Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 2

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

Version 1.0, March 2024.
This research was begun in March 2022.

Part 1: Name, Admin & Location Data

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

Part 2: Management & Landscape

Colab Notebook: Live code
HTML: Read only

- Management Data
- Landscape Data

Part 3: Boundary & Dating

Colab Notebook: Live code
HTML: Read only

Part 4: Investigations & Interior

Colab Notebook: Live code
HTML: Read only

Part 5: Entrance, Enclosing & Annex

Colab Notebook: Live code
HTML: Read only

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

Colab Notebook: Live code
HTML: Read only

Appendix 2: Classification Northwest

Colab Notebook: Live code
HTML: Read only

User Settings

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

To review only confirmed hillforts (see Part 1: Status, Data Reliability), download, save images or to change the background image to show the topography, first save a copy of this document into your Google Drive folder. Once saved, change, confirmed_only, download_data, save_images and/or show_topography to **True** in the code blocks below, **Save** and then select **Runtime>Run all** in the main menu above to rerun the code. If selected, running the code will initiate the download and saving of files. Each document will download a number of data packages and you may be prompted to **allow** multiple downloads. Be patient, downloads may take a little time after the document has finished running. Note that each part of the Hillforts Primer is independent and the download, save_image and show_topography variables will need to be enabled in each document, if this functionality is required. Also note that saving images will activate the Google Drive folder and this will request the user to **allow** access. Selecting show_topography will change the background image to a colour topographic map. It should also be noted that, if set to True, this view will only show the distribution of the data selected. It will not show the overall distribution as a grey background layer as is seen when using the simple coastal outlines.

In [ ]:
```
confirmed_only = False
```

In [ ]:
```
download_data = False
```

In [ ]:
```
save_images = False
```

In [ ]:
```
show_topography = False
```

Bypass Code Setup

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

Go to Review Data Part 2.

Source Data

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

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

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

Reload Data and Python Functions

This study is split over multiple documents. Each file needs to be configured and have the source data imported. As this section does not focus on the assessment of the data it is minimised to facilitate the documents readability.

Python Modules and Code Setup

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

In [ ]:
```
import sys
print(f'Python: {sys.version}')

import sklearn
print(f'Scikit-Learn: {sklearn.__version__}')

import pandas as pd
print(f'pandas: {pd.__version__}')

import numpy as np
print(f'numpy: {np.__version__}')

%matplotlib inline
import matplotlib
print(f'matplotlib: {matplotlib.__version__}')
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.patches as mpatches
import matplotlib.patches as patches
from matplotlib.cbook import boxplot_stats
from matplotlib.lines import Line2D
import matplotlib.cm as cm

import seaborn as sns
print(f'seaborn: {sns.__version__}')
sns.set(style="whitegrid")

import scipy
print(f'scipy: {scipy.__version__}')
from scipy import stats
from scipy.stats import gaussian_kde

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

from slugify import slugify

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

```
Python: 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
Scikit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.25.2
matplotlib: 3.7.1
seaborn: 0.13.1
scipy: 1.11.4
```

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

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

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

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

Ref: https://matplotlib.org/

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

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

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

Plot Figures and Maps functions

The following functions will be used to plot data later in the document.

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

In [ ]:
```
def get_backgrounds():
    if show_topography == True:
        backgrounds = ["hillforts-topo-01.png",
                       "hillforts-topo-north.png",
                       "hillforts-topo-northwest-plus.png",
```

```
                    "hillforts-topo-northwest-minus.png",
                    "hillforts-topo-northeast.png",
                    "hillforts-topo-south.png",
                    "hillforts-topo-south-plus.png",
                    "hillforts-topo-ireland.png",
                    "hillforts-topo-ireland-north.png",
                    "hillforts-topo-ireland-south.png"]
    else:
        backgrounds = ["hillforts-outline-01.png",
                    "hillforts-outline-north.png",
                    "hillforts-outline-northwest-plus.png",
                    "hillforts-outline-northwest-minus.png",
                    "hillforts-outline-northeast.png",
                    "hillforts-outline-south.png",
                    "hillforts-outline-south-plus.png",
                    "hillforts-outline-ireland.png",
                    "hillforts-outline-ireland-north.png",
                    "hillforts-outline-ireland-south.png"]
    return backgrounds
```

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

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

    if location == "n":
        background = os.path.join(folder, backgrounds[1])
        bounds = bounds[1]
    elif location == "nw+":
        background = os.path.join(folder, backgrounds[2])
        bounds = bounds[2]
    elif location == "nw-":
        background = os.path.join(folder, backgrounds[3])
        bounds = bounds[3]
    elif location == "ne":
        background = os.path.join(folder, backgrounds[4])
        bounds = bounds[4]
    elif location == "s":
        background = os.path.join(folder, backgrounds[5])
        bounds = bounds[5]
    elif location == "s+":
        background = os.path.join(folder, backgrounds[6])
        bounds = bounds[6]
    elif location == "i":
        background = os.path.join(folder, backgrounds[7])
        bounds = bounds[7]
    elif location == "in":
        background = os.path.join(folder, backgrounds[8])
        bounds = bounds[8]
    elif location == "is":
        background = os.path.join(folder, backgrounds[9])
        bounds = bounds[9]
    else:
        background = os.path.join(folder, backgrounds[0])
        bounds = bounds[0]

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

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

In [ ]:
```
def add_annotation_plot(ax):
    ax.annotate("Middleton, M. 2024, Hillforts Primer", size='small', \
                color='grey', xy=(0.01, 0.01), xycoords='figure fraction', \
```

```
                horizontalalignment = 'left')
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", \
                size='small', color='grey', xy=(0.99, 0.01), \
                xycoords='figure fraction', horizontalalignment = 'right')
```

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

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

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

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

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

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

```
            n_bins = int(data_range)/10

    return n_bins
```

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

In [ ]:
```
def plot_continuous(data, x_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 [ ]:
```
# 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 [ ]:
```
def location_XY_plot():
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(6400000,8700000)
    add_annotation_l_xy(plt)
```

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

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

In [ ]:
```
def plot_over_grey_numeric(merged_data, a_type, title, extra="", inner=False, \
                           fringe=False, oxford=False,swindon=False):
    plot_data = merged_data
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    add_grey()
    patches = add_oxford_swindon(oxford,swindon)
    plt.scatter(plot_data['Location_X'], plot_data['Location_Y'], c='Red')
    if fringe:
```

```
        f_for_legend = add_21Ha_fringe()
        patches.append(f_for_legend)
    if inner:
        i_for_legend = add_21Ha_line()
        patches.append(i_for_legend)
    show_records(plt, plot_data)
    plt.legend(loc='upper left', handles= patches)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

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

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

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

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

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

In [ ]:
```
def plot_over_grey(merged_data, a_type, yes_no, extra="", inner=False, \
                   fringe=False, oxford=False, swindon=False):
    # plots selected data over the grey dots. yes_no 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)/4147)*100), 2)}%')
    return plot_data
```

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

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

In [ ]:
```
def spider_plot(counts, title):
    df = pd.DataFrame(counts)
    categories=list(df)[1:]
    categories = [x.split("_")[2] for x in categories]
    N = len(categories)
    values=df.loc[0].drop('group').values.flatten().tolist()
    values += values[:1]
    angles = [n / float(N) * 2 * math.pi for n in range(N)]
    angles += angles[:1]

    fig = plt.figure(figsize=(8,8))
    ax = plt.subplot(111, polar=True)
    plt.xticks(angles[:-1], categories, color='black', size=12)
    ax.set_rlabel_position(0)
    plt.yticks([100,200,300], ["100","200","300"], color="grey", size=7)
    plt.ylim(0,300)

    ax.plot(angles, values, linewidth=2, color= "r", linestyle='solid')
    ax.fill(angles, values, 'r', alpha=0.1)
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

Review Data Functions

The following functions will be used to confirm that features are not lost or forgotten when splitting the data.

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

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

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

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

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

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

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

Null Value Functions

The following functions will be used to update null values.

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

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

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

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

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

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

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

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

Reprocessing Functions

In [ ]:
```
def add_density(data):
    new_data = data.copy()
    xy = np.vstack([new_data['Location_X'],new_data['Location_Y']])
    new_data['Density'] = gaussian_kde(xy)(xy)
    return new_data
```

Save Image Functions

In [ ]:
```
fig_no = 0
part = 'Part02'
IMAGES_PATH = r'/content/drive/My Drive/'
fig_list = pd.DataFrame(columns=['fig_no', 'file_name', 'title'])
topo_txt = ""
if show_topography:
    topo_txt = "-topo"
```

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

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

In [ ]:
```
def format_figno(no):
    length = len(str(no))
    fig_no = ''
    for i in range(3-length):
        fig_no = fig_no + '0'
    fig_no = fig_no + str(no)
    return fig_no
```

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

In [ ]:
```
def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_Images/HP_Part_02_images/'
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution, \
                    bbox_inches='tight')
    else:
        pass
```

Load Data

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

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

```
<ipython-input-54-b2855eddaef9>:4: DtypeWarning: Columns (10,12,68,83,84,85,86,165,183) have mixed types. Specify dt
ype option on import or set low_memory=False.
  hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

Out[ ]:

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

Filter confirmed (if selected)

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

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

```
Using all 4147 record in the Hillforts Atlas.
```

Download Function

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

Reload Name and Number

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

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

Out[ ]:

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

Reload Location

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

Out[ ]:

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

Review Data Part 2

Management Data

The Management data is split into two subgroups:

- Condition
- Land Use

In [ ]:
```
management_features = [
 'Management_Condition_Extant',
 'Management_Condition_Cropmark',
 'Management_Condition_Destroyed',
 'Management_Condition_Comments',
 'Management_Land_Use_Woodland',
 'Management_Land_Use_Plantation',
 'Management_Land_Use_Parkland',
 'Management_Land_Use_Pasture',
 'Management_Land_Use_Arable',
 'Management_Land_Use_Scrub',
 'Management_Land_Use_Outcrop',
 'Management_Land_Use_Moorland',
 'Management_Land_Use_Heath',
 'Management_Land_Use_Urban',
 'Management_Land_Use_Coastal',
 'Management_Land_Use_Other',
 'Management_Land_Use_Comments']

management_data = hillforts_data[management_features]
management_data.head()
```

Out[ ]:

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed	Management_Condition_Comments
0	Yes	No	No	Main ditch gone on N and W sides. Visitor eros...
1	Yes	No	No	Natural and animal erosion with sheep scrapes....
2	Yes	No	No	Although the circuit continuous in part, the s...
3	Yes	No	No	Slumping has occurred around the SE corner. Ra...
4	Yes	No	No	Visitor numbers great, but erosion repair has ...

'Management_Condition_Comments' and 'Management_Land_Use_Comments' contain null values.
See: Management Text Data - Resolve Null Values.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Management_Condition_Extant     4147 non-null   object
 1   Management_Condition_Cropmark   4147 non-null   object
 2   Management_Condition_Destroyed  4147 non-null   object
 3   Management_Condition_Comments   1849 non-null   object
 4   Management_Land_Use_Woodland    4147 non-null   object
 5   Management_Land_Use_Plantation  4147 non-null   object
 6   Management_Land_Use_Parkland    4147 non-null   object
 7   Management_Land_Use_Pasture     4147 non-null   object
 8   Management_Land_Use_Arable      4147 non-null   object
 9   Management_Land_Use_Scrub       4147 non-null   object
 10  Management_Land_Use_Outcrop     4147 non-null   object
 11  Management_Land_Use_Moorland    4147 non-null   object
 12  Management_Land_Use_Heath       4147 non-null   object
 13  Management_Land_Use_Urban       4147 non-null   object
 14  Management_Land_Use_Coastal     4147 non-null   object
 15  Management_Land_Use_Other       4147 non-null   object
 16  Management_Land_Use_Comments    1823 non-null   object
dtypes: object(17)
memory usage: 550.9+ KB
```

Management Numeric Data

The Management data package contains no numeric data features.

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

Management Text Data

There are two free text Management features.

In [ ]:
```
management_text_features = [
 'Management_Condition_Comments',
 'Management_Land_Use_Comments']

management_text_data = hillforts_data[management_text_features].copy()
management_text_data.head()
```

Out[ ]:

	Management_Condition_Comments	Management_Land_Use_Comments
0	Main ditch gone on N and W sides. Visitor eros...	Mixed woodland since 19th century with interna...
1	Natural and animal erosion with sheep scrapes....	Potatoes once grown on the site, but vegetatio...
2	Although the circuit continuous in part, the s...	A wooded private site with access problems.
3	Slumping has occurred around the SE corner. Ra...	The interior is arable and pasture at present....
4	Visitor numbers great, but erosion repair has ...	Upland pasture, moorland and scrub. Medieval r...

Management Text Data - Resolve Null Values

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

In [ ]:
```
test_cat_list_for_NA(management_text_data, management_text_features)
```

```
Management_Condition_Comments 0
Management_Land_Use_Comments 0
```

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

In [ ]:
```
management_text_data = update_cat_list_for_NA(management_text_data, \
                                              management_text_features)
management_text_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 2 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   Management_Condition_Comments  4147 non-null   object
 1   Management_Land_Use_Comments   4147 non-null   object
dtypes: object(2)
memory usage: 64.9+ KB
```

Management Encodeable Data

Management features that have the potential to be encoded.

In [ ]:
```
management_encodeable_features = [
 'Management_Condition_Extant',
 'Management_Condition_Cropmark',
 'Management_Condition_Destroyed',
 'Management_Land_Use_Woodland',
 'Management_Land_Use_Plantation',
 'Management_Land_Use_Parkland',
 'Management_Land_Use_Pasture',
 'Management_Land_Use_Arable',
 'Management_Land_Use_Scrub',
 'Management_Land_Use_Outcrop',
 'Management_Land_Use_Moorland',
 'Management_Land_Use_Heath',
 'Management_Land_Use_Urban',
 'Management_Land_Use_Coastal',
 'Management_Land_Use_Other']

management_encodeable_data = hillforts_data[management_encodeable_features].copy()
management_encodeable_data.head()
```

Out[ ]:

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed	Management_Land_Use_Woodland
0	Yes	No	No	Yes
1	Yes	No	No	No
2	Yes	No	No	Yes
3	Yes	No	No	Yes
4	Yes	No	No	No

Condition Data

The Management features are further subdivided into 'Condition' and 'Land-Use'. These will be reviewed independently.

In [ ]:
```
condition_features = [
 'Management_Condition_Extant',
 'Management_Condition_Cropmark',
 'Management_Condition_Destroyed']

management_conditon_data = management_encodeable_data[condition_features]
management_conditon_data.head()
```

Out[ ]:

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed
0	Yes	No	No
1	Yes	No	No
2	Yes	No	No
3	Yes	No	No
4	Yes	No	No

Condition Data Plotted

Most recorded hillforts are extant. The total 'yes' count is greater than the total number of records, meaning there are hillforts which have a 'yes' in one or more of these features. It is possible for a hillfort to be extant, a cropmark and destroyed.

In [ ]:
```
count_yes(management_conditon_data)
```

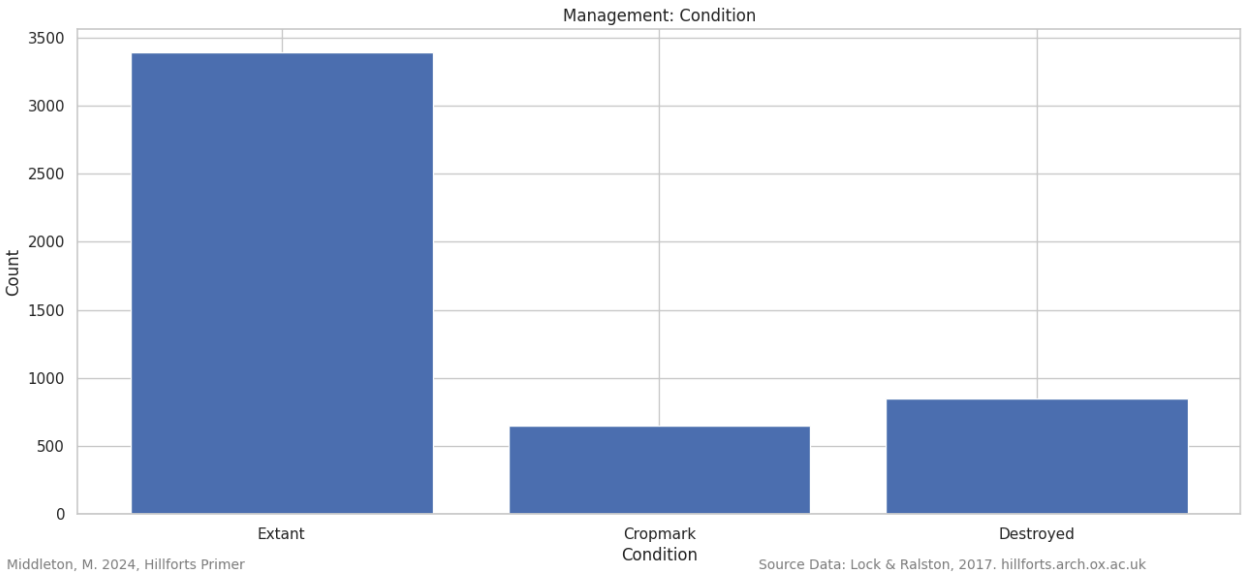
```
Management_Condition_Extant: 3391
Management_Condition_Cropmark: 648
Management_Condition_Destroyed: 852
Total yes count: 4891
```

In [ ]:
```
management_conditon_data.loc\
 [(management_conditon_data['Management_Condition_Cropmark'] == 'Yes') &\
  (management_conditon_data['Management_Condition_Destroyed'] == 'Yes')].head()
```

Out[ ]:

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed
26	Yes	Yes	Yes
158	No	Yes	Yes
340	No	Yes	Yes
375	Yes	Yes	Yes
410	Yes	Yes	Yes

In [ ]:
```
plot_bar_chart(management_conditon_data, 2, 'Condition', 'Count', 'Management: Condition')
```



Condition Data Mapped

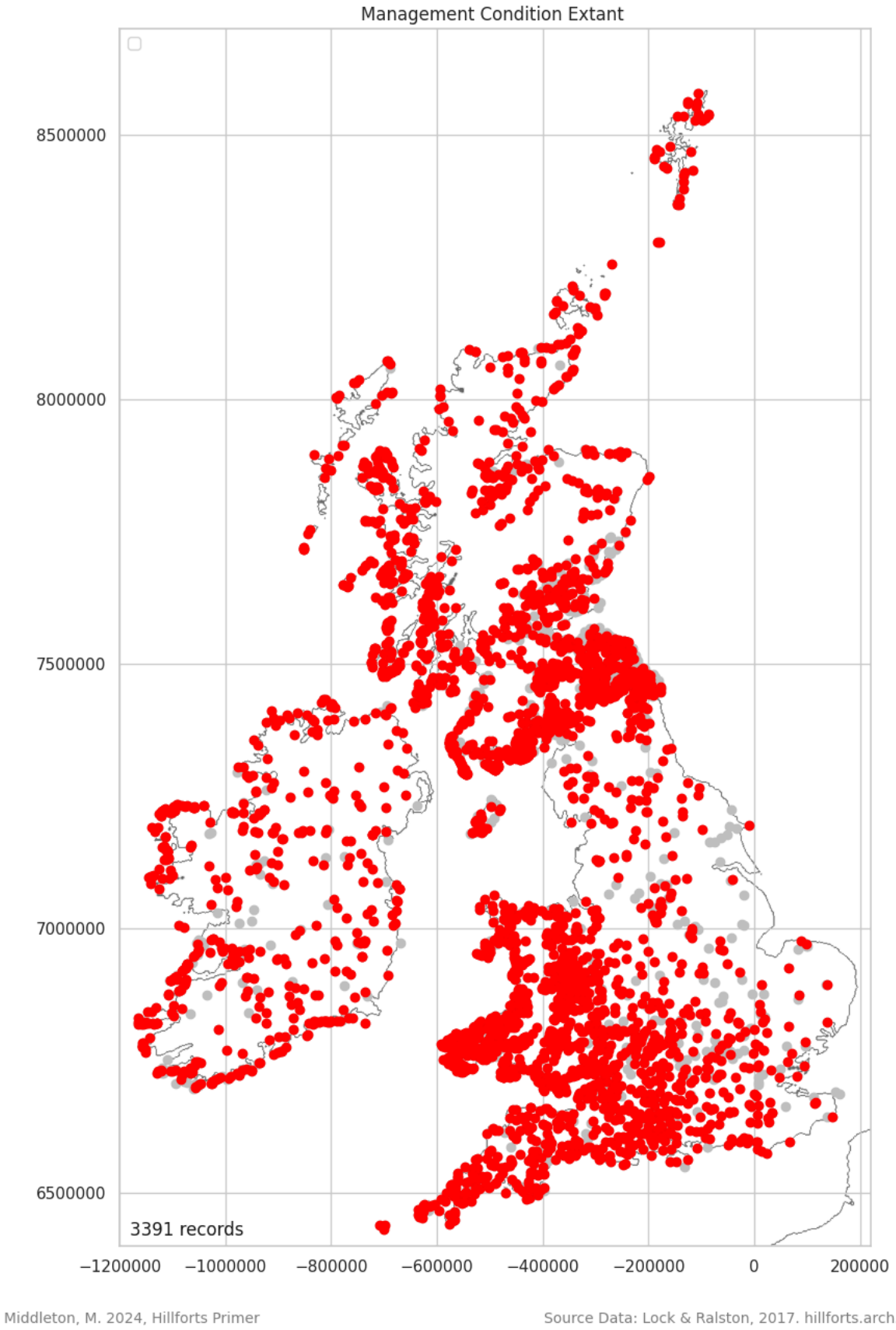
The Condition data is recombined with the 'Location' data so it can be mapped.

In [ ]:
```
location_condition_data = pd.merge(location_numeric_data_short, \
                                   management_conditon_data, left_index=True, \
                                   right_index=True)
```

Extant Mapped

There are 3391 hillforts (81.77%) identified as extant. Of these, 2686 are fully extant.

In [ ]:
```
man_extent = plot_over_grey(location_condition_data, \
                            'Management_Condition_Extant', 'Yes')
```



81.77%

In [ ]:
```
extant_only = management_conditon_data\
 [(management_conditon_data['Management_Condition_Extant']=='Yes') & \
  (management_conditon_data['Management_Condition_Cropmark']=='No') & \
   (management_conditon_data['Management_Condition_Destroyed']=='No')]
print(f'Extant only: {len(extant_only)}')
```

```
Extant only: 2686
```

Cropmark Mapped

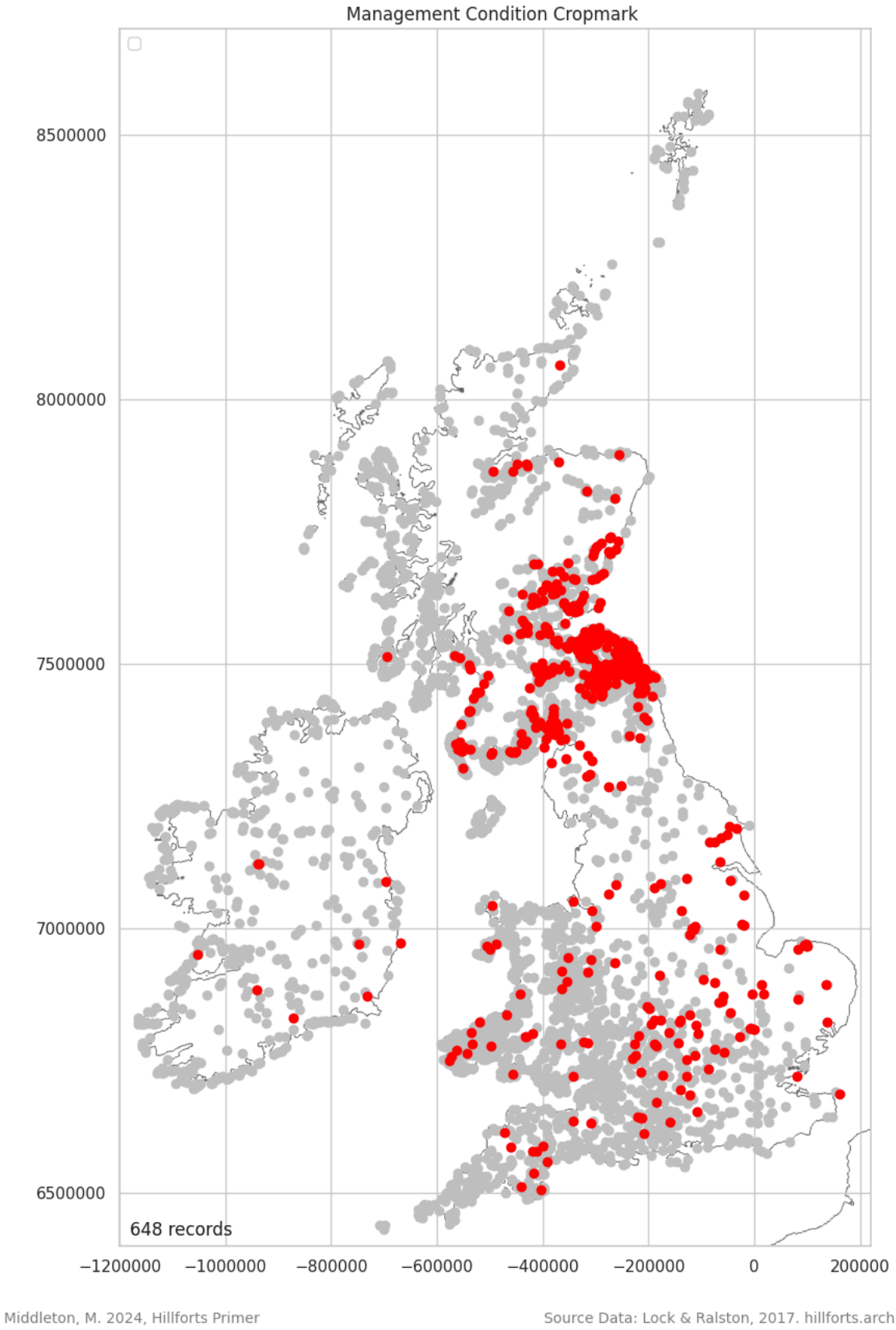
There are 648 hillforts (15.63%), recorded as a cropmark. Of these 477 are known exclusively from the cropmark record.

There are two forms of bias in this data. The first is a recording bias. There are 530 cropmark forts in the northern data compared to 118 in the South. There are only eight in Ireland. Even in the north, the map shows there is a concentration of records across the south-eastern lowlands of Scotland. This corresponds to the second area of bias, cropmark visibility bias toward areas of arable land. 480 of the 530 cropmark forts in the north (90.5%) are in areas of arable land. See: Arable Mapped.

There are 1598 hillforts in the data package isolating the density cluster over the Southern Uplands (See Part 1: Northeast Density). 458 of the hillforts in this area (28.6%) are only known because they have been recorded as a cropmark. This high number of cropmark forts, compared to the very low numbers in other areas, means the analysis of clusters is not comparing like-for-like. The low density of cropmark forts outside the Northeast is depressing the intensity of the clusters in these areas.

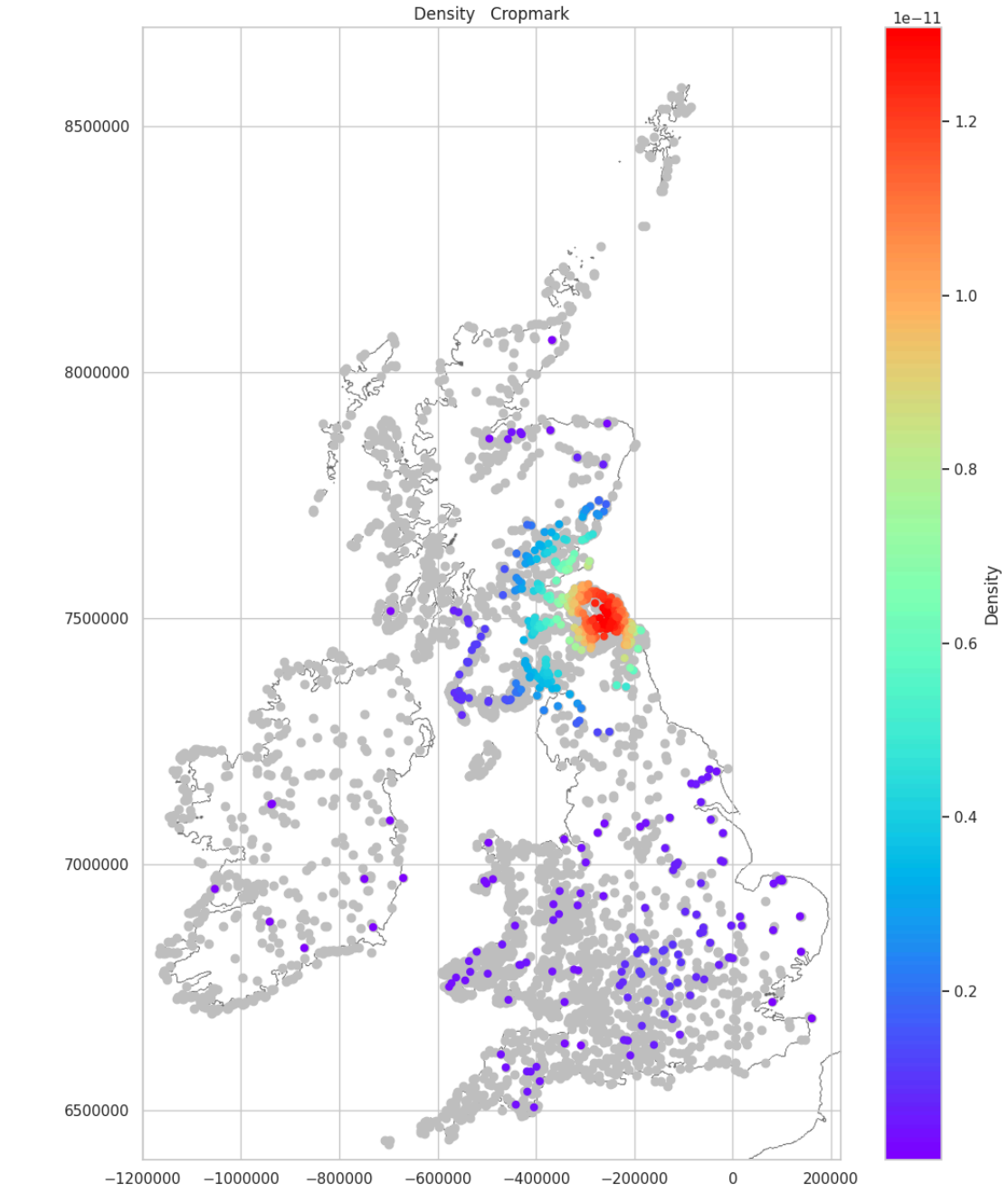
It is not possible to tell from the data if the variation in distribution of cropmark hillforts relate to a difference in recording intensity between Scotland and the other nations or, if a more inclusive definition of hillfort was adopted in Scotland during the data gathering phase of the Hillforts Atlas.

In [ ]:
```
man_cropmark = plot_over_grey(location_condition_data, \
                              'Management_Condition_Cropmark', 'Yes')
```



15.63%

In [ ]:
```
plot_density_over_grey(man_cropmark, 'Cropmark')
```



Middleton, M. 2024, Hillforts Primer

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

In [ ]:
```
cropmark_only = management_conditon_data\
 [(management_conditon_data['Management_Condition_Cropmark']=='Yes') & \
  (management_conditon_data['Management_Condition_Extant']=='No') & \
   (management_conditon_data['Management_Condition_Destroyed']=='No')]
print(f'Cropmark only: {len(cropmark_only)}')
print(f'{round((len(cropmark_only)/4147)*100,2)}% of the total number of hillforts in the Atlas are known exclusivel
```

```
Cropmark only: 477
11.5% of the total number of hillforts in the Atlas are known exclusively from cropmarks.
```

In [ ]:
```
split_location_y = 7070000
split_location_x = -500000
location_management_encodeable_data = pd.merge(location_numeric_data_short, \
                                               management_encodeable_data, \
                                               left_index=True, right_index=True)
north_location_condition_data = \
location_management_encodeable_data[location_management_encodeable_data['Location_Y'] \
                                    >= split_location_y].copy().reset_index(drop=True)
north_east_location_condition_data = \
north_location_condition_data[north_location_condition_data['Location_X'] >= \
                              split_location_x].copy().reset_index(drop=True)
cropmark_north_east_location_condition_data = \
north_east_location_condition_data[north_east_location_condition_data\
 ['Management_Condition_Cropmark']=='Yes']
print(f'There are {len(cropmark_north_east_location_condition_data)} cropmark forts in the Northeast.')
```

```
There are 503 cropmark forts in the Northeast.
```

In [ ]:
```
cropmark_north_east_location_condition_data_arable = \
cropmark_north_east_location_condition_data\
 [cropmark_north_east_location_condition_data['Management_Land_Use_Arable']=='Yes']
print(f'{len(cropmark_north_east_location_condition_data_arable)} of the forts in the Northeast are located in areas
print(f'{round((len(cropmark_north_east_location_condition_data_arable)/len(cropmark_north_east_location_condition_d
```

```
458 of the forts in the Northeast are located in areas of arable land use.
91.05%
```

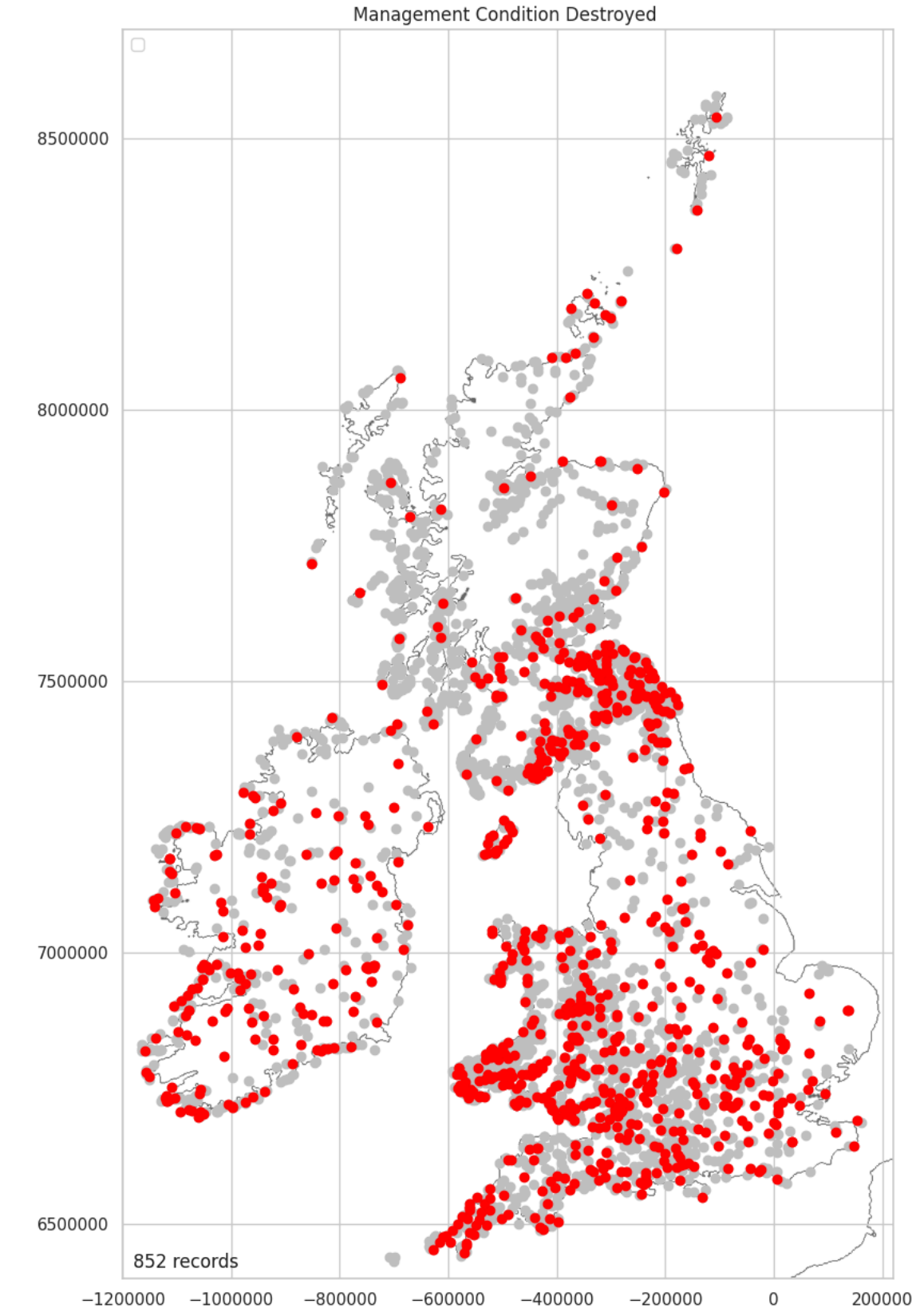
In [ ]:
```
north_east_cropmark_only = \
cropmark_north_east_location_condition_data_arable\
 [(cropmark_north_east_location_condition_data_arable\
  ['Management_Condition_Cropmark'] =='Yes') & \
   (cropmark_north_east_location_condition_data_arable\
    ['Management_Condition_Extant']=='No') & \
     (cropmark_north_east_location_condition_data_arable\
      ['Management_Condition_Destroyed']=='No')]
print(f'Of the 458 cropmark forts in the Northeast, {len(north_east_cropmark_only)} survive only as a cropmark.')
print(f'{round((len(north_east_cropmark_only)/len(cropmark_north_east_location_condition_data_arable))*100,2)}%')
```

```
Of the 458 cropmark forts in the Northeast, 377 survive only as a cropmark.
82.31%
```

Destroyed Mapped

There are 852 hillforts (20.54%) identified as destroyed. Of these 212 (5.1%) have been entirely erased.

In [ ]:
```
man_destroyed = \
plot_over_grey(location_condition_data, 'Management_Condition_Destroyed', 'Yes')
```



20.54%

In [ ]:
```
destroyed_only = \
management_conditon_data[(management_conditon_data\
 ['Management_Condition_Destroyed']=='Yes') & \
  (management_conditon_data['Management_Condition_Extant']=='No') & \
   (management_conditon_data['Management_Condition_Cropmark']=='No')]
print(f'Destroyed only: {len(destroyed_only)}')
```

```
Destroyed only: 212
```

Condition Not Recorded

There are 24 records that are not recorded as being extant, a cropmark or destroyed.

In [ ]:
```
all_no = \
management_conditon_data[(management_conditon_data\
 ['Management_Condition_Destroyed']=='No') & \
(management_conditon_data['Management_Condition_Extant']=='No') & \
(management_conditon_data['Management_Condition_Cropmark']=='No')]
print(f'Condition not recorded: {len(all_no)}')
```

```
Condition not recorded: 24
```

Five records are shown. To see the full list, update the 5, in the square brackets below, to 24 and rerun the document as described in User Settings.

In [ ]:
```
all_no_full = pd.merge(name_and_number, all_no, left_index=True, right_index=True)
all_no_full[:5]
```

Out[ ]:

	Main_Atlas_Number	Main_Display_Name	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_I
355	365	Harborough Hill, Churchill, Worcestershire	No	No	
1742	1830	Prae Wood, Hertfordshire (St. Albans; Verulami...	No	No	
1743	1831	Wheathampstead, Hertfordshire	No	No	
1744	1832	Braughing, Hertfordshire (Gatesbury Camp)	No	No	
1747	1835	Béal Deirg Mór, Mayo	No	No	

Land Use Data

The Land Use data consists of 11 land use types and 'Other'. There is a bias in this data caused by there being a mix of habitat and land use classifications used across the Atlas.

In [ ]:
```
land_use_features = [
 'Management_Land_Use_Woodland',
 'Management_Land_Use_Plantation',
 'Management_Land_Use_Parkland',
 'Management_Land_Use_Pasture',
 'Management_Land_Use_Arable',
 'Management_Land_Use_Scrub',
 'Management_Land_Use_Outcrop',
 'Management_Land_Use_Moorland',
 'Management_Land_Use_Heath',
 'Management_Land_Use_Urban',
 'Management_Land_Use_Coastal',
 'Management_Land_Use_Other']

land_use_data = management_encodeable_data[land_use_features].copy()
land_use_data.head()
```

Out[ ]:

	Management_Land_Use_Woodland	Management_Land_Use_Plantation	Management_Land_Use_Parkland	Management_Land_Use_Pasture	M
0	Yes	No	No	No	
1	No	No	No	Yes	
2	Yes	No	No	No	
3	Yes	No	No	Yes	
4	No	No	No	Yes	

Land Use Data Plotted

The 'yes' count shows that a hillfort can have multiple land use values. The feature, 'Management_Land_Use_Outcrop' contains only negative values and therefore has no predictive worth. It will be dropped. See: Drop Land Management Features.

In [ ]:
```
count_yes(land_use_data)
```

```
Management_Land_Use_Woodland: 880
Management_Land_Use_Plantation: 181
Management_Land_Use_Parkland: 101
Management_Land_Use_Pasture: 2314
Management_Land_Use_Arable: 712
Management_Land_Use_Scrub: 961
Management_Land_Use_Outcrop: 0
Management_Land_Use_Moorland: 875
Management_Land_Use_Heath: 27
Management_Land_Use_Urban: 255
Management_Land_Use_Coastal: 419
Management_Land_Use_Other: 386
Total yes count: 7111
```

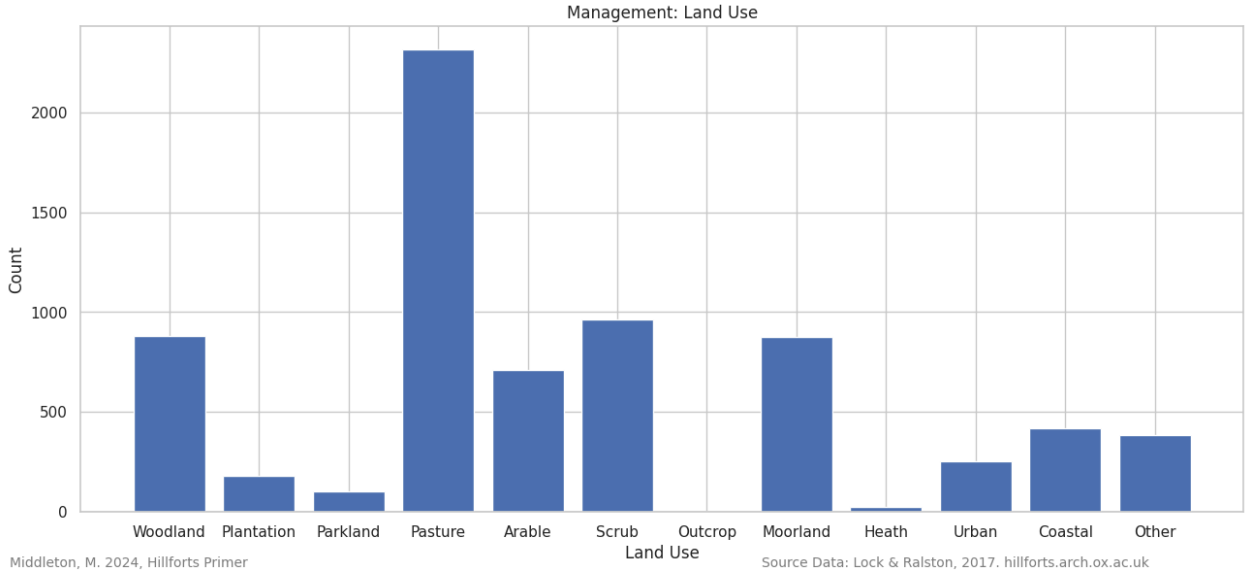
In [ ]:
```
land_use_data.loc[(land_use_data['Management_Land_Use_Scrub'] == 'Yes') & \
 (land_use_data['Management_Land_Use_Moorland'] == 'Yes')].head()
```

Out[ ]:

	Management_Land_Use_Woodland	Management_Land_Use_Plantation	Management_Land_Use_Parkland	Management_Land_Use_Pasture
4	No	No	No	Yes
88	Yes	No	No	No
89	No	No	No	No
236	No	No	No	Yes
265	No	No	No	Yes

The most frequently recorded land use is pasture which contains 2314 hillforts. Woodland, arable, scrub and moorland are roughly similar in proportion with between 700 to 900 hillforts with the remainder having around 400 or less. Heath has only 27 records and this highlights a terminology bias due to an inconsistency of terminology used atlas wide. Scrub and heath are habitat definitions and may better be classified by the land use classification, moorland. Using combined figures, 50.8% of hillforts are in pasture, 40.66% in moorland (including scrub and heath) and 25.58% are under trees.

In [ ]:
```
plot_bar_chart(land_use_data, 3, 'Land Use', 'Count', 'Management: Land Use')
```



The percentages below are the number of hillforts recorded by each land use type compared to the total number of hillforts. It is important to remember that hillforts can contain multiple land use types and for this reason the combined percentage total is greater than 100%. This indicates that the average, across all forts is 1.7 land use types per fort. This in turn tells us that the land use, over most hillforts, is partitioned to some degree.

In [ ]:
```
total = 0
for feature in land_use_features:
    feture_parts = feature.split("_")
    yes_count = land_use_data[land_use_data[feature]=='Yes']
    pcnt = round((len(yes_count)/len(land_use_data))*100,2)
    total+=pcnt
    print(f'{feture_parts[-1]}: {pcnt}%')
print(f'Total: {total}%')
```

```
Woodland: 21.22%
Plantation: 4.36%
Parkland: 2.44%
Pasture: 55.8%
Arable: 17.17%
Scrub: 23.17%
Outcrop: 0.0%
Moorland: 21.1%
Heath: 0.65%
Urban: 6.15%
Coastal: 10.1%
Other: 9.31%
Total: 171.47%
```

Land Use Data Mapped

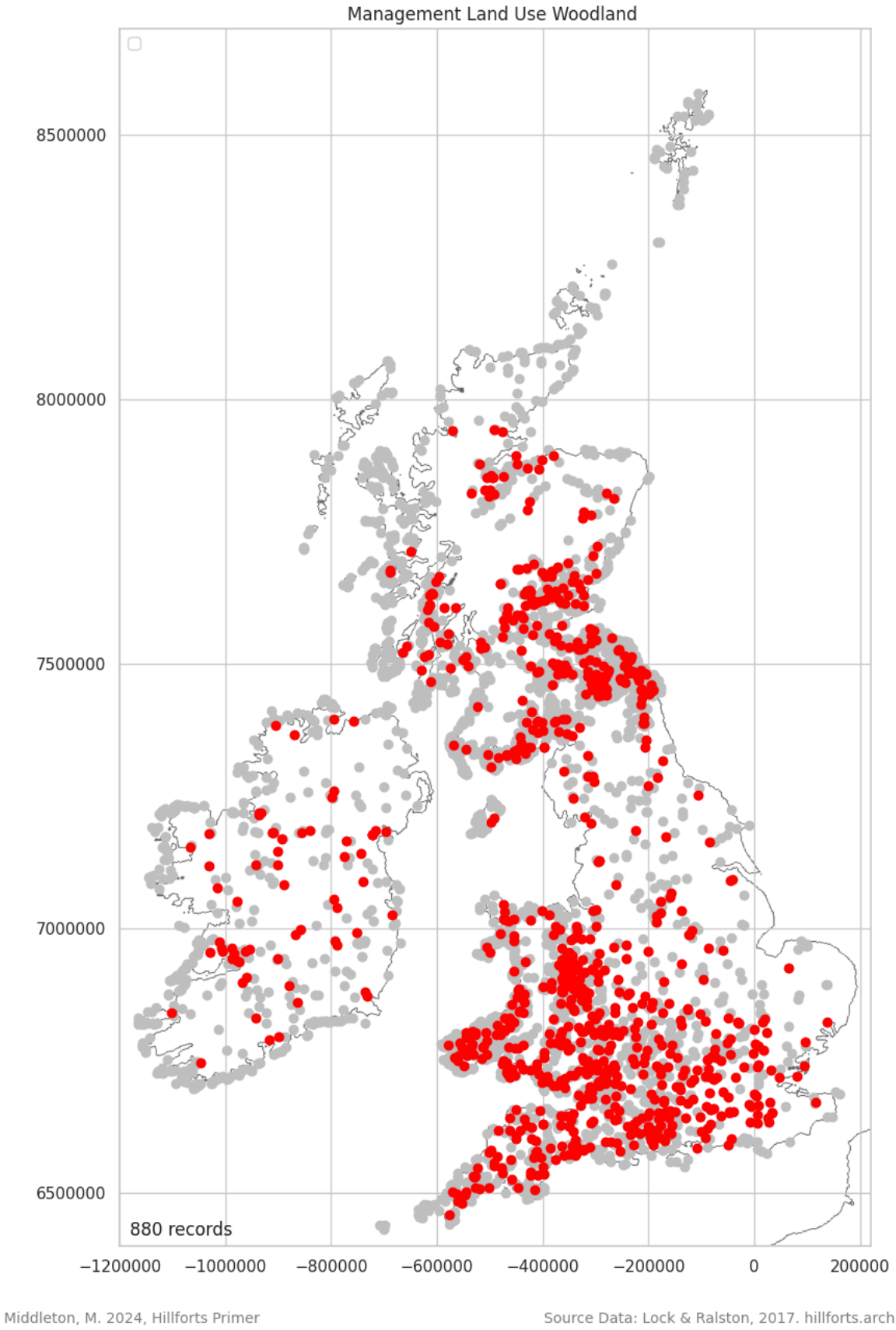
The 'Land Use' data is recombined with the 'Location' data so it can be mapped.

In [ ]:
```
location_land_use_data = pd.merge(location_numeric_data_short, \
                                  land_use_data, left_index=True, \
                                  right_index=True)
```

Woodland Mapped

There are 880 hillforts (22.22%) in woodland.

In [ ]:
```
lu_woodland = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Woodland', 'Yes')
```

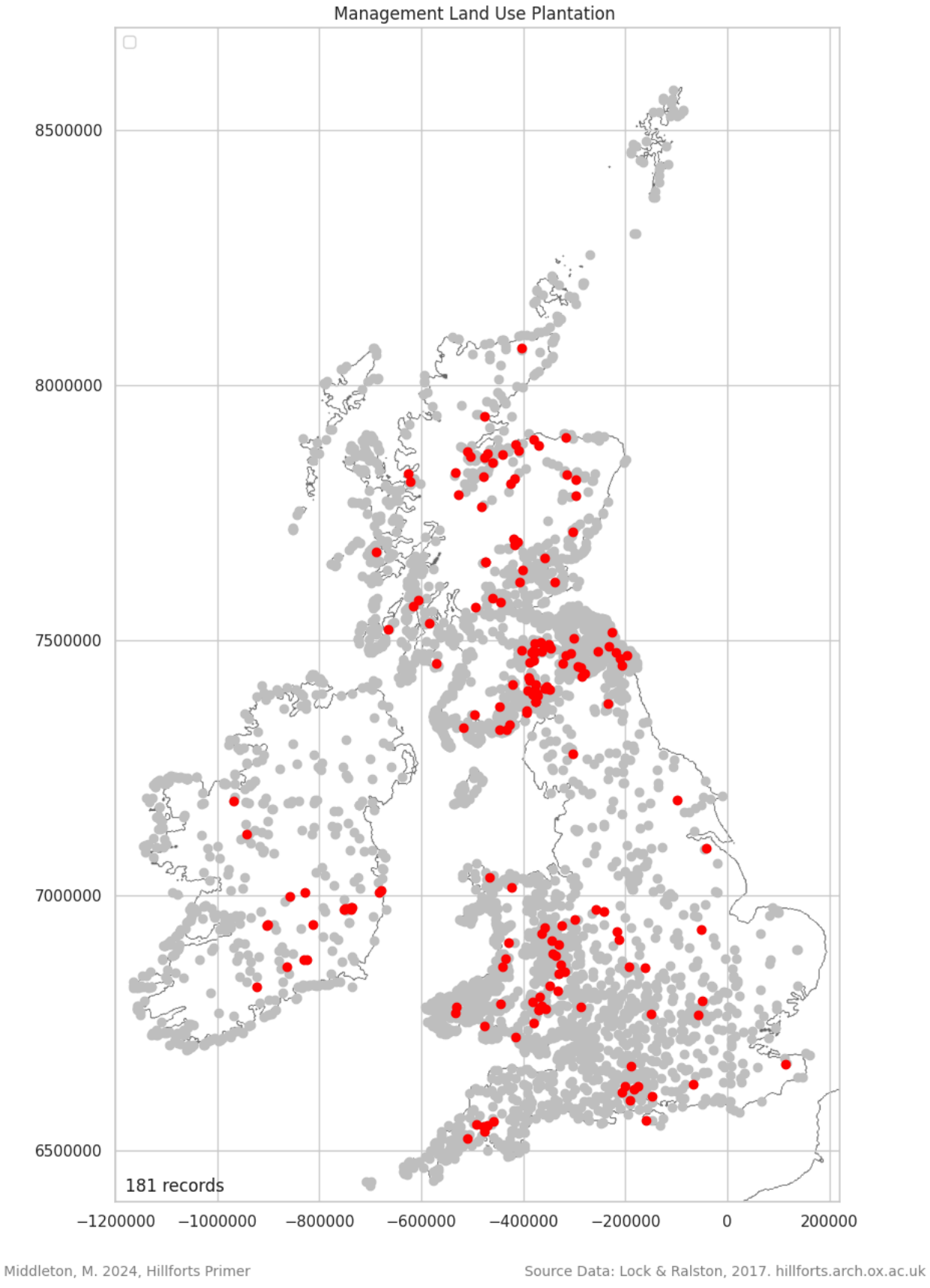



