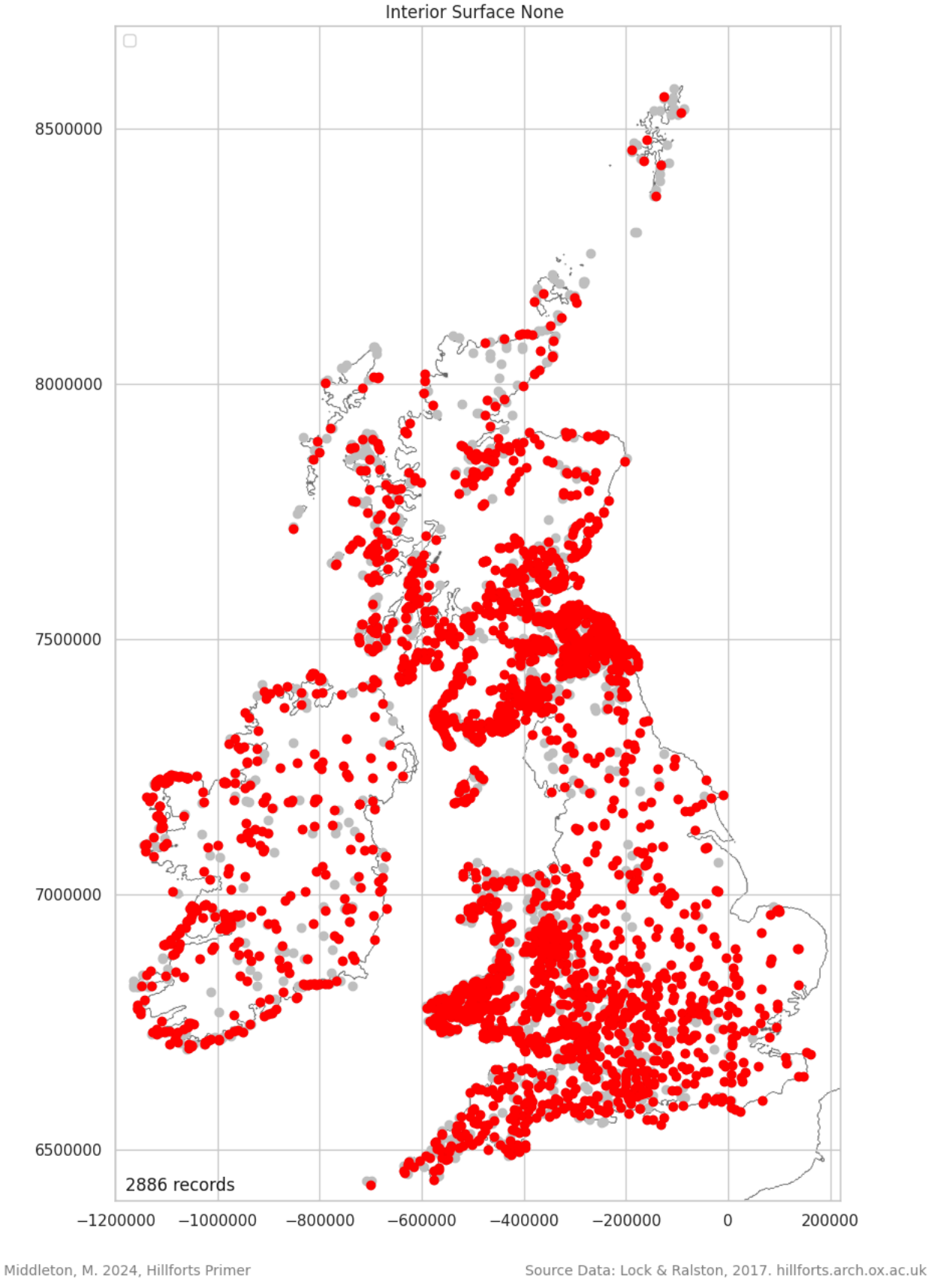
In [102…
```
location_surface_data = pd.merge(location_numeric_data_short, surface_data, left_index=True, right_index=True)
```

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')
```



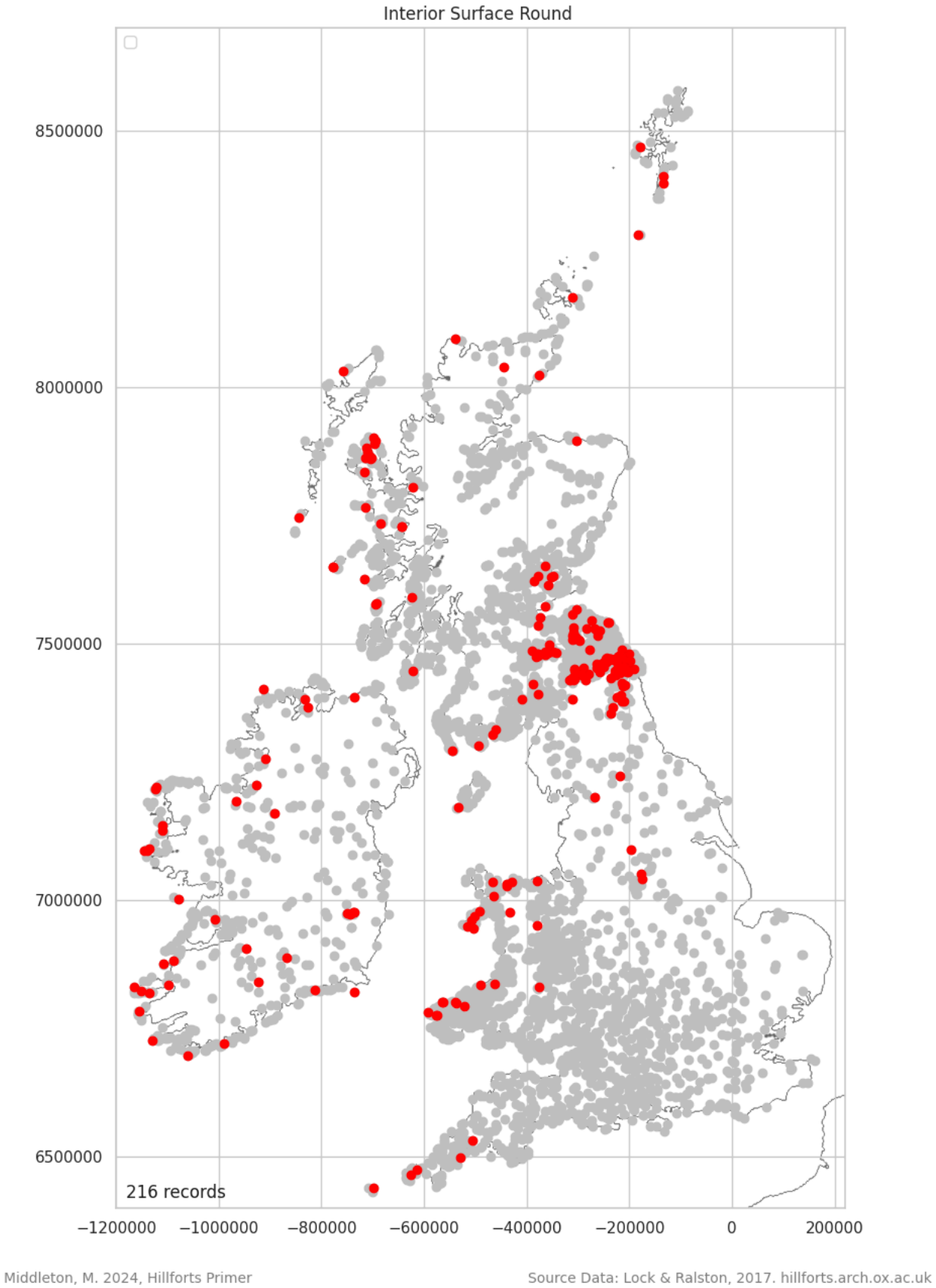

```
69.59%
```

Round Data Mapped

5.21% of hillforts are recorded as having circular internal features visable at the surfave. There is likely to be survey bias in this data, particulalry toward the concentration of data toward the eastern end of the Southern Uplands. It is noteable how few circular internal features have been recorded in England.

In [104…

```
su_round = plot_over_grey(location_surface_data, 'Interior_Surface_Round', 'Yes')
```

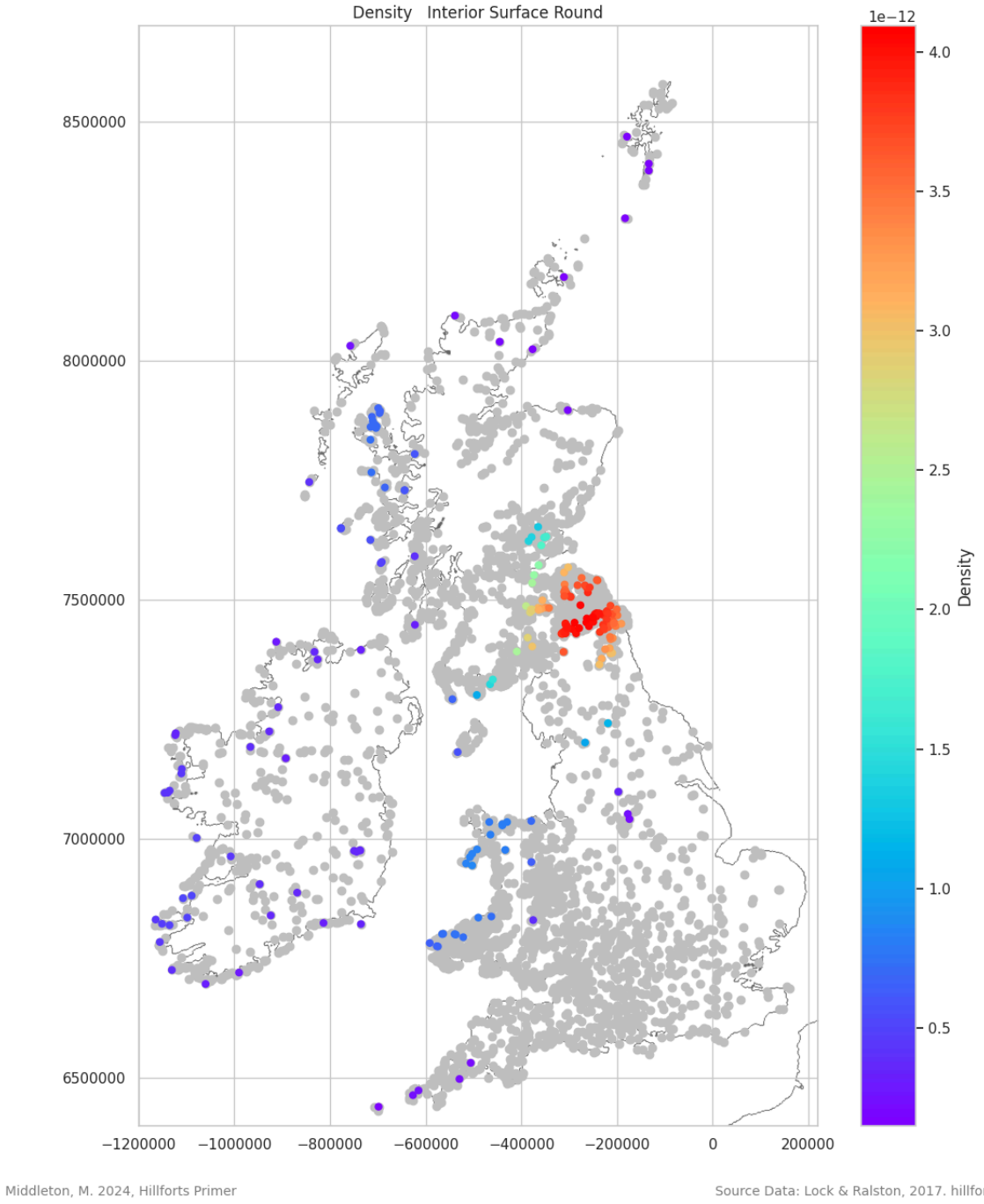



```
5.21%
```

Round Density Data Mapped

The density plot for round interior surface fetures most likley 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')
```

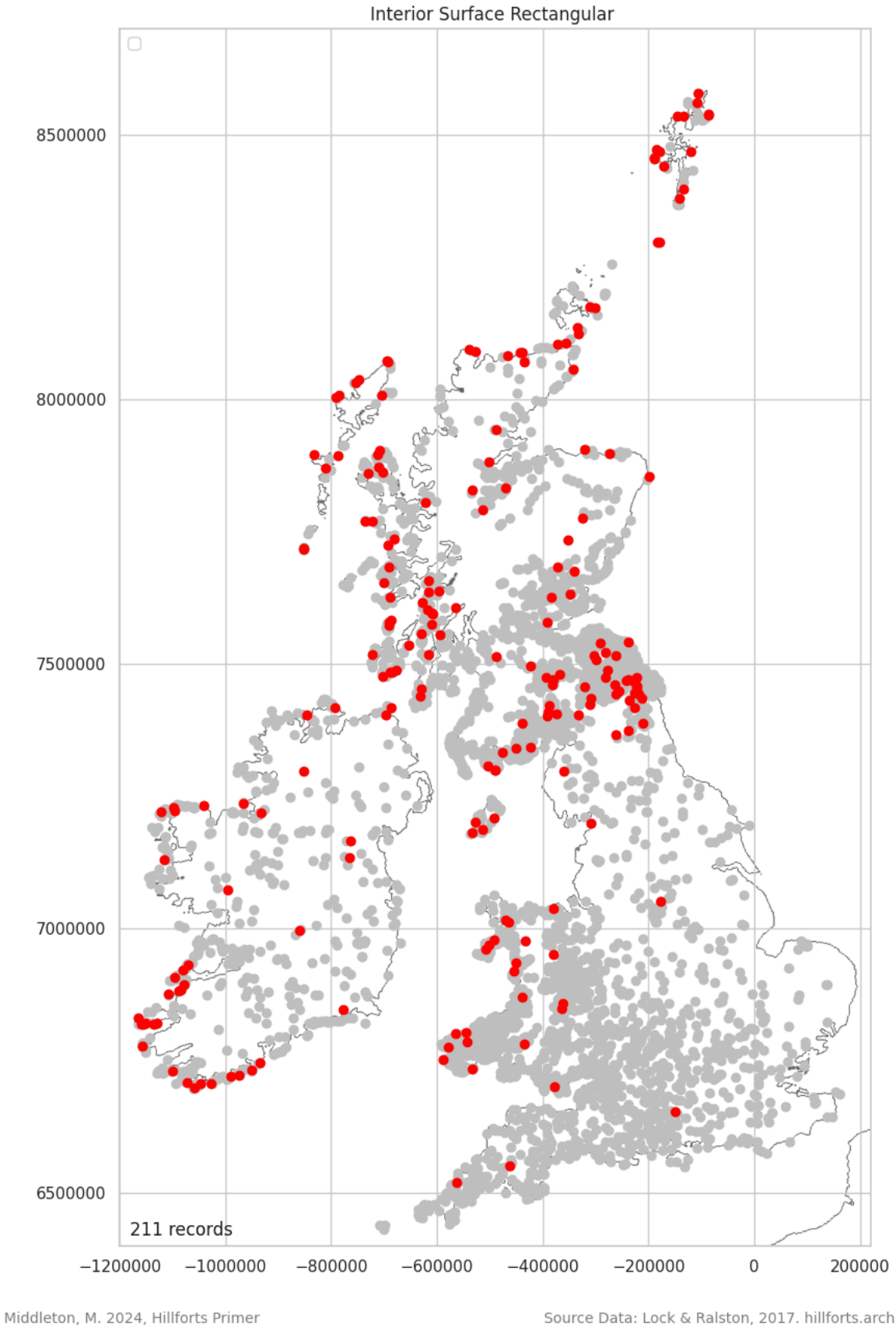


Rectangular Data Mapped

5.09% if hillforts are recorded as having rectangualr internal features. Like the round featrures 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 differnet land management regime within these forts leading to features no showing at the surface.

There is a noticable 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')
```



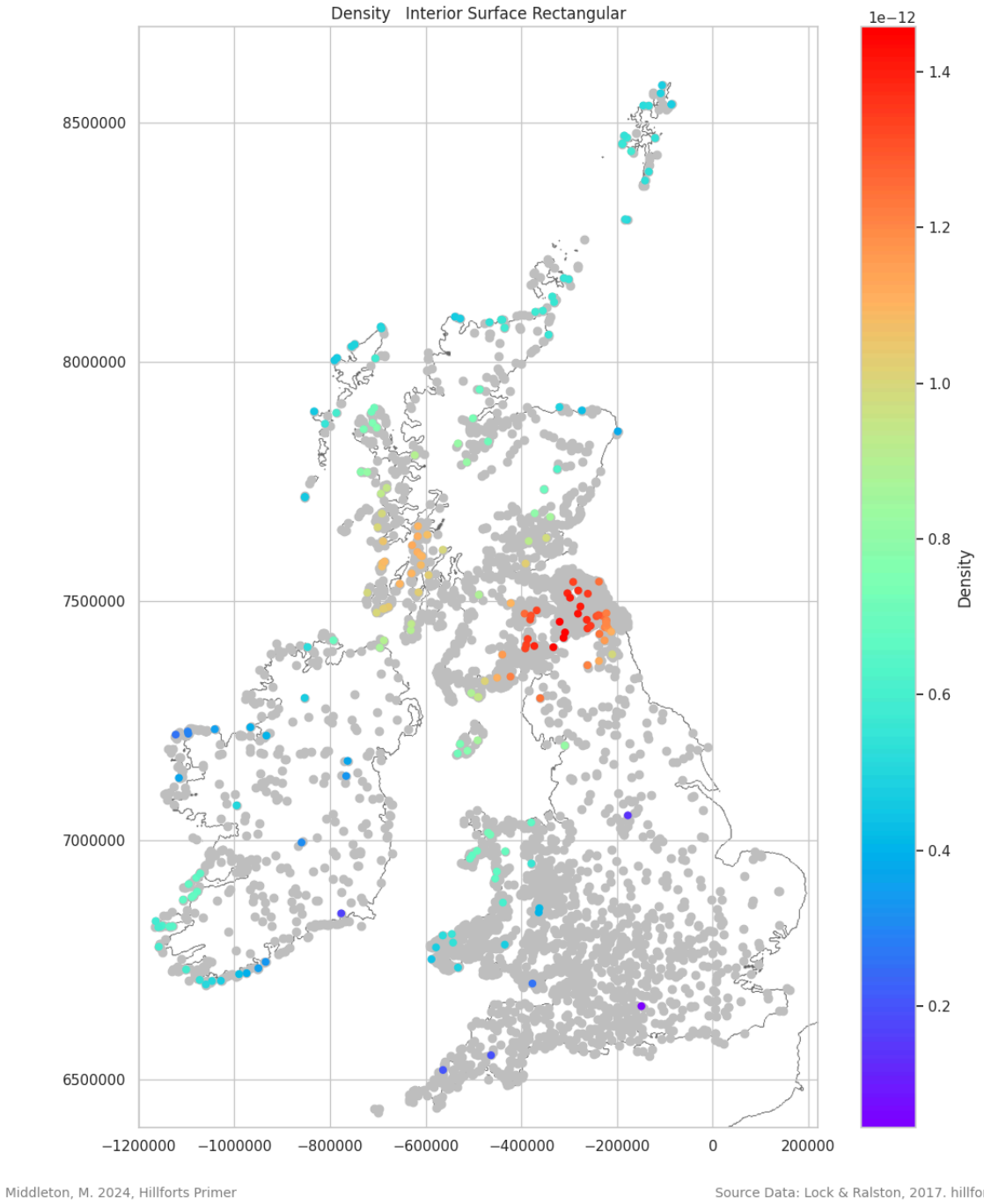
5.09%

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 custer in this plot. The Northwest, around Dunnad, is noteable as a secondary cluster.

In [107…

```
plot_density_over_grey(su_rect, 'Interior_Surface_Rectangular')
```

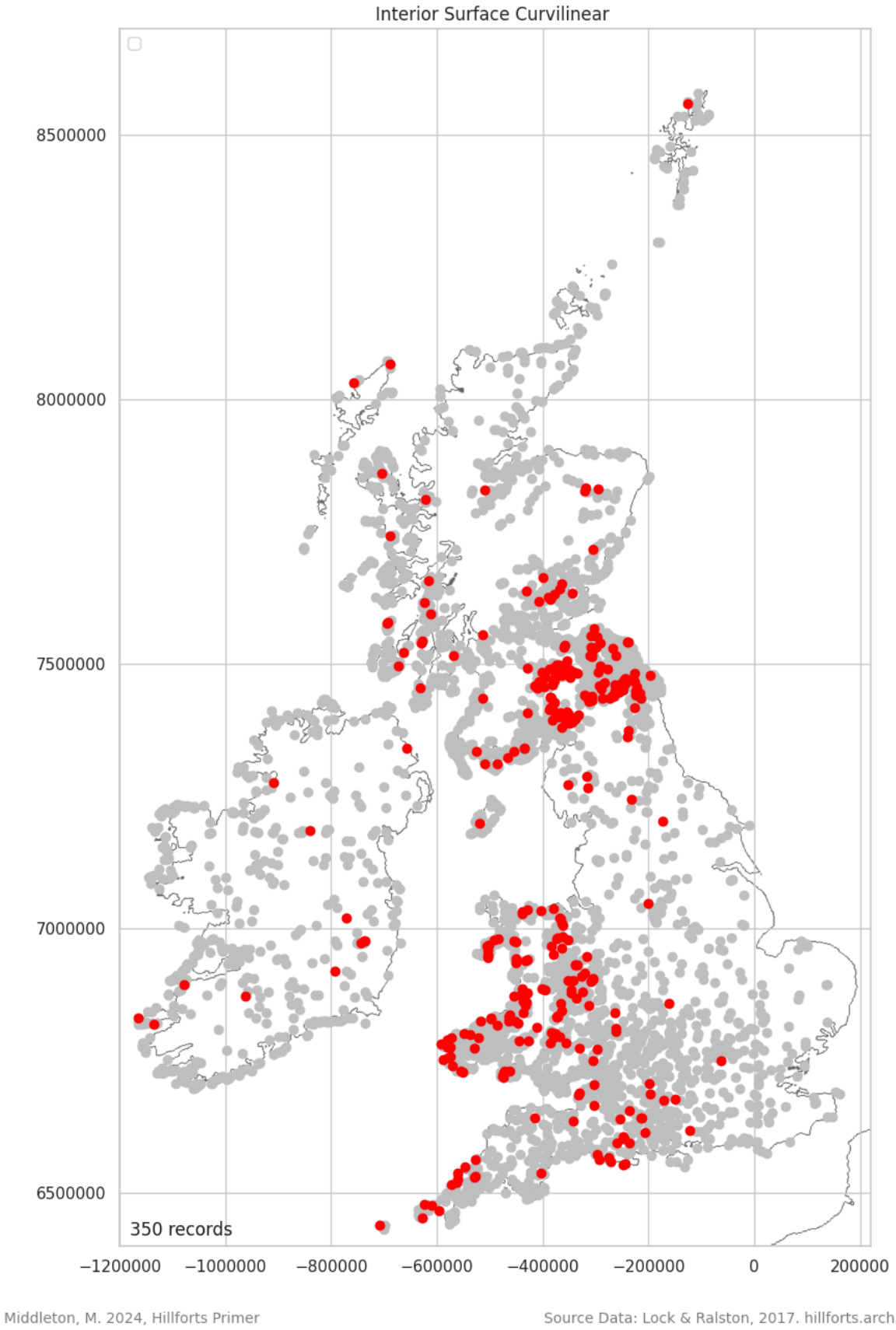


Curvilinear Data Mapped

8.44% of hillforts are recorded as having curvilinear sturctures 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')
```