```
21.22%
```

Plantation Mapped

There are 181 hillforts (4.36%) under plantation.

In [ ]:
```
lu_plantation = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Plantation', 'Yes')
```

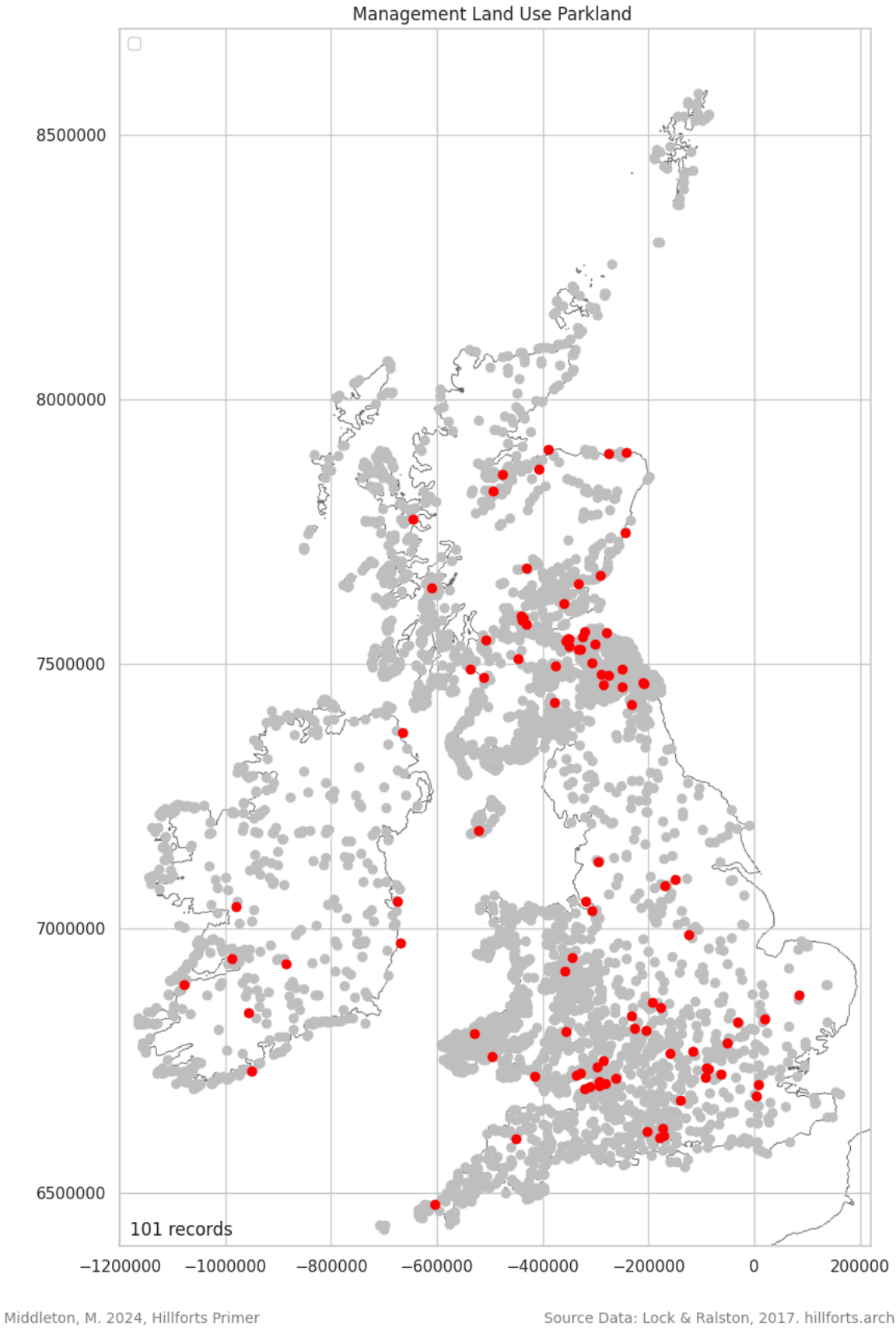


4.36%

Parkland Mapped

There are 101 hillforts (2.44%) in parkland.

In [ ]:
```
lu_parkland = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Parkland', 'Yes')
```

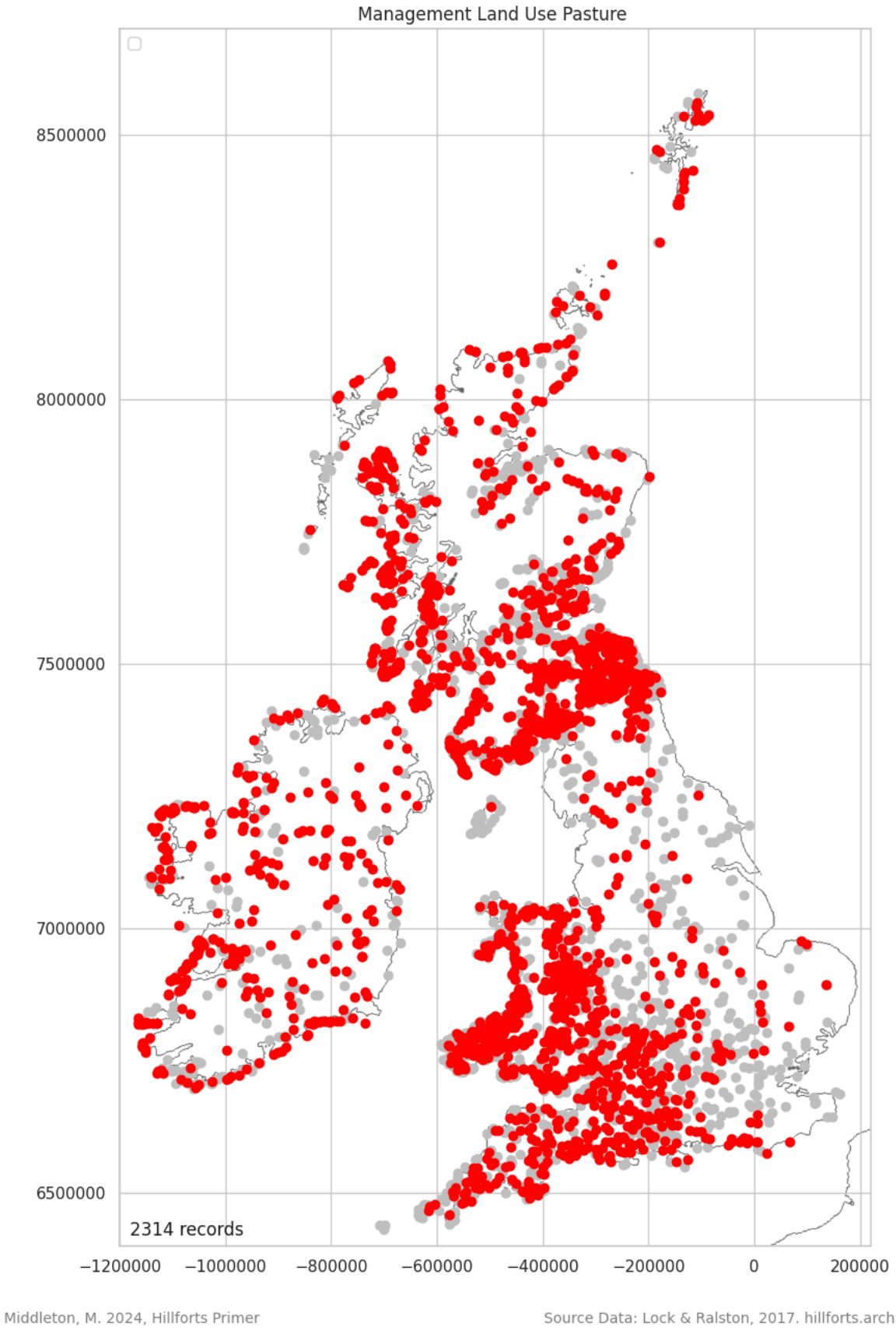


2.44%

Pasture Mapped

There are 2114 hillforts (55.8%) in pasture.

In [ ]:
```
lu_pasture = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Pasture', 'Yes')
```

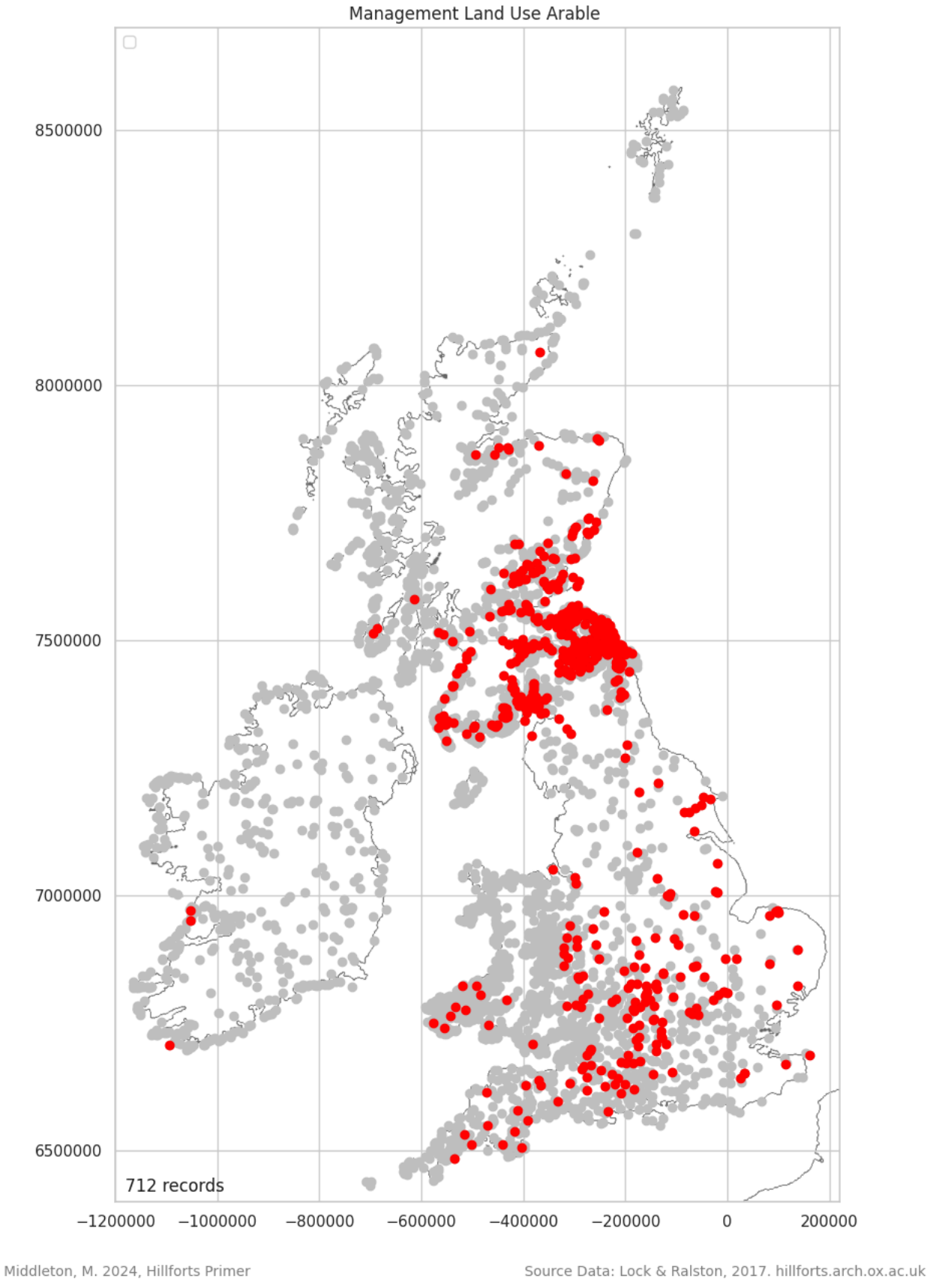


55.8%

Arable Mapped

There are 712 hillforts (17.17%) within arable farmland. 546 of these are all, or in part, cropmarks. See: Cropmark Mapped.

In [ ]:
```
lu_arable = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Arable', 'Yes')
```



17.17%

In [ ]:
```
arable_cm = \
len(management_data.loc[(management_data['Management_Land_Use_Arable']=='Yes') & \
 (management_data['Management_Condition_Cropmark']=='Yes')])
arable_cm
```

Out[ ]: 546

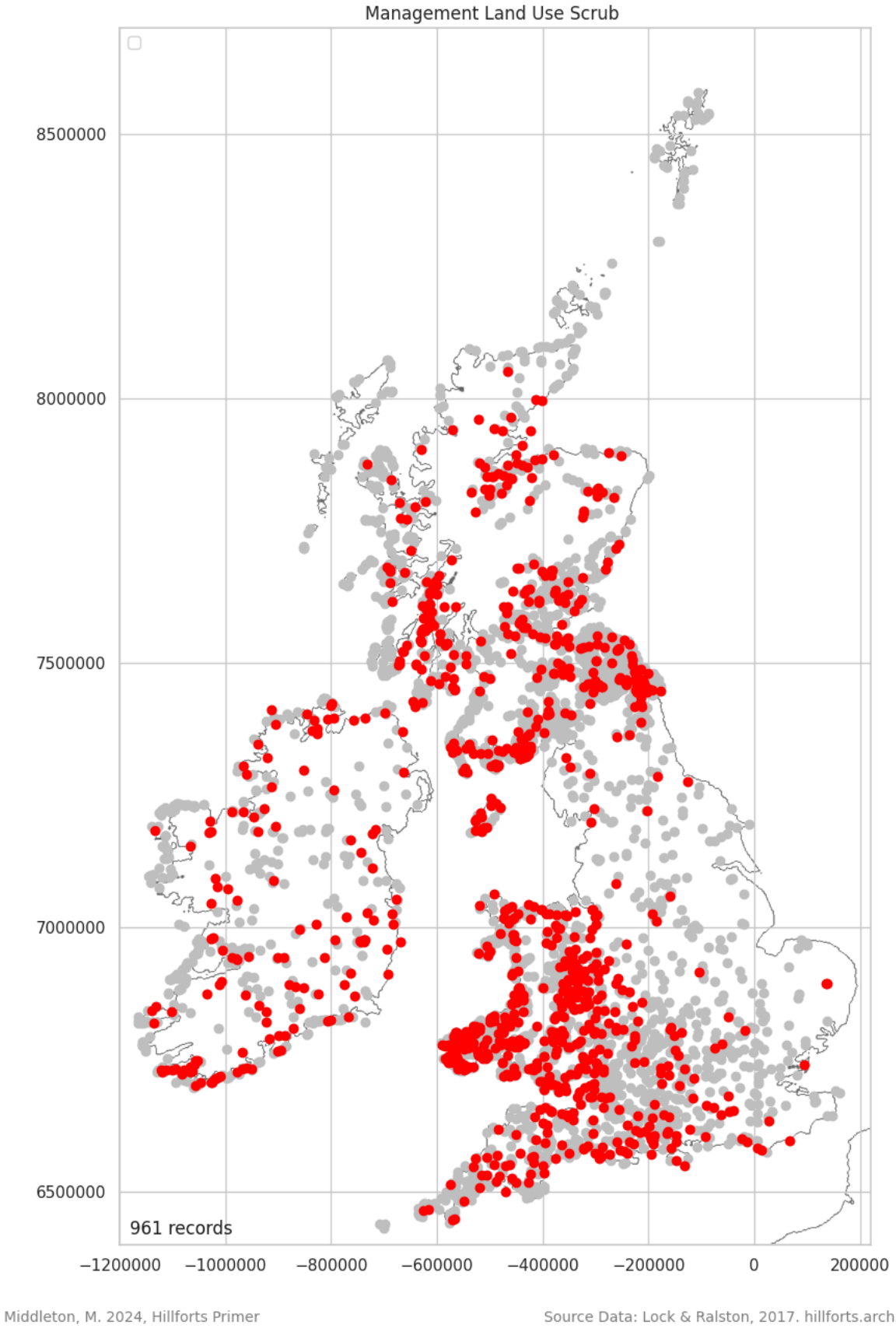
Scrub Mapped

961 hillforts (23.17%) are in scrub. This terminology is potentially confusing, as scrub is a habitat type, and is likely to be interchangeable with the terms moorland and heath.

See: Mountain heath and montane scrub & UK BAP Priority Habitats).

See: Moorland, Scrub and Heath Pooled and Mapped

In [ ]:
```
lu_scrub = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Scrub', 'Yes')
```

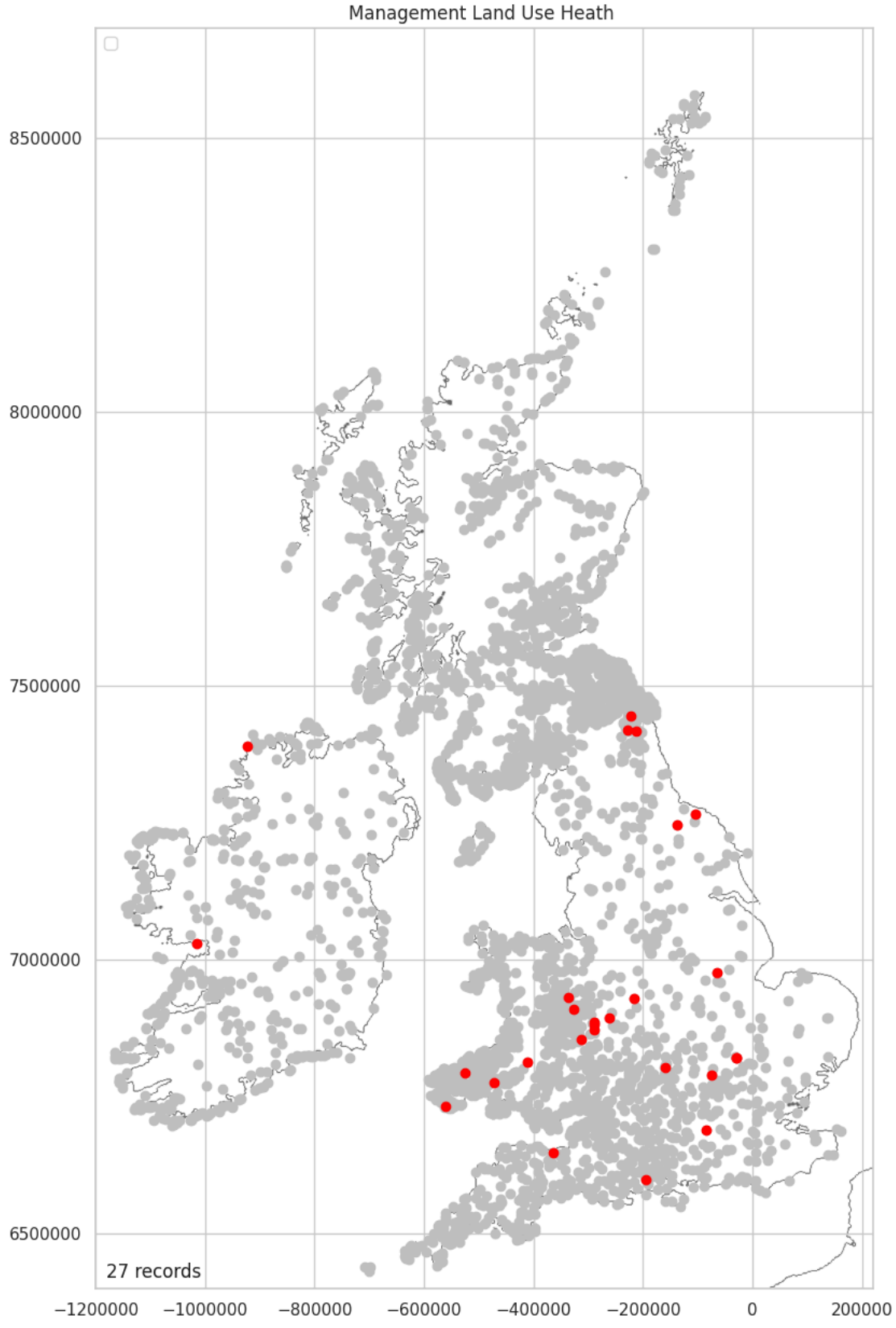



```
23.17%
```

Heath Mapped

Only 27 hillforts (0.65%) are in heath. This terminology is potentially confusing, as heath is a habitat type, and is likely to be interchangeable with the terms moorland and scrub.
See: Mountain heath and montane scrub & UK BAP Priority Habitats).
See: Moorland, Scrub and Heath Pooled and Mapped

In [ ]:
```
lu_heath = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Heath', 'Yes')
```



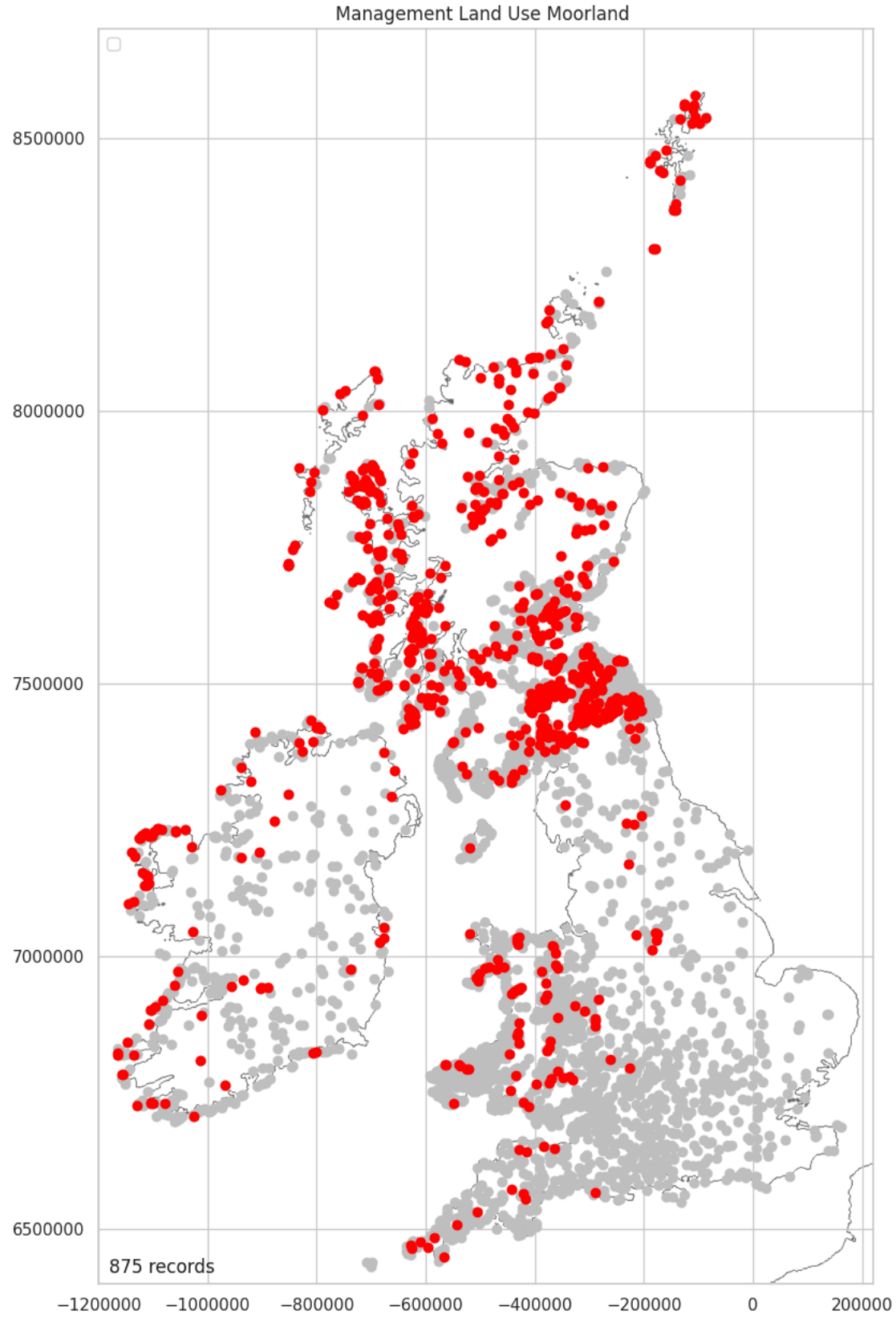

```
0.65%
```

Moorland Mapped

875 hillforts (21.17%) are in moorland. This terminology is potentially confusing, and is likely to be interchangeable with the terms heath and scrub.

See: Moorland, Scrub and Heath Pooled and Mapped

In [ ]:
```
lu_moorland = \
plot_over_grey(location_land_use_data, 'Management_Land_Use_Moorland', 'Yes')
```




```
21.1%
```

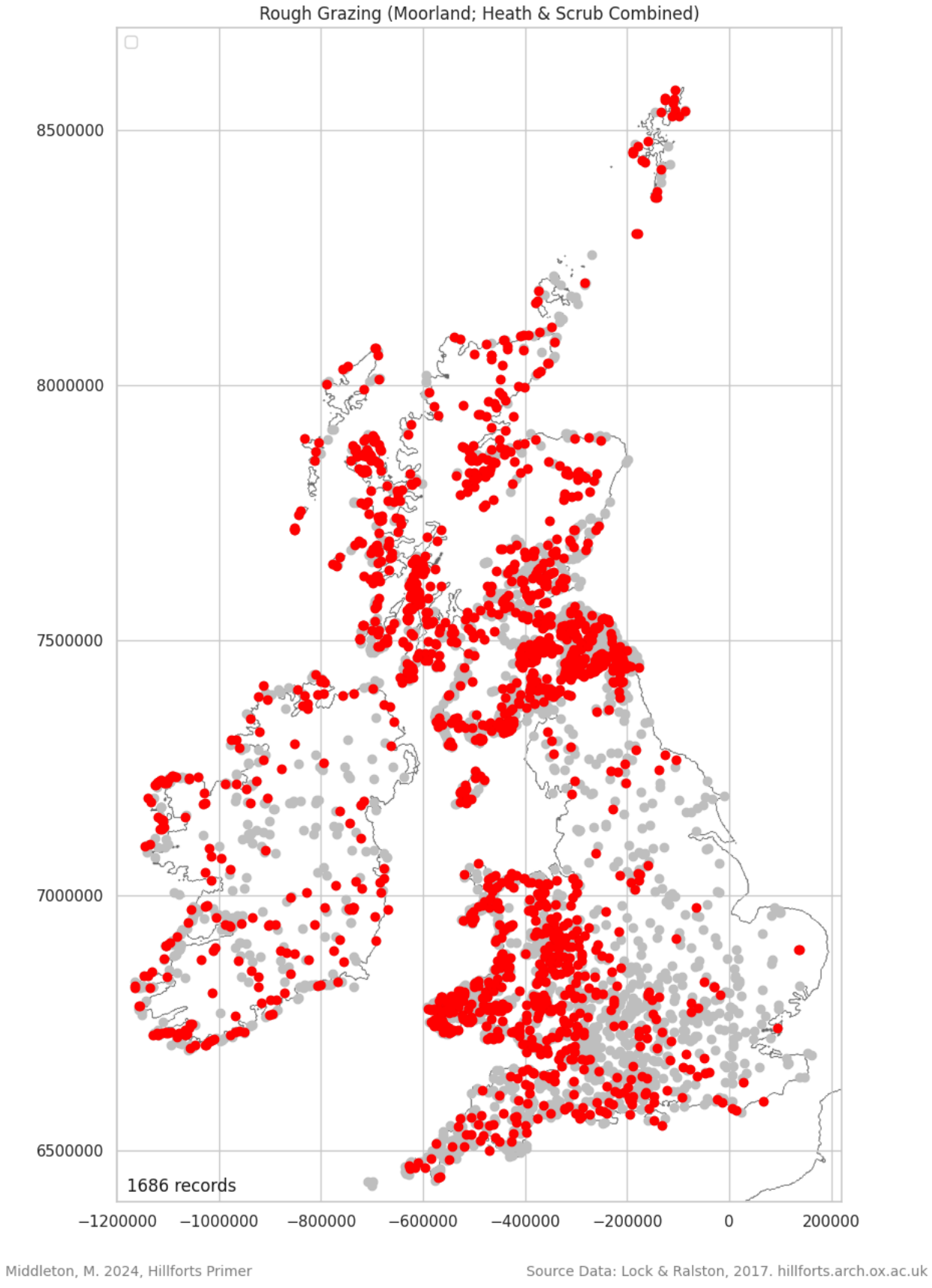
Rough Grazing: Moorland, Scrub and Heath Combined and Mapped

Heath, moorland and scrub may better be referred to as the land use type Rough Grazing. Combined, moorland, scrub and heath gives a total 1686 hillforts (40.66%).

See: Heath Mapped
See: Scrub Mapped
See: Moorland Mapped

In [ ]:
```
temp_moorland = location_land_use_data.copy()
temp_moorland.loc[temp_moorland['Management_Land_Use_Scrub'] == \
                  'Yes', 'Management_Land_Use_Moorland'] = 'Yes'
temp_moorland.loc[temp_moorland['Management_Land_Use_Heath'] == \
                  'Yes', 'Management_Land_Use_Moorland'] = 'Yes'
temp_moorland['Rough_Grazing'] = temp_moorland['Management_Land_Use_Moorland']
```

In [ ]:
```
lu_moorland_temp = plot_over_grey(temp_moorland, 'Rough_Grazing', \
                                  'Yes', '(Moorland, Heath & Scrub Combined)')
```

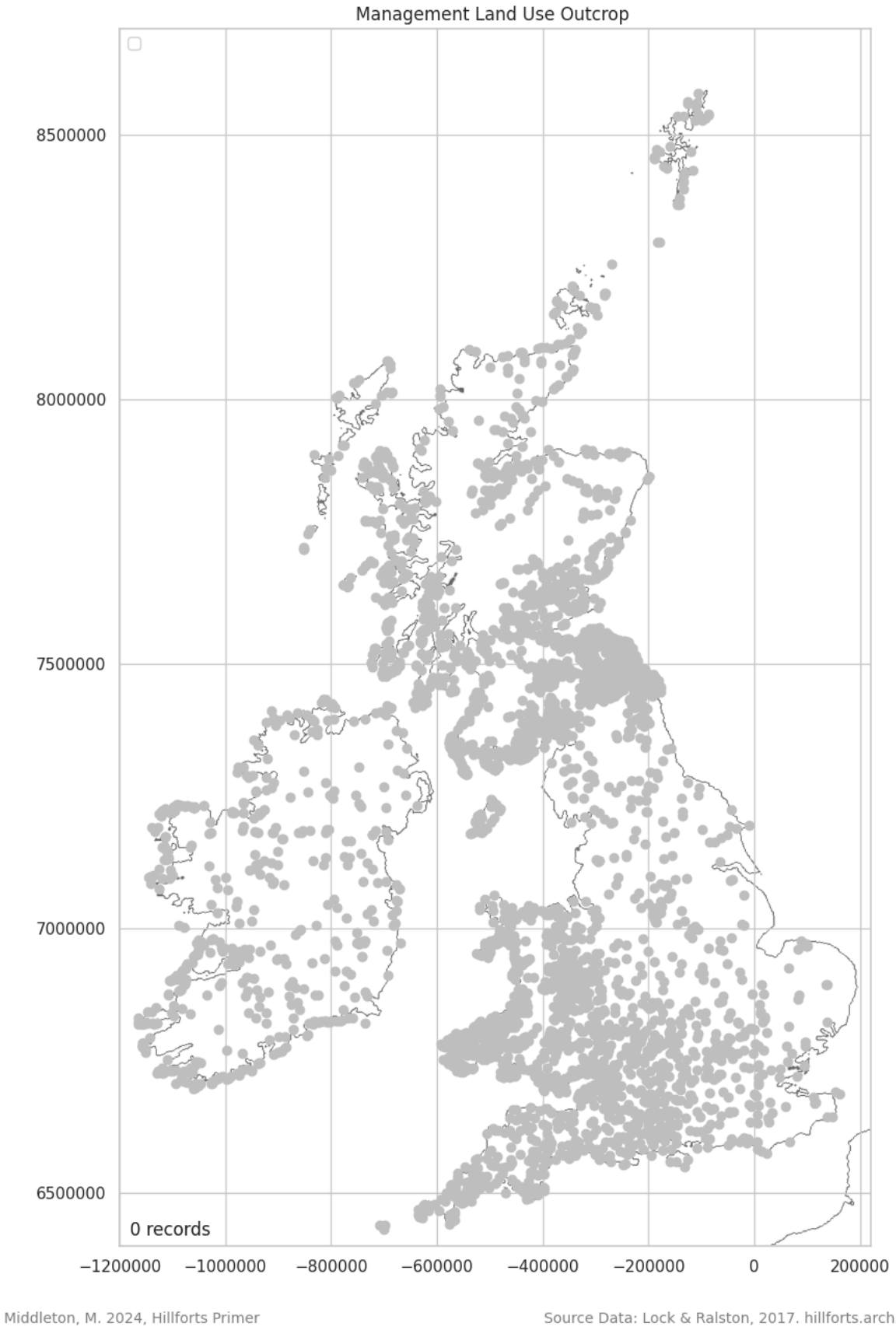


40.66%

Outcrop Mapped

There are no hillforts with the classification 'Outcrop'. As all records contain the same information it has no predictive value. See: Drop Land Management Features

In [ ]:
```
lu_outcrop = plot_over_grey(location_land_use_data, \
                            'Management_Land_Use_Outcrop', 'Yes')
```

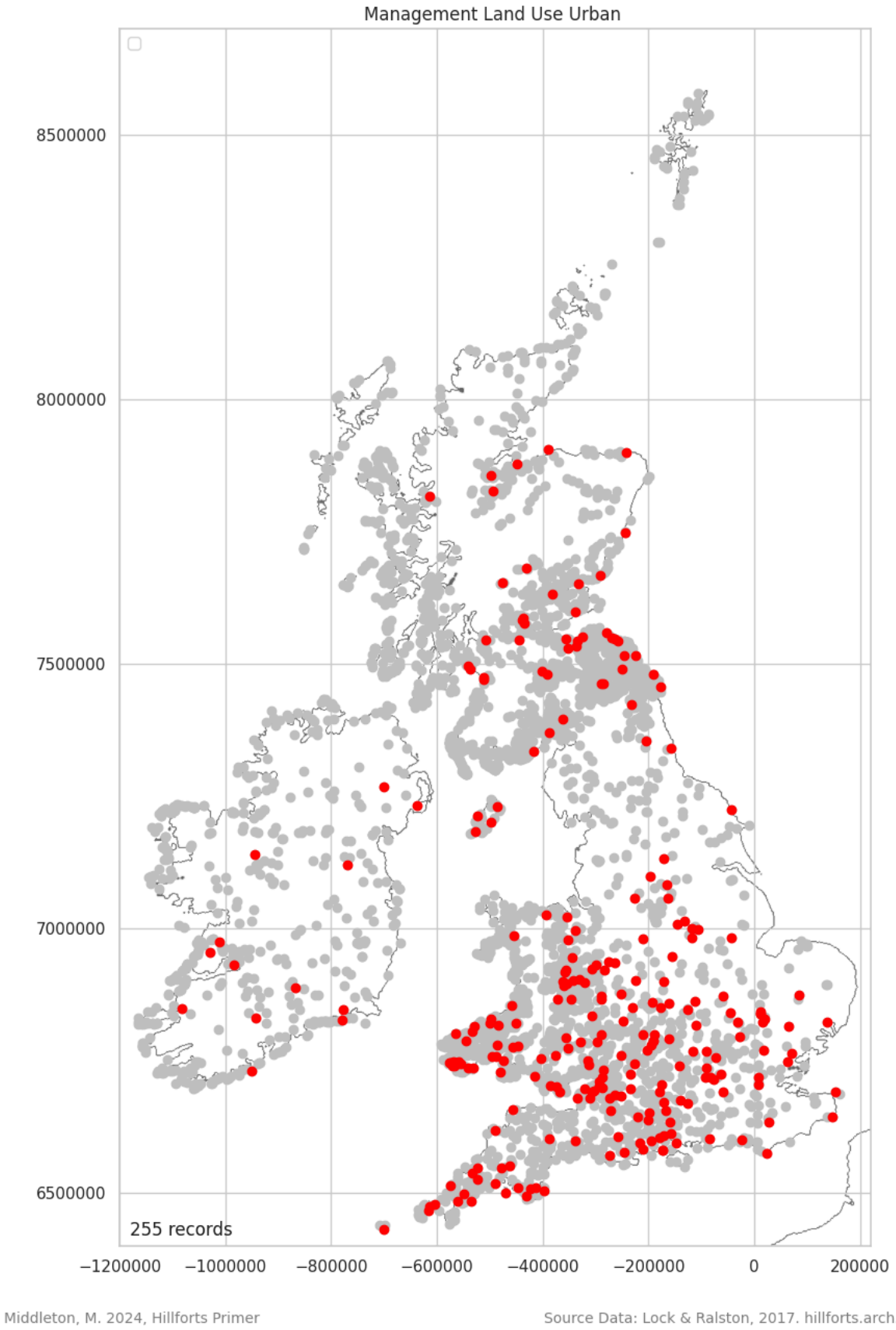



```
0.0%
```

Urban Mapped

255 hillforts (6.15%) are classified as having an urban land use.

In [ ]:
```
lu_urban = plot_over_grey(location_land_use_data, \
                          'Management_Land_Use_Urban', 'Yes')
```

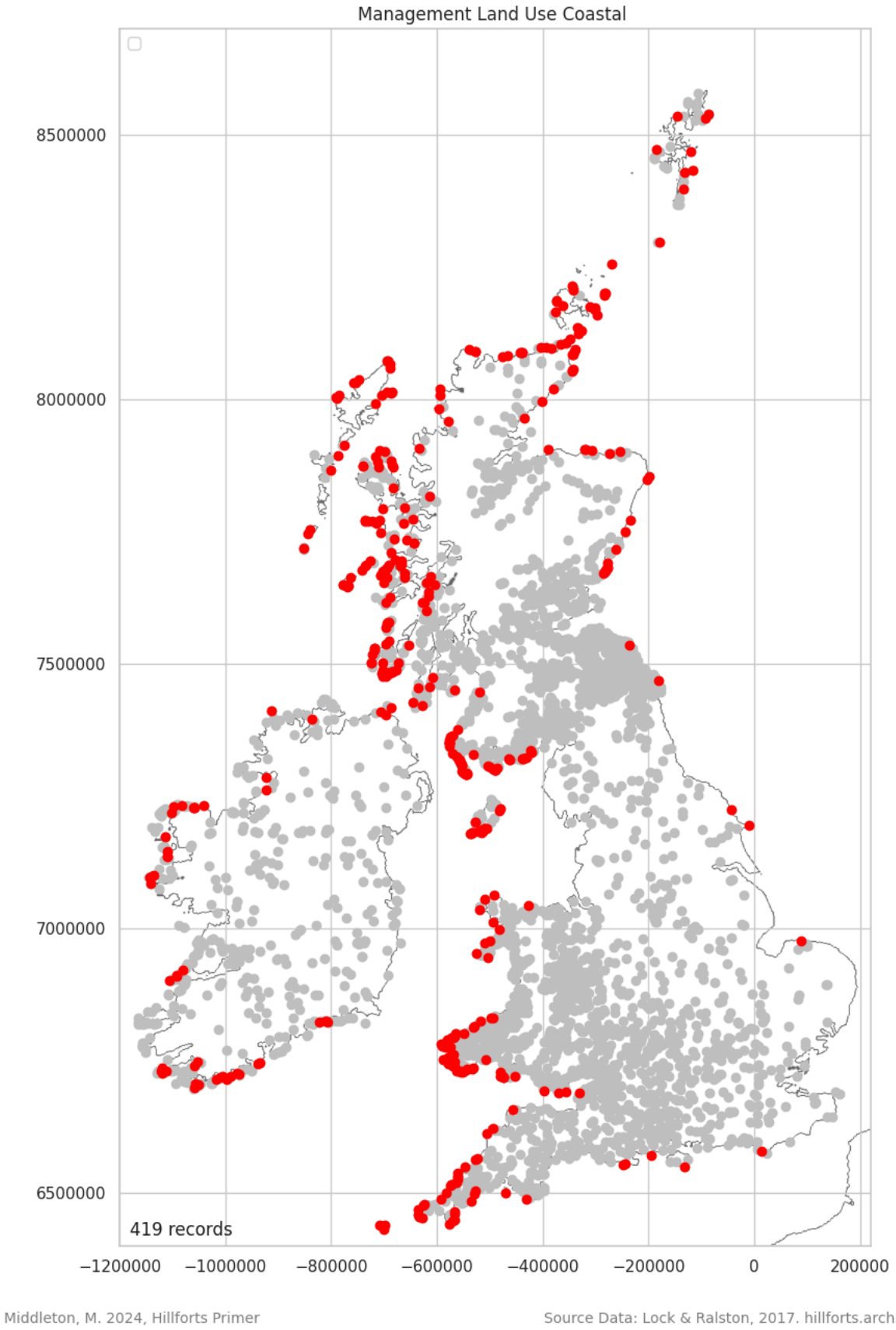


6.15%

Coastal Mapped (Land Use)

419 hillforts (10.1%) are classified as having a coastal land use. Coastal is either a topographic location or a habitat type and does not convey a land use.
See: Coastal Mapped (Topo)

In [ ]:
```
lu_coastal = plot_over_grey(location_land_use_data, \
                            'Management_Land_Use_Coastal', 'Yes')
```

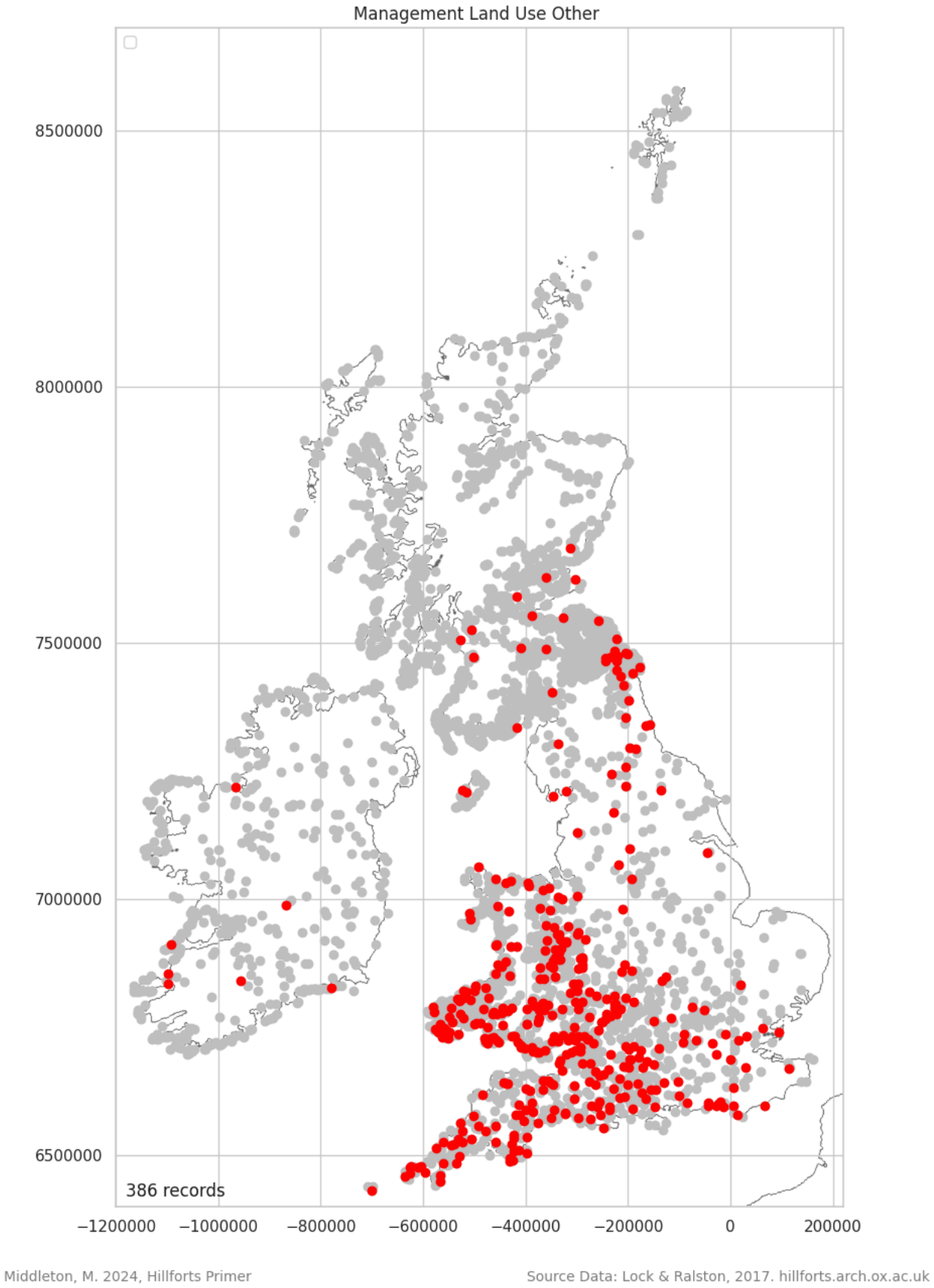


10.1%

Other Mapped

386 hillforts (9.31%) are classified as having an other land use.

In [ ]:
```
lu_other = plot_over_grey(location_land_use_data, \
                          'Management_Land_Use_Other', 'Yes')
```



9.31%

Land Use Not Recorded

There are 37 hillforts with no identified land use.

In [ ]:
```
all_no_lu = land_use_data[(land_use_data['Management_Land_Use_Woodland']=='No') &
                          (land_use_data['Management_Land_Use_Plantation']=='No') &
                          (land_use_data['Management_Land_Use_Parkland']=='No') &
                          (land_use_data['Management_Land_Use_Pasture']=='No') &
                          (land_use_data['Management_Land_Use_Arable']=='No') &
                          (land_use_data['Management_Land_Use_Scrub']=='No') &
```

```
                        (land_use_data['Management_Land_Use_Outcrop']=='No') &
                        (land_use_data['Management_Land_Use_Moorland']=='No') &
                        (land_use_data['Management_Land_Use_Heath']=='No') &
                        (land_use_data['Management_Land_Use_Urban']=='No') &
                        (land_use_data['Management_Land_Use_Coastal']=='No') &
                        (land_use_data['Management_Land_Use_Other']=='No')]
print(f'Land use not recorded: {len(all_no_lu)}')
```

```
Land use not recorded: 37
```

Five records are shown. To see the full list, update the 5, in the square brackets below, to 37 and rerun the document as described in User Settings.

In [ ]:
```
all_no_lu_full = pd.merge(name_and_number, all_no_lu, left_index=True, right_index=True)
all_no_lu_full[:5]
```

Out[ ]:

	Main_Atlas_Number	Main_Display_Name	Management_Land_Use_Woodland	Management_Land_Use_Plantation	Management_Land_Use
105	107	Castle Crag, Bampton, Cumbria	No	No	
106	108	Castle Crag, Borrowdale, Cumbria	No	No	
108	110	Castle Howe, Little Langdale, Cumbria	No	No	
805	828	Baile Iarthach Thuaidh (Ballyieragh North), Co...	No	No	
841	864	Ballydivlin, Cork (Doonlea)	No	No	

Review Management Data Split

In [ ]:
```
review_data_split(management_data, management_numeric_data, management_text_data, management_encodeable_data)
```

```
Data split good.
```

Drop Management Features

The outcrop land use feature is dropped as it has no predictive value.

In [ ]:
```
management_encodeable_data_plus = management_encodeable_data.copy()
management_encodeable_data_plus = \
management_encodeable_data_plus.drop(['Management_Land_Use_Outcrop'], axis=1)
management_encodeable_data_plus.head()
```

Out[ ]:

	Management_Condition_Extant	Management_Condition_Cropmark	Management_Condition_Destroyed	Management_Land_Use_Woodland
0	Yes	No	No	Yes
1	Yes	No	No	No
2	Yes	No	No	Yes
3	Yes	No	No	Yes
4	Yes	No	No	No

Management Data Packages

In [ ]:
```
management_data_list = \
 [management_numeric_data, management_text_data, management_encodeable_data_plus]
```

Management Data Download Packages

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

In [ ]:
```
download(management_data_list, 'Management_package')
```

Landscape Data

The Landscape features are subdivided into four subcategories:

- Type
- Topography
- Aspect
- Altitude

In [ ]:
```
landscape_features = [
 'Landscape_Type_Contour',
 'Landscape_Type_Partial',
 'Landscape_Type_Promontory',
 'Landscape_Type_Hillslope',
 'Landscape_Type_Level',
 'Landscape_Type_Marsh',
 'Landscape_Type_Multiple',
 'Landscape_Type_Comments',
 'Landscape_Topography_Hilltop',
 'Landscape_Topography_Coastal',
 'Landscape_Topography_Inland',
 'Landscape_Topography_Valley',
 'Landscape_Topography_Knoll',
 'Landscape_Topography_Ridge',
 'Landscape_Topography_Scarp',
 'Landscape_Topography_Hillslope',
 'Landscape_Topography_Lowland',
 'Landscape_Topography_Spur',
 'Landscape_Topography_Comments',
 'Landscape_Topography_Dominant',
 'Landscape_Aspect_N',
 'Landscape_Aspect_NE',
 'Landscape_Aspect_E',
 'Landscape_Aspect_SE',
 'Landscape_Aspect_S',
 'Landscape_Aspect_SW',
 'Landscape_Aspect_W',
 'Landscape_Aspect_NW',
 'Landscape_Aspect_Level',
 'Landscape_Altitude']

landscape_data = hillforts_data[landscape_features]
landscape_data.head()
```

Out[ ]:

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Landsca
0	No	Yes	Yes	No	No	
1	Yes	No	No	No	No	
2	No	Yes	No	No	No	
3	No	No	No	Yes	No	
4	Yes	No	No	No	No	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 30 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Landscape_Type_Contour          4147 non-null   object
 1   Landscape_Type_Partial          4147 non-null   object
 2   Landscape_Type_Promontory       4147 non-null   object
 3   Landscape_Type_Hillslope        4147 non-null   object
 4   Landscape_Type_Level            4147 non-null   object
 5   Landscape_Type_Marsh            4147 non-null   object
 6   Landscape_Type_Multiple         4147 non-null   object
 7   Landscape_Type_Comments         2501 non-null   object
 8   Landscape_Topography_Hilltop    4147 non-null   object
 9   Landscape_Topography_Coastal    4147 non-null   object
 10  Landscape_Topography_Inland     4147 non-null   object
 11  Landscape_Topography_Valley     4147 non-null   object
 12  Landscape_Topography_Knoll      4147 non-null   object
 13  Landscape_Topography_Ridge      4147 non-null   object
 14  Landscape_Topography_Scarp      4147 non-null   object
 15  Landscape_Topography_Hillslope  4147 non-null   object
 16  Landscape_Topography_Lowland    4147 non-null   object
 17  Landscape_Topography_Spur       4147 non-null   object
 18  Landscape_Topography_Comments   135 non-null    object
 19  Landscape_Topography_Dominant   2300 non-null   object
 20  Landscape_Aspect_N              4147 non-null   object
 21  Landscape_Aspect_NE             4147 non-null   object
 22  Landscape_Aspect_E              4147 non-null   object
 23  Landscape_Aspect_SE             4147 non-null   object
 24  Landscape_Aspect_S              4147 non-null   object
 25  Landscape_Aspect_SW             4147 non-null   object
 26  Landscape_Aspect_W              4147 non-null   object
 27  Landscape_Aspect_NW             4147 non-null   object
 28  Landscape_Aspect_Level          4147 non-null   object
 29  Landscape_Altitude              4115 non-null   float64
dtypes: float64(1), object(29)
memory usage: 972.1+ KB
```

Landscape Numeric Data

There is a single numeric feature in the Landscape data package.

In [ ]:
```
landscape_numeric_features = [
 'Landscape_Altitude']

landscape_numeric_data = \
hillforts_data[landscape_numeric_features].copy()
landscape_numeric_data.head()
```

Out[ ]:

	Landscape_Altitude
0	276.0
1	150.0
2	225.0
3	150.0
4	338.0

In [ ]:
```
landscape_numeric_data['Landscape_Altitude'].isna().sum()
```

Out[ ]:
```
32
```

There are 32 records containing no altitude information. The first 5 are listed. To see the full list, update the 5, in the square brackets below, to 32 and rerun the document as described in User Settings.

In [ ]:
```
nan_altitude_features = \
name_and_number_features + location_numeric_data_short_features + \
landscape_numeric_features
nan_altitude_data = hillforts_data[nan_altitude_features]
nan_altitude_data[nan_altitude_data['Landscape_Altitude'].isna()][:5]
```

Out[ ]:

	Main_Atlas_Number	Main_Display_Name	Location_X	Location_Y	Landscape_Altitude
355	365	Harborough Hill, Churchill, Worcestershire	-240920	6874690	NaN
446	464	Wadbury Camp, Somerset (Wadbury Hillfort)	-265052	6663609	NaN
531	550	Norham Castle, Northumberland	-239382	7503047	NaN
1376	1437	Glol Camp, Flintshire	-369738	7037341	NaN
1742	1830	Prae Wood, Hertfordshire (St. Albans; Verulami...	-41506	6755099	NaN

Landscape Numeric Data - Resolve Null Values

Test to see if Landscape Altitude contains the value -1.

In [ ]:
```
test_num_list_for_minus_one(landscape_numeric_data, \
 ['Landscape_Altitude'])
```

```
Landscape_Altitude 0
```

Update null values to -1.

In [ ]:
```
landscape_numeric_data = \
update_num_list_for_minus_one(landscape_numeric_data, ['Landscape_Altitude'])
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 1 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Landscape_Altitude  4147 non-null   float64
dtypes: float64(1)
memory usage: 32.5 KB
```

Altitude Data

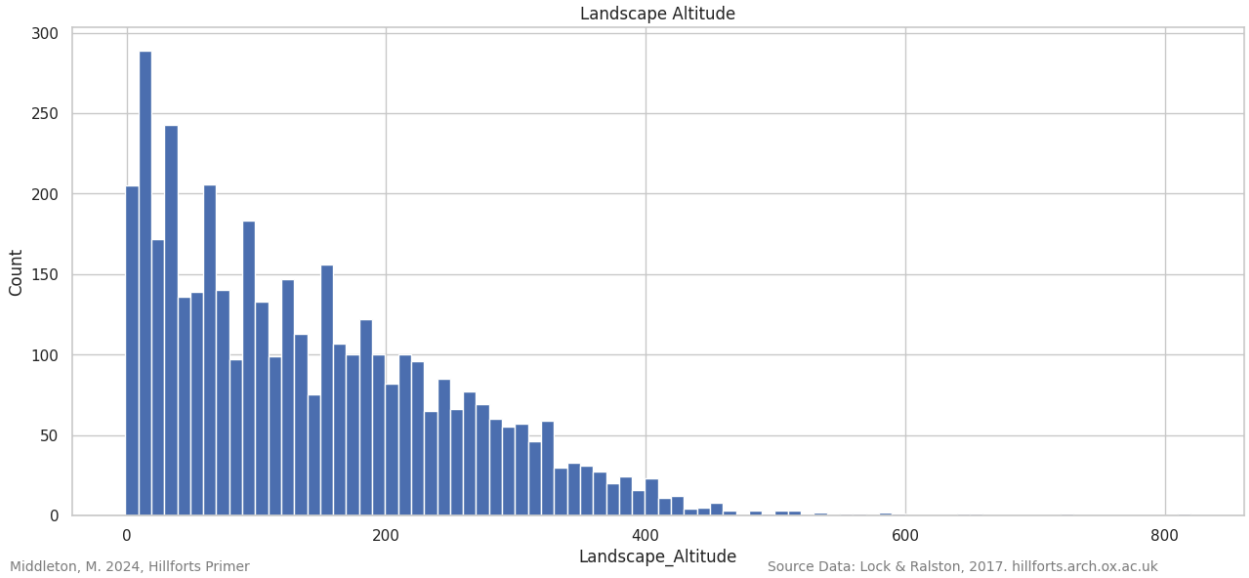
Most hillforts are located below 460m although there are outliers up to 820m. The data below 250m is grouped around peaks in the data at periodic intervals. This suggests the recording of altitude, in some cases, is not precise but, has often been generalised to the nearest 10, 25, 50 or 100m.

Altitude Data Plotted

In [ ]:
```
landscape_numeric_data['Landscape_Altitude'].describe()
```

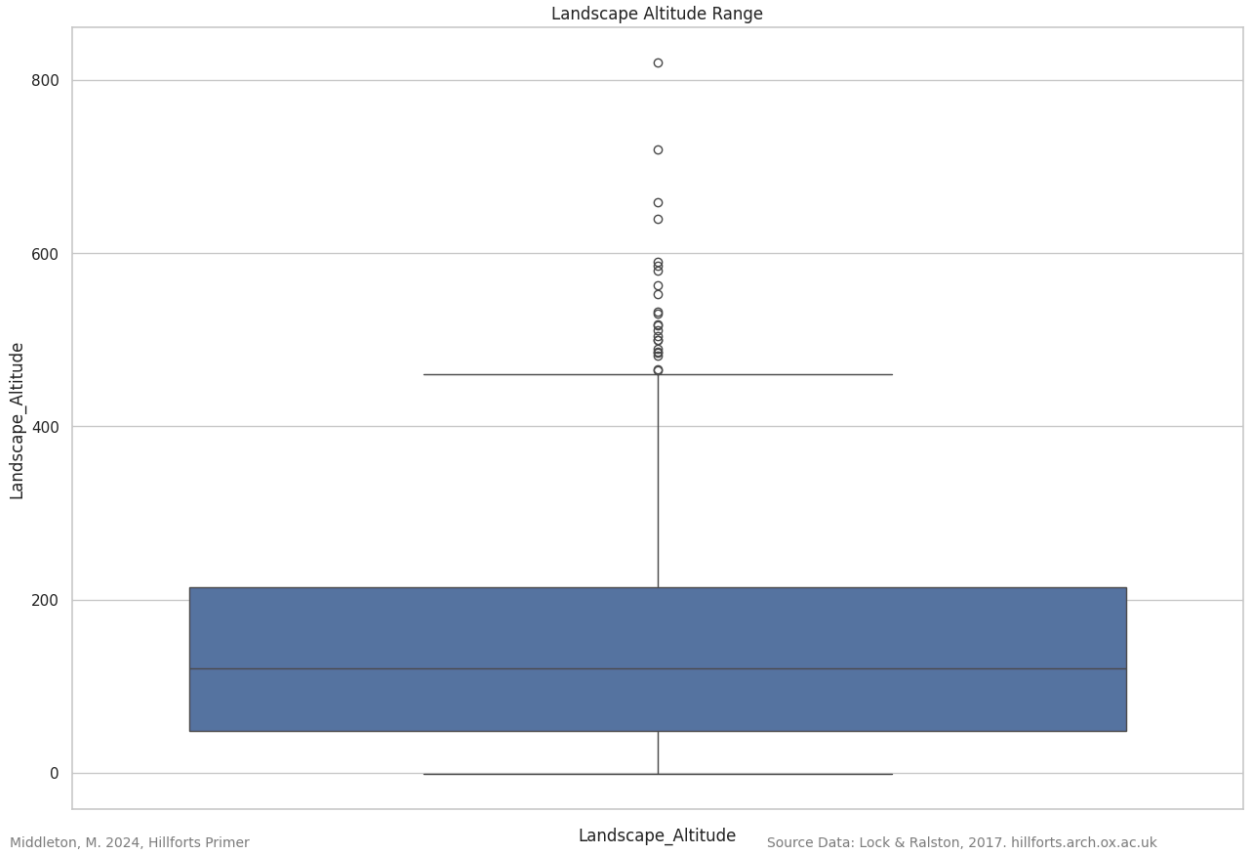
Out[ ]:
```
count    4147.000000
mean      141.504558
std       111.052905
min        -1.000000
25%        48.500000
50%       120.000000
75%       214.000000
max       820.000000
Name: Landscape_Altitude, dtype: float64
```

In [ ]:
```
plot_bar_chart_numeric(landscape_numeric_data, 1, \
                       'Landscape_Altitude', 'Count', 'Landscape_Altitude', 82)
```



95.6% of hillforts are located below 460m. The majority (the first three percentiles - 75% of the data) are located toward the lower end of the range, below 214m; 50% are below 120m and 25% are below 48.5m.

In [ ]:
```
Landscape_Altitude_stats = \
plot_data_range(landscape_numeric_data['Landscape_Altitude'], \
                'Landscape_Altitude')
```



In [ ]:
```
Landscape_Altitude_stats
```

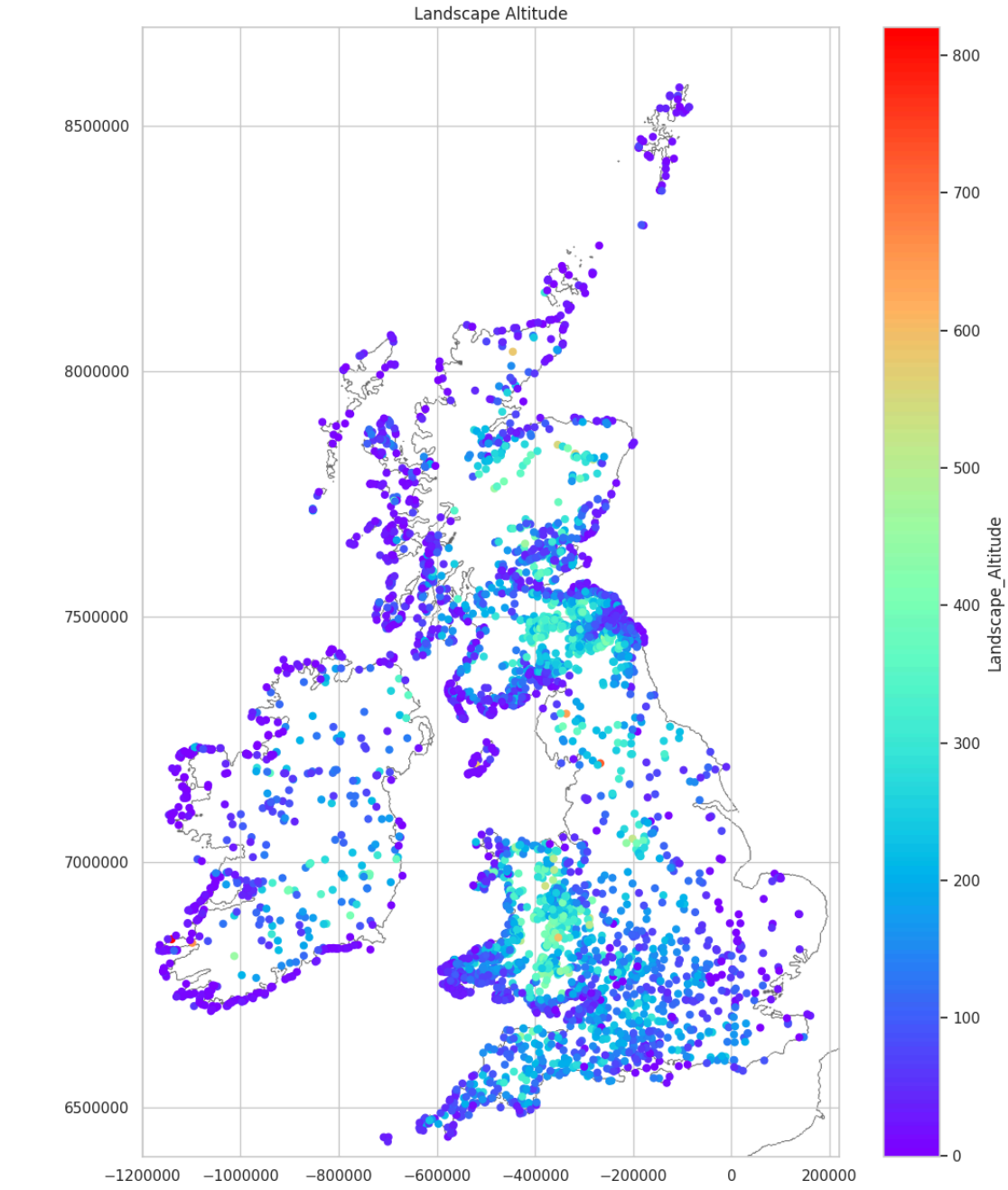
Out[ ]:
```
[-1.0, 48.5, 120.0, 214.0, 460.0]
```

Altitude Mapped

With so many of the hillforts having an altitude toward the lower end of the range, the outliers cause the map of hillforts by altitude to be not particularly revealing.

In [ ]:
```
location_landscape_numeric_data = \
pd.merge(location_numeric_data_short, landscape_numeric_data, left_index=True, \
         right_index=True)
```

In [ ]:
```
plot_type_values(location_landscape_numeric_data, 'Landscape_Altitude', \
                 'Landscape_Altitude')
```



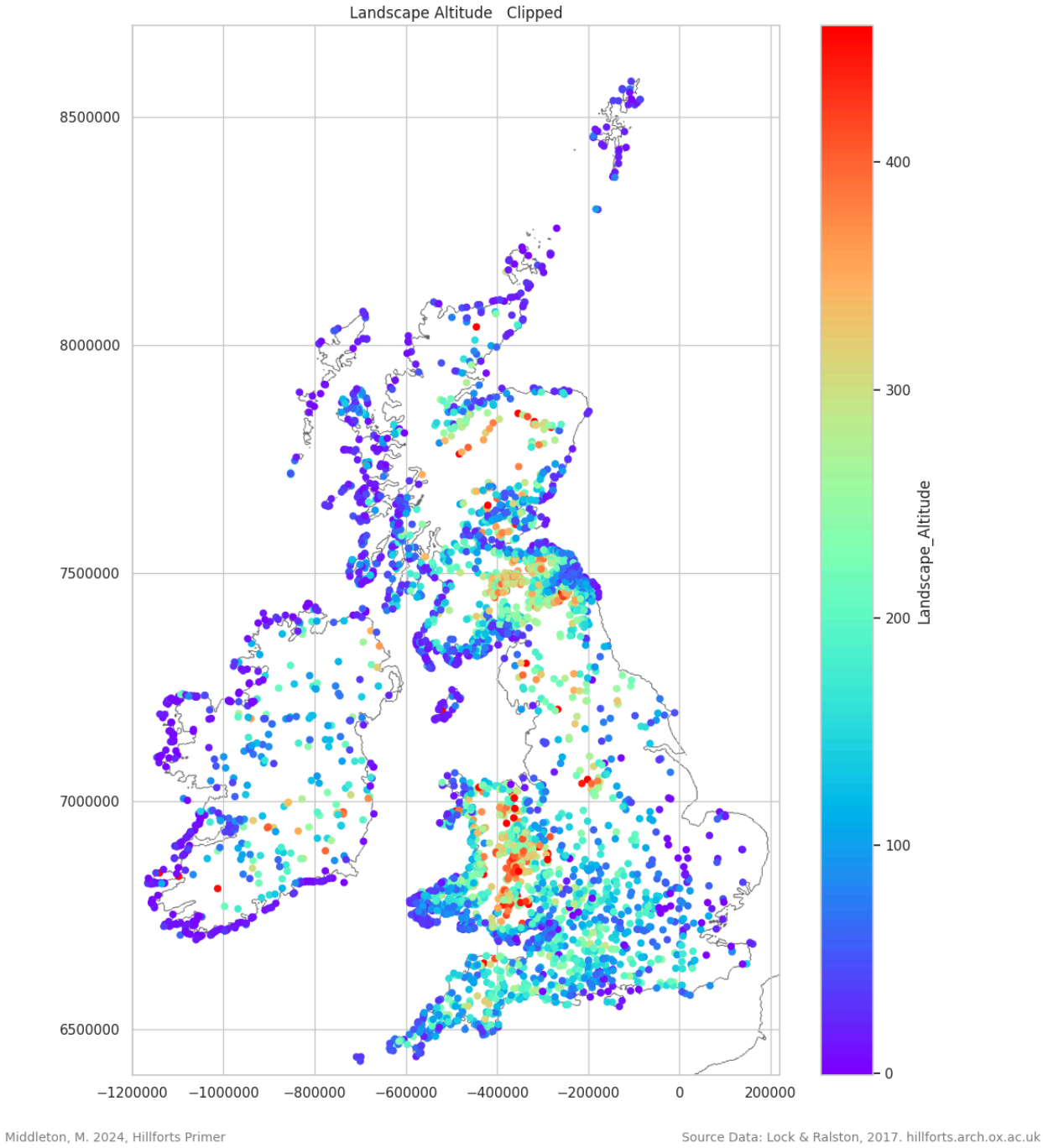
Altitude Capped Mapped

To improve the clarity of the distribution plot, the altitude values are capped at the top end of the fourth quartile, to 460m.

In [ ]:
```
location_landscape_numeric_data_clip = location_landscape_numeric_data.copy()
location_landscape_numeric_data_clip['Landscape_Altitude'].\
where(location_landscape_numeric_data_clip['Landscape_Altitude'] < \
      Landscape_Altitude_stats[4], Landscape_Altitude_stats[4], inplace=True)
location_landscape_numeric_data_clip['Landscape_Altitude'].describe()
```

Out[ ]:
```
count    4147.000000
mean      140.993827
std       109.176613
min        -1.000000
25%        48.500000
50%       120.000000
75%       214.000000
max       460.000000
Name: Landscape_Altitude, dtype: float64
```

In [ ]:
```
plot_type_values(location_landscape_numeric_data_clip, 'Landscape_Altitude', 'Landscape_Altitude - Clipped')
```



Altitude Over 460m Mapped (Outliers)

There are 23 hillforts (0.55%) over 460m. The highest is Faha in Kerry, Ireland at 820m. The ten highest are listed below.

In [ ]:
```
over_460 = \
location_landscape_numeric_data\
 [location_landscape_numeric_data['Landscape_Altitude'] > 460].copy()
len(over_460)
```

Out[ ]:
```
23
```

To see the full list, update the 10, at the end of the second line below, to 23 and rerun the document as described in User Settings. (eg ... =False).head(23))

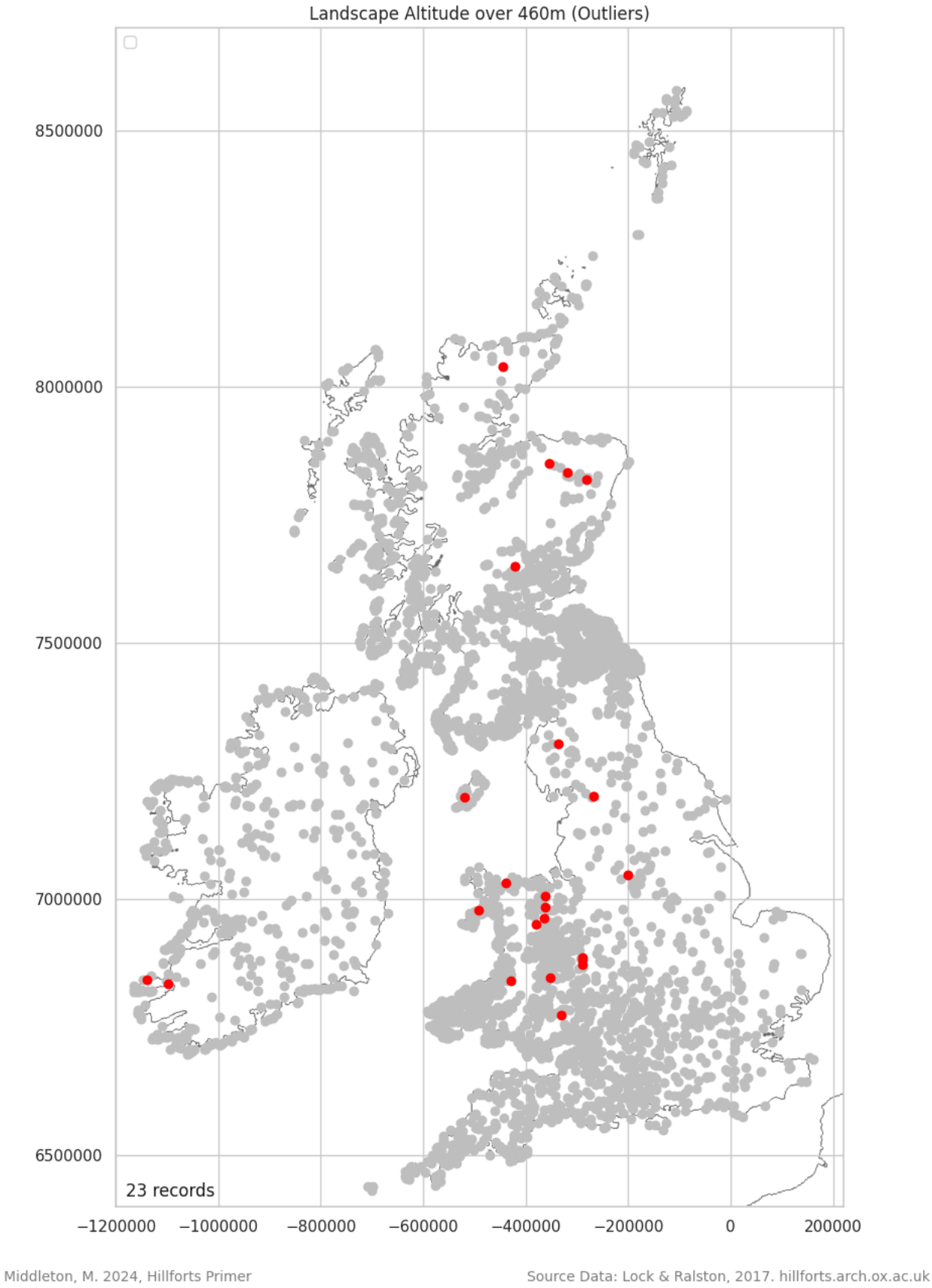
In [ ]:
```
over_460_plus = \
pd.merge(name_and_number, over_460, left_index=True, right_index=True)
over_460_plus[['Main_Display_Name','Landscape_Altitude']].\
sort_values(by='Landscape_Altitude', ascending=False).head(10)
```

Out[ ]:

	Main_Display_Name	Landscape_Altitude
647	Faha, Kerry (Begagh; Binn na Port; An Fhaiche)	820.0
2388	Ingleborough, North Yorkshire	720.0
642	Caherconree, Kerry (Ballyarkane Oughter, Behee...	659.0
104	Carrock Fell, Cumbria	640.0
2607	The Whimble, Powys	590.0
3035	South Barrule, Rushen	586.0
2619	Ben Griam Beg, Highland	580.0
2767	Tap o' Noth, Aberdeenshire (Hill of Noth)	563.0
2762	Little Conval, Moray	553.0
1355	Craig Rhiwarth, Powys	533.0

In [ ]:

```
over_460['Landscape_Altitude'] = "Yes"
over_460_stats = \
plot_over_grey(over_460, 'Landscape_Altitude', 'Yes', 'over 460m (Outliers)')
```



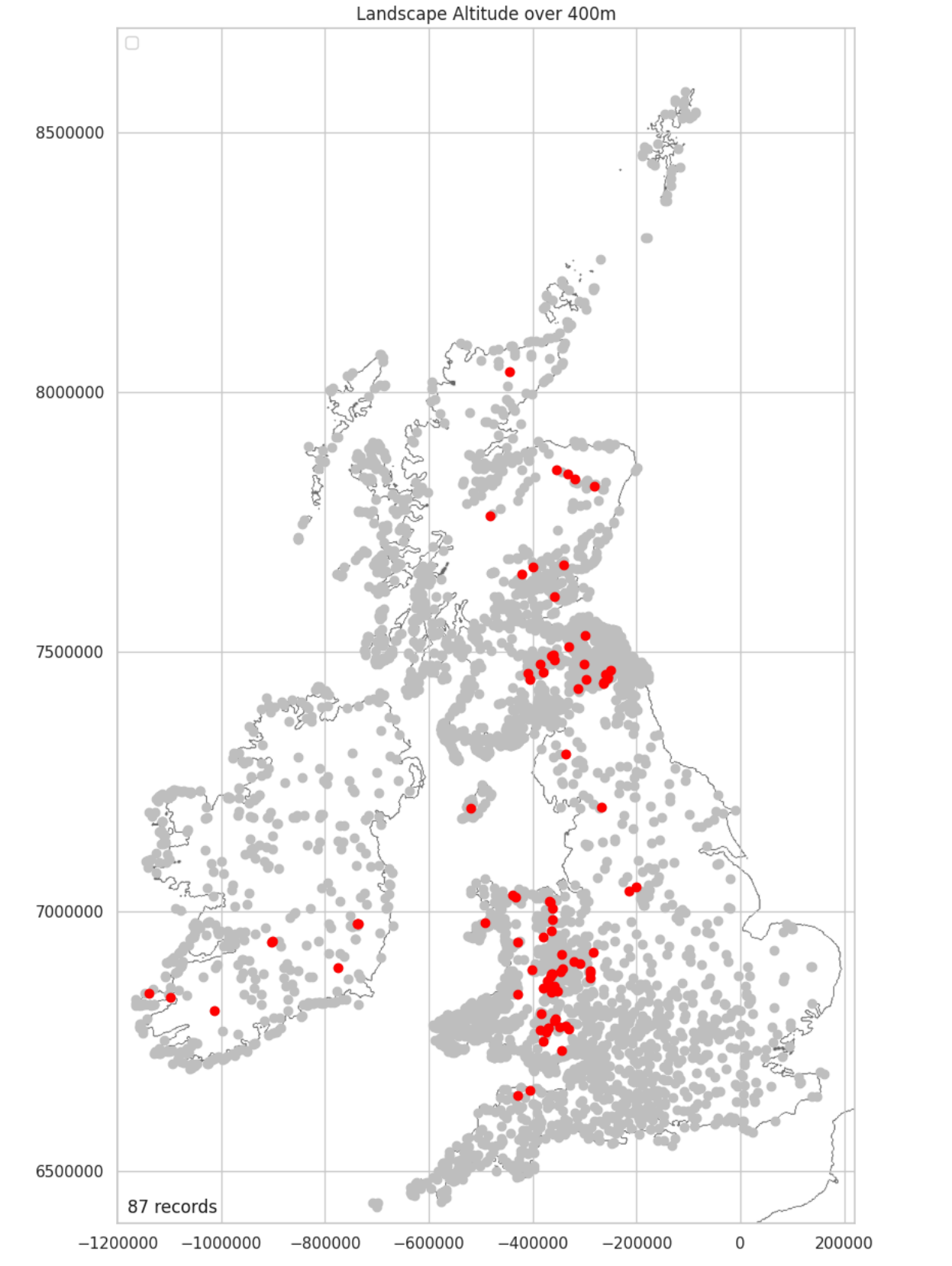
`0.55%`

Altitude Over 400m Mapped

87 hillforts (2.1%) are located above 400m. There is a notable pairing of hillforts (by proximity) across the central region of the southern uplands.

In [ ]:
```
over_400 = \
location_landscape_numeric_data_clip\
 [location_landscape_numeric_data_clip['Landscape_Altitude'] >= 400].copy()
over_400['Landscape_Altitude'] = "Yes"
```

In [ ]:
```
over_400_stats = plot_over_grey(over_400, 'Landscape_Altitude', 'Yes', 'over 400m')
```

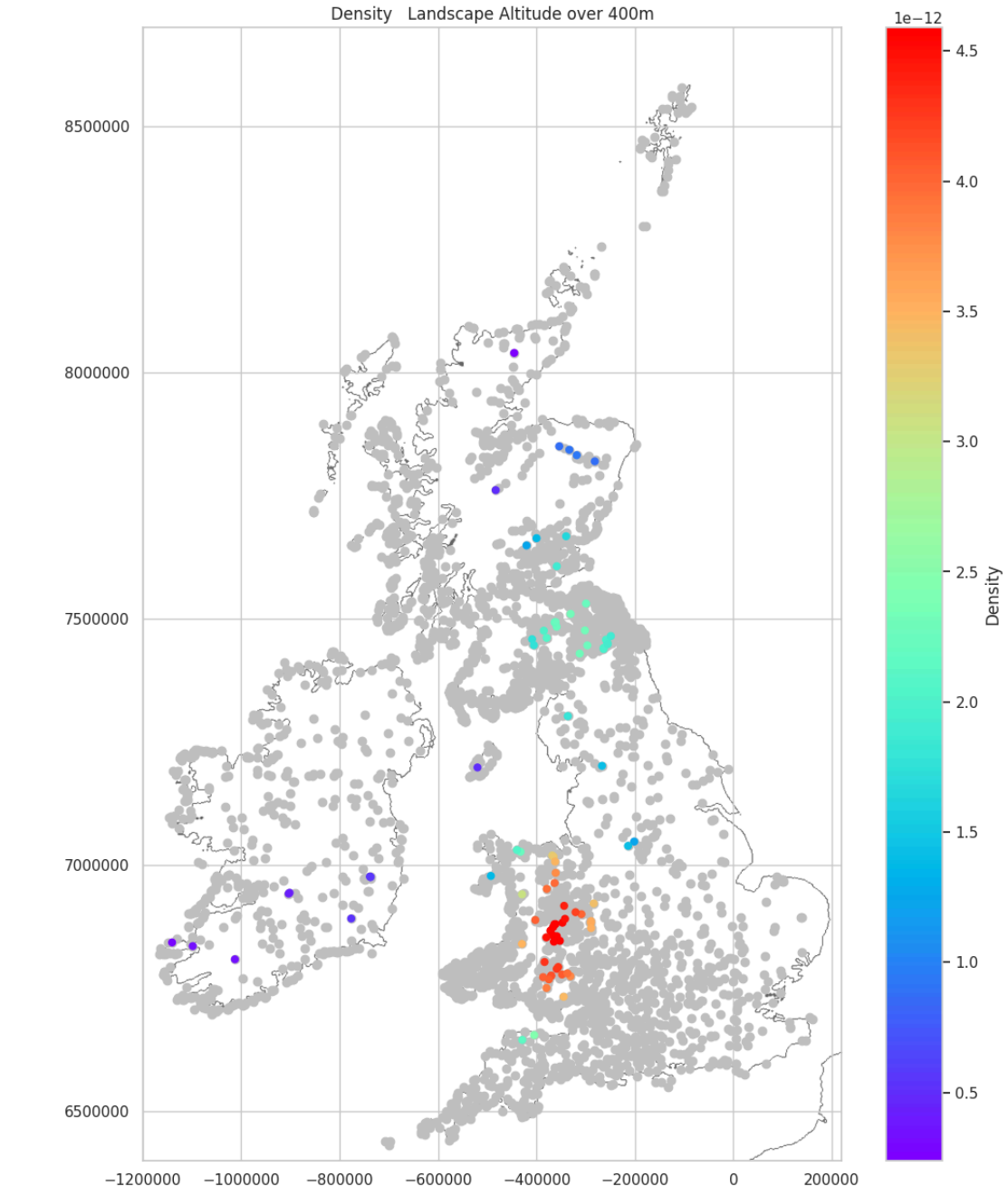


2.1%

Altitude Over 400m Density Mapped

Most hillforts above 400m are in the Cambrian Mountains.

In [ ]:
```
plot_density_over_grey(over_400_stats, 'Landscape_Altitude over 400m')
```

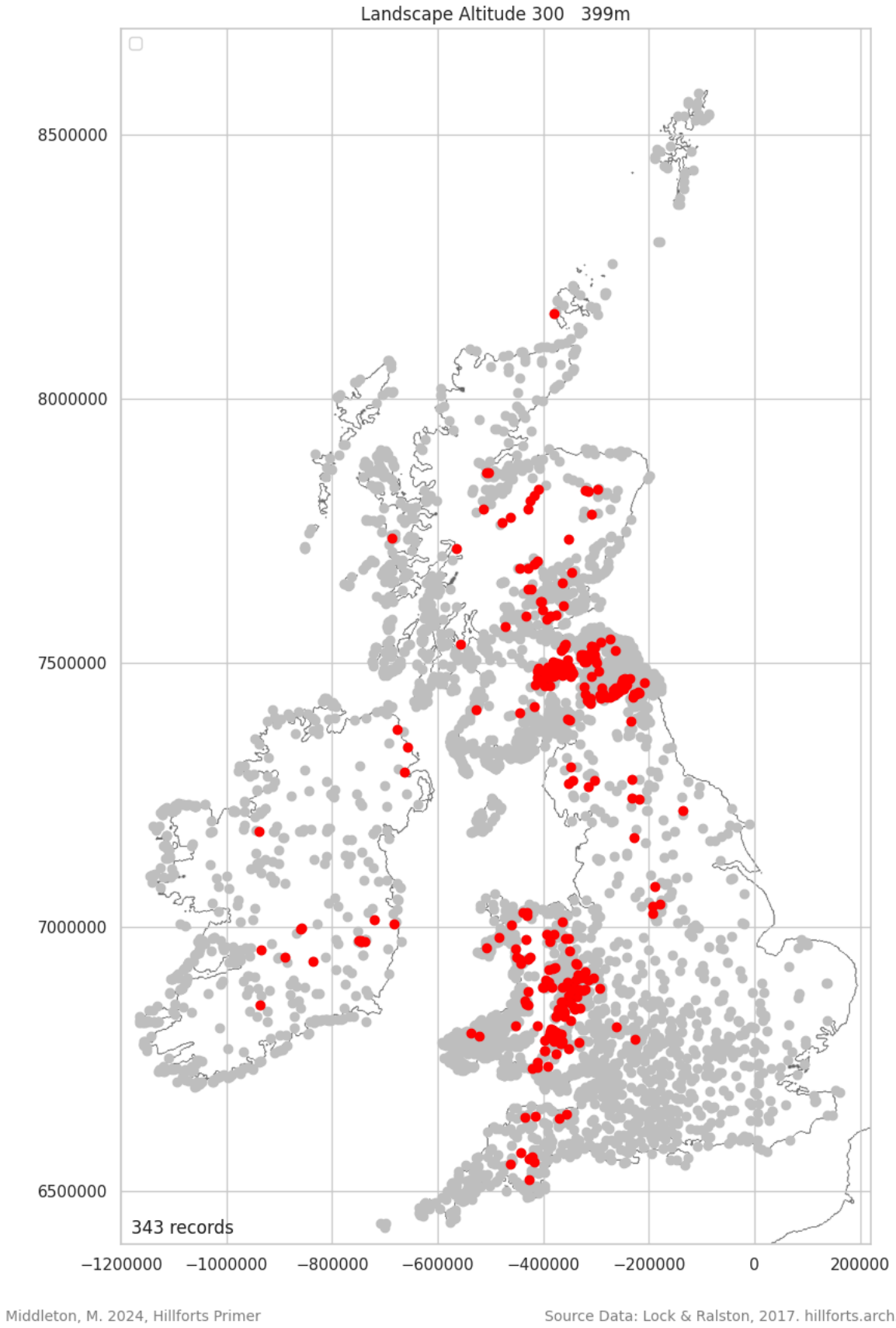


Altitude 300 - 399m Mapped

There are two dominant groups of hillforts in the 300 to 399m data; over the Southern Uplands and along the Cambrian Mountains. There are a number of noticeable alignments along ridges in the southern cluster and the ring of hills around the Tweed basin is clearly distinguishable in the Northeast.

In [ ]:
```
over_300 = \
location_landscape_numeric_data_clip\
 [(location_landscape_numeric_data_clip['Landscape_Altitude'] >= 300) & \
  (location_landscape_numeric_data_clip['Landscape_Altitude'] < 400)].copy()
over_300['Landscape_Altitude'] = "Yes"
```

In [ ]:
```
over_300_stats = plot_over_grey(over_300, 'Landscape_Altitude', 'Yes', '300 - 399m')
```

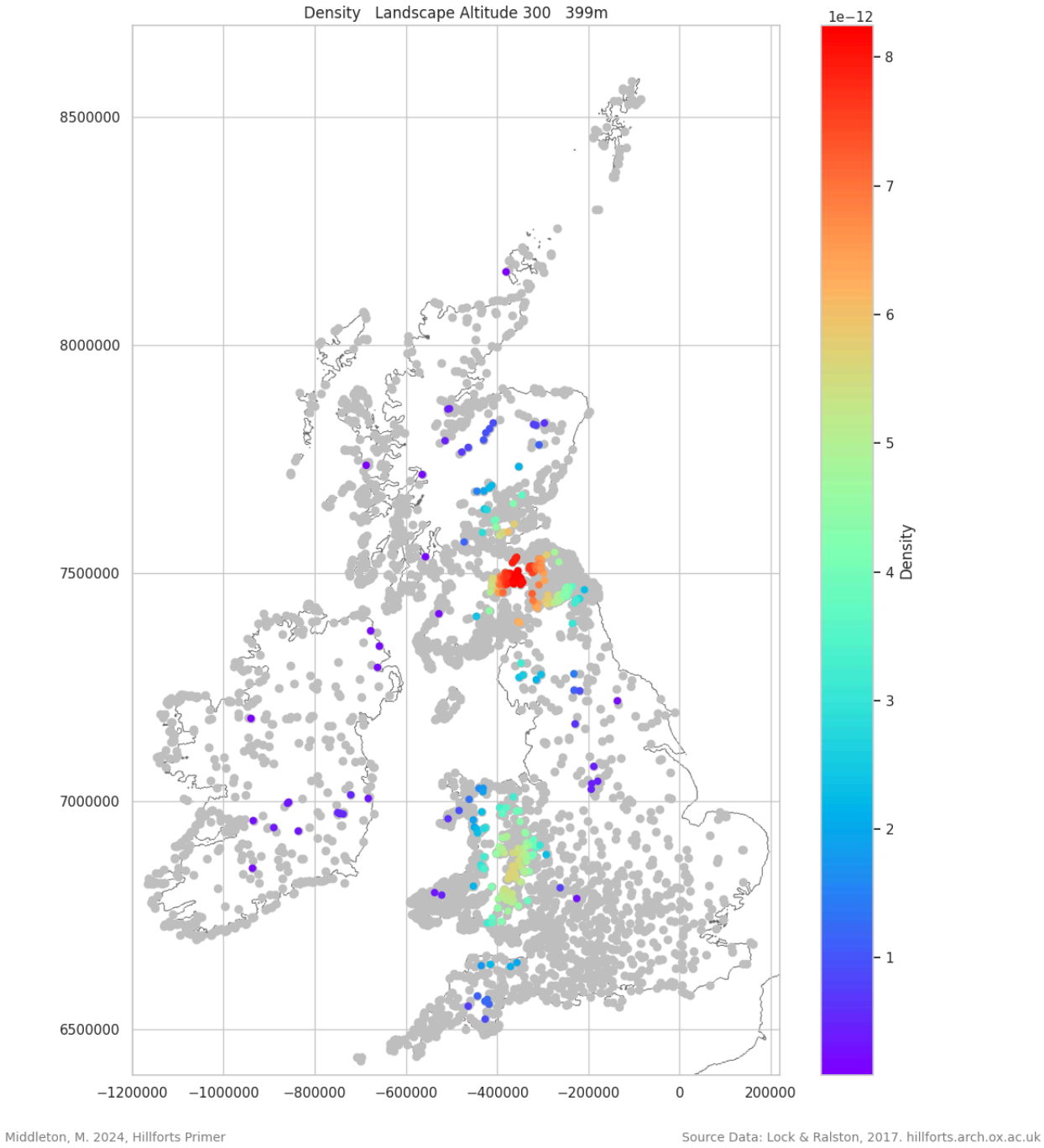


8.27%

Altitude Over 300 - 399m Density Mapped

Most hillforts between 300 and 399m cluster over the Moorfoot and Tweedsmuir Hills. There is a secondary cluster along the eastern flank of the Cambrian Mountains.

In [ ]:
```
plot_density_over_grey(over_300_stats, 'Landscape_Altitude 300 - 399m')
```

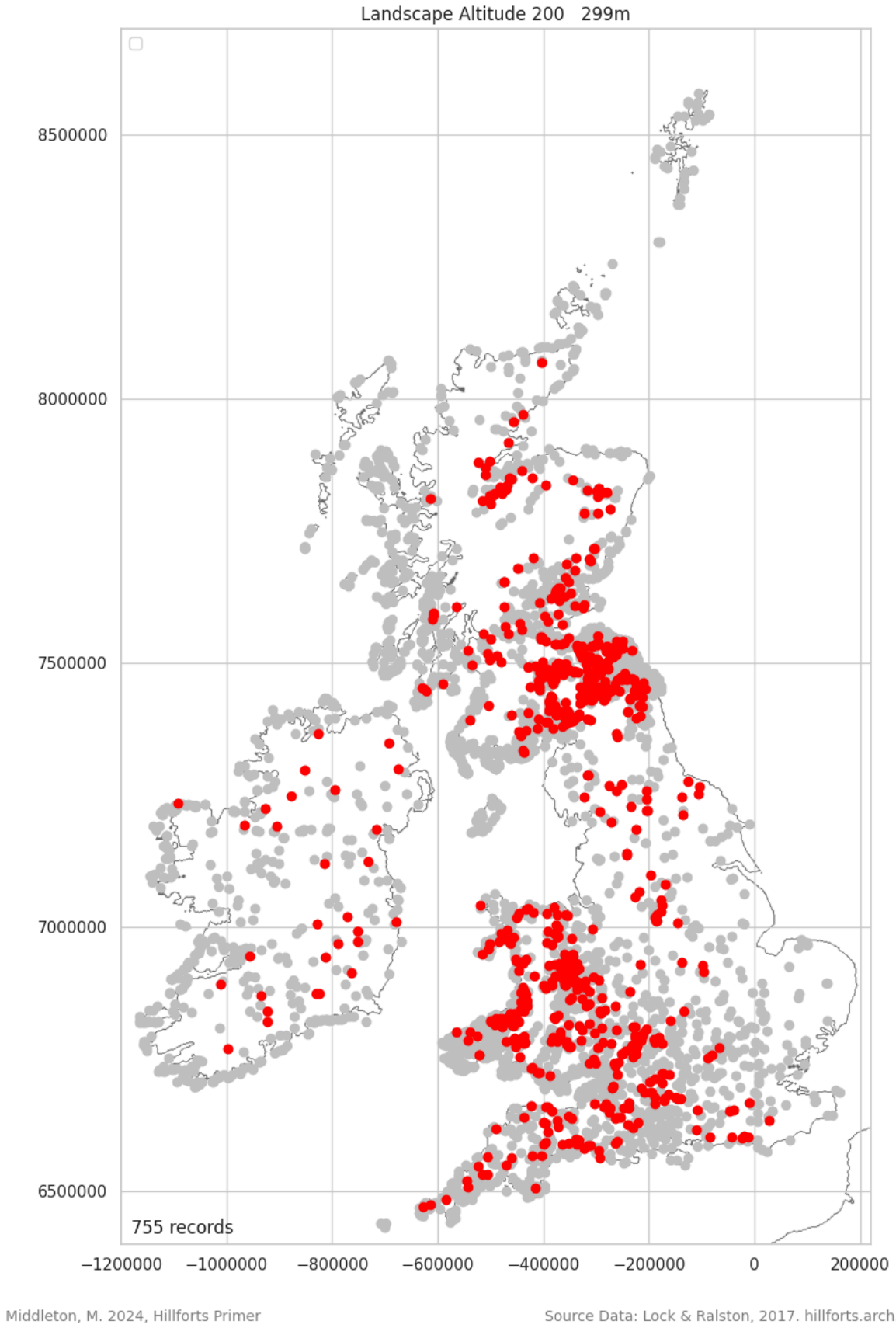


Altitude 200 - 299m Mapped

Most hillforts between 200 and 299m are located in the Southern Uplands and the Cambrian Mountains. There are a good number of forts across the south and southeast of England as well as along the edges of the highlands, along the Highland Boundary fault, and up the east coast, to the north of the Grampian Mountains and into the Great Glen. There are a peppering of forts along the Pennines and across central and eastern Ireland.