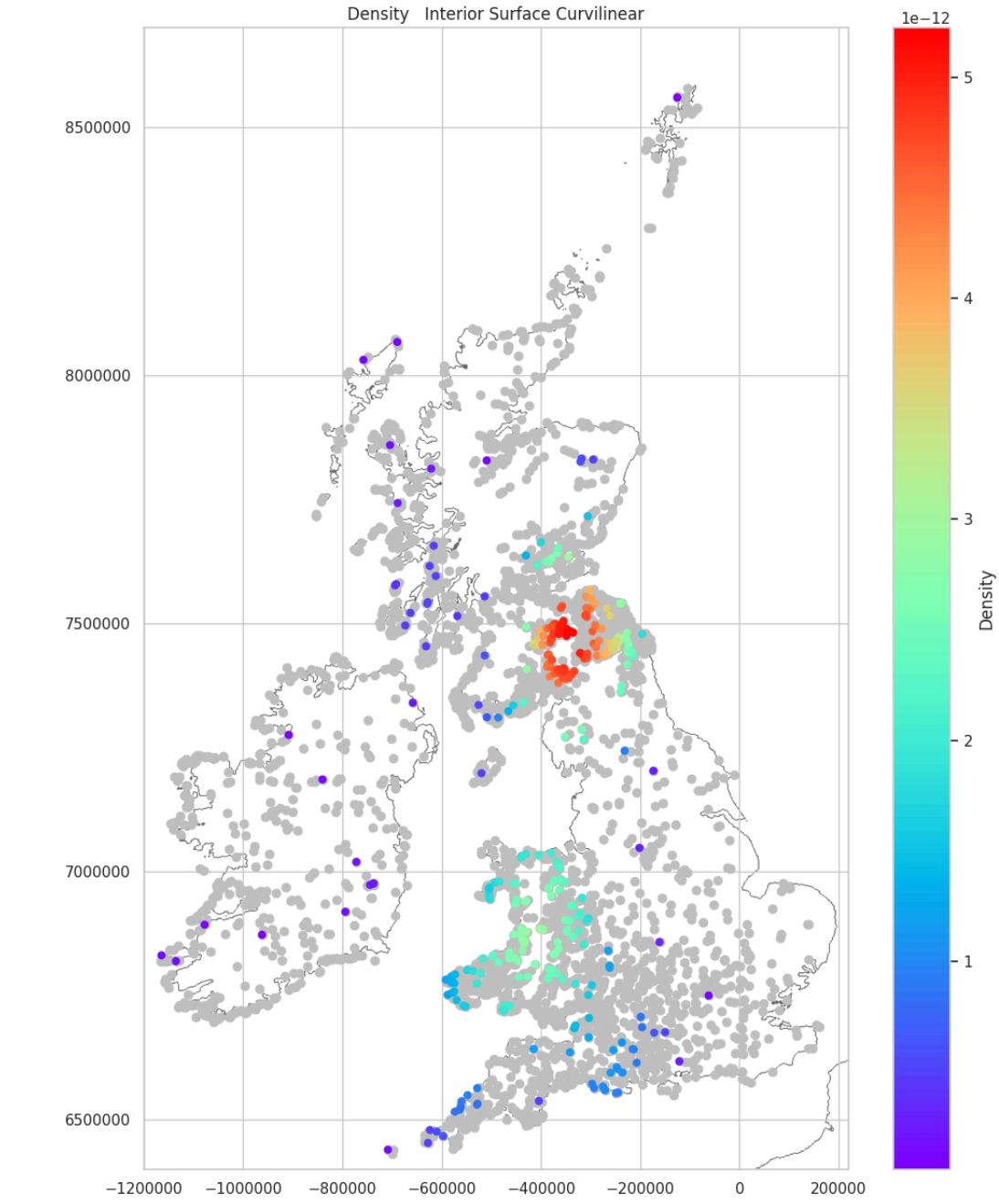
8.44%

Curvilinear Density Data Mapped

There are significant numbers of curviliear 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.

In [109…

```
plot_density_over_grey(su_curvi, 'Interior_Surface_Curvilinear')
```

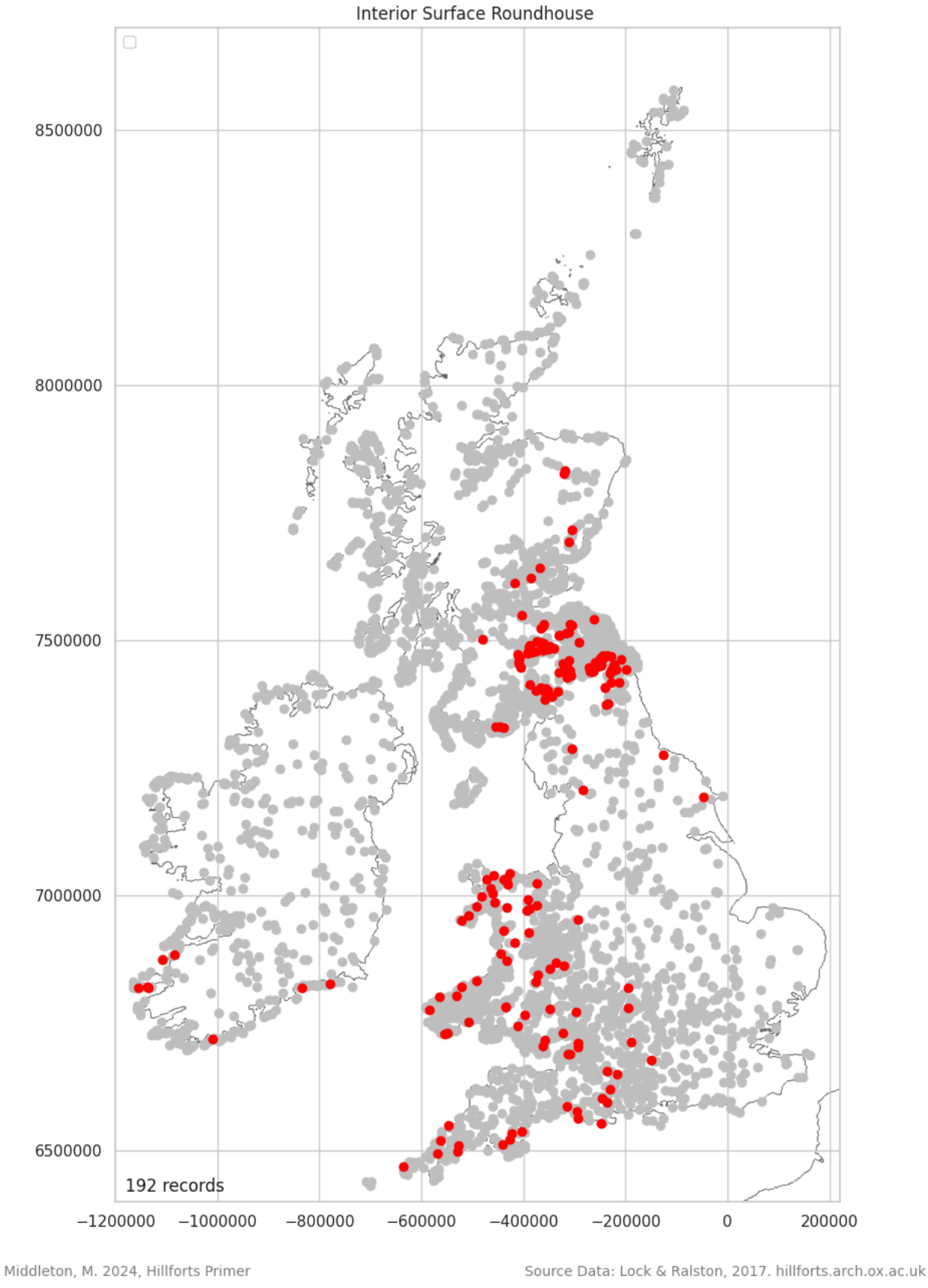


Roundhouse Data Mapped

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')
```



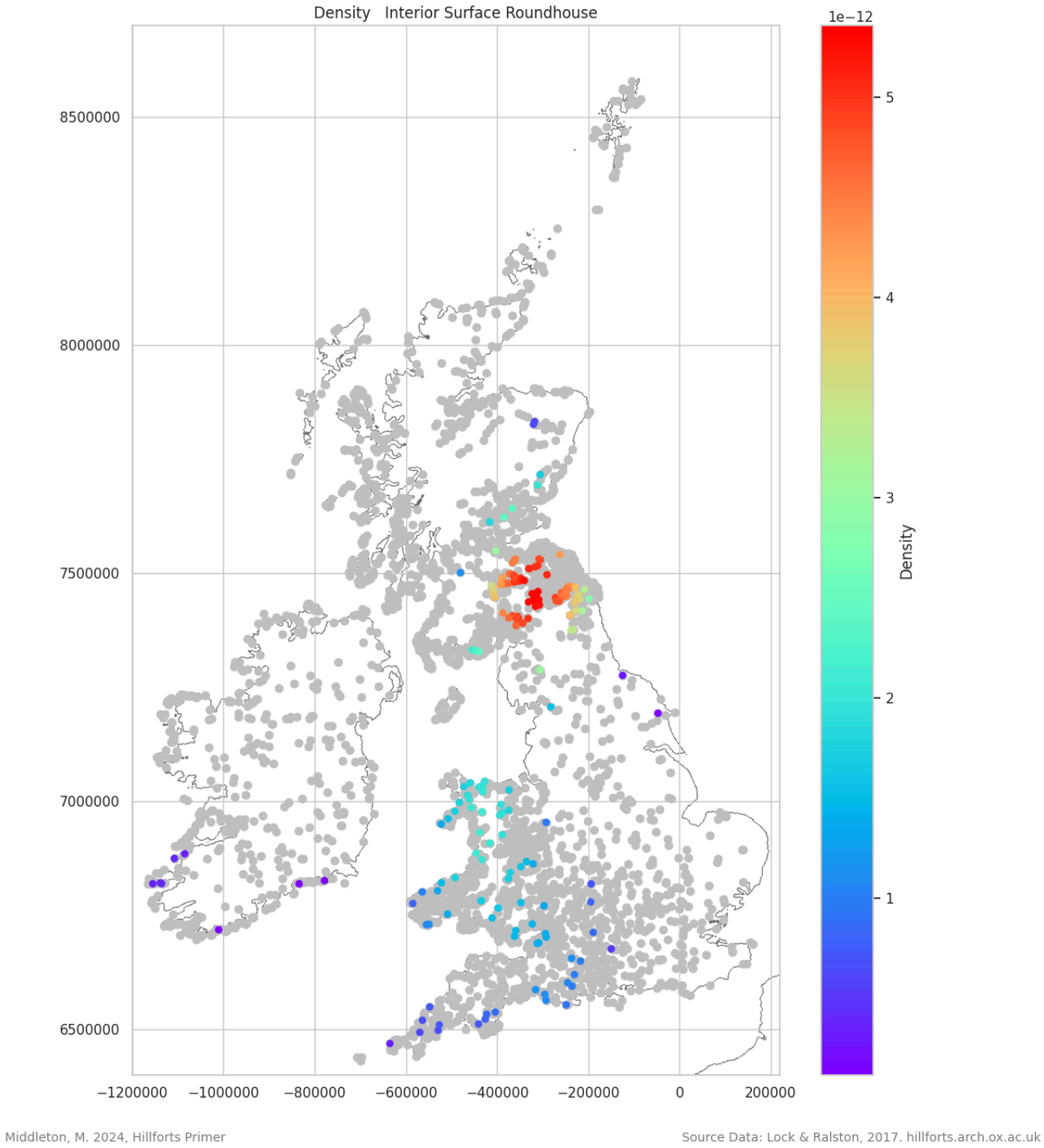

```
4.63%
```

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')
```

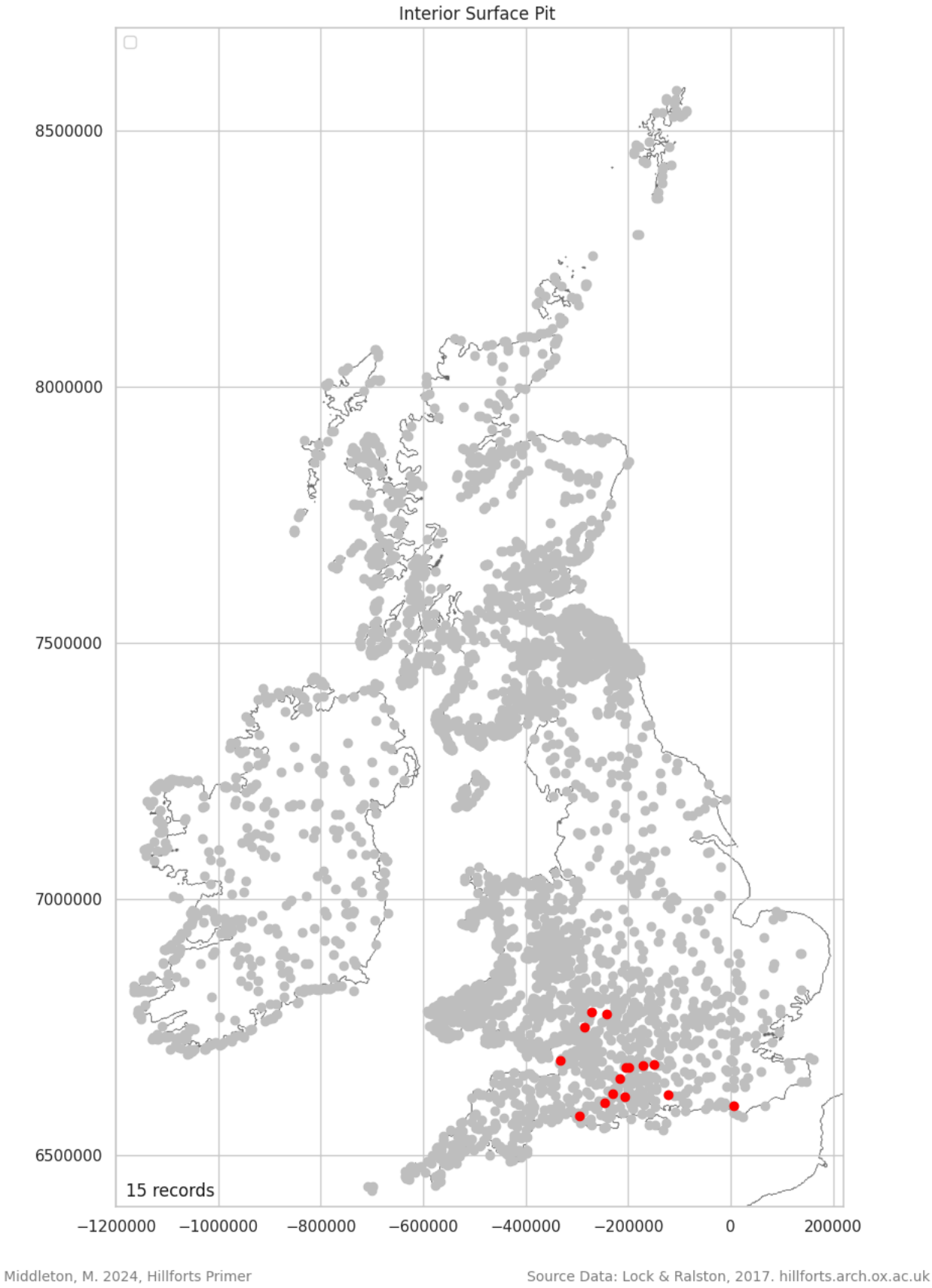


Pit Data Mapped

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 distribtion.

In [112…

```
su_pit = plot_over_grey(location_surface_data, 'Interior_Surface_Pit', 'Yes')
```



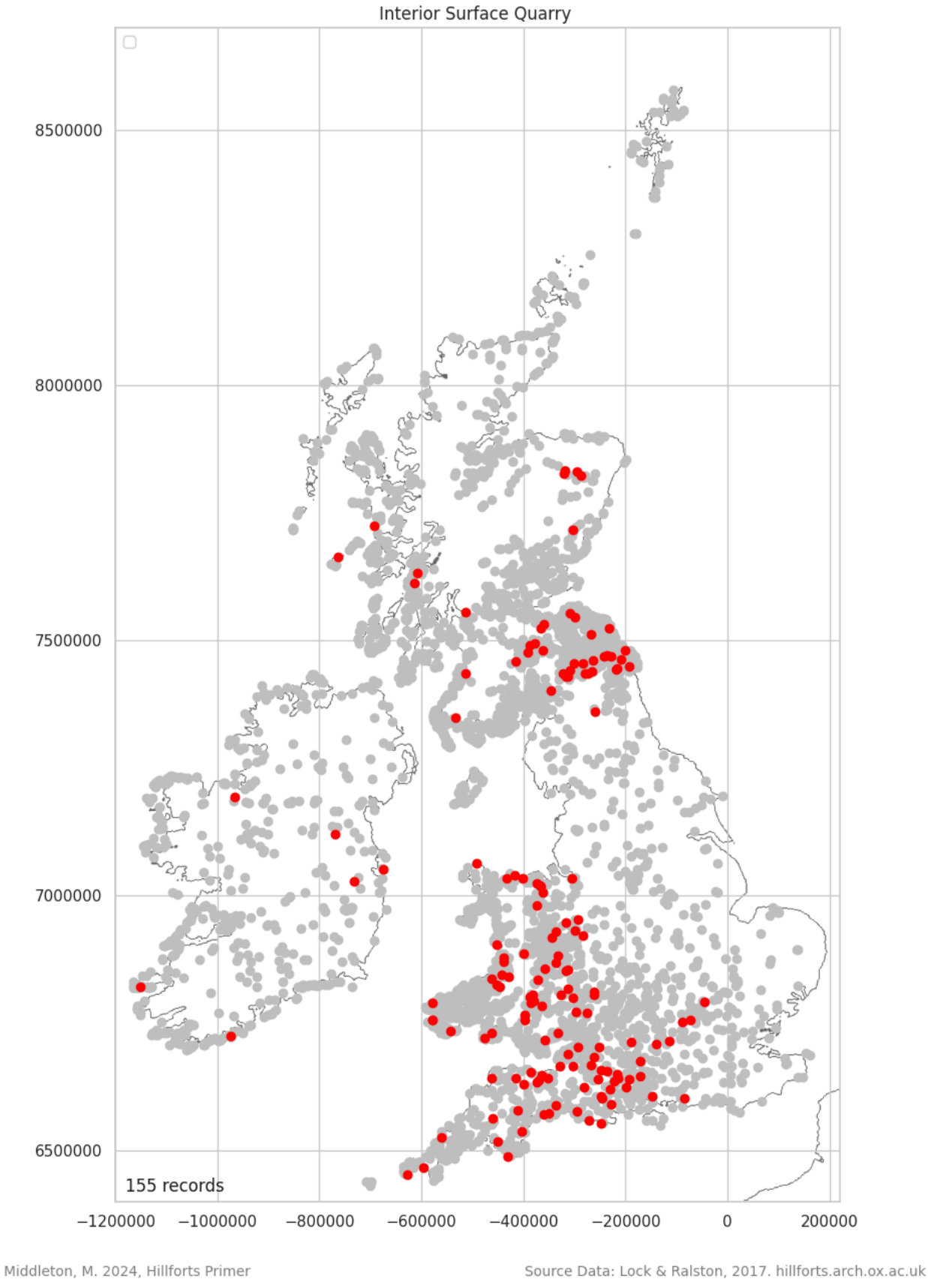
0.36%

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 accross south central England and up along the Welsh border. Generally, there is a survey variability bias accross the whole atlas.

In [113…

```
su_quarry = plot_over_grey(location_surface_data, 'Interior_Surface_Quarry', 'Yes')
```



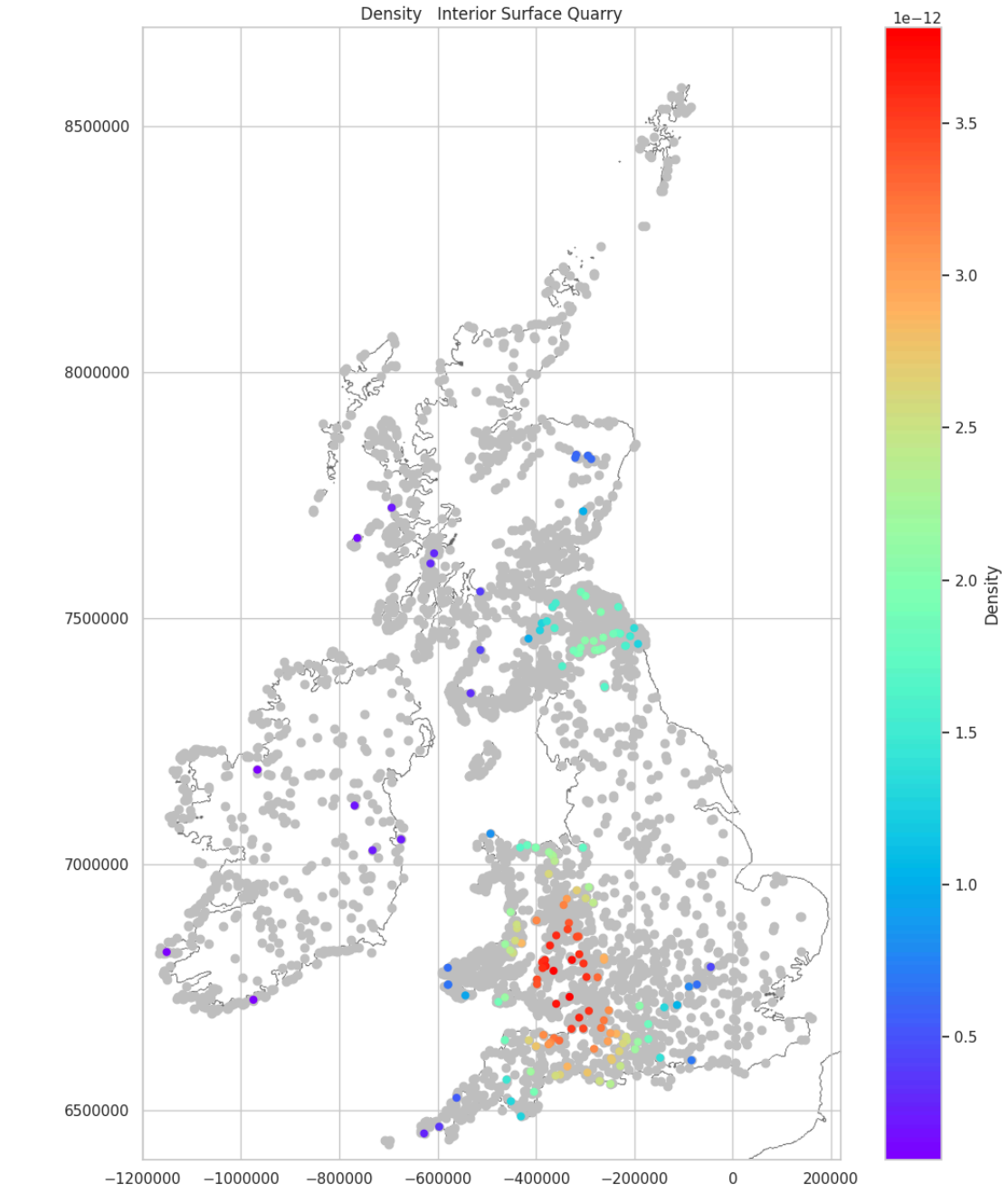
3.74%

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 irratic survey outwith these areas.

In [114…

```
plot_density_over_grey(su_quarry, 'Interior_Surface_Quarry')
```

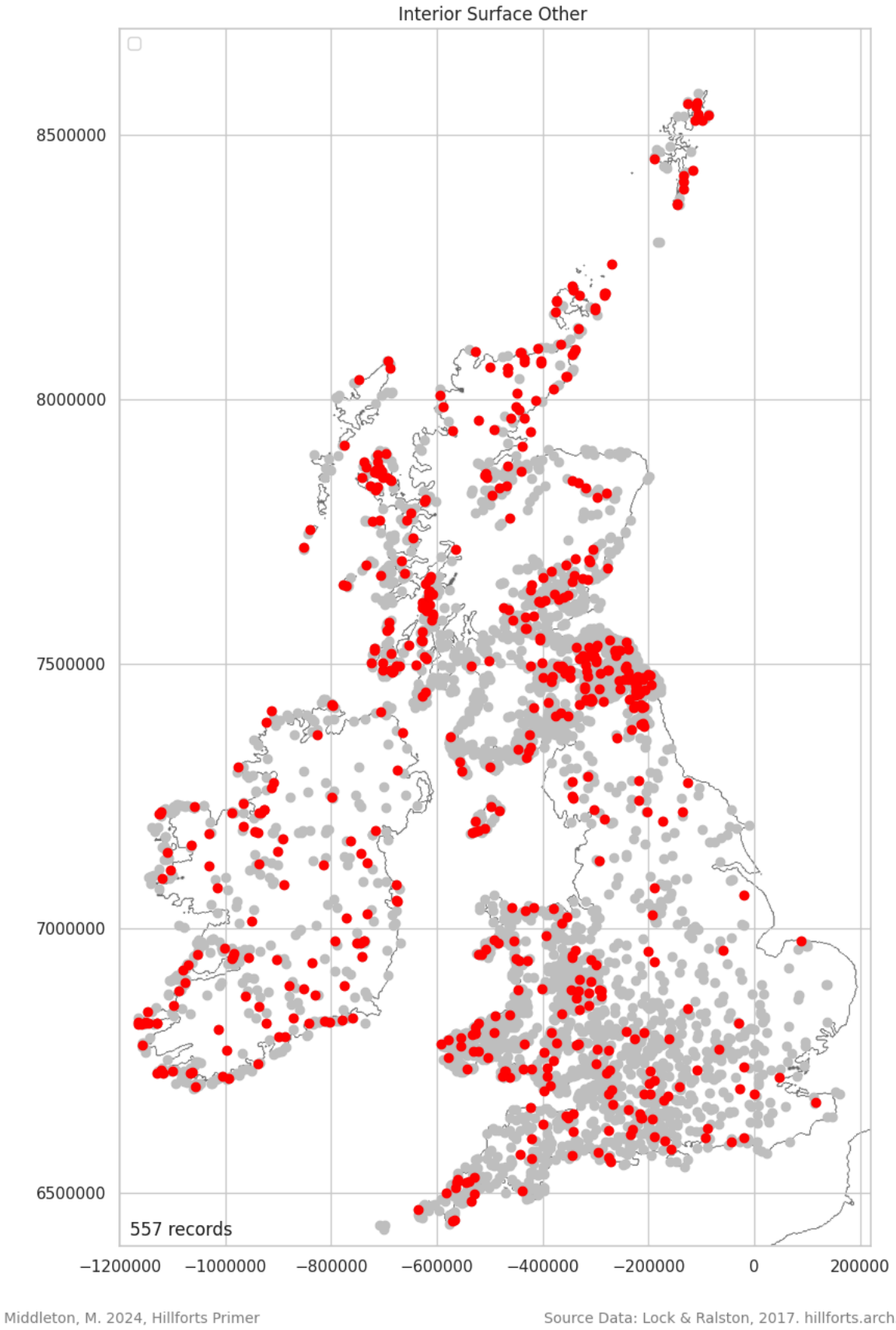


Other Surface Data Mapped

13.43% of hillforts have 'other' surface features recorded.

In [115…

```
su_other = plot_over_grey(location_surface_data, 'Interior_Surface_Other', 'Yes')
```



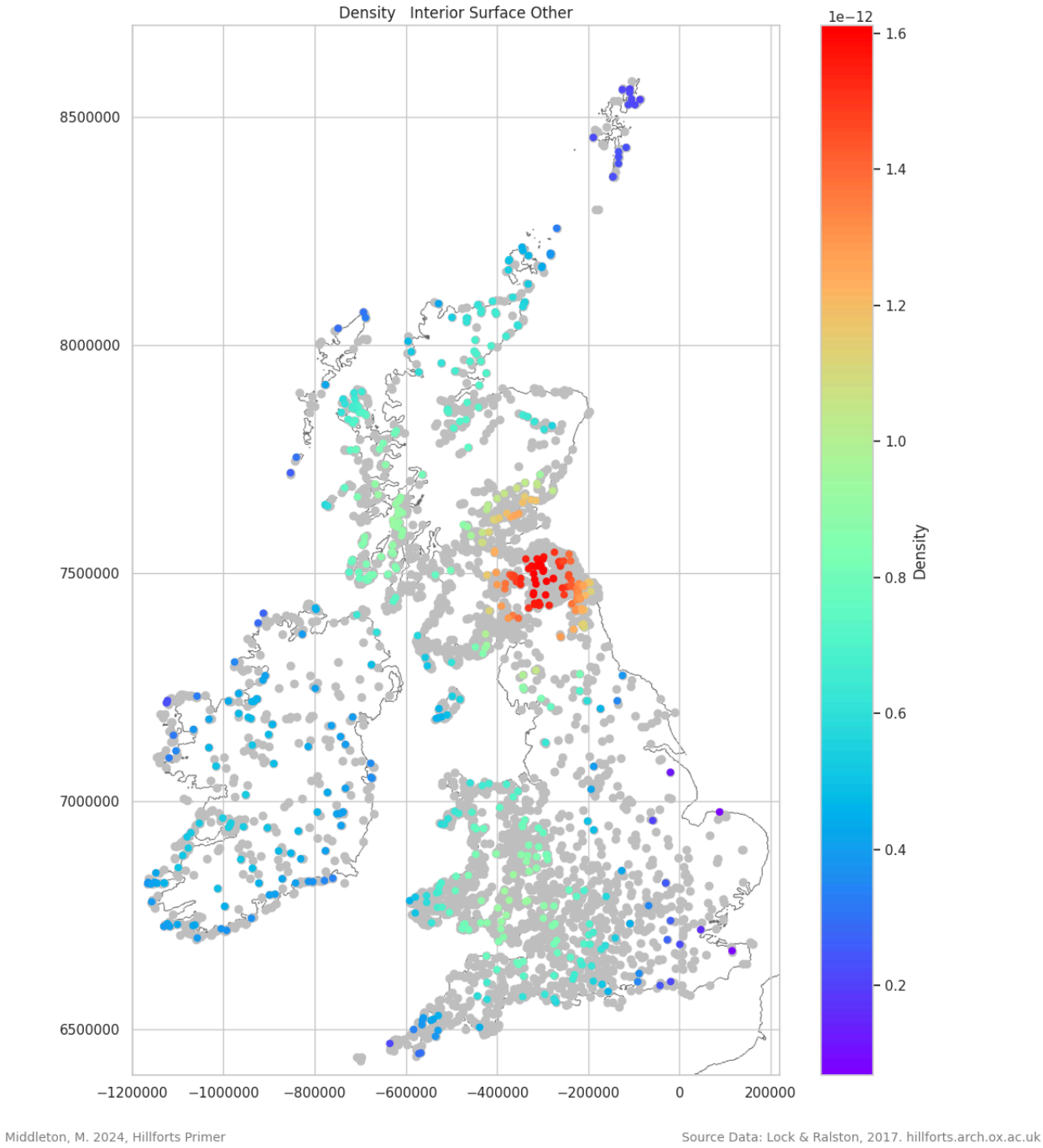
13.43%

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 Cambriam Mountains is off set to the east.

In [116…

```
plot_density_over_grey(su_other, 'Interior_Surface_Other')
```



Excavation Data

The Excavation Data contains nine classes. Most (84.01%) of hillforts have no excavation evidence. Eight of the classes discribe the types of strucuters 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()
```

Out[117]:

	Interior_Excavation_None	Interior_Excavation_Pit	Interior_Excavation_Posthole	Interior_Excavation_Roundhouse	Interior_Excavation_Recta
0	Yes	No	No	No	
1	Yes	No	No	No	
2	Yes	No	No	No	
3	No	Yes	Yes	Yes	
4	No	No	No	No	

There are no null values.

In [118…

```
excavation_data.info()
```

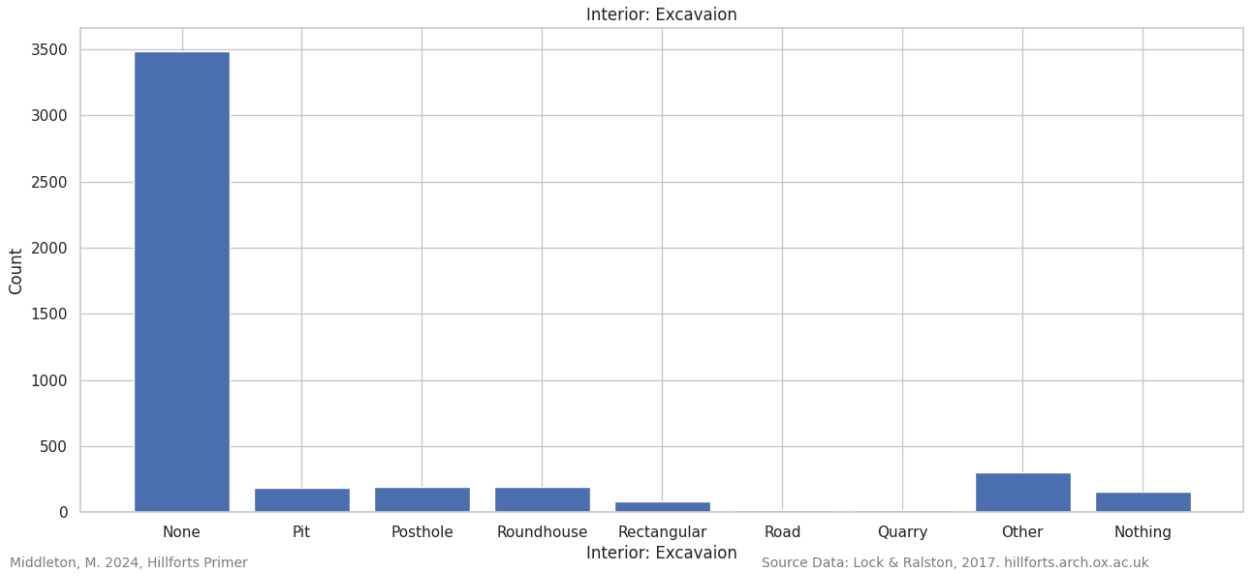
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 9 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Interior_Excavation_None        4147 non-null   object
 1   Interior_Excavation_Pit         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: Excavaion', 'Count', 'Interior: Excavaion')
```



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
```

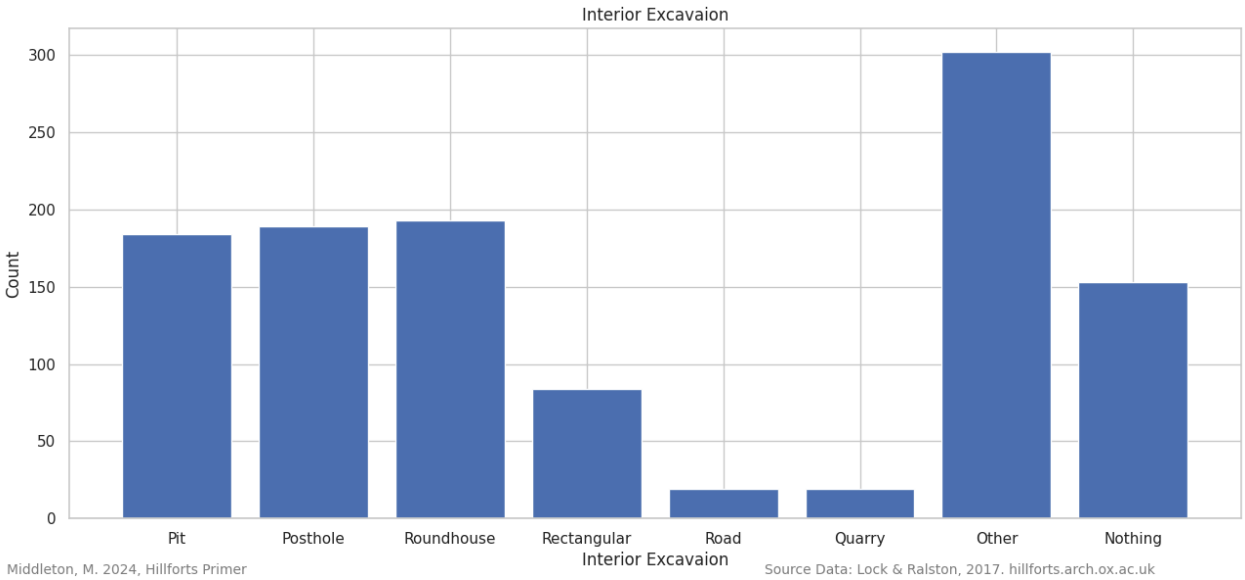
Out[124]:
```
45.55
```

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_
0	No	No	No	No	
1	No	No	No	No	
2	No	No	No	No	
3	Yes	Yes	Yes	Yes	
4	No	No	No	No	

In [126…
```
plot_bar_chart(excavation_data_minus, 2, 'Interior Excavaion', 'Count', 'Interior Excavaion')
```



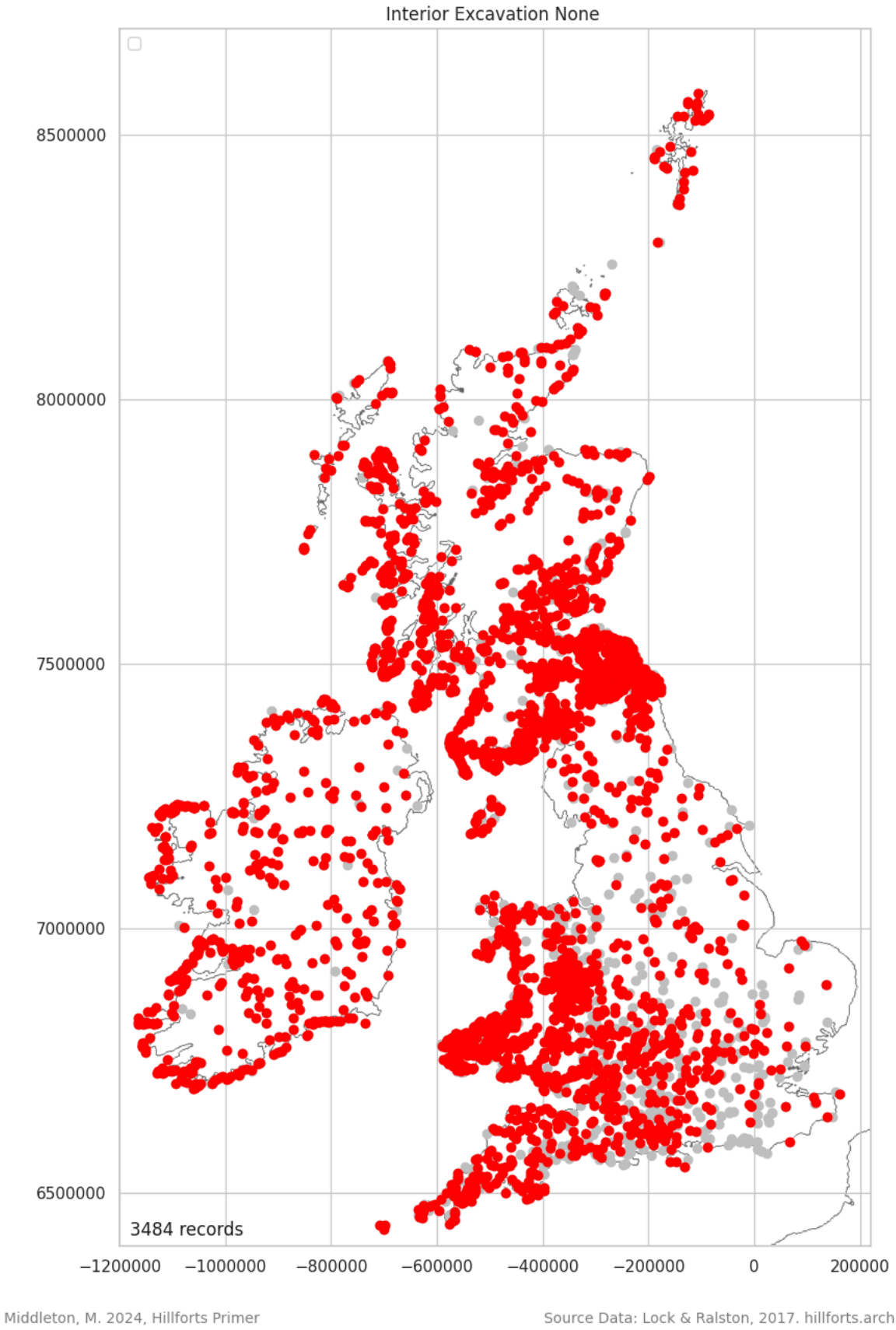
Excavation Data Mapped

In [127…
```
location_excavaion_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_excavaion_data, 'Interior_Excavation_None', 'Yes')
```



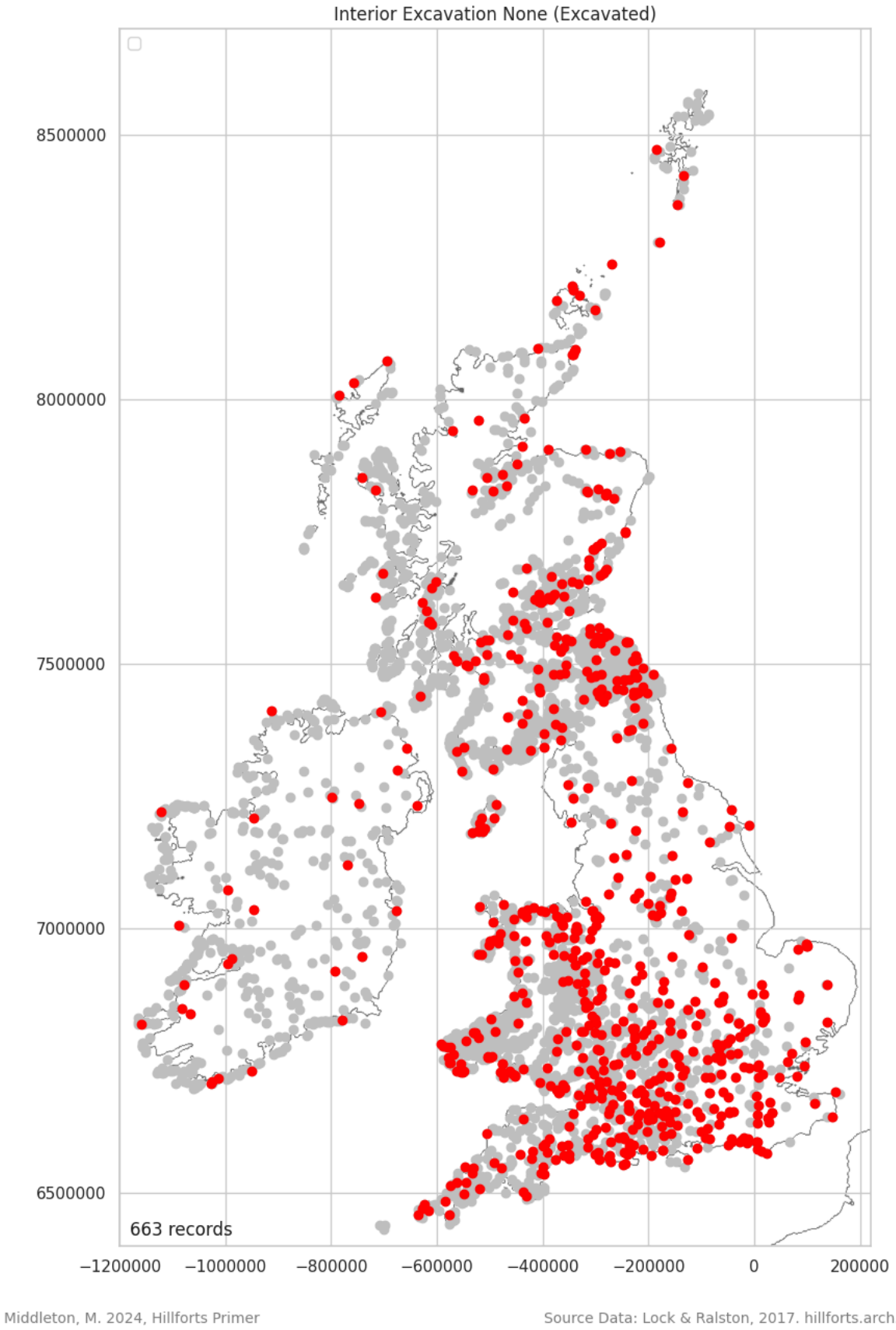
84.01%

Excavation: None Mapped (Excavated)

633 (15.99%) of hillforts have been excavated in part.

In [129…