In [ ]:
```
over_200 = \
location_landscape_numeric_data_clip\
 [(location_landscape_numeric_data_clip['Landscape_Altitude'] >= 200) & \
  (location_landscape_numeric_data_clip['Landscape_Altitude'] < 300)].copy()
over_200['Landscape_Altitude'] = "Yes"
```

In [ ]:
```
over_200_stats = plot_over_grey(over_200, 'Landscape_Altitude', 'Yes', '200 - 299m')
```

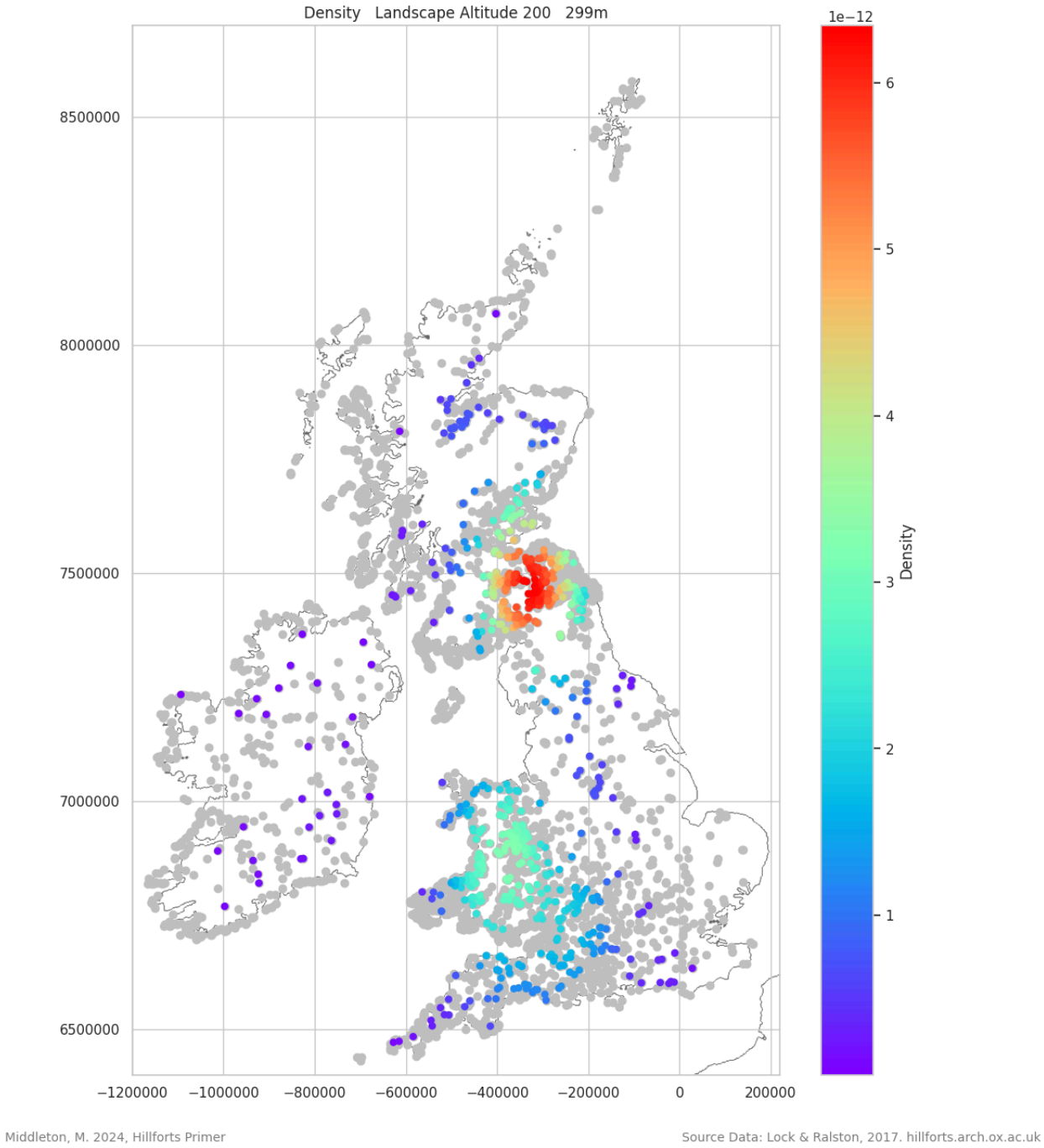


18.21%

Altitude Over 200 - 299m Density Mapped

There is a main cluster in the Southern Uplands with a second cluster across the Cambrian Mountains.

In [ ]:
```
plot_density_over_grey(over_200_stats, 'Landscape_Altitude 200 - 299m')
```

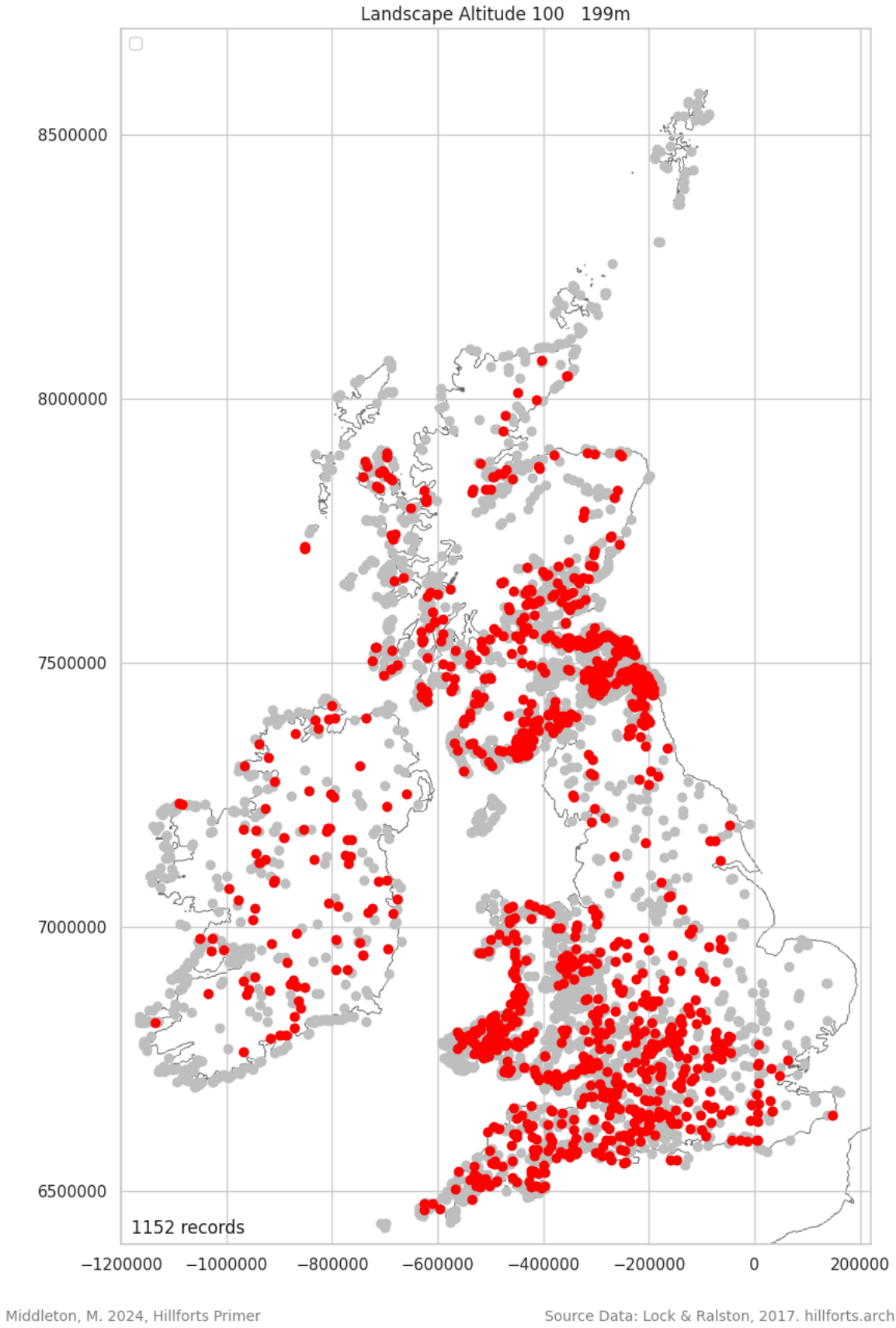


Altitude 100 - 199m Mapped

At lower altitudes we start to see hillforts spreading out over the South and Southeast, Pembrokeshire, the Northeast, the Highland Boundary Fault, either end of the Northwest hillforts density cluster, around the coast of the Moray Firth and peppered right across Ireland.

In [ ]:
```
over_100 = \
location_landscape_numeric_data_clip\
 [(location_landscape_numeric_data_clip['Landscape_Altitude'] >= \
   100) & (location_landscape_numeric_data_clip['Landscape_Altitude'] < \
           200)].copy()
over_100['Landscape_Altitude'] = "Yes"
```

In [ ]:
```
over_100_stats = plot_over_grey(over_100, 'Landscape_Altitude', 'Yes', \
                                '100 - 199m')
```

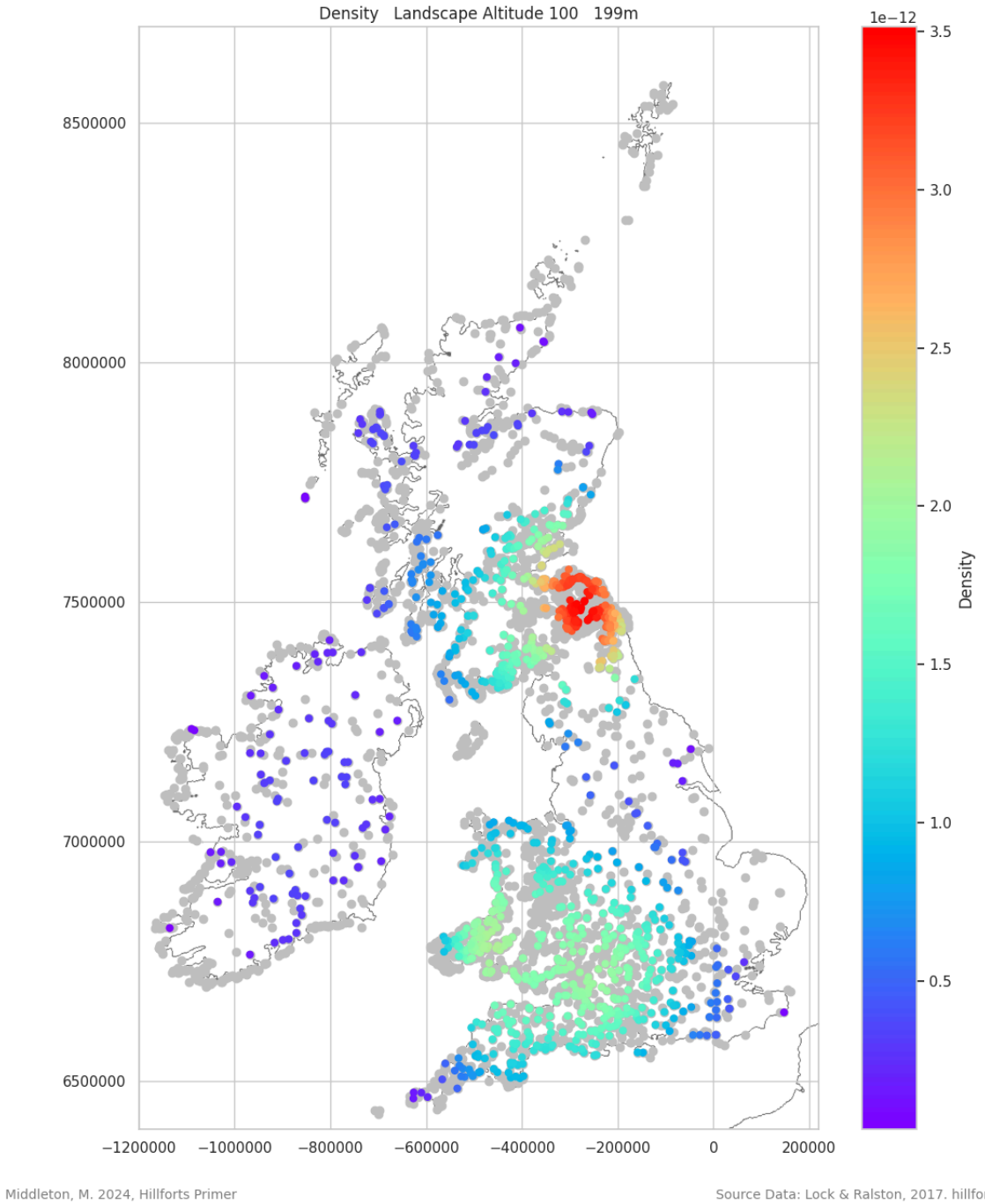


27.78%

Altitude Over 100 - 199m Density Mapped

The most intense cluster continues to be in the Northeast, spreading right across southern Scotland. A second cluster can be seen running from south Wales into south, central England.

In [ ]:
```
plot_density_over_grey(over_100_stats, 'Landscape_Altitude 100 - 199m')
```

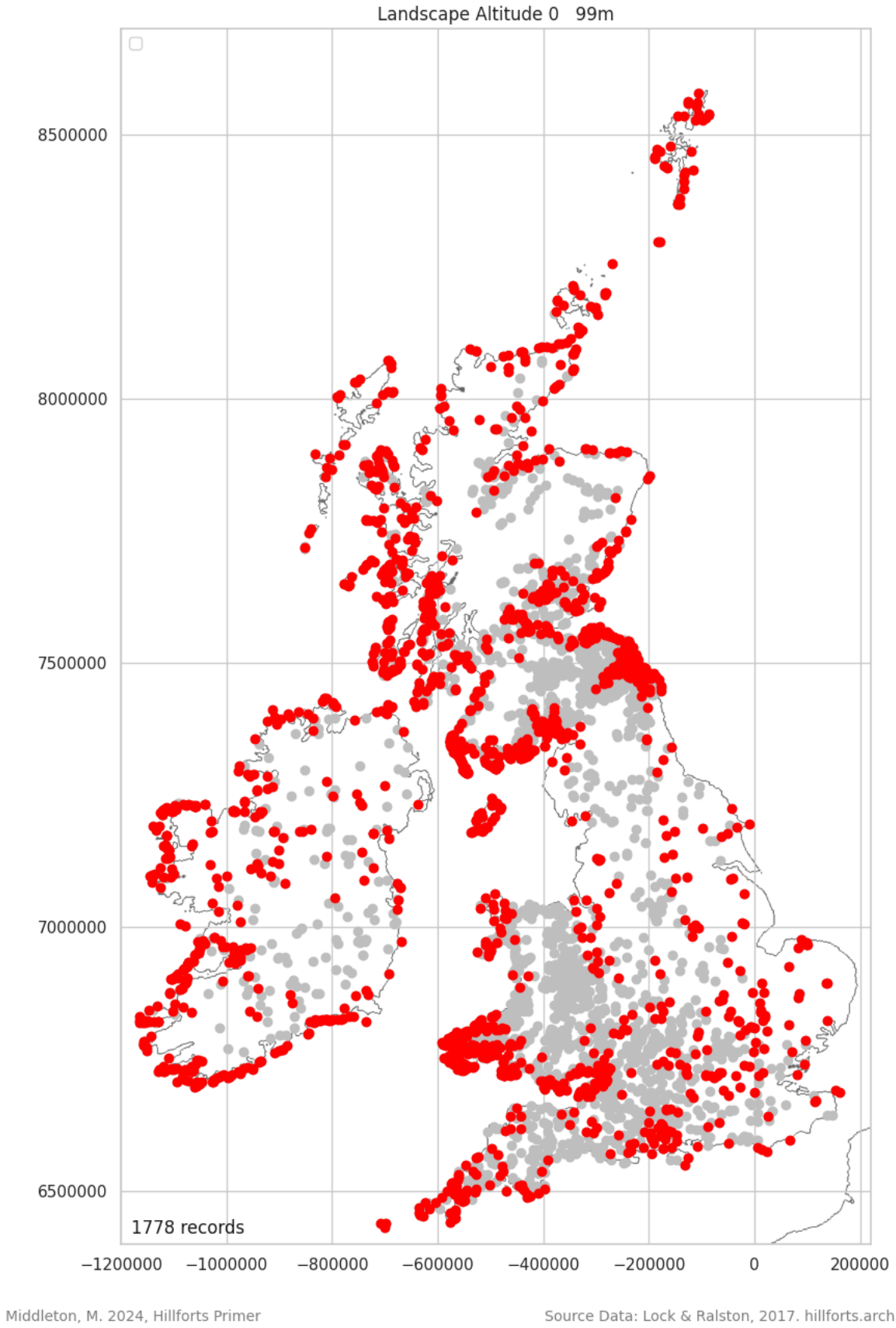


Altitude 0 - 99m Mapped

The distribution of forts below 100m is noticeably more coastal. The majority of hillforts (42.87%) are located in this altitude range.

In [ ]:
```
over_0 = \
location_landscape_numeric_data_clip\
 [(location_landscape_numeric_data_clip['Landscape_Altitude'] >= 0) & \
  (location_landscape_numeric_data_clip['Landscape_Altitude'] < 100)].copy()
over_0['Landscape_Altitude'] = "Yes"
```

In [ ]:
```
over_0_stats = plot_over_grey(over_0, 'Landscape_Altitude', 'Yes', '0 - 99m')
```

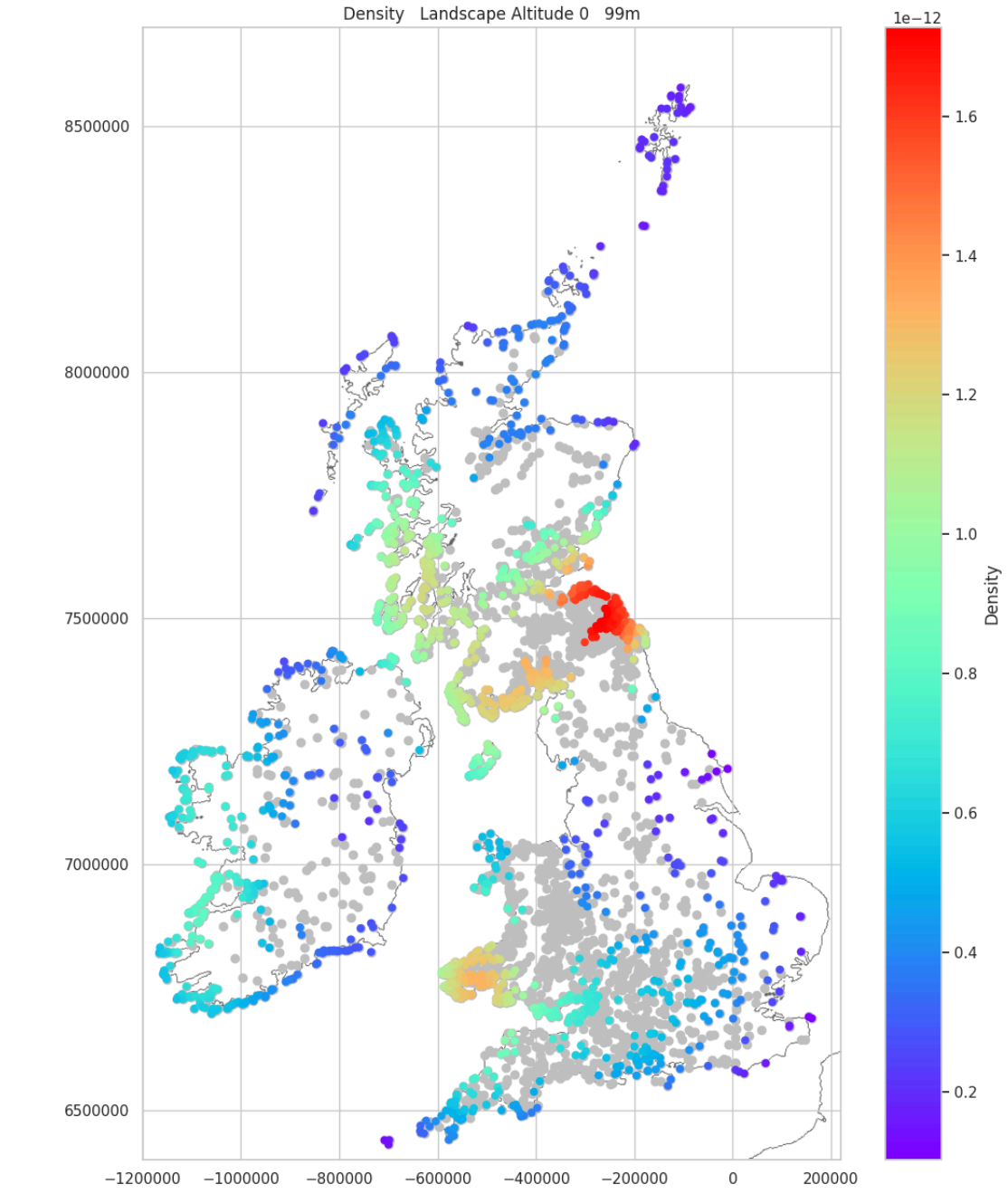


42.87%

Altitude Over 0 - 99m Density Mapped

The main cluster continues to be in the Southern Uplands, within the Tweed Basin, while Pembrokeshire, the Galloway coast, the Northwest and the west of Ireland all show as smaller clusters. All five main distribution clusters seen in, Part 1: Density Map Showing Clusters Adjusted by Region, can be seen in this altitude range.

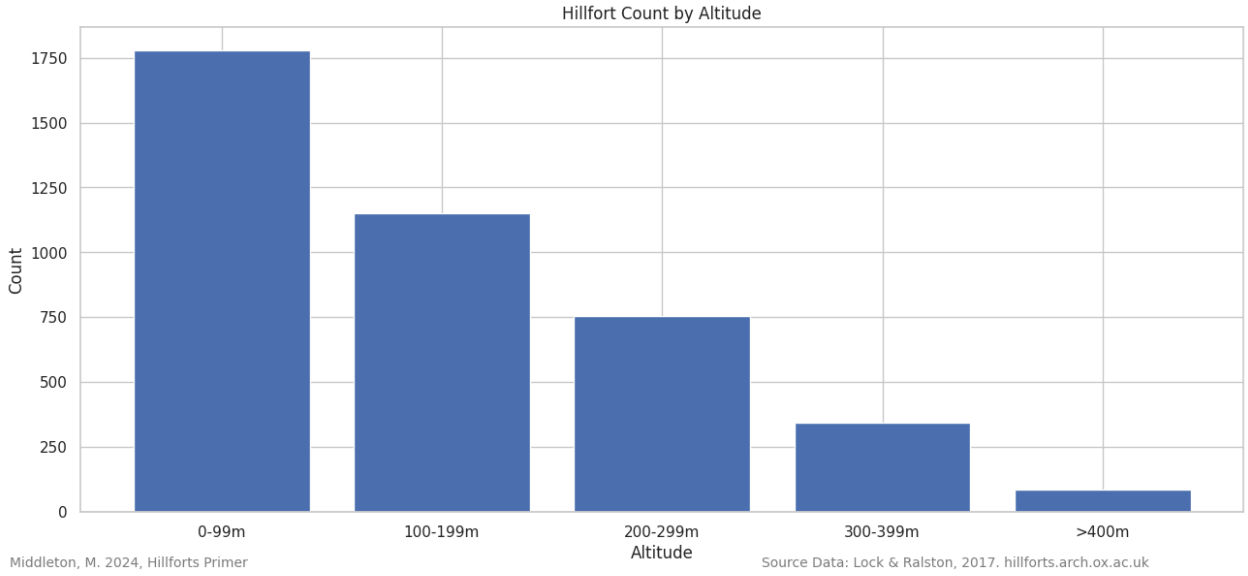
In [ ]:
```
plot_density_over_grey(over_0_stats, 'Landscape_Altitude 0 - 99m')
```



Bracketed Altitude Plotted

42.87% of hillforts are below 100m. Within this bracket, most are in close to the coast. As we climb in altitude, hillforts become less common. 94.86% of hillforts are below 300m.

In [ ]:
```
plot_bar_chart_from_list([over_0,over_100,over_200,over_300,over_400], \
 ["0-99m", "100-199m", "200-299m", "300-399m", ">400m"], "Altitude", \
                          "Count", "Hillfort Count by Altitude")
```



Landscape Text Data

There are three Landscape text features.

In [ ]:
```
landscape_text_features = [
 'Landscape_Type_Comments',
 'Landscape_Topography_Comments',
 'Landscape_Topography_Dominant']

landscape_text_data = hillforts_data[landscape_text_features]
landscape_text_data.head()
```

Out[ ]:

	Landscape_Type_Comments	Landscape_Topography_Comments	Landscape_Topography_Dominant
0	Partial contour fort following the natural con...	NaN	Hill top, part promontory.
1	Univallate, contour hillfort located on summit...	NaN	Hill top spur.
2	Located part on slopes, part level ground. Sit...	NaN	Hilltop including the Adam's Rocks outcrop
3	The site is located on NW slopes just below th...	NaN	Flat-topped steep hill.
4	One of the finest examples of a contour fort i...	NaN	Malvern Hill crest.

Landscape Text Data - Resolve Null Values

Test for the presence of 'NA' in Landscape Text Data features.

In [ ]:
```
test_cat_list_for_NA(landscape_text_data, landscape_text_features)
```

```
Landscape_Type_Comments 0
Landscape_Topography_Comments 0
Landscape_Topography_Dominant 0
```

Fill null values with 'NA'.

In [ ]:
```
landscape_text_data = \
update_cat_list_for_NA(landscape_text_data, landscape_text_features)
landscape_text_data.info()
```

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

Landscape Encodeable Data

There are 26 Landscape Encodeable Features grouped into three subcategories:

- Type

- Topography
- Aspect

In [ ]:
```
landscape_encodeable_features = [
 'Landscape_Type_Contour',
 'Landscape_Type_Partial',
 'Landscape_Type_Promontory',
 'Landscape_Type_Hillslope',
 'Landscape_Type_Level',
 'Landscape_Type_Marsh',
 'Landscape_Type_Multiple',
 'Landscape_Topography_Hilltop',
 'Landscape_Topography_Coastal',
 'Landscape_Topography_Inland',
 'Landscape_Topography_Valley',
 'Landscape_Topography_Knoll',
 'Landscape_Topography_Ridge',
 'Landscape_Topography_Scarp',
 'Landscape_Topography_Hillslope',
 'Landscape_Topography_Lowland',
 'Landscape_Topography_Spur',
 'Landscape_Aspect_N',
 'Landscape_Aspect_NE',
 'Landscape_Aspect_E',
 'Landscape_Aspect_SE',
 'Landscape_Aspect_S',
 'Landscape_Aspect_SW',
 'Landscape_Aspect_W',
 'Landscape_Aspect_NW',
 'Landscape_Aspect_Level']

landscape_encodeable_data = hillforts_data[landscape_encodeable_features]
landscape_encodeable_data.head()
```

Out[ ]:

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Landsca
0	No	Yes	Yes	No	No	
1	Yes	No	No	No	No	
2	No	Yes	No	No	No	
3	No	No	No	Yes	No	
4	Yes	No	No	No	No	

Landscape Type Data

There are seven landscape types. Partial is short for 'Partial Contour' and Multiple refers to multiple enclosures which enclose an area that can support occupation. See: Data Structure.

In [ ]:
```
landscape_type_features = [
 'Landscape_Type_Contour',
 'Landscape_Type_Partial',
 'Landscape_Type_Promontory',
 'Landscape_Type_Hillslope',
 'Landscape_Type_Level',
 'Landscape_Type_Marsh',
 'Landscape_Type_Multiple']

landscape_type_data = landscape_encodeable_data[landscape_type_features].copy()
landscape_type_data.head()
```

Out[ ]:

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Landsca
0	No	Yes	Yes	No	No	
1	Yes	No	No	No	No	
2	No	Yes	No	No	No	
3	No	No	No	Yes	No	
4	Yes	No	No	No	No	

There are no null values in the Landscape Type data.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 7 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Landscape_Type_Contour     4147 non-null   object
 1   Landscape_Type_Partial     4147 non-null   object
 2   Landscape_Type_Promontory  4147 non-null   object
 3   Landscape_Type_Hillslope   4147 non-null   object
 4   Landscape_Type_Level       4147 non-null   object
 5   Landscape_Type_Marsh       4147 non-null   object
 6   Landscape_Type_Multiple    4147 non-null   object
dtypes: object(7)
memory usage: 226.9+ KB
```

A hillfort can be multiple landscape types.

In [ ]:
```
landscape_type_data[(landscape_type_data['Landscape_Type_Contour']==\
                     'Yes')&(landscape_type_data['Landscape_Type_Promontory']==\
                             'Yes')].head(3)
```

Out[ ]:

	Landscape_Type_Contour	Landscape_Type_Partial	Landscape_Type_Promontory	Landscape_Type_Hillslope	Landscape_Type_Level	Lanc
488	Yes	No	Yes	No	Yes	
2051	Yes	No	Yes	No	No	
2278	Yes	No	Yes	No	No	

There are 50 hillforts where a landscape type has not been recorded.

In [ ]:
```
no_type = \
landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='No') \
                    &(landscape_type_data['Landscape_Type_Partial']=='No') \
                    &(landscape_type_data['Landscape_Type_Promontory']=='No') \
                    &(landscape_type_data['Landscape_Type_Hillslope']=='No') \
                    &(landscape_type_data['Landscape_Type_Level']=='No') \
                    &(landscape_type_data['Landscape_Type_Marsh']=='No') \
                    &(landscape_type_data['Landscape_Type_Multiple']=='No')]
print(f'Landscape Type not recorded: {len(no_type)}.')
```

```
Landscape Type not recorded: 50.
```

Five records are shown. To see the full list, update the 5, in the square brackets below, to 50 and rerun the document as described in User Settings.

In [ ]:
```
landscape_data_no_type = \
pd.merge(name_and_number, no_type, left_index=True, right_index=True)
landscape_data_no_type[['Main_Atlas_Number','Main_Display_Name']][:5]
```

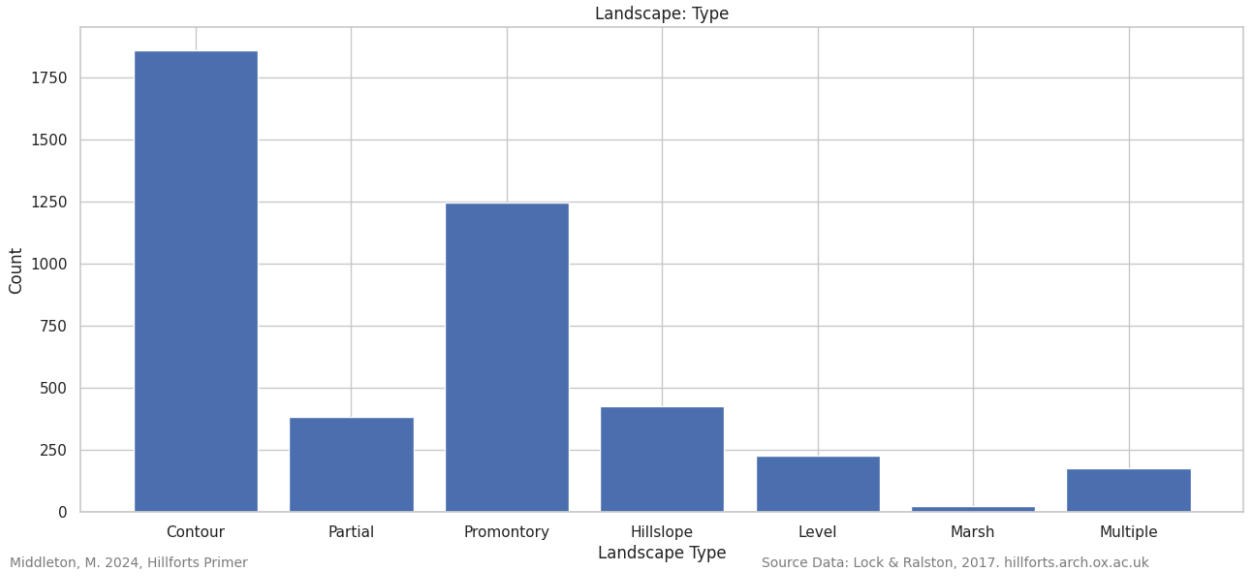
Out[ ]:

	Main_Atlas_Number	Main_Display_Name
178	185	Dunbae Glen, Dumfries & Galloway
275	282	Kennan's Isle, Tongland Loch, Dumfries & Gallo...
531	550	Norham Castle, Northumberland
770	792	Carleton Hill, South Ayrshire
845	868	Broadgate, Dumfries & Galloway

Landscape Type Data Short Plotted

Contour hillforts (44.85%) and Partial Contour hillforts (9.26%) make up 54.11% of all forts. Promontory forts make up the next 30.09%. Hillslope forts account for 10.3%; Level forts 5.45% and Marsh forts make up just 0.55% of hillforts. 177 forts (4.27%) have been classified as multiple types. Hillforts can be more than one landscape type. For this reason, the total count is more than the total number of hillforts in the atlas.

In [ ]:
```
plot_bar_chart(landscape_type_data, 2, 'Landscape Type', 'Count', \
               'Landscape: Type')
```



In [ ]:
```
count_yes(landscape_type_data)
```

```
Landscape_Type_Contour: 1860
Landscape_Type_Partial: 384
Landscape_Type_Promontory: 1248
Landscape_Type_Hillslope: 427
Landscape_Type_Level: 226
Landscape_Type_Marsh: 23
Landscape_Type_Multiple: 177
Total yes count: 4345
```

There are two hillforts identifed as both contour and partial contour.

In [ ]:
```
contour_type = \
landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='Yes') \
                    &(landscape_type_data['Landscape_Type_Partial']=='Yes') \
                    &(landscape_type_data['Landscape_Type_Promontory']=='No') \
                    &(landscape_type_data['Landscape_Type_Hillslope']=='No') \
                    &(landscape_type_data['Landscape_Type_Level']=='No') \
                    &(landscape_type_data['Landscape_Type_Marsh']=='No') \
                    &(landscape_type_data['Landscape_Type_Multiple']=='No')]
print(f'Landscape Type yes for both Contour and Partial Contour: {len(contour_type)}')
```

```
Landscape Type yes for both Contour and Partial Contour: 2
```

There are 13 hillforts classified as both contour and promontory.

In [ ]:
```
contour_prom_type = \
landscape_type_data[(landscape_type_data['Landscape_Type_Contour']=='Yes') \
                    &(landscape_type_data['Landscape_Type_Partial']=='No') \
                    &(landscape_type_data['Landscape_Type_Promontory']=='Yes') \
                    &(landscape_type_data['Landscape_Type_Hillslope']=='No') \
                    &(landscape_type_data['Landscape_Type_Level']=='No') \
                    &(landscape_type_data['Landscape_Type_Marsh']=='No') \
                    &(landscape_type_data['Landscape_Type_Multiple']=='No')]
print(f'Landscape Type yes for both Contour and Promontory: {len(contour_prom_type)}')
```

```
Landscape Type yes for both Contour and Promontory: 13
```

Landscape Type Data Mapped

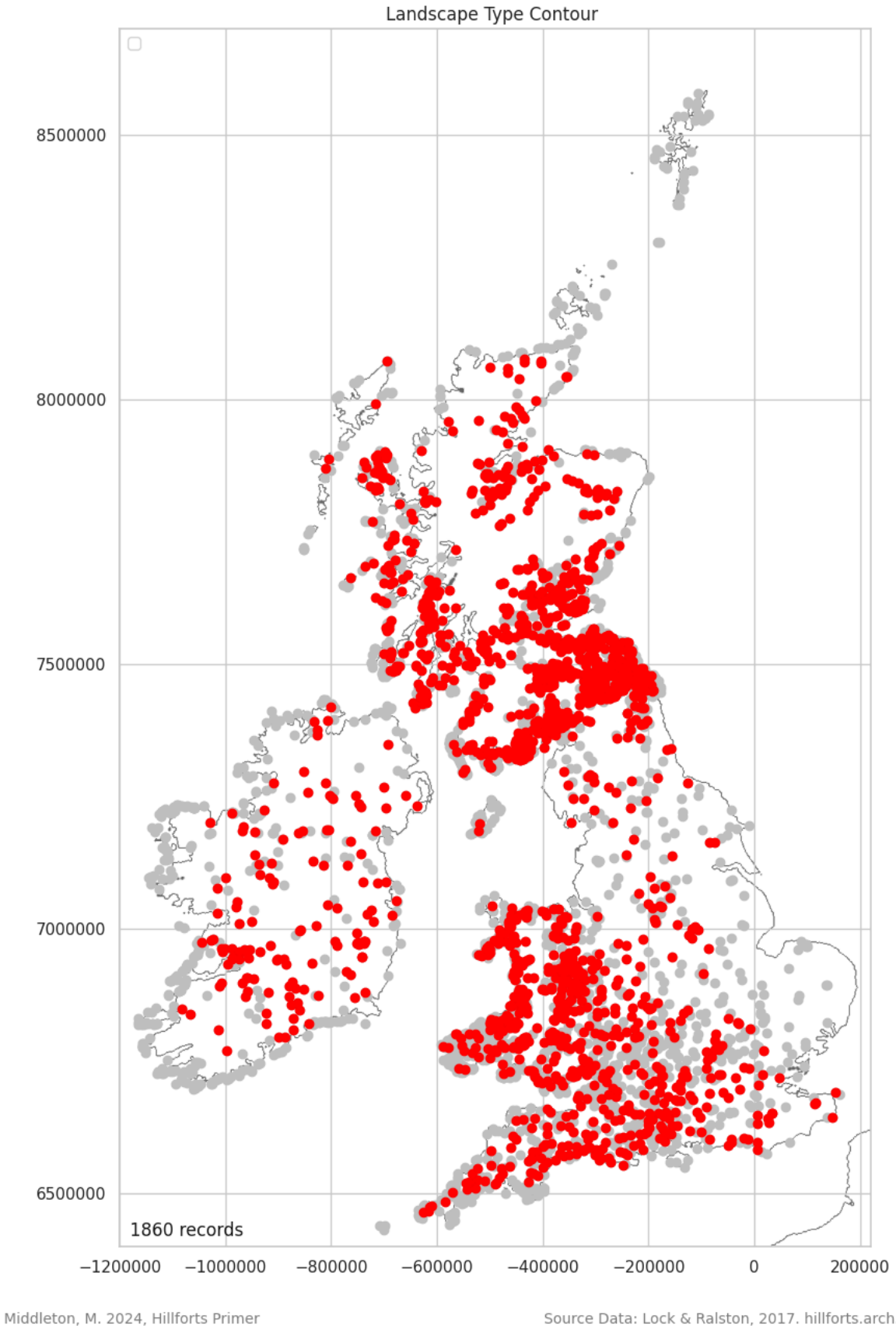
The Landscape Type data is recombined with the Location data so it can be mapped.

In [ ]:
```
location_landscape_data = pd.merge(location_numeric_data_short, \
                                   landscape_type_data, left_index=True, \
                                   right_index=True)
```

Contour mapped

Contour forts are common across all of mainland Scotland, mainland Ireland, Wales and the South West of England. They are noticeably absent from the areas without significant hills, such as most coastlines, the northern Isles and the east of England.

In [ ]:
```
tp_contour = plot_over_grey(location_landscape_data, \
                            'Landscape_Type_Contour', 'Yes')
```

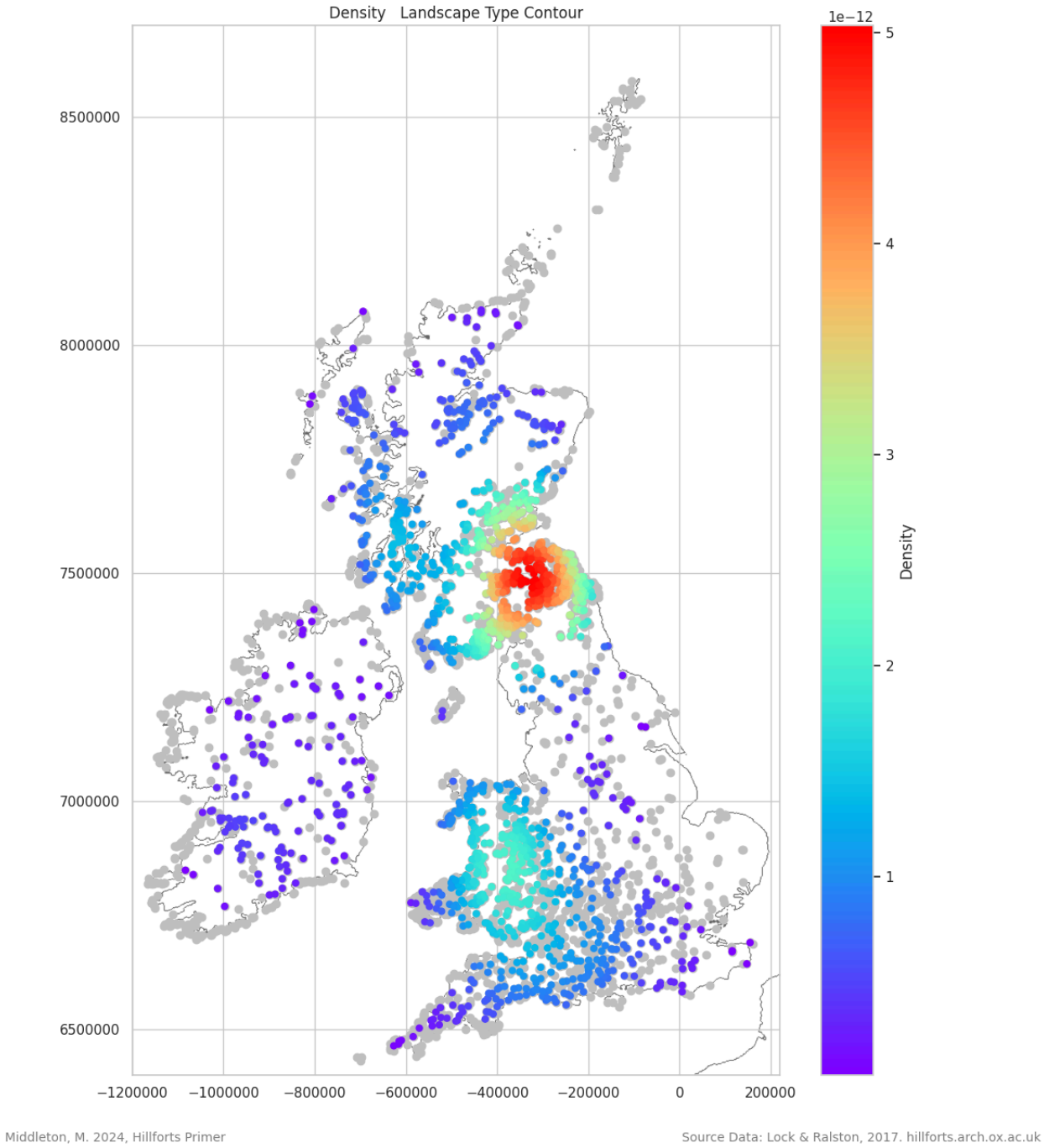


44.85%

Contour Density mapped

The density of contour hillforts matches the southern Cambrian Mountains and eastern Southern Uplands density clusters seen in Part1 (Density Map Showing Clusters Adjusted by Region). In contrast, the clusters seen in the Irish data are not replicated.

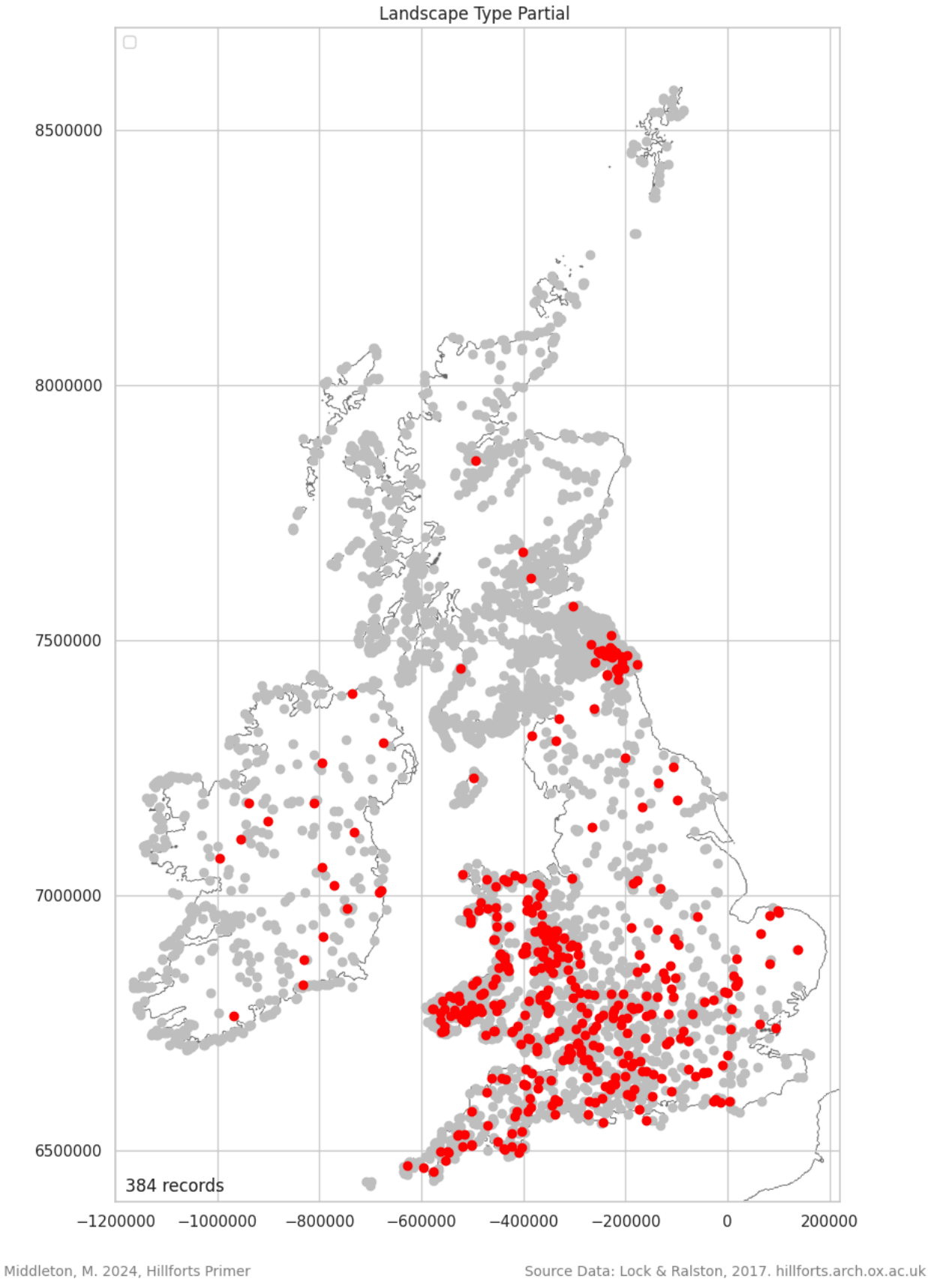
In [ ]:
```
plot_density_over_grey(tp_contour, 'Landscape_Type_Contour')
```



Partial Mapped

There is a bias in that partial contour hillforts is a classification type use mostly in England and Wales. There are a couple of hillforts of this type recorded in Scotland and Ireland, but very few. Combining, 'Landscape_Type_Contour' and 'Landscape_Type_Partial' may minimise this bias.

In [ ]:
```
tp_partial = plot_over_grey(location_landscape_data, \
                            'Landscape_Type_Partial', 'Yes')
```

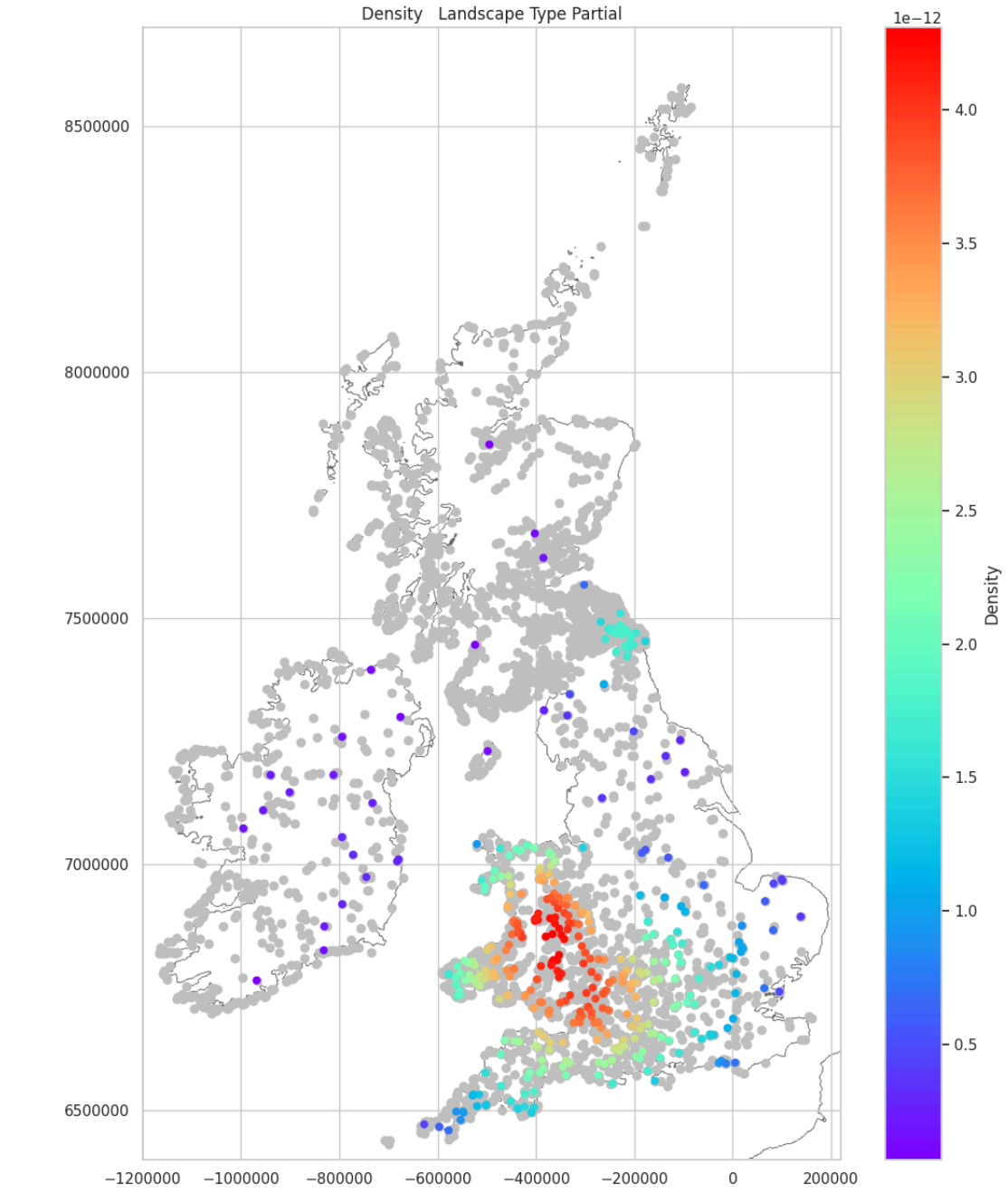


9.26%

Partial Density Mapped

Due to the bias in this data the density plot shows a single strong cluster in the south, over the Cambrian Mountains and into south central England. It is important to be aware of the bias in this data and to compare this density plot with, 'Landscape_Type_Contour', which shows a more meaningful, atlas wide distribution.

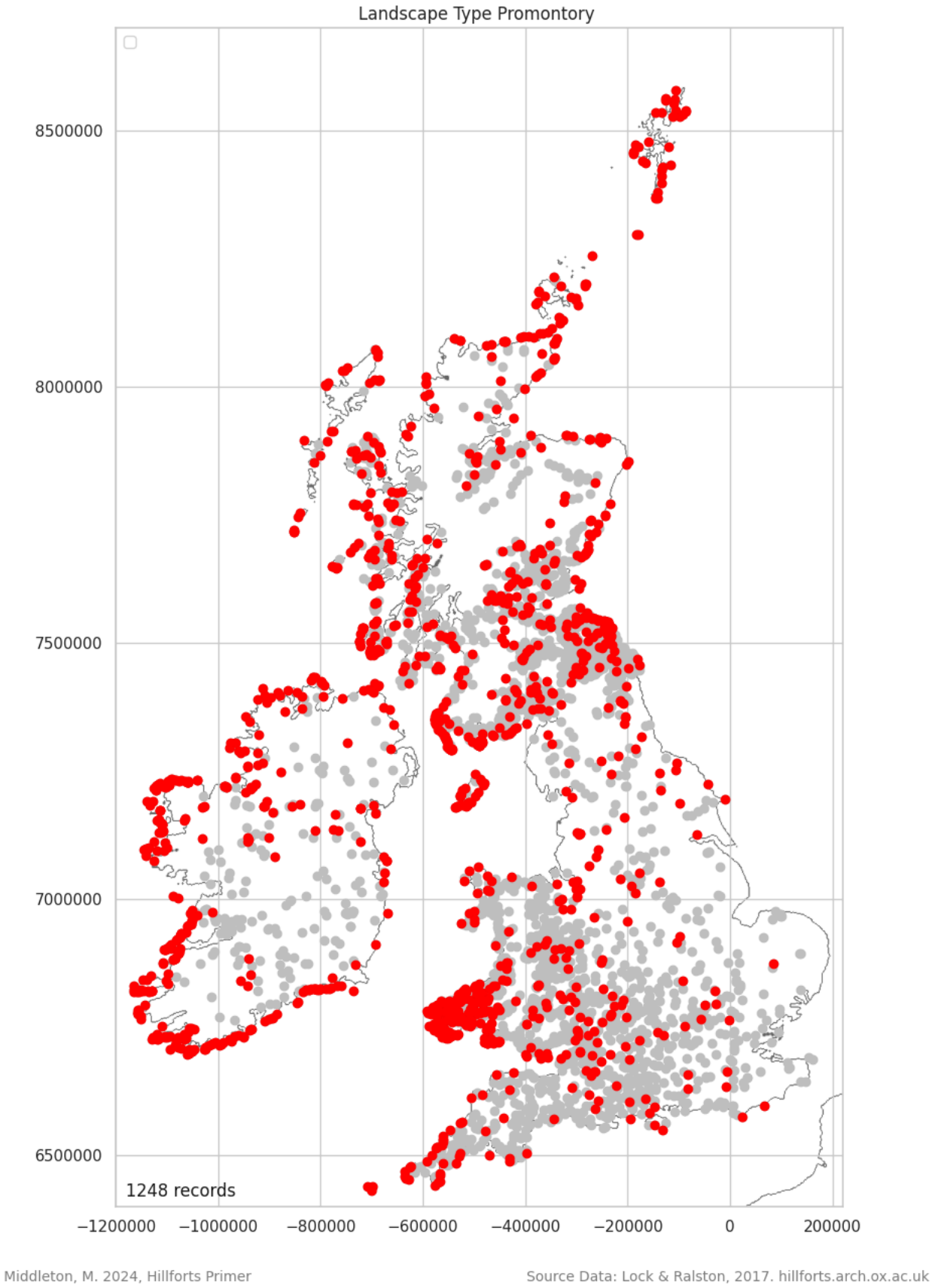
In [ ]:
```
plot_density_over_grey(tp_partial, 'Landscape_Type_Partial')
```



Promontory Mapped

Promontory forts have a strong correlation to the coast, although promontory forts do also present inland.

In [ ]:
```
tp_promontory = plot_over_grey(location_landscape_data, \
                               'Landscape_Type_Promontory', 'Yes')
```



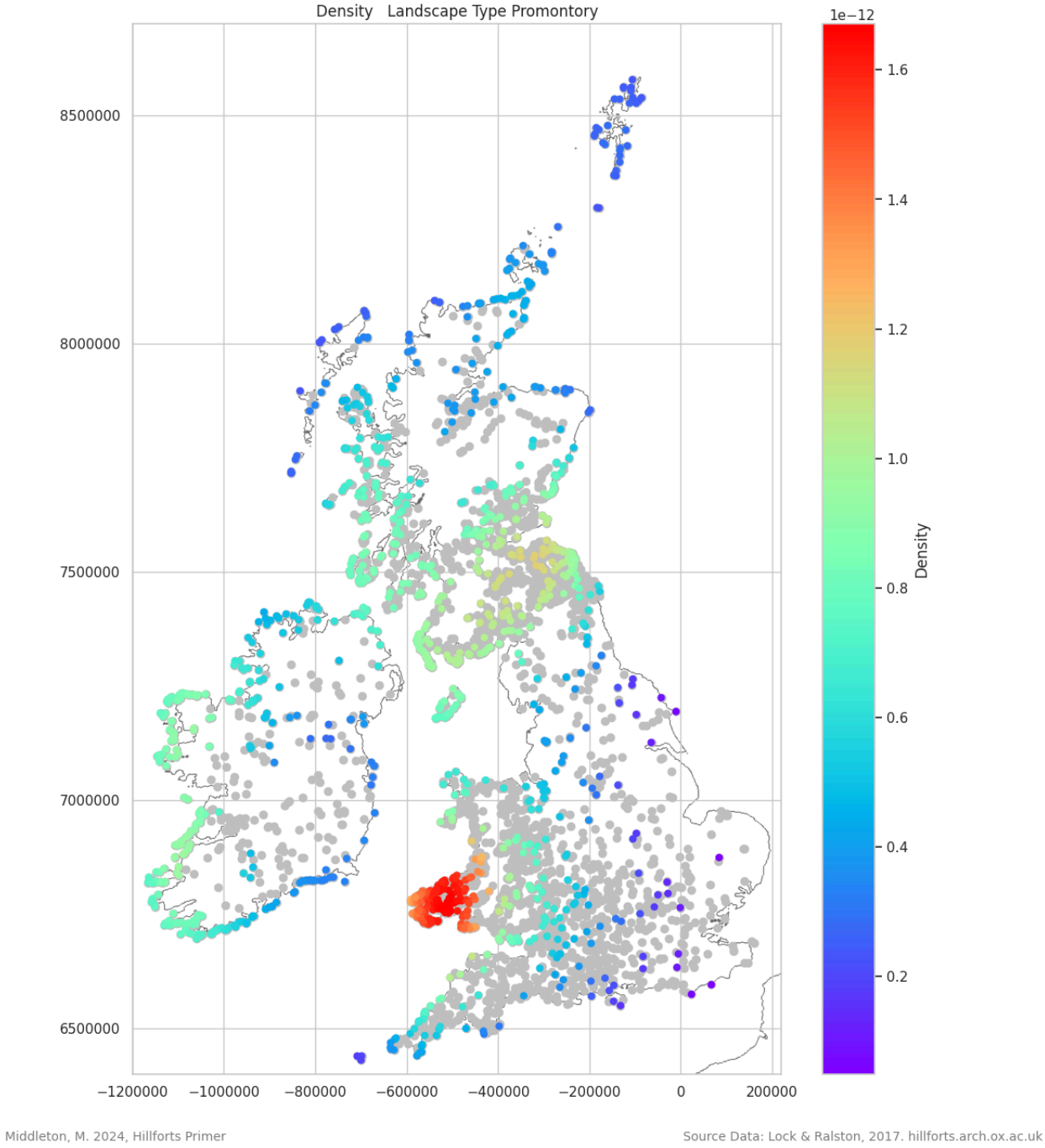
30.09%

Promontory Density Mapped

The highest concentration of promontory forts is along the Pembrokeshire peninsula. The clusters seen in the general density plots over the Irish west coast, also reveal themselves to be promontory forts. Similarly, there are concentrations of promontory forts along the west coast of Scotland and to the east of the Southern Uplands. See: Part1 (Density Map Showing Clusters Adjusted by Region).

Ref: 3. Coastal Promontory Forts — Cliff Castles https://intarch.ac.uk/journal/issue48/5/1.html#3

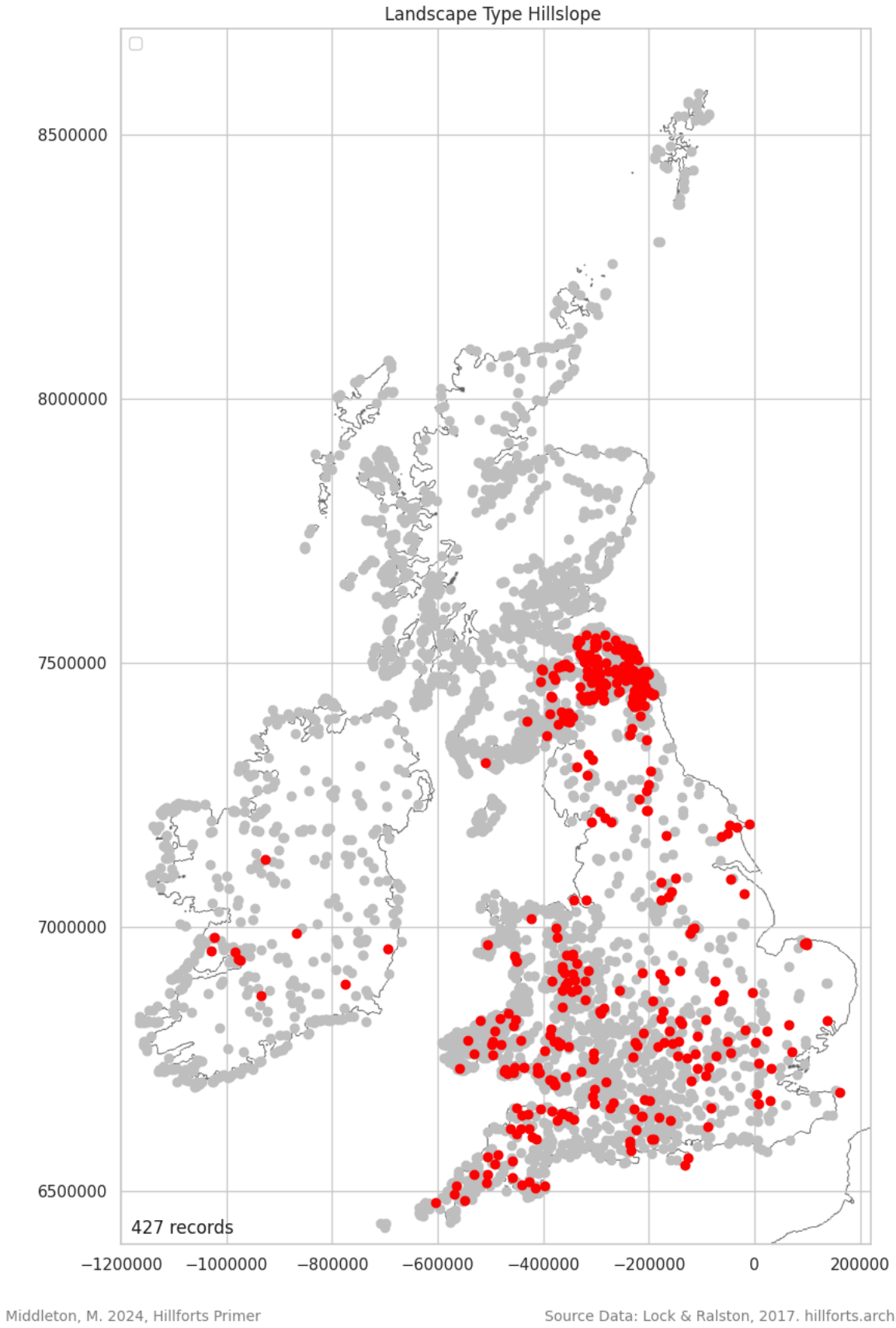
In [ ]:
```
plot_density_over_grey(tp_promontory, 'Landscape_Type_Promontory')
```



Hillslope Mapped (Landscape)

It looks like there is a bias in this data. There are no hillslope hillforts in Scotland outside the Southern Uplands and there are very few in Ireland. This is either a recording bias across southern Scotland or a remarkably concentrated distribution of this type. This seems unlikely. A recording bias seems the most likely.

In [ ]:
```
tp_hillslope = plot_over_grey(location_landscape_data, \
                              'Landscape_Type_Hillslope', 'Yes')
```

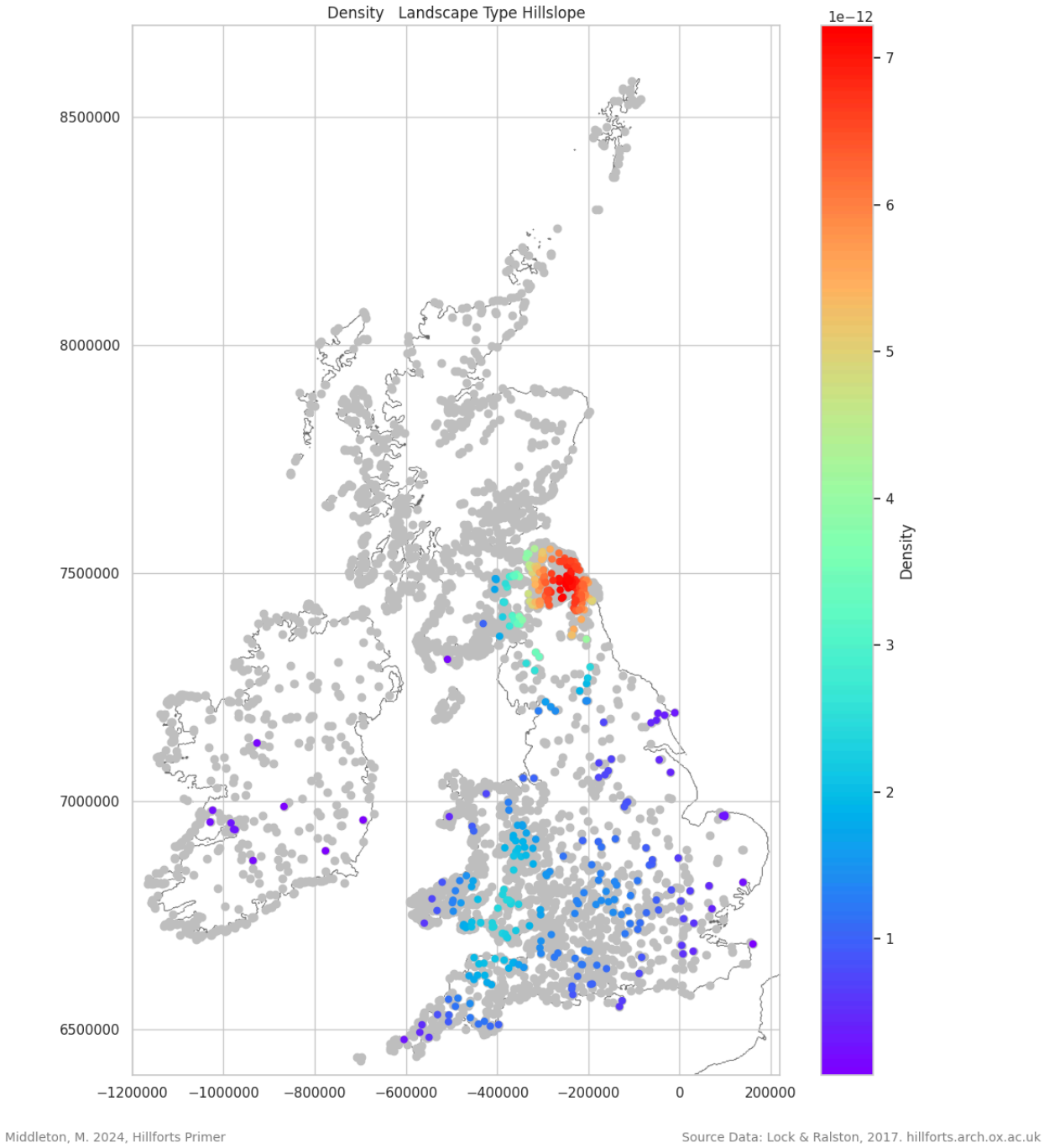



```
10.3%
```

Hillslope Density Mapped (Landscape)

Although caution should be taken with this data due to the probable bias, the hillslope data shows a strong concentration to the east of the Southern Uplands and another, weak concentration, over the Brecon Beacons and Exmoor.

In [ ]:
```
plot_density_over_grey(tp_hillslope, 'Landscape_Type_Hillslope')
```



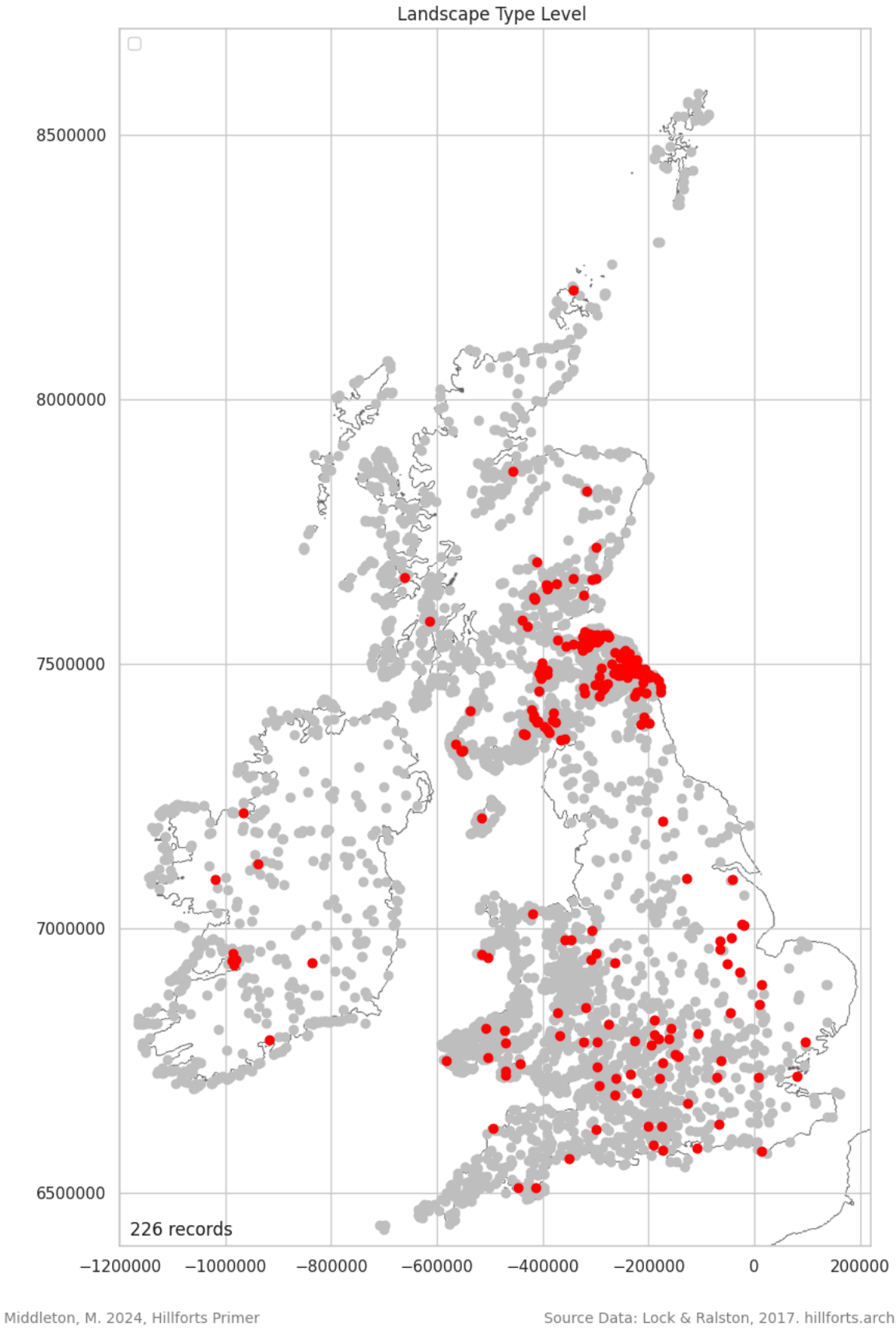
Level Mapped (Landscape)

There is a bias in that 74.83% of 'Level' hillforts, in the north, are known because of intensive cropmark survey carried out in southern Scotland. This can be seen in the distribution of 'Level' hillforts within the Tweed Valley and around the lowland fringes of the eastern Southern Uplands. There are a peppering of this type across the remainder of Scotland, Ireland and southern England.

See: Cropmark Mapped

See: Aspect Level Mapped

In [ ]:
```
tp_level = plot_over_grey(location_landscape_data, \
                          'Landscape_Type_Level', 'Yes')
```



5.45%

In [ ]:
```
split_location = 7070000
north = hillforts_data[hillforts_data['Location_Y']>=split_location]
level_n = north[(north['Landscape_Type_Level']=='Yes')]
print(f'{len(level_n)} of the level hillforts are in the north.')
```

151 of the level hillforts are in the north.

In [ ]:
```
level_cropmark_n = \
level_n[level_n['Management_Condition_Cropmark']=='Yes']
print(f'{len(level_cropmark_n)} ({round((len(level_cropmark_n)/len(level_n))*100,2)}%) of the level hillforts in the
```

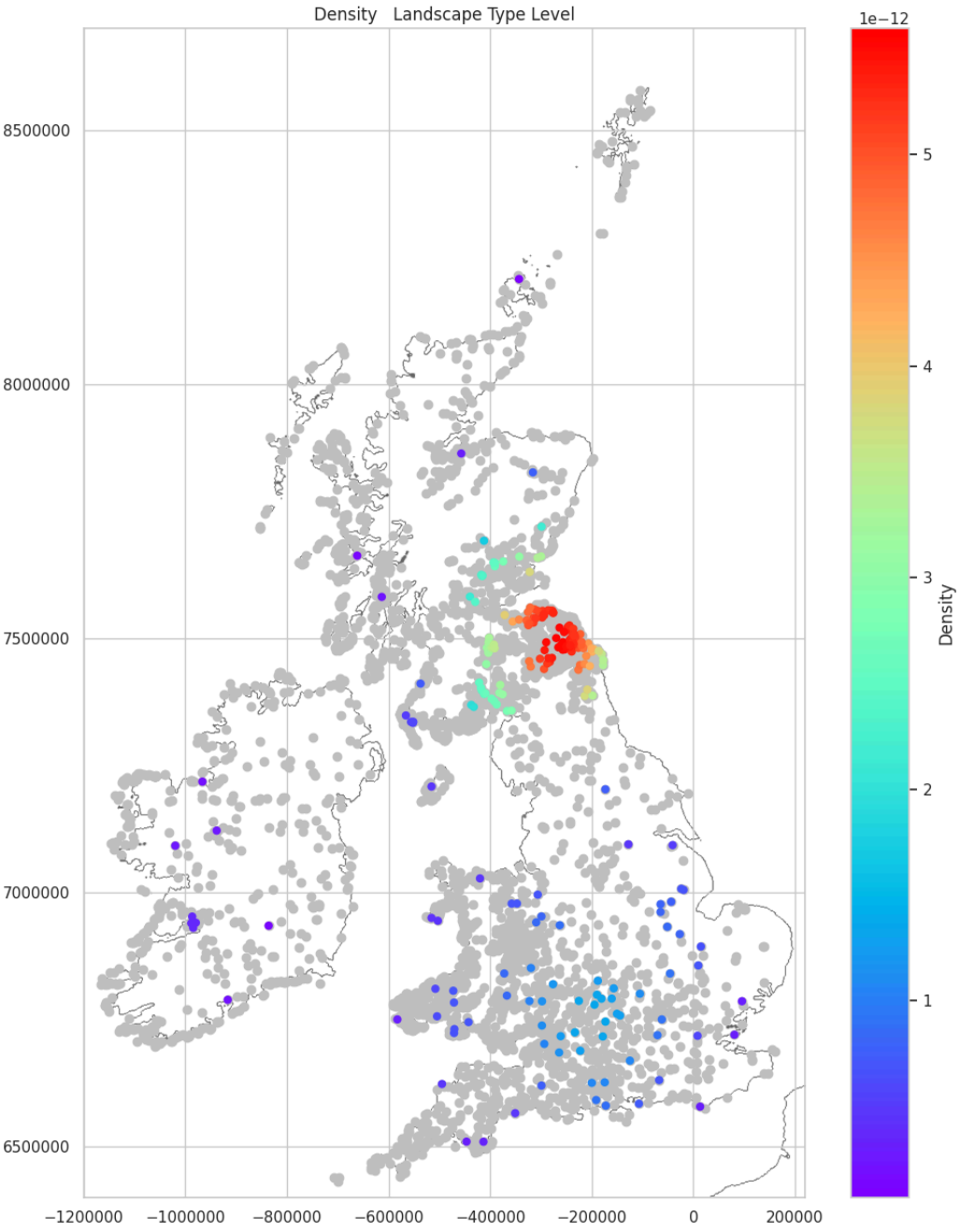
```
#print(f'{len(level_cropmark_n)} ({round((len(level_cropmark_n)/len(level_n))*
#100,2)}%) of the level hillforts in the north are cropmark sites.')
```

```
113 (74.83%) of the level hillforts in the north are cropmark sites.
```

Level Density Mapped

Because of the bias created by the intensive aerial survey work over the Tweed Basin, the northern cluster is exaggerated. There is a second, very slight concentration of level forts between the Cotswolds and the Shropshire Hills.

In [ ]:
```
plot_density_over_grey(tp_level, 'Landscape_Type_Level')
```



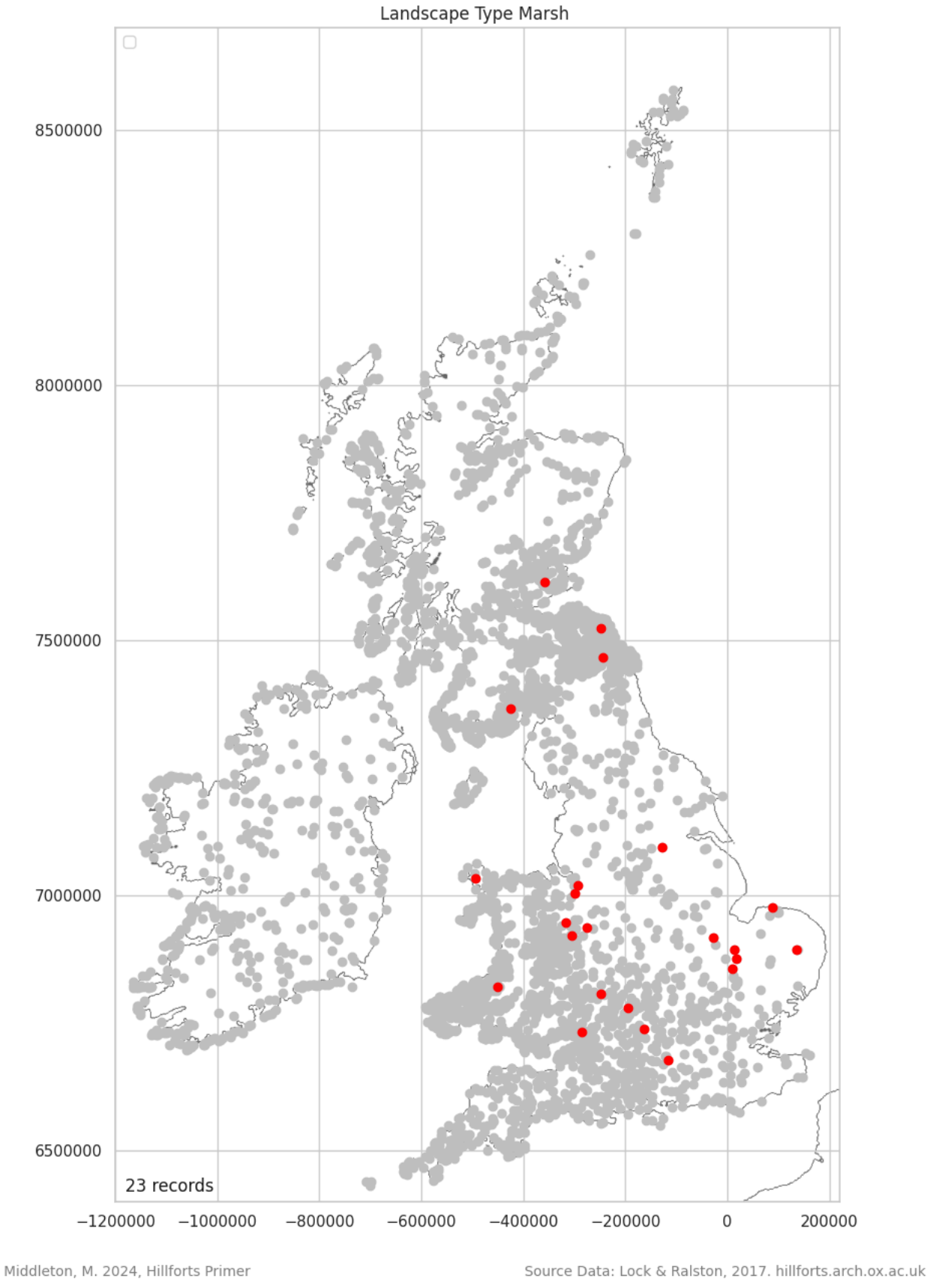
Middleton, M. 2024, Hillforts Primer

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

Marsh Mapped

There are only 23 hillforts identified as type 'Marsh'. A density map has not been produced as a map based on such a small sample will be misleading.

In [ ]:
```
tp_marsh = plot_over_grey(location_landscape_data, \
                          'Landscape_Type_Marsh', 'Yes')
```

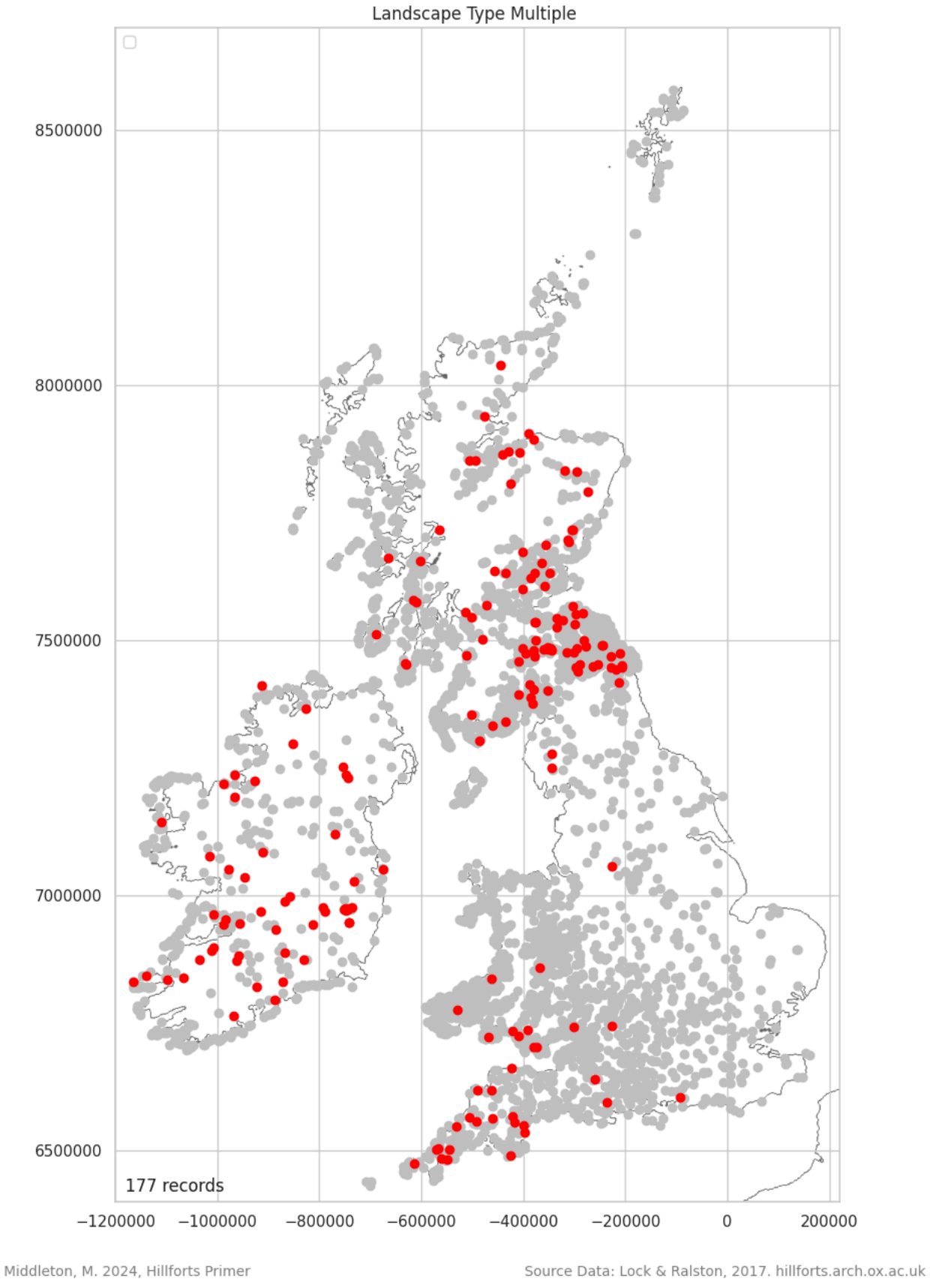


0.55%

Multiple Mapped

There is a recording bias in that the 'Multiple' Hillforts type has not been recorded uniformly. There are very few records in England and Wales, with almost none in the north, east and south England and the north of Wales.

In [ ]:
```
tp_multiple = plot_over_grey(location_landscape_data, \
                             'Landscape_Type_Multiple', 'Yes')
```

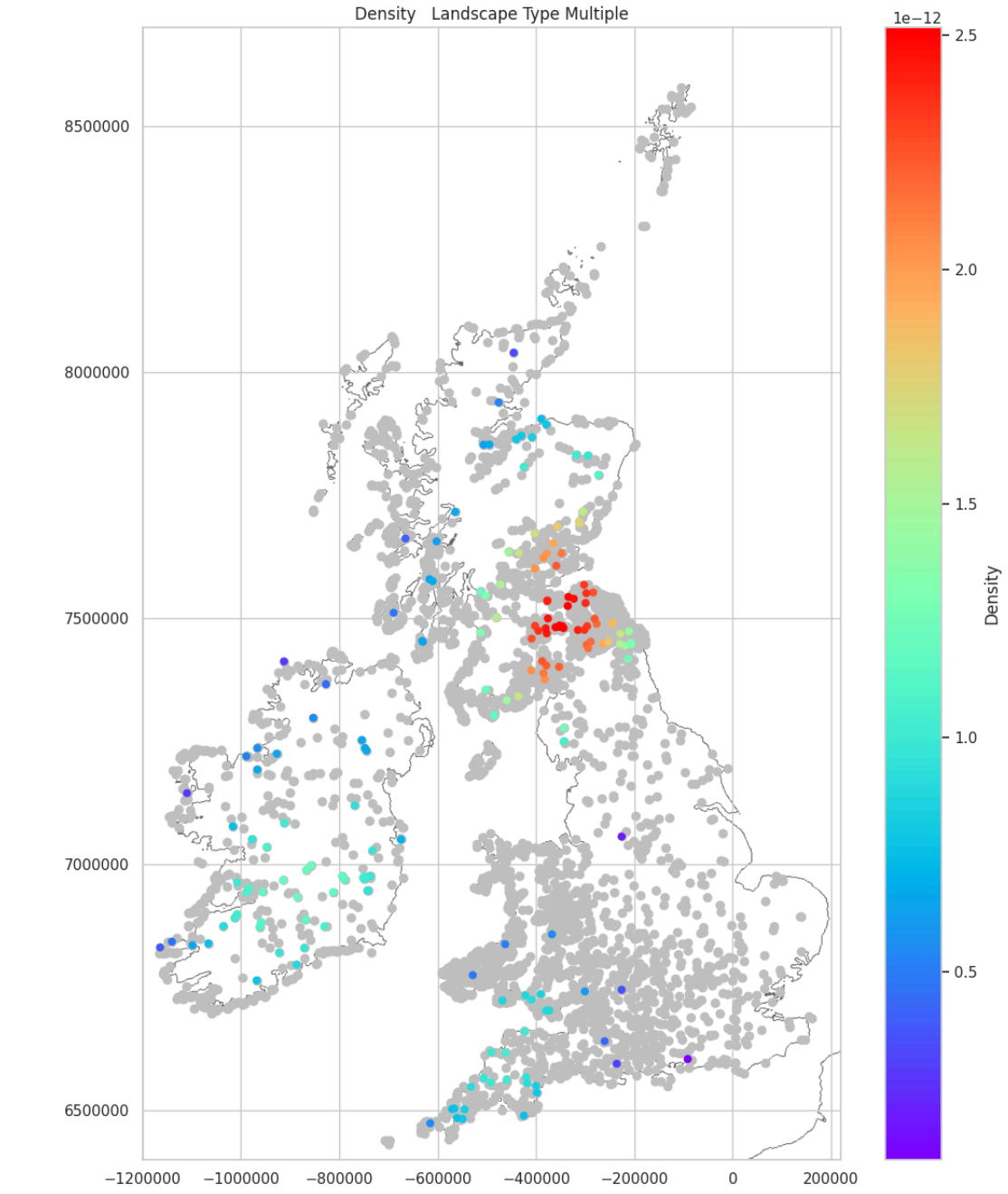


4.27%

Multiple Density Mapped

The bias in this data means the clusters in this density plot are not reliable. Where there is data, the eastern Southern Uplands is the main cluster and central Ireland shows as a slight, secondary concentration.

In [ ]:
```
plot_density_over_grey(tp_multiple, 'Landscape_Type_Multiple')
```



Topography Data

The topography data consists of 10 topographic types. Multiple types can be assigned to a single hillfort.

In [ ]:
```
landscape_topography_features = [
 'Landscape_Topography_Hilltop',
 'Landscape_Topography_Coastal',
 'Landscape_Topography_Inland',
 'Landscape_Topography_Valley',
 'Landscape_Topography_Knoll',
 'Landscape_Topography_Ridge',
 'Landscape_Topography_Scarp',
 'Landscape_Topography_Hillslope',
 'Landscape_Topography_Lowland',
 'Landscape_Topography_Spur']

landscape_topography_data = \
landscape_encodeable_data[landscape_topography_features].copy()
landscape_topography_data.head()
```

Out[ ]:

	Landscape_Topography_Hilltop	Landscape_Topography_Coastal	Landscape_Topography_Inland	Landscape_Topography_Valley	Landscape_T
0	Yes	No	Yes	No	
1	Yes	No	No	No	
2	Yes	No	No	No	
3	Yes	No	No	No	
4	Yes	No	No	No	

The topography data contains no null values.

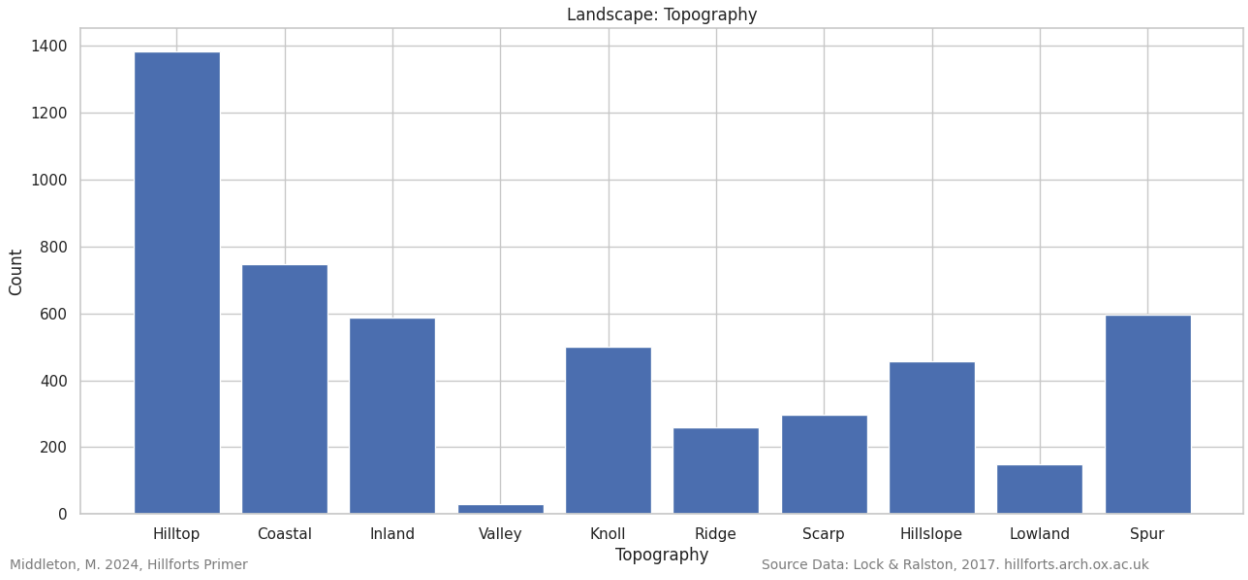
In [ ]:
```
landscape_topography_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Landscape_Topography_Hilltop    4147 non-null   object
 1   Landscape_Topography_Coastal    4147 non-null   object
 2   Landscape_Topography_Inland     4147 non-null   object
 3   Landscape_Topography_Valley     4147 non-null   object
 4   Landscape_Topography_Knoll      4147 non-null   object
 5   Landscape_Topography_Ridge      4147 non-null   object
 6   Landscape_Topography_Scarp      4147 non-null   object
 7   Landscape_Topography_Hillslope  4147 non-null   object
 8   Landscape_Topography_Lowland    4147 non-null   object
 9   Landscape_Topography_Spur       4147 non-null   object
dtypes: object(10)
memory usage: 324.1+ KB
```

Topography Data Plotted

Unsurprisingly, most of the topographic types relate to upland features such as hill tops, ridges, spurs and knolls. Very few hillforts are identified as being in lowland areas or valleys. Coastal, inland and lowland types are notable as these are not topographic features but rather, areas which are defined with reference to the topography. A number of the terms look to have a regional bias.

In [ ]:
```
plot_bar_chart(landscape_topography_data, 2, 'Topography', \
               'Count', 'Landscape: Topography')
```



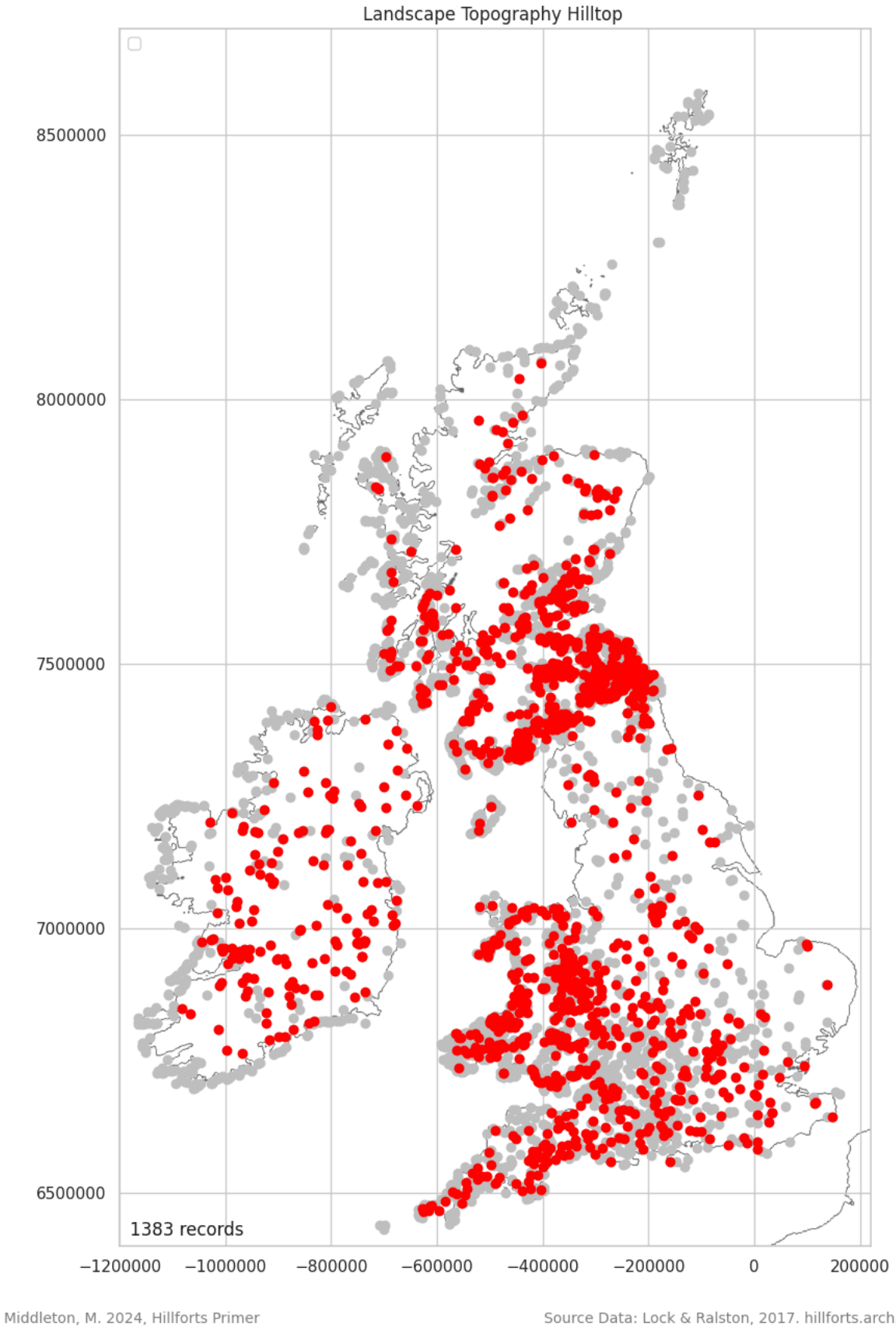
Topography Data Mapped

In [ ]:
```
location_topo_data = pd.merge(location_numeric_data_short, \
                              landscape_topography_data, left_index=True, \
                              right_index=True)
```

Hilltop Mapped

One third (1283) of all hillforts are recorded as being located on a hilltop.

In [ ]:
```
topo_hilltop = plot_over_grey(location_topo_data, \
                              'Landscape_Topography_Hilltop', 'Yes')
```

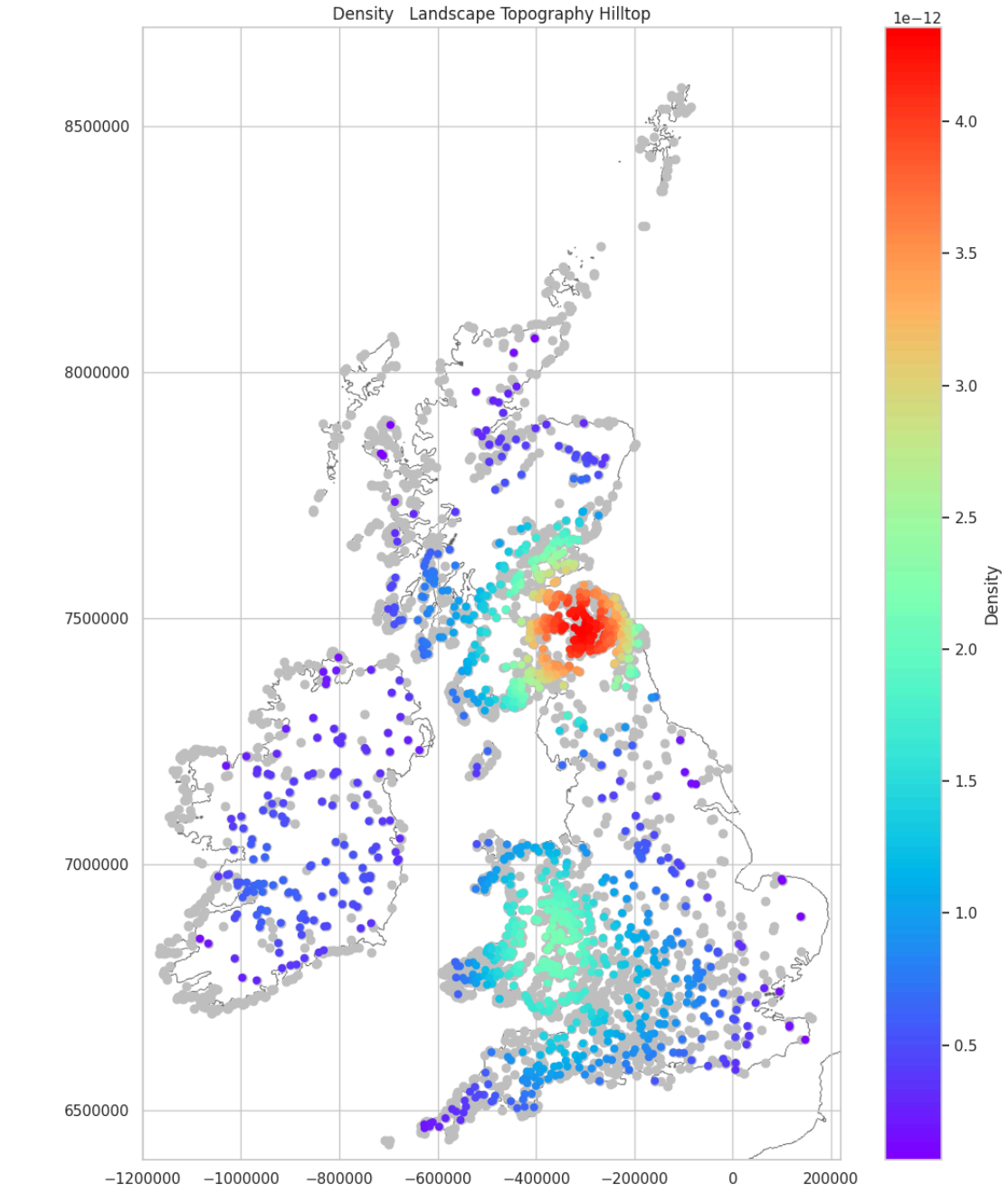


33.35%

Hilltop Density Mapped

Hilltop densities have two main clusters which correspond with the main hillfort density concentrations seen over the Southern Uplands and the Cambrian Mountains. The clusters in the Northwest and the west of Ireland are not present. See Part 1: Density Map Showing Clusters Adjusted by Region.

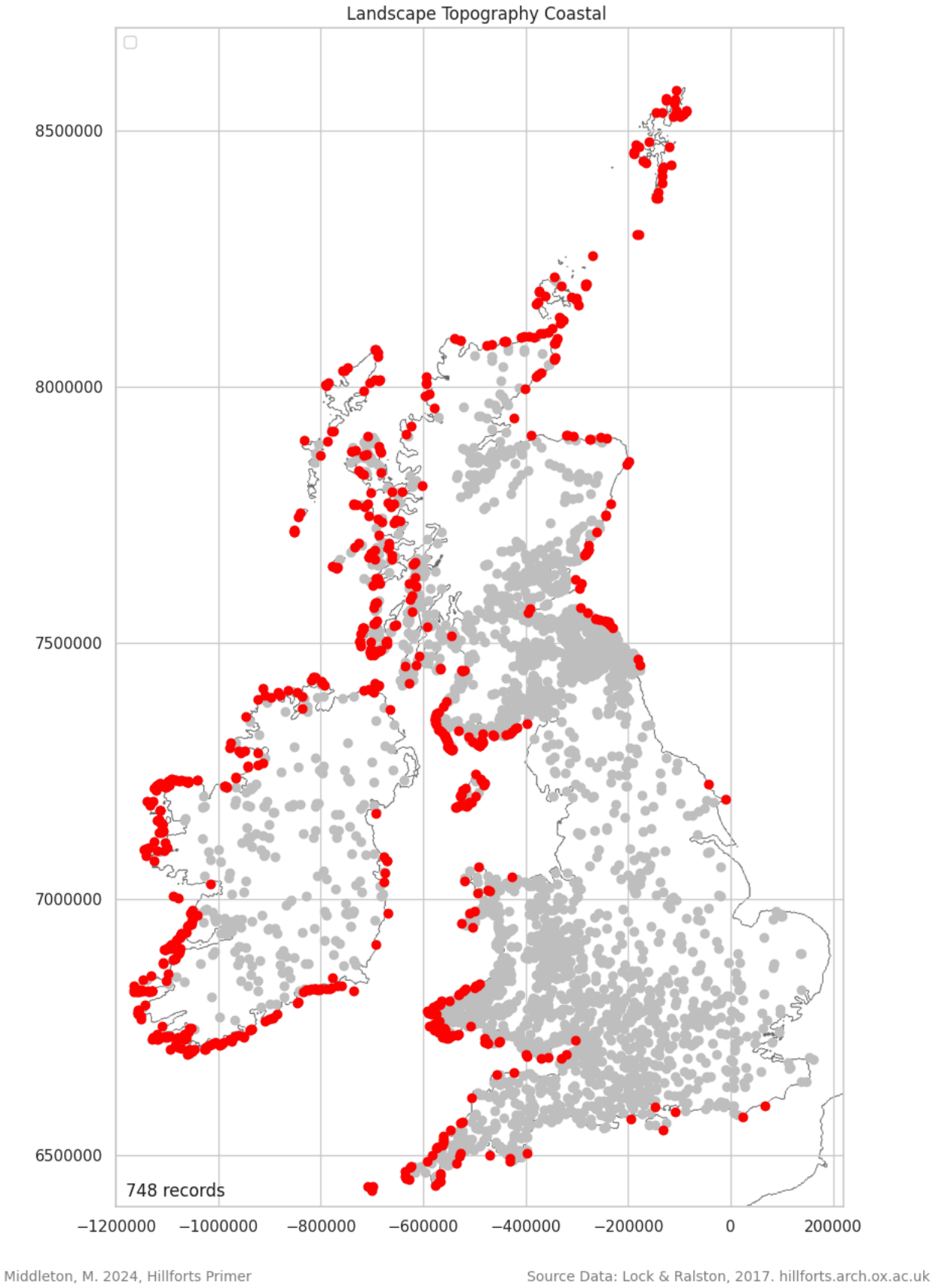
In [ ]:
```
plot_density_over_grey(topo_hilltop, 'Landscape_Topography_Hilltop')
```



Coastal Mapped (Topo)

748 hillforts (18.04%) are identified as coastal, with the west coasts of the mainland and Ireland being the focus for most. There are very few coastal forts along the east and south coasts of England. Coastal erosion, especially in the southeast, is likely to be at least partly responsible for this. There is a notable gap in the distribution of hillforts along the west coast of England between the Scottish and Welsh borders. See: Coastal Mapped (Land Use).

In [ ]:
```
topo_coastal = plot_over_grey(location_topo_data, \
                              'Landscape_Topography_Coastal', 'Yes')
```

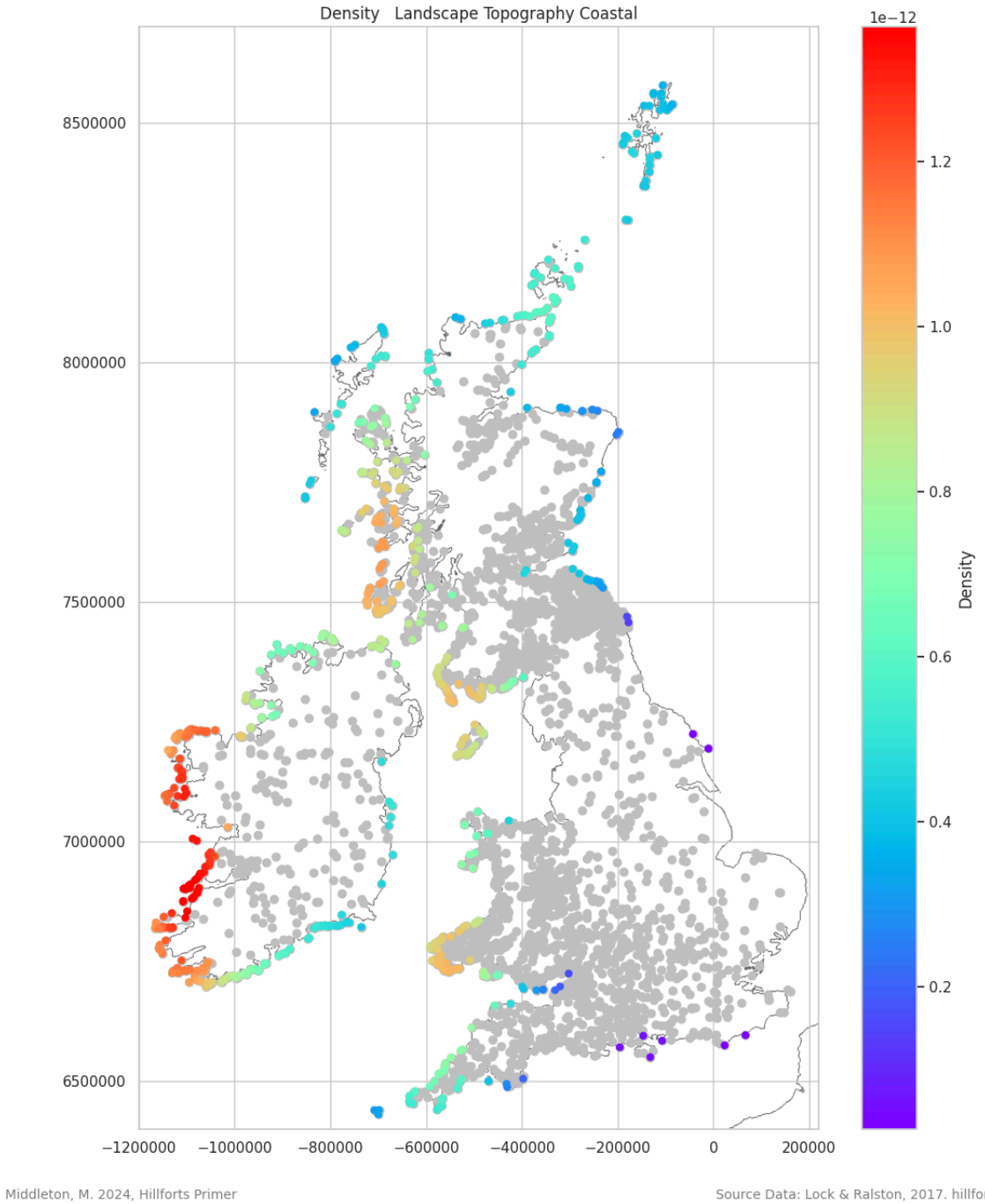