```
int_ex = plot_over_grey(location_excavaion_data, 'Interior_Excavation_None', 'No', "(Excavated)")
```



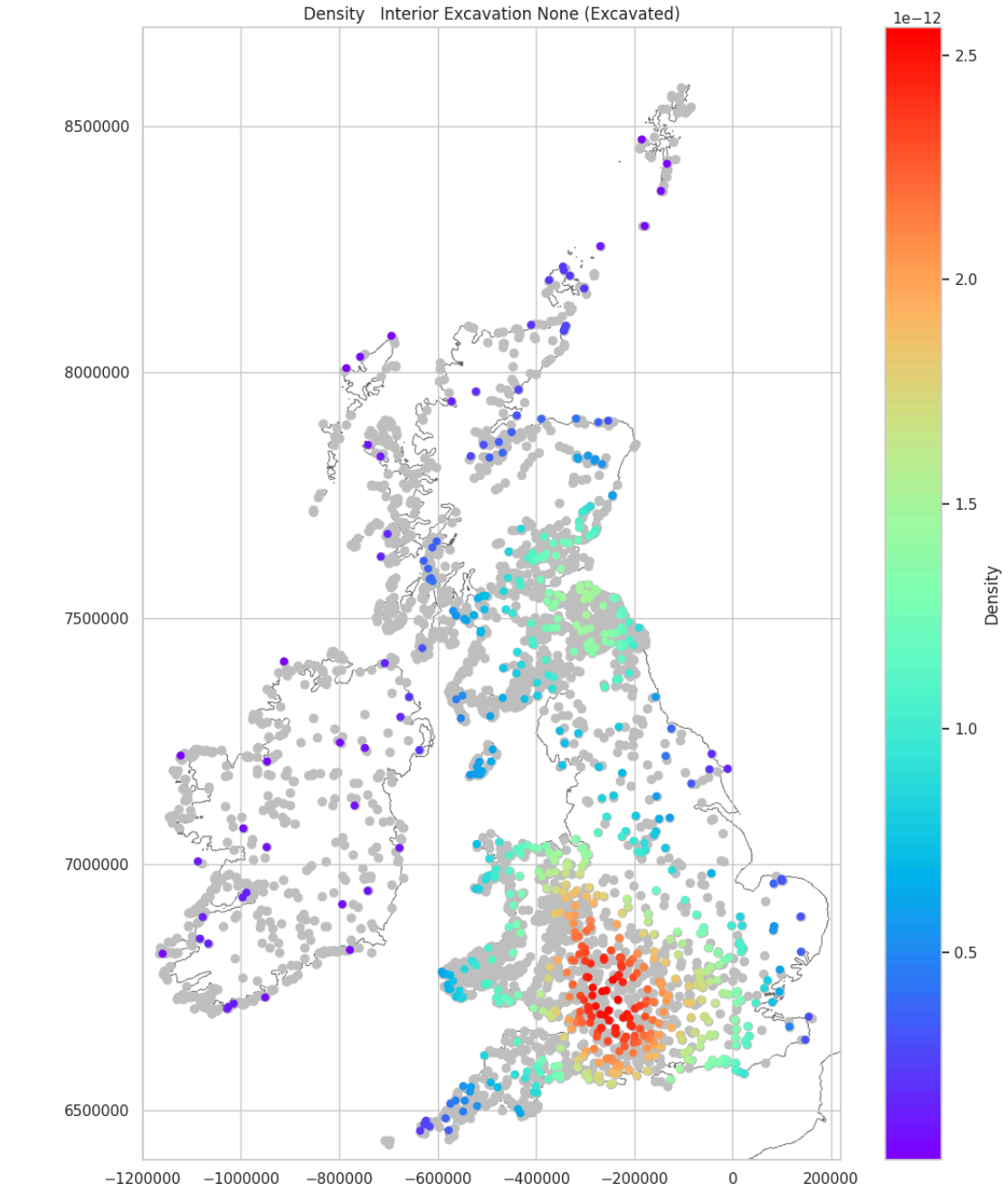
15.99%

Excavation: None Density Mapped (Excavated)

The densest cluster of excavated hillorts 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)')
```

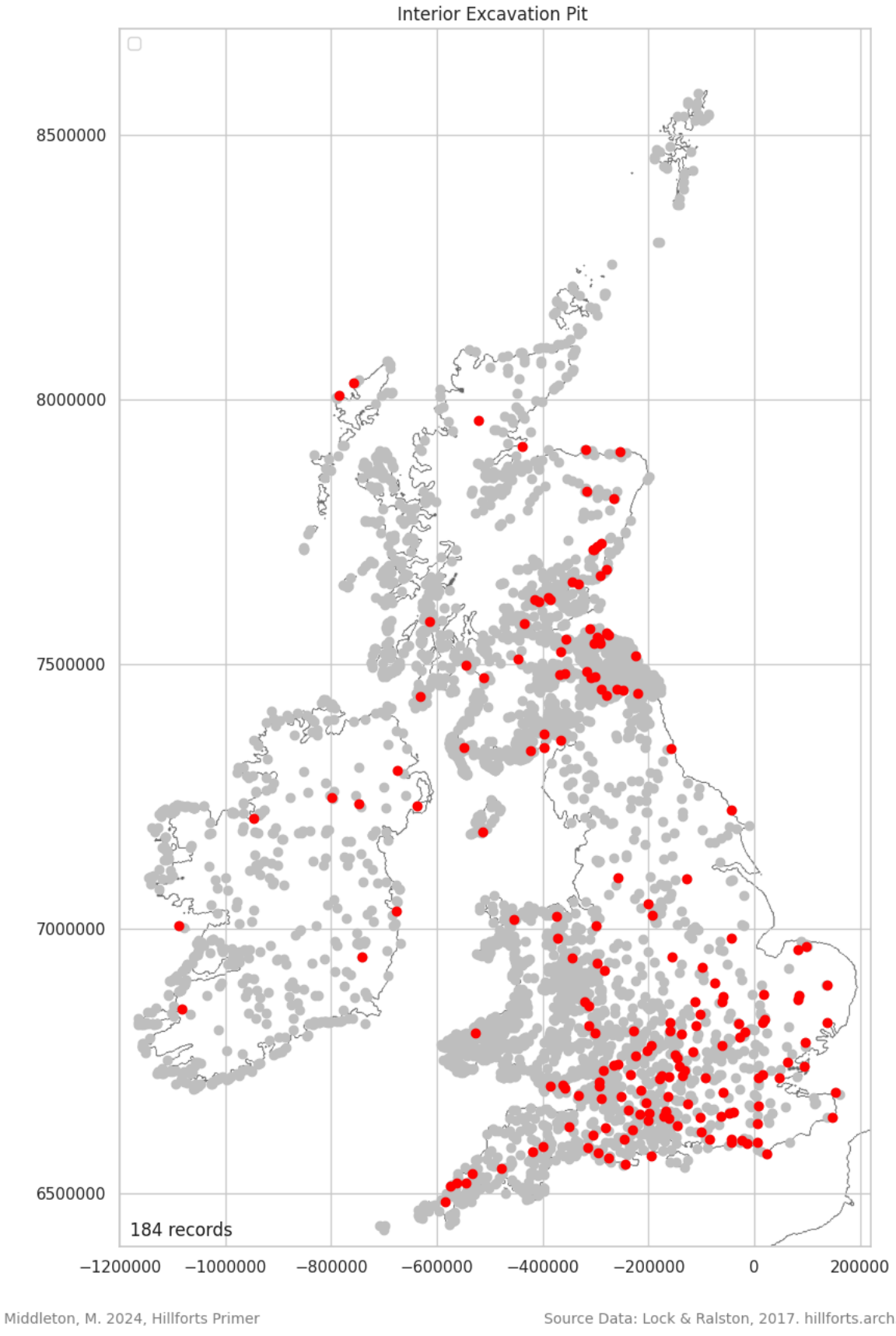


Excavation: Pit Mapped

Pits are recorded at many of the southern hillforts and a good number of the northern forts. It is noticable 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_excavaion_data, 'Interior_Excavation_Pit', 'Yes')
```




```
4.44%
```

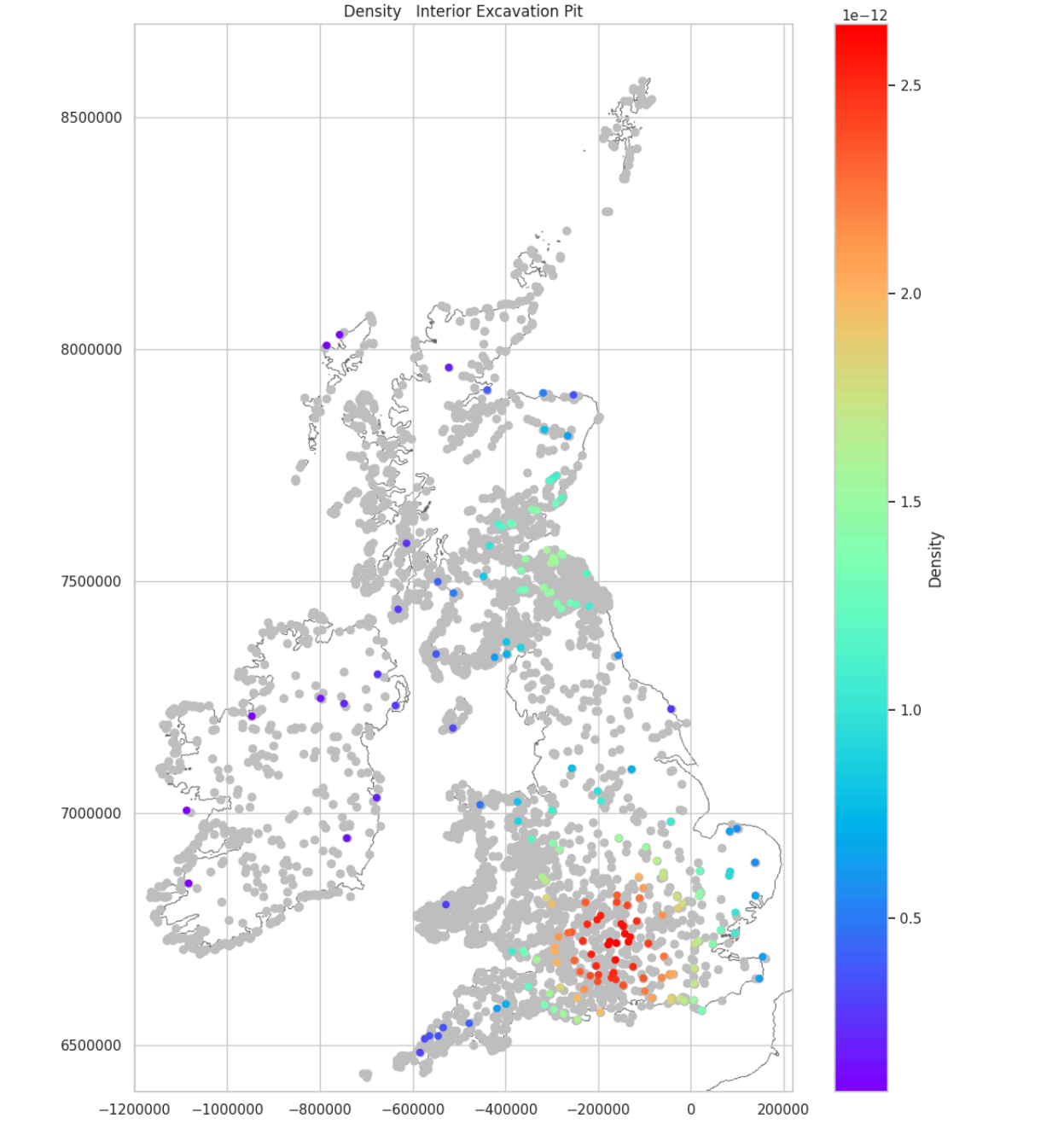
In [131…

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')
```



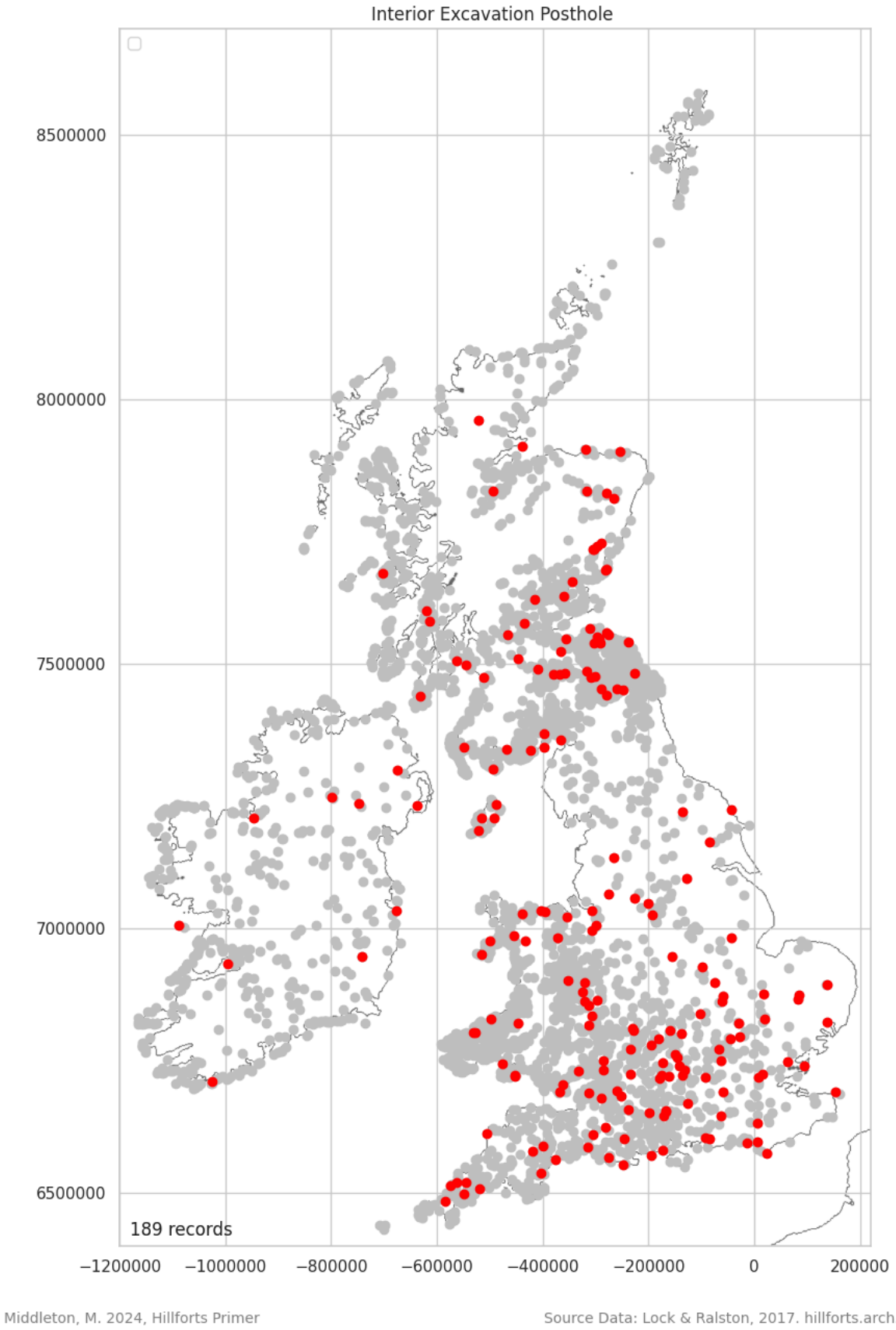
Middleton, M. 2024, Hillforts Primer

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_excavaion_data, 'Interior_Excavation_Posthole', 'Yes')
```



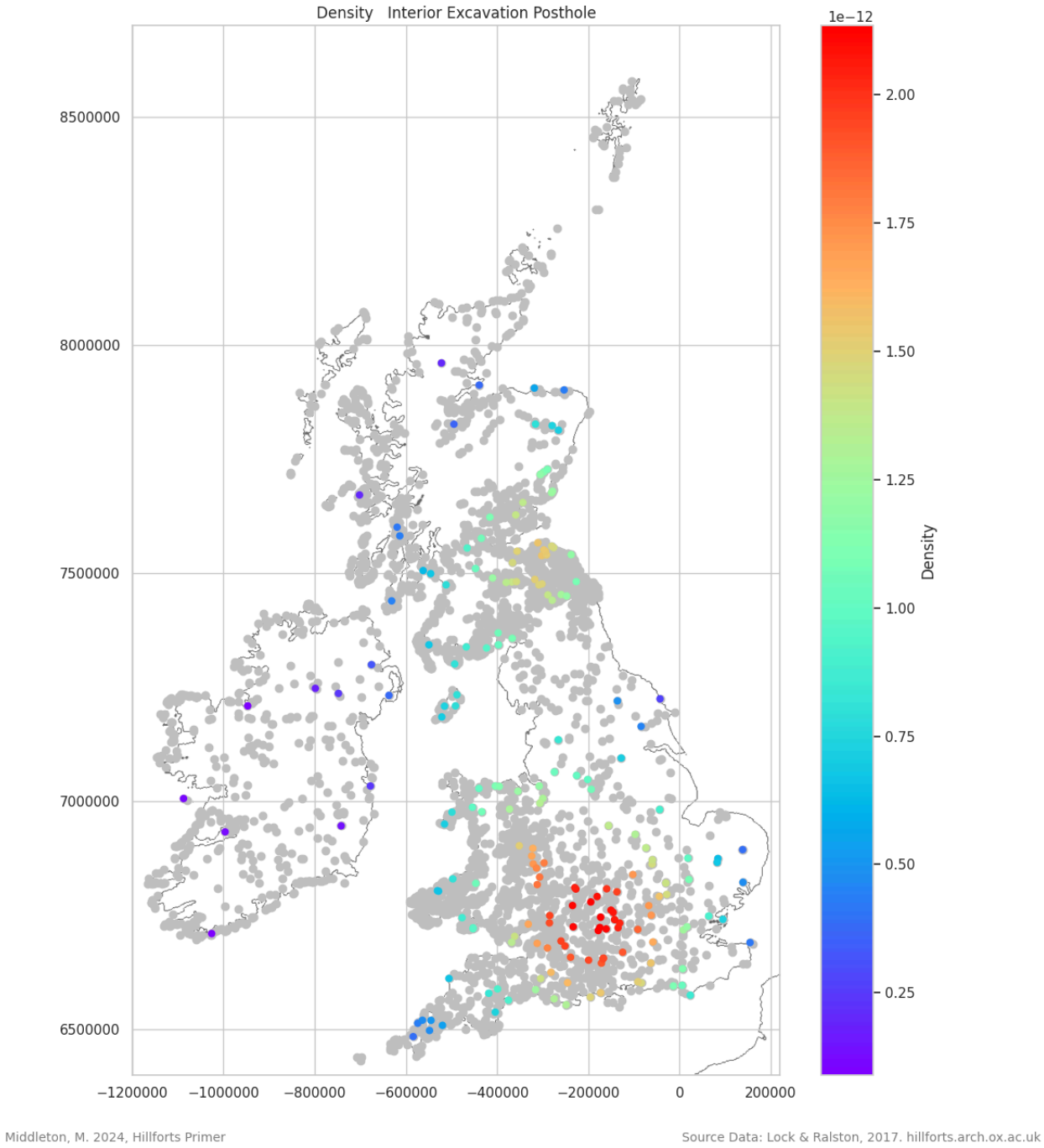
4.56%

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')
```

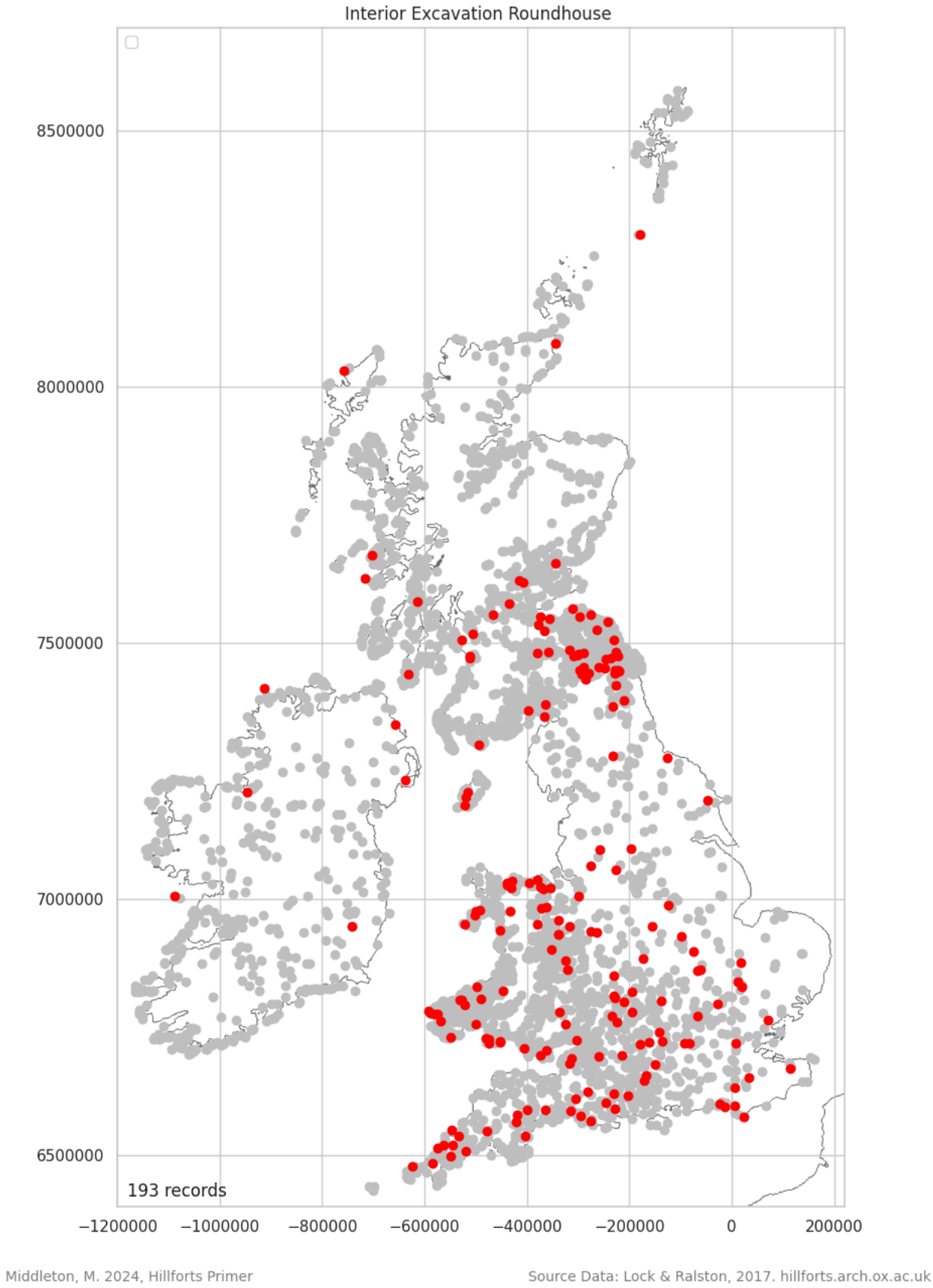


Excavaion: Roundhouse Mapped

Roundhouses have been recorded widely across the excavation record. It is noteable how few have been recorded in northern and westernn Scotland but it is possible that as roundhouses include a timber post ring, they have been recoded as posthole structures, and not roundhouses, in this areas.

In [135…

```
int_ex_rh = plot_over_grey(location_excavaion_data, 'Interior_Excavation_Roundhouse', 'Yes')
```

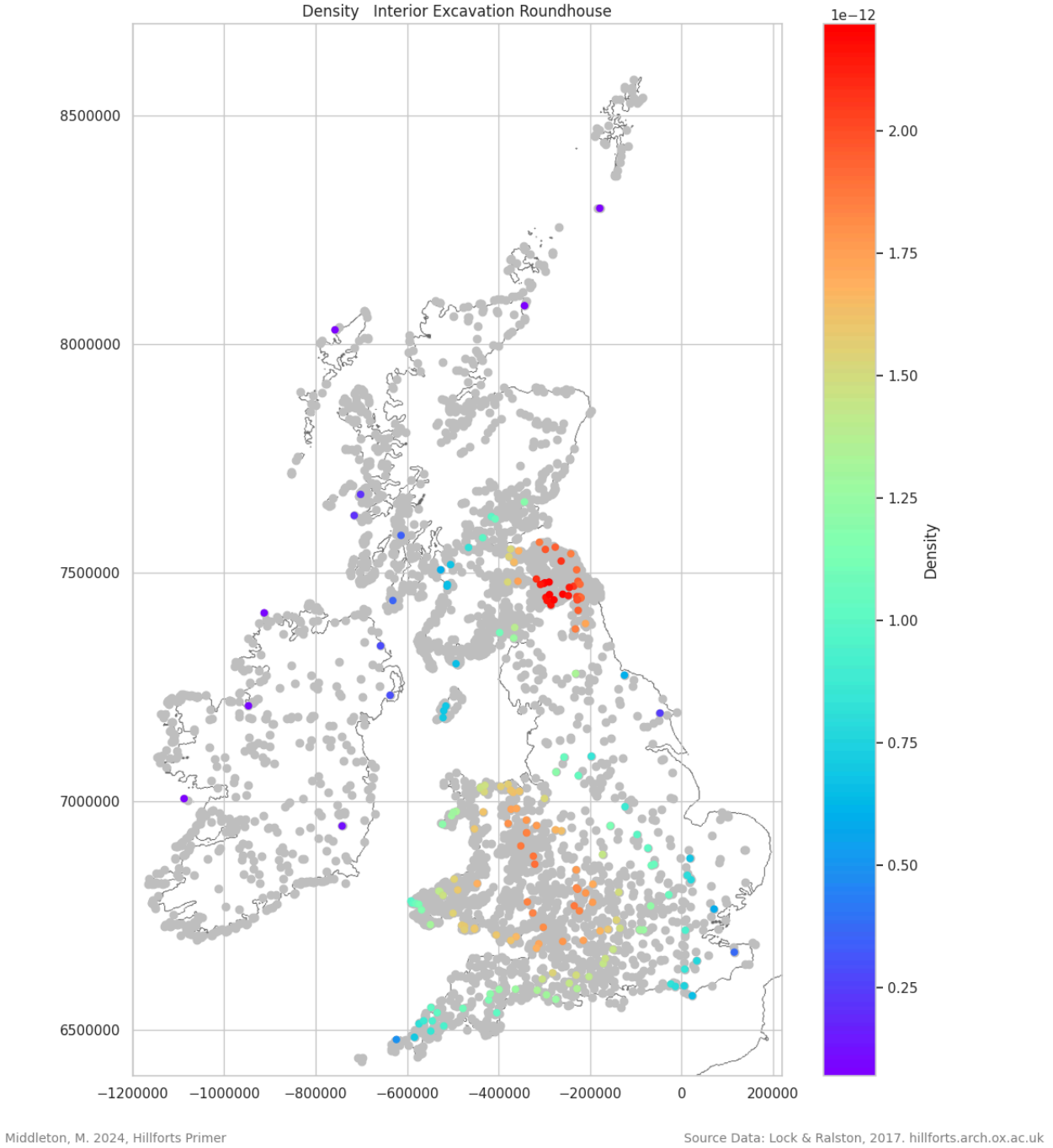


4.65%

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 cluser 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.

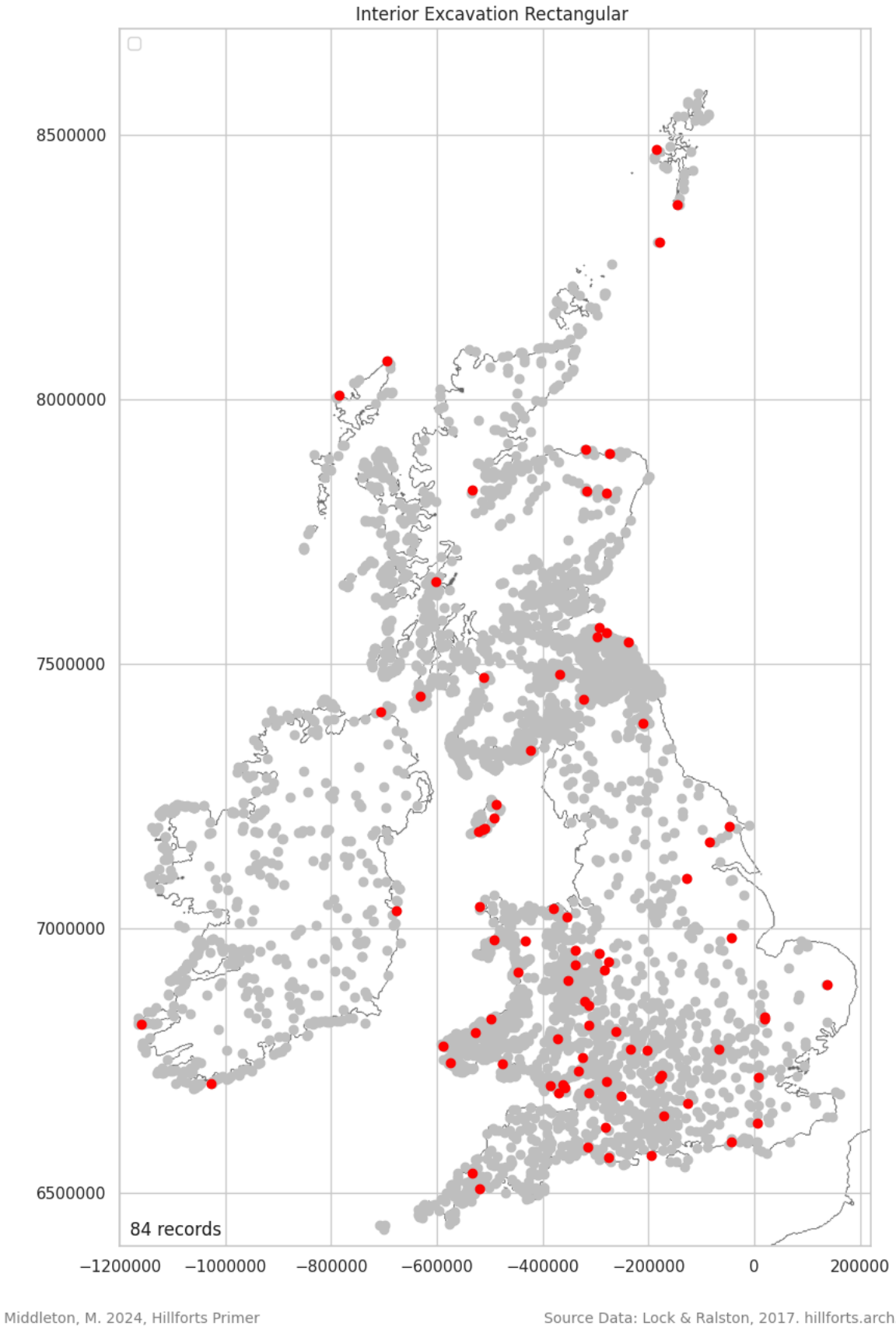
In [136…
```
plot_density_over_grey(int_ex_rh, 'Interior_Excavation_Roundhouse')
```



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_excavaion_data, 'Interior_Excavation_Rectangular', 'Yes')
```



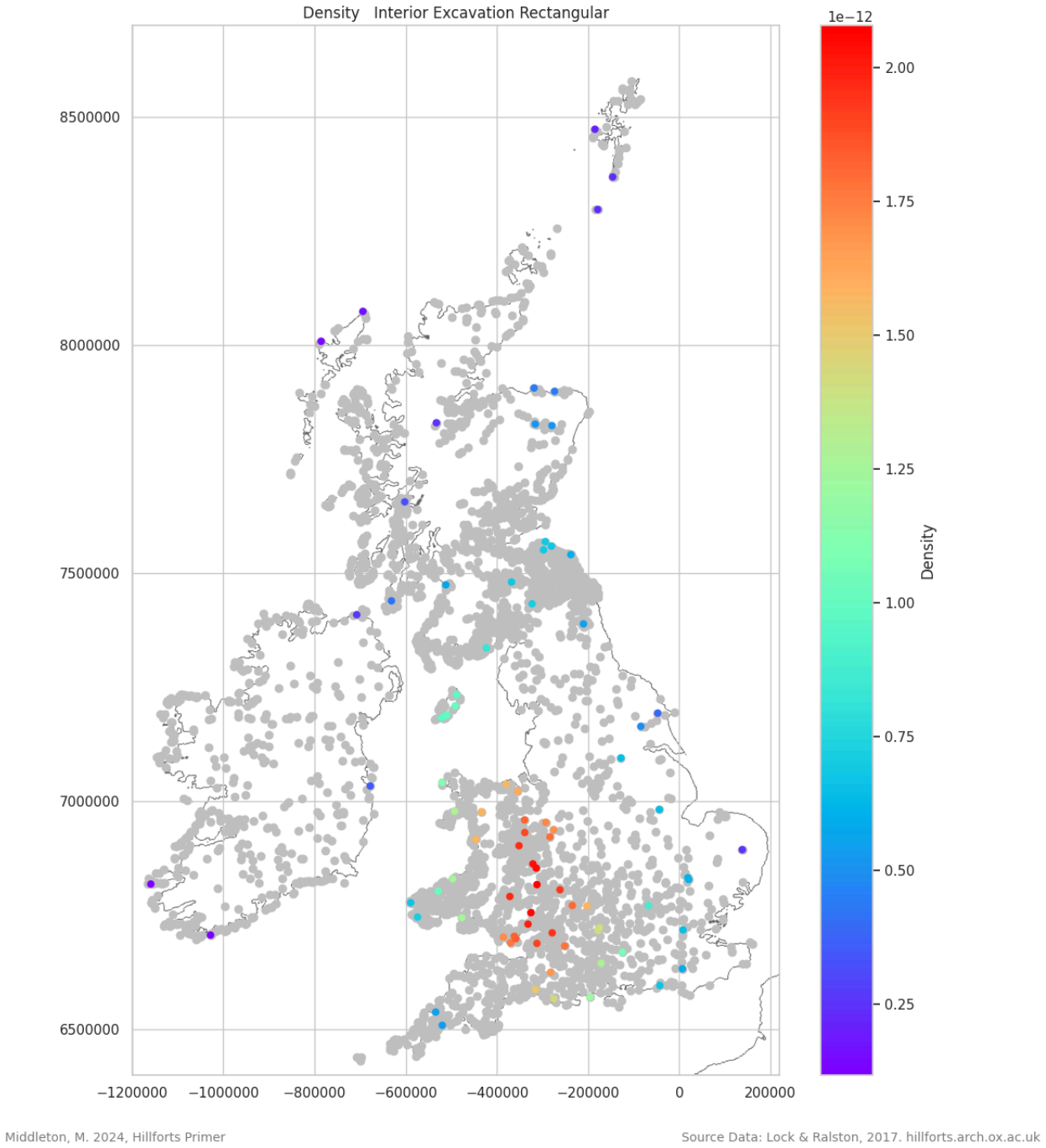
2.03%

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')
```

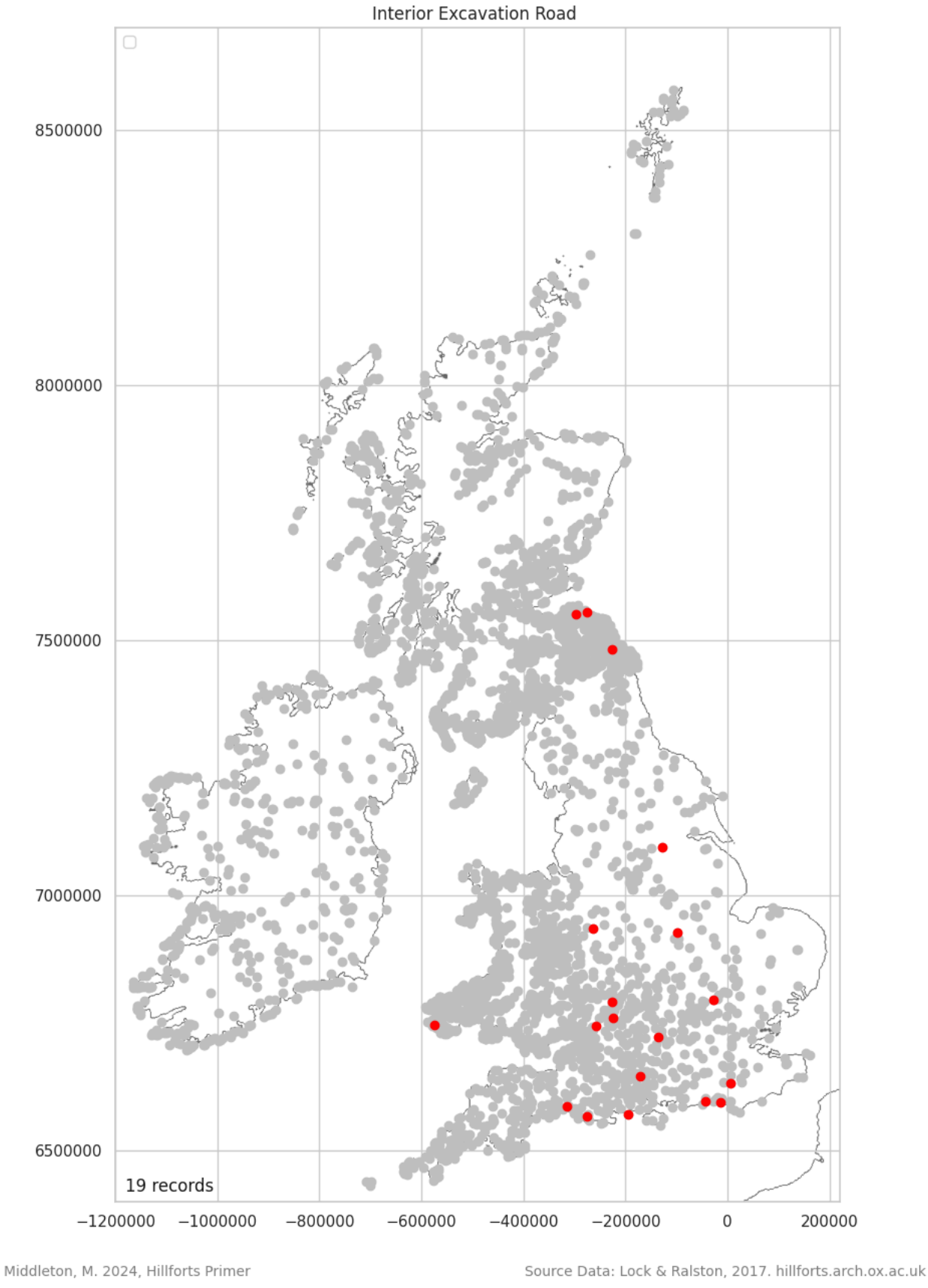


Excavaion: Road Mapped

Excavated examples of roads have been identified at 19 hillforts.

In [139…

```
int_ex_road = plot_over_grey(location_excavaion_data, 'Interior_Excavation_Road', 'Yes')
```



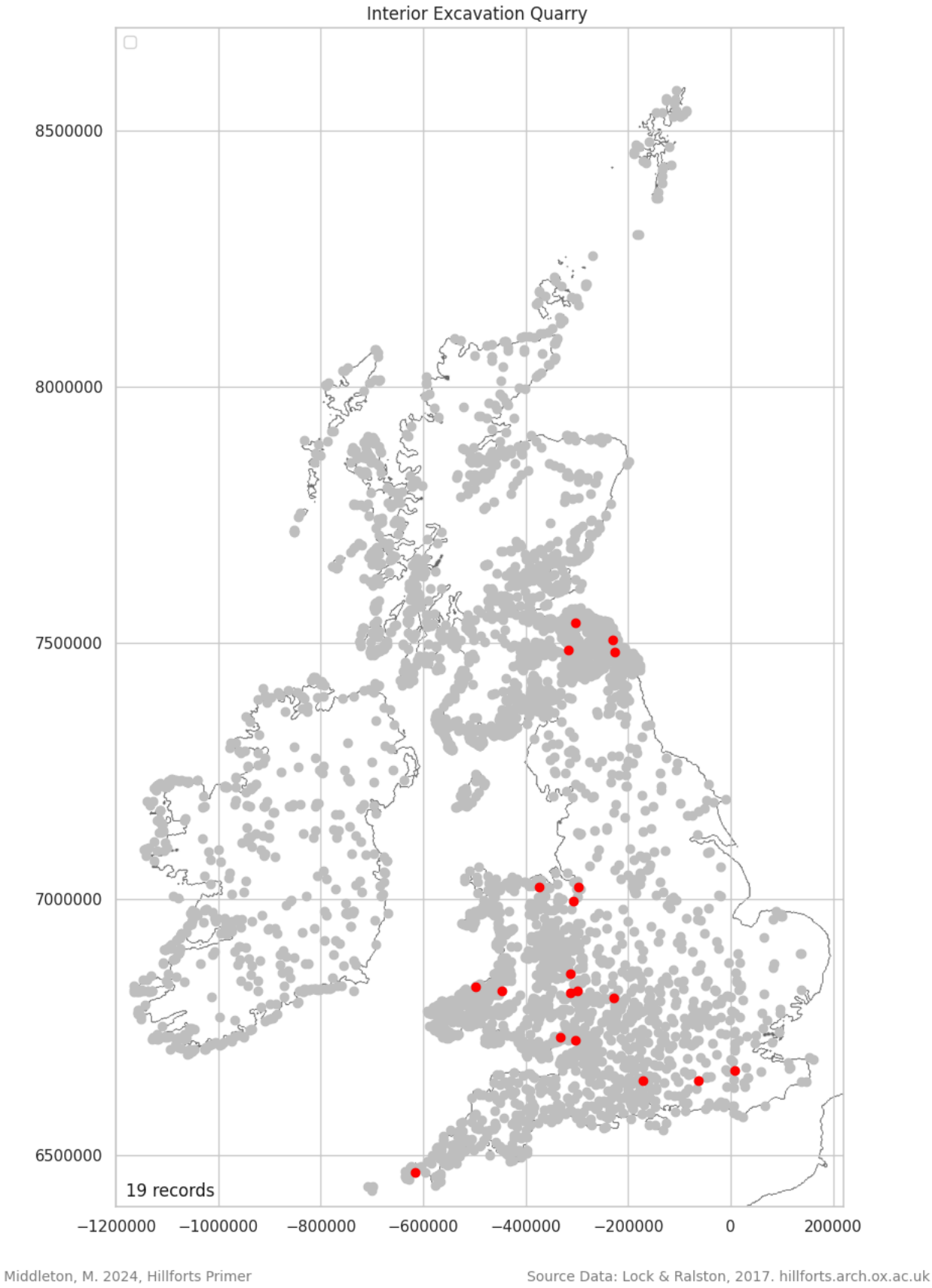
0.46%

Excavaion: Quarry Mapped

Excavated examples of quarries have been identified at 19 hillforts.