```
18.04%
```

Coastal Density Mapped (Topography)

The strongest concentration of coastal forts is along the west coast of Ireland. There are smaller concentrations on the Pembrokeshire coast; around Luce Bay (near Whithorn and including the Isle of Man) and along the line of islands from Islay to southern Skye. See: Coastal Density Mapped (Land Use).

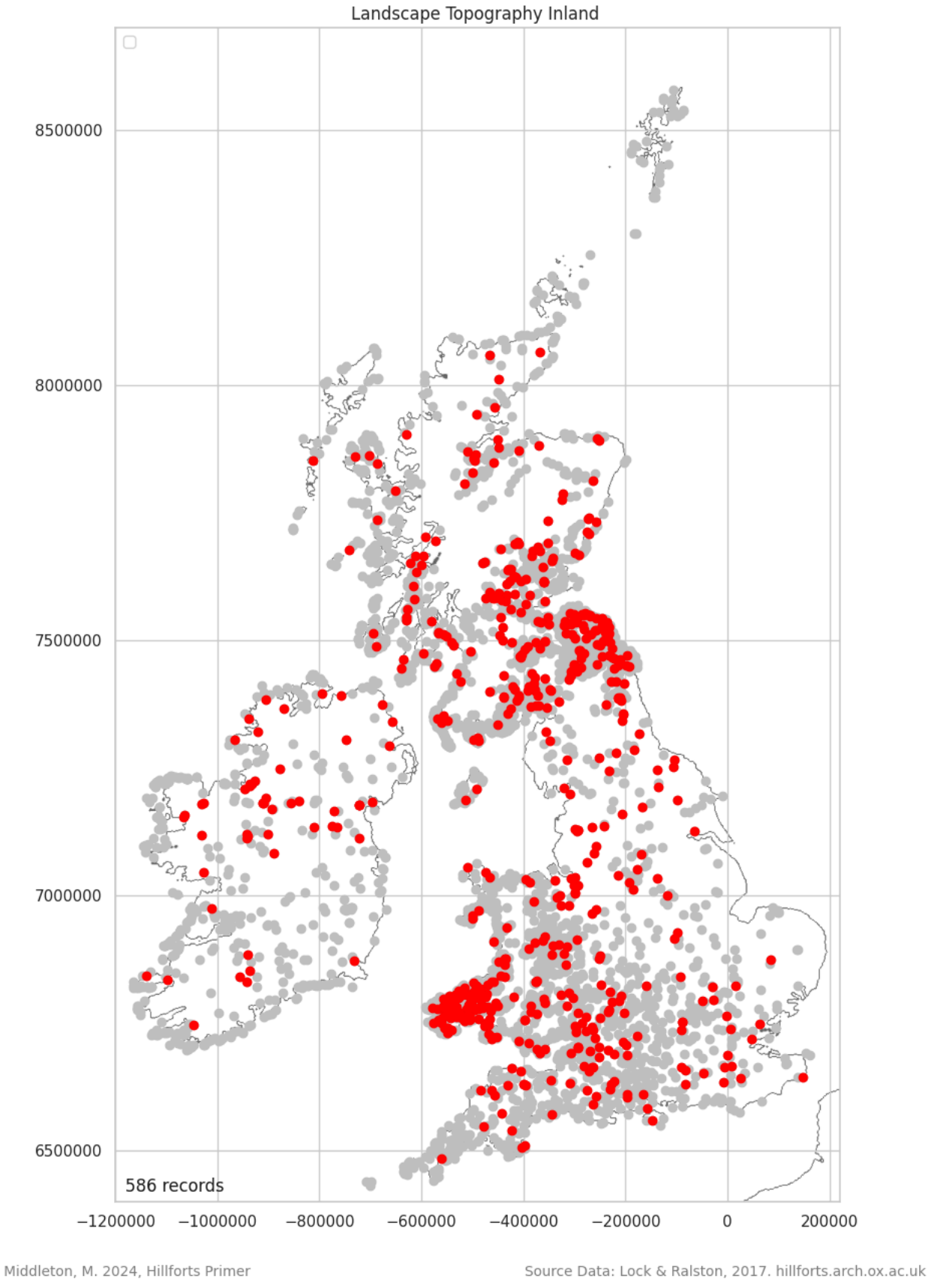
In [ ]:
```
plot_density_over_grey(topo_coastal, 'Landscape_Topography_Coastal')
```



Inland Mapped

It is unclear why some hillforts are classified as inland and why others are not.

In [ ]:
```
topo_inland = plot_over_grey(location_topo_data, \
                             'Landscape_Topography_Inland', 'Yes')
```

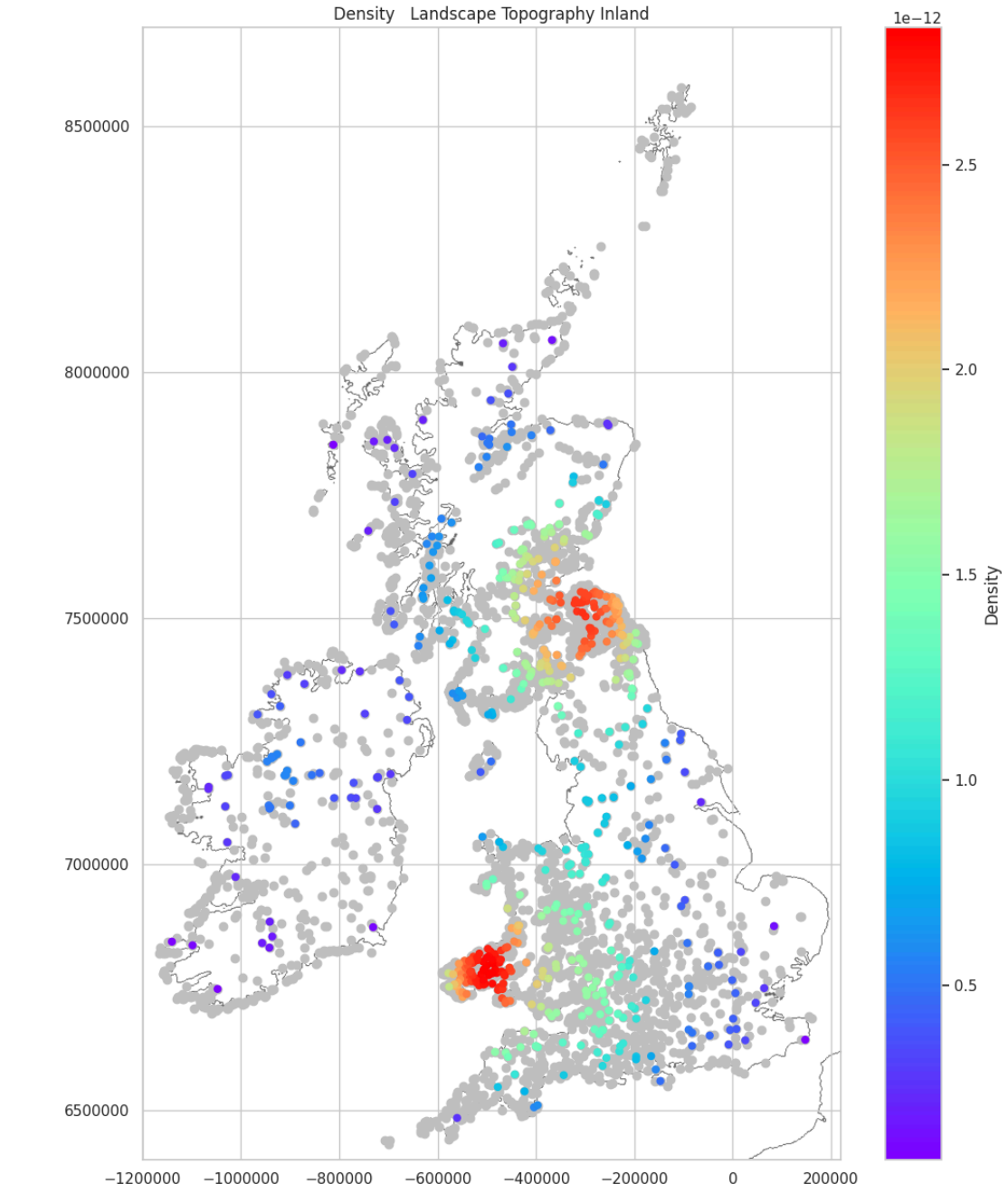


14.13%

Inland Density Mapped

The inland density contains two main clusters. These mirror the two main clusters seen in the main atlas distribution (see: Part 1: Density Map Showing Clusters Adjusted by Region). The Welsh concentration is interestingly focussed slightly further west, with the cluster being located over the Preseli Hills and the Pembrokeshire peninsula.

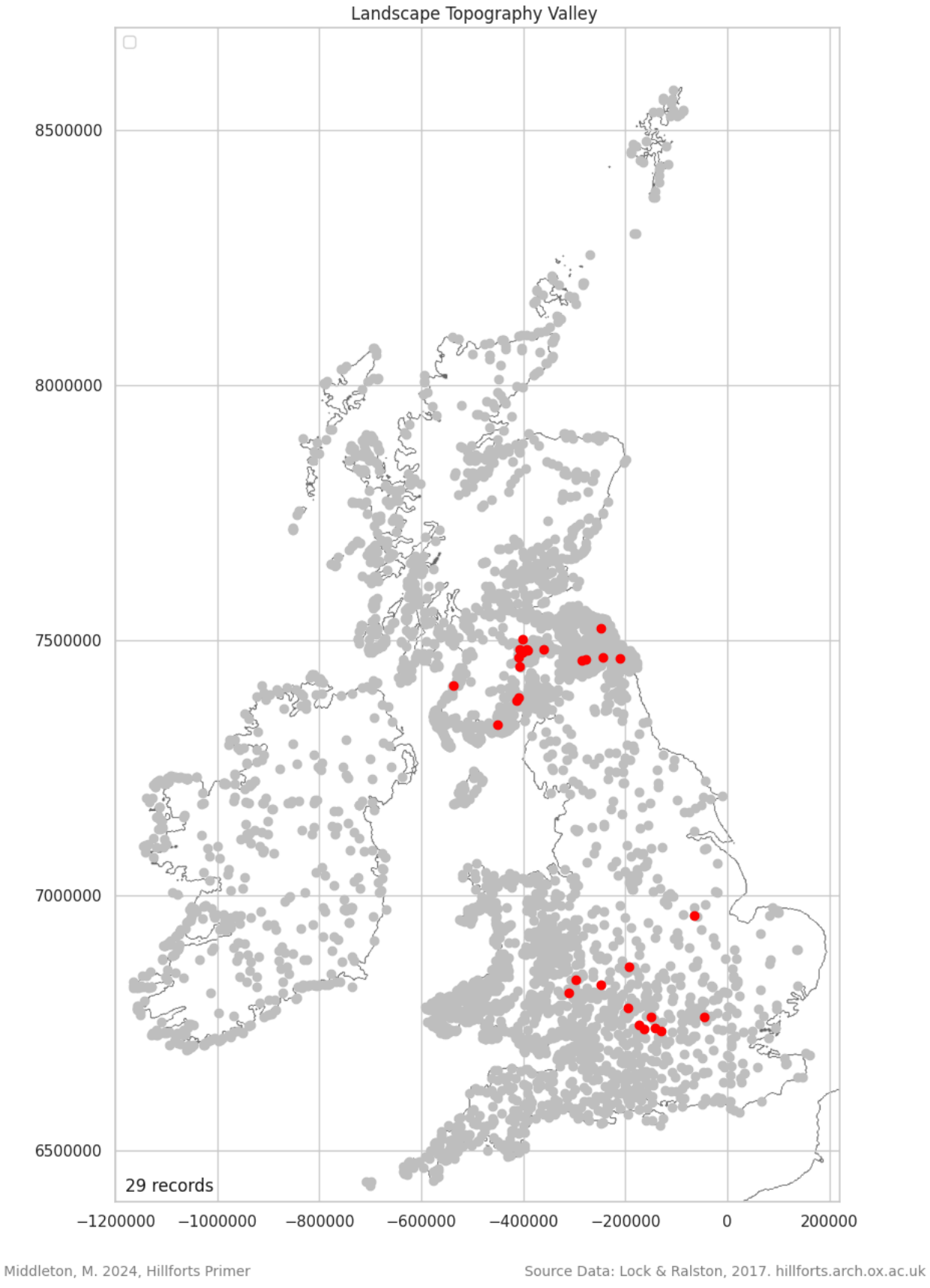
In [ ]:
```
plot_density_over_grey(topo_inland, 'Landscape_Topography_Inland')
```



Valley Mapped

Only 29 hillforts are identified as located in a 'valley'.

In [ ]:
```
topo_valley = plot_over_grey(location_topo_data, \
                             'Landscape_Topography_Valley', 'Yes')
```

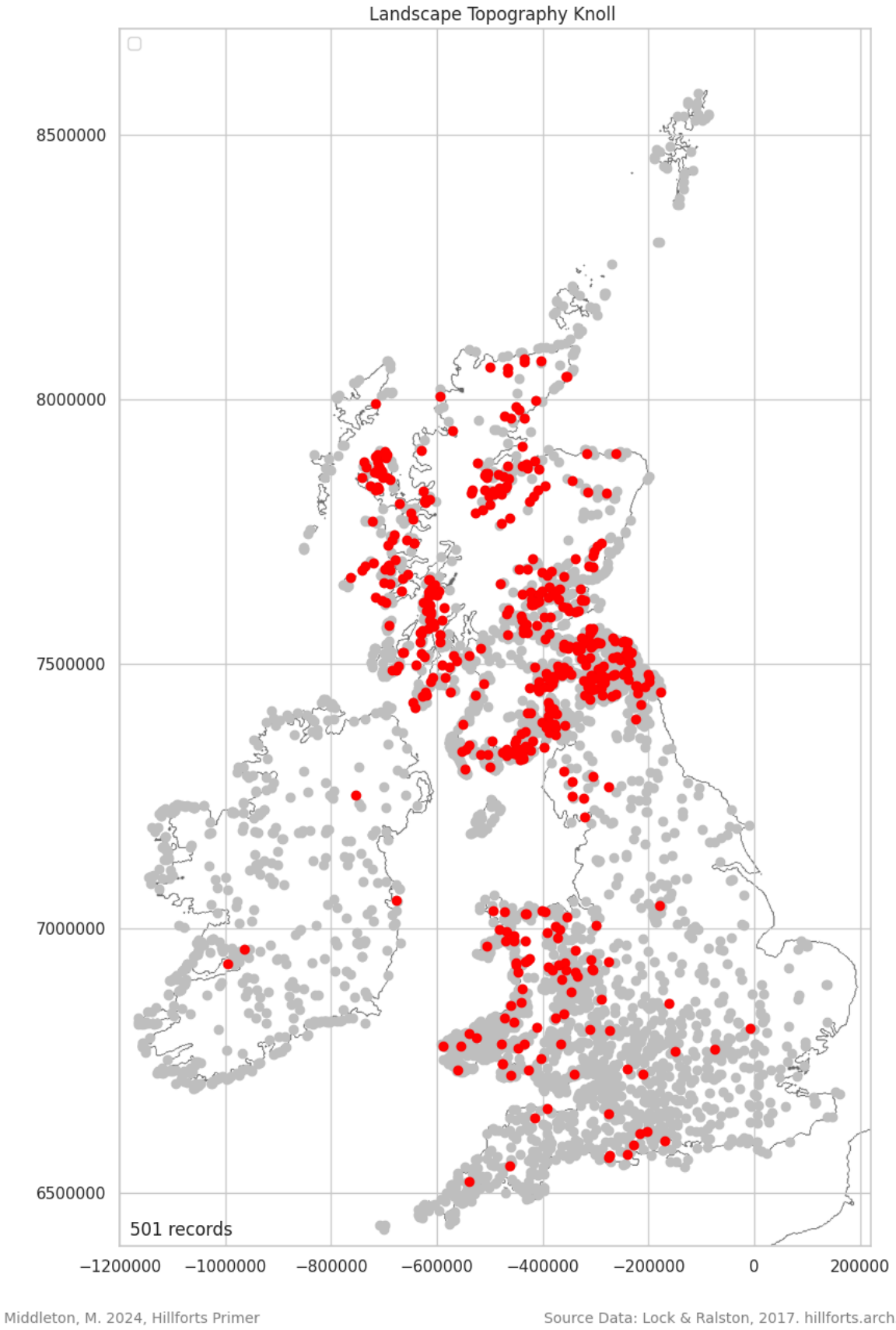


0.7%

Knoll Mapped

Hillforts identified as being located on a 'knoll' are almost exclusively in the upland areas of Scotland and Wales. The term is hardly used in Ireland and is far more common in Scotland. This may indicate a regional bias in where it is used.

In [ ]:
```
topo_knoll = plot_over_grey(location_topo_data, \
                            'Landscape_Topography_Knoll', 'Yes')
```

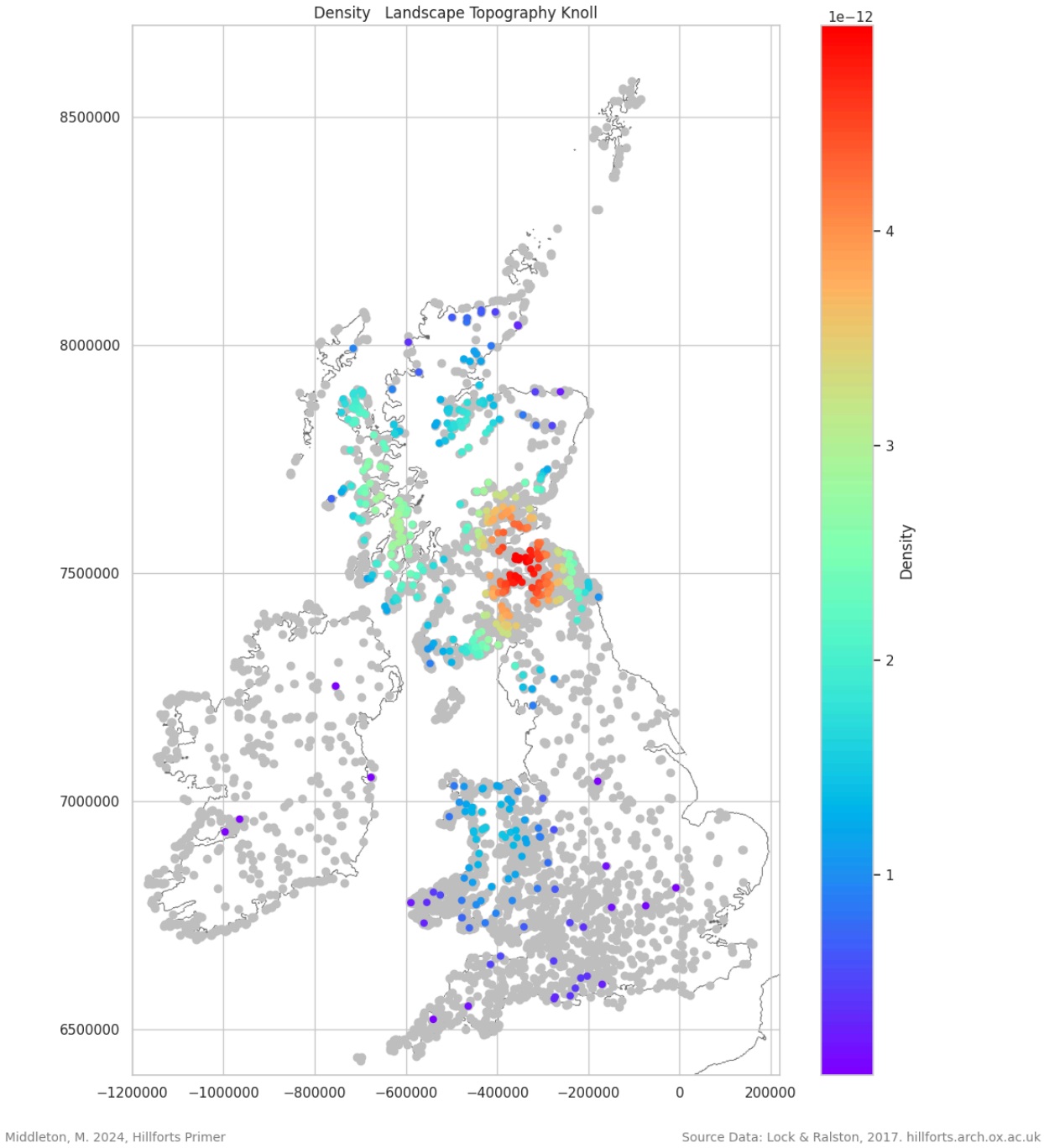


12.08%

Knoll Density Mapped

The main concentration of hillforts located on a 'knoll' is along the northern edge of the Southern Uplands, particularly across the Pentlands and Lammermuir hills, and along the west coast of Scotland, around the sea loch, Loch Linnhe. In Wales the focus is toward the northern end of the Cambrian Mountains.

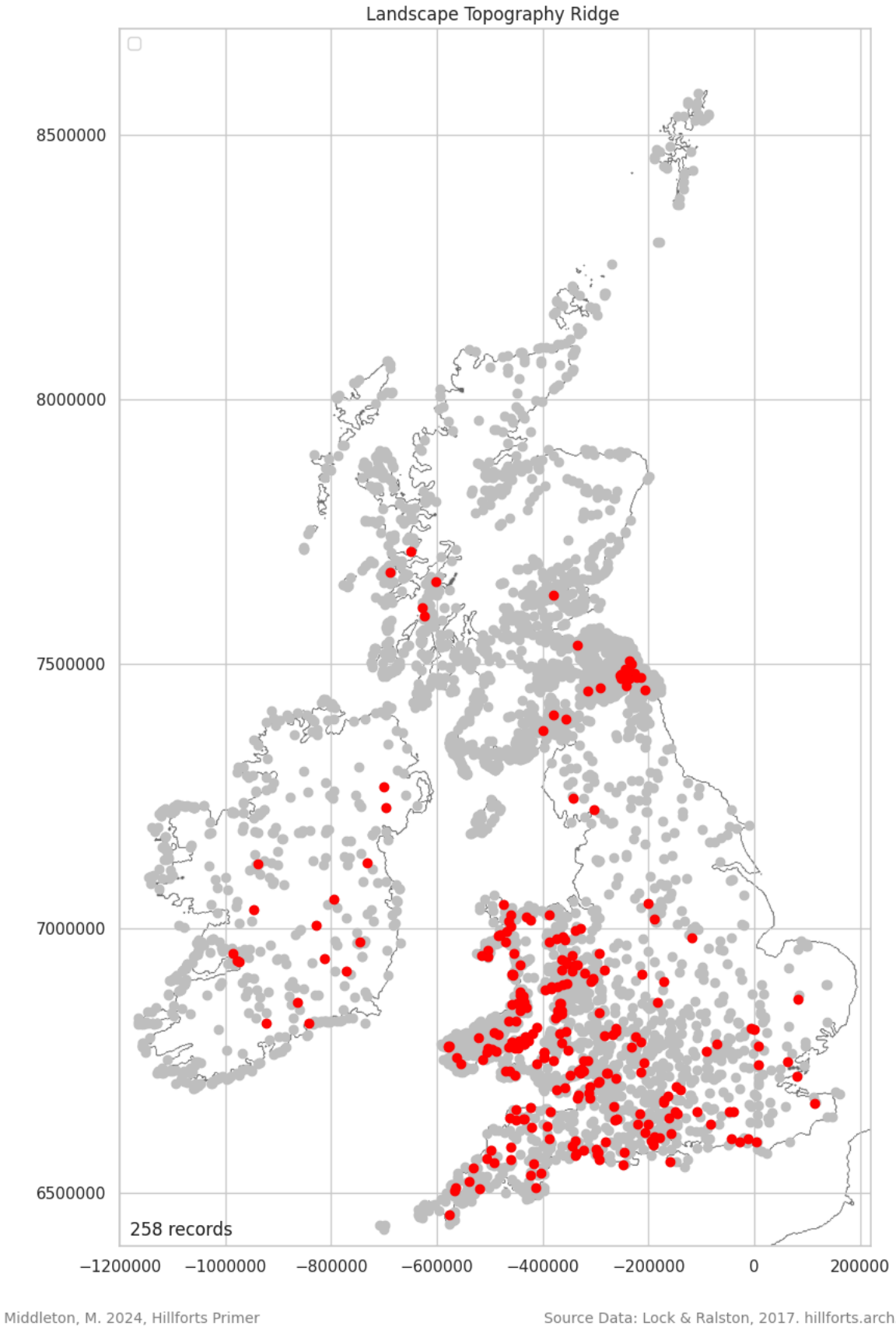
In [ ]:
```
plot_density_over_grey(topo_knoll, 'Landscape_Topography_Knoll')
```



Ridge Mapped

The distribution of hillforts identified as on a 'ridge' is almost entirely focussed toward the South East and Wales. This may reflect a regional terminology bias.

In [ ]:
```
topo_ridge = plot_over_grey(location_topo_data, \
                            'Landscape_Topography_Ridge', 'Yes')
```

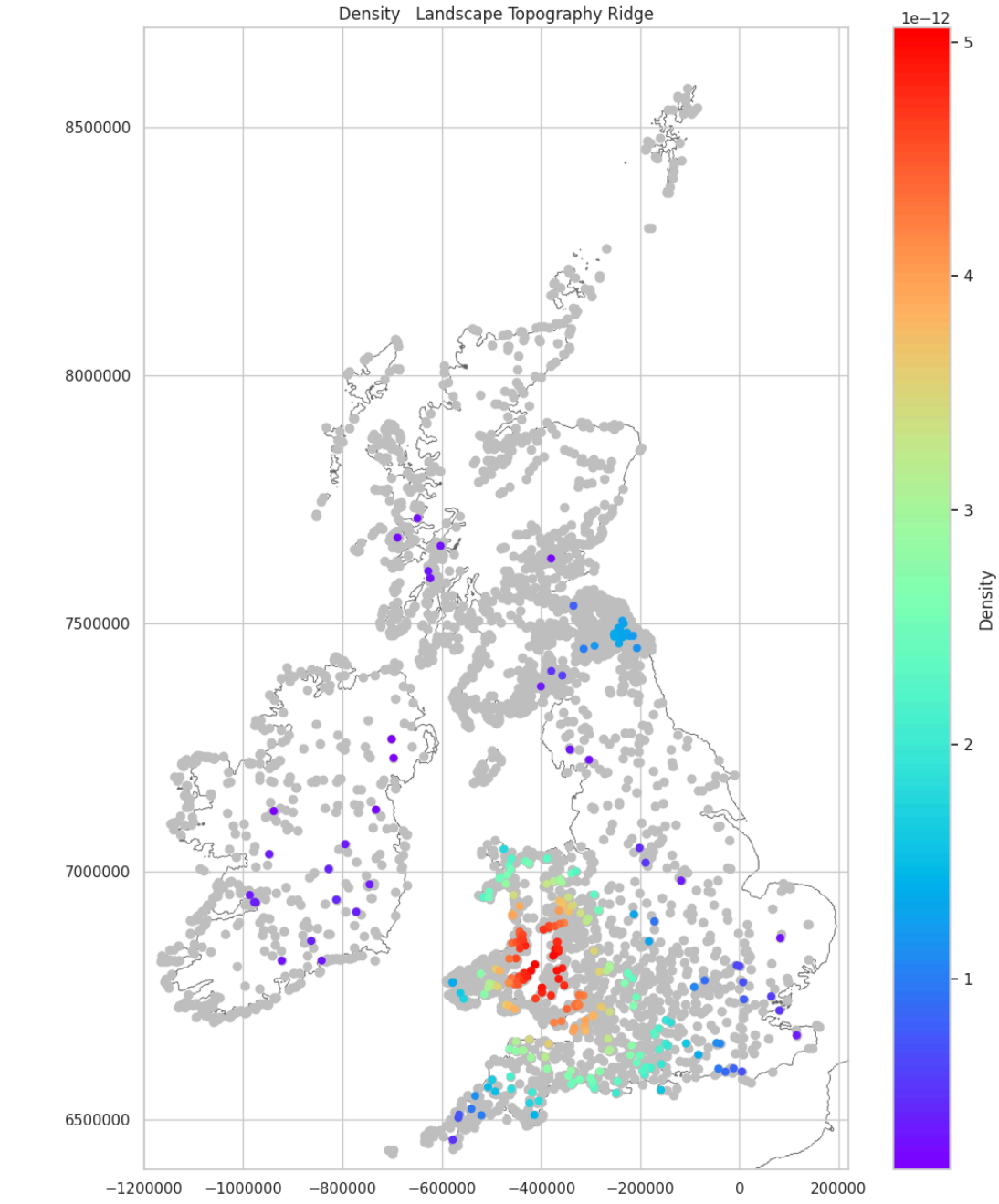


6.22%

Ridge Density Mapped

The southern cluster of hillforts on ridges coincides with the southern cluster seen in the full dataset. As mentioned above, there may be a terminology bias present in this distribution. See: Part 1: Density Map Showing Clusters Adjusted by Region.

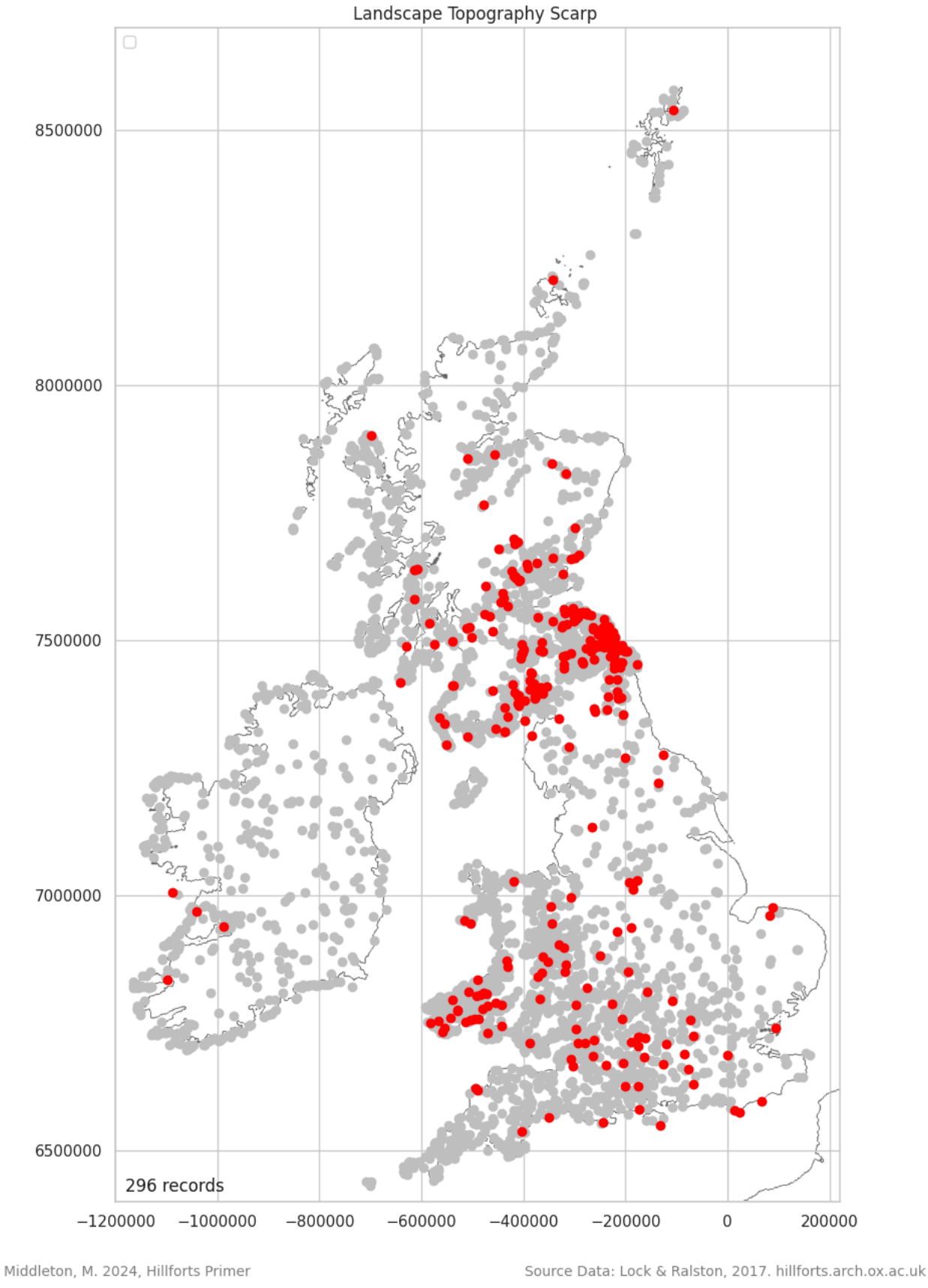
In [ ]:
```
plot_density_over_grey(topo_ridge, 'Landscape_Topography_Ridge')
```



Scarp Mapped

In Scotland, the use of the term 'Scarp' is concentrated mostly over the Southern Uplands and along the Highland Boundary fault. To the south, there is a concentration of hillforts in Pembrokeshire and a spread of hillforts across south, central England. The term is hardly used in Ireland and this may indicate a regional bias.

In [ ]:
```
topo_scarp = plot_over_grey(location_topo_data, \
                            'Landscape_Topography_Scarp', 'Yes')
```

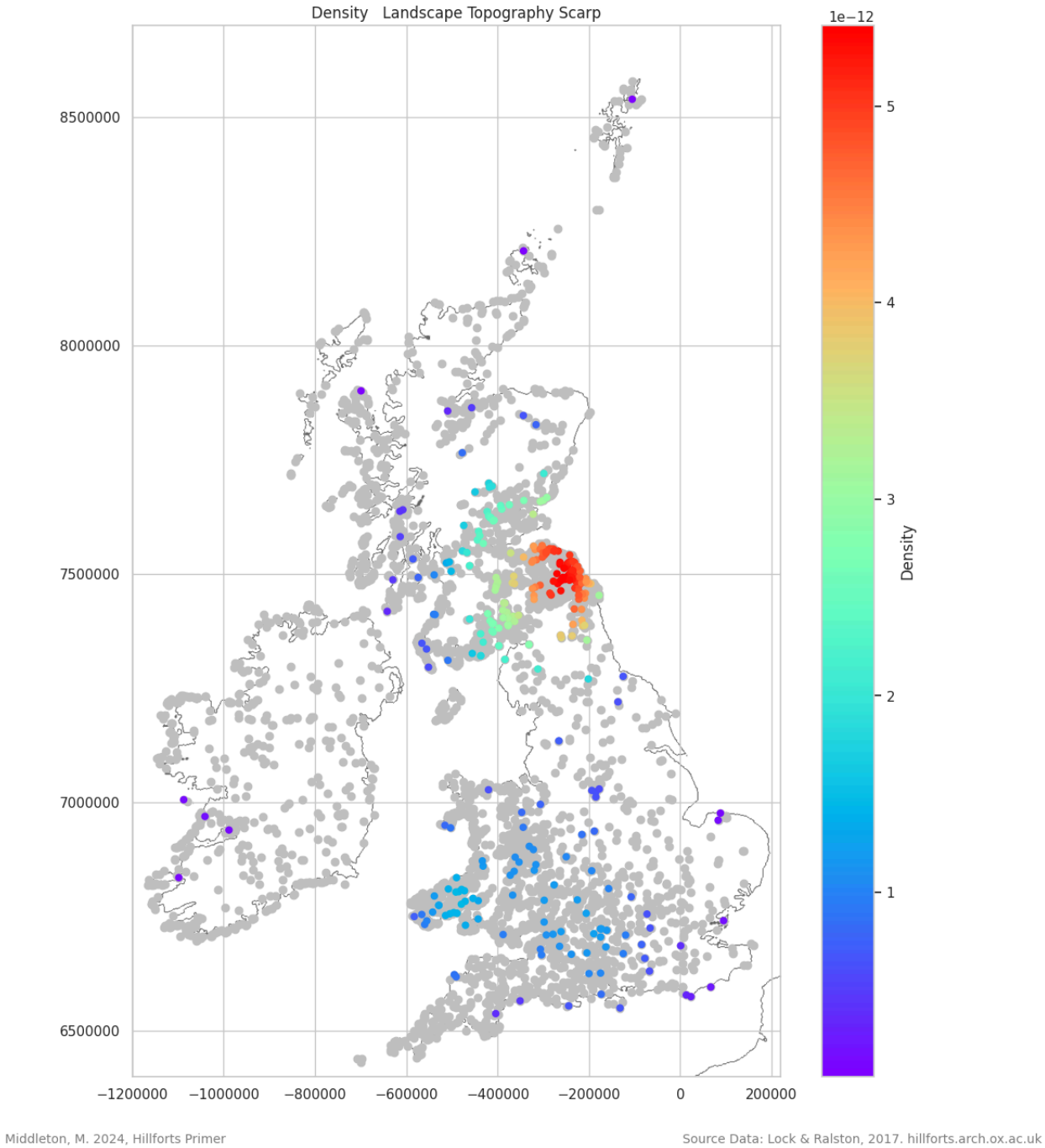


7.14%

Scarp Density Mapped

'Scarp' hillfort density over the Southern Uplands corresponds to the lower altitude forts in the Tweed Basin. (See: Altitude Over 0 - 99m Density Mapped). Just under half (48.34%) are known from the cropmark record. This distribution will contain a survey bias (intensive cropmark survey over Southern Scotland) and may also include a regional bias in the use of this term.

In [ ]:
```
plot_density_over_grey(topo_scarp, 'Landscape_Topography_Scarp')
```



In [ ]:
```
level_n = north[(north['Landscape_Type_Level']=='Yes')]
print(f'{len(level_n)} of the level hillforts are in the north.')
```

```
151 of the level hillforts are in the north.
```

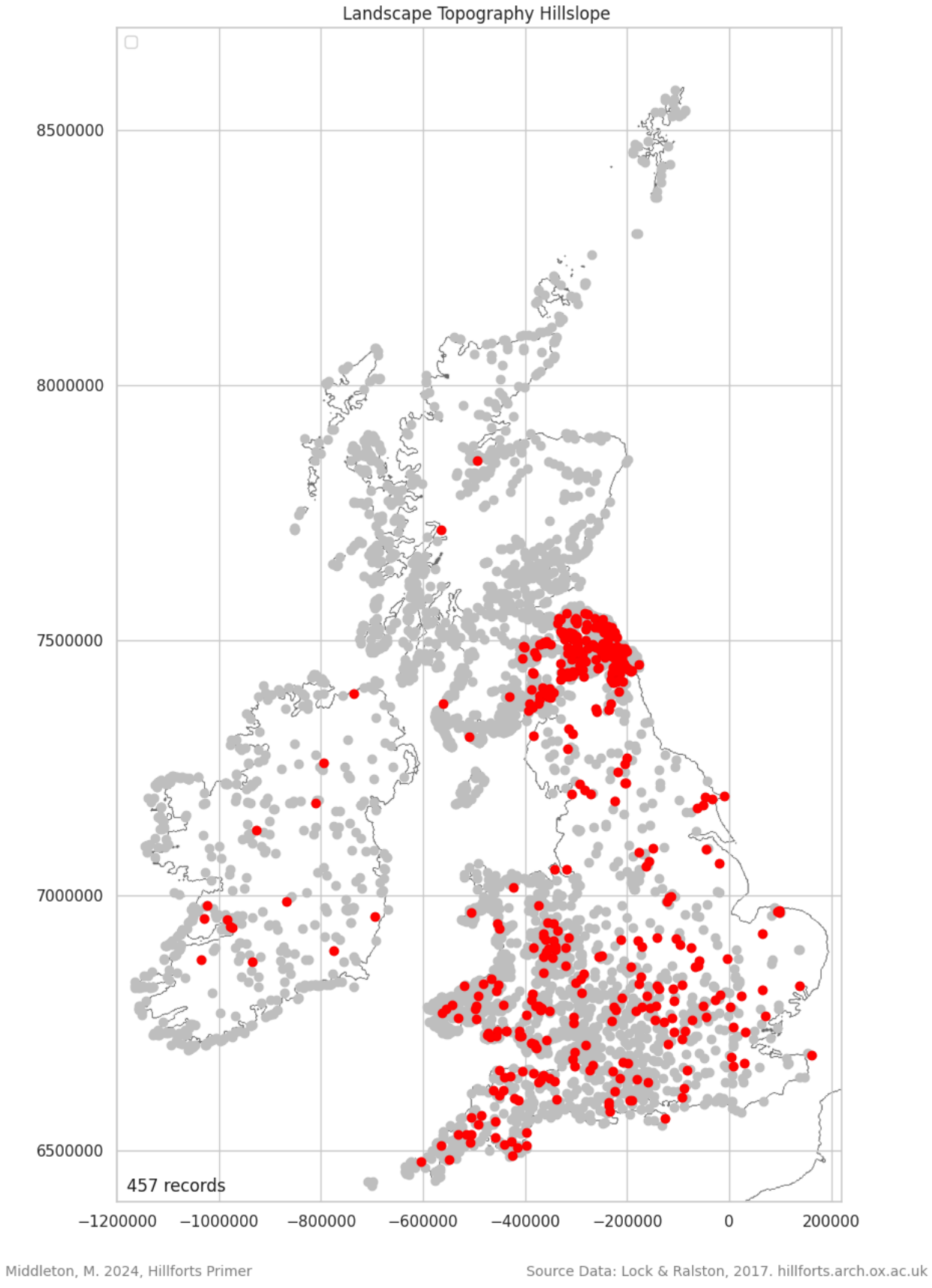
In [ ]:
```
scarp_cropmark_n = level_n[level_n['Landscape_Topography_Scarp']=='Yes']
print(f'{len(scarp_cropmark_n)} ({round((len(scarp_cropmark_n)/len(level_n))*100,2)}%) of the scarp hillforts in the
#print(f'{len(scarp_cropmark_n)} ({round((len(scarp_cropmark_n)/len(level_n))*
#100,2)}%) of the scarp hillforts in the north are cropmark sites.')
```