In [140…

```
int_ex_quarry = plot_over_grey(location_excavaion_data, 'Interior_Excavation_Quarry', 'Yes')
```



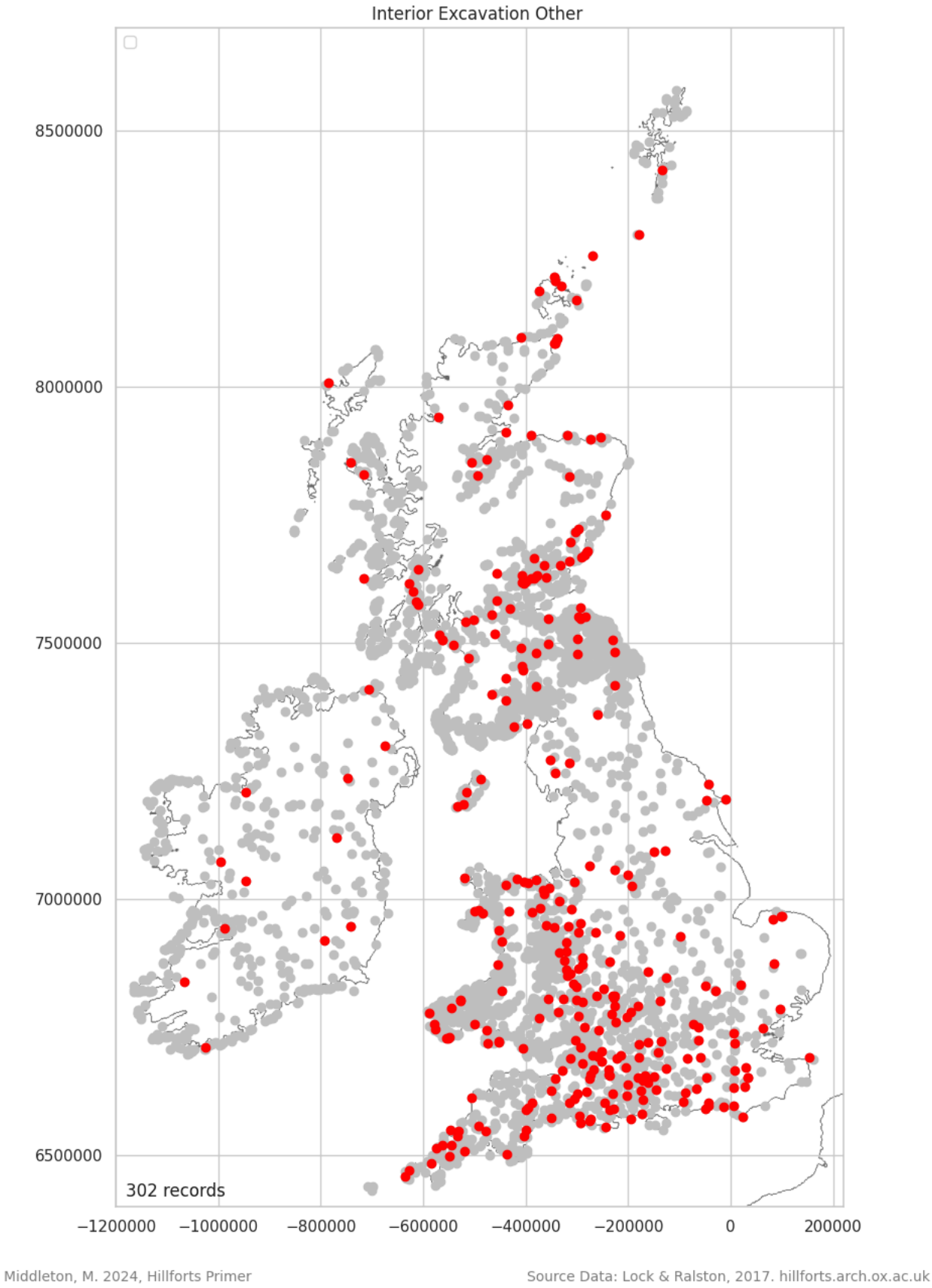
0.46%

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_excavaion_data, 'Interior_Excavation_Other', 'Yes')
```



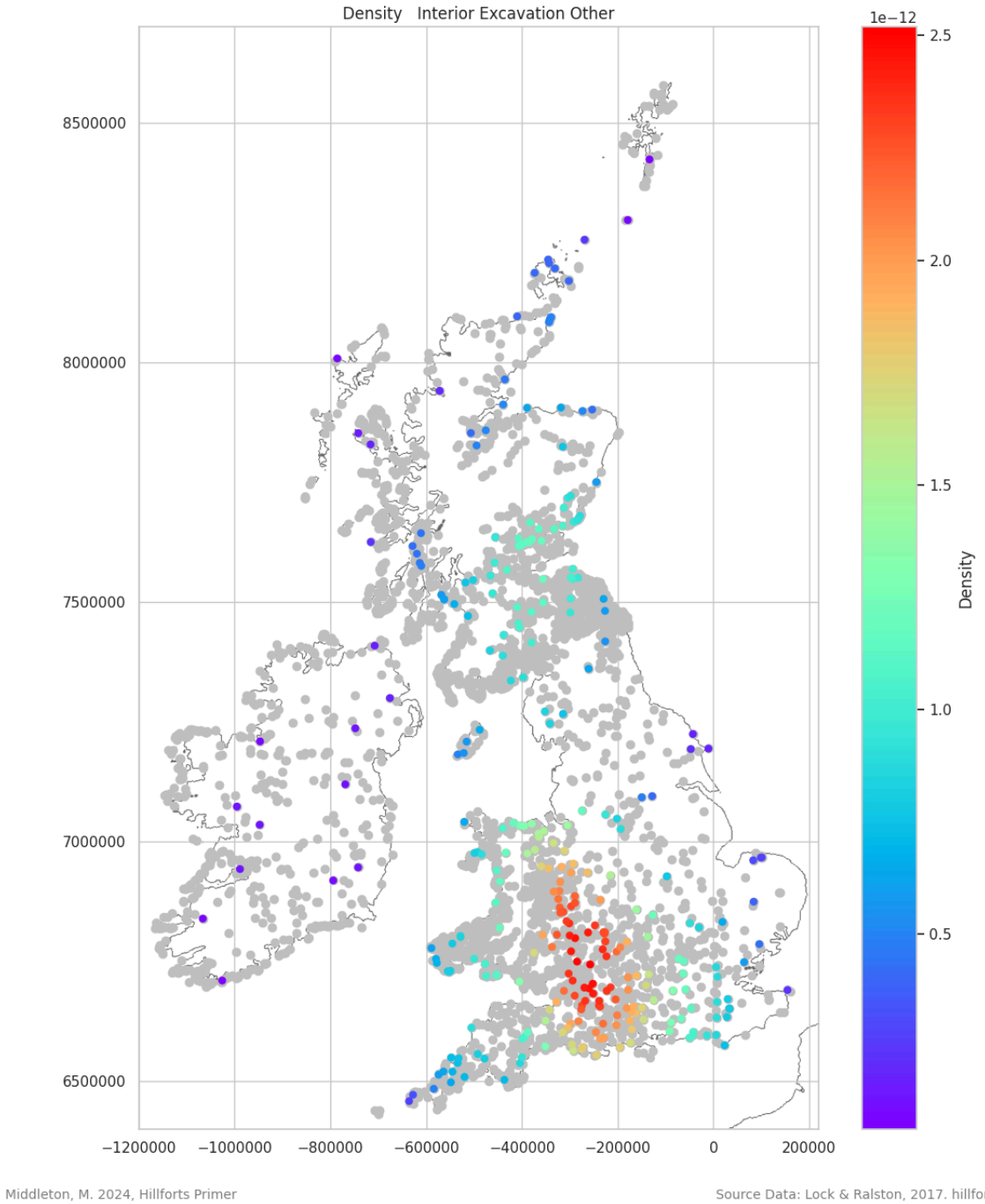
7.28%

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')
```

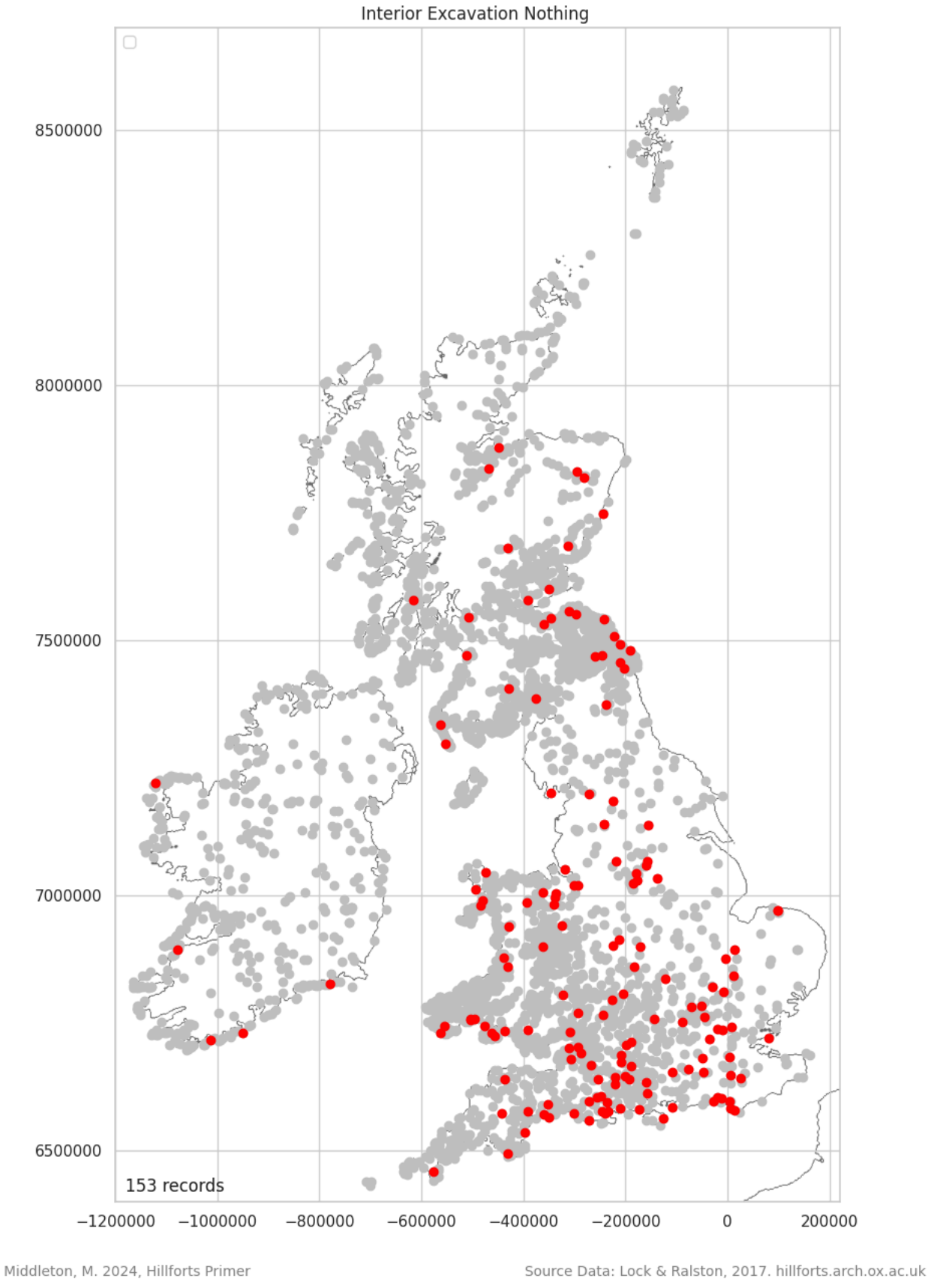


Excavaion: Nothing Mapped

3.69% of excavated hilloftes identified no internal structures. It is not clear if this is because the excavateions 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_excavaion_data, 'Interior_Excavation_Nothing', 'Yes')
```



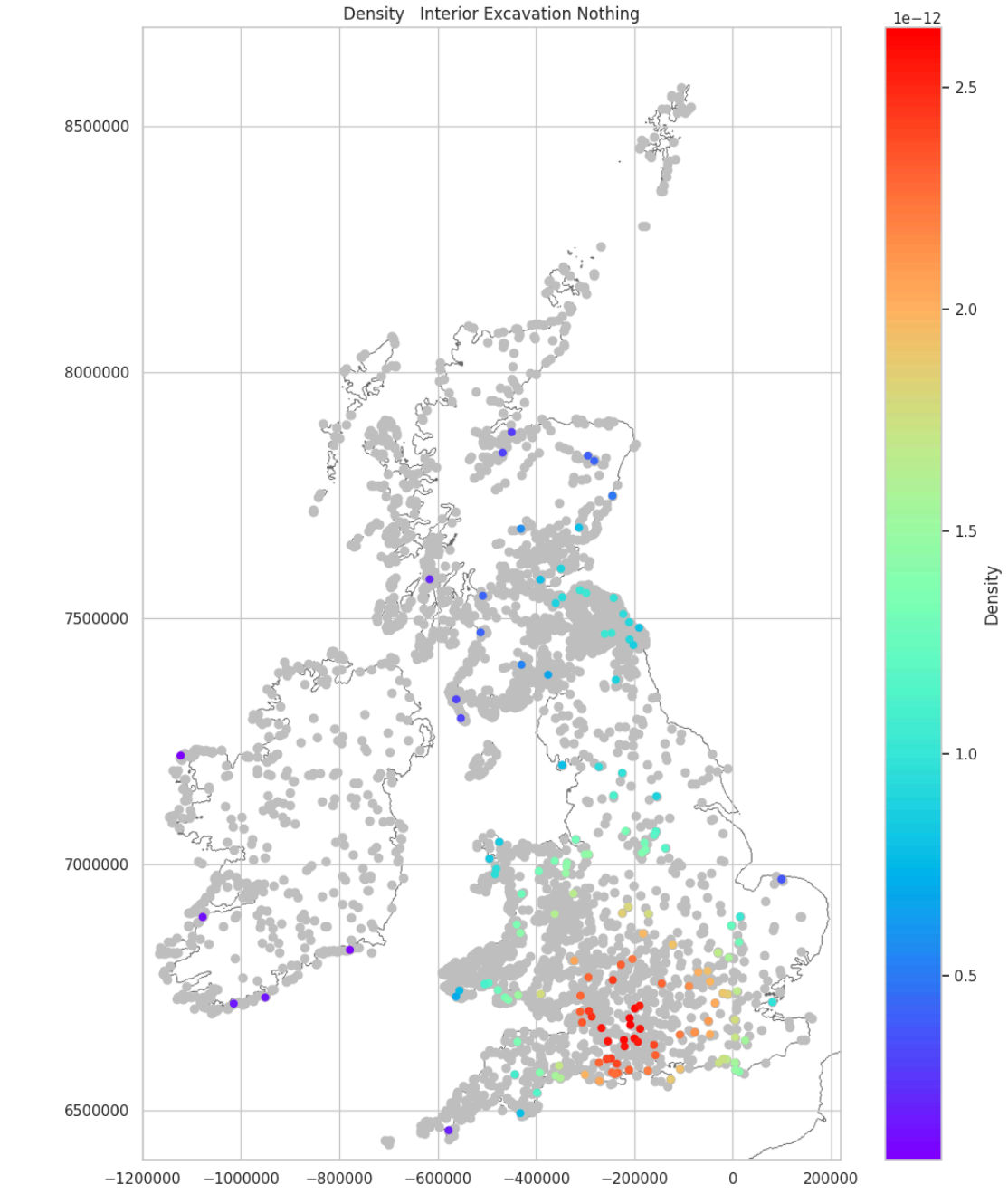
3.69%

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')
```



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()
```

Out[145]:

	Interior_Geophysics_None	Interior_Geophysics_Pit	Interior_Geophysics_Roundhouse	Interior_Geophysics_Rectangular	Interior_Geophysics_
0	Yes	No	No	No	
1	Yes	No	No	No	
2	Yes	No	No	No	
3	Yes	No	No	No	
4	Yes	No	No	No	

There are no null values

In [146...

```
geophysics_data.info()
```

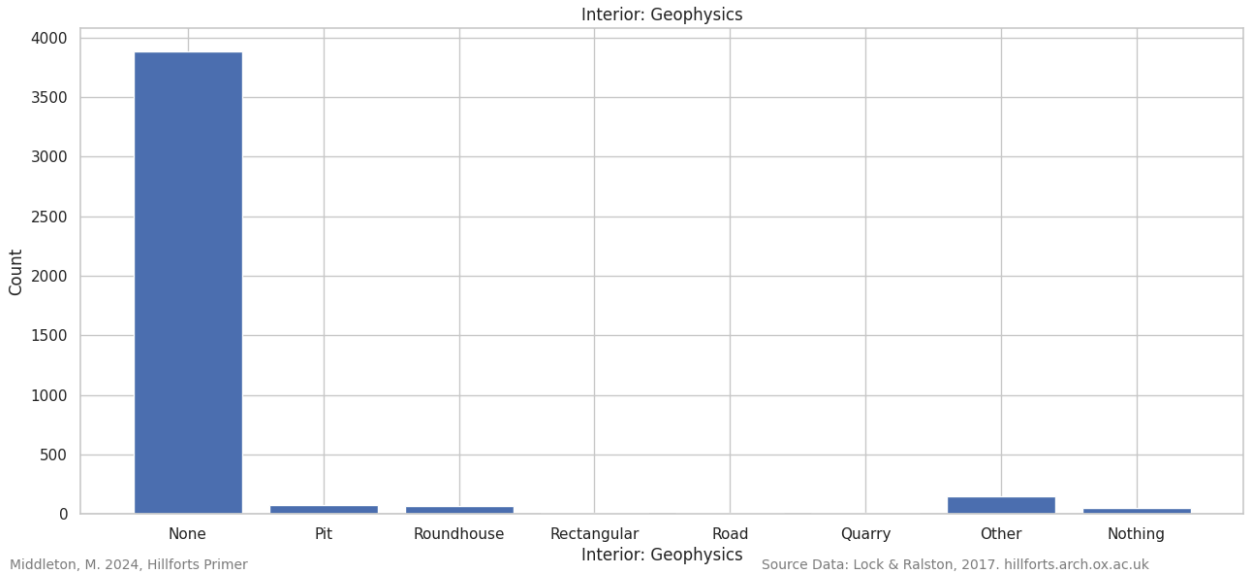
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Interior_Geophysics_None        4147 non-null   object
 1   Interior_Geophysics_Pit         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') dominats the geophysics plot and will be removed to facilate 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 dominent classes in the geophysics data.

In [148...

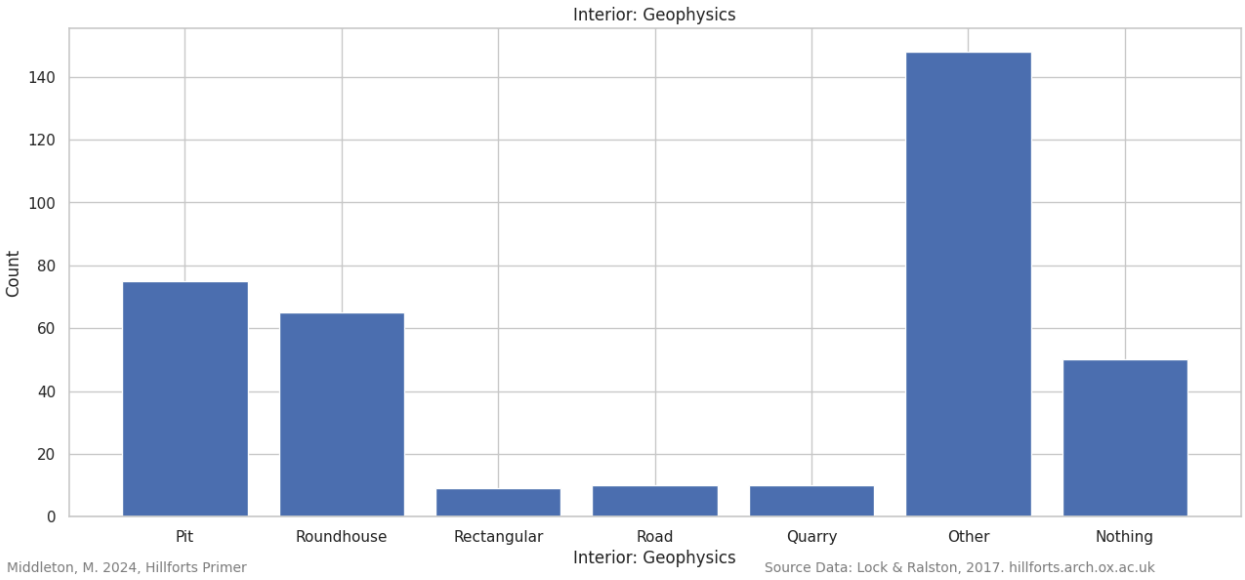
```
geophysics_data_minus = geophysics_data.drop(['Interior_Geophysics_None'], axis=1)
geophysics_data_minus.head()
```

Out[148]:

	Interior_Geophysics_Pit	Interior_Geophysics_Roundhouse	Interior_Geophysics_Rectangular	Interior_Geophysics_Road	Interior_Geophysics_(
0	No	No	No	No	
1	No	No	No	No	
2	No	No	No	No	
3	No	No	No	No	
4	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()
```

Out[150]:

	Interior_Geophysics_Pit	Interior_Geophysics_Roundhouse	Interior_Geophysics_Rectangular	Interior_Geophysics_Road	Interior_Geophysics_(
0	No	No	No	No	
1	No	No	No	No	
2	No	No	No	No	
3	No	No	No	No	
4	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…

```
geophyz_forts = 4147 - sum(geophysics_data['Interior_Geophysics_None']=="Yes")
geophyz_forts
```

Out[152]:

265

50 hillforts (18.87% of those surveyed) revealed no internal features.