```
73 (48.34%) of the scarp hillforts in the north are cropmark sites.
```

Hillslope Mapped (Topography)

The distribution of the term 'Hillslope' shows a bias toward the Southern Uplands. The term has been used widely in England and Wales although there is a notable lack of this type in northwest Wales. There are very few hillforts recorded as 'Hillslope' across the rest of Scotland and the distribution across Ireland is low. The bias seen here is mirrored in the bias seen in Hillslope Mapped (Landscape).

In [ ]:
```
topo_hillslope = plot_over_grey(location_topo_data, \
                                'Landscape_Topography_Hillslope', 'Yes')
```

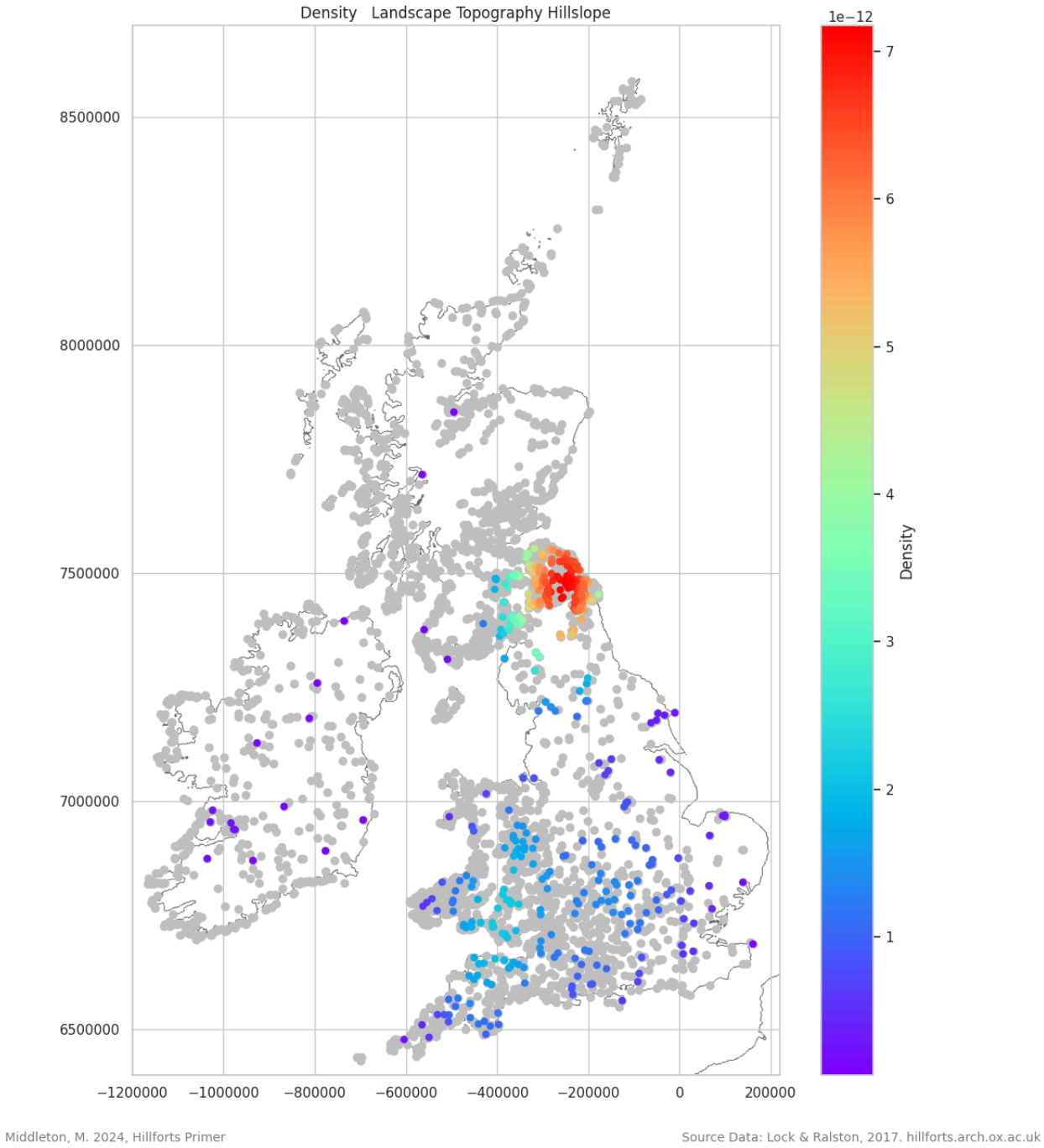



```
11.02%
```

Hillslope Density Mapped (Topography)

The 'Hillslope' density plot is very similar to that seen in Hillslope Density Mapped (Landscape).

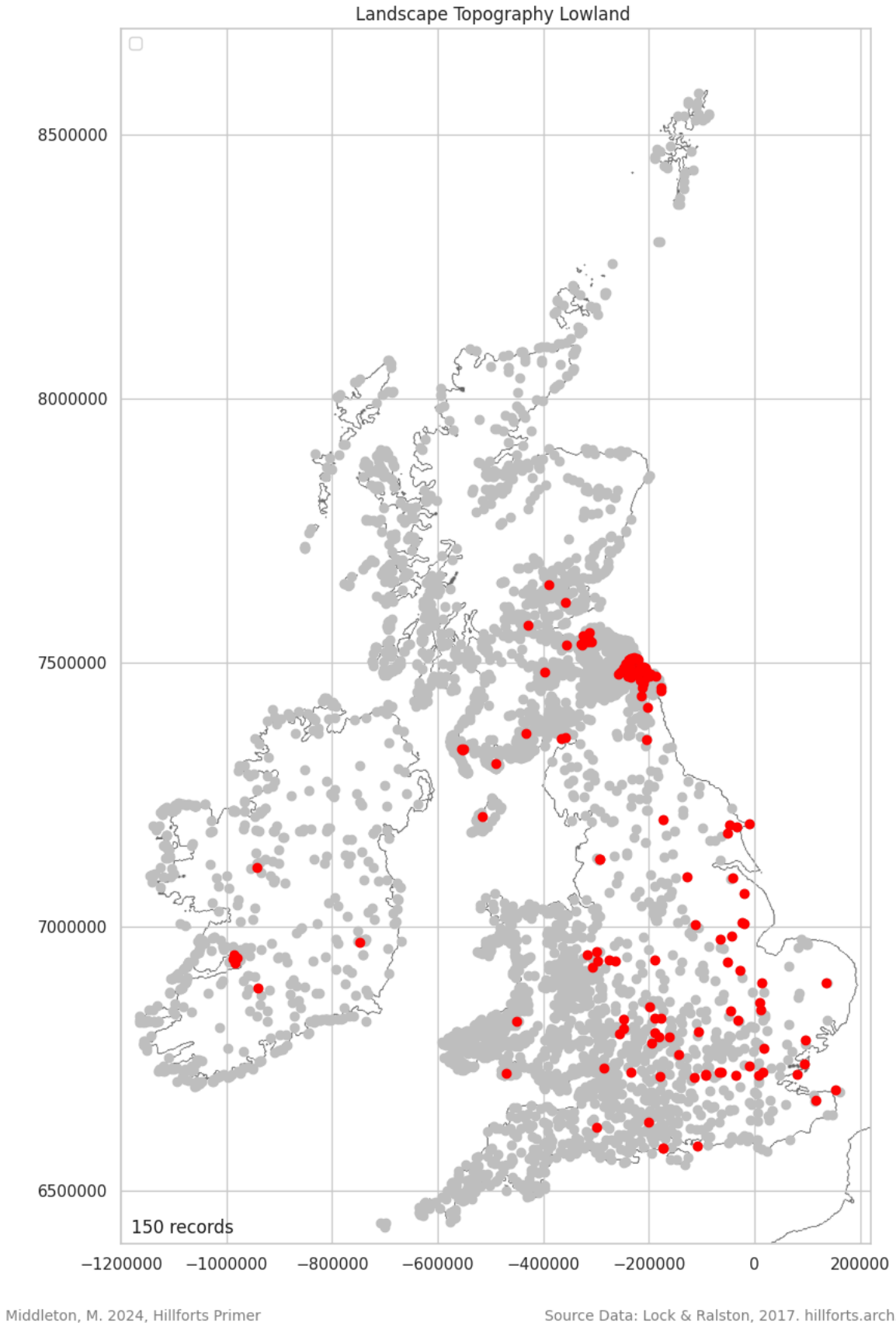
In [ ]:
```
plot_density_over_grey(topo_hillslope, 'Landscape_Topography_Hillslope')
```



Lowland Mapped

'Lowland' is a term that is relative to the topography rather than being a specific topographic type. Like similar relative terms used in this section, some 'lowland' hillforts have multiple topographic types recorded. There is a bias in the use of this term toward eastern England although the term has been used, in small numbers, across the whole of the atlas. Because of the bias, a density plot has not been produced.

In [ ]:
```
topo_lowland = plot_over_grey(location_topo_data, \
                              'Landscape_Topography_Lowland', 'Yes')
```

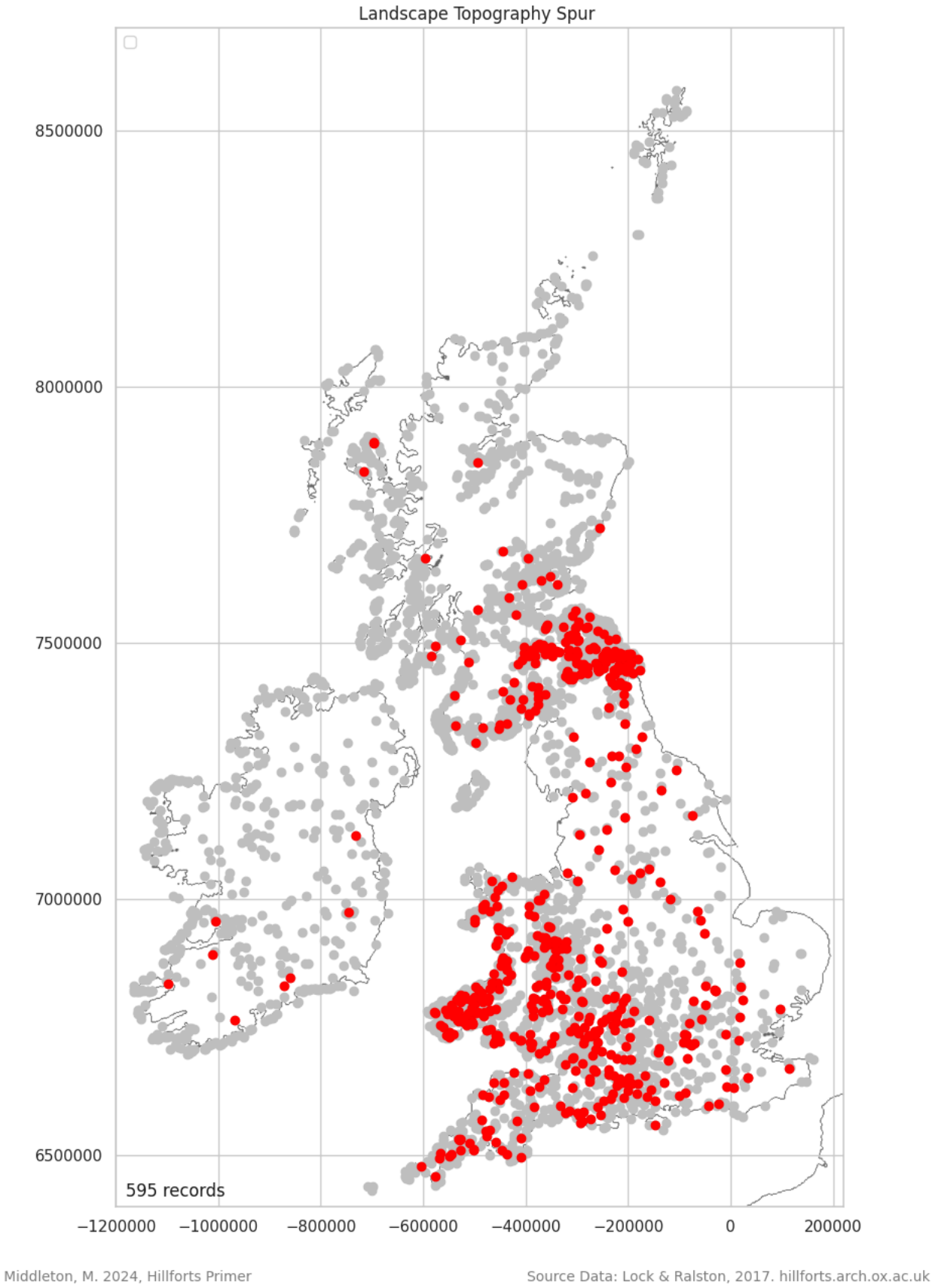


3.62%

Spur Mapped

The distribution of the term 'Spur' is concentrated over the Southern Uplands, Wales and southeast England. The sparce distribution in Ireland and northern Scotland may hint at there being a recording bias.

In [ ]:
```
topo_spur = plot_over_grey(location_topo_data, \
                           'Landscape_Topography_Spur', 'Yes')
```

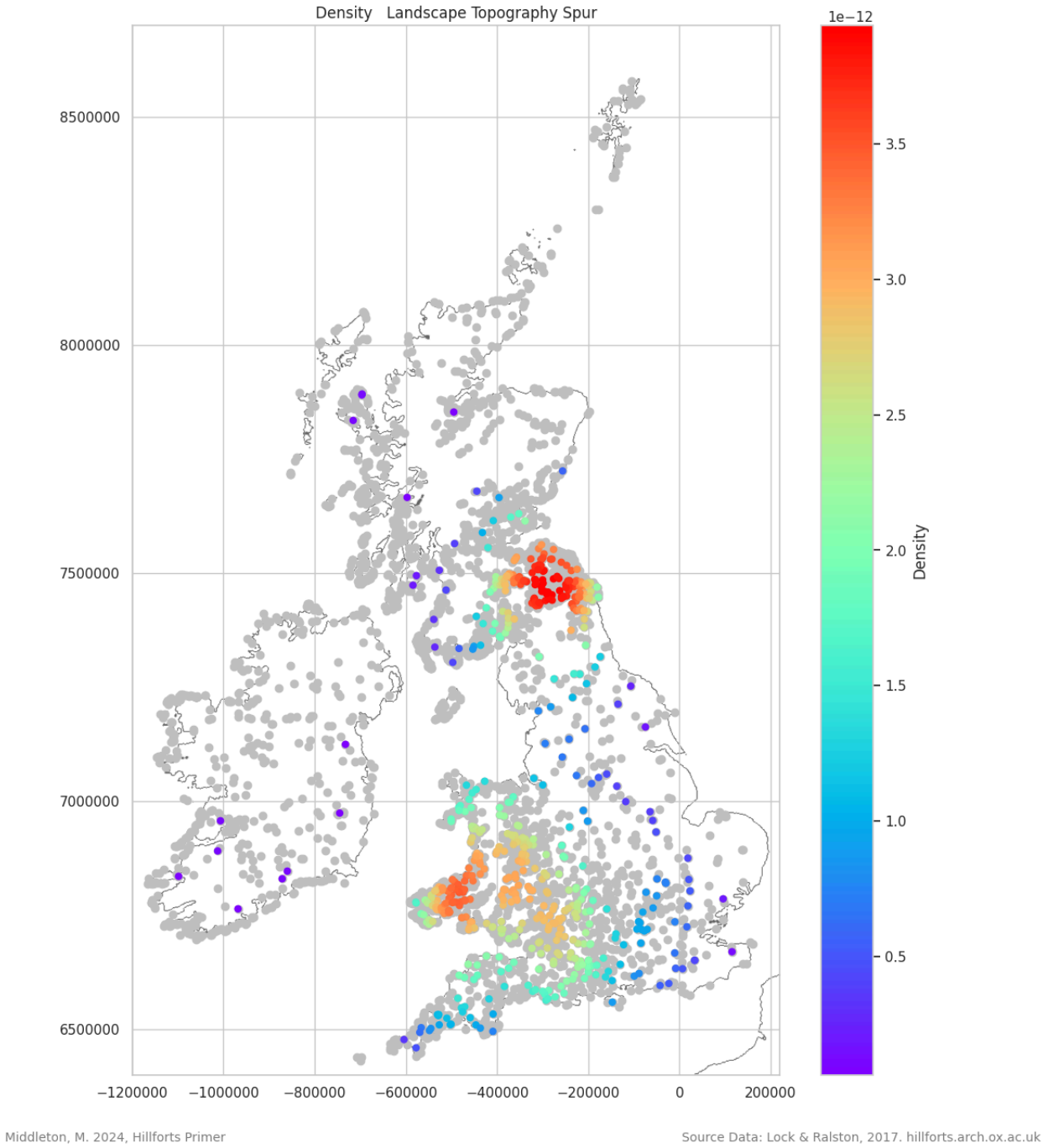


14.35%

Spur Density Mapped

'Spur' density corresponds with the two main clusters in the full dataset (See Part 1: Density Map Showing Clusters Adjusted by Region). The clusters in the Northwest and the west of Ireland are not present.

In [ ]:
```
plot_density_over_grey(topo_spur, 'Landscape_Topography_Spur')
```



Aspect Data

This data records if a hillfort's aspect is level or toward one or more of the eight cardinal or intercardinal angles.

In [ ]:
```
landscape_aspect_features = [
 'Landscape_Aspect_N',
 'Landscape_Aspect_NE',
 'Landscape_Aspect_E',
 'Landscape_Aspect_SE',
 'Landscape_Aspect_S',
 'Landscape_Aspect_SW',
 'Landscape_Aspect_W',
 'Landscape_Aspect_NW',
 'Landscape_Aspect_Level']

landscape_aspect_data = \
landscape_data[landscape_aspect_features].copy()
landscape_aspect_data.head()
```

Out[ ]:

	Landscape_Aspect_N	Landscape_Aspect_NE	Landscape_Aspect_E	Landscape_Aspect_SE	Landscape_Aspect_S	Landscape_Aspect_SW	Lands
0	No	No	No	No	No	No	
1	No	No	No	No	No	No	
2	No	No	No	Yes	Yes	No	
3	No	No	No	No	No	No	
4	No	No	No	No	No	No	

There are no null values in the aspect data.

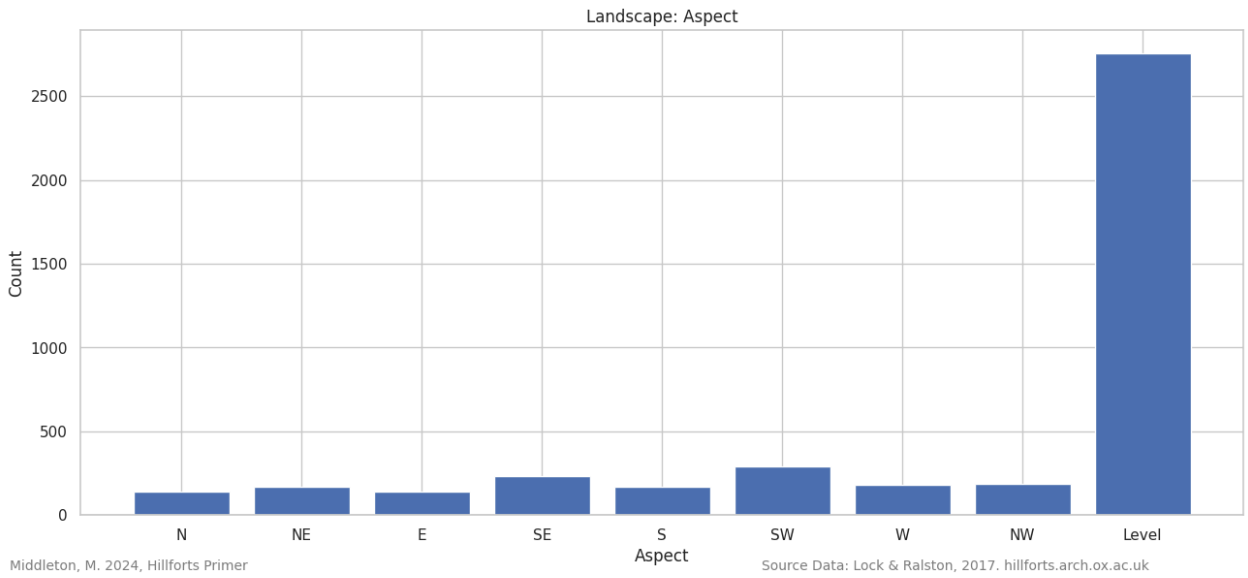
In [ ]:
```
landscape_aspect_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4147 entries, 0 to 4146
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Landscape_Aspect_N     4147 non-null   object
 1   Landscape_Aspect_NE    4147 non-null   object
 2   Landscape_Aspect_E     4147 non-null   object
 3   Landscape_Aspect_SE    4147 non-null   object
 4   Landscape_Aspect_S     4147 non-null   object
 5   Landscape_Aspect_SW    4147 non-null   object
 6   Landscape_Aspect_W     4147 non-null   object
 7   Landscape_Aspect_NW    4147 non-null   object
 8   Landscape_Aspect_Level 4147 non-null   object
dtypes: object(9)
memory usage: 291.7+ KB
```

Aspect Data Plotted

2756 hillforts (66.46%) are recorded as being level. A small number of hillforts have alternative or additional aspect information. There is a recording bias in this data in that most sites with an aspect, other than level, are in England, Wales and Ireland. Aspects other than level have rarely been recorded in Scotland. 34 hillforts have no aspect recorded of which ten are identified as having been destroyed. The 24 extant forts without an aspect are all in England.

In [ ]:
```
plot_bar_chart(landscape_aspect_data, 2, 'Aspect', 'Count', 'Landscape: Aspect')
```



In [ ]:
```
level_df = \
landscape_aspect_data[landscape_aspect_data['Landscape_Aspect_Level'] == "Yes"]
len(level_df)
```

Out[ ]:
```
2756
```

In [ ]:
```
no_aspect_data = hillforts_data.copy()
for feature in landscape_aspect_features:
    no_aspect_data = no_aspect_data[landscape_aspect_data[feature] == "No"]
len(no_aspect_data)
```

```
<ipython-input-202-c24940298d19>:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  no_aspect_data = no_aspect_data[landscape_aspect_data[feature] == "No"]
```

Out[ ]:
```
34
```

In [ ]:
```
not_destroyed_no_aspect = \
no_aspect_data[no_aspect_data['Management_Condition_Destroyed'] == "No"]
len(not_destroyed_no_aspect)
```

Out[ ]:
```
24
```

In [ ]:
```
no_aspect_in_england_not_destroyed =\
 not_destroyed_no_aspect[not_destroyed_no_aspect['Main_Country'] == "England"]
len(no_aspect_in_england_not_destroyed)
```

Out[ ]:
```
24
```

Aspect Data Spiderplot (Excluding Level)

Where an aspect other than level is recorded, there is a slight preference to the southwest. The mapped orientations are all based on very small subsets of the data. It is not clear if the orientations are a result of available local topography or if specific locations were chosen for a preferred aspect. The spider plot shows that the north, Northeast and east are the least favoured but it also shows that an aspect to the south has a similarly low count. It is likely that available local topography was the primary concern and that, if available, a south western aspect was desirable – remembering that level is, by far, the preferred choice.

In [ ]:
```
landscape_aspect_data_minus = \
landscape_aspect_data.drop(['Landscape_Aspect_Level'], axis=1)
landscape_aspect_data_minus.head()
```

Out[ ]:

	Landscape_Aspect_N	Landscape_Aspect_NE	Landscape_Aspect_E	Landscape_Aspect_SE	Landscape_Aspect_S	Landscape_Aspect_SW	Lands
0	No	No	No	No	No	No	
1	No	No	No	No	No	No	
2	No	No	No	Yes	Yes	No	
3	No	No	No	No	No	No	
4	No	No	No	No	No	No	

In [ ]:
```
landscape_aspect_data_minus_reordered = landscape_aspect_data_minus[[
    'Landscape_Aspect_E',
    'Landscape_Aspect_NE',
    'Landscape_Aspect_N',
    'Landscape_Aspect_NW',
    'Landscape_Aspect_W',
    'Landscape_Aspect_SW',
    'Landscape_Aspect_S',
    'Landscape_Aspect_SE']].copy()
landscape_aspect_data_minus_reordered.head()
```

Out[ ]:

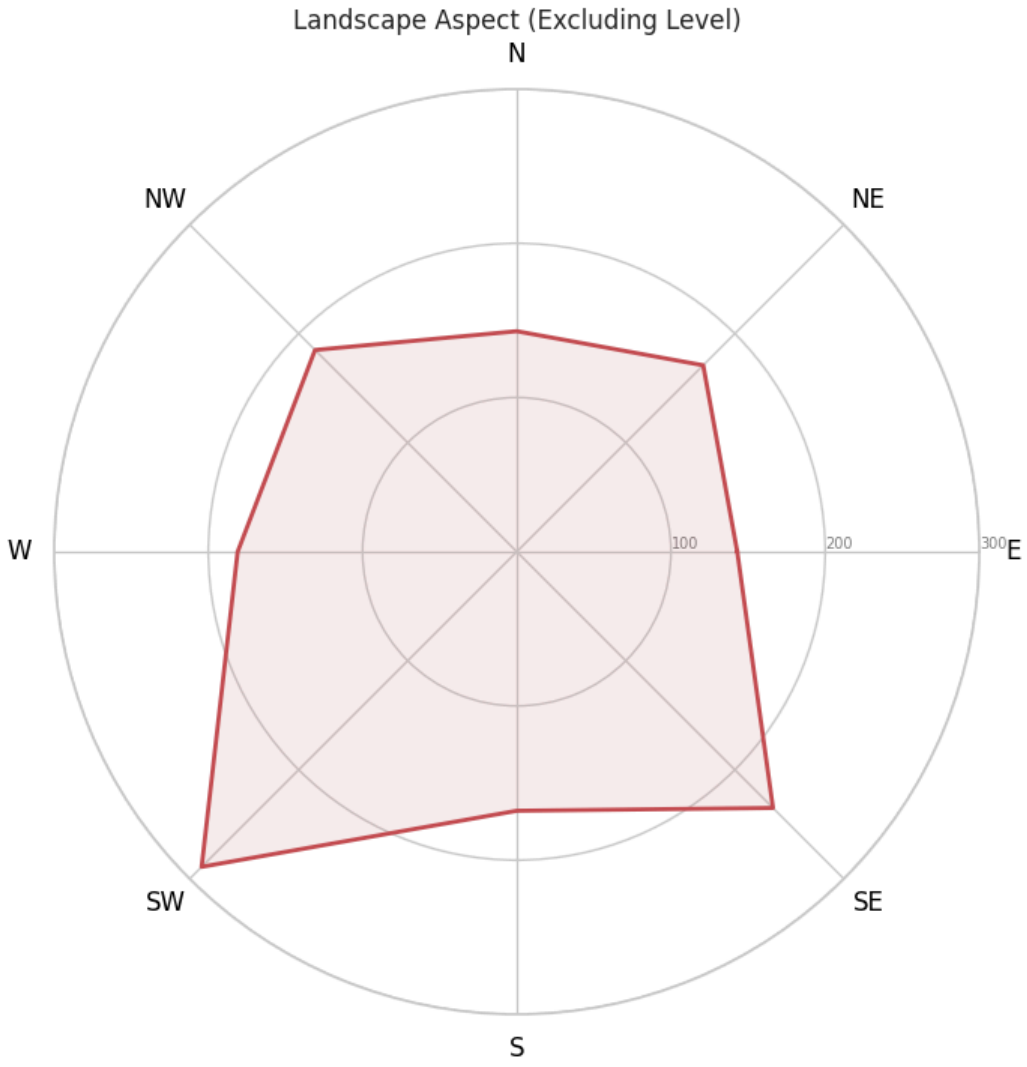
	Landscape_Aspect_E	Landscape_Aspect_NE	Landscape_Aspect_N	Landscape_Aspect_NW	Landscape_Aspect_W	Landscape_Aspect_SW	La
0	No	No	No	No	Yes	No	
1	No	No	No	No	No	No	
2	No	No	No	No	No	No	
3	No	No	No	Yes	No	No	
4	No	No	No	No	No	No	

In [ ]:
```
landscape_aspect_counts = {}
landscape_aspect_counts['group'] = ['Total']
for col in landscape_aspect_data_minus_reordered.columns:
    count = \
    len(landscape_aspect_data_minus_reordered\
     [landscape_aspect_data_minus_reordered[col] == 'Yes'])
    landscape_aspect_counts[col] = [count]

landscape_aspect_counts
```

Out[ ]:
```
{'group': ['Total'],
 'Landscape_Aspect_E': [143],
 'Landscape_Aspect_NE': [171],
 'Landscape_Aspect_N': [143],
 'Landscape_Aspect_NW': [185],
 'Landscape_Aspect_W': [181],
 'Landscape_Aspect_SW': [289],
 'Landscape_Aspect_S': [168],
 'Landscape_Aspect_SE': [235]}
```

In [ ]:
```
spider_plot(landscape_aspect_counts, 'Landscape_Aspect (Excluding_Level)')
```



Aspect Data Mapped

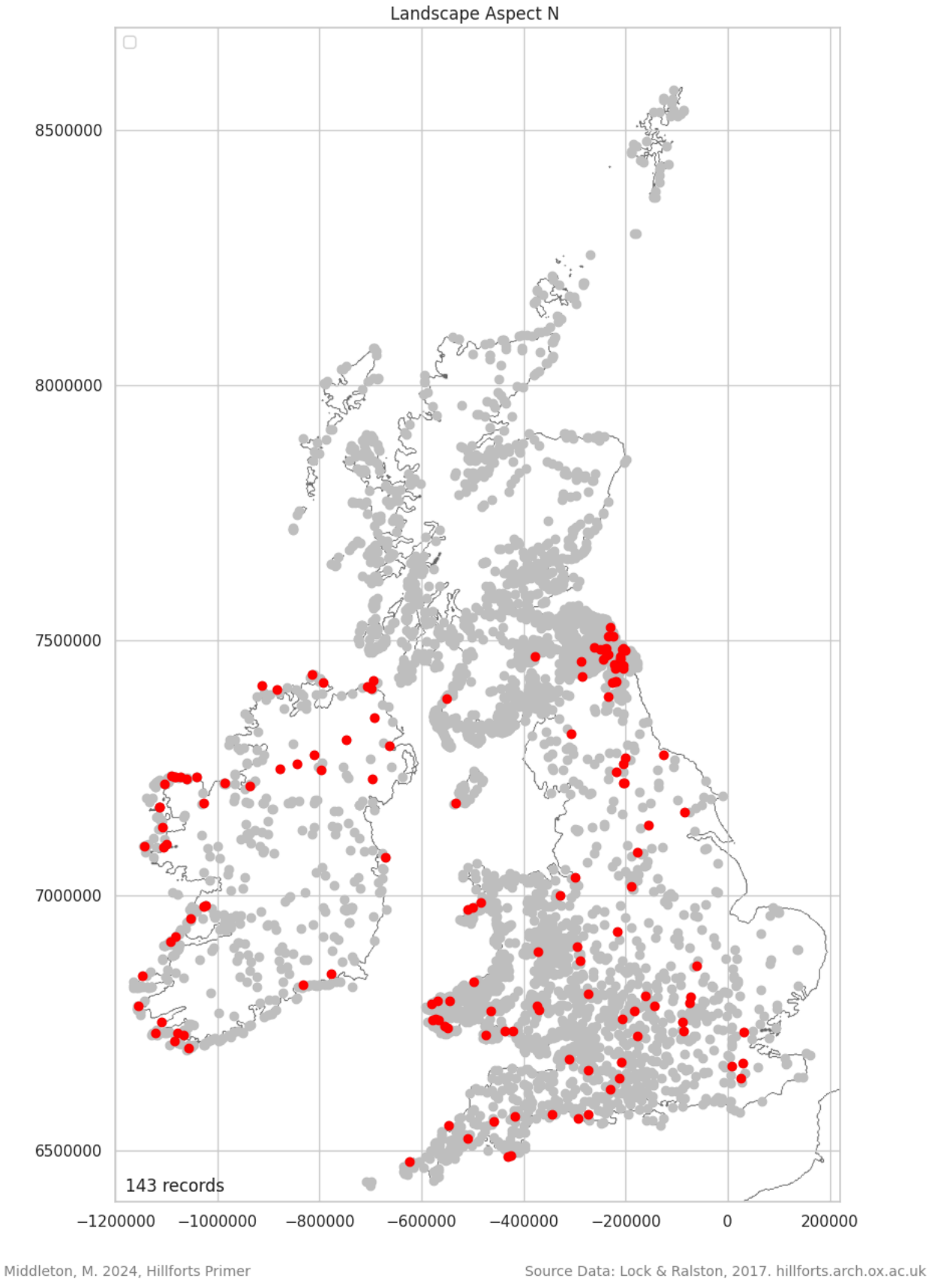
There is a recording bias in that aspects other than level have not been routinely recorded in Scotland.

In [ ]:
```
location_aspect_data = \
pd.merge(location_numeric_data_short, landscape_aspect_data, \
         left_index=True, right_index=True)
```

Aspect North Mapped

143 hillforts (3.45%) have an aspect facing north.

In [ ]:
```
asp_n = plot_over_grey(location_aspect_data, 'Landscape_Aspect_N', 'Yes')
```

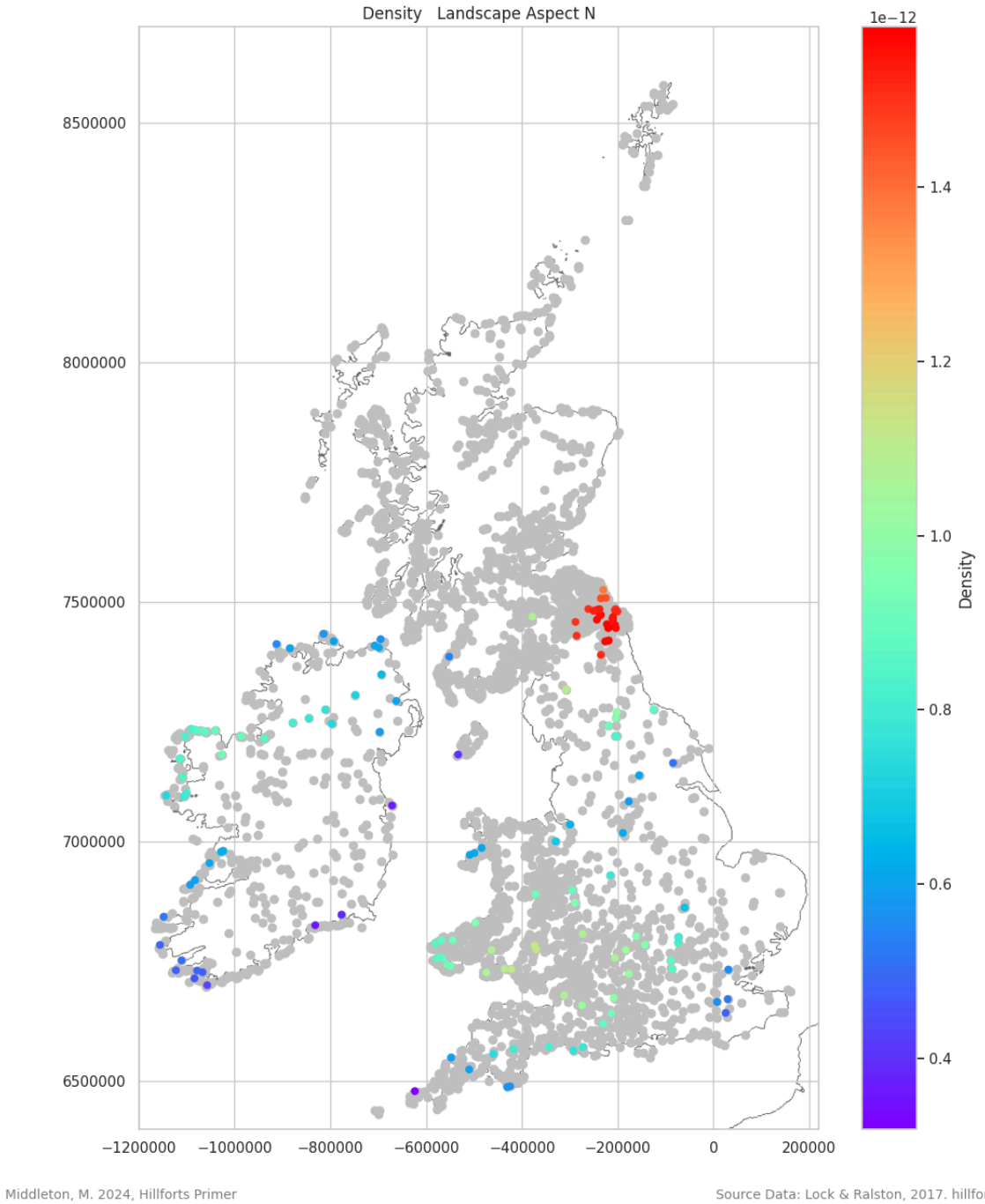


3.45%

Aspect Density North Mapped

The main cluster of hillforts, with a north facing aspect, are in the Southern Uplands. There is also a notable concentration of forts along the west coast of Ireland.

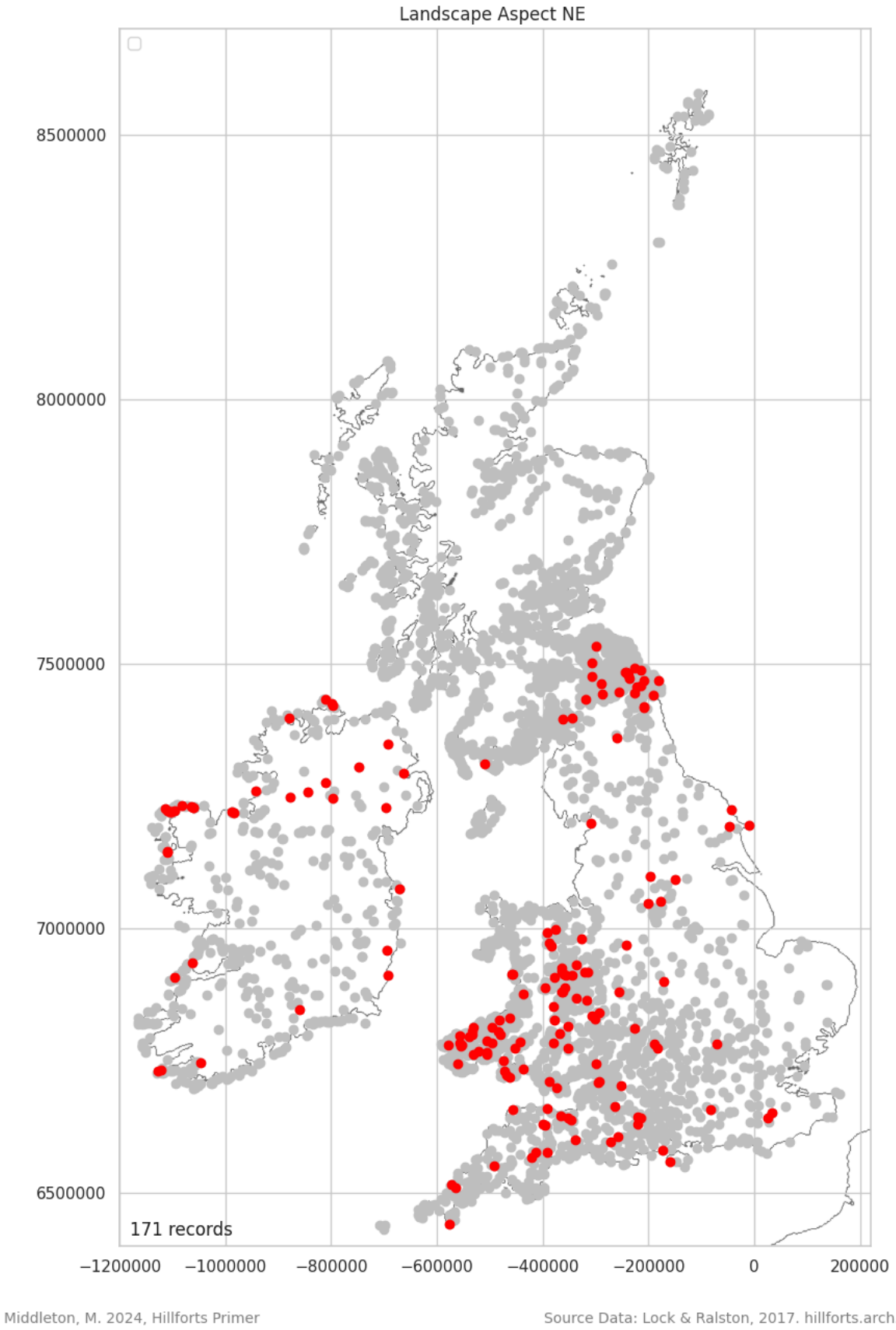
In [ ]:
```
plot_density_over_grey(asp_n, 'Landscape_Aspect_N')
```



Aspect Northeast Mapped

171 hillforts (4.12%) have an aspect facing Northeast.

In [ ]:
```
asp_ne = plot_over_grey(location_aspect_data, 'Landscape_Aspect_NE', 'Yes')
```

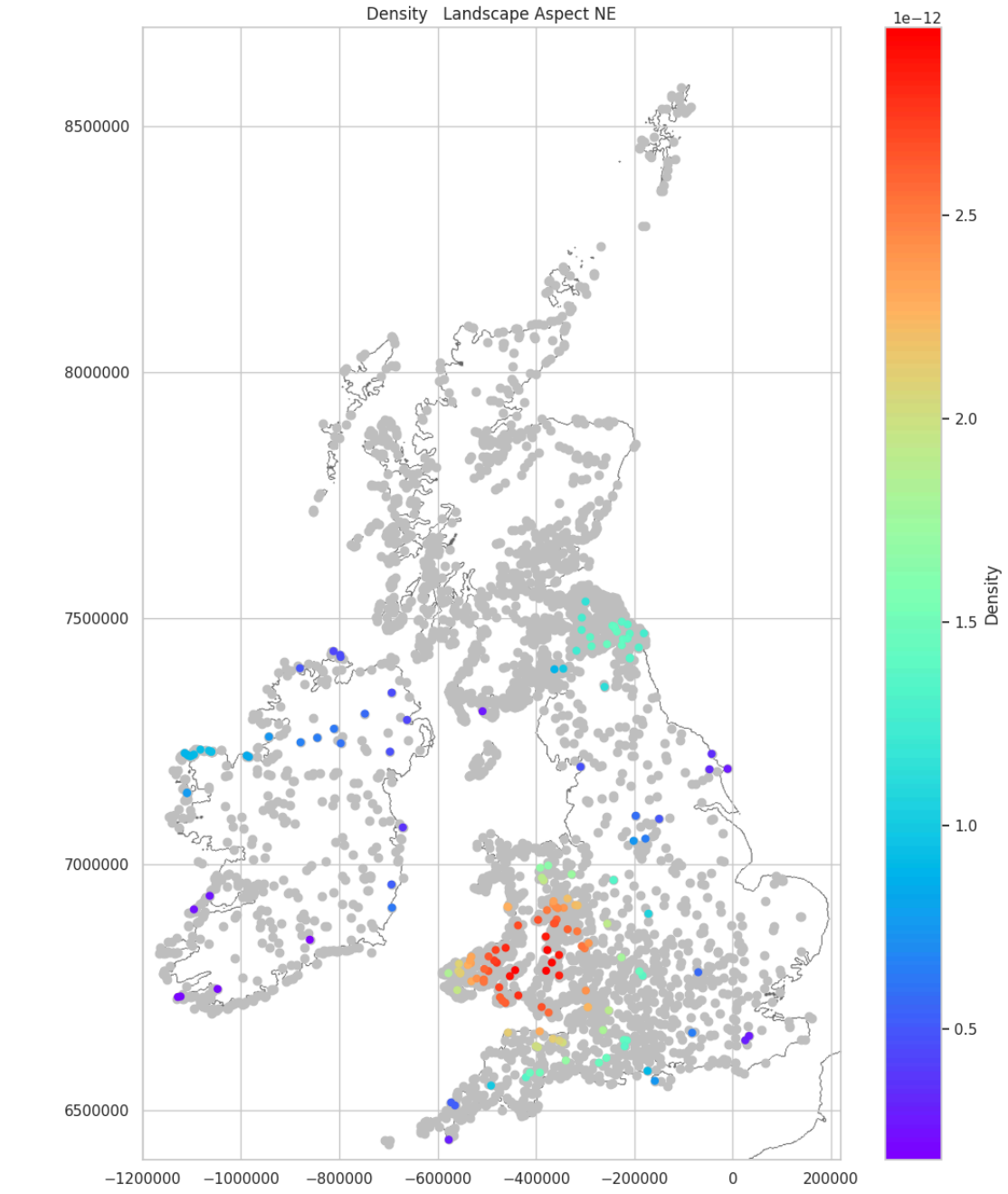


4.12%

Aspect Density Northeast Mapped

For hillforts with a Northeast aspect the focus is in the south, toward the southern end of the Cambrian Mountains.

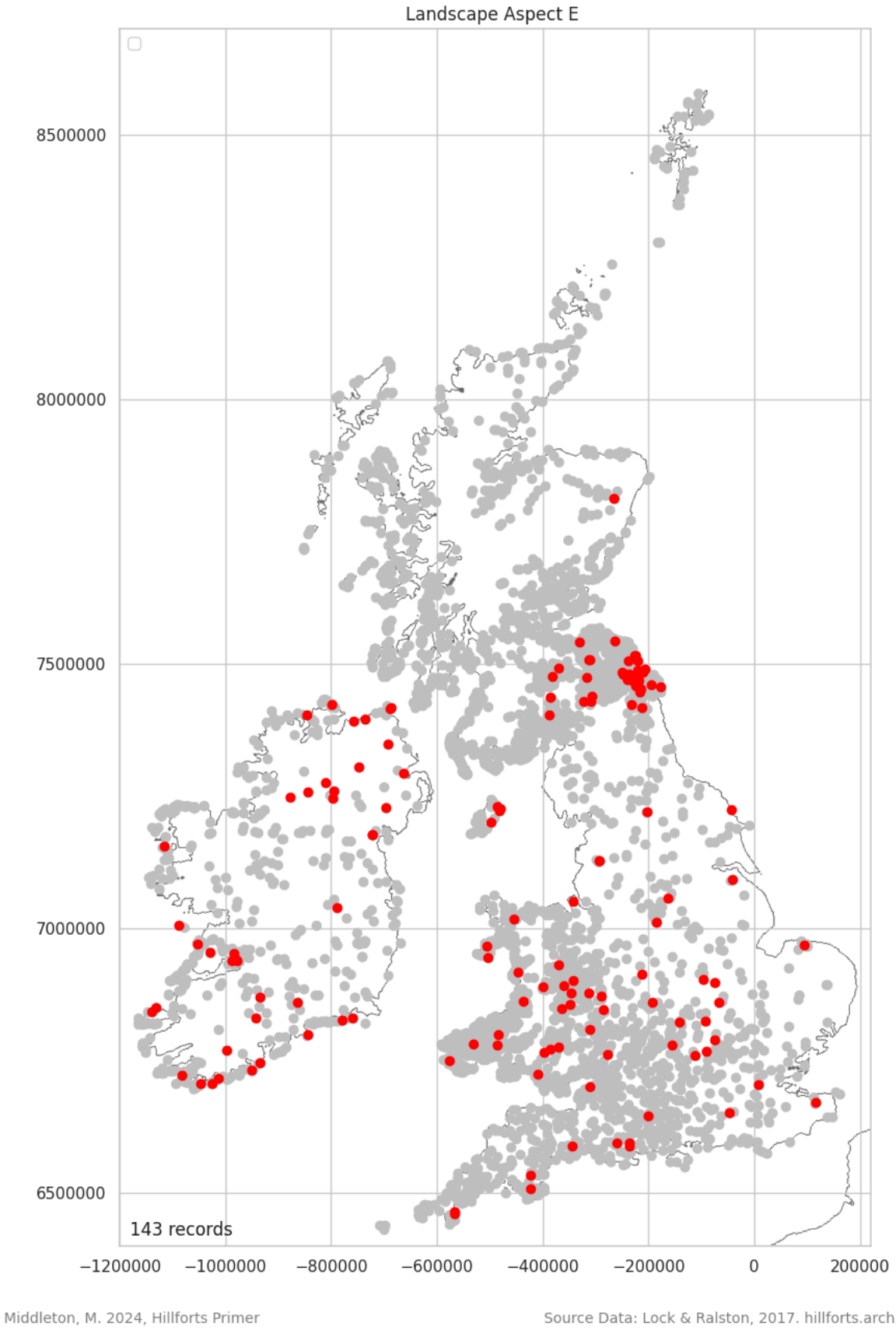
In [ ]:
```
plot_density_over_grey(asp_ne, 'Landscape_Aspect_NE')
```



Aspect East Mapped

143 hillforts (3.45%) have an aspect facing east.

In [ ]:
```
asp_e = \
plot_over_grey(location_aspect_data, 'Landscape_Aspect_E', 'Yes')
```

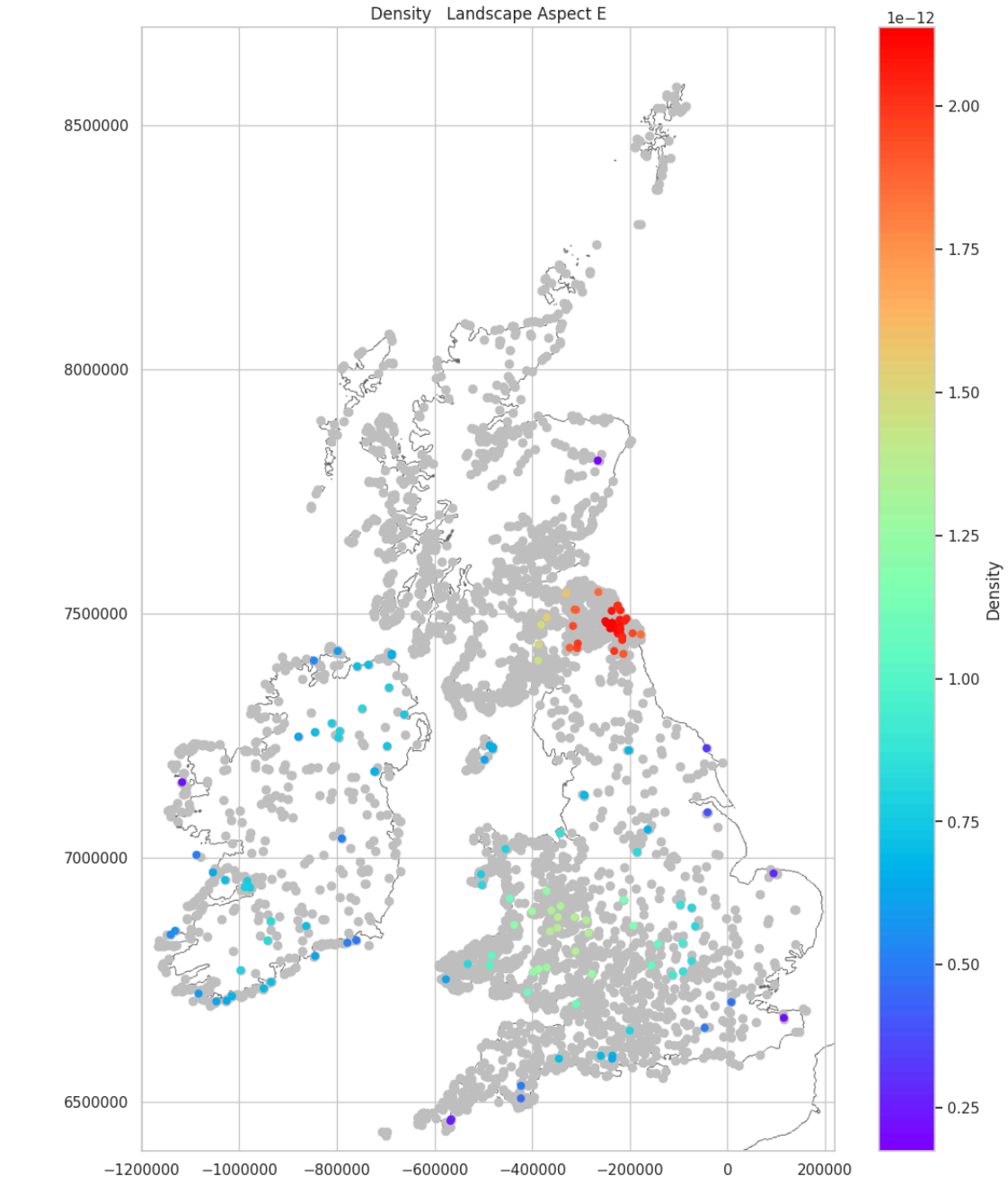


3.45%

Aspect Density East Mapped

For hillforts with an east facing aspect the focus is again over the Tweed Basin.

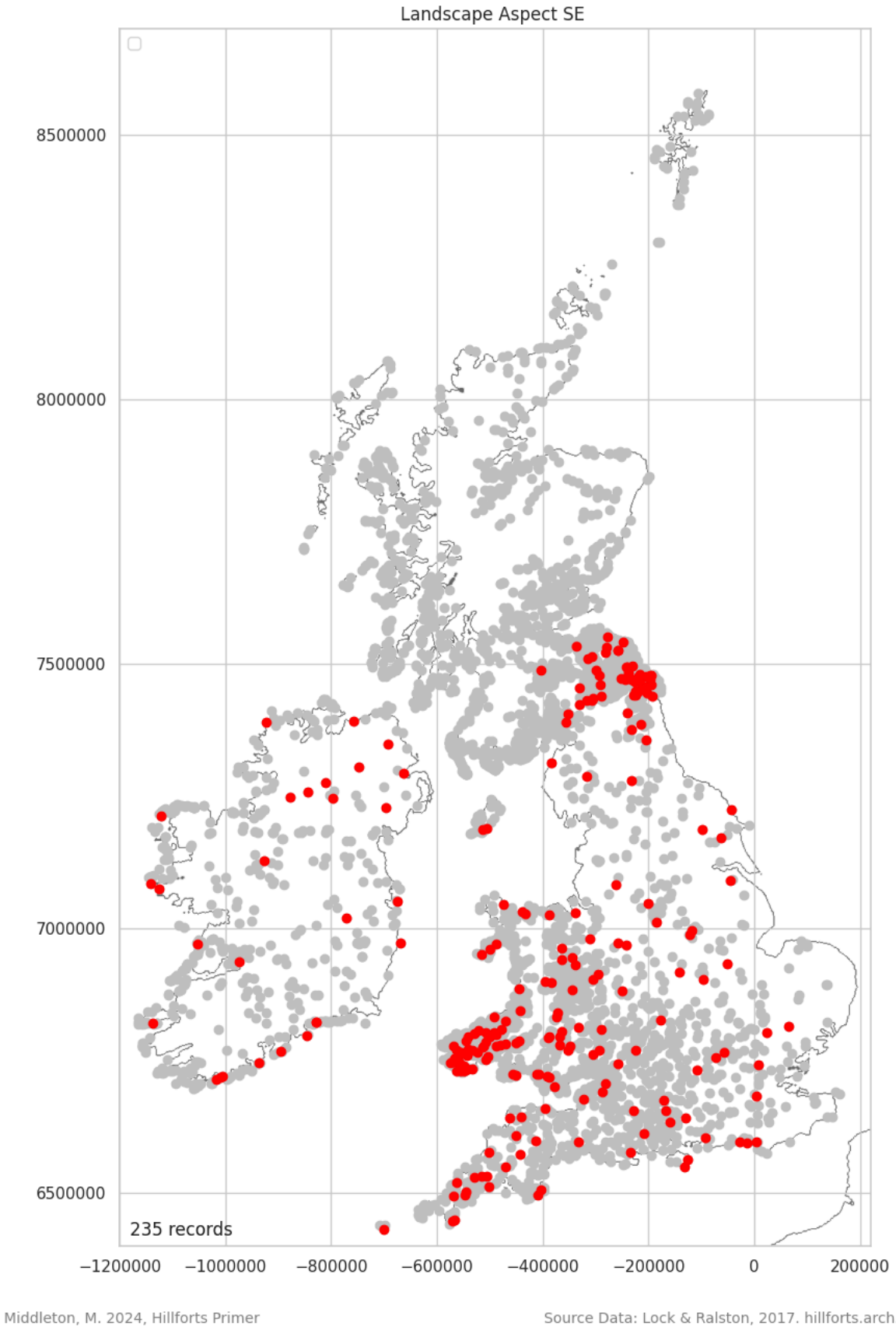
In [ ]:
```
plot_density_over_grey(asp_e, 'Landscape_Aspect_E')
```



Aspect South East Mapped

235 hillforts (5.67%) have an aspect facing south east.

In [ ]:
```
asp_se = plot_over_grey(location_aspect_data, 'Landscape_Aspect_SE', 'Yes')
```

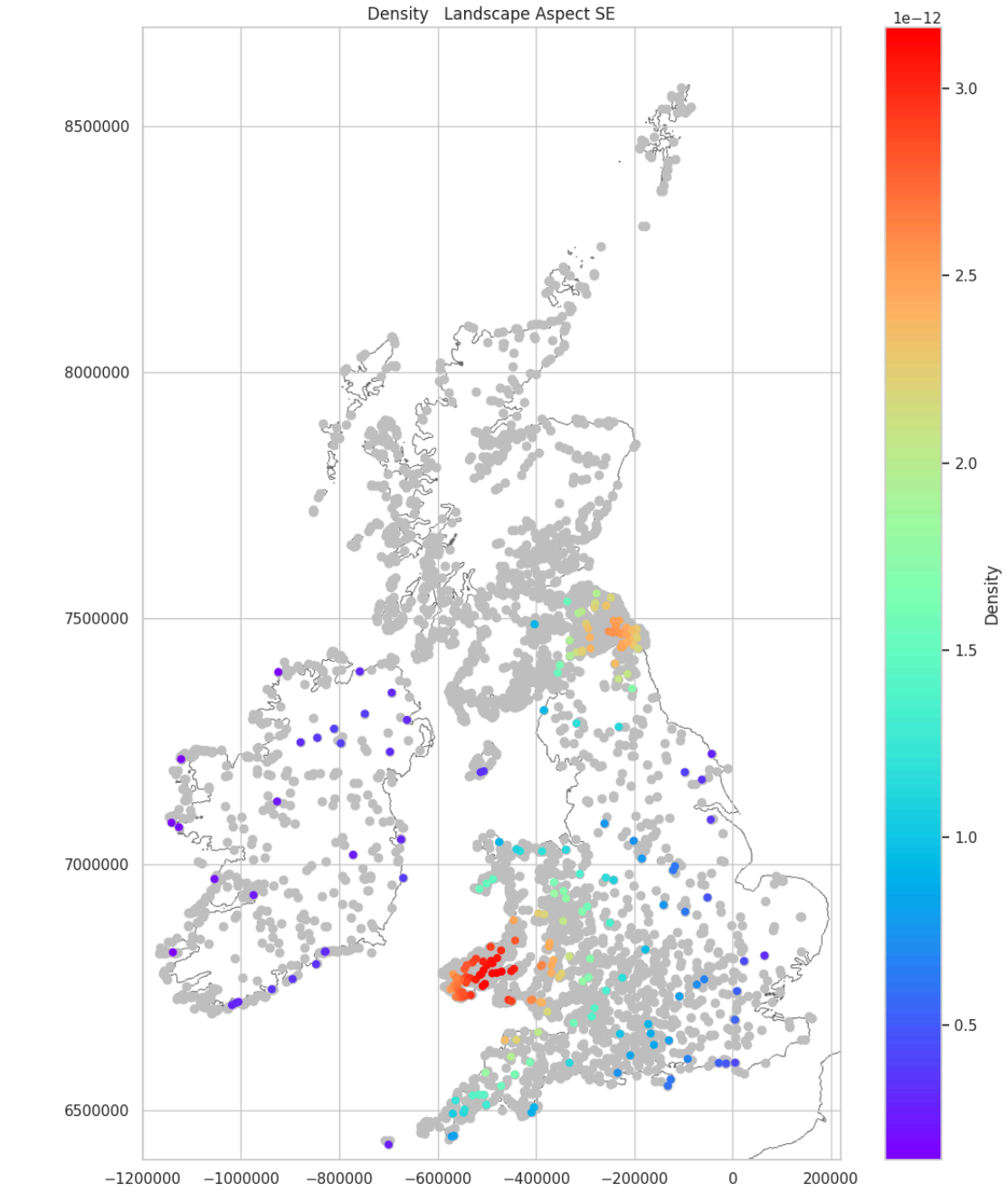


5.67%

Aspect Density South East Mapped

Hillforts with a south-eastern aspect are mostly clustered over the Pembrokeshire peninsula. There is also a moderate concentration over the Tweed Basin and the Cheviots.

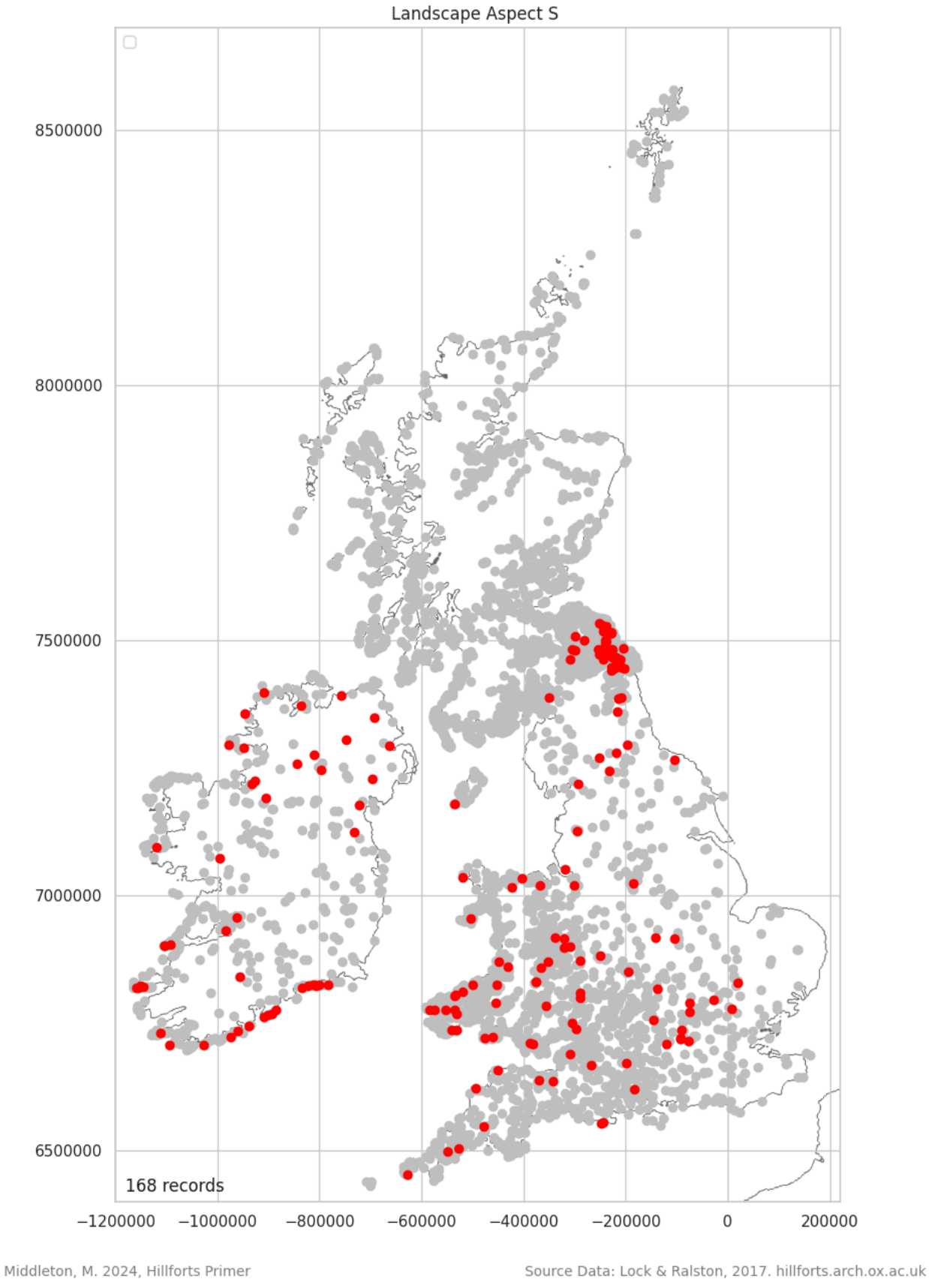
In [ ]:
```
plot_density_over_grey(asp_se, 'Landscape_Aspect_SE')
```



Aspect South Mapped

168 hillforts (4.05%) have an aspect facing south.

In [ ]:
```
asp_s = plot_over_grey(location_aspect_data, 'Landscape_Aspect_S', 'Yes')
```

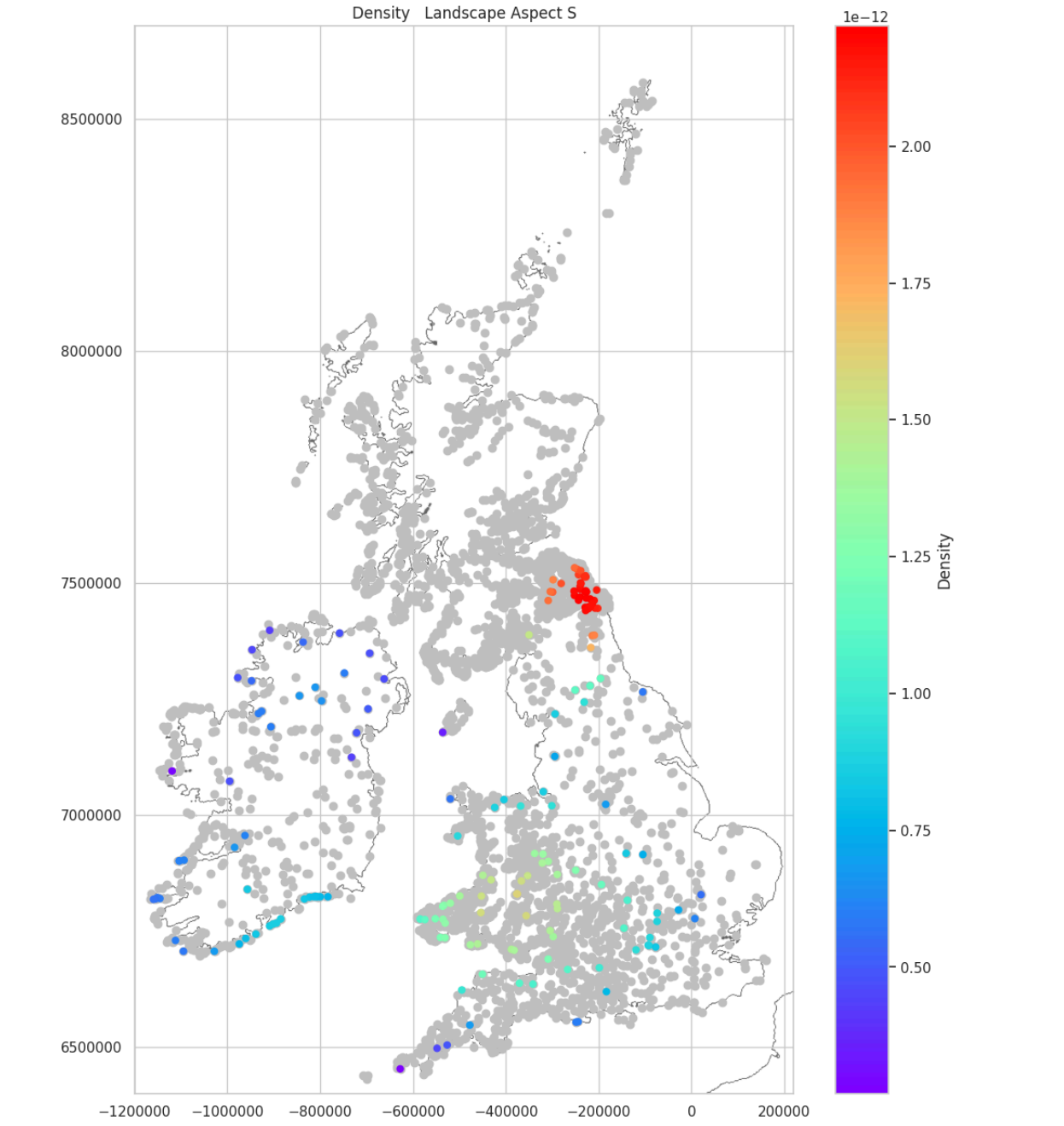


4.05%

Aspect Density South Mapped

Hillforts, with a south facing aspect, are clustered over the Southern Uplands. There is an interesting cluster of these forts along the southern coast of Ireland suggesting local topography/geology is the main influence in this area.

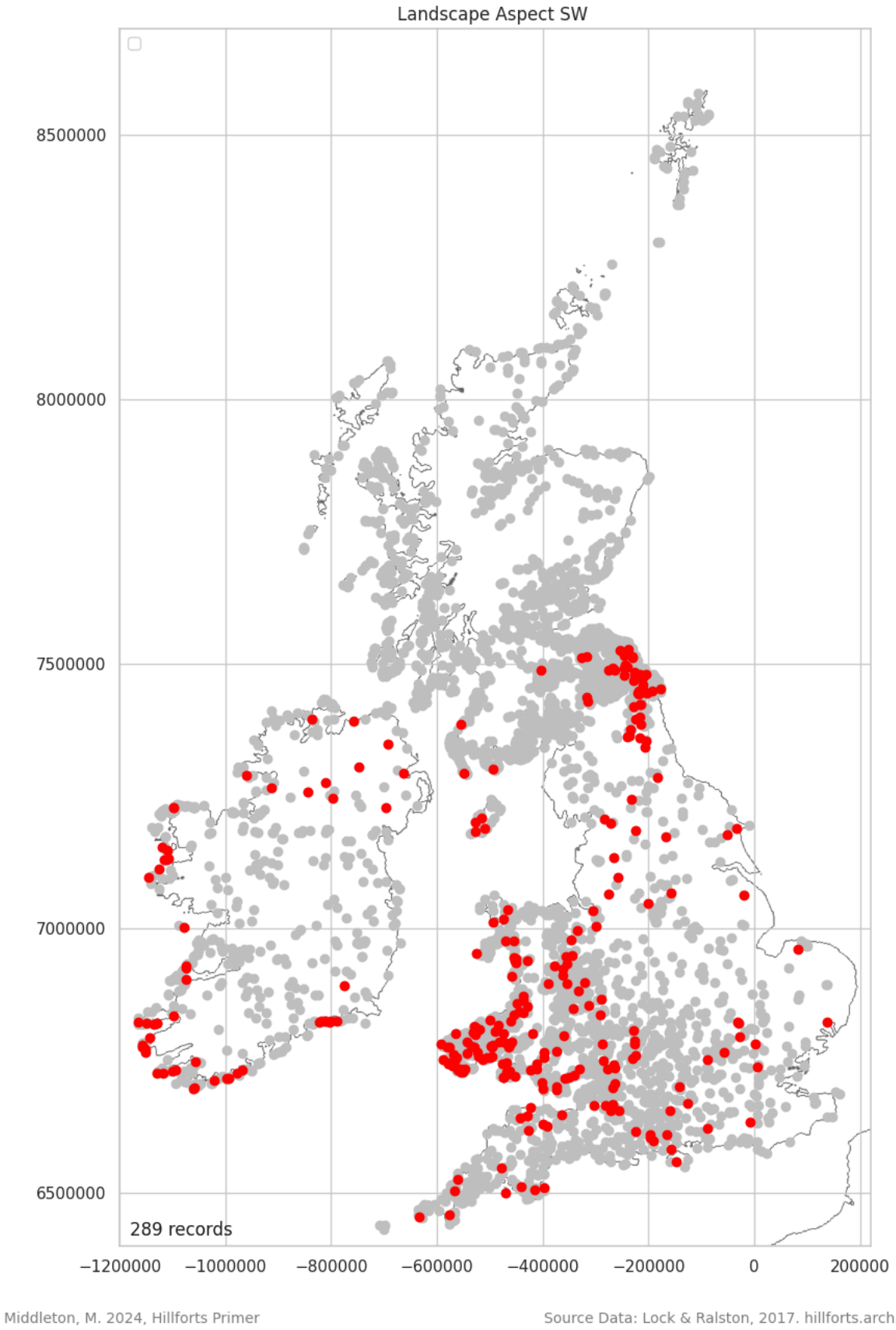
In [ ]:
```
plot_density_over_grey(asp_s, 'Landscape_Aspect_S')
```



Aspect South West Mapped

289 hillforts (6.97%) have an aspect facing south west.

In [ ]:
```
asp_sw = plot_over_grey(location_aspect_data, 'Landscape_Aspect_SW', 'Yes')
```

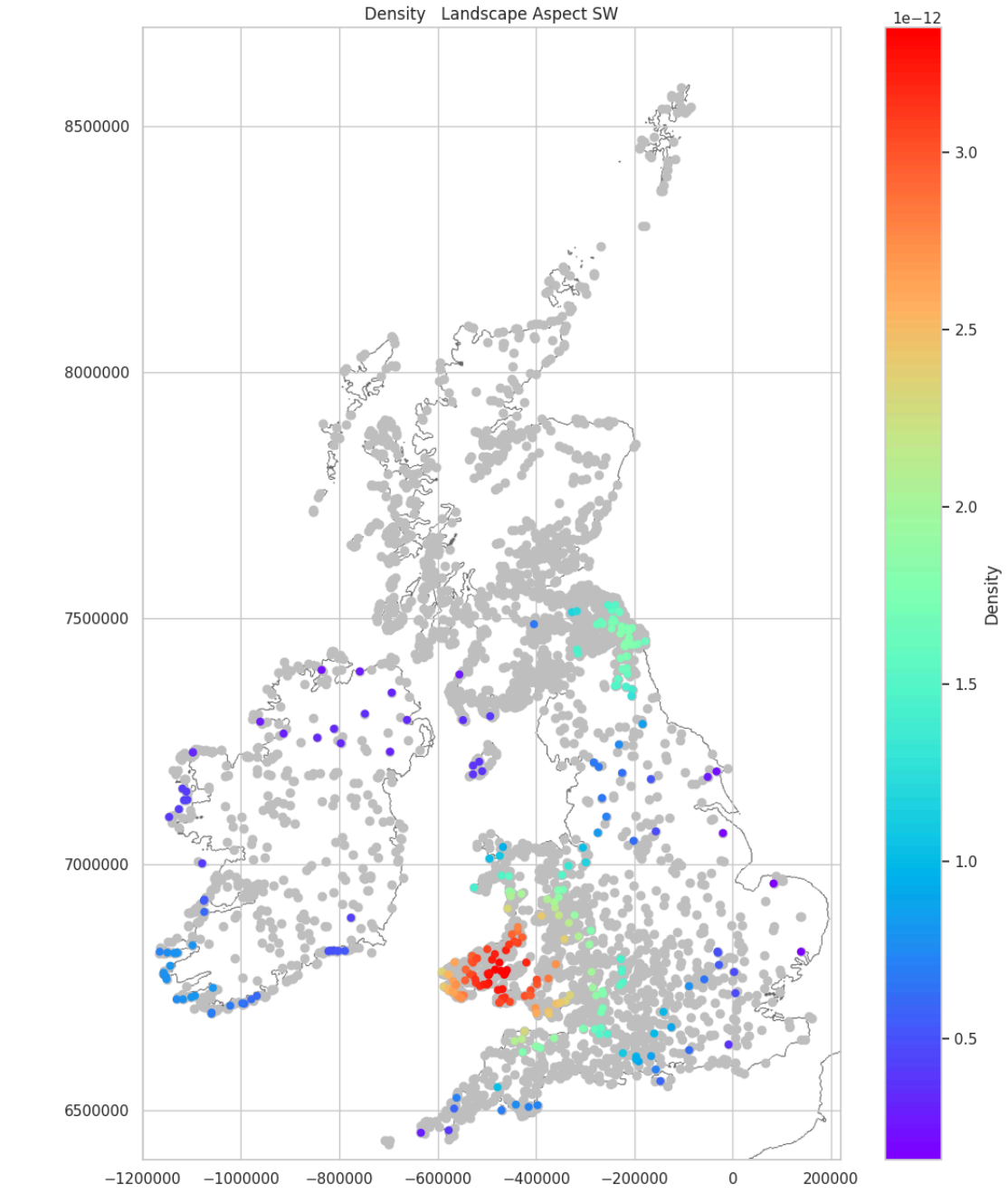


6.97%

Aspect Density South West Mapped

Hillforts with a south-western aspect are mainly clustered toward the southern end of the Cambrian Mountains and spread into the Pembrokeshire peninsula. There are small clusters over the Southern Uplands and the south-western coast of Ireland.

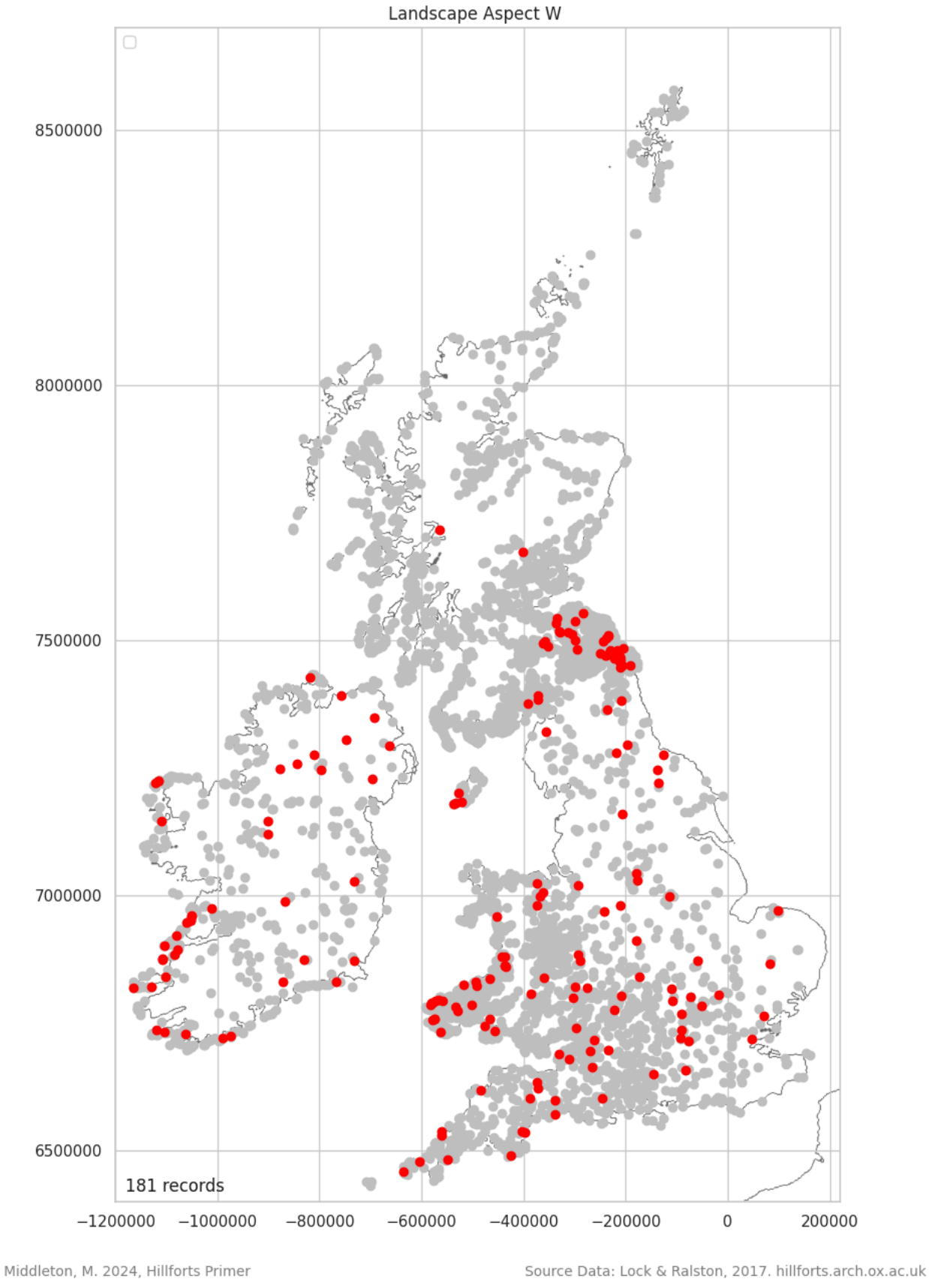
In [ ]:
```
plot_density_over_grey(asp_sw, 'Landscape_Aspect_SW')
```



Aspect West Mapped

181 hillforts (4.36%) have an aspect facing west.

In [ ]:
```
asp_w = plot_over_grey(location_aspect_data, 'Landscape_Aspect_W', 'Yes')
```

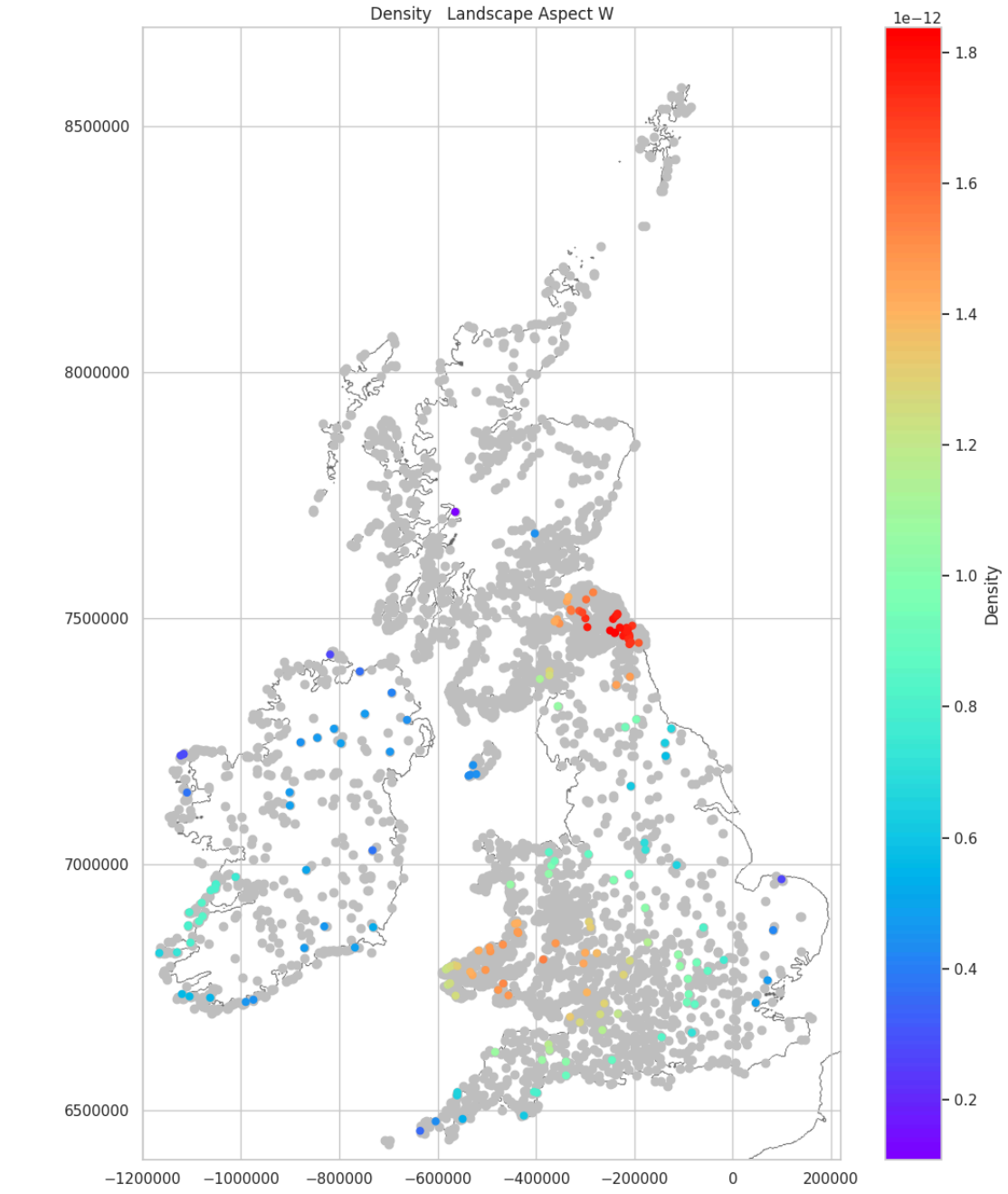


4.36%

Aspect Density West Mapped

Hillforts with a western aspect are clustered over the Tweed Basin and the Cheviots. There are also small concentrations over the southern end of the Cambrian Mountains and, along the coast, in west Ireland

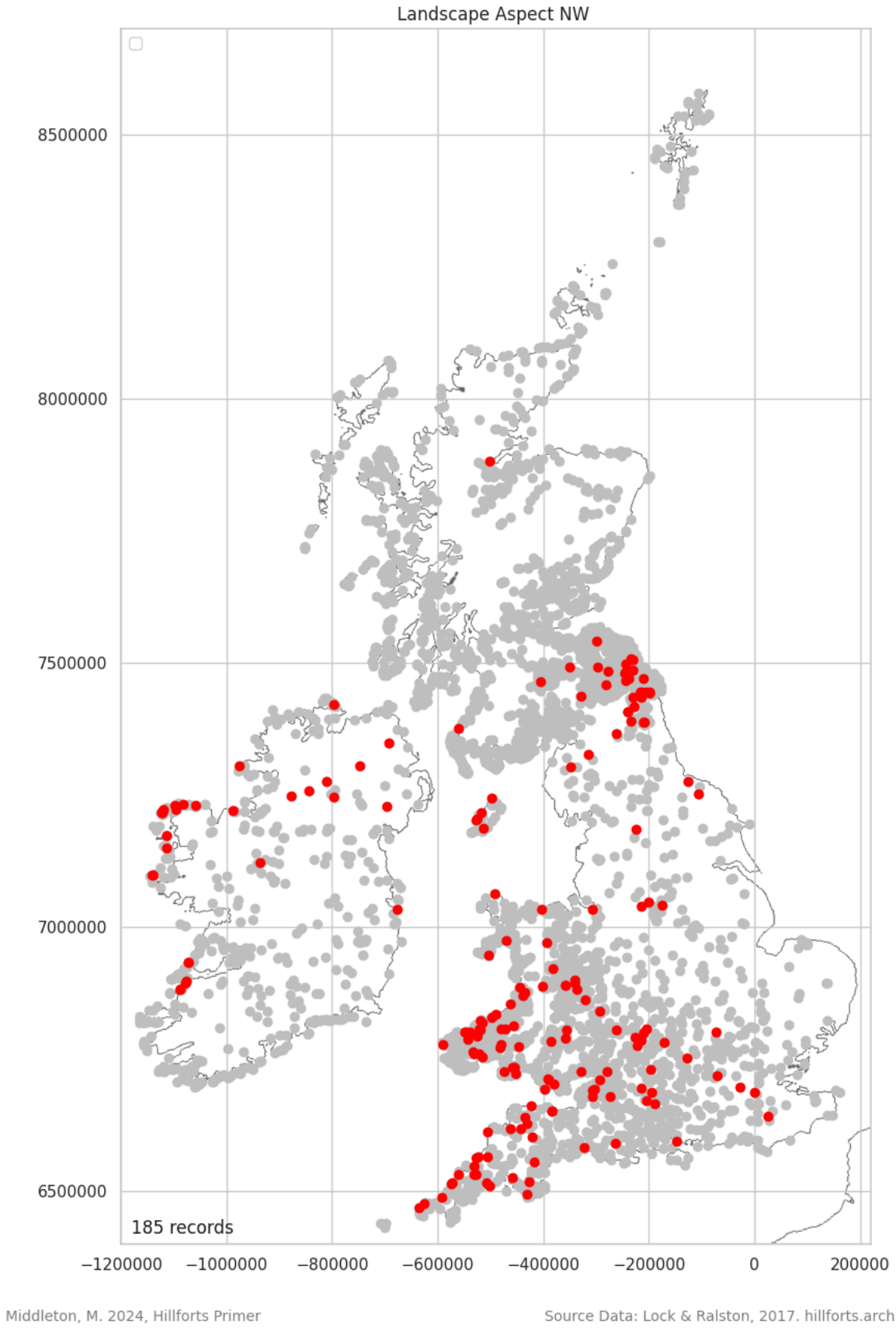
In [ ]:
```
plot_density_over_grey(asp_w, 'Landscape_Aspect_W')
```



Aspect Northwest Mapped

185 hillforts (4.46%) have an aspect facing Northwest.

In [ ]:
```
asp_nw = plot_over_grey(location_aspect_data, 'Landscape_Aspect_NW', 'Yes')
```

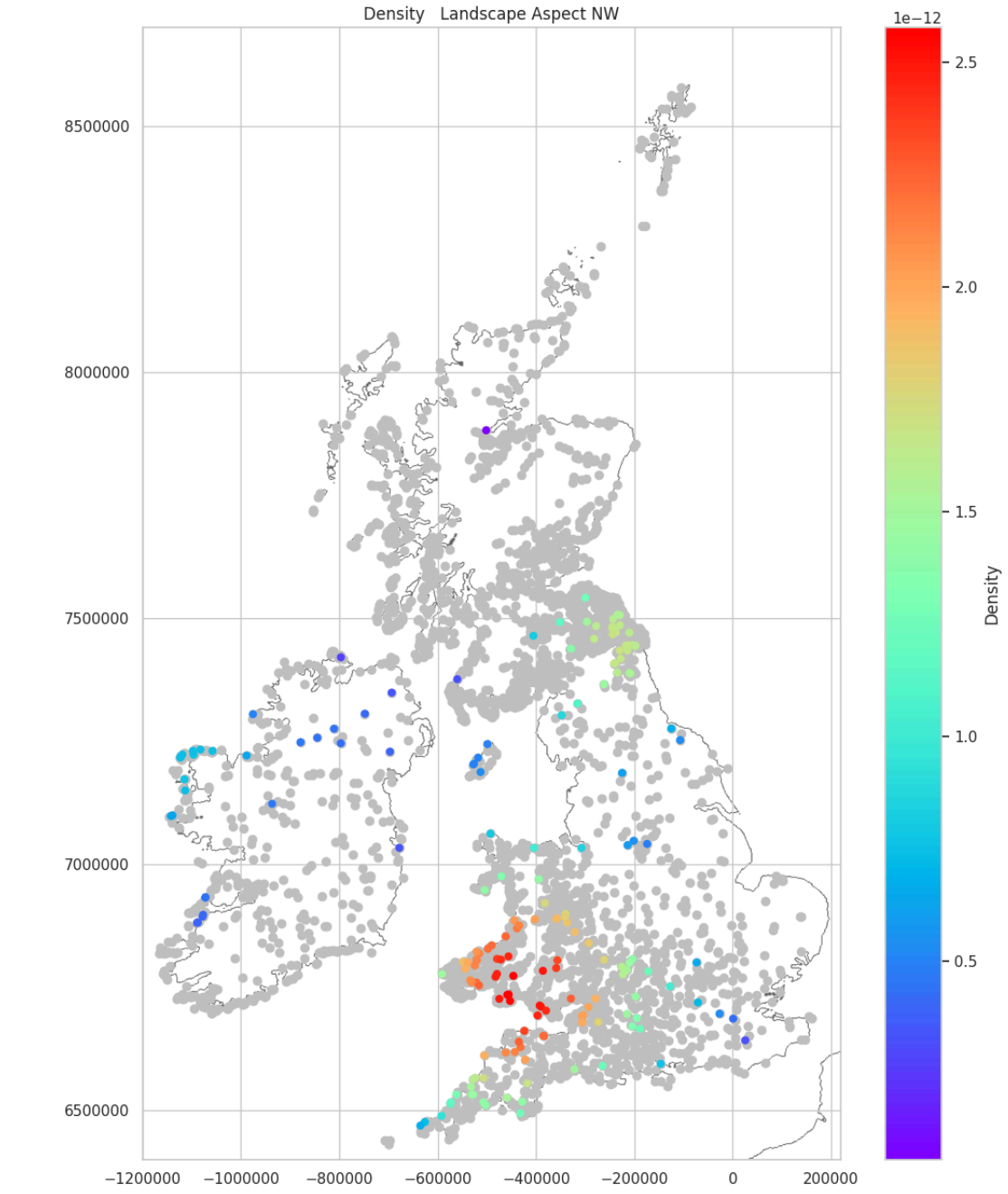


4.46%

Aspect Density Northwest Mapped

Hillforts with a Northwestern aspect are mainly clustered over the southern end of the Cambrian Mountains. There are small clusters over the Tweed Basin and the Cheviots, and along the west coast of Ireland.

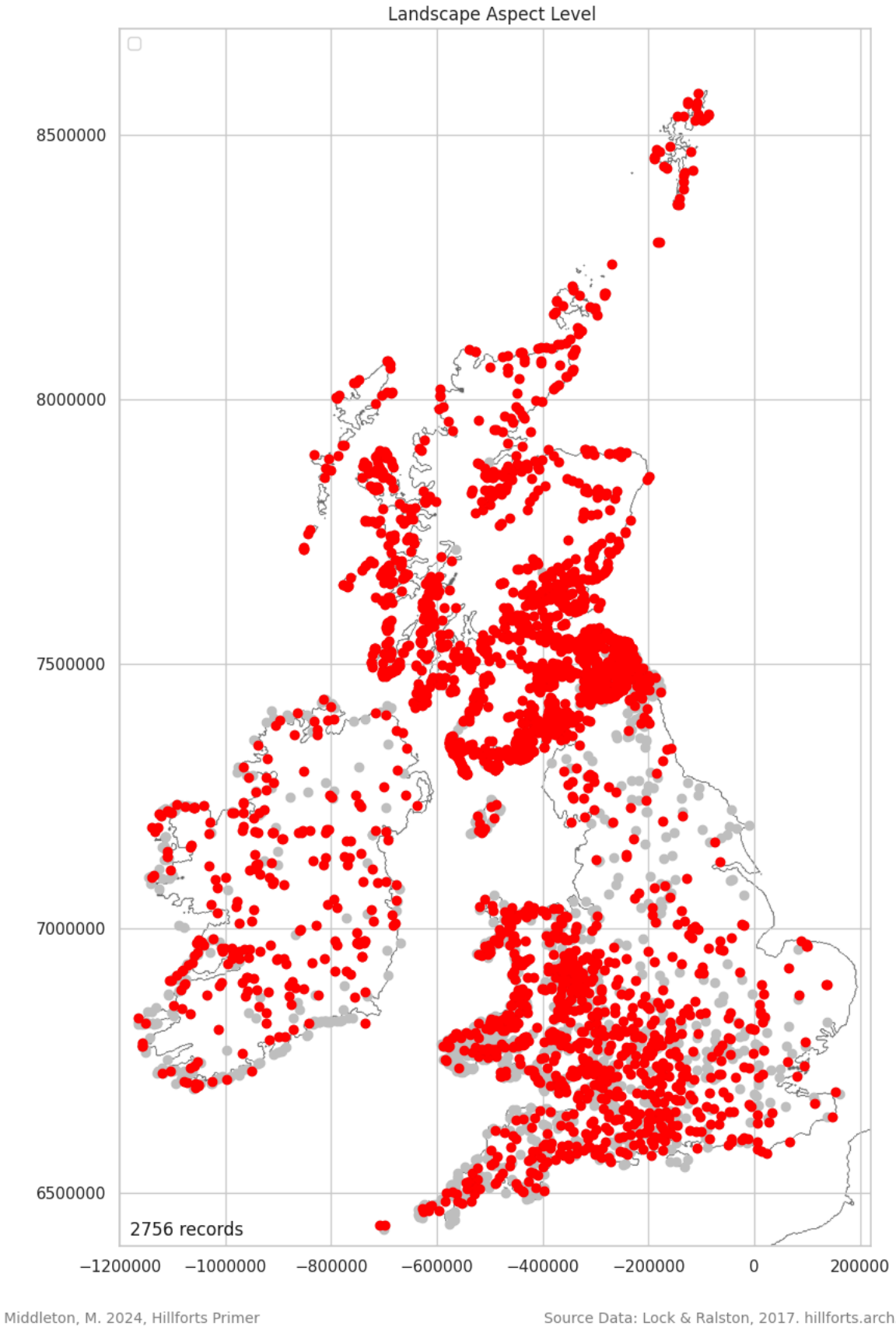
In [ ]:
```
plot_density_over_grey(asp_nw, 'Landscape_Aspect_NW')
```



Aspect Level Mapped

2756 hillforts (66.46%) are level. See: Level Mapped (Landscape).

In [ ]:
```
asp_level = plot_over_grey(location_aspect_data, 'Landscape_Aspect_Level', 'Yes')
```

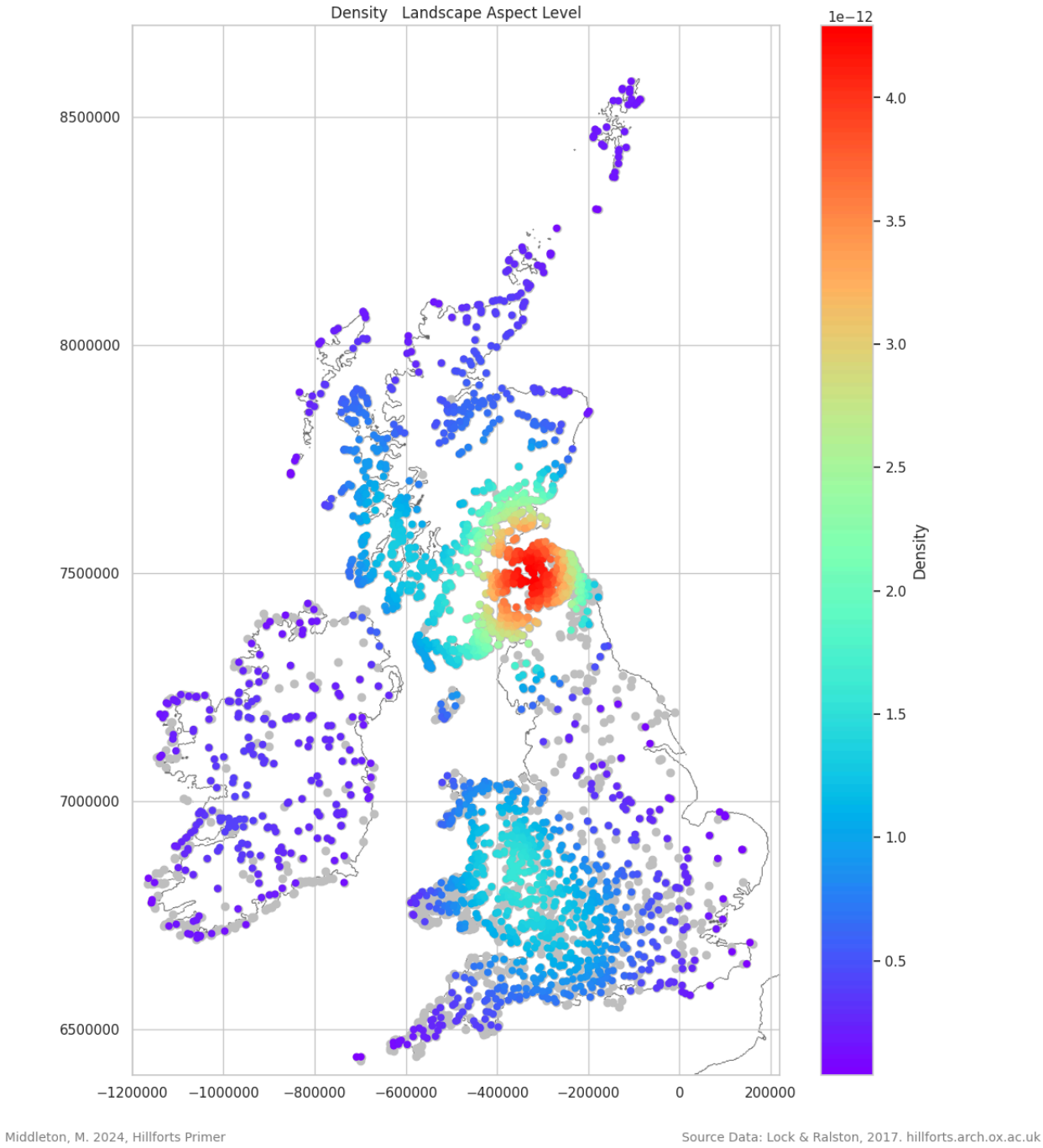


66.46%

Aspect Density Level Mapped

The three main clusters mirrors that seen in Part1 (Density Map Showing Clusters Adjusted by Region). The two smaller clusters in Ireland are not replicated.

In [ ]:
```
plot_density_over_grey(asp_level, 'Landscape_Aspect_Level')
```



Review Landscape Data Split

In [ ]:
```
review_data_split(landscape_data, landscape_numeric_data, \
                  landscape_text_data, landscape_encodeable_data)
```

```
Data split good.
```

Landscape Data Package

In [ ]:
```
landscape_data_list = [landscape_numeric_data, landscape_text_data, \
                       landscape_encodeable_data]
```

Landscape Data Download Packages

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

In [ ]:
```
download(landscape_data_list, 'landscape_package')
```

Save Figure List

In [ ]:
```
if save_images:
    path = os.path.join(IMAGES_PATH, f"fig_list_{part.lower()}.csv")
    fig_list.to_csv(path, index=False)
```

Part 3: Boundary & Dating

Colab Notebook: Live code

HTML: Read only