In [153…
```
geophyz_nothing = sum(geophysics_data['Interior_Geophysics_Nothing']=="Yes")
geophyz_nothing
```

Out[153]:
```
50
```

In [154…
```
geophyz_nothing_pcnt = round((geophyz_nothing / geophyz_forts) * 100, 2)
geophyz_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")))
```

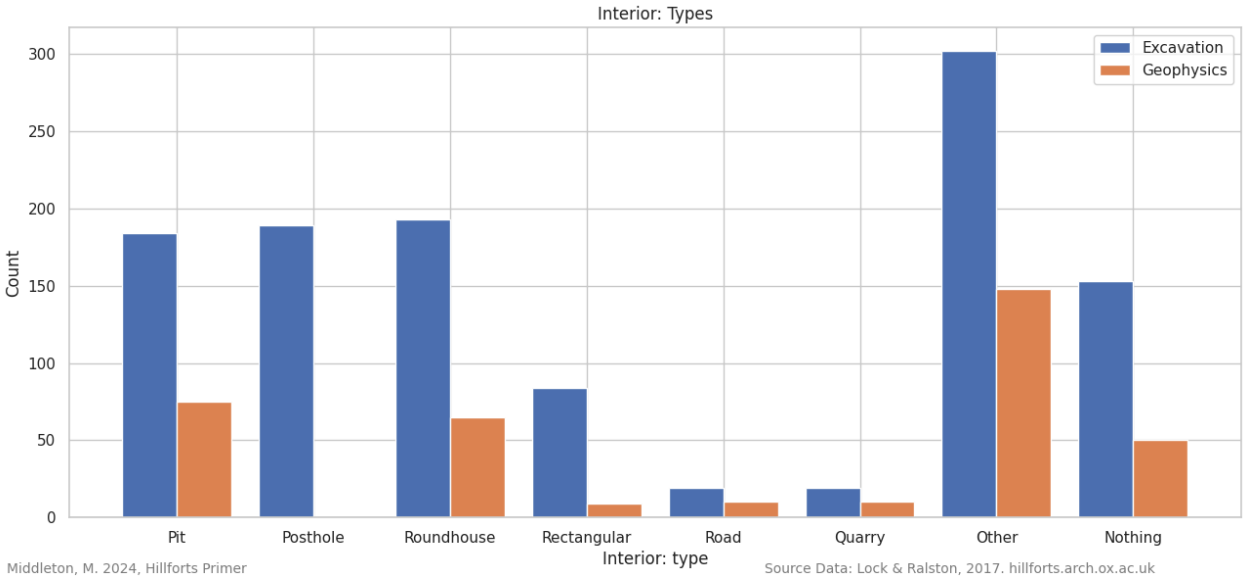
```
Interior_Geophysics_Pit: 75
Interior_Geophysics_Roundhouse: 65
Interior_Geophysics_Rectangular: 9
Interior_Geophysics_Road: 10
Interior_Geophysics_Quarry: 10
Interior_Geophysics_Other: 148
```

In [156…
```
geophyz_other_pcnt = round((sum(geophysics_data['Interior_Geophysics_Other']=="Yes") / geophyz_forts) * 100, 2)
geophyz_other_pcnt
```

Out[156]:
```
55.85
```

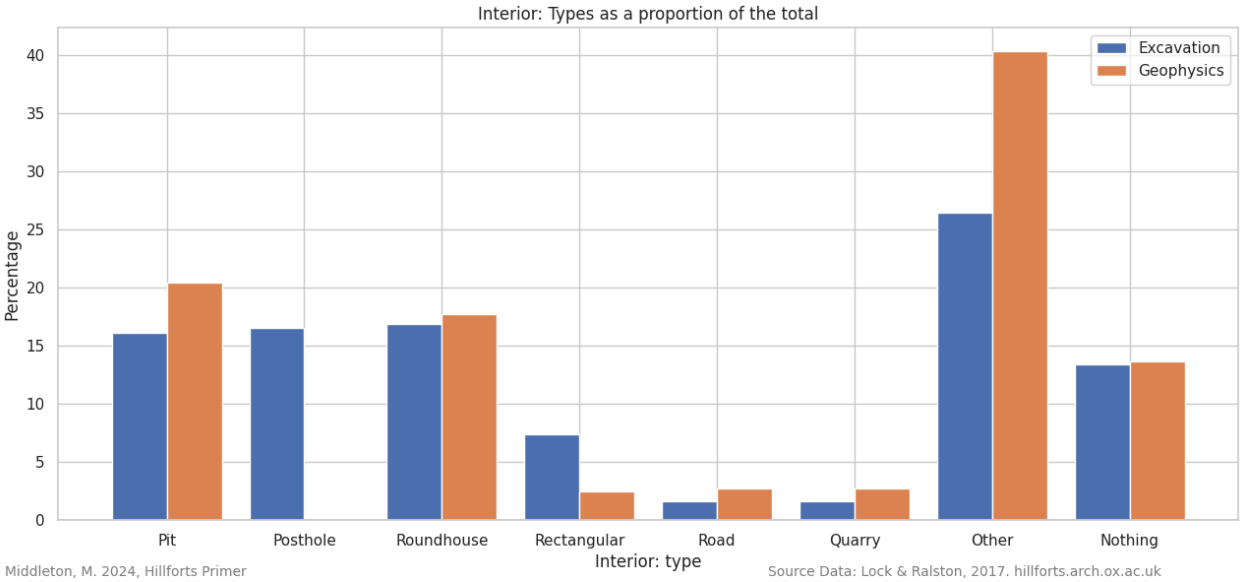
Excavations have found more of each structure because there have been more excavations.

In [157…
```
plot_bar_chart_two(excavation_data_minus, temp_geophysics, 2, 'Interior: type', 'Count', 'Interior: Types')
```



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' catagory. If this is the case, excavation and geophysics are identfying very similar proportions of features within hillforts. The difference in pits may possibley be accounted for by geophysics cataloging naturally occouring 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 interpet these differences.

In [159…

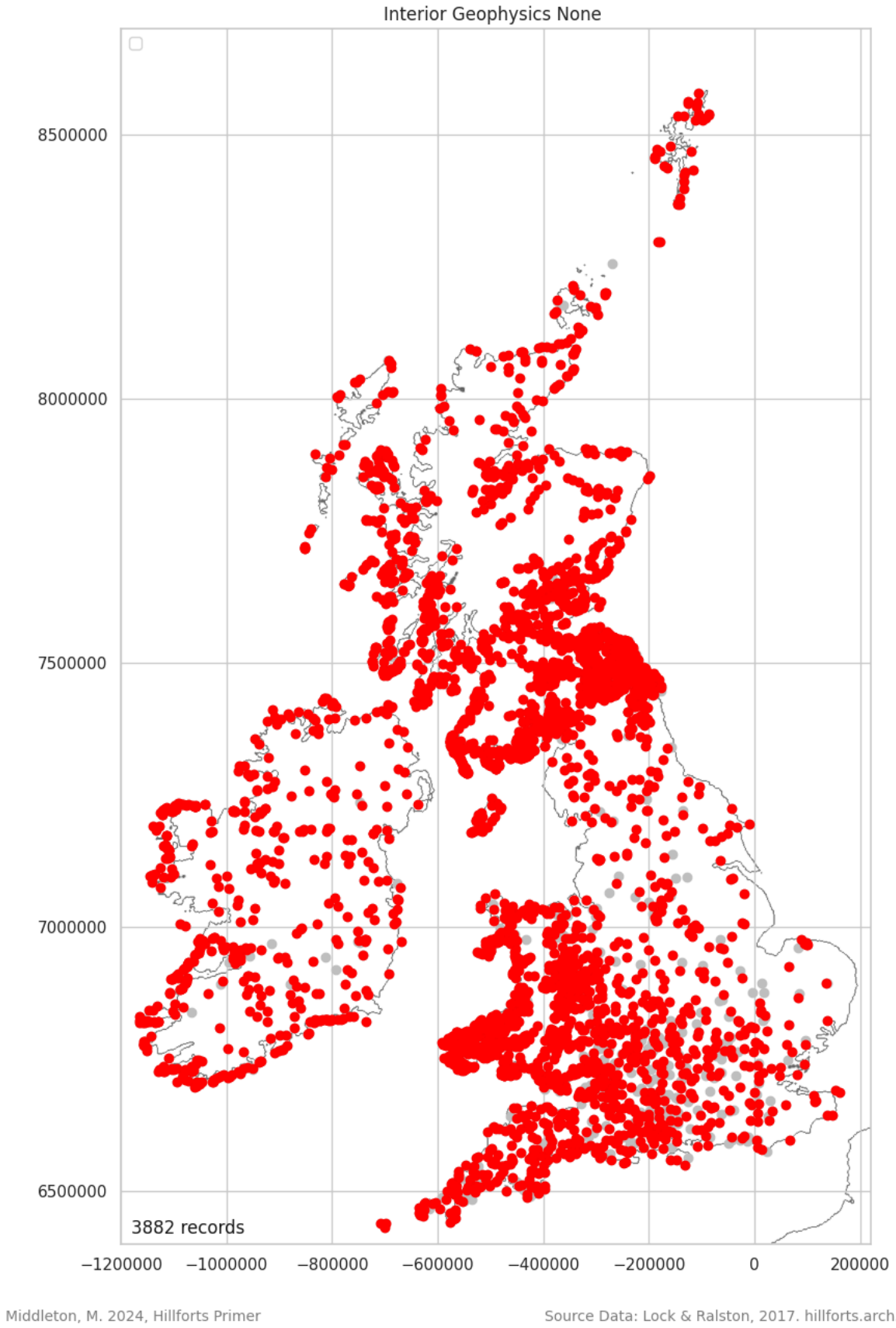
```
location_geophysics_data = pd.merge(location_numeric_data_short, geophysics_data, left_index=True, right_index=True)
```

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')
```



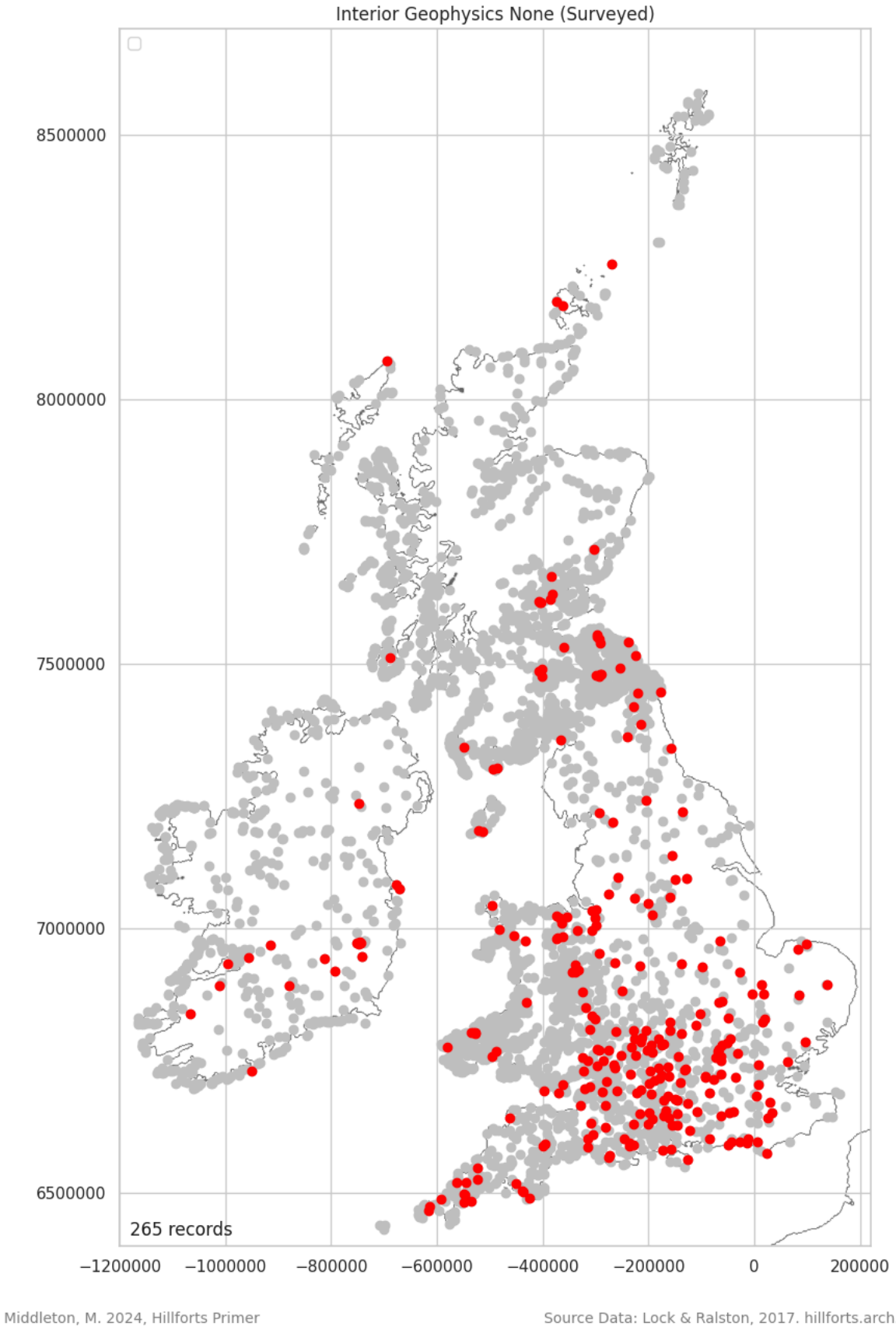
93.61%

Geophysics: None Mapped (Surveyed)

Simiar 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)")
```



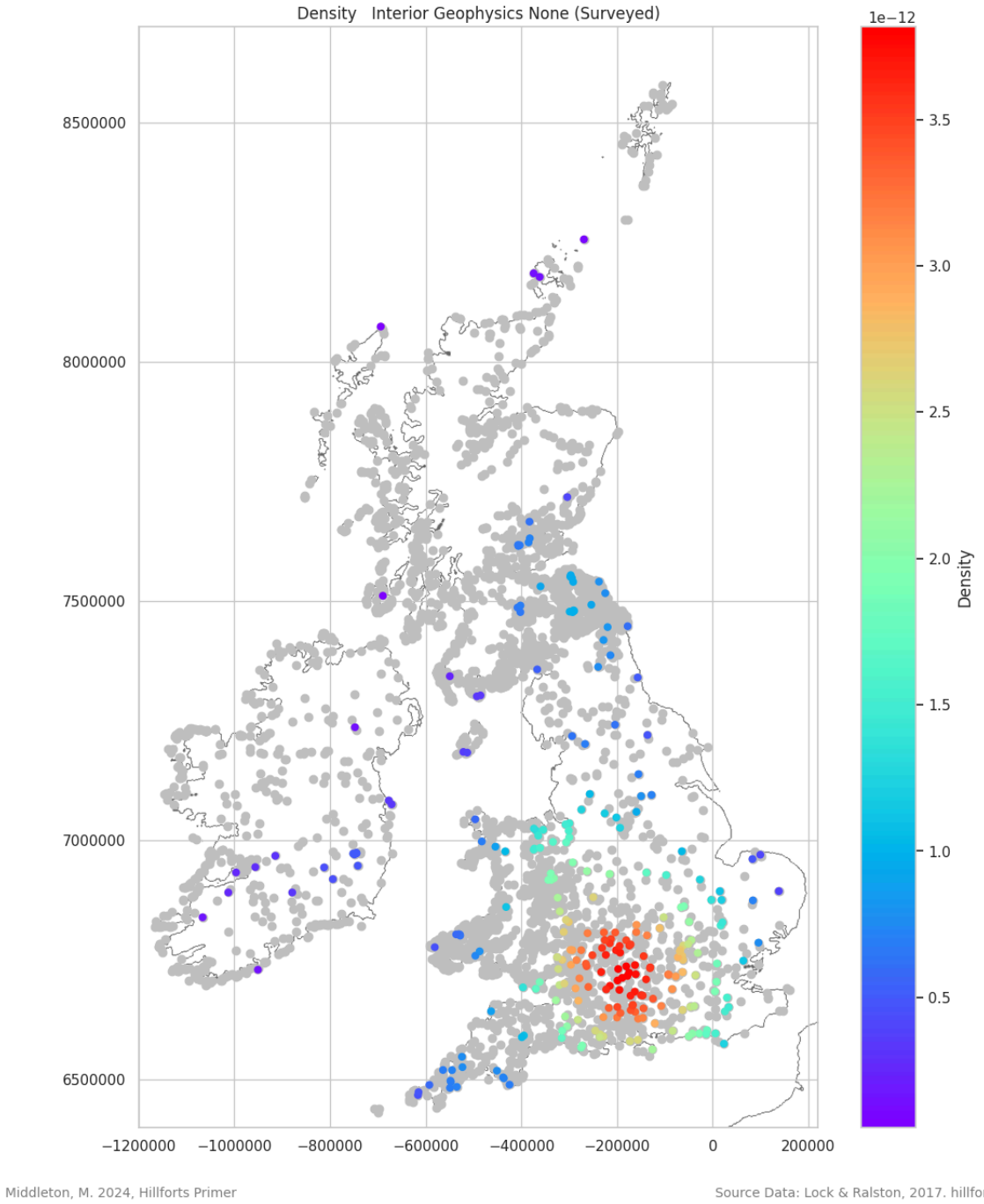
6.39%

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)')
```

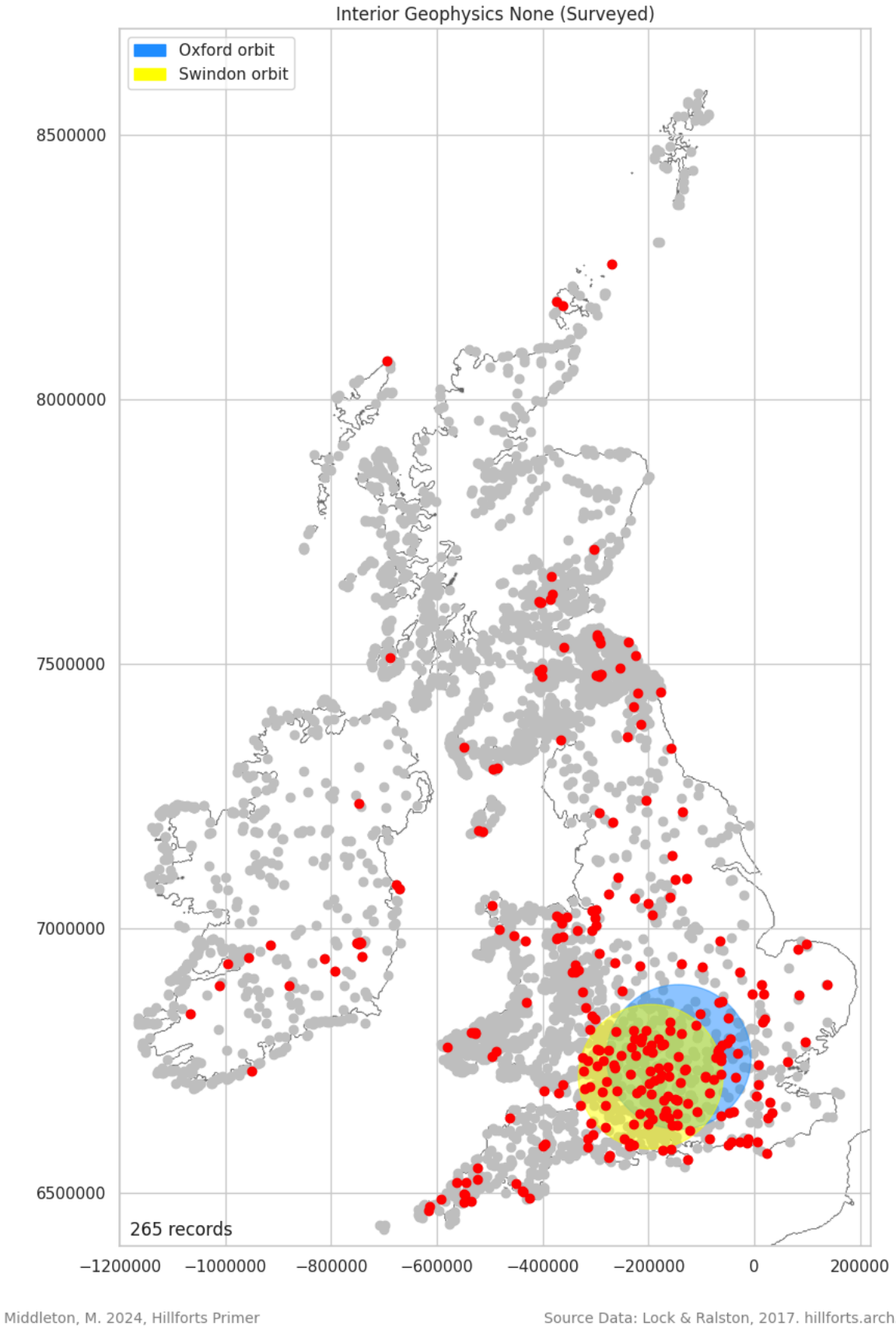


Geophysics: None Mapped (Surveyed) Plus Oxford and Swindon Orbits

There is a sygnificant survey bias. The most dense concentration of surveyed hillforts coincides with the overlapping orbits of Oxford University and the Histoiric England head office in Swindon.

In [163…

```
geophys_none = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_None', 'No', "(Surveyed)", False, False
```



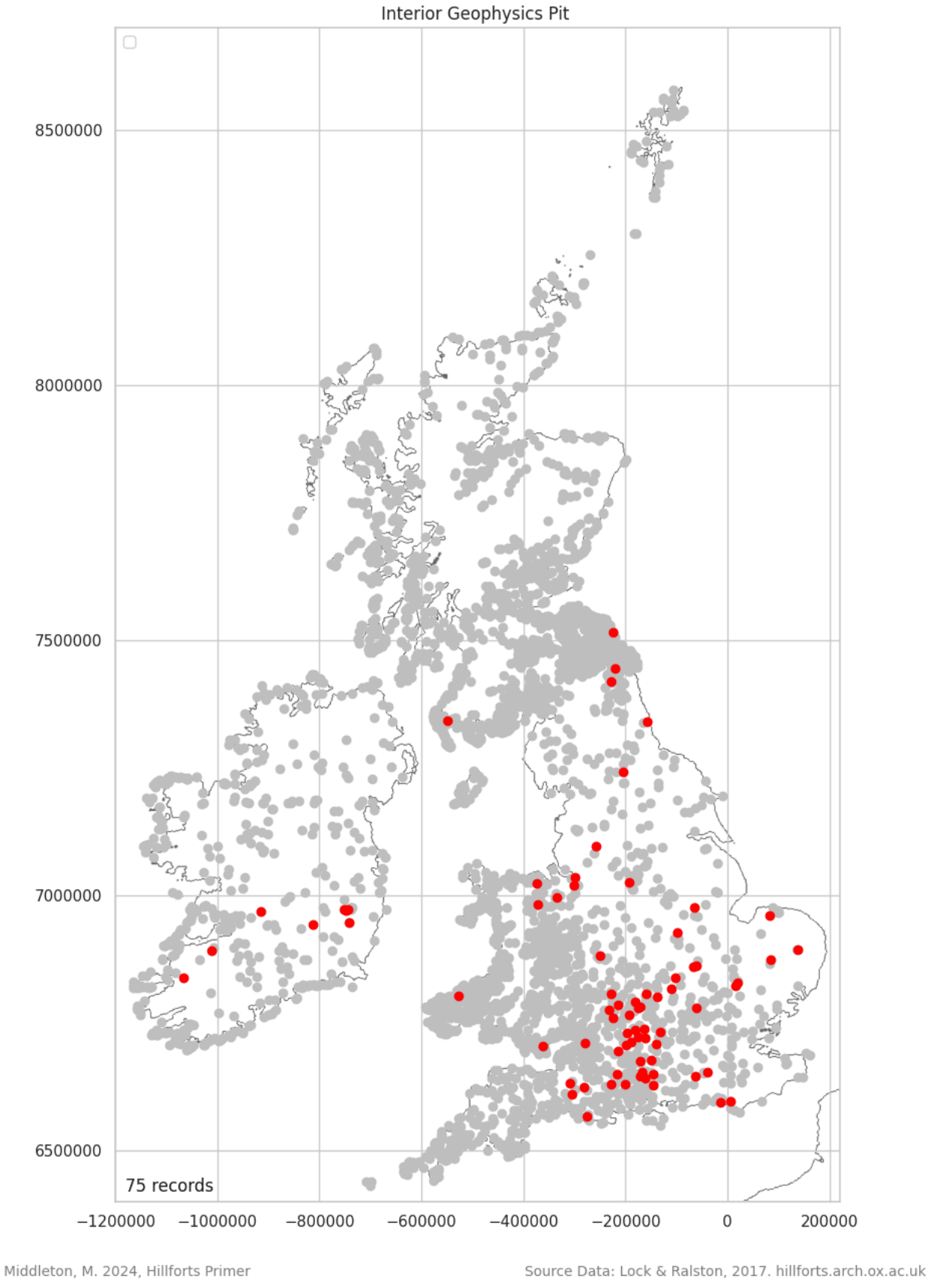
6.39%

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')
```



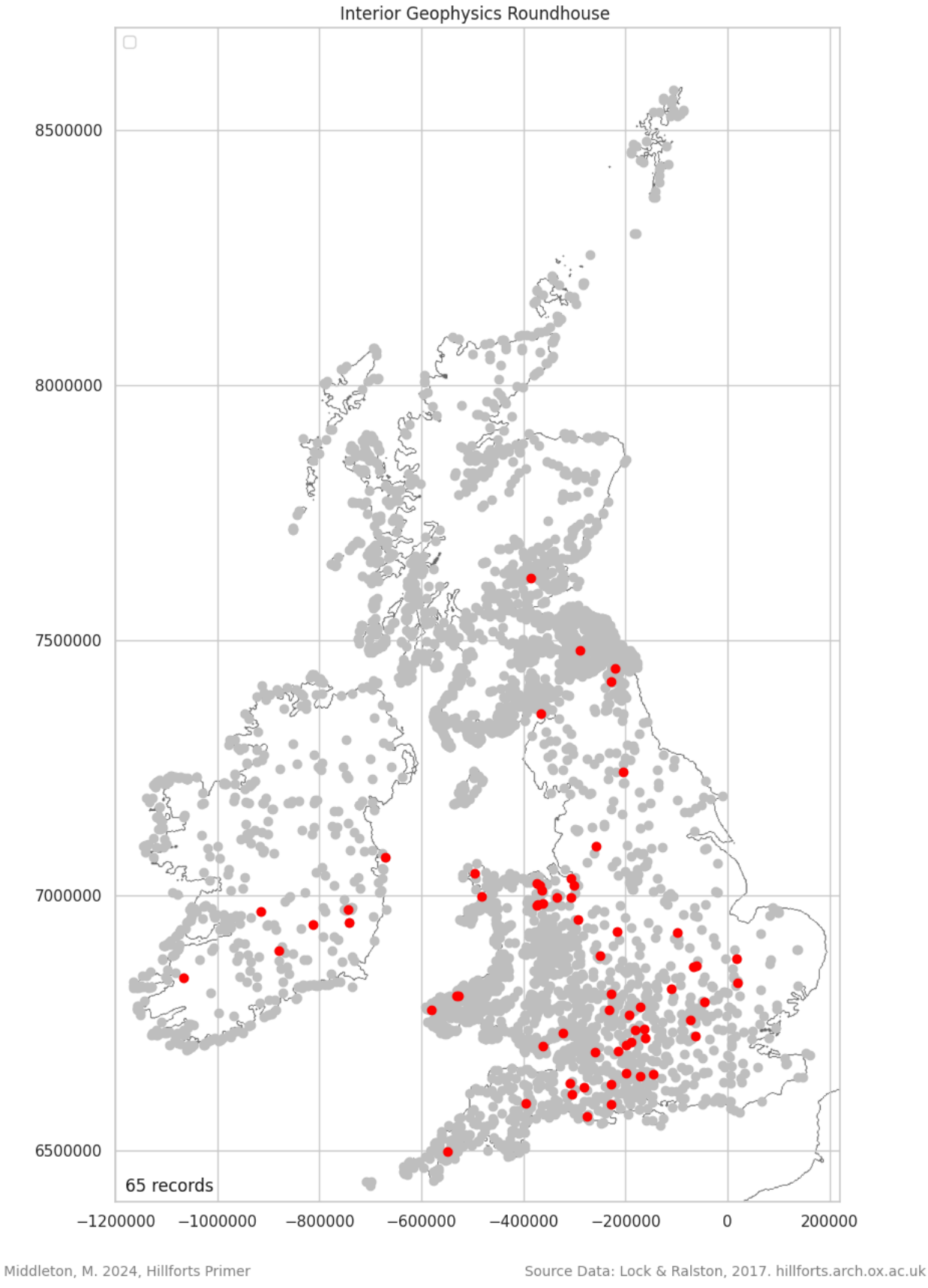
1.81%

Geophysics: Roundhouse Mapped

Roundhouses show the same bias as discussed in Geophysics: None Mapped (Surveyed) Plus Oxford and Swindon Orbits. It is noteable how different the distribution of roundhouses is in this small area to that discussed in Excavaion: Roundhouse Mapped.

In [165…

```
int_geo_rh = plot_over_grey(location_geophysics_data, 'Interior_Geophysics_Roundhouse', 'Yes')
```



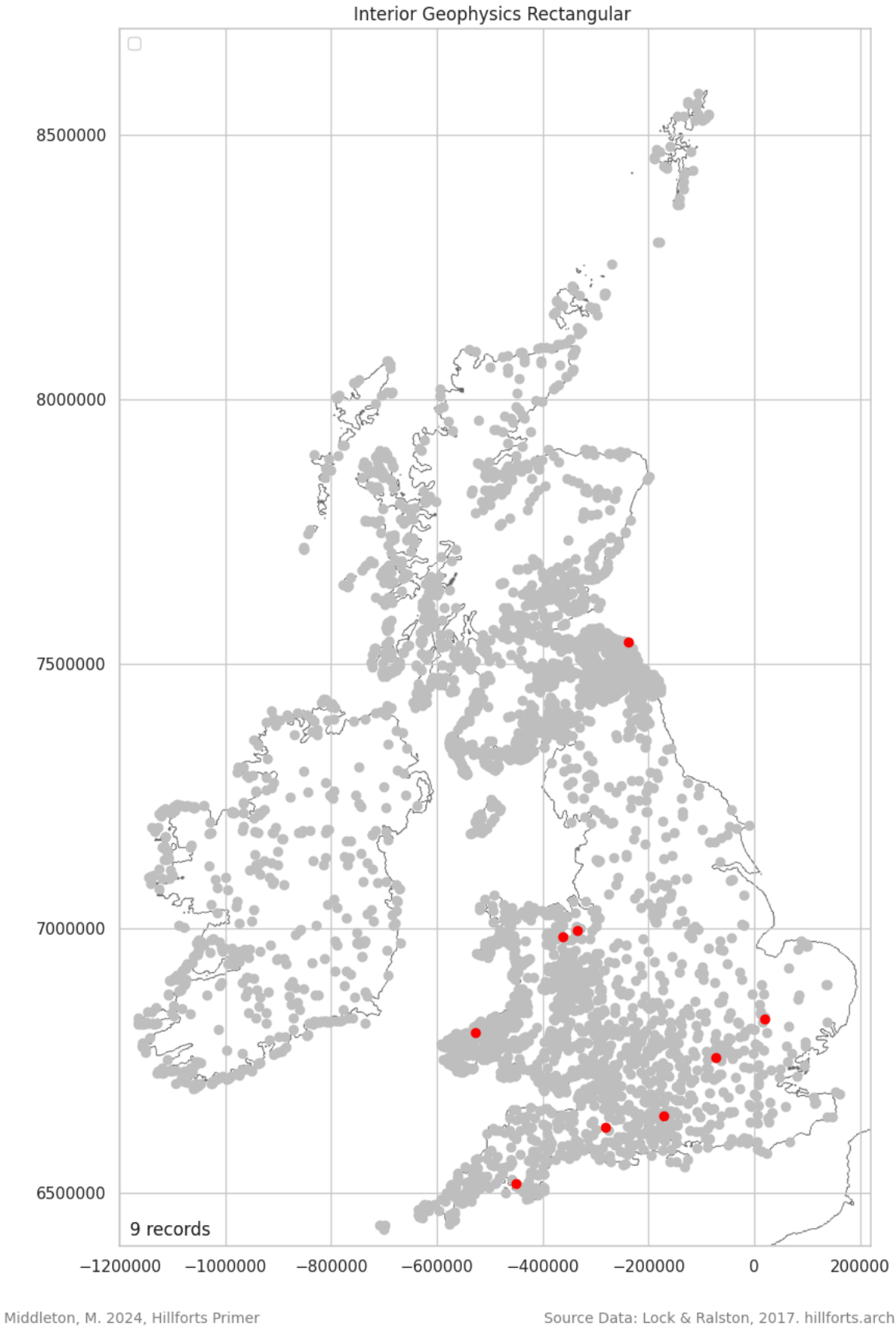
1.57%

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')
```



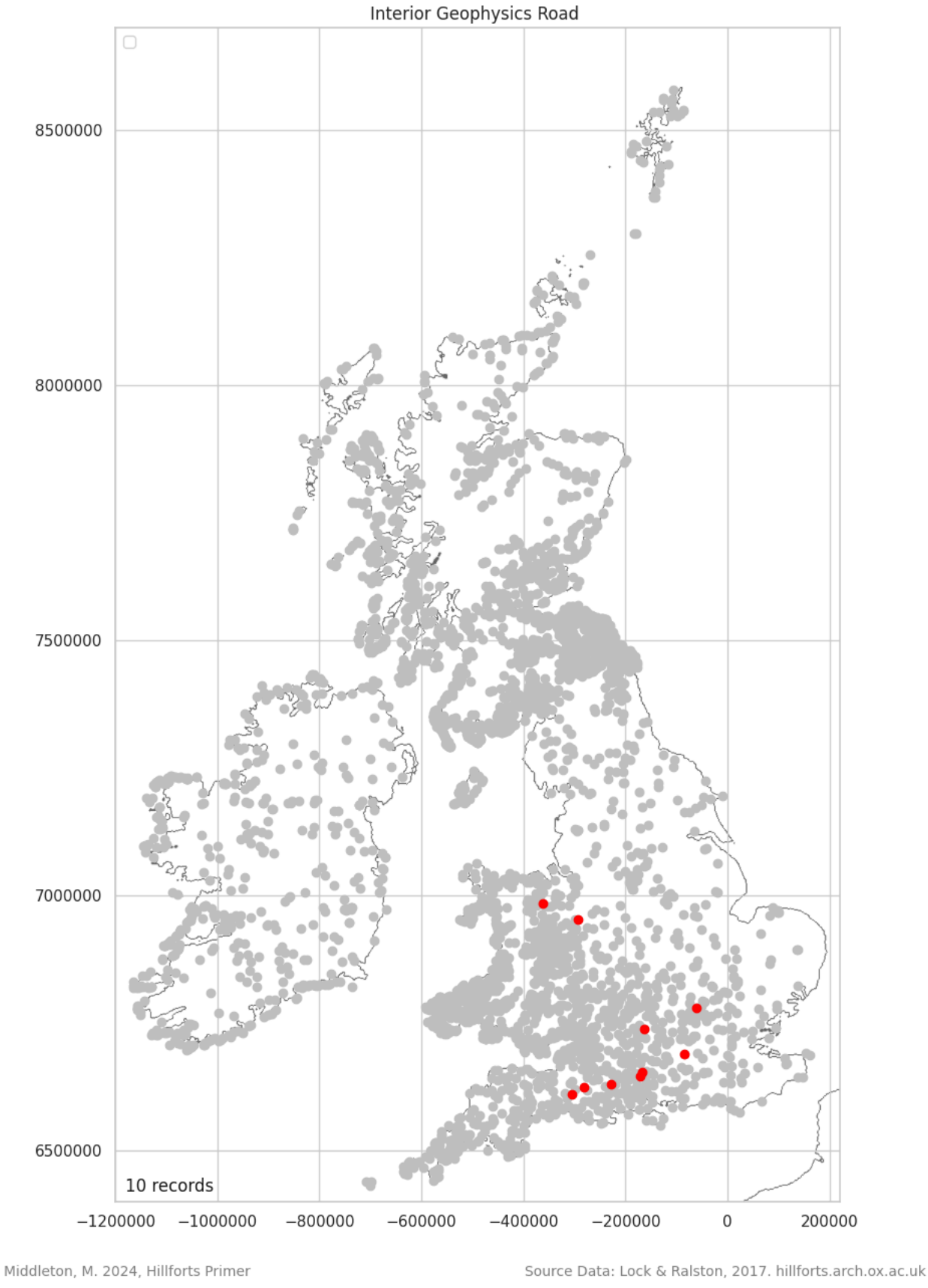
0.22%

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')
```



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')
```

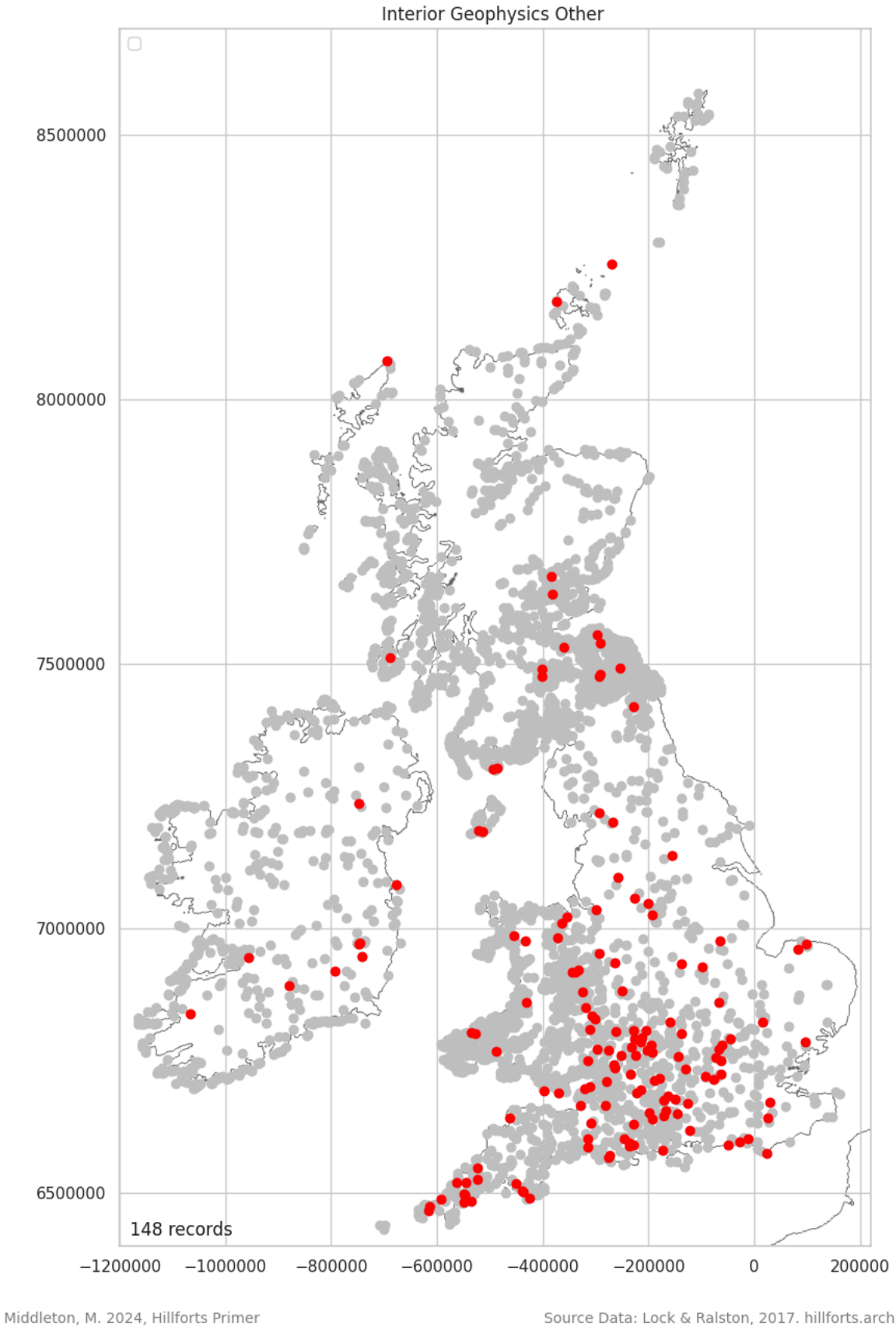

```
0.24%
```

Geophysics: Other Mapped

Other structures, identified in geophysiscs 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')
```



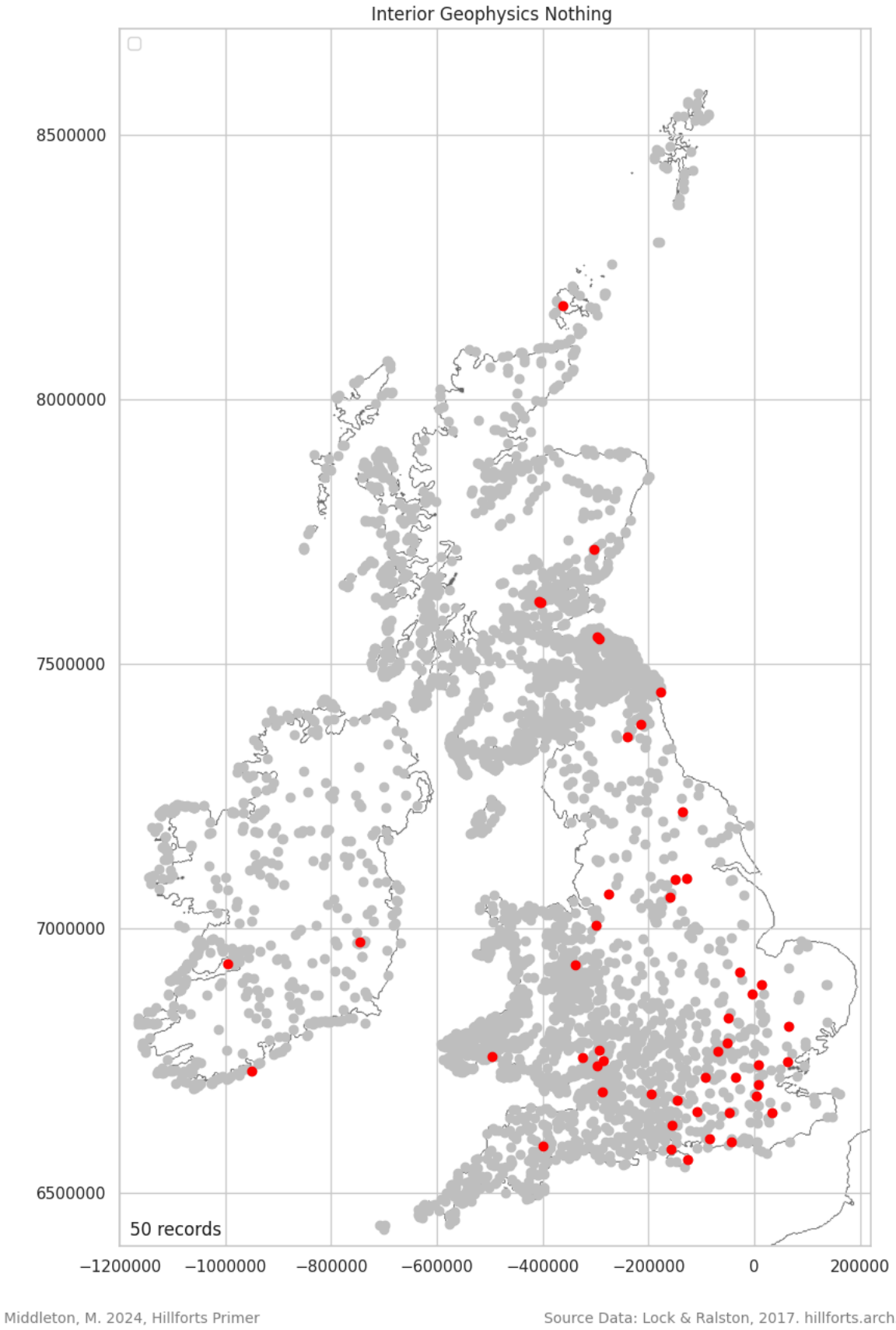
3.57%

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 intersting as it goes against what would be expected consdering 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')
```



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